Optimal Module and Voltage Assignment for Low-Power

Deming Chen[+], Jason Cong[+], Junjuan Xu[*+]
[+]Computer Science Department, University of California, Los Angeles, USA
[*]Computer Science and Technology Department, Peking University, Beijing, PRC
{demingc, cong, irene.xu}@cs.ucla.edu

Abstract – Reducing power consumption through high-level synthesis has attracted a growing interest from researchers due to its large potential for power reduction. In this work we study functional unit binding (or module assignment) given a scheduled data flow graph under a dual-Vdd framework. We assume that each functional unit can be driven by a low Vdd or a high Vdd dynamically during run time to save dynamic power. We develop a polynomial-time optimal algorithm for assigning low Vdd to as many operations as possible under the resource and time constraint, and in the same time minimizing total switching activity through functional unit binding. Our algorithm shows consistent improvement over a design flow that separates voltage assignment from functional unit binding. We also change the initial scheduling to examine power-latency tradeoff scenarios. Experimental results show that we can achieve a 21% power reduction when latency bound is tight. When latency is relaxed by 10 to 100%, the power reduction is 31 to 73% compared to the synthesis results for the case of single high Vdd without latency relaxation. We also show comparison data of energy consumption under the same experimental setting.

I. Introduction

With the exponential growth of the performance and capacity of integrated circuits, power consumption has become the most critical constraining factor in the IC design flow [11]. Excessive power consumption limits the degree of transistor integration on a single chip, requires expensive packaging and cooling systems, shortens battery lifetime for portable devices, and brings on problems of signal integrity.

There are two major sources of power consumption, dynamic power and static power. Dynamic power is consumed when signal transitions take place at gate outputs. Static power (also called leakage power) is consumed when the circuit is either active or idle. According to [13], static power may take up to 42% of total power in 90nm technology. In [14], a similar percentage is reported for certain FPGA architectures in 100nm technology. Therefore, both dynamic and static power needs to be optimized.

Dynamic power consumption is calculated as $P_d=0.5S\cdot C\cdot V_{dd}^2 f$, where S denotes the switching activity of the circuit, C denotes the effective capacitance, V_{dd} is the supply voltage, and f is the operating frequency. To lower dynamic power, each of these factors can be reduced. Deploying multiple supply voltages is one of the most effective techniques to reduce dynamic power. This technique has the advantage of reducing power dissipation without sacrificing the performance of the system by assigning high Vdd to critical paths and low Vdd to non-critical paths. Clusters of high-Vdd cells and low-Vdd cells were first explored in [25]. The work in [24] adopted multiple supply voltages in the real design of a MPEG4 video codec. To reduce static power, power gating is an efficient technique [9][20]. When there are no useful operations executing on a module, it can be shut down to get rid of static power.

Our work focuses on power optimization at the behavioral level. The higher the design level is, the more critical the design decisions are for the quality of the final product. The behavioral synthesis process mainly consists of three stages: scheduling, allocation, and assignment. Scheduling determines when a computational operation will be executed; allocation determines how many instances of each type of resources (functional units or registers) are needed; assignment assigns/binds operations (or variables) to the resources. The number of resources may be limited and the total time (latency) to finish the operations can be constrained. The essence of behavioral synthesis with multiple supply voltages is to assign low-Vdd values to as many operations as possible under latency and resource constraints. In [21], an optimal solution was given for time-constrained scheduling problem under variable voltages. However, it did not consider resource constraints. The works in [12][17][19] proposed different heuristics for the time- and resource-constrained scheduling and binding problem under multiple voltages. These works adopted iterative methods to perform the two sub-tasks simultaneously. However, no switching activity was considered in their formulations. On the other hand, the works in [1][2][18] minimized switching activity for various resources, such as registers, functional units, and buses, but only single Vdd was considered. There is no existing work combining voltage assignment and switching activity reduction together elegantly to reduce power through behavioral synthesis.

In this paper, we derive an optimal algorithm to simultaneously assign maximum number of operations to low Vdd and minimize total switching activity through functional unit binding for the design. We use a network flow formulation. The solution of the min-cost flow will produce the binding and voltage assignment solution. All of these are done under latency and resource bounds. In addition, we change the initial scheduling to study power-latency tradeoffs, and provide power optimization solutions under different design constraints. We design our architecture model in such a way so that each functional unit can be driven by either the high Vdd or the low Vdd, or get into a sleep mode. Thus, we can target reducing dynamic power through dual Vdds and reducing static power through power gating. Experimental results show that we can achieve significant amount of power savings compared to the single-Vdd case.

In the following, Section II and III provide the details of our architecture model and power model. Section IV describes our dual-Vdd binding and assignment algorithm in detail. Section V shows experimental results, and Section VI concludes this paper.

