860T Design Advisory
Rev. 0.4

Introduction

The MPC860T is an enhanced version of the MPC860 with added 10/100 Ethernet capability. Although the additional Ethernet connectivity does not impose any limit on the protocols available for the other (slower) serial channels in the 860T, the Fast Ethernet Controller (FEC) does share DMA resources with the other serial controllers and the virtual IDMA controller. In working with the current revision of Fast Ethernet microcode and 860T silicon (Rev. B.2), the designer should take into account the bottleneck that may be introduced by this shared resource in particular 860T configurations and must be aware of the potential worst-case latency that the FEC may experience during certain boundary conditions of operation. In this paper, we will present example scenarios illustrating these worst-case conditions and give system design suggestions for eliminating potential FEC problems.

The 860T system designer also needs to recognize that the slow serial channels may experience some throughput limitations as a result of the added latency that the FEC introduces on the bus. In this paper, we will quantify the impact that FEC operation has on slow serial channel latency and describe a mechanism for determining whether the 860T can support any proposed level of CPM loading.

Internal Arbitration

The interdependence of the various on-chip controllers is at the heart of the performance issues facing the 860T designer. Therefore, in order to understand the impact of FEC operation on the 860T system, we first need to consider the mechanism used by the FEC and the other serial channels to share the on-chip DMA resources. How do the FEC and the CPM (on behalf of the SCCs, SMCs, SPI, I2C, IDMA) arbitrate for access to the SDMA hardware? Second, it is important to know how the DMA engine then shares the internal U-bus with other internal masters in the 860T. In other words, how does the SDMA controller arbitrate for the bus?

We will start with the subject of arbitration for the SDMA hardware. In general, the FEC and CPM operate in a round-robin fashion in gaining access to the SDMA. If both the FEC and CPM attempt to access the SDMA at once, the CPM wins the first arbitration and then a round-robin follows.

Next, the SDMA arbitrates for the U-bus in order to transfer either 10/100 Ethernet data or slow serial channel data or to perform an IDMA operation. The SDMA requests the bus on behalf of either the FEC or the CPM. Since the SDMA has a higher arbitration priority than the CPU/caches, it will get the bus (even if the CPU also has a request) and may do a single-beat transaction or a burst. After a transfer, the SDMA then yields the bus for one transaction, which, depending on the other requesting bus master(s), may be a single-beat transaction or a burst. By yielding the bus after each transaction, the SDMA allows other bus masters to share the bus and (assuming some statistics about cache utilization) prevents the CPU from stalling.

If the CPU, FEC, and CPM all have bus requests, then the mastership would alternately pass from the SDMA to the CPU and back to the SDMA, etc. Each time the SDMA is bus master, access is granted in round-robin fashion to the FEC and the CPM (for the slow serial channels or the IDMA). The accesses occur in the following sequence:
SDMA ... CPU ... SDMA ... CPU ... SDMA etc.

When the CPU (with caches active) is bus master, accesses to cacheable memory will be bursts. By contrast, when the SDMA is bus master, the transaction may or may not be a burst, depending on the source of the request. FEC and IDMA requests to the SDMA resource do generate bursts to and from memory for data transfers (although buffer descriptor accesses are single-beat transactions). On the other hand, requests from the CPM on behalf of the SCCs, SMCs, SPI, or I2C controllers will generate only single-beat transactions.

Initially, we will assume that the IDMA is the source of requests to the CPM (which in turn arbitrates for use of the SDMA). By including the burst transfers of the IDMA, rather than single-beat transfers of the slow serial controllers, we will arrive at a better representation of the worst case latency for the FEC.

Now, looking more specifically at the source of the request to the SDMA, bus accesses occur in the following sequence:
FEC ... CPU ... IDMA ... CPU ... FEC etc.

Timing Analysis: Base Case

Next, we will consider the capability of the bus interface for a 50MHz 860T, utilizing an SDRAM memory system with equal access timing for burst reads and burst writes: 5-1-1-1. (Note: For a memory system with a longer access time for burst reads than for burst writes, we could use the read timing for all accesses to illustrate the worst case instantaneous access time. Alternatively, we could an average of the read and write timing. We will consider this choice in the examples section ahead.) In our system, one burst transfer of 4 words (=16 bytes) to or from memory takes 8cycles * 20ns/cycle = 160ns. Thus, the 860T's bus interface is capable of transferring one byte every 10ns.

