A Case Study in Networks-on-Chip Design for Embedded Video

Jiang Xu[1], Wayne Wolf[1], Joerg Henkel[2], Srimat Chakradhar[2], Tiehan Lv[1]
1. Dept. of Electrical Engineering, Princeton University
2. NEC Laboratories America, Inc.
{jiangxu, wolf, lv}@princeton.edu, {Henkel, chak}@nec-labs.com

Abstract

In this paper we study bus-based and switch-based on-chip networks for an embedded video application, the Smart Camera SoC (system on chip). We analyze network performance and overall system performance in detail. We explore system performance using crossbars with different sizes, fixed size but different numbers of ports, and different numbers of shared memories. We find that network is a performance bottleneck in our design, and the system using an optimized NoC can outperform one using a bus by 132%. Our simulations are based upon recorded real communication traces, which give more accurate system performance. Our study finds that for the Smart Camera system, a 16-bit/port 3x3 crossbar with two shared memories shows 85.7% performance improvement over the bus-based model and also has less maximum network throughput than the bus-based model. This design example illustrates a methodology to quickly and accurately estimate the performance of NoC's at architecture level.

1. Introduction

This paper studies different NoC's (networks-on-chip) for a real multi-core SoC (system-on-chip) system, the Princeton Smart Camera system. We use recorded real communication traces of a SoC design to conduct a detailed NoC design. We comprehensively explore bus and crossbars with different sizes, different port widths, and different numbers of shared memories. We showed their effects on the system performance, maximum throughput, average throughput, and network utilization. We find an unoptimized network is a performance bottleneck in the Smart Camera System. A system using an optimized NoC can outperform one using a bus by 132%. We also find high average throughput always associates with low network utilization in crossbar-based switch for Smart Camera SoC. Our practice shows a quick and accurate method to estimate the performance of NoC's at architecture level.

We adopted a telecommunication network simulator, OPNET [12], for our study and developed our own simulation method to design the NoC. For the Smart Camera System, we find the 16-bit/port 3x3 crossbar with two shared memories shows an 85.7% performance improvement over bus-based model, while it has less maximum throughput.

The next section describes related work. In section 3, we introduce the Smart Camera SoC, which we used as the case. Section 4 shows several models we use in our research. Our simulation method is described in section 5, and simulation results and analysis are presented in section 6. Section 7 concludes our work.

2. Related work

A key problem in SoC design is overcoming the design difficulties of on-chip communication architectures. Surveys of networks-on-chip are given by several researchers [1] [2] [3] [9]. Some on-chip communication architectures are developed based on buses, for example, CoreConnect from IBM [4], AMBA from ARM [5], MicroNetwork from Sonics [6], and Wishbone from Silicore [13]. Other on-chip communication architectures are inspired by multiprocessor networks, computer networks, and telecommunication networks [7] [8].

3. Smart Camera SoC

We believe the study of on-chip communication architectures should be based on real SoC designs. The designs should have multiple IP cores, which closely cooperate with each other to reach high performance. We chose the Smart Camera System [10] for our research, because it is a multi-core and relative large and complex SoC, and because we can generate traces from the design.

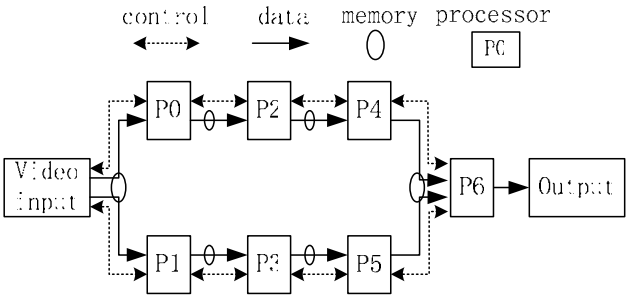

Figure 1. Computation architecture

Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'04)
1530-1591/04 $20.00 © 2004 **IEEE**

The Princeton Smart Camera system is a high-performance video processing application that can process 150 frames per second [14]. To deliver such high performance we use a dual pipelined computation architecture (Figure 1). Each frame will go through 5 processing stages. Processors P0, P2, P4, and P1, P3, P5 form the two pipelines, and processor P6 works for both pipelines. Based on the workload, processor P0 handles the first processing stage, P2 handles the second processing stage, P4 handles the third and fourth processing stages, and P6 handles the fifth processing stages for both pipelines. In the Smart Camera System, two control functions are required. First, a processing control function will decide if a processor should process a new frame or not, depending on the status of itself and the previous stage. If the previous stage finishes a frame and there is enough space to store the result in its own share of memory, a processor can process the finished frame from previous stage. Second, an arbitration function arbitrates requests for a shared communication system, based on the priorities of the requests.