II. Architecture Model

Our proposed architecture model is shown in Fig. 1. We insert two PMOS transistors between the high-Vdd (VddH) and low-Vdd (VddL) power rails and a functional unit (FU). The PMOS transistors are like sleep transistors, and the control bits C_1 and C_2 are used to control them so that an appropriate supply voltage can be chosen for the FU. When both transistors are off, the FU is in sleep mode. This scheme is similar to that used in [15], where each configurable logic block (CLB) in an FPGA is in such an arrangement. We believe functional unit-level granularity for dual-Vdd configuration is natural for high-level synthesis. In addition, we assume that the FU's voltage can be dynamically changed during run time, which dramatically

improves the chances for operations to execute under VddL. A more detailed diagram of the FU shows level converters (LC) at the input ports. A VddL signal needs to go through the level converter if it is going to drive a VddH device. Otherwise, the signal can bypass the converter through the MUX. We use the converter design from [5]. A single level converter contributes 0.08ns delay and 9.7E-15 Joul energy per switch. The MUX associated with the converter contributes 14 ps delay and about 2.0E-15 Joul energy per switch. All of these data were obtained with 100 nm technology [5]. The bit-width of the FU is 24.

According to previous works, the overhead of dual-Vdd power rail and level converters is acceptable compared to the amount of power savings achieved. A new layout style of standard cells for ASIC designs was proposed in [26], showing that adding a second power grid and level converters increased circuit area by 15%, but saved power by 47%. For FPGA designs, the area overhead of sleep transistors was 24% over the original CLB size with 5% delay overhead, and the power consumption of the sleep transistors could be optimized and become almost ignorable [15].

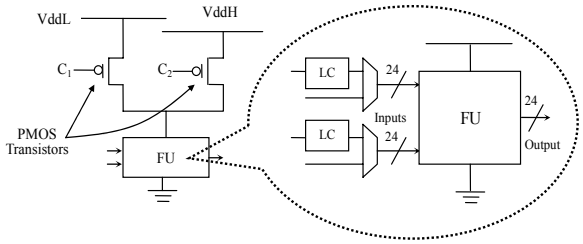


Fig. 1: Proposed architecture scheme for dual supply voltages

III. Power Model and Analysis

We use area, delay and power data extracted from [4] for adders and multipliers driven by VddH. The data was obtained through FPGA evaluation tool *fpgaEva_LP* [14] under 100nm technology. We use VddH as 1.3v and VddL as 0.8v in this work.[1] The clock period is set as 6.5 ns, i.e., the delay of each cycle (control step) in the schedule takes 6.5 ns. Table 1 shows some details. *Exe Cycle* represents the number of cycles for the operation to finish one 24-bit addition or multiplication. *E per Switch* is the energy consumed by the adder or multiplier when the output of the FU has a full voltage swing from 0 to 1. Notice that we use the data related to FPGA only because these data are available in recent publications. Our work can be applied to the ASIC design flow as well.

Items / Data	Adder/Subtractor		Multiplier	
	VddH	VddL	VddH	VddL
Exe Delay (ns)	6.1	10.5	14.6	25.2
Exe Cycle	1	2	3	5
Power (w)	0.016	0.006	0.246	0.093
Dynamic Power %	77%	77%	84%	84%
Static Power %	23%	23%	16%	16%
E per Switch (J)	3.2E-10	1.2E-10	4.9E-09	1.86E-09

Table 1: Characterization of FUs for VddH and VddL

Next, we derive the conditions of applying power gating and compute the power overhead to charge an FU from VddL to VddH. According to the data presented in [16], the circuit controlled by a sleep transistor needs at least one cycle to shut down and another cycle to come back alive. The maximum turn-on charging current can reach up to 87% larger than the normal switching current. Therefore, the turn-on power overhead (dynamic power) is larger than the dynamic power consumed during the normal operation in one cycle. We can quantify this overhead by the following formula:

$$P_{overhead} = \frac{Ratio_{signal\_restore}}{0.5 \cdot SA_{FU}} \cdot DynamicPower$$

where $Ratio_{signal\_restore}$ is the percentage of signals that are to be restored to logic high to power up the FU, and SA_{FU} is the switching activity for the FU. We assume that, on average, half of the signals are to be restored to logic high in the FU, i.e., $Ratio_{signal\_restore} = 0.5$. We can obtain SA_{FU} through simulations on our designs. Since power gating only saves static power (assuming no switches for idle FUs), we need to guarantee that the static power saved will surpass the turn-on power overhead before turning off the FU. Thus, we define the following formula to calculate the number of sleep cycles for a FU to start saving power overall:

