Optimality Study of Resource Binding with Multi-Vdds

Deming Chen
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, USA
dchen@uiuc.edu

Jason Cong, Yiping Fan
Computer Science Department
University of California, Los Angeles, USA
{cong, fanyp}@cs.ucla.edu

Junjuan Xu
Computer Science and Technology Department
Peking University, Beijing, PRC
junjuan@gmail.com

ABSTRACT
Deploying multiple supply voltages (multi-Vdds) on one chip is an important technique to reduce dynamic power consumption. In this work we present an optimality study for resource binding targeting designs with multi-Vdds. This is similar to the voltage-island design concept, except that the granularity of our voltage island is on the functional-unit level as opposed to the core level. We are interested in achieving the maximum number of low-Vdd operations and, in the same time, minimizing switching activity during functional unit binding. To the best of our knowledge, there is no known optimal solution to this problem. To compute an optimal solution for this problem and examine the quality gap between our solution and previous heuristic solutions, we formulate this problem as a min-cost network flow problem, but with special equal-flow constraints. This formulation leads to an easy reduction to the integer linear programming (ILP) solution and also enables efficient approximate solution by Lagrangian relaxation. Experimental results show that the optimal solution computed based on our formulation provides 7% more low-Vdd operations and also reduces the total switching activity by 20% compared to one of the best known heuristic algorithms that consider multi-Vdd assignments only.

Categories and Subject Descriptors
B.6.3 [Hardware]: Design Aids – *optimization*; **G.2.2 [Mathematics of Computing]**: Graph Theory – *network problems*

General Terms
Algorithms, Performance, Design, Experimentation

Keywords
Behavioral synthesis, resource binding, low power design

1. INTRODUCTION
There are two major sources of power consumption – dynamic power and leakage power. Dynamic power is consumed when signal transitions take place at gate outputs. Leakage power (or static power) is consumed when the circuit is either active or idle. With the current technology, dynamic power is still the major part of the total power consumption. It can be calculated as $P_d = 0.5 \cdot S \cdot C \cdot V_{dd}^2 f$, 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.

Deploying multiple supply voltages is one of the most effective techniques to reduce dynamic power under performance constraint. This technique has the advantage of reducing power dissipation without sacrificing the performance of the system. When the threshold voltage for the transistors stays as a constant, the delays of the resources become longer as the supply voltage scales down. Therefore, we can assign high-Vdd to critical paths to maintain high performance and low-Vdd to non-critical paths to save power. Clusters of high-Vdd cells and low-Vdd cells were first used in [21]. The work in [20] adopted multiple supply voltages in the real design of a MPEG4 video codec. Researchers are working from various abstraction levels to explore the advantages of multiple supply voltages. Specifically, there are works on system level [10][12][18], behavioral level [7][11][14][16], logic level [8][9], and physical level [13][22]. Our work focuses on power optimization at the behavioral level. In general, the higher the design level is, the more critical the design decisions are for the quality of the final product.

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 [17], an optimal solution was given for time-constrained scheduling problem under variable voltages. However, it did not consider resource constraints. The works in [11][14][16] 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 [4][5][15] minimized switching activity for various resources, such as functional units and buses. However, these works only targeted single-Vdd case.

In [7], a low-power binding algorithm for dual-Vdd designs was proposed to maximize the number of low-Vdd operations and minimize switching activity, with the assumption that voltages could be dynamically configured at run time for each functional unit. Each functional unit is connected to both low-Vdd and high-Vdd power supplies. Therefore, it is possible that a functional unit may execute an operation with low Vdd and then switch to high Vdd for execution of the next operation. Although dynamic voltage configuration for functional units may have a larger chance to achieve better power savings as it provides a larger flexibility for the voltage assignment of each operation in the design, this approach also introduces extra latency, area, and power costs. As mentioned in [7], the clock period needs to include extra time to accommodate the time required for voltage switching between different voltage configurations and thus leads to longer total latency. Voltage switching itself also consumes power and may cause undesirable signal noise, which was not modeled in [7]. In addition, it requires complex control logic and a full chip dual-rail power supply system, thus with a larger area overhead.

In this work we target architectures where the voltages of functional units are fixed. That is, the voltages of functional units are either high or low and can not change during run time. This approach is similar to the voltage-island concept [10][12], except that the granularity of our voltage island is on the functional-unit level as opposed to the core level. Fixed voltage configuration does not have problems discussed above for dynamically voltage-configurable functional units. Also, it may not require a full-chip dual-rail power supply system because each functional unit is only driven by a single Vdd. However, to the best of our knowledge, there is no known optimal solution to such multi-Vdd resource binding problem with fixed voltage configuration.

