Dynamic interconnection networks: the crossbar switch

Alejandro Ayala
School of Information Technology and Engineering
University of Ottawa
800 King Edward Avenue
Ottawa, Ontario, K1N 6N5, Canada

***Abstract*—This paper describes various aspects and implementaions of the crossbar interconnect. The performance of a multiprocessor system depends on having an efficient bus architecture. In System-on-a-Chip (SoC) there are different architecture types such as single stage networks, multi-stage networks, omega networks and crossbar networks. This paper focuses on the latter, the crossbar network. Most modern architectures adopted the crossbar switch for interconnecting microprocessors and resources such as memory. The use of the crossbar switch in these modern architectures will be evaluated. Furthermore, there is a discussion on the different techniques used to increase the performance of the crossbar switch. Performance is measure in terms of how efficiently the crossbar can send data between input and output.**

I. Introduction

The performance of the multiprocessor SoC is not dependent solely on the speed of the CPU but also on how the bus architecture is designed. Having an efficient bus architecture and arbitrating contention plays an important role in increasing the performance of the system. An M x N crossbars is a switching fabric which allows M inputs to connect to N outputs without blocking. Blocking only occurs when two inputs want to talk to the same output. The use of buffers at either the input or output ports allow to prevent any data loss when trying to get two or more inputs to access the same output. The crossbar switch has being used in several modern architecture designs. The following sections discuss these designs.

II. Modern hardware design

A. Intel Xeon 7500 architecture

The Intel Xeon 7500 series processor architectures uses an 8 port crossbar switch which implements the Intel QuickPath Interconnect (QPI) and routing layers. The Intel QPI is point-point interconnection link developed by Intel [1] to compete with HyperTransport. Prior to using QPI the Intel architecture was using the Front Side Bus (FSB) [2] which was the bus that carried data between CPU and the memory controller(northbridge) and I/O controller(southbridge). The off-chip single memory controller often was the bottleneck which prevented faster CPUs from improving the overall performance of the system. Now, with crossbar implementation the microprocessor were able to integrate the northbridge on-chip for faster access. Figure 1 shows the crossbar switch/router used by the Xeon 7500 series processor.

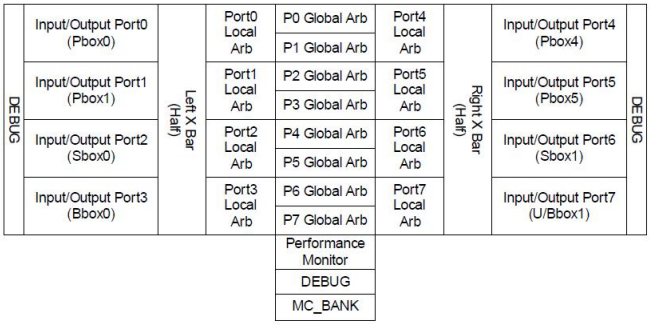

Fig. 1. Intel Xeon 7500 series crossbar [2]

One can see that there are 2 crossbars. Each crossbar has 4 ports, connecting the caching agent, home agent and QPI links. The caching agents are used to provide flow control to the Last Level Cache (LLC) coherence engine, the home agents and the crossbar. The home agents are responsible for processing requests and snoop responces from caching agents, providing data responces to the system via the crossbar and read/write commands through the memory controller. Basically these two agents translate messages from or to the QPI in able to talk to the respective resource either the cores or memory. The crossbar contains arbiters in order to prevent contention to any of the resources. There is a local arbiter on the crossbar and also a global one in case input from one crossbar wants to communicate to the output on the other crossbar.

B. AMD Opteron 64 architecture

The AMD Opteron 64 uses a similar approach to the on-chip interconnect as we saw with Intel's QPI. However, AMD was first to introduce an integrated northbridge approach. They use a 5 port crossbar to interconnect the microprocessor cores with the memory controller and HyperTransport links. AMD was also challenged by the fact that the FSB was slow and connecting multiple cores or microprocessors did not improve the overall system as well as they wanted. Using the crossbar switch to integrate the memory controller onchip helped speedup the system. Figure 2 shows the difference between using the FSB and integrating the memory controller. As one

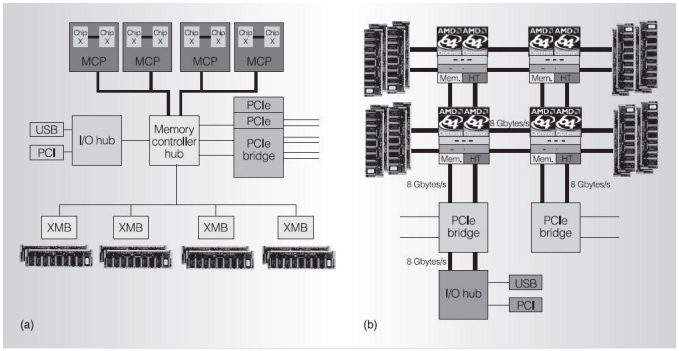

Fig. 2. Evolution of x86 blade server architecture: Traditional FSB architecture (a) and AMD's Direct Connect architecture (b) [3]

can see the bottleneck in Figure 2(a) is the single external memory controller which interconnected the microprocessors with the memory. Using the on-chip crossbar is faster and more cost-effective than having a separate chip for memory and I/O access.

