Universität Stuttgart

Institute of Computer Architecture and Computer Engineering

University of Stuttgart
Pfaffenwaldring 47
D–70569 Stuttgart

Master's Thesis Nr. 3505

Delay Characterization in FPGA-based Reconfigurable Systems

Shihao Zhang

Course of Study:	Information Technology / InfoTECH
Examiner:	Prof. Dr. rer. nat. habil. Hans-Joachim Wunderlich
Supervisors:	M.Sc. Francesco Cervellera Dipl.-Inf. Michael Imhof Dipl.-Inf. Michael Kochte
Commenced:	2013-06-03
Completed:	2013-12-03
CR-Classification:	B.6.1, B.8.1

Acknowledgement

I would like to thank, first and foremost, Prof. Dr. rer. nat. habil. Hans-Joachim Wunderlich for the opportunity to write this thesis at the Institute of Computer Architecture and Computer Engineering, University of Stuttgart.

This thesis would not have been possible without the help of many people. I would like to express the deepest gratitude to my supervisors, Dipl.-Inf. Michael Kochte, Dipl.-Inf. Michael Imhof and M.Sc. Francesco Cervellera. Their insightful viewpoint and continuous support have offered immense help during my work.

In addition, I am very grateful for all of my friends, whose offered suggestions and critiques on this thesis. Last but not least, I would like to give my very special thanks to my family and Huawen Dai, for their continuous love, encouragement and support.

Abstract

Runtime reconfigurable architectures accelerate the operation of a standard processor core by hardware accelerators implemented in Field Programmable Gate Arrays (FPGAs). Partial runtime reconfiguration allows the hardware accelerators to efficiently adapt to different computational tasks dynamically. Nowadays, the FPGAs from major vendors, such as Xilinx and Altera, support this feature, including the Xilinx Virtex-5 FPGA family which is the implementation platform of this work.

Manufactured at 28 nm scaled technological node or lower, concerns rise about the impact of aging-related failure mechanisms on the modern generations of FPGAs. To detect degradation in the reconfigurable gate arrays, dedicated on- and offline test methods must be employed in the field. Design for dependability requires that the degradation is detected and localized, so that the degraded logic elements will not be used as a first choice in the reconfiguration.

This thesis presents the development and the evaluation of a delay characterization method for FPGA CLBs which comprise most of the FPGA logic elements. The purpose of FPGA delay characterization method in this work is to detect and localize the delay variance. This delay variance information may be used for achieving a speed optimized reconfiguration for a FPGA-based runtime system. Different delay characterization methods have been studied in this thesis for determining a suitable method to be implemented in the partial reconfigurable system. The delay characterization is performed in a part of area in the FPGA before a module is placed in this area to avoid the degraded portion. This thesis uses low level hardware description language to generate the fine-grained measurement units which can cover the target area. VHDL is used to generate the test wrapper, control circuit, and the circuit for communicating between the FPGA and the workstation. Several measurement techniques are used to evaluate the accuracy of the delay characterization method. Additionally, this thesis evaluates the temperature influence on the delay characterization.

The results show that this delay characterization method can compare the speed of logic elements in the partial runtime reconfiguration area with high accuracy. The degradation can be detected and localized. The results also show that this method can be adapted to different size and location, fitting in the partial runtime reconfigurable design. Twelve configurations are required to have a full coverage of all the CLBs in the area under test.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Motivation

Reconfigurable systems have been developed and studied in the past two decades. The configuration-ability after fabrication permits the reconfigurable system to adapt to different requirements of the applications by programming the logic elements and the connection among them. Reconfigurable systems require reconfigurable hardware platforms, such as Field Programmable Gate Arrays (FPGAs). An FPGA has pre-fabricated integrated circuits that can be programmed by the designer. To date, the modern FPGAs from the major FPGA vendors have the ability to be partially reconfigured at runtime while the remaining parts continue operating without any data loss or interruption [1, 2]. This feature is called Partial Runtime Reconfiguration and it helps to reduce the power consumption or improve the performance [3]. For example, some applications use this feature of FPGAs to implement hardware application specific accelerators to accelerate the operation of a standard processor core by dynamically replacing the hardware modules to handle the different application tasks requested by the operating system [4]. Due to the flexibility, high performance and power saving, runtime reconfigurable systems are becoming attractive for a wide range of applications.

When FPGAs are manufactured at 28 nm scaled technological node or lower, concerns rise about the impact of aging mechanism like Hot Carrier Effect [5], Time Dependent Dielectric Breakdown (TDDB) [6], Electromigration [7], and Negative Bias Thermal Instability (NBTI) [8]. These degradation mechanisms will slow down the speed of the FPGA fabric, or in the worst case can affect the FPGA lifetime and dependability [9]. For example, electromigration will cause a wear-out failure of interconnect by generating voids or shorts in the metal lines due to the high current density.

The speed of aging in the FPGA depends on different factors including switching activity. The aging effect due to switching activity is presented in [10]. The authors introduced an aging model that shows the proportional relation between aging and the switching rate of the circuit. Because every design on the FPGA is different, the workload of each component inside the FPGA is not the same. Different components of the FPGA have various switching currents and leakage currents. If a design is implemented on the FPGA for a long period of time and certain logic components are configured to have more computational activities than others, the impact of aging on these component is more significant so that different parts of the FPGA will have different aging speed.

To prevent the FPGA from the reliability threat caused by these aging mechanism, many test strategies have been discussed [11, 12]. These online test strategies can be applied to check that the reconfigurable fabric is fault free before the module reconfiguration is performed on the FPGA. The test just provides the pass/fail results, but it does not consider timing explicitly. When the transition on a certain path is too slow, it means the path is faulty. However, when the circuit under test passes the test, the delay of the circuit remains unknown. It is necessary to find out that if the degradation caused by the aging effect affects the reconfigurable fabric and localize it.

If we can find out the delay variation in the FPGA, then with the ability of reconfiguration, different placement and routing methods can be instantiated to avoid the aged area. Consequently, with Partial Runtime Reconfiguration, the delay variation in the FPGA chip can be considered before the circuit is mapped in the FPGA to utilize the actual speed of the available hardware resource and to increase the device reliability with delay-aware placement and routing methods [13, 14, 15].

To detect the degradation and provide the delay information in the FPGA, a method that can characterize the delay of the logic elements in a specific area on the FPGA is needed.

1.2 Goal and Objectives of this work

The goal of this thesis is to develop a delay characterization method for the reconfigurable logic element in the runtime reconfigurable system. The purpose of this FPGA delay characterization method is to detect and localize the degradation of the reconfigurable logic element so that with the spatial delay variation information the runtime reconfigurable systems can select the elements with relatively low delay to implement the design for better performance and reliability.

A delay characterization is performed in the partial runtime reconfigurable area of the FPGA before a module is placed as shown in Figure 1.1. The method has to provide the delay of each partition in this area. By comparing the delay characterization results, the system can select the area with appropriate performance to implement the module which has a specific demand for speed.

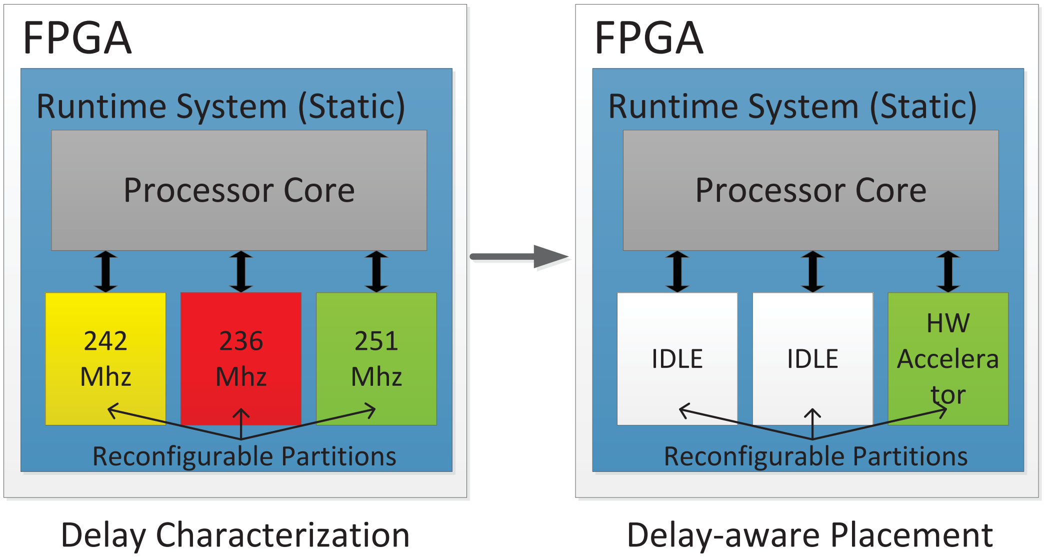


Figure 1.1: The delay characterization method applied on a runtime reconfigurable system

To achieve the goal of this thesis, different delay characterization methods from the literatures have been studied for determining a suitable method in the partial reconfigurable system. A simulation model is required to evaluate the method before the implementation of infrastructure on the FPGA board. The result analysis will be performed on the workstation and the effectiveness of the technique will be evaluated.

1.3 Thesis Organization

After the introduction, the remainder of the thesis is structured as follows. Chapter 2 briefly introduces the background information. Then, the state of art of delay characterization methods is reviewed in Chapter 3. The architecture of the delay characterization method and implementation details are stated in Chapter 4 and 5. Chapter 6 shows the results and the analysis. Finally, the conclusion is given in Chapter 7 with a summary and possible future tasks.

Chapter 2

Background

Contents

In this chapter, the concept of reconfigurable system and partial runtime reconfiguration is introduced. Then the general overview of SRAM-based FPGA architecture is given. Finally the implemented platform in this thesis is presented.

2.1 Reconfigurable Architecture

Reconfigurable architectures are devices with programmable logic blocks and programmable interconnects among them. Reconfigurable architectures compute with logic blocks instead of fixed instruction sets, which avoids multiple stages of fetch, decode, and finally execute of instructions.

Reconfigurable architecture can be seen as a choice to balance the performance, flexibility and power. It has a shorter time-to-market than an application specific architecture. The reconfigurable device is prefabricated and tested. The designer can implement the design in the reconfigurable device by using hardware description language (HDL) and CAD tools provided by the vendor in a relative short time, because when it compares to the application specific architecture, the design doesn't have to be sent to the foundry to go through the manufacture process, including lithographic, etching, testing, packaging. However, application specific architecture gives high performance due to the optimization for a particular application. For a small quantity of products that do not have particular requirements on area or power consumption, a reconfigurable architecture is a better choice. Rapid prototyping is one of the advantages of a reconfigurable architecture compared to application specific architectures.

In reconfigurable architectures, the function of the structure can be modified for adapting to different applications. A feature called Partial Runtime Reconfiguration allows all or part of the hardware function to be changed partially or completely during compile time or run time. This feature has been study in academia during the past decade and now supported by reconfigurable devices and it is discussed in the following subsection.

2.1.1 Partial Runtime Reconfiguration

Partial runtime reconfiguration or dynamic partial reconfiguration, allows reconfiguration of a part of the system while other parts typically keep on running without any loss of data or interruption. The partial runtime reconfiguration helps in several ways [3]:

1. Reduce the area and power consumption. With partial runtime reconfiguration, the device resources can be better utilized. By dynamically configuring an area of a system, different functionalities are implemented for different operational demands. Even though more switching activity in the circuit will increase the dynamic power consumption, the static power consumption will reduce because of the reduced logic resource. As the static power dominates the total power consumption in modern CMOS process technology, the total power consumption will reduce (the power consumption problem will be discussed later in section 3.1.1). An example is the partial runtime reconfiguration in software-defined radio application. The applicability has been evaluated and shows that it has benefits using partial runtime reconfiguration [16, 17].

2. Improve the performance. For example, an array of partial runtime reconfiguration processing and memory cell is proposed. The performance of 18 times higher throughput than a traditional DSP solution shows the partial runtime reconfiguration is feasible for device acceleration [18].

3. Fast start up. For some modern automotive applications there are strict timing requirements for the start up time. Those applications require the device to answer the system request within 100 ms after the system is booted [19]. Meanwhile, coming along with the process technology progress, the number of the logic resource in the reconfigurable system growth, so that the configuration data for the complex design is becoming lager and it takes more time to configure the device. Consequently, even using fast configuration method, only the smaller size device can meet the timing specification [3]. Hence the authors purposed the two steps method to achieve the fast start up requirement. In the first step the critical part are configured in the system when it is booted using the priority configuration data. The remaining part of the system which is not timing critical will be configured in the second partial reconfiguration step. The experiment

results show that the fast start up approach can be up to four times faster than the traditional configuration method.

In some dynamically reconfigurable systems, a reconfigurable region is spared for the partial runtime reconfiguration and the region is divided into several slots for practical reason [4]. These slots are called containers, each of which can implement specific hardware for a particular application. When the containers are not needed, they are shut-down to reduce the power consumption. When one or some of the containers is needed for some applications as accelerator, delay characterization of this work can be used to select the one with better performance/without aging effect.

2.2 FPGA Overview

The Field-programmable gate arrays (FPGAs) is a type of reconfigurable architectures. The programmable logic components provide FPGA the reconfigurability. Unlike the Application Specific Integrated Circuits (ASICs), where the device is customized for particular application, FPGAs can be configured to any desired application or functionality when there is enough logic resource. Currently, most commercial FPGAs are SRAM-based because of its re-programmability and the use of standard CMOS processing technology.

2.2.1 FPGA Architecture

FPGA commonly consists of the Configurable Logic Blocks (CLBs) which implement the logic functions; the Programmable Switch Matrices (PSMs) and interconnect wires that connect these logic functions; the I/O blocks (IOBs) connect the logic blocks to the external connections. Additionally, modern FPGA is also integrated with some application specific blocks including memory blocks, Digital Signal Processor (DSP) block and the clock management blocks. A general circuit schematic of an FPGA is shown in Figure 2.1.

It shows one traditional FPGA routing architecture, island style architecture. Each of the blue box in the two dimensional grids is representing one CLB. The interconnect wires are around the CLBs. They are connected to the CLBs via the PSM in grey and the IOBs at the periphery.

2.2.2 Configurable Logic Block

A Configurable Logic Block (CLB) is the main logic component of an FPGA that makes the FPGA a reprogrammable device. It comprises of the Lookup tables (LUTs), interconnection multiplexers and storage element like flip-flops. Each of these elements is introduced in the following subsection.

A LUT, a storage element and multiplexers form a basic logic block. The CLB consists of one or a cluster of intra-connected basic logic blocks as it is shown in Figure 2.2.

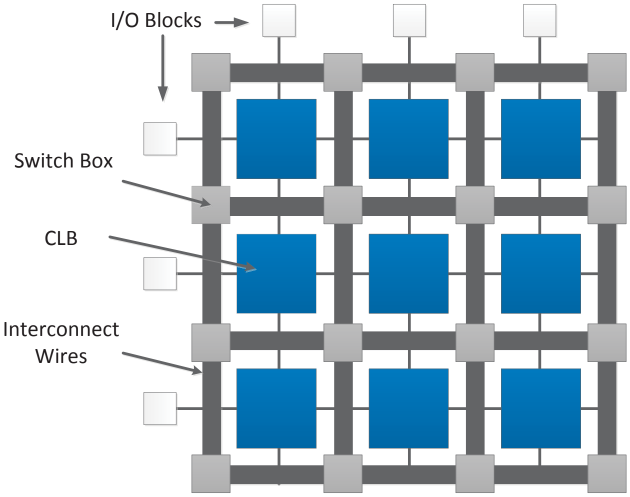


Figure 2.1: Overview of FPGA Architecture

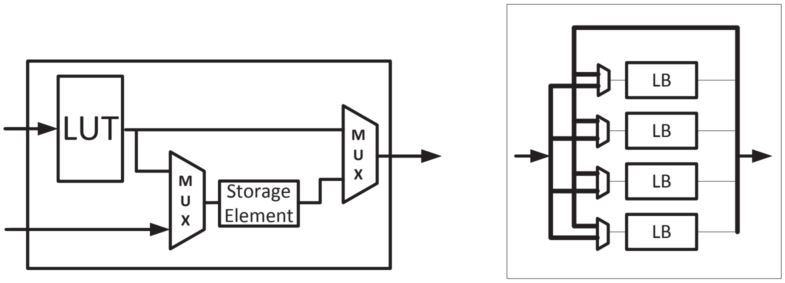


Figure 2.2: Simplified Configurable Logic Block

2.2.2.1 Lookup Table

The Lookup Table (LUT) is the basic logic element that calculates a function in a FPGA. A n-input LUT can calculate an arbitrary n-input Boolean function. It can be implemented as a tree of multiplexers that select an entry of the functions truth table which is stored in a memory element with the length of 2^n bits. In the SRAM-based FPGA, these entries are represented by configuration bits stored in SRAM cells. For example, Figure 2.3 shows a general 3-input LUT. It uses 8 SRAM bits to set the truth table value for any 3-input Boolean function. The input pins control the multiplexers which can be implemented with pass transistors or transmission gates to pass the value to the output [20].

2.2.2.2 Configurable Logic Block internal interconnect

The interconnect multiplexers are used to select the signal connections inside the CLBs for different signal forwarding strategies. The multiplexer selection is controlled by dedicated

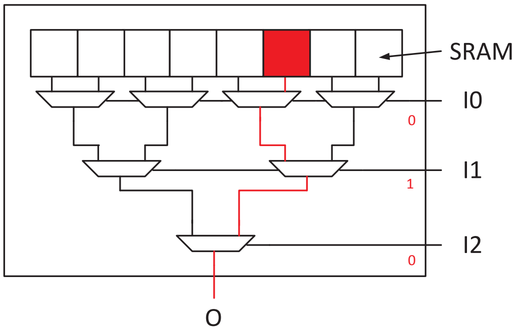


Figure 2.3: A 3-input LUT

configuration bits and does not depend on the input signals. The signal connection is configured after the configuration bits are downloaded. By configuring the multiplexer, the output signal of the CLB can be connected to either the output of the storage elements, or directly to the LUT output, see Figure 2.2.

2.2.2.3 Storage Element

To allow implementation of sequential circuits in the FPGAs, it is essential for the CLB to contain storage elements. The storage elements store the output values of corresponding LUTs every clock cycle. The generic storage element can be configured as an edge sensitive D-type flip-flop or a level sensitive D-type latch.

2.2.3 Interconnect

Figure 2.1 shows that the CLBs are surrounded by the programmable interconnect consisting of switch boxes and interconnect wires. The programmable interconnect routes the signal between the CLBs, to and from the I/O blocks. Routing is conducted by the switch box by connecting or disconnecting the fixed interconnect wires in vertical or horizontal direction. And the input and output pins of the CLB can be connected to the interconnect wire or the connection box depending on different routing architectures [21].