In this paper we concentrate on the dual-Vdd case (one low Vdd and one high Vdd) to demonstrate the optimality of our algorithm. Our method can be easily extended for more Vdds. Given a scheduled DFG (data-flow graph) and resource constraints, we transform the dual-Vdd binding problem into a network flow problem with extra constraints (equal integral flow constraints). We prove that when we achieve the min-cost flow under the equal-flow constraint, we achieve the optimal binding solution under dual-Vdds, i.e., maximizing the total number of low-Vdd operations and minimizing the total switching activity simultaneously. Although the min-cost flow problem with equal integral flow constraint is a difficult problem (NP-hard) [2], our formulation has several values: (i) It leads to an easy reduction to the integer linear programming formulation, so that we can use an exiting ILP solver to get an optimal solution; (2) It enables efficient approximate solution by other mathematical programming method, such as Lagrangian relaxation as shown in [3]; (3) We hope that our formulation provides a graph-theoretical framework for us and others to study the exact complexity of the problem (note that there is no conclusion yet whether the multi-Vdd resource binding problem with fixed voltage configuration is NP-hard or polynomial-time solvable); (4) Finally, using the optimal solution provided by the ILP solver, for the first time, we are able to measure the optimality gap of the previous heuristics to our solution.

The remainder of this paper is organized as follows. Sections II and III provide the problem formulation and network construction. Section IV proves the optimality of our algorithm and generates the binding and voltage assignment solution. Section V shows experimental results, and Section VI concludes this paper.

2. MOTIVATION AND PROBLEM FORMULATION

For data-intensive designs with behavioral description, the operations and their data dependencies can be represented by a data flow graph DFG, $G = (V, E)$. Set V corresponds to operations and set E corresponds to data flowing between operations. An edge $e = (x, y) \mid x, y \in V, a \in E$ indicates there is a data dependency between operation 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 bound. After scheduling, the life time of each operation in the DFG is the time during which the operation is active. The resource binding of operations of different type (e.g., addition and multiplication) is performed separately. We use V_f to represent the group of operations of type f. For two operations v_i and v_j of type f, if their corresponding life times do not overlap and operation v_i comes before v_j, we call operations v_i and v_j *compatible* with each other, and they can be bound into a single functional unit (FU) without life time conflicts. We denote this compatibility as $v_i \rightarrow v_j$. With the resource constraint, the available number of functional units of type f is denoted as N_f. We denote the minimum required number of functional units for V_f as Min_FU_f. Apparently, N_f should be $\geq Min\_FU_f$.

To consider dual-Vdd assignment on a scheduled DFG, we introduce several definitions (some are borrowed from [7]). An operation v is *extendable* if v can be assigned to the low Vdd and executed in multiple clock cycles, and the extended execution delay of v will not violate the overall latency constraint, and in the same time, the data dependencies between v and other operations are still valid. In other words, v will still generate its data in time so that the data can flow to all the other operations that require it. Among V_f operations, the extendable operations are denoted as V_e. If v is assigned to low Vdd in the final binding solution, we say v is *extended.* If v_i is still compatible with v_j after v_i is being assigned to low Vdd, we say that v_i and v_j are *extendable-compatible* and denote this case as $v_i \Rightarrow v_j$. Under the timing- and resource-constraints, and each FU can be assigned to either low-Vdd or high-Vdd, the maximum number of extended operations of type f is denoted as $MaxE_f$. Apparently, $MaxE_f \leq |V_e|$.

Table 1. Extendable and Maximum Extended Operations

Benchmark	Node#	$\lvert V_e \rvert$	*MaxE*	Ratio
aircraft	422	211	56	27%
chem	342	82	26	32%
dir	127	36	12	33%
honda	107	33	8	24%
lee	49	20	11	55%
mcm	94	11	5	45%
pr	42	12	2	17%
u5ml	565	184	35	19%
wang	48	16	6	38%
Avg.	-	-	-	**32%**

Due to the resource constraint, not all extendable operations can be extended eventually. We conducted experiments on a set of real-life benchmarks from [19], including several different DCT algorithms such as *pr*, *wang*, *lee*, and *dir*, and several DSP programs such as *aircraft*, *mcm*, *honda*, *chem*, and *u5ml*. The results are summarized in Table 1. The column "node#" lists the operation number for each benchmark. The column "$|V_e|$" lists the total number of extendable additions and multiplications under a scheduling solution. The column "*MaxE*" lists the maximum number of extended operations under timing and resource constraints. We use an integer-linear program (ILP) to get $MaxE_{add}$ and $MaxE_{mul}$ and then sum them up to get $MaxE$. We can see that the average ratio of the maximum extended operations over extendable operations is only 32%. This means that there are many different ways to select the maximum number of extended operations. Different voltage assignment leads to different compatibility of operations and thus different resource binding solution space. In addition, the total switching activity of the design will be affected by the voltage assignment as well. How to select the extended operations and at the same time minimize the switching activity is the problem to be solved in this paper. We formulate this problem as follows:

Given: (1) A scheduled DFG $G(V, E)$; (2) Cycle latency constraint Φ; (3) A set of available functional units R. Each functional unit can be configured to use a fixed voltage level: either VddH (high Vdd) or VddL (low Vdd); (4) Switching activity between each pair of compatible operations, ($0 \leq s_{i,j} \leq 1 \mid v_i$ and v_j have the same operation type).

