

# Delay Optimal Low-Power Circuit Clustering for FPGAs with Dual Supply Voltages

Deming Chen and Jason Cong  
Computer Science Department  
University of California, Los Angeles  
{demingc, cong}@cs.ucla.edu

## ABSTRACT

This paper presents a delay optimal FPGA clustering algorithm targeting low power. We assume that the configurable logic blocks of the FPGA can be programmed using either a high supply voltage (high-Vdd) or a low supply voltage (low-Vdd). We carry out the clustering procedure with the guarantee that the delay of the circuit under the general delay model is optimal, and in the meantime, logic blocks on the non-critical paths can be driven by low-Vdd to save power. We explore a set of dual-Vdd combinations to find the best ratio between low-Vdd and high-Vdd to achieve the largest power reduction. Experimental results show that our clustering algorithm can achieve power savings by 20.3% on average compared to the clustering result for an FPGA with a single high-Vdd. To our knowledge, this is the first work on dual-Vdd clustering for FPGA architectures.

## Categories and Subject Descriptors

B.6.3 [Logic Design]: Design Aids – Optimization

## General Terms

Algorithms, Design, Performance

## Keywords

Circuit clustering, low-power FPGA, dual supply voltage

## 1. INTRODUCTION

Reducing power consumption for FPGAs has attracted much attention recently [1, 4, 5, 8, 9, 11, 12, 17]. Meanwhile, performance remains the most important factor for FPGA designs. Since most FPGAs are hierarchical in nature, circuit clustering has become an integral part of the FPGA synthesis flow. It has been shown that cluster-based logic blocks can improve the FPGA performance, area and power [11, 13, 17].

An early work on performance-driven circuit clustering was presented by Lawler et al. in [10]. Given a constraint  $M$  on the size of the clusters, Lawler's algorithm produces a delay optimal partitioning of the circuit under the assumption that internal delays within a cluster are zero, and the external delay from one cluster to the other is one (*unit delay model*). Later, Murgai et al. proposed the *general delay model* [14]. In this model, each gate of the network has a delay; no delay is encountered on an interconnection linking two gates internal to a cluster; and an

edge delay is encountered on every interconnection between two different clusters. This model is very powerful and can capture many timing constraints by simple extensions. Rajaraman and Wong derived the first delay optimal clustering algorithm under the general delay model [15]. In [19], Vaishnav and Pedram presented a low-power single-Vdd clustering algorithm with the optimal delay under the general delay model. Their algorithm is power optimal for trees. They enumerated all clustering solutions for a graph and selected a low-power clustering solution from all delay optimal clustering solutions. Both [15] and [19] allowed node duplications, i.e., a gate may be assigned to more than one cluster. There are a few prior research efforts on clustering for FPGA architectures [2, 3, 13, 17]. The optimization goals were on area-delay tradeoff [2], routing track reduction [3], performance improvement [13], and area and power reduction [17]. There is no guarantee that one can achieve the optimal clustering delay under the general delay model in these works. In [6], a performance-driven multi-level (two-level hierarchy) FPGA clustering algorithm was presented.

One of the popular design techniques for power reduction is to lower supply voltage, which results in a quadratic reduction of power dissipation. However, the major drawback is the negative impact on chip performance. A multiple supply voltage design in which a reduction in supply voltage is applied only to non-critical paths can save power without sacrificing performance. Clustered voltage scaling (CVS) was first introduced in [18], where clusters of high-Vdd cells and low-Vdd cells were formed, and the overall performance was maintained. The works in [7, 20] combined CVS with other techniques such as gate sizing and variable supply voltage. The work in [16] assigned variable voltages to functional units at the behavioral synthesis stage. The work in [12] assigned voltage values to logic blocks in an FPGA chip made of pre-defined dual-Vdd/dual-Vt fabric.