4. Networks-on-chip models

To make a fair comparison of the on-chip networks, all the models use the same set of computation units. We use shared embedded memories with a single port for all the models. Because the algorithms for each processing stage may change, memory requirements for each stage will also change. Using a shared memory gives us more flexibility than we would have with distributed memories. For the dual pipeline computation architecture, a shared memory for each pipeline is also a good choice. The arbitration function uses the same priority list in all models. The input has the highest priority, and from the first to the fifth processing stages each one has a lower priority. For the same stage on different pipelines, the first pipeline always has a higher priority.

4.1. Bus-based models

We used two bus-based models, the processor-controlled model and the arbiter-controlled model. In the processor-controlled model, a bus arbiter handles the arbitration function, and the processors and the input handle the processing control function (Figure 2). The processors and the input are connected to the bus arbiter by interconnections, which form a star network. The interconnections are used to send requests to and receive responses from the bus arbiter. The bus includes a 32-bit data bus, a 21-bit address bus, and a 2-bit control bus. There are 2 bytes in the memory to record the status of each processing stage and the input. After a frame is finished, a processor updates its own status and the status of the previous processing stage. A processor needs to check the status bytes in the memory to decide if it can process a new frame.

The arbiter-controlled model uses the same bus as the processor-controlled model except the arbiter handles both the arbitration and processing control functions and each processor and the input are connected to the arbiter by two interconnections. One interconnection is used to send requests to the arbiter and receive responses; the other interconnection is used to send request types. The request types show if a processing stage is finishing or not. The arbiter has a 2-byte register to record the status of each processor and the input, and the register is updated by each processor and the input. The arbiter simply holds the request of a processor or the input if it cannot process.

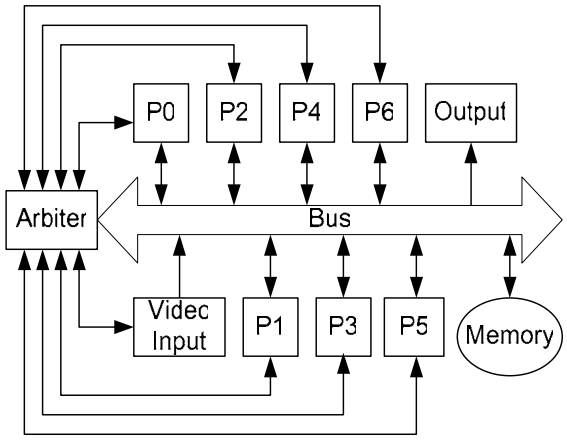

Figure 2. Processor-controlled model

4.2. Switch-based models

Crossbar switches are used to implement the switch-based models (Figure 3). All switches are input-buffered, and the buffers connect to a NxN crossbar, where N is the number of switch ports. A buffer has 55 bits. N is 10 when using a single shared memory and 11 when using 2 shared memories. A control unit handles both the arbitration and processing control functions. We simulated different port widths, which are 5-bit/port, 8-bit/port, 16-bit/port, 32-bit/port, and 55-bit/port.

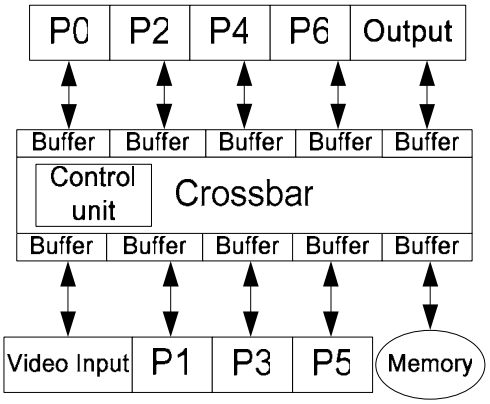

Figure 3. Switch-based model using 10x10 crossbar

A switch uses packets to communicate with the processors, the input, the output, and the memory. Data, address, and control information are capsulated in a packet. There are 4 types of packets: write packet, read packet, switch response packet, and read response packet. A write packet includes 3-bit control, 21-bit address, and 32-bit data. A read packet contains only 3-bit control and 21-bit address. In a read response packet, there are only 32-bit data for a read operation. In the 3-bit control information, one bit is for the request, one bit is the memory control information to show the packet is for a read or write operation, and the other bit is the request type to show if a processing stage will be finished after this request. In a switch response packet, there is only 1-bit information to tell a processor or the input that the switch is sending its packet in current clock cycle.

