Module Assignment for Low Power*

Jui-Ming Chang and Massoud Pedram
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089

Abstract

In this paper, we investigate the problem of minimizing the total power consumption during the binding of operations to functional units in a scheduled data path with functional pipelining and conditional branching for data intensive applications. We first present a technique to estimate the power consumption in a functionally pipelined data path and then formulate the power optimization problem as a max-cost multi-commodity flow problem and solve it optimally. Our proposed method can augment most high-level synthesis algorithms as a post-processing step for reducing power after the optimizations for area or speed have been completed. An average power savings of 28% has been observed after we apply our method to pipelined designs that have been optimized using conventional techniques.

1 Introduction

Low power has become a primary concern for the class of portable computer and consumer electronic devices as well as wireless communications and imaging systems. It has thus become necessary to develop estimation and optimization techniques that help achieve low power in these systems. This is a challenging task that requires power modeling, estimation and minimization at all levels of design abstraction from system and behavioral down to logic and layout levels. This paper focuses on the behavioral level.

The behavioral synthesis process consists of three phases: allocation, assignment and scheduling. These processes determine how many instances of each resource are needed (allocation), on what resources a computational operation will be performed (assignment) and when it will be executed (scheduling) [GaDu92] [Stok91] [DeMi94]. Traditionally, behavioral synthesis attempts to minimize the number of resources to perform a task in a given time or minimize the execution time for a given set of resources. With the increasing demand for low power circuits, it has become necessary to modify the three phases of the behavioral synthesis process to minimize the power dissipation.

A number of researches have addressed the problem of minimizing power dissipation during module allocation and binding [RaJh94], scheduling, register allocation and binding [RaJh94][ChPe95a] and by trading off area for power through pipelining or parallelization combined with voltage scaling [GoOr94] [ChPo92]. The work of [ChPe95a] describes a single-commodity network flow solution for the register assignment in a non-pipelined data path. Of particular relevance to the present work is however the work of [RaJh94] where the authors describe a heuristic module binding scheme for low power based on iterative improvement of some initial solution. In contrast, we address the optimization problem in a functionally pipelined data path with conditional branching under arbitrary input statistics and our solution technique is provably optimal without increasing the controller and multiplexor cost or the circuit delay. Our proposed method can also work with most high-level synthesis algorithms (i.e., those which perform scheduling before resource allocation and binding) as a post-processing stage for reducing power optimization after the optimizations for area or speed have been completed.

The paper is organized as follows. Section 2 provides some terminology and gives an overview of the proposed algorithm. In Section 3, we describe our switching activity calculation procedure. Section 4 describes our power model and the module binding problem. Section 5 casts the problem as a multi-commodity flow problem. Experimental results are reported in Section 6 while further discussions, future extensions and concluding remarks are given in Section 7 and 8, respectively.

2 Terminology and Overview

This paper assumes a data flow graph (DFG) with conditional branches and functional pipelining that has already been scheduled with a scheduling algorithm (such as the feasible-scheduling in Sehwa [PaPa88]). The scheduling algorithm takes the data flow graph and the given latency (the number of time steps between two consecutive initiations of the data flow algorithm) and produces a feasible schedule subject to constraints on the total number of available modules (functional units) of each type.

The resulting information can be compactly represented in a *basic allocation table (AT)*. In this table, rows represent functional units (operators) and columns represent c-steps. A *c-step* refers to a group of concurrent time steps across different pipeline initiations. For example, if a data flow graph is scheduled with a latency of 3, then c-step 1 in the associated AT represents time steps 1 and 4 in the original data flow graph while c-step 2 in that AT represents time steps 2 and 5 in the DFG, etc. Furthermore, we annotate each operation in the table with its initiation index (shown as a superscript on each operation).

In the previous work [PaPa88], after scheduling and allocation of the functionally pipelined data path, the main optimization tasks on the data path are complete as the circuit speed and the hardware resources have already been determined. Although the binding of different operations of the compatible type to a set of functional units *(FU's)* is not yet determined, to a first order, this binding does not

[*] This research was supported in part by SRC under contract no. 94-DJ-559 and ARPA under contract no. F33615-95-C-1627.

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00 ©1996 IEEE

have much of an impact on the circuit speed or area (see Section 7 for a detailed discussion). The binding however has an important effect on the power dissipation as is explained next. Operations that are assigned to a FU in some c-step may be permuted with other compatible operations that are scheduled in the same c-step but are assigned to a different FU. This permutation may have a big effect on the power consumption in the functional units as it changes the sequence of data values going through each functional unit in the data path, thus influencing the switching activity at the inputs of the functional unit.

