# Simultaneous Scheduling, Binding and Floorplanning for Interconnect Power Optimization

*Pradeep Prabhakaran,James Crenshaw, Prithviraj Banerjee and Majid Sarrafzadeh*

**Technical Report No. CPDC-TR-9802-012**

March 15 1998

*Center for Parallel and Distributed Computing*
Department of Electrical & Computer Engineering
Northwestern University
Technological Institute
2145 Sheridan Road, Evanston IL-60208

# Simultaneous Scheduling, Binding and Floorplanning for Interconnect Power Optimization

Pradeep Prabhakaran
Center for Reliable and High-Performance Computing,
University of Illinois,
1308 West Main St, Urbana, Illinois 61801

James Crenshaw, Prithviraj Banerjee and Majid Sarrafzadeh
Department of Electrical and Computer Engineering
Northwestern University
2145 Sheridan Road
Evanston, Illinois 60208

March 16, 1998

**Abstract**

Interconnect power dissipation is becoming a major component of power consumption in a circuit especially in submicron technologies. The energy dissipated by a particular interconnection link is determined by the switching activity on that link and also on its capacitance. The switching activity is determined by the schedule and binding, whereas the capacitance is determined by the floorplan. Scheduling, binding and floorplanning are closely inter-related. A simultaneous scheduling, binding and floorplanning algorithm is presented which attempts to minimize interconnect energy dissipation. In comparison to a traditional approach which separates high-level synthesis from physical design, our algorithm is able to make these stages interact very closely, resulting in solutions with lower power, latency and area. We show that it is possible to reduce the interconnect energy dissipation by upto 60 percent for high-level synthesis benchmark circuits.

KEYWORDS: High-level synthesis, low-power design, timing driven synthesis, floorplanning.

This research was supported in part by the National Science Foundation under grant MIP-9320854, the Defense Advanced Research Projects Agency under contracts DAA-H04-94-G-0273, and DABT-63-97-0035 administered by the Army Research Office.

# 1 Introduction

Recently, there has been a proliferation of battery operated portable devices that demand intensive computation. Prolonging battery life requires design of power-efficient circuits. Submicron features have lead to an increased fraction of the power to be dissipated by interconnects (upto 40 percent as reported in [9], [32]). In order to increase the battery lifetime for portable applications, it is necessary to decrease the average power dissipation. In addition, decreasing the interconnect power is important to increase reliability in submicron technology by reducing electromigration and hot-electron effects. Further, by decreasing the charge drawn by the circuit, ground bounce and excessive voltage drop on buses can be minimized.

Scheduling and module binding are the major steps in high level synthesis. Scheduling assigns the nodes in the control data flow graph (CDFG) to specific time steps,and binding assigns the nodes to specific functional units. Floorplanning determines the actual positions of modules in a physical design.

The interconnect energy dissipation is directly dependent upon the switching activity and the capacitance of the interconnects. The switching activity on an interconnect is determined by the sequence of variables assigned to it, which in turn depends on the schedule and binding. The capacitance of an interconnect is directly dependent on its length, which is determined by the floorplan. Thus, to minimize the interconnect energy dissipation, it is important to consider the switching as well as the floorplan information during scheduling and binding. Similarly, the floorplanner should take the schedule and binding into account to minimize the interconnect energy dissipation. It should try to minimize the length of interconnects which have high switching activity, for example.

The latency of a schedule depends on the cycle time and the number of cycles in the schedule. The cycle time is determined by the longest path in a schedule step, which includes the functional unit delays, multiplexor delays, register delays and interconnect delays. The number of cycles is determined by the schedule. The area of the generated floorplan is determined by the number of modules used, and their positioning.

In this work, we present a simulated annealing algorithm which simultaneously performs scheduling, binding and power-driven floorplanning with the objective of optimizing the interconnect energy dissipation, while trying to minimize the impact on latency and area. We compare the results with that of an algorithm which considers only latency and area, and ignores power and show that it is possible to reduce interconnect energy dissipation by 30 to 60 percent for several benchmark circuits.

In Section 2 a brief survey of existing work is presented. In Section 2, the power estimation algorithm is described. In Section 3, a constructive power-driven floorplanning algorithm is presented whose primary objective is to reduce interconnect energy dissipation. It also tries to reduce delays along critical paths as a secondary objective. In Section 4, the algorithm for combined scheduling,binding and floorplanning which minimizes interconnect energy is described. In Section 5, the results are presented for a set of high-level synthesis benchmarks.

## 2 Related Work

There has been considerable amount of research in considering power during high-level synthesis. A simultaneous scheduling and binding algorithm is described in [4] which tries to minimize the switching activity on buses in a bus-based architecture. A high-level datapath power estimation algorithm which derives accurate power estimates for general $CDFG$s has been presented in [1].

An allocation method and controller optimization algorithm for low power is presented in [3].An interactive improvement method for scheduling,clock selection, module selection and resource allocation with the aim of reducing the power consumption has been presented by the same authors in [2]. An algorithm for optimizing total power consumption and area during selection and placement of modules is presented in [5]. Computational transformations such as operation reduction, operation substitution, word length reduction and transformation ordering have been used by [7] to reduce power. An in-depth survey of power minimization algorithms at architectural,logical and physical levels appears in [8].

Previous researchers have addressed the problem of incorporating physical design information in high level synthesis with the objective of minimizing latency and area. Fang and Wong [12] describes a integrated binding and floorplanning algorithm which performs a constructive binding for each move of a simulated annealing based floorplanning algorithm.

Weng and Parker [17] describe an algorithm for simultaneous scheduling, binding and floorplanning. They perform the floorplanning of modules in critical paths, followed by an iterative improvement phase which performs rebinding to reduce the latency.

