Interconnect-aware High-level Synthesis for Low Power

Lin Zhong and Niraj K. Jha
Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08544
{lzhong, jha}@ee.princeton.edu

Abstract— **Interconnects (wires, buffers, clock distribution networks, multiplexers and busses) consume a significant fraction of total circuit power. In this work, we demonstrate the importance of optimizing on-chip interconnects for power during high-level synthesis. We present a methodology to integrate interconnect power optimization into high-level synthesis. Our binding algorithm not only reduces power consumption in functional units and registers in the resultant register-transfer level (RTL) architecture, but also optimizes interconnects for power. We take physical design information into account for this purpose. To estimate interconnect power consumption accurately for deep sub-micron (DSM) technologies, wire coupling capacitance is taken into consideration. We observed that there is significant spurious (*i.e.*, unnecessary) switching activity in the interconnects and propose techniques to reduce it. Compared to interconnect-unaware power-optimized circuits, our experimental results show that interconnect power can be reduced by 53.1% on an average, while reducing overall power by an average of 26.8% with 0.5% area overhead. Compared to area-optimized circuits, the interconnect power reduction is 72.9% and overall power reduction is 56.0% with 44.4% area overhead.**

I. Introduction

High-level synthesis for low power has attracted significant attention [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. It takes as its input a behavioral description in the form of a control-data flow graph (CDFG) and outputs a power-optimized RTL circuit. Most previous work tries to minimize the power consumed by the datapath while ignoring the power consumed by the interconnects. As shown in [12], interconnects consume a significant fraction of total circuit power. Since wire delay is becoming more significant, wire buffer insertion has become popular [13]. This in turn has increased the portion of circuit power consumed by interconnects.

High-level synthesis can target either bus-based or multiplexer-based interconnections among functional units and registers. It has a significant impact on the switching activity and topology of the interconnects in the resultant design. Thus, it is important to optimize interconnects during high-level synthesis.

In [14], a technique is proposed to optimize bus power by appropriately binding data transfers to busses. It, however, does not take wire length into consideration, and is not applicable to multiplexer-based interconnects.

Taking physical level information into account during high-level synthesis has also attracted a lot of attention [15], [16], [17], [18], [19], [20], [21], [22]. Earlier work used floorplanning information in high-level synthesis to estimate the area and performance of the design more accurately. Only some recent works have tried to take interconnect power consumption into consideration [2], [20]. In [20], the switching activity on CDFG edges is profiled. It is used with the floorplanner to optimize the power consumption of inter-module data transfers. Only the floorplanner is made interconnect power-aware. Moreover, the coupling effect between wires is not taken into consideration. In [2], the authors preserve the locality and regularity in the input behaviors to optimize bus power consumption in bus-based architectures. However, they do not distinguish between busses with different switching activities. Moreover, locality and regularity in the behavior does not necessarily imply locality in the physical design due to the mapping of behavioral identities (operations and variables) to physical identities (functional units and registers).

In this work, we provide a comprehensive treatment of interconnect-aware high-level synthesis for low power. We evaluate the power consumption in the steering logic and clock distribution network in addition to data transfer wires using early floorplanning information. Moreover, since coupling capacitances are expected to dominate the wire capacitance in future technologies, we take coupling into account while estimating wire power consumption. We propose metrics to guide the binding process to preserve and create locality in the physical implementation. Unlike [2], our methodology does not assume behavioral locality and regularity.

It is well-known that there can be significant spurious switching activity (SSA), *i.e.*, activity not required by the behavioral specification, in the functional units of the datapath [23], [24]. We found that there is also significant SSA in the interconnects. This increases the power consumption in the interconnects as well as other circuit components. We propose techniques to suppress interconnect SSA.

The paper is organized as follows. We present our observations and motivational examples in Section II. We discuss RTL interconnect power estimation in Section III. We present methodologies for interconnect-aware binding in Section IV. We address the problem of SSA in interconnects in Section V. We present a framework of our interconnect-aware high-level synthesis system in Section VI. We give experimental results and conclude in Sections VII and VIII, respectively.

II. Observations and Motivational Examples

In this section, we first present some observations concerning RTL interconnect power consumption. Then we give a motivational example for our proposed techniques.

A. Observations

Our observations deal with what impact interconnect power has and how SSA impacts interconnect power. We use a low power high-level synthesis tool, called SCALP [10], to obtain the RTL circuits and the power consumption of the controller/datapaths, aided by our interconnect power estimation methods detailed in Section III. SCALP is interconnect-unaware.

A.1 Interconnect power consumption

We evaluated eight high-level synthesis benchmarks implemented in 0.18μm technology. *Chemical* and *IIR77* are IIR filters used in the industry. *DCT_IJPEG* is the Independent JPEG Group's implementation of digital cosine transform (DCT) [25]. *DCT_Wang* is the DCT algorithm named after the inventor [26]. Both DCT algorithms work on 8×8 pixels. *Elliptic*, an elliptic wave filter, and *Diffeq*, a differential equation solver, are from the NCSU CBL high-level synthesis benchmark suite [27]. *Jacobi* is the Jacobi iterative algorithm for solving a fourth order linear system [28]. *WDF* is an FIR wave digital filter. The smallest benchmark is *Diffeq* with six multiplications, two additions and two subtractions. The largest is *Jacobi* with

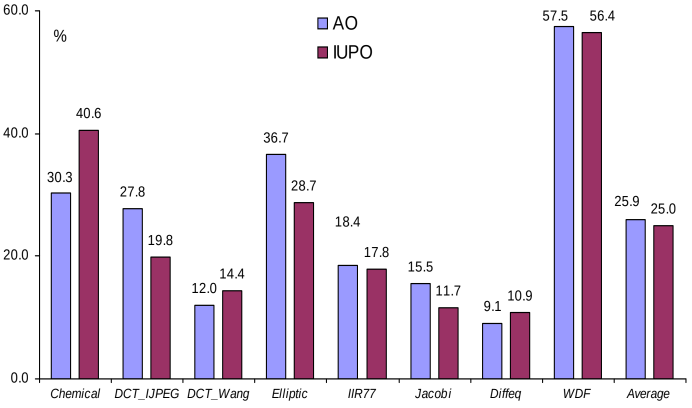

Fig. 1. Percentage of total power consumed by interconnects in area-optimized and power-optimized circuits.

24 multiplications, eight divisions, eight additions and 16 subtractions.

