Precise On-Chip Clock Skew
Measurement using Sub-sampling
and Applications

by
Pratap Kumar Das

Submitted to the Department of Electrical Communication
Engineering
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
INDIAN INSTITUTE OF SCIENCE

September 2012
To

My Mother

and

Motherland
Declaration

I, hereby declare that the work reported in this thesis has been carried out in VLSI Circuits and Systems Lab, Electrical Communication Engineering Department, under the supervision of Dr. Bharadwaj Amrutur. I also declare that this work does not form the basis for the award of any Degree, Diploma, Fellowship, Associateship or similar title of any University or Institution.

Pratap Kumar Das,
VLSI Circuits and Systems Lab,
Electrical Communication Engineering Department,
Indian Institute of Science,
Bangalore-560012,
India
September 2012
Acknowledgements

The years I have spent in IISc have been a wonderful journey, replete with righteous
learning, true loving and happy living. This would not have been possible without
the aid, advice, and support of many of the individuals, whom I want to put in record
through this thesis.

First and foremost, I would like to thank my advisor, Prof. Bharadwaj Amrutur
for his pacific patience, invaluable guidance, and continuing encouragement through-
out the journey of my Ph.D. There have been many instances of me being found in
low-level state. A short meeting with him has always helped me to recover from it,
and fills me with fresh and positive energy and optimism to work harder till the goal
is reached. I am really indebted to him for moulding my scientific career and enabling
me to write the thesis in the final form.

I would like to thank Prof. Navakanta Bhat for his constant encouragement all
through the years. I would like to thank Kuruvilla sir from CEDT for helping me
through his course and also many discussions on FPGA related issues. I would also
like to thank Prof. K. J. Vinoy for his technical input to improve my experimental
setup and also providing some components required for that.

Making a chip with TI in 65 nm process was a wonderful opportunity to learn. I
am thankful to Dr. Vish Visvanathan and Mr. J. Sridhar for their able guidance, sug-
gestions and management during the chip tape out at Texas Instruments, Bangalore.
The team I interacted within TI was very generous. I would like to have a special
mention of the following names from TI: Amit Jindal, Senthil Kumar S, Girishankar,
Jayamurugan, Sujan Manohar, Prashob Kumar.

I have always enjoyed the company of lively and supportive people around me in
the lab which has helped immensely in reducing the stress during various phases of the
PhD. I would like to thank the lab seniors, especially Vijay, Sudhir and Balaji from whom I learnt the initial flow of a chip tapeout. Satyam has always been a source of advice and encouragement for me since my first day in lab. Nayan, Sandeep and Shobi were wonderful project partners to work with during my course work. Extracting the results from the TI chip would not have been possible without the hard work and support of Bishnu. Janakis involvement in the project ensured that we could meet the stiff deadlines for the chip tape out in TI. Thanks to Nandish for many deep and constructive discussions in the middle of the night. Thanks are also due to Viveka and Karthik for their thorough proofreading and the insightful discussions in many occasions. I am thankful to Syam, Yasasvi and Bhavin for their support in the board design aspects. Thanks are also due to my lab mates Rajath, Siva, Basavaraj (BT), Tejasvi, Vikram, Kaushik Ghosal, Manikandan, Abhishek, Mohan, Pramod, and Aditya. I am thankful to Anindyu, from Mathematics department for many stimulating discussions.

I am thankful to Sabyasachi, who in many ways was instrumental in me getting into IISc and for his unrelenting moral support during my initial stressful stage at the institute. His visit to the campus, towards the end of my PhD, also helped in converging faster towards the completion of my thesis. Thanks to Sankarsan for always being besides me in the moments of joy and stress.

I am thankful to my Oriya friends, seniors and juniors, especially Souren, Saumya, Ramakant, Sudhansu, Debabrata, Sanjay, Sudarshan, Manas, Anirudh, Uttam, Ashish, Sumit and others in the campus whose associations have made my stay pleasant and enjoyable. I have always cherished their company that made me feel this campus like ‘a home away from home’.

The ECE cricket team “Silicon XI” and Oriya Sansad Cricket association, “OSCA” needs special mention in my acknowledgment. Thanks to Neeraj, Salil, Harsha, Alok, Dipti, Jayakrishna, Sudhansu, Anant, Debakanta, Tapan, Jaishankar, Venkat and others for some of the most enjoyable moments in the cricket ground and also helping me out in burning some fat through cricket.

Working for the groups like Notebook drive group and Students council in IISc has given me a lot of satisfaction for being able to contribute little back to society.
around us. I would specially like to mention the names of Siva, Shailesh, Sangeeta, Paromita, Sreevalsa, Manoj and Rishikesh for being a great team to work with.

I would like to acknowledge Department of Science and technology for funding my travel and registration to present my paper in A-SSCC 2008. Special gratitude goes to the ECE department office staff, Mr. Srinivas Murthy and Mr. Dwaraknath for keeping cumbersome official formalities smooth and simple.

A supportive family is a gift by Almighty and makes each moment in every walk of life, to pass with joy and happiness. I am fortunate enough to have such a family. I lovingly thank my mother for backing me when I expressed my interest to pursue Ph.D and later on encouraging me to come with flying colours at each and every step of my Ph.D career. My father, sisters and brother-in-law have supported me in all respects during my PhD and I am really indebted to them. It is the patriotic and society conscious feelings of my mother that has moved my heart to a great extent and I have always been driven to work for the good and noble causes. I really owe to my mother for being able to inculcate such virtues in me. When I told my mother that I am going to present a noteworthy paper in the Asian solid-state circuits conference, selected for the first time from India, she expressed her satisfaction and wished me to go further for bringing glory to the nation. I would like to take this opportunity to dedicate my thesis to my mother and my motherland. Finally, I am for ever indebted to all my family members, kith and kin for their undiminished love, affection, encouragement and unrestrained support all through the years.
Abstract

Since the evolution of digital integrated circuits, delay has been one of the critical parameters of interest for designers. With technology scaling and increase in speed of operation, predicting the delay between the clock signals inside the chip accurately has become tougher. Thus, a need of on-chip skew measurement arises. At the same time, since the recent CMOS fabrication processes are optimized for digital technology, there are attempts made to get the desired functionalities in a digital way. In this thesis work, an all-digital scheme for measuring clock skews and generating accurate programmable delays on-chip is developed.

The clock skew measurement system is built around a subsampling system where the clock nodes of interest are subsampled with a near-frequency asynchronous sampling clock to result in beat signals which are themselves skewed in the same proportion but on a larger time scale. The beat signals are then suitably masked to extract only the skews of the rising edges of the clock signals. A histogram of the arithmetic difference of the beat signals is proposed which decouples the relationship of clock jitter to the minimum measurable skew, and allows skews arbitrarily close to zero to be measured with a precision limited largely by measurement time, unlike the conventional XOR based histogram approach. The skew measurement system is validated by building a prototype around a chip, fabricated in TI 65nm CMOS process. The measured results indicate that for an input skew range of 1 fan-out-of-4 (FO4) delay, $3\sigma$ resolution of 0.84 ps can be obtained with an integral error of 0.65 ps.

The delay measurement unit (DMU), capable of measuring delays accurately for the full period range is used as the feedback element to build accurate fractional period delays based on input digital code word. The proposed delay generation system periodically measures and corrects the error to maintain it at the minimum and
does not require any special calibration phase. Up to 40X improvement in accuracy are demonstrated for a commercial programmable delay generator chip. The time-precision trade off the DMU is utilized to reduce the locking time. Loop dynamics are adjusted to stabilize the delay after the minimum error is achieved, thus avoiding chance of additional jitter. Measurement results from a high-end oscilloscope also validate the effectiveness of the proposed system in improving accuracy.

For medium frequencies (few hundreds of MHz) maintaining fine resolution for the entire period is with only fine delay steps is area consuming. Hence a coarse-fine architecture based delay generation system is proposed for on-chip realization of the fractional period delay generating system (FPDGS). The coarse delay units help in reducing area to generate large delays whereas the fine delay units help in achieving high resolution. Some of the conventional coarse and fine delay cell architectures are discussed in this context and improved version of those cells are proposed to enable low power operation along with good linearity.

The delay generated through the coarse-fine architecture drifts in presence of temperature variation and hence a calibration unit and controller is proposed to ensure accuracy in the generated delay. The architecture of controller to handle the discontinuities between the coarse and fine delay settings is also discussed. Measured results from an FPGA implementation as a proof of concept are also discussed, where the coarse-fine architecture was emulated using the FPGA LUTs (for generating coarse delay steps) and an external delay generation chip (for fine delay step generation).
Publications from the thesis work and contributing works

Patent

• Bharadwaj Amrutur and Pratap Kumar Das, “Methods and Systems for Measuring and Reducing Clock Skew” *US patent* application number 20110025391.

Journals


Conference Proceedings


Contents

Acknowledgements i

Abstract iv

Publications from the thesis work and contributing works vi

1 Introduction 1
   1.1 Definition of some commonly used terms . . . . . . . . . . . . . . . . 2
   1.2 Review of techniques to measure delay . . . . . . . . . . . . . . . . 4
   1.3 Scope of the thesis . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
   1.4 Organization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2 Subsampling Technique for Clock Skew Measurement 11
   2.1 Introduction to Subsampling . . . . . . . . . . . . . . . . . . . . . . . 11
       2.1.1 Time Domain view of Subsampling . . . . . . . . . . . . . . . 12
       2.1.2 Frequency Domain view of Subsampling . . . . . . . . . . . 13
   2.2 Subsampling for delay measurement . . . . . . . . . . . . . . . . . . . 13
       2.2.1 System Overview . . . . . . . . . . . . . . . . . . . . . . . . . 13
       2.2.2 Mismatches in the samplers and skews in the reference clock
distribution network . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
       2.2.3 Requirements in the sampling clock . . . . . . . . . . . . . . 19
       2.2.4 Possible ways of generating sampling clock . . . . . . . . . 20
       2.2.5 Impact of relative jitter and metastability . . . . . . . . . 21
List of Figures

1.1 Illustration of a typical clock distribution network with various components of clock skew. ........................................... 3
1.2 Definition of clock skew and jitter in clock signal. [1] ............... 4
1.3 Illustration of a typical clock distribution network with various components of clock skew. ........................................... 5
1.4 Illustration of a typical clock distribution network with various components of clock skew. ........................................... 6

2.1 Illustration of a subsampling system. ............................................ 12
2.2 Illustration of signal period amplification in sub-sampling system. . 13
2.3 Illustration of frequency domain perspective of subsampling action. . 14
2.4 Frequency domain spectrum of input, sampling and subsampled output. 14
2.5 Illustration of skew amplification between two inputs using subsampling. 15
2.6 Illustration of signal period amplification in sub-sampling system. . 16
2.7 Illustration of skew measurement in a distributed clock network. . . 18
2.8 Illustration of different source of errors in the measurement system. . 20
2.9 Glitches caused by sampling at the non-digital values of core clock and noise affecting the trip point. ................................. 22
2.10 Effect of jitter on core and sampling clock on the sub-sampled output. 23
2.11 Frequency domain spectrum of input, sampling and subsampled output in presence of jitter (phase noise). ............................. 24
3.13 Illustration of the capability of the DMU scheme-II in measuring delay of $T_2$ with jitter. .................................................. 39

3.14 Illustration of missing negative skews while starting the DMU counts with the positive transition of a particular input (around zero skew with jitter). .................................................. 39

4.1 Actual Measurement set-up in the lab .................................................. 42

4.2 Schematic of the measurement set-up in the lab .................................................. 43

4.3 Chip layout of the delay measurement front-end taped out in 65nm industrial CMOS process .................................................. 44

4.4 Measurement Illustration of precise delay generator around zero skew .................................................. 45

4.5 Input-output delay characteristic for an input delay range of $\pm 1$FO4 delay and the residual error after linear fit. .................................................. 46

4.6 Measured input-output delay for an input delay range of $\pm 600$ps. .................................................. 47

4.7 Measured standard deviation as a function of number of samples for asynchronous case. .................................................. 48

4.8 Measured standard deviation as a function of number of samples for rationally related sampling clock case with FM, for different modulation indices. .................................................. 49

4.9 Measured delay of internal delay element across different $\Delta T$'s for $T = 8.33$ns with and without applying frequency modulation. .................................................. 50

5.1 Characterization of sampler mismatch. .................................................. 54

5.2 Characterization of dependence of delay of an internal buffer on supply voltage. .................................................. 56

5.3 Illustration of the idea of routing clock signals from leaf nodes to central measurement unit for skew measurement. .................................................. 57

5.4 Illustration of the indirect method of skew measurement using the skew estimates from close-by nets. .................................................. 58
5.5 Illustration of the method of skew measurement using a zero skew reference signal for clock skew estimation at each test node using a TDC. 58

6.1 Functional view of a Programmable Delay Generator. 62

6.2 Illustration of a conventional interpolation based structure in a DLL to generate programmable delays. 64

6.3 Illustration of uncertainty trend in the generated stage delay for an interpolation based based architecture in a DLL. $\sigma$ value for each nominally 100 ps delay buffer is 4 ps and 100 such stages are cascaded. 66

6.4 Block diagram of the proposed system to generate any desired delay from 0 to $T$ accurately. 68

6.5 Architecture of the delay control unit. It takes desired ratio, threshold count, estimate done, $N_T$ and $N_n$ as input and provides the control signals to the controllable delay line as well as sets the fast/slow mode of operation for DMU. 69

6.6 Illustration of a case to explain controller details. (a) illustrates the concept of cyclic compliment of error. (b) shows a snapshot of the loop dynamics as the error crosses zero. 70

6.7 Time Vs simulated error for a starting error of 5 ns and delay generation step size of 10 ps with $\sigma$ of local step size mismatch of 0.3 ps. 72

6.8 Measurement setup used to validate the proposed architecture. 73

6.9 Schematic of the measurement setup used to validate the proposed architecture. 74

6.10 Characterization of INL and DNL errors of the system with and without enabling closed loop. 77

6.11 Measured error (normalized) of the system with and without enabling the closed loop using a 20 GS/s Agilent Infiniium DSO 90404A. 78

6.12 Measured jitter at the output of the delay generating system using a 20 GS/s DSO. 78
7.1 Architecture of coarse-fine delay cell unit.

7.2 Architecture of all-digital fractional period delay generating system.

7.3 Structure of a conventional delay line.

7.4 Structure of inverted and MUX based Lattice Delay Unit.

7.5 Structure of NAND gate based Lattice Delay Unit.

7.6 Structure of NAND gate to reduce the difference of the delays between the even to odd and odd to even setting changes.

7.7 Structure of the proposed Lattice Delay unit.

7.8 Linearity comparison of the various types of coarse delay units across various digital code settings. The

7.9 Illustration for explaining the difference in delays while changing from a even to odd setting or vice-versa.

7.10 Delay of a single coarse delay stage for various pass transistor widths.

7.11 Illustration of glitch generated at the output due to change of setting in the CDU at unsuitable instant.

7.12 Illustration of glitch generated at the output due to change of setting in the CDU at unsuitable instant.

7.13 Synchronizers for ensuring glitch-free coarse delay switching.

7.14 Layout of the coarse delay cell with embedded synchronizer.

7.15 Parasitic extracted delay and incremental power results for the coarse delay unit.

7.16 Monte Carlo simulation result of the parasitic extracted delay of the coarse delay unit.

7.17 Structure of Clock Vernier Devices used in Itanium Processor.

7.18 Linearity analysis of the Clock Vernier Devices used in Itanium Processor.

7.19 Structure of a single capacitor based delay line with constant switch width.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.20</td>
<td>Linearity analysis of the capacitor based delay lines with constant</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>switch width</td>
<td></td>
</tr>
<tr>
<td>7.21</td>
<td>Equivalent Circuit of RC based delay lines with product of series R</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>and C kept constant</td>
<td></td>
</tr>
<tr>
<td>7.22</td>
<td>Structure of Clock Vernier Devices used in Itanium Processor</td>
<td>98</td>
</tr>
<tr>
<td>7.23</td>
<td>Linearity analysis of the Clock Vernier Devices used in Itanium Processor</td>
<td>99</td>
</tr>
<tr>
<td>7.24</td>
<td>Layout of a single fine delay cell</td>
<td>100</td>
</tr>
<tr>
<td>7.25</td>
<td>Layout of the coarse-fine delay generation unit consisting of 10 coarse</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>delay cell</td>
<td></td>
</tr>
<tr>
<td>7.26</td>
<td>Per stage delay of the coarse delay unit and the dynamic range of the</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>5-stage fine delay unit across various temperatures</td>
<td></td>
</tr>
<tr>
<td>7.27</td>
<td>Minimum delay from the FDU with setting “0”</td>
<td>103</td>
</tr>
<tr>
<td>7.28</td>
<td>Illustration of the delay steps from the coarse and fine delay cells</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>with redundant fine delay cells to ensure coverage of coarse delay cells</td>
<td></td>
</tr>
<tr>
<td></td>
<td>through the fine delay unit</td>
<td></td>
</tr>
<tr>
<td>7.29</td>
<td>Calibration unit to find the coarse-fine ratio to minimize the error</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>during the change of coarse setting</td>
<td></td>
</tr>
<tr>
<td>7.30</td>
<td>State machine showing activating signal for generating coarse and fine</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>delay steps</td>
<td></td>
</tr>
<tr>
<td>7.31</td>
<td>State machine showing activating signal for generating coarse and fine</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>delay steps with limited number of fine delay cells</td>
<td></td>
</tr>
<tr>
<td>7.32</td>
<td>Simulated error Vs time plot showing the controller action in coarse-</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>fine architecture</td>
<td></td>
</tr>
<tr>
<td>7.33</td>
<td>Simulation result showing the action of the control loop by changing</td>
<td>111</td>
</tr>
<tr>
<td></td>
<td>coarse and fine delay settings to minimize the error with slow increase</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or decrease in temperature</td>
<td></td>
</tr>
</tbody>
</table>
7.34 Setup used for evaluating coarse-fine architecture. The blocks inside the shaded region are synthesized in a Virtex II Pro FPGA. The Fine delay unit is implemented using a commercial delay generation chip. 113

7.35 Linearity plot of the coarse delay chain implemented using the LUTs inside FPGA. 114

7.36 Look up table (LUT) and slice higherarchy inside FPGA. 114

7.37 Error in the closed loop case with only the coarse delay unit activated. 115

7.38 Measured result from the coarse-fine architecture implemented with FPGA LUT delays and the commercial delay generation chip having nominal step size 10 ps. 117
Chapter 1

Introduction

Precise on-chip delay measurement has been a challenge since the evolution of digital integrated circuits. With technology scaling in accordance with Moore’s law, transistors have become smaller and faster. However, a repercussion of scaling is that the variability of transistors has significantly increased. To leverage the higher speed, designs today employ faster clocks warranting tighter timing budgets. Variability, which translates to increased clock skew, is eating into these already tightened timing margins. Elaborate design techniques [2],[3],[4],[5],[6],[7], involving complex clock routing algorithms have been proposed to minimize this skew. These techniques, requiring rigorous testing and thus prolonged design cycles, have however proved ineffective in the presence of variability.

