A Hightly Testable Pass Transistor based Design Methodology

Abstract

In this paper, we describe a highly testable structured ASIC design methodology which utilizes a regular, pre-fabricated array of pass transistor logic based if-then-else (ITE) cells as the building block for the circuit. Given a logic netlist, we first construct Reduced Order Binary Decision Diagrams (ROBDDs) for the circuit *in a partitioned manner*, thereby allowing the approach to handle large designs. Test generation for of each of these partitions can be performed extremely efficiently. Using scan techniques, the ATPG problem is made independent across partitions. Also, the fault excitation and fault propagation problem within any partition are both performed in linear time (in the size of the ROBDD of the partition). Additionally, the fault excitation and fault propagation problems have non-overlapping variable supports, thereby simplifying the problem further. Finally, we place the ROBDD nodes in a manner that minimizes crossings in the ROBDD graph. Our placement also effectively 'folds' the ITE cells of different variables into a single row, so as to obtain a layout with a more uniform distribution of ITE cells along each physical row of ITE cells. The design methodology has been demonstrated to implement sequential as well as combinational designs, with low area and delay overheads compared to an ASIC approach.

1. Our Approach

In the rest of this section, we describe the various aspects of our approach. Our low-NRE, METAL and VIA mask programmable PTL based circuit design approach consists of a pre-fabricated array of ITE cells, implemented in an area and delay-efficient manner. The details of these cells are provided in Section 1.1.

Given a logic netlist, we create a partitioned ROBDD [1, 2] for this circuit, using a bottom-up construction. When the size of any ROBDD exceeds a user-specified bound B, a new ROBDD variable is created, and used in constructing subsequent ROBDDs. This partitioned construction approach enables the construction of ROBDDs for very large designs. Our synthesis approach is described in Section 1.2. Alternate recursive bi-partitioning based methods for ROBDD construction for PTL based designs [3, 4] could be utilized as well.

Each ROBDD node is implemented as an ITE cell. Before this is done, we replicate ITE cells if their corresponding ROBDD node has more than a user-specified number p of incoming edges. Next we rearrange the ROBDD nodes into rows of the array, such that each row is assigned at most a user-specified number k of variables. This ensures that each row has a balanced number of ROBDD nodes assigned to it, for area efficiency. The resulting circuit graph is next placed so as to minimize the number of edge crossings. Finally, the placed design is routed, using up to 4 METAL layers. The details of our placement and routing approaches are described in Section 1.4.

1.1 ITE Cell Design

The ITE cell we utilize in our pre-fabricated array is based on a NMOS pass-gate structure shown in Figure 1. The outputs of the NMOS MUX is used to generate a buffered output and its complement. In Figure 1, T and E represent the THEN and ELSE branches respectively of the ITE cell. The mux control input is i. The ITE cell generates both the output, out and its complement, $\overline{out}$. The corresponding layout of the ITE cell is shown in Figure 2.

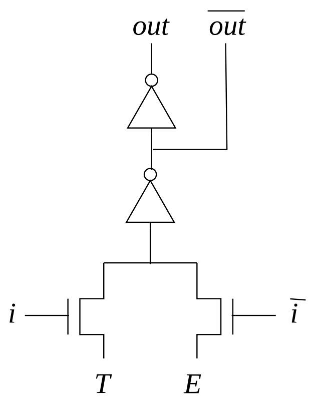

Figure 1: ITE Cell Circuit

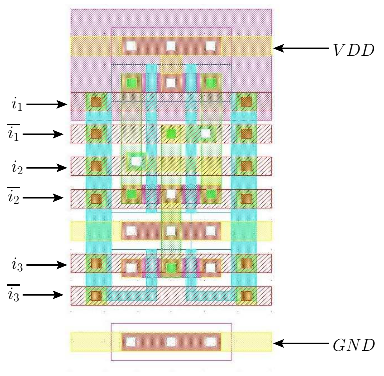

Figure 2: ITE Cell Layout

Note that in the layout, the mux control signals (in polysilicon) run along the length of the cell. Also, each ITE cell has three variable signals and three complemented variable signals that traverse over it horizontally in METAL3. By appropriately placing stacked vias at the horizontal wire of the appropriate variable (and its complement), the ITE cell can be connected to any one of the three variables in the corresponding row of the array. The ITE cell uses METAL1 and METAL2 for the layout, and METAL3 to route the variables and their complements. Routing is done using all 4 metal layers.

