Delay Optimum And Area Optimal Mapping Of *k*-LUT Based FPGA Circuits

Ion I. Bucur *, Ioana Făgărăşan *, Cornel Popescu *, George Culea, Alexandru E. Şuşu *****

** University Politehnica Bucharest, Faculty of Control and Computers,*
E-mail ion.bucur@cs.pub.ro, ioana@shiva.pub.ro, cornel.popescu@cs.pub.ro
***University of Bacau, Faculty of Electrical Engineering,* E-mail: gculea@ub.ro
**** Swiss Federal Institute of Technology Lausanne (EPFL), IC/STI Department*,
Email: alex.susu@epfl.ch

Abstract: *The paper presents several improvements to our synthesis platform Xsynth that was developed targeting advanced logic synthesis and technological mapping for k-LUT based FPGAs. Having implemented an efficient exhaustive k-feasible cone generator it was targeted delay optimum mapping and optimal area. Implemented algorithm can use common unit-delay model and, the more general, the edge-delay model. The last model allows arbitrary delay values assignments to each branch of a circuit net. Such arbitrary delay values my reflect estimates of placement and routing delays. Powerful heuristics targeting minimal area (number of used LUTs in the mapped network) allow determinations of delay minimum solutions but having low used area.*

Keywords: k-LUT based FPGA, algorithms for delay optimal mapping, k-feasible cones.

1. INTRODUCTION

The Field Programmable Devices (FPDs) have been widely used for implementation of small to medium size digital circuits. There are two major types of FPDs - Field Programmable Gate Arrays (FPGAs) that usually consist of small programmable logic cells, such as *k*-input single-output lookup tables, and Complex Programmable Logic Devices (CPLDs) that are based on multiple-input and multiple-output PLA-like logic cells. Both of FPGAs and CPLDs have been widely used. The programmable gate array technology was introduced in the mid-1980s as a lesser-cost substitute for the implementation of application-specific integrated circuits (ASICs). Main difference between the mask-programmable gate array technology and the cell library technology for ASICs, and FPGA is that the last one does not need to go through the fabrication process for circuit implementation and is field programmable and often field reprogrammable (LUT based ones).

Although the FPGA in general has a lower gate density and slower circuit speed, its advantages of programmability, shorter design turnaround time, and lower initial nonrecurring engineering cost (good for low to medium volume production) often offset its disadvantages. FPGAs consist of three kinds of programmable elements: programmable logic blocks (PLBs), routing resources, and input–output (I/O) blocks. Each logic block contains combinational components such as multiplexers (MUXs), simple gates (e.g., OR and AND), programmable lookup tables (LUTs), and sequential components such as flip-flops. Routing resources include segmented interconnects and switching blocks. The segmented interconnects connect to the inputs and outputs of logic blocks while the switching blocks link the segments to form long routing tracks to implement routing topology. The I/O blocks can be programmed to become the primary inputs (PIs) or primary outputs (POs) of the circuits on FPGAs.

Most commonly used FPGAs are based on *k*-input single-output lookup tables (*k*-LUTs). A *k*-input LUT (*k*-LUT) consists of static random access memory (SRAM) cells that can store the truth table of an arbitrary *k*-input function. Every *k*-LUT can implement any function with no more than *k* inputs. In practice, *k* is usually small, for example, 4-LUTs are widely used in commercial FPGAs, as the area of a *k*-LUT grows exponentially with large *k*. Determining if an arbitrary, given wide, function can be implemented by a PLB, unfortunately, it is generally, a very difficult problem. This problem is called the *Boolean matching* problem. An ample survey of representative FPGA technology mapping approaches can be found in (Cong and Ding 1996). Basic reported algorithms and techniques used for mapping *k*-LUT based FPGAs circuits are summarized, compared and evaluated using only delays minimum criteria. We designed and developed one of these algorithms.

