The Effect of High Bandwidth and Low Latency Memory on Computers

Michael Kirzinger

***Abstract*—Adding high bandwidth and/or low latency memory to a computer results in improved performance. This performance gain is as a result of widening the bottleneck caused by the increasing gap in speeds between the processor and main system memory. This paper investigates how large of a gain can be achieved by widening the bus between the CPU and memory, increasing memory speed, and reducing memory latency. Latencies of real modules are used in the simulation to see what is possible by using available high performance memory modules.**

***Index Terms*—DDR, DDR2, GDDR3, high bandwidth, low latency, memory**

I. Introduction

ONE of the main bottlenecks in computer systems today is the speed of the memory available to the processor. This bottleneck can be further divided into two categories: A bottleneck caused by memory latencies, and a bottleneck caused by memory bandwidth.

This paper will investigate the effects of increasing the memory bandwidth available to the processor. This will be done by increasing the bus width between the memory and processor and by increasing the speed at which the memory runs. As well, it will compare increasing the bus width to increasing data rate for a fixed bandwidth.

The effect of reduced latency will also be simulated. This will look at values beyond what is currently available on the market.

II. Simulator

A slightly modified version of the SimpleScalar tool set (a unified diff against 3v0d SimpleScalar is available at http://www.ece.ualberta.ca/~mjk3/ece510/burst_length.diff) was used for all the simulations. One modification was made to the memory system: A new parameter was added to set the minimum and maximum burst length for the memory. This adds one of the differences between DDR and DDR2 (and GDDR3) into the simulation.

III. Simulation Parameters

The processor used in the simulation is similar to a socket 754 AMD Athlon 64. This has a 64KB, 2-way set associative L1 data cache[1]. The cache lines are 64 bytes wide and a least recently used replacement policy is used. The L1 instruction cache is the same.

The processors are available with either a 512KB L2 cache or a 1MB L2 cache. A 512 KB L2 cache was used for the simulations. A clock speed of 2400 MHz was used.

A second cache configuration consisting of 32KB L1 instruction and data caches with a 64KD L2 unified cache was also used to stress the memory. This was necessary to do since there was insufficient time to run simulations on very large data sets. Using a smaller cache should have a similar effect to using a large data set, that effect being cache misses caused by insufficient capacity in the cache to fit all the data.

The RAM parameters in SimpleScalar were based upon the CAS and tRCD latencies of Samsung DDR[2], DDR2[3], and GDDR3[4] memory modules. The available burst length for DDR are 2, 4 and 8[2]. For DDR2, it is 4 and 8[3]. GDDR3 has burst lengths of 4 and 8[4]. A power of 2 line size in the cache cache will always be used so that it is always creates a power of 2 ratio to the burst length.

To convert from *CAS* and *tRCD* latencies and the data rate to the SimpleScalar parameters *-mem:lat* <*first_chunk*> <*inter_chunk*>, equations (1) and (2) were used.

$$first\_chunk = \frac{CPU\_Clock \times (CAS + tRCD)}{Memory\_Clock} \quad (1)$$

$$inter\_chunk = \frac{CPU\_Clock}{Memory\_Data\_Rate} \quad (2)$$

For DDR memory, the memory data rate is twice the memory clock. All values for *first_chunk* and *inter_chunk* have been rounded up.

A summary of the modules used and their corresponding first and inter chunk values is shown in Table 1.

	CAS	*tRCD*	*first_chunk*	*inter_chunk*
DDR 266	2	3	91	10
DDR 333	2.5	3	80	8
DDR 400	3	3	72	6
DDR2 533	4	4	72	5
DDR2 667	5	5	73	4
DDR2 800	5	5	60	3
GDDR3 1000	7	8	72	3
GDDR3 1200	9	10	76	2
GDDR3 1400	10	10	69	2
GDDR3 1600	11	12	69	2
GDDR3 1800	11	12	62	2

Table 1: Latencies of various memory modules used

There were 5 different programs used to test the effects of varying the memory parameters:

*1) matrix: M*ultiplies two 100x100 matrices consisting of double precision floating point numbers together.

