Power-Driven Design Partitioning

Rajarshi Mukherjee, Seda Ogrenci Memik

Electrical and Computer Engineering, Northwestern University
{rajarshi, seda}@ece.northwestern.edu

Abstract. In order to enable efficient integration of FPGAs into cost effective and reliable high-performance systems as well potentially into low power mobile systems, their power efficiency needs to be improved. In this paper, we propose a power management scheme for FPGAs centered on a power-driven partitioning technique. Our power-driven partitioner creates clusters within a design such that within individual clusters, power consumption can be improved via voltage scaling. We tested the effectiveness of our approach on a set of LUT-level benchmark netlists. Further we did constrained placement of the clusters into predefined V_{dd}^{high} and V_{dd}^{low} regions for a single FPGA. Average savings in power consumption with our approach is 48% whereas penalty in channel width and wire length due to constrained placement is 23% and 26% respectively.

1 Introduction

Despite exponential improvements in logic density and performance, energy efficiency of the FPGA technology did not keep up. As heat dissipation becomes an increasingly important concern for wired systems, power consumption of FPGAs needs to match more stringent standards in order to ensure the performance goals and reliability. Similarly, inefficiency in power consumption poses a major obstacle to inclusion of FPGAs in many emerging low power mobile systems.

In this paper, we present a high-level power management methodology for FPGAs. We propose a power-driven partitioning technique that identifies partitions in a design, which can be implemented using lower supply voltage levels. Through voltage scaling we allow longer delay for parts of a design while the overall latency of the design is unchanged. In every design, based on the overall timing constraint, there is a set of *critical* nodes/operations/gates, while the remainder of the design is *non-critical*. Hence, it is possible to identify the inherent time slack possessed by individual building blocks in a design. We analyze this design metric and systematically exploit potential relaxation in timing constraints in order to create opportunities for voltage scaling. Our techniques can be utilized for various FPGA-based systems. In multi-FPGA systems applications are partitioned among several devices. Using our partitioning technique voltage scaling can be applied at chip level, using different supply voltages for different devices. While mapping a design onto a single FPGA chip, the design is partitioned such that portions identified by our partitioning technique can be placed within voltage islands at different V_{dd} levels on the same chip. Finally, for dynamically reconfigurable systems, the voltage supply level can be dynamically adjusted and our partitioning technique can be used to create configuration contexts such that lowering of the supply voltage is feasible for some of the partitions.

Our specific contributions in this paper are as follows:

- We propose a partitioning algorithm that effectively exploits *relaxable* timing constraints within a design to trade-off delay against lower supply voltage levels,
- We present experimental results using LUT-level netlists to demonstrate the improvement in power using our proposed technique, and
- We present an experimental evaluation of the impact of creating voltage islands on the physical design stages.

The rest of the paper is organized as follows. In Section 2 we present an overview of related work. Section 3 presents the problem formulation and our algorithm. Our experimental setup and results will be presented in Section 4. We conclude with a summary in Section 5.

2 Related Work

In the past, various proposed synthesis techniques addressed the problem of power optimization by improving metrics such as switching activity of a schedule or binding, total used logic resources, interconnect, etc. to generate the most efficient circuit in terms of power consumption [1], [2], [3], [4], [5]. In our approach we are addressing a different power optimization paradigm, namely voltage scaling. Our work is complementary to the above-mentioned techniques.

Partitioning has been studied for multi-FPGA systems and for dynamically reconfigurable systems [6], [7], [8], [9]. The main objectives for optimization have been traditionally cut cost, i.e., the number of connections between partitions, and the number of partitions. In this work, we propose a partitioning scheme that addresses a new objective, namely availability of time slack in a partition. We investigate the potential impact of partitioning on power. Depending on the particular application of our technique, we take other partitioning objectives into account as well.