Now, we will focus on the requirements of the FEC. Full-duplex Fast Ethernet has a 200Mbps data rate, or a demand for one byte to be transferred to/from memory every 40ns. (Similarly, half-duplex Fast Ethernet has a 100Mbps data rate, or a demand for one byte every 80ns.) We need to determine whether the 5-1-1-1 SDRAM access timing, is sufficient to support the demand of the FEC (in full- or half-duplex operation) as well as the demands of the other internal bus masters (e.g. CPU, CPM) in the 860T. We begin the calculations by considering the worst case latency conditions that the FEC will experience, and we will determine whether the FEC can still support a bus transfer rate of one byte every 40ns.

Under worst case conditions for the FEC, the CPU would use the bus for a transfer every time the SDMA yielded the bus after its own transaction. In addition, the CPM would request the use of the SDMA resource each time the round-robin between FEC and CPM returns to the CPM. As a result, the bus mastership would proceed as described earlier:

FEC ... CPU ... IDMA ... CPU ... FEC etc.

Adding to our analysis the number of bus cycles required for each bus master to complete its burst transaction, and using the 5-1-1-1 timing from above, we have the following sequence of bus cycles:
FEC (8) ... CPU (8) ... IDMA (8) ... CPU (8) ... FEC (8) etc.

From this example, we can see that the FEC makes one burst transfer every 32 cycles, for a data rate of one byte every 40ns:
(16 bytes / 32 cycles)*(50x10^6 cycles/sec)=1byte/40ns
At first glance, we might say that this rate is exactly what was required for full-duplex Fast Ethernet, (and even exceeds the requirement for half-duplex Fast Ethernet), and therefore conclude that this system will work. On the other hand, we have not yet considered one of the system boundary conditions, namely, the instantaneous effect of accesses to Fast Ethernet buffer descriptors on the latency seen by the FEC.

Each time the FEC needs to close the current buffer descriptor and open a new one, the 860 performs three single-beat accesses to memory: one access to write the status back to the current BD, and two more to read the next BD. The same process occurs for both Tx and Rx buffer descriptors. Furthermore, BD accesses have a higher priority in the FEC than data transfers, so BD accesses will add to the latency described above for burst transfers of Fast Ethernet data.

When the FEC is configured for full-duplex Fast Ethernet, the transmitter and receiver in the FEC will alternate their memory accesses. The worst case added latency for BD accesses occurs when new BDs must be opened for both Tx and Rx at the same time. (Even with half-duplex Fast Ethernet, the system may still experience a simultaneous demand for new Tx and Rx buffer descriptors. Consider the scenario in which the FEC begins transmission of a frame immediately after it completes reception of another frame. In this case, the FEC starts filling up the Tx FIFO with data from memory while still completing the transfer of received data from the Rx FIFO to memory.) If the FEC's transmitter has just used the FEC bus mastership for bursting data from memory into the Tx FIFO, then the receiver will have the next available data access time. In addition, the opening of new BDs will cause six single-beat accesses to memory (three for Tx and three for Rx) to consume the next six times that the FEC becomes bus master. Keeping in mind that single-beat accesses to the 5-1-1-1 memory system take 5 cycles each, we would see the following sequence of cycles on the bus:

FEC-Txdata (8) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-Rxdata (8) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-TxBD (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-RxBD (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-TxBD (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-RxBD (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-TxBD (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-RxBD (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
and loop on the top two lines until a buffer descriptor needs service again.

For this boundary case, the FEC's Tx FIFO obtains one burst of data in a total of 238 cycles: 32 + 32 + 6*(5+8+8+8) = 238 cycles. This translates into a worst case latency of 238 cycles / 50*10^6 cycles/sec = 4.76us between successive accesses to Fast Ethernet transmit data. Generally, the FEC's Tx FIFO can absorb this latency when it occurs in the middle of transmission of a frame. However, if this string of buffer descriptor accesses occurs early in the frame, when the Tx FIFO contains a minimum number of bytes ready

for transmission, an excessive latency for bringing Tx data from memory into the Tx FIFO could cause the FIFO to underrun.

