Dual reference signal post-silicon reconfigurable clock distribution networks

Atanu Chattopadhyay

Department of Electrical and Computer Engineering

McGill University, Montreal

February 2009

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Copyright © Atanu Chattopadhyay, 2009

Abstract

This thesis investigates the use of averaging techniques in the development of clock distribution networks and an on-chip clock skew measurement circuit. Our flexible clock distribution network can be used in both single clock and multiple clock integrated circuit applications. The design moves away from clock trees, using a pair of reference clocks traveling in opposite directions to perform clock synchronization on a *daisy-chained* (serial) clock distribution line. By synchronizing each local clock edge to a position directly in between the forward and reverse reference clock edges, we demonstrate that sub-10 ps variance in clock arrival times can be achieved between local clocks. The design provides a scalable and simple-to-layout solution with multi-point skew compensation useful for large designs. The system provides the benefits of a closed-loop clock de-skewing solution by compensating for process, temperature and power supply variations, with the power savings of an open-loop solution at run-time.

Our technique allows routing switches to be included in the clock path, permitting the post-silicon re-sizing and re-shaping of clock domains. Localized clock switches or a complete chip-wide switch mesh can be used to re-route clock signals – a capability that is impossible without our daisy-chained clock network. We investigate a clock network that emphasizes flexibility and reconfigurability without sacrificing tolerance to clock skew. We show that this approach is realizable with transistor-level schematic and extracted circuit structures in TSMC's 180 nm standard process. We also develop a

modeling infrastructure from which we can create a variety of clock network configurations and synthesizable clock network controllers for arbitrary applications using ModelSim and Quartus II.

An on-chip clock skew management system to detect and potentially correct clock skew between selected points on an IC is also investigated. Our system, BICSS, aids in the debugging of timing errors that may be discovered during testing due to the added visibility of the on-chip clock signals and can repair otherwise defective dies using high-resolution delay lines in the clock path. BICSS is unique in its ability to detect, measure and compensate for clock skew using a single all-in-one solution.

Abrégé

Cette thèse étudie une technique de moyennes pour créer un système de distribution d'horloge et un circuit pour mesurer le désalignement de phase d'horloge sur circuit intégré. Notre circuit de distribution d'horloge est polyvalent et peut être employé pour les systèmes avec une horloge simple ou des horloges multiples. La conception s'éloigne des circuits de distribution par arbres, utilisant une paire de signaux de référence voyageant en directions opposées pour corriger le déphasage de chaque horloge répartie linéairement sur la puce. En synchronisant chaque front ascendant d'horloge locale directement entre ceux des signaux de référence, on démontre que le déphasage peut-être réduit en dessous de 10 picosecondes. La conception permet une distribution d'horloges qui est simple à appliquer et extensible. Cette démarche corrige les variations de processus, d'alimentation et de température, fournissant la correction du désalignement de phase systématique de chaque tranche de la distribution d'horloge.

Notre technique permet l'introduction des commutateurs de cheminement pour commuter les trajets d'horloge et changer la grandeur et forme des domaines d'horloge après la fabrication d'un circuit intégré. Des commutateurs localisés ou un réseau de commutateur en maille qui couvre le circuit au complet peuvent être utilisés – deux possibilités qui sont impossibles sans notre réseau connecté en série. Notre recherche souligne la flexibilité et la reconfiguration dynamique d'un réseau d'horloge sans sacrifier l’alignement de phase des signaux d'horloges locaux. Nous prouvons que cette

approche est réalisable avec des conceptions niveau schémas et niveau circuits-extraites utilisant le processus de fabrication de 180 nanomètre de TSMC. Nous avons également conçu avec ModelSim et Quartus II un modèle pour étudier des diverses configurations de réseaux d'horloges et pour créer des contrôleurs réalisables.

Un système sur circuit intégré qui peut détecter, mesurer et corriger la différence de phase entre les horloges d'endroits présélectionnés dans le circuit est aussi conçu. Notre système, BICSS, peut réparer les puces autrement défectueuses utilisant des lignes à retard à haute résolution. BICSS peut aider à éliminer des erreurs de synchronisation qui peuvent être difficilement découvertes pendant l'essai grâce à la visibilité supplémentaire que le système permet.

Acknowledgements

I would first like to thank my parents for helping me, supporting me and being there for me. I would also like to thank my sister and her kids for providing me with an easy and enjoyable excuse to get away from things in California.

I would next like to thank my supervisor Dr. Zeljko Zilic. He provided me insight and direction while giving me all the freedom I needed to explore my ideas as far as I could take them. I would also like to thank Dr. Frank Ferrie for teaching me the ins and outs of running a class.

Next, I would like to acknowledge the financial and material support of Altera Corporation, the Fonds québécois de la recherche sur la nature et les technologies (FQRNT), CMC Microsystems and McGill University.

I also appreciate the advice and comradery of all of the graduate students, past and present, in the department. Particularly, Marc Boulé, Henry Chan, Jean-Samuel Chenard, Nathaniel Azuelos, Bojan Mihajlovic and Mona Safi-Harb. I can't imagine how quiet things would have been without you guys around to talk to and brainstorm with.

Finally, I would like to thank all my friends: Mario, Peter, Nadia, Catherine, Vishal, all the Marks, Andras, Jordanna, Eric, Geoff and Neil. Time flies by so quickly, it is hard to fathom how long we have all been friends.

I feel lucky that I have had whatever I needed, whenever I needed it. I don’t think I could have come this far without a little something that each of you has provided.

Table of contents

Table of figures

Table of tables

Chapter 1:

Introduction

"Much may be done in those little shreds and patches of time which every day produces, and which most men throw away."

- Charles Caleb Colton [1]

Coordinating activity in a digital integrated circuit is a fundamental problem in circuit design. Modern integrated circuits (ICs) use a *clock* as a common synchronization signal to coordinate events within the device's datapath. Among their many uses, clocks can be used by functional blocks to indicate when the data it requires is ready to be processed, to divide a task into smaller ones requiring more processing cycles but

permitting faster execution of each individual step, or to avoid *signal races* by preventing data from being used before it is stable. Traditionally, high performance circuits are designed to have the highest clock frequency possible, but with newer technologies, more emphasis has been placed on the amount of *work* or operations that can be done in a given amount of time. However, as clock frequencies continue to increase due to newer integrated circuit process technologies, the percentage of unused time in each clock period continues to increase largely due to timing uncertainty in the circuitry designed to transport clock signals to each circuit block. Any uncertainty in clock arrival times must be reserved within a clock period to ensure the correct operation of the device. Since no computation can take place during this reserved time, it is important to minimize the amount of unused computation time, making the *clock distribution network* (CDN) in an IC among the most important components in any synchronous digital system due to its direct effect on circuit performance and functionality.

Clock networks must broadcast a clock signal throughout an IC minimizing clock uncertainty, consuming as little power as possible and providing consistent signal characteristics such as rise time and duty cycle. Further, the system must be robust and easy to implement. Currently, the most common design approach for clock networks relies on automated clock tree synthesis methods that are tailored to a fixed silicon implementation with a rigid clock network using user-specified parameterization. The outcome is a fixed layout clock tree that is difficult to modify once created, often requiring complete re-implementation of the network for every iteration of the design. Once fabricated, absence of flexibility in the clock distribution poses a problem on two major

fronts: first, errors or process variation in the silicon cannot be corrected; second, adapting a design to a variety of configurations and applications become limited by the lack of flexibility in the clock network. This is especially true with the ever-increasing shift to flexible design methodologies that use programmable devices for reconfigurable computing applications.

1.1. Problem description

With the decreased cost and increased availability of silicon area, ICs have become significantly more complex in recent years. With the increase in popularity of "system-on-chip" designs, a single modern IC is capable of doing the job of many of its predecessors and the potential versatility of a system today is unparalleled. However, it is difficult for today's clock distribution networks to cope with flexible, reconfigurable and multi-application integrated circuits due primarily to the fixed tree-based methodologies used to minimize clock arrival time variation, or *clock skew*. With modular design strategies, circuit components may derive from many distinct sources such as intellectual property (IP) blocks [2] with little knowledge concerning the internal circuit characteristics available to IC architects. Field-programmable gate arrays (FPGAs) allow flexible logic and clock domains and avoid the upfront design costs of application specific designs, but when compared to the application specific integrated circuits (ASICs), they require larger devices, consume more power and cannot achieve the same performance. In addition, to create flexibility in their clock networks, FPGAs must allow for more significant variation in

clock signal arrival times, which is unexploited computation time. Balancing flexibility and low skew can pose a significant challenge to clock networks and has not been extensively explored in existing designs.

Today, an ever increasing amount of the ASIC design flow is spent on testing and verification challenges due to the complexity of the problem: every 10% increase in design complexity increases the test problem 100% [3]. This figure becomes even more significant when one considers that a typical design flow places 70% of the effort on verification, compared to 30% for design [4]. According to Collett International Research, the first iteration success rate for ICs had decreased from 48% to 34% between 2000 and 2003 due to the added complexity of designs [5]. Approximately 45% of devices fail based on logic errors, 33% of devices fail due to fast or slow signal paths and 10% of devices fail due to errors in the clock network. It then makes sense to take advantage of the low cost of silicon area to consolidate multiple designs into one, validate them together, effectively distributing development costs over multiple product lines. This *silicon reuse* application is similar to *platform-based* design methodologies and allows personalization of a single device for a number of possibly unrelated applications. ICs with defective sections can be binned to applications that do not require the defective component. Some small changes can be performed after silicon implementation to existing clock networks with simple “pruning” or removing of clock branches, but the modifications required will usually alter the loading of the clock or the frequency required, which necessitates the creation of a new CDN. Instead, when a single silicon design is

used for multiple applications through the use reprogrammable components and processors, the availability of flexible clock distribution networks is useful.

1.2. Thesis objectives

The goal of this doctoral project was to create a flexible clock distribution network that could be used in ASICs, FPGAs or other applications without sacrificing the low clock skew performance achievable with today's clock networks. The desired result is a clock network where every region can be connected to and disconnected from one-of-many clock regions post-silicon, introducing significant flexibility to the clock distribution network. In FPGAs, the benefit of flexibility in the clock network could add additional functionality to current designs. In ASIC applications, the goal is to create a single design that can be reconfigured dynamically for different tasks or programmed at the factory for different applications. Consider the implementation of a single processor design with multiple components such as Bluetooth, USB, Firewire, JTAG, floating point arithmetic units and video encoders. The idea was to not only create blocks which could be included and discarded, but potentially shared between multiple clock domains, all while maintaining adequate skew compensation for all the leaves in a clock domain. Such a configuration would also allow *micro-clock regions,* where every component could operate at or near their minimum frequency considering its operating requirements resulting in a decreased average clock frequency for the design - a helpful trait in minimizing overall energy consumption [6].

One challenge in creating such a system is moving away from clock trees and finding other approaches to distributing a clock signal on an IC: one that allows routing switches to be included in the clock path to connect clock subregions together easily and arbitrarily. A switch mesh can be used to re-route clock signals, but this method requires a *daisy-chained* approach that will create skew between every clock region and will call for a complete change in the approach used to minimize skew in clock networks. No longer would balancing global clock trees be sufficient. We want to first develop a methodology to make serially-distributed clock domains possible, and next want to show that this approach was realizable with transistor-level circuit structures and we finally want to create a modelling infrastructure from which we could create a variety of clock network configurations for arbitrary applications. We explore both single clock and multiple clock reconfigurable systems. This is the first investigation of a clock network that emphasizes flexibility and reconfigurability without sacrificing tolerance to clock skew. Finally, given that process variance can occur between ICs (inter-die) and across a single IC (intra-die) for any clock distribution network, we want to look at a low-cost method to determine the quality of a clock network using an on-chip skew measurement approach known as BICSS. BICSS provides post-silicon debug capability of clock distribution networks that can aid in diagnosing timing errors that may be discovered during testing. Both BICSS and our serial dual reference signal clock distribution networks can improve yield by providing post-silicon repair capability to the clock networks of integrated circuits suffering from certain timing errors.

1.3. Statement of original contributions

1.3.1. Single clock averaging network

We present a single clock distribution scheme using a dual reference signal approach and a single bi-directional conductor between clock taps. We use an *averaging* technique to allow serialization of clock networks. Daisy-chaining the clock decreases the clock interconnect load by eliminating the redundant paths used to equalize delays in traditional H-tree distributions. Clock skew is accounted for by actively synchronizing each local clock to a position directly between forward and reverse-moving reference clocks. The design provides simple-to-layout and scalable multi-point skew compensation useful for large designs. Our design is unique in its use of a truly bi-directional line and buffering in an averaging distribution. This approach compensates for process, temperature and power supply variations, eliminating systematic skew in the clock distribution network. In addition, a dual reference line averaging approach is explored to maximize the system's tolerance to device mismatch. Device mismatch is a key contributor to clock skew in traditional clock distribution networks due to the large distances between ideally-matched devices. We show that our serial clock distribution is much better able to cope with mismatch since the distances between ideally-matched devices in greatly minimized in our system. Tree-based clock distribution networks are also susceptible to skew from cross-chip temperature variation due to the distributed buffers that they employ. We also explore a serial clock distribution that is highly tolerant

to intra-die temperature changes. This work was initially disclosed in 2006 [7]. Additional configurations for performing the averaging of the forward and reverse reference signals were first disclosed in 2007 [8][1].

1.3.2. Multiple clock reconfigurable clock network

We present a multiple clock distribution scheme by expanding our single clock design, adding clock routing structures into an averaging clock network to enable re-shaping clock domains, post-silicon. These reconfigurable, reprogrammable clock networks can be used in ASICs, SoCs and FPGAs. The technique is useful for reconfigurable computing applications to connect the programmable logic to the clock domains of the surrounding logic. The proposed design allows for more flexibility in clock networks than current designs such as those used in Altera and Xilinx FPGAs. It simplifies layout for irregularly-shaped clock domains and provides flexibility to designers by enabling post-fabrication changes to the clock distribution network. This work was initially disclosed in 2007 [9].

[1] While the other listed disclosures were in peer-reviewed publications, the TEXPO poster and extended abstract submission were by open invitation.

1.3.3. On-chip clock skew detection circuitry

To verify the characteristics of arbitrary clock networks, we present a Built-In Clock Skew System (BICSS) that uses a centralized approach to identify, quantify and correct skew using a two-step method. The first step is to assess the time-of-flight between the central debug circuitry and each region, or tap under test to account for the measurement error due to differences in path length. This measurement error is common in existing techniques. Typically, timing errors due to faulty clock networks can be very difficult to diagnose due to the intermittent nature of the errors that they may cause. The result is a scalable solution that provides silicon debug and repair capability of on-chip clock skews. This work was disclosed at the 2008 Design, Automation and Test in Europe conference [10].

1.3.4. Custom circuitry

We present the schematic and extracted designs of the required custom circuit components in a 180 nm standard process. Of particular interest are the phase detectors and delay lines that we explored. The phase detectors are unique due to their use in an all-digital application with finite skew bound. This application allows the phase detector to create and to use a dead-zone period to speed up resolution time. For the delay lines, different configurations are explored to emphasize delay matching through multiple delay line pairs, signal characteristics and duty cycle retention. A current-starved all-digital approach with good linearity between settings and zero static current consumption is

chosen. The circuits involved in creating our clock distribution networks were primarily disclosed in 2007 [11], and partially disclosed in [7]-[10].

1.4. Thesis organization

Chapter 2 describes background on clock networks and some de-skew approaches, Chapter 3 describes implementation of a single clock domain network using our method, Chapter 4 describes a reconfigurable multiple clock network using a clock mesh and our averaging approach, Chapter 5 describes an on-chip approach to detecting, measuring and compensating for clock skew in an integrated circuit environment, Chapter 6 describes a system-level overview of these components including HDL models and the required synchronization controllers, and Chapter 7 highlights some of the circuitry required to implement these systems in a TSMC 180 nm technology.

Chapter 2:

Background

2.1. Introduction

This chapter outlines the conventions and definitions that we have adopted. It also describes some of the existing work and techniques in the area of clock distribution and highlights some of the benefits and drawback of each. The information is compiled from sources that include [12]-[16]. Other sources are used as indicated.

2.2. Clock characteristics

The clock is a periodic signal, usually viewed as having the shape of a sine or square wave used to synchronize two or more events at different locations using a single signal in a synchronous circuit. While clocks are used in other applications such as in printed circuit board designs, we limit our discussion here to clocks used on integrated circuits such as microprocessors, field-programmable gate arrays and application specific integrated circuits. Typical high performance analog circuits use sinusoidal clocks due to the large number of high frequency harmonics present the near infinite slope of a square wave's edge. Some circuits trigger only on the rising or the falling edge of the clock (single clock edged devices) and some trigger events on both rising and falling edges (dual clock edged devices). We will use the term *multiple clock circuit* to describe a circuit which has a plurality of clock domains, whether these domains be for rising, falling or dual edged circuitry.

*Definition 2.1: Clock period: the time, **T**, between rising or falling edges of a clock signal, usually measured as the signal crosses its 50% voltage level. The inverse of the clock period is the clock frequency, **f**, with units Hertz (Hz), or 1/seconds.*

Definition 2.2: Slew rate: The slew rate is the rate of change of a signal's output voltage level at any given point.

In digital circuits, sinusoidal clocks have low slew rates and these sloped edges create delay uncertainty in most logic families including complementary metal-oxide

semiconductor (CMOS) devices, which is the most widely used digital logic family. In practice, the clock for digital devices is usually shaped as a non-ideal square wave with finite sloped edges, Figure 2.1.

Definition 2.3: Duty cycle: for a periodic signal, the duty cycle ***D*** *represents the percentage of time* τ *that the signal spends in a certain state. For digital circuits, this state usually refers to the logical "1" state.*

$$D = 100 \cdot \frac{\tau}{T} \qquad (2.1)$$

We utilize the logical "1" convention for our duty cycle purposes, and further specify that

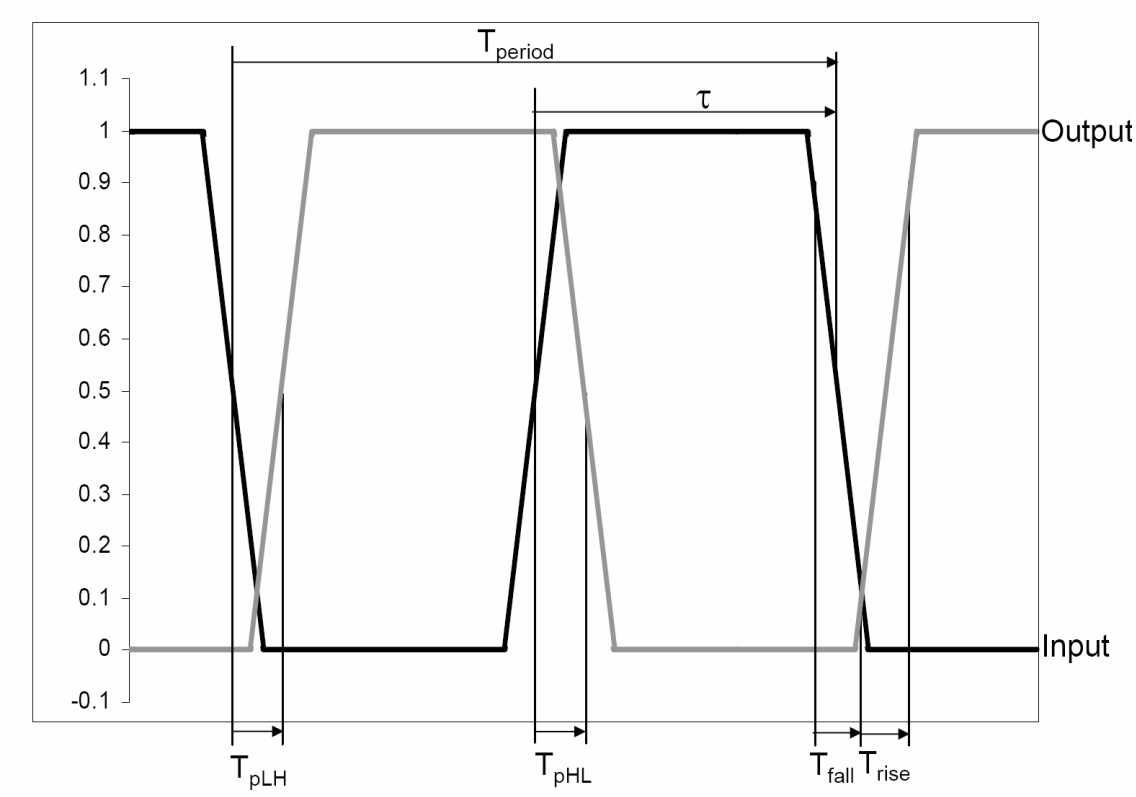

Figure 2.1: Summary of relevant measurements.

the logical “1” state includes all of the time that the signal spends above the 50% voltage level. Clocks are usually assumed to have 50% duty cycles, spending half the time above the 50% voltage level and half the time below. Ideal square waves are impossible to achieve on a physical device due to the non-zero resistance and capacitance in any and every portion of the clock network. Ideal square waves would have zero rise and fall times.

Definition 2.4: Rise time: the rise time t_{rise_X} *of a signal is the time interval between a signal at location* X *transitioning from 10% to 90% of the high voltage level.*

Definition 2.5: Fall time: the fall time t_{fall_X} *of a signal is the time interval between a signal at location* X *transitioning from 90% to 10% of the high voltage level.*

Transition times are affected by a device’s inherent drive strength, the load being switched and external factors such as coupling noise. Higher temperatures and lower power voltage will slow down delay through a device.

Definition 2.6: Propagation delay: If X *and* Y *are two logically connected points on a circuit, and* s_X *and* s_Y *are signals at locations* X *and* Y*, respectively, then the signal propagation delay* t_{pXY} *is the time (either positive or negative) between the 50% signal transition at* X *and the 50% signal transition at* Y*.*

While signal propagation delay can sometimes refer to the time between the input of a gate reaching its switching point to the time that the output of the gate reaches its switching point, the more generic definition of Definition 2.6 is used exclusively. For a

clocked element, the signal propagation refers to its clock-to-output delay (t_{CQ}) since it represents the delay between the clock input passing 50% of its assertion value and the data reaching the 50% signal transition at the output. For both clocked and unclocked elements, the signal propagation can be taken separately for a high-to-low transition (t_{pHL}) and for a low-to-high transition (t_{pLH}). Unless otherwise stated, the propagation delay is the arithmetic average between t_{pHL} and t_{pLH}. This elaboration is necessary due to the asymmetry between pull-up and pull-down transistor blocks in complementary logic design, and the inherent physical differences between the devices used in the circuitry, PFET (p-type metal–oxide–semiconductor field-effect transistor) for pull-up and NFET (n-type metal–oxide–semiconductor field-effect transistor) for pull-down. For signals which are required to have 50% duty cycle, t_{pHL} and t_{pLH} must be equal. If they are not, the duty cycle will get shorter if $t_{pHL} < t_{pLH}$ or it will get longer if $t_{pHL} > t_{pLH}$. These measures are summarized in Figure 2.1.

2.3. Clock uncertainty

Since the clock is used to synchronize events across a circuit, their arrival times at all event points t_i need to be well-controlled. Preventing uncertainty in clock arrival times can add significant complexity to designing robust circuitry since errors could render a circuit unusable. Any signal traveling through a given circuit will have a finite propagation delay. As a clock is broadcast through the IC, each path from clock source

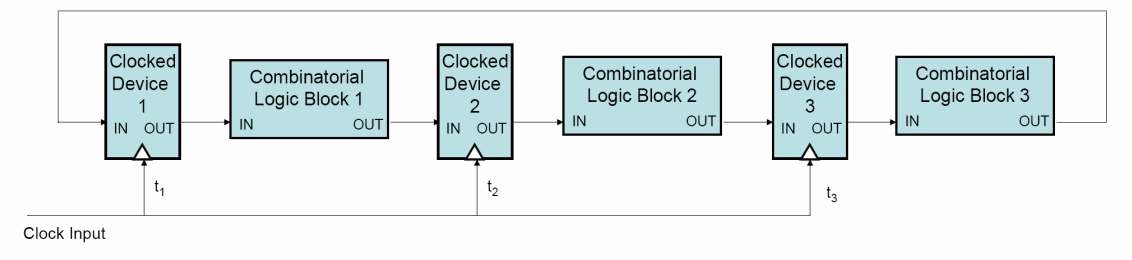

Figure 2.2: 3-register clock routing.

to t_i will be slightly different due to manufacturing and environmental differences. Figure 2.2 shows how a clock signal can be routed to three clocked devices (*R1*, *R2* and *R3*) within a clock domain. The arrival times of clock and data at a register must be properly coordinated for a system to function correctly, since setup or hold time violations can occur if the arrival time of either varies from the expected time.

Definition 2.7: Sequentially adjacent registers: A pair of arbitrary registers connected by a signal path consisting exclusively of unclocked components without any other registers directly in the path. [12]

Definition 2.8: Setup time: The setup time (t_{SU}) is the minimum time that data must be stable before a clock signal is asserted.

Definition 2.9: Hold time: The hold time (t_H) is the minimum time that data must be kept stable after a clock signal is asserted.

Often, hold time is satisfied as the input data propagates from the previous register to the one in question. The time between active clock edges must be long enough for data to propagate between sequential registers while satisfying the setup and

hold times. The *minimum clock period* will be the longest such time between any pair of sequentially adjacent registers in the system. Due to fan-out and fan-in, there are usually many sequentially adjacent registers to any given register in the datapath and every pair must be considered. If the propagation delay of a given combinatorial block is t_{CLi} and the propagation delay of a given register is t_{CQi}, the clock period required between register *Ri* and *Rj* is:

$$t_{pij} = \max(t_{CQi} + t_{CLi}, t_{Hj}) + t_{SUj} \qquad (2.2)$$

For the circuit in Figure 2.2, the minimum clock period will be the maximum between $t_{p(1,2)}$, $t_{p(2,3)}$ and $t_{p(3,1)}$. In a typical synchronous system, there will be many sequentially adjacent paths and they must all be considered when choosing a clock frequency, *f, for a specific clock domain*.

Definition 2.10: Clock domain: a collection of all sequentially-adjacent registers connected to a single clock source. We will consider mutually exclusive sets of sequentially-adjacent registers (with no logical connections between them) to be in different clock domains even if they connected to the same source clock.

Definition 2.11: Clock skew: The clock skew between two clock registers R_i *and* R_j *is the difference between the clock arrival times* t_i *and* t_j*, respectively. The times* t_i *and* t_j *are taken with respect to an arbitrary, but identical reference point. Generically, skew is the variation in arrival times for two signals that are supposed to arrive simultaneously.*

$$t_{skew\_ij} = t_i - t_j \qquad (2.3)$$

Clock skew is especially important between sequentially adjacent registers since it can affect the clock time required between registers.

$$t_{pij} = \max(t_{CQi} + t_{CLi}, t_{Hj}) + t_{SUj} + t_{skew\_ij} \qquad (2.4)$$

When the skew is negative ($t_{skew} < 0$), the data can arrive at register R_j (the destination register) early, potentially violating the hold time constraint or causing a race condition where incorrect data is latched. This phenomenon can be useful in certain datapaths since it increases the time available between clock assertions for t_{CQ}, t_{CL} and t_{SU}, permitting longer datapaths between registers with negative skew. This is sometimes called *beneficial skew*. The opposite is true when $t_{skew} > 0$; this decreases the time available between registers. If not taken into account, this can lead to setup time violations and potential loss of data. Clock skew can occur due to different line lengths, buffer delays, device parameters, noise or environmental variation. *Passive* parameter variations can include changes in resistivity, fringing capacitance and line dimensions. *Active* parameter variation can include changes in transistor threshold voltages (when a device switches) and electron and hole mobility of devices (how quickly a device changes).

For Figure 2.2, the clock is traveling in a left to right direction. It will arrive at *R2* after *R1* and *R3* after *R2*. This creates beneficial skew for these paths. However, between *R3* and *R1*, there will be positive skew. Beneficial clock skew can be designed

into a system and used to increase a system's clock frequency, but only when clock and data are moving in the same direction. In a modern synchronous circuit, there are many complex inter-dependent register-to-register paths, where it is difficult to provide beneficial skew in one area of the circuit without creating harmful skew in another.

Definition 2.12: Clock jitter: Clock jitter is the deviation of a clock's output from its ideal position. Deterministic jitter is bounded in amplitude and originates from non-random specific sources such as device imperfections, cross-talk or power-supply or grounding problems. Random jitter originates from Gaussian noise components in a system such as from substrate or power noise. [17]

The term jitter applies to a change in amplitude, phase or frequency in a clock that occurs from *cycle-to-cycle* or over longer periods of time. Under the broadest definition of jitter, early or late clock arrival times can be considered a form of jitter, but the jitter is usually viewed as a *temporary* phenomenon. This deviation can be periodic in nature as long as the behaviour varies from cycle-to-cycle for a given operating condition. This interpretation creates a non-overlapping distinction between clock skew and jitter. We will apply the term clock skew to a fixed deviation in a clock arrival time, caused by process variation, defects, a change in operating temperature or other such occurrence. This skew can affect rising edges and falling edges asymmetrically. We will apply the term *clock jitter* to a temporary phenomenon that causes a non-static variation in clock arrival time caused by random or predictable phenomena such as a momentary drop in the voltage of the power line. It is important to have well-controlled clock arrival

times throughout an integrated circuit to ensure correct behaviour. The primary sources of jitter are the clock generator (usually a phase-locked loop, PLL) and the effect of power supply and coupling noise on the clock buffers [18].

2.4. Clock networks

Definition 2.13: Clock distribution network: A circuit pathway that delivers a clock signal to every segment of a synchronous circuit that requires it to ensure the correct operation of the system.

In modern integrated circuits, the clock distribution problem is becoming increasingly difficult since device behaviour is becoming increasingly variable, both from chip-to-chip (inter-device) and from device-to-device (intra-device) within an IC. In deep-submicron technologies, wire delays do not shrink as quickly as device delays due to their thin and tall wires having higher resistance and higher capacitance. Consequently, wire delay consumes a greater portion of the clock period and the transportation of any signal across a typical die requires longer than one clock period [19]. This fact and the ever-increasing fan-out of the clock make distributing clock signals even more challenging. Clock power can range from 30-50% [20],[21] in standard high-performance integrated circuits, and up to 70% [22] in some specialized devices like FPGAs. The most common approach is for synchronous systems to have zero skew between all clock arrival points to simplify the timing specification required for the datapath. Even

with systems designed for zero skew, it is necessary to design in a safety margin to maintain correct circuit behaviour in the presence of clock and data uncertainty, a 10% of the clock period rule of thumb is common here [23],[24]. Timing violations that occur at the edge of this range are difficult to detect, as they may be device-dependent and intermittent, only occurring under certain conditions. Many circuits have had to increase tolerances to assure proper operation, which has a negative effect on the performance of the system. Modern devices contain many clock domains, which all must be routed properly, which adds further complexity to the problem.

2.4.1. Symmetric clock tree

Symmetric clock trees are the most commonly studied approach to distribute a clock signal to a large number of clocked elements in a synchronous circuit. This structure takes a source clock and fans it out into n points using a constant wire length to every point. From each of these points, the signal branches out again into n wires resulting in n^2 intermediate end points. This structure continues recursively until all clock destinations are reached. The number of branches required along a single path represents the number of *levels* in the clock tree. In a binary or Y-tree, each branch point splits into 2 branches with the same size and shape, but possibly different orientation. Every branch at a given level should be the same size with similar geometry. In an H-tree, the source clock is split into two, twice per level. At every split, the outgoing wire is placed at a right angle to the incident wire, creating an “H”-like structure. The length of

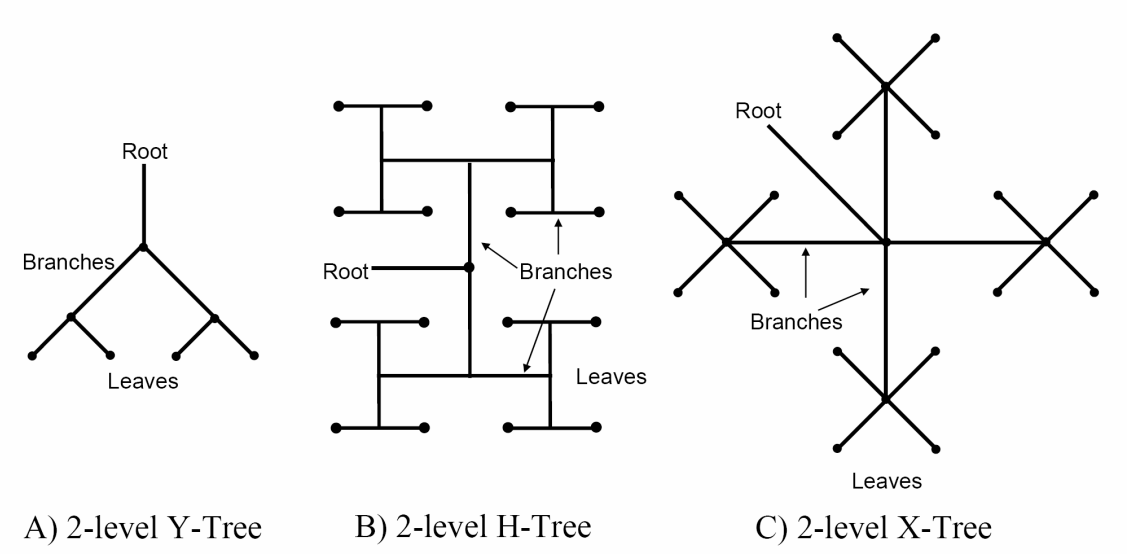

Figure 2.3: 2-level symmetric clock trees.

wire is constant for every segment at a level and is halved for each subsequent level. Each H-structure fans out to 4 branches and uses Manhattan routing since every wire is either vertical or horizontal. In an X-tree structure [25], the horizontal and vertical wires are replaced by diagonal wires saving interconnect length, provided diagonal wires are possible in the technology.

Examples of these three symmetric tree structures are shown in Figure 2.3. Within a clock tree, the clock source is known as the *root*, the single path that transports the signal to the first branching point is known as the *trunk*, the distribution paths are known as *branches* and the individual clock destinations (usually registers) are known as *leaves*. These trees may be completely *passive* with no buffering in the signal path but in practice, the fan-out is too large to do so effectively. Clock buffers are almost always required on the path and they are placed symmetrically across the clock network. They may be placed at every level or at arbitrary (symmetric) intervals in the tree. This creates

an *active* tree with inline buffers or repeaters to regenerate weak signals along the clock path. To maximize signal integrity, impedances need to be matched at every branching point to minimize reflections. Each branch at level *i* in the tree must fan-out into *n* branches at level *i+1* each with *n* times the impedance of the line entering the branch point [22]. This configuration is known as a *tapered* tree. The clock signal travels from root-to-leaf in this network following congruent paths both in wire geometry (width and length of all wire segments) and the clock buffer configuration (size, number and location) so barring any manufacturing and environmental variation in the paths, the total clock delay for each path will be constant, resulting in zero skew between any two paths.

2.4.2. Asymmetric buffered clock trees

An asymmetric buffered clock tree is the most commonly used form of clock distribution for ASICs. The majority of modern clock networks are designed with this approach using specialized CAD tools. These tools start with the location of each of the clocked elements and generate a suitable tree structure by varying the wire length, the buffer sizing and the fan-out at each branch point to achieve near-zero clock skew between all of the sequentially adjacent registers. Figure 2.4 shows a typical asymmetric clock tree. It is also possible for an automated clock tool to use the design specification for each of the clocked elements, specifically the acceptable range of clock arrival times considering its dependencies on a device's sequentially adjacent components. The clock layout tool will take this information, also known as a *skew schedule*, and create a

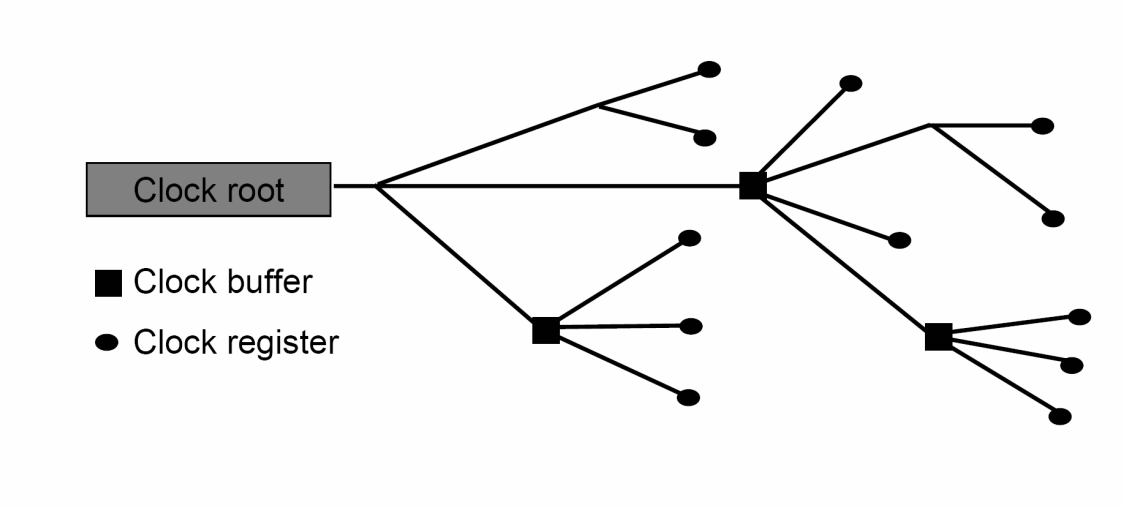

Figure 2.4: Asymmetric buffered clock tree.

suitable clock tree. While this approach can make use of beneficial skew to increase performance, it also creates dependencies on the clock arrival time of many registers so that changing one portion of the design could require complete regeneration of the clock network.

2.4.3. Clock mesh

Since devices and interconnect across an IC will exhibit variances in characteristics due to manufacturing discrepancies, clock skew might occur due to the mismatched clock delays from clock root-to-leaves. The primary sources of the mismatch are the clock buffers, specifically the mismatch between buffers that cause devices at a given level of the network to switch at different times [26]. One approach to counteract this is to shunt all of the buffer outputs at a given level, creating a *clock mesh*. These shunts delay the switching of buffers that are too fast, and speed up the switching

of buffers that are too slow. The primary drawback of this approach is the increased power consumption caused by the short circuit currents that will exist in the presence of device variation. An additional drawback is the large interconnect cost of connecting points that get progressively further from one another for every additional tree level. Due to the transport delay of the shunt signals, this methodology cannot completely eliminate clock uncertainty. However, due to its ability to compensate for some variation in the clock network, this is the most commonly applied clock architecture in high performance microprocessors despite its cost. This approach usually requires a global clock tree to feed the different portions of the mesh with quasi-simultaneous clocks [27].

2.4.4. Resonant clocking

Resonant clocking is a newer approach to clock networks that requires removing all the clock buffers and creating an LC-tank to allow a natural oscillation in the clock signal that requires less energy to maintain than a traditional network [28],[29],[30]. In an ideal configuration, the energy consumption would be zero, but due to the resistance of the metal lines in the network, the configuration is lossy. The LC-tank is formed by adding an inductor and using the capacitance of the clock leaves. Resonant clock networks have demonstrated power savings of over 50% and IC core power savings of between 20-35% when compared to conventional networks [31],[32]. One of the drawbacks of the system is its use of sinusoidal clock signals, which can create short circuit currents because their long rise and fall times, and are much more susceptible to

clock jitter when compared to clocks with sharper edges. Resonant clocking requires modified registers and latches that are compatible with sinusoidal clocks [33],[34]. Current research in resonant clocking deals with approaches and techniques to minimize jitter and noise [30],[35].

2.4.5. Standing and traveling wave network

A *standing* wave clock network presents another approach to distribute clock signals [36]. A standing wave is formed by superimposing a forward clock produced by an AC voltage source with a return clock, produced by reflecting the forward clock back from a ground termination at the opposite end of the conductor. This approach ideally creates a zero skew set of clocks along the length of the line with amplitudes that vary between 0 and the initial amplitude of the AC source depending of the position of sampling [37]. The resulting clocks then need to be regenerated and possibly converted to a square wave for use. Certain portions of the clock line's output will be unusable since the low amplitude signal in that area will render it unrecoverable.

Travelling waves are used in clock networks to create clocks that allow full voltage swing at all sampling points. These networks usually use a rotary clock ring [38],[39],[40] to create the traveling wave along a pair of conductors separated by a series of cross-coupled inverters to regenerate the signals and maintain the oscillations. The frequency of each of these rings will vary depending on the line length, so a number

of these rings are usually interconnected to create a mesh with greater tolerance to variability [41]. The primary drawback of this approach is that unlike with standing wave clock networks, the phase of the clock signal will vary depending on the location of the sampling and every point in a ring will only be in phase with its corresponding point in every other ring. This leads to significant difficulty in designing synchronous circuitry using this technology.

2.4.6. Hybrid structures

Modern clock networks usually use a mixed clocking strategy, pairing one global clock routing technique to another local routing technique [42]. A common pairing would be a global H-tree followed by a local mesh [43], minimizing the power consumption of the mesh while maintaining low local variability. Link insertion [44]-[47] is another approach used to shunt selected portions of the clock networks that should have zero skew, thereby realizing many of the benefits of clock meshes. Transport delay through the shunt wires will prevent this method from eliminating all skew. Typically, the global distribution can be constructed using a mesh or tree, and the local network can use a tree, mesh or fishbone structure [16]. A fishbone structure has a clock trunk with leaves arbitrarily attached orthogonally to the trunk wherever the clock is needed. We will denote any connection point between a global and local clock domain as a *clock tap*.

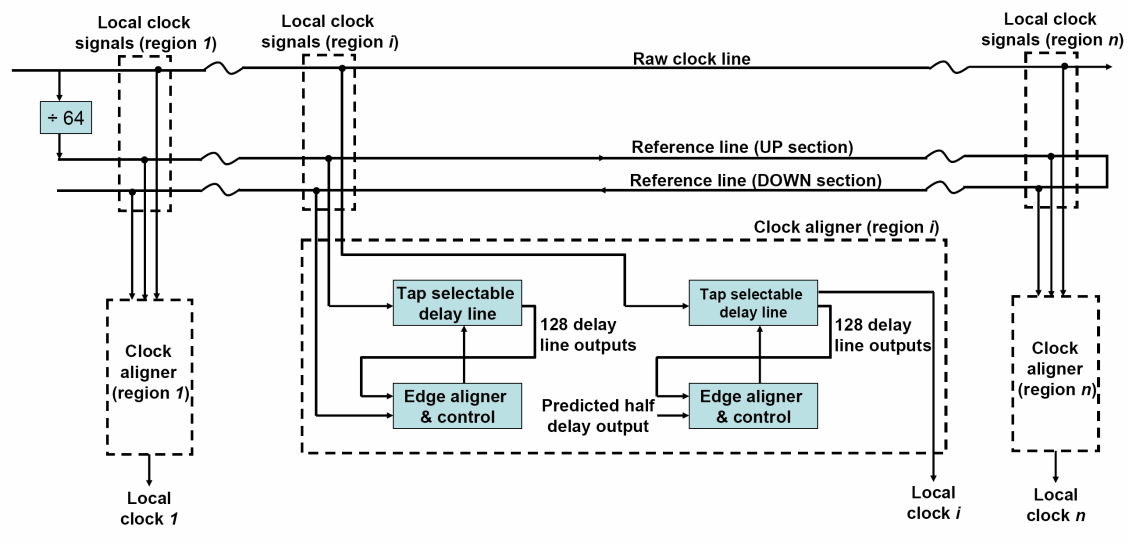

Figure 2.5: Grover's serial clock distribution [48].

2.4.7. Serial clock distributions

Serial clock distributions are a less common form of clocking that aligns each tap clock to a position directly in between two reference clocks traveling in opposite directions (with respect to their phase). This *averaging* technique was first proposed by Grover et al. [48], Figure 2.5. Their scheme uses a 3-wire method with separate raw clock, forward reference line and reverse reference line, with the reference lines tied at the far end of the clock distribution. However this technique requires two distinct clock alignments to synchronize each local tap. First, the pulse interval between the forward and reverse signal is found by delaying the forward (*UP*) signal to align with the reverse (*DOWN*) signal. This interval is halved to create a local half pulse reference signal P_{half}. The P_{half} signals in every clock region will be ideally synchronized due to averaging.

Next, the raw clock is aligned to the region's P_{half} signal to generate the local clock. The delay is created using a 128-element non-inverting buffer chain. A 128:1 multiplexer is used to choose the appropriate tap, setting the accuracy of each alignment to the propagation delay through the non-inverting buffer. Another drawback is that reference clock lines may exhibit different propagation delay than the raw clock line due to differences in geometry and signal frequency. As such, [48] is susceptible to skew from wire mismatches between the reference and the raw clock lines. The system's lack of buffering in the clock path also limits the total load, the distance between taps and the number of taps that the reference line can spawn.

Work in [49] uses an averaging approach for synchronizing digital signals using a two-wire method. This technique has recently been employed by Banu and Prodanov [50] in another configuration that uses a serially distributed layout that is similar to our own[2], but the averaging technique they use is distinct, using analog multipliers at each tap to create the required averaging, Figure 2.6. Their technique provides low skew, but creates a set of sinusoidal clocks that are not of full swing and require level shifting blocks at each tap to be useful in most designs, similar to standing wave systems. Because of the transmission line nature of their clock network, their method does not permit the use of buffers or logic within the clock network, making the system very rigid. The analog components can consume more power than the digital components

[2] Banu and Prodanov's work [50] was published September 10, 2006, shortly after our initial publication [7] on August 6, 2006.

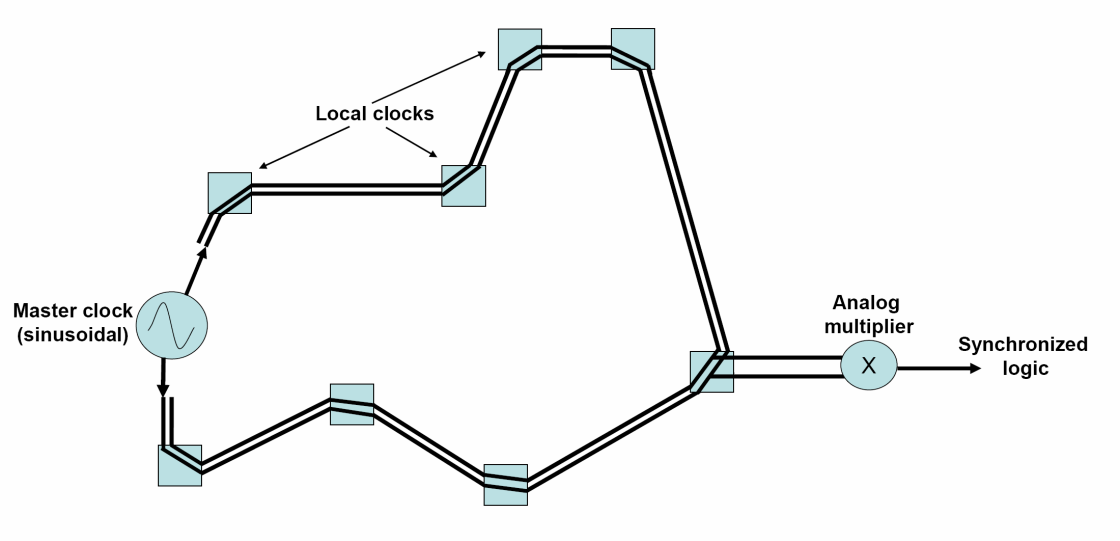

Figure 2.6: Banu and Prodanov's bufferless approach [50].

employed by our system due to non-zero static bias currents. They describe their system as one that uses *bi-directional* signalling, which differs from our use of the term. The authors of [50] use the term to describe the transport of signals propagating in opposite directions along *two* wires, where we will use the *same* conductor (time-multiplexed).

2.4.8. Reconfigurable clocks networks

All of the previously described networks must be created following the design of the logic, and once established cannot be modified since changing the clock delay to one register will affect the clock skew between it and every one of its sequentially-adjacent registers. Present day designs incorporate many clock domains on the same die and special consideration is needed when communicating between them. Many integrated circuits are designed by combining a number of different blocks from different

sources and require that these components work flawlessly with one another for an array of different applications [51]. Connecting these components together in different configurations based on the specific application is not an easy task for standard IC clock networks. Field programmable gate arrays (FPGAs) are generalized devices that may be used for a variety of applications, so their clock domains must be flexible to work for a variety of applications. For example, the Altera Stratix IV series of FPGAs can support up to 16 global clocks that can be routed anywhere on the device and up to 88 regional clocks [52]. For the most part, FPGAs use a spine-and-ribs (fishbone) structure to connect regional and local registers, so the flexibility created by this approach comes at the expense of skew tolerance.

2.5. Skew compensation

Deep submicron technologies exhibit an ever-increasing susceptibility to process variation [53]. When comparing the effect that process variance has on a clock network, the variation between individual devices must be examined. While a certain amount of die-to-die variation is always present, this does not create skew since every device on a die will be equally affected. There is also a discrete component that will exist between every distinct device on a die that is roughly proportional to the effective size of the transistors [54]. Discrete mismatch will always be present no matter how closely located two devices are. With smaller transistor sizes, there is a discretization of the dopant

levels in each device leading to greater fluctuations in threshold voltage and subsequently in delay through a device. Current mobility and mismatch between devices are also phenomena, which can be deemed statistically independent for different devices [55]. Finally there is a distance-related component for variation that will increase depending on how far devices are from one another. This usually occurs gradually, with the possible difference between the behaviour of two devices increasing the further apart they are. Two devices located very close to one another will not be greatly affected by proximity dependent mismatch. While a 10% of clock period rule of thumb has traditionally been assumed for clock skew, this number is easy to surpass with process variance in today's newest technologies [56].

Programmable clock buffers and interconnect which require post-silicon tuning have been introduced to compensate for process variation for selected regions of a clock network [57]. Die temperature can also cause significant variation in the behaviour of buffers in a clock distribution network. Worse, the die temperature will vary depending on local switching activity leading to temperature gradients across the die that will change over time. Increased power density in ICs amplifies this problem by making modern ICs run hotter than before. Localized temperature spikes, or *hotspots* can severely impact the skew in a clock network [58]. Temperature and process variance can result in delay changes of over 50% for sub-65 nm devices [59]. The authors of [60] have developed a clock network with self-adjusting delay buffers to cope with temperature variation, but their methodology is designed to only cope with intra-die temperature fluctuations and

not localized temperature gradients. Traditional dynamic temperature management techniques are limited by the accuracy of their temperature sensors [61].

Traditionally, clock network generation tools distribute clock buffers according to user-defined specifications using a combination of matching wire length and adjusting the placement and sizing of clock buffers to achieve the desired delay, but variability is making this *passive* skew reduction technique less effective. Some clock networks use *active* clock skew reduction techniques to reduce the effect of process variation. Active clock skew reduction techniques use controllers and feedback structures to modify the local clocks and provide de-skewing capability, usually using delay lines to perform the adjustment. This technique can use significant system resources during operation due to the large number of local clocks that must be synchronized in a clock distribution network. Some active techniques do not therefore operate on all of the clock taps, limiting their use to de-skew only two clocks with respect to one another [62], such as with two halves of a clock tree. Multi-point skew reduction techniques have become more popular since they are better able to cope with the large die and high-speed clocking environments that are common [63], but any controllers required must typically be either re-used or replicated for each tap. The overhead of active skew reduction techniques translates to higher power consumption than their passive counterparts.

Some schemes simply employ skew reduction techniques on existing tree distributions [64],[65], but can require extra wiring or a power overhead due to their heavy resource usage. The configuration of Kapoor's approach [66] uses a distribution tree and

a co-located feedback tree and employs skew compensation at every leaf as shown in Figure 2.7. Some schemes perform root-to-leaf skew compensation at the root or leaf for

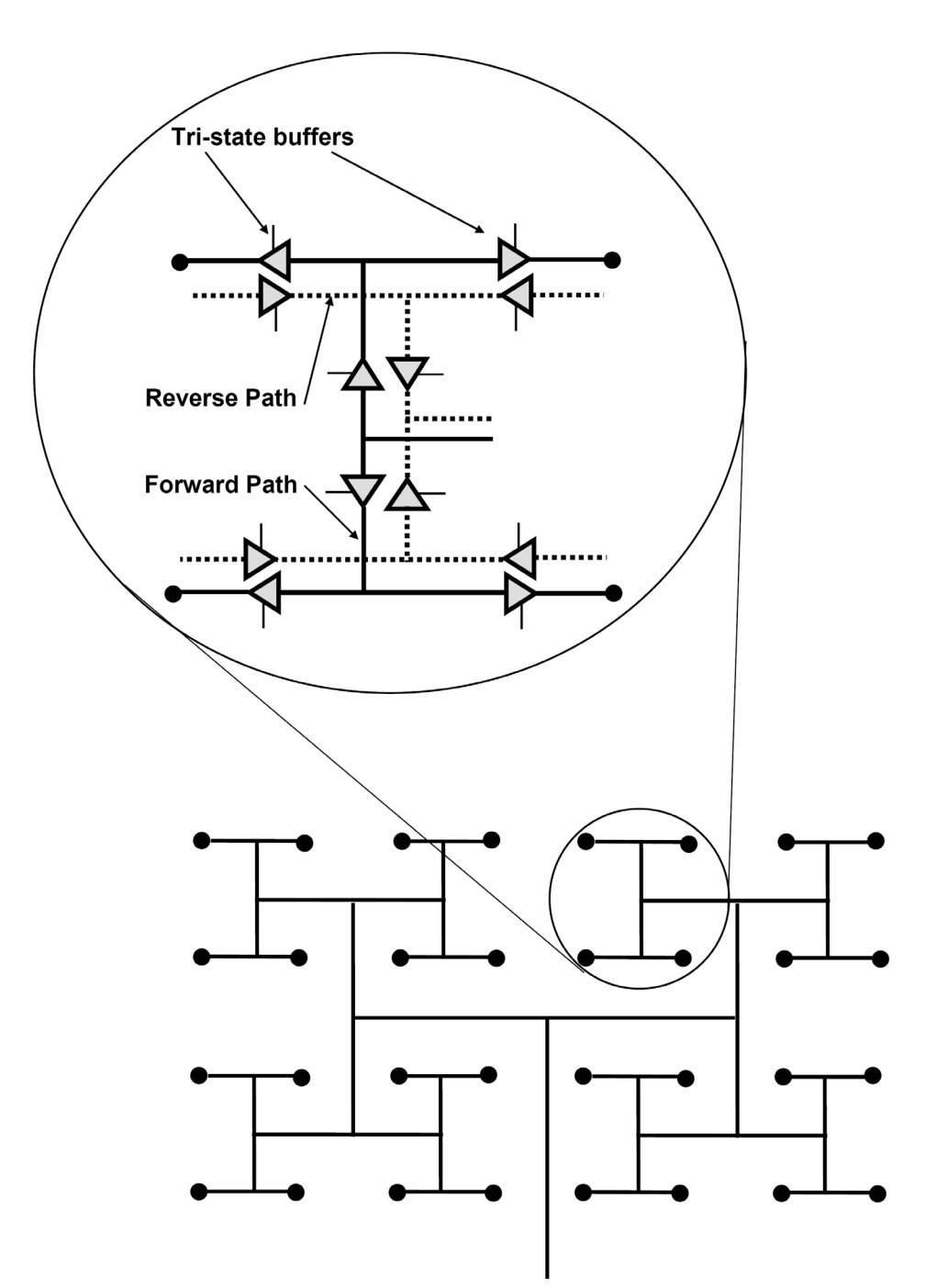

Figure 2.7: Kapoor's skew-tolerant clock tree [66].

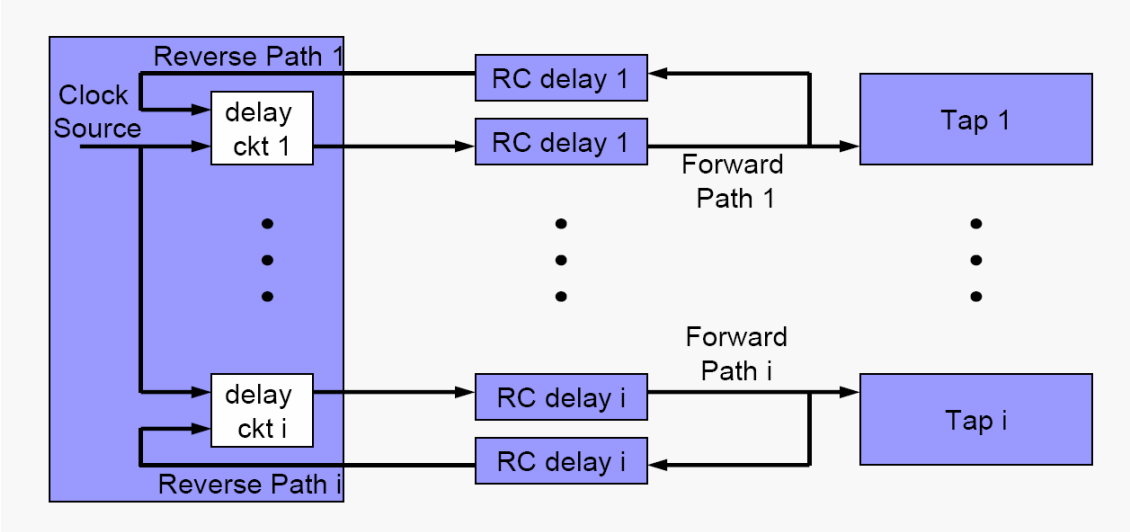

Figure 2.8: Lee's skew compensation scheme [67].

each local area on the IC [67],[68]. This configuration creates a need for a feedback line from every synchronization point to the source. These feedback lines are subject to the same trace matching discrepancies present with H-trees, introducing error to the skew compensation technique [64]. The star configuration of Lee's system [67] employing skew compensation at the root for every leaf is shown in Figure 2.8.

2.6. Clock power

Increased die size and device density of integrated circuits has led to a marked increased in the power consumption of deep submicron designs. The clock can consume the largest portion of on-chip power, often over 25% of the total power [69]. There are three broad sources of power consumption in a device: short-circuit power P_{sc}, leakage

power P_{leak} and dynamic switching power P_{dyn}. All three of these also contribute to clock power P_{clk}. Short-circuit power results from *crow-bar* currents that occur when both pull-up and pull-down portions of a gate simultaneously conduct while the input voltage is transitioning.

Short circuit power is proportional to V_{dd}-$|V_{tn}|$-$|V_{tp}|$, the clock frequency, and the rise and fall times of the input signals. V_{dd} is the supply voltage, and V_{tn} and V_{tp} are the threshold voltages of the pull-down and pull-up devices, respectively. There is no short-circuit current when either pull-up or pull-down blocks are not conducting and it is impossible for both of these to conduct when the sum of their threshold voltages exceeds the supply voltage. Since current technologies rely on decreased supply voltage, short circuit power is decreasing as V_{dd} approaches $|V_{tn}|+|V_{tp}|$. Another useful design approach is to use signals with high slew rates, ensuring that input signal transition times are not much longer than the output signal transition time.

Leakage power creates non-zero static current in an IC. There are three major contributors to leakage power: *gate oxide leakage* which allows current to flow through the gate of devices, *sub-threshold threshold* which allows current between transistor drain and source when devices are supposed to be in a non-conducting state and *junction leakage* across the reverse-biased diodes in the diffusion area of the transistors [13],[70],[71]. Generally speaking, leakage power is increasing in newer technologies due to a decrease in threshold voltage and an increase in operating temperature. However, the total power consumption decreases overall due to lower supply voltages. Clock

networks are less affected by leakage power than logic circuits since clock power is dominated by the dynamic power of the clock buffers due to the large capacitances involved. As a result, clock networks are better able to take advantage of decreased supply voltages and the proportion of device power consumed by clock networks will decrease in future process generations [69].

