Communication Modeling for System-Level Design

Andrew B. Kahng[†‡] and Kambiz Samadi[†]
[‡] CSE Department, University of California, San Diego, La Jolla, CA
[†] ECE Department, University of California, San Diego, La Jolla, CA
Email: {abk,ksamadi}@ucsd.edu

Abstract— **Multiprocessor systems-on-chip (MPSoCs) are emerging as a popular SoC design platform. However, major challenges arise from nonscaling global wire delay and from the reuse of intellectual properties (IPs) from different vendors to meet tight time-to-market constraints. Designing the appropriate communication fabrics for such heterogeneous systems becomes a challenging task. In this paper, we present accurate delay, power, and area models for bus-based and packet-switched communication architectures. We also integrate our models into the COSI-OCC system-level communication synthesis tool [19] and show that the more accurate modeling significantly affects optimal/achievable architectures that are synthesized by the system-level tool. Finally, this paper reviews our relevant contributions in [16], [17], [18].**

I. Introduction

Increase in system and design complexity has led to chip multiprocessor (CMP) [8], [4] and multiprocessor system-on-chip (MPSoC) [29], [21] designs. Such systems rely not only on appropriate design and process technologies, but also on the ability to interconnect cores – processors, memory arrays, etc. – reliably and efficiently. The International Technology Roadmap for Semiconductors (ITRS) [30] predicts that future generations of high-end IC designs will operate in the 10-20 GHz range, with multi-Gbit/s communication between cores. A major challenge designers face is to provide a reliable and functional interconnection between the components of the design [1].

As device sizes shrink to keep up with Moore's Law, major challenges arise from non-scalable global wire delays. Recently, designers have moved from a computation-centric view of chip design to a communication-centric view, largely due to the increasing significance of interconnect delay versus gate delay in current and future technologies. Based on the premise that interconnection technology will be a limiting factor for achieving SoC operational goals, we propose a characterization framework for the two most dominant interconnection architectures in today's designs: (1) bus-based architecture (i.e., shared medium) and (2) packet-switched architecture (i.e., as employed in current NoC designs). We review each of these architectures and then propose accurate and fast system-level models for performance, power, and area to aid fast and efficient design space exploration. For bus-based designs, we develop models for AMBA (Advanced Microcontroller Bus Architecture) [29], a popular on-chip bus for ARM processors. For NoC-based designs, we integrate our proposed models into the COSI-OCC communication synthesis tool [19] and show they substantially change the NoC outcome of system-level design exploration. Because state-of-the-art communication architectures are quite complex, with multiple components, there has been little research on modeling and early-design-stage prediction. In this context, we have developed an integrated communication modeling library that:

- models popular low- and high-level communication structures,
- predicts delay, power, and area at early design stage with as much accuracy as feasible,
- is usable by system-level designer, and
- allows technology extrapolation.

In this invited paper, we have integrated content from our previous works [16], [17], [18]. Section II describes the current state-of-the-art bus-based communication architectures and proposes accurate delay, power, and area models for such architectures. Section III describes network-on-chip communication architectures and presents our modeling approach through two examples of physical link and router power modeling. Section IV shows the impact of increased accuracy of our models on system-level design choices and also presents a case study in which we compare common NoC and bus-based architectures. Finally, Section V concludes and gives directions for future work.

II. Bus-Based Architecture

Most current systems-on-chip (SoCs) use a shared-medium architecture to communicate among different IPs. In this architecture, all communicating devices share the same transmission medium and only one device can drive the network at a given time. Inherent support for asymmetric communication allows the flow of information from a few transmitters to many receivers. This convenient, low-overhead interconnection scheme can handle a few bus masters and many bus slaves that only respond to bus masters. However, an arbitration mechanism is required when several devices attempt to use the bus simultaneously, i.e., to ensure that only one bus master at a time is allowed to initiate data transfer. Even though the arbitration scheme is fixed, any arbitration algorithm, such as highest priority or fair access, can be implemented depending on the application requirements. Power consumption is another challenge with these architectures, as every data transfer is broadcast. With the emergence of many-core designs, bus-based architectures can present critical performance and power bottlenecks [1].

A. Bus Protocol Encapsulation

1) Master/Slave Communication: In bus-based architecture, master/slave communication is the basic protocol between two active IPs. In this configuration, masters send orders to slaves, which then send results as answers [10]. Typically, masters are processing elements and slaves are memory banks; hence, the orders are read or write operations to/from a specific address or block of addresses. Results are typically data or blocks of data, along with the order status (no errors, not ready, etc.) Let $core1$ be a processing element performing an operation O on two integers, and let slave S be a memory unit containing the required integers. To perform the operation O, $core1$ first reads the memory to obtain the values of the operands, say x and y. It then performs the operation and finally writes the result, say z, in the memory. We observe, as was introduced in [12], that the communication transactions (i.e., read and write) are orthogonal to the computation operation (i.e., operation O). Hence, they are separated into two different components: (1) interface, and (2) core. To distinguish between interface-core and interface-interface communications we represent the former by (orders, results) and the latter by (request, responses) as shown in Figure 1. The following is an example of this denotation [10]:

- order $\approx$ (operation, location, data)
- result $\approx$ (status or the status of data)

Requests and responses are simply the translation of the orders and results into bit switches.

