Interconnect-aware Low Power High-level Synthesis

Lin Zhong *Student Member, IEEE,*and Niraj K. Jha *Fellow, IEEE,*

***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. It not only reduces datapath unit power consumption in the resultant register-transfer level (RTL) architecture, but also optimizes interconnects for power. We take into account physical design information and coupling capacitance to estimate interconnect power consumption accurately for deep sub-micron (DSM) technologies. We show that there is significant spurious ($i.e.$, unnecessary) switching activity in the interconnects and propose techniques to reduce it. Compared with interconnect-unaware power-optimized circuits, interconnect power can be reduced by 53.1% on an average, while overall power is reduced by an average of 26.8% with negligible area overhead. Compared with area-optimized circuits, the interconnect power reduction is 72.9% and overall power reduction is 56.0% with 44.4% area overhead. The power reductions are obtained solely through switched capacitance reduction (no voltage scaling is assumed).**

***Index Terms*— High-level synthesis, Interconnect, Low power.**

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 tried to minimize power consumed by datapath units while ignoring that by interconnects. As shown in [12], interconnects consume a significant fraction of total circuit power. Moreover, 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 datapath units [14]. It has a significant impact on the switching activity and topology of the interconnects in the resultant design. Optimizing interconnects during high-level synthesis is therefore important.

Interconnect cost has been considered in high-level synthesis by many researchers [15], [16], [17], [18], [19], [20], [21]. In most cases, the number of interconnects/multiplexers is used as the interconnect cost. None of these works use physical information to estimate interconnect cost accurately. Moreover, all these works are targeted at only bus-based architectures and only area/performance optimization.

For bus-based RTL architectures, bus coding has been proposed to reduce switching activity [22], [23], [24]. The work in [25] proposes to optimize bus power by appropriately binding data transfers to busses. These techniques, however, do not take wire length into consideration, and are not applicable to multiplexer-based interconnects.

Taking physical level information into account during high-level synthesis has also attracted a lot of attention [26], [27], [28], [29], [30], [31], [32], [33], [34], [35]. Earlier work used floorplanning information in high-level synthesis to improve design area or performance estimation. Only some recent works took interconnect power consumption into consideration [2], [33]. In [33], the switching activity on CDFG edges was profiled. It was used with the floorplanner to optimize the power consumption of inter-module data transfers. Only the floorplanner was made interconnect power-aware. Moreover, the coupling effect between wires was not taken into consideration. In [2], the authors preserved the locality and regularity in input behaviors to optimize bus power consumption in bus-based architectures. However, they did not distinguish between busses with different switching activities. Due to the mapping of behavioral identities (operations and variables) to physical identities (functional units and registers), locality and regularity in the behavior do not necessarily guarantee locality in the physical design.

In this work, we provide a comprehensive treatment of interconnect-aware high-level synthesis for low power, targeting multiplexer-based interconnects. We evaluate the power consumption in the steering logic and clock distribution network as well as data transfer wires, using early floorplanning information. Since coupling capacitances are expected to dominate the wire capacitance in future technologies, we take coupling into account while estimating wire power consumption. Moreover, we propose metrics to guide the binding process to preserve and create locality in the physical implementation. Unlike [2], our methodology does not assume behaviorial locality and regularity. While concentrating on the interconnect power issue, our synthesis flow and methodology maximize the orthogonality to other problems such as scheduling, floorplanning, wire and RTL power modeling, $etc.$

It is well-known that there can be significant spurious switching activity (SSA), $i.e.$, activity not required by the behavioral specification, in datapath units [36], [37], [38], [39]. We found that there is significant SSA in interconnects too. It increases the power consumption in the interconnects as well as other circuit components. We propose techniques to suppress it.

The paper is organized as follows. We offer our observations and motivational examples in Section II, and discuss RTL interconnect power estimation in Section III. We present methodologies for interconnect-aware binding in Section IV, and address the problem of interconnect SSA in Section V. After giving the framework of our interconnect-aware high-

Manuscript received September 2, 2002; revised April 9, 2003 and November 18, 2003. This work was supported by the DARPA under contract no. DAAB07-00-C-L516.

The authors are with the Department of Electrical Engineering, Princeton University, NJ 08544.

level synthesis system in Section VI, we provide 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 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 comprehensive low power high-level synthesis tool, called SCALP [10], to obtain RTL circuits and power consumption of controllers/datapaths, aided by our interconnect power estimation methods detailed in Section III. SCALP is interconnect-unaware. However, it tackles all the problems in high-level synthesis including transformations, scheduling, clock selection, module selection, binding, etc.

1) Interconnect power consumption: We evaluated eight high-level synthesis benchmarks implemented in 0.18μm technology. *Chemical* and *IIR77* are infinite impulse response filters used in the industry. *DCT_IJPEG* is the Independent JPEG Group's implementation of digital cosine transform (DCT) [40]. *DCT_Wang* is the DCT algorithm named after the inventor [41]. 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 [42]. *Jacobi* is the Jacobi iterative algorithm for solving a fourth-order linear system [43]. *WDF* is a finite impulse response wave digital filter. Table I gives the number of basic operations in the benchmark behaviors. It also indicates whether there are loops inside the behavior. The smallest benchmark is *Diffeq* with six multiplications, two additions and two subtractions. The largest is *Jacobi* with 24 multiplications, eight divisions, eight additions and 16 subtractions.

Using wire and gate capacitances data from [44], Fig. 1 shows the input capacitance of a minimum-width gate along with the capacitance (as the sum of area, fringing and coupling capacitance) of a minimum-width metal-1 wire of length 20λ (where λ is the minimum feature size) for different DSM technologies. The figure shows that the two are comparable for current and future technologies. In a circuit, a data transfer wire between datapath units can be hundreds of λ long, which is equivalent to tens of gates in terms of input capacitance. Its power consumption is thus non-negligible.

We used the wire capacitances from the same source along with an RTL design library from NEC [45], [46] to estimate power consumption for the benchmarks. The NEC RTL library was characterized for power and area through logic-level power and area estimation, which was verified to be within 10% of SPICE simulation. Fig. 2(a) shows the percentage of total power consumed by RTL interconnects for area-optimized (AO) and interconnect-unaware power-optimized (IUPO) benchmarks, respectively.

In both AO and IUPO circuits, interconnects consume a significant percentage of total power, with the average percentage being 25.9% and 25.0%, respectively. It is worth noting that if the coupling effect is not considered, the interconnect power percentage will drop to 16.2% and 14.3%, respectively.

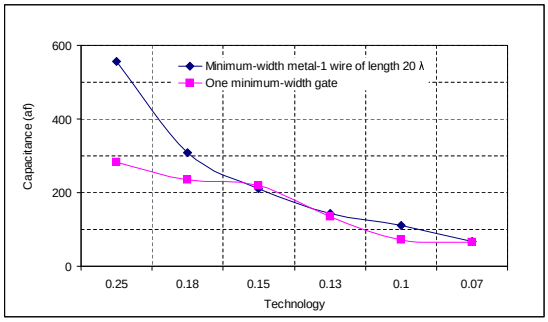


Fig. 1. Wire vs. gate capacitance.

2) Spurious switching activity (SSA): Let us define the RTL interconnects connected to the output of a datapath unit 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, the output value has to be routed to only a small portion of the output network in a given clock cycle. If care is not taken, many other parts of the network may also experience switching activity. Arithmetic functional units typically output a lot of glitches before the output finally settles down to the correct value [47]. These glitches also contribute significantly to the output network power consumption.