In a functionally pipelined data path, for a given latency L, there are L c-steps in the allocation table. Operations which share the same FU across consecutive c-steps form a directed cycle of L vertices starting with some vertex in c-step 1 and ending with the same vertex in c-step $L + 1$. The total switching activity across all cycles can be determined once we select specific permutations of compatible entries in each c-step (cf. Section 3). The problem of minimizing total switching activity is then equivalent to finding the optimal permutation of the entries in each column of the allocation table. This problem can be formulated as a max-cost multi-commodity network flow problem and solved optimally (cf. Section 5). Since latency of most problems is small even when the data flow graph has a large numbers of levels [1], exhaustive search is also possible. Experimentally, we have observed that the ratio of power consumption in a data path between the power-minimal binding and the area-minimal binding is an average of 0.72.

Our method can be used in conjunction with all other techniques aimed to optimize power consumption at the system or behavioral levels of design. For example techniques of Hyper-LP[ChPo92] that permit a reduction of V_{dd} (such as pipelining or parallelism) may be augmented with the technique developed in this paper to further reduce the power consumption without increasing chip area caused by additional multiplexors or increased controller complexity.

3 Switching Activity Calculation

Consider two operations that share some FU consecutively (that is there is no other operation that uses this FU in between). Let the two ordered operands for the first operation be x_1 and y_1, the two ordered operands for the second operation be x_2 and y_2, and the outputs of the two operations be z_1 and z_2, respectively. The switching activity at the inputs of the FU for executing these operations is given by:

$$sw^{FU}(op_1, op_2) = \sum_{(x_1,x_2)\in\mathcal{E}} f_{x_1x_2}(x_1,x_2)\cdot H(x_1,x_2) + \sum_{(y_1,y_2)\in\mathcal{F}} f_{y_1y_2}(y_1,y_2)\cdot H(y_1,y_2) \qquad (1)$$

where $f_{x_1x_2}(x_1,x_2)$ is the (word-level) joint probability density function (pdf) [Papo91] of variables x_1 and x_2 and $H(x_1,x_2)$ is the Hamming distance of the binary representations of x_1 and x_2, sets $\mathcal{E}$ and $\mathcal{F}$ are the $legal\ \ sets$ (domains) of pairs (x_1,x_2) and (y_1,y_2), respectively.

In a functionally pipelined data path, the same algorithm (data flow graph) is initiated every L time steps. We can associate with each different instance of the data flow graph an initiation index. Similarly, the arcs in the data flow graph can also be indexed by an integer tag associated with their data flow instance. Note that the internal arcs of each data flow graph can be converted into only functions of the primary inputs indexed by their initiation index.

Consider a data flow graph with 4 primary inputs a,b,c and d. Suppose we want to calculate the switching activity between two operations $op_1^{(i)}$ and $op_2^{(i+2)}$ that belong to pipeline initiations i and $i+2$, respectively. (We use superscript of an operation to denote its pipeline initiation index). Furthermore, assume that the two ordered operands of op_1 are x_1 and y_1 while the two ordered operands of op_2 are x_2 and y_2 and the outputs of the two operations are z_1 and z_2, respectively. If $x_1 = a + b$, $y_1 = c - d$; $x_2 = a \times b$, y_2=c/d, then $x_1^{(i)}$=$a^{(i)} + b^{(i)}$, $y_1^{(i)} = c^{(i)} - d^{(i)}$; $x_2^{(i+2)}$=$a^{(i+2)} \times b^{(i+2)}$, $y_2^{(i+2)}$=$c^{(i+2)}/d^{(i+2)}$. To utilize equation (1), we must have the the joint pdf of the corresponding random variables. Here we create $v_1^{(i)} = x_1^{(i)}$=$a^{(i)} + b^{(i)}$ and $v_2^{(i)}$=$x_2^{(i+2)}$=$a^{(i+2)} \times b^{(i+2)}$; $u_1^{(i)}$=$y_1^{(i)}$ =$c^{(i)} - d^{(i)}$ and $u_2^{(i)}$=$y_2^{(i+2)}$=$c^{(i+2)}/d^{(i+2)}$; $w_1^{(i)} = z_1^{(i)}$ and $w_2^{(i)}$=$z_2^{(i+2)}$.

If a large number of primary input vectors, say N, is given, then we can calculate the sequence of vector pairs $(v_1^{(i)}, v_2^{(i)})$, $i = 1, 2, 3, \ldots$. If we assume that the sequences of primary inputs $< a^{(1)}, a^{(2)}, \ldots, a^{(N)} >$, $< b^{(1)}$, $b^{(2)}$, …, $b^{(N)} >$, etc. are identically distributed, then we can show that each one of the the intermediate sequences $< v_1^{(1)}$,$v_1^{(2)}$, …, $v_1^{(N)} >$, $< v_2^{(1)}$,$v_2^{(2)}$, …, $v_2^{(N)} >$, etc. is also identically distributed. We can therefore conclude that the time average can be used to approximate ensemble average, and using the classical frequency interpretation of probability, the joint pdf of v_1 and v_2, $f_{v_1v_2}(v_1,v_2)$ is approximated by calculating the frequency of occurrence of each (v_1,v_2) pair in the sequence [Papo91]. Similarly, the joint pdf of u_1 and u_2, $f_{u_1u_2}(u_1,u_2)$ and the joint pdf of w_1 and w_2, $f_{w_1w_2}(w_1,w_2)$ is calculated.

