Efficiency of the AMD Opteron Using K10 Architecture and Multiprocessor Configurations

Italo Borrelli
CSc Undergrad, University of Victoria
British Columbia, Canada
iborrelli@uvic.ca

***Abstract*—This paper studies the AMD Opteron with the K10 Architecture, especially focusing on the 6000-series Magny-Cours. It is a study of the strengths and weaknesses of it's architecture as well as the multi-processor topologies and their contribution to the efficiency of computing speed in the AMD Opteron.**

I. Introduction

THE AMD Opteron processor was announced on April 22, 2003 as the first 64-bit processor with support for the standard x86 architecture without loss of speed [1]. AMD also offered the highest performance for 2- and 4-way processors in their announcement, according to Standard Performance Evaluation Corporation (SPEC) benchmarks. On that date they released the Model 240, 242 and 244, which supported only up to 2-way servers, all based on the K8 microarchitecture. The 100-series 1-way servers and the 800-series for 4- and 8-way servers were not released until June 30 of the same year [2].

Dual-core chipsets for the Opteron were released in 2005, quad-core was introduced in 2007 along with the K10 microarchitecture, the 6-core in 2009 and the 12-core in 2011 [3]. The 12-core Opteron was part of the 6000-series Magny-Cours.

The Opteron has been used in many super-computers since it's release, which speaks to the processors ability as a high level performing CPU especially for use in blade servers. In the TOP500 list of the 500 most powerful non-distributed computers, based on floating-point operations per second (FLOPS), the AMD Opteron was used in 21 of the top 100 systems in the summer of 2006 and in 33 of the top 100 in November 2010 and June 2011 [4]–[6]. Notable uses in supercomputers include the IBM Roadrunner at the Los Alamos National Labratory and the Titan at the Oak Ridge National Labratory, which were the top system on the TOP500 lists in 2008 and 2012 respectively [7]–[8].

Since the 6000-series in 2011, AMD has released the Bulldozer and Piledriver microarchitecture based Opteron CPUs. This research will primarily focus on the K10 architecture, especially with regards to the Magny-Cours series 12-core models. The Magny-Cours series can run two or four socket configurations and the following research will examine the strengths and advantages of the processor using the K10 architecture on a multi-socket system as well as potential downsides as well as challenges when using the K10 microarchitecture.

II. Overview of the Opteron 6000-Series Design & the K10 Architecture

A. Native Legacy Mode for x86 Applications

Prior to the release of the AMD Opteron, any 64-bit processor architecture, such as the IA64 ISA as used by the Intel Itanium, did not have native x86 compatibility and required new code and emulation to run legacy 32-bit applications [10]. The legacy mode feature was released with the K8 and continued in future Opteron ISAs, including the K10 ISA [3]. Essentially, instead of designing a brand new 64-bit architecture and working to make it compatible with the x86 ISA, AMD decided to base their new architecture on x86, and extend their instruction set to function with 64-bit code [11]. This met the needs of clients who's programs hadn't transitioned yet to make full use of 64-bit architectures.

B. Magny-Cours CPU Design

The 12-core Magny-Cours CPUs are comprised of two dies. Each die consists of six cores, four 16-bit wide HT (HyperTransport) links and two channels for DDR3 memory. In each CPU, one full-width and one half-width HT link are used between the dies and four full-width HT link ports are used. The way they are used depends on whether the system is using 1, 2 or 4 processors (hereafter 1P, 2P and 4P respectively), but, in any case, one full-width link is used for I/O [3].

Each of the full HT links are 16-bits comprised of two unidirectional DDR (Double Data Rate) links. The links run at 3.2 GHz and by using DDR links the data rate has a maximum of up to 6400 transfers per second per direction, since, if necessesary, there is a transfer on the rising and falling edge of the clock signal [9].

C. Multi-CPU Arrangments for the AMD Opteron 6000-Series

There are three primary arrangments used with the 8- or 12-core Magny-Cours processors, the 1P, 2P and 4P. A 1P system is much simpler as HT links are only used between the two dies and one for the I/O, leaving three of the four HT ports unused.