A possible implementation of the switch box is shown in Figure 2.4. There are six pass transistors per switch box interconnect point. The pass transistors act as the programmable switches to control the interconnect wires connection. The pass transistors are controlled by configuration bits.

The interconnect wires can have different lengths, such as direct, double and hex line. The direct line can connect the CLB directly to the neighbour CLB. The double line can connect the CLB to the first and the second neighbour CLBs. And the hex line can connect the CLB to the third and the sixth neighbour CLBs. The longer the interconnect wires are, the fewer programmable switches are needed. As a result, the routing area and delay are reduced [22].

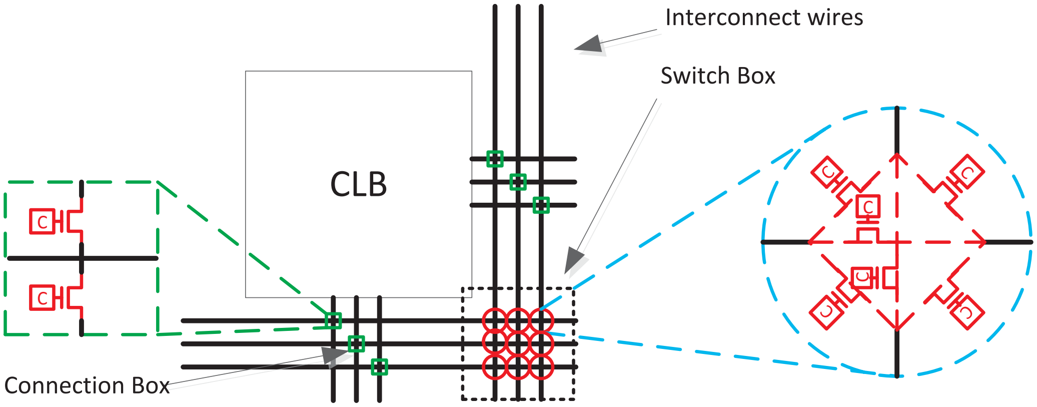


Figure 2.4: Switch box and interconnect wires

However, the long interconnect wires may cause routing flexibility problem, the probability of successfully routing the circuit may decrease. By combining different lengths of interconnect wires in the FPGA, it reaches a better balance among the routing flexibility, area and delay.

2.3 Implementation Platform

Modern FPGAs from different vendors, including Xilinx and Altera, support the partial reconfiguration feature. They also provide the development software to support this function. Xilinx Virtex-5 is one of these FPGAs and it is the implementation platform of this work.

Besides the common components mentioned in Section 2.2, Xilinx Virtex-5 also contains many advanced components including [1]:

- The Embedded Block RAM memory that is for on chip memory in the design;
- The Digital Clock Management (DCM) and Phase-locked Loop (PLL) that provide precise clock signals manipulation;
- The Digital Signal Processor (DSP) slices for digital signal processing like processing multiplication;
- The System Monitor that measures the on-chip physical operating parameters of the FPGA.

This work focus its attention on the CLB logic elements and CLB internal interconnections, and uses carry chain logic. This section introduces the Xilinx Virtex-5 architecture.

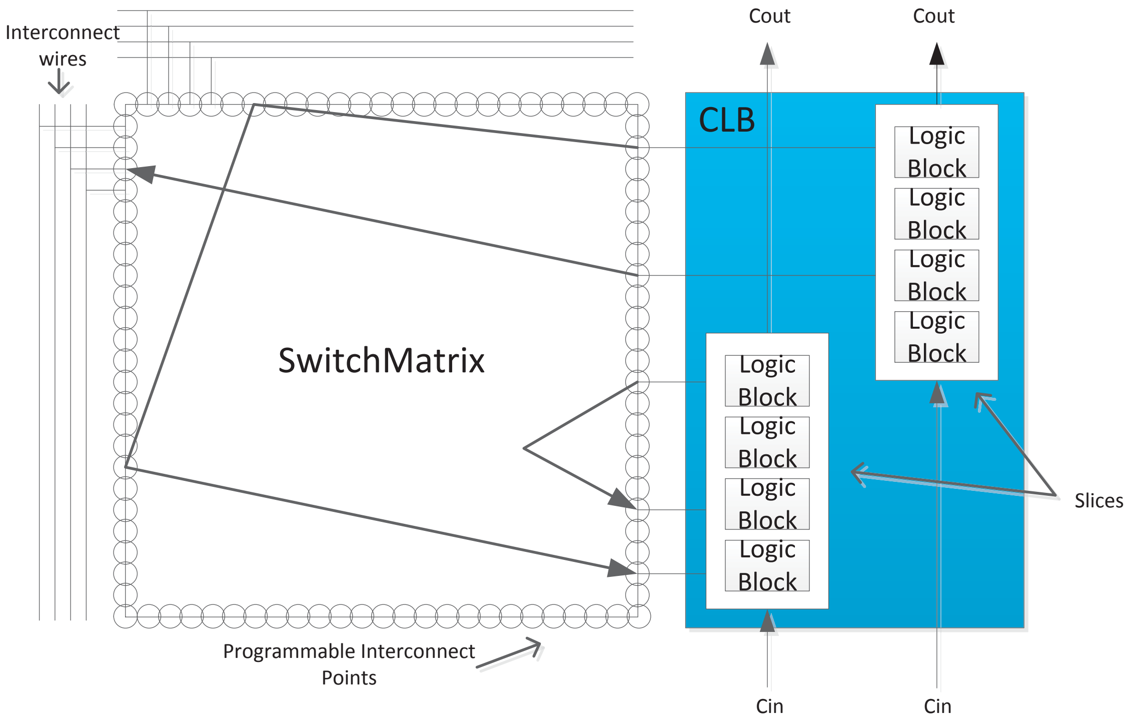


Figure 2.5: Switch matrix and slices within CLB

2.3.1 CLB

In Virtex-5, each CLB is connected to a Programmable Switch Matrix (PSM) to access the global interconnect as shown in Figure 2.5. There are two types of CLB in Virtex-5, CLBLM and CLBLL. Both of them contain two slices. CLBLL has two SLICELs while CLBLM has one SLICEL and one SLICEM. These two types of CLB occur every other column. The two slices in each CLB have no direct connection between each other so the PSM need to be used. Each of the slice belongs to an independent carry chain. For each CLB, Xilinx labels the slice on the left SLICE(0) and the one on the right SLICE(1). So the carry out of the SLICE(0) can be passed to the carry in of the SLICE(0) in the upper row. As the same, SLICE(1) can propagate the carry to SLICE(1).

A SLICEL has four logic blocks which mentioned in Section 2.2.2. Each of the logic block is comprised of a 6-input LUT, multiplexers for path selection, and a storage element that can be configured as a flip-flop or a latch. Additionally, the slice has a carry chain which can perform fast lookahead addition and subtraction, see Figure 2.6.

The SLICEM adds more functionality to the SLICEL. The LUT inside SLICEM can be configured not only as a LUT, but so be configured as random-access memory (RAM) or shift register, see Figure 2.6.

This work considers both SLICEL and SLICEM, but only focus on logic elements and internal interconnections. As a result, we ignore the memory structure of SLICEM.

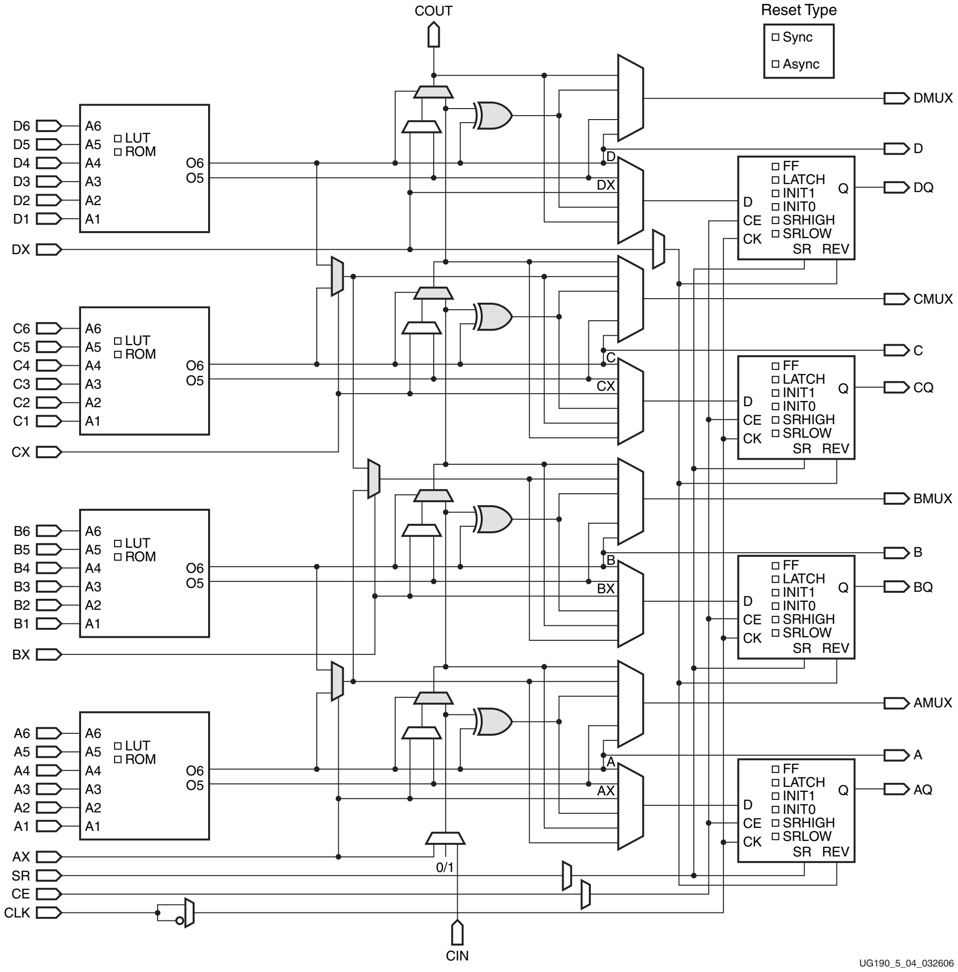


Figure 2.6: Diagram of SliceL [1]

2.3.1.1 Lookup Table

The Lookup Table (LUT) in Virtex-5 has six independent inputs and two outputs. The LUT can be configured as one 6-input LUT or two 5-input LUTs as long as the two implemented Boolean functions share the same inputs. By configuring the multiplexers, up to four LUTs can be combined to implement an arbitrary seven or eight input Boolean function in one slice.

The output signals of the LUT can directly exit the slice, enter a XOR gate or multiplexer of the carry chain, control the multiplexer of the carry chain, or go into the D input of the storage elements as shown in Figure 2.6 and Figure 2.7.

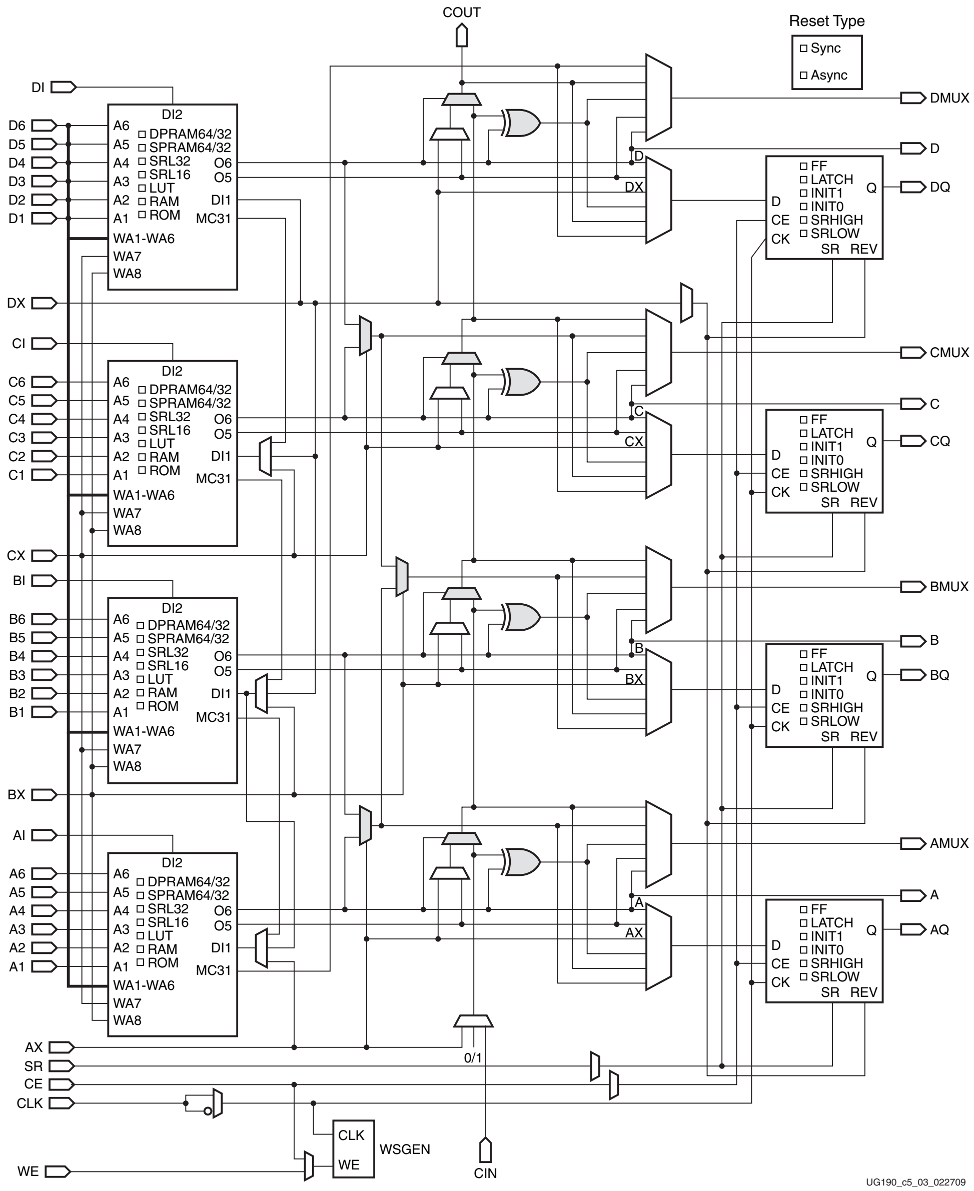


Figure 2.7: Diagram of SliceM [1]

2.3.1.2 Carry Logic

The carry logic allows an area-efficient implementation of addition and subtraction in a slice. Each slice can compute four result bits plus a carry bit. As mentioned earlier in this section, each CLB has two separated carry chains that can pass on the carry bit from the bottom CLB row to the upper CLB row for the add and subtraction operation which is more than four bits. An example implementation with the carry logic is Ripple-carry Adder.

The carry chain is constructed with carry multiplexers and dedicated XOR gates for calculating the operand. The XOR gate at the bottom of SLICE(0) in the carry chain is used in the ring oscillator design of this work.

2.3.2 Programmable Switch Matrix and Interconnect

Xilinx FPGAs use an island-style routing architecture, CLBs are on the “islands” surrounded by the programmable routing network. The switch box is called Programmable Switch Matrix (PSM) and it is adjacent to the CLB. The PSM is formed of wires and the programmable interconnections called the programmable interconnect points (PIPs) represented by the small circles in Figure 2.5. A PIP is implemented as a multiplexer which is controlled by the configuration bits for selecting the wire segment to connect the next PIP or interconnect wires.

The interconnect wires in Virtex-5 have different lengths. Even though the detail information is not documented in the official user guide, the interconnect details can be seen from the Xilinx FPGA Editor. The Virtex-5 wire types can be found in the tool are: Bounceacross, Double, Pent, Long and Global line. Bounceacross and Double are Direct and Double lines mentioned in Section 2.2.3. While the Pent line connects the CLB to the second and the fifth neighbour CLBs, the Long line connects the CLB to sixth, thirteenth, and the twentieth neighbour CLBs. Finally, the Global connects the CLB from the first to the twentieth neighbour CLBs.

Chapter 3

Delay Measurement in FPGAs

Contents

Delay measurement methods have been widely used on both ASIC and FPGA platforms. The main delay measurement methods can be grouped in at-speed delay test method and ring oscillator based method. This chapter first introduces the temperature influence on the circuit delay briefly, then discusses the fault models which are related to the delay measurement, and finally reviews and compares the state of the art in the literature.

3.1 Temperature Influence on Delay

The temperature affects the speed performance, power, and reliability of the system. With higher temperature, the integrated circuits become slower due to reduced carrier mobility and higher interconnect resistivity. In this section, the overall power consumption theory is introduced.

3.1.1 Overall Power Consumption in CMOS Logic Circuit

The power consumption of CMOS logic circuit consists of two parts: dynamic power and static power. Dynamic power consumption has two parts. The first part is the power dissipation that is caused by the charging and discharging activities of the CMOS capacitance load: the power consumption due to the switching current. The second part is the power dissipation that is caused by the short circuit generated during switching: the power consumption due to the short circuit current. The static power consumption is always dissipated even when the CMOS does not perform any activity.

Dynamic power consumption:

$$P_{dynamic} = P_{switching} + P_{sc} \tag{3.1.1}$$

where $P_{switching}$ and P_{sc} are the power consumptions due to the switching current and short circuit current.

Power consumption due to the switching current:

$$P_{switching} = AC_L V_{dd}^2 f \tag{3.1.2}$$

where A is the factor of the gates switching activities, C_L is the total effective load capacitance of all the gates, V_{dd} is the supply voltage, and f is the clock frequency.

Power consumption due to the short circuit current:

$$P_{sc} = I_{sc} V_{dd} \tag{3.1.3}$$

where I_{sc} is the short circuit current generates during signal transaction when both the NMOS and PMOS are active and directly conducts the V_{dd} to the ground.

Static power consumption:

$$P_{static} = I_{leakage} V_{dd} \tag{3.1.4}$$

where $I_{leakage}$ is the leakage current including reverse biased p-n junction current, subthreshold leakage and some other components which would not be discussed here.

The above equations define the overall power consumption:

$$P = P_{dynamic} + P_{static} = P_{switching} + P_{sc} + P_{static} = AC_L V_{dd}^2 f + I_{sc} V_{dd} + I_{leakage} V_{dd} \tag{3.1.5}$$

The activities in the FPGA will not influence the static power dissipation as it always occurs. However, when the switching activities increase, the dynamic power consumption rises as both of the switching current and short circuit current will increase.

3.2 Delay Fault Model

Delay faults affect the propagation delay of the circuit and cause the delay to exceed the clock period. If the circuit is operating at high frequency, it is more likely to fail due to the delay faults. Two specific delay faults are introduced in detail this section: transition faults and path-delay faults [23].

