OPTIMAL AREA AND PERFORMANCE MAPPING OF *K*-LUT BASED FPGAS

Ion I. BUCUR, University Politehnica of Bucharest
Cornel POPESCU, *University Politehnica of Bucharest*
George CULEA, *University of Bacău, Faculty of Electrical Engineering,*
Alexandru E. ŞUŞU, Swiss Federal Institute of Technology Lausanne

FPGA circuits are increasingly used in many fields: for rapid prototyping of new products (including fast ASIC implementation), for logic emulation, for producing a small number of a device, or if a device should be reconfigurable in use (reconfigurable computing). Determining if an arbitrary, given wide, function can be implemented by a programmable logic block, unfortunately, it is generally, a very difficult problem. This problem is called the Boolean matching problem. This paper introduces a new implemented algorithm able to map, both for area and performance, combinational networks using k*-LUT based FPGAs.*
Keywords: *k*-LUT based FPGAs, combinational circuits, performance-driven mapping.

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 [1], [2], [3], and [4].
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.
FPGA circuits are increasingly used in many fields: for rapid prototyping of new products (including fast ASIC implementation), for logic emulation, for producing a small number of a device, or if a device should be reconfigurable in use (reconfigurable computing).
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 [5], [6], [7], and [8].
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). LUTs are the basic logic blocks in many FPGAs today. 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 [5], [9], and [10].

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*. It was showed [30], and [24] that 4-input, single-output LUT cell yields the smallest FPGA area of any *k*-LUT cell for a wide range of programming technologies and routing pitches.
Most of actually commercially available FPGAs are using, indeed, LUTs of input size of 4 or 5 ([10], [11], and [12]). In many FPGAs, small LUTs are connected by fast local connections to form a programmable logic block (PLB) for implementation flexibility, better performance, and proficient utilization of silicon area. A PLB can often implement one arbitrary *k*-input function, where is determined by the PLB architecture or some *large* function of more than *k* inputs [13], [14], [15], [16], [17], and [18].
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.
Most of existing technology mapping algorithms first produce a *k*-LUT mapping solution, and then pack LUTs into PLBs [9], [10], [11], [19], and [20]. An ample survey of representative FPGA technology mapping approaches can be found in [10] and [26].
While placement and routing is strongly connected with the detailed architecture inside of the chip and mostly managed by the commercial FPGA software, the optimization and mapping can be more influenced by the user [21], [22], [23], [24], and [25].
Recent innovations in field-programmable gate array (FPGA) architecture has led to the development of *heterogeneous FPGA* families [23] that combine diverse sets of logic resources on the same silicon substrate, having heterogeneous blocks, similar to Apex20KE [3].
To support wide fan-in, low logic density sub-circuits, such as finite state machines, some contemporary FPGA architectures [20] contain SRAM-configurable PLAs. Unlike fine-grained LUTs, PLAs can implement sets of logic functions with minimal interconnect, the most area expensive resource in contemporary FPGAs. For *m* terms based PLA structures, this area efficiency often comes at the cost of increased minimum delay for PLA paths versus corresponding LUT paths, requiring resource balance. When coupled with fine-grained LUTs, PLAs provide an integrated programmable resource that can be used in many digital system designs to support non-critical-path control logic for LUT-based data paths [27], [28], [29], [30], [31], [32] and [33].
In order to maximize performance and device utilization, recent generations of FPGAs take advantage of speed and density benefits resulting from *heterogeneous* FPGAs, which provide either an array of homogeneous PLBs, each configured to implement circuits with LUTs of different sizes, or an array of physically heterogeneous LUTs.
The PLBs in FPGAs made by Xilinx, XC4000 series [34], Lucent, ORCA2C series [25], and the recently announced Vantis, VF1 [2], can be configured to have heterogeneous LUTs, as examples of versatile heterogeneous FPGAs. These heterogeneous FPGAs do not have limitations on the availability of LUTs of specific configurable sizes (as long as within the chip capacity) due to their PLB configuration flexibility.
The field programmable gate-array appeared last decade of previous millennium as alternative to application-specific integrated circuit (ASIC) projects. Static random-access memory (SRAM) is dominant FPGA technology in which programmability is realized by using SRAM cells to implement both programmable logic blocks and control programmable routing resources.
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. Model used

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 $|input(C_v)| \leq k$ 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 [33]. A Boolean network is *k-bounded* if $|input(v)| \leq k$ 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. Therefore, a mapping solution is said to be *optimal* if the mapping delay is minimized. The secondary objective is to reduce the area used in the technology mapping solution.