$$SleepCycle = \frac{P_{overhead}}{StaticPower} + 2$$

The number 2 at the end counts in one cycle to turn off the FU and one cycle to turn on the FU. By this formula, it will need 9 (13) cycles for an adder (multiplier) to remain idle to guarantee that turning off the FU will save power ($SA_{FU} = 0.5$). These numbers are for the adders and multipliers used in our work. It is easy to see that if static power occupies 50% of the total power, *SleepCycle* will become 4. Charging energy can be calculated as follows [14]:

$$E_{(V_1 \to V_2)} = \frac{C}{2}(V_1 - V_2)(V_1 + V_2 - 2V_{dd})$$

C is load capacitance; V_1 is the initial value of gate output with a rising transition; V_2 is the final voltage. $V_2 = V_{dd}$ in our case. Plug in our VddL and VddH values, the charging energy from VddL to VddH is 15% of the charging energy from GND to VddH. Our *Exe Cycle* numbers assigned to the VddL operations provide enough cushion time.[2] Since the charging from VddL to VddH can be done in a much shorter time than from GND to VddH (turn-on time), we don't need an extra cycle when the FU's voltage changes from VddL to VddH or vice versa.

We use a simulation-based method with random input vectors to estimate switching activities between different operations. The simulation is similar as that used in [4]. After voltage assignment and binding for the operations, we accumulate power for FUs when they are either active (dynamic and static power) or idle (static power). We consider power overhead on the FU due to voltage changes or power gating for the dual-Vdd case. Power consumption due to level converters is also counted for the dual-Vdd case.

IV. Optimal Functional Unit Binding with Voltage Assignment for Low Power

Our objective is to assign low Vdd to maximum number of operations under latency and resource constraints, and in the same time carry out functional unit binding to minimize total switching activity of the design. Both of these optimization techniques will reduce dynamic power, and an efficient method is required to search through the combined optimization solution space for simultaneous voltage and module assignments. Our input is a design after scheduling. The scheduling solution itself fulfills latency and resource constraints and our voltage assignment and binding solution will honor these constraints. We will present an optimal algorithm for our objective in this section. We also apply power gating optimization as a post-processing procedure and examine its

[1] These two values form the best combination in [5], which falls into the optimal VddL/VddH ratio range as indicated in [10].

[2] For example, VddL addition only needs 10.5 ns. There is a 2*6.5-10.5 = 2.5ns cushion time before a new cycle starts.

effectiveness on power saving potentials. In the next, Subsection *A* presents details of our problem formulation. Subsection *B* reduces our problem into a network flow formulation and presents the details on its solution generation and optimality. Subsection *C* presents our power gating approach.

A. Problem Formulation

The operations and their data dependencies can be represented by a data flow graph (DFG), $G = (V, A)$. Set V corresponds to operations and set A corresponds to data flowing between operations. An edge $a = (x, y) \mid x, y \in V, a \in A$ indicates there is a data dependency between operations x and y. Scheduling assigns operations to control steps so that the overall execution latency meets a certain time constraint, and the number of resources used also meets a certain resource constraint. After scheduling, the life time of each operation in the DFG is the time during which the operation is active. A *comparability graph* $G_c = (V_c, A_c)$ for these operations can then be constructed for addition and multiplication separately. V_c corresponds to all the operations of the same type, and there is a directed edge $a_c = (v_i, v_j) \mid a_c \in A_c$ between two vertices if and only if their corresponding life times do not overlap, and operation v_i comes before v_j. In such a case, we call operations v_i and v_j *comparable* with each other, and they can be bound into a single FU without life time conflicts. Let w_{ij} denote the weight of edge a_c, which represents the cost when we bind v_i and v_j into the same FU. This cost is the switching activity between these two operations when v_j executes right after v_i on the FU.

To consider dual-Vdd assignment on a scheduled DFG, we introduce two definitions. An operation O is *extendable* if O can be assigned to VddL, and the extended execution delay of O will not violate the overall latency constraint, and in the same time, the data dependencies between O and other operations are still valid. In other words, O will still generate its data in time so that the data can flow to all the other operations that require it. If O is assigned VddL in the final solution, we say O is *extended*. Due to the resource constraint, not all extendable operations can be extended eventually. Fig. 2 shows an example. Fig. 2(a) shows a scheduled DFG with 6 multiplications. The *Exe Cycle* is 3 cycles for VddH and 5 cycles for VddL, and the number of available multipliers is 3. Multiplication 1, 2 and 3 are extendable, which is shown in Fig. 2(b). However, only two can be extended to meet the resource constraint. If operations 1 and 3 or 2 and 3 are chosen to be extended, although resource constraint is fulfilled for control step 5, it will be violated in step 6. Therefore, we need an efficient way to assign VddL to as many operations as possible. Suppose M_e is the maximum number possible of extended operations given a resource constraint, and the total number of extendable operations is T_e, we have $M_e \leq T_e$ for a design. It is easy to see that there may be different sets of M_e operations and each of such sets fulfills the constraints. Which set of M_e operations to extend will influence power reduction because different extensions will change the original G_c into a different new comparability graph since the life times of the M_e operations in G_c have changed. Let G_c' denote the new comparability graph due to the extensions. G_c' has the same node set V_c but a different A_c.

