PM7384

FREEDM™-84P672

FRAME ENGINE AND DATALINK MANAGER 84P672

PCI BUS UTILIZATION AND LATENCY ANALYSIS

PROPRIETARY AND CONFIDENTIAL

PRELIMINARY

ISSUE 1: JULY 1999

CONTENTS

1 OVERVIEW 1
1.1 DESIGN OBJECTIVES 2
1.2 KNOWN DESIGN CONSTRAINTS 3
1.3 ASSUMPTIONS 4
1.4 REGISTER DESCRIPTIONS 5
1.5 REFERENCES 5
2 DEFINITION OF VARIABLES 6
3 CALCULATION OF PCI BUS CYCLES 8
3.1 RECEIVE PACKETS 8
3.2 TRANSMIT PACKETS 10
3.3 INTERRUPT SERVICE OVERHEAD 12
3.4 AVERAGE NUMBER OF CYCLES PER PACKET BYTE 13
4 BUS UTILIZATION 14
4.1 UTILIZATION WITH MAXIMUM BUS EFFICIENCY 14
4.2 BUS ACCESS PRIORITIZATION 15
4.3 RECEIVE PRIORITIZATION 15
4.4 TRANSMIT PRIORITIZATION 16
4.5 AVOIDING RECEIVE OVERRUN 17
4.6 AVOIDING TRANSMIT UNDERRUN 18
5 BUS LATENCY 19
5.1 MAXIMUM TOLERABLE CHANNEL LATENCY 19
5.2 CALCULATING BUS LATENCY 20

5.2.1 PRIORITY RECEIVE CHANNEL 20
5.2.2 EXPEDITED PRIORITY TRANSMIT CHANNEL 21
5.2.3 RECEIVE CHANNEL 22
5.2.4 TRANSMIT CHANNEL 23
6 PCI PEFORMANCE FOR TYPICAL APPLICATIONS 25
6.1 84 UNCHANNELISED T1/J1 LINKS 25
6.2 63 UNCHANNELISED E1 LINKS 28
6.3 672 CHANNELS AT 192 KBPS EACH 30
6.4 ONE UNCHANNELISED DS-3 LINK AND 128 KBPS CHANNELISED T1/J1 LINKS 33
7 CONCLUSIONS AND RECOMMENDATIONS 37
APPENDIX A – DIFFERENCES BETWEEN FREEDM-32 AND FREEDM-84P67238

LIST OF FIGURES

FIGURE 1 – SYSTEM BLOCK DIAGRAM INCLUDING DATA PATH 1

FIGURE 2 – RECEIVE DMA OPERATION 9

FIGURE 3 – TRANSMIT DMA OPERATION 11

FIGURE 4 – UTILIZATION WITH 84 UNCHANNELISED T1/J1 LINKS 27

FIGURE 5 – UTILIZATION WITH 63 UNCHANNELISED E1 LINKS 29

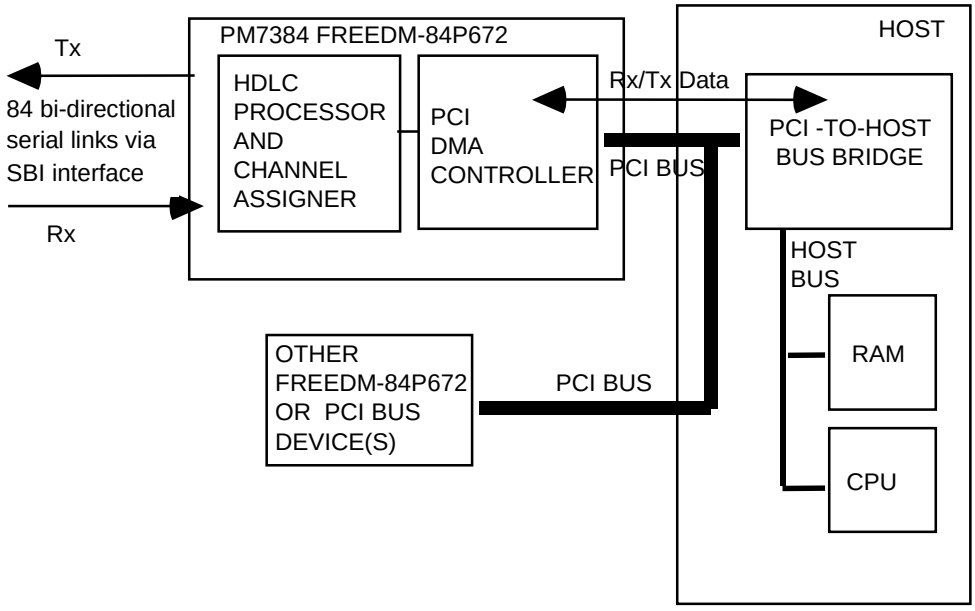
FIGURE 6 – UTILIZATION WITH 672 CHANNELS AT 192 KBPS EACH 32

FIGURE 7 – UTILIZATION WITH ONE DS-3 LINK AND 448 CHANNELS AT 128 KBPS 34

1 OVERVIEW

The FREEDM-84P672 provides HDLC processing of up to 84 bi-directional serial links, conveyed via a Scaleable Bandwidth Interconnect (SBI) interface, in Frame Relay, FUNI, PPP, Internet/Intranet access, Packet over SONET, PPP over SONET, and packet-based DSLAM applications. It interfaces directly to a host via a 66 MHz, 32 bit PCI local bus interface as described in the PCI Local Bus Specification [1], across which data is read or written by the FREEDM-84P672's DMA controller. The bus allows for the transfer of data between the host RAM and the FREEDM-84P672, at a high rate, without intervention by the host CPU.

Figure 1 – System Block Diagram Including Data Path



The flow of data is shown for a typical system in Figure 1. SBI receive circuitry receives data from the SBI interface, and converts the data to serial bit streams. The receive data from a serial link is mapped to one or more HDLC channels by the Receive Channel Assigner (RCAS672). Each channel is independently processed by the Receive HDLC Processor/Partial Packet Buffer (RHDL672), and packet data of a channel is stored in a Channel FIFO. When the level within the buffer reaches a user-defined limit, the data is burst written by the Receive DMA Controller (RMAC672) across the PCI bus. Upon completion of a packet, the FREEDM-84P672 notifies the host CPU by placing a reference to the packet in the Receive Packet Descriptor Reference (RPDR) Ready Queue, by setting bits [23:17] of the RPDR to all zeros, and optionally by interrupt. The host CPU

must replenish the RPDR Small Buffer Free Queue and the RPDR Large Buffer Free Queue, which identify references associated with free buffers in host RAM where the FREEDM-84P672 may write the receive data.

An overrun condition occurs when data from the serial data link must be written to a Receive Channel FIFO, but the Receive Channel FIFO is full. This condition can be caused by excessively long bus access latency, bandwidth starvation as the result of the FREEDM-84P672 serving other channels, or insufficient replenishment of the RPDR Small (or Large) Buffer Free Queue.

The host CPU prepares transmit packets in host RAM and places a reference to each packet in the Transmit Descriptor Reference (TDR) Ready Queue. After the FREEDM-84P672 is notified that this queue is no longer empty, the Transmit DMA Controller (TMAC672) burst reads the packet data across the bus and deposits this data into the Transmit Channel FIFO. The Transmit HDLC Processor/Partial Packet Buffer (THDL672) performs HDLC processing of the data and the Transmit Channel Assigner (TCAS672) maps the data to one of the 84 transmit serial links. SBI transmit circuitry processes the serial links to transmit the data onto the SBI interface. The FREEDM-84P672 returns the packet reference to the TDR Free Queue allowing the host CPU to confirm the transmission and reuse the descriptor memory for subsequent packets.

An underrun condition may occur during transmission of a packet when data must be driven on the serial link but the Transmit Channel FIFO is empty. In this case, the FREEDM-84P672 was unable to read data across the bus in time. This condition can be caused by excessively long bus access latency or bandwidth starvation as the result of the FREEDM-84P672 serving other channels.

1.1 Design Objectives

A system design will typically be optimized for one or more of the following criterion, all of which are affected by the PCI bus utilization of a FREEDM-84P672:

- To maximize the PCI bus efficiency, thus allowing the FREEDM-84P672 to be configured with the highest number of channels, and/or the highest data rate on each channel. Also, the designer may choose to integrate multiple FREEDM-84P672s on the same PCI bus.
- To avoid receive overrun.
- To avoid transmit underrun.
- To minimize the bus latency in either the receive or transmit direction.

- To ensure that the system performs at the aggregate link rate, and that no bandwidth on the serial links is lost.

The purpose of this application note is to outline the factors which affect PCI bus utilization in order to provide enough information for a designer to estimate the optimal configuration of the FREEDM-84P672 for the intended application. The calculations of PCI bus performance presented here are believed to be accurate although the actual bus performance may differ. The results have not been correlated with actual system designs that use the FREEDM-84P672. The objectives are to define limits of performance and to provide insight into optimizing the configuration of a FREEDM-84P672 for the application.

