Design Theory and Implementation for Low-Power Segmented Bus Systems

W.-B. JONE
University of Cincinnati
J. S. WANG
National Chung-Cheng University
HSUEH-I LU
Academia Sinica
I. P. HSU
Faraday Technology Corp.
and
J.-Y. CHEN
Winbond Electronics Corp.

The concept of bus segmentation has been proposed to minimize power consumption by reducing the switched capacitance on each bus [Chen et al. 1999]. This paper details the design theory and implementation issues of segmented bus systems. Based on a graph model and the Gomory-Hu cut-equivalent tree algorithm, a bus can be partitioned into several bus segments separated by pass transistors. Highly communicating devices are placed to adjacent bus segments, so most data communication can be achieved by switching a small portion of the bus segments. Thus, a significant amount of power consumption can be saved. It can be proved that the proposed bus partitioning method achieves an optimal solution. The concept of tree clustering is also proposed to merge bus segments for further power reduction. The design flow, which includes bus tree construction in the register-transfer level and bus segmentation cell placement and routing in the physical level, is discussed for design implementation. The technology has been applied to a - controller design, and simulation results by PowerMill show significant improvement in power consumption.

Categories and Subject Descriptors: B.7.1 [**Integrated Circuits**]: Types and Design Styles—*VLSI (very large scale integration)*; B.7.2 [**Integrated Circuits**]: Design Aides—*Placement and routing*;

Authors' addresses: W.-B. Jone, Department of Electrical and Computer Engineering and Computer Science, University of Cincinnati, Rhodes 836B, P.O. Box 210030, Cincinnati, OH 45221-0300; email: wjone@ececs.uc.edu; J. S. Wang, Department of Electrical Engineering, National Chung-Cheng University, 16 San-Hsing, Ming-Hsiung, Chia-Yi, 621 Taiwan, R.O.C.; email: ieegsw@ccu.edu.tw; H.-I. Lu, Institute of Information Science, Academia Sinica. 128 Academia Rd., Section 2, Taipei 115, Taiwan, R.O.C.; email: hil@iis.sinica.edu.tw; I. P. Hsu, Faraday Technology Corp., 10-2, Li-Hsin First Road, Science-Based Industrial Park, Hsinchu, Taiwan 300, R.O.C.; email: iphsu@faraday.com.tw.; J.-Y. Chen, Winbond Electronics Corp., Computer Product Design Dept. III Section Manager, No. 4, Creation Rd. III, Science-Based Industrial Park, Hsinchu, Taiwan, R.O.C.; email; cychen16@winbond.com.tw.

G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*; J.2 [**Computer-Aided Engineering**]: Computer-Aided Design (CAD)

General Terms: Algorithms, Design, Theory

Additional Key Words and Phrases: ASIC design, low-power design, bus segmentation, bus graph model, OLA tree, bus segmentation cell, low-power design flow

1. INTRODUCTION

As the VLSI technology is moving to even higher densities of integration, power consumption has been one of the most important limitations in the design of VLSI circuits. Thus, VLSI design for power optimization to satisfy the power budget is an important research issue [Chandrakasan et al. 1992; Chandrakasan and Brodersen 1995]. Generally, a significant portion of power in a chip is consumed by its bus system. For example, the bus wires dissipate about 15% to 30% of the total power in DEC Alpha 21064 and Intel 80386 [Liu and Svenoson 1994]. The work by Mehra et al. [1997] also demonstrates that at least 20% to 35% of total power is dissipated by the bus wires in the case of QMF filters. Researchers have been trying to optimize the bus power from several different design aspects. Previous work can basically be divided into four categories: (1) design of bus drivers/receivers to decrease the bus swing [Najagome et al. 1993; Cardarilli et al. 1996; Golshan and Haroun 1994; Bellaouar et al. 1995; Ikeda and K. Asada 1994; Hiraki et al. 1995; Caufape and Figueras 1996], (2) encoding techniques to reduce the bus switching activity [Stan and Burleson 1995a, 1995b], (3) bus structure redesign to take advantage of local communications [Mehra et al. 1997], and (4) implementation of low-power bus using technologies, such as emitter-coupled logic (ECL) and bipolar [Sundstorm et al. 1990]. Here, we focus on complementary metal-oxide semiconductor (CMOS) circuits only.

The concept of bus segmentation was proposed in Chen et al. [1999] to change the bus topology for the purpose of reducing the power consumption. The basic idea is to partition a single (and large) bus into several tree-structured bus segments that are separated by pass transistors as shown in Figure 1. The segmented bus system is designed in such a way that heavily communicating devices are connected to the adjacent bus segments. Thus, by turning off some pass transistors, only part of the bus segments are involved in the data communication when two devices exchange data. It is assumed that the bus operation is *dynamic* and separated into two phases: the *precharge phase* and the *evaluation phase*. In the precharge phase, the precharge p-channel metal-oxide semiconductor (PMOS) is turned on and the entire bus is charged to high voltage. During the bus evaluation phase, the devices on the bus will communicate with one another by enabling the control signal (of the sender) and the input latch (of the receiver). By this design, instead of discharging the entire bus, only a small number of the bus segments are involved in each device communication and power consumption can be greatly reduced. Since each data communication is performed through bus segments which are adjacent, the bus communication speed can be enhanced as well.

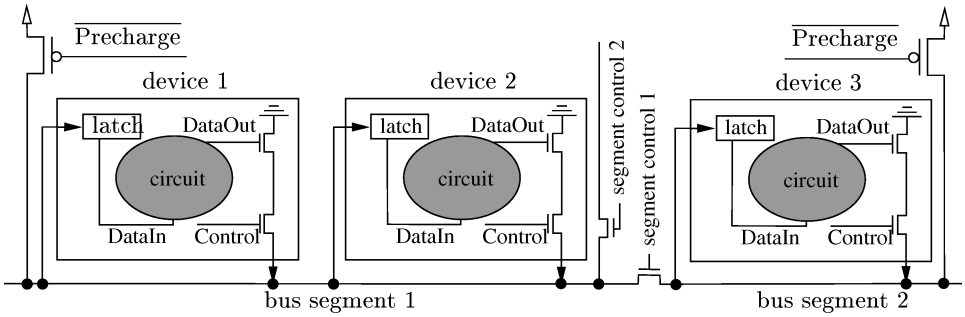


Fig. 1. Bus segmentation.