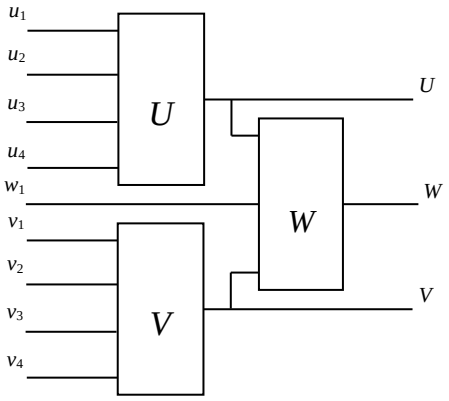

Fig. 1. Schematic Block Diagram of Xilinx 4000 CLB.

3. 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 [34]. 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 or 5. 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:
$X = \{x_1, x_2, \ldots x_n\}$, $n \leq 9$, $X_1, X_2 \subset X$, $|X_1| \leq 4$, and $|X_2| \leq 4$.
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:

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

It 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. Optimization targets and optimality

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. 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.
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.

Power consumption minimization is of relative new interest in logic synthesis. Power consumption can be estimated based on the output load capacitance and transition frequency of *k*-LUT and primary inputs. Load capacitance changes dynamically with the mapping process.
Optimization objectives 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. Related work

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, one of the earliest mapping algorithms [16] divides the Boolean network into a forest of trees and determines an optimal mapping of each tree using the dynamic programming techniques.
It does not exploit the relationship among nodes across the trees. *Chortle-crf* [18] *chortle*'s successor employs bin-packing heuristics for gate-level decomposition and technology mapping, achieving significant improvement over the previous algorithm in the solution quality as well as in run-time efficiency.
As a result, it reduces the area by 14% when compared with the original Chortle algorithm. *Chortle-crf* is optimal for $K \leq 5$. It also exploits reconvergent paths and replication at multi-fan out nodes, but if there are too many reconvergent paths terminating at a node then it tends to lose its run-time efficiency.
The idea in *Chortle-crf* was then extended to depth (delay) minimization in the *Chortle-d* algorithm [17]. In addition, it minimizes area as a secondary objective by using area-optimal node decomposition along critical path, as well as predecessor packing. 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. Both *Chortle-crf* and *Chortle-d* have very efficient implementations. 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* [27] was an extension of the UC Berkeley *MIS-II* logic synthesis system [6]. This system evolved finally in SIS-1.2. It is applicable to general networks for *area minimization*. The algorithm has two close related steps, logic optimization and technology mapping. It's the first implemented algorithm pointing out the importance of the logical optimization of the network before technological mapping.
First step uses partial implemented Roth-Karp decomposition [31] kernel extraction (multilevel optimization) and AND-OR decomposition to make network transformation into a *k-bounded* one. 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)* [29] 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 old version.
Other version of this application was *MIS-pga*(*delay*) [28] targeting *delay minimization*. Algorithm, proceeds by collapsing each critical node, in the network, into its critical fan outs. Such collapsing proved successful, *i.e.* keeping network *k-bounded* or nodes not satisfying this condition are re-decomposed according to a cube-packing based quick estimation procedure, will not increase delay in circuit and will reduce cost area. This procedure is re-iterated until no more collapsing is possible. Best solution for technological mapping is computed using heuristic binate covering.
Notable progress in *k-LUT* based FPGA synthesis was development of delay-optimal technology mapping algorithms based on *DAGMap* algorithm [8].

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 [9]. 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*.

6. Implemented solution

We recently introduced algorithm named *minDepth* [5]. It proceeds using similar approach as *FlowMap* and *MIS-pga*(*delay*). Logical optimization is done with typical procedures implemented with structure and routine in *SIS*-1.2 [32].
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.
Generation of all *k*-feasible cones rooted in every node of a node in network has to be considered in the context of network model.
Let N be a *k*-bounded network, and u a node of N. Then, a *k*-feasible cone of the node u, noted $C(u)$ could be identified by the set:

$$input(C(u)) = \{v_1, v_2, \ldots, v_m\},\ m \le k. \tag{2}$$

Such a set can be represented as the product (conjunction) of the elements (literals) of the set $p = v_1 v_2 \ldots v_m$. The set of all feasible cones of node u, noted $cones(u)$, can be represented as the sum (reunion) of each of the product (cube) representing the respective cone:

$$cones(u) = \bigcup_i input(\mathrm{C}_i(u)) \tag{3}$$

Representing each *k*-feasible cone of the node u as a conjunction, in above relation, it becomes:

$$cones(u) = \bigcup_i v_{i1} v_{i2} \cdots v_{im} \tag{4}$$

Then it holds:

Lemma1. Given a node u having as immediate predecessors $input(u) = \{v, w, \ldots, z\}$, each predecessor having already computed the set of all *k*-feasible cones, respective $cones(v)$, cones(w), …, $cones(z)$, than the set $cones(u)$, of all the *k*-feasible cones of node u, is included in the set:

$$\{ v \cup (cones(v)) \cap (w \cup \text{cones}(w)) \cap \ldots \cap (z \cup cones(z))\} \tag{5}$$

