Efficient Place and Routing CAD Techniques for the Reduction of Power Dissipation in FPGAs

M.V. Sushumna

CVR College of Engineering/ECE Department, Hyderabad, India
Email: sushumna451@gmail.com

Abstract: **The Computer Aided tools (CAD) are used to transform the design into a stream of '1's and '0's. This stream of bits is used to program the FPGA during the configuration time. Generally a huge number of programmable switches are used to implement FPGA. These switches are mainly used to implement functional blocks in FPGA. The CAD tools of FPGAs convert the design entered either using hardware description language or as schematic to a binary bit stream of '0's and '1's. During the configuration time, this bit stream effectively program the FPGA. On the CAD level, the packing and placement stages are modified to allow the possibility of dynamically turning OFF the idle parts of the FPGA to reduce the leakage power and as well as to reduce total power dissipation. A new activity generation algorithm is proposed and implemented that aims to identify the logic blocks in a design that exhibit similar idleness periods. Several criteria for the activity generation algorithm are used, including connectivity and logic function. Several versions of the activity generation algorithm are implemented to trade power savings with runtime. A newly developed placement and packing algorithm uses the resulting activities to minimize leakage power dissipation by packing the logic blocks with similar or close activities together. In this paper, the proposed architecture of FPGA using MTCMOS and the corresponding CAD tools save the average power of 30% in 90mm CMOS technology with less performance penalty.**

Key words—**Threshold voltage, CMOS, FPGA, CAD, Power dissipation.**

I. INTRODUCTION

Fueled by the increase in functionality and size of modern Field Programmable Gate Arrays (FPGAs), the market for FPGAs has witnessed a notable expansion in the past few years. This increase in demand has pushed FPGA vendors to design modern FPGAs using state-of-the-art CMOS technologies. Consequently, modern FPGAs suffer greatly from the deep submicron issues that affected the ASIC industry earlier along the technology scaling road. The biggest deep submicron challenge that hurdles further expansions of the use of FPGAs in the VLSI industry is leakage power dissipation. In order to face the ASIC industry the FPGA vendor must have less leakage power architectures and corresponding CAD techniques.

A. FPGA CLB Slice

The Configurable Logic Block (CLB) slice is the important logic resource of Xilinx Vertex-5 FPGA. It consists of 4 Look-Up Tables (LUT), multiplexers and carry logic. The complete Architecture of a slice in vertex-5 is shown in figure 1. In this slice 6-input logic functions can be realized by four 6-input LUTs and 8-inputs function also can be implemented with multiplexer by combining the outputs of two LUTs.

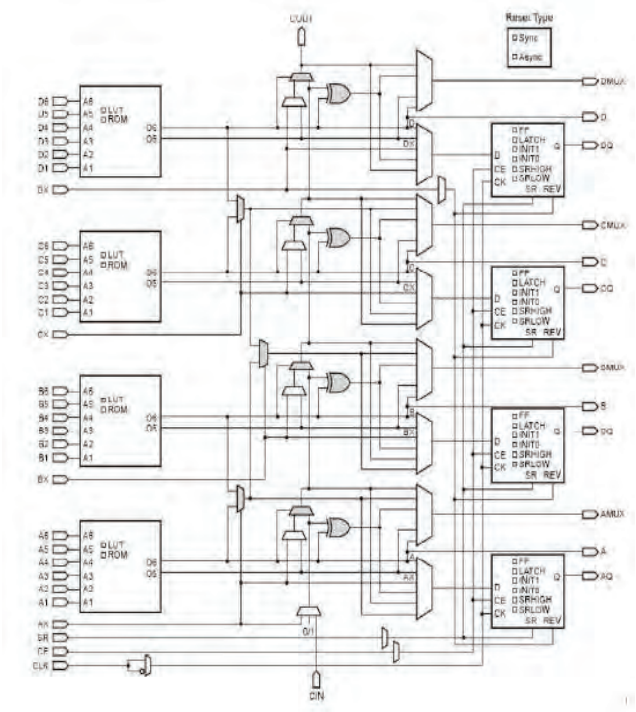


Figure 1: Xilinx's Vertex-5 slice architecture.