2) sort: Does a quick sort of 40000 double precision floating point numbers.

3) fft: Performs a complex discrete fast fourier transform on a 65536 element double precision floating point number array. Uses a general purpose FFT package[5].

4) filter: Runs an 8192 length double precision signal through a 200 tap finite impulse response filter.

5) alphaBlend: Performs an alpha blend of two 800x600 24 bit bitmaps.

IV. Simulation 1: Effect of Bandwidth

The five test programs were run through the simulator for each memory module on a 64, 128, 256, and 512 bit memory bus.

The results, sorted by bandwidth, for the small cache is shown in Figure 1. Results for the large cache show the same thing, just the values are closer together because of most of the data fitting into the cache. Performance is relative to the 128 bit DDR 400. The fft program stresses the memory the most, with an 18.3% spread in performance. Increasing bandwidth showed an increase in most cases. The small differences in performance among two memory configurations with similar bandwidths are due to differences in the latency.

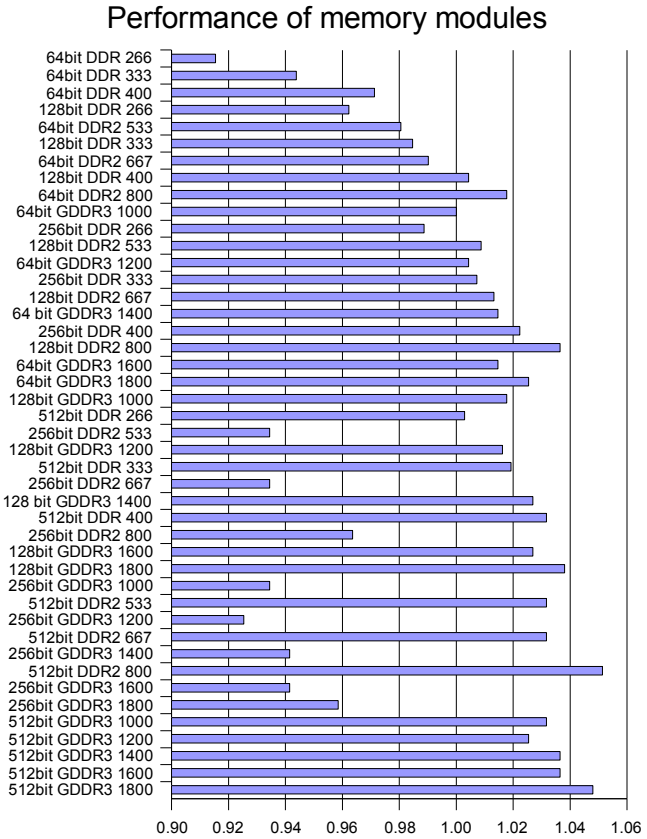

Figure 1: Performance of various memory modules in the small cache processor simulation

The 256 bit DDR2 and GDDR3 performs poorly. This is a result of the cache line only being twice the size of the bus width. The memory is unable to do a burst read of 2, so it has an expensive 2 reads. Once they go to 512 bits, the cache line can be filled in 1 read.

In general, there is only a small improvement in going to a wider bus when the wider bus does not allow the full burst length to be used.

Now that the memory system is being utilized more, the performance gains of a higher bandwidth system become more pronounced. Once again, the 256 bit DDR2/GDDR3 suffers in performance for the same reason as before.

V. Simulation 2: Effect of Cache Line Size on Equal Bandwidth Memory Configurations

The cache line size will effect the performance of the memory, as seen in the 256 bit bus case in the first simulation. The memory's speed will be made better use of with larger cache lines, but this can also harm performance for applications which do not have data accessed in a sequential order.

For this simulation, a constant bandwidth of 12.8 GB per second was used. The memory configurations with this bandwidth are 256 bit DDR 400, 128 bit DDR2 800, and 256 bit GDDR3 1600. These are all at or near the top of their memory types for clock rate and overall latency.