In our work we develop a low-power FPGA clustering algorithm, named *DVpack*, with consideration of two supply voltages. We guarantee an optimal circuit delay under the general delay model. We impose the constraint that the nodes being packed in a single cluster have to be driven by the same Vdd. We extend the idea of [19] to build delay-power-vdd points to form a solution curve for each node in the network. After the optimal circuit delay is determined, the non-critical paths will be relaxed in order to accommodate low-Vdd clusters to reduce power. Our algorithm is delay and power optimal for trees, and delay optimal for directed acyclic graphs (DAGs). We also show that the complexity of solution curve generation is polynomial in terms of the network depth without the need to reduce data precision.

In Section 2, we provide some definitions and formulate the dual-Vdd FPGA clustering problem. Section 3 introduces our FPGA architecture and power model. Section 4 gives a detailed

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISLPED '04, August 9–11, 2004, Newport Beach, California, USA.  
Copyright 2004 ACM 1-58113-929-2/04/0008...\$5.00.

description of our algorithm. Section 5 presents experimental results, and Section 6 concludes this paper.

## 2. DEFINITIONS AND PROBLEM FORMULATION

A Boolean network can be represented by a DAG where each node represents a logic gate, and a directed edge  $(i, j)$  exists if the output of gate  $i$  is an input of gate  $j$ . We use  $input(v)$  to denote the set of nodes which are direct *fanins* of gate  $v$ . We use  $F_v$  to represent the subgraph of the network that contains  $v$  and all of the transitive fanins of  $v$ , including primary inputs (PIs). A *cluster* rooted on a node set  $R$ , denoted as  $C_R$ , is a subgraph such that any path connecting two arbitrary nodes in  $C_R$  lies entirely in  $C_R$ . The roots in  $R$  are also the outputs of  $C_R$ .  $node(C_R)$  represents the set of nodes contained in  $C_R$ .  $input(C_R)$  denotes the set of distinct nodes outside of  $C_R$  that supply inputs to the nodes in  $node(C_R)$ . A cluster is *K-M-feasible* if  $|input(C_R)| \leq K$  and  $|node(C_R)| \leq M$ . In this case,  $1 \leq |R| \leq M$ .  $level(v)$  is the depth (number of nodes) of the longest path from any PI node to  $v$ . A network is *i-bounded* if  $|input(v)| \leq i$  for each node  $v$ . Because the exact layout information is not available during the clustering stage, we model each interconnection edge between two clusters as having a constant delay  $D_{edge}$ , and each node in the network having a delay  $d_H$  or  $d_L$  depending on its Vdd level. Therefore, we approximate the clustered circuit delay with the general delay model. The largest optimal delay of the clustered circuit is called the *optimal clustering delay* of the circuit.

The dual-Vdd clustering problem for min-power FPGA (**DV-CMF** problem) is to cover a given *i-bounded* Boolean network with *K-M-feasible* clusters or equivalently, *K-M-clusters*. This is done in such a way that the total power consumption is minimized under a dual-supply voltage FPGA architecture model, while the optimal clustering delay is maintained. We assume that the input networks are all 4-bounded, i.e., each node  $n$  represents an LUT, where  $|input(n)| \leq 4$ . We set  $K$  to 10 and  $M$  to 4 in this study (these parameters can change). Therefore, our final clustering solution is a DAG in which each node will be a *10-4-cluster*, and the edge  $(C_U, C_V)$  exists if some node  $u \in U$  is in  $input(C_V)$ . The voltages are denoted as  $V_L$  for low-Vdd and  $V_H$  for high-Vdd.

## 3. ARCHITECTURE AND POWER MODEL

### 3.1 Level Converter and Logic Element

A level converter is required when a  $V_L$  device output is to be connected to a  $V_H$  device input. Otherwise, excessive leakage power will occur in the  $V_H$  device due to large short-circuit current. We use the same level converter presented in [5], where delay and power data of the level converter and the 4-input LUT (4-LUT) with various Vdd settings were obtained through SPICE simulation under the 0.1u technology. This work uses these data.