This paper elaborates the design theory and implementation issues of segmented bus systems. A bus graph model will be used to represent the bus communication system under consideration. Each edge of the bus graph is weighted by a communication frequency if both devices connected by this edge have data exchange. Given the weighted bus graph, we apply the efficient Gomory-Hu method [Gomory and Hu 1961] to partition the single bus into a bus tree where each tree segment is connected by a single device. It can be proved that the bus partitioning method gives an optimal solution in polynomial time. A tree-clustering method is then used to connect several devices into a bus segment if such segment sharing can further reduce the power consumption.

To efficiently implement the proposed bus segmentation technique, we incorporate the bus tree construction process into the register-transfer level (RTL) of our design flow. The resultant RTL design is then synthesized to obtain the gate-level circuits. One major enhancement in RTL design is to partition the bus into segments and add the required control signal generation circuits to the control unit. The other major enhancement is in the physical level design where the bus segmentation cells are designed and integrated into the whole chip placement and routing process. The bus segmentation technology has also been implemented to an 8051-compatible 8-bit -controller. Simulation using PowerMill shows significant improvement in power consumption.

Section 2 gives background on the bus graph model and describes the bus graph partitioning problem. We prove that the bus segmentation problem can be solved optimally using the well-known Gomory-Hu algorithm. Section 3 discusses the design and implementation issues including design flow enhancement, design environment integration, and placement and routing guidelines. Experimental results by designing and simulating a -controller chip are reported in Section 4. Finally, concluding remarks are given in Section 5.

2. BUS GRAPH MODEL AND OPTIMAL LINEAR ARRANGEMENT TREE

The basic idea of bus segmentation is to divide the entire bus into several small bus segments, as shown in Figure 1, such that data exchange among devices will result in minimum power consumption. The power model of a circuit is generally expressed as $P = \sum C_i \cdot V_{dd}^2 \cdot f_i$, where C_i is the load capacitance of circuit i, V_{dd} the operation voltage, and f_i the switching frequency. With the

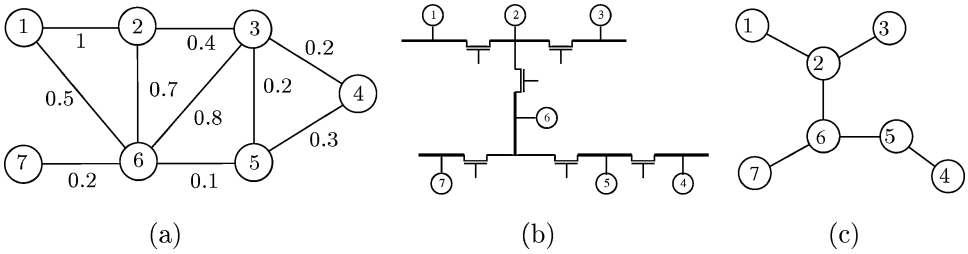


Fig. 2. (a) An edge-weighted graph. (b) A bus tree implementation. (c) The corresponding bus tree.

assumption of a fixed operation voltage, the term V_{dd} can be dropped in the following discussion. Thus, the power model is simplified as $\sum (C_i \cdot f_i)$, where the switching frequency f_i can be minimized at the algorithm, architecture, and logic design levels. Finally, the load capacitance C_i can be greatly reduced at the circuit and physical design levels. Therefore, minimizing the switched capacitance $C_i \cdot f_i$ for each device i at the circuit and physical design levels can be used as the last resort in low-power design. This work concentrates on bus segmentation which leads to switched capacitance reduction.

Based on several assumptions [Chen et al. 1999], the bus segmentation problem can be modeled to partition the bus graph into a bus tree such that the total switched capacitance, and thus the power consumption, by bus switching is minimized. As shown in Figure 2(a), each node in the bus graph represents a device, while each weighted edge represents the communication frequency between a pair of devices. A possible bus tree implementation for the bus graph of Figure 2(a) is shown in Figure 2(b). Figure 2(c) is the corresponding bus tree. The switched capacitance between devices i and j is the product of weight$_G(i, j)$ (i.e., the relative communication frequency) and the capacitance $C(i, j)$ required to be charged or discharged, when devices i and j perform signal exchange. The capacitance cost of a signal communication between devices i and j can be estimated by the formula $k_1(n-1)+k_2$, where k_1 and k_2 are some capacitance values and n is the number of bus segments charged or discharged for communicating devices i and j. Details of the derivation for the capacitance cost formula can be found in Chen et al. [1999].

By the above discussion, in order to achieve low-power design for a bus organization, it suffices to minimize the number $(n-1)$ of traversed bus segments for each pair of data communication. Some definitions are required. The *distance* of two nodes i, j, denoted dist(T, i, j), on a tree T is the number of edges traversed from i to j on T. Let G be a weighted graph. Let T be a spanning tree of G. The *linear arrangement cost* of T corresponding to G, denoted cost$_{\mathrm{LA}}(T, G)$, is $\sum_{e=(i,j)\in E}$ weight$_G(e) \cdot$ dist(T, i, j). The bus segmentation problem is as follows:

> Given an edge-weighted graph $G = (V, E)$, the bus segmentation problem is to identify a spanning tree T whose linear arrangement cost, cost$_{\mathrm{LA}}(T, G)$, is minimum.

The derived spanning tree T is called an *optimal linear arrangement* (OLA) tree. Given an edge-weighted graph G, here, the algorithm by Gomory and

Input: An undirected graph $G = (V, E)$, and a function $\text{weight}_G : E \rightarrow R^+$
Output: A tree $G' = (V, E')$
begin
  Create a super node which is composed of all nodes in V.
  while any super node contains more than one node **do**
    Select a super node that contains at least two nodes.
    Arbitrarily take two nodes u, v in the super node.
    Get a min u-v cut, $\text{cut}_G(A)$.
    Spilt the super node based on the min-cut into two super nodes,
    and have an edge with weight $\text{weight}_G(\text{cut}_G(A))$.
  return the resulting tree of super nodes.
end

Fig. 3. The algorithm for computing a Gomory-Hu cut tree.

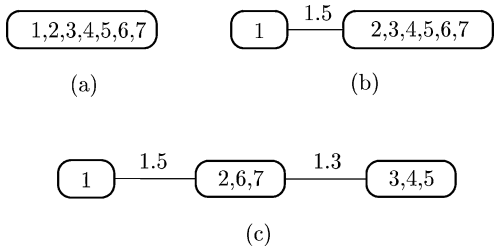


Fig. 4. (a) The initial super node. (b) The tree of super nodes after obtaining the min 1-2 cut. (c) The tree of super nodes after obtaining the min 2-3 cut.