Consider two sequences of data values $< v_1^{(1)}$,$v_1^{(2)}$, …, $v_1^{(N)} >$ and $< v_2^{(1)}$,$v_2^{(2)}$, …, $v_2^{(N)} >$ applied to the inputs of a Mux. Switching activity at the inputs of the Mux is given by

$$sw^{Mux}(v_1,v_2) = sw(v_1) + sw(v_2)$$

$$sw(v_i) = \frac{1}{N-1}\sum_{t=1}^{N-1} H(v_i(t), v_i(t+1)) \qquad (2)$$

Calculation of the switching activity at the inputs of the Mux can be done concurrently with calculation of the joint pdf $f_{u_1u_2}(u_1,u_2)$. Using above procedures, we only need a single scan through the input vectors to obtain all of the switching activities and joint $pdf's$. The run time is thus proportional to the number of the primary input vectors.

We need the sequence of input vectors for the above switching activity calculation procedure to work. This sequence may be a "short" (in hundreds of vectors) sequence of typical data stream obtained by statistical sampling [BuNa93] or may be a "long" (in tens of thousands of vectors) obtained from a dynamic execution trace for a program/application data runs on the target chip. We make no

[1] This is also referred as turn around time or computation time of one input sample for the data flow graph.

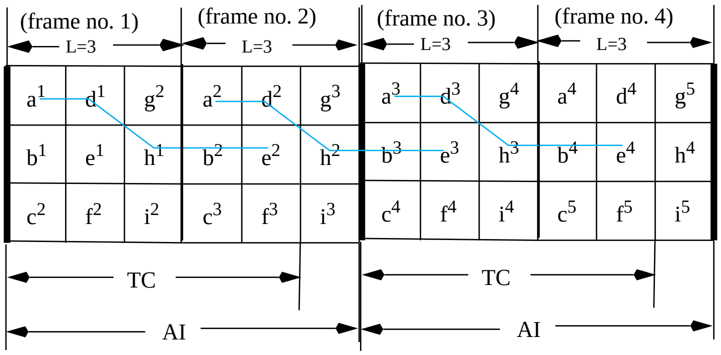

(frame no. 1)
(frame no. 2)
(frame no. 3)
(frame no. 4)
L=3
L=3
L=3
L=3
a1 d1 g2 a2 d2 g3 a3 d3 g4 a4 d4 g5
b1 e1 h1 b2 e2 h2 b3 e3 h3 b4 e4 h4
c2 f2 i2 c3 f3 i3 c4 f4 i4 c5 f5 i5
TC
TC
AI
AI

Figure 1: The definition of AI and TC. Note that the superscript on each operation denotes the pipeline initiation index (or data sample index)

assumption about the data statistics, hence, our technique is applicable to both DSP chips where the data tends to exhibit a pseudo-random white noise behavior and ASICs or general-purpose processor chips where the data may exhibit any probability distribution. Furthermore, we only assume that the sequence of input vectors is identically distributed. A similar assumption is commonly used in other fields like digital modulation, communication system performance evaluation and spectral analysis. This assumption is empirically justified and allows one to use formal methods to analyze the input streams [Gall68] [ViOm79].

4 Low Power Module Binding

4.1 Power dissipation of a functional unit

We assume that the dynamic power dissipation in a functional unit when it executes op_2 after op_1 is given by a simple equation as follows:

$$P_{FU} \;\; = 0.5 \cdot \alpha \cdot V^2 \cdot f \cdot sw^{FU}(op_1, op_2) \qquad (3)$$

where V is the supply voltage, f is the clock frequency, and the proportionality constant α (which represents the physical capacitance of the functional unit) is calculated for each functional unit using circuit or gate level simulation [Deng94][BuNa93] and curve fitting. Obviously this proportionality constant depends on the module type, input data width, technology and logic style used and internal module structure. Equation (3) is the basis of all macro-modeling techniques for power estimation and has been used in the works of [PoCh91][LaRa94][SvLi94] [MeRa94]. Power estimation accuracies of 10-15% have been reported for this model in [LaRa94] [ChPe95b].

4.2 A functionally pipelined data path

Data path is assumed to be functionally pipelined. If the operations that share the same functional unit belong to different pipeline initiations, then the joint pdf's of any two random variables belonging to different pipeline initiations will account for both spatial and temporal dependencies.