2. PROBLEM FORMULATION

A Boolean network *N* can be represented as a directed acyclic graph (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*. Primary input (PI) nodes have no incoming edge, and primary output (PO) nodes have no outgoing edge. It is used *input(v)* to denote the set of fanins of gate *v*. Given a subgraph *H* of the Boolean network, let *input(H)* denotes the set of *distinct* nodes outside which supply inputs to the gates in *H*.

For a node *v* in the network, a *k-feasible cone at v*, denoted C_v, is a subgraph consisting of and its predecessors (*u* is a predecessor of *v* if there is a directed path from *u* to *v*), such that $|input(C_v)| \le k$ and any path connecting a node in C_v and *v* lies entirely in C_v.

The *level* of a node is the length of the longest path from any PI node to *v*. The level of a PI node is zero. The *depth* of a network is the largest node level in the network.

A Boolean network is *k-bounded* if $|input(v)| \le k$ for each node in the network. In this paper, the primary considered objective is to minimize the circuit mapping delay under the LUT-delay model through technology mapping.

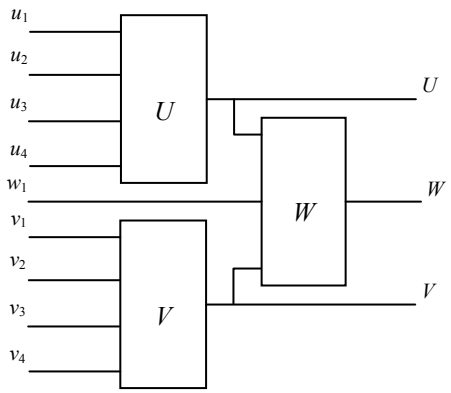

Fig. 1. Schematic Block Diagram of Xilinx 4000 CLB.

Therefore, a mapping solution is said to be *optimum* if the mapping delay is minimized. The secondary objective is to reduce the area used in the technology mapping solution.

3. SPECIFIC LOGIC OPTIMIZATION

A logic block (XC4000 CLB) contains three LUT's noted *U*, *V* and *W* as shown in figure 1. Four independent inputs (u_1, u_2, u_3, and u_4) and (v_1, v_2, v_3, and v_4) are provided for each of two 4-input LUT's *U* and *V*.

The LUT *W* has internal inputs from *U*, *V* and a third input from outside the block (w_1). With this design, a XC4000 CLB (as an example) can be used to implement

(i) Any two functions of up to four variables, or
(ii) Any single function of five variables, or
(iii) Any function of four variables together with some function of five variables, or
(iv) Some function of up to nine variables.

The flexibility provided by XC4000 CLB, however, also creates a great deal of difficulty to have efficient use of all resources. To implement a Boolean network on XC4000 FPGA's, the common approaches is to first map the network into a k-LUT network and then pack the *k*-LUT network into XC4000 CLB's. Parameter *k* can be set to either 4,5 or 6. If one set $k = 4$, then *U* and *V* LUT's in a XC4000 CLB can implement every two 4-LUT's in the mapping solution, but the *W* LUT is left unused.

From a functional decomposition point of view, a XC4000 CLB can implement any function of the form $g(y_1(X_1), y_2(X_2), x_n)$, where:

$X = \{x_1, x_2, \ldots x_n\}$, $n \le 9$, $X_1, X_2 \subset X$, $|X_1| \le 4$, and $|X_2| \le 4$.

Given function $f(X)$, it can be implemented by a XC4000 CLB, if $f(X)$ can be transformed into the XC4000-CLB form. For example, when $n = 5$, the Shannon expansion:

$$f(X) = x_5 f(x_5 = 1) + \bar{x}_5 f(x_5 = 0) \quad (1)$$

Transforms $f(X)$ into the XC4000-CLB form, where $f(x_5 = 1)$ and $f(x_5 = 0)$ correspond to y_1 and y_2 respectively and $g(y_1, y_2, x_5)$ has the functionality of a 2-input multiplexer. There are several criteria to distinct basic design techniques focusing on *k*-LUT FPGA based circuits. One could recognize a first approach named logic optimization, which transforms the gate level network into another equivalent gate-level network more suitable for the subsequent step.

