Power Efficient Register Assignment

Jui-Ming Chang and Massoud Pedram

CENG Technical Report 95-03

Department of Electrical Engineering - Systems
University of Southern
Los Angeles, California 90089-2562
(213)740-4458

July 5, 1995

Abstract

This paper present 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. Based on this, 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.

Contents

List of Figures

List of Tables

Chapter 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 attempts to minimize the number of resources to perform a task in a given time or tries to minimize the execution time for a given set of resources.

It is now necessary to develop behavioral synthesis techniques that also account for power dissipation in the circuit. 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, such as REAL[Kurd87], Facet[Tsen86], EMUCS[HiTh83], 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 [KuDM90]. Either approach has its own advantages and disadvantages. 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 ways to reduce the total number of the registers used. The existing approaches include rule-based [Goos88], greedy or iterative [KuPa90a], branch and bound [PaGa87], linear programming [BaMa90], and graph theoretic, as in the Facet system [TsSi86], the HAL system [PaKn89a] and the EASY system [Stok91] (see [DeMi94] or [PUPe] for more details).

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 produces 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 following chapters. 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* [Golumbic80]). 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 be used. The power optimization is performed once the cross-arc switching activities are known.

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

Chapter 2

Switching Activity Calculation

2.1 Calculation of various pdfs in a data flow graph

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 the 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 [Papoulis]. For example, suppose that we have only two input random variables $\mathbf{x}$ an $\mathbf{y}$, and the data flow graph contains internal arcs (also random variables) $\mathbf{z} = g(x, y)$, $\mathbf{w} = h(x, y)$. We denote the joint pdf of x, y as $f_{xy}(x, y)$.

We can find out the joint pdf $f_{zw}(z,w)$ for **z** and **w** as follows:

1. Find the inverse solution of the system of equations for z and w, i.e.,

$$\begin{aligned} g(x,y) &= z \\ h(x,y) &= w \end{aligned}$$

   Suppose the symbolic real roots of this system are (x_i, y_i), i = 1...n, i.e. $(x_i, y_i) = (u_i(z,w), v_i(z,w))$.

2. The joint pdf of z and w is obtained as:

$$f_{zw}(z,w) = \mid J^{-1}(x_1,y_1) \mid \times f_{xy}(x_1,y_1) + \cdots + \mid J^{-1}(x_n,y_n) \mid \times f_{xy}(x_n,y_n)$$

   where J^{-1} is the 2×2 inverse Jacobian:

$$J^{-1}(x,y) = \begin{vmatrix} \frac{\partial x}{\partial z} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial z} & \frac{\partial y}{\partial w} \end{vmatrix} \quad [\textit{Papoulis, page } 143]$$

The above change-of-variables technique can be extended to the case of a system with n input random variables [Hogg, Craig] as follows. 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:

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

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 $G(y) = \text{prob}(\text{Y} \leq \text{y})$, which is equal to $\text{prob}(w(x_1, x2, \ldots, x_n) \leq y)$. The above probability can be evaluated as:

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

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 $g(y) = \frac{d\,G(y)}{dy}$.

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

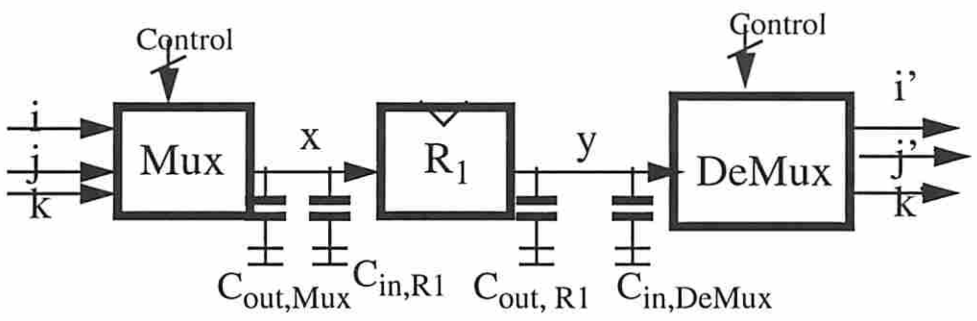

Figure 2.1: Our Register Sharing Model

on its input and output (see Fig. 2.1). 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.

Given a scheduled data flow graph, the life time of each arc is known. Let's denote the life time of each data value x_i by an interval $(birth\_time_i, death\_time_i)$. From the set of intervals, we can easily construct a directed graph G = *(V,A)*, where each vertex corresponds to an interval and arc $(u, v) \in A$ exists whenever the two intervals u and v do not overlap *(i.e. are compatible)* and $death\_time_u \leq birth\_time_v$. It can be easily proved that the directed graph G = *(V,A)* after removing the orientation is a comparability graph (or transitively oriented graph), as will be shown in Chapter 3.1.

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 registers were 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 $X \to Y \to Z \to W$. 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 $\mathcal{B} = \{(x,y) \mid x = x(a_1, a_2, \ldots, a_n), \quad y = y(a_1,a_2, \ldots, a_n), \quad \forall (a_1, a_2, \ldots, a_n) \in \mathcal{A}\}$. The switching activity between the two consecutive data values X and Y is then given by:

$$switching(X,Y) \quad = \quad \sum_{(x,y)\in\mathcal{B}} f_{xy}(x,y) * H(x,y) \tag{2.1}$$

where the summations are 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 (2.1) 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 Chapter 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 type* random variables x and y can be obtained much more easily by the following:

$$switching(X,Y) = \sum_{a_1}\sum_{a_2}\cdots\sum_{a_n} \psi(a_1,a_2,\ldots,a_n) * H(x(a_1,a_2,\ldots,a_n), y(a_1,a_2,\ldots,a_n)) \tag{2.2}$$