Voltage scaling is a well-known tool for improving energy efficiency of electronic systems. It has found applications for a wide variety of circuit technologies and design styles including microprocessors, ASICs and real time embedded systems [10], [11], [12], [13]. In this work, we make use of this general optimization technique and investigate its application on FPGA-based systems. Investigation of circuit level issues to implement voltage scaling is beyond the scope of this paper. However, the feasibility of implementing the necessary hardware for voltage scaling is evident considering the successful implementations in other technologies. In addition, Chen et al. reported recent results on the feasibility of dual supply voltage FPGA fabrics [14], [15].

3 Power Management Using Voltage Scaling

In this section, we will formulate our power management problem and discuss two applications of our proposed technique. Next, we will describe our power-driven partitioning algorithm in detail.

3.1 Problem Formulation

We assume that a LUT-level netlist is represented with a *Directed Acyclic Graph (DAG)*. The longest path from any of the primary inputs to any of the primary outputs defines the longest combinational path, i.e., the *critical path* in the design. Logic blocks that reside on the critical path are called *critical nodes*. The rest is referred to as *non-critical* nodes. Taking the length of the critical path as our timing constraint and assuming that all input signals arrive at the same time, we can assign *arrival* and *required* times for each node. The difference between the required time and the arrival time of a node is called *time slack*. This entity will be equal to zero for critical nodes, while it takes a positive value for non-critical nodes. Note that we are estimating the time slack of each LUT at a high level. Naturally, the timing behavior of a design will highly depend on the net delays, hence, on placement and routing. An accurate estimation of interconnect delay cannot be made before physical synthesis. Being aware of this fact, we will evaluate the impact of placement and routing. Details of this study will be presented in Section 4.3. We would like to stress once again that time slack as defined above is the best estimation we can have at this early stage of the design flow to identify the non-critical portions of a design.

Our power management technique relies on the observation that the slack possessed by individual nodes can be used as a guide to create partitions within which all nodes would maintain a certain level of timing freedom, which in turn can be exploited through scaling the voltage supply fed into that partition. It is well known that the dynamic power reduces by the square of the supply voltage ($P \sim C_{load} V_{dd}^2 f_{switch}$), and the delay increases linearly ($D \sim C_{load} / V_{dd}$) as the supply voltage is decreased. Hence, it is possible to perform a tradeoff between power consumption and performance by changing the supply voltage.

Our aim is to identify partitions in a design, such that the total power consumption is minimized while resource constraints associated with the partitioning problem are satisfied. For a given partition, the length of the longest path and the amount of time slack available along that path will be used to compute the voltage scaling within a partition. The voltage-scaling factor (Sc_Fac) is then defined as

$$Sc\_Fac = \frac{longest\_path\_length\_in\_partition}{longest\_path\_length\_in\_partition + slack}$$

Our partitioning scheme can find applications at various levels. Next we differentiate between two cases and are shown in Figure 1.

3.1.1 Chip-level Voltage Scaling for Multi-FPGA Systems

Many FPGA-based hardware acceleration systems employ multiple FPGAs. Partitioning is frequently used to map a large design onto several FPGA devices. Such systems are generally wired; hence, they do not operate under a tight power budget. However, heat dissipation is becoming a growing concern. Overhead due to cooling systems can be reduced through effective power management. More importantly, reliability issues arising due to excessive heat dissipation require closer attention as the FPGA manufacturing technology reached submicron levels. For multi-FPGA systems we apply our partitioning technique to create partitions such that some

of those partitions can be assigned to a device operating at a lower supply voltage level.

3.1.2 Localized Voltage Scaling for Single FPGA Systems

A design flow targeting a single FPGA device can also benefit from our power-driven partitioning technique. In this case, by embedding voltage islands on a chip we can enable different parts of a design to operate at different voltage levels. Generating these voltage islands on a single FPGA will incur certain hardware costs. Considering the overhead of creating voltage islands –additional circuitry for voltage scaling, level conversion, etc. the number and layout of different voltage islands can be constrained. We believe that the benefits of voltage scaling will justify the additional hardware cost in the next generation FPGA architectures. We will elaborate more on this issue as we present our results in Section 4.