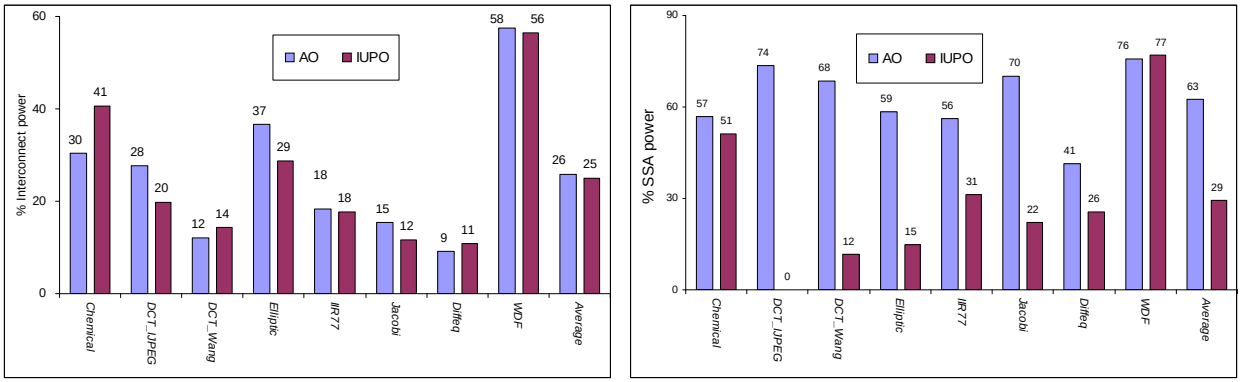
Fig. 2(b) shows the percentage of total interconnect power consumed through SSA for the eight benchmarks optimized for area and power. For both AO and IUPO circuits, SSA consumes a significant percentage of total interconnect power, with the average being 62.6% and 29.2%, respectively, even when glitches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

B. Motivational example

In this section, we present an example to motivate the techniques to be presented. Fig. 3(a) shows the CDFG for *Diffeq* and one of its possible schedules (the numbers on the right indicate clock cycles). 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. Meanwhile, 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. It is obvious that functional unit and register binding has significant impact on the interconnect structure. Nevertheless, interconnect power optimization through judicious binding has not been adequately explored before.

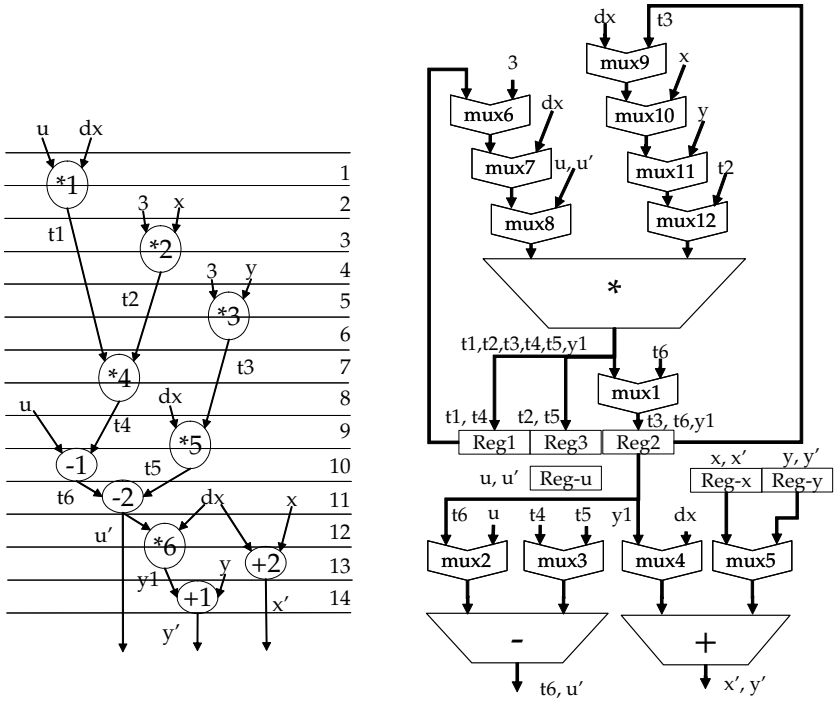
TABLE I
INFORMATION ON BENCHMARK BEHAVIORS.

	Chemical	*DCT_IJPEG*	*DCT_Wang*	*Diffeq*	*Elliptic*	*IIR77*	*Jacobi*	*WDF*
No. of ×	17	12	21	6	0	22	24	8
No. of ÷	0	0	0	0	0	0	8	0
No. of +	9	24	14	2	26	14	8	26
No. of −	6	12	12	2	0	0	16	0
Loop(Y/N)	Y	N	N	Y	Y	Y	N	Y



(a) (b)

Fig. 2. Observations: (a) percentage of total power consumed by interconnects, and (b) percentage of total interconnect power consumed by spurious switching activity.



(a) (b)

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. Each of these registers, however, only needs a subset of these variables. Such data broadcasting introduces SSA not only in data transfer wires but also in the steering logic along these wires, such as multiplexer mux1. The impact of SSA

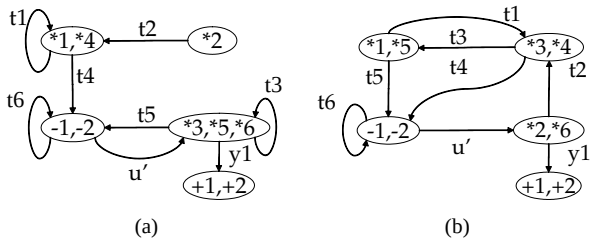


Fig. 4. Different bindings for the multiplication operations.

in interconnects is not limited to themselves. For example, the multiplier, adder, and subtracter only need variable t3, y1 and t6 from Reg2, respectively, while 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. Even worse, 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). Therefore, if care is not taken, the SSA may propagate into the adder and the subtracter. By eliminating SSA in the output networks, we can therefore suppress a significant portion of the SSA in the functional units as well. In Section V, we offer techniques to suppress interconnect SSA.

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, and controllers, $etc.$ RTL interconnects consist of data transfer wires, buffers on the wires, clock distribution network and steering logic. Power, delay and area of interconnects within DPUs are assumed to be included in the corresponding models for the DPUs provided in the RTL design library.

A. Data transfer wires

Several factors decide how much power is consumed by wires transferring data between DPUs: the switching activity due to the data transferred, wire layout, and capacitive parameters. We obtain switching activity through simulation using typical input traces. Then, we use a global wire power model and a local wire power model to get wire power consumption. The global model estimates how the wire is routed, its length, and how it is assigned to different metal layers if multiple metal layers are used. The local model estimates the unit-length switched capacitance (the equivalent capacitance due to switching activity in a unit-length wire) based on the switching activity and specific metal layer wire capacitive parameters. These two models together estimate the power consumed by the entire data transfer wire based on the relative positions of the source and destination, which are obtained from an RTL flooplanner. The data transfer wire power estimation flow is shown in Fig. 5. We next present the local and global power models.

1) Local power model: In DSM technologies, the switched capacitance of a wire depends on its neighbors' switching activity, which determines how the coupling capacitances between wires are charged and discharged [44]. Several interconnect energy models have been proposed to take the coupling effect into consideration [48], [49], [50], [51]. Given the wire capacitive parameters, these models generate the unit-length switched capacitance or energy based on the switching patterns on the wire and its neighbors. We use the table look-up method proposed in [49], which is shown in Fig. 5 as the local power model. Other models, such as the one proposed in [50], can be readily incorporated into the synthesis flow used in our work. According to [49], the unit-length switched capacitance of a wire can be found by observing the switching patterns on the wire and its two immediate neighbors. These are called *three-line switching patterns*, as shown in Fig. 5. There are 10 different three-line switching patterns possible. Using this method, PSPICE and capacitance data from [44], we estimate the unit-length switched capacitance of the center line for different three-line switching patterns. These data are stored in a table, called the *pattern-power table*, for later use. If multiple metal layers are used, wires on different layers will have different capacitive parameters. For each layer, we need to generate a different pattern-power table. Together with the length of a wire, its layer assignment and three-line switching patterns, these tables can be used to estimate the power consumed by the entire wire.

