# MVME6100 DMA Performance Observations and Guidelines

William Dennen
VME Product Operation

| Buffer Size | Write MB/sec | Read MB/sec |
|---|---|---|
| 64KB | *239* | *238* |
| 128KB | *248* | *246* |
| 256KB | *252* | *250* |
| 512KB | *254* | *252* |
| 1MB | *255* | *254* |
| 2MB | *256* | *254* |
| 4MB | *256* | *254* |
| 8MB | *256* | *254* |
| 16MB | *256* | *254* |
| 32MB | *256* | *254* |
| 64MB | *256* | *254* |
| 128MB | *255* | *253* |

**Table 1**
**Tsi148 DMA Performance**

Table 1 shows the impressive performance that has been achieved using a pair of MVME6100 boards under VxWorks 5.5.1 in an industry standard 12 slot VME64x (5-row backplane) chassis. The values were obtained using a single DMA channel utilizing the 2eSST transfer mode of the Tsi148 while the chassis was otherwise "quiet". These values, identical for both direct and linked-list DMA transfers, likely represent the maximum performance that could be expected by users of the MVME6100 when a single Tsi148 DMA channel is utilized. This paper describes the configuration requirements a user will need to perform in order to achieve equivalent performance in their application. The code used to generate the values captured in the various tables of this paper is available through MCG's OnLine Services as Solution S1852. In all instances the transfer mode is 2eSST and, although direct DMA was used, equivalent values were achieved using linked-list (chained) DMA. When both channels of the Tsi148 DMA controller are used 295 MB/sec has been obtained writing 8MB blocks, 256 MB/sec while reading.

Four items have been identified that affect the performance of DMA transfers when using the Tsi148 VME bridge on the MVME6100. Each of these will be discussed in the following paragraphs. Another item, however, does need to be mentioned. The VMEbus does not have unlimited bandwidth. DMA traffic must contend with other VMEbus users for the available capacity. All the transfer rates described in this paper were achieved by ensuring that there were no other users of the VMEbus. Specifically all other boards within the chassis were idling in their firmware, either PPCxBug or MOTLoad.

The first configuration item to address is the value selected for the VME block size in the DMA descriptor. This must be set correctly to achieve the maximum possible performance. The value chosen for maxVmeBlockSize in the structure VME_BUS_USAGE (see tempe.h)

should be 2048. This is the maximum block size supported under VME (ANSI/VITA 1.1-1997, section 11.1.6). The Tsi148 was designed with an 8KB buffer to support two 4KB buffers; 4KB being the maximum block size on the PCI-X bus (PCI-X Protocol Addendum to the PCI Local Bus Specification Revision 2.0a, section 7.2.3). Internally the DMA engine of the Tsi148 VME bridge will attempt to move buffers toward their destination bus simultaneously. An apparent flaw in the Tsi148 precludes two concurrent 4K buffers, so selecting the maximum block size of 4K for PCI-X bus (maxPciBlockSize in the TEMPE_DMA_ATTRIBUTES structure) and 2K for the VMEbus (maxVmeBlockSize in the TEMPE_DMA_ATTRIBUTES structure) allows the DMA engine to process both busses simultaneously. The ability to transfer data on both busses simultaneously provides an approximate 60 MB/sec advantage over transferring on only one bus at a time.

The second configuration item for the user to address is the need to disable snooping on PCI bus 0. This can be accomplished by undefining the macro PCI_DMA_SNOOP_BUS_0_ON in config.h. If shared memory is configured by the user, then this macro will be undefined as a result of defining INCLUDE_SM_NET. Disabling snooping implies that the user will also have to disable caching in the targeted memory region and maintain coherency through software.

There is a caveat, however, of which the user must be aware: if the definition of INCLUDE_SM_NET is performed by means of the Tornado Project Facility then the result is not an optimal configuration. The configuration, either by means of a #undefine of PCI_DMA_SNOOP_BUS_0_ON or a #define of INCLUDE_SM_NET, must be performed by a modification in config.h. This is because sysMv64360Smc.c directly includes config.h; thus the value selected in config.h, not prjComps.h, will control the build of vxWorks. This situation is another instance of differing build results that can occur between use of the Project Facility and command line operations.

An alternative to controlling snooping and caching with the use of build time parameters would be through the use of sysMv64360SpecialMem(). This routine, found in sysMv64360SpecialMem.c, can be used to dynamically control the characteristics of a memory region. The DMA performance that was observed when snooping on PCI bus 0 was not disabled by either mechanism is shown in Table 2.

| Buffer Size | Write MB/sec | Read MB/sec |
|---|---|---|
| 64KB | *181* | *204* |
| 128KB | *187* | *212* |
| 256KB | *190* | *216* |
| 512KB | *191* | *218* |
| 1MB | *192* | *219* |
| 2MB | *192* | *219* |
| 4MB | *192* | *220* |
| 8MB | *192* | *220* |
| 16MB | *192* | *220* |
| 32MB | *192* | *220* |
| 64MB | *191* | *220* |
| 128MB | *190* | *220* |

**Table 2**

**Tsi148 DMA to snooped/cached memory**

It is the architecture of the Discovery II Processor Host Bridge that requires snooping to be turned off on PCI bus 0. As with the Discovery I Processor Host Bridge used on the MVME5500, the device does not use the PPC bus as the path for transfers to or from DRAM. Rather it uses one of the ports of the internal crossbar switch. The advantage of the Discovery architecture is that simultaneous transfers between differing ports are possible. The disadvantage is that if memory is to be snooped, the transaction must stall while being presented to the processor for a snoop cycle. It is the introduction of this latency that leads to the less than optimal DMA performance.

