High-speed structures for dynamically clocked and multi-clock systems

Atanu Chattopadhyay, B. Eng. 2000

Department of Electrical and Computer Engineering

McGill University, Montreal

March 2003

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering.

Abstract

`

With an ever-decreasing minimum feature size, integrated circuits have more transistors, run faster and are larger than ever before. As a result, problems such as heat dissipation, clock generation and clock distribution are at the forefront of challenges facing chip designers today. A Globally Asynchronous, Locally Synchronous (GALS) system combined with dynamic voltage and frequency scaling is an architecture that can combat many of these issues while allowing for high performance operation. In this thesis, we investigate three distinct circuit designs compatible with, but not limited to, such a system. The first uses a novel bi-directional asynchronous FIFO to communicate between independently-clocked synchronous blocks. The second is an All-Digital Dynamic Clock Generator designed to glitchlessly switch between frequencies with very low latency. The third is a Digitally-Controlled Oscillator that can either be used stand-alone or as part of an all-digital PLL (ADPLL) to generate the global fixed frequency clocks required by the All-Digital Dynamic Clock Generator. These designs have been designed, simulated and shown to perform all the tasks required to implement a Globally Asynchronous, Locally Dynamic System (GALDS) in either a traditional ASIC design or a newer System-on-Chip (SoC).

Résumé

Avec la décroissance de la taille minimum des transistors, les circuits intégrés ont plus de transistors, fonctionnent plus rapidement et sont plus grands que jamais. En conséquence, des problèmes tels que la dissipation thermique, la génération et distribution d'horloge sont au premier rang des défis auquel les ingénieurs d'aujourd'hui font face. Un système Globalement Asynchrone et Localement Synchrone (GALS) combiné avec la graduation dynamique de voltage et de fréquence est une architecture qui peut combattre plusieurs de ces défis tout en produisant des circuits de haute performances. Dans cette thèse, nous étudions trois circuits distincts compatibles avec, mais non limité à, un tel système. Le premier emploie un nouveau FIFO asynchrone bi-directionnel pour communiquer entre deux blocs synchronisés qui opèrent avec des horloges indépendantes. Le second est un générateur d'horloge dynamique tout-numérique conçu pour changer rapidement entre les fréquences sans produire de "glitch." Le troisième est un oscillateur controllé numériquement qui peut être utilisé seul ou comme composante dans un circuit à verrouillage de phase (PLL) tout-numérique afin de produire les horloges globales fixes requisent par le générateur d'horloge dynamique tout-numérique. Ces circuits ont été conçues, simulées et montrées capable d'accomplir toutes les tâches exigées pour créer un système globalement asynchrone et localement dynamique (GALDS) dans un ASIC traditionnel ou dans un système sur une puce (SoC) qui devient de plus en plus populaire.

Acknowledgements

I would first like to thank my supervisor Zeljko Zilic. He provided me with the insight and direction I needed, while giving me enough freedom to explore my ideas as far as I wanted. While all the work presented in this thesis is my own, it would not have been possible without his comments and suggestions; they were always well thought out and pertinent. I would especially like to thank him for the financial support he provided me with throughout my Master's and the opportunity he gave me to present my work to others, notably the paper that we co-authored and I presented at ICECS 2002: "High-Speed Structures for Inter-clock Domain Communication."

I would also like to thank the professors at McGill in the Microelectronics and Computer Systems laboratory (MACS) including Prof. Radu Negulescu for introducing me to asynchronous design and Prof. Gordon Roberts for creating an extraordinary environment for learning and research at McGill. I also appreciate the guidance of other graduate students in the department, most notably Ian Brynjolfson who helped me immeasurably during my first year at MACS, as well as some of my supervisor's other students, Stuart McCracken, Marc Boulé and Henry Chan. I would also like to thank Mona Safi-Harb, Sebastian Laberge and Geoffrey Duerden who were always around to answer any of my questions. I must also thank the system administrators in the undergraduate and graduate computer labs including Hugo Levasseur and Ben Mihailescu for always keeping everything up and running.

Next, I would like to thank all my friends from the "activity" group and elsewhere who have always been around when I needed them and were willing accomplices for all the activities that I tried to fit into my life. Finally, but certainly not least, I would like to thank my parents and my sister for being amazing throughout. They gave me whatever I needed, whenever I needed it and thanks to them, I never had to worry about anything but work throughout my entire degree.

Table of Contents

List of Figures

Chapter 1: Introduction

In the past, high-speed operation and minimum silicon area were the two most important criteria in creating an integrated circuit design. Today, with an ever-decreasing minimum feature size, chips have more transistors, run faster and are larger than ever before. Transistors have become both fast and cheap to implement. However, problems such as heat dissipation, clock generation and clock distribution are at the forefront of challenges facing today's hardware engineers and system-level architects. A Globally Asynchronous, Locally Dynamic System (GALDS) architecture can combat many of these issues while allowing for high performance operation. In this thesis, we investigate this design paradigm in detail, looking at the various circuit level components that are required to implement the system. This system is compatible with both traditional ASIC designs and newer Systems-on-Chip (SoCs).

1.1 MOTIVATION

Designing high performance integrated circuits has always been a challenge for hardware designers. Over the years, technology has progressed and computer-aided design (CAD) tools have improved dramatically allowing ever-increasing complexity on a single chip. It is now easier than it has ever been to create sophisticated designs

that could not have even been imagined just a few years ago. Throughout the years, Very Large Scale Integration (VLSI) in integrated circuits (ICs) has become the norm with gate counts in the millions per chip. With the promise of Ultra Large Scale Integration (ULSI) and Wafer Scale Integration (WSI) in the future, there can effectively be hundreds of integrated circuits interconnected at the silicon level working together to perform a common or related task. With increased gate counts and larger die-sizes brought about by new process technologies, today's designers face problems involving clock generation, clock distribution and power consumption. The challenge of creating multi-gigahertz frequencies for an IC is a non-trivial one. The typically used analog components such as PLLs and DLLs simply are not designed to operate at these frequencies. In addition, it is nearly impossible to distribute a single global clock to all the functional blocks on a chip without introducing excessive clock skew to the point where it may actually exceed the length of the clock period. Finally, with transistor counts approaching a billion for a single IC and the tremendously high clock frequencies being used, the power consumption for an IC has reached the point where the package is no longer capable of dissipating the heat required to ensure proper operation [1]. So instead of just accepting these limitations, system-level architects and hardware designers need to find ways of creating future integrated circuit designs that cope with these problems.

1.2 SYSTEM OVERVIEW

Presented in this work is one such architecture. It combines the Globally Asynchronous, Locally Synchronous (GALS) design style with dynamic voltage and frequency scaling to combat the major problems described above: clock generation, clock distribution and power consumption. It resolves many of the circuit issues without adding much complexity to the overall integrated circuit and with very little area overhead. Figure 1.1 shows a system level overview of the proposed architecture. Each distinct shaded block represents a circuit design and has a chapter dedicated to it in this document. All the circuits have been created using Cadence schematic capture and simulated using SpectreS under default process and temperature conditions. The process technology used for all designs is Taiwan

Semiconductor Manufacturing Corporation's (TSMC) CMOS P-well 0.18 μm process.

While GALS was first presented by Chapiro in 1984 [1,2], GALS architectures have only become more popular with the growing popularity of asynchronous design and the present day difficulties faced in designing high performance synchronous systems. A Globally Asynchronous, Locally Dynamic System (GALDS) design style embraces the limitations in the technology and the packaging to function at the highest usable performance level with the slowest frequency possible. With a GALDS design, an integrated circuit is divided into a large number of small digital blocks that are each designed to operate within a range of dynamically changing frequencies. This dynamic frequency scaling allows the incorporation of dynamic voltage scaling that, when used together, can achieve significant power savings for each local block. Since each local block is independently clocked, they each work at a near optimal rate with little power wasted. Traditional methods of performing dynamic frequency scaling [3, 4, 5] are flawed in-so-far as they require long lock-in times to change frequency and consist of complex, difficult to design analog components in the Phase-Locked

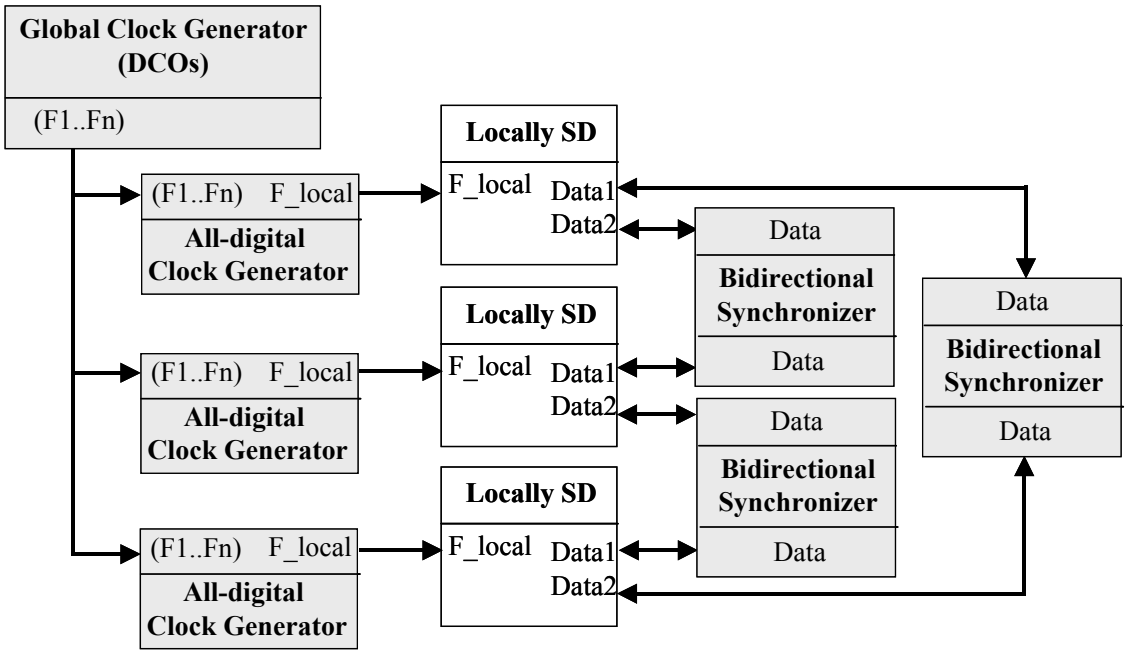

Fig. 1.1: A Globally Asynchronous, Locally Dynamic System (GALDS)

Loops (PLLs) or Delay-Locked Loops (DLLs) that they are made from. While the "locking" aspect of these devices is desirable to eliminate local clock skew, by ensuring that the local blocks in a GALDS system are small and have fine-tuned clock distribution networks, clock skew becomes much less of a concern. In addition, since each block is dynamically clocked based on the actual performance required, knowledge of the exact frequency of operation is not necessary, only whether or not that frequency is high enough or low enough. Thus some deviation in the exact frequencies produced by different instances of the frequency generators due to process variation is not a terminal problem, as long as the clocks that are produced have consistent dynamic characteristics such as rise time, fall time and duty cycle. It is also important that the period of the clocks be constant with as little jitter as possible. Thus in this architecture, clock generation and distribution are much simpler problems to overcome.

To generate each local clock, an all-digital clock generator was developed using a series of clock dividers and a clock selector to select between one of the clock divider outputs. This device is capable of very fast frequency changes, but still requires the use of global, fixed frequency clocks. These global clocks are created using a sophisticated digitally-controlled oscillator (DCO) that could also be used in an All-Digital PLL (ADPLL), which is becoming more and more common today. Once these local clocks are created, the major challenge that must be faced is how to pass data and control information between blocks operating using clocks independent of phase and frequency. An asynchronous strategy is ideal here because it is inherently a low-power approach. In addition, it renders any clock skew between blocks harmless, thus facilitating the clock distribution of the global clocks from the DCOs. This asynchronous inter-clock domain communication strategy uses asynchronous-to-synchronous converters at both ends of either a standard asynchronous FIFO or a novel bi-directional asynchronous FIFO that is capable of transferring data simultaneously in both directions between two blocks over a common data bus.

1.3 OUTLINE

In this document, we will first look at the theory and background behind these circuits and systems in Chapter 2. Then, the Asynchronous Structures for Inter-clock Domain Communication are presented in Chapter 3. The All-Digital Clock Generator assigned to each local block and used to dynamically alter the local frequency is discussed in Chapter 4. Finally, Chapter 5 describes the implementation of a Digitally-Controlled Oscillator that can either be used stand-alone to generate the global frequencies required by the All-Digital Clock Generator or as the oscillator in an ADPLL, replacing the Voltage-Controlled Oscillator (VCO) in a conventional PLL. Each functional block is described in a chapter containing background information, an overview and description of the circuitry used, simulated operating results and some discussion concerning the implementation and performance of the device.

1.4 REFERENCES

[1] Muttersbach, J., T. Villiger, H. Kaeslin, N. Felber and W. Fichtner, "Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of on-chip Systems," ASICSOC, pp. 317-321, 1999.

[2] Chapiro, D. "Globally-Asynchronous Locally-Synchronous Systems," Ph. D. Thesis, Stanford University, October 1996.

[3] Secareanu, R. M., D. Albonesi and E. G. Friedman, "A dynamic reconfigurable clock generator**,"** ASICSOC, pp. 330-333, 2001.

[4] Zhou, J. and H. Chen, "A 1 GHz 1.8 V monolithic CMOS PLL with improved locking," MWSCAS, pp. 458-461, vol. 1, 2001.

[5] Brynjolfson, I., "Dynamic Clock Management Circuits for Low Power Applications," Master's Thesis, McGill University, April 2001.

Chapter 2: Background

This chapter discusses some relevant background concerning items that will be discussed later in this document. While each chapter contains background information specific to each component discussed there in, this chapter provides more general background on traditional methods of clock generation as well as some theory on asynchronous circuits and design principles. Section 2.6 outlines the measurement conventions used throughout this document.

2.1 PLLs AND DLLs

Delay-Locked Loops (DLL) and Phase-Locked Loops (PLL) are the two tried and tested methods of generating clock signals within an integrated circuit. Both structures are feedback systems containing analog elements and can perform clock re-alignment. As shown in Figure 2.1, a DLL outputs a clock signal with the same period as the input reference clock by adjusting the delay of a variable delay line. In a PLL, the phase detector is responsible for measuring the clock skew and the frequency difference between the feedback and reference clocks (a so-called phase-frequency detector: PFD) and creating up/down signals that the charge pump uses to control the voltage-controlled oscillator (VCO). The PLL has many applications besides clock generation including clock recovery from serial data, clock skew

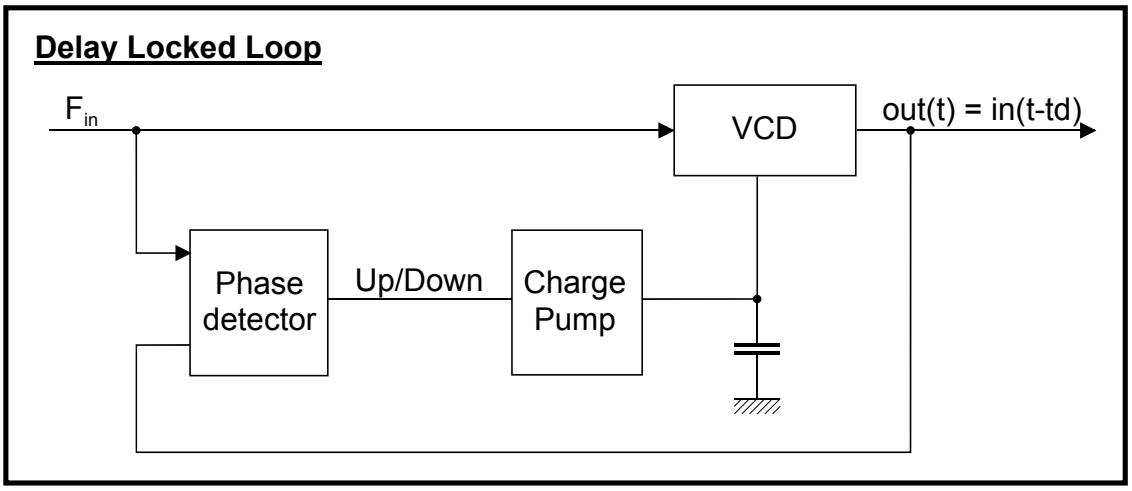

Fig. 2.1: Generic structure of a Delay-Locked Loop (DLL) [2]

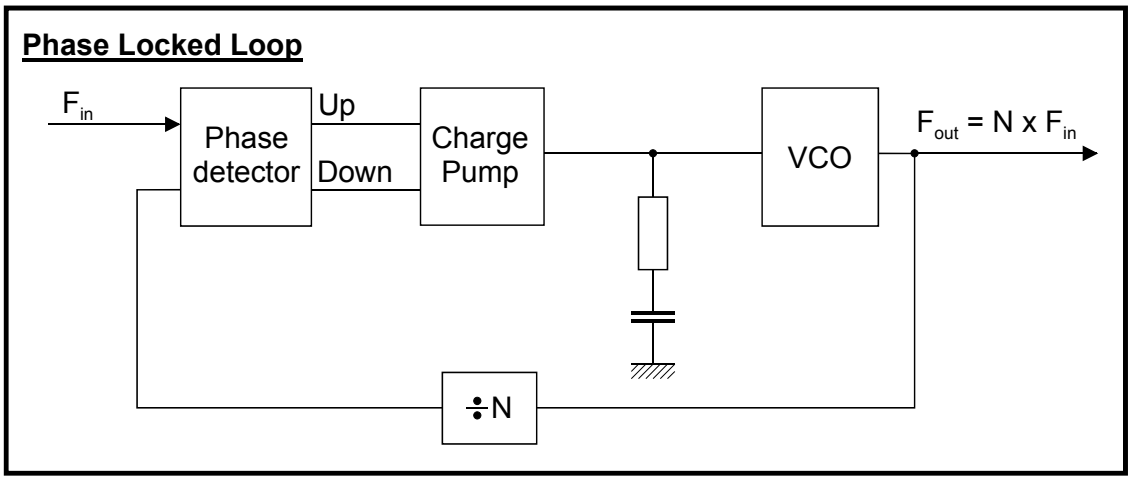

Fig. 2.2: Generic structure of a Phase-Locked Loop (PLL) [2]

reduction, AM/FM demodulation and frequency tuning. A PLL can multiply a reference clock by diving the feedback signal by a factor N as shown in Figure 2.2. If another clock divider (÷M) is placed at the output of the block, an arbitrary frequency output of:

$$F_{out} = F_{in}*N/M$$

can be obtained, where M and N are integers. Since the frequency output range is quite versatile, a PLL is well suited for clock generation. However, the PLL suffers from sensitive analog components that can introduce jitter, often require a long time to lock into a new frequency setting and have limited frequency ranges from which a

specific frequency can be locked into. The PLL is capable of producing an output clock with reduced noise even with a noisy or jittery reference signal. Conversely, a DLL is an inherently stable device and is less complex and easier to design. Even though it is incapable of reducing clock jitter, it only introduces limited jitter and thus it is well suited for skew reduction applications [1].

2.2 DYNAMIC CLOCK MANAGEMENT

The power consumption of an IC is proportional to the frequency, capacitive load, switching activity and the square of the voltage used. If the operating frequency of a chip or a region of a chip is chosen as a function of the demand on that block, the direct relationship between frequency and power consumption can be exploited. If, in addition to this frequency scaling, dynamic voltage scaling is applied to the regions operating at lowered frequencies, even more power savings can be achieved since power consumption is proportional to the square of the operating voltage. Voltage scaling would not be possible with a fixed frequency clock and thus would not be possible without a dynamic clock manager. Having a dynamic clock manager for each local block instead of one for the entire circuit maximizes these power savings. Such a scheme provides the best of both worlds: the possibility of high-speed operation with full voltage swing, and very low power with voltage and frequency scaling. Lowering the power consumption of the overall circuit is done without sacrificing performance since only blocks that are performing non-critical functions are run at reduced speed. The clock generator should be capable of executing rapid frequency changes with low latency, quickly adapting to the ever-changing needs of a local block with as little idle time during frequency changes as possible. Any time the clock is inactive is a performance drain. Any extra time spent in an inappropriate frequency could be wasted power, wasted time or it could even create errors if the current frequency is too high to be supported by the current task. Previously proposed techniques for generating variable frequency dynamic clocks have relied on DLLs or PLLs to perform the frequency changes at the heart of the clock manager. This arrangement requires significant effort to ensure stable clocks over a wide range of frequencies, as well as increased area due to the analog components used in the

design. Further, the lock time associated with these devices can exceed hundreds of clock cycles [3].

2.3 ASYNCHRONOUS DESIGN PRINCIPLES

An asynchronous handshake is the heartbeat of an asynchronous system. Akin to a clock pulse in a synchronous design, an asynchronous handshake consists of a request transition followed by an acknowledge transition. An asynchronous handshake comes in two general flavors: either 2-phase or 4-phase. In a 2-phase design, any request transition followed by an acknowledge transition represents a data cycle with one bit (or packet) of data sent. In a 4-phase design, only specific transitions (either rising or falling for each of the request and acknowledge lines) result in a data transfer. As such, a 4-phase handshake represents a return-to-zero (RTZ) or a return-to-one design. A 2-phase handshake is non-return-to-zero (NRZ). 2-phase designs are generally quicker since the request and acknowledge control lines toggle only once each per data transaction whereas in the 4-phase design, 2 pairs of transitions are required per data item. However, the circuitry required for the 2-phase control cell is more complicated than a 4-phase cell since it must react to and respond with both rising and falling edges. One key benefit of the 4-phase design is that it has a distinct idle state. In a 2-phase design, the cell is always either reading or writing with no distinguishable idle time.

Depending on whether the asynchronous control cell is a push or a pull channel device, it can either be requesting data (pull channel) from an adjacent cell or placing data (push channel) for an adjacent cell. This nomenclature is always with respect to the device that initiates a transaction (the master). Figure 2.3 shows the standard notation for each of these conventions [4]. In both push and pull channel devices, the master cells initiate the transaction by sending a request signal to the slave cell. The slave cells respond with an acknowledge signal when they are ready. In a push channel device, the slave cell reads the data at the appropriate time and the master cell releases the data when the operation is complete. In a pull channel device, the slave writes the data to the data lines at the appropriate time and the master cell is

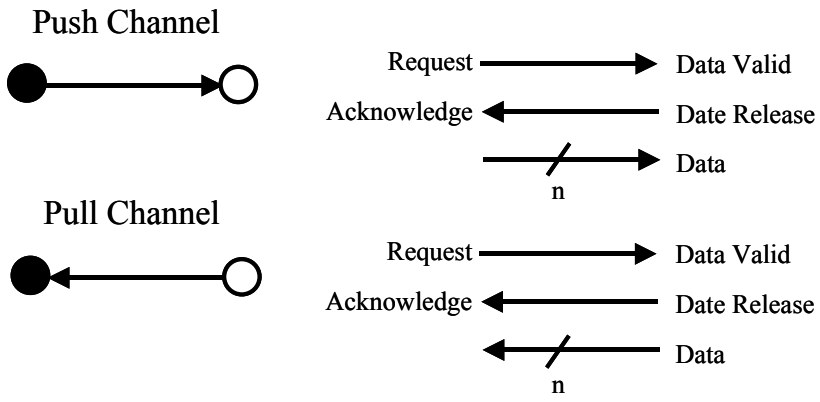

Fig. 2.3: Standard notation to represent push and pull channel devices

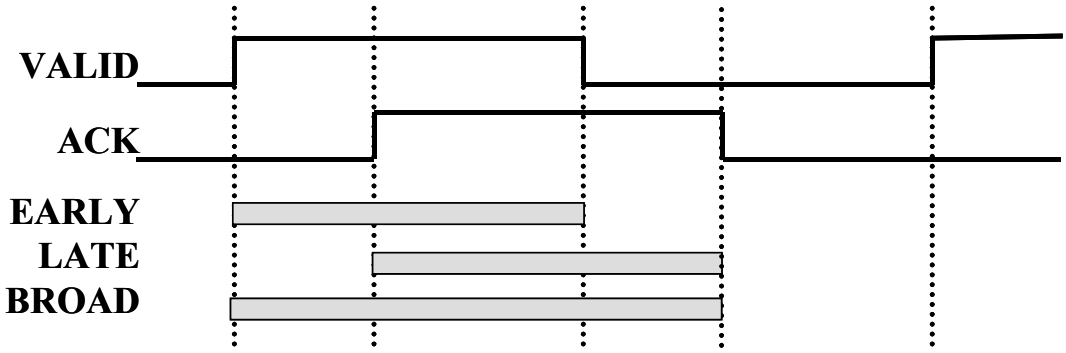

Fig. 2.4: Data validity for asynchronous handshake cells

responsible for reading it at the appropriate time. The slave cell must also release the data when it knows that the operation is complete. The exact moment of reading and writing depends largely on the data scheme used. There are three general classes of data schemes that exist to describe when data is placed on the lines and when it can be read. The differences between them are shown graphically in Figure 2.4. The first, a broad data scheme, describes a handshake where data is present and valid on the data lines from the time the VALID (request) is triggered to time when the ACK (acknowledge) is de-asserted. An early data scheme describes a circuit where data is only guaranteed to be present while the VALID asserted. Finally, a late data scheme describes a circuit where the data is only present on the bus lines while the ACK is asserted.

In discussing asynchronous control cells, it is important to clarify the difference between rails and tracks. The number of rails refers to the number of data lines that

exist between asynchronous cells. The number of tracks refers only to the number of control lines that are present between adjacent cells. Normally, each data bit is sent on a dedicated data line, hence most designs (with 1-bit data) are single rail with dedicated control and data lines. If this design were to have distinct request and acknowledge control lines, it would be a single rail, dual track design. If this design had an n-bit data bus controlled by the request/acknowledge control pair, it would be a bundled data, dual track design. However, some controllers embed the data and request signals together, encoding data based on which request line is asserted. In a dual rail architecture sending one bit of data requires two lines. As such, a request on LINE0 implies that a low value data bit is being sent, whereas a request on LINE1 implies that a high value data bit is being sent. One final categorization of asynchronous circuits is as speed independent (SI) or delay insensitive (DI). Asynchronous speed-independent circuits are only robust to variations in gate delay, but not to variations in wire delays. Delay insensitive circuits are robust for both wire and gate delay variations. It is generally considered overly optimistic to design SI (only) circuits due to the non-negligible nature of present day interconnect time. However, DI circuits are sometimes impractical to build due to the added design effort and additional circuitry that may be required to implement them [5].

One important structure that is used often in asynchronous circuits is the generalized-C element (gC). It is the standard memory element is asynchronous design containing a state conductor to store a bit value and a write enable branches used to modify that value. The gate representation used in this document and the equivalent circuit design is shown in Figure 2.5. This architecture is useful because it allows values to be stored whenever appropriate using the enable branches.