Dynamic power consumption is the dominant factor in overall power consumption in an integrated circuit, decreasing relative to the total power consumed, typically around 80%, but increasing absolutely due to the greater number of devices on an IC and the increase in the operating frequency of newer devices. The dynamic power in an integrated circuit is:

$$P_{dyn} = K \cdot C_L \cdot V_{dd}^{\ 2} \cdot f \qquad (2.5)$$

where K is a configuration constant between 0 and 1, C_L is the total capacitance being switched, V_{dd} is the supply voltage and f is the operating frequency. The C_L term includes clock driver input capacitance, clock interconnect capacitance and the capacitance of the clock loads [72]. Sometimes K and C_L are combined into a signal term describing the *average* capacitance being switched from 0 to 1 during each clock cycle. Charge is *sourced* when the output capacitance toggles from 0 to 1 and *sunk* when it is transitions from 1 to 0, so power is only consumed once every two transitions. For logic, this means that each output charge/discharge cycle will take at least two clock cycles since the output value is only expected to change once per clock cycle. Glitching in the logic circuitry can create spurious transitions resulting in additional power consumption.

Typically then, *K* in Equation 2.5 will be *0.5*q*, where *q* is the probability of the output toggling during a clock cycle, or switching activity. For a clock network, the devices in the clock path will toggle twice *every* clock cycle. As such, the switching activity is of the clock network logic is 2 and *K* is 1! This results in significant dynamic power consumption for clock networks since the switching activity and the total capacitive load in a clock network is large. Clock power due to interconnect wiring will increase for newer technologies as inter-wire capacitance increases due to closer wire spacing and taller wires [69]. While dynamic power can be reduced by shrinking the clock buffers, this has a negative effect on short circuit power since each clock buffer will be forced to switch larger capacitances and the signal transition times will fall as a result. Power can also be decreased by operating at lower voltages, but then leakage power will increase. Currently, most of the power consumed by an IC is dynamic with approximately 10% going to short circuit consumption [73]. Leakage accounts for approximately 30% of dynamic power [74].

Chapter 3:

A dual reference signal averaging single clock distribution network

3.1. Introduction

The clock distribution problem represents an increasingly difficult challenge due to progressively more complex systems, decreased power supply voltages, larger die sizes and higher clock frequencies [66],[75]. Traditionally, passive forms of clock skew reduction were used to balance all the leaves in a clock distribution network by a combination of matching wire length and adjusting clock buffer delays [67],[76]. While the performance of automated clock layout tools has improved significantly, the problem has

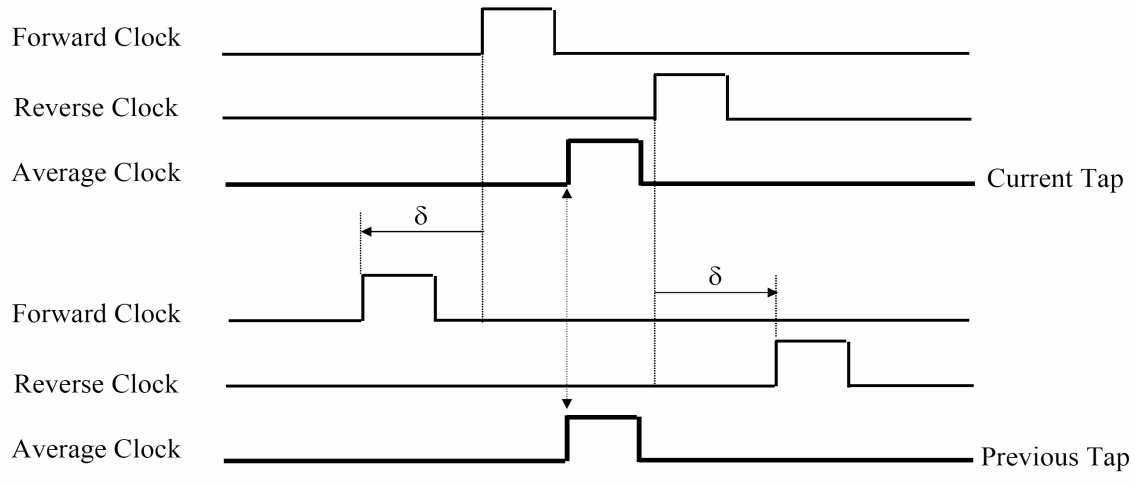

Figure 3.1: Underlying concept of averaging.

been complicated by the move to a component-based design approach using pre-designed blocks and IP cores from several pre-existing sources. Clock buffer mismatches and in-die process variations have become a limiting factor in maintaining tight skew tolerance [14],[68],[77]. Buffer mismatch due threshold voltage (V_t) and the short-cicuit drain current (I_{dss}) increase with newer silicon generations and the mismatch accumulates as clock signals propagate through the distributed buffers in a design [78]. The considerations for appropriate clock distribution networks include clock signal characteristics such as fast transition times, a balanced duty cycle and low clock skew [79]. Floorplanning requirements add additional complication to the clock distribution, since matching clock trace lengths over irregularly shaped domains is difficult.

We discuss here an alternative to traditional H-trees, incorporating a dual reference signal based clocking strategy to distribute clocks serially, using averaging to eliminate systematic skew, post-silicon. Figure 3.1 shows the concept behind averaging. The reference clocks are treated as pulses for presentation purposes, but they are, in

fact, periodic with equal duty cycles. The technique that we employ facilitates the construction of multi-tiered clock distribution networks, such as those in ICs using multiple IP blocks. We use one bi-directional line to connect every tap, so any local process variation present in the line affects all taps equally, and is thus not a source for additional skew. Portions of the clock domain can be pruned easily by pausing the clock at the appropriate taps without disturbing the rest of the clock domain. While [66] and [67] are based on a similar idea, their use of different forward and reverse clock lines between taps can lead to trace length discrepancies that can hurt the effectiveness of the skew compensation circuitry.

This approach compensates for fixed global and local process variation, temperature and power supply variations of the distributed buffers in clock networks, which can create significant skew [80]. Our clock distribution has all the benefits of active skew compensation techniques, using a closed loop synchronization approach to align the clocks for each domain. During operation, the circuitry operates in an open loop with the synchronization hardware disabled, providing significant power savings – a typical benefit of passive clock networks. We are able to easily incorporate clock gating at each tap into our design, without disrupting the loading of the clock drivers or increasing wiring cost [81]. Our system can also provide many of the benefits of clock *root* gating since large portions of the serial clock line can be paused [82]. The system can also be used to provide beneficial skew between taps or to balance different local clock distribution components for different tree depths or latencies, such as those found when incorporating multiple IP blocks [63]. In this chapter, we examine the design

requirements and operating characteristics of reference-based clocking: our dual reference signal averaging clock distribution network.

3.2. Implementation approach

To implement a single clock design using our dual reference signal averaging clock network, the clock domain is divided into *n* smaller subsections, each of which is connected to a tap. Each of these subsections should be roughly equal in size to help match the tap-to-leaf delays for every tap, but this can also be achieved through buffer placement and sizing for arbitrarily sized local regions. The clock taps do not need to be distributed in close proximity, nor do they need to be regularly spaced. The smaller the area of these subregions, the more taps are required, but there is less variability within each subregion.

The underlying concept of our clock network is shown in Figure 3.2 for four taps. Each tap contains the necessary hardware to delay the local clock and to route the reference clocks between subregions. All the taps are connected together as a “thread” using a single wire to create a clock domain with the required shape and size. Both the region *1* input node (forward clock) and the region *n* input node (reverse clock) are tied to the global clock. Since the clock distribution line has a constant delay ($K+\delta_s$) over its entire length (where δ_s is the delay through a switch), if the delay of the forward clock at a tap is δ_+, the delay of the reverse clock will be $\delta_-=K-\delta_+$. For every tap, the local clock is

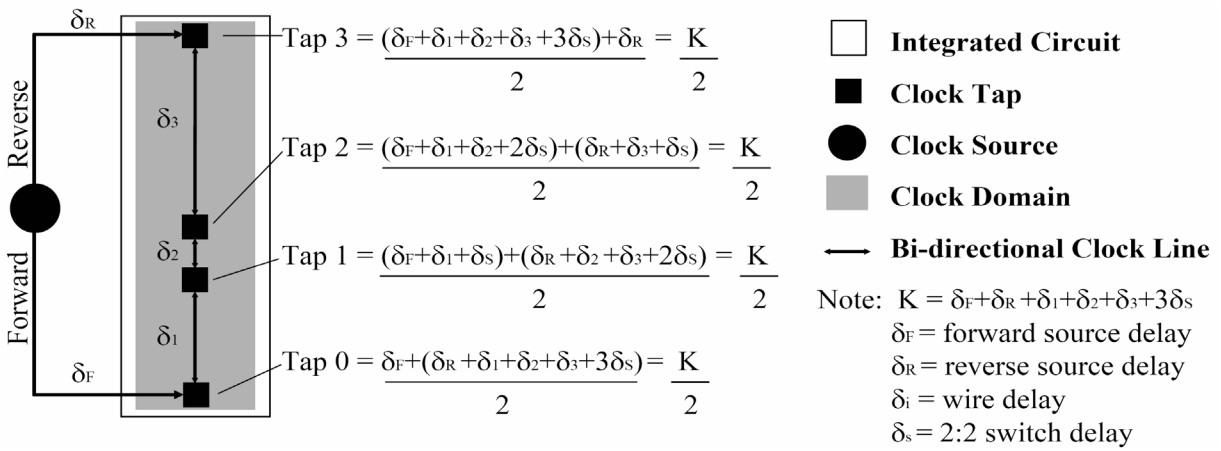

Figure 3.2: Reference-based clocking for a single clock domain.

aligned to the midpoint between the forward and reverse reference clocks, resulting in an averaging of the temporal positioning of the clock edges. For every tap, the resulting rising edges all occur at a fixed time:

$$\frac{\delta_{-} + \delta_{+}}{2} = \frac{(K - \delta_{+}) + \delta_{+}}{2} = \frac{K}{2} \qquad (3.1).$$

Since the reference clocks are distributed along a single signal path, the delay between two adjacent taps can be very closely matched for signals traveling in either direction. The bi-directional clock line can be routed around obstacles easily without compromising skew tolerance since the clock taps are daisy-chained. The clock threads can cross other clock domains easily and are simple to lay out. The clock distribution network requires three distinct phases to work properly: synchronization, calibration and operation. These are discussed in the following sections.

3.2.1. Synchronization

During the synchronization phase, each clock tap is sequentially calibrated, starting with the region *1* tap and ending with the region *n* tap. The forward and reverse delays are all taken modulo the clock period to account for the periodic nature of the signals. Synchronization for each clock thread can be achieved by predicting what each source-to-tap delay might be and hard-coding the required delay line setting into the system. This method has a number of benefits including zero synchronization time and reduced circuitry overhead by eliminating the reverse path of the clock thread, the phase detector and the control circuitry. However, it does not compensate for any variation-induced skew, but it would still be useful for distributing irregularly shaped clock domains where clock skew is not a great concern.

To take full advantage of reference-based clocking, online skew calibration should be included. To produce a fully dynamic system, each tap requires a clock routing switch which can be set to *FORWARD*, *REVERSE* (bypass) or *SYNCHRONIZE*, a delay line, a phase detector and control circuitry. When set to *FORWARD*, the forward reference signal is routed from the current tap (tap *i*) to the next tap (*i+1*). When set to *REVERSE*, the reverse reference signal is sent from the current tap to the previous tap (*i-1*). When sent to synchronize, both the forward and reverse reference signals are routed to the current tap. This configuration allows the forward clock to be *averaged* using delay lines.

3.2.2. Calibration

With all the taps synchronized, the forward clock is routed through the entire clock thread for the calibration and operation phases. The reverse clock is disabled. Because of the nature of the synchronization method, the resulting synchronized tap clocks can either have positive or negative polarity with respect to one another, depending on the relative phase of the reference clocks at each individual tap. This result could be sufficient for a dual clock edged device, but for a single clock edged device, a calibration phase may be required.

To perform the polarity adjustment, the system could examine each pair of adjacent clocks individually and perform an inversion for the higher order (number) clock, starting with the output of taps *1* and *2* and ending with the clocks in taps *n-1* and *n*. The relative phase is known a priori for most configurations, so the simplest method would be to determine the required polarity in advance and program the system accordingly.

3.2.3. Operation

Following calibration, the components of the clock distribution that are unused during operation such as the synchronization controller, phase detector and calibration circuitry are disabled to save power. The taps in the clock domain are not affected by a fixed phase shift in the global source clock, since this simply delays or advances all the clocks in a domain by a constant amount. The resulting phase shift in the clock domain

will equal the shift in the global clock. All of the clock domains derived from this global clock will shift accordingly, maintaining a fixed phase relation between the clock domains. For clocks derived from an unrelated global clock, the phase relation will change with respect to the clocks in our serial clock domain. However, as with most other such instances, synchronizers would typically be employed between these clock domains to enable inter-clock domain communication [83] since it is difficult to maintain a constant phase difference between two unrelated clocks. Centroid layouts and properly sized devices can greatly minimize local intra-die process variation, assuring constant forward path and reverse path delays through the clock routing switches. Inter-die variations can be compensated for using dynamic tap synchronization. The primary source of skew for our method is the in-die process variation that can occur between the delay line pair used for averaging. Variation can alter the drive strength, loading or delay of devices, so it is critical to minimize its impact.

3.3. Wire length savings

Our method goes beyond just eliminating systematic (structural) offsets, as is the case with most active de-skewing circuits [84], by compensating for any line or non-local transistor imbalances that may exist on the global clock routing path better than any other active skew reduction technique. Because the reference-based distribution can simply route a clock in the shortest possible path to each tap, there are significant wire savings with respect to a traditional H-tree that requires redundant wires to match the

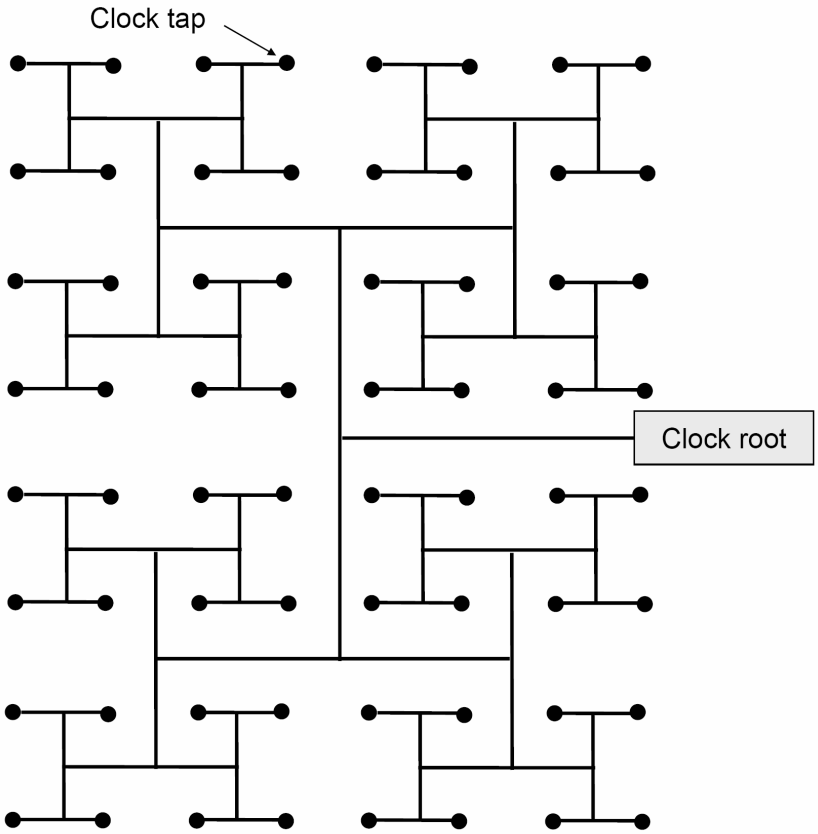

Figure 3.3: A 64-tap H-tree.

path length from source to every tap. It is important to consider wire length since decreasing wire length will also decrease the amount of capacitive loading in the clock network. In deep submicron devices, interconnect and device delays are roughly equal contributors to total device delay [85]. An averaging clock distribution can contain an arbitrary number of nodes and can be laid out manually or using standard cells, whereas H- or other tree solutions require special balancing tools to generate synchronized clocks. Figures 3.3 and 3.4 show how a 64-tap H-tree distribution compares with our reference-based one. A decrease in wire length will reduce clock load. Replacing an H-tree with 5 levels or more with a reference-based design can save over 30% of the clock wire length. Table 3.1 shows a summary of the wire lengths (in units) that can be

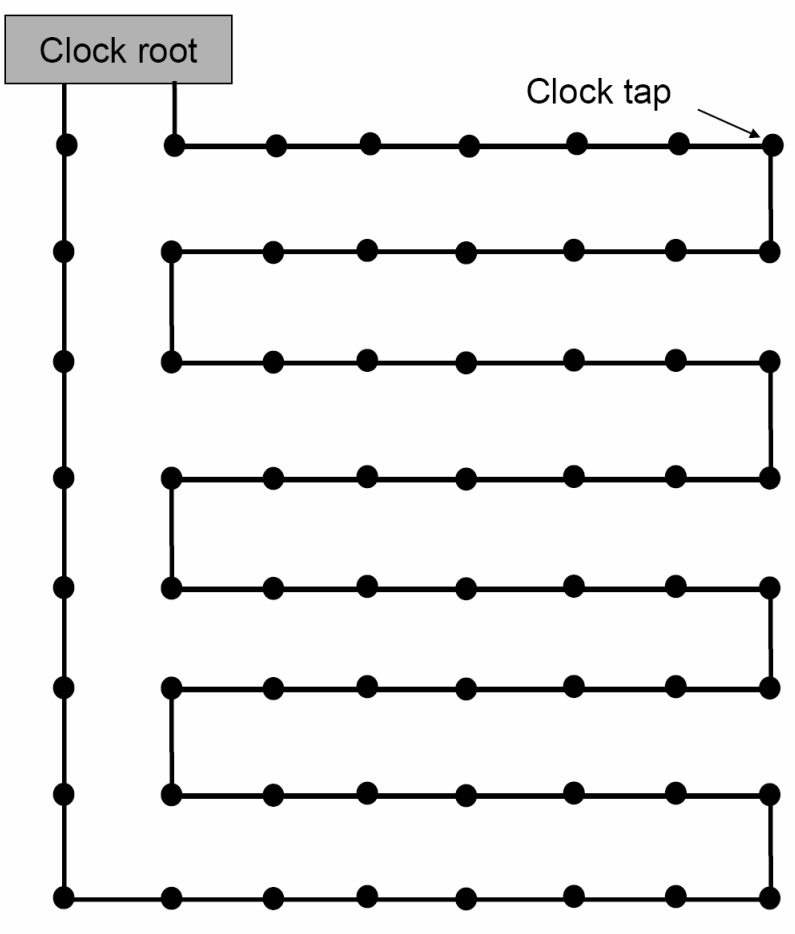

Figure 3.4: A 64-tap reference based clock distribution.

achieved between different depths or levels (*n*) of square H-trees and comparably-sized referenced-based clocking solutions assuming unit length spacing between each of the clock taps. The wire length numbers assume 2^{n-1} length wires for each level of the tree [86]. These numbers ignore the tap-to-leaf distributions, which will be identical for both implementations. The wire length is governed by Equation 3.2 for serial clocks and Equation 3.3 for H-tree clocks.

$$Length_{linear} = n^2 \qquad (3.2).$$

$$Length_{H} = 3 \cdot 2^{n-1} \cdot (2^{n} - 1) \qquad (3.3).$$

The wire length has a direct influence on the power consumption of the clock distribution network, since a longer wire length has a larger capacitance that must be switched twice per clock cycle. The variable wire length is reflected in the C_L term in Equation 2.5.

H-tree depth	Number of taps	Wire length (H-tree)	Wire length (serial)	Savings (%)
1	4	3	4	-33.33
2	16	18	16	11.11
3	64	84	64	23.81
4	256	360	256	28.89
5	1024	1488	1024	31.18
6	4096	6048	4096	32.28
7	16384	24384	16384	32.81
8	65536	97920	65536	33.07
9	262144	392488	262144	33.21
10	1048576	1571328	1048576	33.27

Table 3.1: Wire length comparison

3.4. Architecture variants

The average of the reference clocks at each tap is taken by delaying the forward clock to align with the reverse clock through two delay lines. The placement and architecture of these delay lines affect the area required, the matching between delay lines, the susceptibility to process variation and the usability of the system. Four such variants were explored for an *n* tap single clock structure: one using *2n* delay lines, one with *n+1* delay lines, one with *n* delay lines and one with *2n* delay lines using unidirectional conductors for the forward and reverse reference signals.

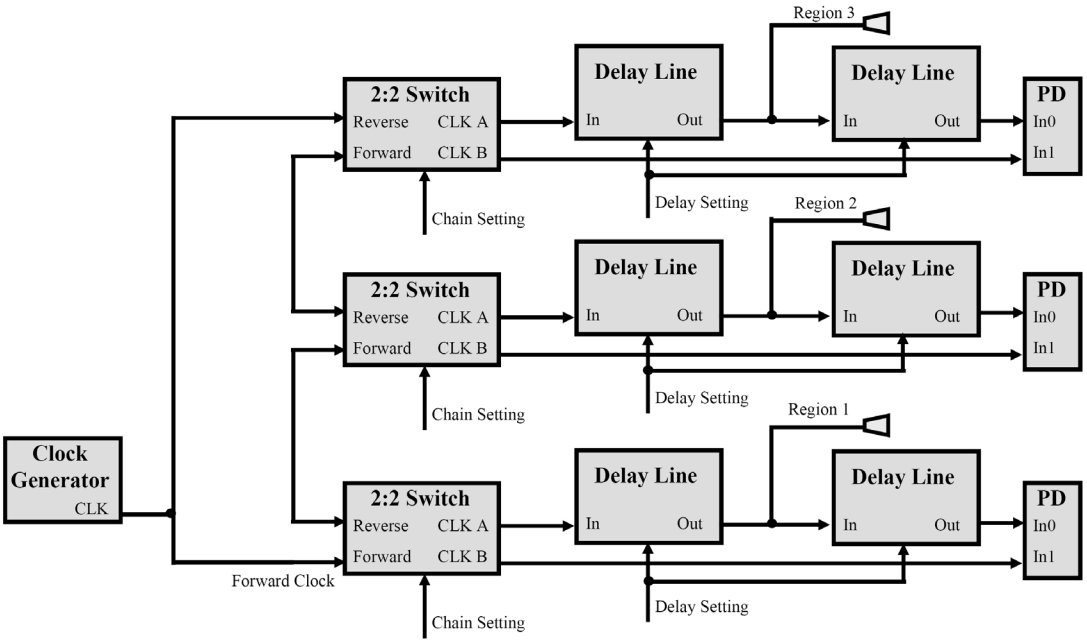

Figure 3.5: Architecture using *2n* delay lines.

3.4.1. Architecture with *2n* delay lines

The most intuitive approach for averaging is to include a pair of delay lines at each tap and use identical settings on both to perform the required clock alignment, as shown in Figure 3.5. Each tap consists of two delay lines, one clock thread switch (*2:2 Switch*) and one phase detector (*PD*). *CLKA* represents the forward clock path within the tap and *CLKB* represents the reverse clock path. Once aligned, either the clock path can be modified to bypass one of the two delay lines, or the delay setting on one of the two delay lines can be set to an arbitrary constant delay for every tap. Modifying the clock path requires the addition of clock routing, which could potentially add skew due to tap-to-tap variances. An arbitrary constant delay will induce unnecessary signal transitions that will add to the power consumption of the device. During operation, half of the delay

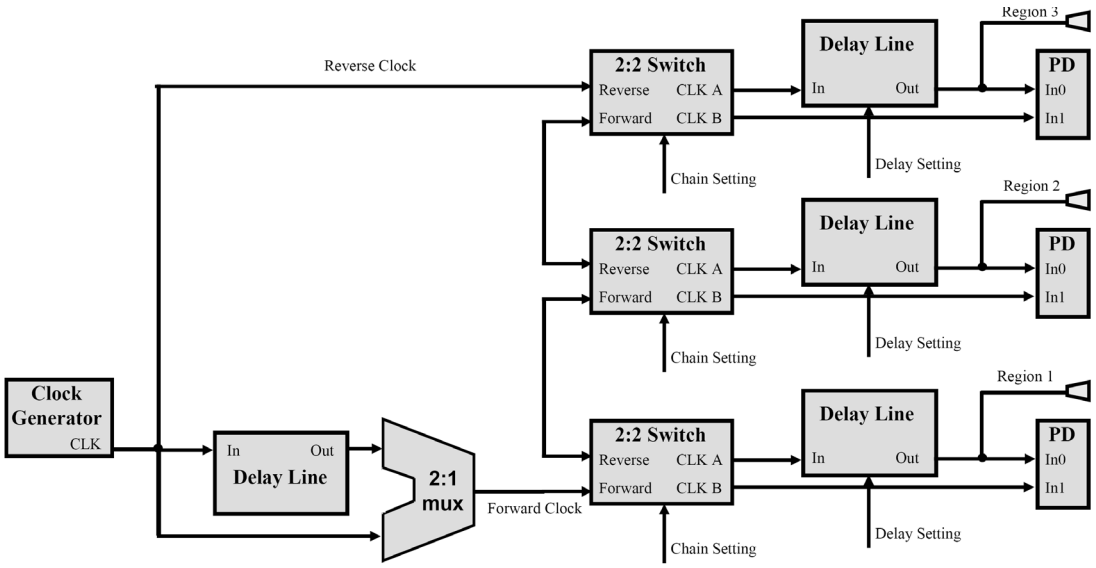

Figure 3.6: Architecture using *n+1* delay lines.

lines are not required, meaning there is wasted circuit area using this configuration. Our 180 nm layout of this configuration requires an area of 6000 μm^2 per tap. This method achieves tolerance to process variability since it only requires good matching between the two adjacent delay lines at each tap, which is expected considering their proximity to one another. The averaging method compensates for tap-to-tap variances since the delay line pairs since the accuracy of the average is solely dependent on the matching between the two in-tap delay lines.

3.4.2. Architecture with *n+1* delay lines

One drawback of the previous configuration is its inefficient use of area. Since one of the two delay lines are only used for synchronization, it is possible to share a single delay line for synchronization for all taps at the clock source, requiring only one dedicated delay line at each tap. The *source* delay line could be kept in the signal path

or bypassed at a designer's discretion during operation. This architecture is shown in Figure 3.6, and the circuit layouts for each tap and a clock selector to bypass the source delay line are shown in Figures 3.7 and 3.8, respectively. The area required for each *2n+1* tap is 3750 μm^2 and the 3400 μm^2 for the clock source. Any constant delay injected at the clock source affects all taps equally, so it does not alter clock skew between taps. In this manner, the source and the tap delay lines must each be programmed separately which complicates the controller design slightly. This configuration also delays the time required to synchronize the clock domain since any

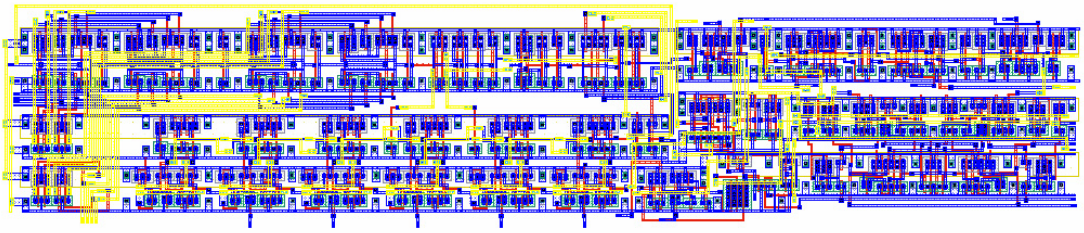

Figure 3.7: Circuit layout of *n+1* delay line tap.

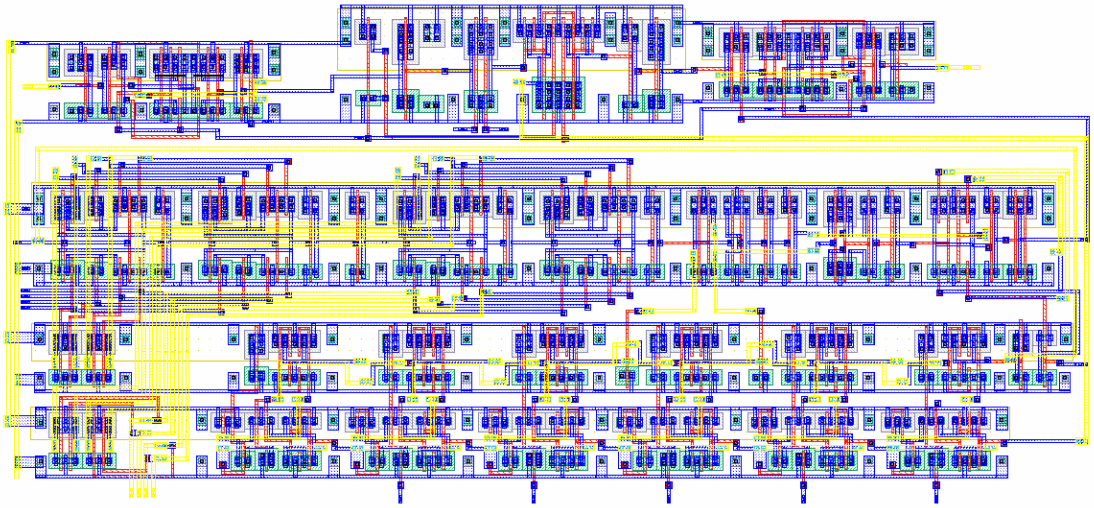

Figure 3.8: Circuit layout of *n+1* delay line clock selector.

change to the source clock must be allowed to propagate to the tap being synchronized between every test. The source clock for the first *2n* method does not change, so the controller needs only to wait for the forward clock to propagate to the next tap between tap synchronizations. For the *n+1* configuration, the matching between delay lines can be affected by intra-die process variation. However, because the source delay line is shared amongst all taps, the amount of skew generated between taps will be proportional to the variance between all of the dedicated delay lines. This is a single instance of variation, which is acceptable when one considers that a conventional tree-distribution has buffers scattered throughout the IC and skew will accumulate as the clock propagates through the IC.

3.4.3. Architecture with *n* delay lines

The third configuration eliminates all the delay lines not required during operation and eliminates the need to match delay lines by using a single delay line at each tap to perform the required averaging. Since the same delay line is re-used twice to perform the averaging step, there is no device mismatch error between the delay lines creating each half delay, making this architecture tolerant to process variation. The delay line is modified to prevent signal races by converting the 50% duty cycle reference clock to a pulse for the clock synchronization phase and a multiplexer is added to choose between the forward reference clock and the feedback clock at the delay line input. The delay line requires additional circuitry to control the input multiplexer and the pulse generator. The

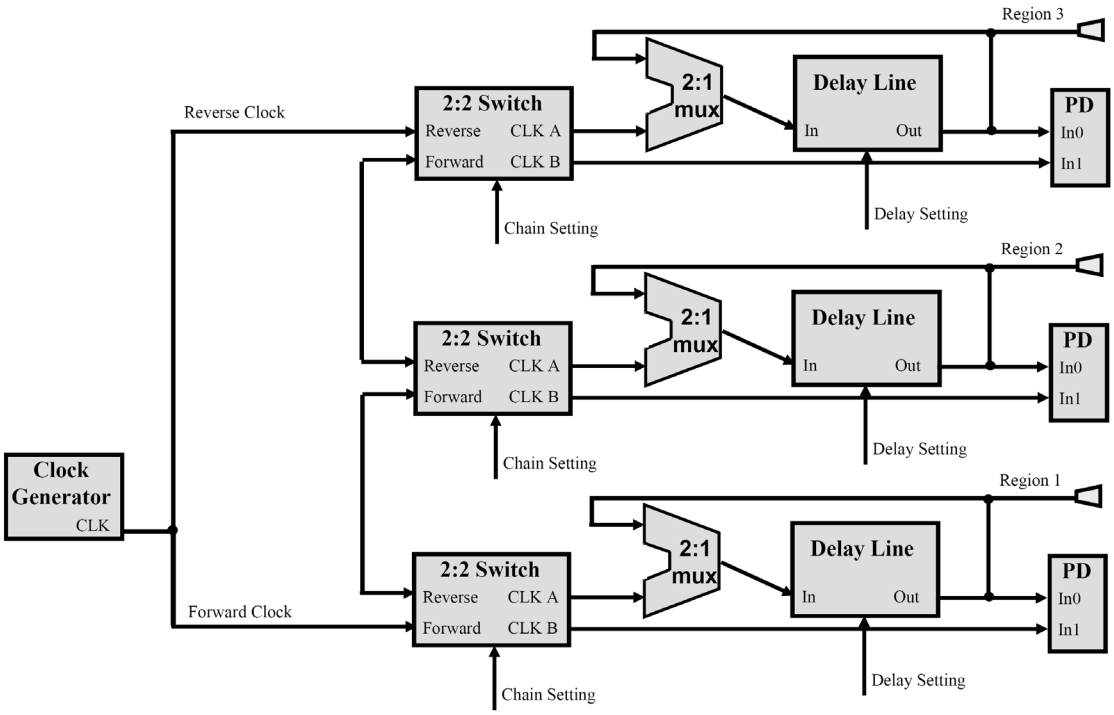

Figure 3.9: Architecture using *n* delay lines.

circuitry is designed to operate autonomously and asynchronously using signal transitions as cues. Since the rising edges are synchronized, the system modifies only the falling edge so the rising edge follows an identical signal path for synchronization and operation. These changes simplify the design of the phase detector, since the reset condition of the detector where both inputs are zero is longer than the previous cases, simplifying the detection when the reference clocks are out of phase by 180°. This architecture, shown in Figure 3.9, contains the best features of the first two variants. It will synchronize as quickly as the first method with less area overhead. Including the clock to pulse conversion circuitry and controller, each shared delay line tap requires 5100 μm^2 of area. The layout is shown in Figure 3.10. One other important benefit is that both delay lines are identically loaded.

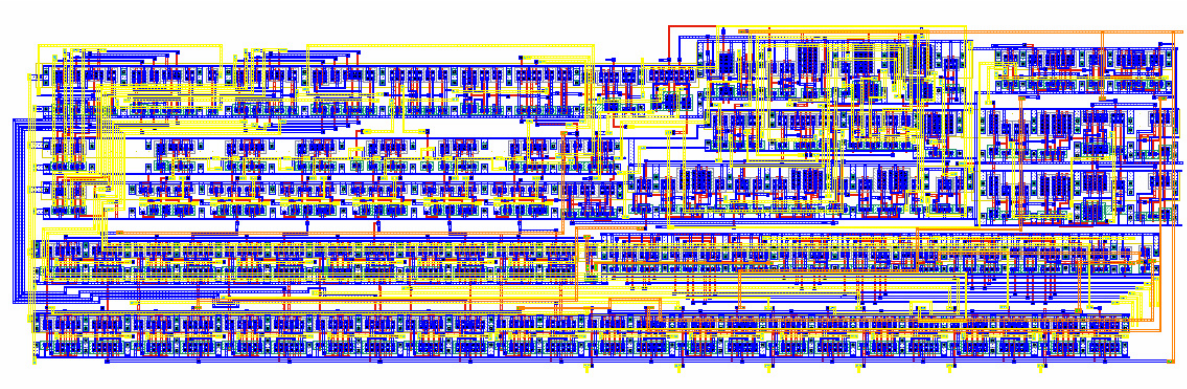

Figure 3.10: Circuit layout of *n* delay line tap.

3.4.4. Hotspot tolerant architecture

Once synchronized, temperature hot-spots can shift, arise or disappear in different areas of the IC. Device delay is significantly affected by temperature so it is important to synchronize the system at the correct operating conditions. If operating conditions shift during operation, the system will need to re-synchronize the clock taps at run-time. If pausing the complete system is not possible, the architecture will require a shift from a bi-directional clock line to a dual reference line based clock network to perform the averaging.

The system can re-synchronize without pausing by controlling changes to the delay setting to prevent glitches in the local clock, or shortened clock pulses [83] and preserving sufficient high and low times. On-line skew corrections are effective when all taps were previously synchronized and require fine adjustment around their initial synchronization point. Two reference signal lines are needed since every tap needs to

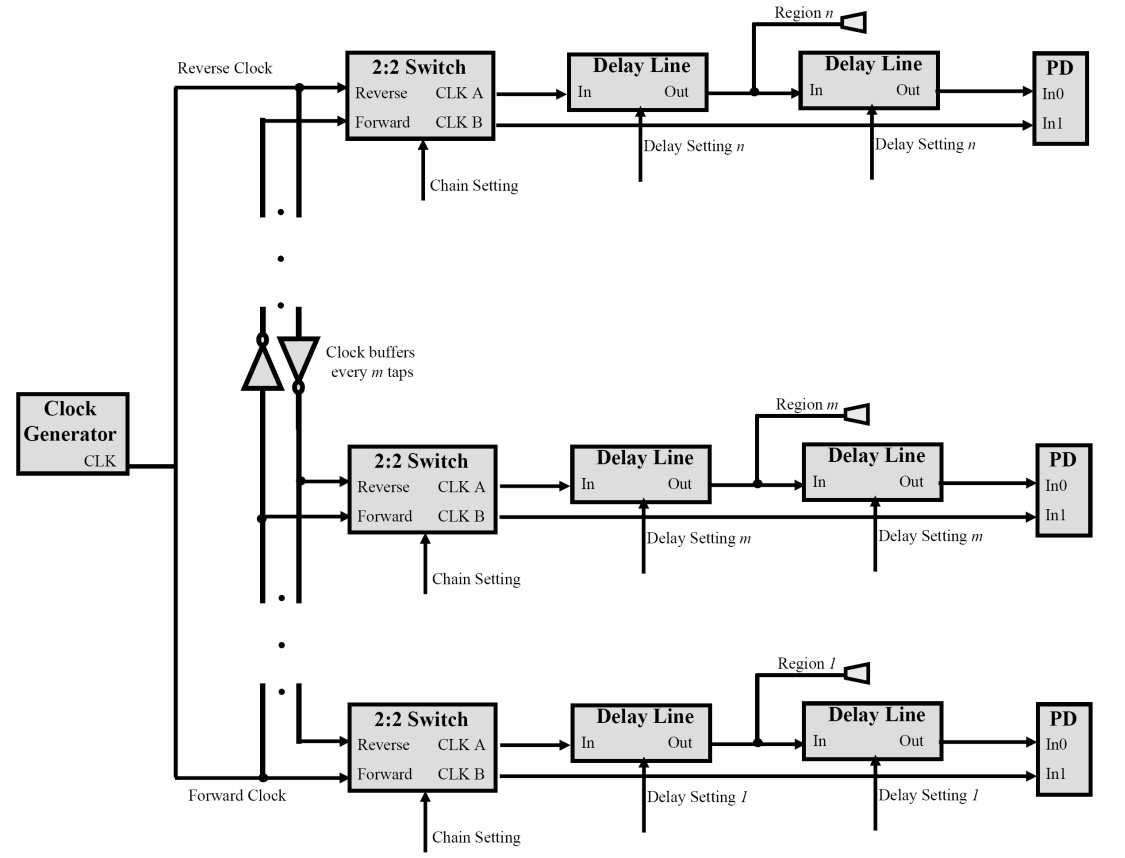

Figure 3.11: Dual reference line hot-spot tolerant configuration.

simultaneously have access to both the forward and reverse reference signals. This configuration is shown in Figure 3.11.

3.5 Clock jitter and skew.

Clock distribution networks must overcome two significant issues that can undermine their effectiveness if ignored. The first is the issue of jitter and the second is the issue of clock skew. Jitter behaviour can change quickly from cycle-to-cycle or slowly over many thousand clock cycles.

3.5.1. Jitter sources

Traditionally, cycle-to-cycle jitter is the most dangerous since it is the most difficult to predict. A late clock edge followed by an early clock edge can reduce the effective clock period significantly enough to cause errors. The primary source of clock jitter is power supply noise cause by switching activity [87],[88]. When a large number of transistors switch simultaneously, noise is generated on the power and ground signals and this noise can alter the delay of the devices near the perturbation. By their architecture, clock trees are a source for clock jitter since large capacitances are simultaneously switched at every buffering instance.

Jitter can accumulate through each buffering level. Since the noise level will vary depending on the switching activity within a particular region, the jitter across the IC will also vary. While the components along the clock distribution path are susceptible to power supply fluctuations, the analog circuits traditionally used to generate the global clocks in a system are significantly more sensitive to noise [89]. These circuits must be designed carefully to reduce jitter levels through techniques such as having dedicated power signals and carefully placed guardbands. These techniques will also be beneficial for generating global clocks for our dual reference signal design. However, our serial distribution will have less impact on the power distribution system, similar to using a spread spectrum approach to the global distribution. Since clock signals toggle at twice the frequency as normal signals, they both generate and are susceptible to significant noise, so similar to currently employed methods, adding a dedicated clock power line

and guarding clock lines against coupling capacitance effects can only positively affect clock jitter. As jitter accumulates through a number of different stages, the total jitter will decrease and be bounded by the random nature of the jitter sources [90].

3.5.2. Skew

Clock jitter has traditionally been a concern with clocks on an IC because of the analog PLL and DLL components that are traditionally used to generate the global clocks. However in deep sub-micron technologies, device and interconnect variance is leading to an ever-increasing amount of fixed uncertainty that must be addressed [91]. Inter-die mismatch and intra-die mismatch are two general kinds of mismatch that will prevent no two dies from being identical [55]. The first variety affects all devices on a die equally, and does not pose significant issues when dealing with clock distribution. The effects of intra-die mismatch can be mitigated somewhat by using larger devices and placing them as close together as possible. These observations have been summarized by Pelgrom et al.'s relation for variance due to parameter (*P*) deviation [54]:

$$\sigma^2(\Delta P) = \frac{A_p^2}{WL} + S_p^2 D^2 \qquad (3.4).$$

Here, the discrete variances are grouped together with the A_P^2 term whose influence decreases as the transistor sizes involved increase. The proximity variances are modeled by the S_P^2 term whose influence increases as the distance between the devices increase. A_P^2 and S_P^2 wind up being technology dependent constants [54],[92],[93]. This

relation is a first order model for process variance reflecting trends in the behaviour of devices and interconnects. With deep submicron technologies, these grouped terms have been elaborated to better reflect the sources of variance. In addition, the transistor sizes *W* and *L* have been updated to reflect the effective size of the transistors (L_{eff} and W_{eff}) [94]. Higher order models will lead to more accurate results, but further study would improve them further still.

3.5.2.1. Effect of skew on an H-tree

Mismatch is an important factor in clock distribution networks since clock buffers are scattered throughout a device and each may exhibit different operating characteristics [56]. The potential skew will increase as leaves become further apart and centroid configurations are not possible with distributed clock buffers. As such, the worst case skew will be present between the two leaf nodes that are furthest from one another. If 4^N represents the number of leaves in the system and the smallest distance between clock leaves in an H-tree clock network is *x*, the distance between the first pair of diagonal clock drivers in the clock tree is:

$$D^2 = 2\left(2^{N-1}x\right)^2 \qquad (3.5).$$

Assuming that every clock buffer will drive four identical loads, *N-1* sets of different clock buffers will be traversed while distributing a clock from source to leaf. The total variance will accumulate as the clock signals propagate through each level of buffering. The

worst-case distance between diagonally opposite clock drivers at any given level will continue to increase causing greater mismatch through every subsequent tree level. The total variance through a 4^N clock tree will be:

$$\sigma^2_{tree}(\Delta P) = \frac{(N-1)\cdot A_p^2}{WL} + S_p^2 \sum_{j=2}^{N} 2\left(\sum_{i=2}^{j}\left(2^{i-1}x\right)\right)^2 \qquad (3.6).$$

The discrete variance is identical for every level of buffering and the proximity variance increases as the devices get further from the clock source.

3.5.2.2. Effect of skew on a serial architecture

Our serial architecture will have many more stages, but each stage is separated by a much smaller, fixed distance. Along the distribution path, the only mismatch that does not get eliminated is the one between the clock drivers which drive the forward and reverse reference signals on a single bi-directional line between adjacent taps. Assuming the same distance x between taps, and an identical 4^N taps (4^N-1 connection segments) to the previous equation, the total variance for a serial clock network is:

$$\sigma^2_{linear}(\Delta P) = \frac{(4-1)^N \cdot A_p^2}{WL} + (4^N - 1)\cdot S_p^2 \cdot x^2 \qquad (3.7).$$

Assuming that the discrete component's effect can be mitigated by using sufficiently large transistors, let us examine the effect of the proximity dependent component. The value of the summed D^2 component of Equation 3.4 is displayed in Table 3.2 for various

values of *N*. This result shows that the serial clock distribution network has better proximity variance levels due to quadratic dependence on distance.

Number of 4^N tree levels	H-tree distribution	Serial Distribution
1	0	$3x^2$
2	$8x^2$	$15x^2$
3	$104x^2$	$63x^2$
4	$808x^2$	$255x^2$
5	$5032x^2$	$1023x^2$
6	$27816x^2$	$4095x^2$
7	$143016x^2$	$16383x^2$
8	$701096x^2$	$65535x^2$
9	$3324584x^2$	$262143x^2$
10	$15387304x^2$	$1048575x^2$

Table 3.2: Sum of squares distance component of variance relation.

3.5.2.3. Centroid layout

Eliminating the directional circuitry at each tap for the serial configuration in Figure 3.11 allows a clock buffer to propagate a clock to *m* sequential taps before requiring regeneration. By placing clock buffers and switches next to each other between clock taps and using common centroid layout techniques, it is possible to practically eliminate all proximity induced variation. Common centroid layout is a typical approach for minimizing mismatch for constant process variance gradients by decreasing proximity mismatch by a significant factor. Equation 3.8 extends Pelgrom's relation for common centroid configurations.

$$\sigma^2{}_{centroid}(\Delta P) = \frac{A_p^2}{WL} + \frac{S_p^2 D_x^2 D_y^2}{12 D_w^2} \qquad (3.8)$$

where D_x and D_y are the horizontal and vertical distances between devices and D_w is the wafer diameter [95]. In practice, however, process gradients are not perfect planes meaning that there will always be a small mismatch component present. Centroid layout are not possible for broadcasted clocks so this is not a technique which can work for eliminating mismatch in the vast majority of clock distribution networks. The proximity of matched components will also lessen the negative effect of hot-spots on the system [58]. The total mismatch of a dual reference line system with centroid layout is shown in Equation 3.9:

$$\sigma^2{}_{linear-centroid}(\Delta P) \cong \frac{A_p^2 \cdot 4^N}{mWL} \qquad (3.9).$$

3.5.3. Temperature variation

These results reflect fixed manufacturing variation, but there are other transient sources of skew such as temperature and power supply defects. In particular, increased power density in integrated circuit have caused cross-die temperature gradients, or so called “hot spots” to have a significant effect on transistor behaviour, both with respect to interconnect delays which can change about 20% with a 75°C variation from ambient and for device delays that can increase by 50% with a temperature shift of 75°C in a modern IC [87]. Our serial distribution will compensate for intra-die temperature gradients present during synchronization. In our serial network, there can be a difference in temperature for the clock drivers located at adjacent taps, but the change will be small

when compared to that present between distributed drivers in a tree network. Our averaging system is unique when compared to similar systems since it uses a pair of delay lines or a single delay line twice to establish the midpoint between reference clocks. The clock buffers in these configurations are located in relatively close proximity to each other, thereby exhibiting similar temperature and power supply characteristics. Even if the operating characteristics of the delay lines change, as long as the two delays are well matched, the average will still be accurate. Using the *n* delay line architecture aids in guaranteeing this.

3.5.3.1. Effect of temperature-induced variation on dual reference line system

The configuration shown in Figure 3.11 achieves a much greater tolerance to hot-spots and mismatch variation in devices due to the proximity of devices that require matching. It will also permit higher clock frequencies due to a potential decrease in the loading each segment of the serial network. This dual line configuration eliminates the buffer spacing component between taps from the variance calculation by switching from a single bi-directional clock line to a dual reference line approach. As will be shown, this move will result in a significant decrease in device mismatch, but could result in a greater wire mismatch so co-locating the forward and reverse reference clock wires and ensuring complementary behaviour between taps is essential in maintaining good clock characteristics. This is a cost-effective approach to skew compensation when compared

with the tree-based techniques such as [66] due to the inherent wire-savings achieved by a serial clock distribution

3.5.4. Dynamic operation

Once synchronized, a reference-based clock network may undergo environmental changes that require resynchronization. We suggest here three possibilities for resynchronization: *periodic*, *on-demand* or *polled*. A periodic resynchronization can be triggered after a user-set period of time. The system can resynchronize each tap sequentially like the initial pre-operation synchronization, however the period synchronization will be significantly quicker since it only requires small adjustments to the previous delay setting. Resynchronization here would only require a delay in the tens of clock cycles per tap. The second is an on-demand approach that requires the inclusion of a dummy tap at the end of the serial distribution. This tap will always have access to both forward and reverse clocks so it can use its phase detector, likely a variable tolerance phase detector (Figure 7.17), to monitor alignment. The forward clock passes through every tap in the clock domain and is affected by all the environmental fluctuations along the clock thread, but the reverse clock will not deviate significantly from its initial position since it follows a different and much shorter path from the clock root in the design. If either the forward or reverse clock shifts, an indication of environmental changes, the phase detector could trigger a thread-wide resynchronization. An *n* delay line architecture can be used if the clock network is

paused while resynchronizing. A third approach would be to poll each tap continuously and sequentially, and resynchronize them during operation. This method requires the use of the dual reference line architecture, Figure 3.11. This method would not require any additional hardware at the taps, but would need the synchronization controller to be on-chip. With the availability of both the forward and reverse reference signals at all times for every tap using this configuration, the controller can adjust each tap delay as necessary to compensate for (long-term) temperature and voltage related changes in the delay behaviour of the system. Each synchronization here would not require a significant amount of time since the system would only require small changes in the delay line setting. A tap can be re-synchronized on the fly without pausing using a *2n* delay line architecture. The dual reference line strategy, combined to our averaging approach makes this system highly tolerant to process variance, device mismatch and cross-die temperature variations including hot-spots.

3.6. Controller requirements

There are many approaches to designing a control system for our reference-based clocking system. A controller needs to read the phase detector output and modify the delay settings for each delay line and the 2-bit direction control of each clock routing switch. These control lines will change in a regular pattern, with a *REVERSE* direction initially, a *SYNCHRONIZATION* state next, and a *FORWARD* direction finally. By propagating the control signals produced by the phase detector and not the reference

clocks themselves, our solution eliminates another source of skew when compared to other solutions like [66]. The controller can be constructed in the hardware itself, or externally on an FPGA, microprocessor or other controller. The phase detector is designed to indicate when the delay line setting is correct, too slow or too fast.

3.6.1 Synchronization time

The time required for synchronization will depend on the architecture being used. The *n+1* method is the most complicated configuration and will be examined first. The lock time for each clock tap is a function of its proximity to the clock source. The time for each test can include up to 4 extra clock periods: two for the worst-case delay before both forward and reverse rising edges arrive, one for the additional delay that can be injected into the path from the *source* and *local* delay lines and one more for the resolution time of the phase detector. The worst-case time for each test is:

$$Time = 4 \cdot T + \sum_{i=1}^{j} z_i \qquad (3.10)$$

where j is the tap number, z_1 is the time-of-flight from the source to tap 1, z_i is the forward path delay from the input of tap $i-1$ to the input of tap i and T is the longest possible clock period. The reverse reference signal root-to-tap delay does not affect the lock time since it is not modified during synchronization.

The synchronization delay is implementation-dependent since it is related to the structure of the variable delay line. The following discussion reflects our delay line, but a

similar discussion can be made for any implementation. Our delay line contains fine and coarse components. One possible run-time synchronization approach would be to begin with the closest tap to the clock source, verify each coarse grain setting from fastest to slowest, using the longest fine grain setting for each test. If *C* represents the number of unique coarse delay settings, the proper coarse setting can be found in a maximum of *c=C-1* tests. To synchronize the fine delay, an appropriate strategy is to use a *binary search* to traverse the fine delay settings. If *F* represents the number of fine settings, a maximum of *f=a* fine tests are needed, where 2^a+1 represents the first integer greater than or equal to *F*. The worst-case synchronization time will occur at the minimum clock frequency that can be found by:

$$Time = \sum_{j=1}^{n} x \cdot \left[4 \cdot T + \sum_{i=1}^{j} z_i \right] \qquad (3.11)$$

where *n* is the number of taps, *x=c+f* is the maximum number of tests per tap and the other variables are defined for Equation 3.10.

To find the average synchronization delay for the mid-point frequency, we assume that each test requires an average of *c* coarse and *f* fine tests. There are 2^a-F-1 fine settings in all, *a* tests unused and the shortest tests are preferred for the remaining settings. The average number of tests needed per tap is:

$$x = c + f = \frac{C^2 + C - 2}{2C} + \frac{\left(\sum_{k=0}^{a-3} 2^k (k+1) \right) + a(F \bmod 2^{a-1} + 1)}{F} \qquad (3.12).$$

The average case delay assuming that each tap reaches a *locked* state is thus:

$$Time = \sum_{j=1}^{n} x \cdot \left[2.5 \cdot T + \sum_{i=1}^{j} z_i \right] \qquad (3.13)$$

using *x* from Equation 3.12. The 2.5 extra clock periods (instead of 4) are divided as follows: one-half to account for the average delay due to the *source* and *local* delay lines, one for the average delay for both forward and reverse edges to arrive at the phase detector and one more to account for the resolution time at the phase detector.

For the *n* and *2n* cases, the synchronization time need not take into account the source-to-tap delays for every test. As such, the worst case for each test is simply:

$$Time = 4 \cdot T \qquad (3.14)$$

where T is the longest possible clock period. The worst-case synchronization time will occur at the minimum clock frequency and can be found by:

$$Time = \sum_{i=1}^{n} x \cdot 4 \cdot T \qquad (3.15).$$

The average number of tests is the same as Equation 3.12. The average case delay assuming that each tap reaches a “locked” state is thus:

$$Time = z_j + \sum_{j=1}^{n} x \cdot 2.5 \cdot T \qquad (3.16)$$

where the forward path delay between taps is z_j.

Using the same synchronization approach as discussed above, another alternative would be to do a linear search for the fine grain setting, starting with the slowest (longest) setting and trying each faster setting in turn until the correct one is

found. In this case, the worst case synchronization times would be the same as above, except the worst case number of fine grain tests would be *F* and the average number of fine grain tests would be *F/2*. The equations would need to be modified for a different delay line, but the underlying analysis approach would remain valid. The initial synchronization could also be sped up by programming the system with cached expected delay data

A post-synchronization calibration phase is needed to obtain the correct polarity of each local clock. This step can be performed using a multiplexer to select between an inverted and a non-inverted signal at each tap without modifying the overall delay of each clock path. This step is quick because the source delay line is no longer used at this point – modifying the polarity simply involves changing a multiplexer setting at the clock tap. This step can be done in four clock cycles per tap: one to measure, one to toggle and two to propagate the current tap's polarity to the next tap. Our method can only be used at the frequency it was calibrated with so in dynamic frequency scaling applications, the system must be recalibrated each time the frequency is changed. However, each desired operating frequency could be synchronized in advance with the required delay settings stored in memory. Thus, the delay settings could be changed quickly to align the tap clocks.

3.7. Simulation results

This single clock averaging clock distribution discussed here has been designed for TSMC's 0.18 μm P-well process using the Cadence Virtuoso design environment and simulated with SpectreS and the Analog Artist simulation tool. Using an *n+1* architecture, the schematic circuit simulated as follows. For each delay line, five alternate coarse grain paths provides a 93.5 ps delay increment over the base setting and the fine grain delay provides and additional 93.5 ps delay, resulting in a total delay of 1122 ps considering the positive and negative polarity signals available at the output. The minimum clock frequency that can be used is 445 MHz, equal to twice the maximum delay through the delay line. The extracted simulations show that the coarse grain delay increment is 80 ps, resulting a total net delay of 960 ps and a minimum clock frequency of 521 MHz. The resolution of the phase detector is +/- 1.5 ps. The error between any two taps can be up to the sum of twice the phase detector error and twice the maximum delay increment. Since one of the variable delay elements are removed during run-time, the net skew will be half this amount. Thus, the expected skew of the system is 6.25 ps. In practice, the expected skew is actually approximately 10 ps using this delay line because of the duty cycle changes that occur through the fine delay line.

The clock distribution circuitry for the extracted system is capable of routing frequencies up to 1.90 GHz, corresponding to periods of 525 ps and higher. The limiting factor is the clock routing switch that drives the large capacitive load bi-directional clock. Should higher performance be required, these nodes can be designed with larger

devices to provide faster transition times. The remaining circuitry is capable of operating at up to 2.12 GHz (470 ps periods). For an *n+1* system, the worst-case synchronization delay is 1587.9 ns, or 708 clock cycles from Equation 3.15. The average synchronization delay is 347.9 ns, or 408 clock cycles from Equation 3.16.

Figure 3.12 shows a simulation at 1.90 GHz of how each forward clock needs to be synchronized with the reverse reference clock. The forward clock in Tap 0 (*CLA_0*) is aligned to the reverse clock (*CLB_0*) between 10 and 20 ns, the forward clock in Tap 1 (*CLA_1*) is aligned to the reverse clock (*CLB_1*) between 20 and 30 and so on. Once the correct delay line setting for every tap is determined, the forward clock can be set to bypass the delay line at clock source to save power in the *n+1* configuration clock network. A maximum skew of under 10 ps was achieved in this case. For clarity, only the final synchronized delay setting is shown. Given that *CLK 0* to *CLK 8* represent each tap's local clock, Figure 3.13 shows that clocks 0, 4, 5, 6 and 7 all have positive polarity, while clocks 1, 2 and 3 have negative polarity at the end of the synchronization stage (at 90 ns). Polarity, in this case, is always taken with respect to clock 0. Figure 3.13 also shows the calibration process for the 8-tap design with *CLK 1* being inverted at 98 ns, *CLK 2* being inverted at 101 ns and *CLK 3* being inverted at 104 ns. Figure 3.14 shows the resulting polarity-adjusted clocks for the 8-tap distribution network. Figures 3.13 and 3.14 use an 891 MHz reference clock. Comparable solutions offer similar or worse levels of skew reduction, sub-10 ps for [96], 28 ps for [76], 70 ps for [67] and 15 ps for [62]. [97] demonstrates a skew reduction scheme capable of reducing skew to within 10% of the clock period, versus under 4% here. Work in [66] ideally achieves 3 ps skew resolution,

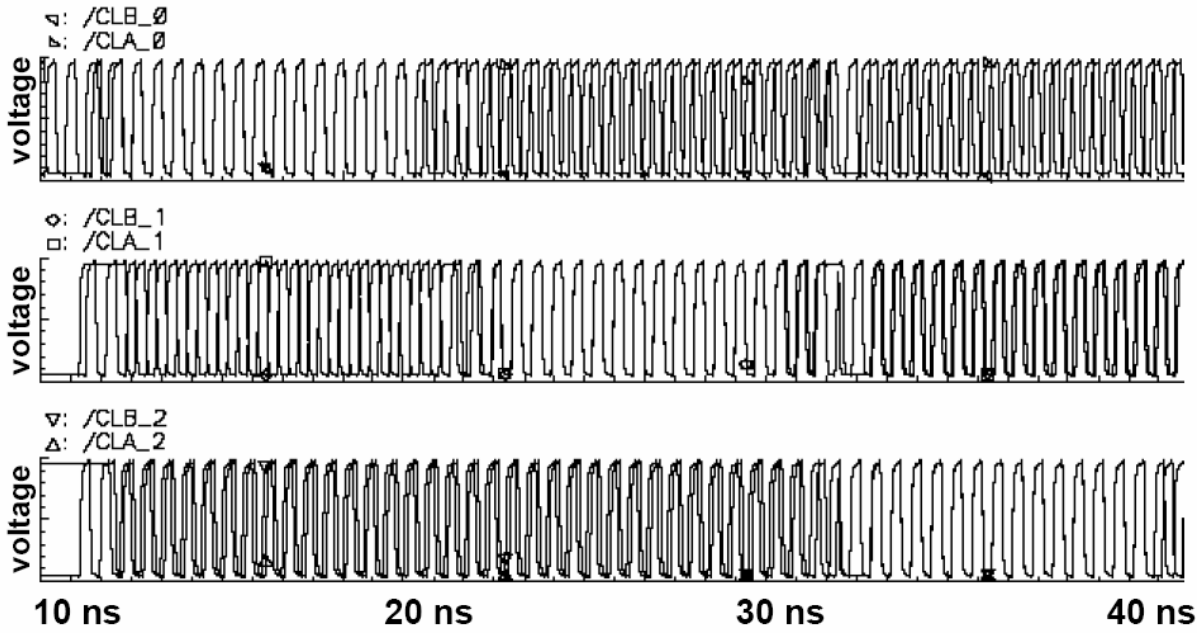

Figure 3.12: First 3 synchronization steps for an 8-tap CDN.