B. FPGA Routing Resources Architecture

The FPGA routing resources are divided into two categories which are segmented local routing and dedicated local routing. The connection between the logic blocks is done by using segmented local routing. The figure 2 shows the segmented local routing connections. These prefabricated wires in channels provide programmable connection between the logic blocks, switch blocks and connection blocks.

The dedicated routing is used to provide connection between global signals. Due to these global signals the logic blocks fan-out is increased. These are clock and reset, this gives low skew due to dedicated routing. These days some extra blocks PLLs and Delay Locked Loops (DLLs) are used in commercial FPGAs to further reduction of skew. In modern FPGAs, the different clock domains are available inside, to provide flexibility with respect to asynchronous designs.

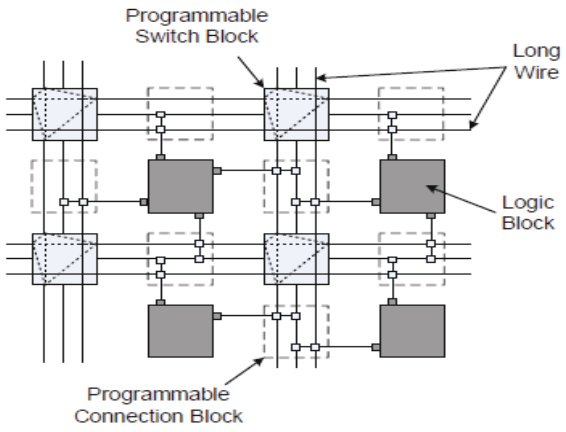


Figure 2: Routing resources in island-style FPGAs.

II. CAD FLOW FOR FPGAS

In an FPGA, to implement logic functions large numbers of programmable switches are required. For this programming, proper CAD tools are required. The CAD tools transform the design into the stream of bits, that program the FPGA during the configuration time. The design may be either entered as a schematic or Hardware Description Language (HDL). The CAD flow diagram of typical FPGA is shown in figure 3.

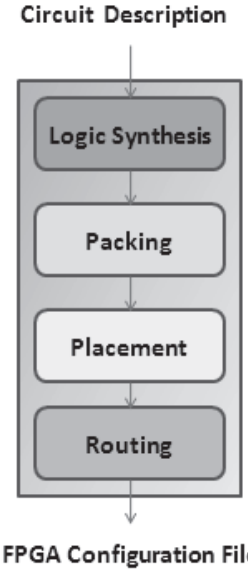


Figure 3: CAD flow diagram of typical FPGA

A. *Logic Synthesis*

Logic synthesis means the hardware description language is converted into equivalent net list of basic logic gates. The logic synthesis is performed in two stages which are "logic optimization" and "technology mapping" [2, 3]. The "logic optimization" is an independent stage that simplifies the logic function without using any technology information. At this stage logic redundancy is eliminated. In "technology mapping" stage, the optimized design is mapped into the corresponding Flip-Flops and LUTs in the CLB. Here each k-bounded logic functions are mapped with k-LUTs. This stage resolves the finding a set of k-feasible cuts. Hence the delay, power and area of the final implemented design are minimized. This technology mapping process is also called as covering problem.

B. *Packing*

The net-list of LUTs and flip-flops are converted to net-list of logic blocks using packing is shown in figure 4. The clusters of logic blocks from input net-list can be mapped into the physical logic blocks of FPGA. The packaging algorithms are used for minimizing the number of logic blocks, the delay along the critical path and the number of connections between the logic blocks. The physical limitations of the actual logic blocks of the FPGA have to consider the packing algorithms in view of total number of distinct input signals and maximum number of LUTs in a logic block.

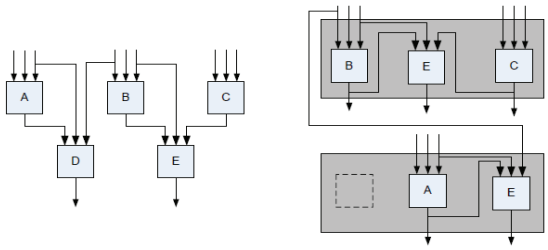


Figure 4: An example of packing.

The packing algorithms are categorized into bottom up [2, 4, 5] or bottom down [6, 7]. In the first algorithm, each cluster is formed individually around the main LUT until the cluster is full. Next, the second algorithm is concentrated on the partition of LUTs into clusters by using successive subdivision concept. The first algorithm is faster and simpler than second one because, only local connections are considered in the first method.

