FIFO Power Optimization for On-Chip Networks

Sudarshan Banerjee Nikil Dutt
banerjee@ics.uci.edu dutt@ics.uci.edu
Architectures and Compilers for Embedded Systems (ACES) Laboratory
Center for Embedded Computer Systems
University of California, Irvine, CA, USA
http://www.cecs.uci.edu/˜aces

Technical Report #03-40
Dept. of Information and Computer Science
University of California, Irvine, CA 92697, USA

Dec 19, 2003

Abstract

As the design community moves towards architecting multiprocessor systems-on-chip (MPSoC), it is widely believed that an on-chip interconnection network is potentially the best candidate to satisfy the high aggregate throughput needed by dozens of IP blocks. In this context, power (energy) estimation and reduction techniques for switches and links, the core components of an interconnection network, gain added significance. FIFO buffers are a key component of a majority of network switches - buffers have been estimated to be the single largest power consumer for a typical switch in an on-chip network. In this report, we analyze energy-power characteristics of FIFOs for on-chip networks and propose an optimization to reduce FIFO energy consumption in the context of an on-chip network. Our experimental results demonstrate promising reductions in energy consumptions (19-33% for 256 and 512 bit wide links). Furthermore, our approach yields increasing energy reduction for wider links that are very likely to be used in future on-chip networks.

Key words: shared memory, switches, FIFO, networks on chip, wide flits, low power design

Contents

List of Figures

1 Introduction

With rapidly decreasing feature size, it has become more feasible to integrate dozens of IP blocks into a Systems-on-Chip (SoC). Shared-medium buses are very unlikely to meet the communication requirement of such systems: this has motivated a series of proposals for generic interconnection templates, often referred to as on-chip networks [4], [3]. Power-energy issues play a very fundamental role in SoC design. With interconnects beginning to dominate the power-energy performance of SoCs, it becomes critical to consider the interconnection paradigm for SoC components.

When considering power-energy issues in an on-chip network, buffers in network switches are potentially the most important components. For instance, input buffering is a common organization for many modern-day switches like the Alpha 21364 router [12]. Buffers in such a switch are typically organized as one or multiple FIFO's per input port (wormhole, virtual channels, etc). It has been estimated that these input buffers contribute 46-61% of total power in a switch modelled on the Alpha 21364 router [1]. This provides the motivation for us to analyze and optimize the power-energy consumption of FIFOs in the context of input buffered switches for an on-chip network.

A distinguishing feature of an on-chip network is very wide links, (typically 256 bits wide or more), which are not feasible in off-chip networks due to significant overheads. However it is recognized [4] that realistically all data transfers on the network are unlikely to be so wide. We propose a technique to optimize energy consumption in FIFO buffers under the assumption of wide data transfers in a high performance on-chip network. We utilize a key property of a network interconnecting the various components of a shared-memory based MPSoC : a significant number of transactions need to carry only limited information. This property enables us to achieve significant reduction in energy consumption for the FIFO buffers: over 33% improvement for buffers in networks with 512 bit wide links. An important aspect of the optimization is that, assuming the control information doesn't increase significantly, the reduction scales well as the flit width increases. This feature makes the opimization even more viable in future on-chip networks.

The rest of this report is organized as follows: in Section 2, we review related research in on-chip networks and power estimation. In Section 3 we summarize some basic, yet very relevant concepts in network switches and shared memory systems. In Section 4, we describe a typical FIFO buffer organization and present our proposed optimization. In Section 5, we present the experiments conducted. We conclude with Section 6.

2 Related work

Design of the communication architecture is one of the most important issues in architecting a MpSoC. This requires interconnecting multiple IP blocks such as multimedia processors with I/O requirements in the range of Gbit/s. It is very unlikely that bus-based architectures would be able to meet this requirement, and this provides impetus for research into on-chip networks [4], [3], [15].

Research in on-chip networks quite naturally leverages the existing significant body of work in the domain of interconnection networks for parallel computer architectures [18]; this body of

work has in turn been built upon extensive research in packet switches for more general purpose computer networks like LANS, the Internet, etc [20]. Traditional research in computer networks has been driven primarily by performance issues. However, issues related to power-energy consumption take on a much more significant role in the context of an on-chip network.

There exists a large body of work related to power estimation and power optimization techniques for processors and memories [13], [14], [7], [6]. The field of power-oriented investigations into interconnection networks is less mature in comparison [16], [17]. More specifically, given that the concept of an on-chip network is still in its nascent stages, there are relatively few studies focussed on power-related issues in such a network [10], [1], [2].