3.2.1 Transition Fault

In a fault-free circuit, all the gates have some delays. Assuming all the gates in the fault-free circuit have delays, when one of the gates becomes faulty and the delay increase is large enough, then transitions on all the paths through this gate can not reach any observable output within the clock period, even for the shortest path. For each gate, the transition is either from 0 to 1 or from 1 to 0, correspondingly there are two types of transition faults for each gate, slow-to-rise (StR) and slow-to-fall (StF).

Here is a simple example of the transition fault with a NAND gate (Figure 3.1)

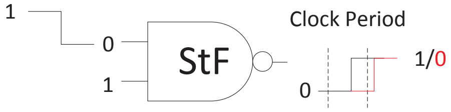


Figure 3.1: Slow-to-fall transition fault

Assume that the initial value of the upper signal is 1, when a 1 to 0 transition happens, the output should rise from 0 to 1 within the clock period if this gate is not faulty. However, if the gate is affected by a transition fault, the effect will be observed as a 0 at the primary output instead of the expected value 1.

3.2.2 Path-Delay Fault

The path-delay fault causes the sum of the propagation delays on a combinational path to exceed a specified clock interval as depicted in Figure 3.2. The combinational path consists of a chain of connected gates and interconnections from a primary input to a primary output. Propagation delay is the time that a signal transition takes to pass though the path and it consists of switching delays of devices and transport delays of interconnects on the path [23].

The number of the paths in the circuit may be exponential in number of the gates. It is not practical to test all of the path-delay faults in the circuit. Primarily, a set of longest paths are selected for testing as the sum of delay on shorter paths may not be large enough to prevent a passing transition from reaching an output within the clock interval.

For each combinational path, there are two path-delay faults considering the rising and falling transitions. Therefore, the total number of path-delay faults is twice the number of paths in the circuit

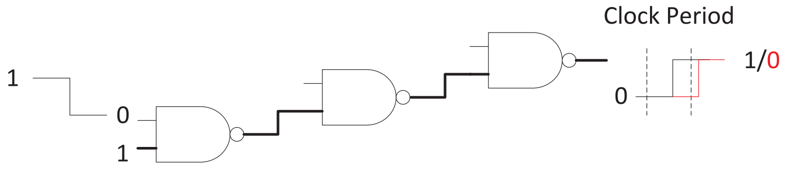


Figure 3.2: Sum of propagation delays exceeds clock period

3.3 Delay Characterization using Delay Test

Most of the existing at-speed delay fault tests are used to make the pass/fail decision of a specific integrated circuit design at a specific frequency, i.e. to find out if a circuit operates without timing failure at a target frequency. However, they can also be adopted to obtain the precise propagation delay information of a combinational path in the design [24]. This section will firstly introduce the basic idea of delay test, and then how this test methodology can be employed to get the propagation delay information of the combinational path.

3.3.1 Delay Test

Depending on the type of Circuit Under Test (CUT), different delay test methodologies can be used. Here in this work, only a simple delay test methodology is introduced, which is later used for the delay characterization.

The test architecture in Figure 3.3 is suitable for combinational circuits or for sequential circuits with registers only at primary inputs and primary outputs [24, 25].

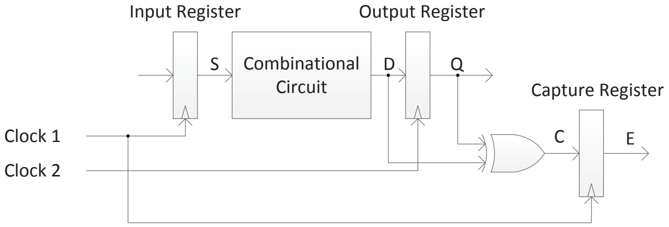


Figure 3.3: Test architecture for delay fault

The input register and the error capture register are clocked by the test clock 1 and the output register is clocked by the test clock 2 which is of the same frequency but skewed by a predefined time period. The input register stores the stimuli S from the test stimuli generator and pass it to the CUT. Then the circuit will have the transient signals and the output signal D toggles after the propagation delay in the CUT. This output signal D will be compared with the value sampled by the output register with an XOR gate and the compared result C will be captured by a register at the rising edge of the test clock 1 to generate the error detection signal E. Figure 3.4 illustrates the test operation.

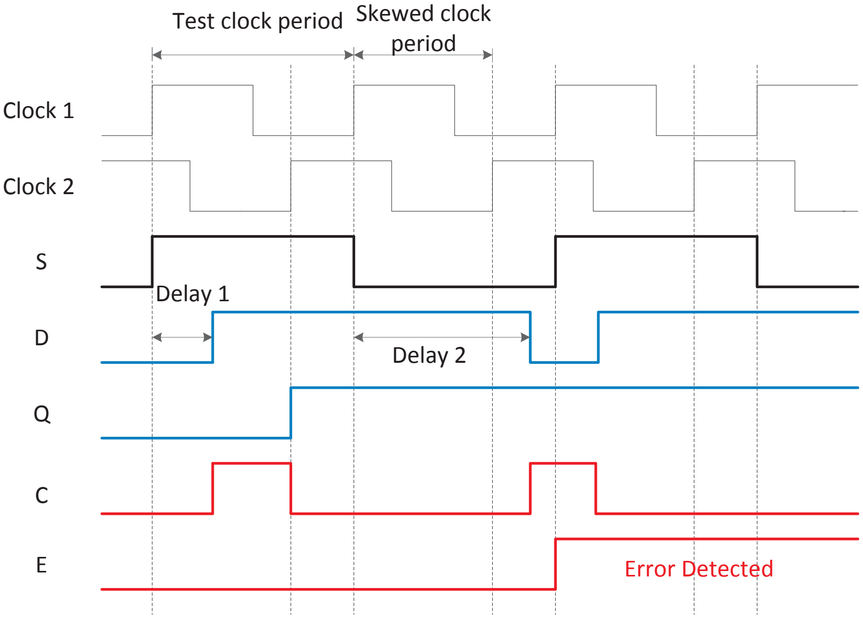


Figure 3.4: Timing diagram of the test circuit

Delay 1 is the propagation delay for the 0 to 1 transition in the CUT which is within the predefined time period, so there is no delay fault. In contrast, Delay 2 is a propagation delay larger than the predefined time period so that it is observed as a slow-to-fall fault. The predefined time period can be changed by the clock source, therefore the delay test can be set to specific frequencies according to the design requirement. When the phased-shift between test clock 1 and test clock 2 is smaller, the predefined time period is shorter and, as a consequence, the delay test becomes more strict. Hence, controlling the phase offset or changing the clock frequency allows to determine the propagation delay in the design. Examples of delay measurement methods using delay test are introduced in the following subsection.

3.3.2 Delay Test for Delay Characterization

The at-speed delay test is adopted for delay characterization in FPGAs [24, 25]. They all use a same idea of changing the time between the clock edges of input and output registers to measure the propagation delay using the on-chip clock generation resources like Digital Clock Manager (DCM) on the Xilinx FPGA.

A double sampling delay characterization method is introduced in [25]. Different from the test architecture in Figure 3.3, another register which is clocked by the test clock 1 is added between the CUT and the XOR gate so that its output value is compared with the output register which is clocked by test clock 2. The test clock 1 remains constant so that the CUT is not interrupted by the delay characterization testing. The phase shift of test clock 2 is increased with a small step to reduce the predefined time period after a certain number of cycles. For each paths in the CUT, the predefined time period is recorded when the error is first time captured. The test process is repeated until all the paths in the CUT are characterized. This method is suitable for charactering combinational path delay in a specific design.

In [24], the test clock 2 is inverted from the test clock 1, so the predefined time period is always half of the test clock period. As the test clock steps from the lower frequency to a higher frequency, the CUT is observed from working correctly to working with delay error. A counter is used to count the total number of the timing errors during a predefined test cycle for each test clock frequency and then calculate the so called failure rate to estimate the delay of the CUT. It claims that all the LUTs on the Altera FPGA can be precisely characterized at a high resolution.

3.4 Delay Characterization using Ring Oscillator

Ring oscillators are widely used for thermal sensing, measuring the effects of manufacturing process variation and delay characterization in ASICs and FPGAs. In this section, the basic concept of a ring oscillator, different ring oscillators in circuit design and the ring oscillator-based delay characterization method will be presented.

3.4.1 Ring Oscillator

A Ring Oscillator (RO) consists of an odd number of inverting elements. These inverting elements are chained in a loop and the output toggles between the low voltage level and the high voltage level, as depicted in Figure 3.5.

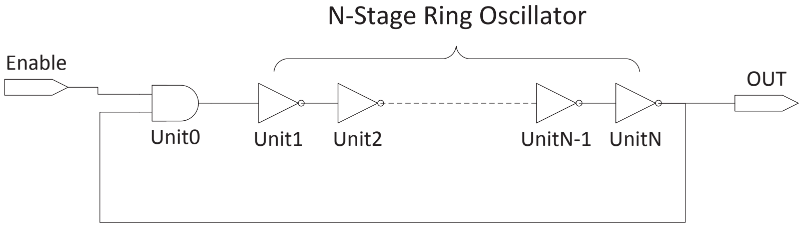


Figure 3.5: N-stage ring oscillator with enable signal

The period T of the ring oscillator equals to twice of the propagation delay of the loop. $T = 2 \times (Delay\ of\ the\ inverters + Delay\ of\ the\ interconnect)$. Hence the delay on the loop can be translated to the frequency $f = 1/T = 1/(2 \times (Delay\ of\ the\ inverters + Delay\ of\ the\ interconnect))$.

3.4.2 Temperature Dependency of Ring Oscillator

As mentioned in Section 3.1, the switching activity in the FPGA will consume power and the resulting heat dissipation will heat up the FPGA. In contrast, the increased temperature will slow down the switching speed in the FPGA. In this section the methods that study and make use of the relation between ring oscillator frequency and temperature are presented. They are classified into two groups.

- Ring oscillator as a heater: study how the ring oscillator frequency influences the temperature in the FPGA.
- Ring oscillator as a thermal sensor: study how the temperature influences the oscillation frequency which is then used to measure the temperature in the circuit.

3.4.2.1 Ring Oscillator as a Heater

From Equation 3.1.1, assume that in FPGA the total effective load capacitance of all the gates and the supply voltage are constant: the dynamic power consumption will increase when the gate switching activity and the clock frequency increase. In the FPGAs, temperature can be increased significantly by maximizing the dynamic power consumption using different resources [26]. For example, a large number of 1-stage ring oscillators can heat the FPGA to 134 ℃ on Virtex-5 LX110t FPGA which is also the used device in this work. In [26], every 1-stage ring oscillator is implemented with a single LUT with its output directly fed back to its input.

3.4.2.2 Ring Oscillator as a Temperature Sensor

In an FPGA, thermal sensors can be implemented as a lookup table (LUT) based ring oscillator style design. The LUTs are configured as separate signal inversion elements similar to individual inverters in a normal ring oscillator with hardware description language (HDL) or other design method [27]. The frequency of ring oscillator is related to the total delay of the logic elements and interconnects in the loop so that the way of designing the ring oscillator, using different FPGA resources, affects the sensitivity of frequency change due to temperature change.

It is demonstrated that the ring oscillator design that includes latches is more sensitive to temperature than the one without latches. That is possibly because the temperature effect is more significant on transistor delay than on wire delay [28]. In [29] the authors also claim for the similar result that the ring oscillator comprising 23 inverters and 24 latches gives the best overall sensitivity to temperature.

These works all showed that the correlation between the temperature and the ring oscillator frequency is linear. A ring oscillator combined with timer counter and capture counter is used as temperature sensor.

The sensor designed in [29] was synthesized from a VHDL specification using vendor tools to route the design automatically, hence resulted in differently routed sensors. As a result, the authors placed more sensors on the FPGA and took the average value of the results to decrease the impacts of routing variations. This implementation method is not suitable for using ring oscillators to characterize delay on FPGA, as the frequency differences are not caused by the speed difference of the logic elements, but the routings.

3.4.3 Ring Oscillators-Based Delay Characterization in ASIC

Ring Oscillators have been used for delay measurement and process variation measurements in ASICs. ASICs are normally not tuned after manufacture, the aforementioned methods are used to monitor the process variation, fabrication parameter and to estimate the maximum speed of system [30, 31, 32, 33, 34]. The on-chip test structures in [30] provide path delay prediction for chip performance evaluation. And the ring oscillators are placed in various locations across the chip to provide information on process variations within-die [31, 32]. Moreover, the ring oscillators are used to identify and localize power-related failures [33]. In [34], the measurement method can characterize the gate delay for early process characterization in manufacturing. Hence, the use of ring oscillators is a standard and well known technique for delay measurement for ASICs.

3.4.4 Ring Oscillators-Based Delay Characterization in FPGA

On FPGA platforms, the main approaches for ring oscillators-based delay characterization are:

1. Delay variation comparison that compares the spatial delay variation [24, 25].
2. Differential delay measurement that directly measures the delay of the path under test [35, 36].

This subsection will introduce the two types of method in detail.

3.4.4.1 Delay Variation Comparison

The idea of delay variation comparison is a delay characterization method that compares the frequency of the ring oscillators at different locations on the FPGA. Due to the process variation and the impact of aging mechanism, the delay of the logic elements and interconnect wires at different locations on the FPGA are not identical. When the ring oscillators are implemented with these logic elements and interconnect wires, the delay variation between these resources result in the variation of ring oscillator frequencies. For example, an array of ring oscillators which covers an area of an FPGA is used for the purpose of within-die delay characterization [37, 38].

How the measurement is performed on FPGAs is described in [38]. The area under test implements a two dimensional array of identical ring oscillators, which are activated one at a time and measured by the counter under the control by a timer. However, the ring oscillators are implemented with a part of the LUTs. Instead of measuring all the LUTs in the area under test to obtain the delay variation of the whole area, the authors use an equation to model and estimate the systematic and stochastic delay variation. And the accuracy of the equation is calibrated by measuring different stages of ring oscillators at each location. In this way, the characterization may not detect the degradation in a certain part of the FPGA due to the low coverage of LUTs in the area under test.

Moreover, the ring oscillator sensor used in [39] can measure variations in delay, leakage, power and temperature. The scattered sensors can also be arranged as an array. Compact shift register counters are introduced and are attached to each ring oscillator. The consuming time of the test is shorten because all the ring oscillators are measured simultaneously. However, as mentioned in Section 2.3, only the LUTs in CLBLM can be configured as shift register. As a result, the ring oscillator design with a compact counter is not possible to be implemented in CLBLL on Virtex-5 FPGA platform.

3.4.4.2 Differential Delay Measurement

The basic idea of differential delay measurement is that if a logic element or an interconnect wire is added to the ring oscillator loop, then the delay on the loop will increase and the frequency will decrease. By comparing the frequency changes, the delay of the added logic element or interconnect wire can be calculated.

In [35] a method which compares two Ring Oscillators to measure the propagation delay on the interconnect wire is proposed. A reference RO is compared with the RO which with similar structure but includes the path under test in the loop. The structure under test (SUT) can be any arbitrary non-inverting path, including many different interconnect wire segments and logic elements. The propagation delay of the SUT shown in Figure 3.6 is measured from the frequency difference between the ring oscillator shown in figure and the same ring oscillator with inverter Unit 2 directly driven by inverter Unit 1. However, this measurement method has a problem, that this propagation delay also includes the delay difference between net n1 in Figure 3.6 and a net directly connecting inverters Unit 1 and 2.

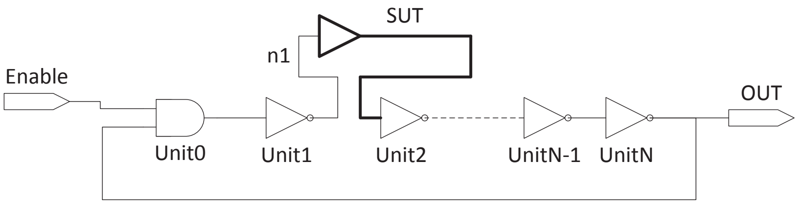


Figure 3.6: Structure under test is added to ring oscillator

The same structure is implemented in different locations to measure the delay variation and also different wire types and length are measured and the results are compared with the estimation from the vendor tool in [35]. The results are directly measured with oscilloscope.

Similar idea is used in [36] to characterize the delay of a single logic element. Their design is based on Xilinx Spartan-3e FPGA. The CLB structure is different from Virtex-5 but it contains eight LUTs as well. One of the LUTs is implemented as a NAND gate, the others are implemented as buffer with identity function as shown in Figure 3.7. The delay of a single 4-input LUT is measured by comparing the frequency of the 8-stage ring oscillator and the 7-stage ring oscillator omitting one of the LUT. When it compares to the previous direct measurement method in [35], this method also has the problem of delay changes in the interconnect wires. The delay estimation information is extracted from the Xilinx timing analysis tool and is used to calculate the delay of a single LUT. The estimation result is pessimistic and it is 80% larger than the actual measurement result. As the interconnect wire composition and the connection order of the LUTs are chosen based on the minimum estimation delay, the input pins of the LUTs of the 8-stage ring oscillator and the 7-stage ring oscillator would be different because of the limited direct connection between the LUTs. As a result, this design is measuring the LUT delay under the assumption that all the inputs of the LUTs have the same speed. This may not be the case when some of them degraded more quickly due to the different aging and the switching rate. Nine test configurations are needed. The CLB based ring oscillators cover all the LUTs in the area under test.

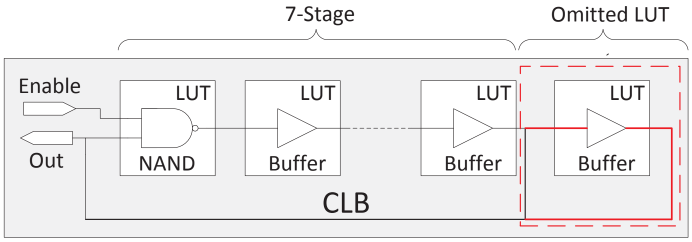


Figure 3.7: Characterizing the delay of omitted LUT

3.5 Delay Measurement Method Comparison

Both of the delay characterization methods using delay test and using ring oscillators can be adopted in the runtime reconfigurable system to test the container. The delay test method in [24] can accurately measure the path delay, but the resolution of the measurement depends on the size of frequency sweeping step and a Xilinx FPGA is used to provide the clock generation to the Altera FPGA. Even though the newer FPGAs can provide on-chip test clock generation, it still increases the design complexity of the system. In contrast, the method using ring oscillator does not require complex clock control. Once the ring oscillator is activated, it automatically oscillates at the maximum frequency depending on the speed of the FPGA fabric. Moreover, the control and analysis of the circuit is simpler than the method using delay test. Considering the complexity of extending the method to a runtime reconfigurable system application, the delay characterization method using ring oscillators is developed and implemented in this work.

Although the ring oscillator method is well-known and has been studied for delay characterization, the existing method is not perfect for detecting the degradation due to the aging mechanism in the reconfigurable system. For example, only one fix routing of ring oscillator is implemented in the design and the input pins of the LUTs are randomly chosen in [36], not to mention that the method in [24] doesn't cover all the LUTs in the area under test. The characterization method in this work complements these approaches and is able to cover all the input pins of the LUTs in the container.