Jang and Pangrle [31] performs connectivity binding based on geometric costs for bit-sliced designs. Fasolt [18] performs floorplanning and analyzes the area impact during scheduling and binding. BUD [19],ELF [21] and Chippe [20] are other algorithms which attempt to incorporate physical design information in high-level synthesis.

In this paper, we incorporate physical design information regarding floorplanning and interconnect capacitances into a simultaneous scheduling, binding and floorplanning algorithm with the objective of minimizing power.

# 3 Power Estimation using switching tables

The power dissipation as a result of switching can be calculated using the equation $P = \frac{1}{2T}\sum_{i \in nodes} V_{DD}^2 s_i c_i$ where $s_i$ is the number of times node $i$ switches during time $T$, $c_i$ is the capacitance of node $i$ and $V_{DD}$ is the supply voltage.

The estimation of switching on a resource can be efficiently performed using the idea of the switching table [6],[3]. A switching table for a resource $r$, for a signal flow graph without conditional statements, is a two-dimensional matrix $S$ where the row $i$ and column $i$ stands for the $i^{th}$ operation, and $S(i,j)$ is the switching that results if $i$ and $j$ are scheduled on $r$ one after the other with no other operation in between. For interconnection links, an operation is essentially the transmission of a variable from one module to another.

The switching table for resource $r$ can be expressed as the sum of two switching tables [1]. One is the intra-iteration switching table $M_r$ and the other is the inter-iteration switching table $N_{rk}$. Intra-iteration switching $M_r(i,j)$ represents the switching incurred when operation $i$ is scheduled immediately before $j$ on resource $r$. Inter-iteration switching $N_{rk}(i,j)$ represents the switching incurred when operation $j$ is scheduled first in loop $k$ on resource $r$ and $i$ is the last operation scheduled in loop $k$ on resource $r$.

The switching table can be calculated by simulation of the operations in the signal flow graph using representative input vectors. Each input vector results in a particular value for each variable, and the switching can be calculated using the switching function for the resource. Details of switching table calculation can be found in [1].

Consider a particular schedule, binding and floorplan for a signal flow graph. Let $l_{ij}$ be the interconnection in the floorplan between modules $i$ and $j$. Let $v_1, v_2, \ldots v_k$ be the variables assigned in sequence on $l_{ij}$ as a result of communications between modules $i$ and $j$ to perform the computations assigned to $i$ and $j$ by the schedule and binding. Then the switching $s_{ij}$ on the link is determined by pairs of variables $(v_1, v_2), (v_2, v_3) \ldots (v_{k-1}, v_k)$ as given by the

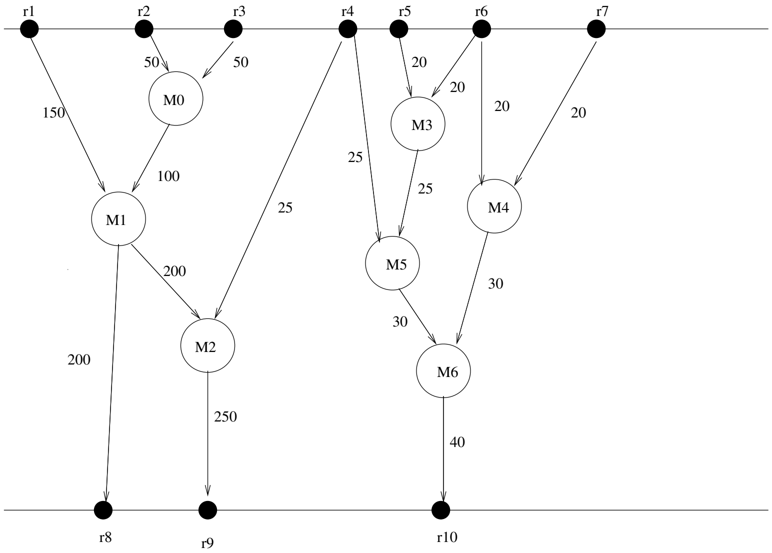



Figure 1: Example showing switching along edges between modules

switching table. The capacitance $c_{ij}$ is proportional to the distance between $i$ and $j$ in the floorplan. Thus, the energy dissipated by $l_{ij}$ is proportional to $s_{ij}c_{ij}$.

# 4 The Floorplanning Algorithm

The constructive power driven floorplanning algorithm is similar to that proposed in [13] for latency reduction. It considers the switching activity on the interconnections between modules to decide which modules have to be adjacent in the floorplan. In addition it also attempts to reduce lengths of critical paths and the overall area of the floorplan. Figure 1 shows an example of the communication between functional modules (shown as arcs between modules) for a particular cycle. The numbers next to each arc represents the switching on that link. Consider modules $r_1$,$M_1$,$M_2$ and $r_9$ in the figure. Since the links between them exhibit high switching activity, the capacitance of the interconnections between the module pairs $(r_1, M_1)$, $(M_1, M_2)$ and $(M_2, r_9)$ could have a major impact on the overall power. Thus, it is desirable to have them as close as possible to each other in the floorplan. However, since the link between modules $r_5$ and $M_3$ have low switching activity, they can be farther away in the floorplan without possibly affecting the power consumption.

The algorithm has two main phases. In the first phase, a cost matrix is constructed for which the entry $(i, j)$ is a function of the switching on the link between modules $i$ and $j$ and also of the slacks of connections between the modules. The higher the value of the entry $(i, j)$, the closer the modules $i$ and $j$ should be in the floorplan, in order to minimize the capacitance of the link between them and also to minimize the communication delay on the link.

In the second phase, modules are clustered using a greedy clustering algorithm in a bottom-up manner, based on the cost matrix calculated in the previous phase. Along with the clustering, the bounding curve of each cluster is determined in a bottom up manner using a curve-addition technique, and the minimum cost point is determined from that, in a manner similar to that proposed by [15]. This point determines the slicing structure of the floorplan generated by the algorithm.