In [1], architectural power models are proposed for various switch components including arbiters, cross-bars and FIFOs: average and peak power for a network switch is analytically estimated based on flit arrival rate. In [10], switch level power models based on bit transitions are used to study relationship between packet size and cache characteristics such as cache miss penalty, cache miss rate, etc, in a distributed shared memory framework. In [2], power dissipation of network micoarchitectures is analyzed in a CMP(chip multiprocessor) context and some power efficient microarchitectures, segmented crossbar, write-through buffer, etc are proposed.

In our work, the power models of a FIFO are similar to [1], essentially based on [7], [6] with some minor modifications. Similar to [10], we use the distributed shared memory framework RSIM [9], for conducting our experiments. We complement the efforts of [10], [1], [2] by focusing on the correlation between energy consumption of SRAM-based FIFO buffers and network data width- this enables us to propose an optimization that achieves significant energy savings without any performance overhead. Our approach scales well with increasing network width, a key feature expected of future on-chip networks. Furthermore, our approach characterizes the energy savings under real network traffic from application benchmarks, whereas the study in [2] observed much smaller energy savings due to network contention.

3 Background in switches and shared memory systems

Figure 1 shows the structure of a basic input-buffered switch in an interconnection network. Flits [1] arriving on the input ports are stored in buffers till a path through the crossbar is made available by the scheduler. The scheduling mechanism essentially handles contention-related issues such as flits arriving at different input ports simultaneously requesting the same outgoing link. A flit successfully traversing the crossbar is sent to the appropriate output port based on the address computed by the routing logic from control bits embedded in the flit.

CC-NUMA (cache-coherent nonuniform memory) [5] is a commercially popular approach to designing scalable shared memory systems. Typically, one or more processors with local caches access a local main memory on a shared bus, and, multiple such nodes are interconnected via a network. Accesses to all non-local data get translated into network transactions, and, cache coherence is maintained by replicating data in caches instead of in local main memory. A key aspect of protocols such as MSI (Modify-shared-invalid), MESI (Modify-exclusive-shared-invalid) [5] dealing with cache states and their transitions is the fact that a cache access fault could potentially trigger

[1] a flit is the logical unit of data transfer in an interconnection network- typically flit width equals link width

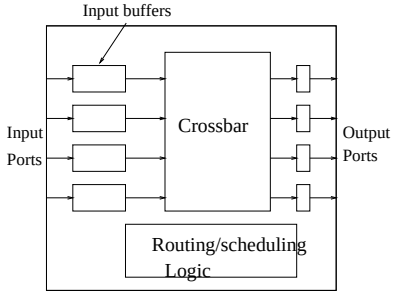

Figure 1. Basic input-buffered switch

multiple network transactions, some of which just need to transfer the state transition information. This information remains constant even when the volume of data transfer in a network transaction increases- in an on-chip network this corresponds to increase in link width, and, width of data flit (a flit carrying actual data).

Studies have shown that the input buffers, typically structured as FIFOs, are potentially the single largest energy consumer in a typical switch [1]. This motivates our investigation into energy consumption of FIFOs and results in our proposal to modify the FIFO structure for reduced energy consumption.

4 Power analysis and optimization of FIFO buffers

For the rest of this paper we use the term power to mean the average energy consumption at a particular instant (i.e, average power as opposed to peak power).

4.1 FIFO Buffer Organization

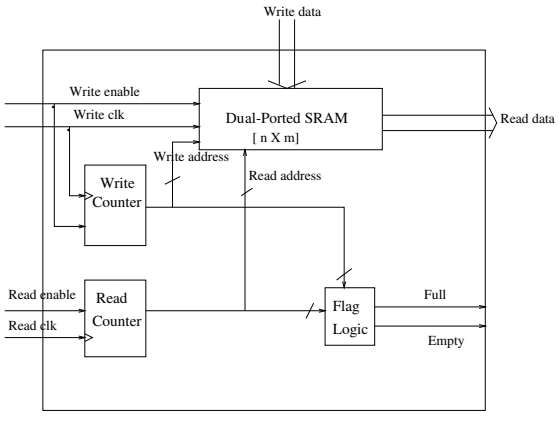

Figure 2. FIFO

Figure 2 shows the block diagram of a typical SRAM based *n X m* FIFO that can store at most *n* flits (words) each of width *m* bits. On a *read*(*write*), the flit is written into (read from) the

appropriate row of the SRAM, with the row indexing done based on the addresses generated from a read(write) counter. The read and write addresses are also sent to some flag logic which generates *full* and *empty* signals interpreted by external logic to prevent *write*s during *full* or, *read*s during *empty*- the external logic is simply the switch control logic in our context.