Objective: Decide the voltage for each functional unit, and bind all operations to functional units under the timing- and resource-constraints, so that the number of operations with VddL is maximized and the total switching activities of functional units are minimized.

3. ALGORITHM DESCRIPTION AND NETWORK FLOW FORMULATION

Network flow formulation has been used previously for binding problems to reduce total switching activity [4][5][7][15]. In [4], an optimal low-power register binding algorithm was presented. In [5], the same authors formulated functional unit binding for low power as a multi-commodity flow problem. The inter-frame binding constraints made the problem hard. They then used ILP to solve the problem. In [15], a single-commodity network flow was used to solve the bus binding problem. It then presented a heuristic to fulfill the inter-frame binding constraints. None of these works considered dual-Vdds in their formulation. In [7], dual Vdds were considered during binding. It derived an optimal solution to achieve the maximum number of low-Vdd operations with switching activity minimization, with the assumption that voltages could be dynamically configured at run time for each functional unit.

In our work, since we have an extra constraint that each FU can only be driven by a single Vdd, the problem becomes harder. We need to guarantee that only extended operations can be bound together for low Vdd, and only un-extended operations can be bound together for high Vdd although all of these operations can be either extendable or un-extendable themselves. We will show that through proper network construction and an optimal solution based on the min-cost flow algorithm, we can assign the maximum number of operations to low Vdd with the minimum switching activity without violating the resource and latency constraints. The direct effect of our solution is to enlarge the total number of FUs driven by low Vdd while guaranteeing that all the FUs switch the minimum. Algorithm 1 highlights our overall algorithm for the dual-Vdd case.

```
Algorithm 1
Input: A scheduled DFG $G = (V, E)$; latency constraint $\Phi$; a set
    of resources $R$; VddH and VddL; switching activity $s_{ij}$ on $v_i \to v_j$, $v_i, v_j \in V$.
Output: A Vdd assignment on operations: $\{A_v$: Vdd to v | for all
    v, where $v \in V$ and Vdd $\in$ {VddH, VddL}}. We denote $v_{VddL}$ if
    v is assigned VddL and $v_{VddH}$ otherwise;
    A functional unit binding: $\{B_u$: v to u | for all v, where $v \in
    V$ and $u \in R\}$. $v_{VddL}$ nodes will be bound to u driven by VddL,
    and $v_{VddH}$ nodes will be bound to u driven by VddH.
Goal: Maximize the number of $v_{VddL}$ under constraints $\Phi$ and $R$;
    minimize the total switching activity on $R$.
Algorithm Outline:
    (1) From G, build a network $H_s$ enclosing extendable and
    compatible information with regard to the nodes in V;
    (2) Assign cost, capacity constraints to the edges of $H_s$;
    (3) Solve the min-cost flow problem on $H_s$ with a set of equal
    integral flow constraints;
    (4) Derive Vdd assignment and binding solution based on the
    solution in (3).
End
```

We will introduce these steps in the current and next sections in details. A network $H_s = (s, t, V_n, E_n, C, K_l, K_u)$ is constructed based on V_f in the DFG and the compatibilities of operations. First, there are source vertex s and sink vertex t. V_n is the vertex set, and E_n is the edge set of the network. We use extra vertices to provide voltage assignment consideration and control the voltage assignment for functional units. If $v \in V_f$ is not extendable, v has only one corresponding vertex in V_n. If v is extendable, v has three corresponding vertices in V_n, v, v_a and v_b. v_a and v_b are called *a-vertex* and *b-vertex* of v. Edges (v_a, v) and (v, v_b) are in E_n. s is connected to all v and v_a, and all v and v_b are connected to t. The connection between v_i and $v_j \in V_f$ can be classified into the following five categories:

1. If $v_i \to v_j$, and both nodes are not extendable, (v_i, v_j) is in E_n.
2. If $v_i \Rightarrow v_j$, and both nodes are extendable, (v_i, v_j) and (v_{ib}, v_{ja}) are in E_n.
3. If v_i and v_j are extendable, $v_i \to v_j$, but $v_i \,!\!\Rightarrow v_j$, only (v_{ib}, v_{ja}) is in E_n.
4. If v_i is not extendable, v_j is extendable, and $v_i \to v_j$, (v_i, v_{ja}) is in E_n.
5. If v_i is extendable, v_j is not extendable, and $v_i \to v_j$, (v_{ib}, v_j) is in E_n.

Each edge is associated with a weight, which represents the cost of binding two operations into a single FU with the voltage assignment on the FU. This cost is the estimated switching activity between these two operations when they execute one after another.