We performed experiments with a complementary pass gate structure, to test its utility in our problem. It turns out that the delay of the complementary pass-gate is about the same as that of the NMOS pass-gate shown in Figure 1. This is because of the increased diffusion capacitance of the additional PMOS transistor. Also, since the output of every NMOS pass-gate is buffered, the delay variation of an ITE cell due to V_T variations is low. Further its area was more than that of the circuit in Figure 1. Hence, we decided to use NMOS pass-gates instead of complementary pass-gates in our experiments.

1.2 Partitioned ROBDD Construction

Logic synthesis in our approach is performed using a partitioned ROBDD construction approach. Initially, the primary input variables of the design are ordered using a DFS ordering. We enable dynamic variable ordering before building ROBDDs. At this point, we begin a bottom-up construction of ROBDDs. At any point in the construction, assume that the set of variables in the ROBDD manager is V. During ROBDD construction, if the size of any ROBDD increases beyond a user-specified threshold B, we introduce a new variable v (which is the output of the ROBDD that reached the threshold B), and continue building ROBDDs, after setting $V \leftarrow V \cup v$. At the end of the ROBDD construction process, we have a series of ROBDDs, each of size bounded by B. The output f_i of these ROBDDs either represents a primary output of the circuit, or an intermediate ROBDD variable v_i. *Using this partitioned approach to ROBDD construction allows us to handle large designs easily.*

Figure 3 illustrates how the intermediate variables in the partitioned ROBDD construction are treated in the ITE array. Consider the multi-level logic network η of Figure 3 a). This network consists of 4 primary input variables $X = \{x_1, x_2, x_3, x_4\}$. As the bottom-up ROBDD construction proceeds, assume that new variables y_1 and y_2 are created. Then the output z is built in terms of y_1, y_2 and X. Note that in the example shown in Figure 3, z is not directly dependent on the primary input variables X. The logic functions of y_1 and y_2 are in turn built in terms of X. For this partitioned set of ROBDD nodes representing the logic network η, the corresponding ITE based array is shown in Figure 3 b). In this figure, we note that the primary inputs are presented to the circuit such that if $x_i < x_j$ in the ROBDD variable ordering, then x_i is above x_j in the ITE array. Also, the two variables y_1 and y_2 are produced as the outputs of some ITE cells (c_1 and c_2 respectively) whose inputs are a subset of X. The outputs of these cells are driven out as the variables y_i and y_j respectively. In Figure 3 b), we have assumed that each row of the array corresponds to a single variable. In practice, each row of the array in our design can contain ITE cells which depend on one of k variables. Also, the outputs c_i are buffered and driven to the variable drivers y_i. All y_i variables are located above the X variables in the ITE array. The task of routing the output of c_i to the corresponding variable drivers y_i is left to the routing algorithm.

1.3 Efficient ATPG

The proposed approach yields an extremely efficient Automatic Test Pattern Generation (ATPG) algorithm. In traditional standard-cell based circuits, even after utilizing scan techniques, the resulting combinational ATPG problem is NP-complete.

In our approach, we scan the output nodes of all ROBDD partitions. In functional mode, these outputs may be the inputs of other ROBDD partitions. In test mode, these outputs are scanned, allowing for different ROBDD partitions to be tested independently. In other words, in test mode, *all* variables of all partitions are independent. Hence, the ATPG problem of each partition is independent.

Also, recall that each ROBDD partition has a maximum size B. This partitioned approach allows the construction of (partitioned) ROBDDs even in circuits for which monolithic ROBDDs cannot be computed. Now since each node in the partitioned ROBDD corresponds to a ITE cell in our approach, the ATPG problem is significantly simplified. Consider the ATPG problem for an internal node x stuck at v. Here x is the output of an ITE cell in our approach, and $v \in \{0, 1\}$. Let the output of the partition be f.

Then a test for x stuck at v is given by:
$T_{x-stuck-at-v} = (x \equiv \overline{v}) \cdot (\frac{\delta f}{\delta x})$
In the above expression, the first parenthesized expression is the fault excitation condition, while the second expression is the fault propagation condition (the boolean difference of f with respect to x). The efficient computation of each of these conditions is performed as follows.

