Simultaneous Multiple-V$_{dd}$ Scheduling and Allocation for Partitioned Floorplan

Dongku Kang, Mark C. Johnson and Kaushik Roy
School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana, 47907-1285
{dkang,mcjohnso,kaushik}@purdue.edu

Abstract

In this paper, we propose a simultaneous scheduling and allocation algorithm for voltage-partitioned multiple-V$_{dd}$ design. By considering voltage partition during scheduling and allocation, we may place the resources of same voltage in one partition, thereby reducing additional power meshes. Also, the partitioned design reduces the energy dissipation of level converters by reducing cutsize between different-voltage partitions. The proposed algorithm starts from a random solution. Then, it performs scheduling and allocation simultaneously while trying to satisfy both resource and time constraints. By gradually changing the schedule and allocation, the algorithm effectively explores solution spaces to achieve low-power and better partitioning in terms of the supply voltages. Relative to the minimum single voltage design, 36% of energy saving was achieved. Also, improvements for interconnect, level-conversion energy, and voltage clusters were observed.

1 Introduction

Power consideration is an important design issue for the modern portable devices. For static CMOS circuits, energy dissipation is dominated by switching power, which is proportional to the square of the supply voltage as shown in Eq.1, where α, C_L, and F_{clk} denote switching activity, load capacitance, and operating frequency.

$$P_{avg} = \alpha V_{dd}^2 C_L F_{clk} \tag{1}$$

Among various technologies for low-power, voltage scaling is an effective technique to reduce the energy dissipation. However, reduced supply voltage results in the increased propagation delay [1]. Therefore, the global reduction in supply voltage slows down the speed of the circuits, thereby degrading the computational throughput. To maintain the computational throughput with reduced energy dissipation, multiple-V$_{dd}$ design was introduced. In multiple-V$_{dd}$ design, a lower supply voltage is applied to the circuits that can be slowed down, while a higher voltage supply is applied to the circuits in the critical paths. For data-flow graphs (DFG), non-uniform path lengths can result in timing slacks in schedule. In Fig.1, node and edge represent operation and data transfer in DFG. As an example, node 4 can be scheduled in either first or second clock-cycle because the result of node 4 is necessary for node 3, which is scheduled in the third clock-cycle. Using the unused time slack, we can schedule node 4 in the lower voltage resource, which has two clock-cycle delay. Therefore, by scheduling nodes with unused time slack in the lower voltage resources, one can reduce the energy dissipation while maintaining the computational throughput.

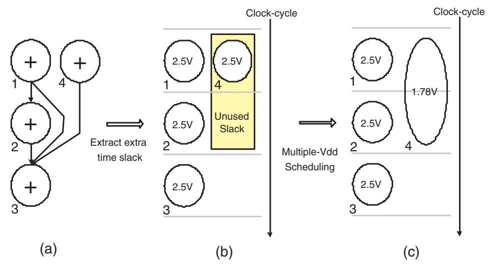

Figure 1. A DFG and multiple-V$_{dd}$ scheduling

Various efforts related to the multiple-V$_{dd}$ design have been made. Usami and Horowitz [2] proposed a design technique to reduce the energy consumption in a circuit by using two supply voltages. The gates on the critical paths are operated at the higher supply voltage, while the gates on the non-critical paths are at the lower supply voltage. Chang and Pedram [3] proposed an algorithm to minimize the energy dissipation in high-level synthesis with time con-

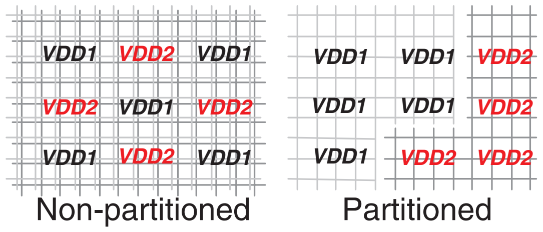

Figure 2. The concept of the voltage-partitioned design.

