Synthesis Challenges for Next-Generation High-Performance and High-Density PLDs

Jason Cong
Department of Computer Science
University of California, Los Angeles, CA 90095, USA
cong@cs.ucla.edu

Songjie Xu
Aplus Design Technologies, Inc.
10850 Wilshre Blvd., Suite 370, Los Angeles, CA 90024, USA
sxu@aplus-dt.com

Abstract

Rapid evolution of programmable logic device (PLD) architectures and the increasingly demanding requirements to meet with high-density and high-performance PLD designs call for a new generation of PLD synthesis tools to support many important capabilities not available in the existing synthesis tools. In this paper, we analyze the synthesis challenges and oppotunities in supporting next-generation high-performance and high-density PLDs. We classify these challenges into two categories: (i) those implied from the advance of PLD architectures, including synthesis for hierarchical architectures and synthesis for heterogeneous architectures, and (ii) those driven by the requirements for high-density and high-performance PLD designs, including layout-driven synthesis, incremental synthesis, and IP-based synthesis. We shall discuss existing and/or potential solutions to these problems and outline research opportunities and directions for the development of next-generation PLD synthesis systems.

1 Introduction

The integrated circuit (IC) technology has enjoyed a spectacular growth in the past thirty years following the Moore's Law [27], with device density doubling every three years. As a relatively young branch of the industry, the programmable logic device (PLD) sector has also experienced an exponential growth in the past fifteen years, in fact, with an even faster pace compared to the rest of the semiconductor industry. For example, during the five year period from 1994 to 1998, in terms of total sales, the average growth of the semiconductor industry is about 28% per year, with Intel (the leading microprocessor company) growing at an average rate of 24.5% per year and LSI Logic (a leading ASIC provider) growing at an average rate of 16.7% per year. In comparison, however, Altera (one of the two leading PLD providers) had an average growth of 36% per year (see Table 1).

Company/Industry	Annual Growth Rate (1994 to 1998) (in terms of total sales)
Semiconductor Industry	27.78%
Altera	36.07%
Intel	24.50%
LSI Logic	15.71%

Table 1: The growth rate of the PLD industry with respect to the rest of the semiconductor industry (source: http://www.marketguide.com).

The rapid growth of the PLD industry is also evident from the significant advance of the programmable logic architectures in the past fifteen years. For example, the Altera MAX 5000 device family [3] launched in 1988 had only 32 to 192 product-term based macrocells with only 600 to 3,750 usable gates, In 1999, however, Altera has shipped its APEX 20K devices [3], each with up to 51,840 logic elements[1], up to 442,368 RAM bits, and up to 3,456 product-term based macrocells, which provide 60,000 to 1.5 million usable gates[2]. Similarly, Xilinx (the other leading PLD provider) introduced its XC2000 device family in 1985, with only 64 to 100 4-LUTs, providing 1200 to 1800 logic gates. In 1998, however, Xilinx has successfully shipped its Virtex devices [34] with 58K to 4M system gates, 1Mb of configurable distributed RAM and up to 832Kb of synchronous embedded memory[3]. We expect that such rapid growth of the PLD industry will continue for next ten years.

With the rapid development of the programmable logic architectures, the PLD synthesis tools and algorithms have also made significant progress. For example, the early work on logic synthesis for PLDs mainly targeted at a simple array of homogeneous LUTs. (A comprehensive survey of existing synthesis and mapping algorithms for LUT-based PLDs up to 1996 can be found in [6].) The recent PLD synthesis algorithms, however, consider using embedded memory blocks for logic implementation [33] [17] [16], take the advantage of certain heterogeneous architectures [15] [23] [16] [8] [21], and take into consideration the special architecture features such

[1] Each logic element (LE) is a 4-input lookup-table (LUT) in APEX 20K devices.

[2] The data is quoted from APEX 20K data sheet [3].

[3] The data is quoted from Virtex data sheet [34].

as complex programmable logic blocks (PLBs) [10] [11] and special interconnect resources [12]. However, most of these recent results are still at research stage, and very few of them have been adopted in available PLD synthesis tools. In general, there is an alarming trend that the capabilities of synthesis tools fall behind the advance of PLD architectures. If such a trend continues, it may seriously limit the growth of the PLD industry, as the new architecture features will not be fully utilized without the efficient support from the synthesis tools.

