TOSHIBA

64-Bit TX System RISC
TX49 Family
TMPR4951B

Rev. 2.1

TOSHIBA CORPORATION

Semiconductor Company

Preface

Thank you for new or continued patronage of TOSHIBA semiconductor products. This is the 2006 edition of the user's manual for the TMPR4951B 64-bit RISC microprocessor.

This databook is written so as to be accessible to engineers who may be designing a TOSHIBA microprocessor into their products for the first time. No prior knowledge of this device is assumed. What we offer here is basic information about the microprocessor, a discussion of the application fields in which the microprocessor is utilized, and an overview of design methods. On the other hand, the more experienced designer will find complete technical specifications for this product.

Toshiba continually updates its technical information. Your comments and suggestions concerning this and other Toshiba documents are sincerely appreciated and may be utilized in subsequent editions. For updating of the data in this manual, or for additional information about the product appearing in it, please contact your nearest Toshiba office or authorized Toshiba dealer.

September 2006

Table of Contents

Handling Precautions

TMPR4951B

Handling Precautions

1. Precautions for Semiconductor Product Use

Toshiba is continually working to improve the quality and reliability of its products. Nevertheless, semiconductor devices in general can malfunction or fail due to their inherent electrical sensitivity and vulnerability to physical stress.

It is the responsibility of the buyer, when utilizing Toshiba semiconductor products, to comply with the standards of safety in making a safe design for the entire system, and to avoid situations in which a malfunction or failure of such Toshiba semiconductor products could cause loss of human life, bodily injury or damage to property.

In developing your designs, please check the most recent product specifications and ensure that the Toshiba semiconductor products are used within the operating ranges. Also, please keep in mind the precautions and conditions set forth in this Handbook.

2.   Safety Precautions

This section lists important precautions which users of semiconductor devices (and anyone else) should observe in order to avoid injury to human body and damage to property, and to ensure safe and correct use of devices. Please be sure that you understand the meanings of the labels and graphic symbols described below before you move on to the detailed descriptions of the precautions, and make every effort to observe the precautions stated.

[Explanation of Labels]

Label	Meaning
⚠ DANGER	Indicates an imminently hazardous situation which will result in death or serious injury[1] if you do not followinstructions.
⚠ WARNING	Indicates a potentially hazardous situation which could result in death or serious injury[1] if you do not followinstructions.
⚠ CAUTION	Indicates a potentially hazardous situation which if not avoided, may result in minor injury[2], moderate injury[2], or property damage[3].

1. Serious injury includes blindness, wounds, burns (low and high temperature), electric shock, fractures, and poisoning, etc. with long-lasting effects or that require hospitalization and/or long-term hospital visits for treatment.
2. Injury includes wounds, burns, electric shock, etc. not requiring hospitalization and/or long-term hospital visits for treatment.
3. Damage includes extensive damage to machines and equipment.

[Explanation of Graphic Symbols]

Graphic Symbol	Meaning
Prohibited	Indicates prohibited (restricted) actions. Prohibited actions are explained in or near the symbols in pictures and sentences.
Instructions	Indicates compulsory (mandatory) actions. Compulsory actions are explained in or near the symbols in pictures and sentences.

⚠ Caution	Indicates cautions. Cautions are explained in or near the symbols in pictures and sentences.
⚠ Caution	Example of a caution symbol: Indicates laser beam caution.

2.1   General Precautions Regarding Semiconductor Products

⚠CAUTION

Prohibited	The absolute maximum ratings of a semiconductor device are a set of ratings that must not be exceeded, even for a moment. Do not exceed any of these ratings. Exceeding the rating(s) may cause the device breakdown, damage or deterioration, and may result injury by explosion or combustion.
Prohibited	Do not insert devices in the wrong orientation or incorrectly. Make sure that the positive and negative terminals of power supplies are connected properly. Otherwise, the current or power consumption may exceed the absolute maximum rating, and exceeding the rating(s) may cause the device breakdown, damage or deterioration, and may result injury by explosion or combustion. In addition, do not use any device that is applied the current with inserting in the wrong orientation or incorrectly even just one time.
Prohibited	Do not touch the heat sink of the device while the device is on or immediately after the device has been turned off. Heat sinks become hot. Contact to the heat sink may result in a burn.
Prohibited	Do not touch the lead tips of a device. Some devices have leads withsharp tips. Contact to sharp tips may result in a puncture wound.
Instructions	On the evaluation, inspection or test, be sure to connect the test equipment's electrodes or probes to the pins of the device before turning the power on. When you have finished, discharge any electrical charge remaining in the device. Insufficient connection, wrong power-on timing or incomplete discharge may cause electric shock, resulting in injury.
Instructions	Check that there is no electrical leakage before grounding measuring equipment or a solder iron. Electrical leakage may cause the device you are testing or soldering to electrically break down or may cause electric shock.
Instructions	Always wear safety glasses when cutting the leads of a device with clippers or a similar tool. Failure to do so may result in eye damage from the small shavings that fly off the cut ends.

2.2 Precautions Specific to Product Group

2.2.1 Optical Semiconductor Devices

<table>
<tr>
<td colspan="2" align="center">⚠ **DANGER**</td>
</tr>
<tr>
<td align="center">🚫
Prohibited</td>
<td>When a semiconductor laser is operating, do not look into the laser beam or look through the optical system.
Doing so may damage your eyesight or, in the worst case, cause blindness. When inspecting the optical characteristics of the laser using laser protective glasses, be sure the glasses comply with JISC6802.</td>
</tr>
<tr>
<td colspan="2" align="center">⚠ **WARNING**</td>
</tr>
<tr>
<td align="center">🚫
Prohibited</td>
<td>Do notapply voltage or current into the LED device that exceeds the device's absolute maximum rating.
With resin-packaged LED devices in particular, excessive voltage or current may cause the package resin to explode, scattering resin fragments and may result injury.</td>
</tr>
<tr>
<td align="center">❗
Instructions</td>
<td>When evaluating and testing the dielectric strength voltage of a photocoupler, use equipment that can shut off the supply voltage if a leakage current exceeding $100\mu A$ is detected.
Failure to do so may result in the continuous flow of a large short-circuit current, causing the device to explode or combust, resulting in fire or injury.</td>
</tr>
<tr>
<td align="center">❗
Instructions</td>
<td>When designing a semiconductor laser, use the built-in or another light-receiving element to stabilize optical output so as to ensure that laser beams exceeding the laser's rated optical output are not emitted.
If this stabilization function does not work properly and the rated output is exceeded, not only the device may break down but also injury may result by the laser beam.</td>
</tr>
</table>

2.2.2 Power Device

<table>
<tr>
<td colspan="2" align="center">⚠ **DANGER**</td>
</tr>
<tr>
<td align="center">🚫
Prohibited</td>
<td>Do not touch a power device while it is on or after it has been turned off and all remaining electrical charge has not yet been discharged.
Touching power element with electrical charge may cause electric shock, resulting in death or serious injury.</td>
</tr>
<tr>
<td align="center">❗
Instructions</td>
<td>When evaluating, inspecting or testing a device, be sure to connect all of the test equipment's electrodes or probes before turning the power on. When you have finished, discharge any electrical charge remaining in the device.
Connecting the electrodes or probes with the power on may cause electric shock, resulting in death or serious injury.</td>
</tr>
</table>

	⚠WARNING	
🚫 Prohibited	Do not use a device under conditions that exceed its absolute maximum ratings (such as current, voltage, safe operation range, temperature.). Using device with exceeded its absolute maximum ratings may cause the device to break down, causing a short-circuit current, which may in turn cause it to explode or combust, resulting in fire or injury.	
❗ Instructions	Use a unit which can detect short-circuit currents and shut off the power supply if a short-circuit occurs. If the power supply is not shut off, a large short-circuit current will flow continuously, causing the device to explode or combust, resulting in fire or injury.	
❗ Instructions	Design the case for enclosing your system taking into consideration the prevention of flying debris in the event the device explodes or combusts. Flying debris may cause injury.	
❗ Instructions	When conducting an evaluation, inspection or test, use protective safety tools such as device covers. The device may explode or combust due to excessive stress in the event of breakdown or arc discharge between the electrode and ground potential, resulting in fire or injury.	
❗ Instructions	Design your product so that it is used with all metal areas (other than electrodes and terminals) grounded. Even with products where the device's electrodes and metal casings are insulated, electrostatic capacitance in the product may cause the electrostatic potential in the casing to rise. Insulation deterioration or breakdown may cause a high voltage to be applied to the casing, causing electric shock when touched, resulting in death or serious injury.	
❗ Instructions	When designing the heat radiation and safety features of a system incorporating Schottky barrier diodes and high-speed rectifiers, take into consideration the device's forward and reverse losses. The reverse current in these devices is greater than that in ordinary rectifiers. If the operating environment is severe (such as high temperature, high voltage), the device's reverse loss may increase, causing a short-circuit current and subsequently explosion or combustion, resulting in fire or injury.	
❗ Instructions	Be sure to design the product so that, except when the main circuits of the device are active, the main circuits will be inactive when electricity is conducted to control circuits. Malfunction of the device may cause a serious accident or injury.	
	⚠CAUTION	
❗ Instructions	When conducting an evaluation, inspection or test, either wear protective gloves or wait until the device cools down prior to handling. Devices become hot when operated. Even after the power supply has been turned off, residual heat may cause a burn when the device is touched.	

2.2.3   Application Specific Standard Products and General-Purpose Linear ICs

⚠CAUTION

Instructions	Use an appropriate power supply fuse to ensure that a large current does not continuously flow in case of over current and/or IC failure. The IC will fully break down when used under conditions that exceed its absolute maximum ratings, when the wiring is routed improperly or when an abnormal pulse noise occurs from the wiring or load, causing a large current to continuously flow and the breakdown can lead smoke or ignition. To minimize the effects of the flow of a large current in case of breakdown, appropriate settings, such as fuse capacity, fusing time and insertion circuit location, are required.
Instructions	If your design includes an inductive load such as a motor coil, incorporate a protection circuit into the design to prevent device malfunction or breakdown caused by the current resulting from the inrush current at power ON or the negative current resulting from the back electromotive force at power OFF. For details on how to connect a protection circuit such as a current limiting resistor or back electromotive force adsorption diode, refer to individual IC datasheets or the IC databook. IC breakdown may cause injury, smoke or ignition.
Instructions	Use a stable power supply with ICs with built-in protection functions. If the power supply is unstable, the protection function may not operate, causing IC breakdown. IC breakdown may cause injury, smoke or ignition.
Instructions	Carefully select external components (such as inputs and negative feedback capacitors) and load components (such as speakers), for example, power amp and regulator. If there is a large amount of leakage current such as input or negative feedback condenser, the IC output DC voltage will increase. If this output voltage is connected to a speaker with low input withstand voltage, over current or IC failure can cause smoke or ignition. (The over current can cause smoke or ignition from the IC itself.) In particular, please pay attention when using a Bridge Tied Load (BTL) connection type IC that inputs output DC voltage to a speaker directly.

2.2.4　Memory Card Products

	WARNING
⊘ Prohibited	Keep out of reach of small children. Accidental swallowing may cause suffocation or injury. Contact a doctor immediately if you suspect a child has swallowed the Product.

	CAUTION
⊘ Prohibited	Do not directly touch the interface pins, put them in contact with metal, strike them with hard objects, or cause them to short. Do not expose to static electricity.
⊘ Prohibited	Do not bend, apply strong force to, drop, expose to strong impact or lay heavy objects on top of the Product.
⊘ Prohibited	Do not put the Product in the back pocket, etc. of trousers. It may break when you sit down or exert strong force on it in other ways.
⊘ Prohibited	Do not disassemble or modify the Product. This may cause electric shock, damage to the Product, or fire.
⊘ Prohibited	Do not expose the Product to moisture. Do not use, store, or place in humid locations or expose to water. Do not expose the Product to excessive heat or cold. Do not use, store, or place in direct sunlight, inside a hot car, near fire or sources of heat or flame, such as a stove. Do not use, store, or place the Product near an air-conditioner outlet. Do not expose the Product to dust, strong magnetic fields, or corrosive chemicals or gas.
⊘ Prohibited	Avoid sudden temperature changes which could cause condensation.

⚠CAUTION

Prohibited	(Applicable only to mini SD) While writing data to or reading data from the Product, do not turn off the power, remove the Product or the miniSD adapter from the device, or permit the Product or device to be shaken or impacted.
Prohibited	(Applicable only to mini SD) Do not insert, remove or change the Product while the miniSD adapter is still inserted into a device. This may cause product failure or the destruction or loss of data.
Prohibited	(Applicable only to mini SD) Do not insert into a device the miniSD adapter that does not contain the Product. This may cause improper function on the device.
Prohibited	(Applicable only to mini SD) Do not insert into the miniSD adapter a memory card other than the Product or other foreign object. This may cause product failure.
Prohibited	While writing data to or reading data from the Product, do not turn off the power, remove the Product from the device, or permit the Product or device to be shaken or impacted.
Prohibited	(Applicable only to USB flash memory) Performing the following operations with the Product connected to your PC may cause improper function on your PC. - Booting - Rebooting - Resuming from standby or suspended mode Please perform these operations after removing the Product from your computer.
Instructions	The Product comes pre-formatted. When formatting, all data stored in the Product will be lost, so make sure to back up the data on the Product. Please use functions relevant to the Product. (Applicable to SD, mini SD and compact flash) Formatting with other devices, for example PCs, may cause problems such as the inability to read, write, or delete data.
Instructions	(Applicable only to USB flash memory) Please take note of the following when formatting the Product using Windows XP or Windows 2000. · Log in as a user with administrator access rights. The Product cannot be formatted while logged in as a user with limited access rights. · Format the Product as FAT. It may not format properly as NTFS or FAT32.
Instructions	When removing the Product from the slot or the USB port of a PC, first follow the removal (stop) procedure for your operating system.

⚠CAUTION

Instructions	Refer to your device's manual to learn how to insert and remove the Product.
Instructions	Insert the Product firmly in the correct orientation. The Product will not operate correctly if it is inserted in an incorrect orientation or not inserted all the way.
Instructions	Always use the Product with the interface pins and connector in a clean state. When cleaning the Product, use a soft, dry cloth.
Instructions	(Applicable only to miniSD) Always use the miniSD adapter when using the Product with a standard SD memory card device. Directly inserting the Product into a standard SD memory card device may cause improper function on the device.

3. General Safety Precautions and Usage Considerations

This section provides information that will help you gain a better understanding of semiconductor devices so as to ensure device safety, quality and reliability.

3.1 From Incoming to Shipping

3.1.1 Electrostatic Discharge (ESD)

When handling individual devices, be sure that the environment is protected against static electricity. Operators should wear anti-static clothing. In addition, containers and other objects that come in direct contact with devices should be made of materials that do not produce static electricity that would cause damage. Please follow the precautions below. This is particularly important for those devices marked "Be careful of static."

3.1.1.1 Work Environment Control

(1) When humidity decreases, static electricity readily occurs due to friction. Taking into consideration the fact that moisture-proof-packed products absorb moisture after unpacking, the recommended humidity is 40 to 60%.

(2) Be sure that all equipment such as jigs and tools installed in the work area are grounded.

(3) Place a conductive mat over the floor of the work area or take other measure to ensure that the floor is protected against static electricity and is grounded to the earth. (Resistance between surface and ground: 1 x $10^9\Omega$ or less)

(4) Place a conductive mat over the surface of worktables to ensure that the tables are grounded. (Resistance between surface and ground: 7.5 x 10^5 to 1 x $10^9\Omega$) Do not construct worktable surfaces of metallic materials. Metallic materials are low in resistance, allowing rapid discharge when a charged device comes in contact with them directly.

(5) Observe the following when using automated equipment:

(a) When picking up a device with a vacuum, use conductive rubber in sections which come into contact with the device surface to protect against electrostatic charge.

(b) Avoid friction on the device surface to the extent possible. If rubbing is unavoidable due to the device's mechanical structure, minimize the friction plane or use material with a small friction coefficient and low electrical resistance. Also, prevent electrostatic charge by using an ionizer.

(c) Use a material which dissipates static electricity in sections which come in contact with device leads or terminals.

(d) Ensure that no statically charged bodies (such as work clothes or the human body) come in contact with the devices.

(e) Use a tape carrier that employs a low-resistance material on sections that come in contact with

electrical machinery.

(f) Make sure that jigs and tools used in the manufacturing process do not touch the devices.

(g) In processing associated with package electrostatic charge, use an ionizer to neutralize the ions in the ambient environment.

(6) Make sure that CRT displays in the work area are protected against static charge by employing a filter, for example. Avoid turning displays on and off to the extent possible. Neglecting to do so can cause electrostatic induction in devices.

(7) Periodically measure the charged potential of devices, systems and fixtures located in the work area to ensure that the area is free of any charge.

(8) Ensure that the work chairs are protected by a conductive cover and grounded to the floor by conductive castors. (Resistance between seat surface and ground: $1 \times 10^{10}\Omega$ or less)

(9) Install anti-static mats on storage shelf surfaces and ground the mat surface. (Resistance between surface and ground: 7.5×10^5 to $1 \times 10^9\Omega$)

(10) For device transport and temporary storage, use containers (boxes, jigs or bags) that are made of a material which does not produce static electricity that could damage the device.

(11) Make sure that cart surfaces which come in contact with product packaging are made of materials which conduct static electricity, and ground the cart surfaces to the floor surface using conductive castors.

(12) In static electricity control areas, install anti-static dedicated ground wires. Use a transmission line circuit ground wire [Type D (previous Class C) or above], or a trunk line ground wire. In addition, separate and ground the various devices individually.

3.1.1.2 Work Environment Control

(1) Operators must wear anti-static clothing and conductive shoes (or a toe or heal strap).

(2) Operators must wear a wrist strap grounded to earth via a resistor.
(Resistance between surface and earth when worn: 7.5×10^5 to $3.5 \times 10^7\Omega$)

(3) Soldering irons must be grounded from the iron tip to earth, and must be used at low voltages (6 to 24V).

(4) If the tweezers you use are likely to touch the device terminals, use anti-static tweezers. Do not use metallic tweezers since they are low in resistance and may cause rapid discharge when a charged device comes in contact with them.
When using a vacuum tweezers, attach a conductive chucking pat to the tip, and connect it to a dedicated anti-static ground. In addition, follow the manufacturer's methods of use and maintenance.

(5) Do not place devices or their containers near sources of strong electrical fields (such as above a CRT).

(6) Place boards with mounted devices in anti-static board containers separated from one another, and do not stack them directly on top of one another. Stacking them directly on top of one another may cause frictional charge or discharge.

(7) Ensure, to the extent possible, that any articles (such as clipboards) which are brought to a static electricity control area are constructed of anti-static materials.

(8) When the human body is to come in direct contact with a device, wear anti-static finger covers or gloves.

(9) The material of equipment safety covers located near devices should have a resistance rating of $1 \times 10^{9}\Omega$ or less.

(10) If a wrist strap cannot be used and there is a possibility of imparting friction to devices, use an ionizer.

(11) The transport film used in tape carrier products is manufactured from materials in which static electricity readily builds up. When using these products, use an ionizer to prevent the film from being charged. Also, to ensure than no static electricity will be applied to the copper foil area, take measures to prevent electrostatic discharge failure of peripheral equipment.

3.1.2   Vibration, Impact and Stress

Handle devices and packaging with care. Dropping or applying impact to devices or packaging causes device damage. Ensure that devices and packaging are not subjected to mechanical vibration or impact to the extent possible. Hollow canister-type devices and ceramic sealed devices contain unsecured wires, making them more susceptible to vibration and impact than plastic sealed devices.
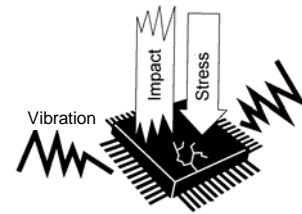
When a location such as a soldered area, connecting area or top surface of a device is subjected to vibration, impact or stress in actual equipment, bonding fault or device destruction may result. Therefore, be sure to keep this in mind at the time of structural design. If a device is subject to especially strong vibration, impact or stress, the package or chip may crack. If stress is applied to a semiconductor chip through the package, changes in the resistance of the chip may result due to piezoelectric effects, resulting in fluctuation in element characteristics. Furthermore, if a stress that does not instantly result in damage is applied continually for a long period of time, product deformation may result, causing defects such as disconnection or element failure. Thus, at the time of structural design, carefully consider vibration, impact and stress.

3.2   Storage

3.2.1   General Packaged Products

(1) Avoid storage locations where devices may be exposed to moisture or direct sunlight.

(2) Follow the precautions printed on the packing label of the device for transportation and storage.

(3) Keep the storage location temperature and humidity within a range of 5°C to 35°C and 45% to 75%, respectively.
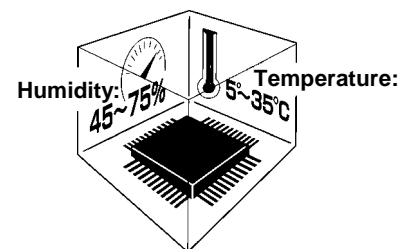
(4) Do not store the products in locations with poisonous gases (especially corrosive gases) or in dusty conditions.

(5) Store the products in locations with minimal temperature fluctuations. Rapid temperature changes during storage can cause condensation, resulting in lead oxidation or corrosion, which will deteriorate the solderability of the leads.

(6) When restoring devices after removal from their packing, use anti-static containers.

(7) Do not allow loads to be applied directly to devices while they are in storage.

(8) If devices have been stored for more than two years under normal storage conditions, it is recommended that you check the leads for ease of soldering prior to use.

3.2.2   Moisture-Proof Packing

Moisture-proof packing should be used while taking into careful consideration the handling methods specified for each packing type. If the specified procedures are not followed, the quality and reliability of the devices may be deteriorated. This section describes the general precautions for handling moisture-proof packing. Since the details may differ from device to device, refer to the individual standards or databooks during handling.

3.2.2.1   Moisture-Proof Packing General Precautions

Follow the precautions printed on the packing label of the device for transportation and storage.

For chip products, follow the individual specifications.

(1) Do not toss or drop device packing. The aluminum laminated bag may be damaged, resulting in a loss in airtightness.

(2) Keep the storage environment at 5°C to 30°C, and the relative humidity at 90% or less. Use devices within 12 months of the date marked on the package seal.

(3) If the 12-month storage period has been exceeded, or if the 30% humidity indicator is pink when the package is opened, remove any moisture under the conditions described in the table below. The effective usage period without moisture removal after the packing has been opened and the product has been stored at 5°C to 30°C and a relative humidity of 60% is listed on the moisture-proof package. If the effective usage period has been exceeded, or if the packing has been stored in a high-humidity environment or an environment that produces condensation, remove any existing moisture.

Packing Type	Moisture Removal Applicability/Procedure
Tray	If the packing indicates a "Heatproof" or temperature label, bake at 125°C for 20 hours. (Some devices may require a different amount of time.) If the packing does not indicate a "Heatproof" or temperature label, transfer the devices to an anti-static container that bears a "Heatproof" or temperature label and then bake. The moisture-proof package itself is not heat resistance. Be sure to remove the devices from the package prior to baking.
Magazine	Transfer devices to antistatic containers bearing the "Heatproof" or temperature label, and then bake at 125°C for 20 hours. (Some devices may require a different amount of time.) The moisture-proof package itself is not heat resistance. Be sure to remove the devices from the package prior to baking.
Tape	Transfer devices to antistatic containers bearing the "Heatproof" or temperature label, and then bake at 125°C for 20 hours. (Some devices may require a different amount of time.) The moisture-proof package itself is not heat resistance. Be sure to remove the devices from the package prior to baking.

(4)   When removing the moisture from the devices, protect the devices from breakdown from static electricity.

(5)   Moisture indicators (for your reference)

Moisture indicators detect the approximate ambient humidity level at a standard temperature of 25°C.

Figure 3.1 shows a 3-point indicator.

HUMIDITY INDICATOR

40

30

DANGER IF PINK

20

READ AT LAVENDER
BETWEEN PINK & BLUE
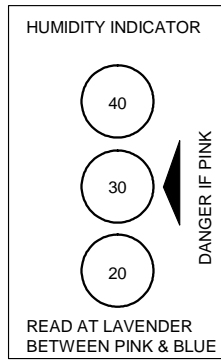
3-point indicator

Figure 3.1 Humidity Indicator

(6)   Do not allow loads to be applied directly to devices while they are in storage.

3.3   Design

To achieve the reliability required by an electronic device or system, it is important not only to use the semiconductor device in accordance with specified absolute maximum ratings and operating ranges, but also to consider the environment in which the equipment will be used, including factors such as the ambient temperature, transient noise and current surges, as well as mounting conditions which affect semiconductor device reliability. This section describes general design precautions. Be sure to refer to the individual ratings of each product at the time of design.

3.3.1   Absolute Maximum Ratings

⚠CAUTION   The absolute maximum ratings of a semiconductor device are a set of ratings that must not be exceeded, even for a moment. Do not exceed any of these ratings.

Exceeding the rating(s) may cause device breakdown, damage or deterioration, and may result injury by explosion or combustion.

If the voltage or current on any pin exceeds the absolute maximum rating, the overvoltage or overcurrent causes the device's internal circuitry to deteriorate. In extreme cases, heat generated in internal circuitry can fuse wiring or cause the semiconductor chip to break down. If the storage or operating temperature exceeds the absolute maximum rating, the device internal circuitry may deteriorate and the bonded areas may open or the package airtightness may deteriorate due to the differences between the thermal expansion coefficients of the materials from which the device is constructed.

Although absolute maximum ratings differ from product to product, they essentially concern the voltage and current at each pin, the allowable power dissipation, the connecting area temperatures, and storage temperatures. Note that the term "maximum rating" which appears in semiconductor technical datasheets and the like refers to "absolute maximum rating."
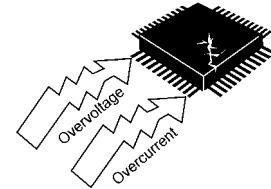
3.3.2   Operating Range

The operating range is the range of conditions necessary for the device to operate as specified in individual technical datasheets and databooks. Care must be exercised in the design of the equipment. If a device is used under conditions that do not exceed absolute maximum ratings but exceed the operating range, the specifications related to device operation and electrical characteristics may not be met, resulting in a decrease in reliability.

If greater reliability is required, derate the device's operating ranges for voltage, current, power and temperature before use.

3.3.3   Derating

The term "derating" refers to ensuring greater device reliability by setting operating ranges reduced from rated values and taking into consideration factors such as current surges and noise.

While derating generally applies to electrical stresses such as voltage, current and power, and environmental stresses such as ambient temperature and humidity, it differs from application to application. Refer to the individual technical datasheets available for each product. Power devices in particular require heat sink consideration as well since the level of derating greatly affects reliability.

For your reference, details are provided in the appendix. Be sure to read the appendix carefully.

3.3.4   Unused Pins

If unused pins are left open, some devices exhibit input instability, resulting in faulty operation such as a sudden increase in current consumption. In addition, if unused output pins on a device are connected to the power supply, GND or other output pin, the IC may malfunction or break down.

Since the treatment of unused input and output pins differs for each product and pin, please follow the directions in the individual technical datasheets and databooks.

CMOS logic IC inputs, for example, have extremely high impedance. If an input pin is left open, it can readily pick up noise and become unstable. In this case, if the input reaches an intermediate level, both the P-channel and N-channel transistors will become conductive, allowing unnecessary power supply current to flow. It is therefore necessary to ensure that the unused input gates of a device are connected to the power supply pin or ground (GND) pin of the same device. For treatment of heat sink pins, refer to the individual technical datasheets and databooks.

3.3.5   Latch-up

Semiconductor devices sometimes transition to an inherent condition referred to as "latch-up." This condition mainly occurs in CMOS devices. This happens when a parasitic PN-PN junction (thyristor structure) built in the device itself is turned on, causing a large current to flow between the power supply voltage and GND, eventually causing the device to break down.

Latch-up occurs when the voltage impressed on an input or output pin exceeds the rated value, causing a large current to flow in the internal element, or when the voltage impressed on the power supply voltage pin exceeds its rated value, forcing the internal element to breakdown. Once the element falls into the latch-up state, even though the excess voltage may have been applied only for an instant, the large current continues to flow between the power supply voltage and GND, potentially causing device explosion or combustion. To avoid this problem, observe the following:

(1) Do not allow the voltage levels on the input and output pins to rise above the power supply voltage or decrease below GND. Consider the timing during power supply activation as well.

(2) Do not allow any abnormal noises to be applied to the device.

(3) Set the electrical potential of unused input pins to the power supply voltage or GND.

(4) Do not create an output short.

3.3.6   Input/Output Protection

Wired-AND configurations in which outputs are connected together directly cannot be used since the outputs short-circuit with the configurations. Outputs should, of course, never be connected to the power supply voltage or GND. In addition, products with tri-state outputs can undergo IC deterioration if a shorted output current continues for a long period of time. Design the circuit so that the tri-state outputs will not be enabled simultaneously.

3.3.7   Load Capacitance

Certain devices exhibit an increase in delay times and a large charging and discharging current if a large load capacitance is connected, resulting in noise. In addition, since outputs are shorted for a long period of time, wiring can become fused. Use the load capacitance recommended for each product.

3.3.8   Thermal Design

The failure rate of semiconductor devices largely increases as the operating temperatures increase. As shown in Figure 3.2, the thermal stress applied to device internal circuitry is the sum of the ambient temperature and the temperature rise caused by the power consumption of the device. For thermal design, therefore, refer to the precautions stated in individual technical datasheets and databooks.

To achieve even higher reliability, take into consideration the following thermal design points:

(1) Conduct studies to ensure that the ambient temperature (Ta) is maintained as low as possible, avoiding the effects of heat generation from the surrounding area.

(2) If the device's dynamic power consumption is relatively large, conduct studies regarding use of forced air-cooling, circuit board composed of low thermal resistance material, and heat sinks. Such measures can lower the thermal resistance of the package.

(3) Derate the device's absolute maximum ratings to minimize thermal stress from power consumption.

$\theta ja = \theta jc + \theta ca$

$\theta ja = (Tj - Ta)/P$

$\theta jc = (Tj - Tc)/P$

$\theta ca = (Tc - Ta)/P$

where, θja: Thermal resistance between junction and ambient air (°C/W)

θjc: Thermal resistance between junction and package surface, or internal thermal resistance (°C/W)

θca: Thermal resistance between package surface and ambient air, or external thermal resistance (°C/W)

Tj: Junction temperature or chip temperature (°C)

Tc: Package surface temperature or case temperature (°C)

Ta: Ambient temperature (°C)

P: Power consumption (W)
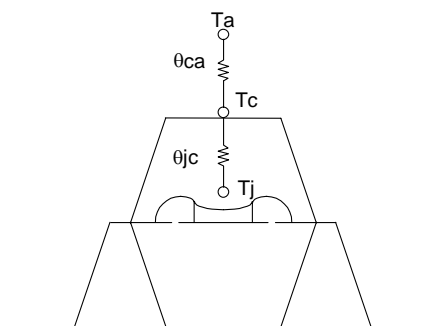
Figure 3.2 Thermal Resistance of Package

3.3.9 Interfacing

When connecting devices with different input and output voltage levels, make sure that the input voltage (V_{IL}/V_{IH}) and output voltage (V_{OL}/V_{OH}) levels match. Otherwise, the devices may malfunction. In addition, when connecting devices with different power supply voltages, such as in a dual power supply system, device breakdown may result if the power-on and power-off sequences are incorrect. For device interface details, refer to the individual technical datasheets and databooks. In addition, if you have any questions about interfacing, contact your nearest Toshiba office or distributor.

3.3.10 Decoupling

Spike currents generated during switching can cause power supply voltage and GND voltage levels to fluctuate, causing ringing in the output waveform or a delay in the response speed. (The power supply and GND wiring impedance is normally 50 to 100Ω.) For this reason, the impedance of the power supply lines with respect to high frequencies must be kept low. Specifically, this is ideally accomplished by routing thick and short power supply and GND lines and by inserting decoupling capacitors (of approximately 0.01 to 1μF) as high-frequency filters between the power supply and GND into each required location on the circuit board.

For low-frequency filtering, it is appropriate to insert a 10 to 100μF capacitor in each circuit board. However, conversely if the capacitance is excessively large (such as 1000μF), latch-up may result. An appropriate capacitance value is therefore required.

On the other hand, in the case of high-speed logic ICs, noise is caused by reflection, crosstalk or common power supply impedance. Reflections cause increased signal delay, ringing, overshoot and undershoot, thereby reducing the device's noise margin. One effective wiring measure for preventing reflections is to reduce the wiring length by increasing the mounting density so as to lower the wiring inductance (L) and capacitance (C). This measure, however, also requires consideration with regard to crosstalk between wires. In actual pattern design, both of these factors must be considered.

3.3.11 External Noise

When externally induced noise or surges are applied to a printed circuit board with long I/O signals or signal lines, malfunction may result, depending on the device. To protect against noise, protective measures against surges must be taken such as lowering the impedance of the signal line or inserting a noise-canceling circuit.

For details of required protection, refer to individual technical datasheets and databooks.
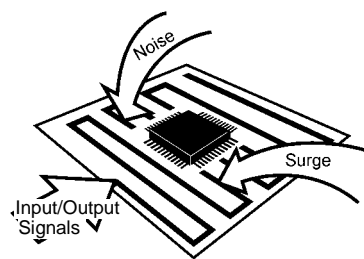
3.3.12   Electromagnetic Interference

Radio and TV reception problems have increased in recent years as a result of increased electromagnetic interference radiated from electrical and electronic equipment. To use radio waves effectively and to maintain the quality of radio communications, each country has defined limitations for the amount of electromagnetic interference which can be generated by designated devices.

The types of electromagnetic interference include noise propagated through power supply and telephone lines, and noise from direct electromagnetic waves radiated from equipment. Different measurement methods and corrective actions are used for each type.

Difficulties in countering electromagnetic interference derive from the fact that there is no means for calculating at the design stage the strength of the electromagnetic waves produced from each component in a piece of equipment. As a result, it is after the prototype equipment has been completed that measurements are taken using dedicated instruments to determine for the first time the strength of the electromagnetic interference. Yet it is possible during system design to incorporate measures for the prevention of electromagnetic interference which can facilitate corrective action after design completion. One effective method, for example, is to design the product with several shielding options, and then select the optimum shielding method based on the results of the measurements subsequently taken.

3.3.13   Peripheral Circuits

In many cases semiconductor devices are used with peripheral circuits and components. The input and output signal voltages and currents in these circuits must be designed to match the specifications of the device, taking into consideration the factors below.

(1) Input voltages and currents that are not appropriate with respect to the input pins may cause malfunction. Some devices contain pull-up or pull-down resistors, depending on specifications. Design your system taking into account the required voltage and current.

(2) The output pins on a device have a predetermined external circuit drive capability. If a drive capability exceeding this value is required, either insert a compensating circuit or take that fact into account when selecting components for use in external circuits

3.3.14   Safety Standards

Each country and region has established safety standards which must be observed. These safety standards sometimes include requirements for quality certification systems and insulation design standards. The safety standards of the respective countries and regions must be taken fully into account to ensure compliant device selection and design.

3.3.15   Other

(1) When designing a system, incorporate fail-safe and other measures according to system application. In addition, debug the system under actual mounting conditions.

(2) If a plastic package device is placed in a strong electric field, surface leakage may occur due to charge-up, resulting in malfunction. When using such a device in a strong electric field, take measures by, for example, protecting the package surface with a conductive shield.

(3) With some memory devices and microcomputers, attention is required at power on or reset release. To ensure that your design is device appropriate, refer to the individual technical datasheets and databooks.

(4) Design the casing so as to ensure that no conductive material (such as a metal pin) can drop from an external source onto a terminal of a mounted device, causing a short.

3.4   Inspection, Testing and Evaluation

3.4.1   Grounding

⚠CAUTION

Check that there is no electrical leakage before grounding measuring equipment or a solder iron.
Electrical leakage may cause the device you are testing or soldering to electrically break down or may cause electric shock.

3.4.2   Inspection Sequence

⚠CAUTION

[1] Do not insert devices in the wrong orientation or incorrectly. Make sure that the positive and negative terminals of power supplies are connected properly. Otherwise, the current or power consumption may exceed the absolute maximum rating, and exceeding the rating(s) may cause the device breakdown, damage or deterioration, and may result injury by explosion or combustion. In addition, do not use any device that is applied the current with inserting in the wrong orientation or incorrectly even just one time.

[2] On the evaluation, inspection or test using AC power with a peak value of 42.4V or DC power exceeding 60V, be sure to connect the electrodes or probes of the testing equipment before activating the power. When you have finished, discharge any electrical charge remaining in the device. Insufficient connection, wrong power-on timing or incomplete discharge may cause electric shock, resulting in injury.

(1) Apply voltage to the device after inserting it into the test jig. At this time, observe the power supply activation or shutdown standards, if existent.

(2) After test completion, be sure that the voltage applied to the device is off before removing the device from the test jig. Removing the device with the power supply on can cause device deterioration or breakdown.

(3) Make sure that no surge voltages from the measuring equipment are applied to the device.

(4) The chips in tape carrier packages (TCPs) are LSI chips and therefore exposed. During inspection, be careful not to crack or scratch the chip.

Electrical contact may also cause chip failure. Therefore make sure that nothing comes into electrical contact with the chip.

3.5   Mounting

There are two types of device packages: lead insertion and surface mount. The items that affect reliability during circuit board mounting include contamination by flux and thermal stress during the soldering process. With surface-mount devices in particular, the most significant problem is thermal stress from solder reflow, when the entire package is subjected to heat. In addition, the mounting method differs according to factors such as chip size and frame design, even for the same package type. For details, refer to the individual technical datasheets and databooks for each device.

When a location such as a soldered area, connecting area or top surface of a device is subjected to vibration, impact or in actual equipment, bonding fault or device destruction may result. Therefore, be sure to keep this in mind at the time of mounting. If a device is subject to especially strong vibration, impact or stress, the package or chip may crack. Thus, at the time of mounting, carefully consider vibration, impact and stress.

3.5.1   Lead Forming

⚠CAUTION   [1] Always wear safety glasses when cutting the leads of a device with clippers or a similar tool. Failure to do so may result in eye damage from the small shavings that fly off the cut ends.

[2] Do not touch the lead tips of a device.

Some devices have leads withsharp tips. Contact to sharp tips may result in a puncture wound.

Semiconductor devices sometimes undergo a process in which the leads are cut and formed before the devices are installed on a printed circuit board. If abnormal stress is applied to the interior of a device during this process, mechanical breakdown or reliability deterioration may result. This is attributable mainly to the relative stress applied between the device itself and the lead, and can result in internal lead damage, adhesive property deterioration and sealant breakdown. Observe the following precautions during the lead-forming process.

(This does not apply to surface-mount devices.)

(1) Lead insertion hole intervals on the printed circuit board should be designed using the same dimension standard as that for the lead interval of the device.

(2) If the lead insertion hole intervals on the printed circuit board do not match the lead interval of the device, do not forcibly insert the device.

(3) For the minimum dimension between a device and printed circuit board, refer to the individual technical datasheets and databooks. When necessary, create space when forming the device's leads. Do not use the spacers for raising devices above the surface of the printed circuit board during soldering. These spacers may continue to expand due to heat even after the solder has solidified, sometimes applying a great amount of stress to the device.

(4) Observe the following when forming the leads of a device:

   (a) When bending a lead, secure the lead at the end of the bending section near the package to ensure that mechanical stress is not applied to the device. Also, do not repeatedly bend or stretch a lead at the same location.

   (b) Do not damage the lead during lead forming.

   (c) Following any other precautions specified in the individual technical datasheets or databooks.
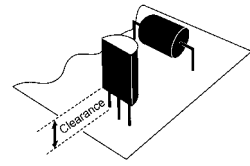
3.5.2   Socket Mounting

(1) When socket-mounting devices on a printed circuit board, use sockets that match the package.

(2) Use sockets with contacts that have the appropriate contact pressure. If the contact pressure is insufficient, the contact may become poor when the device is repeatedly inserted and removed. If the contact pressure is too high, the device leads may bend or become damaged when they are inserted into or removed from the socket.

(3) When soldering sockets to the printed circuit board, use sockets designed to prevent flux from penetrating the contacts and to allow flux to be completely cleaned off.

(4) Ensure that the coating agent applied to the printed circuit board for moisture-proofing does not adhere to the socket contacts.

(5) If the leads are severely bent when inserted into or removed from a socket and you want to repair the leads and continue using the device, repair the leads once only. Do not use devices whose leads have been corrected multiple times.

(6) If external vibration will be applied to a printed circuit board with devices mounted on it, use sockets with strong contact pressure so as to prevent vibration between the devices and sockets.

3.5.3   Lead(Pb)-Free / Lead(Pb)-Free Finish* Soldering Temperature Profile

Perform soldering following the methods and conditions described in the individual technical datasheets and databooks for the device used. The soldering method, temperature and time may be restricted, depending on the device. All soldering temperature profiles and conditions described in the mounting methods below are representative. The profiles and conditions vary from product to product. Therefore, mount the product after first confirming the information described in the individual technical datasheets and databooks with the customer.

For details regarding lead(Pb) soldering, please contact your nearest Toshiba office or distributor.

*Toshiba Semiconductor Company defines capitalized "Lead(Pb)-Free" products as those Containing no more than 0.1 percent lead(Pb) by weight in homogeneous materials. This does not mean that Toshiba Semiconductor products labeled "Lead(Pb)-Free" are entirely free of

lead(Pb). In addition to Lead(Pb)-Free, Toshiba Semiconductor Company will offer products that have Lead(Pb)-Free terminals, which will be referred to as "Lead(Pb)-Free Finish." The Lead(Pb)-Free Finish products may Contain greater than 0.1 percent lead(Pb) by weight in homogeneous materials in portions of the product other than the terminals (based on the exemption(s) in the RoHS Directive), for example, in internal solder used to connect the semiconductor silicon to the package. This does not mean that Toshiba Semiconductor products that are labeled "Lead(Pb)-Free Finish" have terminals that are entirely free of lead(Pb). Furthermore, the expressions "Lead(Pb)-Free" and "Lead(Pb)-Free Finish" will be changed in package labeling as the like below from April 2006.

<<Examples of correspondence with the existing lead (Pb)-free markings>>

[Lead (Pb)-free products]: Lead (Pb)-Free -> [[G]]/RoHS COMPATIBLE

[Lead (Pb)-free finish products]: Lead(Pb)-Free Finish -> [[G]]/RoHS [[Pb]]

3.5.3.1 Using a Soldering Iron

Complete soldering within 10 seconds for lead temperatures of up to 260°C, or within 3 seconds for lead temperatures of up to 350°C.

3.5.3.2 Using Infrared Reflow

(1) It is recommended the top and bottom heating method with long or medium infrared rays. (See Figure 3.3.)
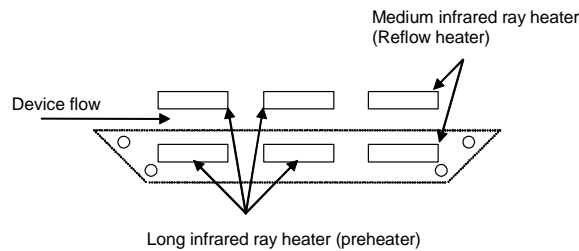
Figure 3.3 Top and Bottom Heating Method with Long or Medium Infrared Rays

(2) Complete the infrared ray reflow process with a maximum package surface temperature of 260°C, within 30 to 50 seconds when a package surface temperature is 230°C or higher.

(3) Refer to Figure 3.4 for an example of a temperature profile.
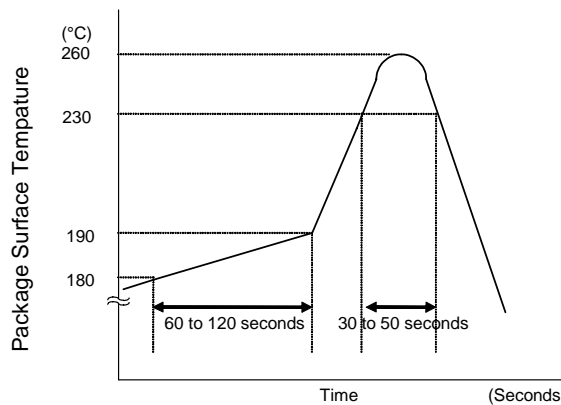
Figure 3.4 Example of Temperature Profile

This profile is based on the device's maximum heat resistance guaranteed value.

Set the preheat temperature/heating temperature to the optimum temperature corresponding to the solder paste type used by the customer within the above-described profile.

3.5.3.3   Using Hot Air Reflow

(1)   Complete hot air reflow with a maximum package surface temperature of 260°C, within 30 to 50 seconds when a package surface temperature is 230°C or higher.

(2)   For an example of a temperature profile, refer to Figure 3.4 in Section 3.5.3.2 (3) above.

3.5.3.4   Using Solder Flow/Dip

(1)   Apply preheating for 60 to 120 seconds at a temperature of 150°C.

(2)   For lead insertion-type packages, mount the device within 10 seconds of solder flow with a maximum temperature of 260°C at the stopper or at a location more than 1.5mm from the body.

(3)   For surface-mount packages, mount the device within 5 seconds at a temperature of 250°C or less in order to avoid thermal stress.

(4)   Figure 3.5 shows an example of the temperature profile of solder flow for a surface-mount package.
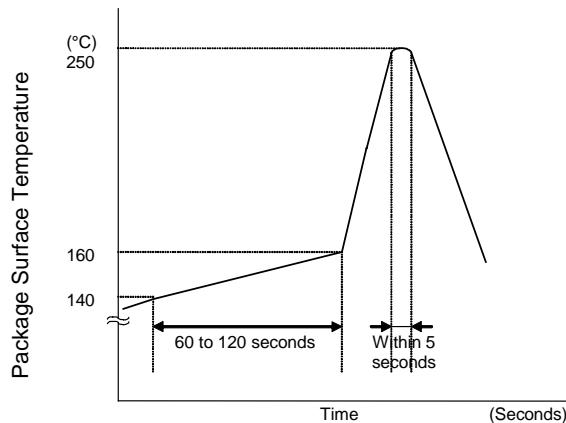
Figure 3.5 Example of Surface-Mount Package Temperature Profile

This profile is based on the device's maximum heat resistance guaranteed value.

Set the preheat temperature/heating temperature to the optimum temperature corresponding to the solder paste type used by the customer within the above-described profile.

3.5.4   Flux Cleaning

(1)   When cleaning circuit boards to remove flux, make sure that no reactive ions such as sodium or chlorine remain. Some organic solvents react with water to generate hydrogen chloride and other corrosive gases which can result in device deterioration.

(2)   When washing devices with water, make sure that no reactive ions such as sodium or chlorine remain particularly.

(3) When washing devices, do not rub markings with a brush or with your hand while the cleansing liquid is still on the device. Doing so can rub off the markings.

(4) Dip cleaning, shower cleaning and steam cleaning processes are performed based on the chemical action of a solvent. When immersing devices in a solvent or steam bath, complete the cleaning for a period of one minute or less at a liquid temperature of 50°C or less, taking into consideration the effects on the devices.

(5) Avoid use of ultrasonic cleaning with hermetically sealed ceramic packages such as a leadless chip carrier (LCC), pin grid array (PGA) or charge-coupled device (CCD). Using the ultrasonic cleaning may cause the internal wires to become disconnected due to resonance. Even if a device package allows ultrasonic cleaning, keep the duration of ultrasonic cleaning in a brief time. Long hours of ultrasonic cleaning may deteriorate the adhesion between the mold resin and frame material.

The basic recommended conditions are as follows:

Recommended Ultrasonic Cleaning Conditions

Frequency: 27 to 29kHz

Ultrasonic output: 15W/L or less

Cleaning time: 30 seconds or less

Suspend the printed circuit board in the solvent bath to ensure that the circuit board and device do not come in direct contact with the ultrasonic vibrator.

3.5.5   No Cleaning

It is recommended that you clean analog devices and high-speed devices. If such devices are not cleaned, flux may cause minute leakage between leads or migration, depending on the flux grade. Be sure therefore to check cleanliness at the time of use. If you are considering no cleaning, be sure to use a flux that does not require cleaning.

3.5.6   Tape Carrier Packages (TCPs) Mounting

(1) When tape carrier packages are mounted, measures must be taken to prevent electrostatic breakdown of the devices.

(2) When separating devices from tape, or carrying out outer lead bonding (OLB) mounting, be sure to take work safety into consideration.

(3) The base film, which is made of polyimide, is hard and thin. Be careful not to injury yourself or damage any objects during handling.

(4) When punching tape, take countermeasures to prevent minute broken pieces from scattering. Scattered pieces may cause injury.

(5) Appropriately treat the tape, reels and spacers left after separating the device as industrial waste.

(6) With tape carrier package (TCPs) devices, the backside of the LSI chips is exposed. To ensure that the chip will not crack, mount the device so that mechanical shock is not applied to the LSI backside. In addition, electrical contact may also cause LSI failure. Mount the device so that there is no electrical contact with the backside of the LSI chip.

If you are mounting the backside of the LSI chip to improve device characteristics, please contact your nearest Toshiba office or distributor in advance.

3.5.7   Chips Mounting

Devices delivered in chip form readily deteriorate or become damaged due to external factors in comparison with plastic-packaged products. Attention is therefore required during handling.

(1) Mount devices in a properly maintained environment so that the chip will not be exposed to contaminated ambient air or other substances.

(2) When handling chips, be careful not to expose the chips to static electricity. In particular, measures must be taken to prevent electrostatic breakdown during chip mounting. For this purpose, it is recommended that you mount all peripheral devices before you mount the chips.

(3) Use chip mounting circuit boards (such as PCBs) that do not have any chemical residue on them (such as the chemicals used during PCB etching).

(4) When mounting chips, use the method of assembly that is most suitable for achieving the appropriate electrical, thermal and mechanical characteristics of the semiconductor product used.

  *   For chip details, refer to the relevant specification sheet.

3.5.8   Circuit Board Coating

When using devices that require high reliability or devices used under extreme environments (where moisture, corrosive gas or dust is present), circuit boards are sometimes coated with a moisture-proof coating. When using a coating resin, choose the coating resin which results in minimal stress to the device.

3.5.9   Heat Sinks

(1) When installing a heat sink to a device, use the specified accessories. In addition, be careful not to apply excessive force to the device during installation.

(2) When installing a device to a heat sink by fixing it in two or more locations, do not tighten one location to the specified torque while the rest are left not tightened. Rather, lightly tighten all locations evenly first and tighten all locations to the specified torque by rotation.

(3) Drill screw holes in the heat since as specified, and smooth the surface of the device installation area by removing burrs and protrusions or indentations.

(4) Thinly applying silicone grease between the heat sink makes device better to improve heat conductivity compared with no grease. If you choose to apply the silicone grease, use a non-volatile type. Volatile type silicone grease can cause
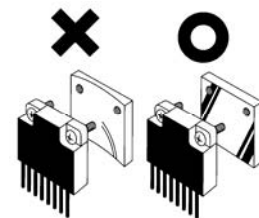
cracks over time, resulting in the deterioration of the heat radiation effect.

(5) With plastic-packaged devices, the base oil of some silicone grease compounds penetrates the package interior, significantly reducing the lifetime of the device. We ask therefore that you use the recommended silicon grease YG6260 from GE Toshiba Silicone. If you choose to use another product, select one that is equivalent to the Toshiba Silicone product.

(6) During device operation, heat sinks become very hot. Be careful not to touch them. A burn may result.

3.5.10   Tightening Torque

(1) Tighten screws to a tightening torque that is within the specified values described in the individual technical datasheets and databooks for the device used.

(2) Be careful not to allow a pneumatic screwdriver to come in contact with devices. Device damage may result.

3.5.11   Repeated Device Mounting and Usage

Do not remount or reuse devices that have histories such as that described below. These devices may cause significant problems with regard to device characteristics and reliability.

(1) Devices that have been removed from the board after soldering.

(2) Devices that have been inserted in the wrong orientation or with reverse polarity and charged.

(3) Devices that have undergone lead forming more than once.

3.6   Operating Environment

3.6.1   Temperature

Semiconductor devices are generally more sensitive to temperature than other electromechanical parts. The various electrical characteristics of a semiconductor device are restricted by the operating temperature. It is therefore necessary to understand the temperature characteristics of a device and incorporate derating into the device design in advance. When a device is used at a temperature outside the specified operating range, electrical characteristics will not be realized and device deterioration will occur more rapidly.

3.6.2   Humidity

Plastic package devices are sometimes not completely sealed.

When these devices are used for an extended period of time under high humidity, moisture can seep into the device and cause semiconductor chip deterioration or failure. Furthermore, when devices are mounted on a regular printed circuit board, the impedance between wiring can decrease under high humidity. In systems with a high signal-source impedance, circuit board leakage or leakage between device leads can cause malfunction. In such a case, moisture-proof treatment to the device surface should be considered. On the other hand, operation under low humidity can damage a device due to the occurrence of electrostatic discharge. Unless moisture-proof treatments have been specifically taken, use devices within the humidity range of 40 to 60%.

3.6.3   Corrosive Gases

Devices react to corrosive gases may cause deteriorating device characteristics. For example, consideration must be given to lead corrosion and leakage between leads caused by the chemical reaction that occurs when a device is placed near a rubber product. The reason is that the rubber product will not only produce condensation but also generate sulfur-bearing corrosive gases under high-humidity conditions.

3.6.4   Radioactive and Cosmic Rays

Standard devices are not designed with protection against radioactive and cosmic rays. Devices must therefore be shielded if the device will be used in environments that may result in exposure to radioactive or cosmic rays above the levels that exist in the natural environment.

3.6.5   Strong Electrical and Magnetic Fields

Devices exposed to magnetic fields can undergo a polarization phenomenon in the plastic material or within the IC chip, which gives rise to abnormal conditions such as impedance changes or leak current increases. Malfunctions have been reported in LSIs mounted near television deflection yokes. In such cases, the device installation location must be changed or the device must be shielded against the electrical or magnetic field. Shielding against magnetism is especially required in an alternating magnetic field due to the electromotive forces generated.

3.6.6   Interference from Light (such as Ultraviolet Rays, Sunlight, Fluorescent Lamps, Incandescent Lamps)

Light striking a semiconductor device generates electromotive force due to photoelectric effects, sometimes causing malfunction. Devices in which the chip is visible through the package are especially affected by such light. When designing the circuits, make sure that the devices are protected against light interference. Not just optical semiconductor devices, but all types of devices are affected by light.

3.6.7   Dust and Oil

Similar to corrosive gases, dust and oil cause chemical reactions in semiconductor products, sometimes adversely affecting product characteristics. Be sure to use semiconductor products in an environment that will not result in dust or oil adhesion. Solvent and oil contained in heat release sheets similarly may result in semiconductor product quality deterioration, characteristic deterioration or disconnection. Be sure to use such products with care.

3.6.8   Smoke and Ignition

Semiconductor devices and modularized devices are not noncombustible; they can emit smoke or ignite when excessive current or failure occurs. When this happens, poisonous gases may be produced.

Be sure to develop a safe design that protects the device from excessive current so as to ensure excessive current does not flow within the device during operation or at the time of failure.

To prevent the propagation of fire caused by a smoking or ignited Toshiba product and to ensure that Toshiba products do not emit smoke or ignite due to surrounding conditions, do not use Toshiba products in close proximity to combustible thing, heat-generating thing, igniting materials or flammable materials.

3.7   Disposal

Each country and region has laws and regulations for the proper disposal of devices and packing materials. Be sure to follow these laws and regulations at the time of disposal.

4. Precautions and Usage Considerations Specific to Each Product Group

This section describes the matters specific to each product group which need to be taken into consideration. The precautions described in this section take precedence over those described in Section 3, "General Safety Precautions and Usage Considerations."

4.1 Microcomputers

4.1.1 Design

(1) Use of Crystal Oscillators Other than Those Recommended

For components such as crystal oscillators that are used in the oscillation circuit of microcomputer products, use the components with usage conditions described in the individual technical datasheets and databooks. If you plan on using components with usage conditions other than those recommended, contact the our engineering department stated in the individual technical datasheet or databook, or consult with the oscillator manufacturer.

(2) Undefined Functions

Microcomputer products have commands that are not individually defined for each product (i.e., undefined commands). These products also similarly have undefined functions (for instance, bits to which functions are not assigned in the register).

Refer to the product's individual technical datasheets and databooks and do not use the undefined commands and undefined functions.

TMPR4951B

Rev. 2.1

1.   Introduction

1.1   Overview

The TMPR4951B (to be called "TX4951B" hereinafter) is a standard microcontroller of 64-bit RISC Microprocessor TX49 family.

The TX4951B uses the TX49/L3 Processor Core as the CPU. The TX49/L3 Processor Core is a 64-bit RISC CPU core Toshiba developed based on the R4000 architecture of MIPS Technologies, Inc ("MIPS"). The TX4951B incorporates peripheral circuits such as SysAD Bus Interface.

1.2   Notation used in this manual

1.2.1   Numerical notation

- Hexadecimal numbers in this manual are expressed as follows:0x2A (example shown for decimal number 42)

- KB (kilobyte) 2^{10} = 1,024 bytes
  MB (megabyte) 2^{20} = 1,024 × 1,024 = 1,048,576 bytes
  GB (gigabyte) 2^{30} = 1,024 × 1,024 × 1,024 = 1,073,741,824 bytes

1.2.2   Data notation

- Byte: Eight bits

- Half word: Two contiguous bytes (16 bits)

- Word: Four contiguous bytes (32 bits)
  "W" may be used for a word data.

- Double word: Eight contiguous bytes (64 bits)
  "D" may be used for a double word data.

1.2.3   Signal notation

- Active-low signals are indicated by adding an asterisk(*) at the end of the signal name (Example: RESET*)

- When a signal is driven to the active voltage level, the signal is said to be "asserted." When the signal is driven to an inactive voltage level, it is said to be "deasserted."

1.2.4   Register notation

- The following nomenclature is used for access attributes.

    R: Read only. Cannot be written.

    W: Write only. The bit value is undefined if read.

    R/W: Read/Write

R4000/R4300/R5000 are a trademark of MIPS Technologies, Inc.

2. Features

- **TX49/L3 Processor Core**
  TX49/L3 Processor Core is a 64-bit RISC CPU core Toshiba developed based on the architecture of MIPS for interactive consumer applications including Printer, Network and set-top terminals.

- **Internal bus width is 64-bit, External bus width is 32-bit**
  Core and **Cache** are connect with 64-bit Internal bus. External bus is 32-bit SysAD-bus I/F. This interface is compatible with the R4300, and R5000 system interfaces.

- **Power management**
  Internal: 1.5 V   I/O: 3.3 V or 2.5 V
  The TX49/L3 Processor support Power management mode (Halt, Doze)

- **Maximum operating frequency**
  The TX49/L3 Processor's maximum operating frequency are 200 MHz.
  The SysAD-bus I/F and TX49/L3 Processor's maximum operating frequency is set by External pin (DivMode[1:0]).