2) Global power model: A DPU's output network consists of the RTL interconnect components connecting its output to the inputs of other DPUs. A data transfer between two DPUs is assumed to be center-to-center and rectilinear for simplicity. In [52], it is shown that more than 95% of wires in an ASIC are routed in a Manhattan fashion, $i.e.$, without any detour from the shortest route. We therefore assume that all data transfers are routed in a Manhattan fashion. A bus-based architecture shares the output networks among DPUs through busses. Since sharing output networks imposes a high performance and power penalty [47], we assume that output networks are not shared in the multiplexer-based architecture we use. Nevertheless, it is still possible to share wires within one output network. Fig. 6(a) shows one DPU sending 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 imply minimization of the total wire length in the output network. Therefore, a minimum spanning tree (MST) output network in the Steiner tree style [53] 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 imply minimal total power consumption because it is hard to distinguish between dedicated and shared interconnects in the case of an MST. We introduce a *trunk-branches* style Steiner tree for the output network, as shown in Figs. 6(b) and 6(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

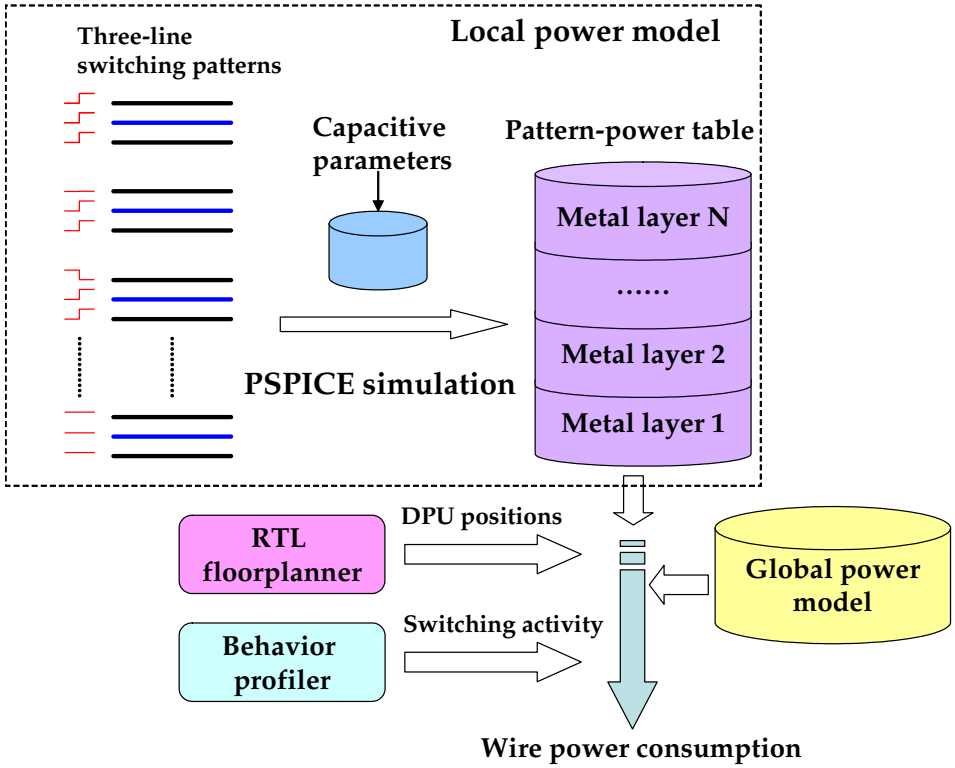


Fig. 5. Local and global wire models, behavior profiler, and RTL floorplanner used to estimate data transfer wire power consumption.

consumption. In Section V, we will compare three output network styles, fully dedicated, optimal total length shared and trunk-branches, in detail. We also assume all metal layers have the same capacitive parameters as metal layer one. This assumption slightly underestimates interconnect power, but obviates the need for layer assignment, and thus accelerates power estimation.

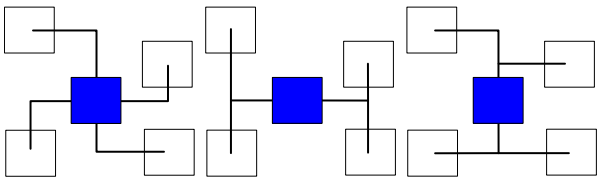


Fig. 6. Different topologies of an output network: (a) fully dedicated, and trunk-branches output network with the trunk being (b) horizontal, and (c) vertical.

3) RTL data transfer power estimation: Since we use the three-line local power model, we simulate the CDFG with its typical input traces to get all the three-line switching patterns for any two CDFG edges when their data transfers occur consecutively (note that each CDFG edge corresponds to one data transfer, or, multi-bit wire in the RTL implementation). With information from the pattern-power tables, we obtain the unit-length switched capacitance for transitions from one variable to another through profiling. This needs to be done only once for each behavior. Some of the unit-length switched capacitances for the *Diffeq* benchmark are shown in Table II. It shows the unit-length switched capacitance when a row

TABLE II
PROFILING RESULTS FOR $Diffeq$.

	$t1$	$t2$	$t3$	$t4$	$t5$	$t6$
$t1$	0.24	0.27	0.25	0.23	0.23	0.27
$t2$	0.25	0.25	0.28	0.25	0.27	0.25
$t3$	0.23	0.30	0.29	0.25	0.27	0.22
$t4$	0.25	0.28	0.29	0.24	0.25	0.24
$t5$	0.25	0.31	0.30	0.25	0.24	0.23
$t6$	0.25	0.22	0.20	0.21	0.22	0.28

variable is transmitted just after a column variable during multiple executions of the *Diffeq* behavior. It uses metal layer one pattern-power table (we call these *basic unit-length switched capacitances*). When two variables are output by a DPU, they have to be transferred through the same output network. The table illustrates how CDFG binding would affect the data transfer energy consumption.

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 get each DPU's position in the floorplan. The sum of basic unit-length switched capacitances for data transfers between two units is used as their communication cost during floorplanning. We partition a datapath into a binary tree hierarchy which has balanced area and minimal communication cost among resultant partitions. The algorithm is an extended version of the one given in [54]. 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, the length of an output network is estimated using the global power model. This information is used along with pattern-power tables to estimate the power consumed by the entire wire, as shown in Fig. 5.

Binding of CDFG nodes to DPUs significantly impacts RTL interconnect power consumption. The wire switching activity depends on the switching activity of the outputs of DPUs they are connected to. The latter depends on binding based on a given schedule. Binding also impacts the topology of the output network, which affects wire length through floorplanning and routing.

4) Rent's rule: Rent's rule [55], [56] states that

$$T = AK^P$$

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. In [57], it is used to estimate the number of interconnects. The total interconnect power dissipation is estimated based on that number, average interconnect length and average fanout. In [58], the authors obtain a stochastic distribution of wire length with the assumption that the same p and A hold for the chip as well as its different parts. They use the distribution to estimate wire power dissipation in [59]. The assumptions made about p and A, e.g., their values and whether they are constant, render the methodology barely useful for design space exploration. In high-level synthesis, different binding and scheduling algorithms may generate circuits with different p and A, as illustrated in Fig. 4.

B. 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]. The power macro-models play a similar role that pattern-power tables play for wire power estimation. When inputs to a multiplexer change, the power macro-model records the energy consumed for the corresponding input switching pattern. By profiling the input trace, we can get the total power consumption by simply counting all the different input switching patterns. 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.

C. Clock distribution network

Many DPUs are clocked. Such units include controllers, 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 switched capacitance and the total length of the MST.

D. Buffers

There are two kinds of buffers related to RTL interconnects: the output buffer driving the output network for each DPU and the buffers (a.k.a. repeaters) inserted along a long wire to reduce its delay. One can also view the latter as a distributed output buffer. Buffers increase interconnect power consumption significantly. Studies show that the total switched capacitance of inserted buffers (repeaters) is about 1.1 times the total switched capacitance of the corresponding wire [60] when buffer insertion is delay-optimized. For output buffers, it has been shown that their total switched capacitance is also approximately 1.1 times that of the corresponding wire [61] (see Appendix for details). Therefore, we do not distinguish between these two kinds of buffers and treat them as an integral part of the wire in the following sections.