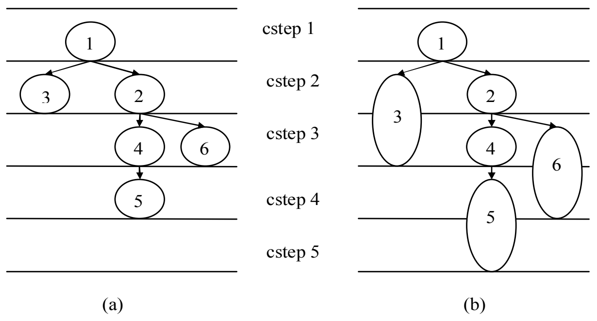

Figure 1. A Scheduled DFG

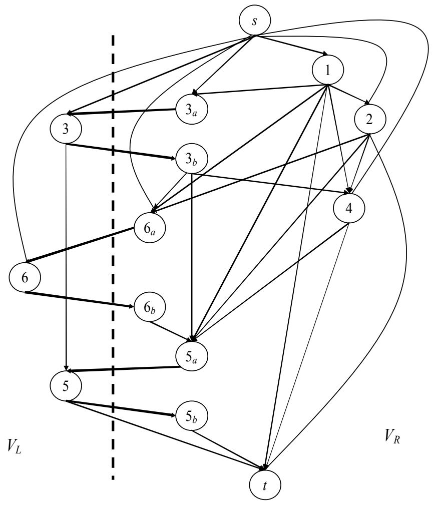

Figure 2. Network H_s

In the following, we will use the example in Figure 1 to illustrate the construction of H_s. Figure 1(a) is a scheduled DFG including six additions, among which operations 3, 5 and 6 are extendable, and operations 1, 2 and 4 are not extendable (Figure 1 (b)). Here we assume additions with VddH need 1 cycle and VddL need 2 cycles to finish the operation. The constructed network is shown in Figure 2. For the purpose of simplification, edges $(s, 5)$, $(s, 5_a)$, $(3, t)$, $(3_b, t)$, $(6, t)$ and $(6_b, t)$ are not drawn in the network. Operations 3 and 5 are extendable, and after extension of operation 3, they are still compatible. Therefore, edges $(3, 5)$ and $(3_b, 5_a)$ exist in E_n (category 2). Operation 6 is extendable, but operation 3 is not compatible with it after extension. Therefore, there is only edge $(3_b, 6_a)$ in E_n (category 3) and no edge from operation 3 to operation 6. Similarly, operations 6 and 5 are compatible only before their extension, so there is only the edge $(6_b, 5_a)$.

In Figure 2, except node s and t, the other nodes of H_s are divided into two parts. The left part includes nodes corresponding to extendable operations, denoted as V_L. The right part is denoted as V_R, including un-extendable nodes and the a-vertex and b-vertex of extendable operations. The flow through a node $v_l \in V_L$ can only come from s, other nodes in V_L, or v_{la}. The outcome flow of v_l can only go to t, other nodes in V_L, or v_{lb}. There is no edge from $v_r \in V_R$ to $v_l \in V_L$. Their connection has to go through node v_{la}. In a similar way, there is no edge from v_l to v_r either. Their connection has to go through node v_{lb}. We can treat v_a and v_b as the connecting nodes between $v_r \in V_R$ and $v_l \in V_L$.

To assign voltages for operations using the network flows of H_s, we define that if a unit flow through $v_l \in V_L$ comes from s or other nodes in V_L, v_l will be assigned to VddL; otherwise, v_l will be assigned to VddH. Node $v_r \in V_R$ always works under VddH. In addition, all the operations visited by one single unit flow will be bound to the same functional unit. Since each functional unit can only have one voltage, those operations bound together will have the same voltage assignment. To guarantee all the operations visited by a unit flow are assigned to the same voltage, we require that edges (v_{la}, v_l) and (v_l, v_{lb}) either both exist in the unit flow, or none of them exists in the flow. With this additional constraint on the network flow, if a unit flow coming into v_l originates from v_{la}, this flow has to go to v_{lb}. If a unit flow coming into v_l originates from s or other nodes in V_L, this flow has to go to t or other nodes in V_L. In this way, a flow coming from V_R region would go back to V_R region eventually, and a flow in V_L region will always stay in V_L region. We call a unit flow satisfying the above constraint a *valid* flow. In the following, all the flows are valid if not specified.

