University of New Mexico
UNM Digital Repository

Electrical and Computer Engineering ETDs | Engineering ETDs

9-12-2014

# Within-Die Delay Variation Measurement And Analysis For Emerging Technologies Using An Embedded Test Structure

fareena saqib

Follow this and additional works at: https://digitalrepository.unm.edu/ece_etds

Recommended Citation
saqib, fareena. "Within-Die Delay Variation Measurement And Analysis For Emerging Technologies Using An Embedded Test Structure." (2014). https://digitalrepository.unm.edu/ece_etds/225

This Dissertation is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Electrical and Computer Engineering ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact disc@unm.edu.

Fareena Saqib
*Candidate*

Electrical & Computer Engineering
*Department*

This dissertation is approved, and it is acceptable in quality and form for publication:

*Approved by the Dissertation Committee:*

Dr. James F. Plusquellic , *Chairperson*

Dr. Nasir Ghani

Dr. Payman Zarkesh-Ha

Dr. Charles Lamech

# Within-Die Delay Variation Measurement And Analysis For Emerging Technologies Using An Embedded Test Structure

**by**

**Fareena Saqib**

B.I.T., National University of Sciences and Technology (NUST), 2006

M.S., Electrical and Computer Engineering, University of New Mexico (UNM), 2010

DISSERTATION

Submitted in Partial Fulfillment of the

Requirements for the Degree of

**Doctor of Philosophy**

**Engineering**

The University of New Mexico

Albuquerque, New Mexico

**July, 2014**

**©2014, Fareena Saqib**

# Dedication

This dissertation is dedicated to my family, for all the love, support, and the many sacrifices made.

# Acknowledgments

First of all I wish to offer my gratitude to Almighty God for the Blessings bestowed upon me.

I would like to express my sincerest gratitude and indebtedness to my advisor and supervisor Dr. Jim Plusquellic for his untiring assistance, timely guidance, encouragement and creativity at every stage of this study. Working with him has been a true privilege, and I have benefited tremendously from his knowledge of science and engineering, both in depth and broadness, and his enthusiasm in supporting me to carry out and complete the research work. He is a role model that I can always look up to.

My deepest appreciation and thanks are extended to Dr. Nasir Ghani, Dr. Charles Lamech, and Dr. Payman Zarkesh-Ha for their continuous support, encouragement and valuable suggestions. I am also grateful to all the group members in the Electrical and Computer Engineering Department at the University of New Mexico for their assistance, motivation and valuable comments to improve the dissertation. I am especially grateful to Jim Aarestad for his dedicated assistance, guidance and professional contributions.

# Within-Die Delay Variation Measurement And Analysis For Emerging Technologies Using An Embedded Test Structure

**by**

**Fareena Saqib**

**B.I.T., National University of Sciences and Technology (NUST), 2006**

**M.S., Electrical and Computer Engineering, University of New Mexico (UNM), 2010**

**Ph.D., Engineering, University of New Mexico (UNM), 2014**

## ABSTRACT

Both random and systematic within-die process variations (PV) are growing more severe with shrinking geometries and increasing die size. Escalation in the variations in delay and power with reductions in feature size places higher demands on the accuracy of variation models. Their availability can be used to improve yield, and the corresponding profitability and product quality of the fabricated integrated circuits (ICs). Sources of within-die variations include optical source limitations, and layout-based systematic effects (pitch, line-width variability, and microscopic etch loading). Unfortunately, accurate models of within-die PVs are becoming more difficult to derive because of their increasingly sensitivity to design-context. Embedded test structures (ETS) continue to

play an important role in the development of models of PVs and as a mechanism to improve correlations between hardware and models. Variations in path delays are increasing with scaling, and are increasingly affected by “neighborhood” interactions. In order to fully characterize within-die variations, delays must be measured in the context of actual core-logic macros. Doing so requires the use of an embedded test structure, as opposed to traditional scribe line test structures such as ring oscillators (RO). Accurate measurements of within-die variations can be used, e.g., to better tune models to actual hardware (model-to-hardware correlations).

In this research project, I propose an embedded test structure called REBEL (Regional dELay BEhavior) that is designed to measure path delays in a minimally invasive fashion; and its architecture measures the path delays more accurately. Design for manufacture-ability (DFM) analysis is done on the on 90 nm ASIC chips and 28nm Zynq 7000 series FPGA boards. I present ASIC results on within-die path delay variations in a floating-point unit (FPU) fabricated in IBM’s 90 nm technology, with 5 pipeline stages, used as a test vehicle in chip experiments carried out at nine different temperature/voltage (TV) corners. Also experimental data has been analyzed for path delay variations in short vs long paths. FPGA results on within-die variation and die-to-die variations on Advanced Encryption System (AES) using single pipelined stage are also presented. Other analysis that have been performed on the calibrated path delays are Flip Flop propagation delays for both rising and falling edge (tpHL and tpLH), uncertainty analysis, path distribution analysis, short versus long path variations and mid-length path within-die variation. I also analyze the impact on delay when the chips are subjected to industrial-level temperature and voltage variations. From the experimental

results, it has been established that the proposed REBEL provides capabilities similar to an off-chip logic analyzer, i.e., it is able to capture the temporal behavior of the signal over time, including any static and dynamic hazards that may occur on the tested path. The ASIC results further show that path delays are correlated to the launch-capture (LC) interval used to time them. Therefore, calibration as proposed in this work must be carried out in order to obtain an accurate analysis of within-die variations. Results on ASIC chips show that short paths can vary up to 35% on average, while long paths vary up to 20% at nominal temperature and voltage. A similar trend occurs for within-die variations of mid-length paths where magnitudes reduced to 20% and 5%, respectively. The magnitude of delay variations in both these analyses increase as temperature and voltage are changed to increase performance. The high level of within-die delay variations are undesirable from a design perspective, but they represent a rich source of entropy for applications that make use of "secrets" such as authentication, hardware metering and encryption. Physical unclonable functions (PUFs) are a class of primitives that leverage within-die-variations as a means of generating random bit strings for these types of applications, including hardware security and trust.

Zynq FPGAs Die-to-Die and within-die variation study shows that on average there is 5% of within-Die variation and the range of die-to-Die variation can go upto 3ns. The die-to-Die variations can be explored in much further detail to study the variations spatial dependance.

Additionally, I also carried out research in the area data mining to cater for big data by focusing the work on decision tree classification (DTC) to speed-up the classification step in hardware implementation. For this purpose, I devised a pipelined

architecture for the implementation of axis parallel binary decision tree classification for meeting up with the requirements of execution time and minimal resource usage in terms of area. The motivation for this work is that analyzing larger data-sets have created abundant opportunities for algorithmic and architectural developments, and data-mining innovations, thus creating a great demand for faster execution of these algorithms, leading towards improving execution time and resource utilization. Decision trees (DT) have since been implemented in software programs. Though, the software implementation of DTC is highly accurate, the execution times and the resource utilization still require improvement to meet the computational demands in the ever growing industry. On the other hand, hardware implementation of DT has not been thoroughly investigated or reported in detail. Therefore, I propose a hardware acceleration of pipelined architecture that incorporates the parallel approach in acquiring the data by having parallel engines working on different partitions of data independently. Also, each engine is processing the data in a pipelined fashion to utilize the resources more efficiently and reduce the time for processing all the data records/tuples. Experimental results show that our proposed hardware acceleration of classification algorithms has increased throughput, by reducing the number of clock cycles required to process the data and generate the results, and it requires minimal resources hence it is area efficient. This architecture also enables algorithms to scale with increasingly large and complex data sets. We developed the DTC algorithm in detail and explored techniques for adapting it to a hardware implementation successfully. This system is 3.5 times faster than the existing hardware implementation of classification.

# Table of Contents

LIST OF FIGURES....XIV

LIST OF TABLES....XVIII

1 INTRODUCTION....1

1.1 Sources of Within-Die Variations....2

1.2 Proposed Embedded Test Structure (ETS)....2

1.3 Decision Tree Classification (DTC) ....3

1.4 Organization....4

2 BACKGROUND....5

2.1 Test Structures for Within-Die Delay Measurement....5

2.1.1 Delay Measurements of Individual Gates....8

2.2 Design and Environment Effects on Within-Die Variations....9

2.3 Within-Die Variations in FPGAs Reconfigurable Logic....10

2.4 Impact of Within-Die Spatial Process Variations ....11

2.4.1 Clock Frequency....11

2.5 Static Timing Analysis....12

2.6 Design for Manufacturing and Yield ....12

2.7 Applications of Die-to-Die and Within-Die Variations....14

2.7.1 PUF Hardware Security....14

**2.8 Summary....20**

**3 REBEL:EMBEDDED TEST STRUCTURE AND MACROS-UNDER-TEST .....22**

**3.1 REBEL-Regional Delay Behavior....22**

**3.2 Floating Point Unit (FPU Macro)....28**

**3.3 Advance Encryption Standard (AES) ....31**

**3.3.1 Step1: Sub-Bytes....33**

**3.3.2 Step2: Shift Rows....33**

**3.3.3 Step 3:Mixed Columns....34**

**3.3.4 Step 4: Add Round Key....34**

**3.4 Chip Layout, with REBEL, AES and FPU Placement....35**

**3.4.1 ASIC Layout with REBEL and FPU....35**

**3.4.2 FPGA Layout with REBEL and AES Round....37**

**4 EXPERIMENTAL SETUP....38**

**4.1 Experimental Setup for IC 90nm Chips....38**

**4.2 Launch-Capture Clocking Sequence and Clock Strobing....39**

**4.2.1 Launch-Capture Clocking Sequence....39**

**4.2.2 Clock Strobing....39**

**4.3 Delay Measurement Process....40**

**4.4 Digital Snapshots and FF Delays ....43**

**4.5 Calibration and Power Rail Voltage Transient Effects ....44**

**4.6 Measuring and Calibrating Path Delays....48**

**4.7 Experimental Setup for FPGA Boards....52**

**4.8 Launch-Capture Clocking Sequence and Clock Strobing....53**

**4.8.1 Launch-capture Clocking Sequence......................................................53**

**4.8.2 Clock Strobing.........................................................................................54**

**4.9 Delay Measurement Process....................................................................54**

**4.10 Measuring and Calibrating Path Delays..................................................56**

**5 ASIC EXPERIMENTAL RESULTS AND ANALYSIS..........................................58**

**5.1 Flip-flops Analysis...................................................................................58**

**5.2 Uncertainty Analysis...............................................................................62**

**5.3 Error in Estimating Path Delays.............................................................64**

**5.4 Path Distribution Analysis......................................................................67**

**5.5 Short vs. Long Path Variation.................................................................69**

**5.6 Within-Die Delay Variation Analysis.......................................................71**

**6 FPGA EXPERIMENTAL RESULTS AND ANALYSIS..........................................75**

**6.1 Flip Flop Analysis.....................................................................................75**

**6.1.1 Propagational delay tpHL and tpLH Analysis...........................................75**

6.1.1.1 Rising Edge Propagation Delay (tpLH):................................................76

6.1.1.2 Falling Edge Propagation Delay (tpHL):...............................................77

6.1.1.3 Comparative Analysis:.........................................................................78

**6.1.2 Average Rising Edge Flip-flops Delays.......................................................79**

**6.1.3 Average Falling Edge Flip-flops Delays.....................................................80**

**6.2 Sample analysis.........................................................................................80**

**6.3 Uncertainty Analysis ................................................................................85**

**6.4 Die-to-Die Variation in Flip-flops for Rising Edge in All Chips.......................86**

**6.5 Die-to-Die Variation in Flip-flops for Falling Edge in All Chips.......................89**

**6.6 Path Distribution Analysis......91**

**6.7 Die-to-Die Delay Variation Analysis ......94**

**6.8 Within-Die Delay Variation Analysis ......96**

**7 PIPELINED DECISION TREE IMPLEMENTATION......100**

**7.1 Introduction......101**

**7.2 Background......104**

**7.3 Decision Tree Classification Architecture......106**

**7.4 Experimental Results......115**

**7.4.1 Accuracy Of The Model......115**

**7.4.2 Comparison with Software Implementations......116**

**7.4.3 Comparison with Previous Hardware Implementation......118**

**7.5 Resource Utilization......119**

**7.6 Data Streaming with High Performance Communication Link......120**

**8 CONCLUSION......123**

**9 FUTURE WORK......127**

**9.1 Path Delay Measurement using TDCs......127**

**9.2 Path Delay Measurement as an Entropy Source for PUF Primitive......129**

**9.3 Defect Analysis......131**

**9.4 Model to Hardware Correlation......131**

**9.5 On-chip Clock using DLLs or PLLs and Bypass Capacitance on I/OPads ......132**

**REFERENCES......133**

# List of Figures

Fig. 3.1: REBEL Integration Strategy ....................24

Fig. 3.2: REBEL Row Control Logic....................25

Fig. 3.3(a): Modified Clocked-LSSD Scan FF ....................27

Fig. 3.3(b): Additional ‘Front-End’ Logic ....................27

Fig. 3.4: Floating Point Unit Block Diagram ....................28

Fig. 3.5: AES Engine....................31

Fig. 3.6: Mux-D Flip Flop....................32

Fig. 3.7: AES Round....................33

Fig. 3.8: ASIC Layout....................35

Fig. 3.9: ASIC Design Flow....................36

Fig. 3.10: FPGA Layout....................37

Fig. 4.1: Experimental Setup . ....................39

Fig. 4.2: REBEL Launch-Capture (LC) Test Sequence. Clock strobing applies a sequence of LCIs of different widths . ....................40

Fig. 4.3: Oscilloscope measured launch-capture intervals (LCIs) for each of the Fine Phase Adjust (FPA) values on the FPGA. Row Header delay of 300 ps is included ....................42

Fig. 4.4: Illustration of a partial sequence of digital snapshots (21 FPAs of the 159)

produced from a path delay test ....................................................................44

Fig. 4.5: Curves of individual FF delays measured at different midpoint delays. An average of all individual curves is superimposed, as well as a curve with only the low frequency components of the 'Average delay curve' ..........................48

Fig. 4.6: Calculating path delay using digital snapshots. .................................................49

Fig. 4.7: Illustration of calibration operation carried out using an individual FF delay curve and 'smoothed' average delay curve .....................................................50

Fig. 4.8: Delay calibration applied to example path from Fig. 4.6 using window1 ($W_1$) .51

Fig. 4.9: Zed Board with Zynq FPGA..................................................................................53

Fig 4.10: Clock Strobing in FPGAImplementation..........................................................54

Fig 4.11: Digital Snapshot from a path delay test............................................................55

Fig 4.12: REBEL Integration with AES Engine...............................................................56

Fig 4.13: Block level Diagram of AES with REBEL Integration.....................................57

Fig.5.1: Flip-Flop Delays for Falling and Rising Transitions...........................................59

Fig.5.2: Mid-point LCI Average Delays for Rising and Falling Transitions.....................60

Fig.5.3: Propagational Delay of Rising Edge...................................................................61

Fig.5.4: Propagational Delay of Falling Edge..................................................................61

Fig. 5.5: Digital snapshots illustrating uncertainty using data given earlier in Fig. 4.4….62

Fig. 5.6: Average Uncertainty as a function of distance (in FFs) from the insertion point for 4 chips at all TV corners...............................................................................63

Fig. 5.7: Calibration process applied to the 2nd of two consecutive windows illustrating error in the estimation of path delay ..............................................................64

Fig. 5.8: Delay errors computed using $W_1$ and $W_2$ of the proposed calibration method

for all paths and chips at 25ºC, 1.2V................................................................65

Fig. 5.9: 3σ (a) and average (b) delay errors for all paths and chips at 25ºC, 1.20V as a function of ‘length of the delay chain’ (x-axis) and span (y-axis)....................66

Fig. 5.10: Stable path distributions for CHIP1 at 25 ºC, 1.20V......................................68

Fig. 5.11: Short vs. long path delay variation analysis at 25ºC, 1.20V.............................70

Fig. 5.12: Within-die delay variation analysis using regression: example scatter plots from distribution of common to all chips.........................................................71

Fig. 5.13: Within-die variation analysis using regression. Average path delay vs. normalized 3σ of residuals expressed as percentage change ...........................73

Fig. 5.14: Within-die variation analysis using regression using data from two additional TV corners .........................................................................................74

Fig. 6.1: Rising Edge Propagational Delay....................................................................76

Fig. 6.2: Falling Edge Propagational Delay....................................................................77

Fig. 6.3: Rise versus Fall delay of Flip-Flops.................................................................78

Fig. 6.4: Average Rising delays of FFs for Chip1..........................................................79

Fig. 6.5: Average Falling delays of FFs for Chip1.........................................................80

Fig. 6.6: Delay versus Sample Analysis for Chip 1........................................................81

Fig. 6.7: Delay versus Sample Analysis for Chip 5........................................................82

Fig. 6.8: Uncertainty Analysis for Chip 5.......................................................................83

Fig. 6.9: Delay Versus Sample Analysis for Chip 7.......................................................83

Fig. 6.10: Uncertainty Analysis for Chip 7.......................................................................84

Fig. 6.11: Overall Uncertainty in path delays...................................................................85

Fig. 6.12: Standard Deviation in Rising Edge for All Chips............................................87

Fig. 6.13: Percentage Change in Rising Edge for All Chips .......................................88
Fig. 6.14: Die-to-die Variation in Falling Edge in All Chips............................................89
Fig. 6.15: Percentage Change in Falling Edge for All Chips............................................90
Fig. 6.16: Path Distribution for Chip 1..................................................................91
Fig. 6.17: Stable Path Distribution for Chip 2...........................................................92
Fig. 6.18: Unique and Common Path ID Counter.........................................................93
Fig. 6.19: Die-to-die Variation Analysis using Regression. Average path delay versus normalized 3σ of residuals...........................................................................94
Fig. 6.20: Overall percentage change of die-to-die variation in path delay......................96
Fig. 6.21: Within-die variation for short and long path pairings......................................97
Fig. 6.22: Within-die delay variation analysis using regression: Example scatter plot form distributions of common to all chips................................................................98
Fig. 6.23: Overall percentage change of path delays variation........................................99
Fig. 7.1: Decision rules in form of decision trees........................................................106
Fig. 7.2: Decision tree stages ..........................................................................107
Fig. 7.3: Decision tree classification system............................................................109
Fig. 7.4: RTL level block diagram of hardware module................................................110
Fig. 7.5: Parallel and pipelined decision tree Engine...................................................112
Fig. 7.6: Streaming architecture.......................................................................121
Fig. 9.1: Time to Digital Conversion (TDC)..............................................................128

# List of Tables

Table 3.1: Configuration States for Row Control Logic....................................................26

Table 7.1: Accurcy of decision tree model.......................................................................116

Table 7.2: Comparison with software implementation....................................................117

Table 7.3: Comparison with Hardware implementations.................................................119

Table 7.4: The resource utilization of the decision tree model........................................120

# CHAPTER 1
# Introduction

Both random and systematic within-die process variations (PV) are growing more severe with shrinking geometries and increasing die size [1] [2] [3]. Embedded test structures (ETS) continue to play an important role in the development of models of PVs and as a mechanism to improve correlations between hardware and models. Variations in delay and power continue to increase with reductions in feature size, which places higher demands on the accuracy of variation models. Their availability can be used to improve yield, and the corresponding profitability and product quality of the fabricated ICs [4].

Decision tree classification (DTC) is a widely used technique in data mining algorithms known for its high accuracy in forecasting. As technology has progressed, and available storage capacity in modern computers increased, the amount of data available to be processed has also increased substantially, resulting in much slower induction and classification times. Many parallel implementations of decision tree classifications algorithms have addressed the issues of reliability and accuracy in the induction process. In the classification process, larger amounts of data require proportionately more execution time, thus hindering the performance of legacy systems. Hence, to cater for big data for data mining, further work on decision tree classification (DTC) to speed-up the

classification step in hardware implementation cannot be overemphasized.

## 1.1 Sources of Within-Die Variations

Sources of within-die variations include optical source limitations, and layout-based systematic effects (pitch, line-width variability, and microscopic etch loading [5] [6] [7]. Unfortunately, accurate models of within-die PVs are becoming more difficult to derive because of their increasingly sensitivity to design-context. Stand-alone embedded test structures such as ring-oscillators (ROs) are becoming less effective for characterizing delay variations in actual product macros because they are typically placed around the layout region of the macro as opposed to being integrated into it. In such circumstances, ring oscillators (ROs) are not exposed to, e.g., the same types of distortions which are introduced by photo-lithography interference patterns [8]. Some of the proposed ETS, such as those that measure delay characteristics of the macro itself [9] [10], offer the best solution, but are difficult to integrate without having an adverse impact on area overhead, yield loss, performance, I/O interface, test cost, etc. of the product design.

## 1.2 Proposed Embedded Test Structure (ETS)

For accurate measurement of within-die variations, we propose and investigate an embedded test structure (ETS), called REBEL (Regional dELay BEhavior), which is designed to measure path delays in macros while minimizing the adverse effects on area overhead, yield loss, performance, etc. The proposed ETS is designed to serve applications such as model-to-hardware correlation [11], detection of hardware Trojans [12], design debug processes, detection of small delay defects [13], and physical unclonable functions [14]. Each of these areas requires accurate measurements of path

delays and/or the ability to differentiate at high resolutions between delays of neighboring paths. The REBEL ETS leverages the scan chain architecture to measure delay variations. In particular, it uses a special configuration of flush delay mode that is available in level sensitive scan design (LSSD) style scan chains. In previous work [15], the promise of capturing regional delay variations using a special launch-capture timing sequence applied while in flush delay mode has been demonstrated. We extended this technique here by allowing output signals from a design macro to be inserted into the flush delay chain for path delay measurements.

A key feature of our work is the evaluation of REBEL in multiple copies of a custom designed test chip fabricated in IBM's 90nm technology. The macro in which REBEL is integrated is an IEEE-754 compliant floating point unit (FPU), with 5 pipeline stages. Random test patterns are applied to the combinational logic within each of the pipeline stages and the measured delays are analyzed, with emphasis on evaluating the magnitude of within-die variations as a function of path length. A second important component of our experiments is the evaluation of delay variations while the chips are subjected to industrial-level temperature and voltage (TV) variations.

## 1.3 Decision Tree Classification (DTC)

The process of converting unidentified or unprocessed data into actionable information that is important and valuable to the user is known as data mining [16]. Recent advances in technology and ever increasing demands for analyzing larger datasets have created abundant opportunities for algorithmic and architectural development and innovations. Hence data mining algorithms have become increasingly significant and complex. Similarly there is a great demand for faster execution of these algorithms,

leading to efforts to improve execution time and resource utilization.

Decision Tree Classification (DTC) is a widely used technique in data mining algorithms known for its high accuracy in forecasting. As technology has progressed and available storage capacity in modern computers increased, the amount of data available to be processed has also increased substantially, resulting in much slower induction and classification times. Many parallel implementations of decision tree classification algorithms have addressed the issues of reliability and accuracy in the induction process. In the classification process, larger amounts of data require proportionately more execution time, thus hindering the performance of legacy systems. We have devised a pipelined architecture for the implementation of axis parallel binary decision tree classification that dramatically improves the execution time of the algorithm while consuming minimal resources in terms of area.

## 1.4 Organization

The balance of this dissertation is organized as follows: chapter 2 discusses related work and background on process variation. chapter 3 describes the details of the REBEL ETS while chapter 4 describes the integration of REBEL into the FPU and AES (Experimental setup). In chapter 5, I present the results of hardware experiments, chapter 6 presents the results of experiments on FPGAs, Chapter 7 discusses the implementation of proposed architecture of parallel implementation of pipelined decision tree classification engine. Chapter 8 summarizes the work and contains conclusion. Chapter 9 encompasses suggestions for the future work.

# CHAPTER 2
# Background

Process variation (PV) is a challenge of Integrated circuit design in the newer technologies with the shrinking node sizes. Variation can be defined in terms of die-to-die and within-die variation, where die-to-die variations can be due to change in temperature or processing conditions. These changes can be captured using scribe line test structures or small amount of embedded test structures are sufficient for die-to-die variations coverage.

Within-die variations on the other hand are caused by across-field effects [17] in compassing the layout design techniques, optical aberration and other random effects such as dopant fluctuations. Within-die characterization requires more density of test structures to capture the variations in all regions.

## 2.1 Test Structures for Within-Die Delay Measurement

Monitoring die-to-die and within-die requires distributing embedded test structures across and within chips in order to capture the overall wafer-to-wafer and within chip systematic variations with spatial correlations. Within-die and die-to-die delay analysis of delay variations continues to be an active research area. Ring oscillators

(RO) based test structures have been successfully used to characterize within-die delay variations in ASICs and FPGAs [9] [10] [17] [18] [19] [20] [21].

The simplicity of the RO design makes it an attractive modality as an ETS. Tuan et al. [17] proposed a scheme of creating an RO from the path-under-test to measure the critical path delays. Das et al. [18] constructed a digitally re-configurable RO structure for measuring the gate-level delays and reported measurement accuracy of 1 ps and up to 26% within-die delay variation in 65nm technology.

Bhushan et al. [19] presented an RO-based variability measurement scheme and illustrated the experimental results from a 90nm technology node. RO based designs are also used to measure within-die variability in 90nm and 65nm FPGAs [17] [20] [21]. Several on-chip analog measurement systems are proposed by Kinniment et al. [22] to accurately measure the on-chip path timing differences with a resolution of 10ps. They explain the time measurement techniques utilizing parallel mutex with a tapped delay line, successive approximation method and amplification of small time differences to a measurable size in different proposed time measurement approaches. A measurement system is proposed to characterize individual gate delays using an on-chip sampling oscilloscope. Also a within-die variation characterization system is proposed by Zhang et al. which also uses an on-chip sampling oscilloscope [23]. Stand-alone RO-based delay measurements lack the ability to account for "circuit context". Macro embedded RO schemes, such as Path RO [10], can only be applied to hazard free and robust paths. These kinds of embedded test structures, where a set of ring oscillators is distributed cross the layout, are capable of capturing within-die variations, but are becoming increasingly less accurate as predictors of delay variations in actual product macros.

Various Time-to-Digital converters (TDCs) have been proposed for on-chip delay measurements with resolution as high as 5 ps and with low thermal sensitivity [11] [24] [25] [26]. Tapped delay lines are used by Dudek et al. [24] TDC using a technique vernier delay line with the a read-out circuitry, and achieved a 30 ps resolution with 128 delay stages and showed up to 5 ps is reachable using this methodology. Similarly, Datta et al. measured path delay by Modified Vernier Delay Line (MVDL), which digitizes the path delay forming MVDL and improves the resolution of delays [27]. However, to achieve the high accuracy measurement is sensitive to the symmetric routing in branches of MVDL, which will constrain the place and route tool [28]. Another TDC is proposed and designed with a resolution of 10ps [11], it measures the delay difference between two path signals out of several paths which are fanned out to the TDC from the macro-under-test. The delay between the two transitions produces a negative-going pulse with a width proportional to that difference of the transition timing of the glitch free paths. The TDC scheme, just like RO restricts their coverage because of condition of having glitch free paths. Also analog measurement systems and TDCs have large area overheads.

Truly embedded test structures, such as those that measure delay [9] [10] [15], and power [29] characteristics of the macro itself, offer the best solution. Path-RO creates an oscillator from the given path to measure the delay on chip, capturing the process variations [10] but this solution is difficult to integrate without having an adverse impact on area overhead, and test cost, etc. of the product design. Whereas another novel technique proposed by Acharyya et al. [29] that leverages the existing power control circuitry, added to reduce the power consumption, and measures the leakage current variation of these modules and this variation thus reflects the current variation across the

chip. The use of multiple power supply port measurement technique is incorporated to measure the within-die leakage current.

A LSSD-style scan-chain-based embedded test structure is proposed by Lamech et al. [11] and Aarestad et al. [15], utilizing the inherent flush delay mode for obtaining single-shot measurements of path delays in product macros induced by the history effect. This test structure leverages the existing scan structure and can be used to characterize the within-die delay variations by measuring the path delays of the macros. The embedded test scheme is called REBEL, a detailed description and analysis is performed in chapter 3. This proposed structure uses at-speed clock to measure the path delays for both short and long paths. The advantage of using the at-speed clock over the faster-than-at-speed clock is that the delays are more realistic and the supply voltage transience does not deteriorate the measured delay values and the path delay measurements are closer to the actual delays. I have used this technique for the experiments performed to examine the within-die variation of the 90nm test chips.

### 2.1.1. Delay Measurements of Individual Gates

Authors [18] proposed a modified re-configurable ring oscillator to measure individual gate delay. The delay was averaged for the falling and the rising transition and showed up-to 26% within-die variation among the identical inverters which were placed close by on the chip. Symmetric multiplexers of large size having balanced delay and a set of inverters were used in the gate delay measurement cell, where cell was replicated in five stages for the delay measurement mechanism. A Pico-second Imaging Circuit Analysis was used to digitize the delay by counting the infra-red photons captured from the chip. The gate delay measurement of individual standard cells helped in

characterizing the stress, neighboring effect and other effects more efficiently.

Furthermore, in another measurement system for characterizing an on-chip within-die delay variations of standard cells, for both falling and rising edges, was proposed by Zhang et al. using sampling oscilloscope with a pico-second resolution [23] [30]. It displayed a strong correlation between the on-chip measurements and Monte-Carlo simulation. The experimental results confirmed that the delay variations of the on-chip are smaller for the gates with the bigger active area, and the NMOS has bigger variation than the PMOS.

## 2.2 Design and Environment Effects on Within-Die Variations

The environmental factors can affect the behavior of a semiconductor circuit, like the temperature or supply voltage for a given process. Also the device parameters, for example the length of transistor, oxide thickness may vary caused by the non-uniformity in the manufacturing process.

There are many sources which can affect the fabrication process and to create an adequate model for characterizing delay variation it is important to identify these parameters. The impact of different layout topologies on variation is presented by Pang et al. [31], the authors measured the variability of various test structures and analyzed the effects of systematic and random components of within-die variations. Lithographic simulation capability in the routing engine can improve the product yield [8] by identifying and modifying the patterns in the layout to avoid lithography hot spots. Reticle Enhancement technology is discussed by Grobman et al. [32] and usage of other optimized techniques to achieve planarity in manufacturing process for the sub-wavelength technologies for better circuit timings.

Analysis of temperature dependence has been performed on the sub-threshold circuit 40nm chip [33]. The device under test had a 16x16 array with each unit having 8 ring oscillators, where they are placed in different supply voltage regions and later the outputs are level shifted to the nominal voltage. The variation was measured as deviation of delay/mean and showed that the variation was 1.4 times more when the temperature was dropped from 25°C to -40°C [33]. This variation was inversely proportional to the temperature, that is, the with-die delay variations were larger when the temperature was lowered.

## 2.3 Within-Die Variations in FPGAs Reconfigurable Logic

In the domain of FPGAs re-configurable logic, techniques for the measurement of within-die variation is discussed by Sedcole et al. [21] in 90nm FPGAs and by Tuan et al. [17] in 65nm FPGAs. The effect of both random and systematic process variation on 18 Altera Cyclone II devices with 5 stage and 7 stage ROs was measured by Wang et al. [34].

For 90nm, the measure of process variation in 10 FPGA's using 135 stage ring oscillators is performed also the within-die delay variation characterization is performed with a small number of oscillator stages and shows that the mean random variation is ± 3.54% and systematic variation can vary the delay up to 3.66% additionally [21]. Whereas, for 65nm FPGA family a detailed analysis of the within-die variations is performed with the test structures implemented on the re-configurable logic, each created with four configurable logic blocks [17]. Using a large number of ring-oscillators, the analysis is performed on rising and falling edges independently and the data is processed to distinguish between the random and systematic variations. The results showed that the

random variation followed Gaussian distribution whereas the systematic variation was further modeled in the software timing models to evaluate the optimization of performance using a variation aware timing model, by calculating the maximum frequency for each design.

## 2.4 Impact of Within-Die Spatial Process Variations

### 2.4.1 Clock Frequency

A design with low overhead for calibration of maximum frequency has been proposed by Paul et al. [9]; where at the given voltage sensitivity, small set of paths is configured into a ring oscillator and the maximum frequency of a given chip is dynamically computed. Hence, the need of delay testing at operating voltages with all frequencies is eliminated through binning each chip into categories of different voltages and frequencies.

The product level variations for single and multi-core processors were simulated for maximum clock frequency and optimized throughput for 22nm technology [35]. They statistically measured the impact of parameter variations and compared the performance of multi-core processors with the single core processors and showed that multi-core processors were more variation tolerant because of the greater impact of memory latency and bandwidth on throughput.

Similarly possibilities of mitigating the performance loss was investigated by Palframan et al. [36], by introducing redundancy along the processor datapath in the form of one or more extra bit slices, leaving the dummy slices in the datapath unused to avoid excessively slow critical paths created by delay variation, which showed the reduction of delay penalty by 10% or more caused by the variation.

## 2.5 Static Timing Analysis

The design environments utilize the static timing analysis for the delay variations in the paths through the chip, by considering the worst case delay by assuming all the segment delays to be maximum or minimum, which are not realistic [8]. Each element of the path has associated delay and models should be incorporated for process variations to do the statistical static timing analysis, to be able to get better model-to-hardware correlations.

A technique for computing the delay distribution task as a function of technology parameter of a circuit was presented by Mehr et al. [37]. They calculated the mean value of delay using normal distribution approximation of normalized delay and skewness of the delay distribution from Gaussian distribution; and compared the results with those of Monte-Carlo simulations. Channel length variation was engrossed which impacted the threshold voltage and load capacitance; and thus affected the gate delay. The study incorporated the impact of transistor stacking on the delay, by using Taylor Series. Also, it combined the load capacitance parameter to get more accurate delays [37].

## 2.6 Design for Manufacturing and Yield

The design for manufacturing (DFM), and design for yield [4] classifies the systematic and statistical variations, caused by the physical defects in the structure of transistors which can be catastrophic, or electrical variations in the composition of transistors, wires or vias, causing parametric defect which allows the chip to function in a specified range with varied power leakage [8]. Catastrophic defects fails the chips, whereas, the chips with parametric defects can function but they do not meet the design requirements, for example they do not function on the specified range or there is more

power leakage than specified.

A need of a DFM aware design tool is vital for the newer technologies where the feature sizes are shrinking. A broad range of work is ongoing to develop mechanisms for characterizing more accurately the within-die and die-to- die process variations (PV). The development of area-efficient structures and methods for validating variation models for the newer technologies is of greater importance, where the feature sizes are getting smaller than the wavelength of the light used to create them [8]. The physical variations also contributes more variation on the scaling devices, as a larger impact is produced on a smaller device size and causes a huge variation as compared to the larger devices.

The physical device parameters determine the behavior of the device and predicts the performance changes with the environmental factors, by the variability in the fabrication process. These variations are captured with the conventional test structures, but the process variations are getting more sensitive to the design context, which the traditional test structures residing on a different region then the actual macro cannot truly capture.

The die-to-die level variations can be captured with the limited number of test structures for testing and measurement. The test structures currently being used include, scribe-line structures, or a fewer number of ring oscillators embedded test structure to capture the variations. This methodology is not effective for characterizing the within-die and context-sensitive variations. Process Variations are more challenging to measure and model in within-die context, which includes the across field effects. The main sources of these are due to optical source limitations, and layout-based systematic effects [17]. Also pitch, line-width variability, and microscopic etch loading are the sources of variations

from the manufacturing process [5] [6] [7].

## 2.7 Applications of Die-to-Die and Within-Die Variations

The process variations in the devices can be used for the identification [34] [35] [36] [37], authentication [38] [39] and generating unique keys for encryption, benefiting the nature of variations being random. The concept of using the manufacturing process variations as identifiers for integrated circuits is recent and is being used in physical unclonable functions (PUF). Where each device has its individual characteristics and it is impossible to have an exact duplicate, even if the same manufacturing process was used in the production. There are a large number of PUFs, which can be classified in many categories, e.g., Memory based, Delay based, and power grid PUFs which uses the process variation to produce unique identification mechanisms.

### 2.7.1. Hardware Security

Computing platforms are being increasingly deployed in many critical infrastructures such as smart grid, financial systems, sensitive governmental organizations *etc.,* where consequences of a successful security attack could be potentially serious. Thus, these applications of computing platforms in high-risk areas motivate the need to build platforms with enhanced security.

The computing platforms are multifaceted and generally comprise of architectures, operating systems and routine libraries. For normal operation, they require interaction between numerous hardware components such as processor, chipset, memory and peripherals. In order to maintain security of these computing platforms it is essentially required to ensure that there are no known security deeds present in the run-

time interaction between these hardware units exploitable by attackers. However, validating complete security of the computing platforms may be difficult and inflexible because of the fact that there are a large number of elements in the hardware units and their mutual interaction is influenced by many control signals [40]. Tempering, duplicating and theft of service have become a grave concern for hardware vendors as it may have adverse effects on their income and reputation. For providing protection against this menace, hardware security built on cryptographic primitives using keys can be used. These keys are usually stored somewhere in the hardware. Therefore, the strength of the security depends mainly on the effort required by attackers to compromise them. The attackers have developed very advanced tools for attacking hardware. This has reduced and minimized the protection provided by storing a key in memory.

Physically Unclonable Functions (PUFs) can also be used to protect devices against attacks on their keys. PUFs are primitives that extract secrets from physical characteristics of integrated circuits (ICs) and can be used, *inter alia*, for protected key storage [41]. A PUF is a function that is embodied in a physical structure that consists of many random uncontrollable components. These primitives are produced during manufacturing mainly because of process variations. Due to this random structure a physical inducement or challenge generates unpredictable responses. Because of their physical properties PUFs are nonreplicable and very promising primitives for the purpose of authentication and storage of cryptographic keys [42].

Random variations in physical properties of chips are used by PUFs to differentiate one chip from another, and are impossible to duplicate even by the

manufacturer. Although process variations are effectively impossible to control or eliminate, but they can be measured. The specific varying properties exploited by the PUF can differ from one PUF design to another. However, common sources of parametric variation include propagation delay, metal resistance, transistor drive strength, and mismatches between complementary transistors [14].

PUFs are used in a number of security applications like authentication, identification, and secure key generation. PUF implementations are evaluated on their security characteristics (i) uniqueness, (ii) randomness, and (iii) reliability as well as conventional VLSI design metrics including area, power, and performance [43]. PUFs can take many forms among which some of the common ones are optical PUFs, paper PUFs, coating PUFs and silicon PUFs. In order to identify silicon devices, further variants include PUFs based on delays in a silicon circuitry such as arbiter PUFs and ring-oscillator PUFs, and PUFs based on the start-up behavior of memory cells such as SRAM PUFs, butterfly PUFs and flip-flop PUFs [44].

Potlapally [40] presented an overview of methods adopted to cater for intricacy of validating security of hardware in an industrial setting, and highlighted opportunities for the security research community pertaining to hardware security validation; while Handschuh [44] provided an overview of the state of the art in research on memory PUFs specifically on SPRAM PUFs and presented results from industrialization of such PUFs. Kursawe et al. [42] introduced a reconfigurable optical PUF, based on phase change memory, with a mechanism to convert it into a new unpredictable and uncontrollable challenge-response behavior. The description of their scheme however omitted possible optimizations. Quality factors of ring oscillator (RO) based PUF are negatively affected

by environmental noise and systematic variations in the die. To address this problem, Maiti and Schaumont [45] proposed two methods to achieve a higher reliability in an RO-based PUF, and attempted to verify their results using a small sample size of only five FPGAs.

The reliability of PUFs implemented in CMOS circuits is normally affected by environmental conditions such as voltage and temperature. Kumar et al. [46] investigated two methods for improving the reliability of delay based PUFs, by reducing temperature sensitivity. The first method focused on improving the gate overdrive by operating the PUF at an optimized supply voltage. The second method explored the negative temperature coefficient property of source feedback resistors. They reported 16% improvement in reliability for both these methods. Subsequently, Kumar et al. [47] proposed a temperature-invariant ring oscillator PUF architecture based on serial-input serial-output (SISO) topology interpreting the relative phase difference between two ring oscillators to a digital response bit. They reported that this phase difference based response generation was superior to frequency based response generation in terms of area and power.

Hori et al. [48] developed a physical unclonable function (PUF) with a hardware architecture structure as a large combinational logic. In this research work, the long feedback signal extracted the device variation. Accordingly, the output IDs generated in the different devices became different from each other. The authors have not yet evaluated the indemnity against the existing attacks such as model-building attacks.

Schrijen and Leest [49] investigated the reliability and uniqueness of static random access memories (SRAMs) ranging from 180nm to 65nm in different technology nodes

when used as PUF. The authors presented quantitative results but did not give any technology or architecture analysis as they did not have access to SRAM architectures of all of the tested memories.

For integration of PUFs into low-power and security applications, Lin et al. [50] studied the effects of process technology and supply voltage scaling on arbiter-based PUF circuit design. Using Monte Carlo-based statistical analysis, they demonstrated that advanced technologies and reduced supply voltage could improve the PUF uniqueness due to increased delay sensitivity.

Simon et al. [41] evaluated and compared reliability and uniqueness of Buskeeper PUF developed by them with those of D Flip-Flop (DFF) PUFs. The quality of bit strings generated by PUFs based on resistance variations, in the power grid metal wires and transistor on-resistance in 90 nm chip as well as in the power grid metal wires of 5 65 nm chip, was analyzed by Ju et al. [51]. The authors also investigated a voltage threshold technique to eliminate unstable bits. They reported that the PUF primitives generated cryptographic quality bit strings of length up to 1.6M bits. Bhargava et al. [43] compared bi-stable based PUFs (SRAM and sense amplifiers) and delay based PUFs (arbiter and ring oscillator) using measurements from a test-chip in 65nm bulk CMOS. Their reliability measurements were based on multiple evaluations of PUF circuits across operating voltage ranging from 1.0V to 1.4V and temperature ranging from -20◦C to 85◦C. They reported that bi-stable PUFs were more area-efficient than the delay-based PUFs.