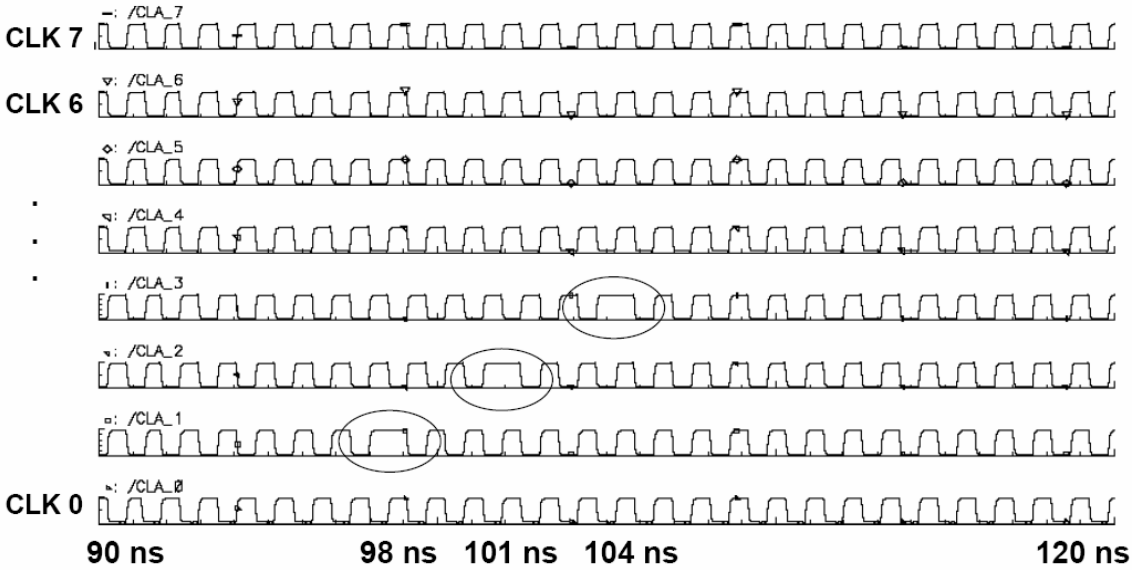

Figure 3.13: Calibration phase to align polarity of resulting clocks.

but is susceptible to intra-die variation, modifying a traditional H-tree distribution and requiring a duplicate co-located return path for all the leaves in the network also entails an area overhead.

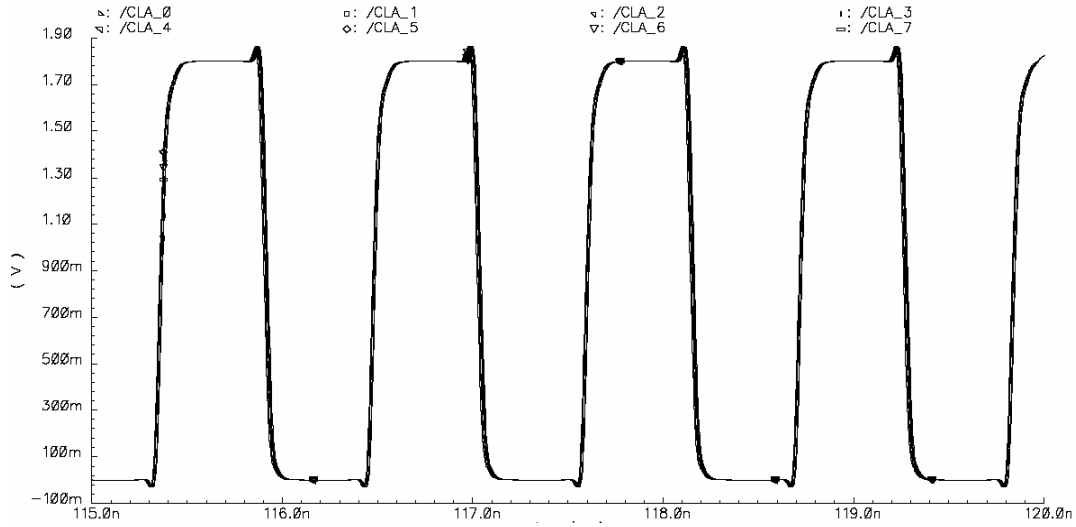

Figure 3.14: Resulting low-skew output clocks for an 8-tap system.

Once the polarity detection is complete, the calibration and phase detection circuitry is turned off, resulting in significant power savings at runtime. Simulations show that for a typical forward and reverse clock skew setting, power consumption for an 8-tap clock distribution circuit at the maximum frequency is 33.2 milliwatts (mW) during the phase alignment cycle and 18.0 mW during run time. At 891 MHz (the minimum frequency of operation without considering the inversion capability of the delay line), the power consumption during phase alignment is 18.6 mW and 9.97 mW during run time. While one would expect quadrupled power consumption for doubling the frequency, the energy consumption is somewhat smaller than expected at higher frequencies because shorter paths are taken through the coarse grain delay line. These power consumption numbers are much better than PLL based solutions that typically consume hundreds of

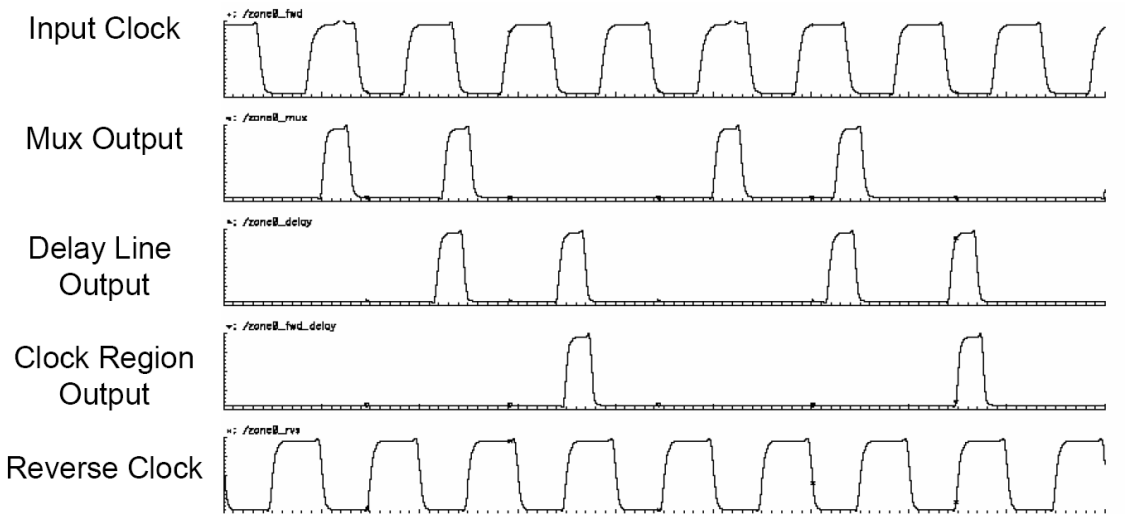

Figure 3.15: Using pulses to align clocks in *n* delay line architecture.

mW [65]. In comparison, [98] demonstrates 0.21 mW power consumption per de-skew tap for a 56 ps skew bound.

Tests on a laid out and extracted 4-tap single clock domain *n* delay line architecture were performed using a 1 GHz sample clock. Figure 3.15 shows how the pulses are used to align the clock region output and the reverse clock rising edges. The performance of this system mirrored that of the other architectures, with an overall skew bound of 12 ps and power consumption of 2.5 mW per tap at 1 GHz.

3.8. Conclusion

Traditionally, most integrated circuits operated using a single global clock. Great care was needed to ensure that the clock had good characteristics and low skew. Today, most clock networks are designed using CAD tools which require precise

information on the exact clock load for each branch, the placement of each tap on the die and the location of the clock root. Once generated to satisfy the required skew and latency metrics using wire and driver re-sizing and placement, the clock network cannot be altered without affecting clock skew. Our dual reference signal clock network allows designers to delay some of the critical clock tuning requirements to facilitate the design flow. It allows circuit blocks to be moved around conveniently and re-sized easily with a simple change in the number or location of the taps. Our cell based approach to clock distribution allows components to be designed independently, connecting components as is convenient and even replacing blocks if needed. The presence of the digitally programmable delay lines allows the system to accommodate blocks with different tree depths and latencies.

We have designed a clock network with multi-point active skew compensation. By using delay lines instead of PLLs and with a single reference and distribution line, our method is small enough to be useful for many clock applications. Using a single common forward and reverse clock reference line to cut down on in-die process variation skew is unique. Using a dual reference signal approach places all critical devices in close proximity, minimizes process variation and permits online clock resynchronization which can help eliminate intra-die temperature deviation. Using a daisy-chained approach minimizes the total clock line length and thus the required clock load and the power consumption can be reduced. The system can also be used to provide beneficial skew between each of the local taps in the IC [62]. Any reduction in clock skew obtained by our clock distribution is extremely useful since this time can be added directly to the available

cycle time within a clock period [68]. Simulations show that the proposed CDN is scalable, compatible with irregularly-shaped distribution areas, and combines low power operation with tight skew bounds.

Chapter 4:

Skew-tolerant reconfigurable clock networks based on averaging

4.1. Introduction

Reference-based clock distributions have a number of advantages when compared to their tree-based counterparts. The modular nature of the system allows components to be easily added and modified throughout the design phase without requiring the clock distribution to be re-synthesized at every step. The ability to postpone the fine tuning of the clock distribution until after fabrication is useful and can be used to account for process variation and to correct for certain defects in the clock network.

Traditional clock distribution networks are built to match the delays from a clock root to every clock leaf in the system. These clock networks are rigid once generated and it is impossible for anything other than a single clock to be broadcast to every leaf in the domain. Some systems get around this by creating multiple clock trees and allowing the clock fed to each individual tree to be selected at the clock root. The main drawback of this approach often used with FPGAs is that the size of each clock domain is not very flexible. In addition, a system requiring many small clock domains could result in a significant waste of resources. Using our serial approach, there is no requirement for the clocks in the global distribution to be synchronized.

To perform accurate dynamic skew compensation at each regional tap, clocks travelling in opposite directions have corresponding delays between destinations and the delay lines used at average the forward and reverse clocks are well matched. This implementation allows programmable clock routing to be added into the path creating a complete fully-programmable and reconfigurable multiple clock distribution network with a fine level of granularity for the first time. Such functionality would not be possible if the system was not designed from the ground up to cope with non-aligned clocks and have a skew-tolerant approach built into the alignment system.

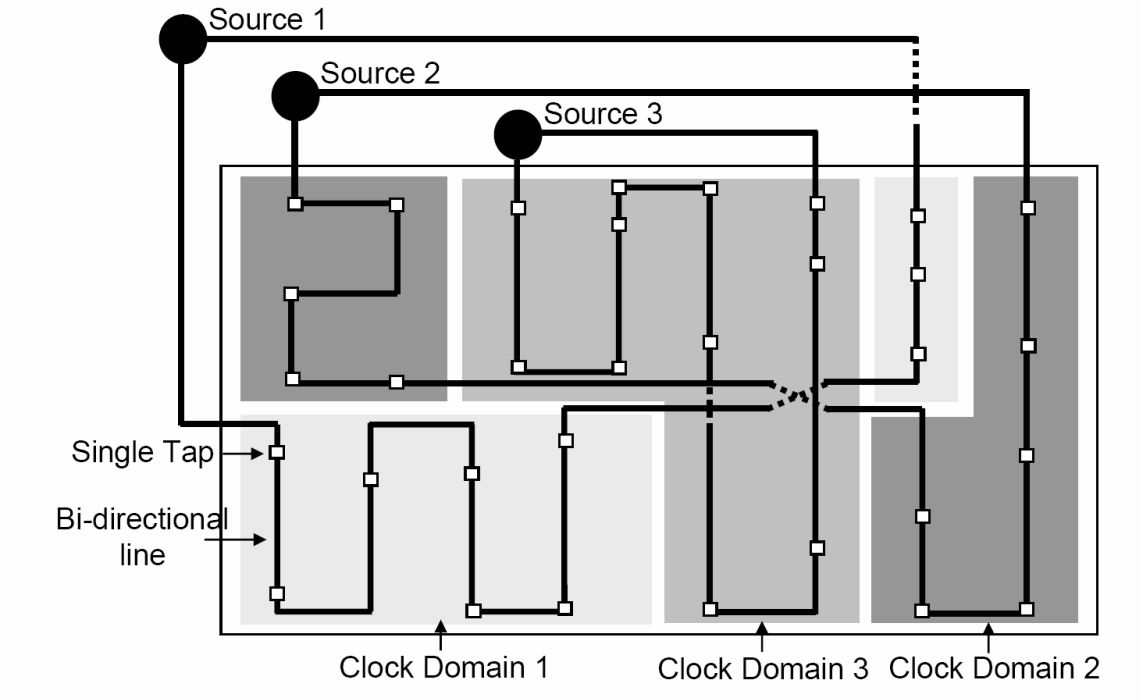

Figure 4.1: A 3-clock domain static clocking solution.

4.2. Multiple clock architectures

4.2.1 Static clock network with multiple clocks

There are multiple methods to deploy our reference-based clock network in a multiple clock environment. Multiple clocked designs with independent clock domains are common for designing large modular ASIC and SoC designs [99]. One option is to deploy multiple static clock threads, as illustrated in Figure 4.1 for an example 3-clock domain distribution. Each component in the clock thread is replicated as needed, including a global clock generator for each domain. The clock generators can either be

placed in close proximity in one area of an IC and suitably routed to and from the clock thread, or spread out over the IC. A single clock generator can also be connected to a series of dividers (or multipliers) to drive all the clock lines if the required frequencies are related by division, multiplication or other realizable operation. The only constraint is that the clock must have 50% duty cycle and be connected to both the head and the tail of the clock thread. These threads can cross other clock domains and are easy to lay out since the taps do not need to be placed at regular intervals. Irregularly-shaped clock regions, like the one shown in Figure 4.1 is much easier to implement using our method, since adjacent areas are all connected serially. Disjoint clock regions in a single domain can also be connected using a single wire with little attention paid to the wire length. This trait is convenient for pin-limited designs where it may not be desirable to locate a large sub-circuit in a given area of an IC. Small blocks can be placed near the required pins and associated to larger blocks elsewhere on the IC, while still maintaining one skew-tolerant synchronous clock domain. Similarly, clock domains can be divided to allow access to certain regional features that would be difficult to implement using an H-tree, but it is performed automatically by the design of our reference-based architecture. Each of these threads will have the same characteristics and synchronization requirements as the single clock solution previously discussed.

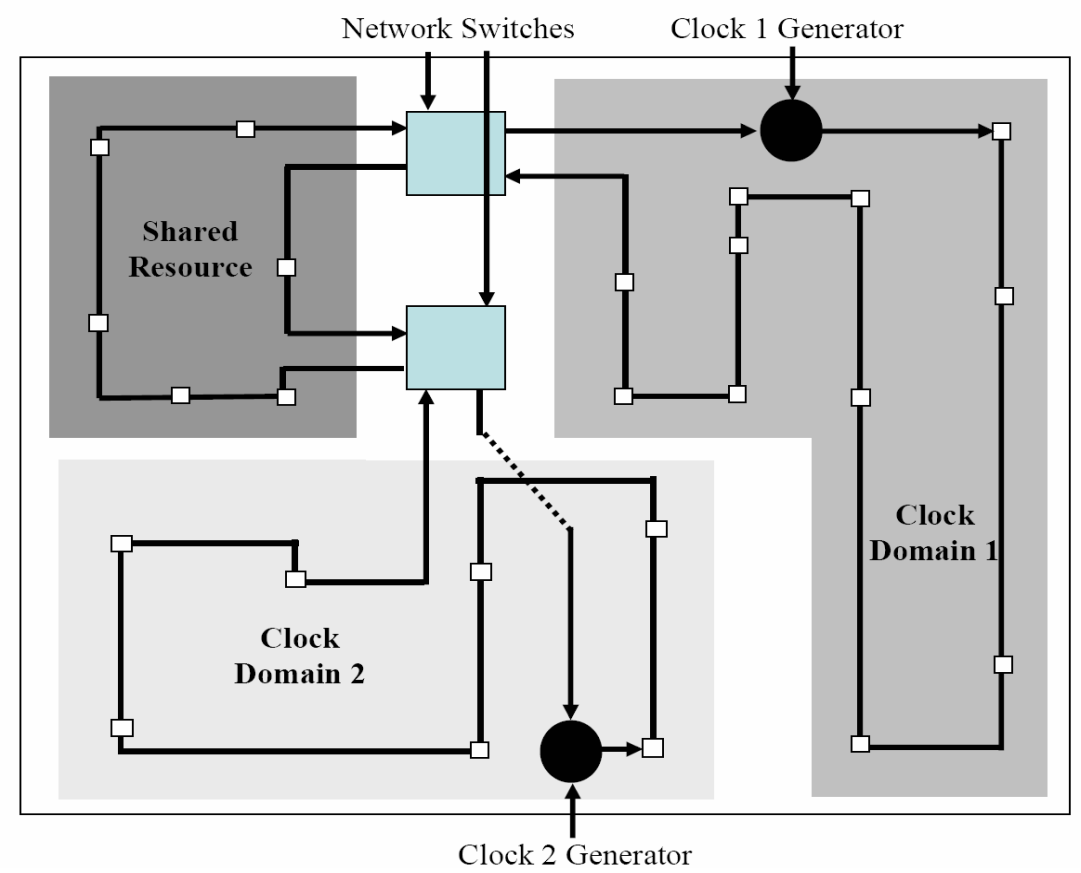

Figure 4.2: Sharing a resource with reference-based clocking.

4.2.2 Locally-reconfigurable clock network

It is possible to create a clock network with flexible clock domains by taking advantage of clock routing in the serial distribution path. In this application, the clock domains are primarily static with sub-sections that can be added or removed from a domain, or switched between domains. This post-fabrication reconfigurability in the clock network facilitates the re-use of common components for multiple clock domains and can also be used to prune areas from the clock network to save power, Figure 4.2. The

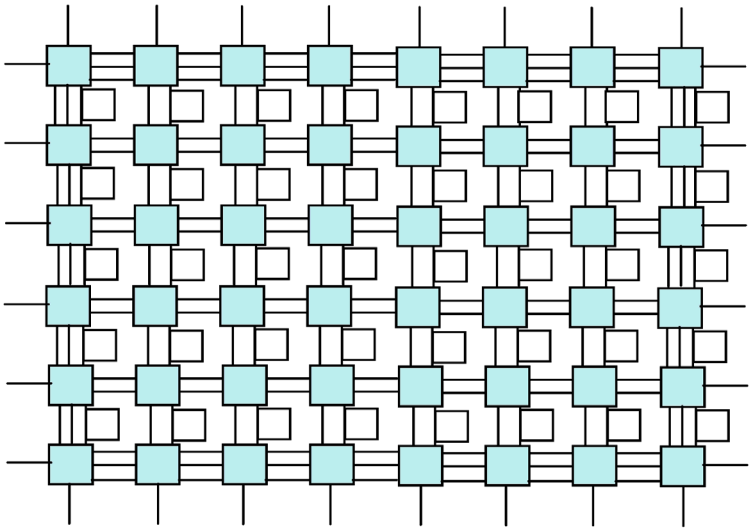

Figure 4.3: A potential fully programmable clocking architecture.

design of the network switches are discussed in Section 7.3. To use a shared resource, the domain must be synchronized with its clock thread extending through the shared device. The shared resource can be attached and detached without altering the phase or alignment of the two clock domains provided it is connected to the end of the affected clock threads. By storing the configuration data required for each mode, the transfer of a shared resource between clock domains can be made nearly instantaneously.

4.2.3. Globally-reconfigurable clock network

The approach can also be used to create a fully reconfigurable and reprogrammable clock network. Figure 4.3 shows one such network with 40-taps. The shaded squares represent switch points and the white squares represent each local clock (tap). To re-route clocks during operation, it is necessary to synchronize and calibrate each modified clock domain for the distribution to remain skew-tolerant.

Predicted delay settings can be used to configure the clock distribution network, resulting in little to no setup time for the distribution at the expense of greater clock skew.

4.3. Versatility of a programmable multiple clock mesh network

Different fully programmable clock networks can be created by inter-connecting various combinations of clock taps and clock switches. The number of taps between switches, the number of switches present and the number of ports within each switch can all be modified to tailor the clock network to the application. While Figure 4.3 shows a 40-tap solution, Figure 4.4 shows three variations on a 15-tap solution. Figures 4.4A and 4.4B both exclusively contain 8-port switches which allow up to 4 simultaneous connections; only the orientation of the ports is different. Figure 4.4C contains a mix of both 4-port switches which allow up to 2 simultaneous connections and 8-port switches. The networks are shown with only one tap on a single vertical edge between each

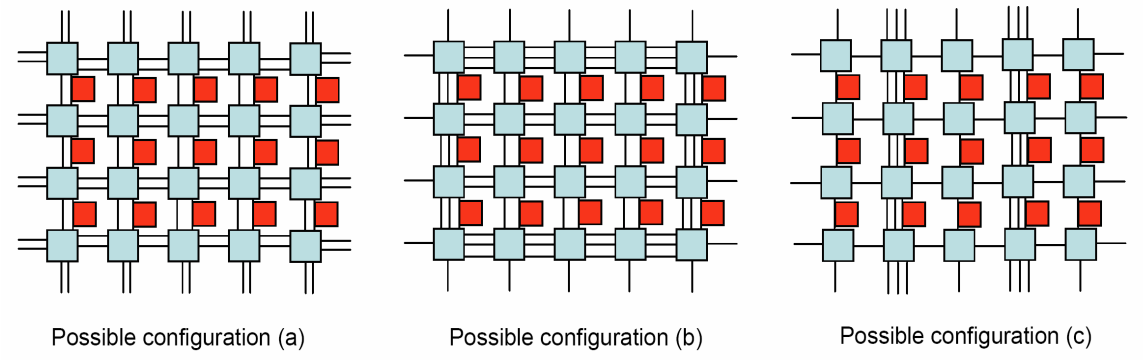

Figure 4.4: 3 potential 15-tap clock distributions.

switch, but the taps can be placed anywhere. The more ports present in a clock switch and the more clock switches present in the system, the greater the number of clock configurations are possible, but the larger the area and power penalty of the network. The number of unique configurations allowed through a *p* port crossbar switch assuming that NULL connections (unconnected ports) are not allowed is:

$$configurations(switch) = \prod_{i=1}^{p}(2i-1) \qquad (4.1).$$

The number of switch configurations possible in an *m x n* switch (rows *x* columns) network using a crossbar switches containing *p* ports is:

$$configurations(network) = \left(\prod_{i=1}^{p}(2i-1)\right)^{m \cdot n} \qquad (4.2).$$

Should the network be made up of an arbitrary assortment of switches with each switch *i* containing p_i ports, the number of possible switch configurations is:

$$configurations(network) = \prod_{i=1}^{m \cdot n}\left(\prod_{j=1}^{p_i}(2j-1)\right) \qquad (4.3).$$

This calculation assumes that every port is connected, but NULL connections are possible and will always occur in pairs. These NULL connections do not need to be dealt with independently when considering the total number of switch configurations since every unconnected state can be considered a special subset of a connected case already in the calculation. The number of unique configurations possible for a given switch counting NULL connections as unique is:

$$configurations(switch) = \sum_{i=1}^{p}\left[\binom{p}{i} \cdot \prod_{j=1}^{p}(2j-1)\right] \qquad (4.4).$$

While the number of possible network configurations for the switches is a relevant concern when designing a clock network using our methodology, a more important consideration is the number of clock domain configurations which can be generated by such structures. While clock taps can be added to any edge in the switch mesh, taps added to the edges around the mesh perimeter cannot easily be re-routed, so we limit the position of taps to the edges located between clock switches. For our mesh networks like the one shown in Figure 4.4 with taps located along a vertical edges, assuming c clock domains, the total number of configurations is:

$$configurations(taps) = \left[n \cdot (m-1)\right]^c \qquad (4.5).$$

Similarly, if we assume that there is a single tap horizontally between each switch, the total number of configurations is:

$$configurations(taps) = \left[(n-1) \cdot m\right]^c \qquad (4.6).$$

If a single tap is added between every vertical and horizontal switch, the number of configurations will be:

$$configurations(taps) = \left[2mn - n - m\right]^c \qquad (4.7).$$

Multiple taps can be added in series between a pair of switches. This approach is often desirable for grouping taps that will always be connected to the same domain. This group of taps represents a single local clock region and is conceptually no different than the single tap case and does not change the analysis. Clock taps can be added to any and all edges leaving a clock switch. However, fully populating a clock switch with taps will limit the edges available for routing, affecting the routability within the mesh network.

Assuming that all the configurations are realizable, the number of clock configurations that can be achieved using clock switches with *p* ports capable of *p/2* connections and with each perimeter switch connected to the clock generation ring by *d* ports is:

$$configurations(taps) = \left\lfloor \frac{n \cdot m \cdot p - d \cdot (2n + 2m - 4)}{2} \right\rfloor^{c} \qquad (4.8).$$

For an arbitrary mesh network, given p_i ports per switch with each switch connected to the clock generation ring by d_i ports, the number of configurations is:

$$configurations(network) = \left\lfloor \frac{1}{2} \left(\sum_{i=1}^{m \cdot n} p_i - d_i \right) \right\rfloor^{c} \qquad (4.9).$$

4.4. Controller requirements

As with the single clock network, a multiple clock distribution network requires three distinct phases to work properly: synchronization, calibration and operation. The complexity of synchronizing each thread is linearly proportional to the number of taps in each thread. Similarly, the complexity of adding another clock thread is proportional to the total number of taps, independent of the number of threads. As such, the synchronization delay analysis performed in the previous chapters can apply to each individual clock thread present in the multiple clock system.

There is a network mapping problem that must be solved to route all of the clock threads through the crossbar matrix. The crossbar matrix is an undirected graph with

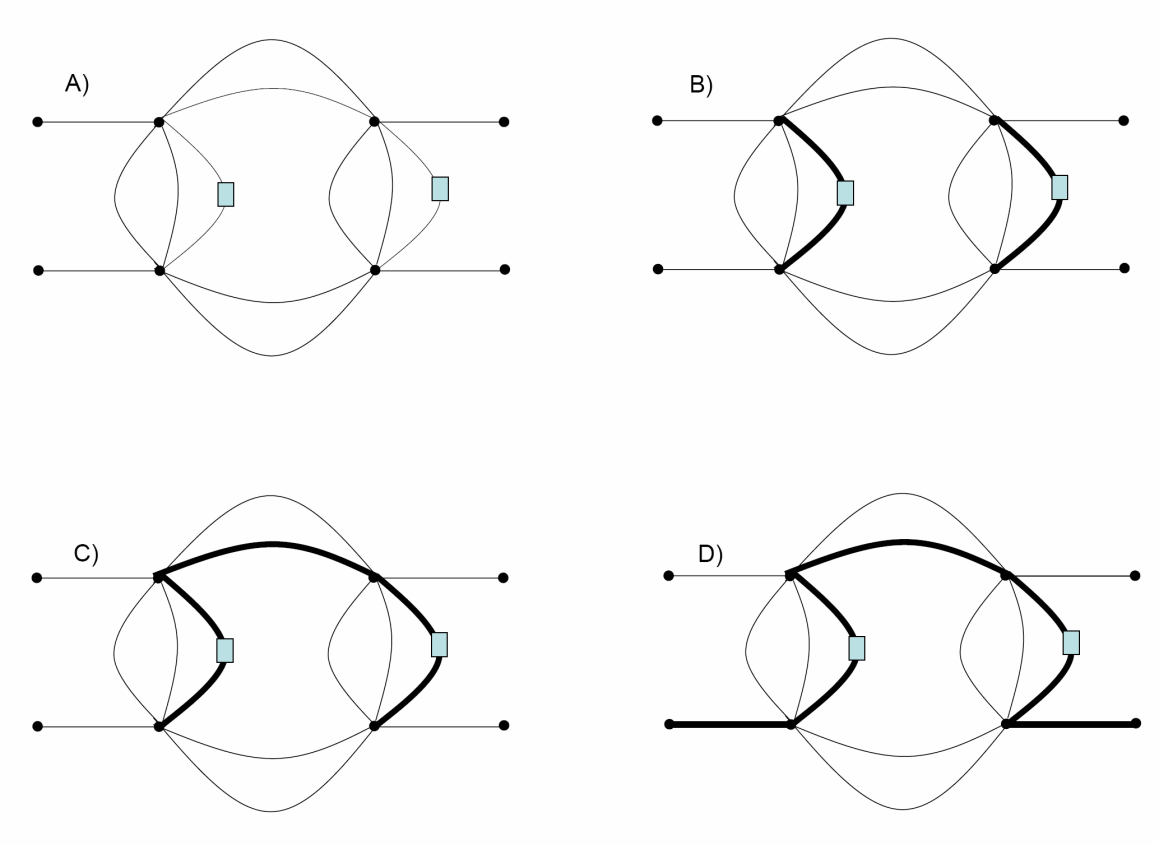

Figure 4.5: The mesh mapping problem.

multiple *edges* connecting each *vertex*. The number of edges can vary depending on the network configuration. This graph, otherwise known as a *multigraph*, is allowed to have *loops*, or feedback paths that can connect a vertex to itself. Since each clock tap is located along a specific graph edge, connecting all the taps in a clock domain together involves covering all of their associated edges and connecting these edges together. Figure 4.5A shows a 2-tap graph with 8 vertexes and 14 edges. Figure 4.5B shows the covering of the two tap edges, Figure 4.5C connects the tap edges together and Figure 4.5D shows the connection of the edges to the entry points. This problem cannot be classified as a typical problem in graph theory as the *paths* that are created are *complex* since vertices may not be uniquely traversed within the path, since the edges that must

be covered are not necessarily adjacent and since we do not need to determine an optimal solution. All clock domains must be solved simultaneously while competing together for the finite edges available in the switch mesh.

To solve the problem, our approach is to cover all edges corresponding to a tap and to start assigning paths at an arbitrary (random) location for each thread. This initial location may end up becoming a starting node, an ending node or an intermediate node since the clock thread will grow out from this initial edge. Then, we use a greedy heuristic algorithm with primary and secondary cost functions to help determine the next step. The primary cost function looks at the actual number of switches that must be crossed from the starting and terminating edge to reach every uncovered edge in the domain. At the beginning, the first arbitrary node chosen represents both the starting edge and the terminating edge. This is done for every domain simultaneously. For a tie, the path with the most remaining edges is chosen first as the secondary cost function. Once a path is established, the primary cost function is recalculated for the new state. If all connections are not possible at this point, the algorithm will backtrack one step and try with the next least costly choice.

To help alleviate the routing problem, it is possible to add express clock paths into the mesh network. This will create a hierarchical mesh network with multiple possible hops that could help transport clocks longer distances in the mesh network without causing excessive congestion and avoid potentially crowded subregions occupied by multiple clock threads. The number of and the spacing between the express paths as well

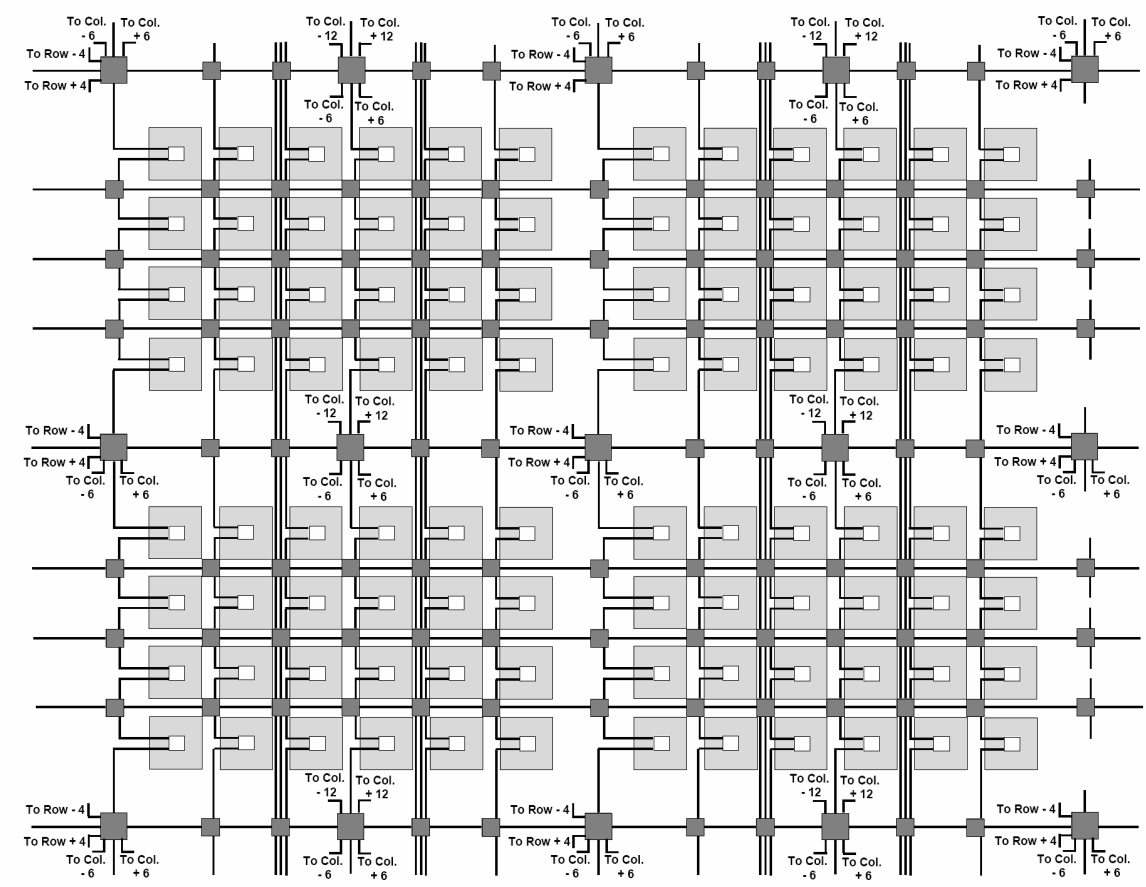

Figure 4.6: Mesh architecture incorporating express paths.

as the distance propagated between each hop on the path is flexible. These paths come at the expense of additional wire cost. Figure 4.6 shows four sections of one such possible network. The unshaded boxes represent the clock taps, and the lightly shaded boxes represent each local clock subregion. The larger of the clock switches represent the ones used for the express paths. Horizontally, there are 2-sets of express paths between each section: one spaced by 3 columns with a 6 column hop between switches and one spaced by 6 columns with a 12 column hop between switches, each creating two horizontal concurrent paths. Vertically, there is 1 extra express path per section. In total,

each section has 8 horizontal routing paths and 5 vertical routing paths in addition to the 6 vertical distribution paths.

4.5. Single clock fixed methodology

Let us examine how to implement a clock distribution using our averaging technique. We will first examine a single clock system and then a reconfigurable one. We first need to establish how many clock taps will be required by the system. Our averaging system assumes a two tiered approach to distribution, a global one using our technique, and a local one using a mesh (with shunts), tree or fishbone. A fishbone approach is the simplest to implement, but also injects the largest amount of skew between local clocks. The clock mesh benefits from near zero skew, but will also consume the most power [46]. A local clock tree is a compromise between the two. So assuming that there are F total registers that must be connected to the clock domain with each regional clock network consisting of G registers, then we will need F/G taps to distribute clocks to the entire system.

To determine an optimal number for G, we need to establish to total variance through the system. This will be the sum of the variances between the global (serial) and local portions of the clock distribution assuming the use of centroid components where possible:

$$\begin{aligned}\sigma^2_{full}(\Delta P) &= \sigma^2_{tree}(\Delta P) + \sigma^2_{linear-centroid}(\Delta P) \\ &= \frac{A_p^2 \cdot \log_4 G}{WL} + S_p^2 \sum_{j=0}^{\log_4 G - 1} \left(\sum_{i=0}^{j} 2^{i+\frac{1}{2}} x \right)^2 + \frac{A_p^2 \cdot F}{mGWL} \\ &= \frac{A_p^2}{WL} \cdot (\log_4 G + \frac{F}{mG}) + S_p^2 \sum_{j=0}^{\log_4 G - 1} \left(\sum_{i=0}^{j} 2^{i+\frac{1}{2}} x \right)^2\end{aligned} \tag{4.10}$$

If another exponential buffer multiplier other than 4 is used, the base of the log expression will need to be changed as will the squared distance relation. Recall that A_P^2 and S_P^2 are constants for a given technology, and F is the total number of clock loads and x is the largest distance between two clock loads located in the same region, which are both fixed for a given design. For a fixed W and L, there will be a maximum number of tap input loads m that can be driven. It is always beneficial to use this maximum number to minimize variance. Once a target variance for the clock distribution is chosen, G can be chosen. The optimal number will depend on the discrete and proximity variance coefficients since the serial architecture is more tolerant to proximity variance and the tree architecture contains less buffers thereby exhibiting less discrete variance. Whenever possible, smaller local trees should be chosen since trees are susceptible to temperature gradients and hotspots. This approach will also minimize the maximum distance between clock taps.

If extremely low levels of variance are required, it is possible to shift to local mesh architectures which contain negligible amounts of variance:

$$\sigma^2_{full}(\Delta P) = \sigma^2_{mesh}(\Delta P) + \sigma^2_{linear-centroid}(\Delta P) = \frac{A_p^2 \cdot F}{mGWL} \tag{4.11}$$

Finally, if large amounts of skew can be tolerated, it is possible to switch to a fishbone architecture where the fishbone skew, or the longest interconnect delay between clock loads, can be added to any skew generated through the serial portion of the clock network.

4.6. Reconfigurable methodology

To design a reference-based clock distribution for a reconfigurable clock distribution like the ones found in FPGAs, we again need to begin with the total number of clock loads, *F*, and the desired number of clock regions, *R*. Then we can calculate the required granularity of the network, *G*. Let *G* represent the number of clock loads per region and can be as little as one load, or alternatively, as many as required.

$$G = F / R \qquad (4.12)$$

We assume an *n x m* clock switch mesh for reconfiguration with each switch containing *P* ports for *P/2* clock signals. We also assume that clock taps are distributed symmetrically on the mesh network with *P/2* ports connected to clock taps except for the ports along the perimeter of the switch mesh. The number of clock regions, *R*, for such a configuration is:

$$R = \frac{nmP}{4} - \frac{(n+m)P}{4} \qquad (4.13)$$

This allows us to figure out suitable values for *n* and *m* to create an appropriate mesh network for this number of independent clock regions. To determine how many taps are

required for a given clock region, a regional skew bound can be established and a similar methodology to that used in Section 4.5 can be applied to determine an appropriate number of taps per region and clock loads per tap.

Let us apply this methodology to a recent Altera FPGA. An FPGA was chosen for the comparison due to the presence of reconfigurable clock domains and readily available data. We take for example, the Stratix IV EP4SE680 containing roughly 680k logic elements (LE) and assume a single clock load per LE. The device literature [52] states that the device is capable of 104 distinct clock domains. Assuming a square clock switch mesh ($n=m$) and 8 clock ports per switch ($P=8$). Solving for n in Equation 4.13 yields a mesh network size of 8.28, or 9 x 9 (81 total) switches with 120 regions. From Equation 4.12, the original clock distribution contains just over 6500 clock loads per region. We could either use this number of clock loads per region, leaving 16 regions unused in our switch mesh, or redistribute the clock loads evenly over the 120 regions we have. Choosing the latter results in just under 5600 clock loads per tap. Since the technology variance constants are proprietary information for the given FPGA, we will arbitrary assume 3 levels of clock buffering and 4 taps between buffers in a dual reference line serial network. This leaves us with 64 clocks per tap, 22 taps per region and 3 clock buffers (F/mG roundest to the lowest integer) per region of the local clock network. The regional skew variance for this configuration would be:

$$\sigma^2_{full}(\Delta P) = \sigma^2_{tree}(\Delta P) + \sigma^2_{linear-centroid}(\Delta P) = 6 \cdot \frac{A_p^2}{WL} + 170 \cdot S_p^2 \qquad (4.14)$$

This type of structure has many beneficial aspects for the construction of an FPGA or an ASIC, including post-silicon correction of clock skews which could otherwise cause the device to fail testing and it eases difficulties in layout by eliminating some difficult path matching constraints and allowing a standard cell based approach to clock distribution. Tests have shown that upto 88% of devices that fail could be salvaged using small changes to the circuit's critical paths [100]. Unlike H-tree distribution networks, our method combines well with a mesh network based interconnect approach used in FPGAs. [51] has shown that increasing the number of global regions in an FPGA can result in significant power savings for the device overall. Since our technique makes every region a global one, it could exploit this trend.

4.7. Simulation results

Figure 4.7 shows a typical application of our reconfigurable multiple clock distribution. The design is that of a 3-clock domain 15-tap distribution with each region numbered in column-wise fashion. Taps 1, 2, 3, 5, 6 and 10 are connected to clock domain A with a frequency of 1.11 GHz. Taps 11, 12, 14 and 15 are connected to clock domain B with a frequency of 1.33 GHz. Taps 4, 7, 8, 9 and 13 are connected to clock domain C with a frequency of 1.66 GHz. Figure 4.8A shows the synchronized clocks produced by our clock network using an extracted-level simulation for each of the domains. Figure 4.8A shows a close up of the resulting rising clock edges. The total skew from the first-to-last edge is 5.5 ps for domain A, 4.7 ps for domain B and 3.9 ps for

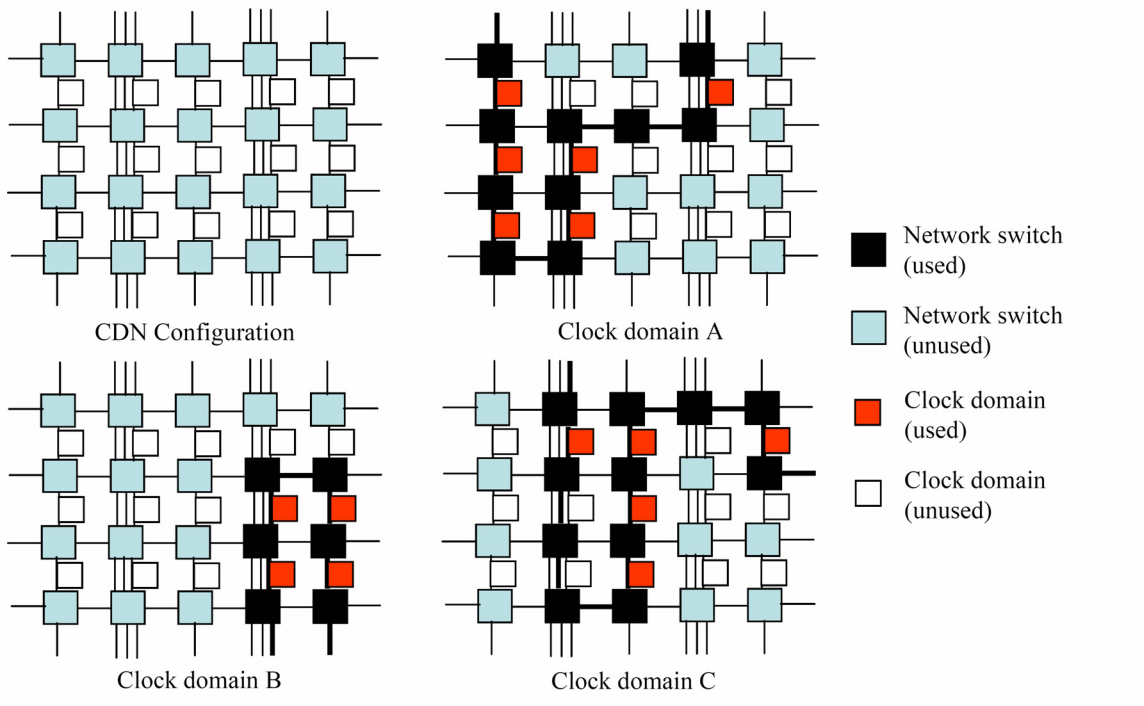

Figure 4.7: Potential fully programmable clocking architectures.

domain C. Comparable solutions offer similar or worse levels of skew reduction for a single domain as discussed in Chapter 3, without the ability to reconfigure clock post-silicon. This method reduces skew to under 4%, versus the typical 10% metric which is usually desired. The total power consumed is 62.82 mW, or 4.188 mW per tap. Roughly half of the power is consumed by the routing switches of the reconfiguration circuitry due to the capacitive load that they each must drive. According to [76], PLL-based clock networks typically consume hundreds of milliwatts due to their analog components. This fact will be true regardless of technology, so it is beneficial to use an all-digital approach for skew compensation such as ours.

The synchronization process will account for loading effects and inter-die process and fixed temperature variation to create negligible skew within a clock domain. Using

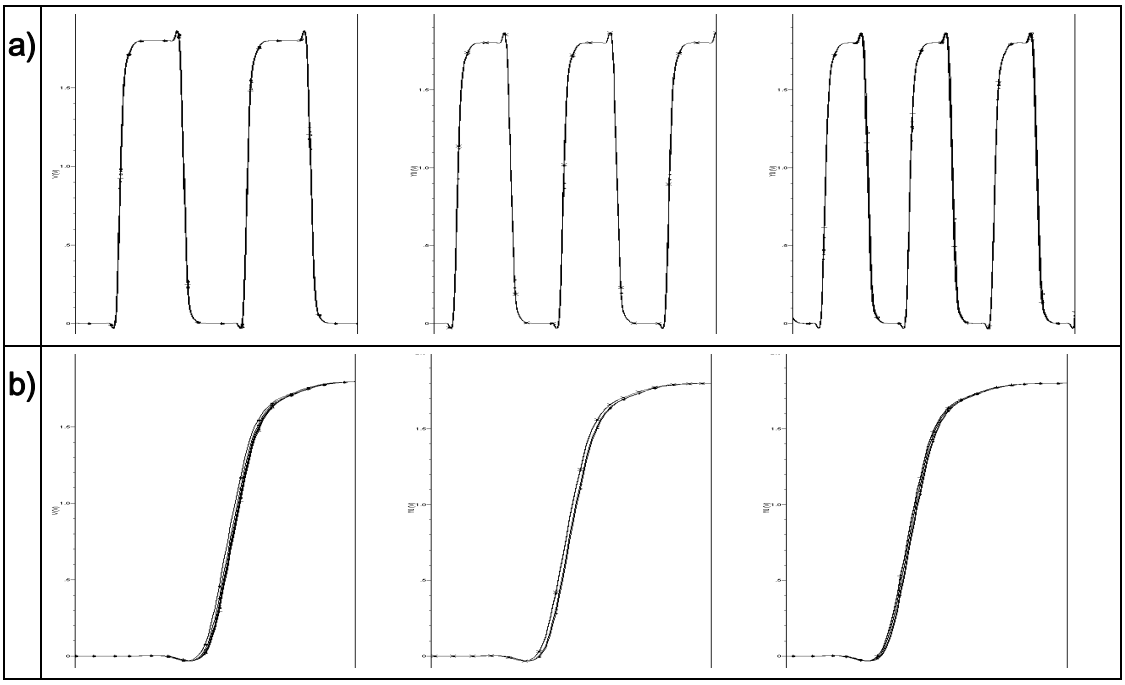

Figure 4.8: Simulation of a 3-clock domain reconfigurable clock network

a) Synchronized, calibrated 3-clock domain reference circuit simulation

b) Clock edges for a 3-clock domain reference circuit

(left to right: Domain A, Domain B and Domain C).

the circuit in Figure 4.7, we study here the durability of the synchronization to thermal and voltage variation in these conditions over time. We define the absolute skew δ to be the range in picoseconds of the first-to-last rising edges of all the clocks in a domain given a temperature or voltage perturbation. The standard deviation σ of the delay between all the rising edges for a given clock event, or sample, is also found. Table 4.1 shows the complete synchronized network's response to changes in temperature. The system can be synchronized at any initial temperature since we are studying the effect of temperature changes, so the 27°C starting point is arbitrary. At 37°C, the worst-case skew is roughly 10% of the clock period, and 20% at 47°C. It is possible to re-

synchronize the clock distribution dynamically to account for the variation to obtain near zero clock skews once again. While this test shows that the circuitry can tolerate small changes to operating temperature, it is important to synchronize it at or near the expected operating conditions.

Table 4.2 shows the system under the influence of a fixed voltage variance of plus and minus 5% of the nominal supply voltage at each of the taps. Here, the worst-case skew of 4.2% occurs at a 5% undervoltage. There is 2.4% worst-case skew at a 5% overvoltage. This is quite acceptable for any clock distribution. Table 4.3 shows the worst-case scenario where the entire distribution experiences a 5% fluctuation in the supply voltage. In this case, the clock fails to propagate to the taps in 3 out of 5 cases for the undervoltage test, and 2 out of 5 cases for the overvoltage test for clock domain C. The worst-case skew for the remaining domains is roughly 13%, occurring at the slowest clock frequency for the 1.71V case. This shows that the circuitry can operate correctly under these conditions, but the range of frequency that can be used is decreased. While a fixed voltage fluctuation is unlikely to occur since such a non-transient variance such as this will be accounted for dynamically at synchronization time. The CDN can also be re-synchronized at any time to account for any of these fixed variations to obtain near zero skew.

Since our clock distribution operates in an open loop after synchronization, the average clock period for a number of consecutive samples is ideal, as long as the clock source generates accurate clock periods. However, voltage fluctuations due to noise can

advance or delay an individual clock edge resulting in the injection of jitter. Noise simulations usually are time-intensive due to the short time steps required to replicate high frequency noise components. However, work in [80] has shown that low frequency noise has the dominant role in clock characteristics. We use a statistical approach to inject the supply with noise in a manner similar to [101] and [102]. Using the Verilog – AMS tool available in Cadence to generate a 1.8V source with standard deviation of 0.3 at a 100 GHz sampling frequency, we obtain 99.73% of supply voltage samples within +/- 5% of nominal. Since the voltage fluctuates over time, a number of consecutive samples must be observed and analyzed. Each sample represents a subsequent rising clock edge. In addition, the simulation is run twice to ensure that the results are well correlated.

The result of the test is shown in Table 4.4. The jitter represents changes in the clock period. The worst-case jitter φ here is the maximum amount that a clock period differs from ideal for all samples in the complete series. The standard deviation σ of the jitter is taken for every jitter measurement in the series. The ideal period is calculated by calculating the average of the time difference between a clock signal's rising edges. The maximum skew we found for our noisy supply test is roughly 2% of the clock period in the worst-case. The jitter of under 4 ps is predictably low due to the open loop nature of the clock distribution. This level of jitter is negligible when compared to the jitter injected by the clock source.

Temp. (ps)	27C		37C		47C	
	δ	σ	δ	σ	δ	σ
CLKA	0.0054	0.0023	51.3100	19.4379	108.859	41.0760
CLKB	0.0047	0.0025	34.6850	15.7428	68.8730	30.7528
CLKC	0.0040	0.0017	63.0500	22.8735	124.288	44.8673

Table 4.1: Temperature effect on synchronized network.

Voltage I (ps)	1.8V		1.71V		1.89V	
	δ	σ	δ	σ	δ	σ
CLKA	0.0054	0.0023	11.3451	4.7666	16.6580	5.8075
CLKB	0.0047	0.0025	21.9480	10.3804	13.4140	5.7539
CLKC	0.0040	0.0017	25.4460	9.1542	14.8680	5.9829

Table 4.2: Effect of voltage variance on taps.

Voltage II (ps)	1.8V		1.71V		1.89V	
	δ	σ	δ	σ	δ	σ
CLKA	0.0054	0.0023	118.210	44.2412	95.8270	36.8799
CLKB	0.0047	0.0025	51.3270	23.5288	19.9130	9.9043
CLKC	0.0040	0.0017	8.2915	N/A	31.0580	N/A

Table 4.3: Effect of voltage variance on synchronized network.

Voltage III (ps)	Trial 1				Trial 2			
	Skew		Jitter		Skew		Jitter	
	δ	σ	δ	σ	δ	σ	δ	σ
CLKA	8.382	1.005	3.028	0.642	8.508	1.001	2.330	0.529
CLKB	6.869	1.448	1.306	0.402	6.945	1.495	3.163	0.797
CLKC	11.374	2.623	3.069	0.916	9.933	2.417	3.959	0.858

Table 4.4: Effect of voltage supply noise on jitter.

4.8. Conclusion

We have designed a novel skew-tolerant multi-point clock distribution that is suitable for irregularly-shaped clock domains. The programmable repeater stages allow us to redirect clocks post-silicon at certain pre-defined switchpoints, making the distribution reconfigurable. The use of a single conductor to provide both forward and reverse reference signals is unique. In addition, our clock network will suppress intra-die process variation by being tolerant to proximity-based device mismatch and variance caused by cross-die temperature fluctuations and hot-spots.

Hot-spots and thermal management are becoming increasingly challenging problems for designers that can also be a significant source of skew in traditional clock networks. Clock networks can be modified to correct for some manufacturing defects, including bypassing certain clock lines and clock buffers. The operating clock frequency can be changed to fit an IC's many possible target applications. This method is scalable and simplifies the way a design can be floorplanned onto an integrated circuit and is useful for both static and programmable designs. Our tests show that the reference-based programmable clock distribution is resilient enough to be used in an ASIC, SoC or FPGA environment, exhibiting good operating characteristics, once synchronized, everywhere in the design envelope.

Chapter 5:

A built-in system for online clock skew debug and correction

5.1. Introduction

We present a low-cost on-line system for clock skew management in integrated circuits. Our Built-In Clock Skew System (BICSS) uses a centralized approach to identify, quantify and correct skew. It is a low-cost design that can be applied using a simple and cheap microprocessor-based tester. The circuitry employs a two-step method to first assess the time-of-flight between the central debug circuitry and each region, or tap under test, to account for measurement error. The system eliminates the difference in the measurement path delay between the clock regions under test and the

central measurement block using an averaging technique. The system then uses a high resolution digitally-controlled delay line in each clock region to perform the required skew compensation. BICSS can be used to detect skew above a user-adjustable margin using a variable tolerance phase detector.

The technique is unique in its ability to assess the time-of-flight between the central debug circuitry and each tap to account for the path length measurement error common in existing techniques. The result is an all-in-one solution that provides silicon debug and repair capability, providing added visibility to clock skew between differenct regions of the clock distribution network, post-silicon. The system has been simulated using an extracted-level design in TSMC's 180 nm standard process technology.

5.2. Background

Clock skew in integrated circuits is a significant problem facing designers and IC architects. The clock periods have shrunk to the point where they are now on the same order as intra-die propagation delays [103]. While device delays decrease with newer process generations, the interconnect delay is increasing due to fringe capacitance and electromigration effects on the interconnect [14]. Since clock signals on an IC are equally susceptible to interconnect variances as data or control signals. As a result, a growing portion of clock skew uncertainty is interconnect related and must be accounted for in a design. This often requires newer designs to dedicate a higher proportion of a clock’s period to clock uncertainty hurting overall performance.

There exist a number of both passive and active clock distribution schemes presented to help alleviate the problem [66],[79],[104],[105],[106]. Some clock skew reduction methods place many PLLs on an IC [97],[107]. This circuit duplication creates wasted silicon area and increased power consumption. Some analog solutions [79],[105] continually adjust to changes in clock delays, but generate jitter and are slow to lock because of the use of feedback in their approach. They also tend to hide exactly how much clock skew is present in a system. Digital solutions [66],[106] are quick to synchronize, but may be susceptible to environmental or operational conditions, such as temperature and power noise. This sensitivity requires digital solutions to be monitored to determine when the system requires resynchronization, although they also tend to handle a variety of clock frequencies, such as those used in a dynamic frequency scaling applicable, much better.

The verification of clock skew reduction techniques is a difficult problem due to the precision required and the non-negligible interconnect delays encountered between measurement points. There is a need for two distinct types of post-silicon verification of clock distribution networks: *functional verification* and *run-time verification* [108]. Off-chip testing is an effective way to perform functional verification, but too often these tests are not performed at speed, yielding a test procedure that may ignore skew-induced timing violations. In a microelectronics world where variances are measured in picoseconds, routing clock signals off-chip through pins or test pads can induce the skew that needs to be measured. The cost of the equipment required to precisely measure clock skew may also be prohibitive to their use. On-chip functional testing approaches, while cheaper,

still suffer from path length variances when not placed directly in between the clock points being measured and accounted for in any skew measurements. Some solutions ignore the transport delay from the clock tap or leaves to the location of the skew detection component which can lead to intrinsic error in the approach [65],[97] since the distances can vary widely across an IC. [109] creates an effective method of measuring clock jitter at a latch with picosecond precision by clocking the latch with a reference clock and feeding the global clock into the data input. However, their methodology will experience error due to skew in the reference clock when used to measure skew in the global clock.

We present here a system designed to be a built-in approach incorporating silicon debug circuitry into integrated circuits to allow users to debug timing faults, determine their source and potentially correct them as well. Our system can be used as a low-cost functional verification solution that can be applied using a simple and cheap microprocessor-based tester. This approach can limit the test time required on more expensive alternatives to correct skew due to process variation in a CDN at the factory. While there has been an increasing amount of design and research effort placed into designing low skew clock distribution networks and fault testing clock distribution networks, there has been relatively little emphasis placed on run-time verification of skew in clock distribution networks to ensure the correct behaviour of designs over real world operating conditions. Changes in temperature and power supply output can create significant changes to device delay in ICs and this can adversely affect the CDN during in-system operation. Our design can be used for run-time verification to detect skew and

correct for environmental changes. No other method can compensate for line length variances and measure, monitor and, if necessary, compensate for clock skew using a single centralized approach.

Our approach uses a dedicated central block to first compensate for mismatches in line length between clock measurement points and then to minimize any detected clock skew. Other solutions place many phase detectors or PLLs on an IC equidistant from their measuring points [107],[110]. This circuit duplication represents wasted silicon area and increased run-time power consumption unless the additional hardware can be disabled. A centralized approach such as ours has the advantage of component reuse [111] when used with multiple test pairs of clocks. Our system differs from others such as [112] by being entirely digital. By precisely accounting for differences in global and local clock distribution, our solution is highly effective in dealing with architectures such as those shown in Figure 5.1. The components in these designs may all have clock distributions with different latencies, highlighted in Figure 5.2 for clock regions A and B. These latencies can result in significant clock skew between the global root and the clock leaves of the different modules, even if the clock inputs to each module have been perfectly synchronized. As such, it is well suited to detect and correct for clock skew in modular designs such as SoCs [113]. BICSS also exhibits many of the advantages of other post-silicon tunability approaches [114],[115] to overcome process variation in clock distribution networks.