Definition 4.1 *In Fig. 1, suppose we have a sequence of operations* $[a^{2i-1} \rightarrow d^{2i-1} \rightarrow h^{2i-1} \rightarrow b^{2i} \rightarrow e^{2i}]$ *(for* $i = 1, 2, \ldots$*) that share the same functional unit. The time span of this sequence is defined as* TC*, which is equal to 5 time steps. The alignment interval of this sequence,* AI *is defined as the number of time steps required to allow the execution of this sequence periodically. In this example,* AI *is 6, which is equal to two (time) frames. We use time frame to refer to a sequence of* c*-steps of length* L *where* L *is the latency of the pipeline.*

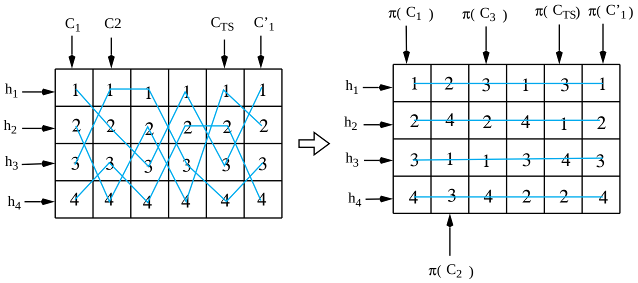


Figure 2: Valid binding resulting from permutation of the column entries in the AT

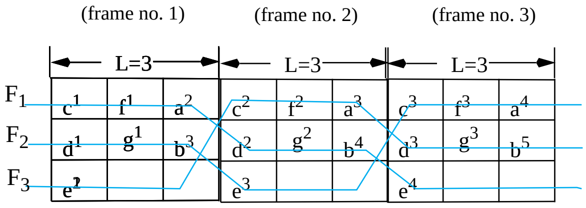


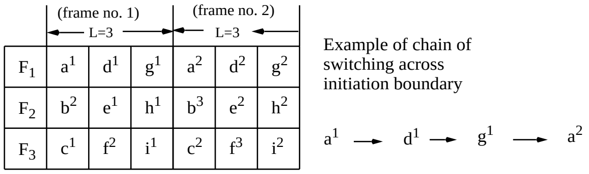
Figure 3: Three whole chains of 9 time steps composed of 3 optimal sub-chains. When we consider FU sharing within one time frame only, then the sharing solution within one frame (say frame # 1) will be replicated across all frames.

Lemma 4.1 *Consider a data flow graph which executes with latency* L *for the functional pipeline. Suppose there are* k *operations,* $op_1, \ldots, op_k$ *that share some functional unit consecutively, and that the time span (in number of time steps) of the sequence of operations is* TC*. To sustain this sequence of operations on the FU in question periodically across many frames in the pipeline, the alignment interval* AI *for this set of* k *operations must be an integer multiple of* L *which is larger than or equal to* TC*. Proof follows from definition of* AI*.*

In the past, most of the work on functionally pipelined data path has focused on the treatment of only a frame of length L. For the problem of sharing a FU among many operations which are scheduled in different c-steps, one should however consider a sharing chain longer than L as two operations which share a FU may belong to different c-steps in *different time frames* (cf. Fig. 3). One can easily show that this kind of sharing results in little reduction in power consumption, but creates large sharing chains and thus leads to large controller complexity and multiplexor cost which tend to offset power reduction due to sharing the FU across multiple frames. For this reason, we have decided to consider FU sharing only for operations in the same time frame.

4.3 The optimization problem

The problem is to find the power optimal way of binding operations to a set of compatible modules. Each binding solution corresponds to a permutation of the column entries of the AT. The solution also specifies multiple chains of operations where each chain denotes all operations that are consecutively executed on a functional unit. The total switching activity of the chain consists of the sum of switching



F_1	a^1	d^1	g^1	a^2	d^2	g^2
F_2	b^2	e^1	h^1	b^3	e^2	h^2
F_3	c^1	f^2	i^1	c^2	f^3	i^2

Figure 4: Boundary switching between two frames

L=3

F_1	a^1	d^1	g^1	a^2
F_2	b^2	e^1	h^1	b^3
F_3	c^1	f^2	i^1	c^2

Figure 5: Extended allocation table (EAT)

activities between two consecutive entries, plus the switching from the last entry of the row in the current frame to the first entry of the same row in the next frame of length L (see Fig. 4). The total binding cost is the sum of the costs of all chains. Because each operation is executed on exactly one module, these chains must be node disjoint.