Active techniques have thus been proposed which adaptively compensate for on-chip clock skew during operation. This not only helps in improving the performance of the system, but also helps in meeting the time-to-market requirements, as the specifications in the design phase are now relaxed. These active de-skewing techniques require accurate measurement and generation of delays in the order of picoseconds. Further, using an all-digital approach, to generate and measure these delays, helps to reduce design time and enables easy portability to future technology nodes.

Accurate delay measurement has also been recently applied for characterizing
the performance of a technology node [8], [9], [10], [11], [12], [13]. These indirect measurements, using simple circuit structures, are simpler and more convenient than measuring intrinsic transistor parameters such as drain-current and threshold-voltage. The delay measurement problem of these circuit structures can be converted to a skew measurement problem by exciting them with a periodic source. But, the measurement technique needs to be accurate for a wide range of delays for measuring the delay of any arbitrary delay structure.

Another area of application for delay measurement and generation is in time-mode-signal processing where the controlling parameter is time rather than voltage [14], [15], [16]. This is one among many digitally assisted analog techniques recently proposed to replace analog components. The design of pure analog components becomes challenging in presence of process variation. Digitally assisted techniques not only help in making the design scalable for the future technology nodes, but also reduce the design effort. In this thesis we explore techniques to precisely measure and generate delays using an all-digital system which caters to the requirements of high precision and wide input range. The next section describes some commonly used terms in this thesis.

1.1 Definition of some commonly used terms

Fig. 1.1 illustrates a typical clock distribution network with various components of clock skew. Clock signal from the source node is distributed to the final clock storage elements through a buffer structure called clock distribution network. There are various buffers inserted in the distribution path to ensure good signal integrity. The inputs of the clock storage elements where the clock signal is finally delivered are called leaf nodes. Fig. 1.2 shows the typical timing diagrams and timing histograms of the clock signal of two different leaf nodes CK1 and CK2 [1]. The peak to peak range of the spread in the timing histogram plot around a single edge is characterized
Chapter 1. Introduction

Figure 1.1: Illustration of a typical clock distribution network with various components of clock skew.

as clock jitter which is dynamic in nature. On the other hand, the difference between
the two mean values of the timing diagrams of CK1 and CK2 around points A₂ and
B₂ is denoted as clock skew which is mostly static in nature. The relative arrival time
difference between clock signals at the leaf nodes of this distribution network, the
clock skew, directly eats into the timing margins [17]. Hence elaborate care is taken
to design these distribution networks, to minimize this skew [2],[3],[4],[5],[6],[7].

Fig. 1.2 shows the typical timing diagrams and timing histograms of the clock
signal of two different leaf nodes CK1 and CK2 [1]. The peak to peak range of the
spread in the timing histogram plot around a single edge is characterized as clock
jitter which is dynamic in nature. On the other hand, the difference between the two
mean values of the timing diagrams of CK1 and CK2 around points A₂ and B₂ is
denoted as clock skew which is mostly static in nature.
1.2 Review of techniques to measure delay

Delay measurement can be done using either analog or digital techniques. Analog phase locked loops (PLLs) and delay locked loops (DLLs) generally use a phase detector followed by a charge pump to generate analog voltages proportional to input delay. As shown in Fig. 1.3 the voltage across the capacitor is precharged to $\frac{V_{dd}}{2}$ during the reset phase in each clock cycle. The delay difference between the inputs controls the turn on times of the up and down pulses causing the capacitor node voltage to change proportional to the input delay. This voltage change can be sampled by an ADC and then digitized to measure delay in each clock cycle as shown in Fig. 1.3. The ADC output samples are averaged out to find the skew in the measurement cycle. This method can measure the jitter in high speed clocks, but the accuracy is very dependent on the matching between the pull up and pull down current sources and also sensitive to the power supply noise coupled to the precharged
capacitor node. It becomes quite challenging to maintain linearity while measuring large delays using such a scheme with high accuracy. This scheme requires an ADC to read out a digital value proportional to the input delay in the measurement interval. Digitizing the capacitor voltage at each sampling instant and then resetting helps in relaxing ADC dynamic range requirement as the averaging is pushed to the digital domain. The ADC consumes a large area and hence methods that do not requiring an ADC would be desirable.

![Diagram](image)

Figure 1.3: Illustration of a typical clock distribution network with various components of clock skew.

The other kind of architecture used to digitize delays is shown in Fig. 1.4 where the relative delay between the reference signal and the input signal is measured by delaying the reference signal by a fixed step size of delay through a chain of inverters or buffers [19], [20]. The average delay of the buffer of inverter is estimated in calibration mode. An external measurement equipment is used for calibration in the technique used in [20], whereas the average inverter delays are estimated in [19] by making the structure oscillate and by observing the oscillation frequency. The authors in [21] survey some popular digital techniques for delay measurements which use tapped and vernier delay line methods. The authors in [22] propose a flash time-to-digital converter which utilizes arbiters and can be calibrated for a very high resolution.
Instead of using fixed delay steps shown in Fig. 1.4 they use the spread of arbiter threshold voltages for getting a set of digital codes from an array of arbiters to measure the delay. But it is limited by the non-uniform distribution of the offsets of the arbiters, and also works only for a small input delay range. The resolution of the delay line based scheme is limited by the delay of a single buffer or inverter.

![Illustration of a typical clock distribution network with various components of clock skew.](image)

A vernier delay line based technique enables measurement of delays lesser than that of a single buffer or inverter. The technique uses two buffer chains with elements in the first buffer chain having a delay $T_1$, and another buffer chain having a delay $T_2$ ($T_1 < T_2$). The input and the reference signals are launched as inputs to the first and the second buffer chains respectively, with the rising edge of the reference signal leading that of the input signal. As the two signals propagate through the chains, the relative delay between them reduces by $T_2 - T_1$ with each successive stage. The input delay is measured by noting the crossover point and the value of $T_2 - T_1$ (measured in calibration phase). The resolution of the technique is limited by the mismatch between the individual buffer delays. The supported range of input delays is proportional to the number of stages in the two buffer chains. Hence, for supporting
a wider input delay range a larger area is necessary.

The issue with larger area requirement and the mismatch between the individual buffer stages in the delay chains are addressed in component-invariant vernier delay line \[23\]. The technique uses only two recirculating inverter stages in place of the two buffer chains, hence reducing the area and addressing the mismatch issue. Authors in \[24\] propose a very precise coarse-fine time-to-digital converter based on a time amplifier. The coarse-fine architecture enables it to support higher input range. The peripheral circuits for calibrating out the nonidealities take relatively larger area. Most of these TDCs have the capability to measure delays between edges of two aperiodic signals. They can also make a single shot measurement - that is, a single occurrence of an edge pair can be used to determine their time separation. The authors in \[25\] propose a scheme to characterize the period jitter by obtaining the cumulative distribution function of the clock edges, and modify it to measure the skew between two leaf nodes of a clock distribution network. There are a few sampling based techniques which use a random sampling signal to sample the input signals and the statistical information of the output sampled signals are processed to get the original skew information. The techniques work for periodic signals, but around zero skew with jitter, these techniques incorrectly estimate a non-zero skew proportional to the relative jitter between the input signals. Also these techniques consume more power than the proposed subsampling technique for delay measurement as discussed in Chapter 3. The details of those techniques will be discussed in chapter 3.

### 1.3 Scope of the thesis

The asynchronous sub-sampling technique is utilized to achieve high accuracy and precision while measuring static skews between two periodic signals in an area efficient manner. The problems in the histogram counter based approaches are discussed along with de-bouncing techniques to handle issues such as metastability and jitter.
Debouncing followed by averaging of the arithmetic difference of the signals (histogram of arithmetic difference) removes any dependency of resolution on sampling clock jitter, unlike in previous works. Measured results from a 65nm test chip indicate the ability to measure skews with a $\pm 3\sigma$ resolution of 0.84ps and integral error of 0.65ps for an input skew range of 1 FO4 delay($\pm 20$ps). The technique can also be used to measure larger skews even close to $\pm \frac{T}{2}$, where $T$ is the clock period. The precision is unaffected by clock jitter as a measurement resolution of 0.84ps has been achieved with clock sources with 30ps rms jitter. This is further validated by experiments where frequency modulation on sampling clock preserves the resolution. In fact, in certain cases where the sampling clock is rationally related to core clock, frequency modulation improves resolution. The proposed technique is thoroughly validated with a prototype made around a chip taped out in 65 nm CMOS process. A few applications of the proposed technique are also illustrated.

The delay measurement technique is utilized to measure the on-chip variability such as mismatches in the rise or fall delays, mismatch between samplers etc.

Another potential application of the technique in the context of clock skew measurement in a distributed environment is also proposed. The adoption of such a scheme can reduce the hardware requirement while measuring skew across multiple leaf nodes.

Another system level application is also shown in generating accurate delays in a programmable delay generator. The delay measurement unit (DMU), capable of measuring delays accurately for the full period range is used as the feedback element to build accurate fractional period delays based on input digital code word. The proposed closed loop control for a delay generation system allows much better accuracy than the open loop case to generate arbitrary fractional unit interval delays. The elements constituting the system need to be carefully designed to enable stability and precision of delay generated across full range.

The proposed system periodically measures and corrects the error to keep it at
the minimum and does not require any special calibration mode for error correction. Therefore, the system can run without interruption for a long time with minimum error even if the slow varying parameters like temperature vary with time. Up to 40X improvement in accuracy is measured by enabling the feedback control, with respect to the open loop case.

A coarse-fine architecture to implement an all digital programmable delay generator on-chip and the related issues are also discussed.

1.4 Organization

This thesis is split into two parts. The first part, Chapter 2, 3, 4 discuss the measurement aspects of precise on-chip clock skew. Chapter 5 discusses about some of potential applications of the proposed technique in measuring precise on-chip skews. Chapter 6 and 7 focus on one of the applications of the proposed clock skew measurement technique to generate accurate programmable delays.

Chapter 2 introduces the method of subsampling and discusses various practical aspects in implementing a skew measurement system using subsampling. Chapter 3 discusses about various methods to estimate clock skew from the subsampled signals. The test setup to validate the skew measurement technique and the results from a prototype involving a 65 nm skew measurement front end chip are discussed in chapter 4. Chapter 5 discusses some of the potential applications of the proposed technique in measuring on-chip uncertainty like sampler mismatch or on-chip standard cell delay with measured results. It also discusses about a potential application of the proposed technique in measuring on-chip clock skew between multiple leaf nodes in a clock distribution network with very less hardware resource per test node. Chapter 6 discusses about the limitation of the programmable conventional delay generation systems and shows the application of the delay measurement system to generate accurate delays in an uninterrupted manner backed by some measurement results. Chapter 7 discusses
about the on-chip implementation aspects of the proposed fractional period delay generation system. Chapter 8 briefly reviews the general scope and limitations of the proposed techniques along with some possible future extensions of this research work.
Chapter 2

Subsampling Technique for Clock Skew Measurement

The concept of subsampling is introduced in this chapter. The practical aspects and the limitations of the delay measurement system using subsampling concept are also discussed following the introduction.

2.1 Introduction to Subsampling

The phenomenon of subsampling is understood since a long time. Some of the alternate names of the phenomenon are: stroboscopic effect, aliasing effect etc. It occurs when a signal is sampled by another signal of close by frequency. One of the common examples of subsampling effect in our day to day life is the “wagon-wheel effect” where the spokes of the wheel seem to be stationary or even moving slowly in opposite to their original direction of motion depending on the relative difference between the strobe light firing rate and the speed of revolution of the wagon wheel. There are also many application of the subsampling effect in playing back from DVD disks, laser microphones, e-beam lithography etc. The concept of subsampling is used to monitor internal signals on-chip [26], [27]. In this thesis, the concept of subsampling is
explored for measuring delays between two periodic signals with good precision. The concept of subsampling can be explained both in time domain as well as frequency domain and discussed in next.

2.1.1 Time Domain view of Subsampling

A sampler (D flip-flop) having a periodic input of period \( T \) (called “input clock”) at the D input and another periodic signal of period \( T + \Delta T \) (called “sampling clock”) at the clock input as shown in Fig. 2.1 exhibits the phenomenon of subsampling. As the rising edge of the sampling clock samples the input clock and holds for one sampling clock period, effectively the sampling clock advances by “\( \Delta T \)” in each sampling clock cycle. Therefore, in \( \frac{T}{\Delta T} \) sampling clock cycles, the subsampled output completes one full cycle. Since, the output of the D-FF (sampled clock) has a period of \( \frac{T(T + \Delta T)}{\Delta T} \) corresponding to input signal period (T), there is an amplification factor of \( \frac{T + \Delta T}{\Delta T} \). Choosing lower values of \( \Delta T \) can lead to good amount of amplification in time axis, which relaxes the measurement infrastructure required while dealing with the subsampled outputs as compared to the original high frequency inputs.

Figure 2.1: Illustration of a subsampling system.
2.1.2 Frequency Domain view of Subsampling

From a frequency-domain perspective, the sampler action is illustrated in Fig. 2.3. The mixer produces frequency components corresponding to the addition and subtractions of the input clock frequency and the sampling clock frequency. Since both the input and sampling clocks are periodic with frequencies $f$ corresponding to $\frac{1}{T}$ and $(f - \Delta f)$ corresponding to $(\frac{1}{T+\Delta T})$, their spectrum will be discrete with components existing at the frequencies of multiples of their fundamental frequency. The spectrum of the subsampled output is shown in Fig. 2.4. It can be seen that the subsampled output consists of components at multiples of beat frequency ($\Delta f$) and hence effectively, the signal is transferred into a lower frequency domain.

2.2 Subsampling for delay measurement

2.2.1 System Overview

Consider an arbitrary buffer and interconnect network, in which we need to measure the skew (or delay) between two nodes $d_1$, $d_2$, as shown in Fig. 2.5. For example, this could be a clock distribution network and the nodes of interest might be leaf nodes.
Chapter 2. Subsampling Technique for Clock Skew Measurement

Figure 2.3: Illustration of frequency domain perspective of subsampling action.

Figure 2.4: Frequency domain spectrum of input, sampling and subsampled output.

of such a network. By exciting the input of this network with a periodic clock source (\textit{clk}) of period \( T \), we can expect periodic outputs at the two leaf nodes, which have a relative skew of \( \delta \). We introduce two samplers at each of these nodes, which are clocked by a separate sampling clock, \( \textit{sclk} \) with a slightly different period \( T_s = T + \Delta T \).\footnote{Note that \( T-\Delta T \) as well as any \( nT\pm\Delta T \) will also work for an integer \( n \). However for \( n>1 \), the measurement time is increased by the factor of \( n \) to achieve the same accuracy.}

We can use a sampling clock which is either asynchronous to \( \textit{clk} \) (obtained by using
Figure 2.5: Illustration of skew amplification between two inputs using subsampling.

an independent crystal source) or is rationally related (via a DLL/PLL), but has additional jitter added (through FM). The output of the two samplers will be beat signals as shown in Fig. 2.6, whose period is given as \( T_s \times \frac{T}{\Delta T} \), which is essentially the sampling clock period amplified by a factor \( \frac{T}{\Delta T} \). The input skew (\( \delta \)) is also amplified as a skew between the sub-sampled outputs to be \( \frac{\delta}{\Delta T} \times T_s \). The skew in terms of unit interval, i.e. the fraction \( \frac{\delta}{\Delta T} \) is then digitally measured by the proposed delay measurement unit.

Mathematically this phenomenon can be represented through the following equations.

\[
\text{Skew between subsampled signals } (T_{sk}) = \frac{T + \Delta T}{\Delta T} \times \delta \quad (2.1)
\]

\[
\text{Period of each subsampled signal } (T_{beat}) = \frac{T + \Delta T}{\Delta T} \times T \quad (2.2)
\]

\[
\text{Hence, } \frac{T_{sk}}{T_{beat}} = \frac{\delta}{T} \quad (2.3)
\]
The delay amplification can also be seen in case of two phase separated sinusoidal signals sampled by another sinusoidal signal. Let \( d_1 \) and \( d_2 \) be two sinusoidal signals with same frequency \( f \) and having a phase difference of \( \delta \). They can be represented as:

\[
d_1(t) = A \cos(2\pi ft + \phi) \\
d_2(t) = A \cos(2\pi f(t - \delta) + \phi)
\]  

Let the sampling signal be represented as:

\[
S(t) = 2A \cos(2\pi (f - \Delta f)t + \theta)
\]
Chapter 2. Subsampling Technique for Clock Skew Measurement

Using the block diagram shown in Fig. 2.3, the subsampled signals from $d_1$ and $d_2$ can be found out to be:

$$q_1(t) = A \cos(2\pi(\Delta f)t + \phi - \theta) + \text{higher frequency terms} \quad (2.7)$$
$$q_2(t) = A \cos(2\pi(\Delta f)(t - \frac{f * \delta}{\Delta f}) + \phi - \theta) + \text{higher frequency terms} \quad (2.8)$$

It can be seen that the phase difference between $q_1$ and $q_2$ are now $\frac{f * \delta}{\Delta f}$ which is $\frac{f}{\Delta f}$ times the original phase difference between the two inputs.

This principle can be used to make distributed clock skew measurement between more than two leaf nodes in a clock distribution network as shown in Fig. 2.7. Here the triangle on the left side represents the clock distribution network of the main core clock to all its sequencing elements. In addition for a subset of the leaf nodes of this network, additional samplers are inserted at these locations, determined a priori during design time. The core clock is fed to the data input of these samplers and they are clocked by a separate sampling clock. The sampling clock is of a period which is slightly different from the core clock. The outputs of these samplers are of the beat period and are routed to a central measurement location for further analysis. Note that in this scheme, the core clock at the leaf nodes are minimally perturbed, thus preserving their original skews. The delay measurement unit can be shared across all the sampled nodes. By using a multiplexer to select two sub-sampled signals, skews between pairs of nodes can be obtained. The pair-wise skew information can then be stitched together to give the overall skew distribution across all the measured nodes.

Since the sub-sampled outputs are in the domain of the sampling clock, which is the same as that for the delay measurement unit, routing of these signals is simplified greatly. The only constraint is the need for the same number of pipeline delays for each of the sub-sampled signals. This also allows for easy measurement of skews in a very high speed clock networks. Because the delay measurement unit can be shared, the area overhead for this approach is very small.
Figure 2.7: Illustration of skew measurement in a distributed clock network.

There are four significant challenges which can limit the efficacy of this approach and these are:


[B] Requirements in the sampling clock.