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 (2.1) and equation (2.2) started from the assumption that the joint pdf $\psi(a_1,a_2,\ldots,a_n)$ is obtained or known. This is a reasonable and necessary condition in order to precisely calculate the cross-arc switching activities. Furthermore, equation (2.2) 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 (2.1) is at most 2^{2*bit_width}. In general, equation (2.2) 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:

$$switching(Unknown,X) + switching(X,Y) + switching(Y,Z) + switching(Z,W) \tag{2.3}$$

$$= constant + \sum_{(x,y)\in\mathcal{B}} f_{xy}(x,y) * H(x,y) + \sum_{(y,z)\in\mathcal{C}} f_{yz}(y,z) * H(y,z)$$
$$+ \sum_{(z,w)\in\mathcal{D}} f_{zw}(z,w) * H(z,w) \tag{2.4}$$

$$= \quad constant + \sum_{a_1} \sum_{a_2} \cdots \sum_{a_n} \psi(a_1, a_2, \ldots, a_n) * (H(x,y) + H(y,z) + H(z,w)) \tag{2.5}$$

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.

Chapter 3

Power Optimization in the Register Allocation Phase

3.1 Max-Cost Flow formulation for the optimization

Definition 3.1 (Golumbic80) *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* transitive orientable graphs *and* partially oderable graphs. □

Definition 3.2 *A directed graph G_0 = (V,A) is called the* compatibility graph *for register allocation problem if the graph G_0 is constructed by the following procedure.*

Each arc$_i$ (data value) in the data flow graph has an interval $(birth\_time_i, death\_time_i)$ *associated with it. Each 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* $death\_time_u \leq birth\_time_v$ □

Definition 3.3 *A Graph G'_0 = (V,E) is called* unoriented compatibility graph *associated with the compatibility graph G_0 = (V,A) if $\forall$ arc a $\in A$ we have a corresponding e $\in E$ which is an edge formed by removing the orientation of arc a.* □

Theorem 3.1 *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.*

Proof: $\forall$ two edges *(u,v)* and *(v,w)* $\in E$ in $G'_0 = (V, E)$, we can find the associate two arcs (u', v') and (v', w') $\in A$ in $G_0 = (V, A)$. This implies that $death\_time_{u'} \leq birth\_time_{v'}$ and $death\_time_{v'} \leq birth\_time_{w'}$. The above two facts implies $death\_time_{u'} \leq birth\_time_{w'}$. This implies (u', w') $\in A$ from the constructing procedure stated before. The corresponding edge (u,w) is just the arc (u', w') with the removal of its orientation. So (u, w) $\in E$. This implies that G'_0 is a *comparability graph.*

□

Theorem 3.2 *Given a data flow graph without loops and branches, the oriented graph* G_0 = *G(V,A) for register allocation problem is acyclic.*

Proof: Assume the above is not true. Assume in the graph G_0, there is a cycle consisted of $v_i, v_{i+1}, \ldots, v_k, v_i, \ldots$. From the construction procedure of the graph G_0, we know that $birth\_time_i \leq death\_timei \leq birth\_time_k \leq death\_time_k$, but the above cycle tell us that $death\_time_k \leq birth\_time_i$. This is contractive. So the assumption is not true, this says that the oriented graph G_0 is acyclic. □

To minimize the total power consumption on the registers, a network $N_G = (V_n, E_n, s, t, C, K)$ is constructed from the directed (oriented) compatibility graph $G_0 = G(V, A)$ for the register allocation problem. This is a similar construction to the network used by [Stok91] in solving the *weighted module allocation* problem *which simultaneously minimizes the number of modules and the amount of interconnection needed to connect all modules.* Conceptually, $N_G = (V_n, E_n, s, t, C, K)$ 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 the source vertex s to every vertex in V of $G(V, A)$, and from every vertex in V of $G(V, A)$ to the sink vertex 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 $N_G = (V_n, E_n, s, t, C, K)$ is defined as the following:

$$\begin{aligned}
V_n &= V \cup \{s,t\} \\
E_n &= A \cup \{(s,v),\ (v,t) \mid v \in V\} \\
w(s,v) &= L - \lfloor switching(Unknown, v) * M \rfloor \\
&= L - \lfloor sw_0 * M \rfloor \qquad (3.1) \\
w(u,v) &= L - \lfloor switching(u,v) * M \rfloor \\
&= L - \lfloor \sum_{(u,v)\in\mathcal{B}} f_{uv}(u,v) * H(u,v) * M \rfloor \\
&= L - \lfloor \sum_{a_1}\sum_{a_2}\cdots\sum_{a_n} \psi(a_1,a_2,\ldots,a_n) * H(u(a_1,a_2,\ldots,a_n), v(a_1,a_2,\ldots,a_n)) \rfloor \qquad (3.2) \\
w(v,t) &= L,\ \forall\, v \in V,\ w(t,s) = L. \qquad (3.3)
\end{aligned}$$

where $\mathcal{A} = \{(a_1, a_2, \ldots, a_n)\}$, $\mathcal{B} = \{(u,v) \mid u = u(a_1,a_2,\ldots,a_n),\ v = v(a_1,a_2,\ldots,a_n),\ \forall (a_1,a_2,\ldots,a_n) \in \mathcal{A}\}$, $L = \lfloor max\ \{switching(u,v)\} * M \rfloor$ over all possible u,v $\in V \cup \{s\}$, and M is a large constant used to magnify 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 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.*