- request $\approx$ (R/W, addr, data) (e.g., R/W = 1 means a read operation at address "addr" is requested)
- response $\approx$ (status or the status of data)

$@x$ and $@y$ are the memory locations of operands x and y, respectively. The master core performs the following operations to fulfil operation O:

1) x = (read $@x$) then wait for a result
2) y = (read $@y$) then wait for a result
3) write $@x$ the result of operation O

In the next subsection we further break down the (orders, results) and (requests, responses) pairs into specific communication events.

2) Communication Events: All communication transactions between different devices in a bus-based architecture can be broken down into sequences of events, which are essentially orders and requests that go back and forth between the communicating devices as well as their interfaces (Figure 1). The operation O mentioned above, can be viewed as the following sequence of communication events:

- master interface receives the read order and produces the request (1 @x);
- slave interface receives this request and transmits the order (read @x) to the slave core;
  - slave core reads its database and comes back with the value of x stored at the address @x along with the result (OK x); this is transmitted back to the master interface, and eventually to the master core
  - master core is now able to issue the second order
- for the write operation the master interface receives the write order and produces the request (0 @x z);
- the slave interface receives this request and translates it to the order (write @x c); and
- the result (OK) is sent back to master core in the same way as in the read operation.

In our modeling, we consider the AMBA AHB bus and will simply use the (orders, results) and (requests, responses) described in the AMBA specification [32].

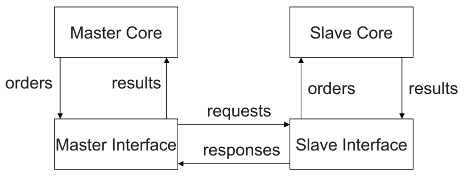


Fig. 1. Communication events [10].

B. Signal Bus Modeling

AMBA contains many signal, address and data buses. The AMBA bus protocol is designed to be used with a central multiplexer interconnection scheme. Using this scheme, all bus masters drive out the address and control signals indicating the transfer they wish to perform, and the arbiter determines which master has its address and control signals routed to all of the slaves. A central decoder is also required to control the read data and responses signal multiplexer, which selects the appropriate signals from the slave that is involved in the transfer. Figure 3 shows the structure required to implement an AMBA design with two masters and two slaves. Buses consume a significant fraction of power and area, and can become performance bottlenecks. Important bus metrics to model are: (1) delay, (2) area, and (3) power. Main parameters that define these metrics are: (1) length, (2) width, and (3) activity factor. For length estimation, we use the bus connectivity known from the AMBA specification and assume that buses are routed as minimum Steiner trees to logic blocks (which their locations are known from floorplan). To estimate the width, we use the width of the connecting logic (i.e., 32-bit, 64-bit, etc.) For activity estimation, we break down the transactions into bit switches as discussed in the previous subsection. Once length, width, and activity factor are known, we use our interconnect modeling library (Section III-A below) to obtain predictions. Figure 2 shows the interface of our AMBA model. The model takes (1) the design floorplan, (2) specific transaction and its characteristics (e.g., read operation with its length and address progression), and (3) technology and design styles which include all the necessary technology and key circuit parameters.

C. Logic Modeling

AMBA contains several logic blocks: (1) bus multiplexers, (2) address decoder, (3) arbiter, and (4) master/slave interfaces. We first

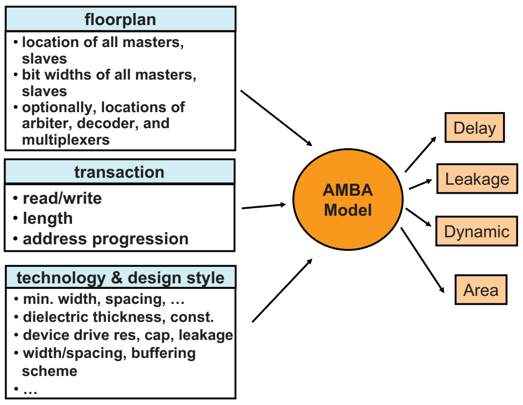

floorplan
• location of all masters, slaves
• bit widths of all masters, slaves
• optionally, locations of arbiter, decoder, and multiplexers
transaction
• read/write
• length
• address progression
technology & design style
• min. width, spacing, ...
• dielectric thickness, const.
• device drive res, cap, leakage
• width/spacing, buffering scheme
• ...
AMBA Model
Delay
Leakage
Dynamic
Area

Fig. 2. Our AMBA model interface.

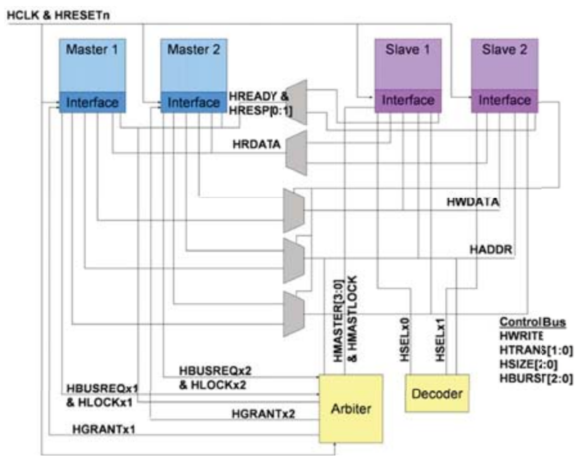


Fig. 3. AMBA signals and logical blocks.