Figure 1 shows a  $K$ -input configurable logic block (CLB) containing  $M$  basic logic elements (BLEs, each one a 4-LUT). The output of a BLE can be programmed to go through a level converter or bypass it. This gives us the capability to insert a level converter when a  $V_L$  BLE drives a  $V_H$  BLE in another cluster. We assume that there are pre-fabricated tracks in the routing channels with either  $V_H$  or  $V_L$  settings. When a  $V_L$  BLE is driving the routing interconnects (wires and buffers), we assume that it can use a set of  $V_L$  routing tracks. This model is similar to that used in

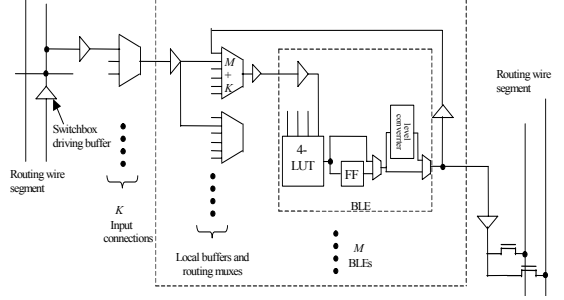


Figure 1: A CLB with  $K$  inputs,  $M$  BLEs and periphery

[5]. Our assumption represents the ideal case, which will provide an upper-bound of power reduction for clustering FPGAs with dual Vdds under timing constraint.

### 3.2 Power Model

For each *K-M-feasible* cluster, the total power of the cluster is:

$$P = \sum_{j=1}^x S_j \cdot P_{LUT} + \sum_{j=1}^x (1 - S_j) \cdot P_{LUT\_static} + P_{inputs} + P_{wire}$$

where  $S_j$  is the switching activity of node  $j$  in the cluster;  $x$  is the number of nodes in the cluster;  $P_{LUT}$  contains both dynamic and static power [5];  $P_{LUT\_static}$  is the static power of an LUT, which is counted when the LUT is not switching;  $P_{inputs}$  is the power consumed on the cluster inputs, which is defined as follows:

$$P_{inputs} = 0.5 f \cdot V_{dd}^2 \sum_{i=1}^k S_i \cdot C_{in}$$

where  $S_i$  is the switching activity on input  $i$  of the cluster.  $C_{in}$  is the input capacitance on an LUT (including MUXes and local buffers in front of the LUT);  $P_{wire}$  is calculated as follows:

$$P_{wire} = 0.5 f \cdot V_{dd}^2 \cdot (C_{local\_wire} \cdot \sum_{j=1}^x S_j + C_{net} \cdot S_o) + P_{buf\_static}$$

where  $C_{local\_wire}$  is the capacitance of the local interconnects inside the cluster driven by node  $j$ .  $S_o$  is the switching activity of the cluster output.<sup>1</sup> All the  $S$  values are calculated beforehand.  $C_{net}$  is the estimated output capacitance of wires and buffers contained in the net driven by the output, and  $P_{buf\_static}$  is the static power of the buffers contained in the net.  $C_{net}$  is changeable gate by gate. We use the *wire-load* model to obtain reasonable wire-capacitance estimation before placement and routing. Experimental results show that the estimated power has an excellent correlation with the reported power after placement and routing. Details are omitted due to page limit.<sup>2</sup>

## 4. ALGORITHM DESCRIPTION

### 4.1 Cluster Enumeration

We carry out a cluster enumeration procedure to get all the single-root clusters in the network. Figure 2 shows an example (ignore the dashed lines for now). Consider  $K = 14$  and  $M = 6$ , and we are to generate clusters rooted on node  $t$ . Following a topological order, the clusters rooted on the predecessors of  $t$ , such as  $r$  and  $s$ ,

<sup>1</sup> We only examine the case where each cluster has just one output here.

