Power and Performance Comparison of Crossbars and Buses as On-Chip Interconnect Structures

Yan Zhang * Mary Jane Irwin
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
(* Yan Zhang is currently working for NEC Electronics Inc. in Santa Clara, California.)

Abstract

Traditionally, buses have been traditionally used as datapath interconnects because of their simplicity. Yet, as technology quickly scales down and the industry embraces Systems-on-a-Chip (SoC), the increasing global interconnect delay and chip power consumption become big concerns, and alternative datapath interconnect structures should be considered. This paper evaluates two datapath interconnection alternatives – full connection crossbars and multiple-input/output-port buses – at the transistor level and compares their power and delay performances. The results show that although a full connection crossbar consumes more energy per cycle and incurs larger delays than buses, crossbars consume less energy per data transfer when the number of input/output ports is small and the crossbar operates in full parallelism. This makes crossbars a good choice for connecting components and transferring parallel data in SoC designs.

1 Introduction

As interconnect capacitance dominates gate capacitance and wire delays bottleneck the critical path speed, interconnect fundamentally controls the overall operating performance and power consumption of a chip [1] [7]. The increased delay and power consumption of datapath buses has become a problem for deep submicron design.

Power dissipation is minimized by a wide variety of approaches at many levels of design, ranging from circuit level up to software/system level and algorithm level. Energy and power are two related but different concepts. Power is the rate at which energy is consumed. When the processing rate is a fixed constraint, energy and power metrics are interchangeable. While the problem is often stated in terms of power dissipation, the real goal is to minimize energy dissipation in the presence of performance constraints. Thus, the metric to minimize is really the energy-delay product. The following are the basic principles of low power design [8] [9]:

- Reduce switching voltage
- Reduce voltage swing
- Reduce capacitance
- Reduce switching frequency
- Reduce leakage and static currents

Among these, reducing supply voltage leads to the most dramatic power savings due to the quadratic effect, but causes negative effect on performance. Reducing capacitance is an important way to reduce power dynamic power consumption.

Physical design solutions to reducing capacitance include increasing the number of routing layers so that shorter interconnect wires are used and replacing the normal oxide (SiO_2) layers with so-called low-k materials such as polymide or even air, so that the dielectric constant can be reduced.

One promising architectural solution to reduce capacitance for on-chip bus-level interconnect is to replace the traditional bus structure by alternative interconnect structures. Industry has begun to look into these types of alternative interconnect structures.

Reducing resistance is an efficient way to reduce delay. One physical design solution to reduce resistance is the use of copper in place of traditional aluminum because copper has less resistivity than aluminum, and therefore transmits electrical signals faster. IBM scientists have been the first to announce a new advance in semiconductor process that entails replacing aluminum with copper [12]. IBM also claims that using copper could increase the speed of microprocessors by up to 40 percent. Resistance can also be reduced

0-7803-5700-0/99/$10.00©1999 IEEE

by increasing the interconnect thickness H, but this causes problems with interwire capacitance, thus is only a partial solution.

Multilevel interconnects with wider and thicker lines at the upper level are another technology solution for reducing delays. These low resistance lines can be used for global data/address bus communication, power distribution, and clock distribution. More densely packed wires at the lower levels can be used for local interconnect. Having more interconnect layers also tends to reduce the average wire length because: (1) a straight connection between two points becomes feasible when fewer obstacles are present, and (2) the overall area is reduced because interconnect wires can be placed on top of each other as well as on top of logic components. Two metal layers are available in almost all state-of-the-art technologies at present, while three and more layers are more common [7]. The Semiconductor Industry Association (SIA) projects as many as six wiring layers by the year of 2001 [4].

Our previous research [10] shows that a crossbar with one active data transfer path at any given time (blocking crossbar) consumes more average power than buses and has longer delays. However, a full-connection crossbar can perform multiple non-conflicting data transfers in one cycle while a bus has to implement the data transfers in several cycles (see Table 1). In practice, whether a crossbar can operate at its full parallelism depends largely on the specific application. To find out whether the crossbar consumes more power per data transfer and if it is a possible alternative datapath interconnect structure in low power and high performance chip designs, we compared the power and delay performance of full-connection crossbar structures and multiple-input/output-port bus structures. The circuits we compared were custom designed and simulated at the circuit level using a 0.35 micron CMOS technology. For both structures, 4, 8, and 16 input ports are used in the simulation.