IV. Interconnect-aware binding

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 in order to accelerate and improve design space exploration. The overall high-level synthesis flow is presented in Section VI.

A. Neighborhood-sensitive binding

For a CDFG node (an operation in the behavior), we define its *behavioral neighbors* to be the other CDFG nodes which have data communication with it in the CDFG. After high-level synthesis, CDFG nodes and edges (variables) are mapped to RTL DPUs like functional units and registers through the binding process. A DPU's *RTL neighbors* are defined as the other DPUs that have data communication with it. After DPUs are floorplanned, we define a DPU's *physical neighbors* as the DPUs adjacent to it in the floorplan. The data communication cost will be reduced if DPUs, which exchange data, are placed close to each other in the floorplan, i.e., RTL neighbors are made physical neighbors. A data transfer is called $local$ if it happens between two neighboring operations or DPUs. It is obvious that behavioral locality, RTL locality and physical locality are different.

To reduce the communication cost, some researchers have proposed techniques to localize data transfers at different levels [2], [62]. In [62], an algorithmic transformation is proposed to localize data transfers in VLSI array processors. In [2], behavioral partitioning is advocated for exploiting behavioral locality and ensuring RTL locality after binding. These methods are only effective for highly regular signal processing behaviors. Besides, they do not distinguish between RTL 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.

To localize data transfers at the physical level, we should ensure that as many RTL neighbors as possible are also physical neighbors, especially those that conduct high unit-length switched capacitance data exchange with each other.

On the other hand, the physical neighborhood capacity of a DPU is limited, and is related to its geometry. We define the *neighborhood crowd* of a DPU, NC, as,

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

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, various moves are defined. Two binding moves that affect NC are DPU sharing and splitting (See Section VI for details). 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.

There are two relevant parameters in connection with the use of NC. First, the threshold for NC above which a move is rejected. When the NC for a DPU is above this threshold, the DPU is said to be overcrowded. We intuitively set the threshold to 4.0 since a DPU can have four physical neighbors of its own size. We observed that very few moves are rejected because of overcrowding of a DPU during iterative improvement. Second, when we combine the NC gain and power gain to evaluate a move, there is a problem of scale. We estimate the power impact of a unit NC increase as that by a data transfer with typical (statistical mean) unit-length switched capacitance and 1.5 times the square root of a typical DPU area.

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 switched capacitance 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

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

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 switched capacitance for data transfers from A to B. $P_{sw}(B, A)$ is similarly defined. A negative communication gain is defined for DPU splitting moves in the same way.

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], [36], [63]. The same idea can be used for interconnects as well. The difference is shown in Fig. 7. Traditional signal gating mechanisms are placed before the DPU that receives the signal ($i.e.$, at the receiver) as shown in Fig. 7(a), while our approach gates the signal right after the source ($i.e.$, at the sender) as shown in Fig. 7(b).

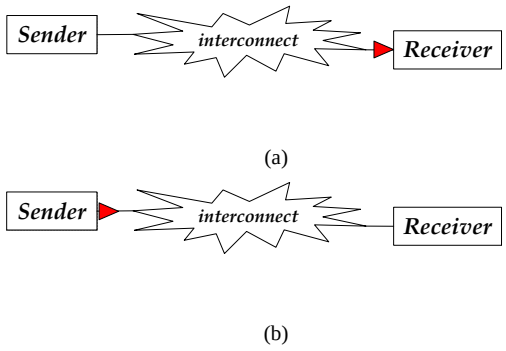


Fig. 7. Signal gating (a) before the DPU and (b) before the interconnect.

The gating mechanism decides whether data can propagate through the gating logic or not. Interconnect-aware power management entails gating a signal containing SSA before the SSA propagates to the output network. When the controller is properly respecified [37], 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 a 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 in [36]. However, often the power savings obtained may not justify the large overhead introduced by multi-bit latches. Another option may be to set a input to tri-state when it is not active. Unfortunately, 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 large power consumption in downstream DPUs. Tri-state gating is justified only in very limited cases, which will be addressed later. In [63], 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 f*.

For the RTL circuit in Fig. 3(b), Fig. 8 gives the names of the variables whose values are propagated from the multiplier to register Reg2 after gating in one iteration of the CDFG,

Cycle #	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Before Gating	X	X	t1	X	t2	X	t3	X	t4	X	t5	X	X	y1
Gated: ideal case	y1'	-	-	-	-	-	-	t3	-	-	-	-	-	y1
Gated: tri-state	Z	-	-	-	-	-	-	t3	Z	-	-	-	-	y1
Gated: filler value	*f*	-	-	-	-	-	-	t3	*f*	-	-	-	-	y1

Fig. 8. Cycle-by-cycle value from the gated multiplier output to register Reg2 in Fig. 3(b).

under four different scenarios. Z refers to tri-state (high-impedance), X implies that the value is not yet stable or visible at the RTL, y1' is the value of variable y1 in the previous iteration and '-' implies that the value remains unchanged. It is obvious that the SSA will be significantly suppressed by gating.

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**
    $P_f^n(0) = P_f^n(0) + P(w,v)P^n(v,1)$;
    $P_f^n(1) = P_f^n(1) + P(w,v)P^n(v,0)$;
  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 RTL circuit. Moreover, this simulation can also yield the probability for the nth bit of v to be 1 and 0, respectively, when v is output right before or after any variable from $V - V'$. Let us denote these probabilities by $P^n(v,1)$ and $P^n(v,0)$, respectively. Note that each value of v will have a lifetime of consecutive cycles. That is, v will hold the same value for these cycles. These consecutive cycles are counted as one single occurrence of the corresponding value when calculating the probabilities, because only transitions from w to v will consume dynamic power. 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 probabilities of transition when f^n is 0 and 1, $P_f^n(0)$ and $P_f^n(1)$, are

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

If $P_f^n(0) > P_f^n(1)$, f^n is set to 1, otherwise 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 say that the filler value thus chosen will yield minimum spurious switched capacitance in the wires. However, it has been found to reduce spurious switched capacitance significantly. When the data are totally random, *i.e.*, $P^n(v,1) = P^n(v,0) = 0.5$, the optimal filler value can be either 1 or 0, which reduces to the method in [63].

The overhead for setting an input to a fixed value is very low compared to the overhead for latches. One AND gate is enough for 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 energy at all. 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 problem. In typical ASICs, 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 a 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 gating techniques proposed in the above subsection, we introduce overheads in the controller for generating gating signals and introduce 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 [64] for overhead computation 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}, of the controller are estimated as

$$SC_{cl} = 0.5N_c{\cdot}10{\cdot}4{\cdot}C_{in}^{min} \cdot 3 = 60N_cC_{in}^{min}$$
$$A_{oh} = 4N_cA^{min}{\cdot}1.2 = 4.8N_cA^{min}$$

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

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

where $Area$ is the final circuit area, $\sqrt{Area}$ is used as the average gating control signal wire length, C_w^{min} is the unit-length capacitance for a minimum-width metal-1 wire, N_{sw} is the frequency of transitions 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. If interconnect SSA is suppressed using the proposed techniques, however, this will not always be true. Fig. 9 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 containing SSA as small triangles. Suppose tri-state

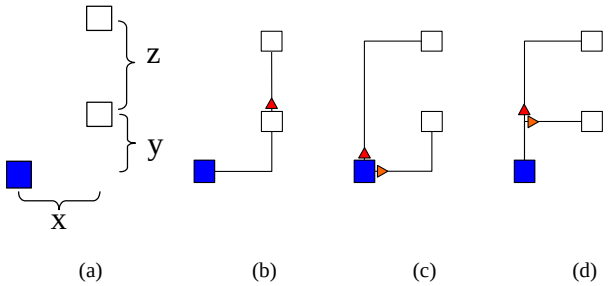


Fig. 9. Different output network topologies for (a) one DPU sending data to two others: (b) MST, (c) fully dedicated, and (d) trunk-branches.