In the final phase, the slicing structure is constructed in a top down manner based on the bounding curves computed in the previous phase. This results in a set of modules, at each level of the slicing structure, each of which is either atomic (a leaf node of the slicing tree) or compound (a non-leaf node in the slicing tree). These modules are then reoriented in a bottom-up manner in order to further reduce the interconnect lengths, employing a technique similar to one proposed in [23],[15]. The various phases are explained in detail in the following sections.

## 4.1 The Cost matrix

The cost matrix $C$ is an $n \times n$ matrix where $n$ is the number of modules. It is computed as

$$C = \lambda T + (1 - \lambda)U$$

where $T$ is the transition matrix where $T(i, j)$ is the total switching on the edge between module $i$ and $j$, $U$ is the urgency matrix where $U(i, j)$ is given by $(1 - Slack(i, j))$ and $\lambda$ is a user defined parameter which indicates the relative importance of power minimization over latency minimization. Computation of $T$ and $U$ are performed as follows.

First an extended signal flow graph [12] is constructed for each step of the schedule. For a given step, at most one operation of the signal flow graph can be assigned to a functional unit. Each node in the extended $SDFG$ is a functional unit, a register or a multiplexor. There is an edge between two nodes in the extended $SDFG$ if there is a data flow between the operations assigned to the modules corresponding to those nodes. For example, consider the $SDFG$ in Figure 2. The operation $b$ of the signal flow graph is assigned to module $m_4$. One of the operands for $b$

is computed in the previous cycle, and stored in register $r_3$. The other operand is computed in the same cycle by $m_1$. Assume that $m_4$ requires a mux ($mux_1$) at the second input. Then, there are edges in the $SDFG$ from $r_3$ to $mux_1$, $mux_1$ and $m_1$ to $m_4$, and $m_4$ to $r_3$.

#### 4.1.1 Transition matrix calculation

Each edge of the extended signal flow graph represents a variable computed by the source node of the edge and consumed by the sink node of the edge. As explained before, the switching on an edge has two components, intra-iteration switching and inter-iteration switching. The transition matrix $T$, which stores the total switching between every module pair $i$ and $j$ is given by

$$T(i,j) = Intra(i,j) + Inter(i,j)$$

where $Intra(i,j)$ and $Inter(i,j)$ denote the intra and inter-iteration switching respectively on the link between modules $i$ and $j$. $T(i,j)$ is zero if there is no communication between modules $i$ and $j$.

Let $V_s(i,j)$ be the variable assigned to edge $(i,j)$ during step $s$ and let $t$ be the earliest step after $s$ during which a different variable is assigned to edge $(i,j)$. Then

$$Intra(i,j) = \sum_s S_{intra}(V_s(i,j), V_t(i,j))$$

where $S_{intra}$ is the intra-iteration switching table computed as described before.

$Inter(i,j)$ is calculated as follows. Let $First(i,j)$ be the first variable assigned to edge $(i,j)$ from the beginning of the schedule. Let $Last(i,j)$ be the last variable assigned to $(i,j)$. Then

$$Inter(i,j) = S_{inter}(Last(i,j), First(i,j))$$

where $S_{inter}$ is the inter-iteration switching table.

#### 4.1.2 Urgency Matrix Calculation

The cycle time consists of the worst case module delays, the multiplexor and register delays, the interconnect delays between the modules, and the delay in the control unit for any step.

The longest delay $T$ for any step is calculated, assuming zero interconnect latency. Based on this, the slacks of each arc in the extended signal flow graph are calculated. It is the longest delay that can be assigned to that arc without affecting the cycle time. Thus, the arcs of the critical paths will be assigned a slack value of zero. The calculation of edge slacks is performed in a manner similar to that proposed in [12], [13] as follows.

Let $T$ be the longest delay for a time step assuming zero interconnect delays. Consider the nodes in a given time step $S$. Let $ES(i)$ and $LS(i)$ denote the earliest start time and latest start time for node $i$ in $S$ of the extended $SDFG$. $ES(i)$ is determined by performing an $ASAP$ (as soon as possible) schedule of the nodes in $S$. By $ASAP$, the starting time of a node is set to the earliest time during which its source operands are available from the predecessors.

$$\begin{aligned} ASAP(n) &= 0 \text{ if } n \text{ has no predecessors,} \\ &= \max_{p \in preds(n)}(ASAP(p) + latency(p)) \text{ otherwise} \end{aligned}$$

$LS_n$ is determined by performing an $ALAP$ (as late as possible) schedule of the nodes in $S$.

$$\begin{aligned} ALAP(n) &= T - latency(n) \text{ if } n \text{ has no successors,} \\ &= \min_{s \in succs(n)}(ALAP(s) - latency(n)) \text{ otherwise} \end{aligned}$$

The slack of edge $e_{ij}$ from module $i$ to module $j$ is calculated as:

$$Slack(i, j) = LS(j) - ES(i) - latency(i)$$

Once the edge slacks are calculated, we assign a delay value to each edge so that the slack is reduced to zero. The edges in the critical path are assigned a delay of zero, for example. These delays can be regarded as a measure of urgency which indicates how close the pair of modules should be in the floorplan. The lower the delay, the closer the modules should be, to minimize the effect of interconnect delays on the overall latency of the schedule. It can be seen that assigning a delay value to an arc will cause the slack values of other arcs to change. In order to treat each arc independent of others, we distribute the slacks among the edges in a path (similar to *zero slack algorithm* [29] for standard cell placement). Details of the slack distribution can be found in [14]. The process is illustrated in the following example.