The formal construction of $H_S = (s, t, V_n, E_n, C, K_l, K_u)$:

$$
\begin{aligned}
V_n &= V_f \cup \{s, t\} \cup \{v_a, v_b \mid v \in V_e\}\\
E_n &= \{(s, v), (v, t) \mid v \in V_f\}\\
&\quad \cup \{(s, v_a), (v_b, t), (v_a, v), (v, v_b) \mid v \in V_e\}\\
&\quad \cup \{(v_i, v_j) \mid v_i \rightarrow v_j;\ i \neq j;\ v_i, v_j \in V_f - V_e\}\\
&\quad \cup \{(v_i, v_j), (v_{ib}, v_{ja}) \mid v_i \Rightarrow v_j;\ i \neq j;\ v_i, v_j \in V_e\}\\
&\quad \cup \{(v_{ib}, v_{ja}) \mid v_i \rightarrow v_j;\ v_i \,!\!\Rightarrow v_j;\ i \neq j;\ v_i, v_j \in V_e\}\\
&\quad \cup \{(v_i, v_{ja}) \mid v_i \rightarrow v_j;\ i \neq j;\ v_i \in V_f - V_e;\ v_j \in V_e\}\\
&\quad \cup \{(v_{ib}, v_j) \mid v_i \rightarrow v_j;\ i \neq j;\ v_i \in V_e;\ v_j \in V_f - V_e\}\\
C(s, v) &= -L \mid v \in V_f - V_e\\
C(s, v_a) &= -L \mid v \in V_e\\
C(s, v) &= -T \mid v \in V_e\\
C(v, t) &= 0 \mid v \in V_f\\
C(v_b, t) &= 0 \mid v \in V_e\\
C(v_a, v) &= 0 \mid v \in V_e\\
C(v, v_b) &= 0 \mid v \in V_e\\
C(v_i, v_j) &= -L \times (1 - s_{i,j}) \mid v_i \rightarrow v_j;\ i \neq j;\ v_i, v_j \in V_f - V_e\\
C(v_i, v_j) &= -T - L \times (1 - s_{i,j}) \mid v_i \Rightarrow v_j;\ i \neq j;\ v_i, v_j \in V_e\\
C(v_{ib}, v_{ja}) &= -L \times (1 - s_{i,j}) \mid v_i \rightarrow v_j;\ i \neq j;\ v_i, v_j \in V_e\\
C(v_i, v_{ja}) &= -L \times (1 - s_{i,j}) \mid v_i \rightarrow v_j;\ i \neq j;\ v_i \in V_f - V_e;\ v_j \in V_e\\
C(v_{ib}, v_j) &= -L \times (1 - s_{i,j}) \mid v_i \rightarrow v_j;\ i \neq j;\ v_i \in V_e;\ v_j \in V_f - V_e\\
f(v_a, v) &= f(v, v_b) \mid v \in V_e\\
K_u(e_n) &= 1 \mid e_n \in E_n\\
K_l(e_n) &= 0 \mid e_n \in E_n
\end{aligned}
$$

where C is the cost assigned on the edges; K_l is the flow capacity lower bound for an edge; and K_u is the flow capacity upper bound. 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 set $C(v_i, v_j)$ as $-T - L \times (1 - s_{i,j})$ when $v_i \Rightarrow v_j$, and $C(s, v)$ as $-T$ for $v \in V_e$, where $T = L \times (|V_n| + 1)$. Value T guarantees that v will be extended if it is the only extendable node within resource constraint as an extreme case. Notice $s_{ij} \leq 1$ always. Therefore, we set the cost $C(v_i, v_j)$ as a non-positive value. The smaller s_{ij} is, the smaller $C(v_i, v_j)$ will be. The maximum capacity of each edge is 1, and the flow of (v_a, v) and (v, v_b) must be equal. From the definition of cost C, it is easy to see that the resource binding solution derived from the network flow with smaller total cost has less total switching activity.

To guarantee that there is only a unit flow to go through each node $v \in V_f$[1] we can apply a node-splitting technique, which was adopted in [7] as well. This technique duplicates every vertex $v \in V_f$ in H_s into another node v^d. There is an edge from v to v^d. All the edges coming out from v will be connected to v^d instead. Both the flow capacity lower bound and upper bound are 1 for the edge (v, v^d). We denote the network after splitting as H_S^d. Figure 3 shows the split version of H_S from Figure 2. For clearness, only part of H_S^d is shown.

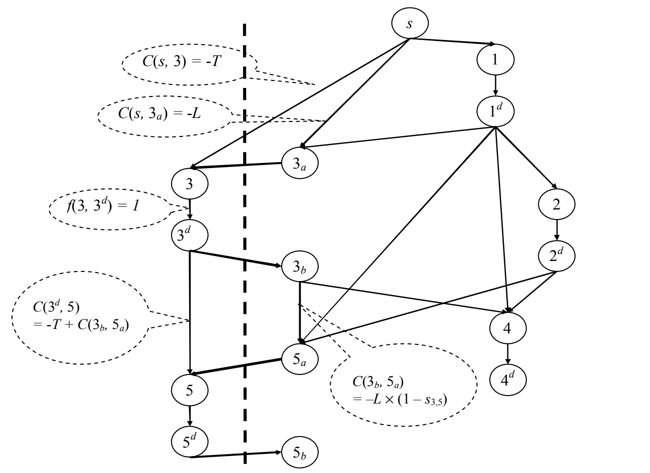

Figure 3. The Split Network H_S^d

4. SOLUTION GENERATION AND OPTIMALITY PROOF

In this section, we will prove that the voltage assignment and resource binding solution decided by the min-cost N_f-flow in H_S^d produces a solution that has the maximum number of extended operations and the minimum total switching activity on functional units after binding. For any proper resource binding P of V_f, we define the *basic cost* of P as:

$$\Omega(V_f, P) = \sum -L \times (1 - S_{ij})$$

where v_i and v_j are bound to a common functional unit and v_j executes right after v_i.