Proof: Since the capacity for all the arcs in $E_n \setminus \{(t,s)\}$ is one. The flow with $\mid f \mid = 1$ will flow through a directed path starting from source vertex s and ending at the sink vertex t. Also, the graph G_0 is acyclic, the directed path will not have repeated vertices. As G_0 is transitive, for any index i, if there are arcs (v_i, v_{i+1}) and (v_{i+1}, v_{i+2}), in the path, then there must be also an arc (v_i, v_{i+2}) in G_0 (for it is transistive). Use mathematical induction, there is an arcs between any two vertices in G_0 if the two vertices are in the directed path flown through by a flow with $\mid f \mid = 1$. The unoriented graph G'_0 is obtained by removing the orientation from all the arcs in G_0. This implies the directed path flown through by a flow of value on is a clique in 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.*

Proof: Since the capacity for all the arcs in $E_n \setminus \{(t,s)\}$ is one. The flow with $\mid f \mid = k$ will flow through a k arc disjoint paths in the network N_G. Each path forms a clique by the previous theorem.

□

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 proposed in [Sarra90]. 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 $N'_G = (V'_n, E'_n, s, t, C', K$ where:

$$\begin{aligned}
V'_n &= V_n \cup V'_0 \\
& \quad \textit{there is a vertex } v' = f(v) \in V'_0 \textit{ for each vertex } v \in V_0 \\
E'_n &= A' \cup \{(s,v),(f(v),t), v \in V_0\} \cup \{(t,s)\} \cup \{(v,f(v)) \mid v \in V_0\} \\
A' &= \{(f(u),v) \mid (u,v) \in A\} \\
C'((t,s)) &= C'((v,f(v)) = L, \ \forall \ v \in V_0 \\
C'((u',v)) &= C((u,v)) \textit{ for all } (u',v) \in A' \cup \{(s,v),(f(v),t) \mid v \in V_0\} \\
K'((t,s)) &= k, K'((u,v)) = 1 \textit{ for all } u \neq t, \textit{ and } v \neq s.
\end{aligned}$$

The transformations from the data flow graph to the final network N'_G are shown in Fig. 3.1.

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 (Papadimitriou, Steiglitz) *Let $N =$ (s,t,V,E,b) 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 b(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:*

$$\min \ c^t f$$

$$Af = -v_0 d \textit{ every node}$$

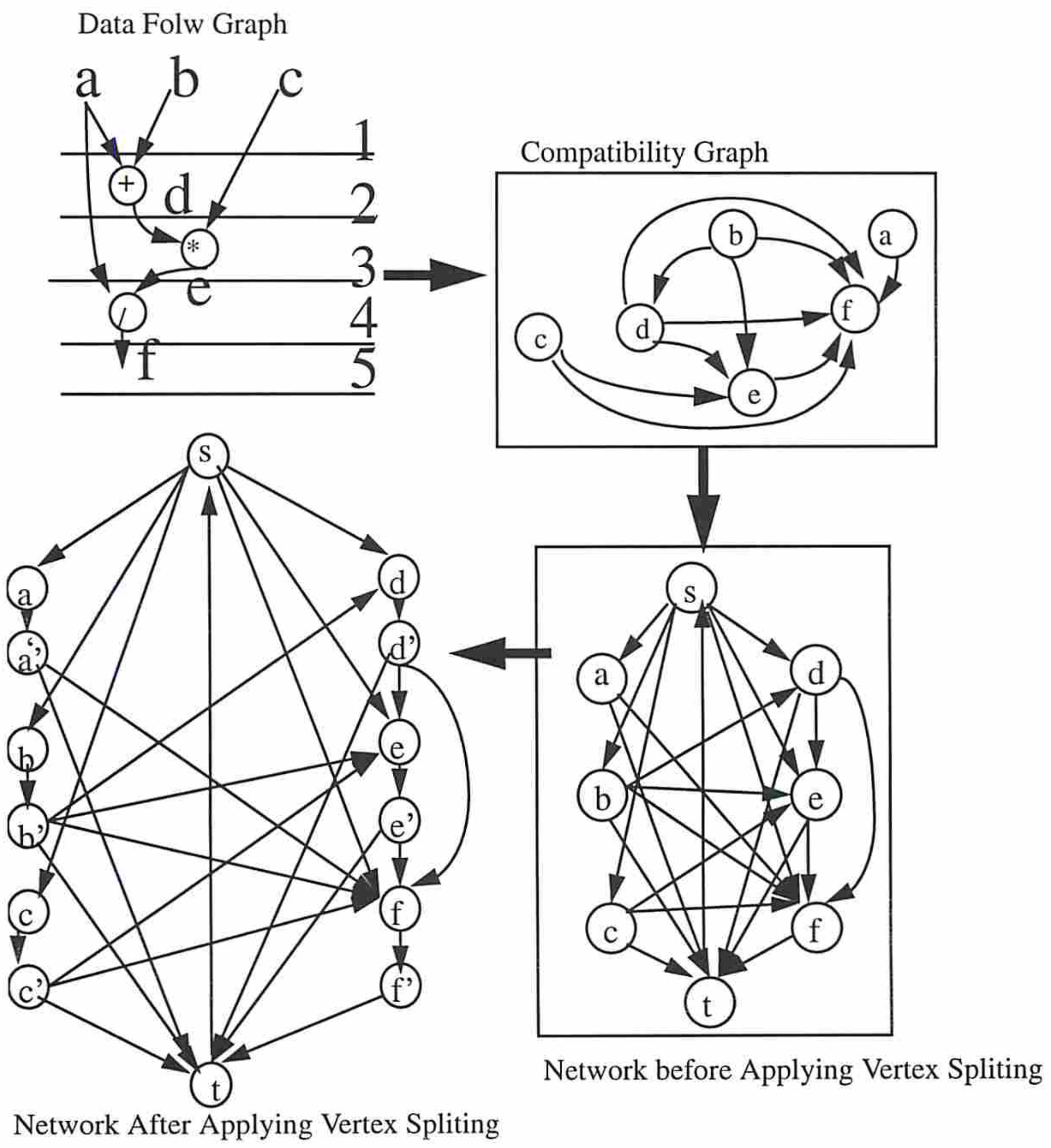
Data Folw Graph
a
b
c
+
d
*
e
/
f
1
2
3
4
5
Compatibility Graph
Network before Applying Vertex Spliting
Network After Applying Vertex Spliting
s
t
a'
b'
c'
d'
e'
f'