We used the wire capacitances from [13] along with an RTL design library from NEC [29], [30] to estimate the power consumption for the benchmarks. Fig. 1 shows the percentage of total power consumed by RTL interconnects for area-optimized (AO) and interconnect-unaware power-optimized (IUPO) circuits, respectively. In both area-optimized and power-optimized circuits, interconnects consume a significant percentage of total power, with the average percentage being 25.9% and 25.0%, respectively.

A.2 Spurious switching activity (SSA)

Let us define the RTL interconnects connected to the output of an RTL unit in the datapath as its *output network*. A functional unit with more than one operation or a register with more than one variable bound to it outputs values of multiple variables onto its output network in different clock cycles. In general, in a given clock cycle, the output value has to be routed to only a small portion of the output network. However, if care is not taken, many other parts of the network may also experience switching activity. Moreover, many functional units, for example, a ripple-carry adder, output a lot of glitches before the output finally settles down to the correct value [31]. These glitches also contribute significantly to the output network power consumption.

Fig. 2 shows the percentage of total interconnect power consumed through SSA for the eight benchmarks optimized for area and power. For both area-optimized and power-optimized cases, SSA consumes a significant percentage of total interconnect power, with the average being 62.6% and 29.2%, respectively, even when glitches are not taken into account. Area-optimized circuits incur a higher percentage of SSA power in their interconnects because their datapath units are usually heavily shared.

B. Motivational example

In this section, we present an example to motivate the techniques presented ahead. Fig. 3(a) shows the CDFG for *Diffeq* and one of its possible schedules. Suppose that there are one adder, one subtracter and three multipliers in the datapath. Then the bindings for operations +1, +2, −1 and −2 are fixed. However, how the multiplication operations are bound to the three multipliers will significantly affect the interconnect structure. For example, Fig. 4 shows two bindings of operations to functional units (an oval depicts a functional unit). It is clear that the bindings in Fig. 4(a) have fewer data exchanges between different functional units than the bindings in Fig. 4(b). Binding of variables to registers has a similar effect on the interconnect structure. The fact that functional unit and register binding have significant impact on the interconnect structure is well-known. However, optimization of interconnect power through judicious binding has not been adequately explored before.

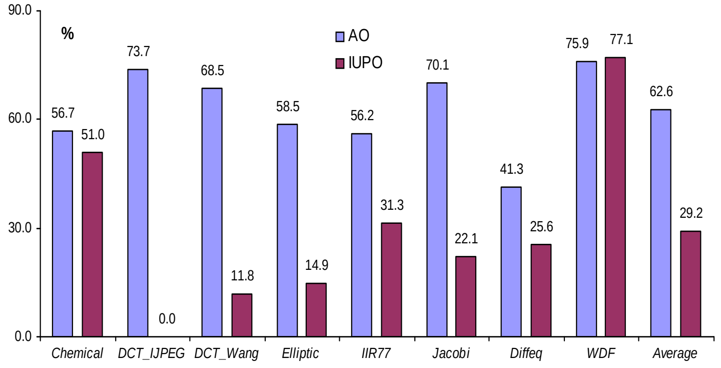

Fig. 2. Percentage of total interconnect power consumed by SSA.

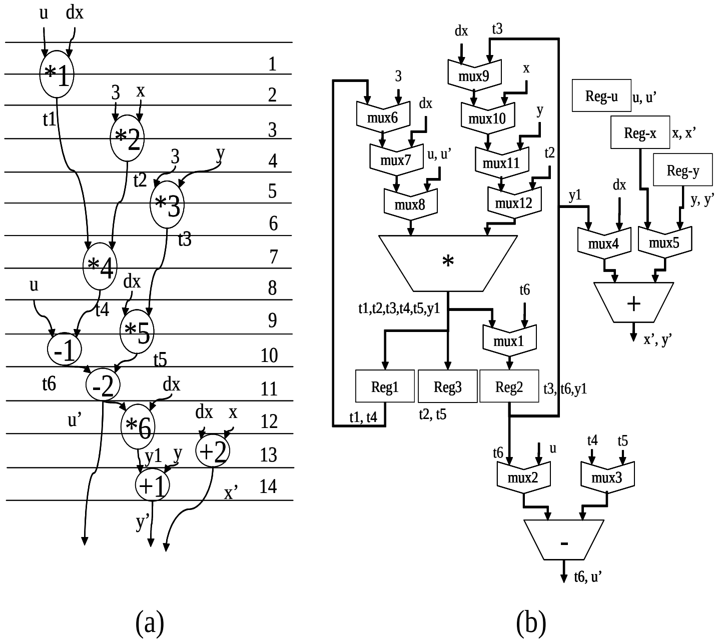

Fig. 3. *Diffeq*: (a) scheduled CDFG, and (b) an RTL implementation.

Next, let us consider an RTL implementation of *Diffeq*, as shown in Fig. 3(b), with one multiplier, one subtracter, one adder, 12 multiplexers and six registers.

The schedule is the same as that in Fig. 3(a). Let us examine the output network of the multiplier. Values of variables t1, t2, t3, t4, t5 and y1 are transmitted to registers Reg1, Reg2 and Reg3. However, each of these registers only needs a subset of these variables. Such a data broadcasting introduces SSA not only in the data transfer wires but also in the steering logic along these wires, such as multiplexer mux1. The impact of SSA in the interconnect is not limited to the interconnect itself. For example, the multiplier, adder and subtracter only need variable t3, y1 and t6 from Reg2, respectively. However, Reg2 sends values of all of these variables to all these functional units. As in the output network of the multiplier, the data broadcasting causes SSA in the wires and along the multiplexer tree above the functional units. Moreover, according to the schedule in Fig. 3(a), when the transition from t3 to t6 occurs in Reg2, the adder is idle (cycle 10 to 11), and when the transition from t6 to y1 occurs, the subtracter is idle (cycle 13 to 14).

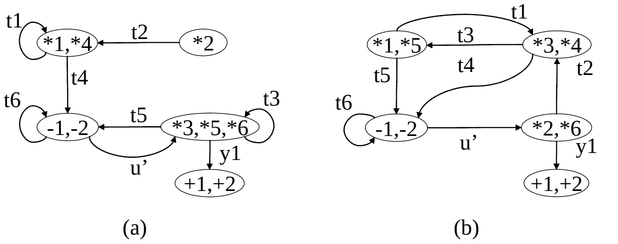

Fig. 4. Different bindings for the multiplication operations.

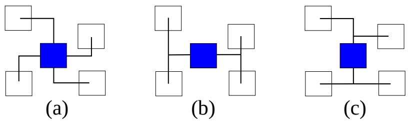

Fig. 5. Output network of different topologies: (a) fully dedicated, and trunk-branches output network with the trunk being (b) horizontal, and (c) vertical.