Kumar and Burleson [52] presented a password based hardware authentication using PUF, called (PHAP), which was able to distinguish between a trusted party

and an adversary based on a simple user password during authentication. They showed that the time difference between real time execution of the system by a trusted party and simulation time by an adversary can be very large. Their simulation results showed that the probability of an adversary successfully attacking the system was very low.

The silicon physical unclonable functions (PUF) utilize the uncontrollable variations during integrated circuit (IC) fabrication process to facilitate security related applications such as IC authentication. Yin et al. [53] described a framework to generate secure PUF secret from ring oscillator (RO) PUF. Their work is based on group-based RO PUF, utilizing the concepts including (i) an entropy distiller to filter the systematic variation, (ii) a simplified grouping algorithm to partition the ROs into groups, (iii) a syndrome coding scheme to facilitate error correction, and (iv) an entropy packing method to enhance coding efficiency and security. They demonstrated that these concepts can create PUF secret that can pass the NIST randomness and stability tests.

A hardware-embedded delay PUF was designed by Aarestad et al. [54] to leverage path delay variations that occur in the core logic macros of a chip to create random bit strings. The bit strings produced by a set of 30 FPGA boards were evaluated for uniqueness, randomness, and stability. They also proposed an error avoidance scheme which provided significant improvement against bit-flip errors in the bit strings. Recently, Aarestad et al. [14] presented a PUF, called HELP, based upon path-delay variations. The HELP is capable of (i) comparing paths of widely differing lengths, (ii) reducing the area cost and providing a relatively small amount of entropy, (iii) minimizing invasive design with low area and performance impact, and 4)

providing a hardware-embedded PUF engine requiring no external testing resources. Further, HELP possesses a large number of paths typically found in logic macros such as the advanced encryption standard (AES). This large source of entropy enable HELP to generate large bitstrings, for achieving bit stability and avoiding errors. To prove this PUF concept, and to demonstrate its effectiveness, the authors designed a complete, functional FPGA-based implementation of this PUF and validated it on FPGA boards. Their results of hammering distance and NIST statistical test analysis established that the bitstrings, being of high quality, are unique and random, and hence appropriate for cryptographic applications.

In their review on some applications of memristor, Mokhtar and Abdullah [55] described that LTspice memristor model is used to simulate memristor behavior and applied to the basic delay element circuit. It controls the current flowing to the parasitic capacitor, thus controlling the delay. As process variations become more prevalent due to technology scaling into the nanometer regime, nano-electronic technologies such as memristors become viable options for improved security in emerging integrated circuits. Rose et al. [56] provided an overview of memristor-based PUF structures and circuits that illustrate the potential for nano-electronic hardware security solutions.

## 2.8 Summary

In my research, I present REBEL (regional delay behavior) as an embedded test structure (ETS), for path delay measurement that can be utilized for measuring within-die variation, and in several other applications, for example, Trojan detection, delay defects detection, and Physical Unclonable Functions. I describe the detailed architecture of REBEL and demonstrate its effectiveness for measuring delays and capturing the within-

die variations caused by the environmental and physical process variations. I have integrated REBEL with a functional macro on the 90nm chips as well as on the 28nm Zynq FPGA to illustrate the effectiveness and efficacy of REBEL in measurement of within-die variations. REBEL is implemented on FPU (Floating Point Unit) in 90nm process technology and on AES (Advanced Encryption Standard) on 28nm FPGA boards; I have gathered and analyzed data from 52 copies of 90nm chips and 11 copies of FPGA boards. The results are elaborated in chapter 5 and chapter 6. Subsequently, in Chapter 7 I discuss the implementation of parallel hardware accelerator on pipelined decision tree classification engine, Chapter 8 concludes with the benefits of REBEL for various applications and summary of the work. Chapter 9 suggests the future work.

# CHAPTER 3
# REBEL: Embedded Test Structure and Macros-Under-Test

Within-die variations are caused by a cross-field effects [17] in compassing the layout design techniques, optical aberration and other random effects such as dopant fluctuations. Within-die characterization requires more density of test structures to capture the variations in all regions.

## 3.1 REBEL- Regional Delay Behavior

The embedded test structure called REBEL (Regional dELay BEhavior) is designed to measure path delays in a minimally invasive fashion; and its architecture measures the path delays more accurately. REBEL can be integrated with the traditional scan design which is used in the design for testability, to improve the observability and control-ability of the sequential design. There are many variants of scan cells, mux-D scan, LSSD scan and clocked LSSD scan are most widely used. In this chapter I discuss in detail the modifications needed to integrate REBEL into a clocked-LSSD-style scan architecture.

A REBEL ETS components consist of a set of scan-chains, row control logic header (RCL) and front-end logic for each scan cell to work in different operational

modes. A row of scan flip-flops (FFs) is shown in Fig. 3.1 along the top which serves to launch transitions into the MUT. The bottom row is used to capture transitions that propagate through the MUT. REBEL ETS components are integrated into this row and are labeled “row control logic” and “front-end-logic” in the figure. Where the macro-under-test (MUT) is the combinational logic from a core logic macro.

Transitions can be launched into the MUT using standard manufacturing delay test strategies such as launch-off-capture and launch-off-shift [28]. In either of these two scenarios, the scan chain is loaded with the initial pattern of the 2-pattern test and the system clk (Clk) is asserted to generate transitions in the MUT by capturing the output of a previous block or by doing a 1-bit shift of the scan chain, resp. The transitions that propagate through the MUT emerge on some of its outputs. REBEL allows only one of these transitions to be measured at a time in a specific region of the MUT, as indicated by the label PUT for path-under-test in the figure. The PUT’s transition normally drives only the D input on the capture FF. However, the REBEL component labeled “front-end” logic allows this transition to be diverted to the scan input (SI) on the FF. This special logic also converts all scan FFs to the right of this insertion point FF into a delay chain. A digital snapshot of the signal as it propagates along the delay chain can be obtained by de-asserting Clk. The digital snapshot can be used to determine the timing of the PUT, and because it captures the temporal behavior of the PUT, it can also be used to determine if any glitching occurred. This is a unique and powerful feature of REBEL that is fully exploited in this work.

A special mode called flush-delay (FD) can be used to implement the delay chain in LSSD-based scan architectures. FD mode is enabled by asserting both the scan A and B

clock signals simultaneously. These signals are labeled “global SCA” and “global SCB” in Fig. 3.1. With both signals asserted, both the master and slave of a scan FF are transparent, allowing any transitions on SI to propagate through both latches after a Δt that represents the delay.

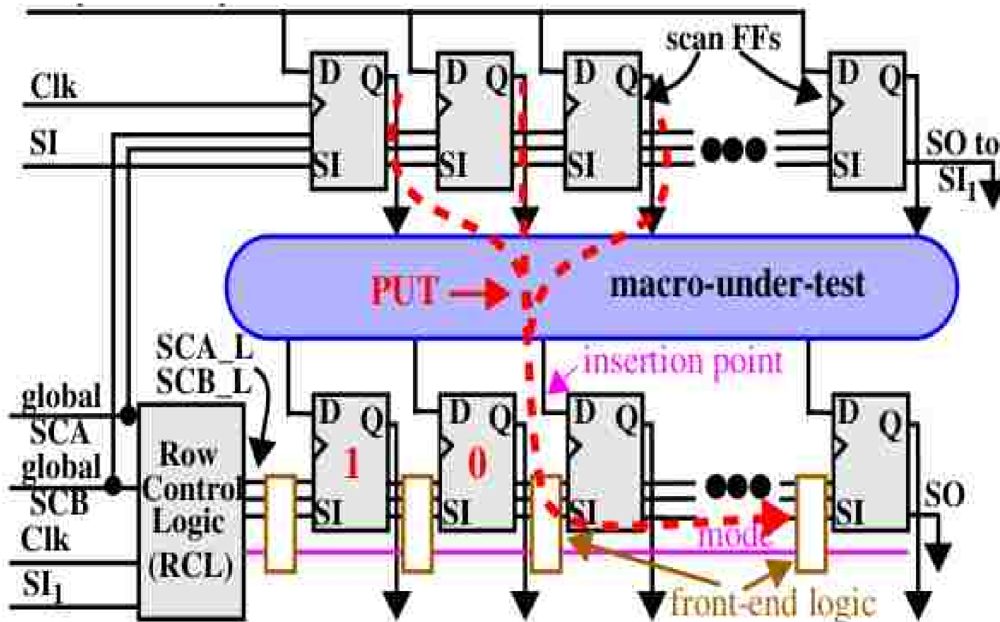


*Fig 3.1: REBEL Integration Strategy*

REBEL is required to implement two additional modes in the capture scan FFs shown along the bottom of Fig. 3.1 (in addition to the usual functional and scan modes). In particular, the scan FFs to the left of the insertion point need to preserve their contents during the Clk launch-capture (LC) event, while the FFs to the right of the insertion point need to implement the delay chain.

These two modes are realized using the RCL block, a special scan chain encoding and the front-end logic shown in Fig. 3.1. The mode is controlled by configuring two FFs in the RCL block while the scan chain encoding serves to specify the insertion point of the PUT.

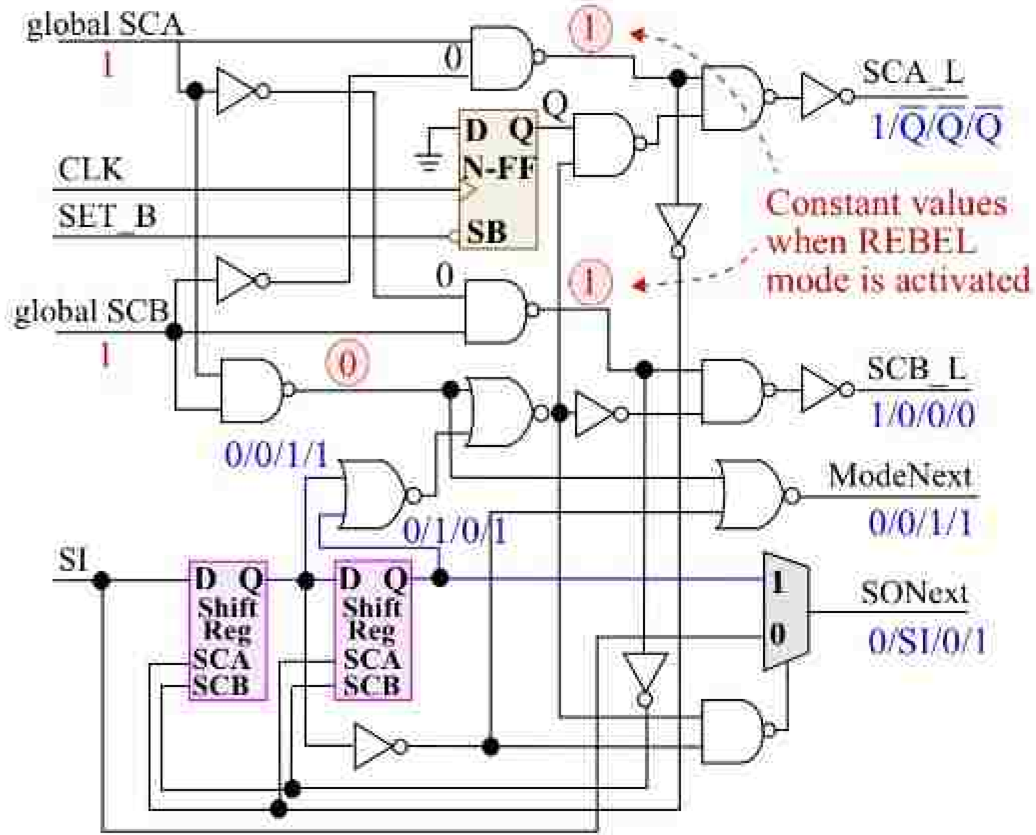


*Fig 3.2: REBEL Row Control Logic*

Fig. 3.2 shows a schematic diagram of the RCL. The top portion of the diagram controls local (row-specifichiefenercon) scan clock signals, labeled SCA_L and SCB_L (L for local) while the bottom portion contains two shift registers (Shift Reg) and mode select logic. A large portion of the RCL logic is dedicated to allow the scan FFs in the capture row, hereafter referred to as row-FFs, to operate in functional or scan modes. The chip-wide scan signals labeled 'global SCA' and 'global SCB' are used to specify one of the three possible operational states for the chip. When both are low, functional mode is in effect. Scan mode is implemented when these signals are asserted in a non-overlapping fashion. The timing mode used by REBEL is in effect when both of these signals are asserted, as illustrated by the annotations in Fig. 3.2.

| Shift Register | Mode of Operation | Functionality |
|---|---|---|
| 00 | functional mode | All Scan FFs in row are in functional mode |
| 01 | Flush Delay Continuation mode | All Scan FFs in row are in Flush Delay Continuation mode |
| 11/10 | Mixed mode | Left scan FFs in preserve-contents mode, right scan FFs in FD mode, referred to as mixed mode |

*Table 3.1 : Configuration modes for REBEL rows*

When REBEL mode is in effect, the specific mode of operation of the associated row-FFs is determined by the two shift registers. Table 3.1 identifies the modes for each of the four configurations. The bit configuration “01” (FD continuation mode) is required only in cases where there are multiple regions in the MUT . Bit configurations “10” and “11” specify the mixed mode described above, where FFs to the left of the insertion point are in preserve-content mode while those to the right are in FD mode. The outputs from the RCL block shown in Fig. 3.2 are annotated to show the values under each of these four bit configurations. Further operational details of the RCL block can be found in [11].

Fig. 3.3(a) shows a clocked LSSD FF (CLSSD) used in the FPU macro. It consists of three latches. The functional path master-slave (MS) pair shown on the left is driven by Clk. The slave latch is dual ported and also serves as the master in the scan path MS pair on the right. Fig. 3.3(b) shows the additional ‘front-end’ logic for REBEL. The functional path’s D-input is fanned out to a 2-to-1 MUX, which allows for the insertion of a macro’s PUT into the delay chain during the REBEL test.

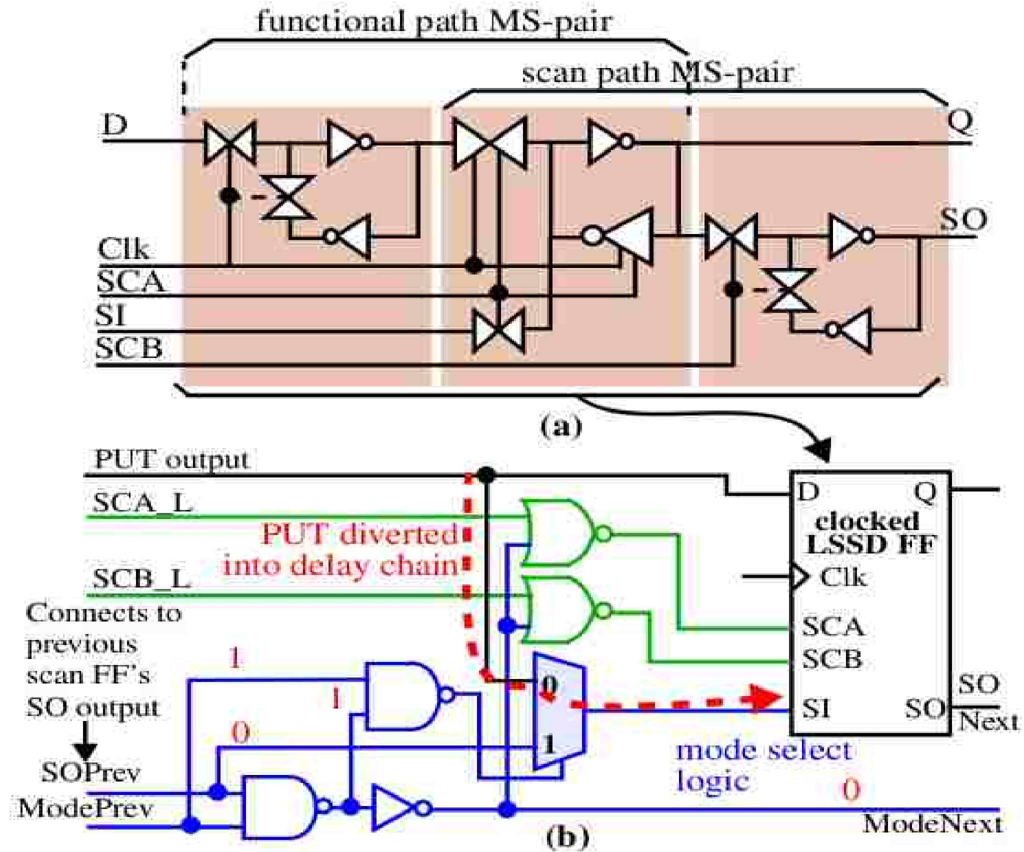


*Fig 3.3 (a) Modified clocked-LSSD scan FF and b) Additional front-end logic*

This is accomplished with the mode select logic shown along the bottom of the figure. A specific insertion point is selected by pre-loading the row-FFs with a pattern of all '1's followed by a '0' from left to right along the row-FFs (see Fig. 3.1). Reference [11] provides specific operational details. Note that the front-end logic adds only a small capacitive load to the functional path and therefore the impact on performance is very small.

## 3.2 Floating Point Unit (FPU) Macro

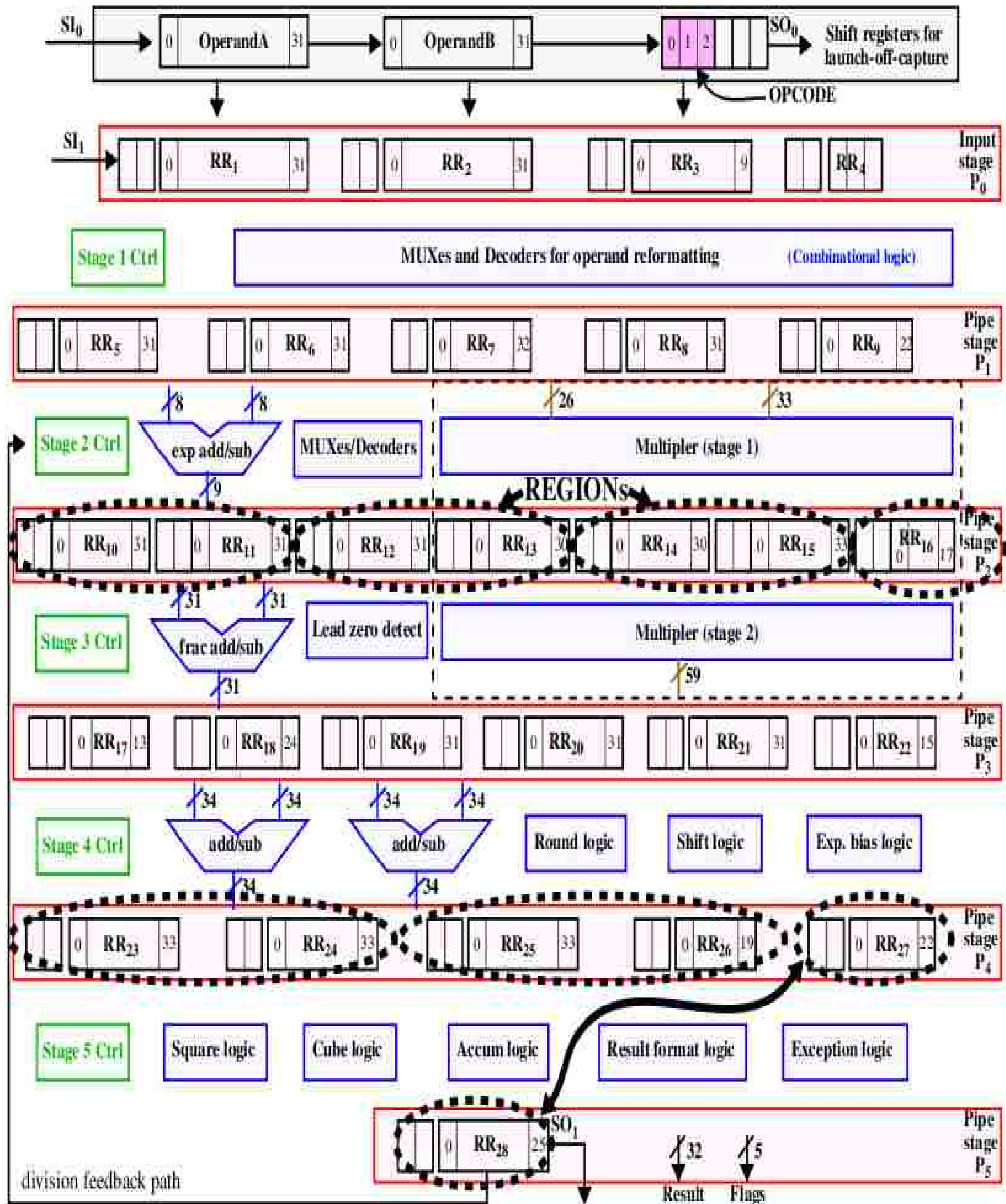


*Fig 3.4: Floating Point Unit Block Diagram*

Fig. 3.4 shows a block level diagram of a floating point unit (FPU) incorporated on the chips, as well as the inserted REBEL rows, labeled RRx from 1 to 28. All of the 817 FFs (56 row header FFs + 761 functional unit FFs) are wired together into a single scan

chain with input SI1 shown in the upper left and output SO1 shown along the bottom of the figure. A separate set of 70 shift registers are inserted on the inputs (top-most row in figure) which serves to enable a launch-off-capture testing strategy [28]. Here, the Input Stage P0 is loaded with the 1st pattern while the shift registers are loaded with the 2nd pattern.

The FPU is designed as a 5-stage pipeline, labeled P1 through P5, with MUXes, decoders, adder/subtractors, a multiplier, etc. inserted between the pipeline registers. The FPU is capable of carrying out 8 different operations, including add, subtract, float-to-integer, integer-to-float, negation, absolute value, multiplication and division. The 3-bit OPCODE shown along the top right in the figure determines the function. All operations except division can be carried out with a throughput of one operation/clk cycle. Division requires 5 clock cycles to complete with data fed back from the output of pipeline stage 5 to pipeline stage 2 through the 'division feedback path'. Given this pipeline structure and the constraints described above regarding REBEL, it is possible to carry out REBEL testing using 4 basic configurations. In the first two configuration, Cfg1 and Cfg2, the REBEL rows in pipeline stages P0, P1 and P3 are configured in functional mode while those in P2, P4 and P5 are configured in REBEL mode. In configurations Cfg3 and Cfg4, the rows in P0, P2 and P4 are configured in functional mode while those in P1, P3 and P5 are configured in REBEL mode. These 4 configurations collectively allow paths in all of the logic blocks to be tested using the REBEL ETS.

We create 2 configurations, Cfg1 and Cfg2, to handle a limitation that is illustrated in Fig. 3.1. In particular, the delay chain for insertion points on the right side of the MUT is very short and, in fact is non-existent for the right-most insertion point. The FD

continuation mode described in reference to Table 3.1 allows REBEL rows to serve as extensions of the delay chain. The dotted circles over row pairings RR 10-RR11, RR12-RR13, etc. (labeled REGIONs) in Fig. 3.4 illustrate how continuation rows are paired with mixed-mode rows. For example, RR10 of the RR10-RR11 pairing is configured in mixed mode which allows delays on each of the PUT outputs in this region to be measured, one at a time, while RR11 is used to extend the delay chain so that all insertion points can be timed along a non-zero length delay chain, particularly those on the right side of RR10. The circles illustrate the organization of mixed-mode (left) and continuation rows (right) for Cfg11.

A complementary pattern is used for Cfg 2 so that PUT outputs in, e.g., R11 can also be timed with R12 serving as its continuation row. Although only one path in each row can be tested at a time, up to 8 paths can be timed simultaneously across the various regions. Given these 4 configurations, there are a total of 684 Ffs that can serve as insertion points. This number excludes the 56 row header FFs and the 77 FFs in pipeline stage P1 (these FFs are always configured in functional mode).

We apply a random testing strategy to the FPU where the values placed in the functional rows are generated by an pseudo-random number generator. In order to allow specific functional units to be the target of our testing, we specify the OPCODE bits in the functional rows instead of inserting random values as we do for the remaining FFs of these rows. The top shift register row identifies the 3 OPCODE bits, which also exist in each of the pipeline stages (not shown). For each random pattern, a sequence of REBEL tests are applied which configure the position of the insertion point incrementally from left-to-right across each of the REBEL mixed-mode rows. Therefore, the same random

pattern is applied multiple times as a means of testing all path outputs. The insertion point in each RRx mixed-mode row is incremented until the right-most FF becomes the insertion point. The number of LC tests per test pattern is determined by the longest rows in each configuration, which are given as 34, 34, 33 and 32 for Cfg 1 through Cfg4, resp.

## 3.3 Advanced Encryption Standard (AES) Macro

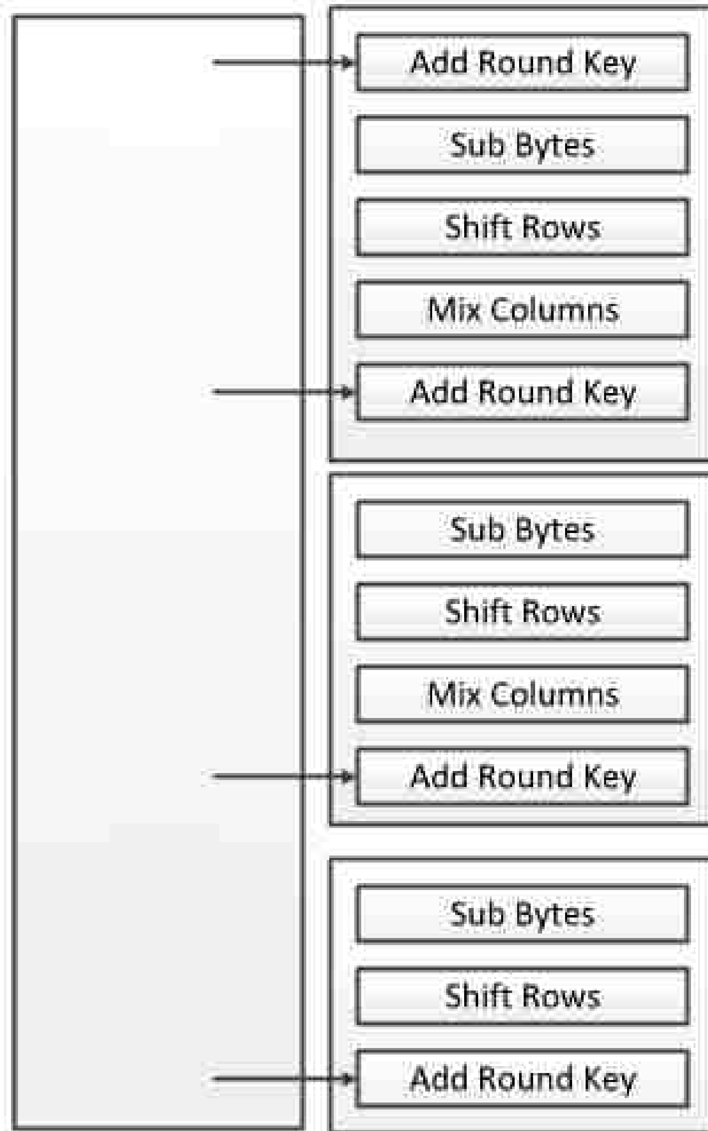


*Fig. 3.5: AES Engine*

In the FPGA implementation of REBEL, the macro under test is a round of AES encryption. The combinational logic, shown in Fig 3.5, has an input of 256 and the output is also 256 bits. We utilize the input flip flops of the existing logic as launch row and use the output row as the capture row. The scan chain is implemented using Mux-D flip flops, which require only the system clock as compared to C-LSSD clock. The schematic diagram of Mux-D flip flop is shown in Fig. 3.6.

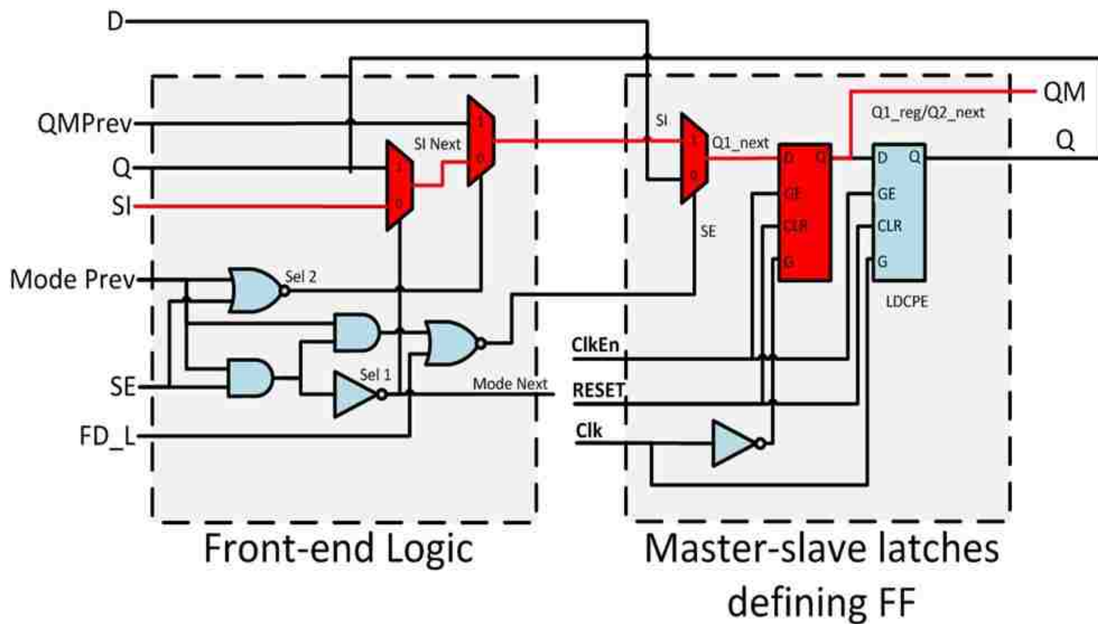


*Fig. 3.6: Mux-D Flip Flop*

AES engine is a block cipher type of encryption with a block length of 128 bits. AES has different versions which allows for three different key lengths generation, that is, 128, 192, or 256 bits. The encryption rounds vary with the desired key size, for 128-bit key the processing includes 10 rounds, for 192 bit keys, 12 rounds and for 256 bits the number of rounds is 14. Each round of processing includes one single byte based substitution step using a non-linear substitution table (s-box), a row-wise permutation step, a column-wise mixing step, and the addition of the round key[57]. Figure (3.7) shows the different steps that are carried out in each round.

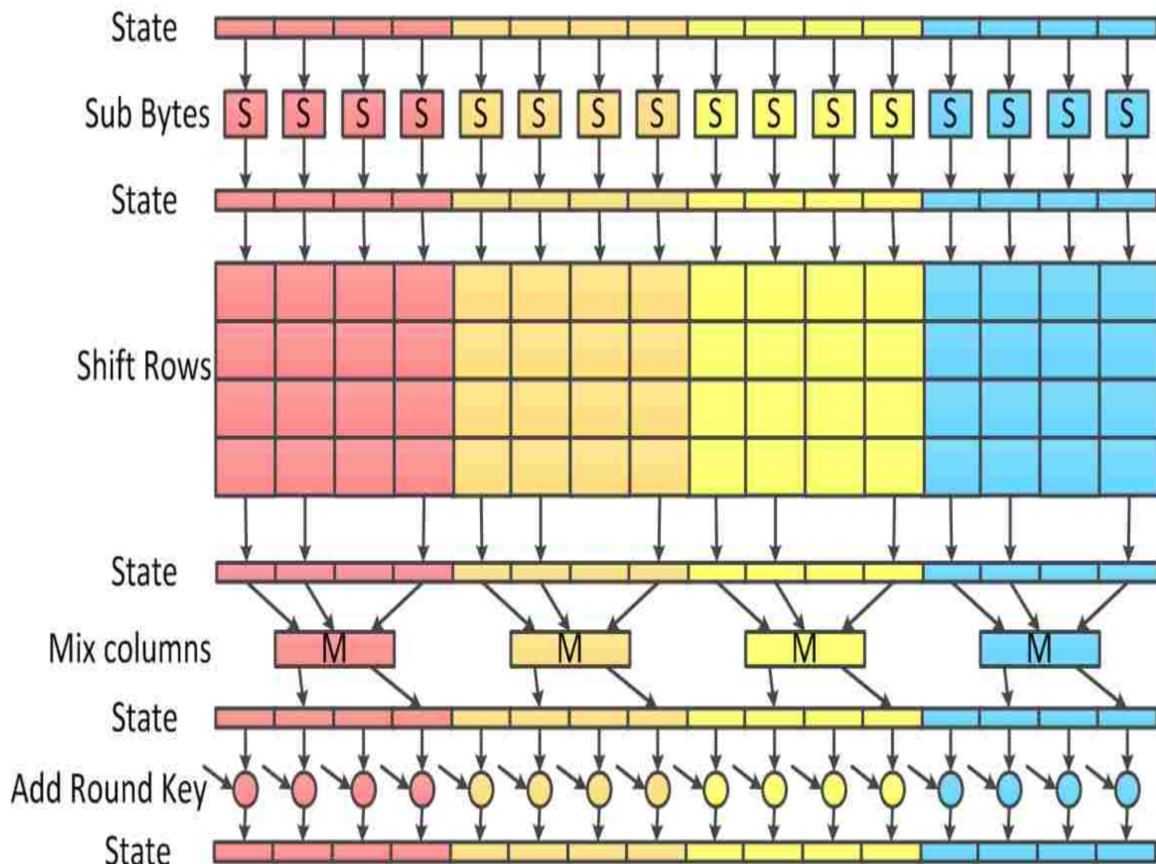


*Fig. 3.7: AES Round*

### 3.3.1 STEP 1, SUB BYTES:

This step performs a byte-by-byte substitution during the forward process of encryption. It consists of using a 16 × 16 lookup table (s-box) to find a replacement byte for a given byte in the input state array. This substitution operates on each of the State bytes independently and the entries in the lookup table are created by using the notions of multiplicative inverses and bit scrambling to destroy the bit-level correlations inside each byte. The corresponding substitution step called InvSubBytes is used during decryption

### 3.3.2 STEP 2, SHIFT ROWS:

Shifting the rows of the state array during the forward process. This step of

transformation hides the byte order inside each 128-bit or 256-bit block. The corresponding transformation during decryption is denoted InvShiftRows for Inverse Shift-Row Transformation.

### 3.3.3 STEP 3, MIX COLUMNS:

This step of mixing up of the bytes in each column separately during the forward process further shuffles up the 128-bit input block. The shift-rows step along with the mix-column step causes each bit of the cipher text to depend on every bit of the plain-text after 10-14 rounds of processing. This way each bit of the plaintext affects every bit of the ciphertext in a block. The corresponding transformation during decryption is denoted InvMixColumns and stands for inverse mix column transformation.

### 3.3.4 STEP 4, ADD ROUND KEY:

Adds the round key to the state of the previous step during the forward process by a simple bitwise XOR operation,that is each column of the state is XORed with a word from the key schedule. The corresponding step during decryption is denoted InvAddRound- Key for inverse add round key transformation.

In the FPGA implementation, the initial pattern and final patterns are given on the input of bock cipher during the launch and capture phase respectively and the path delays are measured for the paths where the transitions appear on the insertion point and reach successfully to the target flip flop. The paths with the glitches are filtered from the analysis to avoid the measurement noise in the die-to-die and with-in die variation measurements.

## 3.4 Chip Layout with REBEL AES and FPU Placement

### 3.4.1 ASIC Layout with REBEL and FPU

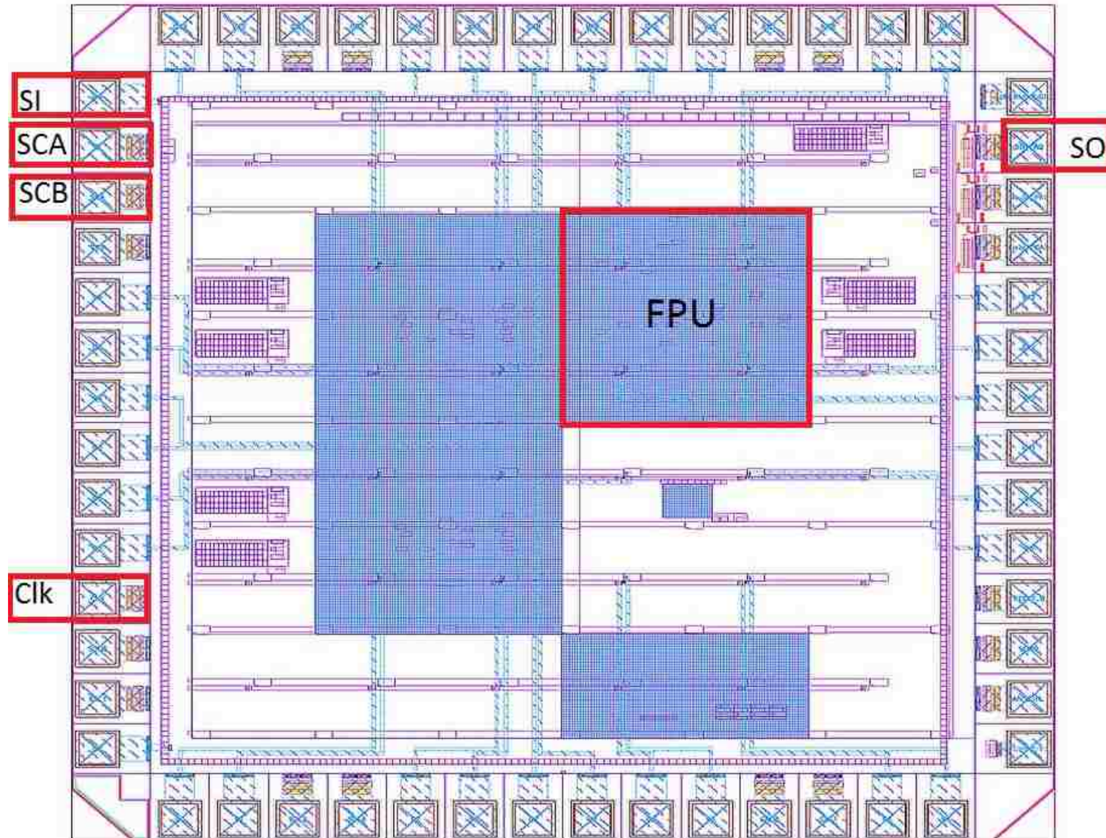


***Fig. 3.8: ASIC Layout***

These macro-under-tests are fabricated in IBM 90nm CMOS bulk technology, layout is shown in Fig. 3.8. REBEL is embedded in Advanced Encryption System (AES) and 32 bit pipelined Floating point unit (FPU) in the design phase. There are 58 copies of the chip on which the analysis are performed.

The flowchart in Fig. 3.9 shows the design automation process followed for the test chip design. The behavioral HDL descriptions of AES and FPU macros are first synthesized using Cadence RTL Compiler. The macros are synthesized using a scan

insertion DFT flow and all the flip-flops are replaced by the scan cells. The REBEL test structure is integrated into this scan-able design using an in-house PERL script. These scan cells are replaced with the REBEL-scan cell, which is a scan cell with the 'front-end' logic. For Pipelined implementation of FPU there are 28 segments of these scan chains where for each segment a row control logic (RCL) is added, and for AES there is 1 segment and 1 RCL is added.

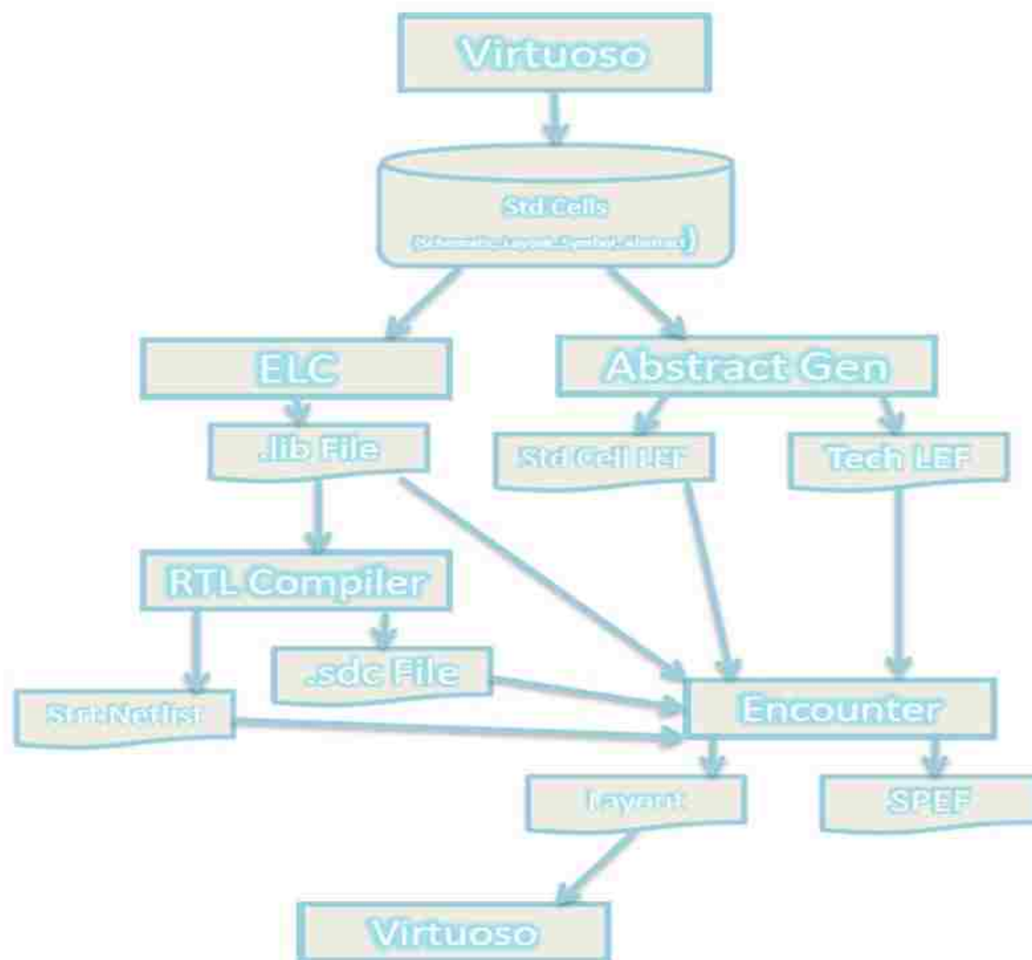


***Fig. 3.9: ASIC Design Flow***

Data is collected from 58 copies of the fabricated chip manufactured in 90nm CMOS technology from MOSIS. The path delays measured in these chips are used to

present REBEL ability to measure variability in path delays for understanding the process variation and within-die variations.