buffers are used so that the last useful value is remembered in the gated idle cycles. Let α_1 denote the unit-length switched capacitance for data, say, for variables $v_1, ..., v_n$, needed by the upper receiver, which is estimated as

$$\alpha_1 = \sum_{i=1}^{n-1} SC(v_i, v_{i+1})$$

where $SC(v_i, v_{i+1})$ is the unit-length switched capacitance in the wire when v_i is followed by v_{i+1}. That for the lower receiver, α_2, and that for all the output variables of the sender, α, can be estimated in the same way. Assuming signal gating is performed at the optimal locations shown in Fig. 9, we can estimate the total wire switched capacitance for the output network implemented as follows (SC_{MST}, SC_{FD}, and SC_{TB} refer to the switched capacitance of an MST, fully dedicated network, and trunk-branches network, respectively).

$$SC_{MST} = (x+y)\alpha + z\alpha_1$$
$$SC_{FD} = (x+y+z)\alpha_1 + (x+y)\alpha_2$$
$$SC_{TB} = y\alpha + (x+z)\alpha_1 + x\alpha_2$$

The above equations imply that

$$SC_{FD} - SC_{TB} = y(\alpha_1 + \alpha_2 - \alpha)$$
$$SC_{TB} - SC_{MST} = x(\alpha_1 + \alpha_2 - \alpha)$$

Therefore, when $(\alpha_1 + \alpha_2) > \alpha$, we have $SC_{FD} > SC_{TB} > SC_{MST}$ 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.

E. SSA in buffers

As mentioned before, buffers can be very power-hungry too. To get the most benefit out of signal gating, a signal with SSA should be gated before it propagates into buffers. When buffers are inserted along a wire to reduce interconnect delay, gating at the most upstream location reduces SSA in both the wire and buffers. When the wire is driven by an output buffer, gating should be placed before the buffer. For a fully dedicated output network, instead of using one large output buffer to drive the whole output network, it is better to use a smaller buffer for each dedicated interconnect and its receiving DPU while gating the signal containing SSA before each buffer. Fig. 10 shows the two different output buffers for a dedicated output network for a DPU output sending data to two receivers (Load1 and Load2). Fig. 10(b) also shows the gating locations and control signals (en1 and en2). We call such output buffers *split output buffers*. For the shared output network implemented in the trunk-branches style, there are shared and dedicated parts in the output network. To maximize the benefit of SSA gating, instead of using a single large output buffer, we need to use split buffers, $i.e.$, a buffer for the shared part and a dedicated buffer for each dedicated part. Then we can gate signals containing SSA before the dedicated buffer. Fig. 11 shows the two different output buffers for a trunk-branches output network. It only shows two receivers (Load1 and Load2). Fig. 11(b) also shows the gating locations and control signals (en1 and en2). In both the dedicated and shared output network cases, split buffers consume no more power or area than the single large buffer. Meanwhile, split buffers facilitate SSA gating.

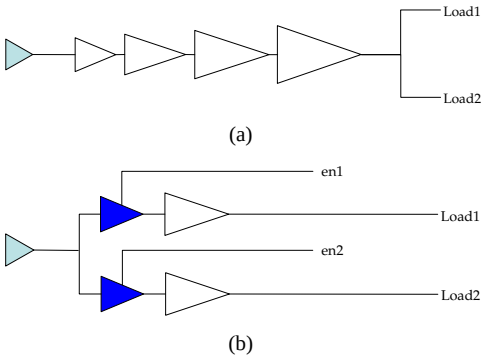


Fig. 10. Output buffers for a fully dedicated output network: (a) single large buffer, and (b) split buffers with signal gating.

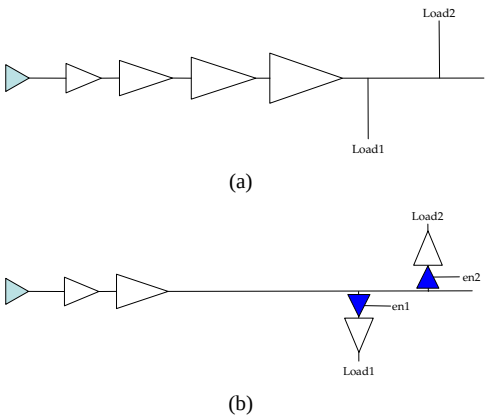


Fig. 11. Output buffers for a trunk-branches output network: (a) single large buffer, and (b) split buffers with signal gating.

VI. INTERCONNECT-AWARE HIGH-LEVEL SYNTHESIS FOR LOW POWER

We have implemented our techniques for interconnect-aware power optimization on top of a low power high-level synthesis tool, SCALP [10]. The implementations constitute about 4000 lines of additional C++ code over about 20,000 lines in SCALP. An overview of the new system is shown in Fig. 12.

First, the CDFG is simulated with typical input traces in order to profile operations and data transfers. 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.

The iterative improvement engine is used to optimize the RTL architecture for different objectives ($e.g.$, area, power) under performance constraints. 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

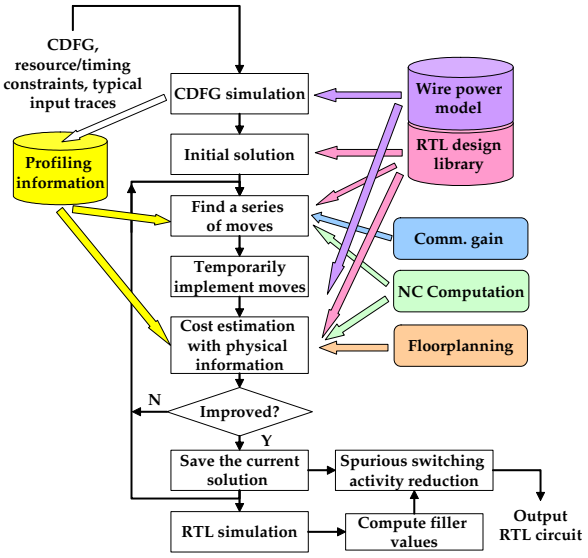


Fig. 12. The framework of the interconnect-aware high-level synthesis tool for low power

the reverse. Functional unit selection and resource sharing may necessitate rescheduling. Resource splitting may adversely affect the targeted objective. However, it enables 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. The neighborhood crowd checking and communication cost gain influence this step. Initially, a series of moves is identified based on their cost gain without floorplan information, and temporarily implemented. If these moves cause any scheduling conflict, the behavior is rescheduled with the binding constraints to solve the conflict. 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 maximum number of iterations has been reached. Then the best RTL architecture solution seen so far is output. This architecture is post-processed using the SSA reduction techniques to obtain the final RTL architecture. The post-processing includes control signal generation, gating logic insertion and controller re-specification. Judged by the synthesis flow, the interconnect-aware binding techniques basically accelerate the iterative improvement algorithm and help it escape local minima by finding better moves; the floorplanner and wire power model improve power estimation accuracy by incorporating physical level information; SSA reduction is simply a post-processing step.

As can be seen from Fig. 12, almost every module of the tool can be implemented independently if their interfaces are preserved. For example, the rescheduler, wire power model, RTL design library and floorplanner, can all be implemented

and upgraded independently. This adds flexibility and scalability to the tool. It also enhances the orthogonality of our interconnect-aware techniques to other problems.

Another feature of SCALP, and hence of our augmented SCALP, is that it can accept the sample period (inverse of throughput) as a performance constraint. Thus, both the interconnect-unaware and interconnect-aware implementations can be compared under the same performance constraint.

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 [44] which are themselves based on the information in [65]. The circuits were synthesized using an RTL design library based on NEC 0.18μm technology [45], [46]. SCALP, the high-level synthesis tool, on top of which we implemented our techniques, already has the capability to optimize circuits for area and power, however, in an interconnect-unaware fashion. Such SCALP-generated circuits are used for comparisons. All implementations of a benchmark are optimized under the same performance constraint.