Therefore, if care is not taken, the SSA may propagate into the adder and subtracter. By eliminating SSA in the output networks, we can also suppress a significant portion of the SSA in the functional units.

III. RTL Interconnect Power Estimation

Circuit components visible at the RTL can be categorized into datapath units (DPUs), which generate or consume data, and interconnects, which route data between DPUs. RTL DPUs consist of functional units, registers, memory banks and controllers, *etc.* RTL interconnects consist of data transfer wires/busses, buffers on the wires, clock distribution network and steering logic. Power, delay and area of interconnects within DPUs are included in the corresponding numbers or models for the DPUs provided in the RTL design library.

A. Output network topology

A DPU's output network is defined as the RTL interconnect components connecting its output to the inputs of other DPUs. The data transfer between two DPUs is assumed to be center-to-center and rectilinear for the sake of simplicity. A bus-based architecture shares output networks among DPUs through busses. Since sharing the output network imposes a high performance and power penalty [31], we assume the output network is not shared in the multiplexer-based architecture we use. However, it is still possible to share wires inside one output network. Fig. 5(a) shows one DPU sending out data to four other DPUs through a fully dedicated output network. If the DPU sends all the data to all the dedicated data transfer wires, minimization of total power consumption in the output network will be the same as minimization of total wire length in the output network. Therefore, a minimum spanning tree (MST) output network in the Steiner tree style [32] would have the least interconnect power consumption. However, if we take steps to reduce the SSA in the interconnect, minimal total length does not necessarily mean minimal total power consumption. We introduce a *trunk-branches* style Steiner tree for the output network as shown in Figs. 5(b) and 5(c). The DPU sends all the data to a trunk which is either vertical or horizontal. All the other DPUs receive data from the shared trunk through perpendicular dedicated branches. Whether the trunk is vertical or horizontal depends on which one yields smaller power consumption. In Section V, we will compare three output network styles, fully dedicated, optimal total length shared and trunk-branches, in detail.

B. Data transfer wires

Unlike older technologies, the effective capacitance for wires in DSM technologies is also dependent on the switching activity due to coupling between neighboring wires [13]. Several interconnect energy models have been proposed to take the coupling effects into consideration [33], [34], [35], [36]. We use the table look-up method proposed in [34]. Other models, such as the one proposed in [35], can be readily incorporated into the profiling driven power estimation technique used in our work. In DSM technologies, the switching activity of a line also depends on that of its neighboring lines. Its capacitance is also affected by the switching activity of it neighbors which determines how the coupling capacitances between lines are charged and discharged. This necessitates consideration of *three-line switching patterns*. Using this method, PSPICE and capacitance data from [13], we estimated the unit-length switched capacitance (the product of switching activity and capacitance) of the central lines for different three-line switching patterns. These data are stored in a table, called *pattern-power table*, for later use. Knowing the length of a wire and its three-line switching patterns, this table can be used to estimate the switched capacitance of the whole line. To obtain the unit-length power consumption for each data transfer, we simulate the CDFG with its typical input traces to get all the three-line switching patterns for each CDFG edge (note that each CDFG edge corresponds to one data transfer, or, multi-bit wires in the RTL implementation). This step needs to be carried out only once for each behavior.

Our high-level synthesis tool, as discussed later, is based on variable-depth iterative improvement of an initial RTL implementation of the behavior. Thus, at each stage of optimization, a complete RTL description of the circuit is available. We floorplan the RTL DPUs to estimate the wire length. The unit-length switched capacitance of data transfers between two units is used as their communication cost during floorplanning. We partition the datapath into a binary tree hierarchy which has balanced area and minimal communication cost between the resultant partitions. The algorithm is an extended version of the one given in [37]. It tends to put DPUs, with high unit-length switched capacitance data transfers between them, into the same partition, thus reducing the total switched capacitance. After floorplanning, output networks are constructed using the proposed models and the length of data transfers is estimated.

The switching activity of wires depends on the switching activity of the outputs of DPUs they are connected to. The latter depends on binding for a given schedule. Moreover, binding also impacts the topology of the output network, which affects wire length through floorplanning and routing.

C. Steering logic

Steering logic includes multiplexers. They are treated in the same fashion as RTL DPUs. The power consumption is based on their RTL power macro-model [1]. Just as in the case of wires, the input switching activity of steering logic depends on the output switching activity of functional units and registers, and hence on binding for a given schedule.

D. Clock distribution network

Many DPUs are clocked. Such units include registers and pipelined functional units. In the floorplanner, we increase the communication cost between clocked units in order to place them closer to each other. After floorplanning, an MST is constructed for these clocked units. The power consumption of the MST is estimated using unit-length wire power and the total length of the MST.

E. Buffers

There are two kinds of buffers related to RTL interconnects. First, the output buffer driving the output network for each DPU, and second, buffers (a.k.a. repeaters) inserted along a long wire to reduce its delay. One can view the latter as a distributed output buffer. These buffers increase interconnect power consumption significantly. Studies show that for delay-optimized buffer insertion, the total switched capacitance of inserted buffers is about 1.1 times the total switched capacitance of the corresponding wire [38]. For output buffers, it has been shown that their total switched capacitance is also approximately 1.1 times that of the corresponding wire [39]. Therefore, in the following, we do not distinguish between these two kinds of buffers and treat them as an integral part of the wire.

F. Rent's rule

Rent's rule [40] states that the following relationship exists between several chip parameters.

$$T = AK^P \tag{1}$$

where T is the number of terminals, K the number of blocks within the chip, A the average number of terminals for one block, and p the Rent exponent. Rent's rule has been widely used to estimate power dissipation in interconnects. However, this requires we make assumptions about p and A, e.g., their values and whether they are constant. In the case of high-level synthesis, different binding and scheduling may generate circuits with different p and A, as shown in Fig. 4. Therefore, we do not use Rent's rule for interconnect power estimation.

IV. Interconnect-aware binding

Just by estimating interconnect power during high-level synthesis and including it in the cost function, one can indirectly optimize interconnect power. However, this is not enough. In this section, we propose explicit methods to make binding interconnect-aware. The overall high-level synthesis flow is presented in Section VI.