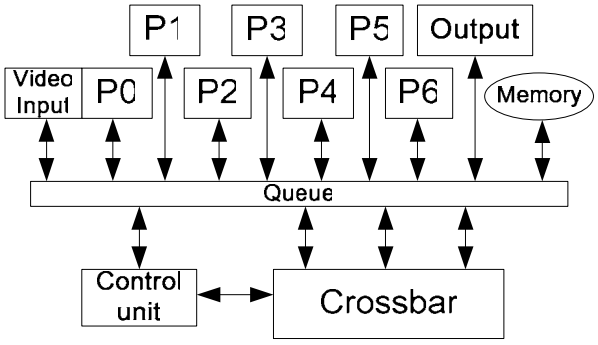

Figure 4. Switch-based model using 3x3 crossbar

We also built switch-based models using two shared memories. To use two shared memories, the 10x10 crossbar is replaced by an 11x11 crossbar in figure 3. Processors 0, 2, 4, and 6 share memory 0, and processors 1, 3, and 5 share memory 1. The input sends frames to memory 1, if and only if the memory 0 has an unprocessed frame. We also build a switch-based model using a 3x3 crossbar and an input queue (Figure 4). The size of the queue is 550-bit when using a single memory and 605-bit when using two shared memories.

5. Simulation environment and method

We develop our own trace-driven methodology for NoC design. Our method uses a hierarchy of models, starting with telecom-style models and working down to circuit models.

5.1. Simulation environment

We adopted OPNET [12] for on-chip communication architecture simulations. OPNET is a matured telecommunication system simulation environment, and we find it can be used to simulate on-chip systems with some adaptations. OPNET has some advantages for on-chip communication analysis. First, it can model many common phenomena in communication systems. Second, it is fast. Simulation speed is very important for on-chip communication architectures, which will be simulated for billions of clock cycles to get meaningful application-level results. Third, it is a packet-based simulator. This feature is particularly useful for our research because we are simulating packet switching.

OPNET also has some disadvantages. First, it is developed for simulating telecommunication systems and some on-chip features are not well supported. For example, the smallest time unit is the second, where on-chip communication architectures need nanosecond or even picosecond, and the smallest distance unit is meter instead of micrometer. Second, OPNET assumes asynchronous communication, so for a synchronous system, designers have to explicitly design a clock scheme and a distribution networks.

We made the following adaptations in OPNET. In link models, 1) disable the propagation delay pipeline stage and 2) disable the error model. In transmitter and receiver models, set data rate high enough to eliminate the effects of transmission delay. (We set data rate to 10^6 bps, which introduces 1 ms transmission delay.) In all node models, state transitions should be on clock edges. For the clock, 1) use one second to represent one clock cycle and 2) build a clock bus to synchronize the system.

5.2. Simulation level and communication trace

We simulated our designs using a cycle-accurate architecture simulator. Architecture level simulations give designers a quick evaluation independent of circuit implementations. The delay or processing time of each architecture element is based on the complexity of its functions and some reference designs. We assume that an bus arbiter or switch control unit needs one clock cycle to make a decision; memory read takes 3 clock cycles and memory write takes 2 clock cycles[11]; bus and long interconnects have one clock cycle delay. Because all the models obey these rules, the relative performances are accurate.

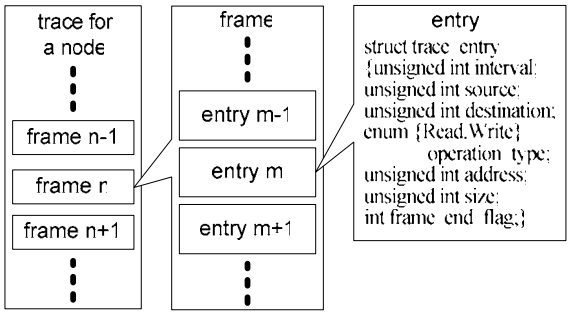

Figure 5. Recorded communication trace

Since the on-chip networks cannot be co-simulated with the computation architecture in OPNET, we recorded real communication traces (Figure 5). The communication traces are recorded on a processor simulator for each processing stage and each frame. In this way, we assume an ideal on-chip network, where communication of one processing stage only affects the other stage at the beginning and the end of a

frame, but not during a frame. Each processor and the video input have their own traces. A trace has entries to record network accesses for each frame. An entry includes access interval (between current and the last access) in term of number of clock cycle, source, destination, operation type, address, data size, and flag to show the end of a frame. Those traces are used to control the communication behaviors of according processors in the models.