The crossbar onchip has five ports connecting the System Request Queue (SRQ), memory controller, and three HyperTransport ports. The command headers and data packets are logically separated each using its own crossbar. The command crossbar routes commands from the HyperTransport at a rate of 1 per clock [3], these commands can be 4 or 8 bytes long. The data crossbar routes the command payloads which are 4 to 64 bytes long. Figure 3 depicts the architecture of the Opteron using the crossbar for interconnection.

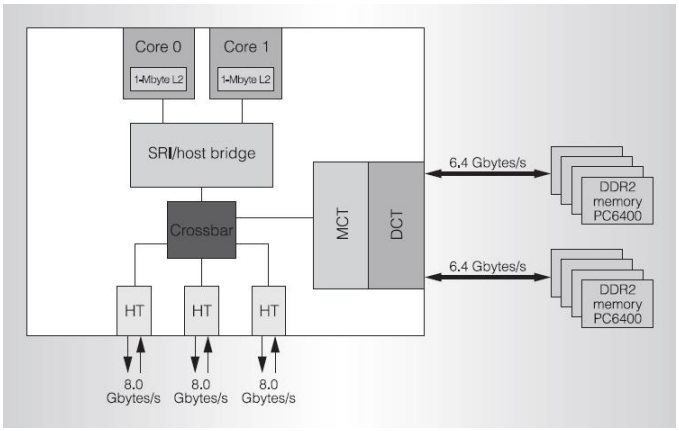

Fig. 3. AMD Opteron Architecture [3]

C. Sun UltraSparc T2

The UltraSparc T2 architecture uses a high bandwidth, non blocking crossbar to communicate with the L2 cache. Figure 4 shows the architecture of the UltraSparc T2. There are 8 cores (inputs) and 8 L2 caches plus the I/O hub (outputs). The good thing about a crossbar is that enables the system to send data simultaneous without blocking. The crossbar is divided into two modules logically: processor to cache(PCX) and cache to processor(CPX) as seen in Figure 5. Each module has its own data slice, through which the data can be transferred from a certain source input to the output according to the authorization message generated by arbiter. The arbiter takes action when several inputs issue requests for the same destination output by issuing authorization messages to the source input who gains priority. The crossbar being used by the UltraSparc is composed of three stage crossbar pipeline: request, arbitration and data transfer as described in [4].

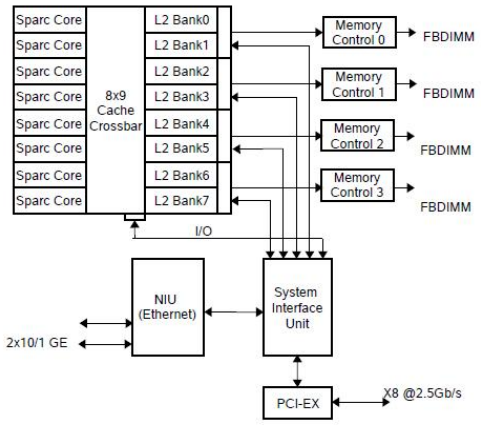

Fig. 4. Sun UltraSparc Architecture [5]

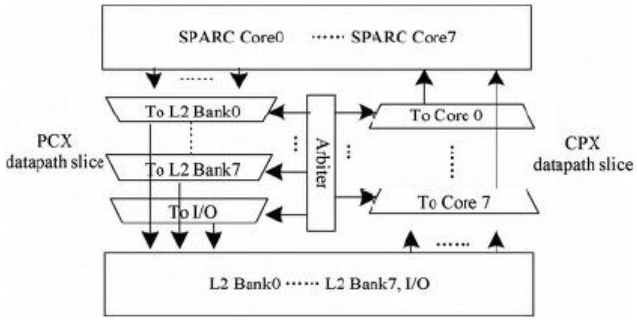

Fig. 5. Sun UltraSparc Crossbar [4]

III. Characteristics of crossbar switches

From previous section modern architecture design using crossbars where described. The Intel Xeon 7500 series, AMD's Opteron 64 and Sun's UltraSparc T2. For crossbar switches, packets can be buffered at either output ports, input ports, or crosspoints of the crossbar.