A. Neighborhood-sensitive binding

To reduce the communication cost, the floorplanner tries to place DPUs, which exchange data, closer to each other. A data transfer is called *local* if it happens between two adjacent DPUs in the floorplan. Some researchers have proposed techniques to localize data transfers [2], [41]. In [41], an algorithmic transformation is proposed to localize data transfers in VLSI array processors. In [2], behavioral partitioning is advocated for exploiting locality. However, they are only effective for highly regular signal processing behaviors. Besides, they do not distinguish between behavioral locality and physical locality. We use a neighborhood-sensitive binding technique which does not rely on the behavior and effectively preserves/creates physical locality in circuits.

For a DPU, its *RTL neighbors* are defined as the DPUs, except itself, that exchange data with it. Binding decides its RTL neighbors. On the other hand, its *physical neighbors* are defined as the DPUs adjacent to it in the floorplan. To localize data transfers, we should ensure that as many RTL neighbors as possible are also physical neighbors, especially those that conduct data exchange with it consisting of high unit-length power consumption. However, the physical neighborhood capacity of a DPU is limited, and is related to its geometry. We define the *neighborhood crowd*, NC, of a DPU, as

$$NC = \sum_i g(\sqrt{Area_i/Area}) \tag{2}$$

where $g(x)$ is x if $x < 1$, 1 if $x \geq 1$, $Area$ is its area, and $Area_i$ is the area of its ith RTL neighbor. The area information is obtained from the RTL design library. The definition of NC is based on the following observations. First, the capacity of a DPU to have physical neighbors is decided by its width and height. Second, it can have more small physical neighbors than big ones. These two observations lead to the use of $\sqrt{Area_i/Area}$ as the argument for function g. Another observation is that when a smaller DPU has a larger physical neighbor, that neighbor tends to just dominate one side of the smaller unit. This leads to the choice for $g(x)$.

As a DPU's NC gets larger, it becomes more difficult to make all its RTL neighbors its physical neighbors as well. In the iterative improvement algorithm we employ for high-level synthesis which is discussed later, various moves are defined. Two binding moves that affect NC are DPU sharing and splitting. When making such moves, in addition to evaluating the power/area gains in the DPUs, we also use an NC-based factor which reflects how much the average DPU NC changes with the move. Moreover, if such a move increases the NC of the new DPU beyond a certain threshold, it is simply rejected. This approach not only reduces power, but also area due to an improved floorplan.

B. Communication-sensitive binding

We add a weighted communication gain to the cost gain for DPU sharing moves. It is based on the unit-length switching power of the data exchange between the corresponding two DPUs. This tends to merge DPUs which have intensive data-exchange between them. It is estimated as

$$G(A, B) = \beta\sqrt{Area}(P_{sw}(A, B) + P_{sw}(B, A)) \tag{3}$$

where A and B are two DPUs of the same type (note that we only merge DPUs of the same type), β is a constant weight to reflect the relative importance of the communication cost, $Area$ is the library area for the DPU, and $P_{sw}(A, B)$ is the unit-length switching power for data transfers from A to B, which can be computed based on CDFG simulation, and functional unit and register bindings. $P_{sw}(B, A)$ is similarly defined.

In our iterative improvement based high-level synthesis system, scheduling changes are implicit. It starts with an as-soon-as-possible schedule. After each binding and functional unit selection move, the behavior is rescheduled if the move so necessitates. Therefore, when we make the binding interconnect-aware, the scheduling is made interconnect-aware implicitly.

V. Reducing Spurious Switching Activity

Signal gating (a.k.a operand isolation) based power management has been used to freeze the inputs to a DPU to suppress SSA [1], [23], [42]. The same idea can be used for interconnects as well. The difference is that we gate a signal with SSA before it propagates onto interconnects. When the controller is properly respecified [24], it also reduces SSA in downstream DPUs. In this section, we propose implementations for the gating mechanism. The methods we propose next are relatively independent of the high-level synthesis process. Thus, they can be applied in a post-processing step after high-level synthesis, *i.e.*, the resultant RTL architecture can be further optimized with these methods even if the architecture is obtained by some other high-level synthesis system.

A. Signal gating with filler values

If one part of the circuit is not doing anything useful in a given cycle, ideally its inputs should be frozen, i.e., remain unchanged. Various options exist for ensuring this. Transparent latches were proposed earlier to freeze the inputs of functional units [23]. However, often the power savings obtained may not justify the large overheads introduced by multi-bit latches. Another option may be to set the output to tri-state when it is not active. However, the tri-state gate output will usually drift to some mid-way voltage value if the gate is not refreshed in a short time. Such a mid-way value may cause a large power consumption in downstream DPUs. Thus, their use is justified in very limited cases, which will be addressed later. In [42], it was mentioned that freezing the inputs to zero (AND gated) or one (OR gated) also reduces SSA. To make full use of data correlations inside the datapath, we propose to freeze the inputs to a fixed (hardwired) value. We call this value *the filler value.*

Algorithm 1 Pseudo-code for computing the filler value for the output network from one DPU (*Sending_DPU*) to an input port of another (*Receiving_DPU_port*).

```
$V$ = get_output_variable($Sending\_DPU$);
$U$ =get_input_variable($Receiving\_DPU\_port$);
$V' = V \cap U$;
$W = V - V'$;
for each bit $n$ do
  $P_f^n(0) = 0$;$P_f^n(1) = 0$;
  for each $w \in W$ and each $v \in V'$ do
    $P_f^n(0) = P_f^n(0) + P(w,v)P^n(w,1)$;
    /*$P^n(w,1)$ is the probability that the $n$th bit of $w$ is 1.*/
    $P_f^n(1) = P_f^n(1) + P(w,v)P^n(w,0)$;
    /*$P^n(w,0)$ is the probability that the $n$th bit of $w$ is 0.*/
  end for
  if $P_f^n(0) > P_f^n(1)$ then
    $f^n = 1$;
  else
    $f^n = 0$;
  end if
end for
return $f$;
```