Other approach is technology mapping, which transforms the gate-level network into a network of cells targeting specific technology by covering initial network with the specific cells.

This classification is used, in general, in logic design domain as alternatives to the same two main low-level design approaches. First approach is making technology independent optimization based on given network particularities, and is generally known as *logic optimization*.

Logic optimization operates using mainly knowledge of gates and network functionality and tools are Boolean optimization techniques. Given a multilevel network of logic gates, combinational logic synthesis transforms it into a network of look-up tables, each of no more than *k* inputs, where parameter *k* is determined by the FPGA technology.

The second approach, named *technology mapping*, ignores any independent optimization, following only structural information of the network, specific technology optimization, and uses only combinatorial optimization techniques. It's essential to outline that these approaches are using different techniques having distinct characteristics.

However many reported synthesis algorithms and systems are using both approaches. In many synthesis systems dedicated to *k*-LUT based FPGAs circuits, a separate mapping or coverage step doesn't exit because a gate-level network *k*-bounded gate input can be considered an a *k*-LUT network.

4. DELAY OPTIMUM AND AREA OPTIMAL

There are several optimization targets for *k*-LUT Based FPGAs. Among them the most used are:

- The delay minimization objective is to minimize the delay from primary inputs to primary output in FPGA implementation.
- The area minimization objective is to use the minimum chip area in order to implement the circuit.

- The routability objective is to make the *k*-LUT network more routable in the subsequent placement and routing steps.
- The power minimization objective is to minimize the power dissipation of the implementation.

Accurate measurement of these objectives requires information that's available only after layout synthesis. That's the reason most of the reported specific applications are using either estimation using function cost models or quick layout synthesis.

Delay of a *k*-LUT network is measured by the length of the longest path from primary input to a primary output, where the length is the computed by the accumulation of delays of nodes and edges along the path. For a given technology the *k*-LUT delay is approximately a constant.

Various delay models focus mostly on the internal various connections delays (Xilinx XC4000 interconnect is presented, as an example, in Fig. 2).

The most commonly used model is the unit delay model, assuming each net has constant delay. This assumption involves that delay minimization is equivalent to depth minimization. Area of a *k*-LUT network is usually estimated by the number of *k*-LUTs, or any actual logic elements used in a specific FPGA architecture.

Interconnect calculation is the process of estimate the routing resource necessity and/or consumption, before actually realizing the routing process.

For the FPGA design flows, the term routability evaluation is more suitable, as the routing resources are fixed for a specified device. A certain a particular FPGA device and a circuit to be mapped onto the device, routability evaluation is the process of identifying the quantity of routing elements needed to perform an entire routing. If the device has sufficient routing elements to satisfy the requirement, then the circuit is routable.

The evaluation process attempts to create routing claim values on every programmable routing item on the device. Useful parameters derived from the evaluation process are the *peak routing claim* and the *routing claim distribution*.

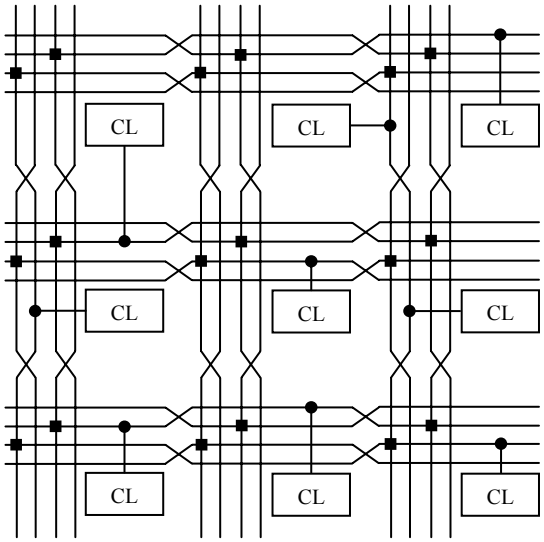

Fig.2. XC4000 Interconnect.