Figure 2 shows the slack values for edges for the example graph, and the delay values assigned by distributing the slacks. Initially, we start with the assumption that all edges have zero delays. Hence, step 1 has latency 9, and step 2

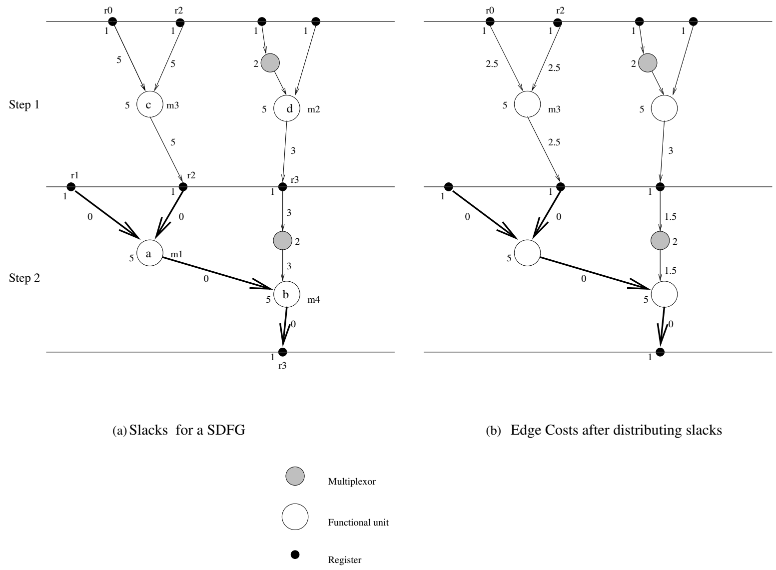



Figure 2: Slacks for a sample SDFG and edge weights after slack distribution

has latency 12 and the cycle time of the schedule is $\max(12, 9) = 12$. Consider the edge between $m_3$ and $r_2$. The earliest start time of $c$, $ES(c)$, the operation assigned to $m_3$ in step 1, is 1, which is the earliest time during which the contents of register $r0$ is available at the beginning of the cycle. The latest start time of register write to $r_2$ is 11, without violating the cycle delay of 12. Thus, the slack of the edge from $m_3$ to $r_2$ is $(11 - 1 - 5) = 5$. Similarly, the slack of edge between $r_0$ and $m_3$ is 5. While distributing the slacks, the edge with the lowest non-zero slack in step 1 is $(r_0, m_3)$ with slack value 5. The longest path in step 1, with slack $\geq 5$, and including edge $(r_0, m_3)$ is $(r_0, m_3, r_2)$. This has two edges. Thus, we assign a delay of $5/2 = 2.5$ each to $(r_0, m_3)$ and $(m_3, r_2)$. This makes the slacks of both the above edges zero. The delay values of other edges are derived in a similar way, and are as shown in Figure 2.

After the edge weights in the extended $SDFG$ are calculated for each step, the overall edge weight for the interconnection between each pair of modules is calculated by finding the minimum of the edge weights for that edge across all steps. These edge weights between every pair of modules $(i, j)$ gives the urgency matrix $U$.

## 4.2 Clustering Phase

We use a greedy algorithm based on the cost matrix to cluster modules. Initially, we start with a set of degenerate clusters which is essentially the set of modules itself, each module being a cluster.

Let $C$ be the cost matrix computed in the previous phase. For a given pair of clusters $m$ and $n$, if both of them are

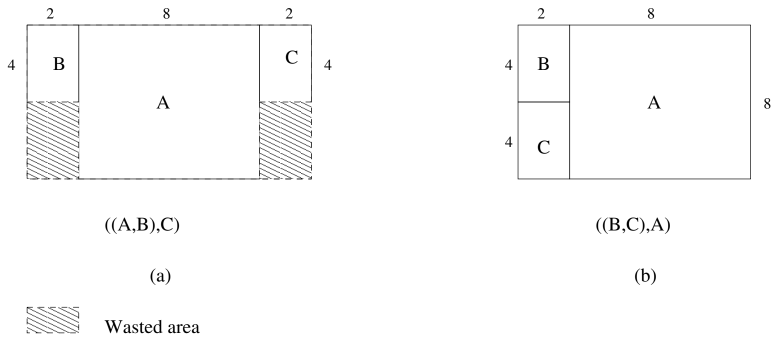



Figure 3: Example showing how choice of clusters combined based on area affects overall area

degenerate, cost $c_{m,n}$ is given by $C(m,n)$. Otherwise, the cost $c_{m,n}$

$$c_{m,n} = \min_{i \in m, j \in n} C(i,j)$$

The greedy algorithm selects the pair of clusters which have the highest cost to form a bigger cluster. If there are multiple pairs with the same cost, that pair having similar area are selected to be clustered. Figure 3 shows an example of how selection of the two clusters based on their sizes can affect the overall area. In (a), modules $A$ and $B$ are clustered together, and the result clustered with $C$. This results in a lot of wasted area. In (b), modules $B$ and $C$ are clustered together, and the result clustered with $A$ resulting in no wasted area, and a floorplan with lower area.

When two clusters are coalesced to form a larger cluster, the set of arcs from that cluster is the union of all arcs from the modules within the cluster to the modules in other clusters. This combining process is repeated until we have one cluster.

### 4.3 Bounding Curve Calculation

The bounding curve [11], [24], [26] of a module is the set of all minimal $(w_i, h_i)$ pairs for the module, where $w_i$ is the minimum width of the module when its height is $h_i$ (and vice versa). Given a slicing tree, we can associate a bounding curve with each node. Consider the leaf nodes. The points in the bounding curve are determined as explained above. The bounding curve of a non-leaf node is determined by combining the bounding curves of the child nodes. The combining process is determined by the type of dissection specified by the parent node, viz vertical or horizontal (denoted by $+$ and $*$ respectively). The combined curve gives the set of all feasible dimensions of the module formed by putting the child modules adjacent to each other horizontally (for a vertical slice), or vertically (for a horizontal slice). Given two bounding curves $\Gamma$ and $\Lambda$, $\Gamma + \Lambda$ is given by the curve $\{(u, v + w)|(u, v) \in \Gamma$ and $(u, w) \in \Lambda\}$. $\Gamma * \Lambda$ is similarly defined by the curve $\{(u + v, w)|(u, w) \in \Gamma$ and $(v, w) \in \Lambda\}$. Only the minimal points need to be stored, ie, there does not exist two points $i$ and $j$ in the bounding curve such that $w_i \leq w_j$ and $h_i \leq h_j$.