1.2 Known Design Constraints

There are a number of known design constraints which limit the configuration of a FREEDM-84P672. Namely, the serial links must comply with the FREEDM-84P672 longform datasheet [2]. The datasheet states the following characteristics of the FREEDM-84P672:

- Supports up to 672 bi-directional HDLC channels assigned to a maximum of 84 channelised or unchannelised links conveyed via a Scaleable Bandwidth Interconnect (SBI) interface.

- Data on the SBI interface is divided into 3 Synchronous Payload Envelopes (SPEs). Each SPE can be configured independently to carry data for either 28 T1/J1 links, 21 E1 links, or 1 unchannelised DS-3 link.

- Links in a SPE can be configured individually to operate in a clear channel mode, in which case all framing bit locations are assumed to be carrying HDLC data.

- Links in an SPE can be configured individually to operate in channelised mode, in which case, the number of time-slots assigned to an HDLC channel is programmable from 1 to 24 (for T1/J1 links) and from 1 to 31 (for E1 links).

- Supports three bi-directional HDLC channels each assigned to an unchannelised link with arbitrary rate link of up to 52 MHz when SYSCLK is running at 40 MHz. Each link may be configured individually to replace one of the SPEs conveyed on the SBI interface.

- The FREEDM-84P672 interfaces directly to a PCI local bus which is revision 2.1 compliant, has a 32 bit data path and can operate at a maximum clock speed of 66 MHz. This places a limit on the maximum number of FREEDM-

84P672 devices interfaced to the PCI bus, and on the aggregate data rate of all of the serial links which can be handled by the bus.

1.3 Assumptions

The bus performance is affected by the system design choices made in integrating the FREEDM-84P672 with an embedded host. The following list of design choices is important:

- RAM access time and contention
- PCI bus clock speed
- PCI bridge design and arbitration algorithm
- System software running on the host CPU, and the processing capability of the CPU
- Other PCI devices on the PCI bus

A thorough discussion of the above choices is outside the scope of this document. For the purposes of this analysis, the following list of assumptions is made. Additional assumptions are stated in this document when encountered.

- The PCI bus Latency Timer is set large enough such that none of the transactions are broken.
- The host RAM is always available when required. The PCI bus does not disconnect the FREEDM-84P672 from accessing the host RAM.
- The host CPU has enough processing power that it will not be a bottleneck.
- The bridge uses a fair round-robin arbitration algorithm to grant PCI devices access to the bus.
- One or more FREEDM-84P672 devices as well as the host CPU utilize bus cycles. No other PCI devices are present.
- The serial link is fully saturated with HDLC data, and there is only one flag byte between frames. The data bit rate of the serial link is the same as the clock rate of the serial link.
- FCS fields are not DMA'd across the PCI bus. The packet length used in the calculations does not include the FCS bytes. The FCS can be optionally

dropped in the receive direction and are not DMA'd across the PCI bus in the transmit direction.

- The effective data rate of a serial link or channel used in the analysis does not account for bit stuffing, bit destuffing and the overhead associated with T1/E1/T3 framing which normally occurs. This overhead is not DMA'd across the PCI bus.

- Each packet is completely stored in a single receive or transmit buffer. In cases where a packet spans multiple buffers, the bus latency estimates are valid if the buffer size is a multiple of the DMA transfer size, XFER[3:0] + 1.

1.4 Register Descriptions

Throughout this document, FREEDM-84P672 registers are described using the following convention: **Register Name** (byte offset from base address).

1.5 References

1. PCI Special Interest Group, PCI Local Bus Specification, June 1, 1995, Version 2.1.

2. PMC-990445, PMC-Sierra, Inc., “Frame Engine and Data Link Manager 84P672” Longform Datasheet, July 1999, Issue 2.

3. PMC-990715, PMC-Sierra, Inc., “FREEDM-84P672 Programmer’s Guide”, June 1999, Issue 1.

2 DEFINITION OF VARIABLES

The following table is a summary of the variables and parameters used throughout this document, in alphabetical order:

Variable	Description	Unit
B	Number of references accessed at one time by host software	References
C_{Byte}	Bus cycles required to receive and transmit one packet byte	Bus cycles/byte
C_{RxByte}	Bus cycles required to receive one packet byte	Bus cycles/byte
$C_{Transaction}$	Maximum number of cycles utilized by a channel	Bus cycles
C_{TxByte}	Bus cycles required to transmit one packet byte	Bus cycles/byte
D	Data transfer size	DWORDs
E	Starving trigger level for expedited transmit channels	Free 16 byte blocks
F	Size of channel FIFO assigned to each channel	16 byte blocks
$f_{Channel}$	Channel data rate	bits/second
f_{PCI}	PCI bus clock frequency	Hz
$L_{Channel}$	Maximum tolerable channel latency	seconds
L_{PCI}	Worst case PCI bus latency	seconds
N	Maximum number of references in TDR Ready queue	References
P	Packet length	bytes
Q	Number of references in each queue	References
r	Read latency cycles	Bus cycles
S	Start transmission level	Free 16 byte blocks
$_{Channel}$	Number of HDLC channels provisioned	Channels
U	PCI bus utilization	

Variable	Description	Unit
w	Write latency cycles	Bus cycles
X	DMA channel transfer size Equal to XFER[3:0] + 1	16 byte blocks

3 CALCULATION OF PCI BUS CYCLES

There are four PCI bus transactions that are used to transfer data between the FREEDM-84P672, the host RAM and the host CPU. The PCI bus utilization during the receive and transmit of packets is composed entirely of these four transactions, which are defined in the longform datasheet [2]:

- **Burst read from host RAM:** This PCI read transaction has $3+D+r$ bus cycles. These cycles consist of an address cycle, bus turn around cycle, data cycles, RAM access latency cycles, and a final bus turn around cycle.

- **Burst write to host RAM:** This PCI write transaction has $2+D+w$ bus cycles. These cycles consist of an address cycle, data cycles, RAM access latency cycles, and a final bus turn around cycle.

- **Read FREEDM-84P672 register:** This PCI read transaction has 9 bus cycles. This value is based on simulation of the FREEDM-84P672.

- **Write FREEDM-84P672 register:** This PCI write transaction has 9 bus cycles. This value is based on simulation of the FREEDM-84P672.

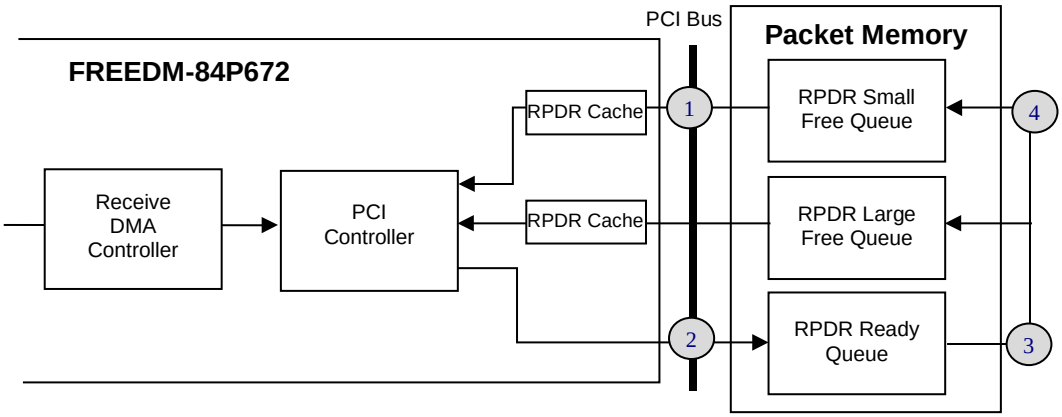
Here, D is the data size in DWORDs (32 bits), r is the read latency cycles, and w is the write latency cycles. The latency cycles are inherent to the host RAM, the bridge, the host CPU, and the system software. This analysis assumes that none of the bus masters are disconnected, and that transactions complete without being stopped.

3.1 Receive Packets

The Receive DMA Controller (RMAC672) and the host exchange information using Receive Packet Descriptors (RPDs). The RPD holds the data buffer size, data buffer address, and packet status information. The Receive Packet Descriptor Reference (RPDR) is a pointer that is used to index into a table of RPDs. The RPDR Ready queue and the RPDR Free queues allow the RMAC672 and the host to pass RPDRs back and forth. The **Error! Unknown document property name.** provides two RPDR Free queues to support small and large buffers.

The receive DMA operation sequence is illustrated in Figure 2.

Figure 2 – Receive DMA Operation



For this analysis, it is assumed that the receive packets are composed of a single descriptor or buffer. Linking of multiple descriptors to form a packet involves additional overhead activity on the PCI bus.

The FCS bytes are assumed to be discarded in the receive direction by setting the STRIP bit high in the **RHDL Indirect Channel Data #1** (0x204) register. Therefore, the packet length in this analysis does not include FCS bytes.

For a receive packet, the following bus transactions are required on a per packet basis:

1. The FREEDM-84P672 reads references associated with small free buffers from the RPDR Small Free queue. When the SCACHE bit in the **RMAC Control** (0x280) register is set high, up to 6 free references are read from the RPDR Small Free queue in one transaction every sixth reference and stored in the RPDR cache. This PCI transaction has $(3+6+r)/6$ bus cycles per packet.