- The fault excitation condition consists of the exercise of setting x to $\overline{v}$. In our approach, this is simply performed by computing *a* path of the logic function $\overline{x}$.

  In the ROBDD of the function f (which was computed at the time of logic synthesis), we find the node corresponding to x. Then we find a path rooted at this node, to the $\overline{v}$ terminal vertex. This can be done in linear time, using a simple ROBDD based recursive algorithm.

- The fault propagation condition is computed in linear time as well, in the following manner. In the partitioned ROBDD of f, we find a path from the node f to the node x. This is performed in linear time.

An important observation is that the support variables for both the above conditions are non-overlapping, making the ATPG problem extremely easy.

1.4 Placement and Routing

After partitioned ROBDD construction, we obtain several ROBDDs representing the decomposed logic netlist. We first replicate ITE cells whose outputs are heavily loaded (this corresponds to ROBDD nodes which have high in-degrees). If the in-degree of ROBDD node is k, we replicate it $\lceil k/K \rceil - 1$ times. We use $K = 3$ in our experiments. This limited the fanout of any ITE cell to a maximum of 3.

Suppose we have a total of N ITE cells (after duplication of high-fanout nodes). We then compute the number of

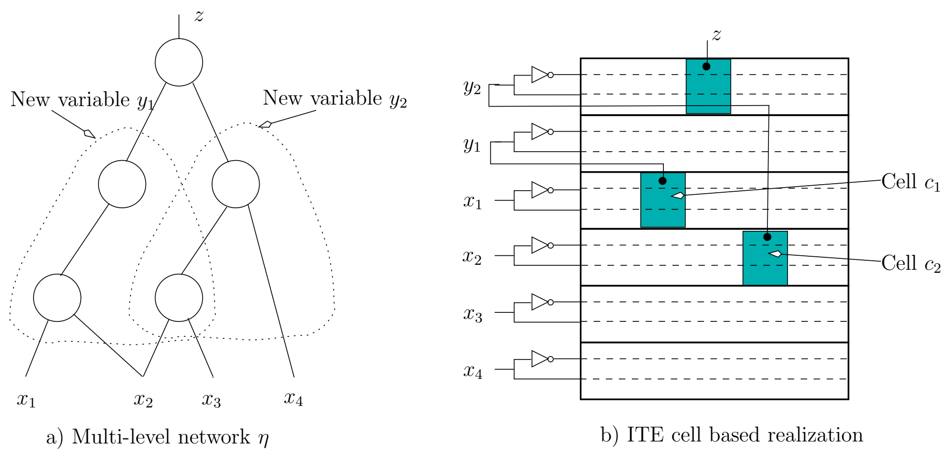

Figure 3: Partitioned ROBDD construction and its corresponding ITE Cell based Realization

ITE cells n in any row of the ITE array, and the number of rows of the ITE array m as follows:

$x \cdot n \simeq y \cdot m$

$n \cdot m \geq N$, where x and y are the width and height respectively of the ITE cell.

In this way, we find an initial estimate of the number of rows m of the ITE array, and the number of ITE cells n in each row.

Now we sort the N ITE cells in an increasing order of their ROBDD variable index. Assume that if variable v_i is closer to the root than variable v_j, then $index(v_i) < index(v_j)$. Now ITE cells are assigned to rows of the ITE array. In order to do this, we consider two situations:

- Suppose there are n_j ITE cells with ROBDD variable index v_j, such that $n_j > n$. In this case, these ITE cells need to span $\lceil n_j/n \rceil$ rows in the ITE array. We sort these n_j ITE cells in decreasing order of the cost C. The cost C for a ITE cell c is:

  $C = \Sigma_{i,j}([index(c) - index(c_j)] - [index(c_i) - index(c)])$

  where c_i corresponds to children of the node c, and c_j corresponds to the parents of nodes c.

  If the nodes of the ITE array were planarized, nodes c with high cost C are nodes which would induce more edge crossings if they were placed in lower ITE rows. Conversely, a lower cost indicates that a node c would induce more crossings if they were placed in higher ITE rows.

- Suppose there are n_j ITE cells with ROBDD variable index v_j, such that $n_j < n$. In this case, we will attempt to populate the corresponding row of the ITE array with additional ITE cells with variables v_{j+1}. If the corresponding row still is not full, we will add ITE cells with variables v_{j+2} as well. Any row of the ITE array has ITE cells which depend on at most 3 variables in it (since the number of variables that can be routed over any ITE cell is 3).

