A HIGH-PERFORMANCE CMOS PROGRAMMABLE LOGIC CORE FOR SYSTEM-ON-CHIP APPLICATIONS

Yi Han

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2005

Program Authorized to Offer Degree:
Department of Electrical Engineering

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

University of Washington

Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Yi Han

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

Carl M. Sechen

Co-Chair of Supervisory Committee:

Larry McMurchie

Reading Committee:

Carl M. Sechen

Larry McMurchie

Scott Hauck

Date: 1/19/05

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, P.O. Box 1346, Ann Arbor, MI 48106-1346, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature Yi Han

Date 12/10/2004

University of Washington

Abstract

A High-Performance CMOS Programmable Logic Core for System-On-Chip Applications

Yi Han

Chairperson of the Supervisory Committee

Professor Carl M. Sechen

Co-Chairperson of the Supervisory Committee

Professor Larry McMurchie

Department of Electrical Engineering

Integrated circuits (ICs) have experienced an astonishing revolution in the last two decades. As predicted by Moore's law, the integration density doubles approximately every eighteen months. Nowadays hundreds of millions of transistors can be integrated on a chip. To lower the cost, system-on-a-chip (SoC) has emerged as a solution and attracted attention. SoC technology is the packaging of all the necessary functionality, including digital, analog and radio frequency, on a single chip. It has been the hot spot of research for both industry and academic and is becoming the mainstream of IC design.

As part of the SoCs, programmable logic cores (PLCs) have become more and more popular for its capability of post-fabrication changes. PLCs provide better flexibility, less cost and shorter time-to-market as well. On the other hand, the post-fabrication flexibility sacrifices speed, resulting in a speed gap between fixed logic and programmable logic, and making programmable logic far from mainstream. Typically, a circuit implemented in programmable logic will run about 3 times slower than that implemented in fixed logic [8]. Improving the speed of programmable logic is one of the primary challenges.

In this dissertation, a novel high-speed PLC architecture is proposed. By combining a high-performance dynamic logic style (output prediction logic or OPL), a product-term-based structure, mixed logic/routing structures, wired-OR structures and a unidirectional routing flow, this architecture achieves an average speedup of 4.1 times over the Xilinx Virtex-E FPGA [50], a Look-Up-Table (LUT) based architecture using static CMOS. It also can achieve up to a 1 GHz throughput, which is 2.4 times that of the Xilinx CoolRunner-II CPLD [49] and 3.2 times that of the Xilinx Virtex-E FPGA (Speed Grade -7) [50]. Measurement results for a prototype block fabricated in the TSMC 0.18μm/1.8V CMOS process are presented.

This thesis work demonstrates that programmable logic can achieve comparable speeds to ASICs with design innovations, and therefore enables the adoption of programmable logic on an SoC.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank those who have helped me during my Ph.D. career at University of Washington. The work presented in this thesis could not have happened without their support.

First of all, I would like to acknowledge my advisor, Professor Carl M. Sechen, who has been supporting and advising me at UW. He has been my role model for his great professionalism, insight and enthusiasm in the research of integrated circuit design. He has been guiding me throughout my courses and research activity and offers lots of insightful suggestions and efforts on my research. It has been my great pleasure to be one member of his group.

I would like to thank my associate advisor, Professor Larry McMurchie, who helped me initialize my research topic and gave me invaluable support whenever I encountered problems.

I want to thank my exam committee members, Professor Scott Hauck and Professor David Allstot for taking time to serve on my general and final exam committees. In particular, I would like to thank Professor Hauck for taking the time to read my general proposal and dissertation, and for his helpful suggestions that improved my research work.

I appreciate the helpful discussions and friendship offered by my colleagues in VLSI&CAD lab at UW. They made my life at UW colorful and worth remembering. I want to especially thank Xinyu Guo for designing our first version of PCB board.

I am grateful for the financial support provided by the National Science Foundation (NSF), MARCO/C2S2, Boeing/DARPA, and Intel Corporation. I also want to

thank MOSIS for their support of our test-chip fabrication, Professor Diorio for providing the testing equipments, Mark for his help on the chip-on-board design and Gigatest Labs for chip measurement support.

I would like to thank my parents for their love, which has been and will be a source of support and encouragement during my whole life. They taught me to be honest, nice and hardworking, which I will remember forever.

Finally, I am indebted to my husband, Dr. Xiaoyong Li, for his love, for his endless support, advice and encouragement throughout my Ph.D career. His kindness, optimism and friendliness have deeply impacted me and will continue in the future.

I thank all the people who helped me in this work, and apologize to those I may have forgotten here.

DEDICATION

TO MY PARENTS

Chapter 1 : Introduction

1.1 Motivation

As VLSI technology grows rapidly, it is feasible to integrate a system on the same die, called System-on-a-Chip (SoC). The SoC approach is attractive for a number of reasons. Using this technique, the designer doesn't need to comprehend the details of each core, but can instead focus on higher, system-level issues. Meanwhile, the cores can be designed by experts in an individual protocol or function. In this way the SoC design methodology provides a natural framework for the implementation of a large chip among many designers. In addition, the SoC methodology provides a way to amortize the cost of the design and verification of cores among multiple chips. The ultimate result will be a significant reduction in both the cost and design time of complex integrated circuits [8].

In highly integrated systems such as SoCs, it will be more efficient in both design time and cost if they can be modified during operation to meet a variety of tasks while installed at a customer's site. Therefore, post-fabrication changes are desired, which can be implemented by integrating programmable logic cores (PLCs) into SoC designs.

Figure 1-1 shows an SoC configuration example containing a programmable logic core [8], where the processor executes the commands configured by software. The functionality of the fixed logic is fixed at design time and that of the programmable logic can be specified through hardware configuration.

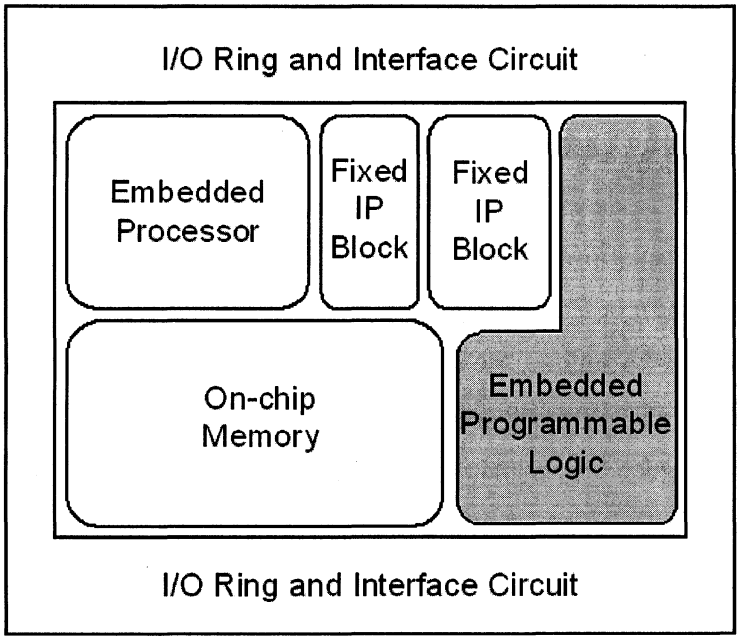


Figure 1-1 Example of SoC configuration [8]

Several companies already provided programmable logic cores [1], [2], [3], [4], [5], [6], [7], [54], [54]. In academia, a lot work has also been done on this topic to observe its opportunities and challenges [8], [9], [10], [11], [44], [58], [59], [60], [68].

The researchers in [58], [59], [60], [68] aimed to automate the process of designing domain specific reconfigurable logic for SoC application to reduce time-to market or design costs. In TOTEM project, three tools in the flow are developed, namely, the architecture generator, the VLSI layout generator and the place and route tool. The architecture generator automatically creates a high-level reconfigurable architecture description based on some description of the targeted applications. The VLSI layout generator is responsible for producing efficient VLSI layouts from the Verilog architecture description. The place and route tool takes the needed application netlists and outputs a bitstream that can be used to configure the

generated architecture. The reconfigurable logic generated by TOTEM package achieves ASIC speed with the capability of post-fabrication modifications. By bridging the performance gap between reconfigurable logic and ASICs, it broadens the application of programmable logic in SoC.

In [44], they proposed a new family of architectures for vendor-supplied synthesizable programmable logic core. The synthesized programmable logic cores are then embedded into SoC as a reconfigurable resource. Unlike the previous architectures which were based on look-up-tables, this family is based on sum-of-product-term structure to obtain higher speed and smaller area. A novel fully-populated routing fabric was developed to make the architecture flexible. Compared to lookup-table based architectures, the new architectures result in density improvements of 35% and speed improvements of 72% on standard benchmark circuits.

However, these programmable cores are, for the most part, based upon existing FPGA circuits and architectures. As appealing as this approach is, a major problem is that a performance gap often exists between the programmable core and other parts of the SoC, particularly microprocessor cores. This performance gap is partly due to the flexible nature of FPGA architectures, and that flexibility necessitates an abundance of routing resources and increased delay.

Another reason for the performance gap is that delay-critical parts of a microprocessor core are normally implemented in dynamic logic styles such as domino or differential domino. Domino circuits, for example, average about 1.6 times the speed of static CMOS circuits [12]. Other more aggressive design styles, such as output prediction logic (OPL) that will be described later, have shown speedups of up to 5 times for circuits that efficiently map to wide gates (*e.g.* NORs). In contrast, the underlying circuit style of programmable cores and FPGAs

is still static CMOS and/or pseudo-nMOS, virtually the same circuit design styles that have been used since the advent of FPGAs.

Unless the performance gap between microprocessors and PLCs is reduced, the use of programmable cores in SoCs may be limited. In particular, programmable logic blocks will be increasingly relegated to applications that allow a high degree of pipelining and that tolerate high latency.

An example of an application for a PLC is where the PLC block acts as a reconfigurable functional unit tightly coupled with a microprocessor core. Researchers Ye, Moshovos, Hauck and Banerjee have proposed an architecture for such reconfigurable units with implementations in static CMOS, obtaining modest speedups of 20% for general purpose computing [13]. Obviously, as microprocessor core speeds increase, an associated PLC functional block must also increase in performance or else it will not be cost effective to include on the same die.

In this dissertation a high performance CMOS programmable logic core for SoC applications is explored. It provides an ideal solution to the problem we discussed above.

1.2 Overview

This dissertation is organized as follows. In Chapter 2, a brief introduction to general field-programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs) is given.

In Chapter 3, the underlying circuit technique – OPL is described. The speed and power consumption of several OPL family candidates, such as OPL-static, OPL-pseudo, OPL-dynamic and OPL-differential logics are presented.

Chapter 4 begins with a complete description of the proposed programmable logic core architecture. Then the discussion of the clock distribution scheme in the proposed architecture is provided. Finally, comparisons between the proposed architecture and commercial products on performance and energy are given.

Experimental data is given in Chapter 5, including detailed implementation, the testing setup, measured results and parasitic effects analysis.

Chapter 6 concludes the thesis and points out the direction to improve the current design.

Chapter 2 : Fundamentals of Programmable Logic

2.1 Introduction

Due to the development of sophisticated programmable logic components over the past few years, the process of designing digital hardware has changed dramatically. FPGAs and CPLDs have become widely accepted for implementations ranging from small devices capable of implementing only a handful of logic equations to entire processor cores. Programmable logic offers a powerful, viable alternative to ASICs because they significantly reduce engineering development time and the cost of multiple silicon iterations. Unlike ASICs that risk re-spins as designs change, programmable logic can be programmed and reprogrammed as needed during the design process. They can be upgraded in the field if requirements change. Also, their ability to re-use a common hardware platform is important. It allows designers to create differentiated systems that support a variety of feature sets with one basic design, resulting in reduced manufacturing costs [30].

2.2 User-Programmable Switch Technologies

Before describing the typical structures of programmable logic, let's first take a look at the user-programmable switch technologies. There are several types of switch technologies, such as floating gate transistors, which are mainly used in CPLDs; static random access memory (SRAM), and the antifuse, which are widely used in FPGAs.

Floating gate transistors used in programmable logics include those used in erasable-programmable read-only memory (EPROM) and electrically erasable read-only memory (EEPROM). Figure 2-1 shows the cross section of a floating

gate. The structure is similar to a traditional MOS device, except that an extra polysilicon strip is inserted between the gate and channel. This strip is not connected to anything and is called a floating gate. It increases the effective gate oxide thickness, which results in reduced device transconductance, as well as an increased threshold voltage. More important, this device has the interesting property that its threshold voltage is programmable. Applying a high voltage (above 10V) between the source and gate-drain terminals creates a high electric field and causes avalanche injection to occur. Electrons with sufficient energy are trapped on the floating gate and effectively drop the voltage on that gate. Therefore, the threshold voltage of this transistor is increased, generally above the normal power supply voltage and the transistor is off under normal operation [14].

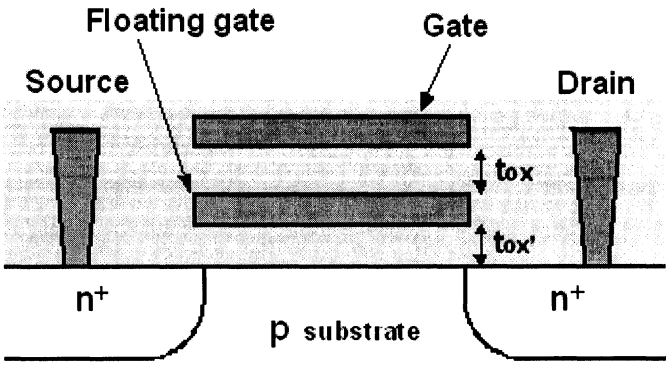


Figure 2-1 Device cross section of floating gate transistor

In order to erase the voltage, EPROM and EEPROM have different procedures. EPROM transistors are erased by shining ultraviolet light on the cell through a transparent window in the package. EEPROM transistors can be erased by reversing the voltage that is used during the program process.

Compared to EPROM, EEPROM occupies more area because an extra transistor is connected in series with the floating gate transistor to remedy the unpredictable threshold voltage of a floating gate transistor. Both of them require a nonstandard CMOS technology to manufacture, and the number of programming times is limited.

The main switch technologies used in FPGAs are SRAM and antifuse. An antifuse is not re-programmable, which makes design update impossible. An SRAM allows the configuration to be changed on the fly during circuit operation, allowing read-write access to the chip.

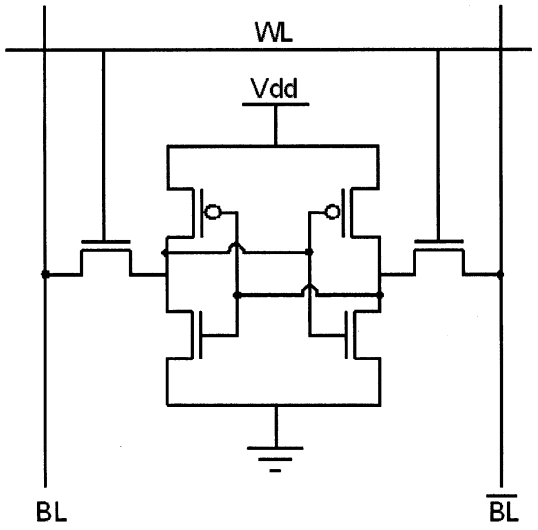


Figure 2-2 Schematic of 6-transistor SRAM (WL: word line; BL: bit line)

Figure 2-2 shows a widely used 6-transistor SRAM cell. It utilizes a full latch to store the data when the power is on. Data is written in or read out through bit lines when the word line is on. Although an SRAM occupies more area than an antifuse, its re-programmability makes it suitable for bug fixes or upgrades. It is also compatible with standard CMOS processes, which lowers the device cost. Since

SRAMs are volatile, an SRAM-based design must be reprogrammed every time the system is powered up.

2.3 Overview of Programmable Logic

2.3.1 Structure of Simple PLDs

There are two basic types of simple PLDs (or SPLDs): programmable logic array (PLA) and programmable array logic (PAL).

As shown in Figure 2-3, the PLA consists of two levels of logic gates, a programmable AND-plane followed by a programmable OR-plane. A PLA is structured so that any of its inputs (or their complements) can be AND'd together in the AND-plane; each AND-plane output can thus correspond to any product term of the inputs. Similarly, each OR-plane output can be configured to produce the logical sum of any of the AND-plane outputs. With this structure, PLAs are well suited for implementing logic functions in sum-of-products form. They are also quite versatile, since both the AND terms and OR terms can have many inputs (this feature is often referred to as wide AND and OR gates) [15].

Figure 2-4 illustrates a PAL structure. PALs feature only a single level of programmability, consisting of a programmable AND-plane that feeds fixed OR-gates. To compensate for the lack of generality incurred because the OR-plane is fixed, several variants of PALs are produced, with different numbers of inputs and outputs, and various sizes of OR-gates [15].

Although a PLA has more flexibility than a PAL since both the AND array and the OR array are programmable, this flexibility results in slower speed. The degradation is mainly due to the fact that a signal has to pass through two

programmable connections instead of one in a PAL. However, a PAL also has a limitation due to the non-shared product terms between different outputs, which decreases the logic density of a PAL. Therefore, for better speed, the PAL is the better choice; for better flexibility, the PLA is preferred. In conventional static-logic-based design, PALs are the most commonly used SPLDs [16], [61].

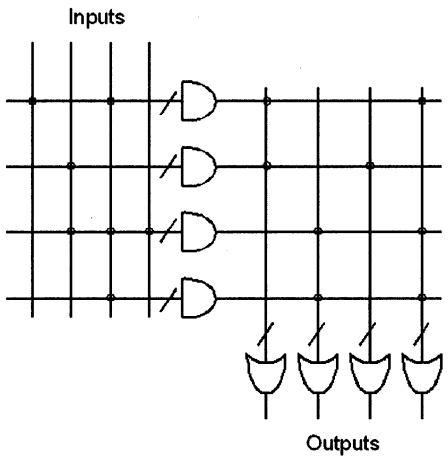


Figure 2-3 Structure of a PLA structure

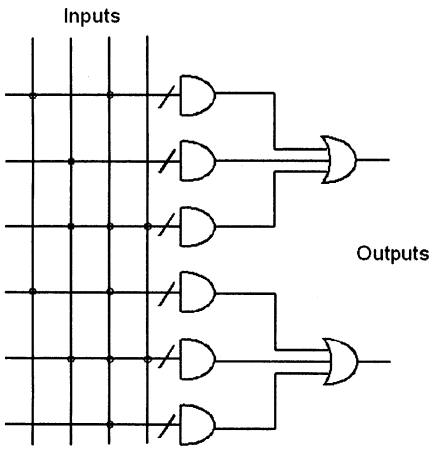


Figure 2-4 Structure of a PAL

The AND-OR architecture has a considerable number of advantages. Because of the simplicity and regularity of SPLDs, it is fairly easy to map functions onto them. Therefore the design software is free or cheaply available. Propagation delays are known and almost constant so that the timing relationship between the inputs and the outputs can be predicted. Due to the large market, prices are kept low [17].

Instead of using static CMOS logic, generally pseudo-nMOS logic is utilized in PLAs/PALs because only the pull-down network must be implemented, thereby reducing transistor stack height and increasing speed. The AND-plane is also implemented with a NOR gate by inverting the polarity of all the inputs. However, the static power is high in pseudo-nMOS due to the always-on pMOS pull-up device.

2.3.2 Architecture of CPLDs

Figure 2-5 shows a typical CPLD structure. The CPLD can be viewed as a collection of SPLDs and an interconnection network [18]. Generally every logic block (LB) is based on the PLA or PAL (two-level structures), which has a large number of inputs and product terms. The LBs communicate via the interconnect matrix (IM), which provides configurable connectivity between the inputs, logic arrays and any feedback from outputs.

Although most of the LBs are based on the PLA or PAL, some researchers have discovered that for many functions, three-level networks are more efficient and may require fewer levels of logic overall compared to two-level AND-OR networks. Especially for some functions, at least three–level realizations are required because their two-level ones are unacceptably expensive in area and delay. A notable example is an n-bit adder, whose AND-OR and OR-AND-OR networks require $O(2^n)$ and at most $O(n^2)$ gates, respectively [20]. In [21], [22], [23], [24], [25], [26],

[27], different combinations of three-level networks were exploited, such as NAND-NAND-NAND, AND-AND-OR, AND-OR-OR, OR-AND-OR, XOR-AND-OR, AND-OR-AND, and so on. Experimental results prove that area-efficient high-performance realizations of functions are achievable by using a three-level architecture.

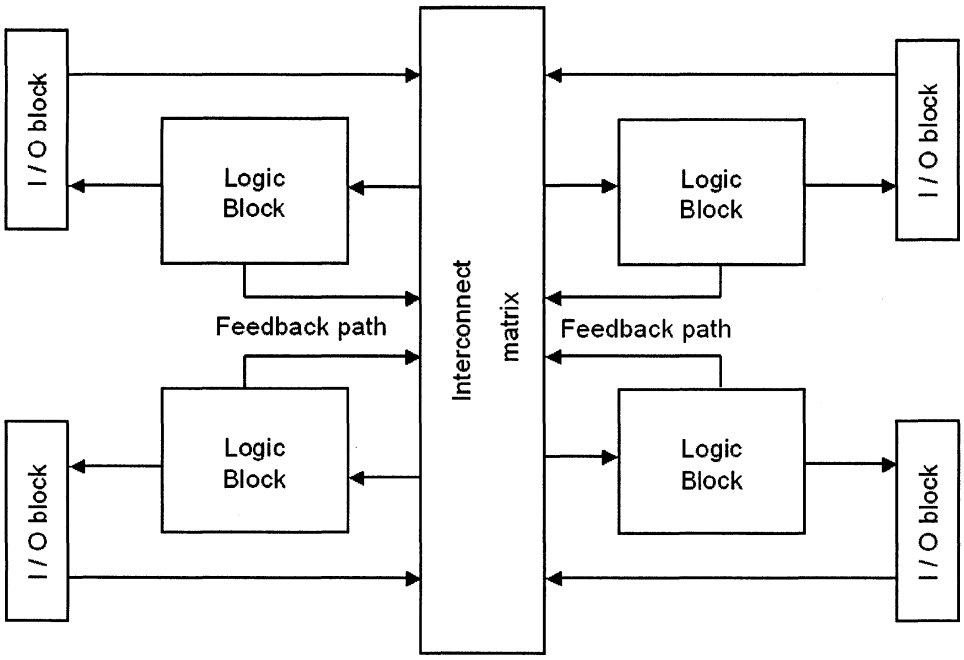


Figure 2-5 Structure of a typical CPLD

In order to realize the feedback paths and thus enable sequential designs, registers are added to the outputs of the OR gates of PLAs/PALs. With sequential devices, the outputs are determined both by the instantaneous inputs and the past states of the inputs. The registers require the connection of a clock signal in order to make them operate and the outputs will now only change when the clock signal is applied. Although an SPLD may be registered, its outputs can still be used as combinational outputs, as the connections can be programmed so that the register is

ignored and the output of the OR gate is connected directly to the output pin of the device.

A CPLD's routing is simpler with its less flexible internal architecture compared to that in FPGAs, but the CPLD's logic delay is predictable and its speed is usually faster than FPGAs. CPLDs are still sometimes used for simple applications like address decoding, but more often contain high-performance control-logic or complex finite state machines. Traditionally, CPLDs have been chosen over FPGAs whenever high-performance logic is required.

2.3.3 Architecture of FPGAs

Figure 2-6 shows the structure of a typical FPGA. It comprises an array of uncommitted circuit elements, also called configurable logic blocks (CLBs), and interconnect resources. Unlike CPLDs, CLBs in an FPGA are usually based on look-up tables (LUTs), which can only implement functions of a small number of inputs (generally LUTs with 4 inputs yield the smallest area [57]). The interconnect resources are more complicated in FPGAs compared to those in CPLDs; several kinds of vertical and horizontal tracks are differentiated by length and routing purpose.

CLB structure (specific to Xilinx): The basic building block of the Virtex-E CLB is the logic cell (LC). An LC includes a 4-input LUT, carry logic, and a storage element. The output from the function generator in each LC drives both the CLB output and the D input of the flip-flop. Each Virtex-E CLB contains four LCs, organized in two similar slices, as shown in Figure 2-7. Figure 2-8 shows a more detailed view of a single slice. In addition to the four basic LCs, the Virtex-E CLB contains logic that combines function generators to provide functions of five or six inputs [29].

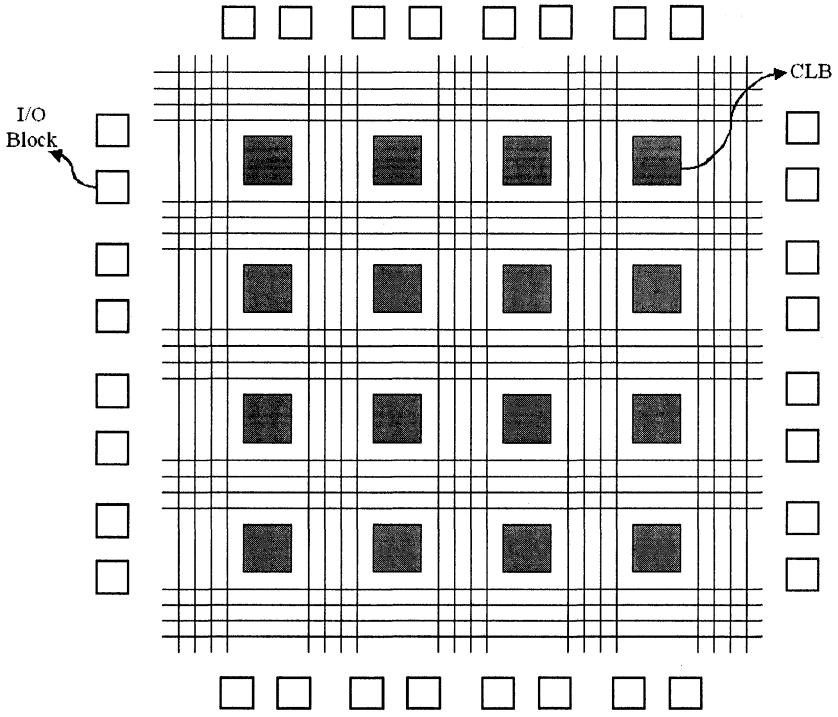


Figure 2-6 Island-style FPGA architecture

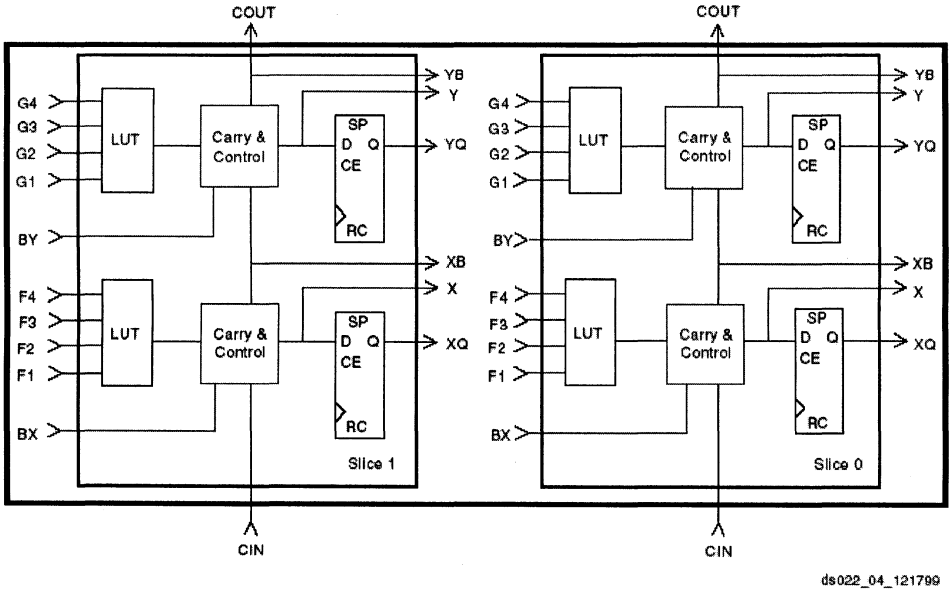


Figure 2-7 Two-slice Xilinx Virtex-E CLB (taken from [29], p.3)

The storage elements in the Virtex-E slice can be configured either as edge-triggered D-type flip-flops or as level-sensitive latches. The D inputs can be driven either by the function generators within the slice or directly from slice inputs, bypassing the function generators. In addition to Clock and Clock Enable signals, each slice has synchronous set and reset signals (SR and BY). SR forces a storage element into the initialization state specified for it in the configuration. BY forces it into the opposite state. Alternatively, these signals can be configured to operate asynchronously. All of the control signals are independently invertible, and are shared by the two flip-flops within the slice [29].

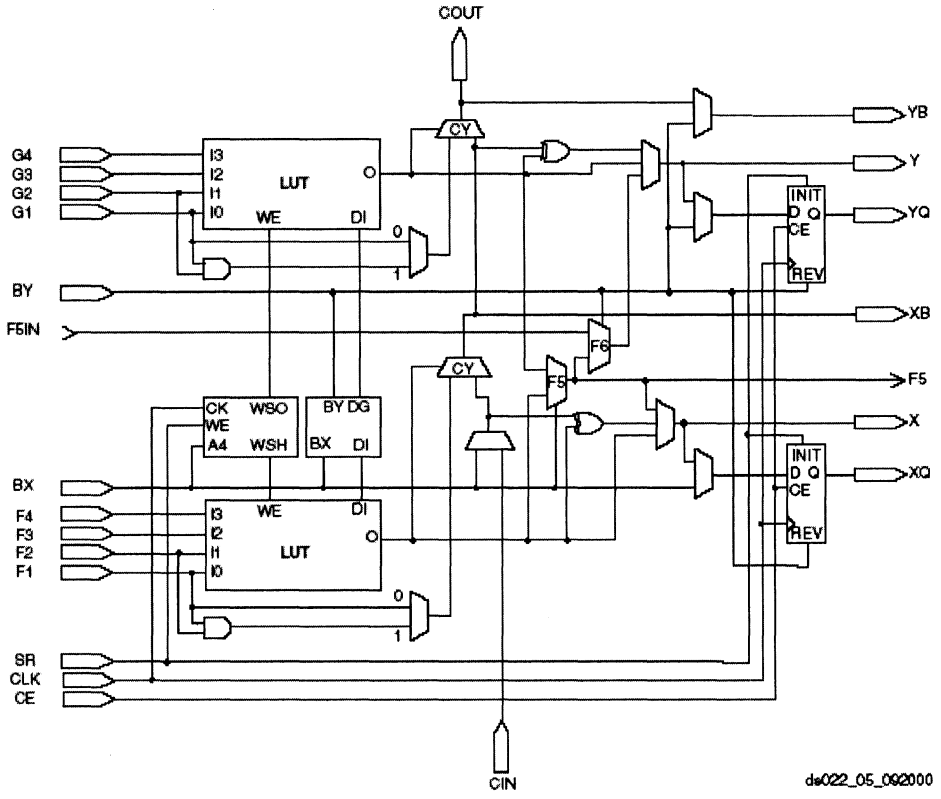


Figure 2-8 Detailed view of Virtex-E slice (taken from [29], p.3)

A LUT is a small memory. A memory with *N*-bit address lines and *M*-bit outputs has $M \times 2^N$ configuration bits. Based on the truth table of the required function, either logic "1" or "0" is stored in each memory cell. The address lines select one of 2^N sets of *M* configuration bits, and the outputs are set to the value of the selected *M* bits. Thus, the memory can be used to implement any *M* functions of *N* inputs, since each combination of these *N* inputs can produce a unique output value on each of the *M* outputs [18]. Figure 2-9 shows a three-input LUT with one output, where *L0~L7* are memory cells and *IN1~IN3* are inputs, which are used as control bits to the multiplexers [28].