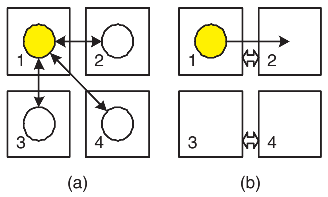

Figure 3. Possible movements in the partitioning algorithms. (a) K-KL (b) K-PM/FM

straint. Johnson and Roy [4] used an integer-linear programming (ILP) formulation with the resource and time constraints. Sarrafzadeh and Raje [5] proposed a dynamic programming approach. Manzak [6] presented scheduling algorithms with resource and time constraints with low complexity.

However, there exist several disadvantages when we design using multiple-V$_{dd}$. First, if we do not partition the voltage domains, then we are forced to either interleave the power rails or use an additional metal layer as shown in Fig.2. Second, we have to consider extra communication cost due to the level converter. Third, there may be noise issues. When two interconnects which operate at different logic levels are coupled, there can be a cross-talk problem. Also, when the higher voltage circuits inject larger amount of current into the substrate, the noise tolerance of lower voltage circuits will be affected.

Therefore, to realize multiple-V$_{dd}$ design, we need additional considerations other than module energy dissipation. By removing the edges between different voltage domains in a DFG, we can reduce the interactions between the resources that operate in the different voltage domains. Hence, we can separate resources with respect to the supply voltage. Also, when generating a floorplan, the reduced interactions between different voltage resources will promote the voltage-partitioned floorplan as shown in Fig.2, which is well partitioned by their respective supply voltage. Furthermore, when the edges between different supply voltages are removed, we can reduce the energy dissipation due to the level converter.

Recently, Kang et al. [7] presented a scheduling and allocation scheme for voltage partitioned floorplan, which is based on the graph-partitioning problem [8], [9]. They generated an initial schedule to determine the supply voltages for each resource. Also, nodes are allocated to serve as an initial solution for partitioning algorithm. By swapping supply voltages for pairs of nodes, the modified Kernighan-Lin algorithm (K-KL) [7] minimized the number of edges between different supply voltages, while maintaning the resource and time constraints. Also, K-KL reduced the energy and number of level converters. Furthermore, it generated a floorplan, which is well partitioned by the supply voltages.

In this paper, we present a different approach for the voltage-partitioned multiple-V$_{dd}$ design, which does scheduling and allocation simultaneously. Starting from a random allocation, proposed partitioning algorithm changes the schedule and the allocation for better partitioning. Also, to reduce the energy dissipation, it schedules as many nodes as possible to lower voltage resources. For partitioning, we adapted Fiduccia-Mattheyses algorithm (FM) [10], which moves a single node at a time. To decide movements for each node, we calculate gains for each node. When moving a node that maximizes the gain, the resource allocation is modified. Then, a list scheduler quickly generates a schedule with the allocation defined by the partitioning algorithm. Compared to $O(n^3)$ complexity of [7], the proposed algorithm shows $O(n^2)$ complexity.

When resource constraints are larger than two, the partitioning problem becomes multiway partitioning. In the previous work [7], by calculating gains for pair-swapping movements, it considered all possible movements between all possible resources. Here, we adopted the scheme proposed in [11], which limits the movements between two specific resources. In Fig.3(a) , a node in resource 1 can be exchanged with a node in any resource. But in (b), a node in resource 1 can move to the resource 2 regardless of the existence of an exchangeable node. But, the movement to the resource 3 or 4 is prohibited in this pass. The rest of the solution space will be explored in the next passes. Comparing to [7], the worst case number of directions are reduced from $O(K(K-1))$ to $O(K)$. We easily observe that $K/2$ pairs are selected out of $K(K-1)/2$. Important decision in each pass is: how to pair resources? In our formulation, random pairing scheme is used. The remainder of this paper is organized as follows; Section 2 presents the problem formulations for the proposed methodology. Section 3 describes the overall algorithm to schedule and allocate nodes in a DFG. Section 4 contains experimental results. Finally, Section 5 shows final remarks on this work.

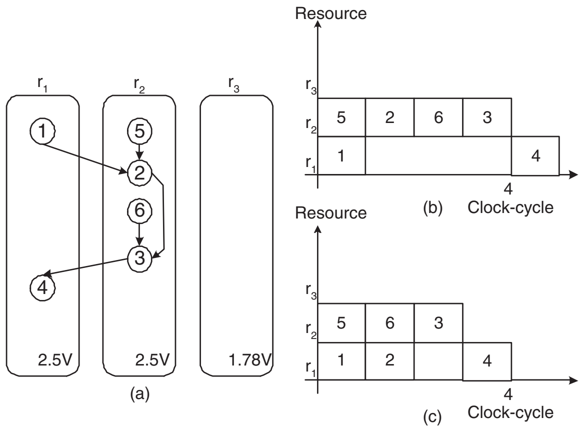

Figure 4. Clock-cycle gain example.