Algorithm 1 contains the pseudo-code for computing the filler value to minimize interconnect switching activity. Suppose the set of variables sent by DPU D_x (*Sending_DPU*) to all the DPUs in its output network is V. Let V' denote the subset of V which contains all the variables used by an input port P_y of another DPU (*Receiving_DPU_port*). Variables in V' are called *desired variables* for P_y. For the RTL circuit in Fig. 3(b), V for the multiplier's output network is {t1,t2,t3,t4,y5,y1} and its V' for register Reg2 is {t3,y1}. For variables $v \in V'$ and $w \in V - V'$, we obtain the probability $P(w,v)$ that values of w and v will be output consecutively, irrespective of their order. $P(w,v)$ can be computed by simulating the CDFG in which the binding and scheduling information has been back-annotated. Moreover, this simulation can also yield the probability for the nth bit of w to be 1 and 0, respectively, when w is output right before or after any variable from $V - V'$. Let us denote these probabilities by $P^n(w,1)$ and $P^n(w,0)$, respectively. Since w is not used by P_y and will be replaced by the filler value f, we need to decide what f should be in order to minimize the switching activity. For the nth bit of f, say f^n, transitions take place when it is different from the values of the desired variables which are output right before or after it. The transition probabilities for f^n to be 0 and 1, $P_f^n(0)$ and $P_f^n(1)$, are

$$P_f^n(0) = \sum_{w \in V-V'} \sum_{v \in V'} P(w,v)P^n(w,1) \tag{4}$$

$$P_f^n(1) = \sum_{w \in V-V'} \sum_{v \in V'} P(w,v)P^n(w,0) \tag{5}$$

When $P_f^n(0) > P_f^n(1)$, f^n is set to 1, else 0. Thus, we can statistically minimize the bit transition activity in the output network from DPU D_x to port P_y by introducing the filler value. Since the switched capacitance is highly dependent on switching activity in the physically neighboring wires (due to coupling), we cannot guarantee that the filler value thus chosen will yield minimal spurious switching power in the wires. However, it has been found to reduce spurious switching power significantly. When the data are totally random, *i.e.*, $P^n(w,1) = P^n(w,0) = 0.5$, the optimal filler value can be either 1 or 0, which reduces to the method in [42].

The overhead for setting the input to a fixed value is very low compared to the overhead for latches. One AND gate is enough for setting the value to 0 and one OR gate for 1.

B. Tri-state buffer based technique

When one idle cycle of part of a circuit is both preceded and followed by an active cycle, setting its inputs to a filler value in that idle cycle will save very limited or no power at all. However, in this case and other situations in which the consecutive idle cycles are not long enough, instead of using a filler value, we can use tri-state gating without letting its output voltage value drift to cause any problems. In $ASIC$s, a properly sized tri-state buffer can easily hold its output for a few clock cycles without causing a problem in the downstream circuit. Therefore, before deciding which gating logic to use, we examine the schedule for data transfers. In case the gating logic has to gate for more than two consecutive cycles, filler value gating is used. Otherwise, a tri-state buffer is used.

C. Overhead for gating

When applying the gating techniques proposed in the above subsection, we introduce overheads in the controller for generating the gating signal and due to the extra control wires. Such overheads should be taken into consideration when examining the benefits of gating a signal to suppress SSA. We adopt a method from [43] and make similar assumptions for this purpose. We assume a typical gating control signal requires 10 gates to generate, a typical such gate has three fanouts, the gate is four times the minimum gate size and the switching activity is 0.5. The switched capacitance, SC_{cl}, and area overhead, A_{oh}, in the controller are estimated as

$$SC_{cl} = 0.5N_c{\cdot}10{\cdot}4{\cdot}C_{in}^{min} \cdot 3 = 60N_cC_{in}^{min} \tag{6}$$

$$A_{oh} = 4N_cA^{min}{\cdot}1.2 = 4.8N_cA^{min} \tag{7}$$

where N_c is the number of gating control signals, C_{in}^{min} the input capacitance for a minimum sized gate, A^{min} the minimum sized gate area and the scaling factor 1.2 for area overhead is to account for spacing between gates and the wiring space. Moreover, the switched capacitance overhead of the gating control signal wires is estimated as

$$SC_{cw} = 2\sqrt{Area}C_w^{min}N_{sw} \tag{8}$$

where $Area$ is the final circuit area, $\sqrt{Area}$ is used as the average gating control signal wire length, C_w^{min} the unit-length capacitance for a minimum width metal-1 wire, N_{sw} the total number of switchings on the gating control signal wires, and the scaling factor 2 is to account for the fact that control signal wires are not usually routed as metal-1 or at minimum width. N_{sw} can be estimated using the profiling and scheduling information. All the overheads are included in the results we report later.

D. Different output network topologies

Three output network topologies were introduced in Section III. If all the output data are sent to all the receivers without regard to whether they need the data or not, the MST output network will consume the least power and the fully dedicated the most. However, if interconnect SSA is suppressed using the proposed techniques, this would not be the case. Fig. 6

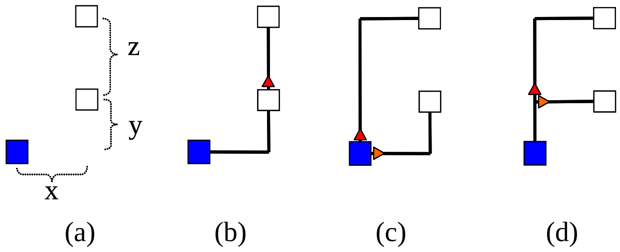

Fig. 6. Different output network topologies for (a) one DPU sending data to two other DPUs: (b) MST, (c) fully dedicated, and (d) trunk-branches.

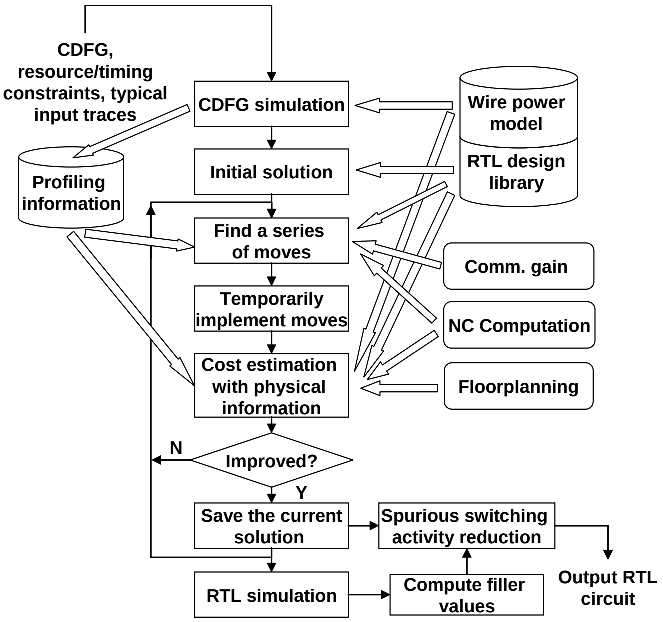

Fig. 7. The framework of the interconnect-aware high-level synthesis tool for low power.

shows a DPU, colored black, sending data to two other DPUs, colored white, using different output network topologies. It also shows the optimal locations for gating signals with SSA.