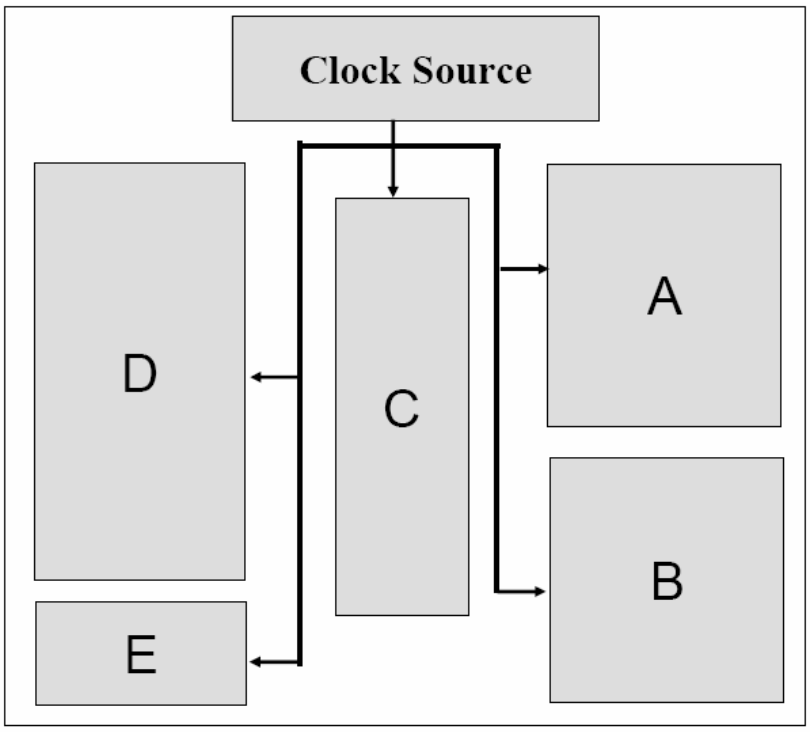

Figure 5.1: Modular design of ICs with regions A-E in a given clock domain.

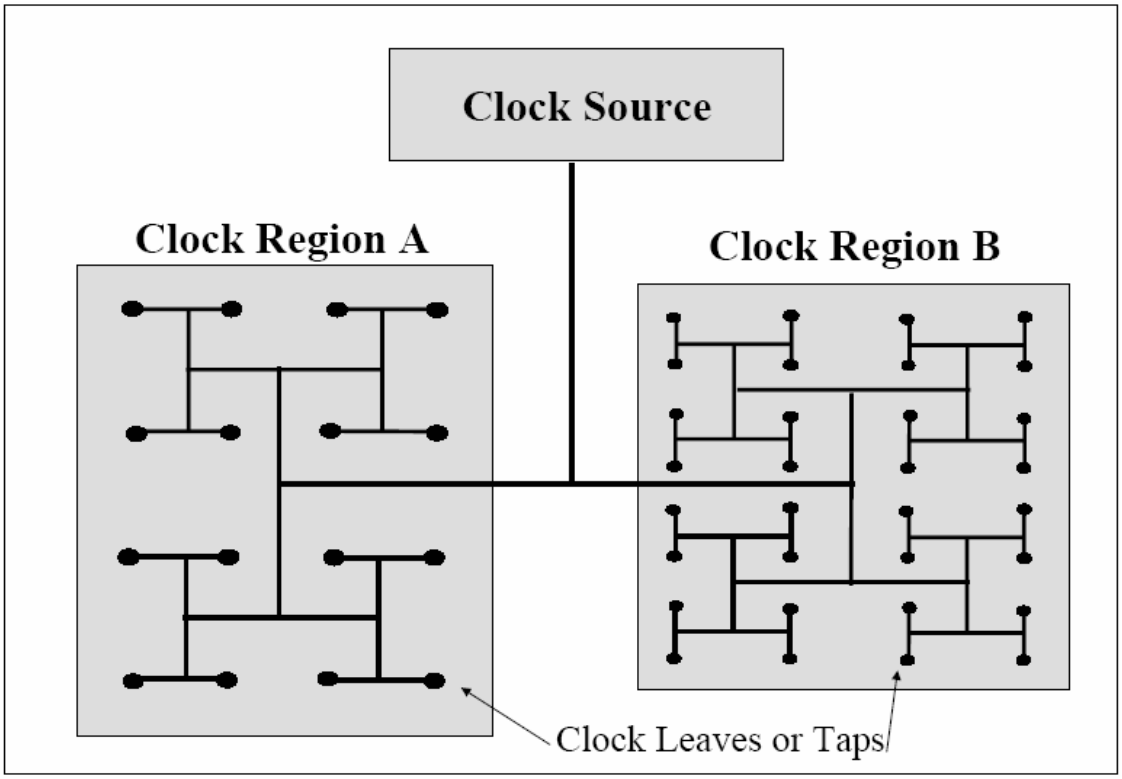

Figure 5.2: Close up of two regions A and B in a clock domain.

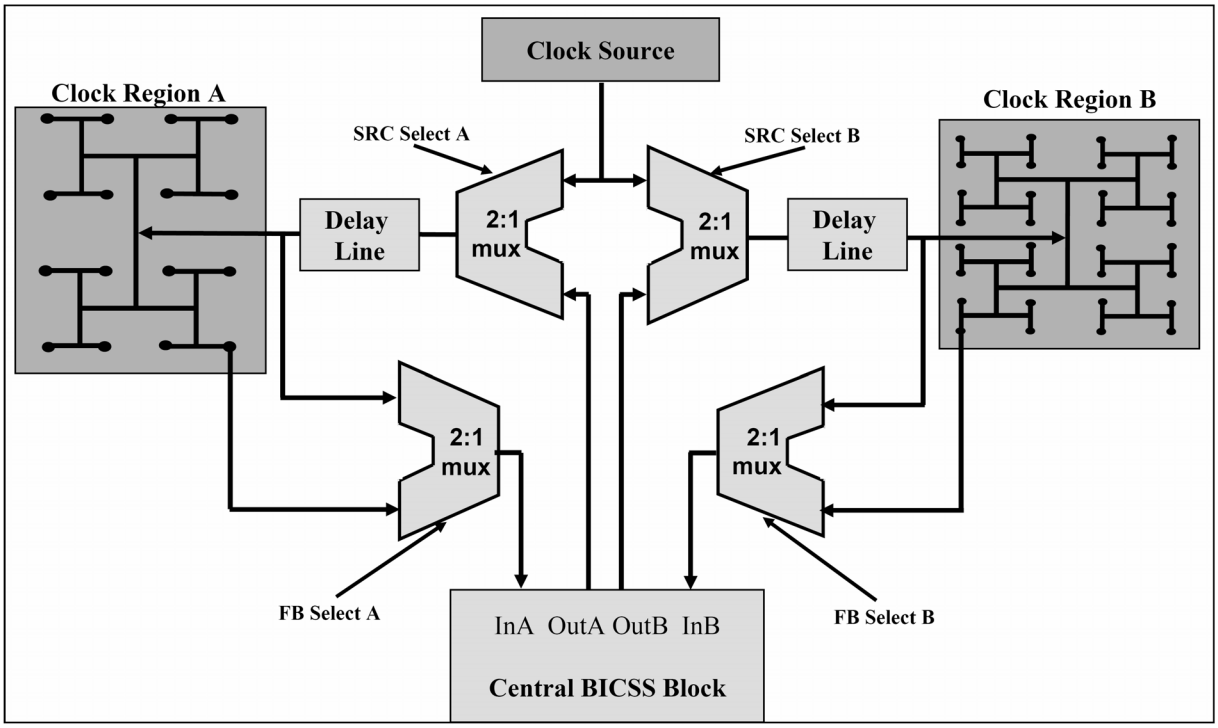

Figure 5.3: The Built-in Clock Skew System (BICSS).

5.3. System architecture

Our Built-In Clock Skew System (BICSS) consists of a central skew management core and an array of distributed evaluation nodes consisting of a delay line and 2 multiplexers placed at each tap under test (TUT). The system operates in 2 stages: a *normalization* stage and a *measurement* stage. Normalization determines the one way propagation delay between a TUT and the central debug hardware. Clocks are compared pair-wise during measurement, with each clock signal delayed by the one way transport delay of the opposing clock. An example 2-TUT design is shown in Figure 5.3. The additional circuitry required at each clock region is only 2 multiplexers and a 1 delay line. The input multiplexer (*SRC*) selects either the source clock or the test clock used to

normalize the measurement path length. The output multiplexer (*FB*) chooses between the test clock during normalization and the leaf clock for clock skew detection and measurement Together, the *SRC* (source) and *FB* (feedback) multiplexers are used to shift between normalization and evaluation modes. The phase detector sees the outbound clock and the return clock for normalization and the operational clocks from the two TUTs for measurement. Delay lines are used to compensate for path length variances and measured skew. Figure 5.4 shows the centralized circuitry required to first perform the normalization and then the skew detection and measurement. A phase detector with an adjustable locked region permits users to configure the amount of skew that will be tolerated based on the application.

The central circuitry consists of 4 delay lines, 2 multiplexers (*mux*), a phase detector (*PD*) and a synchronization controller. The delay lines should be located close together to limit process variance and should be laid out to provide good matching. The

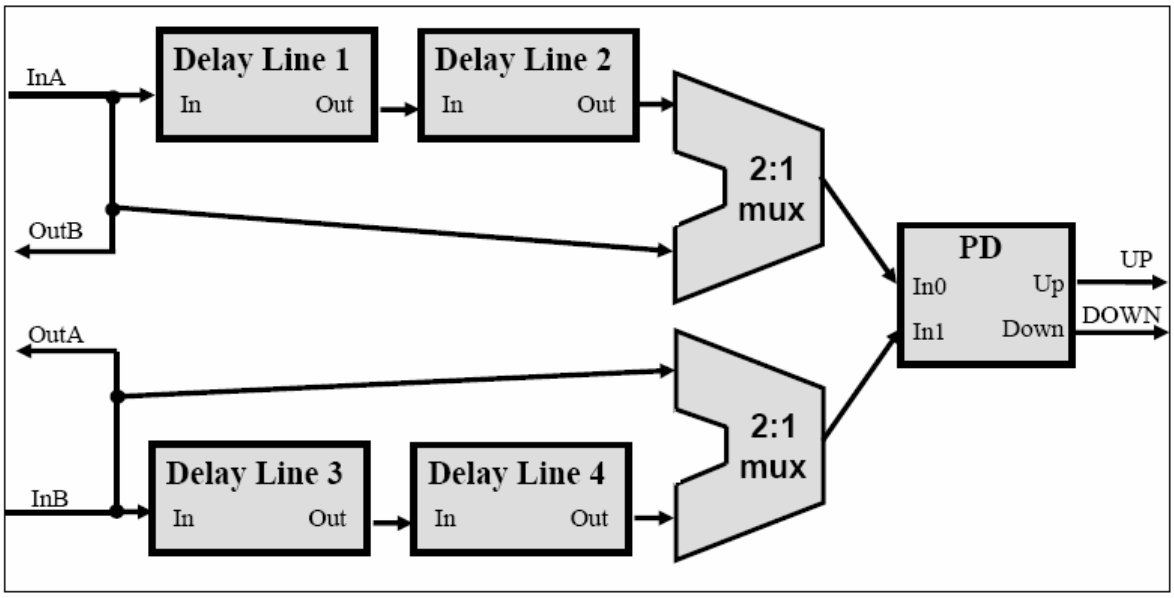

Figure 5.4: Central BICSS circuitry.

controller design can be adjusted to accommodate different applications. It can be implemented as a finite state machine, in software or hardware without restriction, either off-chip or on. Since all the multiplexer delays are accounted for during the calibration, there are many configurations possible when using BICSS to detect and measure skew for many regional clock pairs. For simplicity, a fixed delay equal to the minimum latency through the delay line path is added to the two undelayed multiplexer inputs. If each delay line has a maximum delay (δ), the maximum round trip from the centralized BICSS circuitry to any clock region is $2\cdot\delta$. In this case, the maximum skew that can be handled by the system is $+/-\delta$, independent of the clock frequency.

The skew detection circuitry uses the same averaging principle used to create skew-tolerant clock networks in previous sections. The central BICSS circuitry sends a signal (*Out Clock*) to the clock region and uses two delay lines to align this clock source with the return clock (*In Clock*) using the phase detector as a guide. Removing one of the delay lines from the signal path leaves the a*verage* clock since both the delay lines use identical delay settings. The tap to average clock delay includes both the transport delay to the local clock region and half of the circuit delay in the round-trip path. Since every return path contains exactly one delay line and two multiplexers, they should all ideally have the same circuit delay.

Process variation could result in skew from mismatches in the clock routing circuitry and from duty cycle variations in the delay line, but the maximum skew injected during calibration is halved due to the averaging technique used. The forward and return

paths should be co-located to match propagation delays in both directions. For clock skew detection, the *A* clock input (CLK_a) is delayed by the path delay to region *B* (DEL_b) and the *B* clock input (CLK_b) is delayed by the path delay to region *A* (DEL_a). The phase detector then determines the relation between the two clock inputs, CLK_a+DEL_b and CLK_b+DEL_a: either *up*, *down* or *locked*. The development of a phase detector with an adjustable width locked region adds functionality to the system since it allows the amount of skew that can be tolerated to be configured depending on the specific application. Should the skew between CLK_a and CLK_b ever exceed the programmed tolerance during run-time, this fault can be flagged and compensated for – a feature particularly useful for digital clock skew compensation systems. Figure 5.5A and 5.5B show the normalization stages and Figure 5.5C shows the datapath used for synchronization.

BICSS can also be used to infer the amount of skew that is present between CLK_a and CLK_b with delay lines 1 and 3 used to retain the normalization delay settings, clock delay lines 2 and 4 are available for determining the actual skew. Delay line 2 is incremented when clock *A* lags clock *B*, and delay line 4 is incremented when the opposite is true. Once aligned, the delay settings can then be externally read to determine the inferred (measured) skew. The precision of each synchronization is one half the maximum of either the maximum delay increment or the locked range setting of the phase detector. To achieve more accurate skew measurements in the presence of clock jitter, a clock skew measurement can be performed numerous times, taking the average to represent a more precise skew, similar to taking the center of an eye-diagram

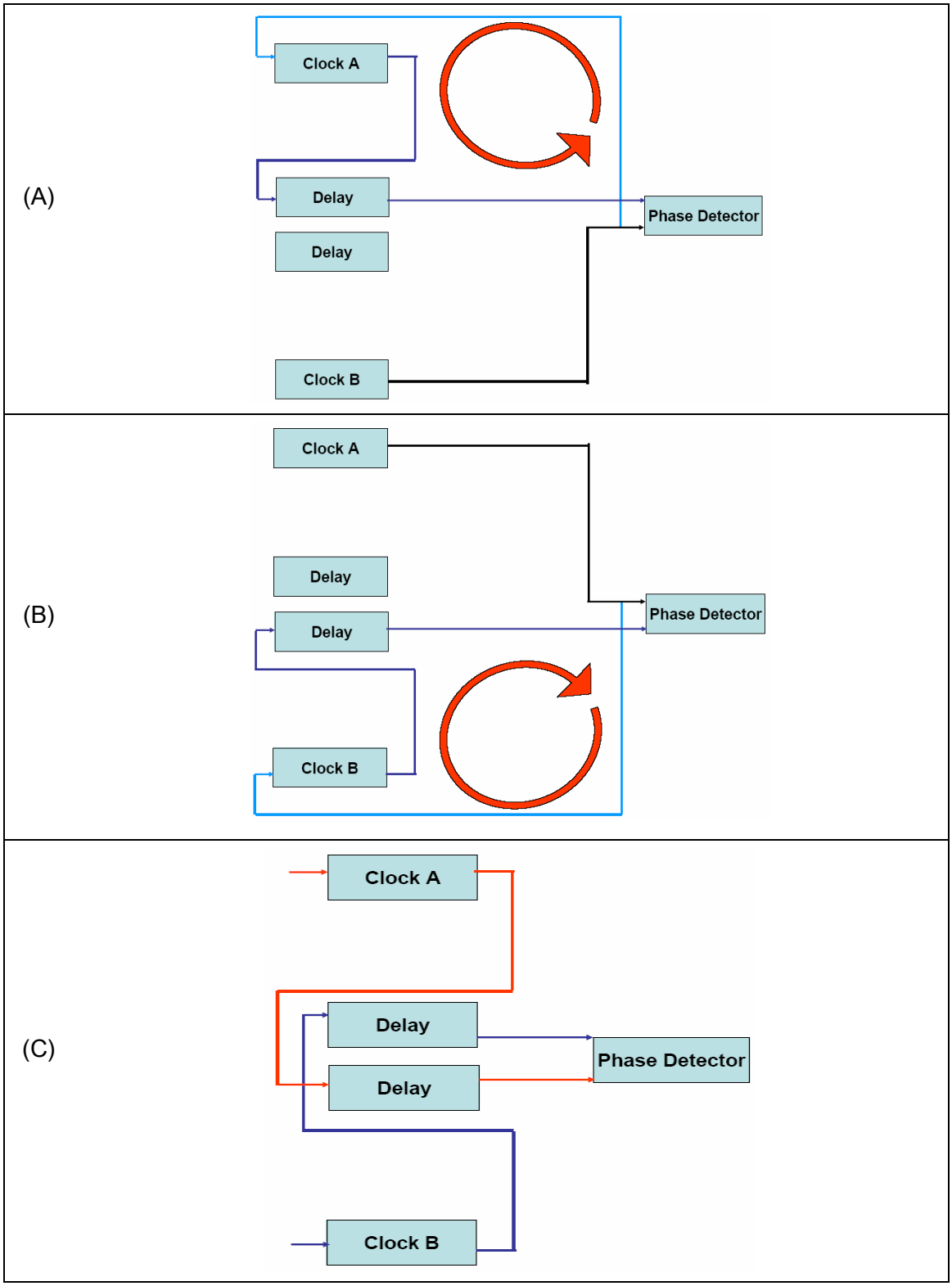

Figure 5.5: Datapath used for normalization stages (A and B) and synchronization (C).

with an off-chip oscilloscope. Copying the delay line settings of the central unit to the local clock regions will effectively minimize the clock skew between the two points. Resetting delay lines 2 and 4 will allow BICSS to enter skew detection mode for the skew calibrated system. The same principle can be applied to comparing any pair of clocks (*I*, *J*) on the device using CLK_I+DEL_J and CLK_J+DEL_I.

BICSS is designed as a regional solution to characterize a representative subset of a clock domain or a few skew critical clock leaves. A typical application could involve placing an evaluation node at specific levels of a clock tree, such as the design in Figure 5.3. The system can be used to correct for operating temperature and process drift in each region. Routing two co-located signals between every evaluation node and the central hardware is not a prohibitive penalty given this approach. There is no constraint on how the signal pairs are routed to different taps since path length variances are compensated. BICSS can also provide a measure for the quality of the device by determining how far the measurement points are out of alignment.

5.4. Operating characteristics

The circuits and system discussed here have been designed and laid out in TSMC's standard 180 nm process using the Cadence Virtuoso design environment and simulated with SpectreS using the Analog Artist tool. The layout of the design is shown in Figure 5.6. The distributed nodes require 1200 μm^2 and the central skew detection block requires 8100 μm^2. The extracted simulations show that the coarse grain

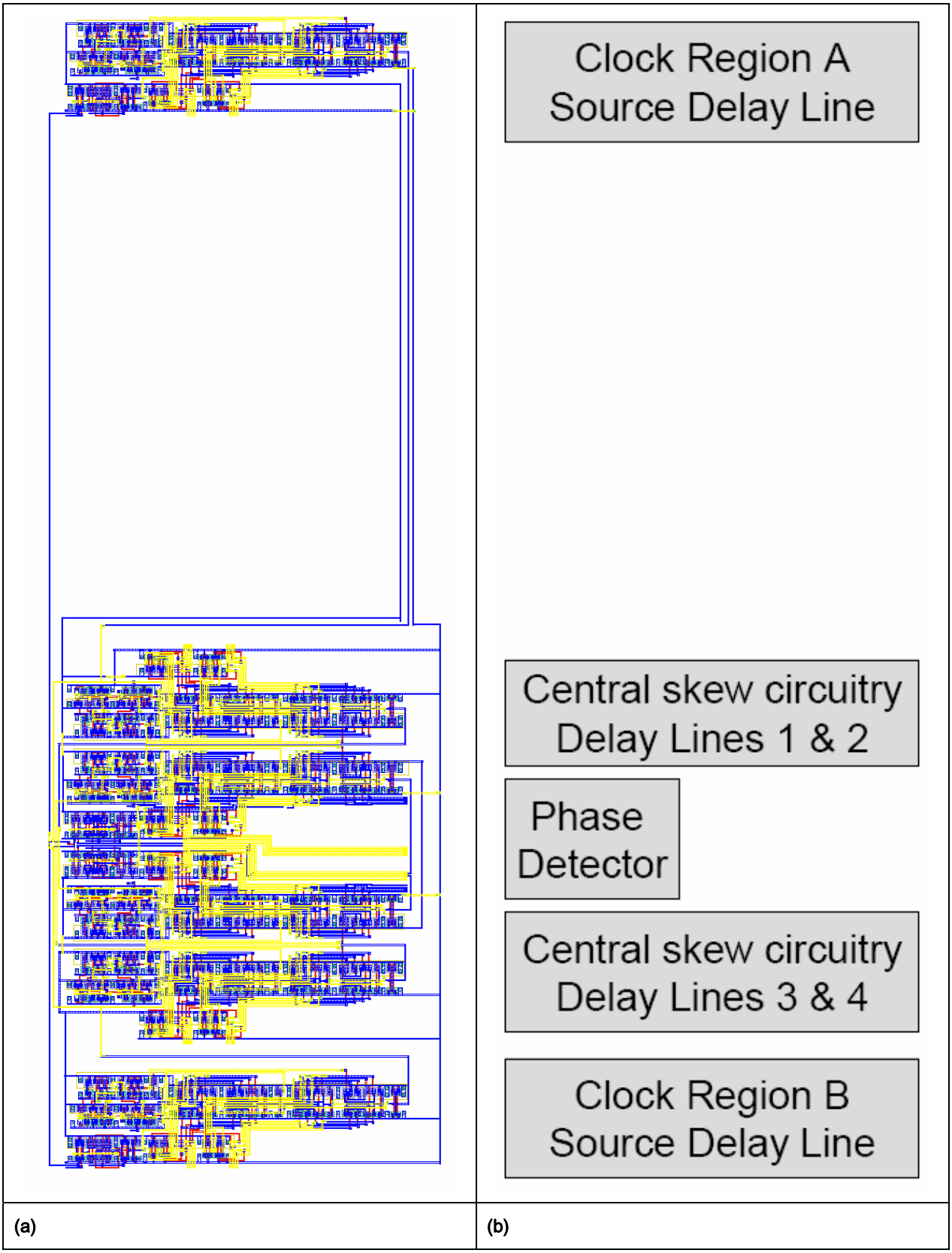

Figure 5.6: The layout of a 2-clock region BICSS implementation.

component is adjustable in 93.5 ps increments. The fine grain delay line is capable of relatively linear delays up to a maximum of 99.3 ps. 196 delay settings are retained between 0 and 99.3 ps for an average delay increment of 0.50 ps and a maximum increment between adjacent settings of 2.93 ps. The total delay of the delay line, δ, fixes the maximum round-trip delay from the central BICSS circuitry to the furthest clock region at $2*\delta$ = 385.6 ps for this implementation. The maximum delay that can be compensated for between clock regions is +/- 192.8 ps. The coarse grain component of the delay line can be increased to allow for larger skews and longer round trip delays.

The fixed phase detector is tuned to allow a +/- 2 ps tolerance, a 4 ps locked region. The variable phase detector can be set to have a 3, 6, 8, 10, 28, 29, 35, 37, 51, 52, 55, 59, 67, 106 or 253 ps locked region. The extracted simulations show that the system is capable of operating up to 2.30 GHz, however the widest locked regions cannot be used with the highest frequencies because of slew rate limitations. The accuracy of each synchronization is half the maximum between the delay line resolution (3 ps) and the locked width of the phase detector (0 ps for the variable tolerance detector, 4 ps for the fixed tolerance detector). Since three synchronizations are required for clock skew measurement and compensation, the intrinsic overall precision of the system is 4.5 ps for a system using the variable detector and 6 ps for one using the fixed detector.

The power consumption is 11.9 mW for a fixed tolerance phase detector system monitoring two taps at 2 GHz. This is comparable to other skew compensation solutions.

Other centralized in-die clock skew measurement tools will have error many times greater than ours since they do not compensate for transport delays. As such, it is difficult to compare BICSS to other low cost solutions. There are other clock skew measurement techniques such as the time interval analyzer (TIA) method or picosecond imaging circuit analysis (PICA) that may have similar or better performance, but they come with much higher cost and complexity. A full description of other clock skew measurement techniques can be found in [116].

The operation of the system is shown in Figure 5.7 using 1 GHz clocks skewed by 150 ps between two clock regions. The top waveform shows the initial skew between the two clocks at their respective regions between 0-10 ns. Next, the system calibrates the line lengths between the centralized BICSS circuitry and clock regions *A* and *B*. The second waveform shows the inputs to the phase detector, and the third and fourth waveforms show the *down* and *up* signals. Times 10-15 and 20-25 ns show the initial state and times 15-20 and 25-30 ns show the aligned final result of the line measurement process for clock regions *A* and *B*, respectively. Recall that simultaneously asserted *up* and *down* signals indicates a locked state. Time 30-40 ns represents the result of the measured clock skew after using delay lines 2 and 4 to perform the alignment and measurement, representing the skew scope functionality of the circuitry. Finally, the appropriate delay line settings are copied back to the local regions resulting in synchronized local clocks, shown in the first waveform between 40-50 ns. The circuitry can either be disabled during run-time, or remain in place to detect excess phase drift. The system can be used for multiple taps by inserting additional

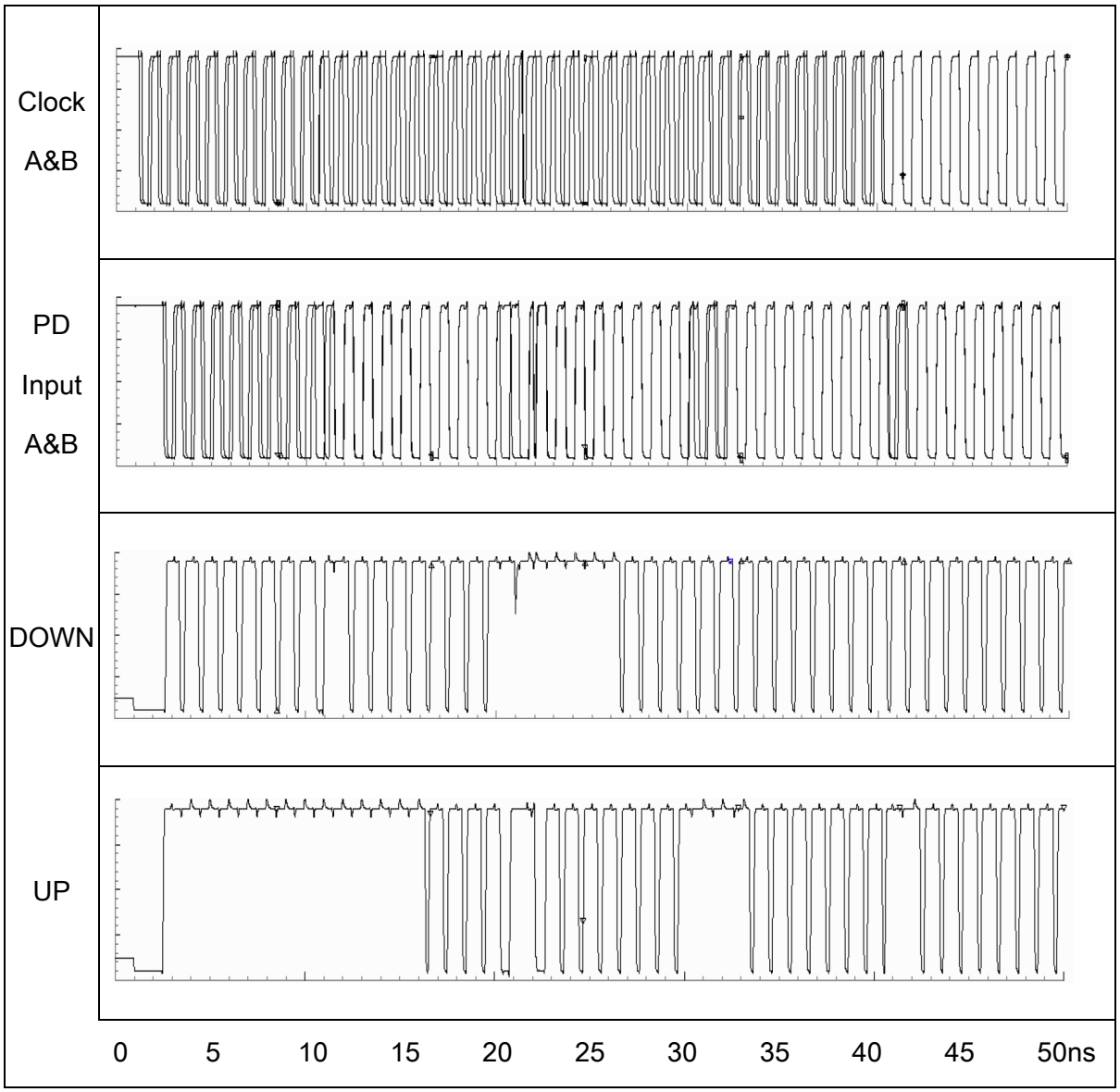

Figure 5.7: Waveforms showing operation of the BICSS circuitry.

multiplexers at the central block. A plurality of taps can either all be verified with respect to a single reference region, or pair-wise. In either case, it is easier to initially normalize every tap and store the required delay settings in memory before measuring skew. Incorporating our clock skew management system on a design does not require a prohibitive amount of area or design effort. This approach also allows the central BICSS

block to poll any pair of taps for clock skew during operation without affecting the circuit's behaviour. The simulations shown are restricted to one pair of clocks for simplicity, without loss of generality.

To compare all the clocks taps (*i*) in a given clock domain for skew, the quickest approach would involve at most $\log_2 N$ comparisons where N is the smallest power of 2 greater than *i*. CLK_1 is compared to CLK_2, CLK_3 is compared to CLK_4 and so on until all or all but one of the local clocks have been compared exactly once. Next, CLK_1 is compared to CLK_3, CLK_5 is compared to CLK_7, and so on until all of the odd numbered clocks have been compared. At this point, every grouping of four clocks will be synchronized. The process continues with every fourth clock (CLK_1 compared to CLK_5, CLK_9 compared to CLK_{13}, etc), eighth clock, sixteenth clock and so on until there are no clocks to verify. The concept is applicable for any number of clock regions (*i*) without modification to the hardware besides the inclusion of additional multiplexers to route the appropriate clock signals to the central BICSS circuitry. One appropriate approach for handling multiple clock pairs during operation could involve storing the calibration settings and polling each pair sequentially to periodically check whether clock skew is still within allowable bounds.

5.5. Conclusion

Clock synchronization is important to ensure the fault-free high performance operation on an integrated circuit. Process variation can affect both transistor delay and

interconnect behaviour. While study in [103] compares the expected delay of an interconnect with an inverter chain delay, our method averages the round trip path of an interconnect creating a more accurate measurement. Variances can create problems for clock distributions, even some of those using active or passive skew reduction techniques. Traditional techniques largely ignore time of flight differences between evaluation points. Once normalized, skew can be quantified using high-resolution delay lines and the measurement read off-chip. The system is scalable and can be used to assess skew at a number of different locations on an IC. Using the skew measures allows the quality of the clock distribution on the fabricated die to be assessed. Using delay lines at each tap, our skew management system can minimize the skew between points to repair otherwise defective dies. BICSS also aids in the debugging of timing errors that may be discovered during testing due to the added visibility of on-chip clock signals.

Our BICSS system is unique in its ability to detect, measure and compensate for clock skew using a single all-in-one solution. The development of a variable tolerance phase detector makes this the first system to allow online detection of a programmable skew bound. The complete solution can provide additional visibility to silicon devices for any existing clock distribution and the ability to repair otherwise defective devices. BICSS enables designers to modify their design flow to include post-fabrication adjustment to the clock distribution network to correct for timing faults or to minimize clock skew for higher frequency operation. It simplifies the design of modular and

system-on-chip architectures since it can detect and compensate for clock skew through unmatched local clock trees and repeater stages.

We use an averaging technique to compensate for different propagation delays between measurement points, which allows a single BICSS unit to be used for multiple test points providing an efficient system through component reuse. Our entirely digital solution requires little additional circuitry and adds visibility to on-chip clocks aiding in the on-line debug and repair of integrated circuits and is a low cost alternative to other, traditionally costly skew measurement techniques.

Chapter 6:

System-level modelling

6.1. Introduction

The individual components required to create the built-in clock skew system (BICSS) and to distribute clocks both in single and multiple clock reconfigurable forms have been designed and tested using extracted level simulation in TSMC's 180 nm technology. The systems can be implemented using a standard cell approach, but are shown using a specific set of components to demonstrate the effectiveness of the system. These circuit blocks can be interchanged with any other available blocks provided they possess similar signal characteristics: equal t_{pLH} and t_{pHL}, sufficient slew

rate, and closely matched delays for the reference line components. Interchanging components will not have an effect on the correct operation of the complete system since proper functionality is based upon component and interconnect matching not specific circuit implementations. In this chapter, we generalize our approach to apply our system to a wide array of clock network architectures for use with common components.

The design of the HDL models and controllers required to synchronize the clock networks and the BICSS unit are outlined and their complexity found for a baseline set of applications. The HDL models discussed in this chapter are generic enough to replicate the behaviour of a wide set of digitally-controlled components. The models can modify the propagation delay of devices and interconnect, they can operate with a variety of delay lines with different delay maps and they can modify the sensitivity of the phase detector to reflect different designs.

6.2. Implementation approach

Coordinating the synchronization and the reconfiguration of BICSS and our averaging clock networks requires controllers to monitor the system, configure the datapath and adjust the delay line settings to synchronize the clock signals. When coupled with the system models and loaded with the performance data of the target technology, it is possible to replicate the operation of a BICSS system and complete clock networks to predict the synchronization delay and the expected delay settings. The HDL code used to create the controller can be synthesized into any given technology

using a suitable library mapping. The controller can be made to handle a wide range of target clock networks, but each degree of freedom in terms of number of taps, number of clock domains and mesh network structures available will add to its complexity so it is important to restrict the flexibility of the controller to the desired silicon implementation.

For modelling clock networks in HDL, our approach is to design a control module for synchronizing an arbitrary tap in the network, making the design modular enough to be applied to a dynamic program that can be applied to a number of different clock network configurations. The number of clock domains, the number of taps per domain, the size of the configuration memory, the location and type of clock switches and the location of the control lines in memory should each be parameterized for and entered into a dynamic program generator to create paired controller and clock network models. The generated clock network model would be fixed in terms of the structure of the clock mesh network and in the number and position of the clock taps in the switch mesh, as would be the case in an integrated circuit implementation. Once the clock network controller HDL code is created, the mesh network configuration used to create the reconfigurable clock domains, the delay setting map of the delay line, the programmable delays for the devices and interconnect, and the target clock period can be modified in the code to synchronize any realizable configuration of the clock network model.

While the clock distribution and skew management systems are designed for an integrated circuit, the controller can be implemented in hardware on the IC, or externally, either in hardware or software. VHDL was chosen for the controllers because of its

flexibility: synthesizable VHDL code can be used externally in an FPGA or can be synthesized into an integrated circuit using an appropriate library and translator. Also, the code is generic enough to easily be translated into a language like C/C++ or subsequently to an assembly language for microprocessor target technologies. The controller programs were written for Altera devices using Quartus II to ensure that they were synthesizable. The models were then created using ModelSim and paired with the controllers for simulation. This is a convenient approach for this application since the simulations assume zero delay between events and all the signal delays can be explicitly coded into the program using VHDL's *transport* commands to replicate specific conditions that would be found on-chip. Controller clock frequencies and clock network frequencies can also be set arbitrarily to reflect different target technologies.

6.3. Configuration memory requirements

To create a single clock controller, the number of clock taps m is the first thing that needs to be set. Each tap will require 2 bits to control the direction of the reference clocks and 1 bit to control the polarity of the delay line's output clock. Assuming each delay line requires n memory bits and that this memory can be shared between the pair of delay lines located at the tap, the number of memory bits required per tap is:

$$memory\_bits_{fixed} = (n+3)\cdot(m+1) \qquad (6.1)$$

with one location reserved as a dummy location for power-up. The idea is to minimize the number of address and data lines, to minimize the number of writes required for updating each delay setting and to minimize the number of wasted bits. The minimum number of bits will be located where the number of data lines is equal to the number of address lines, so we need to solve:

$$memory\_bits_{fixed} = x \cdot 2^x \qquad (6.2)$$

for *x*. However, we also need to consider the constraint to minimize the number of wasted memory bits and to minimize the number of memory writes required per delay setting. To allow some control over this, the word size should be an input parameter to the dynamic program generator. The number of words per delay setting modification is:

$$words = \frac{n+3}{word\_size} + 1 \qquad (6.3)$$

rounded up to the nearest integer. There is trade-off required between the number of writes and the number of memory lines (address plus data lines) required. These memory lines need to be routed to every tap in the system and the time required for each synchronization depends on the number of memory locations per tap.

For a reconfigurable distribution, the memory requirements are divided into two sections. The first is the delay settings required for each tap that is:

$$memory\_bits_{reconfigurable} = (n+4) \cdot (m+1) \qquad (6.4).$$

The additional bit is required to choose which incoming clock is the forward clock. The second is the switch mesh configuration which depends on the size and type of switches

used in the mesh. Assuming *b* ports in a switch supporting *b/2* clocks, there are two possibilities for the number of memory bits required a mixed switch mesh with varying switch sizes:

$$memory\_bits_{mem\_encoded} = \sum_{i=0}^{\infty} a_i \log_2[2 \cdot (i-1)!] \qquad (6.5)$$

$$memory\_bits_{mem\_decoded} = \sum_{i=0}^{\infty} a_i \cdot i \log_2 i \qquad (6.6)$$

where a_i represents the number of switches in the mesh of a given port size *i* and each *log* is rounded to the next largest integer. It is assumed that the memory is local to each tap/switch and that memory cannot be shared between different switch instances. The figure in Equation 6.5 represents encoded switch data that needs additional hardware to generate the required control line data. There are *(b-1)!* connections possible between ports and the times two factor is included to represent the direction of the connection. The figure in Equation 6.6 represents a figure that does not require decoding. Each port can be connected to any other port (forward connected) or no ports (unconnected or reverse connected).

For a BICSS block operating on two clock taps, there are four independently controlled delay lines, three 2-input multiplexers and a phase detector. Since two of the multiplexers share *select* signals and assuming that each of the delay lines possess *n*-bit delay control and that the variable tolerance phase detector requires *n'*-bits for control, the total number of memory bits required is:

$$memory\_bits_{BICSS} = 4 \cdot n + n' + 2 \qquad (6.7)$$

To compare an arbitrary number of regional clocks *z*, two additional z-input multiplexers each requiring $log_2 z$ select lines are needed.

6.4 Synchronization controllers

The only signals required by the controller are the 1-bit *UP* and 1-bit *DOWN* status signals from the clock network or BICSS. The controller generates the appropriate delay setting and updates the clock network or BICSS by writing to memory using address lines - *addr(a..0)*, data lines - *data(d..0)*, and a write enable (*WE*). The biggest obstacle in designing a controller for these devices is the periodic nature of the clocks that need to be aligned. Around the synchronization point, one expects a shift from *UP* to *DOWN* or vice versa to be an indication that the correct coarse or fine grain setting has been found. This scenario can also occur when the signals are 180° out of phase. It is also possible for the signals to enter into a delay range which cannot be synchronized from due to redundancy in the delay range with respect to the clock period.

For example, if the clock period is 1000 units, and we have 8 coarse settings that increment by 200 units, an initial delay of 200 and a target delay of 800, the first command (using the minimum delay) will be a *DOWN*. The delay setting tried will be the maximum delay 7 coarse increments away, so 1600 units. This will result in an *UP* since it is closer to 2000 than 1000 units, returning the system to the previous setting, creating an infinite loop that will not result in synchronization. Since the error is caused by

redundancy in the delay line, it is necessary to limit the range of the coarse delays to a range between 1 and 1.5 clock periods.

The delay line characteristics can easily be altered since arbitrary (non-sequential) coarse and fine delay orderings are allowed and the total number of settings is available as a parameter to the controller. This allows delay data to be extracted from a sample delay line on-chip and used to characterize the IC's delay line behaviour. The actual delay is not necessary to build the controller since the delay settings are read out of memory, only the delay setting order that results in increasing delays is required. This will compensate for a certain degree of inter-die process variance. This approach also makes the controller circuitry compatible for other digitally-controlled delay lines. The approach assumes a 2-stage delay line with a coarse and fine grain control, but one of the two control subroutines can be disabled if the delay line is fine (or coarse) only, with no coarse (or fine) settings.

6.4.1 Single clock domain controller

Assuming *a* address lines and *d* data lines, a single clock domain system requires *4+a+d* bits to communicate between the controller and the clock network, shown in Figure 6.1. To design a controller for a single clock domain with an arbitrary number of taps, the controller begins with the first tap closest to the forward clock source and sequentially synchronizes the taps until they are all synchronized. The generic structure of a single clock controller is shown in Figure 6.2. The *tap_select* block shown

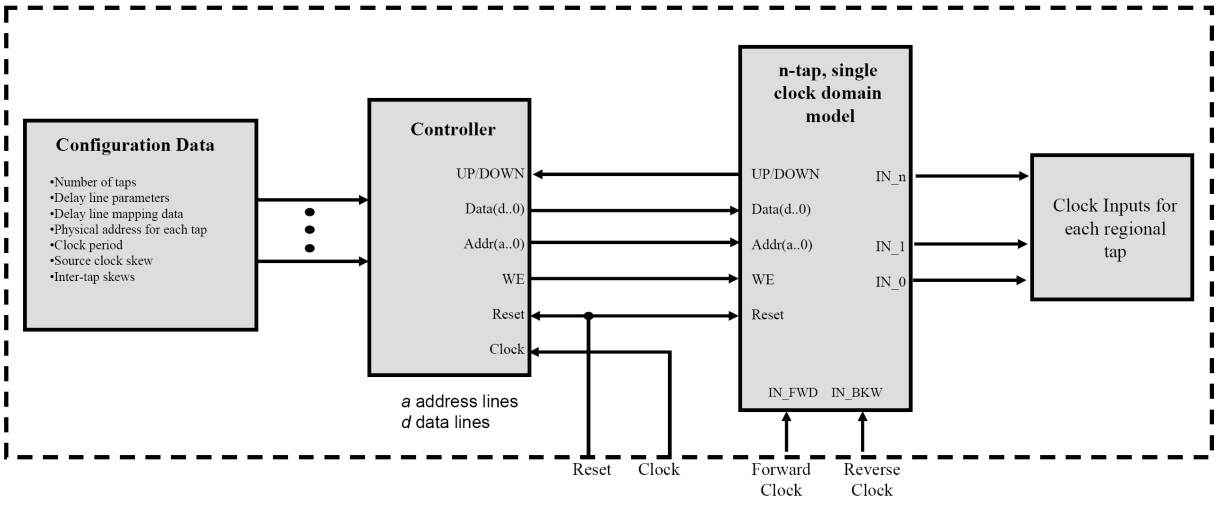

Figure 6.1: The single clock domain, *n*-tap controller and model.

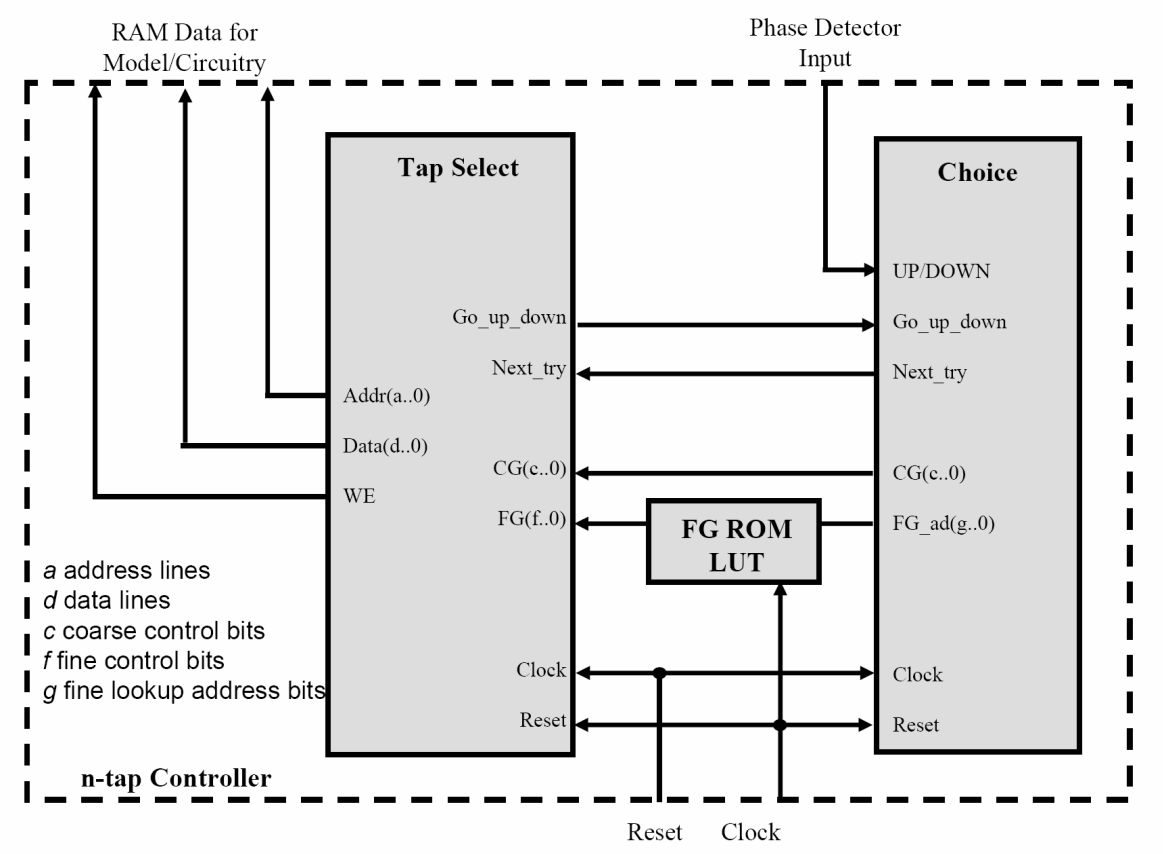

Figure 6.2: Controller used for single clock domain model.

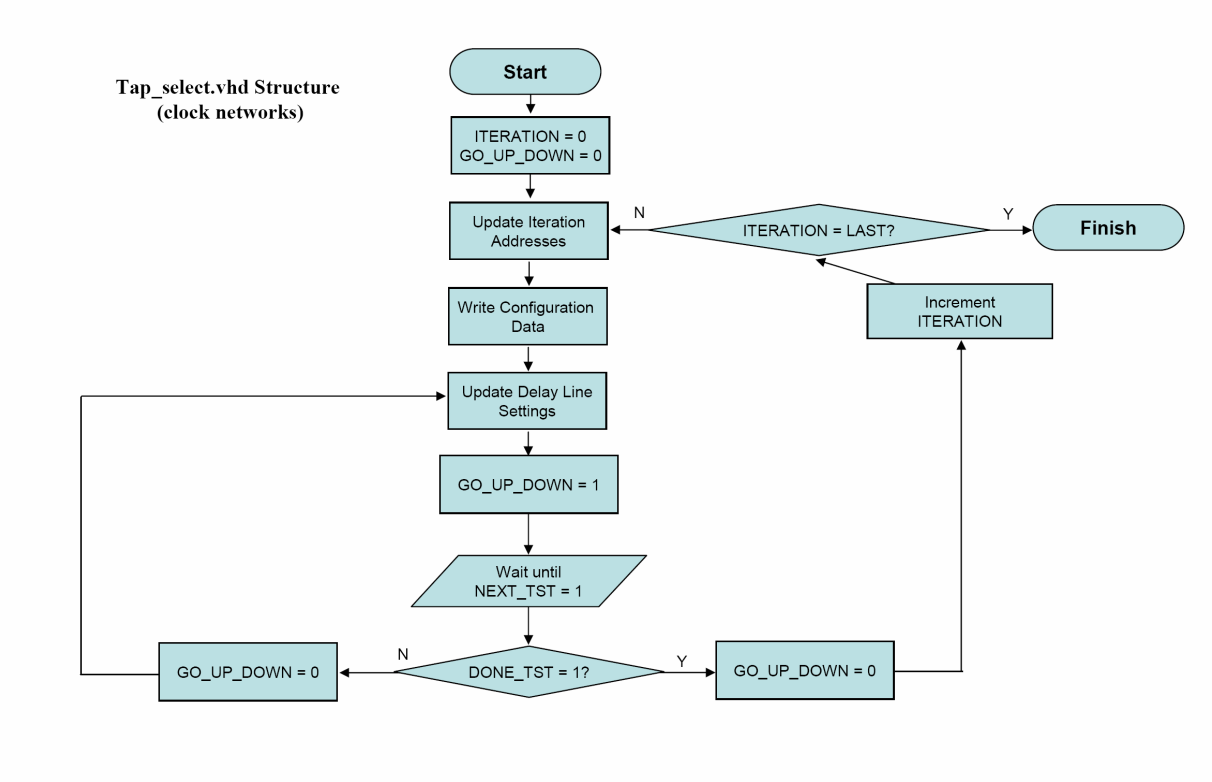

Figure 6.3: Controller component used to select and update taps in clock distributions.

in Figure 6.3 coordinates the activity of the controller. The first step is to configure the clock thread to send both the forward and reverse clocks to the first tap to be synchronized. The speed of the controller clock can be less than the frequency of the thread clock since the *UP* and *DOWN* signals are sampled and held by the *choice* block shown in Figure 6.4. The *UP* and *DOWN* signals only return to they neutral state once the new delay setting is programmed into the clock network and the controller is ready for the next test. The initial delay line settings are the minimum fine and minimum coarse settings for both delay lines in the tap. First, the coarse setting is adjusted according to the phase detector inputs (*UP* or *DOWN*) until an *UP* is found following a *DOWN* or a *DOWN* is found after an *UP*. While the coarse setting is modified directly by the *choice*

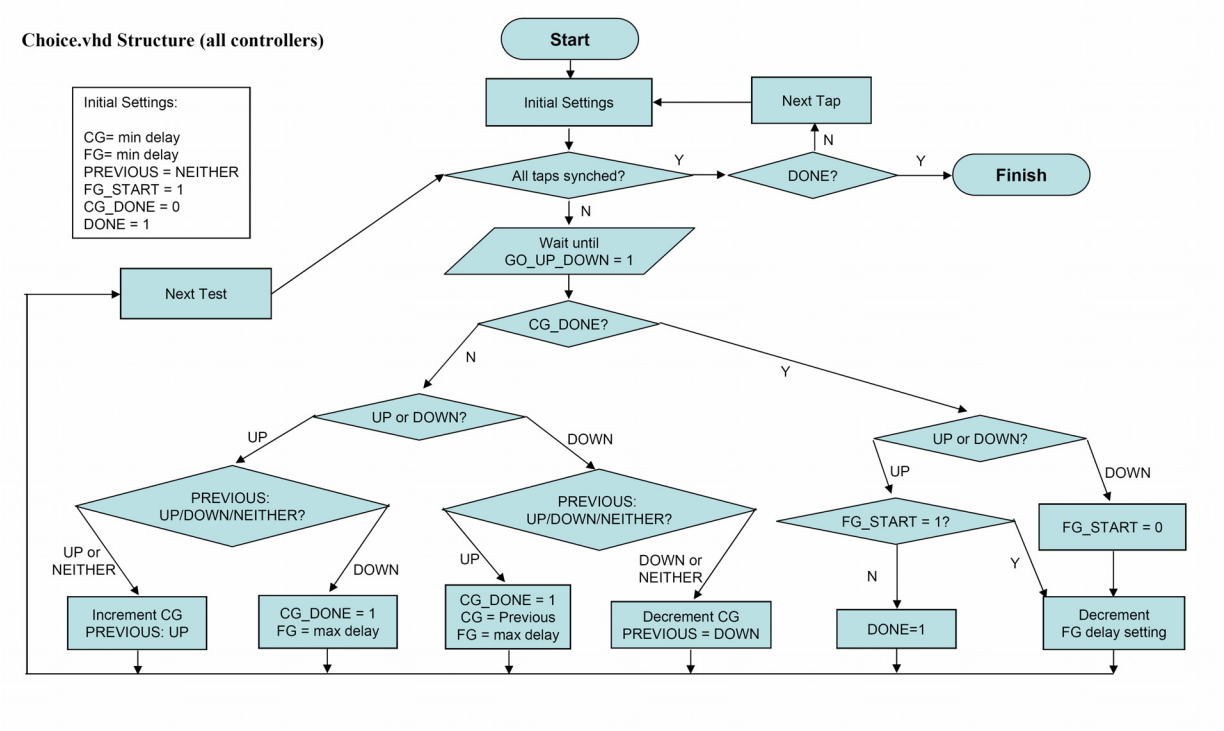

Figure 6.4: Controller unit structure used to choose delay settings.

block, the fine grain delay is controlled as the integer index to an appropriate ROM location. By decrementing or incrementing the fine grain address, you can decrement or increment the delay. As soon as the correct coarse setting is found, the fine setting changed to its longest delay setting. Since we are expecting the fine grain setting to be slower than the initial setting, the controller waits for the first *DOWN* to be signalled before looking for an *UP*. This has to do more with the nature of the phase detector model than the physical circuitry due to the use of the *mod* function. Once the *UP* condition is found, the tap is fully synchronized.

There is a two signal handshake between the *tap_select* and *choice* blocks to coordinate the activity: one to indicate that a new test is ready, and another to indicate

that the test is complete and to load a new delay setting. Once the delay values for the next test are loaded, the *UP* and *DOWN* signals are sampled and held once again. Once the tap is completely synchronized, *tap_select* reconfigures the circuit to bypass the second delay line in the current tap (halving the delay) and reconfigures the clock thread to feed both forward and reverse reference clocks to the next tap. This process will continue until all taps are synchronized.

6.4.2 Reconfigurable clock domain controller

The reconfigurable controller is constructed by expanding the design of the single clock controller. The primary difference is that *tap_select* instances are synthesized for each clock thread. The controller uses a *DONE* signal from each *tap_select* component to begin the synchronization of the next clock thread. The functionality of these *tap_select* instances could be accomplished using a single unit, but the approach makes the system more modular and provides added visibility of the circuit for testing purposes. The *choice* block can be reused for every clock thread since it written to work with any tap, using input parameters to modifying tap-specific configurations. While the controller is scalable, the more alternatives allowed (number of taps, number of clock domains and number of clock taps per domain), the more resources are required for the controller. To use system resources more efficiently, the controller should be designed for the expected clock mesh structure using the maximum number of clock domains and the maximum number of clock taps per domain as parameters. Similar to the single clock

network controller, there are *4+a+d* bits required to communicate between the controller and clock network, Figure 6.5. The structure of the controller is shown in Figure 6.6.

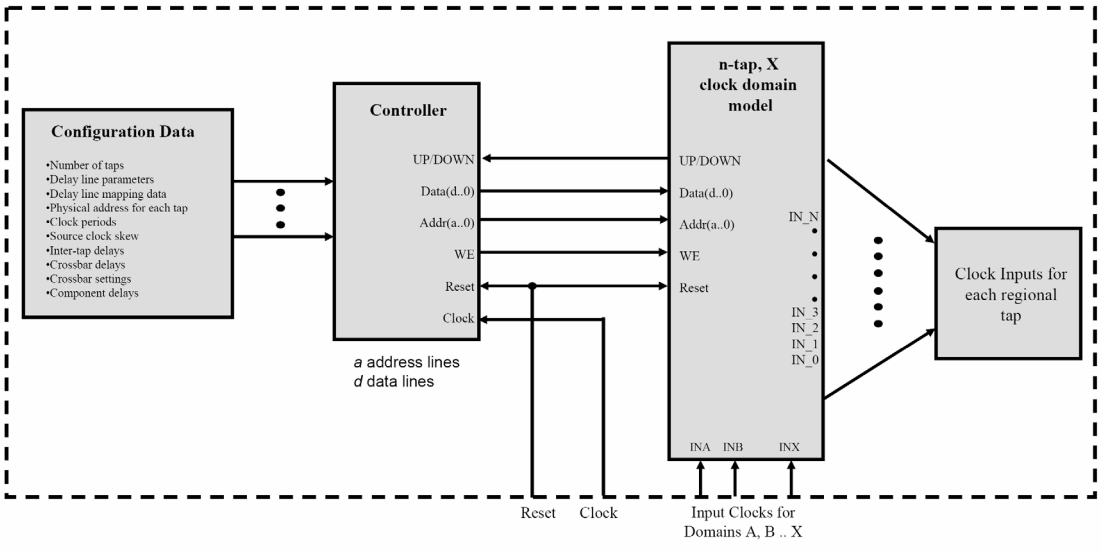

Figure 6.5: The multi-clock, *n*-tap clock distribution controller and model.

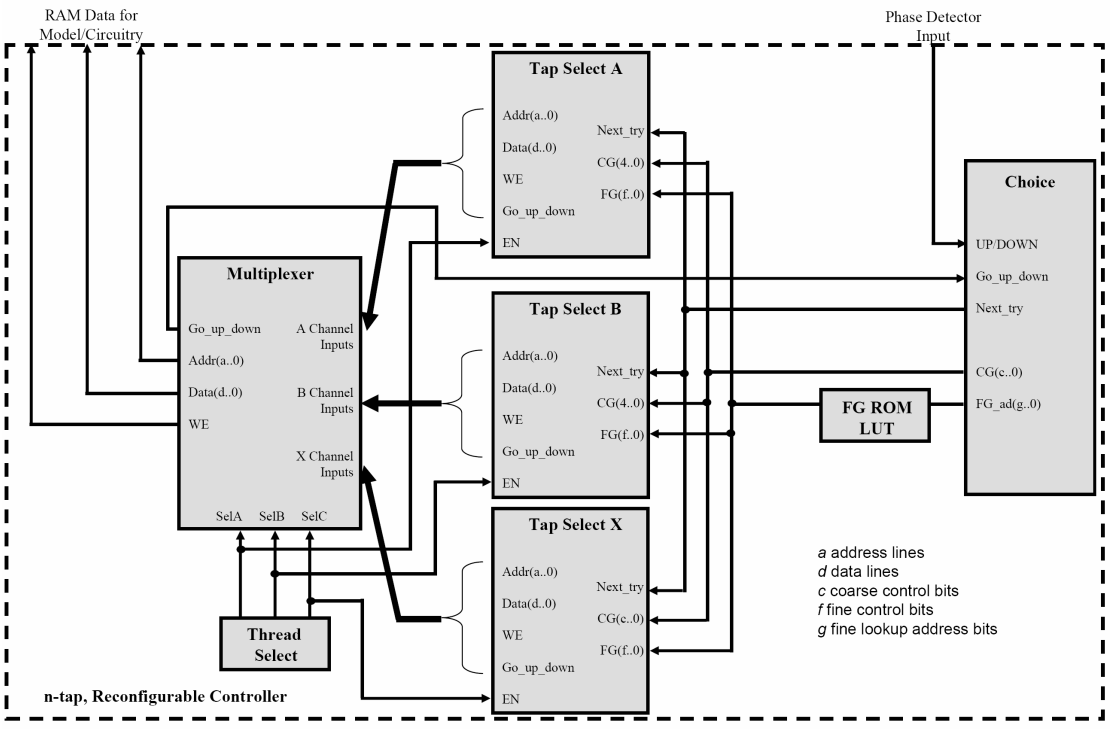

Figure 6.6: Controller used for the multi-clock, *n*-tap reconfigurable clock distribution.