2. The FREEDM-84P672 receives data and adds an RPDR to the RPDR Ready queue:

   i) The FREEDM-84P672 reads 4 DWORDs in the RPD to determine the size and address of the data buffer in host memory. This PCI transaction has $3+4+r$ bus cycles.

   ii) The FREEDM-84P672 writes receive packet data to the buffer. This may require multiple burst write transactions since the FREEDM-84P672 can only DMA X blocks of data per transaction. The number of DWORD data

cycles required to complete the data transfer is $P/4$ [1], the number of PCI transactions required for the entire packet is $P/(16 \cdot X)$, and the number of overhead cycles for each transaction is $(2+w)$. In total, this activity has $P/4 + (2+w) \cdot \lceil P/(16 \cdot X) \rceil$ bus cycles. Here, P is the packet length in bytes, and X is the DMA transfer size in 16 byte blocks.

iii) When the end of packet occurs, the FREEDM-84P672 writes 2 DWORDs in the RPD; the fields written are CE, RCC, Status, Bytes in Buffer, and Offset. This PCI transaction has $2+2+w$ bus cycles.

iv) The FREEDM-84P672 writes the reference (STATUS[1:0] + RPDR[14:0]) to the RPDR Ready queue. This PCI transaction has $2+1+w$ bus cycles.

3. The host reads references from the RPDR Ready queue in blocks of up to B references, thereby writing the **RMAC Packet Descriptor Reference Ready Queue Read** (0x2C0) register every B'th reference and reading the **RMAC Packet Descriptor Reference Ready Queue Write** (0x2BC) register every Q'th reference. Here, Q is the size of the queue and B is the number of references in the block. This activity has $9/B + 9/Q$ bus cycles per packet.

4. The host writes references to the RPDR Small Free queue in blocks of up to B references, thereby writing the **RMAC Packet Descriptor Reference Small Buffer Free Queue Write** (0x2AC) register every B'th reference and reading the **RMAC Packet Descriptor Reference Small Buffer Free Queue Read** (0x2B0) register every Q'th reference. Here, Q is the size of the queue and B is the number of references in the block. This activity has $9/B + 9/Q$ bus cycles per packet.

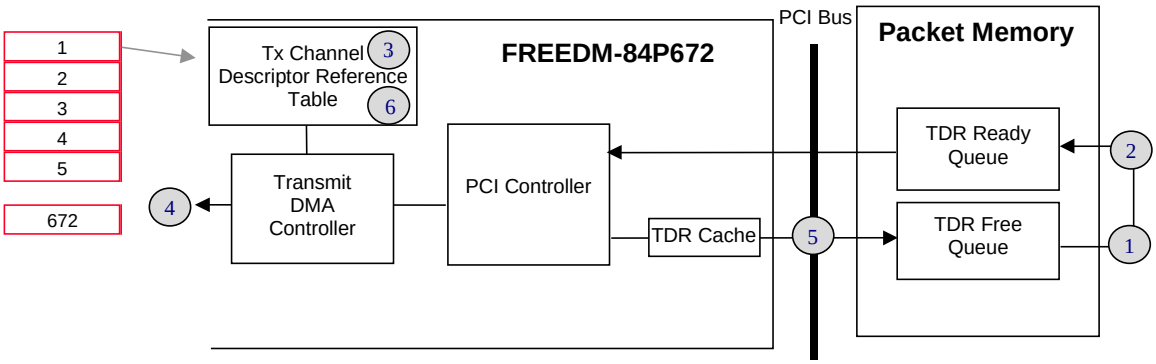
3.2 Transmit Packets

The Transmit DMA Controller (TMAC672) and the host exchange information using transmit descriptors (TDs). The TD holds the data buffer size, data buffer address, and other packet information. The Transmit Descriptor Reference (TDR) is a pointer which is used to index into a table of TDs. The TDR Ready queue and the TDR Free queue allow the TMAC672 and the host to pass TDRs back and forth.

The transmit DMA operation sequence is illustrated in Figure 3.

[1] The Roof function outputs an integer value, whereby a non-zero remainder receives the value 1. For example, the Roof Function $\lceil 3.03 \rceil$ gives an output of 4, whereas the Roof Function $\lceil 3.00 \rceil$ gives an output of 3.

Figure 3 – Transmit DMA Operation



For this analysis, it is assumed that the transmit packets are composed of a single descriptor or buffer. Linking of multiple descriptors to form a packet involves additional overhead activity on the PCI bus.

For a transmit packet, the following bus transactions are required on a per packet basis:

1. The host reads references from the TDR Free queue in blocks of up to B references, thereby writing the **TMAC Descriptor Reference Free Queue Read** (0x320) register every B'th reference, and reading the **TMAC Descriptor Reference Free Queue Write** (0x31C) register every Q'th reference. Here, Q is the size of the queue and B is the number of references in the block. This activity has $9/B+9/Q$ bus cycles per packet.

2. The host writes references to the TDR Ready queue in blocks of up to B references, thereby writing the **TMAC Descriptor Reference Ready Queue Write** (0x32C) register every B'th reference, and reading the **TMAC Descriptor Reference Ready Queue Read** (0x330) register every Q'th reference. Here, Q is the size of the queue and B is the number of references in the block. This activity has $9/B+9/Q$ bus cycles per packet.

3. The FREEDM-84P672 adds a packet to the Transmit Channel Descriptor Reference (TCDR) table:

   i) The FREEDM-84P672 reads one reference from the TDR Ready queue. This PCI transaction has $3+1+r$ bus cycles.

   ii) The FREEDM-84P672 reads 3 DWORDs in the TD to obtain transmit channel and other DMA information for storage in the TCDR table. This PCI transaction has $3+3+r$ bus cycles.

iii) The FREEDM-84P672 links the reference to the end of the linked list of references for the specified HDLC channel and priority level by writing 1 DWORD in the TD at the end of the linked list. The TMAC Next TD Pointer field and the V bit of the last TD are written. This PCI transaction has $2+1+w$ bus cycles.

4. The FREEDM-84P672 reads and transmits buffer data. This may require multiple burst read transactions since the FREEDM-84P672 can only DMA X blocks of data per transaction. The number of DWORD data cycles required to complete the data transfer is $\lceil P/4$, the number of PCI transactions required for the entire packet is $\lceil P/(16 \cdot X)$, and the number of overhead cycles for each transaction is $(3+r)$. In total, this activity has $P/4 + (3+r) \cdot \lceil P/(16 \cdot X)$ bus cycles. Here, P is the packet length in bytes, and X is the DMA transfer size in 16 byte blocks.

5. The FREEDM-84P672 writes references (STATUS[2:0] + TDR[14:0]) to the TDR Free queue. When the CACHE bit in the **TMAC Control** (0x300) register is set high, up to 6 free references from the TDR cache are written to the TDR Free queue in one transaction every sixth reference. This PCI transaction has $(2+6+w)/6$ bus cycles per packet.

6. The FREEDM-84P672 reads 3 DWORDs in the TD of the following packet in the linked list. This PCI transaction has $3+3+r$ bus cycles.

3.3 Interrupt Service Overhead

It is assumed that bus cycles due to interrupt servicing are negligible in comparison to the number and size of packets DMA'd across the bus. This assumption is ensured to be correct by programming the FREEDM-84P672 to interrupt the host less frequently than with every packet or descriptor processed by the FREEDM-84P672. This is done via the LCACHE, SCACHE, RPQ_RDYN, RPQ_LFN and RPQ_SFN bits within the **RMAC Control** (0x280) register and via the CACHE, TDQ_RDYN and TDQ_FRN bits of the **TMAC Control** (0x300) register.

It is assumed that the interrupt service routine processes each interrupt by reading/clearing the **FREEDM-84P672 Master Interrupt Status** (0x008) register, and then storing the status within the software for deferred processing.

The bus cycles due to interrupt servicing is assumed negligible compared to the other PCI bus activity.