In addition to the width and length measures for each module, we add a third dimension, which is the cumulative interconnect cost measure. For a leaf node, it is zero. For a non-leaf node, it is determined by adding the cost of the child nodes and the Manhattan distance between the centers of the modules for the slicing structure being considered, weighted by the cost of the edge. Thus, the points in the augmented bounding curve consists of tuples of the form $\langle w_i, h_i, d_i \rangle$. Consider combining two points $\langle w_1, h_1, d_1 \rangle$ and $\langle w_2, h_2, d_2 \rangle$ assuming a vertical slice. The resulting point is given by $\langle w_1 + w_2, max(h_1, h_2), (d_1 + d_2 + (w_1 + w_2)/2) \rangle$.

In contrast to the conventional schemes for curve addition [24], [26], [11], [28], [27] to compute the bounding curves, we assume that the slicing structure of the layout is not known during the augmented bounding curve calculation. Consider a step during clustering in which a set of child clusters is combined to form a parent cluster. Since the clustering and augmented bounding curve computation is done in a bottom up manner, the augmented bounding curves of all the child clusters are known at this point. The set of all slicing structures of the parent cluster is now enumerated [22]. Corresponding to each slicing structure, the augmented bounding curve of the parent module is computed from the augmented bounding curves of the child clusters as explained above. These augmented bounding curves, corresponding to all the slicing structures of the parent, are now combined using a minimum function. As a result, we get a combined augmented bounding curve $B$ whose individual points are taken from the augmented bounding curves for the different slicing structures and such that the points in the resulting curve are minimal. In other words, if $B_1, B_2 \ldots B_k$ are the bounding curves corresponding to the different slicing structures, the combined curve $B$ consists of the set of all points in $B_1 \cup B_2 \cup \ldots B_k$ and there does not exist a pair of points $i$ and $j$ in $B$ such that $w_i \leq w_j$ , $h_i \leq w_j$ and $d_i \leq d_j$. The identity of the slicing structure associated with each point is recorded, which aids in generating the floorplan from top down.

Once the augmented bounding curve for the highest level cluster is determined, the cost of each point in that curve is computed based on a function which is a combination of area and interconnect costs determined by the bottom-up

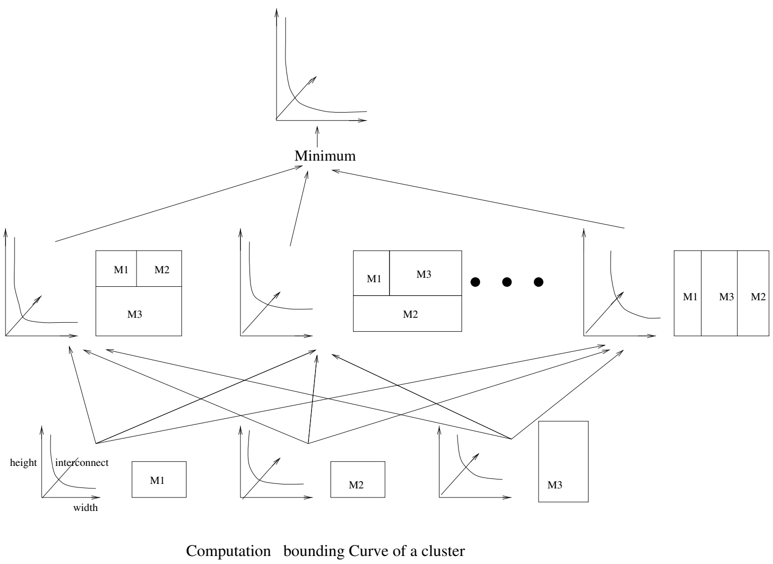


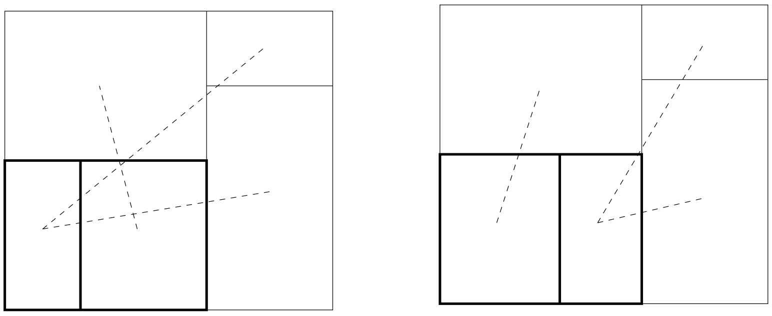

Figure 4: Computation of augmented bounding curve of a cluster

clustering and augmented bounding curve computation. The minimum cost point is selected. This point, along with the topology information stored by the previous phase uniquely determines the slicing structure of the floorplan.

Figure 4 shows an example in which the augmented bounding curve of a cluster consisting of three child clusters are combined to form the augmented bounding curve for the parent cluster.

## 4.4 Reorientation Phase

Once the topology is known, a reorientation of the clusters is performed. It is based on a technique similar to that proposed in [23]. Consider a cluster size of 2 for example. For the top level cluster, there are two child nodes. For the given topology, there are two orientations for each child module which conserve the area of the parent (four, if the child module is square). Each of these requires the same area, but they have different interconnect costs. The orientations which gives the best interconnect cost is determined. This process is recursively performed on each of the child node. When deciding the orientation of a module, the interconnects to all other modules in the floorplan is taken into account (in addition to the sibling). Thus, this approach tries to minimize the interconnection cost in a global fashion. Figure 5 shows a reorientation of a cluster.