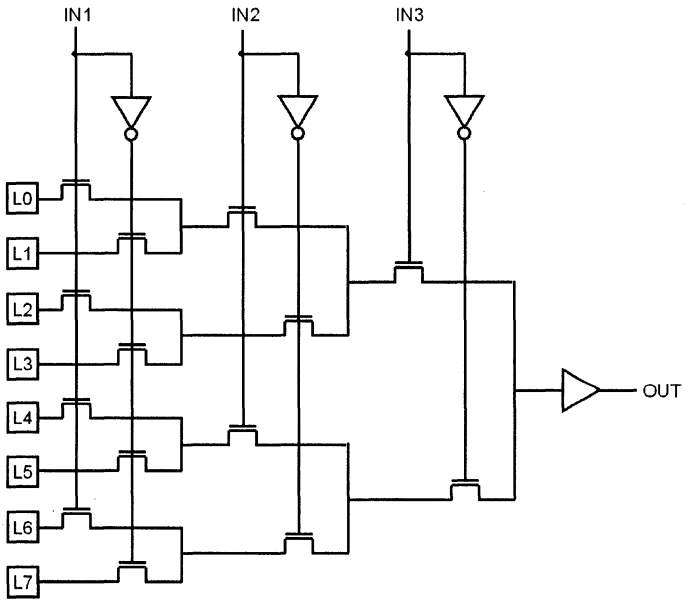


Figure 2-9 Structure of three-input LUT

Interconnect structure: In FPGAs, the routing structure is made up of several lengths of lines. Single-length lines travel the height of a single cell, where they then enter a switch matrix. The programmable switch matrix is shown in Figure 2-10 with details [29]. The switch matrix allows this signal to travel out vertically

and/or horizontally from the switch matrix. Multiple single-length lines can be cascaded together to travel longer distances. Double-length tracks are similar to single-length tracks, except that they travel the height of two cells before entering a switch matrix. Therefore, double-length lines are useful for longer-distance routing, without extra delay. There are also quad-length and long lines, which go through four cells and half of the chip height, respectively. They are used to accommodate longer signal routings [18].

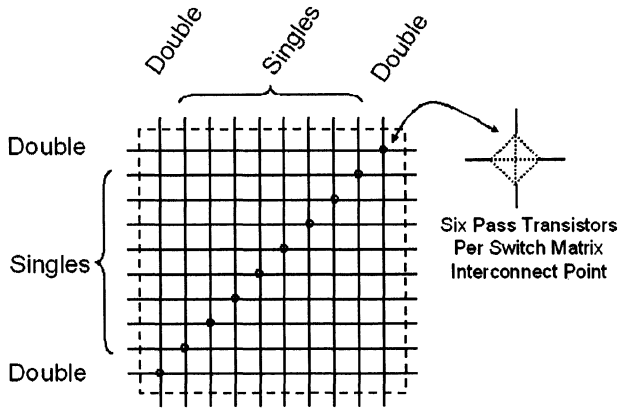


Figure 2-10 Programmable switch matrix (taken from [29], pp.4)

Due to the fine-grained LUT-based logic block and complicated interconnections, the propagation delays of FPGAs are routing dependent, while the architecture of an FPGA is much more flexible than that of a CPLD, which makes FPGAs better in register-heavy and pipelined applications. FPGAs have higher capacity and density but slower clock rates compared to CPLDs, and they are suitable for implementing larger circuits.

2.4 Summary

In this chapter, the fundamentals of programmable logic were introduced. The characteristics of different programming techniques, such as floating gate and SRAM, *etc.*, were discussed first. After that, the architectures and applications of SPLDs, CPLDs and FPGAs were depicted in detail. The pros and cons of each of the programmable logics were also analyzed.

After analyzing different types of programmable logic, what kind of programmable logic is suitable for SoC applications? There are several aspects we have to consider before we start the design. The first thing is speed. As we mentioned before, CPLDs provide better speed than FPGAs. Therefore, CPLDs are preferred when considering speed. The second thing is logic density. FPGAs have more favorable logic density than CPLDs. The third thing is power. Pesudo-nMOS-based CPLDs burn much higher static power than FPGAs. If we can combine the advantage of both FPGAs and CPLDs together with some structural innovations, we will develop an ideal programmable logic candidate for SoC applications.

Choosing the underlying logic family is the first and very important step in our design. Among all the logic families, dynamic logic appears to be the best choice. Dynamic logic has the best speed, and it is suitable for implementing wide NOR gates. In addition, unlike pseudo-nMOS, it doesn't burn static power. But dynamic logic requires a complicated clock network and levelized design, therefore a unidirectional routing structure may be required.

In next chapter, a new dynamic logic technique, output prediction logic (OPL), will be described. It is the fastest known logic technique, and has been chosen in the proposed OPL-PLC design.

Chapter 3 : Output Prediction Logic

3.1 Introduction

Dynamic circuit families such as domino [31] are commonly used in today's high-performance microprocessors [32] [33] [34] [35] for obtaining timing goals that are impossible using static CMOS circuits. There are several reasons why dynamic logic is faster than static CMOS, such as reduced input capacitance, lower switching thresholds, and circuit implementations that typically use fewer levels of logic due to the use of efficient complex gates. It has been shown that dynamic logic can be used to realize average speed improvements of about 60% over static CMOS for random logic blocks when using synthesis tools tailored specifically for dynamic logic [12] [36] [37].

However, existing dynamic circuits have some disadvantages. For example, domino logic must be mapped to a positive unate network, which usually requires duplication of logic. The noise sensitivity is also increased compared to static CMOS. Some performance must be sacrificed to get better noise margin.

Output prediction logic (OPL) is a new technique that can be applied to a variety of inverting logic families to increase speed while retaining the attributes of the underlying family. OPL relies on the alternating nature of logical output values for inverting gates on a critical path. That is, for any critical path, the logical output values of the gates along that path will be alternating ones and zeros. By correctly predicting exactly one half of the gate outputs, OPL obtains significant speedups (at least 2 times) over the underlying logic families (e.g. static CMOS, pseudo-nMOS and dynamic logic). All the details related to OPL are described in [37] [38] [39].

Since OPL was chosen as the underlying circuit technique in the proposed PLC architecture, before describing the proposed architecture, let's review some basic knowledge about OPL.

In this chapter, the brief introduction on applying the OPL technique for different logic families, such as static CMOS, pseudo-nMOS, dynamic logic and differential logic is presented. The comparison of speed and power consumption of different OPL family members with conventional static CMOS or differential domino is shown as well.

3.2 Output Prediction Logic Families

3.2.1 OPL Applied to Static CMOS

In static CMOS logic, every gate is an inverting logic gate. Because of this inverting property, every output on a critical path must fully transition from "0" to "1", or "1" to "0" in the worst case. This is shown in Figure 3-1, where we assume the primary input transitions high.

OPL greatly reduces the worst-case behavior of a critical path. In Figure 3-2, it is shown that OPL predicts that every inverting gate output on a critical path will be logic one after the transitions are completed. Since all gates are inverting, as in static CMOS, the OPL predictions will be correct exactly one-half the time. Every other gate will not have to make any transition. Therefore, OPL obtains significant speedups (at least 2 times) over the underlying logic families (e.g. static CMOS or dynamic logic).

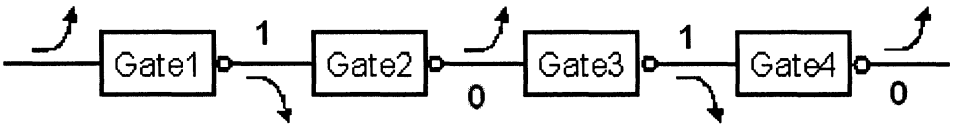


Figure 3-1 Static CMOS worst-case behavior

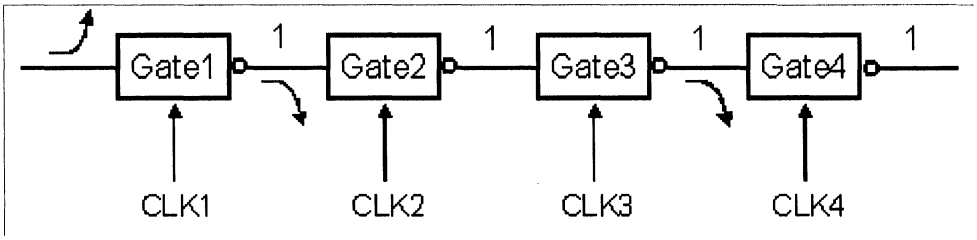


Figure 3-2 OPL worst-case behavior

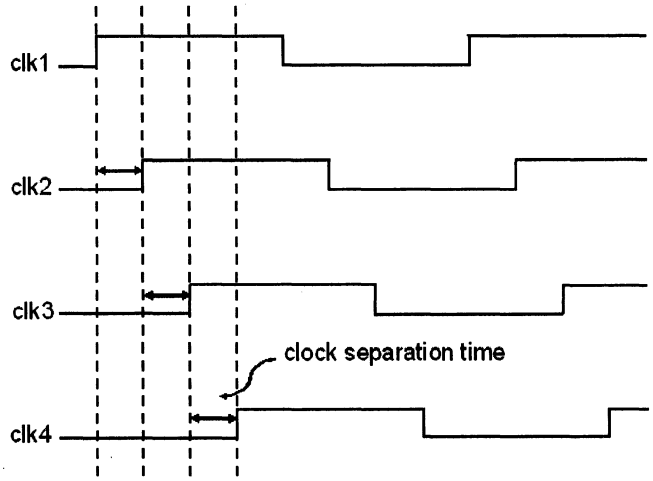


Figure 3-3 OPL clocking scheme

Of course, a one at every output (and therefore input) is not stable for an inverting gate. The one will erode in the latter gates of a circuit path. To prevent this erosion, each gate is disabled with a clocked footer device. The gates remain disabled until

their inputs are ready for evaluation. In this manner, predicted output values are maintained until new input values dictate otherwise. Successive clocks are delayed by a clock separation time as shown in Figure 3-3.

A tri-state, pre-charged-high static CMOS inverting gate is shown in Figure 3-4. When *CLK* is low, the gate is disabled, with the output being charged to a logic "1". When *CLK* goes high, the gate becomes a conventional static CMOS gate.

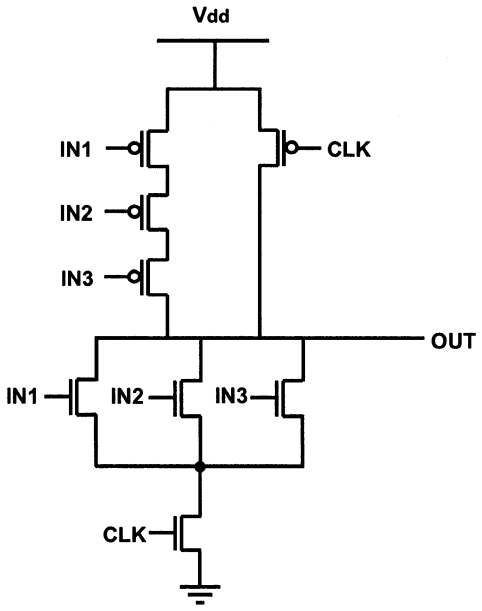


Figure 3-4 OPL-static CMOS NOR3

The speed of an OPL chain will be less than optimal if the clocks come too early or too late. Figure 3-5 shows a chain of three OPL-static inverters and Figure 3-6 shows the behavior of gate 2 as the clock arrival is varied. If the clock rising edge arrives early and at that time the inputs are not ready, the output glitch will be large, even all the way to zero, as in Figure 3-6(a). This will make the circuit throughput data-limited. In this way the precharged (predicted) values are completely lost and there is no reason to believe that any speedup over a conventional static CMOS

inverter chain would be achieved. If the clock rising edge arrives after the inputs to a gate have fully settled, the glitch will be very small, as shown Figure 3-6(c). However, by doing this, it becomes a clock-blocked circuit, in that the throughput of the circuit is limited by the clocks and not the data. Hence, the speedup achieved may not be impressive. Not surprisingly, there is an optimal point between the two extremes (fully clock-blocked and fully lost precharged values). As we see in Figure 3-6(b), the minimum delay occurs when a modest amount of glitch occurs. In the design process for an OPL circuit, we seek the optimal separation times to fully achieve the speed improvement possible with OPL logic.

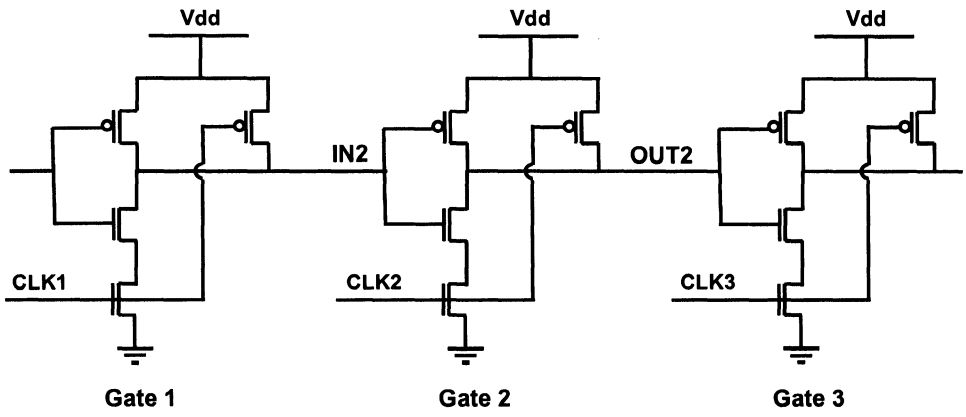
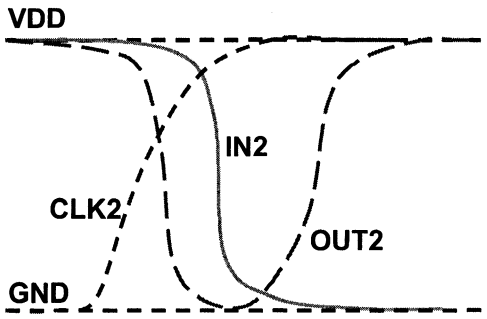
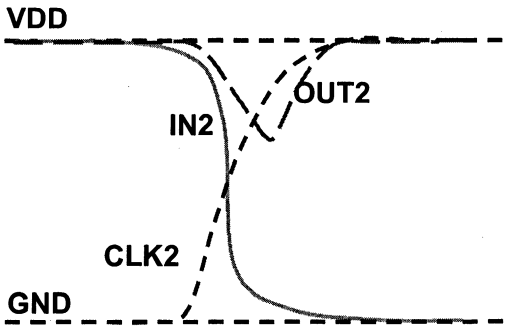


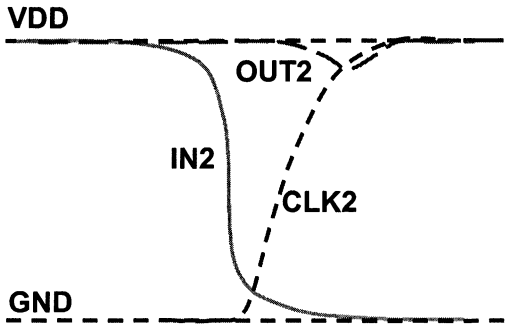
Figure 3-5 Chain of 3 OPL-static inverters



(a) Early clock rising edge



(b) Optimum clock rising edge



(c) Late clock rising edge

Figure 3-6 Dependency of gate output upon clock arrival for gate 2 of Figure 3-5

3.2.2 OPL Applied to Pseudo-nMOS and Dynamic Logic

The OPL technique can be applied to pseudo-nMOS and dynamic logic. A tri-state, pre-charged-high pseudo nMOS gate is shown in Figure 3-7. When *CLK* is low, the gate is disabled, with the output being charged to logic "1". When *CLK* goes high, it becomes a pseudo-nMOS gate. The pull-up device serves both as a precharge device and as a keeper. This pMOS device is properly sized to yield an appropriate output-low voltage. Note that the output-low voltage can be set closer to zero than for conventional pseudo-nMOS since pull-up delay is less of a concern, thus lowering static power dissipation (as will be shown later). The behavior of the gate is similar to that of pseudo-nMOS, thus it is suitable for implementing wide NOR gates [37].

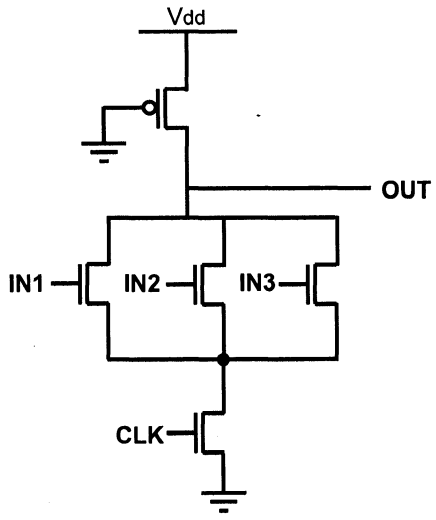


Figure 3-7 OPL-pseudo-nMOS NOR3

The OPL concept can also be applied to dynamic logic gates. Figure 3-8 shows an example of OPL-dynamic gates. It resembles a conventional domino gate, but with the output inverter missing. This is the fundamental difference between domino and

OPL – domino circuits rely upon precharged low inputs and monotonic operation in order to prevent evaluation until inputs have arrived. OPL circuits, on the other hand, are clock-blocking and therefore rely on the arrival of a clock edge to signal the beginning of evaluation. Same as the case in static logic, improvement of worst-case behavior of OPL is obtained because half the gates in critical path experience transitions. In addition, since there is no static inverter along signal path in OPL as opposed to that in domino, additional speedup is achieved. The keeper is used to recover the output glitch, but if the clock arrives too early, an output may glitch so much that the keeper shuts off, causing a failure. OPL-static and OPL-pseudo cannot fail in this manner, but for these two circuit types such large glitches invariably cause severe circuit delay degradation so it's not clear that this is a disadvantage for OPL-dynamic. Obviously it is important to plan clock separations and size the gates such that large glitches never occur, regardless of the OPL circuit type.

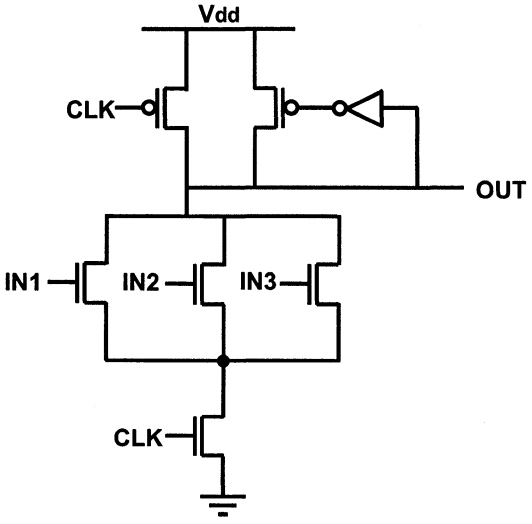


Figure 3-8 OPL-dynamic NOR3

3.2.3 OPL Applied to Differential Logic

Figure 3-9 shows an OPL-differential NAND3 gate. OPL-differential is dual rail. It takes both true and complement input signals and provides both true and complement output signals. When *CLK* is low, both *OUT* and $\overline{OUT}$ are precharged to logic "1". When *CLK* goes high, both *OUT* and $\overline{OUT}$ evaluate, that is, both of them experience the OPL glitch and then one of them ultimately goes low, while the other recovers. The speed advantage of OPL-differential over the single rail OPL families is due to the fact that the cross-coupled pMOS keepers are off when the evaluation begins and thus the keepers do not fight the discharging output.

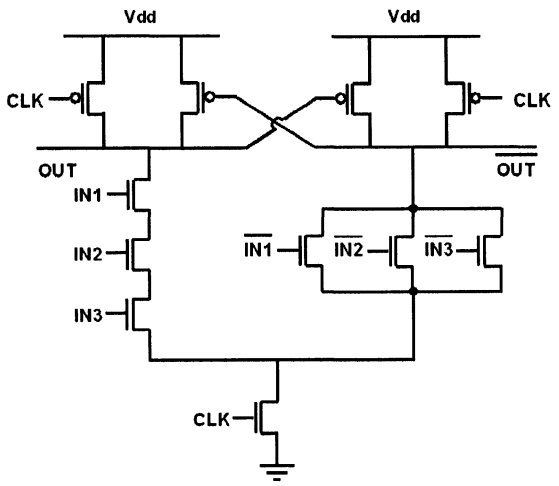


Figure 3-9 OPL-differential NAND3

In general, all OPL-differential gates can be implemented the same way as the NAND3 gate. On the left side we will have the pull down structure of the gate itself. On the right side, we will have the complement of the pull down structure of the gate with all the inputs complemented. We could actually use the OPL-differential NAND3 gate as a NOR3 gate if we switch the inputs of the two sides. However, the speed of the NOR3 gate will then be the same as a NAND3 gate.

Thus, the advantages of NOR gates over NAND gates in OPL will disappear. In addition, we will not be able to use wide fan-in NOR gates since we will have a long stack of series device on the NAND side [38].

A way to solve this problem is shown in Figure 3-10. On the left side, we have the regular NOR3 nMOS in parallel. On the right side, we have the same parallel structure with two nMOS dummy devices and the third nMOS device controlled by *OUT*. The dummy devices are placed there to provide equal loading on both sides. When *CLK* is low, both outputs are precharged high. When CLK goes high, both sides will initially discharge. If the inputs *IN1, IN2* and *IN3* are such that *OUT* is supposed to continue discharging, *OUT* will then turn on the keeper on the right side more and more. Eventually *OUT* will go to '0' and $\overline{OUT}$ will recover from the glitch. On the other hand, if the inputs are all going low, the discharge of *OUT* will be slow. The right hand side will then see a high input and will have a stronger discharge than the left side. $\overline{OUT}$ will discharge all the way to '0' and turn on the keeper on the left; *OUT* will then recover back to Vdd [38].

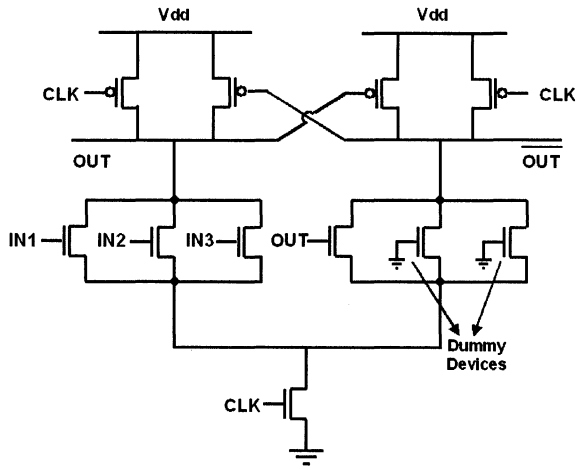


Figure 3-10 OPL-differential NOR3 gate

3.3 OPL Performance

3.3.1 Speed Performance

To determine the speed possible with OPL, critical paths consisting of 10 identical gates were simulated. Each gate in the path has a fan-out of four. The delay results in Table 3-1 compare static CMOS, OPL-static, OPL-pseudo-nMOS (OPL-pseudo) and OPL-dynamic using nominal simulation parameters for the TSMC 0.25μm/2.5V CMOS process (FO4 static inverter delay is 162ps). The delay results in Table 3-2 compare static CMOS, differential-domino and OPL-differential using nominal simulation parameters for the TSMC 0.18μm/1.8V CMOS process (FO4 static inverter delay is 84ps).

On average, OPL-static chains are 2.6 times faster than static CMOS and OPL-pseudo chains are 3 times faster than static CMOS. In fact, OPL-pseudo NOR3 chains are 5.4 times faster than static CMOS, and INV chains are 3.9 times faster. OPL-dynamic has about the same performance as OPL-pseudo. OPL-differential exhibits the best speed improvement among all the OPL family members and is slightly more than twice as fast as differential domino.

Table 3-1 Delays for chains of 10 gates (FO of 4, 0.25μm/2.5V) [37]

Chain Type	Static CMOS (ns)	OPL-Static (ns)	OPL-Pseudo (ns)	OPL-Dynamic (ns)
INV	1.62 (1.0)	0.43 (3.77)	0.42 (3.86)	0.43 (3.77)
NOR3	3.83 (1.0)	1.34 (2.86)	0.71 (5.39)	0.76 (5.04)
NAND2	2.45 (1.0)	0.94 (2.61)	0.93 (2.63)	1.02 (2.40)
NAND3	3.32 (1.0)	1.44 (2.31)	1.54 (2.16)	1.54 (2.16)
NAND4	4.24 (1.0)	1.97 (2.15)	2.16 (1.96)	2.15 (1.97)
AOI22	4.75 (1.0)	2.13 (2.23)	1.81 (2.62)	1.80 (2.64)
AOI222	6.75 (1.0)	3.04 (2.22)	2.63 (2.57)	2.49 (2.71)
Average	(1.0)	(2.59)	(3.03)	(2.96)

Table 3-2 Delays for chains of 10 gates (FO of 4, 0.18μm/1.8V) [37]

Chain Type	Static CMOS (ns)	Diff. Domino (ns)	OPL-Diff. (ns)
INV	0.84 (1.0)	0.62 (1.35)	0.16 (5.25)
NOR2	1.26 (1.0)	0.66 (1.91)	0.25 (5.04)
NOR3	1.59 (1.0)	0.74 (2.15)	0.30 (5.30)
NOR4	2.34 (1.0)	0.89 (2.63)	0.34 (6.88)
NAND2	1.02 (1.0)	0.66 (1.55)	0.30 (3.40)
NAND3	1.38 (1.0)	0.80 (1.73)	0.45 (3.07)
NAND4	1.48 (1.0)	0.89 (1.66)	0.52 (2.85)
AOI21	1.30 (1.0)	0.72 (1.81)	0.35 (3.71)
AOI22	1.74 (1.0)	0.82 (2.12)	0.33 (5.27)
AOI222	2.95 (1.0)	1.01 (2.92)	0.54 (5.46)
AOI31	1.76 (1.0)	0.83 (2.12)	0.52 (3.38)
AOI33	2.60 (1.0)	1.00 (2.60)	0.50 (5.20)
AOI333	4.00 (1.0)	1.19 (3.36)	0.59 (6.78)
AOI321	2.43 (1.0)	0.91 (2.67)	0.54 (4.50)
Average	(1.0)	(2.18)	(4.72)

3.3.2 Power Consumption

The total energy consumptions for static CMOS, OPL-static and OPL-pseudo (including precharging energy), as well as OPL-dynamic were also compared.

Table 3-3 Energy consumption for chains of 10 identical gates [37]

Chain Type	Static CMOS (pJ)	OPL Static (pJ)	OPL Pseudo (pJ)	OPL Dynamic (pJ)
INV	2.00 (1.0)	3.8 (1.90)	4.97 (2.49)	4.41 (2.21)
NOR3	3.19 (1.0)	4.45 (1.39)	6.07 (1.90)	4.47 (1.40)
NAND2	3.83 (1.0)	5.00 (1.31)	8.39 (2.19)	5.60 (1.46)
NAND3	6.23 (1.0)	6.66 (1.07)	12.7 (2.04)	7.51 (1.21)
NAND4	8.65 (1.0)	12.7 (1.47)	19.3 (2.23)	10.0 (1.16)
AOI22	6.13 (1.0)	6.31 (1.03)	12.8 (2.09)	7.01 (1.14)
AOI222	7.08 (1.0)	7.70 (1.09)	16.7 (2.36)	8.09 (1.14)
Average	(1.0)	(1.32)	(2.19)	(1.39)

Table 3-3 shows that the average energy consumption of OPL-static gates is around 1.3 times that of static CMOS. This energy includes the energy required to generate the clocks, precharge outputs, and any energy consumed during evaluation [37]. OPL-pseudo gates consume about 2.2 times as much energy as static CMOS gates, on average. OPL-dynamic consumes less energy than OPL-pseudo but offers similar speed.

3.4 Summary

In this chapter, we briefly introduced OPL, a technique that can be applied to conventional CMOS logic families to obtain considerable speedups. Speedups of 2 to 3 times over conventional static CMOS were demonstrated for a variety of circuits in chains of gates.

There are several candidates in OPL family. As will be discussed in next chapter, OPL-dynamic was chosen in the proposed PLC architecture due to its high speed, relative low power and small area.

Chapter 4 : OPL-PLC Architecture

4.1 Introduction

In this chapter, we describe the proposed PLC architecture, namely OPL-PLC. The architecture combines OPL-dynamic logic with several structural innovations — a flexible product-term-based logic block, mixed logic/routing structures, wired-OR structures and a compact unidirectional routing scheme to obtain both high speed and good flexibility. It also utilizes SRAM cells as the storage element for programming so that it can be fully integrated with other components on the SoC, allowing programming changes to be made on the fly.

Standard-CMOS-process-based design: Advanced fabrication processes, such as the SiGe BiCMOS process, have been used in programmable logic design to achieve giga-hertz speed performance [47]. However, from an economical point of view, using a standard CMOS process is preferable. In addition, CMOS has higher integration density, which is more suitable for SoC. Based on these considerations, we aimed to develop a new architecture to obtain both high-speed and good-flexibility, as well as to keep the manufacturing cost low.

Product-term logic resources: CPLDs exhibit higher speeds than LUT-based FPGAs for two reasons. First, they use the equivalent of wide gates to reduce the number of logic levels. Second, CPLDs use a pseudo-nMOS style to implement these wide gates. Pseudo-nMOS designs can be fast because only the pull-down network must be implemented, thereby reducing transistor stack height. However, their static power consumption is high due to the always-on pMOS pull-up device.

Some researchers have combined the characteristics of FPGAs and CPLDs together to achieve better performance. In [43], it is claimed that compared to LUT-based

design, the PLA-based FPGA uses fewer wiring tracks and levels of logic, which leads to faster speed. In [44], [45] convincing evidence is presented that product-term-based programmable logic has better density and speed characteristics compared to LUT-based FPGAs.

The use of product-term-based structure in OPL-PLC results in mapping simplicity, too. In addition, product terms can be implemented efficiently using an OR structure that is well suited to OPL. Unlike [43], [44], [45], however, OPL-PLC does not use a conventional product-term-based structure where the maximum numbers of inputs, product terms and outputs are fixed. Instead, the structures are organized to accommodate varying numbers of inputs, product terms and outputs efficiently. As we mentioned in chapter 2, a three-level-structure exhibits better speed and logic density. In OPL-PLC, an AND/NOR-AND-NOR structure is utilized, allowing product-term-based structures to be more efficiently packed and the logic density to be better than conventional CPLDs.

Mixed logic/routing structure: Multiplexers are necessary in programmable logic for routing signals. But they increase both the area and delay of conventional programmable logic. OPL-PLC decreases routing area and delay by combining both logic and multiplexers together.

Wired-NOR structures: Wired-NOR structures are extensively used in OPL-PLC. The wires in wired-OR structures may be quite long. Being a "dispersed" logic resource, a wired-OR is able to perform logic on widely separated inputs, while at the same time providing an output to possibly several widely separated destinations. The wired-NOR eliminates the need for wiring signals to a "localized" logic resource. An additional advantage of the wired-or structure is that it can also serve as a routing resource.

Unidirectional routing architecture: Routing resources occupy a large portion of area and delay in circuits implemented in FPGAs. In good part the routing resources are designed for good flexibility. On the minus side, this flexibility increases the distance between different logic blocks as well as the routing resources, resulting in more capacitive load and extra delay. Therefore, speed performance is degraded.

A unidirectional architecture, where data flow is in one direction, clearly has some advantages in terms of routing simplicity. A unidirectional architecture was used in Triptych [46], where it was folded to obtain bidirectionality. OPL-PLC adopts a unidirectional architecture for the same reason, as well as to simplify OPL clock generation and distribution.

Dynamic-logic-based design: Dynamic logic was adopted in the proposed PLC architecture for the purpose of high speed. OPL-dynamic was chosen as the underlying circuit technique based on the following comparisons.

As introduced before, there are OPL-static, OPL-pseudo, OPL-differential and OPL-dynamic in OPL family. First, let's consider OPL-static. It obtains about 2.6 times speed improvement compared to static CMOS, and it has the best noise immunity and lowest power consumption among all the OPL candidates. But it is not suitable for implementing wide NOR gates due to its very complicated keeper, while the wide NOR will be extensively used in the proposed programmable logic core. So OPL-static is not the best choice for PLC.

Then, let's take a look at OPL-pseudo. It exhibits the best speed performance among the three single-rail styles, and is suitable for implementing wide NOR gate. However, due to the grounded pMOS, its noise margin is reduced and extra power is consumed. In programmable logic design, besides speed, power consumption is also a critical concern. Therefore, OPL-pseudo is not the best choice, either.