C. *Placement*

The packed logic blocks are distributed among the physical logic blocks in the FPGA by placement process. The delay along the critical path minimization and the resulting circuit routability is enhanced by the placement algorithms. There are three categories of placement algorithms; min-cut [8, 9], analytic [10, 11], and simulated annealing [12, 14] based algorithms. The Simulated Annealing (SA) placement tools depend on SA algorithms which are suitable for most of the commercial placement tools for FPGAs due to their flexibility to adapt to a wide variety of optimization goals [15]. First, for each logic block, the random initial placements are generated. Later, in order to improve the cost function a pair of logic blocks are selected for swapping as candidates at random. The candidate logic block is directly allowed if the cost function decreases, otherwise, the candidate logic block is allowed randomly, which decreases the process time of the algorithm. Finally, less number of swaps are placed in each and every iteration.

DOI: 10.32377/cvrjst1004

D. Timing Analysis

The placement and routing CAD tools in FPGAs are guided by time analysis [16] to obtain the speed of the routed and placed circuit and to identify the critical path during routing while estimating the slack of each source sink connection. A directed graph representing circuit is used for doing time analysis, where the nodes represent the LUTs or registers and the edges represent connections. The breadth first search method is used for finding the minimum required clock period for the primary inputs and to find the time arrival at node *i* is calculated from the below equation

$$T_{arrival}(i) = \max_{\forall j \in fanin(i)} \{T_{arriaval}(j) + del(j, i)\}$$

Here *del* (*j, i*) = delay on the edge between *j* and *i*.

Using the breadth search algorithm the required time at node *i* is obtained from the following relation starting from the primary outputs.

$$T_{required}(i) = \min_{\forall j \in fanout(i)} \{T_{required}(j) + del(ij)\}$$

Next, the slack on logic blocks connections between nodes *i* and *j* is obtained from the following equation

$$slack(i, j) = T_{required}(j) - T_{arrival}(i) - del(i, j)$$

The connections with zero slack in the logic blocks are called critical connections and connections with positive slack are called as non-critical connections, which are very long routed wires.

E. Routing

The different connections between the logic blocks in placed design of FPGA are programmed using routing resources or switches available, which is done in routing stage [13]. The main objective of the routing algorithm is to remove the congestion and delay in the routing resources and critical paths of FPGA respectively. In general, all the routing algorithms are classified into two groups which are detailed routers and global routers. Detailed routers assign the connections to specific wires in the FPGA, while global routers assign the connections by considering circuit architecture without looking at the number and type of wires available.

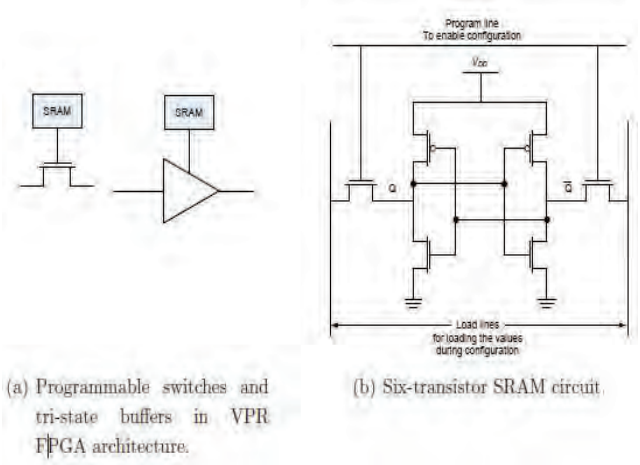
III. VERSATILE PLACE AND ROUTE (VPR) CAD TOOL

In this paper, the proposed CAD flow based on the Versatile Place and Route (VPR) is used for placement and routing in the FPGA. VPR is a popular placement and routing tool for FPGAs [13]. VPR is the core for Alter's modeling toolkit CAD tool [17] and it is preferred in a timing driven logic block packing algorithm in conjunction with T-VPack [4,12].

The VPR CAD tool contains two parts which are a placer and router. The optimum placement and route are found by interacting the placer and router components together for a given assets of conditions. This section gives the overview of VPR for FPGA architecture.