After the initial sorting of the N ITE cells in an increasing order of their ROBDD variable index, and a subsequent sorting of the ITE cells of variables v_j in terms of the cost C, we have a sorted array A of ITE cells. Now we simply assign these ITE cells to rows of the ITE array, starting from the top row, until rows are fully populated.

Finally, the placement of the ITE cells within a row is done in a manner that minimizes edge crossings in the induced graph. We use a graph visualization tool DOT [5] to perform this crossing minimization. The output of DOT effectively re-arranges cells in each ITE row, in a manner that minimizes edge crossings[1]. DOT is not allowed to modify the assignment of ITE cells to rows of the ITE array.

The result of DOT induces the placement we use for our design. At this point, we invoke a routing tool (WROUTE in SEDSM [7]) to route the ITE cell array. The lowest 4 METAL layers and their associated VIA layers are utilized in the routing, allowing us to realize a design using changes only to the METAL and VIA layers.

Sequential designs are handled by co-locating a bank of flip-flops along the edge of each row of the ITE based array (3 flip-flops per row since each row has at most 3 variables in it). The outputs of any of these flip-flops can drive one of the inputs of the corresponding row of the ITE array, by means of a METAL and VIA mask change. If the input of a flip-flop is computed by the combinational logic in the design, then this routing connection is made (using METAL and VIA mask changes only) during the routing phase.

2. Experimental Results

In our experiments, a 0.1μm process was assumed. Our approach was implemented using several tools. Partitioned ROBDD construction was performed using the *frontier method* in VIS [8]. For each of the examples we tried different values (5, 10, 15, 20 and 1000) of the partitioning threshold number B. Partitioning is invoked when the size of a ROBDD exceeds this bound B. We then took the result that gave the smallest number of ROBDD nodes. The resulting ROBDDs, along with an array consisting of pairs of values representing the correspondence between an ROBDD f_i and its internal variable y_i, are supplied to a pre-placement routine, which performs initial ITE cell replication as necessary, as well as assignment of ITE cells to rows of the ITE array. This pre-placed design is passed to DOT, which minimizes the crossings in the graph in-

[1]Details on the algorithms used by DOT can be found in [6]

duced by the interconnections among the ITE cells. In other words, DOT is only allowed to re-arrange ITE cells *within a row*, in a manner that minimizes edge crossings. The result of DOT is used as the final placement of our design, and used to perform routing of the final design using the WROUTE router within SEDSM [7]. Sequential designs are handled as well, as described in Section 1. The only mask layers that require customization in our approach are METAL (lowest 4 layers) and VIA layers.

Table 1 describes the delay and area comparisons of our method versus standard cells. The circuits in the table are from the MCNC91 benchmark suite. The top half of the table reports the delays and areas for combinational designs, while the bottom half of the table reports delays[2] and areas for sequential designs. For standard cell based comparisons, we first performed technology independent optimizations using script.rugged in SIS. We then mapped our design to a library consisting of a total of 20 standard cells (including a DFF). Placement and routing for the standard cell based designs was performed using the same 0.1μm process, using SEDSM [7].

Delays for the ITE-cell based design were computed by doing a topological sort of the ITE array, from inputs to outputs. The delay at any node was computed as

$$D(node) = Max[D(leftchild), D(rightchild)] + D(ITEblock).$$

The delay of the ITE cell was characterized in SPICE for a 0.1μm BPTM process [9] assuming a fanout of 3. Since the maximum load that the output of any ITE cell drove was equal to the load we used in SPICE, the delay estimates are conservative. Further, when variables are internal nodes in the design the delay computation is performed as:

$$D(node) = Max[D(variable),\ D(leftchild), D(rightchild)] + D(ITEblock).$$

This is true since the variable needs to be computed and driven in case the ITE blocks were built in a partitioned manner.

For delay estimates of the standard cell implementation, we used the *sense* [10] package in SIS [11]. This package returns the longest sensitizable path in a design (as opposed to static timing analysis which returns potentially false paths). For both styles of implementation, interconnect delay was not considered.