Definition 4.2 *A basic allocation table consists of m rows corresponding to m compatible modules and L columns corresponding to L c-steps. An extended allocation table (EAT) is obtained by concatenating the first column of the basic allocation table after its rightmost column thus obtaining a new table with $L + 1$ columns where the first and the last columns are identical (cf. Fig. 5).*

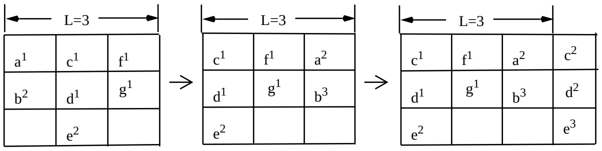
The optimization problem is then equivalent to finding the optimal way to permute the elements in each column (except the first and the last column in the EAT) for the rows corresponding to the set of compatible modules and minimizing the switching cost in all rows.

Definition 4.3 *The requirement that the $(i, 1)th$ entry of the EAT be equal to the $(i, L + 1)th$ entry of the table for $i = 1, \ldots, m$ will be referred as the* inter-frame binding constraint*. This condition is imposed to gaurranttee the cyclic nature of the execution on the functionally pipelined datapath without having to incur a large cost in terms of controller complexity and size of the MUX's.*

5 Network Flow Formulation

In Section 4.2, we described optimization of the total switching activity using the EAT. In the following, we cast the optimization problem as a *Max-Cost Multi-Commodity Flow* problem. Condition that makes the original problem hard is that we must meet the inter-frame binding constraint. Without this constraint, a simple *Max-Cost Flow* would give the optimal solution [ChPe95a].

Suppose there are m rows and n columns ($n = L + 1$) in the EAT. During scheduling and allocation of a functionally pipelined data path, we use the minimum possible number of modules and hence the feasible scheduling results in an allocation table which contains at least one full column. Suppose this column is at c-step i in the basic AT, we can rotate columns of this table until the full column occupies the first position in the table. Then we construct the EAT from the new AT by augmenting it with a rightmost column which contains the entries of the first column in the new AT in the next frame of length L. Fig. 6 shows an example of rotating a basic AT and obtaining the new EAT where first and last columns are full. If the first and the last columns in the EAT have empty entries, then the flow on the network which will be constructed from this EAT will no longer represent all of the feasible permutations. This is the reason for the EAT construction given above.



a^1	c^1	f^1
b^2	d^1	g^1
	e^2	

c^1	f^1	a^2
d^1	g^1	b^3
e^2		

c^1	f^1	a^2	c^2
d^1	g^1	b^3	d^2
e^2			e^3

Figure 6: Rotating the basic AT and obtaining the new EAT

Let N_H=$(V, E, s_1, \ldots, s_m, t_1, \ldots, t_m, C, K)$. There is a vertex $v \in V$ for each nonempty entry in the EAT (with m rows and n columns). We will refer to V as internal node set of N_H. We add a column of vertices s_i, $i = 1, \ldots, m$ in front of the vertices corresponding to the operations in the first column of the EAT. We also add a column of vertices t_i, $i = 1, \ldots, m$ after the vertices corresponding to the operations in the last column of the EAT. There is an arc connecting s_i to the vertex corresponding to the $(i, 1)th$ entry of the EAT and an arc connecting vertex corresponding to the $(i, n)th$ entry of the EAT to vertex t_i, $\forall\ i = 1 \ldots m$. The vertices are then levelized (with s_i vertices at layer 0 and t_i vertices at layer $L + 2$). There are arcs from all of the vertices in layer i to all of the vertices in layer $i + 1$, $\forall\ i = 1 \ldots n - 1$. Besides, there are arcs from all of the vertices at layer $i - 1$ to all of the vertices at layer $i + 1$ if there exists some empty entry in column i of the EAT, $\forall\ i = 1 \ldots n$. If there are empty entries in both columns $i - 1$ and i in the EAT, we add arcs from all vertices at layer $i - 2$ to vertices at layer $i + 1$ in N_H, and so on.

The capacity function K is 1 for every arc $e \in E$. The cost of all arcs incident on s_i or t_i is 0. All other internal arcs have cost $C(u,v) \geq 0$ which is equal to $H - \lfloor M \times sw^{FU}(op_u, op_v) \rfloor$, where M is used to scale the switching activity into an integer and H is a sufficiently large integer that makes the resulting costs for all of internal arcs satisfy the *triangular inequality*. [2] Let $\beta(\gamma)$ be the maximum (minimum) of $\lfloor M \times sw^{FU}(op_u, op_v) \rfloor$ over all u, v. H is any integer that satisfies $H \geq 2\beta - \gamma$. This is needed to ensure that the network flow algorithm covers all of the vertices in N_H. Fig. 7 shows the network constructed from the new EAT shown in Fig. 6. The demand function D is defined as follows: $D_i(v) = 0$, $\forall v \in V$ & $\forall i = 1, \ldots, m.$; $D_i(s_j) = \begin{cases} -1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$; $D_i(t_j) = \begin{cases} +1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$. This function captures the inter-frame binding constraint.

