Register Allocation and Binding for Low Power*

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

Abstract

This paper describes a technique for calculating the switching activity of a set of registers shared by different data values. Based on the assumption that the joint pdf (probability density function) of the primary input random variables is known or that a sufficiently large number of input vectors has been given, the register assignment problem for minimum power consumption is formulated as a minimum cost clique covering of an appropriately defined compatibility graph (which is shown to be transitively orientable). The problem is then solved optimally (in polynomial time) using a max-cost flow algorithm. Experimental results confirm the viability and usefulness of the approach in minimizing power consumption during the register assignment phase of the behavioral synthesis process.

1 Introduction

One driving factor behind the push for low power design is the growing class of personal computing devices as well as wireless communications and imaging systems that demand high-speed computations and complex functionalities with low power consumption. Another driving factor is that excessive power consumption is becoming the limiting factor in integrating more transistors on a single chip or on a multiple-chip module. Unless power consumption is dramatically reduced, the resulting heat will limit the feasible packing and performance of VLSI circuits and systems.

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 resource a computational operation will be performed (assignment) and when it will be executed (scheduling). Traditionally, behavioral synthesis aims to minimize the number of resources required to perform a task in a given time or to minimize the execution time for a given set of resources. It has become necessary to develop behavioral synthesis techniques that also account for power dissipation in the circuit.

*This research was supported in part by the NSF's Young Investigator Award under Contract No. MIP-9457392 and a grant from the Intel Corp.

This extends the two-dimensional optimization problem to a third dimension. The three phases of the behavioral synthesis process must be thus modified to produce low power circuits. Unfortunately, power dissipation is a strong function of signal statistics and correlations, and hence is non-deterministic.

Automatic techniques that minimize the switching activity on globally shared busses and register files, that select low power macros that satisfy the timing constraints, that schedule operations to minimize the switching activity from one cycle step to next, etc. must be developed. This paper considers register assignment for low power.

Most of the high-level synthesis systems perform scheduling of the control and data flow graph (CDFG) before allocation of the registers and modules and synthesis of the interconnect [11][18][7] as this approach provides timing information for the allocation and assignment tasks. Other systems perform the resource allocation and binding before scheduling, in order to provide more precise timing information available during the scheduling [9]. Either approach has its own advantages and shortcomings. The present work assumes that the scheduling of the CDFG has been done and performs the register allocation before the allocation of modules and interconnection.

During the register allocation and assignment, data values (arcs in the data flow graph) can share the same physical register if their life times do not overlap. In the past, researchers have proposed various techniques to reduce the total number of the registers used. The existing approaches include rule-based [6], greedy or iterative [10], branch and bound [13], linear programming [1], and graph theoretic, as in the Facet system [18], the HAL system [16] and the EASY system [17].

Power consumption of well designed register sets depends *mainly* on the *total switching activity* of the registers. In many applications, the data streams which are input to the circuit have certain probability distributions. Various ways of sharing registers among different data values thus produce different switching activities in these registers. This work presents a novel way of *calculating* this switching activity based on the assumption that the *joint pdf (probability density function)* of primary input random variables is known or a sufficiently large number of input vectors has been given. In the latter case, the joint pdf can be obtained by *statistical methods*. After obtaining the joint pdf of primary input variables, the pdf of any internal arc (data value) in the data flow graph and the joint pdf of any pair of arcs (data values) in the data flow graph are calculated by a method that will be described in detail in the follow-

ing sections. The switching activity on a pair of arcs is then formulated in terms of the joint pdf of these arcs, or alternatively, in terms of a function of the joint pdf of all primary input variables.

The *life time* of each arc (data value) in a scheduled data flow graph is the time during which the data value is active (valid) and is defined by an interval $[birth\_time, death\_time]$. A *compatiblity graph G(V,A)* for these arcs (data values) is then constructed, where vertices correspond to data values, and there is a directed arc *(u,v)* between two vertices if and only if their corresponding life times do not overlap and the u comes before v. We will show that the unoriented compatiblity graph for the arcs (data values) in a scheduled data flow graph without cycles and branches is a *comparability graph (or transitively orientable graph)* which is a *perfect graph* [5]. This is a very useful property, as many graph problems (e.g. maximum clique; maximum weight k-clique covering, etc.) can be solved in polynomial time for perfect graphs while they are *NP-complete* for general graphs.