6.4.3 Built-in clock skew system controller

The built-in clock skew system controller requires the same *4+a+d* bits to communicate between the components, Figure 6.7. There are only slight changes in the overall architecture and the configuration data required, Figure 6.8. The overall approach of the BICSS controller is different than the ones used for the clock networks. Where previously, the delay lines are controlled as a pair, here they need to be controlled individually, leading to longer memory write stages between tests. In addition, there needs to be four distinct operating stages in the controller to complete the skew compensation: the first two stages normalize the delay between the taps (*A* and *B*) and the central circuitry, the third determines the amount of clock skew present and the fourth is a write-back stage where this skew information is written back to the taps under test to perform the skew compensation. This requires changes to the BICSS controller coordination component *tap_select*, shown in Figure 6.9.

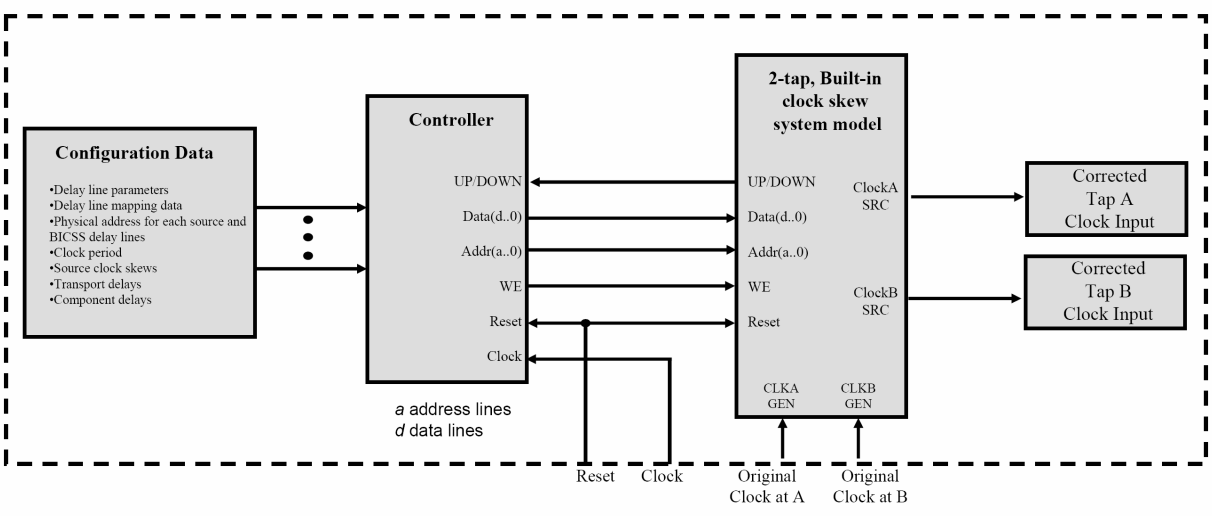

Figure 6.7: The built-in clock skew system model and controller.

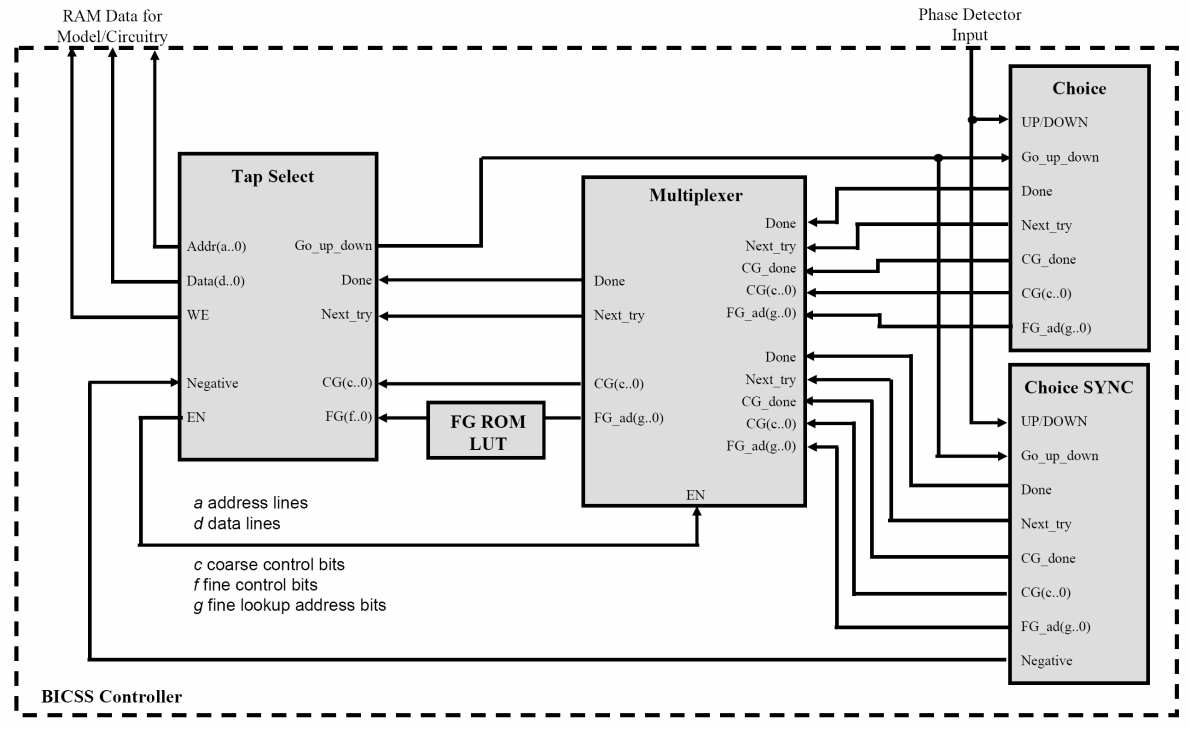

Figure 6.8: The controller used for the Built-in clock skew system.

During the two normalization stages, the delay generator *choice* operates like the *choice* component describes previously. However, to perform clock alignment in the fourth stage, there is no averaging involved, just a synchronization to align the two normalized tap clock signals at the BICSS block. Only one delay line is modified at any given time and depending on which signal leads and which signal lags, the *UP* and *DOWN* inputs can both equate to an increase or a decrease in the delay setting. The difference is an *UP* increases the delay setting when delay line *2* is being modified and a *DOWN* increases the delay setting when delay line *4* is being modified. This change requires the addition of an alternate *choice* block, called *choice_sync*, which is active for the clock skew compensation phase of the controller operation. A multiplexer is used to choose between the signals for *tap_select*. *Choice_sync* replicates *choice* twice, once to

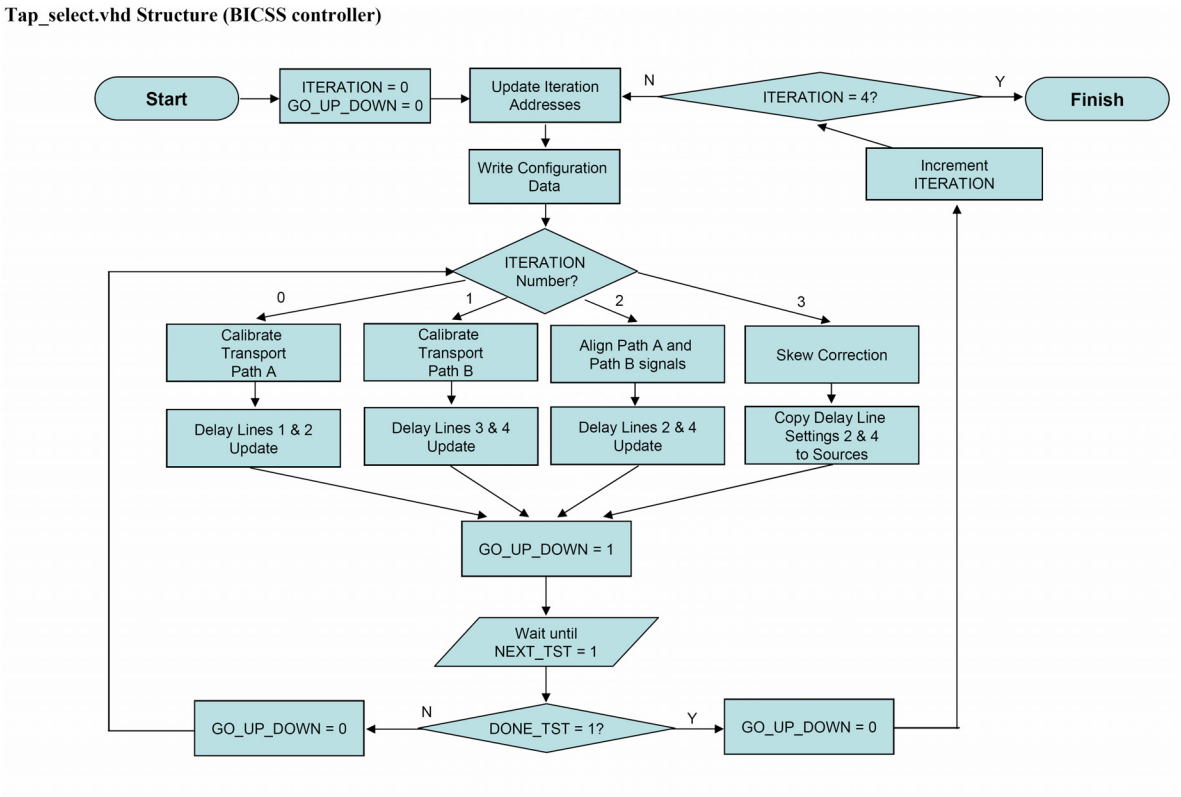

Figure 6.9: Controller component *tap_select* used in the Built-in clock skew system.

choose *delay line 2* values, and once to choose *delay line 4* values. The *NEGATIVE* signal from *choice_sync* is used to determine which delay line (*2* or *4*) is being updated. *Choice_sync* begins at the longest coarse delay setting of *delay line 2* and decrements the coarse setting until an *UP* is found, or the shortest delay setting is reached for this delay line. If no *UP* is found, *delay line 4* is sequentially incremented until an *UP* is found. Once the coarse delay is set, the fine delay is set to its maximum setting and sequentially reduced until an *UP* is found.

6.5 System models

The delay lines are modelled using SpectreS data from the extracted circuit. The total delay through a delay line consists of a fixed latency and a variable delay component. The delays are normalized to eliminate the fixed component. There are 188 fine grain delay settings spanning between 0 and 85 ps; the settings are also rounded to the nearest picosecond and using 8-bit control leading to redundant fine delay settings in the model. The possible fine delay settings for each delay line are shown in Figure 6.10; the horizontal portions of the curve represent repeated settings that if eliminated, could reduce the required synchronization time for each test. The coarse grain settings are also representative of the silicon circuitry and allow 6 distinct settings (including zero), each with an 85 ps difference with respect to its adjacent setting, encoded using 5-bits. The total number of control signals for each delay line is 13.

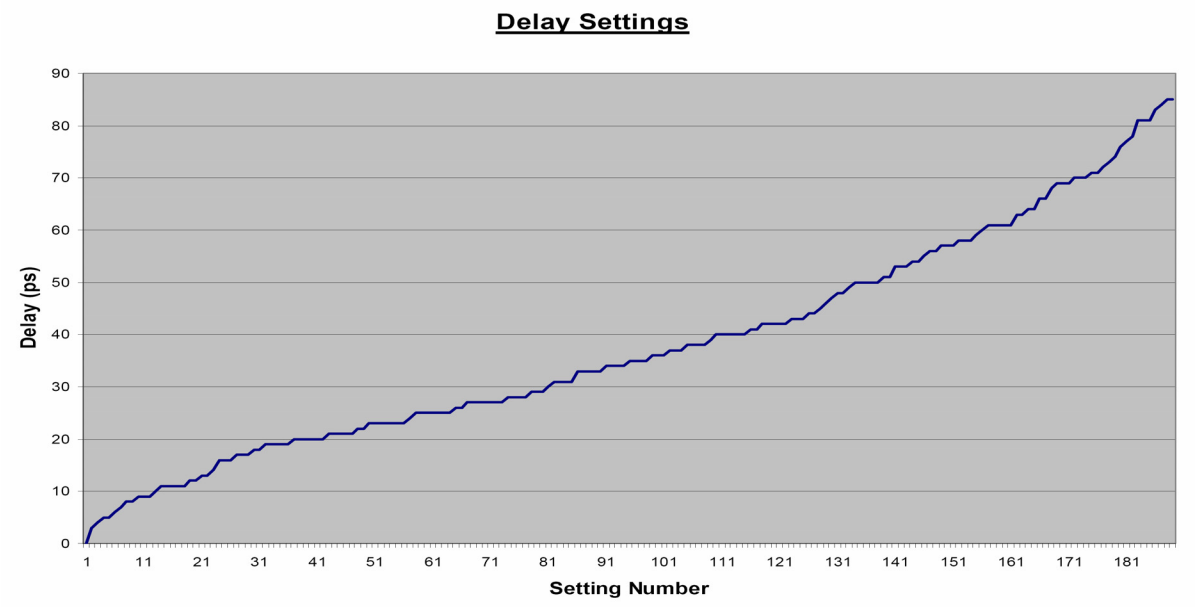

Figure 6.10: Delay setting range for a single fine grain delay block.

6.5.1. Single clock model

The structure of an *n*-tap single clock domain model is shown in Figure 6.11. The configuration data is generic and programmable to model different variations in layout. The clock period used, the delay through the multiplexers and the interconnect delay

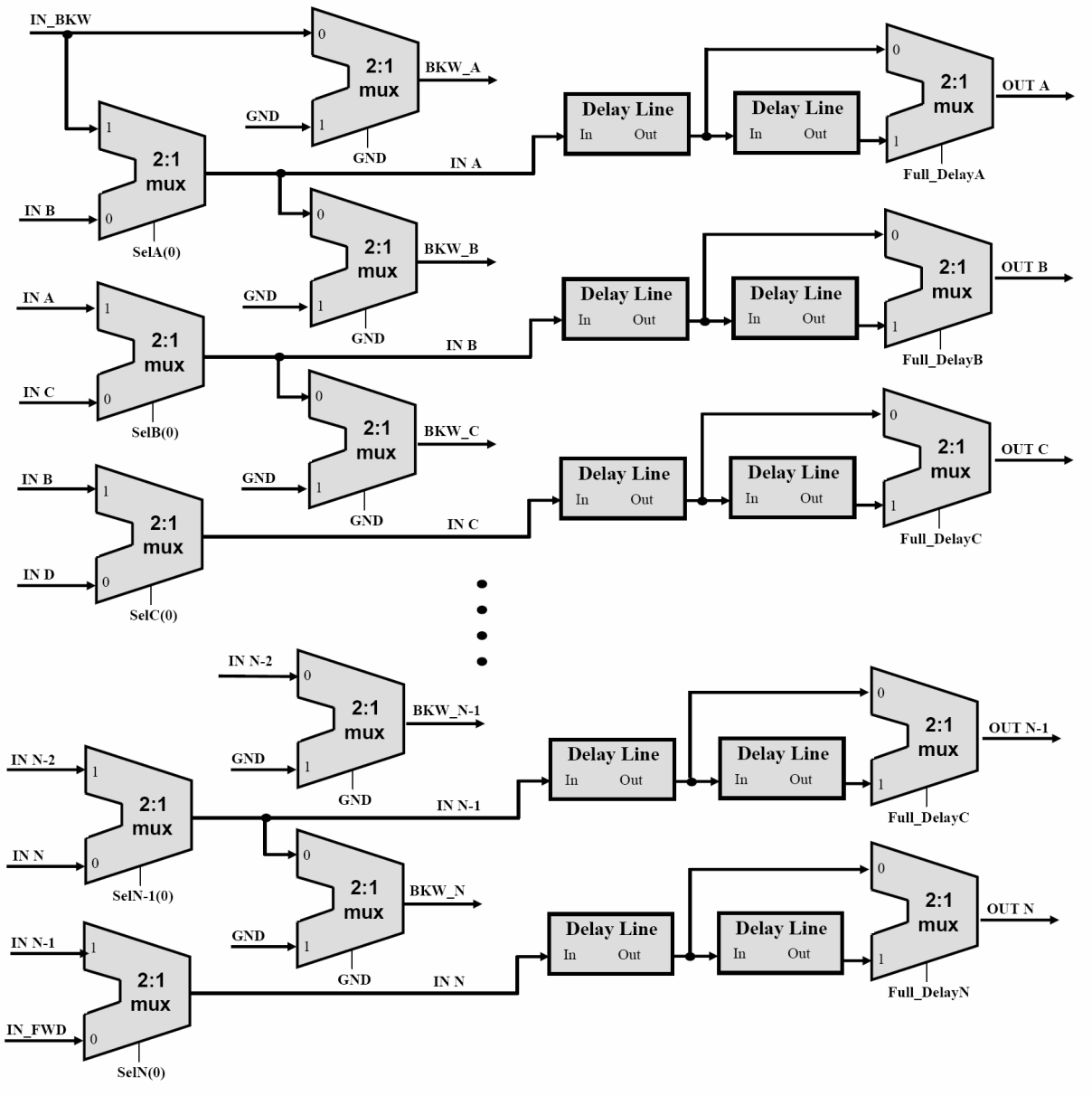

Figure 6.11: The *n*-tap clock distribution model.

between taps are all variables that can be controlled in the model. To simulate properly in HDL, a *clock* signal is added to the clock network model although the extracted circuit does not require such a signal. There are three significant components in a single clock network model: The clock thread, the delay lines and the phase detector. A pair of independent forward and reverse clock threads are used with a *2n* delay configuration, but bi-directional configurations can be modelled by appropriately setting the device and interconnect delay parameters. The phase detector works by comparing the parameterized interconnect and device delays to the current delay line setting for the forward path and comparing it to the expected delay of the reverse path. The calculation is made for every tap and only the tap currently being synchronized is forwarded to the controller, similar to the circuit design of clock network.

6.5.2. Reconfigurable clock network model

To create a reconfigurable network, first the clock mesh network needs to be constructed with switches of arbitrary size (b_i ports) and n taps. The number of clock frequencies and the maximum number of taps per domain also need to be set. For the most generic solution, the maximum number of taps per domain and the number of clock domains can both be set to the total number of taps. The clock network synchronizer can then be generated. Figure 6.12 shows one possible section of mesh network containing switches with 4 and 8 ports, routing 2 and 4 clocks, respectively. Each port can be uniquely connected to any other port in the switch. The horizontal and vertical lines are

labelled using the row/column number and the segment number along the row/column starting from the top most corner. The switch ports are alphabetically labelled from the right-most port on the top edge of the switch, so from *A* to *D* for a 4-port switch, *A* to *H* for an 8-port switch and so on. For the top and left switch ports, the inputs lines end in *A* and the output lines end in *B*. For bottom and left ports, the opposite is true.

Once the controller and circuit model are generated, the path for each clock thread is established by writing to switch configuration memory in the clock network. The tap synchronization order is then followed as programmed into the system to synchronize every tap in a thread. For the tap being synchronized, all of the switch direction control bits are set to *forward* controls for the switches preceding the tap and to *reverse* for the switches following the tap. Between tap synchronizations, the signal direction for the thread must be modified from *reverse* to *forward* for every switch between the current tap *i* and *i+1*. For the reconfigurable clock network model, the interconnect and switch delays between consecutive taps in a domain must also be included as parameters for every clock thread (domain). Multiplexers are added at each clock tap to select which wire carries the forward and which wire carries the reverse reference signals. Since the *UP/DOWN* comparison is done algebraically instead of on the waves themselves, the reverse clock multiplexers can be omitted in the model.

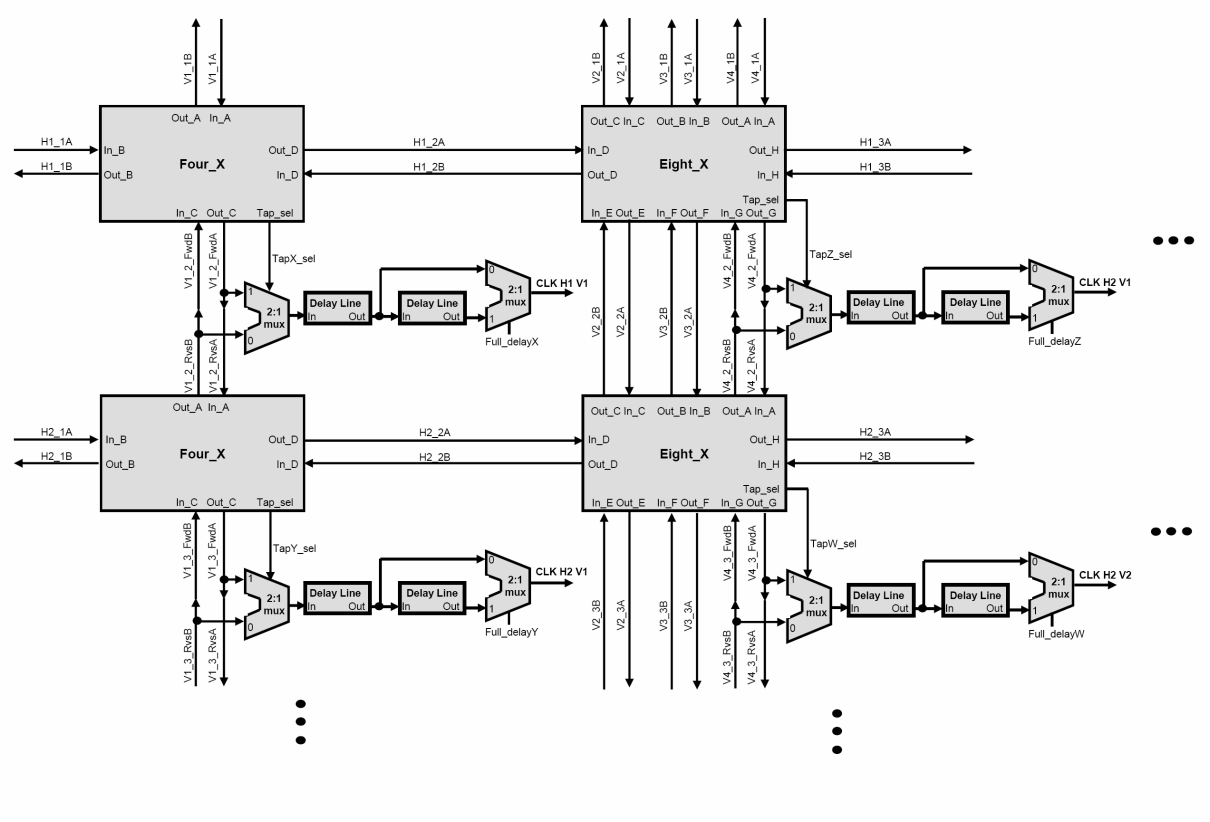

Figure 6.12: An enlarged section of a multi-clock mesh.

The functionality of the phase detector model is divided into two components: *up_down* and *choose_vals*. *choose_vals* calculates the forward and reverse propagation delays using the parameterized delay variables and the *up_down* block determines the current tap's forward and reverse path delays. The required *UP* and *DOWN* signals can be generated by comparing the two path delay values. Where *up_down* operates on the actual tap in question, c*hoose_vals* operates on each thread from first tap to last based on the programmed order of synchronization. The behaviour of the phase detector circuitry is much simpler since it can easily compare the forward and reverse clocks without having to infer the forward and reverse path delay from the configuration settings.

6.5.3. BICSS model

BICSS requires a primary central block located where convenient on an IC and small secondary distributed circuits located at each tap under test, consisting of a multiplexer and a delay line. The central block, shown in Figure 6.13, consists of four delay lines, 3 multiplexers and a phase detector. Delay lines *1* and *2* are used to compensate for the transport delay from clock tap *A* to the primary block using *SRCB_FB* as the source clock. Delay lines *3* and *4* are used to compensate for the transport delay from clock tap *B* to the primary block using *SRCA_FB* as the source clock. Once the paths are normalized, delay lines *2* and *4* are used to determine the skew between the clocks at tap *A* and *B*. Should BICSS be used on a system with more than two taps, the skew detection, measurement and compensation would need to be

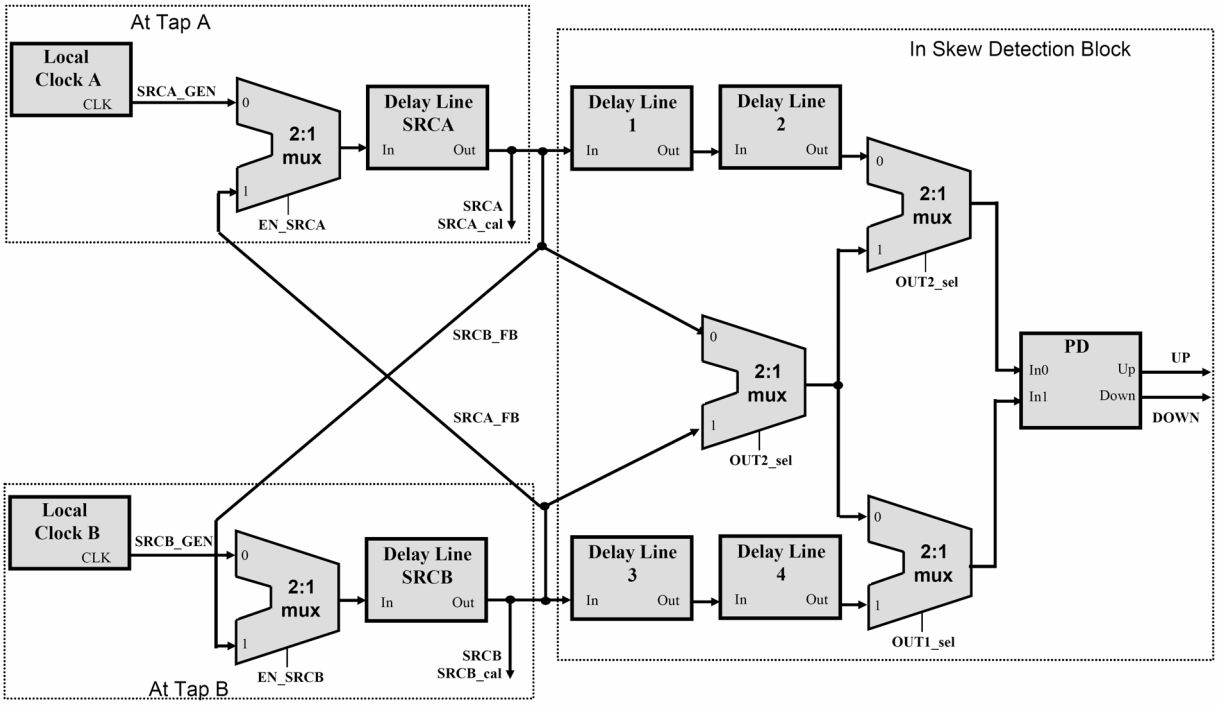

Figure 6.13: The built-in clock skew system model.

performed through a multiplexer which would choose between all the clock signals entering the BICSS block.

Since the only task for the test clock is to drive the round trip path from the block to the tap and back, the same clock source can be used for all normalizations, provided that the clock operates at the same frequency as the tap clocks being observed. Once normalized, the clock taps are compared pair-by-pair. To compensate for the first transport delay from tap *A*, *In0* of the phase detector is the roundtrip delayed test clock and *In1* is the undelayed test clock. When compensating for the transport delay from tap *B*, *In1* of the phase detector becomes the roundtrip delayed clock and *In0* is the test clock. The third mode compares both tap clocks transported to the BICSS block and delayed by the evaluated transport delay of their counterpart. The transport delay compensation uses *delay lines 1* and *3* and the skew compensation uses *delay lines 2* and *4*.

6.6 Operating behaviour of the systems

The models and controllers discussed in this chapter have all been simulated using ModelSim-Altera Edition. As this is a behavioural simulator, clock-to-output delays, propagation delays and minimum clock period are not incorporated into the simulations by default. This allows the propagation and transport delays, and the behaviour of the delay lines present in the models to be easily changed. As a result, the model can reflect the behaviour of different layouts and unevenly spaced taps. It can also use different

components like crossbar switches and multiplexers with different operating characteristics like signal latency and propagation delays. The simulations assume that the root clocks are generated externally to the distribution or testing circuitry. The simulations waveforms are shown in Appendix A.

6.6.1 Single clock domain system

A four tap clock network model and controller was designed using the approach described in this chapter. This 4-tap system using a 13-bit delay line requires 64-bits of memory considering the 16-bits of memory required per tap. A 15-tap system would require 240 bits using the same 13-bit delay line. A 6-bit word was chosen for the memory word size. Since 3 memory locations are required per tap and 12 memory locations in total, 4 memory address bits are needed. However, for additional flexibility, 6 memory address bits are used in the simulation. With standard optimization settings for speed and area, the controller was synthesized using 202 LUTs, 74 registers and 2048 memory bits to store the delay line characteristics in an Altera Stratix II device. The computed maximum frequency of the controller is 314.87 MHz. The number of clock taps is limited to 10 for this implementation. A higher number of taps would require more registers and larger look-up-tables in the FPGA. The four tap controller and clock model was simulated with a controller clock of 1.0 GHz and a thread clock of both 1.0 GHz and 1.5 GHz. There is no relationship needed between the two clock frequencies. In a typical configuration, the controller will be located off-chip and operate at frequency in the

megahertz range, as is typical of a microprocessor or FPGA. The clock frequency of the clock threads are limited to frequencies above 1.0 GHz due to the maximum delay of the variable delay lines used in this proof of concept. These delay lines are scalable and could be extended to be use with lower clock frequencies, as needed. The functionality of the delay lines could also be improved by allowing inverted outputs, as is the case with the silicon implementation of the design, halving the minimum required frequency.

The synchronization time required by the system is not fixed since it is dependent on the number of tests needed to reach the appropriate setting. The complete synchronization waveforms for the 1.0 GHz thread clock in Figure A.1 and the 1.5 GHz thread clock in Figure A.5 show that the synchronization time for each tap can vary greatly. Overall, the 1.0 GHz simulation with typical delay parameters requires under 9 μs (3 μs at the synthesized frequency) to complete and the 1.5 GHz simulation requires under 7 μs (2.3 μs at the synthesized frequency). In these simulations, the *TAP_NUM* signal changes from 1 to 4 as the tap being synchronized changes from *D* to *A*, respectively. The *VALUE2* signal represents the target delay (reverse path delay) for the given tap and configuration and the *VALUE1* signal represents the forward path delay incorporating the delay line settings. The *up_down* component uses these values to generate the correct *UP* and *DOWN* signals until these numbers match. For the currently active tap *X*, the *CG_X* and *FG_X* signals represent the current delay setting being evaluated.

The initialize phase for the 1.0 GHz system is shown in Figure A.2. Note how the initial tap and thread settings are written into memory for the four taps using the *ADDR*, *DATA* and active high write enable (*WE*). The address and data are held constant for one clock period before and one clock period after the write assertion to prevent erroneous memory writes. The synchronization of the first tap (tap *D*) is shown in Figure A.3 for the 1.0 GHz thread clock. Notice the *CG_D* setting and *VALUE1* delay change to find the correct coarse setting up to 200 ns, and the *FG_D* setting and *VALUE1* delay change to find the correct fine setting after 200 ns. Figure A.4 represents the initial and final picture for the synchronization of the tap clocks. *OUT_XH* represents the forward clock delayed through only one delay line. The alignment in Figures A.2 and A.3 were performed using both delay lines. This is the averaging required to achieve synchronization. Similar to the silicon implementation, once the clock edges are aligned, the system requires a calibration phase to invert clocks as required to achieve consistent clock polarity. For the 1.5 GHz clock thread, Figure A.6 shows the synchronization of tap *B* and Figure A.7 represents the initial and final picture for the synchronization. The accuracy of the synchronization is the maximum increment between delay settings (3 ps in our model) due to the digital nature of the delay line and the idealized behaviour of the phase detector.

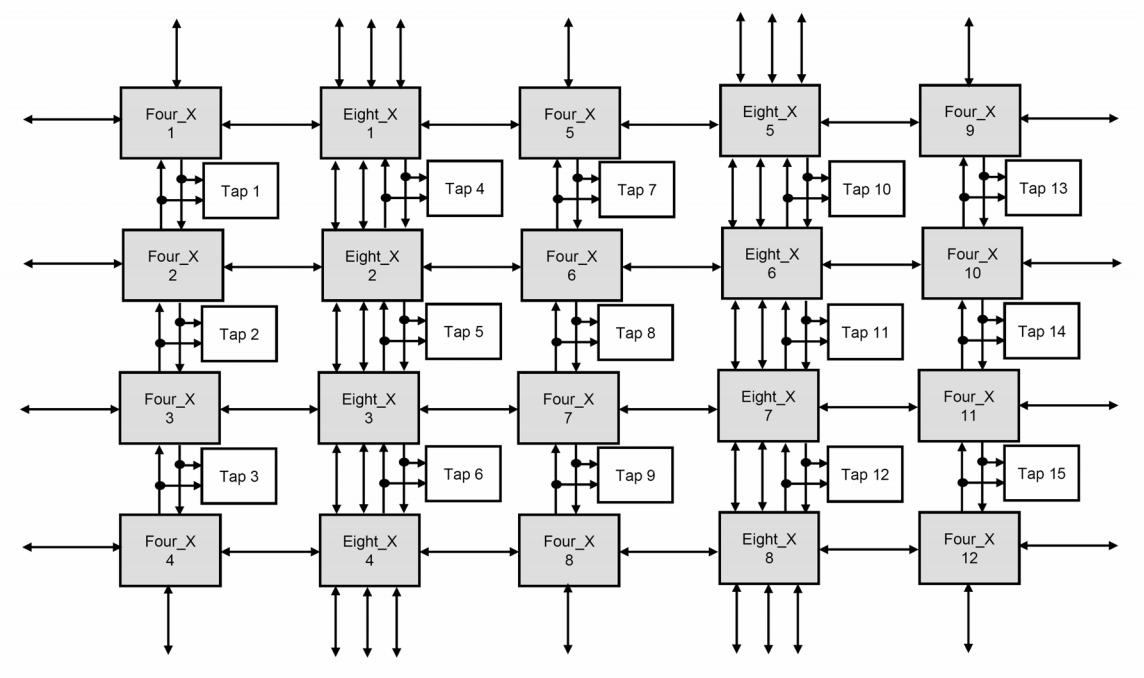

Figure 6.14: The multi-clock domain, 15-tap reconfigurable clock network model.

6.6.2. Reconfigurable clock domain system

To show the functionality of a reconfigurable clock network controller, we simulated a 3 clock domain, 15-tap model and controller, shown in Figure 6.14. It is constructed by elaborating upon the single clock domain model. Like the single clock model, each clock domain is limited to 10 taps. More taps and more clock domains can be added, but this will affect the size and speed of the controller. This limitation does not significantly affect the reconfigurability of the clock infrastructure as shown in Table 6.1. Two scenarios are explored for the number of possible configurations that must be excluded due to the 10 tap per domain restriction: one where all taps are connected: Equation 6.8, and one where taps can be left unconnected: Equation 6.9.

$$\text{Excluded configurations (case 1)} = 3 \cdot \sum_{n=0}^{4} 2^n \cdot \binom{15}{n} \quad (6.8)$$

$$\text{Excluded configurations (case 2)} = 3 \cdot \sum_{n=0}^{4} 3^n \cdot \binom{15}{n} \quad (6.9)$$

	Every tap occupied	Unconnected tap allowed
Total Cases	3^{15}	4^{15}
	14 348 907	1 073 741 824
Realizable Configurations	14 271 114	1 073 370 301
Excluded Configurations	77 793	371 523
Percent Excluded	0.54%	0.03%

Table 6.1: Number of possible clock domain configurations.

For this 15-tap configuration, 272 bits are required for the tap memory. Using 6-bit words and 3 words per tap, 6 address lines are required. In addition to this, a mixed switch mesh with eight 8-port switches and twelve 4-port switches requires 160 encoded bits and 288 decoded bits of memory. Without any optimization for speed or area, the controller for this reconfigurable network was synthesized using 447 LUTs, 127 registers and 2048 memory bits to store the delay line characteristics in an Altera Stratix II device. The computed maximum frequency of the controller is 243.96 MHz. The fifteen tap reconfigurable controller and clock model were simulated using a controller clock period of 100 ps and a sample configuration using thread clock periods: 900 ps for thread *A*, 750 ps for thread *B* and 600 ps for thread *C*. The fifteen taps were divided as in Figure

4.6, so with the taps numbered column-wise from left to right, thread *A* was connected to taps 1, 2, 3, 5, 6 and 10, thread *B* was connected to taps 11, 12 , 14 and 15 and thread *C* as connected to taps 4, 7, 8, 9 and 13. The complete synchronization for all 15 taps is shown in Figure A.8. The *TAP_COUNT* signal represents the number of taps already synchronized, the *TAP_NUM* signal represents the exact tap currently being synchronized and the *COUNTX* signals represent the state of each *TAP_SELECT* component, with 0 being the initial state and 26 being the final state. Figure A.9 shows the synchronization of the six thread *A* taps. The output waveforms *OUT_X* are listed in the waveform in the order in which they are synchronized. Since the model only applies the delay setting once all the taps in the thread have been synchronized, the *OUT_X* waveforms before time 1151 ns show the initial state of the tap clocks and those after 1154 ns show the synchronized end result. Figure A.10 shows the synchronization of the four thread *B* taps. The *OUT_X* waveforms before time 2096 ns show the initial state of the tap clocks and those after 2099 ns show the synchronized end result. Figure A.11 shows the synchronization of the five thread *C* taps. The *OUT_X* waveforms before time 2840 ns show the initial state of the tap clocks and those after 2843 ns show the synchronized end result. The 15 synchronized tap clocks can be seen in Figure A.12.

6.6.3. Built-in clock skew system

The controller for *BICSS* was synthesized using 394 LUTs, 140 registers and 2048 memory bits in an Altera Stratix II device. The computed maximum frequency of

the controller is 269.54 MHz. The simulation uses a 100 ps controller clock period and 1 ns tap clocks with a 50 ps source-to-tap *A* skew, a 20 ps source-to-tap *B* skew, and 135 ps and 95 ps transport delays, respectively, from tap *A* and *B* to the central BICSS block. These numbers can be changed as required to reflect any realizable circuit configuration. The complete calibration and synchronization cycle is shown in Figure A.13. Delay lines *1* and *2* are both updated while *TAP_NUM = 0* until the transport delay from tap *A* is determined. This delay is then halved by resetting delay line *2* to its initial value. The same is true for *TAP_NUM = 1* for the transport delay from tap *B* using delay lines *3* and *4*. For synchronization, delay line *2* or *4* is modified as needed, depending on the configuration. *VALUE1* represents the target delay for calibrating tap *A* and *VALUE2* represents the target delay for calibrating tap *B*. In the case of this test case, delay line *4* needs to be modified and *VALUE2* represents the target delay.

The calibration of the tap *A* clock path to the BICSS circuitry is shown in full in Figure A.14 and the enlarged final stage is shown in Figure A.15. Note how the *PD_1_OUT_CALA* and *PD_2_OUT_CALA* signal achieves alignment between times 177 and 178 ns before *TAP_NUM* switches to 1. The calibration of the tap *B* clock path to the BICSS circuitry is shown in Figure A.16 and the enlarged final stage is shown in Figure A.17. Note how the *PD_1_OUT_CALB* and *PD_2_OUT_CALB* signals achieve alignment at 753 ns before *TAP_NUM* switches to 2. Both delay lines *2* and *4* are written to during an update, however only one of their values is updated each time. The synchronization is highlighted in Figure A.18 and the enlarged final stage is shown in Figure A.19. Alignment is achieved at 1085 ns as shown by the *PD_1_OUT* and *PD_2_OUT* signals.

The final stage of the BICSS scheme is to write-back the delay line data to the clock taps to compensate for the determined clock skew. Figure A.20 shows this write-back stage and the clearing of the delay line *2* and *4* settings in the central BICSS circuitry. The transport delay calibration setting is kept in delay lines *1* and *3*. As such, the *SRCA* and *SRCB* signals reflect the clocks at each tap and are synchronized for their respective source-to-tap delays. The signals *PD_1_OUT* and *PD_2_OUT* reflect the transport delay compensated tap clocks at the BICSS block and should also be aligned after synchronization. As such, the system can monitor clock alignment for a pair of clock taps (tap *A* and *B* in this model) using the BICSS phase detector during operation and clock skew can be detected and potentially corrected.

6.6 Conclusion

The three system/control configurations discussed in this chapter represent two potential clock networks: a fixed single clock variant and a fully reconfigurable three clock variant, and the built-in clock skew system introduced in Chapter 5. Each of the simulations includes a model created using extracted circuit information from a 180 nm TSMC process design and a synthesizable controller in VHDL necessary to operate the circuitry. The hardware that the models represent is designed to be versatile enough to be controlled by an internal controller in silicon, or an external one using a microprocessor, an FPGA, or any other hardware or software approach. The results show that the controller/circuitry interface is simple yet versatile enough to be capable of

operating the circuitry easily and efficiently. The majority of the required communication bits are dedicated to the on-chip configuration memory. The parallel nature of this interface could be replaced by a serial one similar to JTAG, or even JTAG itself, thereby greatly reducing the I/O requirements for synchronization. The resources required to implement the controllers and the time required to perform the synchronization and calibration operations are also reasonable and show that the designs can be practically implemented in a real world system. The modelling of the clock switch meshes allows a variety of clock configurations to be tested to determine the required synchronization time and to predict the ideal delay setting settings. These settings can be pre-loaded into the system speeding up synchronization time when used in silicon. This additional predictive step could greatly simplify the complexity and the synchronization time of the clock network controllers.

Chapter 7:

Core circuit components

7.1. Introduction

While the design and operational overview of the dual reference signal based averaging clock networks and the built-in system for online debug and repair have been discussed in previous chapters, this chapter will discuss the detail behind the circuitry involved in creating these structures. While the systems are designed to be appropriate for a number of on-chip applications and technologies, it is important to get an idea about the area and performance that can be achieved in a mature technology. The idea was not to get the highest performance possible, but to get a realistic view of an average case implementation. Both higher and lower frequencies could easily be achieved using

modified circuits and lengthened or shortened delay lines. If used in a newer technology, both higher clock speeds and smaller circuitry could be obtained. Since mismatch is a greater issue in these newer technologies, the performance of the circuitry could change. The active skew compensation and skew detection circuitry is an approach that will overcome variation much better than traditionally used passive approaches since the system is only susceptible to mismatches between components that are closely spaced. The net variation effect will be smaller than clock trees using a distributed approach. The circuitry is entirely digital with digital control to create the easiest interface possible. Digital circuits are also small in area compared to their analog counterparts and are much easier to port to different technologies. Digital circuits are also much less sensitive to matching than analog circuits since analog circuit techniques rely on similar closely matched components for correct operating behaviour [117].

All of the circuits required to construct the systems were designed and laid out using TSMC's 180 nm standard process. Clock frequencies between 500 MHz and 2 GHz were targeted because this represents a typical frequency range in this technology for ASICs, microprocessors and FPGAs. Since the routing circuitry and control signals are designed to apply to the clock distribution network, novel circuit techniques are required to perform the required task and to communicate with a synchronous controller. The circuitry's ability to function asynchronously an important consideration in justifying the feasibility of the clock networks and the clock skew detection system. There are a number of circuits that achieve better performance or have novel functionality that were

developed in creating the proof of concept systems. Both the novel circuitry and the circuitry that directly affects the performance or practicality of the systems are described in the following sections.

7.2. Delay lines

The delay line is the component that has the single greatest effect on the performance of the system since the attainable skew bound is directly related to the minimal delay increment, or resolution between settings. Typical digital delay lines have large delay increments between settings, which make it difficult to perform accurate signal alignment. Analog delay lines have static power consumption are require relatively large devices that could hurt the practicality of our multi-tap, multi-delay line system [118]. A number of different configurations were investigated and a dual stage, coarse and fine grain approach was selected to get good resolution and a wide delay range. The result is a delay line with a fixed latency component (D) and a variable delay component (δ). The minimum frequency that can be distributed is a function of the total achievable variable delay:

$$F_{\min} = \frac{1}{T_{\max}} = \frac{1}{2 \cdot \delta} \qquad (7.1)$$

The two times factor comes from the ability of the delay line to output both an inverted and a true version of the input. This doubles the effective delay achievable by the delay

line and is only possible since the component is designed for clock signals with 50% duty cycle. The variable delay line should have good linearity between potential delay settings. Here, the maximum delay increment between adjacent settings establishes the worst-case clock skew of the CDN or the resolution of the skew detection circuitry. The delay line implementation needs to be as small as possible and should have a minimum of control lines since it needs to be replicated for each tap in the distribution. The resulting signal should also have full swing outputs with balanced pull-up and pull-down behaviour.

7.2.1 Coarse grain delay lines

A number of different structures were tested before choosing the design shown in Figure 7.1. The first one had a set of even numbered inverter chains (0, 2, 4, 8, 16) with input and output connections that allowed the input clock signal to be routed through the system exactly three times, choosing a different inverter chain for each pass. This method creates a constant delay overhead due to the routing and allows a digitally-controlled variable delay between 1 and 14 two-inverter delays. Each delay increment resulted in an increase of roughly 80 ps. The problem with this method is the total number of inverters required, the networking overhead and the approach's lack of scalability. The next coarse structure employed a ladder system using an inverter chain. Each inverter output is connected to the next inverter in the chain and to a multiplexer (or other device) that selects between all of the inverter outputs, or taps. Two varieties of

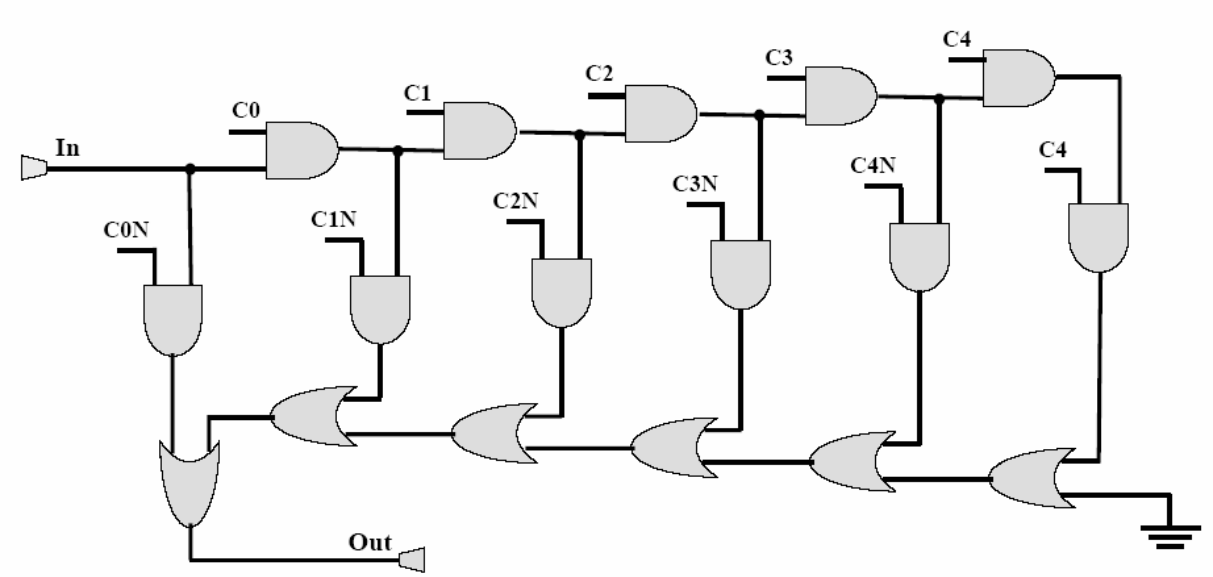

Figure 7.1: Coarse grain delay line.

multiplexers were tried, one using CMOS logic and one using pass transistors. This method was not pursued since the multiplexer prevents the system from being easily scaled. Additionally, the high capacitive loading on the pass transistor circuit and the large multiplexer fan-in of the CMOS variant slowed the signal transition limiting the operating frequency to below our target range. This design in Figure 7.1 was optimal for our application since the multiplexers are built into each coarse cell so the design scales better than a typical tapped inverter chain structure to allow for longer delays and lower frequencies where necessary. The design is easy to control and the amount of delay added between settings is also appropriate. The fact that signals entered and exited the component using the same ports was useful in maintaining consistent delay increments.

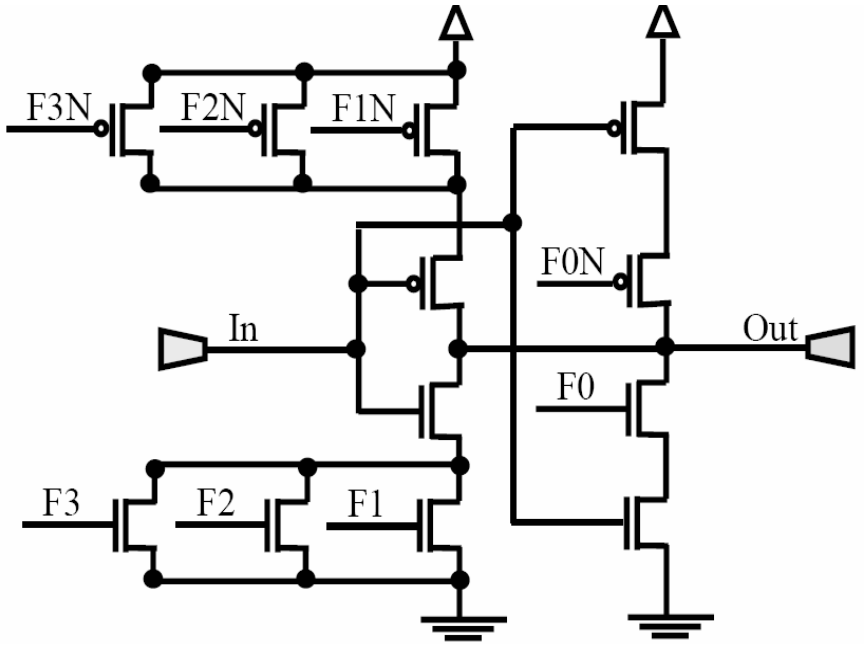

Figure 7.2: Fine grain variable delay inverter.

7.2.2 Fine grain delay lines

A fine grain delay element is needed to fill the gaps between coarse grain settings. To accomplish this role, we use the fine grain variable delay inverter shown in Figure 7.2 to achieve equal high-to-high and low-to-low delays, resulting in matching duty cycles for input and output clocks. A number of these fine grain delay inverters need to be serially connected together to bridge the delay between coarse grain settings. Allowing each fine delay inverter to be programmed individually creates the greatest number of delay settings and the potentially highest resolution, but also requires the most overhead. The number of groups of fine delay inverters that can be uniquely programmed (g) and the number of control lines per inverter (c) sets the achievable resolution of the delay line. The resolution of the complete delay line is:

$$resolution_{ideal} = \frac{coarse\_delay}{(2^c)^g} \qquad (7.2)$$

This is the potential resolution and not the actual one since some unique delay settings might map to nearly identical delays and some delay settings may exceed the required coarse delay. In a typical application with a 100 ps coarse grain delay, an arbitrary number *n* delay lines are controlled as a single group with 4 control lines resulting in a maximum attainable resolution of 6.25 ps. With two uniquely controlled groups, this number decreases to under 0.4 ps. Since the resolution of our coarse delay is in the same range as this example, we choose to have two uniquely controlled groups fore the fine delay. Since the delay range of our variable delay inverter is approximately 25 ps, this would require 4 inverters to span the coarse grain delay.

7.2.2.1. 1-1-1-2 foursome

Having established a fine grain delay line using 4 variable delay inverters in two groups, we explore next a number of potential orderings. This first attempt had three delay lines controlled as a group and a fourth controlled independently. This approach was discarded because the delay range was not sufficiently linear and the output signal had too much duty cycle variation in the clock signal.

7.2.2.2. 1-1-2-2 foursome

The next delay line ordering we tried uses the coarse delay line from section 2.1 and 4 current starved inverters grouped in pairs. Buffers or inverters are required to regenerate the clock edges that are softened by each fine delay stage. While this configuration, shown in Figure 7.3, behaved as expected, there was room for improvement in terms of the rise time and the duty cycle of the output signal.

7.2.2.3. 1-2-1-2 foursome

The third delay line ordering alternated delay line settings since this approach would have better matching and would contain similar signal paths in both halves of the delay fine delay block. In this configuration, shown in Figure 7.4, the output load of each corresponding fine delay inverters is matched. This is important since different output loads will have a nonlinear affect on the delay through a current-starved inverter since the drive strength changes for each delay setting. Simulations showed that this configuration resulted in more consistent signal characteristics, but the duty cycle continued to vary between different settings. This result is not ideal for our system, since the duty cycle change is averaged during synchronization, skewing the resulting clocks. This result is further explained by the fact that the variable delay inverter may affect rising and falling edges differently for a constant setting. Duty cycle shifts originate from

unequal propagation delays through the pull-up and pull-down portions of the fine delay inverter.

7.2.2.4. 2-1-1-2 foursome

The fourth configuration, Figure 7.5, corrects the duty cycle drift by feeding each signal through two identically set fine delay inverters separated by non-inverting logic

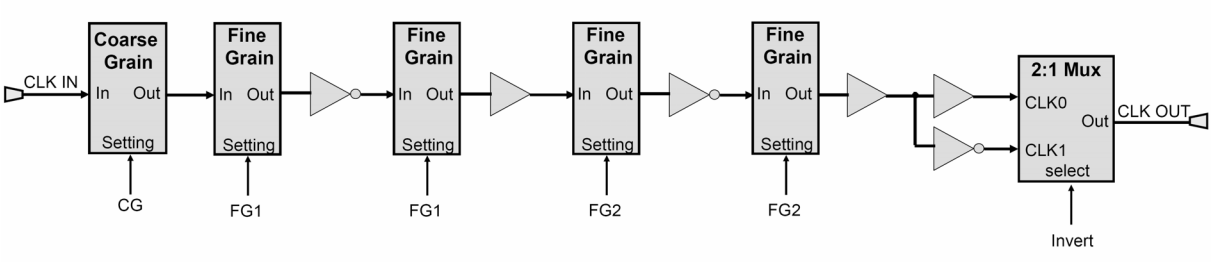

Figure 7.3: 1-1-2-2-fine delay configuration.

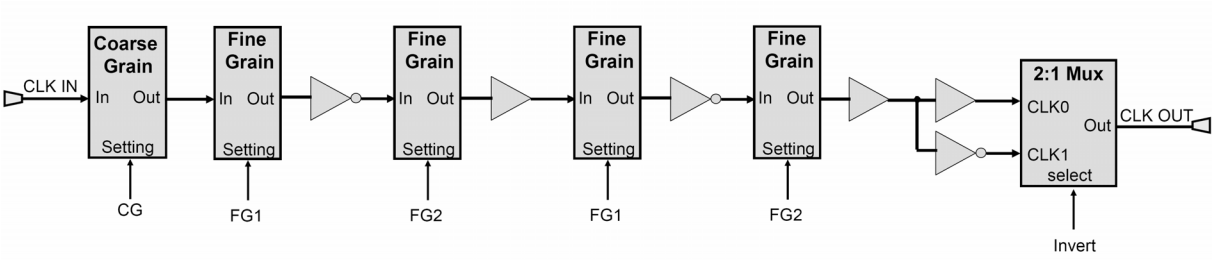

Figure 7.4: 1-2-1-2-fine delay configuration.

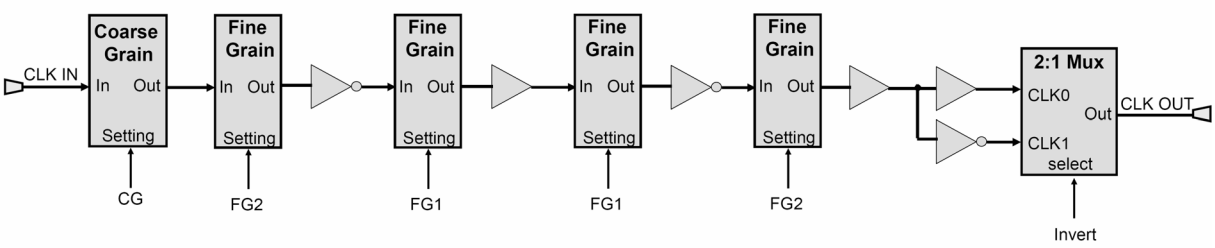

Figure 7.5: 1-2-2-1-fine delay configuration.

and delay lines, ensuring that each transition will propagate through both pull-up and pull-down portions of the delay line. For a pair of identically set delay lines, if the pull-up section delays or advances the signal by d_1, and the pull-down section delays or advances the signal by d_2, then the rising edge of the input signal will be skewed by d_2+d_1 and the falling edge will be skewed by d_1+d_2, resulting in a net zero change in the duty cycle of the signal. While the d_2 and d_1 delays are ideally identical, this relation cannot be guaranteed due to process variance. The previous configurations produced a d_2+d_2 shift in the rising edge and a d_1+d_1 shift in the falling edge, creating a $|2*(d_2-d_1)|$ change in the duty cycle of the output signal.

7.2.2.5. Grouped delay lines

Since matching of the fine grain delay inverter pairs is important to achieving good results, the delay lines should be kept in close proximity to each other. In the previous delay line, the second group of delay lines was split by the first group. The next iteration is shown in Figure 7.6 and the pairs of delay lines are physically located next to

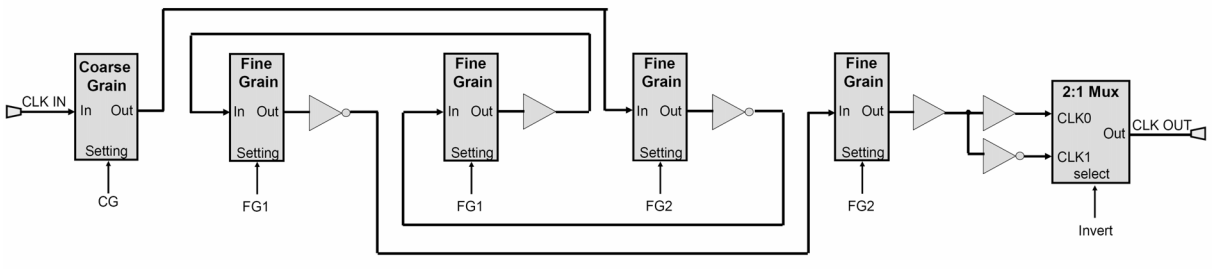

Figure 7.6: Grouped fine delay configuration.

one another, but traversed in the order of the previous configuration. As a result, the delay characteristic between this delay line and the previous one is nearly identical in simulation. This method will be more resilient to process variation due to the smaller distance between the components.