  Ex. Core's operating frequency is 300 MHz

DivMode[1:0]	MasterClock	CPU Clock	
00	50 MHz	200 MHz	1:4
01	80 MHz	200 MHz	1:2.5
10	100 MHz	200 MHz	1:2
11	66.6 MHz	200 MHz	1:3

- **Package**
  TX4951B: 100-pin LQFP

- **Part Number**
  TMPR4951BFG-200: Maximum internal operating frequency = 200 MHz
  (Ask your nearest Toshiba sales representative for the latest part number.)

2.1   Block Diagram
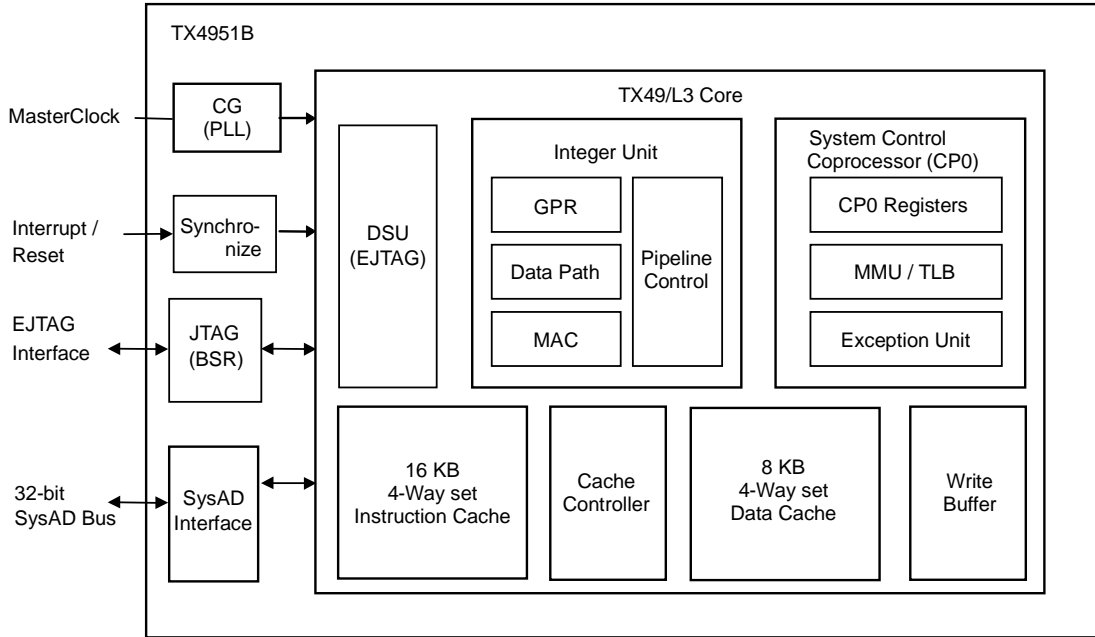
Figure 2.1.1 TX4951B Processor Signals

2.2   Pin Description

2.2.1   TX4951B pin out (100-pin LQFP)

Table 2.2.1 TX4951B Pin Out

1	VccIO	26	Vss	51	Vss	76	Vss
2	MasterClock	27	RdRdy* / (GND)	52	VccInt	77	VccInt
3	VccInt	28	WrRdy* / (EOK*)	53	SysCmd0	78	SysAD24
4	VccPLL	29	ValidIn* / (EValid*)	54	SysCmd1	79	SysAD25
5	VssPLL	30	ValidOut* / (PValid*)	55	SysCmd2	80	SysAD26
6	Vss	31	Release* / (PMaster*)	56	SysCmd3	81	SysAD27
7	PLLReset*	32	VccInt	57	Vss	82	Vss
8	VccIO	33	Vss	58	VccIO	83	VccIO
9	SysAD4	34	ExtRqst*	59	SysCmd4	84	SysAD28
10	SysAD5	35	TRST*	60	SysCmd5 / (GND)	85	SysAD29
11	SysAD6	36	TintDis	61	SysCmd6 / (GND)	86	SysAD30
12	SysAD7	37	VccIO	62	VccInt	87	SysAD31
13	Vss	38	HALTDOZE	63	Vss	88	Vss
14	VccIO	39	NMI*	64	SysCmd7 / (GND)	89	VccIO
15	SysAD8	40	Int0*	65	SysCmd8 / (GND)	90	SysAD0
16	SysAD9	41	Int1*	66	SysAD16	91	SysAD1
17	SysAD10	42	Int2*	67	SysAD17	92	SysAD2
18	SysAD11	43	Int3*	68	SysAD18	93	SysAD3
19	VccInt	44	Vss	69	SysAD19	94	MODE43*
20	Vss	45	VccInt	70	Vss	95	Vss
21	SysAD12	46	DivMode0	71	VccIO	96	VccInt
22	SysAD13	47	DivMode1	72	SysAD20	97	JTDO
23	SysAD14	48	Endian	73	SysAD21	98	JTDI
24	SysAD15	49	ColdReset*	74	SysAD22	99	JTCK
25	VccIO	50	VccIO	75	SysAD23	100	JTMS

Note 1: "*" means the signal is the low-active.

Note 2: MODE43* = 0: R4300 mode; MODE43* = 1: R5000 mode.

Note 3: In R4300 mode, a logic 0 is driven out from the (GND) pins. These pins must be left open on the board. Pin 27 is an input pin with an internal pull-down resistor.

Table 2.2.2 System Interface (when MODE43* = 1)

PIN NAME	I / O	FUNCTION
SysAD[31:0]	I / O	**System addres/data bus** A 32-bit address and data bus for communication between the processor and an external agent.
SysCmd[8:0]	I / O	**System command/data identifier bus** A 9-bit bus for command and data identifier transmission between the processor and an external agent.
ValidIn*	I	**Valid input** The external agent asserts ValidIn* when it is driving a calid address or data on the SysAD bus and valid command or data identifier on the SysCmd bus.
ValidOut*	O	**Valid output** The processor asserts ValidOut* when it is driving a valid address or data on the SysAD bus and a valid command or data identifier on the SysCmd bus.
ExtRqst*	I	**External request** An external agent asserts ExtRqst* to request use of the System interface.
Release*	O	**Release interface** Signals that the system interface needs to submit an external request.
WrRdy*	I	**Write Ready** Signals that an external agent can now accept a processor write request.
RdRdy*	I	**Read Ready** Signals that an external agent can now accept a processor read request (with pull-down register).

Table 2.2.3 System Interface (when MODE43* = 0)

PIN NAME	I / O	FUNCTION
SysAD[31:0]	I / O	**System addres/data bus** A 32-bit address and data bus for communication between the processor and an external agent.
SysCmd[4:0]	I / O	**System command/data identifier bus** A 5-bit bus for command and data identifier transmission between the processor and an external agent.
ValidIn*/Evalid*	I	**External agent valid input** The external agent asserts ValidIn* when it is driving a calid address or data on the SysAD bus and valid command or data identifier on the SysCmd bus.
ValidOut*/ Pvalid*	O	**Processor valid output** The processor asserts ValidOut* when it is driving a valid address or data on the SysAD bus and a valid command or data identifier on the SysCmd bus.
ExtRqst*/EReq*	I	**External request** An external agent asserts ExtRqst* to request use of the System interface.
Release*/ PMaster*	O	**Processor Master** This signal indicates that the processor is a bus master.
WrRdy*/EOK*	I	**External agent Ready** Signals that an external agent can now accept a processor write request.
RdRdy*/GND	I	**Reserved** This signal must be set 0 (with pull-down register).

Table 2.2.4 Clock/Control Interface

PIN NAME	I / O	FUNCTION
MasterClock	I	**MasterClock** MasterClock input that establishes the processor operating frequency.

DivMode[2:0]	I	**Set the operational frequency of the System interface**
		
		DivMode[1:0] EC-bit MasterClock CPUCLK
		  00 010 50 MHz 200 MHz 1:4
		  01 111 80 MHz 200 MHz 1:2.5
		  10 000 100 MHz 200 MHz 1:2
		  11 001 66.6 MHz 200 MHz 1:3

TintDis	I	Timer-Interrupt disable input
		  0 enable Timer-Interrupt
		  1 disable Timer-Interrupt

HALT/DOZE	O	**HALT/DOZE mode output** This signal output the status of HALT or DOZE mode. This signal indicates that the TX4951B is in the HALT or DOZE mode when this signal is "H".

Endian	I	**Endianess input**
		Indicates the initial setting of the endian during a reset.
		  0 Little Endian
		  1 Big Endian

Table 2.2.5 Interrupt Interface

PIN NAME	I / O	FUNCTION
Int[3:0]∗	I	**Interrupt** Six lines of general-purpose processor interrupt inputs, which are sampled at rising edges of the MasterClock (with pull-up register).
NMI∗	I	**Nonmaskable interrupt** Nonmaskable interrupt input, which is sampled at rising edges of the MasterClock (with pull-up register).

Table 2.2.6 JTAG Interface

PIN NAME	I / O	FUNCTION
JTDI	I	**JTAG data input / Debug interrupt input** Run-time mode: Input serial data to JTAG data/instruction registers. Real-time mode: Interrupt input to change the debug unit state from real-time mode to run-time mode (with pull-up register).
JTCK	I	**JTAG clock input** Clock input for JTAG. The JTDI and JTMS data are latched on rising edges of this clock (with pull-up register).
JTDO	O	**JTAG data output / Trace PC output** Data is serially shifted out from this pin.
JTMS	I	**JTAG command** Controls mainly the status transition of the TAP controller state machine. When the serial input data is a JTAG command, apply a high signal (= 1) to this pin (with pull-up register).
TRST*	I	**Test reset input** Reset input for a real-time debug system. When TRST* is asserted (= 0), the debug support unit (DSU) is initialized. TRST* should be asserted when DSU is not used (with pull-down register).

Note1: Leave TPC (3-1) pins open when not using them as PC trace outputs for debugging.

Table 2.2.7 Initialization Interface

PIN NAME	I / O	FUNCTION
ColdReset*	I	**Cold reset** Assert this signal at power-on and for a cold reset. GBUSCLK starts operating synchronously with this signal (with pull-up register).

PLLReset*	I	**PLL reset input**
		A signal to halt the PLL oscillation or the TX4951B built-in clock generator.
		    0 PLL is halt (no oscillation)
		    1 PLL is enabled.

MODE43*	I	**SysAD bus protocol selection**
		The high or low level of this input signal at power-on or cold reset selects the SysAD bus protocol.
		    Low level: R4300 type protocol
		    High level: R5000 type protocol

Table 2.2.8 Power Supply

PIN NAME	I / O	FUNCTION
VccPLL	–	**Vcc for the PLL** This is a static Vcc for the internal Phase Locked Loop. (Apply a 1.5 V power supply.)
VssPLL	–	**Vss for the PLL** This is a static Vss for the internal Phase Locked Loop.
VccIO	–	**VccIO** This is a 3.3 V or 2.5 V power supply pin.
VccInt	–	**VccInt** This is a 1.5 V power supply pin.
Vss	–	**Vss** This is the ground pin.

3.   TX49/L3 Core's Registers

3.1 CPU Registers

The TX4951B with TX49/L3 has the 64-bit CPU registers.

- 32 general-purpose registers
- 64-bit program counters
- HI/LO register for storing the result of multiply and divide operations

Figure 3.1.1 shows the configuration of these registers.

General Purpose Registers (GPR)   Multiply/Divide Registers

63	0
r0 = 0	
r1	
r2	
.	
.	
r29	
r30	
r31 = Link Address	

63	0
HI	

63	0
LO	

Program counter

63	0
PC	
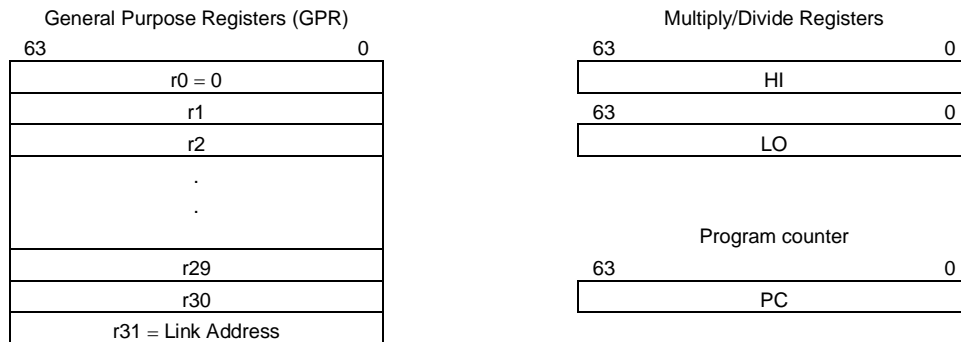
Figure 3.1.1 CPU registers

The r0 and r31 registers of GPR have special functions as follows.

- Register r0 always contains the value 0. It can be a target register of an instruction whose operation result is not needed. Or, it can be a source register of an instruction that requires a value of 0.
- Register r31 is the link register for the Jump and Link instruction. The address of the instruction after the delay slot is placed in r31.

The TX49/L3 Core has the following some special registers that are used or modified implicitly by certain instructions.

- HI - Holds the high-order bits of the result of integer multiply operation or the remainder of integer divide operation.
- LO - Holds the low-order bits of the result of integer multiply operation or the quotient of integer divide operation.

These two registers are used to store that result of an integer multiplication or division. In multiplication, the 64 high-order bits of a 128-bit result are stored in the HI, and the 64 low-order bits are stored in the LO. In division, the resulting quotient is stored in the LO, and the remainder is stored in the HI.

- PC - Program Counter

The register contains the address of the currently executed instruction.

Note: TX49/L3 does not support the floating-point unit, so it does not have the floating-point general purpose registers. If the floating-point instruction execute, it will cause the Coprocessor Unusable Exception.

3.2 CP0 Registers

The TX49/L3 Core has the 32-bit or 64-bit System control coprocessor(CP0) registers. These registers are used for memory system or exception handling. Table 3.2.1 lists the CP0 registers built into the TX49/L3 Core.

Table 3.2.1 CP0 Registers

Register Name	Reg. No.	Register Name	Reg. No.
Index	Reg#0	Config	Reg#16
Random	Reg#1	LLAddr	Reg#17
EntryLo0	Reg#2	(Reserved) (Note 1)	Reg#18
EntryLo1	Reg#3	(Reserved) (Note 1)	Reg#19
Context	Reg#4	XContext	Reg#20
PageMask	Reg#5	(Reserved) (Note 1)	Reg#21
Wired	Reg#6	(Reserved) (Note 1)	Reg#22
(Reserved) (Note 1)	Reg#7	Debug (Note 2)	Reg#23
BadVAddr	Reg#8	DEPC (Note 2)	Reg#24
Count	Reg#9	(Reserved) (Note 1)	Reg#25
EntryHi	Reg#10	(Reserved) (Note 1)	Reg#26
Compare	Reg#11	(Reserved) (Note 1)	Reg#27
Status	Reg#12	TagLo	Reg#28
Cause	Reg#13	TagHi	Reg#29
EPC	Reg#14	ErrorEPC	Reg#30
PRId	Reg#15	DESAVE (Note 2)	Reg#31

Note 1: These registers are used to test the System Control Coprocessor (CP0) and should not be accessed by the user.

Note 2: These registers are exclusively used by external in-circuit emulators (ICE).

3.2.1 Index register (Reg#0)

The Index register is a 32-bit read/write register containing six bits to index an entry in the TLB. The P bit of the register shows the success/failure of a TLB Probe (TLBP) instruction.

The Index register also specifies the TLB entry affected by TLB Read (TLBR) or TLB Write Index (TLBWI) instructions. Figure 3.2.1 shows the format of the Index register and Table 3.2.2 describes the Index register fields.

31	30	6	5	0
P	0		Index	
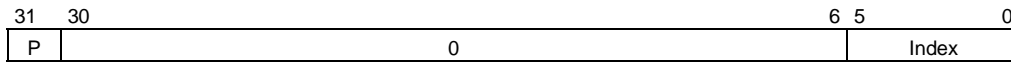
Figure 3.2.1 Index Register Format

Table 3.2.2 Index Register Field Descriptions

Bit	Field	Description	Cold Reset	Read/Write
31	P	Probe failure. Set to 1 when the previous TLB Probe (TLBP) instruction was unsuccessful.	Undefined	Read/Write
30:6	0	Reserved	0x0	Read
5:0	Index	Index to the TLB entry affected by the TLB Read and TLB Write Index instructions.	Undefined	Read/Write

3.2.2   Random register (Reg#1)

The Random register is a read only register containing six bits to index an entry in the TLB. This register decrements as each instruction executes. The values are as follows.

- A lower bound is set by the number of TLB entries reserved for exclusive use by the operating system (the contents of the Wired register).

- An upper bound is set by the total number of TLB entries (47 maximum).

The Random register specifies the TLB entry affected by TLB Write Random (TLBWR) instruction. However the register doesn't need to be read for this purpose, it is readable to verify proper operation of the processor.

To simplify testing, the Random register is set to the value of the upper bound upon system reset. This register is also set to the upper bound when the Wired register is written.

Figure 3.2.2 shows the format of the Random register and Table 3.2.3 describes the Random register fields.

31	6	5	0
0		Random	
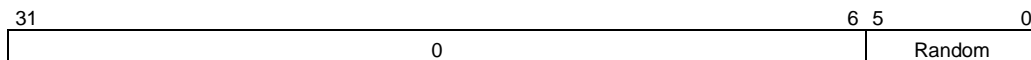
Figure 3.2.2 Random Register Format

Table 3.2.3 Random Register Field Descriptions

Bit	Field	Description	Cold Reset	Read/Write
31:6	0	Reserved.	0x0	Read
5:0	Random	TLB random index for TLBWR instruction.	Upper bound (47)	Read

3.2.3   EntryLo0 register (Reg#2) and EntryLo1 register (Reg#3)

The EntryLo register consists of two registers have identical formats:

- EntryLo0 is used for even virtual pages
- EntryLo1 is used for odd virtual pages

The EntryLo0 and EntryLo1 register are read/write register. These registers hold the physical page frame number (PFN) of the TLB entry for even and odd pages, respectively, when performing TLB read and write operations.

Figure 3.2.3 shows the format of the EntryLo0/EntryLo1 register and Table 3.2.4 describes the EntryLo0/EntryLo1 register fields.

63	32	31	30	29	6	5	3	2	1	0
0		WCE		PFN		C		D	V	G
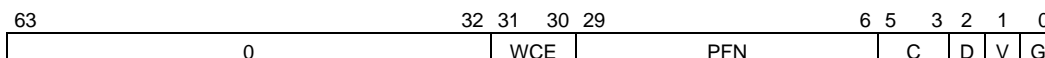
Figure 3.2.3 EntryLo0/EntryLo1 Register Format

Table 3.2.4 EntryLo0/EntryLo1 Register Field Descriptions

Bit	Field	Description	Cold Reset	Read/Write
63:32	0	Reserved	0x0	Read
31:30	WCE	Usable for Win-CE	0x0	Read/Write
29:6	PFN	Page frame number.	Undefined	Read/Write
5:3	C	Specifies the TLB page coherency attribute. 0: Cacheable, noncoherent, write-through, no-WA 1: Cacheable, noncoherent, write-through, WA 2: Uncached 3: Cacheable,noncoherent,write-back,WA 4-7: Reserved	0x0	Read/Write
2	D	Dirty If this bit is set, the page is marked as dirty and, therefore, writable. This bit is actually a write-protect bit that software can use to prevent alteration of data.	0	Read/Write
1	V	Valid If this bit is set, it indicates that the TLB entry is valid; otherwise, a TLBL or TLBS miss occurs.	0	Read/Write
0	G	Global If this bit is set in both EntryLo0 and EntryLo1, then the processor ignores the ASID during TLB lookup.	0	Read/Write

3.2.4 Context register (Reg#4)

The Context register is a read/write register containing the pointer to an entry in the page table entry (PTE) array. This array is an operating system data structure that stores virtual to physical address translations. When there is a TLB miss, the CPU loads the TLB with the missing translation from the PTE array. Normally, the operating system uses the Context register to address the current page map which resides in the kernel mapped segment,kseg3. However the contents of this register duplicates some information of the BadVAddr register, it is arranged in a form that is more useful for TLB exception handler by a software.

Figure 3.2.4 shows the formats of the Context register and Table 3.2.5 describes the Context register fields.

```
31                         23 22                                      4 3    0
+---------------------------+------------------------------------------+-----+
|          PTEBase          |                 BadVPN2                  |  0  |
+---------------------------+------------------------------------------+-----+
                              (32-bit mode)

63                                            23 22                    4 3    0
+-----------------------------------------------+----------------------+-----+
|                   PTEBase                     |       BadVPN2        |  0  |
+-----------------------------------------------+----------------------+-----+
                              (64-bit mode)
```
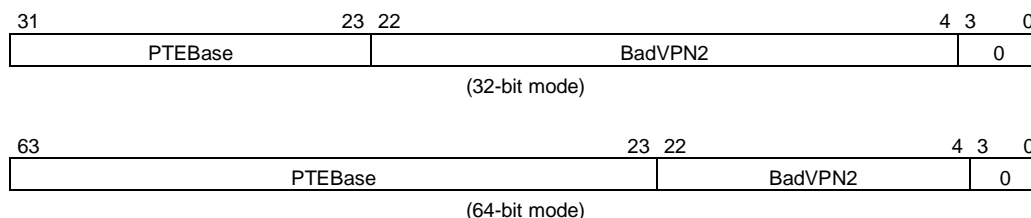
Figure 3.2.4 Context Register Formats

Table 3.2.5 Context Register Field Descriptions

32-bit mode

Bit	Field	Description	Cold Reset	Read/Write
31:23	PTEBase	Page table entry base pointer This field is for use by the operating system. It is normally written with a value that allows the operating system to use the Context register as a pointer into the current PTE array in memory.	Undefined	Read/Write
22:4	BadVPN2	Bad virtual address bits 31-13 This field is written by hardware on a miss. It contains the virtual page number (VPN) of the most recent virtual address that did not have a valid translation.	Undefined	Read
3:0	0	Reserved	0x0	Read

64-bit mode

Bit	Field	Description	Cold Reset	Read/Write
63:23	PTEBase	Page table entry base pointer	Undefined	Read/Write
22:4	BadVPN2	Bad virtual address bits 31-13	Undefined	Read
3:0	0	Reserved	0x0	Read

The 19-bit BadVPN2 field contains bits 31 to 13 of the virtual address that caused the TLB miss; bits 12 is excluded because a single TLB entry maps to an even-odd page pair. For a 4-Kbyte page size, this format can directly address the pair-table of 8-byte PTEs. For other page size and PTE sizes, shifting and masking this value produces the appropriate address.

3.2.5 PageMask Register (Reg#5)

The PageMask register is a read/write register used for reading from/writing to the TLB. This register holds a comparison mask that sets the variable page size for each TLB entry.

TLB read and write operations use this register as either a source or a destination. When virtual addresses are presented for translation into physical address, the corresponding bits in the TLB identify which virtual address bits among bits 24-13 are used in the comparison. When the Mask field is not one of the values shown in Table 3.2.6, the operation of the TLB is undefined.

Figure 3.2.5 shows the format of the PageMask register and Table 3.2.6 describes the PageMask register fields.

31	25 24	13 12	0
0	MASK	0	
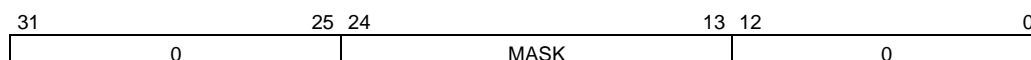
Figure 3.2.5 PageMask Register Format

Table 3.2.6 PageMask Register Field Descriptions

Bit	Field	Description	Cold Reset	Read/Write
31:25	0	Reserved	0x0	Read
24:13	MASK	Page comparison mask 000000000000: page size = 4 Kbytes 000000000011: page size = 16 Kbytes 000000001111: page size = 64 Kbytes 000000111111: page size = 256 Kbytes 000011111111: page size = 1 Mbytes 001111111111: page size = 4 Mbytes 111111111111: page size = 16 Mbytes	0x0	Read/Write
12:0	0	Reserved	0x0	Read

3.2.6   Wired Register (Reg#6)

The Wired register is a read/write register specifies the boundary between the wired and random entries of the TLB as follows. Wired entries are non-replaceable entries, which can not be overwritten by a TLB write random operation. Random entries can be overwritten.

TLB
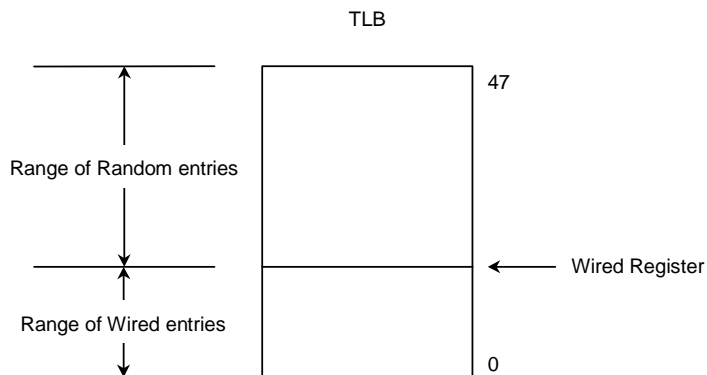
The Wired register is set to 0 upon system reset. Writing this register also sets the Random register to the value of its upper bound. Figure 3.2.6 shows the format of the Wired register and Table 3.2.7 describes the Wired register fields.

31	6 5	0
0		Wired
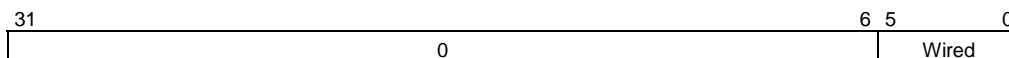
Figure 3.2.6 Wired Register

Table 3.2.7 Wired Register Filed Descriptions

Bit	Field	Description	Cold Reset	Read/Write
31:6	0	Reserved (Must be written as zeroes, and returns zeroes when read.)	0x0	Read
5:0	Wired	TLB Wired boundary.	0x0	Read/Write

3.2.7   BadVAddr Register (Reg#8)

The Bad Virtual Address (BadVAddr) register is a read only register that displays the most recent virtual address that cause one of the following exceptions; Address Error, TLB Invalid, TLB Modified and TLB Refill exceptions.

The processor does not write to this register when the EXL bit in the Status register is set to a 1. Figure 3.2.7 shows the formats of the BadVAddr register and Table 3.2.8 describes the BadVAddr register fields.

```
31                                                                    0
+---------------------------------------------------------------------+
|                        Bad Virtual Address                          |
+---------------------------------------------------------------------+
                         (32-bit mode)


63                                                                    0
+---------------------------------------------------------------------+
|                        Bad Virtual Address                          |
+---------------------------------------------------------------------+
                         (64-bit mode)
```
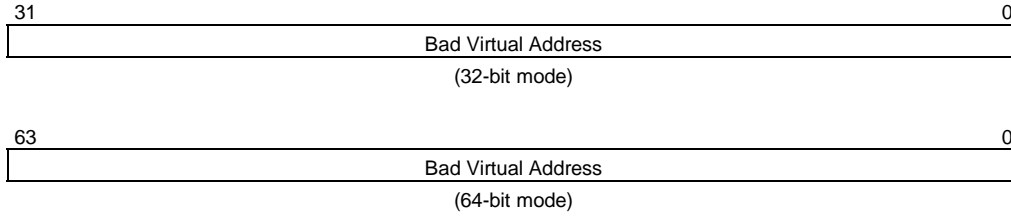
Figure 3.2.7 BadVAddr Register Formats

Table 3.2.8 BadVAddr Register Field Descriptions

32-bit mode

Bit	Field	Description	Cold Reset	Read/Write
31:0	BadVAddr	Bad Virtual address	Undefined	Read

64-bit mode

Bit	Field	Description	Cold Reset	Read/Write
63:0	BadVAddr	Bad Virtual address	Undefined	Read

3.2.8 Count Register (Reg#9)

The Count register is a read/write register. This register acts as a timer, incrementing at a constant rate (1/2 rate of CPUCLK) whether or not an instruction is executed, retired, or any forward progress is made through the pipeline.

This register can be also written for diagnostic purpose or system initialization. Figure 3.2.8 shows the format of the Count register and Table 3.2.9 describes the Count register field.

31   0
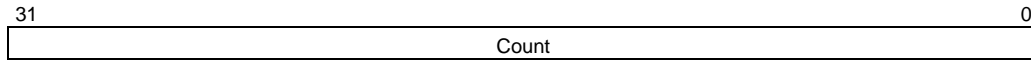
Count

Figure 3.2.8 Count Register Format

Table 3.2.9 Count Register Field Description

Bit	Field	Description	Cold Reset	Read/Write
31:0	Count	32-bit timer, incrementing at half the maximum instruction issue rate (CPUCLK).	0x0	Read/Write

3.2.9 EntryHi Register (Reg#10)

The EntryHi is a read/write register, and holds the high-order bits of a TLB entry for TLB read and write operations. This register is accessed by the TLB Probe (TLBP), TLB Write Ransom (TLBWR), TLB Write Indexed (TLBWI), and TLB Read Indexed (TLBR) instructions.

When either a TLB refill, TLB invalid, or TLB modified exception occurs, this register is loaded with the virtual page number (VPN2) and the ASID of the virtual address that did not have a matching TLB entry. Figure 3.2.9 shows the formats of the EntryHi register and Table 3.2.10 describes the EntryHi register fields.

```
31                                                    13 12        8 7            0
┌──────────────────────────────────────────────────┬───────────┬──────────────┐
│                      VPN2                          │     0     │     ASID     │
└──────────────────────────────────────────────────┴───────────┴──────────────┘
                              (32-bit mode)


63     62 61              40 39              13 12        8 7            0
┌────┬──────────────┬──────────────────────┬───────────┬──────────────┐
│ R  │     FILL     │        VPN2          │     0     │     ASID     │
└────┴──────────────┴──────────────────────┴───────────┴──────────────┘
                              (64-bit mode)
```
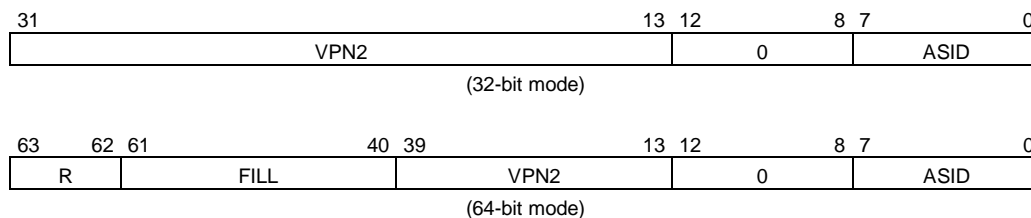
Figure 3.2.9 EntryHi Register Formats

Table 3.2.10 EntryHi Register Field Descriptions

32-bit mode

Bit	Field	Description	Cold Reset	Read/Write
31:1	VPN2	Virtual page number divided by two	Undefined	Read/Write
12:8	0	Reserved	0x0	Read
7:0	ASID	Address space ID field An 8-bit field that lets multiple processes share the TLB; each process has a distinct mapping of otherwise identical virtual page numbers.	Undefined	Read/Write

64-bit mode

Bit	Field	Description	Cold Reset	Read/Write
63:62	R	Region. Used to match vAddr63 and vAddr62. 00: user, 01: supervisor, 11: kernel	Undefined	Read/Write
61:40	Fill	Reserved. 0 on read. Ignored on write.	Undefined	Read
39:13	VPN2	Virtual page number divided by two	Undefined	Read/Write
12:8	0	Reserved	0x0	Read
7:0	ASID	Address space ID field.	Undefined	Read/Write

3.2.10 Compare Register (Reg#11)

The Compare register acts as a timer. When value of the Count register equals the value of the Compare register, interrupt bit IP[7] in the Cause register is set. This causes an interrupt exception as soon as the interrupt is enabled. Writing a value to this register, as a side effect, clears the timer interrupt.

For diagnostic purpose, this register is a read/write register. However, in normal operation this register is write only. Figure 3.2.10 shows the format of the Compare register and Table 3.2.11 describes the Compare register field.

```
31                                                                          0
 ┌────────────────────────────────────────────────────────────────────────┐
 │                              Compare                                     │
 └────────────────────────────────────────────────────────────────────────┘
```
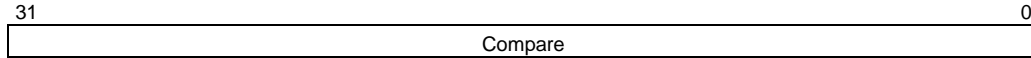
Figure 3.2.10 Compare Register Format

Table 3.2.11 Compare Register Field Description

Bit	Field	Description	Cold Reset	Read/Write
31:0	Compare	Acts as a timer; it maintains a stable value that does not change on its own.	0x0	Read/Write

3.2.11 Status Register (Reg#12)

The Status register is a read/write register that contains the operating mode, interrupt enabling, and diagnostic states of the processor. The more important Status register fields are as following;

- The Interrupt Mask (IM) field of 8 bits controls the enabling of eight interrupt conditions. Interrupt must be enabled before they can be asserted, and the corresponding bits are set in both the IM field of this register and the Interrupt Pending field of the Cause register.

- The Coprocessor Usability (CU) field of 4 bits controls the usability of four possible coprocessors. Regardless of the CU0 bit setting, CP0 is always usable in Kernel mode.

- The Diagnostic Status (DS) field of 9 bits is used for self-testing, and checks the cache and virtual memory system.

- The Reverse Endian (RE) bit reverses the endianness. The processor can be configured as either little/big-endian at reset; reverse-endian selection is used in Kernel and Supervisor modes, and in the User mode when the RE bit is 0. Setting the RE bit to 1 inverts the User mode endianness.

Figure 3.2.11 shows the format of the Status register and Table 3.2.12 describes the Status register field.
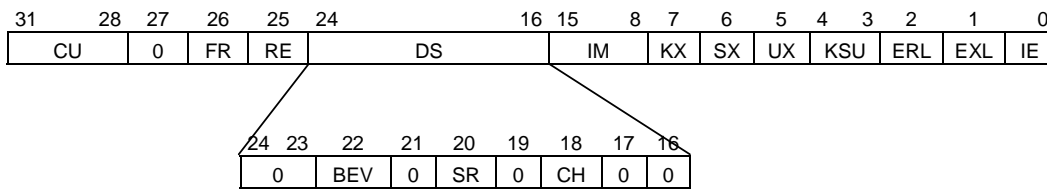
Figure 3.2.11 Status Register Format

Table 3.2.12 Status Register Field Descriptions

Bit	Field	Description	Cold Reset	Read/Write
31:28	CU (3,2,1,0)	Controls the usability of each of the four coprocessor unit numbers. CP0 is always usable when in Kernel mode, regardless of the setting of the CU0 bit. 0: unusable, 1: usable	0000	Read/Write
27	0	Reserved	0	Read
26	FR	Enables additional floating-point registers (Reserved). 0: 16 registers, 1: 32 registers	0	Read
25	RE	Reverse-Endian bit, valid in User mode.	0	Read/Write
24:23	0	Reserved	0x0	Read
22	BEV	Controls the location of TLB refill and general exception vectors. 0: normal, 1: bootstrap	1	Read/Write

Bit	Field	Description	Cold Reset	Read/Write
21	0	Reserved	0	Read
20	SR	1: Indicates NMI has occurred.	0	Read/Write
19	0	Reserved	0	Read
18	CH	"Hit" or "miss" indication for last CACHE Hit Invalidate, Hit Write Back Invalidate, Hit Write Back for a primary cache. 0: miss, 1: hit.	0	Read/Write
17:16	0	Reserved	0x0	Read
15:8	IM	Interrupt Mask Controls the enabling of each of the external, internal and software interrupts. An interrupt is taken if interrupts are enabled, and the corresponding bits are set in both the IM field of the Status register and the IP field of the Cause register. 0: disabled, 0: enabled	0x0	Read/Write
7	KX	Enables 64-bit addressing in Kernel mode. The extended-addressing TLB refill exception is used for TLB misses on kernel addresses. 0: 32-bit, 1: 64-bit	0	Read/Write
6	SX	Enables 64-bit addressing and operations in Supervisor mode. The extended-addressing TLB refill exception is used for TLB misses on supervisor addresses. 0: 32-bit, 1: 64-bit	0	Read/Write
5	UX	Enables 64-bit addressing and operations in User mode. The extended-addressing TLB refill exception is used for TLB misses on user addresses. 0: 32-bit, 1: 64-bit	0	Read/Write
4:3	KSU	Mode. 10: user, 01: supervisor, 00: kernel.	0x0	Read/Write
2	ERL	Error Level. 0: normal, 1: error.	1	Read/Write
1	EXL	Exception Level. 0: normal, 1: exception.	0	Read/Write
0	IE	Interrupt Enable. 0: disable, 1: enable.	0	Read/Write

Status Register Modes and Access States

Fields of the Status register set the modes and access states described in the section that follow.

- Interrupt Enable: Interrupts are enabled when all of the following conditions are met:
    - IE = 1
    - EXL = 0
    - ERL = 0

If these conditions are met, the settings of the IM bits enable the interrupt.

■ Operation Modes: The following CPU Status register bit settings are required for User, Kernel and Supervisor modes (see Section 4.2, *Operation Modes*, for more information about operating modes).

- The processor is in User mode when KSU = 10_2, EXL = 0, and ERL = 0.
- The processor is in Supervisor mode when KSU = 01_2, EXL = 0 and ERL = 0.
- The processor is in Kernel mode when KSU = 00_2, or EXL= 1, or ERL =1.

■ 32- and 64-bit Modes: The following CPU Status register settings select 32- or 64-bit operation for User, Kernel, and Supervisor operating modes. Enabling 64-bit operation permits the execution of 64-bit opcodes and translation of 64-bit addresses. 64-bit operation for User, Kernel and Supervisor modes can be set independently.

- 64-bit addressing for Kernel mode is enabled when KX = 1. 64-bit operations are always valid in Kernel mode.
- 64-bit addressing and operations are enabled for Supervisor mode when SX = 1.
- 64-bit addressing and operations are enabled for User mode when UX = 1.

■ Kernel Address Space Accesses: Access to the kernel address space is allowed when the processor is in Kernel mode.

■ Supervisor Address Space Accesses: Access to the supervisor address space is allowed when the processor is in Kernel or Supervisor mode, as described above in the section above titled *Operating Modes*.

■ User Address Space Accesses: Access to the user address is allowed in any of the three operating modes.

Status Register Reset

The contents of the Status register are undefined at reset, except for the following bits in the Diagnostic Status field:

- ERL and BEV = 1

The SR bit distinguishes between the Reset exception and the Soft Reset exception (caused by Nonmaskable Interrupt [NMI]).

3.2.12 Cause Register (Reg#13)

The Cause register holds the cause of the most recent exception. This register is read-only, except for the IP[1:0] bits. Figure 3.2.12 shows the format of the Cause register and Table 3.2.13 describes the Cause register field.

31	30	29	28	27	16	15	8	7	6	2	1	0
BD	0	CE		0		IP		0	ExcCode		0	
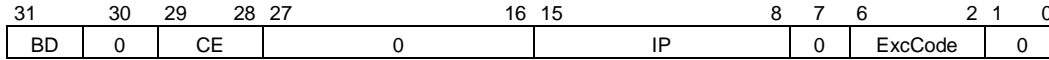
Figure 3.2.12 Cause Register Format

Table 3.2.13 Cause Register Field Descriptions

Bit	Field	Description	Cold Reset	Read/Write
31	BD	Indicates whether or not the last exception was taken while executing in a branch delay slot. 0: normal, 1: delay slot.	0	Read
30	0	Reserved	0	Read
29:28	CE	Indicates the coprocessor unit number referenced when a coprocessor unusable exception is taken. 00: coprocessor 0, 01: coprocessor 1, 10: coprocessor 2, 11: coprocessor 3.	0x0	Read
27:16	0	Reserved	0x0	Read
15:14	IP[7:6]	Reserved	0x0	Read
13:10	IP[5:2]	Indicates whether an interrupt is pending. 0: not pending, 1: pending.	INT[3:0]	Read
9:8	IP[1:0]	Software interrupts. 0: reset, 1: set.	0x0	Read/Write
7	0	Reserved	0	Read
6:2	ExcCode	Exception Code field. 0: Int: Interrupt. 1: Mod: TLB modification exception. 2: TLBL: TLB exception (load or instruction fetch) 3: TLBS: TLB exception (Store) 4: AdEL: Address error exception (load or instruction fetch) 5: AdES: Address error exception (store) 6: IBE: Bus error exception (instruction fetch) 7: DBE: Bus error exception (data reference: load) 8: Sys: Syscall exception 9: Bp: Breakpoint exception 10: RI: Reserved instruction exception 11: CpU: Coprocessor Unusable exception 12: Ov: Arithmetic Overflow exception 13: Tr: Trap exception 14: Reserved: 15: FPE: Floating-Point exception (Reserved) 16-31: Reserved :	0x0	Read
1:0	0	Reserved	0x0	Read

3.2.13 EPC Register (Reg#14)

The Exception Program Counter (EPC) register is a read/write register. This register contents the address at which processing resumes after an exception has been serviced.

For synchronous exceptions, this register contains either;

- the virtual address of the instruction that was the direct cause of the exception.
- the virtual address of the immediately preceding branch or jump instruction (when the instruction is in a branch delay slot, and the Branch Delay bit in the Cause register is set).

The processor does not write to the EPC register when EXL bit in the Status register is set to 1. Figure 3.2.13 shows the formats of the EPC register and Table 3.2.14 describes the EPC register field.

```
31                                                                              0
┌──────────────────────────────────────────────────────────────────────────────┐
│                                    EPC                                         │
└──────────────────────────────────────────────────────────────────────────────┘
                                 (32-bit mode)

63                                                                              0
┌──────────────────────────────────────────────────────────────────────────────┐
│                                    EPC                                         │
└──────────────────────────────────────────────────────────────────────────────┘
                                 (64-bit mode)
```

Figure 3.2.13 EPC Register Formats

Table 3.2.14 EPC Register Field Description

32-bit mode

Bit	Field	Description	Cold Reset	Read/Write
31:0	EPC	Exception program counter	Undefined	Read/Write

64-bit mode

Bit	Field	Description	Cold Reset	Read/Write
63:0	EPC	Exception program counter	Undefined	Read/Write

3.2.14 PRId Register (Reg#15)

The Processor Revision Identifier (PRId) register is a read-only register. This register contents information identifying the implementation and revision level of the CPU and CP0. Figure 3.2.14 shows the format of the PRId register and Table 3.2.15 describes the PRId register field.

31	16	15	8	7	0
0		Imp		Rev	
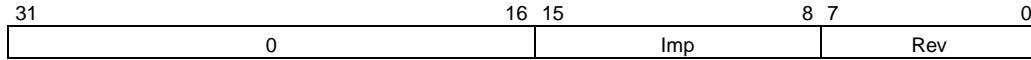
Figure 3.2.14 PRId Register Format

Table 3.2.15 PRId Register Field Descriptions

Bit	Field	Description	Cold Reset	Read/Write
31:16	0	Reserved	0x0	Read
15:8	Imp	Implementation number 0x2d means "TX49 family".	0x2d	Read
7:0	Rev	Revision number +.	+	Read

+ Value is shown in product sheet

3.2.15 Config Register (Reg#16)

The Config register is a read-only register; except for HALT, ICE#, DCE# and K0 fields. This register specifies various configuration options selected on the TX49.

EC, BE, IC, DC, IB and DB fields are set by the hardware during reset and are included in this register as read-only status bits for the software to access. Figure 3.2.15 shows the format of the Config register and Table 3.2.16 describes the Config register field.

31	30	28 27	24 23	19	18	17	16	15	14 13	12 11	9 8	6 5	4	3 2	0
0	EC	0	0	HALT	ICE#	DCE#	BE	1	0	IC	DC	IB	DB	0	K0
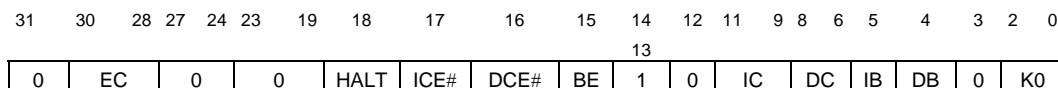
Figure 3.2.15 Config Register Format

Table 3.2.16 Config Register Field Descriptions

Bit	Field	Description	Cold Reset	Read/Write
31	0	Reserved	0	Read
30:28	EC	GBUS clock rate: 0: processor clock frequency divided by 2 1: processor clock frequency divided by 3 2: processor clock frequency divided by 4 7: processor clock frequency divided by 2.5 3, 4, 5, 6 : reserved	pin	Read
27:19	0	Reserved	0	Read
18	HALT	Wait mode. 0: Halt 1: Doze Indicates the power-down behavior of the TX49 when WAIT instruction is executed. The TX49 stalls the pipeline both in halt and doze mode. Cache snoops are possible during Doze mode but not possible during Halt mode. Halt mode reduces power consumption to a greater extent than Doze mode.	0	Read/Write
17	ICE#	Instruction Cache Enable 0: Instruction cache enable 1: Instruction cache disable	0	Read/Write
16	DCE#	Data Cache Enable 0: Data cache enable 1: Data cache disable	0	Read/Write
15	BE	Big Endian 0: Little Endian 1: Big Endian	pin	Read
14:13	1	Reserved	11	Read
12	0	Reserved	0	Read

Bit	Field	Description	Cold Reset	Read/Write
11:9	IC	Instruction cache size. In the TX4951B, this is set to 16 KB (010).	010	Read
8:6	DC	Data cache size. In the TX4951B, this is set to 8 KB (001).	001	Read
5	IB	Primary I-Cache line Size 1: 32 bytes (8 words)	1	Read
4	DB	Primary D-cache line Size 1: 32 bytes (8 words)	1	Read
3	0	Reserved	0	Read
2:0	K0	kseg0 coherency algorithm 0: Cacheable, noncoherent, write-through, no-WA 1: Cacheable, noncoherent, write-through, WA 2: Uncached 3: Cacheable, noncoherent, write-back, WA 4-7: Reserved	0x0	Read/Write

3.2.16 LLAddr Register (Reg#17)

The Load Linked Address (LLAddr) register is a read/wirte register, and contains the physical address read by the most recent Load Linked (LL/LLD) instruction. This register is for diagnostic purposes only, and serves no function during normal operation. Figure 3.2.16 shows the format of the LLAddr register and Table 3.2.17 describes the LLAddr register field.

```
31                                                                          0
┌────────────────────────────────────────────────────────────────────────┐
│                            PAddr[35:4]                                    │
└────────────────────────────────────────────────────────────────────────┘
```
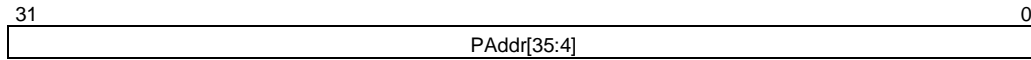
Figure 3.2.16 LLAddr Register Format

Table 3.2.17 LLAddr Register Field Description

Bit	Field	Description	Cold Reset	Read/Write
31:0	pAddr	Physical address bits 35-4	0x0	Read/Write

3.2.17 XContext Register (Reg#20)

The XContext register is a read/write register, and contains a pointer to an entry in the page table entry (PTE) array, an operating system data structure that stores virtual to physical address translations. When there is a TLB miss, the operating system software loads the TLB with the missing translation from the PTE array. However the contents of this register duplicates some information of the BadVAddr register, it is arranged in a form that is more useful for TLB exception handler by a software. This register is for use with the XTLB refill handler, which loads TLB entries for references to a 64-bit address space, and is included solely for operating system use. The operating system sets the PTE base field in the register, as needed. Normally, the operating system uses this register to address the current page map which resides in the Kernel mapped segment, kseg3.

The BadVPN2 field of 27 bits has bits 39-13 of the virtual address that caused the TLB miss; bit 12 is excluded because a single TLB entry maps to an even-odd page pair. For a 4-KByte page size, this format may be used directly to access the pair-table of 8-Byte PTEs. For other page sizes and PTE sizes, shifting and masking this value produces the appropriate address.

Figure 3.2.17 shows the format of the XContext register and Table 3.2.18 describes the XContext register field.

63   33 32 31 30   4 3 0
PTEBase	R	BadVPN2	0
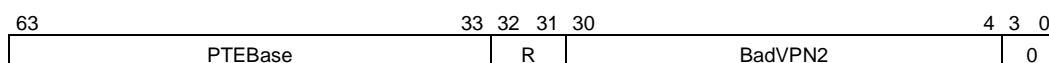
Figure 3.2.17 XContext Register Format

Table 3.2.18 XContext Register Field Description

Bit	Field	Description	Cold Reset	Read/Write
63:33	PTEBase	Page table entry base pointer This field is normally written with a value that allows the operation system to use the Context register as a pointer into the current PTE array in memory.	Undefined	Read/Write
32:31	R	The Region field contains bits 63 to 62 of the virtual address. 00: user, 01: supervisor, 11: kernel	Undefined	Read/Write
30:4	BadVPN2	Bad virtual page number divided by two. This field is written by hardware on a miss. It contains the VPN of the most recent invalidly translated virtual address.	Undefined	Read
3:0	0	Reserved	0x0	Read

3.2.18 Debug Register (Reg#23)

The Debug register is a read-only; except for TLF, BsF, SSt and JtagRst fields. This register holds the information for debug handler. Figure 3.2.18 shows the format of the Debug register and Table 3.2.19 describes the Debug register field.

31	30	29 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBD	DM	0	NIS	TRS	OES	TLF	BsF	0	SSt	JtagRst	0	DINT	DIB	DDBS	DDBL	DBp	DSS
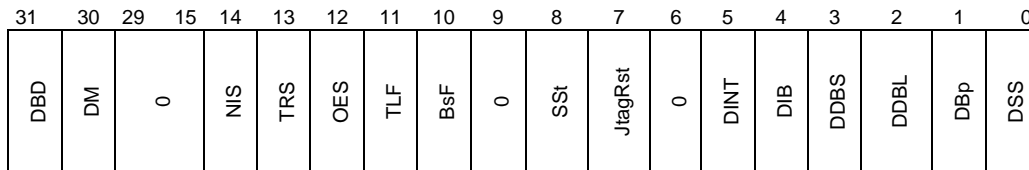
Figure 3.2.18 Debug Register Format

Table 3.2.19 Debug Register Field Descriptions

Bit	Field	Description	Cold Reset	Read/Write
31	DBD	Debug Branch Delay; When a debug exception occurs while an instruction in the branch delay slot is executing, this bit is set to 1.	0	Read
30	DM	Debug Mode; It indicates that a debug exception has taken place. This bit is set when a debug exception is taken, and is cleared upon return from the exception (DERET). While this bit is set all interrupts, including NMI, TLB exception , BUS error exception, and debug exception are masked and cache line locking function is disabled. 0: Debug handler not running. 1: Debug handler running.	0	Read
29:15	0	Reserved	0x0	Read
14	NIS	Non-maskable Interrupt Status; When this bit is set indicating that a non-maskable interrupt has occurred at the same time as a debug exception. In this case the Status, Cause, EPC, and BadVAddr registers assumes the usual status after occurrence of a non-maskable interrupt, but the address in DEPC is not the non-maskable exception vector address (0xbfc0 0000). Instead, 0xbfc0 0000 is put in DEPC by the debug handler software after which processing returns directly from the debug exception to the non-maskable interrupt handler.	0	Read
13	TRS	TLB Miss Status; When this bit is set indicating the Debug Exception and TLB/XTLB refill exception has occurred at the same time. In this case the Status, Cause, EPC, and BadVAddr registers assumes the usual status after occurrence of TLB/XTLB refill. The address in the DEPC is not the other exception vector address. Instead, 0xbfc0 0200 (if BEV = 1) in case of TLB refill exception and 0xbfc0 0280 (if BEV = 1) in case of XTLB refill exception or 0x8000 0000 (if BEV = 0) in case of TLB refill exception and 0x8000 0080 (if BEV = 0) in case of XTLB refill exception is put in DEPC by the debug exception handler software, after which processing returns directly from the debug exception to the other exception handler.	0	Read

Bit	Field	Description	Cold Reset	Read/Write
12	OES	Other Exception Status; When this bit is set indicates exception other than reset, NMI, or TLB/XTLB refill has occurred at the same time as a debug exception. In this case the Status, Cause, EPC, and BadVAddr registers assume the usual status after occurrence of such an exception, but the addressing the DEPC is not the other exception Vector address. Instead, 0xbfc0 0380 (if BEV = 1) or 0x8000 0180 (if BEV = 0) is put in DEPC by the debug exception handler software, after which processing returns directly from the other exception handler.	0	Read
11	TLF	TLB Exception Flag; This bit is set to 1 when TLB related exception occurs for immediately preceding load or store instruction while a debug exception handler is running (DM = 1). TLB exception will set this bit to 1 regardless of writing zero. It is cleared by writing 0 and writing 1 is ignored.	0	Read/Write
10	BsF	Bus Error Exception Flag; This bit is set to 1 when a bus error exception occurs for a load or store instruction while a debug exception handler is running (DM = 1). Bus error exception will set this bit to 1 regardless of writing zero. It is cleared by writing 0 and writing 1 is ignored.	0	Read/Write
9	0	Reserved	0	Read
8	SSt	Single Step; Set to 1 indicates the single step debug function is enable (1) or disabled (0). The function is disable when the DM bit is set to 1 while the debug exception is running.	0	Read/Write
7	JtagRst	JTAG Reset; When this bit is set to 1 the processor reset the JTAG unit.	0	Read/Write
6	0	Reserved	0	Read
5	DINT	Debug Interrupt Break Exception Status; set to 1 when debug interrupts occurs.	0	Read
4	DIB	Debug Instruction Break Exception Status; Set to 1 on instruction address break.	0	Read
3	DDBS	Debug Data Break Store Exception Status; Set to 1 on data address break at store operation.	0	Read
2	DDBL	Debug Data Break Load Exception Status; Set to 1 on data address break at load operation.	0	Read
1	DBp	Debug Breakpoint Exception Status; This bit is set when executing SDBBP instruction.	0	Read
0	DSS	Debug Single Step Exception Status; Set to 1 indicate Single Step Exception.	0	Read

3.2.19 DEPC Register (Reg#24)

The DEPC register holds the address where processing resumes after the debug exception routine has finished. The address that has been loaded in the DEPC register is the virtual address of the instruction that caused the debug exception. If the instruction is in the branch delay slot, the virtual address of the immediately preceding branch or jump instruction is placed in this register. Execution of the DERET instruction causes a jump to the address in the DEPC. If the DEPC is both written from software (by MTC0) and by hardware (debug exception) then the DEPC is loaded by the value generated by the hardware.

Figure 3.2.19 shows the formats of the DEPC register and Table 3.2.20 describes the DEPC register field.

```
31                                                                    0
┌─────────────────────────────────────────────────────────────────────┐
│                              DEPC                                     │
└─────────────────────────────────────────────────────────────────────┘
                          (32-bit mode)


63                                                                    0
┌─────────────────────────────────────────────────────────────────────┐
│                              DEPC                                     │
└─────────────────────────────────────────────────────────────────────┘
                          (64-bit mode)
```
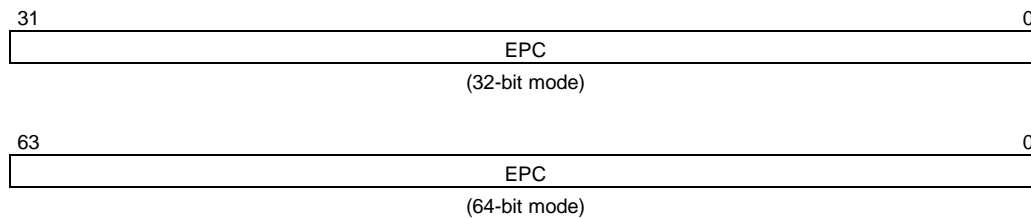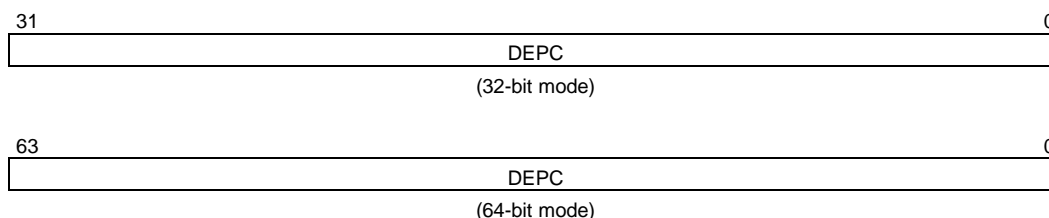
Figure 3.2.19 DEPC Register Formats

Table 3.2.20 DEPC Register Field Description

32-bit mode

Bit	Field	Description	Cold Reset	Read/Write
31:0	DEPC	Debug exception program counter.	Undefined	Read/Write

64-bit mode

Bit	Field	Description	Cold Reset	Read/Write
63:0	DEPC	Debug exception program counter.	Undefined	Read/Write

3.2.20 TagLo Register (Reg#28) and TagHi Register (Reg#29)

The TagLo and TagHi registers are a read/write registers. These registers hold the primary cache tag for cache lock function or cache diagnostics. These registers are written by the CACHE/MTC0 instruction. Figure 3.2.20 shows the formats of the TagLo and TagHi registers and Table 3.2.21 describes the TagLo and TagHi registers field.

```
31                                               8 7    6 5    3  2     1    0
┌───────────────────────────────────────────────┬──────┬──────┬──────┬────┬────┐
│                 PTagLo                          │PState│ RWNT │ Lock │ F0 │ 0  │
└───────────────────────────────────────────────┴──────┴──────┴──────┴────┴────┘
                              (TagLo)
```

```
31   30    29                                                            0
┌────┬────────┬───────────────────────────────────────────────────────────┐
│ F1 │ PtagLo1│                          0                                  │
└────┴────────┴───────────────────────────────────────────────────────────┘
                              (TagHi)
```
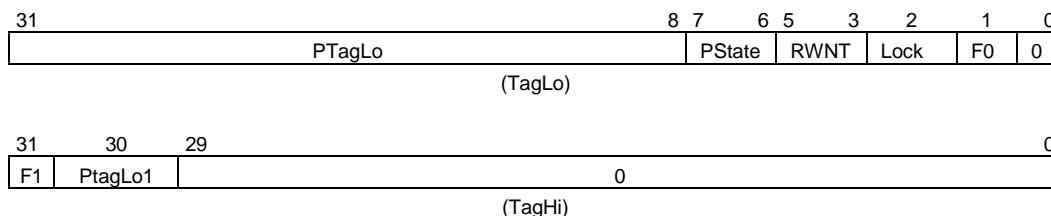
Figure 3.2.20 TagLo and TagHi Register Formats

Table 3.2.21 TagLo and TagHi Register Field Descriptions

TagLo

Bit	Field	Description	Cold Reset	Read/Write
31:8	PTagLo	Bits 35:12 of the physical address	0x0	Read/Write
7:6	PState	Specifies the primary cache state 0: Invalid 1: Reserved 2: Reserved 3: Valid	0x0	Read/Write
5:3	RWNT	Read/Write bits required for Windows NT	0x0	Read/Write
2	Lock	Lock bit (0: not locked, 1: locked)	0	Read/Write
1	F0	FIFO Replace bit 0 (indicates the set to be replaced)	0	Read/Write
0	0	Reserved	0	Read

TagHi

Bit	Field	Description	Cold Reset	Read/Write
31	F1	FIFO Replace bit 1 (indicates the set to be replaced)	0	Read/Write
30	PTagLo1	Bit 11 of the physical address	0	Read/Write
29:0	0	Reserved	0x0	Read

F1 and F0 are concatenated and indicate the set to be replaced.