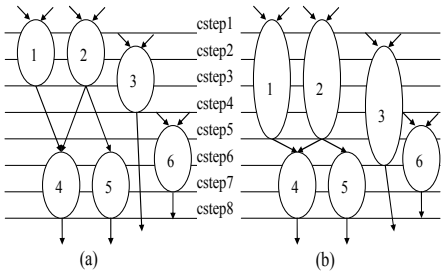


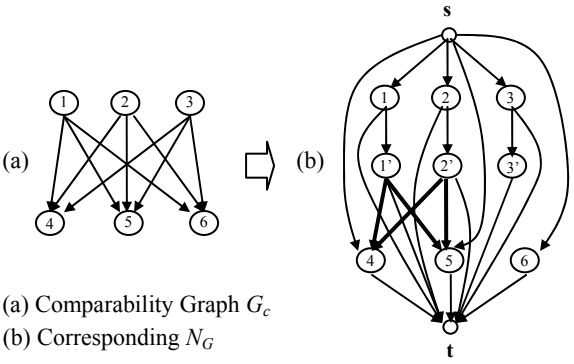
Fig. 2: Example of extendable operations

Given a design represented by $G_c = (V_c, A_c)$, our overall objective is twofold: 1) find a node subset $V_L \subset V_c$ and $|V_L| = M_e$ so the extensions of V_L nodes will give the best new comparability graph G_B among all the G_c' graphs in terms of power reduction; 2) find an edge subset in G_B that covers all the vertices in V_c in such a way that the sum of the edge weights in the subset is the minimum with the constraint that all the vertices can be bound into no more than k FUs. The first goal is voltage assignment, and the second goal is FU binding for reducing switching activity. We can see these two goals are intertwined because we cannot achieve the first goal without achieving the second goal or vice versa. The second goal of the objective can be formulated as a traditional clique partitioning problem. Each clique corresponds to the operations that are to be bound into a single FU. Although clique partitioning problem is NP-hard for general graphs, it is shown that we can find the minimum number of cliques required to bind all the nodes in polynomial time when working with comparability graphs [7]. In our work, k is the minimum number of FUs required. Early works proposed solutions to compute maximum weighted *k-cofamily* in partially ordered sets [6] and *k-covering* in weighted transitive graphs [22] through network flow formulations. Comparability graphs belong to transitive graphs [7] and can also be represented using partially ordered sets [3]. Therefore, there are previous works that used network formulation to solve various binding problems on comparability graphs [1][2][3][18]. In the next subsection, we will discuss more details of these early works, and then present our simultaneous voltage and module binding solution by computing the min-cost flow through network flow formulation.

B. Network Flow Formulation

Various binding algorithms have been proposed previously for reducing circuit power through network flow formulation. In [1], an optimal low-power register binding algorithm to reduce total switching activity was presented. It did not guarantee using the minimum number of k resources during the binding process though. In other words, its network-flow solution might not cover all the nodes in the comparability graph. In [2], the same authors formulated functional unit binding as a multi-commodity flow problem to reduce switching activity. The inter-frame binding constraints made the problem hard (to be discussed later). In [3], a register binding algorithm was presented to reduce total MUX connections in the design. It showed consistent positive impact on area, delay and power optimizations due to reduced interconnect usage. In [18], a single-commodity network flow was used to solve the bus binding problem with improved run time. It then presented a heuristic to fulfill the inter-frame binding constraints and showed promising results. None of these works considered dual Vdds in their formulations. In this work, we will build voltage assignment into our formulation and show that we can assign the maximum number of operations to VddL under latency and resource constraints and achieve min-power functional unit binding simultaneously. We always guarantee that we use no more than k resources. Thus, our objective in Subsection *A* will be achieved.

A network $N_G = (s, t, V_n, E_n, C, K)$ is constructed based on the comparability graph $G_c = (V_c, A_c)$. This is an extension to the one used in [1], where we introduce extra vertices to provide voltage assignment consideration. First, there are source vertex s and sink vertex t. The additional edges are edges from s to every vertex in V_c, and from every vertex in V_c to t. Second, for each extendable vertex v in V_c, there is an extra node v' connecting to v. There are additional edges between v' to the vertices comparable with it, and an additional edge between v' to t. N_G has the cost function C and the capacity K defined on each edge in E_n. Fig. 3 shows an example. Fig. 3(a) is the comparability graph corresponding to the DFG in Fig. 2(a). Fig. 3(b) is the graph N_G for 3(a).