Hu [1961] will be applied to find a tree on G with minimum linear arrangement cost in polynomial time. Some definitions are required for elaborating our tree-construction algorithm. Let $G = (V, E)$ be a connected edge-weighted graph. A subset E' of E is a *cut* of G if $G - E'$ is not connected. A cut E' of G is a *u-v cut* of G if u and v are not connected in $G - E'$. For every subset A of V, let $\text{cut}_G(A)$ consist of the edges of G with exactly one endpoint in A. Define the *weight* of $\text{cut}_G(A)$ to be $\text{weight}_G(\text{cut}_G(A)) = \sum_{e \in \text{cut}_G(A)} \text{weight}_G(e)$. A u-v cut is a *min u-v cut* if its weight is minimum over all u-v cuts of G. For example, in Figure 2(a), if A contains nodes 1, 2, 6, and 7 then $\text{cut}_G(A)$ is a cut containing edges (2, 3), (3, 6), and (5, 6). The weight of the cut, $\text{weight}_G(\text{cut}_G(A))$, equals 1.3. Furthermore, $\text{cut}_G(A)$ is a 6-3 cut and is also a min 6-3 cut. However, $\text{cut}_G(A)$ is a 1-4 cut but not a min 1-4 cut.

The Gomory-Hu (GH) cut-equivalent tree algorithm was proposed to solve the multiterminal network flow problem [Gomory and Hu 1961]. Gomory and Hu showed that a minimum cut between nodes i and j gives the maximal flow between them. The same concept can be applied to solve our OLA-tree problem in polynomial time. The basic idea of the Gomory-Hu cut-equivalent tree algorithm is to arbitrarily select two nodes, then find a minimum cut that disconnects these two nodes. The min-cut is then represented by an edge. This process is repeated until the graph becomes a tree. The algorithm is shown in Figure 3.

The algorithm is illustrated using the graph in Figure 2(a). In the beginning of the algorithm, all nodes are clustered in a super node, as shown in Figure 4(a).

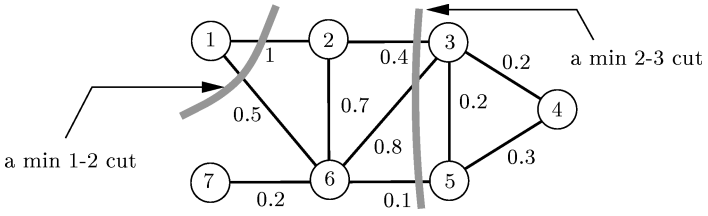


Fig. 5. A min 1-2 cut and a min 2-3 cut.

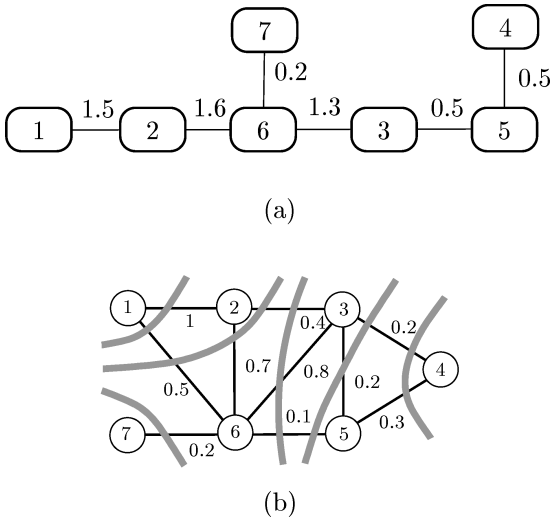


Fig. 6. (a) The final topology of the super nodes. (b) All min cuts.

We arbitrarily select two nodes, say 1 and 2 in Figure 5, to be disconnected. Since a min 1-2 cut consists of edges $(1, 2)$ and $(1, 6)$, the super node can be divided into two super nodes (1) and (2, 3, 4, 5, 6, 7) with weight equal to 1.5 (Figure 4(b)). Next, we try to isolate nodes, say, 2 and 3 as shown in Figure 5. A min 2-3 cut consists of edges (2, 3), (6, 3), and (6, 5), so the super node (2, 3, 4, 5, 6, 7) is then divided into super nodes (2, 6, 7) and (3, 4, 5) with the minimum weight, 1.3, assigned (Figure 4(c)). Note that the min 2-3 cut puts nodes 1, 2, 6, and 7 on the same side. Therefore, super node (2, 6, 7) is attached to super node (1) (Figure 4(c)). By repeating the above process, we obtain an edge-weighted tree (Figure 6(a)) whose corresponding min-cuts are shown in Figure 6(b). The maximum flow (or the weight of minimum cut) for each pair of nodes can be found directly from the tree. For example, the maximum flow between nodes 1 and 3 is 1.3, which can be observed from Figure 6(a). A *GH cut-tree* is the tree generated by the algorithm in Figure 3.

The preceding example demonstrates that the collection of all cuts is generated by greedily deriving the minimum cut between two nodes. It was shown in Gomory and Hu [1961] that the value of the maximum flow between any two nodes, u and v, is the minimum edge weight between u and v in the cut tree.

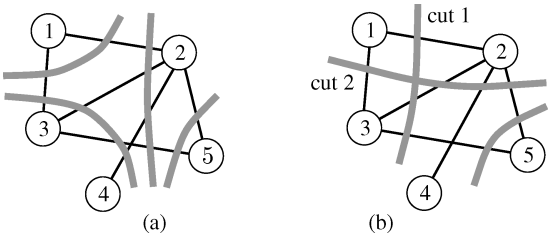


Fig. 7. (a) A set of four noncrossing cuts. (b) Cuts 1 and 2 cross each other.

By the max-flow/min-cut theorem, since such an edge represents the max flow between u and v, it is a min-cut between u and v as well. It can be proved that the collection of cuts derived by the greedy method is minimum over all u-v cuts.

Let $G = (V, E)$ be a connected graph with positive edge weights. If there exist two cuts, say $\text{cut}_G(A)$ and $\text{cut}_G(B)$, such that each one of the four sets $A \cap B$, $A - B$, $B - A$, $V - A - B$ is nonempty, then $\text{cut}_G(A)$ and $\text{cut}_G(B)$ are called *crossing cuts*. A set of cuts is *noncrossing* if no two cuts in this collection cross each other. Figure 7(a) shows a set of noncrossing cuts. Figure 7(b) shows a set of three cuts where two of them cross each other. It was shown in Gomory and Hu [1961] that the collection of min-cuts in the GH cut-tree we found is noncrossing.

THEOREM 2.1 (SEE HARTVIGSEN AND MARGOT [1995]). *The following statements are equivalent:*

—*C is a minimal collection of noncrossing cuts in G such that C contains a min u-v cut for every two nodes u and v in G.*

—*C is the minimum weight collection of noncrossing cuts in G such that C contains a u-v cut for every two nodes u and v in G.*

Theorem 2.1 implies that the weight collection (by summing all edge weights) of Figure 6(a) is minimum over all topologies.