<sup>2</sup> Interested readers are referred to [5] for a thorough analysis based on a similar wire-load model.

have already been generated. All the clusters on  $t$  can be generated from the clusters on  $r$  and  $s$ , following a dynamic programming approach. For example, we can first retrieve all the clusters of size 2 on  $r$ :  $\{m,r\}$ ,  $\{n,r\}$ , and  $\{o,r\}$ , and then combine one such cluster with each of the clusters of size 3 on  $s$  (adding root node  $t$  will make a cluster of size 6 on  $t$ ). If some node, such as  $n$ , appears from both sides, clusters of size 4 on  $s$  will be returned and tested for feasibility. This simple technique will effectively handle the reconvergent paths in the network. Next, we can try to combine clusters of size 3 on  $r$  with clusters of size 2 on  $s$ , and so on. Thus, all the feasible clusters can be generated. The number of clusters in the worst case is on the order of  $O(6^{3M+1})$  [19] for clustering 4-LUTs. The actual number is small when  $M$  is small. When  $M$  is large, heuristics can apply, such as *cluster pruning*, which fits well into our dynamic programming paradigm.<sup>3</sup>

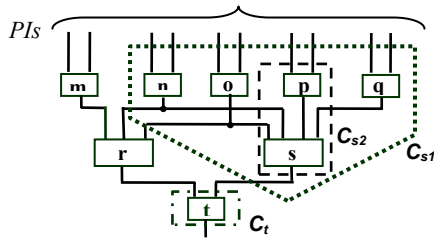


Figure 2: An illustrative example

## 4.2 Delay-Power-Vdd Curve Generation

A clustering solution at a node  $v$  is characterized by a delay-power-vdd point (solution point)  $\{d, p, sv\}$ , where  $d$  gives the arrival time at node  $v$  of the clustered subcircuit  $F_v$ ,  $p$  gives the corresponding power consumption for this clustered  $F_v$ , and  $sv$  is the supply voltage for the involved cluster on  $v$ . Suppose the relative delay values for  $V_H$  LUT ( $d_H$ ) and  $V_L$  LUT ( $d_L$ ) are 1 and 1.4, and the relative power values are 2 and 1 respectively. As a simplified example,  $C_{s1}$  will have two solutions,  $\{2, 10, V_H\}$  and  $\{2.8, 5, V_L\}$  as shown in Figure 2. Notice there are other clusters of various sizes on  $s$ , such as  $C_{s2}$ , and each has its own solution points. We can merge all of the solution points from all the clusters rooted on  $s$  and prune away all the inferior solutions, such as  $\{4, 10, V_H\}$  and  $\{4.8, 5, V_L\}$  (suppose  $D_{edge} = 2$ ). Thus, for the clustering solutions on  $s$ , we guarantee that for any two solution points  $\{d_1, p_1, sv_1\}$  and  $\{d_2, p_2, sv_2\}$ , if  $d_1 > d_2$ , then  $p_1 < p_2$ , and if  $d_1 < d_2$ , then  $p_1 > p_2$ . These solution points will form a curve with discrete points if drawn along delay and power axes. After the solution curves are formed for  $r$  and  $s$ , we can generate solution points for cluster  $C_t$ . Figure 3 shows the concept. In general, when we generate the delay-power-vdd curve for a cluster  $C$  rooted on  $v$ , we first retrieve the solution curves of the nodes in  $input(C)$ . We then calculate all the valid delay values that can be propagated from  $input(C)$  to  $v$  through  $C$ . Two cases are considered for  $C$  using either  $V_H$  or  $V_L$ . For each solution point  $sp_j$  in the curve of node  $i$ , where  $i \in input(C)$ :

- 1)  $C$  uses  $V_H$ . Thus all the LUTs in  $C$  are using  $V_H$ .

$$D_{j-i-H} = sp_j \cdot d + D_{i-H} + [D_{conv}] + D_{edge}$$

where  $D_{j-i-H}$  is the delay propagated to  $v$  through input  $i$  by  $sp_j \cdot d$ ;  $sp_j \cdot d$  is the delay stored in the current  $sp_j$  point;  $D_{i-H}$  is the path delay from  $i$  to  $v$  going through the LUT(s) in  $C$  (a multiple of  $d_H$ );  $D_{conv}$  is the converter delay that only occurs when  $sp_j \cdot sv$  is  $V_L$ .