3.4 Average Number of Cycles Per Packet Byte

For a receive packet, the average number of bus cycles to write the receive data into host RAM is derived by summation of all activities described in section 3.1 for receive packets. The bus cycles per byte of packet received is expressed as:

$$C_{RxByte} = \frac{1}{P}\left\lceil 15.5 + 18 \cdot \left(\frac{1}{B} + \frac{1}{Q}\right) + \left\lceil \frac{P}{4} \right\rceil + (2 + w) \cdot \left\lceil \frac{P}{16 \cdot X} \right\rceil + 1.17 \cdot r + 2 \cdot w \right\rceil$$

For a transmit packet, the average number of bus cycles to read the transmit data from host RAM is derived by summation of all activities described in section 3.2 for transmit packets. The bus cycles per byte of packet transmitted is expressed as:

$$C_{TxByte} = \frac{1}{P}\left\lceil 20.33 + 18 \cdot \left(\frac{1}{B} + \frac{1}{Q}\right) + \left\lceil \frac{P}{4} \right\rceil + (3 + r) \cdot \left\lceil \frac{P}{16 \cdot X} \right\rceil + 3 \cdot r + 1.17 \cdot w \right\rceil$$

Combining the equations for C_{RxByte} and C_{TxByte}, the total cycles to receive and transmit one byte of packet data is expressed as:

$$C_{Byte} = \frac{1}{P}\left\lceil 35.83 + 36 \cdot \left(\frac{1}{B} + \frac{1}{Q}\right) + 2 \cdot \left\lceil \frac{P}{4} \right\rceil + (5 + r + w) \cdot \left\lceil \frac{P}{16 \cdot X} \right\rceil + 4.17 \cdot r + 3.17 \cdot w \right\rceil$$

where P is the packet length in bytes, X is the XFER size in 16 byte blocks, Q is the size of each queue, B is the number of queued references accessed by the host CPU at one time, r is the read latency cycles, and w is the write latency cycles. These equations are derived under the assumption that PCI bus activity due to interrupts, provisioning and unprovisioning of channels, and performance counter polling is negligible.

4 BUS UTILIZATION

Bus utilization is defined as the ratio of the number of clock cycles that transfer data to the number of cycles that is available to transfer data during a time interval. The traffic on the PCI bus due to the FREEDM-84P672 is bursty by nature, and the time interval must be sufficiently large. If too small of a time interval is chosen, and the time interval being measured occurs when there is much traffic on the bus, the measured bus utilization could be higher than the long term average. Alternatively, if the measurement interval is chosen when there is little traffic on the bus, the measured utilization could be lower than the long term average.

4.1 Utilization With Maximum Bus Efficiency

The best estimate for measuring bus utilization is obtained by using the longest time interval to make the measurement. This is expressed mathematically as the average bus utilization, and is derived below.

The average number of bus cycles required to receive and transmit during a one second interval is a function of the data rate for each channel provisioned on a FREEDM-84P672. It is expressed as follows:

$$BusCycles = \sum_{Channel} \left(C_{Byte} \cdot f_{Channel} / 8 \right)$$

where $f_{Channel}$ is the channel data rate in bits per second and C_{Byte} is the average number of cycles to DMA one receive and one transmit byte on that channel.

The number of PCI bus cycles available during the one second interval is f_{PCI}, the PCI bus clock frequency. The PCI bus utilization, expressed as the ratio of these is:

$$U = \frac{1}{8 \cdot f_{PCI}} \cdot \sum_{Channel} \left(C_{Byte} \cdot f_{Channel} \right)$$

where C_{Byte} was derived in the previous section. This expresses the PCI bus utilization due to a bi-directional serial link with channel rate specified in bits per second. The total bus utilization for multiple FREEDM-84P672 devices on a bus is obtained by the summation over all provisioned channels, and on all FREEDM-84P672 devices connected to the bus.

Configuration of the FREEDM-84P672 channels and serial links such that the average utilization does not exceed 1 (or 100%), based on this equation, does

not guarantee the avoidance of receive overrun or transmit underrun. This can only be guaranteed by ensuring that each Channel FIFO is large enough to tolerate the bus latencies. This problem will be examined in section 5 with calculations of bus latency.

4.2 Bus Access Prioritization

Overutilization of the bus may cause receive channel overrun or transmit channel underrun. The channels that are affected would depend on their prioritization in relation to all of the other provisioned channels, and any other activities on the bus. The following list identifies the prioritization of transactions on the PCI bus (1 = highest priority).

1. PCI bus interrupt acknowledge cycles due to PCI bus interrupts may occur on some systems with ISA bus bridges. These are assumed not to occur in this analysis and will not be discussed further.

2. Host CPU accesses to the **FREEDM-84P672 Master Interrupt Status** (0x008) register and the **FREEDM-84P672 Master SBI Interrupt Status** (0x02C) register.

3. Host CPU accesses to any of the FREEDM-84P672 queue write (or read) index registers.

4. FREEDM-84P672 accesses to queue references, descriptors and buffers as required to transfer packets between the FREEDM-84P672 and the host RAM. In the case of multiple FREEDM-84P672s on the same bus, the host bridge would dictate whether each FREEDM-84P672 is granted access in a round-robin basis.

4.3 Receive Prioritization

Receive transactions of a FREEDM-84P672 differ in priority as follows (1 = highest priority):

1. FREEDM-84P672 accesses to the RPDR Small Free queue or the RPDR Large Free queue. This transaction occurs whenever the internal cache of free buffer RPDRs needs to be replenished.

2. Transactions for priority receive channels. These are provisioned channels with the PRIORITY bit set within the **RHDL Indirect Channel Data #2** (0x208) register.

3. Transactions for the remaining receive channels.

Each receive channel of the same priority is polled in a round robin basis for access to the bus, but internal to the FREEDM-84P672, there are two sequences of data transactions that are atomic (i.e. once started, they cannot be interrupted by receive transactions of another channel, although successive receive transactions can be interleaved with transmit transactions). The sequences are:

Start of Packet Sequence

- FREEDM-84P672 reads up to 6 references from the RPDR Small Free queue (or the RPDR Large Free queue). This transaction only occurs once every sixth start of packet sequence.
- FREEDM-84P672 reads 4 DWORDs in the RPD.
- FREEDM-84P672 writes receive data into the buffer of the RPD. This transaction writes up to X blocks of data to host RAM, where X is the maximum number of Channel FIFO blocks that is written per transaction, and is specified upon provisioning of a channel.

End of Packet Sequence

- FREEDM-84P672 writes final bytes of a packet to the receive buffer.
- FREEDM-84P672 writes 2 DWORDs in the RPD.
- FREEDM-84P672 writes a reference to the RPDR Ready queue.

4.4 Transmit Prioritization

Transmit transactions of a FREEDM-84P672 differ in priority as follows (1= highest priority):

1. FREEDM-84P672 accesses to the TDR Free queue. This transaction occurs if the internal cache of transmit references has reached the cache size limit of 6, assuming the CACHE bit of the **TMAC Control** (0x300) register is set.
2. FREEDM-84P672 accesses to host RAM for a transmit channel which is not inhibited from making expedited requests for data and where the Channel FIFO has reached the starving trigger level. These are provisioned channels with PRIORITYB = 0 in the **THDL Indirect Channel Data #2** (0x388) register.
3. FREEDM-84P672 accesses to the TDR Ready queue and linking of each transmit descriptor read. These transactions occur if the host has written transmit packet references into the queue so that this queue is not empty.

4. FREEDM-84P672 accesses packet data, descriptors, and queue references for any transmit channel which has enough space in its Channel FIFO for the remaining bytes of a packet or the DMA transfer size.

Each transmit channel is polled in a round robin basis for access to the bus, but internal to the FREEDM-84P672, there are two sequences of data transactions that are atomic (i.e. once started, they cannot be interrupted by transmit transactions of another channel, although successive transmit transactions can be interleaved with receive transactions). The sequences are:

Chain Packets of TDR Ready Queue Sequence

- FREEDM-84P672 reads a transmit reference from TDR Ready queue.
- FREEDM-84P672 reads 3 DWORDs in the TD.
- FREEDM-84P672 writes 1 DWORD in the TD at the end of the linked list to link packets of a channel.

End of Packet Sequence

- FREEDM-84P672 reads final bytes of a packet from the transmit buffer. This transaction reads up to X blocks of data from host RAM, where X is the maximum number of Channel FIFO blocks that is read per transaction and is specified upon provisioning of the transmit channel.
- FREEDM-84P672 writes 6 transmit references to the TDR Free queue if the cache of free references has reached 6.
- FREEDM-84P672 reads 3 DWORDs in the TD of the following packet.

4.5 Avoiding Receive Overrun

In systems in which there are multiple channel rates, the metric $f_{Channel}/F$ can be used to evaluate which channels should have higher priority. The variable $f_{Channel}$ is the data rate of the channel and F is the number of Channel FIFO blocks which are provisioned for the channel. Channels with a higher value of this metric should be provisioned with the PRIORITY bit set high within the **RHDL Indirect Channel Data #2** (0x208) register.

4.6 Avoiding Transmit Underrun

In systems in which there are multiple transmit channel rates, the metric $f_{Channel}/F$ can be used to evaluate which channels should have higher priority access to the bus. The variable $f_{Channel}$ is the data rate of the channel and F is the number of Channel FIFO blocks which are provisioned for the channel. Channels with a higher value of this metric should be provisioned with the PRIORITYB bit set low within the **THDL Indirect Channel Data #2** (0x388) register. This will ensure that the bus access latency is reduced for high priority channels where the Channel FIFO has reached the starving trigger level.

5 BUS LATENCY

The bus latency is a measure of the time interval beginning when a channel requires access to the bus and ending when access has been granted, or when the DMA transaction is completed. The worst case latency for data transfers to/from host RAM is analyzed here. The bus latency calculation is used to determine the possibility of receive overrun or transmit underrun.

5.1 Maximum Tolerable Channel Latency

The Channel FIFO of each channel must be serviced at least once within a time interval to prevent the occurrence of an overrun, or underrun. This time interval is the channel latency, given by $L_{Channel}$.