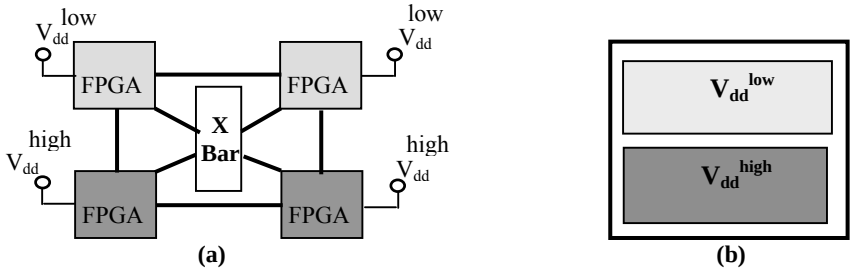

Figure 1. Application of voltage scaling in two different scenarios. (a) Multi-FPGA system. (b) Single FPGA containing voltage islands.

3.2 Power-Driven Partitioning Algorithm

Our algorithm takes in a LUT – level netlist. The input buffer connected to the input port (*IBUF*) is considered a primary input and output buffer connected to the output port (*OBUF*) is considered a primary output. We assume that each LUT has a delay of 1 unit when operating at full supply voltage level.

Our algorithm tries to identify clusters of nodes along a path, which can share the time slack available along that path. The reasons why we consider only nodes along a path are twofold. First, since nodes along a path have data dependencies, they are likely to be placed in the same partition (or be placed close physically in the case of single FPGA systems) even without voltage scaling considerations. Second, the slack values on the nodes along a path are usually close to each other. Clearly, a good power-driven partitioning algorithm clusters nodes with similar slack values together.

Our algorithm first finds the slack values on the nodes, forms an initial set of clusters assuming the availability of arbitrary scaling factors. Then, in a refinement phase the voltage scaling factors of the clusters are adjusted according to the available supply voltage levels in the target hardware.

The slack at each node is calculated by computing the difference of required and arrival times of the signals by sorting the nodes in topological order. The algorithm takes as input the feasible scale factor Sc_Fac – the minimum amount by which volt-

age can be scaled. Then, it selects the nodes in topological order thereby selecting nodes from the input level and going towards the output. If the node has zero slack it is added to the non-scaled partition. After choosing a non-zero slack node *v,* which is not already added to any partition, it is checked whether

$$Sc\_Fac > \frac{longest\_path\_in\_partition}{longest\_path\_in\_partition + slack}$$

If this condition is satisfied, then a new cluster is created and the algorithm tries to grow the cluster. (As we add the first node to a partition, the delay of the longest path in the partition is equal to the delay of the node itself.) Iteratively, minimum slack fanout nodes are selected (not already in another cluster) and added to the cluster if it is feasible to add a new node and the slack of the path is updated as the minimum of the slacks of the nodes in the path. Let us call this entity $slack^{Path}$. Similarly, the length of the longest path in the partition is updated.

Once we stop adding any more nodes to a cluster c_i, we will have the following information about this cluster: $\boldsymbol{M_i}$: number of nodes along the longest path in c_i, $\boldsymbol{L_i}$: length of the longest path in c_i, $\boldsymbol{slack^{Path}}$: available slack along the longest path in c_i. (Without violating any overall timing constraints of the circuit, we can slow down the longest path in this cluster by $slack^{Path}$ time units.), ***node_delay***: amount of time by which each individual node in c_i can be slowed down. We can formally express *node_delay* as follows:

$$node\_delay = \frac{slack^{Path}}{M_i}$$

Since the same voltage-scaling factor will be applied to all nodes within a partition, the slowdown of each node will be same. The voltage supply for this cluster can be scaled down by the factor of

$$scale^{cluster_i} = \frac{L_i}{L_i + slack^{Path}}$$

Iteratively, clusters are created until all nodes are assigned to a cluster.