To determine whether 4.76us is an excessive latency, we must consider the number of bytes in the Tx FIFO, as well as the transmit data rate on the line (100Mbps). By design, the FEC will not begin transmission of a new frame until the Tx FIFO contains 56 bytes of user data. Adding these 56 bytes to the 8 bytes of preamble generated in hardware, the FEC is prepared to transmit for exactly 64 byte-times, i.e. the duration of the collision window at the start of the frame. If the size of the data buffer for this particular frame is equal to 56 bytes, (which is the size of the user data field of a minimum length frame), then the BD accesses begin just when the Tx FIFO has 56 bytes available for transmission. In this case, while the buffer descriptor accesses take place, the Tx FIFO will drain to an empty state in 5.12us:
(64bytes*8bits/byte)/100x10^6bits/sec = 5.12 us

Since the latency of 4.76us is less than the FIFO's drain time of 5.12us, the Tx FIFO will not underrun in these conditions. To alleviate the need for this margin, however, the system should make use of larger buffers. This step would not only minimize the number of times that buffer descriptors must be opened and closed (thereby reducing system overhead), but it would also reduce the number of instances that the boundary case above could occur.

Timing Analysis: Why SDRAM is necessary in an MPC860T System

Suppose that the cost requirements of the design call for the use of a 40MHz 860T and EDO DRAM, rather than a 50MHz system with SDRAM, as described above. How will the new memory interface timing affect the risk of Tx FIFO underrun? We will begin by making a few assumptions. First, let us assume that the access timing for the EDO memory system is as follows:

Burst Read:	4-2-2-2
Burst Write:	3-2-2-2
Single Beat Read:	4
Single Beat Write:	3

Second, since the access timing differs for read and write, we must choose an appropriate value to use for each cycle controlled by the various internal masters of the 860T. We will use the Burst Read value (10 clocks) for all CPU accesses, since this represents the worst case. We will use the average of the Burst Read and Burst Write values (9.5 clocks) for the IDMA accesses, since each IDMA transfer must include one write for every read. For the FEC cycles, we will use each of the Burst Read (10), Burst Write (9), Single Read (4), and Single Write (3) timing values where appropriate.

With these assumptions in mind, we proceed to a timing analysis of the 40MHz 860T system with EDO DRAM, during the boundary case in which buffer descriptor accesses intervene between transmit data accesses:

FEC-Txdata Read (10) ... CPU (10) ... IDMA (9.5) ... CPU (10) ...
FEC-Rxdata Write (9) ... CPU (10) ... IDMA (9.5) ... CPU (10) ...
FEC-TxBD Write (3) ... CPU (10) ... IDMA (9.5) ... CPU (10) ...
FEC-RxBD Write (3) ... CPU (10) ... IDMA (9.5) ... CPU (10) ...

FEC-TxBD Read (4) ... CPU (10) ... IDMA (9.5) ... CPU (10) ...
FEC-RxBD Read (4) ... CPU (10) ... IDMA (9.5) ... CPU (10) ...
FEC-TxBD Read (4) ... CPU (10) ... IDMA (9.5) ... CPU (10) ...
FEC-RxBD Read (4) ... CPU (10) ... IDMA (9.5) ... CPU (10) ...
and loop on the top two lines until a buffer descriptor needs service again.

For the boundary case, the FEC's Tx FIFO obtains one burst of data in a total of 277 cycles: 39.5 + 38.5 + 2*(3+10+9.5+10) + 4*(4+10+9.5+10) = 277 cycles. This translates into a worst case latency of 277 cycles / 40*10^6 cycles/sec = 6.93us between successive accesses to Fast Ethernet transmit data. A comparison of this value with the Tx FIFO drain time of 5.12us reveals that the 40MHz EDO implementation would also experience the possibility of Tx FIFO underruns.

Before accepting the Tx FIFO underrun as a certainty in the design, however, we should consider some redesign alternatives that could potentially prevent the underrun event altogether. We will analyze the impact of each of the following four changes:
(1) Replace EDO DRAM memory system with SDRAM
(2) Use an external DMA controller instead of the 860's IDMA
(3) Run the 860 at a faster clock speed

Redesign Option 1: SDRAM

Beginning with a 40MHz design incorporating full-duplex Fast Ethernet, we consider a single system change in the area of memory. Using the SDRAM timing of 5-1-1-1 for reads and writes, which can be supported in a 40MHz design, we arrive at the following new timing sequence:

FEC-Txdata Read (8) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-Rxdata Write (8) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-TxBD Write (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-RxBD Write (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-TxBD Read (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-RxBD Read (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-TxBD Read (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
FEC-RxBD Read (5) ... CPU (8) ... IDMA (8) ... CPU (8) ...
and loop on the top two lines until a buffer descriptor needs service again.

