PM7364

FREEDM

FREEDM PCI BUS UTILIZATION AND LATENCY ANALYSIS

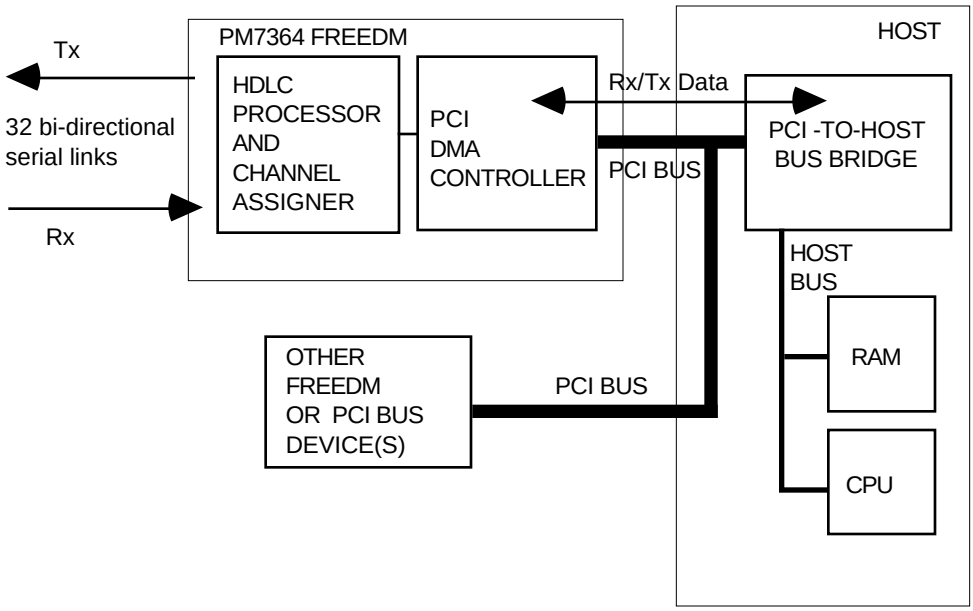
Issue 1: February, 1997

CONTENTS

OVERVIEW

The FREEDM provides HDLC processing of up to 32 bi-directional serial links in Frame Relay, PPP and FUNI applications. It interfaces directly to a host via a 32 bit PCI local bus interface [1], across which data is read or written by the FREEDM's DMA controller. The bus allows for the transfer of data between the host RAM and the FREEDM, at a high rate, without intervention by the host CPU.

Figure 1. System Block Diagram Showing the Data Path



The flow of data is shown for a typical system in Figure 1. The receive data from a serial link is mapped to one or more HDLC channels by the Receive Channel Assigner (RCAS). Each channel is independently processed, 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 (RMAC), across the PCI bus. Upon completion of a packet the FREEDM notifies the host CPU by placing a reference to the packet in the RPDR Ready Queue, by setting the third byte of the RPDR to zero, and optionally by interrupt. The CPU host must replenish the RPDR Small Free Queue, and the RPDR Large Free Queue, which identifies available host RAM addresses where the FREEDM 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 serving other channels, or insufficient replenishment of the RPDR Small (or Large) Free Queue.

The host CPU prepares transmit packets in host RAM and places a reference to each packet in the TDR Ready Queue. After the FREEDM is signaled that this queue is no longer empty, the Transmit DMA Controller (TMAC) burst reads the packet data across the bus and deposits this data into the Transmit Channel FIFO. The Transmit HDLC Processor (THDL) performs HDLC processing of the data and the Transmit Channel Assigner (TCAS) maps the data to one of the 32 transmit serial links. The FREEDM 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 transmit of a packet, when data must be driven on the serial link but the Transmit Channel FIFO is empty. In this case, the FREEDM was unable to read data across the bus in time. This condition can be created by excessively long bus access latency, or bandwidth starvation as the result of FREEDM serving other channels.

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:

- To maximize the PCI bus efficiency. Thus allowing the FREEDM 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's 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 the system performs at the aggregate link rate, and no bandwidth on the serial links is lost.

The purpose of this application note is to outline the factors which affect PCI bus utilization and thereby to provide enough information for a designer to estimate the most optimal configuration of the FREEDM for the intended application. The calculations of PCI bus performance which are presented here are believed to be accurate although the actual bus performance may differ. The objectives are to define limits of performance and to provide insight into optimizing the configuration of a FREEDM for the application.

Known Design Constraints

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

- The FREEDM supports a single bi-directional HDLC channel on an unchannelized arbitrary rate link of up to 45 MHz when SYSCLK is at 25MHz, and up to 52 MHz when SYSCLK is at 33 MHz.

- The FREEDM supports up to 32 bi-directional HDLC channels, each assigned to an unchannelized arbitrary rate link of up to 10 MHz.

- The FREEDM supports up to 128 bi-directional HDLC channels assigned from a maximum of 32 channelised T1 or E1 links, 32 unchannelized links, or a combination of these.

- The FREEDM supports up to 128 bi-directional channels assigned to a maximum of 32 channelised T1 or E1 links. The number of time-slots assigned to an HDLC channel is programmable from 1 to 24 (for T1) and from 1 to 31 (for E1).

- The FREEDM supports a mix of channelised and unchannelised links, subject to the constraint of a maximum of 128 channels, and a maximum aggregate link clock rate of 64 MHz in each direction.

- The FREEDM interfaces directly to a PCI local bus which is revision 2.1 compliant, and has a 32 bit data path. This places a limit on the maximum number of FREEDM devices interfaced to a PCI bus and on the aggregate data rate of all the serial links which can be handled by the bus.

Assumptions

The bus performance is affected by the system design choices made in integrating the FREEDM with an embedded host. The following list of design choices are 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 these choices are outside the scope of this document, and for the purposes of this analysis the following list of assumptions are 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 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.
- There are one or more FREEDM devices and the host CPU which utilize bus cycles. No other PCI devices are present.
- The serial link is fully satured 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.

CALCULATION OF PCI BUS CYCLES

There are four PCI bus transactions that are used to transfer data between the FREEDM, the host RAM and the host CPU. The PCI bus utilization during transmit and receive 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, RAM access latency cycles, data 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 register**: This PCI read transaction has 7 bus cycles (for PCICLK=SYSCLK= 33MHz). This value is based on simulation of the FREEDM and is expected to increase with a slower SYSCLK.