[C] Impact of relative jitter and metastability.


2.2.2 Mismatches in the samplers and skews in the reference clock distribution network

Mismatches in the setup/hold time of the samplers will limit the measurement accuracy. The input voltage offset (\(\Delta V_{\text{sample}}\)) of the two samplers in conjunction with finite slew (\(S_r\)) of the input signals also adds to the skew in the beat signals. Careful sizing and layout of the samplers needs to be done to reduce mismatches which arise
due to both systematic and random process and environmental effects. Since the impact of random local variation reduces with size as $\frac{1}{\sqrt{\text{width} \times \text{length}}}$ [28], by using larger sized transistors and ensuring good rise times in the sampling clock network, good matching in the setup/hold times of the samplers can be achieved. For example in an industrial 65nm process, we did a Monte Carlo analysis of the setup/hold times of a standard cell Master Slave Flip-flop. The maximum observed mismatch in the setup times was 4ps with rise times of 1 FO4 delay for the clocks. We could reduce this mismatch to less than 1ps by increasing the transistor sizes in the master stage by up to 3X its original size. Note that any skew in the sampling clock to the two samplers ($\Delta T_s$) will also add to the skew in the beat signals as shown in Fig. 2.8. But this is inevitable for any skew measurement in a distributed network which requires a reference clock. The authors in [29] have shown that by using the fine mesh distribution, skews of less than 1ps can be achieved in clock distribution network for large digital chips. In our particular measurement problem, the number of measurement points will be very small compared to the total number of flops in large digital chips of the kinds in [29]. Hence it seems very feasible to achieve a very low skew (of less than 1ps) reference clock distribution network. Hence what the delay measurement unit actually measures is $\delta + \Delta T_s + \frac{\Delta V_{\text{Sampler}}}{T_{\text{Sampler}}}$. Thus, the accuracy of measurement is limited by the quality of sampling clock distribution and input sampler mismatch. However, we will show later that the precision of the measurements will be largely determined by the measurement time and will be discussed in future chapters.

2.2.3 Requirements in the sampling clock

The sampling clock, $clk$ can be obtained from either an asynchronous source (obtained by using an independent crystal source) or is rationally related (via a DLL/PLL). However, with rationally related clock sources, the resolution is very sensitive to the relationship between $T$ and $\Delta T$. The resolution can be improved by artificially adding additional jitter (e.g. through Frequency Modulation).
Figure 2.8: Illustration of different source of errors in the measurement system.

From equation 2.1, we can see that getting the skew number, \( \delta \) in ps requires knowledge of \( T \) and \( \Delta T \). However applications which need the skew only in units of UI (unit interval of one period \( \frac{\delta}{f} \)), knowledge of \( \Delta T \) is not required as shown in equation 2.3. Our laboratory measurement results confirm these observations. From equation 2.1, we can see that an error in the knowledge of \( T \) and \( \Delta T \) directly contributes to an error in the measured absolute skew. The error in knowledge of the periods of the clock sources will lead to an uncertainty in the exact value of \( \Delta T \). Thus the lower value of \( \Delta T \) is bounded by this uncertainty.

### 2.2.4 Possible ways of generating sampling clock

The sampling clock having the qualities mentioned in the above subsection poses few design challenges and hence are discussed in this subsection. In the case where the sampling clock is provided from a separate clock source using a separate a crystal, as long as the frequencies are relatively close and the difference is maintained stable, the technique for clock skew measurement can work using subsampling. But when the sampling clock is generated from the input clock on-chip, it needs to be done with careful design considerations. It can’t be through mixing a low frequency with the clock that will leave a spur of the clock which will cause problems in the subsampling algorithm. Another practical phenomenon is the pulling effects [30] where two clocks
in the same chip tend to pull each other unless one is careful. This puts the lower limit to the value of $\Delta T$ even for clock sources from separate crystals as the pulling effect can cause inaccuracy in the measurement. But for suitable range of $\Delta T$s the sampling clock can be generated from the input clock using a divider to divide the input frequency by a factor of $N + 1$ and then using a PLL to multiply the divided signal to get the signal with period $T \cdot \frac{N+1}{N}$ which results in a $\Delta T$ value of $\frac{T}{N}$ and can be made to be small by choosing a large value of $N$ [31]. The resolution of the measurement can be further improved by introducing artificial jitter through FM modulation or any other source as that helps in getting resolution better than $\Delta T$ even in the case where $f$ and $\Delta f$ are rationally related.

### 2.2.5 Impact of relative jitter and metastability

**Time domain view**

When the sampling clock edge comes within the setup/hold window of the measured clock, the output of the samplers can go metastable. However the metastability can be resolved with almost certainty by using the standard technique of double sampling (i.e., the output of the first sampler is re-sampled by another sampler clocked by the reference clock). However the output of the samplers in the metastability region cannot be predicted and can be either logic high or low. So for the duration of the time the reference clock edge traverses the metastability window, the output of the samplers will be either high or low. This is further illustrated in Fig. 2.9, where a practical clock signal has a rise time of few tens of pico seconds, and the reference clock is chosen so that the $\Delta T$ is in pico seconds, leading to a few samples taken in non-digital values of the sampled clocks resulting in meta-stability and random output determined by noise. Hence we can expect multiple closely spaced bounces in the beat signals near their transitions. Phase jitter in the clocks can cause this unpredictability even for sampling points apparently farther away from the metastability window. As
shown in Fig. 2.10, in the absence of jitter, the period of the sampling clock is $T_{s0}$ and $T_{c0}$ respectively, but due to jitter in the clocks, $T_{c1} > T_{c0}$, $T_{c2} < T_{c0}$ which results in the first bounce. Similarly the reduction of the instantaneous value $T_{s4}$ as compared to $T_{s0}$ can give rise to another bounce. A very large variation of the sampling clock period ($T_{s9}$ and $T_{s10}$) and an opposite change in Core clock period ($T_{c9}$ and $T_{c10}$) can also lead to an isolated bounce at the output. Bounces in the beat signals can be avoided by taking a very high value of $\Delta T$, but this leads to a reduction of time amplification, resulting in poor resolution. Hence we need to be able to clean up the bounces and estimate the edge position of the bounced clocks.

![Figure 2.9: Glitches caused by sampling at the non-digital values of core clock and noise affecting the trip point.](image)

Various de-bouncing algorithms are possible. A simple de-bouncing technique is to use the first rising edge of the bounced signal as the rising edge of the de-bounced signal. However an isolated bounce due to jitter can lead to inaccuracies in the skew measurement (Fig. 2.10). To ensure that the estimated de-bounced rising edge is closer to closely spaced bounces rather and is less susceptible to isolated bounces, various de-bouncing and skew estimation algorithms are proposed in this work. They are discussed in the next chapter.
Chapter 2. Subsampling Technique for Clock Skew Measurement

Frequency domain view

From a frequency-domain perspective, the sampler action in presence of phase noise or jitter is illustrated in Fig. 2.11. In presence of phase noise (jitter) in the sampling and sampled clock signals, the spectrum around the fundamental and higher harmonics turns out to be a continuous spectrum peaked at the fundamental frequency and components around the higher order harmonics. The spread about the fundamental frequency is proportional to the amount of jitter. From the figure, it would be apparent that for very small values of $\Delta f$ the first two lobes of the spectrum come very close to each other. But that does not impact the accuracy of the measurement as long as the mean value of $\Delta f$ remains relatively constant during subsequent measurement intervals.

2.2.6 Measurement Time

The downside of time amplification achieved by sub-sampling is that even the measurement time is increased by a factor of $\frac{T}{\Delta T}$. In addition, averaging over many samples is needed to overcome the effects of random phase noise, in order to reduce the measurement errors. Hence this technique puts the following constraints on the
Chapter 2. Subsampling Technique for Clock Skew Measurement

Figure 2.11: Frequency domain spectrum of input, sampling and subsampled output in presence of jitter (phase noise).

clocks and the skews:

[i] The period of the core clock and sampling clock should be stable over the measurement period.

[ii] The skew should not change at a rate higher than half the bandwidth of the measurement.

But these would not be of serious concern in the case of typical clock skew measurements because the skew itself varies very slowly with time (mainly due to temperature variations).

2.3 Conclusion

In this chapter, we looked at the phenomenon of subsampling from both time and frequency domain. With subsampling, along with the output period, the delay between
two subsampled signals generated from two inputs signals of same frequency is also amplified. This delay amplification helps in reducing the complexity of measurement circuit. We also looked at some of practical challenges and requirements faced while using the subsampling principle for high precision clock skew measurement along with some possible solutions.

In the next chapter we will look at various ways for estimating the average skew between the two input signals from the subsampled signals.
Chapter 3

Delay Measurement Unit

Various options for estimating the average skew between the two input signals from the subsampled signals is discussed in this chapter. The delay measurement unit (DMU) makes a precise estimate of the original clock skew from the subsampled signals. The role of the DMU would be pretty straightforward in the absence of jitter and metastability. But the presence of jitter and metastability in the samplers cause bouncing in the subsampled outputs which make the estimation process tricky. In this chapter, various possible options for delay estimation and debouncing are discussed and the optimum solution catering to all issues is also provided. There are three different types of estimation methods those can be potentially used for the debouncing and estimation purpose. They are as follows:

[A] Conventional XOR based histogram counting approach.


[C] Time averaging based estimation.

[D] Estimate through difference of histogram.
3.1 Conventional XOR based histogram counting approach (Code density test)

The conventional XOR based histogram count operation is illustrated in Fig. 3.1 where the counter counts up when \( S_1 \) is at a different logic level than \( S_2 \). It has been used in various contexts in [32], [33], [34]. For clean subsampled signals with equal rise and fall delays this method estimates the correct estimate. But when the rise and fall delays are unequal and only the rise delays between the two input signals are of interest, this method doesn’t provide the correct estimate.

![Figure 3.1: Illustration of XOR based histogram counting method.](image_url)

3.2 Conventional random sampling based histogram counting approach

The conventional random sampling based histogram count operation is illustrated in Fig. 3.2 where the counter counts up when \( S_1 \) is at logic high and \( S_2 \) is at logic low. It has been used in various contexts in [35], [36], [37], [38]. For clean subsampled signals or when the delays to be measured are very high this method works without any problem. But due to presence of jitter on the clocks, one can expect errors for estimated skews of the order of the jitter. This can be easily understood for the case of exactly zero nominal skew between \( d_1 \) and \( d_2 \) (Fig. 3.3). In this case, due to uncorrelated jitter in \( d_1 \) and \( d_2 \), the histogram count will increment to a non-zero value, which is related to the uncorrelated magnitude of jitter \( \sigma_{\text{jitter}} \). This effect
is termed as reordering issue in [32]. The authors in [32] get around this by using a reference signal which has a much larger delay than $\sigma_{\text{jitter}}$ and measure the delays of the individual nodes against this. The values are then subtracted to give the delay between the original nodes. The proposed arithmetic difference in effect achieves this, but without needing an extra reference signal, thus reducing area and power. Fig. 3.4 shows the problem in the conventional histogram counting based estimation approach in presence of bouncing in the subsampled outputs. It can be seen that the two beat signals are delayed by a single sampling clock corresponding to an original delay of $\Delta T$. But, with the conventional histogram counting based approach, the estimated value turns out to be 3 which would wrongly correspond to an original delay of $3*\Delta T$.

![Count up when $S_1=1$ and $S_2=0$](image)

**Figure 3.2**: Illustration of XOR based histogram counting method.

![Average Skew=0](image)

**Figure 3.3**: Illustration of error due to jitter in input clocks for measurement of skews around zero in case of histogram counting method.

To solve the problems of the conventional histogram approach, time averaging
3.3 Time averaging based estimation

The time averaging based method effectively finds out the difference between the virtual rising edges of the two subsampled signals where the virtual rising edges correspond to the average of the rise times of the bouncing signals. The detail procedure is discussed below.

[a] De-bouncing Step: Detect the first rising edges of both the bouncing subsampled clocks in each beat and find the delay between them. This can be implemented by using the state machines shown in Fig. 3.6. In the figure the threshold value is used to ensure that there is a single rising and falling edge in each beat period. The timing diagrams explain the operation of the state machines.

This delay is given by $Y_0 - X_0$ in Fig. 3.5

[b] Estimate the contribution from the bouncing edges for both the bouncing clocks towards the total time amplified skew in each beat.

This is given by $-\left(\sum_{i=1}^{3} X_i - X_0\right)$ for sub-sampled clock1 and $\left(\sum_{j=1}^{2} Y_j - Y_0\right)$ for sub-sampled clock2 in Fig. 3.5

[c] Skew Estimation step: Estimate the contribution from the bouncing edges for
both the bouncing clocks towards the total time amplified skew in each beat. Referring to Fig. 3.5, this is given by

\[
\text{Delay Count Per Beat} = \sum_{i=0}^{2} \frac{Y_j}{3} - \sum_{i=0}^{3} \frac{X_i}{4}
\]

[d] Find the estimate of the delay \( (N_\delta) \) by averaging the above quantity for \( 2^n \) beat cycles.

[e] Estimate the no. of sampling clocks per beat period and average it for \( 2^n \) beat cycles to find \( N_T \).

[f] Find \( \frac{N_\delta}{N_T} \). This quantity is the skew per unit interval i.e. \( \frac{\delta}{T_{\text{input clock}}} \). Where \( T_{\text{input clock}} \) is the input clock time period and \( \delta \) is the actual skew corresponding to the time amplified sub-sampled signals.

The choice of \( \Delta T \) (the difference in time periods of the core clock and sampling clock) affects the accuracy of measurement in the form of time amplification, measurement time and “amount of bounce” of the beat signals.

Fig. 3.7 shows the block diagram of the time averaging based delay measurement unit. The overall system counts to around 4K NAND3 equivalent gates in our implementation.

### 3.4 Estimate through difference of histogram

To simplify the hardware required in case of a time average based DMU, a difference of histogram approach was proposed [40]. The approach is a modified version of the conventional XOR based histogram approach and in this case, the difference between the two signals is calculated in each beat period and an average of that quantity over many cycles is estimated to find the clock skew.
Chapter 3. Delay Measurement Unit

Fig. 3.5: Flow chart for time average based clock skew estimation.

Fig. 3.8 shows the DMU architecture for measuring the actual skew between the input and output of the system from the sub-sampled signals $q_1$ and $q_2$. In a practical digital subsampling system, due to jitter and finite rise-time of the signals and the meta-stability of the samplers, their outputs will have bounces between the digital values, as shown in Fig. 3.9. Since we are interested in finding the skews for only one polarity of edges of the inputs, we need to suitably mask out the sampled signals corresponding to the falling edges. Hence the subsampled signals with bouncing need to be processed with the aid of de-bounce and masking state machines to mask out the falling edge statistics to give $c_1$, $c_2$. Their difference, $c_1 - c_2$, is accumulated in a counter for $2^k$ beat cycles to obtain the digital code word for $\frac{\delta}{\Delta T}$ or $N_\delta$. Similarly, another accumulator accumulates sampling clock cycles in the measurement time to
Figure 3.6: State machines and timing diagram of various signal to be used for time average based clock skew estimation.

give an estimate of $\frac{T}{\Delta T}$ or $N_T$. $\delta T$ is then equal to $\frac{N_1}{N_T}$.

Due to jitter and finite rise-time of the signals and the meta-stability of the samplers, their outputs will have bounces between the digital values, as shown in Fig. 3.9. Since we are interested in finding the skews for only one polarity of edges of the inputs $d_1$ and $d_2$, we need to suitably mask out the sampled signals corresponding to the falling edges. This is done via two state machines, as shown in Fig. 3.9. The timing waveforms of the signals used in the state machines are also sketched. The de-bounce state machine detects the first rising edge on the sampler output, $q_i$ and asserts the $\text{en}_i$ signal to cover for the high duration of the beat signal $q_i$. The high duration of the beat signal is determined by a timer clocked by the sampling clock. After the timer crosses the threshold, the first time either of $q_1$ or $q_2$ fall, the mask signal, $m$, is de-asserted to de-assert $c_1$ and $c_2$ simultaneously. Now the signals $c_1$ and $c_2$ contain only the rising edge information for the input signals $d_1$ and $d_2$, and hence their histogram analysis gives the rising edge statistics.
Figure 3.7: Block diagram of the time average based delay measurement unit. Input to the block are the beat signals and the sampling clock. The outputs are the skew count($\frac{\delta}{\Delta T}$) and period count($\frac{T}{\Delta T}$).
To ensure that the enable signals $e_n$ do not get false triggered and to go to zero due to the bouncing edges of $q_i$ near the rising transitions, the threshold value should be set to an appropriate value. The upper limit on that is imposed by the $T/\Delta T$ ratio. In our implementation, it is set to 16 as for all the cases when $1000 > \frac{T}{\Delta T} > 50$ and was set to 2 when $\frac{T}{\Delta T} < 50$ as for lower $\frac{T}{\Delta T}$ ratios, getting more than two bouncing transitions around a rising edge of $q_i$ is highly unlikely for the practical values of jitter on the clocks.

Two methods of sampling clock generation are shown in Fig. 3.8. The sampling clock can be generated from the input clock by using an asynchronous crystal source or using a clock synthesizer consisting of a PLL (to multiply the input clock frequency, $f_{in}$ by P) and frequency divider (division factor Q) to generate a frequency little less than input clock frequency by keeping the value of P just one less than Q as discussed in [40],[31], so that the mean value of $\Delta T$ is positive. For getting better resolution with larger $\Delta T$s, additional jitter can be added to improve the resolution as discussed in [40]. However, if in case, a separate crystal is used to generate the sampling clock, since the subsampling technique works for a wide range of $\Delta T$s [40], the central frequency can be chosen such that the mean value of $\Delta T$ for a measurement unit is positive even in the worst case.

The DMU can also work for negative $\Delta T$ by doing a polarity inversion of $q_1$ and $q_2$ at the input of the DMU based on the sign of $\Delta T$. If $\Delta T$ is negative, then the clean falling edges of the beat signals correspond to the rising edges of the input signals and hence for estimating the skew between the rising edges, skews between the clean falling edges of the beat signals have to be used. The $\text{sign}(\Delta T)$ signal used to adjust the polarity of the beat signals can be generated by two simple counters to count with the input clock and the sampling clock and an arithmetic comparator. The signal $\text{sign}(\Delta T)$ can be updated at regular intervals by sampling at the logic level at the output of the arithmetic comparator.

Fig. 3.10 illustrates the problem in measuring correct delay for skews around the
Chapter 3. Delay Measurement Unit

Figure 3.8: Block diagram of the delay measurement unit based on histogram of arithmetic difference method. Inputs to the block are the beat signals and the sampling clock. The outputs are the skew count ($\delta/\Delta T$) and period count ($T/\Delta T$).