A. Output buffer

Output queued (OQ) switches need to transfer the packets immediately from input to output queue since there is only space available at the output. Problem with OQ is that it needs a speedup of N for an N x N switch, since at most there could be N inputs trying to transmit data to an output as explained in [6]. This make OQ switches hard to scale up.

B. Input buffer

Instead of having the buffer at each output port, we can also have buffers at the input ports. Therefore, no need to have a speedup for the input port. Input queued (IQ) switches are popular in the market since they are economical hardware architecture and have efficient scheduling algorithms as shown in [6]. AMD's Opteron uses this type of buffer in their crossbars. For the command crossbar the input buffer is 60 bit while the data crossbar uses multiple of 64 bytes to optimize their data throughput. As stated before the size of the command headers are 4 to 8bytes (32 to 64bits) and data payload is 4 to 64bytes.

C. Buffer crossbars

Combined Input-Crosspoint-Output Queued (CICOQ) switches or buffer crossbars, are a special type of Combined Input-Output Queued (CIOQ) switches, where each crosspoint of the crossbar has a small buffer. These switches simplify the scheduling greatly and output contention is eliminated. For example if N inputs try to send to same output there is no need to schedule the packets as before, just send the packets to the queue and then the output will retrieve packets one by one from the queue [6].

D. System request queue

The SRQ in the AMD's Opteron architecture is used to connect the cores to the crossbar. It prioritizes connection to the crossbar switch for both CPU cores, so that access to the system bus is managed more effectively, resulting in a smooth use of system resources as described in [3].

IV. Knockout Switch

The crossbar switch is a great interconneted network fabric where no blocking occurs when packets destined to different outpu[illegible]he problem occurs when multiple packets arrive at the same output simulataneao[illegible]The knockout switch was proposed here [7] to solve this problem. The knockout switch is a fully interconnected switch (ie. all input are connected to all outputs), just like the crossbar. Each output port uses a bus interface that has several functions such as filtering and queuing [7].

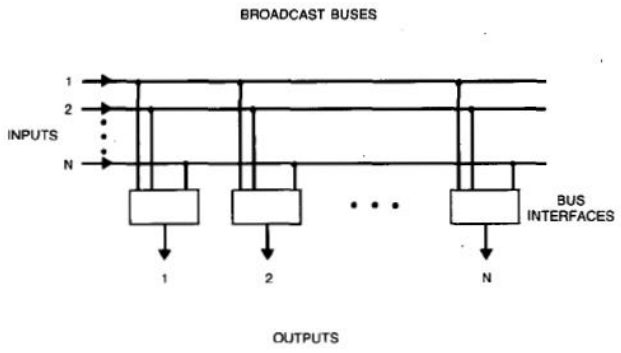

Fig. 6. Interconnection fabric [7]

Figure 6 shows N inputs, each attached to a broadcast bus. Each output has a bus interface attached to each input. The problem the knockout switch is trying to solve is contention to the same output from different inputs. Figure 7 shows what the bus interface looks like. The bus interface has 3 main features, the packet filter, concentrator and share buffer.

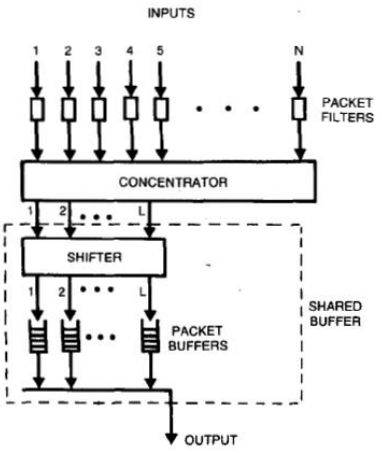

Fig. 7. Bus interface [7]

The packet filter will make sure the packet is destine for the output will pass on. Remember that each bus interface is attach to the broadcast bus, receiving all packets even those not destine to them. N packets will go through (if the output address matches) onto the concentrator. The concentrator will gather N packets and let L packets go through to the shared buffer. L packets entering the shared buffer which is composed of L separate FIFO buffers.