### 3.4.2 FPGAs Layout with REBEL and AES Round

The within-die variation is studied on the FPGA having reconfigurable logic. Delay based variation is studied on ZYNQ FPGA on ZED boards with AES as macro under test to study within-die and die-to-die variation among 11 copies of chips. We measure the path delays as discussed under the section of path delay measurement using target flip-flops. Below is the layout using plan ahead shown in Fig. 3.10.

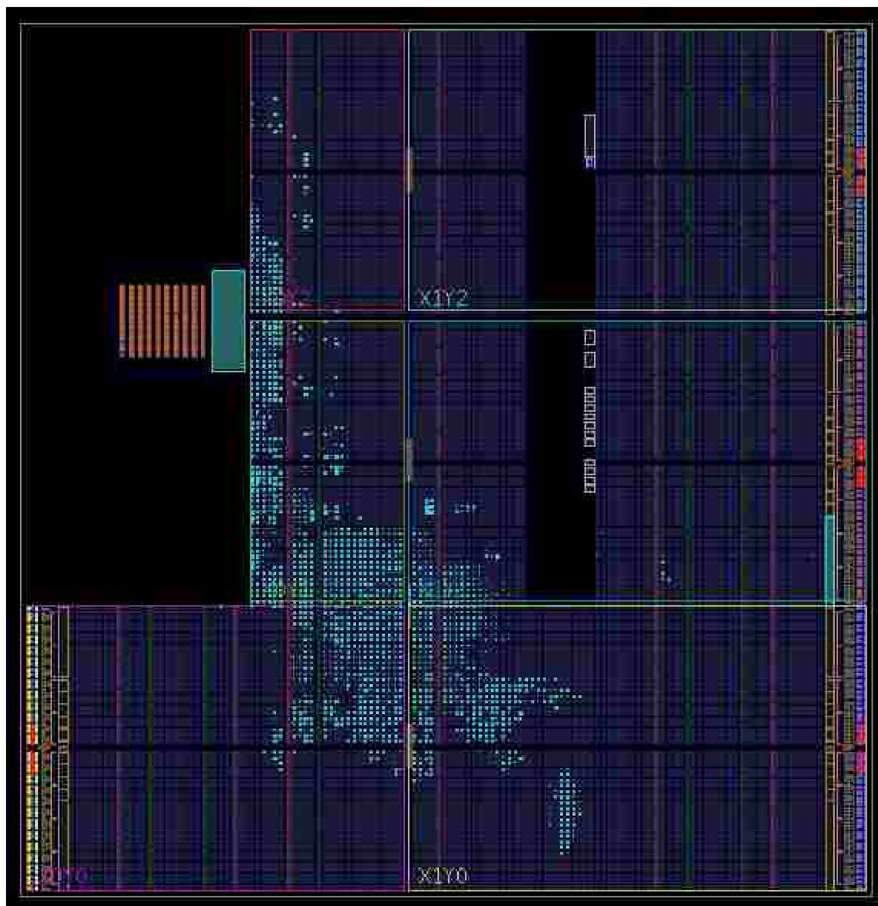


***Fig. 3.10: FPGA Layout***

# CHAPTER 4
# Experimental Setup

An embedded test structure called REBEL, suitable for measuring within-die variations in actual product macros, is employed in our research. A floating point unit fabricated in IBM's 90nm technology is used as a test vehicle in chip experiments carried out at nine different temperature/voltage corners.

## 4.1 Experimental Setup for IC 90nm Chips

A photo of the experimental setup is given in Fig. 4.1. A Linux- based host computer runs a custom LABVIEW application that controls the testing and data collection process through GPIB, USB and serial interfaces. A Virtex-6 FPGA is used to configure the scan chains on the chip-under-test (CUT), as well as deliver the LC clock sequence . A ribbon cable and several high-speed coaxial cables connect from the FPGA to a custom printed circuit board, which includes a zero-insertion-force socket for the CUT. This board and the CUT are not visible in the photo because they are inside the temperature chamber. The temperature setting on the chamber as well as several power supplies are also controlled by the LABVIEW interface.

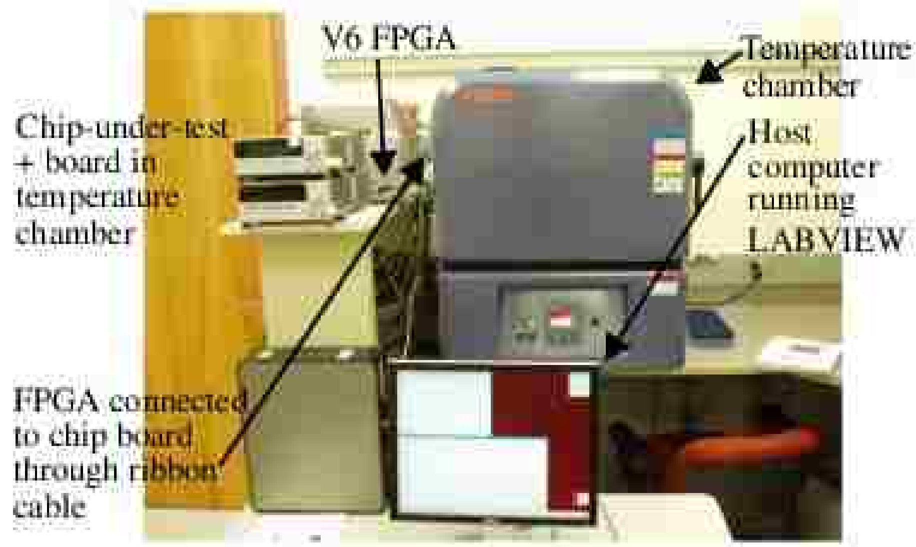


*Fig 4.1: Experimental Setup*

## 4.2 Launch-Capture Clocking Sequence and Clock Strobing

### 4.2.1 Launch-Capture Clocking Sequence

The launch-capture clock sequence is generated using a digital clock manager (DCM) on a Virtex-6 FPGA. The fine phase adjust (FPA) feature on the DCM allows the LCI to be set with a resolution of 17.86 ps. A specific FPA is configured into the DCM by a state machine running on the FPGA which accepts an integer input parameter from the controlling LABVIEW application. Valid values of the FPA are between 0 and 560, which corresponds to a programmed LCI between 0 and 10.000 ns.

### 4.2.2 Clock Strobing

Clock strobing involves repeating the test sequence at incrementally longer LCIs. The RCL and front-end logic for REBEL allow critical timing events, i.e., the launch-capture interval (LCI), to be controlled by the system clock. This is

illustrated by the timing diagram shown in Fig. 4.2.

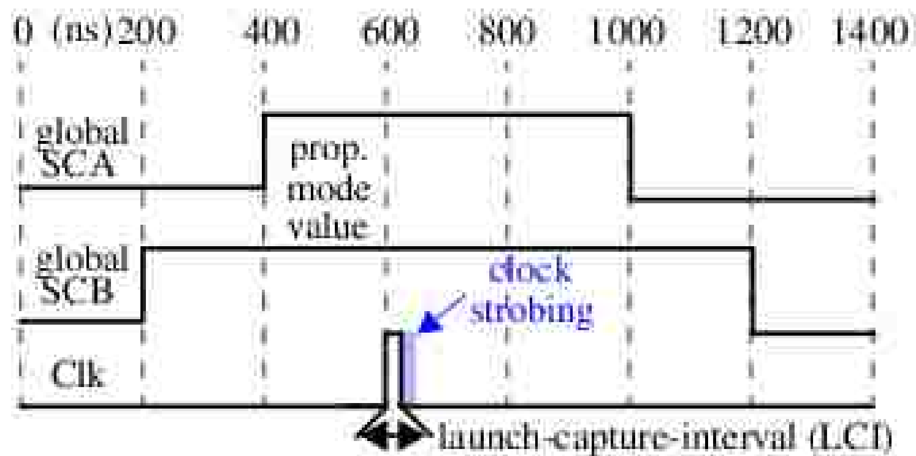


*Fig 4.2: REBEL Launch-Capture(LC) Test Sequence. Clock Strobing applies a sequence of LCIs of different widths*

## 4.3 Delay Measurement Process

A scan operation is first carried out that configures the RCL blocks, the scan chain encoding sequence for mixed mode rows and random test pattern data as described above. Prior to the LCI test, the global SCB signal is asserted and then the global SCA signal. Staggering these events prevents race conditions that would otherwise destroy the encoding sequence. With both scan clocks asserted, the mode control signal shown in Fig. 3.1 propagates along the mixed mode rows setting up the insertion point and delay chain. The LCI test consists of asserting the Clk signal, which launches transitions in the combinational logic (this is referred to as a launch-off-capture delay testing strategy as described in [28]), and de-asserting the Clk signal a fixed $\Delta t$ later, which halts all signals propagating along the delay chains. The delay in a combinational path can be computed using Eq. 1.

$$T_{path} = T_{lc} - T_{dc} \qquad \text{Eq. 1.}$$

where, Tpath = Delay of the combinational path

Tlc = Launch/Capture Interval (LCI) delay

Tdc = Delay through the delay chain.

The resolution of the measured delays is limited by the delay through each of the master-slave FFs that implement the delay chain. This delay is typically larger than the desired resolution, e.g., in our chips, it is approx. 500 ps. To increase the resolution clock strobing is employed and delays are calculated.

The pulse-creation logic within the FPGA as well as the response characteristics of the FPGA pads prevents clock pulses narrower than 1160 ps, given as FPA 65 * 17.85 ps, from being produced on the clock output pin of the FPGA.

Actual LCI is somewhat different than the programmed value. Fig. 4.3 plots the programmed FPA on the x-axis against the actual LCI produced by the FPGA (we round all delays to the nearest 5 ps value to ease with the illustrations in the dissertation).

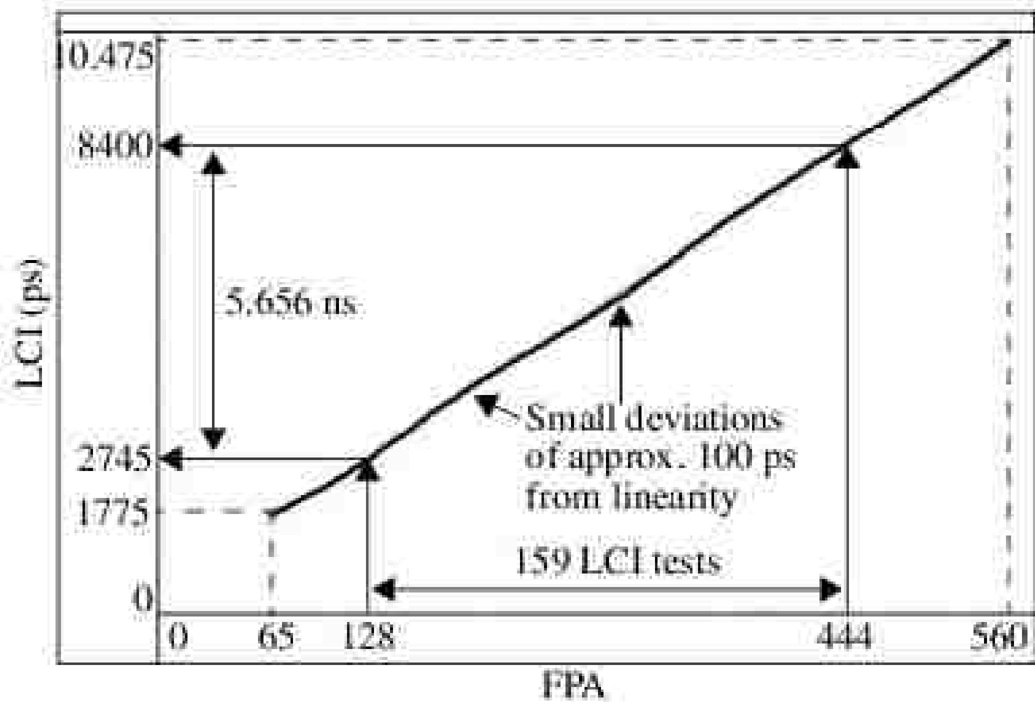


*Fig 4.3: Oscilloscope measured launch-capture intervals (LCIs) for each of the Fine Phase Adjust (FPA) values on the FPGA. Row Header delay of 300 ps is included*

Here, the LCIs are measured at the CUT's clock input pins using a high resolution digitizing oscilloscope. Although the curve is nearly linear, small variations of up to 100 ps occur in several locations, as highlighted in the figure. Also, the curve is shifted upwards, e.g., the programmed delay at FPA 65 is supposed to be 1160 ps but is 1775 ps instead. This occurs because the pulse-creation logic within the FPGA increases the width of the LCI by approx. 615 ps.

In our experiments, we apply a sequence of LCI tests over the range of FPAs between 128 to 444 in FPA increments of 2. This results in the application of (444-128)/2 + 1 = 159 LCI tests with actual LCIs between 2745 and 8400 ps as given by the oscilloscope curve in Fig. 4.3.

## 4.4 Digital Snapshots and FF Delays

The raw data captured in the delay chain is a string of binary bits, one string for each of the 159 LCI tests applied to test a path. Fig. 4.4 shows the digital snapshots for the first 21 LCI tests of a path in a vertical sequence. The insertion point in this example is FF15 of the REBEL mixed-mode row RR12 under Cfg1 from Fig. 3.4. RR13 is shown on the right side and serves as a continuation row under Cfg1. The programmed FPA and the actual LCI (from Fig. 4.3) for each snapshot are displayed on the left side of the figure. The first FPA (128) shows a sequence of 6 0's in the left portion of the snapshot. This indicates that a falling edge propagated along 6 elements of delay chain, i.e., through FF15 through FF20, before being halted by the capture event. In each subsequent snapshot up through FPA 132, the edge continues to propagate through FF20 but fails to reach FF21 until FPA 134 is applied. The falling edge requires 17 more FPAs, i.e., 134 through 166, to propagate completely through FF21. From these snapshots, it is possible to derive the approx. delay through FF 21 as (3465 - 2850) = 615 ps.

A similar analysis can be carried out for FF 22 through FF30 using the remaining snapshots for this path. Note that we cannot determine the delays for FF15 through FF20 because the edge has already propagated into this FF on the first LCI.

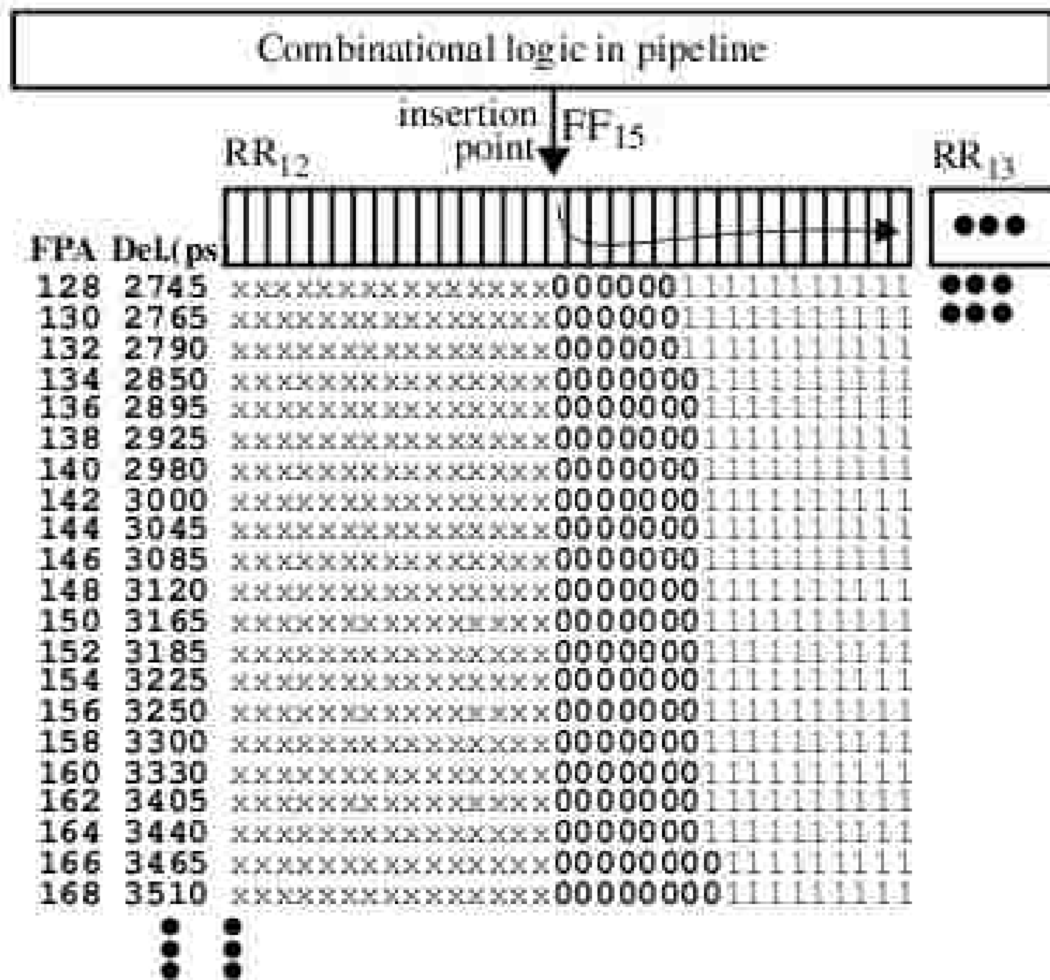


***Fig. 4.4: Illustration of a partial sequence of digital snapshots (21 FPAs of the 159) produced from a path delay test***

## 4.5 Calibration and Power Rail Voltage Transient Effects

A large fraction of the paths tested using our random test patterns are shorter than the Δt associated with the smallest applied LCI. The example in Fig. 4.4 illustrates this situation, which shows that the rising edge has already propagated into FF 20 under the first LCI test. Although it is possible to use shorter LCIs to test this path, thereby eliminating the delay chain elements, doing so requires testing the chip with a faster-than-at-speed clock sequence. It is well known that applying faster-than-at-speed tests results

in false delay measurements, i.e., delay measurements that are ‘distorted’ by the large power supply voltage transient associated with two closely placed (launch-capture) edges. One of the stated advantages of REBEL is that it allows accurate timing information to be obtained for these short paths without using faster-than-at-speed LCI tests. However, in order to do so, a mechanism is needed to eliminate the delay chain components.

We have already described how the delay through each FF can be determined using the example in Fig. 4.4. Unfortunately, the actual delay through the FF is a function of the LCI it is tested with. In other words, the value of 615 ps is computed using FPAs 134 and 166 for FF21, but the actual delay is different for each of these FPAs. This is true because even for LCIs that fall within the valid operational frequency for the chip, a power supply voltage transient is still produced, and this transient impacts delay. We also found that the voltage transient produced for a given FPA is largely independent of the random test sequence applied, i.e., it’s shape is primarily determined by switching events in the clock tree and FFs. However, different FPAs change the shape of the transient and the corresponding delay along the combinational logic paths as well as the delay chain. Therefore, in order to properly capture and analyze the variations which occur along paths within a combinational logic block, it will be necessary to test the paths using a single FPA (ideal) or, as will be necessary for our method, a set of FPAs that fall within a small range.

Since it is impossible to measure the delay through a FF using only one FPA (unlike path delays), we use the technique above to compute the delay and then ‘assign’ this delay to the LCI which represents the midpoint between the FPAs used to time it. In

the example above, we computed the delay difference 615 ps using the delays at FPAs 134 and 166. The midpoint FPA is (134+166)/2 = 150. Therefore, the measured FF delay of 615 ps is the approx. delay through this FF when the LCI used is 3165 ps (delay at 150 from Fig. 4.4). Note that this only approximates the delay and will be the main source of error in the estimation of path delays as we show in Section 5.3.

The delays through the individual delay chain FFs are computed in this fashion to illustrate the impact of the power transient. In this analysis, we use only paths that produce a stable transition. A stable transition is defined as a path that produces exactly

one rising or falling transition across all 159 digital snapshots, i.e., there is no glitching. Using 8 random vectors and the 4 configurations described earlier, a typical chip has approx. 825 stable paths. Each stable path allows, on average, the delays of 12 FFs to

be estimated. Therefore, nearly 10,000 FF delays can be derived from the test data for each chip. Many of the FFs are timed multiple times by different paths and different FPAs. For example, the delay through the FF21 in reference to Fig. 4.4 can be obtained from the path test as described in Section 4.4, and by any other stable path test that drives insertion points to its left in the row.

Fig. 4.5 shows a plot of all FF delays for CHIP1. The x-axis plots the midpoint LCI against the FF delay (y-axis). Each line connected curve in the group labeled “Curves of individual FF delays” represents the delays computed for one FF but at a variety of midpoint LCIs -- whatever became available after processing the snapshots for the stable paths. Therefore, each curve contains only a subset of the 159 FPAs used in the experiments. The vertical dispersion of the “individual” curves shown in Fig. 4.5 is caused by process variations among the FFs, i.e., the curves shown along the top of the

figure belong to slower FFs. Setting aside these differences, there is a underlying pattern to the behavior of the curves.

To more clearly show this, we compute and superimpose a curve labeled “Average delay curve” which represents the average of the individual curves. A second curve labeled “Smoothed average delay curve” is computed from the “Average delay curve” by removing the high frequency components. It completely eliminates process variation effects and best depicts the underlying behavior.

It is a remarkable fact that the delay through any given FF varies by as much as 2X when tested over the range of FPAs between 128 and 444. Dotted lines illustrate that the average delay changes from approx. 300 ps to 600 ps. The impact of the power transient is even more dramatic for LCIs less than the smallest one used in our experiments, as illustrated by the dotted line labeled “Faster-than-at-speed behavior” on the left side in the figure. We purposely avoid the region below approx. 4000 ps because of the large distortion in the FF and path delays.

From this analysis, it is clear that varying the LCI when timing paths will distort the results. As a consequence, we limit the LCIs considered valid for path delay testing to a small region or window between 4355 and 5250 ps (approx. 1 ns), which is delineated and labeled as “$W_1$” in Fig. 4.5. The FF delays in this window remain relatively constant, within 25 ps, and therefore, so will the path delays. The primary benefit of using a range (as opposed to a single LCI) is that it allows a larger number of stable paths to be successfully timed using clock strobing.

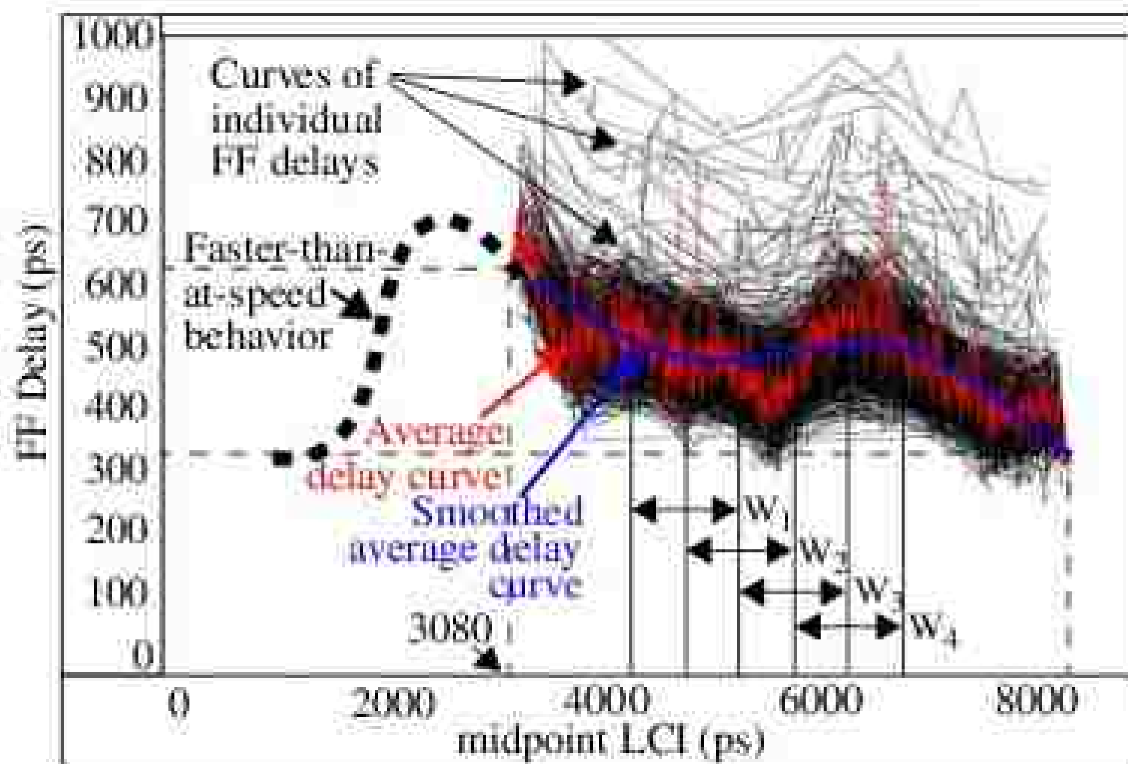


***Fig. 4.5: Curves of individual FF delays measured in different midpoint delays. An average of all individual curves is superimposed, as well as a curve with only the low frequency components of the 'Average delay curve'***

## 4.6 Measuring and Calibrating Path Delays

Digital snapshots, similar to the sequence shown in Fig. 4.4, are parsed to determine the delay of a stable path. The snapshots are parsed in reverse order starting with the digital snapshot associated with largest LCI in the window described above, which is LCI 5250. Parsing in reverse order ensures the last transition is used as the path delay in cases where there is uncertainty (which is described in Section 5.2). This process is illustrated in Fig. 4.6 using snapshots for the path referenced in Fig. 4.4 but at larger FPAs.

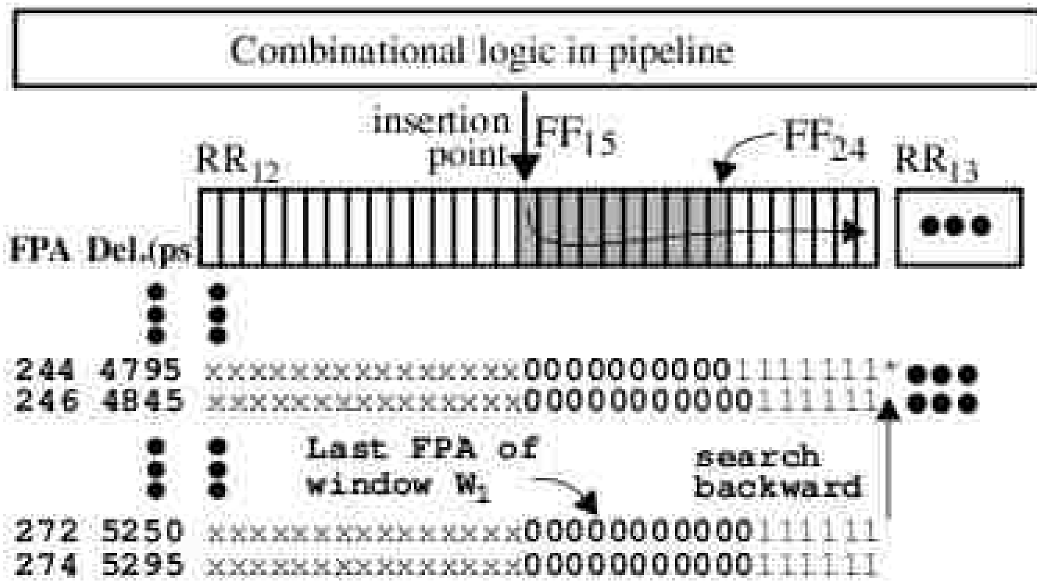


*Fig. 4.6: Calculating path delay using digital snapshots*

The snapshot at LCI 5250 indicates that a falling edge is propagating through FF25. Our algorithm searches backwards stopping with the snapshot where the edge is just about to enter FF25, which occurs at FPA 244. The corresponding LCI of 4795 ps gives the uncalibrated delay. In order to obtain the actual path delay, the delays for FF16 through FF24 need to be subtracted. This component of the path delay is referred to as T dc in Eq. 1. It is possible to obtain the delay of FF21 through FF24 using the snapshots for this path as described in Section 4.4. However, the delays for FF16 through FF20 must be obtained using path delay tests that drive insertion points to the left of FF15 in this row. If tests which allow these FFs to be timed do not exist, then the actual path delay cannot be determined.

The LCI chosen for the path delay, given as 4795 ps in the example, is used to obtain an estimate of the individual FF delays to be subtracted. This is accomplished by 'looking up' the FF delay on the 'smoothed' delay curve (which is reproduced from Fig. 4.5 in Fig. 4.7) and then adding or subtracting an offset to account for process variation effects. This offset is computed for each of the individual FF curves by shifting the smoothed curve vertically until the area difference between the individual curve (which may have only a few points) and the average curve is minimized.

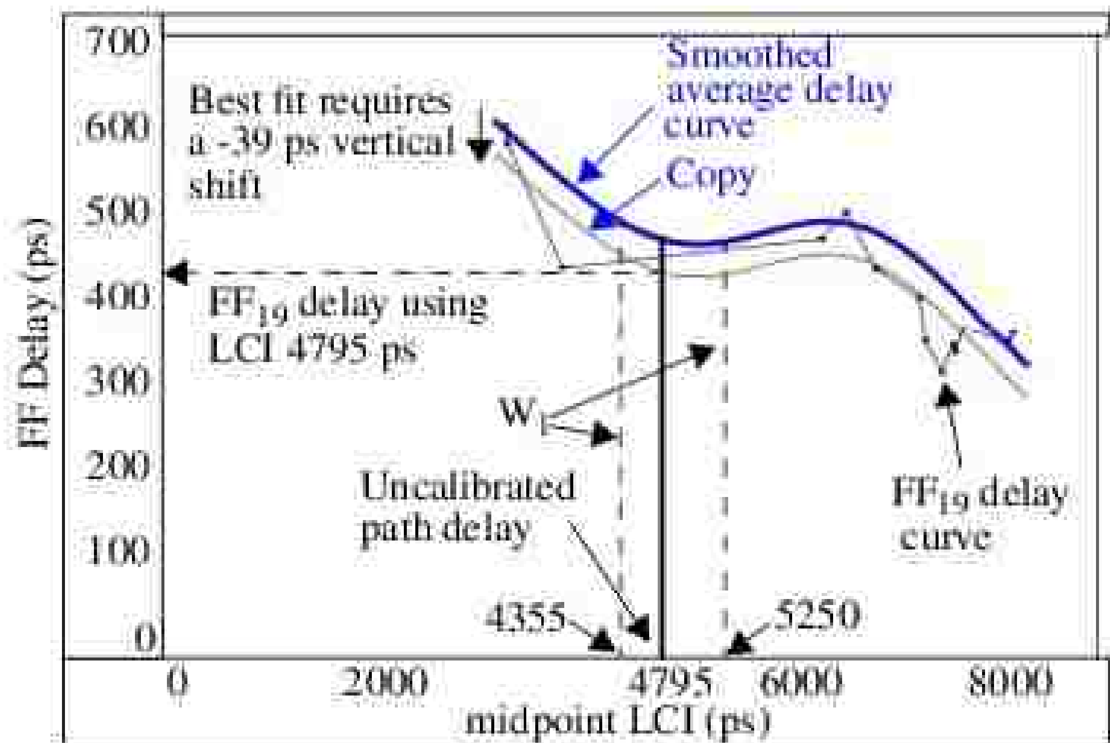


***Fig. 4.7: Illustration of calibration operation carried out using an individual FF delay curve and smoothed average delay curve***

The result of applying this process is shown in Fig. 4.7 for a FF19, one of the FFs in the delay chain of the path from Fig. 4.6. A copy of the smoothed average delay curve, labeled 'Copy' is shown shifted downwards by 39 ps. This vertical offset is the 'best fit' in the sense of minimizing the area between the Copy and the FF19 delay curve. The FF

delay is simply the y-value associated with the intersection of the line labeled 4795 (the uncalibrated delay) and the shifted copy, which is given as 436 ps. Note that this process not only provides an accurate delay through the FF for a specific LCI, it also allows the FF delay to be predicted for LCI's that it was not tested with, i.e., the smoothed curve fills in the gaps in the individual curves.

A similar process is applied to obtain the delay through the remaining FFs in the delay chain of the example path. The results are shown in Fig. 4.8 which lists the individual delays of the delay chain FFs, FF16 through FF24. The delay chain delay of 4005 ps is over 5 times larger than the actual calibrated path delay of 790 ps, and therefore, it is important to maximize the accuracy of the calibration process to keep the error in the path delay estimate low. Error analysis is covered in coming sections.

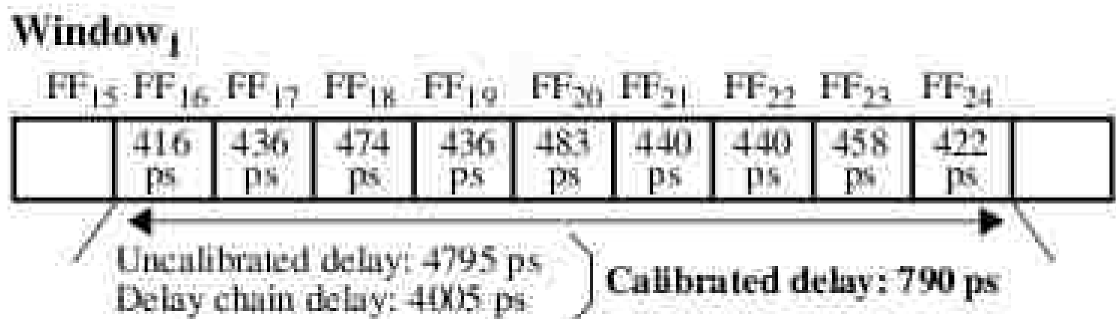


***Fig. 4.8: Delay calibration applied to example path from Fig. 4.6 using window $w_1$***

A key take-away from this section is that if techniques such as clock strobing are used to measure path delays, they must take into account the impact of the changing power transient and its corresponding effect on path delays. Although the impact of the power transient may be more significant in our test chip than in a typical commercial design, it is impossible to completely eliminate the power transient effects, particularly when faster- than-at-speed LCIs are applied1. Also, although the process described here

is difficult to apply in a manufacturing test context because of the overhead in post-processing the digital snapshots, it is practical for DFM purposes where the goal is to measure and characterize within-die delay variations in actual product macros.

## 4.7 Experimental Setup for FPGA Boards

The within-die variation is studied on the FPGA having reconfigurable logic. Delay based variation is studied on ZYNQ FPGA on ZED boards with AES (Fig. 4.9) as macro under test to study within-die and die-to-die variation among 28 copies of chips. I measure the path delays as discussed in the section 4.9 on path delay measurement using target flip-flops. The bit stream is loaded on all set of copies and is kept the same to understand the underlying variation from one chip-under-test to another.

The input test patterns are comprised of 100-150 test vectors with 256 insertions points on the AES engine. A set of 16 Flip-flops is connected to the scan chain of 256 Mux-D Flip-flops to let the transition from the right most insertion point to propagate in much further during the launch capture interval. A Linux- based host computer runs a custom LABVIEW application that controls the testing and data collection process through GPIB, Ethernet interfaces. For each chip data for 10,000 stable paths are collected, for this the required paths to be tested is 25,000.

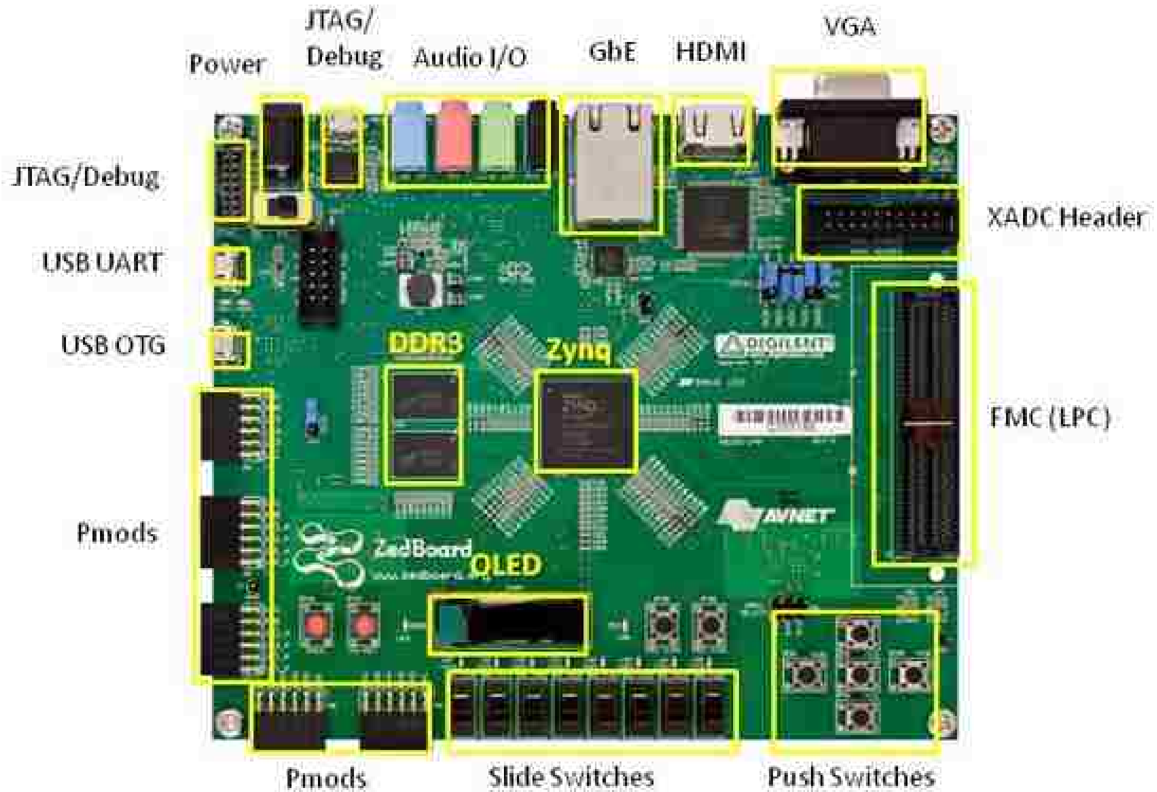


*Fig.4.9 Zed Board with Zynq FPGA*

## 4.8 Launch-Capture Clocking Sequence and Clock Strobing

### 4.8.1 Launch-Capture Clocking Sequence

The launch-capture clock sequence is generated using a digital clock manager (DCM) on a Zynq FPGA. The fine phase adjust (FPA) feature on the DCM allows the LCI to be set with a resolution of 17.86ps. A specific FPA is configured into the DCM by a state machine running on the FPGA which accepts an integer input parameter from the controlling LABVIEW application. Valid values of the FPA are between 300 FPA and 720 FPA with the step size of 2 FPA, which corresponds to a programmed LCI between 5ns to 12.000 ns.

REBEL extends the length of the path-under-test (PUT) at the *insertion*

*point* by creating a horizontal delay chain Transitions are launched into the MUT using Clk1. Any transition(s) occurring at the insertion point propagate down the delay chain, are captured as a **digital snapshot** by asserting Clk2. Fig. 4.10 shows clock strobing in FPGA implementation.

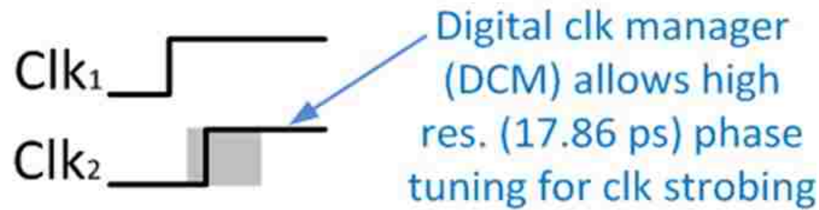


***Fig. 4.10 Clock Strobing in FPGA implementation***

### 4.8.2 Clock Strobing

Clock strobing involves repeating the test sequence at varying longer LCIs. The RCL and front-end logic for REBEL allow critical timing events, i.e., the launch-capture interval (LCI), to be controlled by the system clock. In our experiments, we apply a sequence of LCI tests over the range of FPGAs between 128 to 1120 in FPA increments of 2. This results in the application of (1120-128)/2+1 = 497 LCI tests with actual LCIs between 25 and 20 ns.

## 4.9 Delay Measurement Process

A random 2-pattern test is generated using an LFSR and the MUT outputs are tested, one at a time, by adjusting the insertion point. Once all outputs are tested, another 2-pattern test is generated and the process repeats. A sequence of launch-capture tests are applied, each with a different fine-phase-adjust (**FPA**), until the transition is 'pushed back' into a target FF, e.g., FF21.