Figure 3.9: State machines and timing diagram of various signals to be used for histogram of arithmetic difference method based clock skew estimation (Scheme-I).
value of $\frac{T}{2}$ in presence of jitter. As shown in Fig. 3.10 the skew is calculated by counting $c_1 - c_2$ only within the duration when mask signal $m$ is high and hence, the detected skew can toggle between $0$, $\frac{N}{2}$ or $-\frac{N}{2}$ due to jitter making the total estimate erroneous. Hence, a modification to the above system was proposed to enable skew measurement across full range of a clock period. The problem in the above scheme lies in the mask state machine and hence in the improved version of the delay measurement unit [41] the system is modified as shown in Fig. 3.11 and Fig. 3.12.

Figure 3.10: Illustration of the problem in measuring delay of $\frac{T}{2}$ with jitter in DMU scheme-I discussed above.

The masking of falling edges is done via two state machines, as shown in Fig. 3.12. The timing waveforms of the signals used in the state machines are also sketched. The de-bounce state machine generates signals (en$_1$ and en$_2$) having a single rising and falling edge in a beat cycle from the input beat signals $q_1$ and $q_2$. en$_1$ and en$_2$ are used by the masking state machine to generate the signals $g_1$ and $g_2$ used for generating $c_1$ and $c_1$ which are used in the up/down counter to estimate the skew. The de-bounce state machine also generates signals $q'_1$ and $q'_2$ which are used to generate the signals $c_1$ and $c_1$ as shown in Fig. 3.11. To ensure that the en$_1$ and en$_2$ do not incorrectly get triggered by the falling bouncing edges of $g_1$ and $g_2$ near the legitimate rising edge or vice versa, the starting levels of en$_1$ and en$_2$ are enabled after counting the continuous run of zeros or ones till a timer counts till a threshold value. After initialization, the de-bounce state machine detects the first rising edge on the sampler outputs, $q_1$ and
Figure 3.11: Block diagram of the delay measurement unit. Input to the block are the beat signals and the sampling clock. The outputs are the skew count($\frac{\delta}{\Delta T}$) and period count($\frac{T}{\Delta T}$).

$q_2$ asserts the signals $en_1$ and $en_2$ respectively to cover for the high duration of the beat signals $q_1$ and $q_2$. The high duration of the beat signal is determined by a timer clocked by the sampling clock. After the count value of the timer crosses the threshold, $en_1$ and $en_2$ follow $q_1$ and $q_2$ until the first falling edges of $q_1$ and $q_2$ are detected. During the duration when $en_1$ and $en_2$ are high, the signals $q'_1$ and $q'_2$ follow the input signals $q_1$ and $q_2$, and are tied high when $en_1$ and $en_2$ become zero. The mask signals, $g_1$ and $g_2$ rise on the rising edge of $en_1$ and $en_2$ respectively. From the time of observation, which ever signal of $en_1$ or $en_2$ rises first, waits for the other to rise too and count till a threshold. As soon as threshold count for the one which starts later is reached, both $g_1$ and $g_2$ are de-asserted to de-assert $c_2$ and $c_2$ simultaneously. Now the signals $c_1$ and $c_2$ contain only the rising edge information for the original signals, which are subsampled, and hence their histogram analysis gives the rising edge statistics. The value set for the threshold count value is depends on $N_T$. In our implementation, it is set to 16 and this value works for all values of $N_T$ within the range 100 to 1000 in our setup.
Figure 3.12: State machines and timing diagram of various signal to be used for histogram of arithmetic difference method based clock skew estimation (Scheme-II).

The problem of skew measurement for delays around $T_{2}$ in presence of jitter, in scheme-I is solved by the DMU proposed in scheme-II as shown in Fig. 3.13.

It can be noted that, depending on the initial condition, the estimated skew can be measured as positive or negative depending on whether $g_1$ triggers the counter or $g_2$. E.g. In Fig. 3.12, if the measured skew is denoted as $N_\delta$ if DMU starts operation from the time instance $t_1$, then for the starting time instance $t_2$, the same skew will be measured as $-(N_T - N_\delta)$ since the counting happens with respect to $q_2$ in the later case. It may appear that, by starting the counting operation only with one particular signal makes it single valued. But as shown in Fig. 3.14, in case of skews around zero, and the signals contaminated with jitter, the estimates become erroneous as the
Chapter 3. Delay Measurement Unit

Figure 3.13: Illustration of the capability of the DMU scheme-II in measuring delay of $\frac{T}{2}$ with jitter.

Negative skews get missed out in this method while accumulating. There are some corner cases where there can be erroneous output within the first two beat cycles. Hence the up/down counter of the DMU is enabled after observing 3 beat cycles.

Figure 3.14: Illustration of missing negative skews while starting the DMU counts with the positive transition of a particular input (around zero skew with jitter).

The standard deviation of the estimate from the DMU varies with the number of samples taken for averaging is shown to vary by the relationship (Detailed proof is provided in Appendix-A):

$$\sigma[N_s] \propto \frac{1}{\sqrt{2^k+1}}$$  \hspace{1cm} (3.1)

Which means, for lesser number of samples, the precision of the delays measured from DMU will be less, whereas it can be improved at the cost of higher number
of samples, and hence higher measurement time. This feature of the DMU is used to provide less precise results to the controller at faster rate in fast mode and more precise measurements in slow mode by taking a larger time for estimation.

Even though the skew and period count are computed with respect to $\Delta T$, the absolute value of $\Delta T$ does not matter till it falls within a reasonable range. Hence, as long as the frequency of the sampling clock signal does not drift during a single measurement time, the results obtained would not be affected.

### 3.5 Conclusion

In this chapter, we looked at some of the conventional ways for potentially estimating precise delays between two inputs from their subsampled outputs.

The conventional methods phenomenon of subsampling from both time and frequency domain. With subsampling, along with the output period, the delay between two subsampled signals generated from two inputs signals of same frequency is also amplified. This delay amplification helps in reducing the complexity of measurement circuit. We also looked at some of practical challenges faced while using the subsampling principle for high precision clock skew measurement along with some possible solutions. Asynchronous sub-sampling followed by statistical averaging allows measurement of static skews between periodic signals. The proposed techniques of de-bouncing followed by averaging of the arithmetic difference of the signals remove any dependency of resolution with sampling clock jitter, unlike in previous works.

In the next chapter we will look at the measurement setup for validating the proposed clock skew measurement technique and would discuss about the results.
Chapter 4

Test Setup for validation of DMU and Measured Results

4.1 Introduction

In this chapter, the experimental setup used to validate the precise skew measurement system is discussed. The experimental results from a prototype involving delay measurement front-end test chip fabricated in 65 nm process are also presented.

4.2 Experimental Setup

The setup used for the validation of the delay measurement system is shown in Fig. 4.1. The schematic of the setup is shown in Fig. 4.2. It involves a custom made chip, few cables and a Virtex II Pro FPGA development board. A test chip in a 65 nm process node (Fig. 4.3) was fabricated which was used as the front-end of the delay measurement system. The test structures essentially consist of a number of samplers, buffers and a multiplexer, which provide the front-end for the skew estimator as discussed in the last chapter. The two clock inputs whose skew we want to measure are supplied from outside the chip so that calibrated skews can be introduced and the
performance of the technique can be studied. Similarly, the reference clock was also provided from outside to enable experimentation with different values of $T$ and $\Delta T$. The beat signal outputs from the multiplexers were directly taken out of the chip and processed in a FPGA board, so that various de-bouncing algorithms and digital processing options could be experimented with in a flexible manner.

Since we did not have signal generators to generate two input clocks with skews in the sub-pico second resolution, we synthesize such delays using a cable of fixed length, and varying clock frequency. A single clock source is passed through a cable of length chosen to provide a delay of about one clock period. Then the relative delays between the edges at the input and output of the cable are given by the difference in the propagation delay through the cable and a clock period. Thus, precise delays between the edges of the input and output of the cable can be obtained by adjusting the clock period (Fig. 4.4). Good quality RF coaxial cables are chosen to ensure that the signals through them are not distorted due to attenuation of high-speed
components. For the two clock sources, two vector signal generators (R&S SM300 and Agilent E4428C) were used. The sinusoidal signals were converted to square waves using a high speed comparator ADCMP562. The rms jitter at the input of the test chip was measured to be 30ps.

The highest frequency of operation was limited by the I/O specifications of the test chip and the peripheral devices for reliable operation. Hence the cable delay and the input delay range were chosen so that it gives zero input delay to the test chip around that frequency.
Figure 4.3: Chip layout of the delay measurement front-end taped out in 65nm industrial CMOS process

4.3 Measurement Results

Fig. 4.5 shows the measured output of the delay measurement unit as a function of the input delays and the residue plot after doing a linear fit, with the choice of parameters $T = 8.33\text{ns}$, $\Delta T = 134\text{ps}$ and $k = 24$. For removing the delay offset due to routing from the chip pads to the sampler inputs, the input delay was swept till the output delay was measured to be zero. Within the test chip too, by shorting the sampler inputs, the input delay could be set to zero which also measured delay very close to zero (actually that gives the delay due to sampler offset). In all graphs which report results for externally provided delays, we have canceled the delay offsets due to the routing from chip pads to sampler inputs. For an input delay range of around $\pm 1$
FO4 delay (±20ps), the measured standard deviation for each point varies between 0.2ps and 0.3ps. The measured maximum error after a linear curve fit (integral error) is 0.65ps.

For the wider input delay range of ±600ps, the maximum integral error is 8ps (Fig. 4.6). We have also tested with larger input delay ranges of ±1.5ns, which results in an integral error of 40ps. The larger integral errors for higher input delay ranges are due to fluctuations in the rise time of the input signal to the chip. This has been verified using an Agilent 54854A (4GHz, 20 GSa/s) Oscilloscope to observe the signals at the chip inputs.

Fig. 4.7 shows the measured standard deviation of an on-chip delay element as a function of the number of samples for an internal delay generator with parameters of
Figure 4.5: Input-output delay characteristic for an input delay range of ±1FO4 delay and the residual error after linear fit.

T=8.33ns, ΔT=134ps, and varying k. The standard deviation reduces with square root of the number of samples up to $2^{21}$ samples and matches well with Equation (A.3).
Figure 4.6: Measured input-output delay for an input delay range of ±600ps.

It saturates to 0.14ps beyond $2^{24}$ samples and hence is the limit of the resolution that is achievable with our setup, leading to a resolution of 0.84ps. Theoretically, with the assumption of independence between each of the core clocks and the sampling clocks, the variance of estimated skew decreases monotonically as the number of samples increases. However, in a practical set-up, correlations between the core clocks (leaf nodes) show up as they are derived from the same source. Moreover, the samples across time are also correlated due to $1/f$ noise in the signal sources and their statistics are time dependent. Hence, the variance of estimated skew is lower bounded due to these correlated jitter components, which cannot be reduced by simple averaging.

Adding jitter to the signals doesn’t affect the resolution and in fact improves resolution in certain cases. For example consider the case when the sampling clock is rationally related to the measurement clock with period as $T_s = \frac{P}{Q} T$. In this case, the measurement resolution is limited to $\frac{T}{Q}$. However it can be increased to almost the same level as that of the asynchronous sampling case by using frequency modulation...
on the sampling clock. This acts as jitter and randomizes the sampling edges, mimicking the asynchronous case. For validating this in the lab, we connected the reference out signal of the core clock source to the reference in point of the sampling clock used in Fig. 4.2. With this change both the sources are frequency locked to a single crystal and using their internal PLLs they generate rationally related core and sampling clock frequencies. The sampling clock source has the option of providing frequency modulation, which is used to create artificial jitter. Fig. 4.8 shows the standard deviation of measured delay of an internal delay source for $T=8.33\text{ns}$, $P=119$, $Q = 120$ with and without frequency modulation. The error in measured delays is large for the case of no frequency modulation. However, measured standard deviation decreases as square root of number of samples similar to the asynchronous case of Equation (A.3), for FM deviation of $5\text{kHz}$. The results are same for frequency modulation frequencies of $20\text{Hz}$ or $100\text{Hz}$ except the fact that for smaller modulation indices (FM dev. =20 Hz), the minimum standard deviation obtainable was 0.3 ps where as with higher FM
Figure 4.8: Measured standard deviation as a function of number of samples for rationally related sampling clock case with FM, for different modulation indices.

dev. of 5KHz and 80KHz it could be improved to 0.14ps. Such a phenomenon where noise improves resolution is well known in threshold systems [42]. Fig. 4.9 shows the measured delay of an internal delay element across different ΔT for T=8.33ns. For the cases when $\frac{T}{\Delta T}$ is integer, the error becomes high. The number of samples for each measurement is at least $2^{26}$. With frequency modulation, even for the integer ratio case, the delay error reduces to that for other values of ΔT as shown in Fig. 4.9.

Besides the averaging error, the other sources of error in the skew measurement are the mismatches in the input offset voltage of the two samplers and the skews in the sampling clock. With a voltage offset of $\Delta V_{\text{sampler}}$ between the two samplers and a input slew rate of $S_r$, the error introduced is $\frac{\Delta V_{\text{sampler}}}{S_r}$, and needs to be minimized by careful sizing and fast edge rates. If the two test nodes are close by, then we can also apply offset compensation scheme like in [32] by giving zero delay inputs to the samplers and compensating their offsets by digitally calibrating their trigger points.
The skew at the sampling clock inputs of the two samplers must be minimized by careful routing. This component of error is bound to be there in any scheme where the skew between two distant nodes are to be measured as that necessitates a reference clock.

The measurement time is $2^k$ sampling clock cycles and we can relate it to standard deviation and hence resolution $\sigma_s$, from Equation 3.1, as:

$$\text{measurement time} \propto \frac{T}{\sigma_s^2}$$  \hspace{1cm} (4.1)

The measurement time per conversion is around 140 ms in order to obtain a ±3\(\sigma\) resolution of 0.84 ps. In case of shared measurement across different leaf nodes, the measurement time will be increased by the number of leaf nodes for which the measurement has to be performed. Hence, there exists a trade-off between the area overhead and the measurement time. But, since the skew measurement unit requires
relatively small area (around 1K NAND2 equivalent gates) and doesn’t need any placement constraints, multiple copies of it can be replicated to reduce the total measurement time for shared measurement cases. Because of relatively small gate count and very low activity factor (since the data inputs are sub-sampled inputs), the power consumed in the delay measurement module is quite insignificant.

4.4 Conclusion

In this chapter we looked at the measurement setup to validate the clock skew measurement technique. Measured results from a 65 nm test chip indicate the ability to measure skews with a $\pm 3\sigma$ resolution of 0.84 ps and integral error of 0.65 ps for an input skew range of 1 FO4 delay ($\pm 20$ ps). The technique can also be used to measure larger skews even close to $\pm \frac{T}{2}$, where $T$ is the clock period. The precision is unaffected by clock jitter as a measurement resolution of 0.84 ps is obtained with clock sources with 30 ps rms jitter. This is further validated by experiments where frequency modulation on sampling clock preserves the resolution. In fact, in certain cases where the sampling clock is rationally related to core clock, frequency modulation improves resolution, which is otherwise degraded.

In the next chapter we will look at some of the applications for the delay measurement technique for measuring on-chip uncertainty.
Chapter 5

Characterization of On-Chip uncertainty

5.1 Introduction

With technology scaling, the gap between the actual performance between simulated and the actual results on silicon has widened. Hence elaborate care has been taken to measure, characterize and model various process and environment variations. The impact of the variations can be either static or dynamic. The static measurements like drain current or voltage threshold of a transistor provide information on the process parameters. But dynamic parameters like delay provides information on the parasitics [8]. One of the most commonly accepted methodologies adopted to measure delays in a certain process technologies in silicon is through ring oscillators [8], [9], [10], [11], [12]. But the traditional ring oscillator based methods only provide information on the global variation i.e. the component of process variation that is common among all the circuits under test. With the increased local process variation scenario, it is very useful to characterize local process variation too by measuring the delay provided by each elementary circuit.

One method capable of measuring the delay of a single inverting or non-inverting
stage embedded in a ring oscillator like structure is discussed in [13]. By being able to measure the average delay of each inverter, the authors demonstrate the capability of such technique in evaluating some of the recently discussed factors affecting on-chip performance, like poly-pitch effect, stress effect etc. However, the technique fails in measuring the rise or fall delay on a single structure. In this chapter we show a simple application of this technique proposed in the previous chapters, to measure the mismatches in the sampling instances of samplers using a test structure as well as the individual gate delay of a single buffer. In the next section, we discuss about an area efficient method to evaluate the clock skew in a distributed environment where it needs to be measured across multiple test points.

5.2 Evaluation of sampler mismatch

In a sampling based approach, the mismatch between samplers leads to an error in the measurement. Hence, an on-chip measurement of this uncertainty would provide the information on the expected error component in the measurement. The setup used to measure the setup window mismatches of samplers using is shown in Fig. 5.1(a). Eight pairs of samplers are fed by the same input signal and each pair’s output is fed to the delay measurement unit. Note that a single delay measurement unit can be used to measure delays across many pairs of nodes, thus reducing the area overhead significantly. Since the samplers are laid out in close proximity, there is no input skew in the data as well as the clock inputs and what is measured is the effect of sampler mismatches. Fig. 5.1(b) shows the measured sampler mismatch for the eight pairs to be within 1 ps with a standard deviation of 0.14 ps with $T=8.33$ ns, $\Delta T=134$ ps and $2^{23}$ samples taken for averaging. The rather large offset of the sampler mismatches is due to poor rise time of the sampling clock, which was confirmed with post layout simulations.
(a) Set-up to measure mismatch between eight pairs of samplers.

(b) Measured results.

Figure 5.1: Characterization of sampler mismatch.
5.3 Evaluation of individual gate delay

In another experiment, the delay of an individual buffer was measured using the setup as shown in Fig. 5.2(a). The dependency of delay with respect to the supply voltage was evaluated. The measured delay in Fig. 5.2(b) was close to the layout-simulated delay of the internal buffer. Lesser number of samples were required in this measurement as compared to those taken for Fig. 4.5 and 4.6 as in this case, the input delay being that of an internal buffer is more stable than the externally fed cable delays. As shown in the figure, delay increments of less than 1 ps can be resolved.

The technique can measure rise or fall delay of any circuit. But the measured delay is actually a mixture of the actual delay and the delay arising due to sampler mismatch. To reduce sampler mismatch, relatively bigger size samplers need to be used which can load the design under test (DUT). As long as the DUT is able to drive the sampler, the method can accurately measure it’s propagation delay.

5.4 Distributed Clock Skew Measurement Problem

In the case of a typical clock distribution network over a synchronous timing domain like a large digital IP core or an I/O interface, the maximum clock skew generally exists between two distant leaf nodes. Hence, to evaluate the quality of the clock distribution network through maximum clock skew, the arrival time difference of clock signal between two synchronous periodic signals which are physically separated by a large distance also need to be measured. These timing domains can be physically very large, of the order of millimeters. In such cases, to determine the clock skew between any two leaf nodes, additional techniques are needed to enable skew estimation in a distributed environment with the TDCs.
Figure 5.2: Characterization of dependence of delay of an internal buffer on supply voltage.