The experiments were conducted on a 1.3 GHz Pentium IV based PC with 256 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 for high-level synthesis.

A. Interconnect-aware high-level synthesis

In Fig. 13, 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 with 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 with area-optimized (AO) circuits, power is reduced by an average of 53.3% at an average area overhead of 44.3%.

B. Gating signals with SSA

Table III 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 savings 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 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

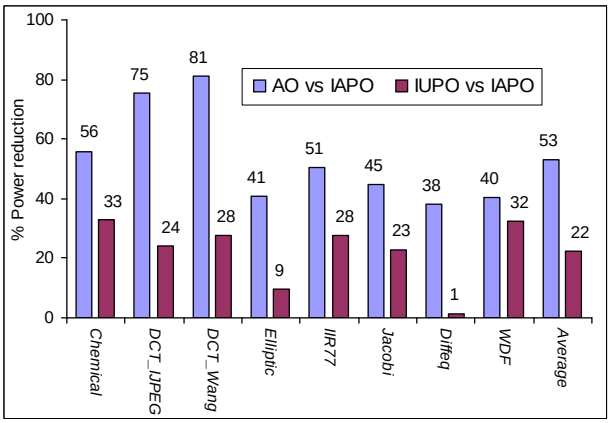


Fig. 13. Power reduction due to interconnect-awareness.

TABLE III
GATING SIGNALS WITH SSA.

	A_{oh} %	P_{oh} %	P_i %	P_d %	P_{tot} %
AO-gated	0.1	2.6	49.4	11.8	16.9
IUPO-gated	0.04	3.5	23.4	9.5	9.9
IAPO-gated	0.03	4.8	15.7	4.8	5.1

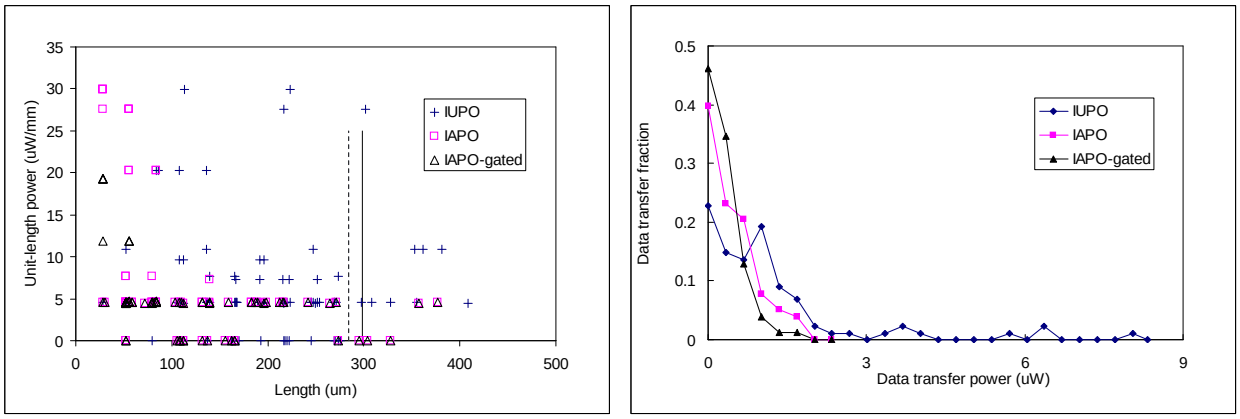
reduction is 5.1% (the power overhead is included in this number) at a negligible area overhead. For AO and IUPO circuits, the power reductions are more because they have more SSA in both interconnects and DPUs. It is worth noting that due to the difficulty in estimating glitches at the RTL, glitching power is not taken into consideration. If it could be estimated, we expect the power reduction to be much more.

C. Interconnect power reduction

Fig. 14 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 and larger designs, our methodology is likely to yield a higher total circuit power reduction.

To better illustrate the effectiveness of the proposed techniques, Fig. 15(a) shows the length and corresponding unit-length power consumption of individual inter-DPU data transfers for the IUPO, IAPO, and IAPO-gated implementations of the $Elliptic$ benchmark. It also shows the square root of the circuit area (the solid line for IUPO and dashed for IAPO) for reference. A data transfer is performed on a multi-bit wire connecting two DPUs. One can see that compared to IUPO circuits, only shorter wires have high unit-length power in IAPO circuits. The unit-length power of these wires is further reduced in IAPO-gated circuits.

Fig. 15(b) redraws the data in Fig. 15(a) as distributions of individual data transfer power for different implementations.



(a) Wire length vs. corresponding unit-length data transfer power. (b) Power distribution.

Fig. 15. Data transfer wire for $Elliptic$ when optimized under different scenarios.

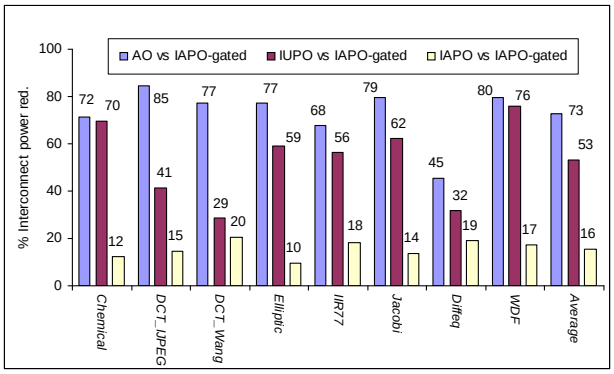


Fig. 14. Interconnect power reduction.

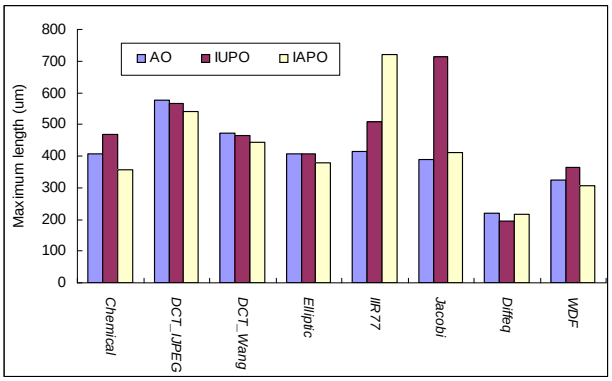


Fig. 17. The length of the longest data-transfer wires for AO, IUPO, and IAPO benchmarks.

The y co-ordinate specifies the fraction of all data transfers that consume power rounded to the number given by the x co-ordinate. These figures demonstrate that interconnect-aware high-level synthesis and signal gating drastically reduce the length of high unit-length power data transfers and the number of high-power data transfers. It should be noted that the IUPO implementation of $Elliptic$ has 87 inter-DPU data transfers while IAPO and IAPO-gated implementations have 85.

D. Total power reduction

Fig. 16 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.

It is worth noting that compared with AO circuits, the interconnect power of IAPO circuits is significantly reduced in spite of the area increase. In IAPO circuits, our techniques try to ensure that units exchanging high-switching data can be floorplanned as close to each other as possible, thus reducing switching activity for data transfer, and also minimizing the number of long data-transfer wires. Therefore, although IAPO circuits are larger, it has fewer power-hungry interconnects (and DPUs), and fewer long interconnects, as shown in Fig. 15.

E. Interconnect length

Although IAPO designs are larger than their AO counterparts, the interconnect-aware binding techniques are very effective in localizing data-transfer wires. Fig. 17 shows the length of the longest data-transfer wires in AO, IUPO, and IAPO benchmarks. The longest data-transfer wires for six (seven) out of the eight benchmarks are shorter for IAPO circuits than their AO (IUPO) counterparts. Figs. 18(a) and 18(b) show the accumulative length distributions of data-transfer wires for benchmarks $Elliptic$ and $IIR77$, respectively. $Elliptic$ is typical for the six benchmarks with shorter IAPO longest data-transfer wires. $IIR77$ is the one for which the length of the longest data-transfer wires is the worst for the IAPO circuits. However, in both of $Elliptic$ and $IIR77$, the IAPO circuit requires the least number of data-transfer wires between datapath units. More importantly, more data-transfer wires in IAPO circuits are distributed in the small length range than those in AO and IUPO ones, even for $IIR77$.