Figure 3.1: From Dataflow Graph to network N'_G

$$f \leq b \; every \; arc$$

$$f \geq 0 \; every \; arc$$

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,b) and a fixed flow value v_0, find the flow pattern that maximize 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.

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 $\sum_{e \in E_n} f(e) * c(e)$, which is a linear function of the *"Total Saved Power"*. The reason is that $\sum_{e \in E_n} f(e) * c(e) = \sum_{e \in E_n} f(e) * [L - M * switching(e)] =$

$$L * \sum_{e \in E_n} f(e) - M * \sum_{e \in E_n} f(e) * switching(e)$$

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 $\mid f \mid \geq k_{min}$. k_{min} can be determined by the left edge algorithm [Kurd87] or simply by finding the maximum number of arcs cut by the transitions of C-steps among all C-steps. 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. The key idea is that whenever $k_{min(max\_cost\_flow)}$ is greater than $k_{min(left\_edge)}$, we try to push those vertices uncovered by the flow back to the $k_{min(left\_edge)}$ paths and minimize the total cost. □

The time complexity for the Max-Cost flow Algorithm is $\mathcal{O}(km^2)$, according to [Edmonds72], where m $= 2\times \mid V \mid +2$ for the graph $G_0 = G(V, A)$ and k is the flow valus.

3.2 Handling of the conditional branches in data flow graphs

In the following discussion, we consider the cases that each *conditional execution (or branching)* is represented by a $D - J$ (*Distribute-Join*) block. Only one of the branches is traversed or executed when the test condition has been evaluated. In the simplest case, the data flow graph in Fig. 3.2 contains only one $D - J$ block. Let A and B denote the left and right branches, respectively. Suppose that the primary inputs to this data flow graph are a,b,c and d. The condition to be evaluated in the D block can be written as function of the primary inputs, $g(a, b, c, d)$. (In general, a branch condition may be a function of all variables including the internal variables. However, the internal variables can be eventually represented in terms of

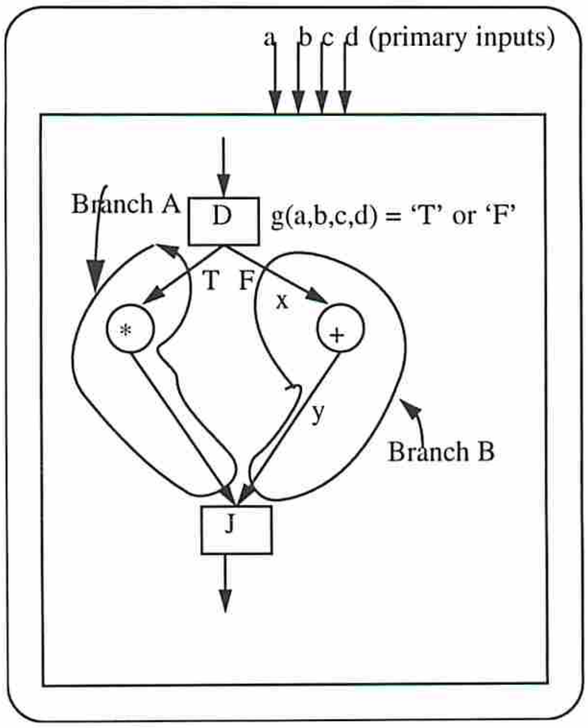

Figure 3.2: The dataflow graph with $D-J$ blocks and two branches

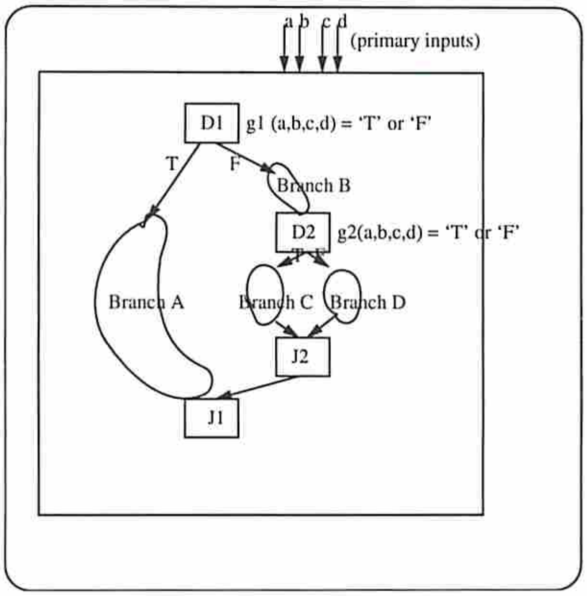

Figure 3.3: The dataflow graph with nested $D-J$ blocks and four branches

functions of primary input variables.) Let's assume that branch A is the branch taken when $g(a,b,c,d)$ = *'true'*. Suppose the joint pdf of the primary inputs a,b,c and d is $\psi(a,b,c,d)$. Let's denote the set $\mathcal{S} = \{(a,b,c,d) \mid g(a,b,c,d) =' true'\}$. The switching activity between any two arcs x and y in branch A of the data flow graph can be written as the follows: [1]

$$switching(x,y) = \sum_{(a,b,c,d)\in\mathcal{S}} \psi(a,b,c,d) * H(x(a,b,c,d), y(a,b,c,d)). \quad (3.4)$$