One natural choice would be routing the signals from each test node to a central delay measurement unit (or TDC) with balanced path delays as shown in Fig. 5.3. But this is challenging when the relative skews amongst a large number of nodes needs to be measured. Any mismatches in routing from the nodes to the measurement unit will limit the measurement accuracy. Balanced propagation of many high speed signals from arbitrary physical locations to a central unit will lead to significant
routing challenges.

An indirect method of skew estimation can be adopted from the principle used in [6]. The global clock skew of the Itanium-2 processor was controlled by comparing the skew between two close-by domains and recursively use this method to control the skew to the minimum. Instead of a phase detector, a TDC can be placed at each test location to measure skew between two close by domains. But, in a practical measurement system, with the actual skew, there exists some residual error which is unavoidable. Therefore, each measurement would be containing an error $\delta_{i,j}$ while measuring skew between the $i^{th}$ and $j^{th}$ node as shown in Fig. 5.4. This would lead to large errors when the delay between any two points is calculated using the skew information from multiple measurements having some error.

An alternative approach is to distribute the skew measurement unit to each node and distribute a single additional reference signal to each of these locations. Similar approach is used in [43] to reduce the maximum clock skew. The measurement units can then measure the skew at each of the nodes to the incoming reference signal as shown in Fig. 5.5. As long as the skew of the reference signal is deterministic, the skew between the nodes can be calculated. Existing time-to-digital converters in literature are very expensive in terms of area and power for this application.
Figure 5.4: Illustration of the indirect method of skew measurement using the skew estimates from close-by nets.

Figure 5.5: Illustration of the method of skew measurement using a zero skew reference signal for clock skew estimation at each test node using a TDC.

An ideal approach would be a method by which the timing information at each leaf node is captured with minimum area cost. The timing information then needs to be sent to a central delay measurement unit with minimal constraints to enable
the accurate and precise measurement of clock skew between the input signals. The rest of the thesis explores such a scheme. Just a sampler is used at each of the node to capture the timing information from the local clock. The samplers are clocked by a sampling clock (which is equivalent to the reference signal). The sampling clock has a slightly different frequency from the node clock. The output of each sampler is a beat signal of much reduced fundamental frequency and is synchronous to the sampling clock. The beat signals from the different nodes preserve the timing information which can then be extracted by further digital processing in a central measurement unit. The routing of the beat signals to the central measurement unit needs to be done with a very relaxed timing constraint of the sampling clock period. A multiplexer is used to select a pair of beat signals to determine their relative skews. Thus relative skews between all pairs of leaf nodes can be determined. The method of generating a beat signal by sampling an input signal with another with its frequency a little offset with respect to the input is called “subsampling”. The proposed delay measurement scheme involving subsampling takes advantage of the periodicity nature of the input leaf node signals to reduce the hardware requirement at each test node (essentially a sampler). Hence, the proposed method scales well for higher number of test nodes with very small incremental area requirement.

5.5 Conclusion

In this chapter we looked at three applications of the proposed subsampling based delay measurement technique. The on-chip gate delay measurement technique can is suitable for characterizing on-chip variability in terms of total delay or rise and fall delays. The sampler mismatch evaluation technique can provide the information on the accuracy that can be obtained using the subsampling or any other sampling based technique, where the mismatch between two samplers would directly contribute to measurement error. The use of subsampling technique in the context of skew
measurement in a distributed network greatly reduces the area overhead per test node. It also relaxes the overall hardware challenge.

In the next chapter we will discuss at a system level application of the delay measurement technique for generating accurate delays in an all digital manner.
Chapter 6

An Accurate Fractional Period Delay Generation System (FPDGS)

6.1 Introduction

A calibrated delay generation system is a standard requirement in any digital integrated circuit both in design and the testing phase. A functional view of a programmable delay generator is shown Fig. 6.1. The desired delay is input to the system by a digital code setting (λ). The output signal coming out of the system is delayed by an amount given by \( \lambda \times T \) where \( T \) is the time period of the input clock. The circuit block with this functionality is interchangeably referred as programmable delay or phase generator (PDG), digital-to-phase-converter (DPC) and digital-to-delay converter in literature. In the field of instrumentation and measurement, a PDG finds major applications in Automatic-Test-Equipment (ATE). Some examples of such ATE are VLSI functional tester, PLL tester, Integrated Circuit pulse parametric tester, system trigger, laser diode tester, timing generator, time-to-digital converter (TDC) tester, delay compensator or pulsed RF measurement equipment etc [31]. A PDG is also an important building block in serial communication systems using source synchronous interfaces, where the clock is used along with the data bits
Figure 6.1: Functional view of a Programmable Delay Generator.

to transmit and receive data. For ensuring the sampling of the high-speed data perturbed by channel noise at the best possible instant at the receiver end, the clock at the receiver end (strobe) is given an optimal delay shift depending on the mode of operation. This is done by controlling the phase of the strobe signal with the PDG [44]. Digital phase modulators used in communication systems also use PDG. Such a system can also be used to modulate or correct the duty cycle of a circuit [45]. Since today’s CMOS fabrication process is optimized for digital technology, efforts are being made to replace the low yield analog blocks with equivalent digital blocks. One such approach is Time-mode-signal processing where the controlling parameter is time rather than voltage [14], [15], [16]. A PDG is a fundamental component in such systems. Recent approaches for building high resolution all-digital PLLs involve PDGs in the control loop as a replacement for analog blocks like charge pump and VCOs [46].

One major challenge while building a high resolution PDG in deep sub-micron process is maintaining accuracy in spite of the increased process variation. In this
paper, we propose a method for building an accurate PDG for generating a delay ranging from 0 to T (time period) for periodic signals. The proposed system is compatible with standard digital technology and its effectiveness is demonstrated on an FPGA driven prototype. An order of magnitude improvement in accuracy is achieved by using the proposed technique over a commercial delay generation chip.

The chapter is organized as follows: in Section 6.2 we discuss some conventional approaches for programmable delay generators and point out their limitation in achieving good accuracy across the entire period. Section 6.3 discusses the proposed system to make an accurate programmable delay generator along with the description of all the building blocks. The results obtained from a prototype built using a Virtex II Pro FPGA board and a commercial programmable delay generator are shown in section 6.4. Section 6.5 summarizes this chapter.

## 6.2 Conventional Methods of Programmable Delay Generation

Existing methods of multiple phase generation can be classified into two broad categories i.e. locking based \[44],\[47],\[48],\[32\] and calibration based \[45],\[49],\[50],\[51],\[52],\[53\]. Locking based approaches use a DLL to lock to a phase of $2\pi$ through a set of controllable delay buffers. The delay buffers can be controlled by an analog voltage or a digital setting. Since the minimum delay that can be achieved from a buffer is quite coarse, interpolators are used to get resolution less than minimum gate delay \[44],\[48\]. In all these architectures, circuit innovations are done to generate delays with good resolution by keeping the sub-phases as much close to each other as possible. But with increasing process variability, it becomes difficult to maintain the accuracy and resolution at the same time.

A simple block diagram of such a system is shown in Fig. 6.2. The delay of the signals tapped at the end of first, second ... $N^{th}$ stage with respect to the feedback point (X) is $D_1$, $D_2$, ..., $D_N$ respectively. Due to local mismatch among the delay stages, the delay step added by each stage will vary. To quantify that effect, a simple
mathematical model of the system is constructed. The delay of each stage ($\tau_i$) can be split to have a global component $\tau_0$ (constant across all stages) and a local random component $\delta T_i$. The $\delta T_i$s are assumed to be independent and identically distributed (i.i.d.) having a Gaussian distribution with zero mean and standard deviation $\sigma[\delta T]$. Hence, The delay of the $i^{th}$ stage can be written as:

$$\tau_i = \tau_0 + \delta T_i \quad (6.1)$$

The delay for the signal tapped after $i^{th}$ stage i.e. $D_i$ will now be given by:

$$D_i = \sum_{k=1}^{i} \tau_k = i\tau_0 + \sum_{j=1}^{i} \delta T_j \quad (6.2)$$

The delay for the signal tapped after $N^{th}$ stage is

$$D_N = \sum_{k=1}^{N} \tau_k = N\tau_0 + \sum_{j=1}^{N} \delta T_j \quad (6.3)$$

Since the delay at the end of the $N^{th}$ stage is kept constant by the Phase detector (PD) and Charge pump (CP) to the match input input clock period, $D_N$ is fixed and
\( \tau_0 \) is adjusted by the loop to make
\[
\tau_0 = \frac{D_N - \sum_{j=1}^{N} \delta T_j}{N} \quad (6.4)
\]
Replacing this value of \( \tau_0 \) in equation 6.2, the values of \( D_i \)'s can be calculated as
\[
D_i = \frac{i}{N} D_N - \frac{i}{N} \sum_{j=1}^{N} \delta T_j + \sum_{j=1}^{i} \delta T_j \quad (6.5)
\]
\[
\Rightarrow D_i = \frac{i}{N} D_N + (1 - \frac{i}{N}) \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=i+1}^{N} \delta T_j \quad (6.6)
\]
The variance of the delay at the \( i^{th} \) tapping point is given by (Detailed derivation is provided in Appendix-B):
\[
Var[D_i] = \frac{i \ast (N - i)}{N} Var[\delta T] \quad (6.7)
\]
which peaks for \( i = \frac{N}{2} \) and the peak value of the uncertainty in terms of standard deviation is given by
\[
Max\{\sigma[D_i]\} = \frac{\sqrt{N}}{2} \sigma[\delta T] \quad (6.8)
\]
Fig. 6.3 shows the standard deviation of the generated delays across \( i \) for a period (\( D_N \)) of 5 ns, \( N=100 \) and \( \sigma[\delta T] = 4ps \). The values used for simulation were assumed considering the standard deviation numbers for intra-die variability from a typical 130 nm CMOS process. It can be observed that the peak value of uncertainty can be around 120 ps (\( \pm 3\sigma \)). Since the minimum buffer delay in a CMOS process is higher than the desired resolution, in most PDGs interpolators are used to get finer delays. The argument discussed above still remains valid for the interpolation based architecture. Thus in conventional DLL based techniques, the delay at the extreme ends of the delay chain are checked using a phase detector. However, if the desired
delay is farther from the two ends of the chain, the accuracy degrades as shown in Fig. 6.3 Therefore, an ideal architecture would use actual generated delay itself in a feedback to ensure accuracy.

[31] uses two PLLs with small frequency offsets to generate precise one-shot delays. But the technique is specifically oriented towards generating one-shot delays proportional to the digital code word and can not be applied for fractional periodic delay generation. Most of the calibration based approaches [50],[51],[52] use a separate calibration phase to reduce the error. Therefore, they can’t be adopted for applications requiring uninterrupted signal to be available for a long time. Moreover, with slow temperature variations, the delays generated by these calibration based systems can change causing an increase in error. Some calibration based approaches generate a physical signal to calibrate the generated delayed signal against a reference. For example, [49] proposes a high resolution TDC within a FPGA using dynamic reconfiguration where a variable frequency oscillator is used for the calibration to ensure the accuracy of the intermediate step delays against temperature variation and mismatch. In the calibration phase, the variable frequency generator is used to generate the reference signal whose phase is compared with each of the delay elements to find
the nearest delay stage for a required delay. The frequency of the variable frequency oscillator drifts with time due to increase in temperature making re-calibration necessary at regular intervals when the system needs to be put on hold. The hardware required and time required for calibration process is also relatively high.

A high resolution digital to time converter was proposed [53] where an integrated Dual Mixer Time Domain (DMTD) circuit was adopted to overcome device mismatch, process variations and temperature for self-calibration during normal operation. Similarly, [45] proposes an on-chip measurement and continuous correction methods for correcting output duty cycle where random sampling technique is used for delay estimation. However, no experimental result is demonstrated in [53] and [45] to reveal the actual performance of the PDG. Both these techniques use a conventional XOR based approach which can give an erroneous estimate for skews around zero [40] in the presence of jitter. A technique to provide solution to all these problems, and to enable a continuous closed loop feedback ensuring good accuracy in achieving a desired fractional period delay with little area overhead is discussed in the next section.

### 6.3 Proposed Accurate Delay Generation System

Fig. 6.4 shows the overall system proposed to implement a feedback based scheme of delay generation. The proposed delay generation system consists of three major blocks: Delay Measurement Unit (DMU), Delay controller, Controllable delay line. The DMU estimates the actual on-chip delay between the input and delayed output in terms of a proportional count value. This count value is used by the delay controller to close the feedback loop by generating appropriate control signals for the controllable delay line.

The delay measurement unit is based on subsampling principle to measure precise delay where the input and delayed output signals are subsampled by another asynchronous clock signal having a small frequency offset with respect to the input signal frequency. The signals coming out of the samplers are the beat signals having much lower frequency (equal to the difference of the input clock and sampling clock
frequency). These beat signals are then processed in a delay measurement unit which does the required processing and averaging to estimate the input skew precisely. With the subsampling phenomenon the beat signals are synchronous to the sampling clock and all the measured delays are in terms of some counter values ($N_\delta$) which can be multiplied with $\Delta T$ (the difference of periods of the input and sampling clock) to extract the absolute time units.

The measured delay ($N_\delta$) is sampled by the delay controller unit at an interval when the estimate of delay is ready. The control unit uses the estimate to control the delay chain to increase/decrease the delay in the signal path to converge to make the difference between the target and the generated delay to the minimum possible value. The target delay is computed from the target ratio (of target delay to the clock period provided as the input to the unit) by multiplying the period count ($N_T$) from the delay measurement unit with it. The delay measurement unit is configured to run in two modes (fast/slow) depending on the difference between the measured delay and the target delay. The mode signal is provided by the delay controller unit.
Figure 6.5: Architecture of the delay control unit. It takes desired ratio, threshold count, estimate done, $N_T$ and $N_\delta$ as input and provides the control signals to the controllable delay line as well as sets the fast/slow mode of operation for DMU.

The delay controller also provides the required signals to activate the controllable delay line to converge towards the target delay and maintain the error to be as small as possible.

The controllable delay line can be of any architecture to provide precise delay steps for the input digital code word. E.g. it can take the shape of a simple inverter and RC chain based delay line structure providing a coarse fine architecture [36].
Figure 6.6: Illustration of a case to explain controller details. (a) illustrates the concept of cyclic compliment of error. (b) shows a snapshot of the loop dynamics as the error crosses zero.

### 6.3.1 Delay Controller

The details of the delay controller unit are shown in Fig. 6.5. The block latches the values of the measured delay \(N_\delta\) and period count \(N_T\) when the signal Estimate done becomes high. Depending on the starting point of the delay estimation unit, \(N_\delta\) can take values from \(-\frac{N_T}{2}\) to \(\frac{N_T}{2}\) or 0 to \(N_T\). The desired count is also computed from the input Targetratio by multiplying it with \(N_T\). Since the desired ratio is given as an input fraction taking values in the range \([0,1)\), the target count \(N_{TC}\) can take values from 0 to \(T\). Hence, for comparing the two count values, they have to be mapped to the same domain/range. That is done by adding \(N_T\) to \(N_\delta\) if \(N_\delta < 0\) and the number thus generated is called \(N_{\delta,map}\). The numerical difference between the \(N_{TC}\) and \(N_{\delta,map}\) quantifies the error from the target. But, this computed error can sometimes be very high due to the adjustment done to map \(N_\delta\) and \(N_T\) into same range, e.g. when the delay changes from a small positive value to a small negative value or vice-versa. In that case, a large difference between the absolute values of the errors is observed after mapping, even though they would be very close before mapping. To deal with this,
the cyclic complement of the error (shown in Fig. 6.6) is computed by taking the complement of absolute value of mapped error ($\epsilon_{map}$) with respect to $N_T$ along with $\epsilon_{map}$ and the entity having minimum absolute value among them is chosen to be the effective error used for generating the final signals for controlling the delay generation module.

The $Z^{-1}$ blocks shown in Fig. 6.5 provide one measurement cycle delay. To make the generated delay stable at the same minimum error, the control unit compares the absolute value of the present error with that of the error generated in the last measurement cycle. If the minimum obtained from this comparison is within a threshold distance from its value in the previous measurement instance, it indicates the achievement of minimum error. Hence, the delay enable signal controlling the delay generation unit need not be activated since the minimum error is inferred until the gap between these differences is within the threshold. The threshold is set to be the count value corresponding to the average step size of the delay generating unit. Otherwise, the enable signal is activated to trigger a change in the delay generation unit. A snapshot of the loop dynamics as the error crosses zero is given in Fig. 6.6.

In this figure, the effective errors are plotted against time. While approaching the minimum from positive side, the effective error takes values 20, 10, -2.3, 9, -2, -2.1, -2.2, -1.9 respectively according to the control loop. The minimum absolute error values estimated during the run are 20, 10, 2.3, 2.3, 2, 2, 2 respectively. Once the sequence of two close values of minimums (2.3, 2.3) is obtained, the enable signal is not activated further and hence a stable delay is maintained. A small variation is expected between successive delay estimated because of the inherent time varying nature of the input skew. The direction of change is evaluated based on the sign of the effective error to minimize the difference between the target and the generated delay.

The control unit also controls the speed mode in which the delay measurement unit works. For converging faster to the target, whenever a new target is desired, the control unit selects the fast mode for the DMU so that the DMU takes fewer samples for averaging and hence the Estimate done signal pulses come at a faster rate and
the delay unit changes the delay more frequently. Once the difference between the
target and the generated delays changes sign (crosses zero), measurements are made
more precise by entering the slow mode where the DMU takes a larger number of
samples for averaging.

The simulation of loop dynamics of the controller to converge to the minimum
possible error is shown in Fig. 6.7. At the outset, the error was 4 ns for an input
period of 10 ns. Since the error value is high, the estimates were done at small time
intervals of 3 ms ($2^8$ beat cycles with a $\sigma$ error of $\approx 6$ ps). Once the zero crossing
is detected through fast estimation, the controller enters into slow mode, taking $2^{16}$
beat cycles for each measurement to ensure high precision ($\sigma \leq 0.3$ ps) and accuracy
in the measurement. It can be observed that within approximately 3 seconds, the
loop converges to the minimum possible error ($\approx 4$ ps) even though traversing the
maximum possible distance $\approx \frac{T}{2}$. The minimum error achievable is decided by the
resolution of the delay chain which is 10 ps in the example. The inc/dec signal coming
out of the controller changes the delay setting through an accumulator.
6.3.2 Programmable Delay Element

The delay generation unit generates a delay proportional to the digital code word provided by the delay controller. It can be of any architecture, such as a coarse-fine architecture or an interpolation based architecture or any other architecture which can increase or decrease the delay of the input clock signal in fine steps. As long as the delay chain is capable of covering the full period, the system can generate delays of any fraction of the input clock period. The resolution of the delay element also limits the accuracy achievable by the proposed closed loop architecture. Ideally, the delay generation unit should provide uniform and predictable delay steps. However, process variation can cause the step size to vary. Hence, for maintaining high accuracy the delay generation unit should be placed in a closed loop.