Table 1. Data Transfer Comparison for Buses and Crossbars

# Data Transfers (Non-Conflicting)	Bus (# I/O Ports)	Crossbar		
		4	8	16
4	4	1	1	1
8	8	2	1	1
16	16	4	2	1

2 Architecture and Design of Crossbars and Buses

2.1 Crossbar Design

An N × N crossbar allows arbitrary one-to-one connections between N input ports and N output ports. A set of N rows, each with the structure shown in Figure 1 and with the same input ports connected to all rows, forms such a crossbar [3]. The basic construction cell is a 2-to-1 MUX. The MUX is implemented by two transmission gates since this type of design based on transmission gate has good power and delay performance [10].

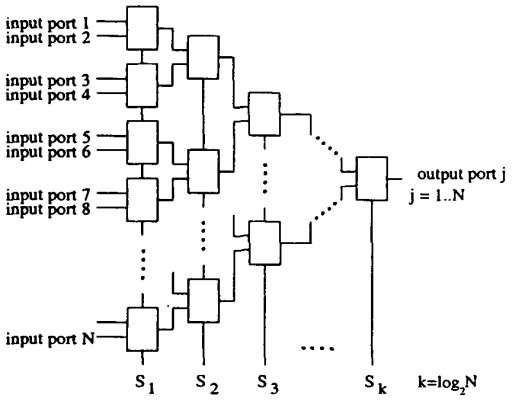

Figure 1. Block Diagram of A Crossbar (One Row)

2.2 Bus Design

There are two basic ways to construct a bus structure: with multiplexers or with tri-state buffers [6]. Since the crossbar is built based on MUXes, we chose to construct the bus structure with multiplexers for the input ports of the bus. This makes the crossbar architecture and bus architecture more comparable. For each output port, there is a tri-state buffer to control whether this destination is receiving data from the bus. Figure 2 shows the architecture of a single-bit, multiple input/output ports bus.

2.3 Discussion

The following kinds of parallelism must be accommodated by data transfers:

- One input source sends data to multiple output destinations.
- Multiple input sources send data to multiple output destinations, and don't block each other.

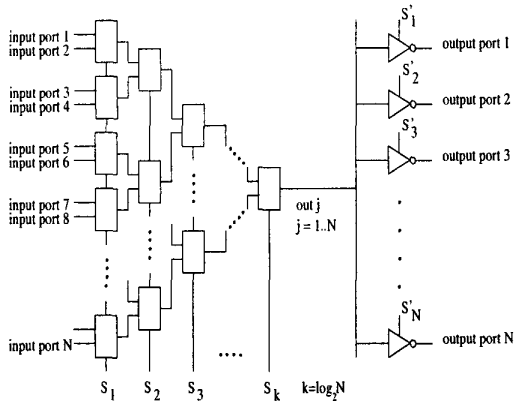

Figure 2. Block Diagram of Multiple Ports Bus

- Multiple input sources send data to one output destination.

Both the buses and crossbars in this paper can handle the first case. Moreover, crossbars can handle the second situation in one cycle while buses cannot. The third situation can be handled by adding a queue to each destination. It is not implemented here since the queue logic for the bus structure and crossbar structure are similar and are additive to both circuits. For simplicity, the implementation in this paper doesn't include a scheduler in either design. A design for a centralized scheduler can be found in [5]. In this paper, the data transfer preprocessing is done manually.

2.4 Implementation

The 0.35 micron CMOS technology is used for both crossbar and bus designs. All circuits are custom designed and simulated using HSPICE [11]. Two buffers have been added to each input drive and output load to account for the driver and load capacitance. Minimum size transistors are used for both designs. This paper will focus on the power and delay of the crossbar and bus circuits themselves. The wiring power consumption and delay are not addressed here since they are additive to the circuit power consumption and delay.

3 Energy and Delay Results

3.1 Data Generation

We used two sets of control data to test both designs. The first set is artificially generated. With this dataset, the crossbar can operate in full parallelism (see Table 1). That is, once all the control signals are set, there are N non-conflicting data transfers handled by a N-way crossbar per cycle. We call this data set *perfect data*. The second data set is pseudo randomly generated, which we call *random data*. Generally, there are less than N data transfers per cycle performed by a N way crossbar. The data input set is generated in such a way that whenever an input port is sending data, its current value always switches to 1 or 0 from a previous 0 or 1. This will give us the longest possible average delay and maximum power consumption.