For a receive channel, the time interval is measured from the point when the receive channel has filled X blocks of its Channel FIFO. The interval ends when the channel has been given access to the bus, or in the worst case, when the Channel FIFO is full.

For a transmit channel, the time interval is measured from the point when the Channel FIFO has reached the start transmission, or starving trigger level. The interval ends when the channel has read X blocks of data across the bus, or in the worst case, when the Channel FIFO is empty.

The equations which specify the channel latency are:

$$L_{Channel} = \frac{128 \cdot (F - X)}{f_{Channel}}$$ for receive channel, PRIORITY = 0 or 1

$$L_{Channel} = \frac{128 \cdot (F - S)}{f_{Channel}} - \frac{3 + 4 \cdot X + r}{f_{PCI}}$$ for transmit channel, PRIORITYB = 1

$$L_{Channel} = \frac{128 \cdot (F + E - 2 \cdot S)}{f_{Channel}} - \frac{3 + 4 \cdot X + r}{f_{PCI}}$$ for transmit channel, PRIORITYB = 0

where F is the Channel FIFO size specified in blocks, X is the DMA transfer size specified in blocks, S is the start transmission level specified in free blocks, E is the starving trigger level, which triggers expedited data transfer requests to the TMAC672, specified in free blocks, and $f_{Channel}$ is the channel rate specified in bits per second. The start transmission level and the starving trigger level are configured by the TRANS bit and the LEVEL[3:0] bits within the **THDL Indirect Channel Data #3** (0x38C) register as described in the longform datasheet [2].

To prevent an overrun or underrun condition that is caused by bus activity from other channels or the host, the bus latency must be less than the channel latency. This is expressed as:

$$L_{PCI} < L_{Channel}$$

where L_{PCI} is the worst case PCI bus latency due to activity from other channels or the host on the bus.

5.2 Calculating Bus Latency

The PCI bus latency is estimated as:

$$L_{PCI} = \frac{1}{f_{PCI}} \cdot \sum_{Channel} C_{Transaction}$$

where $C_{Transaction}$ is the maximum bus cycles utilized by a channel. The summation is performed over provisioned channels of the FREEDM-84P672 that have equal or higher priority access to the PCI bus than the channel being evaluated for overrun or underrun. Utilization of bus cycles by the host is assumed negligible. The following sections provide equations for a variety of channel configurations.

5.2.1 Priority Receive Channel

This section provides an estimate of bus latency for a priority receive channel. This is a receive channel for which PRIORITY = 1. The bus access latency is due to all other priority receive channels, and transmit channels. Each priority receive channel is granted access in a round robin algorithm and utilization of the bus by the host CPU to access FREEDM-84P672 registers is assumed to be negligible.

The time interval is measured starting from when the channel requires bus access and ending when the bus access has been granted. The time to DMA X blocks of data across the bus is not required for this calculation as the FREEDM-84P672 double buffers the X blocks of data before granting access to the bus. The bus access latency is due to:

- completion of the current PCI transaction, which in the worst case is the receive channel start of packet sequence and involves caching of 6 free references.
- completion of the end-of-packet sequence on all of the other priority receive channels.

- DMA of X_{Tx} blocks of transmit packet data between each of the receive channel transactions and sequences.

The equation which estimates the worst case bus latency is:

$$L_{PCI} \cong \frac{1}{f_{PCI}} \cdot \left(24 + 12 \cdot X + w + 4 \cdot r + \sum_{Other\ Priority Rx} \left(18 + 12 \cdot X_{Tx} + 4 \cdot X_{Rx} + 3 \cdot w + 3 \cdot r \right) \right)$$

where X_{Rx} is the transfer size of other priority receive channels in blocks, X_{Tx} is the largest transfer size of a provisioned transmit channel in blocks, X is the largest transfer size of a provisioned transmit or receive channel in blocks, r is the read latency cycles, and w is the write latency cycles. The summation is performed over all other priority receive channels.

5.2.2 Expedited Priority Transmit Channel

This section provides an estimate of bus latency for an expedited priority transmit channel. This is a transmit channel for which PRIORITYB = 0, the Channel FIFO contains less than one packet of data, and the starving trigger level has been reached. The bus access latency is due to all other expedited transmit channels and receive channels. Each expedited transmit channel is granted access in a round robin algorithm and utilization of the bus by the host CPU to access FREEDM-84P672 registers is assumed to be negligible.

The time interval is measured starting from when the channel requires bus access and ending when the X blocks of data has been transferred across the bus. The bus access latency is due to:

- completion of the current PCI transaction, which in the worst case is a transmit channel end-of-packet sequence involving the writing of 6 cached free references, for a transmit channel provisioned with PRIORITYB = 1.

- completion of the end-of-packet sequence on all of the other transmit channels for which PRIORITYB = 0.

- DMA of X_{Rx} blocks of receive packet data between each of the transmit channel transactions and sequences.

The following assumptions are made for this estimate:

- There are less than 6 other expedited transmit channels.

- The number of receive channels waiting for access to the bus is much larger than the number of expedited transmit channels.

The equation for this estimate is:

$$L_{PCI} \cong \frac{1}{f_{PCI}} \cdot \left(21 + 12 \cdot X + 2 \cdot r + 3 \cdot w + \left(10 + 4 \cdot X_{Rx} + 2 \cdot w\right) \cdot \left\lceil \sum_{ExpeditedTxChannel} \frac{1}{6} \right\rceil \right.$$

$$\left. + \sum_{ExpeditedTxChannel} \left(13 + 8 \cdot X_{Rx} + 4 \cdot X_{Tx} + 2 \cdot r + 2 \cdot w\right) \right)$$

where X_{Rx} is the largest transfer size of a receive channel in blocks, X_{Tx} is the transfer size of an expedited transmit channel in blocks, X is the largest transfer size of a provisioned transmit or receive channel in blocks, r is the read latency cycles, and w is the write latency cycles. The summation is performed over all other expedited transmit channels.

5.2.3 Receive Channel

This section provides an estimate of bus latency for a FREEDM-84P672 configuration in which the receive channel has PRIORITY = 0. The worst case bus latency is due to activity from all other receive and transmit channels. The host accesses to FREEDM-84P672 registers are assumed to be negligible.

The time interval is measured starting from when the channel requires bus access and ending when the bus access has been granted. The time to DMA X blocks of data across the bus is not required for this calculation as the FREEDM-84P672 double buffers the X blocks of data before granting access to the bus. The bus access latency is due to:

- completion of the current PCI transaction, which in the worst case is a transmit channel transaction involving DMA of X_{Tx} blocks of packet data.
- completion of the start-of-packet sequence on all of the other receive channels.
- DMA of X_{Tx} blocks of transmit packet data between each of the receive channel transactions and sequences.

The equation which estimates the worst case bus latency is:

$$L_{PCI} \cong \frac{1}{f_{PCI}} \cdot \left(3 + 4 \cdot X_{Tx} + r + \left(4 \cdot X_{Tx} + 12 + 2 \cdot r\right) \cdot \left\lceil \sum_{OtherRxChannel} \frac{1}{6} \right\rceil \right.$$

$$+ \sum_{OtherRxChannel} \left(15 + 8 \cdot X_{Tx} + 4 \cdot X_{Rx} + w + 3 \cdot r\right)$$

where X_{Rx} is the transfer size of other receive channels in blocks, X_{Tx} is the largest transfer size of a provisioned transmit channel in blocks, r is the read latency cycles, and w is the write latency cycles. The summation is performed over all other receive channels.

5.2.4 Transmit Channel

This section provides an estimate of bus latency for a FREEDM-84P672 configuration in which the transmit channel is not in an expedited state, due to PRIORITYB = 1 or the Channel FIFO not having reached the starving trigger level. The worst case bus latency is due to activity from all receive and transmit channels. The host accesses to FREEDM-84P672 registers are assumed to be negligible.

The time interval is measured starting from when the channel requires bus access and ending when the X_{Tx} blocks of data has been transferred across the bus. The bus access latency is due to:

- chaining and linking of packets read by the FREEDM-84P672 from the TDR Ready queue.
- end-of-packet processing on all transmit channels.
- DMA of X_{Rx} blocks of receive packet data between each of the transmit channel transactions and sequences.

The equation for this estimate is:

$$L_{PCI} \cong \frac{1}{f_{PCI}} \cdot \left((10 + 4 \cdot X_{Rx} + 2 \cdot w) \cdot \left\lceil \sum_{OtherTxChannel} \frac{1}{6} \right\rceil \right.$$

$$\left. + N \cdot (19 + 12 \cdot X_{Rx} + 2 \cdot r + 4 \cdot w) + \sum_{OtherTxChannel} (13 + 4 \cdot X_{Tx} + 8 \cdot X_{Rx} + 2 \cdot r + 2 \cdot w) \right)$$

where X_{Rx} is the largest transfer size of a receive channel in blocks, X_{Tx} is the transfer size of a transmit channel in blocks, N is the number of references in the TDR Ready queue, r is the read latency cycles, and w is the write latency cycles. The summation is calculated over all other transmit channels.