An example of how reorientation reduces interconnect cost
(dashed lines indicate interconnections)

Figure 5: Reducing interconnect cost by reorientation

### 4.5 Calculation of Interconnect Delays

The length of the interconnection between two modules in a floorplan is taken as the Manhattan distance between the centers of the modules. Given the interconnect length $l_{int}$, the following equations are used to calculate the interconnect delay [12], [32].

$$R_{int} = \rho \frac{l_{int}}{W_{int} H_{int}}$$

$R_{int}$ = resistance of wire, $W_{int}$ = width of wire, $H_{int}$ = width of wire, $l_{int}$ = length of wire.

$$C_{int} = \epsilon_{ox}\{1.15(\frac{W_{int}}{t_{ox}}) + 2.8(\frac{H_{int}}{t_{ox}})\}$$

$C_{int}$ = capacitance per unit length

$$T = R_{int}C_{int} + 2.3(R_{tr}C_{int} + R_{tr}C_L + R_{int}C_L)$$

where $R_{tr}$ = equivalent resistance of driver transistor, $C_L$ = load capacitance (assume to be equal to gate capacitance of one transistor).

We make the following assumptions (as in [12]) about wire height $H_{int}$, wire width $W_{int}$ and thickness of oxide $t_{ox}$. $H_{int} = 1/3W_{int}$ and $t_{ox} = 1/2W_{int}$ and interconnect material is Aluminium. As in the case of [12], we select

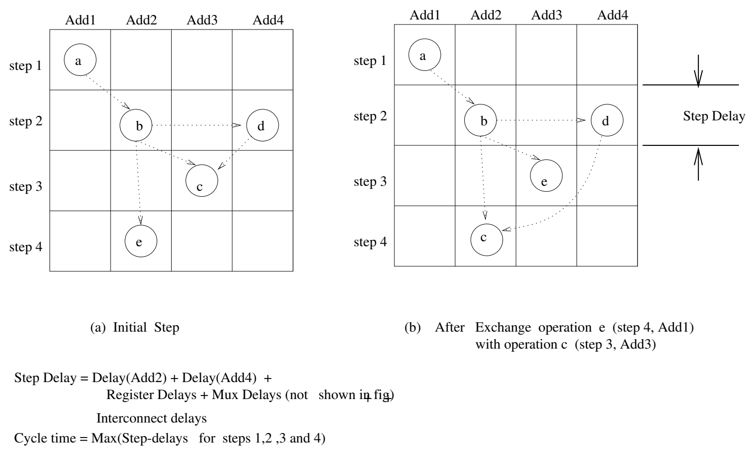



Figure 6: Representation of scheduling and binding problem as a placement problem in a two-dimensional table

the nominal values of $R_{tr}$ and $C_L$ given in [33].

# 5 The Combined Scheduling,Binding and Floorplanning Algorithm

The combined scheduling,binding and floorplanning is a simulated annealing based algorithm. It transforms the combined scheduling and binding into a placement problem in a two-dimensional table [10]. The rows represent the steps and columns represent functional units as shown in Figure 6. There are two types of moves:

- Move an operation from one position in the table to another
- Interchange two operations

Each move is performed only if the precedence constraints are satisfied, and at most one operation is assigned to a functional unit. Two operations are chained together in a step if there is a precedence constraint from one of them to the other and the latter operation is an immediate successor of the former. Each move is evaluated by performing our fast, constructive power driven floorplanning algorithm, and calculating the power, the area of the floorplan and latency of the schedule based on that. The overall interconnect energy dissipation is estimated by summing up the product of the capacitance and the switching across all links. The latency of the schedule is calculated as the product of the number of cycles and the cycle delay, where the cycle delay is the delay of the longest step, which includes the functional unit delays, register and mux delays and the interconnect delays between modules. The interconnect delays between two

POWER-DRIVEN-SCHEME()
COMPUTE-START-TEMPERATURE()
**while** $T \geq 0.1$ and not converged
  **do for** $k \leftarrow 1 to$ moves-per-temp
    **do** GENERATE-MOVE ()
      ALLOCATE-REGISTERS ()
      ALLOCATE-MUXES()
      CONSTRUCT-FLOORPLAN()
      ESTIMATE-INTERCONNECT-ENERGY()
      CALCULATE-LATENCY()
      $C \leftarrow \alpha P + \beta L + \gamma A$
      **if** $\Delta C < 0$
        **then** ACCEPT-MOVE()
        **else** Accept with prob $e^{-\frac{\Delta C}{T}}$
  $T \leftarrow T \times 0.9$

Figure 7: Simultaneous scheduling,binding and floorplanning

modules is calculated based on their positions in the floorplan constructed above. The overall algorithm is as shown in Figure 7.

The initial temperature is determined by finding the average change in cost for a set of random moves from the starting configuration and selecting the temperature which leads to an accept probability of $0.95$. The number of moves per temperature is set as a constant (typically 20 to 30) times the number of operations in the signal flow graph.

## 5.1 Cost Function

The cost function is a linear function of the power, latency and the area of the solution.

$$C = \alpha P + \beta L + \gamma A$$

where the interconnect energy estimate $P$ is as given by the power estimation algorithm, latency $L$ is the product of the number of cycles and the delay of the longest cycle, where the delay is the longest path in a step including the latencies of functional units, mux delays, register delays and the interconnect delays, and area $A$ is given by the layout generated by the constructive floorplanning algorithm.