The target FF is always at a fixed distance from the insertion point. The path delay is represented by the FPA at the point in the sequence when this goal is achieved (highlighted in red as 164 in the Fig. 4.11).

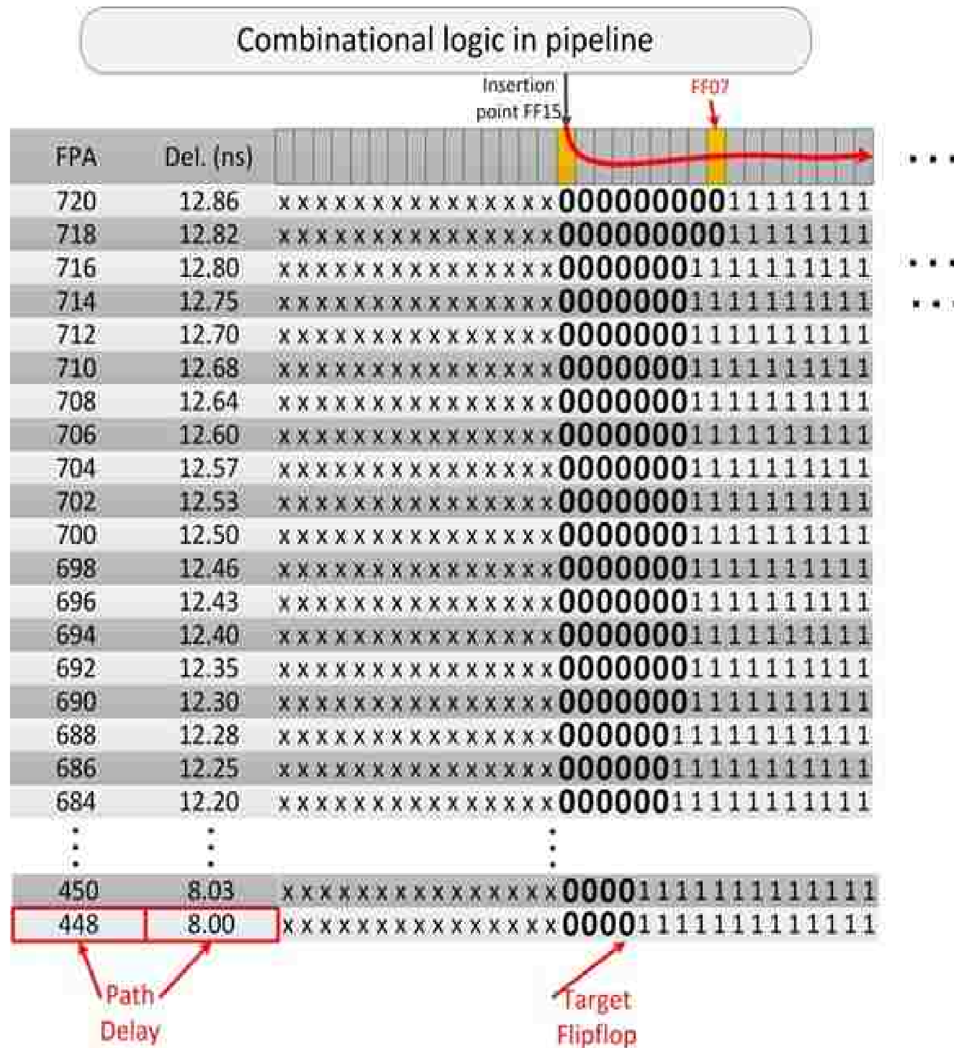


*Fig. 4.11 Digital snapshot produced from a path delay test*

A stable path is defined as a path that has a single transition over the entire sequence of snapshots. All the paths that are not stable are rejected as a means of improving reliability. The set of stable paths is unique across all the boards. The paths that are classified as stable vary from one chip to the next Xilinx Zynq experiments show approx. 5% of the paths from each chip are unique.

From the experiments we analyze all the paths and determine glitch paths, with more than one transition in a given path, and the stable paths, with a single transition in a single snapshot. Short paths are timed, and level of within-die variation that exists in these paths are measured. In the design the target FF are varied to understand the underlying variation.

## 4.10 Measuring and Calibrating Path Delays

The Launch-capture interval (LCI) for which the transition enters the target flip flop is considered as a the path delay of the given path with fixed insertion point. Fig. 4.12 shows REBEL integration with AES engine.

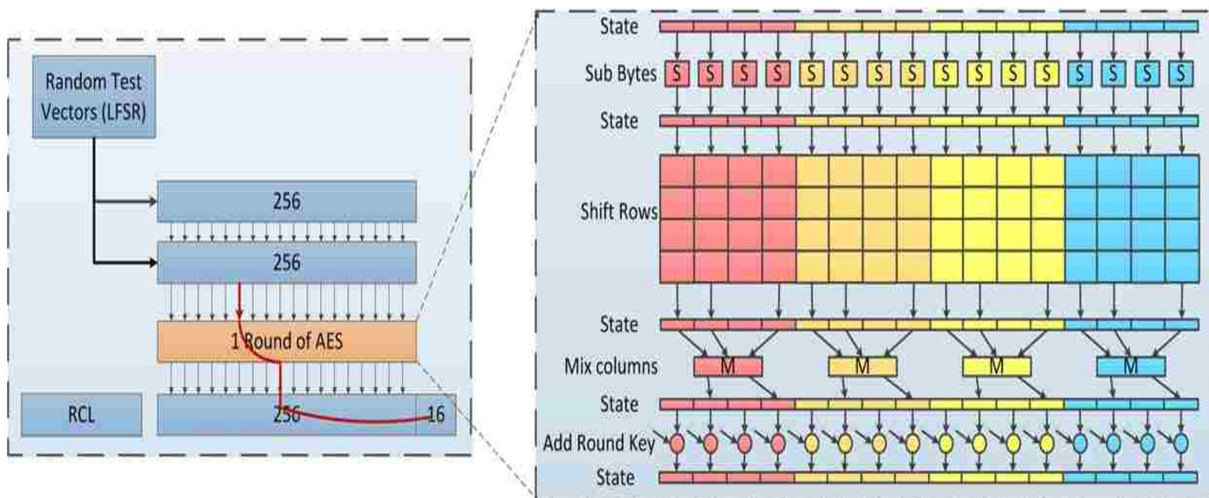


*Fig. 4.12 REBEL Integration with AES Engine*

The combinational logic is implemented using a single round of an AES unit Xilinx PlanAhead is used to create an embedded design. AES and REBEL are implemented in the *programmable logic portion* (PL) of the Zynq SoC and the communications with the host computer is done through Ethernet TCPIP.

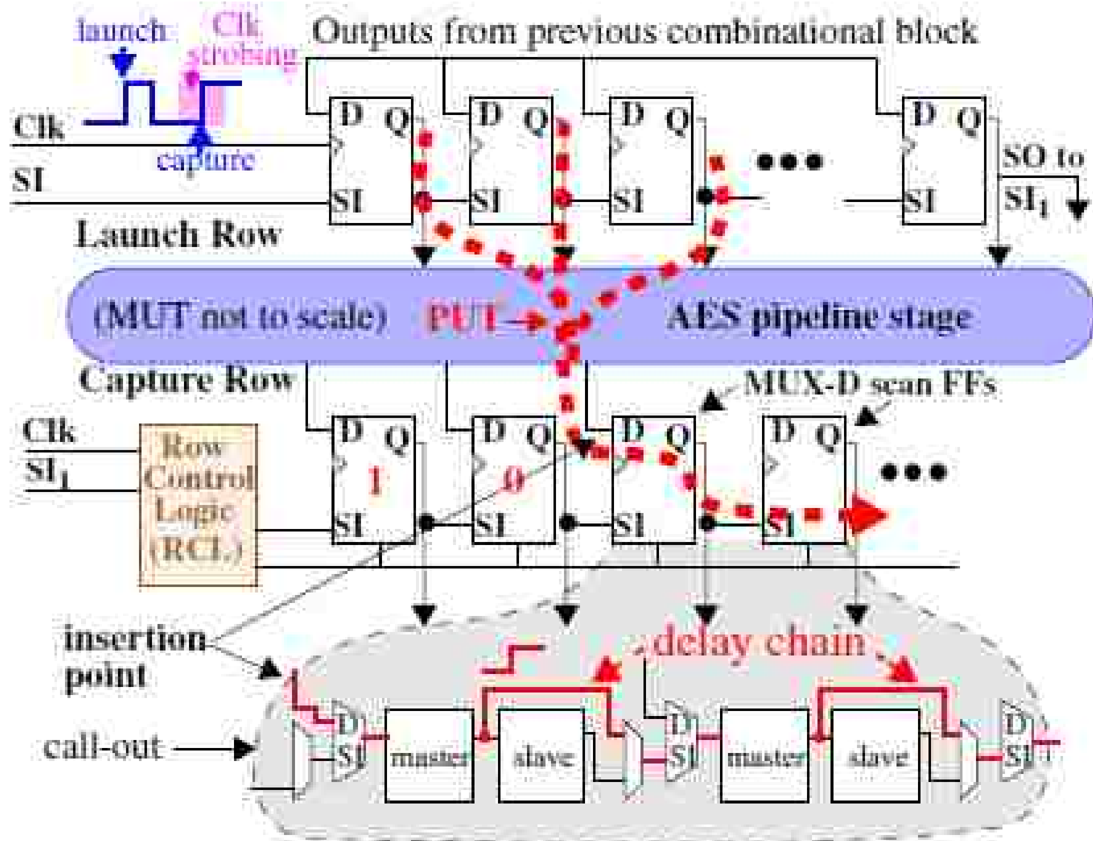


*Fig. 4.13: Block level diagram of AES with REBEL Integration*

Figure 4.13 shows the overall system design implemented on the FPGAs. As discussed the Mux-D scan flip flops requires only one clock for the launch and capture, where firstly the transition is launched on the first rising edge(launch edge) and its propagation in the delay chain is halted on the second rising edge(capture edge) and later the snapshot is scanned out from the delay chain to time the path delays through the combinational logic. The clock of 50Mhz is provided using DCM on board.

# CHAPTER 5
# ASIC Experimental Results and Analysis

We implement an embedded test structure called REBEL (Regional dELay BEhavior), designed to measure path delays accurately in a minimally invasive fashion, in a 90 nm test chip and present results on within-die path delay variations in a floating-point unit (FPU) fabricated in IBM's 90 nm technology, with 5 pipeline stages, used as a test vehicle in chip experiments carried out at nine different temperature/voltage (TV) corners. Data collected from the experiments is subjected to a variety of analysis and are presented in detail in this chapter.

## 5.1 Flip-Flop Analysis

During the path delay calibration the flip flop delays are subtracted from the launch capture interval to remove the delay chain component from the uncalibrated delay to get the delay of combinational logic. To improve the calibration process it is important to understand the variation of the delays in the flip flops. The delays of each flip flop in the delay chain vary from 10 FPA to 22 FPA that is from (10*36ps) 360ps to (22*36ps) 790ps for the falling edge transition and for the rising edge the range of variation is 12FPA to 26 FPA that is, 432ps to 950ps .

On average the delay of the rising edge and falling edge follows the same pattern

and the curve is shifted (Fig. 5.1). The shift is of 2 FPA making it of difference of 72ps. For the calibration process it is negligible hence both delays can be used to calibrate the path delays.

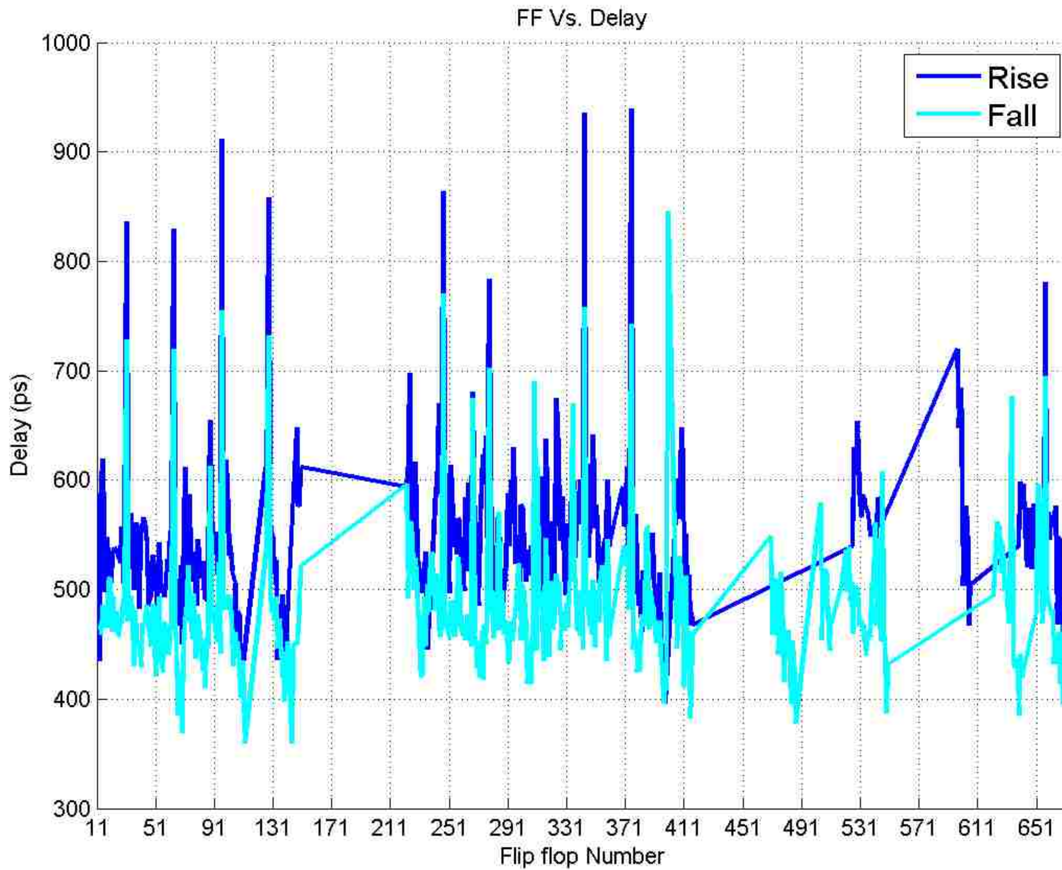


*Fig. 5.1 Flip Flop Delays for Falling and Rising Transitions*

The Power transient effect on the rising edge and falling edge curves follow the same pattren. From this analysis it is obvious that the delay of the rising edge propagation on average is 4 FPAs (144ps) more than the falling edge delays along all the mid point launch capture intervals. The power transient effect is shown in the smooth curve which is steep in case when the launch edge and capture edge are more closer and is seen on the left hand side of the curve (Fig. 5.2).

For the Rising edge on average the delay in window 1 is 17 FPA which is 612ps,

and for the falling edge the delay for the window 1 is 15 FPA that is equal to 540 ps. The delay in both cases is comparatively close with a difference of 72ps to 100ps. To remove the noise from the calibrated path delays, we subtract the rising edge transition delays of flip flops from the paths propagating rising edge and falling edge delays of flip flops for path delays with falling edge transition.

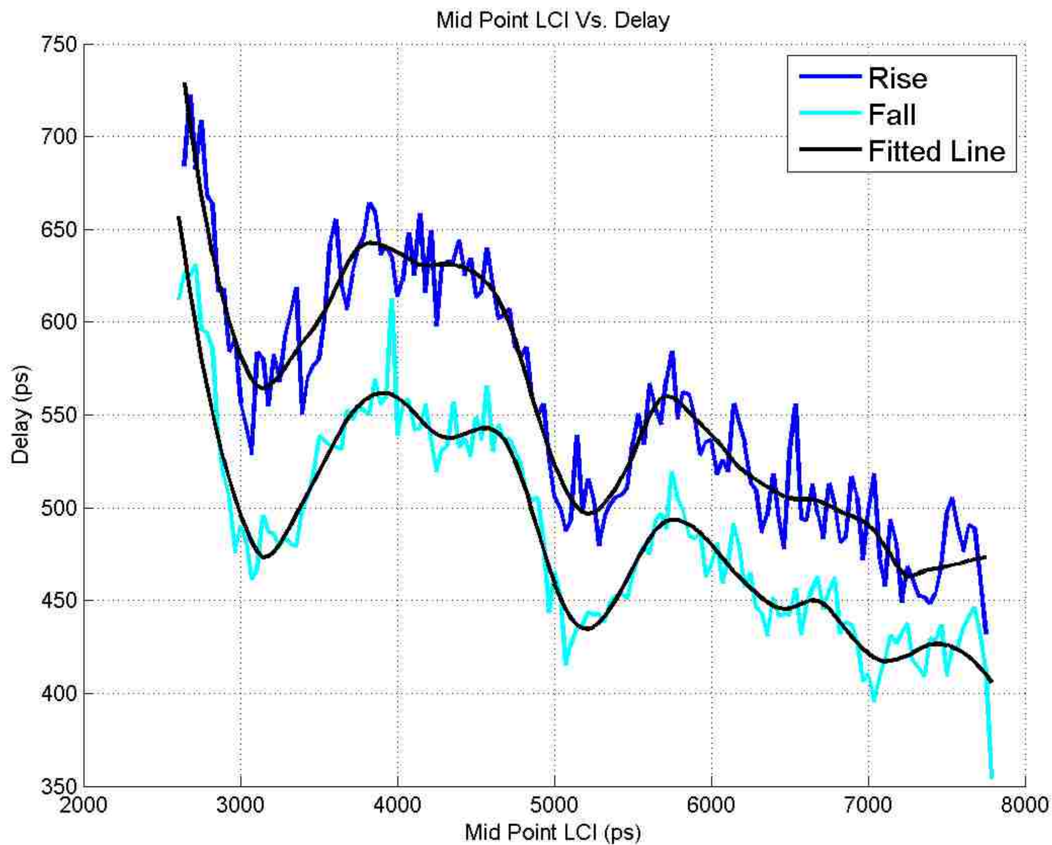


*Fig. 5.2 Mid Point LCI Average Delays for Rising and Falling Transitions*

The Flip flop analysis re-enforces the use of windowing mechanism as the fitted curve has a hump where the launch edge is closer to the capture edge producing power transient. Power transient can be seen in all delays of scan elements as in the Fig. 4.4, and can be seen that it is true for all the flip flop as the pattern of the fitted curve is the same for both rising and falling curves. This is shown in Fig. 5. 3 and Fig 5.4 for the

rising delays and falling delays.

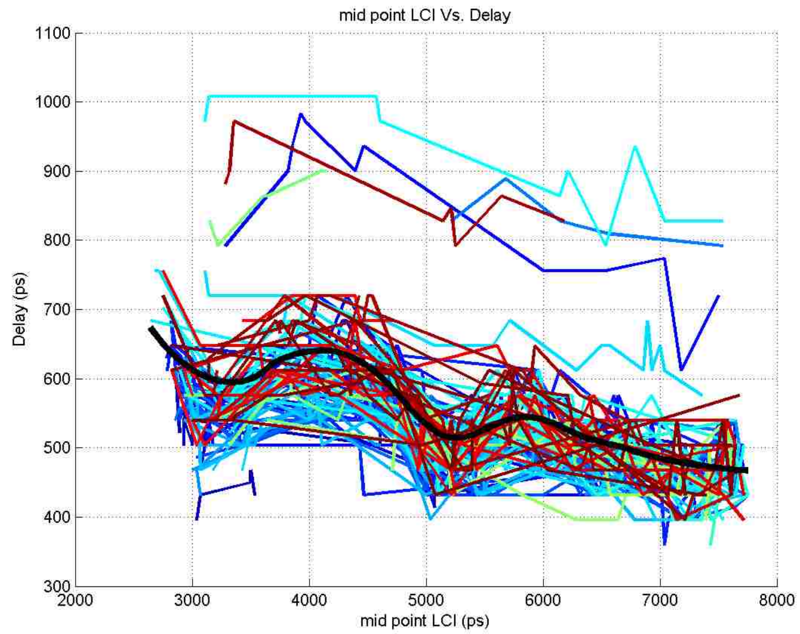


***Fig. 5.3: Propagational Delay of Rising Edge***

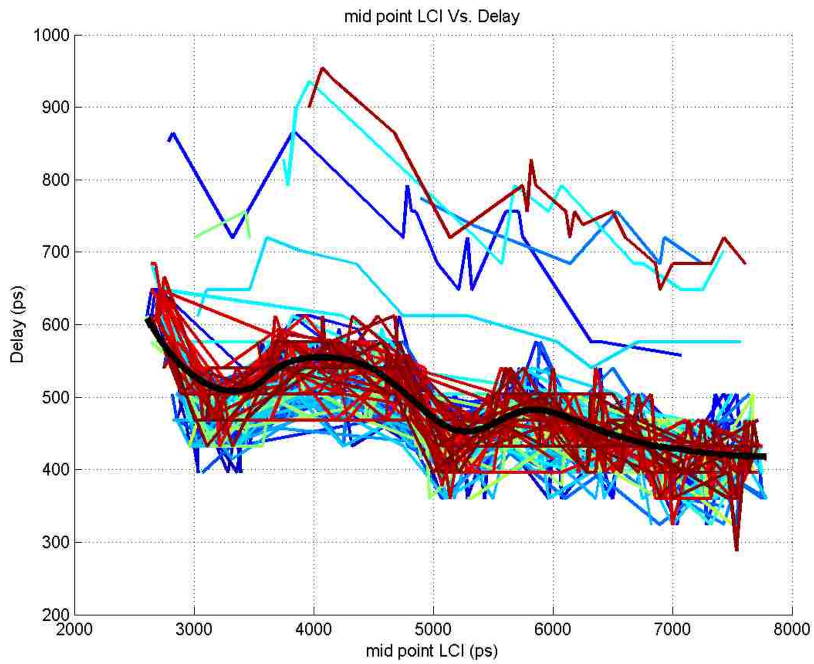


***Fig. 5.4: Propagational Delay of Falling Edge***

## 5.2 Uncertainty Analysis

Meta-stability of the FFs in the delay chain and jitter in the generation and distribution of the clock contribute to uncertainty or error in estimating path delays using REBEL. Uncertainty is captured and can be analyzed from the sequence of digital snapshots associated with a path test. The snapshots shown in Fig. 5.5 are copied (and modified to illustrate uncertainty) from a portion of the snapshots shown in Fig. 4.4.

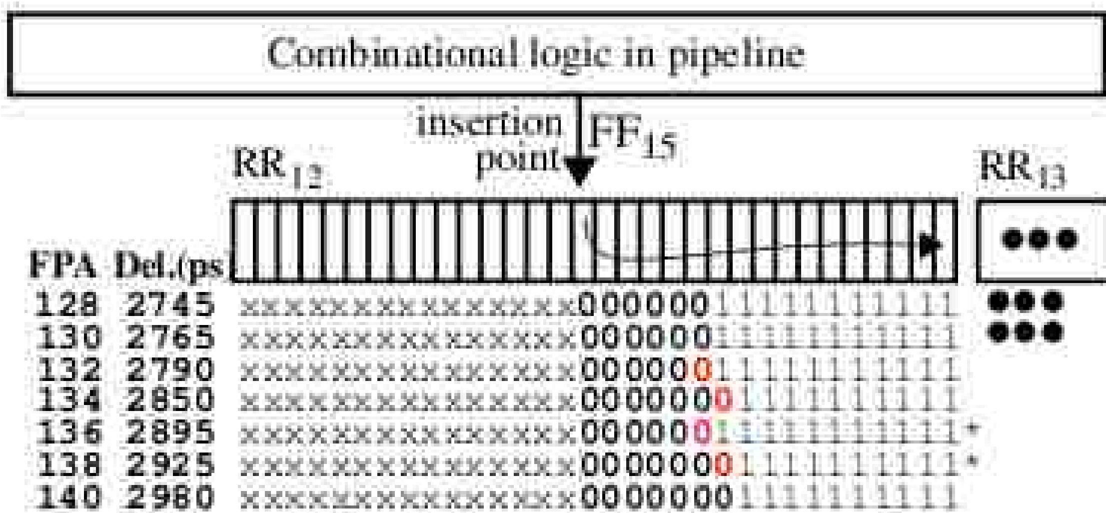


*Fig 5.5: Digital snapshots illustrating uncertainty using data given earlier in Fig 4.4*

Uncertainty, when it occurs, always appears near the transition points between FFs, e.g., between FF20 and FF21 at FPA 136 in this example. The snapshot for 134 shows the edge moving forward to FF 21 but the snapshot for 136 shows it reverting again to FF 20 before advancing to FF21 in the subsequent snapshots. We assign a magnitude of 71 ps (approx. 2 FPAs) to this uncertainty, which also represents the smallest possible measurable value of uncertainty. Larger values of uncertainty can occur when the 'jumping' back-and-forth occurs over a larger consecutive sequence of snapshots.

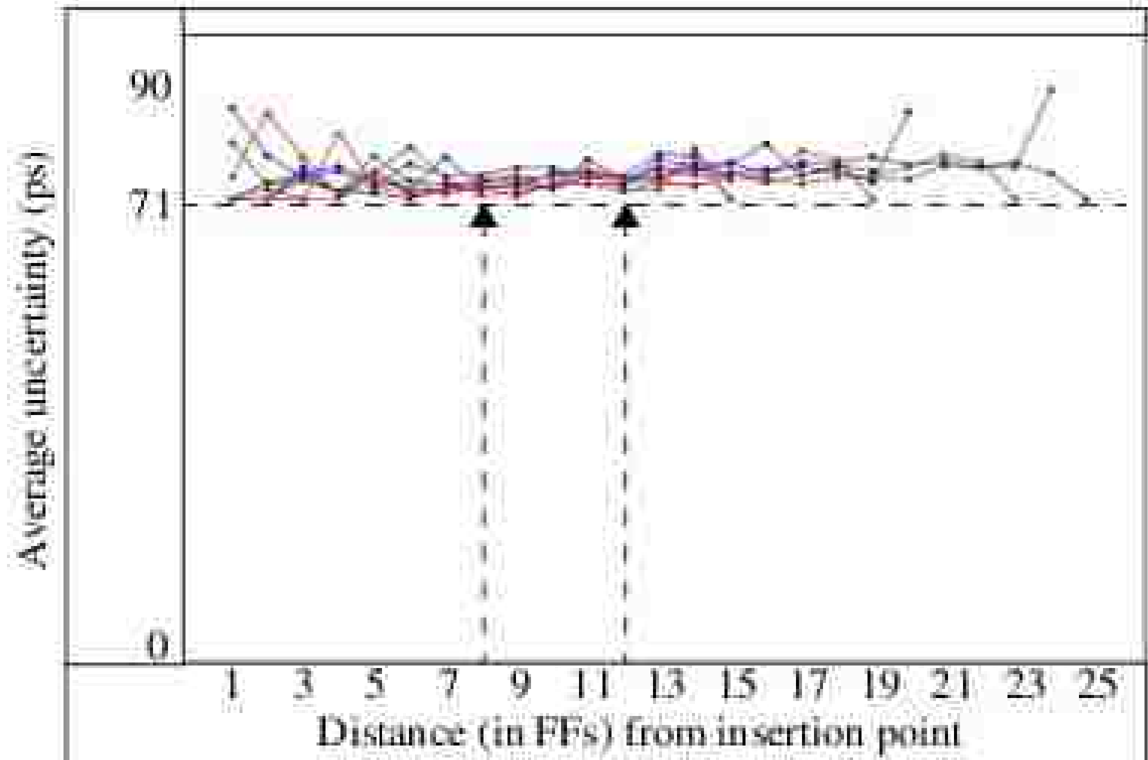


*Fig 5.6: Average Uncertainty as a function of distance (in FFs) from the insertion point for 4 chips at all TV corners*

Fig. 5.6 plots the average uncertainties on the y-axis computed using the 159 digital snapshots of all stable paths from the first 4 chips. One curve is plotted for each of the 9 TV corners to illustrate the impact of temperature and voltage on uncertainty. As indicated above, uncertainty is measured between FFs in the delay chain and we refer to these points as inter-FF transitions. The x- axis plots distance from the insertion point (in units of FFs) to further illustrate whether uncertainty increases as the edge propagates further down the delay chain. The 3 curves that extend to distances > 21 represent the 1.32 V TV corner data. The higher supply voltage allows for faster propagation and therefore more FFs are traversed.

Only inter-FF transitions that exhibit some level of uncertainty are included in the averages. Of the 10,000+ inter-FF transitions that occur per chip under the applied test

sequences, only about 5% exhibit uncertainty. From the curves, it is clear that uncertainty hovers slightly above the minimum of 71 ps. Uncertainty is smallest for distances between 8 and 12 FFs, and increases slightly at smaller and larger distances as shown on the left and right sides of the plot. Overall, uncertainty remains close to the minimum and is relatively insensitive to temperature and voltage variations. The worst case uncertainty that we observed in the 4 chips is 179 ps (approx. 5 FPAs) but these larger levels were very rare, occurring only twice in the data for these 4 chips.

## 5.3 Error in Estimating Path Delays

Our proposed calibration scheme also introduces error in the estimation of path delays. One way to evaluate this error is by shifting the position of the window described earlier in Section 4.5 and then re-calculating the path delays at this new window. For example, by shifting the original window (labeled W1 in Fig. 4.5) to the left by 500 ps (labeled W2), an additional FF (in most cases) is added to the delay chain of the paths. Since calibration removes the delay introduced by the delay chain, ideally, the W1 and W2 estimates of delay should remain the same for each path.

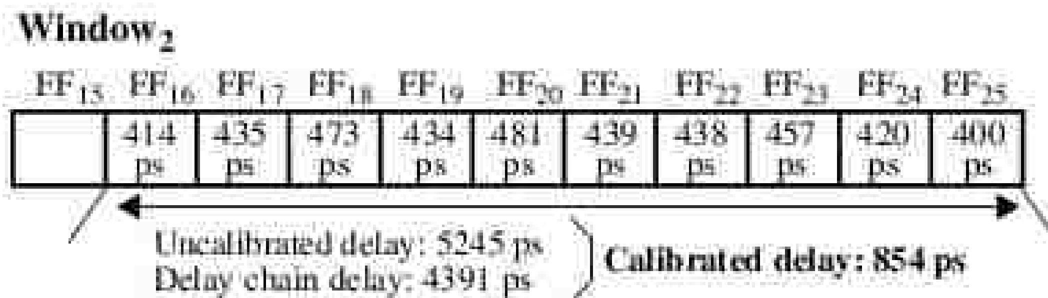


*Fig 5.7: Calibration process applied to the 2nd of two consecutive windows illustrating error in the estimation of path delay*

The differences in the estimates correspond to calibration error. Fig. 5.7 illustrates the calibration process carried out on the same path used in the W1 example of Fig. 4.8,

this time using the $W_2$ window. The un-calibrated delays, given as 4795 ps for W1 and 5245 ps for $W_2$ reflect the additional FF, i.e., FF25, and its delay of 400 ps, that is included in the delay chain under the $W_2$ analysis. Also note that the FF delays are slightly different under the two windows because the FPAs used to time the two paths are different. This follows from the discussion in Section 4.5, which shows the power transient impact on path delay varies as a function of the FPA.

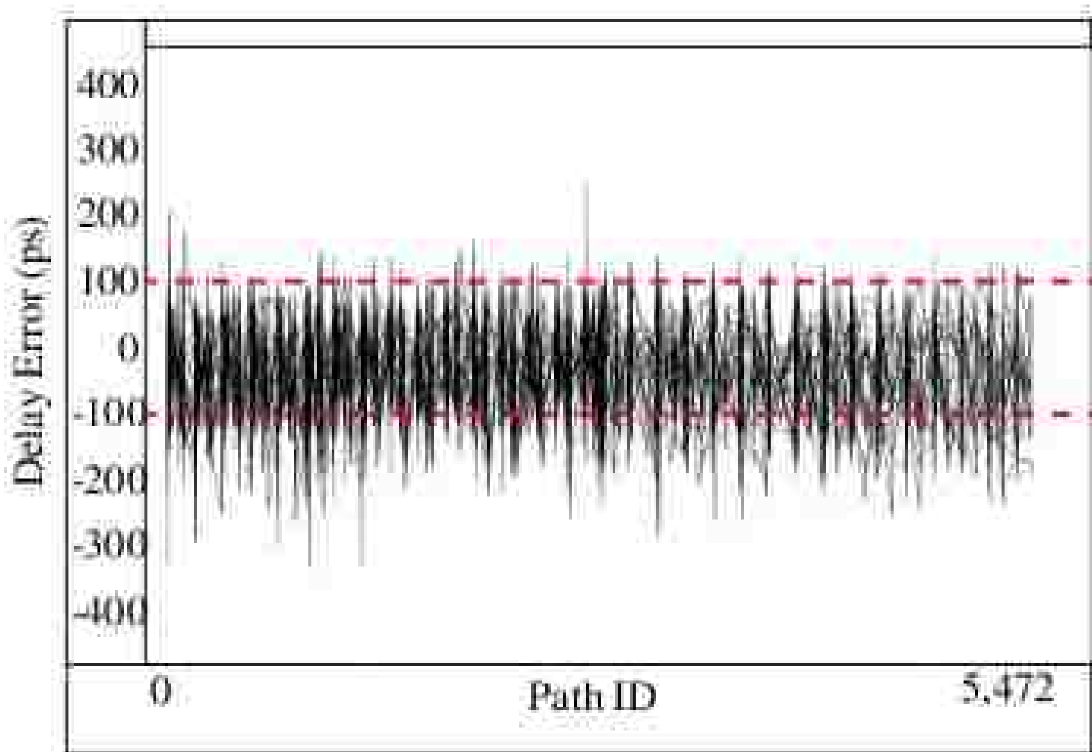


*Fig 5.8: Delay errors computed using $W_1$ and $W_2$ of the proposed calibration method for all paths and chips at 25°C, 1.2V*

Subtracting the sum of the FF delays from the uncalibrated delay in both cases yields calibrated delays of 790 ps and 854 ps for $W_1$ and $W_2$ resp. As indicated above, any difference in these estimates represents error, which is (790 - 854)= -64 ps in this example. Fig. 5.8 plots the errors for all stable paths in all chips at the 25°C, 1.20V TV corner. The path ID of the stable path is given along the x-axis. Although most errors are within +/- 100 ps, there are some that are larger, with worst case values of upto -300 ps.

The histograms plots in Fig. 5.9(a) and (b) partition the delay errors shown in Fig. 5.4 according to the length of the delay chain. As indicated earlier, transitions generated by short paths propagate through more elements of the delay chain, and therefore, their errors are captured in the right portion of the histograms.

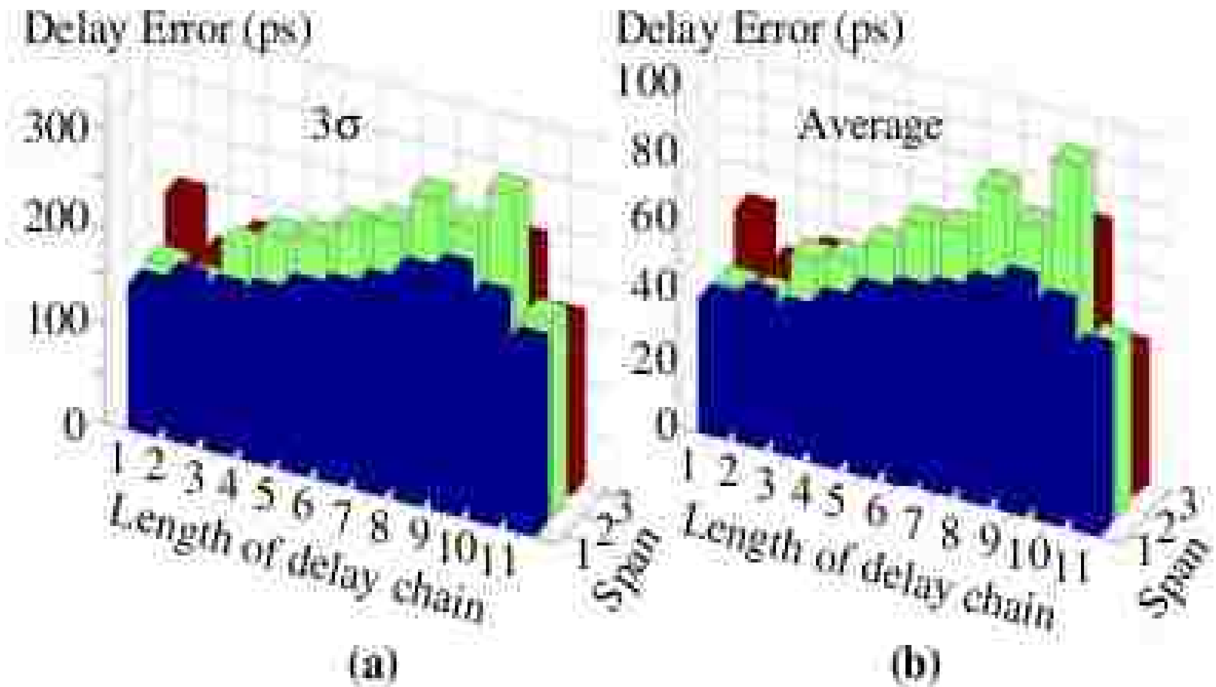


*Fig 5.9: 3σ (a) and average (b) delay errors for all paths and chips at 25°C, 1.2V as a function of 'length of the delay chain' (x-axis) and span (y-axis)*

Fig. 5.5(a) gives the 3σ of the errors while (b) gives the average errors. The results using the data from Fig. 5.9 is portrayed in the row labeled 'Span 1' (front row of values). As expected, the range of the errors (3σ) and average error both increase as the length of the delay chain increases. For example, the 3σ values increase from approx. 140 ps to 240 ps and the average error increases from approx. 38 ps to 66 ps.

This analysis gives the error when only one additional FF is added to the delay chain. Many of the delay chains are longer than 1 FF however, so it is conceivable that the error may be much larger for these paths. This is not the case, however, because the

positive and negative values of the errors act to cancel out, so the cumulative errors along the entire delay chain do not increase. This is illustrated in the results for the rows labeled Span 2 and 3.

Here, we extend the window by additional 500 ps increments, called W3 and W4 (see Fig. 4.5), and again use $W_1$ to compute the errors. Therefore, the $W_3$ experiments introduce 2 additional FFs to the delay chain of the paths (over that for $W_1$) while $W_4$ introduces 3. The term 'span' refers to this number of additional FFs in each of the 3 error analyses. Although the errors increase slightly in the Span 2 row, when compared with the Span 1 row, they actually decrease in the Span 3 row and are in fact similar to the values in row 1. Therefore, the overall error in the estimates of the path delays can be approx. using the errors given by the Span 1 values in the histograms.

In order to keep the contribution of error in our analyses small, we exclude those paths which have 'window errors' greater than 10% of the path length. Note that our error filter handles errors introduced by calibration discussed here and by uncertainty as described in Section 5.2. This is true because uncertainty occurs at the inter-FF boundaries, and therefore impacts the delay for a path independently under each of the two windows W1 and W2. A large uncertainty under either the W 1 or W2 analysis for a path will increase the difference in the error estimates and may cause the path to be excluded in cases where it is large.

## 5.4 Path Distribution Analysis

As indicated in Section 4.6, some of the individual FF delays were not measured under any of the path tests and therefore, it was not possible to compute the actual delays

for paths that use these FFs in their delay chains. In particular for CHIP 1, the total number of stable paths is 1,080, but was reduced to 825 because of this constraint.

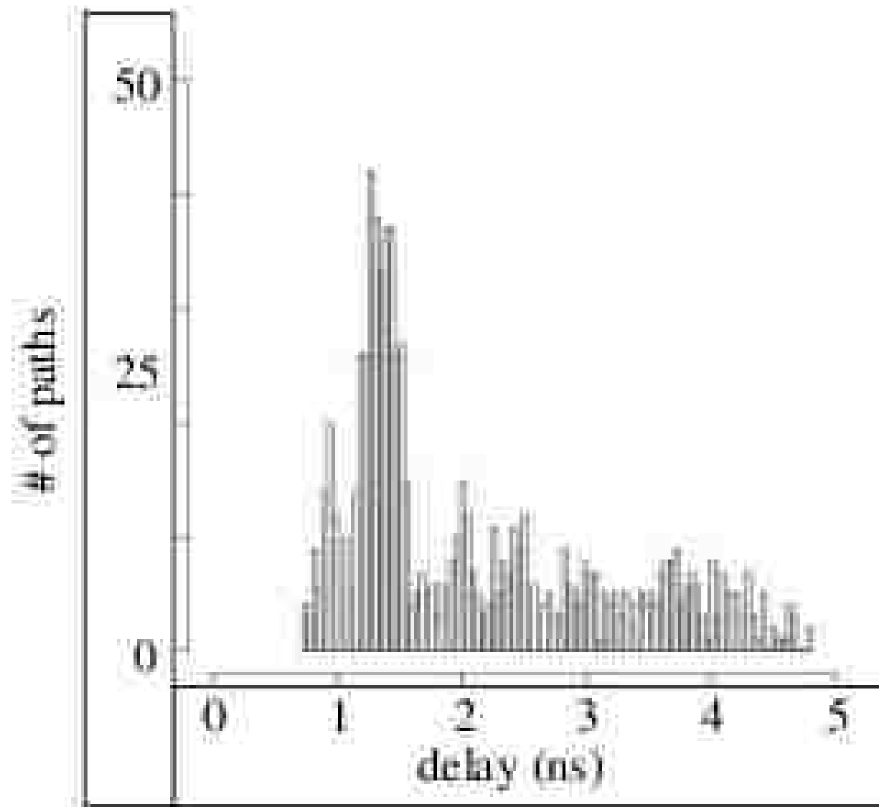


*Fig 5.10: Stable path distributions for CHIP1 at 25ºC,1.2V*

A histogram depicting the distribution of these 825 path delays for CHIP1 using data from the 25ºC, 1.20V TV corner is shown in Fig. 5.10. The x-axis plots the actual calibrated path delay against the number of instances on the y-axis. The distribution is relatively uniform except for the region between 1.0 and 1.5 ns, where the number of instances increases significantly. The longest measured path delay is approx. 4.75 ns which is consistent with the 200 MHz timing constraints used in the synthesis of the FPU.