For a successful routing, the device must satisfy both the peak routing demand and the routing demand distribution necessities.

The problem of interconnect anticipate has been studied extensively for the ASIC design flows (Kannan *et al.* 2002).

Routability is usually modeled by interconnection complexity, more specific, the size and the terminal distribution of the net. Simplified estimations, currently used, are the pin-to-net ratio and the pin-to-cell ratio.

A fine routability evaluation method, in general, should be:

Accurate - Definitely, the evaluation should be insightful of actual information obtained from a complete router. Either the numbers should directly show a relationship with the complete routing results or at least the ratios should go with over a large set of benchmarks.

Fast - Routability evaluation is typically used within other physical design tools and hence rapidity is vital.

Generic - The process should be free of the FPGA device structural design.

Usable - The evaluation should generate usable outcome for a broad range of applications.

Some of the routability evaluation methods that satisfy these necessities and currently under wide practice are *Rent*'s rule derivatives (Parthasarathy *et al.* 2001), (Yang *et al.* 2002), *fGREP* (Kannan *et al.* 2002), *Lou*'s method (Lou *et al.* 2001), and *RISA* (Chang 1994). *fGREP* is a theoretical technique that uses the notion of routing elasticity to represent routability.

Lou's technique is based on the proportion of the number of paths that use a explicit routing region to the total amount of paths possible. *RISA* is a heuristically developed method based on the wiring allocation map, resulting from a large number of arbitrarily generated optimal Steiner trees. All the methods except *fGREP* were initially projected for ASIC design flows. However, they are generic enough in their formulation that a direct conversion to an FPGA design flow is trivially possible.

Power minimization is of relative recent interest in logic synthesis (Sutter and Boemo 1997). Power consumption can be estimated based on the output load capacitance and transition frequency of *k*-LUT and primary inputs (Mashayekhi *et al.* 2008). Load capacitance changes dynamically with the mapping process.

Often, optimization targets may be mutually less-suitable. Minimization the number of *k*-LUTs may lead to a larger delay. It involves necessity of finding a proper balance among different objectives.

5. PREVIOUS RELEVANT WORKS

A number of technology mapping algorithms have been proposed which optimize for area and performance in lookup table-based FPGA designs. Some of these efforts are described below. The *Chortle* program, is one of the earliest mapping algorithms (Francis 1990). It does not exploit the

relationship among nodes across the trees.

Chortle-crf (Francis 1991b) *Chortle*'s successor employs bin-packing heuristics for gate-level decomposition and technology mapping. As a result, it reduces the area by 14% when compared with the original Chortle algorithm. *Chortle-crf* is optimal for $K \leq 5$. The idea in *Chortle-crf* was then extended to depth (delay) minimization in the *Chortle-d* algorithm (Francis 1991a). In addition, it minimizes area as a secondary objective.

Mapping solutions of *Chortle-d* use an average of 35% fewer levels of LUTs than those of *Chortle-crf*, at the cost of an average of 59% larger area. Chortle algorithms have solved the optimal mapping problem for an un-bounded tree, but a prior tree partitioning often could compromises the mapping quality.

Another early algorithm was the original *MIS-pga* (Brayton *et al.* 1987). It was an extension of the UC Berkeley *MIS-II* logic synthesis system. This system evolved finally in SIS-1.2 (Sentovich *et al.* 1992). It is applicable to general networks for *area minimization*. The algorithm has two close related steps, logic optimization and technology mapping. First step uses partial implemented Roth-Karp functional decomposition (Roth and Karp 1962), kernel extraction (multilevel optimization) and AND-OR decomposition to make network transformation into a *k-bounded* one (Sentovich *et al.* 1992).

Last part of the first step proceeds by collapsing nodes into their fan-outs, while maintaining it *k-bounded*. Nodes are collapsed is a heuristically order. Technological mapping step uses node covering based enumeration.