Chapter 4

Architecture for Delay Characterization

Contents

4.1 Overview

A ring oscillator array methodology is used for delay characterization in this work. A new ring oscillator design that can cover all the path in the LUTs of a single CLB is proposed. In this chapter, the idea and the advantages of this CLB based ring oscillator design are given. Then the auxiliary circuit design of counter and control circuit are introduced. The system monitor of Virtex-5 is also presented.

A ring oscillator array that fits in a container is shown in Figure 4.1. Each small block is representing a CLB. Ideally, if we can place the ring oscillator design in every CLB within the container, then we can reach the minimum number of test configurations. However, as

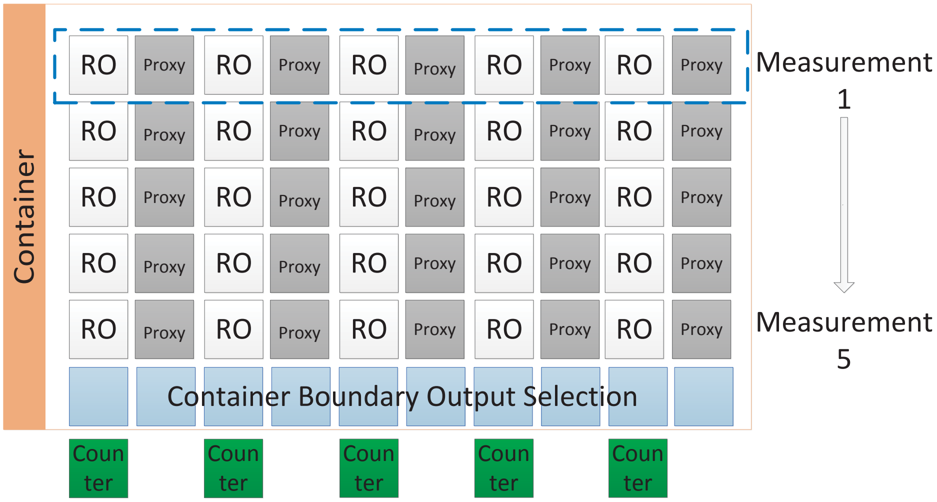


Figure 4.1: Overview of the ring oscillator array design

low level design is not well supported by the CAD tools, a proxy must be attached to each ring oscillator to fix a problem caused by the tool. Therefore, the test configurations have to be doubled to cover all the CLBs in the whole container area. The proxy is explained in Section 4.2.3.

One row of counters measure multiple rows of ring oscillators. For each measurement, one row of ring oscillators are selected and the oscillations in a time period are measured by the counters in parallel.

4.2 Ring Oscillator Design

This work is focusing on the delay variation on the chip of one FPGA device by comparing the speed of ring oscillators in different locations. The reason of using a special design method here is because the ring oscillator in this work is a low level design. Any factor that can contribute to the differences in the speed of the ring oscillator must be avoided as they will disturb the measurement of the speed difference caused by aging. Besides, specific pins are used in the ring oscillator. Using VHDL or Verilog Hardware Description Language (HDL) for the design can not satisfy the low level design requirement.

VHDL and Verilog HDL and the synthesis tool may be used to generate the ring oscillator design. It is possible to place this kind of ring oscillators on different FPGA devices and compare the frequency of the ring oscillators at the same location to measure the chip-to-chip variation. However, it is not suitable for the delay variation measurement within a single chip for the following reason. Even if the descriptions of ring oscillators are identical, for example using the generic statement in VHDL, the LUTs inside the ring oscillators of different locations may have different placements. Moreover, even if the LUTs are placed exactly the same for all the ring oscillators, the interconnect routing may not be exactly the same [29]. The difference of the interconnect routing may have more delay impact than the aging effect.

Using different LUT inputs may also influence the ring oscillator delay. In [21], it is stated that the inputs of the LUT are logically equivalent. However, it is mentioned in Section 2.2 that the LUTs are implemented as tree of multiplexers, the delays through the multiplexers are different. For example, in Figure 4.2 (a) the multiplexer in red is without aging and it is switching as input value I0 is changing. Its delay is smaller than the one with aging effect in Figure 4.2 (b). As a result, it is essential to characterize the delay of each multiplexers by using different input value combinations and selecting different input pins to switch.

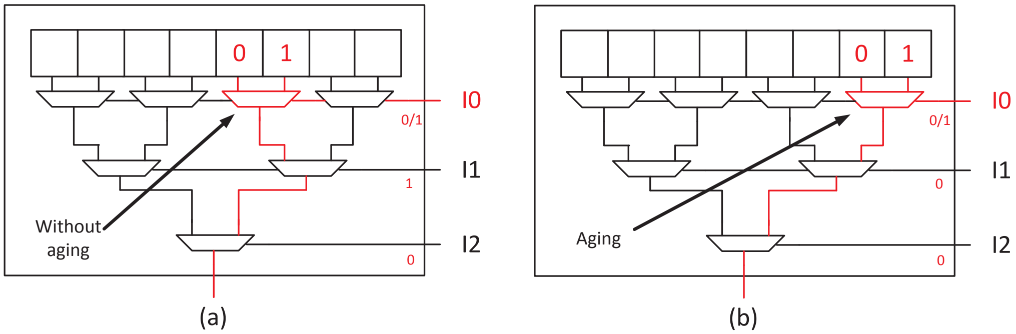


Figure 4.2: LUT inputs select multiplexer with/without aging in a 3-input LUT

The synthesis tool will choose the input of the LUT during place and route stage based on a certain rule. And we could not control each LUT at the same position in the ring oscillators to use a same input by using a high level HDL to generate the design. Hence, not only a new design of ring oscillator is necessary, but also a special design method is needed.

4.2.1 Design to cover all Paths in the Lookup Table

In previous work of delay characterization methods using ring oscillators [35, 37, 38], it is ambiguous how the path on the loop looks like, and which pins of the LUTs are used. In [36] the authors choose the interconnect with minimum estimated delay and we can infer that only one set of LUT pins are used in their work. In [38], the ring oscillators are implemented using some of the LUTs in the logic array block, but not all the LUTs in the area under test are covered.

These methods can have a good spatial evaluation of the process variation assuming the chip is in a healthy condition without fault or aging impact. However, when a certain part of the chip is suffering from the aging effect, a method which only characterizes the delay of one path or some paths within a CLB is not comprehensive enough to give a complete delay characterization of the area under test. For example, if in the worst case one of the LUT inputs of a CLB was almost slow enough to be considered as faulty wasn't on the path of the loop, then the delay characterization result may show that the speed of the CLB was not slow and that it's not seriously affected by aging. As a result, the aforementioned method could not avoid using the CLBs which were actually suffering from aging and a more sophisticated ring oscillator design for delay characterization is needed.

We decide to implement a ring oscillator per CLB for the following reasons:

1. Considering a Virtex-5, CLBs have two slices, four LUTs per slice, in total eight LUTs (see Section 2.3.1). Thus, a CLB has enough logic elements to implement a ring oscillator within a frequency range that can be robustly measured by a built-in counter.

2. One ring oscillator per CLB is a suitable granularity to cover all the paths in the LUTs of all the CLBs in the containers. With the basic unit size, the ring oscillator can fit in different sizes of containers.

The idea of covering all the paths and pass transistors in all the LUTs of a CLB is given:

1. All the input pins of the LUT should be at least once on the ring oscillator loop.

2. All the truth table values should be read out at least once per test configuration.

As a result, we search for a Boolean function that selects all the entries of the functions true table. For Virtex-5, it is a 6-input Boolean function that uses all the inputs of a LUT. Besides, we have to assure that, when enumerating the paths in the LUT, the ring oscillator has an odd number of inversions so that it can oscillate.

Again a 3-input LUT is used as an example, see Figure 4.3:

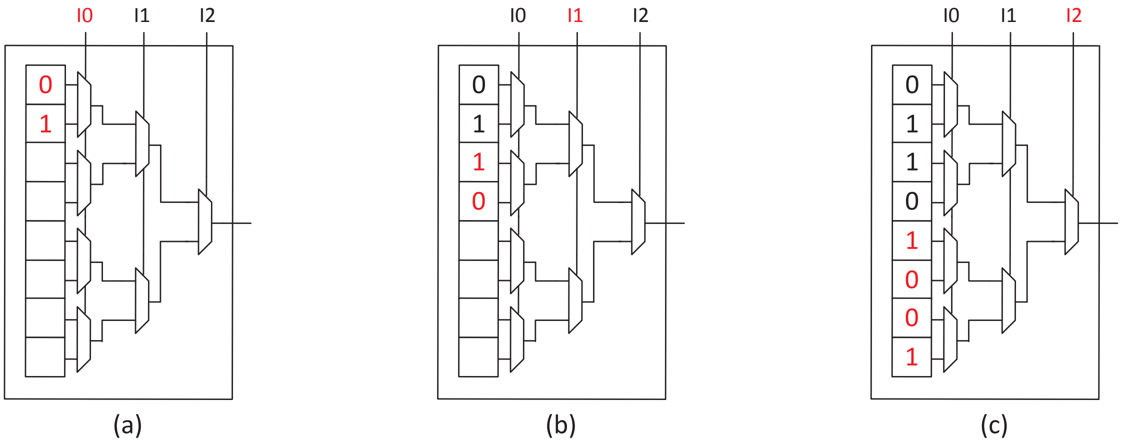


Figure 4.3: Required Boolean function for the ring oscillator design based on a 3-input LUT

When I0 is toggling on the ring oscillator path, the truth table values should invert. So every pair of the SRAM bits controlled by the same multiplexer should be either 01 or 10, as for example in Figure 4.3 (a). When I1 is on the path, it inverts the value of first two bits. For example, if I0 is set to 0, the first and the third bits should have inverted value while I0 is 1, the second and the fourth bits should have inverted values like in Figure 4.3 (b). The situation is the same for I2, so the value of the fifth to the eighth bits are with the inverted values of the first to the fourth bits, see Figure 4.3 (c). If we just consider I0 I1 and write down the input values and SRAM bit value they will fetch, then we see that we are actually implementing the XOR function for I0 and I1. The I1 and I2 are also implementing an XOR. It's a hierarchy of XORs. The Boolean function is O6 = A0 ⊕ A1 ⊕ A2 ⊕ A3 ⊕ A4 ⊕ A5.

The ring oscillator implemented with seven LUTs configured as inverter plus a LUT configured as an AND gate is a practical design choice when it is not necessary to cover all the paths and pass transistors of LUTs within a CLB. However, since we target full coverage here, the test configurations have to be doubled since the pins which are used as enable pins disable half of the paths in the LUT. For example, in Figure 4.4 I2 is chosen as enable pin. When I2 is set to 0, output is always 0 and the ring oscillator won't oscillate. The delay of the three multiplexers on the right can not be characterized. Another configuration which enables the ring oscillator when I2 is set to 0 is needed to cover these three multiplexers.

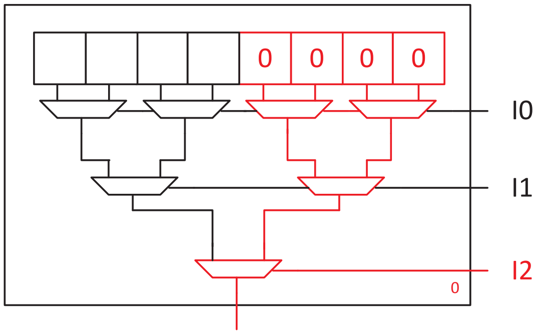


Figure 4.4: Input I2 disable the ring oscillator

If we need to apply this work in a partial runtime reconfigurable system, we have to minimize of the number of test configurations. The less test configurations, the smaller the required storage size of configuration data, and the smaller the required time for delay characterization. A nine stages ring oscillator is purposed in this work and it can reduce the test configurations.

We configured all the eight LUTs in a Virtex-5 CLB as inverters. However, an even number of inverters on the loop will not oscillate, and another inverting element is required. So the XOR gate from the carry chain in CLB can be used as the ninth inverter. One input of the XOR gate is from the ring oscillator loop and another is from the enable signal. The inequality function of the XOR gate determines that once the enable signal is set to 0, the output signal of the XOR gate is identical to the input from the loop; in the other case, the enable signal is set to 1, the output signal of the XOR gate is the inverted value of the input from the loop. In the first case, the design will not oscillate because there is an even number of inverting functions in the loop and in the later case, the design oscillates as a nine stages ring oscillator.

The schematic of the ring oscillator within a CLB is shown in Figure 4.5. The LUTs are first chained within a slice, pass through the first XOR gate of the carry chain and then to another slice.

As the internal connects between the LUTs on ring oscillator loop are fixed after the configuration bits are downloaded, the ring oscillator loop goes through only one set of LUT inputs

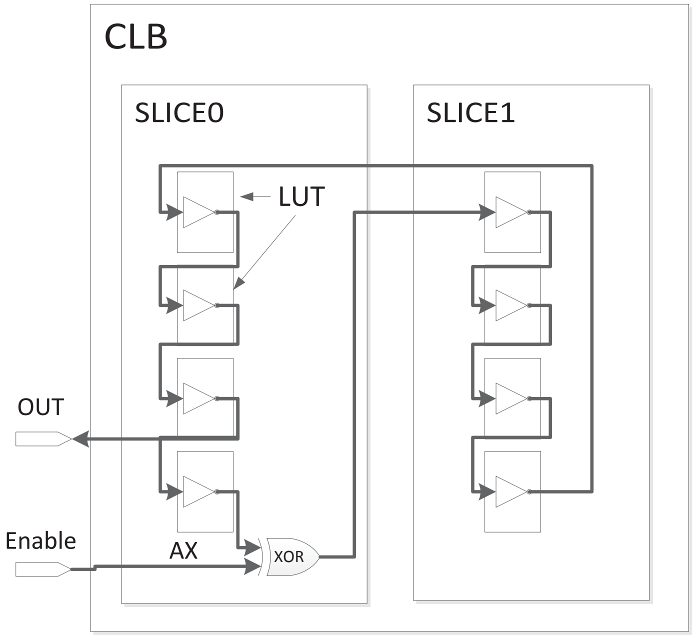


Figure 4.5: One ring oscillator per CLB

and it is fixed before the next reconfiguration. In this design, for each configuration we choose the input of the same pin number for every LUT in the CLB for the delay characterization (Figure 4.6).

For the other five LUT input pins, the input pins with the same pin number are connected to the same input net, for example, all the input pins A6 are connected together. These five input pins nets together are defined as test cases control signal. The test cases control signal has 2^5, i.e. 32 combinations. By changing this 5-bit signal for the six different input pins combinations, all the paths and multiplexers inside the LUTs of a single CLB can be covered.

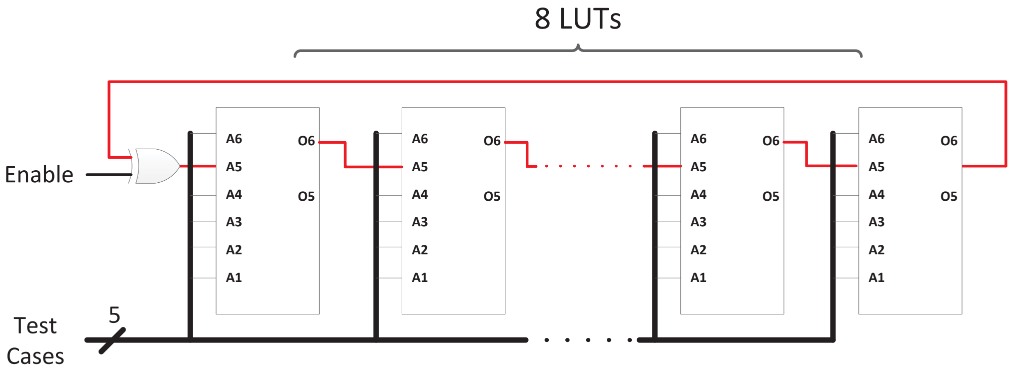


Figure 4.6: The control signals of the CLB ring oscillator

The line in red in Figure 4.6 represents the path on the ring oscillator loop. The test case control signal is five bits wide, each bit connecting to the LUT inputs with the same pin number. In the best case, we will just need six test configurations. However, due to the implementation problem which is explained in Section 4.2.3, additional test configurations are required.

4.2.2 Hard Macro

The ring oscillator is implemented as a reusable physical module called hard macro. A design can contain a hard macro many times. A hard macro is a logical function created from components of a specific device family. It specifies the actual configuration of the contain LUTs and their interconnect. As the ring oscillators in the array are using the same hard macro, it is sure that they have an identical logic function and routing, so the delay of each CLB can be compared.

However, a hard macro can not be instantiated in any location. The hard macro must be placed at a location which has the same component type as the original location. For example, when the ring oscillator hard macro is generated for a CLBLM, it can not be placed in the CLBLL as SLICEL and SLICEM are components with a different type.

4.2.3 Proxy against the Pin swapping problem

We used a low level design specification to generate dedicated interconnection to characterize the delay of a CLB using different inputs of LUTs. The pin to pin connection information can be given in the design. However, the CAD tools provided by the FPGA manufacturers are not prepared for such kind of low level application. Normally the routing details are hidden and the users do not need to worry about that. The CAD tools will optimize the design for different targets, for example using fewer resources or having lower delay. In our design, as long as the routing, the LUTs functions, and the input pins of the LUTs are identical, the delay comparison between the CLBs is reasonable, because, in that case, ideally the speed of the ring oscillator should be the same. Any speed difference detected by this delay characterization method can prove that some of the CLBs under test are suffering from more process variation or aging effects. Hence, the design optimization is not desired and may in the worst case influence routing and the LUT functions with adverse impact on measurement accuracy.

In [21], it is stated that the inputs of the LUT are logically equivalent. By rearranging the bits in the truth table and the routing, the input pin of LUT to which a signal is connected can be changed. Furthermore, the timing of the LUT only depends on the access time of the SRAM cells and the multiplexers instead of the complexity of the functions. As a result, the placement and routing (PAR) tools may swap the logically equivalent LUT input pins to achieve a delay-optimized routing.

The ring oscillator design here configures the Boolean function of each LUT as a 6-input XOR function. So logically the six inputs of a LUT are equivalent. This leads to a situation that even though we use different LUT input pins and routing in different test configurations as mentioned in the previous subsection, the PAR tool considers they are logically equivalent and swaps the used LUT pins for delay-optimized routing. As a result, all the six different test configurations are modified to use identical LUT input pins and routing after PAR. This completely changes our ring oscillator design, and it fails to cover all the multiplexers in the LUTs.