Note that in spite of our method being a mask programmable solution, its delay is quite competitive, being just 1.5× larger (on average) than the standard cell delay for sequential designs. The delay overhead for combinational designs , compared to the standard cell value, is slightly larger (2.01×). The area penalty of our scheme over standard cells is on average 3.41× (6.08×) for sequential (combinational) designs. Combinational designs have higher overheads since they have unused resources in the form of flip-flops.

Figure 4 illustrates the placed-and-routed result for the alu2 benchmark circuit. Note the presence of the flip-flop regions on either side of the array. These flip-flops are not used for combinational designs, but contribute to the area penalty of these designs.

The overheads associated with our technique are comparable with similar structured ASIC methodologies. In [12], the authors report an average area overhead of 3.44× (4.96×) for sequential (combinational) designs and an average delay overhead of 3.58× (2.89×) for sequential (combinational) designs.

[2]For sequential designs, the delay reported is for the combinational part of the circuit.

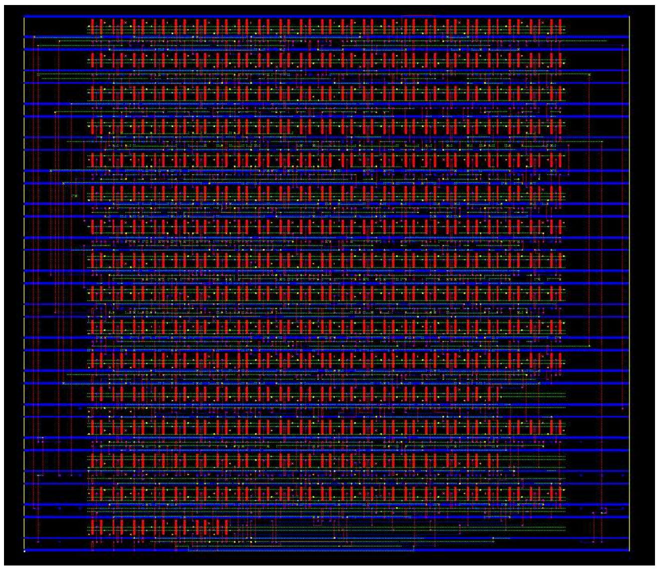

Figure 4: ITE Array for alu2

3. Conclusion

With the increasing cost and complexity of fabrication masks in modern VLSI processing technologies, there is a strong need for VLSI design approaches that amortize the non-recurring expense (NRE) associated with mask generation, over a large number of designs. In this paper, we have proposed a structured IC design methodology which is customizable using METAL and VIA masks only. Our approach utilizes a pre-fabricated array of pass transistor logic based if-then-else (ITE) cells as the building block for circuit. It differs from other approaches, such as those in [13, 14, 15, 12], primarily in the granularity of the logic block used.

Given a logic circuit, we first construct ROBDDs for the circuit in a partitioned manner. We then place the ITE cells (which correspond to ROBDD nodes) in a manner that minimizes crossings in the ROBDD graph. Our placement also effectively 'folds' the ITE cells of different variables into a single row, so as to obtain a layout with a more uniform distribution of ITE cells along each physical row of the ITE array. Also, ITE cells with high fanout are replicated. Experimental results demonstrate that our methodology implements combinational designs with an overhead of 2.01× (in speed) and 6.08× (in delay) compared to a standard-cell based approach. For sequential designs, the corresponding overheads are 1.55× and 3.41× respectively.

References

[1] A. Narayan, J. Jain, M. Fujita, and A. Sangiovanni-Vincentelli, "Partitioned ROBDDs-a compact, canonical and efficiently manipulable representation for boolean functions," in *Proceedings, IEEE/ACM International Conference on Computer-Aided Design*, pp. 547–554, Nov 1996.

[2] A. Narayan, A. J. Isles, J. Jain, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Reachability analysis using