Note: The software running on the host must ensure that there are few transmit references in the TDR Ready queue; otherwise, the value of N would be large, and the bus latency would also be large.

6 PCI PEFORMANCE FOR TYPICAL APPLICATIONS

This section provides PCI bus utilization graphs and latency estimates for four typical applications of the FREEDM-84P672. The utilization graphs are based on the equation for utilization given in section 4.1. The latency estimates are based on the equations given in section 5.

The following notes are guidelines for choosing the configurable parameters:

- Using a mixed configuration of DMA transfer sizes, X, where one channel has a large DMA transfer size compared to other channels may affect bus access latencies for those other channels.

- In the receive direction, the DMA transfer size should be set such that the number of blocks transferred is at least two fewer than the total allocated to the associated channel.

- To prevent lockup, the DMA transfer size in the transmit direction can be configured to be less than or equal to the start transmission level set by LEVEL[3:0] and TRANS in the **THDL Indirect Channel Data #3** (0x38C) register. Also, the DMA transfer size can be set such that the total number of blocks in the logical channel FIFO minus the start transmission level is an integer multiple of the DMA transfer size.

- To obtain the best possible bus utilization, the size of a queue should not be too small, as this would lead to more frequent accesses to the read and/or write index registers of the FREEDM-84P672. The minimum recommended queue size is approximately 32 references. In general, the queue should be large enough to hold one reference per provisioned channel.

6.1 84 Unchannelised T1/J1 Links

Up to 84 unchannelised T1/J1 links can be physically connected to a FREEDM-84P672 device. All three of the SPEs need to be configured to carry 28 T1/J1 links each. In this case, 84 bi-directional HDLC channels are provisioned, each assigned to an unchannelised T1/J1 link.

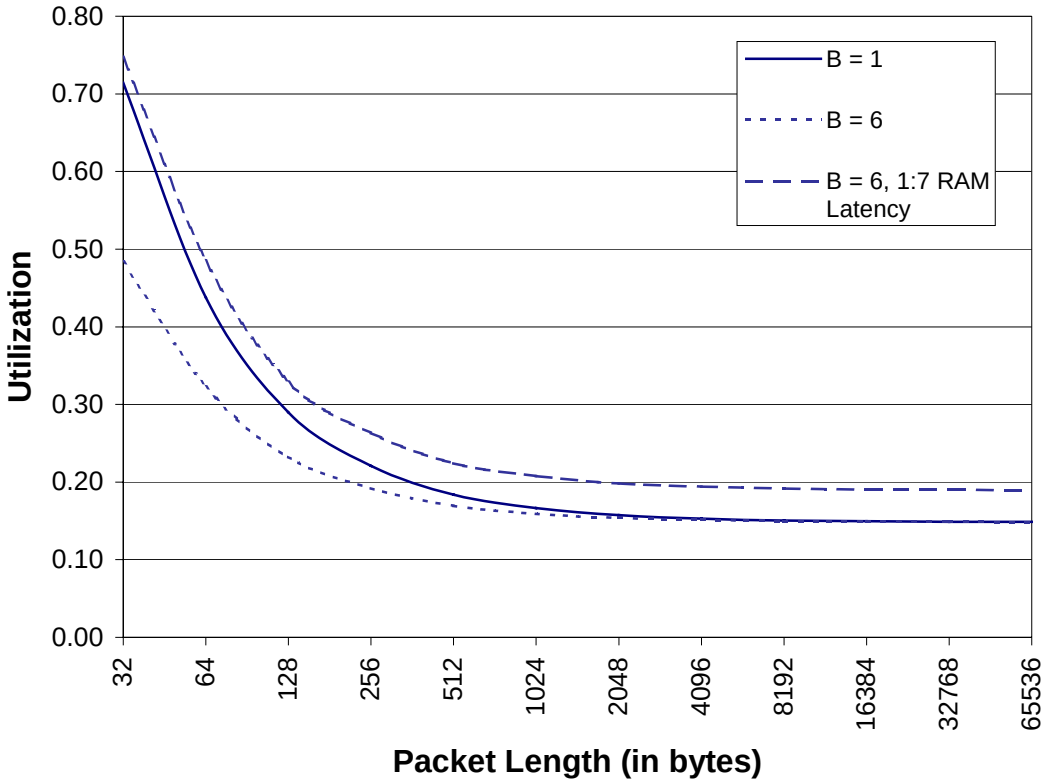
The estimates for the unchannelised T1/J1 application are based on the following configuration:

Configurable Parameter	Value	Description
f_{PCI}	66 MHz	PCI bus clock frequency

Configurable Parameter	Value	Description
$f_{Channel}$	1.544 Mbps	Channel data rate
$_{Channel}$	84	Number of HDLC channels provisioned
F	24	Size of channel FIFO assigned to each channel (in 16 byte blocks)
S	6	Start transmission level (in free blocks)
E	12	Starving trigger level for expedited channels (in free blocks)
X	3	DMA transfer size for receive and transmit (in 16 byte blocks)
Q	168	Number of references in each queue
B	1 or 6	Number of references accessed at one time by the host software
N	84	Maximum number of references in TDR Ready queue
PRIORITY	0	Receive channel priority
PRIORITYB	0	Transmit channel priority

The average PCI bus utilization for 84 unchannelised T1/J1 links is shown in Figure 4 as a function of the packet length. Two of the curves are for zero RAM latency, and the third includes RAM latency with a ratio of 1 read latency cycle for every 7 write latency cycles ($r = 1$ and $w = 7$).

Figure 4 – Utilization with 84 Unchannelised T1/J1 Links



The PCI bus utilization graph indicates that:

- Utilization is strongly dependent on the packet length. For small packets, the utilization increases dramatically.
- Utilization improves as B, the number of references cached in software before being read from, or written to, a FREEDM-84P672 queue, decreases.
- The system design (i.e. – the number of references accessed by the host software at one time, the average packet length, and the host RAM latency) plays a major role in determining the number of FREEDM-84P672s that can be supported.

The following table shows the maximum tolerable channel latencies and the worst case PCI bus latencies, assuming zero RAM latency. Since the worst case PCI bus latencies are less than the tolerable channel latencies in all cases, the graph in Figure 4 should be used to estimate the number of FREEDM-84P672s supported on a PCI bus. All times are in msec.

Latency Estimate	FREEDM-84P672s on PCI bus			
	1	2	3	4
Receive $L_{Channel}$	1.74	1.74	1.74	1.74
Priority Transmit $L_{Channel}$	1.99	1.99	1.99	1.99
Receive L_{PCI}	0.069	0.14	0.21	0.28
Transmit L_{PCI}	0.14	0.20	0.27	0.34
Expedited Priority Transmit L_{PCI}	0.067	0.13	0.20	0.27

6.2 63 Unchannelised E1 Links

Up to 63 unchannelised E1 links can be physically connected to a FREEDM-84P672 device. All three of the SPEs need to be configured to carry 21 E1 links each. In this case, 63 bi-directional HDLC channels are provisioned, each assigned to an unchannelised E1 link.

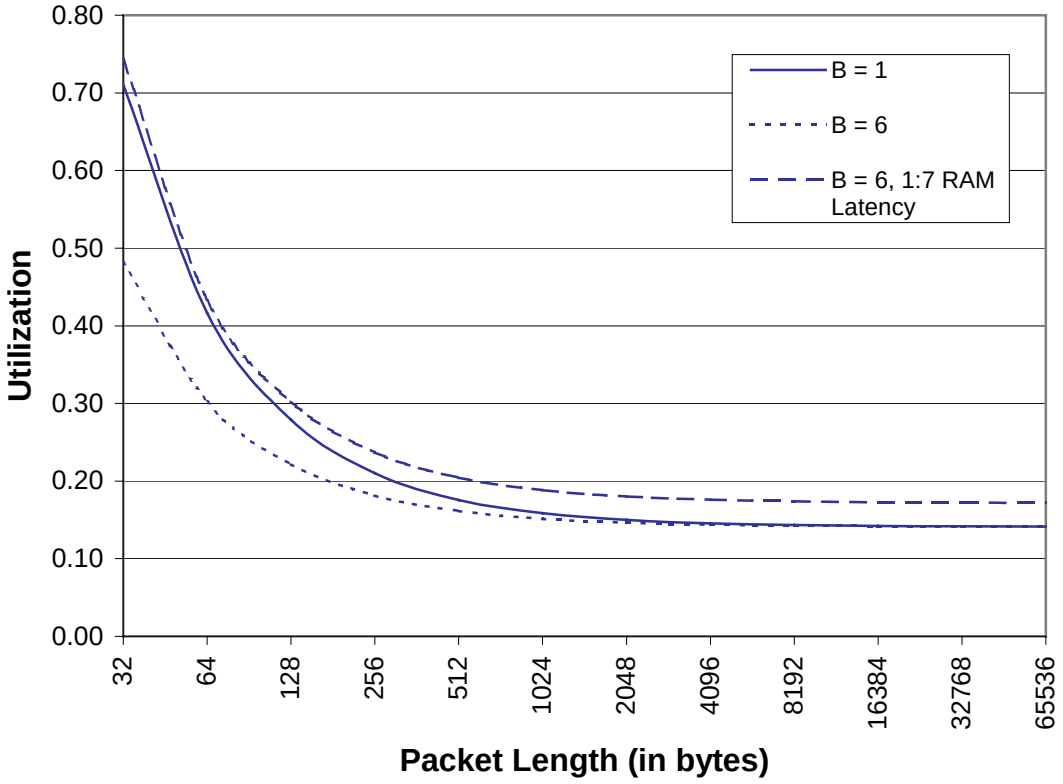
The estimates for the unchannelised E1 application are based on the following configuration:

Configurable Parameter	Value	Description
f_{PCI}	66 MHz	PCI bus clock frequency
$f_{Channel}$	2.048 Mbps	Channel data rate
$_{Channel}$	63	Number of HDLC channels provisioned
F	32	Size of channel FIFO assigned to each channel (in 16 byte blocks)
S	8	Start transmission level (in free blocks)
E	16	Starving trigger level for expedited channels (in free blocks)
X	4	DMA transfer size for receive and transmit (in 16 byte blocks)
Q	126	Number of references in each queue
B	1 or 6	Number of references accessed at one time by the host software

Configurable Parameter	Value	Description
N	63	Maximum number of references in TDR Ready queue
PRIORITY	0	Receive channel priority
PRIORITYB	0	Transmit channel priority

The average PCI bus utilization for 63 unchannelised E1 links is shown in Figure 5 as a function of the packet length. Two of the curves are for zero RAM latency, and the third includes RAM latency with a ratio of 1 read latency cycle for every 7 write latency cycles ($r = 1$ and $w = 7$).

Figure 5 – Utilization with 63 Unchannelised E1 Links



The PCI bus utilization graph indicates that:

- Utilization is strongly dependent on the packet length. For small packets, the utilization increases dramatically.

- Utilization improves as B, the number of references cached in software before being read from, or written to, a FREEDM-84P672 queue, decreases.

- The system design (i.e. – the number of references accessed by the host software at one time, the average packet length, and the host RAM latency) plays a major role in determining the number of FREEDM-84P672s that can be supported.

The following table shows the maximum tolerable channel latencies and the worst case PCI bus latencies, assuming zero RAM latency. Since the worst case PCI bus latencies are less than the tolerable channel latencies in all cases, the graph in Figure 5 should be used to estimate the number of FREEDM-84P672s supported on a PCI bus. All times are in msec.

Latency Estimate	FREEDM-84P672s on PCI bus			
	1	2	3	4
Receive $L_{Channel}$	1.75	1.75	1.75	1.75
Priority Transmit $L_{Channel}$	2.0	2.0	2.0	2.0
Receive L_{PCI}	0.064	0.13	0.19	0.26
Transmit L_{PCI}	0.13	0.19	0.25	0.31
Expedited Priority Transmit L_{PCI}	0.063	0.12	0.19	0.25

6.3 672 Channels at 192 Kbps Each

This configuration involves a mix of channelised T1/J1 or E1 links, where three time-slots are assigned to each HDLC channel of the FREEDM-84P672. There are 672 bi-directional HDLC channels assigned in total, each running at 192 Kbps (3 × DS-0 rate).

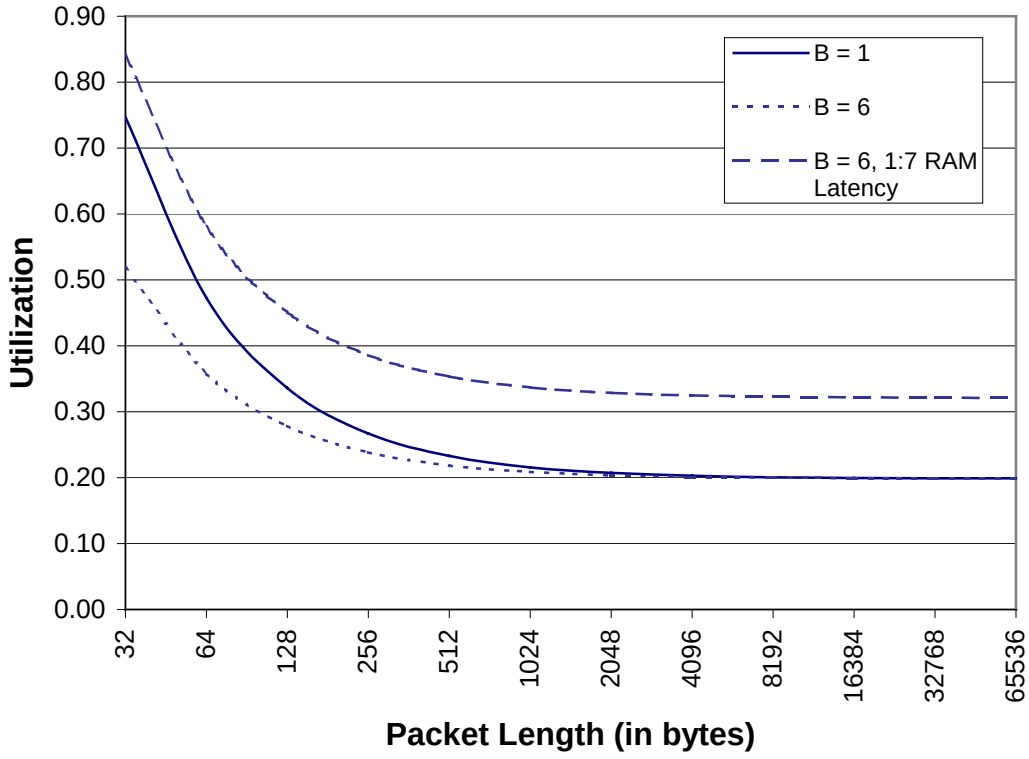
The estimates for this application are based on the following configuration:

Configurable Parameter	Value	Description
f_{PCI}	66 MHz	PCI bus clock frequency
$f_{Channel}$	192 Kbps	Channel data rate

Configurable Parameter	Value	Description
$Channel$	672	Number of HDLC channels provisioned
F	3	Size of channel FIFO assigned to each channel (in 16 byte blocks)
S	1	Start transmission level (in free blocks)
E	2	Starving trigger level for expedited channels (in free blocks)
X	1	DMA transfer size for receive and transmit (in 16 byte blocks)
Q	1344	Number of references in each queue
B	1 or 6	Number of references accessed at one time by the host software
N	672	Maximum number of references in TDR Ready queue
PRIORITY	0	Receive channel priority
PRIORITYB	0	Transmit channel priority

The average PCI bus utilization for 672 channels at 192 Kbps each is shown in Figure 6 as a function of the packet length. Two of the curves are for zero RAM latency, and the third includes RAM latency with a ratio of 1 read latency cycle for every 7 write latency cycles ($r = 1$ and $w = 7$).

Figure 6 – Utilization with 672 Channels at 192 Kbps Each



The PCI bus utilization graph indicates that:

- Utilization is strongly dependent on the packet length. For small packets, the utilization increases dramatically.
- Utilization improves as B, the number of references cached in software before being read from, or written to, a FREEDM-84P672 queue, decreases.
- The system design (i.e. – the number of references accessed by the host software at one time, the average packet length, and the host RAM latency) plays a major role in determining the number of FREEDM-84P672s that can be supported.

The following table shows the maximum tolerable channel latencies and the worst case PCI bus latencies, assuming zero RAM latency. Since the worst case PCI bus latencies are less than the tolerable channel latencies in all cases, the graph in Figure 6 should be used to estimate the number of FREEDM-84P672s supported on a PCI bus. All times are in msec.

Latency Estimate	FREEDM-84P672s on PCI bus			
	1	2	3	4
Receive $L_{Channel}$	1.33	1.33	1.33	1.33
Priority Transmit $L_{Channel}$	2.0	2.0	2.0	2.0
Receive L_{PCI}	0.30	0.60	0.91	1.21
Transmit L_{PCI}	0.59	0.87	1.15	1.43
Expedited Priority Transmit L_{PCI}	0.28	0.56	0.84	1.11

6.4 One Unchannelised DS-3 Link and 128 Kbps Channelised T1/J1 Links

In this configuration, one bi-directional HDLC channel is assigned to one unchannelised DS-3 link with a rate of 44.736 Mbps, and 448 bi-directional HDLC channels are assigned to two time-slots (channel data rate of 128 Kbps) each from channelised T1/J1 links.