In this paper, we analyze the synthesis challenges and opotunities in supporting next generation of high-performance and high-density PLDs. We classify these challenges into two categories: (i) those implied from the advance of PLD architectures, including synthesis for hierarchical architectures and synthesis for heterogeneous architectures, and (ii) those driven by the requirements for high-density and high-performance PLD designs, including layout-driven synthesis, incremental synthesis, and IP-based synthesis. We shall discuss existing and/or potential solutions to these problems and outline research opportunities and directions for the development of next generation of PLD synthesis systems. We hope that this paper may stimulate an increasing level of research activities in addressing the needs for next generation of PLD synthesis systems.

The remainder of this paper is organized as follows. We briefly discuss the design process for PLDs in Section 2. Section 3 discusses the synthesis challenges implied by the new PLD architectures. Section 4 presents the synthesis challenges resulting from the ever increasing demand of designing higher density and higher performance PLDs. Section 5 concludes the paper.

2 Definitions and PLD Design Flow

We first define a few general terms used in this paper.

PLDs (programmable logic devices) can be roughly divided into two classes: CPLDs and FPGAs. *CPLDs* (complex PLDs) are extensions of early PALs. They usually consist of PLA-like programmable logic blocks with a wide number of inputs. In contrast, *FPGAs* (field-programmable gate array) usually consist of fine-granularity programmable logic blocks, such as lookup-tables or mux-based logic blocks, each with a limited number of inputs (typically 3 to 6). In general, FPGAs offer higher logic density and a higher ratio of flip-flops to logic resources than that of CPLDs.

An *LUT* (look-up table) is the most commonly used basic programmable logic element in SRAM-based FPGAs. A K-LUT refers to a K-input one-output lookup-table, capable of realizing any Boolean function of up to K variables. A *macrocell* is a basic logic block commonly used in CPLDs. A macrocell is a (programmble) OR gate with a set of product terms as the potential inputs to the OR gate. A *Pterm logic block* is a group of macrocells that may share some of the product terms.

The design process for PLDs is similar to that for conventional technologies such as standard cell or gate array. In general, it consists of high-level synthesis, logic synthesis and layout synthesis. In high-level synthesis, resources (ALUs, registers, etc.) are allocated and operations are scheduled. The logic synthesis of PLDs consists of *sequential logic synthesis*, which optimizes the assignment and/or arrangement of storage elements, and *combinational logic synthesis*, which optimizes the logic gate netlists. Finally, in layout synthesis, LUT netlists obtained from the logic-level design are placed on the PLD devices and routed with the available PLD routing resources. The combinational logic synthesis for PLDs can be further divided into two steps: technology independent synthesis, also called *logic optimization*, which transforms the gate-level network into another equivalent gate-level network that is more suitable for the subsequent design processes, and technology dependent synthesis, also called *technology mapping*, which transforms the gate-level network into a network of logic elements (or cells) in the target PLD technology by covering the network with such cells. The design objectives may be area minimization, delay minimization, simultaneous area and delay minimization, or routability optimization, etc..

3 Implication from Architectures

In this section, we discuss two important trends in PLD architecture development, the use of hierarchical architectures and the use of heterogeneous architectures, and analyze their implication on the needs of synthesis tools.

3.1 Synthesis Needs for Hierarchical Architectures

As the capacity of PLD devices increases, we see a clear trend that hierarchical architectures are being widely used, in which basic programmable logic blocks, such as LUTs or macrocells, are grouped into a logic cluster and connected by local programmable interconnects inside the cluster. These clusters are then connected using global programmable interconnects. It is possible to have multiple levels of hierarchy as well, where the first level of clusters can be further grouped into a second level cluster, and so on. The use of hierarchical architectures clearly exploits the inherent locality of interconnections that exists in most applications. It leads to the improvement in both performance (through use of fast local interconnects for the majority of connections) and density (due to the reduction of the amount of global interconnect needed). There are basically two types of clusters, the *hard-wired connection based clusters* (HCC) and the *programmable interconnect based clusters* (PIC). Evaluation results on several aspects of the architectures using these two types of clusters have been reported (*e.g.* [10] [11] [1]).