In a 2P system there are four dies. Each die has a connection to one die in the other processor with a full-width HT link and to the other die in the other processor it has a half-width HT link (See Fig. 1). One die in each processor has a full-width

link to I/O. This compromises the four HT link ports each processor has, but two of the dies only use three of the four HT links they have available to them.

In a 4P system there are eight dies. One die in each processor has a half-width link to one die in each of the other CPUs, while the other die has a half-width link to each of the other dies (See Fig. 1). Again, one die in each has a full-width link to I/O leaving unused HT links for one die in each processor.

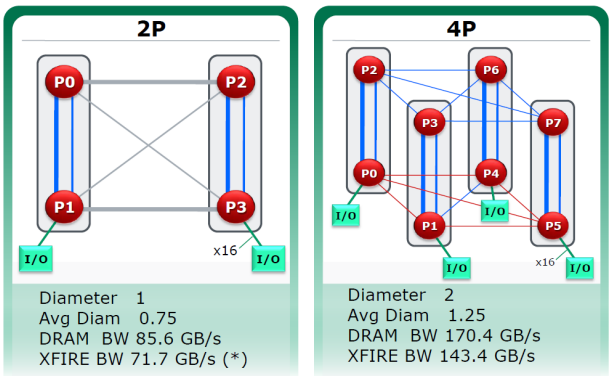

Fig. 1. 2P and 4P Topology for Magny-Cours [3]

D. Cache Structure and Coherency

In the Magny-Cours die, there is 6MB of L3 cache, and 512kB of L2 cache per core, so 3MB in a 6-core die. With a maximum of 48 threads running in a 4P system with 12-core processors, it is critical that the cache remains coherent and processors are using the most up-to-date versions of data. To avoid having on-chip probe filter for cache entries, which would be unused in a 1P system, the cache directory uses 1MB of the L3 cache in each die. Using L3 cache minimizes access latency, port occupancy for read, write and modify operations and latency across cache on probe filter misses [3].

Each die holds 256k probe filter cache lines which are each 64 Bytes, comprised of sixteen 4 Byte entries. This keeps track of the state of a total of 16MB of cache.

III. Advantages of the AMD Opteron K10 Architecture

A. Efficiency of Running x86 Architecure Code

The Opteron was the first processor with the ability to execute x86 code without a loss of speed. By having a legacy mode for 32-bit programs, there was no decrease in speed, since the same registers and instruction set were made natively available on the system. In long mode there is no decrease or increase in speed. The x86 code is not recompiled for the extra registers accessible to 64-bit operations, but has full access to the legacy registers and thus there is no detriment in running x86 code on the K8 or K10 architectures [11].

B. Advantages of HyperTransport Assist for Reduced Latency

The HT Assist system used in Magny-Cours reduces latency because caches on other nodes don't need to be checked. Instead, the directory easily identifies whether the cache line is modified, exclusive, owned, shared and/or invalid. This is the MEOSI cache coherency protocol, which won't be detailed here. In addition, the topology of the dies through the HyperTransport and the coverage of the cache directories ensures that a transaction doesn't need to be broadcast across every node in the system [3][12].

C. Branch Prediction for K10 Architecture

The K10 architecture uses a Branch Target Buffer (BTB) with 2048 entries in the L1 cache. There are four options for indicating how to deal with a branch. Any branch that is not taken does not get an indicator. The branches indicators are set as follows [13]:

- If a branch is not taken it is indicated as 'never branch' and is not included in the BTB.
- When a branch is taken it is then set to 'always branch' in the BTB.
- If a branch indicated as 'always branch' is not taken, it is then set as 'make dynamic predictions' in the BTB. This branch will not change back to 'always branch' at any point during this execution.
- If a branch is a call, it is indicated to use the return stack buffer in the BTB which has 24 entries in the BTB.

This system has very low overhead since the branch indicators are identified by only 2-bits. This system of updating branches is efficient since no indicator is set until there is a branch on the code, and the updates to it do not further fill the buffer, and instead replace the previous information. The initial predictions are extremely simple and take up only 2^{16} Bytes of L1 cache [14].