Therefore we tried to stop the automation routing of the tools from modifying the routing of the ring oscillator design by adding constraints in the User Constraints File (UCF). However, it doesn't work. As long as the a net is unrouted in our design, the tool will rearrange the pins and update the routing. Because the net of the test case signal has no driver in the hard macro design stage, the tool can not route the net. A proxy is introduced to fix this problem as shown in Figure 4.7. A proxy is a CLB which is adjacent to the CLB ring oscillator and it provides the unrouted nets with a driver so they can be routed. The LUTs in the proxy are configured as identity functions. All the external pins are connected to the inputs of the identity function, and the outputs of the identity function are connected to the net these external pins drive.

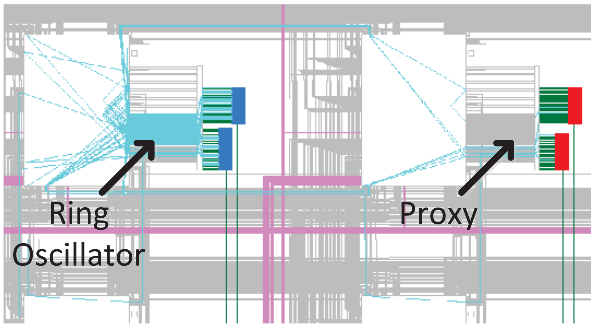


Figure 4.7: CLB ring oscillator hard macro with proxy (screen-shot from FPGA editor)

The PAR tool will not swap the pins of the LUTs as all the nets in the ring oscillator design hard macro are routed. The input pins of the LUTs are routed in a order as we designed. We can generate six different test configurations with 32 test cases by using the proxy. However, as the design uses two CLBs (see Figure 4.7), the CLB which is implemented as a proxy can not be tested using these test configurations. Since we target full coverage of the container, extra test configurations are needed.

4.3 Counter Design and Connection to Ring Oscillator

The counter is used to measure the frequency of the ring oscillator. The counter counts the number of transactions of the ring oscillator during a known time period. This measurement time is defined by another counter clocked with the reference clock. For example, if the fixed reference clock frequency in this work is 100 Mhz, then the time period for 100 reference clock cycles is 1 μs. Hence, two counters are needed. One counter is used as a timer that control the measurement time. It counts the reference clock until it reaches the predefined cycles.

Another counter is used for counting the oscillations of the unknown ring oscillator frequency and it is controlled by the timer.

The equations to calculate the ring oscillator frequency are listed below.

$$t_{measurement} = N_{Ref} * T_{Ref} = \frac{N_{Ref}}{f_{Ref}} \tag{4.3.1}$$

Measurement time, $t_{measurement}$, is the known time period that we define for a single measurement. It can be controlled by the number of reference cycles N_{Ref} and the reference clock period T_{Ref} or frequency f_{Ref}. Here, we define N_{Ref} as the value that is set in the timer.

This measurement time will be also equal to:

$$t_{measurement} = N_{Oscillation} * T_{Oscillation} = \frac{N_{Oscillation}}{f_{Oscillation}} \tag{4.3.2}$$

$N_{Oscillation}$ is the oscillation number of the ring oscillator measured by the counter, and $T_{Oscillation}$ and $f_{Oscillation}$ are the ring oscillator period and frequency.

With Equation 4.3.1 and Equation 4.3.2 we have:

$$\frac{N_{Ref}}{f_{Ref}} = \frac{N_{Oscillation}}{f_{Oscillation}} \tag{4.3.3}$$

So the equation of the ring oscillator frequency can be written as:

$$f_{Oscillation} = \frac{N_{Oscillation}}{N_{Ref}} f_{Ref} \tag{4.3.4}$$

Using Equation 4.3.4 we can compute the ring oscillator frequency by relating the number of oscillations during the measurement time with N_{Ref} and $N_{Oscillation}$ (Figure 4.8).

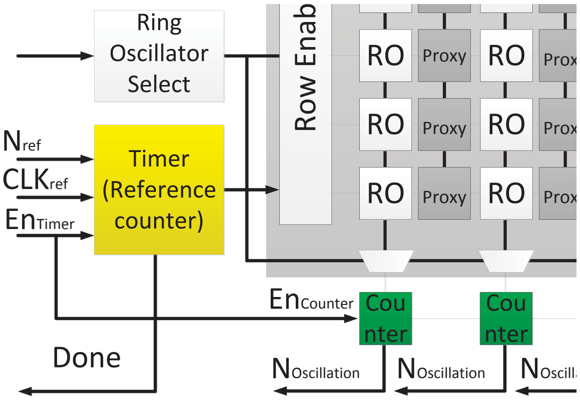


Figure 4.8: Measuring ring oscillator frequency by counters

On one hand, a longer period of measurement may increase the measurement precision. On the other hand, as mentioned earlier in Section 3.1, the ring oscillator will heat up the substrate and the delay will be changed around the loop, a shorter period of measurement is needed to avoid the self-heating [38]. By setting different values in the timer, we can not only study how the ring oscillator self-heating is influencing the temperature and delay, but also study the relation between the measurement time and measurement precision and find out a suitable duration of the measurement. The timer value is set by a dedicated register. With a 16-bit counter, the measurement time can be up to $2^{16} - 1$, i.e. 65535 reference cycles. It's a long measurement time at reference clock considering the short measurement time of 2000 cycles in [38]. So even if the ring oscillator has higher frequency than the reference clock, we can still set a long measurement time at reference clock to conduct the aforementioned experiments and prevent the counter from overflowing. For example, if the ring oscillator is at 300 Mhz, 3 times faster than the reference clock, the timer value can still be set to 20000 reference cycles so that the timer counts until 20000 reference cycles and disable the counters.

As the output of the ring oscillator is connected to a input of the LUT which implements an identity function, the load capacitance of the ring oscillator is fixed. With the decoupling input and output, the measurement result neither depends on the capacitance of enable wire, nor depends on the wire capacitance from the output of the proxy to the counter (Figure 4.9). Hence, no matter where the counters are placed, or how long the interconnect wire between the proxy and the counter is, the results are the same.

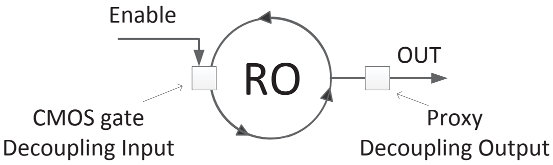


Figure 4.9: Decoupling input and output of ring oscillator

4.4 Control Circuit Design

In this section the three main components (shown in Figure 4.10) of the control circuit will be introduced:

- A serial communication component between FPGA and workstation for receiving/transmitting the commands and collecting the data;
- A data control finite state machine to handle the received data from the workstation and control the serial communication component to send the measurement result of the ring oscillator array to the workstation;
- A ring oscillator array control finite state machine to handle the measurement procedure.

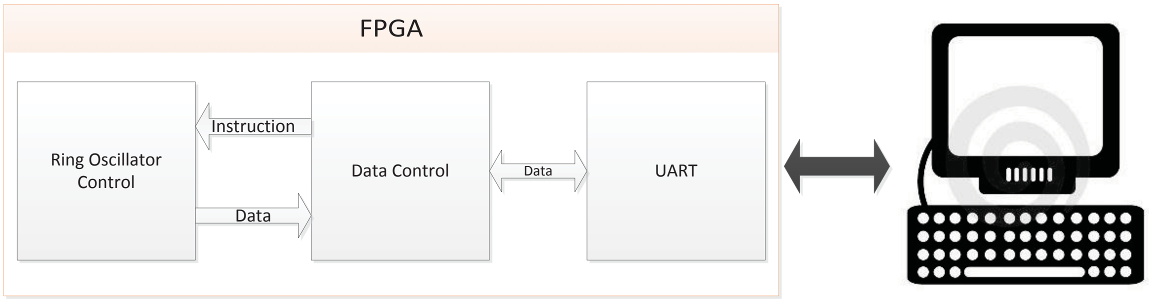


Figure 4.10: Using UART serial communication to control the FPGA

4.4.1 Serial Communication

A Universal Asynchronous Receiver/Transmitter (UART) is used for the serial communication between the FPGA and the workstation. The communication standard RS-232 is used in this work because of easy connection to the workstation and easy software access. One male DB-9 RS-232 serial port is provided by the Virtex 5 ML505 platform [40]. On this platform, the FPGA chip is only connected to the data transmission pin (TX) and data receive pin (RX) on the serial port. Hence the hardware flow-control signals are not used. Therefore the flow control is not used in the communication with the workstation. The configuration of UART is set to odd parity, 9600 Baud rate.

Once the Receiver on the FPGA side notices that a data transmission is coming from the workstation, it will translate a sequence of individual bits into a complete byte. Then it will set the Read Data Available (RDA) signal to '1' to inform the data controller which will be introduced in the following subsection that a data is received. Before the data controller has a new request or the UART is reset, the received data will be stored in the read register and the Receiver will not receive new data from the data transmission line.

The Transmitter on the other hand translates bytes of data and transmits the data bit by bit. As long as the UART receives the write request from the data control, the data will be stored in the shift register and shifted out along with the start bit, parity bit and stop bits. Since transmitting a single byte takes a while, the UART will raise a Transmission Block End (TBE) signal when the previous byte transmission has been finished to inform the data controller and to request the next write.

4.4.2 Data Control Circuit

While the UART receives or transmits data between FPGA and workstation, the Data Control Finite State Machine (FSM) in this work interprets the data from the workstation to command and organise the result transmitting for the UART. The Data Control FSM has six different states: *Wait*, *Start Measurement*, *Set Measure Time*, *Set Test Case*, *Set Ring Oscillator (RO) Select*, and *Send Results* (Figure 4.11). The entry state is the *Wait* state. It waits for the commands from the workstation which are defined in Table 4.1, and enters different states according to the commands.

Table 4.1: Definition of Control Command

Received Data	Command	Action
0	Reset	Set the Instruction signal that tells the ring oscillator array controller to “RESET”; Stay in the *Wait* state.
1	Start	Set the Instruction signal that tells the ring oscillator array controller to “START”; Enter the *Start Measurement* State.
2	Set Timer	Enter the *Set Measure Time* State.
3	Set Test Case	Enter the *Set Test Case* State.
4	Set RO Select	Enter the *Set ring oscillator (RO) Select* State.
5	Send Result	Enter the *Send Results* State.
Others	None	Stay in the *Wait* state.

As it is shown in Figure 4.11, the Data Control FSM drives three different signals. The signal $INTSTR$ is the instruction signal that starts or resets the ring oscillator control FSM. Signal $READ$ and signal $WRITE$ are the control signals of the UART for receiving and transmitting data to the workstation, respectively.

In the *Start Measurement* state, it waits until the WR signal from the ring oscillator array controller changes to 1. The WR signal tells that the measurement is finished and the FSM enters the *Send Results* state to control the UART to send the data of the counters and the temperature sensor to the workstation. After all the results of a single measurement are transmitted, the FSM will go back to the *Wait* state and wait for the next command.

The *Send Results* state can be entered in one of two ways. The first way is entered from the *Start Measurement* state and the second way is entered directly from the *Wait* state when the “Send Result” command is received. By taking the second way, we can resend the measurement results.

The “Set Timer” command drives the FSM to the *Set Measure Time* Sate. To configure the 16-bit timer, two bytes of data are needed. So the following two bytes of received data will be stored in the timer control register of 16-bit width. After receiving the two bytes, it will again return to the *Wait* state and inform the UART to take a new command by setting the READ signal to 1.

The *Set Test Case* and Set RO Select states are similar to the Set Measure time state. But instead of reading two bytes of data, they only read one byte of data. The data is stored in the test case control register and the ring oscillator select control register respectively.

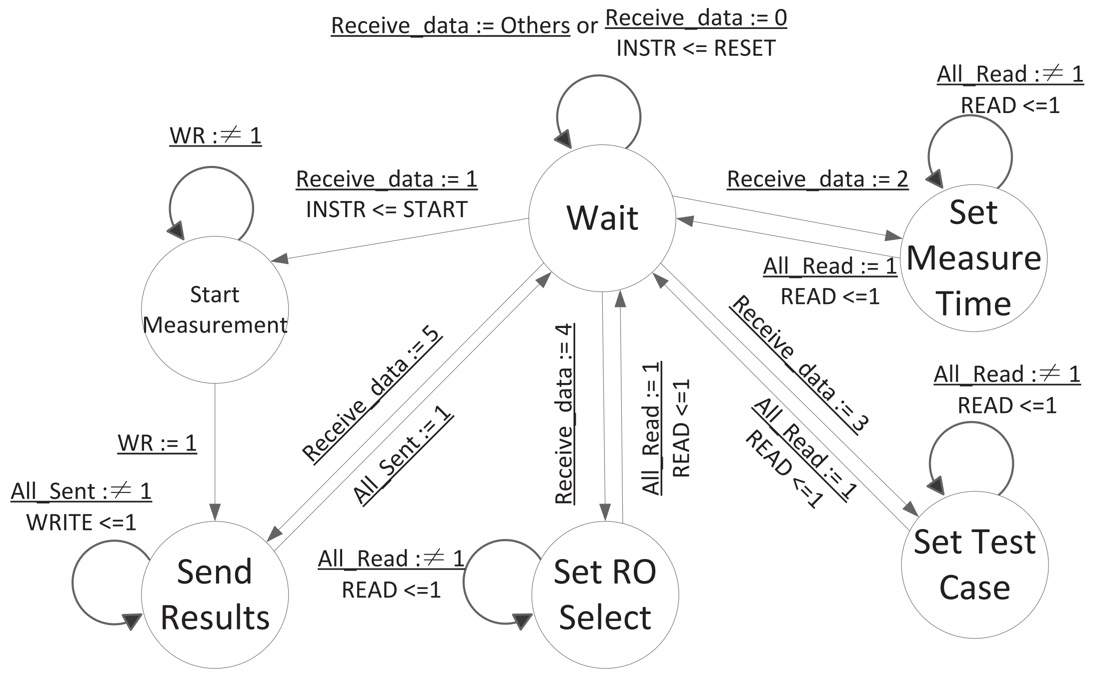


Figure 4.11: The Finite State Machine of the Ring Oscillator Data Control

4.4.3 Ring Oscillator Control Circuit

The ring oscillator control circuit is a FSM, that is used to control the measurement process as is shown in Figure 4.12. It has six different states: *Idle*, *Pre Run*, *Timer Reset*, *Measure*, *Send out* and *Measure End*. The entry state is *IDLE*, after a start signal from the data control FSM is received, it will enter the *Pre Run* state. After the ring oscillator is oscillating at a stable frequency, it will reset the timer in the *Timer Reset* state and then enter the *Measure* state to record the number of oscillations during a measurement time set by the work station. When the measurement is finished, it will enter the *Send Out* state to notify the data control FSM to send out the results in the counters and the temperature sensor. Finally it waits in the *Measure End* state and keeps the results in the counters to be read by the data control FSM to read until it is reset.

As it is shown in Figure 4.12, the FSM drives six different signals. The signals *rst_t*, *en_t* are the reset and enable signals of the timer, while *rst_c*, *en_c* are the reset and enable signals of the counter. Signal *en_r* is the enable signal for the ring oscillator and signal WR is the write signal for informing the Data Control FSM to send of the measurement results.

The *Idle* state is the initial state after reset that waits for the measurement “START” signal to trigger the measurement. In this state, the timer and the counters are reset and the ring oscillator is disabled.

The *Pre Run* state enables the ring oscillator and the timer. When the ring oscillator is enabled, the signal passes through each inverter to be inverted and amplified. Due to the

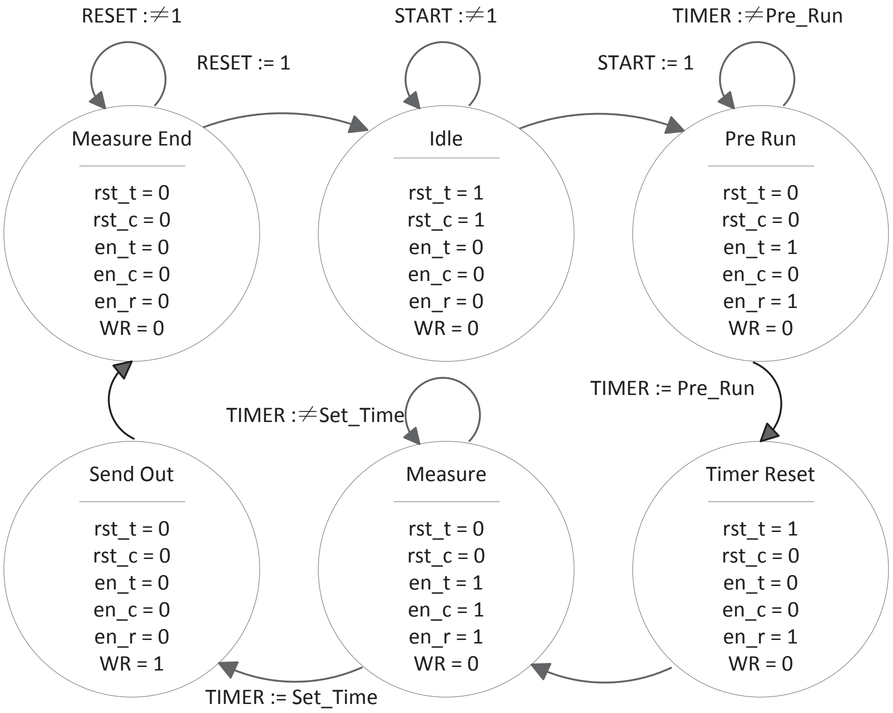


Figure 4.12: The Finite State Machine of the Ring Oscillator Control

delay in the loop, the oscillator needs a start up time to gain a constant frequency. Here in this work, the timer counts for 2^{12} reference clock cycles, following the technique in [29]. However, no *Pre Run* state is mentioned in [38]. In the next state*Timer Reset* state, the timer is reset while the ring oscillator is kept oscillating.

In the *Measure* state, the timer and counters are enable simultaneously. The timer counts the reference clock until it reaches the number stored in the timer register which is received from the workstation and then the FSM will enter the *Send Out* state which disables the timer and counters and informs the data control FSM that the results are ready. All the components are disabled in the *Measure End* state but they are not reset. Therefore, the data control FSM can send out the measurement results when it's requested by the workstation.

4.4.4 Selection wrapper

The container under test comprises a two dimensional array of CLBs configured as identical ring oscillators, which are enabled one row at a time using the row enable signal. A row of counters measure the selected row of ring oscillators simultaneously. Each counter is dedicated for a column and its input is connected to the output of the multiplexer which selects the ring oscillators in the same column depends on the row select signal.

The selection wrapper is designed with VHDL and is implemented in the container. It's also the interface between the static part and the runtime reconfigurable partition of the system. The identical ring oscillator hard macros are instantiated as black-boxes in the selection wrapper.