In a hard-wired connection based cluster (HCC), a group of basic logic blocks are connected directly by wires without going through programmable switches (called "hard-wires"). Obviously, hardwires provide much faster connections than programmable interconnections. An example of the HCC is the configurable logic block (CLB) in Xilinx XC4000 FPGAs [34] which consists of two 4-LUTs with their outputs hardwired to the inputs of a 3-LUT.

In a programmable interconnection based cluster (PIC), a group of basic logic blocks are connected by a local programmable interconnection array that usually provides full connectivity and is much faster than the global or semi-global programmable interconnects (which normally have to go through a lot more programmable switches). A number of commercial PLDs use the PIC architecture, such as the logic array block (LAB) in Altera FLEX 10K and APEX 20K devices, and the MegaLAB in APEX 20K devices [3]. For example, in FLEX 10K devices, each LAB consists of eight 4-LUTs connected by the local interconnect array. Multi-level hierarchy can be formed easily using PICs, in which a group of small (lower-level) PICs can be connected through a programmable interconnect array at this level to form a larger (higher-level) PICs. For example, in Altera APEX

20K FPGAs (see Figure1), each LAB consists of ten 4-LUTs connected by local interconnects, which forms the first-level PIC. Then, 16 such LABs, together with one embedded system block and another level of programmable interconnects, form a second level PIC, called MegaLAB. Finally, global interconnects are used to route between MegaLAB structures and to I/O pins.

Most of the existing PLD synthesis algorithms transform a given design into a flat netlist of basic programmable logic blocks (such as LUTs or macrocells) without consideration of the device hierarchy. There are very few studies that consider synthesis directly for hierarchical PLD architectures. For HCC-based hierarchical architectures, most works first map the given logic netlist into LUTs and then combines the LUTs to form HCCs in a heuristic post-processing step. A noticeable exception is the work in [10], which uses Boolean matching techniques [11] to completely characterize the set of functions that can be implemented in a HCC, and map the given logic network directly into a netlist of HCCs. For PIC-based hierarchical architectures, a common approach is again to map the given logic network first into a flat netlist of basic logic blocks, and then group them into clusters under size and pin constraints. (Since the local interconnects in each PIC usually provide full connectivity, there is no routability constraints inside a cluster.) Such a two-step approach may benefit from recent progress on circuit clustering, such as the results on performance-driven clustering for combinational circuits under size and pin constraints ([30], [35], [1] and [20]) and simultaneous clustering with retiming for sequential circuits ([28] and [14]). In particular, the methods in [28] and [14] enable the efficient clustering for a sequential circuit so that the clock period of the clustered circuit is quasi-optimal under retiming (differ by at most an inter-cluster delay from the optimal solution). Both methods have polynomial-time worst-case time complexity. But the method [14] improves that in [28] significantly in runtime and scales very well to large designs in practice.

Clearly, the biggest challenge and opportunity in this area is to be able to synthesize a given design directly into a multi-level hierarchical architecture, with consideration of different interconnect delays and clustering constraints at each level. The demand for such a hierarchical synthesis/mapping solution is even greater today, as more and more new PLD architectures start using multi-level hierarchy (such as APEX 20K from Altera and ProASIC 500K from Actel [2]).

3.2 Synthesis Needs for Heterogeneous Architectures

Early PLDs usually have a simple array of homogeneous programmable logic blocks. As the PLD capacity increases, the resources on a PLD become heterogeneous. The heterogeneity of a PLD architecture can be roughly classified into the following types. (1) A heterogeneous architecture may have several sizes and/or configurations of the same type of logic blocks. For example, it may contain LUTs of different sizes. (2) A heterogeneous architecture may have different types of programmable logic blocks, such as LUTs, macrocells, and muxes. (3) A heterogeneous architecture may have different kinds of resources on the same chip, such as programmable logic blocks, embedded memory blocks (EMBs), and embedded processors. How to efficiently utilize different types of logic blocks and different kinds of resources is a challenging yet important problem to the next generation of PLD synthesis tools, especially in the coming system-on-a-chip era.