By standardizing the signal characteristics, the number of realizable delay settings decreases since permutations of the delay setting order no longer modifies the delay. In addition, the structure of the variable delay inverter requires that one delay setting be reserved. As a result, the potential resolution is now:

$$resolution(ideal) = \frac{Coarse\_delay}{\left(2^c - 1\right)^g - \sum_{i=1}^{g}\binom{g}{i}} \qquad (7.3)$$

To obtain a greater number of unique settings the second pair of delay lines was simply resized, resulting in much better resolution:

$$resolution(ideal) = \frac{Coarse\_delay}{\left(2^c - 1\right)^g} \qquad (7.4)$$

7.2.3 Performance

These delay lines provide high resolution delay increments using a current starved approach without requiring static power consumption as is usually the case when digitally controlled switches [119],[120] and have full rail-to-rail operation. Our fine

grain delay components use the grouped configuration and are capable of delays that exceed the coarse grain increment by over 20%. This overlap creates a built in robustness and tolerance to process variability as demonstrated in [121]. The layout of the Figure 7.5 delay line is shown in Figure 7.7. It achieves equal high-to-high and low-to-low delays, resulting in matching duty cycles for input and output clocks. The layout area of the design is 2300 μm^2. This is a single delay line instance and is usually paired to allow the delay to be averaged. In a *paired* delay line structure, the total delay through a pair of delay lines sets the minimum clock period that can be used. The alternative is a *shared* delay line structure where given a delay setting δ, a $2*d$ delay is created by propagating a single pulse through the same delay line twice instead of two distinct delay lines. This shared delay line structure will achieve ideal matching characteristics for averaging. Note that the matching of delay lines between taps has no effect on the clock skew since averaging is performed completely within each tap. Variation may require different delay settings to achieve the same delay in different taps, but the *average* clock at each tap will remain synchronized.

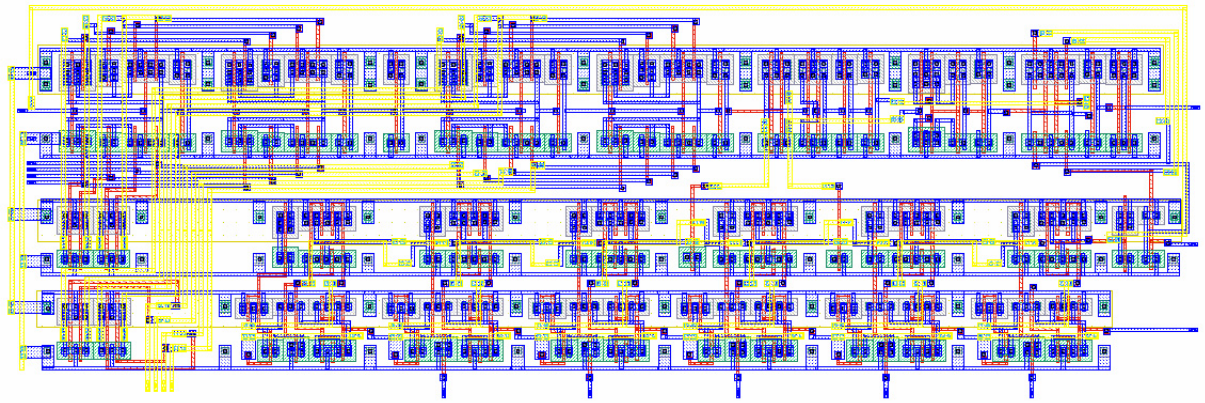

Figure 7.7: Layout of complete delay line.

The total coarse grain delay through a datapath containing two delay lines was 188 ps for the schematic version of the circuit and 160 ps for the extracted one for both paired and shared cases. Assuming a fine grain delay equal to the coarse grain increment, 6 coarse cells are required to handle clocks with a maximum period up to 2 ns. The maximum clock period for the schematic implementation is 2244 ps using these 6 coarse settings and 1920 ps for the extracted circuit. Additional coarse settings can be added as needed to function with arbitrarily long clock periods. The all zero setting is not allowed for the fine grain inverter. The achievable resolution given this configuration is 0.84 ps. Through the paired delay line, the schematic circuit simulations show an actual resolution of 4.74 ps (upto 160 ps) and the extracted simulations show an actual resolution of 5.85 ps. The extracted version of the shared delay line achieves an actual resolution of 5.72 ps between delay settings (full delay prior to averaging). The average delay increment is approximately 1 ps. The performance of both the paired and shared extracted delay lines is compared in Figure 7.8. For the clock distribution network, the largest delay increment (resolution) of the delay line is twice the achievable skew bound due to the averaging involved.

An additional, simplified delay line is used in the built-in clock skew system. It uses four variable inverters and two coarse grain settings to lengthen the maximum total delay of 192.8 ps (δ). Two variable delay inverters are paired together to create each variable delay block in Figure 7.9. The maximum increment between delay line settings, 2.93 ps, will fix the resolution of the skew measurement and the skew correction that can

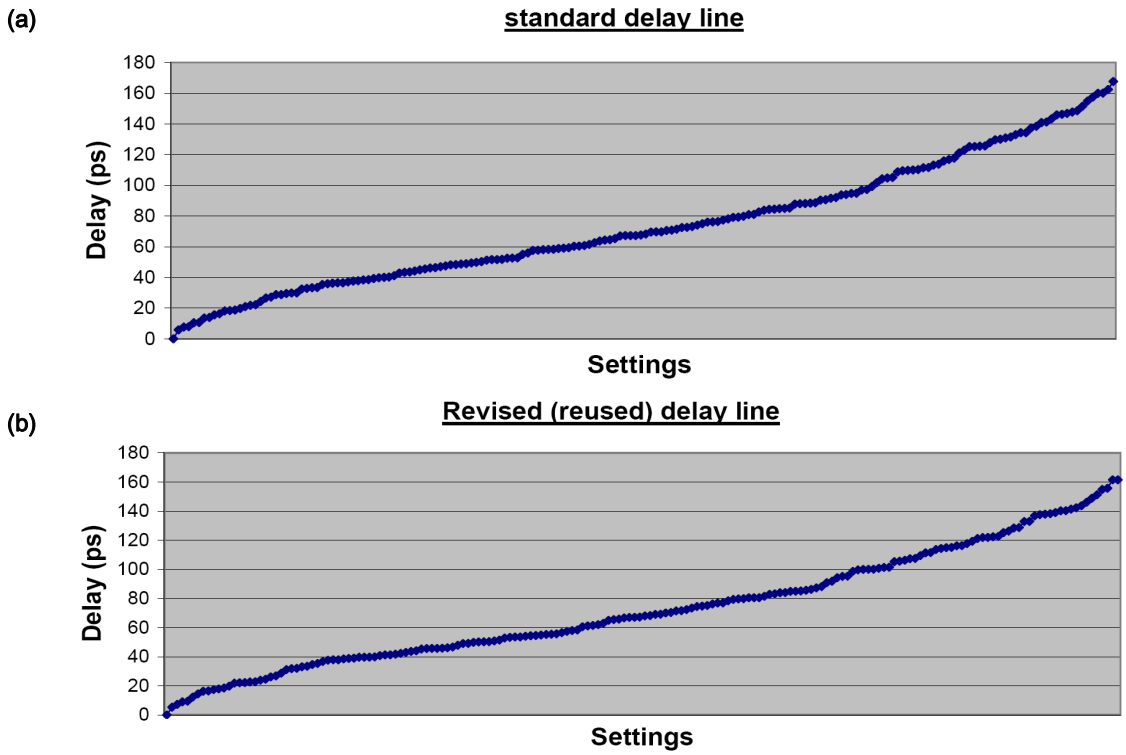

Figure 7.8: Delay range of extracted grouped and shared delay lines.

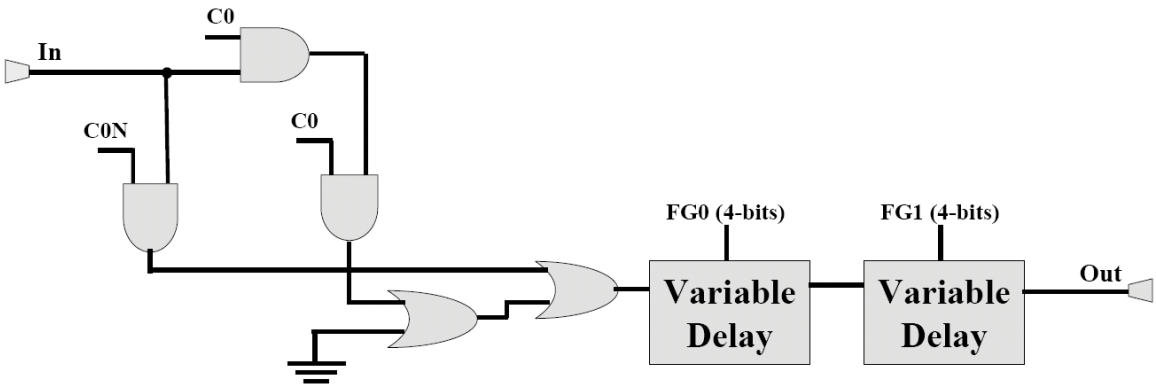

Figure 7.9: Simplified delay line used in BICSS system.

be achieved. The maximum total delay (δ) will bound the maximum round trip delay between the centralized BICSS hardware and the local clocks, as well as the maximum skew that can be tolerated between clock regions for the system to work. As such, the delay line is scalable and can be lengthened with additional coarse grain settings to suit different applications.

7.3. Clock switches

The design of the clock switches will have a significant effect on the behaviour of both the single clock and reconfigurable clock single conductor networks. These switches must be bi-directional so signals can propagate between ports in both forward and reverse directions using a single wire to connect clock network segments. This helps diminish the effect or wire mismatch and process variance. This tap switch operates in three modes, *forward*, *synchronize* and *reverse*. In *forward* mode, the forward reference clock is connected to the current tap and sent to the next tap. In *synchronize* mode, both the forward and reverse reference inputs are connected to the tap. In *reverse* mode, the reverse reference input signal is connected to the forward port. This allows a single bi-directional clock line to be threaded as needed throughout the clock domain and shared between the forward and reverse clocks during synchronization. The switch is shown in Figure 7.10, and its layout is shown in Figure 7.11. The area of the switch is 172 μm^2.

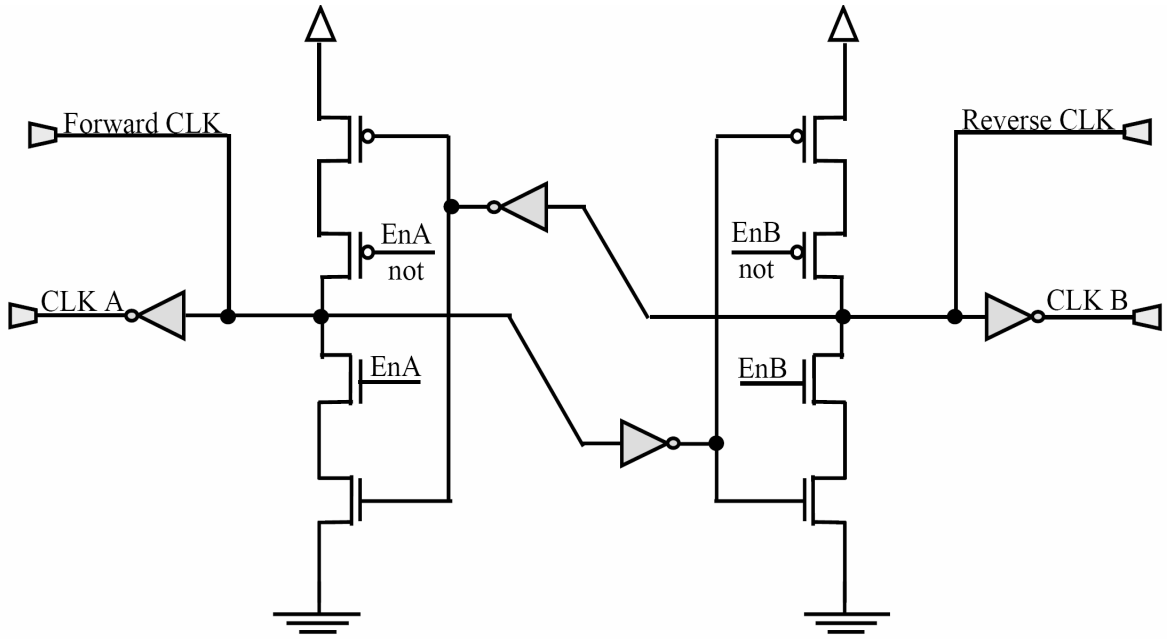

Figure 7.10: Tap bypass switch.

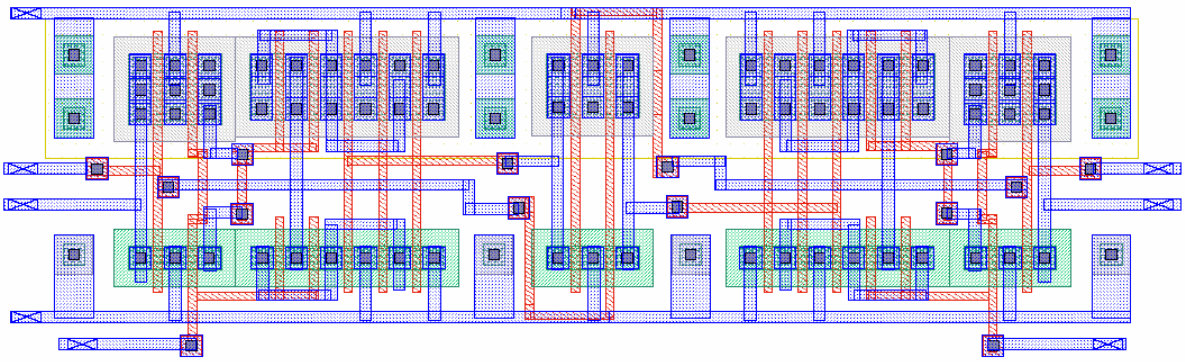

Figure 7.11: Layout of tap bypass switch.

To reconfigure clock domains, other routing switches are required. These have a more stringent design requirement as each port must be able to route an input signal to any other port, while matching delays in both directions between associated ports. The 2 clock, 4-port routing switch is shown in schematic in Figure 7.12 and in layout in Figure 7.13. It has an area of 1300 μm^2 and uses pass transistors to control access to an

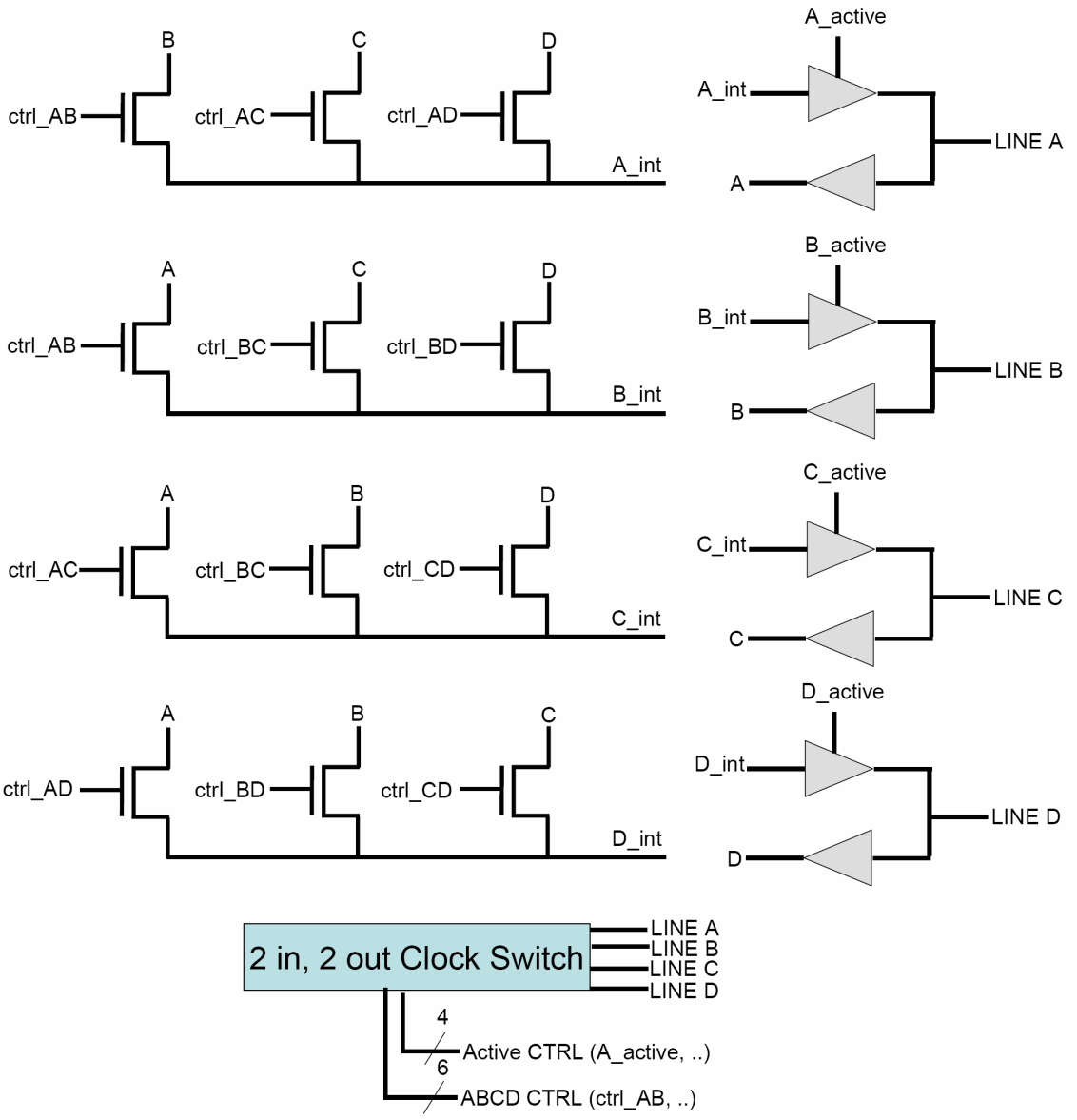

Figure 7.12: 4-port clock routing switch.

intermediate signal, which gains access to the clock port through a Z-buffer. The Z-buffers establish the direction of the clock signals through the crossbar and are controlled by four *active* control bits of which only two can simultaneously be asserted. This design is scalable to create larger routing constructs. For instance, a 4-clock switch containing 8 ports has been designed in a similar fashion.

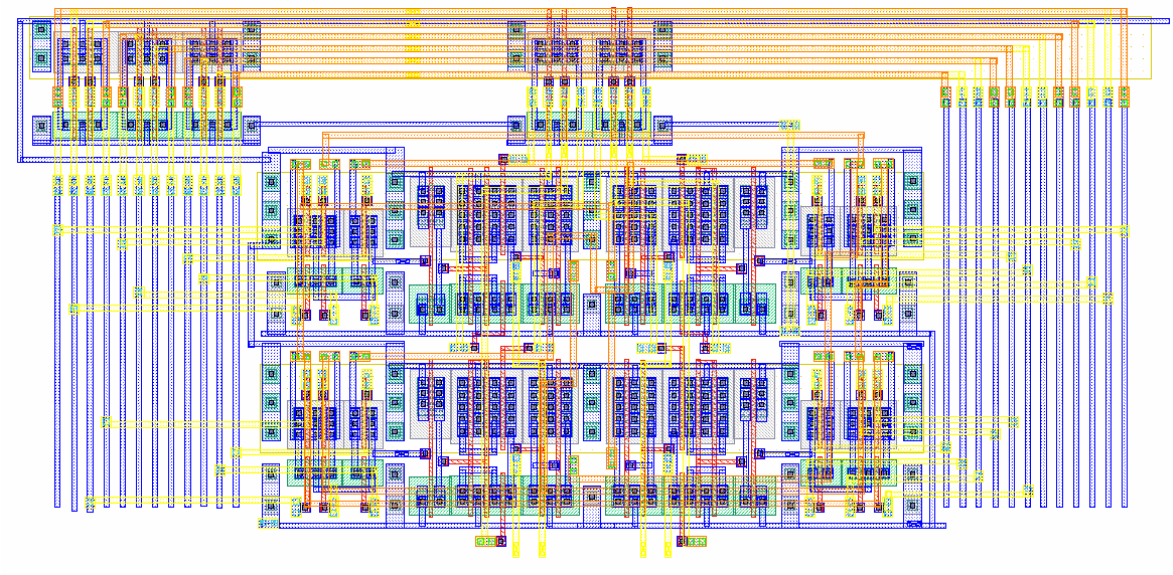

Figure 7.13: Layout of 4-port clock routing switch.

To be used in a dual reference line, hot spot and variation tolerant system, this routing switch can be modified and replaced with circuitry that has the functional of a set of multiplexers. This circuitry should be laid out to have equal propagation delays from any input port to output port and be as close to centroid as possible, Figure 7.14. The controller line memory is omitted for the unidirectional switch for clarity.

7.4. Phase detectors

The phase detectors that we designed are non-traditional approaches to the problem since the goal is not to create solution that eliminates metastable conditions. Instead, since the target application is one that will have a finite resolution due to the use of a digital delay line, the system performs two simultaneous comparisons on slightly

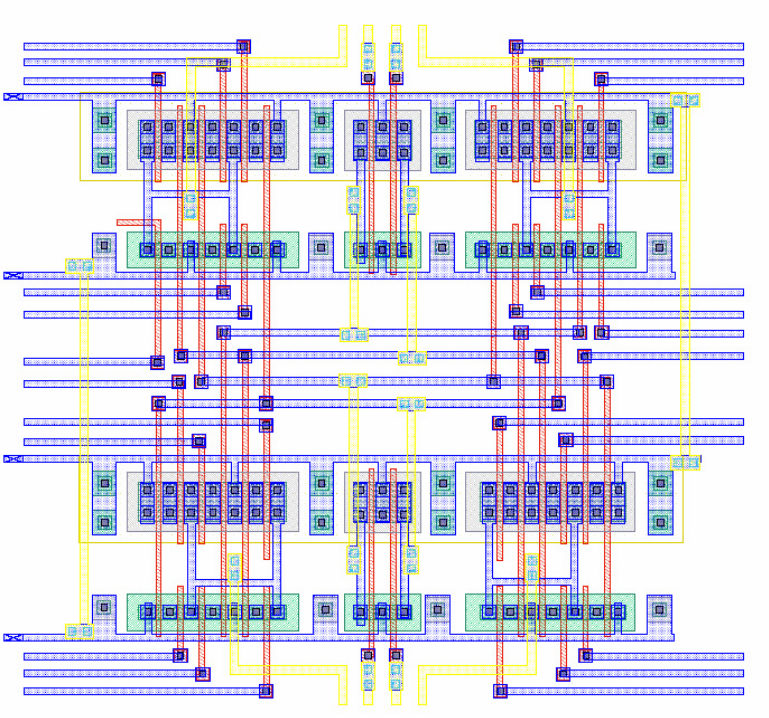

Figure 7.14: Layout of unidirectional 4-port clock routing switch.

skewed versions of the inputs to guarantee that one of the two comparisons will resolve and produce a useful result.

7.4.1 Fixed tolerance phase detector

There are many phase detectors presented in literature [122],[123]. Our design is an original one, designed to solve the unique challenges of our system. Our phase detector is a sample-and-hold type, sampling the state of the system around the clock edge and retaining the result for roughly a third of one clock cycle, independent of the overlap between input clocks. The system features a nonlinear three state phase detection system [124]. Along with an *UP* or *DOWN* detection, it can also signal a

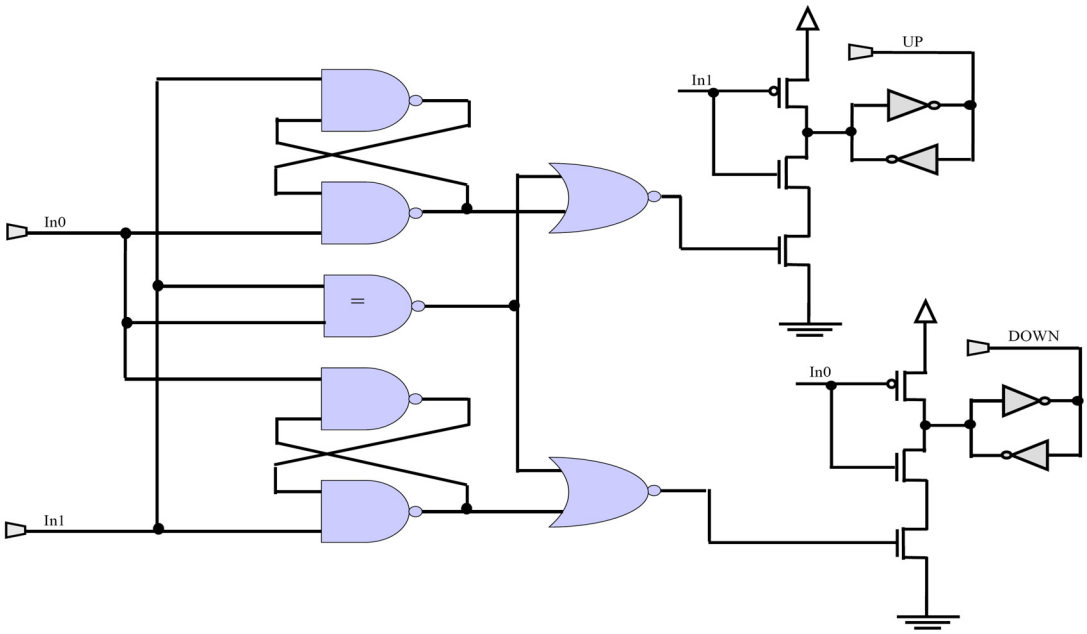

Figure 7.15: Fixed tolerance phase detector.

LOCKED condition for a defined skew bound between clock inputs. Like most detectors, the design in Figure 7.15 uses a cross-coupled NAND gate latch structure to perform the signal comparison. Two interconnected latches are used in a method similar to [125]. In the first latch, when *CLKA* arrives before *CLKB,* the *UP* signal is asserted. Here *CLKB* is the controlling input since it prevents the *CLKA* from causing an assertion. Conversely, when *CLKB* arrives before *CLKA,* the *DOWN* signal is asserted by the second latch and *CLKA* is the controlling input. However, in our case, we modify the latch to be more sensitive to the assertion of the controlling input. In this way if the inputs arrive very close to one another, neither an *UP* nor a *DOWN* signal is asserted. Where other phase detectors use multiple latches to accelerate metastability resolution, our design moves the problem point away from the center of the locked region by performing two edge detections simultaneously. In doing so, when the input clocks *CLKA* and *CLKB* are at the

edge of the “nearly locked” region, the result can either be resolve to *UP* or *DOWN* (depending on the relative arrival time) or remain *LOCKED*. Since our system is digital and allows bounded skew, both results are acceptable. This resolves potential metastability while allowing for a small dead zone, differentiating our design from others that we have seen in the literature.

Our phase detector only needs to be as precise as a half of the maximum increment between fine delay settings, emphasizing finite resolution time over absolute precision. The phase detector cannot detect variation with inputs that are shifted in phase by 180^{o}. However, non-overlapped clocks are easy to detect with simple circuitry, so this trait is not a significant drawback with our system and the result, *UP* or *DOWN*, is equivalent for the clock synchronization. The layout of the design requires 500 μm^2 and is shown in Figure 7.16.

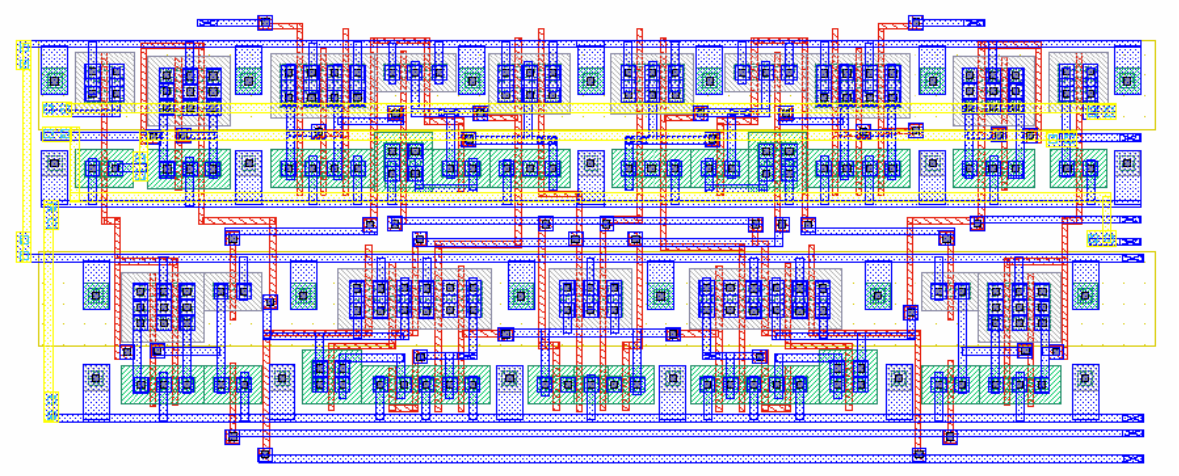

Figure 7.16: Layout of fixed tolerance phase detector.

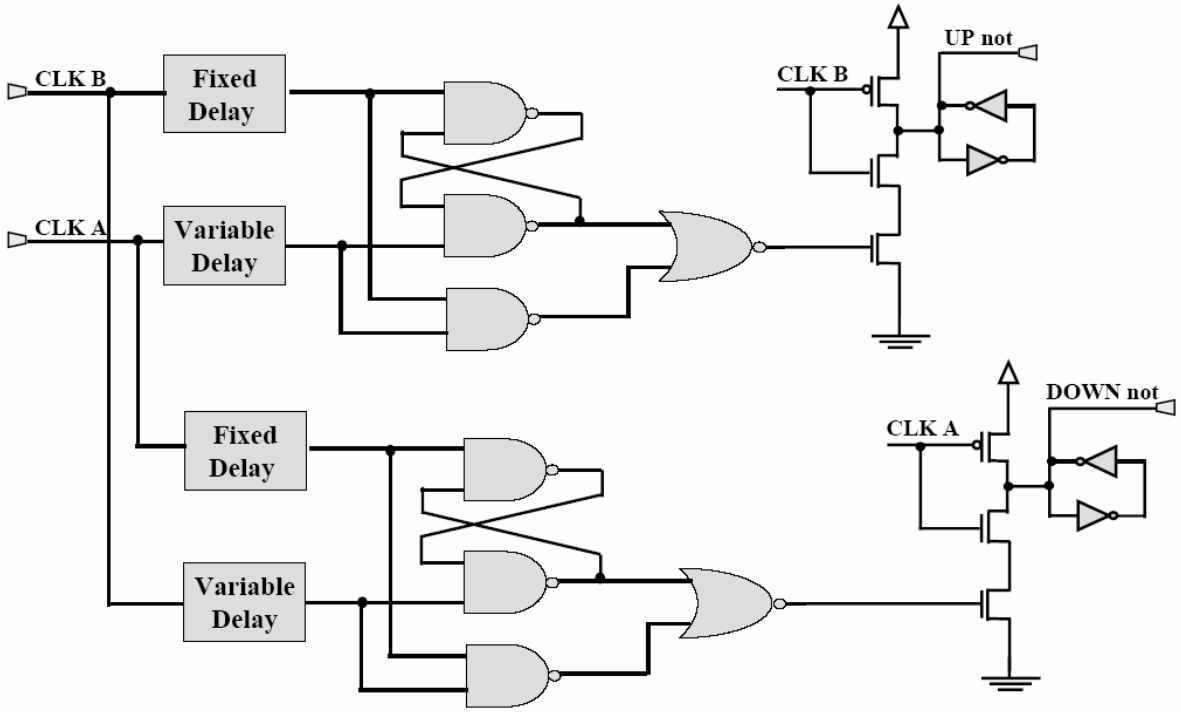

Figure 7.17: Variable-tolerance phase detector.

7.4.2 Variable-tolerance phase detector

While the width of the nearly locked state is fixed electrically for the fixed phase detector in Figure 7.15, we have also developed a phase-detector with variable locked width since the amount of skew that should be allowed in a circuit is system dependent. The variable tolerance phase detector works by taking its two input signals and delaying each one using two parallel delay lines. The first delay line is variable through a digitally controlled delay line (DCDL) and the second one is fixed to just over the static minimum latency of the DCDL, Figure 7.17. This component was developed to be used with the BICSS system, but the system will not work with too small a fixed delay and too large a fixed delay will limit the round trip time that can be compensated for. Two cross-coupled

NAND gates are used for each latch, similar to the fixed tolerance phase detector, with each latch here designed with no input preference. In this case, the “sufficiently locked” state is found when both *UP* and *DOWN* signals are simultaneously asserted. Using the control lines of the delay line, this sufficiently locked state can be adjusted from a window of nearly zero to twice the maximum variable delay line setting. The variable phase detector can be set to have a 3, 6, 8, 10, 28, 29, 35, 37, 51, 52, 55, 59, 67, 106 or 253 ps locked region. This trait is useful since most systems can always tolerate some skew and it may not always be desirable to detect minute levels of skew. This phase detector is used to detect programmable amounts of skew, specific to the needs of a particular system. This is a functionality that is unavailable for other nonlinear phase detectors and makes this design particularly suitable for skew detection applications. This functionality also allows for some post-silicon tunability to correct for process and temperature variations. The layout of the variable tolerance phase detector requires 1300 μm^2 and is shown in Figure 7.18.

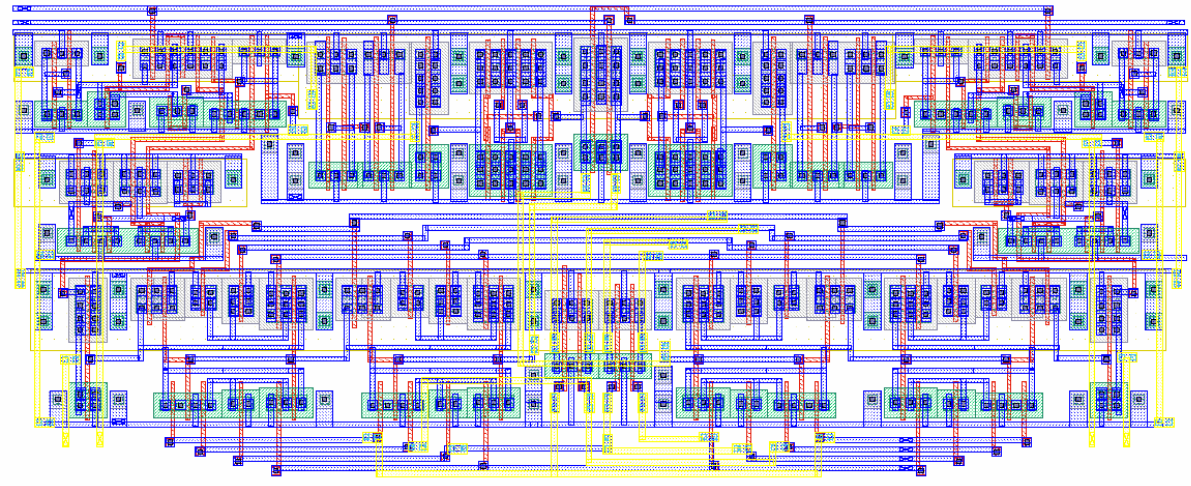

Figure 7.18: Layout of variable tolerance phase detector.

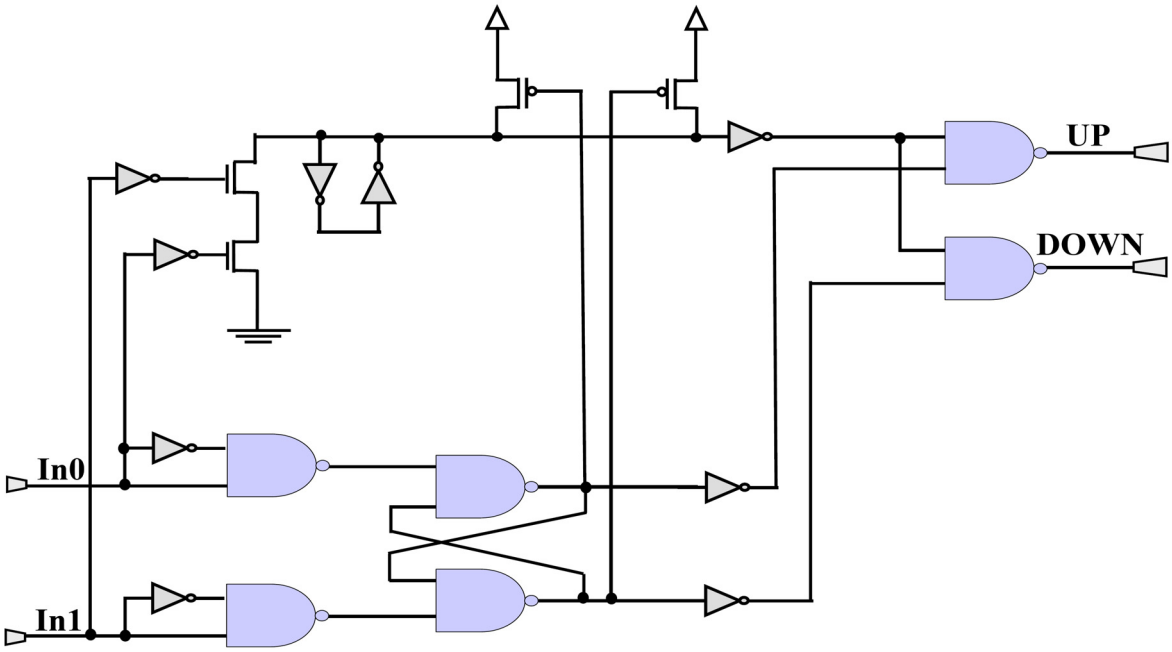

Figure 7.19: Modified phase detector for shared delay line systems.

7.4.3 Modified phase detector for shared delay line implementations

The challenge tin designing a phase detector for the shared delay line clock network is dealing with two signals with different duty cycles. While the reverse clock possesses a 50% duty cycle, the forward clock's duty cycle is diminished to a constant width shorter than the loop delay. This creates an interesting scenario where there is guaranteed to be a time where both signals are low that will be held long enough to be read. This is not the case with the previous phase detectors which require special care to deal with reference clocks that are 180 degrees out of phase. The modified phase detector uses the both input zero state to arm itself and is shown in Figure 7.19. If more there is more than once instance where both inputs are zero per clock period,

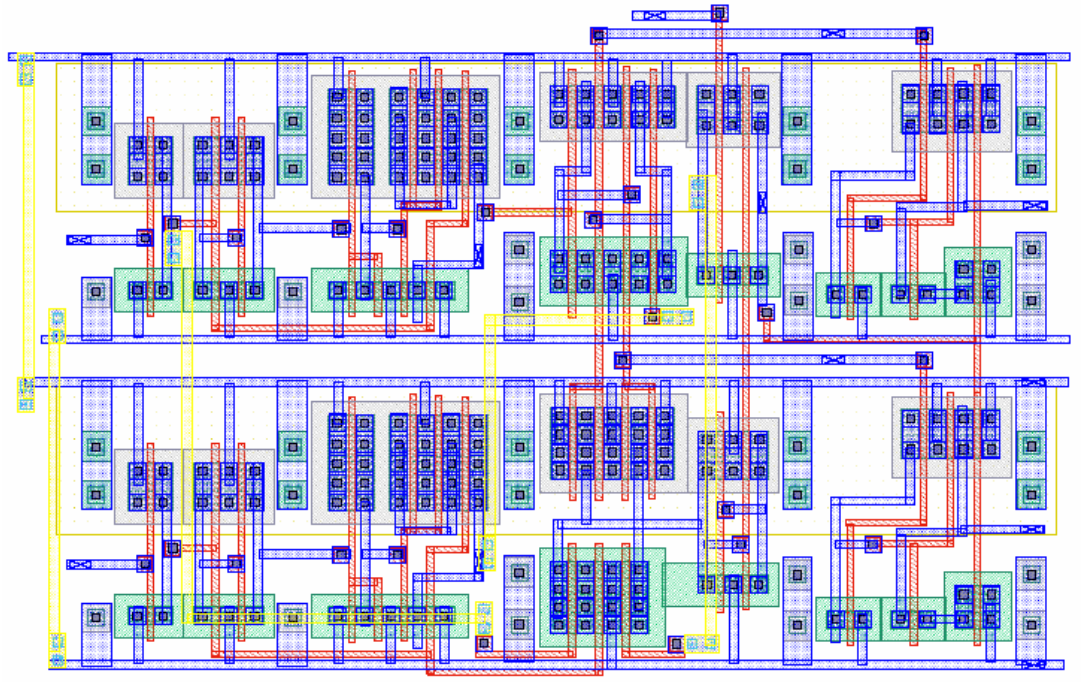

Figure 7.20: Layout modified phase detector for shared delay line system.

preference is given to the one immediately following the high-to-low transition of the reverse reference clock. The layout of the design, Figure 7.20, has an area of 540 μm^2.

Chapter 8:

Conclusions and future work

8.1. Summary

The single and multiple clock reconfigurable clock networks were designed to suit a variety of applications, clock domain shapes and sizes using a standard cell approach. The clock networks are designed to be implementation-independent – simplifying the design of clock distribution networks. An averaging clock distribution can contain an arbitrary number of nodes and can be laid out manually or using standard cells, whereas H- or other tree solutions require special balancing tools to generate

synchronized clocks. Our serial clock network can have a beneficial effect on the power consumption of a CDN by decreasing the capacitive load that must be switched in the clock distribution network by decreasing the total wire length – typically requiring only two-thirds of the wire length of comparable clock trees. The circuits and approach to implementing these clock networks are outlined here-in.

Due to the large number of potential network configurations using averaging that are possible, it is necessary to demonstrate the design and operation of the system for a given configuration. The clock networks were not only simulated in schematic and extracted layout form using TSMC's 180 nm standard process, but their functionality was also modelled with the system level controllers in hardware description language (VHDL). The models were paired with synchronization controllers to demonstrate the system-level operation of clock networks using our technique. These models and controllers are designed using a generic approach, but require specific configurations to demonstrate their operation. To maintain consistency between simulations, two specific configurations were examined using our techniques: a 4-tap single clock configuration and a 15-tap three clock domain reconfigurable configuration.

The synchronization controllers can be realized using an internal controller in silicon, or an external one using a microprocessor, an FPGA, or any other hardware or software approach. The results show that the controller/circuitry interface is simple yet versatile enough to operate the circuitry easily and efficiently. The majority of the required communication bits are dedicated to the on-chip configuration memory. The

resources required to implement the controllers and the time required to perform the synchronization and calibration operations are also reasonable and show that the designs can be practically implemented in a real world system.

8.1.1. Single clock averaging network

Today, most clock networks are designed using CAD tools which require precise information on the exact clock load for each branch, the placement of each tap on the die and the location of the clock root. Once generated, the clock network cannot be altered without affecting clock skew. Our cell-based approach to clock distribution allows components to be designed independently, connecting components as is convenient and even replacing blocks if needed. Using a dual reference signal averaging technique in the clock network allows designers to delay some of the critical clock tuning requirements to facilitate the design flow. It allows circuit blocks to be moved around conveniently and re-sized easily with a simple change in the number or location of the taps.

The system provides multi-point active skew compensation during the system's synchronization phase to compensate for device mismatch and process variance – enabling our design to have all the benefits of other active clock skew reduction methods. However, since this synchronization circuitry can be disabled at run-time, the system can operate in an open loop to save power – a typical benefit of traditional clock trees. By using delay lines instead of PLLs and with a combined reference signal and

clock distribution conductor, our method is small enough to be useful for many clock applications.

A dual reference line averaging approach is explored to maximize the system's tolerance to device mismatch. By permitting a much smaller distance between ideally-matched devices, our system overcomes the significant effect of device mismatch on traditional clock trees. Tree-based clock distribution networks are also susceptible to skew from cross-chip temperature variation due to the distributed buffers that they employ. By using selective re-synchronization approaches, it is possible to also overcome changes to intra-die thermal gradients and hot-spots using our approach. Overcoming delay mismatch due to process and environment conditions is a major concern in modern clock distribution networks.

8.1.2. Reconfigurable multiple clock averaging network

Using programmable repeater stages allow us to redirect clocks post-silicon at certain pre-defined switchpoints, making the network reconfigurable. Clock networks can be modified to correct for some manufacturing defects, including bypassing certain clock lines and clock buffers. The operating clock frequency can be changed to fit an IC's many possible target applications. Simulations show that the proposed CDN is scalable, compatible with irregularly-shaped distribution areas, and simplifies the way a design can be floorplanned onto an integrated circuit. The clock networks are applicable to both static and programmable designs and combine low power operation with tight skew

bounds. Our tests show that the reference-based programmable clock distribution is resilient enough to be used in an ASIC, SoC or FPGA environment. The possibility of using of a single conductor to provide both forward and reverse reference signals is unique.

8.1.3. Built-in clock skew system

In addition to the clock networks, we have investigated BICSS: an on-chip clock skew management system to detect, infer and potentially correct clock skew between selected points on an IC to repair otherwise defective dies using high-resolution delay lines. Using the skew measures allows the quality of the clock distribution on the fabricated die to be assessed. BICSS also aids in the debugging of timing errors that may be discovered during testing due to the added visibility of on-chip clock signals. Our BICSS system is unique in its ability to detect, measure and compensate for clock skew using a single all-in-one solution. The development of a variable tolerance phase detector makes this the first system to allow online detection of a programmable skew bound. BICSS enables designers to modify their design flow to include post-fabrication adjustment to the clock distribution network to correct for timing faults or to minimize clock skew for higher frequency operation.

We use an averaging technique to compensate for different propagation delays between clock tap pairs which allows a single BICSS unit to be used for multiple test points providing an efficient system through component reuse. Our entirely digital

solution requires little additional circuitry and is a low cost alternative to the other, traditionally costly skew measurement techniques. The circuitry and synchronization controller for BICSS was also modelled in VHDL to demonstrate the behaviour and the operation of the system.

8.1.4. Circuit implementations

While the circuits discussed here-in have been designed and laid out using specific criteria in a 180 nm standard CMOS process, there are a number of other potentially suitable component implementations which could work in creating our clock network and clock skew detection circuitry. While care has been taken to create easy to design digital structures that would facilitate the construction of circuitry using *averaging*, there were a number of novel circuit elements created with respect to previously published work. Our digitally-controlled digital delay line had good resolution with good signal characteristics including full swing outputs is one such design. Traditional digital delay lines do not need to pay as much attention to duty cycle, for example. When using delay lines to perform averaging, the use of a single delay line is a unique and ideal solution to the problem. The bi-directional components allow single wire routing throughout a design with built in skew compensation facilitating the implementation of critical clock routing circuits. Finally, the finite resolution aware fixed and variable tolerance phase detectors represent a novel approach to phase detection adding new

functionality and a response that is compatible with digital circuitry, unlike many of their counterparts.

8.2. Future work

8.2.1. On-chip clock networks

The major benefit of using averaging techniques in the creation of clock distribution networks is two-fold: to eliminate skew generated by process variance and thermal gradients and to make the clock network easier to implement and correct. While these properties can be combined in many cases, they are often non-overlapping qualities since easy to implement implies having a robust design whose functionality is paramount and eliminating variance and clock skew usually implies trying to achieve the highest performance possible. The structures we have examined here were designed as a balance between these options, but the next step in exploring our averaging based clocking technique is to verify the performance and functionality of the system for each of these individual applications. To design a high-performance system, the circuitry should be ported to a newer technology and the design optimized for higher clock frequencies using larger devices and more robust circuitry. The addition of a dual edge synchronizer here would help considerably. Synchronizing the forward reference signal's rising and falling edges independently will add the functionality of tuneable buffers for duty cycle and delay compensation in the circuitry and help ensure greater tolerance to

process variability, since pull-up and pull-down portions of the delay lines will have matched behaviour. For medium performance applications, our system should be constructed using standard cell components to verify its robustness and functionality. The design should be built with programmable delay lines using a memory to load expected values at the center of the systems operating corners to determine how much the clocks can skew during operation. This will help determine what skew margins must be incorporated into the system. Each of the high performance, balanced and medium performance systems should also be implemented in silicon and compared to ensure that the components used in their design are appropriate for the application. A variety of other components can also be tried, allowing designers to select between different families of standard cells: one allowing performance, one emphasizing ease of implementation and another being a balanced of the two.

8.2.2. Automate system implementation

While we have discussed how to design and implement our systems on an IC, the process can be automated into the design flow. The system should be designed to make the use of an averaging clock network as easy as possible. Circuits can be synthesized with the taps in place based on a preferred clock load per tap. Each clocking region in the design should be associated with a list of clock domains that could be attach to it. Considering the required reconfigurability in the system, a clock routing mesh can be built to satisfy the number of taps, the number of local clocking regions, the

number of clock domains and the required flexibility of the network. Once the silicon design is complete, the delay and configuration layout, the switch mesh and tap configuration and the device and interconnect data can be extracted out and incorporated into a clock synchronizer and HDL model created using a dynamic code generator. If needed, the clock synchronization controller can be synthesized into the given technology and included on the IC. In addition, the model can be used to determine the appropriate delay settings required by the system and pre-program them into the delay lines for different configurations. This step can either be used to skip the synchronization step or to speed up the synchronization time when switching between configurations.

8.2.3. Alternative applications

The averaging technique demonstrated in this thesis has a number of useful qualities concerning applications that require correct synchronization. As long as delays are matched for forward and reverse interconnect segments between clock taps, it will be possible to align clocks at multiple points that may be located are arbitrary distances from one another. As long as the averaging technique used is reliable, the clocks can be constructed to be well-synchronized regardless of the technology used at each of the tap points. The use of digital circuitry also makes the system easy to design and implement. This makes the system appropriate for printed circuit board applications which require synchronization between different device clocks or multi-processor clusters designed to

operate using a synchronized global clock [126]. Other distributed applications such as sensor networks could also benefit from an averaging approach. These applications would require modification to the circuitry to deal with potentially different signal voltages and technologies. The diversity of these applications highlights the ease of use and potential of our averaging technique in synchronizing events between different and distributed components.

Appendix A:

System-wide simulations

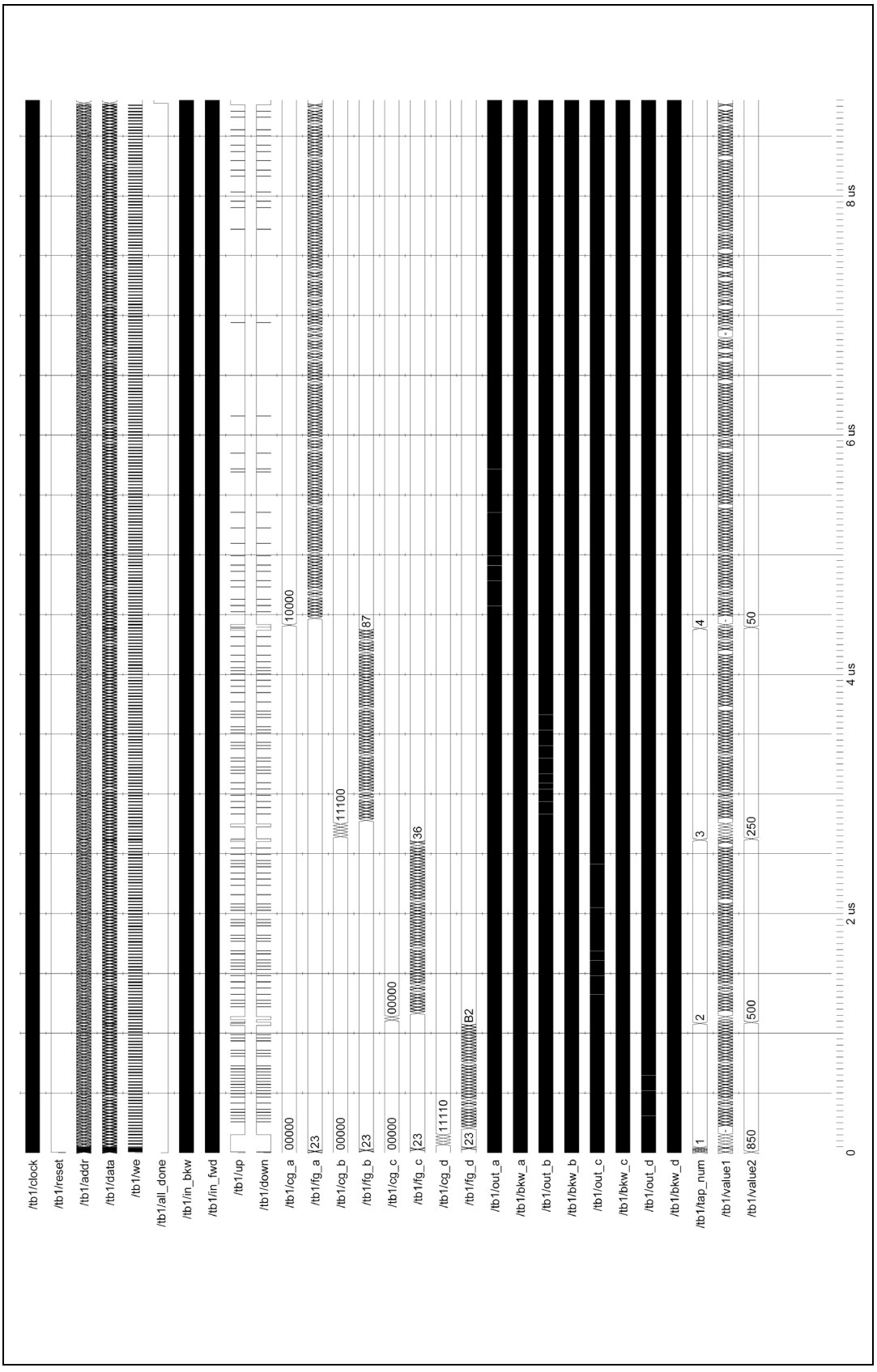

Figure A.1: Complete simulation of a 4-tap distribution at 1 GHz.

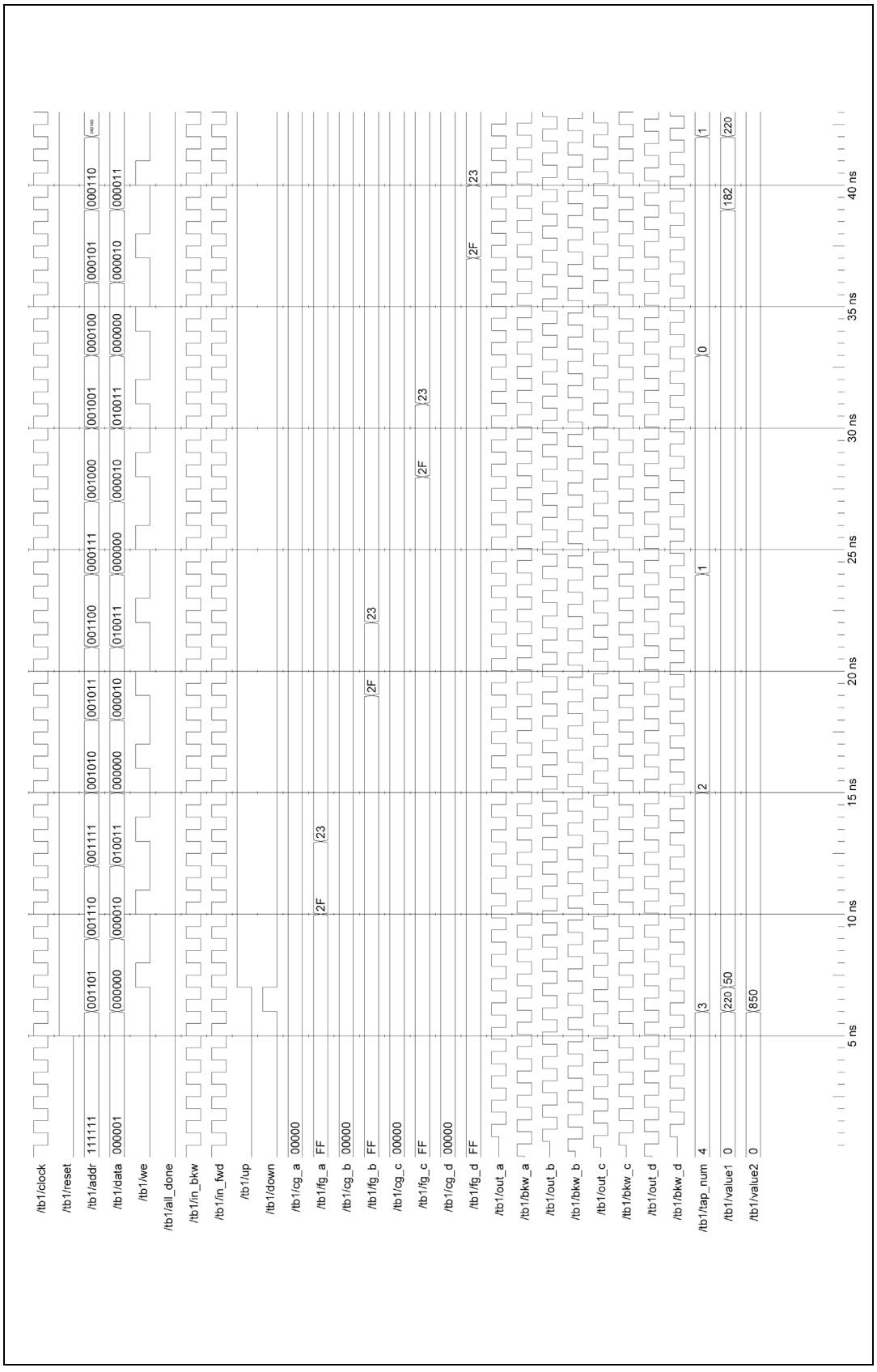

Figure A.2: Result of clock thread initialization of 4-tap clock distribution at 1 GHz.