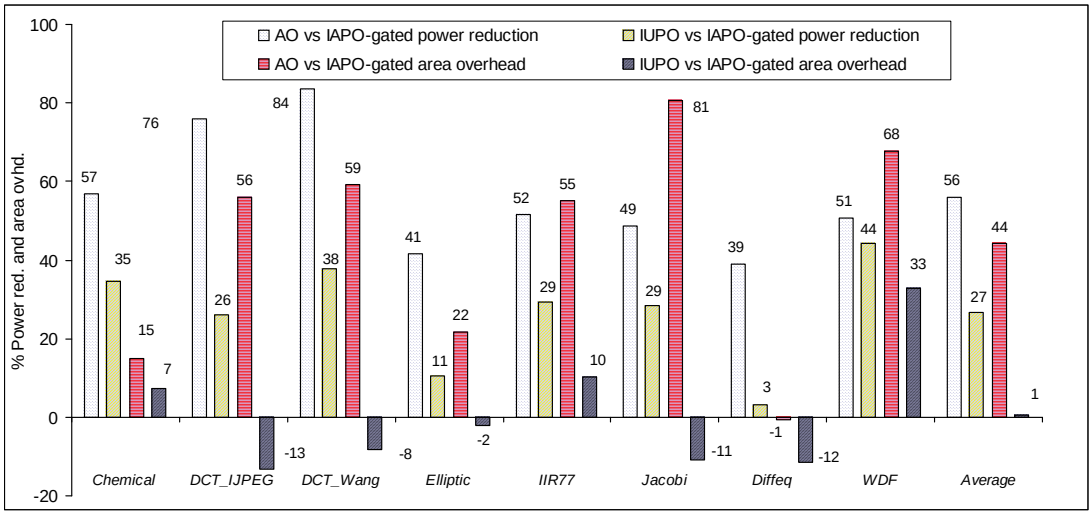
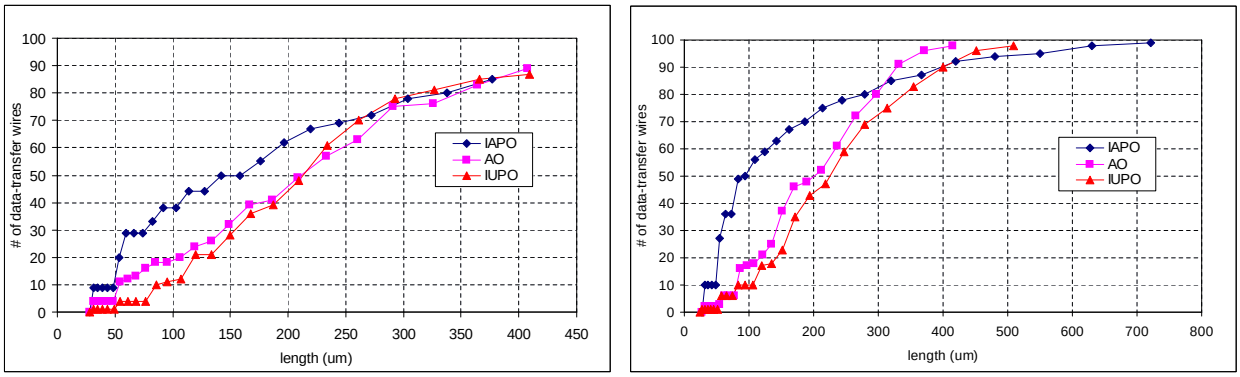


Fig. 16. Total power reduction and area overhead.



(a) $Elliptic$ (b) $IIR77$

Fig. 18. The accumulative length distributions of data-transfer wires for AO, IUPO, and IAPO circuits.

With localized data-transfer wires, IAPO circuits will, in general, have longer slacks in the interconnects. Because of the equal performance constraints we imposed on both AO and IAPO circuits, the slacks in datapath units are similar for both. Therefore, IAPO circuits are more likely to have shorter critical paths, which implies a performance advantage for interconnect-aware power optimization.

F. Commercial tools

Most commercial tools do not take directives from RTL floorplanners to lay out a circuit. Therefore, the physical locality created by our tool at the RTL may be undone when logic and physical synthesis are performed on IAPO RTL circuits using commercial tools. In spite of this current drawback of the commercial tools, we decided to check if the benefits of our techniques are preserved at the lower level. We used Synopsys Design Compiler to synthesize both IUPO and IAPO RTL circuits, and Cadence Silicon Ensemble for physical synthesis. A standard cell library for the TSMC 0.25μm process developed by the Virginia Tech. VLSI for Telecommunications (VTVT) group [66] was used. Table IV shows the number of semi-global and global wires (wires longer than 1μm) in the benchmarks synthesized by commercial tools. Not so surprisingly, seven of the eight benchmarks demonstrate fewer such wires for IAPO designs. Although the commercial tools may destroy the circuit hierarchy and not take any floorplan hints from the RTL design, our techniques are still powerful enough to have a positive impact in nearly all of the benchmarks. That is, a design with a better RTL interconnect topology turns out to have a better interconnect topology after final routing. (For the benchmark WDF, the negative impact may be due to the large area overhead of its IAPO implementation over its IUPO implementation, which may have overwhelmed the interconnect topology improvement when going through the RTL-hint-free commercial tools.) Many commercial tools plan to take RTL directives for logic and physical synthesis in the future. Thus, we expect our techniques to be more beneficial in conjunction with these future commercial tools.

VIII. Conclusions

Interconnects for data transfers consume a significant fraction of overall circuit 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. A

TABLE IV
NUMBER OF SEMI-GLOBAL AND GLOBAL WIRES IN COMMERCIAL-TOOL SYNTHESIZED IAPO AND IUPO DESIGNS.

	Chemical	*DCT_IJPEG*	*DCT_Wang*	*Diffeq*	*Elliptic*	*IIR77*	*Jacobi*	*WDF*
IAPO	8720	15298	13037	2872	11520	13919	11264	7682
IUPO	8770	17419	13999	2942	11726	18499	11584	6406

method for estimating data transfer power consumption at the RTL was proposed. Using this method, RTL interconnect power was 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 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 be easily incorporated into other high-level synthesis systems. Because the underlying tool, SCALP, is only applicable to data-dominated behaviors, the benchmarks used in this paper were data-dominated. Nevertheless, our techniques are equally applicable to control-flow intensive behaviors.

ACKNOWLEDGMENTS

The authors would like to thank Robert Dick for the floorplanner used in this work, Anand Raghunathan for the NEC ASIC RTL library, and Pallav Gupta for the Perl script used to parse netlist files. Also, the authors would like to thank the anonymous reviewers, whose comments significantly improved this paper.