One commonly used asynchronous protocol is GasP. It was developed by Ivan Sutherland of Sun Microsystems based on an earlier scheme by Charles E. Molnar dubbed asP* which stood for “asynchronous symmetric pulse protocol” [6, 7]. It utilizes a single rail datapath, with single-track control. The request is sent as a rising (or falling) transition on the control line and the acknowledge is returned as a resetting of that signal to its original value. Even though a fight occurs on the common control

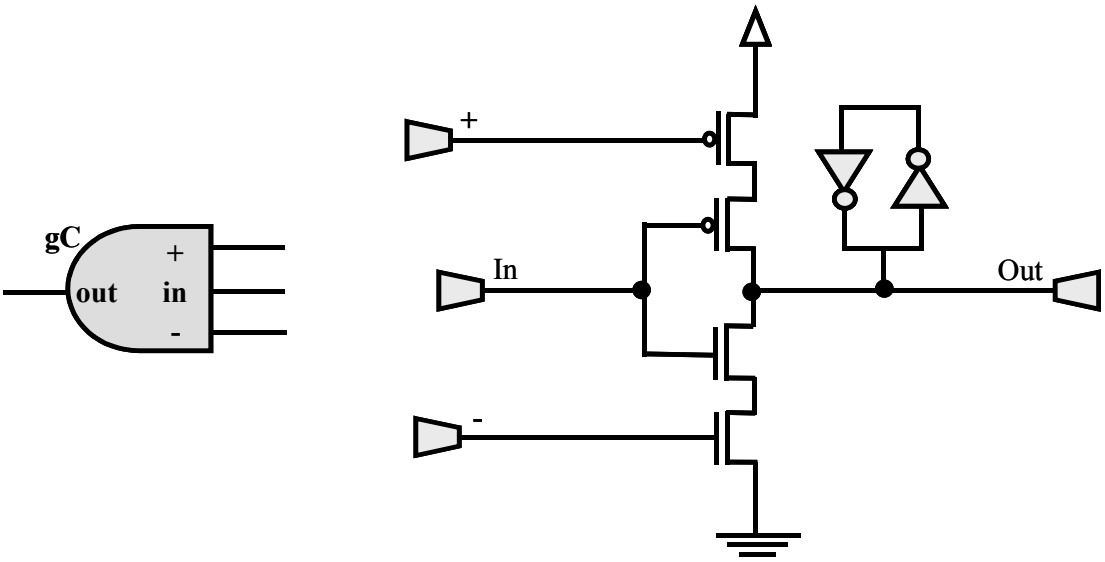

Fig. 2.5: Generalized-C element

line whenever a request or acknowledge is sent, this method results in very fast circuits. A *fight* is a phenomenon that occurs when two transistors try to temporarily and simultaneously drive the same output node with different values. This is normally only allowed in self resetting circuits such as GasP where one of the driving nodes gets disabled as soon as the new value is detected to ensure that the fight occurs for as little time as possible. Using high-speed GasP circuits, a throughput of up to 4.5 giga-items per second has been achieved for an interlocked asynchronous pipeline [8]. However, like 2-phase control circuits, there is no distinct idle state that exists in the system.

2.4 GALS

While transistors switch faster and integrated circuits are now constructed with larger die sizes than ever before, interconnect performance has not improved at the same rate. Copper interconnect may be a significant breakthrough in this domain, but distributing a single clock over a large chip is still nearly impossible without creating clock skew that can easily approach or even exceed the period of the clock. Power consumption is another key problem in the current generation of integrated circuits. Large synchronous designs with speeds in excess of 2 GHz pose a heavy burden on

power distribution networks and can lead to dynamic faults that are very difficult to test for. Today's designs have begun to incorporate multiple clock domains to combat both of these issues. This also limits the maximum local frequency to the longest critical path in the region, and not to the longest critical path in the entire circuit as is the case with single clock domain designs.

Globally Asynchronous, Locally Synchronous (GALS) systems provide an ideal solution for many of these issues [9]. Synchronous blocks, each operating with a different local clock, are interconnected using an asynchronous framework. By creating links to communicate between local blocks that work independently of phase and frequency, clock distribution becomes less of a concern since clock skew between blocks is rendered harmless. In fact, random or specially tuned clock skew reduces the number of simultaneously switching transistors and thus can be a desirable trait by flattening the power spectrum and minimizing the instantaneous current consumption of the circuit. This skew has the added benefit of reducing electro-magnetic emissions (EMI) [10]. Presented throughout this document are components of a design for, but not limited to, use in a GALS system.

2.5 INTERCONNECT

Smaller feature sizes have created chips with more functional blocks and thus the requirement for more interconnect hardware. Today's processes use narrower, taller wires that have higher capacitances per unit length and are thus slower with higher time constants (RC). They also suffer from higher coupling, which can result in unwanted cross-talk between wires. There are a variety of steps that can be used to improve interconnect performance. Repeaters can be used to divide one long wire into many smaller ones. This can decrease the overall latency of an interconnect line since there is an exponential increase in propagation time versus wire length. The use of repeaters acting as pipelining stages has the added affect of increasing the overall throughput of the interconnect. Breaking a wide datapath into many smaller ones (for example, splitting a 32-bit data bus into eight 4-bit data busses) each with independent control circuitry can speed up the design by lessening the load on the control lines,

eliminating some of the cross-talk and decreasing the instantaneous power consumption by introducing a small amount of skew between the exact times that each bundle of data switches on the data lines. Asynchronous circuits are ideal for interconnect circuitry since they are inherently low power due to the random nature of each transition and since asynchronous structures are unaffected by the clock skew that would create problems for synchronous circuitry when transporting data across a chip. One additional benefit of asynchronous interconnect is that it allows you to take advantage of the average-case delay between blocks and not the worst-case. This is to say, the circuitry only uses the amount of time required to perform the transaction and then the bus is released. In the synchronous case, the clock cycle has to be set for the worst-case and every operation requires the worst-case time to be executed [11].

2.6 DYNAMIC SIGNAL CHARACTERISTICS

A standardized set of guidelines is required to compare and evaluate the signal quality and the dynamic characteristics of clock signals. Figures 2.5 and 2.6 show the measurement conventions used for evaluating the performance of a clock signal throughout this document. The *period* of a sinusoidal signal can be defined in a number of ways, but the most generic definition is the smallest time between any two points that share the same value and derivatives. Even though the period of a signal can be measured between any two identical points, it only represents the time between identical 50% points in this document. So in TSMC's 1.8 V 0.18 μm process, this implies that period measurements are taken between 900 mV samples of a signal with identical slopes (ideally). The *high time* and *low time* of the signal are respectively defined as the contiguous time that the signal is above its 50% threshold, and the contiguous time that the signal is below its 50% threshold. The *rise time* of a signal is defined as the amount of time taken for a signal to go from 10% to 90% of its peak value. Conversely, *fall time* is the time taken for a signal to go from 90% to 10% of its peak value. With the 1.8 V being used for VDD in all circuits, this creates a 180 mV 10% voltage level and a 1.62 V 90% voltage level. The simulations discussed throughout this document assume 50 ps rise and fall times for the inputs to any circuit under test.

Figure 2.7 defines the conventions used for the key dynamic characteristics of a latch. The *setup time* is the amount of time that a signal is stable before an active clock edge and the *hold time* is the amount of time that is a signal is stable after an active clock edge. Only the minimum setup and the minimum hold times are relevant in determining the performance of a latch. An active clock edge is an edge that can initiate a modification to the value stored within the latch. In other words, the active edge initiates a *transparent time* for the latch. Conversely, when the latch is not sensitized to the value at its input, it is said to be *opaque*. The *propagation delay* is defined as the time between 50% points of an input and an output transition. In the case of a latch, the propagation delay is measured between an input clock transition and an output signal transition. As such, it is also known at the clock-to-output time. There are two propagation delay times that are relevant for conventional latches and flip flops, a high-to-low delay and a low-to-high delay. Since every clock transition generates a transparent time with double-edged latches, there are two transparent times per clock period of a double-edged latch. Thus, there are 4 distinct propagation delays possible with a double-edged latch since each rising edge and each falling edge can result in either a high-to-low or a low-to-high transition. For a latch to perform well, it needs to have all its propagation delays well matched. In addition, the rise and fall times of data exiting the latch should be nearly identical. The latches shown all use triangles at their clock inputs to show that they are clocked devices even though this goes against the standard convention of only being used with edge-triggered devices. However, since the level-sensitive latches are only transparent for very short periods of time (typically 125 ps), the terms “level sensitive” and “edge-triggered” will both be used when describing the operation of these latches.

Clock skew is an important consideration in dealing with clocks. It measures the amount of delay that may exist in the interconnect or the processing circuitry in a clock path. It is measured as the time between matching points between different clock signals. In this document, clock skew is measured by comparing 50% values (900 mV) that are either both rising or falling. *Clock jitter* is the amount of deviation that exists between the periods, the high times or the low times of a clock. There are two variants of clock jitter: long term jitter and cycle-to-cycle jitter. The former

represents the difference between the absolute maximum and minimum values of high time, low time or clock period that can be reached by a given circuit running continuously over a very long period of time. The latter is the maximum deviation that can occur in clock period, high time or low time between any two consecutive clock cycles. Clock jitter hurts the performance of a circuit since the delay between sequential elements in a design cannot exceed the minimum possible clock period. Thus, the circuit has to be run at a slightly slower than optimal frequency to account for the time deviations that may occur in the clock due to jitter. Jitter may also cause transitions to occur at marginally different times with a circuit in a slightly different electrical state thereby altering the dynamic properties of identical transitions depending on when they occur. This can lead to inconsistent operating behaviour from cycle to cycle. For these reasons, jitter should be kept as small as possible within any clock generation and clock distribution circuitry.

Additionally, most of the sequential circuits have been designed to avoid race conditions. A *signal race* is when an active edge at a latch or a flip flop creates a transition that generates another transition at the output of another (feed-forward) or the previous (feed-back) sequential element farther down the data path at the SAME active edge.

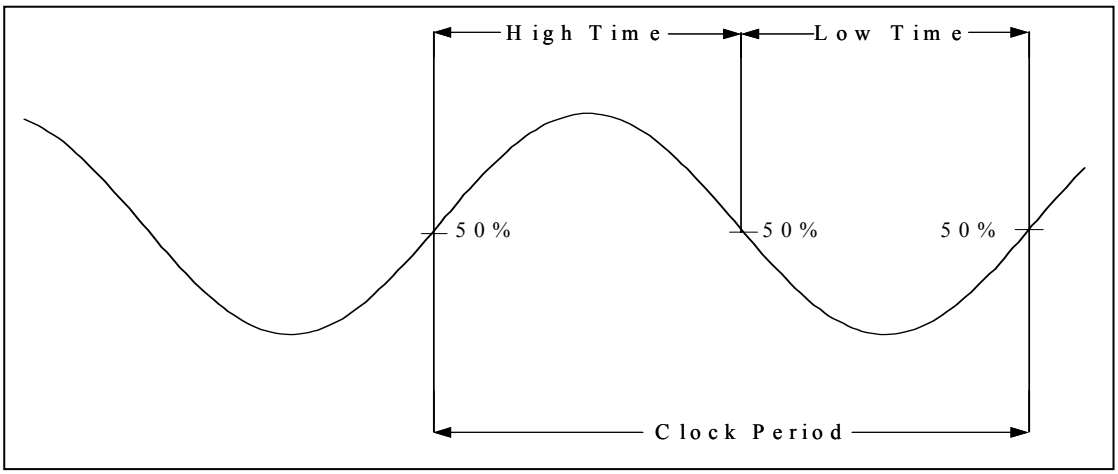

Fig. 2.6: High time, Low time and Period of a signal

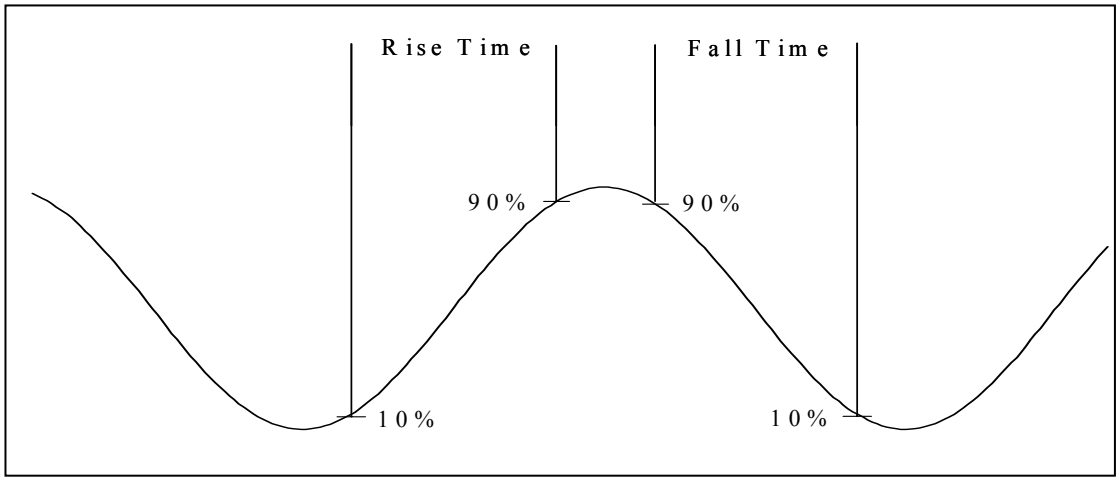

Fig. 2.7: Rise time and Fall time

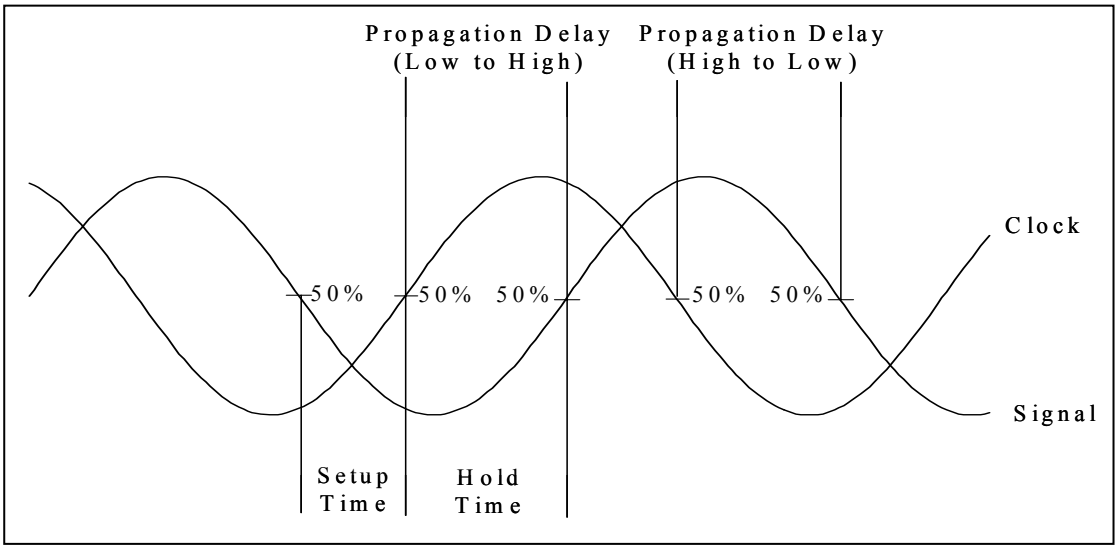

Fig. 2.8: Propagation delay of a signal relative to a clock reference. "Signal" is considered as an INPUT for Setup and Hold Times, and as an OUTPUT for Propagation Delays.

2.7 REFERENCES

[1] Brynjolfson, I., "Dynamic Clock Management Circuits for Low Power Applications," Master's Thesis, McGill University, April 2001.

[2] Moreira, P. "CMOS Sequential Circuits," (Data File), retrieved November 9, 2002, from http://paulo.moreira.free.fr/microelectronics/trieste/sequentialCircuits.pdf.

[3] Brynjolfson, I. and Z. Zilic,"Dynamic Clock Management for Low Power Applications in FPGAs," CICC, pp. 139-142, 2000.

[4] W. Zhu et al., "A Single-Rail Handshake CDMA Correlator," ICECS, pp. 505-508, 2002.

[5] Saito, Hiroshi, A. Kondratyev, J. Cortadella, L. Lavagno and A. Yakovlev, "What is the cost of delay insensitivity?" ICCAD, pp. 316-323, 1999.

[6] Sutherland, Ivan E., "Micropipelines. The Turing Award Lecture," Communications of the ACM, pp. 720-738, vol. 32, no. 6, June 1989.

[7] Sutherland, I. E. and J. Ebergen, "Computers without Clocks," Scientific American, July 15, 2002.

[8] Schuster, S. and P. Cook, "Asynchronous Interlocked Pipelined CMOS Circuits Operating at 3.3-4.5 GHz," ISSCC, pp. 292-293, 2000.

[9] Muttersbach, J., T. Villiger, H. Kaeslin, N. Felber and W. Fichtner, "Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of on-chip Systems," ASICSOC, pp. 317-321, 1999.

[10] Bluno, L, F. Gregoretti, C. Passerone, D. Peretto and L. M. Reyneri, "Designing Low Electro Magnetic Emissions Circuits through Clock Skew Optimization," ICECS, pp. 417-420, 2002.

[11] Bainbridge, W.J. and S. B. Furber, "Delay Insensitive System-on-Chip Interconnect using 1-of-4 Data Encoding," ASYNC, pp 118-126, 2001.

Chapter 3: Asynchronous Structures for Inter-clock Domain Communication

This chapter describes a Globally Asynchronous, Locally Dynamic System (GALDS) design paradigm. In a GALDS design, many synchronous blocks are interconnected using dedicated asynchronous links. Each synchronous block is associated with a local clock generator and features dynamic voltage and frequency scaling to utilize the least possible power for the required performance to be achieved. Two different asynchronous structures are explored in this chapter. They both feature high throughput, modular design and high tolerance to metastability errors that can occur when communicating between clock domains. These structures utilize a 4-phase dual-track asynchronous control circuit to control either a single direction FIFO with data traveling uniquely in one direction or a bi-directional FIFO that is capable of transmitting data simultaneously in two directions by precisely controlling when data has access to a shared datapath.

3.1 BACKGROUND

Designing integrated circuits for deep sub-micron processes is an ever-changing art that poses new challenges for designers with each passing generation of IC.

Transistors switch faster and larger die sizes allow more logic to be placed on a chip than ever before. However, interconnect performance has not improved at the same rate as transistors. While copper interconnect may be a significant breakthrough in this domain, distributing a single clock over a large chip is nearly impossible since the skew of a clock edge can very easily approach or even exceed the period of the clock. Power consumption is another key problem in the current generation of integrated circuits. Large synchronous designs with speeds in excess of 2 GHz pose a heavy burden on power distribution networks and can lead to dynamic faults that are very difficult to test for. Today's designs have begun to incorporate multiple clock domains to combat both of these issues. An added benefit of this approach is that the switching activity is significantly spread out due to the random skew of the multiple clocks over a large chip. This also permits the maximum local frequency to reflect the longest critical path in the region, not the longest critical path in the entire circuit, as is the case with single clock domain designs.

Globally Asynchronous, Locally Synchronous (GALS) systems seem to provide an ideal solution [1]. Synchronous blocks, each operating with a different local clock, are connected using an asynchronous framework. The presence of an asynchronous handshake at the boundary of each local block greatly simplifies the synchronization of data between the multiple clock domains. While GALS has been presented in the past, the proposed solution incorporates dynamic clock management into the locally synchronous blocks to allow for significant power savings. The two fundamental principals behind a Globally Asynchronous, Locally Dynamic System (GALDS) are the ability to control the frequency of the local blocks independently and the ability to robustly communicate between local blocks running at a wide range of varying frequencies. Methods of dynamically changing frequencies for clock management applications are becoming more and more popular as a way of reducing power consumption for circuits where performance requirements vary over time [2]. One possibility is the Programmable Clock Manager (PCM) since it is capable of producing a continuous range of frequencies using PLLs and clock dividers, but it suffers from long lock-in times like any device that uses a PLL [3]. Another possibility is the all-digital clock generation circuitry to be discussed in Chapters 4

and 5. In addition to the clock management infrastructure to slow local clocks in times of lower demand, a GALDS design requires a set of structures designed to interface the blocks. At the heart of our interface design is a novel bi-directional asynchronous FIFO buffer using a 4-phase asynchronous control cell that is capable of transferring data between two local blocks at over 1.5 giga-items per second in both directions simultaneously using a shared common data bus. One potential solution is shown in Figure 3.1 using global fixed frequency clock generators, local dynamic frequency generators and asynchronous interface circuitry to communicate between independent clock domains.

The problem of robustly communication between independently clocked local blocks is addressed using two distinct circuit structures created from a common family of building blocks. The first block is a unidirectional synchronizer that performs the clock domain conversion through an intermediate asynchronous stage. The structure is well suited to transferring sporadic data quickly between adjacent clock domains when only one asynchronous FIFO cell is used. The second structure is a bi-directional synchronizer that is capable of transferring data in both directions between two blocks over a common data bus. As a result, only half the data lines are required between asynchronous FIFO cells to simultaneously transfer data between two blocks.

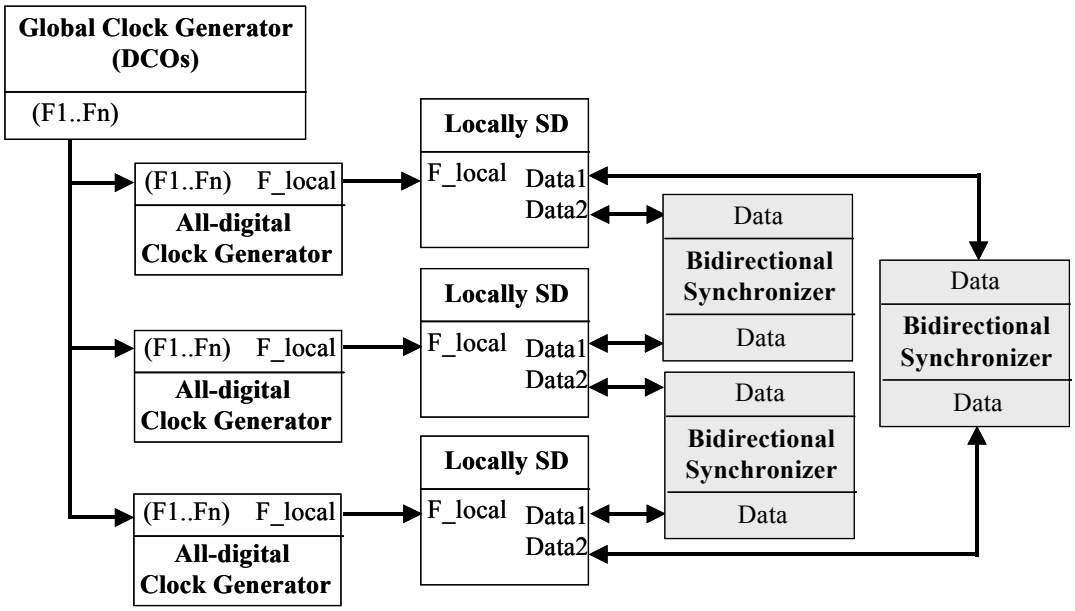

Fig. 3.1: Bi-directional Synchronizers in a GALDS system

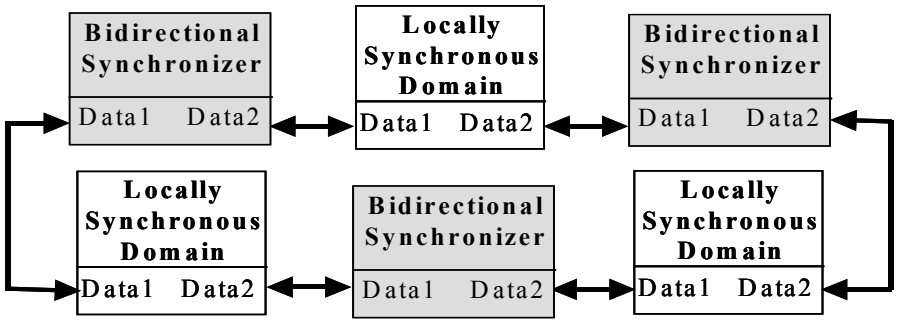

Fig. 3.2: System with three independent blocks interconnected with bi-directional synchronizers

Figure 3.2 shows how independent blocks can be interconnected using bi-directional synchronizers. Some other solutions that incorporate bi-directional data flow such as the one-two-one 6-phase FIFO described in [4], require that data movement occur in strictly alternating directions. The solution described here imposes no such restriction, making it suitable for on-chip communication where the exact direction and throughput requirements cannot be determined in advance. The asynchronous intermediate step makes this solution ideal for dynamic clock management since the exact relation between the clocks used in the locally dynamic/synchronous domains does not need to be known in advance. One additional benefit of this architecture is that each buffer cell acts as a repeater distributing the line impedance, which lowers latency and increases throughput by having multiple data items in transit simultaneously [5].

3.2 BUILDING BLOCKS

3.2.1 4-Phase Asynchronous Handshake Controller

The main reason for choosing a 4-phase scheme for the controller is to ensure that the control lines are encoded as return-to-zero (RTZ). Many higher speed designs ([6], [7]) utilize 2-phase non-return-to-zero (NRZ) control to minimize the number of transitions required per data item. However, this scheme is not appropriate for a bi-directional FIFO since the end of a transaction must be precisely known to allow for data in one direction to be removed in favor of data traveling in the opposite direction. The 4-phase asynchronous handshake cell developed for this application is shown in Figure 3.3. A transition on the VALID (request) input causes the data to be latched

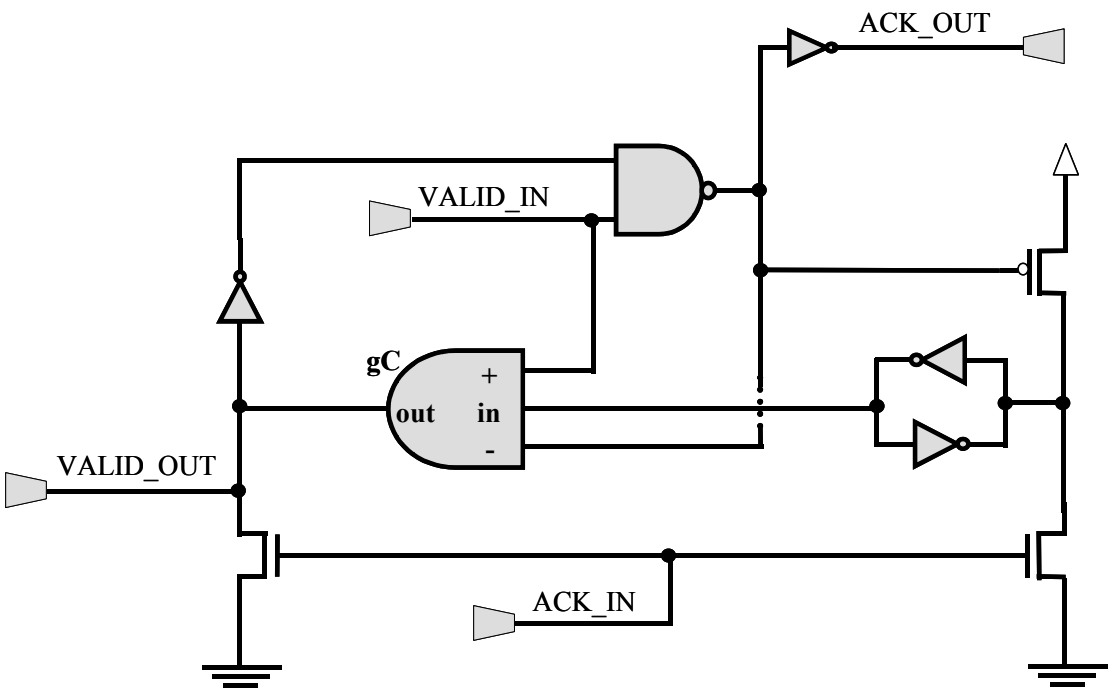

Fig. 3.3: A 4-phase asynchronous control cell. The reset circuitry has been removed for clarity.

into the current cell and generates an output request when the data is ready to be transferred to a subsequent cell. A cell should not be allowed to send and receive data simultaneously since this would corrupt data in the FIFO memory cell. The control cell should also de-assert VALID_out once the ACK_in is asserted since it is then legal for the data on the data lines to be replaced by adjacent control cells. In this way, a state where ACK_in and VALID_out are both simultaneously asserted is merely a momentary transient using this controller. While the correct sequence of state transitions can be reached using simpler control cell, this design is nearly completely delay insensitive since each transition can only occur when triggered by a specific transition in a previous or subsequent cell. This property is essential in ensuring data integrity with potentially chaotic bidirectional data flow. Figure 3.4 shows the State Transition Graph (STG) for the asynchronous handshake cell used in these circuits. These transitions are used to control traffic on the data lines in the bi-directional synchronizer, ensuring that data can freely commute in both directions over the common line. The control cell does suffer from transient shorts in various areas of the circuit, but using these "fights" greatly increases overall performance by

allowing a control signal to start a transition in a cell before the control signal has had a chance to fully propagate through any intermediate logic that may exist in the system [8]. For example, in Figure 3.4, the ACK_in line directly resets the ACK_out signal before it can propagate through the state conductor and the generalized-C element in the control cell. Should this circuitry create excessive noise on the power supply lines, the gC element can be modified to shorten the duration of the transient short by incorporating the ACK_in signal into its control.