- **Write FREEDM register**: This PCI write transaction has 7 bus cycles (for PCICLK=SYSCLK= 33MHz). This value is based on simulation of the FREEDM and is expected to increase with a slower SYSCLK.

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.

Transmit Packets

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:

- Host writes packets to transmit ready queue in blocks of up to B references, thereby writing the **TMAC Descriptor Reference Ready Queue Write** register every B'th reference, and reading the **TMAC Descriptor Reference Ready Queue Read** register approximately every Q'th reference. Here Q is the size of the queue and B is number of references in the block. This activity has approximately $7/B+7/Q$ bus cycles per packet.

- FREEDM reads one reference from the transmit ready queue. This activity has $3+1+r$ bus cycles per packet.

- FREEDM reads 3 DWORD's of transmit descriptor to determine the TCC (transmit channel code) as well as other information. This activity has $3+3+r$ bus cycles per packet.

- FREEDM chains the reference to the current list of references by writing 2 DWORDS into the transmit descriptor. This activity has $2+2+w$ bus cycles per packet.

- FREEDM reads 4 DWORDs of the transmit descriptor. This activity has $3+4+r$ bus cycles per descriptor.

- FREEDM writes the reference to transmit free queue. If the CACHE bit is enabled within the **TMAC Control** register then this activity is completed once every 6 references, and has $(2+w+6)/6$ bus cycles per descriptor.

- Host reads references from the transmit free queue in blocks of up to B references, thereby writing the **TMAC Descriptor Reference Free Queue Read** register approximately every B'th reference, and reading the **TMAC Descriptor Reference Free Queue Write** register every Q'th reference. Here Q is the size of the queue and B is number of references in the block. This activity has approximately $7/B+7/Q$ bus cycles.

- FREEDM reads the transmit descriptor buffer. This may require multiple burst read transactions since the FREEDM can only DMA XFER blocks of data per transaction. This activity has $\lceil P/4 \rceil + (3+r) \cdot \lceil P/(16 \cdot X) \rceil$ bus cycles.[1] Here P is the packet length in bytes, and X is the XFER size in 16 byte blocks.

Receive Packets

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.

[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.

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

- Host writes references to receive free queue in blocks of up to B references, thereby writing the **RMAC Packet Descriptor Reference Small Buffer Free Queue Write** register aproximately every B'th reference, and reading the **RMAC Packet Descriptor Reference Small Buffer Free Queue Read** register every Q'th reference. Here Q is the size of the queue and B is number of references in the block. This activity has $7/B + 7/Q$ bus cycles per packet.

- FREEDM writes reference to receive ready queue. This activity has $2 + 1 + w$ bus cycles per packet.

- Host reads references from receive ready queue in blocks of up to B references, thereby writing **RMAC Packet Descriptor Reference Ready Queue Read** register aproximately every B'th reference queued, and reading the **RMAC Packet Descriptor Reference Ready Queue Write** register every Q'th reference. Here Q is the size of the queue and B is number of references in the block. This PCI transaction has $7/B + 7/Q$ bus cycles.

- FREEDM writes 2 DWORDs of receive descriptor. This activity has $2 + 2 + w$ bus cycles.

- FREEDM reads 4 DWORDs of receive descriptor. This activity has $3 + 4 + r$ bus cycles.

- FREEDM writes to the receive descriptor buffer. This may require multiple burst write transactions since the FREEDM can only DMA XFER blocks of data per transaction. This PCI transaction has $\lceil P/4 \rceil + (2 + w) \cdot \lceil P/(16 \cdot X) \rceil$ bus cycles. Here P is the packet length in bytes, and X is the XFER size in 16 byte blocks.

- FREEDM reads references from the receive small buffer queue. The SCACHE bit of the **RMAC Control** register is set such that 6 free references are read from the queue every 6'th reference. This PCI transaction has $(3 + r + 6)/6$ bus cycles.

Interrupt Service Overhead

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

It is assumed that the interrupt service routine processes each interrupt by reading/clearing the **Master Interrupt Status** register, and then storing the status within the software for deferred processing.

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

Average Number of Cycles Per Packet

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 the previous section on receive packets. The bus cycles per byte of packet received is expressed as

$$C_{Rx\ Byte} = \frac{1}{P}\left[15.5 + 14 \cdot \left(\frac{1}{B} + \frac{1}{Q}\right) + \left\lceil \frac{P}{4} \right\rceil + (2 + w) \cdot \left\lceil \frac{P}{X \cdot 16} \right\rceil + 1.17 \cdot r + 2 \cdot w\right]$$

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 the previous section on transmit packets. The bus cycles per byte of transmit packet is expressed as

$$C_{Tx\ Byte} = \frac{1}{P}\left[22.33 + 14 \cdot \left(\frac{1}{B} + \frac{1}{Q}\right) + \left\lceil \frac{P}{4} \right\rceil + (3 + r) \cdot \left\lceil \frac{P}{X \cdot 16} \right\rceil + 3 \cdot r + 1.17 \cdot w\right]$$

The total cycles to transmit and receive one byte of packet data is expressed as

$$C_{Byte} = \frac{1}{P}\left[37.83 + 28 \cdot \left(\frac{1}{B} + \frac{1}{Q}\right) + 2 \cdot \left\lceil \frac{P}{4} \right\rceil + (5 + r + w) \cdot \left\lceil \frac{P}{X \cdot 16} \right\rceil + 4.17 \cdot r + 3.17 \cdot w\right]$$

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

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 is bursty by nature, and the time interval must be sufficiently large. If too small a time interval is choosen, 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 choosen when there is little traffic on the bus the measured utilization could be much lower than the long term average.

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 transmit and receive is a function of the data rate for each channel provisioned on a FREEDM. It is expressed as follows,

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

where $f_{Channel}$ is the channel data rate in bits per second and C_{Byte} is the average number of cycles to DMA one transmit and one receive 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 devices on a bus is obtained by the summation over all provisioned channels, and on all FREEDM devices connected to the bus.

Configuration of the FREEDM channels and serial ports such that the average utilization does not exceed 1 (or 100%), based on this equation, does not guarantee the avoidance of underrun or overrun. These can only be guaranteed by ensuring each Channel FIFO is large enough to tolerate the bus latencies. This problem will be examined in a later section with calculations of bus latency.

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 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 Master Interrupt Status** register

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

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

Receive Prioritization

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

1) FREEDM accesses to the RPDRF Large Queue, or the RPDRF Small Queue. This transaction occurs whenever the internal cache of free buffers 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** register.