model delay, power and area of these components, and then extend our model to the entire bus architecture. For example, to estimate the dynamic power we take into account the specific protocols for each transaction (i.e., read/write operation). We model a many-to-one multiplexer as a tree of two-to-one multiplexers and use parallel multiplexers to model ones with multiple bitwidth. Area and power estimations for a two-to-one multiplexer are based on standard multiplexer design. Equation (1) represents the dynamic power consumed by a $n:1$ multiplexer, where the numbers of NAND2x1 ($N_{NAND2x1}$), and INVx1 (N_{INVx1}) gates are computed using Equations (2) and (3), respectively.

$$P_{dynamic} = \frac{1}{2}\cdot((N_{NAND2x1}\cdot(4C_{in}+6C_d)) \\ +(N_{INVx1}\cdot(3C_{in}+3C_d)))\cdot V_{dd}^2\cdot f \quad (1)$$

$$N_{NAND2x1} = 2^{(h+1)}-1 \quad (2)$$

$$N_{INVx1} = h = log_2 n \quad (3)$$

where C_{in} and C_d are input gate capacitance and diffusion capacitance of NMOS, respectively.

A decoder decodes the address to select a particular slave. We model a n:2^n decoder as a tree of inverters and two-to-one AND gates. Dynamic power of a n:2^n decoder can similarly be estimated as shown in Equation (4), where the number of AND2x1 (N_{AND2x1}) is computed using Equation (5) and the number of INVx1 is just the number of inputs.

$$P_{dynamic} = \frac{1}{2}\cdot((N_{AND2x1}\cdot(7C_{in}+9C_d)) \\ +(N_{INVx1}\cdot(3C_{in}+3C_d)))\cdot V_{dd}^2\cdot f \quad (4)$$

$$N_{AND2x1} = \sum_{i=1}^{n-1} 2^i \quad (5)$$

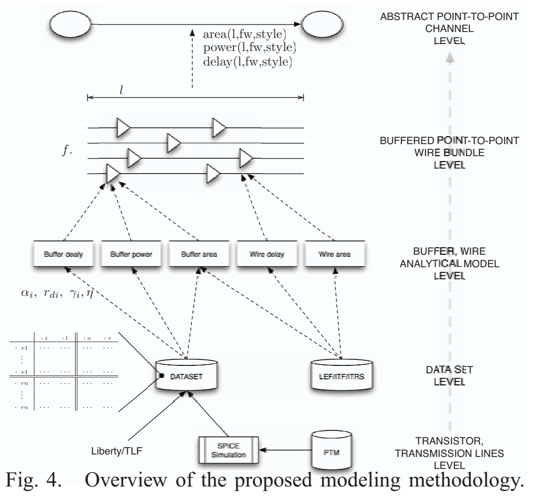
All of the above parameters can be extracted from industry Liberty files [31] or SPICE simulation. To estimate leakage power we obtain the cell leakage power (from a Liberty file) for each of the building block gates (i.e., NAND2x1, AND2x1, and INVx1) and sum up all the values to compute the leakage power of the entire block. Area can be similarly computed from cell areas of the building block logic gates, extracted from a Liberty file. For delay computations, we use our gate delay model [17] described in Section III-A below. Current literature on bus modeling such as [25], [24], [23] are focused at RTL or higher level (i.e., transaction level) whereas we propose physical-level models to estimate delay, power, and area. Our models can be plugged into any high-level synthesis tool or network simulator to aid in estimation of metrics.

III. NoC-Based Architecture

Although bus-based communication architecture has simple topology, low area cost, and extensibility, technology scaling limits the practical number of devices that can be connected and hence limits the architecture's usability. New structured communication fabrics, networks-on-chip (NoCs), have emerged for use in SoC designs [1]. In the NoC approach, communication between different devices takes place in the form of packets. Network routers and wires are the basic building blocks of such communication architectures. In the next two subsections we propose accurate delay, power, and area models for network routers and on-chip interconnects (i.e., point-to-point links). We later integrate our proposed models in the COSI-OCC tool [19] to confirm the need for, and value of, accurate physical-level models in early-stage NoC design space exploration.

A. Physical Link Modeling

Figure 4 shows a representation of our proposed interconnect modeling methodology with its main levels of abstraction [17]. We have developed a set of tools and Application Programming Interfaces (APIs) that allow us to abstract the interconnect cost-performance tradeoffs from detailed SPICE simulations up to a system-level view. Our modeling methodology can be partitioned into the following modeling subtasks.



ABSTRACT POINT-TO-POINT CHANNEL LEVEL
area(l,fw,style)
power(l,fw,style)
delay(l,fw,style)
l
$f.$
BUFFERED POINT-TO-POINT WIRE BUNDLE LEVEL
Buffer dealy
Buffer power
Buffer area
Wire delay
Wire area
BUFFER, WIRE ANALYTICAL MODEL LEVEL
$\alpha_i,\ r_{di},\ \gamma_i, \eta$
DATASET
LEF/ITF/ITRS
DATA SET LEVEL
Liberty/TLF
SPICE Simulation
PTM
TRANSISTOR, TRANSMISSION LINES LEVEL

Fig. 4. Overview of the proposed modeling methodology.

1) Buffered Interconnect Model: Delay, power and area models of repeaters and wires are used to compute the delay, power, and area metrics of a bus with flit size F and length l. The total delay of the buffered interconnect is computed as the sum of the delays of all repeaters and wire segments into which the length l is partitioned. The area and power consumption of a point-to-point interconnect is also computed by summing up the area and power consumption of the constituent repeaters and wire segments, respectively.[1]

[1]All these metrics depend on the way in which the interconnect is implemented, i.e., the design style.