(a) Comparability Graph G_c
(b) Corresponding N_G

Fig. 3: A comparability graph and its network graph

In Fig. 3(b), there still exist edges between nodes 1, 2, 3 and nodes 4, 5, 6 exactly as they do in Fig. 3(a). They are not drawn simply because it is too crowded. The dark edges in 3(b) represent that after node 1 or 2 become extended, they are still comparable with node 4 or 5. However, after node 3 becomes extended, it is no longer comparable with other nodes. Let V_e denote the set of all the extendable nodes in V_c. We have $V_e \subset V_c$. We use the symbol $\rightarrow$ to represent that two vertices are comparable with each other. Formally, the network $N_G = (s, t, V_n, E_n, C, K)$ is defined as the following:

$$V_n = V_c \cup \{s, t\} \cup \{v' \mid v \in V_e\}$$
$$E_n = A_c \cup \{(s, v), (v, t) \mid v \in V_c\} \cup \{(v, v'), (v', t) \mid v \in V_e\}$$
$$\cup \{(v_i', v_j) \mid v_i' \rightarrow v_j ; v_i \in V_e ; v_j \in V_c\}$$
$$C(s, v) = 0 \mid v \in V_c$$
$$C(v, t) = 0 \mid v \in V_c$$
$$C(v', t) = 0 \mid v \in V_e$$
$$C(v_i, v_j) = -L \times (1 - s_{ij}) \mid v_i \rightarrow v_j ; v_i, v_j \in V_c$$
$$C(v_i', v_j) = -L \times (1 - s_{ij}) \mid v_i' \rightarrow v_j ; v_i \in V_e ; v_j \in V_c$$
$$C(v, v') = -T \mid v \in V_e$$
$$K(e_n \mid e_n \in E_n) = 1$$

where C is the cost assigned on the edges and K is the capacity on the edges. s_{ij} is the switching activity on the edge (v_i, v_j). L is a positive constant and is set as 100. L is used to scale the costs into integer numbers. To maximize the number of extended operations, we need to guarantee that $C(v_i, v_i') + C(v_i', v_j) < C(v_i, v_j)$. That is the reason that $C(v, v')$ is set as $-T$, where $T = L \times |V_c|$. Value T guarantees that v will be extended if it is the only extendable node within resource constraint as an extreme case, no matter what the values of $C(v_i, v_j)$ are for the edges. Notice $s_{ij} < 1$ always. Therefore, we set the cost $C(v_i, v_j)$ as a negative value. The smaller s_{ij} is, the smaller $C(v_i, v_j)$ will be. Notice N_G captures all the possible configurations of G_c'.

Lemma 1: A flow f, with $|f| = 1$, in the network N_G corresponds to a clique χ in the original comparability graph G_c with voltage assignment. An edge (v_i, v_j) in the flow indicates operations v_i and v_j will be bound into the same FU U. An edge (v, v') in the flow indicates operation v will be assigned to VddL when executing in U.

Lemma 2: A flow f, with $|f| = k$, that passes through each and every node $v \in V_c$ by a unit flow is equivalent to finding k disjoint paths in N_G, thus generating k cliques in G_c with voltage assignment.

To guarantee that there is only a unit flow to go through each node $v \in V_c$, we can apply a node-splitting technique, which was adopted in [1] as well. This technique duplicates every vertex $v \in V_c$ in N_G into another node v^d. There is an edge from v to v^d. If there is an edge $(v_i, v_j) \in A_c$, there is an edge (v_i^d, v_j) in the new network called N_G^d. $C(v_i^d, v_j)$ is the same as $C(v_i, v_j)$. The original edge (v_i, v_j) is removed from N_G^d. Meanwhile, node v' will be connected to v^d instead of v. All the edges are assigned with a capacity of 1. In addition, we assign cost $C(v, v^d) = -X$, where X is a positive constant and $X \geq 2T$. We can show that this cost assignment will guarantee that all the nodes in V_c will be covered when the min-cost flow in N_G^d generates the binding and voltage assignment solution. Fig. 4 shows an example.

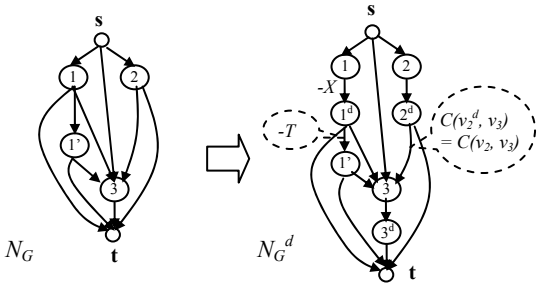


Fig. 4: A simple N_G and its split graph N_G^d

Theorem 1: The min-cost flow f, with $|f| = k$ on the network N_G^d gives the largest number of extended operations in the design with the minimum total switching activity on k functional units for the circuit represented by G_c.