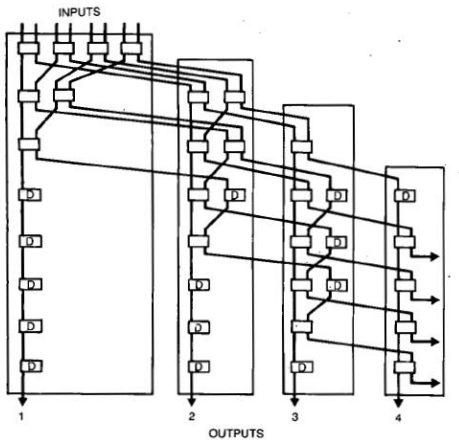

Fig. 8. The 8-input/4-output concentrator [7]

Figure 8 shows how the concentrator gath[illegible]which L packets will go through and no get knocked out (here N = 8 and L =4). As one can see each packet entering the concentrator has L chances of going through. Figure 7 shows that L outputs from the concentrator enter an L x L shifter, in order to filled the L separate buffers in a cyclic fashion [7]. Figure 9 shows

an example of how the shifter fills the buffer in the right order.

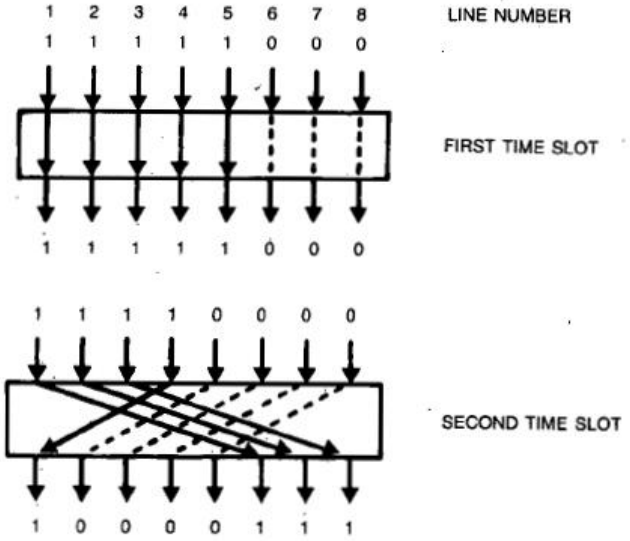

Fig. 9. The 8-input/4-output concentrator [7]

Here we first see that 5 packets got through the concentrator (bit set to 1 means it got through) on the first time slot. Buffers 1 - 5 will get filled by theses packets. On the second time slot 4 packets get through, now in order to fill buffers starting at 6 we need to shift by 5, so buffers 6,7,8, and 1 will get filled.

V. Conclusion

As one can see, having an efficient bus architecture for interconnecting the microprocessor with resources such as the memory and other cores is very important. Not only having fast throughput but being able to arbitrate memory contention is one of the goals when determining the type of interconnection network to use. The knockout switch was proposed to solve the problem of memory contention while still maintaining high performance. Furthermore, different approaches were discussed where modern designers like AMD, Intel and Sun are using the crossbar to achieve higher data throughput by integrating the memory controller on-chip versus the old approach of using the front side bus. The benefits of using the crossbar to integrate memory controller, router and HyperTransport/QPI interfaces on-chip include lower latency, power and cost. The crossbar switch has proven to be a very valuable tool to increase performance and cost effective switching fabric and was selected by the aforementioned Sun, Intel and AMD architectures.

References

[1] P. Stillwell, V. Chadha, O. Tickoo, S. Zhang, R. Illikkal, R. Iyer, and D. Newell, "Hippai: High performance portable accelerator interface for socs," in *High Performance Computing (HiPC), 2009 International Conference on*, 2009, pp. 109 –118.

[2] Intel, http://www.intel.com/Assets/en_US/PDF/datasheet/323341.pdf.

[3] P. Conway and B. Hughes, "The amd opteron northbridge architecture," *IEEE Micro*, vol. 27, pp. 10–21, 2007.

[4] A. Huang, J. Gao, C. Feng, and M. Zhang, "Optimization techniques of on-chip memory system based on ultrasparc architecture," jan. 2009, pp. 428 –431.

[5] M. Shah, J. Barren, J. Brooks, R. Golla, G. Grohoski, N. Gura, R. Hetherington, P. Jordan, M. Luttrell, C. Olson, B. Sana, D. Sheahan, L. Spracklen, and A. Wynn, "Ultrasparc t2: A highly-treaded, power-efficient, sparc soc," nov. 2007, pp. 22 –25.

[6] D. Pan and Y. Yang, "Localized independent packet scheduling for buffered crossbar switches," *IEEE Transactions on Computers*, vol. 58, pp. 260–274, 2009.

[7] Y.-S. Yeh, M. Hluchyj, and A. Acampora, "The knockout switch: A simple, modular architecture for high-performance packet switching," *Selected Areas in Communications, IEEE Journal on*, vol. 5, no. 8, pp. 1274 – 1283, oct. 1987.