The control cell developed here is a "push channel" device, meaning that the master control cell initiates a transaction when it wants to send data to a subsequent cell. Conversely, a "pull channel" device initiates a transaction when it wants to receive data. Figure 3.5 shows the standard notation used to differentiate between these conventions [9]. Besides knowing which device is initiating a transaction, the data scheme used is an important concept in describing what kind of asynchronous circuit is being used. There are three general classes of data schemes that exist to define when data is placed on the lines and when it can be read. The differences between

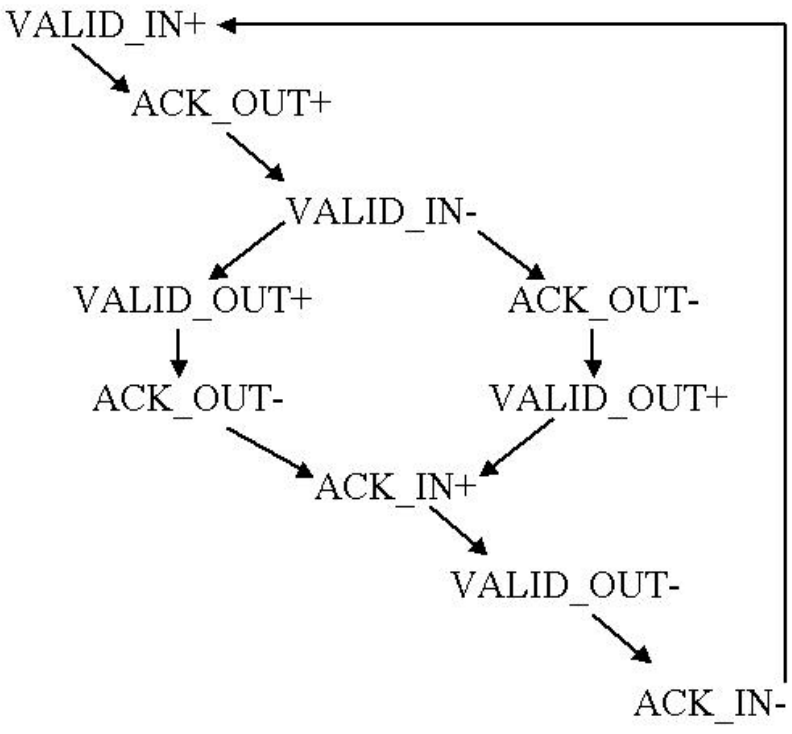

Fig. 3.4. State Transition Graph for the 4-phase control cell used in both the unidirectional and bi-directional FIFOs

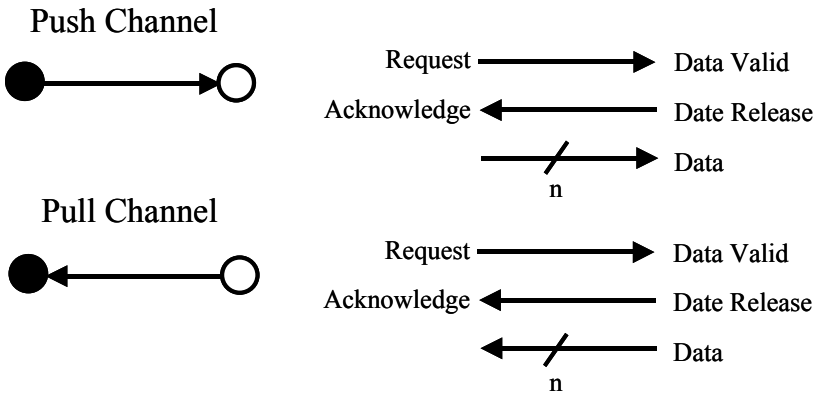

Fig. 3.5: Standard notation to represent push and pull channel devices

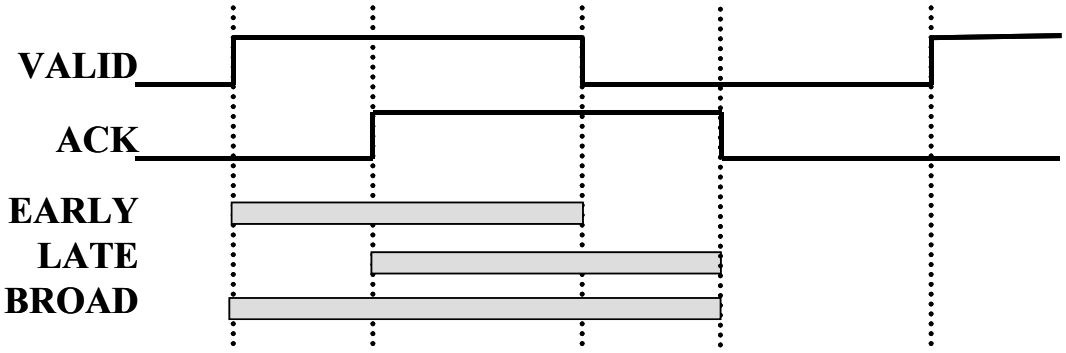

Fig. 3.6: Data validity for asynchronous handshake cells

them are shown graphically in Figure 3.6. The first, a broad data scheme, describes a handshake where data is present and valid on the data lines from the time the request is triggered to time when the acknowledge is de-asserted. An early data scheme describes a circuit where data is only guaranteed to be present while the request is asserted. Finally, a late data scheme describes a circuit where the data is only present on the bus lines while the acknowledge is asserted. As will be shown, a broad data scheme is used for the unidirectional devices, and an early data scheme is used for the bi-directional ones.

3.2.2 Async-Sync Converter

Since our solution uses an asynchronous intermediate stage between clock domains, a synchronizer is required to translate the asynchronous control signals into synchronous ones for the local clock domain. The synchronizer shown in Figure 3.7 operates in three distinct phases as can be seen in the timing waveform of Figure 3.8. The system needs two precisely skewed clocks to operate correctly. In the transparent phase when both clocks are high, the data is allowed to pass through the generalized-C element to the input of the latch. A finite "dead-time" exists while the leading clock is low allowing transients to settle before the register latches the data. Finally, at the rising edge, when the leading clock rises and the lagging clock is still low, the signal is latched into the local clock domain. While this method may increase latency, it nearly guarantees robust communication. It can only fail when an input change occurs at very end of the transparent phase and does not have enough time to settle at X before the next rising edge. These metastability simulations were performed by

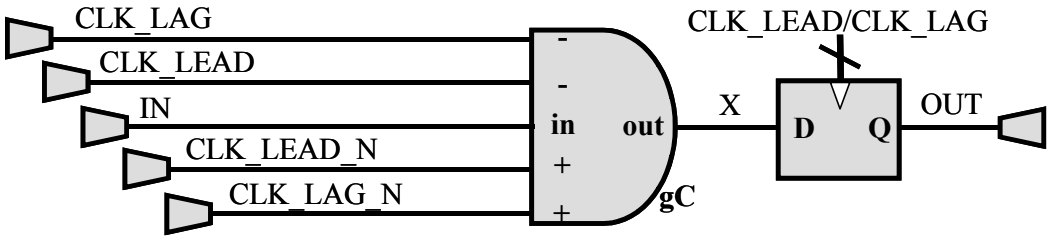

Fig. 3.7: The ASYNC-SYNC converter used to latch asynchronous control signals into a local clock domain

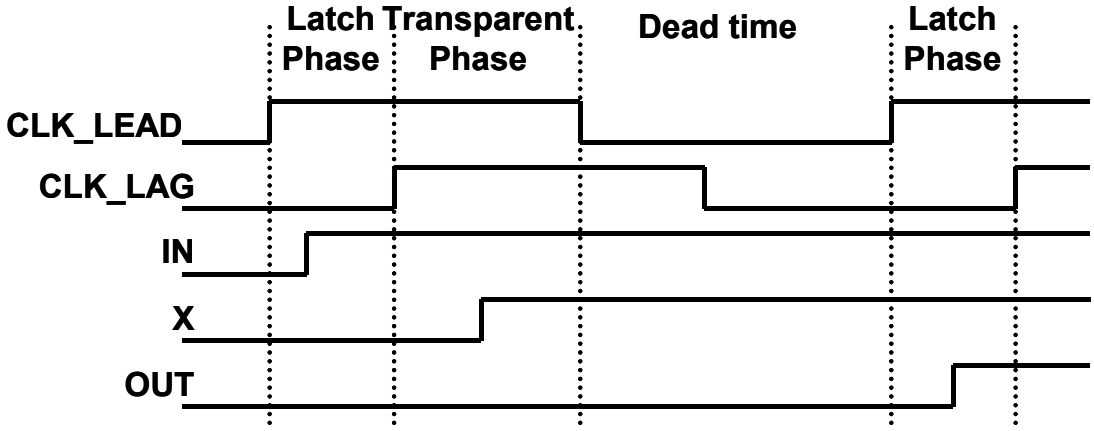

Fig. 3.8: Waveform of ASYNC-SYNC converter latching asynchronous control signals into a local clock domain

toggling the data input of the latch at varying intervals from an active clock edge (near the end of a transparent phase). While no input could be found to cause the circuitry to fail, in real world conditions, it is impossible to fully eliminate metastability. Since the effectiveness of independent clock domain switches can never be 100% guaranteed, a metastability failure should still be observable on an IC. Only a quantitative analysis on an implemented version of this protocol could truly determine the effectiveness of our inter-clock domain communication blocks. Should additional tolerance to metastability be required, additional flip-flops can be placed in the signal path, reducing the probability of a metastability error, albeit with increased latency. Even if a metastability error occurs in the control path, it should not affect the data regardless of whether the failure occurs on a read or a write. This is true because in the write (write-to-FIFO) path, the synchronous block writes the request and data items simultaneously and thus the amount of time required to synchronize the acknowledge and be processed by the synchronous block is inconsequential because at this point, the data has already been safely latched into the asynchronous domain. In the read (read-from-FIFO) path, the data is ready in advance of the request signal due to the request processing through the synchronizer in Figure 3.7. Thus if the request signal satisfies the setup time requirements for the latch, the data will too. The system's performance is limited by the frequency of the local clock domains because they hold their control lines (ACK, VALID) for a full clock cycle and the FIFO cannot proceed until they return to their idle states. While employing self-resetting control lines could eliminate this limitation, doing so would hurt the delay insensitivity of the circuit. One benefit of this design is that there is no possibility of the FIFO overflowing since the asynchronous handshake will not complete until space is available for new data to enter.

While this interface method has less latency than using multiple latches to synchronize data (like double buffering), it is not as robust as some other methods such as plausible locking [10] or asynchronous wrapping [11]. Pausible clocking involves pausing or stretching clock pulses to ensure that data and control signals have sufficient setup time at a synchronous/asynchronous boundary. Asynchronous wrapping involves making the external interface of a synchronous block completely

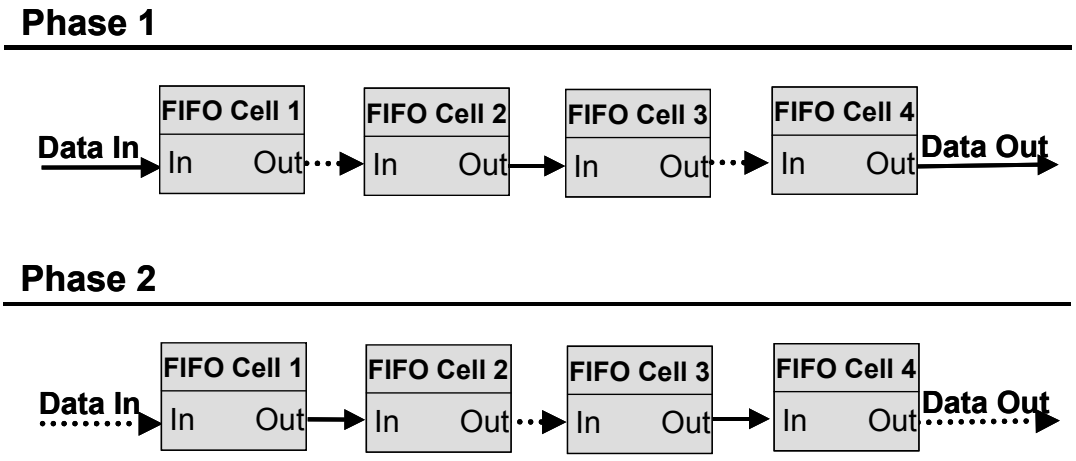

Fig. 3.9: Structure of a traditional asynchronous FIFO where optimal conditions occur when FIFO is half full with data present in every alternate cell. Dashed lines represent unused bandwidth.

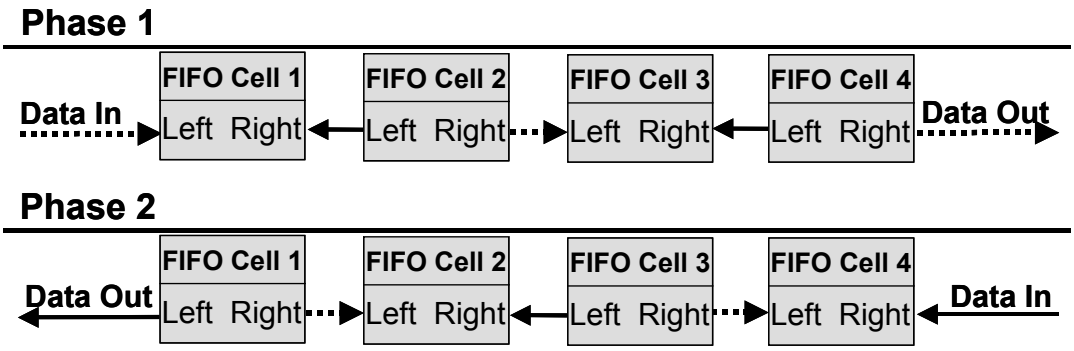

Fig. 3.10: Structure of a bi-directional FIFO utilizing all available bandwidth in the optimal case. Dashed lines represent data propagating from Left to Right.

asynchronous by utilizing a scheme like plausible clocking. While these methods also benefit from lower latency than the traditional double (or more) buffering, they are significantly more complex than our interface method.

3.3 SYNCHRONIZERS

3.3.1 Asynchronous FIFOs

The structure of a traditional unidirectional asynchronous FIFO is shown in Figure 3.9. Like all asynchronous FIFOs, the optimal condition (in terms of maximum throughput) occurs when the FIFO is exactly half full with data present in every alternate cell. In this way, every data element is in movement during every

asynchronous handshake. Once data is present in adjacent cells, data must be removed from the target cell before new data can enter from the source cell and thus there is a decrease in throughput. Thus in the optimal case, the design works in two distinct steps. The first step has all the odd cells receiving data and all the even cells sending data. In the second step, this pattern is reversed. The real problem with this system is that half of all the data lines are unused at any given time. To make better use of this leftover bandwidth, it is possible to interleave data flowing in the opposite direction, thus creating a significant resource sharing advantage in a so-called bi-directional FIFO structure versus a traditional asynchronous FIFO. As shown in Figure 3.10, every cell is either performing two reads or two writes at the same time, thus utilizing every data path present between blocks, in effect doubling the bandwidth of the system by using only slightly more complicated control circuitry. It is possible to create a bi-directional FIFO with a single controller, but this would require that data be guaranteed to be present and valid in both directions for every handshake. A more realistic approach requires two dedicated controllers in each FIFO cell controlling data flowing in either direction. This allows data to stall in one direction while continuing to propagate unaffected in the other direction.

3.3.2 Unidirectional Synchronizer

Inter-clock domain communication in a single direction can be performed using a unidirectional FIFO and asynchronous-to-synchronous (SYNC) converters as shown in Figure 3.11. In addition to the 4-phase control cell, a unidirectional FIFO implementation requires a latch to store data within each FIFO cell. A broad data scheme is used because it allows simple latches without read enables to be used, as shown in Figure 3.12. The distributed control creates a modular design that can easily be expanded to fit many different applications and architectures. Pairing a set of independent unidirectional FIFOs creates a structure that can be useful for bi-directional communication. This solution using a single FIFO stage between adjacent clock domains is ideal when buffering is not needed since the data latency is kept to a minimum. In this case, two independent paths and control blocks are used because

the resource sharing that is achieved with a bi-directional FIFO is not possible with only a single FIFO cell.

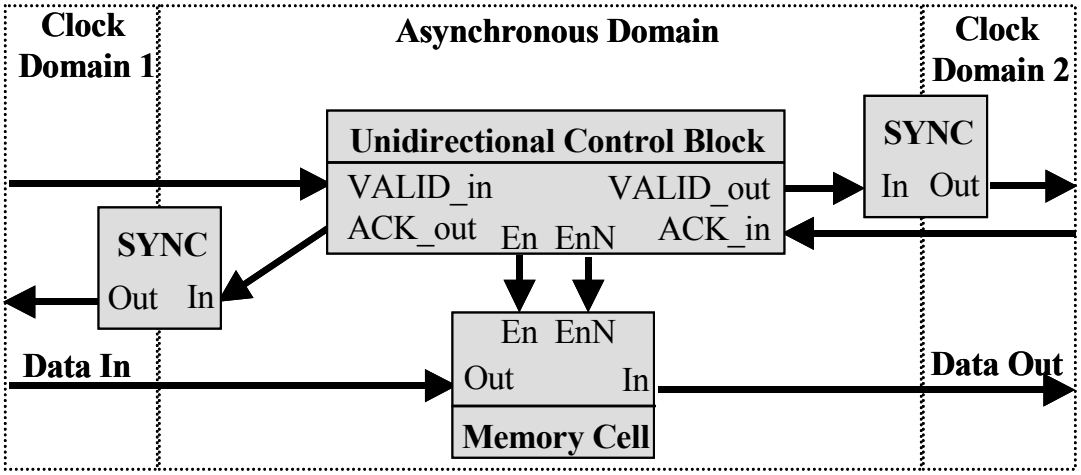

Fig. 3.11: Unidirectional Synchronizer to transfer data between clock domains where transport delay is not an issue and buffering is not required

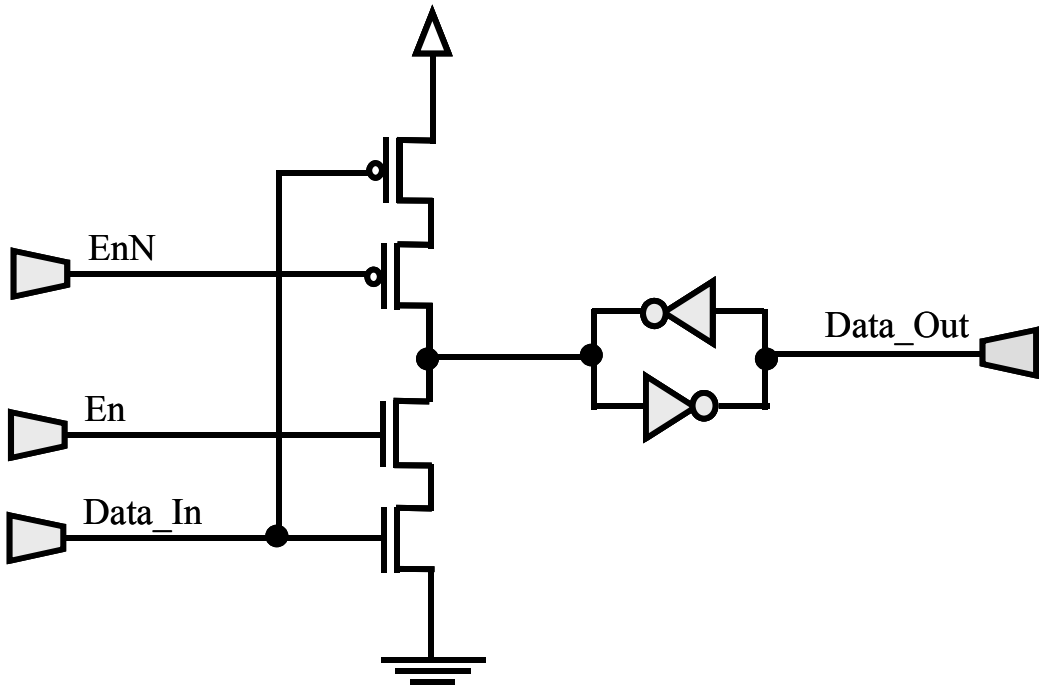

Fig. 3.12. 8-transistor data latch used by the unidirectional FIFO. With write enable transistors at the input, it is only valid for use with a broad data scheme.

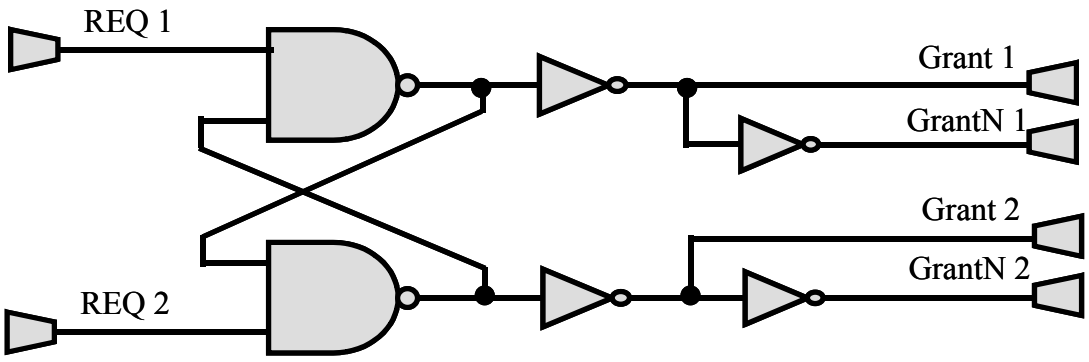

Fig. 3.13. The mutual exclusion element uses cross-coupled NAND gates to ensure that adjacent cells do not try to write to the same line simultaneously

3.3.3 Bi-directional Synchronizer

A bi-directional synchronizer is especially useful when there are long distances between blocks because it maximizes the resource sharing of a bi-directional datapath. To create the control cell for this synchronizer, two identical 4-phase control cells are combined. One cell is used to control the data flow in the A-to-B direction, and the other to control the data flow in the B-to-A direction with additional control placed between the cells for coordinating access to the common data path. The bi-directional FIFO is capable of working in a single direction (either left to right, or right to left), simultaneously in both directions and in any combination in between. If one direction becomes saturated, the other direction can still function normally. It is important that adjacent cells do not try to simultaneously write to the same data line thereby corrupting data. To prevent this from happening, the mutual exclusion element (mutex) shown in Figure 3.13 is included in the system. The inclusion of the mutex element and the nearly delay insensitive nature of the 4-phase handshake controller guarantees that the FIFO will not deadlock as long as the control signals within a cell propagate faster than the inter-cell signals, which is a fairly easy constraint to guarantee due to the longer distances external signals have to travel. The inclusion of the mutex element creates a three-track design with distinct request (REQ, VALID), acknowledge (ACK), and GRANT signals. As shown in Figure 3.14, the termination latches have separate read and write ports to simplify access to the FIFO and asynchronous-to-synchronous converters are used to synchronize the control signals wherever required. A broad data scheme can no longer be used here since the data

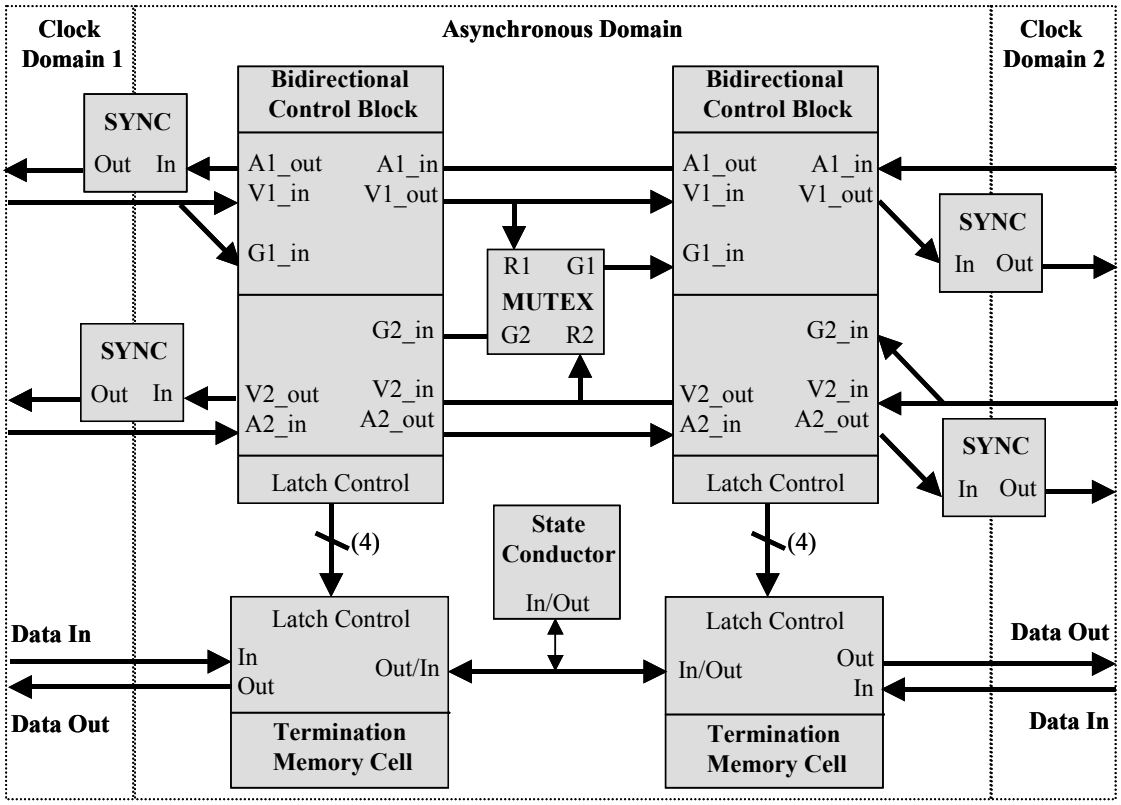

Fig. 3.14: A 2-cell bi-directional synchronizer for transferring data between independent clock domains

needs to be removed from the common data bus lines as quickly as possible to free up the resource for data flowing in the opposite direction. To this end, an early data scheme is used in the bi-directional application. As a result, the data latches need to be modified to include both read and write enables as shown in Figure 3.15 to ensure data integrity in the common data lines DATA_L and DATA_R. Since the data lines are not always driven by the output of a latch, a state conductor is required to hold the bit-values on the data lines between latches. Disabling transistors in the feedback inverters in Figure 3.15 was tried, but was found to hinder performance.