3) Transactions for remaining receive channels.

Each receive channel of the same priority is polled in a round robin basis for access to the bus, but there are two sequences of bus transactions that are atomic (ie each sequence of transactions cannot be pre-empted by receive transactions of another channel, although successive read-read or write-write transactions can be pre-empted by transmit transactions). The sequences are:

<u>Start of Packet Sequence</u>

- FREEDM reads upto 6 references from the Receive Packet Descriptor Reference Small Free Queue (or the Receive Packet Descriptor Reference Large Free Queue). This transaction only occurs once every sixth start of packet sequence.

- FREEDM reads 4 DWORD's of descriptor.

- FREEDM writes receive data into buffer of the descriptor.

End of Packet Sequence

- FREEDM writes final bytes of a packet to the receive buffer. This transaction writes upto XFER bytes to host RAM, where XFER is the maximum number of Channel FIFO blocks that is written per transaction, and is specified on provisioning of a channel.

- FREEDM writes 2 DWORDs of receive descriptor.

- FREEDM writes a receive packet reference to the Receive Packet Descriptor Reference Ready Queue.

Transmit Prioritization

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

1) FREEDM accesses to the Transmit Packet Descriptor Reference 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** register is set.

2) FREEDM accesses host RAM for a transmit channel which is not inhibited and where the Channel FIFO has reached the expedite level. These are provisioned channels with the PRIORITYB=0 within the **THDL Indirect Channel Data #2** register.

3) FREEDM accesses to the Transmit Packet Descriptor Reference 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 accesses packet data, descriptors, and queue references for any transmit channel which has enough space in it's Channel FIFO for the remaining bytes of a packet or the XFER size.

Each transmit channel is polled in a round robin basis for access to the bus, but there are two sequences of bus transactions that are atomic (ie each sequence of transactions cannot be pre-empted by transmit transactions of another channel, although successive read-read or write-write transactions can be pre-empted by receive transactions). The sequences are:

Chain Packets of TDR Ready Queue Sequence

- FREEDM reads transmit reference from TDR Ready Queue.

- FREEDM reads 3 DWORD's of the transmit descriptor.

- FREEDM writes 2 DWORDs of another transmit descriptor to link packets of a channel.

End of Packet Sequence

- FREEDM reads final bytes of a packet from the transmit buffer. This transaction reads upto XFER bytes from host RAM, where XFER is the maximum number of Channel FIFO blocks that is read per transaction, and is specified on provisioning of the transmit channel.

- FREEDM writes 6 transmit references to the Transmit Descriptor Free Queue if the cache of free references has reached 6.

- FREEDM reads 4 DWORDs of transmit descriptor of the following packet.

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 partial packet buffer blocks which are provisioned for the channel. Channels with a higher value of this metric could be provisioned with the PRIORITY bit set within the **RHDL Indirect Channel Data #2** register.

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 reset within the **THDL Indirect Channel Data #2** register. This will ensure that the bus access latency is reduced for channels where the Channel FIFO has reached the expedite level.

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 underrun or overrun.

Maximum Tolerable Channel Latency

The Channel FIFO of each channel must be serviced at least once within a time interval to prevent the occurance 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 when the receive channel has filled XFER 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 when the Channel FIFO has reached the start, or expedite level. The interval ends when the channel has read XFER 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 channel XFER size specified in blocks, S is the start transmit level specified in free blocks, E is the expedite DMA level specified in free blocks, and $f_{Channel}$ is the channel rate specified in bits per second. The start transmit level and the expedite transmit level are configured by the TRANS bit and the LEVEL[3:0] bits within the **THDL Indirect Channel Data #3** register as described in the longform datasheet[2].

To prevent an underrun or overrun 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.

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 that have equal or higher priority access to the PCI bus than the channel being evaluated for underrun or overrun. Utilization of bus cycles by the host is assumed negligable. The following sections provide equations for a variety of channel configurations.

Priority Receive Channel

This section provides an estimate of bus latency for a priority receive channel. This is a receive channel for which the PRIORITY bit is set. 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 registers is assumed negligable.

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 XFER blocks of data across the bus is not required for this calculation as the FREEDM double buffers the XFER 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 the other priority receive channels, and pre-emption of each sequence by transmit channel transactions.
- DMA of XFER blocks of transmit packet data between each of the receive channel transactions.

The equation which estimates the worst case bus latency is:

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

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

Expedited Transmit Channel

This section provides an estimate of bus latency for an expedited transmit channel. This is a transmit channel for which the PRIORITYB bit is zero, the Channel FIFO contains less than one packet of data, and the expedite 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 registers is assumed negligable.

The time interval is measured starting from when the channel requires bus access and ending when the XFER blocks of data has been transfered 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, and the channel was provisioned with PRIORITYB=1.
- completion of the end-of-packet sequence on all the other transmit channels for which PRIORITYB=0.
- DMA of XFER blocks of receive packet data between each of the transmit channel transactions.

The equation for this estimate is

$$L_{PCI} \cong \frac{1}{f_{PCI}} \cdot \left(20 + 8 \cdot X + 2 \cdot (r + w) + (8 + w) \cdot \left\lceil \left(\sum_{\text{Expedited Tx Channel}} 1/6 \right) - 5 \right\rceil + \sum_{\text{Expedited Tx Channel}} (12 + 4 \cdot X_{Rx} + 4 \cdot X_{Tx} + 2 \cdot r + w) \right)$$

where X_{Rx} is the largest value of XFER blocks of a receive channel, X_{Tx} is the XFER blocks of an expedited transmit channel, r is the read latency cycles, and w is the write latency cycles. The summation is performed over all expedited transmit channels.

Receive Channel

This section provides an estimate of bus latency for a FREEDM 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 registers are assumed negligable.

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 XFER blocks of data across the bus is not required for this calculation as the FREEDM double buffers the XFER 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 XFER blocks of packet data.
- completion of the start-of-packet sequence on all the other receive channels, and pre-emption of these transactions by transmit channel transactions.
- DMA of XFER blocks of transmit packet data between each of the receive channel transactions.