Definition 5.1 *[Leng90] Min-Cost Multi-Commodity Flow:*

- *Instances: A directed graph G=(V,E), edge capacity K(e) $\in R_+$ and edge costs C(e) $\in R$ for $e \in E$, demands $D_i(v) \in R$ for all vertices v $\in$ V and for m commodities $i = 1, \ldots, m$.*

[2] That is, $C(i,j) \geq 0$, $\forall$ (i,j) and $C(i,j)$+$C(j,k) \geq C(i,k)$, $\forall$ internal arcs (i,j),(j,k) and (i,k) (if the arcs exist).

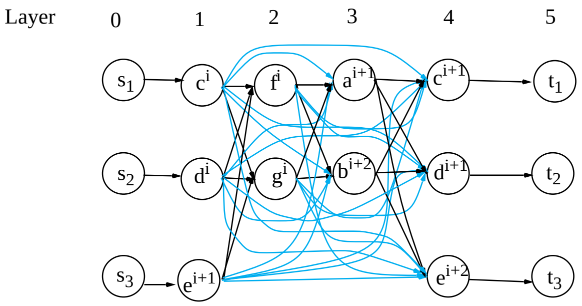


Figure 7: Network N_H construct from the new EAT shown in Fig. 6. Dark edges represent edges from level i to $i+1$ while light edges represent edges from level i to $j > i+1$

- *Configurations: All sequences of edge labeling* f_i*:* $E \rightarrow R_+$*, i=1,* $\ldots$*,* m
- *Solutions: All sequences of edge labeling that satisfy the following constraints:*
  1. *Capacity Constraints: For all* $e \in E$*,* $\sum_{i=1}^{m} f_i(e) \leq K(e)$
  2. *Flow-Conservation Constraints: Define the net flow of commodity i into vertex v to be* $f_i(v) = \sum_{e:\ u \rightarrow v} f_i(e) - \sum_{e:\ v \rightarrow w} f_i(e)$ *Then, for all* $i = 1, \ldots, m$ *and* $v \in V$*, we have* $f_i(v) = D_i(v)$
- *Minimize: C(f)=* $\sum_{e \in E}(C(e) \times \sum_{i=1}^{m} f_i(e))$

In [Stok91], a multi-commodity flow formulation of the register allocation problem in a cyclic data flow graph is proposed. We however proposed a multi-commodity flow formulation of the module allocation and binding in a functionally pipelined data flow graph. In addition to this, our network structure is quite different from that of [Stok91] due to empty entries that we have in the EAT which leads to cross-over edges (arcs that connect non-adjacent layers in the network). In [Stok91], the network has a node for each register at each time step and uses a dummy node for every time step boundary even when no variable has to be stored there. We could not minimize the total switching activity by simply replacing the cost on each arc in their network by the actual switching activity. The reason is that the switching activity to and from the dummy nodes could not be defined and there is no way to make the resulting network flow represent the total switching activity. In our network, we do not have any dummy nodes and each arc has its own (possibly) distinct cost. Eliminating the dummy nodes however creates the situation that some internal node may not be covered in the max-cost multi-commodity flow. We therefore enforce the triangular inequality on the costs of all arcs (and this is always doable) to ensure that the max-cost flow covers all of the internal vertices. This enforcement has no impact on the optimality of our solution as will be proved in Theorem 5.1.

Definition 5.2 *In the max-cost multi-commodity-flow problem, we maximize the total cost of flow* $C(f)$ *in the network while satisfying constraints 1 and 2 of definition 5.1.*

The extra vertices s_i and t_i, i = 1, . . . ,m, will serve as the sources and sinks of commodity i, respectively. We ship from s_i one unit of i and sink one unit of i at sink t_i. To ensure that the flow paths are node-disjoint, we apply a node splitting technique on N_H. After applying the node splitting on the internal node set of N_H, we obtain N'_H=(V', E', $s_1, s_2, \ldots, s_m, t_1, t_2, \ldots, t_m, C, K$). We will refer to V' as the internal node set of N'_H.

Module	α
add16	18.91
mult16	400.64
Mux16: 2 to 1	3.96
Mux16: 4 to 1	11.16

Table 1: Different α's obtained from simulation

The flow with value m on the new network N'_H gives m node disjoint paths; each path starting from source s_i and ending at sink t_i, for all i. We conduct the *max-cost* m-commodity flow on N'_H, which minimizes the total switching activity while satisfying the inter-frame binding constraint. The network formulation provides the exact solution to the original problem as shown by the following theorem.

Theorem 5.1 *A max-cost multi-commodity flow of value* m *on* N'_H *gives the minimum total power consumption for the* m *compatible modules in the circuit while satisfying the inter-frame binding constraint of Definition 4.3.*

Although our network is constructed differently from [Stok91], a similar method for solving the remaining step can be used after the multi-commodity network flow problem is translated into a LP. Since we are considering a functionally pipelined data path where the latencies of most pipeline designs are quite small, exhaustive search on the EAT can also give the optimal solution while meeting the inter-frame binding constraint in a very short time.