3.2.1 Post processing of Clusters

After creating an initial set of clusters our next goal is to assign these clusters into voltage scaled FPGAs or individual voltage islands on a single chip. The circuitry employed for scaling will have a pre-defined sensitivity. In other words, the incremental steps by which we can adjust the voltage level are quantized, e.g. voltage scaling can be within a range of 0.64 Volts in 8 steps of 0.08 Volts. First, we perform a pass over all clusters and round up their voltage levels to the nearest feasible level.

If the number of clusters in the initial partition is less than or equal to the number of FPGA devices in the system then the assignment is straightforward. For the single FPGA case, we will identify two distinct voltage levels: V_{dd}^{high} and V_{dd}^{low}. Hence, each cluster will be merged to either one of these two levels. For the multi-FPGA case, we can allow more voltage levels since one voltage regulator will serve one individual chip. We developed two heuristics to achieve the finalized partitioning of the netlist into *N* voltage scaled partitions (*N*=2 for single FPGA scenario). We refer to these schemes as Delay Based Merging and Connectivity Based Merging.

Delay Based Merging tries to satisfy the resource constraint while searching for the best possible merging in terms of power gain. Essentially, this method tries to group nodes with the most similar slack values together in the same merged cluster. Connectivity Based Merging puts higher emphasis on reducing the cut cost of the final partitioning result. It achieves this by merging clusters with highest number of mutual connections together. We omitted the details of these post-processing heuristics and present only the results of Connectivity Based Merging due to space considerations. After completing the partitioning phase, the total power consumption now becomes P_{scale}. If the total power consumption without any voltage scaling was P, the ratio is given by

$$\frac{P_{scale}}{P} = \frac{(V_{dd}^{scale})^2 \times k + (n-k)}{n}$$

where, *n* is the total number of nodes in the circuit and *k* is the number of nodes in the voltage scaled partition. For an *r-way* partition into *r* FPGA devices, we take the same approach. Assume that the partition sizes are k_1, k_2, k_3,...k_r. The ratio of the scaled power consumption to the non-scaled power consumption is given by

$$\frac{P_{scale}}{P} = \frac{(V_{dd}^{scale_1})^2 \times k_1 + (V_{dd}^{scale_2})^2 \times k_2 + \ldots + (V_{dd}^{scale_r})^2 \times k_r}{n}$$

4 Experiments

We present our experimental results in this section. First, we summarize our experimental setup and the parameters we used. Then, we report achieved reduction in total power consumption on a collection of benchmarks.

4.1 Experimental Setup and Parameters

The experiments are formed on a set of MCNC combinational benchmarks. Further we also tested with synthetic benchmarks of 1000, 5000, 10000 and 15000 nodes. Table 1 summarizes relevant characteristics: number of nodes, edges and *critical nodes* - the number of nodes with zero slack in the DAG representation of each MCNC benchmark. In Table 2 we show the average values of the same for synthetic benchmarks. It is to be noted that though the results are presented for combinational circuits, the algorithm is equally applicable to sequential circuits where its combinational block can be partitioned into voltage clusters.

The LUT level netlist for MCNC benchmarks were in .net format. The power-driven partitioner reads in an .edif or .net file and produces a partition for *N* FPGAs in a multi-FPGA system or *N=2* voltage islands on a single FPGA. The initial delay of each LUT is assumed to be 1unit. We further assumed that voltage scaling is done in increments of 0.06 Volts and scale factor is 0.62. Finally, we assume that for the multi-FPGA scenario, all FPGA devices in the system are identical, i.e., all have the same capacity. Our techniques are independent of the actual values of these parameters; hence, they can operate under different assumptions.