2) System-Level Model of Buffered Interconnect: System-level design tools for NoC employ graph-based optimization algorithms. A node in a graph represents an IP core or a router while an edge represents a point-to-point on-chip interconnect. Edges are labeled with properties such as length (typically approximated by the Manhattan distance between two points on a chip) and flit size (i.e., the number of parallel bits of the point-to-point link). The way in which the interconnect is implemented is usually not a concern of the system-level designer. However, different design styles have different power-performance tradeoffs that may be of interest at the system level. Therefore, we export the `style` parameter that captures the design style of the interconnect.

The APIs available to the system-level design tools are:

- *area(l,F,style)* returns the area occupied by repeaters and wiring, given the length of interconnection, the flit size, and the design style. This is considered a fixed installation cost in the optimization algorithms.
- *power(l,F,style)* returns the power consumed by the interconnection, given the length, the flit size and the design style. This is actually further divided into two APIs for dynamic and leakage power estimations.
- *delay(l,F,style)* that, given a length to span, the flit size and the design style, returns the delay of the interconnect. This information is used by the optimization algorithm such that, given a target clock frequency, checks whether a distance can be spanned by the interconnect.

The system-level design tools are able to explore different solutions corresponding to different design styles, or the design style can be fixed a priori by the user. We have validated our models by physically implementing a buffered interconnect across three technology nodes (90nm, 65nm, and 45nm), two routing layers (global and intermediate) and two design styles (single-width-single-spacing and single-width-double-spacing). Validation details of our models are discussed in [17].

B. Router Power and Area Modeling

We have also developed accurate power and area models for network routers. We model both dynamic and leakage power components. We adopt the router models from [18] and the interconnect models from [17].

1) Router Power Models: Dynamic power consumption in CMOS circuits is given by $P = E \cdot f_{clk}$, where energy $E = \frac{1}{2}\alpha C V_{dd}^2$ and f_{clk} is the clock frequency, α the switching activity, C the switched capacitance, and V_{dd} the supply voltage. We derive detailed parameterized equations to estimate switching capacitance of (1) register-based FIFO buffers, (2) clocking due to routers, and (3) physical links.

Clock. Clock distribution and generation comprise a major portion of power consumption in synchronous designs [15], representing up to 33% of power consumption in a high-performance router [14]. We estimate the term C_{clk} as shown in Equation (6).

$$C_{clk} = C_{sram-fifo} + C_{pipeline-registers} + C_{register-fifo} + C_{wiring} \qquad (6)$$

Throughout our modeling approach it is assumed that all components are built using static CMOS gates. Given that the load of the clock distribution network heavily depends on its topology, we assume an H-tree distribution style where $C_{sram-fifo}$, $C_{pipeline-registers}$, $C_{register-fifo}$, and C_{wiring} are capacitive loads due to memory structures, pipeline registers, FIFO registers, and clock distribution wiring, respectively.

Memory structures. We adopt the original ORION model [20] for SRAM buffers to determine the precharge circuitry capacitive load on the clock network. The pre-charging circuit is just the pre-charging transistor, T_c, which commonly is two PMOS transistors per bitcell. Hence, its capacitance, C_{chg}, is due to its gate and drain end capacitances, $C_g(T_c)$ and $C_d(T_c)$ respectively as shown in Equation (7) (Figure 5). In an SRAM FIFO with B buffers and flit size F, the total capacitance due to pre-charging circuitry can be derived using

Equation (8), with P_r and P_w being the number of read and write ports, respectively.

$$C_{chg} = C_g(T_c) + C_d(T_c) \quad (7)$$

$$C_{sram-fifo} = (P_r + P_w) \cdot F \cdot B \cdot C_{chg} \quad (8)$$

Pipeline registers. Typical interconnection network routers have different pipeline stages. To advance, each flit must proceed through the steps of: (1) routing computation, (2) virtual-channel allocation, (3) switch allocation, and (4) switch traversal.[2] We assume DFF as the building block of the pipeline registers. In a router with flit size of F bits and $N_{pipeline}$ pipeline stages, the capacitive load on the clock due to pipeline registers can be computed as:

$$C_{pipeline-registers} = N_{pipeline} \cdot F \cdot C_{ff} \quad (9)$$

where C_{ff} is the flip-flop capacitance and is extracted from 65nm *HP* (high-performance) and *LP* (low-power) libraries.

Register-based FIFOs. FIFO buffers can be implemented as a series of flip-flops. We assume simple DFF to construct the FIFO. In a B-entry register-based FIFO with flit size of F bits, the capacitive load on the clock can be computed as:

$$C_{register-fifo} = F \cdot B \cdot C_{ff} \quad (10)$$

For the registers we assume D flip-flop (DFF) is used as the building block. We obtain the capacitance value across different drive strengths from TSMC 65*G* and 65*LP* standard cell library data sheets.[3] Architectural parameters change the effective loading of each gate in the design. Hence, to use the appropriate drive strength for the registers we use their load capacitance and timing requirements. In this work, we assume minimum-size DFFs are used in all the registers.