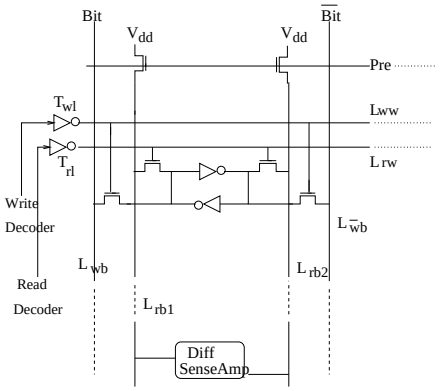

Figure 3. RAM

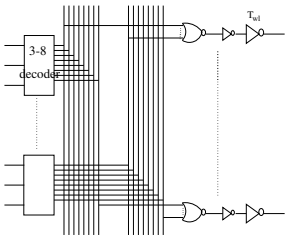

Figure 4. DECODER

We next describe the internal structure of the SRAM. This is a matrix of *n* rows with *m* memory cells in each row, as shown in Figure 3. As mentioned above, there are two separate decoders which drive the read and write wordlines on the appropriate row of the memory array based on the addresses generated by the corresponding counters. The decoder structure, shown in Figure 4 is assumed to be similar to discussions in [6]. Essentially a $log(n)$ bit address is input to $log(n)/3$ decoders, each one a 3-8 decoder implemented as NAND gates- the decoder outputs drive NOR gates, one NOR for each wordline. For the memory cell, we assume a structure with a double-ended write bitline and a double-ended read bitline terminating in a differential sense amplifier.

A read operation on such a cell is executed by the following micro-sequence: the precharge signal is raised high, charging the read bitlines. This is followed by activation of the read wordline L_{rw} which causes one of the read bitlines to be pulled low depending upon the value stored in the cell. On a write, similarly, the write wordline L_{ww} is activated enabling the cell to store the data available on the write bitlines L_{wb}, $L_{\overline{wb}}$.

4.2 Power estimation of FIFO Buffers

Estimation of power consumption of SRAMs is a well-studied field [7], [6]. In a MOS circuit, most of the power consumption is due to the switching capacitance: i.e, the power consumed by a read or write operation can be estimated with reasonable accuracy from the capacitances of the transistors and metal lines involved in the operation. The estimation process necessarily includes estimation of transistor sizes based on the loads they need to drive, assumptions on appropriate thresholds they need to be driven to, etc. For instance, the precharge transistors need to be large enough to drive the read bitlines.

For our estimation purposes, we use the following set of equations at a high level of abstraction for P_{read} and P_{write}, which represent the power consumptions of a single read and write operation respectively:

$P_{read} = P_{decode} + P_{readwordline} + m * (2 * P_{precharge} + P_{readbitline} + P_{senseamp})$

$P_{write} = P_{decode} + P_{writewordline} + \alpha * (P_{writebitline} + P_{cell})$

where:

• P_{decode} is the power consumption of a decoder in decoding an address. This is estimated from the capacitances of the NAND and NOR gates constituting the decoder

• $P_{readwordline}$ and $P_{writewordline}$ are the power consumptions when a wordline is activated. This is estimated from capacitances of the wordline driver and the metal wordline length

• $P_{readbitline}$ and $P_{writebitline}$ are the power consumptions of the corresponding bitline. This is estimated from the bitline length

• P_{cell} corresponds to the memory cell capacitance

• α is the number of memory cells that change state on a specific write operation.

Detailed estimation of the individual capacitances with suitable technology scaling follow the guidelines of [7], [6] with some minor modifications. An example of a modification: in [6], for the wirelength estimation needed to compute decoder capacitances, it is assumed that there are multiple memory arrays arranged as 2 X 2 blocks with the decoder in the middle. However, we assume a single memory array which results in a different wirelength estimate and a corresponding change in the capacitance and power estimate.

4.3 Optimized FIFO Buffer Organization

From our prior discussions of a shared memory MPSoC, as the width *m* of a data flit (or, number of columns in the memory array) grows without a corresponding increase in the control flit size, a significant amount of power is wasted for control flits- this wastage is essentially due to extra capacitance on the long wordlines for both read and write operations. Additionally, for read operations, there is significant power wastage in precharging the bitlines for cells that don't contain any useful information. Thus we can reduce power consumption in control flits for $m - C$ bits where width of control flit is C bits.