This synthesis application had an improvement named *MIS-pga*(*new*) (Murgai *et al.* 1991a) where first step was substantially enhanced with three other decomposition techniques: *cube partitioning*, *cofactoring* and *cube-packing*. All decomposition methods are tried and best result is kept.

This enhancements and many other did lead to area reduction by 28.2% compared to the older version. Other version of this application was *MIS-pga*(*delay*) (Murgai *et al.* 1991b) targeting *delay minimization*. Best solution for technological mapping is computed using heuristic binate covering. *MIS-pga* family is known in literature as *the synthesis-based mapping* (Murgai *et al.*1991b).

Notable progress in *k-LUT* based FPGA synthesis was development of delay-optimal technology mapping algorithms based on *DAGMap* algorithm (Cong, J., and Y. Ding 1992).

This was the first polynomial-time depth-optimal mapping algorithm for general *k-bounded* circuits. *DAGMap* transforms a general network into a depth-minimum 2-*bounded* simple gates network using AND-OR and Huffman tree decomposition.

Mapping section is performed using *dag-map* algorithm. Obtained solution is improved using two post-processing operations. This algorithm evolved in *FlowMap* algorithm (Cong and Ding 1994). It uses same logic optimization as *DAGMap* but has an improved procedure for technological mapping, named *FlowMap*, and post processing uses one new step named *flowpack*.

FlowMap is computing one minimum height *k-bounded* cone rooted in each internal node of the network. This feature guarantees depth-optimal mapping for general *k-bounded* networks.

However, if there are more than one such *k-bounded* cone rooted in each internal node of the network, it is ignored, narrowing solutions space.

The used *MinCut-MaxFlow* procedure implies this feature. Mapping results proved is superior to *Chortle-d* and *MIS-pga*(*delay*) because these algorithms use 9 – 50% more LUTs with up to 7% larger depth on average compared to *FlowMap*. A different approach is used to attempt the area-minimal mapping in the algorithm *Praetor* (Cong *et al* 1999).

6. IMPLEMENTED ALGORITHM

One of recent introduced algorithm is *minDepth* having an advanced clustering system (Bucur 2007).

Table 1. *minDepth –minLevel* Experimental results.

	FlowMap		*minLevelMapII*	
Circuit	Delay	Area	Delay	Area
5xp1	3	24	2	19
9symml	5	61	3	7
C499	5	154	4	60
C880	8	232	7	117
alu2	8	162	5	118
alu4	10	268	5	236
apex6	4	257	4	198
apex7	4	89	4	62
count	3	76	3	74
des	5	1308	5	1001
duke2	4	187	4	130
misex1	2	15	2	17
rd84	4	83	3	14
rot	6	268	6	208
vg2	4	45	3	35
z4ml	3	13	2	5

It proceeds using algorithm exposed in (Bucur 1999) used also in *DAOmap* (Chen and Cong 2004).

Logical optimization is done with typical procedures implemented with structure and routine in SIS-1.2 (Sentovich *et al.* 1992).

The algorithm *minDepth* is designed to compute *all k-bounded cones rooted in each internal node of the network.* This feature provides more freedom degrees in building-up technological mapping solutions.

It was proved that this algorithm computes all possible mapping solution for each node. Details of the implemented algorithm are exposed in (Bucur 1999).

However heuristics used in order to obtain optimal area solutions were recently notable enhanced and entire algorithm evolved to the improved one, named *minLevelMapII*.

Our mapping algorithm is structural. This algorithm is included in *XSYNTH* tool. *XSYNTH* was developed targeting *k*-LUT based FPGA logic synthesis, featuring almost every facility of logical optimization including powerful kernel optimization.

Computation of all *k-feasible cones* rooted in each node *u* of the DAG associated to the optimized network is typically a run-time and memory bottleneck of a structural mapper (Mishchenko *et al.* 2007). There are several published attempts to compute efficiently all *k-feasible cones* (Manohararajah *et al.* 2006) and our procedure is one of the most efficient (Bucur 1999).