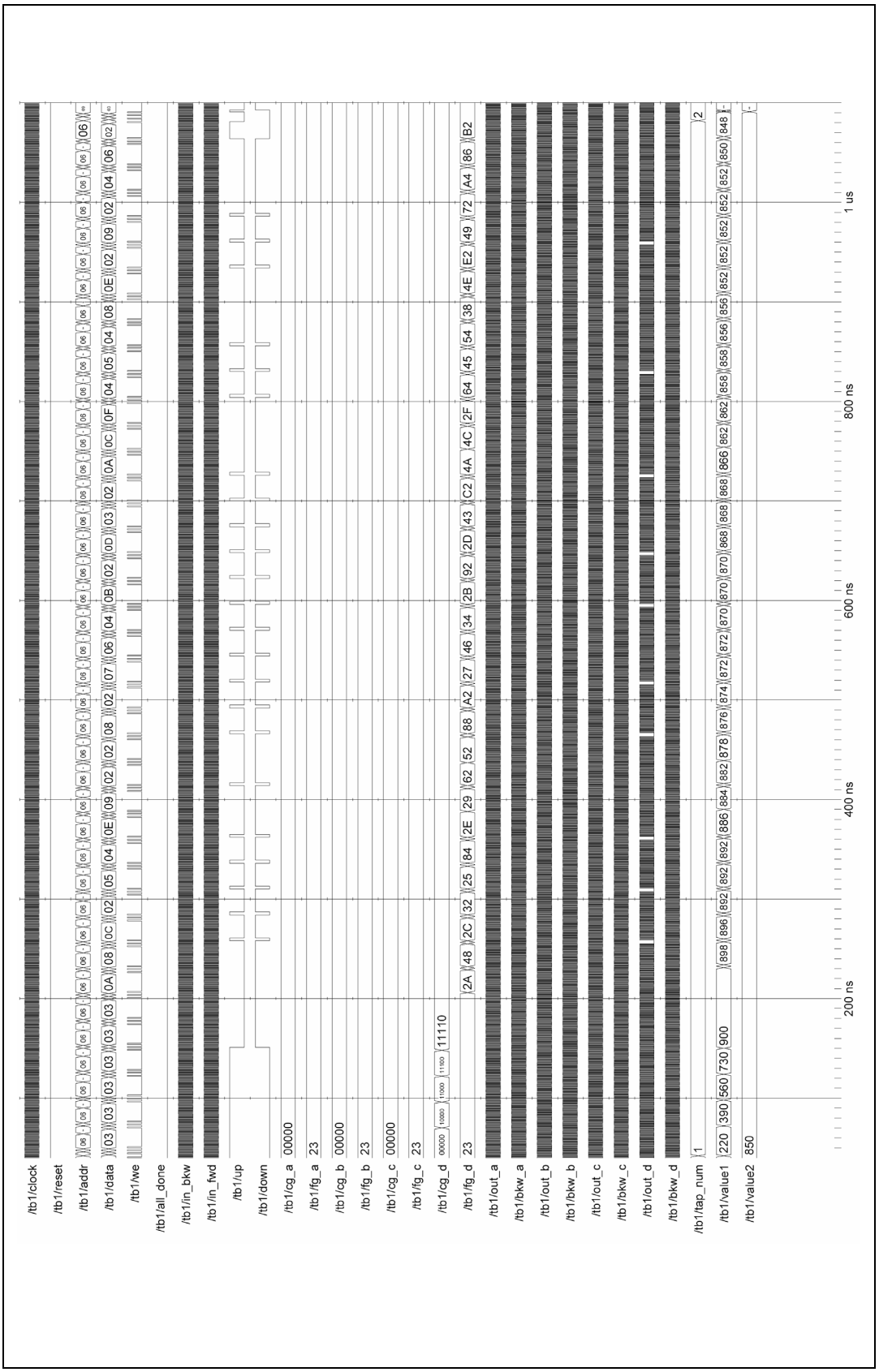

Figure A.3: Simulation result of first tap synchronization of 4-tap clock distribution at 1 GHz.

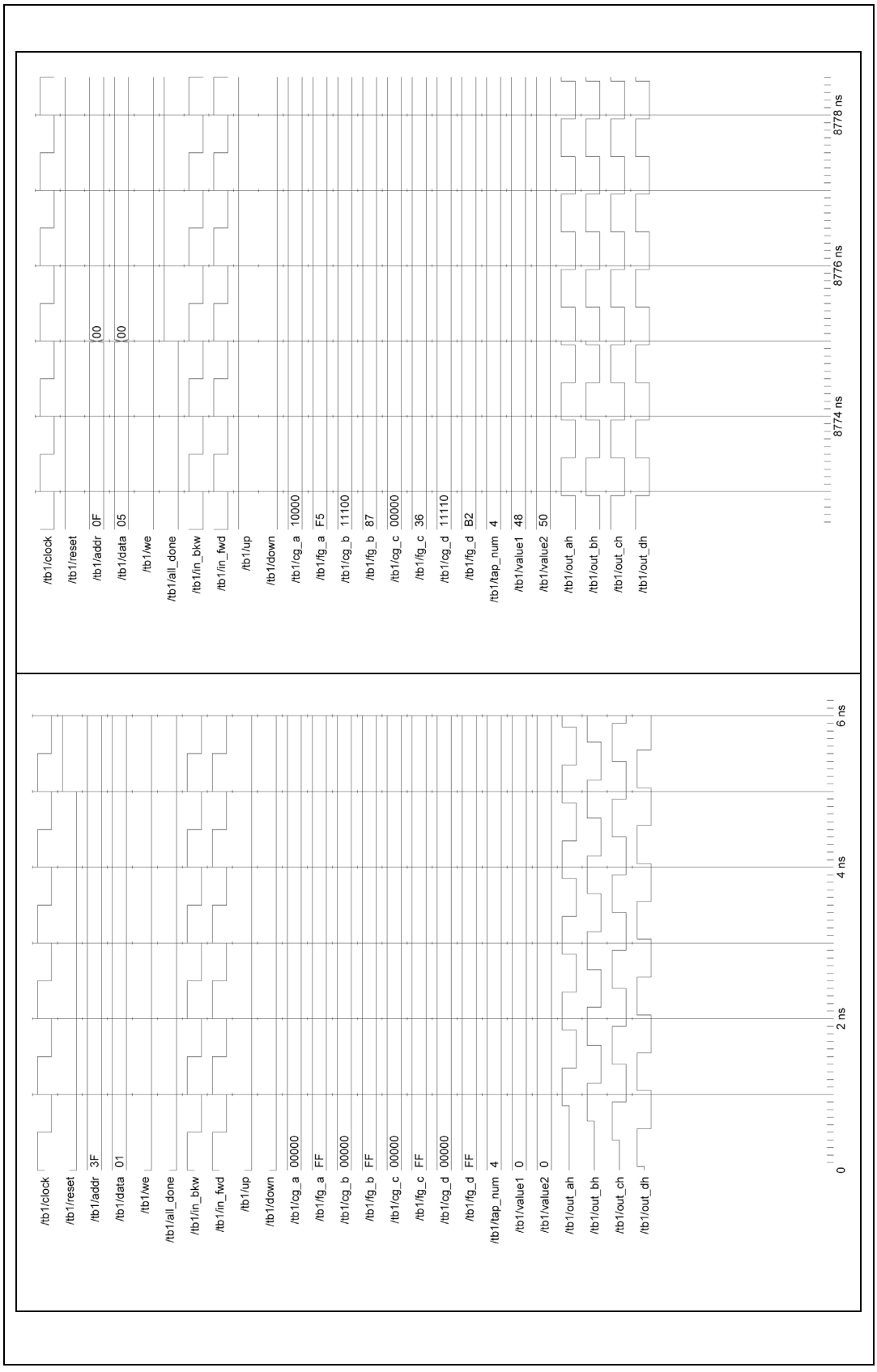

Figure A.4: Comparison of initial and final tap clocks of 4-tap clock distribution at 1 GHz. (left: initial, right: final)

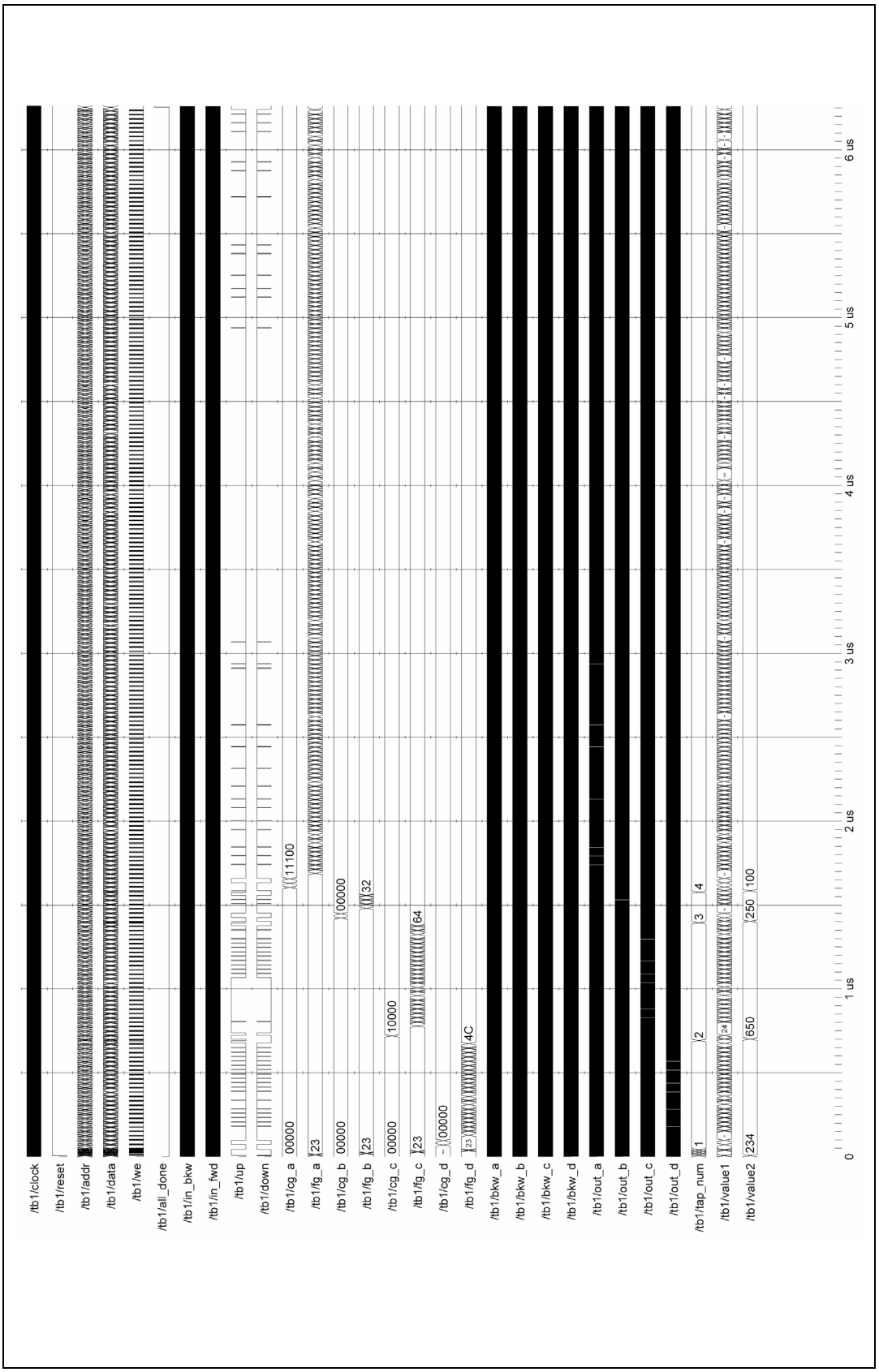

Figure A.5: Complete simulation of a 4-tap distribution at 1.5 GHz.

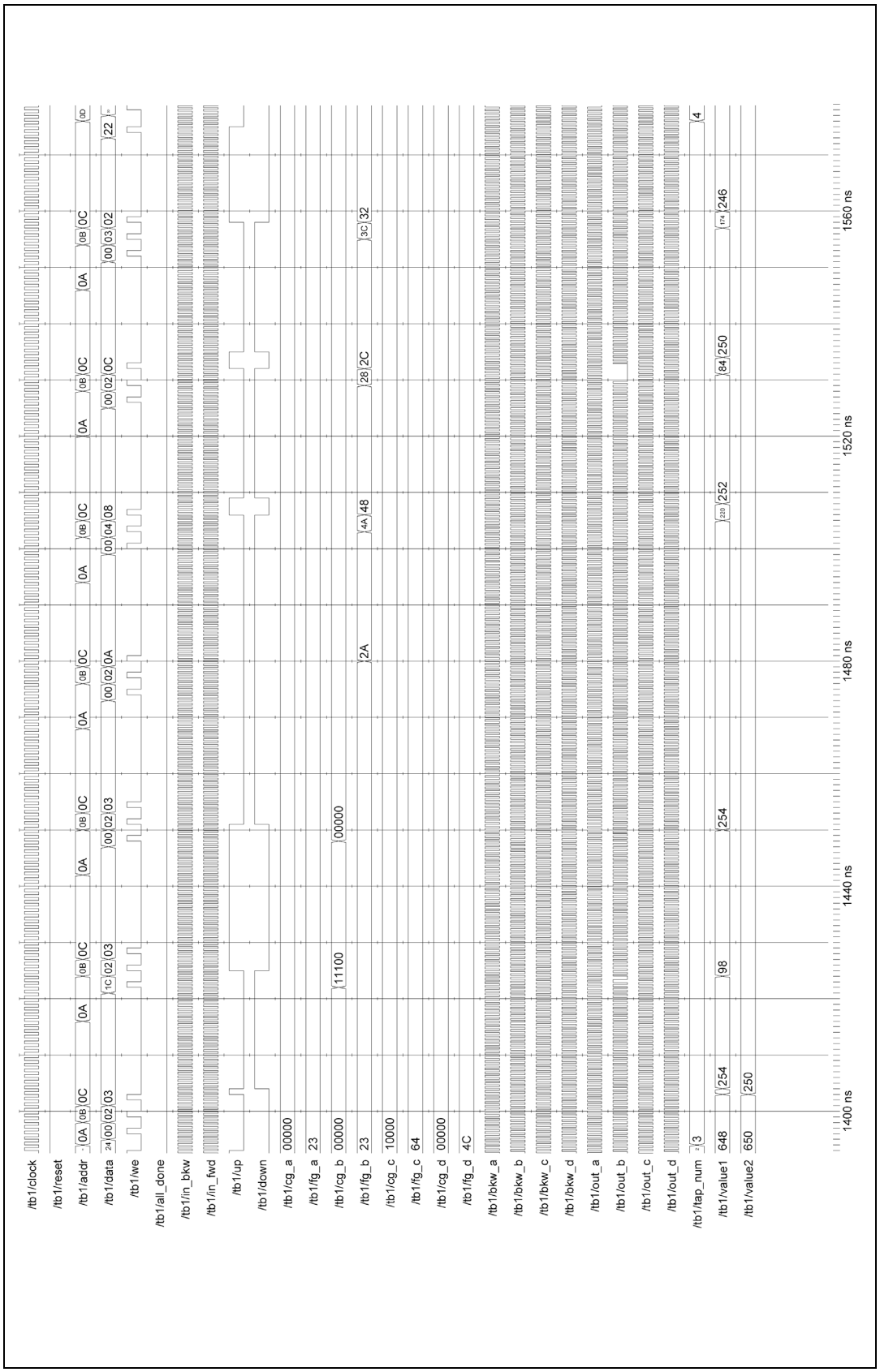

Figure A.6: Simulation result of first tap synchronization of 4-tap clock distribution at 1.5 GHz.

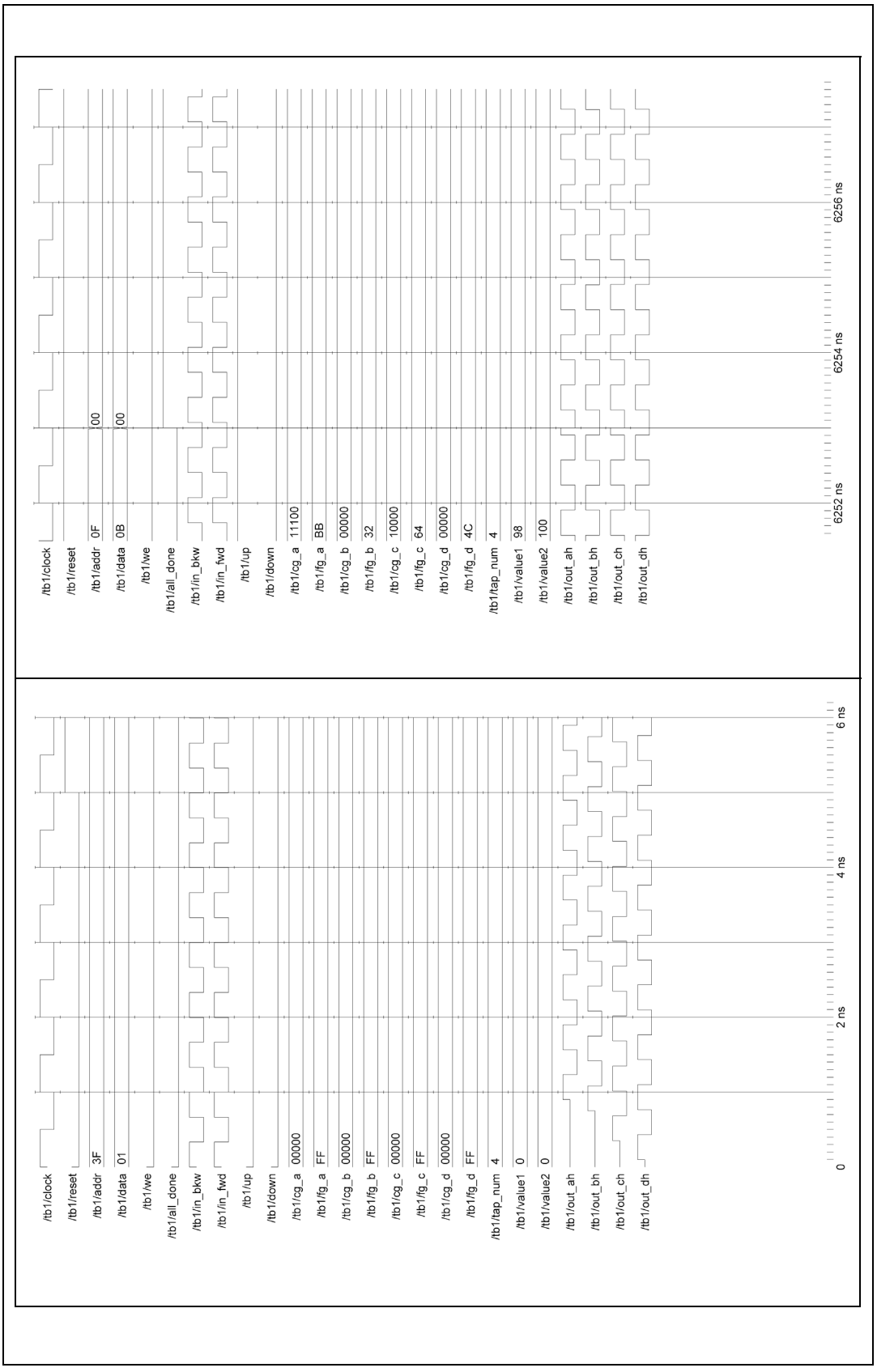

Figure A.7: Comparison of initial and final tap clocks of 4-tap clock distribution at 1.5 GHz. (left: initial, right: final)

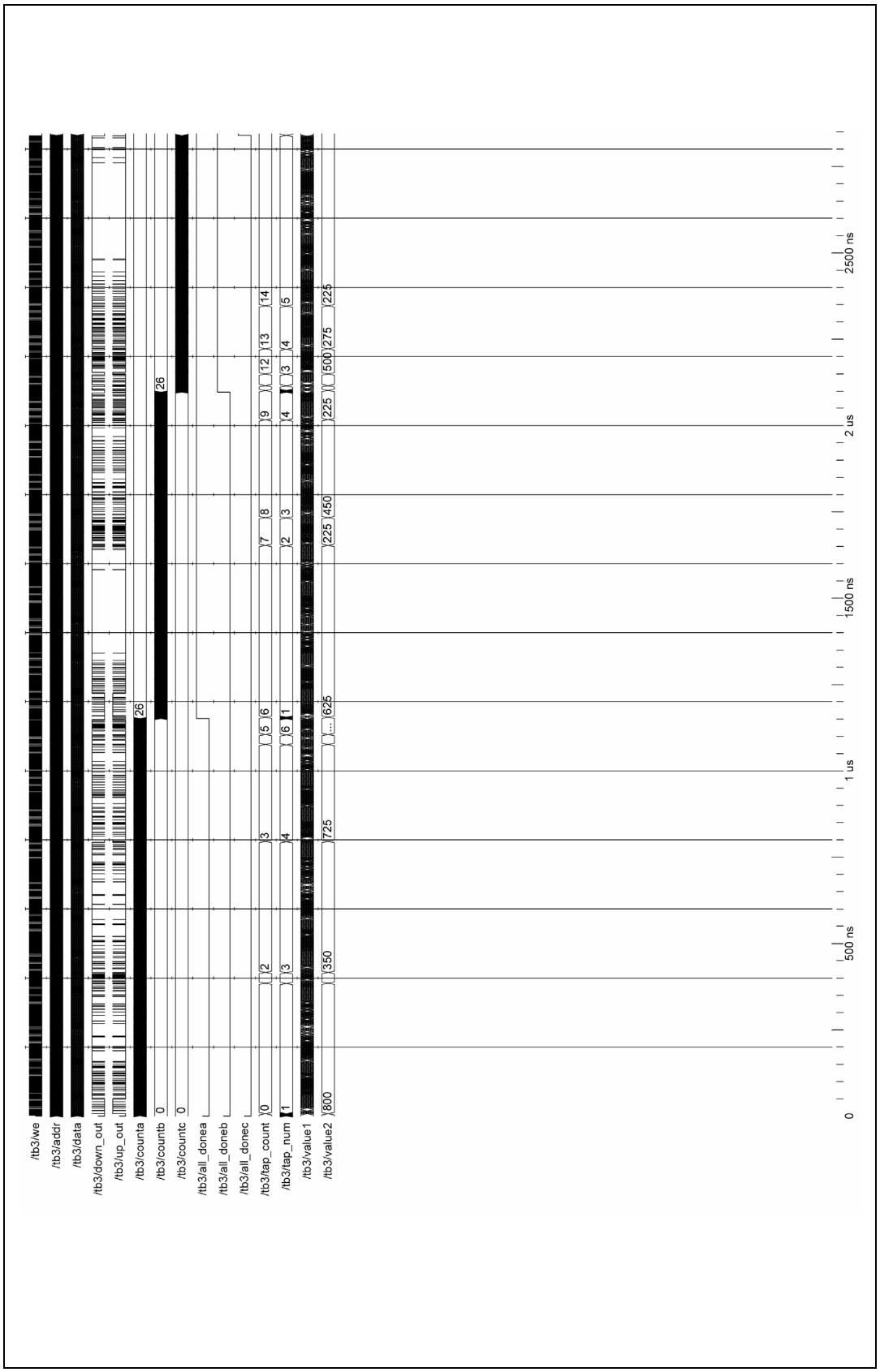

Figure A.8: Complete simulation of a 15-tap, 3 clock domain reconfigurable clock distribution.

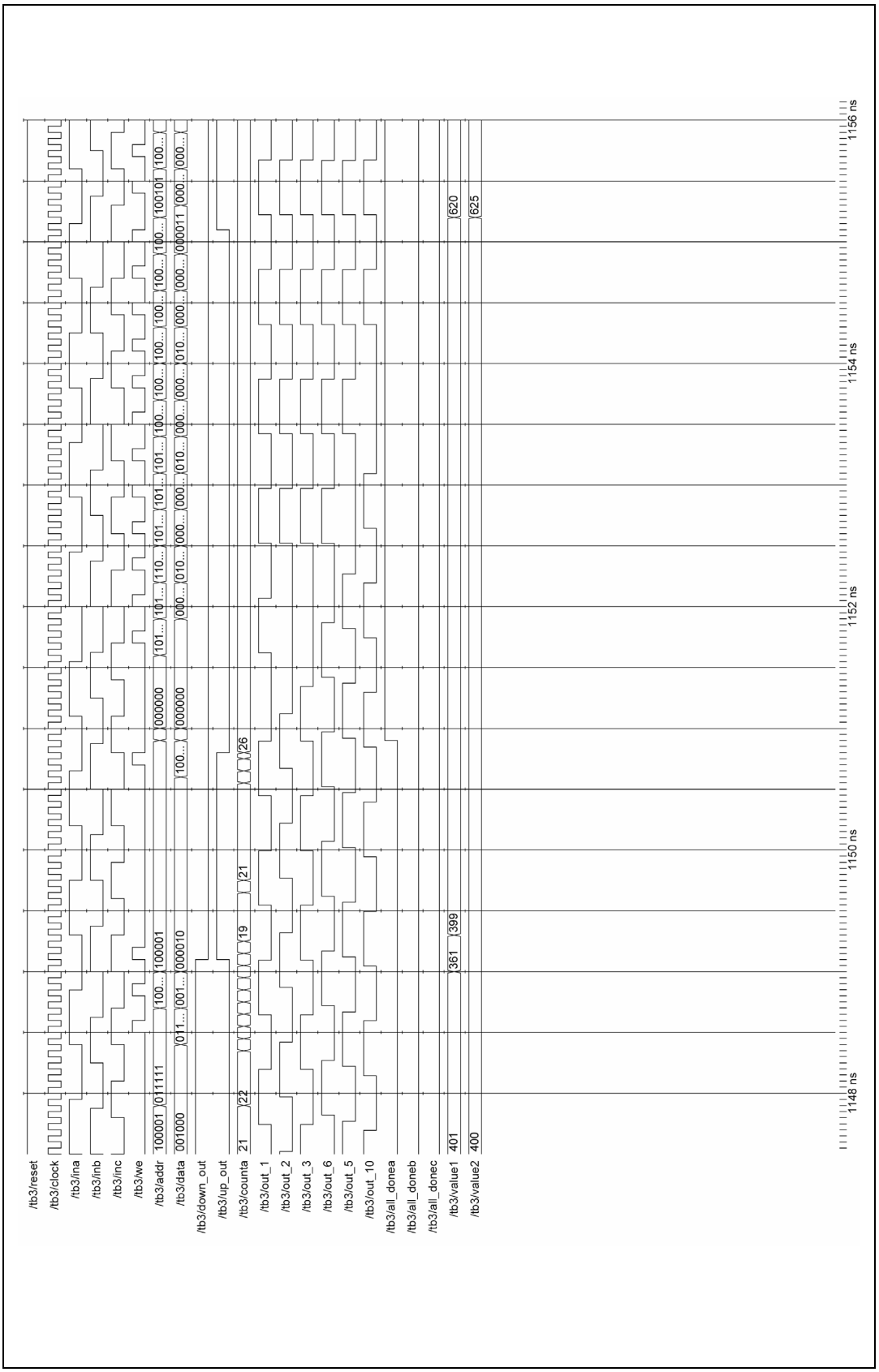

Figure A.9: Final stage of clock domain synchronization of clock thread A (clock period of 900 ps).

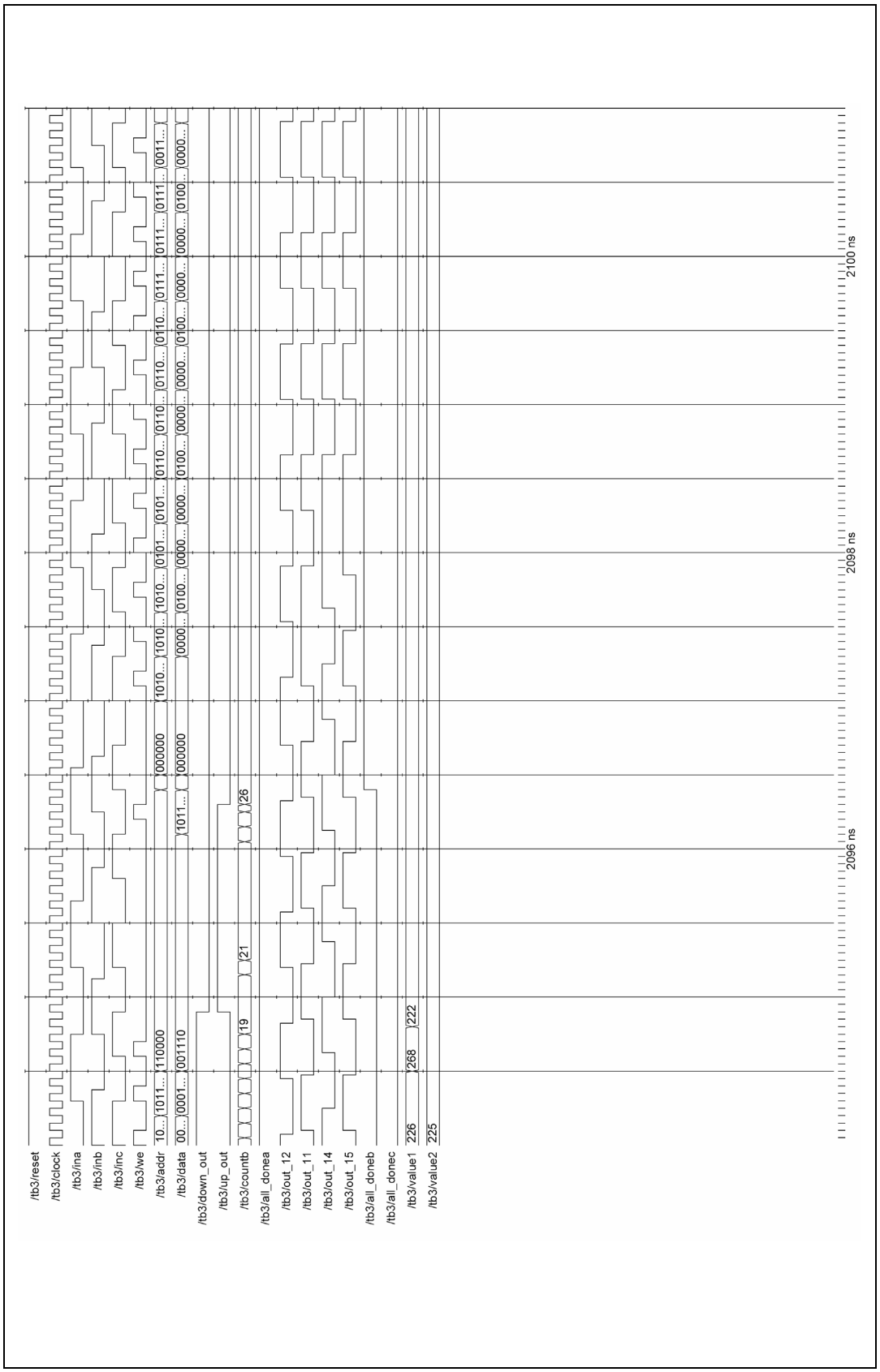

Figure A.10: Final stage of clock domain synchronization of clock thread B (clock period of 750 ps).

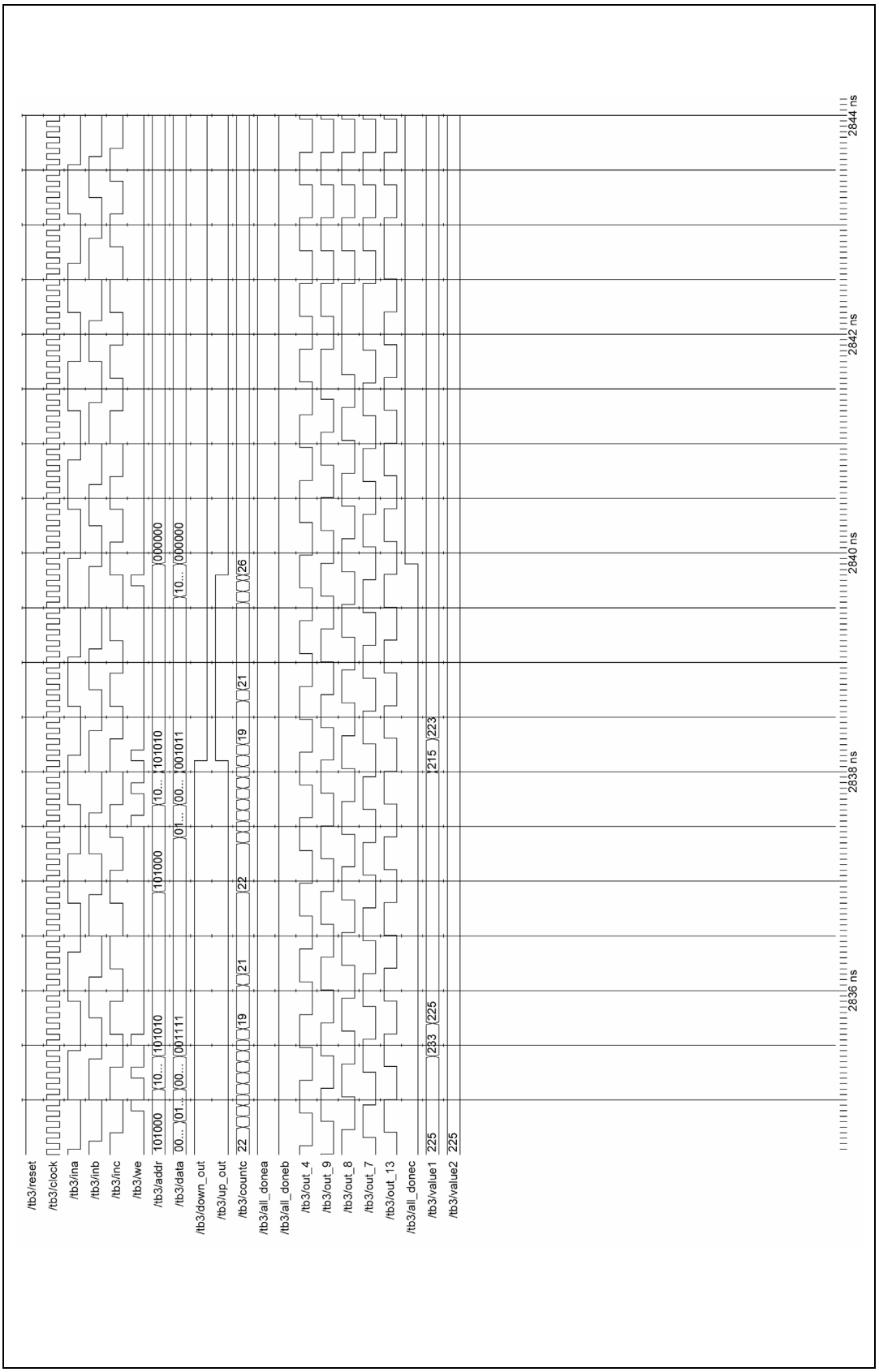

Figure A.11: Final stage of clock domain synchronization of clock thread C (clock period of 600 ps).

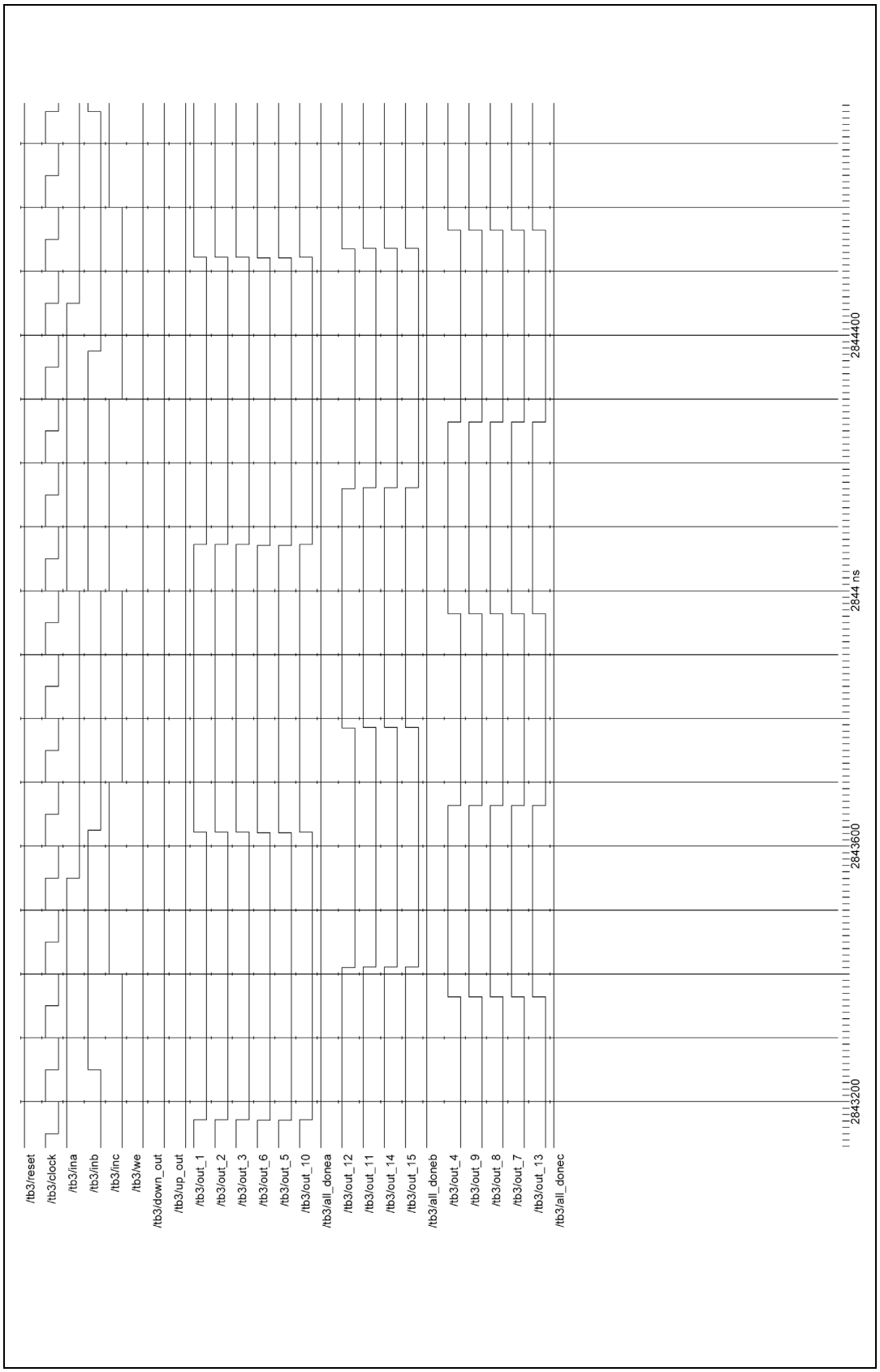

Figure A.12: Final synchronized end result of 15-tap reconfigurable clock distribution operating with three clock domains.

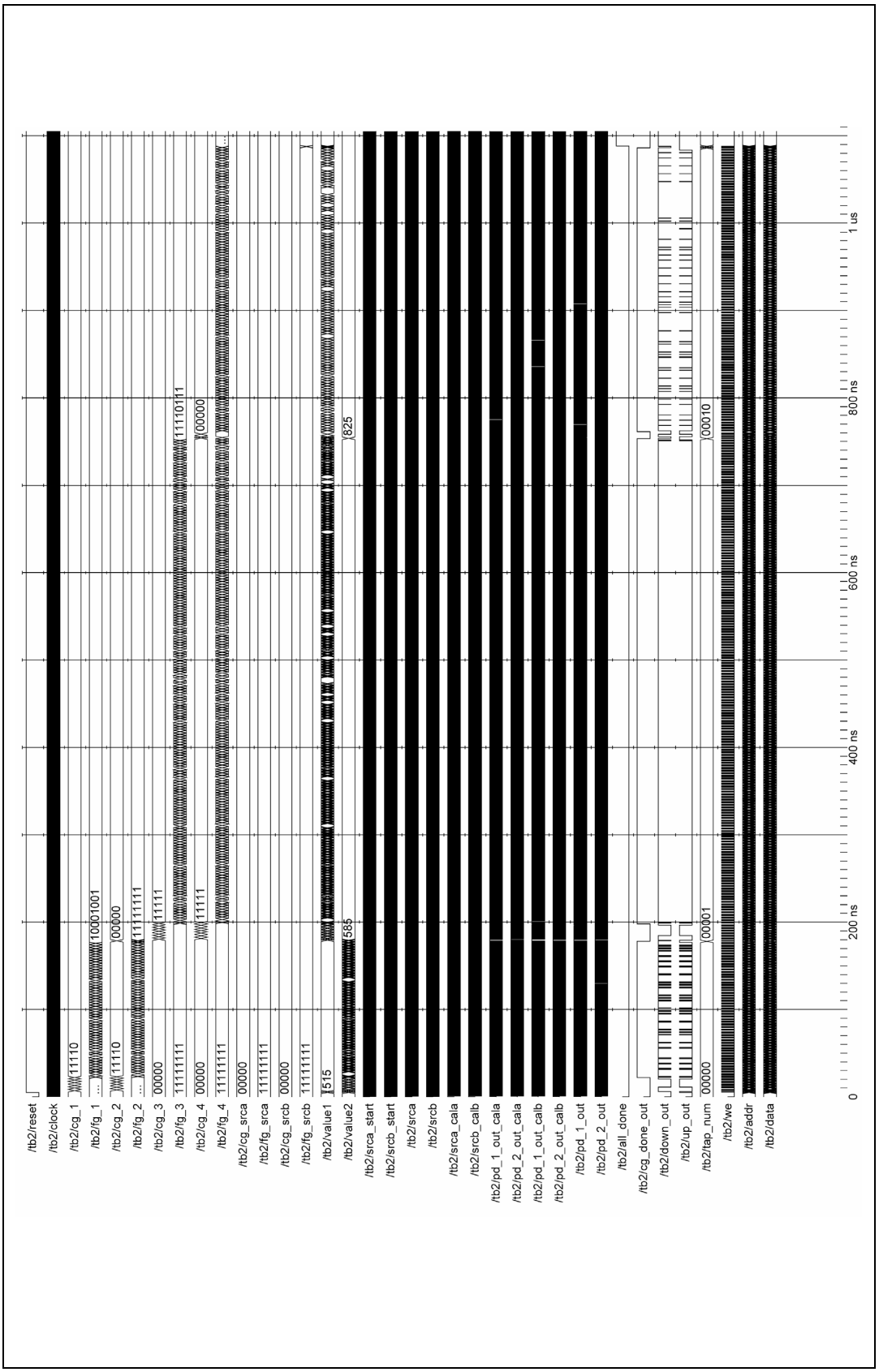

Figure A.13: Complete calibration and synchronization of two clock taps using BICSS.

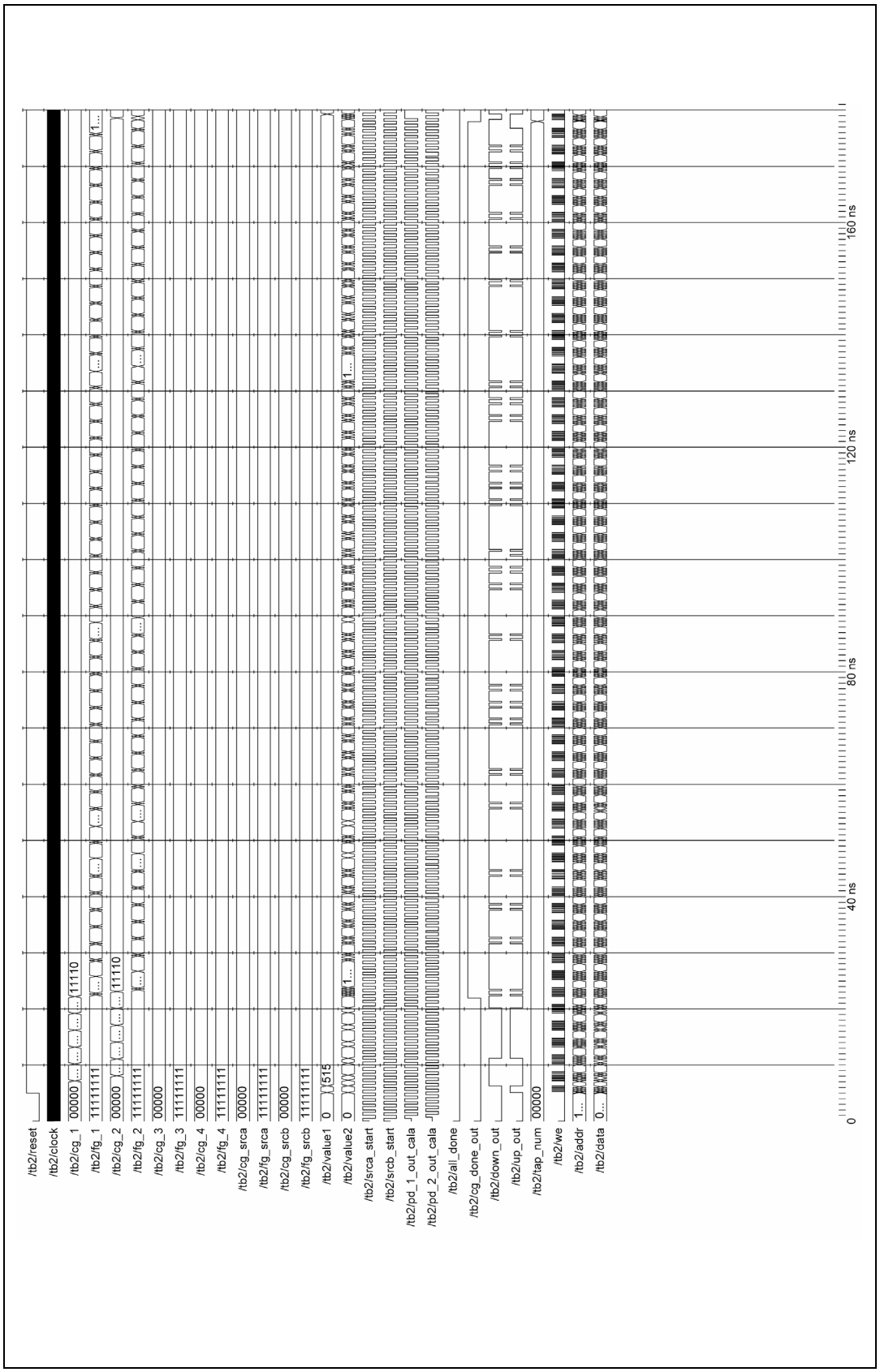

Figure A.14: Calibration of BICSS for transport delay from tap A to central BICSS circuitry.

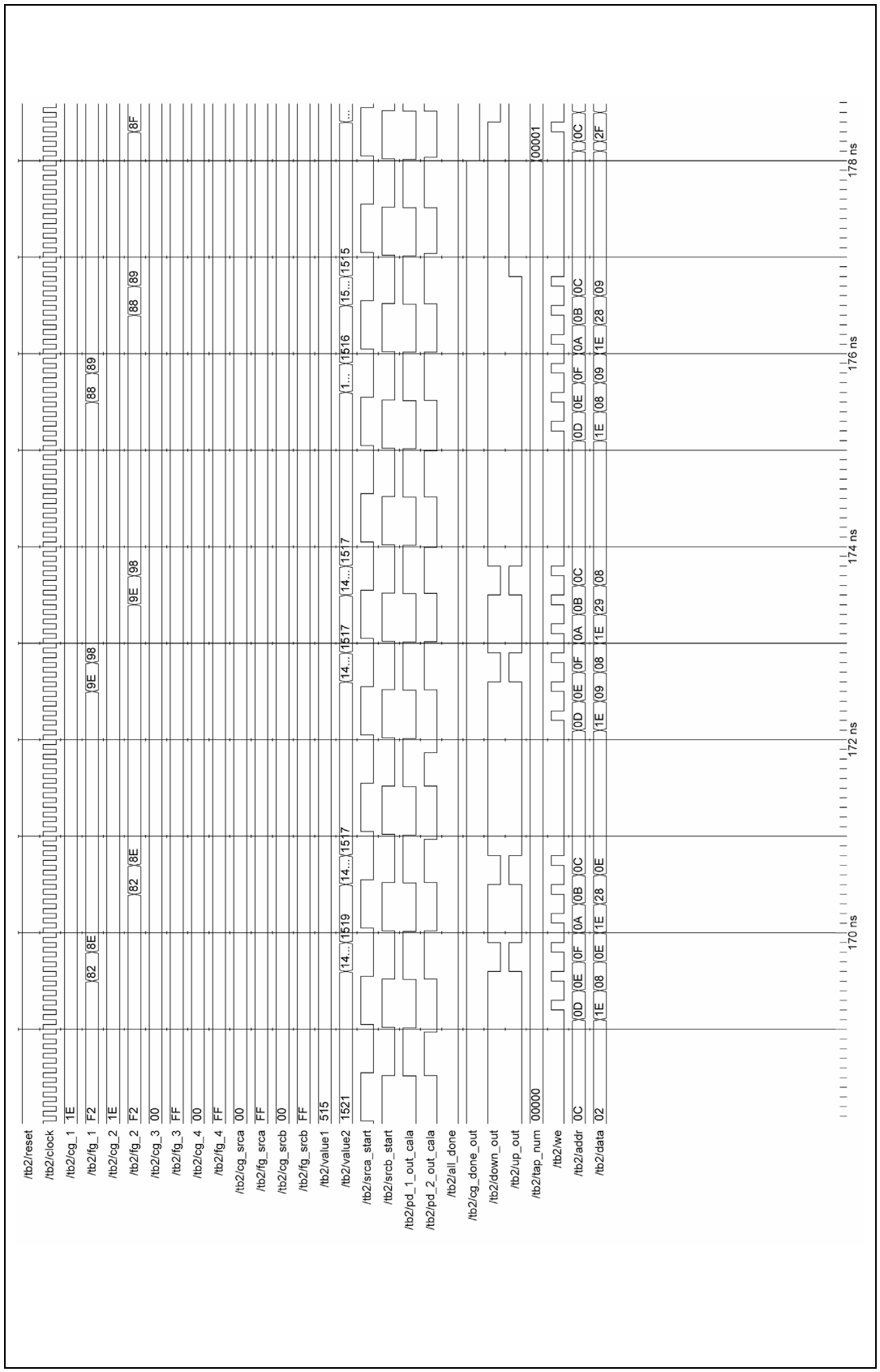

Figure A.15: Enlarged final stage of calibration of BICSS for transport delay from tap A to central BICSS circuitry.

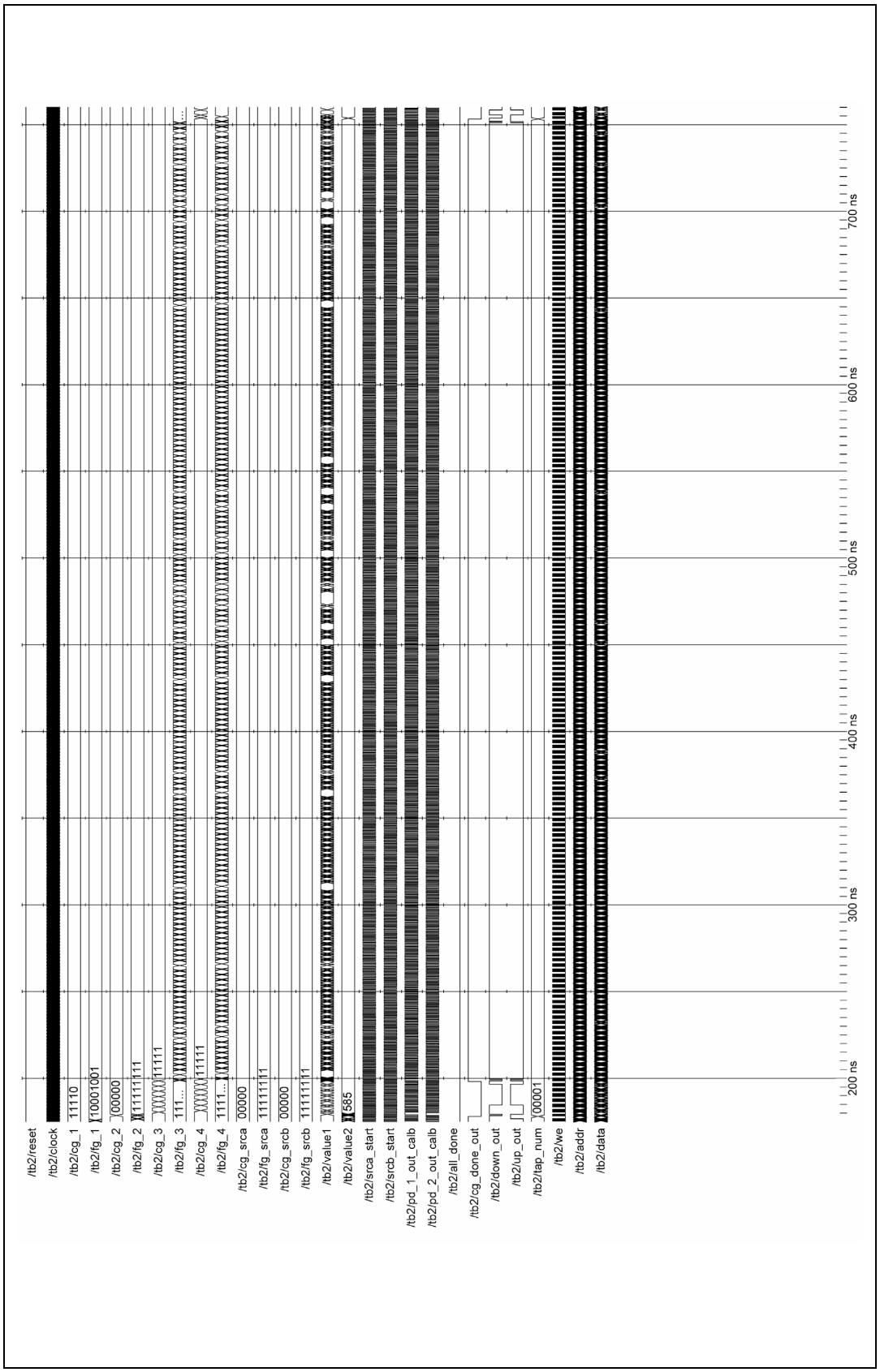

Figure A.16: Calibration of BICSS for transport delay from tap B to central BICSS circuitry.

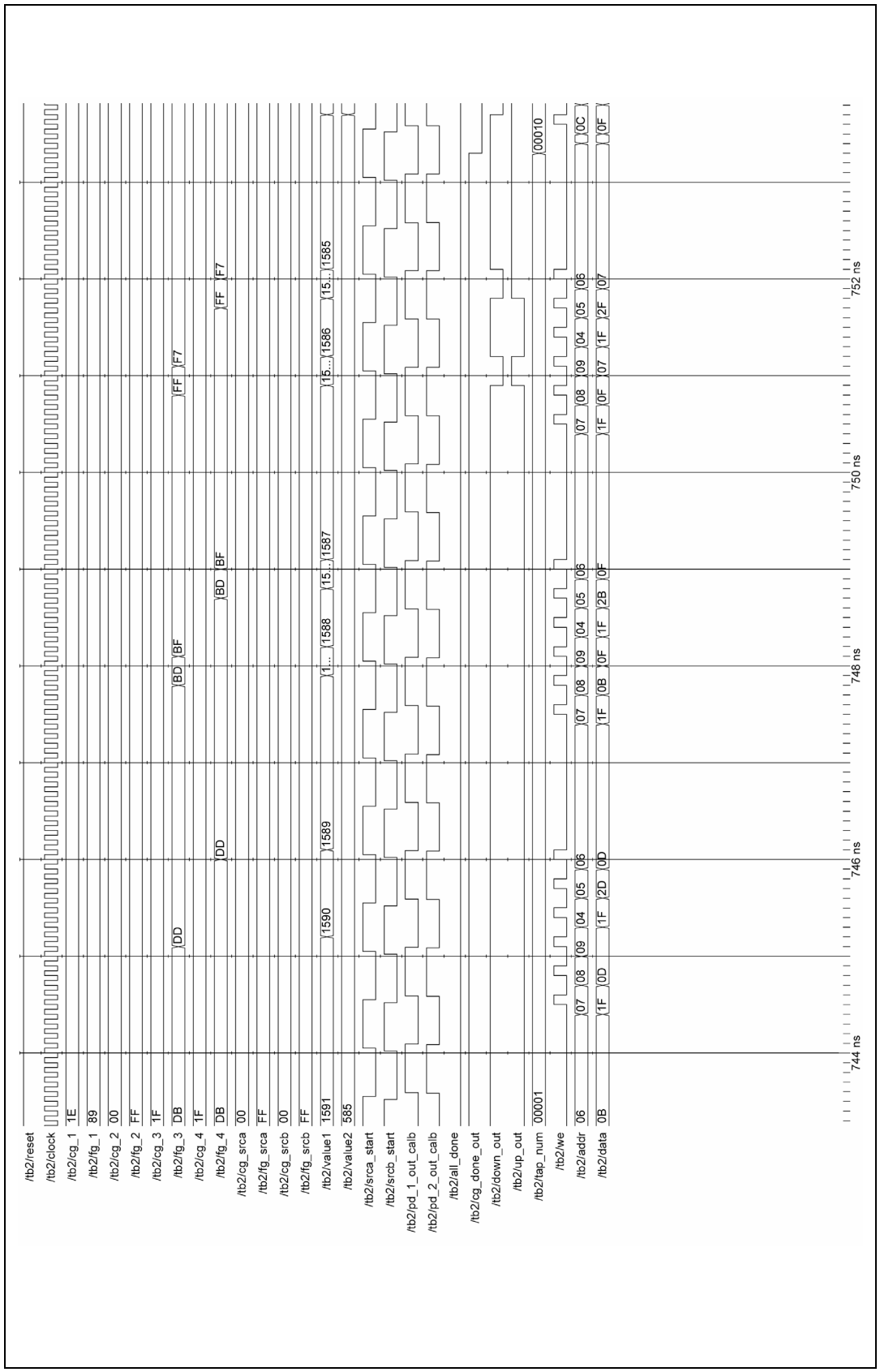

Figure A.17: Final end stage of calibration of BICSS for transport delay from tap B to central BICSS circuitry.

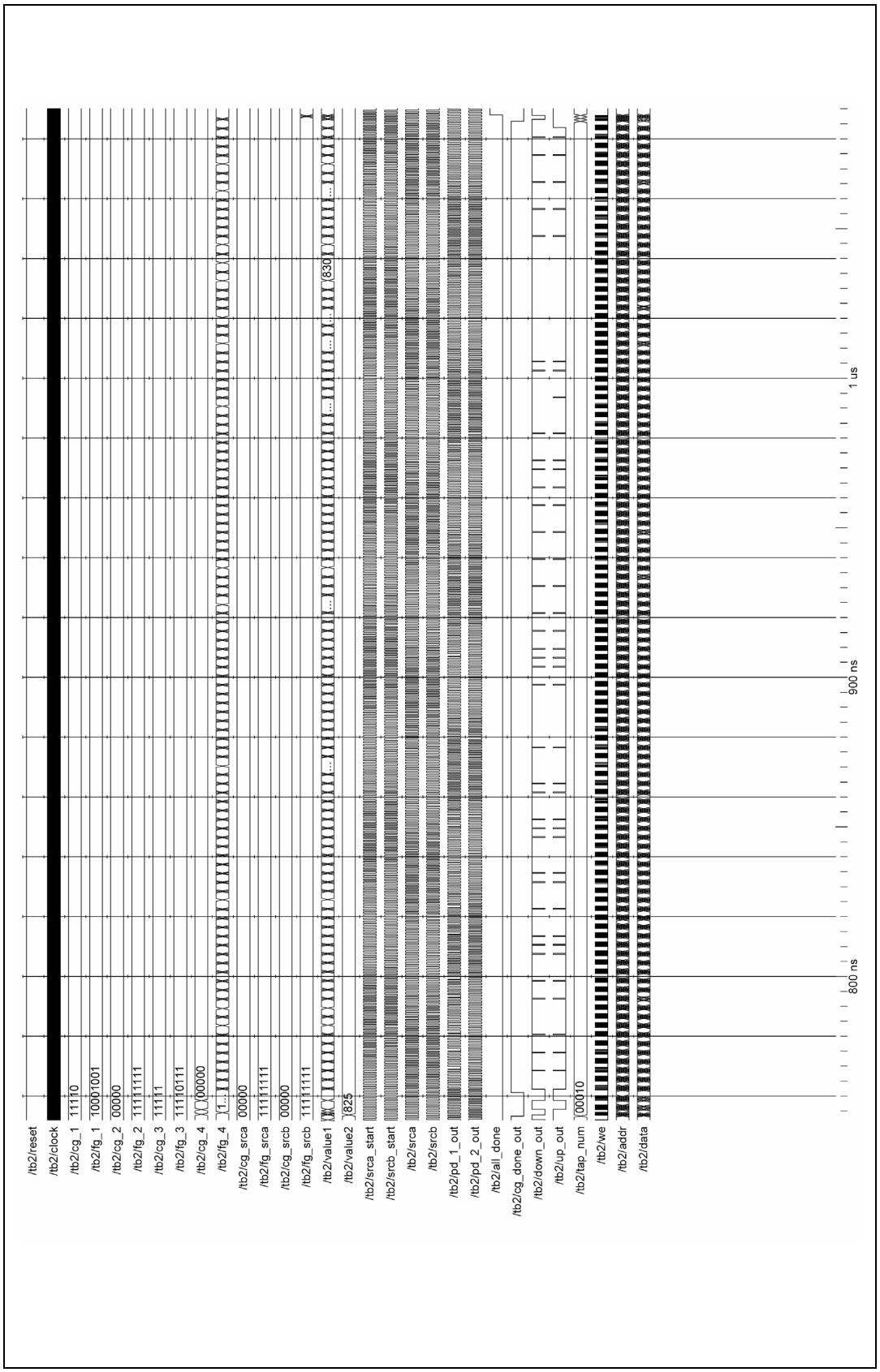

Figure A.18: Synchronization of tap clocks A and B using BICSS.