Proof the Lemma1 is exposed in [4].It was proved that this algorithm computes all possible mapping solution for each node (all details of the implemented algorithm are exposed in [4]).
Computing the sum-of-products (SOP) form of the expression (5), and eliminating (as soon as possible) all the products having more than k literals, one can determine $cones(u)$, the set of all *k*-feasible cones of the node u.
It is not difficult to remark that there is only polynomial number of *k*-feasible cones in the predecessor's maximum transitive cone of the node u, since the total number of possible combinations of k or fewer nodes is $O(n^k)$, where n is the number of nodes in the predecessor's maximum transitive cone of the node

u.

7. Results

Measurements, made using *minDepth* 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.
Results of *minDepth* algorithm are compared with published homologues results of *FlowMap* algorithm. Mapping results proved it is superior to *FlowMap*, because this algorithm uses 19% more LUTs with up to 25.8% larger depth on average, compared to *minDepth* algorithm.

Table 1

Experimental results.

	FlowMap		*minDepth*	
Circuit	Delay	Area	Delay	Area
5xp1	3	24	2	19
9symml	5	61	3	7
C499	5	154	4	65
C5315	-	-	8	498
C880	8	232	7	126
alu2	8	162	5	120
alu4	10	268	5	546
apex6	4	257	4	221
apex7	4	89	4	67
count	3	76	3	74
des	5	1308	5	1010
duke2	4	187	4	151
misex1	2	15	2	21
rd84	4	83	3	14
rot	6	268	6	209
vg2	4	45	3	35
z4ml	3	13	2	5

8. Conclusions and future work

Internal potential of *minDepth* algorithm makes it suitable for technological mapping refinements. 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.
Latest version of this algorithm, named *minLevelMapIII* 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 compliant with balanced usage of connecting and I/O resources in mapping FPGAs circuits.

R E F E R E N C E S

1. Altera Corp., Programmable Logic Devices Data Book, 1998.
2. AMD, Vantis VF1 FPGA Data Sheet, 1998.

3. Apex20KE Data Sheet, Altera Corporation, San Jose, CA, 1999.
4. *I. Bucur*, "Performance Mapping of *k*-LUT based FPGAs", in Scientific Bulletin of University Politehnica of Bucharest, Series C: Electrical Engineering,Vol 69,Number 2, 2007, pp. 49-60.
5. *I. Bucur, Al. Susu*, "Mapping Large Combinational Circuits with *k*-LUT Based PFGAs Using Dominating Cones", in Control Engineering and Applied Informatics,Vol. 8, Number 3, 2006, pp. 364-368.
6. *R.K. Brayton, R. Rudell, A.L. Sangiovanni-Vincentelli, and A.R. Wang*, "MIS: A multiple-level logic optimization system", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **Vol. 6**, No. 6, 1987, pp. 1062-1081.
7. *G. Chen, and J. Cong*, "Simultaneous logic decomposition with technology mapping in FPGA designs", in the Proceedings of the 2001 ACM/SIGDA ninth International Symposium on Field Programmable Gate Arrays, 2001, pp. 48-55.
8. *J. Cong, and Y. Ding*, "An optimal technology-mapping algorithm for delay optimization in lookup-table based FPGA designs", in *Proceedings of the IEEE International Conference on Computer-Aided Design*, 1992, pp. 48-53.
9. *J. Cong, and Y. Ding*, "FlowMap: An Optimal Technology mapping algorithm for delay optimization in lookup-table based FPGA designs", in IEEE Transactions on Computer-Aided Design and Integrated Circuits and Systems, **Vol. 13**, No. 1, January 1994, pp. 1-12.
10. *J. Cong, and Y. Ding*, "Combinational Logic Synthesis for LUT Based Field Programmable Gate Arrays (Tutorial and Survey Paper)", in ACM Transactions on Design Automation of Electronic Systems, **Vol. 1**, No. 2, April 1996, pp. 145-204.
11. *J. Cong, and Y-Y. Hwang*, "Boolean Matching for LUT-Based Logic Blocks With Applications to Architecture Evaluation and Technology Mapping", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **Vol. 20**, No. 9, 2001, pp. 1077-1090.
12. *J. Cong, and S. Xu*, "Performance-Driven Technology Mapping for Heterogeneous FPGAs", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **Vol. 19**, No. 11, November 2000, pp. 1268-1281.
13. *J. Cong, and C. Wu*, "An Efficient Algorithm for Performance-Optimal FPGA Technology Mapping with Retiming", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **Vol. 17**, No. 9, September 1998, pp. 738-748.
14. *E. Detjens, G. Gannot, R. Rudell, A. Sangiovanni-Vincentelli, and A.R. Wang*, "Technology Mapping in MIS", in Proceedings of the IEEE International Conference on Computer-Aided Design, 1987, pp. 116-119.
15. *R.J. Francis*, "A tutorial on logic synthesis for lookup-table based FPGAs" in Proceedings of the IEEE International Conference on Computer-Aided Design, 1992, pp. 40-47.
16. *R.J. Francis, J. Rose, and K. Chung*, "Chortle: A technology mapping program for lookup table-based field gate arrays", in *Proceedings of the ACM/IEEE Design Automation Conference*, 1990, pp. 613-619.
17. *R.J. Francis, J. Rose, and Z.G. Vranesic*, "Technology mapping of lookup table-based FPGAs for performance", in Proceedings of the IEEE International Conference on *Computer-Aided Design*, November 1991, pp. 568-571.
18. *R.J. Francis, J. Rose, and Z.G. Vranesic*, "Chortle-crf: Fast technology mapping for lookup table-based FPGAs", in Proceedings of the ACM/IEEE Design Automation Conference, June 1991, pp. 227-233.