This theorem holds when we ignore the inter-frame constraints presented in [2], which capture the switching activity in the cyclic executions of the DFG, i.e., the switching activity when a new set of vectors arrives on the inputs of the FUs to start execution from the beginning of the DFG again. However, we count these switches in our power estimation to make our experimental results more accurate. Our formulation can be easily extended to consider inter-frame constraints by building a multi-commodity flow network as shown in [2]. The min-cost multi-commodity flow solution will provide the largest extended-operation number and the minimum switching activity with inter-frame constraints. Since our goal is to show that we can achieve optimality under dual-Vdd consideration, multi-commodity flow is not the focus of this work. We do plan to add this extension in the future. Notice that our formulation can also be extended to consider more than two voltages. For example, to support three voltages, we can use new v'' nodes connecting to v nodes in N_G. v'' nodes will be similarly processed as v' nodes, and their associated costs can be designed and assigned. The min-cost flow will decide either picking v' or v'' nodes in its solution.

Our task then becomes finding the min-cost flow in the network N_G^d. It can be obtained through capacity scaling and successive shortest path computation and has running complexity $O(|E| \log k (|E|+|V|\log|V|))$. After we obtain the min-cost flow, each edge with a unit flow in N_G^d, (v_i^d, v_j), represents that operations v_i and v_j should be bound together into the same FU and v_i is operating under VddH. Each edge (v_i', v_j) represents v_i and v_j should be bound together and v_i is operating under VddL. If a flow passes $s \Rightarrow v \Rightarrow v^d \Rightarrow [v'] \Rightarrow t$, it represents that v is occupying a single FU just by itself. It operates either under VddH or VddL (when v' exists).

C. Power Gating

We follow a simple power gating scheme. After we obtain the binding solution, we search through the operations bound in each FU and find whether the FU is idle for a certain period of time (*idle_cycle*) that is longer than *SleepCycle* (Section III) between two consecutive operations. If this is the case, we count the static power saved during the number of cycles = *idle_cycle* – *SleepCycle*. This simple scheme is used because our main goal in this work is to reduce dynamic power. If static power reduction is the main goal, we

can modify our network flow formulation so the cost on an edge represents the idle cycles between the two operations on the edge. We expect that the max-cost flow solution from the network can dramatically increase the total idle time spent by functional units.

V. Experimental Results

We adopt a heuristic algorithm from [17] to perform the resource- and time-constrained scheduling to maximize the number of extended operations. The main idea is to iteratively make an operation extended, and then use a list scheduling algorithm to validate the choice. The choice is reversed if the extension violates constraints. This heuristic will generate voltage assignment along the way. Although dramatic increases of extended operations are observed, this algorithm does not guarantee to extend the optimal number of operations for the schedule it produces.

Since there is no existing algorithm that combines voltage assignment and switching activity reduction simultaneously, we will compare our algorithm, named *opti-dvdd*, with an experimental flow *sep-flow* set up by ourselves. *sep-flow* has two stages. First, it obtains the initial voltage assignment from the scheduling result. All the nodes with VddL assignment will be extended and a corresponding comparability graph is built. Second, we minimize the switching activity on the comparability graph as if we are working for the single-Vdd case. We use the binding algorithm presented in [1] for this stage because the algorithm gives an optimal binding solution without considering inter-frame constraints. However, its resource usage may exceed the minimum required number *k*. For *opti-dvdd*, we use the same schedule but ignore all the voltage assignments because *opti-vdd* will generate the optimal voltage assignment and binding simultaneously. To simulate the DFG for switching activity estimation on the edges, we use 1000 consecutive random input vectors. We carry out experiments based on real-life benchmarks from [23]. Both *opti-dvdd* and *sep-flow* have the power gating feature. The initial scheduling uses the tightest latency and resource bounds. Table 2 shows the results. We observe that the number of extendable nodes in the design usually is larger than the number of extended nodes. *opti-dvdd* always produces larger or equal number of extended operations than *sep-flow* does. The power values of *opti-dvdd* are consistently better than those of *sep-flow* (11.8% better on average). This is due to two reasons: 1) the initial voltage assignment of *sep-flow* is not optimal. Even for the cases where it extends the maximum number of operations, its choices may not be good because there is no switching activity considered; 2) binding of *sep-flow* sometimes exceeds the resources required. For example, *sep-flow* uses one more multiplier than *opti-dvdd* does for design *lee*.