Lemma 1: The following inequality holds: $-T < \Omega(V_f, P) - L \cdot b \leq 0$, where $0 \leq b \leq |N_f|$.

Lemma 2: If a *flow* is a *valid* unit-flow in the network H_S^d, all the operations visited by the *flow* either all work with VddH or all with VddL. Conversely, the operations that can be bound to the same functional unit with a voltage assignment correspond to a *valid* unit-flow in H_S^d.

Lemma 3: If a *flow* is a k-flow in H_S^d, $k \geq Min\_FU_f$, *flow* must be

[1] This is to guarantee that we will have a legal binding solution. If two unit flows enter the same node, it means that the operation represented by the node is assigned to two different functional units, which is illegal.

composed of k disjoint unit-flows and include all operations in V_f. Also, *flow* can decide a voltage assignment and resource binding solution for V_f.

Theorem 1: The min-cost N_f-flow of H_S^d, *min-flow*, binds V_f to N_f functional units. In this solution, $MaxE_f$ operations are assigned to voltage VddL, and the total switching activity of all functional units are minimized.

The proofs of the above lemmas and theorem are omitted due to space limit.

If there are three available voltages, we can extend the network by adding one more region to the left of region V_L and adding needed a-vertices and b-vertices into V_L and V_R. In the same way, we can extend H_S^d to accommodate more voltages.

Notice that Theorem 1 will not guarantee that the solution will always produce the minimum dynamic power although in general it will be the case because power reduction due to voltage scaling is in quadratic order. Low-Vdd extensions will change the compatible relations among the nodes in the original graph so it is possible that we should only extend $MaxE_f - 1$ number of nodes if the effect of total switching activity reduction by doing so overweighs the effect of one less low-Vdd operation. We can actually change our network formulation slightly to achieve the minimum dynamic power solution theoretically. The only changes will be on the cost assignments. There is no $-T$ in the cost any more. Instead, there is a quantity measuring the actual dynamic power reduction ratio due to voltage scaling. We can have the following two different cost assignments:

$$C(s, v) = -L \times (VddH^2 / VddL^2) \mid v \in V_e$$

$$C(v_i, v_j) = -L \times (1 - s_{i,j}) \times (VddH^2 / VddL^2) \mid v_i \Rightarrow v_j; \quad i \neq j; v_i, v_j \in V_e.$$

Other costs are kept the same as before. We can prove that the *min-flow* in **Theorem 1** with these new cost assignments will provide us the minimum dynamic power solution. However, there may be less extended nodes in the solution compared to the number $MaxE_f$. In reality, this may not be desirable because we cannot guarantee that the switching activities are estimated 100% accurate. In practice, we believe people would prefer having the largest number of extensions because it will guarantee to reduce power in a large scale no matter how inaccurate the estimated switching activity will be. In our experimental results, we only show the case where we achieve $MaxE_f$ low-Vdd extensions.

Traditionally, the min-cost network flow can be solved by shortest path based algorithms [1]. However, except the general characteristics of networks, H_S^d requires that the flow on edges (v_a, v) and (v, v_b) are equal. The min-cost flow problem with equal integral flow constraint is a difficult problem (NP-hard) [2]. First of all, after we form the flow network, it becomes straight forward to reduce the problem into an ILP problem so we can take advantage of existing ILP solvers. Meanwhile, we are aware of the potential drawback of high runtime complexity of ILP. Since we are interested in the optimality study for our problem, we take the ILP approach in our experimental results. We observe that the average runtime for benchmarks with less than 100 nodes is 0.01s. However, for large designs with more than 100 nodes, the average runtime can be in the range of several hours. To trade off solution quality with runtime, we can also use heuristic algorithms. One such an algorithm was presented in [3], where the authors used a Lagrangian relaxation technique to tackle the min-cost equal-flow problem. They could solve problems with up to 1500 nodes and 6000 edges. They reported solutions within 1% of the optimum with 1%-65% of the runtime compared to a traditional branch and bound algorithm. Applying this heuristic and other heuristics and observing their quality/runtime tradeoff potentials belong to our future work.