The results are all relative to the 128 bit DDR2 800 with a 64 byte cache L2 line size. These results for the small cache processor are shown in Figure 2. The large cache simulation data shows the same results, except that the differences in performance are smaller.

The wider bus performs better than the faster memory running on a smaller bus. The slower memory ends up sending more data per burst read. The extra first chunk access times that the faster memory is encountering is causing it to perform worse than the slower memory on a wider bus. The cache line size creates improvements in the performance until

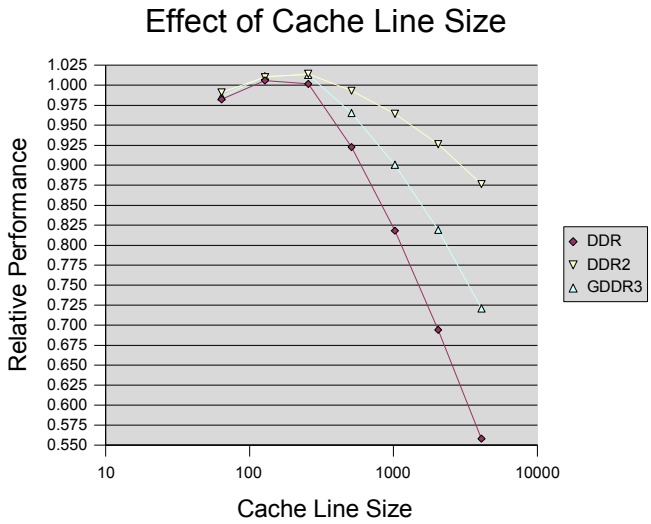

Figure 2: Relative performance of modules with a bandwidth of 12.8GB per second with varying cache line size