A. VPR Architectural Assumptions

The VPR architecture is assumed as SRAM based architecture and each SRAM contains configuration bits for all tri-state buffers and pass transistor multiplexers in the FPGA in both logic and routing resources shown in figure 5(a). The six transistor SRAM cells used in manufacturing SRAMs are shown in figure 5(b).



(a) Programmable switches and tri-state buffers in VPR FPGA architecture.

(b) Six-transistor SRAM circuit

Figure 5: The building blocks of SRAM programmable FPGAs used by VPR.

B. VPR Logic Architecture

VPR targets the hierarchal or cluster based logic architecture. In this architecture, a cluster of logic blocks are formed from the group of Basic Logic Elements (BLE). This each BLE consists of k- LUT and a 2:1 MUX and a D flip-flop as shown in the figure 6.

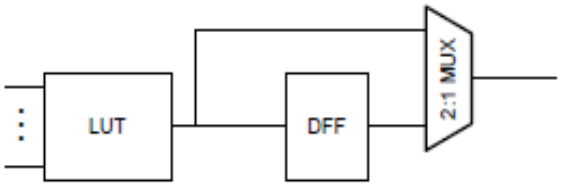


Figure 6: VPR BLE architecture.

BLEs inside the clusters are connected to each other. The input and output of the clusters are connected by using local routing which is shown in figure 7. The local routing is also used for connecting any of the inputs of BLEs to any of the outputs of the BLEs or any of the external inputs.

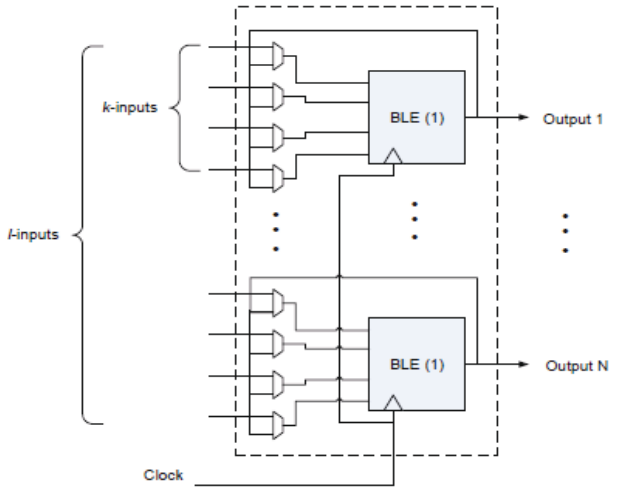


Figure 7: VPR logic cluster architecture.

The VPR logic cluster architecture consists of four main parameters in the FPGA architecture which are size of LUTs (k), no. of cluster BLEs (n), no. of external inputs of the cluster (I) and no. of external clock signals(Mclk). The LUTs in the VPR architecture are mainly implemented by small size transistors, then the parasitic capacitance of these small transistors are ignored.

C. VPR Routing Resources Architecture

In this section, the VPR routing resources architecture is presented. The VPR routing resources are characterized into 3 categories which are Switch block, Channel and wire parameters. The connections between input-output pads, channel width information is obtained by using channel routing resource. The channel width parameter includes the width of the vertical and horizontal routing channels, width of the input-output channel and no. of input-output pads of the one column or row of the logic clusters. The figure 8 shows the FPGA VPR architecture and routing model and the corresponding channel parameters.

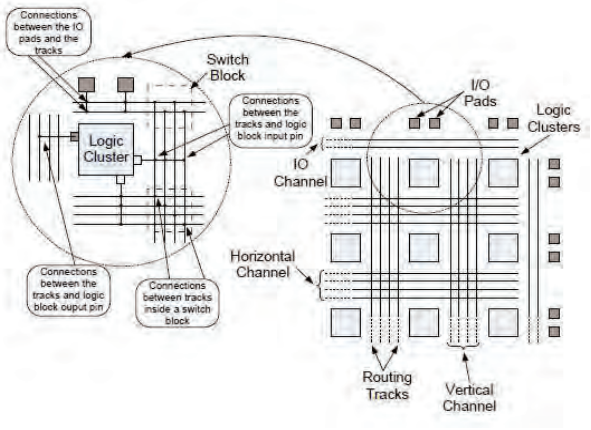


Figure 8: VPR FPGA routing architecture.