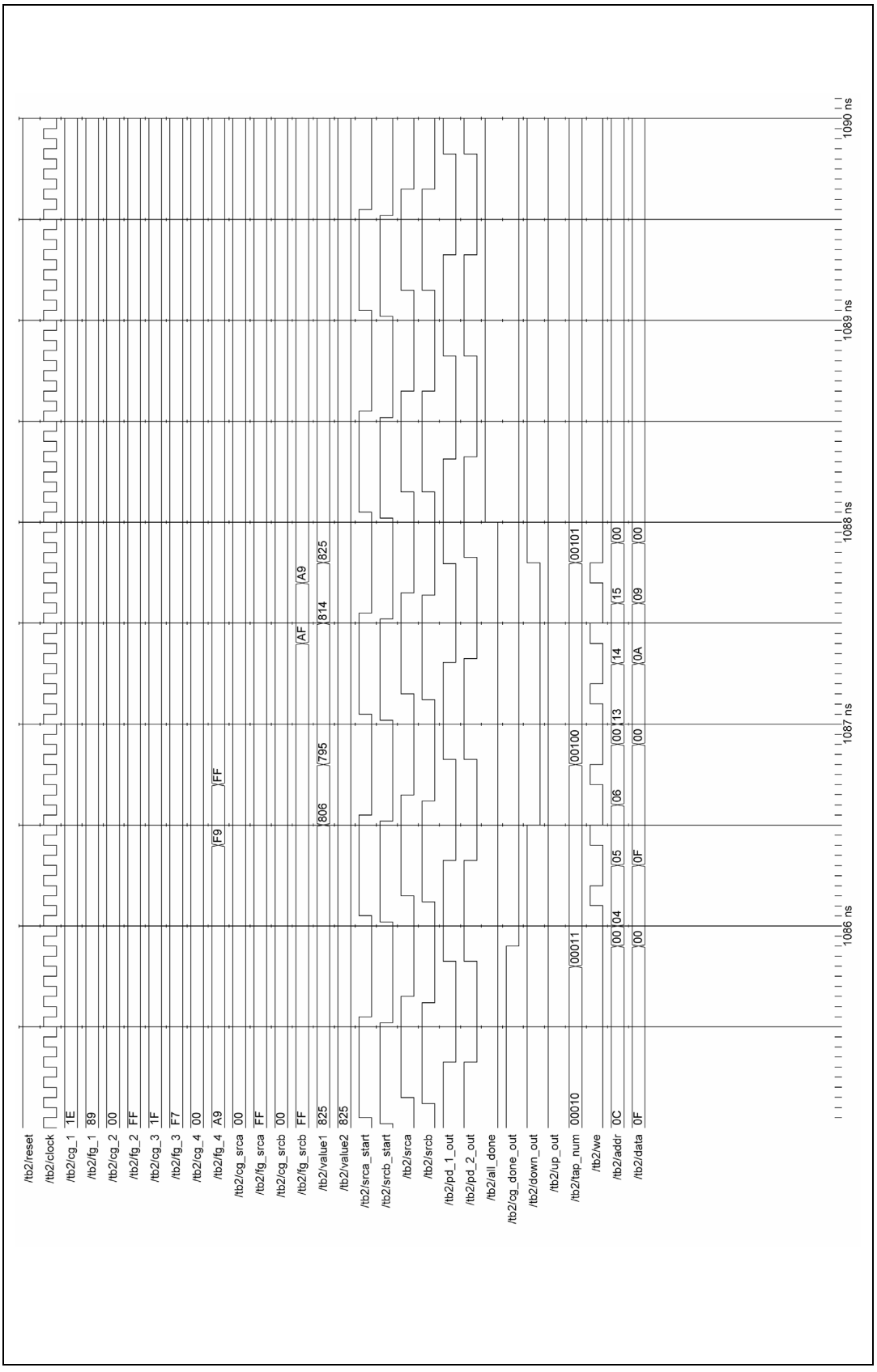

Figure A.19: Enlarged final stage of synchronization of tap clocks A and B using BICSS.

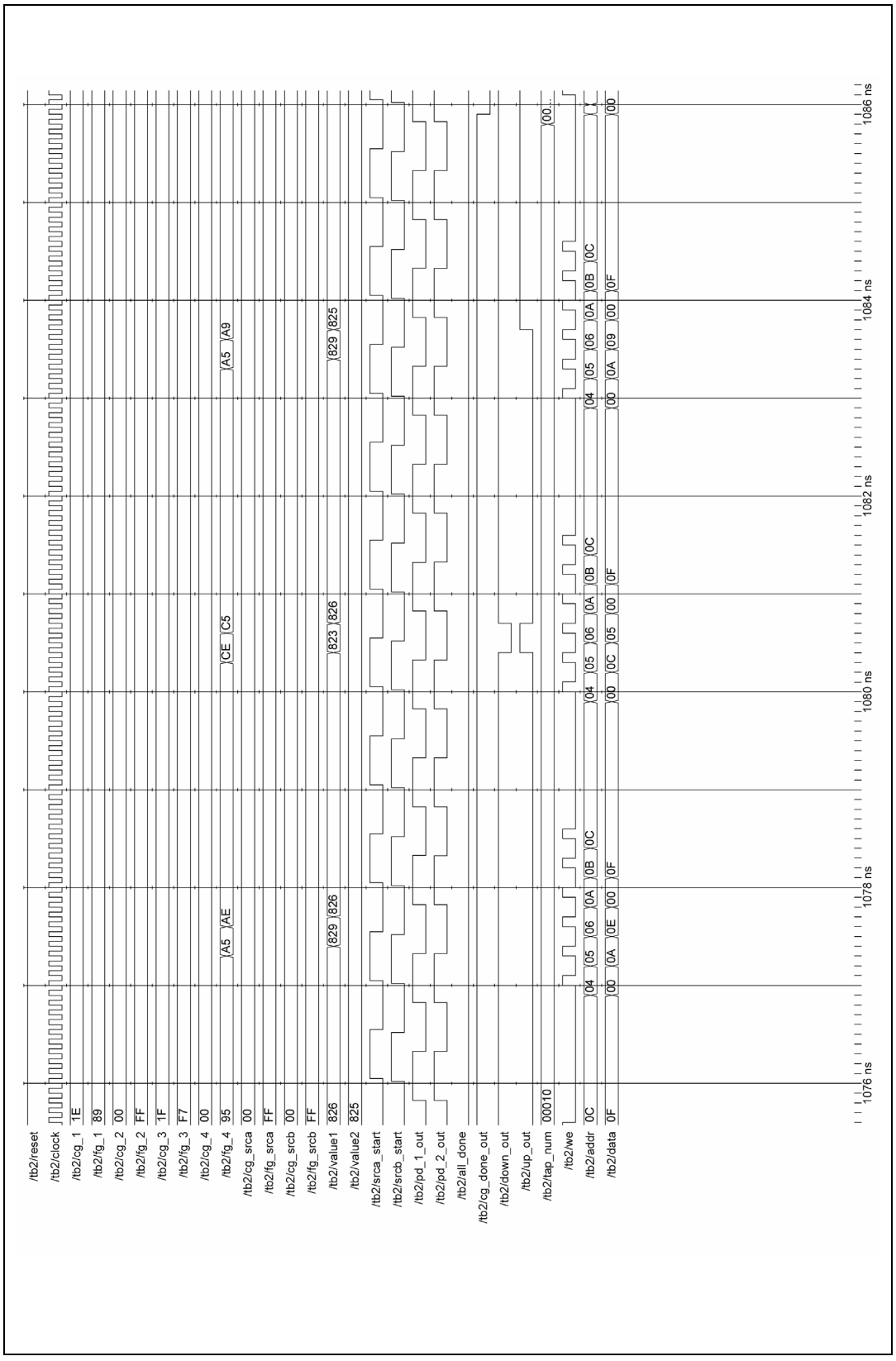

Figure A.20: Write-back and end stages of clock synchronization of tap clocks A and B using BICSS.

Appendix B:

Selected circuit drawings with transistor sizes

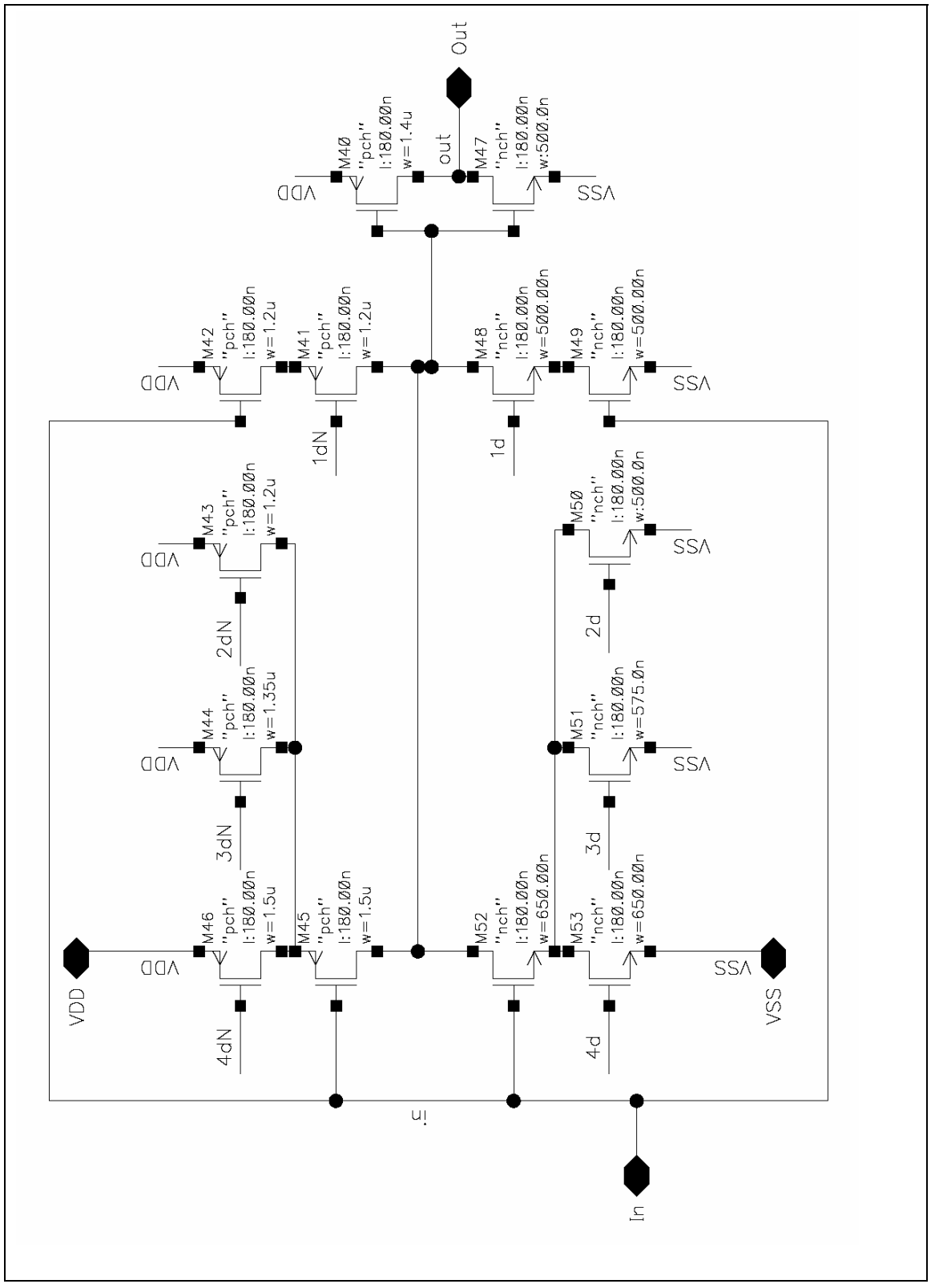

Figure B.1: Transistor sizes for fine grain delay cell (non-inverting) from Figure 7.2 for delay range 1.

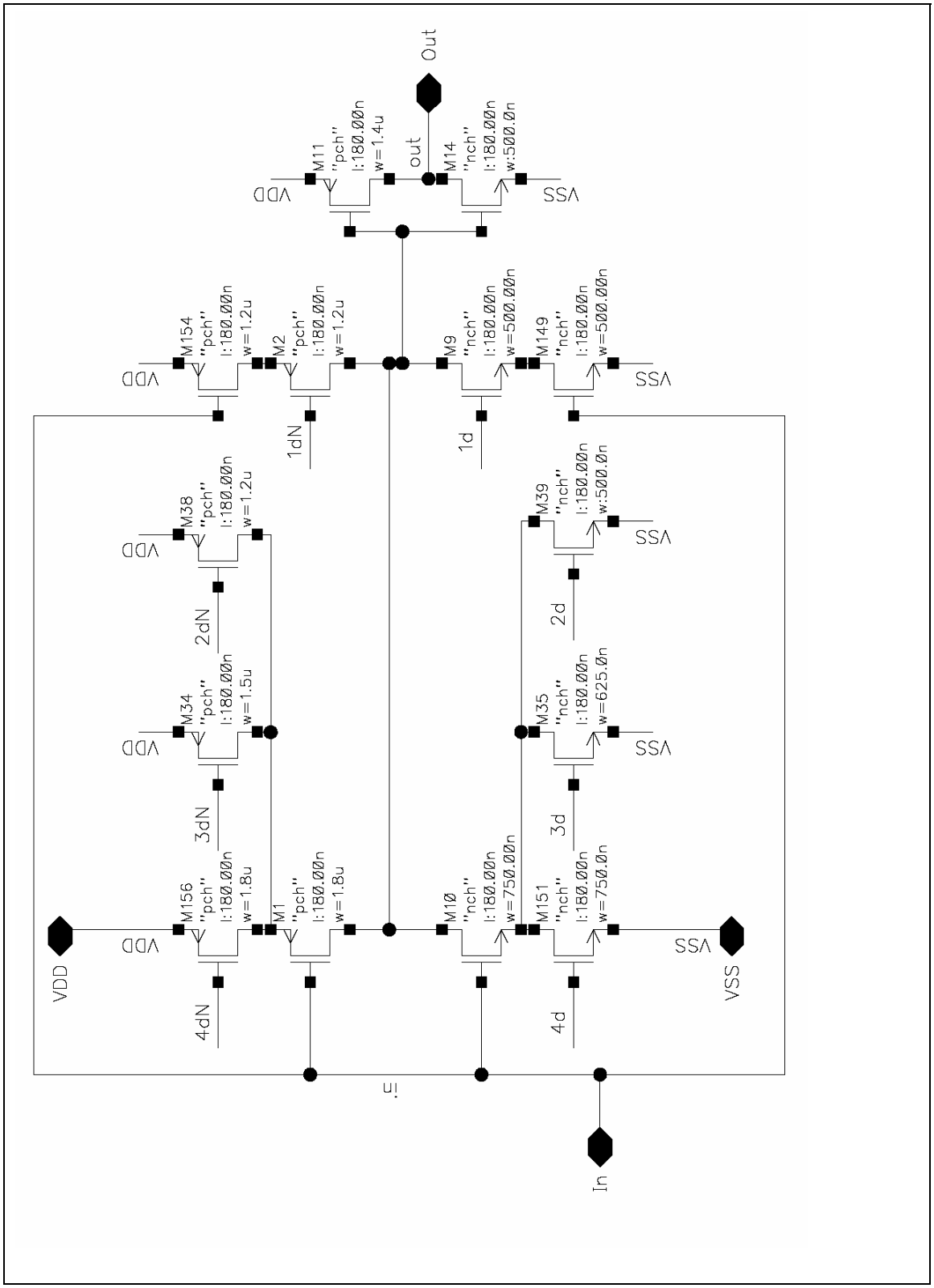

Figure B.2: Transistor sizes for fine grain delay cell (non-inverting) from Figure 7.2 for delay range 2.

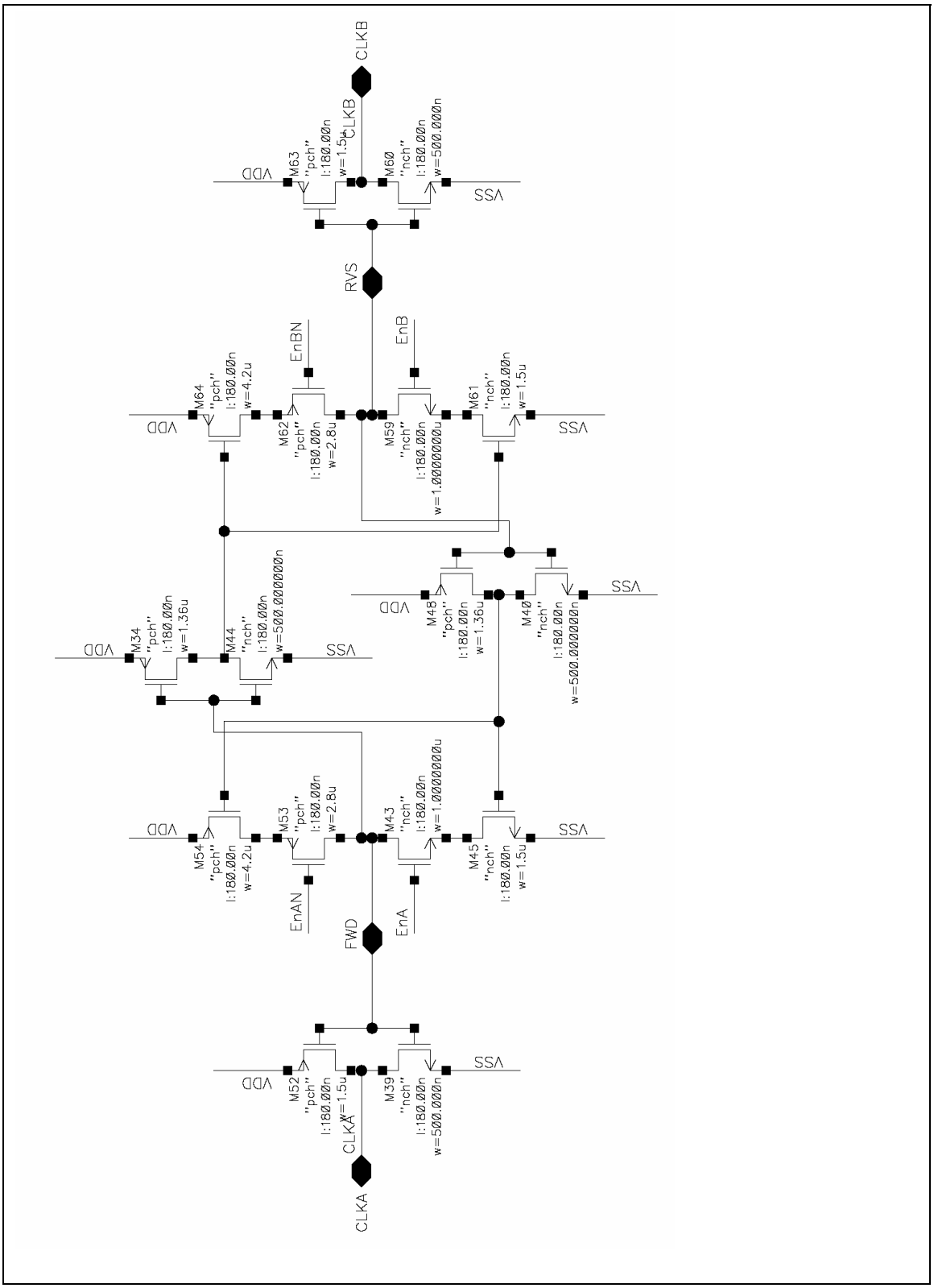

Figure B.3: Transistor sizes for tap bypass switch from Figure 7.10.

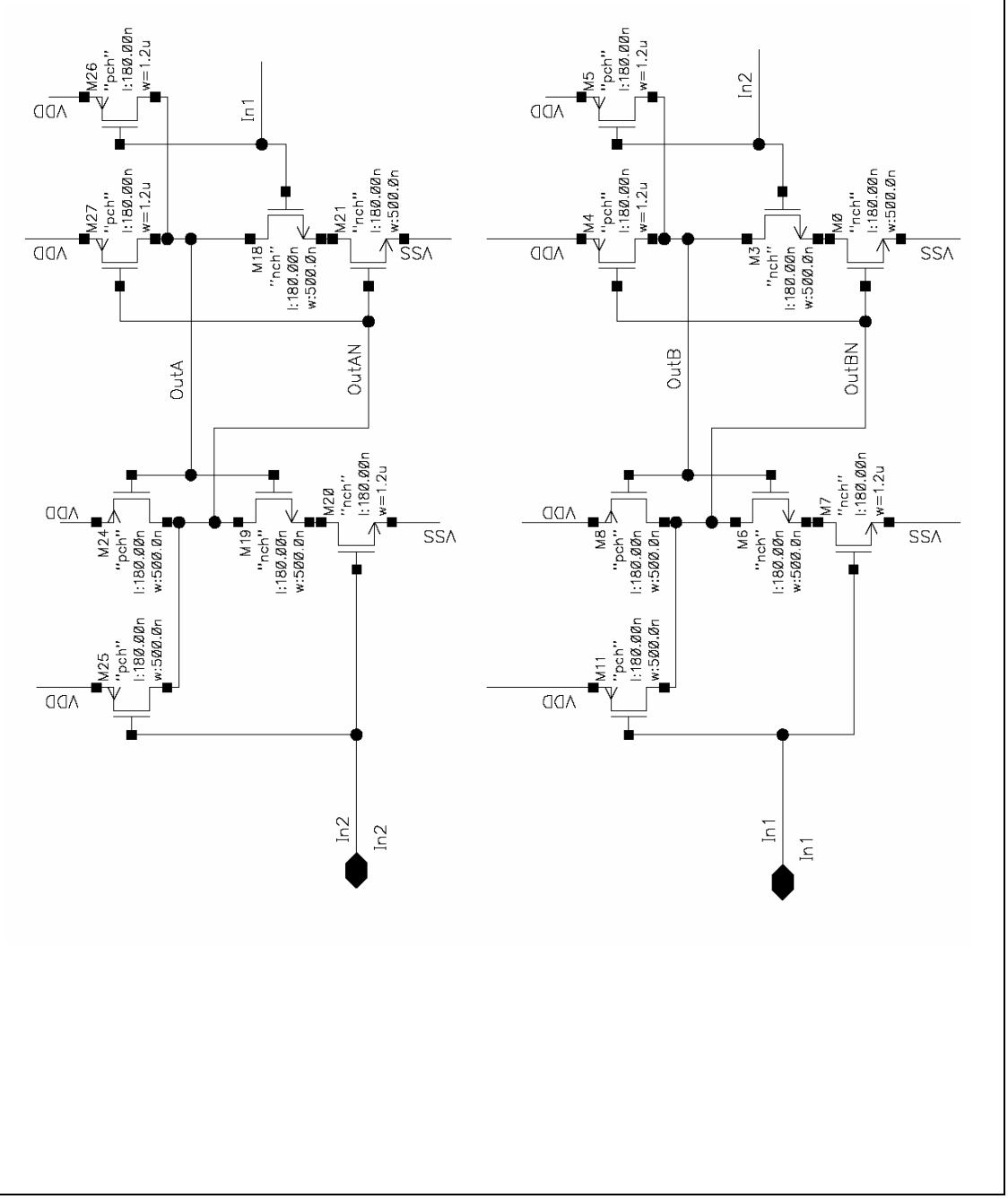

Figure B.4: Transistor sizes for input section of the fixed tolerance phase detector from Figure 7.15.

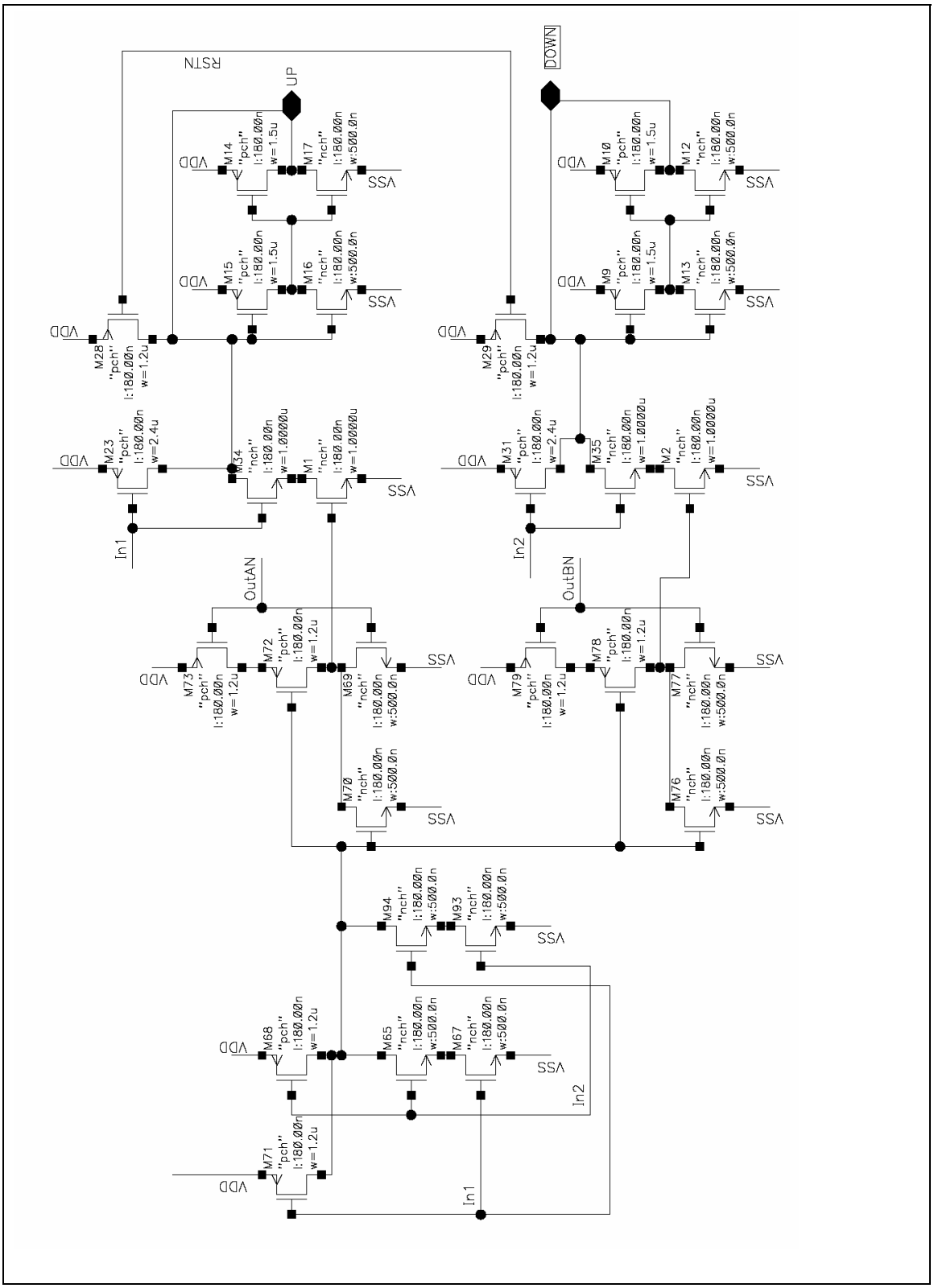

Figure B.5: Transistor sizes for memory section of the fixed tolerance phase detector from Figure 7.15.

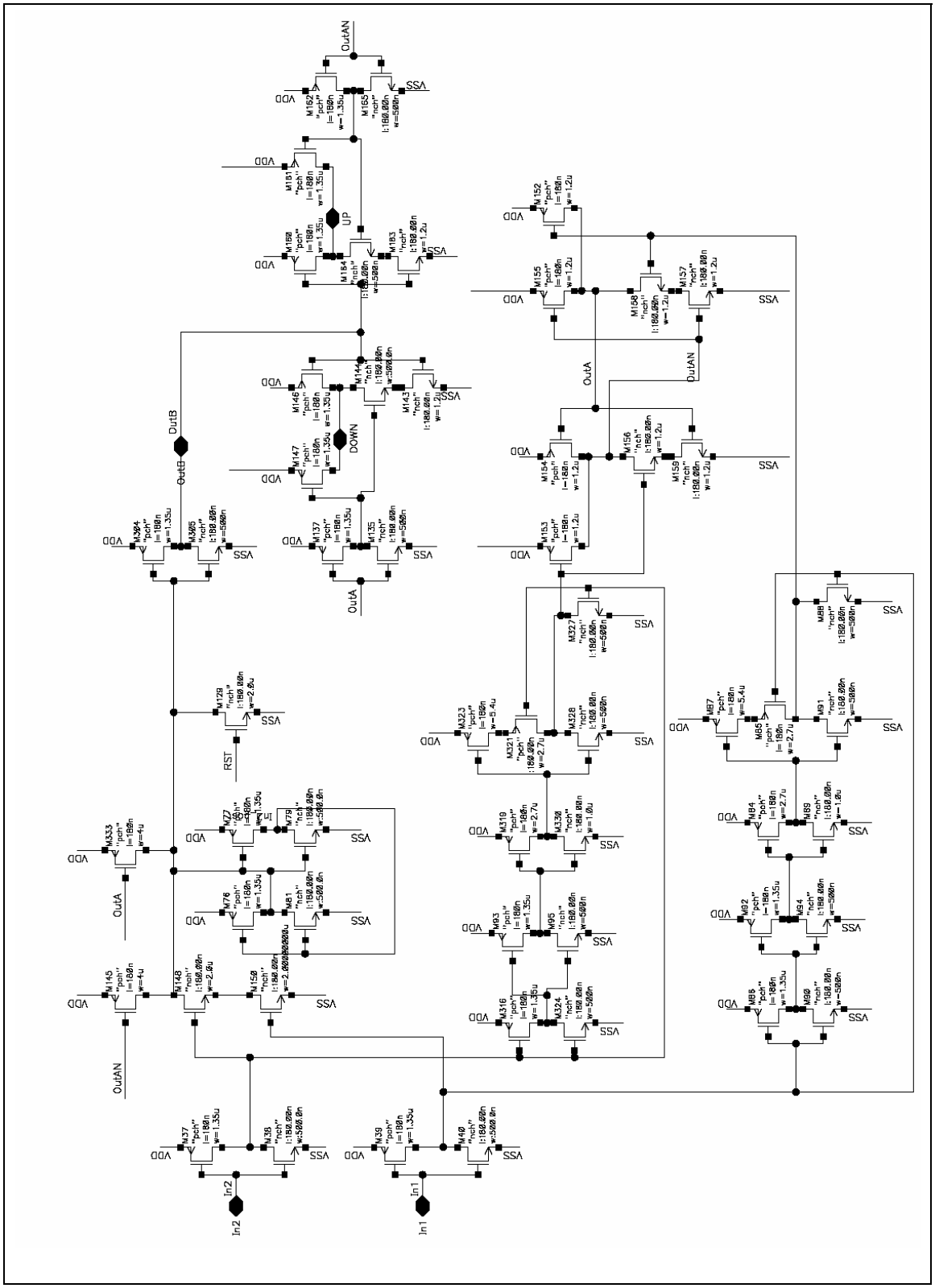

Figure B.6: Transistor sizes for modified phase detector for shared delay line system from Figure 7.19.

List of references

[1] BrainyMedia.com, "Charles Caleb Colton Quotes," http://www.brainyquote.com/quotes/quotes/c/charlescal108128.htm.

[2] H. Kalte, D. Langen, E. Vonnahme, A. Brinkmann, and U. Ruckert, "Dynamically reconfigurable system-on-programmable-chip," *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (EUROMICRO-PDP 2002),* 235-242.

[3] Wikipedia.org, “ASIC Verification,” http://en.wikipedia.org/wiki/Asic_verification.

[4] M. Radu, "Extensive Coverage of Functional Verification of Hardware Designs" *Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education (MSE 2007),* 101-102.

[5] P. Cheung, "Introduction to Digital Integrated Circuit Design," *Imperial College London,* http://www.ee.ic.ac.uk/pcheung/teaching/ee4_asic/notes/1-intro.pdf, 2007.

[6] G. Magklis, M. L. Scott, G. Semeraro, D. H. Albonesi, and S. Dropsho, "Profile-based dynamic voltage and frequency scaling for a multiple clock domain microprocessor," *Proceedings of the 30th International Symposium on Computer Architecture (ISCA 2003),* 14-25.

[7] A. Chattopadhyay, and Z. Zilic, "Reference-Based Clock Distribution Architectures," *Proceedings of the 49th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS 2006),* II-704-708.

[8] A. Chattopadhyay and Z. Zilic, "A CMOS Averaging Circuit for Programmable Clock Distributions," *CMC TEXPO Research Exhibition*, Oct. 2007, poster.

[9] A. Chattopadhyay, and Z. Zilic, "Design and operating characteristics of a reconfigurable clock distribution network," *Proceedings of the 2007 IEEE Northeast Workshop on Circuits and Systems (NEWCAS 2007),* 9-12.

[10] A. Chattopadhyay, and Z. Zilic, "Built-in Clock Skew System for On-line Debug and Repair," *Proceedings of the 2008 Design, Automation and Test in Europe Conference (DATE 2008),* 248-251.

[11] A. Chattopadhyay, and Z. Zilic, "Reconfigurable Clock Distribution Circuitry," *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS 2007),* 877-880.

[12] I. S. Kourtev, and E. G. Friedman, "Timing optimization through clock skew scheduling," *Kluwer Academic Publishers*, 2000.

[13] J. M. Rabaey, "Digital integrated circuits," *Prentice Hall Inc.*, 1996.

[14] E.G. Friedman, “Clock distribution networks in synchronous digital integrated circuits,” *Proceedings of the IEEE*, vol. 89, issue 5 (May 2001), 665-692.

[15] E. Buterbaugh, "Perfect Timing," *Cypress Semiconductor Corp.*, 2002.

[16] C. Yeh, G. Wilke, H. Chen, S. Reddy, H. Nguyen, T. Miyoshi, W. Walker, and R. Murgai, "Clock distribution architectures: a comparative study," *Proceedings of the 7th IEEE International Symposium on Quality Electronic Design (ISQED 2006)*, 7.

[17] Maxim Integrated Products, “Application Note 1916: An Introduction to Jitter in Communications Systems,” http://www.maxim-ic.com/appnotes.cfm/an_pk/1916.

[18] V. Bhargava, N. Haider, and N. Sarpotdar, “IO Clock Network Skew & Performance Analysis: A Pentium-D Case Study,” *Proceedings of the IEEE 2006 Custom Integrated Circuits Conference (CICC 2006)*, 345-348.

[19] D. Matzke, "Will Physical Scalability Sabotage Performance Gains?" *Computer*, 30, 9 (Sept. 1997), 37-39.

[20] G. Tosik, F. Abramowicz, Z. Lisik, and F. Gaffiot, “Clock Skew Analysis in Optical Clock Distribution Network,” *Proceedings of the 2007 International Conference on CAD Systems in Microelectronics (CADSM 2007)*, 422-425.

[21] N. Nedovic, and V. G. Oklobdzija, “Dual-edge triggered storage elements and clocking strategy for low-power systems,” *IEEE Transactions on VLSI Systems*, 13, 5 (May 2005), 577-590.

[22] M. A. El-Moursy, and E. G. Friedman, “Exponentially tapered H-tree clock distribution networks,” *IEEE Transactions on VLSI Systems*, 13, 8 (Aug. 2005), 971-975.

[23] M. Omana, D. Rossi and C. Metra, “Fast and low-cost clock deskew buffer,” *Proceedings of the 2004 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2004)*, 202-210.

[24] V. Varghese, T. Chen and P. Young, “Stability analysis of active clock deskewing systems using a control theoretic approach,” *Proceedings of the 2005 American Control Conference (ACC 2005)*, 3, 1758-1763.

[25] M. A. Karami, A. Afzali-Kusha, R. Faraji-Dana, and M. Rostami, “Quantitative Comparison of Optical and Electrical H, X, and Y clock Distribution Networks,” *Proceedings of the 2007 IEEE Computer Society Symposium on VLSI (ISVLSI 2007),* 488-489.

[26] A. Iyer, and D. Marculescu, "Power efficiency of voltage scaling in multiple clock multiple voltage cores," *Proceedings of the 2002 IEEE/ACM International Conference on Computer Aided Design (ICCAD 2002),* 379-386.

[27] S. Abe, M. Hashimoto, and T. Onoye, “Clock Skew Evaluation Considering Manufacturing Variability in Mesh-Style Clock Distribution,” *Proceedings of the 9th IEEE International Symposium on Quality Electronic Design (ISQED 2008),* 520-525.

[28] A. J. Drake, K. J. Nowka, T. Y. Nguyen, J. L. Burns, and R. B. Brown, “Resonant clocking using distributed parasitic capacitance,” *IEEE Journal of Solid-State Circuits*, 39, 9 (Sept. 2004), 1520-1528.

[29] Jang-Ying Chueh, M. C. Papaefthymiou, and C. H. Ziesler, “Two-phase resonant clock distribution,” *Proceedings of the 2005 IEEE Computer Society Symposium on VLSI (ISVLSI 2005)*, 65-70.

[30] B. Mesgarzadeh, M. Hansson, and A. Alvandpour, “Jitter Characteristic in Charge Recovery Resonant Clock Distribution,” *IEEE Journal of Solid-State Circuits*, 42, 7 (July 2007), 1618-1625.

[31] B. Mesgarzadeh, M. Hansson, and A. Alvandpour, “Low-power bufferless resonant clock distribution networks,” *Proceedings of the 50th IEEE Midwest Symposium on Circuits and Systems (MWSCAS 2007)*, 960-963.

[32] Cyclos Semiconductor Inc., Elizabeth Resonant-Clocked ARM926EJ-S (Information brochure), http://www.cyclos-semi.com/NewFiles/ElizLitFront.pdf, 2008.

[33] V. S. Sathe, J. C. Kao, and M. C. Papaefthymiou, “Resonant-Clock Latch-Based Design,” *IEEE Journal of Solid-State Circuits*, 43, 4 (Apr. 2008), 864-873.

[34] M. Hansson, B. Mesgarzadeh, and A. Alvandpour, “1.56 GHz On-chip Resonant Clocking in 130nm CMOS,” *Proceedings of the IEEE 2006 Custom Integrated Circuits Conference (CICC 2006)*, 241-244.

[35] Zheng Xu, and K. L. Shepard, “Low-Jitter Active Deskewing Through Injection-Locked Resonant Clocking,” *Proceedings of the IEEE 2007 Custom Integrated Circuits Conference (CICC 2007)*. 9-12.

[36] S. C. Chan, K. L. Shepard, and P. J. Restle, "Design of resonant global clock distributions," *Proceedings of the 21st International Conference on Computer Design (ICCD 2003),* 248-253.

[37] V. H. Cordero, and S. P. Khatri, “Clock Distribution Scheme using Coplanar Transmission Lines,” *Proceedings of the 2008 Design, Automation and Test in Europe Conference (DATE 2008)*, 985-990.

[38] G. Venkataraman, Jiang Hu, and F. Liu, “Integrated Placement and Skew Optimization for Rotary Clocking,” *IEEE Transactions on VLSI Systems*, 15, 2 (Feb. 2007), 149-158.

[39] Z. Yu, and X. Liu, “Low-Power Rotary Clock Array Design,” *IEEE Transactions on VLSI Systems*, 15, 1 (Jan. 2007), 5-12

[40] E. Ogunti, M. Frank, and S. Y. Foo, “Power analysis of resonant clocks,” *Proceedings of the 2008 International Conference on Computer and Communications Engineering (ICCCE 2008)*, 336-340.

[41] Wood, J., Lipa, S., Franzon, P., and Steer, M. "Multi-gigahertz low-power low-skew rotary clock scheme," *Digest of Technical Papers of the 2001 IEEE International Solid-State Circuits Conference (ISSCC 2001)*, 400-401, 470.

[42] A. L. Sobczyk, A. W. Luczyk, and W. A. Pleskacz, “Power Dissipation in Basic Global Clock Distribution Networks,” *Proceedings of the 10th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2007)*, 1-4.

[43] G. Tosik, L. M. S. Gallego, and Z. Lisik, “Different Approaches for Clock Skew Analysis in Present and Future Synchronous IC's,” *Proceedings of the IEEE 2007 Region 8 Eurocon (EUROCON 2007)*, 1227-1232.

[44] W.-C. D Lam, J. Jam, C.-K. Koh, V. Balakrishnan, and Y. Chen, “Statistical based link insertion for robust clock network design,” *Proceedings of the 2005 IEEE/ACM International Conference on Computer Aided Design (ICCAD 2005)*, 588-591.

[45] R. Saeidi, and N. Masoumi, “Clock Skew Reduction by Link-region Technique,” *Proceedings of the 49th IEEE Midwest Symposium on Circuits and Systems (MWSCAS 2006)*. 213-216.

[46] G. Venkataraman, N. Jayakumar, J. Hu, P. Li, S. Khatri; A. Rajaram; P. McGuinness, and C. Alpert, “Practical techniques to reduce skew and its variations in buffered clock networks,” *Proceedings of the 2005 IEEE/ACM International Conference on Computer Aided Design (ICCAD 2005)*, 592-596.

[47] M. Mori, Hongyu Chen; B. Yao, and Chung-Kuan Cheng, “A mulitple level network approach for clock skew minimization with process variations,” *Proceedings of the 2004 Asia and South Pacific Design Automated Conference (ASP-DAC 2004)*, 263-268.

[48] W.D. Grover, J. Brown, T. Friesen and S. Marsh, “All-digital multipoint adaptive delay compensation circuit for low skew clock distribution,” *Electronics Letters*, vol. 31, issue 23 (9 Nov. 1995), 1996-1998.

[49] T. Knight, and H. M. Wu, "A method for skew-free distribution of digital signals using matched variable delay lines," *Digest of Technical Papers from the 1993 Symposium on VLSI Circuits (ISVLSI 1993),* 19-20.

[50] V. Prodanov, and M. Banu, “GHz Serial Passive Clock Distribution in VLSI Using Bidirectional Signaling,” *Proceedings of the IEEE 2006 Custom Integrated Circuits Conference (CICC 2006)*, 285-288.

[51] J. Lamoureux, S. J. E. Wilton, “Clock-Aware Placement for FPGAs,” *Proceedings of the 2007 International Conference on Field Programmable Logic and Applications (FPL 2007)*, 124-131.

[52] Altera Inc., Stratix IV Handbook, http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf, 2008.

[53] G. Venkataraman, C. N. Sze, and Hu Jiang, "Skew scheduling and clock routing for improved tolerance to process variations," *Proceedings of the 2005 Asia and South Pacific Design Automation Conference (ASP-DAC 2005),* 1, 594-599.

[54] M. J. M Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, “Matching properties of MOS transistors” *IEEE Journal of Solid-State Circuits*, 24, 5 (Oct 1989), 1433-1439.

[55] A. Maxim, and M. Gheorghe, “A novel physical based model of deep-submicron CMOS transistors mismatch for Monte Carlo SPICE simulation,” *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems (ISCAS 2001)*, 5, 511-514.

[56] A. Narasimhan, and R. Sridhar, “Impact of Variability on Clock Skew in H-tree Clock Networks,” *Proceedings of the 8th IEEE International Symposium on Quality Electronic Design (ISQED 2007),* 458-466.

[57] M. Nekili, Y. Savaria, and G. Bois, "Minimizing process-induced skew using delay tuning," *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems (ISCAS 2001),* 4, 426-429.

[58] S. A. Tawfik, and V. Kursun, “Low-Power Low-Voltage Hot-Spot Tolerant Clocking with Suppressed Skew,” *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS 2007)*, 645-648.

[59] D. P. Dimitrov, “Deep-Submicron MOS Transistor Matching: A Case Study,” *Proceedings of the 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2008)*, 1-4.

[60] Long Jieyi, Ku Ja Chun, S. O. Memik, and Y. Ismail, "A self-adjusting clock tree architecture to cope with temperature variations," *Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2007).* 75-82.

[61] K. Duraisami, P. Sithambaram, A. Sathanur, A. Macii, E. Macii, and M. Poncino, "Design Exploration of a Thermal Management Unit for Dynamic Control of Temperature-Induced Clock Skew," *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS 2007),* 1061-1064.

[62] G. Geannopoulos, and X. Dai, “An adaptive digital deskewing circuit for clock distribution networks,” *Digest of Technical Papers of the 1998 IEEE International Solid-State Circuits Conference (ISSCC 1998)*, 400-401.

[63] S. Hassoun, and C. J. Alpert, "Optimal path routing in single- and multiple-clock domain systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 22, 11 (Nov. 2003), 1580-1588.

[64] M. Saint-Laurent, M. Swaminathan and J.D. Meindl, “On the micro-architectural impact of clock distribution using multiple PLLs,” *Proceedings of the 2001 IEEE International Conference on Computer Design (ICCD 2001)*, 214-220.

[65] T. Yamashita, T. Fujimoto, and K. Ishibashi, “A dynamic clock skew compensation circuit technique for low power clock distribution.” *Proceedings of the 2005 IEEE International Conference on Integrated Circuit Design and Technology (ICICDT 2005)*, 7-10.

[66] A. Kapoor, N. Jayakumar and S. P. Khatri, “A novel clock distribution and dynamic de-skewing methodology,” *Proceedings of the 2004 IEEE/ACM International Conference on Computer Aided Design (ICCAD 2004)*, 626-631.

[67] H. Lee, H. Q. Nguyen and D.W. Potter, “Design self-synchronized clock distribution networks in an SoC ASIC using DLL with remote clock feedback,” *Proceedings of the 13th IEEE International ASIC/SOC Conference (ASICSOC 2000)*, 248-252.

[68] S. Tam, S. Rusu, U. Nagarji Desai, R. Kim, Ji Zhang and I. Young, “Clock generation and distribution for the first IA-64 microprocessor,” *IEEE Journal of Solid-State Circuits*, 35, 11 (Nov. 2000), 1545-1552.

[69] D. Duarte, N. Vijaykrishnan, M. J. Irwin, H.-S. Kim, and G. McFarland, "Impact of scaling on the effectiveness of dynamic power reduction schemes," *Proceedings of the 2002 IEEE International Conference on Computer Design (ICCD 2002),* 382-387.

[70] Liu Zhiyu, and V. Kursun, "Temperature dependent leakage power characteristics of dynamic circuits in sub-65 nm CMOS technologies," *Proceedings of the 48th Midwest Symposium on Circuits and Systems (MWSCAS 2005),* 1, 551-554.

[71] R. Tessier, "Power Reductions Techniques for FPGAs," *University of Massachusetts Amherst,* http://www.ecs.umass.edu/ece/tessier/courses/697ff/lect22-ece697f.ppt, 2006.

[72] P. Zarkesh-Ha, and J. D. Meindl, "Optimum chip clock distribution networks," *Proceedings of the 1999 IEEE International Conference Interconnect Technology (ICIT 1999),* 18-20.

[73] K. Masselos, "Low Power Design," *Imperial College London,* http://cas.ee.ic.ac.uk/people/kostas/web%20page%20material/Lecture%208%20-%20Low%20power%20design.pdf, 2005.

[74] M. Horowitz, E. Alon, D. Patil, S. Naffziger, K. Rajesh, and K. Bernstein, "Scaling, power, and the future of CMOS," *Technical Digest of 2005 IEEE International Electron Devices Meeting (IEDM 2005)*, 2005. 7.

[75] R. L. Aguiar and D. M. Santos, “Wide-area clock distribution using controlled delay lines,” *Proceedings of the 5th IEEE International Conference on Electronics, Circuits & Systems (ICECS 1998)*, 2, 63-66.

[76] D.E. Brueske and S.H.K. Embabi, “A dynamic clock synchronization technique for large systems,” *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging*, 17, 3 (Aug. 1994), 350-361.

[77] S. Zanella, A. Nardi, A. Neviani, M. Quarantelli, S. Saxena and C. Guardiani, “Analysis of the impact of process variations on clock skew,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 13, issue 4 (Nov. 2000), 401-407.

[78] S. Zanella, A. Nardi, M. Quarantelli, A. Neviani, and C. Guardiani, "Analysis of the impact of intra-die variance on clock skew," *Proceedings of the 4th International Workshop on Statistical Metrology (IWSM 1999),* 14-17.

[79] C.-Y. Yang and S.-I Liu. “A one-wire approach for skew-compensating clock distribution based on bidirectional techniques,” *IEEE Journal of Solid-State Circuits*, vol. 36, issue 2 (Feb. 2001), 266-272.

[80] M. M. Gourary, S. G. Rusakov, S. L. Ulyanov, M. M. Zharov, K. K. Gullapalli and B. J. Mulvaney, "Approximation approach for timing jitter characterization in circuit simulators," *Proceedings of the 2003 Design, Automation and Test in Europe Conference (DATE 2003)*, 156-161.

[81] D. Garrett, M. Stan, and A. Dean, "Challenges in clockgating for a low power ASIC methodology," *Proceedings of the 1999 International Symposium on Low Power Electronics and Design (ISLPED 1999),* 176-181.

[82] Wang Qi, and S. Roy, "Power minimization by clock root gating," *Proceedings of the. 2003 Asia and South Pacific Design Automation Conference (ASP-DAC 2003),* 249-254.

[83] A. Chattopadhyay, and Z. Zilic, “GALDS: a complete framework for designing multiclock ASICs and SoCs,” *IEEE Transactions on VLSI*, 13, 6 (June 2005), 641–654.

[84] Harris, D., and Naffziger, S. Statistical clock skew modeling with data delay variations. *IEEE Transactions on VLSI Systems*, 9, 6 (Dec. 2001), 888-898.

[85] V. Agarwal, J. Sun, A. Mitev, and J. Wang, "Delay Uncertainty Reduction by Interconnect and Gate Splitting," *Proceedings of the 2007 Asia and South Pacific Design Automated Conference (ASP-DAC 2007)*, 690-695.

[86] I. Chanodia and D. Velenis, “Effects of parameter variations and crosstalk on H-tree clock distribution networks,” *Proceedings of the 48th IEEE Midwest Symposium on Circuits and Systems (MWSCAS 2005)*, 547- 550.

[87] S. A. Bota, J. L. Rossello, C. de Benito, A. Keshavarzi, and J. Segura, “Impact of Thermal Gradients on Clock Skew and Testing,” *IEEE Design & Test of Computers*, 23, 5 (May 2006), 412-424.

[88] M. J. Figueiredo, and R. L. Aguiar, “Noise and Jitter in CMOS Digitally Controlled Delay Lines,” *Proceedings of the 13th IEEE International Conference on Electronics, Circuits & Systems (ICECS 2006)*, 1356-1359.

[89] P. Heydari, "Characterizing the effects of the PLL jitter due to substrate noise in discrete-time delta-sigma modulators," *IEEE Transactions on Circuits and Systems I*, 52, 6 (June 2005), 1073-1085.

[90] Jang Jinwook, O. Franza, and W. Burleson, "Period Jitter Estimation in Global Clock Trees," *Proceedings of the 12th IEEE Workshop on Signal Propagation on Interconnects (SPI 2008),* 1-4.

[91] J. Rosenfeld, and E. G. Friedman, "Design methodology for global resonant H-tree clock distribution networks," *Proceedings of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, 4.

[92] F. Liu, “A General Framework for Spatial Correlation Modeling in VLSI Design,” Proc. of DAC 2007, 817-822.

[93] B. Linares-Barranco, and T. Serrano-Gotarredona, “A Physical Interpretation of the Distance Term in Pelgrom's Mismatch Model results in very Efficient CAD,” *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS 2007)*, 1561-1564.

[94] S. J. Lovett, M. Welten, A. Mathewson, and B. Mason, “Optimizing MOS transistor mismatch,” *IEEE Journal of Solid-State Circuits*, 33, 1 (Jan. 1998), 147-150.

[95] B. Linares-Barranco, and T. Serrano-Gotarredona, “Cheap and easy systematic CMOS transistor mismatch characterization,” *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS 1998)*, 2, 466-469.

[96] Doyle, B., Mahoney, P., Fetzer, E., and Naffziger, S. Clock distribution on a dual-core, multi-threaded Itanium family microprocessor. *Proceedings of the 2005 International Conference on Integrated Circuit Design and Technology (ICICDT 2005)*, 1-6.

[97] Omana, M., Rossi, D., and Metra, C. Low cost scheme for on-line clock skew compensation. *Proceedings of the 2005 IEEE VLSI Test Symposium (VTS 2005)*, 90-95.

[98] Hong-Yean Hsieh, Wentai Liu, M. Clements and P. Franzon, “Self-calibrating clock distribution with scheduled skews,” *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS 1998)*, 470-473.

[99] R. Watn, T. Njolstad, F. Berntsen, and J. F Lonnum, "Independent clocks for peripheral modules in system-on-chip design," *Proceedings of the 2003 IEEE International SOC Conference (SOC 2003)*, 25-28.

[100] S. Sivaswamy, and K. Bazargan, “Statistical Generic and Chip-Specific Skew Assignment for Improving Timing Yield of FPGAs,” *Proceedings of the 2007 International Conference on Field Programmable Logic and Applications (FPL 2007)*. 429-434.

[101] D. Harris and S. Naffziger, “Statistical clock skew modeling with data delay variations,” *IEEE Transactions on VLSI*, 9, 6 (Dec. 2001), 888-898.

[102] R. Darapu, C. W. Zhang and L. Forbes, “Analysis of Jitter in Clock Distrbution Networks,” *Proceedings of the 2004 IEEE Workshop on Microelectronics and Electron Devices (WMED 2004)*, 45-47.

[103] A. Attarha and M. Nourani, “Testing interconnects for noise and skew in gigahertz SoCs”, *Proceedings of the 2001 IEEE International Test Conference (ITC 2001)*, 305-14.

[104] A. Maxim, “A 0.16-2.55-GHz CMOS active clock deskewing PLL using analog phase interpolation,” *IEEE Journal of Solid-State Circuits*, 40, 1 (Jan. 2005), 110-131.

[105] K.-H. Cheng, C.-L. Wu, Y.-L. Lo, and C.-W. Su, “A phase-detect synchronous mirror delay for clock skew-compensation circuits,” *Proceedings of the 2005 IEEE International Symposium on Circuits and Systems (ISCAS 2005)*, vol. 2. 1070-1073.

[106] C.-S. Hwang, W.-C. Chung, C.-Y. Wang, H.-W. Tsao, and S.-I. Liu, “A 2 V clock synchronizer using digital delay-locked loop,” *Proceedings of the 2000 IEEE Asia-Pacific Conference on ASIC (AP-ASIC 2000)*, 91-94.

[107] M. Saint-Laurent, and M. Swaminathan, “A multi-PLL clock distribution architecture for gigascale integration,” *Proceedings of the 2001 IEEE Computer Society Workshop on VLSI (IWVLSI 2001)*, 30-35.

[108] Wikipedia.org, " Functional verification," http://en.wikipedia.org/wiki/Functional_verification.

[109] K. A. Jenkins, K. L. Shepard, and Z. Xu, "On-Chip Circuit for Measuring Period Jitter and Skew of Clock Distribution Networks," *Proceedings of the 2007 IEEE Custom Integrated Circuits Conference (CICC 2007),* 157-160.

[110] V. Gutnik, and A. P. Chandrakasan, “Active GHz clock network using distributed PLLs,” *IEEE Journal of Solid-State Circuits*, 35, 11 (Nov. 2000), 1553-1560.

[111] S.-D. Mai, H.-W. Lune, R.-C. Hsu and C. Su, “An autonomous multiple module clock synchronization methodology for SoC,” *Proceedings of the 2003 International Systems-on-Chip Conference (SOCC 2003)*, 39-42.

[112] Y.-M. Wang, and J.-S. Wang, “A low-power half-delay-line fast skew-compensation circuit,” *IEEE Journal of Solid-State Circuits*, 39, 6 (June 2004), 906-918.

[113] Y. Elboim, A. Kolodny, and R. Ginosar, “A clock-tuning circuit for system-on-chip,” *IEEE Transactions on VLSI*, 11, 4 (Aug. 2003), 616-626.

[114] J.-L. Tsai, L. Zhang, and C. C.-P. Chen, "Statistical timing analysis driven post-silicon-tunable clock-tree synthesis," *Proceedings of the 2005 IEEE/ACM International Conference on Computer Aided Design (ICCAD 2005)*, 575-581.

[115] V. Khandelwal, and A. Srivastava, "Variability-driven formulation for simultaneous gate sizing and post-silicon tunability allocation," *Proceedings of 6th Internationsal Symposium on Parallel and Distributed Computing (ISPD 2007)*, 11-18.

[116] T. J. Yamaguchi, M. Soma, J. P. Nissen, D.E. Halter, R. Raina, and M. Ishida, “Skew measurements in clock distribution circuits using an analytic signal method,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, 7 (July 2004), 997-1009.

[117] M. Nekili, Y. Savaria, and G. Bois, "Design of clock distribution networks in presence of process variations," *Proceedings of the 8th Great Lakes Symposium on VLSI (GLSVLSI 1998),* 95-102.

[118] Park Joonbae, Koo Yido, and Kim Wonchan, "A semi-digital delay locked loop for clock skew minimization," *Proceedings of the 12th International Conference on VLSI Design (VLSI 1999),* 584-588.

[119] J. Jansson, A. Mantyniemi,and J. Kostamovaara, "A delay line based CMOS time digitizer IC with 13 ps single-shot precision," *Proceedings of the 2005 IEEE International Symposium on Circuits and Systems (ISCAS 2005)*, vol. 5, V-4269-4272.

[120] A. H. Atrash, and B. Butka, "A technique to deskew differential PCB traces," *Proceedings of the 2004 IEEE International Symposium on Circuits and Systems (ISCAS 2004)*, 2, II-565-568.

[121] P. Raha, S. Randall, R. Jennings, B. Helmick, A. Amerasekera, and B. Haroun, "A robust digital delay line architecture in a 0.13 /spl mu/m CMOS technology node for reduced design and process sensitivities," *Proceedings of the 3rd IEEE International Symposium on Quality Electronic Design (ISQED 2002)*, 148-153.

[122] Jun Zhou, D. J. Kinniment, C. E. Dike, G. Russell, and A. V. Yakovlev, "On-Chip Measurement of Deep Metastability in Synchronizers," *IEEE Journal of Solid-State Circuits*, 43, 2 (Feb. 2008), 550-557.

[123] M. Renaud, and Y. Savaria, “A linear phase detector for arbitrary clock signals,” *Proceedings of the 2002 IEEE International Symposium on Circuits and Systems (ISCAS 2002)*, IV-775-778.

[124] J. Savoj, and B. Razavi, “A 10-Gb/s CMOS clock and data recovery circuit with a half-rate linear phase detector,” *IEEE Journal of Solid-State Circuits*, 36, 5 (May 2001), 761-768.

[125] S. Soliman, F. Yuan, and K. Raahemifar, “An overview of design techniques for CMOS phase detectors,” *Proceedings of the 2002 IEEE International Symposium on Circuits and Systems (ISCAS 2002)*, 5, V-457-460.

[126] R. B. Watson Jr., and R. B. Iknaian, "Clock buffer chip with multiple target automatic skew compensation," *IEEE Journal of Solid-State Circuits*, 30, 11 (Nov. 1995), 1267-1276.