Before proving the equality between the GH cut-tree and the OLA tree, we mention a couple of interesting observations. First, the total edge weight of the GH cut-tree (e.g., Figure 6(a)) equals the sum of all minimal cuts of its corresponding graph (e.g., Figure 6(b)). This can be verified by calculating the number of times where each weight has been summed. For example, the weight between nodes 1 and 6 of Figure 6(b) is calculated twice in totaling the minimal cuts. Similarly, the weight of (1, 6) has been summed twice in obtaining the weights of the GH cut-tree, that is, once for edge $(1, 2)$ $(1.5 = 1 + 0.5)$ and the other for edge $(2, 6)$ $(1.6 = 0.4 + 0.7 + 0.5)$. Second, the distance between any two nodes, i and j, in the GH cut-tree equals the number of cuts lying on edge (i, j) of the corresponding graph. Again, for example, there are two cuts on edge (1, 6) of Figure 6(b), and it can be found that the distance between nodes 1 and 6 is 2, as shown in Figure 6(a). Theorem 2.2 below is proved based on these two observations.

Again, let $G = (V, E)$ be a graph, $T = (V, E')$ a spanning tree of G, and (i, j) an edge of T. For brevity, let $V(P)$ be the vertex set of graph P, and $S_i(T)$

and $S_j(T)$ the connected components obtained by removing edge (i, j) from T, where $i \in V(S_i(T))$, $j \in V(S_j(T))$. The *cut-tree cost*, $\text{cost}_{\text{CT}}(T, G)$, of a spanning tree T derived from G is the sum of each cut weight in constructing T. Clearly, the cut-tree cost is exactly $\sum_{(u,v)\in E'} \text{weight}_G(\text{cut}_G(V(S_u(T))))$. For example, the cut-tree cost for the spanning tree of Figure 6(a) derived from the graph of Figure 6(b) is the sum of all cut weights (i.e., the values of all edges crossed by the dotted lines). In fact, the cut-tree cost can also be derived by summing all weights associated with each edge in Figure 6(a).

THEOREM 2.2. *A GH cut-tree of G is a spanning tree of G with minimum linear arrangement cost.*

PROOF. Let T be a spanning tree of G. By Theorem 2.1, the weight collection of the GH cut-tree is minimum. Therefore it suffices to prove $\text{cost}_{\text{LA}}(T, G) = \text{cost}_{\text{CT}}(T, G)$ as follows. Let $\text{path}_T(i, j)$ be the edge set of the path connecting nodes i and j in T. Let $[c]$ be 1 if condition c holds, and 0 otherwise. Clearly, $\text{cost}_{\text{LA}}(T, G)$ is equal to

$$\begin{aligned}
&\sum_{(i,j)\in E} \text{weight}_G(i, j) \cdot \text{dist}(T, i, j) \\
&= \sum_{(i,j)\in E} \left(\text{weight}_G(i, j) \cdot \sum_{(u,v)\in \text{path}_T(i,j)} 1 \right) \\
&= \sum_{(i,j)\in E} \left(\text{weight}_G(i, j) \cdot \sum_{(u,v)\in E'} [(u, v) \in \text{path}_T(i, j)] \right) \\
&= \sum_{(u,v)\in E'} \sum_{(i,j)\in E} \left(\text{weight}_G(i, j) \cdot [(u, v) \in \text{path}_T(i, j)] \right).
\end{aligned}$$

Now, each edge (i, j) of E can be involved in the computation of $\text{cost}_{\text{LA}}(T, G)$ in the following two cases: (1) nodes i and j are in different components obtained by removing (u, v) from T, that is, $i \in V(S_u(T))$, $j \in V(S_v(T))$ or $j \in V(S_u(T))$, $i \in V(S_v(T))$; (2) nodes i and j are in the same component obtained by removing (u, v) from T, that is, $i \in V(S_u(T))$, $j \in V(S_u(T))$ or $j \in V(S_v(T))$, $i \in V(S_v(T))$. Clearly, $[(u, v) \in \text{path}_T(i, j)] = 0$ for case 2, since the condition never holds when nodes u and v are not in $\text{path}_T(i, j)$. Therefore the theorem is proved by

$$\begin{aligned}
\text{cost}_{\text{LA}}(T, G) &= \sum_{(u,v)\in E'} \sum_{(i,j)\in E; i\in V(S_u(T)); j\in V(S_v(T))} \text{weight}_G(i, j) \\
&= \sum_{(u,v)\in E'} \text{weight}_G(\text{cut}_G(V(S_u(T)))) \\
&= \text{cost}_{\text{CT}}(T, G).
\end{aligned}$$

□

To further lower the power cost, we give a heuristic *tree-clustering* algorithm to incrementally (and locally) improve the switched capacitance [Chen et al. 1999]. The algorithm arbitrarily selects an edge e in the bus tree already derived, and tries to merge the corresponding two end-nodes (devices) of e into a generalized node (the same bus segment). If this merging results in a smaller

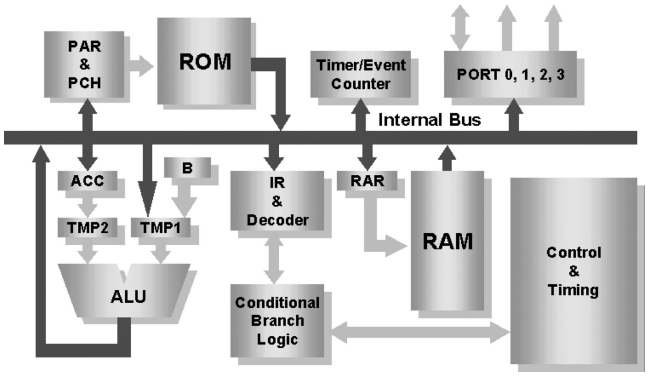


Fig. 8. The architecture of 80C51.