5.3. Circuit models

We designed some circuit models to determine the timing restriction. A NoC includes two types of circuits: logic circuits and interconnections. Bus arbiter and switch control unit are logic circuits. The bus and the point to point interconnections in the switch are interconnections. Crossbar has a little logic circuit and is mainly interconnections. Other interconnections are between the arbiter and the processors and the video input. An interconnection includes an input driver, a wire, and an output driver. The input driver is usually a chain of sized inverters. Transmission gates will be used to connect input drivers to a wire, if multiple input drivers are used. Both the bus and point to point interconnections have multiple input buffers.

6. Simulation results and analysis

We simulated each model in OPNET using recorded communication traces for $3x10^8$ clock cycles. We assume the on-chip communication architectures are synchronous and running at the same speed as the processors. This is an optimistic assumption for a bus, while a switch can easily fulfill this assumption. A bus stretches over a chip to connect multiple bus masters and slaves, and it usually run at lower speed than processors. However, for a switch, point to point interconnects are shorter and thinner than bus interconnects, and it can run at the same speed as processors.

6.1. Definitions

We use the following definitions for related metrics. The system performance is the number of frames being processed in a fixed time. The network throughput of a bus or a switch is the number of bits transferred by the bus or the switch in a clock cycle. The network utilization is defined by formula (1).

$$Utilization = \frac{Average \quad throughput}{Maximum \quad throughput} \qquad (1)$$

Because the system need to process 150 frames per second, we use the formula (2) to get the required system frequencies.

$$System \quad Frequency = \frac{3\times10^8 \times 150}{\mathrm{Pr}\, ocessed \quad Frames} \qquad (2)$$

6.2. Network throughput and utilization

When using the same number of shared memories, the crossbars with wider ports have higher throughput but lower utilization (Figure 6 and 7). Higher throughput usually gives better system performance, while lower utilization wastes network resource (Table 1 and 2). When the port width is 16-bit/port, the crossbar has similar throughput as the arbiter-controlled system, and the network utilization is only 7.2%, comparing to 18.7% of the arbiter-controlled system. The 16-bit/port 3x3 crossbar with two shared memory has the highest utilization 44.2%, while the system performance is 85.7% higher than the arbiter-controlled model.

Table 1. Network throughput and utilization

Model Name	Maximum throughput (bit per clock cycle	Average throughput (bit per clock cycle)	Network utilization
32-bit/port 11x11 crossbar with 2 memories	352	27.0	7.7%
16-bit/port 11x11 crossbar with 2 memories	176	22.0	12.5%
16-bit/port 3x3 crossbar with 2 memories	48	21.2	44.2%
55-bit/port 10x10 crossbar	550	21.0	3.8%
32-bit/port 10x10 crossbar	320	18.9	5.9%
8-bit/port 11x11 crossbar with 2 memories	88	13.6	15.5%
16-bit/port 10x10 crossbar	160	11.5	7.2%
16-bit/port 3x3 crossbar	48	11.4	23.8%
Arbiter-controlled	54	10.1	18.7%
Processor-controlled	54	10.2	18.9%
8-bit/port 10x10 crossbar	80	7.3	9.1%
5-bit/port 10x10 crossbar	50	4.7	9.4%

Table 2. System performance

Model Name	Performance (processed frames in 3x108 clock cycles)	Performance improvement (refer to the arbiter-controlled model)	Required frequency (MHz) to reach 150 frame per second
32-bit/port 11x11 crossbar with 2 memories	130	132.1%	346
16-bit/port 11x11 crossbar with 2 memories	106	89.3%	425
16-bit/port 3x3 crossbar with 2 memories	104	85.7%	433
55-bit/port 10x10 crossbar	102	82.1%	441
32-bit/port 10x10 crossbar	91	62.5%	495
8-bit/port 11x11 crossbar with 2 memories	67	19.6%	672
16-bit/port 10x10 crossbar	57	1.8%	789
16-bit/port 3x3 crossbar	56	0	804
Arbiter-controlled	56	reference	804
Processor-controlled	49	-12.5%	918
8-bit/port 10x10 crossbar	36	-35.7%	1250
5-bit/port 10x10 crossbar	23	-58.9%	1957

6.3. System performance

The simulation results (Table 2) show that in $3x10^8$ clock cycles the arbiter-controlled model processed 56 frames, while the processor-controlled model processed only 49 frames, which is a 12.5% performance decrease. Although using the same bus, in processor-controlled system, the processors and the input need to communicate

with memory to handle the processing control function, and those control communications reduce the effective bus throughput for data communications.