Bench marks	total nodes	extend-able	*sep-flow* extended	*sep-flow* power(w)	*opti-dvdd* extended	*opti-dvdd* power(w)
air	422	211	79	4.4445	89	3.5015
chem	342	76	36	4.2972	39	3.8960
dir	127	32	23	2.0245	23	1.7430
honda	107	25	19	2.2949	19	1.9036
lee	49	15	15	0.7596	15	0.5861
mcm	94	10	8	2.7628	8	2.7565
pr	42	7	6	1.4163	6	1.4027
u5ml	565	183	119	3.3751	124	2.9798
wang	48	8	4	1.4583	6	1.3226

Table 2: Experimental results of our algorithm *opti-dvdd* vs. a heuristic algorithm *sep-flow*

To examine how dual-Vdd architecture itself helps on power reduction and gain some insights on power-latency tradeoffs, we carry out a series of experiments to compare *opti-dvdd* with an algorithm *opti-hvdd*. *opti-hvdd* only considers the single high Vdd. It uses the same network formulation as presented in Section IV but without extendable nodes (the *v'* nodes). The nodes in V_c are still split with cost assignment $C(v, v^d) = -X$. It will provide an optimal solution to minimize switching activity within the resource constraint for the single-Vdd case. To examine different trade-off scenarios, we change our initial scheduling to work with different latency bounds. Fig. 6 collects the results for comparisons on power and energy. The relaxed latency will be $(1+\alpha)$**CriticalPath*, where α is the relaxation percentage shown on the *x*-coordinate, and *CriticalPath* is the minimum number of clock cycles a scheduled DFG needs without any relaxation, i.e., its smallest critical path length. For example, suppose *CriticalPath* is 10 cycles for a design, $\alpha = 0.5$ will relax the latency of the design to 15 cycles. We still use the heuristic scheduling algorithm from [17]. The scheduling algorithm will take this new latency constraint and generate the schedule accordingly. The power and energy reduction percentages are average values over the benchmarks. We observe that we can achieve power and energy reduction of 21% over the single-Vdd case when there is no latency relaxation. The largest power reduction is 73% when latency is relaxed by 2X, i.e., 100% for the dual-Vdd case, comparing to the single-Vdd case with no latency relaxation (as the base). On the other hand, the energy reduction is 46% for the same relaxation. The percentage is smaller compared to that of power reduction because of the increased computation latency. The curve of energy reduction is not as steep as the power reduction curve due to the same reason.

The direct reason that latency relaxation can help on power and energy is that the maximum number of extended operations increases significantly along the relaxation. Another reason is that the number of resources required becomes smaller when more operations can share common resources due to latency relaxation. To find out how much savings dual-Vdd scheme itself achieves, we compare the power and energy consumptions between *opti-dvdd* and *opti-hvdd* for each individual relaxation point[3]. The power reduction percentages of *opti-dvdd* over *opti-hvdd* are 25%, 31%, 37%, 42%, and 43% along corresponding relaxation points from 10% to 100% respectively. This shows that dual-Vdd savings are significant. Energy comparison shows similar trends.

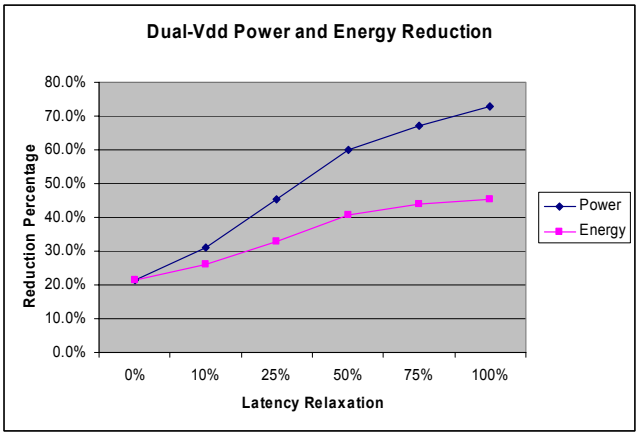


Fig. 6: Power and energy reduction results comparing *opti-dvdd* to the base *opti-hvdd* case with different latency relaxation

To understand how much power is saved due to power gating, we collect some data using 100% latency relaxation. The largest savings are for benchmark *pr*, where power gating provides 8% power savings for adders. However, power gating does not offer much savings overall and is almost negligible on average. We believe this

[3] Both *opti-dvdd* and *opti-hvdd* use the same latency and resource numbers for each relaxation point.

is due to the following reasons. First, power gating applied to a DFG doesn't have much opportunities compared to a CDFG (control data flow graph). For example, in a CDFG, once a conditional branch is not taken, the functional units executing the operations specifically in that branch can be completely turned off. This is not considered in DFG optimizations. Second, we treat power gating as a post processing step after dynamic power optimization as explained in Section IV. Therefore, the optimization space is reduced. Third, our *SleepCycle* is large because the leakage power percentage in our resources is not that significant. We plan to study power gating in the future that will take leakage power reduction as our first priority.