Several commercial architectures, such as those from Lucent Technologies [31], Vantis [32], and Xilinx [34], fall into the category of the first type of heterogeneous PLDs. They provide programmable logic blocks (PLBs) that can be configured to implement LUTs of different sizes, as studies suggest that "one-size fits all" may not provide the best performance and/or density for LUT-based FPGAs, especially for large designs. Much progress has been made recently on synthesis for heterogeneous LUT-based architectures. In [19], a technology mapping algorithm for heterogeneous LUTs of two sizes with a specific ratio is developed for area minimization. This algorithm is then used to evaluate the heterogeneous FPGA architectures with LUTs of different sizes and some interesting results are obtained. The same problem is studied in [23] and an area optimal algorithm is developed targeted for trees. The cut enumeration based approach presented in [8] can also be applied to minimize the area for mapping a general logic netlist into heterogeneous LUTs. In [15], a polynomial-time delay-optimal algorithm, named HeteroMap, is presented for heterogeneous FPGAs with LUTs of different sizes, assuming different delays for LUTs of different sizes. Using this algorithm, several interesting experiments are conducted to evaluate the heterogeneous FPGA architectures with the comparison to the corresponding homogeneous ones. From the comparison results, it is clearly shown that heterogeneous FPGAs are superior to the homogeneous ones in terms of both delay and area optimization.

The APEX 20K device family [3] can be considered as a heterogeneous PLD architecture of the second type, as it provides two different types of programmable logic resources, LUTs and P-term based logics (through a clever reconfiguration of embedded memory blocks). In general, LUT-based FPGAs tend to offer higher density, while P-term logic based CPLDs tends to offer better performance. A proper combination of both types of logic blocks may achieve a good density/performance trade-off. However, very little result is available on synthesis or architecture study of such hybrid architectures. The only work found in this area [21] conducted a preliminary study on a hybrid PLD with both LUTs and product term based logic and proposed some heuristic approaches to the synthesis problem for such a hybrid architecture.

There are a lot of research activities towards developing the third types of heterogeneous PLDs, aiming at high-degree on-chip integration of different kinds of resources, including programmable logic blocks, embedded memory blocks (EMBs), embedded processors, or even customized IP cores. We also call such a device *field-programmable system-on-a-chip* (FPSOC), as it is intended for a single-chip implementation of an entire system on a PLD device. The introduction of on-chip EMBs is the first step in this direction (as first introduced in Altera FLEX 10K devices [3]). It is worthwhile to mention that the recent PLD synthesis algorithms can consider using EMBs for logic implementation [33] [17] to reduce the circuit area (when EMBs are not used by designers as on-chip memories). Another work on delay-oriented technology mapping for heterogeneous FPGAs with bounded resources on specific types of LUTs was introduced in [16], which can be directly applied to the mapping of logic into EMBs for delay reduction. However, the general synthesis problem for FPSOC is largely untouched. We believe that synthesis solutions to FPSOC designs require more advanced capabilities, such as hardware/software partitioning and interface synthesis, which go much beyond what existing

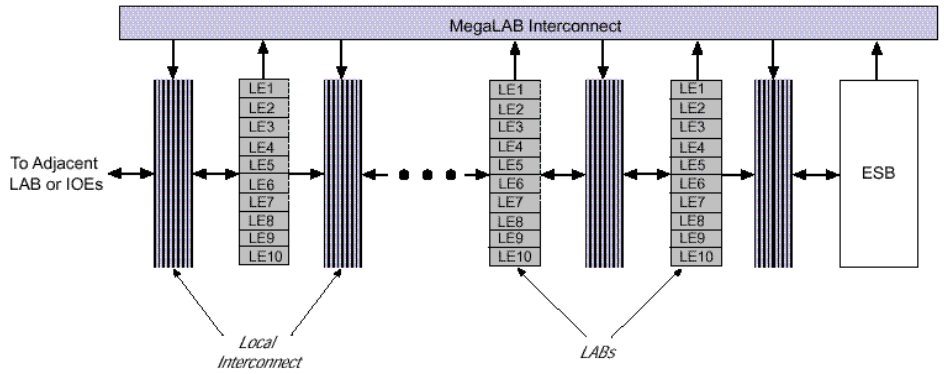

Figure 1: APEX 20K megaLAB structure.

PLD synthesis tool can offer.

4 Implication from Demand of Density and Performance

The requirements for supporting high-performance and high-density PLD designs have also generated a number of interesting and important synthesis problems. We rank layout-driven synthesis, incremantal synthesis, and IP-based design to be the three most critical ones for next-generation of PLD synthesis systems. We shall discuss the needs in these areas in more details in the remaining of this section.

4.1 Layout-Driven Synthesis

With the rapid scaling of transistor feature sizes, the integrated circuit performance is increasingly determined by interconnects instead of devices. Interconnect delays are even more significant in PLD designs due to the extensive use of programmable switches. As a result, the delay between two logic blocks is highly dependent on their placement on the chip and the routing resources used to connect them. For example, Table 2 shows the timing parameters of Altera FLEX10K100 device in -3 speed grade [3], whose device diagram is shown in Figure 2. Local interconnects refer to the connections between logic elements (LEs) in the same logic array block (LAB). Row interconnects refer to the connections between LEs in the same row, but in different LABs. Column interconnects refer to the connections between LEs in different rows. Given such a high variation of different types of interconnect delays, it would be almost impossible to perform accurate timing optimization during synthesis without proper consideration of the layout result. Therefore, we rank layout-driven synthesis to be the number one problem required to support high-performance PLD designs.

Delay Resource	Delay Value (ns)
Logic Element (LE)	2.0
Local Interconnect	0.4
Row Interconnect	5.1
Column Interconnect	10.0

Table 2: Timing parameters of Altera EPF8282A part in A-2 speed grade.

In general, there are two approaches to integrate logic and layout synthesis. One approach is to employ a highly

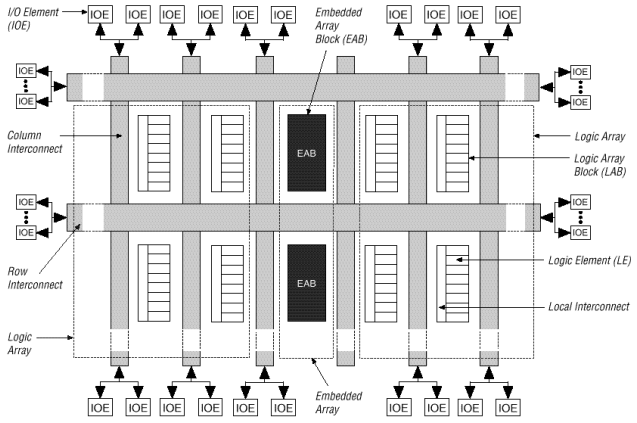

Figure 2: FLEX10K device block diagram.

iterative design flow. It follows the design steps in the traditional design flow, but feeds the layout result in the current iteration back to logic synthesis tools for improving the synthesis results in the next iteration. It is yet to be proved if such a "construct-by-correction" approach will converge to a delay-optimal solution. The preliminary studies based on such an iterative approach as reported in [7] were not very encouraging.

Another approach is to use a concurrent design flow, which performs logic synthesis/technology mapping and placement/routing concurrently. For example, the early work of using a companion placement during technology mapping [29], the hierarchical approach in [4], and the recent work of simultaneous technology mapping with linear placement [25] [26] all belong to this category. However, the optimality of such approaches usually holds for very special circuit structures (such as trees) and the main difficulty associated with this approach is its high computational complexity.

Clearly, much more study is needed to develop an effective and efficient layout-driven synthesis flow. Possible directions include developing better delay models during synthesis and mapping with consideration of layout information, efficient area and delay estimation techniques which enable early design planning, and use of floorplan and interconnect planning techniques to guide the synthesis and subsequent layout procedures, etc. Before we leave this topic, we would like to highlight two recent results which considered certain degree of layout information during synthesis

and design planning, and reported very encouraging results.

1. It was shown in [14] that by combining logic partitioning with retiming with proper consideration of global and local interconnect delays, one can achieve 40 to 50% delay reduction compared to that of the state-of-art conventional circuit partitioning approaches, such as the LSR algorithm [13], which consider only cutsize minimization.

2. It was shown in [12] that by exploiting fast interconnects available in many PLD architectures (such as fast carry-chains in Altera architectures) during technology mapping, one can achieve 20 to 30% delay improvement compared to the conventional depth-optimal mapping approach [5].

Both results indicate that layout-driven synthesis may have a great potential in achieving much higher performance.

4.2 Incremental Synthesis

As the capacity of PLD devices increases, the complexity of the designs to be implemented on PLD devices increases significantly. Similar to the design of all complex systems, the design process for high-capacity PLDs is likely to go through many incremental changes for debugging, functional modification, performance enhancement, etc, before the design is finalized. As a result, it may require many iterations of synthesis, placement and routing, simulation/verification, and timing analysis to complete a complex design. Clearly, such a highly incremental design process requires fast incremental synthesis capabilities.

Traditionally, after each iteration of design modification, the entire design needs to go through a pass of synthesis, mapping, placement and routing. It might be acceptable for small designs ($10,000$ to $100,000$ gates) to re-synthesis the whole network after each iteration. However, as million-gate PLDs become available on the market [3] [34], re-synthesis the entire design is no longer acceptable for supporting large designs with consideration of possible multiple design iterations. For example, running the well-known depth-optimal LUT mapping algorithm FlowMap [5] 50 times for 50 incremental changes already takes over 1.5 hours even for a circuit with less than $30,000$ gates ('clma' in the MCNC benchmark suite). Given the quadratic time complexity of the algorithm, the runtime for multiple iterations for circuits of 10 - 20x larger will not be tolerable.

In practice, however, incremental changes tend to affect only a small portion of the design and therefore only a small part of the synthesis solution needs to be revised. It is more desirable to have an approach that updates only the affected part in the synthesis solution and leaves the rest part of the solution untouched. Such an incremental synthesis system should consider the following three important objectives:

1. "Preservability". The incremental synthesis system should preserve as much information as possible from the existing synthesis solution. This will lead to fast convergence in the design process. If only a small portion of the original synthesis solution is modified, the related steps, such as timing analysis, placement, routing, and verification may run much faster (if they are designed to take the advantage of incremental changes).

2. Efficiency. A faster synthesis system will enable more design iterations and shorten the overall design time.

3. Quality of the synthesis solution (such as the delay or area) should be as close as possible to that by complete re-synthesis.

Very few papers addressed the incremental design issues for PLDs. An early work considered the engineering order change problem in FPGA design [24]. However, it requires that no change of the structure of the mapping solution nor the placement and routing solution. The only allowed change is the functionality of LUTs. As a result, this method can be applied to very limited incremental changes. In a very recent study, an incremental technology mapping algorithm is proposed. It can efficiently handle incremental changes while preserving the optimal mapping depth. It achieved over 300 times speed-up for moderate size circuits (close to 100,000 gates) when compared with re-mapping using the FlowMap algorithm for a sequence of incremental changes [9].

4.3 IP-Based Design

As the design complexity of PLD devices increases, design reuse is another important mean of improving the design productivity. Effective reuse of existing designs or intellectual properties (IP) requires proper representation, abstraction, and characterization of an IP in terms of its functionality and performance, so that it can be seamlessly integrated into the entire system for synthesis, simulation and verification. For example, given a complex IP block with many inputs, outputs, and multiple clocks inside, it is not a trivial problem to precisely characterize the timing behavior of the block so that it can be fully considered by the synthesis tool. Many existing synthesis algorithms also need to be extended in order to consider such 'black-boxes' in the design. For example, Figure 3 illustrates a typical design flow for using LPMs (Library of Parameterized Modules) [18]. There are many challenges for the synthesis tools to identify and extract certain logic functions from a given design and implement them using the available LPM library or IP library.

For performance-critical IP blocks, it is necessary to have certain layout information associated with them in order to achieve the best performance. However, such representation and abstraction should also support efficient update of an existing IP when we migrate from one PLD device to another one in the same family, or even from one PLD vendor to another.

IP protection is another important challenge, where the goal is to prevent un-authorized use of IPs. It may require adding watermarks into IP designs, so that un-authorized use of such IPs can be easily identified. The work in [22] presented an interesting study of embedding watermarks in an FPGA mapping solution.

5 Concluding Remarks

This paper identified the critical needs and challenges for the next-generation PLD synthesis tools for high-density and high-performance PLD designs. We classified these needs into two categories: those implied from the advance of PLD architectures, including synthesis for hierarchical architectures and synthesis for heterogeneous architectures, and those driven by the requirements for high-density and high-performance PLD designs, including layout-driven synthesis, incremental synthesis, and IP-based synthesis. These needs and challenges have created many new and important research directions and business opportunities, and are receiving more and more attention from the research community

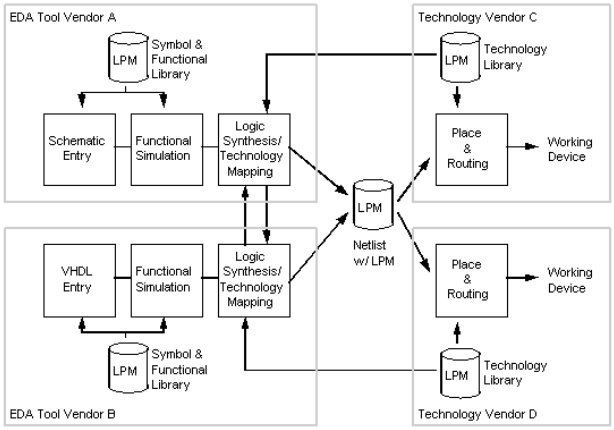

Figure 3: The typical design flow using LPMs.

and the EDA industry. For example, the UCLA VLSI CAD Laboratory has a very active research program in the area of PLD synthesis and architecture evaluation, and is currently pursuing several research topics discussed in the paper. The reader is welcome to visit http://cadlab.cs.ucla.edu/project/fpga/ for most updated results. Aplus Design Technologies, Inc. (http://www.aplus-dt.com/), a new start-up company for providing design automation tools for PLD devices, are working closely with several major PLD vendors in developing efficient synthesis, design planning, and architecture evaluation tools for next generation of high-density and high-performance PLDs. We hope that this paper can help to stimulate more research and development activities in these areas.

References

[1] J. Rose A. Marquardt, V. Betz. Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density. In *Proc. ACM Seventh International Symposium on Field Programmable Gate Arrays*, pages 37–46, 1999.

[2] Actel. *FPGA Data Book*. 1999.

[3] Altera. *Programmable Logic Devices Data Book*. Altera Corp., San Jose, CA, 1999.

[4] C-S. Chen, Y-W. Tsay, T-T. Hwang, A-C. Hu, and Y-L. Lin. Combining Technology Mapping and Placement for Delay Optimization in FPGA Designs. In *Proc. IEEE Int'l Conf. on Computer-Aided Design*, pages 123–1027, 1993.

[5] J. Cong and Y. Ding. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits And Systems*, 13(1):1–12, 1994.

[6] J. Cong and Y. Ding. Combinational Logic Synthesis for SRAM Based Field Programmable Gate Arrays. *ACM Transactions on Design Automation of Electronic Systems*, 1(2):145–204, 1996.

[7] J. Cong, Y. Ding, T. Gao, and K. C. Chen. LUT-Based FPGA Technology Mapping Under Arbitrary Net-Delay Models. *Computers and Graphics*, 18(4):137–148, 1994.

[8] J. Cong, Y. Ding, and C. Wu. Cut Ranking and Pruning: Enabling A General And Efficient FPGA Mapping Solution. In *ACM 7th Int'l Symp. on Field Programmable Gate Arrays*, pages 29–35, 1999.

[9] J. Cong and H. Huang. Depth Optimal Incremental Mapping for Field Programmable Gate Arrays. In *UCLA Computer Science Department Technical Report CSD-TR 990055*, 1999.

[10] J. Cong and Y.-Y. Hwang. Partially-Dependent Functional Decomposition with Applications in FPGA Synthesis and Mapping. In *ACM 5th Int'l Symp. on Field Programmable Gate Arrays*, pages 35–42, 1997.

[11] J. Cong and Y.-Y. Hwang. Boolean Matching for Complex PLBs in LUT-based FPGAs with Application to Architecture Evaluation. In *ACM 6th Int'l Symp. on Field Programmable Gate Arrays*, 1998.

[12] J. Cong, Y.-Y. Hwang, and S. Xu. Technology Mapping for FPGAs with Nonuniform Pin Delays and Fast Interconnections. In *Proc. 36th ACM/IEEE Design Automation Conference*, pages 373–378, 1999.

[13] J. Cong, H. Li, S. Lim, T. Shibuya, and D. Xu. Large Scale Circuit Partitioning With Loose/Stable Net Removal And Signal Flow Based Clustering. In *IEEE International Conference on CAD*, pages 441–446, 1997.

[14] J. Cong, H.-C. Li, and C. Wu. Simutanious Circuit Partitioning/Clustering with Retiming for Performance Optimization. In *Proc. 36th ACM/IEEE Design Automation Conference*, pages 460–465, 1999.

[15] J. Cong and S. Xu. Delay-Optimal Technology Mapping for FPGAs with Heterogeneous LUTs. In *Proc. 35th ACM/IEEE Design Automation Conference*, pages 704–707, 1998.

[16] J. Cong and S. Xu. Delay-Oriented Technology Mapping for Heterogeneous FPGAs with Bounded Resources. In *Proc. IEEE Int'l Conf. on Computer-Aided Design*, pages 40–45, 1998.

[17] J. Cong and S. Xu. Technology Mapping for FPGAs with Embedded Memory Blocs. In *Proc. ACM Int'l Symposium on FPGA*, pages 179–188, 1998.

[18] EIA. LPM Module Specification (210). In *http://www.edif.org/lpmweb*, 1999.

[19] J. He and J. Rose. Technology Mapping for Heterogeneous FPGAs. In *Proc. ACM Int'l Symposium on FPGA*, 1994.

[20] J.-D. Huang, J.-Y. Jou, W.-Z. Shen, and H.-H. Chuang. On Circuit Clustering for Area/Delay Tradeoff under Capacity and Pin Constraints. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(4):634–642, 1998.

[21] Kaviani. Novel Architecture and Synthesis Methods for High Capacity Field Programmable Devices. In *PH.D. Thesis, University of Toronto*, 1999.

[22] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong. Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions. In *Proc. ACM/IEEE International Conference on Computer Aided Design*, pages 194–198, 1998.

[23] M. Korupolu, K. Lee, and D. Wong. Exact Tree-based FPGA Technology Mapping for Logic Blocks with Independent LUTs. In *Proc. 35th ACM/IEEE Design Automation Conference*, pages 708–711, 1998.

[24] Y. Kukimoto and M. Fujita. Rectification Method for Lookup-Table Type FPGAs. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 54–61, 1992.

[25] J. Lou, A. Salek, and M. Pedram. An Exact Solution to Simultaneous Technology Mapping and Linear Placement Problem. In *Proc. IEEE Int'l Conf. on Computer-Aided Design*, pages 671–675, 1997.

[26] J. Lou, A. Salek, and M. Pedram. An Integrated Flow for Technology Remapping and Placement of Sub-half-micron Circuits. In *Proc. Asia and South Pacific Design Automation Conf.*, pages 295–300, 1998.

[27] G. E. Moore. Cramming More Components onto Integrated Circuits. *Electronics Magazine*, 38:114–117, 1965.

[28] P. Pan, A. K. Karandikar, and C. L. Liu. Optimal Clock Period Clustering for Sequential Circuits with Retiming. *IEEE Trans. on Computer-Aided Design of Integrated Circuits And Systems*, 17(6):489–498, 1998.

[29] M. Pedram and N. Bhat. Layout Driven Technology Mapping. In *Proc. ACM/IEEE 28th Design Automation Conf.*, pages 99–105, 1991.

[30] R. Rajaraman and D. F. Wong. Optimal Clustering for Delay Minimization. In *Proc. 30th ACM/IEEE Design Automation Conference*, pages 309–314, 1993.

[31] Lucent Technologies. *ORCA OR2C-A/OR2T-A Series FPGAs Data Sheet*. Lucent Technologies, Inc., Allentown, PA, 1999.

[32] Vantis and AMD Company. *Vantis VF1 Field Programmable Gate Array*. Advanced Micro Devices, Inc., Sunnyvale, CA, 1998.

[33] S. J. E. Wilton. SMAP: Heterogeneous Technology Mapping for Area Reduction in FPGAs with Embedded Memory Arrays. In *Proc. ACM Int'l Symposium on FPGA*, pages 171–178, 1998.

[34] Xilinx. *The Programmable Logic Data Book*. Xilinx Inc., San Jose, CA, 1999.

[35] H. Yang and D. F. Wong. Circuit Clustering for Delay Minimization under Area and Pin Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(9):976–986, 1997.