Wiring load. For an H-tree clock distribution with a clock level of 5, then total wire capacitance is given in Equation (11), where C_{int} is the per-unit-length wire capacitance and D represents the chip dimension.

$$C_{wiring} = (\frac{16}{2}D + \frac{1\times 8}{2}D + \frac{2\times 4}{2}D + \frac{4\times 2}{2}D + \frac{8\times 1}{2}D) \cdot C_{int} \quad (11)$$

Register-based FIFO buffers. FIFO buffers consume up to 22% of the total router power in [14]. As mentioned earlier, FIFO buffers can be implemented as either SRAM or shift registers. The ORION 1.0 model supports only the use of SRAM-based FIFOs. We use flip-flops as the building block of the shift registers. Hence, a B-entry FIFO buffer can be implemented as a series of B flip-flops (FF).

Write operation. The write operation occurs at the tail of the shift register. Assuming the new flit is f_n and the old flit is f_o, the number of switched flip-flops is the Hamming distance between them. Therefore, the write energy is:

$$E_{write} = H(f_n, f_o) E^{ff}_{switch} \quad (12)$$

where E^{ff}_{switch} is the energy to switch one bit. To simplify the analysis, let $\overline{H}$ denote the average switching activity; then, the average write energy is:

$$\overline{E}_{write} = \overline{H} \cdot E^{ff}_{switch} \quad (13)$$

Read operation. The read operation has two steps:

1) The flit stored at the header of the buffer is read into the crossbar. Since the header of the buffer is directly connected to the input port of the crossbar, this step does not consume any energy in the buffer.
2) Subsequent flits in the buffer are shifted one position towards the header. If the buffer holds n flits before the read operation, n-1 flip-flop writes are performed to shift the data.

Hence, the average read energy is:

$$\overline{E}_{read} = (n-1) \cdot \overline{E}_{write} \quad (14)$$

We obtain the capacitance value across different drive strengths from TSMC 65*G* and 65*LP* standard cell library data sheets.

Leakage power modeling. As technology scales to deep submicron processes, leakage power becomes increasingly important as compared to dynamic power. There is thus a growing need to characterize and optimize network leakage power as well. Chen *et al.* [7] proposed an architectural methodology for estimating leakage power. However, [7] only considered subthreshold leakage whereas from 65nm and beyond gate leakage gains importance and becomes a significant portion of the leakage power. This is even more visible for high-performance applications where gate oxides are much thinner (i.e., ∼1.5nm in 65nm *HP* library). We follow the same methodology proposed in [7] with addition of gate leakage in our leakage analysis. We also use different V_{th} flavors to better represent leakage power consumption for different applications (i.e., high-performance vs. low-power).

To derive an architectural leakage model, we can separate the technology-independent variables such as transistor width from those that stay invariant for a specific process technology:

$$I_{leak}(i,s) = W(i,s) \cdot (I'_{sub}(i,s) + I'_{gate}(i,s)) \quad (15)$$

where I_{leak} is total leakage current. I'_{sub} and I'_{gate} are subthreshold and gate leakage currents per unit transistor width for a specific technology, respectively. $W(i,s)$ refers to the effective transistor width of component i at state s. We measure I'_{sub} and I'_{gate} for a variety of circuit components, input states, operating conditions (i.e., voltage and temperature), and different V_{th} flavors. The modeling methodology is as follows.

1) Identify the fundamental circuit components, and derive $I'_{sub}(i,s)$ and $I'_{gate}(i,s)$ for each at different input states, operating condition, and V_{th} flavors. Examples are single NMOS and PMOS transistors, 2-input NAND gates, etc.
2) Define major architectural building blocks. For interconnection networks, typical building blocks will be buffers, crossbar, arbiters and links [6].
3) Compose architectural leakage power model in a bottom-up fashion for each building block.

Derivation of I_{leak}. For each circuit component i and input state s, we simulate I'_{sub} and I'_{gate} using HSPICE and 65nm foundry SPICE model. Circuit structures can then be hierarchically composed from the fundamental circuit components.

Input state probabilistic analysis. We analyze the probability distribution of each input state of a circuit component by examining how architectural units function. Given the I'_{sub} and I'_{gate}, and the probabilities of each input state $Prob(i,s)$, the leakage current for a building block is:

$$I_{leak}(Block) = \sum_i \sum_s Prob(i,s) \cdot W(i,s) \cdot (I'_{sub}(i,s) + I'_{gate}(i,s)) \quad (16)$$

$$I_{leak}(Block,t) = \sum_i W(i,s(t)) \cdot (I'_{sub}(i,s(t)) + I'_{gate}(i,s(t))) \quad (17)$$

where $I_{leak}(Block,t)$ is the leakage current at time t, and $s(t)$ is the state of circuit type i at time t within this circuit block.

2) Router Area Model: We use a recent model by [9] and the analysis in [11] to estimate the areas of transistors and gates such as inverters, NAND, and NOR gates. This is a fast technique to estimate standard cell characteristics before the cells are laid out. Using this model and for every router building block (i.e., FIFO buffer, crossbar switch and arbiter) our area modeling methodology is as follows.

- Decompose the block into its fundamental circuit components (i.e., gate-level netlist)
- Compose the area model in a bottom-up fashion for every block
- Sum the area of individual blocks as well as global whitespace

FIFO buffers. Designers typically implement buffers as SRAM arrays. Some on-chip networks, such as the Raw microprocessor [4], use shift registers due to less demanding buffer space. We model

[2]The number of pipeline stages can be different for various applications and is a function of clock frequency. We assume this is input by the user.