Additionally, small loops can be easily predicted with a global history stored with 16k entries [14].

D. Strengths of the K10 ISA

There are a few notable benefits in the K10 architecture that greatly improve the speed and efficiency of operations.

1) Efficiency of Read-Modify and Read-Modify-Write Instructions: Most read-modify and read-modify-write instructions only take one macro-operation [13]. Three macro-operations are able to be run per clock cycle greatly increasing performance for simple instructions such as move and many simple arithmetic and logical functions [15].

2) Stack Operations: Stack operations use the Sideband Stack Optimizer (SSO) rather than the ALU. The SSO tracks the stack-pointer value which speeds up some operations on the stack, even if they are reliant on each other, such as some near CALL and RET instructions. They can be executed parallely without having to be sent to the ALU which greatly speeds up execution time [14].

3) Floating Point Pipelining: Some floating point operations have been designed to be easily pipelined. This is done in addition and multiplication, for example, by keeping multiplier and adder pipes busy and performing simultaneous operations that are independent and do not rely on each others completion. Efficient instruction packing can maximize the core to perform two floating point instructions per clock cycle [14].

IV. Possible Weaknesses and Bottlenecks in the K10 Microarchitecture

A. Scheduling with Mixed Latency

There may be delays in operations when scheduling operations with different latencies. Take MOVQ for instance. It is comprised of two macro-operations with latencies of 2 and 5 that use any of the 3 floating point units (FADD, FMULT, FMISC) [14]. Suppose there are two MOVQ instructions one after another. The first macro-operation starts at time 0 and ends at time 2. The second starts at time 1 and ends at time 6. There is one available floating point unit now that the first macro-operation of the second MOVQ instruction can use, and it will start at time 0 and end at time 2. The problem through is taht the second macro-operation for the second MOVQ can't start until time 3 now and won't finish until time 8.

There are ways around this latency delay, but it adds an extra level of difficulty to designing programs in this architecture. This deadlock can cause delays if not dealt with by the programmer.

B. Floating Point Limitations

In relation to the above, there are limitations in capabilities for floating point operations. Macro-operations with varying latencies are not efficiently scheduled and throughput is limited to one floating point macro-operation per clock cycle [14]. This can be fixed by mixing in integer operations to balance out the throughputs and latencies associated with floating point operations, but this requires other operations to be available [13]. If a system is computing primarily floating point operations, as is the case for many super computers, there may not be enough available non-floating point operations to mix in.

C. Dependency Accumulation and Operation Delays

It is possible that in some operation schedules, dependencies will stack up and code that could be run may be delayed waiting for other instructions to complete [13]. This may happen with regards to registers of partial flag accesses.

If a register is presumed to be affected there may be a delay if a different portion of a register is being accessed by two lines of code. For example, if the low-bits are operated on by one line and high-bits by another instruction, the second instruction may wait until it's exact place in the schedule, rather than running immediately. This error may be fixed by a clear operation on the register which would identify to the second instruction that it does not have to wait for the previous instructions to all complete.

If two instructions affect the same flags, but need access to a different partial bit of the flag register, there will be an inefficient delay because of the presumed dependency. This process may be helped by implementing an instruction between the two that will affect the flag so the second instruction does not wait for the first flag to be used [13].

V. Analysis

The AMD Opteron K10 architecture is well optimized to run 64-bit code and 32-bit legacy code. It's instruction set for floating point operations is efficient when balanced among the floating point units. The K8 and K10s strengths with floating point calculations is definitely a reason for it's adoption by many super computing research centers.

There is no benefit or native optimization for x86 code, but that was not it's intention in the design of the processor. Legacy code on the AMD Opteron does not have the downfalls of the Intel Itanium and it makes it easy for developers since they did not have to rewrite their legacy code, and rather, only had to consider how to developing for 64-bit CPUs.

A heavy competitor for the Opteron was the Intel Xeon, but in benchmarks against the X5000 series, the Opteron often comes out in the lead in a range of tests including SQL queries, graphics and simulations and encryption /citeform16.