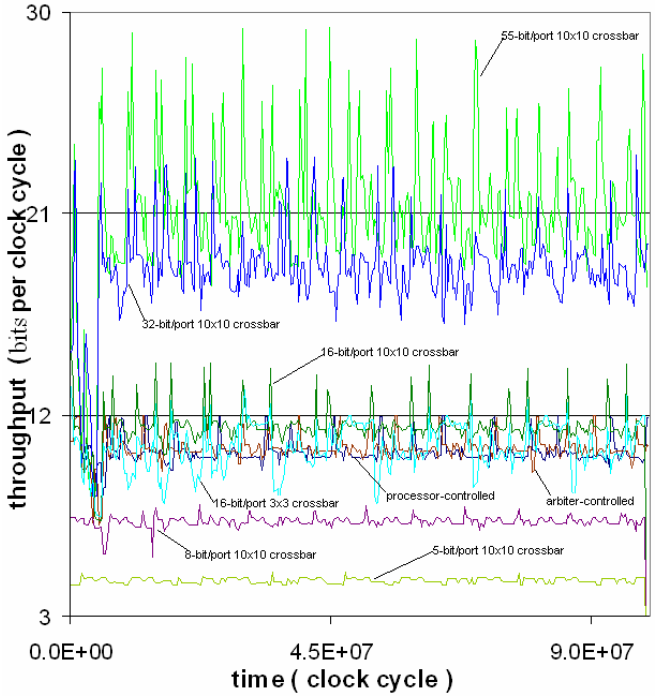

Figure 6. Network throughput using a single shared memory

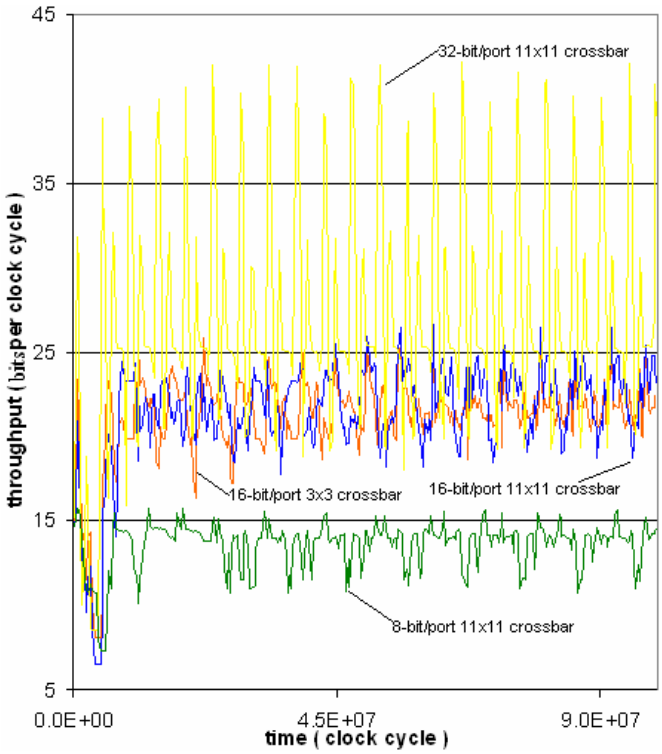

Figure 7. Network throughput using two shared memories

If the number of shared memories is the same, switch-based models with wider ports have higher system performances (Table 2), higher maximum throughputs, and higher average throughput (Table 1), and they have lower transmission clock cycles (Table 3). Except the 5-bit/port and 8-bit/port 10x10 crossbars, all the models using crossbars have higher performance than the models using bus. High effective throughput and low latency make switch-based models have higher performance than bus-based models. Switch-based models with two shared memories have higher system performance than those with a single shared memory. Because all the processors and the input need to access memory for data, memory is a performance bottleneck. Two shared memories increase both the system performance and average throughput by doubling memory bandwidth.