We achieve this by modifying the basic FIFO structure as shown in Figure 5. Additionally, we modify the SRAM structure as shown in Figure 6. Essentially we

(a) split the memory array into **RAM1**, a *n X C* array, and **RAM2**, a *n X (m - C)* array

(b) gate the wordlines, precharge lines of **RAM2** by appropriate signals as described later.

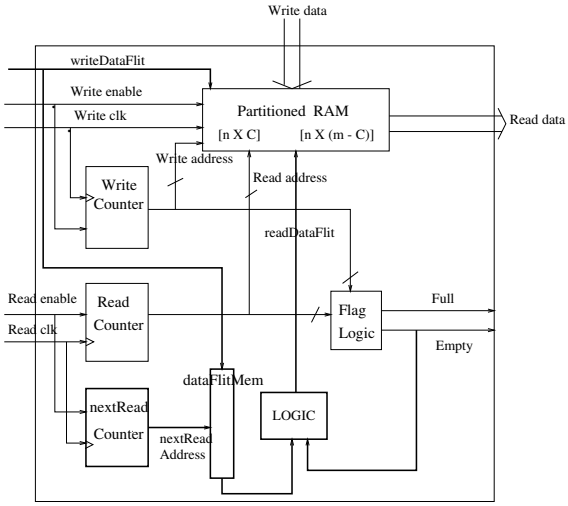

Figure 5. optimized FIFO

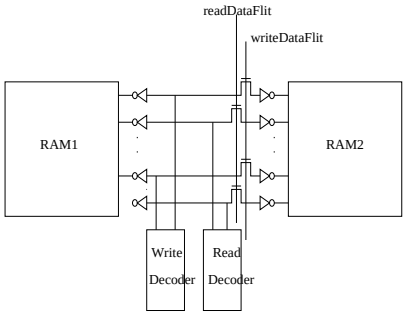

Figure 6. partitioned RAM

The modified FIFO functions in the following manner:

In the delay model of a pipelined wormhole router as in [19], when a flit arrives at a switch, it is buffered in the input queue: the head flit of a packet goes through the states of *routing, switch arbitration* and *switch traversal* while successive flits simply wait in the input queue till a buffer becomes available on the next hop. Along with checking whether an incoming flit is a head flit, we add a simple comparison to check if the incoming flit is a data flit or control flit, and the result of the comparison is the new control signal $writeDataFlit$. This signal $writeDataFlit$ gates the write wordlines on **RAM2**. Thus, for a control flit, only *C* bits are stored in **RAM1**- for a data flit, the first *C* bits are stored in **RAM1** and the remaining *(m - C)* bits are stored in **RAM2**. Also, the value of the signal $writeDataFlit$ is stored in the *n X 1* memory **DataFlitMem** at the current write address generated by the **write counter**.

Similarly, while retrieving the flit from the FIFO, $readDataFlit$ gates the read wordlines of **RAM2**. This signal is essentially the bit previously stored in **DataFlitMem** on a write at the corresponding address. The logic driving $readDataFlit$ can be succintly described as:

if ($FIFO\_empty$)
    $readDataFlit$ = $writeDataFlit$
else
    $readDataFlit$ = **DataFlitMem [nextRead]**

The **nextRead counter** is functionally similar to the **read counter**. However, instead of generating the current read address for the FIFO, it generates the address for the next entry stored in the FIFO. As an example, if the **read counter** was a simple binary up counter with a current count of *i*, the current count generated by **nextRead counter** would be $(i+1) modulo (maxcount+1)$. The signal $readDataFlit$ also gates the precharge logic for **RAM2** to take care of the fact that the precharge transistors are significant power consumers.

4.4 Analysis of optimized FIFO

4.4.1 Overhead for power

We now analyze the overhead incurred by our proposed optimization. In terms of additional logic estimated from transistor count, the overhead is minimal. Overhead for the additional counter and control logic is insignificant compared to any reasonable SRAM. The introduction of a new one bit memory array and gating transistors cause an estimated area overhead of less than 1% for a *64 X 128* SRAM. This very small fraction effectively disappears as data flit width increases.

When we consider power issues, we have of course reduced significantly (almost eliminated) the power consumption corresponding to the unused bits in a control flit. But on the flip side, with our scheme every read and write operation on the FIFO now has an extra memory access to the 1-bit wide memory **DataFlitMem** increasing the power consumption on every access.