Having calculated the switching activity between pairs of arcs that could potentially share the same register and given the number of registers that are to be used, the register assignment problem for minimum power consumption is formulated as a minimum cost clique covering of the compatibility graph. The problem is then solved optimally (in polynomial time) using a max-cost flow algorithm.

The two problems, calculation of the cross-arc switching activities (which must be performed $O(\mid E \mid)$ times, where $\mid E \mid$ is the number of edges in the compatibility graph) and power minimization during register assignment, are independent. The calculation of the cross-arc switching activities can be performed by any means. We present one such technique later. Other techniques may however be used. The power optimization is performed once the cross-arc switching activities are known.

The remainder of this paper is organized as follows: Section 2 shows the method to calculate the switching activity between pairs of data values (arcs). Section 3 shows the method to optimize the power consumption of registers in the register allocation phase in behavioral synthesis. Section 4 are some examples to demonstrate the methodology.

2 Switching Activity Calculation

2.1 Calculation of various pdfs

In many instances, the input data streams are *somewhat known*, and can be thus described by some probabilistic distributions. (Our proposed method applies not only to the well known probability distributions, such as joint Gaussian distribution, but also to *arbitrary probability distributions.*) Given a sufficient number of input vectors, it is possible to find the symbolic expressions for the pdf's and the joint pdf of all inputs using methods in *statistics*. For example, one way to do this is to calculate the frequency of the occurence for each vector among the set of input vectors, and then perform the interpolation on the sets of discrete points to obtain the symbolic expression of the joint pdf. Alternatively, one can work directly with the input vectors without having to find the symbolic expression of the joint pdf, that is, for a sufficiently large number of the input vectors, the *frequency of occurence* for each input vector can serve as the value of the joint pdf for that pattern.

If we are given the joint pdf of the input random variables of a data flow graph, then the joint pdf of *any pair* of values (arcs in the data flow graph) can be calcualted [15]. We want to find the joint pdf of any two arcs. Suppose that the two arcs are $y_1 = u_1(x_1, x_2, \ldots, x_n)$ and $y_2 = u_2(x_1, x_2, \ldots, x_n)$. We can add another $(n-2)$ free functions $y_3, y_4, \ldots, y_n$ and form a system of n equations in n input variables. Let's denote the joint pdf of the n input variables as $\psi(x_1, x_2, \ldots, x_n)$. If the inverse solution $x_1 = w_1(y_1, y_2, \ldots, y_n), x_2 = w_2(y_1, y_2, \ldots, y_n), \ldots, x_n = w_n(y_1, y_2, \ldots, y_n)$ can be obtained symbolically, then the joint pdf of $y_1, y_2, \ldots, y_n$ which is denoted by $\psi'(y_1, y_2, \ldots, y_n)$ is:

$$\begin{aligned}\psi'(y_1, y_2, \ldots, y_n) = \\ \mid J^{-1} \mid \times \psi[w_1(y_1, y_2, \ldots, y_n), \\ w_2(y_1, y_2, \ldots, y_n), \ldots, \\ w_n(y_1, y_2, \ldots, y_n)]\end{aligned}$$

where J^{-1} is the nxn inverse Jacobian:

$$J^{-1}(y_1, y_2, \ldots, y_n) = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} & \cdots \frac{\partial x_1}{\partial y_n} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} & \cdots \frac{\partial x_2}{\partial y_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial x_n}{\partial y_1} & \frac{\partial x_n}{\partial y_2} & \cdots \frac{\partial x_n}{\partial y_n} \end{vmatrix}$$

Once we have the $\psi'(y_1, y_2, \ldots, y_n)$,we can calculate the pairwise pdf of y_1 and y_2, $f_{y_1 y_2}(y_1, y_2)$, as

$$f_{y_1 y_2}(y_1, y_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \psi'(y_1, y_2, \ldots, y_n) \, dy_3 dy_4 \ldots dy_n.$$

The integration can be performed either symbolically or numerically. The numerical integration over $(n-2)$ variables involves much more computation, but is an alternative approach which is always possible whenever the symbolic integration over the $(n-2)$ variables is not possible.