4.2 Results: Power Driven Partitioning

After creating clusters we perform an initial refining, where we round off voltage scaling values according to the smallest scaling step. Next we perform our post processing heuristics of Delay Based Merging and Connectivity Based Merging. In both the merging we generate N-voltage clusters each of roughly the same size. We observe that while we obtain better power improvement using Delay Based Merging the cut cost is consistently inferior to the Connectivity Based Merging. Symmetrically, Connectivity Based Merging always yields better-cut cost at the expense of reduced power improvement. Hence, the possible trade-off between the cut cost objective and power improvement is evident. Table 1 and Table 2 shows the results of only connectivity based merging. We report the cut cost, i.e., the number of inter partition connections as well as the percentage power improvement obtained. For MCNC benchmarks, the average power improves 13.4% for 2-way to 14.7% for 8 way whereas the average cut cost increases from 2389.5 for 2-way to 2530.9 for 8-way. For Synthetic benchmarks the average power improvement is 27.45% for 2-way and 48.2% for 8-way.

Table 1. Partitioning Results for MCNC benchmarks

				2-way		4 way		8-way	
MCNC	Nodes	Edges	Crit Nodes	Cost	%Imp	Cost	%Imp	Cost	%Imp
Alu4	1522	5401	1180	1467	13	1483	13	1556	17
apex2	1878	6690	1453	2158	13	2190	14	2335	14
apex4	1262	4461	1169	440	6	440	6	440	6
des	1591	5863	1326	1851	18	1888	18	1890	19
ex1010	4598	16069	3557	5660	13	5748	13	5868	13
ex5p	1064	3940	941	663	9	663	9	663	9
misex3	1397	4955	992	1405	19	1501	22	1576	22
pdc	4575	17154	3179	4901	13	5141	14	5315	14
seq	1750	6159	936	1685	21	1804	21	1832	21
spla	3960	13763	2919	3665	9	3803	9	3834	12
Maximum				5660	21	5748	22	5868	22
Average				2389.5	13.4	2466.1	13.9	2530.9	14.7

Table 2. Partitioning Results for synthetic benchmarks (average)

Synthetic			2-way		4 way		8-way	
Nodes	Edges	Crit Nodes	Cost	%Imp	Cost	%Imp	Cost	%Imp
1000	1751	129	360	27.2	794	39.4	941	47.6
5000	8323	209	588	29.6	1392	41.8	1662	50.2
10000	16270	276	775	27.8	1962	41.6	2357	48.4
15000	24220	359	1023	25.2	2578	37.6	3087	46.6
Maximum			1023	29.6	2578.2	41.8	3087	50.2
Average			686.45	27.45	1681.45	40.1	2011.8	48.2

4.3 Results: Using Power Driven Partitioning in Physical Design

We now present the results of the constrained placement of the clusters identified by our algorithm onto for voltage islands supplied by different V_{dd}s on a single FPGA. We propose to use four quadrants as four voltage islands, each of which can be potentially supplied by either V_{dd}^{high} (say 1.3V) or V_{dd}^{low} (say 0.8V) rather than having each individual logic block being V_{dd} programmable. The location voltage islands is an architectural parameter. Figure 2 shows the 3 possible configurations. We assume that it is possible to have level converters along the borders of the quadrants. FPGA having dimensions D_X, D_Y has total ($D_X \times D_Y$) logic blocks. Our placement uses f – the fraction of critical nodes to total logic blocks. Configuration 2(a) is used if is $f \leq ¼$, 2(b) is used if $¼ \leq f \leq ½$ and 2(c) is used if $½ \leq f \leq ¾$. Obviously for $¾ \leq f \leq 1$, all the quadrants must be supplied with V_{dd}^{high} and there is no power improvement possible by V_{dd} scaling.

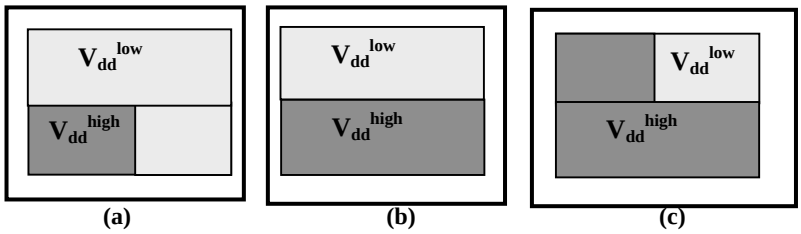

Figure 2. Proposed Voltage Island configurations in a FPGA

Our placement works as follows: The cluster of *critical nodes* identified by power driven partitioning algorithm presented in Section 3.2 must be placed in a V_{dd}^{high} region – (determined by f) otherwise it would mean slowing down the critical path. The power improvement is higher if all *non-critical nodes* get placed in the V_{dd}^{low} region although they have a freedom to move to V_{dd}^{high} quadrant if it improves the congestion cost.