Suppose tri-state buffers are used so that the last useful value is remembered in the gated cycles. Let α_1 denote the unit-length switched capacitance for data needed by the upper receiver, α_2 denote that for the lower receiver, and α for all the outputs of the sender. Let SC_{MST} denote the total wire switched capacitance for the MST output network, SC_{FD} for fully dedicated and SC_{TB} for trunk-branches. It can be proved that when $(\alpha_1 + \alpha_2) > \alpha$, $SC_{FD} > SC_{TB} > SC_{MST}$ and when $(\alpha_1 + \alpha_2) < \alpha$, $SC_{FD} < SC_{TB} < SC_{MST}$. Since $(\alpha_1 + \alpha_2)$ can be either larger or smaller than α, none of these topologies is always better than the others in terms of power consumption. In this work, the trunk-branches topology is assumed due to its never-the-worst power consumption and the algorithmic simplicity for constructing it.

VI. Interconnect-aware high-level synthesis for low power

We have implemented our proposed techniques on top of the low power high-level synthesis tool, SCALP [10]. They constitute about 4,000 lines of additional C++ code over about 20,000 lines in SCALP. An overview of the new system is shown in Fig. 7.

First, the CDFG is simulated with typical input traces in order to profile each operation and data transfer edge. The profiling information combined with the RTL design library (with its associated power macro-models) is used to evaluate the RTL circuit in terms of power, area and performance. The initial solution consists of a fully parallel implementation in which each CDFG node is bound separately to the fastest functional unit in the library that can implement it, and every CDFG edge is bound to a separate register. The initial schedule is as-soon-as-possible. Then the iterative improvement engine is used to optimize the RTL architecture for different objectives (*e.g.*, area, power). The different moves used for this purpose consist of functional unit selection, resource sharing and resource splitting. Functional unit selection involves choosing an appropriate functional unit for a given CDFG node among the many available, *e.g.*, choosing a carry-lookahead adder vs. a ripple-carry adder. Resource sharing involves merging of functional units or registers and resource splitting the reverse. Functional unit selection and resource sharing may need rescheduling. Resource splitting may adversely affect the targeted objective. However, it allows the algorithm to escape local minima. Physical and interconnect information is used for cost gain computation and move identification using techniques proposed in Section IV. Neighborhood crowd checking and communication cost gain presented earlier influence this step. Initially, a series of moves is identified based on their cost gain without floorplan information, and temporarily implemented. Then this temporary solution is floorplanned to determine the cost of the solution for the targeted objective. If there is a cost reduction, the temporary solution is accepted as the new starting point for the next iteration (note that individual moves in the series can have a negative impact on the objective; however, the whole series should not). Otherwise, it is rejected and a new iteration uses those series of moves which have not yet been temporarily implemented. The algorithm terminates if there is no improvement or a pre-specified maximal number of iterations has been reached. Then the best RTL architecture solution seen so far is provided as the output. This architecture is post-processed using the SSA reduction techniques presented earlier to obtain the final RTL architecture. The post-processing includes control signal generation, gating logic insertion and controller re-specification [24].

VII. Experimental Results

In this section, we present experimental results for the proposed techniques which are implemented as described in Section VI. As mentioned before, the wire capacitances are taken from [13] which are themselves based on the information in [44]. The circuits were synthesized using an RTL design library based on NEC 0.18μm technology [29], [30]. SCALP, the high-level synthesis tool, on top of which we implemented our techniques, already has the capability to do optimization for area and power, however, in an interconnect-unaware fashion. Such SCALP-generated circuits are used for comparisons. All the circuits are given the same performance constraints. The experiments were conducted on a 1.3 GHz Pentium IV based PC with 128 MB memory. The CPU times for high-level synthesis vary from less than one second for *Diffeq* to 207 seconds for *Jacobi*. The CPU times for CDFG and RTL simulation are proportional to the size of the input traces. In our experiments, they are comparable to CPU times of high-level synthesis.

A. Interconnect-aware high-level synthesis

In Fig. 8, power reduction obtained as a result of incorporating interconnect-awareness into high-level synthesis for low power is shown. Interconnect-awareness is achieved by taking RTL interconnect power into account and using the two proposed metrics (see Section IV) to guide iterative improvement. Note that no signal gating is performed in this case. Compared to interconnect-unaware power-optimized (IUPO) circuits, interconnect-aware power-optimized (IAPO) circuits consume 22.3% less power on an average while incurring only 0.5% area overhead. Compared to area-optimized (AO) circuits, power is reduced by an average of 53.3% at an average

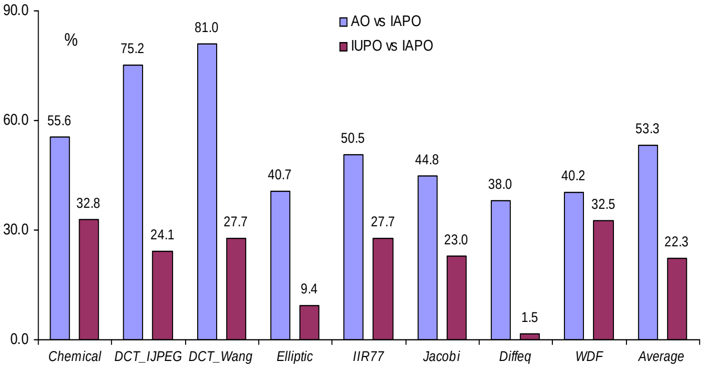

Fig. 8. Power reduction due to interconnect-awareness.

area overhead of 44.3%.

B. Gating signals with SSA

Table 1 shows the average area and power overheads (A_{oh} and P_{oh}) for the extra gating circuitry as a percentage of the respective area and power of the AO, IUPO and IAPO circuits. The corresponding gated circuits are called AO-gated, IUPO-gated and IAPO-gated, respectively. Such small overheads are almost negligible compared to the large power saving achieved. The table also shows the average power reduction in interconnects (P_i), DPUs (P_d) and the total circuit (P_{tot}) when signal gating (see Section V) is applied to AO, IUPO and IAPO circuits (thus generating AO-gated, IUPO-gated and IAPO-gated circuits, respectively). Even for IAPO circuits, the net total power reduction is 11.1% (the power overhead is included in this number) at a negligible area overhead. For AO and IUPO circuits, the power reductions are more due to the fact they have more SSA in both interconnects and DPUs. It is worth noting that due to the difficulty in estimating glitches at the RTL, glitching power is not taken into consideration. If it could be estimated, we expect the power reduction to be much more.

Table 1
Gating signals with SSA.