5. EXPERIMENTAL RESULTS

To examine the quality gap between our optimal solution and heuristic solutions, we implemented a heuristic algorithm published in [14] for comparison purpose. The main idea in [14] was to perform the resource- and time-constrained scheduling to maximize the number of extended operations for low Vdd. It started with an initial schedule assuming every operation was using VddH. Then, it iteratively assigned VddL to operations followed by a validation step using a list scheduling algorithm. The low-Vdd assignment was reversed if the operation extension would violate constraints. In other words, [14] implicitly bound operations to functional units and thus decided the voltages for these functional units under the resource constraint. Therefore, at the end of scheduling, a binding and voltage assignment solution was also generated. Although this algorithm dramatically increased the number of extended operations compared to a generic list scheduling without such a voltage assignment feature, it could not guarantee to extend the maximum number of operations for the scheduling solution it produced. In addition, there was no switching activity considered in this process.

We use the benchmarks mentioned in Section II and compare our solution to the solution generated by [14]. To have a fair comparison, we use the same scheduling result generated by [14] but ignore its voltage and binding solution. We then run our algorithm honoring the time and resource constraints specified by the schedule. We take the voltage and timing data from [7], where adders need 1 cycle at VddH and 2 cycles at VddL, and multipliers need 3 cycles at VddH and 5 cycles at VddL. VddH is 1.3v and VddL is 0.8v under 100nm technology. We use simulation-based method with random input vectors to estimate switching activities between different operations. The simulation is similar to that used in [6]. The timing constraint for both [14] and our algorithm is the critical path latency (can be determined by an ASAP scheduling algorithm at the beginning).

Table 2 lists the results of extended operations. The second and third columns are the numbers of extendable additions and multiplications. The fourth and fifth columns list the numbers of extended operations by the algorithm in [14]. The last two columns list the results, $MaxE_f$, from our algorithm. For certain benchmarks, we observe a large percentage of improvement (e.g., 67% improvement on multiplications for *mcm*). On average, we observe 7% improvement over [14]. This indicates that the number of VddL extensions decided by [14] is already close to the optimal values for this set of benchmarks.

Table 3 summarizes switching activity (SA) and power results. The column *Lower Bound* shows the minimum switching activities when all operations work at VddH. These data can be calculated by the algorithm in [4], which is an optimal resource binding algorithm for switching activity reduction under single Vdd. We denote the binding solution space as Q_H when all operations work at VddH. After extending some operations, the new binding solution space Q_{HL} would be equal or smaller than Q_H, since some compatible operations with VddH may not be compatible any more after assigning to VddL, and thus the solution space is reduced. Therefore, given a scheduled DFG, the minimum switching activity of any binding solution after extending some operations must be equal to or greater than the results given by [4]. We need to mention that given the maximum number of VddL extensions (i.e., given a solution space Q_{HL}), our algorithm can achieve the minimal switching activity as shown in Section IV. The *heuristic* column corresponds to the binding solutions in [14], and the *min-flow* column lists the result given by our algorithm. We can observe that our algorithm achieves obvious improvement in switching activity reduction. On average, our switching activity result is 20% smaller and our power result is 23% smaller (power reduction is larger due to larger number of low-Vdd operations in our solution). Also, compared to switching activity *lower bound*, our algorithm is only 4% higher, which shows that we do not need to sacrifice much on switching activity for assigning low-Vdd to operations.

It is worth mentioning that because both *heuristic* and *min-flow* have the same resource bound, the leakage values of the designs from these two algorithms are similar. When *min-flow* uses a larger number of VddL functional units, the leakage power of the design actually will become smaller than what *heuristic* does. The reason behind this is that when threshold voltage is maintained as a constant as the case in our study, leakage power scales down for VddL due to the scaling of V_{DS} (drain/source potential difference) and V_{GS} (gate/source potential difference).

Table 2. Comparison of Extended Operations

Bench-marks	Max Extendable		Heuristic [14]		Min-flow	
	ADD	MUL	ADD	MUL	ADD	MUL
aircraft	31	180	0	56	0	56
chem	5	77	5	21	5	21
dir	2	34	0	12	0	12
honda	7	26	4	4	5	4
lee	12	8	5	6	6	6
mcm	5	6	2	3	2	5
pr	6	6	0	2	0	2
u5ml	29	155	7	28	8	28
wang	10	6	4	2	4	2

Table 3. Comparison of Switching Activities (SA) and Power

Bench-marks	Lower Bound	Heuristic [14]		Min-flow	
	SA	SA	Pow(W)	SA	Pow(W)
aircraft	0.0474	0.0602	4.695	0.0484	3.648
chem	0.0445	0.0557	3.716	0.0450	2.971
dir	0.0421	0.0520	1.213	0.0435	0.986
honda	0.0441	0.0508	1.006	0.0443	0.880
lee	0.0780	0.1047	0.873	0.0826	0.561
mcm	0.0150	0.0221	0.346	0.0163	0.255
pr	0.0838	0.1384	0.914	0.0936	0.618
u5ml	0.0431	0.0575	6.670	0.0434	4.967
wang	0.1020	0.1190	1.006	0.1044	0.863
Avg.	-	**1**	**1**	**-20%**	**-23%**

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented a network flow-based formulation for the low-power resource binding problem, considering multi-Vdds and switching activity simultaneously, while assuming that each functional unit has a fixed Vdd level. Experimental results show that the optimal solution computed based on our formulation provides 7% more low-Vdd operations and also reduces the total switching activity by 20% compared to one of the best known heuristic algorithms that consider multi-Vdd assignments only. Currently, we are studying the exact complexity of this problem based on our network flow formulation. Moreover, we are considering operational chaining and pipelined functional units into our problem formulation.