4.5 System Monitor

The Virtex 5 FPGA contains a System Monitor which is located in the center of the FPGA chip [41]. The on-chip sensors can measure the FPGA physical operating parameters on real time, for examples the on-chip power supply voltages and temperature. The measurement data can be accessed via the JTAG port using ChipScope Pro Tool or be instantiated as a hardware blocks and directly accessed as a module whose maximum measurement error range is ±4 ℃. The resolution of the temperature sensor depends on the LSB of the AD converter, and each LSB approximately makes a difference of 0.49 ℃.

In this work we need to study the correlation between the temperature and the ring oscillator activities so we rely on the System Monitor temperature sensor. The System Monitor block is instantiated with the LogiCORE IP System Monitor Wizard.

Chapter 5

Implementation

Contents

The ring oscillator in this work is a low level design, therefore a suitable implementation platform is needed. This chapter describes the design tools used in this work and the implementation flow in details.

5.1 Design Tools

5.1.1 Xilinx Design Language

As shown in Figure 5.1, the conventional Xilinx Design flow is:

1. Describe the design with high level Hardware Description Language(HDL), for example VHDL or Verilog;

2. Synthesize the design;

3. Implement the design (Map, Place and Route);

4. Generate the bit file.

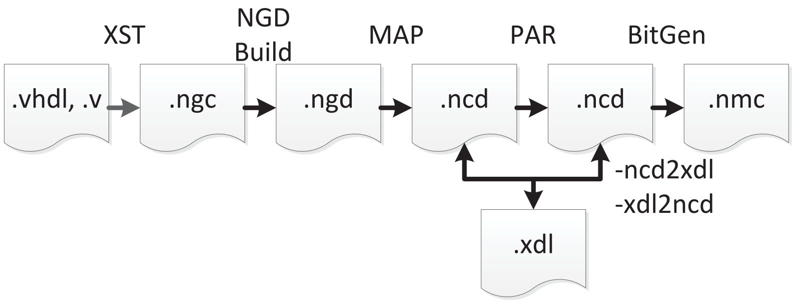


Figure 5.1: Xilinx Design flow with XDL

A Native Circuit Description (NCD) format file is generated when the design is mapped. The Xilinx Description Language (XDL) is human readable format and equivalent to the NCD format. It can be used to manipulate the design, either before or after the design is full or partially placed or routed. Meanwhile, Xilinx provides the command line tool "xdl" for the conversion between these formats: $xdl - ncd2xdl$ and $xdl - xdl2ncd$.

Even though XDL is not well documented, it gives the researcher or designer an opportunity to study all the features of the Xilinx device, to constraint the system or to directly manipulate the internal routing and logic elements to create a hard macro. In Section 4.2.2, the hard macro is already introduced as a pre-placed and pre-routed module for a specific device family.

Because it can directly manipulate the tiles in Xilinx FPGAs that represent the CLB, interconnect, clock, block RAM and I/O block, XDL is a suitable language for low level design. It also has the advantage to reduce the design time because the synthesis and mapping steps can be skipped. However, to use this powerful language, a compatible tool is needed. One possible tool is called RapidSmith and it is introduced in the next section.

5.1.2 RapidSmith

RapidSmith is an open source project that is based on XDL. It is a framework that is written as a collection of Java packages for reading, writing, and manipulating Xilinx designs in XDL format. The detailed FPGA device information is in the XDLRC FPGA resource description which has been parsed by RapidSmith. With the RapidSmith library, it's possible to manipulate logic function in the CLB tile and connect the components in the tile to a specific pin or port. Hence the low level design of ring oscillator can be directly generated with the help of RapidSmith as a XDL file which can be later converted to the NMC hard macro format file.

5.1.3 Xilinx ISE Design Suit

The ISE design suite is the Xilinx FPGA design environment that includes exclusive tools which are used in this work including: ISE, FPGA editor, PlanAhead and iMPACT.

ISE Except for the ring oscillator design, the control circuit including the UART communication control unit, data control unit, ring oscillator control unit, the selection wrapper and the system monitor are all synthesized in ISE to generate the NGC file, which is later used as the static part in PlanAhead.

FPGA editor is used to route the ring oscillator design with the proxy with a script which is generated along with the XDL file of the ring oscillator.

PlanAhead PlanAhead is used to combine the static part and the reconfigurable part of the design. It can generate the whole bit stream for the complete design and the partial bit streams for the different configurations. As the ring oscillator hard macros are used in the implementation flow, their placement has to be specified in the user constraints file (ucf). In the ucf file the placement has to be individually set for each macro with a reference point.

iMPACT iMPACT is used to program the FPGA.

5.1.4 Python Backend

In section 4.4.1, it is mentioned that the FPGA communicates with the workstation using UART serial communication. The Python pySerial package can access the serial port. With the help of python, we can easily control the measurement process and record the data in a local file in workstation.

5.2 Implementation Flow

The implementation flow of the delay characterization method in this work is shown in Figure 5.2. Firstly, the ring oscillator hard macro with proxy is generated by RapidSmith and converted to the NMC file. Then this hard macro is routed using a script in FPGA Editor. The static part of the design is synthesized in ISE and combined with the ring oscillator array partial reconfigurable partition in PlanAhead which instantiates the routed hard macro in the black box (container) of the static part. The design combined the static and partial reconfigurable partition will be completely placed and routed in PlanAhead. Finally the generated bitstream is downloaded to the FPGA via iMPACT.

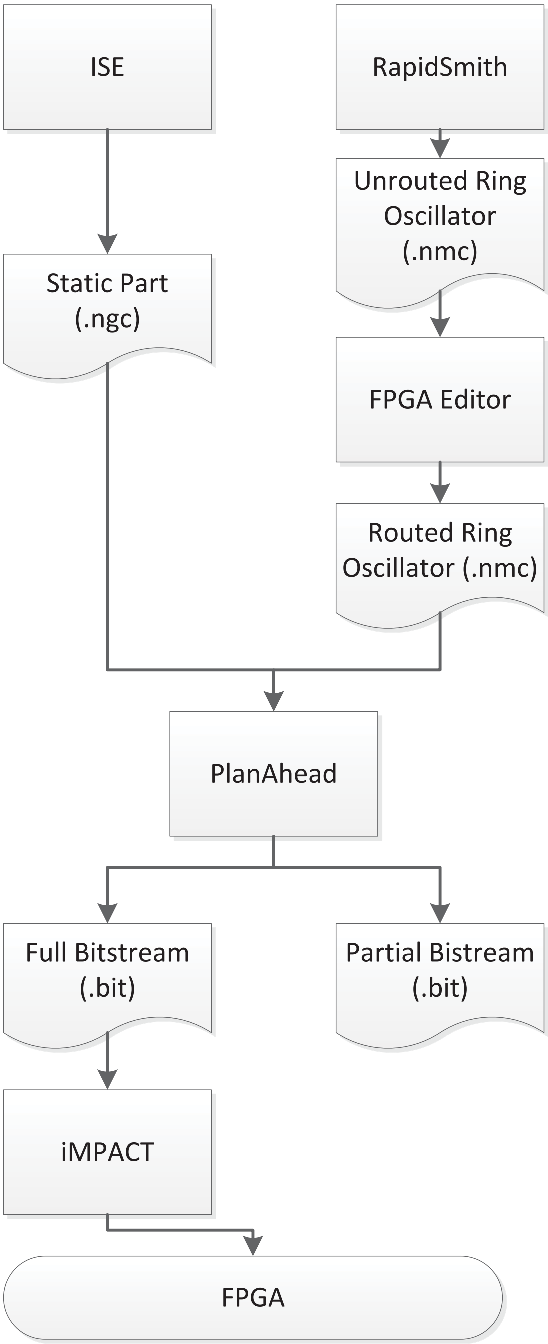


Figure 5.2: Implementation flow of the delay characterization method

Chapter 6

Results

Contents

In this chapter, the validation of the delay characterization method and the obtained results are presented. Firstly, a case study of a single ring oscillator shows the low level implementation flow. Then the arrays of ring oscillators are validated and used to characterize the delay in the FPGA. The validation considers the measurement precision, accuracy, and proves the absence of systematic problems. The measurement results are also discussed with respect to delay variation and temperature dependency.

6.1 Case Study: Single Ring Oscillator

In the first experiment, a single ring oscillator is used to prove the methodology using the design flow with low level HDL can be performed. The ring oscillator is designed and simulated, then implemented on the FPGA hardware.

A 31-stage and a 63-stage ring oscillators are designed. The design is controlled by a very simple FSM, once the enable button is pushed, the measurement starts, and timer, counter and the ring oscillator are enabled. The timer counts for 100 cycles of the reference clock at 100 MHz and then disables the counter. The result is stored in the counter. And the ring oscillator is kept oscillating until it is reset.

6.1.1 Simulation Result

The design generated with XDL has to be verified before it's implemented in the FPGA with simulation. Xilinx design tool is used to convert the implemented design into a VHDL simulation model using the SIMPRIM library. The SIMPRIM library is used for structural simulation netlists produced after implementation including timing simulation. The simulation model is generated along with a Standard Delay Format (SDF) file which annotates the delays for the design.

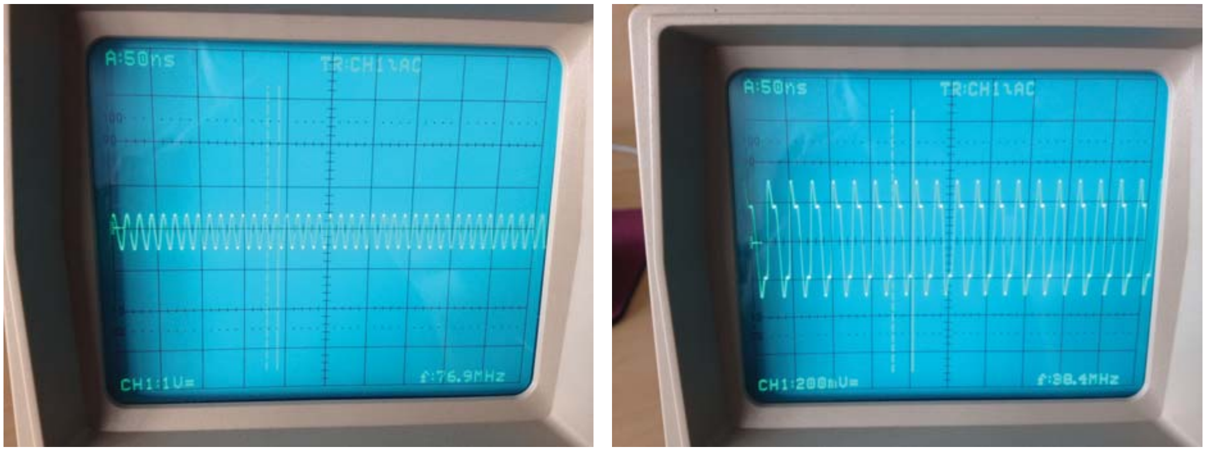
The simulation is performed in Modelsim and shows that the ring oscillator design is oscillating and the counter is counting the oscillations. Once the test starts, the counter is enabled until it is disabled by the timer after 1000 ns, 50 transactions are recorded so that the 31-stage ring oscillator runs at a frequency of 50 MHz in simulation.

6.1.2 Implementation Result

Design is now loaded onto a FPGA and the frequency of the implementation is measured in two ways:

- External measurement using oscilloscope. The output of the ring oscillator is connected to the expansion I/O connectors which are connected to an oscilloscope.
- Internal measurement using built-in counter.

In this experiment, the used oscilloscope has a measurement range of up to 100 MHz, hence frequency divider which reduces the ring oscillator frequency to 1/2 and 1/4 of the original frequency is used to improve the measurement precision. First, several expansion I/O connectors on the FPGA board are used, one of them is connected to the output of the ring oscillator directly and the others are connected to the output of the frequency divider. The oscilloscope is connected to the expansion I/O pin and the ground pin on the FPGA board to measure the frequencies. Figure 6.1 shows the frequency of the 31-stage ring oscillator is 76.9 MHz.



(a)

(b)

Figure 6.1: Measurement result with oscilloscope via the expansion I/O pin

As the communication circuit between the FPGA and the workstation was not designed before this case study, the 8-bit LEDs on the FPGA board are used to display the counter value. The LEDs read from the MSB to LSB is 01001100, which is equal to 76 MHz.

The case study shows that both the oscilloscope and the built-in counter can be used to measure the ring oscillator frequency. The results from the oscilloscope and the built-in counter are mutually confirming the correctness of each other. As a comparison, the frequency of the 63-stage ring oscillator was also measured and the result is 38.4 MHz, approximately half the frequency of the 31-stage ring oscillator.

However, the oscilloscope in this experiment has a limited frequency measure range and it's difficult to record the results. By contrast, the built-in counters will work at frequencies higher than 500 Mhz according to the timing report and the test results can be stored in an automated way. Besides, in the real design, it may not be possible to spare a pin for the oscilloscope measurement. Hence, the oscilloscope is used here as a reference.

6.1.3 Comparison of Simulation and Implementation

The simulation result (50 MHz) is pessimistic when it is compared to the implementation result (76.9 MHz) . One explanation could be that the manufacturers have taken the process variation and aging effects into account, and thereby pessimistically estimate the speed of the fabric. Similar results can be seen in [36] where the measured delay is 55.9% of the maximum simulation delay.

The first experiment setup shows that the ring oscillator can be implemented on the FPGA properly. The read out from the oscilloscope proves that the built-in counter measures the oscillations correctly. However, the accuracy has to be improved as the measurement time, 100 cycles, is too short. When an 8-bit counter is used the maximum frequency can be measured is $2^8 = 256$ Mhz for a the reference clock of 100 Mhz and 100 measurement cycles is set to 100.

6.2 Validating the measurement of Ring Oscillator Arrays

The delay characterization method in this work is physical measurement that is not possible to be completely accurate. In order to reliably interpret measurement data, different experiments are proposed in this section to validate the measurement. The feasibility of the method has been proved by the case study of single ring oscillator. In the following experiment, the ring oscillator design mentioned in Chapter 4 are validated and it is instantiated in different size of arrays.

Each time the one row of ring oscillators in the array is enabled and a row of counters measured the enabled row of ring oscillators simultaneously. One of the six test configurations using different ring oscillator paths is shown. As each ring oscillator path consists of the input pins with the same pin number of LUTs, the five remaining LUT input pins are with different assignments to evaluate all paths in the LUTs. The presented results show that the measurement method is precise and accurate.

6.2.1 Delay Estimation

The delay estimation is based on the information from FPGA Editor and the SIMPRIM model of the ring oscillator. The Xilinx timing analysis tool can estimate the delay for the route and this delay estimation result is used to compare with the measurement result.

For example, the test configuration which use input Pin 6 on the loop, the total path delay including the delay of the LUTs and the intra CLB path is calculated. Estimation of the LUTs delay is $8 \times 0.086\ ns = 0.688\ ns$, the intra CLB path delay is $2.545\ ns$, the XOR gate delay is $0.117\ ns$, total delay is $3.351\ ns$ so that the frequency is calculated with the formula $f = 1/T = 1/(2 \times (Delay\ of\ the\ inverters + Delay\ of\ theinterconnect)) = 1/(2 \times 3.351\ ns) = 149\ MHz$.

The interconnect resources mentioned in Section 2.2.3 connect the LUTs within a CLB. However, some of the LUTs are directly connected, while others need to use more PIPs as shown in Figure 2.4. The delay of a connection passing through more PIPs is larger than that of a direct connection. In this work, the pin to pin connections between LUTs are particularly specified, for example, in the configuration that all input pins 1 are used in the oscillation loop, the output of one LUT is always connected to the input pin 1 of the next LUT. However not all the connections are direct connections and this specified connections may have large delay due to the structure of the programmable switch matrix. Since the connections are not optimized for delay, the path delay of some test configurations are larger than the other. Table 6.1 lists the path delay estimation for different test configurations.

Table 6.1: Estimated delays for six test configurations

Test Configuration	Intra CLB path delay (ns)
Pin 1	7.825
Pin 2	6.838
Pin 3	4.737
Pin 4	4.737
Pin 5	2.963
Pin 6	2.545

6.2.2 Measurement Precision

Before using the proposed method to characterize the delay of the CLB, it is required to find out the precision of the built-in counters. As it is mentioned in Section 4.3 Counter Design, the effect of the measurement period on the measurement precision will be examined.

The measurement precision is reviewed for different measurements period, varying from 1000 to 25000 reference clock cycles. For each measurement period, the experiment is conducted at the room temperature and the Virtex-5 FPGA device is first put in the idle mode until it reaches the temperature equilibrium. An array of 5 × 20 ring oscillators is measured. The experiment uses one test configuration and one test case. Ten consecutive measurements for each measurement period were taken. The standard deviation of each ring oscillator is calculated and the maximum standard deviation in the array in each case is listed in the table 6.2.

Table 6.2: The standard deviations (Mhz) for different measurement cycles

Measurement Cycles	Max. Standard Dev.
1000	0.485
2000	0.255
3000	0.098
4000	0.136
5000	0.119
6000	0.137
7000	0.105
8000	0.120
9000	0.138
10000	0.141
15000	0.153
20000	0.112
25000	0.108

The maximum standard deviation among the results is 0.485 Mhz, considering the mean frequency of the ring oscillator array is 232 Mhz, it is a 0.2% measurement error.

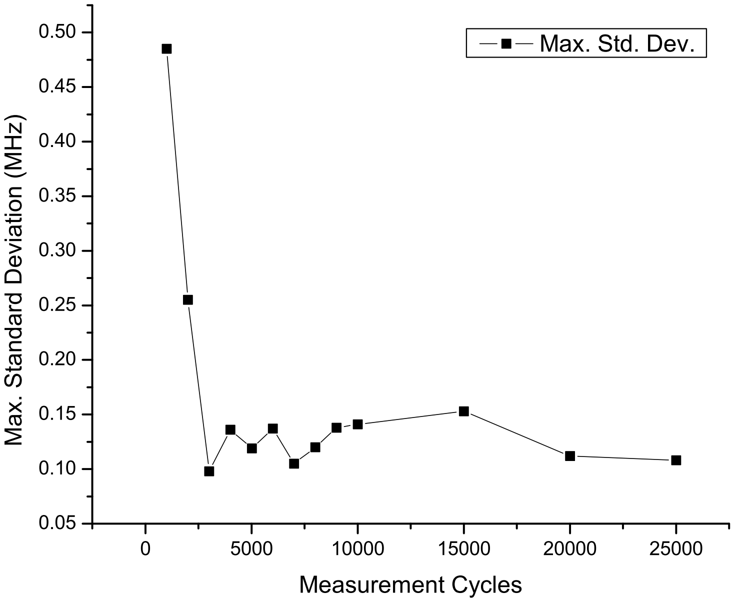


Figure 6.2: The correlation between reference clock cycles and measurement precision

We can see from Figure 6.2 that the maximum standard deviation declines as the measurement time increases from 1000, and it reaches a minimum value at 3000 cycles. After that the longer measurement cycles can not reduce the maximum standard deviation as the self-heating of the ring oscillator may decrease its frequency during the measurement. Considering the trend of the maximum standard deviation and the fact that longer measurement takes more time and increases the temperature by self-heating, 3000 cycles is a suitable measurement time.