Another benefit of this design is that there is no possibility of FIFO overruns since the asynchronous handshake cannot complete until space is available for new data to enter. Provided that the VALID signal is properly synchronized with the data, the data need only be latched when VALID is asserted in the local clock domain since it can be assumed that the data is present as soon as the VALID is asserted in the asynchronous domain. Any information entering the FIFO can be synchronous since

the circuitry is designed to read synchronous and asynchronous signals in the same way. It is the order of the transitions, not their duration or relation to a clock that is important.

3.3.4 Operation

The most important component of these structures is the asynchronous FIFO used at the core of each synchronizer. The performance metrics shown in Table 1 and 2 represent the maximum transfer rates achievable using the asynchronous control cells and latches (without considering the performance of the asynchronous-to-synchronous converter). For a unidirectional FIFO, the maximum speed was determined by requesting another transaction whenever a previous one was complete. A transaction though a cell consists of a read phase followed by a write phase, each consisting of a 4-phase control transaction. Figure 3.16 represents the total latency of the data

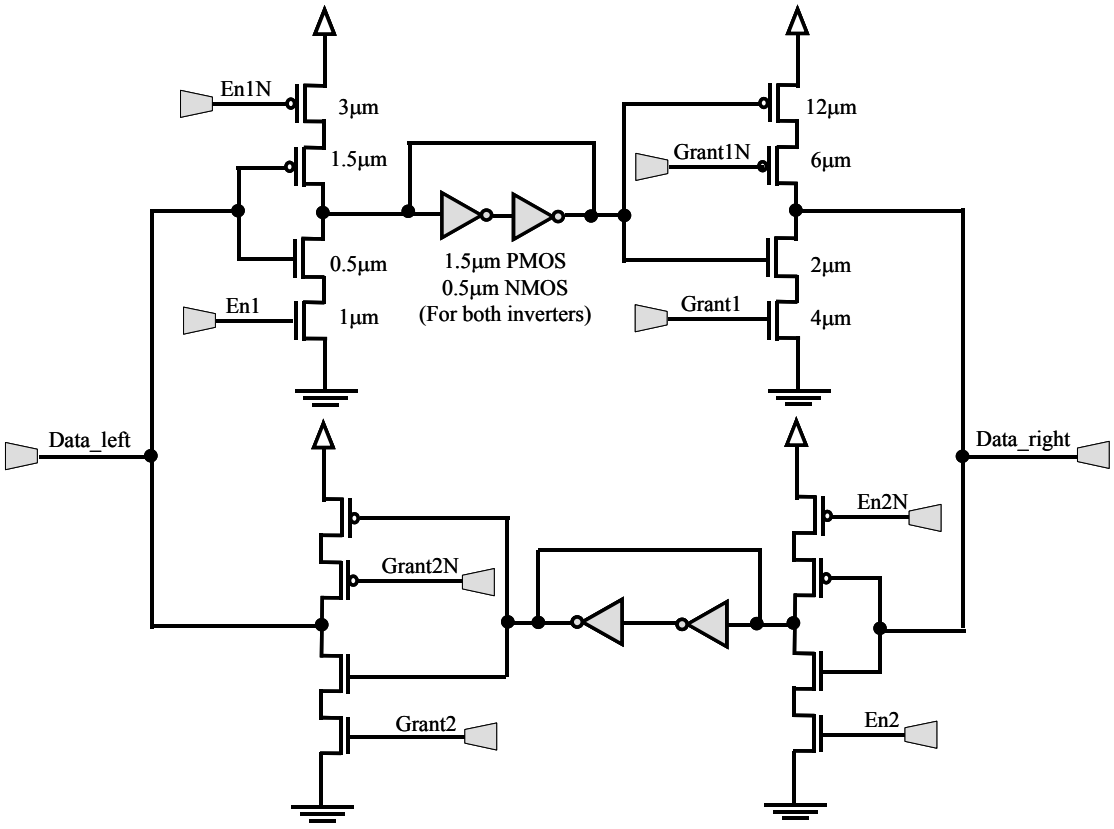

Fig. 3.15. 12-Transistor (per bit) data latch used by the bi-directional FIFO. Indicated transistor widths are identical for reverse direction.

through an 8-cell unidirectional FIFO, which is under 1.8 ns. As with all asynchronous FIFOs, maximum performance can only be achieved while it is half full, with valid data in every alternate cell. Figure 3.17 shows how the Grant (Gx_12) and the Acknowledge (Ax_12) signals in a bi-directional FIFO are non-overlapping so that the data does not collide on the common data line. The optimal situation arises when the Grant and Acknowledge lines are synchronized with the corresponding control lines for data traveling in the opposite direction between adjacent cells so that each cell is simultaneously performing two reads or two writes. This situation allows all the data lines to be used at the same time. Note that the performance of the bi-directional FIFO is latch, not control, limited. Additional delay was added to the control logic to ensure that there was sufficient hold time for the data to be properly latched. As such, if a faster, more efficient latch is built, the system could be made to have even higher throughput than that of the current design.

Like all asynchronous FIFOs that require the transport of data from cell to cell, this design suffers from increased power consumption and latency between request and availability of data. A recently explored approach to solving this problem is to write data to a circular queue and use a pair of token rings to determine where to write data to and where to read data from [12]. While good for tackling both of these issues, this solution is not appropriate for our application due to the transport nature of our FIFO. Data has to be physically moved from one place to another and each FIFO cell has an important role as a repeater, hence there is very little that can be done architecturally to lower the power consumption of the design. Another approach explored in [6] obtains high throughput by using a simple broad data scheme where latches are kept transparent until data is read to preserve data integrity. Such a method is very fast (3.51 giga-items/sec in a 0.25 μm process), but is not compatible with bi-directional communication and cannot benefit from the same resource sharing advantage as our design. Compared to conventional asynchronous FIFO designs in a similar process [8], this design achieves very comparable results (~2.4 giga-items/sec) in the unidirectional case. It is important to note that our asynchronous-to-synchronous converter is only capable of operating up to 2 giga-items/second, thus limiting the performance of any of the FIFOs that use it to this maximum transfer rate in any given

direction. However, should a higher performance synchronizer be built for this application, the theoretical maximum throughput outlined here may be achieved. In the optimal configuration without the asynchronous-to-synchronous converter, the bi-directional control (and thus the bi-directional FIFO) is able to function with a transaction time of 605 ps, which would yield a maximum throughput of 1.65 giga-samples per second (3.3 giga-samples/sec taking into account the bi-directional data flow).

FIFO Type	Throughput (G-items/s)	Transaction Time (ps)
Unidirectional FIFO	2.38	420
Bi-Directional FIFO One Direction Only	1.69	590
Bi-Directional FIFO Both Directions, each direction	1.45	690
Bi-Directional FIFO Both Directions, effective total throughput	2.90	345

Table 1: Performance of unidirectional and bi-directional FIFOs.

FIFO Type	Throughput (G-items/s)	Notes
Bi-directional FIFO	2.9	Effective Throughput
One-Two-One Track Asynchronous FIFO [4]	0.5	Requires strictly alternating flow 0.8 μm process
MOUSETRAP [6]	3.51	No resource sharing possible due to unidirectional flow
Asynchronous Interlocked Pipeline [7]	3.3+	2-phase single track design is incompatible with bi-directional flow
Asynchronous FIFO with fights [8]	2.4	Very similar unidirectional throughput
Low-latency FIFO using Token Rings [12]	0.454	Incompatible with the inter-block transport nature of the design

Table 2: Performance comparison with other Asynchronous FIFOs.

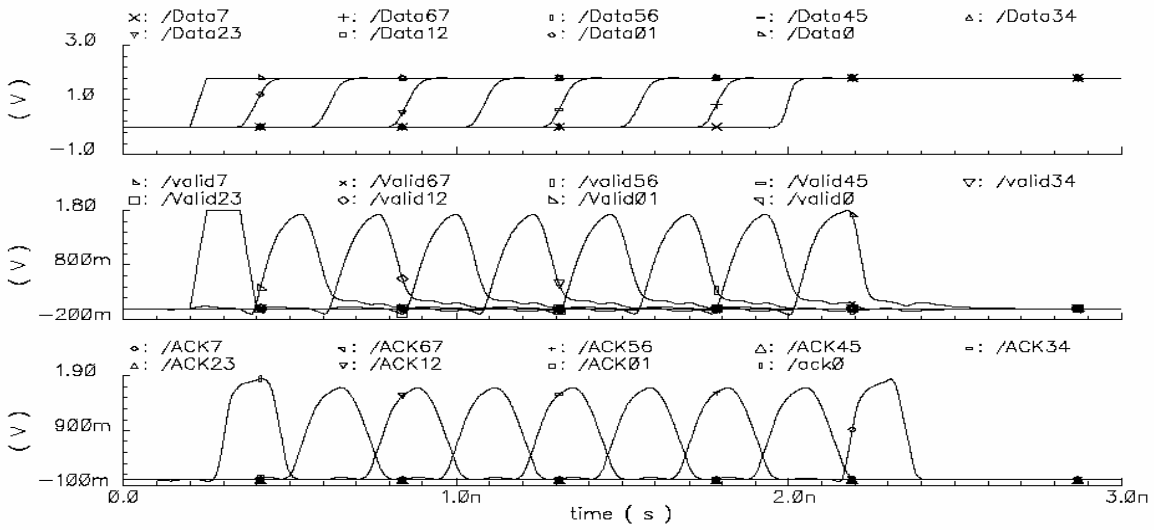

Fig. 3.16: Data propagation through an 8-cell unidirectional FIFO. Each pair of VALID/ACK handshakes corresponds to one data transaction.

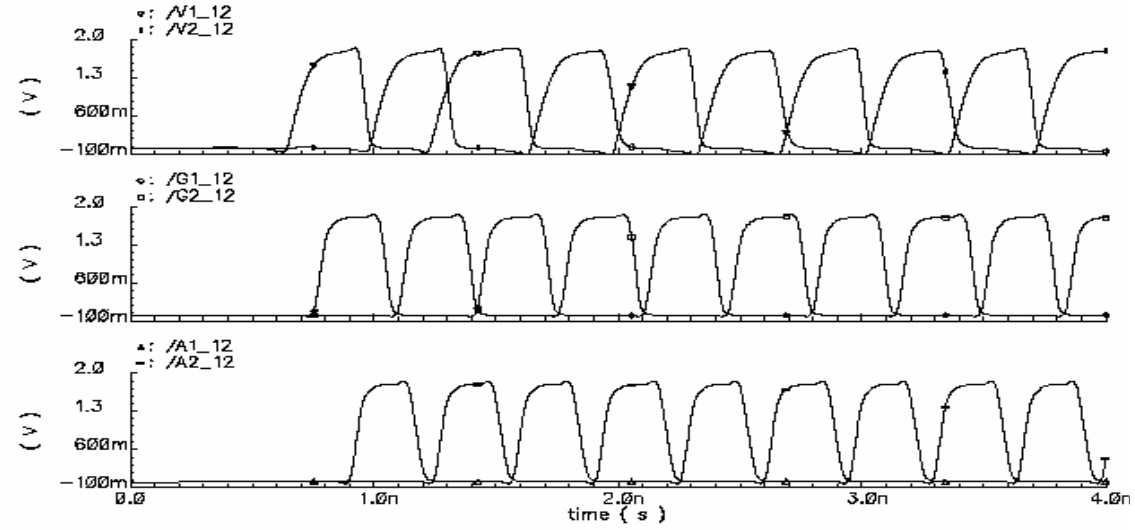

Fig. 3.17: Operation of a bi-directional FIFO

3.4 SUMMARY

This chapter describes a low-power solution for inter-clock domain communication for large ASICs and SoCs. These structures include a unidirectional and a bi-directional synchronizer that perform clock domain conversions through an intermediate asynchronous stage. While there is some overhead in implementing a bi-directional datapath in the method described here, this overhead is easily justified by the obvious resource sharing benefit that this scheme enables. This benefit is especially apparent for wide data busses that transport data over long distances since this scheme effectively halves the number of data lines required. In addition, the total throughput of the system is significantly better than that of a unidirectional FIFO of similar architecture. When paired with synchronizers at each end, such a structure is especially useful in connecting blocks operating with different, non-aligned clock domains such as those found within a GALS or a GALDS design. In fact, Globally Asynchronous, Locally Synchronous designs would not be possible without an asynchronous infrastructure used to transfer and synchronize information between the locally synchronous blocks. Thus, without circuits like the ones described herein, the undeniable advantages that such a paradigm can produce like low power operation, high performance, reduced EMI and higher tolerance to clock skew could not be achieved.

3.5 REFERENCES

[1] Hemani, A., T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, Ellervee and D. Lundqvist, "Lowering Power Consumption in clock by using Globally P. Asynchronous Locally Synchronous Design Style," DAC, pp. 873-878, 1999.

[2] Geppert, L. and T.S. Perry, "Transmeta's Magic Show," IEEE Spectrum, pp. 26-33, vol. 37, no. 5, May 2000.

[3] Brynjolfson, I. and Z. Zilic,"Dynamic Clock Management for Low Power Applications in FPGAs," CICC, pp. 139-142, 2000.

[4] Varshavsky, V. I. and V.B. Marakhovsky, "One-Two-One Track Asynchronous FIFO," APCCAS, pp. 743-746, 1998.

[5] Bainbridge, W. J. and S. B. Furber, "Delay Insensitive System-on-Chip Interconnect using 1-of-4 Data Encoding," ASYNC, pp. 118-126, 2001.

[6] Singh, M. and S.M. Nowick, "MOUSETRAP: Ultra-high-speed Transition-signaling Asynchronous Pipelines," ICCD, pp. 9-17, 2001.

[7] Schuster, S., W. Reohr, P. Cook, D. Heidel, M. Immediato and K. Jenkins, "Asynchronous Interlocked Pipelined CMOS Circuits Operating at 3.3-4.5 GHz," ISSCC, pp. 292-293, 2000.

[8] Laberge, S. and R. Negulescu, "An Asynchronous FIFO with Fights: Case Study in Speed Optimization," ICECS, pp. 755-758, vol. 2, 2000.

[9] Zhu, W., Z. Zilic and R. Negelescu, "A Single-Rail Handshake CDMA Correlator," ICECS, pp. 505-508, vol. 2, 2002.

[10] Yun, K. Y. and A.E. Dooply, "A.E Pausible Clocking-based Heterogeneous Systems," IEEE Trans. VLSI Systems, pp. 482-488, vol. 7, issue 4, Dec. 1999.

[11] Muttersbach, J., T. Villiger, H. Kaeslin, N. Felber and W. Fichtner, "Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of on-chip Systems," ASICSOC, pp. 317-321, 1999.

[12] Chelcea T. and S.M. Nowick, "A Low-latency Asynchronous FIFO's using Token Rings," ASYNC, pp. 210-220, 2000.

Chapter 4: All-Digital Dynamic Clock Generator

This chapter focuses on an all-digital clock generator for clock management applications. The generator uses a set of clock dividers capable of glitchless switching between divisions of a high-speed base clock and a clock selector that can quickly switch between these dividers. These components use custom-built blocks including double-edge high-speed latches and synchronous multiplexers. To demonstrate the effectiveness of such a system, a clock generator was designed with 3 independent clock dividers each capable of generating integers divisions between 1 and 8. The clock divider monitors the current status of the divider to determine the appropriate time to switch divisions. A dead time is inserted between independent clock domain switches in the clock selector ensuring seamless switching without glitches. Since the components are to be used for clock generation, they are sized to obtain consistent duty cycles and rise and fall times. This type of device is particularly useful as a local clock domain generator in a multi-clock system. The all-digital nature of the system allows it to be small and easily distributed throughout an integrated circuit. Additionally, the fast frequency switches make the device ideal for dynamically clocked systems.

4.1 BACKGROUND

Large single chip solutions are becoming more and more common for a vast array of common applications. While this phenomenon is made possible by today's large die-sizes, this also creates difficulties with clock distribution as well as excessive power consumption and heat dissipation in integrated circuits such as ASICs and SoCs. A Globally Asynchronous, Locally Synchronous (GALS) [1,2] design style is an effective method of dealing with some of these issues. Here, locally synchronous blocks with independent operating frequencies are connected together with an asynchronous infrastructure. By creating links to communicate between local blocks that work independently of phase and frequency such as those described in Chapter 3, clock distribution becomes less of a concern since clock skew is rendered harmless. In fact, random or specially distributed clock skew reduces the number of simultaneously switching transistors and thus can be a desirable trait by flattening the power spectrum and minimizing the instantaneous current consumption in the circuit. This skew has the added benefit of reducing electro-magnetic emissions (EMI) [3].

One of the key features of such a system is a clock generator capable of providing a stable frequency without consuming too much power. To obtain maximum versatility, a clock generator should be capable of dynamic frequency scaling. When used in conjunction with dynamic voltage scaling, operating the circuitry in a load-based manner where the lowest clock frequency possible to get a task done is used can yield significant power savings. The power consumption in an IC is proportional to the frequency, capacitive load, switching activity and the square of the voltage used. In this case, both frequency and voltage are reduced. The voltage scaling would not be possible with a fixed frequency clock and thus would not be possible without a dynamic clock manager. Having a dynamic clock manager for each local block instead of one for the entire circuit maximizes these power savings. Such a scheme provides the best of both worlds: the possibility of high-speed operation with full voltage swing, and very low power with voltage and frequency scaling. Lowering the power consumption of the overall circuit is done without sacrificing performance since only blocks that are performing non-critical functions are run at reduced speed.

By incorporating dynamic frequency and voltage scaling into a GALS design, this so-called Globally Asynchronous, Locally Dynamic System (GALDS) [4] is able to adapt to the varying loads placed on the different synchronous blocks in the system. This places yet another requirement on the local clock generator besides being small and capable of operating using low power. The generator should have low latency, quickly adapting to the ever-changing needs of the local block with as little idle time between frequency changes as possible. Any time the clock is inactive is a performance drain. Any extra time spent in an inappropriate frequency could be wasted power, wasted time, or it could even create errors if the current frequency is too high to be supported for the current task.

It has been shown that both multi-clock architectures [5] and dynamic frequency scaling [6] can significantly reduce power consumption while generating little to no performance penalty in clock management overhead. The clock generator described herein specifically targets multi-clock systems with dynamic frequency scaling for all the system clocks and would fit into a GALDS system as shown in Figure 4.1. The clock generator is capable of executing rapid frequency changes, with low latency and minimum area overhead. Previously proposed techniques for generating variable

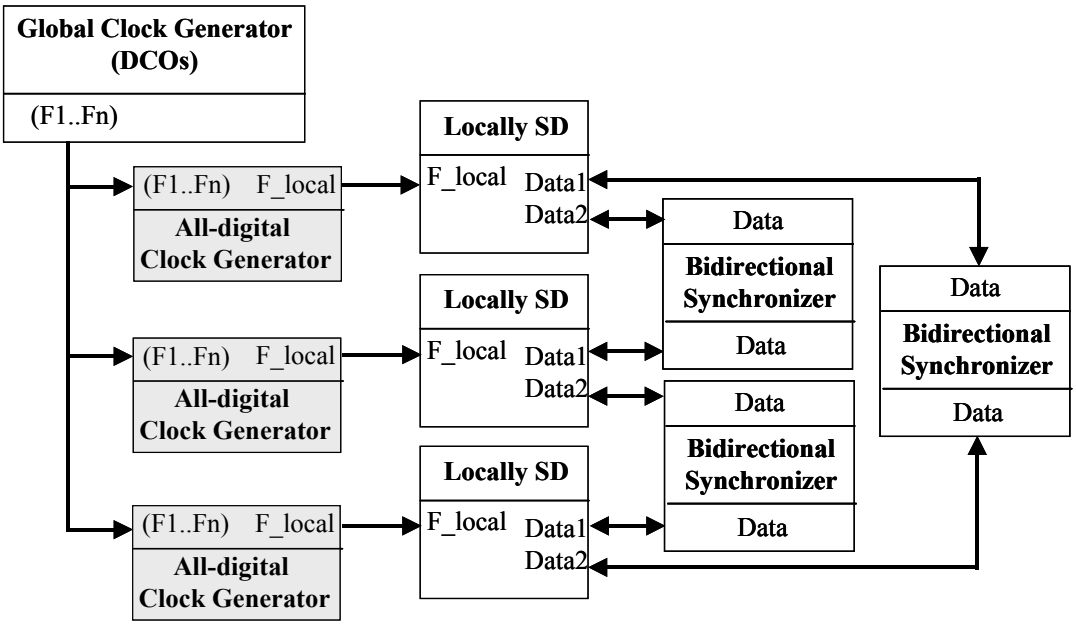

Fig. 4.1: All-digital Clock Generators in a GALDS system. Bidirectional Synchronizers are used to communicate between locally synchronous domains (SDs).

frequency dynamic clocks like the one discussed in [7] have relied on DLLs or PLLs to perform the frequency changes at the heart of the clock manager. This arrangement requires significant effort to ensure stable clocks over a wide range of frequencies, as well as increased area due to the analog components used in the design. Clearly, utilizing a dedicated PLL or DLL for each synchronous block is not an optimal solution. Further, the lock time associated with these devices can exceed hundreds of clock cycles [8]. The key difference between this clock generator and traditional designs is that a PLL or a DLL is no longer used to dynamically switch between frequencies. Instead, a sophisticated clock divider is implemented using custom built latches and multiplexers to generate integer divisions of a globally generated fixed frequency clock (base clock). By using an independent set of these structures (each running off a different base clock) and combining their outputs using a clock selector, sufficient frequency resolution can be achieved to provide high performance while maintaining the lowest possible power consumption. Being all-digital allows it to be small enough to be used for every locally synchronous block in an integrated circuit with little penalty in overall area. As shown in Figure 4.2, the design that is presented here uses a clock divider capable of performing integer divisions between 1 and 8 and the clock selector has been designed to combine up to three clock frequencies coming from independent clock domains. The design is scalable, allowing for more or less divisions and/or more or less dividers. These particular values were chosen as a proof of concept since they provide a wide range of frequencies with fairly good resolution between them.

4.2 CLOCK DIVIDER

The clock divider is constructed as a series of high-speed double-edged latches with taps following each latch. Each tap can be selected, inverted and used as the feedback signal. The delay through each latch depends on the base clock. Specifically, each latch used in the signal path provides a half period delay relative to the base clock so that each latch provides an increment in the division factor by 1. To generate one full clock cycle, the signal needs to propagate through the ring twice, once for the positive and once for the negative cycles. Conventional latches with only one transparent

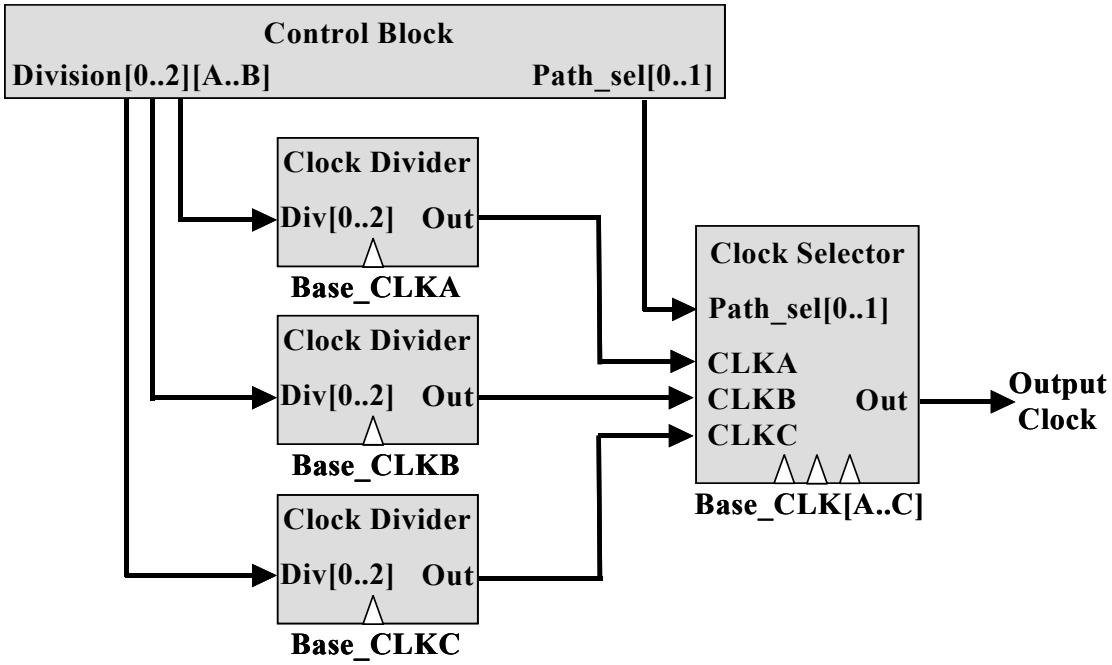

Fig. 4.2: System level overview of the all-digital clock generator using divide by 1 to 8 Clock Dividers and a 3:1 Clock Selector

phase per clock cycle cannot be used for this application because each latch would act as a divide by two block at the maximum frequency and skew the duty cycles of other inputs that were odd divisions of the base clock. Using latches with two transparent phases per cycle (ideally) results in identical frequencies and duty cycles at the input and output of any latch. A target frequency of 2 GHz was chosen for the purpose of demonstrating the potential of this design so all the components used are sized with this target in mind. Higher frequency operation would be possible by using larger transistors, but this would have a negative impact on the area and power consumption of the unit. At this frequency, 2 clock cycles are required to multiplex the 6 taps of the clock divider as shown in Figure 4.3. To generate a divide by 4, the second tap of the ring is chosen providing 2 half period delays through the ring, and 4 overall taking into account the mux delay. To provide divide by 1 and 2 capabilities, dedicated oscillator blocks are used and incorporated into the system's overall clock output using two 2:1 muxes (each with 1/2 clock cycle latency). The divide by 1 block may seem trivial since the base clock could be used directly to obtain this frequency, but it is included to ensure that the dynamic characteristics of the clock (rise time, fall time, near-50% duty cycles) are consistent from one clock division to another. In addition,

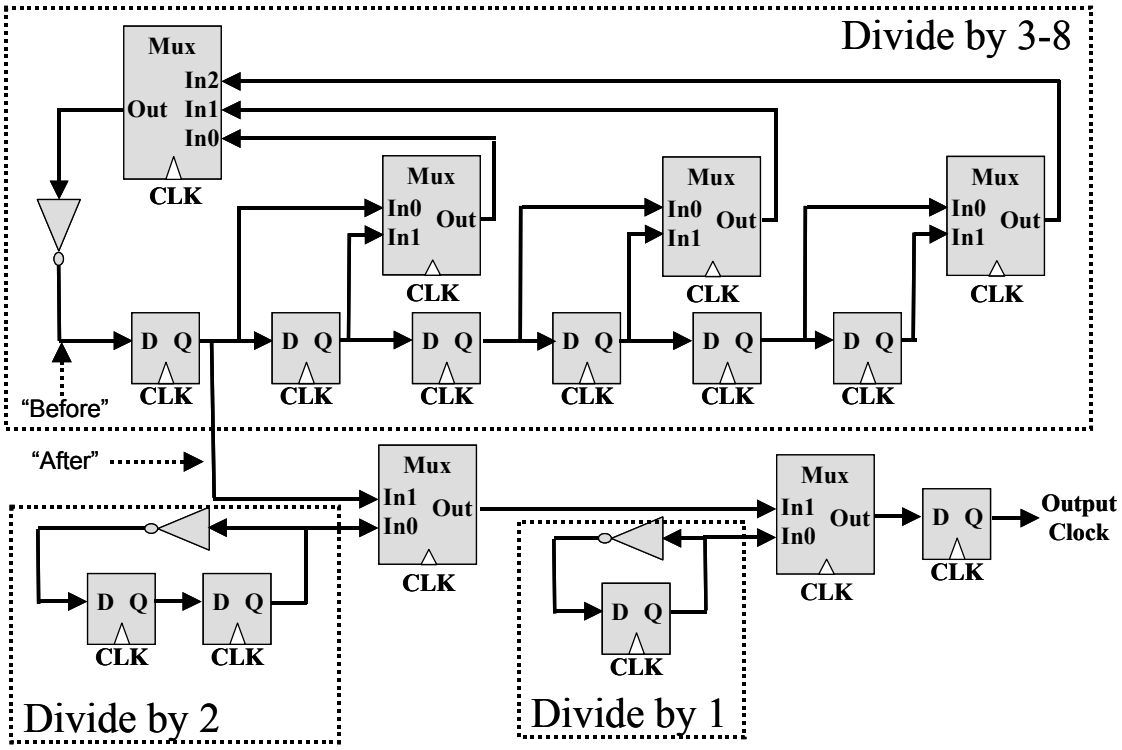

Fig. 4.3: Architecture of the 1 to 8 Clock Divider. All the clocked elements are active on both rising and falling edges.

its presence is vital for the control logic to glitchlessly switch into and out of the divide by 1 state. The design of these building blocks is discussed in subsections 4.2.1 to 4.2.4.