With these values, one burst of Tx data occurs every 238 cycles, for a latency of 238 cycles / 40*10^6 cycles/sec = 5.95us. This value still exceeds the FIFO drain time, so we must conclude that upgrading from EDO to SDRAM is not sufficient to eliminate the Tx FIFO underrun risk.

Redesign Option 2: Eliminate IDMA

If we support the system's IDMA needs outside of the 860T, then the CPM's internal bus mastership tenures are used solely for serial channel activity. Since the accesses for supporting serial channels are always single beat accesses, rather than bursts, we alter the baseline case by replacing the IDMA burst timing (9.5 cycles for the average of burst read and burst write) with the single-beat read timing of 4 cycles. We use the read access time rather than an average of read and write (3.5 cycles) in order to evaluate the worst case latency during this boundary condition:

FEC-Txdata Read (10) ... CPU (10) ... non-IDMA CPM (4) ... CPU (10)
FEC-Rxdata Write (9) ... CPU (10) ... non-IDMA CPM (4) ... CPU (10)
FEC-TxBD Write (3) ... CPU (10) ... non-IDMA CPM (4) ... CPU (10)
FEC-RxBD Write (3) ... CPU (10) ... non-IDMA CPM (4) ... CPU (10)
FEC-TxBD Read (4) ... CPU (10) ... non-IDMA CPM (4) ... CPU (10)
FEC-RxBD Read (4) ... CPU (10) ... non-IDMA CPM (4) ... CPU (10)
FEC-TxBD Read (4) ... CPU (10) ... non-IDMA CPM (4) ... CPU (10)
FEC-RxBD Read (4) ... CPU (10) ... non-IDMA CPM (4) ... CPU (10)
and loop on the top two lines until a buffer descriptor needs service again.

The new timing analysis yields 233 total cycles and 233 cycles / 40*10^6 cycles/sec = 5.825us of latency for the Fast Ethernet transmit data. Thus, eliminating support for IDMA in the 860 is not sufficient to eliminate the potential for Tx FIFO.

Redesign Option 3: 50MHz 860T

If we maintain EDO DRAM in the system but increase the 860T's clock speed, we arrive at a new timing pattern for the memory interface:

Burst Read:	5-2-2-2
Burst Write:	3-2-2-2
Single Beat Read:	5
Single Beat Write:	3

The access sequence with new timing is as follows:

FEC-Txdata Read (11) ... CPU (11) ... IDMA (10) ... CPU (11) ...
FEC-Rxdata Write (9) ... CPU (11) ... IDMA (10) ... CPU (11) ...
FEC-TxBD Write (3) ... CPU (11) ... IDMA (10) ... CPU (11) ...
FEC-RxBD Write (3) ... CPU (11) ... IDMA (10) ... CPU (11) ...
FEC-TxBD Read (5) ... CPU (11) ... IDMA (10) ... CPU (11) ...
FEC-RxBD Read (5) ... CPU (11) ... IDMA (10) ... CPU (11) ...
FEC-TxBD Read (5) ... CPU (11) ... IDMA (10) ... CPU (11) ...
FEC-RxBD Read (5) ... CPU (11) ... IDMA (10) ... CPU (11) ...
and loop on the top two lines until a buffer descriptor needs service again.

Once again, the analysis reveals that Tx FIFO underrun is not eliminated with the increase in 860 clock speed. The latency is 302 cycles / 50*10^6 cycles/sec = 6.04us, which still exceeds 5.12us.