## 5.5 Short vs. Long Path Variation

Only paths that stable and common across the set of chips are included in the analysis of short vs. long path variation (this is also true for the within-die variation analysis described in the Section 5.6). The 825 path identifiers (IDs) from CHIP1 are used as the reference when finding the set of common paths. Only 599 of these path IDs are found in all chips using data from 25ºC, 1.20V. The term ‘path’ used in the following sections refers ONLY to these common paths.

Path delay variation is expressed as a percentage change in this section, and is computed separately for each path. Chip-to- chip variation is eliminated by computing the mean delay across all paths i for a given chip j. The path delays for chip j are then normalized by dividing them by this mean as given by Eq. 2. Here, NPij is the “normalized path” delay for path i and chip j.

$$NP_{ij} = \frac{P_{ij}}{\mu_{chip\, j}} \qquad \text{Eq. 2.}$$

For each path i, the largest NPix, and smallest, NPiy, across all chips is used to determine the range, where x and y are two chip IDs from the set. The range is then divided by the mean delay, μNPi, computed across all chips as given by Eq. 3. Pchi rep-

$$Pch_i = \frac{largest(NP\, ix) - smallest(NP\, iy)}{\mu_{NPi}} \qquad \text{Eq. 3.}$$

resents the percentage change of a path i, and reflects the level of variation in this path as a function of its length, i.e., for a fixed range, a smaller average path length, μNPi, increases Pch. Pch expresses the differences in the level of variations for short paths vs. long paths.

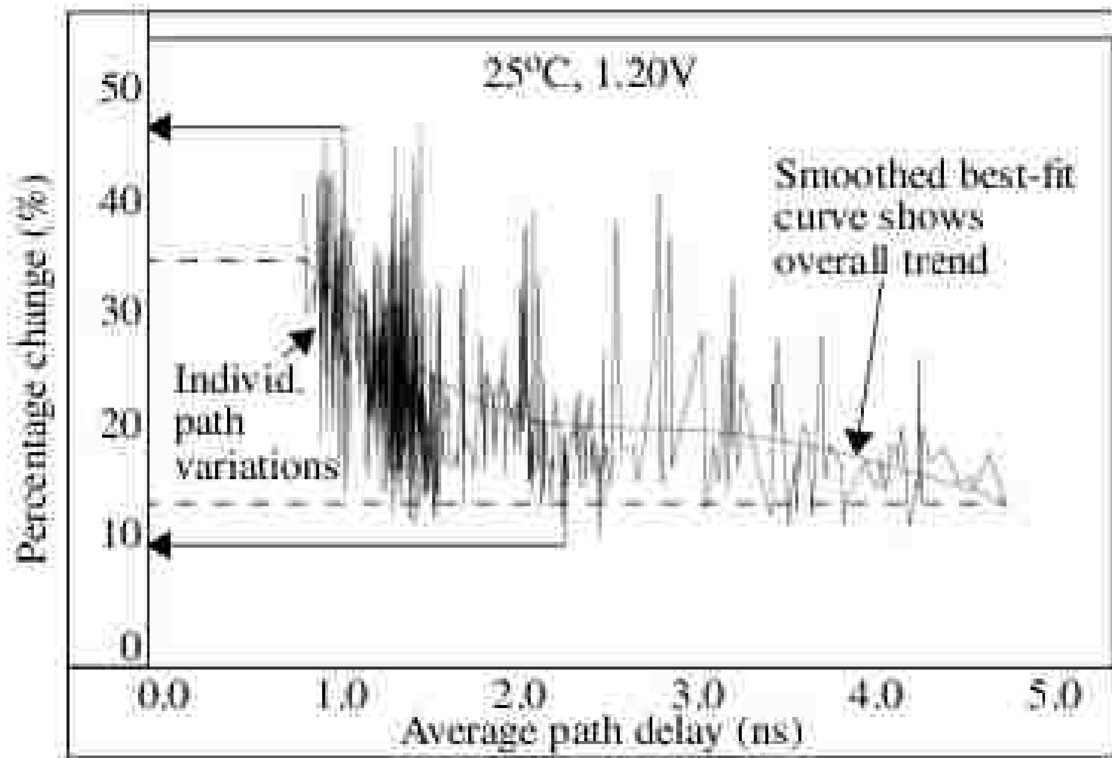


*Fig 5.11: Short vs. long path delay variation analysis at 25°C, 1.2V*

Fig. 5.11 plots the results with average path delay plotted along the x-axis against percentage change on the y-axis. Average path delay is computed using the original (un-normalized) delays, $P_{ij}$. The individual path delays vary by less than 10% to more than 45%.

A smoothed best-fit curve is superimposed on the individual path results to illustrate the overall trend. Paths longer than 2 ns vary between 12 and 20% on average while those less than 1 ns vary between 30 and 35% on average. The law of averaging works to keep the variation of longer paths smaller. However, the level of variation per gate is much larger, and is captured by the shorter paths.

## 5.6 Within-Die Delay Variation Analysis

Regression analysis is an effective technique for measuring and analyzing within-die variations. Linear regression is applied to scatter plots which are constructed from the delays of two separate paths, i.e., a path pairing. Fig. 5.12 plots 6 path pairing in a sequence of 6 scatter plots, illustrating variations that occur across the range of short (lower left) and long (upper right) path pairings.

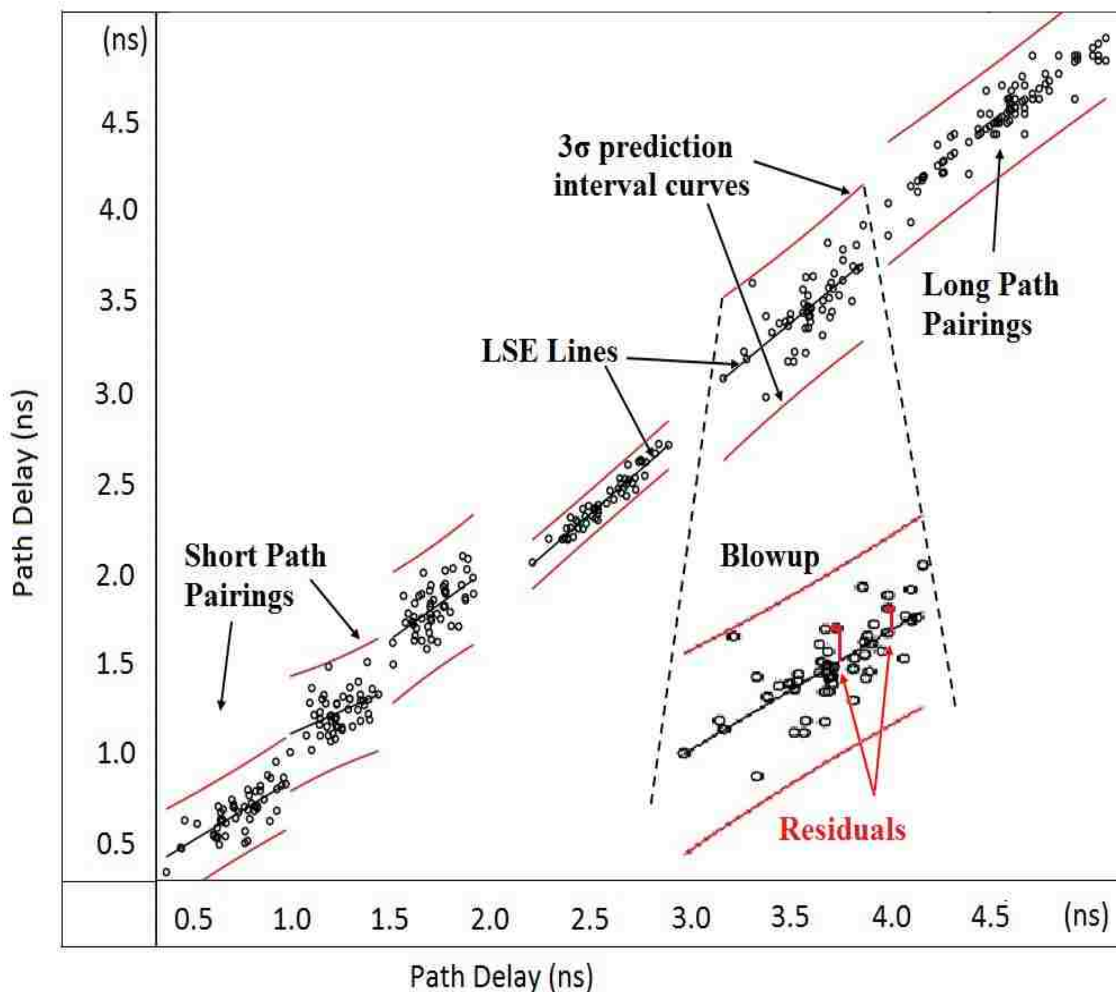


***Fig 5.12: Within-Die delay variation analysis using regression: example scatter plots from distributions of common to all chips***

Each data point in a given scatter plot represents the pair of path delays from one of the chips. As noted above, we exclude paths which have 'window errors' larger than

10%. Given that the delays from two paths define each data point, if either path for a given chip has an error > 10%, the data point is omitted. Also, path pairing that have fewer than 30 data points because of this constraint are excluded from this analysis. Therefore, all scatter plots include between 30 and 52 data points. We create path pairings by sorting the n delays from CHIP1 and then creating n-1 scatter plots by pairing data sets in the order given by the sorted delays from CHIP1. This ensures that the paths of each pairings have similar delays.

Linear regression analysis first computes a least squares estimate (LSE) of a best fit line through the data points of each scatter plot separately (see [32] for defining equations). Several of the 6 LSE lines are labeled in Fig. 5.12. The LSE line tracks chip-to-chip process variations. Within-die variations (and noise) are represented by the vertical offsets of the data points from the LSE line.

The vertical offsets are called 'residuals' (see the blow-up illustration in the figure). Three σ prediction interval curves are also derived for each scatter plot, and reflect the overall spread of the points around the LSE line. Given the prediction interval curves, which nicely portray within-die variations, are parabolic and difficult to use directly, we define a simpler and more robust metric to express within-die variations.

The 3σ of the residuals are first computed and then 'normalized' by the average path delay. The average path delay is the mean x-value from Fig. 5.12 for each of the scatter plots. The normalized 3σ are then multiplied by 100 to express them as percentage change. This metric scales the 3σ according to the length of the path, making comparisons of within-die variations between short and long paths more meaningful.

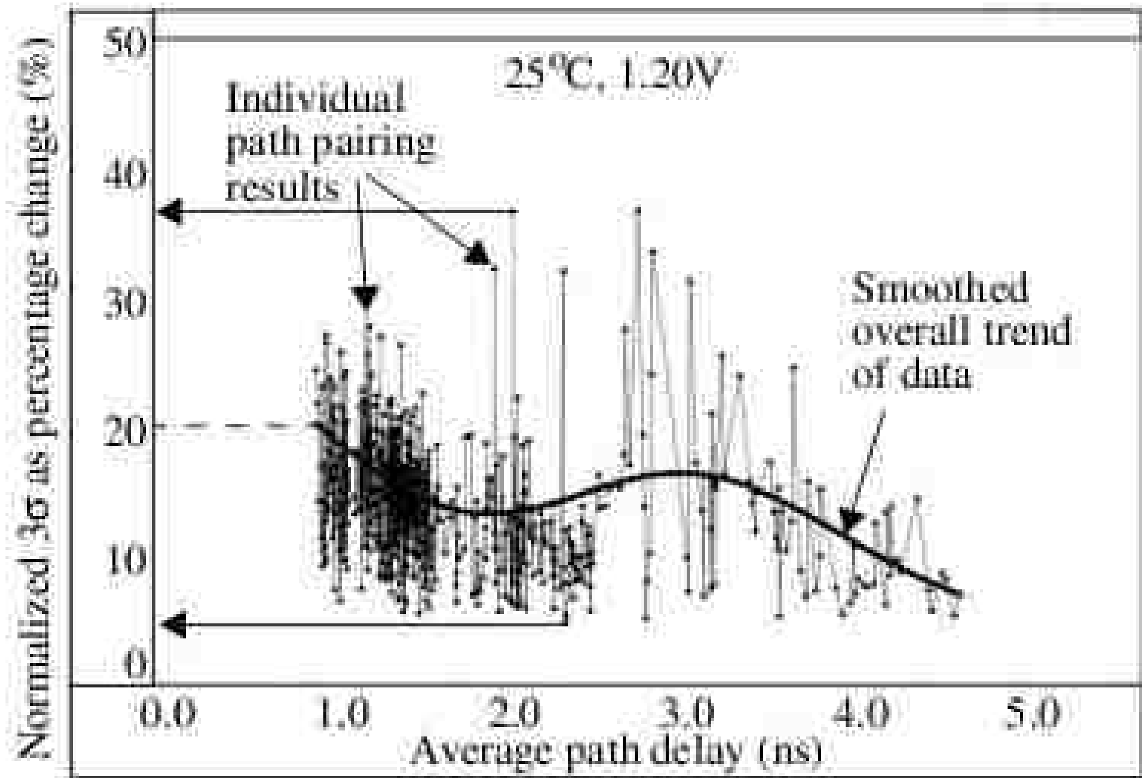


*Fig 5.13: Within-die variation analysis using regression. Average path delay vs. normalized 3σ of residuals expressed as percentage change*

The results obtained by applying regression analysis on the 25ºC, 1.20V data is shown in Fig. 5.13. The average path delay for each of the 551 path pairings is given along the x-axis, plotted against the normalized 3σ metric described above. A trend similar to that shown in the short vs. long path variation analysis of Section 5.5 (see Fig. 5.11) occurs here. The most significant difference is the larger peak in the regression analysis around 3.0 ns, which suggests that within-die variations are largest for median length paths. Interestingly, decreasing temperature appears to exacerbate within-die variations, particularly for the shorter paths, as shown on the left side of -40ºC and 85ºC overall trend curves in Fig. 5.14. Although not shown, the overall trend curve for 1.08V tracks the behavior of the -40ºC curve while the 1.32V is similar to the 85ºC

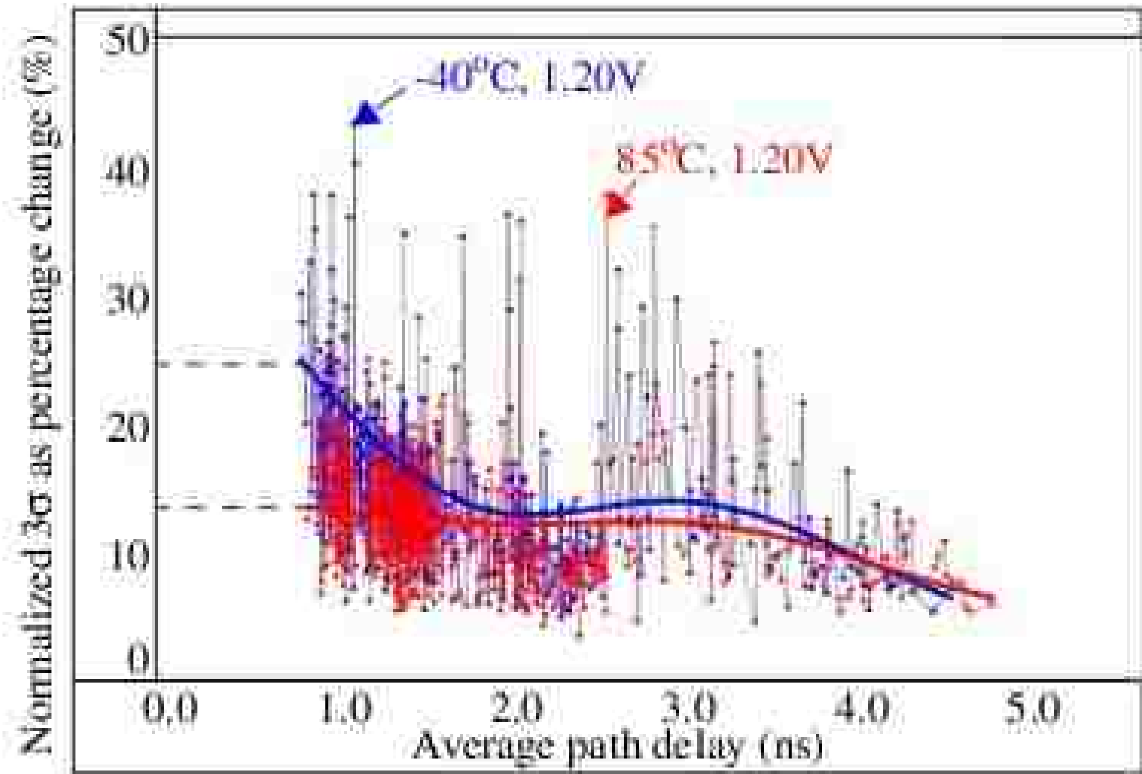


***Fig 5.14: Within-die variation analysis using regression using data from two additional TV corners***

# CHAPTER 6
# FPGA Experimental Results and Analysis

The DFM experiments on FPGAs are performed on 28 Zed Boards, with Zynq 7020 FPGAs. The design with REBEL and AES is synthesized on a clock of 50 MHz frequency. The DCM is used to do fine phase adjustment with the steps of 36 ps. A total of 720 steps are tested with Launch Capture Interval ranging from 1.5ns to 13ns.

## 6.1 Flip Flop Analysis

### 6.1.1 Propagational Delay tpHL and tpLH Analysis

Propagation delay is the delay from where input crosses 50%Vdd to when the output crosses 50%Vdd. Theoretically tpHL is a propagation delay when output switches from “High to Low” and tpLH is propagation delay when output switches from “Low to High” and, it depends on the input slew rate and output capacitive load.

This delay can be computed from the experimental data using the digital snapshots. Each path is tested with a range of launch capture intervals, which captures the temporal behavior of the transitions. Subtracting the time when transition enters the flip flop from the time transition leaves the flip flop is the propagation delay of the given flip flop. Ideally the design should yield the tpHL and tpLH equal but variation exists because

the basic devices nfet and pfet have different characteristics, for example the mobility of pfet is much lower than nfet. Usually the pfet is 1.5 to 2 times larger than the nfet to compensate this difference to meet the timing constraints and bringing the tpHL and tpLH delays approximately equal.

#### 6.1.1.1 Rising Edge Propagation Delay (tpLH):

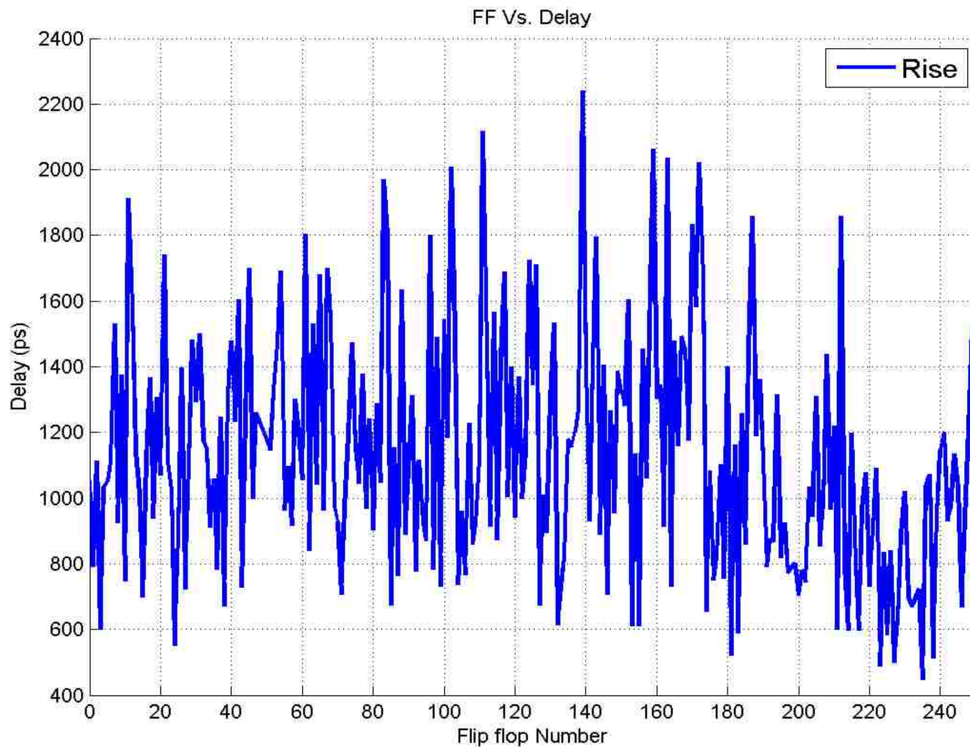


***Fig. 6.1: Rising Edge Propagational Delay***

Fig. 6.1 shows the rising edge propagation delay. The delay of each flip flop ranges from 17 FPA to 62FPA which is equal to (17*36ps) 600 ps to (62*36ps) 2.2 ns. This delay is higher because it includes the interconnect delay and three multiplexers added because of the scan chain logic. When this delay range is compared to the ASIC 90nm chip, the delay of ASIC chip is much less and is around 500ps.

The power transient effect is not seen in the data collected from the FPGA, the

variation in the flip flop delays is seen to be between 35 FPA to 45 FPA when the average delay for every launch capture interval is compared. The variation is not too high and thus can be ignored in the calibration of path delays.

#### 6.1.1.2 Falling Edge Propagation Delay (tpHL):

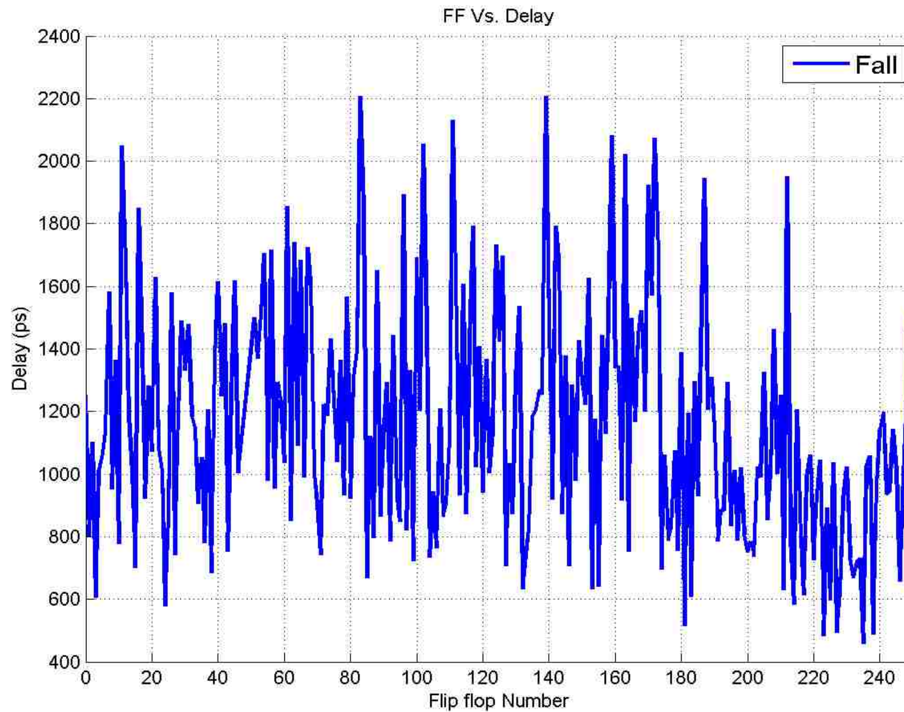


***Fig.6.2 Falling Edge Propagational Delay***

Fig. 6.2 shows the falling edge propagation delay. Similar to the rising edge propagation delay, the falling edge propagational delay of flip flops of the delay chain are ranging from 17 FPA to 62 FPA. From the delay curve plotted against the mid point launch capture intervals for the average delay of all the flip flops measured for each given midpoint interval reveals that on average the variation of flip flop delay is ranging from 25 FPA to 45 FPA which is about the same as the rising edge variation.

### 6.1.1.3 Comparative Analysis:

An interesting observation from the rising and falling edge propagation delay curves is that the range of delays of the flip flops of delay chain is same. The plots to observe the variation in the falling edge and rising edge delays (Fig. 6.3) reveals that the design of FPGA is very robust and the propagation delay of rising edge overlaps the propagational delay of falling edge. This analysis is performed on all 28 chips and same results are observed. The delay is in the range of approximately 20 FPA to 60 FPA that is from 600ps to 2.2 ns.

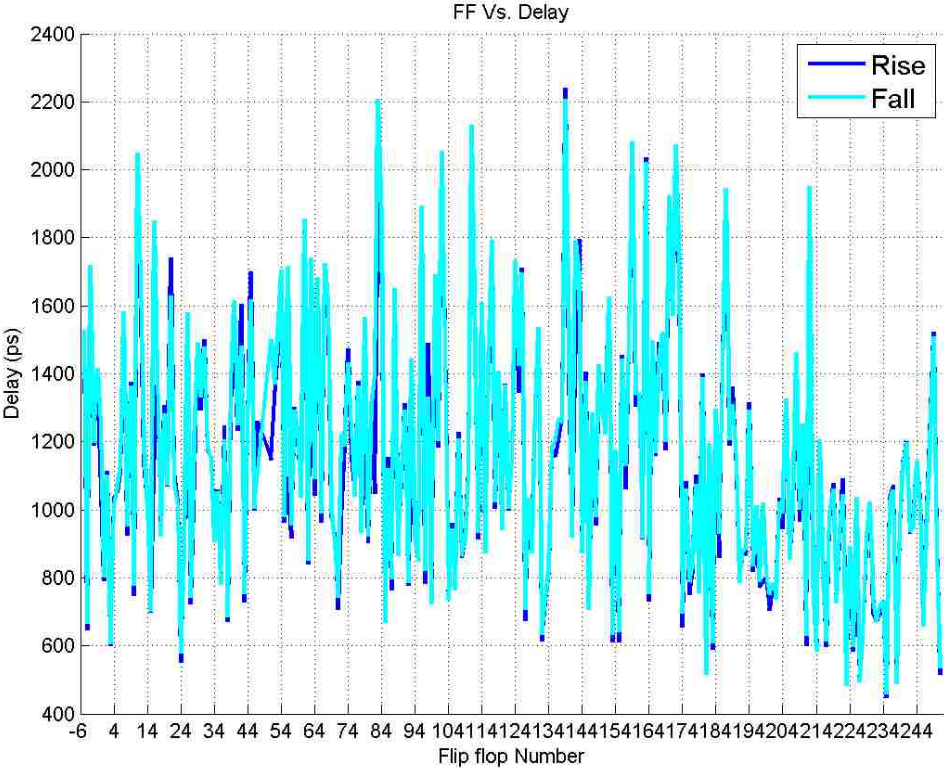


***Fig. 6.3: Rise versus Fall delays of Flip Flops***

### 6.1.2 Average Rising Edge Flip-Flop Delays

Fig. 6.4 shows rising edge flip-flop delays on different launch capture intervals. The Flip flops ranging from 20FPA to 65 FPA that is 600ps to 2,2ns. As the delay of each flip-flop has a smaller range the power transient effect is not seen in this data, but we do observe few noise points in the data. On average the delay is ranging from 20FPA to 35 FPAs and the smooth curve the following a straight line.

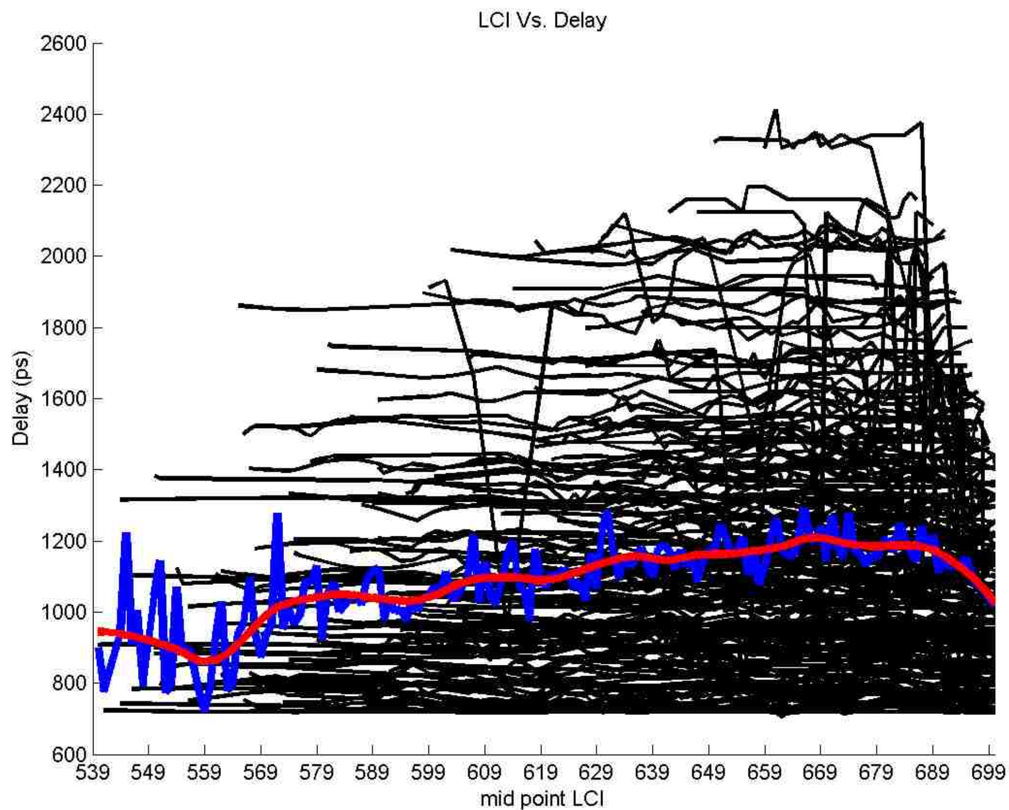


*Fig. 6.4 Average Rising Delays of Flip Flop*

### 6.1.3 Average Falling Edge Flip-Flop Delay

The falling edge flip-flop delays on different launch capture intervals are shown in Fig. 6.5. Similar results are observed in the falling edge. The range of variation on average is 15 FPA that is equal to 540ps. This range is of importance to understand and mitigate the noise points from the data set. This analysis helps us to incorporate the design parameter of range of variation allowed with a flip flop delay to calculate more accurately the die to die and within-Die variation.

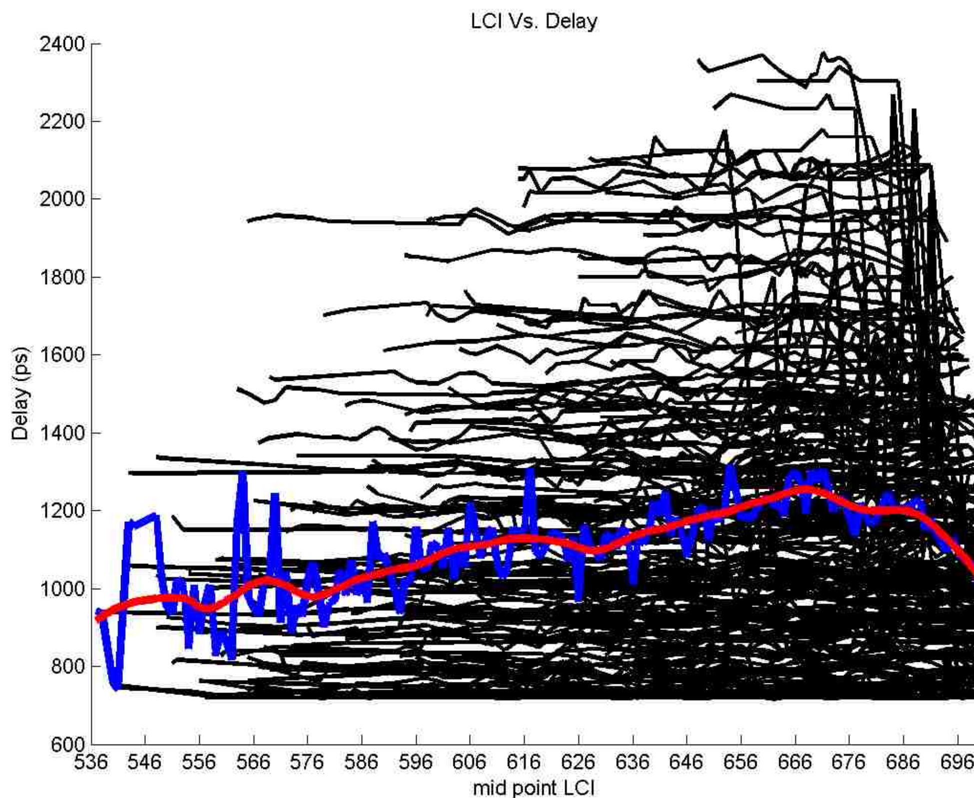


*Fig. 6.5: Average Falling Delays of Flip Flop*

## 6.2 Sample Analysis

A path is defined as a combination of a test vector and an insertion point and is equal to (test vector *256 + Insertion point - 255). Eight samples are collected for each tested path. This analysis helps in understanding the measurement noise, based on the

correlation of the results among all the samples.

For this analysis data from path with test number 1 and insertion point 255 is observed. The test number 1 yields a rising transition. The Figure 6.6 shows delay vs distance from insertion point for the flip-flops, whose delay can be measured from the given path, for 8 samples. The delay is in terms of FPA steps. For the given insertion point a set of flip flop delays can be measured by taking difference of timestamps for when the transition enters and when it leaves the flip-flop.

The sample analysis captures the measurement noise and interestingly we see that the delays are within the range of approximately 70ps. The variation of delays from one sample to another is shown in the following graphs (Fig. 6.6 – Fig. 6-10). Interesting observation is that the transition is in chip 1,5,7,10,11 and not in 2,3,4,6,8,9 making the transition to appear in 50% of the chips. This variation can be used as an entropy source for the PUF applications. For this plot the delay of each flip flop is not changing more

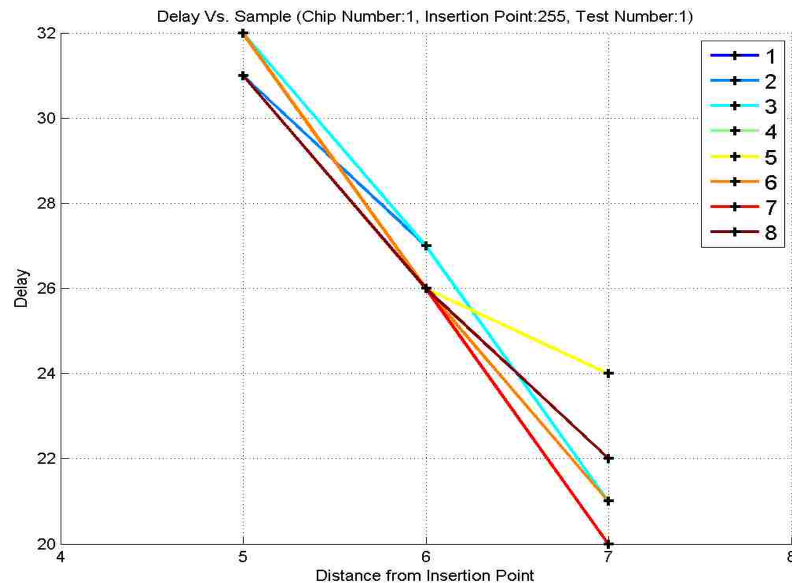


*Figure 6.6: Delay vs Sample analysis for Chip 1*

than 36 ps when the flip flop is closer to the insertion point where as the variation or noise is approximately doubled for the further most flip-flop. For the flip-flop with a distance of 5 flip-flops from the insertion point takes on average 31 FPA steps to propagate completely through the flip flop.

The data collected from Chip-under-test 5 (Fig. 6.7) shows more variation than the chip 1, where the transition has propagated further along to next flip flop. On average the delay is 70 ps which is within the same range as chip 1.

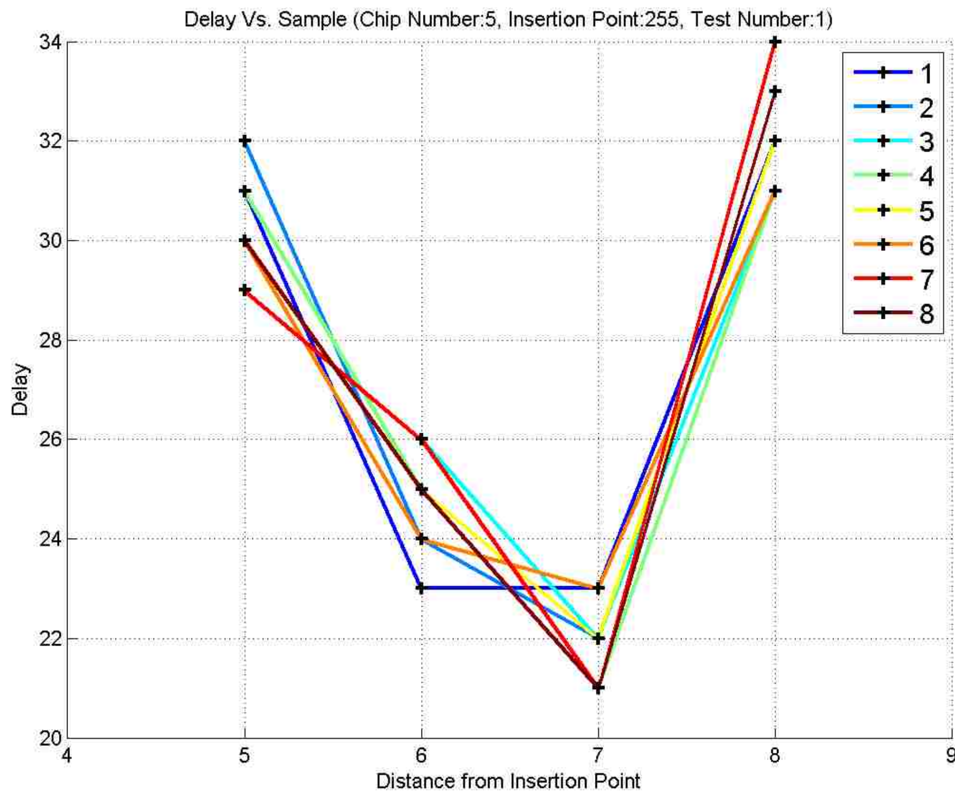


*Figure 6.7: Delay vs Sample analysis for Chip 5*

To better understand the variation within the samples we see uncertainty in sample 3 on Flip flip 6 of 2 FPA steps and in sample 7 for Flip flop 7 an uncertainty of 1 FPA step, shown in the Fig. 6.7.

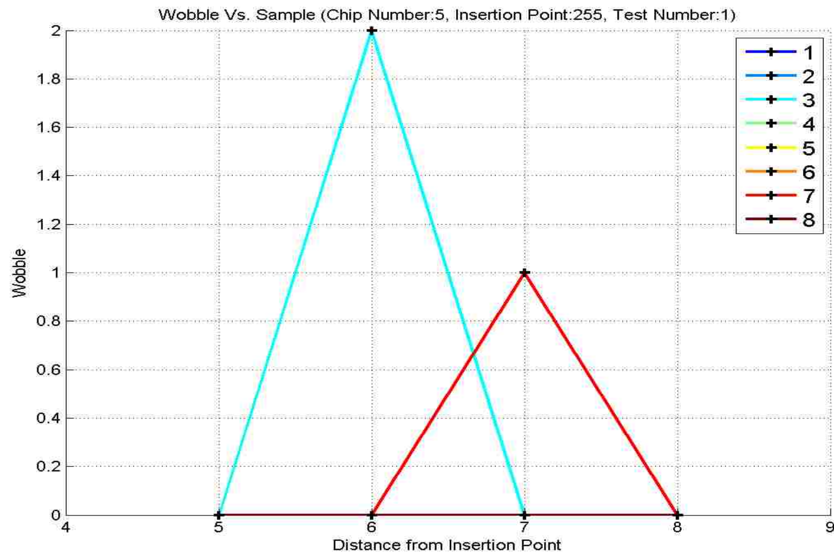


*Figure 6.8: Uncertainty analysis for Chip 5*

Similar analysis on Chip 7 yields that it is also a fast chip where the transition has propagated in the delay chain for 8 flip-flops, giving an average variation within all samples of 5 FPAs that is approximately 90ps

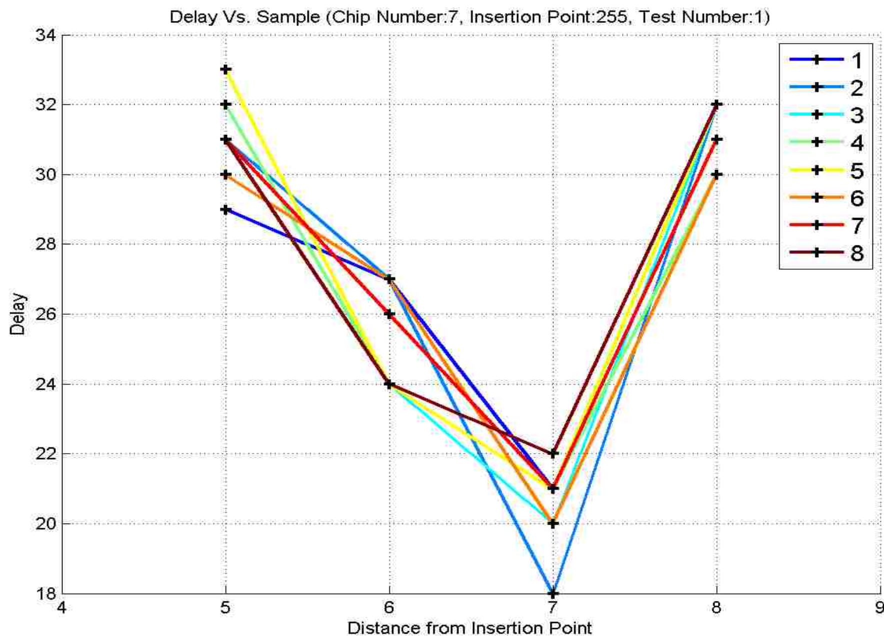


*Figure 6.9: Delay vs Sample analysis for Chip 7*

Figure 6.10 shows the uncertainty analysis, where wobbling count is shown against distance of transition propagated from the insertion point. The uncertainty is observed in Flip flop 6 and 8, in samples 8 and 2 respectively.