The placement is based on the simulated annealing implemented in VPR [16]. The linear congestion cost function used in VPR is given by $C_{linear\_conj} = \sum_{i=1}^{N_{nets}} q(i)\left[\frac{bb_x(i)}{C_{av,x}(i)^{\beta}} + \frac{bb_y(i)}{C_{av,y}(i)^{\beta}}\right]$ where for each net $bb_i(x)$ and $bb_i(y)$ denote the horizontal and vertical spans of its bounding box, q(i) is the compensating factor and $C_{av,x}(i)$ and $C_{av,y}(i)$ are average channel capacitances in the x and y directions respectively. Our cost function C is similar to that presented in [16] and is given as:

$$\Delta C = \Delta C_{linear\_conj} + \alpha \Delta matched(j) + \gamma(1 - matched(j))$$

where matched(j) returns 1 if the j^{th} logic block is placed in its matching voltage quadrant and 0 if not. Δmatched is the difference between matched(j) in the previous placement and matched(j) in the present placement and penalizes a move that brings a block from matched quadrant to unmatched quadrant; (1-matched(j)) penalizes a move that moves a block in unmatched quadrant to another location in the unmatched quadrant; α, γ are appropriate constants and $\gamma > \alpha$.

For FPGAs it is common practice to pack the LUTs into logic blocks. We used T-Vpack [16] to pack LUTs with clusters of size 4 and 10 inputs per cluster option. After packing, the power driven partitioner partitions the logic blocks for V_{dd}^{high} and V_{dd}^{low} regions. For apex2 benchmark the characteristics are: the smallest FPGA size is 23×23 logic array, out of 485 clusters there are 66 critical logic blocks and f is 12%. The placement is of type shown in Figure 2(a) and the placed circuit is shown in Figure 3.

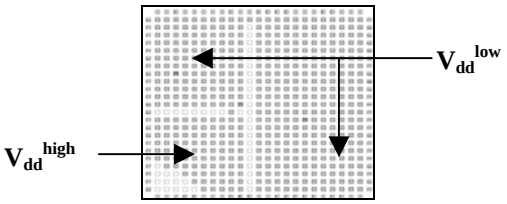

Figure 3. Constrained Placement of LUTs clusters of apex2

Ten MCNC benchmarks were clustered and Table 3 shows their total number of logic blocks, primary inputs (PI) and outputs (PO), % f and the FPGA size $D_X \times D_Y$. Wire length (WL) and channel width (CW) are shown for unconstrained placement vs. power driven partitioning and region-constrained placement. We also report the number of non-critical blocks in the V_{dd}^{low} region and the power improvement over unconstrained partitioning. The results show that it is possible to get an average power improvement of 48% with 23% penalty in channel width and 26% penalty in wire length.

Table 3. Placement and Routing results for MCNC benchmarks

							Unconstrained		Constrained –Power Driven			
Circuits	Logic Blocks	Crit Block	Dx=Dy	%f	PI	PO	WL	CW	WL	CW	Blocks in V_{low}	%Imp
alu4	389	51	20	13	14	8	20262	33	22151	37	289	46
apex2	485	66	23	12	38	3	31738	40	37348	47	385	49
apex4	334	77	19	21	9	19	20317	40	23383	48	247	46
des	415	47	32	5	256	245	25199	25	44587	41	357	53
ex1010	1201	250	35	20	10	10	72916	43	104744	69	908	47
ex5p	280	22	17	8	8	63	17572	41	19625	47	213	47
misex3	362	57	20	14	14	14	21650	36	24606	39	281	48
pdc	1190	141	35	12	16	40	98312	61	116923	71	911	48
seq	448	94	22	19	41	35	27904	39	35767	48	342	47
spla	955	133	31	14	16	46	71330	57	86536	66	717	47
Average							40720	42	51567	51	465	48
Avg. Penalty for Const-Power Driven					Wire length		26.64%	Channel width			23.61%	