- 2)  $C$  uses  $V_L$ . Thus all the LUTs in  $C$  are using  $V_L$ .

$$D_{j-i-L} = sp_j \cdot d + D_{i-L} + D_{edge}$$

where  $D_{i-L}$  is the path delay from  $i$  to  $v$  going through the LUT(s) in  $C$  (a multiple of  $d_L$ ). There is no need for  $D_{conv}$  here.

After we go through all the  $sp_j$  for each node in  $input(C)$ , all the valid delay values that could appear for  $C$  are collected. However, we need to validate the range of such values. We calculate the earliest and the latest time a signal may arrive at  $v$ . For  $C$  using  $V_H$ :

$$A_{min\_vH} = \text{MAX}_{i \in input(C)} \{ \text{MIN}_{sp_j \in \text{curve of } i} (D_{j-i-H}) \}$$

$$A_{max\_vH} = \text{MAX}_{i \in input(C)} \{ \text{MAX}_{sp_j \in \text{curve of } i} (D_{j-i-H}) \}$$

For  $C$  using  $V_L$ , a similar formula is defined but replaces  $D_{j-i-H}$  with  $D_{j-i-L}$ . The number of valid delay values is restricted by the range  $[A_{min\_vH}, A_{max\_vH}]$  for  $C$  with  $V_H$  and  $[A_{min\_vL}, A_{max\_vL}]$  for  $C$  with  $V_L$ . This number determines how many delay-power-vdd points are going to be generated for  $C$ . We define the *arrival time set* as in [19]. It is a set with the cardinality of  $|input(C)|$ , with each element corresponding to an arrival time at one input  $i$ . For each valid delay  $D_x$ , there is a corresponding arrival time set  $A_x$  on  $input(C)$ . Given a  $D_x$ , on each input  $i$ , the point  $sp_{x,i}$  with the smallest power  $sp_{x,i} \cdot p$ , whose  $sp_{x,i} \cdot d \leq D_x$  is picked ( $sp_{x,i} \cdot d$  is the arrival time at input  $i$  in the corresponding arrival time set  $A_x$ ). All such power values determined by one arrival time set are accumulated. The accumulated power amount plus the power consumed by  $C$  itself form the power part of one delay-power-vdd point, whose delay is  $D_x$ , and whose voltage is  $V_H$  when working with  $[A_{min\_vH}, A_{max\_vH}]$  and  $V_L$  when working with  $[A_{min\_vL}, A_{max\_vL}]$ . After all the points are generated for  $C$ , inferior solutions will be pruned away. This process repeats for each feasible cluster rooted on  $v$ , and then all the solution points will be merged together and will go through pruning. This forms the entire solution curve for current node  $v$ . This curve represents all the feasible solutions available for  $F_v$ . The solution generation procedure starts from PIs in a topological order and continues iteratively until all the PO nodes are reached. One concern is that the number of  $D_x$  in  $[A_{min}, A_{max}]$  may explode because all of our delay values, such as  $d_H$ ,  $d_L$ , and  $D_{conv}$ , are real numbers although they are different constants. Theorem 1 guarantees that we still have an efficient algorithm.

**Theorem 1:** The solution-pruning procedure keeps the minimum number of solution points,  $W$ , without loss of optimality.  $W$  is upper bounded by  $L \left(\frac{L}{2} + 1\right)^2$ , where  $L = level(v)$  for node  $v$ . The complexity of curve generation is  $O(K \cdot W \cdot \log(W))$  for each cluster.

## 4.3 Final Clustering Solution

After cluster enumeration with solution curve generation, the optimal clustering delay can be obtained through the propagated minimum delay values among all the non-inferior solutions. This optimal clustering delay is set as the required time for the design. The critical path is always driven by  $V_H$ , and clusters on non-critical paths can be driven by  $V_L$  to reduce power when such a

<sup>3</sup> In the example, when the numbers of clusters are controlled on node  $r$  and  $s$  through pruning, the cluster number on  $t$  will be small as well.