For each implementation, a sequence of thirty-two data transfers are simulated. The clock cycle time used is $20ns$, which is a typical value. At the beginning of every cycle, a single-bit data is ready to be transferred at each input port.

3.2 Perfect Control Data

Figure 3 through Figure 7 show the results for the perfect data. Figure 3 shows the average energy consumption per cycle for crossbars and buses. As the number of input/output ports increases, the energy consumption increases. Not surprisingly, crossbars consume more energy than buses with the same number of input/output ports. Figure 4 shows the energy per data transfer. It can be seen that crossbars consume less energy than buses per data transfer when averaged over a series of non-conflicting data transfers (N in the case of a N way crossbar). The high throughput of crossbars decreases the energy consumption per data transfer.

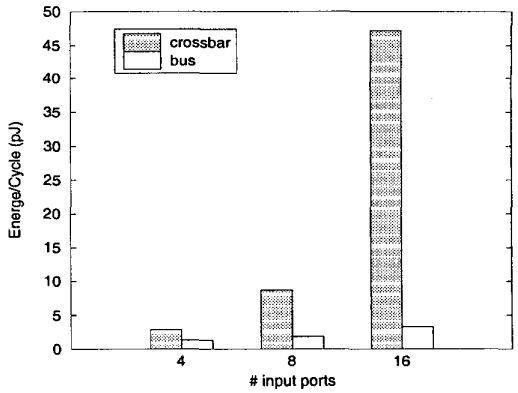

Figure 3. Energy Per Cycle for Perfect Data

Figure 5 presents the worst case delay for crossbars and buses. The crossbar logic incurs more delay than the bus logic.

Figure 6 and 7 show the energy-delay products per cycle and per transfer, respectively. A crossbar has a higher energy-delay product per cycle than a bus. However, it has a lower energy-delay product per transfer than a bus for a small number of input/output ports (four and eight). When there are sixteen input/output ports, the crossbar has a larger

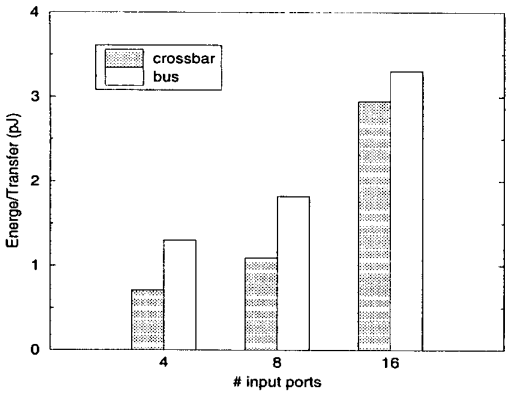

Figure 4. Energy Per Data Transfer for Perfect Data

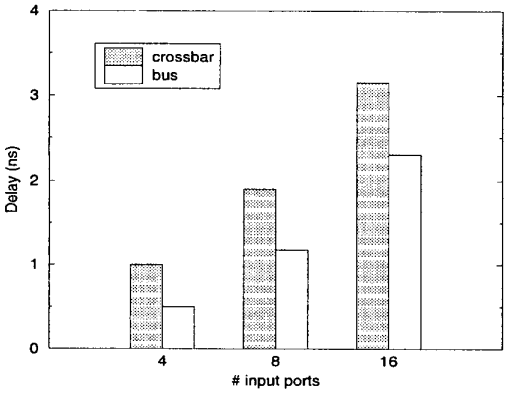

Figure 5. Delays for Perfect Data

energy-delay product per transfer.