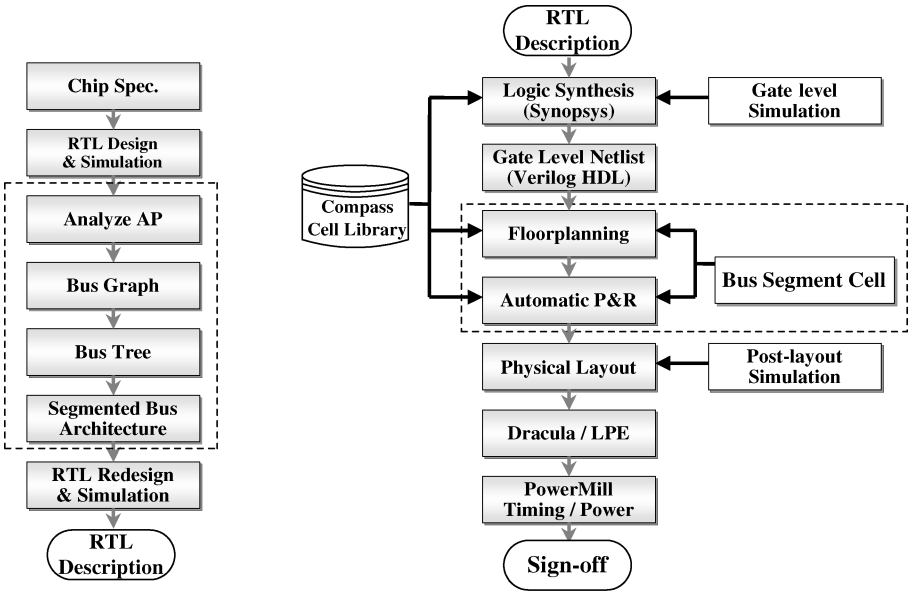
power cost, then these two nodes are melded into the same cluster. Otherwise, the algorithm gives up the merging process for e. This process repeats until merging two generalized nodes for any edge no longer reduces the switched capacitance. Note that the power cost is derived by enumerating the (weighted) switched capacitance for each pair of data communication using the formula $k_1(n-1)+k_2$ [Chen et al. 1999].

3. CHIP DESIGN AND IMPLEMENTATION

After the above theoretical analysis, implementation issues will be discussed in this section to demonstrate the feasibility and efficiency of utilizing the bus segmentation technique. A popular -controller, Intel's 80C51 [Mackenzie 1992], will be used as the test vehicle, and its architecture (without implementing bus segmentation) is shown in Figure 8. Several building blocks, such as arithmetic and logic unit (ALU), static random access memory (SRAM), read-only memory (ROM), Timer/Counter, and input/output (I/O) ports, are found in this design, and they are all connected through a common bus. In this design, the width of the bus is only 8 bits and some components such as interrupt controller are not implemented.

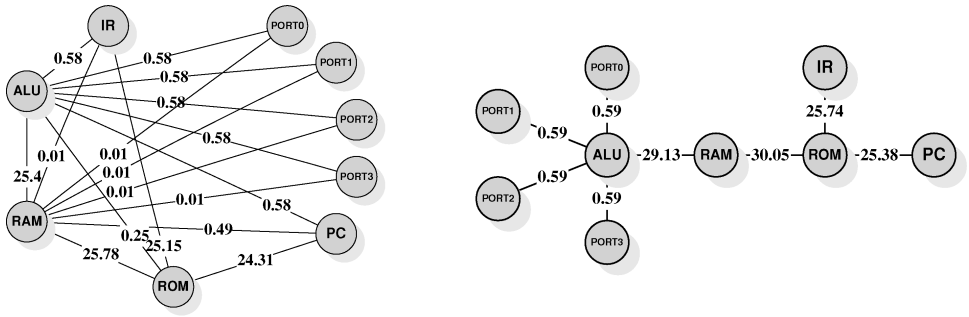
3.1 Design Flow and Design Environment

We have modified a typical application-specific integrated circuits (ASIC) design flow to include the steps for implementing the bus segmentation technique. The complete design flow is shown in Figure 9, where the dotted lines represent the design flow dedicated to the bus segmentation technique. Several extra steps are required in the register-transfer level design and physical level design. First, after the RTL design has been finished for the -controller, the circuit components and buses can be determined. The application program must be simulated on the RTL description. The communication frequencies among the components can be derived, and the bus graph can be generated as shown in Figure 10(a). Using the OLA tree analysis method discussed before, we can transfer the bus graph into a bus tree (Figure 10(b)) where highly communicating components can exchange data by traveling the minimum number of



(a) RTL design (b) Logical and physical design

Fig. 9. The flow of complete design.



(a) (b)

Fig. 10. (a) The bus graph. (b) The bus tree.

bus segments. Based on the bus tree, the location of each bus segmentation cell (BSC) can be determined. Note that each BSC physically implements the pass transistor for segment control and the tree-clustering process is not implemented. In the real implementation, each BSC is designed using an n-channel metal-oxide semiconductor (NMOS) transistor. Finally, the controller must be redesigned such that proper control signals can be issued to control the BSCs to accomplish the required data communication for each instruction. Thus, the RTL for the controller and bus communication must be redesigned as shown in Figure 9(a).

The modified RTL description can then be synthesized into a gate level circuit, and physical level design can be started. As shown in Figure 9(b), the BSC layout must be designed and included into the cell library with the sizing issue considered. The floorplan design also must consider the effect of BSC cells. Then, the BSCs will be placed and routed. Meanwhile, we know that the segmented bus architecture is strongly dependent on the particular application program (AP). Two application programs have been studied in this research. The first program (AP1) was created to do the memory test, while the second one (AP2) was designed for a keyboard controller. The bus graph and the bus tree for implementing AP2 are shown in Figure 10.

The corresponding electronic design automation (EDA) tools used in each design step are shown in Figure 9. The Verilog HDL is used to describe the RTL design, and the design compiler provided by Synopsys [Synopsys, Inc. 1994] is used to synthesize the circuits except the 128*8 SRAM and the 2K*8 program ROM. Compass cell library [Compass Design Automation. Inc. no date] is used for the design which is technology-mapped to TSMC 0.6- m single-poly double-metal (SPDM) CMOS process [TSMC no date]. Finally, PowerMill [Epic Design Technology, Inc. no date] is used to characterize the power consumption of the post-layout design.

3.2 Controller Design

The instrcutions of our -controller consume one, two, or four machine cycles, each machine cycle has six stages, and each stage has two phases. Each instruction performs data exchange between components through the bus (if necessary) under the control of machine cycles, stages, and phases. Thus, each bus segment might be turned on or off by a BSC signal for certain instructions, machine cycles, stages, and phases. In our example, an application program (AP) is stored in the embedded ROM of the -controller. The instruction set of a -controller is large, but not all of the instructions are used in this AP. Thus, we have developed a software tool to analyze the AP and identify a set of never-used and rarely used instructions.