We have just completed an analysis of various redesign options and their impact on the risk of Tx FIFO underrun. Let us briefly look at the impact of bus latency on the FEC's receive path. In this direction, the key factor related to Rx FIFO overrun is simply the Rx FIFO size, which defaults to 256 bytes but is user-programmable. (The registers of interest are the FIFO Receive Bound Register, FIFO.R_BOUND, and the FIFO Receive Start Register, FIFO.R_FSTART.) In contrast to the Tx path, the Rx path can withstand a significant latency on the bus before the Rx FIFO fills to capacity. The transfer of receive data to memory begins as soon as a collision window (64 bytes) of Rx data is available in the FIFO. Therefore, if the Rx FIFO is empty when frame reception begins, and the Rx FIFO size is equal to the default, then the Rx FIFO can receive up to 192 bytes (= 256 - 64)

beyond the collision window before bus access latency would pose a threat of Rx FIFO overrun. Thus, although there is a boundary condition for the receive path, Tx FIFO underrun is the risk of greater concern.

CPM Performance Analysis

Returning to the transmit case, and now complicating matters a bit further, consider the scenario in which the IDMA is not the only controller making requests to the CPM for access to the SDMA resource. If we add, for example, one 10BaseT channel on SCC1, we need to consider whether the FEC, IDMA, and CPU data transfers will cause excessive latency to the SCC. Ignoring the FEC traffic for a moment, we know that an 860 at 50MHz can support up to 4 10BaseT channels. (Using Table A-1 in Appendix A: Serial Performance of the 860 User's Manual, CPM utilization with 4 half-duplex 10BaseT channels is [4*(10/22)*(25MHz/50MHz)]=91%.) This assessment is based on the assumption that the CPM will experience a latency no greater than 12 cycles between accesses. Such an assumption is reasonable when the CPU, caches, and IDMA are the only other internal bus masters requesting the bus (i.e. no active FEC), for two reasons: (1) the SDMA, arbitrating on behalf of the CPM, has a higher priority than the CPU and caches, and (2) when the IDMA is programmed to have a lower request priority within the CPM than the SCCs, serial data transfers may occur whenever the CPM has bus mastership.

In an 860T, however, in which the SCCs and FEC must alternate use of the SDMA, and both must share the bus with the CPU, the CPM may regularly experience a latency greater than 12 cycles. As a result, the maximum loading of the CPM, as documented in the Serial Performance Appendix of the 860 User's Manual, must be derated. In order to ensure adequate bus access for the SCCs, and thereby prevent serial channel FIFO overruns or underruns, the system designer should be certain that the planned CPM loading does not exceed the percentage stated in Table 1 (below) for the corresponding CPM bus access latency:

CPM Bus Access Latency (cycles)	Maximum CPM Loading (%)
0 <= x <= 12	100
12 <= x <= 20	100 - 2 (x - 12)
20 <= x <= 50	84 - 1 (x - 20)
50 <= x <= 100	54 - .5 (x - 50)

Table 1. CPM Bus Access Latency vs. Maximum CPM Loading

Using the relationship outlined in Table 1 to determine the effect of increased latency on an 860T system's aggregate serial bandwidth capability, Table 2 (below) presents the results for a variety of latency values:

CPM Bus Access Latency (cycles)	Maximum CPM Loading (%)
12	100

20	84
30	74
40	64
60	49
80	39
100	29

Table 2. Examples of CPM Bus Access Latency and corresponding Maximum CPM Loading

With this new information, we now consider the original sequence of accesses to the 860T's U-bus, and evaluate the CPM bus access latency and corresponding maximum CPM loading for the system. We have the following sequence of bus mastership: SDMA ... CPU ... SDMA ... CPU ... SDMA etc.

Assuming either the IDMA is programmed with lower priority than the SCCs, or the IDMA is not used in this application, then the following sequence illustrates the latency to the CPM:
FEC ... CPU ... CPM ... CPU ... FEC etc.

Adding cycles, based on the SDRAM timing for a 50MHz 860T:
FEC (8) ... CPU (8) ... CPM (5) ... CPU (8) and repeat

Here, the latency between successive CPM accesses may be as great as 29 cycles: 5+8+8+8. Using Table 1 as a guide, the designer should recognize that the CPM loading of the system must not exceed 75%: 84 - 1(29-20) = 75. With only a single 10BaseT channel on SCC1 and no IDMA, the CPM loading is well under the limit at just 23%: (10/22)*(25MHz/50MHz)=0.227.

To examine the relationship between latency and loading even further, we will consider several 860 system scenarios (assuming 0 to 12 clocks of latency) and then determine the impact of added latency on a comparable 860T system.