### 5.2 Register and Multiplexor Allocation

For simplicity, we assume that there is a set of registers associated with each module. This set can have size zero if the module does not need to store any values, in which case, it will be collapsed before area calculation. The number of registers for any given module is determined by calculating the maximum clique size of an interval graph representing the lifetimes of values computed by the module. This is done efficiently using the left edge algorithm for non-hierarchical graphs with no loops.

We assume that there is a $k$ to 1 multiplexor at each input of the modules, where $k$ is the number of different modules feeding into that input. If $k$ has value 1, that multiplexor need not exist and will be collapsed before area calculation.

## 6 Experimental Results

We conducted experiments on a set of high-level synthesis benchmarks used by other researchers, which include a 2-step FIR filter, 3-step FIR filter and a 8-step elliptic filter. In the following, 'power-driven scheme' refers to our simultaneous scheduling, binding and floorplanning algorithm which considers interconnect energy, and 'latency-driven scheme' refers to a simulated annealing algorithm which simultaneously performs scheduling, binding and floorplanning with the objective of minimizing latency, but ignoring power [13]. We use two library sets for functional modules as in [12]. The libraries are as shown in Table 1. Tables 2,3 and 4 shows the comparison between the designs generated by the power-driven scheme and the latency-driven scheme. The energy estimate is the sum of the product of switching and interconnect lengths over all interconnect links (denoted by $SC$). The results show that the power-driven scheme gives better results in almost all cases when compared to the latency-driven algorithm. In many cases, it was able to reduce the interconnect energy dissipation by more than a factor of two, at the expense of increased latency and minimal impact on area.

## 7 Conclusion

With small device features in sub-micron technologies, since the interconnect power is a major component of the power consumed by the circuit, it is important to consider the effect of physical design on high-level synthesis. We have pre-

Table 1: The library set

| Library 1 | | | | | | |
|---|---|---|---|---|---|---|
| Fabrication | 16 bit adder | | 16 bit multiplier | | 16 bit register | |
| Technology | Delay | Area | Delay | Area | Delay | Area |
| 1.6 $\mu$m | 18 | 746875 | 200 | 8711250 | 3 | 597500 |
| 1.2 $\mu$m | 13 | 420000 | 150 | 4900000 | 3 | 336000 |
| Library 2 | | | | | | |
| Fabrication | 8 bit adder | | 8 bit multiplier | | 8 bit register | |
| Technology | Delay | Area | Delay | Area | Delay | Area |
| 1.2 $\mu$m | 13 | 46875 | 32 | 564375 | 2 | 37500 |
| 1.0 $\mu$m | 12 | 21250 | 30 | 261875 | 1.5 | 17000 |

Table 2: Comparison between latency-driven and power-driven schemes for 2-step FIR

| Fabrication Technology $\mu$m | latency-driven | | | power-driven scheme | | | |
|---|---|---|---|---|---|---|---|
| | Area $mm^2$ | Latency $ns$ | Energy Est ($SC \times 10^{-4}$) | Area $mm^2$ | Latency $ns$ | Energy Est ($SC \times 10^{-4}$) | Energy Reduction (%) |
| 1.6(16 bit) | 57.19 | 574.39 | 1774.02 | 56.48 | 691.64 | 1224.75 | 30.96 |
| 1.2(16 bit) | 31.37 | 430.47 | 1298.83 | 31.24 | 545.06 | 902.08 | 30.55 |
| 1.2(8 bit) | 3.397 | 183.76 | 411.05 | 3.27 | 212.59 | 306.46 | 25.44 |
| 1.0(8 bit) | 1.59 | 167.23 | 314.12 | 1.58 | 193.91 | 221.02 | 29.64 |

Table 3: Comparison between latency-driven and power-driven schemes for 3-step FIR

| Fabrication Technology $\mu$m | latency-driven | | | power-driven scheme | | | |
|---|---|---|---|---|---|---|---|
| | Area $mm^2$ | Latency $ns$ | Energy Est ($SC \times 10^{-4}$) | Area $mm^2$ | Latency $ns$ | Energy Est ($SC \times 10^{-4}$) | Energy Reduction(%) |
| 1.6 (16 bit) | 47.12 | 753.7 | 3437.90 | 46.84 | 914.57 | 1459.08 | 57.56 |
| 1.2 (16 bit) | 26.24 | 568.6 | 2071.82 | 26.02 | 755.42 | 806.67 | 61.06 |
| 1.2 (8 bit) | 2.798 | 198.99 | 482.03 | 2.73 | 297.9 | 287.98 | 40.26 |
| 1.0 (8 bit) | 1.26 | 179.47 | 473.79 | 1.39 | 309.06 | 218.72 | 53.83 |

Table 4: Comparison between latency-driven and power-driven schemes for 8-step Elliptic filter

| Fabrication Technology $\mu$m | latency-driven | | | power-driven scheme | | | |
|---|---|---|---|---|---|---|---|
| | Area $mm^2$ | Latency $ns$ | Energy Est ($SC \times 10^{-4}$) | Area $mm^2$ | Latency $ns$ | Energy Est ($SC \times 10^{-4}$) | Energy Reduction(%) |
| 1.6 (16 bit) | 15.6 | 371.19 | 1387.16 | 15.85 | 565.09 | 574.05 | 58.61 |
| 1.2 (16 bit) | 8.91 | 284.90 | 1054.41 | 8.03 | 448.07 | 397.01 | 62.35 |
| 1.2 (8 bit) | 0.896 | 256.1 | 387.03 | 0.81 | 388.91 | 173.14 | 55.26 |
| 1.0 (8 bit) | 0.42 | 229.4 | 291.29 | 0.41 | 462.47 | 284.77 | 2.24 |

sented a new algorithm which combines physical design and high level synthesis with the objective of minimizing interconnect energy dissipation. Experimental results show that the approach is quite effective in reducing the interconnect energy dissipation in addition to layout area and latency of the schedule.