19. *J. He, and J. Rose*, "Advantages of Heterogeneous Logic Block Architectures for FPGAs," IEEE Custom Integrated Circuits Conf. 1993, (CICC 93), May 1993 pp. 7.4.1 - 7.4.5.
20. *F. Heile, and A. Leaver*, "Hybrid Product Term and LUT Based Architectures Using Embedded Memory Blocks", in Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field Programmable Gate Arrays, February 1999, pp. 13–16.
21. *U. Hinsberger, and R. Kolla*, "Boolean Matching for Large Libraries", in Design Automation Conference, 1998, pp. 206-211.
22. *A. Kaviani, and S. Brown*, "The Hybrid Field-Programmable Architecture", in IEEE Design and Test of Computers, **Vol. 16**, No. 2, April 1999, pp. 74–83.
23. *A. Kaviani, and S. Brown*, "Technology Mapping Issues for an FPGA with Lookup Tables and PLA-like Blocks", in *Eighth International ACM Symposium on Field-Programmable Gate Arrays* (FPGA'00), February 10 - 11, 2000, pp. 60-66.
24. *J.L. Kouloheris, and A. El Gamal*, "PLA-based FPGA Area versus Cell Granularity", in Proceedings of the IEEE Custom Integrated Circuits Conference, 1992, pp. 4.3.1 - 4.3.4.
25. *Lucent Technologies, Inc.*, "ORCA OR2C-A/OR2T-A Series FPGAs Data Sheet", 1996.
26. *H-G. Martin, and W.A. Rosenstiel*, "A Comparing Study of Technology Mapping for FPGA", in Design Automation and Test in Europe (DATE '98) February 23 - 26, 1998, pp. 939-944.
27. *R. Murgai, Y. Nishizaki, R.K. Brayton, and A. Sangiovanni-Vincentelli*, "Logic synthesis algorithms for programmable gate arrays", in Proceedings of the ACM/IEEE Design Automation Conference, June 1990, pp. 620-625.
28. *R. Murgai, N. Shenoy, R.K. Brayton, and A. Sangiovanni-Vincentelli*, "Performance directed synthesis for table look up programmable gate arrays", in Proceedings of the IEEE International Conference on Computer-Aided Design, November 1991, pp. 572-575.
29. *R. Murgai, N. Shenoy, R.K. Brayton, and A. Sangiovanni-Vincentelli*, "Improved logic synthesis algorithms for table look up architectures", in Proceedings of the IEEE International Conference on Computer-Aided Design, 1991, pp. 564-567.
30. *J. Rose, R.J. Francis, D. Lewis, and P. Chow*, "Architecture of Field Programmable Gate Arrays: The effect of Logic Block Functionality on Area Efficiency" in IEEE Journal of Solid State Circuits, **Vol. 25**, No. 5, October 1990, pp. 1217-1225.
31. *J.P. Roth, and R.M. Karp*, "Minimization over Boolean graphs", in IBM J. Res. Dev., April 1962, pp. 227-238.
32. *E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephen, R.K. Brayton, and A. Sangiovanni-Vincentelli*, "SIS: A System for Sequential Circuit Synthesis", University of California, Berkeley, CA, Tech. Rep. UCB/ERL M92/41, 1992.
33. *K.J. Singh, A.R. Wang, R.K. Brayton, and A. Sangiovanni-Vincentelli*, "Timing optimization of combinational logic", in Proceedings of the IEEE International Conference on Computer-Aided Design, 1988, pp. 282–285.
34. *Xilinx*, "The Programmable Logic Data Book", 1997.