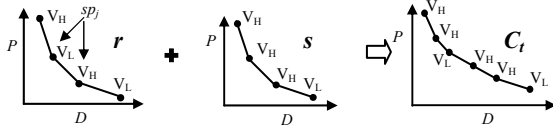


Figure 3: Solution curve generation

delay relaxation will not violate the required time of the design. Clusters rooted on the POs are generated first, and then the inputs of the generated clusters are iteratively processed.

There is one complication because of the involvement of level converters. When we try to select a cluster  $C$  on  $v$ ,  $C$  can either be a  $V_L$  cluster or a  $V_H$  cluster depending on which setting provides larger power savings. If we pick a solution of  $C$  where  $sp_j.sv = V_H$ , it is fine as long as  $sp_j.d \leq R(v)$ , where  $R(v)$  is the required time on  $v$ . However, if we pick a solution with the  $V_L$  setting, there may be a converter required between this cluster and the clusters on its fanouts if some of the fanout clusters are using  $V_H$ . If this is the case,  $R(v)$  may not be valid anymore due to  $D_{conv}$ . To deal with this issue, we use two types of required times:  $R$  if  $C$  tries  $V_H$  and  $R_{lv}$  for  $V_L$ . These two required times propagate from POs to PIs separately with the same initial value and interact with each other. If the picked cluster  $C$  on  $v$  is a  $V_H$  cluster, an input  $i$  on  $C$  has:

$$R(i) = R(v) - D_{i,H} - D_{edge}; \quad R_{lv}(i) = R(i) - D_{conv}$$

If  $C$  is a  $V_L$  cluster

$$R(i) = R_{lv}(v) - D_{i,L} - D_{edge}; \quad R_{lv}(i) = R(i)$$

The minimum  $R(i)$  and  $R_{lv}(i)$  propagated back among all the fanouts ( $v$  is one of them) of  $i$  are final  $R$  and  $R_{lv}$  for  $i$ . Next, to pick a cluster on  $i$ , we go through each delay-power-vdd point of every cluster rooted on  $i$  to find the best power solution:

$$P_{min} = \text{MIN}_{\forall C \text{ on } i} [\text{MIN}_{\forall sp \text{ on } C} sp.p]$$

given the corresponding delay of  $P_{min}$  fulfills the following:

$$\begin{aligned} D_{P_{min}} &\leq R(i) && \text{if } V_{P_{min}} \text{ is } V_H \\ D_{P_{min}} &\leq R_{lv}(i) && \text{if } V_{P_{min}} \text{ is } V_L \end{aligned}$$

The cluster with  $P_{min}$  is picked for node  $i$ . The cluster uses the voltage  $V_{P_{min}}$ . However, if  $D_{P_{min}}$  violates the specified required time, the next best  $P_{min}$  will be examined until finding a feasible  $P_{min}$ . This procedure continues until all the PIs are reached.

**Theorem 2:** The presented clustering algorithm will generate delay and power optimal solutions for Boolean networks that are trees, and generate delay optimal low-power solutions for networks as DAGs, targeting FPGAs with dual-Vdd architecture.

## 5. EXPERIMENTAL RESULTS

We will show the comparison results between the dual-Vdd clustering algorithm and the single-Vdd clustering algorithm to examine how much power savings a dual-Vdd FPGA architecture can achieve through effective circuit clustering. We implement a single-Vdd clustering algorithm, *SVpack*. *SVpack* follows the delay and power propagation procedure shown in Sections 4.1 and 4.2, and relaxes non-critical paths to select clusters with smaller power cost. All the LUTs have the same delay under a 1.3v single Vdd. On the other hand, dual-Vdd settings use  $V_H$  as 1.3v and  $V_L$  as 0.8v, 0.9v or 1.0v respectively. We call our dual-Vdd clustering algorithm *DVpack*. We run both *SVpack* and

*DVpack* on 25 MCNC benchmarks. It shows that the combination of  $V_H$  as 1.3v and  $V_L$  as 0.8v offers the best power savings of 20.3% on average. The power savings of 1.3v/0.9v is 19.5%, and 1.3v/1.0v is 18.4%. Details are omitted due to page limit.