6.2.3 Measurement Accuracy

All the measurements are prone to systematic error, if the measurement contains a systematic error, then increasing the sample size only increases the measurement precision but does not improve accuracy. The result would be a consistent inaccurate results from the flawed measurement method. Eliminating systematic errors, if present, improves accuracy. The following experiments aim to investigate systematic errors in the measurement setup/design.

6.2.3.1 Measurement with different Counter Placement

The following experiments aim to investigate the influence of counter position to the measurement results. In Section 4.3 we discussed the decoupling at the input and output of the ring oscillator design that provides a static load capacitance of the ring oscillator. The load capacitance of the interconnect between the ring oscillator output and the counter input should, thereby not influence the ring oscillator frequency.

The experiments are conducted at room temperature. An array of 5 × 20 ring oscillators is placed at the location X48Y94. And the counters are placed in five different locations including center and the four corners of the FPGA chip. To increase the measurement precision, each ring oscillator is measured ten times and the average value is used for the comparison. We name the five configurations *Center*, *Upper Left*, *Upper Right*, *Lower Left* and *Lower Right*, representing the location of the counters. The measurement results of the ring oscillators in the array in *Center* case are used as reference results and are compared with other cases. The maximum difference in the array of each comparison is listed in the table 6.3 with the average value of the array of each case. The temperature is at 38.39 ℃.

Table 6.3: Results comparison for the ring oscillator array of different counter locations

Case	Average Feq. (MHz)	Average Diff. (MHz)	Max. Diff. (MHz)
Center	232.48	NaN	NaN
Upper Left	232.47	0.01	0.79
Upper Right	232.52	0.04	0.57
Lower Left	232.52	0.04	0.74
Lower Right	232.68	0.20	0.56

The maximum measurement difference due to the changed counter locations is 0.79 MHz and minimum frequency of the ring oscillator in the measured array is 227.94 MHz. As a result, the maximum influence of the counter location is only 0.35%. As a consequence, we can conclude that the measurement of ring oscillator frequency doesn't depend on the placement of the counters.

6.2.3.2 Measurement of Random Area

The intention of this experiment is to show that the purposed delay characterization method can measure delay variation patterns in different locations of the FPGA and thereby prove absence of the systematic problem. An array of 5 × 5 ring oscillators is randomly placed at different locations in the FPGA. Detailed experiment results of one test configuration and one test case are listed in Table 6.4.

The Table 6.4 shows that the maximum frequencies in different locations are not the same. The highest maximum frequency is 236.53 MHz in the array at X8Y154, while the lowest maximum frequency is 230.22 MHz in the array at X72Y74. The difference between the them is 2.7%. Moreover, the highest minimum frequency is 231.14 MHz in the array at X48Y94 and the lowest minimum frequency is 223.00 Mhz in the array at X8Y84 with a difference of 3.6%. And within the arrays, the difference between the maximum and minimum frequency is from 1.8% to 5%.

Table 6.4: Results comparison for the ring oscillator array at random locations

Location	Maximum Feq. (MHz)	Minimum Feq. (MHz)	Max. − Min. Feq. (MHz)	Average Feq. (MHz)	Std. Dev. (MHz)	Temp. (℃)
X8Y4	234.82	227.07	7.75	231.66	2.09	40.85
X8Y84	230.75	223.00	7.75	228.17	2.21	39.37
X8Y90	231.80	224.11	7.69	229.15	2.17	40.85
X8Y94	232.59	224.64	7.95	229.37	1.97	40.36
X8Y154	236.53	224.90	11.63	231.22	3.03	40.85
X48Y90	236.07	230.49	5.58	232.72	1.57	40.36
X48Y94	236.33	231.14	5.19	232.80	1.41	40.85
X72Y4	232.39	226.28	6.11	229.01	1.61	41.83
X72Y70	230.95	225.62	5.33	228.85	1.52	39.87
X72Y74	230.22	226.15	4.07	228.84	1.11	40.85
X72Y154	235.02	226.68	8.34	231.07	2.41	40.85

The experiment results show that the setup/design can actually characterize the delay of the CLBs and in a certain degree prove that there is no systematic problem.

6.2.3.3 Measurement of Overlapping Area

The purpose of this experiment is to show that the measured frequency of the ring oscillator at any location does not depend on the placement of the ring oscillator to prove that there is no systematic problem in this work.

One of the experiments is shifting the 5×5 ring oscillator array by one row or/and one column to measure an area which is an intersection of four ring oscillator arrays (Figure 6.3). The measurements are taken for one identical configuration and test case. The system monitor reports the temperature from 41.83 to 42.33 ℃.

The 4×4 area selected by the black rectangle shows the overlapped CLBs which are measured by ring oscillator arrays at four adjacent locations. For example, the location in Figure 6.3 (b) is equal to the one in Figure 6.3 (a) shifted one column to the right and the one in Figure 6.3 (d) shifted to an upper row. The delay variations which are measured in these four cases of this overlapped area have identical spatial distribution which is also seen in the patterns in Figure 6.3. The standard deviation of these four cases is less than 0.5 Mhz.

6.2.4 Summary

The experiment results in this section show that the delay characterization method in this work has high precision and accuracy, and does not have serious systematic errors. The

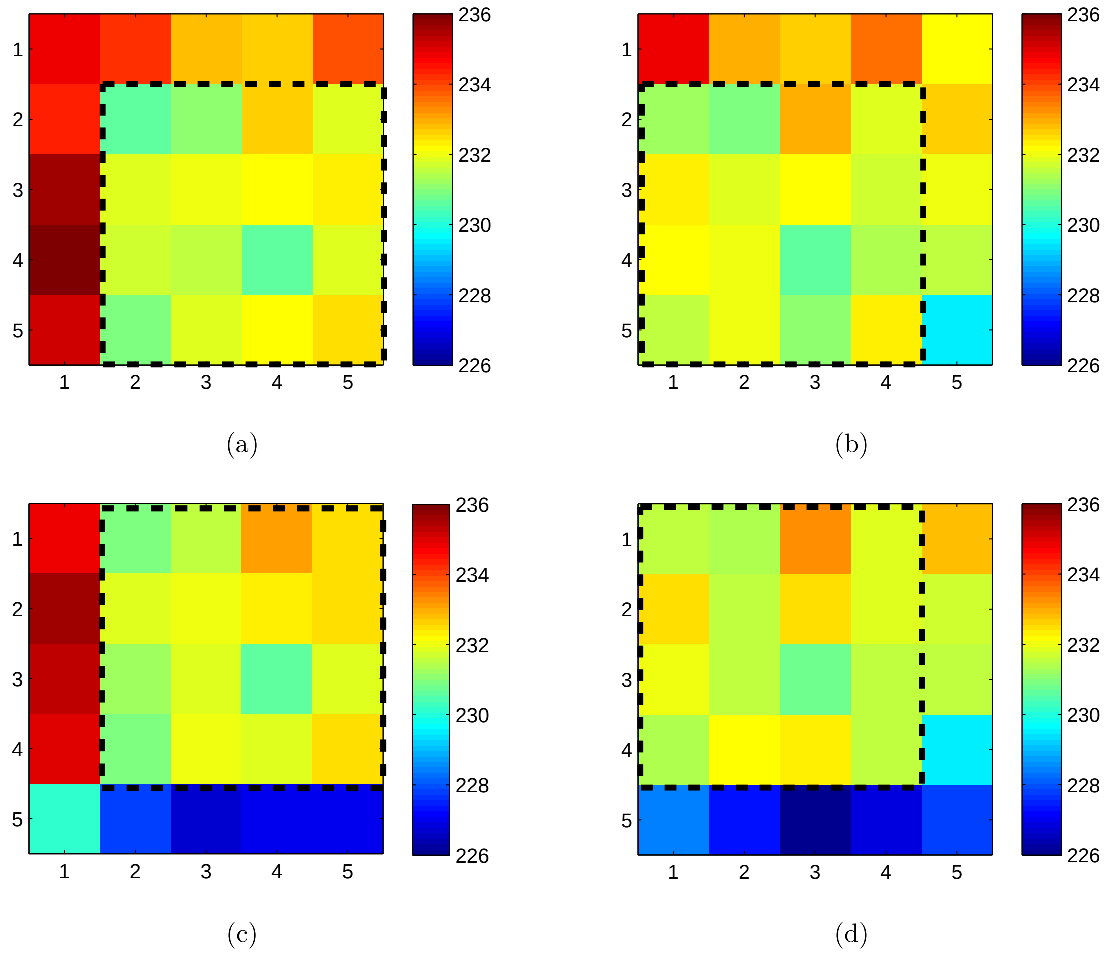


Figure 6.3: Shifting the ring oscillator array in four positions

measurement results of the delay values at the same location are repeatable and it is not depending on the placement of the ring oscillator array or the counter.

6.3 Delay Variation

The purpose of FPGA delay characterization method is to detect and localize the degradation of the basic reconfigurable logic element, LUT. The following experiment aim to demonstrate that the method can be adapted to different size of the container and characterize the delay. Here in this experiment, we assume the container size is 10 × 20 CLB.

As mentioned in Chapter 4, because of an implementation problem, a proxy is introduced to the ring oscillator design. As a result, the ring oscillator array can only measure half of the CLBs in the area it covers. To characterize the delay of the whole target area, at least two configurations of ring oscillator array are needed. The characterization result of a 10 × 20 CLB area measured by two 5 × 20 ring oscillator arrays is shown in Figure 6.4.

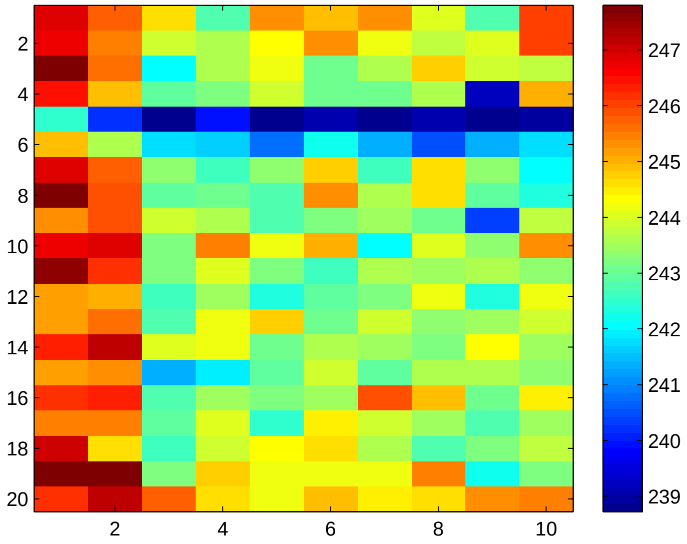


Figure 6.4: A 10 × 20 CLB area measured by two ring oscillator arrays

In this case, the temperature is at 40 ℃, the Input pins 6 of the LUTs are on the oscillation loop and the test case control signal is set to 31. The average frequency of this measured area is 243.82 Mhz. The standard deviation of it is 1.79 Mhz (Figure 6.5). The maximum frequency is 247.80 Mhz while the minimum is 238.72. The frequency difference between the maximum and minimum is 9.08 Mhz, which is 3.7% of the average frequency in this area. The result also shows that the delay estimation by the timing analysis tool is pessimistic, as the estimated frequency of the ring oscillator is 149 MHz, 61% of the average frequency in this case.

The slowest rows of CLBs are close to the global interconnect routing channels in the FPGA, hence it is highly possible that they are more frequently used than the one which has a longer distance to the global interconnect, and for that reason, may have already aged during previous operation, for example the CLBs in row 19 and row 20 in Figure 6.4.

6.3.1 Test Cases

In this work the ring oscillator is designed with the feature to cover all the paths in the LUTs based on the assumption that the LUTs are with the structure mentioned in Section 2.2.2.1. In the following experiment, we enumerate the 32 different test cases by changing the 5-bit test case control signal for one of the six test configurations each time. We assume that for each test cases, the delay on the ring oscillator loop would be different because different multiplexers in the LUT and thereby different paths are elected. A 5 × 20 CLB array is

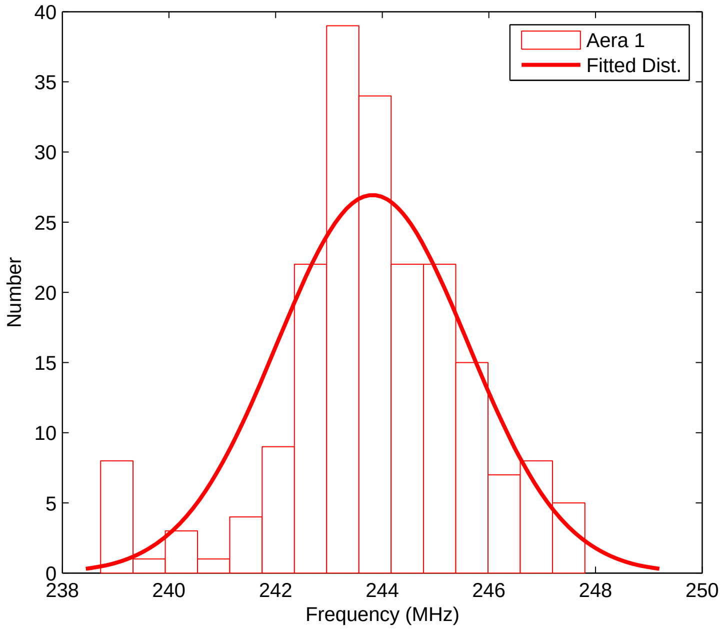


Figure 6.5: Frequency distribution of 10 × 20 CLB area

placed at coordinate X48Y94, and Figure 6.6 shows the test results of the CLB at coordinate X48Y114 as an example.

The experimental result differs from the assumption. The result shows that the ring oscillator frequencies can be grouped into two groups, where the first group is the high frequency group for the test cases with a frequency at around 247 Mhz, and the second group is the low frequency group for the test cases with a frequency at around 235 Mhz. The difference between the average frequency of these two group is 11.64 MHz, which is about 5% of the average frequency. The standard deviations of the fast and slow group are 0.193 and 0.192 respectively. These results also show that this observation is not only true for a single CLB but all the CLB ring oscillators. When we observe the frequency distribution of these 32 test cases, we can see that actually it is of some pattern, as seen in Figure 6.6. The observation result for the first nine test cases is shown in Table 6.5.

The frequency change seems to depend on the parity of the 5-bit test case control signal. When the parity of the test case is even, this test case belonged to the low frequency group. In contrast, the odd parity test cases have a high frequency. This can not be explained by the LUT structure which we assumed, as the change of the test cases is only choosing a different path in the LUTs. The path lengths are the same when the test configuration is not changed. Considering the test cases 0 and 1, the paths are very slightly changed. These kinds of the small difference may not cause such a significant change in ring oscillator frequency up to 5%. The current understanding of the LUT structure seems to be inaccurate. But still, as the LUTs are implemented with the 6-input XOR functions and all the input combinations are

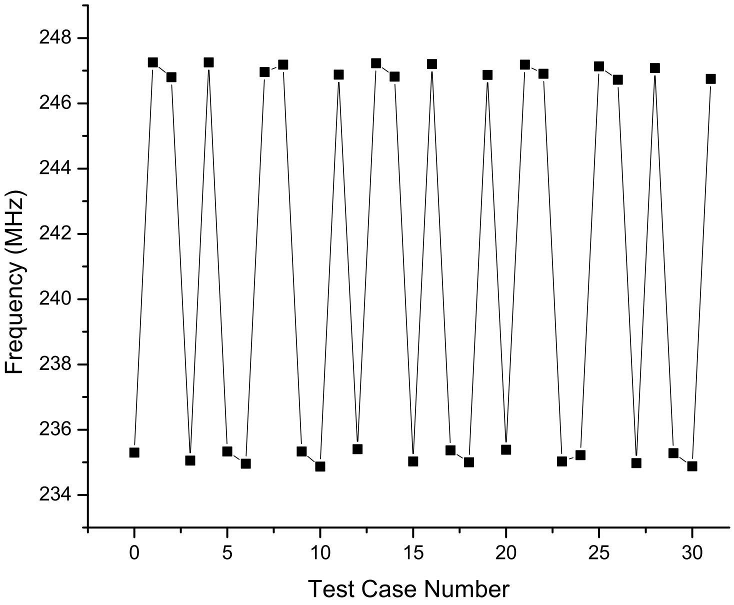


Figure 6.6: Frequency comparison of 32 different test cases for a single CLB ring oscillator in one test configuration

Table 6.5: Parity dependence of the frequency

Test Case	Binary	Frequency
0	0000	low
1	0001	high
2	0010	high
3	0011	low
4	0100	high
5	0101	low
6	0110	low
7	0111	high
8	1000	high

covered, every path that accesses the SRAM bit entries is tested and characterized.

No matter what the hidden LUT structure is, the function and the input pins of the LUTs are identical for all the ring oscillators in a same test configuration and test case. As the ring oscillators in the array are using the same hard macro, it is ensured that they have an identical logic function and routing, so the delays of each CLB in different test cases are compared. Hence, the ring oscillator frequencies reflect the delay variation from CLB to CLB.

6.4 Temperature Dependency of Delay

The following experiments investigate the temperature influence on the ring oscillator frequency and the measurement operations influence on the temperature. Two types of experiments are presented in this section, one is using an external heating equipment to control the temperature, the other is measuring the self-heating in the FPGA caused by the proposed delay characterization method.

6.4.1 External Heating

The setup of the external heating experiment is shown in Figure 6.7. The heating equipment is refitted from a hot air gun which is normally used for desolering. The fan of the FPGA board is removed and the muzzle of the hot air gun is attached to the heat sink of the FPGA board. We can adjust the heating speed by controlling the temperature and the air pressure of the hot air gun.