The equation which estimates the worst case bus latency is:

$$L_{PCI} = \frac{1}{f_{PCI}} \cdot \left(3 + 4 \cdot X_{Tx} + r + \left(4 \cdot X_{Tx} + 12 + 2 \cdot r\right) \cdot \left\lceil \frac{1}{f_{Channel}} \cdot \sum_{Other\ Rx} f_{Channel} \Big/ 6 \right\rceil \right.$$

$$\left. + \frac{1}{f_{Channel}} \cdot \sum_{Other\ Rx} f_{Channel} \cdot \left(15 + 8 \cdot X_{Tx} + 4 \cdot X_{Rx} + w + 3 \cdot r\right) \right)$$

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

Transmit Channel

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

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

- chaining and linking of packets read by the FREEDM from the TDR ready queue.
- end-of-packet processing on all transmit channels.
- DMA of XFER blocks of receive packet data between all of the transmit channel transactions.

The equation for this estimate is

$$L_{PCI} \cong \frac{1}{f_{PCI}} \cdot \left(\left(10 + 4 \cdot X_{Rx} + 2 \cdot w\right) \cdot \left\lceil \frac{1}{f_{Channel}} \cdot \sum_{Tx\,Channel} f_{Channel} \Big/ 6 \right\rceil + \frac{1}{f_{Channel}} \cdot \sum_{Tx\,Channel} f_{Channel} \cdot \left(14 + 4 \cdot X_{Tx} + 8 \cdot X_{Rx} + 2 \cdot r + 2 \cdot w\right) \right) + \frac{N}{f_{PCI}} \cdot \left(20 + 12 \cdot X_{Rx} + 2 \cdot r + 4 \cdot w\right)$$

where X_{Rx} is the largest XFER blocks of a receive channel, X_{Tx} is the XFER blocks of a transmit channel, 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 transmit channels provisioned.

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

PCI PEFORMANCE FOR TYPICAL APPLICATIONS

High Speed Serial Interface (HSSI)

HSSI is a 52 Mbps link that must be physically connected to either port 0, 1 or 2 of a FREEDM device. Only one bi-directional HSSI link is attached, thereby ensuring the maximum aggregate link rate of 64 MHz is not exceeded. The designer could choose to use multiple FREEDM devices on a PCI bus where more than one HSSI link is required. Only one channel of each FREEDM is provisioned.

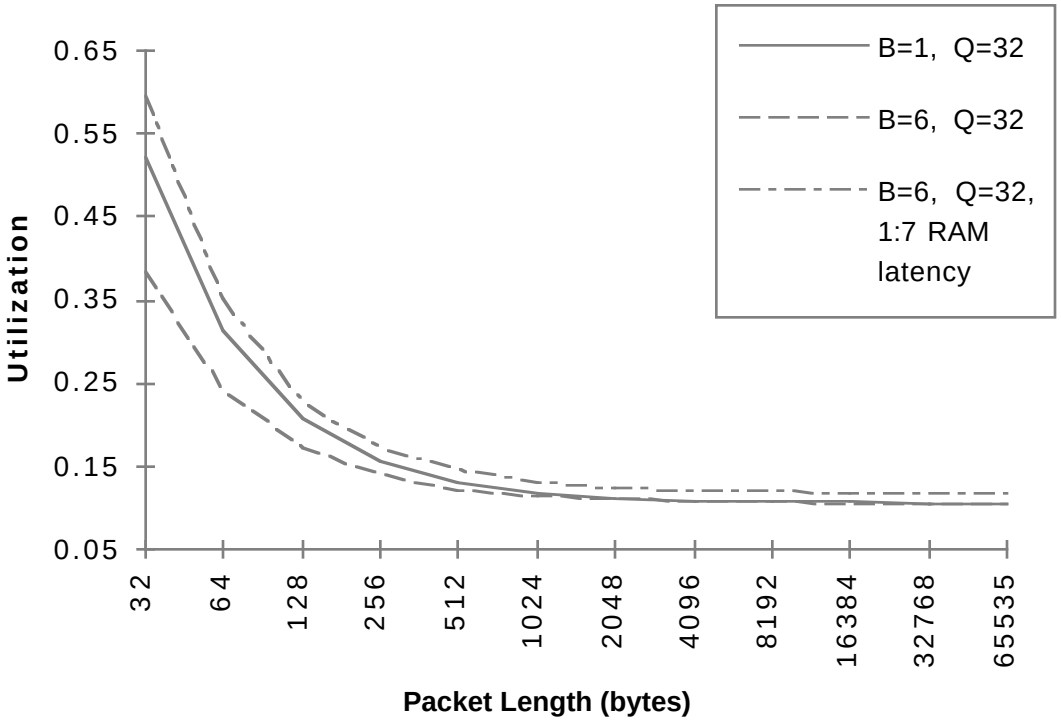
The estimates for the HSSI application are based on the following configuration:

Configurable Parameter	Value	Description
F	512 blocks	size of buffer assigned to channel
X	8 blocks	DMA transfer size for transmit and receive
S	256 blocks	transmit start level of buffer (free blocks)
E	384 blocks	transmit expedite level of buffer (free blocks)
Q	32	size of each queue
B	6	number of references accessed at a time by the host software
N	32	maximum references in TDR ready queue
f_{PCI}	33 MHz	PCI bus clock frequency
$f_{Channel}$	52 MHz	Channel data rate
PRIORITY	X	receive channel priority not relevant
PRIORITYB	X	transmit channel priority not relevant

The average PCI bus utilization for one HSSI is shown in figure 3 as a function of the packet length. Two of the curves are for zero RAM latency, the third includes RAM latency with a ratio of 1:7 for read and write latency. The figure shows that utilization improves with B- the number of references cached in software before being read from, or written to, a FREEDM queue.

The PCI bus utilization is strongly dependant on the packet length. For small packets the utilization increases dramatically.

Figure 3. Utilization of a FREEDM With One HSSI link



The utilization graph indicates that multiple FREEDM devices, each with one HSSI channel, could be handled on a PCI bus. It also indicates that the system design (ie the number of references accessed at a time by the host software, the average packet length, and the host RAM latency) plays a major role in determining exactly how many HSSI channels, or FREEDM devices, can be supported.

Table 1 below 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 the graph of figure 3 should be used to estimate the number of FREEDM's supported on a PCI bus.

Table 1. Latency Estimates For HSSI

FREEDM's on bus	1	2	3	4	8
receive $L_{Channel}$ (msec)	1.24	1.24	1.24	1.24	1.24
transmit $L_{Channel}$ (msec)	0.94	0.94	0.94	0.94	0.94
receive L_{PCI} (msec)	0.001	0.0058	0.0091	0.013	0.027
transmit L_{PCI} (msec)	0.1	0.12	0.13	0.13	0.14
expedited transmit L_{PCI} (msec)	0.005	0.0074	0.0097	0.012	0.021

Unchannelized E1

Up to 32 unchannelized E1 links may be physically connected to a FREEDM device. In this case there are 32 FREEDM channels provisioned.