The first approach for the controller optimization is to simplify the circuit that generates the BSC signals. This can be done by removing the stage and phase signals out of the BSC control circuit for instructions which are rarely used. We emphasize that the change in the control circuit will not affect the normal function of data transfers among the functional blocks. The basic idea behind the first approach is that synthesizing the BSC control circuit without considering the stage and phase information will reduce its size, and thus the power consumption by the control unit will be decreased. However, synthesizing the BSC control circuit without considering the stage and phase information will unnecessarily turn on some bus segments and thus consume more bus power. That is the reason why only rarely used instructions are allowed to perform this optimization. It should be noted that, using the first approach, the increase in bus power consumption will rarely occur (since the corresponding instructions are rarely used), but the saving for the BSC control circuit is valid for every instruction. Thus, it is beneficial in most cases.

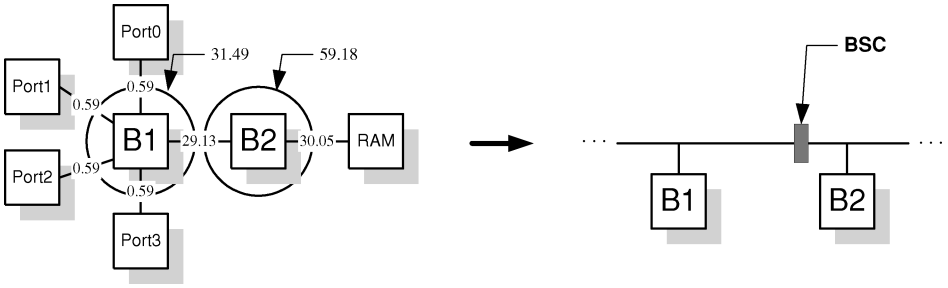


Fig. 11. The guideline for BSC placement.

The second approach pursues a more ambitious optimization process by removing all instructions which are not used in the AP. Since most APs do not use all instructions, taking unused instructions out of the control unit results in significant circuit reduction and thus in the power consumption. For example, only about 37% of instructions are used in the AP of our case. Unlike the first approach, the second approach does not introduce the side-effect of consuming more bus power since the instructions used in the AP are not affected at all.

3.3 Placement and Routing Guidelines

This subsection discusses several guidelines for physical design. (1) Better floorplan always produces a smaller chip area and results in better chip performance. Especially, considering the implementation of the bus segmentation technique, the placement of the BSCs affects not only the whole chip area but also the interconnection length of each bus segment. The interconnection length of each bus segment will in turn affect the parasitic capacitance and the power consumption. (2) Those blocks that communicate very often must be placed closer in order to reduce their interconnection length. If the switching activity among the blocks is quite frequent, it is not even necessary to insert BSCs among them (by tree clustering). Furthermore, the pins of each block that are connected to the BSCs must also be arranged to be as near the BSCs as possible. (3) When a BSC must be inserted between two blocks, its best position must also be determined. If the total switching activity of all the buses connected to B2 is more than that of all the buses connected to B1, as shown in Figure 11, the BSC between B1 and B2 must be adjusted to be as close to B2 as possible.

4. EXPERIMENTAL RESULTS

Different versions of the -controller chip are simulated using EPIC PowerMill after design rule check (DRC) and electrical rule check (ERC) processes have been finished. Table I lists the features of each version. The features include the embedded program tested (the RAM test program or our application program), bus segmentation, stage removal for BSC signals (if the utilization rate is smaller than 1% using static analysis), and unused instruction removal. These features are represented by different symbols as shown in Table I. For

Table I. The Features of All Versions (the symbols used in the version names have the following meanings: "1" stands for AP1; "2" stands for AP2; "S" stands for segmented bus design; "F" stands for full-custom design; "B" stands for removing stage for BSC signals; "U" stands for removing unused instructions for controller)

Feature \ -controller version	1	1SF	1SFB	1SFU	2	2SF	2SFB	2SFU
BSC with full-custom design		v	v	v		v	v	v
RAM test code (AP1)	v	v	v	v				
Application code (AP2)					v	v	v	v
Segmented bus design		v	v	v		v	v	v
Remove stage for BSC signals			v				v	
Remove unused instructions for controller				v				v

example, the -controller of version 1SFU is designed using full-custom BSCs (represented by "F") with the RAM test program (represented by "1") executed by PowerMill. Moreover, the bus is segmented (represented by "S") and all control circuits for unused instructions are removed from the controller (represented by "U"). Both application programs are stored in the ROM of the -controller. However, for the AP1 test (versions 1, 1SF, 1SFB, and 1SFU), only the memery testing code is executed; for the AP2 test (versions 2, 2SF, 2SFB, and 2SFU), only the keyboard controller code is executed. Both are achieved by using a long jump instruction. Design versions 1 and 2 are implemented without bus segmentation, while in others the bus segmentation technique is applied. When implementing the bus segmentation technique for design versions 1SF and 2SF, we consider only the communication frequencies between the blocks. For versions 1SFB and 2SFB, the stage information is further considered when generating the BSC control signals. For versions 1SFU and 2SFU, all unused instructions are removed from the controller design.

Here, we only show the chip layout for version 1SF in Figure 12, although all different versions have been implemented and fully simulated. Figure 13(a) shows the bus segments IB10, IB11, and IB12 of version 1SF. The interconnection structures are further detailed in Figure 13(b). It can be found from Figure 13(a) that the long wires have been dramatically shortened by the corresponding BSCs. For example, when ALU is communicating with RAM, we turn off BSC0–BSC3, and BSC5 to reduce the capacitance loading. Table II demonstrates the power consumption for all versions using different execution clock times (rather than CPU times). For example, each version starts execution for 10 ms with the clock rate of 25 Mhz. Then, the chip is reset to the initial state and executes for 15 ms, and the process is repeated until 100 ms is executed. The power consumption is estimated by averaging the current flow measured by mA using PowerMill.

According to Table II, the power consumption of version 1SF is reduced by about 24.6% from the original version 1, by executing the RAM test mode. Thus, the power consumption can be significantly reduced if the bus is segmented by full-custom BSCs. If the stage controls for BSC signals are removed for rarely utilized instructions, then the power consumption can be further reduced by 25.55% (by comparing versions 1 and 1SFB). More ambitiously, if the control circuits for all unused instructions are removed from the controller, the