The switch blocks are used to make the programmable connection between the horizontal and vertical routing racks, and is shown in Fig. 8. The VPR characterizes the switch blocks by their resistance (R), input capacitance (Cin), output capacitance (Cout), connection flexibility (Fs), intrinsic delay (Tdel), switch type and switch block topology. The Fs - connection flexibility of a switch is defined as the no. of connections of the pins of the two sides of the switch. figure 9. shows the different topologies of switches supported by VPR.

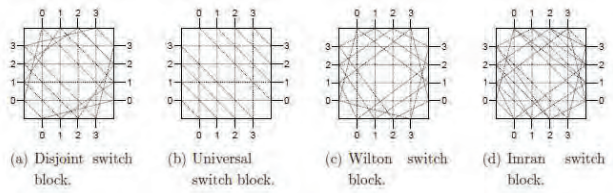


Figure 9: Switch topologies supported by VPR.

VPR describes wire segments by: the number of logic clusters spanned by the wire, the usage frequency of the segment in the FPGA, the resistance and capacitance per unit length, the switch type that connects the wire with other wires and the switch type that connects the wire and logic clusters.

D. Placement: VPR

The optimum algorithm for placement in VPR is basically an adaptive annealing scheme. This adaptive algorithm is called as Stimulated Annealing (SA), which is used to adapt the current placement at any instant of time. From the basic features of the circuit, the initial temperature is selected. Assuming that the total no. of functional block or logic clusters in the given design is denoted by n. The random initial placement is obtained, N cluster pair wise swaps are taken and initial temperature is evaluated as 20 times the standard deviation of the cost of the different n clusters. Moreover, the number of inner moves performed at each temperature is calculated as

$$MovesPerTe\ mparature = InnerNum \times N_{clusters}^{4/3}$$

Here innerNum is a constant and usually taken as 10. The way the temperature is updated is the feature of the adaptive SA algorithm used in VPR. Almost all moves are accepted at high temperatures and only improving moves are accepted at low temperatures in conventional Stimulated Annealing (SA). In adaptive Stimulated Annealing (SA) [13], the cooling technique tried to increase the time with respect to cost of the temperatures (low and medium temperatures) at the highly cost worsening temperatures using the following equation

$$T_{new} = \gamma \times T_{old}$$

Here γ is evaluated with respect to the % of moves accepted, as shown in table1.

Table 1: VPR temperature update schedule [13].

α	γ
$0.96 < \alpha$	0.5
$0.8 < \alpha \leq 0.96$	0.9
$0.15 < \alpha \leq 0.8$	0.95
$\alpha \leq 0.15$	0.8

DOI: 10.32377/cvrjst1004

E. Routing: VPR

The VPR CAD tool consists of two routing algorithms which are VPR routability driven router and timing driven router. The VPR routability is considered the path finding algorithm. The path finding algorithm repeatedly rips up and re routes all the nets of the circuit for every iteration. This process repeats until all congestions are completely eliminated. All available nets are initially routed to decrease the delay, but it produces congestion. Later routing iterations resolve the congestions by applying on overused routing blocks and resources. Then the cost is calculated for the given routing resource-n, which is connected to routing resource-m in the VPR and is given by

$$Cost(n) = h(n) \times b(n) \times p(n) + BendC(n, m)$$

Here h(n), b(n) and p(n) are historical congestion, base cost and present congestion respectively. h(n) is raised after every routing iteration in which n is overused. b(n) is set to the delay of n i.e. delay(n). p*(n)* is set to '1' if routing in which n is overused. p*(n)* is set to '0' if routing of n is not overused with congestion. The BendC (n, m) is used to detailed routability.

The timing driven router is based on path finding algorithm. In this case the timing information is considered for all routing iterations. The Elmore delay model equations are used to calculate timing information, delays and the cost of including a node n in a nets routing is given by

$$Cost(n) = Crit(i, j) \times del(n, topo \log y) + [1 - Crit(i, j)] \times h(n) \times b(n) \times p(n)$$

Here a connection criticality $Crit(i; j)$ is given by

$$Crit(i, j) = \max\left\{\left[MaxCrit - \frac{slack(i, j)}{D_{\max}}\right]^{\eta}, 0\right\}$$

Here MaxCrit is the parameter that controls how the connections slack impacts the congestion delay trade off in the cost function and Dmax is the critical path delay.