4.2.1 High-speed Double-edge Latch

In designing a general-purpose data latch for this clock management application, there were several important criteria that had to be met. The latch needed to be double-edged to accept data twice per clock cycle. Traditional clock dividers have trouble with odd integer divisions because they rely on single-edged latches. So generating a divide by 5, for example, requires duty cycle skewing in the form of a 3/2 high time-low time ratio (or vice versa). Other methods will generate both a 3/2 duty cycle ratio clock and a 2/3 one and use sophisticated circuitry to combine the two into a single equal duty cycle clock [8]. However, using a double-edged latch to create odd divisions is simple because a double-edged protocol allows data to be read at twice the frequency of the latch clock, as opposed to a single-edged latch where the signaling rate cannot exceed the latch clock frequency. Thus, you can count half

clock cycles. For example, a divide by 5 requires counting 2.5 clock cycles twice, once for each state (low and high). The next requirement is that the transition times of the latch must be consistent, thus a low-to-high transition and a high-to-low transition must have the same Clock-to-Output delay whether the transition is occurring on the rising or the falling edge of the latch clock. This requirement is essential in maintaining a near-50% duty cycle of the output clock. The dynamic characteristics of the double-edged latch are highlighted in Figure 4.4. A third important requirement of this latch is that it should allow the clock generator to create clocks with consistent and comparable rise and fall times between divisions. The final consideration is that the "transparent" time, or the time that the latch is capable of responding to changes at the input, must be large enough to allow the latch to switch without being so long as to exceed the clock-to-output time of a previous latch plus the setup time of the current latch. Thus, the transparent time is independent of the base clock used. A transparent time (t_{trans}) of 125 ps was chosen for this application and technology (TSMC CMOSP18) because it satisfies the equation:

$$t_{trans} < t_{clock\text{-}to\text{-}output(previous)} + t_{setup(current)}.$$

Thorough testing has shown that this time is sufficient to ensure that a latch cannot read two consecutive signals (the current input and the intended input for the next clock edge) on the same transparent cycle, thereby avoiding race conditions.

The method of creating transparency in the latch is simple. Four clocks are applied to the system. The first clock is a reference clock. The second is a skewed version of the reference clock with a t_{trans} phase shift. The final two clocks are inverted versions of the first two clocks. Using the skewed clocks allows us to create a transparent period in the t_{trans} length of time preceding each clock edge of the reference clock. A simple, specially sized inverter chain can be used to create the 4 clocks from a single input clock. While the delay through the inverter chain will generate clock skew with respect to the input, this is not harmful in a GALDS system.

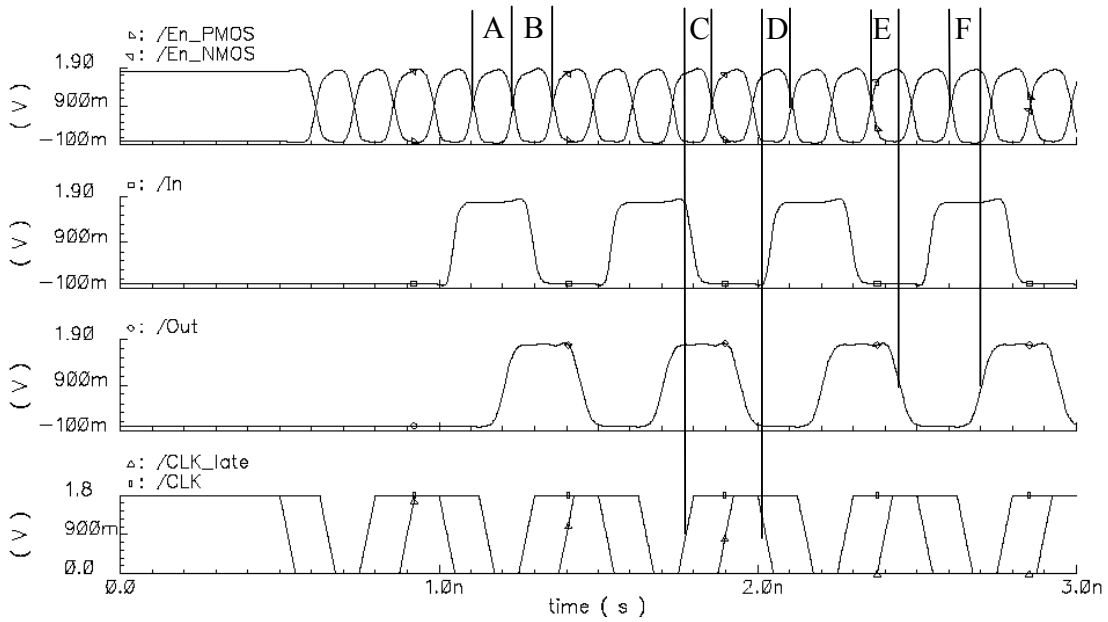

Fig. 4.4: Dynamic characteristics of the high-speed double-edged latch. CLK and CLK_late shown respectively represent CLK1 and CLK1_late in Figure 4.5.

A: Transparent time where latch reads input
B: Opaque Time where latch does not read input
C: Clock-to-Enable time for the NMOS branch (80.274 ps)
D: Clock-to-Enable time for the PMOS branch (84.408 ps)
E: Enable-to-Output time for the NMOS branch (97.303 ps)
F: Enable-to-Output time for the PMOS branch (87.226 ps)
Not Shown: Clock-to-Disable for the NMOS branch (85.248 ps)
Clock-to-Disable for the PMOS branch (83.323 ps)
Total Clock-to-Output (NMOS branch) (177.577 ps)
Total Clock-to-Output (PMOS branch) (170.549 ps)
Rise time at output (60.952 ps)
Fall time at output (59.430 ps)
High time/Duty Cycle (243.905 ps / 48.8% high)

As shown in Figure 4.5, there is circuitry designed to create precise enable signals for the latch utilizing the four clocks. The enable signals have been carefully generated to have fast rise and fall times as well as simultaneous and equal length transparent periods for both the PMOS and NMOS portions of the latch. Thus, the high duty cycle of the En_NMOS line equals the low duty cycle of the En_PMOS line. One important caveat that stems from using such a latch design is that there is roughly a 35 ps setup/hold time requirement at the input of a latch. This time is in reference to the enable signals and not the clock, since there is a skew that comes between the two as a result of the intermediate circuitry (roughly 80 ps, see Figure 4.4). Since these latches have been designed to work at a maximum speed of 2 GHz (a 4 GHz latching

frequency keeping in mind the doubled-edged nature of the design), there is a 250 ps minimum delay between active edges of the latch. As a result of this and the 35 ps setup/hold time, there is a maximum propagation delay of 215 ps for any logic placed between any two of these latches.

4.2.2 2:1 Multiplexer

In creating the clock divider, two different muxes are needed. The first type is used in the output path of the clock divider as shown in Figure 4.3. These two 2:1 muxes are used to integrate the divide by 1 (abbreviated div/1) and the div/2 clocks into the output. It is designed as an open-collector circuit with mutually exclusive enables controlling access to the common output node. The resulting signal is then buffered through an inverter and output, as shown in Figure 4.6. The inverter is not only beneficial for the signal quality, but it is also required to return the signal back to its original state due to the inverting quality of the enable branches. The most important

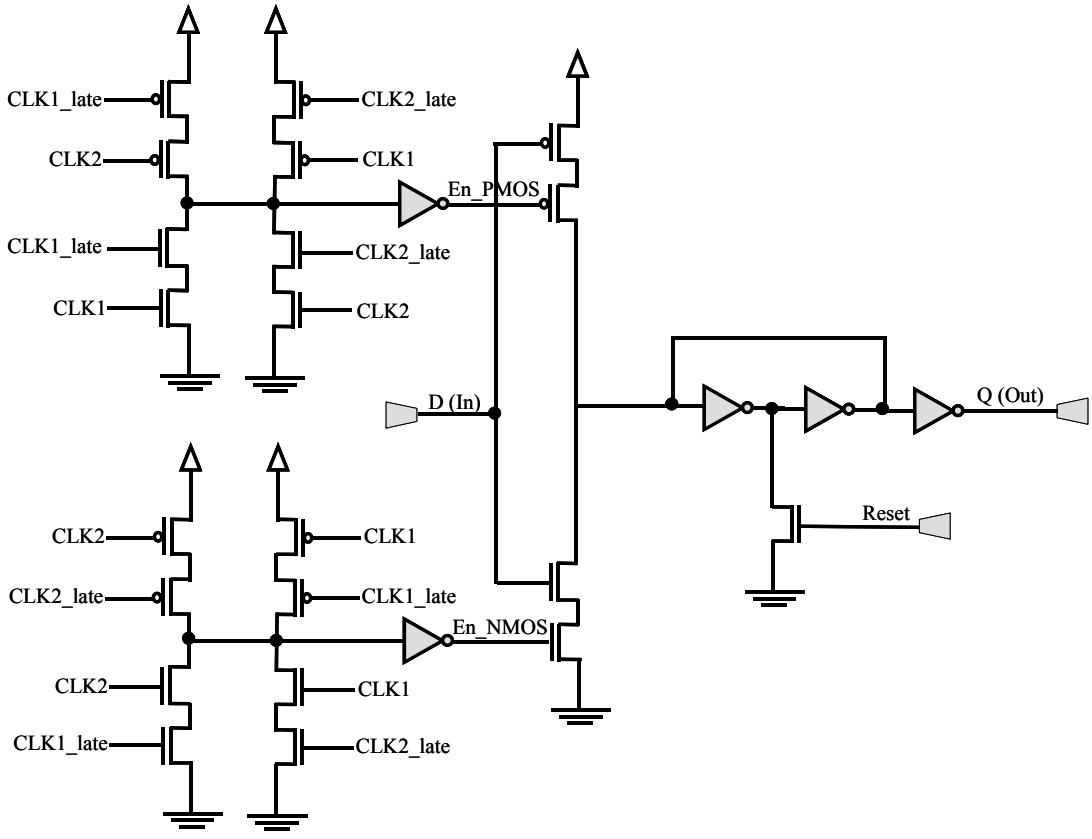

Fig. 4.5: High-speed double-edged latch. En_NMOS and En_PMOS are asserted simultaneously and represent the "enable" as discussed in Figure 4.4.

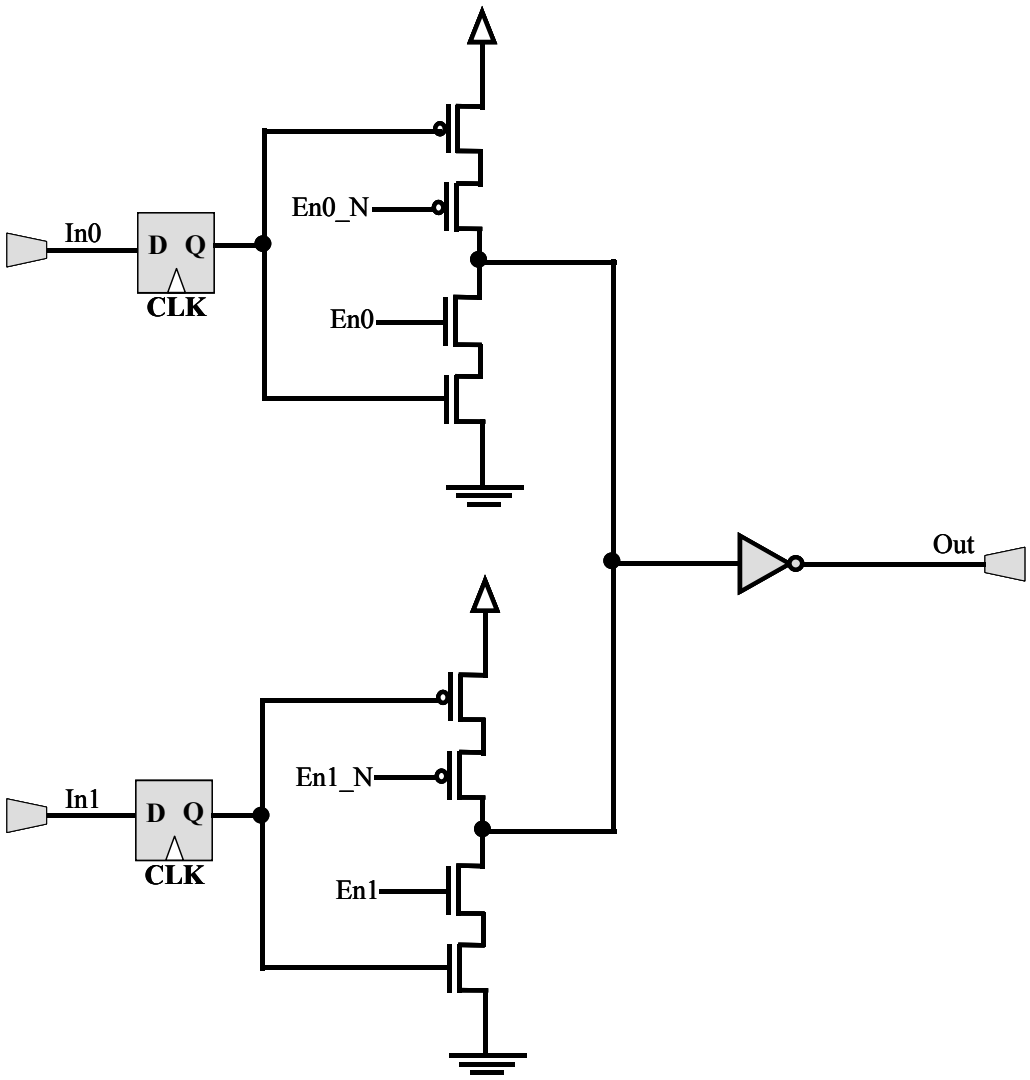

Fig. 4.6: 2:1 high-speed multiplexer with input synchronization

dynamic characteristic of this block is again to ensure consistent high-to-low and low-to high transitions (both in terms of skew with respect to the clock, and rise and fall times) through careful transistor sizing.

4.2.3 6:1 Multiplexer

The 215 ps critical path requirement imposed by the use of the high-speed latches described in 4.2.1 make it impossible to combine the six feedback tap signals in one clock cycle. Both with a single stage (6:1) and a two-stage (6:2:1 or 6:3:1) design, the task could simply not be achieved in 215 ps. The key problem with the single stage was that the common output node in the 6:1 multiplexer was simply too large a

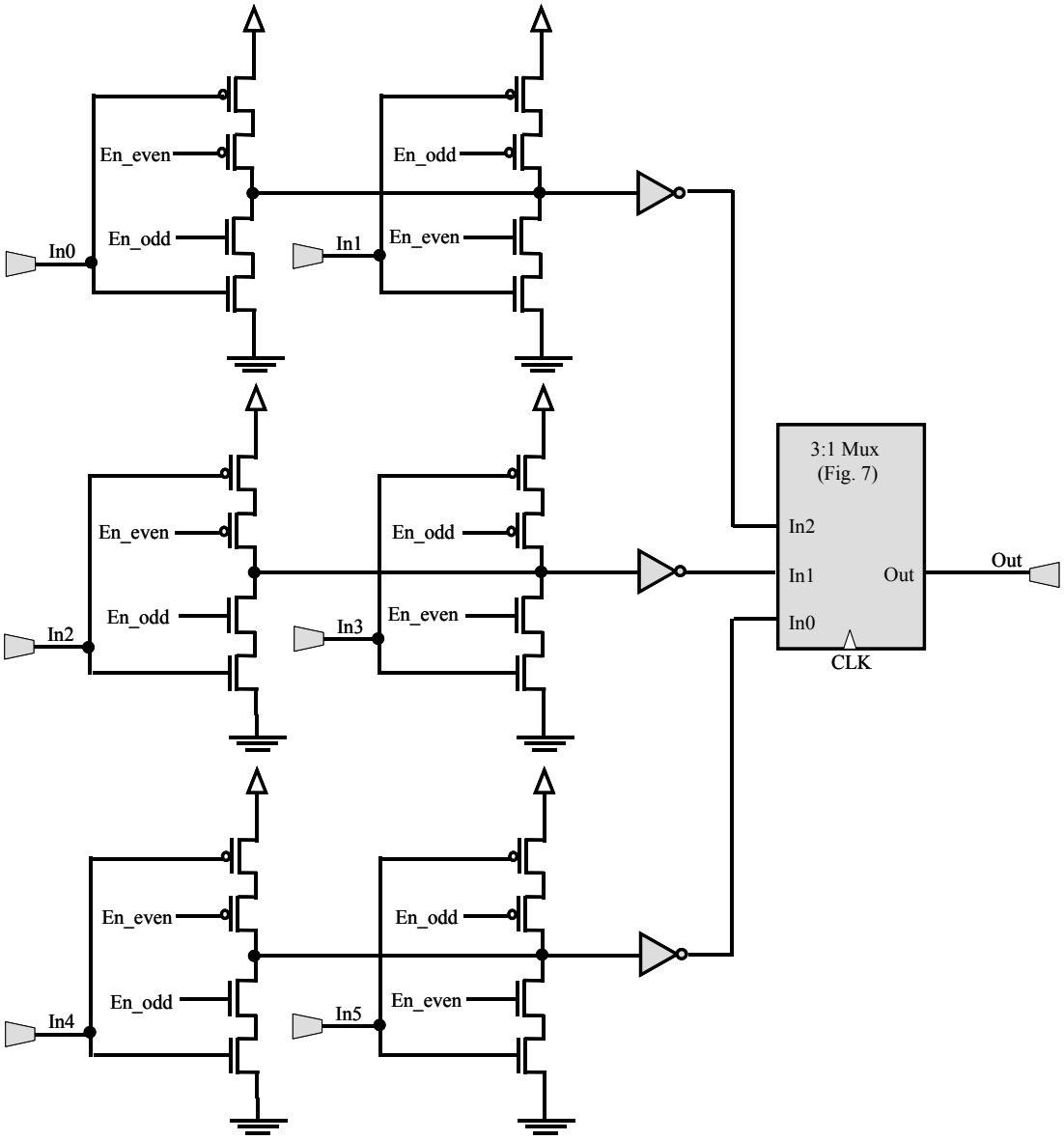

Fig. 4.7: 6:1 multiplexer working in 2 clock stages (6 into 3 into 1). Latches are present at the output of each multiplexing stage and are found within the 3:1 multiplexer (Fig 7).

capacitive load to have the fast and consistent transitions required for this application. Instead, the final design incorporates three 2:1 muxes (described in 4.2.2) without input latches (since the six 2:1 mux inputs are fed directly from the output of a latch) followed by a 3:1 one of roughly the same design. Being pipelined at the output of the 2:1 latches and at the output of the 3:1 mux results in a 2-clock-cycle total delay. Figure 4.7 and 4.8 respectively show the overall 6:1 mux and the 3:1 mux.

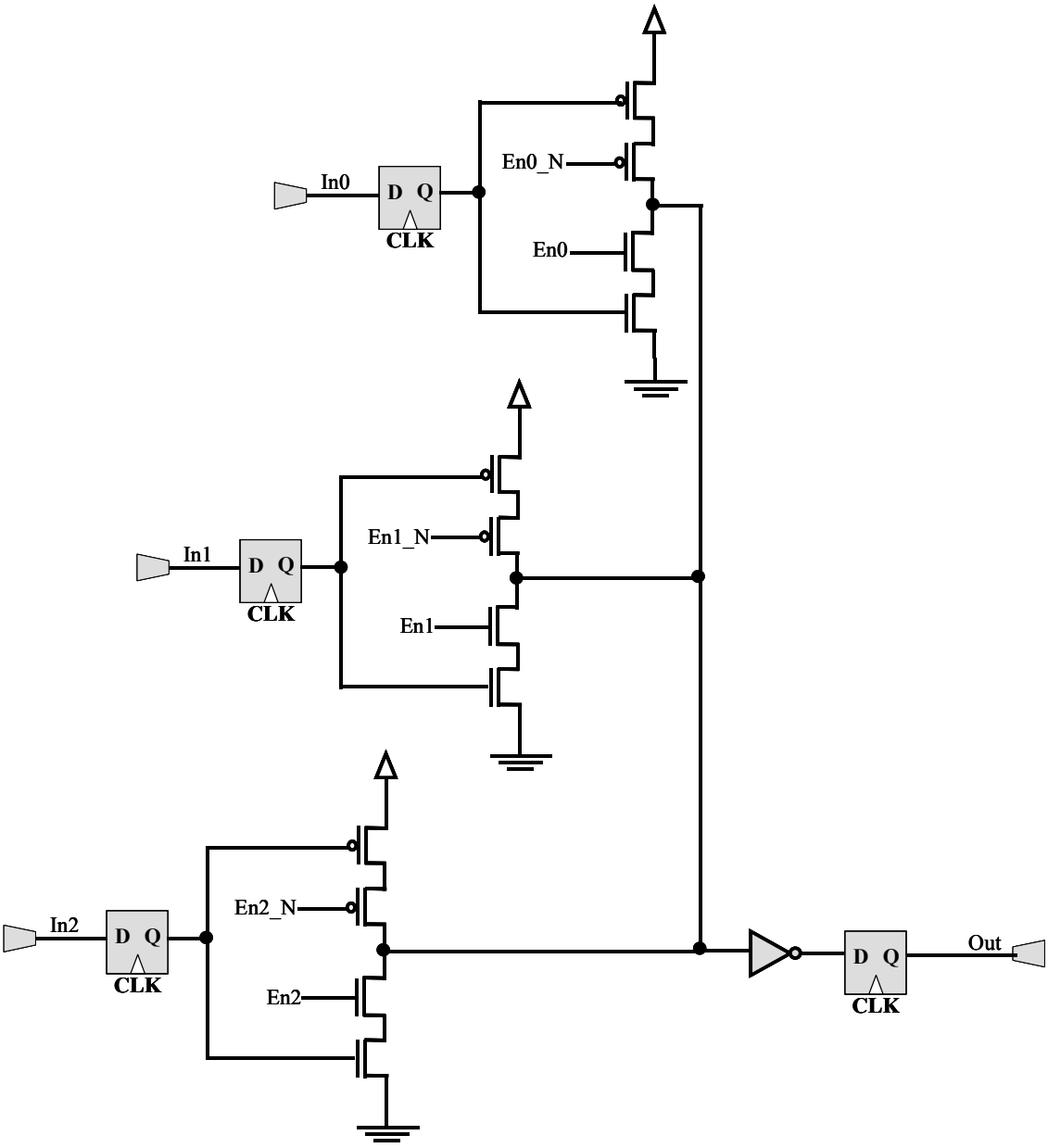

Fig. 4.8: 3:1 multiplexer used to combine the feedback taps in the Clock Divider. The propagation delay between the latches is very close to the 215 ps limit imposed by the 2 GHz requirement of the circuitry.

In one clock cycle, the multiplexer could perform a 4:1 multiplexing in the available 215 ps. Thus, the current architecture of the design could be easily expanded to 16 taps, which would yield 18 available divisions for the clock divider without having to lower the maximum reference frequency that can fed through the circuit. Incorporating more divisions would necessitate a third clock cycle to be used in the multiplexing stage, requiring the move of the div/3 circuitry into the output path with

the div/2 and div/1 blocks. However, for the clock management application being performed here, the 8 divisions produced by the clock divider is more than adequate to achieve the desired frequency range. The inclusion of additional larger integer divisions is unnecessary due to the excessively slow frequencies that they would create. It would have been simpler to lower the 2 GHz frequency requirement and perform the required feedback multiplexing in 1 clock cycle. However, this would have severely limited the performance of the system. By allowing the multiplexing to take an extra clock cycle, we end up with a more scalable architecture, which is far more useful for the broad range of applications and performance requirements of ASICs and SoCs. It should be noted that all the muxes here are sensitive to when their select lines are changed since a mistimed switch could produce glitches at the output of the clock divider. When they can and should be switched is discussed in section 4.2.4.

4.2.4 Control Circuitry

It is important to ensure that every clock pulse be predictable and well-controlled so every pulse switch will either have the width of the initial frequency half-period or the final one during a division switch. To achieve this, great care must be taken in designing the control unit for the clock divider. The three externally generated division control lines are first latched into the local clock domain, then decoded using a set of 3-input NOR gates and then latched again to synchronize the control line bus with the base clock of the clock divider. The control lines are then converted into one-hot encoded, mutually exclusive enable lines for the asynchronous controller. Additional circuitry like a synchronizer is required to combat metastability issues if the clock divider controller is not in the same clock domain as the global frequency selector. Conversely, if the 3-bit control lines are asynchronous, an asynchronous to synchronous decoder is needed. Both of these structures were discussed in Chapter 3. The circuitry involved in modifying the state of the multiplexers within the divider must be capable of responding to control line changes within 215 ps. The fundamental principle behind all frequency switches is that a switch should only be allowed when it does not instantaneously affect the logic level of the output because

this could lead to inappropriate pulse widths. The clock divider performs most of the division changes using the 6:1 tap multiplexer in the feedback path of the oscillator. Changes here are only allowed when the circuitry is on the verge of a 1-0 transition to ensure that an active pulse terminates and thus that every pulse has the desired width. Additional hardware is placed between the latches in the oscillator to guarantee that every inactive node (nodes that are downstream from the active tap) is at a known state (preset high) at all times. The 1-0 transition is detected by a NAND gate monitoring the "Before" and "After" lines of the first latch in the oscillator (see Figure 4.3). The "Before" line is inverted since it is taken from before the tap inverter present in the feedback loop. The "Top" and "Bottom" signals that are created from these signals are then used to control when the enable signals (En_in) are passed into the system. The enable signals (En_out and En_outN) are stored in a state conductor until the next time a 1-0 transition occurs when they are re-sampled by the divider to reflect any recent changes. Figure 4.9 shows the hardware block used to create the enable signals for divisions 3-8. To make the circuitry a little simpler, when divisions 3-8 are inactive, the div/3-8 oscillator comes to rest with the top and bottom signals preset so that the control circuitry can instantly respond to a division change when the div/1 and div/2 enable signals are rendered inactive. Similarly, when div/2 is inactive, its dedicated oscillator is stopped.