The third item that contributes to the performance of the Tsi148 DMA engine is not controlled within software. Rather it is the position of the MVME6100 relative to the System Controller of the rack in which the MVME6100 resides. The closer the MVME6100 is to the System Controller, the greater the observed transfer rate. Testing has determined that it does not matter what VME bridge device is the System Controller; identical transfer rates were observed when using a VME6000 (on an MVME147), a Universe II (on an MVME2100), and a Universe IID (on an MVME5500). It is the length of the daisy chain through which the bus request and grant has to traverse that is significant. This effect has been observed before with MBLT transfers, but is startling when first observed with 2eSST transfers.

The values shown in Table 1 were achieved with the MVME6100 master in slot 1 and the MVME6100 target in slot 2. In this case the DMA engine co-resides with the System Controller. When the target MVME6100 is moved to slot 10 the results shown in Table 3 were observed. The slots between the master MVME6100 and target MVME6100 were occupied with a representative sample of MCG PowerPC boards, all idling in firmware. A slight, one MB/sec, performance drop is observed. This indicates that although the length of the backplane that the transfer must traverse does affect performance, it does not do so significantly.

Table 4 shows the transfer rates achieved when the MVME6100 master was moved to slot 9 of the chassis while the MVME6100 target remained in slot 10. Other than moving the card formerly in slot 9 to slot 1, the payload of the chassis remained identical to that used for Table 3. A noticeable drop in achieved performance, on the order of 26 MB/sec, is observed. Spot checks with the MVME6100 master located in various slots between 2 through 8 indicate that the achievable throughput drops in a linear fashion the further the DMA engine is located from the System Controller.

The observed performance drop can be somewhat mitigated by isolating the MVME6100 pair onto an otherwise unused bus request level. This is the fourth, and final, element for users to manage. Tables 3 and 4 show the performance observed when all boards in the chassis were on bus request level 3. Table 5 provides the results obtained when the MVME6100 pair in slots 9 and 10 were isolated onto bus request level 2. A 13 MB/sec improvement is observed. When the MVME6100 pair, isolated on Bus Request Level 2, was

located in slots 1 and 2 there was no difference in the performance observed from when all boards where on Bus Request Level 3. The bus request level was changed by altering the line YES_INIT_TEMPE_REG (TEMPE_VMCTRL, 0x00000707) in sysTempe.c (at line 183) to YES_INIT_TEMPE_REG (TEMPE_VMCTRL, 0x00000706).

| Buffer Size | Write MB/sec | Read MB/sec |
|---|---|---|
| 64KB | *238* | *237* |
| 128KB | *246* | *245* |
| 256KB | *251* | *249* |
| 512KB | *253* | *252* |
| 1MB | *254* | *253* |
| 2MB | *255* | *253* |
| 4MB | *255* | *253* |
| 8MB | *255* | *254* |
| 16MB | *255* | *254* |
| 32MB | *255* | *253* |
| 64MB | *254* | *253* |
| 128MB | *253* | *253* |

**Table 3**

**DMA engine in slot 1, Target in slot 10**

| Buffer Size | Write MB/sec | Read MB/sec |
|---|---|---|
| 64KB | *220* | *219* |
| 128KB | *227* | *225* |
| 256KB | *231* | *229* |
| 512KB | *232* | *231* |
| 1MB | *233* | *232* |
| 2MB | *234* | *232* |
| 4MB | *234* | *233* |
| 8MB | *234* | *233* |
| 16MB | *234* | *233* |
| 32MB | *234* | *232* |
| 64MB | *233* | *232* |
| 128MB | *232* | *232* |

**Table 4**

**DMA engine in slot 9, Target in slot 10**

| Buffer Size | Write MB/sec | Read MB/sec |
|---|---|---|
| 64KB | *231* | *230* |
| 128KB | *239* | *238* |
| 256KB | *243* | *242* |
| 512KB | *245* | *244* |
| 1MB | *246* | *245* |
| 2MB | *247* | *245* |
| 4MB | *247* | *245* |
| 8MB | *247* | *245* |
| 16MB | *247* | *245* |
| 32MB | *247* | *245* |
| 64MB | *247* | *245* |
| 128MB | *247* | *245* |

**Table 5**

**DMA engine in slot 9, Target in slot 10**

**Isolated at Bus Request Level 2**

To summarize, optimal DMA performance is achieved by

1. setting the VME block size to 2048,
2. ensuring that the memory buffers are unsnooped and uncached,
3. ensuring that the DMA engine is as close as possible to the system controller, and
4. isolating the master and target boards on a Bus Request Level.