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

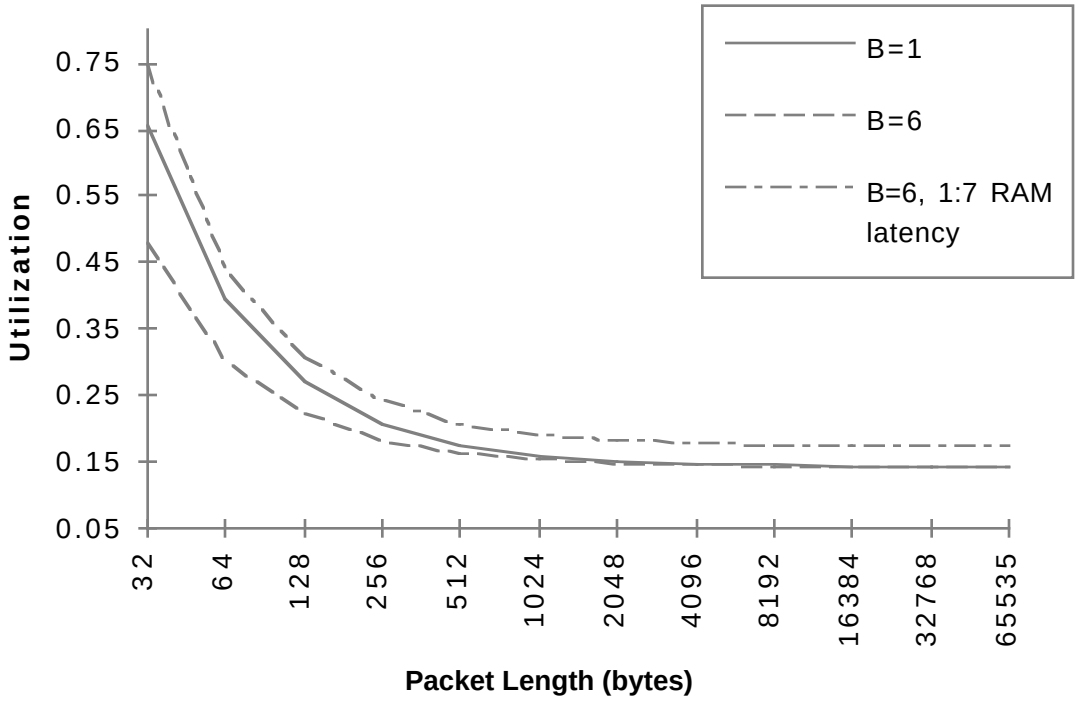
Configurable Parameter	Value	Description
F	16 blocks	size of buffer assigned to channel
X	4 blocks	DMA transfer size for transmit and receive
S	4 blocks	transmit start level of buffer (free blocks)
E	8 blocks	transmit expedite level of buffer (free blocks)
Q	64	size of each queue
B	6	number of references accessed at a time by the host software
N	32	maximum references in TDR ready queue
f_{PCI}	33 MHz	PCI bus clock frequency
$f_{Channel}$	2.048 Mbps	Channel data rate
PRIORITY	0	receive channel is not high priority

PRIORITYB	0	transmit channel is not inhibited

The average PCI bus utilization for one FREEDM is shown in figure 4 as a function of the packet length. Two of the curves are for zero RAM latency, the third includes RAM latency with a ratio of 1:7 for read and write latency states. The figure shows that utilization improves with B- the number of references cached in software before being read from, or written to, a FREEDM queue.

The PCI bus utilization is strongly dependant on the packet length. For small packets the utilization increases dramatically.

Figure 4. Utilization of a FREEDM With 32 Unchannelized E1 Links



The utilization graph indicates that multiple FREEDM devices, each with 32 E1 channels, could be handled on a PCI bus. It also indicates that the system design (ie the number of references accessed at a time by the host software, the average packet length, and the host RAM latency) plays a major role in determining exactly how many FREEDM devices can be supported.

Table 2 below 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 the average utilization should be used to determine how many FREEDM devices can be supported.

Table 2. Latency Estimates With 32 E1 Channels Per FREEDM

FREEDM's on bus	1	2	3	4	8
receive $L_{Channel}$ (msec)	0.75	0.75	0.75	0.75	0.75
transmit $L_{Channel}$ (msec)	1.0	1.0	1.0	1.0	1.0
receive L_{PCI} (msec)	0.065	0.13	0.20	0.26	0.52
transmit L_{PCI} (msec)	0.13	0.19	0.26	0.32	0.58
expedited transmit L_{PCI} (msec)	0.044	0.087	0.13	0.17	0.34

Unchannelized T1

Up to 32 unchannelized T1 links may be physically connected to a FREEDM device. In this case there are 32 FREEDM channels provisioned.

The estimates for the Unchannelized T1 application are based on the following configuration:

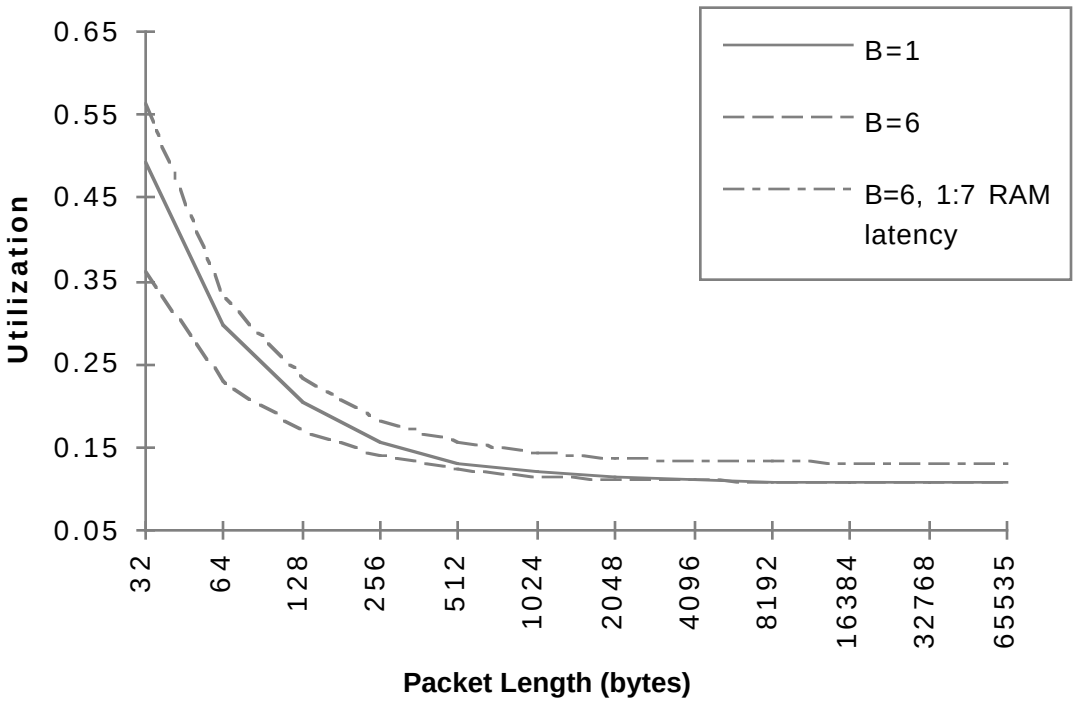
Configurable Parameter	Value	Description
F	16 blocks	size of buffer assigned to channel
X	4 blocks	DMA transfer size for transmit and receive
S	4 blocks	transmit start level of buffer (free blocks)
E	8 blocks	transmit expedite level of buffer (free blocks)
Q	64	size of each queue
B	6	number of references accessed at a time by the host software
N	32	maximum references in TDR ready queue

f_{PCI}	33 MHz	PCI bus clock frequency
$f_{Channel}$	1.544 Mbps	Channel data rate
PRIORITY	0	receive channel is not high priority
PRIORITYB	0	transmit channel is not inhibited

The average PCI bus utilization for one FREEDM is shown in figure 5 as a function of the packet length. Two of the curves are for zero RAM latency, the third includes RAM latency with a ratio of 1:7 for read and write latency. The figure shows that utilization improves with B- the number of references cached in software before being read from, or written to, a FREEDM queue.

The PCI bus utilization is strongly dependant on the packet length. For small packets the utilization increases dramatically.

Figure 5. Utilization of a FREEDM With 32 Unchannelized T1 Links



The utilization graph indicates that multiple FREEDM devices, each with 32 T1 channels, could be handled on a PCI bus. It also indicates that the system design (ie the number of references accessed at a time by the host software, the average packet length, and the host RAM latency) plays a major role in determining exactly how many FREEDM devices can be supported.

Table 3 below 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 the average utilization should be used to determine how many FREEDM devices can be supported.

Table 3. Latency Estimates With 32 T1 Channels Per FREEDM

FREEDM's on bus	1	2	3	4	8
receive $L_{Channel}$ (msec)	1.3	1.3	1.3	1.3	1.3
transmit $L_{Channel}$ (msec)	0.99	0.99	0.99	0.99	0.99
receive L_{PCI} (msec)	0.065	0.13	0.20	0.26	0.52
transmit L_{PCI} (msec)	0.13	0.19	0.26	0.32	0.58
expedited transmit L_{PCI} (msec)	0.044	0.087	0.13	0.17	0.34

64 Kbps Channelized E1 or T1

In this configuration any combination of T1 or E1 links, where some or all of the time-slots within a link are assigned to FREEDM channels, is possible. There are a total of 128 T1 and/or E1 time-slots mapped to FREEDM channels, for a total of 128 channels.

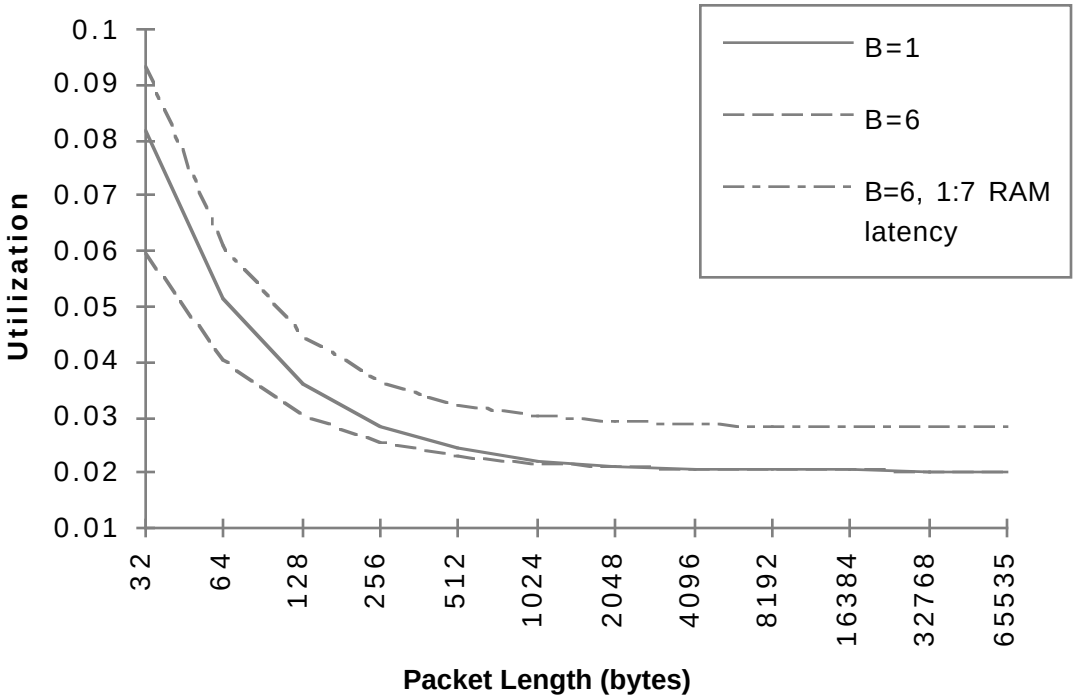
The estimates for the 64Kbps application are based on the following configuration:

Configurable Parameter	Value	Description
F	4 blocks	size of buffer assigned to channel
X	2 blocks	DMA transfer size for transmit and receive
S	1 blocks	transmit start level of buffer (free blocks)
E	2 blocks	transmit expedite level of buffer (free blocks)
Q	256	size of each queue
B	6	number of references accessed at a time by the host software
N	128	maximum references in TDR ready queue
f_{PCI}	33 MHz	PCI bus clock frequency
$f_{Channel}$	64 Kbps	Channel data rate
PRIORITY	0	receive channel is not high priority
PRIORITYB	0	transmit channel is not inhibited

The average PCI bus utilization for one FREEDM is shown in figure 6 as a function of the packet length. Two of the curves are for zero RAM latency, the third includes RAM latency with a ratio of 1:7 for read and write latency. The figure shows that utilization improves with B- the number of references cached in software before being read from, or written to, a FREEDM queue.