7. EXPERIMENTAL RESULTS

Measurements, made using *minLevelMapII* on MCNC91 benchmark circuits, are presented in Table 1. Most of these well-known benchmark circuits are currently used in order to evaluate technological mapping algorithms.

This algorithm was optimized for area but preserving previously obtained optimum depth results. Methods it were used previously to compute area (Bucur 2007) and improved implementing several published heuristics (Manohararajah *et al.* 2006), in fact, cost functions (Mishchenko *et al.* 2007). Actual area optimization implemented in *minLevelMapII* is using a look-ahead area estimator based on dynamic programming. Following the initial description (Bucur 2006), subsequent work has employed the heuristics within an integrated mapping application able to produce networks basically implemented with *k*-LUT FPGAs for delay, area power and placement.

Mapping process is carried-out in network traversing from primary outputs to primary inputs. At each node the algorithm is making a choice among several *k-feasible cones*, all of them having the minimum depth (involving minimum delay). The choice is made such the global area is minimal (optimal area).

Actual mapping results are using 14% less LUTs, on average, compared to *minLevelMap* (Bucur 2007), but preserving same depth level (Bucur 1999). Results of *minLevelMapII* algorithm are compared against previous published homologues results of *FlowMap* algorithm.

Mapping results proved that *minLevelMapII* is superior to *FlowMap*, because circuits mapped with *FlowMap* algorithm uses 28% more LUTs with up to 25.8% larger depth on average, compared to *minLevelMapII* algorithm.

8. CONCLUSIONS AND FUTURE WORK

Internal potential of *MinDepth* algorithm makes it suitable for technological mapping refinements. In fact our generator of all *k-feasible cones* at each node potentially helps us to build-up every possible solution.

It was tested flexible delay mapping with optimal area targeting different delays (greater than *minimum* possible but with *less* LUT count) for each primary output.

Such solution is useful in practical environment because designer could tailor more appropriate solutions involving not-necessary best delay (seldom used in practice) but an optimum equilibrium between delay, on one side, and LUT count (area), on other side, in order to satisfy particular design features.

New version of this algorithm, named *minLevelMapIII* (Bucur *et al.* 2008) was released for technological mapping using clusters partitions of circuits in order to achieve best solutions using signal flow in networks.

Such an approach will make this algorithm more cooperative with power consumption estimations, and the balanced usage of connecting and I/O resources in mapping FPGAs circuits. Future developments intend to approach power-aware minimal depth and minimal area.

REFERENCES

Brayton, R.K., R. Rudell, A.L. Sangiovanni-Vincentelli, and A.R. Wang (1987). MIS: A multiple-level logic optimization system. In: *IEEE TCAD*, **Vol. 6***,* No. 6, pp. 1062-1081.

Bucur, I.(1999). An Optimal Technology Mapping for Delay Optimization of Lookup Table-Based FPGAs. In: *Proc. CSCS'99,* pp. 127-132.

Bucur, I. (2007). Performance Mapping of *k*-LUT Based FPGAs. In: *Univ. Politehnica of Bucharest, Scient. Bull., Series C: Electrical Engineering,* **Vol. 69**, No.2, pp 49-60.

Bucur, I., I. Fagarasan, C. Popescu, C.-A. Boiangiu, and G. Culea (2008). On K-LUT Based FPGA Optimum Delay and Optimal Area Mapping. In: *Proc. WSEAS Intl. Conf. on Math. and Comput. Methods in Science and Engineering,* pp.137-142.

Chang, C.-L.E.(1994). RISA: Accurate and efficient Placement Routability Modeling. In: *Proc. ICCAD'94*, pp. 690-695.

Chen, D. and J. Cong (2004). DAOmap : A depth-optimal area minimization mapping algorithm. In: *Proc. ICCAD'04*, pp.752-757.

Cong, J., and Y. Ding (1992). An optimal technology-mapping algorithm for delay optimization in lookup-table based FPGA designs. In: *Proc. ICCAD'92*, pp. 48-53.