VI. Conclusion

The K10 architecture is well suited to multi-processor Opteron designs. It's architecture allowed the execution of 32-bit programs with no drawbacks and set a standard for future architectures to have legacy integration.

The topology using HyperTransport wasa huge part of the success of the AMD Opteron in blade computing making it a great architecture for super computers.

Although AMD has moved on to other designs and newer architectures, the K10 architecture was an important step for future technologies, and, along with the K8, laid a foundation for the extension of x86–64 into the Intel Core architecture, which is ubiquitous today.

REFERENCES

[1] AMD, 'AMD Transforms Enterprise Computing With AMD Opteron™ Processor, Eliminating Barriers To 64-Bit Computing', 2003.

[2] AMD, 'AMD Expands Options for 4-Way and 8-Way Enterprise Computing with the AMD Opteron Processor 800 Series', 2003.

[3] P. Conway, N. Kalyanasundharam, G. Donley, K. Lepak and B. Hughes, 'Blade computing with the AMD Opteron™ processor ("Magny-Cours")', 2009 IEEE Hot Chips 21 Symposium (HCS), Stanford, CA, 2009, pp. 1–19.

[4] TOP500, 'TOP500 List (June 2006)', 2006. [Online]. Available: https://www.top500.org/list/2006/06. [Accessed: 02-Mar-2019].

[5] TOP500, 'TOP500 List (November 2010)', 2010. [Online]. Available: https://www.top500.org/list/2010/11. [Accessed: 02-Mar-2019].

[6] TOP500, 'TOP500 List (June 2011)', 2011. [Online]. Available: https://www.top500.org/list/2011/06. [Accessed: 02-Mar-2019].

[7] TOP500, 'TOP500 List (June 2008)', 2008. [Online]. Available: https://www.top500.org/list/2008/06. [Accessed: 02-Mar-2019].

[8] TOP500, 'TOP500 List (June 2012)', 2012. [Online]. Available: https://www.top500.org/list/2012/06. [Accessed: 02-Mar-2019].

[9] B. Holden, J. Trodden and D. Anderson, 'HyperTransport 3.1 Interconnect Technology'. Colorado Springs, Colo.: MindShare Press, 2008.

[10] F. Weber, 'Opteron and AMD64: A Commodity 64 bit x86 SOC', 2003 Salishan High Performance Computing Conference (HPC), Glenenden Beach, OR, 2003, pp. 1–32.

[11] A. Shimpi, 'AMD's 64-bit strategy: x86–64', AnandTech, 23-Apr-2003. [Online]. Available: https://www.anandtech.com/show/1098/5. [Accessed: 03-Mar-2019].

[12] J. D. Gelas, 'AMD's Six-Core Opteron 2435', AnandTech, 01-Jun-2009. [Online]. Available: https://www.anandtech.com/show/2774/2. [Accessed: 03-Mar-2019].

[13] A. Fog, '3. The microarchitecture of Intel, AMD and VIA CPUs'. Technical University of Denmark, 2018, pp. 31–34, 190–203, 226–227, 232–235 [Online]. Available: https://www.agner.org/optimize/microarchitecture.pdf. [Accessed: 03-Mar-2019].

[14] AMD, 'Software Optimization Guide for the AMD Family 10h and 12h Processors', 2011, pp. 59–60, 229–230. [Online]. Available: https://www.amd.com/system/files/TechDocs/40546.pdf. [Accessed: 03-Mar-2019].

[15] A. Fog, '4. Instruction tables'. Technical University of Denmark, 2018, pp. 29–38 [Online]. Available: https://www.agner.org/optimize/instruction_tables.pdf. [Accessed: 03-Mar-2019].

[16] J. De Gelas, 'AMD's 12-core "Magny-Cours" Opteron 6174 vs. Intel's 6-core Xeon', AnandTech, 29-Mar-2010. [Online]. Available: https://www.anandtech.com/show/2978/amd-s-12-core-magny-cours-opteron-6174-vs-intel-s-6-core-xeon. [Accessed: 03-Mar-2019].