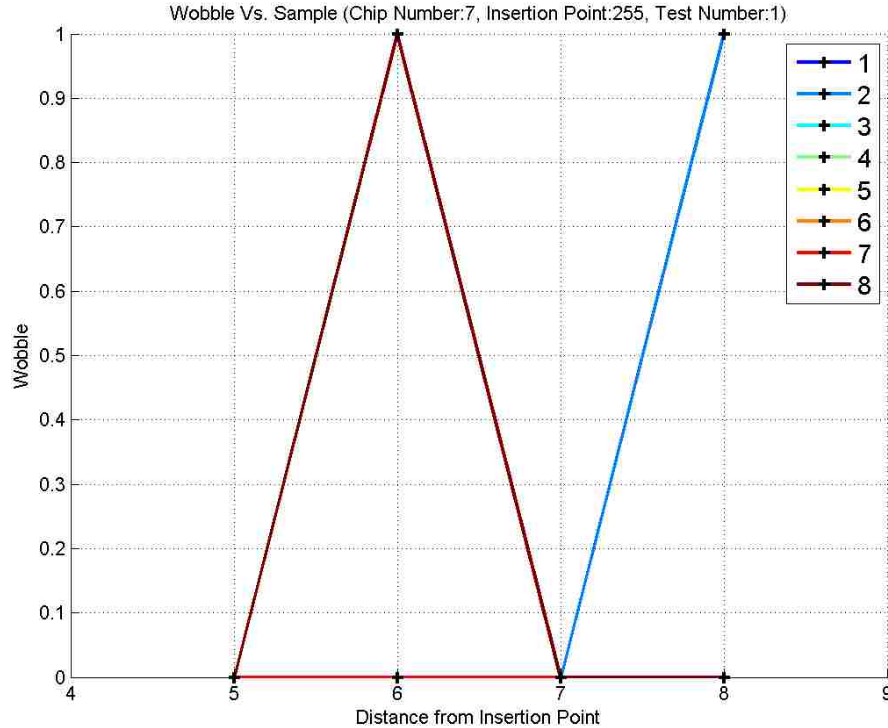


*Figure 6.10: Uncertainty analysis for Chip 7*

Similar results are seen from analysis on chip 10 and chip 11. For chip 10, the transition propagated completely to the 8$^{th}$ flip flop from the insertion point. The flip-flops closer to the insertion point have less variation within-samples (close to 30ps). On the other hand we observe more variation on the flip flop with a distance of 7 from the insertion point (about to be 90ps). Here we see an uncertainty of 1 FPA in sample 1 for Flip flop 6 and 1 FPA for flip flop 7 in sample 7. Chip 11 shows the transition only in two samples, and has no data for the other 6 samples. There is no uncertainty during this transition.

This analysis shows that the measurement noise is minimal and the variation in results

from 1 sample to another is not huge.

## 6.3 Uncertainty Analysis

To study the meta-stability of the FFs in the delay chain and jitter in the generation and distribution of the clock contribute to uncertainty or error in estimating path delays using REBEL in the Zynq FPGA. Uncertainty is captured and analyzed from the sequence of digital snapshots associated with a path test. For the uncertainty we observe the transition propagating from one flip flop to another as discussed earlier in section 5.2.

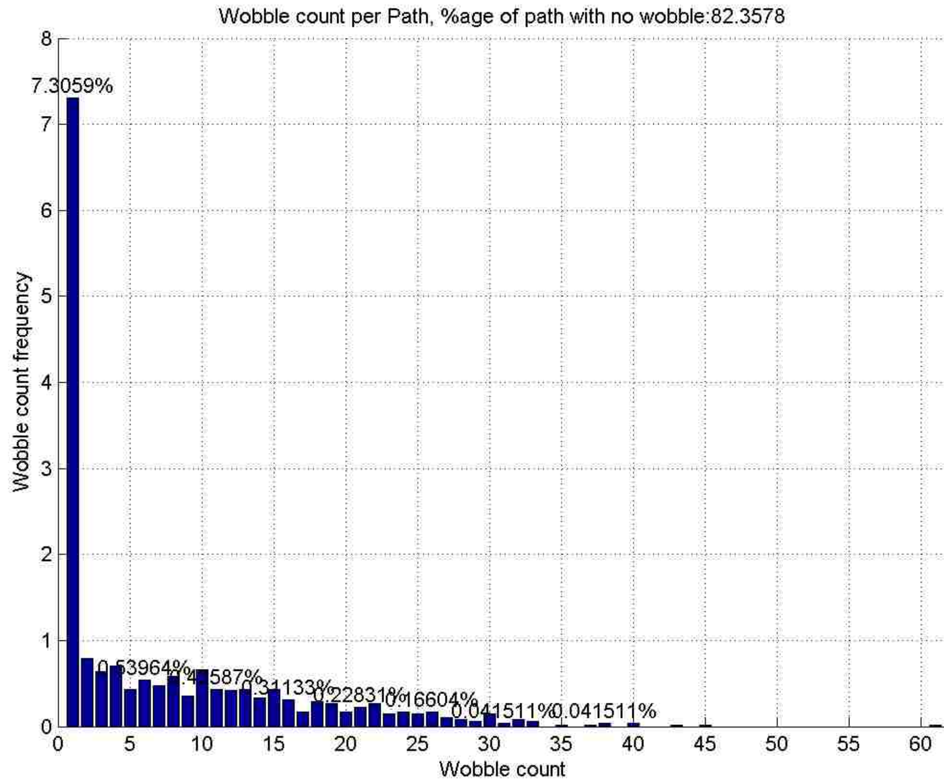


***Fig. 6.11 Over All Uncertainty in path delays***

The study on uncertainty which is defined as transition moving back in the previous flip flop when the more time is allowed to propagate. This kind of behavior is not allowed in as it introduces the measurement noise and directly affects the correctness

of the delays. This analysis shown in Fig 6.11 shows that this kind of unwanted behavior is shown in 20% of data and is most of the time only shown once in all the launch capture intervals tests performed on a given path-under-test. To remove the noise in the path and flip-flop delay calibration we omit all the paths which show this noise in them.

## 6.4 Die-to-Die Variation in Flip Flops for Rising Edge in All Chips

For the analysis of die to die variation in flip-flops we select the flip-flops which are common across all the chips and whose delay variation is not more than 15 FPA steps that is not more than 0.5ns. The data is plotted with the median value of each flip flop against the delay of all the chips. For each flip-flop we have 28 data points from which the range and percentage variation is calculated.

The standard deviation is a measure of amount of variation / dispersion from the average. The results in Fig 6.12 shows a dispersion of 2.5 on average. The percentage change, defined as range or 3 σ variation / mean delay * 100, is on average 30 %, and is ranging from 25% to 40 % for the rising edge. As the delays of each flip flop is small and the impact of small variation will be bigger so the results of 40% variation is reasonable shown in Fig 6.13. Once the delay analysis done on paths we will see that over all percentage variation will be much less.

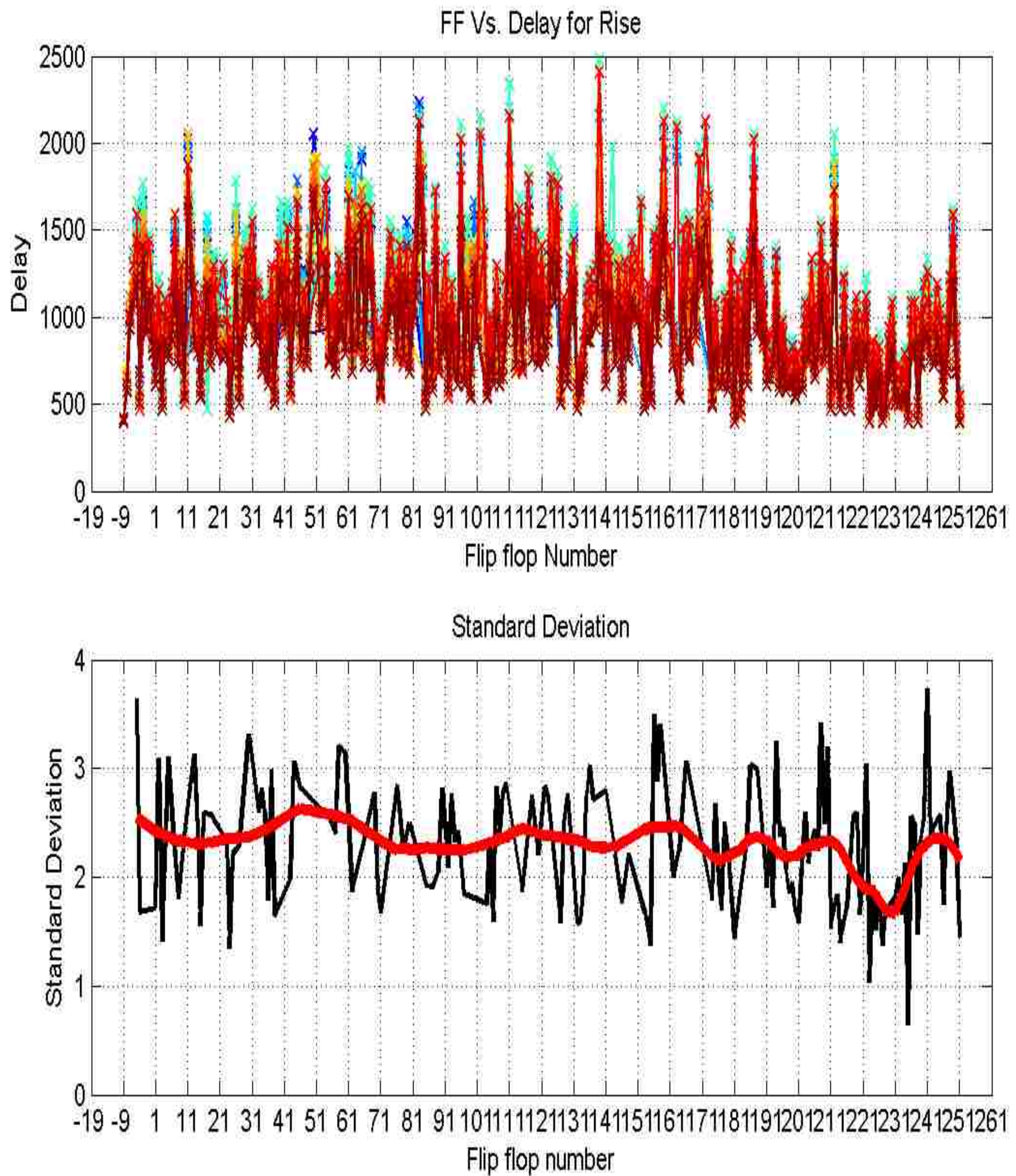


***Fig. 6.12: Standard deviation in Rising Edge in all chips***

Fig 6.12 (a) shows the distribution of delays of scan cells of the capture row once tested with the different random test vectors and insertion points. The delay is ranging

from 0.5ns to 2.4ns. This variation is the chip to chip or die to die variation. Some chips are faster and the propagation delay is less then the chip with data points on the top of the graph. Over all standard deviation is less and is 2.5 showing that the data dispersion is closer to the mean.

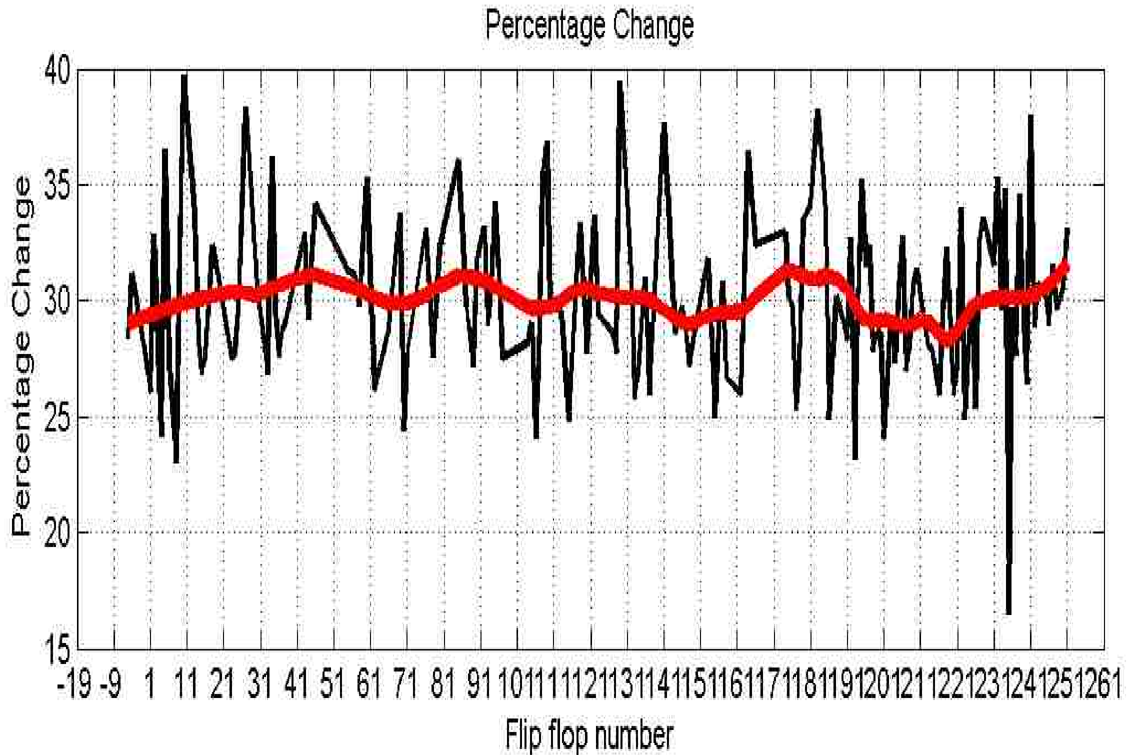


***Fig. 6.13: Percentage Change in Rising Edge for All Chips***

The die-to-die variation in rising edge in all chips, shown in Fig. 6.13, is captured by measuring the variation in the measurements of flip flop delays from one chip to another by also considering the mean delays of each flip flop and dividing that mean from the range of delay across all the chips for that given flip flop. It is a good measure in terms of relative difference while taking the average delay in to account. 15% to 40%.

## 6.5 Die-to-Die Variation in Flip-flops for Falling Edge in All Chips

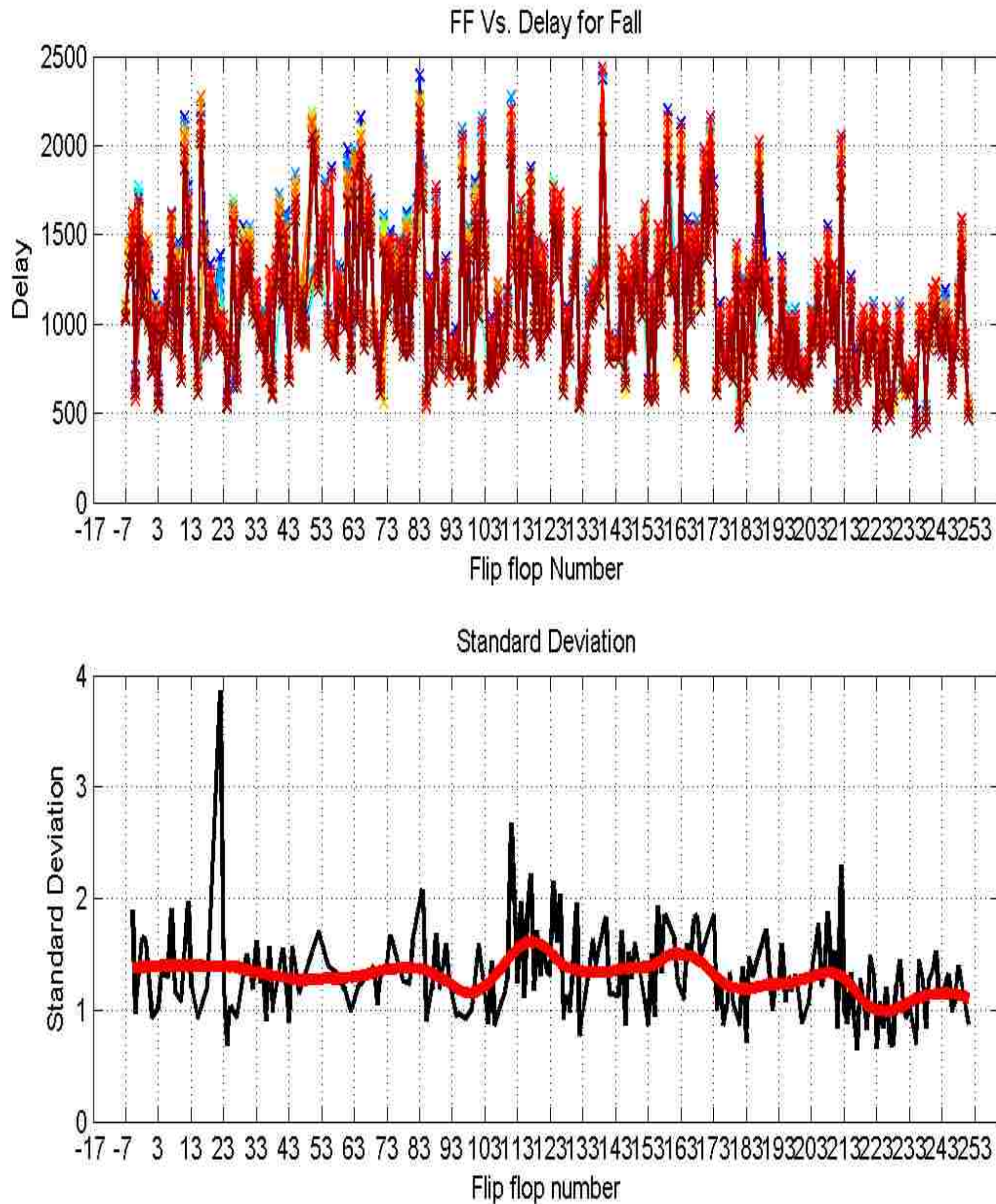


*Fig. 6.14: Die-to-Die Variation in Falling Edge in all Chips*

Similar analysis is performed on the falling edge propagational delay, the results show that the standard deviation is even less for the falling edge transition, that is on

average 1.5, which means that the data is much closer to the average.

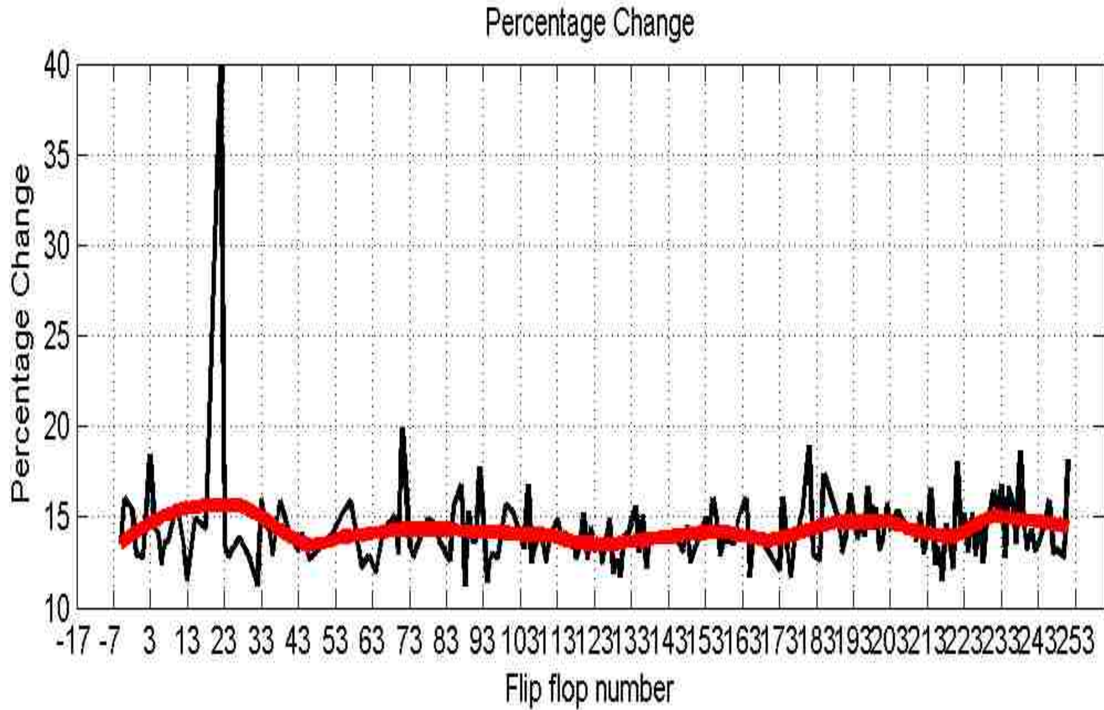


***Fig. 6.15: Percentage Change for Falling Edge***

The die-to-die variation in rising edge in all chips is shown in Fig. 6.15.The results of percentage change across all the 28 boards shows that it is on average 15% , but can reach upto 40%. The curve that is showing the percentage change of 40% is from the flip flop 22 in the delay chain, which is on the further end of the delay chain. For this analysis we know that with in a chip the delay did not change more than 15 FPA steps each of size 36ps. Across all the 28 chips we see a variation of 40 %, this variation can be caused because of the spatial location in side the design.

## 6.6 Path Distribution Analysis

Path delay distribution shown for two chips Chip 1 and Chip 2. Graphs are constructed with 10,000 stable paths (required 25,000 paths to be tested). With 2 samples per path, it takes approx. 25 seconds to find these 10,000 paths.

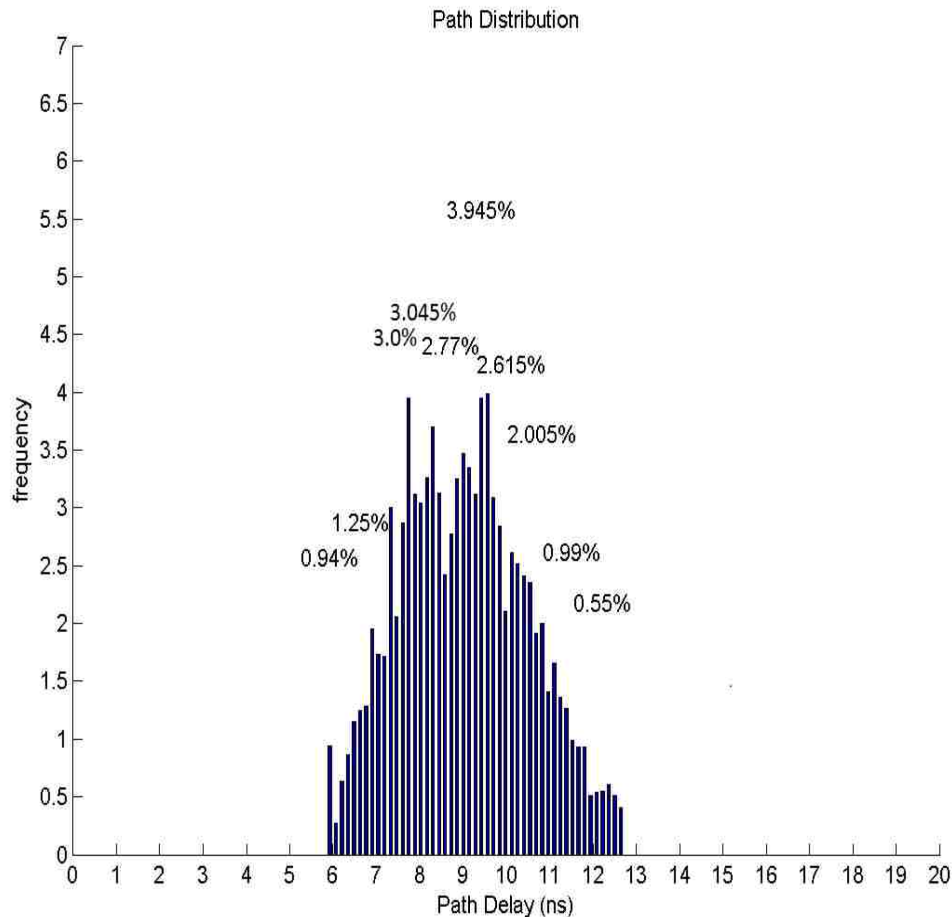


*Fig. 6.16 Path Distribution for Chip C1*

A plot depicting the distribution of these 10,000 path delays for CHIP1 using data from the 25°C, 1.20V TV corner is shown in Fig. 6.16. The x-axis plots the actual calibrated path delay against the number of instances on the y-axis. The path delay ranges from 6ns to 12.5ns, the mean is at 9.5ns. The distribution has more median length paths, it has a uniform distribution with a bell curve. Similar trend of path distribution is noticed

in Chip 2 shown in Fig. 6.17 and other chips.

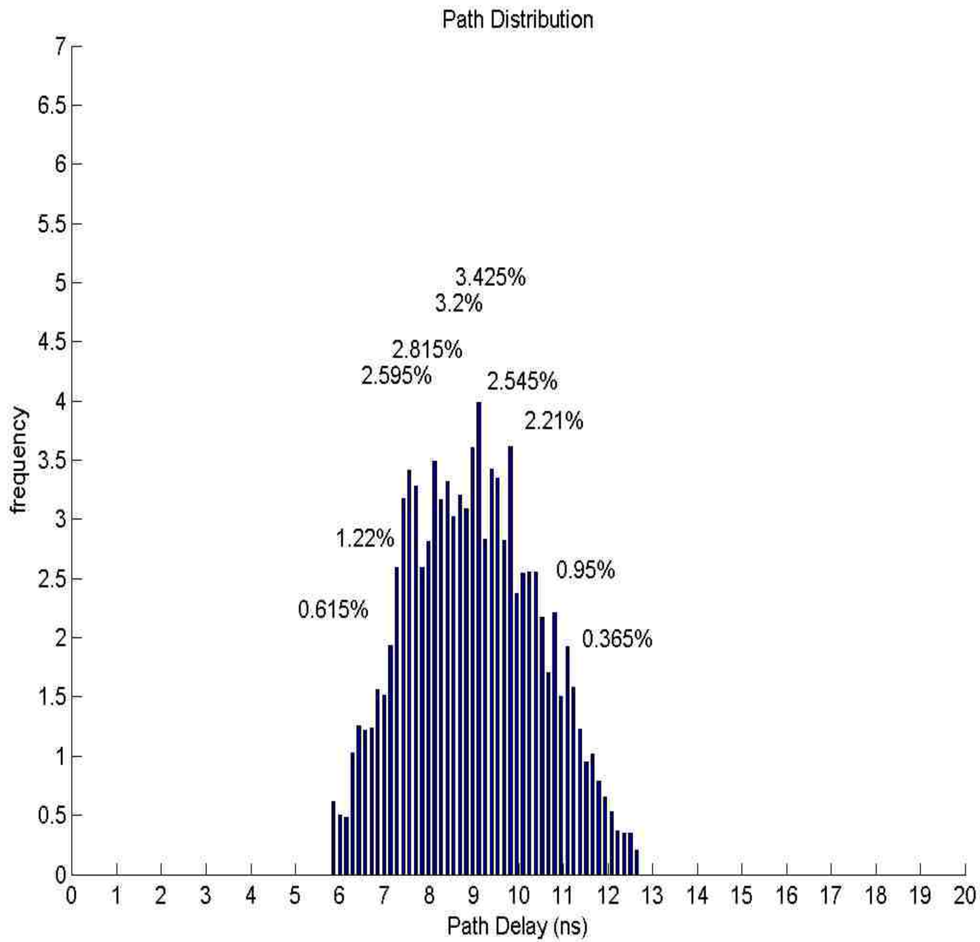


***Fig. 6.17: Stable path distributions for Chip 2***

Figure 6.18 shows the uniqueness of paths across chips shown on right. It is constructed with 10,000 stable paths which can range from 0 to 25,000 paths ids which are tested to get the 10,000 stable paths. The process of data collection for all these paths with at least 2 samples is 25 seconds and hence is quite fast. Here we see that the paths which are common in all chips where the tests performed are the same and the bit stream is kept the same but the tested hardware, that is the FPGA are different. We see that only 2.2% of the paths are the same on all chips in a set of 28 chips.

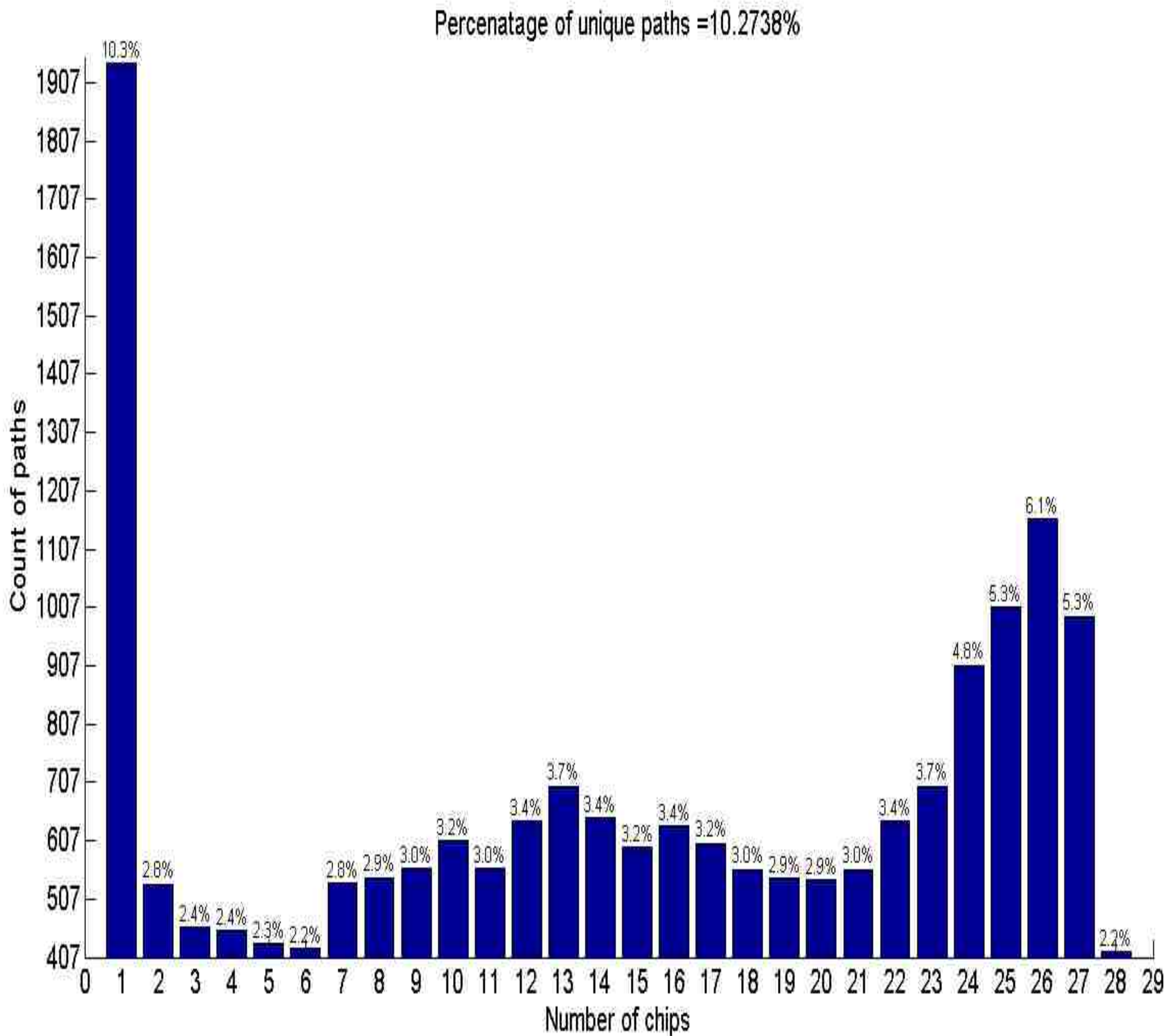


*Fig. 6.18: Unique and Common Path ID Counter*

We have total unique path IDs equal to 18883, and among them 2.2% are common in ALL chips which equals 417 paths that are common and unique paths are 10.3% which equals to 1940 paths that are present only in 1 chip. Remaining paths are 16526 which equals to 87.517873%

## 6.7 Die-to-Die Delay Variation Analysis

The die-to-die delay variation is computed using the regression analysis. The paths are sorted on their delays, and path pairings are formed using the two neighboring paths with closer path delays. This process of path pairings selection is completed on chip 1 and these path pairings are used across all the chips to plot the scatter plots from all the 28 chips. The LSE and Three σ prediction interval curves are plotted for the measurement of die-to-die and within-die variation. The vertical offsets of the data points from the LSE

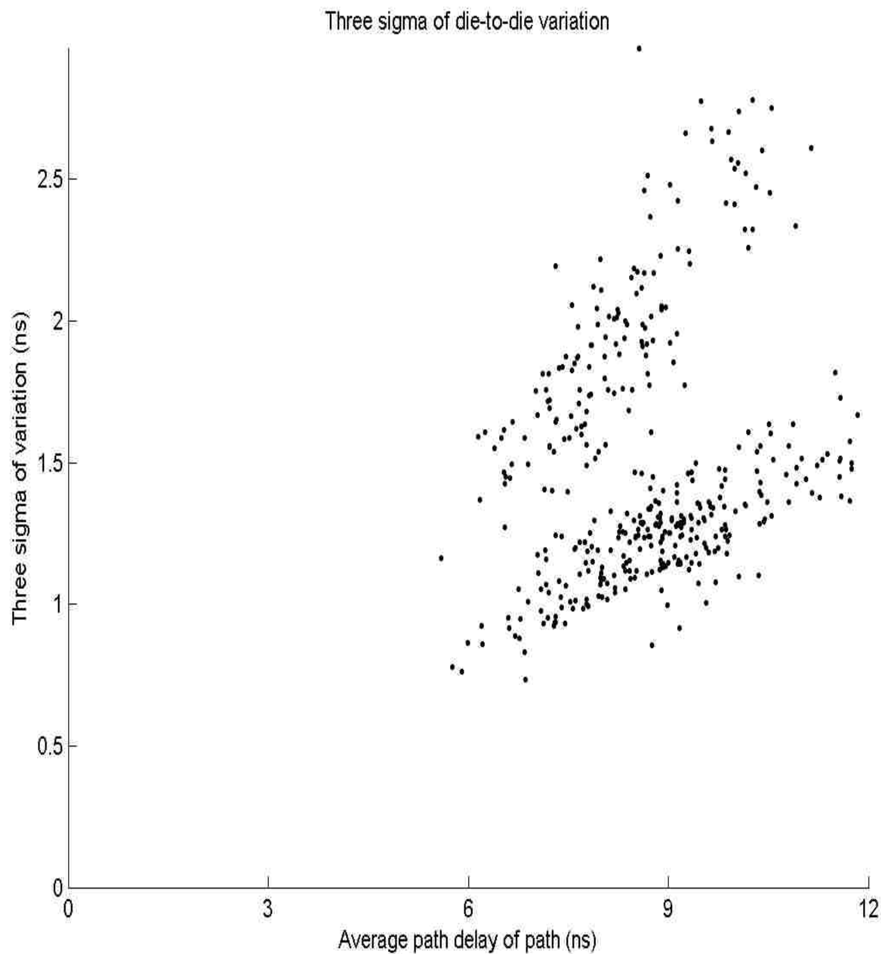


*Fig 6.19: Die-to-die variation analysis using regression. Average path delay vs. normalized 3σ of residuals*

are called ‘residuals’. The three σ prediction interval curves driven from the scatter plot reflect the overall spread of the points around the LSE line. The 3σ of the residuals are first computed for all the path pairings and then are ‘normalized’ by the average path delay.

The average path delay is the mean x-value from Fig. 6.19 for each of the scatter plots which ranges from 5ns to 12ns.The 3σ shows the die-to-die variation that is from one chip to another. The range of variation is in the range of 1ns to 3ns. The graphs shows two humps, one at 1ns and the other at 2ns showing that some of the paths are more affected with the variation from chip to chip and some are less. The placement and spatial variations of the logic components which are being timed plays an important role, as some regions (the corners) have more effect of temperature and voltage than others (the center).

The percentage change in Fig 6.20 shows the two humps, identifying that variations in path delays of equal magnitude can have from 14% variation to 24% variation, spatial dependencies of the logic playing an important role. The path delays with the similar delays can fall into the range with an average variations of 14% or can be in the other region where the die-to-die variation is much higher and can reach up to 25%. The placement of these path delays which are studied under this test, and the composition of the paths, that is which gates are included in the paths and the fanout of each gate can factor in the variation in the percentage change as seen in the analysis.

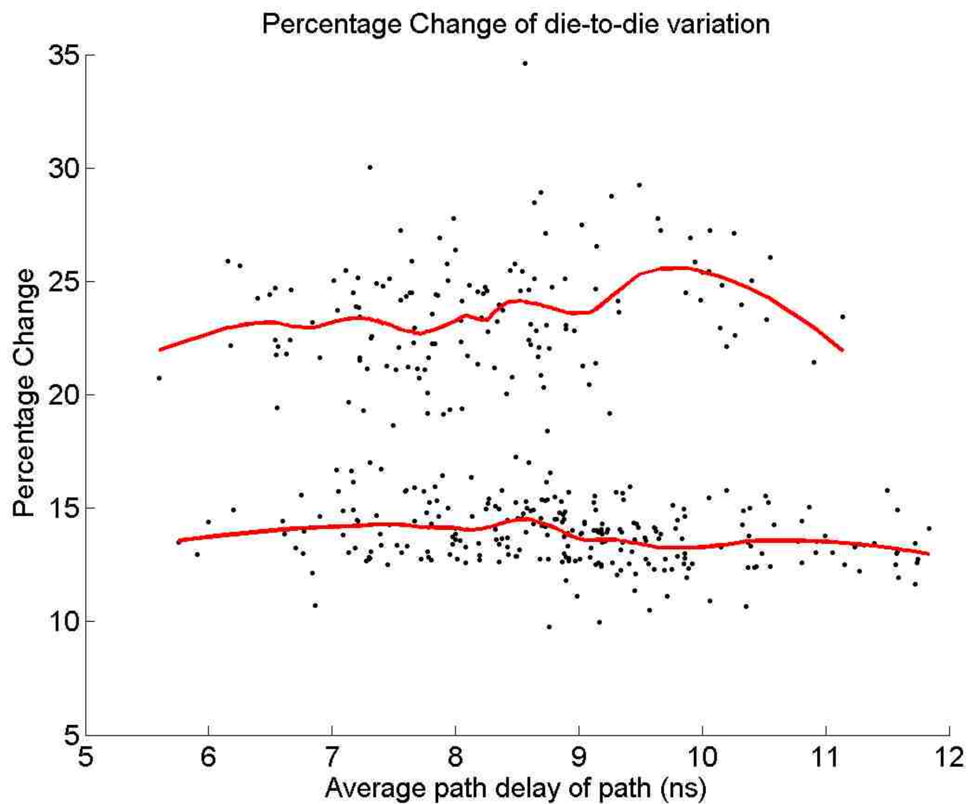


*Fig. 6.20: Overall percentage change of die-to-die variation in path delay*

## 6.8 Within-Die Delay Variation Analysis

Regression analysis is used for measuring and analyzing within-die variations by measuring the least estimate square (LSE) of a best fit line through the data points. Linear regression is applied to scatter plots which are constructed from the delays of two separate paths, i.e., a path pairing. The pairing are formed from the sorted list of delays, such that the neighboring paths will have closer delays. We create the path pairings by sorting the delays from CHIP1 and then creating n-1 scatter plots by pairing data sets in the order given by the sorted delays from CHIP1. This ensures that the paths of each pairings have similar delays. Within-die variations (and noise) are represented by the vertical offsets of the data points from the LSE line (Fig. 6.21).