Cong, J., and Y. Ding (1994). FlowMap: An Optimal Technology mapping algorithm for delay optimization in lookup-table based FPGA designs. In: *IEEE TCAD,* **Vol. 13**, No. 1, pp. 1-12, 1994.

Cong, J., and Y. Ding (1996). Combinational Logic Synthesis for LUT Based Field Programmable Gate Arrays. In: *ACM TODAES*, **Vol. 1**, No. 2, pp. 145-204.

Cong J., Wu C., and Y. Ding (1999). Cut Ranking and Pruning: Enabling A General And Efficient FPGA Mapping Solution. In: *Proc. Int. Symp. on FPGAs'99*, pp. 29-35.

Francis, R.J., J. Rose, and K. Chung (1990). Chortle: A technology mapping program for lookup table-based field gate arrays. In: *Proc. DAC'90,* pp. 613-619.

Francis, R.J., J. Rose, and Z.G.Vranesic (1991a). Chortle-crf: Fast technology mapping for lookup table-based FPGAs. In: *Proc. DAC'91,* pp. 227-233.

Francis, R.J., J. Rose, and Z.G. Vranesic (1991b). Technology mapping of lookup table-based FPGAs for performance. In: *Proc. CAD'91,* pp. 568-571, 1991.

Kannan, P., S. Balachandran, and D. Bhatia, D (2002). On metrics for comparing routability estimation methods for FPGAs. In: *Proc. DAC'02*, pp. 70- 75.

Kaviani, A., and S. Brown (1999). The Hybrid Field-Programmable Architecture. In: *IEEE Design and Test of Computers*, **Vol. 16**, No. 2, pp. 74–83, 1999.

Lou, J., S. Krishnamoorthy, S., and H. Sheng (2001). Estimating Routing Congestion using Probabilistic Analysis. In: *Proc. Int'l Symp. Physical Design*.

Manohararajah, V., S.D. Brown, and Z.G. Vranesic (2006). Heuristics for Area Minimization in LUT-Based FPGA Technology Mapping. In: *IEEE TCAD*, **Vol. 25**, No.11, pp 2331-2340.

Mashayekhi, M., Z. Jeddi, and Z. Amini (2008). Power Optimization of LUT Based FPGA Circuits. In: *Proc OPTIM'08*, pp 37-40.

Mishchenko, A., S. Chaterjee, and R.K. Brayton (2007). Improvements to Technology Mapping for LUT-Based FPGAs. In: *IEEE TCAD*, **Vol.26**, No2, pp240253.

Murgai, R., N. Shenoy, R.K. Brayton, and A.L. Sangiovanni-Vincentelli (1991a). Improved logic synthesis algorithms for table look up architectures. In: *Proc. ICCAD'91*, pp. 564-567.

Murgai, R., N. Shenoy, R.K. Brayton, and A.L. Sangiovanni-Vincentelli (1991b). Performance directed synthesis or table look up programmable gate arrays. In: *Proc. ICCAD'91*, pp. 572-575.

Parthasarathy, G., M. Marek-Sadaowska, A. Mukherjee, and A. Singh (2001). Interconnect Complexity-aware FPGA Placement Using Rent's Rule. In *Proc. Int. Workshop on System Level Interconnect Prediction*.

Roth, J.P. and R.M. Karp (1962). Minimization over Boolean graphs. In: *IBM J. Res. Dev.*, pp. 227-238.

Sentovich, E., K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephen, R.K. Brayton, and A.L. Sangiovanni-Vincentelli (1992). SIS: A System for Sequential Circuit Synthesis. In: *UC Berkeley, TR. UCB/ERL M92/41*.

Sutter, G., E. Boemo (2007). Experiments in Low Power FPGA Design. In: on-line http://www.scielo.org/pdf/laar/v37n1a18.pdf.

Xilinx (1997). The Programmable Logic Data Book.

Yang, X, R. Kastner, and M. Sarrafzadeh (2002). Congestion Estimation During Top-down Placement. In: *IEEE TCAD*, **Vol.21**, pp. 72-80.