6.4 Experimental Setup and Measurement Results

The experimental setup for the complete system shown in Fig. 6.4 is built in the lab for evaluating the proposed architecture. The lab setup is shown in Fig. 6.8.
Figure 6.9: Schematic of the measurement setup used to validate the proposed architecture.

The schematic view of the lab setup is shown in Fig. 6.9. All other blocks in the system except the programmable delay element are synthesized in a Virtex II FPGA development board. The programmable delay element is implemented by using an external board with the CDCF5801A chip mounted in it. This chip is capable of generating fine delays in steps of roughly \( \frac{1}{3072} \)th of the clock period. The chip requires two signals, \textit{enable} and \textit{direction} \[54\]. For every rising edge in the \textit{enable} signal, the delay produced by the chip is increased or decreased corresponding to a logic zero or one on the \textit{direction} input. Since the accumulator shown in Fig. 6.5 is actually embedded into the variable delay line, in the experimental setup only two signals come out of the FPGA to control the variable delay line. The input clock given to FPGA is routed as an input to TI’s CDCF5801A chip and the delayed signal from the chip is fed back to the FPGA. The delays between the signal routed from FPGA to delay chip and imported back from the delay chip to FPGA are measured inside FPGA. Control signals are also connected between the FPGA and the delay control chip.
Because of the external signal routing of the input clock signal and specifications of
the delay chip for obtaining the finest steps, the frequency of operation was limited
to within 50 MHz. Beyond that frequency, the shape of the high frequency signals
deteriorate impacting the precision.

To measure the effectiveness of the implemented architecture, a comparison was
made between the error estimated with and without implementing the proposed closed
loop architecture. In the open loop case the desired steps required to generate any
specific delay is precomputed as per the data sheet\cite{54} and the delay chip is triggered
that many times. Since the initial delay provided by the delay chip along with the
routing delays is unknown, first the zero crossing of delay is detected before counting
the steps to generate specified delay. In the closed loop case, the proposed architecture
is implemented. The difference between the measured and the desired delay (INL) is
plotted in Fig. 6.10(a). Similarly, the difference between the measured step sizes be-
tween two consecutive settings and the ideal step size (DNL) is shown in Fig. 6.10(b).
Although the data points could be plotted in steps of 10 ps , to maintain clarity,
the target delays were provided to span the entire period of 27 ns (37 MHz input
clock) with 64 steps. The value of sampling clock frequency was chosen to be 36.927
MHz for the graph shown. The same trend is also observed by choosing other nearby
sampling clock frequencies, proving the insensitivity of the proposed architecture to
drift in the sampling clock frequency (e.g. due to temperature variation). Choosing
different input clock frequencies also gives similar result and hence, for avoiding re-
dundancy, only one example case is shown. In the fast mode the DMU takes $2^8$
beat cycles for each measurement, whereas it takes $2^{16}$ cycles in slow mode. A precision of
around 1 ps is ensured in the slow mode by taking $2^{16}$ beat cycles for averaging. Up
to 26X improvement in DNL and 40X improvement in INL accuracy is measured by
implementing the proposed architecture.

To cross check the measured delay in FPGA through the DMU, a copy of the input
clock and the delayed output clock signal are taken to a high end Agilent oscilloscope
(20 GS/s Agilent Infinium DSO 90404A). Fig. 6.11 shows the error measured by
the oscilloscope. The maximum error measured by the scope is more than the DMU.
readings because of the fact that, in case of the scope the number of continuous samples taken is quite less. The uncertainty in delay caused by the output buffers of the FPGA driving long external cables and also the limited sampling rate of the scope also makes the scope readings worse. An uncertainty of the order of tens of pico-seconds was observed at the input of the scope for multiple readings for a fixed setting. Still, it can be observed that even when measured with the oscilloscope, the error decreases by 6X by implementing the proposed closed loop architecture.

Fig. 6.12 shows the measured jitter of the delayed output clock (copy taken to the scope through a splitter) with and without the closed loop activated. It can be observed that both the curves take similar shape proving that there is negligible additional jitter added due to the loop. This can be attributed to the care taken in the control unit to avoid multiple switching of delays even though the minimum error is achieved.

Since the delay is linearly incremented or decremented through the delay element, the time taken to lock to a desired delay is linearly proportional to the distance of the present delay to the desired delay. But, because of the fast mode in the DMU, the maximum time required to lock can happen when the distance is $\approx \frac{T}{2}$. For the implemented setup with 37 MHz input clock, and 36.927 MHz sampling clock, this upper boundary is around 4 s which will be scaled down proportionally for higher input frequency. The gate count of the entire unit is $< 6K$ (NAND3 equivalent gates). Because of the relatively small gate count and very low activity factor (since the data inputs are sub-sampled inputs), the power estimate of the entire control logic would be quite insignificant when implemented in an ASIC design.

### 6.5 Conclusion

The proposed closed loop control for a delay generation system allows much better accuracy than the open loop case to generate arbitrary fractional unit interval delays. The elements constituting the system need to be carefully designed to enable stability and precision of delay generated across full range. Asynchronous subsampling
Chapter 6. An Accurate Fractional Period Delay Generation System (FPDGS)

Figure 6.10: Characterization of INL and DNL errors of the system with and without enabling closed loop.

followed by statistical averaging allows accurate and precise measurement of static
Figure 6.11: Measured error (normalized) of the system with and without enabling the closed loop using a 20 GS/s Agilent Infinium DSO 90404A.

Figure 6.12: Measured jitter at the output of the delay generating system using a 20 GS/s DSO.
skews between periodic signals. The proposed system periodically measures and corrects the error to keep it at the minimum and does not require any special calibration mode for error correction. Therefore, the system can run without interruption for a long time with minimum error even if the slow varying parameters like temperature vary with time. Up to 40X improvement in accuracy is measured by enabling the feedback control. Some of the numbers e.g. frequency of operation, jitter at the output etc can be much better if all the components are built on-chip. Since the loop control is slow, the input jitter directly propagates to output.
Chapter 7

On-chip realization of an all-digital FPDGS

7.1 Introduction

In this chapter, we look at the ASIC implementation issues for the design of fractional period delay generating system (FPDGS) discussed in the previous chapter. For medium frequencies (few hundreds of MHz) maintaining fine resolution for the entire period is very challenging and area consuming. Hence, a coarse-fine delay generator based architecture is proposed for on-chip realization of the delay generating system. The coarse delay units help in reducing area to generate large delays whereas the fine delay units help in achieving high resolution. Many all-digital PLLs and DLLs use some variant of coarse-fine delay structures. The actual architecture of the coarse and fine delay generation units is dictated by the specifications of desired resolution, power and area. Unlike the general frequency and delay locking architecture, in the proposed system the delay loop is closed using a digital delay measurement unit (DMU) as the feedback element. A wide range of delays are generated by appropriately setting the control bits of the coarse and fine delay chain as shown in Fig. 7.1. The coarse delay chain can be controlled to generate delays in steps of about 100 ps. The fine
delay chain is then controlled to generate delays in fine steps (< 10 ps) to set the input-output delay as close to the target as possible. The delay measurement unit measures and minimizes the error between the target delay and the actual delay. While the system is in operation, the change in on-chip temperature causes drift in the generated delays with respect to the target delay which in turn is reduced due to the feedback action through the DMU and the controller action. A detailed discussion of the proposed system consists of the issues and implementation aspects related to each of the building blocks which are: coarse delay unit, fine delay unit, synchronizing unit, calibration unit and the controller as shown in Fig. 7.2.

![Coarse Delay Chain](image1)

![Fine Delay Chain](image2)

Figure 7.1: Architecture of coarse-fine delay cell unit.

The organization of the chapter is as follows. First the structure of coarse delay chain and the related issues are discussed in section 7.2. In section 7.3 the possible structures to generate fine delay steps are discussed along with a comparison of their linearity performance. While discussing about the coarse and fine delay chains, some of the existing circuit topologies for the unit cells are discussed and some modifications are suggested to achieve improvement in linearity, power and resolution. Section 7.5 discusses the system response of the coarse-fine delay generation architecture in presence of temperature variation. Section 7.6 describes the calibration unit used in the proposed system to estimate the vital numbers to be used in the delay controller to ensure minimum error across various PVT corners. Section 7.7 discusses the role of the delay controller for the proposed coarse-fine architecture. Section 7.8 shows
Fractional Period Delay Generator based on Coarse-Fine architecture

Figure 7.2: Architecture of all-digital fractional period delay generating system
the measured results from an FPGA implementation as a proof of concept, where the coarse-fine architecture was emulated using the FPGA LUTs (for generating coarse delay steps) and an external delay generation chip (for fine delay step generation).

### 7.2 Coarse Delay Chain

The coarse delay chain consists of several coarse delay cells where each cell is used to provide delays in steps of about 100 ps. Several coarse delay generation techniques exist in literature. Some are inverting in nature whereas others generate non-inverting delays. In the proposed design, since only non-inverting delay is of interest, the latter is discussed. One of the fundamental delay generator unit can be thought of having the structure shown in Fig. 7.3 where cascading the multiple cells of the basic unit can generate higher delays. But with larger number of delay cells, the intrinsic delay (minimum delay in the signal path) also increases and is given by \((\text{Number of delay stages cascaded}) \times \text{delay(NAND2)}\). This limits the maximum operating frequency of the delay unit. A delay unit where the intrinsic delay would be independent of number of delay stages would be ideal to support a large dynamic range of delays.

A delay structure consisting of inverters and multiplexers (MUXes) is discussed in [55] where the delay chain is made by cascading multiple regular structures called lattice delay unit (LDU) as shown in Fig. 7.4. The select lines are set in a thermometric pattern to increase or decrease the delays. By adopting such a structure, the delay per stage is limited to the sum of delays of two inverters and a multiplexer. The maximum operable frequency of this delay structure is governed by the stage delay (around 50 ps in 130 nm process). Hence, this doesn’t create an issue for circuits running at speeds of 1-2 GHz. But the disadvantage of this type of lattice structure is the power consumption with larger number of cells. To provide delays even with lower frequencies, more lattice units are needed to cover the delay equal to the input clock period. Since in such a structure for each transition in the input signal, all the
nets in the forward path of the LDUs make a transition from VDD to GND or vice versa. This leads to unnecessary power wastage since the nets which are outside the signal path are also toggled.

![Conventional Delay Unit](image)

Figure 7.3: Structure of a conventional delay line.

![Inverter and MUX based LDU](image)

Figure 7.4: Structure of inverted and MUX based Lattice Delay Unit.

Authors in [56] propose a LDU consisting of four two input NAND gates as shown in Fig. 7.5. A dummy 2-input AND gate is added in each unit to match the output load capacitance seen by each NAND gate in the signal path. But, due to asymmetry of fall delays for the two inputs of the NAND gate in the bottom returning path, there is a difference in delay for odd to even setting transition and vice versa. The simulated incremental delays for different change of settings in the delay chain are shown in Fig. 7.8. It can be seen from the figure that the mismatch between odd-to-even setting and even-to-odd setting transitions is 7.5 ps. This mismatch can be reduced by using
a symmetric structure for the 2-input NAND gate as shown in Fig. 7.6. With this adjustment the mismatch is reduced to 2.5 ps. But the design takes larger area.

![NAND gate based LDU diagram](image1)

Figure 7.5: Structure of NAND gate based Lattice Delay Unit.

![NAND gate to reduce delays](image2)

Figure 7.6: Structure of NAND gate to reduce the difference of the delays between the even to odd and odd to even setting changes.

The structure shown in Fig. 7.4 is modified by adding a multiplexer in the forward path as shown in Fig. 7.7. The problem of unnecessary power wastage in the inverters out of the signal path is solved in this structure as the multiplexer in the forward path enables the signals to reach only till the inverters in the signal path. The inputs to the inverters outside the signal path are maintained to be static thus resulting is no switching power loss in them. The control words in both the above structures follow a
thermometric pattern. Depending on the desired delay, a chain of continuous ones are fed to the select inputs of the delay chain starting from left to generate incremental delays with steps of approximately two multiplexers and two inverter delays. For example in Fig. 7.7 an extra path is added by changing the setting from \([1000]\) to \([1100]\) for \(S_{K-2}\), \(S_{K-1}\), \(S_K\) and \(S_{K+1}\) respectively, which leads to selection of the adjacent right arrowed path as shown in the figure. The newly followed transition path would be \(M_{U1}(0) \rightarrow M_{U2}(0) \rightarrow MU3(0) \rightarrow MD3(0) \rightarrow MD2(1) \rightarrow MD1(1)\) as compared to the previously followed path \(M_{U1}(0) \rightarrow M_{U2}(0) \rightarrow MD2(0) \rightarrow MD1(1)\).

![Proposed LDU](image)

**Figure 7.7:** Structure of the proposed Lattice Delay unit.

Fig. 7.8 shows the step size of this proposed structure. It can be seen that the mismatch between odd-to-even setting and even-to-odd setting transitions is 2.3 ps. This is due to asymmetry of rise and fall delay of a single LDU when it is in loop back mode. Fig. 7.9 illustrates the case, where for the delay provided to the rising edge of the input the path from \(X\) to \(N_1\) may not match the delay provided to a falling edge. The mismatch between the even to odd setting change or vice versa can be reduced by using notinverting stages both in the top and bottom path as shown in Fig. 7.8(b) whch makes sure that each intermediate stage between two LDUs sees a same edge. But this solution costs increased single stage delay.

In the coarse-fine architecture, the continuity in delay generation is maintained by
Figure 7.8: Linearity comparison of the various types of coarse delay units across various digital code settings. The

Figure 7.9: Illustration for explaining the difference in delays while changing from an even to odd setting or vice-versa.
ensuring that the fine delay chain, even in extreme corners, covers the delay generated by a single coarse delay cell. Hence, a higher coarse delay step size leads to more number of fine delay cells in the fine delay chain, implying larger area requirement. Similarly, the coarse delay cells can’t be arbitrarily over-sized to generate the minimum delay as with higher width of the transistors the internal parasitics increases along with overall area. For determining the appropriate size of the NMOS and PMOS transistors for the coarse delay cell multiplexers and inverters, the width of the NMOS ($W_n$) was swept from 500 nm to 3.5 $\mu$m. The width of the PMOS transistors for the multiplexers was made to be same as $W_n$ while that for the inverters was $2*W_n$. The result of the sweep is as shown in Fig. 7.10. The delay of a single coarse delay cell was found to be minimum for $W_n \approx 1.2\mu m$. Beyond this value of $W_n$ the parasitic capacitance of the transistors dominates the increased drive strength of the inverters. Hence there is eventually an increase in the delay value after $W_n=1.2\mu m$. The corresponding delay step generated by the coarse delay cell is 77$ps$ in Typical NMOS and Typical PMOS corner (in simulation results without layout parasitics).

![Figure 7.10: Delay of a single coarse delay stage for various pass transistor widths.](image)
In the cases where multiplexers are used for generating coarse delays, there exists a possibility of glitch at the output because of change in control setting at an instance where the levels of the two inputs to the MUX being switched are different. Fig. 7.11 shows the scenario which results in a glitch being generated. Initially the input to the delay chain \(D_i\) is set to 0, and the control signals are configured such that the input is propagated to the output after passing through the first two coarse delay cells (actual setting is \([100]\)). We then observe the effect on the output when the input to the coarse delay chain is changed to 1, and at the same time, the control signals are changed to propagate the input edge through three CDCs.

We note that the input of the upper input for MUX \(M_{D3}\) (\(M_{D3}(0)\)) is initially 0 (Case (a) in Fig. 7.11). This is because the MUX \(M_{U3}\) propagates the static value of “1” at its second input (\(M_{U3}(1)\)) which is inverted at \(M_{D3}(0)\). When the input to the CDC as well as the MUX setting are changed simultaneously, the MUX passes the value held at \(M_{U3}(0)\). This causes \(M_{D3}(0)\) to go high (Case (b) in Fig. 7.11). Then the edge transition at the input propagates and reaches \(M_{U3}(0)\), and causes \(M_{D3}(0)\) to go back to 0 (Case (c) in Fig. 7.11). Thus, the change in the MUX setting (which gets applied immediately) causes a change in the output level, which is then restored once the input edge reaches the \(M_{U3}\) input. This generated glitch then propagates to the output.

To solve this, we ensure that the change in the MUX control signals are activated only when both the inputs to the MUX are identical and settled. The input edge which takes time to propagate to the MUX under consideration is used to change the MUX control settings.

This phenomenon is illustrated with the timing diagrams in Fig. 7.12. The control bit settings set corresponding to the coarse delay chain in discussed in Fig. 7.7 are as follows: \(S_{K-2} = 1\), \(S_{K-1} = 0\), \(S_{K+1} = 0\) where as the control signal \(S_K\) changes value from “0” to “1”. In Fig. 7.12 a) the change in control signal \(S_K\) is directly controlled and is set to change at the same instance when \(S_K\) changes. The separation between
Figure 7.11: Illustration of glitch generated at the output due to change of setting in the CDU at unsuitable instant.
Figure 7.12: Illustration of glitch generated at the output due to change of setting in the CDU at unsuitable instant.
dotted lines in the time axis is equal to the delay through one inverter and a MUX. As shown in Fig. 7.12(a), there is a glitch at a signal in the output path (\(M_{D1}(1)\)) which directly propagates to the final output stage. In case of Fig. 7.12(b), where the re-timing of the control signal input \(S_K\) is done to generate the signal \(S'_K\) which is used for the MUXes \(M_{D2}\) and \(M_{U3}\). The signal \(S'_K\) is generated by sampling the original control signal \(S_K\) by a negative edge triggered flip-flop clocked with the output of the corresponding coarse delay stage. This ensures that the inputs of the MUX whose settings are being altered are stable by the time the select signals are actually changed. Similar synchronizing technique for ensuring a seamless transition from one MUX setting to another is employed in [44]. Fig. 7.13 shows the coarse delay chain with the re-timing flip-flops embedded.

![Diagram](image_url)

Figure 7.13: Synchronizers for ensuring glitch-free coarse delay switching.

Fig. 7.14 shows the layout of a single coarse delay cell with the embedded re-timing logic for MUX select signals. The structure takes around 27 \(\mu m \times 7 \mu m\) area of which the re-timing logic takes approximately 14 \(\mu m \times 7 \mu m\).