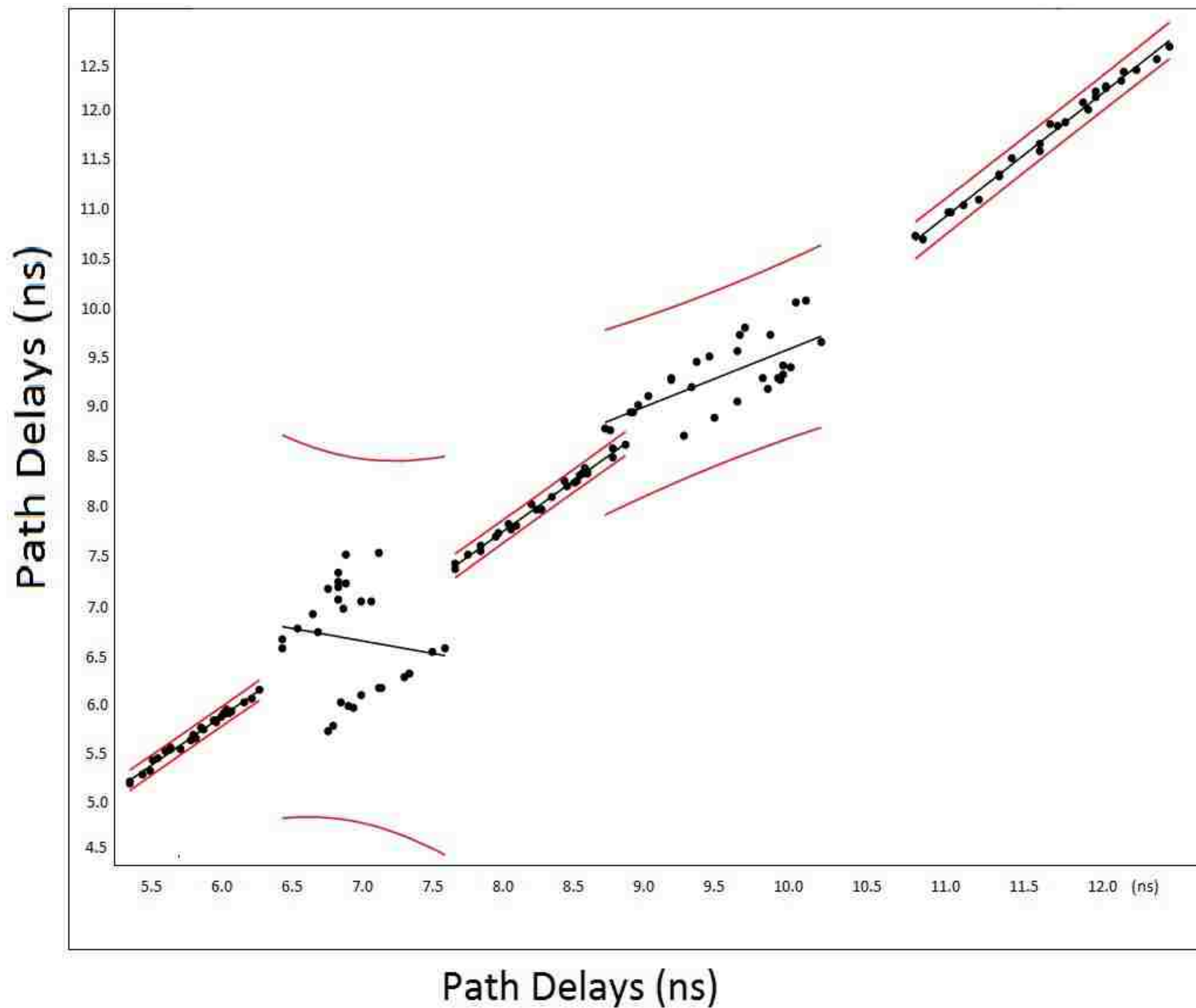


***Fig. 6.21: Within-die Variation for short and long path pairings***

Figure 6.21 shows the regression analysis of short path, medium as well as long paths. It is seen that overall the data points are lying very close to the regression line and the 3 sigma intervals are very tight. Some path pairings show more dispersion of data from the regression line. To better understand the within-Die variation, it is more obvious once we analyze the range of variation along the regression line.

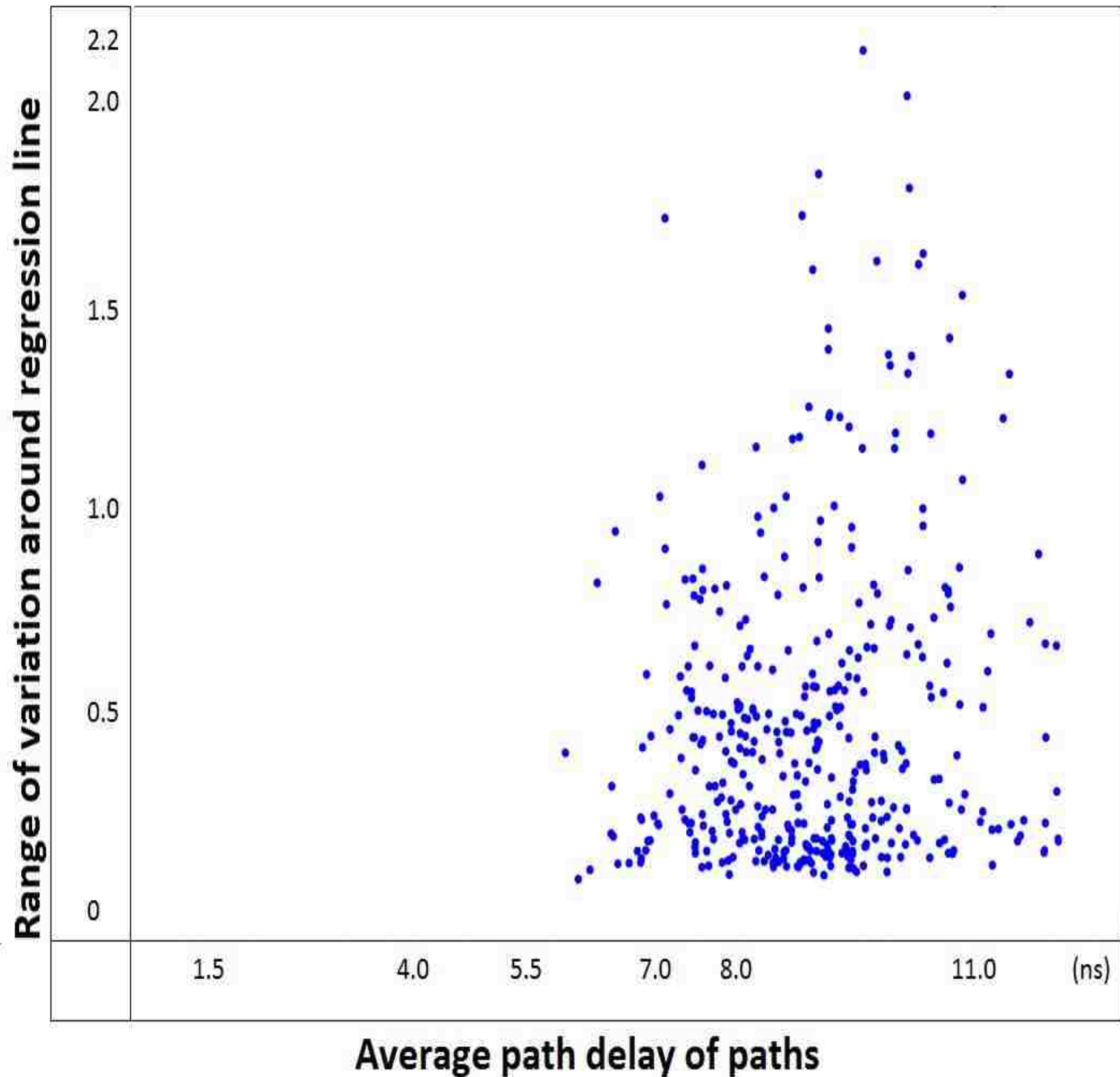


*Fig 6.22: Within-Die delay variation analysis using regression: example scatter plot from distributions of common to all chips*

The x-axis is the mean path delay of all the data points in the curve and is calculated for each path pairing, and y-axis plots the range of dispersion of the data points from the regression line for each curve. We have total of 399 curves which are common in all the chips, that is we have 399 path pairings in total. The mean of all the delay for the given path pairing shows on average how much is the range for the given mean delay. The mean path delays range from 5.5ns to 12.5ns.

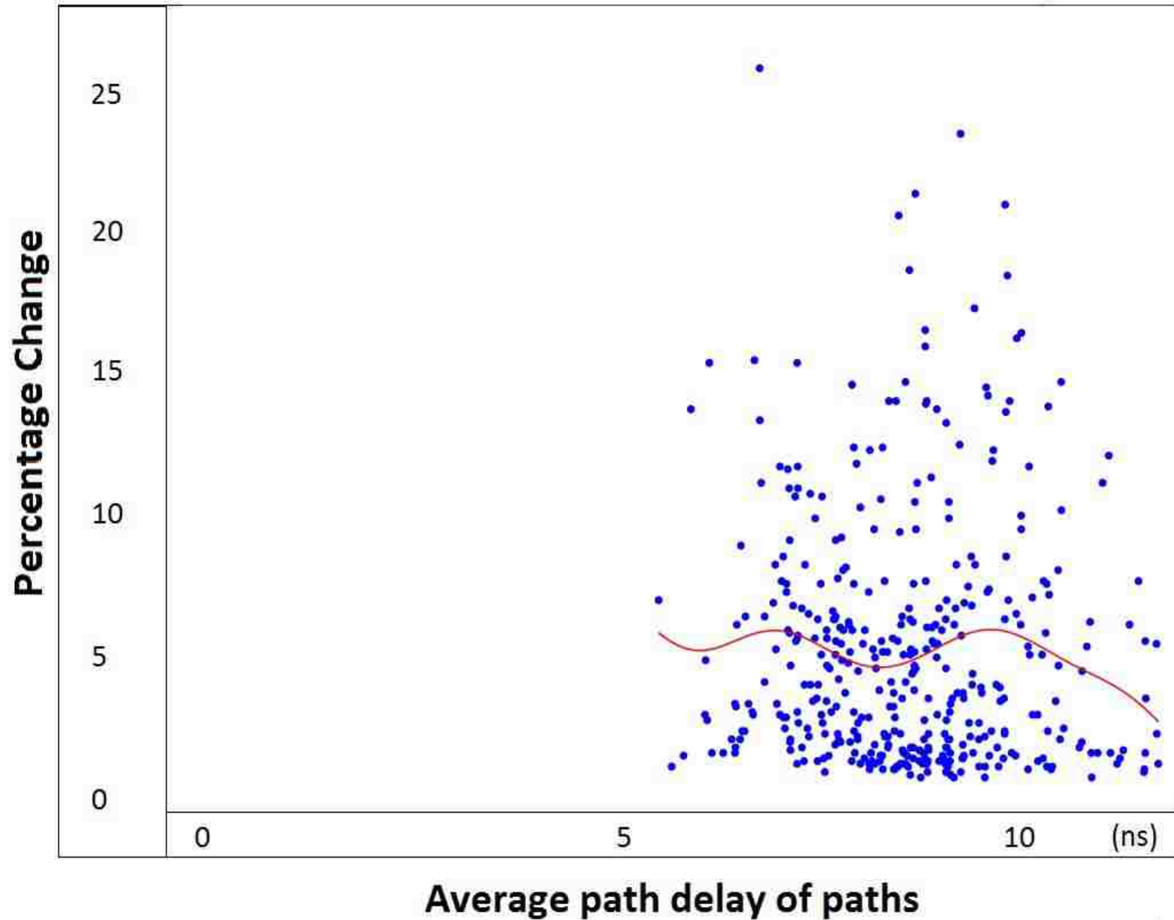


*Fig. 6.23: Overall percentage change of path delay variation*

Fig. 6.23 shows that the over all percentage change of the path delay variation is much less than the flip flop variation. The percentage variation is around 5 % and it can reach upto 27%. This analysis is of vital importance in case of variation aware design and can be used for both FPGAs and custom integrated circuits. From the results it is observed that more variation exists in the ASIC as compared to the FPGAs, where reasons can be FPGAs are commercial designed and multiple copies are created so more focus is to improve the design with minimal variations, where as for ASIC the design correctness and efficiency is of more importance.

# CHAPTER 7
# Pipelined Decision Tree Implementation

Decision Tree Classification (DTC) is a widely used technique in data mining algorithms known for its high accuracy in forecasting. As technology has progressed and available storage capacity in modern computers increased, the amount of data available to be processed has also increased substantially, resulting in much slower induction and classification times. Many parallel implementations of decision tree classification algorithms have already addressed the issues of reliability and accuracy in the induction process. In the classification process, larger amounts of data require proportionately more execution time, thus hindering the performance of legacy systems. We have devised a pipelined architecture for the implementation of axis parallel binary decision tree classification that dramatically improves the execution time of the algorithm while consuming minimal resources in terms of area. Scalability is achieved when connected to a high speed communication unit capable of performing data transfers at a rate similar to that of the decision tree classification (DT) engine. We propose a hardware accelerated solution composed of parallel processing nodes capable of independently processing data from a streaming source. Each engine processes the data in a pipelined fashion to use resources more efficiently and increase the achievable throughput. The results show that

this system is 3.5 times faster than the existing hardware implementation of classification.

## 7.1 Introduction

The process of converting unidentified or unprocessed data into actionable information that is important and valuable to the user is known as data mining [16]. Recent advances in technology and ever increasing demands for analyzing larger datasets have created abundant opportunities for algorithmic and architectural development and innovations. Hence data mining algorithms have become increasingly significant and complex. Similarly there is a great demand for faster execution of these algorithms, leading to efforts to improve execution time and resource utilization.

Decision Tree Classification (DTC) is a widely used classification technique in data mining algorithms. It has applications in daily life; for example, the detection of spam e-mail messages. It is also used in highly sophisticated fields of medicine and astronomy. Several diverse predictive models in classification algorithms including artificial neural networks [58], decision trees [59] and support vector machines [60] have also been previously described in the literature. A number of solutions have also been suggested for hardware implementation by various authors [61] [62] [63]. Decision tree classification techniques categorizes each data records/tuples, having set of attributes/properties into subgroups or classes. Assigning of a category or class to each input dataset consists of a two-step process in DTC.

The initial step is induction which involves construction of the decision tree model, where internal nodes and leaves constitute a decision tree model. Each internal node has a characteristic splitting decision and splitting attribute, while the leaves have

particular category classification. Construction of a decision tree model from a training dataset/tuple constitutes of two phases. A splitting attribute and a split index are chosen by the model during the first phase. While during the second phase sorting of the tuples among the child nodes is performed based on the decision made in the first phase. This repetitive process is continued till the depth of the tree reaches a desired level. At this point, the decision tree can be used to predict the class of an input tuple which has not been classified yet.

The second step is the classification that includes application of the decision tree model to the test dataset to predict its respective class. The primary goal of such a classification algorithm is to utilize the given training dataset to construct a model which subsequently can be used to sort unclassified datasets into one of the defined classes [64]. Breiman et al [65] presented decision trees approximately two decades ago, and described the decision trees as rooted tree structures, with leaves representing classifications and nodes representing tests of features that lead to those classifications. The accuracy of decision trees has been shown to be better or comparable to other models including artificial neural networks, statistical, and genetic models. The prediction in the classification process commences at the root, and a path to a leaf is followed by using the decision rules governed at each internal node. The characteristic class label to the leaf is then assigned to the incoming tuple.

DTC continues to function at high accuracy even in analysis of large data sets. Current technology advancements in data extraction and storage permit large amount of historic data to be preserved and utilized for data analysis and creation of more realistic classification rules. The property of DTC to function at high accuracy even when

handling in large data sets makes it an appealing tool.

Decision trees have since been implemented in software programs. Although the software implementation of DTC is highly accurate the execution times and the resource utilization still require improvement to meet the computational demands in the ever growing industry. Whereas hardware implementation of Decision trees has not been investigated or reported in detail. Only a few researchers [66] [67] [68] proposed hardware realization of various decision trees using different architectures for specific problems.

Our work focuses on the speedup of the classification step using hardware acceleration. We propose a pipelined architecture for the hardware implementation of axis-parallel binary decision tree classification that meets the current demands of increased throughput with minimal resource utilization. The proposed design supports a streaming architecture by using double-buffered input and output memories to simultaneously receive and process data. Our experiments prove that our proposed hardware acceleration of classification algorithms increases throughput by reducing the number of clock cycles required to process the data and generate results. The architecture also requires minimal resources and is therefore area efficient. For scalability this proposed architecture, when configured with a high speed communication unit, enables processing and data transfer simultaneously. As long as the performance of the decision tree classification engine meets or exceeds that of the communication unit, processing time is not affected by the transfer of data.

We developed the decision tree classification algorithm in detail and explored techniques for adapting it to a hardware implementation successfully.

## 7.2 Background

A number of hardware implementations of decision tree examples are reported in the literature [66] [67]. The approach of using single level classification technique instead of staged or multi-level technique limits the throughput because of having a restraint in the design that new instance cannot be applied to the input before completion of the classification of the previous data instance, resulting in low throughput. On the other hand the staged/ leveled technique allows a new instruction/data fetch every clock cycle and thus optimizes the throughput.

A more advanced approach, proposed by [66] is based on the equivalence between decision trees and threshold networks hence resulting in fast throughput since the signals have to propagate through two levels only, irrespective of the depth of the original decision tree. Most of the architectures for hardware implementation of decision trees mentioned in the literature require a considerable number of hardware resources [68].

Past research work has been reported on hardware implementations of data mining algorithms. Baker and Prasanna [69] used FPGAs to implement and accelerate the Apriori [70] algorithm, a popular association rule mining technique. They developed scalable systolic array architecture to efficiently carry out the set operations, and used a "systolic injection" method for efficiently reporting unpredicted results to a controller. In [71], the same authors used a bitmapped CAM architecture implementation on an FPGA platform to achieve significant speedups over software implementations of the Apriori algorithm. Several software implementations of DTC have been proposed [72] [73], which used complex data structures for efficient implementation of the splitting and

redistribution process. These implementations focused on parallelizing DTC using coarse-grain parallelization paradigms.

Li and Bermak [74] suggested a decision tree classifier based on an axis-parallel decision tree. Bachir et al. [75] presented both a hardware-dedicated decision tree technique for the generation of exponential variates and a derived architecture implemented in FPGA.

Podgorelec and Kokol [76] proposed a self-adapting evolutionary algorithm for the induction of decision trees and described the principle of decision making based on multiple evolutionary induced decision trees – decision forest. Chrysos et. al [77] presented data mining on the web for classifying and mining huge amounts of e-data by an implementation of data mining algorithm on a modern FPGA to accelerate certain very CPU intensive data-mining/data classification schemes. Subsequently they exploited parallelism at the decision variable level and evaluated its implementaion on a modern high-performance reconfigurable platform [78].

The objective of this paper was to find an architecture that could ensure high throughput with significant reduction in hardware complexity. Generally, with an increase in the data sizes, the running time stretches to several hours. In the architecture designed for this research, each data record is assigned to a class using the predefined classification rules. The developed solution yielded high accuracy while handling large datasets. The hardware implementation for this study helped enhance the performance over software implementations.

## 7.3 Decision Tree Classification Architecture

In this paper we propose an efficient pipeline based implementation of a decision tree classification algorithm. The hardware accelerator for decision tree classification performs parallel operations using concurrent engines, where each engine implements pipeline technique and thus fetches data records in every cycle, enhancing the performance of classification process.

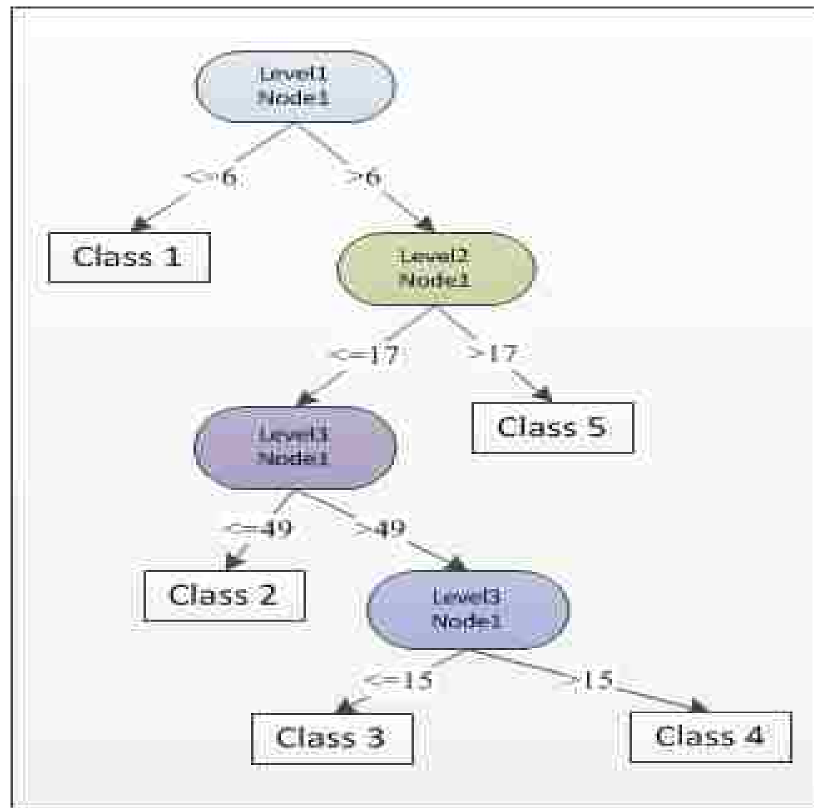


*Fig. 7.1: Decision Rules in form of Decision Tree*

In our solution we proposed and adopted a two phased decision tree classification process. Firstly in the induction Phase a training dataset is used in order to determine the rules, based on which the classification is to be done, at each node. We have opted to provide these induced decision rules from the Microblaze softcore microprocessor to the decision tree classification engine. In the next phase, the classification is performed at the

hardware level. Our proposed architecture employs a pipelined data path, where the data is distributed in a pipeline to execute concurrently, which is of significant importance for large datasets to reduce the clock cycles.

The decision tree classification engine architecture concentrates on axis-parallel binary trees, where each node in the tree can have no more than two child nodes and only one of the attributes comprising the dataset is compared against a constant at each node. These constants are determined in the induction phase for each node. Figure 7.1 shows an example of binary decision tree, for a given dataset, where the leaf nodes represent the classes that divide the data into different categories, and each internal node represents the test conditions, from which it traverses and reaches one of the classifications.

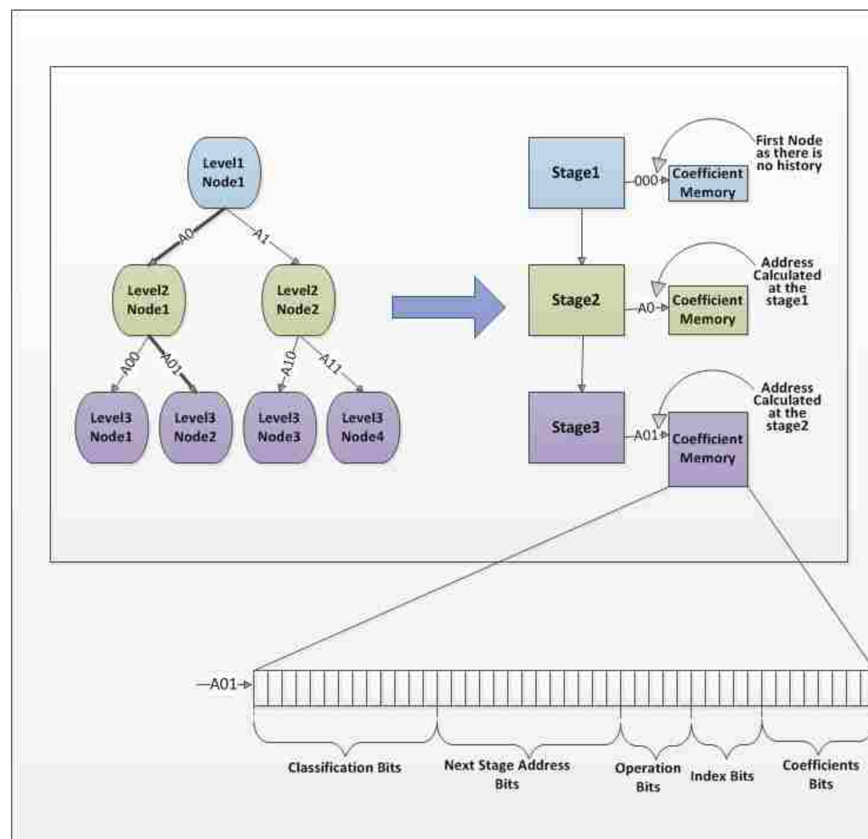


*Fig. 7.2: Decision Tree Stages*

The decision tree classification subsystem implements each level of tree using a stage as represented in Figure 7.2. Each stage consists of a decision logic, coefficient memory and internal registers. The input address to the coefficient memory is a function of the path through the decision tree that was taken to arrive at that particular node. Each coefficient memory stores coefficient values, attribute index of the incoming data from which to compare the coefficient, operation to be performed and a pointer to either the memory location of the next stage or the class assigned. The output of the coefficient memory contains all the information needed to perform the operation associated with the node in the tree being addressed.

The decision tree classification engine has three major parts: a) the double-buffered input block RAM b) the decision tree classification subsystem, and c) the double-buffered output block RAM. The decision logic reads the incoming data and takes the rules from its associated coefficient memory, processes them and forwards the data to the next stage with the processed results. The intermediate results decide whether a category is assigned to the data or further processing is required in the next stage. In case when the classification is complete for a data, the data is forwarded to the next stages without further processing, otherwise the processing and comparison is repeated until it is assigned to a class and then stored in the output memory. All these operations are performed in a pipelined manner where in every clock cycle the data is forwarded into next stage and newer data is fetched.

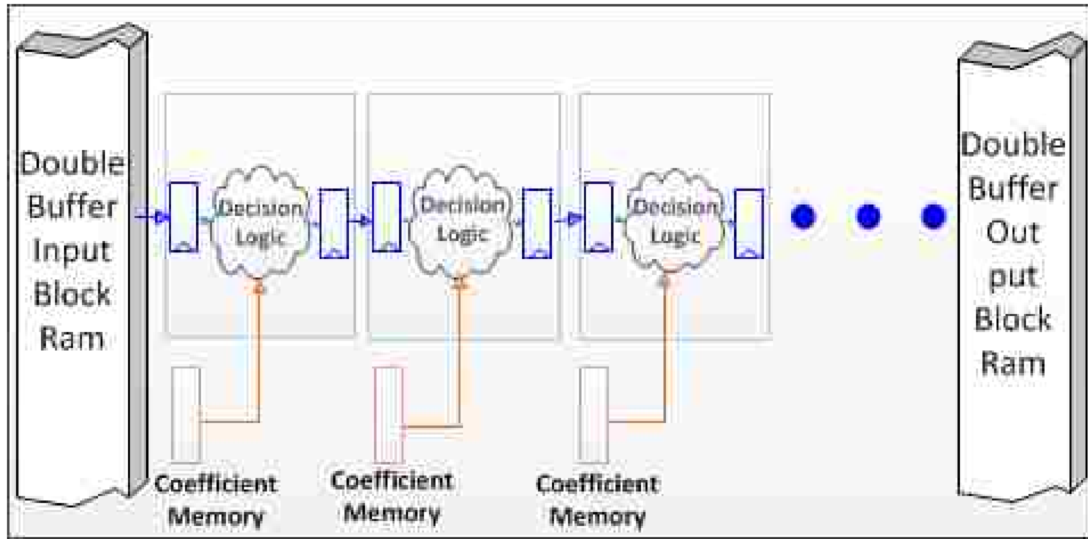


*Fig. 7.3: Decision Tree Classification Subsystem*

Figure 7.3 represents a decision tree with depth of *n*, having *n* stages from which the data passes through, and then the classification is stored in the output block memory. The unclassified data is provided by the double-buffered input block RAM to the first stage of the engine, from where it is processed and propagated down the pipeline. The classifications for each tuple, are stored in the double-buffered output block RAM. The Xilinx Logicore IP Block Memory Generator has been used in order to implement the input and output memories. Where, block memory generator uses embedded block memory primitives in Xilinx FPGAs to implement memories of different depths and widths. Our proposed design implemented on Digilent Nexys2 Spartan 3E board uses two fully independent ports each with its own read and write interfaces and access to a shared memory space. These ports can operate at different clock frequencies thus making it possible for the classification subsystem to operate at double the frequency of the on-board system clock.

Figure 7.4 shows the RTL level block diagram of one such hardware module/stage of the classification subsystem. In each module there is a memory element, namely

coefficient memory associated with it. These memory elements are also generated using the Xilinx Logicore Distributed Memory Generator IP. During the memory configuration stage the RAM_access bit is set high. This allows the Microblaze to access the coefficient memory, in order to write the rules for each node associated to that level. The control unit ties the address lines of the coefficient memory to the address value received from the previous hardware module in the pipeline. The size of the coefficient memory depends on which level of the tree it is associated. Hence the size varies from one 64 bit wide line to $2^n$ 64 bit wide lines where n is the number of levels in the decision tree.

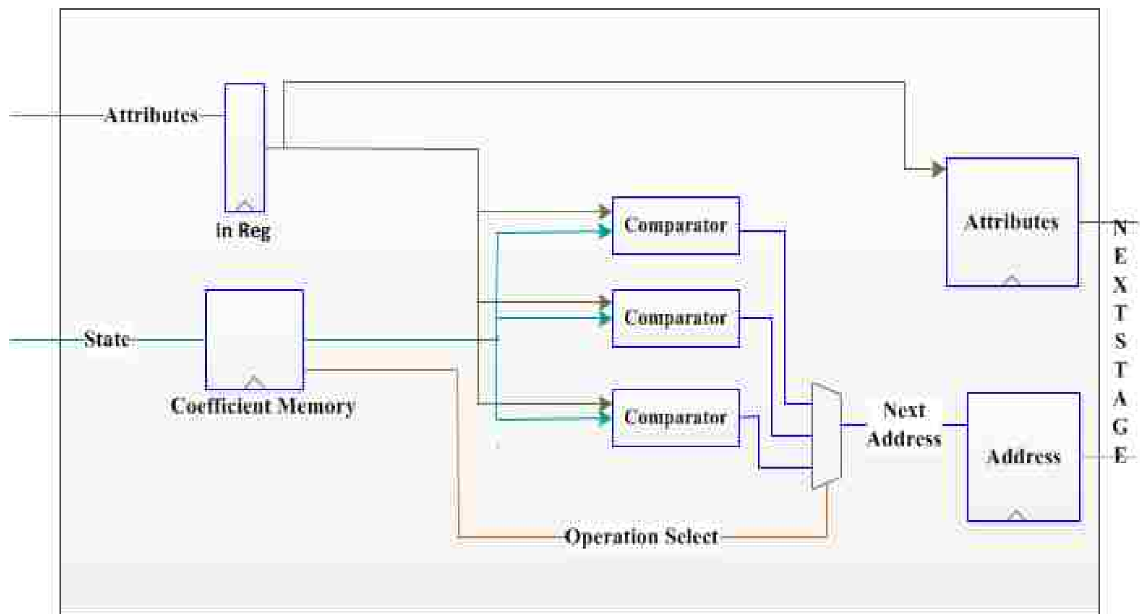


*Fig. 7.4: RTL level Block Diagram of Hardware Module*

The attributes are transferred to the module from the double-buffered input block RAM or the previous stage in the pipeline. Depending on the Attribute Index the attribute to be compared is selected and transferred to the comparators. The constant that it is to be compared against is fetched from the coefficient memory based on the address received from the previous stage. The new address for the coefficient memory of the next stage signifying the path to be taken (left child or right child) while traversing the decision tree

is sent to the next stage in the pipeline based on the operation select lines and the comparator outputs.

The decision tree classification has been implemented as a Hardware-Software Co-Design. The Xilinx soft-core microprocessor Microblaze has been used to supply and fetch data to and from the reconfigurable decision tree classification engine. The data coming in is read by the Microblaze which sits on the Peripheral Local Bus (PLB). Microblaze in turn transfers the data to the double-buffered input block RAM of the decision tree classification engine. The engine is a custom peripheral designed as a slave module of the PLB. Once the double-buffered input RAM is written to with a given batch of data the Microblaze activates the classification engine by asserting a signal. The classified data is written into the double-buffered output block RAM.

In order to increase the efficiency of the engine it has been made parallel. Figure 7.5 shows the overall pipelined and parallel architecture where the decision tree subsystem is instantiated eight times thus facilitating computation of eight classification result every clock cycle. After the initial latency, equivalent to the number of levels in the tree, 8 tuples of the dataset are categorised every clock cycle. Our tested design of the proposed architecture allows a depth of up to 13 levels, therefore the maximum latency for this design is 13. The address management for writing to the double-buffered input block RAM and reading from the double-buffered output block RAM has been done in such a way that eight consecutive tuples can be read and classified in every clock cycle. The double-buffered input and output RAMs are designed to allow for simultaneous buffering and processing. The operations of each RAM are switched after the given batch of data records are processed by the classification subsystem.

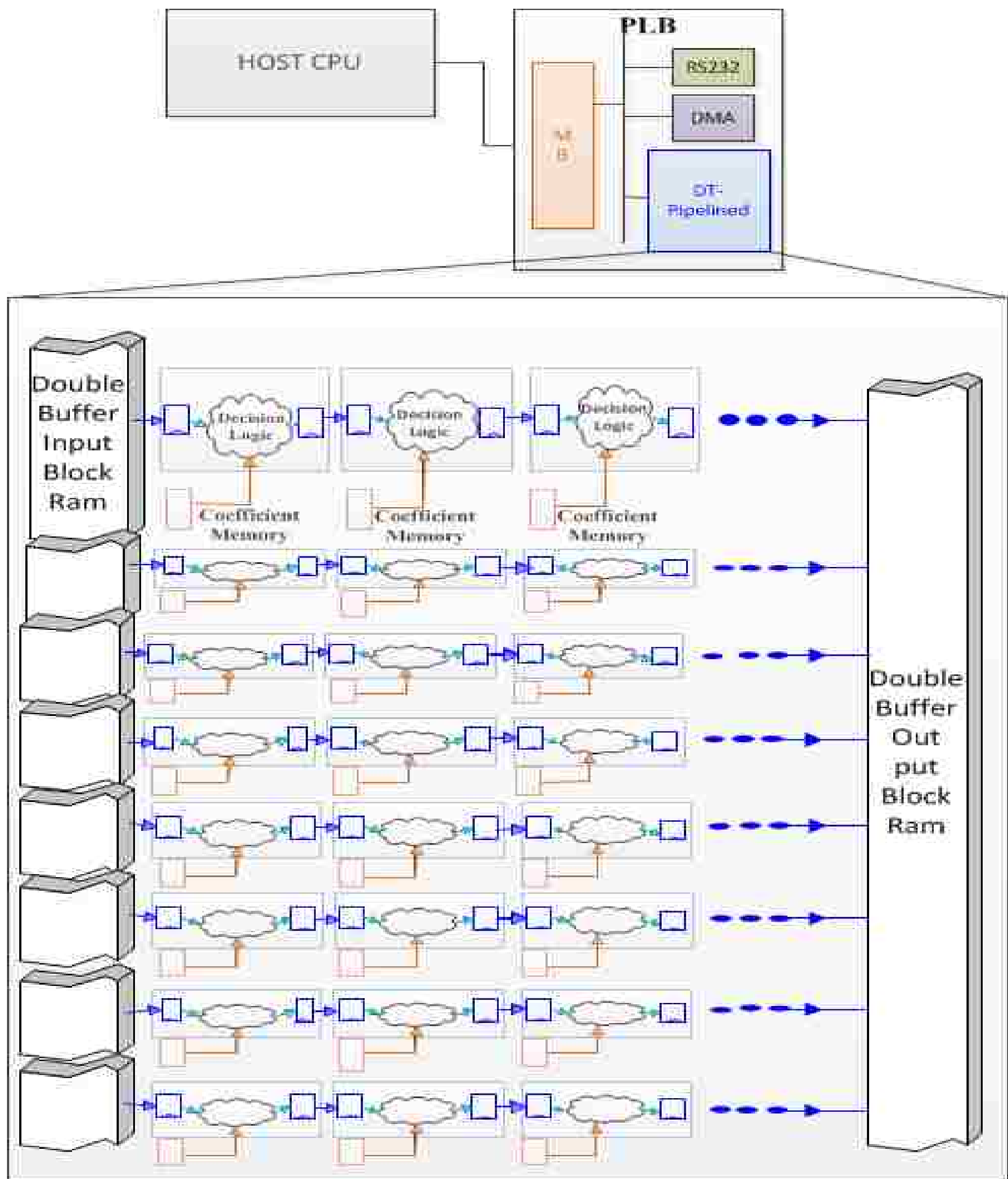

*Fig.7.5: Parallel and Pipelined Decision Tree Engine*

In theoretical analysis we analyzed following characteristics and limitations of the hardware architectures designed previously.

i. Only single data record is fetched in every cycle, thus requiring more clock

cycles.

ii. Data record is fetched in sequential order from single input memory.

iii. The engine performs the classification and stores in output memory and only then fetches the new data record. Thus wasting the processing cycles.

Following are the enhancements in our proposed architecture where we utilize the hardware pipelines and parallelism to overcome the above mentioned limitations:

i. Engine is made of pipelined stages, each stage implements rules of one level of the tree.

ii. Pipeline to make use of processing cycles when data is written in memory, thus to increase the performance.

iii. Engine works on clock frequency double to that of the interface clock.

iv. Multiple data records are read as well as written simultaneously in every cycle, exhibiting parallelism, thus reducing the overall clock cycles.

v. Distributed memories are used for the coefficient lookup tables inside the peripheral for making the engine memory efficient, and to reduce the clock cycles to access the data.

vi. The block RAMs are placed in the peripheral such that the bus is not used in the memory accesses, thus reducing the clock cycles required for setting-up bus protocol.

vii. Also, the on-chip block memories are used for the pre-processed datasets, the classification rules and storing the classification results.

Consequently, we are able to optimize the access to memories in one clock cycle, in the given architecture. This results in overall reduction of clock cycles and hence a greater impact on the performance.

The development board used for this work is the Diligent Nexys-2 Spartan-3E FPGA Board featuring a single Xilinx XC3S1200E-FG320 FPGA. This particular component does not support PCI Express and without access to a high-performance interface, the proof-of-concept design discussed in this paper is implemented using RS232 to move data back and forth from the host to the FPGA. As the bandwidth of an RS232 link is inappropriate for an application requiring high performance, the I/O transfer time in the performance results as their inclusion would have completely hidden the performance increases realized by our parallel architecture.

The theoretical performance of the Gen-2 PCIe hard core in the Virtex-6 FPGA is 500 MB/s/lane, giving an x8 design a raw bandwidth of 4 GB/s in each direction. Assuming to achieve 80% efficiency due to bursting and DMA, this would be equivalent to transferring 3.2 GB/s, or 800 Mwords/s in each direction. Our design, for 4 attributes processes eight 32-bit samples in parallel at 100 MHz, the raw bandwidth of our logic is also 800 Mwords/s. Therefore, if we replace the RS232 interface with a PCIe interface, the I/O bandwidth would, at a first-order estimate, match that of our parallel implementation. For this reason, it is reasonable at this stage to include only the performance results for the parallel implementation and ignore the transfer time represented by our legacy RS232 interface, as a modern interface such as PCI Express would be able to keep up with our design's classification rate. The FPGA Implementation and experimental results are discussed in the next section.

## 7.4 Experimental Results

We have implemented the proposed architecture on Digilent Nexys2 Spartan 3E FPGA board to perform classification in hardware accelerator. Variety of datasets, varying from benchmark to synthetic datasets have been used. The Number of tuples also varies to verify and validate the performance dependencies of the engine. Data pre-processing includes data cleansing, that is to normalise the data and conversion into hex-decimal number, to feed in the engine.

An open source tool WEKA [79], which is an open source tool under the GNU GPL license, was used for induction to establish the rules. For the induction, classification algorithm J48 was exploited for all the datasets used in the experiments conducted for the implementation. The rules were extracted from the binary decision tree generated through the induction. Further, the rules were formulated and provided to the micro-blaze for the classification process.

The Xilinx Platform Studio was used to program the micro-blaze; and to program hardware we used Xilinx ISE 12.4. Micro-blaze was provided with the rules of the classification; where with different datasets we have different classifications rules. With each test performed the data is fed into the memory. The speed of the clock is 50 MHz, whereas our proposed hardware accelerator operates on double clock frequency that is 100 MHz.

### 7.4.1 Accuracy of the Model

The accuracy of our parallel implementation of the pipelined architecture is shown in Table 7.1. Here Iris and Contact lenses from UCI machine learning repository

[80] are the bench mark datasets, whereas synthetic datasets 1, 2 and 3, generated using Datagen [81]. A number of attributes varying from 4 to 6, are used with each configuration, having the number of tuples ranging from 100 to 1000. The results validate that our architecture supports varying number of attributes and tuples without deteriorating the accuracy of the model.

| Dataset | Total number of instances | Correctly classified | %Correctly classified |
|---|---|---|---|
| Iris | 150 | 147 | 98.00% |
| Contact Lenses | 24 | 20 | 83.30% |
| Dataset1- 4 attributes | 1000 | 1000 | 100.00% |
| Dataset2- 5 attributes | 1000 | 1000 | 100.00% |
| Dataset3- 6 attributes | 1000 | 1000 | 100.00% |

Table 7.1: The accuracy of the Decision Tree model

### 7.4.2 A Comparison with software implementations

For the comparison with the software, execution times of the decision tree classification engine is compared with WEKA data mining software, R-project and C implementation of classification process. In R-project the tree is implemented by recursive partitioning using Rpart routines and classification is performed using predict routine. The WEKA tool uses the ID3 for induction process, and performs classification on the test data. The same datasets were used for all the software and hardware implementations.

Detailed results of the study are shown in Table 7.2, presenting the time each implementation takes as well as the overall speedup/performance gains of hardware accelerator compared to software. The results show that the speed of C implementation is

in microseconds and it takes less time than WEKA and R-project. WEKA, a java based tool, shows better performance than R-project. R-project is an interpreted language which is implemented in C, but in orders of magnitude slower than specialized C implementation of the classification.