[3]TSMC represents the high-performance domain with *G* library.

TABLE I

COMPARISON OF DYNAMIC POWER (P_{dyn}), STATIC POWER (P_{leak}), DEVICE AREA (A_d) AND TOTAL AREA (A_{tot}) METRICS RELATIVE TO WIRES, AND NUMBER OF HOPS, BETWEEN THE ORIGINAL AND PROPOSED MODELS.

SoC		P_{dyn} (mW)		P_{leak} (mW)		A_d (mm^2)		A_{tot} (mm^2)		ave. # hops		max. # hops	
		Orig.	Prop.	Orig.	Prop.	Orig.	Prop.	Orig.	Prop.	Orig.	Prop.	Orig.	Prop.
VPROC	65nm	51.1	179.9	69.9	86.7	0.036	0.007	0.217	0.223	3.10	3.42	4	6
dVOPD	65nm	27.3	73.2	25.7	33.2	0.013	0.003	0.082	0.085	1.76	1.91	3	4

TABLE II

COMPARISON OF NETWORK POWER (P), AREA (A), TOTAL NUMBER OF ROUTERS, AND HOP COUNT, USING ORION 1.0 AND ORION 2.0 MODELS.

SoC		P (mW)		A (mm^2)		# routers		max. # router ports		max. # hops	
		v1.0	v2.0	v1.0	v2.0	v1.0	v2.0	v1.0	v2.0	v1.0	v2.0
VPROC	65nm	0.857	0.924	2.043	2.329	33	25	8	12	6	5
dVOPD	65nm	0.412	0.486	1.217	1.343	18	16	6	6	11	10

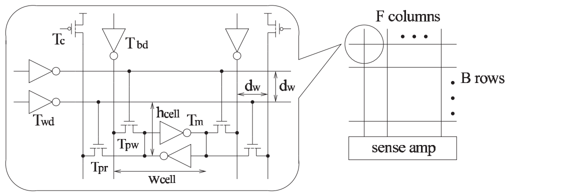


Fig. 5. SRAM-based FIFO buffer with one read and one write port. T_c is the pre-charging transistor, T_{wd} the wordline driver, T_{bd} the write bitline driver, T_m the memory cell inverter, and T_{pr} and T_{pw} the pass transistors connecting read and write ports to memory cells respectively [6].

the area of both implementations, but explain only the SRAM-based model here. Figure 5 shows the structure of a SRAM-based FIFO buffer. Equations (18) and (19) compute the wordline and bitline lengths of the FIFO, respectively.

$$L_{word-line} = F \cdot (w_{cell} + 2(P_r + P_w)d_w) \quad (18)$$

$$L_{bit-line} = B \cdot (h_{cell} + (P_r + P_w)d_w) \quad (19)$$

where F, B, w_{cell}, h_{cell}, d_w, P_r, and P_w are flit size in bits, buffer size in flits, memory cell width, memory cell height, wire spacing, number of read ports and number of write ports, respectively. Hence, the total area for a B entry buffer with flit size of F is calculated as follows. In this model, h_{cell} and w_{cell} are computed using the gate area model described earlier.

Crossbar switches. We consider two common crossbar implementations – multiplexer-tree and matrix. Here, we explain just the matrix crossbar model. The area of a matrix crossbar with I input ports, O output ports and flit size of F can be estimated as follows.

$$Area_{crossbar} = (O \cdot F \cdot w_t) \times (I \cdot F \cdot h_t) \quad (20)$$

where w_t and h_t are track width and height, respectively.

Arbiters. We model three types of arbiters: matrix, round-robin, and queuing. Here, we explain just the matrix arbiter model. For a matrix arbiter with R requesters, its priorities can be represented by an $R \times R$ matrix, with a 1 at the intersection of row i and column j if requester i has higher priority than requester j, and a 0 otherwise. Let req_i be the i^{th} request, gnt_n the n^{th} grant, and m_{ij} the i^{th} row and j^{th} column element in the matrix. Using these variables [6],

$$gnt_n = req_n \times \prod_{i<n}(\overline{req_i} + \overline{m_{in}}) \times \prod_{i>n}(\overline{req_i} + \overline{m_{ni}}) \quad (21)$$

In this arbiter there are $2(R-1)R$ 2-input NOR gates, R inverters and $\frac{R(R-1)}{2}$ registers. We assume that D flip-flops (DFF) are used for registers. Hence, the area of a matrix arbiter with R requests is

$$Area_{arbiter} = (Area_{NOR2X1} \cdot 2(R-1)R) + (Area_{INVX1} \cdot R) + (Area_{DFF} \cdot \frac{R(R-1)}{2}) \quad (22)$$

3) System-Level Router Power and Area Models: The way in which the router is implemented is usually not a concern of the system-level designer. However, different architectural parameters and circuit implementations have different power tradeoffs that may be of interest at the system level. Therefore, we export the `architectural` and `circuit` parameters that capture all the architectural parameters and circuit implementations, respectively.

For each of the router building blocks, the APIs available to the system-level design tools are:

- *area(architectural,circuit)* returns the area occupied by repeaters and wiring, given the length of interconnection, the flit size, and the design style. This is considered a fixed installation cost in the optimization algorithms.
- *power(architectural,circuit)* returns the power consumed by the interconnection, given the length, the flit size, and the design style. This is actually further divided into two APIs for dynamic and leakage power estimations.