In addition to the calculation of pairwise joint pdfs, the pdf of any internal arc is needed to calculate the total switching activity of the set of registers. Suppose function $y = w(x_1, x_2, \ldots, x_n)$ is some arc (data value) in the data flow graph depending on n input random variables $x_1, x_2, \ldots, x_n$. The cdf (cumulated distribution function) of the new random variable y is defined as $G(y) = \text{prob(Y} \leq \text{y)}$, which is equal to $\text{prob}(w(x_1, x2, \ldots, x_n) \leq y)$. The above probability can be evaluated as:

$$G(y) = \int \int \cdots \int_A \psi(x_1, x_2, \ldots, x_n)$$

where $\psi(x_1, x_2, \ldots, x_n)$ is the joint pdf of the n input random variables $x_1, x_2, \ldots, x_n$, and $A = \{(x_1, x_2, \ldots, x_n) \mid w(x_1, x2, \ldots, x_n) \leq y\}$. The pdf of y as $g(y)$ is then obtained by $g(y) = \frac{d\, G(y)}{dy}$.

Figure 1: Our register sharing model

2.2 The power consumption model

Switched capacitance refers to the product of the load capacitance and the switching activity of the driver. The power consumption of a register is proportional to the switched capacitance on its input and output (see Fig. 1). Suppose register R_1 can be shared between three data values i, j and k. We assume that an input multiplexor picks the value that is written into R_1 while an output demultiplexor dispatches the stored value to its proper destination. Now, $P(R_1) \propto switching(x) \times (C_{out,Mux} + C_{in,R_1}) + switching(y) \times (C_{out,R_1} + C_{in,DeMux})$. Since $switching(x) = switching(y)$, $P(R_1) = switching(y) \times C_{total}$. Note that C_{total} is fixed for a given library. In any case, minimizing the switching activity at the output of the registers will minimize the power consumption regardless of the specific load seen at the output of the registers. Here we ignore the power consumption internal to registers and only consider the external power consumption.

In the register allocation phase, if several compatible arcs are assigned to the same register R, the switching on R will occur whenever one stored data value is replaced by another data value. For example, suppose X,Y,Z and W are four compatible data values that share register R and the arcs $(X,Y),(Y,Z),(Z,W) \in A$. Suppose that in the beginning, the register was reset to some unknown value. We assume the switching activity from the unknown value to X is some constant value. Then the following is the *chain* of the data transitions $X \rightarrow Y \rightarrow Z \rightarrow W$. If the input variable values are known, then the exact switching activity is calculated as $constant + H(X,Y) + H(Y,Z)$ where $H(i,j)$ is the *Hamming distance* between two numbers i and j. If, however, the circuit has even one input random variable, the whole system has to be described in a probabilistic way as described next.

Assume that the n primary input random variables are $a_1, a_2, \ldots, a_n$ and set $\mathcal{A} = \{(a_1, a_2, \ldots, a_n)\}$ is the set containing all possible combinations of input tuples. Let set $\mathcal{B} = \{(x,y) \mid x = x(a_1, a_2, \ldots, a_n),\ y = y(a_1, a_2, \ldots, a_n),\ \forall (a_1, a_2, \ldots, a_n) \in \mathcal{A}\}$. The switching activity between the two consecutive data values X and Y is then given by:

$$switching(X,Y) = \sum_{(x,y)\in\mathcal{B}} f_{xy}(x,y) \times H(x,y) \qquad (1)$$

where the summation is over all possible patterns of $(x,y) \in \mathcal{B}$, and the function $H(x,y)$ is the *Hamming distance* between two numbers x and y which are represented in a certain number system in binary form. Equation (1) requires that the *discrete type* joint pdf for x, y be known. The method for calculating the joint pdf of two random variables described in section 2.1 is mainly suitable for the case when the variables in the system are of *continuous type*. When however the precision used to represent the discrete numbers is high enough or the variance of the underlying distribution is not too large, the continuous type pdf $g_{xy}(x,y)$ can be used as a good approximation for the discrete type pdf $f_{xy}(x,y)$ after being multiplied by the scaling factor $(\sum_{(x,y)\in\mathcal{B}} g_{xy}(x,y))^{-1}$.

The symbolic computation method is however not very practical because it involves the tasks of finding the *symbolic inverse* solution of *the system of nonlinear equations* and *symbolic or numerical integration* of complicated expressions over the region defined by a combination of inequalities and/or equalities. Fortunately, the same switching activity for a pair of *discrete* random variables x and y can be obtained much more easily by the following:

$$switching(X,Y) = \sum_{a_1}\sum_{a_2}\cdots\sum_{a_n} \psi(a_1, a_2, \ldots, a_n) \times H(x(a_1, a_2, \ldots, a_n), y(a_1, a_2, \ldots, a_n)) \qquad (2)$$

where $\psi(a_1, a_2, \ldots, a_n)$ is the joint pdf of the input variables $a_1, a_2, \ldots, a_n$.