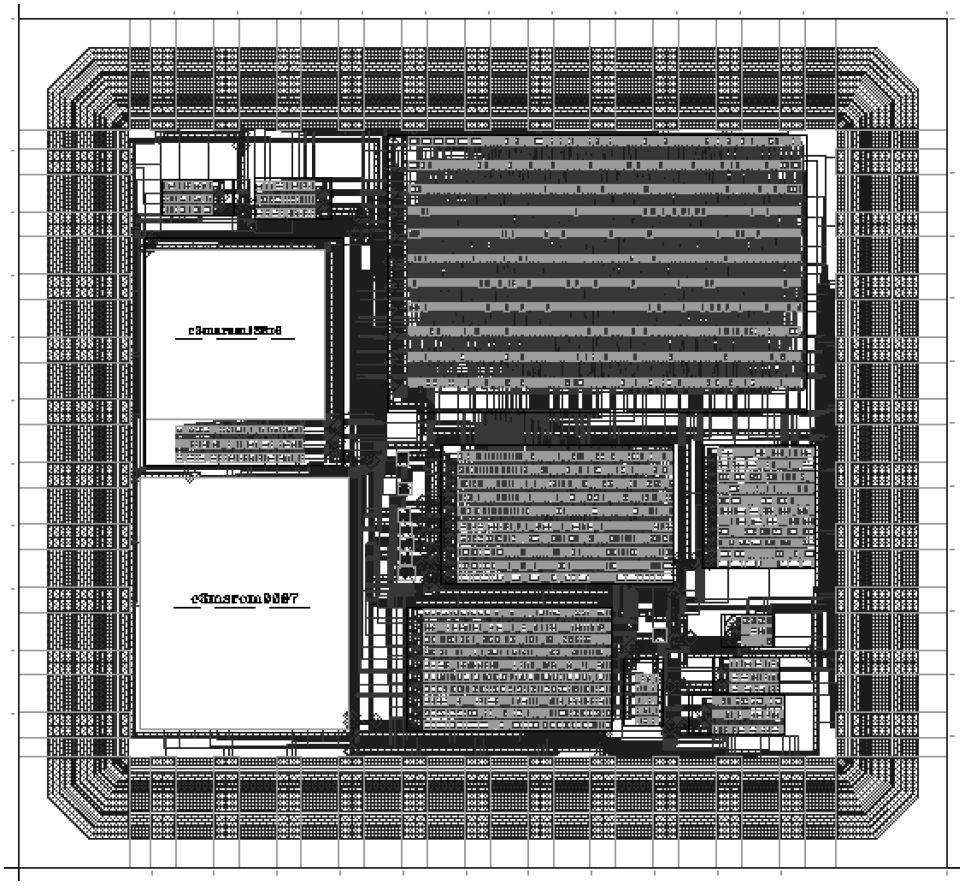


Fig. 12. Version 1SF chip layout of the -controller.

power consumption can be dramatically reduced by about 37.21% (by comparing versions 1 and 1SFU). The application program of a keyboard controller is simulated in versions 2, 2SF, 2SFB, and 2SFU. Again, one can see that significant amount of power can be saved using the bus segmentation technique by comparing version 2 with versions 2SF, 2SFB, and 2SFU. The task of testing RAM (AP1) is very trivial in that only a small number of components are involved. Thus, only a small number of bus segments need to be used (as shown in Figure 13(b)), and this provides a very good opportunity to test the performance extreme of the proposed method. From Table II, it can be observed that the power saved in RAM testing is much more significant than that in the case of keyboard controller.

Table III gives the chip sizes for different versions of the -controller. Each area includes the space occupied by I/O pads. About 6% of area overhead is introduced when the bus segmentation technique is used (by comparing versions 1 and 1SF or versions 2 and 2SF). However, the hardware overhead can be completely compensated for by removing circuits from the -controller. For example, when the control signals for all unused instructions are removed from

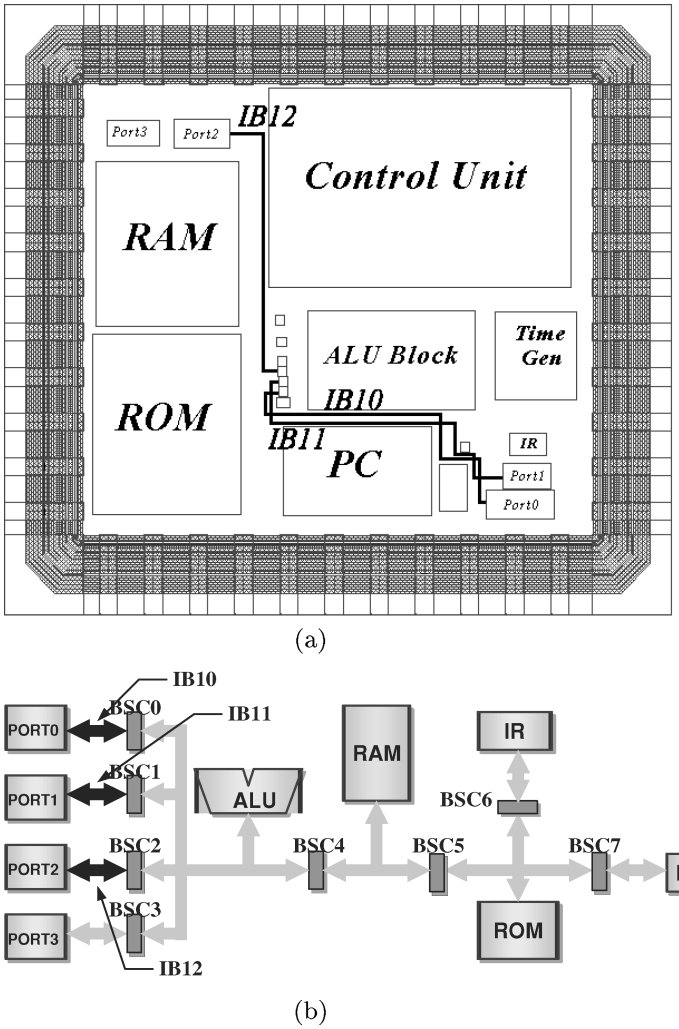


Fig. 13. (a) The locations of IB10-IB12. (b) The floorplan and IB10-IB12 routed.