7. ACKNOWLEDGEMENTS

This work is partially supported by the National Science Foundation under the grants CCR 0096383 and CCR 0306682.

8. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, 1993.

[2] R. K. Ahuja, J. B. Orlin, G. M. Sechi, and P. Zuddas, “Algorithms for the Simple Equal Flow Problem,” *Management Science*, 45(10):1440~1455, 1999.

[3] A. I. Ali, J. Kennington, and B. Shetty, “The Equal Flow Problem,” *European Journal of Operational Research*, 36, 107~115, 1988.

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

[5] J. M. Chang and M. Pedram, “Module Assignment for Low Power,” *Conf. on European Design Automation*. 1996. 376~381.

[6] D. Chen, J. Cong, and Y. Fan, “Low-Power High-Level Synthesis for FPGA Architectures,” *Intl. Symp. on Low Power Electronics and Design*. 2003. 134~139.

[7] D. Chen, J. Cong, and J. Xu, “Optimal Module and Voltage Assignment for Low-Power,” *IEEE/ACM Asia South Pacific Design Automation Conf.* 2005. 850~855.

[8] D. Chen, J. Cong, F. Li, and L. He, “Low-Power Technology Mapping for FPGA Architectures with Dual Supply Voltages,” *Intl. Symp. on Field-Programmable Gate Arrays*. 2004. 109~117.

[9] D. Chen and J. Cong, “Delay Optimal Low-Power Circuit Clustering for FPGAs with Dual Supply Voltages,” *Intl. Symp. on Low Power Electronics and Design*. 2004. 70~73.

[10] J. Hu, Y. Shin, N. Dhanwada, and R. Marculescu, “Architecting Voltage Islands in Core-based System-on-a-Chip Designs,” *Intl. Symp. on Low Power Electronics and Design*, 2004. 180~185.

[11] M. C. Johnson and K. Roy, “Datapath Scheduling with Multiple Supply Voltages and Level Converters,” *ACM Trans. on Design Automation of Electronic Systems*. 1997. 2(3): 227~248.

[12] D. E. Lackey et al, “Managing Power and Performance for System-on-Chip Designs Using Voltage Islands,” *Intl. Conf. on Computer-Aided Design*. 2002. 195~202.

[13] F. Li, Y. Lin, and L. He, “FPGA Power Reduction Using Configurable Dual-Vdd,” *Design Automation Conf.* 2004. 735~740.

[14] Y. R. Lin, C. T. Hwang, and A. C. H. Wu, “Scheduling Techniques for Variable Voltage Low Power Design,” *ACM Trans. on Design Automation of Electronic Systems*. 1997. 2(2): 81~97.

[15] C. G. Lyuh and K. Taewhan, “High-level Synthesis for Low-Power Based on Network Flow Method,” *IEEE Trans. on VLSI Systems*. 2003. 11(3): 364~375.

[16] A. Manzak, and C. Chakrabarti, “A Low Power Scheduling Scheme with Resources Operating at Multiple Voltages,” *IEEE Trans. on VLSI Systems*. 2002. 10(1): 6~14.

[17] S. Raje and M. Sarrafzadeh, “Variable Voltage Scheduling,” *Intl. Symp. on Low Power Design*. 1995. 9~14.

[18] T. Simunic et al, “Dynamic Voltage Scaling and Power Management for Portable Systems,” *Design Automation Conf.* 2001. 524~529.

[19] M. B. Srivastava and M. Potkonjak, “Optimum and Heuristic Transformation Techniques for Simultaneous Optimization of Latency and Throughput,” *IEEE Trans. on VLSI Systems*. 1995. 3(1): 2~19.

[20] M. Takahashi et al, “A 60mW MPEG4 Video Codec Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme,” *Journal of Solid-State Circuits*, 1998. 33(11): 1772~1780.

[21] K. Usami and M. Horowitz, “Clustered Voltage Scaling Technique for Low Power,” *Intl. Symp. on Low Power Design*. 1995. 3~8.

[22] J. Wang, S. Shieh, J. Wang, and C. Yeh, “Design of Standard Cells Used in Low-power ASIC's Exploiting the Multiple-Supply-Voltage Scheme,” *IEEE Intl. ASIC Conf.* 1998.