2 Formulations

2.1 Basics

Suppose nodes in a DFG are partitioned into K non-empty disjoint resources. We introduce $N = \{n_1, n_2, ..., n_p\}$ and $E = \{e_1, e_2, ..., e_q\}$, which represents the set of nodes and edges, respectively, in a DFG. Also, we introduce $R = \{r_1, r_2, ..., r_k\}$, which denotes possible resources and specifies resource constraints. Time constraint T_c is given to determine whether the solution can satisfy the desired time constraint or not. An edge between different resources is called cut edge, and the sum of the weights of all cut edges is called cutsize. The optimal schedule and allocation by the partitioning algorithm should satisfy the following conditions;

- Each node is allocated to exactly one resource.
- The schedule should be finished within the time constraint T_c.
- The cutsize should be minimized.
- Energy dissipation is minimized by scheduling as many nodes as possible to lower voltage resources.

2.2 Gain Calculations

To evaluate the movements for nodes, gains are calculated for required clock-cycle, energy dissipation and cutsize. In our single node movement algorithm, one node is moved at a time. Hence, the result may have different number of nodes allocated in each resource. In the extreme case, all nodes may be allocated to only one resource to minimize the cutsize. To prevent that situation, the concept of "balance" was introduced in [10], which balances the number of nodes. But, in this problem, the number of nodes allocated in each resource is not critical as long as the schedule

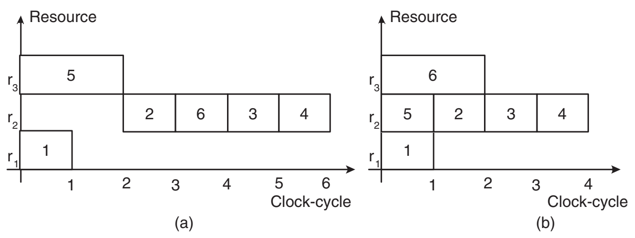

Figure 5. Energy gain example.

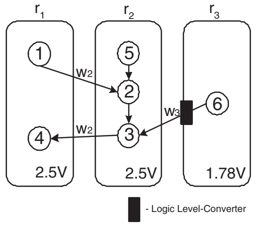

Figure 6. Cutsize gain example.

satisfies the time constraint T_c. Even though the number of nodes is balanced in each resource, it may be a poor quality schedule if the time constraint is not satisfied. Therefore, in our formulation, required clock-cycle for schedule is considered rather than the number of nodes in each resource. In Fig.4(a), an example DFG is shown with resource allocation. All nodes are assumed to be add operations. Then, a schedule that requires 5 clock-cycles is generated using list scheduler, as shown in (b). The list scheduler analyzes the data-dependency to find available nodes, then, schedules a node with the longest critical-path delay. Let us suppose that resource r_1 and r_2 are paired for partitioning. As an example, when node 2 is moved from r_2 to r_1, we can save one clock-cycle as shown in (c). Hence, the clock-cycle gain for node 2 is 1 as formulated in Eq.2, where C_{rq} denotes the required clock-cycle.

$$G_{clk} = C_{rq,before\_move} - C_{rq,after\_move} \qquad (2)$$

Let us now consider energy dissipation. Suppose, in Fig.5, r_2(2.5V) and r_3(1.78V) are paired for partitioning. Then, the gains of nodes in r_2 and r_3 are calculated. For all nodes in r_2, energy gains are positive when one of them is moved to r_3 because r_3 has lower energy dissipation. In Table 1, propagation delay of r_3 is defined as 2 clock-cycles. Hence, the movement for a node from r_2 to r_3 may require more clock-cycles depending on the topology of the DFG. The energy gain for a node is calculated based on the Eq.refeq3, where E denotes energy dissipation of a node.