Both equation (1) and equation (2) started from the assumption that the joint pdf $\psi(a_1, a_2, \ldots, a_n)$ is obtained or known. This is a necessary condition in order to precisely calculate the cross-arc switching activities. Furthermore, equation (2) can be used directly once the input vectors are given without obtaining the symbolic expression for $\psi(a_1, a_2, \ldots, a_n)$. Here we assume that the *bit_width* of a register is finite, so the total number of the patterns that can be stored in a register is also finite. If we assume all of the numbers in our system are integers (positive or negative), then the total number of different (x,y) pairs involved in equation (1) is at most $2^{2\times bit\_width}$. In general, equation (2) involves multidimensional nested summations over intervals of integral values. When the joint pdf of primary input variables is band-limited (e.g. Gaussian), we can narrow down the interval of summation in each dimension and thereby significantly speed up the computation.

Let's denote the set $\mathcal{A} = \{(a_1, a_2, \ldots, a_n)\}$, set $\mathcal{B} = \{(x,y) \mid x = x(a_1, a_2, \ldots, a_n),\ y = y(a_1, a_2, \ldots, a_n),\ \forall (a_1, a_2, \ldots, a_n) \in \mathcal{A}\}$, $\mathcal{C} = \{(y,z) \mid y = y(a_1, a_2, \ldots, a_n),\ z = z(a_1, a_2, \ldots, a_n),\ \forall (a_1, a_2, \ldots, a_n) \in \mathcal{A}\}$, and $\mathcal{D} = \{(z,w) \mid z = z(a_1, a_2, \ldots, a_n),\ w = w(a_1, a_2, \ldots, a_n),\ \forall (a_1, a_2, \ldots, a_n) \in \mathcal{A}\}$.

The total switching activity in the above example with register R shared by four arcs (data values) is formulated as follows:

$$\begin{aligned} constant &+ \sum_{(x,y)\in\mathcal{B}} f_{xy}(x,y) \times H(x,y) \\ &+ \sum_{(y,z)\in\mathcal{C}} f_{yz}(y,z) \times H(y,z) \\ &+ \sum_{(z,w)\in\mathcal{D}} f_{zw}(z,w) \times H(z,w) = \\ constant &+ \sum_{a_1}\sum_{a_2}\cdots\sum_{a_n} \psi(a_1, a_2, \ldots, a_n) \times (H(x,y) \\ &+ H(y,z) + H(z,w)) \end{aligned} \qquad (3)$$

The total switching activity for a register can be calculated after the the set of variables that share that register

are found. Note that the sequence of data transitions are known at that time.

3 Power Optimization

3.1 Max-cost flow formulation

Definition 3.1 *A directed graph $G_0 = (V,A)$ is called the compatibility graph for register allocation problem if the it is constructed by the following procedure. Each arc (data value) i in the data flow graph has an interval $(birth\_time_i, death\_time_i)$ associated with it. Each open interval i corresponds to a vertex i in $G_0 = (V,A)$. There is a directed arc $(u,v) \in A$ if and only if $interval_u \cap interval_v = \emptyset$ and $death\_time_u < birth\_time_v$.*

All proofs can be found in [2].

Theorem 3.1 *Given a data flow graph without loops and branches, the compatibility graph $G_0 = G(V,A)$ for register allocation problem is acyclic.*

Definition 3.2 *[5]An undirected graph $G = (V,E)$ is a comparability graph if there exists an orientation (V,F) of G satisfying*

$$F \cap F^{-1} = \emptyset, \quad F + F^{-1} = E, \quad F^2 \subseteq F$$

where $F^2 = \{(a,c) \mid (a,b),(b,c) \in F$ for some vertex b $\}$. Comparability graphs are also known as transitively orientable graphs and partially oderable graphs.