F1 ││ F0

0   0 : way0

0   1 : way1

1   0 : way2

1   1 : way3

3.2.21 ErrorEPC Register (Reg#30)

The ErrorEPC is a read/write register, and is similar to the EPC register. This register is used to store the program counter (PC) on ColdReset, SoftReset and NMI exceptions.

This register contains the virtual address at which instruction processing can resume after servicing an error. This address can be;

- The virtual address of the instruction that caused the exception
- The virtual address of the immediately preceding branch or jump instruction, when this address is in a branch delay slot.

There is no branch delay slot indication for this register. Figure 3.2.21 shows the formats of the ErrorEPC register and Table 3.2.22 describes the ErrorEPC register field.

```
31                                                                          0
┌────────────────────────────────────────────────────────────────────────┐
│                              ErrorEPC                                    │
└────────────────────────────────────────────────────────────────────────┘
                              (32-bit mode)


63                                                                          0
┌────────────────────────────────────────────────────────────────────────┐
│                              ErrorEPC                                    │
└────────────────────────────────────────────────────────────────────────┘
                              (64-bit mode)
```
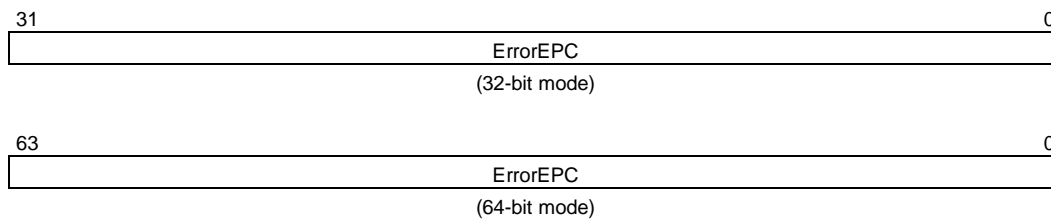
Figure 3.2.21 ErrorEPC Register Formats

Table 3.2.22 ErrorEPC Register Field Descriptions

32-bit mode

Bit	Field	Description	Cold Reset	Read/Write
31:0	ErrorEPC	Error Exception Program Counter.	Undefined	Read/Write

64-bit mode

Bit	Field	Description	Cold Reset	Read/Write
63:0	ErrorEPC	Error Exception Program Counter.	Undefined	Read/Write

3.2.22 DESAVE Register (Reg#31)

This register is used by the debug exception handler to save one of the GPRs, that is then used to save the rest of the context to a pre-determined memory are, e.g. in the processor probe. This register allows the safe debugging of exception handlers and other types of code where the existence of a valid stack for context saving cannot be assumed.

Figure 3.2.22 shows the formats of the DESAVE register and Table 3.2.23 describes the DESAVE register field.

Note: This register can use for ICE system only.

63   0
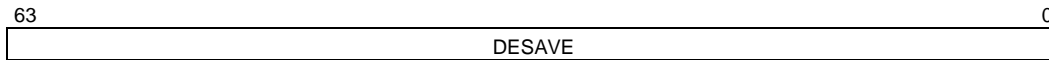
DESAVE

Figure 3.2.22 DESAVE Register Format

Table 3.2.23 DESAVE register Field Description

32-/64-bit mode

Bit	Field	Description	Cold Reset	Read/Write
63:0	DESAVE	Save one of the GPRs	Undefined	Read/Write

4.   Memory Management System

The TX4951B provides a full-featured memory management unit (MMU) which uses an on-chip translation look aside buffer (TLB) to translate virtual addresses into physical addresses.

4.1 Address Space Overview

The TX4951B physical address space is 4 Gbytes using a 32-bit address. The virtual address is either 64 or 32 bits wide depending on whether the processor is operating in 64- or 32-bit mode. In 32-bit mode, addresses are 32-bits wide and the maximum user process size is 2 Gbytes (2^{31}). In 64-bit mode, addresses are 64-bit wide and the maximum user process is 1 Tbytes (2^{40}). The virtual address is extended with an Address Space Identifier (ASID) to reduce the frequency of TLB flushing when switching context. The size of the ASID field is 8 bits. The ASID is contained in the CP0 EntryHi register.

4.1.1   Virtual Address Space

The processor virtual address can be either 32 or 64 bits wide, depending on whether the processor is operating in 32-bit or 64-bit mode.

- In 32-bit mode, addresses are 32 bits wide.

    The maximum user process size is 2 Gbytes (2^{31}).

- In 64-bit mode, addresses are 64 bits wide.

    The maximum user process size is 1 Tbytes (2^{40}).

Figure 4.1.1 shows the translation of a virtual address into a physical address.
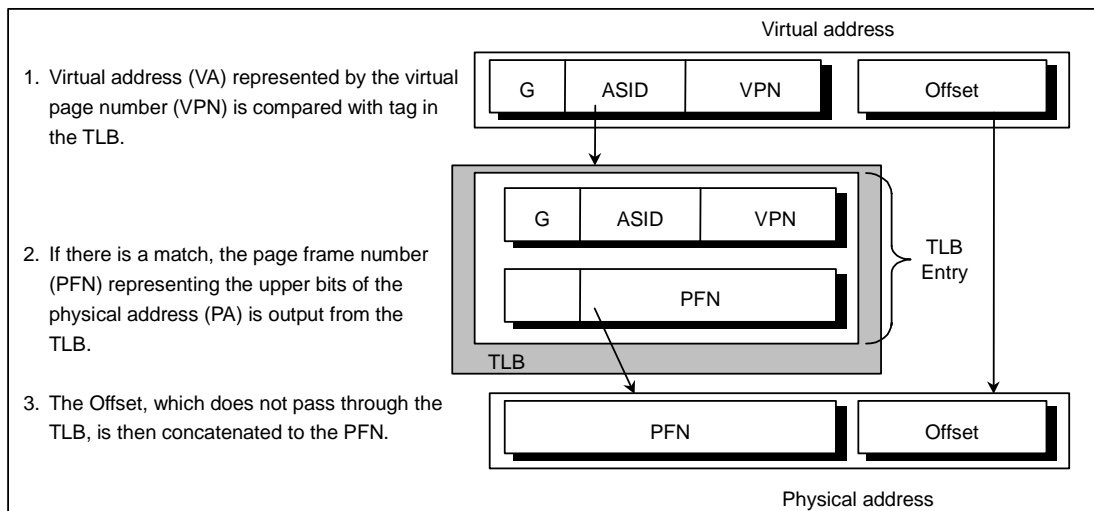
Figure 4.1.1 Overview of a Virtual-to-Physical Address Translation

As shown in Figure 4.1.2 and Figure 4.1.3, the virtual address is extended with an 8-bit address space identifier (ASID), which reduces the frequency of TLB flushing when switching contexts. This 8-bit ASID is in the CP0 *EntryHi* register, described later in this chapter. The *Global* bit (G) is in the *EntryLo0* and *EntryLo1* registers, described later in this chapter.

4.1.2   Physical Address Space

Using a 32-bit address, the processor physical address space encompasses 4 Gbytes. The section following describes the translation of a virtual address to a physical address.

4.1.3   Virtual-to-Physical Address Translation

Converting a virtual address to a physical address begins by comparing the virtual address from the processor with the virtual addresses in the TLB; there is a match when the virtual page number (VPN) of the address is the same as the VPN field of the entry, and either:

- the Global (G) bit of the TLB entry is set, or
- the ASID field of the virtual address is the same as the ASID field of the TLB entry.

This match is referred to as a *TLB hit*. If there is no match, a TLB Miss exception is taken by the processor and software is allowed to refill the TLB from a page table of virtual/physical addresses in memory.

If there is a virtual address match in the TLB, the physical address is output from the TLB and concatenated with the *Offset*, which represents an address within the page frame space. The *Offset* does not pass through the TLB.

Virtual-to-physical translation is described in greater detail throughout the remainder of this chapter; Figure 4.4.1 is a flow diagram of the process shown at the end of this chapter. The next two sections describe the 32-bit and 64-bit address translations.

4.1.4   32-bit Mode Address Translation

Figure 4.1.2 shows the virtual-to-physical-address translation of a 32-bit mode address. This figure illustrates two of the possible page sizes: a 4-Kbyte page (12 bits) and a 16-Mbyte page (24 bits).

- The top portion of Figure 4.1.2 shows a virtual address with a 12-bit, or 4-Kbyte, page size, labeled *Offset*. The remaining 20 bits of the address represent the VPN, and Index the 1M-entry page table.
- The bottom portion of Figure 4.1.2 shows a virtual address with a 24-bit, or 16-Mbyte, page size, labeled *Offset*. The remaining 8 bits of the address represent the VPN, and index the 256-entry page table.
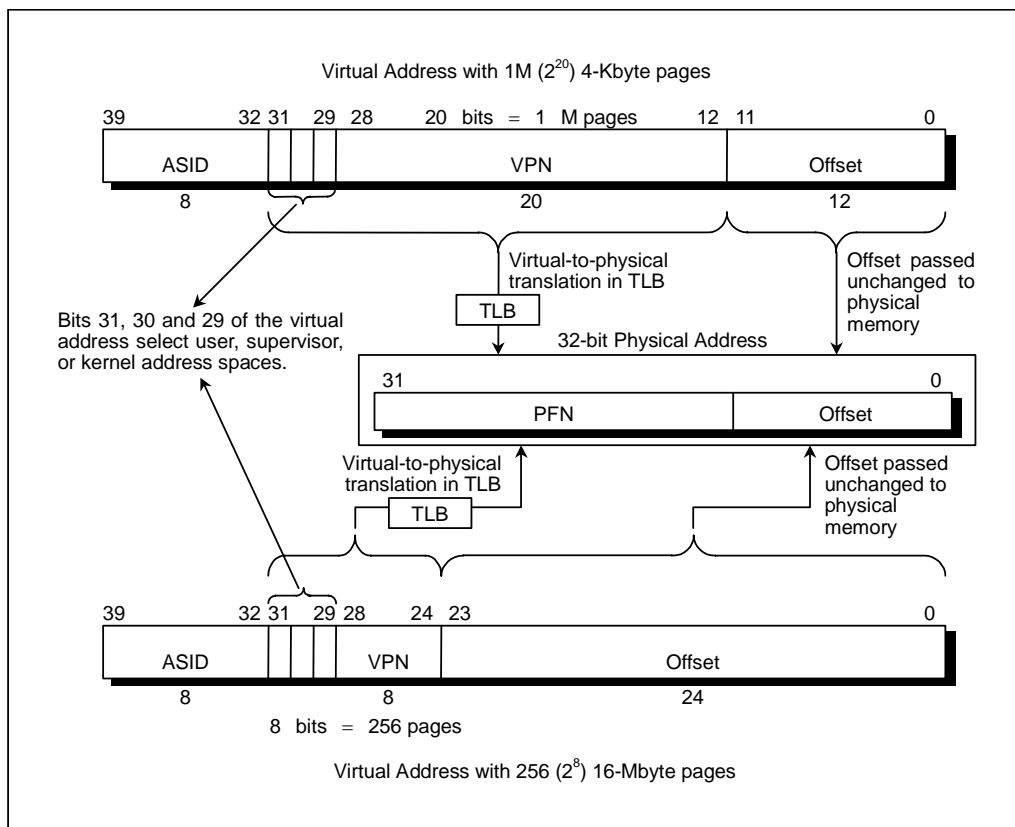
Figure 4.1.2 32-bit Mode Virtual Address Translation

4.1.5   64-bit Mode Address Translation

Figure 4.1.3 shows the virtual-to-physical-address translation of a 64-bit mode address. This figure illustrates two of the possible page sizes: a 4-Kbyte page (12 bits) and a 16-Mbyte page (24 bits).

- The top portion of Figure 4.1.3 shows a virtual address with a 12-bit, or 4-Kbyte, page size, labelled *Offset*. The remaining 28 bits of the address represent the VPN, and index the 256M-entry page table.

- The bottom portion of Figure 4.1.3 shows a virtual address with a 24-bit, or 16-Mbyte, page size, labelled *Offset*. The remaining 16 bits of the address represent the VPN, and index the 64K-entry page table.
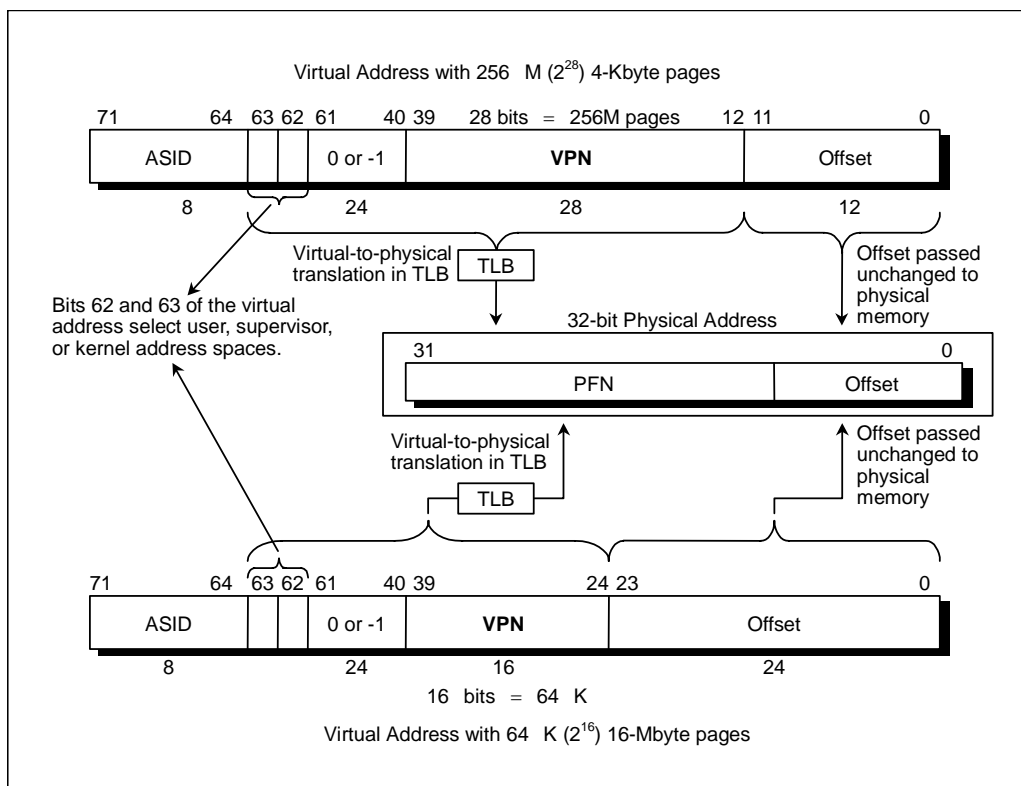
Figure 4.1.3 64-bit Mode Virtual Address Translation

4.2 Operating Modes

The TX49 has the three operating modes, User mode, Supervisor mode and Kernel mode, for 32- and 64-bit operation. The KSU, EXL and ERL bit in the Status register select User, Supervisor or Kernel mode. The UX, SX and KX bit in the Status register select 32- or 64-bit addressing in user, supervisor and kernel mode respectively.

KSU	EXL	ERL	UX	SX	KX	Mode
10	0	0	0	-	-	32-bit addressing in user mode
10	0	0	1	-	-	64-bit addressing in user mode
01	0	0	-	0	-	32-bit addressing in supervisor mode
01	0	0	-	1	-	64-bit addressing in supervisor mode
00	-	-	-	-	0	32-bit addressing in kernel mode
-	1	-	-	-	0	32-bit addressing in kernel mode
-	-	1	-	-	0	32-bit addressing in kernel mode
00	-	-	-	-	1	64-bit addressing in kernel mode
-	1	-	-	-	1	64-bit addressing in kernel mode
-	-	1	-	-	1	64-bit addressing in kernel mode

4.2.1 User Mode Operations

In User mode, a single, uniform virtual address space-labelled User segment-is available; its size is:

- 2 Gbytes (2^{31} bytes) in 32-bit mode (*useg*)
- 1 Tbytes (2^{40} bytes) in 64-bit mode (*xuseg*)

Figure 4.2.1 shows User mode virtual address space.
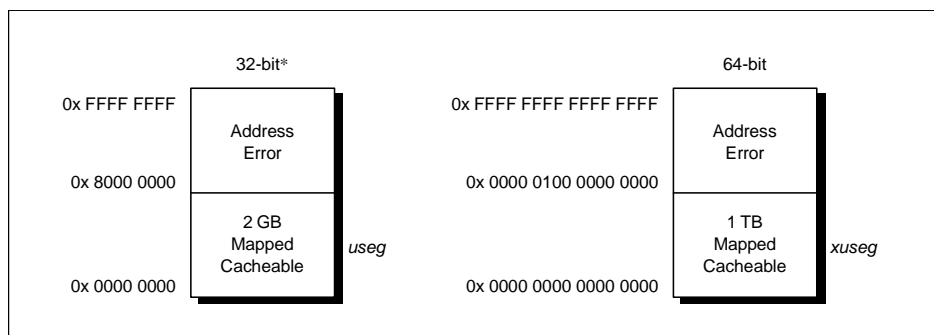
Figure 4.2.1 User Mode Virtual Address Space

*Note: In 32-bit mode, bit 31 is sign-extended through bits 63-32. Failure results in an address error exception.

The User segment starts at address 0 and the current active user process resides in either *useg* (in 32-bit mode) or *xuseg* (in 64-bit mode). The TLB identically maps all references to *useg/xuseg* from all modes, and controls cache accessibility.

The processor operates in User mode when the Status register contains the following bit-values:

- *KSU* bits = 10_2
- *EXL* = 0
- *ERL* = 0

In conjunction with these bits, the *UX* bit in the Status register selects between 32- or 64-bit User mode addressing as follows:

- when *UX* = 0, 32-bit useg space is selected and TLB misses are handled by the 32-bit TLB refill exception handler

- when *UX* = 1, 64-bit xuseg space is selected and TLB misses are handled by the 64-bit TLB refill exception handler

Table 4.2.1 lists the characteristics of the two user mode segments, useg and xuseg.

Table 4.2.1 32-bit and 64-bit User Mode Segments

Address Bit Values	Status Register Bit Values				Segment Name	Address Range	Segment Size
	KSU	EXL	ERL	UX			
32-bit A[31] = 0	10_2	0	0	0	*useg*	0x0000 0000 through 0x7FFF FFFF	2 Gbytes (2^{31} bytes)
64-bit A[63:40] = 0	10_2	0	0	1	*xuseg*	0x0000 0000 0000 0000 through 0x0000 00FF FFFF FFFF	1 Tbytes (2^{40} bytes)

32-bit User Mode (*useg*)

In User mode, when *UX* = 0 in the Status register, User mode addressing is compatible with the 32-bit addressing model shown in Figure 4.2.1, and a 2-Gbyte user address space is available, labelled useg.

All valid User mode virtual addresses have their most-significant bit cleared to 0; any attempt to reference an address with the most-significant bit set while in User mode causes an Address Error exception.

The system maps all references to *useg* through the TLB, and bit settings within the TLB entry for the page determine the cacheability of a reference.

64-bit User Mode (*xuseg*)

In User mode, when *UX* = 1 in the Status register, User mode addressing is extended to the 64-bit model shown in Figure 4.2.1. In 64-bit User mode, the processor provides a single, uniform address space of 2^{40} bytes, labelled xuseg.

All valid User mode virtual addresses have bits 63-40 equal to 0; an attempt to reference an address with bits 63-40 not equal to 0 causes an Address Error exception.

The system maps all reference to *xuseg* through the TLB, and bit settings within the TLB entry for the page determine the cacheability of a reference.

4.2.2   Supervisor Mode Operations

Supervisor mode is designed for layered operating systems in which a true kernel runs in Kernel mode, and the rest of the operating system runs in Supervisor mode.

The processor operates in Supervisor mode when the Status register contains the following bit-values:

- $KSU = 01_2$
- $EXL = 0$
- $ERL = 0$

In conjunction with these bits, the *SX* bit in the Status register selects between 32- or 64-bit Supervisor mode addressing:

- when $SX = 0$, 32-bit supervisor space is selected and TLB misses are handled by the 32-bit TLB refill exception handler
- when $SX = 1$, 64-bit supervisor space is selected and TLB misses are handled by the 64-bit XTLB refill exception handler

The system maps all references through the TLB, and bit settings within the TLB entry for the page determine the cacheability of a reference.

Figure 4.2.2 shows Supervisor mode address mapping. Table 4.2.2 lists the characteristics of the supervisor mode segments; descriptions of the address spaces follow.
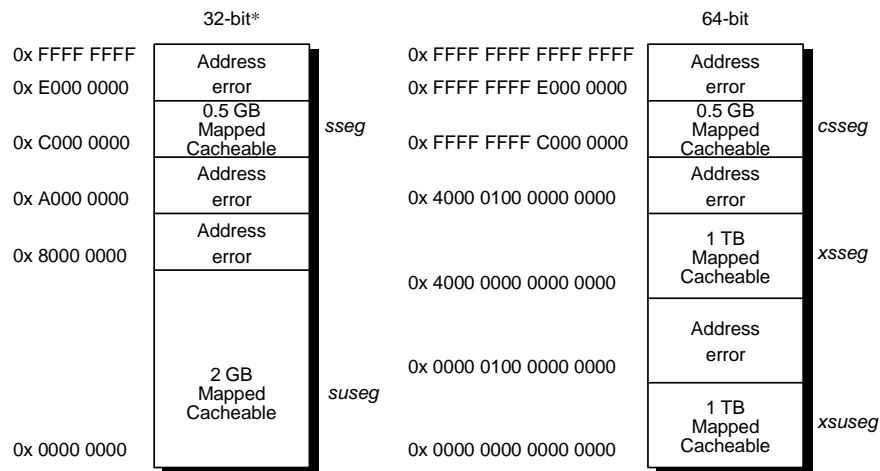
Figure 4.2.2 Supervisor Mode Address Space

*Note: In 32-bit mode, bit 31 is sign-extended through bits 63-32. Failure results in an address error exception.

Table 4.2.2 32-bit and 64-bit Supervisor Mode Segments

Address Bit Values	Status Register Bit Values				Segment Name	Address Range	Segment Size
	KSU	EXL	ERL	SX			
32-bit A[31] = 0	01_2	0	0	0	*suseg*	0x0000 0000 through 0x7FFF FFFF	2 Gbytes (2^{31} bytes)
32-bit A[31:29] = 110_2	01_2	0	0	0	*ssseg*	0xC000 0000 through 0xDFFF FFFF	512 Mbytes (2^{29} bytes)
64-bit A[63:62] = 00_2	01_2	0	0	1	*xsuseg*	0x0000 0000 0000 0000 through 0x0000 00FF FFFF FFFF	1 Tbytes (2^{40} bytes)
64-bit A[63:62] = 01_2	01_2	0	0	1	*xsseg*	0x4000 0000 0000 0000 through 0x4000 00FF FFFF FFFF	1 Tbytes (2^{40} bytes)
64-bit A[63:62] = 11_2	01_2	0	0	1	*csseg*	0xFFFF FFFF C000 0000 through 0xFFFF FFFF DFFF FFFF	512 Mbytes (2^{29} bytes)

32-bit Supervisor Mode, User Space (*suseg*)

In Supervisor mode, when *SX* = 0 in the *Status* register and the most-significant bit of the 32-bit virtual address is set to 0, the *suseg* virtual address space is selected; it covers the full 2^{31} bytes (2 Gbytes) of the current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space starts at virtual address 0x0000 0000 and runs through 0x7FFF FFFF.

32-bit Supervisor Mode, Supervisor Space (*sseg*)

In Supervisor mode, when *SX* = 0 in the *Status* register and the three most-significant bits of the 32-bit virtual address are 110_2, the *sseg* virtual address space is selected; it covers 2^{29} bytes (512 Mbytes) of the current supervisor address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space begins at virtual address 0xC000 0000 and runs through 0xDFFF FFFF.

64-bit Supervisor Mode, User Space (*xsuseg*)

In Supervisor mode, when *SX* = 1 in the *Status* register and bits 63-62 of the virtual address are set to 00_2, the *xsuseg* virtual address space is selected; it covers the full 2^{40} bytes (1 Tbytes) of the current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space starts at virtual address 0x0000 0000 0000 0000 and runs through 0x0000 00FF FFFF FFFF.

64-bit Supervisor Mode, Current Supervisor Space (*xsseg*)

In Supervisor mode, when *SX* = 1 in the *Status* register and bits 63-62 of the virtual address are set to 01_2, the *xsseg* current supervisor virtual address space is selected. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space begins at virtual address 0x4000 0000 0000 0000 and runs through 0x4000 00FF FFFF FFFF.

64-bit Supervisor Mode, Separate Supervisor Space (*csseg*)

In Supervisor mode, when *SX* = 1 in the *Status* register and bits 63-62 of the virtual address are set to 11_2, the *csseg* separate supervisor virtual address space is selected. Addressing of the csseg is compatible with addressing *sseg* in 32-bit mode. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space begins at virtual address 0xFFFF FFFF C000 0000 and runs through 0xFFFF FFFF DFFF FFFF.

4.2.3   Kernel Mode Operations

The processor operates in Kernel mode when the Status register contains one or more of the following values:

- $KSU = 00_2$
- $EXL = 1$
- $ERL = 1$

In conjunction with these bits, the *KX* bit in the Status register selects between 32- or 64-bit Kernel mode addressing:

- when *KX* = 0, 32-bit kernel space is selected and all TLB misses are handled by the 32-bit TLB refill exception handler
- when *KX* = 1, 64-bit kernel space is selected and all TLB misses are handled by the 64-bit XTLB refill exception handler

The processor enters Kernel mode whenever an exception is detected and it remains in Kernel mode until an Exception Return (ERET) instruction is executed and results in ERL and/or EXL = 0. The ERET instruction restores the processor to the mode existing prior to the exception.

Kernel mode virtual address space is divided into regions differentiated by the high-order bits of the virtual address, as shown in Figure 4.2.3. Table 4.2.3 lists the characteristics of the 32-bit kernel mode segments, and Table 4.2.4 lists the characteristics of the 64-bit kernel mode segments.

| 32-bit*			64-bit		
0x FFFF FFFF	0.5 GB Mapped Cacheable	*kseg3*	0x FFFF FFFF FFFF FFFF 0x FFFF FFFF E000 0000	0.5 GB Mapped Cacheable	*ckseg3*
0x E000 0000			0x FFFF FFFF C000 0000	0.5 GB Mapped Cacheable	*cksseg*
0x C000 0000	0.5 GB Mapped Cacheable	*ksseg*	0x FFFF FFFF A000 0000	0.5 GB Unmapped Uncached	*ckseg1*
	0.5 GB Unmapped Uncached	*kseg1*	0x FFFF FFFF 8000 0000	0.5 GB Unmapped Cacheable	*ckseg0*
0x A000 0000			0x C000 00FF 8000 0000	Address error	
0x 8000 0000	0.5 GB Unmapped Cacheable	*kseg0*	0x C000 0000 0000 0000	Mapped Cacheable	*xkseg*
	2 GB Mapped Cacheable	*kuseg*	0x 8000 0000 0000 0000	Unmapped (For details see figure 4.2.4)	*xkphys*
			0x 4000 0100 0000 0000	Address error	
			0x 4000 0000 0000 0000	1 TB Mapped Cacheable	*xksseg*
			0x 0000 0100 0000 0000	Address error	
0x 0000 0000			0x 0000 0000 0000 0000	1 TB Mapped Cacheable	*xkuseg*

Figure 4.2.3 Kernel Mode Address Space

*Note 1: In 32-bit mode, bit 31 is sign-extended through bits 63-32. Failure results in an address error exception.

*Note 2: 0xff00_0000 through 0xff3f_ffff in 32-bit mode and 0xffff_ffff_ff00_0000 through 0xffff_ffff_ff3f_ffff in 64-bit mode are reserved (unmapped, uncached) for use by registers in the Debug Support Unit and TX4951B's peripherals.

0xBFFF FFFF FFFF FFFF

4 × 64 GB
Unmapped
Reserved

0xA000 0000 0000 0000
0x9FFF FFFF FFFF FFFF

64 GB
Unmapped
Cacheable
noncoherent
WB

0x9800 0000 0000 0000
0x97FF FFFF FFFF FFFF

64 GB
Unmapped
Uncached

0x9000 0000 0000 0000
0x8FFF FFFF FFFF FFFF

64 GB
Unmapped
Cacheable
noncoherent
WT-WA

0x8800 0000 0000 0000
0x87FF FFFF FFFF FFFF

64 GB
Unmapped
Cacheable
noncoherent
WT-no-WA

0x8000 0000 0000 0000

Figure 4.2.4 xkphys Address Space

Table 4.2.3 32-bit Kernel Mode Segments

Address Bit Values	Status Register Is One Of These Values				Segment Name	Address Range	Segment Size
	KSU	EXL	ERL	KX			
$A[31] = 0$				0	*Kuseg*	0x0000 0000 through 0x7FFF FFFF	2 Gbytes (2^{31} bytes)
$A[31:29] = 100_2$				0	*Kseg0*	0x8000 0000 through 0x9FFF FFFF	512 Mbytes (2^{29} bytes)
$A[31:29] = 101_2$	KSU = 00_2 or EXL = 1 or ERL = 1			0	*Kseg1*	0xA000 0000 through 0xBFFF FFFF	512 Mbytes (2^{29} bytes)
$A[31:29] = 110_2$				0	*Ksseg*	0xC000 0000 through 0xDFFF FFFF	512 Mbytes (2^{29} bytes)
$A[31:29] = 111_2$				0	*Kseg3*	0xE000 0000 through 0xFFFF FFFF	512 Mbytes-4 Mbytes (2^{29} bytes)
				0	(*Reserved*)	0xFF00 0000 through 0xFF3F FFFF	4 Mbytes

32-bit Kernel Mode, User Space (*kuseg*)

In Kernel mode, when *KX* = 0 in the *Status* register, and the most-significant bit of the virtual address, A31, is cleared, the 32-bit *kuseg* virtual address space is selected; it covers the full 2^{31} bytes (2 Gbytes) of the current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. When ERL = 1 in the *Status* register, the user address region becomes a 2^{31} bytes unmapped (that is, mapped directly to physical addresses) uncached address space.

32-bit Kernel Mode, Kernel Space 0 (*kseg0*)

In Kernel mode, when *KX* = 0 in the *Status* register and the most-significant three bits of the virtual address are 100_2, 32-bit *kseg0* virtual address space is selected; it is the 2^{29} bytes (512 Mbytes) kernel physical space. References to *kseg0* are not mapped through the TLB; the physical address selected is defined by subtracting 0x8000 0000 from the virtual address.   The *K0* field of the *Config* register, described in this chapter, controls cacheability and coherency.

32-bit Kernel Mode, Kernel Space 1 (*kseg1*)

In Kernel mode, when *KX* = 0 in the Status register and the most-significant three bits of the 32-bit virtual address are 101_2, 32-bit *kseg1* virtual address space is selected; it is the 2^{29} bytes (512 Mbytes) kernel physical space. References to *kseg1* are not mapped through the TLB; the physical address selected is defined by subtracting 0xA000 0000 from the virtual address. Caches are disabled for accesses to these addresses, and physical memory (or memory-mapped I/O device registers) are accessed directly.

32-bit Kernel Mode, Supervisor Space (*ksseg*)

In Kernel mode, when *KX* = 0 in the *Status* register and the most-significant three bits of the 32-bit virtual address are 110_2, the *ksseg* virtual address space is selected; it is the current 2^{29} bytes (512 Mbytes) supervisor virtual space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

32-bit Kernel Mode, Kernel Space 3 (*kseg3*)

In Kernel mode, when *KX* = 0 in the *Status* register and the most-significant three bits of the 32-bit vital address are 111_2, the *kseg3* virtual address space is selected; it is the current 2^{29} bytes (512 Mbytes-4 Mbytes) kernel virtual space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

Note: These is the 4 Mbytes Reserved area, begin at virtual address 0xFF00_0000 and runs through 0xFF3F_FFFF.

Table 4.2.4 64-bit Kernel Mode Segments

Address Bit Values	Status Register Is One Of These Values				Segment Name	Address Range	Segment Size
	KSU	EXL	ERL	KX			
$A[63:62] = 00_2$				1	*xkuseg*	0x0000 0000 0000 0000 through 0x0000 00FF FFFF FFFF	1 Tbytes (2^{40} bytes)
$A[63:62] = 01_2$				1	*xksseg*	0x4000 0000 0000 0000 through 0x4000 00FF FFFF FFFF	1 Tbytes (2^{40} bytes)
$A[63:62] = 10_2$				1	*xkphys*	0x8000 0000 0000 0000 through 0xBFFF FFFF FFFF FFFF	8×2^{32} bytes
$A[63:62] = 11_2$				1	*xkseg*	0xC000 0000 0000 0000 through 0xC000 00FF 7FFF FFFF	$2^{40} - 2^{31}$ bytes
$A[63:62] = 11_2$ $A[61:31] = -1$	$KSU = 00_2$ or $EXL = 1$ or $ERL = 1$			1	*ckseg0*	0xFFFF FFFF 8000 0000 through 0xFFFF FFFF 9FFF FFFF	512 Mbytes (2^{29} bytes)
$A[63:62] = 11_2$ $A[61:31] = -1$				1	*ckseg1*	0xFFFF FFFF A000 0000 through 0xFFFF FFFF BFFF FFFF	512 Mbytes (2^{29} bytes)
$A[63:62] = 11_2$ $A[61:31] = -1$				1	*cksseg*	0xFFFF FFFF C000 0000 through 0xFFFF FFFF DFFF FFFF	512 Mbytes (2^{29} bytes)
$A[63:62] = 11_2$ $A[61:31] = -1$				1	*ckseg3*	0xFFFF FFFF E000 0000 through 0xFFFF FFFF FFFF FFFF	512 Mbytes -4 Mbytes
				1	(*Reserved*)	0xFFFF FFFF FF00 0000 through 0xFFFF FFFF FF3F FFFF	4 Mbytes

64-bit Kernel Mode, User Space (*xkuseg*)

In Kernel mode, when $KX = 1$ in the *Status* register and bits 63-62 of the 64-bit virtual address are 00_2, the *xkuseg* virtual address space is selected; it covers the current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

When $ERL = 1$ in the *Status* register, the user address region becomes a 2^{31} bytes unmapped (that is, mapped directly to physical addresses) uncached address space.

64-bit Kernel Mode, Current Supervisor Space (*xksseg*)

In Kernel mode, when $KX = 1$ in the *Status* register and bits 63-62 of the 64-bit virtual address are 01_2, the *xksseg* virtual address space is selected; it is the current supervisor virtual space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

64-bit Kernel Mode, Physical Spaces (*xkphys*)

In Kernel mode, when *KX* = 1 in the *Status* register and bits 63-62 of the 64-bit virtual address are 10_2, one of the two unmapped *xkphys* address spaces are selected, either cached or uncached. Accesses with address bits 58-36 not equal to 0 cause an address error.

References to this space are not mapped; the physical address selected is taken from bits 35-0 of the virtual address. Bits 61-59 of the virtual address specify the cacheability and coherency attributes, as shown in Table 4.2.5.

Table 4.2.5 Cacheability and Coherency Attributes

Value(61-59)	Cacheability and Coherency Attributes	Starting Address
0	Cacheable, non-coherent, write-through, no write allocate	0x8000 0000 0000 0000
1	Cacheable, non-coherent, write-through, no write allocate	0x8800 0000 0000 0000
2	Uncached	0x9000 0000 0000 0000
3	Cacheable, non-coherent	0x9800 0000 0000 0000
4-7	Reserved	0xA000 0000 0000 0000

64-bit Kernel Mode, Kernel Space (*xkseg*)

In Kernel mode, when *KX* = 1 in the *Status* register and bits 63-62 of the 64-bit virtual address are 11_2, the address space selected is one of the following:

- kernel virtual space, *xkseg*, the current kernel virtual space; the virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address
- one of the four 32-bit kernel compatibility spaces, as described in the next section.

64-bit Kernel Mode, Compatibility Spaces (*ckseg1~0, cksseg, ckseg3*)

In Kernel mode, when *KX* = 1 in the *Status* register, bits 63-62 of the 64-bit virtual address are 11_2, and bits 61-31 of the virtual address equal-1, the lower two bytes of address, as shown in Figure 4.2.3, select one of the following 512 Mbytes compatibility spaces.

- *ckseg0.* This 64-bit virtual address space is an unmapped region, compatible with the 32-bit address model *kseg0*. The *K0* field of the *Config* register, described in this chapter, controls cacheability and coherency.
- *ckseg1.* This 64-bit virtual address space is an unmapped and uncached region, compatible with the 32-bit address model *kseg1*.
- *cksseg.* This 64-bit virtual address space is the current supervisor virtual space, compatible with the 32-bit address model *ksseg*.
- *ckseg3.* This 64-bit virtual address space is kernel virtual space, compatible with the 32-bit address model *kseg3*.

4.3 Translation Lookaside Buffer

4.3.1   Joint TLB

The TX49 has a fully associative TLB which maps 48 pairs (odd/even entry) of virtual pages to their corresponding physical addresses.

4.3.2   TLB Entry format

32-bit addressing

127	121	120		MASK	109	108		0	96

95	VPN2	77	76 G	75	72 0	71	ASID	64

63	62 61 0	PFN	38	37	35 34 C	33 D	V	32 0

31	30 29 0	PFN	6	5	3 C	2 D	1 V	0 0

64-bit addressing

255	217	216		MASK	205	204		0	192

191 190 189 R	168 0	167	VPN2	141 140 139 G	136 135 0		ASID	128

127	94 93 0	PFN	70	69	67 66 C	65 D	V	64 0

63	30 29 0	PFN	6	5	3 C	2 D	1 V	0 0

MASK : Page comparison mask. This field sets the variable page size for each TLB entry.

VPN2 : Virtual page number divided by two (maps to two pages)

ASID : Address space ID field.

R : Region. (00: user, 01: supervisor, 11: kernel) used to match Vaddr[63:62].

PFN : Page frame number; upper bits of the physical address.

C : Specifies the cache algorithm to be used (see the "C" field of the EntryLo0, 1).

D : Dirty. If this bit is set, the page is marked as dirty and therefore, writable. This bit is actually a write-protect bit that software can use to prevent alteration of data.

V : Valid. If this bit is set, it indicates that the TLB entry is valid. If a cache hit occurs through a TLB entry when this bit is cleared, a TLB invalid exception occurs.

G : Global. If this bit is set in both Lo0 and Lo1, then ignore the ASID during TLB lookup.

0 : Reserved. Returns zeroes when read.

4.3.3 Instruction-TLB

The TX49 has a 2-entry instruction TLB (ITLB). Each ITLB entry is a subset of any single JTLB entry. The ITLB is completely invisible to software.

4.3.4 Data-TLB

The TX49 has a 4-entry data TLB (DTLB). Each DTLB entry is a subset of any single JTLB entry. The DTLB is completely invisible to software.

4.4 Virtual-to-Physical Address Translation Process

During virtual-to-physical address translation, the CPU compares the 8-bit ASID (if the Global bit, G, is not set) of the virtual address to the ASID of the TLB entry to see if there is a match. One of the following comparisons are also made:

- In 32-bit mode, the highest 7 to 19 bits (depending upon the page size) of the virtual address are compared to the contents of the TLB VPN2 (virtual page number divided by two).
- In 64-bit mode, the highest 15 to 27 bits (depending upon the page size) of the virtual address are compared to the contents of the TLB VPN2 (virtual page number divided by two).

If a TLB entry matches, the physical address and access control bits (*C*, *D*, and *V*) are retrieved from the matching TLB entry. While the *V* bit of the entry must be set for a valid translation to take place, it is not involved in the determination of a matching TLB entry.

Figure 4.4.1 illustrates the TLB address translation process.
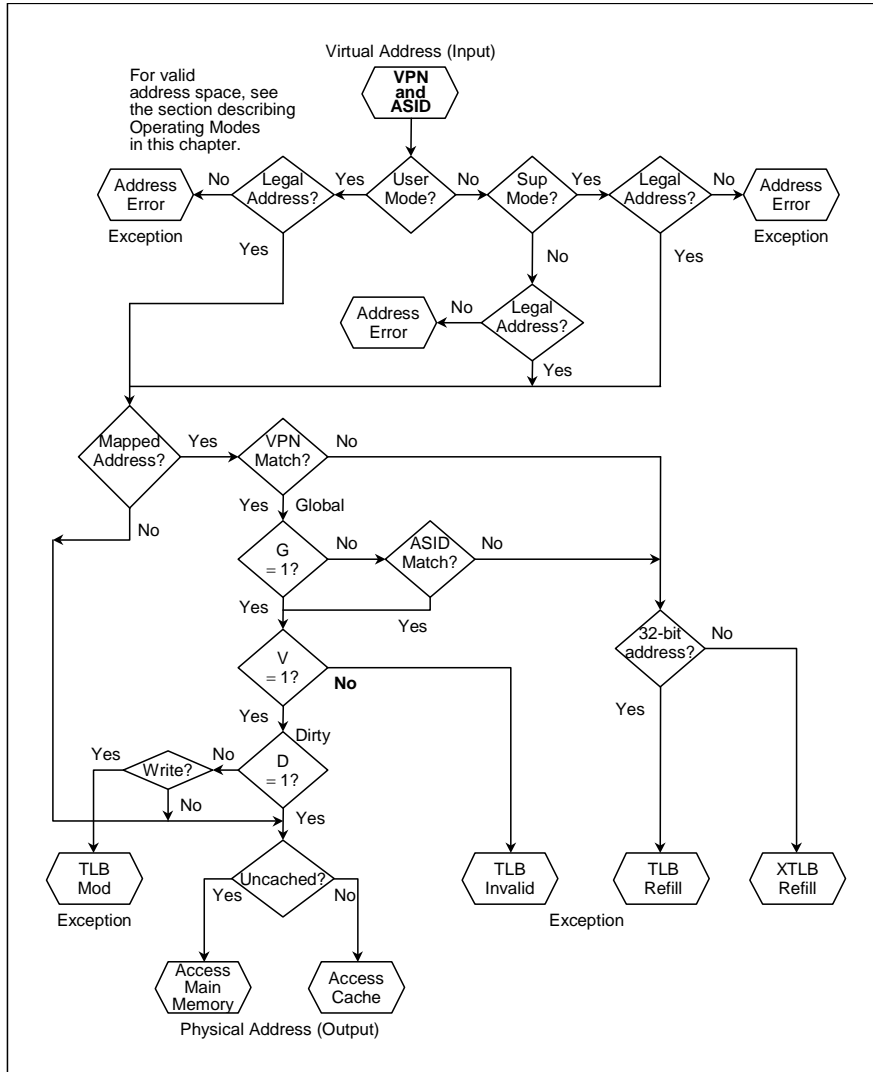
Figure 4.4.1 TLB Address Translation

TLB Misses

If there is no TLB entry that matches the virtual address, a TLB refill exception occurs. (TLB refill exceptions are described in Chapter 8.) If the access control bits (D and V) indicate that the access is not valid, a TLB modification or TLB invalid exception occurs. If the C bits equal 010_2, the physical address that is retrieved accesses main memory, bypassing the cache.

TLB Instructions

Table 4.4.1 lists the instructions that the CPU provides for working with the TLB. See Appendix A for a detailed description of these instructions.

Table 4.4.1 TLB Instructions

Op Code	Description of Instruction
TLBP	Translation Lookaside Buffer Probe
TLBR	Translation Lookaside Buffer Read
TLBWI	Translation Lookaside Buffer Write Index
TLBWR	Translation Lookaside Buffer Write Random

5.   Cache Organization

This chapter describes in detail the cache memory: its place in the TX4951B memory organization, and individual organization of the caches.

This chapter uses the following terminology:

- The data cache may also be referred to as the D-cache.

- The instruction cache may also be referred to as the I-cache.

These terms are used interchangeably throughout this book.

5.1   Memory Organization

Figure 5.1.1 shows the TX4951B system memory hierarchy. In the logical memory hierarchy, both primary and secondary caches lie between the CPU and main memory. They are designed to make the speedup of memory accesses transparent to the user.

Each functional block in Figure 5.1.1 has the capacity to hold more data than the block above it. For instance, physical main memory has a larger capacity than the caches. At the same time, each functional block takes longer to access than any block above it. For instance, it takes longer to access data in main memory than in the CPU on-chip registers.
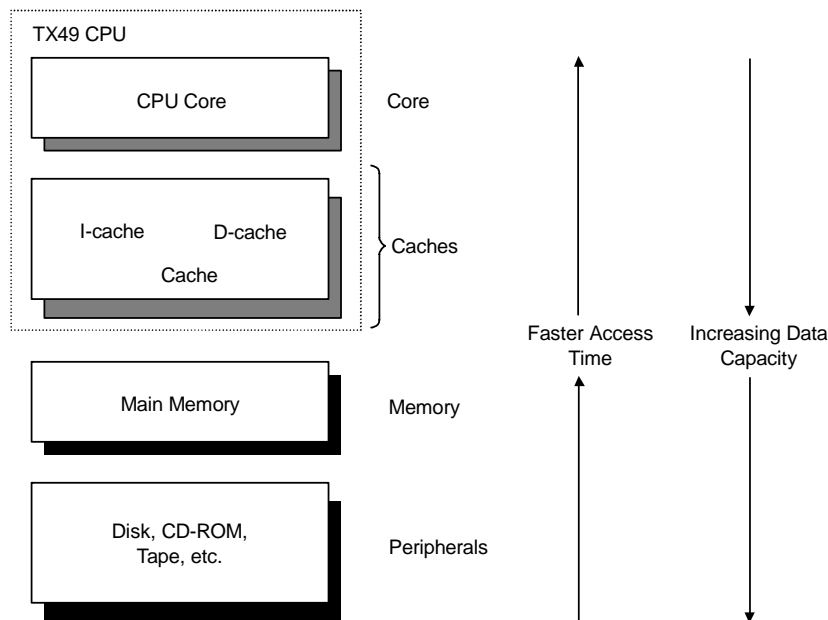
Figure 5.1.1 Logical Hierarchy of Memory

The TX4951B processor has two on-chip caches: one holds instructions (the instruction cache), the other holds data (the data cache). The instruction and data caches can be read in one **CPUCLK** cycle.

Data writes are pipelined and can complete at a rate of one per **CPUCLK** cycle. In the first stage of the cycle, the store address is translated and the tag is checked; in the second stage, the data is written into the data RAM.

5.2   Cache Organization

This section describes the organization of the on-chip data and instruction caches. Figure 5.2.1 provides a block diagram of the TX4951B cache and memory model.
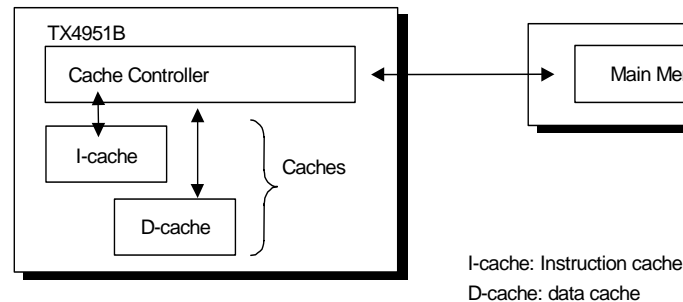
Figure 5.2.1 TX4951B Cache Support

5.2.1   Cache Sizes

The TX4951B instruction cache is 16 Kbytes; the data cache is 8 Kbytes.

5.2.2   Cache Line Lengths

A cache line is the smallest unit of information that can be fetched from main memory for the cache, and that is represented by a single tag.[Note]

The line size for the instruction cache is 8 words (32 bytes) and the line size for the data cache is 8 words (32 bytes).

Note: Cache tags are described in the following sections.

5.2.3   Organization of the Instruction Cache (I-Cache)

Each line of I-cache data (although it is actually an instruction, it is referred to as data to distinguish it from its tag) has an associated 24-bit tag.

The TX4951B processor I-cache has the following characteristics:

- Cache size: 16 KB
- Four-way set associative
- FIFO replacement
- Indexed with a virtual address
- Checked with a physical tag
- Block (line) size: 8 words (32 bytes)
- Burst refill size: 8 words (32 bytes)
- Lockable on a per-line basis (way1 ~ way3)
- All valid bits, lock and FIFO bits are cleared by a Reset exception

5.2.4   Instruction cache address field

Figure 5.2.2 shows the instruction cache address field.

35 12	11 5	4 3	2 0
Physical Tag (24 bits)	Cache Tag Index (7 bits)	Word (2 bits)	Byte (3 bits)
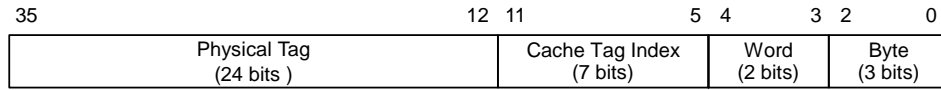
Figure 5.2.2 Instruction cache address field

5.2.5   Instruction cache configuration

Each line in the 4 ways of the instruction cache share F1, F0 replacement bits. Figure 5.2.3 shows the format of replacement bits. These bits are shared by way0, way1, way2 and way3 for 16 KB cache, and indicate next set to which replacement will be directed; when lock bit is set to 1, indicate this set is not locked.

Each line of instruction cache data has an associated 26-bit tag that contains a 24-bit physical address, a single Lock bit and a single valid bit, except for the line in way0, which has an 25-bit tag that excludes a lock bit. Figure 5.2.4 shows the formats of tag and data pair.

1	0
F1	F0
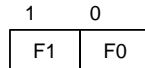
F0: FIFO replace bit 0
F1: FIFO replace bit 1

Figure 5.2.3 Format of replacement bits

Format for way0

Format for way1, 2 and 3
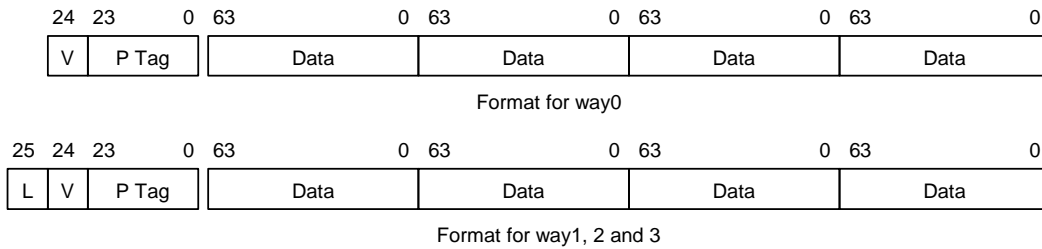
L: Lock bit (1: enable, 0: disable)
V: Valid bit (1: valid, 0: invalid)
PTag: Physical tag (bits 35-12 of the physical address)
Data: Instruction cache data

Figure 5.2.4 Format of tag and data pair for I-cache

5.2.6   Organization of the Data Cache (D-Cache)

Each line of D-cache data has an associated 25-bit tag.

The TX4951B processor D-cache has the following characteristics:

- Cache size: 8 KB

- Four-way set associative

- FIFO replacement

- Indexed with a virtual address

- Checked with a physical tag

- Block (line) size: 8 words (32 bytes)

- Burst size: 8 words (32 bytes)

- Store buffer

- Lockable on a per-line basis (way1 ~ way3)

- Write-back or write-through on a per-page basis

- All write-back, CS, FIFO and lock bits are cleared by a Reset exception

5.2.7   Data cache address field

Figure 5.2.5 shows the data cache address field.

35	11 10	5 4	3 2	0
Physical Tag (25 bits)	Cache Tag Index (6 bits)	Word (2 bits)	Byte (3 bits)	
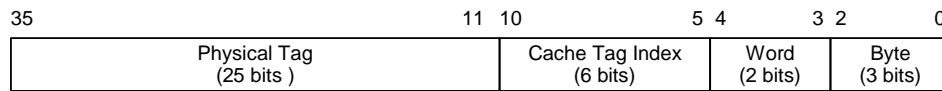
Figure 5.2.5 Data cache address field

5.2.8   Data cache configuration

Each line in the 4 ways of the data cache share F1, F0 replacement bits. Figure 5.2.6 shows the format of replacement bits. These bits are shared by way0, way1, way2 and way3 for 8 KB cache, and indicate next set to which replacement will be directed; when lock bit is set to 1, indicate this set is not locked.

Each line of data cache data has an associated 29-bit tag that contains a 25-bit physical address, a single Lock bit, a single write-back bit and a 2-bit cache state, except for the line in way0, which has an 28-bit tag that excludes a Lock bit. Figure 5.2.7 shows the formats of tag and data pair.

1	0
F1	F0

F0: FIFO replace bit 0
F1: FIFO replace bit 1

Figure 5.2.6 Format of replacement bits

27 26 25	24	0	63	0	63	0	63	0	63	0
W	CS	PTag	DATA		DATA		DATA		DATA	

Format for way0

28	27 26 25	24	0	63	0	63	0	63	0	63	0
L	W	CS	PTag	DATA		DATA		DATA		DATA	

Format for way1, 2 and 3
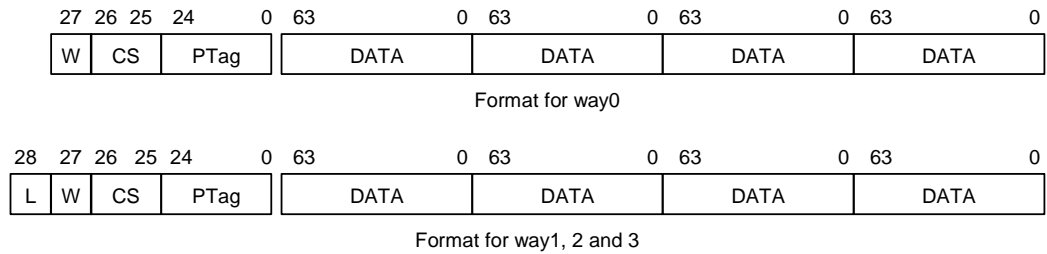
L: Lock bit (1: enable, 0: disable)

W: Write-back bit (set if cache line has written)

CS: Primary cache state (0: Invalid, 1: Reserved, 2: Reserved, 3: Valid)

PTag: Physical tag (bits 35-11 of the physical address)

Data: Data cache data

Figure 5.2.7 Format of tag and data pair for D-cache

In the TX4951B, the W (write-back) bit, not the cache state, indicates when the primary cache contents modified data that must be written back to memory. The states Invalid and Dirty Exclusive are used to describe the cache line. That is, there is no hardware support for cache coherency.

5.3   Lock function

The lock function can be used to locate critical instruction/data in one instruction/data cache set and they are not replaced when the lock bit is set.

5.3.1   Lock function

Setting the Lock bit in each line cache enable the instruction/data cache lock function. When the lock function is enabled, the instruction/data in the valid line is locked and never be replaced. The set to be locked is pointed by FIFO bit. Refilled instruction/data during the lock function is enabled is locked. When a store miss occurs for the write-through data cache without write allocate, the store data is not written to the cache and will therefore not be locked.

The lock function is disabled by clearing the Lock bit in each line.

In order to clear or set the Lock bit in the cache, Cache instructions (Index store I-cache /D-cache Tag) can be used, and in order to load the instruction/data to cache from memory, another Cache instructions (Fill I-cache/D-cache) can be used (refer to Cache instruction).

Clear the lock bit as follows when data written to a locked line should be stored in main memory.

(1)   Read the locked data from cache memory

(2)   Clear the lock bit

(3)   Store the data that was read

5.3.2   Operation during lock

After the lock bit is set for a line, the line can be replaced only when it's line state is invalid. The locked valid line can never be replaced. FIFO bit should point only to the set of locked invalid line or unlocked line.

A write access to a locked valid line takes place only to the cache not to the memory at Write Back mode. Both of the cache and the memory are replaced at Write Through mode.

5.3.3   Example of Data cache locking

During the load operation to the locked line of the cache, any interrupt should be disabled in order to avoid to lock the wrong data.

To lock data cache lines, the following sequence of codes could be used.

```
......................        /* Disable the interrupt */
mtc0   t0, TagLo             /* Load data into TagLo reg */
cache  2 (D), offset (base)  /* Invalidate and lock line in desired set using
                                   Index_Store_Tag cache instruction */
cache  7 (D), offset (base)  /* Fill the cache line from desired memory location */
......................        /* Enable the interrupt */
```

5.3.4   Example of Instruction cache locking

To lock instruction cache lines, the following sequence of codes could be used:

```
......................        /* Disable the interrupt */
mtc0   t0, TagLo             /* Load data into TagLo reg */
cache  2 (I), offset (base)  /* Invalidate and lock line in desired set using
                                   Index_Store_Tag cache instruction */
cache  5 (I), offset (base)  /* Fill the cache line from desired memory location */
......................        /* Enable the interrupt */
```

5.4   The primary cache accessing

Figure 5.4.1 shows the virtual address (VA) index to the primary cache. Instruction cache size is 16 KB and Data cache size is 8 KB. The virtual address bits be used to index into the primary cache decided by the cache size.

Instruction Cache: VA[11:5]
Data Cache: VA[10:5]

VA[11:5]
VA[10:5]

W Stage Tag

64
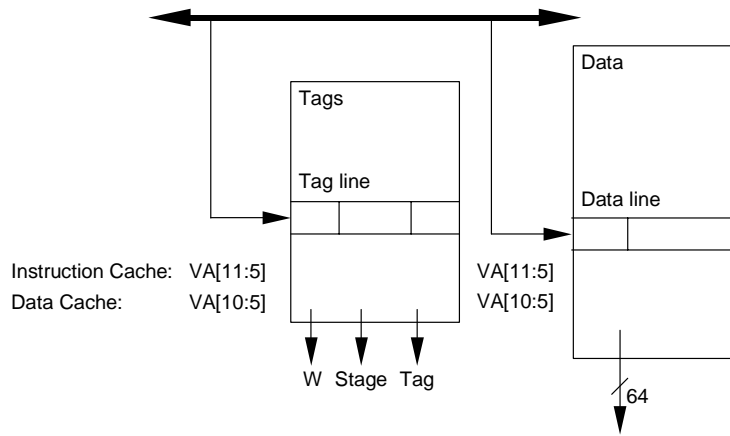
Figure 5.4.1 Primary Cache Data and Tag Organization

5.5   Cache States

The section describes about the state of a cache line. The cache line in the TX4951B are in one of states described in Table 5.5.1.