On every write access to the partitioned RAM, the same decoded address is used to access the **DataFlitMem** location where $writeDataFlit$ is stored- so, we don't need a write decoder for **DataFlitMem**. However, on a read access to the partitioned RAM, the decoded address is different from the address to retrieve $readDataFlit$: so we need a separate read decoder for **DataFlitMem**

These observations lead to the following equations representing the power consumption of an optimized FIFO:

For data flits, (utilizing the entire width of the original SRAM)

$P_{read} = 2 * P_{decode} + P_{readwordline}(RAM1) + P_{readwordline}(RAM2) + (m+1) * (2 * P_{precharge} + P_{readbitline} + P_{senseamp})$

$P_{write} = P_{decode} + P_{writewordline}(RAM1) + P_{writewordline}(RAM2) + (1+\alpha) * (P_{writebitline} + P_{cell})$

The equations make a pessimistic assumption that any read or write on **DataFlitMem** causes the addressed cell to change state.

With similar assumptions for control flits, we have

$P_{read} = 2 * P_{decode} + P_{readwordline}(RAM1) + (C+1) * (2 * P_{precharge} + P_{readbitline} + P_{senseamp})$

$P_{write} = P_{decode} + P_{writewordline}(RAM1) + (1+\alpha) * (P_{writebitline} + P_{cell})$

4.4.2 Timing issues

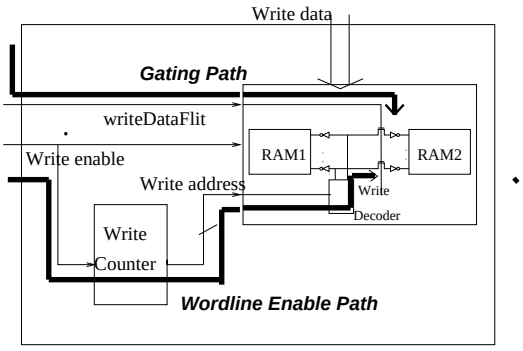

Figure 7. Signal propagation Paths

From the preceding description it is not immediately clear if the optimized FIFO functions correctly when we consider timing issues. For the write operation, if *writeDataFlit* and the *writeEnable* signal are issued simultaneously there seems to be the possibility that the wordline might be incorrectly enabled or disabled. Note that once the *writeEnable* signal is activated by the FIFO controller, the write wordline corresponding to the current flit can get activated only after propagation delay through the write counter and the delay through the decoder (illustrated by **Wordline Enable Path** in Figure 7). However, *writeDataFlit* propagates only on metal wire and would always gate the wordline correctly.

We next consider the read operation. Obviously when reading a flit it is not practical to fetch the stored bit (indicating whether it is a data or control flit) in the same cycle: this would imply 2 consecutive memory read accesses in one cycle and would potentially have a severe impact on the circuit delay by stretching the clock cycle. To get over this obstacle, we have used the simple property of the FIFO that it would be always possible to calculate the address of the location to be read next. Thus, when doing a read access to retrieve a flit, in the same cycle we fetch the stored bit

corresponding to the next read address; by the time the next flit is to be retrieved, *readDataFlit* has the correct value. Note that for the simple case of a FIFO read counter implemented as a binary up counter, a simple incrementer (adder) would generate the next read address. However, often counters in FIFOs are implemented as Gray or Johnson counters and therefore, a simple adder may not suffice to generate the next read address. That motivated the addition of the **nextRead counter** to the optimized FIFO shown in Figure 5.

5 Experiments

The platform for our experiments was the RSIM simulator [9], widely used in academia for investigations into shared-memory multiprocessor systems. In RSIM, the individual processors aggressively exploit ILP (Instruction level parallelism). The flit-level interconnect network submodule in RSIM models a wormhole router with various configurable parameters like the flit width, depth of FIFO's in the network switches, cache-coherence protocols, etc.

We classified the different categories of network transactions in RSIM, and hence their corresponding flit types, into data or control. For example, flits corresponding to *invalidation* transactions are categorized as control flits while flits corresponding to *readexclusive* transactions are categorized as data flits. Control information in RSIM needs 128 bits: for the default network width/flitsize of 8 bytes, each control transaction generates two control flits.

Given the power estimate for a single FIFO, we modelled the FIFO energy consumption of a complete interconnection network by a minor modification to the RSIM simulator to aggregate the data from all input FIFO's for all switches in the interconnection network.