IV. VALIDATIONS

A. Experimental Results and Significance Assessment

To assess the impact of improved accuracy on system-level design-space exploration, we integrate our models in the COSI-OCC tool [19]. We use two representative SoC designs as test cases. The first design (VPROC) is a video processor with 42 cores and 128-bit datawidths. The second design is based on a dual video object plane decoder (dVOPD), where two video streams are decoded in parallel by utilizing 26 cores and 128-bit datawidths. The clock frequency used is 2.25 GHz for 65nm technology node. The main differences between NoC obtained using original models with the one using the improved models are in the area and hop count. The original models are relatively optimistic in that they allow excessively long wires. Table I compares the interconnect power, delay, and area when the original [13] and proposed models [17] are used. The original model uses the Bakoglu delay model [26] and does not consider any of the improvements (i.e., hybrid buffering scheme, design styles, etc.) It also takes its technology inputs from PTM models, which are relatively inaccurate compared with industry technology files. We have shown that inaccurate models lead to design solutions that are actually not implementable.

Table II compares network power, area, total number of routers, and hop count when previous models [20] and our proposed models are used. The clock frequency used is 2.25 GHz for 65nm technology node. We can observe that with our models, fewer routers with more ports are used. Since previous models were missing a number of important power components (i.e., clock power, link power, etc.) they tend to underestimate the power. Also, we observe that relative power due to an additional port (i.e., buffers and crossbar port) is not as high in our models as opposed to previous models. Finally, more accurate models lead to a better-performing NoC that satisfies requirements at a lower hop count.

B. Bus vs. NoC Case Study

We apply our proposed physical models to predict power and area of three different communication architectures. We use star and mesh architectures for our NoC-based communication designs. Figure 6(c) shows our testcase in which we have five communicating cores connected in a bus-based fashion. Figures 6(a) and (b) show star and mesh architectures, respectively.

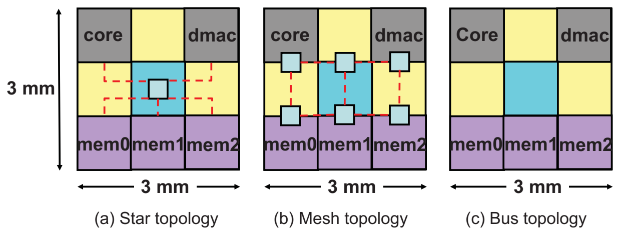


Fig. 6. Communication architectures: (a) Star topology, (b) Mesh topology, and (c) Bus topology.

In our experiments, data width and address width are 32 bits each for the bus-based architecture. For the NoC-based architectures the flit size, F, is 32 bits and we use 5×5 routers with fifo depth of 2. We use TSMC 65*GP* library and a clock frequency of 2.25*GHz*. Figure 7(a) shows the area occupied by each of the three configurations. We observe that the bus-based architecture takes up more area largely due to its substantial wiring for different buses. The mesh topology requires more area than the star topology due to the larger number of routers needed. However, we notice dynamic power consumption of the bus-based architecture is less than that of either NoC-based architecture. Since bus-based architecture uses a relatively larger amount of wiring, and since every masters broadcasts its communications to all the slaves, we anticipate that dynamic power consumption in bus architecture should be larger than in NoC architectures. We attribute the observed outcome to two causes: (1) in our NoC simulation all the routers are on, and (2) mismatch in real switching activity. The former arises because we do not have flow control; this effectively assumes that all routers are active and consuming power. The latter arises because the ORION model uses one external activity factor which does not correspond to an individual transaction (i.e., write transaction in our case). Figures 7(b) and (c) show the dynamic and leakage power for a write transaction between "dmac" and "mem0" cores. We observe that in NoC-based architectures, routers are the main contributors to the dynamic power.

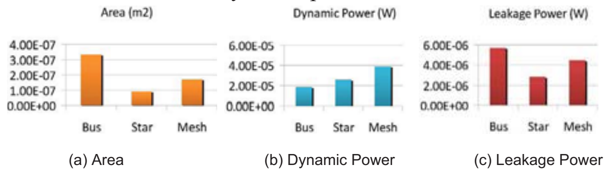


Fig. 7. Area, dynamic power, and leakage power of our three communication architectures. Dynamic and leakage power values are for a write transaction between "dmac" and "mem0" cores.

V. Conclusions and Future Directions

Accurate estimation of delay, power, and area of interconnection fabrics (i.e., point-to-point links, routers, and logical blocks) early in the design process can be enabling to effective system-level exploration. In this work, we have proposed accurate delay, power and area models used in bus-based and NoC-based communication architectures. Our results suggest that – depending on design objectives and constraints – both the bus-based and NoC-based architectures can be effective. Our models can be used in any system-level design tool that seeks a hybrid solution to next generation heterogenous interconnection fabrics.

VI. Acknowledgments

We would like to thank Prof. Luca Carloni of Columbia University, Dr. Swamy Muddu of NextWave Wireless, Dr. Alessandro Pinto of UTRC and Dr. Puneet Sharma of Freescale for their collaboration on interconnect modeling [16], [17]. We would also like to thank Ms. Bin Li and Prof. Li-Shiuan Peh of Princeton University for their collaboration on router power modeling [18]. This research is supported by the MARCO Gigascale Systems Research Center.