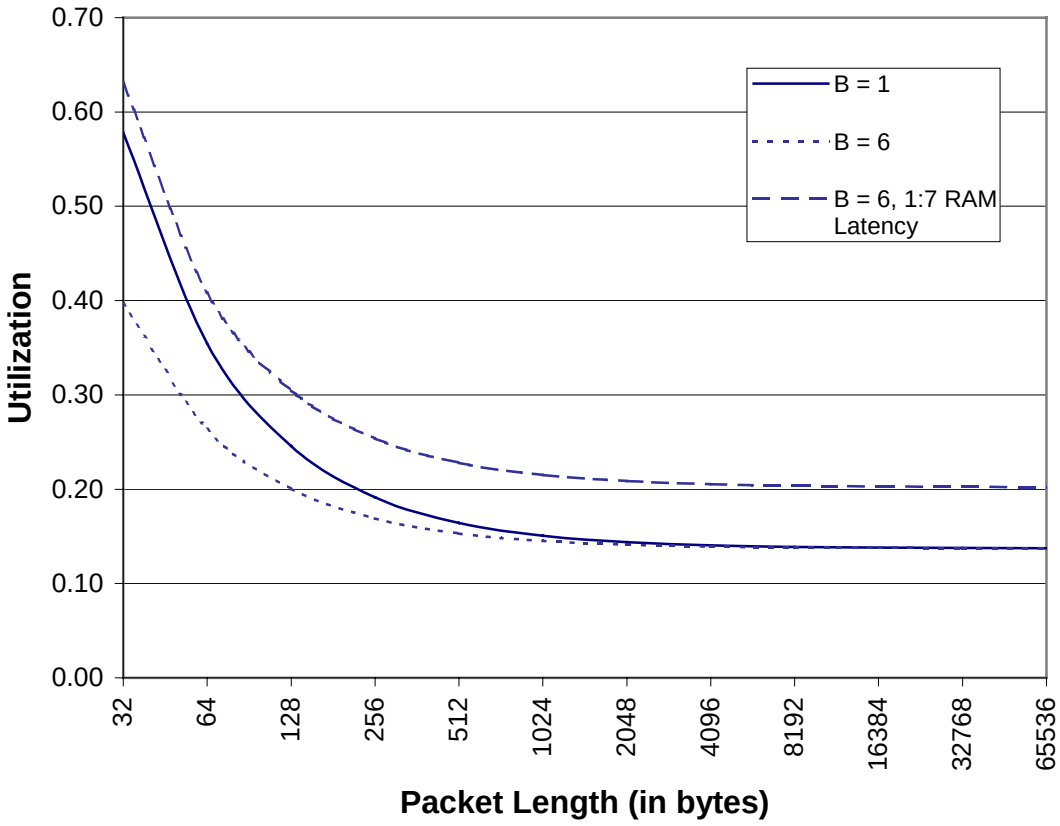
As shown below with the PRIORITY and PRIORITYB bits, the HDLC channel assigned to the DS-3 link is the high priority channel relative to the 448 channels with a data rate of 128 Kbps. The estimates for this application are based on the following configuration:

Configurable Parameter	Value for DS-3 Link	Value for 128 Kbps	Description
f_{PCI}	66 MHz	66 MHz	PCI bus clock frequency
$f_{Channel}$	44.736 Mbps	128 Kbps	Channel data rate
$_{Channel}$	1	448	Number of HDLC channels provisioned
F	682	3	Size of channel FIFO assigned to each channel (in 16 byte blocks)
S	192	1	Start transmission level (in free blocks)
E	384	2	Starving trigger level for expedited channels (in free blocks)
X	4	1	DMA transfer size for receive and transmit (in 16 byte blocks)
Q	512	512	Number of references in each queue

Configurable Parameter	Value for DS-3 Link	Value for 128 Kbps	Description
B	1 or 6	1 or 6	Number of references accessed at one time by the host software
N	128	128	Maximum number of references in TDR Ready queue
PRIORITY	1	0	Receive channel priority
PRIORITYB	0	1	Transmit channel priority

The average PCI bus utilization for one unchannelised DS-3 link and 448 channels at 128 Kbps each is shown in Figure 7 as a function of the packet length. Two of the curves are for zero RAM latency, and the third includes RAM latency with a ratio of 1 read latency cycle for every 7 write latency cycles ($r = 1$ and $w = 7$).

Figure 7 – Utilization with One DS-3 Link and 448 Channels at 128 Kbps



The PCI bus utilization graph indicates that:

- Utilization is strongly dependent on the packet length. For small packets, the utilization increases dramatically.
- Utilization improves as B, the number of references cached in software before being read from, or written to, a FREEDM-84P672 queue, decreases.
- The system design (i.e. – the number of references accessed by the host software at one time, the average packet length, and the host RAM latency) plays a major role in determining the number of FREEDM-84P672s that can be supported.

The following table shows the maximum tolerable channel latencies and the worst case PCI bus latencies for the channel assigned to the DS-3 link, assuming zero RAM latency. All times are in msec.

Latency Estimate	**FREEDM-84P672s on PCI bus**			
	1	**2**	**3**	**4**
Receive $L_{Channel}$	1.94	1.94	1.94	1.94
Priority Transmit $L_{Channel}$	1.95	1.95	1.95	1.95
Priority Receive L_{PCI}	0.00091	0.0022	0.0034	0.0046
Transmit L_{PCI}	0.49	0.86	1.22	1.58
Expedited Priority Transmit L_{PCI}	0.00086	0.0022	0.0031	0.0040

The following table shows the maximum tolerable channel latencies and the worst case PCI bus latencies for a channel assigned to two time-slots from the T1/J1 links, assuming zero RAM latency. All times are in msec.

Latency Estimate	**FREEDM-84P672s on PCI bus**			
	1	**2**	**3**	**4**
Receive $L_{Channel}$	2.0	2.0	2.0	2.0
Transmit $L_{Channel}$	2.0	2.0	2.0	2.0
Receive L_{PCI}	0.38	0.76	1.14	1.52
Transmit L_{PCI}	0.42	0.79	1.15	1.51

Since the worst case PCI bus latencies are less than the tolerable channel latencies in all of the above cases, the graph in Figure 7 should be used to estimate the number of FREEDM-84P672s supported on a PCI bus.

7 CONCLUSIONS AND RECOMMENDATIONS

This document provides insight into the performance of a FREEDM-84P672 on the PCI bus. A summary of what has been learned is as follows:

- Overutilization of the PCI bus can lead to receive packet overrun or transmit packet underrun. A designer needs to ensure that the rate of the links attached to a FREEDM-84P672 does not exceed the capabilities of the PCI bus and the FREEDM-84P672. Registers of a FREEDM-84P672 need to be configured to optimize bus utilization for the intended application.

- The application can greatly affect the PCI bus utilization. Transfers involving smaller packets tend to increase the bus utilization. Since the data rate, rather than the link rate, affects bus utilization, and since the estimates have assumed that links are fully saturated with data, the utilization will be lower for LAN applications that have bursty traffic.

- The estimates have shown that RAM latency of the host has an adverse affect on the PCI bus utilization, especially with small packets.

- The host software should be designed to reduce utilization of the bus by accessing the FREEDM-84P672 registers less frequently than with every packet. This can help to reduce the utilization during transfers of small packets.

- The host software should configure the size of queues so that they are large enough to handle the number of channels provisioned. The size should be at least 32 to ensure that there are no adverse effects on the bus utilization, and generally must be larger than the number of channels provisioned. The TDR Ready queue should never be too large as the number of references in this queue will adversely affect the bus latency for transmit channels.

- A FREEDM-84P672 channel can be provisioned to optimize use of the bus. The configurable parameters have been presented.

- The bus utilization and latencies should be calculated to ensure a configuration does not lead to transmit underrun or receive overrun. Equations for estimating these have been presented.

- When an application has multiple channel rates, the metric $f_{Channel}/F$ can be used to identify which channels should be provisioned with receive priority, or inhibited from making expedited requests for transmit data.

APPENDIX A – DIFFERENCES BETWEEN FREEDM-32 AND FREEDM-84P672

The following table outlines the parameter differences between the FREEDM-32 and the FREEDM-84P672 that are relevant to PCI bus utilization and latency analysis:

Parameter	FREEDM-32	FREEDM-84P672
Number of high speed links with arbitrary link rate of up to 52 MHz (when SYSCLK = 40 MHz)	2	3
Number of HDLC channels	128	672
Size of channel FIFOs	8 Kbyte	32 Kbyte
Number of 16 byte partial packet buffer blocks in channel FIFO	512	2048
Range of XFER (indirect channel transfer) size in 16 byte blocks	1 – 8	1 – 16
Maximum PCI burst size	128 bytes	256 bytes
PCI bus clock frequency	0 - 33 MHz	25 - 66 MHz
Maximum number of addressable descriptors	16,384	32,768

Note: For the changes in the Receive Packet Descriptor and the Transmit Descriptor, please refer to the appendix sections of the FREEDM-84P672 Programmer's Guide [3].

CONTACTING PMC-SIERRA, INC.

PMC-Sierra, Inc.
105-8555 Baxter Place Burnaby, BC
Canada V5A 4V7

Tel: (604) 415-6000

Fax: (604) 415-6200

Document Information: document@pmc-sierra.com
Corporate Information: info@pmc-sierra.com
Application Information: apps@pmc-sierra.com
(604) 415-4533
Web Site: http://www.pmc-sierra.com

PMC-990863 (P1) Issue date: July 1999