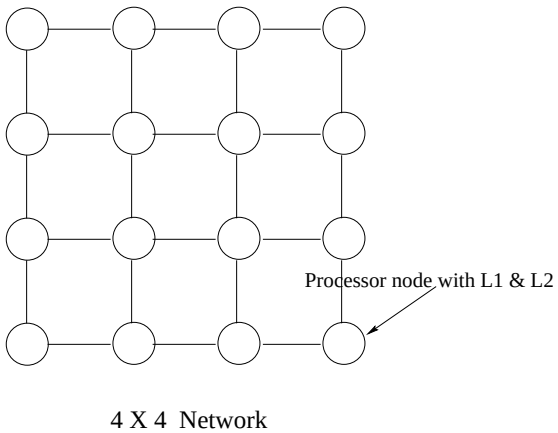

Figure 8. mesh

Our base configuration, as shown in Figure 8 was 16 processors arranged in a 4 X 4 mesh, as suggested in [4]. In RSIM, each processor is modelled with a L1 and L2 cache and is connected to the network through a network interface.

For conducting our experiments, we chose the technological parameters to match those of the Alpha router [12], i.e

Feature size: 0.18 microns,

Voltage: 1.65 V,
Frequency: 1.2 GHz

For each *n X m* FIFO we generated power consumption data from the equations outlined in **Sec 4.2** and **Sec 4.4.1**: *n* (FIFO depth) and *m* (flit width) are the primary experimental parameters. This data represents the write and read power consumed when a single flit is stored into, and retrieved from a FIFO. The data was computed for both the original, unoptimized FIFO and, the optimized FIFO. We made a pessimistic estimate for the control flit that all *C* bits of control change every time.

5.1 Analytical estimates

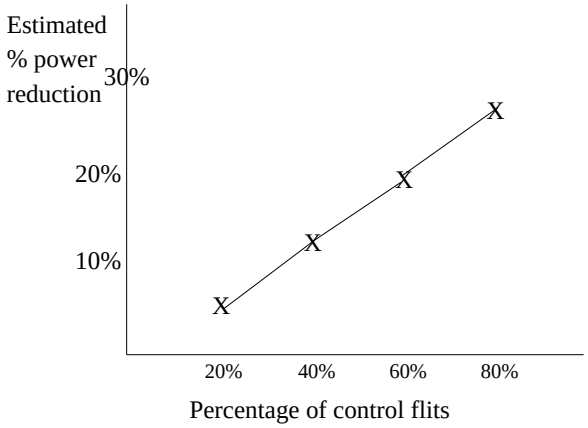

Figure 9. 64x256 FIFO

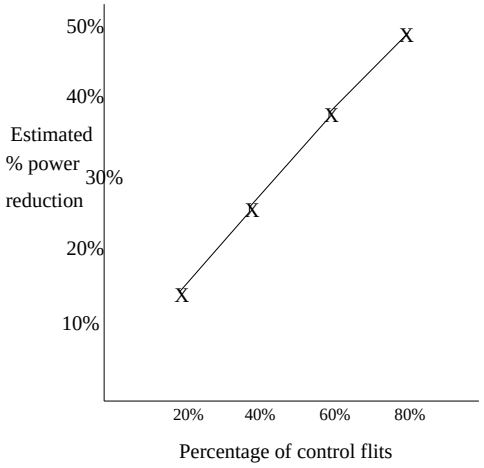

Figure 10. 64x512 FIFO

We first analytically plotted the potential power reduction against the control flit percentage for some FIFOs with different parameters. In each of the graphs (Figures 9-12), the X-axis represents