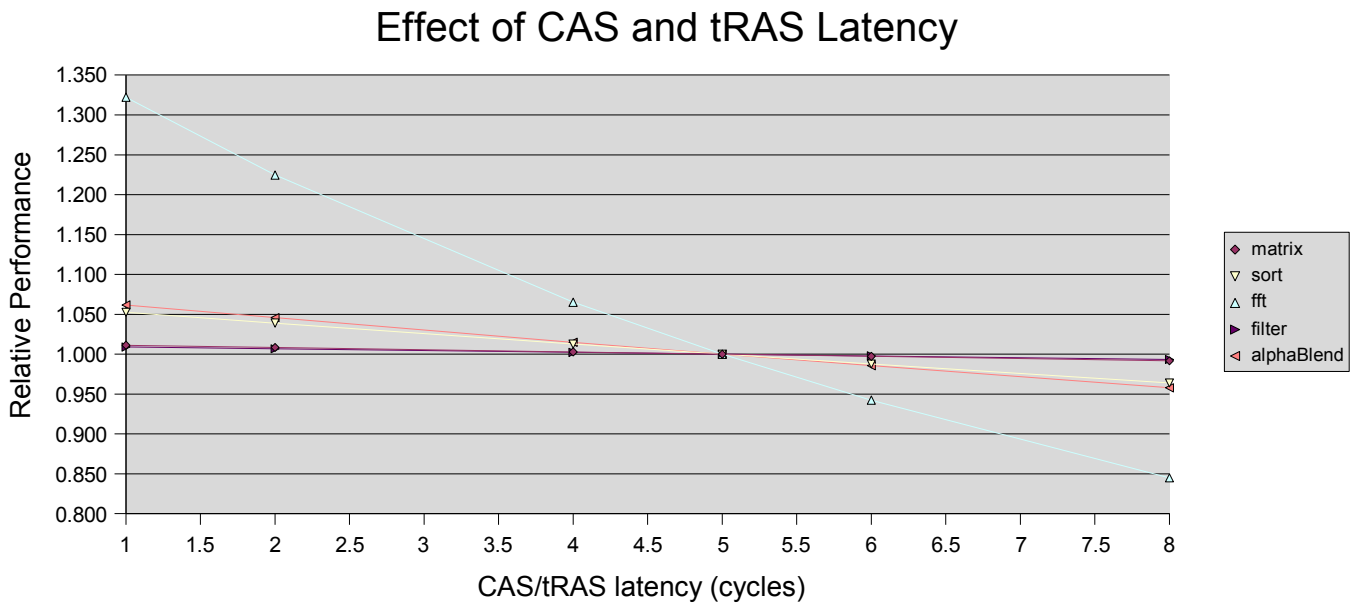

Figure 3: The effect of CAS latency using 128bit DDR2 800 memory modules.

the point at where it starts taking multiple bursts to fill the cache line.

Once again, the fft is the most demanding of all the test programs. This shows poor performance on large caches because of the non-sequential order the data is accessed in. The 64 bit GDDR3 1600 (4096 byte cache line size) performs at 5.3% of the 128 bit DDR2 (64 byte cache line size) in the fft program. Overall, the relatively lower latency of the DDR2 800 makes it the best performer for the smaller, more commonly found line sizes on processor caches.

VI. Simulation 3: Effect of Latency

The final simulation looked at the effect of latency. This takes a 128 bit DDR2 800 module and varies it's CAS and tRAS latencies from 1 to 8.

Once again, the simulation results are similar for the simulations with the large and small caches. The results of the small cache simulation are shown in Figure 3.

It shows that reducing latency increases overall performance. The performance gain can be quite large, as seen in the results of the fft program. The gains are larger than those seen by increasing bandwidth.

VII. Conclusions

Performance can be increased by decreasing latency and/or by increasing bandwidth. Lowering latency is a more effective method of improving performance than increasing bandwidth, but that is not a simple task. Really low latency memory (such as the really low latency that was looked at in simulation 3) does not exist. However, all the bandwidth configurations looked at besides the 512 bit memory bus are in wide use in a variety of special applications, such as on video accelerators.

For larger bus widths, the cache size should be increased. The 256 but bus showed best performance with a 256 byte cache line size. The ideal line size to cache line size is 1 bit on the bus to 1 byte in the cache line. This is because it takes full advantage of the burst read length of 8 of the DDR RAM. The 64 and 128 bit buses currently found on most computer interfaces are well matched to the processor cache line size of 64 bytes found in desktop processors such as the AMD Athlon 64. To move to a wider bus, the caches in current processors will need larger cache lines.

In conclusion, improving the memory latency and increasing the bandwidth can have significant benefits for specific applications, and a general speedup for all applications.

VIII. Future Work

Future work would involve enhancing the simulator to simulate all the aspects of DDR and DDR2 memory. As well, instead of using actual modules, it might shed more light on what is going on by using timings that just look at bandwidth or latency and not both at the same time.

References

[1] "sandpile.org – AA-64 implementation – AMD K8" http://www.sandpile.org/impl/k8.htm

[2] Samsung. "512Mb C-dir DDR SDRAM Specification" http://www.samsung.com/Products/Semiconductor/DDR_DDR2/DDRSDRAM/Component/512Mbit/K4H510438C/ds_k4h51xx38c_fbga_rev11.pdf

[3] Samsung. "DDR2 Unbuffered SDRAM MODULE " http://www.samsung.com/Products/Semiconductor/DDR_DDR2/DDR2SDRAM/Module/UnbufferedDIMM/M378T3253FZ3/ds_ddr2_256mb_f_die_base_udimm_rev13.pdf

[4] Samsumg. "512Mbit GDDR3 SDRAM" http://www.samsung.com/Products/Semiconductor/GraphicsMemory/GDDR3SDRAM/512Mbit/K4J52324QC/ds_k4j52324qc_rev12.pdf

[5] Takuya OOURA. "FFT Package 1-dim/2-dim" http://momonga.t.u-tokyo.ac.jp/~ooura/fft.html