Since independent circuits generate the div/1 and the div/2 clocks with no real relationship besides a common base clock, the control circuitry for them is slightly more complicated. In the div/2 case shown in Figure 4.10, the timing of the switch depends on what division is currently active. To enable the div/2 path, the circuitry

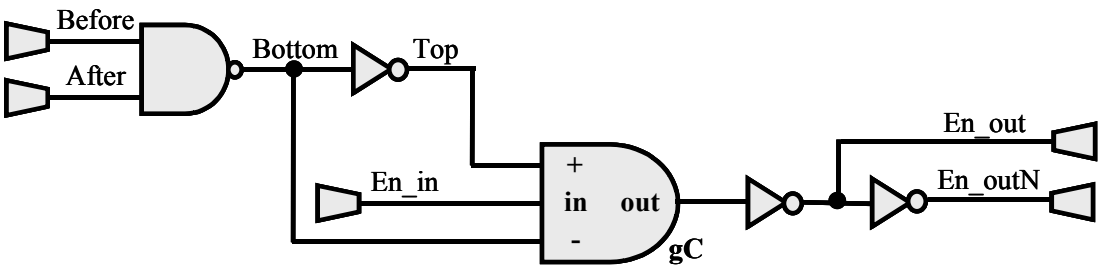

Fig. 4.9: The 1-0 transition detection circuitry (NAND gate) and the sample control block for the div/3 through div/8 enable lines

either waits for a div/3-8 1-0 transition to occur or, if the clock divider is producing a div/1 clock, it performs the transition immediately since 2:1 multiplexer Number 1 is not in the signal path of the current output and the timing of the clock division switch will depend on 2:1 multiplexer Number 2 located further downstream from the div/2 oscillator. A similar distinction is made in de-selecting the div/2 path. The div/1 path requires independent mechanisms to enable and disable the division due to the high-speed nature of the div/1 clock. As shown in Figure 4.11, both the div/1 clock entering IN0 and the output of the multiplexer must both simultaneously be low for this division change to occur smoothly. This ensures that all clock pulses are high for their intended length of time. To disable this division, it is sufficient for both inputs of the multiplexer to simultaneously be low. This simplification is allowed because no pulse can be cut short in the div/1 path since all pulses are a maximum of one half base clock cycle to begin with and due to the synchronous nature of the mux, this is as small a resolution as the system can have.

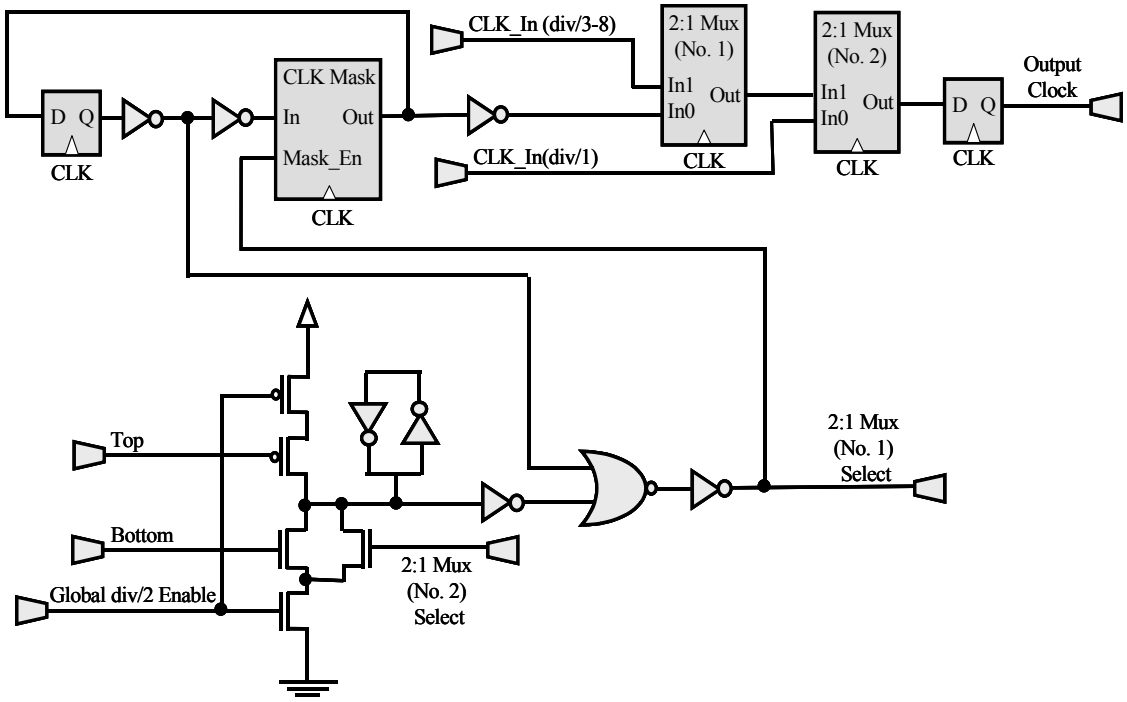

Fig. 4.10: Control block and multiplexing logic for the divide by 2 (div/2) area of the oscillator

In deciding upon this criterion to allow frequency switches, we obtain a variable low time between clock pulses during a frequency change depending on the state of the system and the moment that a change request arrives. This value is bounded to:

$$\max(T_{initial}, T_{final}) + T_{base} > t_{low} > \min(T_{initial}, T_{final})/2$$

where T represents a clock period and t_{low} represents the low time between clock pulses. Not all of this low time is undesirable since there needs to be some low time between clock pulses for the output to be useful. The latency of the frequency transition depends on the state of the system since the control block is built to allow for the current clock pulse to terminate before switching. In addition, there is a three half-base clock cycle pipeline in the control path as well as a three half-base clock cycle maximum latency through the output path 2:1 multiplexers and the output buffer. This gives us a three base clock cycle nominal latency through the clock divider. Additional delay may be required to account for the completion of the initial clock's current pulse, but this does not effect the low time between pulses. The worst-case latency occurs on a div/8 to div/2 or div/1 switch. There is a 3 half-cycle

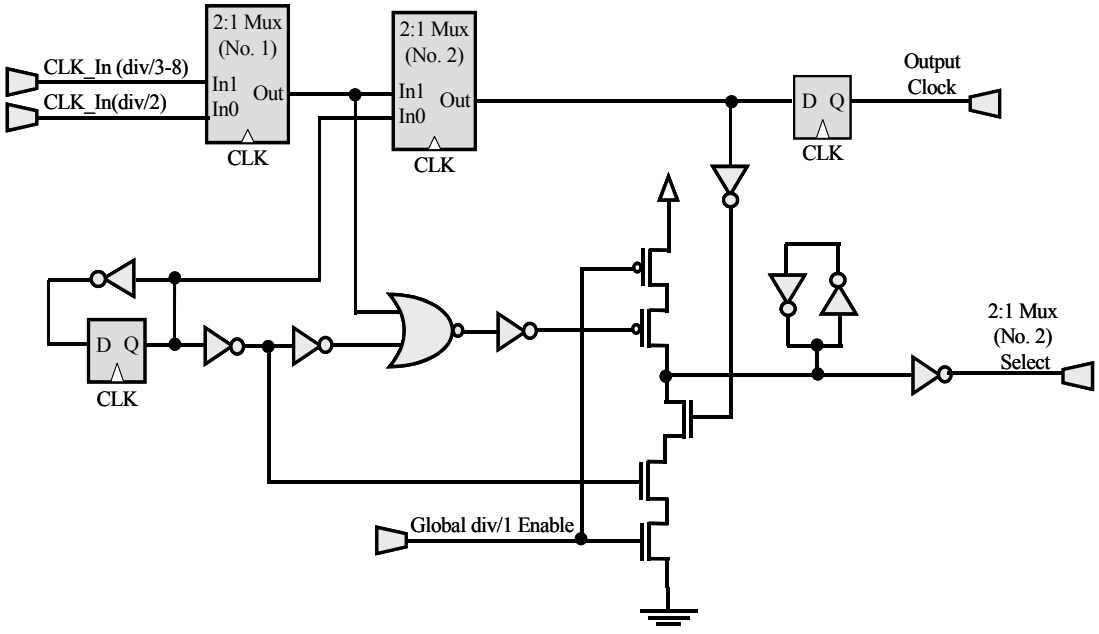

Fig. 4.11: Control block and multiplexing logic for the divide by 1 (div/1) area of the oscillator

latency in the control, followed by a worst-case 15 half-clock cycle waiting period until the initial clock returns to a switchable state (a 1-0 transition), followed by a 3 half-clock cycle delay from the first output mux to the output port. This represents a total of 21 half-clock cycles, which is less than 1.5 periods of the initial clock (which has 16 half-cycles). It is important to note here that most of the transition time is not "lost" time and that the clock is still mostly active throughout the transition.

4.2.5 Operation

Figure 4.12 shows the operation of the divider ramping up through clock divisions 1 to 8 using a 650 ps period base-clock. The div[0:2] signals represent the external input to the divider. First these signals are brought into the local clock domain using the synchronizer described in Chapter 3 and latches operating on by the local base clock. Then these signals are decoded to the En[0..7] signals and reflected in the clock divider when the appropriate switching conditions are met. The unequal length of time spent in each division is due to the variable amount of time the system needs to wait until the system reaches a switchable state between transitions.

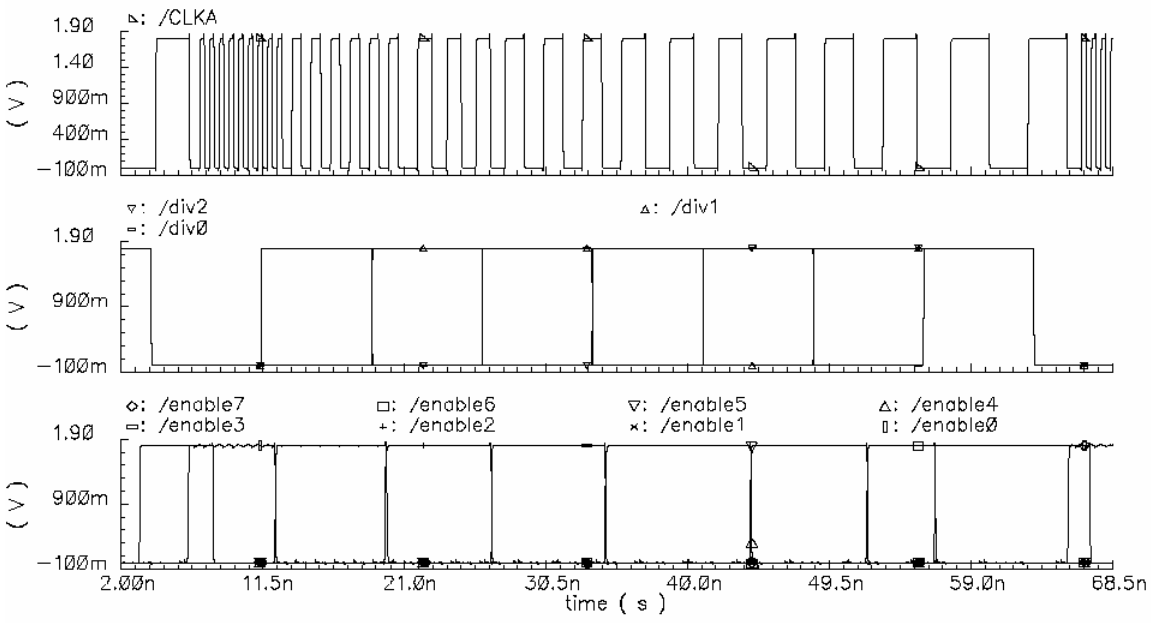

Fig. 4.12: Sample operation of the clock divider demonstrating a frequency ramp-down while operating with a 650 ps period base-clock

Besides satisfying the 250 ps cycle-to-cycle requirement brought about from using both edges of a 2 GHz clock, obtaining consistent dynamic characteristics is an important concern when building the clock divider. These include near-50% duty-cycles and fast and matching rise and fall times regardless of which division is selected. All duty cycles for the frequencies generated by the divider are within 5% of nominal with typical values under 2%. For example, with a 550 ps base clock, the duty cycle of the output clock in divide by 1 mode varies by 2.66% versus nominal (267.68 ps versus 275 ps half-period). All other divisions using this base clock (2 through 8) are under 1%. Table 1 compares the duty cycle of clock divisions using a number of different base clocks.

	500 ps base clock			550 ps base clock		
Division	High time (ps)	Nominal high time (ps)	Duty Cycle (% high)	High time (ps)	Nominal high time (ps)	Duty Cycle (% high)
1	247.210	250	49.44	267.680	275	48.67
2	492.909	500	49.29	544.756	550	49.52
3	735.469	750	49.03	819.655	825	49.68
4	985.539	1000	49.28	1094.72	1100	49.76
5	1235.88	1250	49.44	1369.76	1375	49.81
6	1485.76	1500	49.53	1644.79	1650	49.84
7	1736.30	1750	49.61	1919.91	1925	49.87
8	1984.67	2000	49.62	2194.86	2200	49.88

	650 ps base clock			1000 ps base clock		
Division	High time (ps)	Nominal high time (ps)	Duty Cycle (% high)	High time (ps)	Nominal high time (ps)	Duty Cycle (% high)
1	314.012	325	48.31	484.37	500	48.44
2	638.681	650	49.13	984.32	1000	49.22
3	963.785	975	49.42	1484.41	1500	49.48
4	1288.70	1300	49.57	1984.31	2000	49.61
5	1613.69	1625	49.68	2484.33	2500	49.69
6	1938.72	1950	49.71	2984.31	3000	49.74
7	2263.07	2275	49.74	3484.35	3500	49.78
8	2588.77	2600	49.78	3984.33	4000	49.80

Table 1: Simulated duty cycle of clock divider output for different divisions using a spread of possible base clocks.

4.3 CLOCK SELECTOR

Combining the outputs of multiple clock dividers operating with independent base clocks with no phase or frequency relation is the main requirement of the clock selector. The structure of the clock selector used in this design is shown in Figure 4.13. For this design, a 3-input path clock selector was chosen since it can provide ample frequency resolution as shown in Table 2. The clock selector has been designed to select between any three independent clock inputs provided that they are less than 2 GHz and that the fastest clock is less than twice as fast as the slowest clock.

T/2 (ns)	f (MHz)	Δf
0.500	2000	95
0.525	1905	87
0.550	1818	818
1.000	1000	48
1.050	952	43
1.100	909	242
1.500	667	32
1.575	635	29
1.650	606	106
2.000	500	24
2.100	476	22
2.200	455	55
2.500	400	19
2.625	381	17
2.750	364	30
3.000	333	16
3.150	317	14
3.300	303	17
3.500	286	14
3.675	272	12
3.850	260	10
4.000	250	12
4.200	238	11
4.400	227	

Table 2: Possible frequency spread achievable with this system using 2 GHz, 1.905 GHz and 1.818 GHz base clocks.

4.3.1 Clock Mask

A modified version of the 2:1 mux described in 4.2.2 is used to mask (filter) the clock signals in the clock selector to make the signals compatible with the OR-mux in

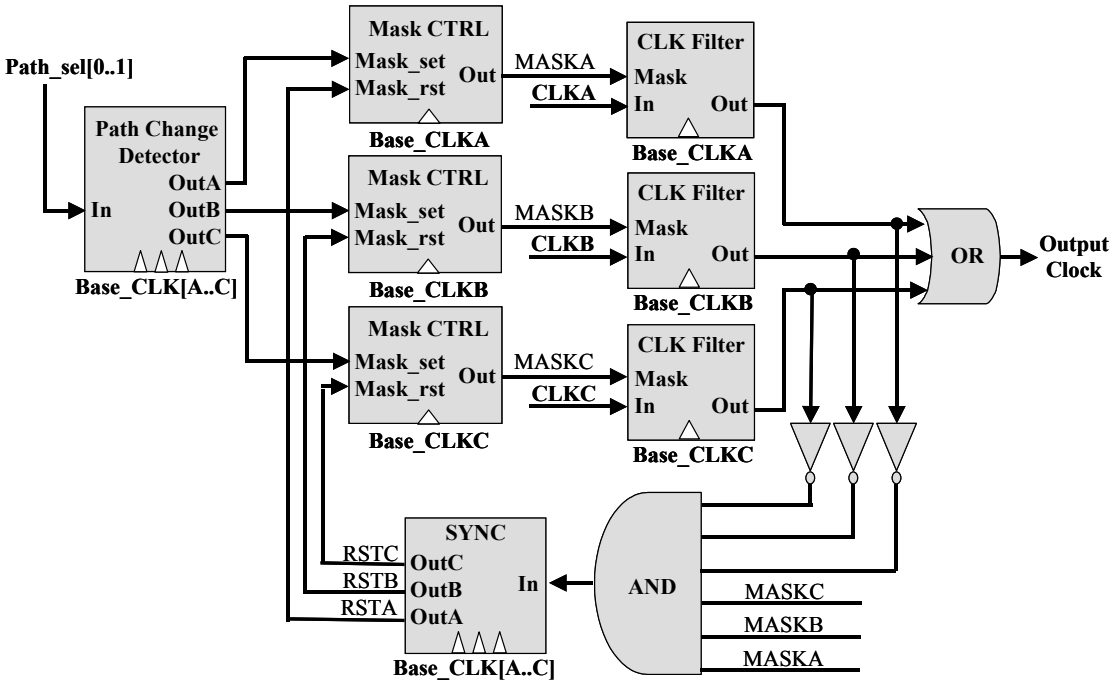

Fig. 4.13: Simplified structure of the Clock Selector used to quickly change between three independent clocks

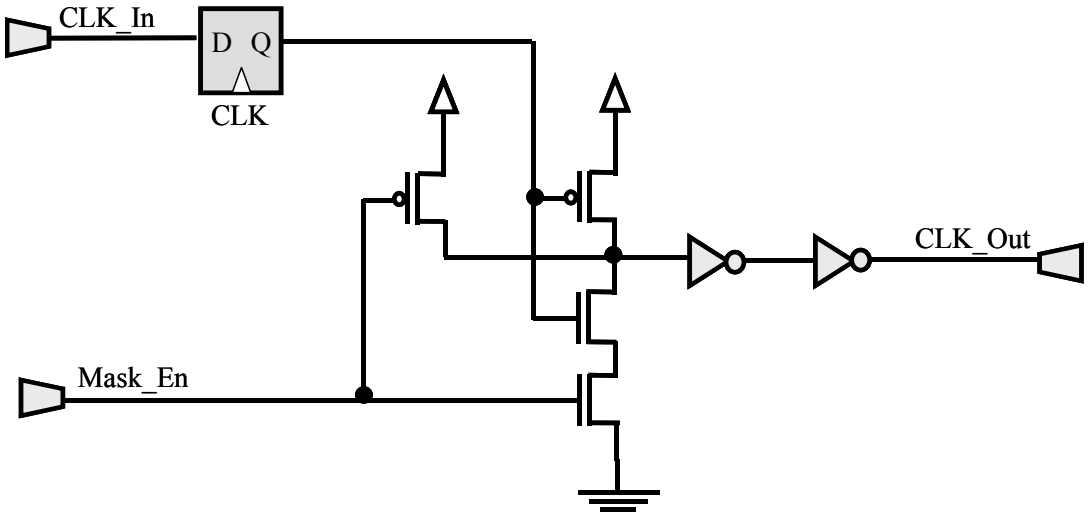

Fig. 4.14: Clock Filter circuitry used to mask/unmask the clocks in the Clock Selector

section 4.3.2. As shown in Figure 4.14, the dual input branches of the mux are replaced by a NAND gate. When the enable input is high, the NAND input is sensitized to the input of the mask. When it is low, the NAND output is pulled high and the output of the mask is tied low through the inverting buffer. Again, careful sizing ensures that the dynamic properties of the input are maintained at the output.

The mask should only be enabled while the output is low. The mask should only be disabled if the input is low. This ensures that activating or de-activating the mask never causes an immediate transition at the output. Due to the asymmetric nature of the PMOS and NMOS areas of the NAND gate, it would otherwise be difficult to maintain consistent rise times or fall times through the mask. In the current scheme, only the transistor closer to the output node is capable of initiating a change at the output and the mask is sized accordingly.

4.3.2 OR-Mux

The actual method of combining the three clocks with no phase relation was not a trivial problem to overcome. The muxes used previously required latches to re-synchronize and pipeline the design. While it would have been possible to use a 3:1 open-collector type design with proper buffering, the control for this design would have been very complicated since there is no precise way to predict the nature and phase of each input clock. A logic solution is easier to control and better capable of maintaining consistent dynamic characteristics of the clock. One important fact that allowed us to do this was that there is no more propagation delay requirement once the signals reach this portion of the circuit. The clocks cannot be latched after they are combined since there is no available clock with a common reference to all the input clocks. As shown in Figure 4.15, a 3-input NOR gate is used to combine the

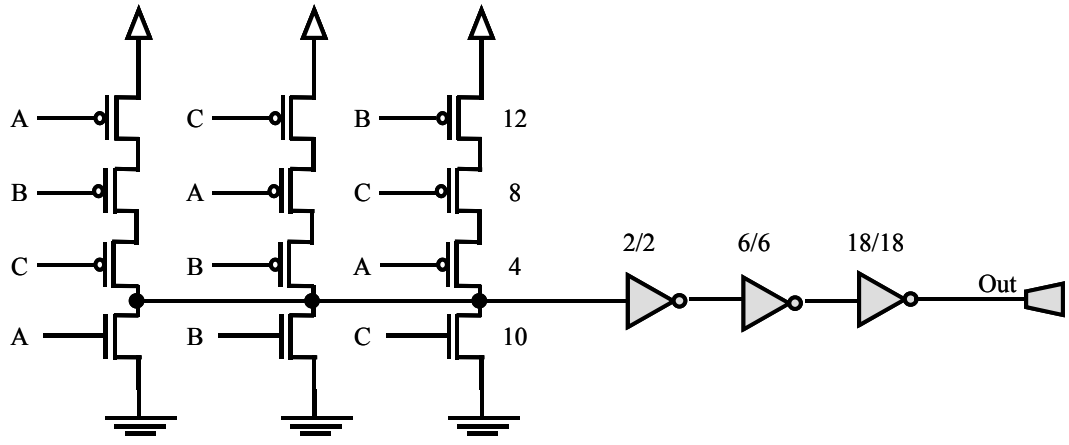

Fig. 4.15: OR-multiplexer used to combine three independent clocks whose high states are mutually exclusive (performed by the Clock Mask in Fig. 4.14)

clocks. If any of the inputs are high, the output is low. An OR-mux is used because it ensures that the dynamic properties of the input clock are not lost as they could be with a more heavily loaded output node design such as an open-collect type mux with output enables on each branch. It has been carefully designed with redundant transistors in the PMOS block that repeat the inputs in all-possible orientations to ensure that the delay from transition to output is constant and thus that the output will have consistent dynamic properties regardless of which input is switching. This method is made possible by the use of the mask described in 4.3.1. By masking out the inactive clock branches to a low idle level, only an active clock can generate a high level through the OR-mux.

4.3.3 Control Circuitry

One clock filter is used for each path in the Clock Selector. The three clock filters (masks) are mutually exclusive so that only one can be active (propagate a high logic level) at a given time. The outputs of these filters are then OR-muxed to create the final output through a buffer. The system works using an asynchronous control block that traps any path change requests in the Path Change Detector. Once a path change is trapped, the signal is then converted to the three base clock domains in the Clock Selector (Out[A..C] in Figure 4.13) and then used to mask the output of the three paths to zero in the clock filters. The filter is only activated when the input clock is at a low state ensuring that no clock pulses are shortened and thus maintaining the correct pulse width in all cases. The time to reach an “all-masked” state is very short and a function of only the active clock because only one path is active at a time. The low time between consecutive clock pulses of different frequencies is related to how a path change is trapped. The path code is read into each of the three independent base clock domains separately and held for at least one clock cycle of each base clock frequency in the Path Change Detector. As soon as one of the paths detects a switch, a signal is generated and transferred into each of the three base domains to set the clock filters to mask the outputs. Once all the masking signals have propagated, the selector waits until the input of the next desired path is low to switch off the mask of that specific path to restart the output. The setting and unsetting of the clock filters is

performed in a memory element with independent set and reset circuitry (Mask CTRL) similar to a SR-latch. The low time is thus completely based on the relative difference in frequency and phase of the base-clocks. Since the circuitry is designed to function with base-clock frequencies between n and n/2 (where max n is 2 GHz), the maximum low time is 3 times the half-period of the slowest clock plus any additional time required for the input to reach a low state (and this is dependant on the exact division being used). This results in a low time bound of:

$$T_{initial} + T_{final} + 3*\max(T_{baseA,}\ T_{baseB,}\ T_{baseC)} > t_{low} > T_{final}/2$$

where T represents a clock period and t_{low} represents the low time between clock pulses. Division changes without a path change occur with a 1-base-clock cycle latency through the block without alteration of the input clock.

4.3.4 Operation

Figure 4.16 shows the operation of the clock selector with 3 fixed divisions of independent base clocks. After a path switch is detected, note how only one mask needs to be applied before the all-masked signal transitions (middle waveform). This is because of the three masks are already active. Once all the masks are active and the three clock paths are tied low, the de-assertion of the all-masked signal is used to start

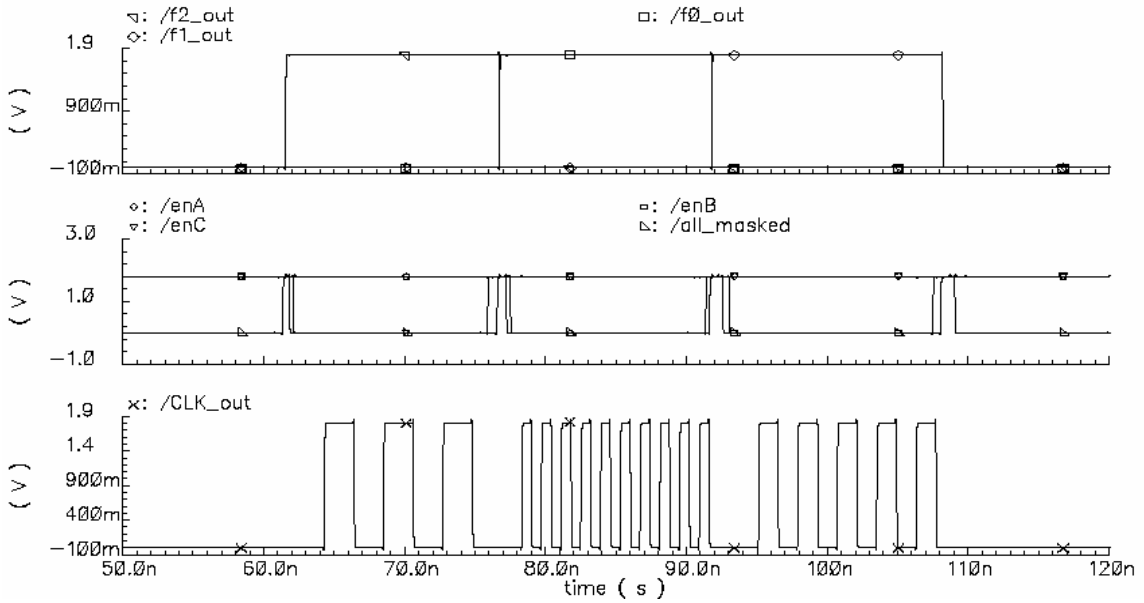

Fig. 4.16: Sample operation of the clock selector switching between independent clocks

the new clock path and reset the system back to a running state. There is also an idle state for the clock selector where all the input clocks are masked. This state can be used to turn off local blocks when they are not in use to save maximum power.