Definition 3.3 *The unoriented compatibility graph $G_0' = (V,E)$ is obtained by removing the edge orientations of $G_0 = (V,A)$.*

Theorem 3.2 *Given a data flow graph without loops and branches, the unoriented compatibility graph $G_0' = (V,E)$ for register allocation problem is a comparability graph.*

To minimize the total power consumption on the registers, a network $N_G = (s,t,V_n,E_n,C,K)$ is constructed from the compatibility graph $G_0 = G(V,A)$. This is a similar construction to the one used in [17] to solve the *weighted module allocation* problem which simultaneously minimizes the number of modules and the amount of interconnection needed to connect all modules. Conceptually, $N_G = (s,t,V_n,E_n,C,K)$ is constructed from $G_0 = G(V,A)$ with two extra vertices, the source vertex s and the sink vertex t. The additional arcs are the arcs from s to every vertex in V of $G(V,A)$, and from every vertex in V of $G(V,A)$ to t. We use the Max-Cost Flow algorithm on N_G to find a maximum cost set of cliques that cover the $G_0 = G(V,A)$. The network on which the flow is conducted has the cost function C and the capacities K defined on each arc in E_n. Assuming that each register has an unknown value at time t_{0-}, we use a constant sw_0 to represent the $switching(Unknown, v)$ for each vertex v. More formally, the network $N_G = (s,t,V_n,E_n,C,K)$ is defined as the following:

$$\begin{aligned} V_n &= V \cup \{s,t\} \\ E_n &= A \cup \{(s,v),\ (v,t) \mid v \in V\} \\ w(s,v) &= L - \lfloor sw_0 \times M \rfloor \end{aligned} \tag{4}$$

$$\begin{aligned} w(u,v) &= L - \lfloor \sum_{(u,v)\in\mathcal{B}} f_{uv}(u,v) \times H(u,v) \times M \rfloor \\ &= L - \lfloor \sum_{a_1}\sum_{a_2}\cdots\sum_{a_n} \psi(a_1,a_2,\ldots,a_n) \\ &\times H(u(a_1,a_2,\ldots,a_n), v(a_1,a_2,\ldots,a_n)) \rfloor \end{aligned} \tag{5}$$

$$w(v,t) = L, \ \forall\, v \in V,\ w(t,s) = L. \tag{6}$$

where $\mathcal{A} = \{(a_1,a_2,\ldots,a_n)\}$, $\mathcal{B} = \{(u,v) \mid u = u(a_1,a_2,\ldots,a_n),\ v = v(a_1,a_2,\ldots,a_n), \forall (a_1,a_2,\ldots,a_n) \in \mathcal{A}\}$, $L = \lfloor max\ \{switching(u,v)\} \times M \rfloor + 1$ over all possible u,v $\in V \cup \{s\}$, and M is a large constant used to scale up the smallest switching activity value to an integer.

For each arc e $\in E_n$, a cost function C: $E_n \rightarrow N$ is defined, which assigns a non-negative integer cost to each arc . The cost function C for network N_G is : $c(u,v) = w(u,v)$ for all $(u,v) \in E_n$. The cost function is defined to indicate the *power savings* on the arc.

For each arc $e \in E_n$, a capacity function K: $E_n \rightarrow N$, is defined that assigns to each arc a non-negative number. The capacity of all the arcs is one, except for the return arc from t to s which has capacity k, where k is user-specified flow value.

$$\begin{aligned} K(u,v) &= 1, \quad \forall (u,v) \in E_n \setminus \{(t,s)\} \\ K(t,s) &= k \end{aligned}$$

For each arc $e \in E_n$, a flow function f: $E_n \rightarrow N$ is defined which assigns to each arc a non-negative number. The flow $f(e)$ on each arc $e \in E_n$ must obey the following: $0 \leq f(e) \leq K(e)$ and the flow on each vertex $v \in V_n$ must satisfy the flow conservation rule.

Theorem 3.3 *A flow f: $E_n \rightarrow N$ with $\mid f \mid = 1$, in the network N_G corresponds to a clique χ in the unoriented compatibility graph G_0'.*

Theorem 3.4 *A flow f: $E_n \rightarrow N$, with $\mid f \mid = k$, in the network N_G corresponds to a set of cliques $\chi_1, \chi_2, \ldots, \chi_k$ in the unoriented compatibility graph G_0'.*

The generated cliques may not be vertex disjoint because the k paths in the N_G may not be vertex disjoint. One way to ensure that the resulting cliques are vertex disjoint is to employ a node-splitting technique. This technique duplicates every vertex $v \in V$ in the graph $G_0 = G(V,A)$ into another node v'. There is an arc from v to v' for each $v \in V$. If there is an arc $(u,v) \in A$ in the graph $G_0 = G(V,A)$, there is an arc (u',v) in the new network N_G'. There is also an arc from the source vertex s to every vertex $v \in V$ and from every duplicated vertex v' to the sink vertex t.