The above method can be extended to calculate the switching activity for a data flow graph consisting of several $D-J$ blocks. For example, in Fig. 3.3, there are two $D-J$ blocks, where $D2-J2$ is nested in $D1-J1$. Let's denote the test condition in $D1$ as g_1, and the test condition in $D2$ as g_2. Obviously, branch A in Fig. 3.3 is executed when g_1 = 'true', branch B and branch C are executed when g_1 = 'false' and g_2 = 'true', and branch B and branch D are executed when

[1]Note that $switching((x,y) \mid branch\ A\ is\ taken) = \sum_{(a,b,c,d)\in\mathcal{S}} \frac{\psi(a,b,c,d)}{P_A} * H(x(a,b,c,d), y(a,b,c,d))$, where P_A is the probability that branch A is taken. $P_A = \sum_{(a,b,c,d)\in\mathcal{S}} \psi(a,b,c,d)$. Thus, $switching(x,y) = P_A * switching((x,y) \mid branch\ A\ is\ taken) = \sum_{(a,b,c,d)\in\mathcal{S}} \psi(a,b,c,d) * H(x(a,b,c,d), y(a,b,c,d))$.

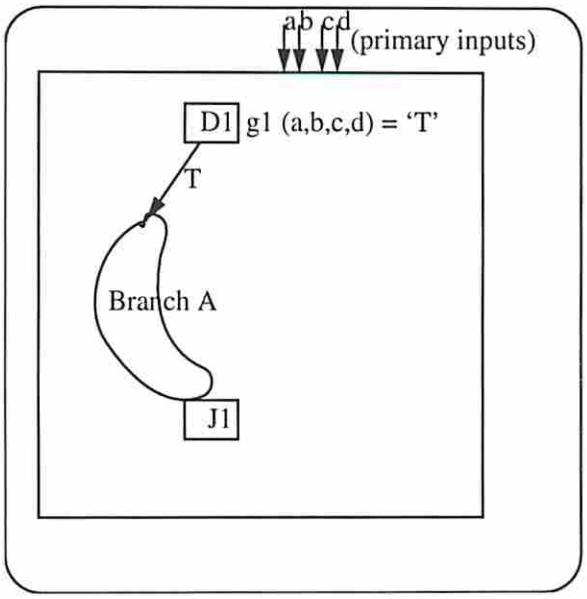

Figure 3.4: The dataflow graph G_1 with Branch A

g_1 = 'false' and g_2 = 'false'. Again, assume the data flow graph has primary inputs a,b,c and d, with joint pdf as $\psi(a,b,c,d)$. Let's denote the set $S_1 = \{(a,b,c,d) \mid g_1(a,b,c,d) = 'true'\}$, set $S_2 = \{(a,b,c,d) \mid g_1(a,b,c,d) = 'false' \;\; and \;\; g_2(a,b,c,d) =' true'\}$, and set $S_3 = \{(a,b,c,d) \mid g_1(a,b,c,d) = 'false' \;\; and \;\; g_2(a,b,c,d) =' false'\}$. When $(a,b,c,d) \in S_1$, we obtained a new unconditional data flow graph G_1 consisting of everything in the original dataflow graph except branches B,C and D, as shown in Fig. 3.4 When $(a,b,c,d) \in S_2$, we obtaind a new unconditional data flow graph G_2 consisting of everything in the original dataflow graph except branches A and D, as the one shown in Fig. 3.5 When $(a,b,c,d) \in S_3$, we obtaind a new unconditional data flow graph G_3 consisting of everything in the original dataflow graph except branches A and C, as shown in Fig. 3.6 Since the three new data flow graph G_1, G_2, and G_3 are themselves unconditional, we can use the previous method to find the switching between any two arcs x and y in the three new graphs.