The I-Cache line is in one of the following states:

- invalid
- valid

The D-Cache line is in one of the following states:

- invalid
- valid

Table 5.5.1 Cache States

Cache line State	Description
Invalid	A cache line that does not contain valid information must be marked invalid, and cannot be used. A cache line in any other state than invalid is assumed to contain valid information.
Valid	A cache line contains valid information. The cache line may or not be consistent with memory and is owned by the processor (see Cache Line Ownership in this chapter).

5.6   Cache Line Ownership

The TX4951B becomes the owner of a cache line after it writes to that cache line (that is, by entering the dirty exclusive), and is responsible for providing the contents of that line on a read request. There can only be one owner for each cache line.

5.7   Cache Multi-Hit Operation

The TX4951B is not guaranteed the operation for the multi-hit of primary cache.

Thus, in case of locking the specified program/data in the primary cache, the program/data must be used after locked in the cache by Fill instruction.

Such as the previous description the cache multi hit does not guarantee in the TX4951B, however, if it occurs, the TX4951B will do the operation as follows.

- Load: read from the higher way of priority (Way0 > Way1 > Way2 > Way3).

- Store: write to all way that are multi-hits.

5.8   FIFO Replacement Algorithm

The instruction and data caches in the TX4951B use the FIFO replacement algorithm.

- Usually, cache elements are replaced in this order: Way0, Way1, Way2, Way3.
- The FIFO[1:0] replacement bits do not point to a locked, valid cache line.
- Data is first written to a cache line marked invalid, if any.
- The FIFO replacement bits change every time memory data is written to the cache or a CACHE instruction is executed.

Figure 5.8.1 shows several examples of how the FIFO replacement bits change.
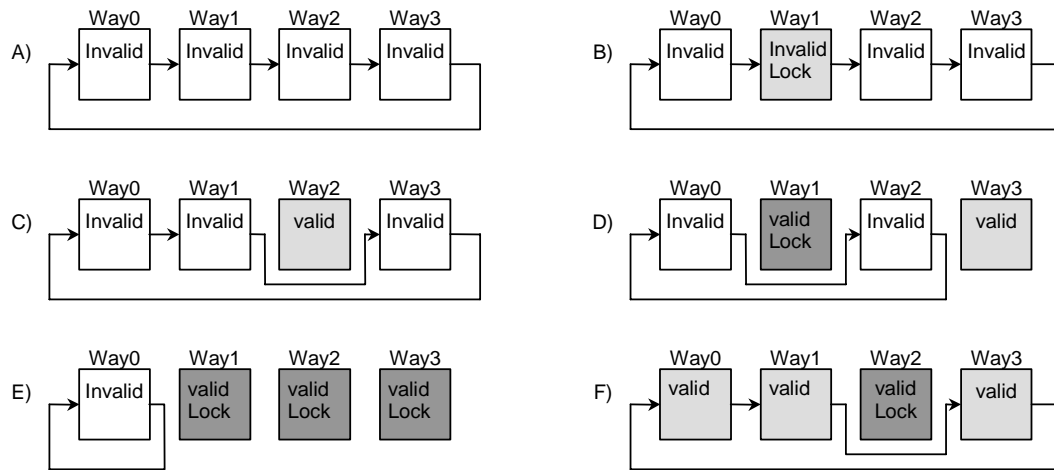
Figure 5.8.1 Examples of Cache State Transitions by the FIFO Replacement Algorithm

5.9   Cache Testing

5.9.1   Cache disabling

The ICE# and DCE# bits in the Config register enable and disable the instruction and data caches respectively.

When the cache is disabled, any attempt to access the cache causes a cache miss; therefore, a cache refill does not occur. (A burst bus cycle does not occur, either, as is the case with an access to a uncached memory space.) With the cache disabled, the Valid (V) and Cache State (CS) bits for each entry remain unchanged.

Note: When the instruction cache is disabled

– All instruction fetches cause an instruction cache miss. External memory accesses will occur as a single-read operation.
– Instruction cache operations by the CACHE instruction are valid.

Note: When the data cache is disabled

– All memory accesses by the load and store instructions cause a data cache miss. At this time, no cache refill occurs. External memory accesses will occur as a single-read or single-write operation.
– Data cache operations by the CACHE instruction are valid.

Note: How to disable the instruction cache reliably

– To disable the instruction cache, stop instruction streaming by following the MTC0 instruction with a jump instruction, as shown below:

```
Example: MTC0 Rn, Config        (Set ICE# bit.)
         J     L1                (Jump to L1 and stop streaming.)
         NOP                     (Jump delay slot)
     L1: CACHE IndexInvalidate, offset (base)
```

5.9.2   Cache Flushing

Both the instruction and data caches are flushed by a Cold Reset or Warm Reset exception. i:e., all the Valid and CS bits are cleared to zeroes.

The instruction cache is also flushed by the Index Invalidate and Hit Invalidate operations with a CACHE instruction. The data cache is flushed by the Hit Invalidate operation with a CACHE instruction.

Data is written back to the main memory when an Index Writeback Invalidate or Hit Writeback Invalidate operation is performed, when a Hit Writeback operation is performed, and when a cache line is replaced. When the write-back policy is employed, it is required to consciously maintain cache coherency when flushing the cache.

5.10 Cache Operations

As described earlier, caches provide fast temporary data storage, and they make the speedup of memory accesses transparent to the user. In general, the processor accesses cache-resident instructions or data through the following procedure:

1. The processor, through the on-chip cache controller, attempts to access the next instruction or data in the appropriate cache.

2. The cache controller checks to see if this instruction or data is present in the cache.

   • If the instruction/data is present, the processor retrieves it. This is called a cache *hit*.

   • If the instruction/data is not present in the cache, the cache controller must retrieve it from memory. This is called a cache *miss*.

3. The processor retrieves the instruction/data from the cache and operation continues.

It is possible for the same data to be in two places simultaneously: main memory and cache. This data is kept consistent through the use of a *write-back* methodology; that is, modified data is not written back to memory until the cache line is to be replaced.

Instruction and data cache line replacement operations are listed in described as the following sections.

Table 5.10.1 Cache Instruction

Name	Caches	Operation
Index Invalidate	Instruction	Sets the cache state of cache block to invalid.
Index Write Back Invalidate	Data	Examines cache state, if Valid Dirty, then that block is written back to main memory. Then the cache block is set to invalid.
Index Load Tag	Instruction & Data	Read the tag for the cache block at the specified index and place it into TagLo.
Index Store Tag	Instruction & Data	Write the tag for the cache block at the specified index from the TagLo and TagHi register.
Create Dirty Exclusive	Data	If the cache does not contain the specified address, and the block is Valid Dirty the block will be written back to main memory. Then the tag will be set to the specified physical address and will be marked valid.
Hit Invalidate	Instruction & Data	If the cache block contains the specified address, cache block will be marked invalid.
Hit Write Back Invalidate	Data	If the cache block contains the specified address, and it is Valid Dirty, the data will be written back to main memory. Then, the cache block is marked invalid.
Fill	Instruction	Fill the Instruction cache block from main memory.
Fill	Data	Fill the Data cache block from memory.
Hit Write Back	Data	If the cache block contains the specified address, and it is marked Valid Dirty, the block will be written back to main memory, and marked Valid Clean.

5.10.1 Cache Write Policy

The TX4951B processor manages its data cache by using a write-back and a write-through policy. A write-back stores write data into the cache, instead of writing it directly to memory. Some time later this data is independently written into memory. In the TX4951B implementation, a modified cache line is not written back to memory until the cache line is to be replaced either in the course of satisfying a cache miss, or during the execution of a write-back CACHE instruction.

When the processor writes a cache line back to memory, it does not ordinarily retain a copy of the cache line, and the state of the cache line is changed to invalid.

A write-through is written simultaneously to cache and memory.

5.10.2 Data Cache Line Replacement

Since the data cache uses a write-back and a write-through methodology, a cache line load is issued to main memory on a load or store miss, as described below. After the data from memory is written to the data cache, the pipeline resumes execution.

TX4951B does not support "Critical Data Word First". Always it transfer the data of first address.

Rules for replacement on data load and data store misses are given below.

- **Data Load Miss**
  If the missed line is not dirty, it is replaced with the requested line.
  If the missed line is dirty, it is moved to the write buffer. The requested line replaces the missed line, and the data in the write buffer is written to memory.

- **Data Store Miss**
  If the missed line is not dirty, the requested line is merged with the store data and written to memory.
  If the missed line is dirty, it is moved to the write buffer. The requested line is merged with the store data and written to cache, and data in the write buffer is written to memory.
  The data cache miss penalties, in number of GBUSCLK cycles, are given in Table 5.10.2.

Table 5.10.2 Data Cache Refill Penalty Cycle Count

Number of CPUCLK Cycles	Action
1	Stall the DC stage.
1	Transfer address to the write buffer and wait for the pipeline start signal
1-2	Transfer address to the internal SysAD bus on the GBUSCLK.
2	Transfer to the external SysAD bus.
M	Time needed to access memory, measured in CPUCLK cycles.
4	Transfer the cache line form memory to the SysAD bus.
2	Transfer the cache line from the external bus to the internal bus.
0	Restart the DC stage.

5.10.3 Instruction Cache Line Replacement

For an instruction cache miss, refill is done using sequential ordering, starting from the first word of the retrieved cache line.

During an instruction cache miss, a memory read is issued. The requested line is returned from memory and written to the instruction cache. At this time the pipeline resumes execution, and the instruction cache is reaccessed.

The replacement sequence for an instruction cache miss is:

1. Move the instruction physical address to the processor pads.
2. Wait for a CPUCLK cycle, aligned with an GBUSCLK boundary, to occur.
3. Read the line from memory and write it out to the instruction cache array.
4. Restart the processor pipe.

The instruction cache miss penalties, in number of CPUCLK, is given in Table 5.10.3.

Table 5.10.3 Instruction Cache Refill Penalty Cycle Count

Number of CPUCLK	Action
1	Stall the RF stage.
1	Transfer address to the write buffer and wait for the pipeline start signal.
1-2	Transfer to the external SysAD bus.
2	Transfer to the external SysAD bus.
M	Time needsd to access memory, measured in CPUCLK cycles.
8	Transfer the cache line from memory to the SysAD bus.
2	Transfer the cache line from the external bus to the internal bus.
0	Restart the RF stage.

5.11 Manipulation of the Caches by an External Agent

The TX4951B does not provide any mechanisms for an external agent to examine and manipulate the state and contents of the caches.

6.   Write Buffer

The TX4951B contains a write buffer to improve the performance of writes to the external memory. Every write to external memory uses this on-chip write buffer. The write buffer holds up to four 64-bit address and data pairs.

For a cache miss write-back, the entire buffer is used for the write-back data and allows the processor to proceed in parallel with the memory update. For uncached and write-through stores, the write buffer uncouples the CPU from the write to memory. If the write buffer is full, additional stores will stall until there is room for them in the write buffer.

The TX4951B core might issue a read request while the write buffer is performing a write operation. Multiple read/write operations are serviced in the following order:

- If there is only a write request, the data in the write buffer is written to an external device.
- If there is only a read request, a read operation is performed to bring in data from an external device.
- If a read request and a write request occur simultaneously, the read request is serviced first, except for the following cases:
  - when the processor issues a read request to the target address of one of the write buffer entries
  - when the processor issues an uncacheable read reference while the write buffer has uncacheable write data

The BC0T and BC0F instructions can be used to determine whether any data is present in the write buffer:

If there is data in the write buffer, the coprocessor condition signal is false (0).

If there is no data in the write buffer, the coprocessor condition signal is true (1).

Following is the assembly language code to freeze the processor until the write buffer becomes empty.

```
            SW
            NOP
            NOP
     Loop: BC0F Loop
            NOP
```

The following sequence of instructions also causes the TX4951B to perform the same action. Appended to a store instruction, the SYNC instruction ensures that the store instruction initiated prior to this instruction is completed before any instruction after this instruction is allowed to start.

```
            SW
            SYNC
```

7.   Debug Support Unit

7.1   Features

1. Utilizes JTAG interface compatible with IEEE Std. 1149.1.

2. Processor access to external processor probe to execute from the external trace memory during debug exception and boot time. This is to eliminate system memory for debugging purpose.

3. Supports DMA access through JTAG interface to internal processor bus to access internal registers, host system peripherals and system memory.

4. Debug functions

    - Instruction Address Break
    - Data Bus break
    - Processor Bus Break
    - Reset, NMI, Interrupt Mask

5. Instructions for Debug

    - SDBBP, DERET, CTC0, CFC0

6. CP0 Registers for Debug

    - Debug, DEPC, DESAVE

7.2   EJTAG interface

This interface consists of two modes of operation a Run Time Mode and Real Time Mode. The Run Time Mode provides functions such as processor Run, STOP, Single Step, and access to internal registers and system memory. The Real Time mode provides additional status pins used in conjunction with JTAG pins for Real Time Trace information.

Table 7.2.1 JTAG Interface

PIN NAME	I / O	FUNCTION
JTDI	I	**JTAG data input / Debug interrupt input** Input serial data to JTAG data/instruction registers.
JTCK	I	**JTAG clock input** Clock input for JTAG. The JTDI and JTMS data are latched on rising edges of this clock.
JTDO	O	**JTAG data output / Trace PC output** Data is serially shifted out from this pin.
JTMS	I	**JTAG command** Controls mainly the status transition of the TAP controller state machine. When the serial input data is a JTAG command, apply a high signal (= 1) to this pin.
TRST*	I	**Test reset input** Reset input for a real-time debug system. When TRST* is asserted (= 0), the debug support unit (DSU) is initialized.

Note1: Leave TPC (3-1) pins open when not using them as PC trace outputs for debugging.

7.3   Debug Unit

7.3.1   Extended Instructions

- SDBBP
- DERET
- CTC0
- CFC0

7.3.2   Extended Debug Registers in CP0

- Debug Register
- Debug Exception PC (DEPC)
- Debug SAVE

7.4   Register Map

Table 7.4.1 Register Map

Address	Mnemonic	Description
0xf ff30 0000	DCR	Debug Control Register
0xf ff30 0008	IBS	Instruction Break Status
0xf ff30 0010	DBS	Data Break Status
0xf ff30 0018	PBS	Processor Break Status
0xf ff30 0100	IBA0	Instruction Break Address 0
0xf ff30 0108	IBC0	Instruction Break Control 0
0xf ff30 0110	IBM0	Instruction Break Address Mask 0
0xf ff30 0300	DBA0	Data Break Address 0
0xf ff30 0308	DBC0	Data Break Control 0
0xf ff30 0310	DBM0	Data Break Address Mask 0
0xf ff30 0318	DB0	Data Break Value 0
0xf ff30 0600	PBA0	Processor Bus Break Address 0
0xf ff30 0608	PBD0	Processor Bus Break Data 0
0xf ff30 0610	PBM0	Processor Bus Break Mask 0
0xf ff30 0618	PBC0	Processor Bus Break Control 0

7.5   Processor Bus Break Function

This function is to monitor the interface to core and provide debug interruption or trace trigger for a given physical address and data.

7.6   Debug Exception

Three kinds of debug exception are supported.

- Debug Single Step (DSS bit)
- Debug Breakpoint Exception (SDBBP Instruction)
- JTAG Break Exception (Jtagbrk bit in JTAG_Control_Register)

Note: During real time debugging, first two functions are disabled.

8.   CPU Exception

8.1 Introduction

This chapter describes the explanation of CPU exception processing. The chapter concludes with a description of each exception's cause, together with the manner in which the CPU processes and services these exceptions.

8.2 Exception Vector Locations

Exception vector addresses are stored in an area of kseg0 or kseg1 except for Debug exception vector. The vector address of the ColdReset, SoftReset and NMI exception is always in a non-cacheable area of kseg1. Vector addresses of the other exceptions depend on the BEV bit of Status register. When BEV is 0, these exceptions are vectored to a cacheable area of kseg0. When BEV is 1, all vector addresses are in a non-cacheable area of kseg1.

Table 8.2.1 shows the list of the exception vector locations.

Table 8.2.1 Exception Vector Locations

Exception	Vector Address (virtual address)	
	(BEV = 0)	(BEV = 1)
ColdReset, NMI	0xffff_ffff_bfc0_0000	0xffff_ffff_bfc0_0000
TLB refill, EXL = 0	0xffff_ffff_8000_0000	0xffff_ffff_bfc0_0200
XTLB refill, EXL = 0 (X = 64-bit TLB)	0xffff_ffff_8000_0080	0xffff_ffff_bfc0_0280
Others (common exception)	0xffff_ffff_8000_0180	0xffff_ffff_bfc0_0380

Exception	Vector Address (physical address)	
	(BEV = 0)	(BEV = 1)
ColdReset, NMI	0x0_1fc0_0000	0x0_1fc0_0000
TLB refill, EXL = 0	0x0_0000_0000	0x0_1fc0_0200
XTLB refill, EXL = 0 (X = 64-bit TLB)	0x0_0000_0080	0x0_1fc0_0280
Others (common exception)	0x0_0000_0180	0x0_1fc0_0380

The cache error exception is not occurred because the TX49 does not have the parity bit into the primary cache. Debug exception needs the care, it has the special address. Table 8.2.2 shows the list of the debug exception vector locations.

Table 8.2.2 Debug Exception Vector Locations

Exception	Debug Exception Vector Address (virtual address)	
	(ProbEnb = 0)	(ProbEnb = 1)
Debug	0xffff_ffff_bfc0_0400	0xffff_ffff_ff20_0200

Exception	Debug Exception Vector Address (physical address)	
	(ProbEnb = 0)	(ProbEnb = 1)
Debug	0x0_1fc0_0400	0xf_ff20_0200

8.3 Priority of Exception

More than one exception may be raised for the same instruction, in which case only the exception with the highest priority is reported. The TX49 Processor Core instruction exception priority is shown in Table 8.3.1.

Table 8.3.1 Priority of Exception

Priority	Exception		Mnemonic
High	Cold Reset		
	NMI		
	Address error	Inst. Fetch	AdEL
	TLB refill	Inst. Fetch	TLBL
	TLB invalid	Inst. Fetch	TLBL
	Bus error	Inst. Fetch	IBE
	Integer overflow, Trap, System Call, Breakpoint, Reserved Instruction or Coprocessor Unusable		Ov, Tr, Sys, Bp, RI, CpU
	Address error	Data access	AdEL/AdES
	TLB refill	Data access	TLBL/TLBS
	TLB invalid	Data access	TLBL/TLBS
	TLB modified	Data write	Mod
Low	Bus error	Data access	DBE
	Interrupt		Int

General exceptions (i.e., exceptions other than debug exceptions) are prioritized as follows:

1. If more than one exception condition occurs for a signal instruction, only the exception with the highest priority is reported, as shown in Table 8.3.1 (from highest to lowest priority).

2. If two instructions cause exception conditions in the M and E stages of the pipeline simultaneously, the instruction in the M stage causes the processor to take an exception.

3. When 64-bit instructions are executed in 32-bit mode, the Reserved Instruction (RI) exception can occur simultaneous with other exception, as shown below. In that case, the RI exception is given precedence.

   - RI and CpU

   - RI and Ov

   - RI and AdEL/S (data)

   - RI and TLBL/S (data)

General and debug exceptions are prioritized as follows:

1. If a general exception condition and a debug exception condition occur for a single instruction, the debug exception is serviced first, and then the general exception is serviced.

2. If two instructions cause exception conditions in the M and E stages of the pipeline simultaneously, only the instruction in the M stage generates an exception.

8.4 ColdReset Exception

8.4.1 Cause

This ColdReset exception occurs when the GCOLDRESET* signal is asserted and then deasserted. This exception is not maskable.

8.4.2 Processing

A special interrupt vector that resides in an unmapped and uncached area is used. It is therefore not necessary for hardware to initialize TLB and cache memory in order to process this exception. The vector location of this exception is;

- In 32-bit mode, 0xbfc0 0000 (virtual address), 0x1fc0_0000 (physical address)
- In 64-bit mode, 0xffff ffff bfc0 0000 (virtual address), 0x1fc0_0000 (physical address)

The most register's contents are cleared when this exception occurs. The values of these bits are listed into the table of Section 7.

Valid bits, Lock bits and FIFO replacement bits in the instruction cache are all cleared to 0. W bits, CS bits, Lock bits and FIFO replacement bits in the data cache are all cleared to 0.

If a ColdReset exception occurs during bus cycle, the current bus cycle is aborted and an exception is taken.

8.4.3 Servicing

The ColdReset exception is serviced by;

- initializing all registers, coprocessor registers, caches and the memory system
- performing diagnostic tests
- bootstrapping the operating system

8.5 NMI (Non-maskable Interrupt) Exception

8.5.1 Cause

The NMI (Non-maskable Interrupt) exception occurs at the falling edge of the GNMI* signal. This interrupt is not maskable, and occurs regardless of the EXL, ERL and IE bits of the Status register.

8.5.2 Processing

The same special interrupt vector as for Cold-reset/Soft-reset exception (0xbfc0_0000/ 0xffff_ffff_bfc0_0000). This vector is located within unmapped and uncached area so that the cache and TLB need not be initialized to process this exception. When this exception occurs, the SR bit of Status register is set.

Because NMI exception can occur in the midst of another exception, it is not normally possible to continue program execution after servicing NMI exception.

Unlike the Cold-reset/Soft-reset exception, but like other exceptions, this exception occurs at an instruction boundary. The state of the primary cache and memory system are preserved by this exception.

All register contents are retained except for the following.

- ErrorEPC register, which contains the restart PC
  If the exception-causing instruction is in a branch delay slot, the ErrorEPC register points at the preceding branch instruction.
- ERL, SR and BEV bits of the Status register, which is set to 1.

8.5.3 Servicing

The NMI exception is serviced by saving the current processor state for diagnostic purposes, and reinitializing the system for the ColdReset exception.

8.6 Address Error Exception

8.6.1 Cause

The Address Error exception occurs when an attempt is made to execute one of the following.

- load or store a doubleword that is not aligned on a doubleword boundary
- load, fetch or store a word that is not aligned on a word boundary
- load or store a halfword that is not aligned on a halfword boundary
- reference Kernel mode address while in User or Supervisor mode
- reference Supervisor mode address while in User mode

This exception is not maskable.

8.6.2 Processing

The common exception vector is used. ExcCode AdEL or AdES in Cause register is set depending on whether the memory access attempt was a load or store. When this exception is raised, the misalign virtual address causing the exception, or the protected virtual address that was illegally referenced, is placed in BadVAddr register. The contents of the VPN field of Context and EntryHi registers are undefined, as are the contents of EntryLo register.

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the instruction causing the exception. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register and BD bit of Cause register is set to "1".

8.6.3 Servicing

The process executing at the time is handed a segmentation violation signal. This error is usually fatal to the process incurring the exception.

8.7 TLB Refill Exception

8.7.1 Cause

The TLB refill exception occurs when there is no TLB entry to match a reference to a mapped address. This exception is not maskable.

8.7.2 Processing

There are two special exception vectors for this exception; one for references to 32-bit virtual address, and one for references to 64-bit virtual address. The KX, SX and UX bits of Status register determine whether the User, Supervisor or Kernel address referenced are 32-bit mode or 64-bit mode. When EXL bit of Status register is set to "0", all references use these vectors. When this exception occurs, TLBL or TLBS code is set in the ExcCode field of Cause register. This code indicates whether the instruction, as shown by EPC register and BD bit of Cause register, caused the miss by an instruction reference, load operation, or store operation.

When this exception occurs;

- BadVAddr, Context, XContext and EntryHi registers hold the virtual address failed address translation
- EntryHi register contains ASID from which the translation fault occurred, too
- A valid address in which to place the replacement TLB entry is contained into Random register
- The contents of EntryLo register are undefined

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the instruction causing the exception. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register and BD bit of Cause register is set to "1".

8.7.3 Servicing

To service this exception, the contents of the Context or XContext register are used as a virtual address to fetch memory locations containing the physical page frame and access control bits for a pair of TLB entries. The two entries are placed into the EntryLo0/EntryLo1 register; the EntryHi and EntryLo registers are written into the TLB.

It is possible that the virtual address used to obtain the physical address and access control information is on a page that is not resident in the TLB. This condition is processed by allowing a TLB refill exception in the TLB refill handler. This second exception goes to the common exception vector because the EXL bit of the Status register is set.

8.8 TLB Invalid Exception

8.8.1 Cause

The TLB Invalid exception occurs when a virtual address reference matches a TLB entry that is marked invalid (TLB valid bit cleared). This exception is not maskable.

8.8.2 Processing

The common exception vector is used for this exception. When this exception occurs, TLBL or TLBS code is set in the ExcCode field of Cause register. This code indicates whether the instruction, as shown by EPC register and BD bit of Cause register, caused the miss by an instruction reference, load operation, or store operation.

When this exception occurs;

- BadVAddr, Context, XContext and EntryHi registers hold the virtual address failed address translation
- EntryHi register contains ASID from which the translation fault occurred, too
- A valid address in which to place the replacement TLB entry is contained into Random register
- The contents of EntryLo register are undefined

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the instruction causing the exception. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register and BD bit of Cause register is set to "1".

8.8.3 Servicing

A TLB entry is typically marked invalid when one of the following is true;

- a virtual address does not exist
- the virtual address exists, but is not in main memory (a page fault)
- a trap is desired on any reference to the page (for example, to maintain a reference bit or during debug)

After servicing the cause of a TLB Invalid exception, the TLB entry is located with TLB Probe (TLBP) instruction, and replaced by an entry with that entry's Valid bit set.

8.9 TLB Modified Exception

8.9.1   Cause

The TLB Modified exception occurs when a store operation virtual address reference to memory matches a TLB entry that is marked valid but is not dirty and therefore is not writable. This exception is not maskable.

8.9.2   Processing

The common exception vector is used for this exception, and Mod code in Cause register is set.

When this exception occurs;

- BadVAddr, Context, XContext and EntryHi registers hold the virtual address failed address translation
- EntryHi register contains ASID from which the translation fault occurred, too
- The contents of EntryLo register are undefined

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the instruction causing the exception. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register and BD bit of Cause register is set to 1.

8.9.3   Servicing

The kernel uses the failed virtual address or virtual page number to identify the corresponding access control information. The page identified may or may not permit write accesses; if writes are not permitted, a write protection violation occurs.

If write accessed are permitted, the page frame is marked dirty/writable by the kernel in its own data structures. The TLB Probe (TLBP) instruction places the index of the TLB entry that must be altered into the Index register. The EntryLo register is loaded with a word containing the physical page frame and access control bits (with the D bit set), and the EntryHi and EntryLo registers are written into the TLB.

8.10 Bus Error Exception

8.10.1 Cause

The Bus Error exception occurs when GBUSERR* signal is asserted during a memory read bus cycle. This exception is raised by board-level circuitry for events such as bus time-out, backplane bus parity errors, and invalid physical memory addresses or access types. This occurs during execution of the instruction causing the bus error. The memory bus cycle ends upon notification of a bus error. When a bus error is raised during a burst refill, the following refill is not performed. A bus error request made by asserting GBUSERR* signal will be ignored if TX49 is executing a cycle other than a bus cycle. It is therefore not possible to raise a Bus Error exception in a write access using a write buffer. A general interrupt must be used instead. This exception is not maskable.

8.10.2 Processing

The common interrupt vector is used for a Bus Error exception. The IBE or DBE code in the ExcCode field of the Cause register is set, signifying whether the instruction (as indicated by the EPC register and BD bit in the Cause register) caused the exception by an instruction reference, load operation, or store operation.

The EPC register contains the address of the instruction that caused the exception, unless it is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set.

8.10.3 Servicing

The physical address at which the fault occurred can be computed from information available in the CP0 registers.

- If the IBE code in the Cause register is set (indicating an instruction fetch reference), the virtual address is contained in the EPC register (or 4+ the contents of the EPC register if the BD bit of the Cause register is set).
- If the DBE code is set (indicating a load or store reference), the instruction that caused the exception is located at the virtual address contained in the EPC register (or 4+ the contents of the EPC register if the BD bit of the Cause register is set).

The virtual address of the load and store reference can then be obtained by interpreting the instruction. The physical address can be obtained by using the TLB Probe (TLBP) instruction and reading the EntryLo register to compute the physical page number.

The process executing at the time of this exception is handed a bus error signal, which is usually fatal.

The bus error treats only the read bus cycle because the TX49 support the write buffer. Because the TX49 supports the non-blocking load and the streaming, please stop the program counter by SYNC instruction or the depending registers instruction, for using EPC register to return from this exception.

8.11 Integer Overflow Exception

8.11.1 Cause

The Integer Overflow exception occurs when ADD, ADDI, SUB, DADD, DADDI or DSUB instruction results in a 2's complement overflow. This exception is not maskable.

8.11.2 Processing

The common exception vector is used for this exception, and the Ov code in Cause register is set.

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the instruction causing the exception. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register and BD bit of Cause register is set to 1.

8.11.3 Servicing

The process executing at the time of the exception is handed a floating-point exception/integer overflow signal. This error is usually fatal to the current process.

8.12 Trap Exception

8.12.1 Cause

The Trap exception occurs when TGE, TGEU, TLT, TLTU, TEQ, TNE, TGEI, TGEIU, TLTI, TLTIU, TEQI or TNEI instruction results in a TRUE condition. This exception is not maskable.

8.12.2 Processing

The common exception vector is used for this exception, and the Tr code in Cause register is set.

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the instruction causing the exception. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register and BD bit of Cause register is set to 1.

8.12.3 Servicing

The process executing at the time of a Trap exception is handed a floating-point exception/integer overflow signal. This error is usually fatal.

8.13 System Call Exception

8.13.1 Cause

The System Call exception occurs during an attempt to execute the SYSCALL instruction. This exception is not maskable.

8.13.2 Processing

The common exception vector is used for this exception, and the Sys code in Cause register is set.

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the SYSCALL instruction. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register.

If the SYSCALL instruction is in a branch delay slot, BD bit of Status register is set, otherwise this bit is cleared.

8.13.3 Servicing

When this exception occurs, control is transferred to the applicable system routine.

To resume execution, the EPC register must be altered so that the SYSCALL instruction does not re-execute; this is accomplished by adding a value of 4 to the EPC register (EPC register + 4) before returning.

If a SYSCALL instruction is in a branch delay slot, a more complicated algorithm, beyond the scope of this description, may be required.

8.14 Breakpoint Exception

8.14.1 Cause

The Breakpoint exception occurs when an attempt is made to execute the BREAK instruction. This exception is not maskable.

8.14.2 Processing

The common exception vector is used for this exception, and the Bp code in Cause register is set.

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the BREAK instruction. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register.

If the BREAK instruction is in a branch delay slot, BD bit of Status register is set, otherwise this bit is cleared.

8.14.3 Servicing

When the Breakpoint exception occurs, control is transferred to the applicable system routine. Additional distinctions can be mode by analyzing the unused bits of the BREAK instruction (bits 25-6), and loading the contents of the instruction whose address the EPC register contains. A value of 4 must be added to the contents of the EPC register (EPC register + 4) to locate the instruction if it resides in a branch delay slot.

To resume execution, the EPC register must be altered so that the BREAK instruction does not re-execute; this is accomplished by adding a value of 4 to the EPC register (EPC register + 4) before returning.

If a BREAK instruction is in a branch delay slot, interpretation of the branch instruction is required to resume execution.

8.15 Reserved Instruction Exception

8.15.1 Cause

The Reserved Instruction exception occurs when one of the following condition occurs:

- an attempt is made to execute an instruction with an undefined major opecode (bits 31-26)

- an attempt is made to execute a SPECIAL instruction with an undefined minor opcode (bits 5-0)

- an attempt is made to execute a REGIMM instruction with an undefined minor opcode (bits 20-16)

- an attempt is made to execute 64-bit operations in 32-bit mode when in User or Supervisor modes

- an attempt is made to execute a COPz rs instruction with an undefined minor opcode (bits 25-21)

- an attempt is made to execute a COPz rt instruction with an undefined minor opcode (bits 20-16)

64-bit operations are always valid in Kernel mode regardless of the value of the KX bit in Status register. This exception is not maskable.

8.15.2 Processing

The common exception vector is used for this exception, and the RI code in Cause register is set.

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the instruction causing the exception. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register and the BD bit of Cause register is set to 1.

8.15.3 Servicing

No instruction in the MIPS ISA are currently interpreted. The process executing at the time of this exception is handed an illegal instruction/reserved operand fault signal. This error is usually fatal.

8.16 Coprocessor Unusable Exception

8.16.1 Cause

The Coprocessor Unusable exception occurs when an attempt is made to execute a coprocessor instruction for either.

- attempting to execute a coprocessor CPz instruction when its corresponding CUz bit in Status register.

- in User or Supervisor mode attempting to execute a CP0 instruction when CU0 bit is cleared to "0". (In Kernel mode, an exception is not raised when a CP0 instruction is issued , regardless of the CU0 bit setting)

- an attempt is made to execute a FPU instruction in TX49 without FPU

8.16.2 Processing

The common exception vector is used for this exception, and the CpU code in Cause register is set. The coprocessor number referred to at the time of the exception is stored in Cause register CE (Coprocessor Error) field.

If EXL bit of Status register is only set to 0, the following operation is executed. EPC register points to the address of the instruction causing the exception. If, however, the affected instruction was in the branch delay slot (for execution during a branch), the immediately preceding branch instruction address is retained in EPC register and BD bit of Cause register is set to 1.

8.16.3 Servicing

The coprocessor unit to which an attempted reference was mode is identified by the Coprocessor Usage Error field, which results in one of the following situations:

- If the process is entitled access to the coprocessor, the coprocessor is marked usable and the corresponding user state is restored to the coprocessor.

- If the process is entitled access to the coprocessor, but the coprocessor does not exist or has failed, interpretation of the coprocessor instruction is possible.

- If the BD bit is set in the Cause register, the branch instruction must be interpreted; then the coprocessor instruction can be emulated and execution resumed with the EPC register advanced past the coprocessor instruction.

- If the process is not entitled access to the coprocessor, the process executing at the time is handed an illegal instruction/privileged instruction fault signal. This error is usually fatal.

8.17 Interrupt Exception

8.17.1 Cause

The Interrupt exception is raised by any of eight interrupts (two software and six hardware). A hardware interrupt is raised when GINT* signal goes active. A software interrupt is raised by setting the IP[1]/IP[0] bit in Cause register. The significance of these interrupts is dependent upon the specific system implementation.

Each of the eight interrupts can be masked individually by clearing its corresponding bit in the IM(Interrupt Mask) field of Status register, and all interrupts can be masked at once by clearing IE bit of Status register to "0".

If the GTINTDIS is low when a Reset exception occurred, GINT[5]* is disabled and the timer exception is enabled.

8.17.2 Processing

The common exception vector is used as following;

- In 32-bit mode, 0x8000 0180 (BEV = 0)

  0xbfc0 0380 (BEV = 1)

- In 64-bit mode, 0xffff ffff 8000 0180 (BEV = 0)

  0xffff ffff bfc0 0380 (BEV = 1)

8.17.3 Servicing

If the interrupt is caused by one of the two software-generated exceptions (SW1 or SW0), the interrupt condition is cleared by setting the corresponding Cause register bit to 0.

If the interrupt is hardware-generated, the interrupt condition is cleared by correcting the condition causing the interrupt pin to be asserted.

If the timer interrupt is caused, the interrupt condition is cleared by changing the value of the Compare register or setting the corresponding Cause register bit (IP[7]) to 0.

Interrupts are not acceptable when the settings of the Status register are EXL = 1 and ERL = 1.

Note: due to the write buffer, a store to an external device will not necessary occur until after other instructions in the pipeline finish. Thus, the user must ensure that the store will occur before the return from exception instruction (ERET) is executed otherwise the interrupt may be serviced again even though there should be no interrupt pending.

8.18 Exception Handling and Servicing Flowcharts

The remainder of this chapter contains flowcharts for the following exceptions and guidelines for their handlers:

- general exceptions and their exception handler
- TLB/XTLB miss exception and their exception handler
- Cold Reset, Soft Reset and NMI exceptions, and a guideline to their handler.

Generally speaking, the exceptions are handled by hardware (HW); the exceptions are then serviced by software (SW).

Exceptions other than Reset, Soft Reset, NMI or first-level miss

Note1: Interrupts can be masked by IE or IMs

Note2: TX4951B doesn't occur Soft Rose exception

Table 8.18.1 General Exception Handler (HW)

Comments

* Treat in unmapped area
* Save the Context (Status, Cause, X/Context, EPC, etc.) to stack area
* Avoid all other exceptions except to NMI, Soft Reset and Cold Reset until saving the Context

Save the Context

Exception cause is interrupt? — * Check ExcCode field of Cause register

No

Set Status register (KSU←00, EXL←0, IE = 1)

Interrupt enabled

Yes

Check of interrupt cause — * Check the interrupt cause from IP field of Cause register and IM field of Status register

Service routine of each exception

Service routine of each interrupts — * Negate the current interrupt cause in the service routine

Interrupt disabled — * Set 1 to EXL field of Status register, and disable interrupt on kernel mode

Restore Context — * Restore the saved registers

ERET — * Close the exception handler by ERET instruction

Table 8.18.2 General Exception Servicing Guidelines (SW)
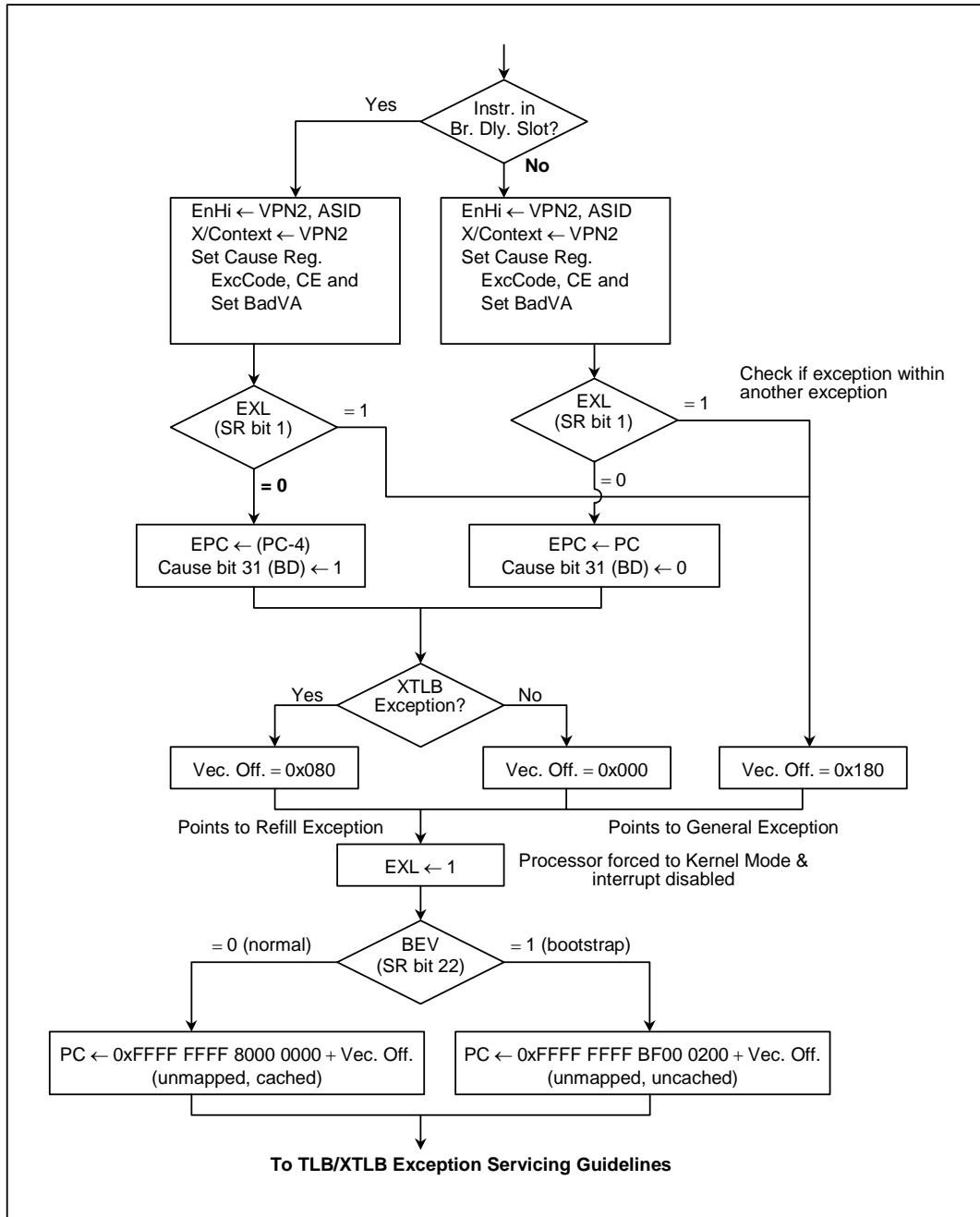
Table 8.18.3 TLB/XTLB Miss Exception Handler (HW)

Comments

MFC0 -

CONTEXT

* Unmapped vector TLBMod, TLBInv,
  TLB Refill exceptions not possible

* EXL = 1 so Interrupt exceptions disabled

* OS/System to avoid all other exceptions

* Only Cold Reset, Soft Reset, NMI exceptions
  possible.

Service Code

* Load the mapping of the virtual address in Context Reg.
  Move it to ENLO and Write into the TLB

* There could be a TLB miss again during the mapping
  of the data or instruction address.   The processor will
  jump to the general exception vector since the EXL is 1.
  (Option to complete the first level refill in the general
  exception handler or ERET to the original instruction
  and take the exception again)

ERET

* ERET is not allowed in the branch delay slot of
  another Jump Instruction

* Processor does not execute the instruction which is
  in the ERET's branch delay slot

* PC ← EPC; EXL = 0

* LLbit ← 0

Table 8.18.4 TLB/XTLB Exception Servicing Guidelines (SW)

Cold Reset & NMI Exception Handling (HW)

Cold Reset & NMI Servicing Guidelines (SW)

NMI Exception

Status:

BEV ← 1
TS ← 0 (*)
SR ← 1
ERL ← 1

Cold Reset Exception

Random ← TLBENTRIES-1
Wired ← 0
Status:

BEV ← 1
TS ← 0 (*)
SR ← 0
ERL ← 1

Instr. in
Br. Dly. Slot? → Yes

No

ErrorEPC ← PC   ErrorEPC ← PC - 4

PC ← 0xFFFF FFFF BFC0 0000

NMI?   Yes

No

Note: There is no indication from the
proessor to differentiate between
NMI & Soft Reset;
there must be a system level indication.

NMI Service Code

Status bit 20
(SR)   = 0

= 1

(Optional)   ERET

Soft Reset Service Code   Cold Reset Service Code

(*) Reserved for TX49

Table 8.18.5 Cold Reset, Soft Reset & NMI Exception Handling (HW) and

Servicing Guidelines (SW)

9.   Initialization Interface

This chapter describes the TX4951B Initialization interface, and the processor modes. This includes the reset signal description and types, and initialization sequence, with signals and timing dependencies, and the user-selectable TX4951B processor modes.

Signal names are listed in bold letters—for instance the signal **MasterClock** indicates the processor clock. Low-active signals are indicated by a trailing asterisk, such as **ColdReset***, the power-on/cold reset signal.

9.1   Functional Overview

The TX4951B processor has the following two types of resets; they use the **PLLReset*** and **ColdReset*** input signals.

- **PLL Reset** is asserted to initialize the clock Generator.
- **Cold Reset** is asserted after the power supply is stable and then restarts all clocks. A cold reset completely reinitializes the internal state machine of the processor without saving any state information.

After reset, the processor is bus master and drives the *SysAD* bus.

For reset vector address, use 0xbfc0 0000.

In the TX4951B processor core, the reset vector is located in address space without caching. Therefore, the cache and TLB need not be initialized at reset processing.

9.1.1   System Coordination

Care must be taken to coordinate system reset with other system elements. In general, bus errors immediately before, during, or after a reset may result in unpredicted behavior. Also, a small amount of processor state is guaranteed as stable after a reset of the TX4951B processor, so extreme care must be taken to correctly initialize the processor through software.

The operation of each type of reset is described in sections that follow. Refer to Figure 9.2.1 and Figure 9.2.2 later in this chapter for timing diagrams of the cold and warm resets.

9.2   Reset Signal Description

This section describes the two reset signals, **PLLReset*** and **ColdReset***.

- **PLLReset**:   Assertion of **PLLReset*** initializes the on-chip PLL. **PLLReset*** should be asserted, for example, when the system power is turned on.

- **ColdReset***: The **ColdReset*** signal must be asserted[Note] (low) to reset the processor. Internal clock begins to cycle and is synchronized with the deasserted edge (high) of **ColdReset***. **ColdReset*** can be asserted and deasserted asynchronously with the rising edge of **MasterClock**.

Note: Asserted means the signal is true, or in its valid state. For example, the low-active Reset* signal is said to be asserted when it is in a low (true) state.

9.2.1   Power-On Reset

Power-on reset is a reset that occurs when the system power is turned on. This reset initializes the PLL inside the on-chip Clock Generator. After **MasterClock** became stable and the PLL stabilization time has elapsed, **PLLReset*** must be deasserted and then **ColdReset*** must be deasserted. Figure 9.2.1 illustrates the power-on reset timing.
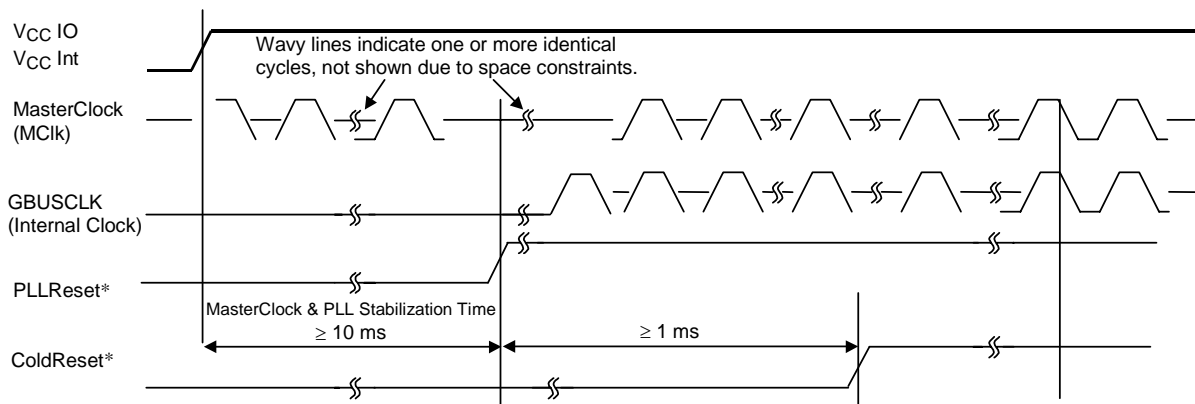
Note: PLLReset* need not be synchronized to MasterClock.
PLLReset* is sampled by the on-chip logic.

Figure 9.2.1 Power-On Reset Timing

9.2.2   Cold Reset

A cold reset is used to completely reset the processor, including the processor clock. Processor states should be saved as follows.

After the processor power has stabilized, **ColdReset*** must be kept asserted for a minimum of 16 **MasterClock** cycles to ensure that the processor clock locks with respect to the **MasterClock** input. **ColdReset*** may be asserted and deasserted asynchronously from the rising edge of **MasterClock**.

While **ColdReset*** is asserted, the processor assumes bus ownership and drives the **SysAD** bus, as follows:

	In R5000 Mode	In R4300 Mode
SysAD:	32'hXXXXXXXX (unknown output)	32'hXXXXXXXX (unknown output)
SysCmd:	9'b111010000 (output)	9'b111010000 (output)
ValidOut*:	1'b1 (output)	1'b1 (output)
Release*:	1'b1 (output)	1'b1 (output)

After **ColdReset*** is deasserted, the processor branches to the Reset Exception vector and begins executing the Cold Reset Exception handler. Information about the CPU register bits during a cold reset is provided in the chapter on CPU exception processing.

When **ColdReset*** is asserted during a **SysAD** transfer, all external agents must be reset to avoid bus contention on the **SysAD** bus. Figure 9.2.2 illustrates the cold reset timing.
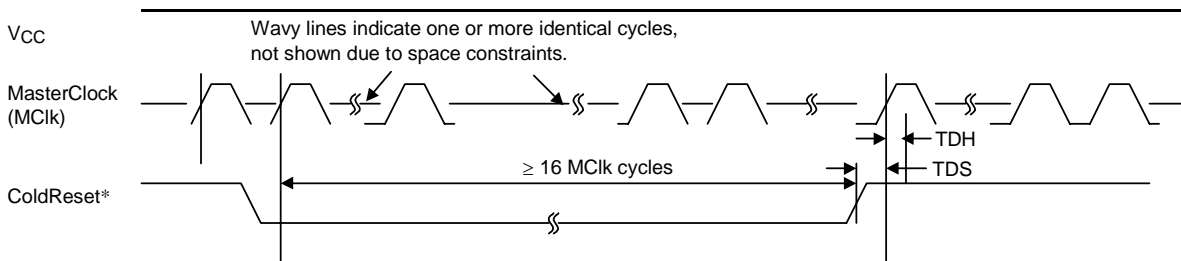
Figure 9.2.2 Cold Reset

9.3   User-Selectable Mode Configurations

The TX4951B supports several user-selectable modes, which are designated during initialization.

9.3.1   System Bus Interface Modes

The TX4951B provides two system bus (**SysAD**) interface modes: R4300 mode and R5000 mode. The **MODE43*** signal selects which interface mode is used.

9.3.2   Clock Divisor for the System Bus

The **DivMode[1:0]** signals specify the frequency relationship between the system bus (**SysAD**) interface and the CPU core. The TX4951B supports four divide ratios: 2, 2.5, 3 and 4.

9.3.3   System Endianness

The value of the **Endian** signal when **ColdReset*** is released (High) controls the system endianness: 0 for little-endian and 1 for big-endian. The *BE* bit in the *Config* register is read-only. Setting the *RE* bit in the *Status* register reverses the User-mode endianness.

9.3.4   Enabling and Disabling the Timer Interrupt

The Timer interrupt, an internal interrupt of the TX49/L3 core, can be enabled and disabled through the **TintDis*** signal:

0: Enabled

1: Disabled

10. Clock Interface

This chapter describes the clock signals ("clocks") used in the TX4951B processor.

The subject matter includes basic system clocks, system timing parameters, operating the TX4951B processor in reduced power (RP) mode, connecting clocks to a phase-locked system, and connecting clocks to a system without phase locking.

10.1 Signal Terminology

The following terminology is used in this chapter (and book) when describing signals:

- *Rising edge* indicates a low-to-high transition.
- *Falling edge* indicates a high-to-low transition.
- *Clock-to-Q delay* is the amount of time it takes for a signal to move from the input of a device (*clock*) to the output of the device (*Q*).

Figure 10.1.1 and Figure 10.1.2 illustrate these terms.
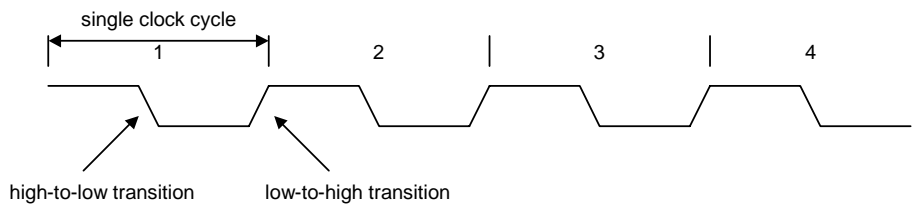
Figure 10.1.1 Signal Transitions
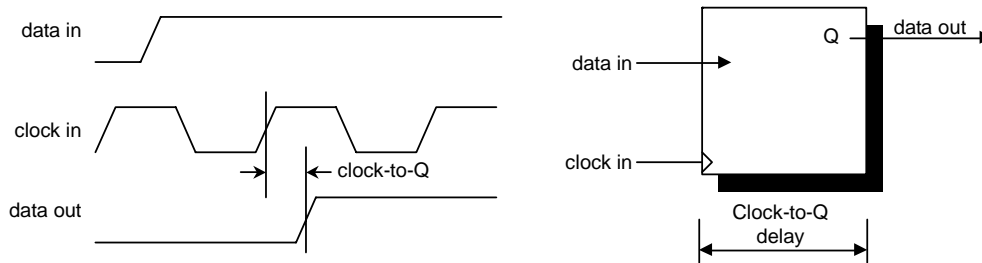
Figure 10.1.2 Clock-to-Q Delay

10.2 Basic System Clocks

The various clock signals used in the TX4951B processor are described below, starting with **MasterClock**, upon which the processor bases all internal and external clocking.

The clocks on the TX4951B processor are controlled by an on-processor Phase-locked Loop (PLL) circuit. This circuit keeps the TX4951B processor's internal clock edges aligned with the clock edges of the **MasterClock** signal, which itself acts as the master system clock.

Inside the TX4951B processor, the **MasterClock** signal can be multiplied by a factor set by the **DivMode[1:0]** inputs to the processor. All internal clocks are then derived from this clock. The TX4951B processor has two primary internal clocks, the pipeline (also referred to as *processor*) clock, **CPUCLK**, and the system interface clock, **GBUSCLK**. **GBUSCLK** has the same frequency and phase as **MasterClock**.

10.2.1 MasterClock

The Processor bases all internal and external clocking on the single **MasterClock** input signal. **MasterClock** specifications are shown in Figure 10.2.1.
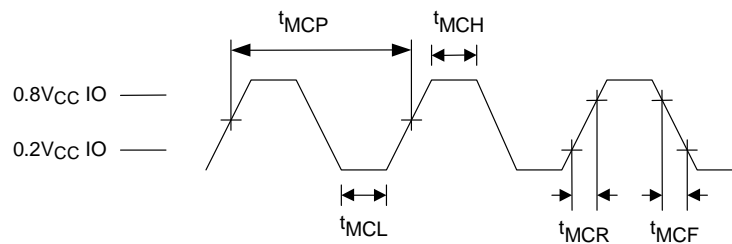
Figure 10.2.1 MasterClock

10.2.2 CPUCLK

The pipeline (or *processor*) clock, **CPUCLK**, can be 2, 2.5, 3, 4 times the **MasterClock** frequency. This multiplication factor is determined by **DivMode[1:0]** pins, which are static signal inputs to TX4951B.

All internal registers and latches use **CPUCLK**.

10.2.3 GBUSCLK

The system interface clock, **GBUSCLK**, is the same as the **MasterClock** frequency. **GBUSCLK** is always derived from **CPUCLK.** The TX4951B processor drives its outputs on this clock edge.

The first rising edge of **GBUSCLK**, after **ColdReset*** is deasserted, is aligned with the first rising edge of **MasterClock**.

10.2.4 CPUCLK-to-GBUSCLK Division

Figure 10.2.2 shows the clocks for a CPUCLK-to-GBUSCLK division by 2.
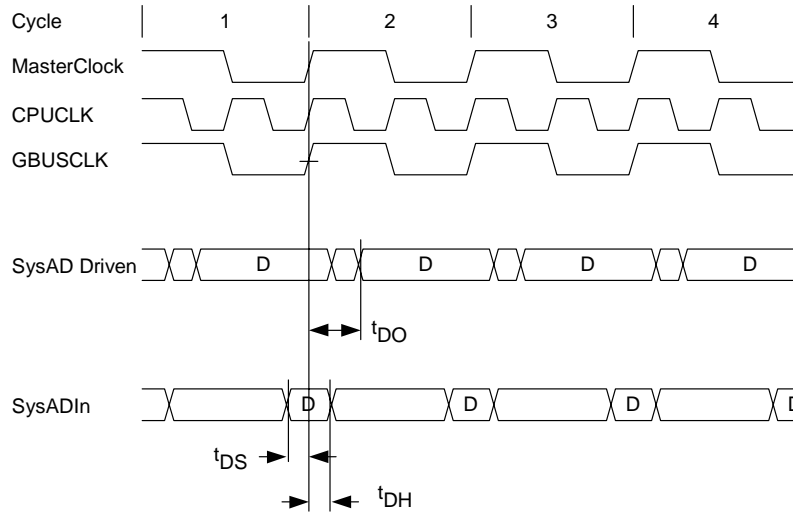
Figure 10.2.2 Processor Clock, CPUCLK-to-GBUSCLK Divisor of 2

10.2.5 Phase-Locked Loop (PLL)

The TX4951B clocks are controlled by a Phase-locked Loop circuit (PLL).

10.3 Connecting Clocks to a Phase-Locked System

When the processor is used in a phase-locked system, the external agent must phase lock its operation to a common **MasterClock**. In such a system, the delivery of data and data sampling have common characteristics, even if the components have different delay values. For example, *transmission time* (the amount of time a signal takes to move from one component to another along a trace on the board) between any two components A and B of a phase-locked system can be calculated from the following equation:

$$\text{Transmission Time} = (\text{GBUSCLK period}) - (t_{DO} \text{ for A}) - (t_{DS} \text{ for B}) -$$
$$(\text{Clock Jitter for A Max}) - (\text{Clock Jitter for B Max})$$

Figure 10.3.1 shows a block-level diagram of a phase-locked system using the TX4951B processor.
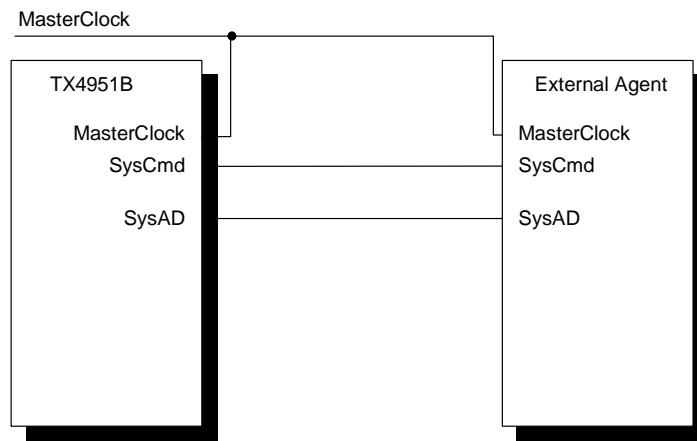
Figure 10.3.1 The TX4951B Processor Phase-Locked System

11. TX4951B System Interface

11.1 Terminology

The following terms are used in this section.

- External agent: Logic device that is directly connected to the processor via the system interface so a processor can issue (instructions).

- System event: Event issued inside a processor which, when generated, means that access to external system resources is required.

- Sequence: Strict order of requests that the processor generates in order to provide service for system events.

- Protocol: Shift of signals for each cycle generated on the system interface so processor requests or external requests can be asserted.

- Syntax: Strict definition of the bit pattern on the encoded bus (command bus, etc.).

11.2 Explanation of System Interface of R5000 type protocol mode

In TX4951B, it is built in system interface function corresponding to R4300 type protocol. A selection of above-mentioned R5000 type protocol mode or R4300 type protocol mode increases by external pin (MODE43* : 117 pin).