Table II. Experimental Results for Power Consumption (each row corresponds to a version of -controller, each column corresponds to an amount of simulation time in ms, and each cell is the amount of consumed power in mA; operating frequency is 25 Mhz)

Ver.\time	10	15	20	25	30	35	40	45	50	100
1	20.018	20.156	20.209	20.287	20.286	20.217	20.272	20.250	20.225	20.384
1SF	16.126	15.843	15.624	15.583	15.501	15.396	15.405	15.367	15.339	15.360
1SFB	16.031	15.706	15.524	15.461	15.363	15.256	15.273	15.220	15.195	15.175
1SFU	13.556	13.048	13.198	12.905	13.007	12.884	12.894	12.854	12.832	12.799
2	20.398	20.737	20.904	20.423	20.683	20.776	20.566	20.552	20.670	20.822
2SF	18.210	18.011	18.073	17.694	17.751	17.763	17.630	17.585	17.657	17.711
2SFB	18.097	17.844	17.888	17.503	17.603	17.592	17.447	17.399	17.474	17.494
2SFU	15.398	15.570	15.826	15.603	15.746	15.781	15.720	15.702	15.787	15.899

Table III. Chip Sizes of All Versions

Version	Width (m)	Height (m)	Area (m^2)	Area ratio
1	3843.025	3745.700	14394818	1
1SF	4083.575	3730.900	15235409	1.0583953
1SFB	3944.150	3752.300	14799634	1.0281223
1SFU	3711.075	3550.625	12991081	0.9024831
2	3843.025	3745.700	14394818	1
2SF	4083.575	3730.900	15235409	1.0583953
2SFB	3944.150	3752.300	14799634	1.0281223
2SFU	3738.125	3649.150	13640978	0.9476312

the -controller, the area is reduced by about 9.75% when versions 1 and 1SFU are compared, and by about 5.24% when versions 2 and 2SFU are compared.

5. CONCLUSION

We presented the design theory and implementation issues of a bus segmentation method for lowering the power dissipation on system buses. If the highly communicating devices are clustered and separated from the rest of devices, the bus capacitance required to be charged/discharged becomes smaller and the power dissipation can be saved. Computer simulation shows that the bus delay can also be improved [Chen et al. 1999]. The bus segmentation method can be used in ASIC design where the application program is fixed, so the communication frequency between each pair of devices can be well estimated by computer simulation. We have applied the method to a low-power -controller where the bus is partitioned into several bus segments. Power analysis based on the postlayout simulation demonstrates that a significant amount of power can be saved. We expect to save more power if the bus contains more data bits. In Mehra et al. [1997], a method proposed to partition the given algorithm into spatially local clusters will ensure the majority of data transfers take place within clusters. Thus, our method can be regarded as an extension of Mehra et al. [1997], with the objective of further reducing the power consumption for intercluster communications. This paper only concentrates on dynamic bus trees, and it will be interesting to extend the results to general bus structures. Additionally, guidelines for tree-clustering should be investigated.

ACKNOWLEDGMENT

We thank R. Ravi for bringing Hartvigsen and Margot [1995] to our attention and showed us its relationship with the OLA tree.

REFERENCES

BELLAOUAR, A., ABU-KHATER, I. S., AND ELMASTRY, M. I. 1995. An ultra-low-power CMOS on-chip interconnect architecture. In *Symposium on Low Power Electronics. Digest of Technical Papers* (Oct. 1995). 52–53.

CARDARILLI, G. C., SALMERI, M., SALSANO, A., AND SIMONELLI, O. 1996. Bus architecture for low-power VLSI digital circuit. In *International Symposium on Circuits and Systems*, vol. 4 (May 1996). 21–24.

CAUFAPE, S. AND FIGUERAS, J. 1996. Power optimization for delay constrainted CMOS bus drivers. In *European Design and Test Conference* (Mar. 1996). 205–211.

CHANDRAKASAN, A. P. AND BRODERSEN, R. W. 1995. *Low power Digital CMOS design*. Kluwer Academic Publishers, Boston, MA.

CHANDRAKASAN, A. P., SHENG, S., AND BRODERSEN, R. W. 1992. Low-power CMOS digital design. *IEEE J. Solid-State Circ. 27*, 4 (Apr.), 472–484.

CHEN, J. Y., JONE, W. B., WANG, J. S., LU, H.-I., AND CHEN, T. F. 1999. Segmented bus design for low-power system. *IEEE Trans. VLSI Syst. 7*, 1 (Mar.), 25–29.

COMPASS DESIGN AUTOMATION, INC. no date. *0.6- m Passport Preliminary Design Kit*. Compass Design Automation, Inc.

EPIC DESIGN TECHNOLOGY, INC. *PowerMill User Manual*. Epic Design Technology, Inc., Mountain View, CA.

GOLSHAN, R. AND HAROUN, B. 1994. A novel reduced swing CMOS bus interface circuit for high speed low power VLSI systems. In *International Symposium on Circuits and Systems*, vol. 4 (May 1994). 351–354.

GOMORY, R. E. AND HU, T. C. 1961. Multi-terminal network flows. *J. SIAM 9*, 4 (Dec.), 551–569.

HARTVIGSEN, D. AND MARGOT, F. 1995. Multiterminal flows and cuts. *Operat. Res. Lett. 17*, 5, 201–204.

HIRAKI, M., KOJIMA, H., MISAWA, H., AKAZAWA, T., AND HATANO, Y. 1995. Data-dependent logic swing internal bus architecture for ultralow-power LSI's. *IEEE J. Solid-State Circ. 30*, 4 (Apr.), 397–402.

IKEDA, M. AND ASADA, K.. 1994. A reduced-swing data transmission scheme for resistive bus lines in VLSIs. In *European Design Automation Conference* (Feb. 1994). 546–550.

LIU, D. AND SVENOSON, C. 1994. Power consumption estimation in CMOS VLSI chips. *IEEE J. Solid-State Circ. 29*, 6 (June), 663–670.

MACKENZIE, S. 1992. *The 8051 Microcontroller*. Macmillan Publishing Company, London, U.K.

MEHRA, R., GUERRA, L. M., AND RABAEY, J. M. 1997. A partitioning scheme for optimizing interconnect power. *IEEE J. Solid-State Circ. 32*, 3 (Mar.), 433–443.

NAJAGOME, Y., ITOH, K., ISODA, M., TAKEUCHI, K., AND AOKI, M. 1993. Sub-1-v swing internal bus architecture for future low-power Ulsi's. *IEEE J. Solid-State Circ. 28*, 4 (Apr.), 414–419.

STAN, M. R. AND BURLESON, W. P. 1995a. Bus-invert coding for low-power I/O. *IEEE Trans. VLSI Syst. 3*, 1 (Mar.), 49–58.

STAN, M. R. AND BURLESON, W. P. 1995b. Coding a terminated bus for low power. In *Fifth Great Lake Symposium on VLSI* (Mar. 1995). 70–73.

SUNDSTORM, R., PHAM, P., ESGAR, D., AND PETTY, C. 1990. A low power differential bus untilizing novel level bus technique. In *Bipolar Circuits and Technology Meeting* (Sept. 1990). 144–147.

SYNOPSYS, INC. no date. *HDL Compiler for Verilog Refernce Manual*. Synopsys, Inc., Mountain View, CA.

TSMC. no date. *0.6- m Logic CMOS Process (0.6U-TW-14L-DS)*. TSMC, Hsin-Chu, Taiwan, R.O.C.

Received August 1999; accepted January 2002