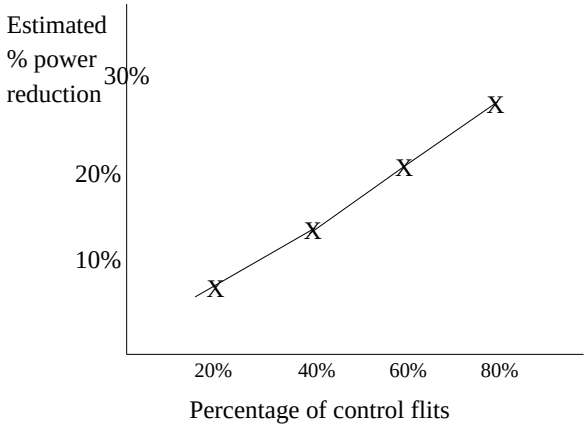

Figure 11. 128x256 FIFO

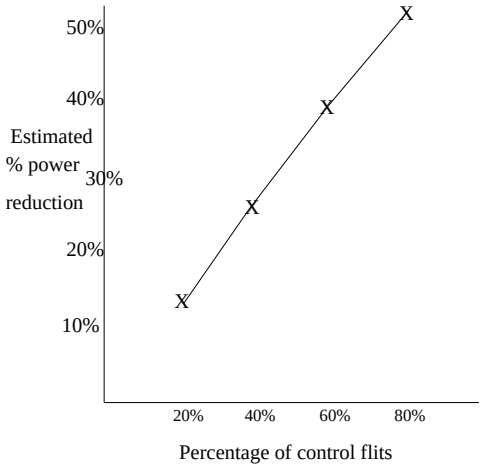

Figure 12. 128x512 FIFO

the percentage of control flits out of the total number of flits that are read from and written into the FIFO; the Y-axis represents the potential power reduction as a percentage of the original power consumption. The power estimates are based on the assumption that on an average half of the memory cells switch on a write, i.e $\alpha = 0.5$.

Figure 9 represents the data for a $64X256$ FIFO where the width of the control flit is 128. Figure 10 represents the data for a $64X512$ FIFO where the width of the control flit is 128. Figure 11 represents the data for a $128X256$ FIFO where the width of the control flit is 128. And, Figure 12 represents the data for a $128X512$ FIFO where the width of the control flit is 128.

From the plots, it is apparent that the percentage power reduction increases linearly with increase in percentage of control flits. Also, comparing Figure 9 with Figure 10, we clearly see that the estimated power reduction for a wider FIFO, i.e the $64X512$ FIFO, is significantly more than the FIFO with less wide flits, the $64X256$.

5.2 Benchmark results

We next ran experiments on an interconnect network for a distributed shared memory multiprocessor to obtain data in realistic scenarios- for example, we assessed the effect of the cache coherence protocol on the power reduction.

We chose the following set of benchmarks from the popular Stanford SPLASH suite [8]: *quicksort, SOR, water, lu, mp3d*. For each benchmark, we ran the simulator to generate the cumulative energy consumption of all FIFOs in the interconnection network switches over the entire execution period. The data in each table shows the percentage reduction in FIFO energy consumption for an interconnection network built with optimized FIFOs in the switches as compared to the FIFO energy consumption for a network built with unoptimized FIFOs in the switches.

The initial set of data shown in **Table 1** was generated with the default FIFO depth of 64, the default MESI cache-coherence protocol, and flit width of 256 bits. For the next experiment shown in **Table 2**, the flit width was changed to 512. As expected, with increasing flit width and constant width of the control information, the percentage improvement increases significantly from a mean of 19% to a mean of 35%.

Benchmark	% reduction in FIFO energy
lu	13%
mp3d	19%
quicksort	20%
sor	19%
water	21%
MEAN	19%

Table 1. FIFO depth 64, flit width 256, MESI

To assess the effect of cache coherence protocols, the next set of experiments shown in **Table 3** and **Table 4** were conducted with the MSI protocol. We expected a larger percentage improvement with our optimization on the MSI protocol given that the number of control transactions is expected

Benchmark	% reduction in FIFO energy
lu	27%
mp3d	35%
quicksort	38%
sor	37%
water	39%
MEAN	35%

Table 2. FIFO depth 64, flit width 512, MESI

to be higher with this protocol than with MESI. The performance data does show an improvement of 22% vs 19% (with MESI) for the *sor* benchmark. However, for most benchmarks the difference is negligible.

Benchmark	% reduction in FIFO energy
lu	13%
mp3d	19%
quicksort	21%
sor	22%
water	21%
MEAN	19%

Table 3. FIFO depth 64, flit width 256, MSI

Benchmark	% reduction in FIFO energy
lu	26%
mp3d	35%
quicksort	39%
sor	42%
water	39%
MEAN	36%

Table 4. FIFO depth 64, flit width 512, MSI

The final set of experiments were conducted to observe the impact of modifying the FIFO depth, i.e, the number of rows, on the percentage improvement. **Table 5** and **Table 6** show the results corresponding to FIFO depths of 32 and 128 respectively. These results indicate that the percentage improvements don't change significantly with change in FIFO depth.

Though the experiments were conducted on wormhole routers with simple FIFOs, they are equally applicable to other input buffer organizations, including virtual channel routers. In DAMQ or virtual channel routers, the input buffers are partitioned into multiple subqueues, but each subqueue is a FIFO.