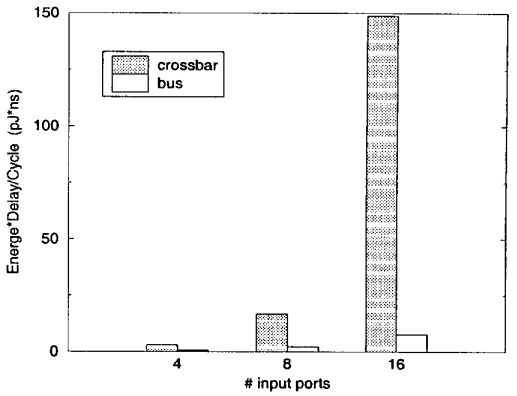

Figure 6. Energy-Delay Products (Per Cycle) for Perfect Data

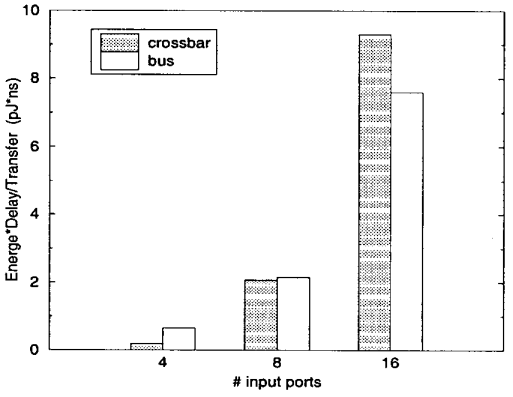

Figure 7. Energy-Delay Products (Per Transfer) for Perfect Data

3.3 Random Control Data

The results for the random control data are shown in Figure 8 through Figure 12. Figure 8 presents the average energy dissipation per cycle. Compared to Figure 3, both designs consume less energy per cycle for the random data than for the perfect data. This is because less than N data transfers are handled per cycle for the random data. Figure 9 shows the energy consumption per transfer. Unlike the corresponding results for perfect data, a crossbar consumes more energy per transfer than a bus.

Figure 10 shows the average delays of crossbars and buses for random data. Crossbars have larger delays than buses in all cases.

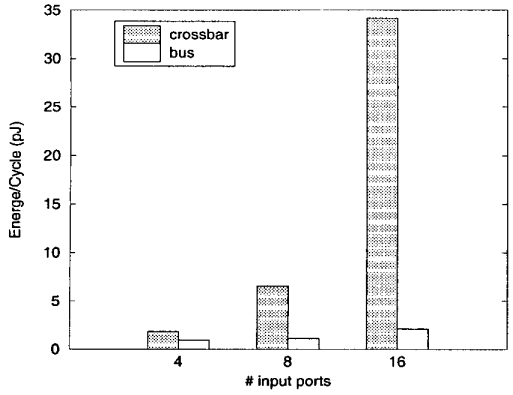

Figure 8. Energy Per Cycle for Random Data

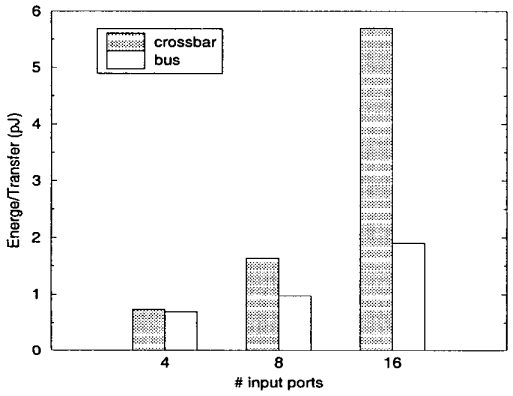

Figure 9. Energy Per Data Transfer for Random Data

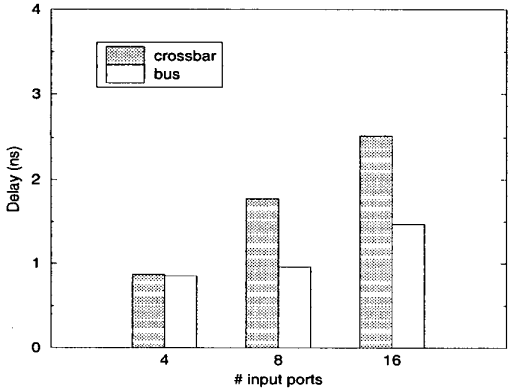

Figure 10. Delay for Random Data

Figure 11 and Figure 12 present the energy-delay product results per cycle and per transfer, respectively. Notice that a crossbar has larger energy-delay per transfer than a bus.