MODE43* = 0: R4300 type protocol

MODE43* = 1: R5000 type protocol

The TX4951B processor supports 32-bit address/data interfaces. This processor makes it possible to construct a processor system by processors and main memory. System interfaces consist of the following components:

- 32-bit address/data bus, SysAD

- 9-bit command bus, SysCmd

- 6 handshake signals
    - RdRdy*, WrRdy*
    - ExtRqst*, Release*
    - ValidIn*, ValidOut*

The TX4951B processor accesses external resources using the system interface in order to correct cache misses, uncached operation, and other problems.

11.2.1 Interface bus

Figure 11.2.1 illustrates the 32-bit address/data bus SysAD[31:0], which is the main communication bus of the system interface, and the 9-bit command bus SysCmd[8:0]. SysAD and SysCmd are bidirectional busses. In other words, these two busses are used for the processor to issue processor requests and for the external agent to issue external requests.
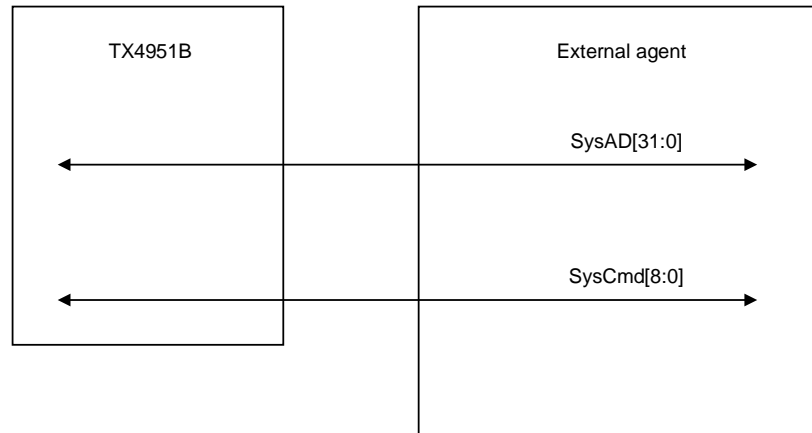
Figure 11.2.1 System Interface Bus

Requests sent via the system interface consist of the following:

- Address
- System interface command that strictly specifies the type of request
- Series of data elements for when the request is a particular write or read process.

11.2.2 Address cycle and data cycle

Cycles during which valid addresses exist on the SysAD bus are referred to as address cycles. Also, cycles during which valid data exist on the SysAD bus are referred to as data cycles. Validity is determined depending on the ValidIn signals and ValidOut signals.

The SysCmd bus is used to identify the contents of the SysAD bus for all cycles at which it is to be valid. The most significant bit of the SysCmd bus is used to indicate whether the current cycle is an address cycle or a data cycle.

- In the case of an address cycle [SysCmd[8] = 0], the remaining bits SysCmd[7:0] of the SysCmd bus contain the system interface commands.
- In the case of a data cycle [SysCmd[8] = 1], the remaining bits SysCmd[7:0] of the SysCmd bus contain the data identifier.

11.2.3 Issue cycle

Two types of processor issue cycles exist with the TX4951B.

- Processor read request issue cycles.
- Processor write request issue cycles.

The TX4951B judges the issue cycle of the processor read request by sampling the RdRdy* signal. It also judges the issue cycle of the processor write request by sampling the WrRdy* signal from the external agent.

As illustrated in Figure 11.2.2, RdRdy* must be asserted two cycles before the processor read request address cycle in order to define the address cycle as an issue cycle.
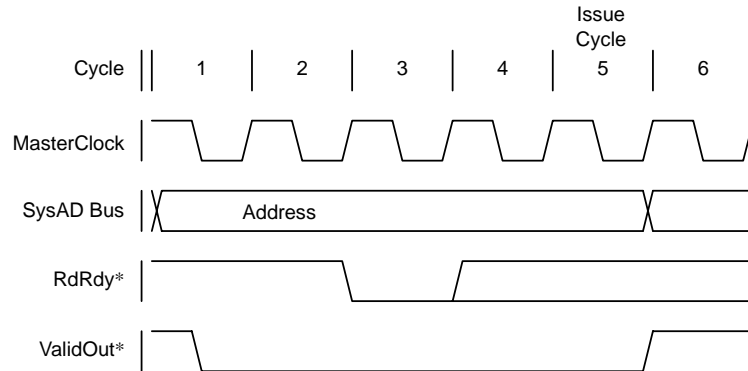
Figure 11.2.2 RdRdy* Signal Status in case of Read Request

As illustrated in Figure 11.2.3, WrRdy* is asserted two cycles before the initial address cycle of the processor write request, and the address cycle must be defined as the issue cycle.
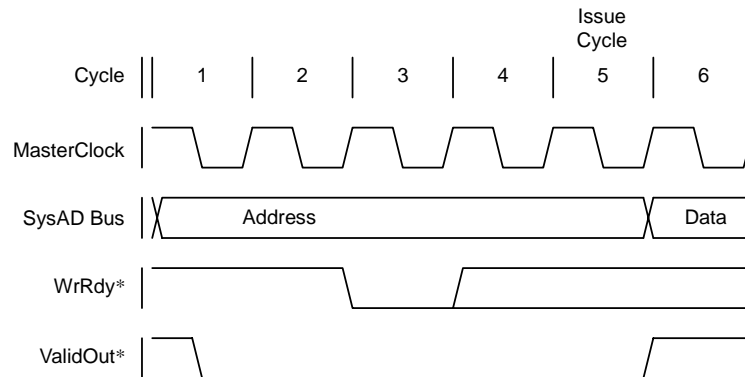
Figure 11.2.3 WrRdy* Signal Status in case of Write Request

The TX4951B repeats the request address cycle until the conditions of the valid issue cycle are met. If the processor request is a data transmission, then data transmission starts at the point when the issue cycle is complete. There is only one issue cycle no matter what the processor request is.

The TX4951B accepts external requests even while trying to issue processor requests. If the external agent asserts ExtRqst*, the processor responds to the external agent by releasing the system interface and going into the slave state. Rules relating to the issue cycle of processor requests are strictly applied in determining the processor run operation as well. The TX4951B performs one of the following:

- Complete issuing of processor requests before external requests are received.
- Release the system interface and go into the slave mode without completing issuance of the processor requests.

In the latter of the above situations, the TX4951B issues processor requests after external requests are complete. Rules relating to issuing are also provided to processor requests.

11.2.4 Handshake signal

The processor uses the eight control signals explained below to manage the flow of requests.

- RdRdy* and WrRdy* are used by the external agent to indicate that it is ready to accept a new read or write transaction.
- ExtRqst* and Release* are used to transfer SysAD bus and SysCmd bus control. ExtRqst* is used by the external agent to indicate the necessity of controlling the interface. Release* is asserted by the processor when transferring the system interface access privileges.
- The TX4951B processor uses ValidOut* and the external agent ValidIn* signals to indicate the valid command/data on the SysCmd/SysAD bus.

11.2.5 System Interface Protocol of R5000 type

Figure 11.2.4 illustrates the system interface that operates between registers. In other words, processor output is directly transferred from the output register and changes and the MasterClock rising edge.

Processor input is directly transferred to the input register and the input register latches these input signals at the rising edge of the MasterClock. In this way, it becomes possible for the system interface to operate at the fastest clock frequency.

Figure 11.2.4 Operation of the System Interface Between Registers

11.2.5.1 Master state and slave state

The system interface is placed in the master state when the TX4951B processor is driving the SysAD bus and SysCmd bus. In contrast, the system interface is in the slave state when the external agent is driving the SysAD bus and SysCmd bus.

The processor asserts the ValidOut* signal if the SysAD bus and SysCmd bus become valid when the system interface is in the master state. The external agent asserts the ValidIn* signal if the SysAD bus and SysCmd bus become valid when the system interface is in the slave state.

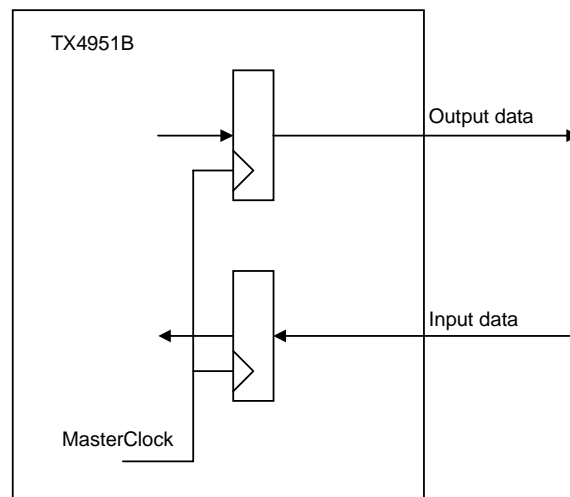### 11.2.5.2 Shifting from the master state to the slave state

The system interface remains in the master state unless it enters one of the following states:

- The external agent issues a request, then usage of the system interface is granted (external arbitration).

- The processor issues a read request and shifts into the slave mode by itself.

11.2.5.3 External arbitration

The external agent cannot issue external requests via the system interface unless the system interface goes into the slave state. Shifts from the master state to the slave state are arbitrated by the processor using the system interface handshake signals ExtRqst* and Release.* This shift is performed as follows below.

1) The external agent sends notification that it would like to issue an external request by asserting the ExtRqst* signal.

2) The processor releases the system interface and changes its state from the master state to the slave state by asserting the Release* signal for 1 cycle.

3) The system interface returns to the master state when issuing of the external request is complete.

11.2.5.4 Shifting to the slave state on its own

Shifting to the slave state on its own means that the shift from the master state to the slave state is started by the processor when the processor read request is still on hold. The Release signal is automatically asserted after the read transaction. Self-invoked shifting to the slave state occurs either during the issue cycle of the read request or several cycles after that.

After shifting to the slave state on its own, the processor returns to the master state at the end of the next external request. This is made possible by a read request or other type of external request.

The SysAD bus and SysCmd bus drives must start after the external agent confirms that the processor autonomously shifted to the slave state. While the system interface is in the slave state, the external agent can start making external requests without requesting access to the system interface (without asserting the ExtRqst* signal).

The system interface returns to the master state when the external request ends.

If a processor read request is on hold after a read request is issued, the processor automatically changes the system interface into the slave state even if the system interface access necessary for the system agent to issue the external request has not been requested. By shifting to the slave state in this manner, the external agent becomes able to return read response data.

11.2.6 Processor Requests and External Requests

Requests are broadly categorized as processor requests and external requests. This section will describe these two categories.

When a system event is generated, either a single request or a series of requests (referred to as processor requests) are issued via the system interface so the processor can access an external resource and invoke the service for the event. In order for this operation to be performed properly, the processor system interface must be connected to a system agent that meets the two following conditions:

1) It is in compliance to the system interface protocol.

2) It can regulate access to system resources.

An external agent that requests access to the processor cache or the status registers generates an external request. This access request is transferred via the system interface. Figure 11.2.5 illustrates the system event and request cycles.
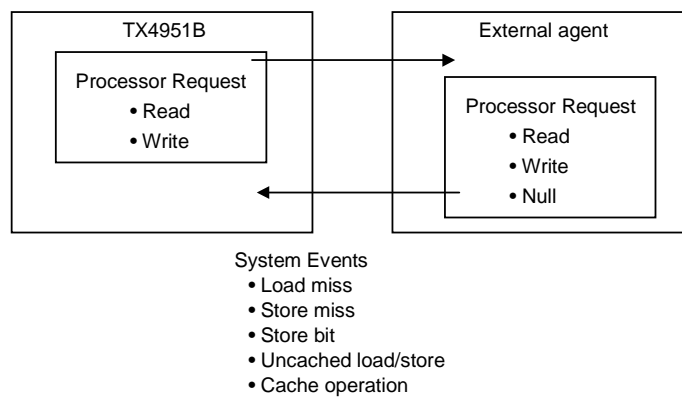
Figure 11.2.5 Requests and System Event

11.2.6.1 Rules relating to processor requests

The following rules apply to processor requests.

- After a processor read request is issued, the processor cannot issue the next read request until after it receives a read response.

- When in the R4000 compatible mode, after a write request is issued, at least 4 cycles must pass from when the write request issue cycle is complete until the processor can issue the next request. This is because two dummy system cycles are inserted as illustrated in Figure 11.2.6 by consecutive write requests of single data cycles.
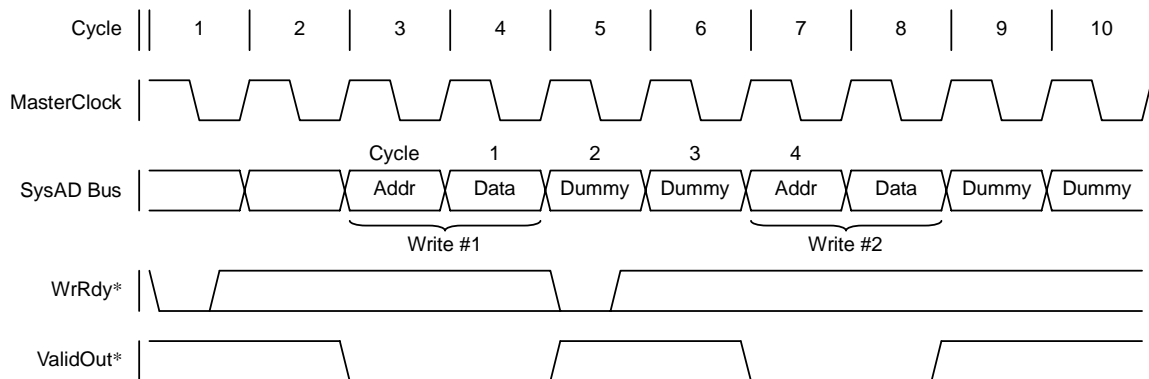
Figure 11.2.6 Timing of Consecutive Write Cycles

11.2.6.2 Processor requests

The term "processor request" refers to either a single request or a series of requests issued via the system interface in order to access external resources. As illustrated in Figure 11.2.7, there are two types of processor request: read and write.
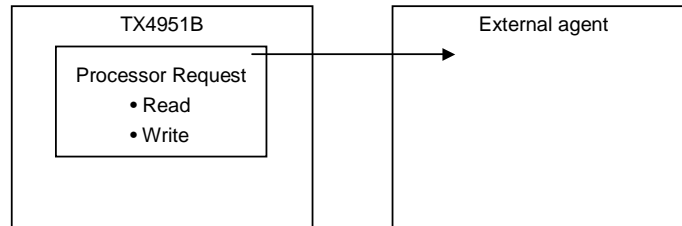
Figure 11.2.7 Processor Requests

Read requests are requests that read data from main memory or other memory resources in block double word, partial double word, word, and partial word units.

Write requests are requests that write data to main memory or other system resources in block, double word, partial double word, word, and partial word units.

Processor requests are managed by the TX4951B processor in the same manner as the R4000/R4400 non-secondary cache mode.

The processor issues requests strictly according to a sequential method. In other words, the processor cannot issue the next request while a previous request is on hold. For example, after issuing a read request, the processor waits for a read response before issuing the next request. The processor only issues write requests when there are no read requests on hold.

When using processor input signals RdRdy* and WrRdy*, the external agent can control the processor request flow. RdRdy* is the signal that controls the processor read flow, and WrRdy* controls the processor write request flow. Figure 11.2.8 illustrates the sequence of the processor request cycle.
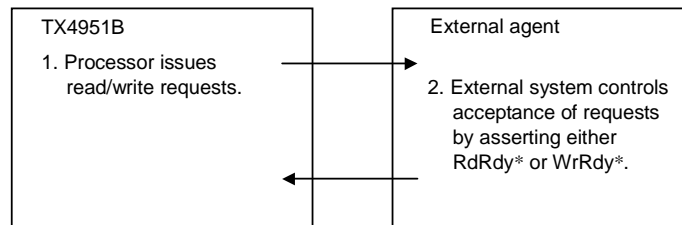
Figure 11.2.8 Processor Requests

11.2.6.3 Processor read requests

When the processor issues a read request, the external agent must access the specified resource and return the requested data.

The external agent returns response data for processor read requests so they can be executed separately from the requests. In other words, the external agent can start an external request before returning response data for the processor read request. A processor read request is complete when the final word of the response data is received from the external agent.

Depending on the data identifier combined with the response data, an error in the response data may be pointed out. The processor would then treat this error as a bus error.

If data have not been returned to the issued processor read request, the applicable request is said to be "on hold." This state continues until the requested read data are returned.

The external agent must be able to accept processor read requests at any time if either of the two following conditions is met.

- There is no processor read request that is on hold.
- The RdRdy* signal is asserted for 1 cycle 2 cycles before the issue cycle.

11.2.6.4 Processor write request

When the processor issues a write request, the specified resources are accessed, then the data are written to those resources.

Processor write requests are complete when the final data word is transferred to the external agent.

The external agent must be able to accept processor write requests at any time if either of the two following conditions are met.

- There is no processor read request that is on hold.
- The WrRdy* signal is asserted for 1 cycle 2 cycles before the issue cycle.

11.2.6.5 External requests

As illustrated in Figure 11.2.9, there are three types of external request: read, write, and null. This section will explain read responses, which are special external request cases.
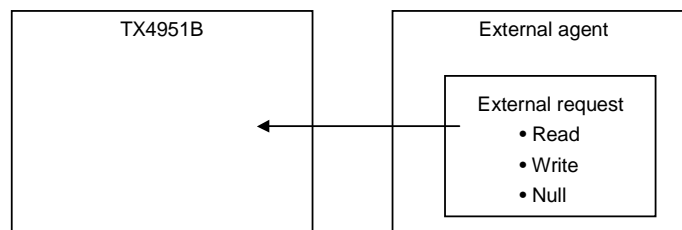
Figure 11.2.9 External Requests

Read requests are used to call 1-word data from processor internal resources. Write requests are used to write 1-word data to the processor internal resources. Null requests are requests that do not require processor operation.

As illustrated in Figure 11.2.10, the processor uses arbitration signals ExtRqst* and Release* to control the flow of external requests. The external agent cannot issue external requests unless access privileges to the system interface are obtained. In order to do so, the external agent asserts the ExtRqst* signal, then waits until the processor asserts the Release* signal for 1 cycle.
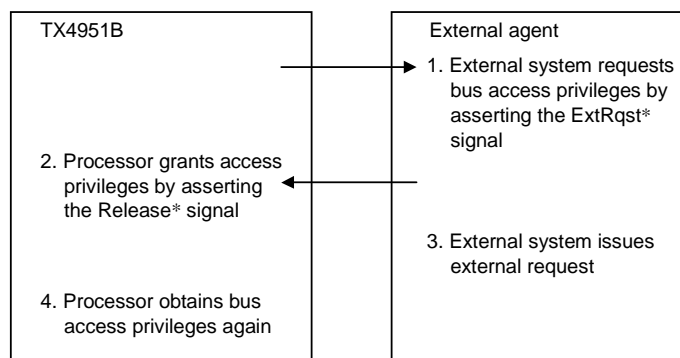
Figure 11.2.10 External Requests

After the external request is issued, the system interface access privileges always return to the processor. The processor will not accept another external request until the current one is complete.

If there is no processor request that is on hold, the processor decides based on the interior state whether to accept an external request or to issue a new processor request. The processor can issue a new processor request even if the external agent requested access to the system interface.

The external agent sends notification that it would like to start an external request by asserting the ExtRqst* signal. After that, the external agent waits for the processor to assert the Release* signal and send notification that preparations have been made to accept this request. The processor sends notification based on the next judgement criterion to be listed that preparations have been made to accept an external request.

- The processor ends processor requests that are in progress.
- The processor can accept an external request while waiting for the RdRdy* signal to be asserted so a processor read request can be issued. However, this request must be transferred to the processor at least 1 cycle before the RdRdy* signal is asserted.
- The processor can accept an external request while waiting for the WrRdy* signal to be asserted so a processor write request can be issued. However, this request must be transferred to the processor at least 1 cycle before the WrRdy* signal is asserted.
- If waiting for a response to a read request after the processor shifted itself to the slave state, the external agent can issue an external request before sending read response data.

11.2.6.6 External read requests

In contrast to processor read requests, data are directly returned as a response to the request for external read requests. No other requests can be issued until the processor returns the requested data. External read requests are complete when the processor returns the requested data word. Depending on the data identifier combined with the response data, an error in the response data may be pointed out. The processor would process the error as a bus error.

Note: The TX4951B does not have any resources that can read external read requests. The processor returns to the external read request undefined data and data identifiers in which SysCmd[5] of the errant data bit is set.

11.2.6.7 External write requests

When the external agent issues a write request, the specified resources are accessed, then the data are written to those resources. External requests are complete when the data word is transferred to the processor.

The only processor resource that an external write request can use are the Interrupt registers.

11.2.6.8 Read responses

As illustrated in Figure 11.2.11, read responses return data to processor read requests. Read responses are external requests, strictly speaking, but there is only one difference with other external requests: read responses do not request permission to use the system interface. Therefore, read responses are handled separately from other external requests and are simply referred to as read responses.
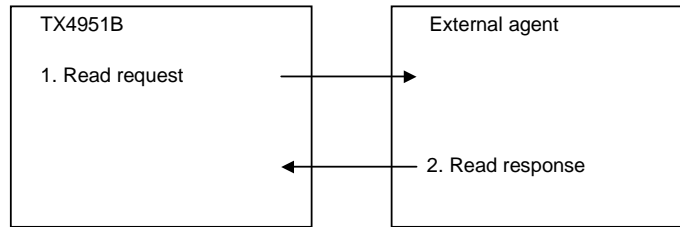
Figure 11.2.11 Read Response

11.2.7 Handling of Requests

This section will describe in detail sequences, protocol, and syntax for both processor and external requests.

- Load miss
- Store miss
- Store bit
- Cache operation
- Load Linked/Store Conditional

11.2.7.1 Load miss

If a processor load miss occurs in the primary cache, the processor cannot proceed to the next process if the cache line that contains the loaded data elements is not received from the external agent.

If the current cache line set in which the write back bit (W bit) is set is replaced by a new cache line, the current cache line must be written back.

The processor checks the coherency properties in the TLB entries for pages including the requested cache lines. If the coherency properties are non-coherent, then a non-coherent read request is issued. Table 11.2.1 indicates the measures that can be taken when a load miss occurs in the primary cache.

Table 11.2.1 Load Miss to the Primary Cache

Page Properties	State of the replaced data cache line	
	Dirty (W = 0)/Invalid	Dirty (W = 1)
Non-coherent	NCR	NCR/W

NCR: Processor non-coherent, block read request

NCR/W: Processor non-coherent, block write requests continue after the block read request

11.2.7.2 Store miss

When a store miss occurs in the primary cache, the processor cannot proceed to the next process if it does not receive from the external agent a cache line that includes a store target address. The processor checks the coherency properties in the TLB entries for pages including the requested cache line, then confirms whether to invalidate write transactions to that cache line or not.

After that, the processor executes one of the following requests:

- If the coherency properties are non-coherent write back or non-coherent write through (write allocate), then a non-coherent block read request is issued.

- If the coherency properties are non-coherent write through (non-write allocate), then a non-block write request is issued. Table 11.2.2 indicates the measures taken when there is a store miss to the primary cache.

Table 11.2.2 Store Miss to Primary Cache

Page Properties	State of the replaced data cache line	
	Dirty (W = 0)/Invalid	Dirty (W = 1)
Non-coherent write back or non-coherent write through (write allocate)	NCR	NCR/W
Non-coherent write through (non-write allocate)	NCW	NA

NCR: Processor non-coherent, block read request
NCR/W: Processor non-coherent, block write requests continue after the block read request
NCW: Processor non-coherent write request

11.2.7.3 Store hits

Operation in the system interface is determined by whether a line is write back or write through. When in the primary cache mode, all lines set to write back are set to the dirty exclusion state (W = 1). In other words, burst transactions do not occur even if a store hit occurs. Lines set to write through generate processor write requests for store data.

11.2.7.4 Uncached load or store

When performing uncached load operations, the processor issues non-coherent read requests for double words, partial double words, words, or partial words. Also, when performing uncached store operations, the processor issues write requests for double words, partial double words, words, or partial words.

The TX4951B judges that there is valid parity and data in the entire 32-bit SysAD bus even for data requests of less than words. Even if there was a partial word request for example, all parity must be correctly returned for all 32 bits. If not, then parity check must be disabled.

All write transactions by the TX4951B are buffered in the 4-stage write buffer of the system interface. If there are entries in the write buffer when a block request is required, the write buffer is flushed before a read request is generated (for cache misses or read transactions to uncached areas). Data cache misses or uncached data load transactions flush the write buffer.

11.2.7.5 Cache instruction operation

Various operations are made available to the Cache instruction in order to maintain the primary cache status and contents. When Cache instruction operations are in progress, write requests or invalidate requests can be issued from the processor.

11.2.8   Processor Request and External Request Protocol

This section explains the bus arbitration protocol for both processor requests and external requests on a cycle-by-cycle basis. Table 11.2.3 below describes the abbreviations used in the following timing diagram of the bus.

Table 11.2.3 System Interface Request

Range	Abbreviation	Meaning
Total	Unsd	Unused
SysAD bus	Addr	Physical address
	Data<n>	Data number *n* of the data block
SysCmd bus	Cmd	Undefined system interface command
	Read	Processor or external read request command
	Write	Processor or external write request command
	SINull	External null request command that releases the system interface
	NData	Non-coherent data identifier for datum other than the final datum
	NEOD	Non-coherent data identifier for the final datum

11.2.8.1 Processor request protocol

Processor request protocol is as follows.

- Read
- Write
- Null

11.2.8.2 Processor read request protocol

The processor read request protocol is as described in the following sequence. The next step numbers correspond to the numbers in Figure 11.2.12.

1. RdRdy* is asserted to Low by the external agent. This means that the external agent is ready to accept read requests.

2. When the system interface is in the master state, the read command is transmitted to the SysCmd bus, then the processor read request is issued by transmitting the read address to the SysAD bus.

3. At the same time, the processor asserts the ValidOut* signal for one cycle. This means that valid data are being transmitted to the SysCmd bus and SysAD bus.

4. The processor goes into the slave state by itself either at the issue cycle of a read request or after the Release* signal is asserted for one cycle and the issue cycle of the read request is complete.

   Note: The external agent must not assert the ExtRqst∗ signal as a means of returning a read response. It must however wait to shift to the slave state on its own. If an external request other than a read response is issued, ExtRqst∗ can be asserted either before the read response or in the process of the read response.

5. The SysCmd bus and SysAD bus are released from the processor one cycle after the Release* signal is asserted.

6. The SysCmd bus and SysAD bus are driven by the external agent within two cycles after the Release* signal is asserted.

When shifting to the slave state (from Cycle 5 in Figure 11.2.12), the external agent can return the requested data as a read response. Notification of an error in the returned data is sent if either the data requested by a read response were returned or if the requested data could not be fetched. In this case, the processor handles the result as a bus error exception.

Figure 11.2.12 illustrates a situation in which the slave state is autonomously shifted to after a processor read request is issued.

Note: The timing of the SysADC bus and SysCmdP bus are the same as the timing of the SysAD bus and SysCmd bus timing, respectively.
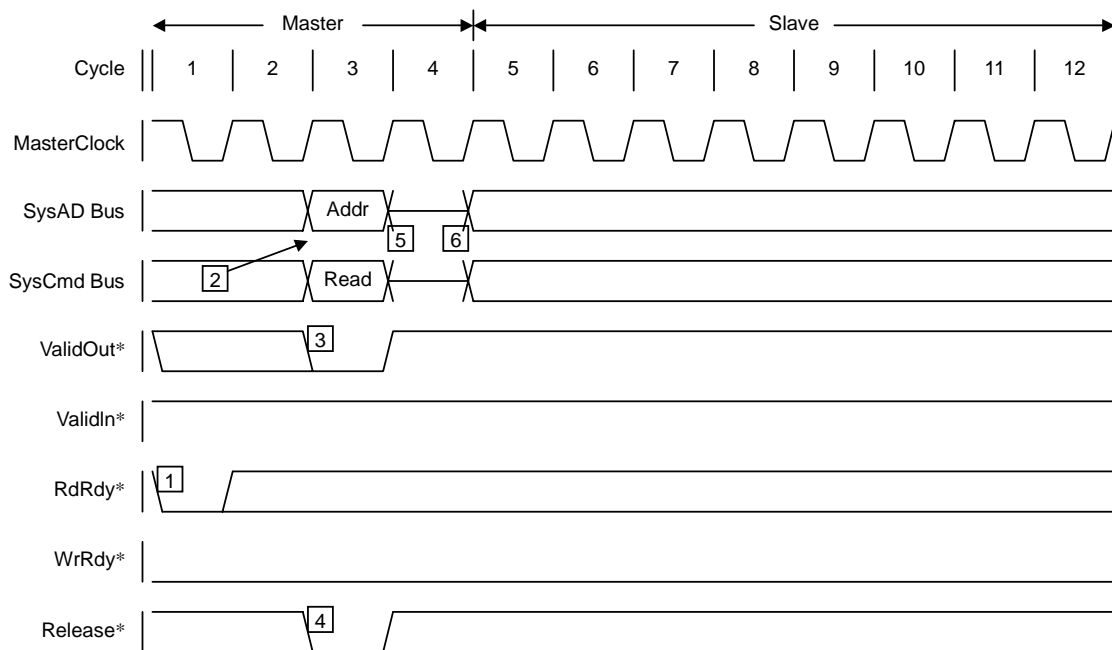
Figure 11.2.12 Processor Read Request Protocol

If the Release* signal is asserted, this means that either there is autonomous shifting to the slave state or there is a response to the ExtRqst* signal assertion. In this case, the processor can accept either a read response or an external request other than a read response. If an external request other than a read response is issued, the processor asserts Release* for 1 cycle, then autonomously shifts to the slave state again after the external request process.

11.2.8.3 Processor write request protocols

Either of the two following protocols is used in issuing processor write requests.

- The word write request protocol (see Note below) is used for double word, partial double word, word or partial word writing.

  Note: Words are called to differentiate from the block request protocol. It is actually possible to transfer data in double word, partial double word, word, or partial word units.

- The block write request protocol is used for block write transactions.

The system interface is used in the master state to issue processor double word write requests. Figure 11.2.13 illustrates processor non-coherent single word write request cycles.

1. In order to issue a processor single word write request, a write command is sent to the SysCmd bus, and a write address is sent to the SysAD bus.

2. The processor asserts the ValidOut* signal

3. The processor sends the data identifier to the SysCmd bus and transmits data to the SysAD bus.

4. The data identifier for this data cycle must receive an indication that this is the final data cycle. ValidOut* is deasserted at the end of the cycle.
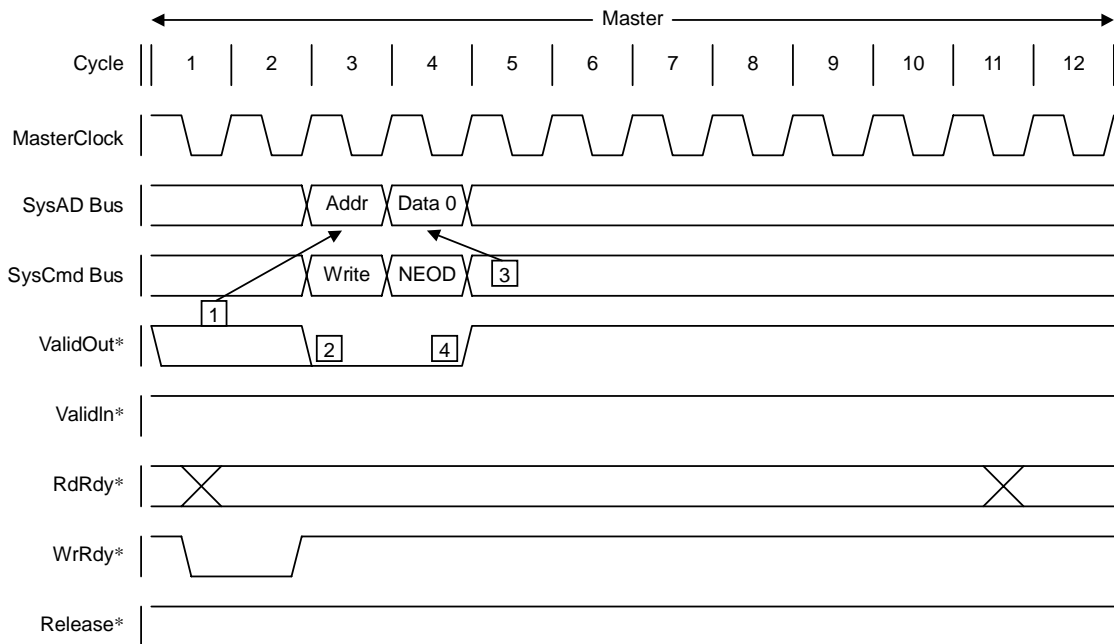
Figure 11.2.13 Processor Non-coherent Single Word Write Request Protocol

11.2.8.4 Processor single write requests

There are three processor single write requests as follow below.

With later G2SConfig-Regster, these modes are selected.

1. R4000 compatible write

2. Reissue write

3. Pipeline write

Table 11.2.4 Data Transfer Rate, Data Pattern and Setting at single write requests

Maximum Data Transfer Rate	Data Pattern	Setting bits 2-1	Write mode
1 word/3 MasterClock cycle	Wxx	00	R4000 compatible
Reserved	Reserved	01	Reserved
1 word/1 MasterClock cycles	W	10	Pipeline write
1 word/1 MasterClock cycles	W	11	Reissue write

Note: The setting bits 2-1 is bits 2-1 of the G2Sconfig register (0xF FF10 0000).
      Bits 2-1 is set to "00" when initialized.

1. R4000 compatible write
When in the R4000 compatible write mode, 4 cycles are required for single write operation. After the address is asserted for 1 cycle, it is followed by 2 cycles of dummy data. Figure 11.2.14 illustrates its basic operation.

In the case of the TX4951B, the WrRdy* signal must be asserted for 1 cycle 2 cycles before the write operation is issued. When in the R4000 compatible signal write mode, the external agent receives the write data then immediately asserts WrRdy*, making it possible to stop write operation that continues after 4 cycles. The 2 cycles of dummy data that follow these write data give the external agent time to stop the next write operation.
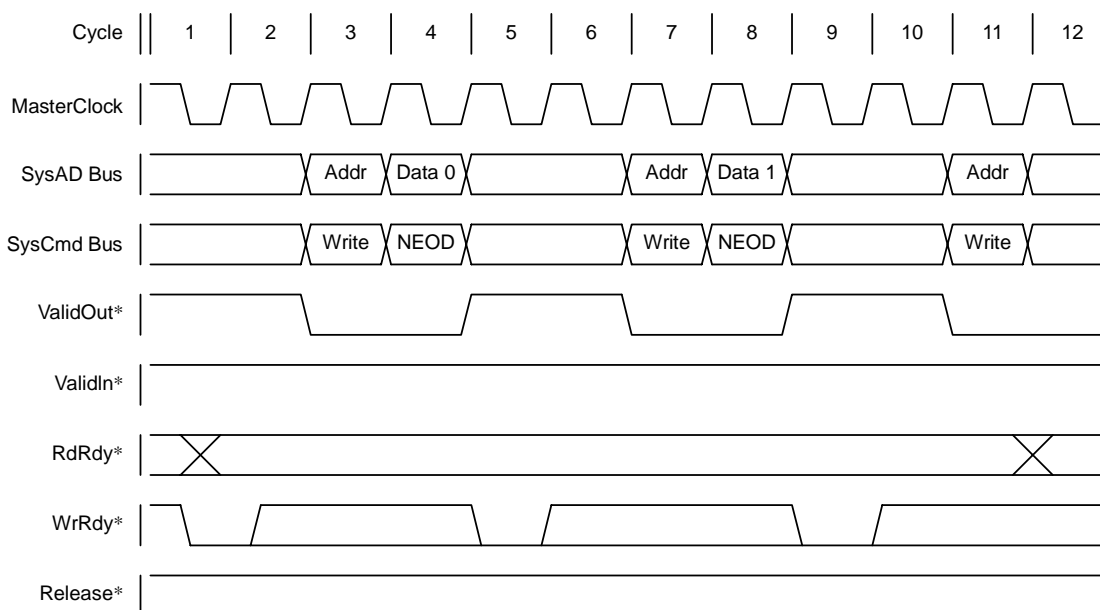
Figure 11.2.14 R4000 Compatible Write

2. Reissue write

When in the reissue write mode, the WrRdy* signal is asserted for 1 cycle 2 cycles before the address cycle, and the write operation is reissued when the WrRdy* signal is asserted during the address cycle. Figure 11.2.15 illustrates the reissue write protocol.

- By asserting (Low) the WrRdy* signal in the first and third cycles, Addr0/Data0 issues a write operation in the third or fourth cycle.

- By deasserting (High) the WrRdy* signal in the fifth cycle, Addr1/Data1 does not issue a write operation in the fifth and sixth cycles.

- By asserting (Low) the WrRdy* signal again in the eighth and tenth cycles, Addr1/Data issues a write operation in the tenth and eleventh cycles.
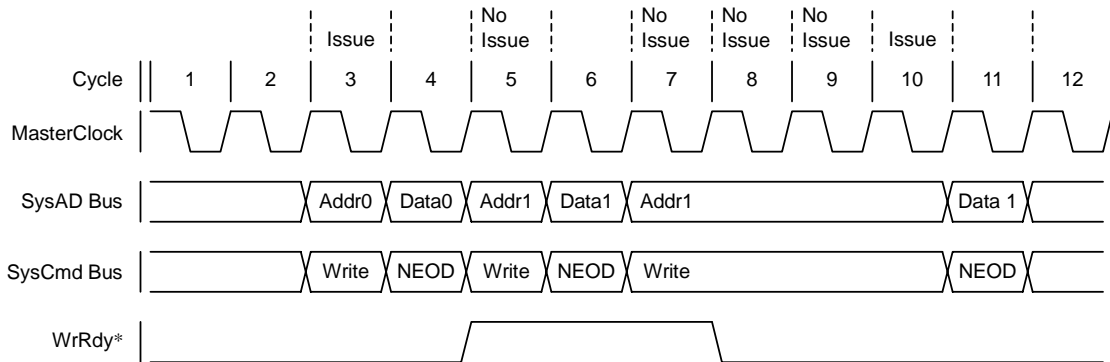
Figure 11.2.15 Reissue Write Protocol

3. Pipeline write

Similar to when in the R4000 compatible write mode, the pipeline write protocol issues a write operation if the WrRdy* signal is asserted for 1 cycle 2 cycles before the write operation is issued. However, the 2 cycles of dummy data after the write operation are deleted. The external agent must be able to accept one write operation or more after WrRdy* is deasserted. Figure 11.2.16 illustrates this protocol.

- Third, fourth cycle Addr0/Data0 is issued by asserting (Low) the WrRdy* signal in the first cycle.

- Fifth, sixth cycle Addr1/Data1 is issued by asserting (Low) the WrRdy* signal in the third cycle.

- Addr2 is not issued in the seventh cycle when the WrRdy* signal is deasserted (High) in the fifth cycle. Addr2/Data2 is issued in the tenth, eleventh cycle by asserting the WrRdy* signal again in the eighth cycle.
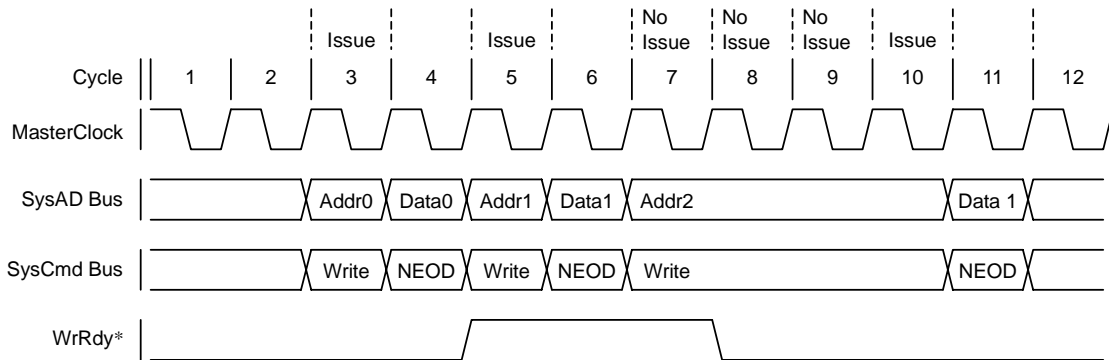
Figure 11.2.16 Pipeline Write Protocol

11.2.8.5 Processor block write request

The master state system interface is used to issue processor block write requests. Figure 11.2.17 illustrates a processor non-coherent block request made for 8-word data with the "D" data pattern.

1. Processor sends a write command to the SysCmd bus, then sends a write address to the SysAD bus.

2. Processor asserts the ValidOut* signal.

3. Processor sends data identifier to the SysCmd bus and sends data to the SysAD bus.

4. Processor asserts the ValidOut* signal only for the number of cycles required to transfer the data block.

5. Final data cycle directive must be included in data identifiers for the final data cycle.

Note: There is transmission protocol of Processor block write request three kinds same as Processor single write requests. With later G2SConfig-Regster, these modes are selected.
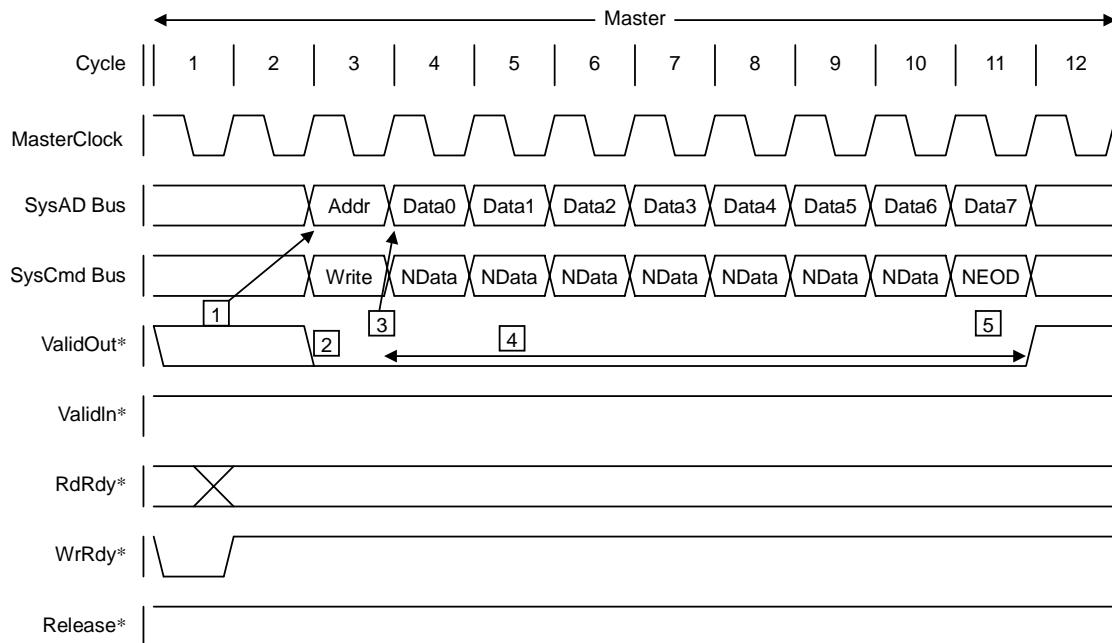
Figure 11.2.17 Processor non-coherent block request protocol

11.2.8.6 External request protocol

External requests can only be issued when the system interface is in the slave state. The external agent asserts the ExtRqst* signal, and requests use of the system interface. The processor asserts the Release* signal, releases the system interface, waits for it to enter the slave state, then the external agent issues an external request. If the system interface is already in the slave mode, namely, if the processor has put the system interface in the slave state on its own, then the external agent can immediately issue an external request.

In the case of the external agent, the system interface must be returned to the master state after issuing an external request. When the external agent issues a single external request, ExtRqst* must be deasserted 2 cycles after the cycle at which Release* is asserted. Also, when issuing a series of external requests, the ExtRqst* signal must be asserted before the last request cycle.

The processor continues processing external requests while ExtRqst* is asserted. However, until the processor completes a request that is currently being processed, it will not be able to release the system interface and put it into the slave state in preparation for the next external request. Also, until ExtRqst* is asserted, a series of external requests cannot be interrupted by a processor request.

11.2.8.7 External arbitration protocol

As previously mentioned, the ExtRqst* signal and Release* signal are used in system interface arbitration. Figure 11.2.18 illustrates the timing of the arbitration protocol when the slave state changes to the master state.

The arbitration cycle sequence is as follows.

1. The external agent asserts ExtRqst* when it becomes necessary to issue external requests.

2. The processor asserts Release* for 1 cycle when it becomes possible to process an external request.

3. The processor sets the SysAD bus and SysCmd bus to tri-state.

4. The external agent must start transmission to the SysAD bus and SysCmd bus 2 cycles after Release* is asserted.

5. The external agent deasserts ExtRqst* 2 cycles after Release* is asserted. This does not apply however to situations where an attempt is made to issue another external request.

6. The external agent sets the SysAD bus and SysCmd bus to tri-state when processing of the external request is complete.

The processor becomes able to issue processor requests 1 cycle after the external agent sets the busses to tri-state.
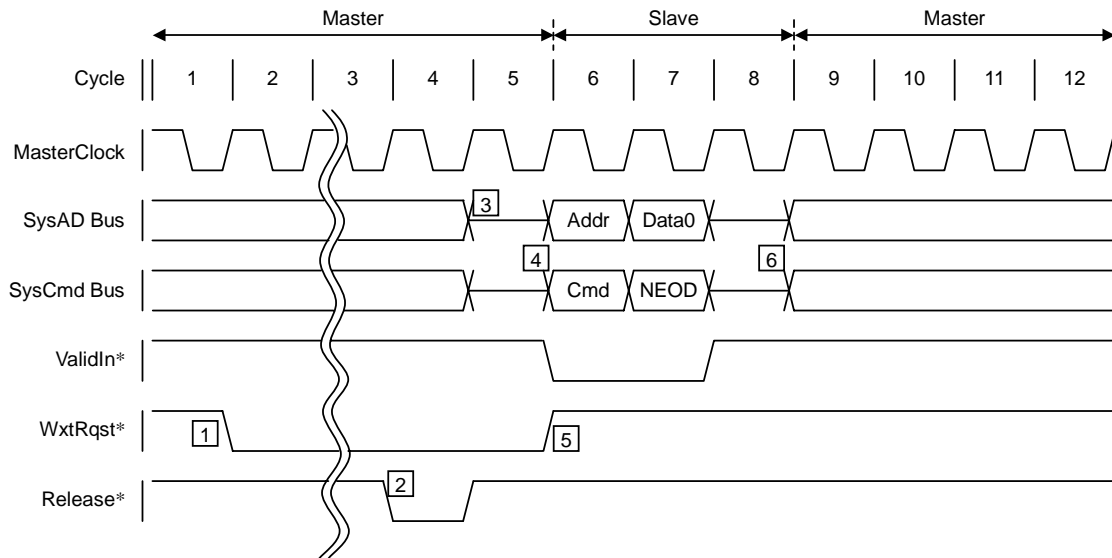
Figure 11.2.18 Arbitration Protocol Relating to External Requests

11.2.8.8 External read request protocol

External read requests are requests that read 1 word of data from processor-internal resources such as registers. External read requests cannot be partitioned. Namely, it is not possible to generate other requests between an external read request and the corresponding read response.

Figure 11.2.19 illustrates the timing of external read requests, which consist of the following steps.

1. The external agent requests use of the system interface by asserting ExtRqst*.

2. The processor asserts Release* for 1 cycle, then releases the system interface by deasserting Release* and puts the interface into the slave state.

3. After Release* is deasserted, the SysAD bus and SysCmd bus are set to tri-state for 1 cycle.

4. The external agent sends a read request command to the SysCmd bus, sends a read request address to the SysAD bus, and asserts ValidIn* for 1 cycle.

5. After sending the above address and command, the external agent sets the SysCmd and SysAD busses to tri-state, makes it possible for the processor to drive them, then releases them both. The processor that accessed the data to be read returns the data to the external agent. Therefore, the processor sends the data identifier to the SysCmd bus, sends the response data to the SysAD bus, then asserts ValidOut* for 1 cycle. This data identifier indicates that data are the response data of the final data cycle.

6. The system interface is in the master state. The processor continues to drive the SysCmd bus and SysAD bus even after the read response is returned.

External read requests can read data from only a single word in the processor. If data elements other than a word are requested, the processor response is not defined.
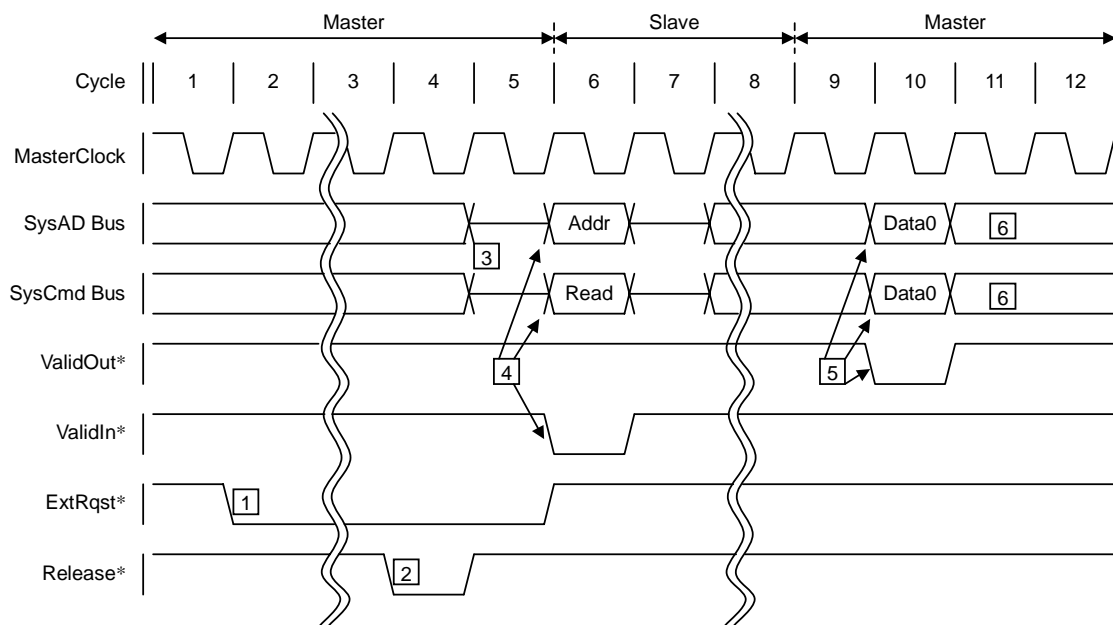
Figure 11.2.19 External Read Request when System Interface is in the Master State

Note: The processor contains no resources that can read by way of external read requests. The processor returns data identifiers with SysCmd[5] of the error data bits set along with undefined data when it receives an external read request.

11.2.8.9 External null request protocol

The processor only supports one external null request. The system interface release external null request returns the system interface from the slave state to the master state. This request does not affect any other processors.

The only processing the external null request does is to have the processor return the system interface to the master state.

Figure 11.2.20 illustrates the timing of the external null request, which consists of the following steps.

1. The external agent drives the system interface, sends the external null request command to the SysCmd bus, then asserts the ValidIn* signal for 1 cycle.

2. The SysAD bus is not available during the external null request address cycle (there are no valid data in the bus).

3. The null request ends when the address cycle is issued.

In the case of a system interface release null request, the external agent releases the SysCmd bus and SysAD bus so the system interface can return to the master state.
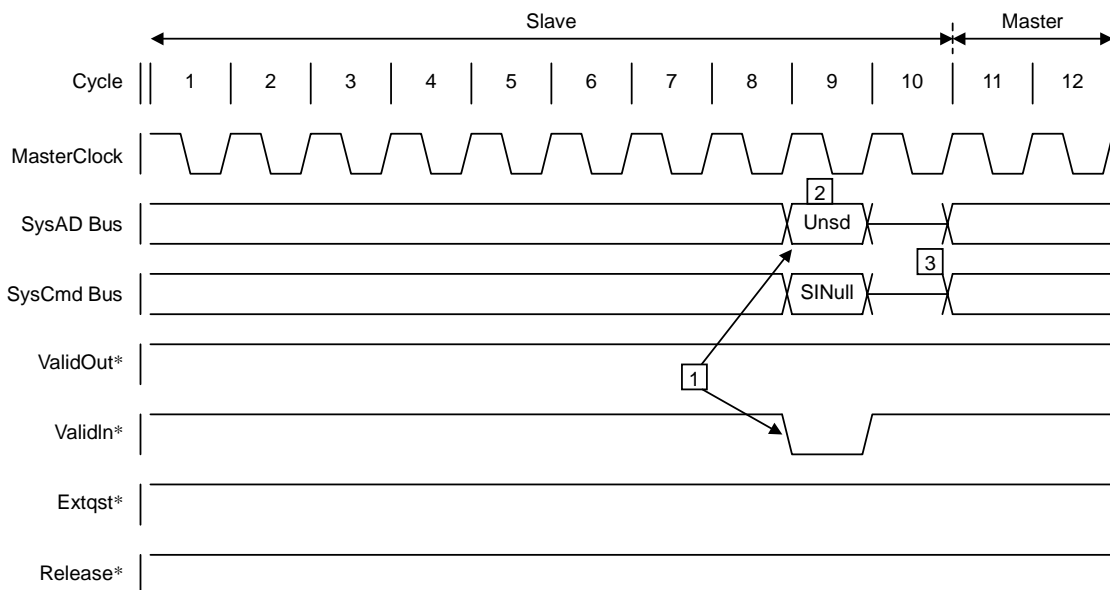
Figure 11.2.20 System Interface Release External Null Request

11.2.8.10 External write request protocol

The same protocol as the processor single word write protocol is used for external write requests, except when the ValidIn* signal is asserted instead of the ValidOut* signal.

Figure 11.2.21 illustrates the timing of the external write request, which consists of the following steps.

1. The external agent requests use of the system interface by asserting ExtRqst*.

2. The processor asserts Release*, then the system interface is released from the processor and goes into the slave state.

3. The external agent sends a write command to the SysCmd bus, and sends a write address to the SysAD bus while asserting ValidIn*.

4. The external agent sends data identifiers to the SysCmd bus, and sends data to the SysAD bus while asserting ValidIn*.

5. Data identifiers for this data cycle must contain an indication of a coherent or non-coherent final data cycle.

6. After a data cycle is issued, the write request is complete, the external agent sets the SysCmd and SysAD busses to tri-state, then the system interface returns to the master state. Timing of the SysADC bus and SysCmdP bus are each the same as the SysAD bus and SysCmd bus, respectively.

External write requests can write only 1 word of data to the processor. Operation of processors that have specified data elements other than a single word of data by an external write request is not defined.
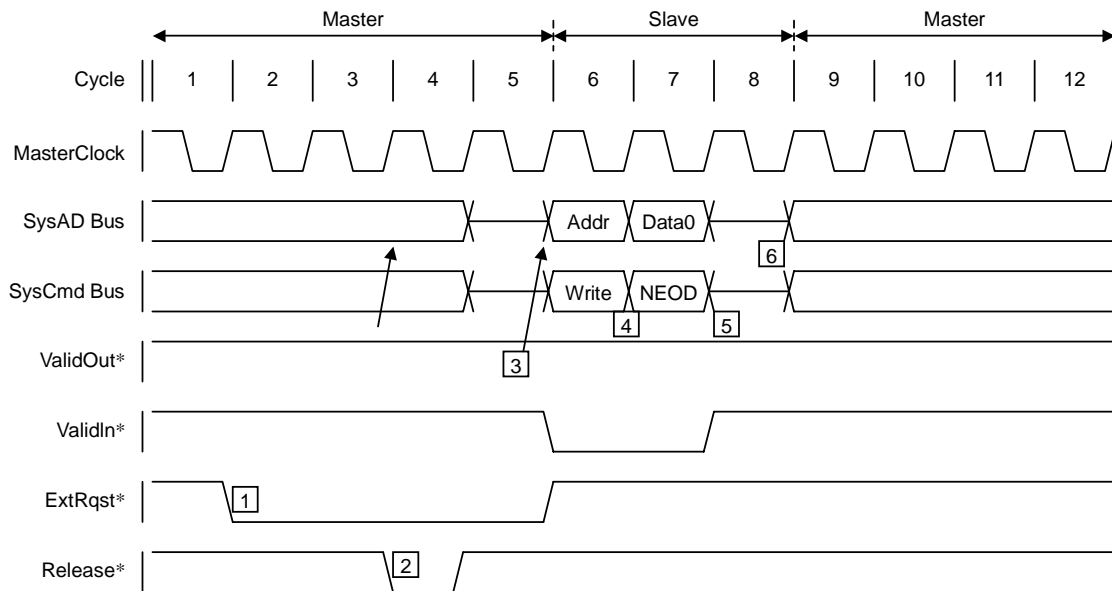
Figure 11.2.21 External Write Request when System Interface Starts in the Master State

11.2.8.11 Read response protocol

The external agent must use the read response protocol to return data to the processor if a processor read request has been received. The sequence of the read response protocol is as follows.

1. The external agent waits for the processor to automatically execute a shift into the slave state.

2. The external agent uses either a single data cycle or a series of data cycles to return data.

3. After issuing the final data cycle, the read response ends, then the external agent sets the SysCmd and SysAD busses to tri-state.

4. The system interface returns to the master state.

Note: After issuing a read response, the processor automatically shifts to the slave state.

5. Data identifiers of a data cycle must indicate that the data are response data.

6. Final data cycle identifiers must contain an indication that a cycle is the final data cycle.
In the case of read responses to non-coherent block read requests, it is not necessary for the response data to check the initial cache state. The cache state is automatically set to exclusively dirty.
Data identifiers of data cycles can send notification of transfer data errors in those cycles. The external agent must return data blocks with the correct size even when there is an error in the data. Whether there is a single error or multiple errors in the read response data cycles, the processor processes them as bus errors.

Read responses must always be returned to the processor when a processor read request is being held. Processor operation is not defined if a read response was returned in a state where there were no processor read requests on hold.