VI. Conclusions

In this paper we presented technologies for optimizing power considering dual Vdds and switching activity reduction simultaneously. We developed a polynomial-time optimal algorithm and showed that our algorithm was consistently better than an optimization flow that separated voltage assignment from functional unit binding. In addition, we studied power-latency tradeoffs and power-gating potentials.

Acknowledgements

This work is partially supported by Altera Corporation under the California MICRO program, and the NSF Grants CCR-0306682 and CCR-0096383.

References

[1]. J. M. Chang and M. Pedram, "Register Allocation and Binding for Low Power," *Design Automation Conf.*, 1995.

[2]. J. M. Chang, and M. Pedram, "Module Assignment for Low Power," *EURO-Design Automation Conf.*, 1996.

[3]. D. Chen, and J. Cong, "Register Binding and Port Assignment for Multiplexer Optimization," *Asian Pacific Design Automation Conf.*, Jan. 2004.

[4]. D. Chen, J. Cong, and Y. Fan, "Low-Power High-Level Synthesis for FPGA Architectures," *Intl. Sym. On Low Power Electronics and Design,* Aug. 2003.

[5]. D. Chen, J. Cong, F. Li, and L. He, "Low-Power Technology Mapping for FPGA Architectures with Dual Supply Voltages," *Intl. Sym. On Field Programmable Gate Arrays*, Feb. 2004.

[6]. J. Cong and C. L. Liu, "On the k-Layer Planar Subset and Topological Via Minimization Problems," *Trans. on CAD*, Vol. 10, Aug. 1991.

[7]. Giovanni De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, Inc., 1994.

[8]. R. P. Dilworth, "A Decomposition Theorem for Partially Ordered Set," *Ann. Math*, Vol.51, pp.161-166, 1950.

[9]. D. Duarte, Y. Tsai, N. Vijaykrishnan, and M. J. Irwin, "Evaluating Run-time Techniques for Leakage Power Reduction," *Intl. Conf. on VLSI Design*, 2002.

[10]. M. Hamada et al., "A Top-Down Low Power Design Technique Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme," *Custom Integrated Circuits Conf.*, 1998.

[11]. "International Technology Roadmap for Semiconductors," *Semiconductor Industry Association*, 2003.

[12]. M. C. Johnson, and K. Roy, "Datapath Scheduling with Multiple Supply Voltages and Level Converters," *Trans. on Design Automation of Electronic Systems*, 1997.

[13]. J. Kao, S. Narendra, and A. Chandrakasan, "Subthreshold Leakage Modeling and Reduction Techniques," *Intl. Conf. On Computer-Aided Design,* 2002.

[14]. F. Li, D. Chen, L. He and J. Cong, "Architecture Evaluation for Power-efficient FPGAs," *Intl. Sym. On Field Programmable Gate Arrays*, Feb. 2003.

[15]. F. Li, Y. Lin and L. He, "FPGA Power Reduction Using Configurable Dual-Vdd," *Design Automation Conf.,* Jun. 2004.

[16]. F. Li and L. He, "Maximum Current Estimation with Consideration of Power Gating," *Intl. Sym. On Physical Design*, April 2001.

[17]. Y. R. Lin, C. T. Hwang, and A. C. H. Wu, "Scheduling Techniques for Variable Voltage Low Power Design," *Trans. on Design Automation of Electronic Systems*, 1997.

[18]. C. G. Lyuh, and K. Taewhan, "High-level Synthesis for Low-Power Based on Network Flow Method," *Trans. on VLSI Systems*, 2003.

[19]. A. Manzak, and C. Chakrabarti, "A Low Power Scheduling Scheme with Resources Operating at Multiple Voltages," *Trans. on VLSI Systems*, 2002.

[20]. S. Mutoh et al, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," *Journal of Solid-State Circuits*, 1995.

[21]. S. Raje and M. Sarrafzadeh, "Variable Voltage Scheduling," *Intl. Sym. on Low Power Design*, 1995.

[22]. M. Sarrafzadeh and R. D. Lou, "Maximum k-Covering of Weighted Transitive Graphs with Applications," *Algorithmica*, Vol. 9, No. 1, pp. 84-100, 1993.

[23]. M. B. Srivastava and M. Potkonjak, "Optimum and Heuristic Transformation Techniques for Simultaneous Optimization of Latency and Throughput," *Trans. on VLSI Systems*, 1995.

[24]. M. Takahashi et al, "A 60mW MPEG4 Video Codec Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme," *Journal of Solid-State Circuits*, vol. 33, no. 11, 1998.

[25]. K. Usami, and M. Horowitz, "Clustered Voltage Scaling for Low-Power Design," *Intl. Sym. on Low Power Design*, 1995.

[26]. K. Usami, et al, "Automated Low-Power Technique Exploiting Multiple Supply Voltages Applied to a Media Processor," *Journal of Solid-State Circuits*, vol.33. no.3, Mar. 1998.