More formally, the node splitting technique generates the following network $N_G' = (s,t,V_n',E_n',C',K')$ where:

$$\begin{aligned} V_n' &= V_n \cup V_0' \\ &\quad there\ is\ a\ vertex\ v' = f(v) \in V_0' \\ &\quad for\ each\ vertex\ v \in V_0 \\ E_n' &= A' \cup \{(s,v),(f(v),t), v \in V_0\} \cup \{(t,s)\} \\ &\quad \cup \{(v,f(v) \mid v \in V_0\} \end{aligned}$$

Figure 2: From data flow graph to network N'_G

$$
\begin{aligned}
A' &= \{(f(u), v) \mid (u, v) \in A\} \\
C'((t,s)) &= C'((v, f(v)) = L, \ \forall \ v \in V_0 \\
C'((u',v)) &= C((u,v)) \ \ for \ \ all \ \ (u',v) \in A' \cup \{(s,v) \\
&\quad , (f(v),t) \mid v \in V_0\} \\
K'((t,s)) &= k, K'((u,v)) = 1 \ \ for \ \ all \ \ u \neq t, \\
&\quad and \ \ v \neq s.
\end{aligned}
$$

The transformations from the data flow graph to the final network N'_G are shown in Fig. 2.

Theorem 3.5 *A flow f: $E_n \rightarrow N$, with $\mid f \mid = k$, in the network N'_G corresponds to a set of* **vertex disjoint cliques** *$\chi_1, \chi_2, \ldots, \chi_k$ in the unoriented compatibility graph G'_0.*

Definition 3.4 *[14]Let N = (s,t,V,E,C,K) be a flow network with underlying directed graph G=(V,E), a weighting on the arcs $c_{ij} \in R^+$ for every arc (i,j) $\in$ E, a capacity K(e) for every arc e $\in$ E, and a flow value $v_0 \in R^+$. The min-cost flow problem is to find a feasible s-t flow of value v_0 that has minimum cost. In the form of an LP:*

$$
\begin{aligned}
min \ \ & c^t f \\
& Af = -v_0 d \ \ every \ \ node \\
& f \leq b \ \ every \ \ arc \\
& f \geq 0 \ \ every \ \ arc
\end{aligned}
$$

where A is the node-arc incidence matrix and

$$
d_i = \begin{cases} -1 & i = s \\ +1 & i = t \\ 0 & otherwise \end{cases}
$$

Definition 3.5 *The maximum cost flow problem is that given a network N=(s,t,V,E,C,K) and a fixed flow value v_0, find the flow that maximizes the total cost.*

The easiest method to solve the max-cost flow problem is to negate the cost of each arc in the network, and run the *min-cost flow algorithm* on the new network [14].

The previous network construction N'_G ensures that the resulting paths are vertex disjoint cliques in G_0 (or G'_0). When the *max-cost flow* algorithm is applied on this network,we obtain cliques that maximize the total cost. The flow value on each path is one, this implies that the total cost on each individual path is the sum over all individual arcs on that path according to their topological order in the graph $G_0 = G(V, A)$, where the cost on each arc is a linear function of the "Saved Power". For example, if (s, b), (b, c), (c, d), (d, t) is a path from source s to sink t. The total cost on this path is $cost(s, b) + cost(b, c) + cost(c, d) + cost(d, t)$. Also, from the above information, we can conclude that the set of variables $\{b, c, d\}$ will share the same register according to the order $b \rightarrow c \rightarrow d$.

Theorem 3.6 *The max-cost flow algorithm on the network N'_G gives the minimum total power consumption on the registers in the circuit represented by the compatibility graph G_0.*

Proof: The total cost is $\sum_{e \in E_n} f(e) \times c(e)$, which is a linear function of the *"Total Saved Power"*. The reason is that

$$
\begin{aligned}
\sum_{e \in E_n} f(e) \times c(e) &= \\
\sum_{e \in E_n} f(e) &\times [L - M \times switching(e)] = \\
L \times \sum_{e \in E_n} f(e) &- M \times \sum_{e \in E_n} f(e) \times switching(e)
\end{aligned}
$$