Fig. 7.15(a) shows the layout parasitic extracted results for the coarse delay cell across various corners and input digital settings. It can be noted that the delay step size varies from 82 ps in Fast NMOS-Fast PMOS (FF) corner to 134 ps in Slow...
NMOS- Slow PMOS (SS) corner with the step size being around 105 ps in Typical NMOS and Typical PMOS (TT) corner. Fig. 7.15(b) shows the power consumed by using a single coarse delay cell across various corners. It is around 9 µW at 100 MHz, with a supply voltage of 1.2 V.

Figure 7.15: Parasitic extracted delay and incremental power results for the coarse delay unit.

(a) Parasitic extracted delay of the coarse delay unit across different settings and corners

(b) Incremental power drawn by each delay stage of the coarse delay unit across various corners

Fig. 7.16 shows the Monte Carlo simulation results for 100 instances with local
mismatches in the coarse delay cell at TT corner. It can be seen that the delay spans around 9 ps and has a standard deviation of 1.5 ps.

![Histogram of parasitic extracted delay](image)

**Figure 7.16:** Monte Carlo simulation result of the parasitic extracted delay of the coarse delay unit.

### 7.3 Fine Delay Chain

The role of the Fine Delay Unit is to generate incremental delays in steps of few picoseconds as well as to cover the delay generated by the coarse delay unit. Since any discontinuity between the maximum delay generated by the fine delay unit and the delay generated by a single coarse delay cell can lead to error in delay generation, the fine delay unit is designed such that its dynamic range covers the delay of a coarse delay cell across all PVT conditions. There are some techniques in literature that can be considered for delay generation in fine steps. One such approach is adopted in [57] where the Clock Vernier Devices (CVDs) are used to generate delays to tune the delays of the clock path in fine steps of few pico seconds. Fig. 7.17 shows the schematic of the basic element. Four such elements were cascaded to span the range of a coarse delay cell and the linearity of such a structure was evaluated. Fig. 7.18 shows the linearity of such a structure across various digital settings of the switches.
The delay generation is based on the principle of modulation of the fight current from the delayed non-inverting version of the signal through switches of appropriate size. The switches were sized with binary weight with the largest switch producing the largest delay (since it provides highest fight current). As shown in Fig. 7.18 the delay steps obtained from the CVD elements do not show good linearity and the absolute maximum error after the linear fit goes up to 2.5 ps.

The other popular digital technique used to generate small delays involve digital version of current starved inverter [58]. But the delay steps generated by such scheme can’t be reduced to less than 10 ps in 130 nm technology. The delay generation techniques based on the control of current through the active elements are also vulnerable to the PVT variations.

The capacitor bank based delay lines [59], [36] are less variant to the PVT variation as reported in [59]. Hence, the capacitor bank based architectures for fine delay generation are considered for the design of fine delay cell. Fig. 7.19 shows the schematic of a single cell of capacitor based delay line with CMOS switches of equal widths. Four such elements were cascaded to span the range of a coarse delay cell. Fig. 7.20 shows the linearity of such a structure across various digital settings of the switches. As shown in the figure, the index of non-linearity, i.e. magnitude of residual error after the linear fit, goes up to 1.5 ps.

To improve the linearity of the capacitor based delay lines, a little change to the R-C structure can be considered. Fig. 7.21(a) shows a circuit where the R-C values
across two parallel legs are kept constant. It can be shown that, in this case, the intermediate nodes, $N_1$ and $N_2$ are equipotential. Hence, the same circuit can equivalently be shown as Fig. 7.21(b) where the capacitances and the resistances are connected in parallel and hence the value of any additional capacitor is directly added to the equivalent capacitance. Fig. 7.22 shows the schematic of the implemented capacitor based fine delay cell. The width of the CMOS switches are varied proportional to the capacitor value to maintain the R-C product constant. The capacitors are also made of MOS transistors. Four such elements were cascaded to span the range of a coarse delay cell and the linearity of such a structure was evaluated. Fig. 7.23 shows the linearity of such a structure across input digital settings of the switches. The fine delay chain offers a resolution of about 4 ps in the TT corner. The maximum
Chapter 7. On-chip realization of an all-digital FPDGS

Figure 7.19: Structure of a single capacitor based delay line with constant switch width.

Figure 7.20: Linearity analysis of the capacitor based delay lines with constant switch width.
magnitude of residual error after the linear fit has been kept less than 1 ps. Number of switches in each delay element is limited to 3 to ensure good linearity even with process variation. The impact of process variation is maximum when the setting changes from setting “011” to “100” as a different set of capacitors in the switch are connected with this change. It was calculated that the total delay spanned by the coarse delay unit with the layout parasitics across all corners and with temperature variation from 0°C to 100°C can be covered by cascading five such fine delay units. Even though the digital setting increment within a single element is binary weighted, the setting beyond those are thermometric. E.g. when the first FDC setting reaches “111” it stays “111” for the next higher bit settings and the binary code word for the next FDC is incremented.

![Diagram](image1)

Figure 7.21: Equivalent Circuit of RC based delay lines with product of series R and C kept constant.

![Diagram](image2)

Figure 7.22: Structure of Clock Vernier Devices used in Itanium Processor.
Figure 7.23: Linearity analysis of the Clock Vernier Devices used in Itanium Processor.
Since each fine delay cell contains two inverters, the minimum delay offered by the fine delay unit would scale with the number of fine delay cells used to span the coarse delay cell. This minimum delay offered by the fine delay unit needs to be considered at the system level as the offset varies with temperature.

Fig. 7.24 shows the layout of a fine delay cell. It takes approximately 11 \( \mu \)m X 12\( \mu \)m area. Fig. 7.25 shows a cascade of the 10 coarse and 5 fine delay cells where the input signal is fed from top left corner and the output is taken from the bottom left corner. The pitch of the coarse and fine delay cells are matched so that they can be easily placed and routed by an auto-route tool.

![Figure 7.24: Layout of a single fine delay cell.](image)

![Figure 7.25: Layout of the coarse-fine delay generation unit consisting of 10 coarse delay cell.](image)
7.4 Synchronizer

Apart from the re-timing done in the coarse delay unit to avoid glitches while control bit transition, since the control bits are triggered by the sampling clock and is not synchronously related to the input clock, the control bits are double buffered by the delayed output clock to avoid any chance of going into metastability state while latching the control signals from the controller.

7.5 System Performance with Process and Temperature Variation

The delay generated by the coarse and fine delay cells vary with temperature and process. Hence, the number of fine delay cells to be instantiated in the fine delay unit (FDU) depends on their relative variation across process and temperature variations. In this section the effect of temperature and process on the performance of the coarse fine delay architecture are discussed.

Fig. 7.26 plots the simulated per stage delay of the coarse delay chain along with the dynamic range of the fine delay unit having 5 stages and allowing for 35 different fine delay settings across temperature for various process corners. It can be observed that, the per stage CDC delay can change by an amount of 18 ps due to temperature change from 0 to 95 degree centigrade. With the 5 stage FDU, the dynamic range of the FDU is sufficient to cover for the per stage delay of the coarse delay cell across all temperatures in all process corners. It can also be noted that the slope of the delay curves with temperature for the coarse delay cells and the fine delay unit are different and the variation changes with process corner. Hence, the actual number of fine delay setting to be used to span the delay generated by a coarse delay cell also varies with temperature and process. Table 7.1 lists the number of fine delay cells required to cover one coarse delay cell for different process corners and temperature. It can be seen
that, the value varies from 24 to 30 depending on the process corner and temperature. Hence, to reduce the error due to discontinuity caused while changing the setting for the next coarse delay step, there is a need for on-chip calibration. Similarly, only due to temperature, the number can vary by 2 units and to reduce the error due to temperature variation or any other slow varying parameter, calibration should be done at appropriate instances. The calibration unit has to track the coarse-fine ratio which will in turn be used for ensuring a smooth change in delay while changing the coarse delay setting in the control loop.

Figure 7.26: Per stage delay of the coarse delay unit and the dynamic range of the 5-stage fine delay unit across various temperatures.

Fig. 7.27 plots the dependence of delay of the fine delay unit with setting “0” with temperature. It can be observed that the offset delay added due to addition of each fine delay cell also varies with temperature, which can make the control loop unstable if more fine delay cells are added, because in that case the loop has to track
the variation in the delay bias added by the fine delay unit.

![Graph showing the variation in delay bias](image)

Figure 7.27: Minimum delay from the FDU with setting “0”.

Table 7.1: Number of fine delay cells needed to cover for a coarse delay cell (C/F)

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>SS</th>
<th>TT</th>
<th>FF</th>
<th>SNFP</th>
<th>FNSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>28</td>
<td>25</td>
<td>24</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td>45</td>
<td>29</td>
<td>26</td>
<td>24</td>
<td>27</td>
<td>26</td>
</tr>
<tr>
<td>90</td>
<td>30</td>
<td>27</td>
<td>25</td>
<td>28</td>
<td>26</td>
</tr>
</tbody>
</table>

Since redundant fine delay cells are instantiated in the fine delay unit to ensure
that the fine delay unit is capable of generating delays to cover the delay generated by the coarse delay cell across all process corners, the effect of these redundant delay cells have to be dealt with appropriately. If there are number of fine delays is sufficient enough to cover for the variation in the coarse delay cells due to temperature variation or other slow varying factors, the redundant cells will not have much impact except an increased area. However, in case of medium or low frequency input signal it becomes less economical in terms of area. E.g. for an input period of 10 ns, in 130 nm process, around 100 coarse delay cells are needed in TT corner. With a variation of delay by around 15 ps per cell, around 1.5 ns has to be covered by the fine delay unit which takes around 15 fine delay cells taking an area of around 2000 $\mu m^2$ (This includes the cells needed to cover the variation in the bias shift of the fine delay cells). For even slower signals, the requirement becomes even higher. This area requirement can be minimized to that of 5 fine delay cells if the coarse delay cells are also used along with the fine delay cells even after the loop is closed. In that case, the delay controller has to tap the delayed signal from the next or previous coarse delay cell whenever the fine delay cell setting is zero or reaches its maximum limit. However, because of the redundant fine delay cells, a large jump in the error can be observed which will subsequently die down with the control loop action. To reduce this momentary error at the boundaries there is need of calibration to find out the actual fine delay controller setting to cover one coarse delay on-chip. Fig. 7.28 shows an illustration of the overlapping of the delays from the coarse delay cell and fine delay unit due to the redundant fine delay cells. Ideally, the effective number of fine delay settings per coarse delay cell ($\frac{C}{F}$) should be calculated for each coarse delay cell. But this would be area consuming and it will not be possible to evaluate this number with the control loop in operation. Hence, instead of multiple numbers, a single number $\frac{C}{F}$ is estimated using a replica of coarse and fine delay chain which is activated in the calibration mode. This allows for a dynamic estimation of $\frac{C}{F}$ with a low area, but with an error caused due to the mismatch between the actual coarse delay cells and
the replica delay cells. Based on the montecarlo simulation results, the error in \( \frac{C}{T} \) due to local mismatch can be up to \( \pm \) which can lead to a maximum error of \( \pm 5 \text{ ps} \).

### 7.6 Calibration Unit

Fig. 7.29 shows the circuit realization of the calibration unit. The controller activates the `en_{FDU-cal}` and `cal` signals which route the delayed signal from the calibration unit to the DMU instead of taking it from the actual coarse-fine delay chain. With the `cal` signal, the delay from cascaded 7 coarse delay cells are chosen. In two phases of calibration, the Fine delay chain is activated with setting “0” and “7” successively to find out the dynamic range of the fine delay unit with 7 different settings. During the two phase calibration process, the coarse cell delay is also calculated, which is then used to find out the actual limit to be set in the fine delay chain to reduce the errors caused because of redundant fine delay cells.

![Figure 7.29: Calibration unit to find the coarse-fine ratio to minimize the error during the change of coarse setting.](image)

### 7.7 Controller for Coarse-Fine Architecture

The core of the control loop is the same as that discussed in the previous chapter. But extra signals are generated to generate coarse and fine delay control words and separately activate the coarse and fine delay cells. The state machine generating the coarse and fine delay change activation signals is shown in Fig. 7.30. The actual trigger signal for the state machine is the `estimate-done` signal of the DMU. Initially,
only the coarse delay unit is incremented till a zero crossing of the error is achieved. Instead of setting the initial setting of the coarse delay unit to zero, it is set to a higher value higher such that even in cases where the coarse delay value increases, the target corresponding to the zero coarse setting can be met by reducing it to lower values. For all target values less than this the generated delay will actually be a period more than the actual delay. To reduce the locking time, the delay chains are operated in 4 different modes. Initially the coarse delay cells are only incremented to hunt for the zero crossing of the error. After finding the zero crossing, the coarse delay setting is reverted to the previous delay setting and then the fine delay cells are used to hunt for the zero crossing. During these two modes, the DMU is operated in fast mode as the error value would be high. Once the zero crossing with the fine delay cells is achieved using the fine steps, the DMU is set to operate in slow mode to fetch precise delays. Once the minimum error is obtained in slow mode, the controller is set to be in hysteresis mode where the delay settings are not changed till the error does not exceed a certain threshold. If, due to temperature, the error increases more than the threshold, the controller again hunts for the minimum error point and enters back to the hysteresis mode. The signal $Coarse_{step}$ in Fig. 7.30 is used to trigger for a change in the coarse delay chain based on the sign of the estimated error. The signal $Dir$ is calculated based on the sign bit of the error. The signal $Fine_{step}$ is used to trigger for a change in the fine delay chain based on the sign of the estimated error ($Dir$ signal). Fig. 7.32 shows the simulated behavior of the control loop with the coarse-fine cell architecture and coarse and fine measurement modes. The above control loop works if the fine delay unit can work only if the fine delay unit can provide the change required across all process corners. However, for supporting programmable delays at medium and low frequencies, more coarse delay cells are needed whose delay will vary with temperature. Hence, while larger delays are generated, with a small temperature change results in a larger variation in the generated delay. With limited number of fine delay cells, this error may exceed the range of delay generated by the
fine delay unit. The temperature dependent offset delay added by each fine delay cell also limits the number of fine delay cells that can be added. Hence, there is a need to change the coarse delay setting even after the loop is locked with a particular coarse and fine delay setting.

Fig. 7.31 shows the state machine to handle the situation where the coarse delay setting (CC) is changed even after the loop is locked to the minimum error point. With the signal inc\_CC when set to logic “1, the coarse delay chain is appended by a single coarse delay cell. Similarly the signal dec\_CC when set to logic “1, reduces one coarse delay cell from the chain. Similarly, the signals inc\_FC and dec\_FC are used to increase or decrease the fine delay unit setting to change the delay appropriately. The control loop triggers change in the fine delay unit as long as it remains within a boundary. The lower limit of the boundary is the value zero and the upper limit is dependent on the coarse to fine delay ratio $C/F$. When there is a need to decrease the fine setting while the fine delay setting (FC) is at zero, the coarse setting is decreased to a lower value to a value corresponding to the coarse to fine delay ratio. At this instance, the fresh value of the coarse to fine ratio is calculated from the calibration unit in 3 measurement cycles. Similarly when the fine setting reaches a value corresponding to a threshold value above the coarse to fine delay ratio ($FC_{th}$), the coarse delay setting is incremented and the fine delay setting is loaded to with the setting corresponding to $FC_{th} - \frac{C}{F}$. By adopting such approach, some of the redundant fine delay cells are used and also frequent transition of coarse setting is avoided for the targets falling close to the coarse delay unit boundaries. Since the calibration unit to find the coarse-fine delay ratio is implemented in a replica of the actual coarse-fine delay cells, there can be an error caused because of the local mismatch between them. With the montecarlo simulation results, it can is limited to $\pm 1$. This estimation error can cause a transient offset in the generated delay for a measurement cycle. Ideally, there the $\frac{C}{F}$ values should be calibrated separately for each coarse delay cell as shown in fig. 7.28 to minimize the error due to the use of a
single $\frac{C}{F}$ value for all coarse delay setting changes (due to the local mismatch between the coarse delay cells). But, during the operation of the delay generator system, the cell wise calibration can not be done without halting the system which can lead to error caused due to temperature variation or any other slow varying parameter. Hence using the replica based calibration unit helps in reducing the hardware as well as tracking the slow varying parameters like temperature with a little loss inaccuracy because of the local mismatch.

Figure 7.30: State machine showing activating signal for generating coarse and fine delay steps.
Figure 7.31: State machine showing activating signal for generating coarse and fine delay steps with limited number of fine delay cells.
Chapter 7. On-chip realization of an all-digital FPDGS

Figure 7.32: Simulated error Vs time plot showing the controller action in coarse-fine architecture.
Figure 7.33: Simulation result showing the action of the control loop by changing the coarse and fine delay settings to minimize the error with slow increase or decrease in temperature.
7.8 Measurement results from FPGA implementation

The proposed coarse-fine architecture was tested in a prototype involving FPGA and a commercial delay generation chip. The coarse delay unit along with all the other controller blocks are implemented inside the FPGA as shown in Fig. 7.34, where the blocks shaded are implemented inside the FPGA. The LUTs (Look-up tables) in the FPGA are used as the coarse delay cells. The fine delay unit was implemented using a commercial delay generation chip CDCF5801A which generated programmable delays in fine steps from the signal fed from the coarse delay unit’s output. Each of the important components in the setup are marked as “A”, “B”, “C” and “D” respectively. The elements “B” and “D” are coaxial cables as the fine delay unit is implemented outside the FPGA and the good quality coaxial cables are used to connect the output of the coarse delay unit to the commercial fine delay step generation unit. Another coaxial cable (marked as “D”) is used to connect the delayed output from the fine delay generation block back to the FPGA so that the delay at the destination point inside the FPGA can be estimated. In our setup, a delay of about 8 ns is caused by the components marked as “B” and “D”.

7.8.1 Coarse delay unit

The coarse delay unit was implemented by using the resources inside the FPGA in the form of the basic units of an FPGA, i.e. the LUTs. Hence, the delay values that can be achieved by using the LUTs was checked by cascading 128 LUTs and measuring the delay using the internal delay measurement unit implemented inside FPGA. Fig. 7.35 plots the measured error across the number of activated coarse delay cells. The input signal was of period 32 ns and hence there delay values fall to zero just after the value of 32 ns. It can also be noted that the delay corresponding to the zero value of the coarse delay unit, (i.e. for a case where only one coarse delay cell is
Figure 7.34: Setup used for evaluating coarse-fine architecture. The blocks inside the shaded region are synthesized in a Virtex II Pro FPGA. The Fine delay unit is implemented using a commercial delay generation chip.

activated) the generated delay is close to 25 ns. This larger value of delay is because the fine delay unit is implemented externally. The steps vary from 100 ps to 1.3 ns and the period of 32 ns is covered by approximately 45 coarse delay cells. This large variation is because in the cases where the LUTs are in the same slices, the delay added is less where as it is more in cases they are instantiated across multiple slices as shown in 7.36.