The PCI bus utilization is strongly dependant on the packet length. For small packets the utilization increases dramatically.

Figure 6. Utilization of a FREEDM With 128 Channels of 64 Kbps Rate



The utilization graph indicates that more than 10 FREEDM devices could be supported. This value is too great for the loading constraints of the PCI bus. The PCI local bus specification only allows up to 4 devices, and the Compact PCI specification[3] allows up to 8 devices on a PCI bus.

Table 4 below shows the maximum tolerable channel latencies and the worst case PCI bus latencies, assuming zero RAM latency. These values in combination with the utilization graph indicates that the maximum number of FREEDM devices on a compact PCI bus can be supported.

Table 4. Latency Estimates With 128 64Kbps Channels Per FREEDM

FREEDM's on bus	1	2	3	4	8
receive $L_{Channel}$ (msec)	4.0	4.0	4.0	4.0	4.0
transmit $L_{Channel}$ (msec)	8.0	8.0	8.0	8.0	8.0
receive L_{PCI} (msec)	0.16	0.33	0.49	0.66	1.3
transmit L_{PCI} (msec)	0.33	0.48	0.65	0.81	1.44
expedited transmit L_{PCI} (msec)	0.11	0.22	0.33	0.44	0.87

384 Kbps Channelized T1

Up to 128 channels can be configured on a FREEDM by assigning a number of time-slots to each channel. For this example there are 6 time-slots assigned to each channel for a total of 128 channels, each with a channel rate of 384 Kbps. All 32 ports of the FREEDM are attached to T1 links.

The estimates for the 384Kbps T1 application are based on the following configuration:

Configurable Parameter	Value	Description
F	4 blocks	size of buffer assigned to channel
X	2 blocks	DMA transfer size for transmit and receive
S	1 blocks	transmit start level of buffer (free blocks)
E	2 blocks	transmit expedite level of buffer (free blocks)
Q	256	size of each queue
B	6	number of references accessed at a time by the host software
N	128	maximum references in TDR ready queue
f_{PCI}	33 MHz	PCI bus clock frequency

$f_{Channel}$	384 Kbps	Channel data rate
PRIORITY	0	receive channel is not high priority
PRIORITYB	0	transmit channel is not inhibited

The average PCI bus utilization for one FREEDM is shown in figure 7 as a function of the packet length. Two of the curves are for zero RAM latency, the third includes RAM latency with a ratio of 1:7 for read and write latency. The figure shows that utilization improves with B- the number of references cached in software before being read from, or written to, a FREEDM queue.

The PCI bus utilization is strongly dependant on the packet length. For small packets the utilization increases dramatically.

Figure 7. Utilization of a FREEDM With 384Kbps Channelized T1

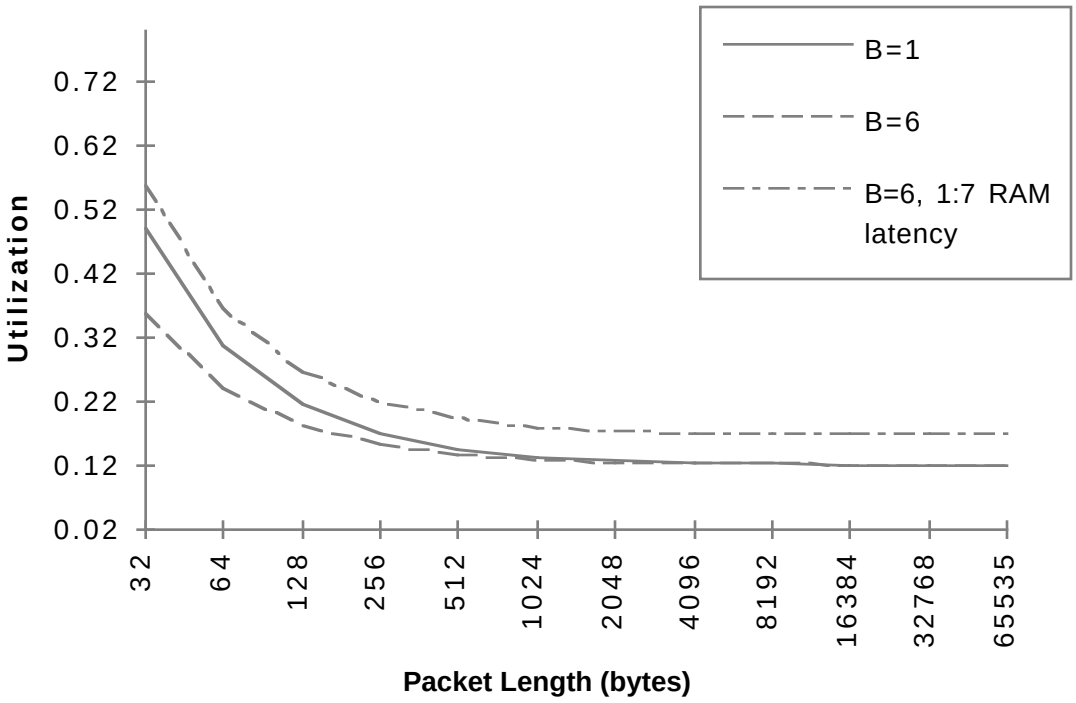


Table 5 below shows the maximum tolerable channel latencies and the worst case PCI bus latencies, assuming zero RAM latency. Both the latency estimates and the utilization graph indicate that 8 FREEDM cannot be supported.

Table 5. Latency Estimates With 128 384Kbps T1 Channels Per FREEDM

FREEDM's on bus	1	2	3	4	8
receive $L_{Channel}$ (msec)	6.7	6.7	6.7	6.7	6.7
transmit $L_{Channel}$ (msec)	1.3	1.3	1.3	1.3	1.3
receive L_{PCI} (msec)	0.16	0.33	0.49	0.66	1.3
transmit L_{PCI} (msec)	0.33	0.48	0.65	0.81	1.4
expedited transmit L_{PCI} (msec)	0.11	0.22	0.33	0.44	0.87

Combining Unchannelized E1 with 64Kbps Channelized E1

For this configuration the FREEDM serial ports are attached to E1 links. Four of the links are assigned to 4 unchannelized E1 channels. The remaining 124 channels are assigned from time-slots within the remaining E1 links, each channel being assigned to one time-slot, for 124 64Kbps channels.