Figure 11.2.22 illustrates a processor word read request and the subsequent word read response. Also, Figure 11.2.23 illustrates the read response to a processor block read request when the system interface is in the slave state.
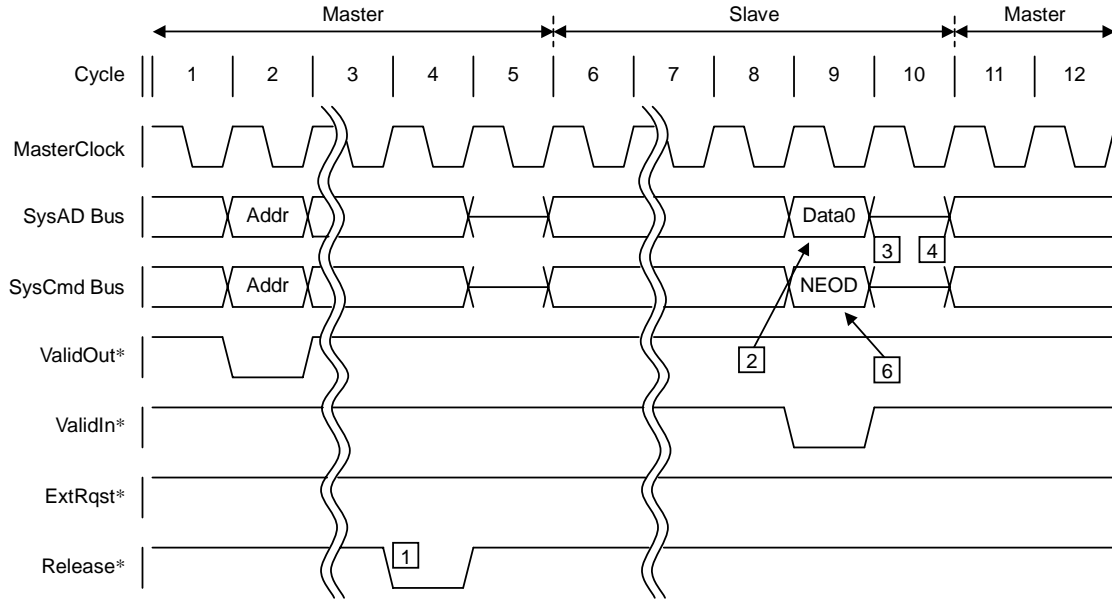
Figure 11.2.22 Processor Word Read Request and Subsequent Word Read Response
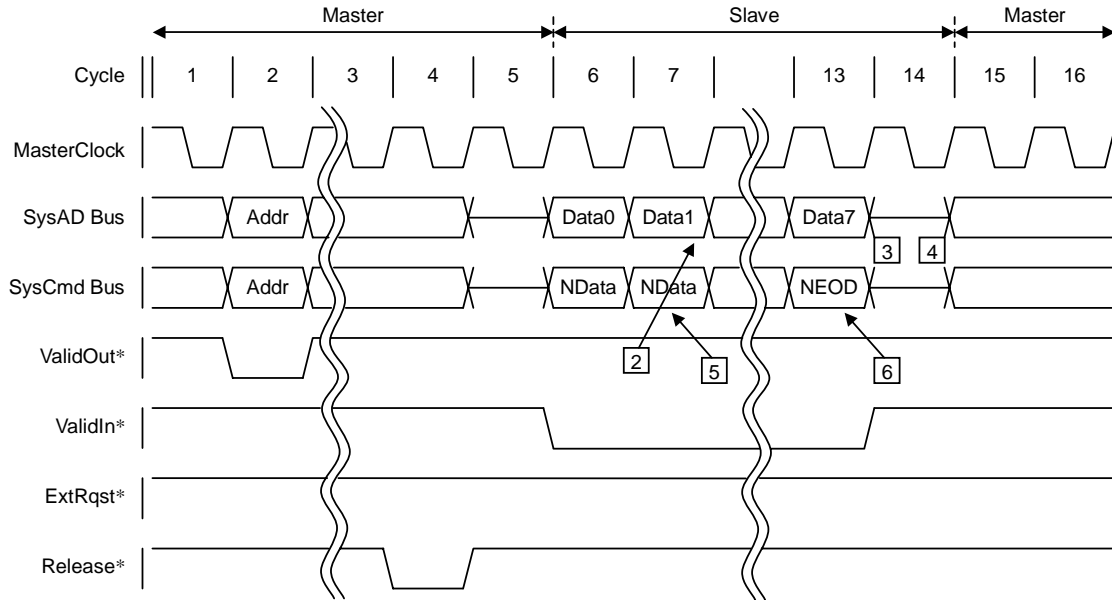
Figure 11.2.23 Block Read Response when System Interface is in the Slave State

11.2.9 Data Transfer

The maximum data transfer rate of the system interface is 1 double word per cycle.

The external agent can select the data transfer rate to the processor. For example, it is possible to transfer data and assert the ValidIn* signal just once for every *n* cycles instead of at all cycles. The external agent can transfer data to the processor at the selected transfer rate.

The processor interprets cycles as being valid when the ValidIn* signal is asserted and the SysCmd bus includes the data identifier. After a cycle has been interpreted as valid, the processor continues to accept data until a data word with an indication that it is the final data word appears.

11.2.9.1 Data transfer pattern

The term "data pattern" in the case of block write operations, refers to a string of text that indicates the "data" cycles repeated so that the appropriate data transfer speed can be obtained and the "unused" cycles. Wxx data patterns indicate a repetition data transfer rate where there is one word in three cycles followed by two unused cycles. The data transfer rate is set by G2Sconfig register(0xF FF10 0000).

Table 11.2.5 indicates the data transfer rate, data pattern, and setting.

Table 11.2.5 Data Transfer Rate, Data Pattern and Setting at Block write requests

Maximum Data Transfer Rate	Data Pattern	Setting Bit0
1 word/3 MasterClock cycle	Wxx	0
1 word/1 MasterClock cycle	W	1

Note: The setting Bit 0 is bit 0 of the G2Sconfig register (0xF FF10 0000).
      Bit 0 is set to "0" when initialized.

11.2.9.2 Independent transfer on SysAD bus

A majority of applications connect the processor and external agent interior (both directions), and register format transceivers together in a point-to-point manner via the SysAD bus. The only two SysAD bus drives available for such applications are the processor and the external agent.

Depending on the application, it may be necessary to make additional connections on the SysAD bus for drivers and receivers to transfer data using the SysAD bus without involving the processor. Such transfers are referred to as independent transfers. In order to perform independent transfers, the external agent must use arbitration handshake signals and external null requests to properly tune SysAD bus control.

Independent transfer is performed on the SysAD bus according to the following steps.

1. The external agent requests access to the SysAD bus in order to issue an external request.

2. The processor releases the system interface and puts it in the slave state.

3. The external agent can independently transfer data using the SysAD bus. However, the ValidIn* signal must be asserted during that transfer.

4. When transfer is complete, the external agent must issue a system interface release null request and return the system interface to the master state.

11.2.10 System Interface cycle time

In the case of a processor, there is a response time for each kind of processor transaction and for each external request. A minimum and maximum cycle count has been prescribed for each response time. Since processor requests themselves are restrained by system interface request protocols, checking the protocols makes it possible to determine the cycle count required for requests. The interval for the next interface operation is variable within the range of the minimum and maximum cycle counts.

- Stand-by time from when an external request is received and the processor releases the system interface, until when the interface enters the slave state (release latency).
- Response time to external request that requires a response (external response latency).

11.2.10.1 Release latency

Broadly defined, release latency is the number of cycles for which it is possible to wait from when the processor receives an external request until when the system interface is released and shifts to the slave state. If there are no processor requests currently in progress, the processor must delay release of the system interface for a few cycles since it is internal operation. Therefore, if release latency is strictly defined, it becomes the cycle count from when the ExtRqst* signal is asserted until when the Release* signal is asserted.

There are three types of release latency.

- Category 1: If external request signal is asserted 2 cycles before the final cycle of the processor request
- Category 2: If external request signal is asserted during processor request or is the final cycle even if it is asserted
- Category 3: If processor automatically shifts to the slave state

Table 11.2.6 indicates the minimum and maximum release latency inherent to categories 1, 2, and 3. However, note that these cycle counts may be changed at any time.

Table 11.2.6 Release Latency for External Requests

Category	Minimum cycle count	Maximum cycle count
1	3	5
2	1	24
3	0	0

11.2.11 System Interface Command and Data Identifiers

System interface commands specify the type of system interface and its properties. This specification is performed in the request address cycle. The system interface data identifiers specify the properties of the data transferred during the system interface data cycle.

Of the system interface commands and data identifiers used for external requests, set the bits and fields that are reserved to "1." In the case of system interface commands and data identifiers used for processor requests, bits, field contents and data identifiers that are reserved in the commands are undefined.

11.2.11.1 Syntax of commands and data identifiers

System interface commands and data identifiers consist of 9 bits. These commands are sent by the SysCmd bus from the processor to the external agent or from the external agent to the processor during either the address cycle or the data cycle. Bit 8 (MSB) of the SysCmd bus specifies whether the SysCmd contents at that time are a command or a data identifier (namely, whether the current cycle is an address cycle or a data cycle). If the contents are a system interface command, then SysCmd[8] must be set to "0." If the contents are a system interface data identifier, then SysCmd[8] must be set to "1."

11.2.11.2 Syntax of system interface commands

Following is an explanation of the SysCmd bus structure in the case of a system interface command. Figure 11.2.24 illustrates the structure common to all system interface commands.

8	7 5	4 0
0	Request type	Request specific
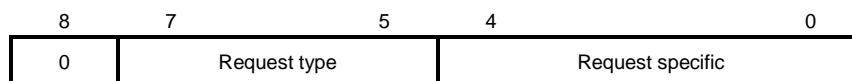
Figure 11.2.24 Syntax Bit Structure of System Interface Command

SysCmd[8] must always be set to "0" in the case of system interface commands.

SysCmd[7:5] specifies the type of system interface request (read, write, null). Table 11.2.7 indicates the SysCmd[7:5] specification method.

Table 11.2.7 SysCmd[7:5] Specification Method for System Interface Commands

SysCmd[7:5]	Command
0	Read request
1	Reserve
2	Write request
3	Null request
4-7	Reserve

SysCmd[4:0] varies depending on the type of request. Each specification type is indicated below.

11.2.11.3 Read requests

Figure 11.2.25 illustrates the SysCmd format in the case of read requests.

8	7		5	4		0
0		000			Read request specific	
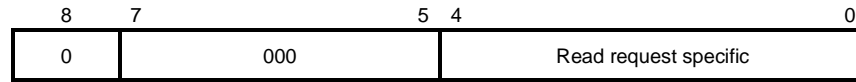
Figure 11.2.25 Bit Definition of SysCmd Bus for Read Requests

Table 11.2.8, Table 11.2.9, and Table 11.2.10 indicate the methods for specifying SysCmd[4:0] for read requests.

Table 11.2.8 Specification Method of SysCmd[4:3] for Read Requests

SysCmd[4:3]	Read Properties
0-1	Reserved
2	Non-coherent block read
3	Word or partial word read

Table 11.2.9 Specification Method of SysCmd[2:0] for Block Read Requests

SysCmd[2]	Reserved
SysCmd[1:0]	Read Block Size
0	Reserved
1	8 words
2-3	Reserved

Table 11.2.10 Data Size Expressed by SysCmd[2:0] for Word or Partial Word Read Requests

SysCmd[2:0]	Read Data Size
0	1 byte valid (byte)
1	2 bytes valid (half-word)
2	3 bytes valid (tri-byte)
3	4 bytes valid (word)
4	Reserved
5	Reserved
6	Reserved
7	Reserved

Note: 6.10.4 of SysCmd[2:0] is only valid when in the 64-bit bus mode.

11.2.11.4 Write requests

Figure 11.2.26 illustrates the SysCmd format for write requests.

Table 11.2.11 indicates the methods of specifying write properties using SysCmd[4:3]. Table 11.2.12 indicates the methods of specifying replacements properties using SysCmd[2:0] for block write requests. Table 11.2.13 indicates the methods of specifying the data size using SysCmd[2:0] for write requests.

8	7	5	4	0
0		010	Write request specific	
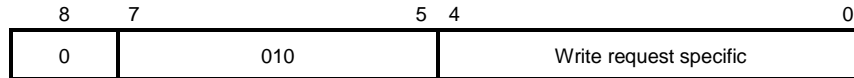
Figure 11.2.26 SysCmd Bus Bit Specification for Write Requests

Table 11.2.11 Methods of Specifying SysCmd[4:3] for Write Requests

SysCmd[4:3]	Write Properties
0-1	Reserved
2	Block write
3	Word or partial word write

Table 11.2.12 Specification Method of SysCmd[2:0] for Block Write Requests

SysCmd[2]	Reserved
SysCmd[1:0]	Write Block Size
0	Reserved
1	8 words
2-3	Reserved

Table 11.2.13 Methods for Specifying SysCmd[2:0] for Word or Partial Word Write Requests

SysCmd[2:0]	Write Data Size
0	1 byte valid (byte)
1	2 bytes valid (half-word)
2	3 bytes valid (tri-byte)
3	4 bytes valid (word)
4	Reserved
5	Reserved
6	Reserved
7	Reserved

11.2.11.5 Null requests

Figure 11.2.27 illustrates the SysCmd format in the case of read requests.

8	7	5	4	0
0	011		Null request specific	
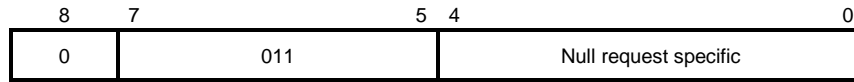
Figure 11.2.27 Bit Definition of the SysCmd Bus for Null Requests

Null request commands are always used for system interface release external null requests. Table 11.2.14 indicates methods of specifying SysCmd[4:3] for system interface release external null requests. SysCmd[2:0] is reserved for null requests.

Table 11.2.14 Method of Specifying SysCmd[4:3] for External Null Requests

SysCmd[4:3]	Null Properties
0	Release system interface
1-3	Reserved

11.2.11.6 Syntax of system interface data identifiers

This section defines methods of specifying the SysCmd bus for system interface identifiers. The bit structure illustrated in Figure 11.2.28 is common to all system interface data identifiers.

8	7		5	4	3	0
0	Final Data	Response Data	Error Data	Data Check	Reserved	
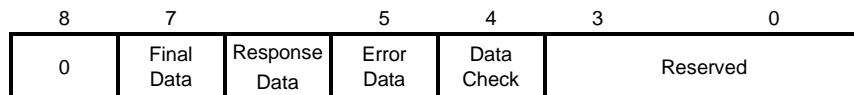
Figure 11.2.28 Bit Definition of the SysCmd Bus for Null Requests

SysCmd[8] must always be set to "1" for system interface data identifiers. System interface data identifiers are in the non-coherent data format.

11.2.11.7 Non-coherent data

Non-coherent data are data such as the following.

- Data that are the subject of a processor block write request or a processor double word/partial double word/word/partial word write request.
- Data that are returned to a processor non-coherent block read request or a processor double word/partial double word/word/partial word read request.
- Data that are the subject of an external write request.
- Data that are returned to an external read request as a response.

11.2.11.8 Bit definition of data identifiers

In the case of processor or external coherent data identifiers and processor or external non-coherent data identifiers, SysCmd[7] indicates that they are the final data element, and SysCmd[6] indicates whether they are response data. Response data are the data that are returned to a read request as a response.

SysCmd[5] indicates whether there is an error in a data element. Uncorrectable errors are included in the error data. When such an error is returned to the processor, a bus error is generated. If a primary parity error is detected in the data items to be transferred, the processor deasserts the good data bits and sends data.

SysCmd[4] indicates to the processor whether the data bits and check bits of a data element should be searched.

SysCmd[3] is reserved in the case of external data identifiers.

SysCmd[4:3] is reserved in the case of non-coherent processor data identifiers.

SysCmd[2:0] is reserved in the case of non-coherent data identifiers.

Table 11.2.15 indicates methods of specifying SysCmd[7:3] for processor data identifiers. Table 11.2.16 indicates methods of specifying SysCmd[7:3] for external data identifiers.

Table 11.2.15 Methods of Specifying SysCmd[7:3] for Processor Data Identifiers

SysCmd[7]	Final Data Element Indication
0	Final data element
1	Is not final data element
SysCmd[6]	Response Data Indication
0	Response data
1	Is not response data
SysCmd[5]	Good Data Indication
0	No errors
1	Is error data
SysCmd[4:3]	Reserved

Table 11.2.16 Method of Specifying SysCmd[7:3] for External Data Identifiers

SysCmd[7]	Final Data Element Indication
0	Final data element
1	Is not final data element
SysCmd[6]	Response Data Indication
0	Response data
1	Is not response data
SysCmd[5]	Good Data Indication
0	No errors
1	Is error data
SysCmd[4]	Enable Data Check
0	Check data bits and check bits
1	Do not check data bits and check bits
SysCmd[3]	Reserved

11.2.12 System Interface Addresses

System interface addresses are complete 32-bit physical addresses. They are sent to the lower 32 bits of the SysAD bus during the address cycle.

11.2.12.1 Addressing rules in the 32-bit bus mode

Addresses that are to be used in word or partial word transactions are arranged to match the size of the data element. The following rules are used by this system.

- Target addresses of the block request are aligned to the word boundaries. In other words, the lower 2 bits of the address become "0".
- Word requests set the lower 2 bits of the address to "0".
- Half-word requests set the least significant bit of the address to "0".
- Byte requests and 3-byte requests use the byte address.

11.2.13 Mode Register of System Interface (G2Sconfig)

The Mode Register of System Interface (G2Sconfig) is a read/write register. This register is only Word-Access.

Table 11.2.17 G2SConfig

Address	Field	Description
0xF_FF10_0000	G2Sconfig	Mode Register of System Interface

31	20	19	4	3	2	1	0	Bit
Reserved		TOut Val		TOut Enable		Write mode		Data rate

Bit	Field	Description	ColdReset	Read/Write
31:20	—	Read as undefined. Must be written as 0xfff.	Undefined	Read
19:4	—	Read as undefined. Must be written as 0xffff.	0xFFFF	Read/Write
3	—	Read as undefined. Must be written as 0.	0	Read/Write
2:1	Write mode	Processor write protocols set 00: R4000 compatible write 01: Reserved 10: Pipeline write 11: Reissue write	00	Read/Write
0	Data rate	At the Block write data rate set 0: WWWWWWWW 1: WxxWxxWxxWxx WxxWxxWxxWxx	0	Read/Write

Note1: When initialized   Single write: AWxx (R4000 compatible)
                           Block write: AWWWWWWWW
Note2: In case of MODE43∗ = H, Read Time Out Counter does not work.
        It can generate Read Time Out Error when wrote in 0x0 at TOut Val. It does not let a value of
        TOut Enable affect it. Please write in a value except 0x0 at TOut Val.

Figure 11.2.29 G2Sconfig Register Formats

11.2.14 Data Error Detection

The TX4951B internal system interface uses the following two methods to detect data errors:

- Indication of good data by data identifier SysCmd[5]
- Determination by a check bit

11.2.14.1 Indication of good data by data identifier SysCmd[5]

This bit indicates whether data is good or not for all data. Therefore, in a block refill transfer, for example, no matter in what number of data SysCmd[5] may have been set, a bus error exception is always generated.

11.2.14.2 Determination by a check bit

When an error is detected by check bit determination, a bus error exception occurs. Because this determination also is made for all data, no matter in what number of data in a block refill transfer an error may have occurred, a bus error exception is always generated.

11.2.14.3 Timing at which a bus error exception occurs

Indication of good data (SysCmd[5]): Two cycles after read data

Check bit error detection: Three cycles after read response data

11.2.14.4 Precautions

- There is no means of identifying whether a bus error exception has been generated by indication of good data or by check bit determination. Nor does the TX49 core have a cache parity bit, but when a bus error exception occurs during a block refill transfer, the cache line is referenced INVALID.
- Regardless of whether a bus error exception has been generated by indication of good data or by check bit determination during a block refill transfer, a designated block size of data needs to be transferred.

11.3 System Interface of R4300 type protocol mode

In TX4951B, it is built in system interface function corresponding to R4300 type protocol. A selection of above-mentioned R5000 type protocol mode or R4300 type protocol mode increases by external pin (MODE43* : 94 pin).

MODE43* = 0: R4300 type protocol

MODE43* = 1: R5000 type protocol

Note: In R4300 type protocol mode of TX4951B, there is not PReq signal. Therefore there is the need that PReq terminal of an external agent is always fixed in "L" Level. And there is not a function too to show that protocol errors were detected.
But search of protocol errors is possible when I have a counter built-in in TX4951B inside, and there is not lead reply between constant time because it is established the function that an exception generates bus error.

11.3.1 System Interface Description of R4300 Type Protocol Mode

The TX4951B processor has a 32-bit address/data interface. The System interface consists of:

- 32-bit address and data bus, **SysAD**
- 5-bit command bus, **SysCmd**
- five handshake signals:
    - **EValid***, **PValid***
    - **EReq***
    - **PMaster***, **EOK***

Table 11.3.1 System Interface Signals

(R4300 type protocol mode is MODE43* = 0)

Signal	I/O	Function
SysAD[31:0]	I/O	Address and data transfer bus, multiplexed between the processor and an external agent.
SysCmd[4:0]	I/O	Used for command and data identifier transmission between the processor and an external agent.
SysCmd[8:5] / GND	O	When R4300 type protocol mode, those pin output "L".
ValidIn* / EValid*	I	During the cycle it is asserted, EValid* indicates an external agent is driving a valid address or valid data on the SysAD bus, and a valid command or data identifier on the SysCmd bus.
ValidOut* / PValid*	O	During the cycle it is asserted, PValid*, indicates the processor is driving a Valid address or Valid data on the SysAD bus, and a valid command or data indentifier on the SysCmd bus.
ExtRqst* / Ereq*	I	Indicates an external agent is requesting System interface bus ownership.
Release* / PMaster	O	Indicates an external agent is capable of the system interface bus.
WrRdy* / EOK*	I	Indicates an external agent is capable of accepting a processor request.
RdRdy* / GND	O	When R4300 type protocol mode this pin output "L".

Table 11.3.2 Pin Assign

(MODE43∗ = 0)

1	VccIO	26	Vss	51	Vss	76	Vss
2	MasterClock	27	GND	52	VccInt	77	VccInt
3	VccInt	28	EOK∗	53	SysCmd0	78	SysAD24
4	VccPLL	29	EValid∗	54	SysCmd1	79	SysAD25
5	VssPLL	30	PValid∗	55	SysCmd2	80	SysAD26
6	Vss	31	PMaster∗	56	SysCmd3	81	SysAD27
7	PLLReset∗	32	VccInt	57	Vss	82	Vss
8	VccIO	33	Vss	58	VccIO	83	VccIO
9	SysAD4	34	ExtRqst∗	59	SysCmd4	84	SysAD28
10	SysAD5	35	TRST∗	60	GND	85	SysAD29
11	SysAD6	36	TintDis	61	GND	86	SysAD30
12	SysAD7	37	VccIO	62	VccInt	87	SysAD31
13	Vss	38	HALTDOZE	63	Vss	88	Vss
14	VccIO	39	NMI∗	64	GND	89	VccIO
15	SysAD8	40	Int0∗	65	GND	90	SysAD0
16	SysAD9	41	Int1∗	66	SysAD16	91	SysAD1
17	SysAD10	42	Int2∗	67	SysAD17	92	SysAD2
18	SysAD11	43	Int3∗	68	SysAD18	93	SysAD3
19	VccInt	44	Vss	69	SysAD19	94	MODE43∗
20	Vss	45	VccInt	70	Vss	95	Vss
21	SysAD12	46	DivMode0	71	VccIO	96	VccInt
22	SysAD13	47	DivMode1	72	SysAD20	97	JTDO
23	SysAD14	48	Endian	73	SysAD21	98	JTDI
24	SysAD15	49	ColdReset∗	74	SysAD22	99	JTCK
25	VccIO	50	VccIO	75	SysAD23	100	JTMS

Note 1: Active-low signals have a trailing asterisk (∗).

Note 2: The TX4951B is put in R4300 mode when MODE43∗ = 0 and in R5000 mode when MODE43∗ = 1.

Note 3: In R4300 mode, GND is driven Low and Preq is not available.

11.3.2 System Events

System events include:

- Fetch miss in the instruction cache
- Load miss in the data cache
- Store miss in the data cache
- an uncached load or store
- actions resulting from the execution of cache instructions

When a system event occurs, the processor issues a request or a series of requests through the system interface to access some external resource to service that event. The system interface must be connected to an external agent that coordinates access to system resources.

Processor requests include both read and write requests:

- a read request supplies an address to an external agent
- a write request supplies an address and a word or block of data to be written to an external agent

Processor read requests that have been issued, but for which data has not yet been returned, are said to be *pending*. The processor will not issue another request while a read is already pending. A processor read request is said to be *complete* after the last transfer of response data has been received from an external agent. A processor write request is said to be complete after the last word of data has been transmitted.

External requests include both read responses and write requests:

- a read response supplies a block or single transfer of data from an external agent in response to a read request
- a write request supplies an address and a word of data to be written to a processor resource

When an external agent receives a read request, it accesses the specified resource and returns the requested data through a read response, which may be returned any time after the read request and at any data rate.

By default, the processor is the master of the system interface. An external agent becomes master of the system interface either through arbitration, or by default after a processor read request. The external agent returns mastership to the processor after the external request completes and/or after the processor read request has been serviced.

11.3.3 System Event Sequences and the SysAD Bus Protocol

The following sections detail the sequence and timing of processor and external requests.

Note: The following sections describe the SysAD bus protocol; the TX4951B processor always meets the conditions of this protocol. The TX4951B processor is capable of receiving sequences of transactions on the bus at full protocol speed and of receiving data on every cycle. At a minimum, the design of external agents must meet the requirements of this protocol, and would ideally take full advantage of the maximum speed of the TX4951B processor.

11.3.3.1 Fetch Miss

When the processor misses in the instruction cache on a fetch, it obtains a cache line of instructions from an external agent. The processor issues a read request for the cache line and waits for an external agent to provide the data in response to this read request.

11.3.3.2 Load Miss

When the processor misses in the data cache on a load, it obtains a cache line of data from an external agent. The processor issues a read request for the cache line and waits for an external agent to provide the data in response to this read request. If the cache data which the incoming line will replace contains valid dirty data, this data is written to memory. The read completes before the write of the dirty cast-out data.

11.3.3.3 Store Miss

When the processor misses in the data cache on a store, it issues a read request to bring a cache line of data into the cache, where it is then updated with the store data. If the cache data which the incoming line will replace contains valid dirty data, the data is written to memory. The read completes before the write of the dirty cast-out data.

To guarantee that cached data written by a store is consistent with main memory, the corresponding cache line must be explicitly flushed from the cache using a cache operation.

11.3.3.4 Uncached Load or Store

When the processor performs an uncached load, it issues a read request and waits for a single transfer of read response data from an external agent.

When the processor performs an uncached store, it issues a write request and provides a single transfer of data to the external agent.

The processor does not consolidate data on uncached writes. For example, writes of two contiguous halfwords takes two write cycles, they are never grouped into a single word write.

11.3.3.5 Cache Instructions

The TX4951B processor provides a number of cache instructions for use in maintaining the state and contents of the caches. Cache operations supported in the TX4951B processor are described in Chapter 5.

11.3.3.6 Byte Ordering (Endian)

The System interface byte order is set by the **Endian** of external pin. The byte order is big-endian when **Endian** is high, and little-endian when **Endian** is low. The *RE* (reverse-endian) bit in the *Status* register can be set by software to reverse the byte order available in User mode.

11.3.3.7 Physical Addresses

Physical addresses are driven on **SysAD[31:0]** during address cycles.

11.3.3.8 Interface Buses

Figure 11.3.1 shows the primary communication paths for the System interface: a 32-bit address and data bus, **SysAD[31:0]**, and a 5-bit command bus, **SysCmd[4:0]**. These **SysAD** and the **SysCmd** buses are bidirectional; that is, they are driven by the processor to issue a processor request, and by the external agent to issue an external request.

A request through the System interface consists of:

- an address
- a System interface command that specifies the precise nature of the request
- a series of data elements if the request is for a write or read response.
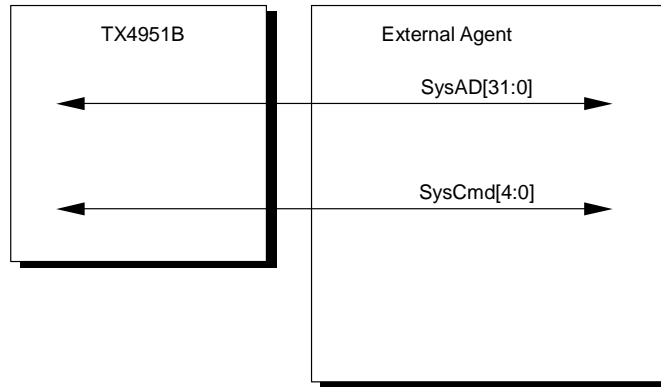
Figure 11.3.1 System Interface Buses

11.3.3.9 Address and Data Cycles

The **SysCmd** bus identifies the contents of the **SysAD** bus during any cycle in which it is valid. Cycles in which the **SysAD** bus contains a valid address are called *address cycles*. Cycles in which the **SysAD** bus contains valid data are called *data cycles*. The most significant bit of the **SysCmd** bus is always used to indicate whether the current cycle is an address cycle or a data cycle.

When the TX4951B processor is driving the **SysAD** and **SysCmd** buses, the System interface is in *master state*. When the external agent is driving the **SysAD** and **SysCmd** buses, the System interface is in *slave state*.

- When the processor is master, it asserts the **PValid*** signal when the **SysAD** and **SysCmd** buses are valid.
- When the processor is slave, an external agent asserts the **EValid*** signal when the **SysAD** and **SysCmd** buses are valid.

The **SysCmd** bus identifies the contents of the **SysAD** bus during valid cycles.

- During address cycles [**SysCmd[4]** = 0], the remainder of the **SysCmd** bus, **SysCmd[3:0]**, contains a *System interface command*, described later in this chapter.
- During data cycles [**SysCmd[4]** = 1], the remainder of the **SysCmd** bus, **SysCmd[3:0]**, contains a *data identifier*, described later in this chapter.

11.3.4   System Interface Protocols

Figure 11.3.2 shows the register-to-register operation of the System interface. That is, processor outputs come directly from output registers and begin to change with the rising edge of MasterClock.

Processor inputs are fed directly to input registers that latch these input signals with the rising edge of MasterClock. This allows the System interface to run at the highest possible clock frequency.
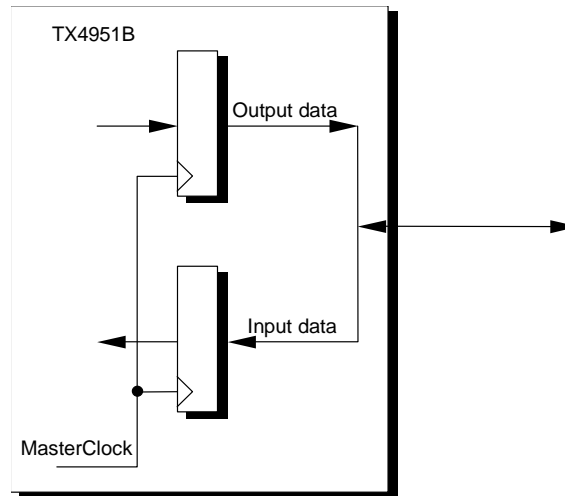

Figure 11.3.2 System Interface Register-to-Register Operation

11.3.4.1 Master and Slave States

When the TX4951B processor is driving the **SysAD** and **SysCmd** buses, the System interface is in *master state*. When the external agent is driving the **SysAD** and **SysCmd** buses, the System interface is in *slave state*.

In master state, the processor asserts the signal **PValid*** whenever the **SysAD** and **SysCmd** buses are valid.

In slave state, the external agent asserts the signal **EValid*** whenever the **SysAD** and **SysCmd** buses are valid.

11.3.4.2 Moving from Master to Slave State

The processor is the default master of the system interface. An external agent becomes master of the system interface through arbitration, or by default after a processor read request. The external agent returns mastership to the processor after an external request completes.

The System interface remains in master state unless one of the following occurs:

- The external agent requests and is granted the System interface (external arbitration).
- The processor issues a read request (uncompelled change to slave state).

The following sections describe these two actions.

11.3.4.3 External Arbitration

The System interface must be in slave state for the external agent to issue an external request through the System interface. The transition from master state to slave state is arbitrated by the processor using the System interface handshake signals **EReq*** and **PMaster***. This transition is described by the following procedure:

1. An external agent signals that it wishes to issue an external request by asserting **EReq***.

2. When the processor is ready to accept an external request, it releases the System interface from master to slave state by negating **PMaster***.

3. The System interface returns to master state as soon as the issue of the external request is complete.

11.3.4.4 Uncompelled Change to Slave State

An *uncompelled* change to slave state is the transition of the System interface from master state to slave state, initiated by the processor itself when a processor read request is pending. **PMaster*** is negated automatically after a read request. An uncompelled change to slave state occurs either during or some number of cycles after the issue cycle of a read request.

The uncompelled release latency depends on the state of the cache. After an uncompelled change to slave state, the processor returns to master state at the end of the next external request. This can be a read response, or some other type of external request.

An external agent must note that the processor has performed an uncompelled change to slave state and begin driving the **SysAD** bus along with the **SysCmd** bus. As long as the System interface is in slave state, the external agent can begin an external request without arbitrating for the System interface; that is, without asserting **EReq***.

After the external request, the System interface returns to master state.

11.3.4.5 Signal Timing

The System interface protocol describes the cycle-by-cycle signal transitions that occur on the pins of the system interface to realize requests between the processor and an external agent. Figure 11.3.3 shows the timing relationships between System interface signal edges.

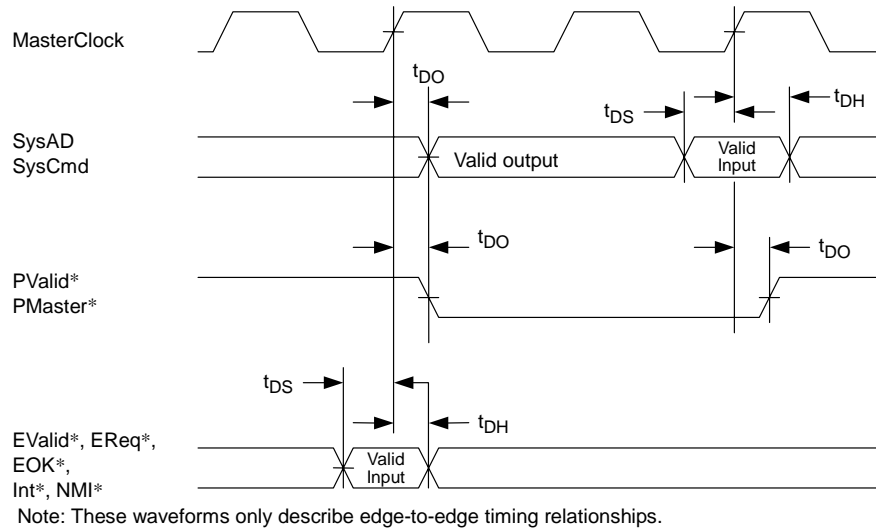Note: These waveforms only describe edge-to-edge timing relationships.

Figure 11.3.3 System Interface Edge Timing Relationships

The Timing Summary section below describes the minimum and maximum timing values of each signal. The sections that follow describe the timing requirements for various bus cycles.

11.3.5   Timing Summary

In the following timing diagrams, gray-scale signals indicate values that are either Unknown or Don't Cares, within the specification limits. They may be any value as long they do not violate any bus value or timing specification. The timing diagrams illustrate cycles using the following signals:

- **PMaster***
- **EValid***
- **PValid***
- **EOK***
- **EReq***

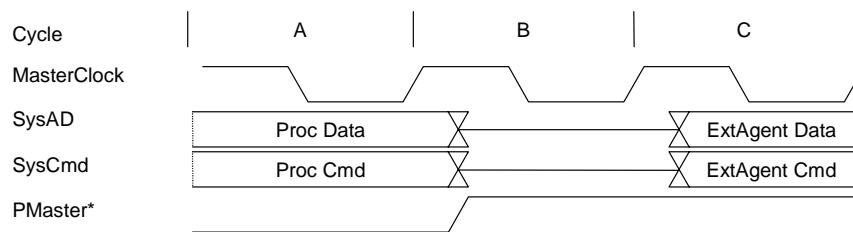**PMaster***   (O)   Indicates the processor is the master of the system interface bus.

Figure 11.3.4 Sample Cycle with *PMaster** Asserted, Then Deasserted

A   Processor drives SysAD and SysCmd buses (processor is master).

B   **PMaster** is deasserted. SysAD and SysCmd buses are set to a tri-state (no bus master).

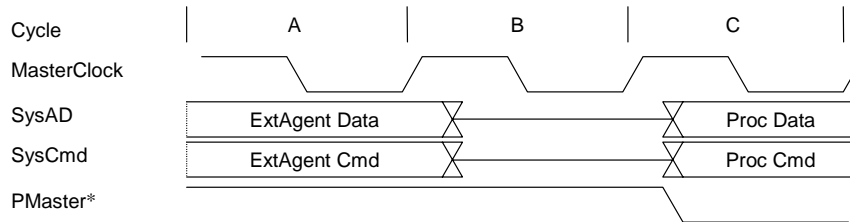C   External agent drives SysAD and SysCmd buses (external agent is master).

Figure 11.3.5 Sample Cycle with *PMaster*∗ Asserted

A External agent drives SysAD and SysCmd buses (external agent is master).

B SysAD and SysCmd buses are set to a tri-state (no bus master).

C **PMaster**∗ is asserted. Processor drives SysAD and SysCmd buses (processor is master).

EValid∗ **(I), PValid**∗ **(O)**   During a cycle in which either signal is asserted, the signal indicates a new valid address or valid data is on the SysAD bus, and a new valid command or data identifier is on the SysCmd bus. **EValid**∗ indicates an external agent is driving new SysAD and SysCmd values. **PValid**∗ indicates the processor is driving new SysAD and SysCmd values.
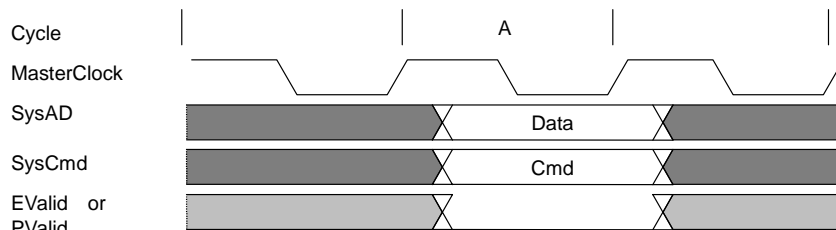

Figure 11.3.6 Sample Cycle with *PValid*∗ and *EValid*∗

A: New SysAD and SysCmd values.

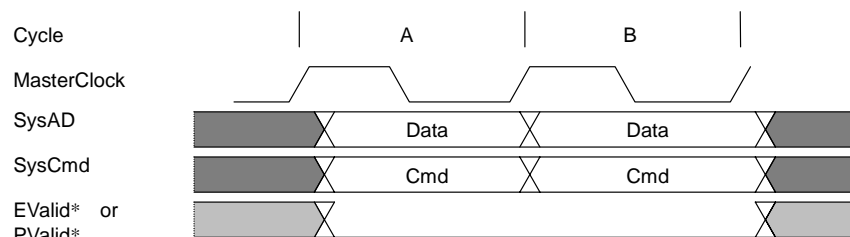Each cycle either of these signals remains asserted indicates there is a new SysAD and SysCmd value.

Figure 11.3.7 Sample Cycles with Multiple *PValid*∗ and *EValid*∗

A New SysAD and SysCmd value.

B Another new SysAd and SysCmd value.

EOK*   **(I)**   Indicates an external agent accepts a processor request. An external agent has excepted the processor read/write command if and only if the following has occurred:

A   **EOK*** is active.

B   The processor asserts **PValid*** and drives a read or write command. **EOK*** is asserted and the external agent accepts the processor command.

Once the external agent has accepted a processor write command, the agent must be able to accept the entire data size at the programmed data rate immediately following this command.

The external agent may provide read response data to the processor at any rate.

Deasserting EOK may kill a processor read/write request in progress. If this occurs, the external agent must ignore command and data from the processor in the following cycle.
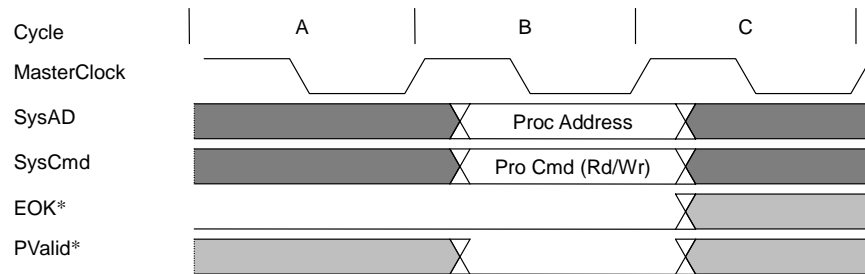
Figure 11.3.8 Sample Cycle with *EOK** Asserted

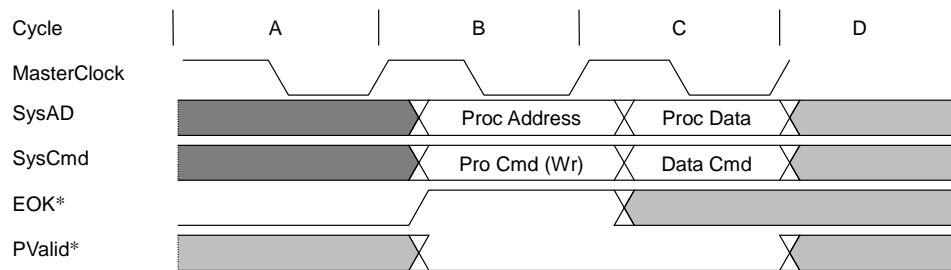Figure 11.3.9 Sample Cycle with *EOK** Asserted, Then Deasserted

A   **EOK*** is active.

B   Processor asserts **PValid*** and drives a read or write command. **EOK*** is deasserted (external agent has killed the processor's command).

C   The external agent must ignore any SysAD and SysCmd data from the processor.

D   The external agent *does not* ignore any SysAD and SysCmd data from the processor.

EReq*   **(I)**   Indicates an external agent is requesting bus ownership of the System interface. To gain mastership of the bus, an external agent must arbitrate with the processor as follows:

A   External agent asserts EReq*

B   Wait for PMaster* to be deasserted (1 to N cycles).

C   External agent drives SysAD and SysCmd buses. The external agent is guaranteed to maintain mastership of the bus as long as **EReq** is asserted.

If at any time **EReq*** is deasserted, the external agent must go back to step A and re-arbitrate for the bus.
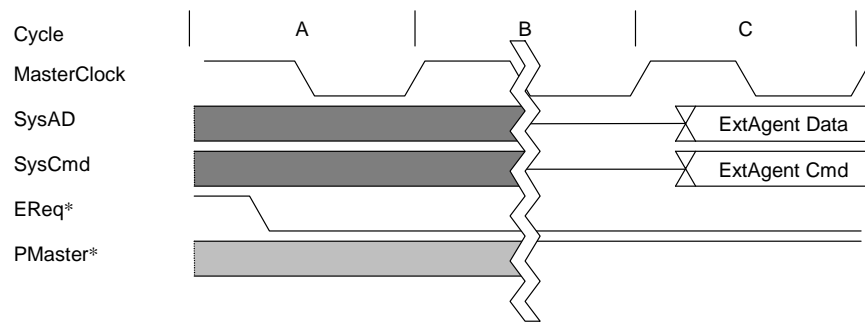
Figure 11.3.10 Sample Cycle with **EReq*** Asserted

From the time that **EReq*** is asserted, the external agent is guaranteed to gain mastership of the bus after at most one processor request. However, if **EOK*** is being deasserted, the external agent will gain mastership of the bus without having to accept any processor requests.

The external agent relinquishes bus mastership by deasserting **EReq*** as shown below:
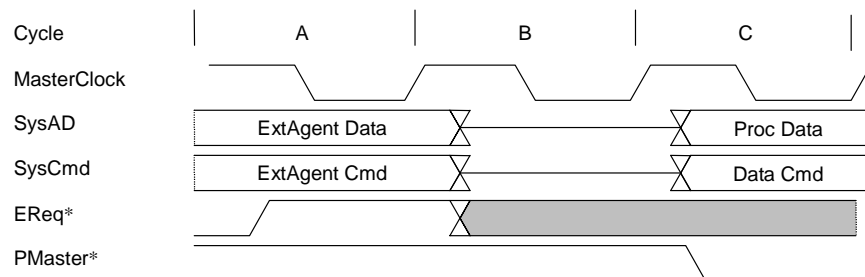
Figure 11.3.11 Sample Cycle with Deassertion of **EReq***

A   External agent deasserts EReq* and external agent drives the bus.

B   Bus is set to a tristate.

C   Processor regains mastership of bus.

Except for a processor read request (see below), assertion of **EReq*** is the only way the external agent gets and maintains bus mastership.
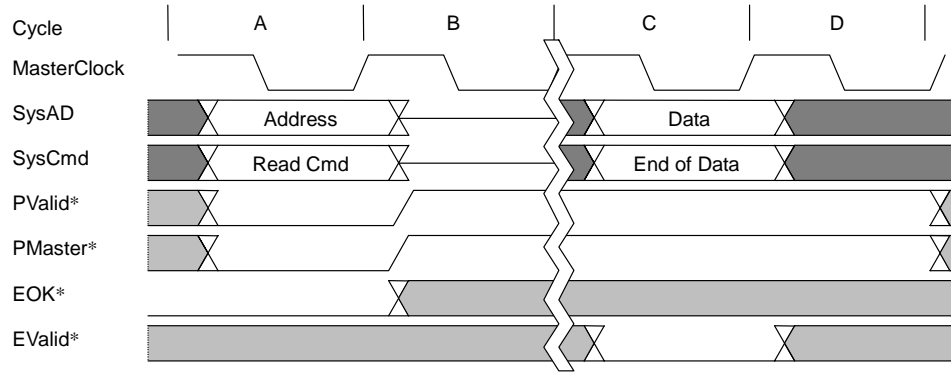
Figure 11.3.12 Sample Cycle with Assertion of *EReq*

A Processor drives a valid read command and an external agent accepts it.

B **PMaster*** is deasserted and the bus is set to a tristate.

C External agent drives last of requested data. During all cycles between B and C the external agent is guaranteed mastership of the bus.

11.3.6 Arbitration

The processor is the default master of the bus. It relinquishes ownership of the bus either when an external agent requests and is granted the system interface, or until the processor issues a read request. The transition from processor master to processor slave state is arbitrated by the processor, using the System interface handshake signals **EReq*** and **PMaster***.

When a processor read request is pending, the processor transitions to slave state by deasserting **PMaster***, allowing an external agent to return the read response data. The processor remains in slave state until the external agent issues an *End Of Data* read response, whereupon the processor reassumes mastership, signalled by the assertion of **PMaster***. Note that an external agent is able to retain mastership of the bus after an End Of Data read response if the external agent arbitrates for mastership using **EReq***.

When the processor is master, an external agent acquires control of the system interface by asserting **EReq***, and waiting for the processor to deassert **PMaster***. The processor is ready to enter slave state when it deasserts **PMaster***. The external agent must go through a three-step arbitration process (see the **EReq*** cycle in the Timing Summary) before driving the bus. Once the external agent has become master through **EReq*** arbitration, it can remain master as long as it continues to assert **EReq***. The System interface returns to master state (with the processor driving the bus) two cycles after **EReq*** is deasserted. Figure 11.3.13 illustrates an arbitration for external requests.
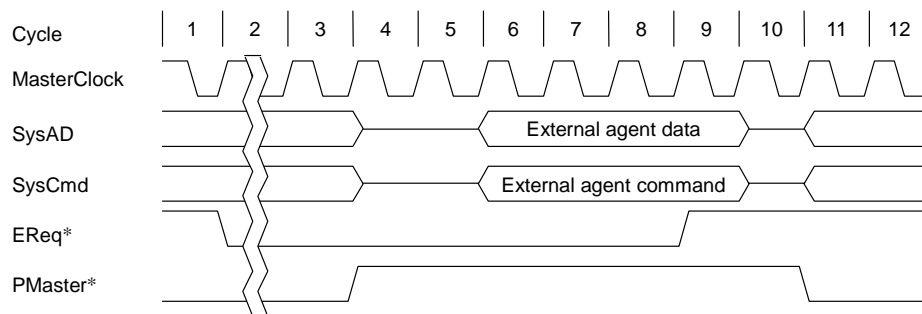
Figure 11.3.13 External Request Arbitration

When an external agent is master, it may always respond to a read with data. If the external agent has become master by **EReq***, it may issue transactions at will; that is, the processor must always accept any command or data on the bus at any time. There is no means for the processor to hold off the external agent once the external agent is master.

If the processor is in slave state and needs the bus,wait untile the external agent EReq* deasserted. Thereafter, when the processor sees **EReq*** deasserted, it resumes bus ownership, asserts the **PMaster*** line, and issues its own command. The processor becomes master and drives the bus two cycles after **EReq*** is deasserted.

An illustration of a processor request for bus mastership and the release of the bus by the external agent is illustrated in Figure 11.3.14.

Cycle 1 2 3 4 5 6 7 8 9 10 11 12

MasterClock

SysAD   External   agent data   Processor address/data

SysCmd   External   agent command   Processor command

PMaster*

EReq*

EOK*
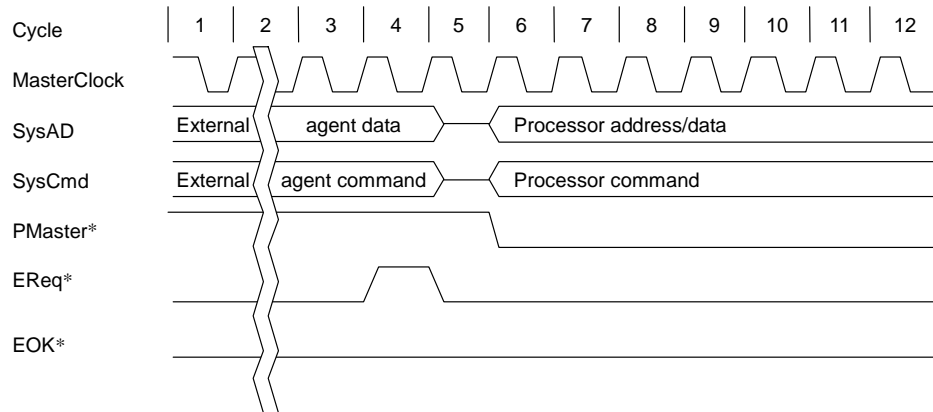
Figure 11.3.14 Processor Request For Bus Arbitration And External Agent Release

Upon assertion of **Reset*** or **ColdReset***, the processor becomes bus master and the external agent must become slave.

This protocol guarantees that either the processor or an external agent is always bus master. The master should never tristate the bus, except when giving up ownership of the bus under the rules of the protocol.

11.3.7   Issuing Commands

When the processor is master of the bus and wishes to issue a command, it cannot successfully issue the command until the external agent signals that it is ready to accept it. This readiness is indicated by assertion of the **EOK*** signal. Being master, the processor may place the command on the bus and continually reissue it while waiting for **EOK*** to be asserted; however, the command is not considered issued until **EOK*** has been asserted for two consecutive cycle (see the Timing Summary for **EOK*** earlier in this chapter).

If the **EOK*** signal is asserted in one cycle and then deasserted in the next, during which time a command is issued, that command is considered killed and must be retried. When a command is killed in this way, the processor begins to execute the read/write command. This action must be ignored by the external agent. If a write command is killed, the data cycle following this killed transaction must be ignored. If a read is killed, the processor releases the bus one cycle after and (assuming no **EReq***) regains mastership two cycles later. This allows the processor to retry the transaction.

11.3.8   Processor Write Request

A processor write request is issued by the following:

- driving a write command on the SysCmd bus
- driving a write address on the SysAD bus
- asserting **PValid*** for one cycle
- driving the appropriate number of data identifiers on the SysCmd bus
- driving data on the SysAD bus
- asserting **PValid***

For 1-to 4-byte writes, a single data cycle is used. 5-, 6- and 7-byte writes are broken up into two address/data transactions; one 4 bytes in size, the next handling the remaining 1, 2, or 3 bytes.

For all transactions larger than 7 bytes (e.g. 8, 16, 32), 4 bytes are sent on each data cycle until the appropriate number of bytes has been transferred. The final data cycle is tagged as end of data (EOD) on the command bus.

To be fully compliant with all implementations of this protocol, an external agent should be able to receive write data over any number of cycles with any number of idle cycles between any two data cycles. However, for the TX4951B processor implementation, data begins to arrive on the cycle immediately following the write issue cycle, and continues to arrive at a programed data rate thereafter. The processor drives data at the rate specified by the data rate configuration signals (see the section describing Data Rate Control, later in this chapter).

Writes may be cancelled and retried with the **EOK*** signal (see the section earlier, Issuing Commands).

Figure 11.3.15 illustrates the bus transactions for a 4-word data cache block store.
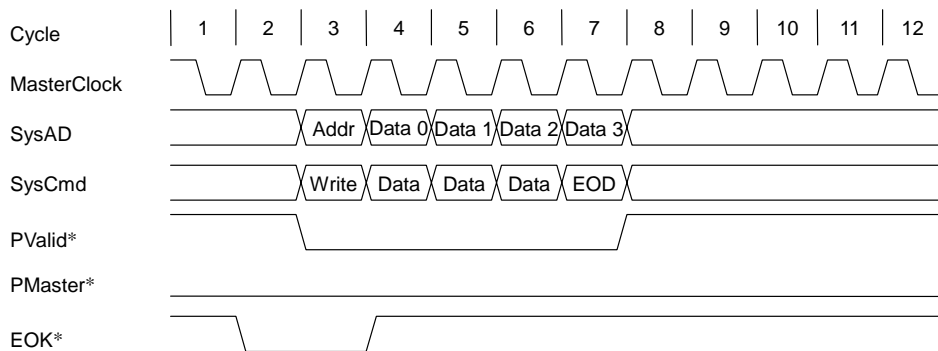

Figure 11.3.15 Processor Block Write Request With W Data Rate

Figure 11.3.16 illustrates a write request which is cancelled by the deassertion of **EOK*** during the address cycle of the second write, and which is retried when **EOK*** is asserted again.
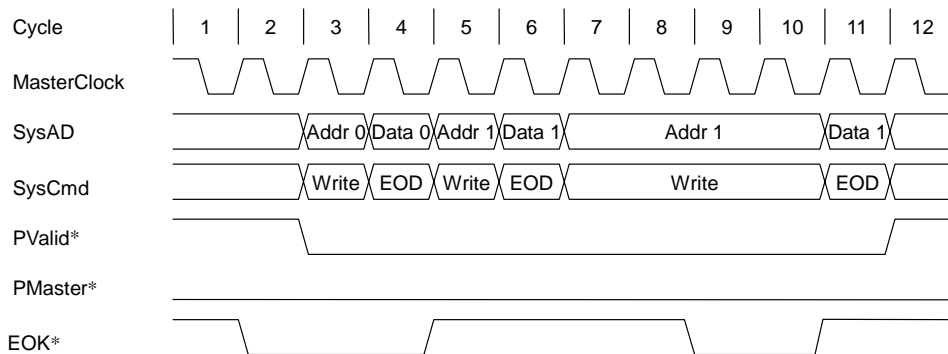
Figure 11.3.16 Processor Single Write Request Followed By A Cancelled And Retried Write Request

11.3.9 Processor Read Request

A processor read request is issued by the following:

- driving a read command on the SysCmd bus

- driving a read address on the SysAD bus

- asserting **PValid***

Only one processor read request may be pending at a time. The processor must wait for an external read response before starting a subsequent read.

The processor moves to slave state after the issue cycle of the read request, by deasserting the **PMaster*** signal. An external agent may then return the requested data through a read response. The external agent, which is now bus master, may issue any number of writes before sending the read response data.

An example of a processor read request and an uncompelled change to slave state occurring as the read request is issued is illustrated in Figure 11.3.17.
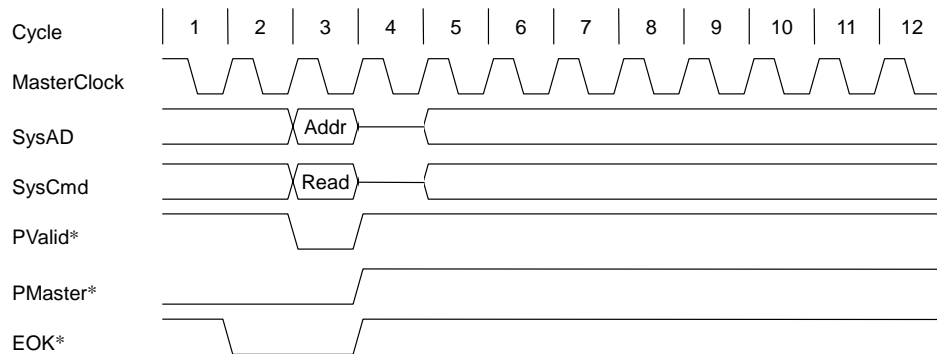
Figure 11.3.17 Processor Read Request

The TX4951B support the Read Time Out Function. This Function is when the retern data wait but the time out counter. See chapter 11.3.20 Mode Register of System Interface (G2SConfig).

11.3.10 External Write Request

External write requests are similar to a processor single write except that the signal EValid is asserted instead of the signal **PValid***. An external write request consists of the following:

- an external agent driving a write command on the SysCmd bus and a write address on the SysAD bus

- asserting **EValid*** for one cycle

- driving a data identifier on the SysCmd bus and data on the SysAD bus

- asserting **EValid*** for one cycle.

The data identifier associated with the data cycle must contain a last data cycle indication. Note that the external agent must gain and maintain bus mastership during these transactions (see **EReq*** in the Timing Summary, earlier in this chapter).

An external write request example with the processor initially in master state is illustrated in Figure 11.3.18.

Figure 11.3.18 External Write Request

An example of a read response for a processor single word read request that is interrupted by an external agent write request is illustrated in Figure 11.3.22. External writes can not occur in the middle of a data response block; they can, however, occur before the first data response of the data block or after the last EOD response, but it can not occur between them.

Note: The only writable resources are processor interrupts. An external write to any address is treated as a write to the processor interrupts.

11.3.11 External Read Response

An external agent returns data to the processor in response to a processor read request by waiting for the processor to move to slave state. The external agent then returns the data through either a single data cycle or a series of data cycles sufficient to transmit the requested data. After the last data cycle is issued, the read response is complete and the processor becomes master (assuming **EReq*** was not asserted).

If, at the end of the read response cycles, **EReq*** has been asserted, the processor remains in slave state until the external agent relinquishes the bus. When the processor is in slave state and needs access to the SysAD bus, it waits until **EReq*** is deasserted.

The data identifier associated with a data cycle may indicate that the data transmitted during that cycle is erroneous; however, an external agent must return a block of data of the correct size regardless of this erroneous data cycle indication. If a read response includes one or more erroneous data cycles, the processor takes a bus error.

Read response data must only be delivered to the processor when a processor read request is pending. The state of the processor is undefined if a read response is presented to it when no processor read is pending.