In our specially constructed network, $f(e)$ in every arc e except (t, s) has value either zero or one. The first term in the above, $\sum_{e \in E_n} f(e)$, is a constant ($= 2 \times \mid V \mid + k$ for $G_0 = G(V, A)$) among all possible clique coverings that cover all of the vertices in the original graph G_0. When we maximize the total cost for a given flow value in N'_G, we are indeed minimizing the total power consumption given that the number of registers is equal to this flow value. Note that, the max-cost flow on N'_G always finds the clique covering that covers all of the vertices in the original graph G_0 whenever the flow value $\mid f \mid$ is larger than or equal to k_{min}. k_{min} can be determined by the left edge algorithm [11] or simply by finding the maximum number of arcs that cross any c-step boundary. In most cases, the k_{min} found by the left edge algorithm is equal to the k_{min} for max-cost flow. However, in some pathological cases, the two values are not the same. In that case, a post-processing step is needed [2]. □

The time complexity for the max-cost flow algorithm is $\mathcal{O}(km^2)$, according to [4], where $m = 2\times \mid V \mid +2$ for the graph $G_0 = G(V, A)$ and k is the flow valus.

Conditional branches can be easily handeled in our system by relaxing the conditional data flow graph into several

Figure 3: The scheduled data flow graph.

We used equation (2) in Section 2.2 to calculate the cross-arc switching activities for every pair of arcs in $G(V, A)$.

The switching activity of for any variable x from time $= t_{0-}$ which is assumed to have some unknown value to the time that the variable gets its first value was taken to be a constant equal to $(1/5)\times[switching(0,a)+switching(0,b)+switching(0,c)+switching(0,d)+switching(0,e)]$.

After calculating the switching activities, we construct the max-cost flow network. The weight on each arc is calculated by equation (4)-(6) in Section 3.

Here we choose M = 1000, and so L = 10837. The following weights are obtained:

$w(x,x') = L,\ \forall x \in V;\ w(s,x) = 5271,\ \forall x \in V$ and other w's are given by the following table:

	f	g	h	i	j	k	t
a	4699	3315	2065	1906	2697	1233	10837
b	4687	3755	2102	1998	3066	1329	10837
c	3373	4599	2225	2291	2617	1703	10837
d	3811	4552	2258	2296	2708	1641	10837
e				2145	3884	2291	10837
f				679	2827	650	10837
g				3614	2418	2074	10837
h				2718	1	610	10837
i						2916	10837
j						1487	10837

Applying the max-cost flow on the network N'_G with the vertex splitting technique, the following results for number of registers, cliques and actual total switching activity are obtained:

No. of reg; cliques	tot. S.A.
7; $\{\{a,f\},\{c,g,i\},\{e,j\},\{b\},\{d\},\{h\},\{k\}\}$	65.514
6 ; $\{\{a,f\},\{c,g,i,k\},\{e,j\},\{b\},\{d\},\{h\}\}$	67.872
5 ; $\{\{a,f\},\{c,g,i,k\},\{d,h\},\{e,j\},\{b\}\}$	70.882

Note that our method finds the minimum power register assignment for the given number of registers.

To demonstrate that the switching activity calculation based on the joint pdf is necessary to obtain a low power register assignment we performed an experiment where every arc weight in the compatibility graph was set to some constant ($C = 100$) and then ran the max-cost flow for different flow values. For flow value 5, we obtained:

Number of registers is equal to 5,
cliques $= \{\{a,h,k\},\{b,f,i\},\{c,g,j\},\{d\},\{e\}\}$, actual total switching activity $= 80.487882$, which is 13.55% worse than the optimum solution.

Next, we generated register assignment solution using Real [11] which finds the minimum number of registers need (in this case) and obtained the following result:

Number of registers is equal to 5,
cliques $= \{\{a,f,i,k\},\{b,g,j\},\{c,h\},\{d\},\{e\}\}$, actual total switching activity $= 78.471$, which is 10.71% worse than the optimum solution.

Indeed, among all valid register assignment of given size, our proposed algorithm finds the one that minimizes the power consumption.

The percentage power reduction increases for larger data flow graphs. For example, we obtained 22.5% improvement in power (compared to the minimum register count register assignment procedure) on 7-input data flow graph using similar assumptions about the joint pdf and the data types. Specifically,

For the Min-Power Register Assignment, we obtained:

Number of registers is equal to 9, cliques $=$
$\{\{a,i\},\{b,h\},\ \{e,j,k\},\{l,m\},\{n\},\{c\},\{d\},\{f\},\{g\}\}$ and actual total switching activity $= 6.861$; Number of registers is equal to 8,
cliques $= \{\{a,l,m\},\{b,h\},\{e,j,k\},\{f,i\},\{c\},\{d\},\{g\},\{n\}\}$ and actual total switching activity $= 7.272$; Number of registers is equal to 7,
cliques $= \{\{a,l,m,n\},\{d,h\},\{e,j,k\},\ \{f,i\},\{b\},\{c\},\{g\}\}$ and actual total switching activity $= 7.763$.