Table 3. Clock cycles needed to transmit

Model Name	Clock cycles needed to transmit 32-bit data, 21-bit address, and 1-bit or 2-bit control
55-bit/port 10x10 crossbar	1 (2-bit control)
32-bit/port 10x10 crossbar	2 (2-bit control)
32-bit/port 11x11 crossbar with 2 memories	2 (2-bit control)
16-bit/port 10x10 crossbar	4 (2-bit control)
16-bit/port 11x11 crossbar with 2 memories	4 (2-bit control)
16-bit/port 3x3 crossbar	4 (2-bit control)
16-bit/port 3x3 crossbar with 2 memories	4 (2-bit control)
Processor-controlled	1 (1-bit control)
Arbiter-controlled	1 (1-bit control)
8-bit/port 10x10 crossbar	7 (2-bit control)
8-bit/port 11x11 crossbar with 2 memories	7 (2-bit control)
5-bit/port 10x10 crossbar	11 (2-bit control)

In all the models which have a maximum throughput around 55 bits per clock cycle, the 3x3 crossbar model with two shared memories has the highest system performance and the lowest maximum throughput (Table 2). The 3x3 crossbar model with a single shared memory has the same performance as the arbiter-controlled model. The 5-bit/port 10x10 crossbar model with a single shared memory has the lowest performance.

Overall, the 3x3 crossbar model with two shared memories is a good choice for Smart Camera SoC. It only needs a 443MHz system frequency to reach 150 frames per second system performance. Since the 3x3 crossbar has the lowest maximum throughput, it will use relatively small area.

7. Conclusion

Our study shows the NoC is a performance bottleneck in our case, and it is also possible in other designs using the similar communication architecture. Using the Smart Camera SoC as a case study, this research explores bus and crossbars with different sizes, different port widths, and different numbers of shared memories. Larger crossbar (or

crossbar with wider ports) has higher average throughput and higher system performance. High average throughput usually associates with high maximum throughput and high system performance, but with low network utilization. For the Princeton Smart Camera system, we find the 16-bit/port 3x3 crossbar with two shared memories shows an 85.7% performance improvement over the processor-controlled model, while it has lowest maximum throughput. And the 16-bit/port 3x3 crossbar with a single shared memory has the same system performance and lowest maximum throughput as the processor-controlled model. We adopted the telecommunication system simulator, OPNET, and simulate the cycle-accurate architecture-level model of each NoC. All the simulations use the recorded communication traces, which give better accuracy. This method is only need less than one day to finish $3x10^8$ clock cycle simulation for each model.

References

[1] L. Benini, G. De Micheli, "Networks on chip: a new paradigm for systems on chip design", *Design, Automation and Test in Europe Conference*, 2002.

[2] William J. Dally, Brian Towles, "Route packets, not wires: on-chip interconnection networks", *Proceedings of the 38th Design Automation Conference*, 2001.

[3] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage, "Networks on Silicon: Combining Best-Effort And Guaranteed Services" *Design, Automation and Test in Europe Conference*, March, 2002.

[4] R. Hofmann, B. Drerup, "Next generation CoreConnect processor local bus architecture", *Annual IEEE International ASIC/SOC Conference*, 25-28 Sept. 2002.

[5] D. Flynn, "AMBA: enabling reusable on-chip designs", *IEEE Micro*, Volume: 17 Issue: 4, July-Aug. 1997

[6] D. Wingard, "MicroNetwork-based integration for SOCs", *Design Automation Conference*, 18-22 June 2001.

[7] T. Dumitras, R. Marculescu, "On-chip stochastic communication" *Design, Automation and Test in Europe Conference*, 2003.

[8] M. Galles, "Spider: a high-speed network interconnect", *IEEE Micro*, Volume: 17 Issue: 1 ,1997, Page(s): 34 -39.

[9] A. Hemani, A. Jantsch, ect, "Network on chip: An architecture for billion transistor era", *Proceeding of the IEEE NorChip Conference*, November 2000.

[10] Wayne Wolf, Burak Ozer, and Tiehan Lv, "Smart cameras for embedded systems," *IEEE Computer*, 35(9), September 2002, pp. 48-53.

[11] Micron 18Mb SYNCBURST SRAM specification, http://www.micron.com

[12] http://www.opnet.com

[13] http://www.silicore.net

[14] Wayne Wolf, Tiehan Lv, and Burak Ozer, "An Architectural Design Study for a High-Speed Smart Camera", *IEEE MICRO MSP-02 Workshop*, 2002