6 Experimental Results

In Table 1, we show values of α parameter for 16-bit adder, multiplier, 2 to 1 Mux and 4 to 1 Mux. This is all the data we need since our benchmark circuits have a datapath width of 16-bits and a latency of less than 4. Table 2 gives P_{total}, the total power dissipation in each circuit after scheduling, allocation and binding. Values of α in equation (3) are in unitrs of pF, while values of $sw^{FU}(op_1, op_2)$ are obtained from the binding solution as detailed in Section 4. Table 3 gives P'_{total} which is P_{total} + $P_{Mux's}$; Again α for the Mux is read from Table 1 while switching activity for Mux is calculated from the binding solution.

We performed feasible scheduling and our new method on various other benchmarks including an example taken from [PaPa88], AR Filter, Elliptical Wave Filter[GeEl92], 2nd order Adaptive Transversal Filter [Hayk91], Robotic Arm Controller, Differential Equation Solver [CaWo91], and Discrete Cosine Transform. Power dissipation results are given in Tables 2 and 3. Latency used in each benchmark is shown in the 2nd column of the tables. In our experiment, we also generate all possible minimal-area bindings from the same basic AT for each DFG using feasible scheduling algorithm [PaPa88]. From these tables, we can see that the ratios of P_{total} for minimum-power binding to maximum-power and average-power bindings are 56.88% and 70.64%, respectively. Even after including the power

Circuit	$L.$	max.pw	avg.pw	min.pw	$\frac{min}{max}$	$\frac{min}{avg}$
					%	%
Ex # 1	3	2.09e4	1.73e4	1.24e4	59.21	71.57
ARF	3	1.06e6	7.76e5	5.04e5	47.60	65.01
EWF	3	5.32e5	4.00e5	3.03e5	56.86	75.64
ATF2	2	3.10e5	2.61e5	1.75e5	56.55	67.13
Robo	2	1.21e6	1.07e6	5.53e5	45.49	51.40
DifE	2	3.16e5	2.68e5	2.04e5	64.68	76.31
FDCT	3	1.01e6	8.22e5	6.23e5	61.15	75.77
Avg.	-	-	-	-	56.88	70.64

Table 2: P_{total} (power dissipations (μW) in $FU's$), where L is the latency of the functional pipeline

Circuit	$L.$	max.pw	avg.pw	min.pw	$\frac{min}{max}$	$\frac{min}{avg}$
					%	%
Ex # 1	3	2.65e4	2.29e4	1.79e4	67.79	78.49
ARF	3	1.07e6	7.87e5	5.15e5	48.12	65.49
EWF	3	5.47e5	4.15e5	3.17e5	58.04	76.52
ATF2	2	3.14e5	2.65e5	1.79e5	57.08	67.61
Robo	2	1.22e6	1.08e6	5.65e5	46.05	51,97
DifE	2	3.19e5	2.71e5	2.07e5	64.96	76.53
FDCT	3	1.03e6	8.34e5	6.35e5	61.61	76.13
Average	-	-	-	-	58.97	72.07

Table 3: P'_{total} (power dissipations (μW) in $FU's$ + $Mux's$)

dissipation due to $Mux's$ (which our algorithm does not directly attempt to minimize), we obtained power saving ratios of 58.97% and 72.07%, respectively.

7 Discussion

The main idea is to permute the compatible operations during the module binding step so as to minimize the switching activity at the inputs of the functional units in a functionally pipelined design. This permutation takes place over a time frame of L c-steps, and therefore, the number of multiplexors doesn't vary by much; and in any case, the power dissipation in Mux's is significantly lower than that in functional units as seen by the relative magnitude of α parameters in Table 1. Hence, we can safely assume that P_{Mux} will vary minimally as a function of the binding solutions and when it does, it causes little change in P_{total}. This is also seen in Tables 2 and 3, where $\frac{P'_{total}}{P_{total}}$ is only 1.072 on average.

The number of registers (obtained by performing register allocation on a functionally pipelined data path by a program such as REAL [KuAl87]) will also not change as the life time of data values in the data flow graph will not change after the permutation (see Section 4.3). The circuit speed will not change as we only permute compatible operations in the same c-step (same column in the EAT) and the number of c-steps is not altered. The only impact of this permutation is to vary the interconnect structure of the design in some undetermined fashion (for worse or for better).

Extensions to handle multicycle operations, conditional branches and register binding can be found in [ChPe95b].