References

[1] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Computer*, 35(1), 2002, pp. 70-78.
[2] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", *Proc. DAC*, 2001, pp. 684-689.
[3] D. Pamunuwa, L.-R. Zheng and H. Tenhunen, "Maximizing Throughput Over Parallel Wire Structures in the Deep Submicrometer Regime," *IEEE Transactions on VLSI*, 11, 2003, pp. 224-243.
[4] M. B. Taylor *et al.*, "Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams," *Proc. ISCA*, 2004, pp. 2-13.
[5] S. Heo and K. Asanovic, "Replacing Global Wires With an On-Chip Network: A Power Analysis," *Proc. ISLPED*, 2005, pp. 369-374.
[6] H. Wang, L.-S. Peh and S. Malik, "A Power Model for Routers: Modeling Alpha 21364 and InfiniBand Routers," *IEEE Micro*, 2003, pp. 26-35.
[7] X. Chen and L.-S. Peh, "Leakage Power Modeling and Optimization in Interconnect Networks," *Proc. ISLPED*, 2003, pp. 90-95.
[8] P. Kongetira *et al.*, "Niagara: A 32-Way Multithreaded SPARC Processor," *IEEE Micro*, 25(2), 2005, pp.21-29.
[9] H. Yoshida, D. Kaushik and V. Boppana, "Accurate Pre-Layout Estimation of Standard Cell Characteristics," *Proc. DAC*, 2004, pp. 208-211.
[10] J. Schmaltz and D. Borrione, "Validation of a Parameterized Bus Architecture Model," *Proc. TIMA-VDS*, 2003.
[11] S. Thoziyoor, N. Muralimanohar, J. H. Ahn and N. P. Jouppi, "CACTI 5.1," *Technical Report HPL-2008-20*, HP Laboratories, 2008.
[12] J. A. Rowson and A. Sangiovanni-Vincintelli, "Interface-Based Design," *Proc. DAC*, 1997, pp. 178-183 .
[13] A. Pinto, L. P. Carloni and A. L. Sangiovanni-Vincentelli, "A Methodology and an Open Software Infrastructure for Constraint-Driven Synthesis of On-Chip Communications," *Technical Report* UCB/EECS-2007-130, Nov. 2007.
[14] Y. Hoskote, S. Vangal, A. Singh, N. Borkar and S. Borkar, "A 5-GHz Mesh Interconnect for a Teraflops Processor," *IEEE Micro*, 2007, pp. 51-61.
[15] D. E. Duarte, N. Vijaykrishnan and M. J. Irwin, "A Clock Power Model to Evaluate Impact of Architectural and Technology Optimization," *IEEE Transactions on TVLSI* 10(6), 2002, pp. 844-855.
[16] L. P. Carloni, A. B. Kahng, S. Muddu, A. Pinto, K. Samadi and P. Sharma, "Interconnect Modeling for Improved System-Level Design Optimization," in *Proc. ASPDAC*, 2008, pp. 258-264.
[17] L. P. Carloni, A. B. Kahng, S. Muddu, A. Pinto, K. Samadi and P. Sharma, "Interconnect Modeling for Improved System-Level Design Optimization," *draft*, 2008.
[18] A. B. Kahng, B. Li, L.-S. Peh and K. Samadi, "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," *Technical Report*, UCSD/CS-2008-0929, September 2008.
[19] A. Pinto, L. P. Carloni and A. L. Sangiovanni-Vincentelli, "COSI: A Public-Domain Design Framework for the Design of Interconnection Networks," *Technical Report* UCB/EECS-2008-22, Mar. 2008.
[20] H. Wang, X. Zhu, L.-S. Peh and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks," *Proc. MICRO 35*, 2002, pp. 294-395.
[21] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, *et al.*, "The Design and Implementation of a First-Generation Cell Processor," *Proc. ISSCC*, 2005, pp. 184-185.
[22] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, Reading, MA, 1990.
[23] M. Caldari, M. Conti, M. Coppola, P. Crippa, S. Orcioni, *et al.*, "System-Level Power Analysis Methodology Applied to the AMBA AHB Bus," *Proc. DATE*, 2003, pp. 32-37.
[24] G. Schirner and R. Domer, "Quantitative Analysis of Transaction Level Models for the AMBA Bus," *Proc. DATE*, 2003, pp. 230-235.
[25] M. Caldari, M. Conti, M. Coppola, S. Curaba, L. Pieralisi, *et al.*, "Transaction-Level Models for AMBA Bus Architecture Using SystemC 2.0," *Proc. DATE*, 2003, pp. 26-31.
[26] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, 1990.
[27] J. Cong and D. Z. Pan, "Interconnect Delay Estimation Models for Synthesis and Design Planning," *Proc. IEEE ASPDAC*, 1999, pp. 507-510.
[28] S. N. Adya and I. L. Markov, "Fixed-Outline Floorplanning: Enabling Hierarchical Design," *IEEE Transactions on VLSI*, 11(2), 2003, pp. 1120-1135.
[29] *ARM* Integrated Multiprocessor Core, 2006 http://www.arm.com/ .
[30] *International Technology Roadmap for Semiconductors*, http://www.itrs.net
[31] *Liberty File Format*, http://www.synopsys.com/products/libertyccs/libertyccs.html
[32] *AMBA Specification (Rev 2.0)*, http://www.arm.com
[33] *LEF/DEF Exchange Format*, http://openeda.si2.org/projects/lefdef