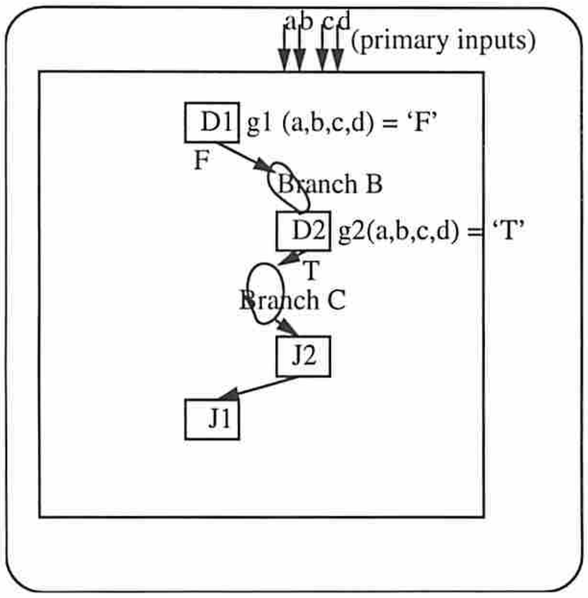

Figure 3.5: The dataflow graph G_2 with Branch B and Branch C

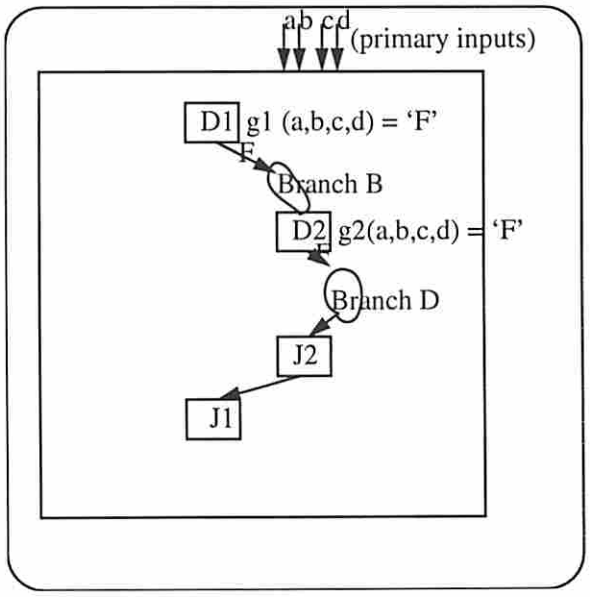

Figure 3.6: The dataflow graph G_3 with Branch B and Branch D

For example, the switching activity between any two arcs x and y on G_1 is the following:

$$switching((x,y)) = \sum_{(a,b,c,d)\in S_1} \psi(a,b,c,d) * H(x(a,b,c,d), y(a,b,c,d)) \tag{3.5}$$

After we obtaind all of the cross-arc switching activities on G_1, we can construct a new network N'_{G_1} out of G_1, and perform the *Max-Cost Flow* algorithm on the N'_{G_1}, and find out n_1, the minimum number of cliques needed in G_1. Similarly, we can obtain n_2, and n_3 from G_2 and G_3. *Let* $N = max(n_1, n_2, n_3)$, this is the actual total number of the registers that will be used on the original conditional data flow graph. After we obtain N, we perform the *Max-Cost Flow* on the three networks N'_{G_1}, N'_{G_2}, and N'_{G_3} respectively, given the fixed flow value equal to N. From that we obtain the cliques in G_1, G_2, and G_3 and calculate the total switching activity on G_1, G_2, and G_3 as $total\_switching(G_1)$, $total\_switching(G_2)$, and $total\_switching(G_3)$. The actual total switching activity of the original conditional data flow graph is just the following:

$$\begin{aligned} total\_switching(Conditional\ DFG) &= total\_switching(G_1) + total\_switching(G_2) \\ &\quad + total\_switching(G_3) \end{aligned} \tag{3.6}$$

Within each individual unconditional data flow graph G_1, G_2, or G_3, data values share the N physical registers. The same N register are shared by the three data flow graphs G_1, G_2, and G_3. Here we are considering the non-pipelined design, and the sharing of the same N registers by the three data flow graphs takes place sequentially. Given an input vector, one and only one of the three unconditional data flow graphs will be traversed and executed and the new unconditional data flow graph will be executed only when a new input vector comes in. The sharing of the N registers among the three unconditional data flow graphs G_1, G_2 and G_3 is achieved by using multiplexors and issuing control signals at correct C-Steps.

Chapter 4

Example:

The following example is based on a scheduled data flow graph as the one shown in Fig 4.1. This simple data flow graph has five primary input variables a,b, c, d and e. For the sake of presentation, we choose the 5-variate joint Gaussian distribution as the joint pdf of a,b, c, d and e. Note however that our method works for arbitrary joint pdf's. The 5-variate Gaussian is a good choice as this pdf is commonly seen in many application domains, like DSP. Let

$$X = \begin{pmatrix} a - \mu_a \\ b - \mu_b \\ c - \mu_c \\ d - \mu_d \\ e - \mu_e \end{pmatrix}, \quad where \begin{pmatrix} \mu_a \\ \mu_b \\ \mu_c \\ \mu_d \\ \mu_e \end{pmatrix} = \begin{pmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{pmatrix}$$

The 5-variate Gaussian distribution is given by:

$$Gaussian5(a,b,c,d,e) = f(X) = \frac{1}{\sqrt{(2\pi)^5 * det(C)}} exp\{-\frac{1}{2}X^t\, C^{-1}\, X\}$$

The matrix C is the covariance matrix for the 5-variate Gaussian joint pdf ad is given by:

$$C = \begin{pmatrix} \sigma_{aa} & \sigma_{ab} & \sigma_{ac} & \sigma_{ad} & \sigma_{ae} \\ \sigma_{ba} & \sigma_{bb} & \sigma_{bc} & \sigma_{bd} & \sigma_{be} \\ \sigma_{ca} & \sigma_{cb} & \sigma_{cc} & \sigma_{cd} & \sigma_{ce} \\ \sigma_{da} & \sigma_{db} & \sigma_{dc} & \sigma_{dd} & \sigma_{de} \\ \sigma_{ea} & \sigma_{eb} & \sigma_{ec} & \sigma_{ed} & \sigma_{ee} \end{pmatrix} = \begin{pmatrix} 69.3092 & -25.3706 & -38.2909 & -12.6694 & 6.57557 \\ -25.3706 & 73.2355 & -11.9927 & 4.95206 & -63.0632 \\ -38.2909 & -11.9927 & 91.0169 & -21.7674 & 2.55902 \\ -12.6694 & 4.95206 & -21.7674 & 66.7315 & 10.3661 \\ 6.57557 & -63.0632 & 2.55902 & 10.3661 & 74.7069 \end{pmatrix}$$