$$G_{energy} = E_{before\_move} - E_{after\_move} \qquad (3)$$

(V)	2.5		1.78		1.54		1.38		1.26		1.17		1.09	
	E	D	E	D	E	D	E	D	E	D	E	D	E	D
*	263	2	135	3	101	4	81	5	67	6	57	7	50	8
+	6.5	1	3.3	2	2.5	3	2.0	4	1.6	4	1.4	5	1.2	5

Table 1. Energy(in pJ) and Delay(in clock-cycle) for multiplier(*) and adder(+) using TSMC 0.25μ technology. (t$_{clock}$=8ns)

Finally, we have to minimize the cutsize. For partitioned floorplan, we have to reduce the number of cut edges between two end-points with different supply voltages. Therefore, the weights of cut edges are defined according to the supply voltages of their end-points as shown in Fig.6. To reflect the effect for voltage-partitioned floorplan, the weights for cut edges are defined such as $w_1 < w_2 < w_3$, where w_1, w_2 and w_3 denote the weight of cut edge between low voltage, high voltage, and different supply-voltage resources, respectively. Higher weight for the cut edges between different supply-voltages will help to reduce the number of cut edges between different supply-voltages, thereby separating resources in terms of their supply voltages. Cutsize gain is calculated based on the Eq.4, where W_E stands for the weight of a cut edge.

$$G_{cutsize} = \sum_{\forall cuts, before\_move} W_E - \sum_{\forall cuts, after\_move} W_E \tag{4}$$

By combining the gains described above, we weight each factor by α, β and γ. Eq.5 describes total gain for a node movement.

$$G_{total} = \alpha G_{clk} + \beta G_{energy} + \gamma G_{cutsize} \tag{5}$$

3 Algorithm

For partitioning algorithm, we adapted the partitioning algorithm proposed in [11]. The proposed algorithm is a simple and effective hill-climbing method called K-PM/FM, which reduces the multiway partitioning problem into sets of bipartitioning problems. The algorithm generates pairs of resources at the beginning of an each pass. Then, movements for nodes are limited between the paired resources. For two paired resources, we apply FM algorithm to partition. Different from K-KL, K-PM/FM moves a node at a time.

As an initial partition, we randomly generate a solution. Therefore, the initial schedule and allocation may not satisfy the time constraint. Before calculating gains for movements, we pair the resources as shown in Fig.3. We observe that $K/2$ pairs are selected out of $K(K-1)/2$. In our formulation, a random resource-pairing scheme is used. Compared to [7], the worst case number of multiway directions are reduced from $O(K(K-1))$ to $O(K)$. When resources are paired for partitioning, we calculate and update the gains for each node until all nodes are moved. Also, compared to [7], the worst case complexity of the partitioning algorithm is reduced from $O(n^3)$ to $O(n^2)$.

As stated in [6], energy minimization for higher Energy/Delay ratio has to be done first. Therefore, partitioning for the resource pairs with higher Energy/Delay is performed first to assign available unused slacks to the higher Energy/Delay resources. Stopping criteria is set such that the pass stops immediately when there is no overall gain for all possible movements in a pass. However, to give more hill-climbing capability, we set a threshold such that several consecutive passes with no immediate gain stops the loop.

In Fig.7, optimizing curves are shown for a selected example. Normalized to their final value, the costs are shown for required clock-cycle, energy dissipation and cutsize. Initial energy savings are from the result with random allocation. Hence, it can be higher or lower than its final value. However, at the end of the loop, it will converge to a minimal energy that satisfies the resource and time constraints. Also, with the random initial allocation, the schedule may not satisfy the time constraint T_c. However, the schedule will converge to the desired T_c in the subsequent passes. When the time constaint T_c is met, we set α in Eq.5 to zero such that the energy savings and cutsize are optimized. Also, we don't choose a sequence of movements that violates time constraint once we achieve desired time constraint. The algorithm is described in Fig.8.

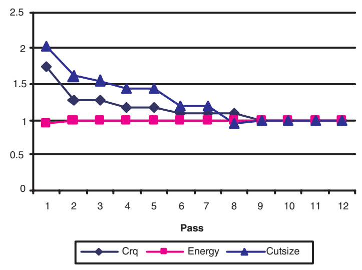

Figure 7. Normalized costs in each pass.

```
Random_Initial_Partition();
while(no_gain_pass<Threshold)
  random_resource_pairing();
  do_partition();
```

Figure 8. Partitioning algorithm.