An example of a processor single read request followed by a read response is illustrated in Figure 11.3.19.
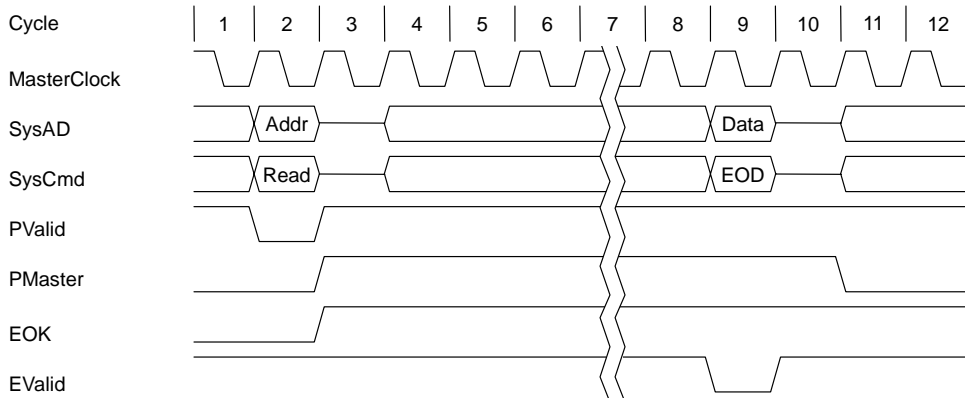
Figure 11.3.19   Single Read Request Followed By Read Response

A read response example for a processor block read with the system interface already in slave state is illustrated in Figure 11.3.20.
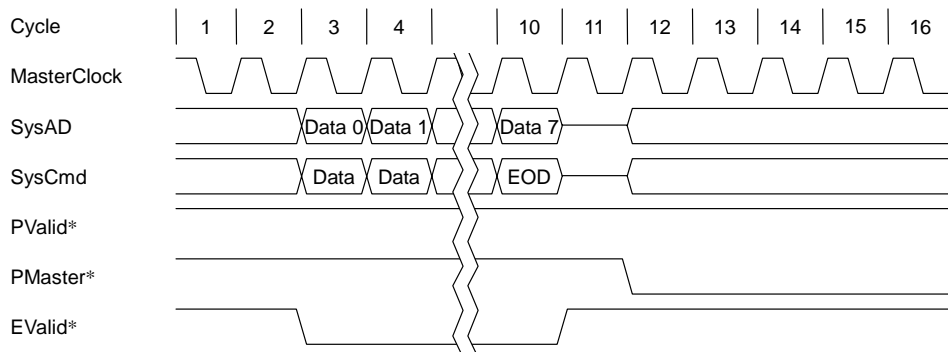
Figure 11.3.20 Block Read Response, System Interface Already In Slave State

A read response example for a processor single read request followed by an external agent write request is illustrated in Figure 11.3.21.

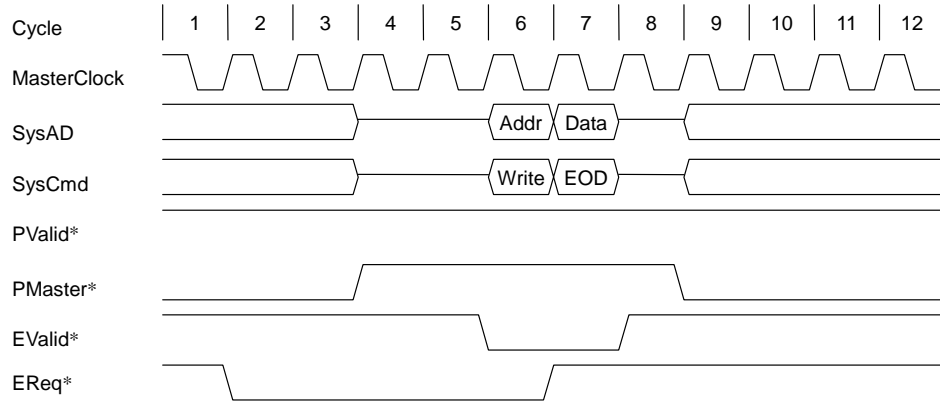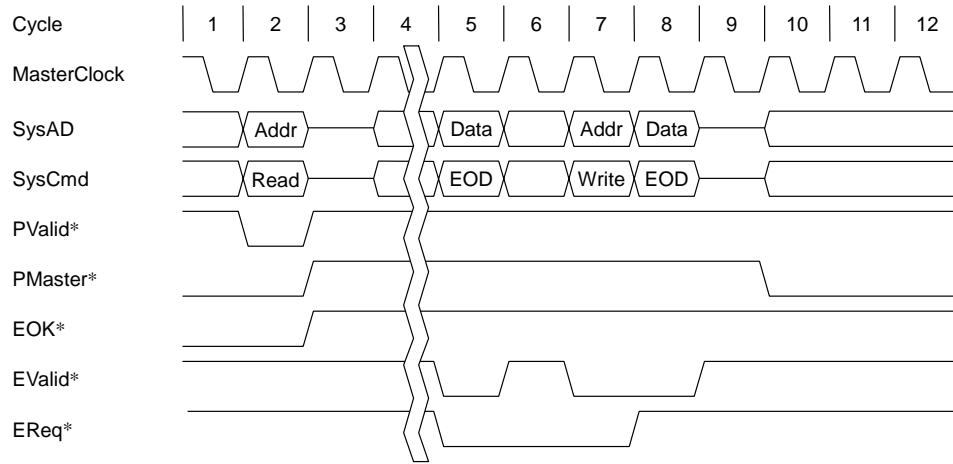Figure 11.3.21 Single Read Request Followed By External Write Request
(External Agent Keeps Bus)

An example of a read response for a processor single word read request that is interrupted by an external agent write request is illustrated in Figure 11.3.22. Cycle 5 is the data for the external write request in cycle 4. Cycle 7 is the read response data.
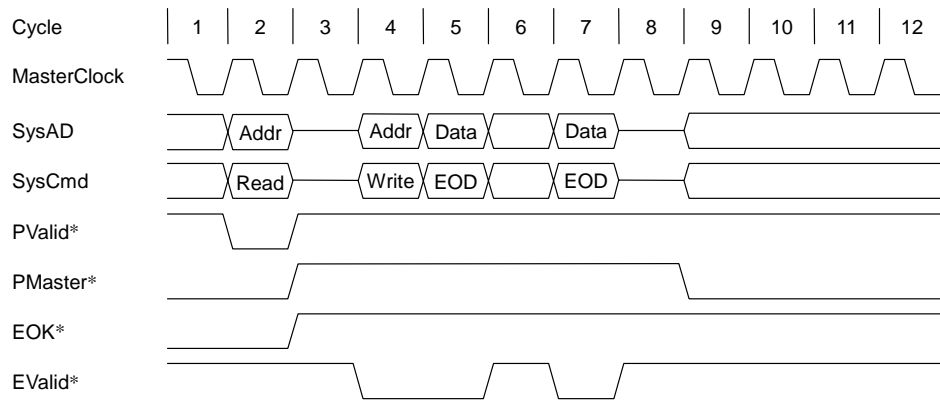
Figure 11.3.22 External Write Followed By External Read Response, System Interface In Slave State

11.3.12 Flow Control

EOK* may be used by an external agent to control the flow of processor read and write requests; while **EOK*** is deasserted the processor will repeat the current address cycle until an external agent signals it is ready, by asserting **EOK***. There is a one cycle delay from the assertion of **EOK*** to the state in which the Read/Write command becomes valid. **EOK*** must be asserted for two consecutive cycles for the command issue completion. Examples of **EOK*** use are given in Figure 11.3.23 and 6.3.24.
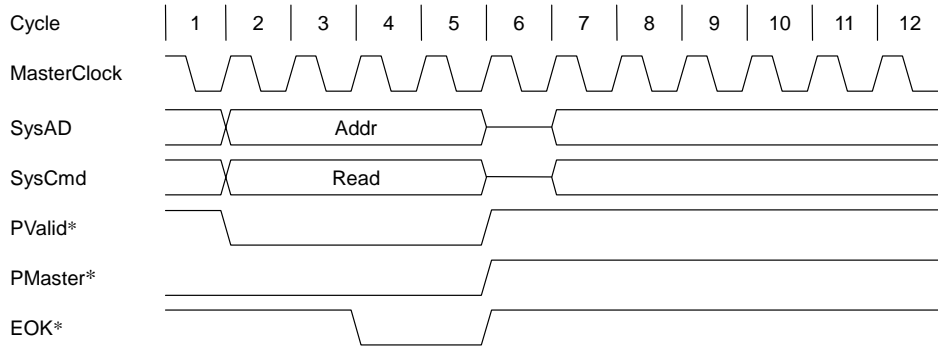
Figure 11.3.23 Delayed Processor Read Request
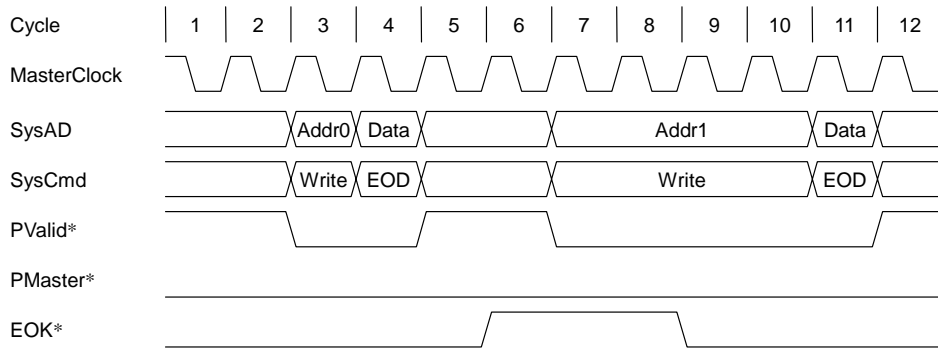
Figure 11.3.24 Two Processor Write Requests, Second Write Delayed

11.3.13 Data Rate Control

The System interface supports a maximum data rate of one word per cycle, and an external agent may deliver data to the processor at this maximum data rate. The rate at which data is delivered to the processor can be controlled by the external agent by driving data and asserting **EValid*** only when it wants data to be available.

The processor interprets cycles as valid data cycles when **EValid*** is asserted and the SysCmd bus contains a data identifier. The processor continues to accept data until the end of data (EOD) indicator is received.

The rate at which the processor transmits data to an external agent is programmed in the *WBRATE* field in *G2S Config* register. Data patterns are specified using the letters **W** and **x** (**W** indicates a word size data cycle and **x** indicates an unused, or idle, cycle). A data pattern is specified as a sequence of data and unused cycles that will be repeated to provide the appropriate number of data cycles for a given transfer. For example, a data pattern of **WWxx** indicates a data rate of two words every four cycles.

TX4951B supports two data rates, **W** and **Wxx**. During a cycle indicated by an **x**, the processor continues to hold the same data as the previous cycle.

A processor block write request for two words with Dxx pattern is illustrated in Figure 11.3.25; this transaction results from a store doubleword instruction.
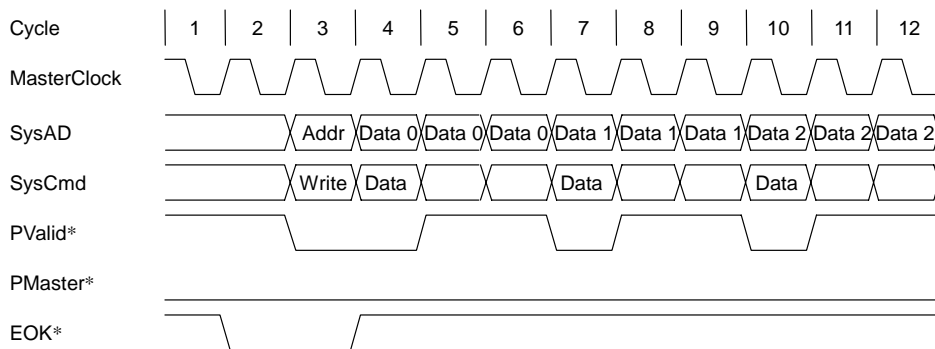
Figure 11.3.25 Processor Block Write Request With Wxx Data Rate

11.3.14 Consecutive SysAD Bus Transactions

The following figures (Figure 11.3.26 to Figure 11.3.29) illustrate the minimum cycles required between consecutive bus transactions.
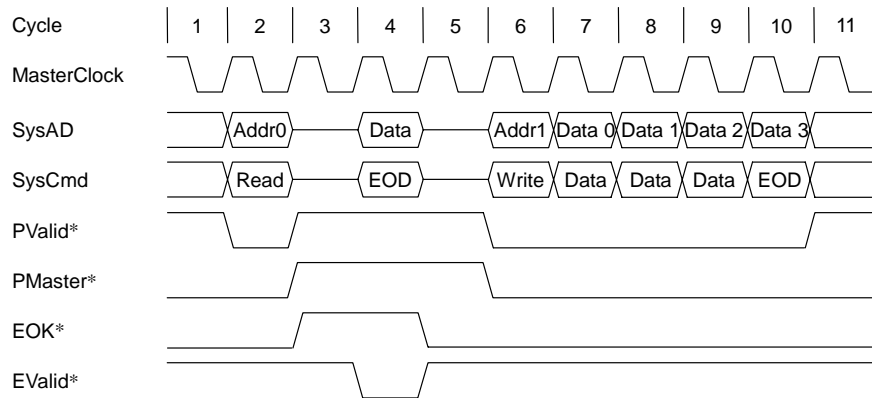
Figure 11.3.26   Processor Single Word Read Followed By Block Write Request
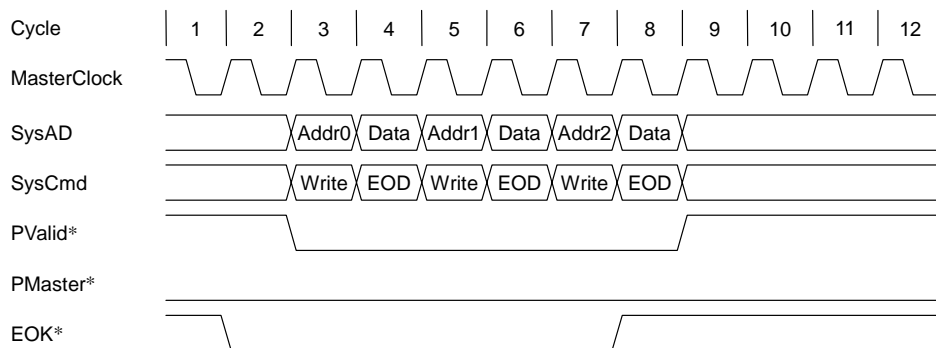
Figure 11.3.27   Consecutive Processor Single Word Write Requests With W Data Rate
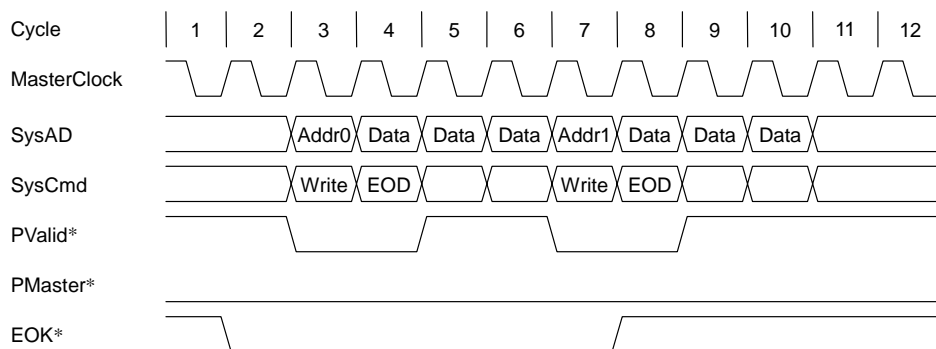
Figure 11.3.28   Consecutive Processor Single Word Write Requests With Wxx Data Rate
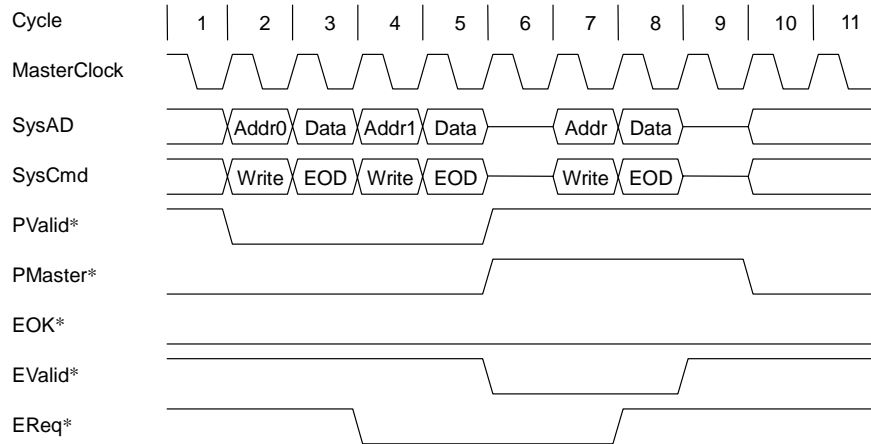
Figure 11.3.29 Consecutive Processor Write Requests Followed By External Write Request

Block Read Maximum Rate

Maximum block reads can occur with the following data rate:

AxW...WxAxW...W (1 cycle between W and A)

where **A** is the address, **W** is data (four words, or WWWW, in the data cache miss, and 8 words, or WWWWWWWW, in an instruction cache miss), and **x** is an idle cycle.

Back-to-Back Instruction Cache Misses

With a CPUCLK to GBUSCLK ratio of 2:1, back-to-back instruction cache misses can be refilled with the following data rate:

AxWWWWWWWWxxxxxxAxWWWWWWWW (6 cycles between W and A)

That is, the address is followed by an idle cycle, the instruction is executed, six idle cycles occur, followed by the next address. This pattern is valid for the case in which two sequential instructions miss in the instruction cache, each instruction residing on a different cache line.

Running completely in uncached space (every instruction is uncached and a cache miss) results in a similar data pattern:

AxWxxxxxxAxW (6 cycles between W and A)

Back-to-Back Uncached Loads

With a CPUCLK to GBUSCLK ratio of 2:1, back-to-back uncached doubleword loads have the following data rate:

AxWWxxxxxxAxWW (6 cycles between W and A)

That is, the address is followed by an idle cycle, a doubleword data (2-word data), six idle cycles, and the next address.

With a CPUCLK to GBUSCLK ratio of 2:1, back-to-back uncached word loads have the following data rate:

AxWxxxxxxAxW (6 cycles between W and A)

That is, the address is followed by an idle cycle, a word data, six idle cycles, and next address.

11.3.15 Starvation and Deadlock Avoidance.

Careful use of the **EReq*** signal allows a system to avoid starvation and deadlock situations.

Whenever an external agent needs the bus, it can request the bus by asserting **EReq***. The external agent is guaranteed to gain mastership of the bus after accepting at most one read/write request from the processor. If the external agent also deasserts **EOK***, it is guaranteed to gain mastership of the bus without accepting any read/write request from the processor.

The external agent can allow the processor to gain bus mastership, perform one read/write request and then relinquish mastership by the following sequence of actions:

1. deassert **EReq***
2. assert **EReq***
3. arbitrate for the bus while asserting **EOK***

The minimum deassertion of **EReq*** can be one cycle in length.

shows an external agent relinquishing the bus to allow a single read/write request from the processor. The external agent must be ready to accept this request by keeping **EOK*** asserted, otherwise the read/write request is held off or killed and the processor relinquishes bus mastership without extending a request. This could lead to starvation of the processor.
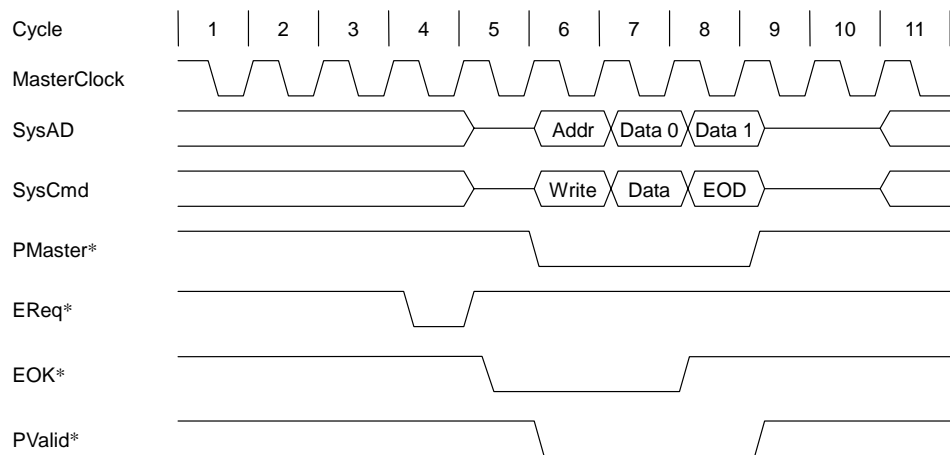
Figure 11.3.30 External Agent Gives Up Bus for One Processor Request

11.3.16 Discarding and Re-Executing Read Command

Figure 11.3.31 illustrates how a processor single read request is discarded and reexecuted. The following sequence describes the protocol.

- Because the **EOK*** signal is low in cycle 5, the processor tries to issue an address (cycle 6).
- If the **EOK*** signal is high at this point, the processor discards this read request and enters the slave status in the next cycle.
- Because the **EReq*** signal is inactive, the processor returns to the master status again and reissues a read request. Because the **EOK*** signal is low in both the cycles 7 and 8, the issuance cycle of the read request is determined.
- The external agent outputs data at the requsted address.
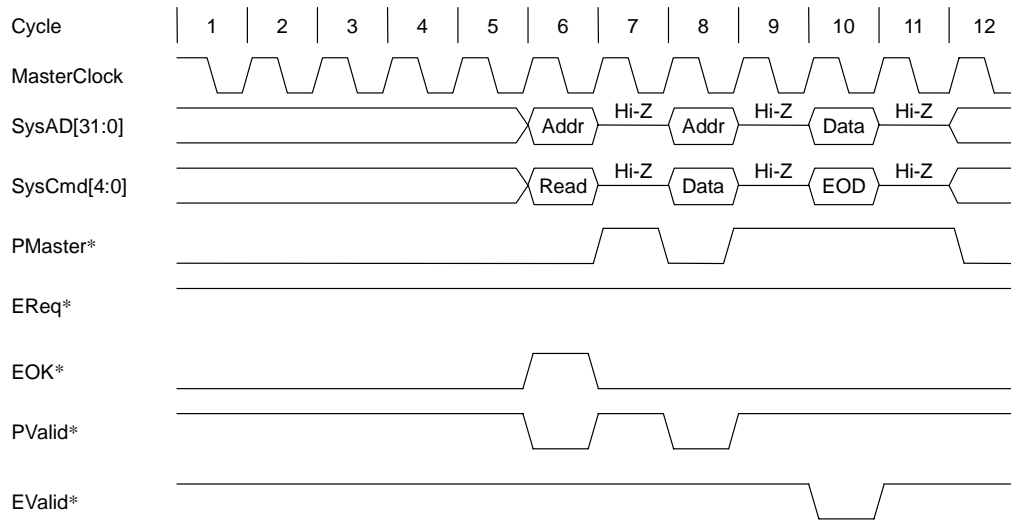
Figure 11.3.31 Discarding and Re-executing Processor Single Read Request

11.3.17 Multiple Drivers on the SysAD Bus

In most applications, the SysAD bus is a point-to-point connection between the processor and a bidirectional, registered transceiver located in an external agent. In this application, the SysAD bus has two possible drivers: the processor and the external agent.

However, an application may add additional drivers and receivers to the SysAD bus, allowing transmissions over the SysAD bus that bypass the processor. To accomplish this, the external agent(s) must coordinate its use of the SysAD bus by using arbitration handshake signals such as **EReq*** and **PMaster***.

To implement an independent transmission on the SysAD bus that does not involve the processor, the system executes the following sequence of actions:

1. The external agent(s) requests the SysAD bus by asserting **EReq***.

2. The processor releases the System interface to slave state.

3. The external agent(s) allows independent transmission over the SysAD bus, making certain the **EValid*** input to the processor is not asserted while the transmission occurs.

4. When the transmission is complete, the external agent(s) deasserts **EReq*** to return the system interface to master state.

To implement multiple drivers, separate **Valid** lines are required for non-processor chips to communicate.

11.3.18 Signal Codes

System interface commands and data identifiers are encoded in five bits on the SysCmd bus and transmitted between the processor and external agent during address and data cycles.

- When **SysCmd[4]** is a 0, the current cycle is an address cycle and **SysCmd[3:0]** contains a command.
- When **SysCmd[4]** is a 1, the current cycle is a data cycle and **SysCmd[3:0]** identifies data.

For commands and data identifiers associated with external requests, all bits and fields have a value or a suggested value.

For System interface commands and data identifiers associated with processor requests, reserved bits and reserved fields in the command or data identifier are undefined, except where noted.

For all System interface commands, the SysCmd bus specifies the system interface request type. The encoding of **SysCmd[4]** for system interface commands is Table 11.3.3.

Table 11.3.3 Encoding of System Interface Commands in **SysCmd[4]**

SysCmd[4]	Command
0	Address Cycle
1	Data Cycle

For address requests, the remainder of the SysCmd bus specifies the attributes of the address request, as follows:

- **SysCmd[3]** encodes the address request type.
- **SysCmd[2:0]** indicates the size of the address requests.

The encoding of SysCmd[3:2] for address requests is shown in Table 11.3.4.

Table 11.3.4 Encoding of **SysCmd[3]** and **SysCmd[2]** for Address Cycle

SysCmd[3]	Command	SysCmd[2]	Request Size
0	Read Request	0	Single data
1	Write Request	1	Block data

Note:TX4951B support only External Single data write request.

The encoding of **SysCmd[1:0]** for block or single address requests is shown in Table 11.3.5 and Table 11.3.6, respectively.

Table 11.3.5 Encoding of **SysCmd[1:0]** for Block Address Requests

SysCmd[1:0]	Block Size
0	Reserved.
1	Four words(only Data cache).
2	Eight worde.
3	Reserved.

Table 11.3.6   Encoding of **SysCmd[1:0]** for Single Address Requests

SysCmd[1:0]	Data size.
0	One byte valid (byte)
1	Two bytes valid (halfword).
2	Three bytes valid (tribyte).
3	Four bytes valid (single word)

The encoding of SysCmd[3:0] for processor data identifiers is described in Table 11.3.7. The encoding of SysCmd[3:0] for external data identifiers is illustrated in Table 11.3.8.

Table 11.3.7   Encoding of **SysCmd[3:0]** for Processor Data Identifiers

SysCmd[3]	Last Data Element Indication
0	Last data element
1	Not the last data element
SysCmd[2]	Reserved
SysCmd[1]	*Reserved for*: Good Data Indication
	Processor drives 0 (Data is error free)
SysCmd[0]	*Reserved for*: Data Checking Enable
	Processor drives 1 (Disable data checking)

Table 11.3.8   Encoding of **SysCmd[3:0]** for External Data Identifiers

SysCmd[3]	Last Data Element Indication
0	Last data element
1	Not the last data element
SysCmd[2]	Response Data Indication
0	Data is response data
1	Data is not response data
SysCmd[1]	*Reserved for*: Good Data Indication
0	Data is error free
1	Data is erroneous
SysCmd[0]	*Reserved for*: Data Checking Enable
	Processor ignores this field (Suggested drive of 1, disable data checking)

Note: External read requests for processor resources are not supported in the TX4951B processor.

11.3.19 Physical Addresses

Physical addresses are driven on all 32 bits (bits 31 through 0) of the SysAD bus during address cycles. Addresses associated with single read and write requests are aligned for the size of the data element; specifically, for single word requests, the low order two bits of the address are zero, for halfword requests, the low order bit of the address is zero. For byte and tribyte requests, the address provided is a byte address.

External agents returning read response data must support subblock ordering. Addresses associated with block read requests are aligned to the word of the desired data. The order in which data is returned in response to a processor block read request is:

- the word containing the addressed data word is returned first
- the remaining word(s) in the block are returned next, sequentially

Block writes are always block aligned

11.3.20 Mode Register of System Interface (G2SConfig)

The Mode Register of System Interface (G2SCofig) is a read/write register.

This register extend from TX4300.

This register is only WORD-Access.

Table 11.3.9 G2SConfig

Address	Field	Desctiption
0xF_FF10_0000	G2SConfig	Mode Register of System Interface

31	20	19	4	3	2	1	0
Reserved		TOutVal		TOut Enable	Reserved=0	BURST8	WBRATE
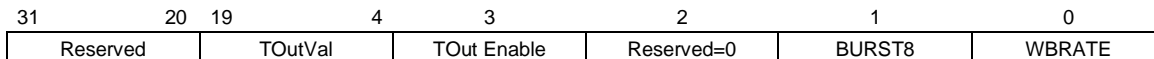
Figure 11.3.32 G2SConfig Register Format

Table 11.3.10 G2SConfig Register Formats (MODE43* = 0)

Bit	Field	Description	ColdReset	read/write
31:20	—	Reserved	Undefined	read
19:4	TOutVal	Set Data of Read Time Out Counter	0xFFFF	read/write
3	TOutEnable	Enable bit of Read Time Out Counter 0: Disable (Please set 0xffff in TOutVal) 1: Enable	0	read/write
2	—	Reserved	Undefined	
1	BURST8	Data Formats at 8-word burst write 0: Double burst mode 4 words × 2 1: Single burst mode 8 words	0	read/write
0	WBRATE	Set bit of Data Out Formats 0: Every cycle Data Out 1: 4-word Data Out per 12 cycles	0	read/write

TOSHIBA

Chapter 11 TX4951B System Interface

11.3.21 Read Time Out Counter (MODE43∗ = 0)

This counter is used to detect time-out when data is not returned during read.

The counter normally is set by loading the G2SConfig register's TOutVal as its

initial value. When one of the conditions below is met, the counter counts down

one every bus cycle and upon reaching the terminal count of 0, generates a time-out signal and asserts a bus error signal for one cycle before entering an idle state. If none of the following conditions is met, the value of TOutVal is reloaded into the counter.

- TOutEnable = 1
- Waiting for data

Note1: For TX4951B, Read Time Out Counter does not work when MODE43∗ = 1.
When the Read Time Out Counter functions is used, the value of the more than which added 24 the twice of the wait cycle of the main memory must be set to the TOutVal field.

Note2: Read Time Out Error occurs by writing 0x0 into TOutVal while Read Time Out Counter is disable.
So TOutVal must be set except for 0x0.

12. TX4951B Processor Interrupts

Four types of interrupt are available on the TX4951B. These are:

- one non-maskable interrupt, NMI
- six external interrupts
- two software interrupts
- one timer interrupt

These are described in this chapter.

12.1 Nonmaskable Interrupt

The non-maskable interrupt is signaled by asserting the **NMI*** pin (low), forcing the processor to branch to the Reset Exception vector. This pin is latched into an internal register by the rising edge of **GBUSCLK**, as shown in Figure 12.4.1. An NMI can also be set by an external write through the **SysAD** bus. On the data cycle, **SysAD[22]** acts as the write enable for **SysAD[6]**, which is the value to be written as the interrupt.

NMI only takes effect when the processor pipeline is running. Thus NMI can be used to recover the processor from a software hang (for example, in an infinite loop) but cannot be used to recover the processor from a hardware hang (for example, no read response from an external agent). NMI cannot cause drive contention on the **SysAD** bus and no reset of external agents is required.

This interrupt cannot be masked.

The **NMI*** pin is latched by the rising edge of **GBUSCLK**, however the NMI exception occurs in response to the falling edge of the **NMI*** signal, and is not level-sensitive.

Figure 12.4.1 shows the internal derivation of the **NMI** signal. The **NMI*** pin is latched into an internal register by the rising edge of **GBUSCLK**. Bit 6 of the Interrupt register is then ORed with the inverted value of **NMI*** to form the nonmaskable interrupt.

12.2 External Interrupts

External interrupts are set by asserting the external interrupt pins **Int[3:0]***. They also may be set by an external write through the **SysAD** bus. During the data cycle, **SysAD[19:16]** are the write enables for bits **SysAD[3:0]**, which are the values to be written as interrupts.

These interrupts can be masked with the *IM*, *IE*, and *EXL* fields of the *Status* register.

Note: Int5* is doubled as a Timer Interrupt. Therefore either is selected with the External pin (TintDis).

12.3 Software Interrupt

Software interrupts use bits 1 and 0 of the interrupt pending, *IP*, field in the *Cause* register. These may be written by software, but there is no hardware mechanism to set or clear these bits.

These interrupts are maskable.

12.4 Timer Interrupt

The timer interrupt signal is bit 15 of the *Cause* register, which is bit 7 of the interrupt pending, *IP*, field. The timer interrupt is set whenever the value of the *Count* register equals the value of the *Compare* register.

This interrupt is maskable through the *IM* field of the *Status* register.

Note: Interrupt register : Please see Chapter 6 External Write Request

Figure 12.4.1 TX4951B Interrupt Control Circuit

12.5 Asserting Interrupts

External writes to the CPU are directed to various internal resources, based on an internal address map of the processor. An external write to any address writes to an architecturally transparent register called the *Interrupt* register; this register is available for external write cycles, but not for external reads.

During a data cycle, **SysAD[22:16]** are the write enables for the seven individual *Interrupt* register bits and **SysAD[6:0]** are the values to be written into these bits. This allows any subset of the *Interrupt* register to be set or cleared with a single write request. Figure 12.4.1 shows the mechanics of an external write to the *Interrupt* register, along with the nonmaskable interrupt described earlier.

Figure shows how the TX4951B hardware interrupts are readable through the *Cause* register.

- Bits 3-0 of the *Interrupt* register are bit-wise ORed with the current value of the interrupt pins **Int*****[3:0]** and the result is directly readable as bits 13-10 of the *Cause* register.

- Bit5 of the Interrupt register is ORed with the current value of the interrupt pin Int5*. Bit15 of Cause register is selected this ORed signal or Timer Interrupt by TintDis.

IP[1:0] of the *Cause* register, which are described in Chapter 8 8.3, are software interrupts. There is no hardware mechanism for setting or clearing the software interrupts.

Figure 12.5.1 shows the masking of the TX4951B interrupt signals.

- *Cause* register bits 15-8 (IP7-IP0) are AND-ORed with *Status* register interrupt mask bits 15-8 (IM7-IM0) to mask individual interrupts.

- *Status* register bit 0 is a global Interrupt Enable (IE). It is ANDed with the output of the AND-OR logic to produce the TX4951B interrupt signal. The *EXL* bit in the *Status* register also enables these interrupts.
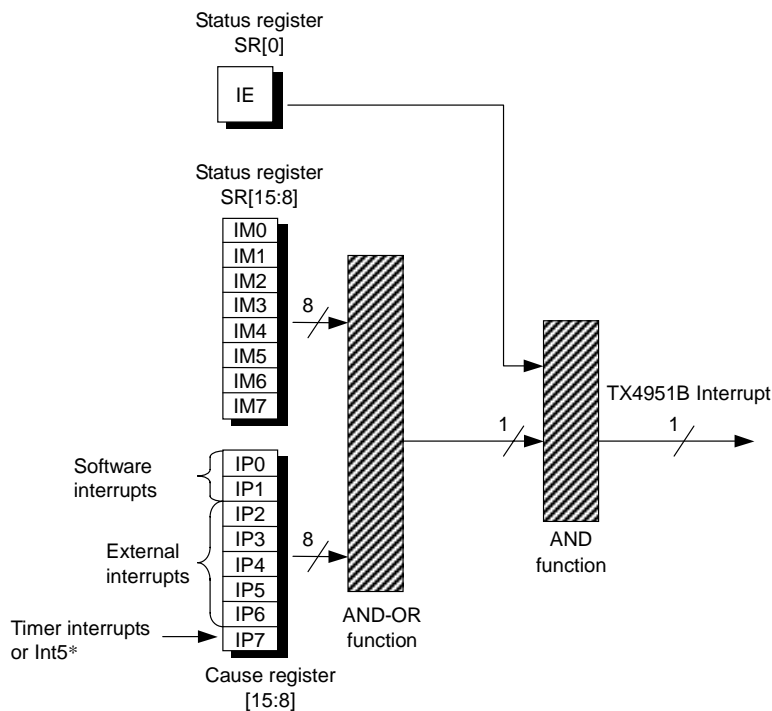

Figure 12.5.1 Masking of the TX4951B Interrupts

13. Power-Saving Modes

The TX4951B has these power-saving modes:

- Halt mode
- Doze mode

The TX4951B enters neither Halt nor Doze mode by just programming the Config register. The WAIT instruction puts the TX4951B in Halt or Doze mode.

13.1 Halt Mode

The Halt mode reduces power consumption by halting the TX4951B almost altogether. In Halt mode, the internal clocks of the TX49/L3 core within the TX4951B are partially stopped. If the *HALT* bit in the *Config* register is cleared, executing the WAIT instruction in Normal Operation mode causes the TX4951B to enter Halt mode.

When the TX4951B enters Halt mode, the **HALTDOZE** signal is asserted. Even when an external agent has ownership of the **SysAD** bus, executing the WAIT instruction puts the TX4951B in Halt mode, but the **HALTDOZE** signal is asserted after the current bus cycle is completed. If the on-chip write buffer has any store information, the WAIT instruction puts the TX4951B in Halt mode after it is emptied.

Assertion of **Int[3:0]***, **NMI*** or **ColdReset*** brings the TX4951B out of Halt mode. The TX4951B recognizes the **Int[3:0]*** inputs, irrespective of the settings of the *IntMask* bits in the *Status* register. On a return from Halt mode except via a masked **Int[3:0]*** signal, the TX4951B takes a corresponding exception. At this time, the *EPC* register points to the address of the instruction following the WAIT instruction. When the TX4951B is taken out of Halt mode as the result of a masked **Int[3:0]*** signal, processing resumes from the instruction following the one that was being executed when the TX4951B went into Halt mode.

13.2 Doze Mode

The Doze mode is also a software-controlled feature that reduces power consumption of the TX4951B. In Doze mode, the CPU pipeline status is retained and the internal clocks of the TX4951B are not stopped. If the *HALT* bit in the *Config* register is set, executing the WAIT instruction causes the TX4951B to enter Doze mode.

When the TX4951B enters Doze mode, the **HALTDOZE** signal is asserted. Even when an external agent has ownership of the **SysAD** bus, executing the WAIT instruction puts the TX4951B in Doze mode, but the **HALTDOZE** signal is asserted after the current bus cycle is completed. If the on-chip write buffer has any store information, the WAIT instruction puts the TX4951B in Doze mode after it is emptied.

Assertion of **Int[3:0]***, **NMI*** or **ColdReset*** brings the TX4951B out of Doze mode. The TX4951B recognizes the **Int[3:0]*** inputs, irrespective of the settings of the *IntMask* bits in the *Status* register. On a return from Doze mode except via a masked **Int[3:0]*** signal, the TX4951B takes a corresponding exception. At this time, the *EPC* register points to the address of the instruction following the WAIT instruction. When the TX4951B is taken out of Doze mode as the result of a masked **Int[3:0]*** signal, processing resumes from the instruction following the one that was being executed when the TX4951B went into Doze mode.

13.3 Status Shifts

Figure 13.3.1 shows the status shifts in the operation mode of the TX49.
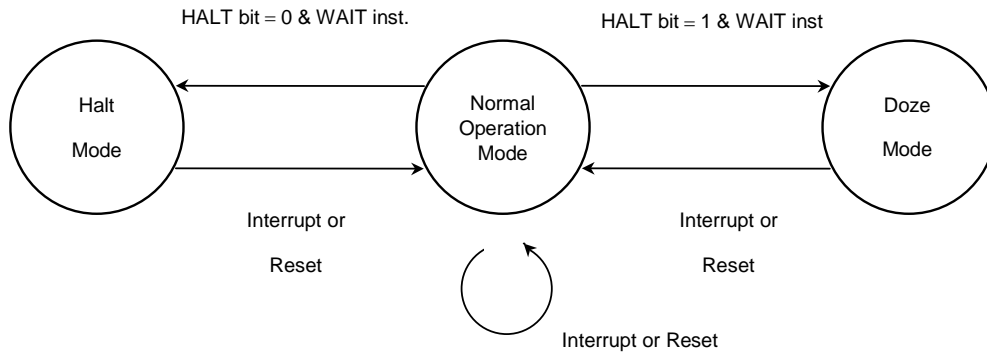
Figure 13.3.1 Status Shift Among Normal Operation Mode and Low Power Consumption Modes

When operation status shifts from the normal operation mode to the halt mode, it is returned to the normal operation mode by an interrupt or a reset. Similarly, when it shifts from the normal operation mde to the doze mode, it is returned to the normal operation mode by an interrupt or a reset. After a reset, the TX4951B is initialized to the normal operation mode.

14. JTAG Interface

The TX4951B processor provides a boundary-scan interface that is compatible with Joint Test Action Group (JTAG) specifications, using the industry-standard JTAG protocol (IEEE Standard 1149.1/D6).

This chapter describes that interface, including descriptions of boundary scanning, the pins and signals used by the interface, and the Test Access Port (TAP).

14.1 What Boundary Scanning Is

With the evolution of ever-denser integrated circuits (ICs), surface-mounted devices, double-sided component mounting on printed-circuit boards (PCBs), and buried vias, in-circuit tests that depend upon making physical contact with internal board and chip connections have become more and more difficult to use. The greater complexity of ICs has also meant that tests to fully exercise these chips have become much larger and more difficult to write.

One solution to this difficulty has been the development of *boundary-scan* circuits. A boundary-scan circuit is a series of shift register cells placed between each pin and the internal circuitry of the IC to which the pin is connected, as shown in Figure 14.1.1. Normally, these boundary-scan cells are bypassed; when the IC enters test mode, however, the scan cells can be directed by the test program to pass data along the shift register path and perform various diagnostic tests. To accomplish this, the tests use the four signals described in the next section: **JTDI**, **JTDO**, **JTMS**, **JTCK, and TRST***.
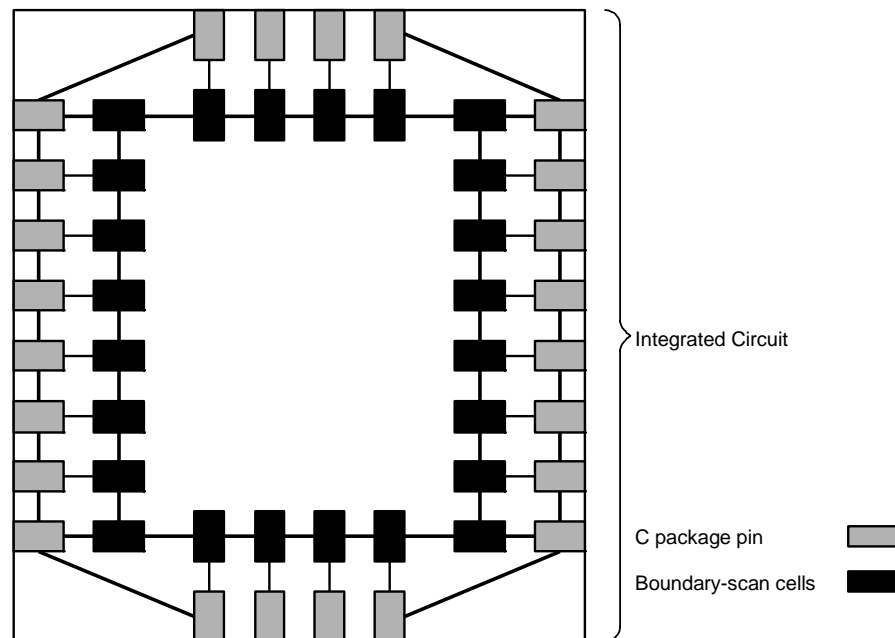
Figure 14.1.1 JTAG Boundary-scan Cells

14.2 Signal Summary

The JTAG interface signals are listed below and shown in Figure 14.2.1.

- JTDI   JTAG serial data in
- JTDO   JTAG serial data out
- JTMS   JTAG test mode select
- JTCK   JTAG serial clock input
- TRST   JTAG test reset input
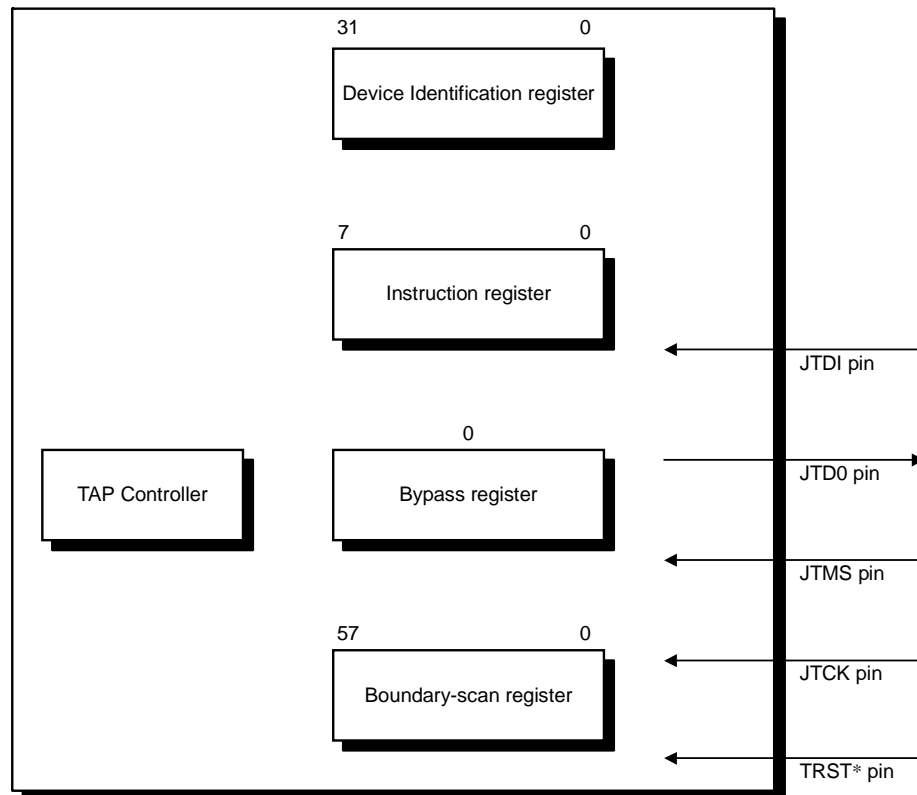
Figure 14.2.1 JTAG Interface Signals and Registers

The JTAG boundary-scan mechanism (referred to in this chapter as *JTAG mechanism*) allows testing of the connections between the processor, the printed circuit board to which it is attached, and the other components on the circuit board.

The JTAG mechanism does not provide any capability for testing the processor itself.

14.3 JTAG Controller and Registers

The processor contains the following JTAG controller and registers:

- *Instruction* register
- *Boundary-scan* register
- *Bypass* register
- *ID Code* register
- Test Access Port (TAP) controller

The processor executes the standard JTAG EXTEST operation associated with External Test functionality testing.

The basic operation of JTAG is for the TAP controller state machine to monitor the JTMS input signal. When it occurs, the TAP controller determines the test functionality to be implemented. This includes either loading the JTAG instruction register (IR), or beginning a serial data scan through a data register (DR), listed in Table 8-1. As the data is scanned in, the state of the JTMS pin signals each new data word, and indicates the end of the data stream. The data register to be selected is determined by the contents of the *Instruction* register.

14.3.1 Instruction Register

The JTAG *Instruction* register includes eight shift register-based cells; this register is used to select the test to be performed and/or the test data register to be accessed. As listed in Table 14.3.1, this encoding selects either the *Boundary-scan* register or the *Bypass* register or Device Identification register.

Table 14.3.1 JTAG Instruction Register Bit Encoding

Instruction Code (MSB → LSB)	Instruction	Selected Data Register
00000000	EXTEST	Boundary Scan Register
00000001	SAMPLE/PRELOAD	Boundary Scan Register
00000010	Reserved	Reserved
00000011	IDCODE	Device Identification register
00000100 ~ 01111111	Reserved	Reserved
10000000 ~ 11111110	Debug Support Unit	Please refer DSU section
11111111	BYPASS	Bypass register

Figure 14.3.1 shows the format of the *Instruction* register

7	6	5	4	3	2	1	0
MSB							LSB

Figure 14.3.1 Instruction Register

The instruction code is shifted out to the Instruction register from the LSB.

Figure 14.3.2 Instruction Register Shift Direction

14.3.2 Bypass Register

The *Bypass* register is 1 bit wide. When the TAP controller is in the Shift-DR (Bypass) state, the data on the JTDI pin is shifted into the *Bypass* register, and the *Bypass* register output shifts to the JTDO output pin.

In essence, the *Bypass* register is a short-circuit which allows bypassing of board-level devices, in the serial boundary-scan chain, which are not required for a specific test. The logical location of the *Bypass* register in the boundary-scan chain is shown in Figure 14.3.3. Use of the *Bypass* register speeds up access to boundary-scan registers in those ICs that remain active in the board-level test datapath.
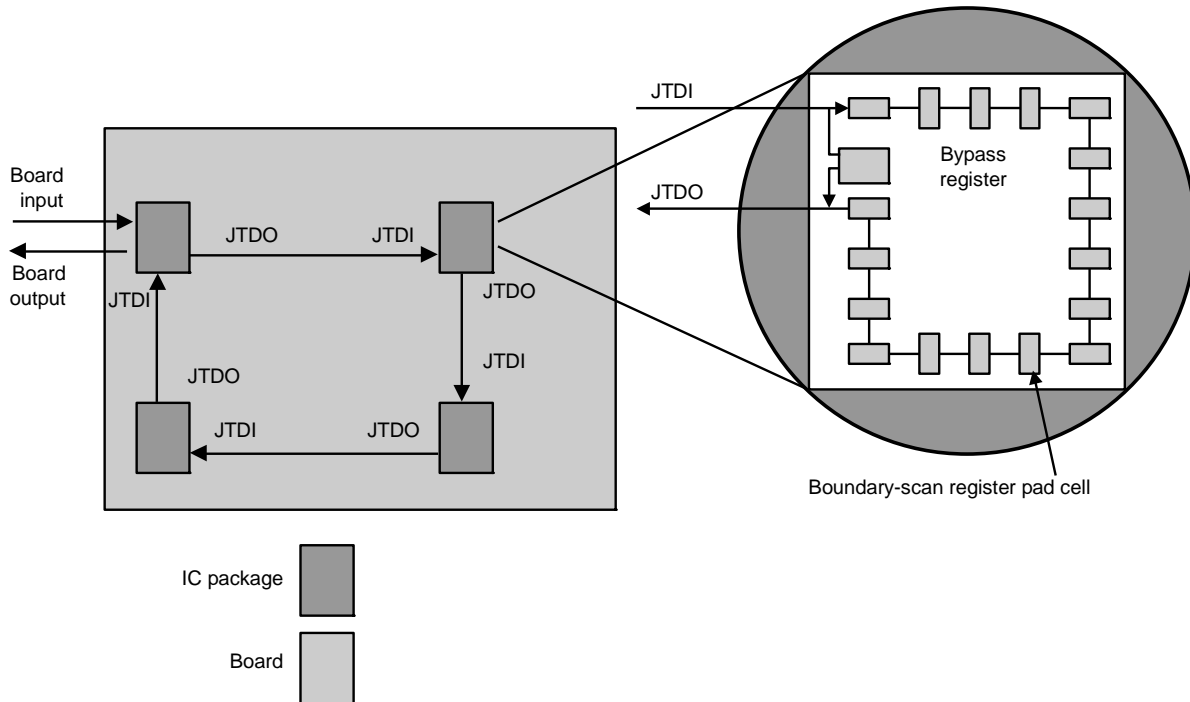
Figure 14.3.3 Bypass Register Operation

14.3.3 Boundary-Scan Register

The *Boundary Scan* register includes all of the inputs and outputs of the TX4951B processor, except some clock and phase lock loop signals. The pins of the TX4951B chip can be configured to drive any arbitrary pattern by scanning into the *Boundary Scan* register from the Shift-DR state. Incoming data to the processor is examined by shifting while in the Capture-DR state with the *Boundary Scan* register enabled.

The *Boundary-scan* register is a single, 58-bit-wide, shift register-based path containing cells connected to all input and output pads on the TX4951B processor. Figure 14.3.4 shows the *Boundary-scan* register.

57   0

See Table 14.3.2
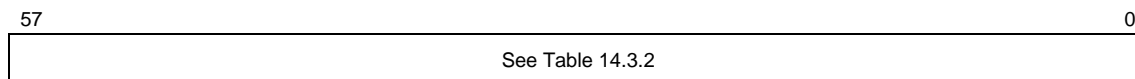
Figure 14.3.4 Format of the Boundary-scan Register

The most-significant bit, SysCntl (bit 57 to bit 56), are the JTAG output enable bit for all outputs of the processor. Output is enabled when those bit are set to 1 (default state).

The remaining 56 bits correspond to 56 signal pads of the processor.

At the end of this chapter, Table 14.3.2 lists the scan order of these 60 scan bits, starting from **JTDI** and ending with **JTDO**.

The JTDI input is loaded to the LSB of the Boundary Scan register. The MSB of the Boundary Scan register is retrieved from the JTDO output.

14.3.4 Device Identification Register

The Device Identification register is a 32-bit shift register. It is used to read serially from the IC the identification code indicating the IC manufacturer, part number, and version.

The following shows the Device Identification register structure.

31   28 27   12 11   1   0

Version	Product number	manufacturer identification code	1
4 bits	16 bits	11 bits	
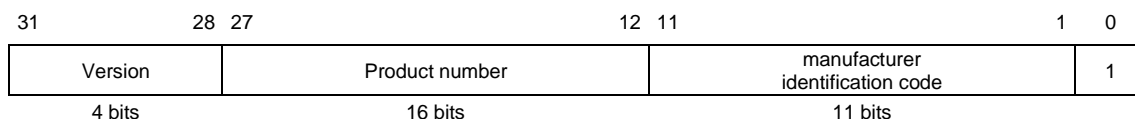
Figure 14.3.5 Device Identification Register

The TX4951B device identification code is 0x0002d031.

The device identification code is shifted out starting from the LSB.
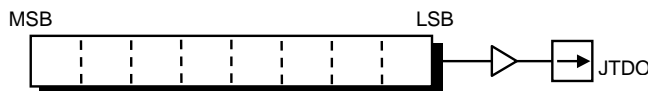
Figure 14.3.6 Devide Identification Register shift Direction

14.3.5 Test Access Port (TAP)

The Test Access Port (TAP) consists of the five signal pins: **TRST* JTDI**, **JTDO**, **JTMS**, and **JTCK**. Serial test data and instructions are communicated over these five signal pins, along with control of the test to be executed.

As Figure 14.3.7 shows, data is serially scanned into one of the four registers (*Instruction* register, *Bypass* register, Device Identification register, or the *Boundary-scan* register) from the **JTDI** pin, or it is scanned from one of these four registers onto the **JTDO** pin.

The **JTMS** input controls the state transitions of the main TAP controller state machine.

The **JTCK** input is a dedicated test clock that allows serial JTAG data to be shifted synchronously, independent of any chip-specific or system clocks.
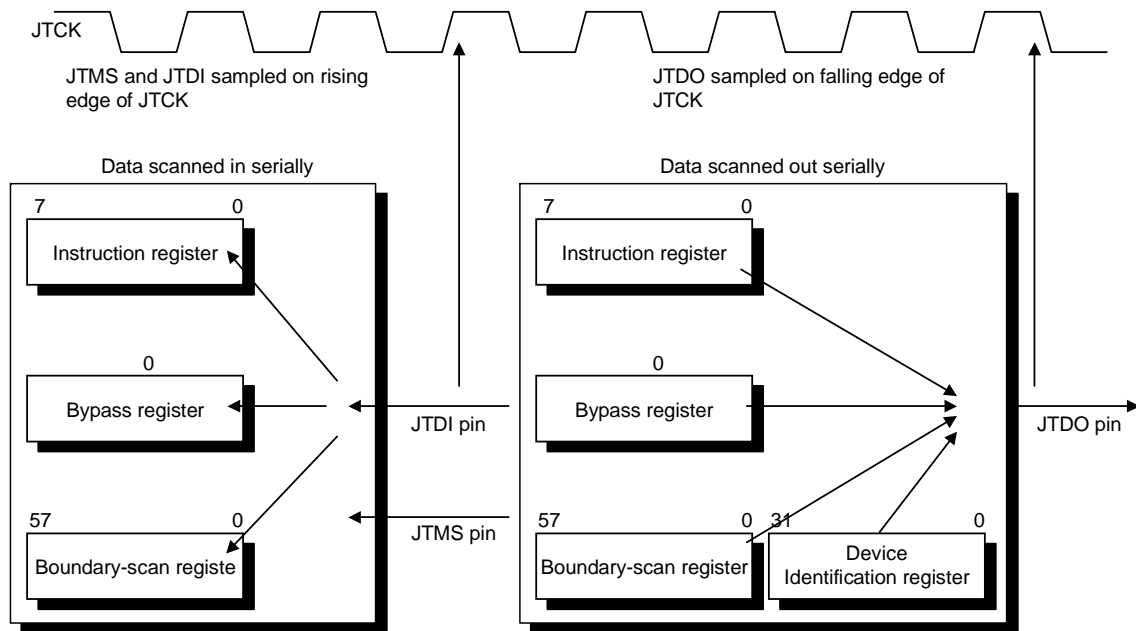
Figure 14.3.7 JTAG Test Access Port

Data on the **JTDI** and **JTMS** pins is sampled on the rising edge of the **JTCK** input clock signal. Data on the **JTDO** pin changes on the falling edge of the **JTCK** clock signal.

14.3.6 TAP Controller

The processor implements the 16-state TAP controller as defined in the IEEE JTAC specification.

14.3.7 Controller Reset

The TAP controller state machine can be put into Reset state the following:

- assertion of the **TRST*** signal (Low) resets the TAP controller.
- keeping the **JTMS** input signal asserted through five consecutive rising edges of **JTCK** input.

In either case, keeping **JTMS** asserted maintains the Reset state.