## 6. CONCLUSIONS AND FUTURE WORK

We presented a delay optimal low-power clustering algorithm *DVpack* for FPGA architecture with a dual-Vdd setting. We built all the non-inferior delay-power-vdd solution points for a gate with consideration of dual Vdds and level converters. Our algorithm is delay and power optimal for trees and delay optimal for DAGs under the general delay model. Our future work is to explore different techniques for area reduction and examine the impact of these techniques on power consumption.

## 7. ACKNOWLEDGMENTS

This work is partially supported by NSF Grant CCR-0306682 and by Altera Corp. under the California MICRO program. We would like to thank Professor Lei He from UCLA Electrical Engineering Department for helpful discussions on this project.

## REFERENCES

- [1] J. Anderson and F. N. Najm, "Power-Aware Technology Mapping for LUT-Based FPGAs," *IEEE Intl. Conf. on FPT*, 2002.
- [2] V. Betz and J. Rose, "Cluster-Based Logic Blocks for FPGAs: Area-Efficiency vs. Input sharing and Size," *CICC*, May 1997.
- [3] E. Bozozzadeh, et al., "RPACK: Routability-driven Packing for Cluster-Based FPGAs," *ASPDAC*, Jan. 2001.
- [4] D. Chen, J. Cong, and Y. Fan, "Low-Power High-Level Synthesis for FPGA Architectures," *ISLPEd*, Aug. 2003.
- [5] D. Chen, J. Cong, F. Li, and L. He, "Low-Power Technology Mapping for FPGA Architectures with Dual Supply Voltages," *FPGA*, Feb. 2004.
- [6] J. Cong and M. Romesis, "Performance-Driven Multi-Level Clustering with Application to Hierarchical FPGA Mapping," *DAC*, Jun. 2001.
- [7] M. Hamada, et al., "A Top-down Low Power Design Technique Using Clustered Voltage Scaling with Variable Supply-voltage Scheme," *CICC*, May 1998.
- [8] E. Kusse and J. Rabaey, "Low-Energy Embedded FPGA Structures," *ISLPEd*, Aug. 1998.
- [9] J. Lamoureux and S.J.E. Wilton, "On the Interaction between Power-Aware CAD Algorithms for FPGAs," *ICCAD*, Nov. 2003.
- [10] E.L. Lawler, K.N. Levitt, and J. Turner, "Module Clustering to Minimize Delay in Digital Networks," *IEEE Transactions on Computers*, Vol. C-18, No. 1, Jan. 1966.
- [11] F. Li, D. Chen, L. He, and J. Cong, "Architecture Evaluation for Power-efficient FPGAs," *FPGA*, Feb. 2003.
- [12] F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using Dual-Vdd/Dual-Vt Techniques," *FPGA*, Feb. 2004.
- [13] A. Marquardt, V. Betz, and J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," *FPGA*, Feb. 1999.
- [14] R. Murgai, R.K. Brayton, and A. Sangiovanni-Vincentelli, "On Clustering for Minimum Delay/Area," *ICCAD*, Nov. 1991.
- [15] R. Rajaraman and DF Wong, "Optimal Clustering for Delay Minimization," *DAC*, Jun. 1993.
- [16] S. Raje and M. Sarrafzadeh, "Variable Voltage Scheduling," *ISLPEd*, Apr. 1995.
- [17] A. Singh and M. Marek-Sadowska, "Efficient Circuit Clustering for Area and Power Reduction in FPGAs," *FPGA*, Feb. 2002.
- [18] K. Usami and M. Horowitz, "Clustered Voltage Scaling for Low-Power Design," *ISLPEd*, Apr. 1995.
- [19] H. Vaishnav and M. Pedram, "Delay Optimal Clustering Targeting Low-Power VLSI Circuits," *TCAD*, Vol.18, No.6, Jun. 1999.
- [20] S. S. C. Yeh, et al., "Gate Level Design Exploiting Dual Supply Voltages for Power-driven Applications," *DAC*, Jun. 1999.