IV. CONCLUSIONS

This paper proposed several methodologies for leakage power reduction in modern nanometer FPGAs. The use of supply gating using Multi-Threshold CMOS (MTCMOS) techniques was proposed to enable turning OFF the unused resources of the FPGA, which are estimated to be close to 30% of the total FPGA area. Moreover, the utilized resources are allowed to enter a sleep mode dynamically during run-time depending on certain circuit conditions. To increase the benefits from the MTCMOS FPGA architecture, new packing techniques (AT-VPack, FAT-VPack, and T-MTCMOS) were proposed to include the logic blocks activities and pack those with similar activity profile together. The proposed techniques were applied to several FPGA benchmarks, and it was found out that the combination of R-LAP and T-MTCMOS for activity and packing algorithms respectively, yields the most leakage savings while incurring the minimum performance penalty. An average, the R-LAP and T-MTCMOS combination yields about 52% leakage savings, while the performance loss affecting the critical paths was kept at 3%.

REFERENCES

[1] J. Rose and S. Brown, "Flexibility of Interconnection Structures for Field Programmable Gate Arrays," *IEEE J. Solid-State Circuits*, vol. 26, no. 3, pp. 277-282, 1991.

[2] J. Cong and M. Smith, \A Parallel Bottom-Up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design," in *Proc. of IEEE/ACM Design Automation Conf.*, 1993, pp. 755-760

[3] A. Ling, D. P. Singh, and S. D. Brown, "Fpga technology mapping: A study of optimality," in *Proc. of IEEE/ACM Design Automation Conf.*, 2005, pp. 427-432.

[4] A. Marquardt, V. Betz, and J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," in *Proc. Of ACM Intl. Symp. on Field Programmable Gate Arrays*, 1999, pp. 37-46.

[5] J. Cong and S. K. Lim, "Edge Separability Based Circuit Clustering with Application to Circuit Partitioning," in *Proc. of IEEE/ACM Asia South Pacific Design Automation Conf.*, 2000, pp. 429-434.

[6] L. W. Hagen and A. B. Kahng, "Combining Problem Reduction and Adaptive Multi-Start: a New Technique for Superior Iterative Partitioning," *IEEETrans. Computer-Aided Design*, vol. 16, no. 7, pp. 709-717, July 1997.

[7] D. J.H. Huang and A. B. Kahng, "When Clusters Meet Partitions: New Density-Based Methods for Circuit Decomposition," in *Proc. of European Design and Test Conf.*, 1995, pp. 60-64.

[8] A. E. Dunlop and B. W. Kernighan, \A Procedure for Placement of Standard Cell VLSI Circuits," *IEEE Trans. Computer-Aided Design*, vol. 4, no. 1, pp. 92-98, 1985.

[9] D. J. H. Huang and A. B. Kahng, \Partitioning-Based Standard-Cell Global Placement with an Exact Objective," in *Proc. of ACM Intl. Symp. on Physical Design*, 1997, pp. 18-25.

[10] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," *IEEE Trans. Computer-Aided Design*, vol. 10, no. 3, pp. 356-365, Mar. 1991.

[11] A. Srinivasan, K. Chaudhary, and E. S. Kuh, Ritual: A Performance Driven Placement Algorithm for Small Cell ICs," in *Proc. of Intl. Conf. on Computer Aided Design*, 1991, pp. 48-51.

[12] A. Marquardt, V. Betz, and J. Rose, "Timing-Driven Placement for FPGAs," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2000, pp. 203-213.

[13] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep Submicron FPGAs*. Norwell, MA: Kluwer Academic Publishers, 1999.

[14] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE J. Solid-State Circuits*, vol. 20, no. 4, pp. 510-522, Apr. 1985.

[15] S. Kirpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing," *Science*, vol. 220, 4598, pp. 671-680, 1983.

[16] R. B. Hitchcock, “Timing Verification and the Timing Analysis Program," in *Proc. of IEEE/ACM Design Automation Conf.*, 1982, pp. 594-604.

[17] D. Lewis, V. Betz, D. Je®erson, A. Lee, C. Lane, P. Leventis, S. Marquardt, C. McClintock, B. Pedersen, G. Powell, S. Reddy, C. Wysocki, R. Cli, and J. Rose, “The StratixTM Routing and Logic Architecture," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2003, pp. 12-20.

 DOI: 10.32377/cvrjst1004