	Non-P		K-KL		K-PM/FM	
	pJ	%	pJ	%	pJ	%
IIR	339.7	57.7	339.7	57.7	339.7	57.7
FFT1	94.7	10	79.6	24.4	76.4	27.4
FFT2	97.74	6.05	85.64	17.69	76.2	26.7
Ellip1	160.4	6.2	153.28	10.37	145.3	15.1
Ellip2	152.3	11	141.7	17.2	114.7	32.9
Lattice1	2061.3	30.6	1926.6	35.1	1672.7	43.7
Lattice2	1559.6	47.5	1550.4	47.8	1562.8	47.4

Table 2. Energy dissipations and savings including level converters.

	Non-P	K-KL	K-PM/FM
IIR	22.8	22.8	22.8
FFT1	15.1	0	0
FFT2	18.1	6.1	3
Ellip1	37.9	22.8	22.8
Ellip2	15.1	0	0
Lattice1	9.1	6.1	9.1
Lattice2	21.2	15.1	21.2

Table 3. Energy dissipations (pJ) by level converters.

4 Experimental Results

Results are presented for selected examples. Energy dissipation is calculated for each functional unit (implemented in TSMC 0.25u technology) with random input vectors. Each example is modeled for one sample period. Using list scheduler with resource constraints, the minimum required clock-cycle (T_{min}) was determined for each example. Results are shown when $T_c = 1.5T_{min}$. Supply voltages for each resource are determined by the Multiple-V$_{dd}$ scheduler introduced in [7]. Also, the results of the Multiple-V$_{dd}$ scheduler are used as an initial solution for K-KL. The results from the Multiple-V$_{dd}$ scheduler are labeled as *Non-P*. Also, the solutions generated by the proposed algorithm are labeled as *K-PM/FM* in the following presentation. For interconnect energy calculations, switching activity is assumed to be 50%. For cut edge weights, w_1, w_2, and w_3 are set to 0.5, 1, and 2, respectively.

In Table 2, the energy savings by previous algorithm and the proposed algorithm are presented. On average, relative to the single voltage design, 24%, 30%, and 36% of the energy were saved by Non-P, K-KL, and K-PM/FM, respectively. The results by K-KL are generated based on the solution by Non-P. On the other hand, the solution by K-PM/FM does not depend on the solution of Non-P. Hence, in some examples, the energy savings by K-PM/FM is better than the solutions by K-KL. In general, the energy savings are comparable.

In Table 3, the energy savings by level converters are

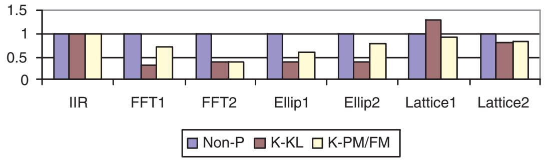

Figure 9. Interconnect energy dissipations. Results are normalized to *Non-P*.

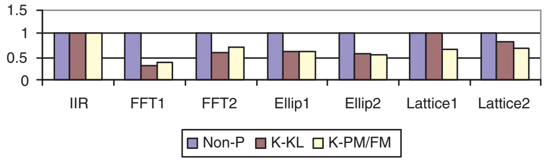

Figure 10. Interconnect lengths. Results are normalized to *Non-P*.

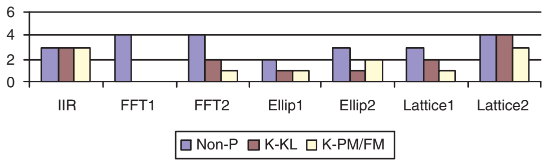

Figure 11. Required number of level converters.

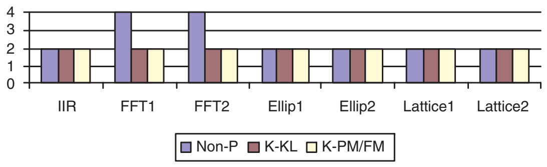

Figure 12. Number of voltage clusters.