	A_{oh} %	P_{oh} %	P_i %	P_d %	P_{tot} %
AO-gated	0.1	2.6	49.4	11.8	22.6
IUPO-gated	0.04	3.5	23.4	9.5	17.5
IAPO-gated	0.03	4.8	15.7	4.8	11.1

C. Interconnect power reduction

Fig. 9 presents interconnect power reduction of IAPO-gated circuits compared to AO, IUPO and IAPO circuits. The average reductions are 72.9%, 53.1% and 15.7%, respectively. This shows that our interconnect power reduction techniques are quite effective. Since our methodology is to trade some DPU power savings for more interconnect power savings, when interconnects consume a higher fraction of total circuit power in future technologies, our methodology is likely to yield a higher total circuit power reduction.

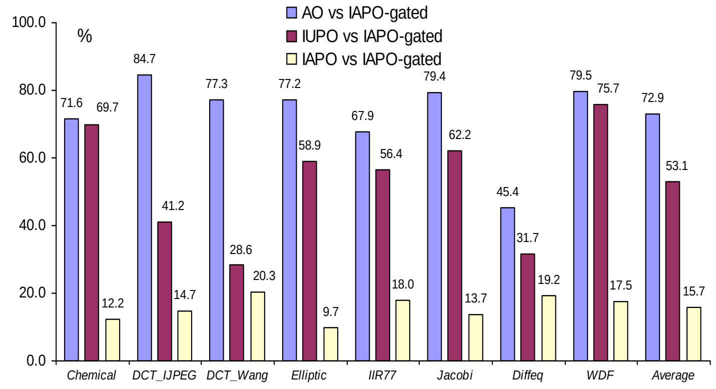

Fig. 9. Interconnect power reduction.

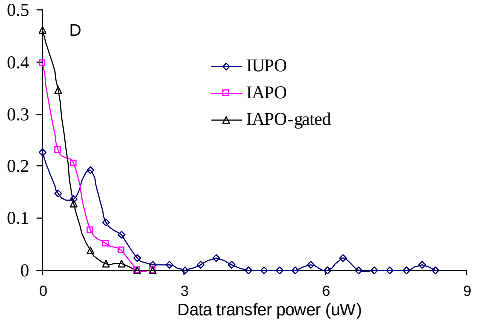

Fig. 10. Data transfer power distribution for *Elliptic* under different scenarios.

To better illustrate the effectiveness of the proposed techniques, Fig. 10 shows the distribution of individual inter-DPU data transfer power for IUPO, IAPO and IAPO-gated implementations of the *Elliptic* benchmark. The y co-ordinate specifies the fraction of all data transfers that consumed power given by the x co-ordinate. It demonstrates that interconnect-aware high-level synthesis and signal gating drastically reduce the number of high-power data transfers. It should be noted that the IUPO implementation of *Elliptic* has 88 inter-DPU data transfers while IAPO and IAPO-gated implementations only have 78.

D. Total power reduction

Fig. 11 shows the percentage power reduction and area overhead when all the proposed techniques are applied (*i.e.*, for IAPO-gated circuits) compared to IUPO and AO circuits. The area overhead for some benchmarks is negative, which means the proposed techniques also reduced the circuit area. For all the benchmarks, an average 26.8% total power reduction is achieved compared to IUPO circuits with 0.5% area overhead. Compared to AO circuits, the average power reduction is 56.0% with 44.4% average area overhead.

VIII. Conclusions

Interconnects for data transfers consume a significant fraction of overall power. We showed that high-level synthesis has a significant impact on the interconnect power consumption. We presented a comprehensive study of interconnect-aware high-level synthesis for low power. Rent's rule is typically used for estimation of interconnect power consumption. However, it is based on assumptions which may not be justified in high-level synthesis. A method for estimating data transfer power consumption at the RTL was, therefore, proposed. Using this method, RTL interconnect power is taken into consideration during high-level synthesis for low power. Since binding has a great impact on the interconnect topology and switching activity, we adopted two metrics to guide binding for low power. Moreover, we presented extended signal gating techniques to suppress SSA in the interconnects as well as DPUs. Experimental results demonstrate the benefits of incorporating interconnect-awareness into high-level synthesis for low power and the effectiveness of the proposed techniques. Our techniques are general and can also be easily incorporated

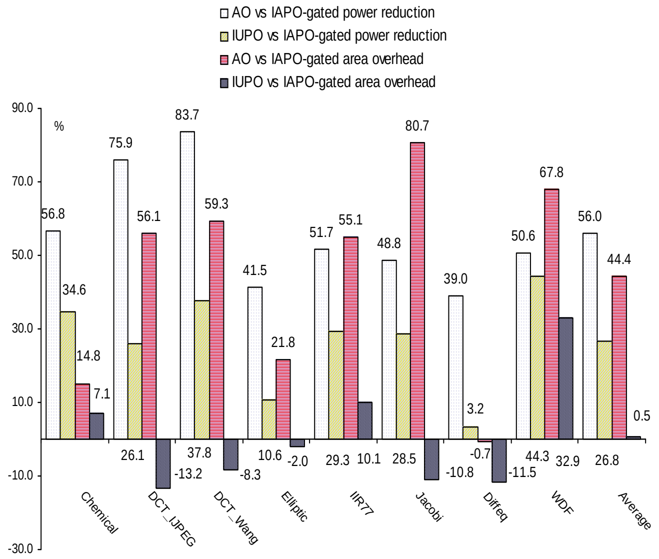

Fig. 11. Total power reduction and area overhead

into other high-level synthesis systems. Although the benchmarks used in this paper were data-dominated, constrained by the fact that the underlying tool, SCALP, is only applicable to data-dominated behaviors, our techniques are equally applicable to control-flow intensive behaviors.

Acknowledgments

The authors would like to thank Robert P. Dick for the floorplanner used in this work and Anand Raghunathan for the NEC ASIC RTL library. This work was supported by DARPA under contract no. DAAB07-00-C-L516.

References

[1] A. Raghunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*, Kluwer Academic Publisher, Norwell, MA, 1998.

[2] R. Mehra, L. M. Guerra, and J. M. Rabaey, "Low-power architectural synthesis and the impact of exploiting locality," *J. VLSI Signal Processing*, vol. 13, no. 8, pp. 877–88, Aug. 1996.

[3] L. Goodby, A. Orailoglu, and P. M. Chau, "Microarchitecture synthesis of performance-constrained, low power VLSI designs," in *Proc. Int. Conf. Computer Design*, Oct. 1994, pp. 323–326.