14.3.8 TAP Controller

The state transition diagram of the TAP controller is shown in Figure 14.3.8. Each arrow between states is labeled with a 1 or 0, indicating the logic value of JTMS that must be set up before the rising edge of JTCK to cause the transition.
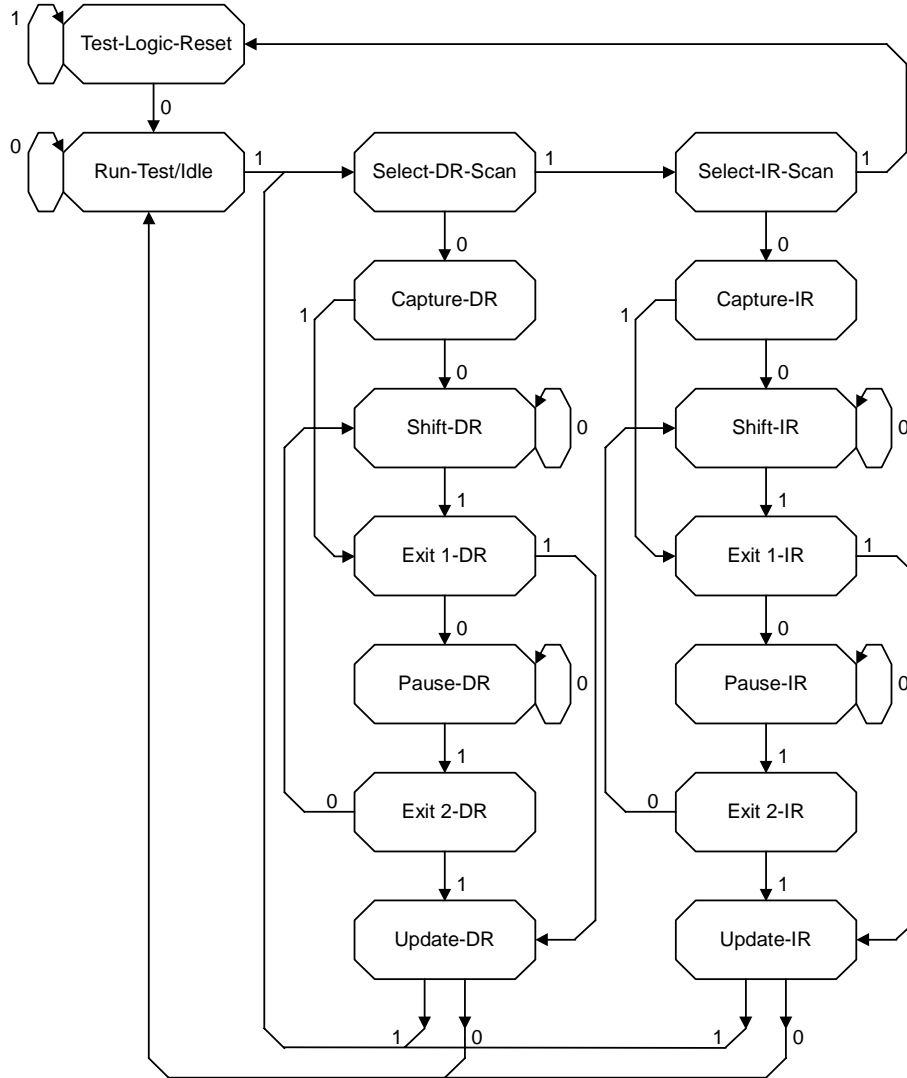
Figure 14.3.8 TAP Controller State Diagram

The following paragraphs describe each of the controller state. The left vertical column in Figure 14.3.8 is the data column, and the right vertical column is the instruction column. The data column and instruction column reference data register (DR) and instruction register (IR), respectively.

- Test-Logic-Reset

  When the TAP controller is in the Reset state, the value 0x3 is loaded into the parallel output latch, selecting the Device Identification register as default. The three most significant bits of the Boundary-scan register are cleared to 0, disabling the outputs.

  The controller remains in this state while JTMS is high. If JTMS is held low while the controller is in this state, then the controller moves to the Run-Test/Idle state.

- Run-Test/Idle

  In the Run-Test/Idle state, the IC is put in a test mode only when certain instructions such as a built-in self test (BIST) instruction are present. For instructions that do not cause any activities in this state, all test data registers selected by the current instruction retain their previous states.

  The controller remains in this state while JTMS is held low. When JTMS is high, the controller moves to the Select-DR-Scan state.

- Select-DR-Scan

  This is a temporary controller state. Here, the IC does not execute any specific functions.

  If JTMS is held low when the controller is in this state, then the controller moves to the Capture-DR state. If JTMS is held high, the controller moves to the Select-IR-Scan state in the instruction column.

- Select-IR-Scan

  This is a temporary controller state. Here, the IC does not execute any specific functions.

  If JTMS is held low when the controller is in this state, then the controller moves to the Capture-IR state. If JTMS is held high, the controller returns to the Test-Logic-Reset state.

- Capture-DR

  In this controller state, if the test data register selected by the current instruction on the rising edge of JTCK has parallel inputs, then data can be parallel-loaded into the shift portion of the data register. If the test data register does not have parallel inputs, or if data need not be loaded into the selected data register, then the data register retains its previous state.

  If JTMS is held low while the controller is in this state, the controller moves to the Shift-DR state. If JTMS is held high, the controller moves to the Exit1-DR state.

- Shift-DR

  In this controller state, the test data register connected between JTDI and JTDO shifts data one stage forward towards its serial output.

  When the controller is in this state, then it remains in the Shift-DR state if JTMS is held low, or moves to the Exit1-DR state if JTMS is held high.

- Exit 1-DR

  This is a temporary controller state.

  If JTMS is held low when the controller is in this state, then the controller moves to the Pause-DR state. If JTMS is held high, the controller moves to the Update-DR state.

- Pause-DR

  This state allows the shifting of the data register selected by the instruction register to be temporarily suspended. Both the instruction register and the data register retain their current states.

  When the controller is in this state, then it remains in the Pause-DR state if JTMS is held low, or moves to the Exit2-DR state if JTMS is held high.

- Exit 2-DR

  This is a temporary controller state.

  When the controller is in this state, then it returns to the Shift-DR state if JTMS is held low, or moves on to the Update-DR state if JTMS is held high.

- Update-DR

  In this state, data is latched, on the falling edge of JTCK, onto the parallel outputs of the data registers from the shift register path. The data held at the parallel output does not change while data is shifted in the associated shift register path.

  When the controller is in this state, it moves to either the Run-Test/Idle state if JTMS is held low, or the Select-DR-Scan state if JTMS is held high.

- Capture-IR

  In this state, data is parallel-loaded into the instruction register. The two least significant bits are assigned the values "01". The higher-order bits of the instruction register can receive any design specific values. The Capture-IR state is used for testing the instruction register. Faults in the instruction register, if any exists, may be detected by shifting out the data loaded in it.

  When the controller is in this state, it moves to either the Shift-IR state if JTMS is low, or the Exit1-IR state if JTMS is high.

- Shift-IR

  In this state, the instruction register is connected between JTDI and JTDO and shifts the captured data toward its serial output on the rising edge of JTCK.

  When the controller is in this state, it remains in the Shift-IR state if JTMS is low, or moves to the Exit1-IR state if JTMS is high.

- Exit 1-IR
  This is a temporary controller state.

  When the controller is in this state, then it moves to either the Pause-IR state if JTMS is held low, or the Update-IR state if JTMS is held high.

- Pause-IR
  This state allows the shifting of the instruction register to be temporarily suspended. Both the instruction register and the data register retain their current states.

  When the controller is in this state, it remains in the Pause-IR state if JTMS is held low, or moves to the Exit2-IR state if JTMS is held high.

- Exit 2-IR
  This is a temporary controller state.

  When the controller is in this state, it moves to either the Shift-IR state if JTMS is held low, or the Update-IR state if JTMS is held high.

- Update-IR
  This state allows the instruction previously shifted into the instruction register to be output in parallel on the rising edge of JTCK. Then it becomes the current instruction, setting a new operational mode.

  When the controller is in this state, it moves to either the Run-Test/Idle state if JTMS is low, or the Select-DR-Scan state if JTMS is high.

Tables 7.3.2 shows the boundary scan order of the processor signals.

Table 14.3.2 TX4951B JTAG Boundary-Scan Ordering

[JTDI]	1: SysAD[4]	2: SysAD[5]	3: SysAD[6]	4: SysAD[7]	5: SysAD[8]
6: SysAD[9]	7: SysAD[10]	8: SysAD[11]	9: SysAD[12]	10: SysAD[13]	11: SysAD[14]
12: SysAD[15]	13: RdRdy	14: WrRdy*	15: ValidIn*	16: ValidOut*	17: Release*
18: ExtRqst*	19: TIntDis	20: HALTDOZE	21: Int[0]*	22: Int[1]*	23: Int[2]*
24: DivMode[0]	25: DivMode[1]	26: Endian	27: SysCmd[0]	28: SysCmd[1]	29: SysCmd[2]
30: SysCmd[3]	31: SysCmd[4]	32: SysCmd[5]	33: SysCmd[6]	34: SysCmd[7]	35: SysCmd[8]
36: SysAD[16]	37: SysAD[17]	38: SysAD[18]	39: SysAD[19]	40: SysAD[20]	41: SysAD[21]
42: SysAD[22]	43: SysAD[23]	44: SysAD[24]	45: SysAD[25]	46: SysAD[26]	47: SysAD[27]
48: SysAD[28]	49: SysAD[29]	50: SysAD[30]	51: SysAD[31]	52: SysAD[0]	53: SysAD[1]
54: SysAD[2]	55: SysAD[3]	56: MODE43*	[JTDO]		

14.4 Instructions for JTAG

This section defines the instructions supplied and the operations that occur in response to those instructions.

14.4.1 The EXTEST Instruction

This instruction is used for external interconnect test, and targets the boundary scan register between JTDI and JTDO. The EXTEST instruction permits BSR cells at output pins to shift out test patterns in the Update-DR state and those at input pins to capture test results in the Capture-DR state.

Typically, before EXTEST is executed, the initialization pattern is first shifted into the boundary scan register using the SAMPLE/PRELOAD instruction. In the Update-DR state, the boundary scan register loaded with the initialization pattern causes known data to be driven immediately from the IC onto its external interconnects. This eliminates the possibility that bus conflicts damage the IC outputs. The flow of data through the boundary scan register while the EXTEST instruction is selected is shown in Figure 14.4.1, which follows:
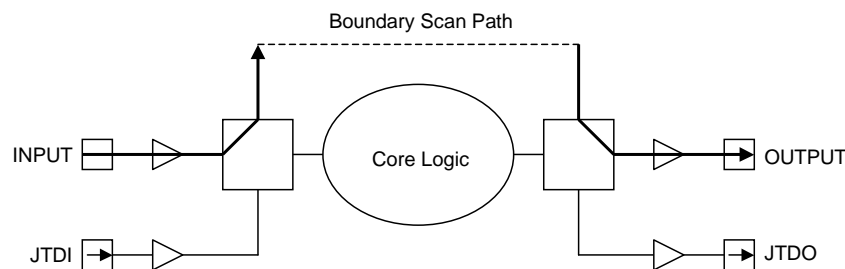
Figure 14.4.1 Test Data Flow While the EXTEST Instruction is Selected

The following steps describe the basic test algorithm of an external interconnect test.

1. Initialize the TAP controller to the Test-Logic-Reset state.

2. Load the instruction register with SAMPLE/PRELOAD. This causes the boundary scan register to be connected between JTDI and JTDO.

3. Initialize the boundary scan register by shifting in determinate data.

4. Then, load the initial test data into the boundary scan register.

5. Load the instruction register with EXTEST.

6. Capture the data applied to the input pin into the boundary scan register.

7. Shift out the captured data while simultaneously shifting in the next test pattern.

8. Read out the data in the boundary scan register onto the output pin.

Steps 6 to 8 are repeated for each test pattern.

14.4.2   The SAMPLE/PRELOAD Instruction

This instruction targets the boundary scan register between JTDI and JTDO. As the instruction's name implies, two functions are performed through use of the SAMPLE/ PRELOAD instruction.

- SAMPLE allows the input and output pads of an IC to be monitored. While it does so, it does not disconnect the system logic from the IC pins. The SAMPLE function occurs in the Capture-DR controller state. An example application of SAMPLE is to take a snapshot of the activity of the IC's I/O pins so as to verify the interaction between ICs during normal functional operation. The flow of data for the SAMPLE phase of the SAMPLE/PRELOAD instruction is shown in Figure 14.4.2.
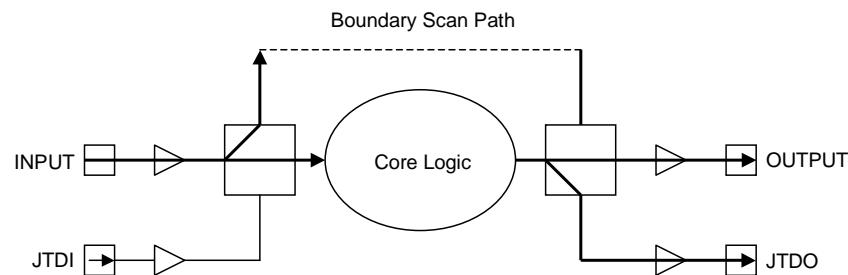
Figure 14.4.2 Test Data Flow While SAMPLE is Selected

- PRELOAD allows the boundary scan register to be initialized before another instruction is selected. For example, prior to selection of the EXTEST instruction, initialization data is shifted into the boundary scan register using PRELOAD as described in the previous subsection. PRELOAD permits shifting of the boundary scan register without interfering with the normal operation of the system logic. The flow of data for the PRELOAD phase of the SAMPLE/PRELOAD instruction is shown in Figure 14.4.3.
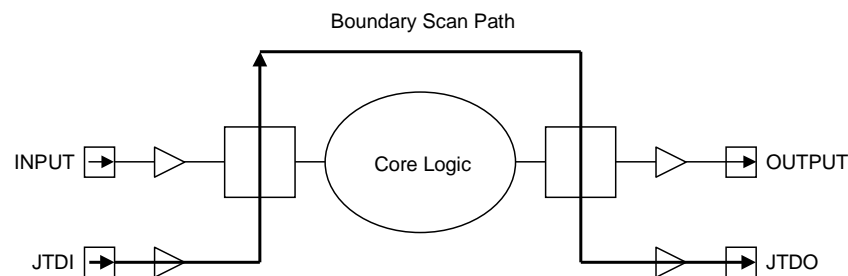
Figure 14.4.3 Test Data Flow While PRELOAD is Selected

14.4.3 The BYPASS Instruction

This instruction targets the bypass register between JTDI and JTDO. The bypass register provides a minimum length serial path through the IC [or between JTDI and JTDO) when the IC is not required for the current test. The BYPASS instruction does not cause interference to the normal operation of the on-chip system logic. The flow of data through the bypass register while the BYPASS instruction is selected is shown in Figure 14.4.4.
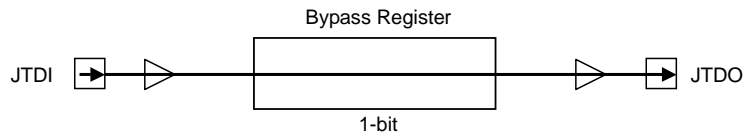
Figure 14.4.4 Test Data Flow While the Bypass Instruction is Selected

14.4.4 The IDCODE Instruction

This instruction targets the device identification register between JTDI and JTDO to identify manufacturer identity, part number, and version number for the part.

14.5 Note

This section discribes details of JTAG boundary-scan operation that are specific to the processor.

- The **MasterClock**, and **DivMode2** signal pads do not supposrt JTAG.

- When performing a JTAG operation, be sure to run the **MasterClock** before and after a reset operation to properly release the processor reset.

- Reset for JTAG

  (1) JTAG circuit is initialized by **TRST*** assertion. And then deassert **TRST***.

  (2) At input to **JTMS** = 1 and asserted for more 5 JTCK cycles.

15.　CPU Instruction Set Summary

15.1 Introduction

　　Each instruction is 32 bits long. These instructions are upward compatible with the MIPS I, II and III instruction set architecture.

15.2 Instruction Format

　　There are three instruction formats: Immediate (I-type), Jump (J-type) and Register (R-type), as shown in Figure 15.2.1. Having just three instruction formats simplifies instruction decoding. If more complex functions or addressing modes are required, they can be produced with the compiler using combinations of the instructions.

Immediate (I-type)

31　　　26	25　　21	20　　16	15　　　　　　　　　　　　　　0
op	rs	rt	immediate

Jump (J-type)

31　　　26	25　　　　　　　　　　　　　　　　　0
op	target

Register (R-type)

31　　　26	25　　21	20　　16	15　　11	10　　6	5　　0
op	rs	rt	rd	sa	funct

op	Operation code (6 bits)
rs	Source register (5 bits)
rt	Target (source or destination) register, or branch condition (5 bits)
rd	Destination register (5 bits)
immediate	Immediate, branch displacement, address displacement (16 bits)
target	Branch target address (26 bits)
sa	Shift amount (5 bits)
funct	Function (6 bits)

Figure 15.2.1 Instruction formats and subfield mnemonics

15.3 Instruction Set Overview

15.3.1 Load and Store Instructions (Table 15.3.1)

Load and Store instructions move data between memory and general purpose registers, and are all I-type instructions. The only directly supported addressing mode is "base register plus 16-bit signed immediate offset".

Table 15.3.1 CPU Instruction Set: Load and Store Instructions

Instruction	Description	Note
LB	Load Byte	MIPS I
LBU	Load Byte Unsigned	MIPS I
LH	Load Halfword	MIPS I
LHU	Load Halfword Unsigned	MIPS I
LW	Load Word	MIPS I
LWL	Load Word Left	MIPS I
LWR	Load Word Right	MIPS I
SB	Store Byte	MIPS I
SH	Store Halfword	MIPS I
SW	Store Word	MIPS I
SWL	Store Word Left	MIPS I
SWR	Store Word Right	MIPS I
LD	Load Doubleword	MIPS III
LDL	Load Doubleword Left	MIPS III
LDR	Load Doubleword Right	MIPS III
LL	Load Linked	MIPS II
LLD	Load Linked Doubleword	MIPS III
LWU	Load Word Unsigned	MIPS III
SC	Store Conditional	MIPS II
SCD	Store Conditional Doubleword	MIPS III
SD	Store Doubleword	MIPS III
SDL	Store Doubleword Left	MIPS III
SDR	Store Doubleword Right	MIPS III
SYNC	Sync	MIPS II

15.3.2 Computational Instructions (Table 15.3.2)

Computational instructions perform arithmetic, logical or shift operations on values in registers. This instruction format can be R-type or I-type. With R-type instructions, the one/two operands and the result are register values. With I-type instructions, one of the operands is 16-bit immediate data. Computational instructions can be classified as follows.

- ALU immediate
- Three-operand register-type
- Shift
- Multiply/Divide

Table 15.3.2 CPU Instruction Set: Computational Instructions

Instruction	Description	Note
	(ALU Immediate)	
ADDI	Add Immediate	MIPS I
ADDIU	Add Immediate Unsigned	MIPS I
SLTI	Set on Less Than Immediate	MIPS I
SLTIU	Set on Less Than Immediate Unsigned	MIPS I
ANDI	AND Immediate	MIPS I
ORI	OR Immediate	MIPS I
XORI	Exclusive OR Immediate	MIPS I
LUI	Load Upper Immediate	MIPS I
DADDI	Doubleword Add Immediate	MIPS III
DADDIU	Doubleword Add Immediate Unsigned	MIPS III
	(ALU 3-Operand, register type)	
ADD	Add	MIPS I
ADDU	Add Unsigned	MIPS I
SUB	Subtract	MIPS I
SUBU	Subtract Unsigned	MIPS I
SLT	Set on Less Than	MIPS I
SLTU	Set on Less Than Unsigned	MIPS I
AND	AND	MIPS I
OR	OR	MIPS I
XOR	Exclusive OR	MIPS I
NOR	NOR	MIPS I
DADD	Doubleword Add	MIPS III
DADDU	Doubleword Add Unsigned	MIPS III
DSUB	Doubleword Subtract	MIPS III
DSUBU	Doubleword Subtract Unsigned	MIPS III
	(Shift)	
SLL	Shift Left Logical	MIPS I
SRL	Shift Right Logical	MIPS I
SRA	Shift Right Arithmetic	MIPS I
SLLV	Shift Left Logical Variable	MIPS I
SRLV	Shift Right Logical Variable	MIPS I
SRAV	Shift Right Arithmetic Variable	MIPS I
DSLL	Doubleword Shift Left Logical	MIPS III
DSRL	Doubleword Shift Right Logical	MIPS III
DSRA	Doubleword Shift Right Arithmetic	MIPS III
DSLLV	Doubleword Shift Left Logical Variable	MIPS III
DSRLV	Doubleword Shift Right Logical Variable	MIPS III

Instruction	Description	Note
DSRAV	Doubleword Shift Right Arithmetic Variable	MIPS III
DSLL32	Doubleword Shift Left Logical +32	MIPS III
DSRL32	Doubleword Shift Right Logical +32	MIPS III
DSRA32	Doubleword Shift Right Arithmetic +32	MIPS III
	(Multiply and Divide)	
MULT	Multiply	MIPS I
MULTU	Multiply Unsigned	MIPS I
DIV	Divide	MIPS I
DIVU	Divide Unsigned	MIPS I
MFHI	Move From HI	MIPS I
MTHI	Move To HI	MIPS I
MFLO	Move From LO	MIPS I
MTLO	Move To LO	MIPS I
DMULT	Doubleword Multiply	MIPS III
DMULTU	Doubleword Multiply Unsigned	MIPS III
DDIV	Doubleword Divide	MIPS III
DDIVU	Doubleword Divide Unsigned	MIPS III

15.3.3 Jump and Branch Instructions (Table 15.3.3)

Jump and branch instructions change the control flow of a program. All jump and branch instructions occur with a delay of one instruction: that is, the instruction immediately following the jump or branch (this is known as the instruction in the delay slot) always executes while the target instruction is being fetched from storage. Branch-likely instructions are used for static branch prediction. The instruction in the delay slot is executed only when the branch is taken; the instruction in the delay slot is nullified if the branch is not taken.

Table 15.3.3 CPU Instruction Set: Jump and Branch Instructions

Instruction	Description	Note
J	Jump	MIPS I
JAL	Jump And Link	MIPS I
JR	Jump Register	MIPS I
JALR	Jump And Link Register	MIPS I
BEQ	Branch on Equal	MIPS I
BNE	Branch on Not Equal	MIPS I
BLEZ	Branch on Less Than or Equal to Zero	MIPS I
BGTZ	Branch on Greater Than Zero	MIPS I
BLTZ	Branch on Less Than Zero	MIPS I
BGEZ	Branch on Greater than or Equal to Zero	MIPS I
BLTZAL	Branch on Less Than Zero And Link	MIPS I
BGEZAL	Branch on Greater than or Equal to Zero And Link	MIPS I
BEQL	Branch on Equal Likely	MIPS II
BNEL	Branch on Not Equal Likely	MIPS II
BLEZL	Branch on Less Than or Equal to Zero Likely	MIPS II
BGTZL	Branch on Greater Than Zero Likely	MIPS II
BLTZL	Branch on Less Than Zero Likely	MIPS II
BGEZL	Branch on Greater Than or Equal to Zero Likely	MIPS II
BLTZALL	Branch on Less Than Zero And Link Likely	MIPS II
BGEZALL	Branch on Greater Than or Equal to Zero And Link Likely	MIPS II

15.3.4 Special Instructions (Table 15.3.4)

There are special instructions used for software trap. The instruction format is R-type for all two.

Table 15.3.4 CPU Instruction Set: Special Instructions

Instruction	Description	Note
SYSCALL	System Call	MIPS I
BREAK	Break	MIPS I

15.3.5 Exception Instructions (Table 15.3.5)

These instructions (R-type or I-type) cause a branch to the general exception handling vector based upon the result of a comparison.

Table 15.3.5 CPU Instruction Set: Exception Instructions

Instruction	Description	Note
TGE	Trap if Greater Than or Equal	MIPS II
TGEU	Trap if Greater Than or Equal Unsigned	MIPS II
TLT	Trap if Less Than	MIPS II
TLTU	Trap if Less Than Unsigned	MIPS II
TEQ	Trap if Equal	MIPS II
TNE	Trap if Not Equal	MIPS II
TGEI	Trap if Greater Than or Equal Immediate	MIPS II
TGEIU	Trap if Greater Than or Equal Immediate Unsigned	MIPS II
TLTI	Trap if Less Than Immediate	MIPS II
TLTIU	Trap if Less Than Immediate Unsigned	MIPS II
TEQI	Trap if Equal Immediate	MIPS II
TNEI	Trap if Not Equal Immediate	MIPS II

15.3.6 Coprocessor Instructions (Table 15.3.6)

Coprocessor instructions invoke coprocessor operations. The format of these instructions depends on which coprocessor is used.

Table 15.3.6 CPU Instruction Set: Coprocessor Instructions

Instruction	Description	Note
LWCz	Load Word to Coprocessor z (z = 1,2)	MIPS I
SWCz	Store Word from Coprocessor z (z = 1,2)	MIPS I
MTCz	Move To Coprocessor z (z = 1,2)	MIPS I
MFCz	Move From Coprocessor z (z = 1,2)	MIPS I
CTCz	Move Control To Coprocessor z (z = 1,2)	MIPS I
CFCz	Move Control From Coprocessor z (z = 1,2)	MIPS I
COPz	Coprocessor Operation z (z = 1,2)	MIPS I
BCzT	Branch on Coprocessor z True (z = 0,1,2)	MIPS I
BCzF	Branch on Coprocessor z False (z = 0,1,2)	MIPS I
BCzTL	Branch on Coprocessor z True Likely (z = 0,1,2)	MIPS II
BCzFL	Branch on Coprocessor z False Likely (z = 0,1,2)	MIPS II
LDCz	Load Double Coprocessor z (z = 1,2)	MIPS III
SDCz	Store Double Coprocessor z (z = 1,2)	MIPS III
DMTCz	Doubleword Move To Coprocessor z (z = 1,2)	MIPS III
DMFCz	Doubleword Move From Coprocessor z (z = 1,2)	MIPS III

15.3.7 CP0 Instructions (Table 15.3.7)

Coprocessor 0 instructions are used for operations involving the system control coprocessor (CP0) registers, processor memory management and exception handling.

Table 15.3.7 Instruction Set: CP0 Instructions

Instruction	Description	Note
MTC0	Move To CP0	MIPS I
MFC0	Move From CP0	MIPS I
DMTC0	Doubleword Move To CP0	MIPS III
DMFC0	Doubleword Move From CP0	MIPS III
TLBR	Read Indexed TLB Entry	
TLBWI	Write Indexed TLB Entry	
TLBWR	Write Random TLB Entry	
TLBP	Probe TLB for Matching Entry	
CACHE	Cache	MIPS III
ERET	Exception Return	MIPS III
WAIT	Enter power management mode	

15.3.8 Multiply and Divide Instructions (Table 15.3.8)

Table 15.3.8 Extensions to the ISA: Multiply and Divide Instructions

Instruction	Description	Note
MULT	Multiply (3-operand)	
MULTU	Multiply Unsigned (3-operand)	
DMULT	Doubleword Multiply (3-operand)	
DMULTU	Doubleword Multiply Unsigned (3-operand)	
MADD	Multiply and ADD (3-operand)	
MADDU	Multiply and ADD Unsigned (3-operand)	

15.3.9 Debug Instructions (Table 15.3.9)

Table 15.3.9 Extensions to the ISA: Debug Instructions

Instruction	Description	Note
CTC0	Move Control To Coprocessor 0	
CFC0	Move Control From Coprocessor 0	
SDBBP	Software Debug Breakpoint	
DERET	Debug Exception Return	

15.3.10   Other Instructions (Table 15.3.10)

Table 15.3.10 Other Instructions

Instruction	Description	Note
PREF	Prefetch	

15.4 Instruction Execution Cycles

Because the TX49 employs the high-speed Multiply and Add Calculator (MAC), multiply instructions, such as MULT, MULTU, DMULT and DMULTU are executed faster. And, TX49 is improved the execution of divide instructions, too.

Instruction	Latency (2op/3op)	Repeat (2op/3op)
MULT 2/3 operand	4/4	1/3
MADD 2/3 operand	4/4	1/3
DMULT 2/3 operand	7/7	6/6
DIV	37	36
DDIV	69	68

15.5　Defining Access Types

Access type indicates the size of a TX4951B data item to be loaded or stored, set by the load or store instruction opcode.

Regardless of access type or byte ordering (endianness), the address given specifies the low-order byte in the addressed field. For a big-endian configuration, the low-order byte is the most-significant byte; for a little-endian configuration, the low-order byte is the least-significant byte.

The access type, together with the three low-order bits of the address, define the bytes accessed within the addressed doubleword (shown in Figure 15.5.1). Only the combinations shown in Figure 15.5.1 are permissible; other combinations cause address error exceptions.

Access Type Mnemonic (Value)	Low-Order Address Bits 2	1	0	Bytes Accessed — Big Endian (63--------31--------0) Byte								Little Endian (63--------31--------0) Byte							
Doubleword (7)	0	0	0	0	1	2	3	4	5	6	7	7	6	5	4	3	2	1	0
Septibyte (6)	0	0	0	0	1	2	3	4	5	6			6	5	4	3	2	1	0
	0	0	1		1	2	3	4	5	6	7	7	6	5	4	3	2	1	
Sextibyte (5)	0	0	0	0	1	2	3	4	5					5	4	3	2	1	0
	0	1	0			2	3	4	5	6	7	7	6	5	4	3	2		
Quintibyte (4)	0	0	0	0	1	2	3	4							4	3	2	1	0
	0	1	1				3	4	5	6	7	7	6	5	4	3			
Word (3)	0	0	0	0	1	2	3									3	2	1	0
	1	0	0					4	5	6	7	7	6	5	4				
Triplebyte (2)	0	0	0	0	1	2											2	1	0
	0	0	1		1	2	3									3	2	1	
	1	0	0					4	5	6			6	5	4				
	1	0	1						5	6	7	7	6	5					
Halfword (1)	0	0	0	0	1													1	0
	0	1	0			2	3									3	2		
	1	0	0					4	5				5	4					
	1	1	0							6	7	7	6						
Byte (0)	0	0	0	0															0
	0	0	1		1													1	
	0	1	0			2											2		
	0	1	1				3									3			
	1	0	0					4							4				
	1	0	1						5					5					
	1	1	0							6			6						
	1	1	1								7	7							

Figure 15.5.1　Byte Access within a Doubleword

15.6 Bit Encoding of CPU Instruction OPcodes

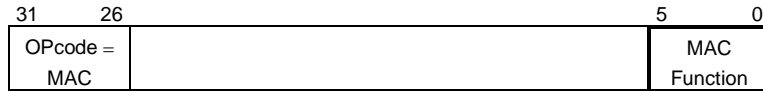The Table 15.6.1 shows the bit codes for all TX4951B CPU instructions (ISA and extended ISA)

Table 15.6.1 CPU Operation Code Bit Encoding

OPcode

31	26		0
OPcode			

[28:26]

[31:29]	0	1	2	3	4	5	6	7
0	SPECIA λ	REGIMM λ	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	COP0 α	COP1 α	COP2 α	COP3 α θ	BEQL	BNEL	BLEZL	BGTZL
3	DADDI ε	DADDIU ε	LDL ε	LDR ε	MAC λ	*	*	*
4	LB	LH	LWL	LW	LBU	LHU	LWR	LWU ε
5	SB	SH	SWL	SW	SDL ε	SDR ε	SWR	CACHE
6	LL	LWC1 α	LWC2 α	PREF	LLD ε	LDC1 α	LDC2 α	LD ε
7	SC	SWC1 α	SWC2 α	*	SCD ε	SDC1 α	SDC2 α	SD ε

SPECIAL Function

31	26		5	0
OPcode = SPECIAL			SPECIAL Function	

[2:0]

[5:3]	0	1	2	3	4	5	6	7
0	SLL	*	SRL	SRA	SLLV	*	SRLV	SRAV
1	JR	JALR	*	*	SYSCALL	BREAK	SDBBP	SYNC
2	MFHI	MTHI	MFLO	MTLO	DSLLV ε	*	DSRLV ε	DSRAV ε
3	MULT	MULTU	DIV	DIVU	DMULT ε	DMULTε	DDIV ε	DDIVU ε
4	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	*	*	SLT	SLTU	DADD ε	DADDU ε	DSUB ε	DSUBU ε
6	TGE	TGEU	TLT	TLTU	TEQ	*	TNE	*
7	DSLL ε	*	DSRL ε	DSRA ε	DSLL32 ε	*	DSRL32 ε	DSRA32 ε

REGIMM rt

31	26	20	16	0
OPcode = REGIMM		REGIMM rt		

[18:16]

[20:19]	0	1	2	3	4	5	6	7
0	BLTZ	BGEZ	BLTZL	BGEZL	*	*	*	*
1	TGEI	TGEIU	TLTI	TLTIU	TEQI	*	TNEI	*
2	BLTZAL	BGEZAL	BLTZALL	BGEZALL	*	*	*	*
3	*	*	*	*	*	*	*	*

COPz rs

31	26	25	21	0
OPcode = COPz		COPz rs		

[23:21]

[25:24]	0	1	2	3	4	5	6	7
0	MF	DMF ε	CF	γ	MT	DMT ε	CT	γ
1	BC	γ	γ	γ	γ	γ	γ	γ
2	CO							
3								

COPz rt

31	26	20	16	0
OPcode = COPz		COPz rt		

[18:16]

[20:19]	0	1	2	3	4	5	6	7
0	BCF	BCT	BCFL	BCTL	γ	γ	γ	γ
1	γ	γ	γ	γ	γ	γ	γ	γ
2	γ	γ	γ	γ	γ	γ	γ	γ
3	γ	γ	γ	γ	γ	γ	γ	γ

COP0 Function

31	26	5	0
OPcode = COP0		COP0 Function	

[2:0]

[5:3]	0	1	2	3	4	5	6	7
0	φ	TLBR	TLBWI	φ	φ	φ	TLBWR	φ
1	TLBP	φ	φ	φ	φ	φ	φ	φ
2	φ	φ	φ	φ	φ	φ	φ	φ
3	ERET	φ	φ	φ	φ	φ	φ	DERET
4	WAIT	φ	φ	φ	φ	φ	φ	φ
5	φ	φ	φ	φ	φ	φ	φ	φ
6	φ	φ	φ	φ	φ	φ	φ	φ
7	φ	φ	φ	φ	φ	φ	φ	φ

MAC Function

31 26		5 0
OPcode = MAC		MAC Function

	[2:0]							
[5:3]	0	1	2	3	4	5	6	7
0	MADD	MADDU	γ	γ	γ	γ	γ	γ
1	γ	γ	γ	γ	γ	γ	γ	γ
2	γ	γ	γ	γ	γ	γ	γ	γ
3	γ	γ	γ	γ	γ	γ	γ	γ
4	γ	γ	γ	γ	γ	γ	γ	γ
5	γ	γ	γ	γ	γ	γ	γ	γ
6	γ	γ	γ	γ	γ	γ	γ	γ
7	γ	γ	γ	γ	γ	γ	γ	γ

Key :

∗:   This opcode is reserved for future use. An attempt to execute it causes a Reserved Instruction exception.

γ: This opcode is reserved for future use. An attempt to execute it causes a Reserved Instruction exception.

λ: This opecode indicates an instruction class. The instruction word must be further decoded by examining additional tables that show the values for another instruction field.

α: This opcode is a coprocessor operation, not a CPU operation. If the processor state does not allow access to the specified coprocessor, the instruction causes a Coprocessor Unusable exception. It is included in the table because it uses a primary opecode in the instruction encodeing map.

φ: This opcode is reserved for future use, but does not cause a Reserved Instruction exception in TX4951B implementations. It is treated as "NOP".

θ: This opcode is valid when BC is only selected in COPz rs; In other case, it causes a Reserved Instruction exception .

ε: This opcode is valid when the processor is operating either in the Kernel mode or in the 64-bit non-Kernel (User or Supervisor) mode; In other case, it causes a Reserved Instruction exception .

16. Electrical Characteristics

ESD Precautions: For handling precautions, see Section 1.1, Electrostatic Discharge (ESD), in the chapter on General Safety Precautions and Usage Considerations.

16.1 Electrical Characteristics

16.1.1 Absolute Maximum Ratings

V_{SS} = 0 V (GND)

Parameter	Symbol	Ratings	Unit
Supply Voltage (I/O)	$V_{CC}IOMax$	−0.3 to 3.9	V
Supply Voltage (Core)	$V_{CC}IntMax$	−0.3 to 3.0	V
Input Voltage	V_{IN}	−0.3 to $V_{CC}IO + 0.3$	V
Storage Temperature	T_{STG}	−40 to + 125	°C

Note: The absolute maximum ratings are rated values that must not be exceeded during operation,
even for an instant. Any one of the ratings must not be exceeded. If any absolute maximum
rating is exceeded, a device may break down or its performance may be degraded, causing it to
catch fire or explode resulting in injury to the user. Thus, when designing products which include
this device, ensure that no absolute maximum rating value will ever be exceeded.

(∗1) V_{IN} Min = 1.5 V for pulse width less than 10 ns.

(∗2)   Even $V_{CC}IO$ + 0.3 shall not exceed the $V_{CC}IOMax$ rating.

16.1.2 Recommended Operating Conditions

V_{SS} = 0 V (GND)

Parameter	Symbol	Conditions	Min	Max	Unit
Supply Voltage (I/O)	$V_{CC}IO$	I/O = 3.3 V	3.0	3.6	V
		I/O = 2.5 V	2.3	2.7	V
Supply Voltage (Core)	$V_{CC}Int$		1.4	1.6	V
Operating Case Temperature	T_C		−20	+85	°C

Note: The recommended operating conditions for a device are those under which it can be guaranteed
that the device will operate as specified. If the device is used under operating conditions other
than the recommended operating conditions (supply voltage, operating temperature range,
specified AC and DC values, etc.), malfunction may occur. Thus, when designing products which
include this device, ensure that the recommended operating conditions for the device are always
adhered to.

16.1.3   DC Characteristics

16.1.3.1 DC Characteristics

$T_C = -20°C$ to $85°C$, $V_{CC}Int = 1.5\ V \pm 0.1\ V$, $V_{CC}IO = 3.3\ V \pm 0.2\ V$ or $2.5\ V \pm 0.2\ V$

Parameter	Symbol	Conditions	Min	Max	Units
Output High Voltage	V_{OH}	$I_{OH} = -4$ mA (4-mA buffer)	$V_{CC}IO - 0.6$	—	V
Output Low Voltage	V_{OL}	$I_{OL} = 4$ mA (4-mA buffer)	—	0.4	V
Input High Voltage	V_{IH}	Except for MasterClock input	2	$V_{CC}IO + 0.3$	V
	V_{IHC}	Applies to MasterClock	$0.8V_{CC}IO$	$V_{CC}IO + 0.3$	
Input Low Voltage	V_{IL}	Except for MasterClock input when I/O = 3.3 V	-0.5 [*1]	0.8	V
		Except for MasterClock input when I/O = 2.5 V	-0.5 [*1]	0.6	
	V_{ILC}	Applies to MasterClock	-0.5 [*1]	$0.2V_{CC}IO$	
Input Leakage	I_{LI}	Applies to pins except (*3)	—	±10	μA
Input Leakage (with Pull-up) [*2]	R_{inu}		-70	-10	μA
Input Leakage (with Pull-down) [*3]	R_{ind}		10	70	μA
Output Leakage	I_{LO}		—	±20	μA
Input Capacitance	C_{IN}		—	10	pF

(*1) V_{IL} Min = -1.5 V for pulse width less than 10 ns.

(*2)   Applies to the following input pins that have an internal pull-up resistor. Int[3:0]*, NMI*, STMS, JTCK, JTDI, ColdReset* (If they need be pulled down, use a resistor with ≤ 20 kΩ.)

(*3)   Applies to the following input pins that have an internal pull-down resistor. TRST*, RdRdy*
(If they need be pulled up, use a resistor with ≤ 20 kΩ.)

16.1.3.2 Operating Current

$T_C = -20°C$ to $85°C$, $V_{CC}Int = 1.5\ V \pm 0.1\ V$, $V_{CC}IO = 3.3\ V \pm 0.2\ V$ or $2.5\ V \pm 0.2\ V$

Parameter	Symbol	Conditions	Typ.	Max	Units
Operating Current 1 (Core Power Supply) (Normal Operating Mode)	I_{CCInt1}	CPUCLK = 200 MHz	250	350	mA
Operating Current 1 (Core Power Supply) (Drystone 2.1)	I_{CCInt2}	CPUCLK = 200 MHz	200	250	mA
Operating Current 1 (Core Power Supply) (MasterClock Stopped)	I_{CCInt3}	CPUCLK = 200 MHz	30	100	mA
Operating Current 1 (I/O Power Supply)	I_{CCInt4}	MasterClock = 0 MHz CPUCLK = 0 MHz	20	60	mA
Operating current (Power supply for I/O pin)	I_{CCIO}	MasterClock = 100 Hz $V_{CC}IO = 3.6$ V Load = 25 pF	60	80	mA
		MasterClock = 100 MHz $V_{CC}IO = 2.7$ V Load = 25 pF	30	40	mA

16.1.4   AC Characteristics

16.1.4.1 Clock Timing

$T_C = -20°C$ to $85°C$, $V_{CC}Int = 1.5\ V \pm 0.1\ V$, $V_{CC}IO = 3.3\ V \pm 0.2\ V$ or $2.5\ V \pm 0.2\ V$

Parameter	Symbol	Conditions	Min	Max	Units
MasterClock High Width	t_{MCH}	Transition $\leq$ 5 ns	3	—	ns
MasterClock Low Width	t_{MCL}	Transition $\leq$ 5 ns	3	—	ns
MasterClock Frequency [*1]	f_{MCK}		33	100	MHz
Internal Operation Frequency			120	200	MHz
MasterClock Period	t_{MCP}		10	33	ns
MasterClock Rise Time	t_{MCR}			2	ns
MasterClock Fall Time	t_{MCF}			2	ns

[*1]   Operation of the TX4951B is only guaranteed with the Phase Lock Loop enabled.

[*2]   All output timings assume a 25-pF capacitive load.

16.1.4.2 System Interface

$T_C = -20°C$ to $85°C$, $V_{CC}Int = 1.5\ V \pm 0.1\ V$, $V_{CC}IO = 3.3\ V \pm 0.3\ V$

Parameter	Symbol	Conditions	Min	Max	Units
Data Output [*1, 2, 3]	t_{DO}		1.0[*4]	4.5	ns
Data Setup [*3]	t_{DS}		2.5		ns
Data Hold [*3]	t_{DH}		1.0		ns

$T_C = -20°C$ to $85°C$, $V_{CC}Int = 1.5\ V \pm 0.1\ V$, $V_{CC}IO = 2.5\ V \pm 0.2\ V$

Parameter	Symbol	Conditions	Min	Max	Units
Data Output [*1, 2, 3]	t_{DO}		1.0[*4]	5.0	ns
Data Setup [*3]	t_{DS}		3.0		ns
Data Hold [*3]	t_{DH}		1.0		ns

[*1]   Timings are measured from 1.5 V of MasterClock to 1.5 V of each signal.

[*2]   Capacitive load for all output timings is 25 pF.

[*3]   Applies to all system interface signals.

[*4]   Offers a guarantee of design.
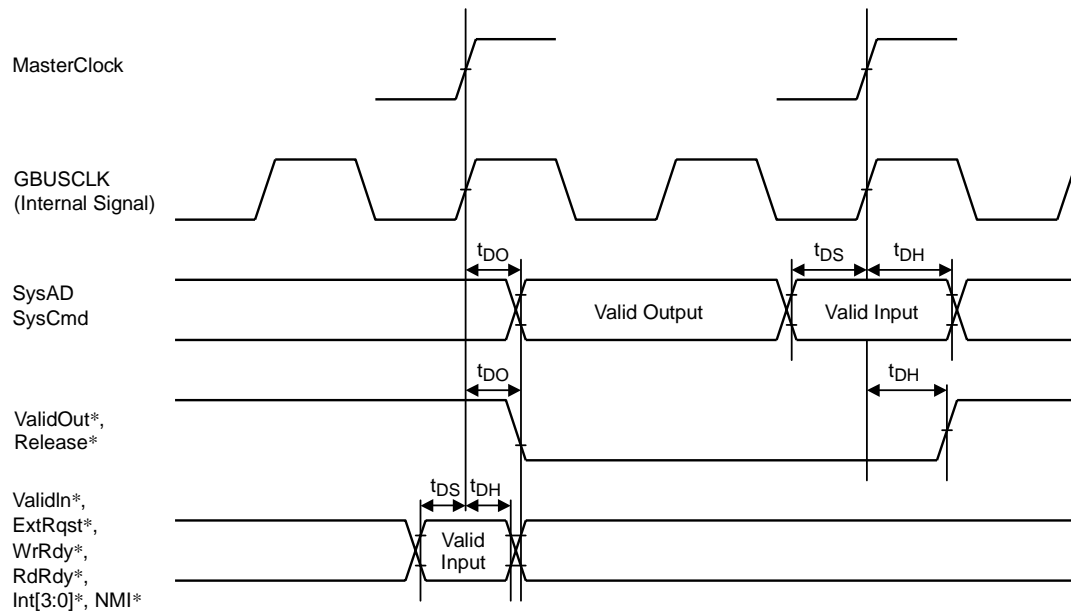
16.1.5   Timing Diagrams
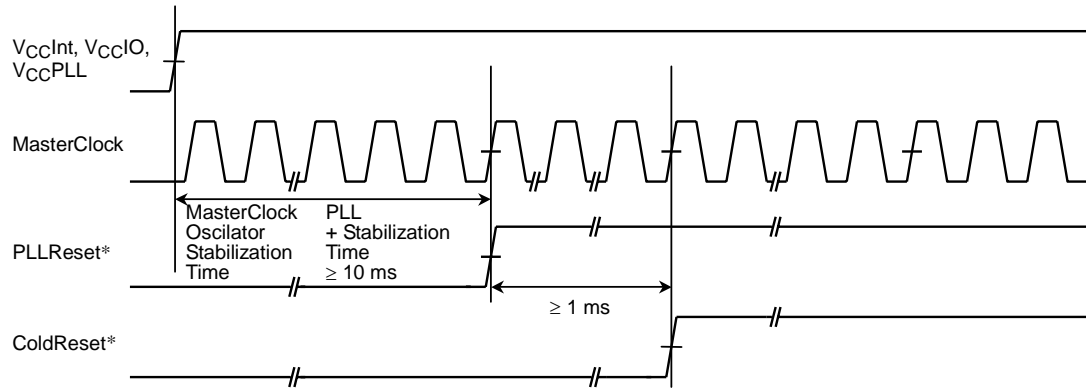
16.1.5.1 Clock Timing
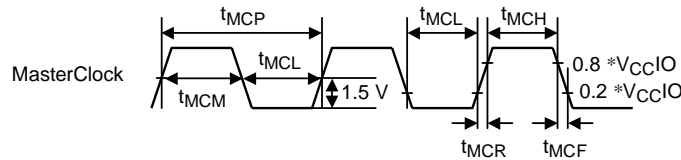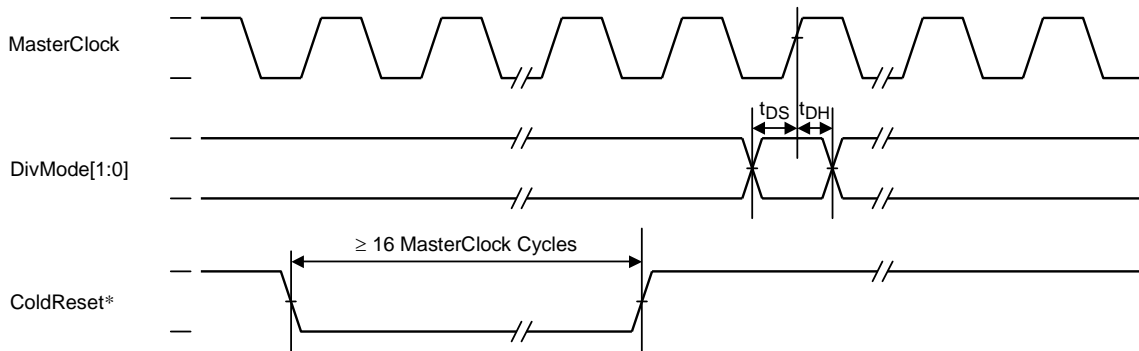
16.1.5.2 Clock Relationships
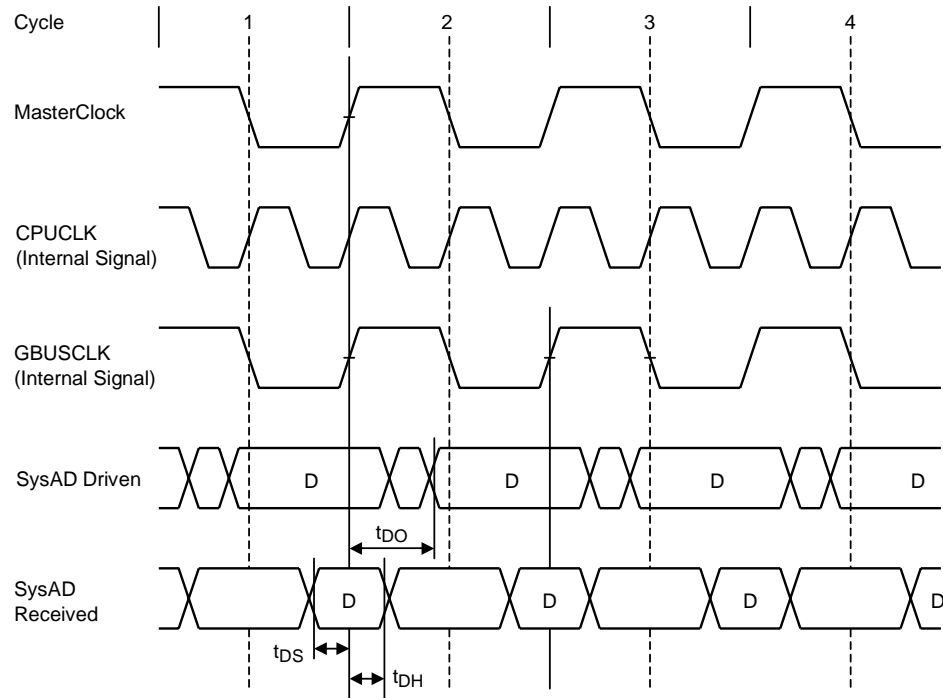
16.1.5.3 System Interface Timing

16.1.5.4 Power-On Reset Timing
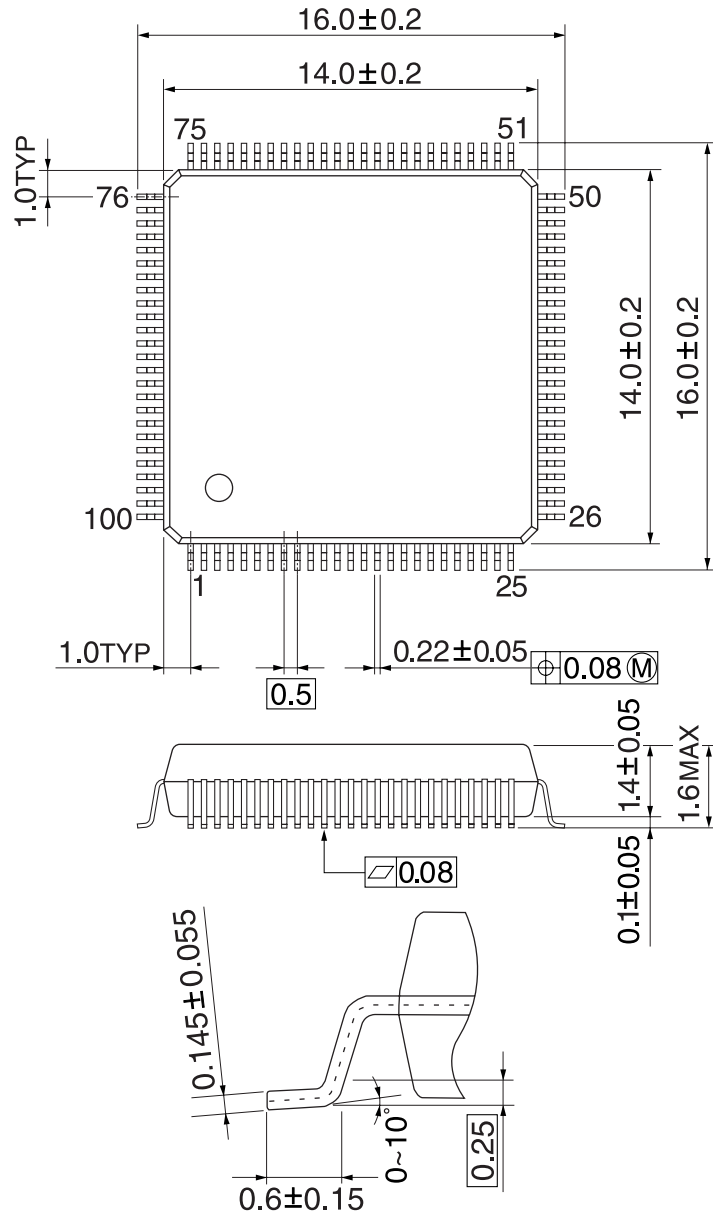

16.1.5.5 Cold Reset Timing

17. Package Dimension

LQFP100-P-1414-0.50F

Unit: mm

Appendix A.   PLL Passive Components

The Phase Locked Loop circuit requires several passive components for proper operation, which are connected to VccPLL, and VssPLL, as illustrated in Figure A.1.
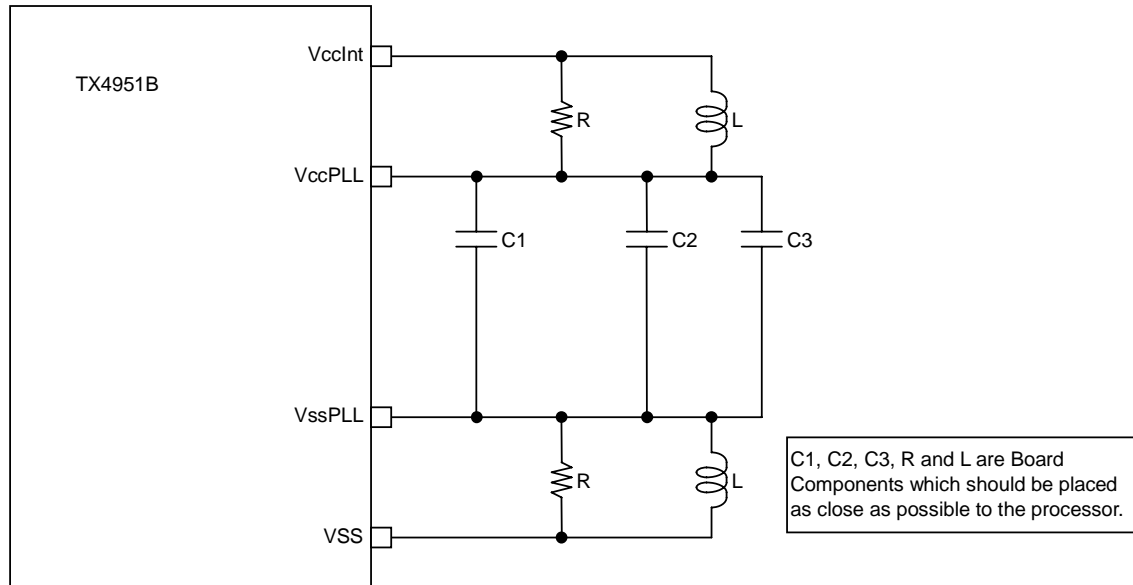

C1, C2, C3, R and L are Board Components which should be placed as close as possible to the processor.

Figure A.1 PLL Recommended Circuit

Values:

- R = 5.6 Ω [*1]
- L = 2.2 μH
- C1 = 1 nF [*1]
- C2 = 82 nF [*1]
- C3 = 10 μF [*1]
- VccInt = 1.5 V ± 0.1 V

The inductors (L) can be used as alternatives to the resistors (R) to filter the power supply.

It is essential to isolate the analog power and ground for the PLL circuit (VccPLL/VssPLL) from the regular power and ground (VccInt/Vss).

*1: These value should be changed to suitable value for each board.

Appendix B.   Movement parameter setting of a processor

A table explains movement parameter with a processor.

Item	Description
Single write protocol	These modes are selected by G2SConfig-Register. Write mode — 00 R4000 compatible / AWxx 01 reserved / reserved 10 Reissue write / AW 11 Pipeline write / AW
Writeback data rate	These modes are selected by G2SConfig-Register. Date rate — 0 AWWWWWWWWW 1 AWxxWxxWxxWxxWxxWxxWxxWxx
Clock multiplier	These modes are selected by external pin (DivMode[1:0])

Endian set	Endian is selected by external pin (Endian).
	 Indicates the initial setting of the endian during areset.
	    0: Little Endian
	    1: Big Endian

Timer Interrupt	Timer-Interrupt is selected by external pin (TintDis).
	 It is Selection of Timer-Interrupt or Int5.
	    0: Timer-Interrupt enable Int5 is disable
	    1: Timer-Interrupt disable Int5 is enable

SysAD bus protocol type	SysAD bus protocol type is selected by external pin (MODE43*).
	 It is selection of TX4300 type or R5000 type.
	    0: R4300 type
	    1: R5000 type

Clock multiplier table:

DivMode[1:0]	MasterClock	CPUCLK	ratio
00	50 MHz	200 MHz	1:4
01	80 MHz	200 MHz	1:2.5
10	100 MHz	200 MHz	1:2
11	66 MHz	200 MHz	1:3

Note1: A: Address (32 bits), W: Word (32 bits)

Note2: Initial set of data rate
        Single write: AWxx (R4000 compatible)
        Block write: AWWWWWWWWW

Appendix C.   Differences Between the TMPR4951BFG and the TMPR4955BFG

	Item	**TMPR4951BFG**	**TMPR4955BFG**
1	Full Part Number(Abbr.)	TMPR4951BFG-200 (TX4951B-200)	TMPR4955BFG-200/-300 (TX4955B-200/-300)
2	Packaging	LQFP100-P-1414-0.50F (Lead-free)	QFP160-P-2828-0.65A (Lead-free)
3	Integrated Processor Core	TX49/L3 Core (without FPU)	TX49/H3 Core
4	Instruction Cache Size	16 KB	32 KB
5	Data Cache Size	8 KB	32 KB
6	Max. Internal Operating Frequency	200 MHz	200 MHz / 300 MHz
7	Max. External Bas Frequency and Bus Width	100 MHz: 32-Bit	133 MHz: 32-Bit
8	External Bus to Internal Core Frequency Multiplication Factors	1:2, 1:2.5, 1:3, 1:4	1:2, 1:2.5, 1:3, 1:4, 1:4.5, 1:5
9	PRId (CP0) FCR0 (CP1) JTAGID	PRId: 0x00002d38 FCR0: — JTAGID: 0x0002d031	PRId: 0x00002d30 FCR0: 0x00002d30 JTAGID: 0x10017031
10	Output Buffer Choices	4 mA only	4 mA, 8 mA, 12 mA
11	External Pins	Fllowing signals have been deleted from TX4955B. SysADC[3:0], SysCmdP, Int[5:4]*, Reset*, DivMode[2], BufSel[1:0], PCST[8:0], TPC[3:1] DCLK	—
12	EJTAG Debugging	Run Control	Run Control and PC Trace
13	AC/DC Characteristics	See the appropriate datasheet	See the appropriate datasheet
14	Recommended Operating Temperature Range	Tc = -20 to 85 °C	Tc = 0 to 70 °C
15	Floating-Point Instructions	Invalid (Reserved Instruction Exception)	Valid
16	Floating-Point Registers (FGR)	none	32