Example 1

The 860's CPM is 98% loaded when handling 64 HDLC channels at 50 MHz. (Based on Table A-1 in Appendix A of the 860 User's Manual: [(64*0.064)/2.1]*(25MHz/50MHz)=0.975.) When the latency increases to 29 clocks in an 860T system (as in the discussion above), the CPM loading must be at or below 75%. Starting from this value, we calculate that the CPM can handle up to 49 HDLC channels, rather than 64: [(49*0.064)/2.1]*25/50=0.75.

Example 2

If a 50MHz 860 is configured for one half-duplex 10BaseT channel and one 2 Mbps full-duplex HDLC channel, in a system with under 12 clocks of latency for the CPM, then the CPM utilization is only 35%: [(10/22)+(2/8)]*(25MHz/50MHz)=0.35. In evaluating the same serial configuration for an 860T, in which the CPM may experience greater latency -- we will again use 29 cycles -- we calculate that the maximum supported CPM loading is 75%: 84 - 1(29-20) = 75. Therefore, with an actual CPM loading well under the limit, the

50MHz 860T will not experience a latency problem while supporting a CPM load consisting of one Ethernet channel and one 2Mbps HDLC channel, in addition to a Fast Ethernet channel. (Of course, the system must also have adequate bus bandwidth to support the aggregate data throughput.)

Example 3

Some 860 designs might include an external master, such as a PCI bridge device, on the MPC8xx bus. Just like the 860's internal masters, this device must arbitrate for access to the system bus in order to access the memory subsystem. From the perspective of the CPM, the external master adds latency and therefore reduces the system's maximum supported CPM loading.

When the 860's SIU Module Configuration Register (SIUMCR) is programmed for internal arbitration, the external master's arbitration request priority may be set anywhere from 7 for highest priority to 0 for lowest priority. Assuming priority level 6 is used (as in the case of Tundra's Q-Span PCI bridge device), the external master always has higher priority than the 860's internal masters (with the exception of the refresh controller). Furthermore, once the 860's internal arbiter grants the bus to the requesting external master, the external device may maintain assertion of the BB* (Bus Busy) signal and keep control of the external bus indefinitely.

Therefore, the designer should incorporate external logic in the interface to the external master to ensure that this device will yield the bus before causing serial channel underruns or overruns or starving the CPU. Once a limitation is placed on the number of cycles during which the external master may continuously control the bus, the user can calculate the latency that the CPM will experience when the external master is an active part of the system.

The following sequence of external bus master mastership characterizes an 860T system with a bursting external master (denoted as EXT) that is permitted only one burst transaction during each bus tenure. You will notice that the round-robin of internal masters continues despite the presence of an external master which alters the external bus mastership sequence.

FEC (8) ... EXT (8) ... CPU (8) ... EXT (8) ... CPM (5) ... EXT (8) ...
... CPU (8) ... EXT (8) ... and repeat

In this scenario, the CPM may experience a latency as great as 61 cycles from one bus access to the next, during a time when multiple masters -- including the external master -- have bus access requests: 5+8+8+8+8+8+8+8=61. Consulting Table 1, we determine that the maximum CPM loading for this system is 48%: 54 - 0.5(61-50) = 48.5. To provide one example of the capacity of the CPM under these loading conditions, the following equation illustrates that a 50MHz 860T can support one T1 and 2 HDLC channels at 768kbps each and still be less than 48% loaded:
[(24*0.064)/2.1+(2*0.768)/8)]*(25MHz/50MHz)=0.46

Conclusion

The impact of the boundary conditions and worst-case scenarios described in this paper will be reduced in future 860T designs with the help of enhancements to the current 860T.

First, the microcode will be revised to permit only one buffer descriptor access, rather than six consecutive accesses, to intervene between accesses to Fast Ethernet data. Another planned enhancement is the addition of user-programmability for the water-mark in the Tx FIFO that determines when the transmitter can begin transmission of a new frame. To allow for greater latency without Tx FIFO underrun, the user may select 64, 128, or 192 bytes of data in the Tx FIFO as the starting point for frame transmission. These design changes will occur in the next revision of the 860T silicon (Rev. D), currently targeted for 1Q99. Until these enhancements are implemented, the designer must pay careful attention to the factors affecting 860T performance and their impact on any proposed system design.