The control loop was implemented inside the FPGA using only the coarse delay unit as the controlled entity. Fig. 7.37 shows the error from that implementation for 64 different targets set at equal timing intervals between 0 ns and 32 s. It can be observed that the error goes up to 600 ps on either side due to nonuniformity in the coarse delay steps.
Figure 7.35: Linearity plot of the coarse delay chain implemented using the LUTs inside FPGA.

Figure 7.36: Look up table (LUT) and slice hierarchy inside FPGA.
Chapter 7. On-chip realization of an all-digital FPDGS

Figure 7.37: Error in the closed loop case with only the coarse delay unit activated.

7.8.2 Fine delay unit

The blocked marked as “C” in Fig. 7.34 is the fine delay unit implemented by using a prototype board made of a commercial programmable delay generator chip (Part number CDCF5801A from TI [54]). The signal coming out from the coarse delay unit from FPGA is routed as an input to fine delay generation board and the delayed signal from the chip is fed back to the FPGA. The delays between the signal routed from FPGA to delay chip and imported back from the delay chip to FPGA are measured inside FPGA.

Control signals are also connected between the FPGA and the delay control chip. Because of the external signal routing of the input clock signal and specifications of the delay chip for obtaining the finest steps, the frequency of operation was limited to within 50 MHz. Beyond that frequency, the shape of the high frequency signals deteriorate impacting the precision. Even though the proposed architecture can support any low frequency clock signal whose period can be covered by the total coarse delay that can be generated within FPGA through LUTs, the lower limit is imposed
by the commercial delay chip used (25 MHz). From 25 MHz to 75 MHz, the chip can provide delays at a step of $\frac{1}{3072}$ of the input clock period. Beyond 75 MHz, the resolution in terms of UI becomes $\frac{1}{1536}$ till 156 MHz. The fine delay generator chip is controlled by injecting a rising edge at the “DLYCTRL” pin with the “LEADLAG” pin setting the direction of the change.

### 7.8.3 Controller

The controller shown in Fig. 7.30 is adopted in the actual implementation as there was a large variation of the step size of the CDCs which is unusual in an ASIC implementation. Since the fine delay unit was capable of generating fine delays which can cover the variation of the coarse delay cells there was no need to use the coarse to fine calibration ratio and only the fine delay cell was used to track the error to the minimum value. Still to verify the effectiveness of the proposed coarse-fine architecture in reducing the number of fine delay cells, the maximum number of fine delay cells was limited to 200 and the number of coarse delay cells to 40. The control loop was then activated to lock to minimum error point for 64 equi-distant target delays for a period of 37 ns and the error for each target was plotted.

### 7.8.4 Result

Fig. 7.38 plots the measured error from the coarse-fine architecture implemented inside FPGA. It can be noted that the error is limited to within 10 ps in most cases for an input clock period of 32 ns. The maximum error observed is 25 ps.

### 7.9 Conclusion

In this chapter, we looked at the ASIC implementation issues for the design of fractional period delay synthesizer unit capable of generating delays of the order of few nano seconds. Circuit techniques to reduce the total area by adopting a coarse-fine
architecture and the issues arising because of such architecture are discussed along with possible solutions. Improvements in the coarse and fine delay cell architectures are proposed to achieve low power as well as better linearity. Results from an FPGA implementation also validate the effectiveness of the proposed architecture in achieving lesser area by reducing the number of fine delay cells.

Figure 7.38: Measured result from the coarse-fine architecture implemented with FPGA LUT delays and the commercial delay generation chip having nominal step size 10 ps.
Chapter 8

Conclusion

With technology scaling, the MOS transistors have become much faster as well as more prone to variations. Hence, recently time based signal processing has gained popularity as it takes advantage of the faster transistors. This thesis is an effort towards measuring clock skews precisely with very minimal hardware resources. A system level application of the proposed clock skew measurement technique to generate accurate programmable fractional period delays is also demonstrated.

8.1 Contributions of the Thesis

This thesis work shows that asynchronous sub-sampling followed by statistical averaging allows measurement of static skews between periodic signals in an area efficient manner. The proposed techniques of de-bouncing followed by averaging of the arithmetic difference of the signals (histogram of arithmetic difference) remove any dependency of resolution with sampling clock jitter, unlike in previous works. Measured results from a 65 nm test chip indicate the ability to measure skews with a $\pm 3\sigma$ resolution of 0.84 ps and integral error of 0.65 ps for an input skew range of 1 FO4 delay ($\pm 20$ps). The technique can also be used to measure larger skews even close to $\pm \frac{T}{2}$, where T is the clock period. The precision is unaffected by clock jitter.
as a measurement resolution of 0.84 ps is obtained with clock sources with 30 ps rms jitter. This is further validated by experiments where frequency modulation on sampling clock preserves the resolution. In fact, in certain cases where the sampling clock is rationally related to core clock, frequency modulation improves resolution, which is otherwise degraded.

Three potential applications of the proposed technique were shown to evaluate on-chip variability. In the first application, the mismatch between the sampling instances between multiple samplers in terms of time is evaluated. The second application shows the effectiveness of the technique in measuring individual gate delay. The method to use the technique to potentially reduce the hardware resources for measuring clock skew from multiple test nodes is also proposed.

A system level application of the proposed technique to measure delay in generating an accurate programmable delay is also discussed. The delay measurement unit (DMU), capable of measuring delays accurately for the full period range is used as the feedback element to build accurate fractional period delays based on input digital code word. The proposed closed loop control for a delay generation system allows much better accuracy than the open loop case to generate arbitrary fractional unit interval delays. The elements constituting the system need to be carefully designed to enable stability and precision of delay generated across full range. Asynchronous sub-sampling followed by statistical averaging allows accurate and precise measurement of static skews between periodic signals. The proposed system periodically measures and corrects the error to keep it at the minimum and does not require any special calibration mode for error correction. Therefore, the system can run without interruption for a long time with minimum error even if the slow varying parameters like temperature vary with time. Up to 40X improvement in accuracy is measured by enabling the feedback control. Some of the numbers e.g. frequency of operation, jitter at the output etc can be much better if all the components are built on-chip. Since the loop control is slow, the input jitter directly propagates to output. An all digital
Chapter 8. Conclusion

delay synthesizer capable of generating fractional period delays based on coarse-fine
architecture is also discussed. The architecture helps in generating any fractional
period delay for a wider input frequency range with very less area. The proposed
coarse and fine delay cells help in achieving better linearity and power savings. The
proposed control loop along with the calibration unit for handling the discontinuities
between the coarse and fine jumps in delays ensures the error to be within limit.
Measured error from an FPGA implementation of coarse-fine architecture shows the
error to be within 25 ps with a nominal delay step size of 10 ps.

8.2 Future Scope

Although some amount of theoretical framework has been used to explain the effec-
tiveness of the subsampling based system to measure delay, there is a scope for deeper
analysis.

The sampling clock in the prototypes used to validate the proposed system is pro-
vided from outside. A monolithic solution of the system with the sampling system
generated internally from the input clock can be designed to avoid a need of another
clock source. As in that case the beat frequency and the input frequencies are ra-
tionally related, artificial jitter can help in getting better resolution as illustrated in
chapter-4.

Although the programmable fractional delay generation system is validated with a
prototype involving a commercial delay generation chip, the technique can be directly
ported into ASIC. Some of the issues to be addressed while making an on-chip all-
digital programmable fractional period delay generation system and those ideas can
be validated by making an ASIC with the technique and basic cells proposed in
chapter-6.

Apart from a delay generation system, there are many potential applications where
there is a need for delay measurement between two periodic signals and in all those
systems the proposed technique of skew estimation can be used. Some examples of such applications are, time-to-digital converter based ADCs, serial interfaces etc.
Appendix A

Analytical overview for the Skew Estimated by the DMU

In this appendix, an analytical overview of the delay measurement unit is provided. It is also shown that the proposed histogram of arithmetic difference leads to an unbiased estimator of skew. The derivations for equations (3.1) in chapter 3 are also sketched.

When the sampling clock is asynchronous to the clock driving the measurement nodes, \( \frac{T}{\Delta T} \) is an irrational number and hence can be written as \( T = N\Delta T + \alpha \) where \( N \) is an integer and \( 0 < \alpha < \Delta T \). This causes the sampling edge to fall uniformly across the entire period of the sampled signal. Hence, the percentage of time the sampling edge falls in between the sampled edges is directly proportional to the skew as a fraction of the period.

Let \( T_1, T_2 \) be the times, within a clock period when \( d_1, d_2 \) cross the logic high threshold and let \( T_s \) be the time when the sampling clock crosses the sampling threshold. Due to jitter, these are random variables. The average of the arithmetic difference between the sampler outputs is used as an estimate for the skew:

\[
S = \frac{1}{2^k} \sum_{i=1}^{2^k} X_i
\]

(A.1)
Appendix A. Analytical overview for the Skew Estimated by the DMU

where $X_i = q_1^i - q_2^i$, the arithmetic difference between the $i^{th}$ sampler outputs. It is shown later in this Appendix that:

$$E[S] = \frac{\delta}{T} \quad (A.2)$$

Thus, the delay measurement statistic is an unbiased estimator of the skew as a fraction of the clock period (UI).

A theoretical loose upper bound for the standard deviation of the estimate is derived in this Appendix is:

$$\sigma_s \leq \frac{1}{\sqrt{2^k+1}} \quad (A.3)$$

**Derivation**

Let $T_1, T_2$ be the times within a clock period when data clocks $d_1, d_2$ cross the logic high threshold respectively, and let $T_s$ be the time when the sampling clock crosses the sampling threshold. Due to jitter, these are random variables. Without loss of generality, let the mean of $T_1$ be zero. The mean of $T_2$ is $\delta$, the quantity to be estimated.

Let

$$\tilde{T}_2 = T_2 - \delta$$

and let

$$T_s = t_s + \tilde{T}_s \quad (A.4)$$

where $t_s$ is the mean value of $T_s$, and $\tilde{T}_s$ is the random component.

It is of interest to determine the probability that the samplers sample a logic high. A sampler samples a logic high if the sampled clock edge occurs earlier than the sampling clock edge. Hence,

$$P(q_1 = 1) = P(T_1 < T_s) = P(T_1 - \tilde{T}_s < t_s) \quad (A.5)$$
Appendix A. Analytical overview for the Skew Estimated by the DMU

Let \( Z_1 = T_1 - \tilde{T}_s \). Let \( \Phi_1(\cdot) \) be the CDF of \( Z_1 \). From equation (A.5),

\[
P(q_1 = 1) = P(Z_1 < t_s) = \Phi_1(t_s)
\]  (A.6)

Let \( Z_2 = \tilde{T}_2 - \tilde{T}_s \). Let \( \Phi_2(\cdot) \) be the CDF of \( Z_2 \). Then,

\[
P(q_2 = 1) = P(T_2 < T_s)
= P(\tilde{T}_2 + \delta < \tilde{T}_s + t_s)
= P(Z_2 < t_s - \delta)
= \Phi_2(t_s - \delta)
\]  (A.7)

The output of the delay measurement unit of Fig. 3.11 is given as

\[
S = \frac{1}{2^k} \sum_{i=1}^{2^k} X_i
\]  (A.8)

with \( X_i = q_i^1 - q_i^2 \), the difference of the i\textsuperscript{th} samples. It follows that

\[
\mathbb{E}[X_i] = \Phi_1(t_s^i) - \Phi_2(t_s^i - \delta)
\]  (A.9)

where \( t_s^i \) is the i\textsuperscript{th} sampling instant.

Let the clock period be \( T \) and the sampling clock period be \( T + \Delta T \), where \( T = N \Delta T + \alpha \), where \( N \) is an integer and \( 0 < \alpha < \Delta T \). This causes the sampling edge to fall uniformly across the entire period of the sampled signal to create one beat period. Let the measurement be taken over \( M \) beat periods, so \( MN = 2^k \). Hence, (A.8) can be rewritten as:

\[
S = \frac{1}{MN} \sum_j \sum_k X_{jk}
\]  (A.10)
Appendix A. Analytical overview for the Skew Estimated by the DMU

Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_M)$ be the starting phases in each beat period. Then

$$E[S|\alpha] = \frac{1}{MN} \sum_j \sum_k E[X_{jk}(\alpha_j + k\Delta T)] \quad (A.11)$$

Substituting from (A.9), applying the law of iterated expectation and reordering the summation, we get

$$E[S] = E[E[S|\alpha]] = \frac{1}{N} \sum_k \frac{1}{M} \sum_j E[\Phi_1(\alpha_j + k\Delta T)$$

$$- \Phi_2(\alpha_j + k\Delta T - \delta)] \quad (A.12)$$

Since $\alpha_j$s are uniform over 0 to $\Delta T$, (with PDF of $\frac{1}{\Delta T}$), the inner expectation is identical for each $j$ and can be evaluated as the following integral:

$$E[S] = \frac{1}{N} \sum_k \frac{1}{\Delta T} \int_{k\Delta T}^{(k+1)\Delta T} \Phi_1(t) - \Phi_2(t - \delta) \, dt \quad (A.13)$$

The above summation can be replaced by an integral over the entire clock period $T$. However, if we assume that the skew $\delta$ and the jitter of the clocks are small compared to the period $T$, then the limits of the integration can be replaced by $\pm \infty$ as

$$E[S] = \frac{1}{T} \int_{-\infty}^{\infty} \Phi_1(t) - \Phi_2(t - \delta) \, dt \quad (A.14)$$

In general, evaluating this integral is difficult. However, in this particular case, we can revert to the following trick of differentiating Equation (A.14) with respect to $\delta$:

$$\frac{dE[S]}{d\delta} = \frac{1}{T} \int_{-\infty}^{\infty} \Phi'_2(t - \delta) \, dt = \frac{1}{T} \quad (A.15)$$

Since the term inside the integral is a PDF and integrates to unity, the fact that the histogram of arithmetic difference leads to an unbiased estimator of skew is proved.

$$E[S] = \frac{\delta}{T}$$
Appendix A. Analytical overview for the Skew Estimated by the DMU

The variance of $X_i$ can be bounded as:

$$VAR(X_i) = P(q_1^i = 0)P(q_2^i = 1) + P(q_1^i = 1)P(q_2^i = 0) < 1/2 \quad (A.16)$$

From equations (A.8) and (A.16),

$$VAR(S) = \sigma_s^2 \leq \frac{1}{2^k} \cdot \frac{1}{2} \quad (A.17)$$

from which the bound on the standard deviation of S given in Equation (A.3) follows.

$$\sigma_s \leq \frac{1}{\sqrt{2^k + 1}}$$
Appendix B

Derivation steps for Variance
Calculation of buffer delays in a loop

In this Appendix, the detailed derivation for calculating the variance of individual buffer delays ($D_i$) is provided.

Derivation

From equation (6.5) of Chapter 6,

$$D_i = \frac{i}{N} D_N - \frac{i}{N} \sum_{j=1}^{N} \delta T_j + \sum_{j=1}^{i} \delta T_j = \frac{i}{N} D_N + \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=1}^{N} \delta T_j$$  \hspace{1cm} (B.1)

Variance of $D_i$ can be calculated as

$$Var[D_i] = Var \left[ \frac{i}{N} D_N + \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=1}^{N} \delta T_j \right]$$  \hspace{1cm} (B.2)
Appendix B. Derivation steps for Variance Calculation of buffer delays in a loop

\[ \begin{align*}
&= \mathbb{E} \left[ \left( \frac{i}{N} D_N + \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=1}^{N} \delta T_j \right)^2 \right] \\
&\quad - \left( \mathbb{E} \left[ \frac{i}{N} D_N + \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=1}^{N} \delta T_j \right] \right)^2 \\
\end{align*} \] (B.3)

Since \( \mathbb{E}[\delta T_j] = 0, \ \forall \ 1 \leq j \leq N, \)

\[ \begin{align*}
&= \mathbb{E} \left[ \frac{i}{N} D_N + \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=1}^{N} \delta T_j \right] = \frac{i}{N} D_N. \\
\end{align*} \] (B.4)

Similarly,

\[ \begin{align*}
&= (\frac{i}{N} D_N)^2 + 2 * \frac{i}{N} D_N * \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=1}^{N} \delta T_j \right) \right] \\
&\quad + \left( \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=1}^{N} \delta T_j \right) \right] \right)^2 \\
&= (\frac{i}{N} D_N)^2 + \left( \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=1}^{N} \delta T_j \right) \right] \right)^2 \\
\end{align*} \] (B.5)

Substituting the above two equations in Eqn (B.3) we get,

\[ \begin{align*}
Var[D_i] \\
&= \left( \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j - \frac{i}{N} \sum_{j=1}^{N} \delta T_j \right) \right] \right)^2 \\
&= \left( \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j \right) \right] \right)^2 + (\frac{i}{N})^2 \left( \mathbb{E} \left[ \sum_{j=1}^{N} \delta T_j \right] \right)^2 \\
&\quad - \frac{2i}{N} \left( \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j * \sum_{j=1}^{N} \delta T_j \right) \right] \right) \\
&= \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j * \sum_{j=1}^{N} \delta T_j \right) \right] \\
&\quad \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j * \sum_{j=1}^{i} \delta T_j + \left( \sum_{j=1}^{i} \delta T_j * \sum_{j=i+1}^{N} \delta T_j \right) \right] \right) \\
\end{align*} \] (B.6)

\[ \begin{align*}
&= \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j * \sum_{j=1}^{i} \delta T_j \right) \right] \\
&= \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j * \sum_{j=1}^{i} \delta T_j \right) \right] + \\
&\quad \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j * \sum_{j=i+1}^{N} \delta T_j \right) \right] \\
\end{align*} \] (B.7)
Appendix B. Derivation steps for Variance Calculation of buffer delays in a loop

\[ \Delta T_i \]s are independent, \( \forall j \neq k \), \( \mathbb{E}[\delta T_j \ast \delta T_k] = 0 \) \( \forall j \neq k \)

\[ \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j \ast \sum_{j=i+1}^{N} \delta T_j \right) \right] = 0 \]

Hence, Equation (B.7) becomes:

\[ \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j \ast \sum_{j=1}^{N} \delta T_j \right) \right] = \mathbb{E} \left[ \left( \sum_{j=1}^{i} \delta T_j \right)^2 \right] \quad (B.8) \]

Putting this value in Equation (B.6), it becomes:

\[ Var[D_i] = i \ast Var[\delta T_j] + \frac{i^2}{N} \ast Var[\delta T_j] - \frac{2i^2}{N} \ast Var[\delta T_j] \]

\[ \Rightarrow Var[D_i] = \frac{i \ast (N - i)}{N} Var[\delta T] \quad (B.9) \]
Bibliography