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

Configurable Parameter	Value (E1 Channel)	Value (64Kbps Channel)	Description
F	4 blocks	4 blocks	size of buffer assigned to channel
X	2 blocks	2 blocks	DMA transfer size for transmit and receive
S	1 block	1 block	transmit start level of buffer (free blocks)
E	2 blocks	2 blocks	transmit expedite level of buffer (free blocks)
Q	256	256	size of each queue (select per FREEDM)
B	6	6	number of references accessed at a time by the host software (select per FREEDM)

N	128	128	maximum references in TDR ready queue (select per FREEDM)
f_{PCI}	33 MHz	33 MHz	PCI bus clock frequency (select per PCI bus segment)
$f_{Channel}$	2.048 Mbps	64 Kbps	Channel data rate
PRIORITY	1	0	1 = priority receive channel
PRIORITYB	0	1	1 = inhibited transmit channel

The average PCI bus utilization for one FREEDM is shown in figure 8 as a function of the packet length. Two of the curves are for zero RAM latency, the third includes RAM latency with a ratio of 1:7 for read and write latency. The figure shows that utilization improves with B- the number of references cached in software before being read from, or written to, a FREEDM queue.

The PCI bus utilization is strongly dependant on the packet length. For small packets the utilization increases dramatically.

Figure 8. Utilization of a FREEDM With E1 and 64Kbps Channels

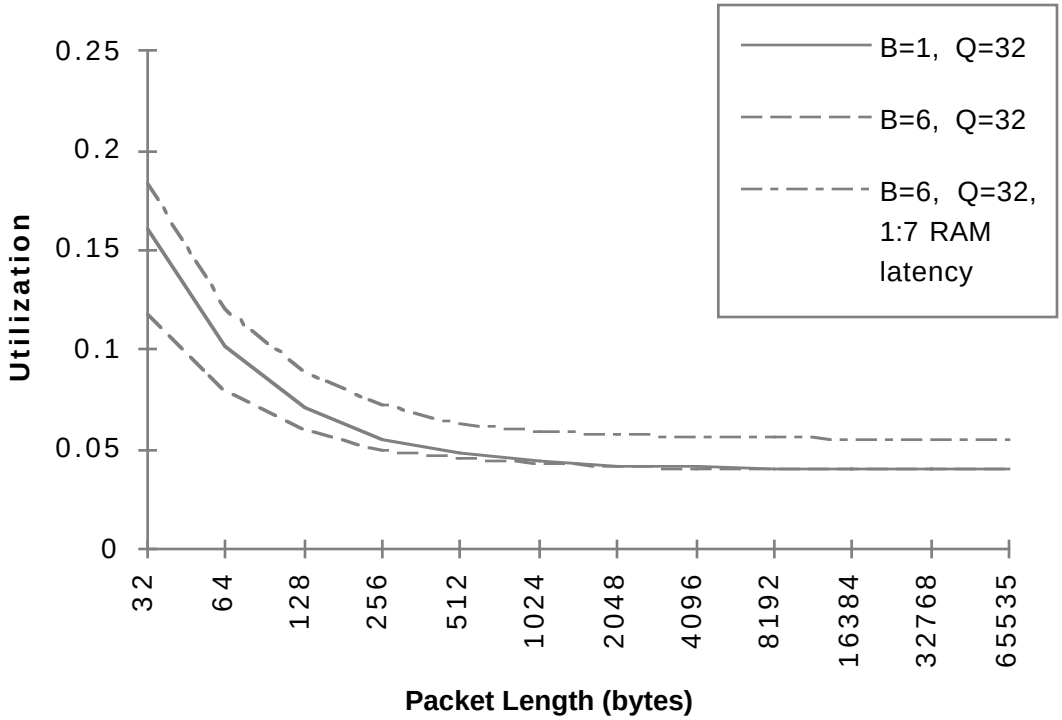


Table 6 and 7 below shows the maximum tolerable channel latencies and the worst case PCI bus latencies, assuming zero RAM latency. Both the latency estimates and the utilization graph indicate that 8 FREEDM can be supported with this configuration.

Table 6. Latency Estimates For Unchannelized E1 Channels

FREEDM's on bus	1	2	3	4	8
receive $L_{Channel}$ (msec)	0.12	0.12	0.12	0.12	0.12
transmit $L_{Channel}$ (msec)	0.24	0.24	0.24	0.24	0.24
priority receive L_{PCI} (msec)	0.006	0.012	0.018	0.024	0.049
expedited transmit L_{PCI} (msec)	0.0034	0.0068	0.010	0.013	0.027

Table 7. Latency Estimates For 64Kbps Channels

FREEDM's on bus	1	2	3	4	8
receive $L_{Channel}$ (msec)	4.0	4.0	4.0	4.0	4.0
transmit $L_{Channel}$ (msec)	5.9	5.9	5.9	5.9	5.9
receive L_{PCI} (msec)	0.31	0.47	0.63	0.79	1.4
transmit L_{PCI} (msec)	0.48	0.80	1.1	1.4	2.7

CONCLUSIONS AND RECOMMENDATIONS

This document provides insight into the performance of a FREEDM on the PCI bus. A summary of what has been learned in this analysis includes:

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

- The application can greatly affect the PCI bus utilization. Transfers involving more small 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 the 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 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 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 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 transmit.

REFERENCES

[1] PCI SIG, PCI Local Bus Specification, June 1, 1995, Version 2.1

[2] PMC-960113, PMC-Sierra, "Frame Relay Protocol Engine and Datalink Manager" Standard Product Datasheet, December, 1996, Issue 2

[3] PCI Compact Specification, PCI Industrial Computers Manufacturers Group, 1995, Version 1.0

CONTACTING PMC-SIERRA

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

Telephone: 604-415-6000
Facsimile: 604-415-6200

Product Information: info@pmc-sierra.bc.ca
Applications information: apps@pmc-sierra.bc.ca

World Wide Web Site: http://www.pmc-sierra.com

PMC-961061(R1) Issue date: February, 1997