4.4 SUMMARY

Power consumption is a key issue in integrated circuit design today. Dynamic voltage and frequency scaling applications are becoming essential in circuit design. Traditional methods of performing dynamic frequency scaling are flawed since they require significant silicon area and utilize many analog components that do not easily move from one fabrication process to the next. An all-digital clock generator allows this design to be more easily parameterized and thus makes it useful for a variety of process generations. The key to this design is in the transistor architecture and the novel logic behind the frequency changes. Thus, it is ideal as an intellectual property (IP) block to add value to existing designs. The clock generator's performance will scale with decreased feature size to provide operating frequencies above 2 GHz.

With the multiple possible transitions between oscillators in the divider and the wide range of usable base clock frequencies, the controller for the clock divider proved to be the most challenging component to design. However, the resulting circuit remains simple and small despite the complexity due to the asynchronous nature of the control. Similarly, the control for the clock selector had to be designed to ensure that the integrity of all the clock pulses is maintained regardless of transition. Thus the real innovation in this design is not in the core of the clock divider, but in the method and timing of the frequency switches. Having an active clock for as long as possible is important in dynamically clocked systems where frequencies can change often and this design manages to perform these transitions with a very short time overhead. In addition, this solution is smaller than conventional analog designs and thus cheaper to build. In fact, the design is certainly small enough to be used in each clock domain of a multi-clock domain system. As such, each block can be operated at an optimal frequency for the load requirements of the block, thus providing for maximum power savings without compromising performance. This clock generator can be particularly

useful in generating local clocks for a Globally Asynchronous, Locally Dynamic System (GALDS). The all-digital design also switches frequencies on the order of nanoseconds instead of micro- or milli-seconds as is the case with traditional dynamic frequency generators. The external controller that chooses the frequencies that each local block runs at can easily be created in either hardware or in software by a dedicated clock control driver. Since the local clock generator control chooses the most appropriate time to apply the frequency changes it receives, all that the external control block is required to do is to send a five bit code (3-bits for the divider, 2-bits for the selector) to each of the clock generators and the generators themselves will perform the frequency switch in a glitch-free, predictable manner. The speed, simplicity in design and ease of use of this all-digital clock generator combine to make it a particularly appropriate solution for generating frequencies in multi-clock domain systems, especially when rapid, well-controlled, and glitch-free frequencies are required.

4.5 REFERENCES

[1] Hemani, A., T. Meincke; S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee and D. Lundqvist, "Lowering Power Consumption in clock by using Globally Asynchronous Locally Synchronous Design Style," DAC, pp. 873-878, 1999.

[2] Muttersbach, J., T. Villiger, H. Kaeslin, N. Felber and W. Fichtner, "Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of on-chip Systems," ASICSOC, pp. 317-321, 1999.

[3] Bluno, L, F. Gregoretti, C. Passerone, D. Peretto and L. M. Reyneri, "Designing Low Electro Magnetic Emissions Circuits through Clock Skew Optimization," ICECS, pp. 417-420, 2002.

[4] Chattopadhyay, A. and Z. Zilic, "High Speed Asynchronous Structures for Inter-clock Domain Communication," ICECS, pp. 517-520, 2002.

[5] Peiliang, D., R. Yu, H. Xu and C. Yu, "Multi-clock driven system: a novel VLSI architecture," ASIC, pp. 555-558, 2001.

[6] Krishna, V., N. Ranganathan and N. Vijaykrishnan, "Energy Efficient Datapath Synthesis Using Dynamic Frequency Clocking and Multiple Voltages," VLSI Design, pp. 440-445, 1999.

[7] Brynjolfson, I. and Z. Zilic,"Dynamic Clock Management for Low Power Applications in FPGAs," CICC, pp. 139-142, 2000.

[8] Secareanu, R. M., D. Albonesi and E. G. Friedman, "A dynamic reconfigurable clock generator," ASICSOC, pp. 330-333, 2001.

Chapter 5: Digitally-Controlled Oscillator

This chapter focuses on a digitally-controlled oscillator (DCO) designed to generate fixed global clocks for a Globally Asynchronous, Locally Synchronous (GALS) or a Globally Asynchronous, Locally Dynamic System (GALDS). It is the third and final component in the proposed ASIC/SoC architecture described in this thesis and fits into the overall scheme as shown in Figure 5.1. The oscillator is versatile enough to be used with little modification in an all-digital PLL (ADPLL), replacing the Voltage-Controlled Oscillator (VCO) of a conventional Phase-Locked Loop (PLL). ADPLLs are able to provide a variable frequency output with fast lock-in times. Their use is also becoming more common because of their reliability and flexibility compared to PLLs [1]. Should the frequency spread of the DCO be sufficient, it can be used as a dynamic local clock generator with the inclusion of additional control circuitry. At the core of the digitally-controlled oscillator is a variable delay inverting cell and a re-alignment buffer designed to generate fully complementary output clocks regardless of the delay setting used. This feature is necessary due to the presence of a novel asymmetrical delay structure used in the variable delay cell. While a number of different architectures can be implemented using the basic building blocks created for this DCO, the approach that will be investigated in greatest detail involves using a variable delay cell (VDC) with a clock mask in the feedback loop to pause the clock

when necessary. The re-aligning clock buffer is placed outside the feedback loop to shape the complementary output signals produced by the VDC/clock mask pair.

5.1 BACKGROUND

Traditional methods of clock generation in integrated circuits generally require the use of a delay locked loop (DLL) or a phase locked loop (PLL). While the advantages and disadvantages of both architectures have been discussed in other publications [2], the primary advantage of a PLL is its high quality output, whereas the primary advantage of a DLL is its simplicity and efficiency. Researchers have recently begun investigating all-digital PLLs (ADPLLs) to combine the positive aspects of both designs [3]. An ADPLL closely resembles a PLL in architecture with the notable difference of using a DCO rather than a more traditional VCO [4]. The main drawbacks of the VCO architecture are its complexity and long lock-in delays. Being an analog device, incorporating a VCO into an often-noisy digital circuit can be extremely difficult [5]. The digital techniques used for the ADPLL are not only cheaper, more versatile [6] and robust [7], but can also create circuits with fast lock-in times and achieve good stability as well [5, 8].

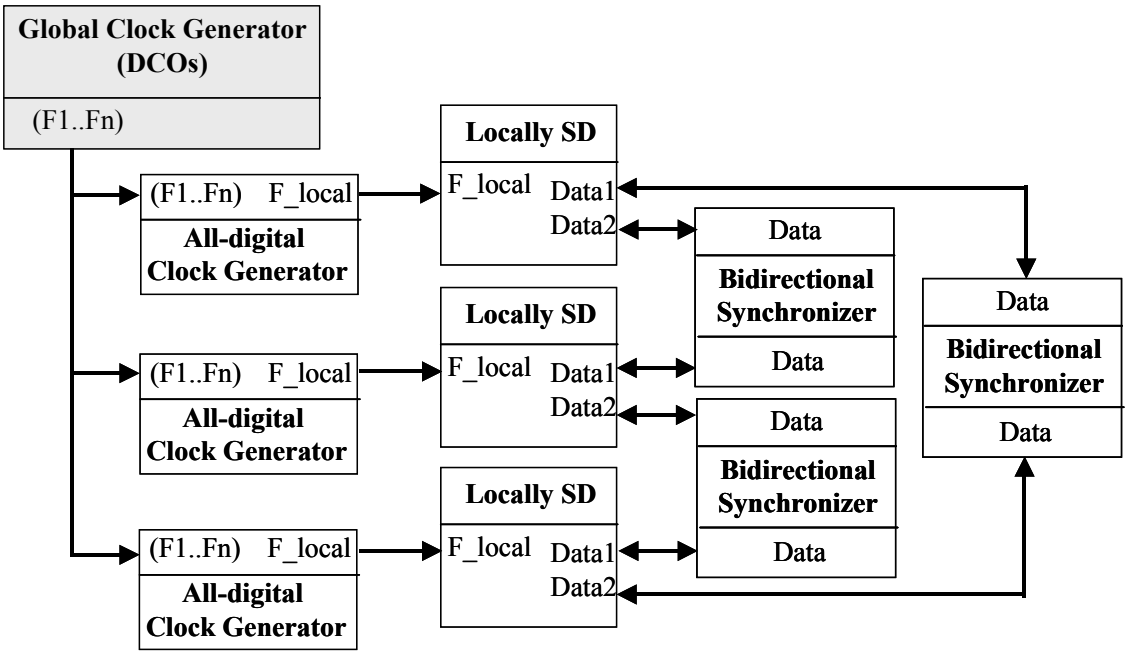

Fig. 5.1: Global clock generator and a GALDS system

In this chapter, we look at a DCO designed to be used either stand-alone in dynamically clocked circuits where the exact operating frequency is not of primary concern or as a key building block in an ADPLL. DCOs and ADPLLs are convenient for digital systems because of their short lock-in times and the convenience of having the frequency controlled by a digital word [9]. There are four general categories of DCOs currently available to designers. The first is a path delay oscillator that uses a chain of logical elements to form a circular ring oscillator. The number of logical elements can be changed to generate different delays through the ring, and subsequently, different output frequencies. This kind of device suffers from low precision and is not appropriate for high frequency operation. The second DCO design is the Schmidt-trigger based current-driven oscillator. This design requires a Schmidt-trigger inverter and a large capacitance whose large area overhead makes it difficult to implement on integrated circuits. The third type is the Direct Digital Synthesis (DDS) DCO that is constructed from a phase accumulator, lookup table and a D/A converter. This method places digital samples of a sinusoid in the lookup table and reads them off periodically to generate an output clock through the D/A. This type of oscillator is otherwise known as a Numerically-Controlled Oscillator (NCO) and is useful due to the signal quality and stability of its output [10]. However, it requires a stable reference clock to operate correctly and can only output a signal at half the frequency of the reference clock (Nyquist rate) [11]. The fourth DCO design is a current-starved ring oscillator where the output frequency can by modified by controlling different MOS switches that alter the delay through the oscillator. This last design is generally known to have good frequency linearity, which is an important characteristic of any DCO [3, 4].

The DCO designed here is fundamentally of the fourth variety, but can incorporate many features of the path delay oscillator by allowing the device to toggle between different feedback paths similar to the design proposed in [8]. The key difference is that control transistors are placed asymmetrically in only the NMOS group, thereby minimizing the amount of area required to implement the device. Traditionally, silicon area is a major drawback of this form of the DCO [5]. The delay cell of the DCO designed here uses only a third to a quarter of the area of a conventional current-

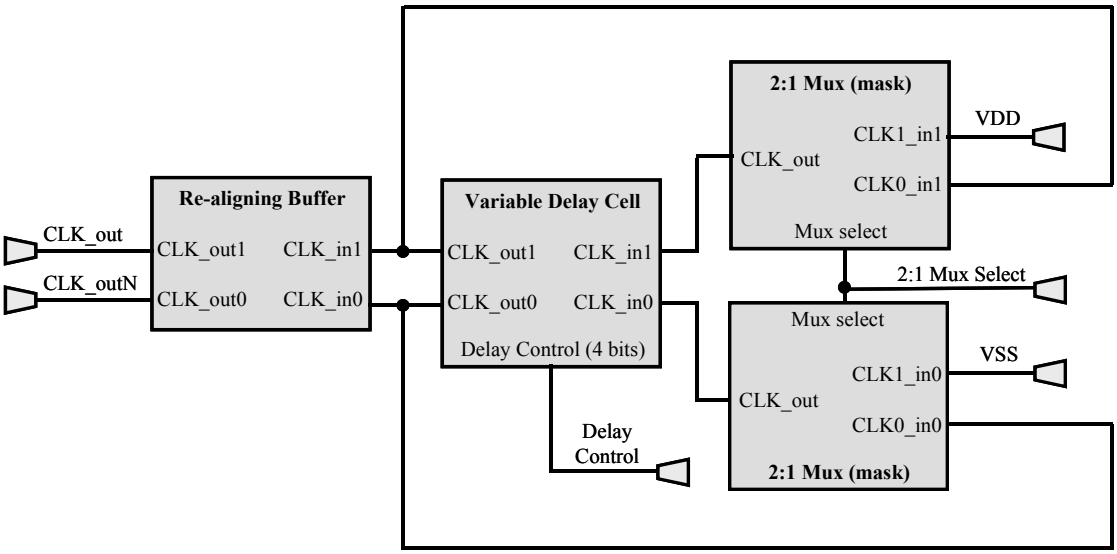

Fig. 5.2: Architecture of the Digitally-Controlled Oscillator (DCO). Note that the variable delay cell is inverting, while the Mux/Mask (used here as a mask) is non-inverting.

starved variable delay cell requiring both PMOS and NMOS delay control transistors. The design of the basic DCO is shown in Figure 5.2 without the inclusion of any additional feedback taps. Other potential architectures of the DCO are discussed in Section 5.6.

5.2 DIGITALLY-CONTROLLED DELAY CELL

There are two critical requirements for the digitally-controlled variable delay cell in this application. First, it has to provide a variable signal delay for the DCO to generate its variable frequency output. Second, the delay cell is responsible for synchronizing the complementary clocks. The transistors in the NMOS group of the first stage of the VDC shown in Figure 5.3 can be controlled to vary the delay of the cell. The four NMOS transistors in the CTRL (control) set are sized as binary weighted powers of 2 (2, 4, 8 and 16) to give the transistor group an effective size range of even numbered multiples between 2 and 30 (inclusive) of the base frequency size (2 μm for this application). Similar to the approach used in [8], this technique achieves propagation delays inversely proportional to the equivalent MOS width. This approach gives the final DCO a frequency range with maximum resolution occurring at the highest frequencies. Clock synchronization occurs in the next stage of the variable delay cell

using inverters controlled by the complementary output of the VDC. For this method to work, the output of the delay cell is not allowed to rise before the output of the complementary branch has been driven low. While this technique does synchronize the two clock signals, it suffers from two drawbacks. First, it generates non-zero delay between the falling edge of one clock and the rising edge of its complement. Next, the circuit can reach an equilibrium point where both outputs are driven high and the circuit will not oscillate. This scenario is avoided using three techniques. First, using particular transistor sizing; second, by slowing the fall time of the first stage of the VDC by placing two series transistors in the NMOS group and third, by ensuring that the complementary input of the variable delay cell has non-overlapping zeroes. The final inverter in the variable delay cell is required to buffer the complementary outputs for the feedback and output paths of the design.

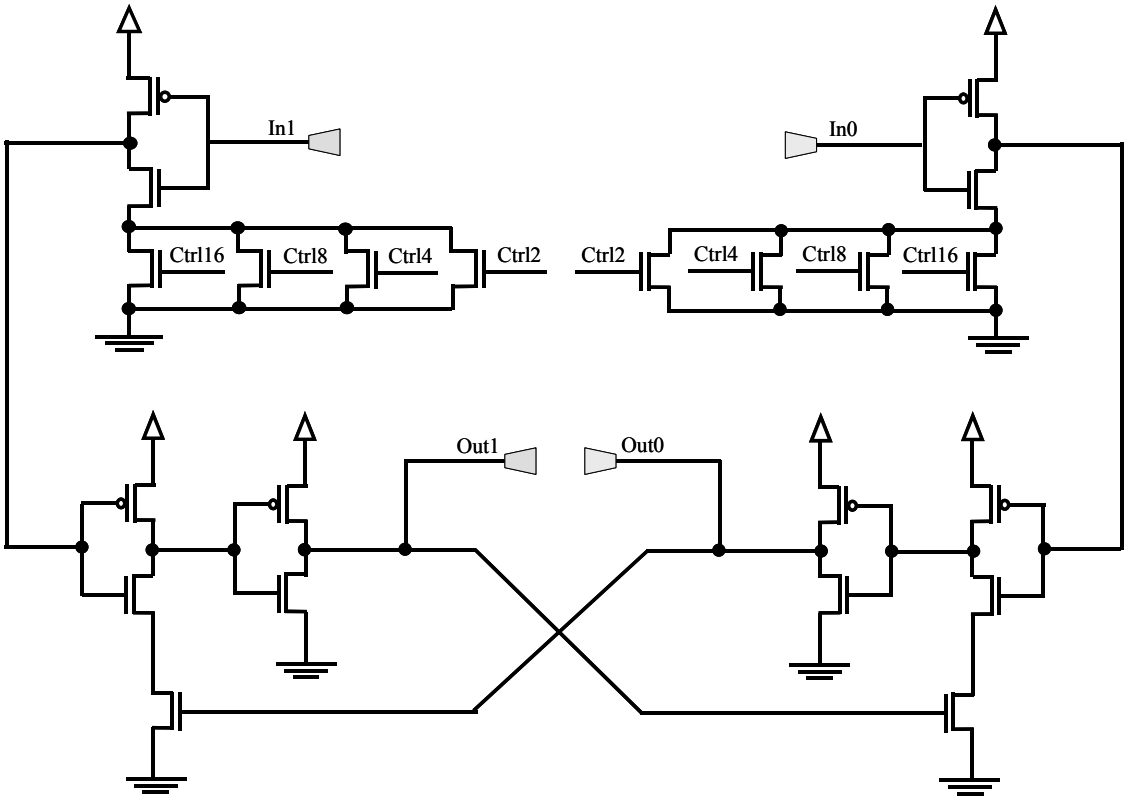

Fig. 5.3: Digitally-controlled variable delay cell (VDC)

5.3 2:1 MULTIPLEXING CELL (CLOCK MASK)

One multiplexing cell is required for each complementary pathway to choose the appropriate feedback path for the DCO. In this application, the feedback path is either the current output of the VDC or constant (VDD in one path, VSS in the complementary one). The use of the constant configuration is equivalent to a clock mask and can be used to pause the clock. The constant input could be replaced with a feedback of slightly longer delay to increase the versatility of the DCO by adding another range of clock periods it can generate. This extension to the hardware block will be discussed in Section 5.6. The complementary select (En, EnN) lines of the logical mux shown in Figure 5.4 are used to control which of the inputs are allowed to propagate to the output. The cross-coupling of the final inverter ensures that the high to low propagation delay is constant regardless of which input is selected (path 1 or 0). Note that the logical mux is equivalent to a more traditional multiplexer design (such as the one shown in Figure 4.6) with additional transistors used to shape the signals to guarantee that the fall time of the output is long enough to be compatible with the variable delay cell.

5.4 RE-ALIGNING BUFFER

The two complementary outputs of the VDC are guaranteed to have identical periods due to the interlocked architecture of the variable delay cell. The output of the VDC will also have partially overlapping outputs due to the nature of signals in the oscillator. The re-alignment buffer is a specialized block that takes advantage of these properties to re-align the clocks. A traditional buffer cannot be used because the amount of overlap will vary depending on the delay setting selected due to the asymmetrical delay control transistors. Utilizing a mutual exclusion element in the re-aligning buffer transforms the output of the VDC with varying overlap into non-overlapping clocks with a fixed non-overlapping time. It is important to re-iterate that this technique will only work if the input of the buffer is overlapping, because with signals that are already non-overlapping, the mutual exclusion element will not alter the amount of non-overlap and thus a fixed amount of non-overlap cannot be obtained.

Once the signals with constant non-overlap are created, a simple, specially sized inverter chain is used to re-align the signals to obtain complementary outputs. The structure of this buffer and the sizing of the inverter chain are shown in Figure 5.5. The sizes shown represent PMOS transistor multiplier over NMOS transistor multiplier. The PMOS transistors have an additional multiplier (2.7 is the value used in this technology) to compensate for the electron mobility differences between these devices.

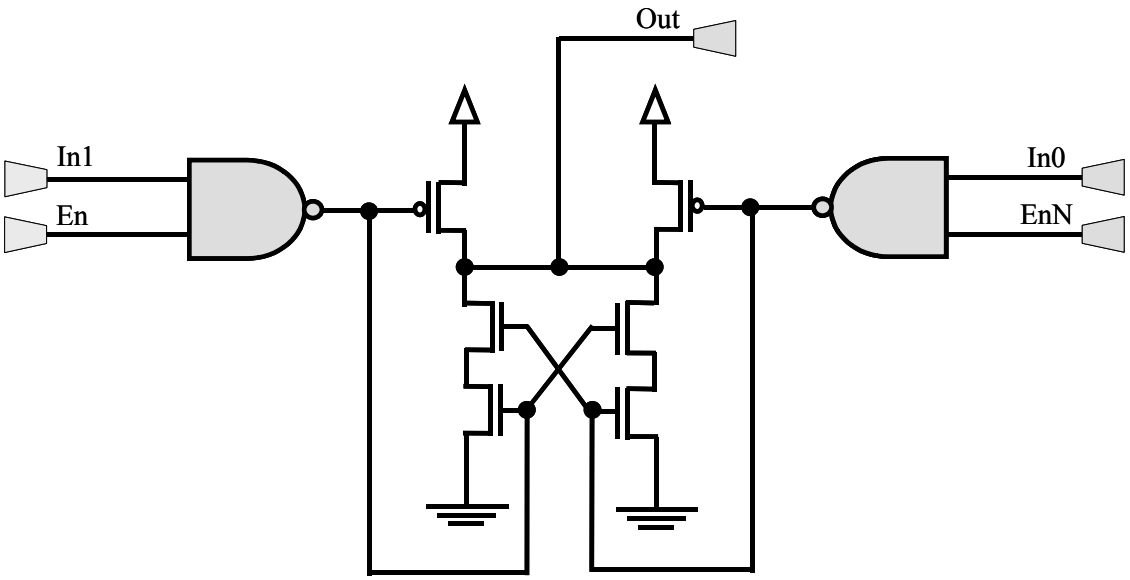

Fig. 5.4: 2:1 Logical Mux used in DCO. A pair of these structures needs to be used in the feedback path due to the complementary nature of the DCO. Second set of inputs (In1 in both instances of the Mux) is constant (and complementary) when Mux is used as a mask.

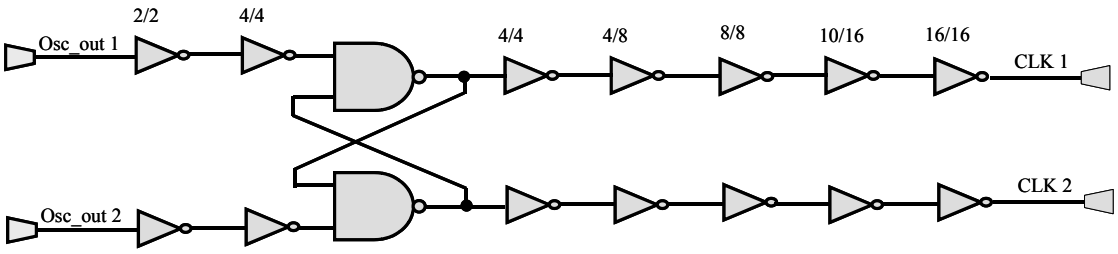

Fig. 5.5: Re-aligning complementary clock buffer

5.5 OPERATION

In testing the DCO, the circuit starts from a reset (masked) state. Once the circuit achieves a steady state, the control transistors of the Clock Mux (mask) are toggled to reflect the desired frequency. The DCO created here is capable of operating in a frequency range between 2.065 to 2.502 GHz in its fastest setting. This represents more than 20% flexibility in the frequencies that can be created using the digital control of the variable delay cell. Should a different frequency range be required, it is possible to configure the device with any even number of additional inverters in the feedback path to create a roughly 120 ps period extension of the final output clock per pair of inverters used. Since the design of the variable delay cell and re-aligning buffer place a restriction on the allowable overlapping/non-overlapping nature of the signals, a pair of inverters is necessary to maintain the required transition times and duty cycles of the asymmetric feedback signals created by this VDC and the 2:1 logical multiplexers. Regardless of the configuration (number of inverters added) or the speed setting (delay code sent to the VDC), fully complementary outputs and fast transition times (around 35 ps) are maintained. For example, a simulated rise time of 37.264 ps and fall time of 34.366 ps are achieved by the DCO in Configuration 2 at controller setting 30. This result is typical for all modes and settings due to the nature of the re-alignment buffer. The duty cycle of the signals does not differ from ideal by more than 1% of clock period regardless of the setting used. Table 1 shows the frequencies and clock periods achievable by such an architecture using 0 (Configuration 1), 2 (Configuration 2) or 4 (Configuration 3) feedback inverters. Controller setting is the total multiplier used in the variable delay cell.

Controller Setting	Configuration 1		Configuration 2		Configuration 3	
	Period (ps)	Frequency (GHz)	Period (ps)	Frequency (GHz)	Period (ps)	Frequency (GHz)
2	484.324	2.065	605.609	1.651	725.508	1.378
4	437.718	2.285	557.780	1.793	677.315	1.476
6	423.113	2.363	542.228	1.844	661.509	1.512
8	415.899	2.404	536.131	1.865	655.569	1.525
10	411.009	2.433	531.054	1.883	650.559	1.537
12	408.720	2.447	528.099	1.894	647.287	1.545
14	405.708	2.465	525.937	1.901	645.209	1.550
16	404.982	2.469	525.143	1.904	644.679	1.551
22	401.684	2.490	521.973	1.916	641.100	1.560
30	399.605	2.502	519.947	1.923	639.226	1.564

Table 1: Possible frequency spread achievable with this system.

Figures 5.6 to 5.9 show the simulated output of the DCO operating with different configurations and controller settings. As long as the output of the variable delay cell has a duty cycle greater than 50% and there is some overlap in the complementary clock pulses, the final output will be fully complementary due to the fixed nature of the overlap following the cross-coupled NANDs in re-alignment buffer. Conversely, the output of the 2:1 multiplexers (clock mask) should be as non-overlapping as possible. In the slowest configuration (Figures 5.6 and 5.8), the duty cycle of the bottom-most waveforms is just over 50% high. This is near the limit of acceptable behaviour since there is very little noise margin available when operating the circuit in this region. In addition, since the complementary signals overlap slightly, there is an imbalance introduced into the system that creates a long-term sinusoidal duty cycle drift in this region of operation that is not present when using higher controller settings. However, the amount of drift present is small (oscillating at roughly 66.7 MHz with less than a +/- 0.5% duty cycle drift in Configuration 2) and the output frequency remains constant throughout so the output of the system is still acceptable for the specified application.

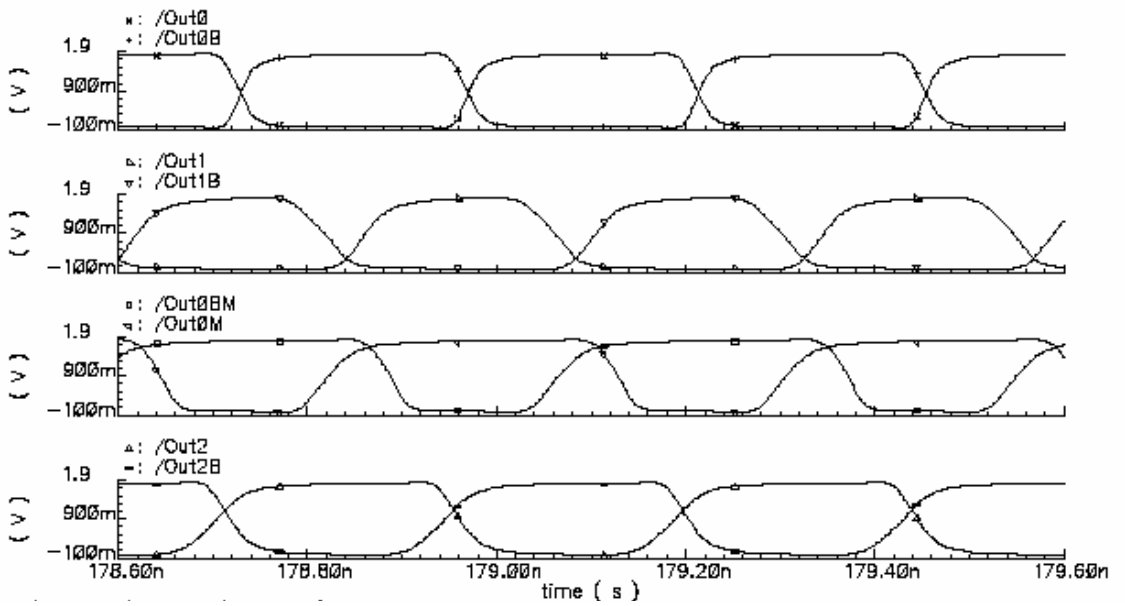

Fig. 5.6: DCO operating in mode 0 (no additional inverters in feedback path) in the smallest possible frequency setting (T = 484.32 ps, f = 2.065 GHz).