For the Min-Count Register Assignment, we obtained:

Number of registers is equal to 7,
cliques $= \{\{a,j,m\},\{b,h,k,n\},\{c,i,l\},\{d\},\{e\},\{f\},\{g\}\}$ and actual total switching activity $= 10.017$.

5 Conclusion

This paper presented a novel way to *calculate* the switching activity external to a set of registers based on the assumption that the joint pdf of the primary input random variables is known or can be calculated. For a scheduled data flow graph without cycles, the compatibility graph for register allocation and assignment problem was proven to be a transitively orientable graph. A special network was then constructed from the above compatibility graph and the max-cost flow algorithm (a variation of min-cost flow algorithm) was performed to obtain the *minimum power consumption register assignment*. Due to properties of transitively orientable graph, the time complexity is polynomial. Our future work will focus on the register assignment for pipelined design and data flow graph with outer loops.

Acknowledgement

The first author acknowledges helpful discussions with Chin-Chi Hsu, Tung-Shen Lin and Kuo-Chun Lee from department of EE-systems at the University of Southern California.

References

[1] M. Balakrishnan and P. Marwedel. "Integrated Scheduling and Binding: A Synthesis Approach for Design Space Exploration". In proc. of the 26th ACM/IEEE DAC, pp. 68-74, 1989.

[2] J-M. Chang, M. Pedram,"Power Efficient Register Assignment", CENG Technical Report 95-03, University of Southern California, Mar. 1995.

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

[4] J. Edmonds and R.M. Karp,"Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", Journal of the ACM, vol 19, no2, pp 248-264, April 1972.

[5] M.C.Golumbic, "Algorithmic Graph Theory and Perfect Graphs", Academic Press 1980.

[6] G. Goossens, D Lanneer, J. Vanhoof, J Rabaey, J. Van Meerbergen, and H. De Man. "Optimization-Based Synthesis of Multiprocessor Chips for Digital Signal Processing with Cathedral-II". In G. Saucier and P.M. McLellan, editor, Logic and Architecture Synthesis, 1988.

[7] C.Y. Hitchcock and D.E. Thomas, "A Method of Automatic Data Path Synthesis", in Proc. of the 20th Design Automatic Conf. New York, NY:ACM/IEEE, pp. 484-489, June 1983.

[8] R. Hogg, A. Craig, "Introduction to Mathematical Statistics". fourth Edition. pp.147-152.

[9] D. Ku, G. De Micheli, "Relative Scheduling Under Timing Constraints", in DAC, Proceedings of Design Automation Conference, pp. 59-64, June 1990.

[10] K. Kücükcakar and A. Parker. "MABAL, A Software Package for Module and Bus Allocation". International Journal of Computer Aided VLSI Design, pp. 419-436, 1990.

[11] F. Kurdahi, A. Parker, "REAL: A Program for REgister Allocation", in Proceeding of 24th Design Automation Conference, June 1987.

[12] P. Michel, U. Lauther, P. Duzy, "The Synthesis Approach to Digital System Design", Kluwer Academic Publishers.

[13] B.M. Pangrle and D.D Gaski, "Design Tools for Intelligent Silicon Compilation", IEEE trans. on CAD, 6(6), Nov. 1987.

[14] C. Papadimitriou, K. Steiglitz, "Combinatorial Optimization, Algorithms and Complexity". Prentice-Hall, inc., 1982.

[15] A. Papoulis, "Probability, Random Variables, and Stochastic Processes". Third Edition, section 6.3, McGraw-Hill, 1991.

[16] P.G. Paulin and J.P. Knight,"Scheduling and Binding Algorithms for High-Level Synthesis", in Proc. of the 26th ACM/IEEE Design Automation Conference (DAC), page 1-6, 1989.

[17] L. Stok,"An Exact Polynomial Time Algorithm for Module Allocation", in Fifth International Workshop on High-Level Synthesis, Bühlerhöhe, pp.69-76, March 3-9 1991.

[18] C.-J. Tseng and D.P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems", IEEE trans. on CAD, 5(3), July 1986.