Benchmark	% reduction in FIFO energy
lu	13%
mp3d	18%
quicksort	19%
sor	18%
water	21%
MEAN	19%

Table 5. FIFO depth 32, flit width 256, MESI

Benchmark	% reduction in FIFO energy
lu	13%
mp3d	18%
quicksort	19%
sor	19%
water	21%
MEAN	19%

Table 6. FIFO depth 128, flit width 256, MESI

6 Summary

On-chip networks offer very high performance potential with new features, including the feasibility of very wide links. However, power and energy issues are a significant constraint in moving forward with their realization.

In this work, we focussed on optimizing power of FIFO buffers, which are one of the most power-hungry components of network switches. We studied the traffic characteristics for an on-chip network interconnecting the component processors of a distributed shared memory system. This led us to propose an optimization that yields promising reductions in buffer energy consumptions (19-33% for 256 and 512 bit wide links).

In future work, we will continue to focus on application characteristics to provide us more insight into similar optimizations to improve power-energy characteristics of an on-chip network. Also, in this work we considered energy consumption from dynamic switching only; energy consumption due to leakage currents is becoming very important and is the subject of future work.

7 Acknowledgments

This work was partially supported by NSF Grants CCR-0203813 and ACI-0205712

References

[1] HS Wang, X Zhu, LS Peh, S Malik, ”Orion: A Power-performance Simulator for Interconnection Networks” *IEEE Micro*, 2002.

[2] HS Wang, LS Peh, S Malik, ”Power-driven design of router Microarchitectures in On-chip networks” *IEEE Micro*, 2003.

[3] P Guerrier, A Greinier, ”A generic architecture for On-Chip Packet-Switched Interconnections” *Proc. DATE*, 2000

[4] WJ Dally, B Towles, ”Route packets, not wires: on-chip interconnection networks” *Proc. 38’th DAC*, 2001

[5] DE Culler, JP Singh, ”Parallel Computer Architecture: A Hardware/Software Approach” *Morgan Kaufmann Publishers*, 1998

[6] SJE Wilton, N Jouppi, ”CACTI: An enhanced Cache Access and Cycle Time Model” *IEEE. Jrnl. Solid-State Circuits*, Vol. 31 No 5, May 1996

[7] D Brooks, V Tiwari, M Martonosi, ”Wattch: A framework for architectural-level power analysis and optimizations” *ISCA*, 2000

[8] JP Singh, WD Weber, A Gupta, ”SPLASH: Stanford Parallel Applications for Shared-Memory” *ACM SIGARCH Computer Architecture News*, Vol 20, No 1, 1992

[9] CJ Hughes, VS Pai, P Ranganathan, S Adve, ”RSIM: simulating shared-memory multiprocessors with ILP processors” *Computer* Vol 35, Issue 2, Feb 2002

[10] TT Ye, L Benini, GD Micheli, ”Packetized On-chip Interconnect Communication analysis for MPSoC” *DATE* 2003

[11] A Kumar, LN Bhuyan, ”Evaluating virtual channels for Cache-Coherent shared memory Multiprocessors” *International Conference on Supercomputing* 1996

[12] S S Mukherjee, P Bannon, S Lang, A Spink, D Webb, ”The Alpa 21364 network architecture” *IEEE Micro* Vol 22, No 1, 2002

[13] M Pedram, ”Power Minimization in IC Design: Principles and Applications” *ACM TODAES* Vol 1, No 1, Jan 1996

[14] AP Chandrakasan, RW Brodersen, ”Minimizing power consumption in digital circuits” *Proc. of the IEEE* Vol 83, No 4, Apr 1995

[15] L Benini, GD Micheli, *”Networks on Chips: A new paradigm for System on Chip Design” DATE* 2002

[16] C Patel, S Chai, S Yalamanchili, D Shimmel, *”Power constrained design of multiprocessor interconnection networks” IEEE International Conference on Computer Design* 1997

[17] AG Wassal, MA Hassan, *”Low-power system-level design of VLSI packet switching fabrics” IEEE Trans on CAD of Integrated Circuits and Systems”* June 2001

[18] J Duato, S Yalamanchili, L Ni, *”Interconnection networks: an engineering approach” IEEE Computer Society Press* 1997

[19] Li-Shiuan Peh, *”Flow Control and Micro-Architectural Mechanisms for Extending the Performance of Interconnection Networks” Ph.D. Thesis, Stanford University* August 2001.

[20] MJ Karol, MG Hluchy, SP Morgan, *”Input versus Output Queuing on a Space-Division Packet Switch” IEEE Trans on Communications* Dec 1987