[4] A. Dasgupta and R. Karri, "Simultaneous scheduling and binding for power minimization during microarchitecture synthesis," in *Proc. Int. Symp. Low Power Design*, Apr. 1994, pp. 255–270.

[5] J. M. Chang and M. Pedram, "Register allocation and binding for low power," in *Proc. Design Automation Conf.*, June 1995, pp. 29–35.

[6] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-driven behavioral synthesis for low power VLSI systems," *IEEE Design & Test Comput.*, pp. 70–84, Sept. 1995.

[7] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. Brodersen, "Optimizing power using transformations," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 12–51, Jan. 1995.

[8] R. S. Martin and J. P. Knight, "Power profiler: Optimizing ASICs power consumption at the behavioral level," in *Proc. Design Automation Conf.*, June 1995, pp. 42–47.

[9] M. Johnson and K. Roy, "Optimal selection of supply voltages and level conversion during datapath scheduling under resource constraints," in *Proc. Int. Conf. Computer Design*, Oct. 1996, pp. 72–77.

[10] A. Raghunathan and N. K. Jha, "SCALP: An iterative-improvement-based low power data path synthesis system," *IEEE Trans. Computer-Aided Design*, vol. 16, no. 11, pp. 1260–1277, Nov. 1997.

[11] K. S. Khouri, G. Lakshminarayana, and N. K. Jha, "High-level synthesis of low power control-flow intensive circuits," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 12, pp. 1715–1729, Dec. 1999.

[12] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE J. Solid-State Circuits*, vol. 29, no. 6, pp. 663–670, June 1994.

[13] J. Cong and Z. Pan, "Interconnect performance estimation models for design planning," *IEEE Trans. Computer-Aided Design*, vol. 20, no. 6, pp. 739–752, June 2001.

[14] S. Hong and T. Kim, "Bus optimization for low-power data path synthesis based on network flow method," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2000, pp. 312–317.

[15] D. W. Knapp, "Fasolt: A program for feedback-driven data-path optimization," *IEEE Trans. Computer-Aided Design*, vol. 11, no. 6, pp. 677–695, June 1992.

[16] J.-P. Weng and A. C. Parker, "3D scheduling: High-level synthesis with floorplanning," in *Proc. Design Automation Conf.*, June 1992, pp. 668–673.

[17] Y.-M. Fang and D. F. Wong, "Simultaneous functional-unit binding and floorplanning," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 317–321.

[18] V. G. Moshnyaga and K. Tamaru, "A floorplan based methodology for data-path synthesis of sub-micron ASICs," *IEICE Trans. Inf. & Syst.*, vol. E79-D, no. 10, 1996.

[19] M. Xu and F. J. Kurdahi, "Layout-driven RTL binding techniques for high-level synthesis using accurate estimators," *ACM Trans. Design Automation Electronic Systems*, vol. 2, no. 4, pp. 312–343, Oct. 1997.

[20] P. Prabhakaran and P. Banerjee, "Simultaneous scheduling, binding and floorplanning in high-level synthesis," in *Proc. Int. Conf. VLSI Design*, Jan. 1998, pp. 428–434.

[21] K. Choi and S. P. Levitan, "A flexible datapath allocation method for architectural synthesis," *ACM Trans. Design Automation Electronic Systems*, vol. 4, no. 4, pp. 376–404, Oct. 1999.

[22] W. E. Dougherty and D. E. Thomas, "Unifying behavioral synthesis and physical design," in *Proc. Design Automation Conf.*, June 2000, pp. 756–761.

[23] E. Mussoll and J. Cortadella, "High-level synthesis techniques for reducing the activity of functional units," in *Proc. Int. Symp. Low Power Design*, Apr. 1995, pp. 99–104.

[24] S. Dey, A. Raghunathan, N. K. Jha, and K. Wakabayashi, "Controller-based power management for control-flow intensive designs," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 10, pp. 1496–1508, Oct. 1999.

[25] Independent JPEG Group, *http://www.ijg.org*.

[26] K. R. Rao and P. Yip, *Discrete Cosine Transform*, Academic Press, 1990.

[27] *http://www.cbl.ncsu.edu/benchmarks/*.

[28] S.-Y. Kung, *VLSI Array Processors*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[29] NEC cell-based ASIC CB-11, *http://www.necel.com/ASIC/*, 2000.

[30] A. Raghunathan, *Personal communication.*

[31] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*, Prentice Hall, Englewood Cliffs, NJ, 1996.

[32] N. Sherwani, *Algorithms for VLSI Physical Design Automation: Second Edition*, Kluwer Academic Publishers, Norwell, MA, 1995.

[33] T. Uchino and J. Cong, "An interconnect energy model considering coupling effects," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 7, pp. 763–776, July 2002.

[34] C. N. Taylor, S. Dey, and Y. Zhao, "Modeling and minimization of interconnect energy dissipation in nanometer technologies," in *Proc. Design Automation Conf.*, June 2001, pp. 754–757.

[35] P. P. Sotitriadis and A. Chandrakasan, "A bus energy model for deep sub-micron technology," *to appear in IEEE Trans. VLSI Systems.*

[36] P. Heydari and M. Pedram, "Interconnect energy dissipation modeling in high-speed ULSI circuits," in *Proc. Asia and South Pacific Design Automation Conference*, Jan. 2002, pp. 132–137.

[37] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partition," in *Proc. Design Automation Conf.*, June 1982, pp. 173–181.

[38] P. Kapur, G. Chandra, and K. C. Saraswat, "Power estimation in global interconnects and its reduction using a novel repeater optimization methodology," in *Proc. Design Automation Conf.*, June 2002, pp. 461–466.

[39] B. S. Cherkauer and E. G. Friedman, "A unified design methodology for CMOS tapered buffers," *IEEE Trans. VLSI Systems*, vol. 3, no. 1, pp. 99–111, Mar. 1995.

[40] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, Reading, MA, 1990.

[41] V. P. Roychowdhury, S. K. Rao, L. Thiele, and T. Kailath, "On the localization of algorithms for VLSI processor arrays," in *Proc. VLSI Signal Processing III*, 1988, pp. 459–470.

[42] M. Munch, B. Wurth, R. Mehra, J. Sproch, and N. Wehn, "Automating RT-level operand isolation to minimize power consumption in datapaths," in *Proc. Design, Automation & Testing in Europe*, Mar. 2000, pp. 624–631.

[43] C.-T. Hsieh and M. Pedram, "Architectural energy optimization by bus splitting," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 4, pp. 408–414, Apr. 2002.

[44] *National Technology Roadmap for Semiconductors*, Semiconductor Industry Association, 1997.