# References


[1] J. Crenshaw and M. Sarrafzadeh, "Accurate High Level Datapath Power Estimation," in *European Design and Test Conference*, 1997.

[2] A. Raghunathan and N. K. Jha, "An Interative Improvement Algorithm for Low Power Data Path Synthesis," in *International Conference on Computer-Aided Design*, 1995.

[3] A. Raghunathan, and N. K. Jha, "Behavioral Synthesis for Low Power," in *International Conference on Computer Design*,1994.

[4] A. Das and R. Karri, "Simultaneous Scheduling and Binding for Power Minimization During Microarchitecture Synthesis", in *International Symposium on Low Power Design,* 1995.

[5] K. Chao and D. F. Wong, "Low Power Considerations in Floorplan Design", in *International Workshop on Low-Power Design, 1994*.

[6] P. E. Landman and J. M. Rabaey, "Power Estimation for High-level Synthesis", in *European Design Automation Conference*, 1993.

[7] A. Chandrakasan, M. Potkonjak, J. Rabaey and R. Brodersen, "HYPER-LP: A system for power minimization using architectural transformations", in *International Conference on Computer-Aided Design 1992*.

[8] M. Pedram, "Power Minimization in IC Design:Principles and Applications", in *ACM Transactions on Design Automation of Electronic Systems*, 1996.

[9] C. Svenson and D. Liu,"A Power Estimation tool and Prospects of Power Savings in CMOS VLSI chips", in *International Workshop on Low-Power Design*, 1994.

[10] S. Devadas and A. R. Newton, "Algorithms for Hardware Allocation in DataPath Synthesis," *IEEE trans. on CAD*, vol. 8, No. 7, 1989

[11] D. F. Wong and C. L. Liu, "A New Algorithm for FloorPlan Design," in *Proc. of 23rd DAC,* 1986.

[12] Y. Fang and D. F. Wong, "Simultaneous Functional Unit Binding and Floorplanning," *Proc. of International Conference on Computer-Aided Design*, pp. 317-321, 1994.

[13] P. Prabhakaran and P. Banerjee, "Simultaneous Scheduling,Binding and Floorplanning in High-level Synthesis," *International Conference on VLSI Design*", 1998.

[14] P. Prabhakaran and P. Banerjee, "Simultaneous Scheduling,Binding and Floorplanning in High-level Synthesis," *Technical Report CRHC-97-16*,1997.

[15] T. Yamanouchi, K. Tamakashi and T. Kambe, "Hybrid Floorplanning Based on Partial Clustering and Module Restructuring", *International Conference on Computer Aided Design*", 1996.

[16] V. Moshnyaga, H. Mori, H. Onodera, K. Tamary, "Layout-Driven Module Selection for Register-Transfer Synthesis od Submicron ASICs," *Proc. of International Conference on Computer-Aided Design*, pp. 100-103, 1993.

[17] J. Weng and A. C. Parker, "3D Scheduling : High-Level Synthesis with Floorplanning," *Proc. of 28th DAC, pp. 668-673, 1991*.

[18] D. W. Knapp, "Fasolt: A Program for Feedback-Driven Data-Path Optimization," *IEEE Trans. on CAD*, vol. 11, No. 6, pp. 677-695, 1992.

[19] M. C. McFarland, "Using Bottom-Up Design Techniques in the Synthesis of Digital Data-Path Optimization," *Proc. of 23rd Design Automation Conference, pp. 474-480, 1986*.

[20] F. Brewer and D. Gajski, "Chippe: A system for constraint driven Behavioral Synthesis," *IEEE Trans. on CAD*, Vol. 9, No. 7, pp. 681-695, 1990.

[21] E. Girczyc, "Automatic Generation of Microsequenced Data Paths to Realize ADA Circuit Descriptions," *PhD thesis, Carleton University, July 1984*

[22] W. Dai and E. S. Kuh, “Simultaneous Floor Planning and Global Routing for Hierarchical Building Block Layout,” *IEEE trans on CAD, 1987*

[23] D. P. La Potin, and S. W. Director, “Mason: A Global Floorplanning Approach for VLSI Design,” *IEEE trans on CAD, 1986*

[24] L. StockMeyer, “Optimal Orientations of Cells in Slicing Floorplan Designs”, *Information and Control, Vol 59*,(1983), 510-522.

[25] R. H. J. M. Otten, “Automatic Floorplan Design,”, *Proc. Design Automation Conference 82*, pp 261-267.

[26] R. H. J. M. Otten, “Efficient Floorplan Optimization,”, *Proc. International Conference on Computer Design 83*, pp 96-98.

[27] D. F. Wong and P. S. Sakhamuri, “Efficient Floorplan Area Optimization”, *Design Automation Conference* 1989

[28] T. W. Wang and D. F. Wong, “An Optimal Algorithm for Floorplan Area Optimization”, *Design Automation Conference* 1990

[29] R. Nair, C. L. Berman, P. S. Hauge and E. J. Yoffa, “Generation of Performance Constraints for Layout,” *IEEE Trans. Computer-Aided Design*, 1989.

[30] A. Mujumdar, M. Rim, R. Jain and R. De Leone, “BITNET: An Algorithm for Solving The Binding Problem,” *International Conference on VLSI Design*, pp. 668-673, 1991.

[31] H. Jang and B. Pangrle, “A Grid-Based Approach for Connectivity Binding with Geometric Costs,” *Proc. of International Conference on Computer-Aided Design*, pp. 94-99, 1993.

[32] H. B. Bakoglu, “Circuits, Interconnections and Packaging for VLSI,” pp. 194-225, 1990.

[33] D. Zhou, F. P. Preparata and S. M. Kang, “Interconnection Delay in Very High-Speed VLSI,” *IEEE Trans. on Circuits and Systems*, Vol 38, No. 7, pp.779-790, 1991.