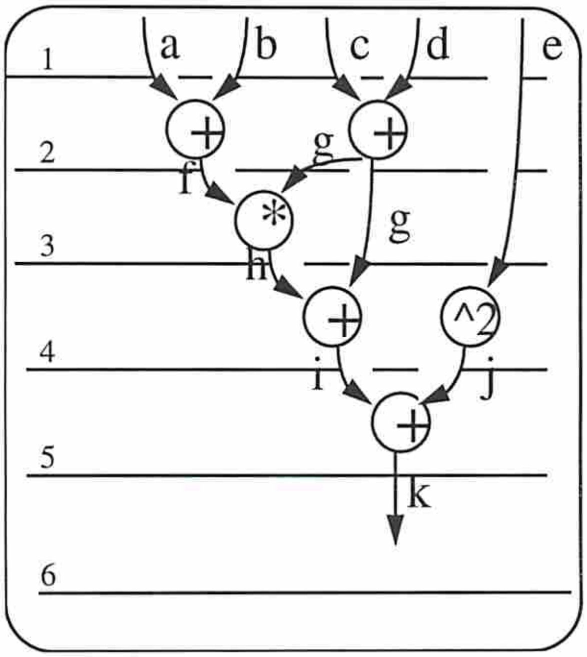

Figure 4.1: The Scheduled Data Flow graph

The numerical values on the right hand side are provided as an example. From Fig. 4.1, we know that there are six intermediate variables in the data flow graph, that is, random variables *f, g, h, i, j, and k.*

$$\begin{aligned} f &= a+b \\ g &= c+d \\ h &= (a+b)*(c+d) \\ i &= (a+b+1)*(c+d) \\ j &= e^2 \\ k &= e^2+(a+b+1)*(c+d) \end{aligned}$$

The variables' life times are:

$\{a[1,2],\ b[1,2],\ c[1,2],\ d[1,2],\ e[1.4],\ f[2,3],\ g[2,4],\ h[2,4],\ i[4,5],\ j[4,5],\ k[5,6]\}$

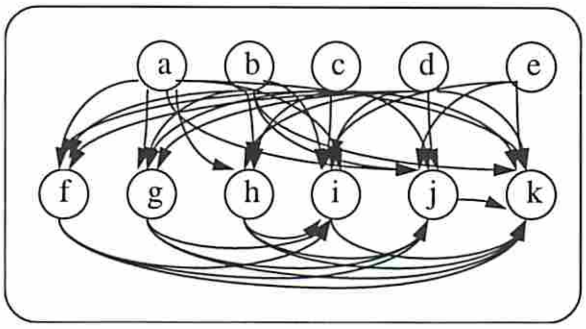

Figure 4.2: The Oriented Compatibility Graph for register allocation problem

From the above life times, the procedure used to construct the oriented compatibility graph for the register allocation problem generates $G_0 = G(V, A)$ shown in Fig. 4.2.

In this example, we assume 16-bit registers and the two's complement representation for the values. All numbers are in the range [-32768, 32767].

We used equation (2.2) in Chapter 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 $(1/5)*[switching(0,a)+switching(0,b)+ switching(0,c)+switching(0,d)+ switching(0,e)]$.

Here are the results:

After calculating the switching activities, we construct the max-cost flow network. The weight on each arc is calculated by equation (3.1)-(3.3). in Chapter 3.

Here we choose M = 1000, and so L = 10836. The following weights are obtained:

$w(x, x') = L$, $\forall x \in V$, and

	a	b	c	d	e	f	g	h	i	j	k	t
a						6.138	7.522	8.772	8.931	8.140	9.604	0
b						6.150	7.082	8.735	8.839	7.771	9.508	0
c						7.464	6.238	8.612	8.546	8.220	9.134	0
d						7.026	6.285	8.579	8.541	8.129	9.196	0
e									8.692	6.953	8.546	0
f									10.158	8.010	10.187	0
g									7.223	8.419	8.763	0
h									8.119	10.836	10.227	0
i											7.921	0
j											9.350	0
s	5.566	5.566	5.566	5.566	5.566	5.566	5.566	5.566	5.566	5.566	5.566	0

Table 4.1: Cost Matrix for network

	a	b	c	d	e	f	g	h	i	j	k	t
a						4698	3314	2064	1905	2696	1232	10836
b						4686	3754	2101	1997	3065	1328	10836
c						3372	4598	2224	2290	2616	1702	10836
d						3810	4551	2257	2295	2707	1640	10836
e									2144	3883	2290	10836
f									678	2826	649	10836
g									3613	2417	2073	10836
h									2717	0	609	10836
i											2915	10836
j											1486	10836
s	5270	5270	5270	5270	5270	5270	5270	5270	5270	5270	5270	

Table 4.2: Cost Matrix for network

No. of reg	cliques	actual total switching activity
7	$\{\{a,f\},\{c,g,i\},\{e,j\},\{b\},\{d\},\{h\},\{k\}\}$	65.514
6	$\{\{a,f\},\{c,g,i,k\},\{e,j\},\{b\},\{d\},\{h\}\}$	67.872048
5	$\{\{a,f\},\{c,g,i,k\},\{d,h\},\{e,j\},\{b\}\}$	70.882

Table 4.3: Results

No. of reg	cliques	actual total switching activity
5	$\{\{a,h,k\},\{b,f,i\},\{c,g,j\},\{d\},\{e\}\}$	80.487882

Table 4.4: Results

Applying the Max-Cost Flow on the network in Fig. 4.3 with the vertex splitting technique, the following results are obtained:

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:

which is 13.55% worse than the optimum solution.