APPENDIX

The fixed-taper buffer structure [67] is commonly used for output buffer design. It consists of a series of inverters, in which each inverter is larger than its upstream neighbor by the same factor. From [61], the dynamic power dissipation of the entire tapered buffer system is

$$P_{Dyn_{total}} = V_{DD}^2 f (C_x + FC_y)(\frac{\frac{C_L}{C_y} - 1}{F - 1})$$

where V_{DD} is the supply voltage, f the frequency, C_x the output diffusion capacitance of the first inverter, C_y the input gate capacitance of the first inverter, F the fixed taper factor, and C_L the load capacitance. C_L consists of two components, C_l, the downstream gate input capacitance which is the real load, and C_w, the capacitance of the wire connecting the driver and the load. Then the total capacitance is

$$C_{tot} = (C_x + FC_y)(\frac{\frac{C_L}{C_y} - 1}{F - 1})$$
$$= (\frac{F + \frac{C_x}{C_y}}{F - 1})C_w + (\frac{F + \frac{C_x}{C_y}}{F - 1})(C_L - C_y)$$

The second term on the right is independent of the interconnect. It can be incorporated into the input capacitance of the corresponding downstream DPU(s). The power consumption due to it can be taken into consideration by the DPU's power model. Therefore, we only need to worry about the first term when computing interconnect-related power consumption. F is equal to $\sqrt[N]{C_L/C_y}$, where N is the number of inverters, or stages, in the buffer. For power-delay product minimum buffers, F ranges from 4.5 to 10 [68]. C_x is usually smaller than C_y. Thus, we can estimate the interconnect-related equivalent capacitance of the buffer, C_{buffer}^w, as 1.1 to 1.3 times the wire capacitance C_w. In our estimation, we assume $C_{buffer}^w = 1.1C_w$. Hence, the interconnect-related dynamic power dissipation of the buffer is about 1.1 times that of the metal wire.

REFERENCES

[1] A. Raghunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*. Norwell, MA: Kluwer Academic Publisher, 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, no. 1, 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, "A interconnect-centric design flow for nanometer technologies," *Proc. IEEE*, vol. 89, no. 4, pp. 505–528, Apr. 2001.
[14] D. D. Gajski, N. D. Dutt, A. Wu, and S. T. Lin, *High-level Synthesis: Introduction to Chip Design*. Norwell, MA: Kluwer Academic Publisher, 1992.
[15] P. G. Paulin and J. P. Knight, "Scheduling and binding algorithms for high-level synthesis," in *Proc. Design Automation Conf.*, June 1989, pp. 1–6.
[16] C. A. Papachristou and H. Konuk, "A linear program driven scheduling and allocation method followed by an interconnect optimization algorithm," in *Proc. Design Automation Conf.*, June 1990, pp. 77–83.
[17] T. A. Ly, W. L. Elwood, and E. F. Girczyc, "A generalized interconnect model for data path synthesis," in *Proc. Design Automation Conf.*, June 1990, pp. 168–173.

[18] E. D. Lagnese and D. E. Thomas, "Architectural partition for system level synthesis of integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. 10, no. 7, pp. 847–860, July 1991.
[19] C. Monahan and F. Brewer, "Communication driven interconnection synthesis," in *Proc. 6th Int. Workshop High-level Synthesis*, 1992.
[20] C. Jego, E. Casseau, and E. Martin, "Interconnect cost control during high-level synthesis," in *Proc. Design Circuits & Integrated Systems Conf.*, Nov. 2000, pp. 507–512.
[21] S. Tarafdar and M. Leeser, "The DT-model: High-level synthesis using data transfers," in *Proc. Design Automation Conf.*, June 1998, pp. 114–117.
[22] M. R. Stan and W. P. Burleson, "Low-power encoding for global communication in CMOS VLSI," *IEEE Trans. VLSI Systems*, vol. 5, no. 4, pp. 49–58, Dec. 1997.
[23] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "A coding framework for low-power address and data busses," *IEEE Trans. VLSI Systems*, vol. 7, no. 2, pp. 212–221, June 1999.
[24] P. P. Sotiriadis and A. Chandrakasan, "Low power bus coding techniques considering inter-wire capacitances," in *Proc. Custom Integrated Circuits Conf.*, May 2000, pp. 507–510.
[25] 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.
[26] M. C. McFarland and T. J. Kowalski, "Incorporating bottom-up design into hardware synthesis," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 9, pp. 938–950, Sept. 1990.
[27] 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.
[28] J.-P. Weng and A. C. Parker, "3D scheduling: High-level synthesis with floorplanning," in *Proc. Design Automation Conf.*, June 1992, pp. 668–673.
[29] C. Ramachandran and F. J. Kurdahi, "Combined topological and functionality based delay estimation using a layout-driven approach for high level applications," *IEEE Trans. Computer-Aided Design*, vol. 13, no. 12, pp. 1450–1460, Dec. 1994.
[30] 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.
[31] 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.
[32] 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.
[33] 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.
[34] 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.
[35] W. E. Dougherty and D. E. Thomas, "Unifying behavioral synthesis and physical design," in *Proc. Design Automation Conf.*, June 2000, pp. 756–761.
[36] 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.
[37] 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.
[38] G. Lakshminarayana, A. Raghunathan, N. K. Jha, and S. Dey, "Power management in high-level synthesis," *IEEE Trans. VLSI Systems*, vol. 7, no. 1, pp. 7–15, Mar. 1999.
[39] L. Zhong, J. Luo, Y. Fei, and N. K. Jha, "Register binding based power management for high-level synthesis of control-flow intensive behaviors," in *Proc. Int. Conf. Computer Design*, Sept. 2002, pp. 391–394.
[40] Independent JPEG Group. *http://www.ijg.org*.
[41] K. R. Rao and P. Yip, *Discrete Cosine Transform*. London: Academic Press, 1990.
[42] NCSU CBL high-level synthesis benchmark suite. *http://www.cbl.ncsu.edu/benchmarks/*.
[43] S.-Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
[44] 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.
[45] NEC cell-based ASIC CB-11. *http://www.necel.com/ASIC/*, 2000.
[46] A. Raghunathan, *Personal communication.*
[47] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Englewood Cliffs, NJ: Prentice Hall, 1996.
[48] 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.
[49] 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.
[50] P. P. Sotitriadis and A. Chandrakasan, "A bus energy model for deep submicron technology," *IEEE Trans. VLSI Systems*, vol. 10, no. 3, pp. 341–350, June 2002.
[51] P. Heydari and M. Pedram, "Interconnect energy dissipation modeling in high-speed ULSI circuits," in *Proc. Asia & South Pacific Design Automation Conf.*, Jan. 2002, pp. 132–137.
[52] P. Gupta, L. Zhong, and N. K. Jha, "A high-level interconnect power model for design space exploration," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003, pp. 551–558.
[53] N. Sherwani, *Algorithms for VLSI Physical Design Automation: Second Edition*. Norwell, MA: Kluwer Academic Publishers, 1995.
[54] 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.
[55] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.
[56] P. Christie, "A fractal analysis of interconnection complexity," *Proc. IEEE*, vol. 81, no. 10, pp. 1492–1499, Oct. 1993.
[57] D. Sylvester and K. Keutzer, "System-level performance modeling with $BACPAC$ - Berkeley advanced chip performance calculator," in *Proc. Workshop on System-Level Interconnect Prediction*, Apr. 1999, pp. 109–114.
[58] J. A. Davis, V. K. De, and J. D. Meindl, "A stochastic wire-length distribution for gigascale integration (GSI) - Part I: Derivation and validation," *IEEE Trans. Eletronic Devices*, vol. 45, no. 3, pp. 580–589, Mar. 1998.
[59] ——, "A stochastic wire-length distribution for gigascale integration (GSI) - Part II: Application to clock frequency, power dissipation, and chip size estimation," *IEEE Trans. Eletronic Devices*, vol. 45, no. 3, pp. 590–597, Mar. 1998.
[60] 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.
[61] 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.
[62] 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*. IEEE Press, NY, 1988, pp. 459–470.
[63] 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.
[64] 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.
[65] *National Technology Roadmap for Semiconductors*. Semiconductor Industry Association, 1997.
[66] Virginia Tech. VLSI for Telecommunications Group: Standard cell library, *http://www.ee.vt.edu/∼ha/cell_library/distribution.html*.
[67] H. C. Lin and L. W. Linholm, "An optimized output stage for MOS integrated circuits," *IEEE J. Solid-State Circuits*, vol. SC-10, no. 2, pp. 106–109, Apr. 1975.
[68] J.-S. Choi and K. Lee, "Design of CMOS tapered buffer for minimum power-delay product," *IEEE J. Solid-State Circuits*, vol. 29, no. 9, pp. 1142–1145, Sept. 1994.