The next candidate is OPL-differential. It achieves the highest speed among all the OPL family members. It is also at least 5 times faster for NOR gate implementations compared to static CMOS. However, each OPL-differential gate generates both polarities of the output, which is not necessary in the proposed architecture. Also, the dual rail structure requires more area and more complex routing, which is not desired.

The last candidate is OPL-dynamic. Its speed is comparable with OPL-pseudo, and its power is just a little bit higher than OPL-static. From the speed and power point of view, OPL-dynamic is the best choice among all the single-rail OPL styles. Meanwhile, OPL-dynamic is well suitable for implementing wide NOR gates, which matches the requirement of OPL-PLC very well.

From the above discussion, we found that OPL-dynamic is the best choice among OPL families. We will need to assign clock separations and size the gates appropriately to achieve the desired speed. Also, crosstalk on an output node may increase the glitch, so this must be given due attention.

An OPL-dynamic gate resembles a conventional domino gate, but with the output inverter missing. This is the fundamental difference between domino and OPL – domino circuits rely upon precharged low inputs and monotonic operation in order to prevent evaluation until inputs have arrived. OPL circuits, on the other hand, are clock-blocking and therefore rely on the arrival of a clock edge to signal the beginning of evaluation. Same as the case in static logic, improvement of worst-case behavior of OPL is obtained because half the gates in critical path experience transitions. While Domino-based circuits are positive unate; under the worst-case scenario, every gate on the critical path needs to make a transition during evaluation. In addition, since there is no static inverter along signal path in OPL as opposed to that in domino, additional speedup is achieved.

In contrast to OPL circuits, domino circuits generally require logic duplication to map to positive unate functions. Considering that the proposed OPL-PLC architecture is based on product-term-based structures, there also exist two problems: 1) domino cannot implement the optional-inversion function, which is necessary and extensively used in OPL-PLC, and 2) in order to improve speed, wide NOR gates are typically used to implement the AND-plane by changing the polarity of the inputs. In domino the polarity of the inputs cannot be changed and the NOR gate cannot be implemented.

Based on the comparison between OPL-dynamic and domino, we draw the conclusion that OPL-dynamic is able to provide not only better speed but it is also necessary to implement the logic functions needed for the proposed OPL-PLC architecture. OPL-dynamic logic perfectly matches the criteria of programmable logic for SoC applications we mentioned in the previous chapter: it has the best speed in single-rail styles and is suitable for implementing wide NOR gates. In addition, unlike pseudo-nMOS, it doesn't burn static power.

Inherent-pipeline capability: Besides the superior speed performance, OPL-dynamic logic provides built-in pipeline capability without requiring any register.

There are several other pipelined FPGAs developed previously. In [51], the authors proposed a novel high-throughput FPGA architecture, *PITIA*, which uses the concept of Wave Steering and pipelined interconnects to achieve throughputs in excess of 500 MHz in a 0.25μm 2.5V CMOS technology for the implementations of datapath circuits. Wave Steering is a design methodology that realizes high throughput circuits by embedding layout friendly synthesized structures in silicon [66], [67]. Designs with predominantly local interconnects are the best fit in the fine-grained PITIA structure. Experimental results showed that, in PITIA, random designs can operate at speeds of 250MHz, and the locally interconnected designs

can operate at 625 MHz. Locally interconnected designs here mean the designs in which connections between different blocks are constrained to be local (not necessarily next neighbor). But PITIA only targets at the throughput-intensive applications. If the application requires both high throughput and low latency, it is not an ideal candidate. For example, in PITIA, the latency for a 4×4 pipelined multiplier is 11.2ns. While in OPL-PLC, only 2.57ns is required to implement the same circuit. Since OPL-PLC is implemented in a 0.18μm 1.8V CMOS process, after normalization, the equivalent OPL-PLC delay in a 0.25μm 2.5V CMOS process is 5.14ns. The latency of the 4×4 pipelined multiplier implemented in OPL-PLC is about 2.2 times faster than that implemented in PITIA. This supports that OPL-PLC has broader applications compared to PITIA in terms of low latency.

Retiming is a very popular scheme for obtaining high throughput frequency in programmable logic design. As a powerful logic optimization technique for synchronous circuits, retiming is based upon the property that flip flops can be taken from the outputs of gates and moved to their inputs, or vice versa, without changing the perceived behavior of the circuit.

In [63], they developed a registered routing switch in FPGAs. It utilizes the concept of retiming to reschedule circuit operations in such a way that a signal may use multiple clock cycles to traverse a long route, rather than requiring a single long clock period. The pipelined frequency is increased by adding registered routing switches into the long interconnect wires. In the proposed FPGA structure, the number of registered routing switches added in the interconnects is optimized to provide good speedup without excessive area penalties. There is an extra input register per LUT in the proposed structure to decrease the number of routing resources. The experimental results demonstrated that the registered routing structure obtained 12% ~25% speedup with area penalty of more than 10%, which is due to those switches.

In [62], they introduced a high-speed hierarchical synchronous reconfigurable array, namely HSRA. To achieve high frequency operation, sequential retiming was adopted in HSRA, too. Like the structure in [63], they added mandatory pipeline registers to break both logic and routing delays up into clock cycle segments. Unlike [63], before adding those registers, they set a single cycle time and define everything in terms of that cycle time. This device can work under a 250MHz operation frequency in a 0.4μm process, where common FPGAs only work in the range of 25-70MHz. However, due to the existence of registers in HSRA, in the 4ns period, 1ns is spent by register clock-to-Q delay, which is 33% of the data cycle latency. By obtaining higher frequency, HSRA sacrifices the circuit latency.

The authors in [64] also presented a novel interconnect structure, called "corner-turn", to define a fixed-frequency FPGA architecture, synchronous and flexible reconfigurable array (SFRA). The corner-turn switch-box maintains the any-to-any connectivity of a crossbar while sacrificing the capacity that can be routed. Compared to the full crossbar switch-box, it saves number of switches inside the switch-box, but only a limited number of total signals can be routed between the two channels. Corn-turn interconnects have the advantages such as fast routing, register savings for pipelined switch points, tolerance for defects, etc. In SFRA, the clock rate is fixed to 300MHz, regardless of the configuration mapped onto it. The frequency of SFRA is 3~8 time faster than Xilinx Virtex unoptimized design. However, when normalized for area, the speedup of their architecture is reduced to 1~2.2 times.

Similar to those pipelined FPGAs introduced above, OPL-PLC is suitable for implementing high frequency pipelined applications. Unlike them, OPL-PLC doesn't require extra registers to pipeline the design, because it is based on a dynamic logic family with every gate controlled by a clock. Therefore, there are no extra penalties in both area and latency delay due to the introduction of registers. Of

course, the clock network is more complicated in OPL-PLC than those in static logic based designs, and it needs to be optimized to reduce area and power.

In this chapter, first of all we will describe the details of the proposed OPL-PLC architecture. Then we will discuss the clock distribution scheme in OPL-PLC, an important issue for OPL circuit design. After that we will show the circuit mapping results. And finally we will analyze the power dissipation of OPL-PLC.

4.2 OPL-PLC Architecture

4.2.1 Design Flow

As we introduced in the previous section, OPL-PLC is a brand-new flexible architecture with high performance. The structure flexibility and speed performance were optimized by following a systematic design flow.

Figure 4-1 shows the design flow of designing the OPL-PLC from functionality verification to final chip measurement. The functionality of the proposed architecture was first verified with gate level Verilog simulation. There were several iterations to improve the architecture flexibility. After the initial conception of the architecture was decided, the schematic was built in Cadence and simulation was performed using Hsim [53]. In order to get more accurate sizing, wire load capacitances were roughly estimated and added into the schematic. After sizing optimization, the layout was implemented. Next, the layout was extracted and post-layout simulation was performed. Then we went back to the schematic and added more accurate parasitic wire loads. After several iterations in steps 2, 3 and 4, the circuit performance was optimized. Finally the circuit was fabricated and OPL-PLC performance was measured on the real chip.

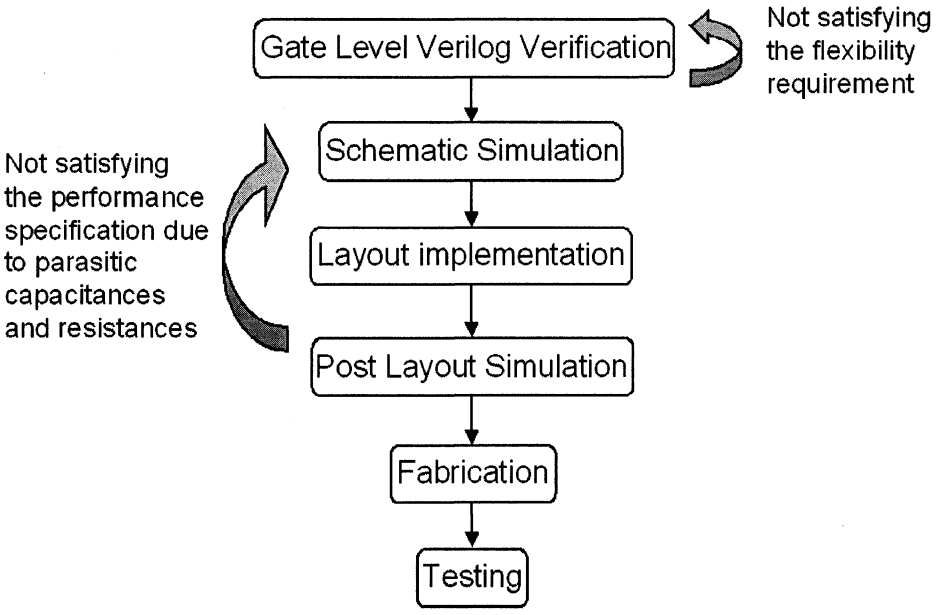


Figure 4-1 Design flow of OPL-PLC

4.2.2 High Level Architecture

The top-level architecture of OPL-PLC, shown in Figure 4-2, comprises two parts: hybrid logic-routing blocks (HLRBs) and long tracks. The HLRB is the basic configurable logic block providing both the functional units and the local routing. The long tracks implement a wired-NOR4 as well as provide global routing.

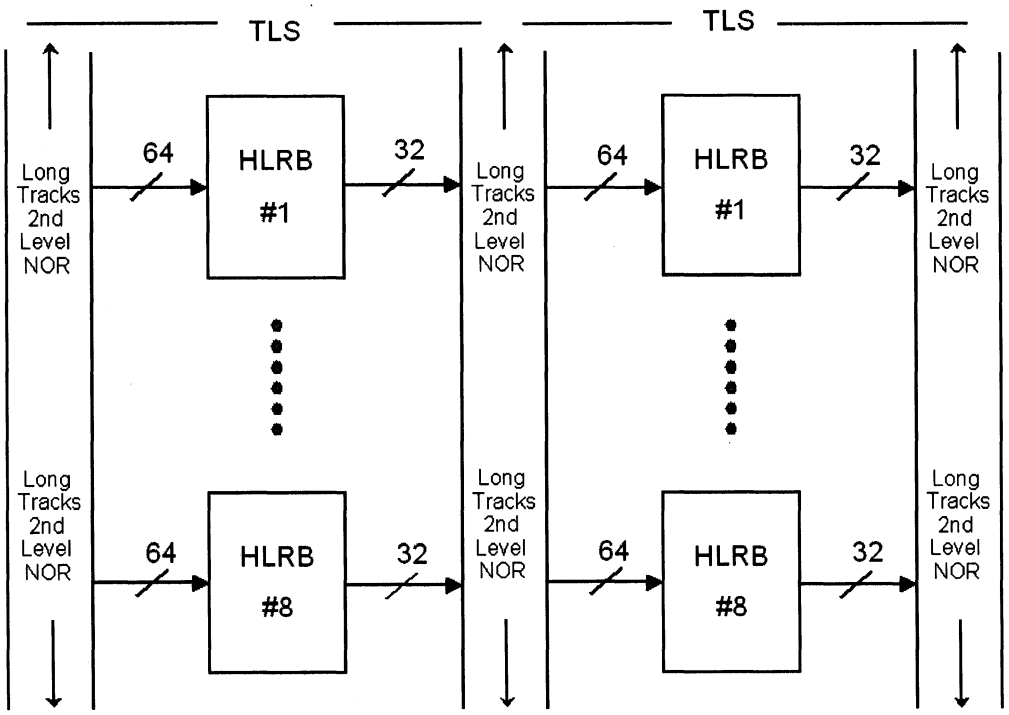


Figure 4-2 Top level architecture of OPL-PLC (two TLSs in series)

The OPL-PLC includes several three-level structures (TLSs) in series (as shown in Figure 4-2). The TLS is so-named because it has the three-level AND/NOR-AND-NOR structure. For each TLS, there are eight HLRBs in parallel. The HLRB is implemented in a PLA-like structure. Each HLRB can access 64 different inputs, which is the total number of signals that one group of long tracks can provide. Each HLRB can generate at most 8 different outputs, which can be NOR'd with outputs of other HLRBs on up to 32 tracks. The dataflow is unidirectional. By connecting four different outputs together, long tracks implement a wired-NOR4. Also, the long tracks can be used either to pass output signals from the current TLS or to provide primary input signals to the next TLS.

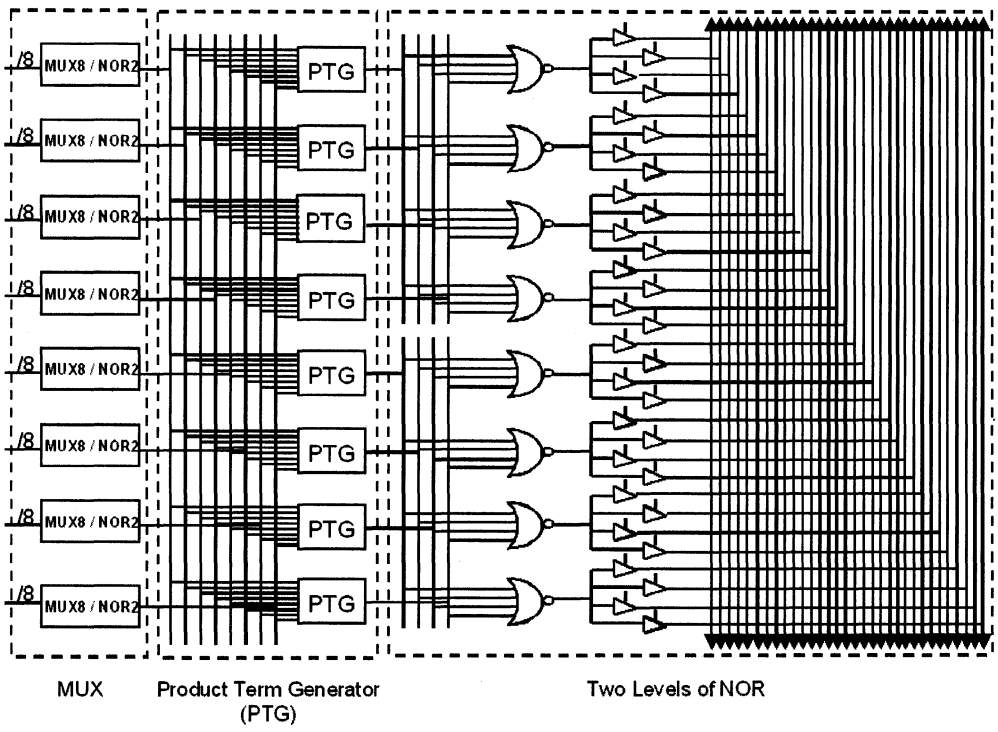


Figure 4-3 Structure of Hybrid Logic-Routing Block (HLRB)

Figure 4-3 shows the structure of an HLRB. It includes three main parts: the MUX, the Product Term Generator (PTG) and the wired NOR (two levels of NOR). The MUX is used to select from inputs; the PTG is equivalent to the AND-plane in a PLA structure; and the wired NOR is equivalent to the OR-plane in a PLA structure. All the three parts are programmable.

The first set of local tracks exists in front of the product term generators (PTGs). There are eight short vertical tracks, whose lengths are equal to the height of eight rows of logic. Each track passes one selected input, and after choosing the inputs from the local tracks, eight product terms are generated. Figure 4-4 shows the detail of the PTG. Those product terms share the same inputs, and by programming the control signals, each PTG can have up to eight inputs. As we will describe in the

gate level structure, MUX8/NOR2 can be used as a two-input logic gate. Combining with those MUX8/NOR2s, up to a 16-input AND can be implemented in this structure.

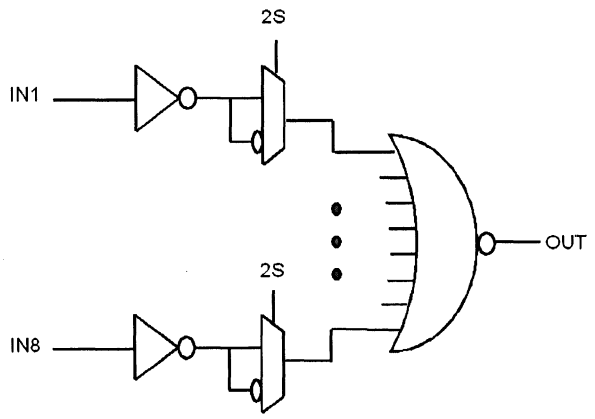


Figure 4-4 Detail of the Product-Term Generator (PTG)

The outputs of PTGs are then programmably NOR'd together in two levels. In the first-level NOR, those eight PTGs are separated into two groups. Each first level NOR4 can access one group of PTGs out of two. The second set of local tracks is used to distribute the outputs of PTGs; the track length equals the height of four rows. Four sets of drivers programmably NOR the outputs of the first-level NORs to give the second-level NOR. The outputs of those first-level NORs are from four different HLRBs and are combined together by the global tracks. Although this operation can effectively NOR together a larger number of first-level (NOR) outputs depending upon the number of drivers connected to a given long track, we limited it to four drivers in order to limit the loading on a long track. Through the two levels of NOR4s, a total of 16 PTGs may be NOR'd together. Thus, the largest function this structure can implement in one TLS is a 16-bit AND followed by a 16-bit NOR.

This two-level NOR structure enables OPL-PLC to flexibly implement either large or small functions. When a small NOR function needs to be implemented (e.g., a NOR4), the output of the first level NOR4 can be passed directly to the final long tracks, without being combined with the other three NOR4s. Thus, the two levels of NOR4 can be used separately as four distinct NOR4 functions. This structural improvement avoids the low logic density of the traditional PLA structure.

There are some similarities between OPL-PLC and the architecture described in [44], such as the unidirectional routing fabric and product-term-based blocks. Compared to the LUT-based FPGAs, product-term-based structure exhibits superior speed performance, which is attractive for bridging the speed gap between fixed logic and programmable logic. However, conventional PLA, as used in [44], has lower logic density. In OPL-PLC, this is resolved through structural innovations and the favorable speed is kept meanwhile. For example, in OPL-PLC, the MUX portion can also be used to implement a small AND function. Consequently, we can view the AND plane in HLRB as being two levels of AND, where the MUX is the first level and the PTG is the second level. In addition, OPL-PLC has two compact levels of NORs. Both of them considerably increase logic density. In [44], it is claimed that product-term based structures require fewer routing multiplexers than LUT-based structures, resulting in smaller area. In OPL-PLC, some of the multiplexers are combined together with the logic gates. We believe this will reduce the area further.

This structure also has similarity to the *k/m* macrocells described in [45] (*k* represents the number of inputs for each macrocell, *m* represents the number of product terms generated in each macrocell), but allows considerably more flexibility. In [45], the number of unique inputs required for an output is restricted to *k*. In the OPL-PLC, the limitation of *k* inputs applies only to a single product term. Also, unlike *k/m* macrocells, the size of the OR plane is not fixed to *m* terms.

In the OPL-PLC structure, the OR plane size depends on the number of product terms that are actually required to compute the output. Each *k/m* macrocell has only one output, while in OPL-PLC, each OR function has up to 4 outputs.

4.2.3 Gate Level Structure

Now let's take a closer look at the gate level implementation of OPL-PLC. Figure 4-5 illustrates the critical path through a single TLS. There are eight gates in series, with six different clock phases. In OPL circuits, two gates can share the same clock phase (as illustrated in Figure 4-5 for CLK3 and CLK5) as long as the first (driving) gate of the pair is much faster than the second gate. In CLK5, although NOR4_EN is more complicated than INV_EN, we made NOR4_EN has fanout less than 2 but INV_EN has fan out of 4. Therefore, NOR4_EN is faster than INV_EN, and it satisfies the criteria of combining two gates in one clock phase.

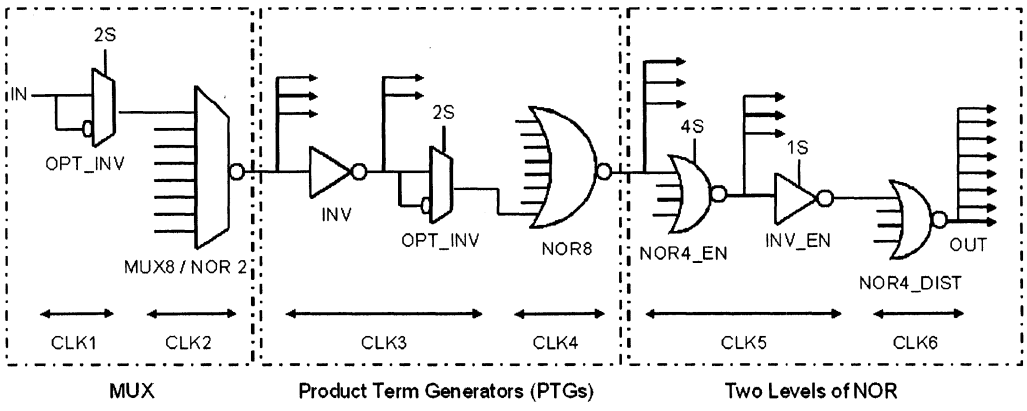


Figure 4-5 Critical path through a TLS

4.2.3.1 MUX

In the MUX part, there are two gates: OPT_INV and MUX8/NOR2. OPT_INV, shown in Figure 4-6, is an optional inverter, acting as a tri-state gate. Table 4-1 lists the programming conditions for each state of OPT_INV. State-1 chooses the same polarity of the input; state-2 chooses the opposite polarity of the input and state-3 sets the output to low. In the first two states, only nMOS pass transistors are required. This is because the output is precharged high and therefore the only transition is the passing of logic low to the output. In state-3, by setting S1 to logic low, the two pass transistors are shut off and the output is disabled to low, independent of the clock signals.

Table 4-1 State conditions of OPT_INV

	S0&S1	S0_b&S1	S1_b	Output
State 1	1	0	0	In
State 2	0	1	0	In_b
State 3	0	0	1	Disabled (logic low)

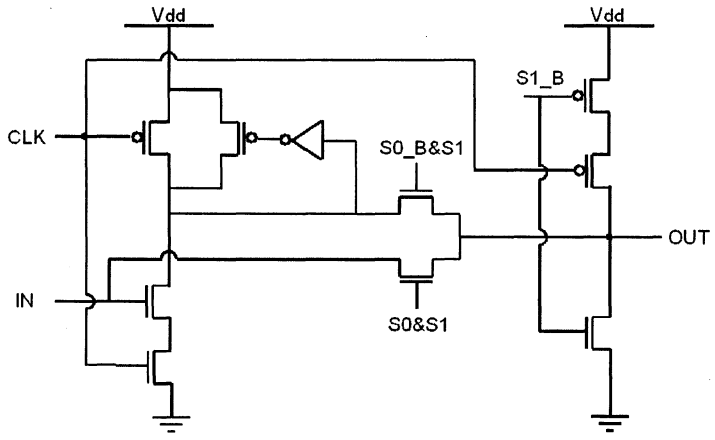


Figure 4-6 Schematic of OPT_INV

The outputs of eight OPT_INVs are fed into a MUX8/NOR2 (schematic is shown in Figure 4-7), implemented as an OPL dynamic NOR8 gate since the control signals have been pushed back to the former OPT_INV. For example, there are 8 OPT_INVs in front of a MUX8/NOR2. If only one input is enabled in OPT_INV, then all the others are disabled by setting *S1_B* to logic '1', and the circuit acts as a MUX 8-1. The MUX8/NOR2 then sees only one input varying and the remainder stay low.

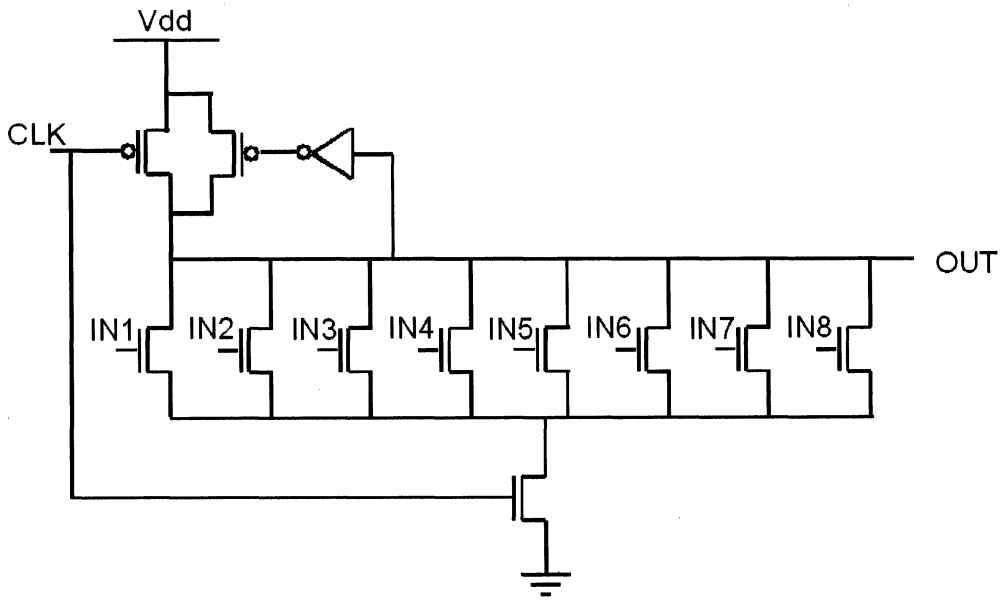


Figure 4-7 Schematic of MUX8/NOR2

This two-stage OPL-MUX has a significant speed advantage over a single-stage multiplexer. In the single-stage multiplexer, the enabling devices are stacked with the input transistors, which increases the transistor size as well as the delay. In OPL-MUX, by pushing the enable devices back to the former gate, the enabling device is no longer on the critical path, which is part of the reason OPL-MUX is faster.

By allowing up to two OPT_INV outputs to be enabled at the same time, the MUX8/NOR2 can also be used to implement a NOR2 or an AND2. For example, if two inputs A and B are enabled to pass to the MUX8/NOR2, the MUX8/NOR2 acts as a NOR2. The output is $F = \overline{A+B}$. However if the inversion of A and B are chosen in the OPT_INV, the MUX/NOR2 acts as an AND2. The output is $F = \overline{\overline{A}+\overline{B}} = A \cdot B$. Since NOR2 and AND2 are used quite often in arithmetic logic blocks, OPL-PLC implements small functions efficiently compared to conventional PLAs. The mixed logic and multiplexer structure also decreases routing area.

4.2.3.2 Product-Term Generator (PTG)

There are three gates in the PTG part, where the same structure as for the MUX is used to generate the product terms. The first gate is just an OPL-dynamic inverter, which is shown in Figure 4-8. It is used as buffer to drive the following two optional inverters for better speed.

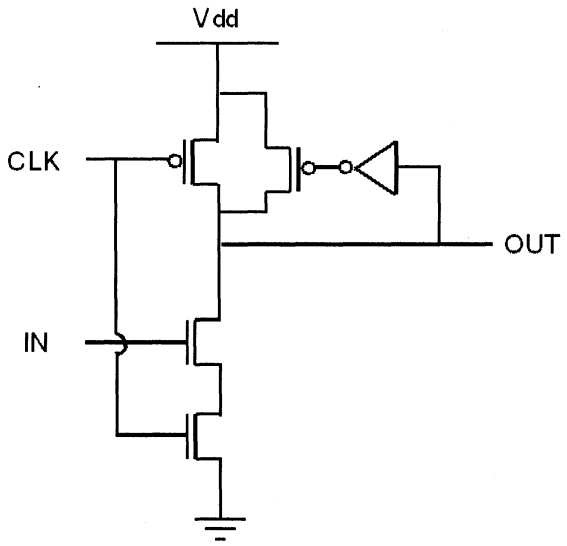


Figure 4-8 Schematic of INV

The second OPT_INV in the critical path provides the inputs whichever polarity that the NOR8 gate requires to generate a product term. Note that the NOR8 and the MUX8/NOR2 share the same circuit diagram. The MUX8/NOR2, however, may have at most two inputs active as restricted by mapping to obtain good speed, whereas there is no such constraint for the number of inputs in NOR8. The speed of the NOR8 will be slower because the high output is dependent upon all inputs falling simultaneously while the MUX8/NOR2 is dependent upon only two inputs falling. By enabling all eight OPT_INVs at the same time, an AND8 can be implemented in the PTG. Including the AND2 generated in the MUX part, up to an AND16 can be realized.

4.2.3.3 Wired NOR

For the two levels of NOR part, the first-level NOR4, shown in Figure 4-9, introduces enables that lie on the critical path. The structure used for the MUXes and PTGs cannot be used here since the PTGs and the NOR4 must be separated by an even number of inversions. Inserting two inverters between the PTG and NOR4 and incorporating the enable on the second inverter is a possible solution; this turned out to be not as fast as simply incorporating them into the NOR4 gate.

The second-level NOR4 (NOR4_DIST) drives the long vertical routing tracks. An odd number of inversions must lie between the two levels of NOR4, so adding INV_EN, which incorporates the enables, is advantageous. The INV_EN, shown in Figure 4-10, works similarly to the OPT_INV, except that it cannot choose the polarity of the input. It is particularly important to use INV_EN here, because the long tracks have large parasitic capacitance as well as resistance. Adding drive capability to a NOR4 containing enables introduces larger devices and adds significant additional internal capacitance and delay. By moving the enables to the

previous stage, device sizes can be reduced significantly and the output tracks can be driven efficiently.

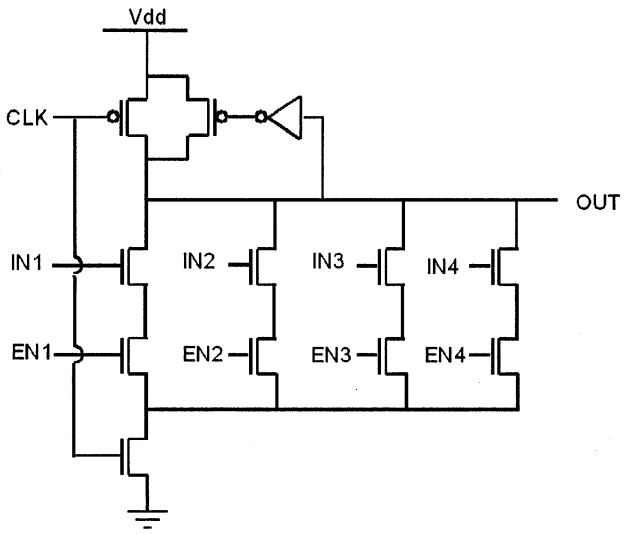


Figure 4-9 Schematic of NOR4_EN

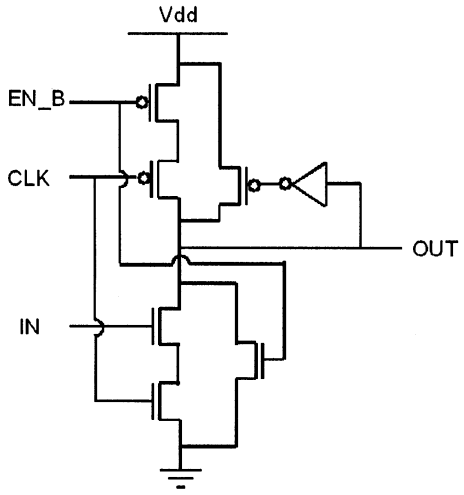


Figure 4-10 Schematic of INV_EN

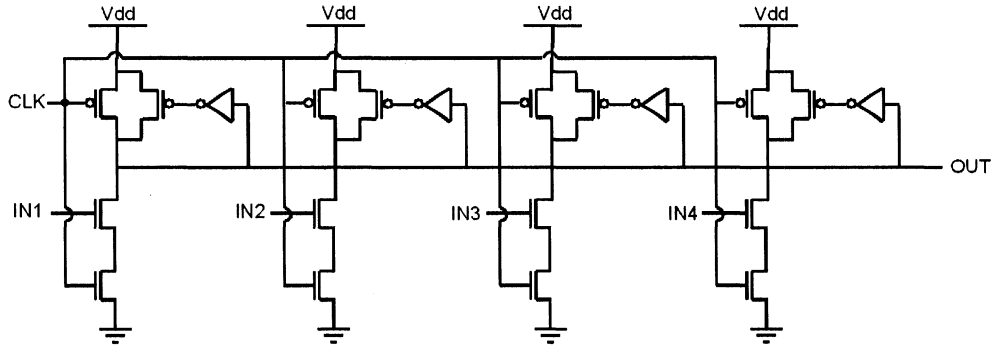


Figure 4-11 Schematic of NOR4_DIST

Figure 4-11 shows the schematic of the second level NOR4 (NOR4_DIST). There are four distributed OPL dynamic inverters, whose outputs are connected by the long tracks. In the following section, it will be shown that this NOR4 is distributed over as many as 4 first-level NOR4s, for a total NOR of up to 16 product terms. Each driver contains its own evaluation device controlled by the clock.

4.2.4 Routing Structure

Just like other programmable logic structures, OPL-PLC has both local and global routing tracks for the sake of programmability. We already described the local tracks in the high-level architecture section. In addition, we proposed a novel global routing structure in the OPL-PLC, in which the output tracks are employed to implement wired-NOR4 in addition to pass signals. Figure 4-12 shows the input and output track distribution. Each TLS has 64 long tracks for inputs and 64 long tracks for outputs, where every global track goes throughout all the 64 rows of logic. Each HLRB can access any of the 64 inputs and select at most 16 of them. Also, each HLRB has the same input distribution. For the output tracks, each HLRB can generate at most 32 outputs, where every four of them are the same.

Because the long tracks are used as routing paths to implement the second-level NOR, each HLRB has its own output distribution.

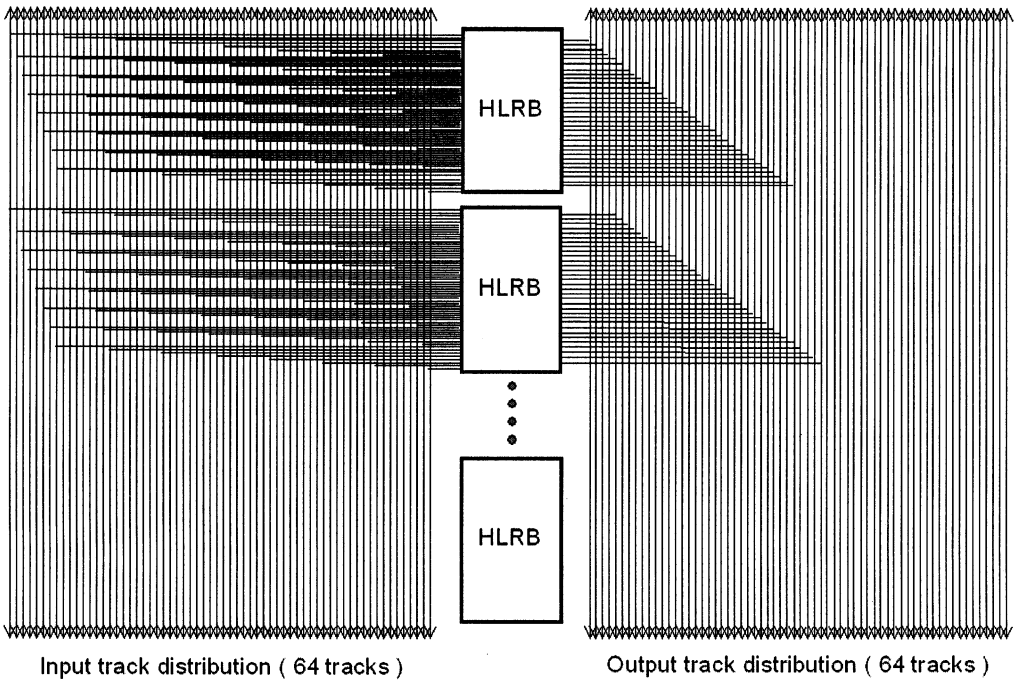


Figure 4-12 Input and output track distribution

Figure 4-13 shows how the output tracks are connected. Every "×" in the diagram represents one output of the HLRB. Since there are thirty-two outputs in one HLRB, there are totally thirty-two "×" in each row, which represents the outputs of one HLRB. There are eight rows in the diagram since eight HLRBs are present in one TLS structure. Every four neighboring outputs with identical logic value are grouped together as illustrated in Figure 4-3. In each column, which is also a global track in TLS, outputs from four different HLRBs are combined together to implement the wired-NOR4. As mentioned previously, this eliminates the need to wire signals to a "localized" logic resource and reduces the wiring length.

The output track distribution shown in Figure 4-13 is not the only way to connect HLRB outputs. Other combinations are possible, as long as the outputs of four outputs from different HLRBs are combined together. It should be noted that for a specific application, different output connections result in different logic efficiency.

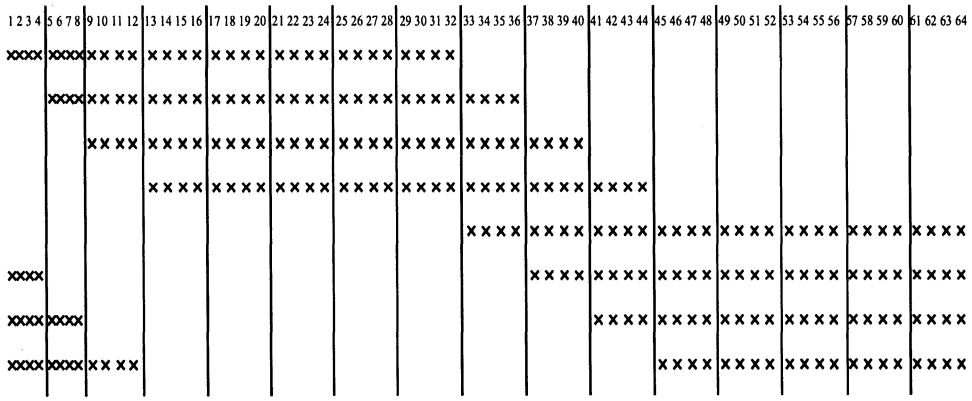


Figure 4-13 Detail of output track distribution

Each output in OPL-PLC can be an AND-NOR function with at most 16×4 = 64 different inputs. Such a wide AND-NOR function decreases the number of logic levels for implemented functions and increases the PLC performance.

4.2.5 Summary of OPL-PLC Features

Table 4-2 lists the features of one TLS. As we mentioned earlier, it has up to 64 different inputs and generates up to 64 different outputs. There are total 2112 gates for logic and routing purposes. 2816 static memory cells are included for logic programming. There are 6 consecutive clock phases to control OPL gates.

Table 4-3 lists the features of the test chip, which has three TLSs in series. The long track before the first TLS is designated for inputs only, which provides 64 inputs.

Each group of long tracks between two TLSs can either pass outputs from the previous TLS or provide primary inputs for the next TLS. Therefore, the test chip can have up to 192 inputs. Only the 64 output tracks of last TLS are designed to pass the final outputs. Since every TLS is identical, the number of gates, SRAM bits and clock phases are just tripled.

Table 4-2 Summary of one TLS features

Feature	Number
Inputs	64
Outputs	64
Logic/routing Gates	2112
SRAM Bits	2816
Clock Phases	6

Table 4-3 Summary of test chip features (3 TLS in series)

Feature	Number
Inputs	64~192
Outputs	64
Logic/routing Gates	6336
SRAM Bits	8512
Clock Phases	18

4.3 Clock Distribution

Clock distribution is also an important issue for OPL circuits. In this section, we will discuss the clocking scheme used in OPL-PLC.

4.3.1 Clock Structure

Since OPL-PLC is implemented with fast OPL gates, the required successive clock phases have short separation times, which cannot be generated by regular buffers. An ultra-high speed clock buffer is necessary to provide the required separation times.

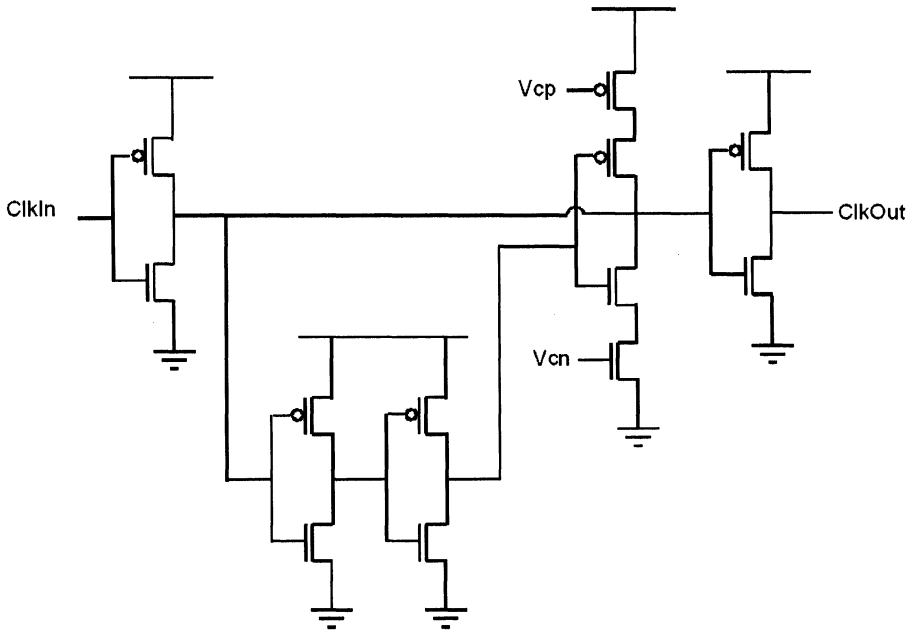


Figure 4-14 Schematic of revised Reduced-Swing Buffer (RSB)

The original reduced-swing buffer (RSB) was described in [48]. The buffer used in OPL-PLC is an improved version (schematic is shown in Figure 4-14) based on the original RSB. The two inverters at the bottom of the figure combined with the one having the control voltages Vcp and Vcn preset the output of the very first inverter of the buffer (with input Clkin) to an intermediate voltage prior to the next clock transition. Since the swing of the intermediate node is reduced, the buffer speed is improved. Compared to the original RSB, by adding a controllable

pMOS in the corresponding position as the controllable nMOS, the revised buffer has a tuning capability for both rising and falling clock edges. Vcn tunes the evaluation separation time and Vcp tunes the precharge separation time. By supplying off-chip voltages to the controllable nMOS and pMOS, the delay of the RSB can be changed post-fabrication. The buffer can provide a minimum delay of 50ps in measurements, whereas the static CMOS fanout-of-four inverter delay is 87ps in the 0.18μm/1.8V TSMC process.

Figure 4-15 shows the clock chain used in the OPL-PLC design. Note that OPL circuits are levelized, in which all gates at a given logic level receive the same clock phase. *Clkin* is the fast clock received from off-chip. The OPL clock phases are then generated on-chip, derived from *Clkin*. The correspondence between clock phases and logic gates on the critical path is shown in Figure 4-5. Every clock phase comprises two parts: the RSB, which is used to provide the separation time; and the buffer tree, which is used to drive the clock load in the logic cells. Each TLS needs 6 clocks and totally 18 clock phases are required for 3 TLSs in series. All the RSBs in the test chip share the same off-chip control voltages, VCN and VCP. The static buffer tree with the maximum load is shown in Figure 4-16.

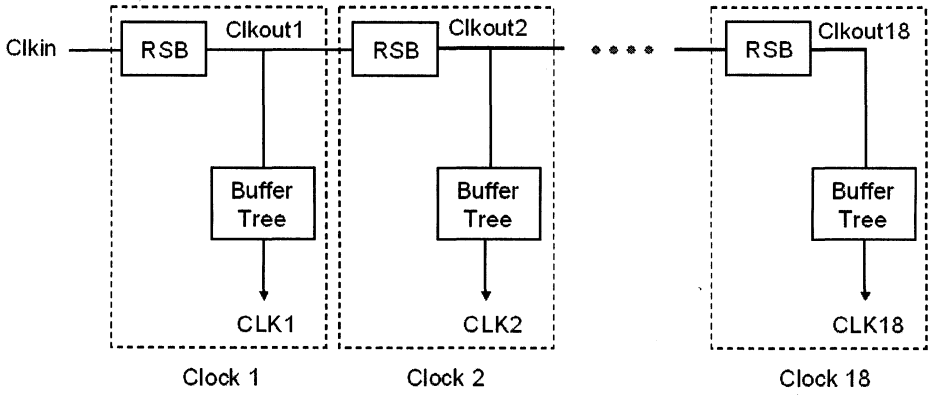


Figure 4-15 Clock Chain used in OPL-PLC

Since different clock phases have different loads, in order to achieve the same slew rate, dummy cells were added into the buffer trees that have loads less than the maximum. A static buffer tree with dummy loads is shown in Figure 4-17. Considering that back-gate (kick-back) effects introduce mismatches in the slew rates of fully loaded and unloaded static buffers, nMOS devices are used to load the dummy cells to avoid this mismatch.

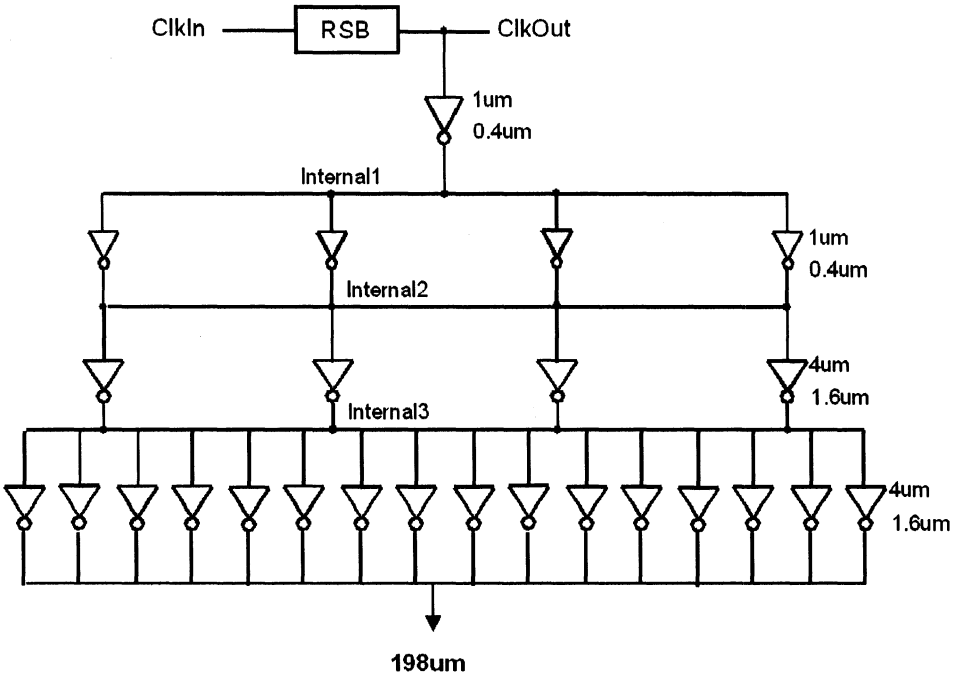


Figure 4-16 Static buffer tree with maximum load

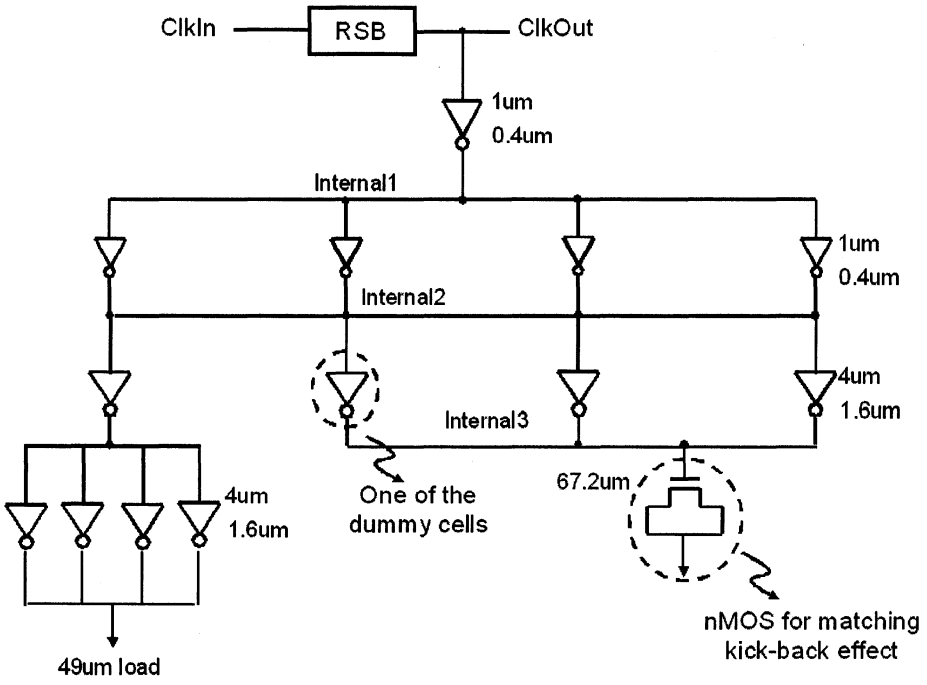


Figure 4-17 Static buffer tree with dummy load

In Figure 4-18, the floor plan of OPL-PLC is illustrated. Because OPL is a dynamic logic technique, levelized placement is used for both the clocks and the logic cells. As we mentioned before, the length of each global track is equal to the height of 64 rows of logic. In order to minimize the length of those global vertical tracks and therefore the wire load for the second-level NOR4s, the clock chain described above was inserted into the rows of logic gates, where the heights of the logic gates and clock cells were minimized.

Clock skew is another crucial concern for dynamic circuit design. To minimize the clock skew, all the same nodes (all the nodes shown in Figure 4-16) in each clock phase were interconnected. For those nodes with longer connection wires, such as Clkout, a clock grid was added. The maximum clock skew among all the 18 clock

phases was designed to be no more than +/- 5ps, which is quite acceptable given the required separation times. The skews were determined by simulation of 3D extracted layout generated by Assura RCX from Cadence.

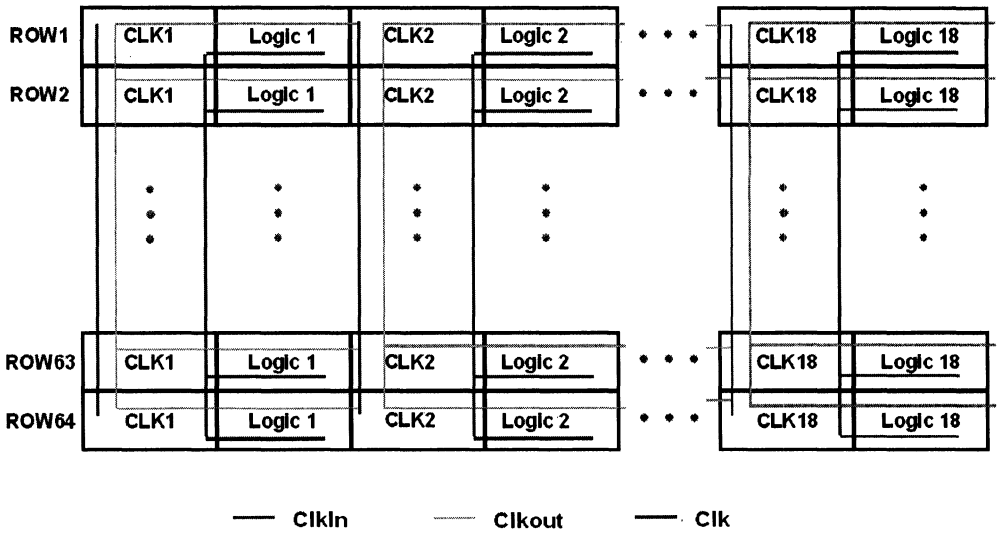


Figure 4-18 Floor plan of the OPL-PLC clock distribution

4.3.2 Circuit Simulation Results Using Ideal Clocks

In order to design the clock chain, we must know what separation times are required for the OPL-PLC. Circuit simulation with ideal clocks was performed first to offer the clock chain design target.

The circuit-level simulations of a chain with 3 TLSs in series were performed by using the TSMC 0.18μm/1.8V process with typical-typical models. MOSIS native design rules were used, stipulating a drawn gate length of 0.18μm. The measured delay of a static inverter with a fan-out of four in this process is 87ps.

In schematic simulation, the length of the global long tracks was estimated to be 672μm (there are 64 rows in the whole layout, and the height of each row is 10.5μm). This is a large load for the second-level NOR4s. In order to get more accurate schematic simulation results, parasitic RC modeling of the long tracks was included in the schematic simulation. For all the other internal nodes, only parasitic capacitances were added.

All gates were configured with worst-case inputs. For example, the worst-case low-output NOR gate is obtained by one input remaining at its precharged-high value and the remainder at Gnd, which is illustrated in Figure 4-19. A worst-case high-output NOR gate is obtained by all inputs falling, which is illustrated in Figure 4-20.

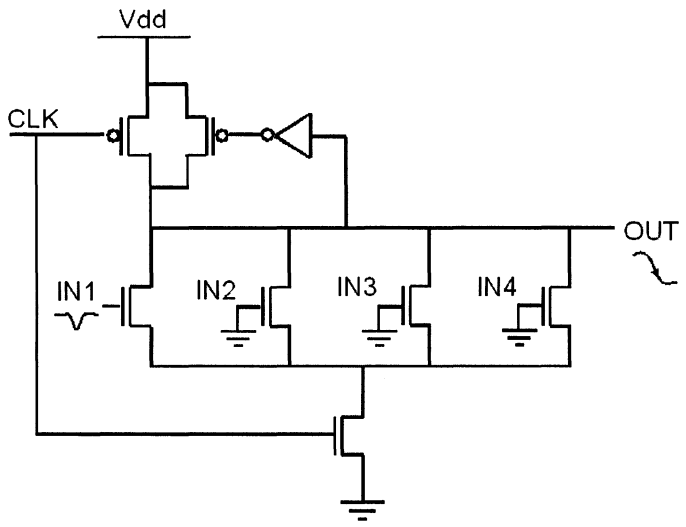


Figure 4-19 Input condition for Worst-case low-output NOR4

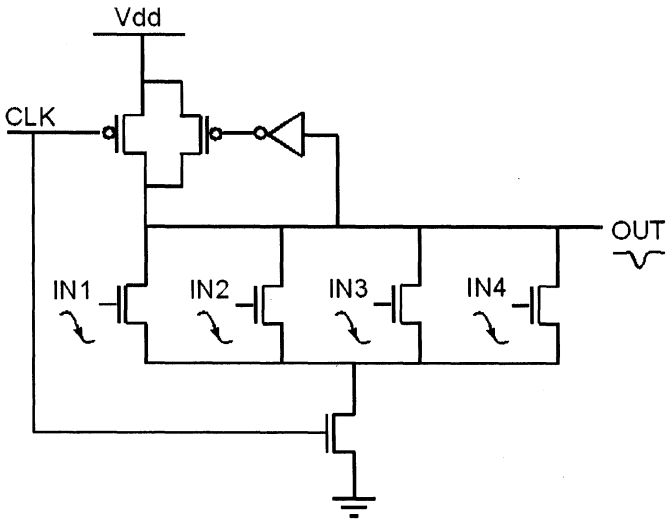


Figure 4-20 Input condition for Worst-case high-output NOR4

In this simulation, ideal clocks were used, having 100ps slew rates. By varying the clock separation times, the set of clock separation times that produces the minimum overall delay was found. Table 4-4 shows the nominal clock separation times for one TLS and Figure 4-21 shows the corresponding clock waveforms.

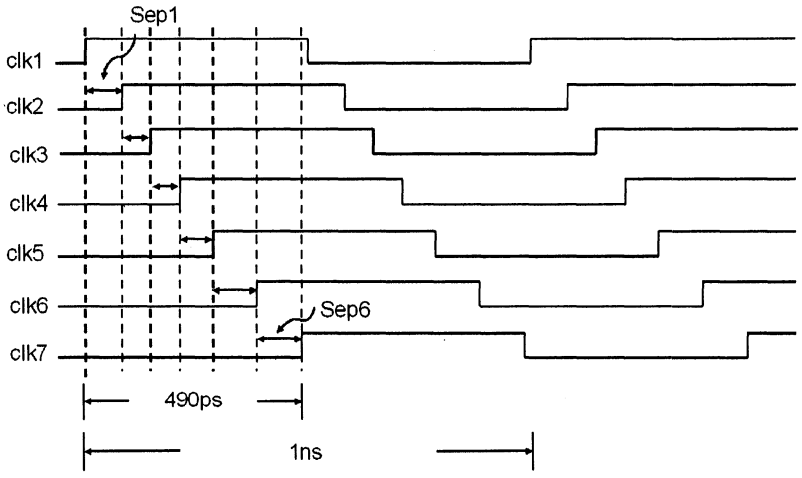


Figure 4-21 Clock waveforms for the nominal clock separations

Table 4-4 Schematic simulation result of clock separation time in one TLS

Nominal Case	Sep1	Sep2	Sep3	Sep4	Sep5	Sep6	Delay
	90ps	60ps	60ps	70ps	100ps	110ps	490ps

Here *Sep1* represents the separation time between *clk1* and *clk2*, *Sep2* represents the separation time between *clk2* and *clk3*, and so on. Please refer to Figure 4-5 for the correspondence between each separation time and the gates on the critical path. The six separation times vary from 60ps to 110ps, and the sum of the six separation times is the total delay for one TLS, which is 490ps. The circuit works correctly for a 1 GHz clock frequency. In other words, the clock period is 1ns, as shown in Figure 4-21.

4.3.3 Clock Separation Time Simulation Results

4.3.3.1 Schematic Simulation Results

After obtaining the nominal separation times using ideal clocks, the clock chain was designed to provide the required separation times.

Initially, schematic simulation of the clock chain was performed to get the initial sizes of the RSBs and clock buffers. Table 4-5 shows the schematic simulation results for the clock chain for the nominal case, including estimated parasitic capacitive wire load information. Here Vcn is set to 1.4V and Vcp is set to 0.4V. The results in Table 4-5 are close to the ideal separation times presented in Table 4-4.

Table 4-5 Schematic simulation results for clock chain of one TLS

	Sep1	Sep2	Sep3	Sep4	Sep5	Sep6	Total Delay
Nominal	90.8ps	57.0ps	63.9ps	71.8ps	98.3ps	113ps	494.8ps
Fastest	75.2ps	43.7ps	42.3ps	54.6ps	90.3ps	96.3ps	402.4ps
Slowest	163ps	139ps	157ps	153ps	167ps	193ps	972ps

In reality, due to the process, voltage and temperature (PVT) variations, the measured separation times won't be exactly the same as the simulated results. Tunable clock separation times are preferred to ensure successful first-try fabrication. In the RSB, this can be realized by varying the voltages on both Vcn and Vcp. These two control signals can be varied from 0V to 1.7V. The simulation results for the fastest case were obtained when Vcn was set to 1.7V and Vcp was set to 0.1V. The simulation results for the slowest case were obtained when Vcn was set to 0V and Vcp was set to 1.8V.

From Table 4-5, we know the shortest delay is 402.4ps and longest delay is 972ps for one TLS. The available tuning range is about 2.4. The nominal delay is in between the shortest and longest delay, which demonstrates that even with PVT variation in the real chip, we still be able to verify the functionality and delay of OPL-PLC on each chip.

4.3.3.2 Post-layout Simulation Results

The post-layout simulation was performed based on the netlist extracted from the final layout of clock chain and logic core with parasitic capacitances and resistances included. The results are listed in Table 4-6. The post-layout simulation result is about 13% slower than the schematic simulation result for the nominal case. This is mainly due to two reasons: the first one is the difference between the estimated

parasitic wire capacitance in the schematic and the more accurate parasitic wire capacitance extracted from layout. Secondly, the parasitic wire resistance was not included in the schematic simulation.

Table 4-6 Post-layout simulation results for clock chain

	Sep1	Sep2	Sep3	Sep4	Sep5	Sep6	Total Delay
Nominal	127ps	89ps	65ps	102ps	137ps	122ps	642ps
Fastest	109ps	73ps	57ps	93ps	124ps	105ps	561ps
Slowest	231ps	188ps	182ps	206ps	242ps	192ps	1231ps

In post-layout simulation, the shortest delay is 561ps and longest delay is 1231ps, which results in the clock tuning range factor of 2.2, a little bit smaller than for the schematic simulation results.

4.4 Data Path Circuits Mapped into OPL-PLC

4.4.1 OPL-PLC Mapping Examples

Before considering the mappings of data path circuits, some simple mapping examples are demonstrated to explain how to map certain functions onto OPL-PLC.

The first example is a detailed mapping of an XOR in one TLS. Figure 4-22 shows the data flow.

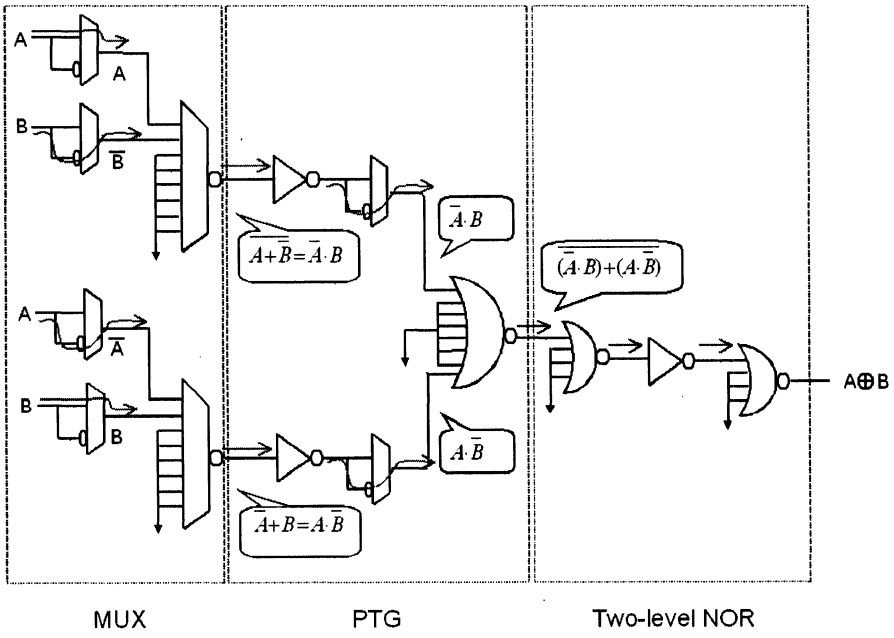


Figure 4-22 Mapping of XOR

The inputs A and B are at the front of the circuit. After choosing the appropriate polarity of the inputs, MUXs implement the two AND2 functions of $\overline{A}\cdot B$ and $A\cdot\overline{B}$. In the PTG part, only two inputs are activated and the other inputs are shorted to GND. The two inputs are combined together to get $\overline{(\overline{A}\cdot B)+(A\cdot\overline{B})}=\overline{A\oplus B}$. The wired-NOR part is only used to pass the result to the output track. Since there is an odd number of inversions in this part, the final output is $A\oplus B$.

In a conventional PLA, the whole PLA, including AND-plane and OR-plane, is necessary to implement the functionality of XOR. However, in OPL-PLC, only the MUX and PTG are required. The two-level NOR is used to buffer the result to an output global track. The rest of three NOR4 gates connected to the same global track are still available to implement more functions. This example demonstrates

that the flexibility provided by OPL-PLC, namely, the AND/NOR-AND-NOR scheme, effectively increases the logic density compared to a conventional PLA. As a result, OPL-PLC is also suitable for implementing small functions.

Now let's look at a random logic mapping example. The relationship between those inputs and outputs is expressed in a truth table style, as shown in Table 4-7. There are 10 inputs (*a, b, c, d, e, f, g, h, i and j*), 9 product terms (*m, n, o, p, q, r, s, t and u*) and 2 final outputs (*x and y*) for this logic. Each row represents a product term and all the "1"s in columns of x and y represent the product terms involved in the corresponding final output. The equations for output *x* and *y* are:

$$x = m + n + o + p + q$$

$$= \bar{a}b\bar{d}\bar{e} + \bar{c}\bar{d}fg + \bar{d}\bar{f}g + a\bar{d}\bar{g} + ac\bar{d}\bar{e}fg \qquad (4\text{-}1)$$

$$y = r + s + t + u$$

$$= \bar{b}cd\bar{e}\bar{f}g + b\bar{c}\bar{d}eg + f\bar{g}\bar{h}i\bar{j} + fghij \qquad (4\text{-}2)$$

It is natural to use a truth table to map the random logic onto a two-level structure. Table 4-7 shows the truth table for the example random logic. In the table, '0' means that the literal is complemented in the corresponding product term, '1' means the literal is uncomplemented in the corresponding product term, and '-' means that the literal is a don't care for that output.

Table 4-7 Truth table of random logic functions

Inputs										Outputs		Product Terms
a	b	c	d	e	f	g	h	i	j	x	y	
0	1	-	0	0	-	-	-	-	-	1	0	m
-	-	0	0	-	1	1	-	-	-	1	0	n
-	-	-	0	-	0	1	-	-	-	1	0	o
1	-	-	0	-	-	0	-	-	-	1	0	p
1	-	1	0	0	1	1	-	-	-	1	0	q
-	0	1	0	0	0	1	-	-	-	0	1	r
-	1	0	0	0	-	1	-	-	-	0	1	s
-	-	-	-	-	1	0	0	1	0	0	1	t
-	-	-	-	-	1	1	1	1	1	0	1	u

Figure 4-23 depicts the mapping structure of the random logic in OPL-PLC. Outputs x and y include only one level of AND-OR, and up to AND6 and OR5 are needed to implement the equations. Therefore one level of TLS is enough to implement both x and y. There are 8 OPL-MUXes in parallel in one HLRB, where each OPL-MUX can select up to 2 inputs and implement them as a NOR2 or AND2. So totally up to 16 inputs can be selected in one HLRB, where every 2 of them are grouped together. The random logic has 10 inputs. However, we cannot group any two inputs in the MUX part and treat them as a unit in the later logic. Although it appears that $\overline{de}$ could be treated as a unit since it is used in the product terms for x and y, we still chose d and e separately. The reason is that d is used separately in some product terms and therefore we have to choose a separated d in the MUX. Choosing d and e separately simplifies the mapping.

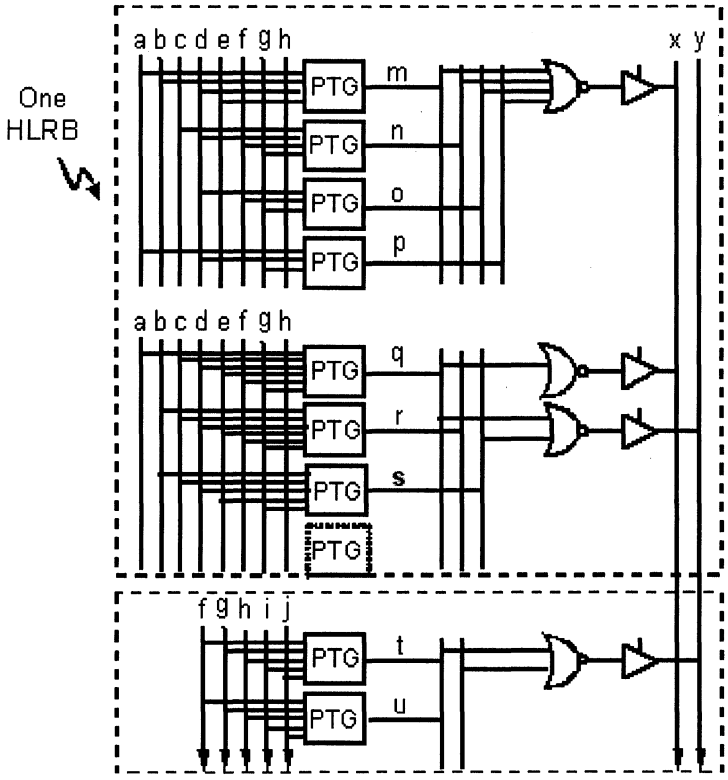


Figure 4-23 Mapping of a random logic function

From the above analysis, more than one HLRB is necessary to implement this function. The first HLRB, represented by the first block of the circuit, selects 8 inputs *a, b, c, d, e, f, g, h.* Seven product terms, *m, n, o, p, q, r, s,* are generated. Two first-level NOR4 gates and one second-level wired-NOR4 gate are utilized for final output *x.* In the second HLRB, only 5 inputs, *f, g, h, i, j,* are selected and 2 product terms *t, u,* are generated. By combining 2 outputs of the first-level NOR4 on the wired-NOR4, output *y* is implemented. The 2 outputs of first-level NOR4 come from 2 different HLRBs and they are combined together on the long track. In this case, the long track acts as both logic and routing resource.

When implementing the output *x*, two outputs from the same HLRBs are connected on the global track, which is not allowed in our current test chip. However, as we mentioned in chapter 4.2.4, other output connections are possible. For this mapping example, the connection shown in Figure 4-23 provides better logic efficiency than the output connection used in current test chip.

There is a dashed PTG block in the first HLRB. This represents the PTG available for other functions. In the second HLRB there are only two PTGs used for output *y*; the rest of the 6 PTGs are still available for other logic. This complicated random logic occupies only a small portion of one TLS. The remaining gates in the TLS can be used by other logic functions. This example demonstrates the high logic density and flexibility of OPL-PLC.

4.4.2 Data Path Logic Mapping

Several representative data path circuits were manually mapped to the OPL-PLC architecture. They are: MUX16-1, 16-bit row decoder, 9-bit parity tree, 16-bit adder and 8-bit pipelined multiplier.

MUX16-1: It chooses one signal out of 16 inputs with 4 different control signals. The Boolean function for the MUX16-1 is:

$$\begin{aligned} F = {} & A_0 \cdot \overline{S_3} \cdot \overline{S_2} \cdot \overline{S_1} \cdot \overline{S_0} + A_1 \cdot \overline{S_3} \cdot \overline{S_2} \cdot \overline{S_1} \cdot S_0 + A_2 \cdot \overline{S_3} \cdot \overline{S_2} \cdot S_1 \cdot \overline{S_0} + A_3 \cdot \overline{S_3} \cdot \overline{S_2} \cdot S_1 \cdot S_0 \\ & + A_4 \cdot \overline{S_3} \cdot S_2 \cdot \overline{S_1} \cdot \overline{S_0} + A_5 \cdot \overline{S_3} \cdot S_2 \cdot \overline{S_1} \cdot S_0 + A_6 \cdot \overline{S_3} \cdot S_2 \cdot S_1 \cdot \overline{S_0} + A_7 \cdot \overline{S_3} \cdot S_2 \cdot S_1 \cdot S_0 \\ & + A_8 \cdot S_3 \cdot \overline{S_2} \cdot \overline{S_1} \cdot \overline{S_0} + A_9 \cdot S_3 \cdot \overline{S_2} \cdot \overline{S_1} \cdot S_0 + A_{10} \cdot S_3 \cdot \overline{S_2} \cdot S_1 \cdot \overline{S_0} + A_{11} \cdot S_3 \cdot \overline{S_2} \cdot S_1 \cdot S_0 \\ & + A_{12} \cdot S_3 \cdot S_2 \cdot \overline{S_1} \cdot \overline{S_0} + A_{13} \cdot S_3 \cdot S_2 \cdot \overline{S_1} \cdot S_0 + A_{14} \cdot S_3 \cdot S_2 \cdot S_1 \cdot \overline{S_0} + A_{15} \cdot S_3 \cdot S_2 \cdot S_1 \cdot S_0 \end{aligned} \quad (4\text{-}3)$$

This equation includes 16 AND5 functions followed by an OR16, which can be easily implemented in one TLS level.

Figure 4-24 illustrates the mapping of the MUX16-1 in OPL-PLC. In the TLS, only four HLRBs are used and the final result is achieved on the wired-NOR4.

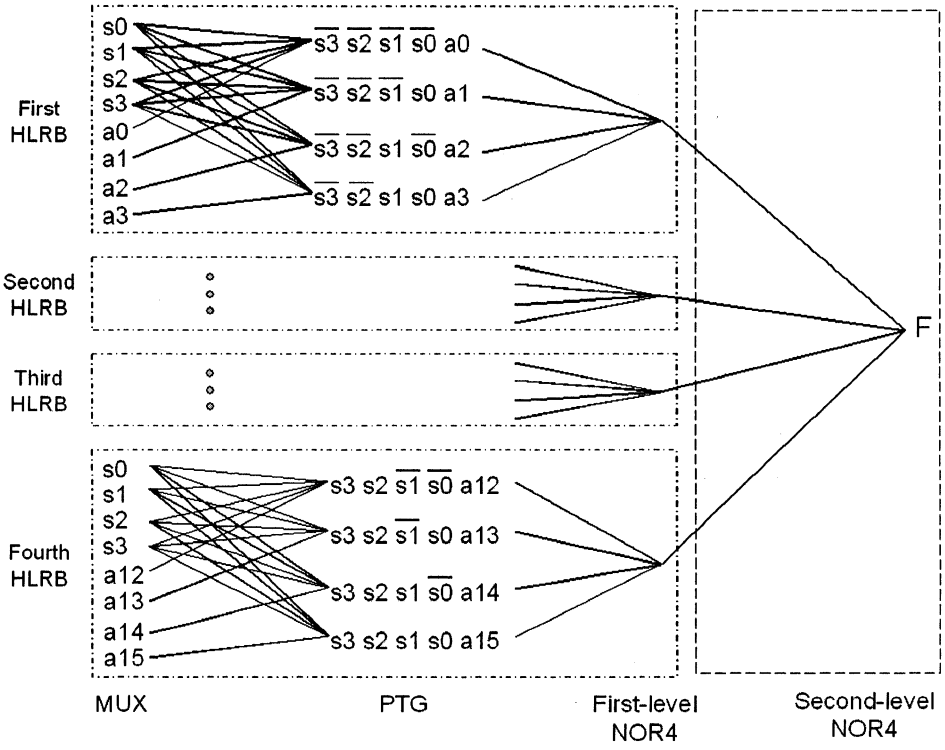


Figure 4-24 Mapping of MUX16-1

16-bit row decoder: A row decoder is required for memories, allowing random address-based access. The 16-bit row decoder has 2^{16} different combinations. Sixteen combinations were mapped into the proposed architecture. One example of a 16-bit row decoder Boolean function is:

$$F = \overline{a_{15}} \cdot \overline{a_{14}} \cdot \overline{a_{13}} \cdot \overline{a_{12}} \cdot \overline{a_{11}} \cdot \overline{a_{10}} \cdot \overline{a_9} \cdot \overline{a_8} \cdot \overline{a_7} \cdot \overline{a_6} \cdot \overline{a_5} \cdot \overline{a_4} \cdot \overline{a_3} \cdot \overline{a_2} \cdot \overline{a_1} \cdot \overline{a_0} \qquad (4\text{-}4)$$

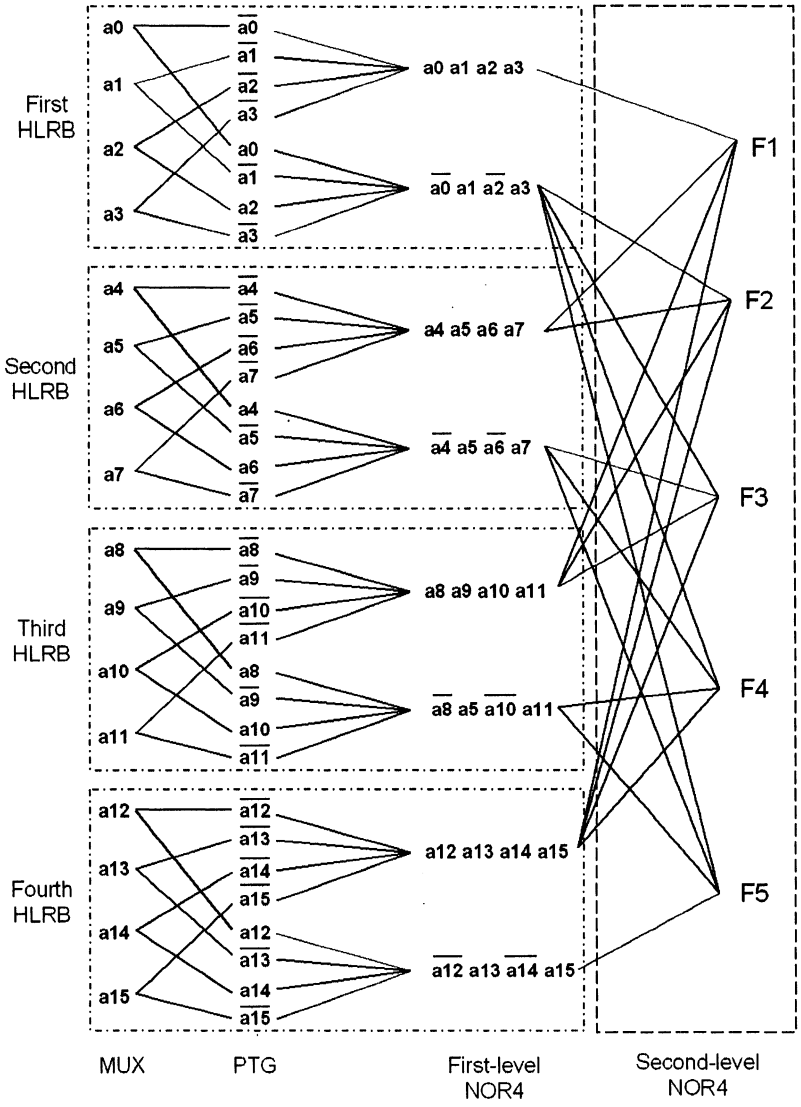


Figure 4-25 Mapping of 16-bit row decoder (several outputs are shown as examples)

$a_0 \sim a_{15}$ are the 16 address bits. The output F is used to control the word line of the memory. This equation consists of only an AND16, which can be handled by one TLS level. Figure 4-25 gives several mapping examples for a 16-bit row decoder. As we described earlier, OPL-PLC can implement an AND16 and a NOR16, so the function of a 16-bit row decoder can be realized either in the MUX and PTG or in the two-level NOR. We chose the two-level NOR because it can implement more combinations in one TLS than using PTGs. If we use the MUX and PTG, every HLRB only implements one equation. Considering the characteristics of a row decoder, many equations share common product terms. By using the two-level NOR, the second level wired-NOR has flexible access to a variety of common product terms, and these common product terms can be shared between different HLRBs. Totally 16 different combinations can be achieved in one TLS, where all eight HLRBs are utilized. The following equation describes the logic conversion from AND to NOR.

$$\begin{aligned} F &= \overline{a_{15} \cdot a_{14} \cdot a_{13} \cdot a_{12} \cdot a_{11} \cdot a_{10} \cdot a_9 \cdot a_8 \cdot a_7 \cdot a_6 \cdot a_5 \cdot a_4 \cdot a_3 \cdot a_2 \cdot a_1 \cdot a_0} \\ &= \overline{(a_{15} \cdot a_{14} \cdot a_{13} \cdot a_{12}) + (a_{11} \cdot a_{10} \cdot a_9 \cdot a_8) + (a_7 \cdot a_6 \cdot a_5 \cdot a_4) + (a_3 \cdot a_2 \cdot a_1 \cdot a_0)} \\ &= (a_{15} + a_{14} + a_{13} + a_{12}) + (a_{11} + a_{10} + a_9 + a_8) + (a_7 + a_6 + a_5 + a_4) + (a_3 + a_2 + a_1 + a_0) \end{aligned} \quad (4\text{-}5)$$

9-bit parity tree: The parity tree is widely used in testing to check for single-bit errors in words (*e.g.*, words in a memory). The Boolean function for a 9-bit parity tree is:

$$\begin{aligned} F &= a_9 \oplus a_8 \oplus a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1 \\ &= (a_9 \oplus a_8 \oplus a_7) \oplus (a_6 \oplus a_5 \oplus a_4) \oplus (a_3 \oplus a_2 \oplus a_1) \end{aligned} \quad (4\text{-}6)$$

F includes nine XORs. As we mentioned before, each TLS can implement at most one and a half XORs, and therefore three levels of TLS are necessary to implement F.

If the XOR is converted into the AND-OR style, it is found that two levels of TLS are enough for the parity tree function, and the total delay is decreased by 2/3.

$$\begin{aligned} F &= (a_9 \oplus a_8 \oplus a_7) \oplus (a_6 \oplus a_5 \oplus a_4) \oplus (a_3 \oplus a_2 \oplus a_1) \\ &= A \oplus B \oplus C \\ &= A \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} \end{aligned} \tag{4-7}$$

$$\begin{aligned} A &= a_9 \oplus a_8 \oplus a_7 \\ &= a_9 \cdot a_8 \cdot a_7 + \overline{a_9} \cdot \overline{a_8} \cdot a_7 + \overline{a_9} \cdot a_8 \cdot \overline{a_7} + a_9 \cdot \overline{a_8} \cdot \overline{a_7} \end{aligned} \tag{4-8}$$

$$\begin{aligned} B &= a_6 \oplus a_5 \oplus a_4 \\ &= a_6 \cdot a_5 \cdot a_4 + \overline{a_6} \cdot \overline{a_5} \cdot a_4 + \overline{a_6} \cdot a_5 \cdot \overline{a_4} + a_6 \cdot \overline{a_5} \cdot \overline{a_4} \end{aligned} \tag{4-9}$$

$$\begin{aligned} C &= a_3 \oplus a_2 \oplus a_1 \\ &= a_3 \cdot a_2 \cdot a_1 + \overline{a_3} \cdot \overline{a_2} \cdot a_1 + \overline{a_3} \cdot a_2 \cdot \overline{a_1} + a_3 \cdot \overline{a_2} \cdot \overline{a_1} \end{aligned} \tag{4-10}$$

A, B and C are implemented in the first-level TLS and the final result F is implemented in the second level TLS. Figure 4-26 describes the mapping of the 9-bit parity tree in two consecutive TLSs based on (4-7), (4-8), (4-9) and (4-10).

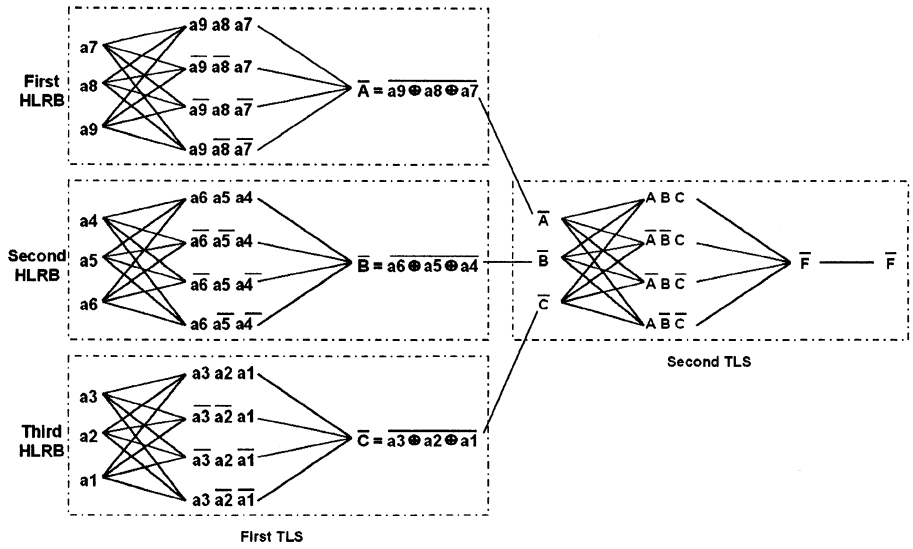


Figure 4-26 Mapping of 9-bit parity tree

Only three HLRBs are occupied in the first TLS and one HLRB is occupied in the second TLS, which is a small portion of the whole OPL-PLC.

16-bit Adder: An adder is a very popular data path circuit. It is much more complicated than the three circuits described above. Figure 4-27 shows the structure of a 16-bit adder mapped into three levels of TLS. A carry-look-ahead (CLA) structure is adopted to implement the adder with fewer levels of TLS.

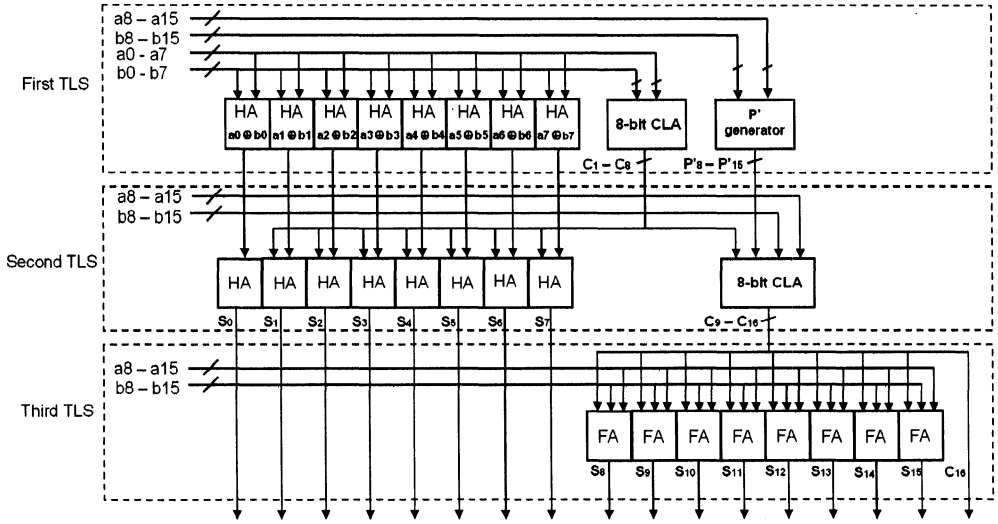


Figure 4-27 Mapping of 16-bit adder

The inputs of the 16-bit adder are $a_0 \sim a_{15}$, $b_0 \sim b_{15}$, and outputs are $S_0 \sim S_{15}$ and C_{16}. The first level TLS is used to compute $C_1 \sim C_8$, the half sums of $S_0 \sim S_7$, and the group propagates $P'_8 \sim P'_{15}$.

$C_1 \sim C_8$ are carry-outs of the first 8 bits. They have similar equations, for example,

$$C_3 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0 \qquad (4\text{-}11)$$

$$\text{where } G_i = a_i \cdot b_i \qquad (4\text{-}12)$$

$$P_i = a_i + b_i \qquad (4\text{-}13)$$

And we assume $C_0 = 0$.

The equations for P_i and G_i have only two primary inputs, and they are used almost everywhere in the CLA adder. The MUX8/NOR2 alone can implement P_i and G_i,

and thus the number of TLS levels is greatly reduced compared to a conventional PLA structure.

The half sums $S_0 \sim S_7$ represent the sum of each corresponding pair of inputs, without adding the carry-in of that bit. Equation (4-14) gives an example of a half sum.

$$S_0 = a_0 \oplus b_0 \tag{4-14}$$

The group propagates $P'_8 \sim P'_{15}$ are used to calculate the carry outs of the next 8 bits ($C_9 \sim C_{16}$). They have similar expressions. Equations (4-15) and (4-16) are two examples of group propagates.

$$P'_{10} = P_{10} \cdot P_9 \cdot P_8 \tag{4-15}$$

$$P'_{11} = P_{11} \cdot P_{10} \cdot P_9 \cdot P_8 \tag{4-16}$$

The second level TLS gets the final sum of the first 8 bits ($S_0 \sim S_7$) and computes the carry out of the next 8 bits ($C_9 \sim C_{16}$). Since the half sums and carries of the first 8 bits are produced by the first TLS, only a half adder is necessary for each bit to get the final sum. For the carry-outs of the next 8 bits, their Boolean function is quite similar to the first 8 bits. For instance:

$$C_{11} = G_{10} + P_{10} \cdot G_9 + P_{10} \cdot P_9 \cdot G_8 + P'_{10} \cdot C_8 \tag{4-17}$$

where P_i and G_i have the same definition as above.

The third level TLS passes the sum of first 8 bits, the final carry-out (C_{16}) and computes the sum of the next 8 bits by using 8 full adders. The total delay of the 16-bit adder is 3 times the delay of one TLS.

8-bit pipelined multiplier: The multiplier is another commonly used data path circuit. Figure 4-28 shows the structure of an 8-bit pipelined multiplier mapped onto the OPL-PLC architecture. The inputs are $x_0 \sim x_7$ and $y_0 \sim y_7$, and the 16 outputs are $z_0 \sim z_{15}$.

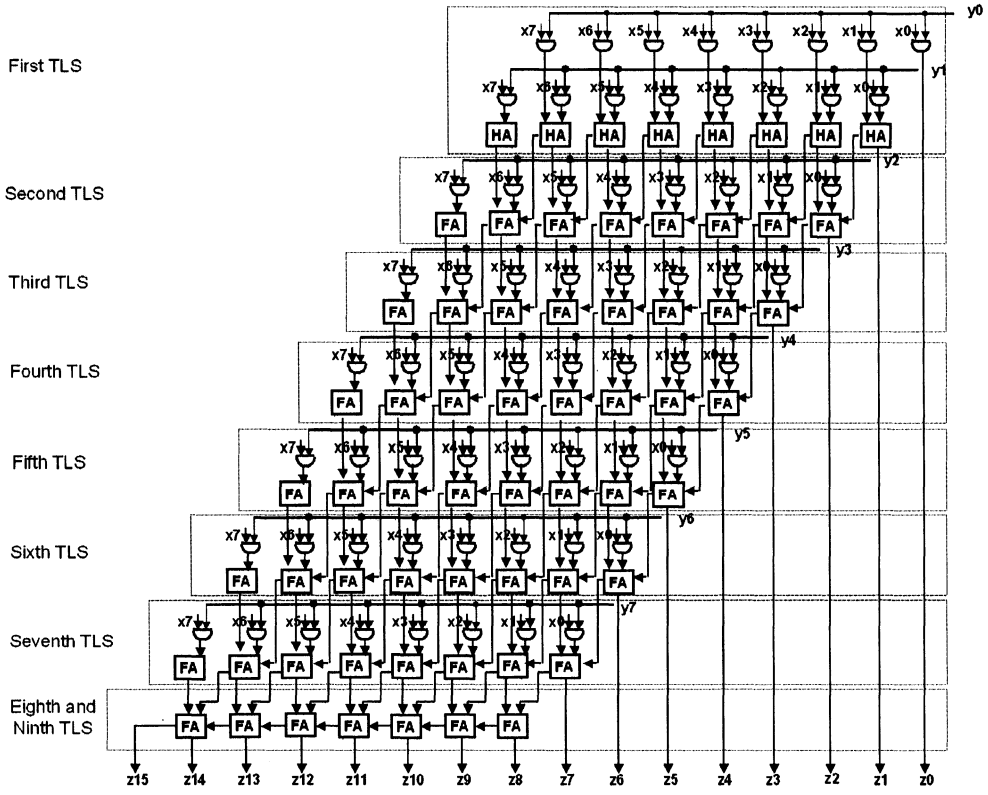


Figure 4-28 Mapping of 8-bit pipelined-multiplier

There are eight stages as shown in the figure, and the AND2 function appears in almost every stage. Due to the presence of the MUX8/NOR2, each stage can be effectively implemented within one TLS, except the last one. The last level is a 7-bit carry-propagate adder, which needs two levels of TLS. The whole multiplier requires 9 levels of TLS. When this structure is used as a pipeline, the clock cycle time is equal to one TLS delay.

4.4.3 Mapping Results for Data Path Circuits

After manually mapping the above five circuits, the mapping was functionality verified by converting them into gate level Verilog code and simulating. After that they were simulated in both schematic and final layout views.

Schematic simulation results: Table 4-8 (column 5) shows the schematic simulation results for the above five circuits mapped onto the OPL-PLC architecture. The results are compared to the Xilinx Virtex-E (Speed Grade -7, [50]) published results. The Xilinx Virtex-E family is fabricated using the UMC 0.18μm/1.8V process. In schematic simulation, only the worst-case scenario was simulated, therefore, the worst-case delay is used to estimate the delay of different circuits. In fact, the performance is design-dependent; many designs can operate faster than the estimated delay. Nevertheless, in Table 4-8, the OPL-PLC achieved at least a factor of 2.9 speed improvement, with significantly greater speedups achieved for several circuits. The average speedup obtained is 5.86. (Geometric mean is used for average speedup.)

Table 4-8 Mapping results for datapath circuits and comparison with Xilinx Virtex-E FPGA

Function	Bits	Xilinx Virtex-E (ns)	OPL-PLC		
			# Levels of TLS	Schematic Delay (ns)	Post-layout Delay (ns)
16:1 Multiplexor	16	4.6 (1.0)	1	0.49 (9.4)	0. 51 (9.0)
Row Decoder	16	3.8 (1.0)	1	0.49 (7.8)	0. 55 (6.9)
Parity Tree	9	3.5 (1.0)	2	0.98 (3.6)	0.99 (3.5)
Adder	16	4.3 (1.0)	3	1.47 (2.9)	1. 84 (2.3)
Pipelined multiplier	8 * 8	4.4 (1.0)	9	0.49 (9.0)	0. 64 (6.9)
Average speedup		(1.0)		(5.86)	(5.1)

Post-layout simulation results: Post-layout extracted simulation results (shown in Table 4-8, column 6) were obtained by mapping all the datapath circuits except the

multiplier into layout, and thus more accurate delays were obtained compared to the schematic results. Both parasitic capacitances and resistances were included in the netlist. Ideal Vdd and Gnd voltages were assumed in the post-layout simulation. This was done to achieve feasible run times for the transistor-level circuit simulation. Therefore, the results reported here are a little optimistic compared to the real delay. The non-ideal Vdd and Gnd introduced about a 7% speed degradation, which will be mentioned in a later chapter.

The delay of the pipelined multiplier is estimated by using post-layout worst-case simulation results. The reason is that only 3 levels of TLS were implemented in the final layout, but the pipelined multiplier needs 9 levels of TLS.

The comparison results for post-layout simulation are a little bit slower than the schematic results, primarily due to the inclusion of parasitic wire resistances.

4.5 Power Consumption

Power consumption is a major concern in programmable logic devices. They consume more power than their ASIC counterparts. For instance, a Xilinx 4000 series device can consume as much as 1000nW/MHz/gate whereas its ASIC counterpart consumes only 20~30nW/MHz/gate [51], which is about 1/40 of the FPGA power.

Power dissipation is proportional to the load capacitance. Therefore, by reducing the overall routing area and wire length, we can directly reduce the parasitic capacitance and lower power dissipation. At the structural level, there are several considerations in OPL-PLC for reducing power. Firstly, logic and routing are combined together in the OPL-MUX to decrease routing area. Secondly, long tracks are used as the wired-NOR4 as well as the global routing, avoiding the need

to route the signals locally. Therefore, the wire length for routing is reduced. At the layout level, a 6-layer-metal process was chosen to enable signal routing above the logic gates as much as possible to decrease routing lengths.

4.5.1 Energy Analysis and Comparison

The energy of OPL-PLC was compared with the Xilinx Virtex-E design. The energies of four datapath circuits mapped in both OPL-PLC and Xilinx Virtex FPGA are listed in Table 4-9.

The energy of circuits mapped in OPL-PLC was obtained from post-layout simulation with 3D extracted layout, which includes both dynamic and quiescent energy. For example, the energy of 16-bit adder was obtained as follows. First, we mapped the worst-case configurations into one TLS. Then we got the total energy of one TLS in the post-layout simulation. Since all the TLSs are identical, we can derive the total energy with three TLSs, which is the number of levels used to map the 16-bit adder. After that we calculated the percentage of logic and routing gates used in 16-bit adder over the total logic and routing gates in three TLSs. The final energy of the 16-bit adder was obtained by multiplying the total energy of three TLSs with the percentage. The energy of clocks in unused logic blocks was excluded to ensure a fair comparison with other designs.

The energy for the same circuits mapped onto the Xilinx Virtex-E FPGA (speed grade-7) was obtained from Xpower [52], a power-analysis and detailed power estimation software. The XCV50E in Virtex-E family was chosen to implement those four circuits. Xpower reports two kinds of power consumption, one is the dynamic power consumption of mapped circuit, and the other is the quiescent power consumption of the XCV50E device. Only a portion of the XCV50E was used to implement those datapath circuits, therefore the quiescent energy of unused

blocks in XCV50E device was subtracted from the total quiescent energy to ensure a fair comparison. The energy values Xpower provides are not as accurate as the OPL-PLC energies obtained from layout simulation at the transistor level. All the values provided in Table 4-9 include the energy for both the logic and the clock network.

Table 4-9 Energy comparison between OPL-PLC and Xilinx Virtex-E FPGA

Functions	OPL-PLC		Xilinx Virtex-E	
	Energy (pJ)	EDP (pJ·ns)	Energy (pJ)	EDP (pJ·ns)
MUX	260 (0.57)	132.6 (0.063)	460 (1.0)	2116 (1.0)
Row Decoder	720 (0.52)	396.0 (0.075)	1398 (1.0)	5312.4 (1.0)
Parity tree	200 (1.19)	198.0 (0.337)	168 (1.0)	588 (1.0)
Adder	1950 (1.83)	3588 (0.783)	1065 (1.0)	4579.5 (1.0)
Overall	0.90	0.19	1.0	1.0

From Table 4-9, we find that OPL-PLC consumes similar energy to that of the Xilinx Virtex-E FPGA on average, where the MUX and row decoder consume only half of the energy that Virtex-E design burns, and the parity tree consumes a little bit higher energy. Only the adder burns 83% more energy than that of Virtex-E. The reason is that a 16-bit ripple-carry adder was mapped in Xilinx Virtex-E FPGA, which is more suitable for LUT-based structure, whereas a CLA structure was adopted in OPL-PLC. A CLA adder requires more hardware compared to the ripple-carry adder due to the difference between the two underlying algorithms. However, OPL-PLC exhibits a 2.3 times speed improvement for the 16-bit adder compared to the Virtex-E FPGA. Thus the energy-delay-product (EDP) for OPL-PLC is better than for the Xilinx Virtex-E FPGA. As shown in Table 4-9, the EDP of the 16-bit adder implemented in OPL-PLC is only 78% of that implemented in Virtex-E. Of course, the other three mapped circuits in OPL-PLC have even better EDPs than the 16-bit adder. The overall EDP of OPL-PLC is only 19% of the EDP

in Virtex-E FPGA, which supports that OPL-PLC achieves much better overall performance than the Xilinx Virtex-E FPGA.

4.5.2 Energy Improvements

There are still some possibilities to improve the power consumption of OPL-PLC. Figure 4-29 illustrates the energy distribution of OPL-PLC. There are three contributing parts: logic and routing, RSBs and static buffer trees. The total energy of one TLS in the worst-case is 1152pJ, where 27.8% is from logic and routing, 17.9% is from RSBs and the rest is from static buffer trees. In OPL-PLC, clock generation and distribution burns the majority of the energy. From Figure 4-29, we find that the static buffer tree consumes half of the energy. This is due to the fact that the current OPL-PLC test chip was designed to optimize speed. Therefore, the clock edge rates were designed to be very sharp, around 100ps, by limiting the electrical effort of the inverters to two.

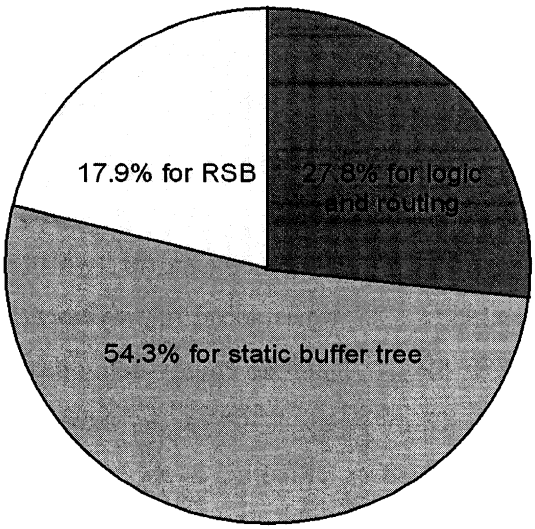


Figure 4-29 Energy distribution in one TLS with FO2 in static buffer

If we increase the electrical effort of the static buffers to four, the clock edge rate degrades to around 145ps. The energy distribution is shown in Figure 4-30. The total energy is 989pJ, only 86% of our current design. It only makes the latency 3% slower than for our current design, which is negligible. It also affects the throughput due to the worse edge rate. The highest throughput decreases from 1GHz to 926MHz. The revised design sacrifices a little bit of performance, but gains a lot in energy consumption.

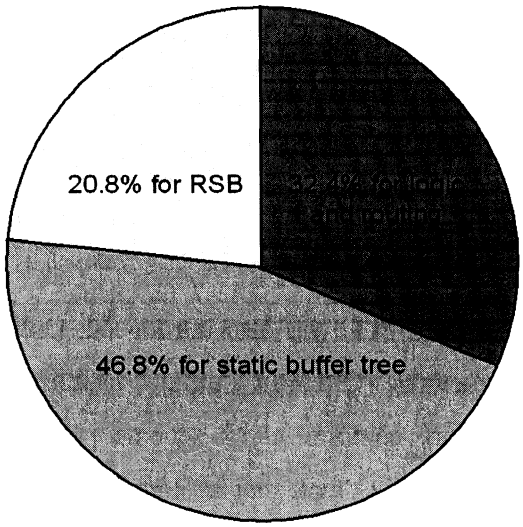


Figure 4-30 Energy distribution in one TLS with FO4 in static buffer

Second, the RSBs we used in the test chip consume considerably higher power (17.9% of the total energy) than what would be necessary, due to short circuit current. By resizing the RSBs, energy can be reduced. Alternative lower power clocking schemes are also possible solutions.

Third, the whole circuit can be sized down to get better energy. When we designed the current version, we optimized for speed. If the logic gates are sized down, the

energy for both logic and routing will be decreased. Meanwhile, the clock load in the logic gates will be reduced and thus clock energy will be reduced.

4.6 Summary

In this chapter, the proposed architecture OPL-PLC is described from the top level to the gate level. OPL-PLC utilizes high-speed, wide OPL NOR gates, a flexible logic block based upon three-level structures, combined logic-routing structures and a unidirectional data flow to achieve high speed and high logic density. In conventional PLAs, they have one level for AND and one level for OR. No matter wide or small functions, they have to occupy the whole two levels. In OPL-PLC, there are two levels in AND and two levels in NOR. When implementing small functions, OPL-PLC can utilize the logic more efficiently than conventional PLAs.

The clock distribution of OPL-PLC was also introduced. Unlike the conventional programmable logic structures, OPL-PLC utilizes dynamic logic family and requires the associated more complex clock scheme. Therefore, clock network consumes more area in OPL-PLC than that in conventional programmable logic structures. Careful optimization of clock network is critical to minimize both silicon area and power dissipation.

The detail explanations of the mappings of five data path circuits demonstrated that OPL-PLC is a very flexible and logic-efficient architecture. Simulation results of mapped data path circuits show that OPL–PLC exhibits much better speed performance than Xilinx Virtex-E FPGA [50]. Energy analysis and comparison with Xilinx Virtex-E also support that OPL-PLC achieves better overall performance.

Chapter 5 : Measurement Results

5.1 Introduction

Testing of high-speed dynamic circuits is another important aspect of this thesis. In this section we will first explain how the test circuits were built for the OPL-PLC chip measurements. After that we will show the measurement results for the test chip and compare its performance with commercial products. Finally the analysis of parasitic effects on OPL-PLC performance will be presented.

5.2 Testing Strategy

The test chip with three TLSs in series was fabricated using the TSMC 0.18μm/1.8V standard CMOS process. In the test chip, besides the proposed architecture, a test structure was included for chip measurement.

Figure 5-1 illustrates the floorplan of the final OPL-PLC layout. The proposed architecture is at the center of the layout, which is controlled by a high frequency clock. SPB represents static programming bit (SRAM cell). During the testing, both the functionality and circuit delay need to be measured. There are 64 to 192 inputs in OPL-PLC test chip, and up to 64 outputs will be generated at the end. It is impossible to access all the input and output pins off-chip directly. In order to reduce the number of required input and output pins, static shift-register chains were adopted. There are four static shift-register chains used as the test circuits to verify the functionality of the architecture, and they are located at the four sides of the proposed architecture. Every static shift-register chain has its own clock, which works at low frequency. There is another column of dynamic flip-flops between the proposed architecture and the shift-register chain on the right side, which is used to

measure the delay of the circuit as well as verify the functionality. It shares the same clock as OPL-PLC's with a certain delay.

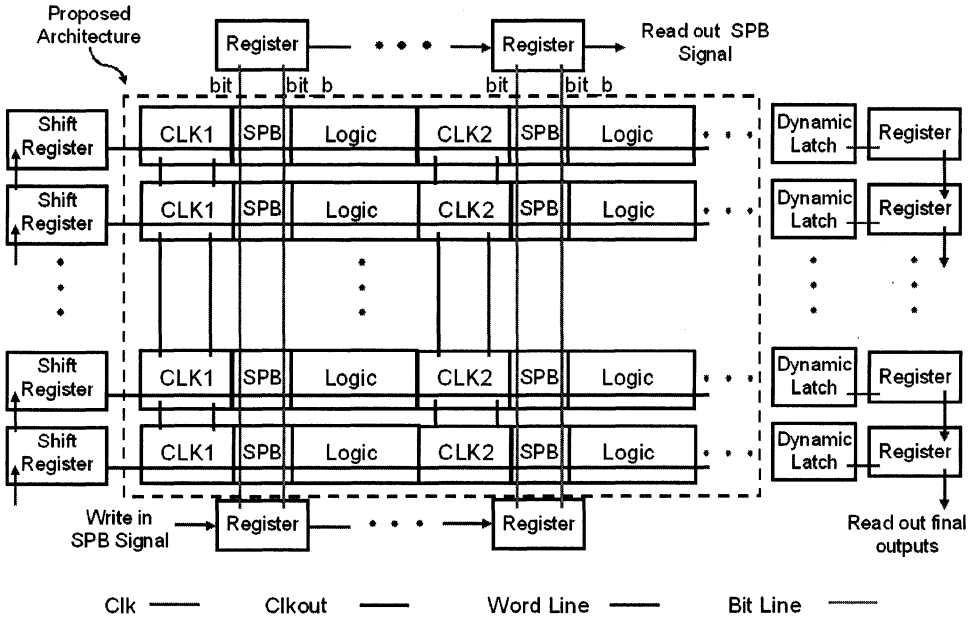


Figure 5-1 Floorplan of the final OPL-PLC layout

The basic cell in each static shift-register chain is a static DFF. Two different versions of the static DFF are used in different register chains. Figure 5-2 shows the first version of the static DFF, a rising-edge-triggered static DFF with asynchronous reset. In normal mode, the reset is set to logic '0' and the correct value of the output is latched at each rising edge of the clock. Under the reset mode, the reset is set to logic '1', which results in output low, no matter what the clock value is at that time.

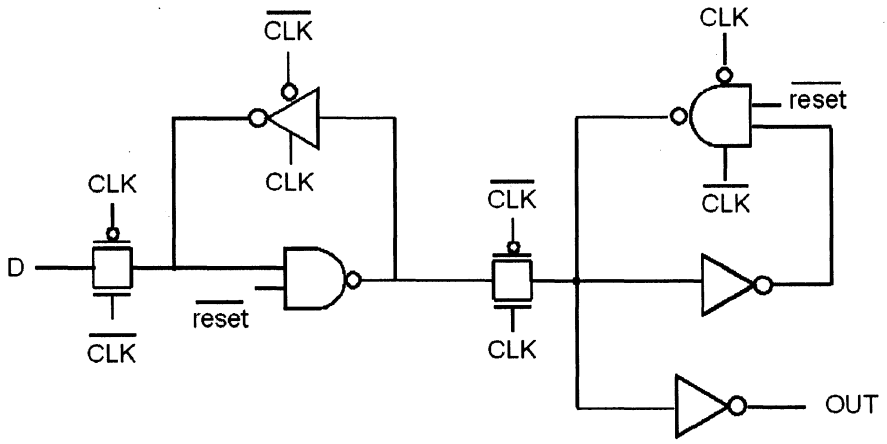


Figure 5-2 Static DFF with asynchronous reset

Figure 5-3 is the second version of the static DFF, a static DFF without reset. It works exactly the same as the previous DFF in normal mode. The initial value of the second version DFF is random due to the lack of a reset mode.

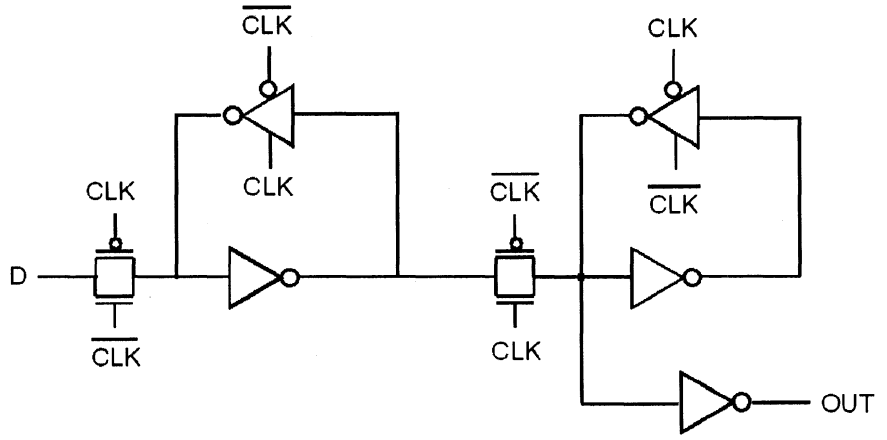


Figure 5-3 Static DFF without asynchronous reset

In order to test the real chip, there are several steps to be considered during the measurement. The first step is how to program the circuit to implement a specific

function. The second step is how to check the functionality of the mapped circuit. The last step is how to measure the circuit delay. The testing method for each step is described as follows.

Programming the real chip. There are more than 8000 memory cells in the test chip to be programmed to configure the OPL-PLC. As we can see from Figure 5-1, they are embedded in the logic gates to decrease the interconnect wire length between memory cells and programming devices. Three shift-register chains are used to write these memory cells; they are on the left side, at the bottom and at the top of the layout of the proposed architecture.

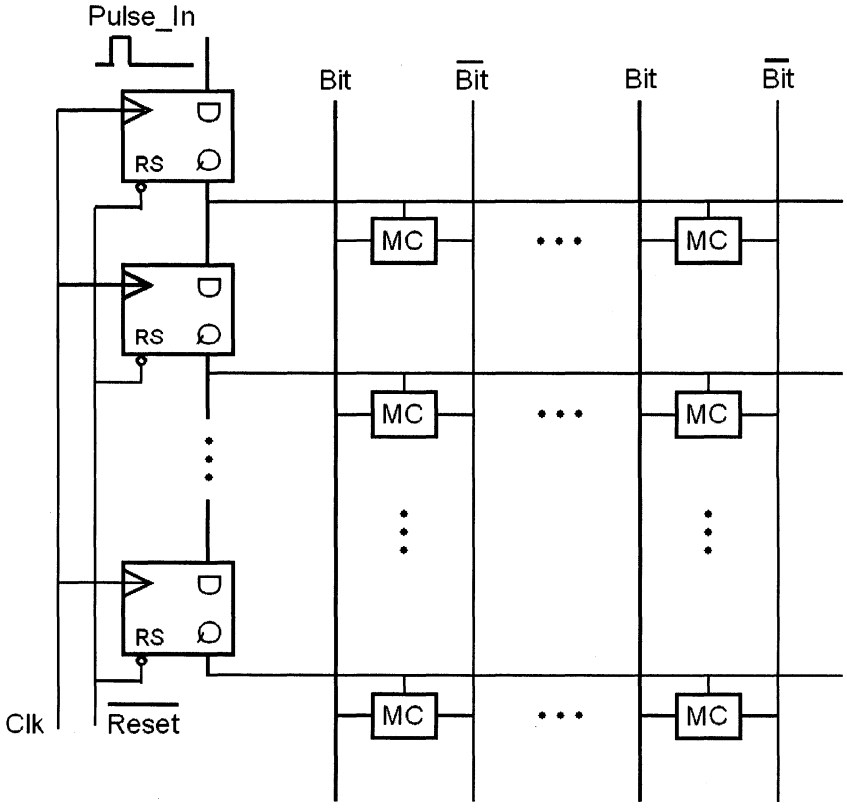


Figure 5-4 Static shift-register chain for controlling the memory word lines.

The shift-register chain on the left side is designed to provide the word line signal for the memory, as shown in Figure 5-4. Unlike typical static memory, no address decoder is used since random access isn't required. Instead, each row is accessed in turn during the programming. By using shift registers and providing a pulse at the input of the register chain, every row of memory cells will be accessed once for each programming. There is a reset function in this register chain to ensure that only one row of memory cells is activated in each clock cycle during programming and no word line is activated when the mapped logic is operating. Therefore, the first version of static DFF is used.

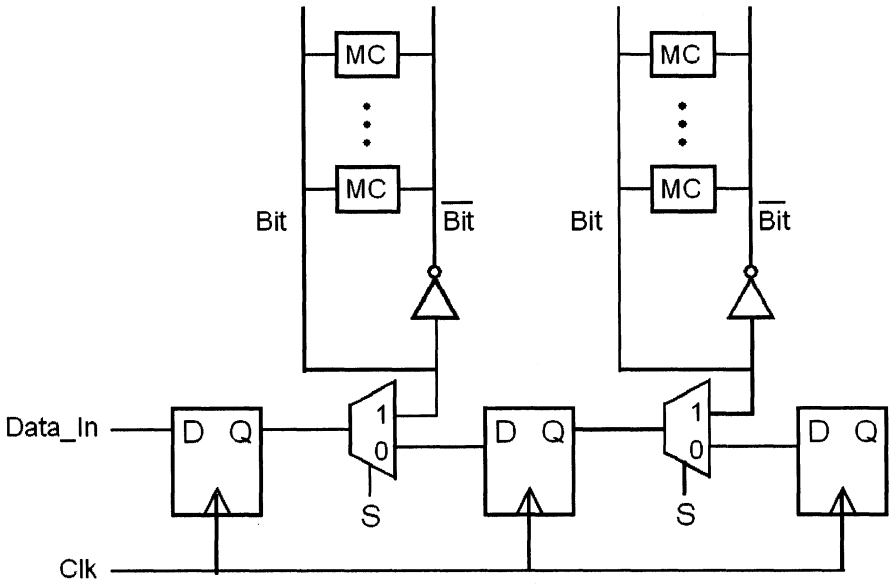


Figure 5-5 Shift-register chain for writing in the static programming bit (SPB) values

The shift-register chain at the bottom is used to shift in the designated value of each memory cell for a specific mapping, as shown in Figure 5-5. At the output of each static DFF, there is a DEMUX, which is used to switch the mode of the shift-

register chain between serial and parallel. When the word line of a certain row is activated, the corresponding programming values for all the memory cells in that row are serially shifted into the bottom register chain. Under this mode, the control signal *S* is set to logic '0'. After all the values are shifted in, the control signal is changed to '1' and the outputs of the bottom register chain become parallel. All the values are written into the memory cells through the bit lines at the same time.

The shift-register chain at the top is used for debugging, in case there is any problem in the memory cells after fabrication. It works similar to the chain at the bottom, except that it reads out the values stored in memory cells.

Checking the functionality of the real chip. For the proposed architecture, after getting the final outputs, we must latch them and shift them out to check the functionality.

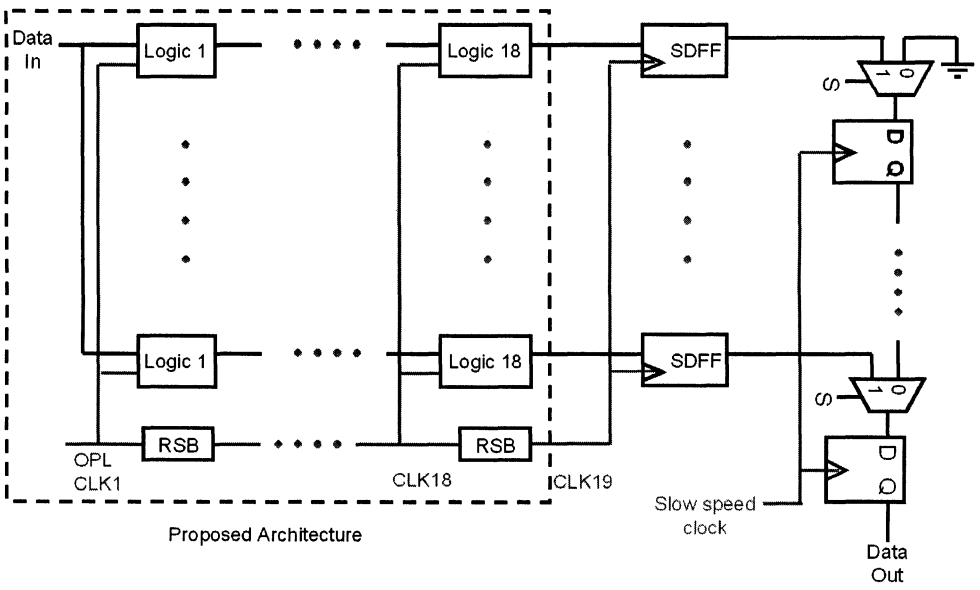


Figure 5-6 Structure of test circuits for reading out the outputs

Figure 5-6 shows the structure of test circuits used to check the functionality. A column of semi-dynamic flip-flops is located at the right side of the proposed architecture. The semi-dynamic flip-flops are designed (sized) to have a nominal setup time of zero. After the outputs are captured by those dynamic flip-flops, they are shifted out to another static shift-register chain, located at the right side of dynamic flip-flops. A 2:1 MUX is used to switch the shift-register chain from input-parallel mode to output-serial mode.

Figure 5-8 shows the schematic of the semi-dynamic flip-flop used in OPL-PLC and Figure 5-7 shows the schematic of original semi-dynamic flip-flop developed by SUN Microsystems [32]. Schematic in Figure 5-8 is a revised version of the schematic in Figure 5-7. There exist two differences. In the revised version, first, the top nMOS transistor of gate 1 is controlled by a signal independent of gate 1's output. Second, the bottom node of gate 6 is connected to the internal node of gate 1, instead of ground to ensure correct functionality. This flip-flop is pulse-triggered and the correct signal is latched at clock falling edge. The output will be held until the next clock rising edge comes.

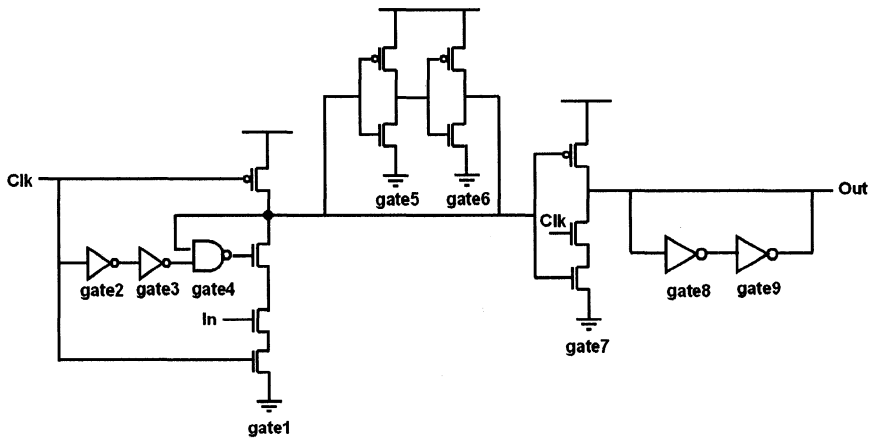


Figure 5-7 Schematic of semi-dynamic flip-flop developed by SUN Microsystems

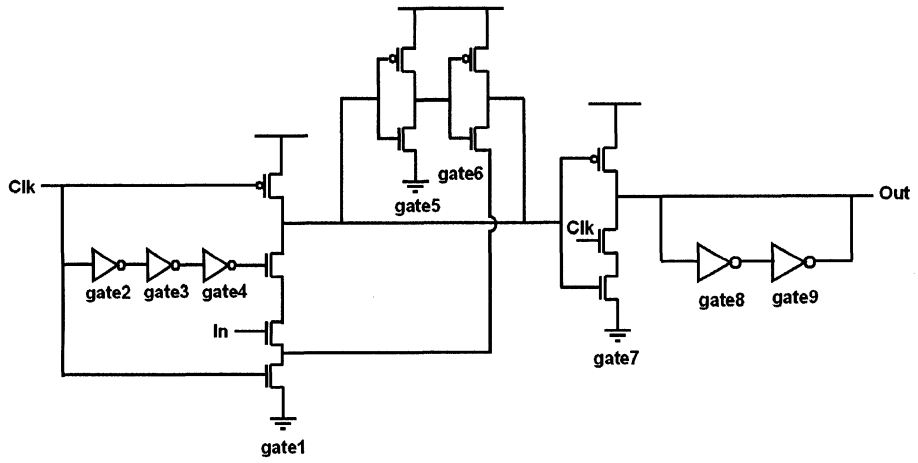


Figure 5-8 Schematic of semi-dynamic flip-flop used in OPL-PLC

Measuring the delay of the real chip. After getting the correct outputs, we need to measure the delay of the circuit. OPL logic is clock-controlled (or clock-blocked) logic, which means the clock separation time represents the delay of the logic. As shown in Figure 5-6, all the dynamic latches share the same clock, the 19th clock phase of the high frequency clock. After the 19th clock rising edge, if every output is latched correctly, by measuring the delay between the first logic clock rising edge and the capturing dynamic flip-flop's clock rising edge (given its zero setup time), the delay can be obtained.

The OPL-PLC delay from post-layout simulation is around 2ns, which is hard to measure accurately after passing through I/O pads. In order to get the accurate delay between the 1st clock and 19th clock from measurements, we have to minimize the mismatch introduced from both on-chip and off-chip. Exceptional care has been paid in layout to maintain the symmetry and minimize the on-chip mismatch. However, the off-chip mismatch might introduce more inaccuracy, which has to be considered. Figure 5-9 shows the scheme to cancel out the mismatch introduced

from output pads and off-chip sources of mismatch, such as probes. When we measure the chip delay on oscilloscope, we always treat *delay_out1* as the first edge to capture and *delay_out2* as the second edge to capture. When the control signal *S* is set to '1', a positive delay between *Delay_out1* (1st clock) and *Delay_out2* (19th clock) is obtained. Then *S* is set to '0' and a negative delay is obtained. By averaging the absolute values of these two delays, an accurate delay for OPL-PLC is achieved. It cancels out all the mismatches introduced after the two MUXes.

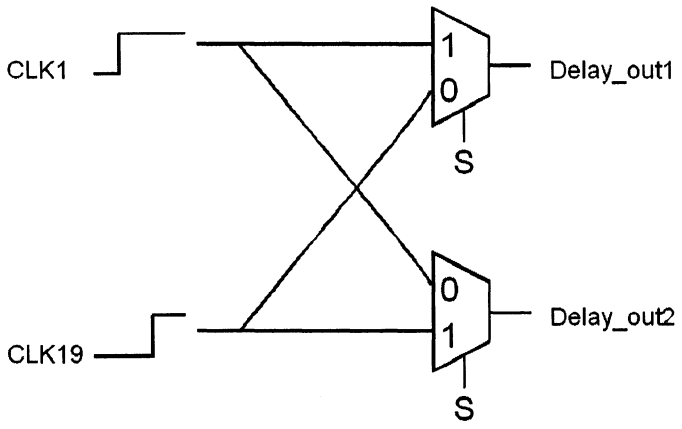


Figure 5-9 Scheme to cancel out mismatch introduced after multiplexers

Advanced measurement equipment was utilized and we found that the pads and off-chip components only introduced about 40ps of mismatch. Compared to the 2ns delay, 40ps is negligible. But this method did provide us with a way to measure the actual mismatch.

5.3 Measurement Results

5.3.1 Features of Test Chip

Figure 5-10 shows the die microphotograph of the final chip. At the center of the figure, we can clearly see the proposed architecture: four groups of long tracks separate the whole layout into 3 identical parts, where each part is a TLS.

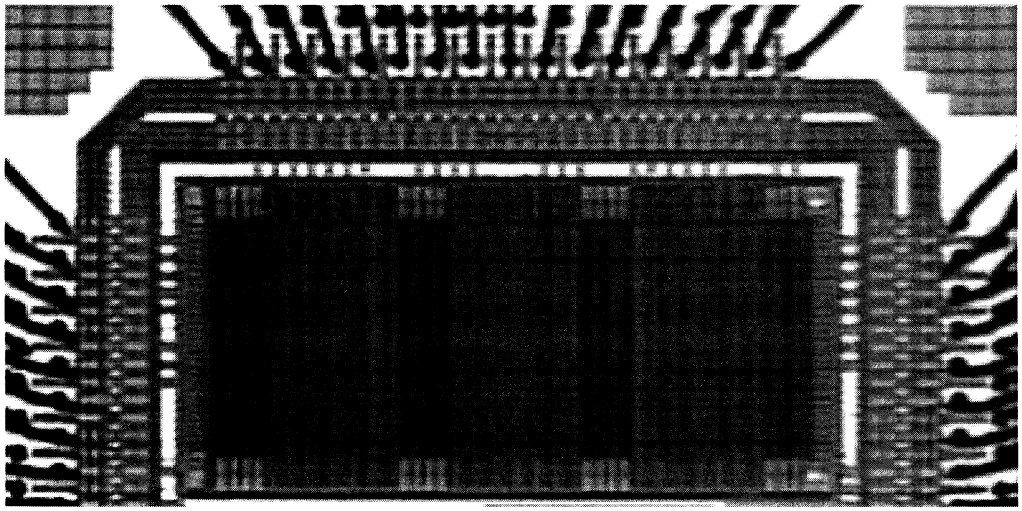


Figure 5-10 Die microphotograph of OPL-PLC

Table 5-1 lists the specifications of the OPL-PLC test chip. The chip size includes the proposed architecture, test circuits and die pads. The array size includes only three levels of TLS. The two input analog pads are VCN and VCP, which are used to tune the separation times of RSB off-chip.

In the proposed architecture, the logic and routing gates occupy 36.4% of the total area, SRAM cells occupy 21.2% of the total area, 33.4% of the area is occupied by clock network and the rest of the area (9.1%) is for decoupling capacitors.

Table 5-1 Specifications of OPL-PLC test chip

Number of TLS	3
Chip Size	2.8mm × 1.4mm
Core Circuit Size	1.66mm × 0.67mm
Process	TSMC 0.18μm/1.8V CMOS
Supply Voltage	1.8V (core), 3.3V (pad)
Input Pads	14 (including 2 analog pads)
Output Pads	4
Power Supply Pads	20 pairs (1.8V), 4 pairs (3.3V)

5.3.2 Testing Setup

First version of test board: In order to verify the functionality and measure the delay of the prototype OPL-PLC, a PGA package was selected and a printed-circuit-board (shown in Figure 5-11) was designed. The dimension of this PCB is 4" by 4". In order to decrease the power supply inductance introduced by the PCB, two layers of the board were chosen to supply Vdd and Gnd. Some surface mount capacitors were soldered on the board between Vdd/Vdd3.3/VCN/VCP and Gnd to filter out the off-chip noise.

The measurement results obtained from this test board were 13% slower than the post-layout simulation results, where the voltage drops on Vdd and Gnd were considered in post-layout simulation. This was due to the large parasitic resistance (R) and inductance (L) introduced by the PGA package and PCB board. In the PGA package, the longest bond wire introduced about 4nH of inductance; the package trace also introduced big parasitic Rs and Ls. The package model is shown in Figure 5-12. Some sample values for the parasitic Rs, Ls, Cs are also listed. A ZIF socket was located at the center of the board. By using it, we can use one board to test many chips, but the socket adds more parasitic components to the chip. The long routing traces on the board and thick board layers introduced more parasitics

as well. All of the above caused our measured results to fall short of what we expected based on simulations.

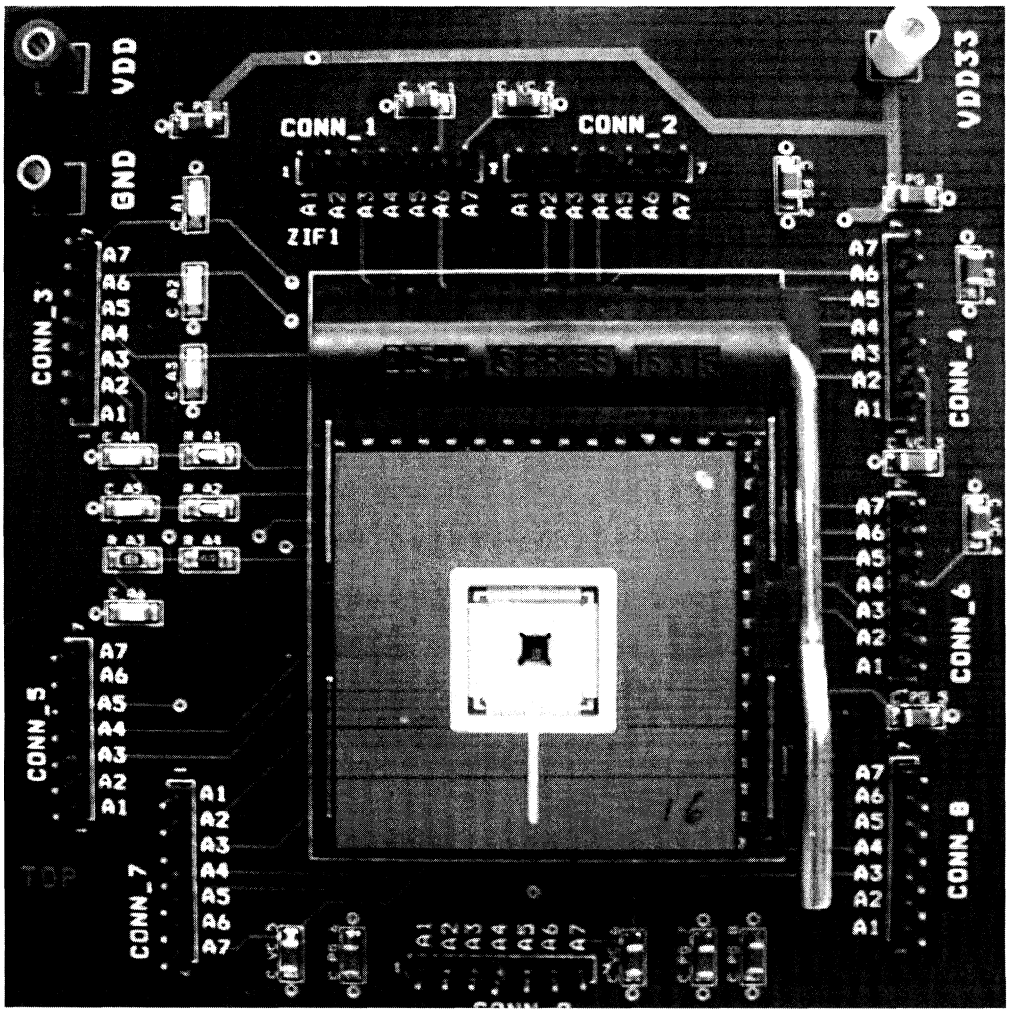


Figure 5-11 First version of printed-circuit board

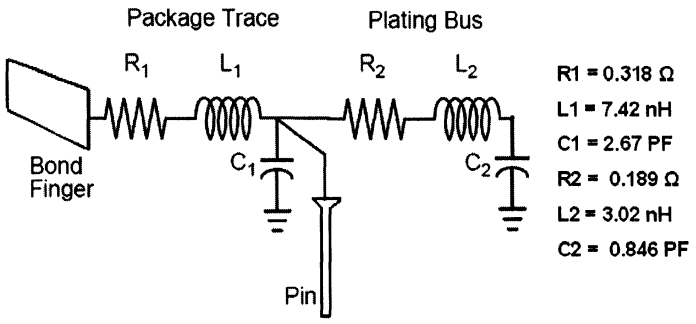


Figure 5-12 PGA package model and some sample values

In order to measure the inherent delay of OPL-PLC without being affected by off-chip effects, a much better package was designed and fabricated.

Chip-on-board package: An advanced chip-on-board package was designed and fabricated, as shown in Figure 5-13. The size of this board is 2.15"× 2.19", which is only ¼ of the previous board. There are several considerations in this board design. Firstly, we avoided the parasitic effects of packages by attaching the die onto the board directly, without using the PGA package. The die and all the bond wires are just inside the red box of Figure 5-13, and the detail is shown in Figure 5-14, where all red color traces and vias are for Vdd; all black colors are for Gnd, all blue colors are for Vdd3.3 and all green color pads are for input and output signals. Secondly, in order to make the on-chip power supply voltages more stable, the off-chip inductances of Vdd and Gnd pads were minimized in several ways: 1) Using three layers of PCB to provide Vdd, Gnd and Vdd3.3, respectively. 2) Make Vdd, Gnd, Vdd3.3 traces and input, output pads close to the die located at center of the board as much as possible to decrease the bond wire lengths. 3) Add as many vias as possible for Vdd, Gnd and Vdd3.3 on the board to decrease the inductance from surface of the board to their corresponding layers. 4) Make Vdd and Gnd traces alternate at each side of the die to balance the parasitics introduced from board. 5)

Make the material thickness much thinner than the previous PCB. The effective bond wire inductance for each power supply pad is around 1nH, which is several times smaller than previous version. The bond wire inductances for input and output pads are around 2nH. Finally, there are more than 10 off-chip decoupling capacitances between Vdd/Vdd3.3 and Gnd to filter out the off-chip noise. There are off-chip decoupling capacitors between the two control signals VCN/VCP and Gnd, too.

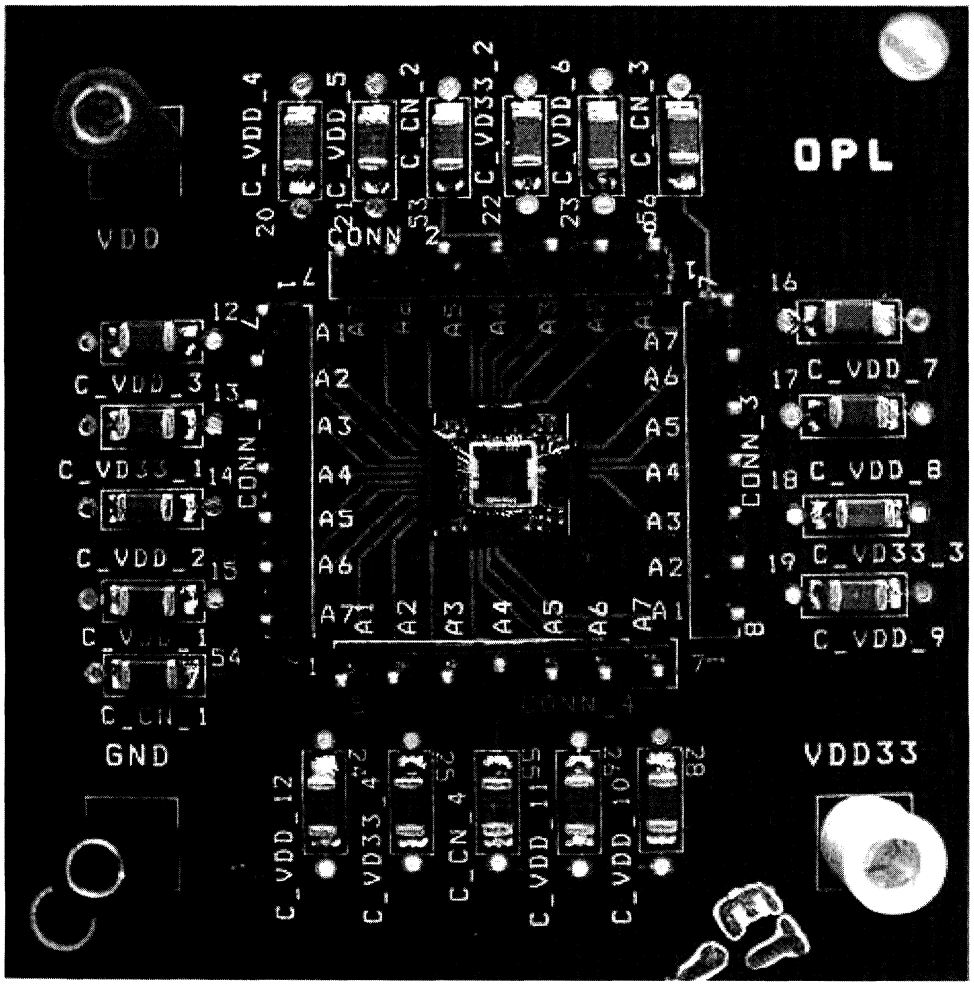


Figure 5-13 Chip-on-board design

As we will show later in this chapter, the measured results from this chip-on-board package match the post-layout simulated results very well. This demonstrates that by carefully choosing and designing the package, the inherent circuit delay can be obtained.

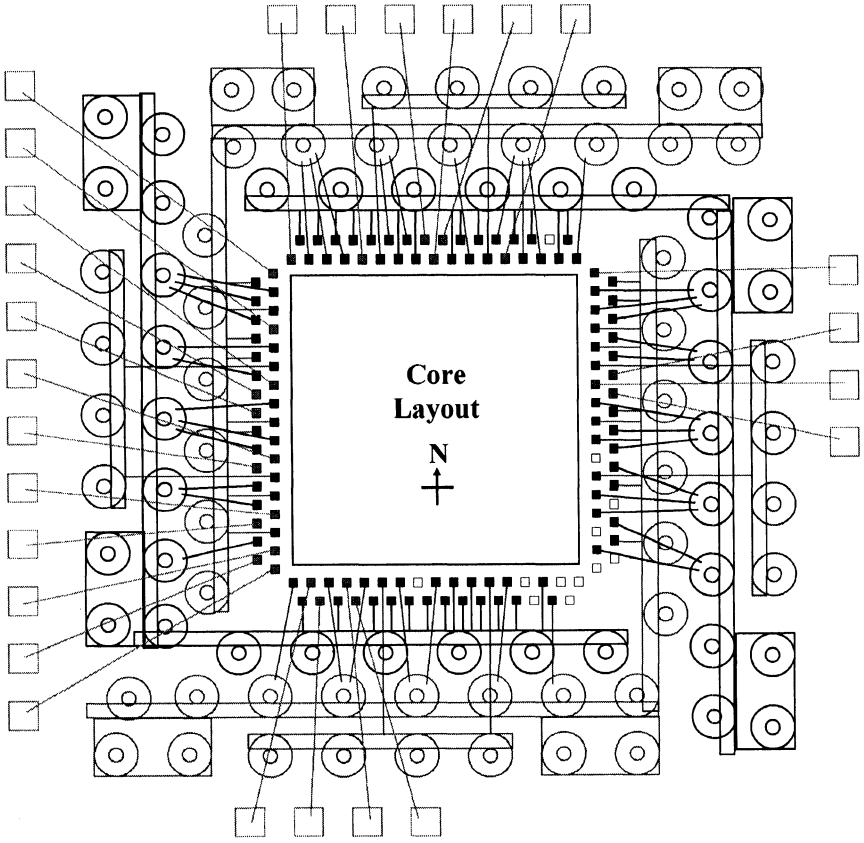


Figure 5-14 Detail of connection inside red box

The circuit was functionally tested using the Tektronix TLA 704 Logic Analyzer, as illustrated in Figure 5-15. All the test vectors were imported into the logic analyzer. After passing through the Tektronix P6470 TTL/CMOS probe, all the inputs for OPL-PLC were generated. A DC power supply provided VDD, VDD3.3,

GND and control signals VCN, VCP. After getting the final outputs, they were serially shifted back to the logic analyzer. The final outputs were exported and checked with the expected values.

The delay measurement was achieved by using an oscilloscope. The set-up is illustrated in Figure 5-16 and the real set-up is shown in Figure 5-17.

During the testing, the PCB board is necessary to support the measurement, but it introduces some off-chip parasitic resistance (R), inductance (L), capacitance (C), and impairs input, output and power supply signals. Other testing equipment will also introduce some mismatch and degradation to the signals. Even though the delay of the OPL-PLC is in the range of nano-seconds, this is hard to measure accurately in the presence of off-chip effects.

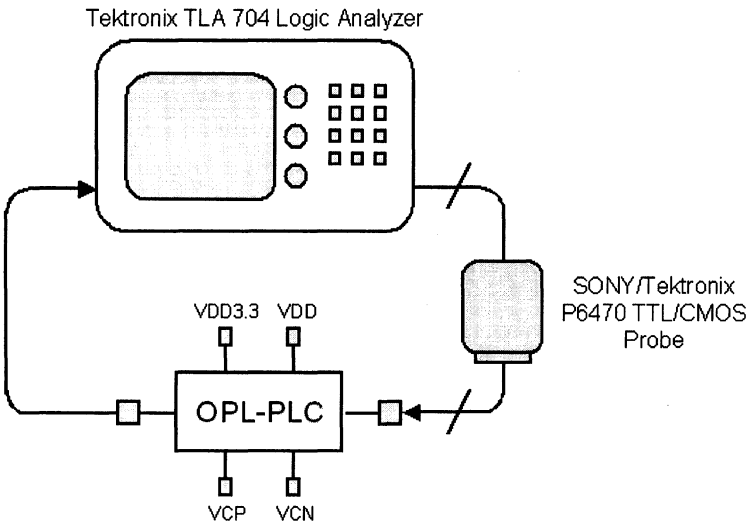


Figure 5-15 Illustration of testing setup for functionality verification.

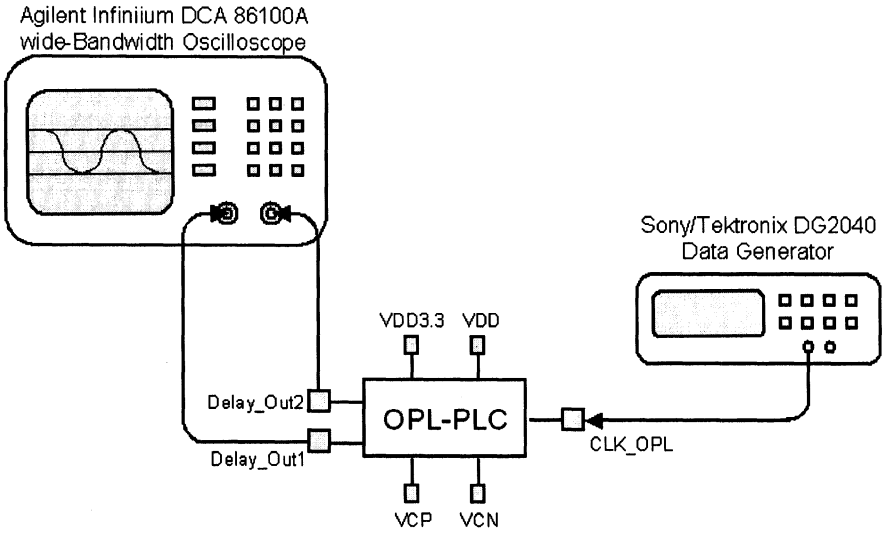


Figure 5-16 Illustration of testing setup for delay measurement

Figure 5-17 Measurement setup

Off-chip parasitic effects don't affect inputs so much because each input signal passes through several static CMOS buffers on the chip before it enters the logic

gates. Due to the characteristic of restoration in static inverters, the curve will be shaped well enough for use. However, there is problem for output signals. The off-chip parasitics will deteriorate the shape of output signals and make the measured delay inaccurate. In order to get accurate delay measurement result, output pads *Delay-Out1* and *Delay_Out2* were on-chip probed. Figure 5-18 show the on-wafer probing of two output pads. The bond wires of these two pads were cut and no external load were included during measurement.

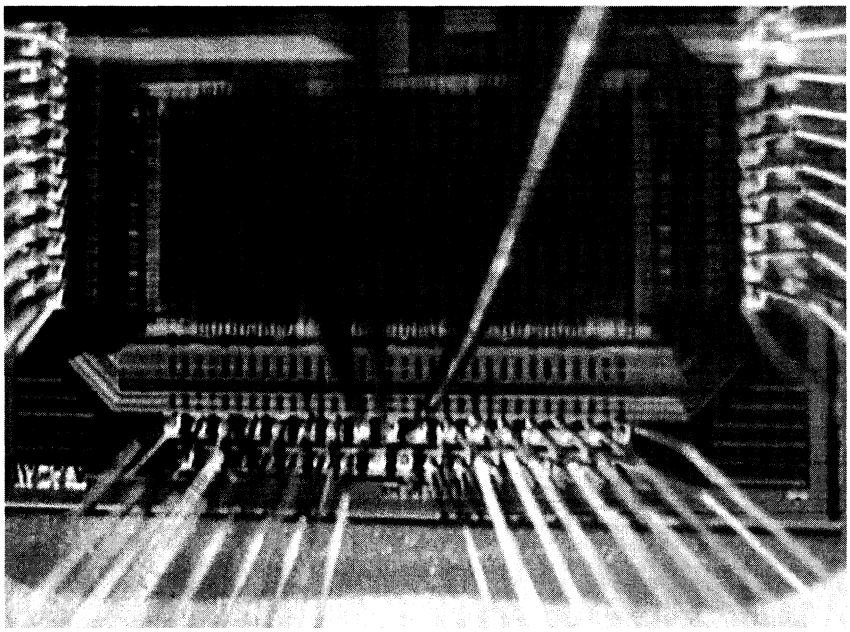


Figure 5-18 On-wafer probing two output pads

5.3.3 Measurement Results

Throughput of OPL-PLC: The throughput of OPL-PLC was simulated with 3D extracted layout generated by Assura RCX from Cadence. The netlist included both parasitic resistances and capacitances. The highest throughput for OPL-PLC and the comparison with commercial products are listed in Table 5-2. In this table,

OPL-PLC is compared with published results for the Xilinx CoolRunner-II CPLD family [49] and the Xilinx Virtex-E FPGA family (Speed Grade -7) [50]. Both the Xilinx CoolRunner-II CPLD and the Xilinx Virtex-E FPGA are fabricated in a 0.18μm/1.8V process. Each family has several different modules and the highest frequency among all the modules was selected for the comparison.

In the following table, it shows that the highest throughput for OPL-PLC is 1GHz, which is 2.4 times that of the Xilinx CoolRunner-II CPLD [49] and 3.2 times that of the Xilinx Virtex-E FPGA (Speed Grade -7) [50]. These comparisons demonstrate that OPL-PLC is suitable for throughput-intensive applications.

Table 5-2. OPL-PLC throughput and comparison with commercial products

	OPL-PLC	CoolRunner-II	Virtex-E
Frequency (MHz)	1000	417	311

Latency of OPL-PLC: The measured delays of those data path circuits we mentioned in chapter 4 were obtained on the test chip. In Table 5-3, OPL-PLC's measurement results are compared with the Xilinx Virtex-E FPGA (Speed Grade -7) published results [50].

In Table 5-3, the delays of the circuits mapped onto Virtex-E FPGA are treated as the references. Both the fastest delay and typical delay of OPL-PLC are reported. The typical delay of OPL-PLC was obtained by averaging the delays of 4 different chips, where their delays with three level TLSs are 2.02ns, 2.06ns, 2.10ns and 2.13ns, respectively. The fastest delays OPL-PLC achieves are faster than Xilinx Virtex-E FPGA by at least a factor of 2.3, with significantly greater speedups achieved for several circuits. The average speedup provided by the fastest delays is 4.1X. Even the typical delays for OPL-PLC provide a 3.7 times speed improvement over Xilinx Virtex-E FPGA (Speed Grade -7) published results [50].

Table 5-3. Performance of OPL-PLC and comparison with Virtex-E FPGA

Function	Bits	OPL-PLC			Virtex-E
		# levels of TLS	Fastest Delay (ns)	Typical Delay (ns)	Delay (ns)
16:1 MUX	16	1	0.63 (7.3)	0.69 (6.7)	4.6 (1.0)
Row Decoder	16	1	0.63 (6.0)	0.69 (5.5)	3.8 (1.0)
Parity Tree	9	2	1.26 (2.8)	1.39 (2.5)	3.5 (1.0)
Adder	16	3	1.88 (2.3)	2.06 (2.1)	4.3 (1.0)
Average Speedup			(4.1)	(3.7)	(1.0)

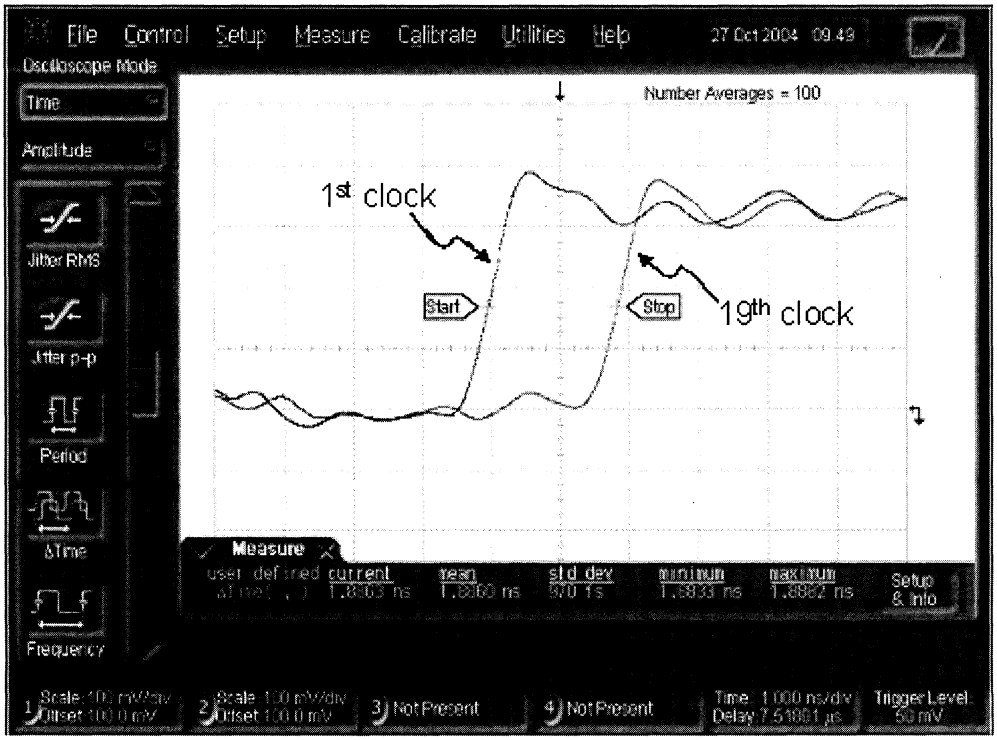


Figure 5-19 Measured delay waveforms of OPL-PLC

Figure 5-19 gives the waveforms for the fastest measured delay in OPL-PLC for its three levels of TLSs in series. The first and last clock rising edges were captured and the time difference between them was measured, which is 1.886ns. Since every TLS is identical, the delay of one TLS will be 1/3 of the total delay. The delays of

small mapped circuits, such as MUX, row decoder, parity tree, were derived from the total measured delay. The output pad works under a 3.3V power supply voltage. In these waveforms, the signals were attenuated to 1/10 of the original voltage by the probes, and therefore their swings are 0.33V in this figure.

The above measured delays are average values taken over a hundred cycles, to cancel random error. The standard deviation in this figure is very small, only 970fs, and therefore the averaged delay accurately reflects OPL-PLC's delay.

From the above results and comparisons, it shows that OPL-PLC is suitable for low latency as well as high-throughput applications.

Compared to post-layout simulations, the measured typical result is about 7.3% slower. In post-layout simulations, ideal Vdd and Gnd were assumed without considering IR drop. As we will discuss in the next section, IR drop on supply voltages made OPL-PLC 7.2% slower under the worst-case scenario. Including this effect, the measurement results agree very well with simulated results.

5.4 Parasitic Effects

The impact of interconnection parasitics on integrated circuits keeps growing along with the scaling down of process dimensions. Interconnections introduce three types of parasitic effects: capacitive, resistive, and inductive, all of which influence signal integrity and degrade the performance of the circuit [14].

5.4.1 Parasitic Capacitance

As shown in Figure 5-20, there are two neighbouring signals, *Aggressor* and *Victim_in*, with long wire length. There exist two kinds of parasitic wire

capacitances: Cg is from signal to Vdd or Gnd, and Cc is from signal to signal. Cg results in more wire capacitive load on both *Aggressor* and *Victim_in* for the driving gates. Cc introduces not only extra capacitive load but also unwanted coupling from signal *Aggressor* to signal *Victim_in*, which is generally called *cross talk*. Cross talk can degrade circuit performance, and can even cause functional failure.

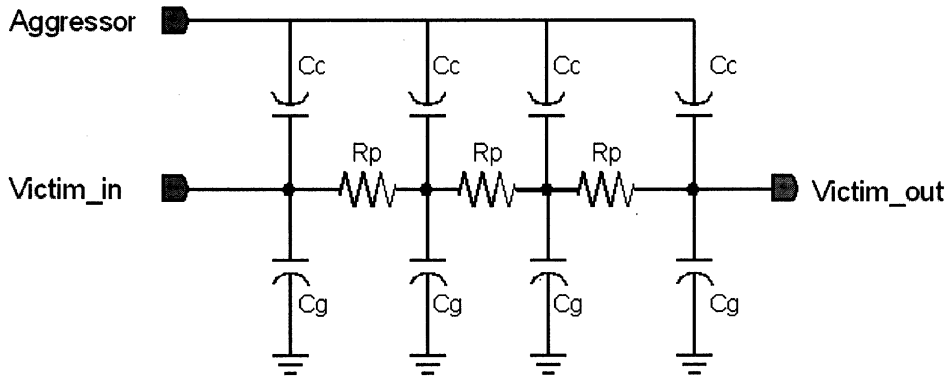


Figure 5-20 Parasitic wire capacitances and resistances

Long interconnection wires are very common in programmable logic. In order to minimize the effect of parasitic wire capacitance, several considerations were included in OPL-PLC design.

Minimizing wire load: Parasitic wire capacitance introduces more wire load for the driving gates. The layout of OPL-PLC was therefore made as compact as possible to decrease the wire length and therefore wire load. Estimated wire loads were added into the schematic level simulation during design iterations to ensure more accurate gate sizing.

Avoiding cross talk: Cross talk is an important noise source, especially for dynamic circuits. Adding a sufficiently sized half-keeper at the output of a dynamic gate

helps prevent functional failures due to the cross talk. The keepers for all the gates in OPL-PLC were carefully sized for functionality as well as performance. In layout, *Cc* is proportional to the spacing of two neighboring wires. By increasing the wire spacing to three times that of minimum allowable spacing, coupled capacitance for the parallel wires on the same layer is decreased by 4X and the cross talk effect is greatly reduced. Clocks in OPL-PLC have very long wires, which are sensitive to cross talk. In the layout, all the clock wires are laid out away from other long wire signals. Besides all the above considerations, there is also cross talk between parallel wires on two neighboring layers. In OPL-PLC, layer metal 3 to layer metal 6 are used for global signal routing, which are sensitive to cross talk. We made signal routing on neighboring layers perpendicular to avoid the cross talk effect.

5.4.2 Parasitic Resistance

There are parasitic resistances in the interconnection wires, as shown in Figure 5-20. Current flowing through a resistive wire results in an ohmic voltage that degrades the signal levels, generally called *IR drop*. This is especially important in the power-distribution network [14]. IR drop affects not only circuit reliability but also circuit speed.

In OPL-PLC, power grids were added above the circuit layout to decrease the IR drop in Vdd and Gnd, as shown in Figure 5-21. Layer metal 5 (bottom layer in the figure) and metal 6 (top layer in the figure) were used to implement power grid because of the smaller series unit resistance.

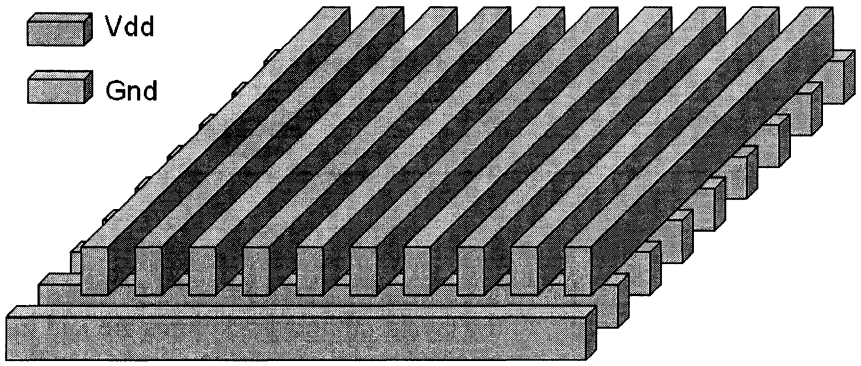


Figure 5-21 Power grids for Vdd and Gnd

In addition to the power grid, on-chip decoupling capacitors were extensively inserted everywhere in the layout to decrease the transient voltage drop [53]. MOS capacitance has high density in a CMOS process, which is adopted in our design. The total area of on-chip decoupling capacitances in OPL-PLC is about 20% of the total chip area.

The IR drop in OPL-PLC was simulated post-layout, where the parasitic resistances and capacitances in the layout were extracted. Due to the limitation of simulation time, a worst-case scenario was considered in IR drop analysis: only parasitic resistances on Vdd and Gnd nets were considered, and only parasitic capacitances for the signal nets were included in the netlist. With the power grid and decoupling capacitances, the lowest value of Vdd dropped to 1.7V and Gnd bounced up to less than 0.2V. This caused the circuit delay to degrade about 7.2%. As we reported earlier, the worst-case delay of one TLS in post-layout simulation is 642ps, which doesn't include IR drop. After considering the non-ideal power supply, the delay is changed to 688ps.

5.4.3 Parasitic Inductance

Besides having parasitic resistance and capacitance, interconnect wires also exhibit an inductive parasitic [50]. An important source of parasitic inductance is introduced by the bonding wires and chip package. A bond wire with a 1mm length introduces 1nH of inductance. During each switching action, a transient current charges or discharges the circuit capacitance, as shown in Figure 5-22. Due to $L\frac{di}{dt}$ voltage drop, there is a voltage difference between the external and internal supply voltages.

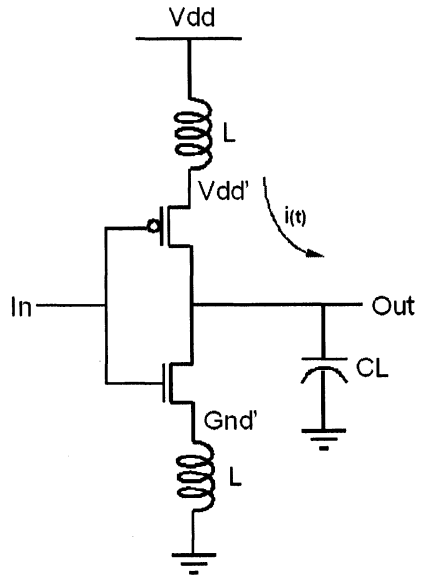


Figure 5-22 Inductive coupling between external and internal supply voltages

Several approaches were applied in OPL-PLC to decrease the effect of $L\frac{di}{dt}$ voltage drop. One important approach is adding on-chip and off-chip decoupling capacitors. The bypass capacitor, combined with the inductance,

actually acts as a low-pass network that filters away the high frequency components of the transient voltage spikes on the supply lines. As we mentioned earlier, 20% of the total circuit area is on-chip decoupling capacitances. Off-chip capacitors were soldered on the printed-circuit board to filter out the off-chip noise, as shown in Figure 5-13.