Out0/0B: Top waveform represents the final output
Out1/1B: Second waveform represents the output of the cross-coupled NAND gates in the re-aligning clock buffer.
Out0M/0BM: Third waveform represents the output of the digitally-controlled delay cell.
Out2/2B: Bottom waveform represents the output of the 2:1 multiplexing cell.

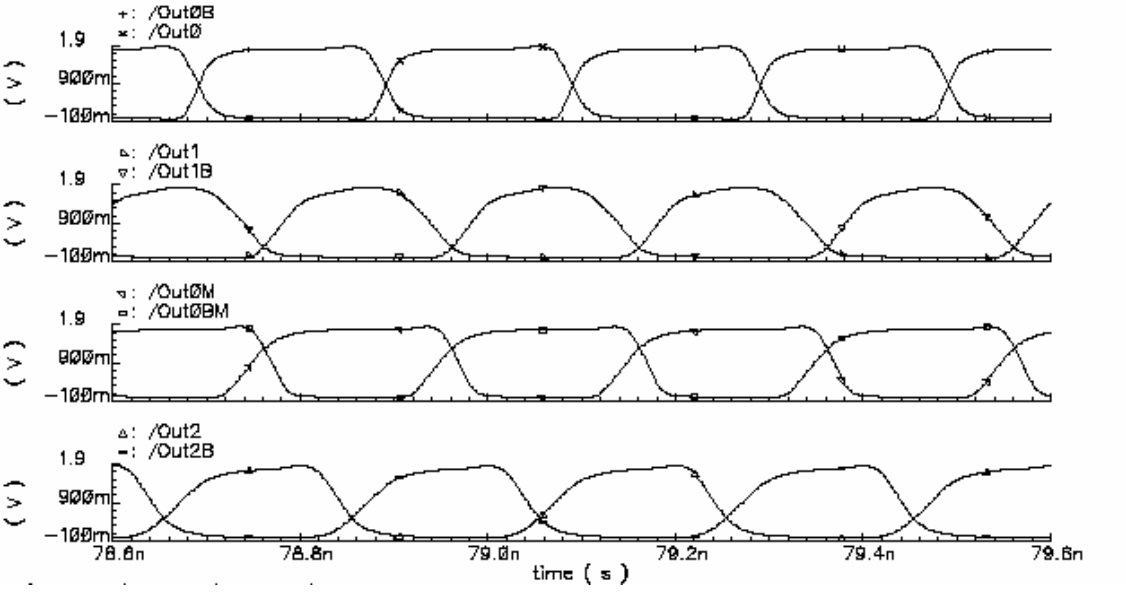

Fig. 5.7: DCO operating in mode 0 (no additional inverters in feedback path) in the highest possible frequency setting (T = 399.60ps, f = 2.502 GHz). Waveforms are as described in Figure 5.6.

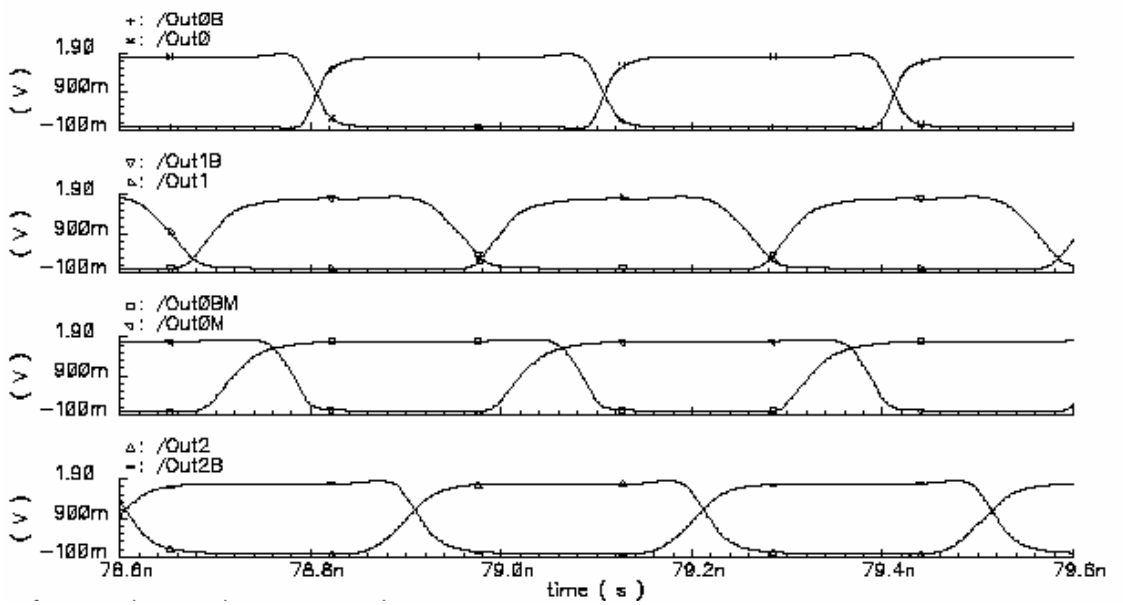

Fig. 5.8: DCO operating in mode 2 (two additional inverters in feedback path) in the slowest possible frequency setting (T = 605.61ps, f = 1.651 GHz). Waveforms are as described in Figure 5.6.

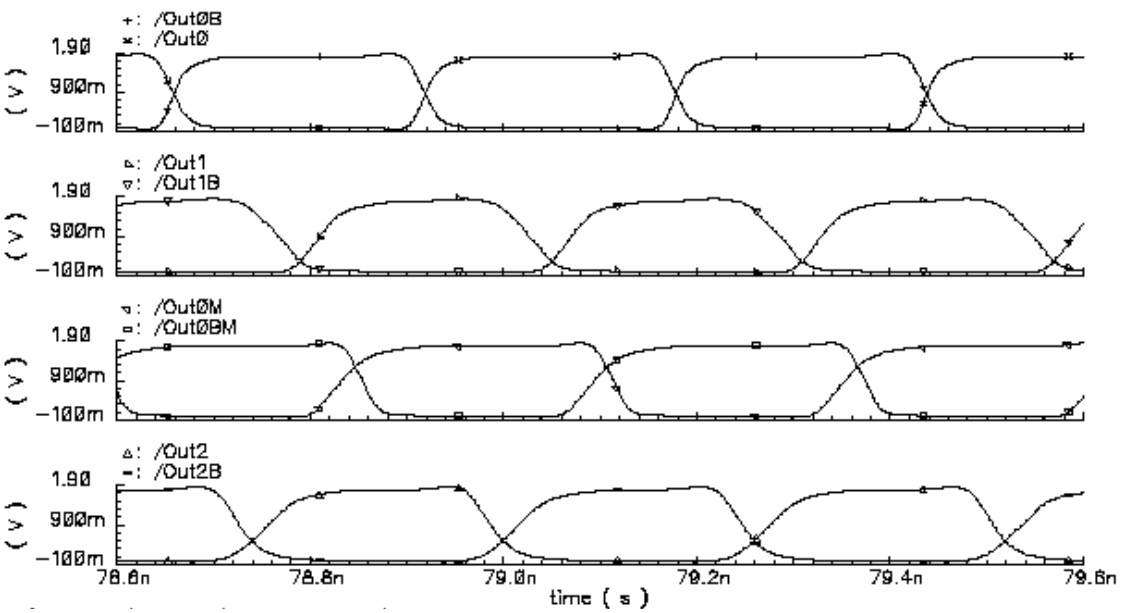

Fig. 5.9: DCO operating in mode 2 (two additional inverters in feedback path) in the highest possible frequency setting (T = 519.95ps, f = 1.923 GHz). Waveforms are as described in Figure 5.6.

5.6 FUTURE WORK

There are a number of possible configurations that can be explored for the DCO depending on the exact nature of the application being considered. If a clock mask (clock idle) is not required, the second input (In1 in Figure 5.2) can be replaced by a 2-inverter delayed feedback signal to give the design the ability to operate in either Configuration 1 or 2 without changing the hardware. However, the clock periods will increase slightly from those presented in Table 1 as a result of the additional loading of the feedback line. The multiplexer could also be removed altogether to create a higher speed mode in excess of 3 GHz. As shown in Figure 5.10, a buffer will need to be placed in the feedback path of the oscillator to make the input of the variable delay cell compatible with its output (in terms of overlap/non-overlap). Another possible architecture involves using pairs of 2:1 multiplexers in a scalable configuration to produce a wide range of adjustable frequencies. Each 2x2:1 mux block and non-inverting buffer pair approximately adds a 375 ps extension to the period of the output, although the design could be optimized to reduce this delay. As is, the system shown in Figure 5.11 is capable of producing clock periods of roughly 400, 775, 1150 and 1525 ps with each period adjustable by up to +85 ps using the variable delay cell. Yet another potential architecture that could be implemented for a range-shifting DCO is as shown in Figure 5.12. This design utilizes three individual DCOs in Configurations 1, 2 and 3 as discussed in Section 5.5 and merges them using the OR-mux from Chapter 4 to produce the complete range of frequencies from Table 1. This technique works because DCOs are small enough to be cheaply duplicated on chip [7]. Such a design is able to obtain higher frequency resolution than the design in Figure 5.11 since a 120 ps range shift is far more desirable than the 375 ps range shift there-in. However, the OR-mux limits the number of possible ranges that can be reached in any given implementation by the design. In addition, either the complementary outputs would need to be eliminated while in “masked” mode or the OR-mux would need to be replaced in one of the signal paths by a structure sensitized to 0’s (instead of 1’s) to pass both the complementary outputs of the masked DCO’s through the multiplexer.

Currently, no control circuitry has been implemented to dynamically change the frequency output of these oscillators. Dynamic shifting is unnecessary due to the DCO's target application as a global clock generator for the all-digital dynamic clock generator described in Chapter 4. However, many of the control mechanisms implemented for the clock divider in Chapter 4 could be re-used for this application with little alteration since the conditions required for a frequency change should largely remain unchanged for the DCO with respect to the previously described clock divider.

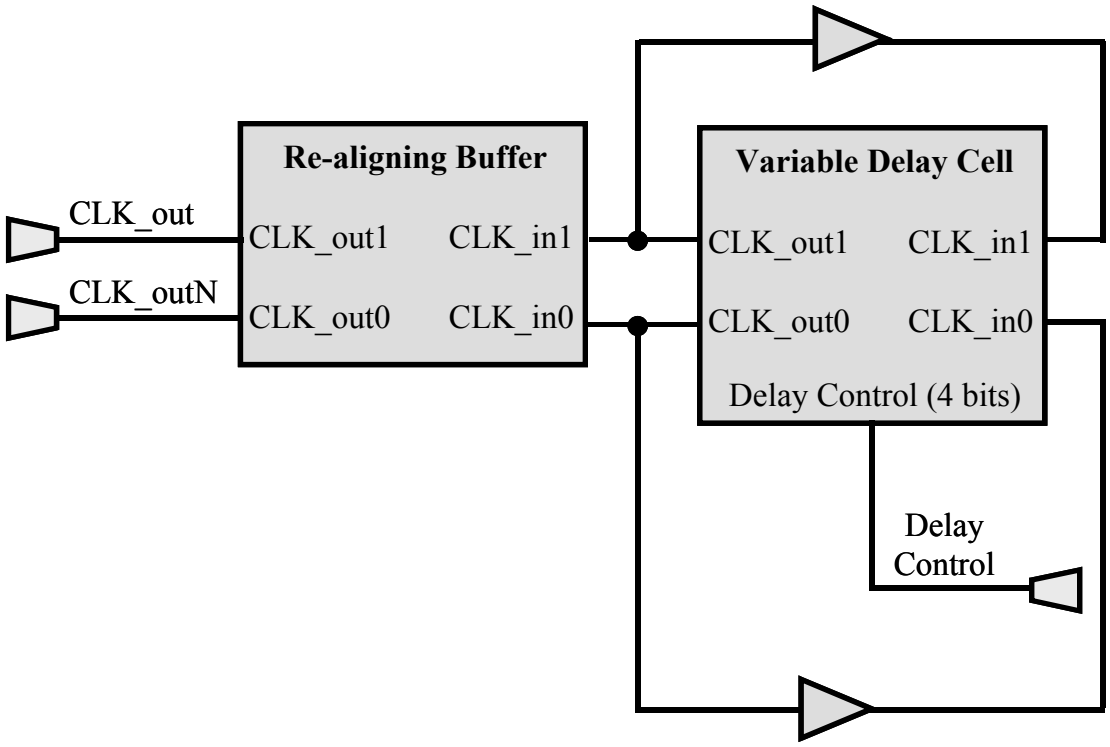

Fig. 5.10: Architecture of the Digitally-Controlled Oscillator (DCO) without idle mode for operating at the highest possible frequency

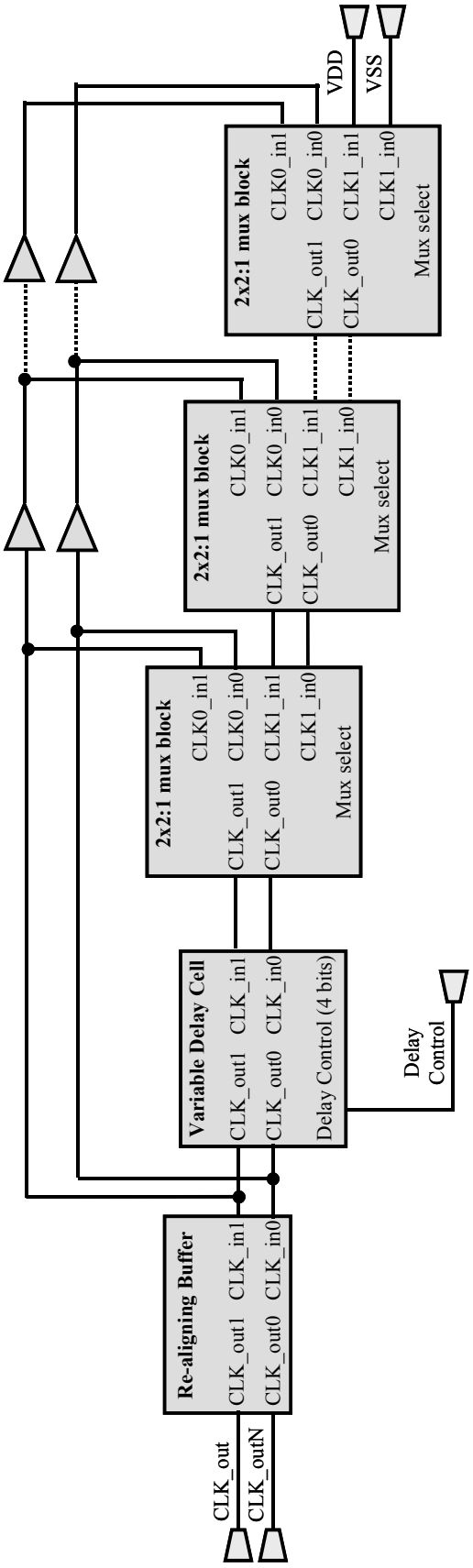

Fig. 5.11: Proposed architecture of the Digitally-Controlled Oscillator (DCO). Note that the Variable delay cell is inverting, while the Mux/Mask is non-inverting.

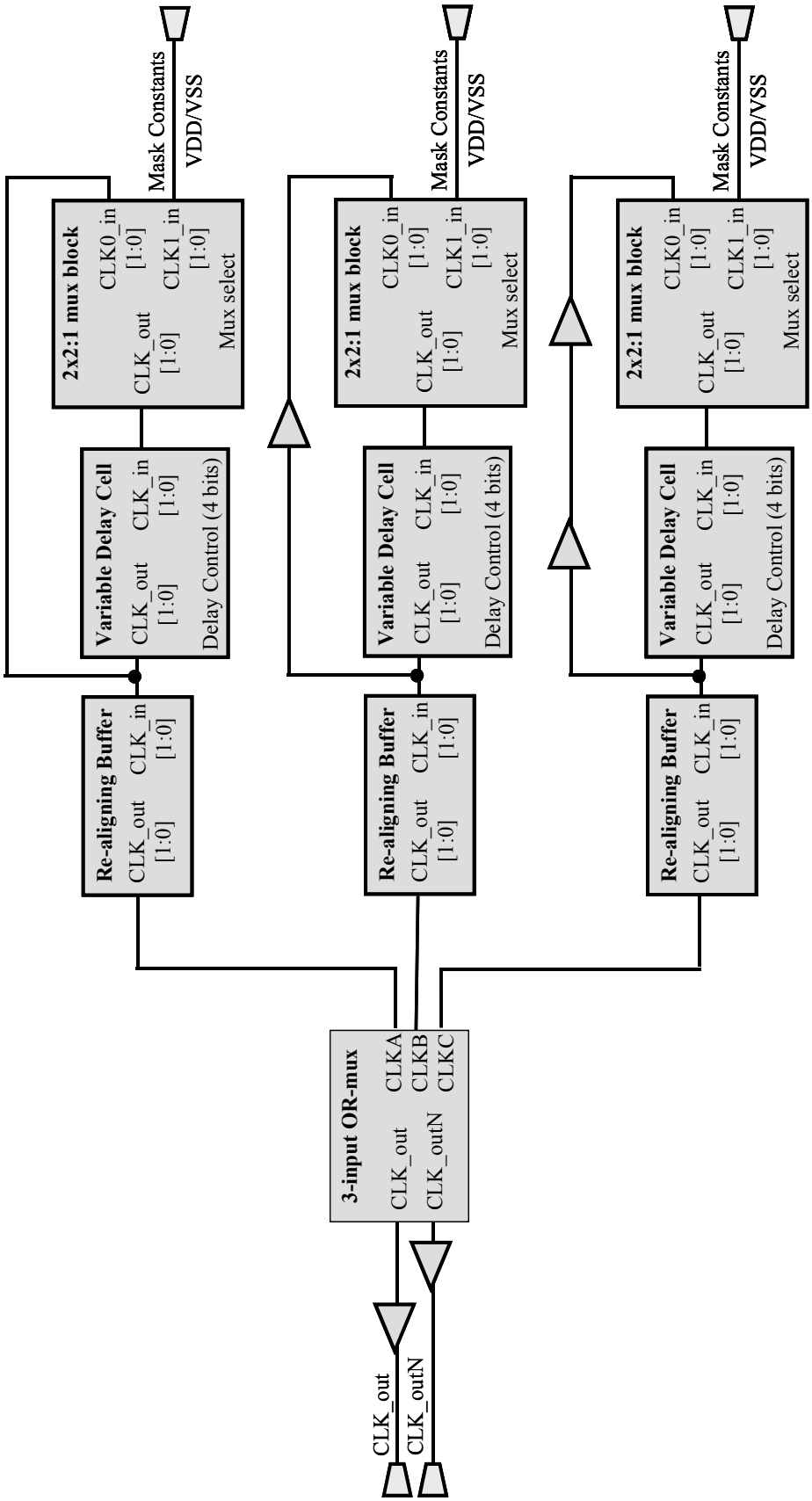

Fig. 5.12: Promising architecture for implementing DCO using three independent variable frequency oscillators with different ranges. The inherent presence of a clock mask allows us to use a 3-input OR-mux as the clock selector as in Chapter 4. Control bits have been omitted for clarity.

5.7 SUMMARY

This chapter presents a versatile Digitally-Controlled Oscillator (DCO) that is capable of producing complementary output clocks with a current-starved ring oscillator design. By utilizing a mutual exclusion element, this design is able to limit the amount of area required for the variable delay cell while still achieving adjustable frequency clocks with excellent dynamic characteristics including fast rise and fall times and near-50% duty cycles. While the current design of the DCO is non-linear in its frequency range, specific controller settings can be ignored and the control NMOS transistors can be re-sized or increased in number to achieve a more linear output frequency spread. Even though a more linear frequency range can be achieved with little additional effort, the DCO will always suffer from lower frequency resolution than a conventional analog VCO due to the digitally-controlled nature of the design. Even though DCOs are generally immune to input control noise, their clock output nonetheless suffers from more jitter than their analog counterpart [7]. However, the fast lock times and the smaller area of a DCO are all benefits that cannot be overlooked in comparing it to a conventional VCO. The versatility, simplicity and small area achieved by the asymmetrical current-starved architecture DCO developed here are all elements that combine to make this design particularly attractive compared to other digitally-controlled oscillator architectures in use today.

5.8 REFERENCES

[1] Kim, Nam-Guk and In-Joong Ha, "Design of a new Clock Recovery ADPLL for Disk Drives," IECON, pp. 1204-1209, vol. 3, 1999.

[2] Cheng, Li Jin and Qiu Yu Lin, "The performances comparison between DLL and PLL based RF CMOS oscillators," ASIC, pp. 827-830, 2001.

[3] Almeida, Teresa M. and Moisés S. Piedade, "High Performance Analog and Digital PLL Design," ISCAS, pp. 394-397, vol. 4, 1999.

[4] To, Cheuk-Him, Cheong-Fat Chan, and Oliver Chiu-Sing Choy, "A Simple CMOS Digital Controlled Oscillator with High Resolution and Linearity," ISCAS, pp. 371-373, vol. 2, 1998.

[5] Chiang, Jen-Shiun, and Kuang-Yuan Chen, "The Design of an All-Digital Phase-Locked Loop with Small DCO Hardware and Fast Phase Lock," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 46, issue 7, pp. 945-950, July 1999.

[6] Abdollahi, S. R., S. Kiaei, B. Bakkaloglu, S. M. Fakhraie, R. Anvari, and S. E. Abdollahi, "An All-Digital Programmable Digitally-Controlled-Oscillator (DCO) for Digital Wireless Applications," ISCAS, pp. 101-104, vol. 4, 2002.

[7] Saint-Laurent, Martin, and Gabriel Patrick Muyshondt, "A Digitally Controlled Oscillator Constructed Using Adjustable Resistors," SSMSD, pp. 80-82, 2001.

[8] Jong, Jin-Jer and Chen-Yi Lee, "A Novel Structure for Portable Digitally Controlled Oscillator," ISCAS, pp. 272-275, vol. 1, 2001.

[9] Yunhua, Shi, Sheng Shimin, Liu Yue and Ji Lijiu, "Implementation of a 6.5 MHz 34-B NCO," ISSCC, pp. 205-207, 1995.

[10] Janiszewski, Ireneusz, Bernhard Hoppe and Hermann Meuth, "Numerically Controlled Oscillators with Hybrid Function Generators," IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, pp. 995-1004, vol. 49, no. 7, July 2002.

[11] Ertl, R. and J. Baier, "Increasing the Frequency Resolution of NCO-Systems Using a Circuit Based on a Digital Adder," IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing, pp. 266-269, vol. 43, no. 3, March 1996.

Chapter 6: Conclusion

In this thesis, we have looked at all the components required to implement a complete globally asynchronous, locally dynamic system (GALDS). Firstly, a low-power solution for inter-clock domain communication between independently-clocked blocks in large ASICs and SoCs was discussed. These structures included a unidirectional and a bi-directional synchronizer that performed clock domain conversions through an intermediate asynchronous stage. While there was some overhead in implementing the bi-directional datapath in the method described, this overhead was easily justified by the obvious resource sharing benefit that this scheme enabled. This benefit was especially apparent for wide data busses that transport data over long distances since this scheme effectively halved the number of data lines required. In addition, the total throughput of the system was significantly better than that of a unidirectional FIFO of similar architecture. Without circuits like the ones described in Chapter 3, the inherent advantages of a GALDS system such as low power operation, high performance operation, reduced EMI and higher tolerance to clock skew could not have been achieved.

Next we looked at an all-digital local clock generator for dynamic frequency scaling within each synchronous block. The clock generator used a set of dividers operating on independent base clocks and a clock selector to glitchlessly switch between the

outputs of these dividers on demand. Power consumption is a key issue in integrated circuit design today and the dynamic voltage and frequency scaling that can be achieved in part due to this dynamic clock generator can result in significant power savings. Traditional methods of performing dynamic frequency scaling are flawed since they require significant silicon area and utilize many analog components that do not easily move from one fabrication process to the next. The all-digital clock generator presented in Chapter 4 is easily parameterized and thus portable to a variety of process generations. Having an active clock for as long as possible is important in dynamically clocked systems since frequencies can change often and this design managed to perform these transitions with a very short time overhead. The design switched frequencies on the order of nanoseconds instead of micro- or milli-seconds as is the case with traditional dynamic frequency generators. The speed, simplicity in design and ease of use of the all-digital clock generator combine to make the design a particularly appropriate solution for generating frequencies in multi-clock domain systems, especially when rapid, well-controlled, and glitch-free frequencies are required.

Finally, we looked at a versatile Digitally-Controlled Oscillator (DCO) that is capable of producing complementary clock outputs with a current-starved ring oscillator design. By utilizing a mutual exclusion element and an asymmetric variable delay cell design, the design was able to limit the amount of area required while still achieving adjustable frequency clocks with excellent dynamic characteristics including fast rise and fall times and near-50% duty cycles. Even though DCOs are generally immune to input control noise, their output clock nonetheless suffers from more jitter than their analog counterpart. However, the fast lock times and the smaller area of a DCO are all benefits that cannot be overlooked in comparing it to a conventional VCO. The versatility, simplicity and small area achieved by the asymmetrical current-starved architecture DCO developed in Chapter 5 are all elements that combine to make the design particularly attractive compared to other digitally-controlled oscillator architectures in use today. Should a more stable global clock reference be required, the DCO is versatile enough to be used in an all-digital PLL (ADPLL) replacing the VCO of a traditional PLL.

By combining an inter-clock domain communication structure, a local clock generator and a global clock generator, we have essentially created all the components required to build a GALDS system. The only thing required to implement a complete system are the local blocks themselves, preferably built with dynamic voltage scaling in mind. This system is capable of low power, high performance operation while reducing many of the common problems with present day integrated circuits such as clock distribution, clock skew and excessive heat dissipation. Overall, each component has been designed, created, simulated and shown to be effective for the application for which they were intended. While these three distinct circuits are designed to be used together in a GALDS system, they are versatile enough to not be limited to use in this architectural scheme. While this solution is by no means ideal in every respect, it does show that a GALDS solution can be practically implemented in today's technology for high performance operation without a prohibitive amount of design effort and without sacrificing a significant amount of silicon area since each design does not use a exorbitant number of transistors and all the transistors used are of reasonable size. There are many conflicting requirements in creating clocks and asynchronous structures using the methods described here in. The issues that arise are not necessarily obvious upon first glance and have solutions that are non-trivial in nature. Thus, going through the process of designing and implementing these blocks is of academic interest if only to discover the problems that can arise and all the potential paths by which these problems can be solved. However, the fact that all of these concerns can be balanced in a single circuit to achieve the designs presented here is a promising result for all-digital circuit structures that will inevitably become more popular as time goes on.