	KL	K-PM/FM
IIR	3	2
FFT1	578	17
FFT2	324	16
Ellip1	742	108
Ellip2	3306	139
Lattice1	1446	28
Lattice2	2694	31

Table 4. CPU time (in sec).

shown. Partitioned scheme saves more energy by removing the cut edges between the different supply voltages. Relative to Non-P, 53% and 46% of the level converter energy were saved by K-KL and K-PM/FM, respectively. The savings were significant for the examples with high parallelism.

As noted in [7], minimizing cutsize reduced both interconnect energy and lengths. In Fig.9 and Fig.10, comparisons for energy dissipations and lengths of the interconnect are presented. For convenience, results are normalized to Non-P. The lengths of the interconnect were estimated based on the Manhattan distance between the modules. The floorplan algorithm introduced in [12] was used to generate the floorplan. With respect to Non-P, interconnect energy savings of 34% and 24% were achieved by K-KL and K-PM/FM, respectively. For Lattice1 example, we can observe the trade off between cutsize and module energy savings.

Fig.11 shows required number of level converters. Required numbers of level converters were comparable to the K-KL. In Fig.12, the number of voltage clusters is shown. A voltage cluster is defined as a set of contiguous modules that operates at the same voltages. For the examples with large number of resources and high-parallelism, the clustering effects were significant. In Table 4, cpu times are presented. Due to the reduced overall complexity of the proposed algorithm, 32X improvement in CPU time is achieved on average.

5 Conclusion

We proposed a scheduling and allocation scheme for voltage-partitioned design. The voltage-partitioned design reduced the communication between the resources operating at different supply voltages. The proposed algorithm performs scheduling and allocation simultaneously for voltage-partitioned designs with resource and time constraints. Compared to the previous approach, the complexity of the algorithm was reduced from $O(n^3)$ to $O(n^2)$. Also, the multiway partitioning complexity was reduced from $O(K(K-1))$ to $O(K)$. Compared to the non-partitioned single voltage schedule, 36% of the energy savings were achieved. Also, improvements for the energy dissipation and length of interconnects were observed. In the floorplan, the proposed algorithm reduced the number of voltage clusters. Overall, the proposed algorithm showed comparable results to the previous approach with reduced computational time. Due to the reduced complexity of the proposed algorithm, 32X improvement in CPU time was achieved.

References

[1] A.Chandrakasan, S.Sheng and R.Brodersen. Low-power cmos digital design. *Journal of Solid-State Circuits*, 24:473–483, April 1992.

[2] K. Usami and M. Horowitz. Clustered voltage scaling technique for low power. In *ISLPED '95*, volume 99, pages 3–8, Dana pt. CA, January 1995.

[3] J. Chang and M. Pedram. Energy minimization using multiple supply voltages. *IEEE Transactions on VLSI systems*, 5(4):436–443, Dec 1997.

[4] M. Johnson and K. Roy. Datapath scheduling with multiple supply voltages and level converters. *ACM Transactions on Design Automation of Electronic Systems*, 2(3):227–248, July 1997.

[5] S. Raje and M. Sarrafzadeh. Scheduling with multiple voltages under resource constraints. In *IEEE International Symposium on Circuits and Systems*, pages 350–353, July 1999.

[6] A. Manzak and C. Chakrabarti. A low power scheduling scheme with resources operating at multiple voltages. *IEEE Transactions on VLSI Systems*, 10:6–14, Feb 2002.

[7] Dongku Kang, Mark Johnson, and Kaushik Roy. Multiple-v_{dd} scheduling/allocation for partitioned floorplan. In *International Conference on Computer Design*, pages 412–418, San Jose, CA, 2003.

[8] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.*, 49:291–307, Feb 1970.

[9] L. Sanchis. Multiple-way network partitioning. *IEEE Transactions on Computers*, 38:62–81, Jan 1989.

[10] C. Fiduccia and R. Mattheyses. A linear time heuristic for improving network partitions. In *Design Automation Conference*, pages 175–181, 1982.

[11] J. Cong and S. K. Lim. Multiway partitioning with pairwise movement. In *International Conference on Computer Aided Design*, pages 175–181, 1998.

[12] H. Murata et al. Vlsi module placement based on rectangle-packing by the sequence-pair. *IEEE Transactions on CAD Design of Integrated Circuits and Systems*, 15:1518–1524, Dec 1996.