8 Conclusion

This paper presented a new method to calculate the switching activity of modules in a functionally pipelined data path with conditional branches based on the assumption that a large number of primary input data vectors are given. The power consumption model for a functionally pipelined data path was presented and its properties were explored. The power optimization problem was then formulated as a multi-commodity flow problem and solved optimally without increasing the area or delay of the data path or the controller complexity compared to the pipelined data path design before using our new method. Both techniques cover a general class of applications and are practical for larger problem sizes with complicated control flow. Experimental results demonstrate that the above methods can reduce the power consumption substantially.

References

[CaWo91] R. Camposanoand and W. Wolf, "High-level VLSI synthesis", PP. 256, Kluwer Academic Publishers, c1991.

[ChPo92] A. Chandrakasan, M. Potkonjak, J. Rabaey, and R.W. Brodersen,"HYPER-LP: A System for Power Minimization Using Architectural Transformations", In Proc. of the Int'l Con. on Computer Aided Design 1992.

[ChPe95a] J.-M. Chang and M. Pedram, "Low Power Register Allocation and Binding", In Proc. of the 32nd DAC, June, 1995.

[ChPe95b] J.-M. Chang and M. Pedram, "Power Efficient Module Allocation and Binding", CENG Technical Report 95-16, Computer Engineering Division, Dept. of EE-Systems, Univ. of Southern California, 1995.

[DeMi94] G. De Micheli, "Synthesis and Optimization of Digital Circuits", McGraw Hill 1994.

[Deng94] C. Deng, Power Analysis for CMOS/BiCMOS circuits. In Proceedings of the 1994 International Workshop on Low Power Design, pp. 3-8, April 1994.

[GaDu92] D. Gajski, N. Dutt, A. Wu, and Y.-L. Lin, "High-Level Synthesis, Introduction to Chip and System Design", Kluwer Academic Publisher, 1992.

[Gall68] R. Gallager, "Information Theory and Reliable Communication", Chap. 2, Wiley, New York, 1968.

[GeEl92] C. Gebotys and M. Elmasry, "Optimal VLSI Architectural Synthesis", pp. 148, Kluwer Academic Publishcation, pp. 148, 1992.

[GoOr94] L. Goodby, A. Orailoglu and P.M. Chau,"A High-Level Synthesis Methodology for Low Power VLSI Design". In Proc. of the IEEE Symposium on Low Power Electronics, 1994.

[Hayk91] S. Haykin, "Adaptive Filter Theory", 2nd edition, Chap5, 6, and 8, Printice Hall, 1991. Network Programming", John Wiley and Sons, 1980.

[KuAl87] Fadi Kurdahi, Alice Parker, "REAL: A Program for Register Allocation", In Proceeding of 24th Design Automation Conference, June 1987.

[LaRa94] P. Landman and J. Rabaey, "Black-Box Capacitance Models for Architectural Power Analysis", International Workshop Low Power workshop pp. 165-170, Napa Valley, CA, April 1994.

[Leng90] T.L. Lengauer, "Combinatorial Algorithms for Integrated Circuit Layout", John Wiley and Sons, 1990.

[MeRa94] R. Mehra and J. Rabaey. "Behavioral Level Power Estimation and Exploration.", In proceeding of the 1994 International Workshop on Low Power Design, pp. 197-202, Apr, 1994.

[BuNa93] R. Burch, F. Najm, P. Yang and T. Trick, "A Monte Carlo approach for power estimation". In IEEE Trans. on VLSI page 63-71, March 1993.

[PaPa88] N. Park, A. Parker,"Sehwa: A Software Package for Synthesis of Pipelines from behavioral Specifications", In IEEE Trans. on CAD, vol 7, no. 3, pp. 356-371, March 1988.

[Papo91] Athanasios Papoulis, Probability, Random Variables, and Stochastic Processes. Third Edition, section 6.3.

[PoCh91] R. Powell and M. Chau, "A Model for Estimating Power Dissipation in a Class of DSP VLSI Chips", In IEEE Trans. on Circuits and Systems, Vol 38, No. 6, pp. 646-650, June 1991.

[RaJh94] A. Raghunathan and N. Jha, "Behavioral Synthesis for Low Power", In Proc. Int'l Conf. on Computer Design 1994.

[Stok91] L. Stok, "Architectural Synthesis and Optimization of Digital Systems", Ph.D Dissertation, Eindhoven University of Technology, 1991.

[SvLi94] C. Svensson and D. Liu, "A Power Estimation Tool and Prospects of Power Savings in CMOS VLSI Chips". In Proc. of the 1994 International Workshop on Low Power Design, pp. 171-176, Apr 1994. of Data Paths in Digital Systems," IEEE Trans. on CAD, vol CAD-5, no.3, pp. 379-395, July, 1986.

[ViOm79] A. Viterbi and J. Omura, "Principle of Digital Communication and Coding", McGraw-Hill, New York, 1979.