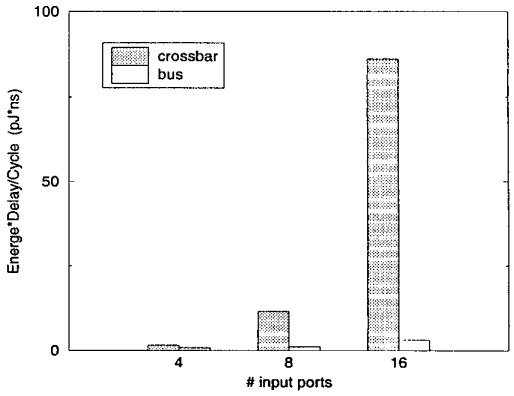

Figure 11. Energy-Delay Products (Per Cycle) for Random Data

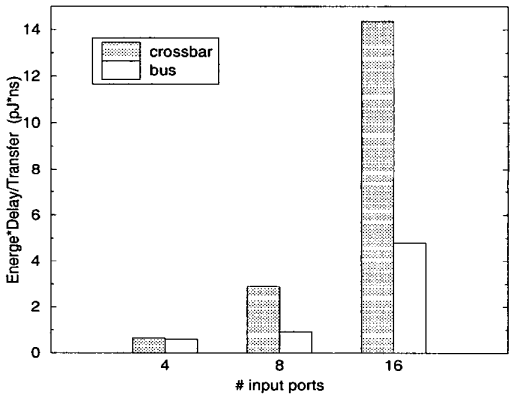

Figure 12. Energy-Delay Products (Per Transfer) for Random Data

In summary, crossbars have an advantage in energy-delay products when the number of input/output ports is small and the crossbars operate in full parallelism. But crossbars pay a large penalty in area as shown in Table 2. However, as effective chip areas have increased due to reduced feature sizes and increased chip dimensions, area is somewhat less of a concern. Indeed, area is now often sacrificed to reduce power [2]. Also, a crossbar is not a good choice for a "hotspot" when data transfer conflicts occur a lot.

Table 2. Area Comparison for Buses and Crossbars

Sizes	Bus ($10^3\lambda^2$)	Crossbar ($10^3\lambda^2$)
4	25.944	46.354
8	62.964	268.8
16	120.744	1562.107

4 Conclusions and Future Work

In this paper, full connection crossbars and multiple-input/output-port buses were simulated at the transistor level using 0.35 micron CMOS technology. The results for perfect data show that a crossbar consumes more energy per cycle, incurs more delay than a bus, and therefore has a larger energy-delay product per cycle. However, since an N × N crossbar can accommodate up to N non-conflicting data transfers per cycle, it has a much larger throughput. Although more energy is consumed in one cycle, a crossbar has a smaller energy consumption per data transfer. As a result, it also has a smaller energy-delay product per transfer than a bus when the number of input/output ports is small (< 8) and it operates in full parallelism. However, when there are sixteen input/output ports or at a "hotspot", the crossbar has larger energy-delay product per transfer than a bus. That can be seen from the results for random data. Therefore, a crossbar is a good choice for connecting components and accommodating parallel data transfers when the total number of components doesn't exceed eight and the crossbar can operate in full parallelism.

Future work includes modeling wire effects at the transistor level and modeling power and performance of crossbars and buses that scale cross technologies with various output loads.

References

[1] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley Publishing Company, Inc. 1990.

[2] A. Chandrakasan, R. Brodersen, *Low power Digital CMOS Design*, Kluwer Academic Publishers, Boston, MA, 1998.

[3] K. Choi, William S. Adams, "VLSI implementation of a 256 × 256 crossbar interconnection network", *Proc. of the Sixth IPPS Conf.*, pp.289, March 1992.

[4] J. F. Freedman, S. Sibbett, "Report of the ad hoc working group on interconnect", April 1997.

[5] P. Gupta, N. McKeown, "Designing and implementing a fast crossbar scheduler", *Micro*, pp. 20-28, January-February 1999.

[6] M. M. Mano, *Computer System Architecture*, Prentice-Hall, Inc. 1993.

[7] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*, Prentice-Hall, Inc. 1996.

[8] J. M. Rabaey, M. Pedram, *Low Power Design Methodologies*, Kluwer Academic Publishers, Boston, MA, 1996.

[9] G. Yeap, *Practical Low Power Digital VLSI Design*, Kluwer Academic Publishers, Boston, MA, 1998.

[10] Y. Zhang, W. Ye, R. M. Owens, M. J. Irwin, "The power analysis of interconnect structures", *Proc. of the 10th IEEE International ASIC Conference*, pp. 25-29, September 1997.

[11] *HSPICE User's Manual*, Meta-Software, Inc. 1996.

[12] IBM, *http://www.research.ibm.com/topics/serious/chip/*, "Copper chip technology".