5 Conclusions

In this paper, we presented a power optimization technique for FPGA-based systems. Our proposed approach aims to create opportunities for voltage scaling by grouping nodes in a LUT-level netlist into clusters. The partitioning approach targets to exploit the maximum flexibility in timing constraints and convert it into voltage scaling. We developed a partitioning algorithm to perform this task. Within the general framework of our algorithm resource constraints can be resolved. Our experimental results reveal that it is indeed effective. Based on the high-level power improvement estimation, we did a constrained placement of the clusters onto voltage islands in a single FPGA and showed power improvement vs. penalty in wire length and channel width increase. Hence possible tradeoffs between cost function and power improvement is evident. An immediate extension to our work is to evaluate and actually compare delay after physical design due to constrained placement and area-power tradeoffs due to the presence of level converters.

6 References

1. Boemo, E.I., et al., eds. Some Notes on Power Management on FPGA-based Systems. Lecture Notes in Computer Science. Vol. 975. 1995, Springer-Verlag: Berlin. 149--157.
2. Chen, C., T. Hwang, and C.L. Liu. Low Power FPGA design - A Re-engineering Approach. in Design Automation Conference. 1997.
3. Sutter, G., et al. FSM Decomposition for Low Power in FPGA. in Design Automation Conference. 1998.
4. Chen, D., J. Cong, and Y. Fan. Low Power High-Level Synthesis for FPGA Architectures. in International Symposium on Low Power Electronic Design. 2003.
5. Lamoureux, J. and S.J.E. Wilton. On the Interaction Between Power-Aware FPGA CAD Algorithms. in International Conference on Computer Aided Design. 2003.
6. Brasen, D.R. and G. Saucier, Using Cone Structures for Circuit Partitioning into FPGA Packages. IEEE Transactions on CAD of Integrated Circuits and Systems, 1998. 17(7): p. 592--600.
7. Chan, P.K., M.D.F. Schlag, and J.Y. Zien. Spectral-Based Multi-Way FPGA Partitioning. in International Symposium on Field Programmable Gate Arrays. 1995.
8. Chang, D. and M. Marek-Sadowska, Partitioning Sequential Circuits on Dynamically Reconfigurable FPGAs. IEEE Transactions on Computers, 1999. 48(6): p. 565--578.
9. Liu, H. and D.F. Wong. Circuit Partitioning for Dynamically Reconfigurable FPGAs. in International Symposium on Field Programmable Gate Arrays. 1999.
10. Govil, K., E. Chan, and H. Wasserman. Comparing Algorithms for Dynamic Speed Setting of a Low Power CPU. in MOBICON. 1995.
11. Iyer, A. and D. Marculescu. Power Efficiency of Voltage Scaling in Multiple Clock, Multiple Voltage Cores. in International Conference on Computer Aided Design. 2002.
12. Yeh, C., et al. Gate-Level design Exploiting Dual Supply Voltages for Power-Driven Applications. in Design Automation Conference. 1999.
13. Simunic, T., et al. Dynamic Voltage Scaling and Power Management for Portable Systems. in Design Automation Conference. 2001.
14. Chen, D., et al. Low-Power Technology Mapping for FPGA Architectures with Dual Supply Voltage. in International Symposium on Field-Programmable Gate Arrays. 2004.
15. Li, F., et al. Low-Power FPGA Using Pre-Defined Dual-Vdd/Dual-Vt Fabrics. in International Symposium on Field-Programmable Gate Arrays. 2004.
16. Betz, V., J. Rose, and A. Marquardt, Architecture and CAD for Deep-Submicron FPGAs. Kluwer Academic Publishers, Feb 1999.