Ckt	Evaluation Delay			Area		
	Stdcell	ITE	Ovh	Stdcell	ITE	Ovh
alu2	770	500	0.65	1314.1	2560	1.95
alu4	1020	527	0.52	2500	5068.8	2.03
apex6	500	1310	2.57	2678.1	14585.6	5.45
apex7	440	1030	2.34	885.1	4608	5.21
C1908	880	2590	2.91	1827.6	8288	4.53
C3540	1250	3050	2.44	4323.1	29491.2	6.82
C432	930	3070	3.30	715.6	4640	6.48
C499	600	1070	1.78	1827.6	3974.4	2.17
C880	1210	2750	2.27	1463.1	8985.6	6.14
dalu	1110	2460	2.22	3164.1	39916.8	12.62
frg2	810	1700	2.10	2575.6	24441.6	9.49
i8	880	1560	1.77	4064.1	40320	9.92
i9	850	810	0.95	2383.2	14035.2	5.89
t481	720	600	0.83	2626.6	6080	2.31
term1	320	730	2.28	663.1	2355.2	3.55
too_large	510	1550	3.04	1105.6	10560	9.55
vda	650	600	0.92	1508.03	6080	4.03
x1	380	950	2.50	1105.6	9625.6	8.71
x3	510	1660	3.25	2756.25	16844.8	6.11
x4	440	650	1.48	1314.1	11264	8.57
Avg			2.01			6.08
s1488	630	650	1.03	3277.6	6240	1.90
s1494	650	600	0.92	3108.1	6400	2.06
s208	270	550	2.04	105.1	1459.2	13.88
s344	390	650	1.67	715.6	2649.6	3.70
s349	410	650	1.59	742.6	2649.6	3.57
s386	290	550	1.90	885.1	2060.8	2.33
s444	380	700	1.84	1105.6	2880	2.60
s510	390	400	1.03	1105.6	3161.6	2.86
s526	330	700	2.12	1314.1	2355.2	1.79
s526n	330	700	2.12	1314.1	2457.6	1.87
s820	560	650	1.16	1827.6	3968	2.17
s832	570	650	1.14	1827.6	3968	2.17
Avg			1.55			3.41

Table 1: Comparison of our technique with standard cells

partitioned-ROBDDs," in *Proceedings, IEEE/ACM International Conference on Computer-Aided Design*, pp. 388–393, Nov 1997.

[3] S. Shelar and S. Sapatnekar, "Recursive bipartitioning of BDDs for performance driven synthesis of pass transistor logic circuits," in *IEEE/ACM International Conference on Computer Aided Design*, pp. 449–452, Nov 2001.

[4] F. Aloul, I. Markov, and K. Sakallah, "Improving the efficiency of circuit-to-BDD conversion by gate and input ordering," in *Proceedings, IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 64–69, Sept 2002.

[5] "Graphviz - graph visualization software." http://www.graphviz.org.

[6] "A method for drawing directed graphs- dot's algorithm." http://www.graphviz.org/Documentation/TSE93.pdf.

[7] Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA, *Envisia Silicon Ensemble Place-and-route Reference*, Nov 1999.

[8] The VIS Group, "VIS: a system for verification and synthesis." Proceedings of the 8th International Conference on Computer Aided Verification, p428-432, Springer Lecture Notes in Computer Science, #1102, Edited by R. Alur and T. Henzinger, New Brunswick, NJ, July 1996.

[9] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," in *Proc. of IEEE Custom Integrated Circuit Conference*, pp. 201–204, Jun 2000. http://www-device.eecs.berkeley.edu/ ptm.

[10] P. McGeer, A. Saldanha, R. Brayton, and A. Sangiovanni-Vincentelli, *Logic Synthesis and Optimization*, ch. Delay Models and Exact Timing Analysis, pp. 167–189. Kluwer Academic Publishers, 1993.

[11] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Tech. Rep. UCB/ERL M92/41, Electronics Research Laboratory, Univ. of California, Berkeley, CA 94720, May 1992.

[12] N. Jayakumar and S. Khatri, "A metal and via maskset programmable VLSI design methodology using PLAs," in *Proceedings, IEEE/ACM International Conference on Computer-Aided Design*, pp. 590–594, Nov 2004.

[13] A. Koorapaty, V. Chandra, K. Y. TONG, C. Patel, L. Pillegi, and H. Schmit, "Heterogeneous programmable logic block architectures," in *Design Automation and Test in Europe Conference*, Mar 2003.

[14] A. Koorapaty, L. Pillegi, and H. Schmit, "Heterogeneous logic block architectures for via-patterned programmable fabrics," in *13th International Conference on Field Programmable Logic and Applications*, Sep 2003.

[15] A. Koorapaty, V. Kheterapal, P. Gopalakrishnan, and L. Pillegi, "Exploring logic block granularity for regular fabrics," in *Design Automation and Test in Europe Conference*, vol. 1, pp. 468–473, Feb 2004.