We also tested our proposed system on datasets with 4, 5 and 6 attributes by varying the number of tuples from 100 to 8000 and it was established that there is no impact of the number of attributes on the performance of the engine. This occurred mainly due to the fact that we have implemented an axis-parallel decision tree, the hardware takes the same number of cycles for classification regardless of the number of attributes of the dataset.

| **No. of Tuples** | **Time for the hardware implementation** | | | **Time for the software implementation approx.** | | | **Speedup compared to C Impleman-tation** |
|---|---|---|---|---|---|---|---|
| | No. of clock cycles | Lat-ency | Total Time (ns) | **Weka** (ms) | **R-project** (ms) | **C** (us) | |
| 100 | 7 | 4 | 220 | 0.5 | 1.33 | 12.3 | 55x |
| 250 | 17 | 4 | 420 | 0.5 | 1.38 | 33.84 | 80x |
| 500 | 32 | 4 | 720 | 1 | 1.52 | 76.92 | 106.83X |
| 750 | 48 | 4 | 1040 | 1.5 | 1.66 | 112.3 | 107.9X |
| 1000 | 63 | 4 | 1340 | 2 | 1.81 | 175.38 | 130X |
| 2000 | 125 | 4 | 2580 | 2 | 2.42 | 286.45 | 111X |
| 3000 | 188 | 4 | 3840 | 2 | 3.16 | 430.21 | 111X |
| 4000 | 250 | 4 | 5080 | 5 | 3.87 | 570.23 | 112X |
| 5000 | 313 | 4 | 6340 | 5 | 4.54 | 720.42 | 113X |
| 6000 | 375 | 4 | 7580 | 6 | 5.4 | 860.3 | 113X |
| 7000 | 438 | 4 | 8840 | 11 | 5.89 | 1006 | 113X |
| 8000 | 500 | 4 | 10080 | 15 | 6.81 | 1154 | 114X |

Table 7.2:Comparison with Software Implementations

Our design is currently limited by the locally available memory and no high speed communication link to stream data, the maximum number of dataset tested is 8000 data records. If streaming data is available decision tree classification engine is designed to process at a fixed throughput that is linearly related to data set size. Theoretically the number of clock cycles = (data records /8 + latency)+ clock cycle for switching the buffered memory. For example for the dataset of 1 million records it will take 2.5 miliseconds.

### 7.4.3 Comparison with previous hardware implementations

For the comparison with the previous hardware implementations, the clock cycles required by the FPGA implementation of decision tree classification engine are compared with the SMpL and SmpL-p implementation proposed by Struharik et al. [68]. The SMpL-p architecture employs one hardware module per level of the decision tree. In the experiments performed by Struharik et al for the classification, 16 of the 23 datasets, used are binary trees. We have performed the experiments on the subset of the datasets used in the SmpL and SmpL-P, and compared the performance in terms of the clock cycles in Table 7.3 and it shows that decision tree classification engine has on average 3.5x speedup over these implementations.

| Data set | DT | SMpL | Speedup | SMpL-P | Speedup |
|---|---|---|---|---|---|
| Glass | 2.24 | 6.46 | 2.88x | 6.96 | 3.12x |
| Balance-scale | 1.5 | 4.97 | 3.31x | 4.27 | 2.85x |
| Heart | 1.26 | 4.99 | 3.96x | 4.54 | 3.60x |
| Diabetes | 1.48 | 6.72 | 4.54x | 7.14 | 4.82x |
| ionosphere | 1.88 | 6.46 | 3.43x | 6.25 | 3.32x |
| Liver | 1.74 | 6.37 | 3.66x | 4.05 | 2.32x |
| Soner | 1.92 | 6.55 | 3.41x | 6.16 | 3.20x |
| Page block | 1.29 | 6.93 | 5.37x | 6.25 | 4.84x |
| Zoo | 2.57 | 4.99 | 1.96x | 5.88 | 2.28x |

Table 7.3: Comparison with Hardware Implementations

## 7.5 Resource Utilization

Based on the RTL level hardware requirements SmpL and SmpL-P requires M.n multipliers, whereas our implementation requires 0 multipliers. Also the number of adders for our implementation is 2.M*8 adders, whereas the SMpL requires approximately M.[ $2^n$ ] adders, where *n* is the number of attributes and *M* is the level of the trees. Hence the hardware requirements are also minimal for our proposed hardware engine.

The devised architecture area utilization, in terms of lookup tables and flip flops and the block RAM utilization is also optimized. Table 7.4 shows the utilization summary at different hierarchies of the design.

| Implementation | LUTs | FF | Block RAM |
|---|---|---|---|
| DT Engine 1 Stage | 62 | 96 | 0 |
| DT Engine 4 Stage | 240 | 332 | 0 |
| DT Engine 8 parallel instances of 4 Stages | 2952 | 3100 | 18 |
| Whole design | 6442 | 5336 | 22 |

Table 7.4: The Resource Utilization of the Decision Tree model

The utilization of number of slices of the decision tree classification engine with 8 parallel classification subsystems instances is 29%, whereas each instance of 4 stage pipelined decision tree module uses 205 slices bringing it to 2% total utilization. The whole design number of slices utilization is 5386 which is 62% utilization. Thus the proposed architecture in comparison with SMpL-p has reduced hardware complexity of the modules and reduced execution time.

## 7.6 Data Streaming with High Performance Communication Link

Our ideal architecture would consist of a streaming interface between PCIe and the decision tree classification engine. Using this interface, the host computer could set up DMA transfers to a fixed destination address on the DT peripheral and continually stream data to the limits of the communication link. At the DT peripheral, onboard logic would manage the streaming data such that a double-buffered input memory could be used to maintain constant bandwidth between the host and peripheral. In this way, the decision tree classification engine would hide the addressing complexity from the host. As long as the processing capability of the engine met or exceeded that of the

communication link, saturation would not occur.

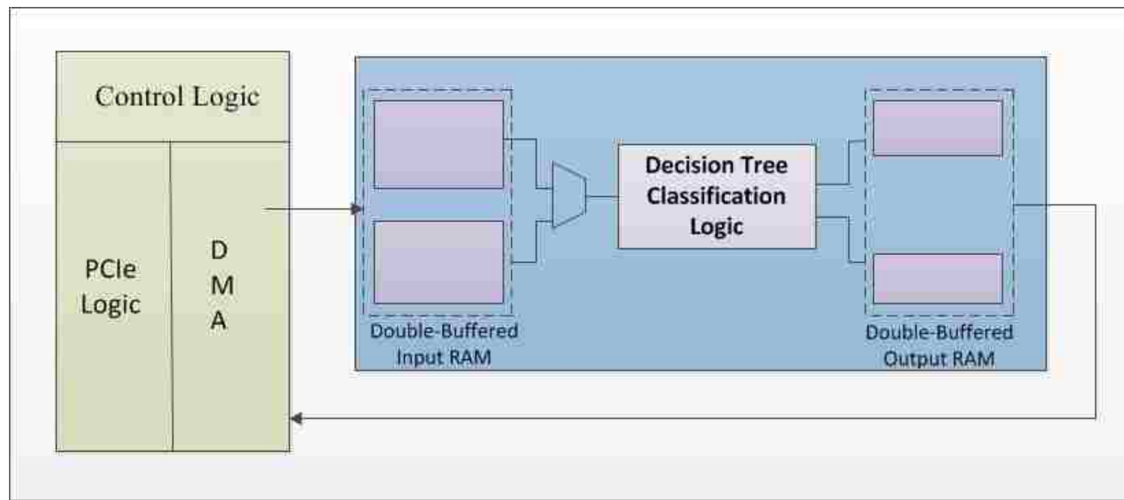


Fig. 7.6: Streaming Architecture

This architecture allows classification of big data in a streaming manner. Figure 7.6 shows the streaming architecture, the double-buffered input and output RAMs are designed to support simultaneous buffering and processing of data. A memory controller switches the first memory from buffering mode to processing mode once the memory is filled, and connects the other memory to the communication unit for buffering. In such a manner the communication overhead is hidden.

The proposed architecture has the advantage of being highly scalable and exhibits high levels of parallelism. The performance of pipelined architecture is linearly dependent on the number of data records/tuples and independent of the number of attributes in a particular data-set. Higher levels of parallelism can be achieved by increasing the number of parallel pipelines, also trees of greater depth upto 13 can be modeled by increasing the number of pipeline stages.

The design has the minimum resource utilization thus the power consumed is also

an advantage of the binary decision tree classification accelerator engine. Contrary to the previous implementations, we focused on the pipelining of different stages; efficient use of the on-chip memories; and registers to optimise the area used and minimize the clock cycles, thus helping in accelerating the process of classification.

# CHAPTER 8
# Conclusion

My research project mainly focuses on the die-to-die and within-die variation measurement for analyzing the variations using the embedded test structure REBEL. I have successfully collected data and analyzed the path delays from 52 copies of 90nm chips 28 copies of 28nm Zynq FPGAs 7000 series on Zed Boeards. I have performed various analysis to better understand the quality of each dataset; and consequently have established a methodology of calibrating short path delays and have devised a more accurate process of measuring die-to-die and within-die varirationin ASIC and FPGAs.

The key contributions of the work presented in this proposal includes:

1) The within-die variation measurement and evaluation of REBEL is carried out in multiple copies of a custom designed test chip fabricated in IBM's 90nm technology. The macro in which REBEL is integrated is an IEEE-754 compliant floating point unit (FPU), with 5 pipeline stages. Random test patterns are applied to the combinational logic within each of the pipeline stages and the measured delays are analyzed, with emphasis on evaluating the magnitude of within-die variations as a function of path length. A second important component of my

experiments is the evaluation of delay variations while the chips are subjected to industrial-level temperature and voltage (TV) variations.

2) In ASIC analysis I propose a calibration methodology and introduced windowing mechanism to avoid the voltage transient effect. The impact of the power transient is particularly evident when the LC interval is dynamically changed as a means of obtaining high resolution delay measurements. The calibration process is designed to eliminate this environmental source of delay variation.

3) Also for ASIC I show a new error estimation scheme for measuring the error estimation in path delays is proposed here, which uses different windows to measure the same path, where each subsequent window increases the launch capture interval and adds another flip-flop in the path-under-test.

4) For FPGAs I have performed analysis on the propagational delay through the flip flops for both rising and falling edges and have confirmed the robust design where the delays are overlapped.

5) Path Distribution of the AES design implemented on FPGA and a comparison of common paths among all copies of FPGAs and how many of them are unique and are present only in 1 chip.

6) I have performed die-to-die and within-die variation measurement and evaluation of REBEL on 28 copies of Zynq 7000 series FPGAs on ZED boards and shown that on average a 5% of percentage change variation exists with in the FPGAs.

In my research, I present REBEL (regional delay behavior) as an embedded test

structure (ETS), for path delay measurement which is later utilized for measuring within-die and die-to-die variation. I have described the detailed architecture of REBEL and demonstrated its effectiveness for measuring delays and capturing the within-die variations caused by the environmental and physical process variations. The experimental results obtained so far are elaborated in Chapter 5 and Chapter 6 for custom 90nm chip and 28nm Zynq FPGA analysis respectively. There are several applications, for example, Trojan detection, delay defects detection, and Physical Unclonable Functions for encryption, identification and authentication.

My experimental results presented in Chapter 5 show that the magnitude of within-die delay variations in ASIC is dependent on the length of the path, and the delays are highly sensitive to the power transient effect introduced by the launch-capture (LC) clock event. Additionally I have performed uncertainty analysis to estimate the noise contribution and specified a region where the paths with 4 to 12 flip-flops are included in the uncalibrated path delay have the minimum uncertainty.

In Error estimation Analysis for measuring the error estimation in path delays I use different windows to measure the same path, ideally all the windows should get the same path delay after calibration, but this is not the case as the calibration error is added with every additional flip-flop of the scan chain. So far the errors are within ± 100 ps but they can increase upto -300ps in worst scenario. Also in our designed chip power grid noise is contributing more variation in the calculated path delays, as the bypass capacitance on the chip was not included in the design. To minimize the error I have added an external bypass capacitance on the I/O pins still a large transience in the supplied power voltage is observed. The clock for launch capture intervals is provided by

FPGA using DCMs with fine phase adjustment. We observe the clock jitter and variation in the launch capture intervals in clock strobing. To account for this variability the exact LCI is recorded using oscilloscope measurements and are used in the calculations instead of using the estimated value. Hence reducing the errors in the calibration of path delays.

In Chapter 6, I performed die-to-die and within-die variation analysis on the Zynq FPGAs and have shown that the the did-to-die variation have spatial dependencies and can range from 14% to 24% on average. The within-die variations in FPGA for the path delays is on average 5% and can go upto 27%,. This analysis will help in the variation aware layout design to avoid the violations and improve the performance. The experiments that have been performed on the calibrated path delays are Flip flop rising and falling transition delay analysis, uncertainty analysis, path distribution analysis, short versus long path variations and within-die variation analysis and die to die variations on the data collected so far.

Additionally, to cater for big data by employing decision tree classification (DTC) to speed-up the classification step in hardware implementation, we devised a pipelined architecture for the implementation of axis parallel binary decision tree classification for meeting up with the requirements of execution time and minimal resource usage in terms of area. Our hardware acceleration of pipelined architecture incorporates the parallel approach in acquiring the data by having parallel engines working on different partitions of data independently. Also, each engine is processing the data in a pipelined fashion to utilize the resources more efficiently and reduce the time for processing all the data records/tuples.

# CHAPTER 9
# Future Work

In this chapter, I suggest further research work in the area of path variations, based on our current research. Particular areas for future research to be focused *inter alia* include path delay measurements using IDCs, path delay measurements as an entropy source for PUF primitive, defect analysis, model to hardware correlation, and on-chip using DLLs or PLLs and bypass capacitance on I/O pads.

## 9.1 Path Delay Measurement using TDCs

Within die delay variation analysis using REBEL 0is performed on the path delays with a measurement precision having a resolution of 70ps, Time-to-Digital TDC is a test structure which can provide the path delay measurement with 10ps precision. The TDC is designed to measure the relative delay between two input signals which are provided by a pair of tap points on an FPU macro. The relative delay is digitized by the TDC using a pulse-shrinking mechanism. The digital code is 'scanned out' of the TDC and the width of the transition propagated is compared with the ring oscillator frequency to calculate the delay, pulse-shrinking behavior of the TDC allows very high timing resolution, i.e., 10's picoseconds, in measurements of the width of the input pulse. The TDC occupies an

area of 176 um x 60 um (10k um2).

The TDC is implemented as two components, labeled Path Select/Pulse Gen Unit and Pulse Shrinking Delay Chain. On chip there are in total seven TDCs, one connected with Arbitor PUF, 2 with FPU macro-under-test, 2 with AES and 2 with AES with Trojan. All the TDC are connected in a chain and hence require configuration to implement FPU with TDCs. There is a Ring Oscillator whose input can go in the TDC, we can measure its frequency as it is connected with one of the outputs. Hence a table mapping the frequency of the ring oscillator and its thermometer code is once created, can be used to measure the delays of the paths by comparing there thermometer code within this table and get the quantized delay as an output.

The component pulse generation unit has scan flip flops sel A and sel B drives the inputs of two 8-to-1 Muxes. Each of these inputs requires a transition. The combination of these transitions can be both rising, both falling, one rising and one falling or vise versa.  The outputs of the 8-to-1 MUXes is given to a negative pulse generator, XNOR gate, for the Pulse Shrinking Delay Chain.

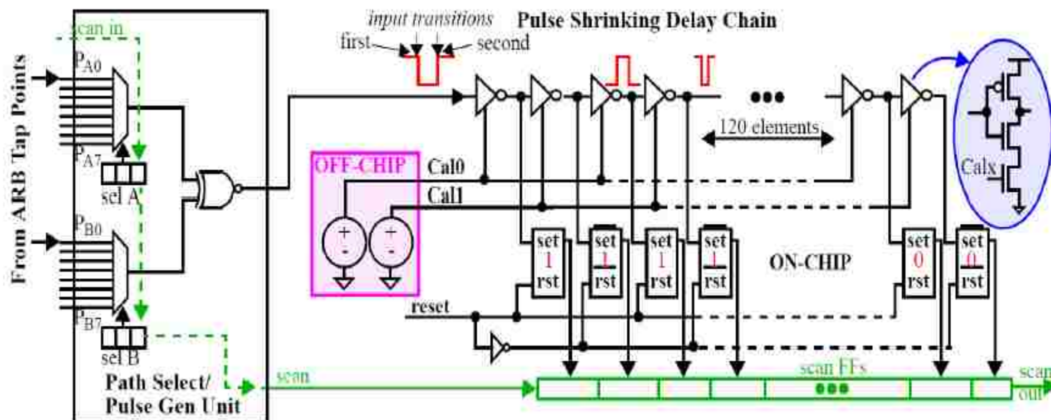


*Fig. 9.1:  Time to Digital Conversion (TDC)*

The current starved inputs of all the even numbered inverters are connected to Cal0 while the inputs of the odd numbered inverters are connected to Cal1. With Cal0 fixed at a specific voltage, larger assigned Cal1 voltages allows the first edge propagates more quickly and hence makes the pulse disappear after the trailing edge catches up with the leading edge. A set of flip-flops up stores '1' to the point where the pulse disappears , while those beyond this point store ’0’. In this set of experiments, both of these voltages are controlled using off-chip power supplies.

In case of calibration, the rising and falling edges of ring oscillator are provided to the TDC and simultaneously the frequency of the ring oscillator is measured from the output pins using oscilloscope. The TDC is designed to ’pulse shrink’ the negative output pulse from the XNOR as it propagates down a current-starved inverter chain. As the pulse moves down the inverter chain, it activates a corresponding set of set-reset latches to record the passage of the pulse, where activation is defined as storing a ’1’. A thermometer code, i.e., a sequence of ’1’s followed by a sequence of ’0’s, represents the digitized delay between the rising edge and falling edge of the ring oscillator. In this context the digitized delay is called thermometer code is zeros followed by all '1’s. Once the calibration process is complete we get a mapping of frequency and thermometer code.

In the path delay measurement process, the inputs of scan flip flops, sel A and sel B is connected back to the FPU macro under test. The thermometer code of the delay is mapped with the history table and the delays are digitized.

## 9.2 Path Delay Measurement as an Entropy Source for PUF Primitive

Within-die variations in path delays are increasing with scaling, and are

increasingly affected by “neighborhood” interactions. Although higher levels of within-die delay variations are undesirable from a design perspective, they represent a rich source of entropy for applications that make use of ‘secrets’, such as authentication, hardware metering and encryption. Physical Unclonable Functions or PUFs are a class of circuit primitives that leverage within-die variations as a means of generating random bit strings for these types of applications.

We can implement hardware embedded delay PUF that leverages within-die path delay information. PUF obtains accurate measurements of path delays within core logic macros using an embedded test structure called REBEL. REBEL provides capabilities similar to an off-chip logic analyzer, and allows very fast analysis of the temporal behavior of signals emerging from paths in a core logic macro. Statistical characteristics related to the randomness, reproducibility and uniqueness of the bit strings which can be used as key for encryption and authentication applications.

The FPA timing value that we obtain for the stable paths can be used as a PUF Numbers or PNs. The statistical analysis requires the number of bits to be equal across all chips, so we can reduce the number of PNs to the smallest number produced by one of our chips, which is X. Using all combinations in the bit generation process, this allows bit strings of length $X*(X-1)/2 = Y$ . The actual bitstring size however is dependent on the bit string generation methods. Some of the methods for bit generation are, Dual P/N, Dual P/N Count (DPNC), Universal No- Modulus Method (UNM) and Universal No-Modulus Difference. There are thresholding techniques to avoid the weak bits in the bit string during the enrollment to avoid the bit flips during re-generation. To generate error free bitstring TMR technique can be incorporated to generate an error-tolerant bit string. That

PUF can be evaluated across industrial-level temperature and supply voltage variations based on the data collected for the chips.

## 9.3 Defect Analysis

Cadence Encounter Test ATPG tool is used to generate transition/path delay test vectors for various path delay measurements. Several path delays that represent various path lengths of the design are measured using REBEL embedded test structure. These path delays can be measured in 52 copies of the chip and analyzed for die-to-die and within-die variations. In these macros several defect emulation circuits, that are designed to introduce delay anomalies along the selected paths, are inserted. An analog control input is added in the defect Emulation circuit that enables the controlled insertion of additional capacitive load that models a defect. With this mechanism, once the input patterns create a falling/rising edge transition, the delay will wary from the path with no defect. In the test vectors generated using ATPG 6 paths are tested which produce a transition on the paths with the defect. REBEL can be used to measure the delays along these paths, which are then processed for detecting the affects of defects.

Further more incorporating the correct flip flop delay based on the launch capture interval to remove the power transient from the analysis will provide more accurate delays and will give us the ability to calculate the delay variation introduced by the defect.

## 9.4 Model to Hardware Correlation

Focusing on the application of model to hardware correlation, Cadence tools for measuring the path delays with the same input patterns provided to the chip and comparing and correlating the results of the hardware data with simulations can provide

meaningful results and can help in creating the variation models. Encounter Test is a Cadence tool that helps in the analysis of the paths which are actually tested, thus figuring out the paths and correlating the path delays with the gates. Analysis on the gate composition and number of gates in each path are beneficial for more accurate variation models. Static timing analysis and statistical timing analysis of the design for manufacturing (DFM) can also be performed to improve the design for manufacturing and increasing the yield.

## 9.5 On-Chip Clock using DLLs or PLLs and Bypass Capacitance on I/O Pads

In the experiments perform on the ASIC chip, the clock for launch capture intervals is provided by FPGA using DCMs with fine phase adjustment. We observe the clock jitter and variation in the launch capture intervals in clock strobing. To account for this variability we record the exact LCI using oscilloscope measurements and use them in the calculations instead of using the estimated value. Hence reducing the errors in the calibration of path delays but not completely eliminating it. A better solution would be to have an engine for clock strobing on chip and generating the launch capture interval with less clock jitter and minimized the clock skew.

Also in our designed chip power grid noise is contributing more variation in the calculated path delays, as the bypass capacitance on the chip was not included in the design. We have added an external bypass capacitance on the I/O pins. The incorporation of by-pass capacitance on chip will help in reducing the power supply transient effect

# References

[1] Nassif, S.R., "*Modeling and Analysis of Manufacturing Variations",* IEEE Conference on Custom Integrated Circuits (CIC 2001), San Diego, CA, USA, May 2001, pp. 223-228.

[2] I. Ahsan, N. Zamdmer, O. Glushchenkov, R. Logan, E.J. Nowak, H. Kimura, J. Zimmerman, G. Berg, J. Herman, E. Maciejewski, A. Chan, A. Azuma, S. Deshpande, B. Dirahoui, G. Freeman, A. Gabor, M. Girbelyuk, S. Huang, M. Kumar, K. Miyamoto, D. Mocuta, A. Mahorowala, E. Leobandung, H. Utomo, and B. Walsh, "*RTA-Driven Intra- Die Variations in Stage Delay and Parametric Sensitivities for 65 nm Technology",* Digest of Technical Papers, 2006 Symposium on VLSI Technology, Honolulu, HI, USA, June 2006, pp. 170-171.

[3] T.J. Yamaguchi, J.A. Abraham, G.W. Roberts, S. Natarajan, and D. Ciplickas, "*Panel Session 12B: Post-Silicon Validation and Test in Huge Variance Era*", 2013 IEEE 1st VLSI Test Symposium (VTS), Berkeley, CA, USA, April-May 2013, pp.1.

[4] R. Raina, "*What is DFM & DFY and Why Should I Care?*" IEEE International Test Conference (ITC), Santa Clara, CA, USA, October 2006, pp. 1-9.

[5] D.G. Chesebro, J.W. Adkisson, L.R. Clark, S.N. Eslinger, M.A. Faucher, S.J. Holmes, R.P. Mallette, E.J. Nowak, E.W. Sengle, S.H. Voldman, and T.W. Weeks, "*Overview of Gate Linewidth Control in theManufacture of CMOS Logic Chips*", IBM Journal of Research and Development, vol. 39 no. 1/2, 1995, p. 189-200.

[6] J.Y. Lai, N. Saka, and J-H Chun, "*Evolution of copper-oxide damascene structures in chemical mechanical polishing*", J. of Electro-Chem. Soc., vol. 149,

no. 1, 2002: p. G31- G40.

[7] C. Hedlund, H. O. Blom, and S. Berg, "*Microloading effect in reactive ion etching*", J. of Vacuum Science and Tech, vol. 12, no. 4, 1994, pp. 1962-1965.

[8] D. Burek, "*True design-for manufacturability critical to 65-nm design success*", http://www.eetimes.com/showArticle.jhtml?arti-cleID=202803596, Nov. 2007.

[9] S. Paul, S. Krishnamurthy, H. Mahmoodi, and S. Bhunia, "*Low-overhead design technique for calibration of maximum frequency at multiple operating points*", IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, Nov. 2007, pp. 401-404.

[10] W. Xiaoxiao, M. Tehranipoor, and R. Datta, "*Path-RO: A novel on-chip critical path delay measurement under process variations"*, IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, Nov. 2008, pp. 640-646.

[11] C. Lamech, J. Aarestad, J. Plusquellic, R. Rad, and .K. Agarwal, "*REBEL and TDC: Two embedded test structures for on-chip measurement of within-die path delay variations*", IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, Nov. 2011, p. 170-177.

[12] J. Li, and J. Lach, "*At-speed delay characterization for IC authentication and Trojan Horse detection*", IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, June 2008, pp. 8-14.

[13] H. Yan, and A.D. Singh, "*Experiments in detecting delay faults using multiple higher frequency clocks and results from neighboring die*", in Proc. IEEE International Test Conference (ITC), 2003, pp. 105-111.

[14] J. Aarestad, P. Ortiz, J. Plusquellic and D. Acharyya, "*HELP: A hardware-embedded delay PUF*", IEEE Design and Test, March/April 2013, vol. 30, no. 2, pp. 17-25.

[15] J. Aarestad, C. Lamech, J. Plusquellic, D. Acharyya and K. Agarwal, "*Characterizing within-die and die-to-die delay variations introduced by process variations and SOI history effect*", in Proc. 48th Design Automation Conference (DAC), San Diego, CA, USA, June 2011, pp. 534-539.

[16] R. Narayanan, D. Honbo, G. Memik, A. Choudhary, J. Zambreno, "*An FPGA implementation of decision tree classification*", in Proc. IEEE International Design, Automation and Test in Europe Conference and Exhibition (DATE), Nice, France, April 2007, pp. 1-6.

[17] T. Tuan, A. Lesea, C. Kingsley, S. Trimberger, "*Analysis of within-die process variation in 65nm FPGAs*", IEEE 12th International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, March 2011, pp. 1-5.

[18] B.P. Das, B. Amrutur, H.S. Jamadagni, N.V. Arvind, V. Visvanathan, "*Within-die gate delay variability measurement using re-configurable ring oscillator*", IEEE Custom Integrated Circuits Conference (CICC), San Jose, CA, USA, September 2008, pp. 133-136.

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

[20] K. Katsuki, M. Kotani, K. Kobayashi, and H. Onodera, "*Measurement results of within-die variations on a 90nm LUT array for speed and yield enhancement of*

*reconfigurable devices*", in Proc. 6th IEEE Asia and South Pacific Design Automation Conference (ASP-DAC), Yokohama, Japan, January 2006, pp. 110-111.

[21] P. Sedcole, and P.Y.K. Cheung, "*Within-die delay variability in 90nm FPGAs and beyond*", IEEE International Conference on Field Programmable Technology (FPT), Bangkok, Thailand, December 2006, pp. 97-104.

[22] D.J. Kinniment, O.V. Maevsky, A. Bystrov, G. Russel, and A.V. Yakovlev, "*On-chip structures for timing and measurement*", in Proc. IEEE 8th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), Manchester, UK, April 2002, pp. 190-197.

[23] X. Zhang, K. Ishida, M. Takamiya, and T. Sakurai, "*An on-chip characterizing system for within-die delay variation measurement of individual standard cells in 65-nm CMOS*", IEEE 16th Asia and South Pacific Design Automation Conference (ASP-DAC), Yokohama, Japan, January 2011, pp. 109-110.

[24] P. Dudek, S. Szczepanksi, and J.V. Hatfield, "*A high-resolution CMOS time-to-digital converter utilizing a Vernier delay line*", IEEE Journal of Solid-State Circuits, vol. 35, no. 2, February 2000, pp. 240-247.

[25] C.C. Chen, P. Chen, C.S. Hwang, and W. Chang, "*A precise cyclic CMOS time-to-digital converter with Low thermal sensitivity*", IEEE Transactions on Nuclear Science, vol. 52, no. 4, August 2005, pp. 834-838.

[26] A. Mantyniemi, and T. Rahkonen, and J. Kostamovaara "*A CMOS time-to-digital converter (TDC) based on a cyclic time domain successive approximation interpolation method*", IEEE Journal of Solid-State Circuits, vol. 44, no. 11,

November 2009, pp. 3067-3078.

[27] R. Datta, A. Sebastine, A. Raghunathan and J.A. Abraham, "*On-chip delay measurement for silicon debug*", in Proc. 14th ACM Great Lakes Symposium on VLSI (GLSVLSI), April 2004, pp. 145- 148.

[28] R. Datta, G.D. Carpenter, J.K. Nowka and J.A. Abraham, "*A scheme for on-chip timing characterization*" in Proc. IEEE 24th VLSI Test Symposium (VTS), Berkley, CA, USA, April-May 2006, pp. 24-29.

[29] D. Acharyya, K. Agarwal, and J. Plusquellic, *"Leveraging existing power control circuits and power delivery architecture for variability measurement",* in Proc. IEEE International Test Conference (ITC), Austin, TX, USA, November 2010, pp. 1-9.

[30] X. Zhang, K. Ishida, H. Fuketa, M. Takamiya, and T. Sakurai, *"On-chip measurement system for within-die delay variation of individual standard cells in 65-nm CMOS",* IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 10, October 2012, pp. 1876-1880.

[31] L.T. Pang, and B. Nickolic, *"Measurements and analysis of process variability in 90 nm CMOS"*, IEEE Journal of Solid-State Circuits, vol. 44, no. 5, May 2009, pp. 1655 – 1663.

[32] W. Grobman, M. Thompson, R. Wang, C. Yuan, R. Tian, and E. Demircan, *"Reticle enhancement technology: implications and challenges for physical design",* in Proc. IEEE Design Automation Conference (DAC), 2001, pp. 73-78.

[33] R. Takahashi, H. Takata, T. Yasufuku, H. Fuketa, M. Takamiya, M. Nomura, H. Shinohara, and T. Sakurai, *"Large within-die gate delay variations in sub-*

*threshold logic circuits at low temperature"*, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 59, no. 12, December 2012, pp. 918 – 921.

[34] L.T. Wang, C.W. Wu, and X. Wen, *"VLSI test principles and architectures: design for testability (Systems on Silicon)"*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[35] K.A. Bowman, A.R.S. Alameldeen, S.T. Srinivasan, and C.B. Wikerson, *"Impact of die-to-die and within-die parameter variations on the clock frequency and throughput of multi-core processors"*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 17, no. 12, December 2009, pp. 1679-1690.

[36] D.J. Palframan, S.K. Nam, and M.H. Lipasti, *"Mitigating random variation with spare RIBs: Redundant intermediate bitslices"*, IEEE/IFIP 42nd Annual International Conference on Dependable Systems and Networks (DSN), Boston, MA, USA, June 2012, pp. 1-11.

[37] S.K. Mehr, A.R.A. Mehr, S.N. Mozaffari, A. Afzali-Kusha, "*A new block-based SSTA method considering within-die variation*", IEEE 2nd Asia Symposium on Quality Electronic Design (ASQED), Penang, Malaysia, August 2010, pp.260-263.

[38] G.E. Suh, and S. Devadas, "*Physical unclonable functions for device authentication and secret key generation*", in Proc. IEEE 44th annual Design Automation Conference (DAC), San Diego, CA, June USA, 2007, pp. 9-14.

[39] T. Holotyak, S. Voloshynovskiy, O. Koval, and F.Beekhof, "*Fast physical object identification based on unclonable features and soft fingerprinting*", in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing

(ICASSP), Prague, Czech Republic, May 2011, pp. 1713-1716.

[40] N. Potlapally, "*Hardware Security in Practice: Challenges and Opportunities*", 2011 IEEE International Workshop on Hardware oriented Security and Trust (HOST), San Diego, CA, USA, June 2011, pp. 93-98.

[41] P. Simons, E. Sluis, and V. Leest, "*Buskeeper PUFs, a promising alternative to D flip-flop PUFs*", 2012 IEEE International Symposium on Hardware oriented Security and Trust (HOST), San Francisco, CA, USA, June 2012, pp. 7-12.

[42] K. Kursawe, A.R. Sadeghi, D. Schellekens, B. Skoric, and P. Tuyls, "*Reconfigurable Physical Unclonable Functions – Enabling Technology for Tamper-Resistant Storage*", 2009 IEEE International Symposium on Hardware oriented Security and Trust (HOST), San Francisco, CA, USA, July,2009, pp. 22-29.

[43] M. Bhargava, C. Cakir, and K. Mai, "*Comparison of Bi-stable and Delay-based Physical Unclonable Functions from Measurements in 65nm bulk CMOS*", 2012 IEEE Custom Integrated Circuits Conference (CICC), San Jose, CA, USA, September 2012, pp 1-4.

[44] H. Handschuh, "*Hardware intrinsic security based on SRAM PUFs: Tales from the industry*", 2011 IEEE International Symposium on Hardware oriented

Security and Trust (HOST), San Diego, CA, USA, June 2011, pp. 127.

[45] A. Maiti, and P. Schaumont, "*Improving the quality of a physical unclonable function using configurable ring oscillators*", in proc. IEEE International Conference on Field Programmable Logic and Applications (FPL), Praigue, Czech, August-September, 2009, pp.703-707.

[46] R. Kumar, H.K. Chandrikakutty, and S. Kundu, "*On improving reliability of delay based physically unclonable functions under temperature variations*", 2011 IEEE International Symposium on Hardware oriented Security and Trust (HOST), San Diego, CA, USA, June 2011, pp. 142-147.

[47] R. Kumar, V.C. Patil and S. Kundu, "*On design of temperature invariant physically unclonable functions based on ring oscillators*", 2012 IEEE Computer Society Annual Symposium on VLSI, Amherst, MA, USA, August 2012, pp. 165-170.

[48] Y. Hori, H. Kang, T. Katashita, and A. Satoh, "*Pseudo-LFSR PUF: A compact, efficient and reliable physical unclonable function*", 2011 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, Nov-Dec 2011, pp. 223-228.

[49] G. Schrijen, and V. Leest, "*Comparative analysis of SRAM memories used as PUF primitives*", IEEE Design, Automation and Test in Europe Conference & Exhibition (DATE), Dresden, Germany, March 2012, pp. 1319-1324.

[50] L. Lin, S. Srivathsa, D. K. Krishnappa, P. Shabadi, and W. Burleson, *"Design*

*and validation of arbiter-based PUFs for sub-45-nm low-power security applications*", IEEE Transactions on Information Forensics and Security, vol. 7, no. 4, August 2012, pp. 1394-1403.

[51] J. Ju, Jim Plusquellic, R. Chakraborty, and R. Rad, "*Bit String Analysis of Physical Unclonable Functions based on Resistance Variations in Metals and Transistors*", 2012 IEEE International Symposium on Hardware oriented Security and Trust (HOST), San Francisco, CA, USA, June 2012, pp. 13-20.

[52] R. Kumar, and W. Burleson, "*PHAP: Password based hardware authentication using PUFs*", 2012 IEEE/ACM 45th International Symposium on Microarchitecture Workshops(MICROW),Vancouver,BC,Canada, December 2012, pp. 24-31.

[53] C. Yin, G. Qu, and Q. Zhou, "*Design and implementation of a group-based RO PUF*", IEEE Design, Automation and Test in Europe Conference & Exhibition (DATE), Grenoble, France, March 2013, pp. 416-424.

[54] J. Aarestad, J. Plusquellic, and D. Acharyya, "*Error-tolerant bit generation techniques for use with a hardware-embedded path delay PUF*", 2013 IEEE International Symposium on Hardware oriented Security and Trust (HOST), Austin, TX, USA, June

2012, pp. 151-158.

[55] S.M.A.B. Mokhtar, and W.F.H.W. Abdullah, "*Memristor Based Delay Element Using Current Starved Inverter*", IEEE Regional Symposium on Micro and Nanoelectronics (RSM 2013), Langkawi, Malaysia, September 2013, pp. 81-84.

[56] G.S. Rose, N. McDonald, L.K. Yan, B. Wysocki, and K. Xu,"*Foundations of memristor based PUF architectures*", 2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), Paris, France, July 2013, pp. 52-57.

[57] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[58] C.M. Bishop, "*Neural networks for pattern recognition*", Oxford University Press, Oxford, UK, 1995.

[59] L. Rokach, and O. Maimon, "*Top-down induction of decision trees – A Survey*", IEEE Trans. on Systems, Man and Cybernetics, Part C, vol. 35, no. 4, November 2005, pp. 476-487.

[60] V.N. Vapnik, "*Statistical Learning Theory*", Wiley & Sons Inc., New York, September 1998, 768 pp.

[61] D. C. Hendry, A. A. Duncan, and N. Lightowler. "*IP core implementation of a self-organizing neural network*" in Proc. IEEE Trans. on Neural Networks, vol. 14, no. 5, September 2003, pp. 1085-1096.

[62] S. Himavathi, D. Anitha, and A. Muthuramalingam, "*Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization*", IEEE Trans. on Neural Networks, vol. 18, no. 3, May 2007, pp. 880-888.

[63] D. Anguita, S. Pischiutta, S. Ridella, and D. Sterpi, "*Feed-forward support vector machine without multipliers*", IEEE Trans. on Neural Networks, vol. 17, no. 5, September 2006, pp. 1328-1331.

[64] J. Han and M. Kamber, "*Data mining: Concepts and techniques*", Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006, 772 pp.

[65] L. Breiman, J. H. Freidman, C. J. Stone, and R. A. Olshen, "*Classification and regression trees*", Chapman and Hall/CRC, Boca Raton, FL, USA, January 1984, 368 pp.

[66] A. Bermak, and D. Martinez, "*A compact 3D VLSI classifier using bagging threshold network ensembles*", IEEE Trans. on Neural Networks, vol. 14, no. 5, September 2003, pp. 1097-1109.

[67] S. Lopez-Estrada, and R. Cumplido, "*Decision tree based FPGA-architecture for texture sea state classification*", in Proc. IEEE International Conference on Reconfigurable Computing and FPGAs (ReConFig), San Luis Potosi, Mexico, September 2006, pp. 191-197.

[68] J.R. Struharik, "*Implementing decision trees in hardware*", in Proc. IEEE 9th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, September 2011, pp. 41-46.

[69] Z. Baker and V. Prasanna, "*Efficient hardware data mining with the Apriori*

*algorithm on FPGAs*”, in Proc. IEEE 13th Symposium on Field Programmable Custom Computing Machines (FCCM), Seattle, WA, USA, April 2005, pp. 3-12.

[70] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo, “*Fast discovery of association rules*”, in Book Advances in Knowledge Discovery and Data Mining”, American Association for Artificial Intelligence (AAAI)/MIT Press, Menlo Park, CA, USA vol. 12, no. 1, 1996, pp. 307–328.

[71] Z.K. Baker and V.K. Prasanna, “*An architecture for efficient hardware data mining using reconfigurable computing systems*”, in Proc. IEEE 14th annual Symposium on Field Programmable Custom Computing Machines (FCCM), Napa, CA, USA, April 2006, pp. 67-75.

[72] J.C. Shafer, R. Agrawal, and M. Mehta, “*SPRINT: A scalable parallel classifier for data mining*”, in Proc. Society for Industrial and Applied Mathematics (SIAM) 3rd International Conference on Very Large Databases (VLDB), Mumbai (Bombay), India, September 1996, 544-555.

[73] M.V. Joshi, G. Karypis, and V. Kumar, “*ScalParC: A new scalable and efficient parallel classification algorithm for mining large datasets*”, in Proc. IEEE 11th International Parallel Processing Symposium (IPPS), Orlando, FL, USA, March-April 1998, pp. 573-579.

[74] Q. Li and A. Bermak, “*A low-power hardware-friendly binary decision tree classifier for gas identification*”. Journal of Low Power Electronics and Applications, vol. 1, no. 1, April 2011, pp. 45-58.

[75] T.O. Bachir, M. Sawan, and J. Brault, “*A new hardware architecture for sampling the exponential distribution*”, in Proc. IEEE Canadian Conference on Electrical

and Computer Engineering, Niagara Falls, ON, Canada, May 2008, pp 1393-1396.

[76] V. Podgorelec, and P. Kokol, “*Evolutionary induced decision trees for dangerous software modules prediction*”, Information Processing Letters, Elsevier Science Publishers B.V., vol. 82, no. 3. February 2002, pp. 31-38.

[77] G. Chrysos, P. Dagritzikos, I. Papaefstathiou, and A. Dollas, “*Novel and highly efficient reconfigurable implementation of data mining classification tree*”, in Proc. IEEE 21st International Conference on Field Programmable Logic and Applications, Chania, Greece, September 2011, pp. 411-416.

[78] G. Chrysos, P. Dagritzikos, I. Papaefstathiou, and A. Dollas, “*HC-CART: A parallel system implementation of data mining classification and regression tree (CART) algorithm on a multi-FPGA system*”, Association for Computing Machinery (ACM) Transactions on Architecture and Code Optimization (TACO), vol. 9, no. 4, article 47, 25 pp, January 2013.

[79] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, “*The WEKA data mining software: An update*”, Association for Computing Machinery (ACM) SIGKDD Explorations Newsletter, volume 11, issue 1, June 2009, pp. 10-18.

[80] A. Frank and A. Asuncion, *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2010.

[81] G. Melli, “*Dataset Generator (Datgen)*”, version 3.1, 1999 http://www.datasetgenerator.com