Decreasing the value of L is another solution. One way of decreasing power supply inductance due to bond wires is by using multiple power and ground pins. If one bond wire has 1nH inductance, N identical bond wires in parallel make the effective inductance to be $1/N$ nH. 20 pairs of Vdd and Gnd pins were used in OPL-PLC.

The $L\frac{di}{dt}$ voltage drop in OPL-PLC was simulated at the post-layout level by adding estimated bond wire inductance. There was no speed degradation due to the careful layout and on-chip decoupling capacitances. From the measured results, it also presents that by carefully designing a test board, the off-chip inductance can be minimized, such that inductive effects are negligible.

5.5 Summary

In this chapter, the testing structure of OPL-PLC was discussed in detail. Because of the high-speed performance, some special test methods were included. We have shown that for the 0.18μm/1.8V standard CMOS process, the throughput of OPL-PLC can reach 1 GHz, which is 2.4 times of the frequency of Xilinx CoolRunner-II CPLD [49] and 3.2 times of the frequency of Xilinx Virtex-E FPGA (Speed Grade -7) [50]. Mapping common data path circuits to this architecture suggests that an average speedup of 4.1 times over Xilinx Virtex-E FPGA in measurement is attainable. These figures support the premise that PLC blocks can be used alongside

aggressively designed microprocessors and ASICs to provide a flexible computational resource with low latency and high throughput. We also analyzed the parasitic effects on circuit performance as well as the sources of speed difference between post-layout simulation and measurement results.

Chapter 6 : Conclusions and Future Work

6.1 Summary of This Work

Embedding the programmable logic cores into the SoC design becomes more and more desirable due to their capability of post-fabrication changes. They make the SoC design more cost-effective and time-to-market faster. In this dissertation, we investigated the feasibility of generating a programmable logic core for SoC applications with speed comparable to ASIC designs. We gave elaborate discussions on both OPL, the underlying circuit technique of OPL-PLC, and the architecture of OPL-PLC. We also fabricated a prototype chip to verify our idea. The speed of the OPL-PLC test chip was measured and the energy consumption was analyzed.

The main contribution of this work was the introduction of a novel high-performance CMOS programmable logic core. This thesis described several critical architectural innovations utilized by OPL-PLC. These are reviewed next.

(1) Utilization of the highest performance dynamic logic style (OPL) as the underlying circuit technique. OPL-PLC is the first programmable logic core implemented with the dynamic logic family, OPL. Like pseudo-nMOS, OPL is well suited for implementing wide NOR gates, which are used extensively in OPL-PLC. In addition to being dramatically faster than pseudo-nMOS, OPL avoids static power dissipation, reducing the energy consumption of OPL-PLC.

(2) The development of an improved product-term-based logic structure having unprecedented flexibility. In [20], experimental results demonstrated that for many functions, multiple-level networks are more efficient and may require fewer levels of logic overall compared to two-level AND-OR networks. In OPL-PLC, like the

conventional product-term-based structure, OPL-PLC is well suited for wide logic implementation. Unlike the conventional product-term-based structure, which has low logic density when implementing small functions, OPL-PLC includes two logic levels in the AND-plane as well as two levels in NOR-plane. This compact structure makes OPL-PLC more suitable for implementing small functions. It reduces logic levels and therefore circuit delay and increases logic density for small function implementations.

(3) Combined logic and routing. In programmable logic, routing occupies a lot of area and delay. The mixed logic and routing structure used in OPL-PLC increases circuit speed and logic density, in addition to saving area and energy.

(4) Extensive use of wired-NOR. The wires in the wired-NOR structures used in OPL-PLC may be quite long. Being a "dispersed" logic resource, a wired-OR is able to perform logic on widely separated inputs, while at the same time providing an output to possibly several widely separated destinations. The wired-OR eliminates the need for wiring signals to a "localized" logic resource. An additional advantage of the wired-NOR structure is that it can also serve as a routing resource.

(5) Adoption of a unidirectional routing flow. In order to simply the routing structure and enable fast interconnects, in OPL-PLC, a unidirectional routing flow was adopted. Since OPL-PLC is based on a dynamic logic technique, this routing architecture also simplifies clock distribution.

Besides the above innovations, the novel OPL-PLC was designed for a standard CMOS process, enabling a low manufacturing cost as well as a high integration density. In addition, SRAM cells were chosen to store configuration bits to enable design change on the fly.

OPL-PLC provides up to 1 GHz throughput, which is 2.4 times that of the Xilinx CoolRunner-II CPLD [49] and 3.2 times that of the Xilinx Virtex-E FPGA (Speed Grade -7) [50]. Measurement results showed that this architecture achieves an average speedup of 4.1 times (fastest delay) and 3.7 times (typical delay) over Xilinx Virtex-E FPGA (Speed Grade -7) [50] while consuming similar energy. The overall EDP of OPL-PLC is only 19% of the EDP of XilinxVirtex-E FPGA. All the results above demonstrated that OPL-PLC achieves better overall performance compared to currently available cutting–edge commercial programmable logic and can be used in SoC applications without worrying about the speed gap between programmable logic and fixed logic.

6.2 Future Work

There are still some possible improvements for our current OPL-PLC architecture, which will be the future work of this project.

First, the current architecture is not suitable for implementing large digital systems. Since this was just the OPL-PLC prototype, the current size was large enough to verify the concepts. When extending the scale of OPL-PLC, both horizontal and vertical directions have to be considered. It is easy to extend the levels of logic in series by adding more TLSs. However, as we explained in chapter 4, there are only 8 HLRBs in parallel in our prototype chip, which is not enough for much more complicated logic. More TLSs need to be connected in parallel. It is not as easy as extending TLSs in series, and a special routing architecture needs to be developed.

Figure 6-1 shows a possible routing structure for communicating signals vertically. In both case1 and case2, there are 3 TLSs in parallel. Instead of having only 64 global tracks in each group, we add 64 redundant tracks in the revised routing structure. In Figure 6-1, the tracks without connections on left side are redundant.

Each rectangular box represents a switch box between two neighboring TLSs to route the signals. The pass transistors driven by static inverters are used to make connections between signal track and redundant track (unidirectional) or between two redundant tracks (bi-directional).

When mapping the circuit onto OPL-PLC, we need to pay attention to the congestion problem. As shown in Figure 6-1 case1, signal A and B are passed to track 2 in the upper TLS at the same time, which results in congestion problem. By changing the position of signal A (shown in case2), the congestion problem is solved.

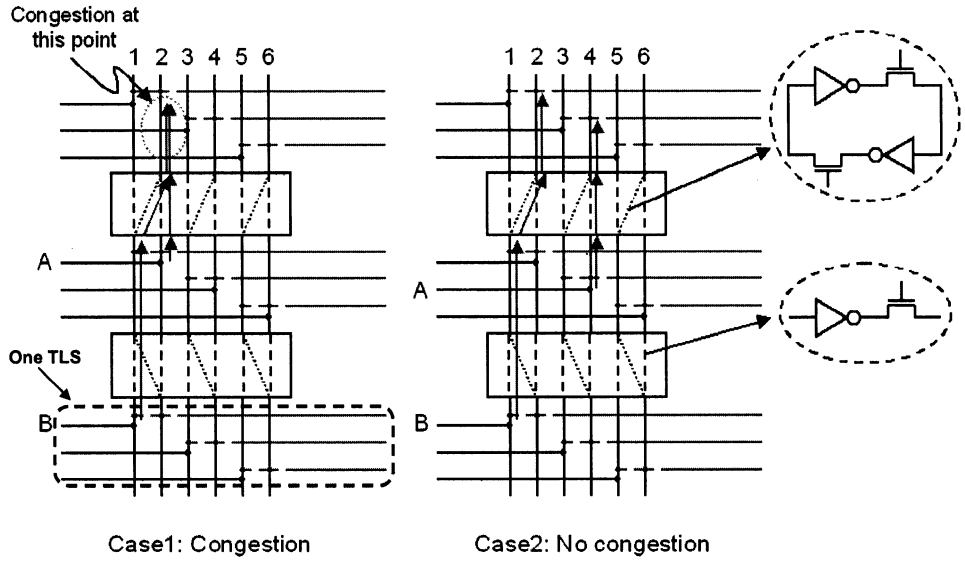


Figure 6-1 Suggested routing structure for vertical routing

From our simulation, the delay of passing a signal through one switch (one static inverter followed by one pass transistor) is about 100ps, which means we need to increase the clock separation time between the last gate of the previous TLS and first gate of the current TLS.

Although the larger scale of OPL-PLC makes it suitable for implementing more complicated circuits, the latency of the implemented circuit is increased due to the presence of switches. The more switches the signal passes, the longer latency the circuit has. So in order to minimize the latency of the circuit, during the mapping, we need to make the signals pass the switches as less as possible.

After enlarging the OPL-PLC to a larger scale, in order to use it efficiently, we can make the input clock trigger to each TLS programmable. As shown in Figure 6-2, there is a 2-1 MUX in front of each TLS. By changing the control signal of that MUX, each TLS can be triggered either by the primary global clock, or by the clock derived from the previous TLS. By adding this feature, OPL-PLC can work either in parallel or in series.

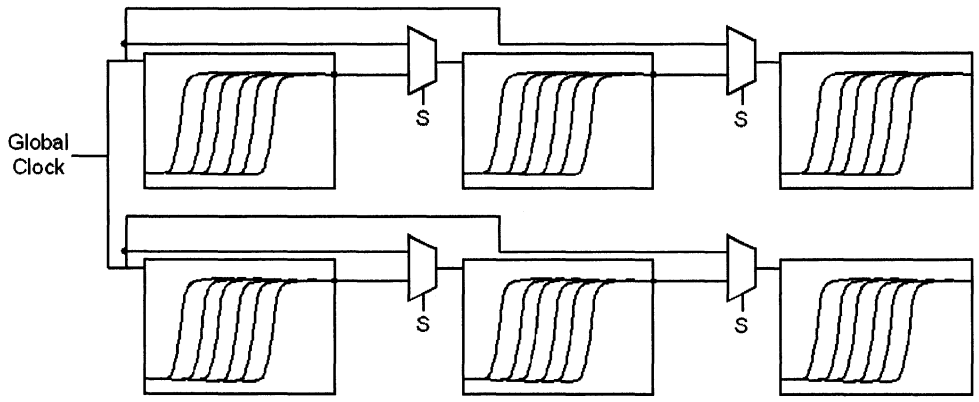


Figure 6-2 Programmable input clock trigger for each TLS

Second, the energy consumption of OPL-PLC can be further improved. Although the energy of the current OPL-PLC is similar to that of the Xilinx Virtex-E FPGA (while being several times faster), it is still possible to improve its energy consumption. We already discussed the possible ways of improvement in chapter 4 under the power consumption section.

Clock gating is also a good method to save energy on unused blocks. By disabling the clock of unused logic and routing gates, those unused gates together with the clock buffers driving those gates won't switch and energy will be saved.

Figure 6-3 shows how to implement clock gating in OPL-PLC. All the gates inside the red box are disabled when the gate in the box is not used in the circuit mapping. Although clock gating introduces a few more transistors, it saves significant power when many OPL gates are not used in the circuit mapping.

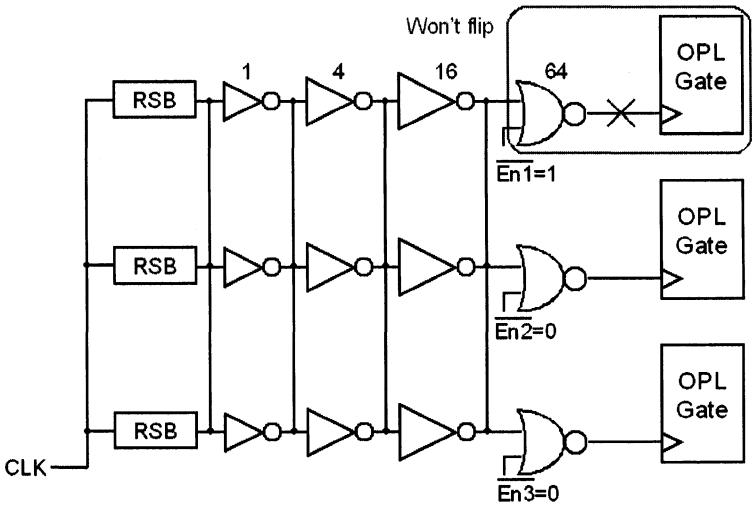


Figure 6-3 Clock gating in OPL-PLC

By making those improvements, we believe a better energy-delay product will be achieved.

Third, OPL-PLC is based on a dynamic logic style. The unidirectional routing structure was adopted for higher speed and the simplicity of routing and clock distribution. The unidirectional routing structure works very well for mapping data path circuits. There is one limitation comes together with all other advantages: it is

impossible to implement sequential logic in the current OPL-PLC design due to the lack of a feedback path. In the future, additional routing structure can be developed based on our current design to provide a feedback path in OPL-PLC.

Figure 6-4 shows one possible way to implement the feedback path in OPL-PLC. In each group of long track, the solid lines represent signal tracks, and the dashed lines represent redundant tracks. The signals generated on the third group of long tracks are feedback to the previous groups of long tracks through the tracks with static buffers (for speed purpose) and pass transistors (for programming purpose). Each feedback signal is passed to 4 different redundant tracks in the previous group for routing flexibility. The feedback path with different length are available for routing flexibility as well.

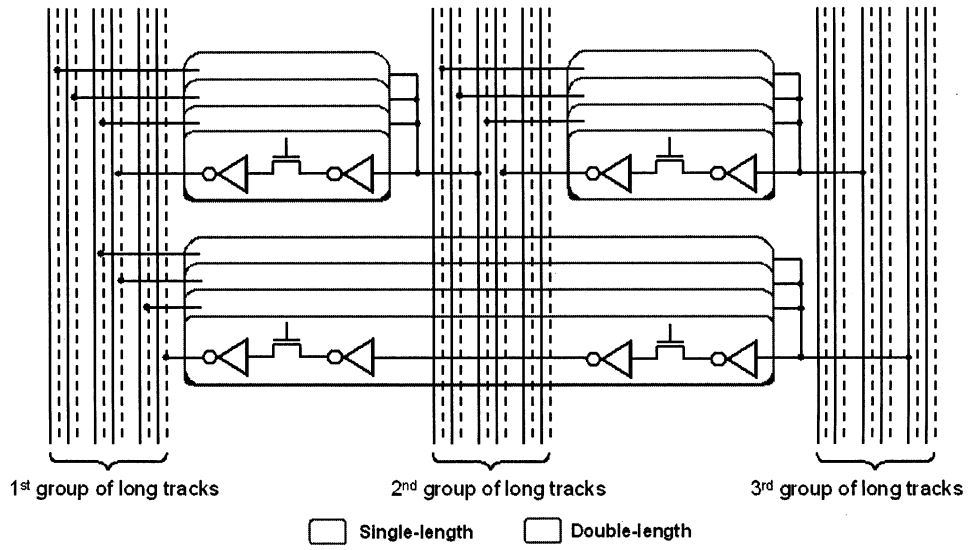


Figure 6-4 Feedback path in OPL-PLC

In order to implement the sequential logic correctly, there are timing constrains. An example is shown in Figure 6-5. There are 6 OPL gates in series and the output of

Gate6 has to be feedback to Gate1. In order to make Gate1 obtain correct feedback value before next clock cycle, two timing restrictions has to be met.

t_d >= delay of Gate6 + delay of feedback path (6-1)

T >= delay of Gate1 to Gate6 + delay of feedback path (6-2)

From the above two equations, we know that the longer the feedback path, the longer the clock period. This is true for all the sequential logic implementations, since the first gate has to wait for the feedback signal before next clock cycle, no matter the logic is implemented in static or dynamic gates. From our previous comparison, the latency of OPL-PLC is much shorter than some other commercial programmable logic devices, therefore the clock period will also be shorter as long as the delay of feedback path is not dominant.

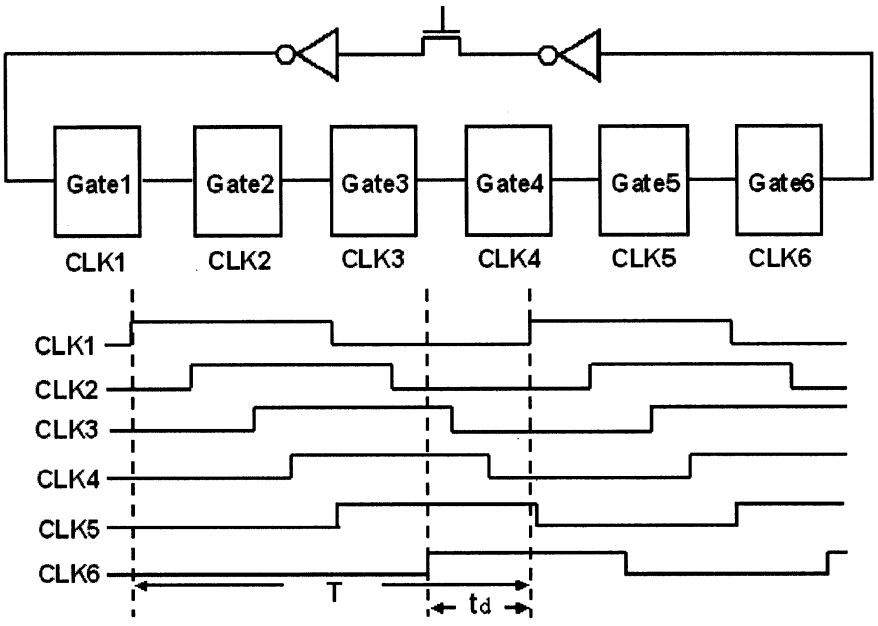


Figure 6-5 Timing issue of OPL-PLC with feedback path

Fourth, a mapping algorithm for mapping the circuits into OPL-PLC needs to be developed in the future. In our current experimental results, all the circuits were manually mapped into OPL-PLC, which is time-consuming. A CAD tool to automatically map the circuits will shorten the whole implementation procedure. In [65], they provided an algorithm to map circuits into a two-level-structure-based architecture with minimized delay. That two-level-structure-based architecture is similar to OPL-PLC with less flexibility. Therefore, this algorithm can be used as the starting point of the mapping tool developed for OPL-PLC.

According to the architectural characteristics of OPL-PLC, several points need to be included in the mapping algorithm. (1) Collapse the network into AND-OR style. (2) If there are two-input functions inside AND, don't collapse them, but treat them as a group and implement in MUX. (3) Implement the equations with same inputs or same product terms in the same HLRB. (4) If the equations share more than 4 product terms, use the second-level wired-NOR4 to share all the common product terms. (5) If there's NOR function with small number of product terms, separate the NOR-plane (16 inputs) into several NORs (with less number of inputs) for better logic density.

All the points listed above work well for general case. For some specific cases, more considerations might be included.

REFERENCES

[1] C. Matsumoto, "LSI Logic ASICs to Add Programmable Logic Cores", *E.E. Times*, August, 1999.

[2] S. Ohr, "ADI Taps Systolix Processor Array", *E.E. Times*, April, 2000.

[3] "Lucent Introduces ORCA Series 4 FPGA", Programmable Logic News and Views, pp. 7-11, 2000.

[4] R. Merritt, "Quick Logic Steps up Merger of FPGA with IP Cores–DSP First Target", *E.E. Times*, August, 2000.

[5] "Embedded FPGA Cores Enable Programmable ASICs, ASSPs", *E.E. times*, September, 1999.

[6] C. Matsumoto, "Actel Plans to Produce FPGAs as ASIC Cores", *E.E. Times*, June, 2000.

[7] C. Matsumoto, "Startup Puts a Fresh Spin on Programmable Cores", *E.E. Times*, September, 2000.

[8] S. Wilton and R. Saleh, "Programmable Logic IP Cores in SoC Design: Opportunities and Challenges," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 63-66, May 2001.

[9] J. Wu, V. Aken'Ova, S. Wilton, R. Saleh, "SoC Implementation Issues for Synthesizable Embedded Programmable Logic Cores", in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 45-48, September 2003.

[10] N. Kafafi, K. Bozman and S. Wilton, "Architecture and Algorithms for Synthesizable Embedded Programmable Logic Core," in *Proceedings of the ACM International Symposium on Field-Programmable Gate Arrays*, pp.1-9, February 2003.

[11] J. Greenbaum, "Reconfigurable logic in SoC systems," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 5-8, May 2002.

[12] T. Thorp, G. Yee and Carl Sechen, "Monotonic Logic and Dual Vt technology", in *Proceedings of the 1999 international symposium on Low power electronics and design,* pp. 151-155, 1999.

[13] Z. Ye, A Moshovos, S. Hauck and P. Banerjee, "CHIMAERA: A High-Performance Architecture with a Tightly-Coupled Reconfigurable Functional unit", in *Proceedings of the 27th International Symposium on Computer Architecture*, pp. 225-235, 2000.

[14] J. M. Rabaey, A. Chandrakasan, B. Nikolic, "Digital Integrated Circuits: A Design Perspective (Second Edition)", Pearson Education Inc., 2003.

[15] S. Brown and J. Rose, "Architecture of FPGA and CPLDs: A Tutorial", in *IEEE Transactions on Design and Test of Computers*, Vol. 13, No. 2, pp. 42-57, 1996.

[16] J. F. Wakerly, "Digital Design: Principles and Practices", Prentice Hall, 1994.

[17] R.C. Seals, G.F. Whapshott, "Programmable Logic: PLDs and FPGAs", a Division of McGraw-Hill Companies, 1997.

[18] S. Hauck, "Multi-FPGA Systems", Ph.D Dissertation, University of Washington, Seattle, Washington, 1995.

[19] Z. Zilic, G. Lemieux, K. Loveless, S. Brown, Z. Vranesic, "Designing for High Speed-Performance in CPLDs and FPGAs", in *Proceeding of 3rd Canadian Workshop Field-Programmable Devices*, pp. 108-113, 1995.

[20] T. Sasao, M. Fujita, Eds. "OR-AND-OR three-level networks", in *Representations of Discrete Functions*, Norwell, MA: Kluwer, 1996.

[21] D. Debnath, Z. G. Vranesic, "A Fast Algorithm for OR-AND-OR Synthesis", in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 22, No. 9, pp. 1166-1176, September 2003.

[22] V. Ciriani, "Logic Minimization using Exclusive OR Gates", in *Proceeding of IEEE/ACM 38th Design Automation Conference*, pp. 115-120, 2001.

[23] E. Dubrova, P. Ellervee, "A Fast Algorithm for Three-Level Logic Optimization", in *Proceeding of IEEE/ACM International Workshop Logic Synthesis*, pp. 251-254, 1999.

[24] M. Karpovsky, "Multilevel Logical Networks", in *IEEE Transaction on Computer*, Vol. C-36, pp. 215-226, 1987.

[25] A. A. Malik, D. Harrison, R. K. Brayton, "Three-level Decomposition with Application to PLD's", in *Proceeding of IEEE International Conference of Computer Design*, pp. 628-633, 1991.

[26] T. Sasao, "Logic Synthesis with EXOR Gates", in Logic Synthesis and Optimization, Norwell, MA: Kluwer 1993.

[27] D. Warma, E. A. Trachtenberg, "Design Automation Tools for Efficient Implementation of Logic Functions by Decomposition", in *IEEE Transaction on Computer-Aided Design*, Vol. CAD-8, pp.901-916, 1989.

[28] V. George, "Low Energy Field-Programmable Gate Array", Ph.D Dissertation, University of California, Berkeley, 2000.

[29] http://direct.xilinx.com/bvdocs/publications/ds022-2.pdf

[30] http://www.altera.com/

[31] D. Harris, "Skew-tolerant circuit design," Ph. D. dissertation, Stanford University, CA, 1999.

[32] F. Klass, C. Amir, A. Das, K. Aingaran, C. Truong, R. Wang, A. Mehta, R. Heald, G. Yee, "A new family of semi-dynamic and dynamic flip-flops with embedded logic for high-performance processors", in *IEEE Journal of Solid-State Circuits*, Vol. 34, No. 5, pp. 712-716, May 1999.

[33] B. Benschneider, A. J. Black, W. J. Bowhill, S. M. Britton, D. E. Dever, D. R. Donchin, R. J. Dupcak, R. M. Fromm, M. K. Gowan, P. E. Gronowski, M. Kantrowitz, M. E. Lamere, S. Mehta, J. E. Meyer, R. O. Mueller, A. Olesin, R. P. Preston, D. A. Priore, S. Santhanam, M. J. Smith, G. M. Wolrich, "A 300-MHz 64-b quad-issue CMOS RISC microprocessor", in *IEEE J. Solid-State Circuits*, Vol. 30, No. 11, pp. 1203-1214, Nov. 1995.

[34] A. Charnas, A. Dalai, P. deDood, et el., "A 64b microprocessor with multimedia support", in *International Solid State Circuits Conference (ISSCC) Digest of Technical Papers*, pp. 177-179, Feb. 1995.

[35] R. P. Colwell, R. L. Steck, "A 0.6um BiCMOS processor with dynamic execution", in *International Solid State Circuits Conference (ISSCC) Digest of Technical Papers*, pages 176-177, February 1995.

[36] T. Thorp and C. Sechen, "Design and Synthesis of Dynamic Circuits," in *IEEE Transaction on VLSI Systems*, Vol. 11, pp. 141-149, February 2003.

[37] L. McMurchie, S. Kio, G. Yee and C. Sechen, "Output prediction Logic: A High-Performance CMOS Design Technique", in *Proceedings of the IEEE International Conference Computer Design,* pp. 247-254, September 2000.

[38] S. Kio, L. McMurchie and C. Sechen, "Application of Output Prediction Logic to Differential CMOS", in *Proceedings of the IEEE Computer Society Workshop on VLSI,* pp. 57-65, April 2001.

[39] S. Sun, L. McMurchie and C. Sechen, "A High-Performance 64-bit Adder Implemented in Output Prediction Logic", in *Proceedings of the Conference on Advanced Research in VLSI,* pp. 213-222, March 2001.

[40] N. Weste, D. Harris, "CMOS VLSI Design: A Circuit and System Perspective", Pearson Education, Inc., 2004.

[41] G. Yee and C. Sechen, "Clock-delayed Domino for Adder and Random Logic Design," *Proceeding of IEEE International Conference on Computer Design (ICCD),* Austin, TX, October, 1996.

[42] Z. Zhu, B. S. Carlson, "Critical Voltage Transition Logic: An Ultrafast CMOS Logic Family", in *Proceeding of IEEE International Conference on Computer Design (ICCD)*, Austin, TX, October 1997.

[43] J. kouloheris and A. Gamal, "PLA-based FPGA Area Versus Cell C+ Granularity," in *Proceedings of IEEE Custom Integrated Circuits Conference (CICC)*, pp. 4.3.1-4.3.4, May 1992.

[44] A. Yan and S. Wilton, "Product-Term based Synthesizable Embedded Programmable Logic Cores", in *Proceedings of IEEE International Conference on Field-Programmable Technology*, pp. 162-169, December 2003.

[45] J. Cong, H. Huang and X. Yuan, "Technology Mapping for k/m-macrocell Based FPGAs", in *Proceedings of Eighth ACM International Symposium on FPGAs (FPGA'2000),* pp. 51-59, February 2000.

[46] G. Borriello, C. Ebeling, S. Hauck and S. Burns, "The Triptych FPGA Architecture", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* Vol. 3, No. 4, Dec. 1995, pp.491-501.

[47] K. Zhou, Channakeshav, M. Chu, J. Guo, S. Liu, R. Kraft, C. You, J. McDonald, "Gigahertz SiGe BiCMOS FPGAs with new architectures and

novel power management schemes", in *Proceedings of IEEE International Conference on Field-Programmable Technology*, pp.182-188, December 2002.

[48] S. Kio, K.H. Chong, and C. Sechen, "A Low Power Delayed-Clocks Generation and Distribution System", in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '03),* pp. 445-448, May 2003.

[49] http://direct.xilinx.com/bvdocs/publications/ds090.pdf, July, 2004, pp. 1.

[50] http://direct.xilinx.com/bvdocs/publications/ds022-1.pdf, July, 2002, pp. 3.

[51] A. Singh, A. Mukherjee, L. Macchiarulo and M. Marek-Sadowska, "PITIA: An FPGA for Throughput-Intensive Applications", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* Vol. 11, No. 3, June 2003, pp.354-363.

[52] http://www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=dr_dt_xpower

[53] Q. K. Zhu, "Power Distribution Network Design for VLSI", John Wiley&Sons, Inc., 2004.

[54] HSIM manual, Nassda Corporation.

[55] C. Souza, "Atmel Plugs in Customizable SoC", *E.E. Times*, September 22, 2003.

[56] A. Cataldo, "FPGAs Morph on Path to SoC Designs", E.E. Times, September 13, 2004.

[57] 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," *IEEE Journal of Solid State Circuits*, Vol. 25, No. 5, October 1990, pp. 1217-1225.

[58] K. Compton, S. Hauck, "Totem: Custom Reconfigurable Array Generation", *IEEE Symposium on FPGAs for Custom Computing Machines Conference*, 2001.

[59] K. Compton, A. Sharma, S. Phillips, S. Hauck, "Flexible Routing Architecture Generation for Domain-Specific Reconfigurable Subsystems", *International Symposium on Field Programmable Logic and Applicaions*, 2002.

[60] S. Phillips, S. Hauck, "Automatic Layout of Domain-Specific Reconfigurable Subsystems for System-on-a-Chip", *ACM/SIGDA Symposium on Field-Programmable Gate Arrays*, 2002.

[61] "UBC lecture: APSC 380: Introduction to Microcomputers", Instructor: Ed Casas, 1999.

[62] W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, G. Varghese, J. Wawrzynek, A. DeHon, "HSRA: High-Speed, Hierarchical Synchorous Reconfigurable Array", in *Proceedings of the ACM International*

Symposium on Field-Programmable Gate Arrays, pp. 125-134, February 1999.

[63] D. P. Singh and S. D. Brown, "The Case for Registered Routing Switches in Field Programmable Gate Arrays", in *Proceedings of the ACM International Symposium on Field-Programmable Gate Arrays*, pp. 161-169, February 2001.

[64] N. Weaver, J. Hauser, J. Wawrzynek, "The SFRA: a Corner-Turn FPGA Architecture", in *Proceedings of the ACM International Symposium on Field-Programmable Gate Arrays*, pp. 3-12, February 2004.

[65] J. Ciric, G. Yee, C. Sechen, "Delay Minimization and Technology Mapping of Two-Level Structures and Implementation Using Clock-Delayed Domino Logic", in *Proceedings of theDesign, Automation and Testin Europe (DATE' 2000),* pp. 277-282, March 2000.

[66] A. Mukherjee, M. Marek-Sadowska, S.I. Long,"Wave Pipelining YADDs—A Feasibility Study", in *Proceedings of IEEE Custom Integrated Circuits Conference*, pp. 559-562, May 1999.

[67] A. Mukherjee, R. Sudhakar, M. Marek-Sadowska, S.I. Long, "Wave Steering in YADDs: A Novel Non-iterative Synthesis and Layout Technique", *in Proceedings of IEEE Design Automation Conference*, pp. 466-471, June 1999.

[68] M. Holland, S. Hauck, "Automatic Creation of Reconfigurable PALs/PLAs for SoC", in *Proceedings of International Symposium on Field-Programmable Logic and Applications*, pp. 536-545, 2004.

VITA

Education

1996 – 2000	B.S., Microelectronics Peking University, Beijing, China
2000 – 2002	M.S.E.E., Electrical Engineering University of Washington, Seattle, WA
2002 – 2004	Ph.D., Electrical Engineering University of Washington, Seattle, WA

Honors

Ao De scholarship, Peking University, 1999

Prize of Excellent Social Work, Peking University, 1999

Canon Scholarship, Peking University, 1998

Outstanding Undergraduate Student, Peking University, 1998

Legend Scholarship, Peking University, 1997

Excellent Member of Social Work Community, Peking University, 1997

Excellent Cadre of Peking University, Peking University, 1997