Next, we generated register assignment solution using Real [Kurd87] which finds the minimum number of registers need (in this case) and obtained the following result:

No. of reg	cliques	actual total switching activity
5	$\{\{a,f,i,k\},\{b,g,j\},\{c,h\},\{d\},\{e\}\}$	78.471137

Table 4.5: Results

	Min-Power Register Assignment	
No. of reg	cliques	actual total switching activity
9	$\{\{a,i\},\{b,h\},\{e,j,k\},\{l,m\},\{n\},\{c\},\{d\},\{f\},\{g\}\}$	6.861
8	$\{\{a,l,m\},\{b,h\},\{e,j,k\},\{f,i\},\{c\},\{d\},\{g\},\{n\}\}$	7.272
7	$\{\{a,l,m,n\},\{d,h\},\{e,j,k\},\{f,i\},\{b\},\{c\},\{g\}\}$	7.763

Table 4.6: Results

	Min-Count Register Assignment	
No. of reg	cliques	actual total switching activity
7	$\{\{a,j,m\},\{b,h,k,n\},\{c,i,l\},\{d\},\{e\},\{f\},\{g\}\}$	10.017

Table 4.7: Results

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 shown in Fig. 4.4 using similar assumptions about the joint pdf and the data types. Specifically,

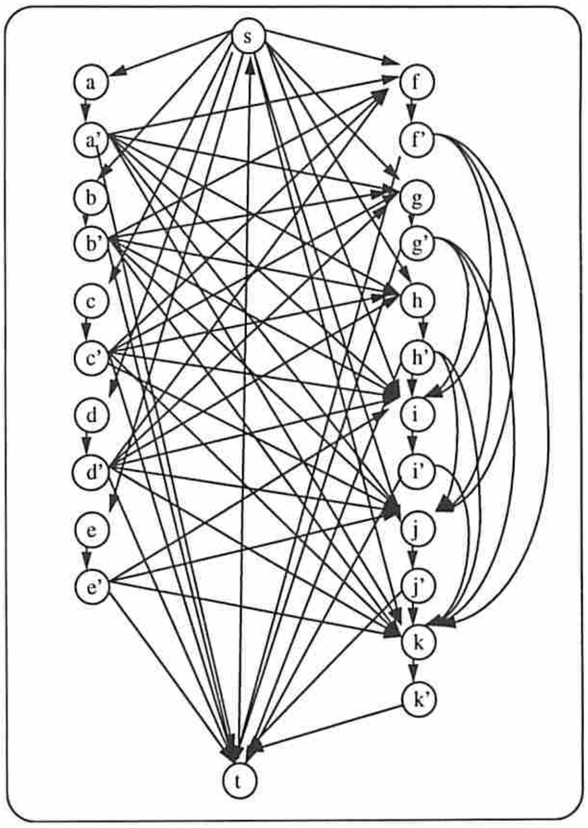

Figure 4.3: The network After applying the vertex splitting techniques

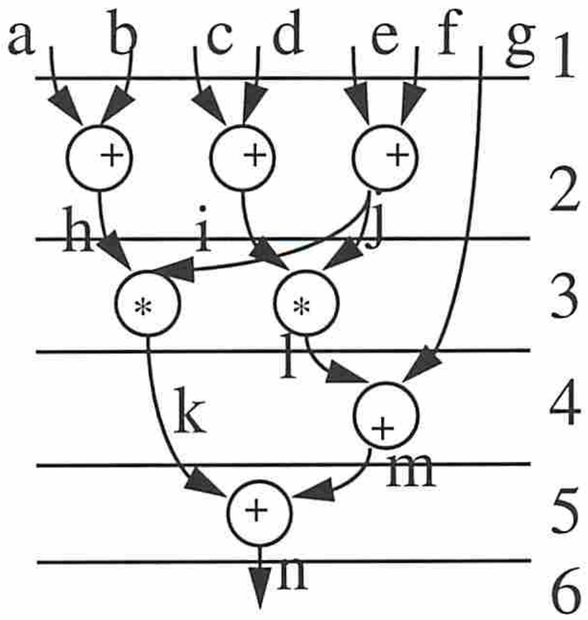

Figure 4.4: The data flow graph with 7 primary inputs.

Chapter 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 (probability density function) 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.

Bibliography

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

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

[Hogg, Craig] Robert Hogg, Allen Craig, Introduction to Mathematical Statistics. fourth Edition. pp.147-152.

[Golumbic80] M.C.Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press 1980.

[Sarra90] M. Sarrafzadeh and R.D. Lou, "Maximum k-coverings of weighted Transistive Graphs with Application," in proc. of the intl. Symposium on Circuits and Systems, New Orleans, May 1990.

[Stok91] Stok, L.,"An Exact Polynomial Time Algorithm for Module Allocation," in Fifth International Workshop on High-Level Synthesis, Bühlerhöhe, pp.69-76, March 3-9 1991.

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

[Kurd87] Fadi Kurdahi, Alice Parker, "REAL: A Program for REgister Allocation", in Proceeding of 24th Design Automation Conference, June 1987.

[Tsen86] C.Tseng and D.P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems," IEEE Trans. on CAD, vol CAD-5, no.3, pp 379-395, July, 1986.

[HiTh83] 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.

[KuDM90] David C. Ku, G. De Micheli, "Relative Scheduling Under Timing Constraints," in DAC, Proceedings of Design Automation Conference, pp. 59-64, June 1990.

[Edmonds72] Edmonds, J 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.

[PUPe] Petra Michel, Ulrich Lauther, Peter Duzy, "The Synthesis Approach to Digital System Design", Kluwer Academic Publishers.

[Goos88] 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.

[KuPa90a] 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.

[PaGa87] B.M. Pangrle and D.D Gaski, "Design Tools for Intelligent Silicon Compilation", IEEE trans. on CAD, 6(6), Nov. 1987.

[BaMa90] 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.

[TsSi86] C.-J. Tseng and D.P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems", IEEE trans. on CAD, 5(3), July 1986.

[PaKn89a] 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.