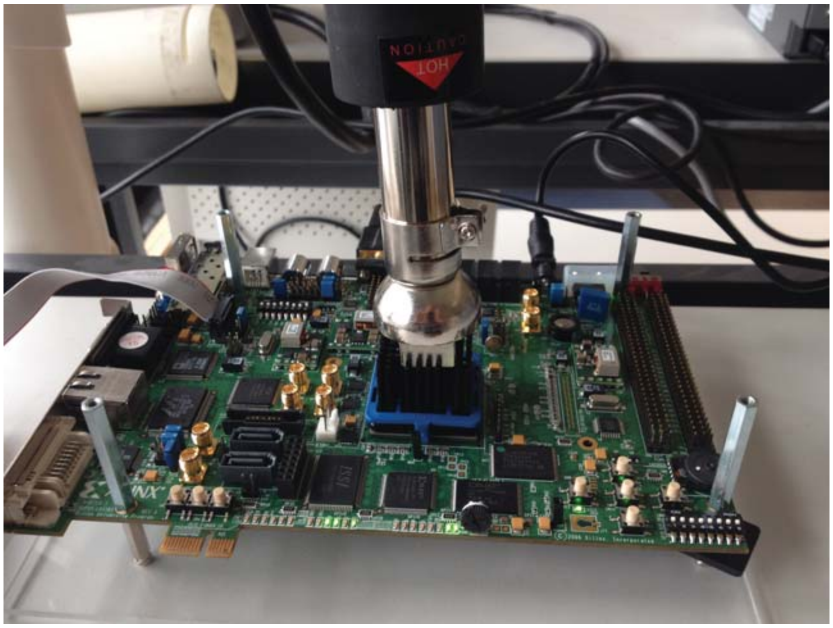
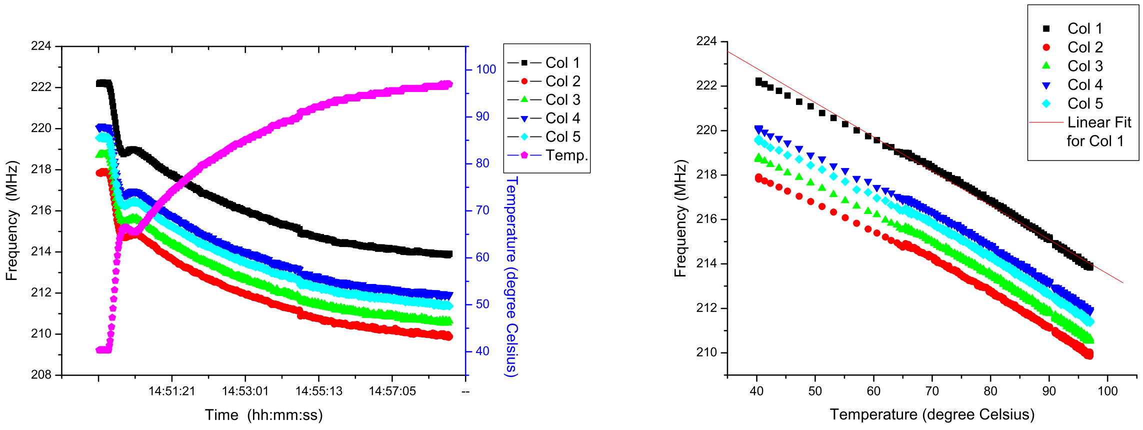


Figure 6.7: The experiment setup for external heating

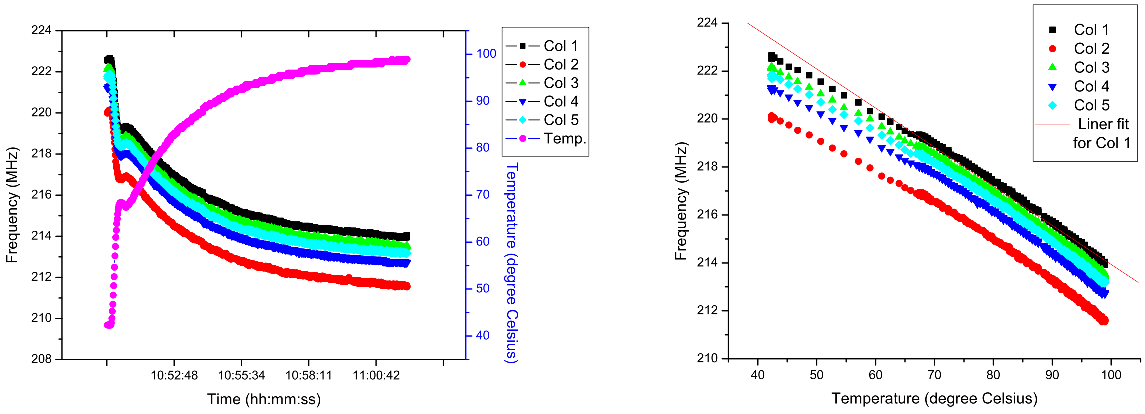
In the external heat up experiment using the hot air gun, two rows of the ring oscillators in the array mentioned in Section 6.3 are chosen. In both of these two cases, hot air gun temperature is set to 100 ℃, used the same heating air pressure. After each experiment, the hot air gun is turned off, the FPGA is put in the idle mode until it cools down and reaches the temperature equilibrium and then we start another experiment. The heating experiment lasts for ten minutes and results are sampled every second.

As we would expect when the FPGA is heated up, the temperature raises and the ring oscillator frequency drops down, shown in Figure 6.9 (a). From Figure 6.9 (b) we can observe that the temperature and frequency have an approximately linear correlation. Moreover, as the lines which depict the frequency decline of the ring oscillators are parallel, it shows that the frequency-temperature correlation is not depending on the ring oscillator speed. That means in a reasonable operating temperature range (below 100 ℃), the frequency differences between every two ring oscillators are constant. Even though the absolute path delay will increase as temperature rise, the delay variation among the CLBs can be well measured by the ring oscillator array.



(a) (b)

Figure 6.8: Temperature and frequency correlation of CLB row 5 for array at coordinate X48Y94



(a) (b)

Figure 6.9: Temperature and frequency correlation of CLB row 9 for array at coordinate X48Y94

Linear regression analysis is run for the two rows of ring oscillators. The correlation coefficient is -0.996, and the slope is from -0.14 to -0.16 Mhz/Kelvin, which means the frequency will drop 140 to 160 Khz when the temperature increases by 1 K.

6.4.2 Self-Heating

This experiment is performed on a Virtex-5 FPGA with heat sink and a disabled fan. An array of 5 × 20 ring oscillators is placed closely to the System Monitor Sensor in the middle of the FPGA chip. The FPGA device is first put in the idle mode until it reaches the temperature equilibrium and then whole CLB array is measured continuously for 600 seconds.

The temperature raised by 3 ℃ during the 600 seconds and reaches a temperature equilibrium as shown in Figure 6.10. The chart looks discrete due to the minimum temperature resolution provided by the System Monitor temperature sensor is 0.49 ℃. Continuous measurement keeps the ring oscillator in the CLB oscillating and the temperature change is not significant when compared to the [26] as fewer resources are used and the toggle rate is lower in this experiment.

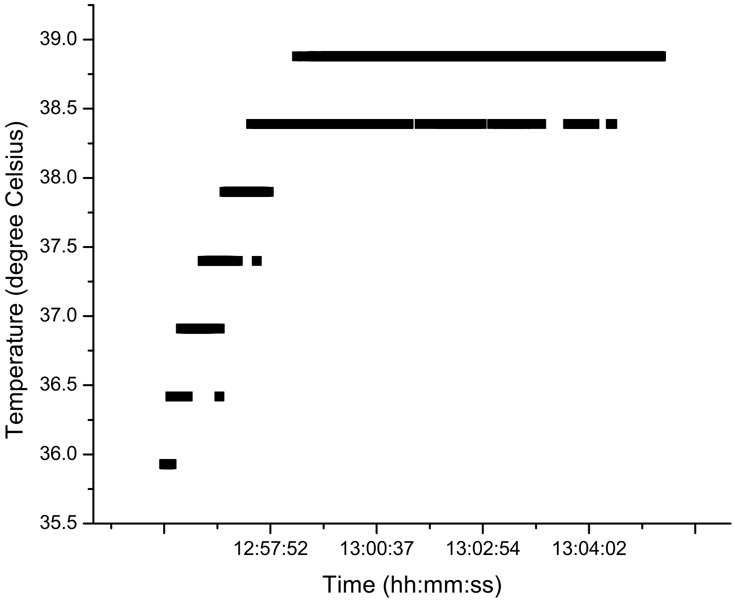


Figure 6.10: Heat up due to continuous measurement

In a complete measurement for all the 32 test cases for each configuration, the maximum temperature changes is 0.49 ℃, which results in a frequency decrease between -0.07 and - 0.08 MHz according to the aforementioned experiment result. This factor contributes to the measurement inaccuracy.

6.4.3 Summary

In this section, the external heating experiment shows that the frequency-temperature correlation is not depending on the ring oscillator speed. The delay variation among the CLBs can be measured by the ring oscillator array at different temperatures. Moreover, in the self-heating experiment, the switching activity of the measurement does not have big impact on the temperature, and thereby the influence to the measurement accuracy is small.

Chapter 7

Conclusion and Future Work

Contents

7.1 Conclusion

In this work, a delay characterization method for FPGA was developed and studied. This method can be used in runtime reconfigurable system. In a runtime reconfigurable system, a part of the FPGA can be used as hardware accelerator dynamically. To avoid using degraded logic elements, a delay characterization can be performed to characterize the target area before configuration and use as accelerator.

Literature about delay measurement was reviewed. Different methods were studied, analysed and compared to determine a suitable delay characterization method that provide fine granularity and high accuracy. The delay characterization using ring oscillator was chosen because the ring oscillator can be running at its maximum speed which is determined by the FPGA fabric itself. Comparing to the at-speed delay characterization, the ring oscillator characterization method does not require extra clock control and can reduce the implementation complexity.

The ring oscillator mentioned in the literature was extended by using specific input pins of the LUTs and by configuring the LUTs with XOR functions. Instead of only testing one path in the LUT, all the paths in the LUTs were tested with the proposed method. The ring oscillator design was generated with XDL language using the open-source tool RapidSmith. The implementation was done using the Xilinx tool flow. The characterization and measurement experiment was controlled by the workstation via a UART interface. Data was processed and analysed on the workstation side.

A series of experiments were proposed to ensure the precision and accuracy of the method. The implementation results on Virtex-5 show that this delay characterization method assesses the speed of CLBs in the area under test at a high accuracy. It allows to detect and localize slow elements in the FPGA fabric. The method can be applied to reconfigurable area of different sizes, as required by the reconfigurable system. Only 6×2 configurations are required to have a full overage of all the CLBs in area under test.

The temperature dependency of the delay characterization method was also considered. The FPGA was heated up with the external heating equipment. The delay characterization was conducted at different temperatures. While the ring oscillator frequency decreased approximately linearly with increased temperature, the spatial delay variation between different FPGA elements was constant. Besides, the self-heating of the method did not have a significant influence on the temperature.

7.2 Future Work

Due to the implementation problem mentioned in Chapter 4, a proxy is introduced. Hence the number of the test configurations is doubled to 6×2. To fix the problem and reach a minimum test configurations, the PIPs should be coded using XDL to generate a completely routed ring oscillator in each CLB. In that case the whole ring oscillator array can be generated as a large hard macro instead of instantiating independent ring oscillator hard macros in the selection wrapper.

The method can also be extended for characterizing latches in CLBs by a small modification. Also, adapting the ring oscillator design for the advanced Xilinx 7 series FPGA and applying the delay characterization method for online measurement would be an interesting topic for future work.

Finally, the runtime system of the reconfigurable system needs to be extended so that the delay information is exploited during instantiation of hardware accelerators for optimal performance and reliability.

Bibliography

[1] *Virtex-5 FPGA User Guide (UG190)*, 5th ed., Xilinx, March 2012.

[2] *Stratix V Device Overview*, 13th ed., Altera, May 2013.

[3] D. Koch, *Partial Reconfiguration on FPGAs: Architectures, Tools and Applications.* Springer, 2013, vol. 153.

[4] E. Lubbers and M. Platzner, "Cooperative multithreading in dynamically reconfigurable systems," in *International Conference on Field Programmable Logic and Applications.* IEEE, 2009, pp. 551–554.

[5] S. Mahapatra, V. Rao, B. Cheng, M. Khare, C. Parikh, J. C. S. Woo, and J. Vasi, "Performance and hot-carrier reliability of 100 nm channel length jet vapor deposited Si3N4 MNSFETs," *IEEE Transactions on Electron Devices*, vol. 48, no. 4, pp. 679–684, 2001.

[6] J. Stathis, "Reliability limits for the gate insulator in cmos technology," *IBM Journal of Research and Development*, vol. 46, no. 2.3, pp. 265–286, 2002.

[7] J. R. Black, "Electromigration—a brief survey and some recent results," *IEEE Transactions on Electron Devices*, vol. 16, no. 4, pp. 338–347, 1969.

[8] J. Massey, "NBTI: what we know and what we need to know - a tutorial addressing the current understanding and challenges for the future," in *IEEE International Integrated Reliability Workshop Final Report*, 2004, pp. 199–211.

[9] S. Srinivasan, P. Mangalagiri, Y. Xie, N. Vijaykrishnan, and K. Sarpatwari, "Flaw: Fpga lifetime awareness," in *Proceedings of the 43rd annual Design Automation Conference*, ser. DAC '06. New York, NY, USA: ACM, 2006, pp. 630–635. [Online]. Available: http://doi.acm.org/10.1145/1146909.1147070

[10] F. N. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, pp. 310–323, 1993.

[11] M. Abdelfattah, "Evaluation of advanced techniques for structural FPGA self-test," Master's thesis, Nr.3161, University of Stuttgart, 2011.

[12] M. Abdelfattah, L. Bauer, C. Braun, M. Imhof, M. Kochte, H. Zhang, J. Henkel, and H. Wunderlich, "Transparent structural online test for reconfigurable systems," in *IEEE 18th International On-Line Testing Symposium (IOLTS)*, 2012, pp. 37–42.

[13] K. Katsuki, M. Kotani, K. Kobayashi, and H. Onodera, "A yield and speed enhancement scheme under within-die variations on 90nm LUT array," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2005, pp. 601–604.

[14] S. K. Morrison, A. K. Percey, J. D. Logue, J. M. Simkins, and N. J. Sawyer, "Method and apparatus for clock signal performance measurement," Patent, 01 2006, uS 6983394. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_6983394/en/

[15] L. Cheng, J. Xiong, L. He, and M. Hutton, "FPGA performance optimization via chip-wise placement considering process variations," in *International Conference on Field Programmable Logic and Applications*, 2006, pp. 1–6.

[16] P. Manet, D. Maufroid, L. Tosi, G. Gailliard, O. Mulertt, M. Di Ciano, J.-D. Legat, D. Aulagnier, C. Gamrat, R. Liberati *et al.*, "An evaluation of dynamic partial reconfiguration for signal and image processing in professional electronics applications," *EURASIP Journal on Embedded Systems*, pp. 1–11, 2008.

[17] E. J. McDonald, "Runtime FPGA partial reconfiguration," in *IEEE Aerospace Conference*, 2008, pp. 1–7.

[18] T. Lenart, "Design of reconfigurable hardware architectures for real-time applications," Ph.D. dissertation, Lund University, Sweden, 2008.

[19] J. Meyer, J. Noguera, M. Huebner, L. Braun, O. Sander, R. M. Gil, R. Stewart, and J. Becker, "Fast start-up for spartan-6 FPGAs using dynamic partial reconfiguration," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2011, pp. 1–6.

[20] S. C. C. Tao Pi and S. J. C. Patrick J. Crotty, "FPGA lookup table with transmission gate structure for reliable low-voltage operation," Patent, 12 2003, uS 6667635. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_6667635/en/

[21] S. Trimberger, *Field programmable gate array technology.* Springer, 1994.

[22] U. Farooq, Z. Marrakchi, and H. Mehrez, *Tree-Based Heterogeneous FPGA Architectures.* Springer, 2012.

[23] M. Bushnell and V. D. Agrawal, *Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits.* Springer, 2000.

[24] J. Wong, P. Sedcole, and P. Y. K. Cheung, "Self-characterization of combinatorial circuit delays in FPGAs," in *International Conference on Field-Programmable Technology*, 2007, pp. 17–23.

[25] J. Li and J. Lach, "Negative-skewed shadow registers for at-speed delay variation characterization," in *25th International Conference on Computer Design*, 2007, pp. 354–359.

[26] M. Happe, H. Hangmann, A. Agne, and C. Plessl, “Eight ways to put your FPGA on fire - a systematic study of heat generators,” in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, 2012, pp. 1–6.

[27] D. Sheldon, R. Roosta, M. Sadigursky, and A. Farrokhy, “Monitoring temperature in SRAM-based FPGAs using a ring-oscillator design,” *Military and Aerospace FPGA and Applications (MAFA) Meeting*, 2007.

[28] K. M. Zick and J. P. Hayes, “Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, no. 1, pp. 1:1–1:26, Mar. 2012. [Online]. Available: http://doi.acm.org/10.1145/2133352.2133353

[29] C. Ruething, A. Agne, M. Happe, and C. Plessl, “Exploration of ring oscillator design space for temperature measurements on FPGAs,” in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 559–562.

[30] M. Bhushan, A. Gattiker, M. B. Ketchen, and K. K. Das, “Ring oscillators for CMOS process tuning and variability control,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 19, no. 1, pp. 10–18, 2006.

[31] H. Masuda, S.-i. Ohkawa, A. Kurokawa, and M. Aoki, “Challenge: Variability characterization and modeling for 65-to 90-nm processes,” in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2005, pp. 593–599.

[32] M. Nourani and A. Radhakrishnan, “Testing on-die process variation in nanometer VLSI,” *IEEE Design & Test of Computers*, vol. 23, no. 6, pp. 438–451, 2006.

[33] Z. Abuhamdeh, B. Hannagan, J. Remmers, and A. Crouch, “A production IR-Drop screen on a chip,” *IEEE Design & Test of Computers*, vol. 24, no. 3, pp. 216–224, 2007.

[34] B. P. Das, B. Amrutur, H. Jamadagni, N. Arvind, and V. Visvanathan, “Within-die gate delay variability measurement using reconfigurable ring oscillator,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 22, no. 2, pp. 256–267, 2009.

[35] M. Ruffoni and A. Bogliolo, “Direct measures of path delays on commercial FPGA chips,” in *6th IEEE Workshop on Signal Propagation on Interconnects*, 2002, pp. 157–159.

[36] H. Yu, Q. Xu, and P.-W. Leong, “Fine-grained characterization of process variation in FPGAs,” in *International Conference on Field-Programmable Technology (FPT)*, 2010, pp. 138–145.

[37] X. Li, F. Wang, T. La, and Z. Ling, “FPGA as Process Monitor-an effective method to characterize poly gate CD variation and its impact on product performance and yield,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 17, no. 3, pp. 267–272, 2004.

[38] P. Sedcole and P. Y. K. Cheung, “Within-die delay variability in 90nm FPGAs and beyond,” in *IEEE International Conference on Field Programmable Technology*, 2006, pp. 97–104.

[39] K. M. Zick and J. P. Hayes, “On-line sensing for healthier FPGA systems,” in *Proceedings of the 18th annual ACM/SIGDA International Symposium on Field programmable gate arrays*, ser. FPGA ’10, New York, NY, USA, 2010, pp. 239–248. [Online]. Available: http://doi.acm.org/10.1145/1723112.1723153

[40] *ML505/ML506/ML507 Evaluation Platform User Guide (UG347)*, 3rd ed., Xilinx, May 2011.

[41] *Virtex-5 FPGA System Monitor*, 1st ed., Xilinx, February 2011.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature