Dynamic Clock Management Circuits for Low Power Applications

Ian Brynjolfson, B. Eng. 1999

Department of Electrical and Computer Engineering

McGill Univeristy, Montreal

April 2001

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

0-612-75264-X

Abstract

Low power methods employing dynamically controlled clock rates offer potentially powerful energy saving capabilities. Dynamic clock management is a system level technique that benefits from the relationship between operating frequency and power consumption through the variation of clock speeds based on demand. This thesis provides robust and scalable clock management circuits for dynamic clock control. A complete Programmable Clock Manager (PCM) has been designed, implemented, integrated into a Systems-on-a-Chip and tested. The PCM incorporates two novel circuits who enable dynamic clock managment, the Range Shifting Phase-Locked Loop (RSPLL) and the Dynamic Programmable Clock Divider (DPCD). The self-calibrating RSPLL extends operating bandwidth while reducing jitter. Such a Phase-Locked Loop design also provides easier system-level integration and range-independent usage. The design is scalable to the higher bandwidths and lower voltages, associated with future technologies. The DPCD is capable of dynamic clock division without exhibiting glitches. Both of these robust circuits are easier to use for all clock managment purposes.

Résumé

Le contrôle dynamique de fréquence d'horloge est une méthode de basse puissance qui permet de potentiellement économiser de l'énergie. La fréquence d'horloge et la consommation énergétique d'un système sont directement interdépendantes. Le contrôle dynamique de fréquence d'horloge est une technique qui utilise cette dépendance en variant la fréquence du système selon la demande. Cette thèse présente un circuit de gestion d'horloge dynamique qui est robuste et facilement transférable d'une technologie à l'autre. Pour cette thèse un Gestionnaire de Fréquence Programmable (GFP) est développé, implémenté, intégré sur un "System-on-a-Chip", et vérifié. Le GFP incorpore deux nouveaux circuits qui permettent la gestion dynamique de fréquence, une Boucle à Verrouillage de Phase à Plage de Fréquence Contrôlable (BVPPFC), et un Diviseur de Fréquence Programmable Dynamique (DFPD). Le BVPPFC auto-calibré augmente la bande de fréquence d'opération tout en réduisant la gigue de phase. Une telle boucle à verrouillage de phase permet une intégration plus aisée, puisqu'elle est indépendante de la bande de fréquence d'opération du système. Ces circuits sont facilement transférables pour des utilisations de haute fréquence et bas voltage, comme le requiert la tendance des technologies futures. Le DFPD est capable de dynamiquement diviser la fréquence d'horloge sans produire d'interférence. Les deux nouveaux circuits proposés sont robustes et facilement utilisables pour tout système de gestion d'horloge.

Acknowledgments

I would first like to thank my supervisor Zeljko Zilic. He provided the direction, advice and attention to detail that propelled my work to a higher level of quality. I also appreciate the opportunity he gave me to pursue areas of research that interested me. I also want to thank him for sending me to several conferences to compare my work with others. His guidance has prepared me for this mysterious land they call, "the real world."

I thank Micronet for their financial support. Also, the Canadian Microelectronics Corporation (CMC) provided my with the unparalleled experience of fabricating custom chips through access to Taiwan Semiconductor's CMOS processes, free of charge. They also supplied a large share of the equipment used for this work.

There are several other people whom I would like to thank also. My parents have helped support me throughout this endeavor. I would not have made it this far without them. The other member of the Microelectronics and Computing Systems (MACS) lab have been indispensable throughout this work. Most notably, Nazmy Rasmi Danial Abaskharoun Paul Tadros (aka "the manual") seemed to know everything about anything you would need to know about all the tools and equipment in the MACS lab. I was impressed by his intensity. Another valuable asset is Arshan Aga for his technical help, advice and insight, both in the lab and at the pub. Mohamed Hafed also provided the additional experience and advice that comes with being a Ph.D. student. I would like to also thank all of the other members of the MACS lab, students and professors, that I had the privilege to work with: Henry Chan, Boris Polianskikh and Yanai Danan (the MCSoC team); Prof. G. W. Roberts, Prof. R. Negulescu, Prof. M. El-Gamal, Prof. N. Rumin, Sebastien Laberge, Lige "B" Wang, Ramez Rafla, Ahmed Mostafa, Yang Wu, Geoffrey Duerden, Naveen Chandra, Bardia "paper boy" Pishdad, Antonio Chan and Clarence Kar Lun Tam. I must also thank the system administrators who did so much with so little: Ehab Lotayef, Ben Mihailescu, and Weiwen Zhu. Also, thanks to Prof. J. Clark and Prof. F. Ferrie.

Contributions of Authors

The core of the original work on the development of dynamic clock management circuitry is located in Chapters 3, 4, and 5. The work in Chapter 6 is a collaboration of four students: Henry H. Y. Chan, Boris Polianskikh, Yanai Danan and the author of this thesis, Ian Brynjolfson.

Chapter 3 presents dynamic clock management in its application to Field Programmable Gate Arrays (FPGAs). The development of the patent pending Dynamic Programmable Clock Manager (DPCD) enabled dynamic clock management to occur in FPGAs, in a glitchless on-the-fly manner. This method of power reduction was not previously employed in FPGAs. The ideas are also outlined in [1], coauthored with Prof. Zeljko Zilic.

Chapter 4 presents a Programmable Clock Manager (PCM) design that is based, in part, on the PCM developed by Lucent for use in the ORCA Series FPGAs, as discussed in Section 3.1.1.3. The Phase-Locked Loop (PLL) design used in the PCM was based on the original design presented in [38]. The novelty of the PCM design comes from the use of the DPCD in place of the clock dividers in Lucent's original design. Also, the PLL design was adapted for the requirements of dynamic clock management. Using the PCM for dynamic clock management is a novel application for the circuit.

Chapter 5 describes the design of a Range Shifting Phase-Locked Loop (RSPLL). The concept of optimizing PLL design for clock management application by the automatic variation of the length of the Voltage Controlled Oscillator (VCO), is to our knowledge original. The substitution of the RSPLL in the PCM design for a standard PLL is novel also. The patent pending RSPLL was presented in [2], coauthored with Prof. Zeljko Zilic.

The work in Chapter 6 describes the contribution of the thesis author towards the development of the Managed Clock System-on-a-Chip (MCSoC) project. Table 3 in Chapter 6, outlines the contributions of the other students working on the MCSoC project.

Table of Contents

List of Figures

Chapter 1 - Introduction

A major challenge facing today's system and circuit designers is to create a new generation of products which consume a limited quantity of power. Increasing the challenge to design, power reduction must be achieved without compromising area or performance. Traditionally, the main design constraints were high speed operation and minimal area consumption. State-of-the-art products were driven by lower delays and smaller chip area. All of the design tools and methodologies were geared towards achieving these two goals. Power dissipation was an afterthought and not a design criterion.

1.1 - Motivation

At the end of the last century, power dissipation had become a main design concern in many applications. Two contributing factors were portable electronics and high-performance designs that exceed power dissipation limits. Battery-powered systems such as laptop/notebook computers, cellular phones and electronic organizers, to name a few, create a need for an extended battery life. Although the battery industry has been making efforts to develop batteries with higher energy capacity, a strident increase does not seem imminent. Even with advanced battery technologies, the life time of the battery is still small. Therefore, low power design techniques are essential for portable devices.

Low power design is not only needed for portable applications, but also to reduce the power consumption in high-performance systems. With large integration density and reduced gate delays, higher clock frequencies are becoming more easily attainable. Microprocessor fabrication has seen significant increases in costs associated with packaging, cooling and fans to remove heat. There are also reliability problems due to circuit failures at high on-chip temperatures.

The rising importance and concerns regarding energy consumption are manifested by the amount of attention being paid to this issue, from both academics and industry. This fervour focused on power dissipation has resulted in the publication of numerous textbooks and publications. Most publications discussing power optimization primarily center on circuit level techniques [5,6,7,8,9,10,11]. These techniques range from architectural configurations that consume less power, to circuits capable of operating at lower voltage levels [5,6,7]. Several publications are devoted to low power synthesis methods [12] or power analysis at various hierarchical levels [13].

Processor developers, such as Intel, Transmeta, and Motorola, have developed some means to reduce power consumption. They outline their power reduction methods in much of their documentation and product briefs [14,15,16]. Intel has a thermal monitor to interrupt the processor when excessive heat is present, and uses clock gating as their primary power reduction mechanism. The approach used by Motorola and Transmeta is to vary the frequency. Where Motorola has three states, doze, nap and sleep, Transmeta uses on-the-fly clock frequency changes to "dynamically pick just the right clock speed" [16].

1.2 - Dynamic Clock Management

Circuit level techniques and reduced voltage levels have resulted in great reductions in power dissipation from the gates and logic of a system, but the clock distribution networks have received less attention. The clock distribution networks drive the largest numbers of gates in a circuit, and alone consume substantial power. Also, the dynamic power dissipated by the clock increases as the number of clocked devices and the chip dimensions grow. In CMOS microprocessors, a global clock may account for up to 40% of the total system power dissipation [17]. To directly affect the power consumption in clock distribution networks, as well as in the system logic, *dynamic clock management* takes advantage of changing demand by supplying a variable frequency clock signal. Overall power is reduced without loss in performance.

Initially, most Integrated Circuits (ICs) had only one clock domain. Today, an increasing trend is to incorporate several independent systems in the same chip, each with their own clock domain. Most subsystems do not need to run at GHz frequencies to satisfy timing requirements. When many subsystems are incorporated into a larger system, they can be clocked at a power saving, reduced frequency. Possibly only one or a few subsystems need to be operated at maximal operating frequencies at any given moment.

In this thesis, we investigate dynamic clock management to better understand the large potential gains in power conservation by using this system-level technique. We also develop clock generation circuitry essential for performing dynamic frequency synthesis without endangering system operation. The clock generation circuitry must adhere to stringent requirements in order to be used in these applications where the clock frequency changes quickly and often. Transitions between frequencies must be fast and forestall glitching. The extent of possible frequencies should range from low idle speeds in kHz, to the maximum operating frequency of the system in the high MHz, or even GHz now. These traits should be exhibited by the on-chip dynamic clock generators, or *clock managers*, while producing nominal jitter.

1.3 - Outline

A brief introduction to dynamic clock management and its components is provided by Chapter 2. This includes a discussion of the relevant features, such as clock skew, jitter and the dynamic behavior of clock managers. The interesting low power benefits of dynamic clock management are demonstrated, followed by a brief discussion of its possible applications.

Our exploration into dynamic clock management begins with a Field Programmable Gate Array (FPGA) implementation in Chapter 3. The flexibility afforded by FPGA programmability leads to a high level of power consumption when compared to an equivalent Application Specific Integrated Circuit (ASIC) design. We see that existing FPGAs can perform dynamic clock management by including a novel clock divider in the user logic. The Dynamic Programmable Clock Divider (DPCD) is capable of performing clock division on-the-fly, without exhibiting negative effects such as glitches and inconsistent duty cycles. We also consider several of the methods and concerns when developing dynamic clock management as a program for reducing power consumption.

Although the DPCD enables dynamic clock management in existing FPGAs, a custom clock manager implementation would enlarge the clock frequency range. This range expansion would result in an increase in flexibility and thus increase the potential of dynamic clock management. A Programmable Clock Manager (PCM) was designed for this purpose and is described in Chapter 4. The PCM generates low jitter clock signals and includes DPCDs to satisfy the requirements of dynamic clock management. The PCM is also easy to use and a complete component. Therefore, the PCM may be placed in the clock distribution network along with controlling software, and does not require additional hardware for support.

The development of the PCM exposed several areas which have room for improvement. One of these areas consisted of the Phase-Locked Loop (PLL). A PLL with a wide operating range and limited jitter resulted in the contradictory design objectives. From these challenges, the Range Shifting Phase-Locked Loop (RSPLL) was developed, as seen in Chapter 5. This novel circuit is capable of supplying multiple operating ranges by changing the length of its Voltage Controlled Oscillator (VCO).

Last, in Chapter 6, we develop a research platform to advance exploration into dynamic clock management. A System-on-a-Chip (SoC) implementation includes a processor, a memory core, external communication subsystems, an internal asynchronous bus and the PCM to enable dynamic clock management.

Additional pertinent information is provided in the appendices. Appendix A develops the PLL model used throughout this document. The register configurations for the PCM are outlined in Appendix B. The numerical values for the measurements discussed in Chapter 4 can be found in Appendix C. Also, Appendix E shows additional simulations to support the claim of functional operation of the RSPLL. Finally, Appendix F contains the layout schematics for the printed circuit boards used during testing and debugging.

Chapter 2 - Dynamic Clock Management

With the increased speed of Field Programmable Gate Array (FPGA) and Systems-on-a-Chip (SoC) designs, there is more focus on the creation of high quality clock management schemes and the comprehensive exploration of their applications. Initially, clock management concentrated on solving distribution concerns relating to skew and waveform shape. Many benefits can be obtained by dynamically scaling the frequency of a system, including interesting low power possibilities [19]. Dynamically altering clock frequencies can be performed in several ways; if not done properly, disastrous results can occur.

2.1 - Clock Distribution

Global clocks are essential in the operation of synchronous digital systems. They define a time reference for the transfer of data throughout such a system. As a result, a great deal of attention is paid to clock distribution networks [20].

2.1.1 - Clock Distribution Concerns

There are many concerns when designing a clock distribution network, however two are of particular concern: clock *skew*, *jitter* and *hazards* (or *glitching*). An ideal clock is

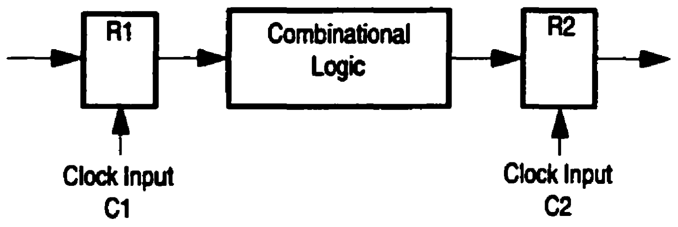


Figure 2.1: Clocked Data Path

distributed as a perfect square wave with matched transition times everywhere on a chip. Due to skew, this ideal clock is hard to achieve.

2.1.1.1 Clock Skew

Clock skew is defined as the difference in transition times of two disjoint clock signals:

$$T_{Skew} = T_{C1} - T_{C2} \tag{2.1}$$

where T_{C1} and T_{C2} are the times at which clocks *C1* and *C2* transition, respectively. The cause of skew depends on location. With regards to Inputs/Outputs (I/Os), the asynchronous nature of external data communication inherently leads to alignment difficulties. On the other hand, skew within a chip is caused by delay incurred during propagation of the clock signal from its input pin to the distant points of the clock distribution network. Therefore, the clock-driven cells within close proximity to the network origin will transition before those cells further along the network.

For registers and flip-flops to work properly, the following inequality must be satisfied:

$$T_{CP} > T_{Skew} + T_{Data} + T_{Setup} \tag{2.2}$$

where T_{CP} is the clock period, T_{Skew} is the clock skew, T_{Setup} is the setup time needed by the register for the incoming data to hold its state, and T_{Data} is the delay of the data path. The reduction of clock skew will directly influence the period of the clock and thus the speed of the design.

If clock skew is positive, then the clock signal at register R2 (T_{C2}) leads the signal at register R1 (T_{C1}), in Figure 2.1. Skew becomes the amount of time that must be added to the clock period in order for the second register to properly receive data. The solution is to simply slow down the frequency of the clock. In the case of negative clock skew, the clock signal at register R2 (T_{C2}) lags the signal at register R1 (T_{C1}). Negative skew can reduce the clock period, but the amount of allowed negative skew is limited. If there is excessive negative skew, then data will arrive at R2 before the clock signal. This implies that the data latched at R1 overwrites R1's past output before R2 had a chance to latch it. As a result, a *race condition* occurs. Slowing the clock frequency will not solve this problem. Only by ensuring that the skew is smaller than the data propagation delay will reliable operation be achieved.

2.1.1.2 Jitter

If skew is analogous to a DC condition, then jitter is considered to be AC. Skew is a consistent shift in phase by an amount $\Delta\phi$, shown over time in Figure 2.2(a). Whereas jitter is a fluctuation in frequency and phase around a desired value, or an inconsistent wandering of the clock edges, shown in Figure 2.2(b). The maximum jitter, T_{Jitter}, both positive and negative must be considered in design. If the tolerances do not incorporate jitter, the possibility of race conditions or the clock period being too short may occur. Therefore, Equation (2.2) must be altered to incorporate jitter.

$$T_{CP} > T_{Skew} + T_{Jitter} + T_{Data} + T_{Setup} \tag{2.3}$$

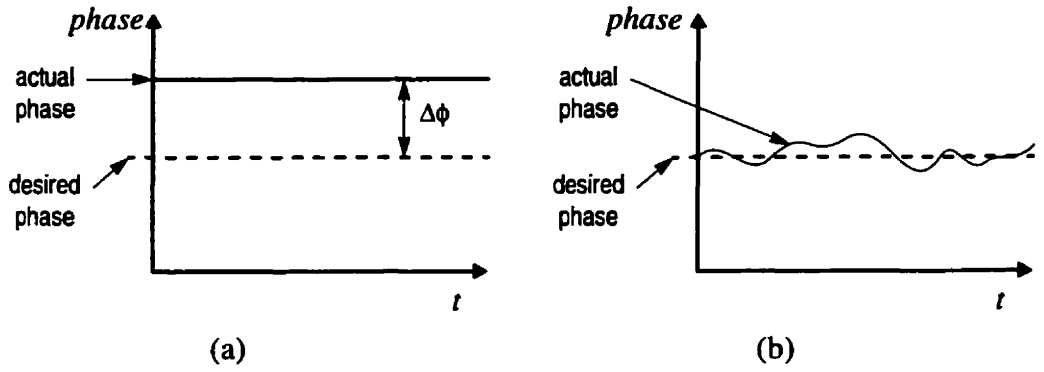


Figure 2.2: Time Domain Representation of Phase (a) Skew (b) Jitter

2.1.1.3 Waveform Deformation

Skew and jitter only affect the phase of the clock waveform; hazards and glitches can cause distortions. If the waveform becomes sufficiently distorted, as in Figure 2.3, additional edges may appear within a clock period leading to faulty latch attempts or metastable states. As a result, hazards and glitches are very dangerous in clock distribution networks. A duty cycle, defined as the time in which a clock signal is high during its period, of 50% is also desirable.

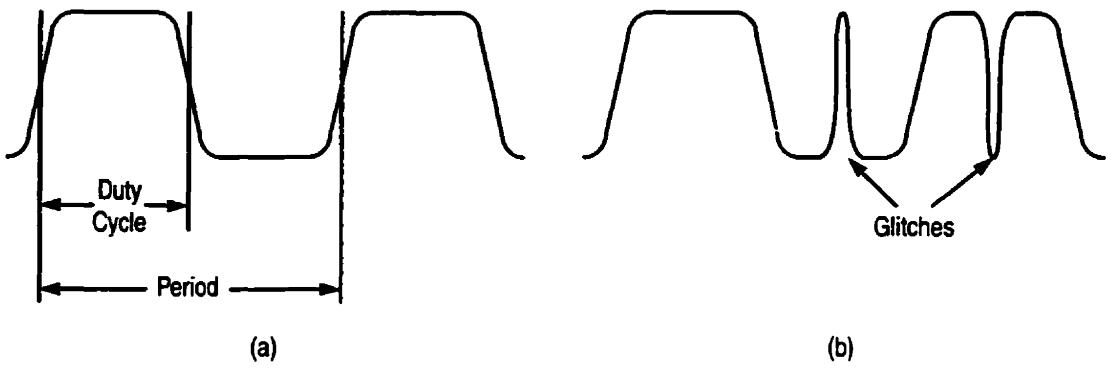


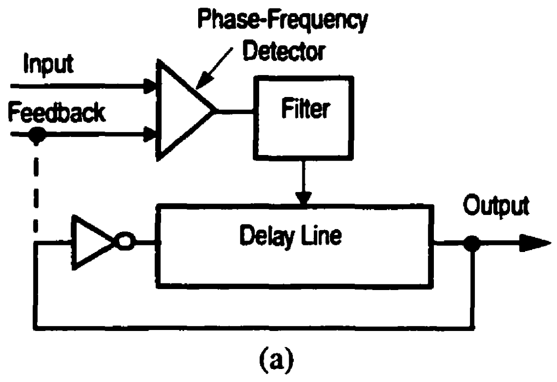
Figure 2.3: Clock Waveforms (a) Desirable Waveform (b) Waveform Containing Glitches

2.1.2 - Solutions

To reduce clock skew, one can utilize components that align clocks, implemented either as a Phase-Locked Loop (PLL) or a Delay-Locked Loop (DLL). Both circuits align the feedback clock edges, taken from any specific location in a system, with the edges of a reference clock. In clock distributions, the reference is either an external clock, or an internally generated clock. Figure 2.4 shows simplified diagrams of a PLL and DLL.

2.1.2.1 PLL and DLL Operation

In the PLL, the phase-frequency detector determines the degree as to which the reference clock and internal clock are out of phase, and whether the internal clock is leading or lagging the reference clock. The low-pass filter is used to average the phase detector output, and hence remove high-frequency oscillations. This average value is then used to



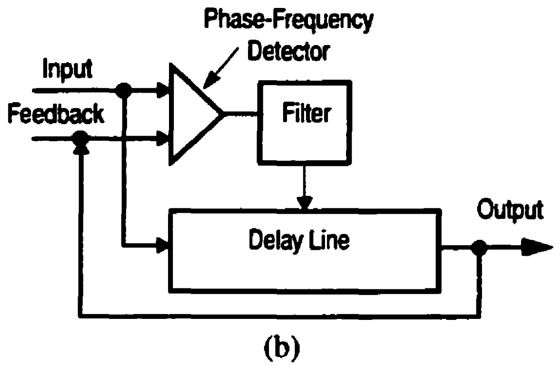


Figure 2.4: (a) Phase-Locked Loop (b) Delay-Locked Loop

increase or decrease the frequency of the controlled ring oscillator. The DLL employs a controlled delay line whose delay is equal to one clock period.

PLLs are capable of reducing jitter, or frequency variations, in the reference clock. However, the oscillator introduces noise and thus jitter from internal circuitry. On the other hand, since DLLs output a delayed version of the reference clock, it cannot remove jitter, but introduces limited jitter.

2.1.2.2 Skew Reduction

To reduce skew, PLLs and DLLs are introduced in the clock distribution networks. The feedback can come from any point on the chip; the clock edge at this point will be aligned to the input clocks. The dashed line between the PLL output and the feedback input in Figure 2.4(a) denotes this. In clock distribution networks, this feedback point should be selected to minimize overall clock skew while avoiding race conditions associated with negative skew between any points in the network.

2.1.2.3 Frequency Synthesis

An important feature of PLLs is frequency synthesis, i.e. multiplication of the input clock rate by a rational number. By placing a clock divider in the feedback loop, the PLL will increase the output frequency until it matches the divided feedback frequency. If the feedback is divided by N, then the output clock rate will be multiplied by N. By simultaneously dividing the clock rate of the reference or output clock by *M*, a clock ratio of *N/M* can be achieved.

2.1.2.4 PLL and DLL Comparison

In the case where the reference clock is noisy or clock recovery is needed, a PLL is used. When choosing a circuit to use as a clock buffer or for skew reduction, a DLL is the best choice because it introduces limited jitter and is inherently stable, thus less complex and easier to design. However, a DLL is not used for frequency synthesis and is less desirable for applications involving varying frequencies.

2.2 - Low Power Using Dynamic Clock Management

There are four areas to consider when reducing the power consumption of systems implemented in an FPGA, an Application Specific Integrated Circuit (ASIC) or an SoC. Dynamic power consumption (P_d) is given by the equation:

$$P_d = kV_{DD}^2 C_L f \tag{2.4}$$

where V_{DD} is the power supply voltage, C_L is the load capacitance, f is the frequency of switching and k is a switching activity factor. Low power techniques attempt to reduce the power supply voltage, the load capacitance, or switching activity in a circuit. One can also explore the direct proportionality between the operating clock rate and power consumption [21]. If the system clock is slowed to the minimum rate that will meet the computational requirements, both the instantaneous power and the overall energy consumption will be reduced without loss of performance.

While the reduction of power consumption happens whenever the switching activity is reduced, only asynchronous design techniques [22] can use that reduction to the full. In synchronous systems, the existence of large clock distribution networks diminishes the effects of switching activity reduction in logic gates alone. Hence, the power reduction in synchronous systems can be realized to its full potential either by clock gating, with the

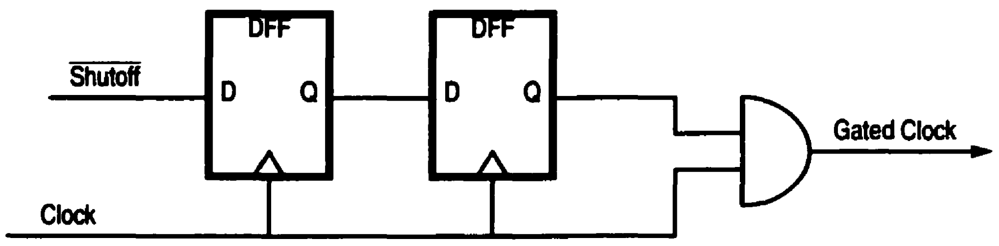


Figure 2.5: Clock Shutoff Circuitry used for Clock Gating

circuit shown in Figure 2.5, or by controlling the clock rate based on computational demand. While clock gating has been used widely, demonstrations of the latter technique for reducing power consumption have been presented only recently [23]. Studies [19,24] show that dynamic clock scaling is an efficient power reduction technique with large potential power savings.

Clock management circuitry capable of performing clock synthesis include programmable frequency multipliers and dividers. As a result, scheduling algorithms [19,25] can be used to control the division/multiplication of the system clock to meet the demand, without wasting energy. As the computational demand in separate subsystems changes, each subsystem can operate independently at a speed that will allow it to satisfy current individual demands, as shown in Figure 2.6.

Dynamic frequency scaling can be used in conjunction with other methods, such as dynamic voltage scaling. When the clock is slowed, a controlled voltage source can reduce the system voltage level and still maintain the switching speed. For comparison, dynamic voltage scaling does not bring any benefit in conjunction with clock gating.

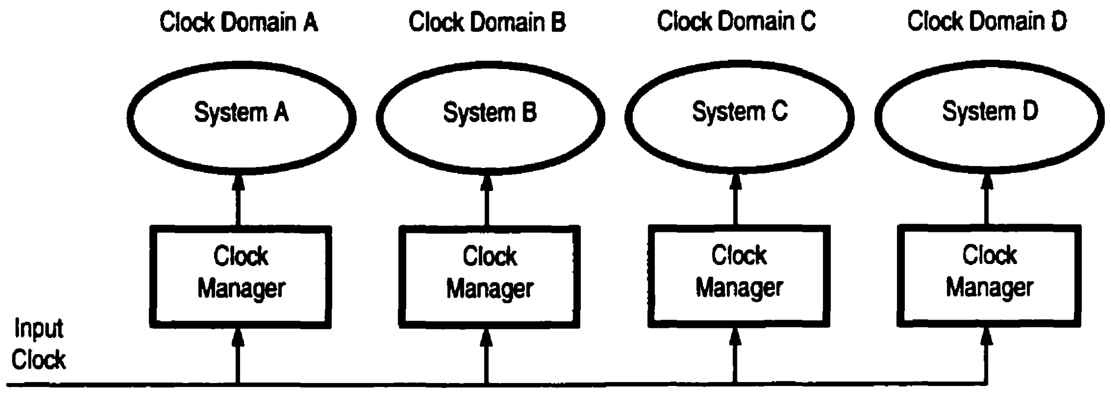


Figure 2.6: Embedded Application Using Clock Synthesis

2.2.1 - Voltage Scaling with Gating and Dynamic Clock Management

To better understand the benefits derived from voltage scaling and dynamic clock management in comparison to clock gating, consider the following example. Clock gating, represented in Figure 2.7, runs at the maximum frequency and full voltage for a specified period of time, represented in the time span from 0 to t_I. Then, the clock is gated and thus the frequency is dropped to 0. At this time, the voltage may also be shut off, however with the clock gated, $f = 0$ and $P_d = 0$, Equation (2.4).

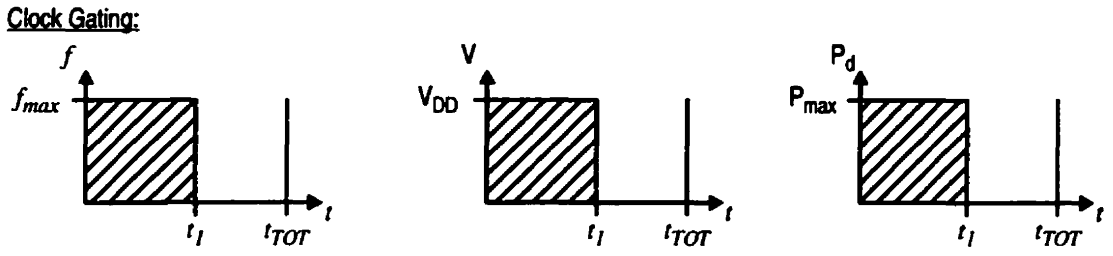


Figure 2.7: The Frequency, Voltage, and Dynamic Power Resulting from Clock Gating

For clock gating, the total power consumption over time is represented by:

$$P_{dCG} = \int_0^{t_{TOT}} P_d(t)dt = P_{max} \cdot t_1 = (kV_{DD}^2 C_L f_{max}) \cdot t_1 \tag{2.5}$$

On the other hand, when we consider dynamic clock management, the system does not operate at full capacity over the time span, as shown in Figure 2.8. Because of this reduced frequency, the supply voltage can be also be reduced.

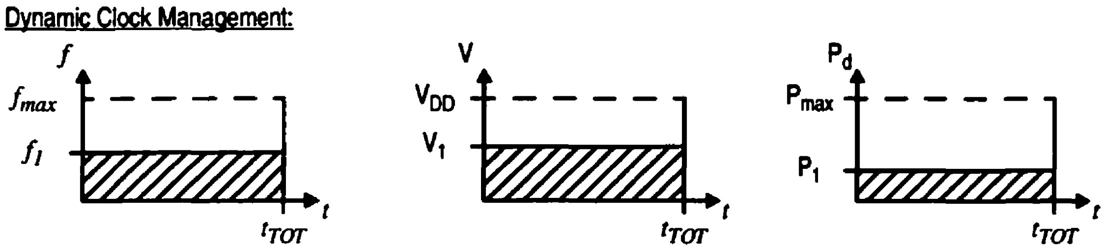


Figure 2.8: The Frequency, Voltage, and Dynamic Power Resulting from Dynamic Clock Management

In this case, for dynamic clock management, the total power consumption over time is:

$$P_{dDCM} = \int_0^{t_{TOT}} P_d(t)dt = P_1 \cdot t_{TOT} = (kV_1^2 C_L f_1) \cdot t_{TOT} \tag{2.6}$$

However, we should note:

$$f_1 = f_{max} \cdot \frac{t_1}{t_{TOT}} \tag{2.7}$$

So Equation (2.6) becomes:

$$P_{dDCM} = \left(kV_1^2 C_L\left(f_{max} \cdot \frac{t_1}{t_{TOT}}\right)\right) \cdot t_{TOT} = (kV_1^2 C_L f_{max}) \cdot t_1 \tag{2.8}$$

Since $V_1 < V_{DD}$ then $P_{dDCM} < P_{dCG}$. Therefore, voltage scaling performed concurrently with dynamic clock management has greater power savings than clock gating.

2.3 - Dynamic Clock Management Implementations

There are three scalable implementations of dynamic frequency scaling needed to perform the proposed low power methods. The simplest dynamic clock management circuit is an open-loop implementation with a clock divider inserted into the desired paths, shown in Figure 2.9(a).The maximum desired clock rate must be input to the system, which is then

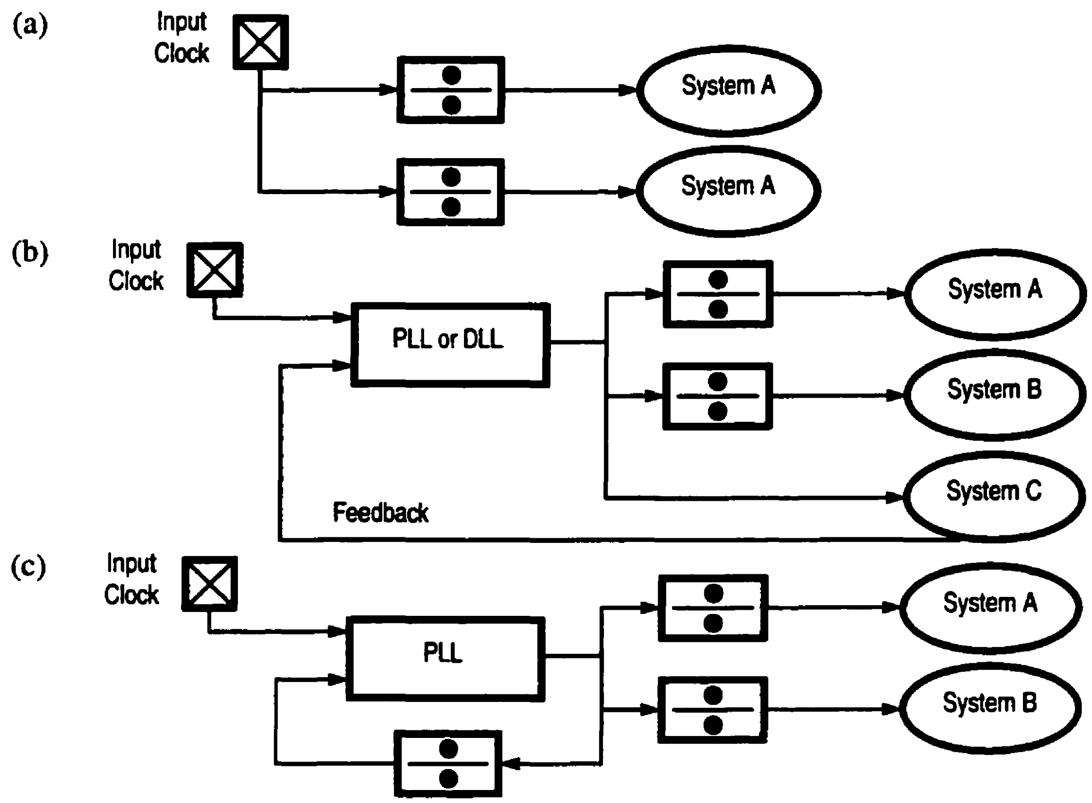


Figure 2.9: Dynamic Clock Management Implementations

reduced dynamically using the clock divider. This circuit is inexpensive and simple, but cannot compensate for clock skew and may introduce excessive additional skew.

Skew can be compensated by introducing a Phase-Locked Loop (PLL) or Delay-Locked Loop (DLL) into the circuitry. The simplest dynamically scaled structure is obtained by taking feedback from a point that does not change frequency, as shown in Figure 2.9(b). This scheme can successfully apply dynamic clock division. For dynamic multiplication, the signal in the feedback path must be divided, as in Figure 2.9(c).

2.3.1 - Frequency Changes and PLL Response

In the case of a large change in input or feedback frequency, the second-order step response of an underdamped PLL will exhibit the characteristics shown in Figure 2.10. If the overshoot of the lock frequency exceeds the maximum operational capabilities of the synchronous system, it may result in latching errors and possibly critical failures. The circuitry dependent on the output clock should then be stalled, or gated. To avoid an invalid output, the frequency transitions of the input of feedback path must be guaranteed to happen within the lock range [26]. If the PLL is designed for an overdamped step response to eliminate the possibility of an overshoot, the clock will not have to be gated, but will take longer to lock on to the new output frequency.

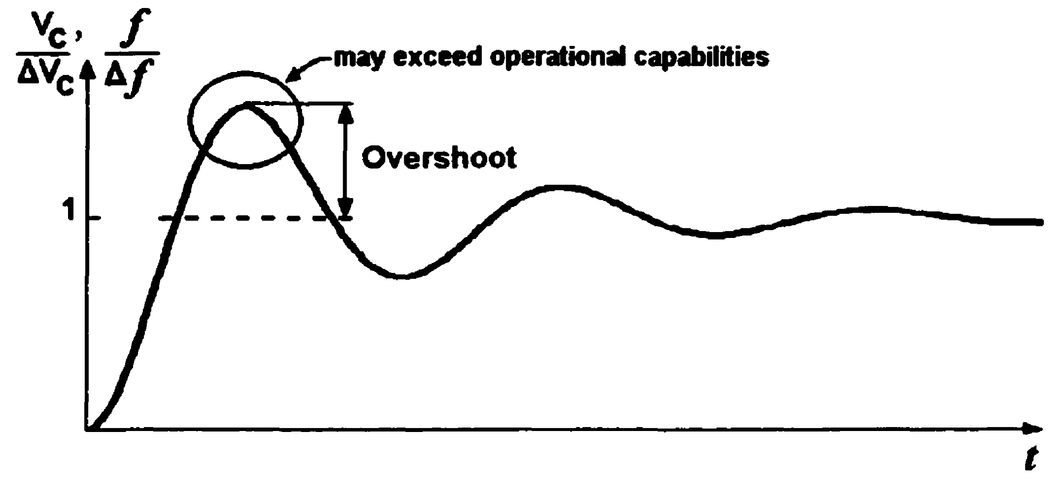


Figure 2.10: Step Response of Underdamped Second-Order System

Also, oscillations in the step response indicate that the output clock from the PLL may take some time to settle and regain a lock on the input signal. During this time, the output clock will not be properly aligned. In many cases, this frequency oscillation will not be of concern. However if the system is communicating with another that is expecting a known frequency, then communication should be temporarily suspended or the clock should be gated while the PLL is not in lock.

Appendix A discusses the dynamic behaviors and develops the mathematical formulae used to model the PLL. The design parameters are derived from this model and can be used to manipulate the PLL performance.

2.4 - Additional Applications of Clock Managers

Using a clock manager, the problems of clock distribution, mentioned in Section 2.1, are solved with the PLLs and DLLs. The FPGAs and SoCs now have built-in methods of clock skew reduction, frequency synthesis and phase alignment. If the clock manager is able to connect its output and feedback to I/O pads on the device, it can be used to synchronize with other devices and reduce skew on the board as well.

Frequency multipliers and dividers inside clock managers can be used for various applications of clock synthesis. By multiplying the clock rate, parts of the design inside Integrated Circuits (ICs) can be clocked at a rate higher than the board frequency. Multiple clock frequencies can be obtained by using several clock managers, or dividers. Similar to the current microprocessor based systems that run internal circuits and the I/O bus clock at different speeds, proper clock management is critical to realizing large systems on and off chip.

Time-domain multiplexing, that alternates between multiple sets of inputs, has been often considered in ICs for I/O multiplexing. The common design approach has been to use reduced frequency clocks with different phases. With a clock manager, both frequency division and phase shift can be programmed. Other applications, like duty cycle manipulation, can be programmed as well.

2.5 - Summary

Dynamic clock management and clock managers have many applications in FPGAs, SoCs and ASICs. They are used to solve clock distribution problems related to skew and jitter, but can also be used to reduce power consumption. Current low-power methods use circuit-based techniques at the penalty of performance. However, the system-level technique of dynamic frequency scaling can be used to reduce the average power consumption over time, while meeting the demands placed on the system. The basics of power reduction using dynamic clock managers have been presented in this chapter. Basic dynamic clock management solutions and alternative architectures have been outlined. The dynamic behavior of clock generation circuitry and its impact to proper functioning of dynamic clock management has been explained.

Chapter 3 - FPGA Implementation

Until recently, Field Programmable Gate Array (FPGA) designs have focused on optimization for performance and area [27,28]. Due to their programmability, FPGAs are inherently more energy-inefficient than the equivalent Application Specific Integrated Circuit (ASIC) implementation. The research on reducing FPGA energy consumption has been directed at interconnect [27], the combinational logic architecture, and the clock distribution network [29]. In commercial FPGAs, effort was directed at operating the FPGA core at a reduced voltage level. The outlined approaches lead to power reduction at the expense of the interconnect speed. An alternative to these circuit-level techniques is the system-level technique of dynamic clock management.

3.1 - Dynamic Clock Management in FPGAs

Due to the existence of increasingly large clock distribution networks in synchronous systems, reducing switching activity in logic gates has a diminished effect on reducing overall power consumption. This phenomena is especially present in FPGAs because of large capacitances associated with routing switches, markedly for global signals such as a clock [30]. The actual loads on signals are not known in advance, causing buffers to be sized for the worst case scenario. Also, FPGAs achieve flexibility by adding many routing pass gates. These high capacitances can cause approximately 21% of the overall dissipation of energy in an FPGA to be generated by the clock signal. This ratio can rise

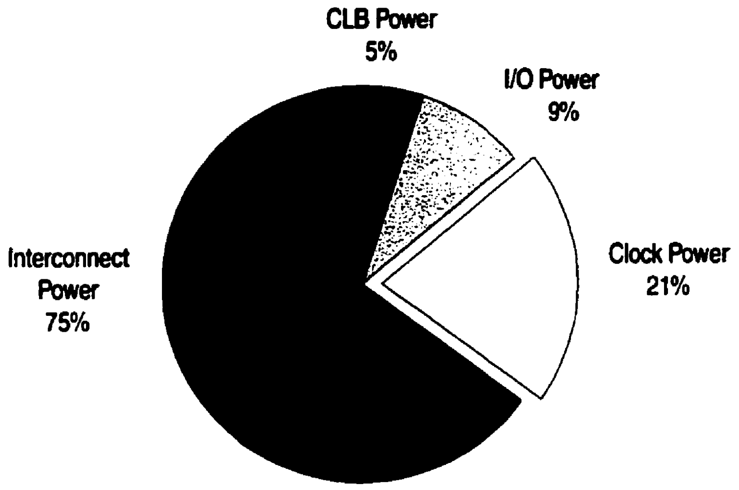


Figure 3.1: Typical FPGA Power Breakdown

above 50% in highly pipelined circuits [31,29]. Dynamic clock management can have a strong impact on the reduction of power consumption in FPGAs.

3.1.1 - Examples of Existing Clock Managers

Most modern FPGAs have dedicated clock managers for solving high speed clock distribution problems in high density designs. They include Phase-Locked Loops (PLLs) and/or Delay-Locked Loops (DLLs) together with clock dividers and interface circuits. Several clock managers have been introduced in FPGAs. The three characteristic examples are by Xilinx [32], Altera [33] and Lucent [34].

3.1.1.1 The Virtex FPGA Series by Xilinx

In the Virtex FPGA series, Xilinx Inc. decided to focus on DLL implementations and against the inclusion of PLLs. The clock manager is broken into four DLLs, with each being able to drive two global clock networks. Delay in the DLL is exactly one to four clock cycles and is therefore synchronized with the input clock edges. As a result, clock buffer delay is eliminated. Additional features of the Virtex clock manager include phase shifts, clock multiplication, and clock division. The reference clock can be shifted by 0°, 90°, 180° or 270°. The clock can also be doubled, the only multiplication factor a DLL can produce. The clock may also be divided by 1.5, 2, 2.5, 3, 4, 5, 8, or 16. In order to multiply the clock by four, two DLLs can be used in series. The Virtex DLL can be programmed to exhibit a 50% duty cycle regardless of the input duty cycle. A lock signal will indicate when output clocks are valid. Until then, the output clocks can exhibit transients.

3.1.1.2 The APEX 20K by Altera

Altera's APEX 20K and APEX 20KE FPGAs have a clock manager with ClockLock and ClockBoost features. The former uses a PLL with a balanced clock tree to minimize on-chip clock delay and skew. The later provides clock multiplication. On the APEX 20K, this multiplication factor is two or four. In the enhanced version, the APEX 20KE, the clock rate can be multiplied by a factor of $m/(n \cdot k)$, where m, n and k are integers ranging from 1 to 16. The ClockShift features a programmable delay and phase shift for the clock. The output clock delay is programmable to within the range of 2ns leading or lagging. The phase shift can be programmed to 0°, 90°, 180° or 270°, similar to the Virtex FPGAs. The Altera clock manager also provides low jitter clocks to be used by external components. As a result, the clock manager can be used to synchronize with other components. The lock signal on the Altera APEX 20K and 20KE is set when the PLL is locked onto the input clock.

3.1.1.3 The ORCA Series 3 by Lucent

Another family with a highly functional clock manager is the ORCA Series 3. These FPGAs contain two Programmable Clock Managers (PCMs), Figure 3.2, per chip. The ORCA PCM can be used for clock skew reduction (internally and externally), clock delay reduction, duty-cycle adjustment, clock phase adjustment, and clock rate multiplication and division. Each of the two PCMs on the FPGA contains a PLL and DLL. In DLL mode, the PCM can implement phase adjustment, clock doubling, and duty cycle adjustment. PLL mode performs clock skew reduction and multiplication.

Unlike the aforementioned clock managers, the ORCA PCM can be reconfigured during operation. With a simple read/write interface internal to the FPGA, the ORCA PCMs internal resources may be manipulated. By writing to the registers, multiplexers S_0-S_{10} are

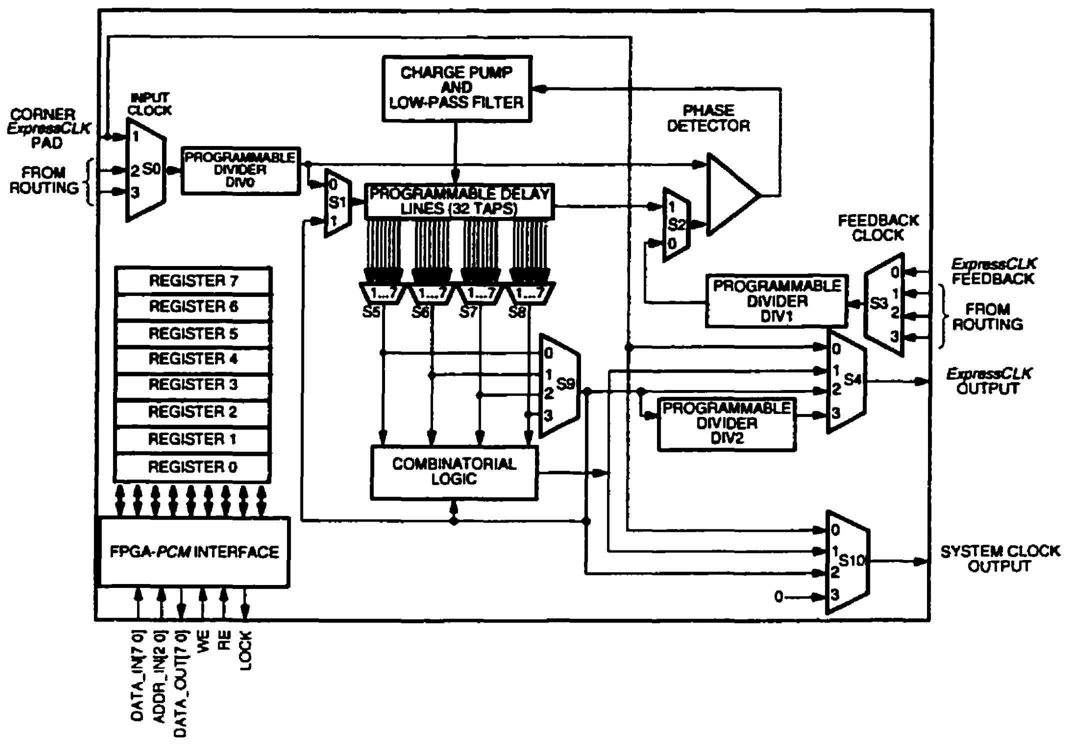


Figure 3.2: ORCA Programmable Clock Manager - Functional Block Diagram

directly manipulated to select PLL or DLL modes, as well as the clock dividers. The PCM also has a lock signal to indicate a stable output clock.

When incorporating the ORCA PCM into a design, the interconnection between the PCM and other components within the chip is highly flexible. There are dedicated pads for both an input clock and feedback, but general routing can be used as well. The dedicated paths are designed specifically for proper I/O buffering and low skew. Each PCM can drive two different, but interrelated clock networks. One output drives the Express Clock, specifically designed to distribute the fast clock signal, and the other system clock feeds the System Clock spine network through general routing. The output clock from the PCM can be connected directly to an I/O pad to drive external devices.

3.1.1.4 Limitations of Existing Clock Managers

Although they accomplish their intended function, current FPGA clock managers are incapable of performing dynamic clock management because their dividers cannot execute dynamic division or multiplication. The Xilinx and Altera clock managers can

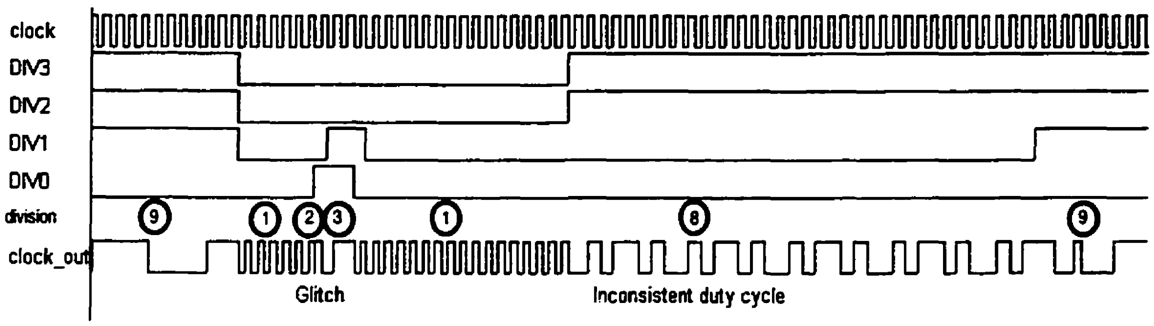


Figure 3.3: Standard Clock Divider Operation

only be programmed during initial configuration. The Lucent Programmable Clock Manager can be programmed during its operation, but this can lead to dangerous clock outputs if the clock is not gated.

3.2 - Dynamic Programmable Clock Divider

Changing the settings of existing clock dividers during their operation can lead to metastability and latching errors due to glitches, distortions, asymmetry, transient frequencies and additional clock edges of the output clock signal, as shown in Figure 3.3. Even shutting off the system during the change to the new frequency does not help in that case, as the inconsistent duty cycle clocks may be non-transient.

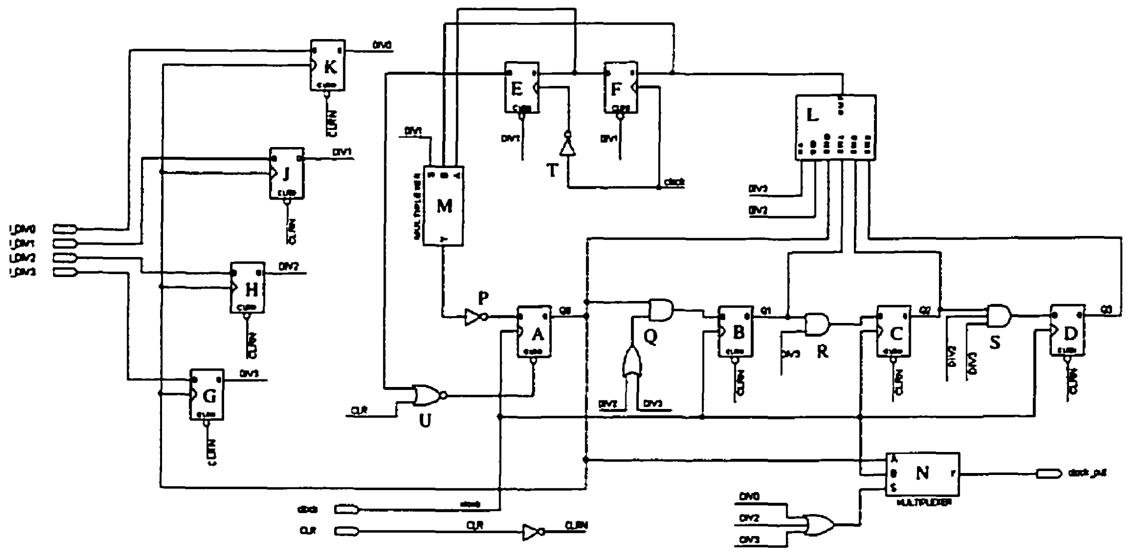


Figure 3.4: Dynamic Programmable Clock Divider

If dynamic clock dividers are added to the existing FPGA clock managers, they can be used for dynamic clock management. As a result, clean frequency changes can take effect within a clock cycle. The Dynamic Programmable Clock Divider (DPCD), of Figure 3.4, is capable of performing dynamic frequency division, for all integers from 1 to 9 inclusive, without undesired effects at the output.

3.2.1 - DPCD Operational Description

Division of the input clock is performed by creating a loop of rising edge triggered D-flip-flops {**A** - **D**} driven by the input clock, and feeding the signal back into the loop through an inverter {**P**} to create the necessary clock level inversion. The length of the output clock period is varied by using a multiplexer {**L**} to change the number of flip-flops in the loop. The design may be easily expanded to provide a division value greater than 9 by increasing the number of flip-flops that can be placed in the loop. In order to perform odd division, flip-flops {**E**,**F**} extend the loop by half a period with an asynchronous clear of flip-flop {**A**} on the falling edge of the input clock. Multiplexer {**M**} is used to switch between the output of multiplexer {**L**} during even division, and the output of flip-flop {**F**} during odd division. For the divider output, multiplexer {**N**} chooses between the original input clock for a division value of one or zero, and the output of {**A**} when the division value is 2 or greater.

3.2.1.1 Preventing Glitches and Preserving a 50% Duty Cycle

The output line of flip-flop {**A**} is labeled as Q0. One of the uses of line Q0 is to drive the clock inputs of flip-flops{**G**,**H**,**J**,**K**}. This implies that the value of I_DIV0, I_DIV1, I_DIV2 and I_DIV3 will be latched by flip-flops {**K**,**J**,**H**,**G**] respectively on the rising edge of the output signal generated by flip-flop {**A**}. As a result, the output lines of these flip-flops, labeled DIV0, DIV1, DIV2 and DIV3 will only change immediately after the rising edge of line Q0.

Between flip-flops {**A** - **D**} there exists combinational logic {**Q**,**R**,**S**} comprised of AND and OR gates. The purpose of this combinational logic is to cause a logical '0' to be input to the flip-flops in the chain that are not in the loop for the current division setting. They also prevent irregular oscillations in the circuit, and eliminate unnecessary switching. For example, in the case of a division by two, a feedback loop is formed consisting of flip-flop {**A**}, multiplexers {**L,M**}, and inverter {**P**}. Logic components {**Q,R,S**} will only allow a logical '0' to be placed on the inputs of flip-flops {**B,C,D**}. This case also holds true when the division setting is also 3. As a result, flip-flops {**B,C,D**} will not transition to a '1' during the time when the division value is two or three. When the division setting is a value of 4 or 5, the logic components at {**Q**} will allow the logical signal of Q0 to pass through to the input of flip-flop {**B**}. However, logic components {**R,S**} will only pass a logical '0'.

In order to understand the reasoning behind the DPCD design, an example of operation will be explored for division value settings of 1 and then transitioned to 8. As mentioned previously, when the division value is one or zero, the output of multiplexer {**N**} is the original input signal and there is no alteration to the reference clock.

The continued oscillation of the output from flip-flop {**A**} is important even though the output of the DPCD is not affected by this signal. This is to ensure that flip-flops {**G,H,J,K**} will still have an oscillating input clock so they can latch the new division values placed on I_DIV[3..0]. When the new division setting of 8 is placed on I_DIV[3..0], the value of DIV[3..0] will continue to hold the previous division value until the rising edge of the output of flip-flop {**A**}. By waiting until this point in the operation to update the logical values of DIV[3..0], the reference clock and Q0 will be a logical '1'. The new division value will cause the output of multiplexer {**N**} to be Q0 instead of the reference clock as previously output. Since both Q0 and the reference clock are '1', the output will

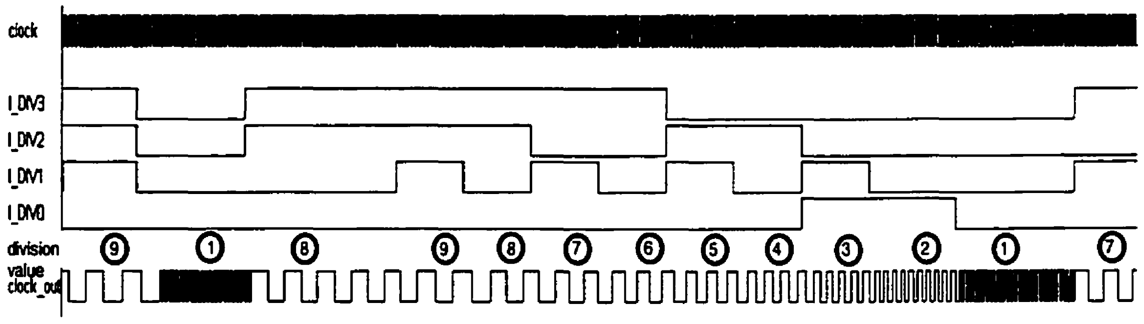


Figure 3.5: DCPD Operation

not glitch. Also, this will force transitions at the beginning of each clock cycle, thus preventing the output from transitioning before the clock cycle is complete and ensuring a 50% duty cycle at all times.

This procedure of changing from a division of 1 to 8 exposes the critical path of the DPCD. The output of Q0 must force flip-flops {**G**,**H**,**J**,**K**} to latch the new values of DIV[3..0] and update the selector settings of multiplexer {**N**} before the output of the reference clock becomes low. Otherwise, a glitch could occur.

Also with this new division value, the loop now consists of flip flops {**A**,**B**,**C**,**D**}, multiplexers {**L**,**M**}, and inverter {**P**}. Therefore, logic {**Q**,**R**,**S**} must allow the output of each flip-flop in the loop to reach the input of the successive flip-flop in the loop. If this logic had not been forcing a logical '0' during the divide by one, then the periodic oscillation of flip-flop {**A**}, toggling on each rising edge of the reference clock, would propagate through flip-flops {**A** - **D**} and then be captured and fed back into the loop when the transition to the new division value occurs (such as in Figure 3.3, division by 8).

3.2.2 - DPCD Benefits

As can be noted in Figure 3.5, all clock transitions are completely glitchless and require merely one clock cycle for the input division value to take effect. The duty cycle of 50% is maintained throughout operation, even during frequency transitions. Also, by preventing a logical '1' to propagate through flip-flops {**B** - **D**} when they are not included in the loop, unnecessary switching is prevented. This results in less power consumption

3.2.3 - Other Clock Dividers

Another method of dynamic frequency synthesis is performed with a chain of dividers capable of only even divisions, shown in Figure 3.6 and described in [21]. Each divider would statically divide the clock to each successive frequency, and a multiplexer would be used to choose the desired frequency. In Figure 3.6, each divider halves the frequency with a divide by two, or toggle flip-flop. There is no additional circuitry to ensure that the multiplexer changes the output frequency in a manner that will prevent glitches. Also,

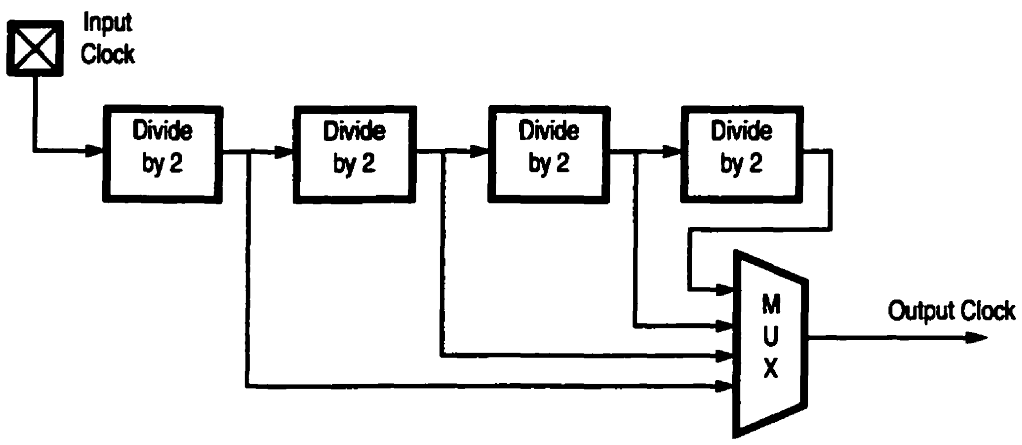


Figure 3.6: Frequency Synthesis Using Divider Chain

each delay stage in the divider will introduce additional skew equal to the propagation delay of the toggle flip-flop.

The design of a clock divider with the topology in Figure 3.6 results in an exponential divide ratio and the necessity for multiple static dividers. Also, the design does not shut off dividers that are not being currently used. However, the DPCD, being smaller and more power efficient, is a single circuit capable of performing divisions from 1 to 9. The DPCD design in Figure 3.4 is expandable to greater division ratios by simply adding more D-flip-flops in series.

Table 1: DCPD Performance

	ORCA or3t55-6 ‡	Altera EPF10K20-3 ‡	Custom♥
Size	4 PFUs	15 Logic Cells	13 DFF, 47 gates
Max Operating Frequency	251 MHz	76 MHz	1.2GHz
Power Consumption @ 25MHz	8.3 mW*	16.7 mW*	1.47mW

‡ compiled VHDL code with no specific optimization *Based on estimation formulae provided by manufacturers
♥ 0.18μm 1.8V CMOS TSMC process, capable of divisions up to 15, simulations can be found in Appendix D

The DPCD improves upon the clock divider in Lucent's ORCA PCM [35,36] in the fact that it is fast, glitchless and does not require a reset. The divider can directly replace the current divider in the ORCA PCM without modification to the internal structure. The DPCD can also be implemented in the existing logic cells of both the Altera and Lucent FPGAs.

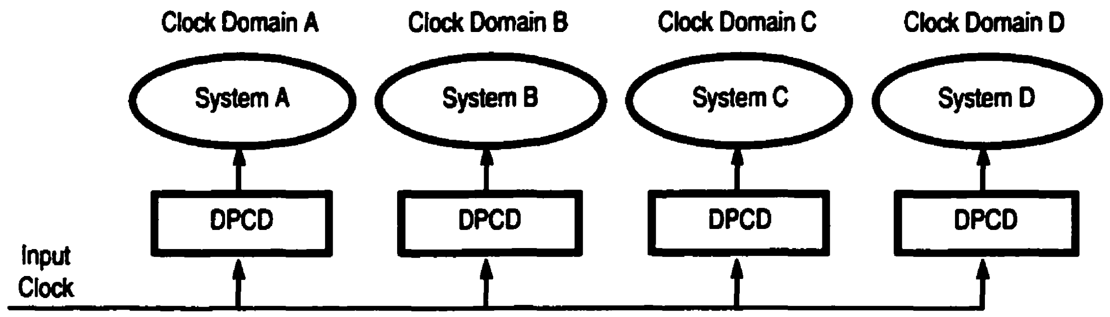


Figure 3.7: Embedded Application Using DPCDs For Clock Synthesis

3.2.4 - DPCD Applications

As possible use of the DPCD is depicted in Figure 3.7. Multiple DPCDs may be used to divide an input clock for separate systems. Each system operates at the minimum clock frequency that still meets the performance objectives of that particular system. Therefore, while some logic blocks have high demand and need to run at a high frequency, other blocks will less demand may run at a lower frequency and conserve energy. This method allows flexibility and gained power efficiency over a large system that is not partitioned to allow independent clock management. Without partitioning, the minimum clock frequency would be limited to the subsystem with the highest demand, thus resulting in energy waste in those subsystems having less demand.

The DPCD can also be used in a PCM. With DPCDs as the clock dividers, the PCM would be quickly reprogrammed to a new division or multiplication value without glitching or transient errors that could cause failure in the PLL. A PLL, including a DPCD in its feedback, can be used for clock multiplication, then several other DPCDs can divide the PLL clock signal for all of the subsystems, as shown in Figure 3.8. By using DPCDs in these systems, they can be quickly reprogrammed to a new division or multiplication value without glitching or transient errors that could cause failure in the PLL or subsystems. This increases the flexibility of dynamic clock management.

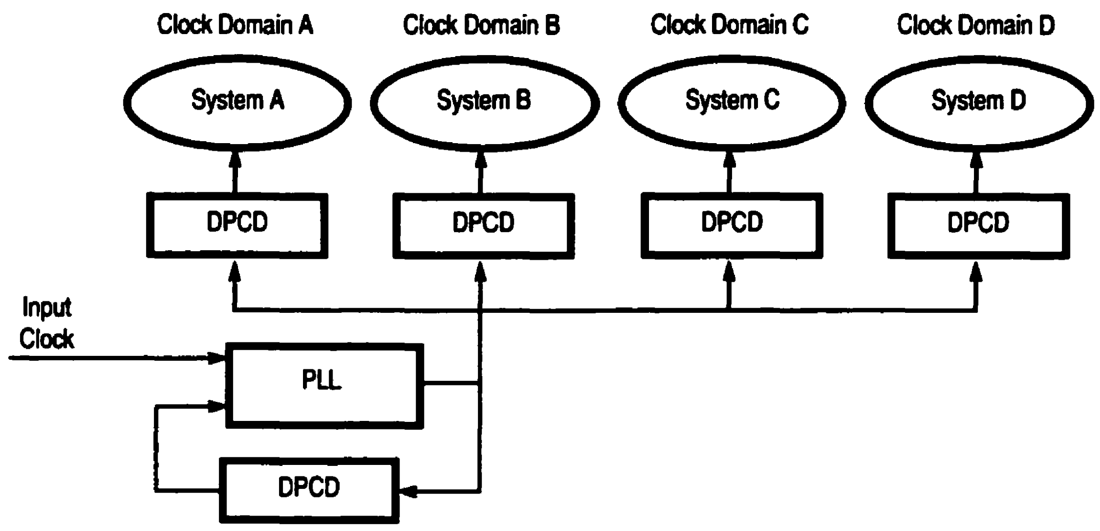


Figure 3.8: Embedded Application Using DPCDs In Clock Synthesis With a PLL for Clock Multiplication

3.3 - Partitioning Issues

Three issues arise from partitioning a large system into multiple subsystems for independent dynamic clock management. The first issue is the importance on each independent subsystem to perform a function that is not heavily temporally dependent on another subsystem. For example, say system B required data from system A every four clock cycles and it takes four clock cycles for system A to produce that data. Then running system A and B at different clock frequencies would produce either a constant wait for the data, or an over abundance of data that cannot be processed. However, system A and B may be dependent on data produced by the opposing system as long as the data may be accumulated and stored, or if the data is produced or required infrequently. In the case of an SoC, there are natural partitions already present in the integration of several systems that originally were completely independent. These systems should be partitioned so that they can communicate through a channel dedicated to passing data and a limited number of control signals. The communication channel may be common among several components in a bus structure, or may be dedicated with a standard or non-standard protocol.

3.3.1 - Communication

A second issue is the communication between those systems operating at independent frequencies may be accomplished. This issue can be solved with synchronization techniques used in many systems. One technique is to bring the systems, that wish to communicate, to the same operating frequency. If the PCMs and/or the DPCDs are designed for limited and constant skew, then the skew between the clock signals for both systems will be negligible. Therefore, communication can continue in a synchronous manner without metastability concerns. For this technique, data must be accumulated by each system for a large data transfer, or small data segments should be passed infrequently, so that the systems are not consistently running at the same frequency to pass data. Another approach is to use an asynchronous FIFO. Each system can communicate with the FIFO at its own operating frequency and therefore do not have to be matched during communication. In addition, synchronizer circuits may be employed, as discussed in chapter 10 of [37]. They can consume less space, but they are more complex and must be extremely robust. These circuits can pass data from one synchronous circuit to another even if they are running at different frequencies and do not contain coincident clock edges.

3.3.2 - Control Logic

The third partitioning issue concerns the control logic used to determine the setting for the current operating frequency. First, a circuit or method must be devised to determine the level of demand placed on the system. For example, in a microprocessor, this can be derived from the percentage of time running the idle process. Another method is to incorporate a FIFO on the input data stream that relays the number of data cells that are being used, thus conveying how full is the FIFO. Then based on the current frequency and the occupancy percentage of the FIFO, the next frequency setting can be computed. There are several published algorithms in existence that can compute the best frequency based on past demands and operating frequencies [25]. There are also publications concerning algorithms and systems that vary the voltage of a system based on demand, and similar methods can be applied to dynamic clock management [24].

A possible implementation of the previously mentioned scheduling algorithms is as follows. A FIFO is included in the input data path. After a specified time period, the demand of the system is measured based on the percentage of data cells occupied in the FIFO. If the number of cells that can be processed in the time period for a particular frequency is known, then the frequency setting for the next time period can be set to process all of the cells in the FIFO. After the time period, the situation is re-evaluated and a new setting is made. Another possible algorithm simply maps the number percentage of occupied cells in the FIFO directly to a frequency setting. As the percentage rises, the operating frequency is increased proportionally.

For added complexity and flexibility in the overall system, the control logic for each independent subsystem can be shared, or communicate with one another, in order to coordinate frequency settings or predict the future demand of each subsystem.

3.4 - Case Study

An embedded system coprocessor has been designed to speed up digital signal processing functions. The system takes advantage of dynamic clocking for reducing its power consumption. The system comprises a Universal Serial Bus (USB) client communication front-end and a reconfigurable signal processing unit. Here, we consider a matrix multiplication circuit. Power to the coprocessor is delivered by the USB cable from the host. Depending on the frame rate and data stream format, the bandwidth requirements of the multiplier will change. Therefore, the required clock rate is based on the bandwidth

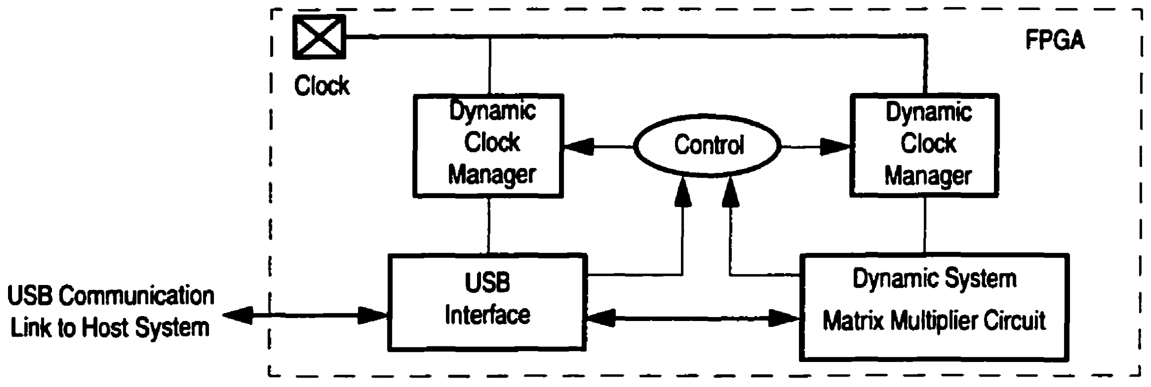

FPGA
Clock
Dynamic Clock Manager
Control
Dynamic Clock Manager
USB Communication Link to Host System
USB Interface
Dynamic System Matrix Multiplier Circuit

Figure 3.9: USB/Matrix Multiplier System

demand. Also, the standard protocol for USB includes two operating speeds that can be changed on the fly.

3.4.1 - Partitioning

The entire system was easily separated into the natural partitions of the USB client and the matrix multiplication unit. Each partition was assigned a clock domain and a DPCD in the user logic to divide the reference clock. The bandwidth of the USB client is the limiting factor in the operating speed of the matrix multiplication unit. With the advancement to USB 2.0 and the increased bandwidth the protocol provides, there is a large increase in the rate at which the communication client can transmit the data manipulated by the signal processing unit.

3.4.2 - Communication

Several ideas for the communication channel between the matrix multiplier unit and the USB client were explored. First, an asynchronous FIFO was designed to bridge the gap between the two subsystems. This design worked well, but in order to increase flexibility in the timing of the data transmissions to the USB host and allow for large burst transmissions, the FIFO had to be sufficient in size to accumulate large quantities of data.

A second method was to store the data in registers within the matrix multiplier unit and then match the frequencies of the two subsystems for a data transfer to a set of registers in the USB client. Each register consumed less area than the asynchronous FIFO due to the reduction in control logic, but the requirement for a register in each subsystem created an increase in area for the final design. This also consumed much overhead during the data transfers and limited the autonomy of each subsystem.

The third method takes advantage of the periodic alignment of rising clock edges from integer divisions of the reference clock. Both clock domains are generated with DPCDs driven by the same reference clock. Not only will rising edges be periodically aligned, but those rising edges that are not aligned will be spaced by at least one period of the input clock. Sample output waveforms demonstrating this method are displayed in Figure 3.10.

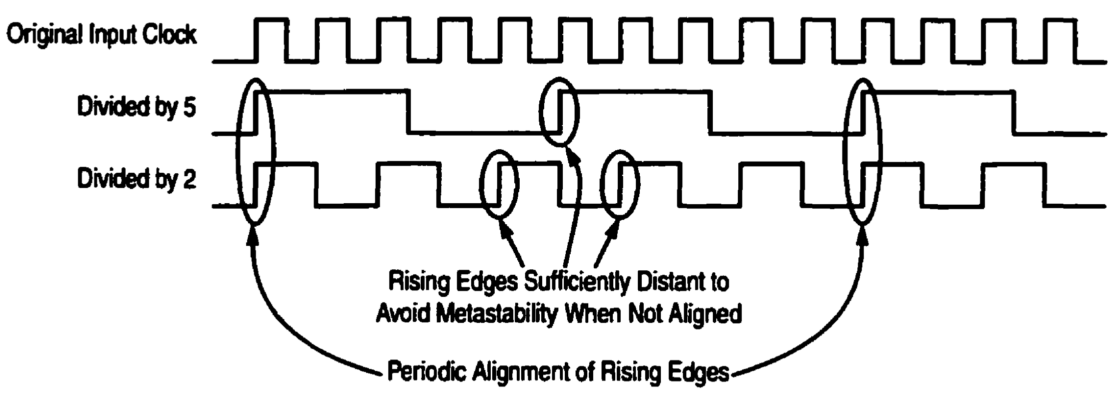

Original Input Clock
Divided by 5
Divided by 2
Rising Edges Sufficiently Distant to Avoid Metastability When Not Aligned
Periodic Alignment of Rising Edges

Figure 3.10: Output Clock Waveforms Demonstrating Periodic Alignment of Rising Edges When a Clock Signal is Divided by Integer Values

With the spacing of the unaligned rising edges, metastability errors will not occur due to violation of setup and hold times. If each subsystem knows its own speed and the speed of the subsystem it is communicating with, then all subsystems can be designed to latch data only when valid and to present new data after the previous data has been properly latched.

3.4.3 - Control Logic

Control logic receives information from the system subcomponents and updates the control registers of the clock managers. For the coprocessor, one control method uses a FIFO at the input of the data stream and the speed is determined by the quantity of occupied cells. Another method is to simply indicate the rate of incoming data and to set the speed accordingly. When a transfer from the USB host is preceded by a low speed preamble packet, the USB interface signals the control logic to reduce its clocking speed. The USB host will use the low speed to poll the client and receive information regarding the quantity of data to be transferred. The host then determines the communication speed. Most of the USB circuit is shut off after the final end-of-packet signal for that transfer is received. Full speed operation is resumed when communication recommences.

Table 2 shows power consumption of system components running at different clock rates for Altera and Lucent implementations. Figure 3.11 and Figure 3.12 show power consumption vs. operating frequency based on manufacturer estimates. The total power consumed for typical scenarios of system use is outlined.

Table 2: Implementation Details and Power Consumption*

	ORCA 3T55-6				
	f_{MAX}	size	P_{fMAX}	P_{12MHZ}	$P_{1.5MHz}$
USB	21.69 MHz	115 PFUs	228.4 mW	126.3 mW	15.8 mW
Multiplier	28.48 MHz	136 PFUs	326.5 mW	114.7 mW	22.9 mW
	Altera EPF10K20-3				
	f_{MAX}	size	P_{fMAX}	P_{10MHz}	P_{2MHz}
USB	30.48 MHz	703 LCs	1241 mW	519 mW	108.6 mW
Multiplier	17.12 MHz	563 LCs	584.14 mW	363 mW	112.6 mW

*Based on results and formulae provided by manufacturers

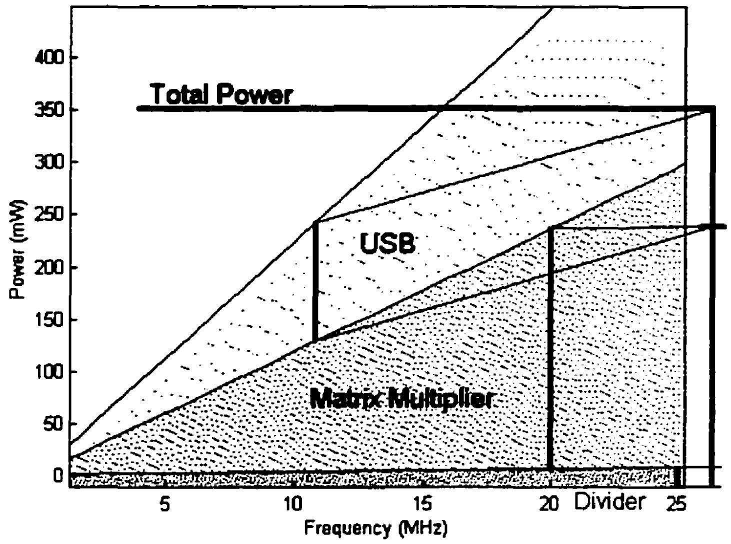


Figure 3.11: ORCA Implementation

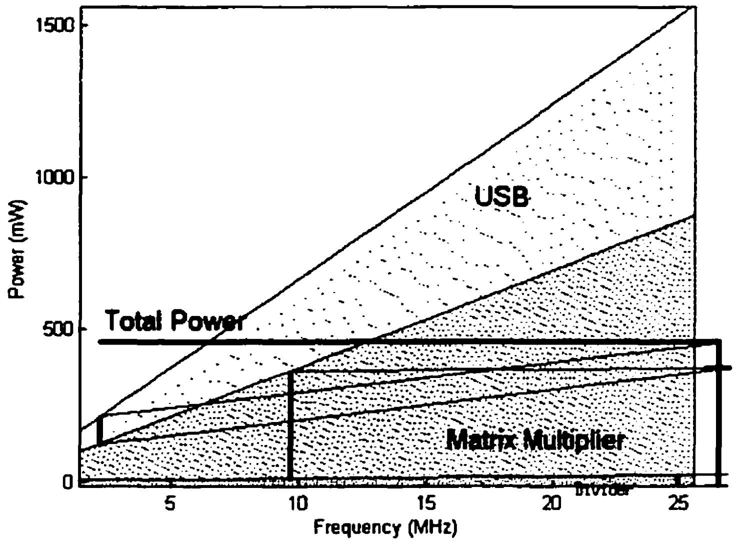


Figure 3.12: Altera Implementation

3.5 - Summary

Current low-power methods for FPGAs use circuit-based techniques at the penalty of performance. Dynamic clock management is a system-level technique that takes advantage of the clock managers present in most modern FPGAs. Dynamic frequency scaling is used to reduce the average power consumption. A dynamic programmable clock divider was presented that enables dynamic operation by eliminating glitches, transient and non-transient divider output errors that are very harmful when introduced in clock signals. The divider can be either included as a part of the FPGA clock managers, or as part of the user logic. The results show that this technique can be used efficiently in two FPGA families.

Chapter 4 - Programmable Clock Manager

The original development of *Clock Managers* was aimed at solving the problems associated with clock signals driving high speed, high-density designs. They include Phase-Locked Loops (PLLs) and/or Delay-Locked Loops (DLLs), together with clock dividers and interface circuits. With the growing requirements for reduced power consumption, designers should focus on using clock managers to perform dynamic clock management. In order to dynamically change operating frequency, the clock manager must be programmable and capable of modifying frequency on-the-fly, without glitches or deformations of the clock signal. A Programmable Clock Manager (PCM) was created with a PLL designed to exhibit these characteristics, in addition to reduced jitter and a wide operating range. The PCM was implemented in two iterations and the experimental results demonstrate the improvements.

4.1 - Functional Overview

A clock manager was developed to meet the stringent requirements for the clock generation circuitry used in dynamic clock management. With the functional block diagram shown in Figure 4.1, the Programmable Clock Manager (PCM) is capable of producing two independent variable rate clock signals. The control logic is fully

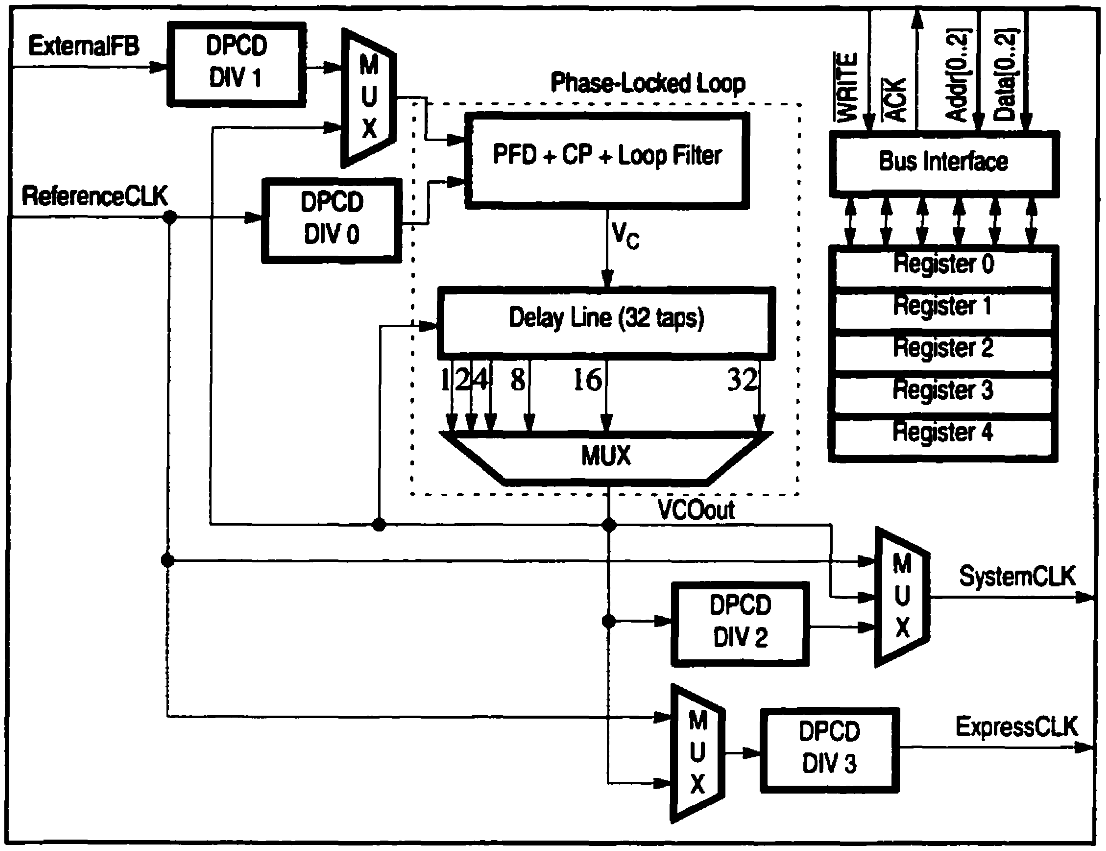


Figure 4.1: Programmable Clock Manager - Functional Block Diagram

programmable and can be changed instantaneously during normal operation. The proposed design can be scaled or modified to provide a larger number of clock outputs and incorporate other forms of PLLs. The PLL in the design can also be easily modified to function as a DLL.

The PCM contains four Dynamic Programmable Clock Dividers (DPCDs) capable of integer divisions between 1 and 15 inclusive, a charge-pump PLL with a 32 tap delay line Voltage Controlled Oscillator (VCO), selection multiplexers (MUX), a bus interface, and control registers. There are two input clock signals, reference clock (ReferenceCLK) and feedback clock (ExternalFB). Also, the PCM has two clock outputs, system clock (SystemCLK) and express clock (ExpressCLK). SystemCLK is intended for distribution across the entire system, where ExpressCLK is used for fast I/O distributions. The naming convention of the output clock signals is unimportant, but each clock output is programmed to generate a clock signal in different ways.

The VCO is designed to be programmable so the length can be varied. This flexibility changes the range of possible output frequencies generated by the PLL. Both the feedback and reference clock signals may be divided by the DPCDs described in Section 3.2. There is also a multiplexer to choose the source of the feedback signal, either from the output of the VCO, from a location found within the Integrated Circuit (IC), or from an I/O pad. The external feedback signal can also be divided in order to provide frequency multiplication.

SystemCLK is generated by the PCM in three different ways, as can be seen in Figure 4.1. The default setting bypasses the PLL and connects the SystemCLK output directly to the ReferenceCLK input. A second program setting connects SystemCLK to the output of the PLL. The final possibility is to pass the output clock signal from the PLL through a DPCD and then link it to SystemCLK. Thus, in a case of a divide by zero or one, the signal should bypass the DPCD to reduce the delay on the SystemCLK output. ExpressCLK only has two sources for its clock signals. The default, similar to SystemCLK, comes directly from the ReferenceCLK and a second setting derives the clock signal from the PLL output. However, the multiplexer used to select which signal drives ExpressCLK is placed before the DPCD. Therefore, the ExpressCLK signal is always passed through a divider. When the PLL is generating a very high frequency clock signal, this configuration allows the ExpressCLK to generate a very low frequency signal by dividing ReferenceCLK instead of the PLL output clock.

The circuitry for maintaining and updating the current programmed state in the PCM is implemented by four registers and a bus interface. These registers are each composed of eight Static Random Access Memory (SRAM) cells that are updated from the bus through address, data and write signals. The outputs from the SRAM cells drive select lines for all the programmable components in the PCM. By updating the registers, the program settings are changed immediately.

A complicated reset circuitry has been included in the design of the PCM in order to improve its functionality. If a system-wide reset occurs, having the clock generation circuitry settings change is not always desirable. Therefore, a global set-reset signal, capable of disabling the reset of the registers, is included in the PCM control registers.

Each DPCD also has an individual reset disable. Also, a register cell is devoted to storing a reset signal for the PCM. Therefore, an external system may induce a reset of the PCM to the default settings. In the default state, both output clock signals are connected to ReferenceCLK.

4.2 - Phase-Locked Loop Design

The most complicated circuit within the PCM is the charge-pump PLL, shown in Figure 4.2. The design is partly based on that presented in [38], but there are some changes to the original design, most notably in its application to dynamic clock management, and the structure of the VCO. The PLL is comprised of a Phase-Frequency Detector (PFD), Charge-Pump (CP), loop filter, and variable length VCO. The length of the VCO is set by the system through the control registers. Section 2.1.2 outlines the PLL operation. In this particular structure, changing the length could result in a glitch at the output. Therefore, a reset of the VCO and gating the output clock is recommended during length changes. The circuits are designed to have reduced jitter and to operate with a reduced supply voltage.

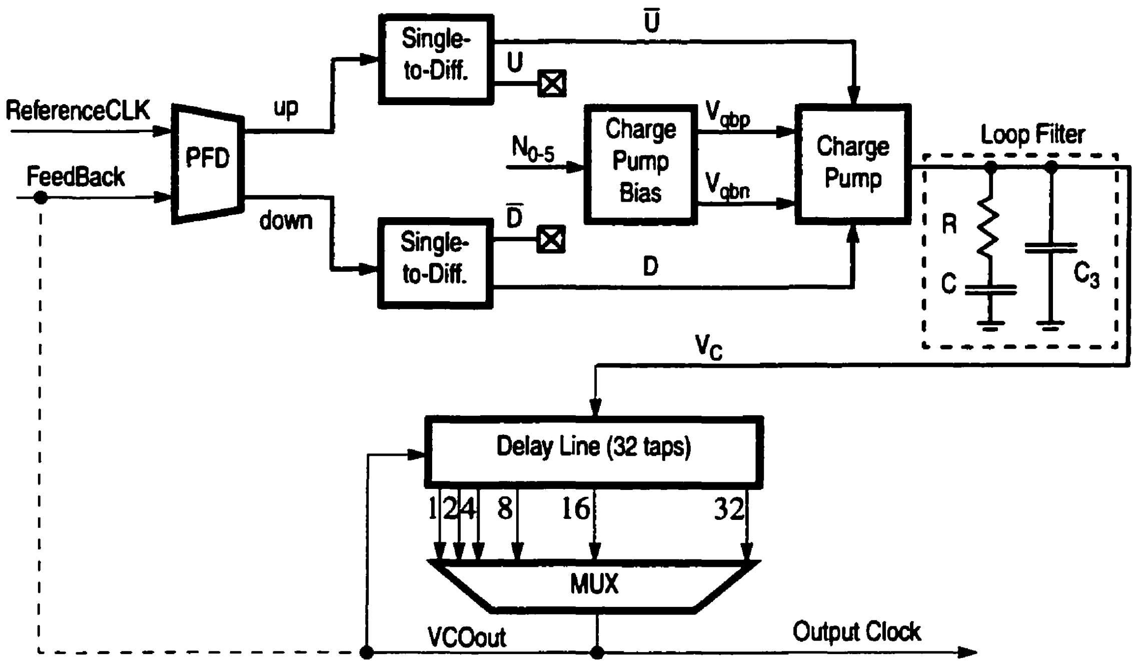


Figure 4.2: Phase-Locked Loop Block Diagram

4.2.1 - Voltage Controlled Oscillator Design

Low noise VCOs utilizing high-swing complementary signals have been presented and used in several PLL designs [38,39]. The delay cell design, shown in Figure 4.3, is an analog style differential pair, thus reducing jitter due to supply noise. PMOS diodes are used in the differential delay cell to clamp the output voltage to a minimum level of $V_{DD} - |V_{tp}|$. This clamping prevents the output swing and common mode, or the crossover point of the differential signals, from changing significantly with frequency. This gives the VCO a large frequency tuning range.

With the intention of providing low oscillator noise, the rise and fall times of the output nodes should be made equal [40]. This is accomplished by ensuring that both currents through the NMOS transistors of the differential pair are equal and opposite in direction.

A major advantage to the design of this delay cell is its ability to operate at a low V_{DD} voltage. The disadvantage of this delay cell design is the additional parasitic capacitance of the PMOS diodes that reduces the maximum operating frequency. The additional gate capacitance of the diode loads can be eliminated by using NMOS diodes [38,41].

4.2.1.1 Varying the Number of Taps

For the VCO design, the number of taps, or delay cells, in the delay line may be programmed to vary the output frequency operating range. The multiplexer can choose between 1, 2, 4, 8, 16 and 32 taps. In simulation, the delay through the multiplexer is

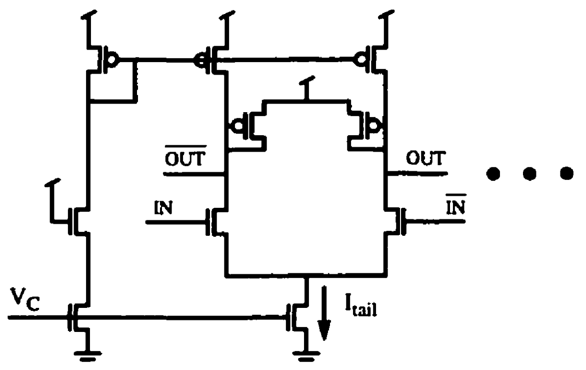


Figure 4.3: Differential Delay Cell

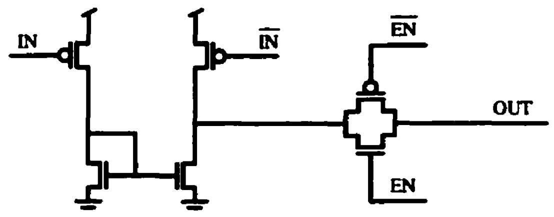


Figure 4.4: Differential-to-Single Ended Converter with Enabled Transmission Gate

sufficient that oscillation occurs with only one delay cell. However in practice, the VCO would not oscillate when programmed to this length.

In order to achieve high operating speeds, it might seem that a single-level multiplexer composed of differential-to-single ended converters with transmission gates, as shown in Figure 4.4 and Figure 4.5, would incur the least delay. However, the parasitic capacitances associated with connecting six PMOS drains and six NMOS drains together became

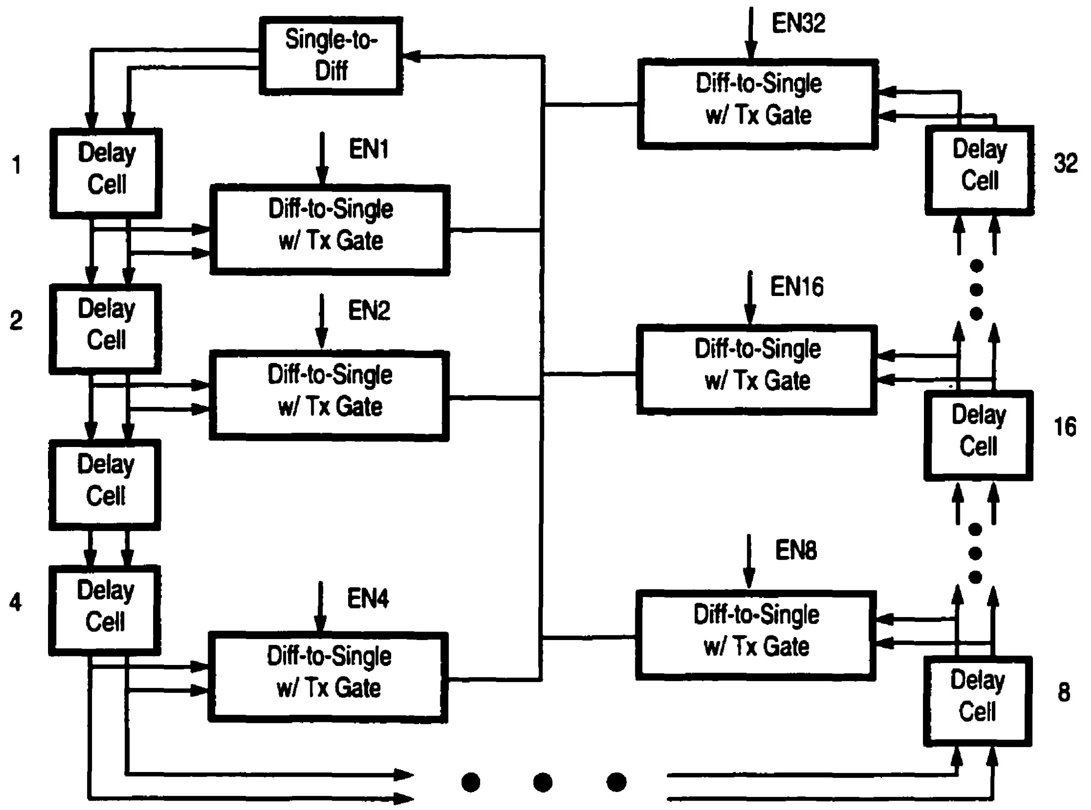

Single-to-Diff
EN32
Diff-to-Single w/ Tx Gate
1
Delay Cell
EN1
Diff-to-Single w/ Tx Gate
Delay Cell
32
2
Delay Cell
EN2
Diff-to-Single w/ Tx Gate
EN16
Diff-to-Single w/ Tx Gate
Delay Cell
16
Delay Cell
4
Delay Cell
EN4
Diff-to-Single w/ Tx Gate
EN8
Diff-to-Single w/ Tx Gate
Delay Cell
8

Figure 4.5: VCO with Transmission Gate Multiplexer Implementation

excessive and resulted in a slower design than a multi-level NAND/NOR gate implementation.

4.2.2 - Charge-Pump Design

When designing for the worst case, the PLL bandwidth and damping factor should be sufficiently distant from the stability limits for all possible variations in the manufacturing process, the reference clock frequency and the division factor in the feedback path *N*. Also, a high PLL bandwidth is beneficial for reducing peak-to-peak jitter due to VCO noise. In the application of a PCM, *N* can vary by an order of magnitude, causing variation in the stability constraint to be dominated by *N*. Without a static value for *N*, maximum noise suppression cannot be achieved.

4.2.2.1 Bandwidth and Peaking Compensation

The loop filter used in this PLL design is a typical series resistance-capacitance (*RC*) filter configuration with a smaller capacitor in parallel (C_3), as shown in Figure 4.2. Therefore, the stability limit of a charge-pump PLL is given by [42,38]:

$$f_{ref} \geq \frac{K_O I_p R}{N 4\pi} \tag{4.1}$$

where f_{ref} is the input reference frequency (or effectively the sampling rate of the phase detector), K_O is the VCO gain, I_p is the charge-pump current, and *R* is the loop filter resistance. Bandwidth and damping factor, along with several other PLL design parameters, also change with *N*.

A modification to the charge pump bias circuitry enables the PLL to always operate with a maximum bandwidth and fixed damping factor without endangering stability. This is accomplished by compensating loop parameters for changes in *N*, through setting the charge pump current to:

$$I_p = N \cdot I_{ref} \tag{4.2}$$

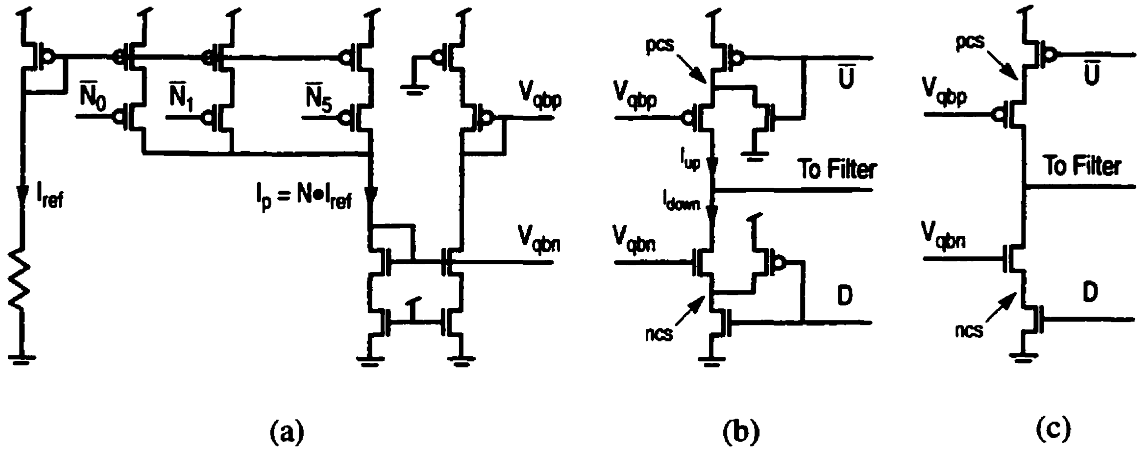


Figure 4.6: (a) Current Multiplier Generating Charge-Pump Biasing Voltages V_{qbn} and V_{qbp}
(b) Charge-Pump With Additional Transistors to Remove Charge Sharing
(c) Charge-Pump Without Transistors to Remove Charge Sharing

where I_{ref} is a fixed reference current set by a resistor. This relation is implemented by the current multiplier in Figure 4.6(a), which generates the charge-pump current I_p through control of weighted current sources. The programmability of I_p allows for adjustments to the gain of the charge pump if the gain of the VCO is inconsistent for various length settings.

4.2.2.2 Charge Sharing and Balanced Currents

Parasitic capacitances within charge pumps lead to the common problem of charge sharing. The circuit in Figure 4.6(b) was implemented in the PCM in order to combat this problem. When $\overline{U}$ is activated, the node pcs is charged to V_{DD}. In order to discharge this node after $\overline{U}$ is released, an NMOS transistor is used to create a path to ground. Otherwise, the charge stored at pcs will leak through the current source and into the filter. A similar effect can occur at node ncs, where a PMOS transistor is used to charge the node to V_{DD}. Since the parasitics at these nodes cannot be matched, a static phase offset would occur if the excess charge was not removed.

In many applications, a static phase offset would not be critical. However, the nature of this phase offset results in increased output jitter. When the PLL is locked, a leakage mismatch resulting from a larger current from pcs than ncs would be compensated by activating D earlier than $\overline{U}$. This forces I_{down} to flow longer than I_{up}, as shown in Figure 4.7. Since the compensation charge is applied in the early portion of the charge-pump activation time, a voltage ripple on the loop filter will ensue, as shown in Figure 4.8. This will eventually result in phase jitter at the output of the VCO. The charge removal transistors result in a well-balanced $\overline{U}$ and D activation time such that I_{up} and I_{down} cancel each other, as shown in Figure 4.9.

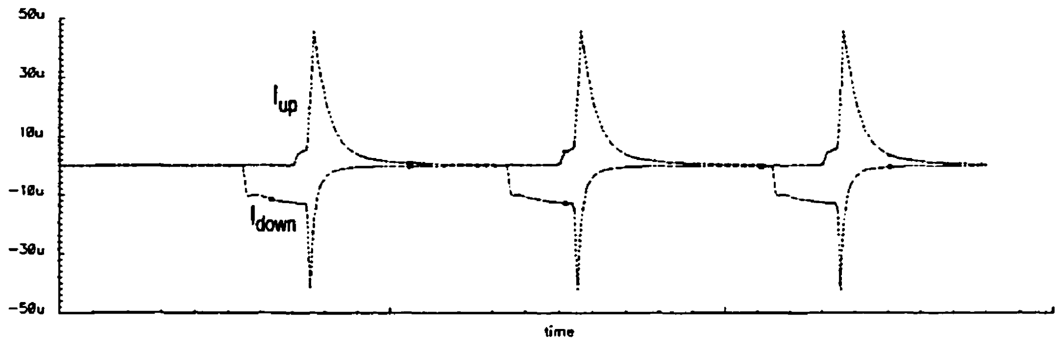


Figure 4.7: Up (I_{up}) and Down (I_{down}) Currents in Charge Pump Without Charge Sharing Removal Transistors

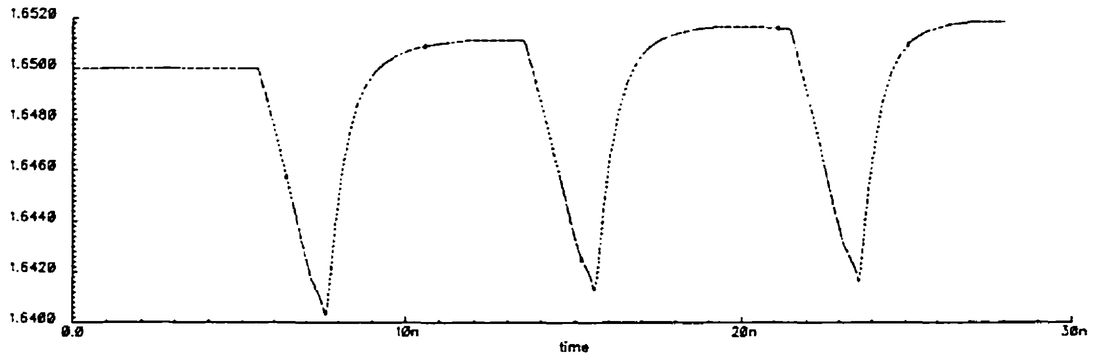


Figure 4.8: Control Voltage (V_C) Ripple Resulting From a Charge Pump Without Charge Sharing Removal Transistors

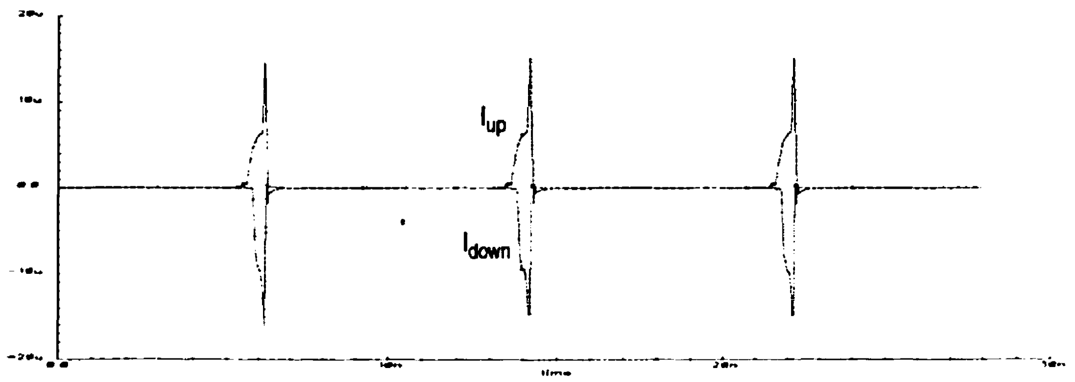


Figure 4.9: Up (I_{up}) and Down (I_{down}) Currents in Charge Pump With Charge Sharing Removal Transistors

Unfortunately, this charge-pump has a reduced dynamic range in the VCO control voltage (V_C). If V_C is greater than $V_{qbp} + |V_{tp}|$, the PMOS transistor between the output and pcs will discharge V_C through the NMOS transistor to ground when $\overline{U}$ is deactivated. A negative current for I_{up} can be seen between pulses in Figure 4.10. This leads to the slow discharge of V_C seen in Figure 4.11. Therefore, the range of V_C is limited to:

$$V_{qbn} - |V_{tn}| < V_C < V_{qbp} + |V_{tp}| \tag{4.3}$$

Through proper sizing of the PMOS and NMOS transistors between the output and pcs and the output and ncs, respectively, the desired current flow can be achieved with a higher V_{qbp} and a lower V_{qbn}, thus expanding the range of V_C.

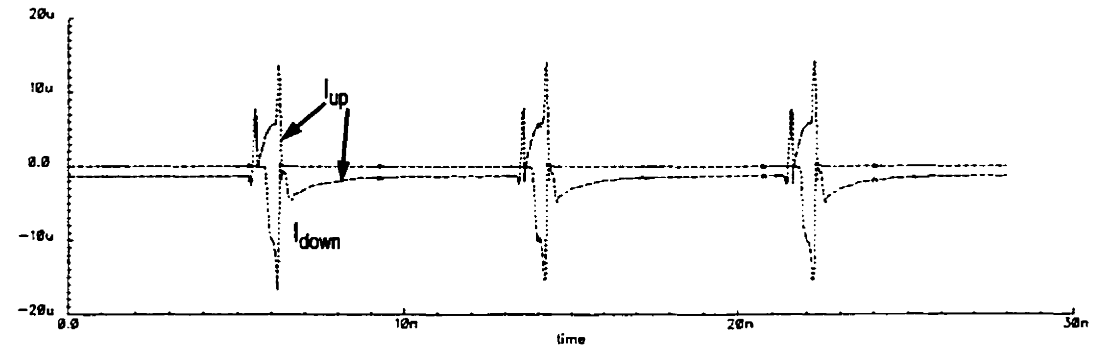


Figure 4.10: Up (I_{up}) and Down (I_{down}) Currents in Charge Pump with V_C at 3.0V. Notice I_{up} Flows Negative Between Corrective Pulses from the PFD

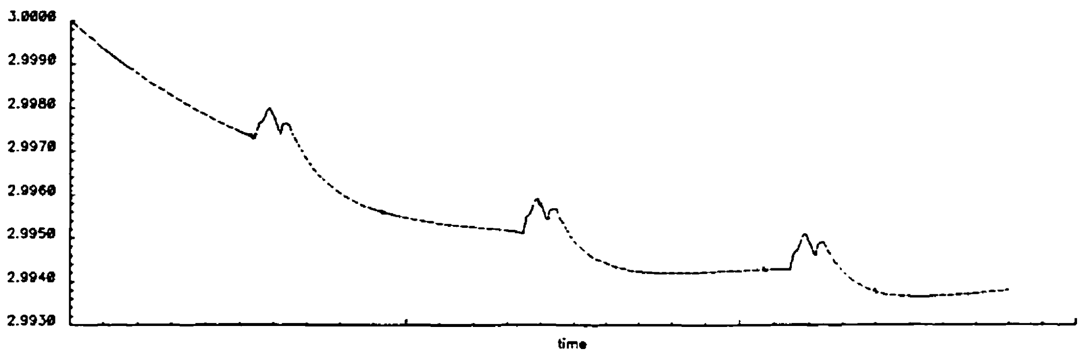


Figure 4.11: Discharge of the Control Voltage (V_C) due to Reverse Flow of I_{up} from V_C at 3.0V

4.2.3 - Phase-Frequency Detector

The differences in phase and frequency between the reference and feedback clock signals are determined by the Phase-Frequency Detector (PFD). The circuit then produces the up and down control signals to bring the feedback clock signal into lock with the reference. If the phase of the reference clock leads that of the feedback clock, then the up signal will be activated in an attempt to increase the phase of the feedback clock. Conversely, if the reference clock is lagging, then the down signal will be activated so as to decrease feedback phase. In the charge pump itself, these signals will be converted to an average *error current* over a cycle.

4.2.3.1 PFD Operation

A PFD with low logic depth is shown in Figure 4.12. Assuming that the PFD is in an initial state, the reference clock, feedback clock, up and down signals, and c are all zero. Also, the $\overline{\text{Rst}}$, a and b signals are ones, or a high voltage. A rising edge on the reference

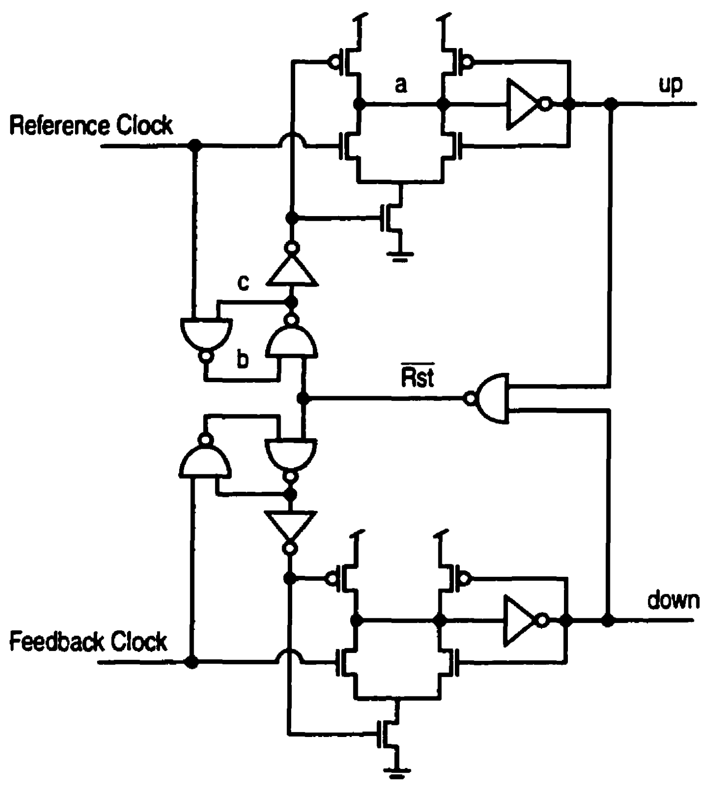


Figure 4.12: Phase-Frequency Detector

clock discharges a and charges up to a high value, without changing the state of the RS flip-flop formed by signal b, signal c, and two NAND gates. The weak feedback path will ensure that up will remain high after the reference clock falls. At the rising edge of the feedback clock, down will be activated and since both output signals are high, $\overline{\text{Rst}}$ will be set to zero. Consequently, a will be recharged and up is shut off. The signal down is simultaneously deactivated in the same manner. Therefore, a rising edge on the reference clock sets up = 1 and is then reset by the next rising edge from the feedback clock.

The operation of this PFD is equivalent to the classical PFD's implemented by two resettable D-flip-flops or four RS flip-flops. The advantage of this design is the reduced logic depth and thus a shorter delay. Thus reducing phase detector jitter caused by power supply dependent delays and device noise.

4.2.3.2 The Dead Zone

When designing the PFD for a charge-pump PLL, it is undesirable to allow the system to suffer from a *dead zone* problem. Consider the classical PFD and charge-pump model shown in Figure 4.13. Say, for example, ReferenceCLK leads FeedbackCLK. If the delay between the rising edge of ReferenceCLK and FeedbackCLK is sufficient, as shown in Figure 4.14(a), then Q_{Ref} will be able to fully charge to a logical one before the rising edge of FeedbackCLK causes a reset. However, if the delay between the rising edge of the

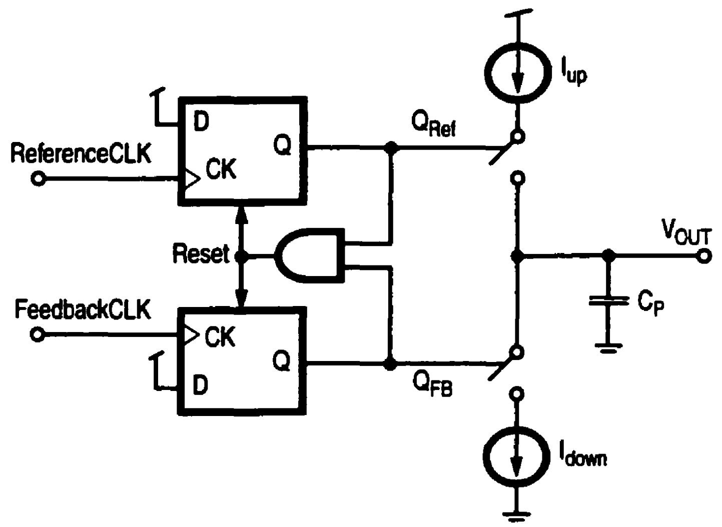


Figure 4.13: Classical Phase-Frequency Detector With Charge Pump

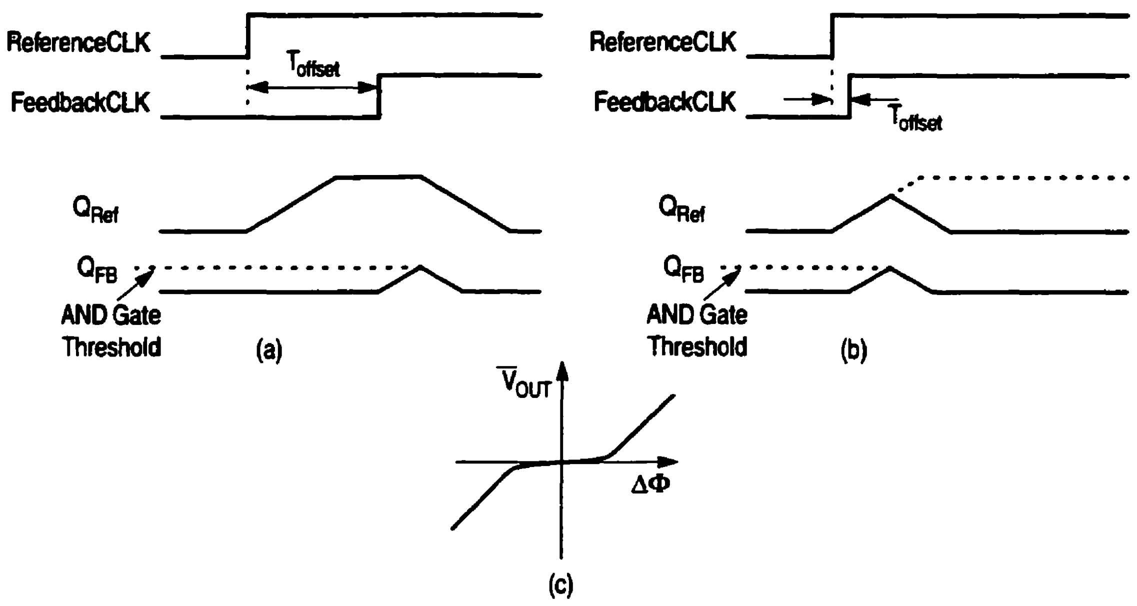


Figure 4.14: Dead Zone in a Phase Frequency Detector With a Charge Pump
(a) Normal Operation, ReferenceCLK is Advanced and Q_{Ref} Attains Full Value
(b) Advance of ReferenceCLK is Small, Q_{Ref} Does Not Attain Full Value
(c) Gain of PFD and Charge-Pump With a Low Gain *Dead Zone*

reference clock and the rising edge of the feedback clock is small, then Q_{Ref} may not reach a full logical one, as shown in Figure 4.14(b). As a result, the charge pump switch may fail to turn on, or be turned on for an ill-defined length of time. This causes the gain of the PFD and charge pump to drop, as shown in Figure 4.14(c), as the delay between ReferenceCLK and FeedbackCLK become comparable with the transition time of Q_{Ref} or Q_{FB}. The area where the gain is reduced is known as the dead zone.

If the phase difference between ReferenceCLK and FeedbackCLK varies within the dead zone, the output of the charge pump will not change significantly enough to correct the resulting error. Therefore, a peak-to-peak jitter, that is approximately equal to the width of the dead zone, may arise at the output of the VCO.

A common design technique to avoid a dead zone is to ensure that both the Q_{Ref} and Q_{FB} output signals are fully activated before turning them off. This can be implemented by ensuring that there is a delay in the reset path of the PFD, thus causing both Q_{Ref} and Q_{FB} signals to be simultaneously high for a short period of time. However, if the charge sharing in the charge pump is not perfectly cancelled or if there is a mismatch of I_{up} and I_{down}, current compensation will lead to a phase offset and loop filter ripple. A longer reset delay leads to a longer period in which the VCO is shifted in frequency. Therefore, the reset delay should be minimized with the constraint that it must be longer than the response time of the PFD.

Since the logic depth of the PFD in Figure 4.12 is short, the reset path delay for eliminating the dead zone may also be shorter. The reset delay of this PFD design can be reduced by permitting the reset of the precharged gate to be performed directly by $\overline{Rst}$ simultaneously as the RS flip-flop is reset. However, this is a less conservative design that could result in a dead zone.

4.2.3.3 Driving the Charge Pump

In order to reduce output jitter, the paths from ReferenceCLK and FeedbackCLK through the PFD to the charge-pump inputs should be as symmetrical as possible. Symmetry eliminates the offsets that lead to current compensation. However, the charge-pump requires not an U and D signal, but an $\overline{U}$ and D signal. The inverse value of U creates the necessity for an inverter to flip the logical level. However, placing a logical element in the path between the PFD and the charge-pump will cause the delay of $\overline{U}$ to differ from the delay of D.

The solution to this delay discrepancy is to also place a logical element in the path of the down signal of equal delay. A type of non-inverting delay element is a transmission gate. The circuit designed to generate a delayed version of the input signal and its inverted value is depicted in Figure 4.15. The single-to-differential ended converter passes the input signal through two paths. The upper path of the converter delays the input signal through the transmission gate and then inverts the signal by an inverter. The lower path equals the delay of the upper path with dual inverters which generates the non-inverted value. Through proper sizing of the transistors, the delays can be equalized. However, as an extra precaution to equalize the delays, a cross coupled diode connected differential pair was

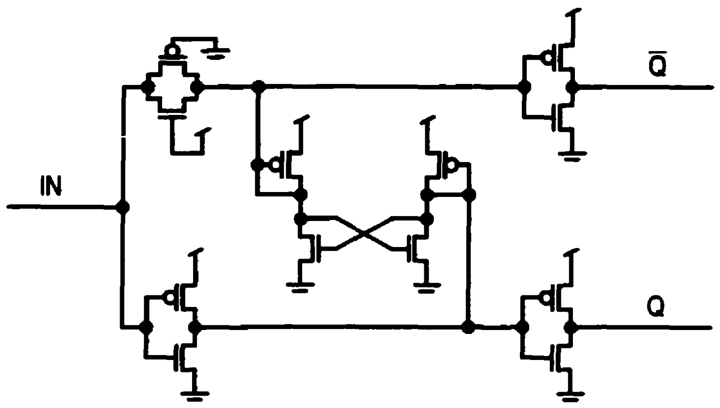


Figure 4.15: Single-to-Differential Converter

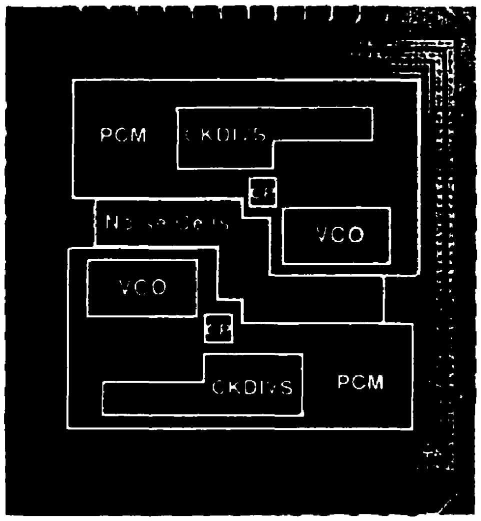
connected to the intermediate nodes of both paths. This circuit configuration equalized the delay in the same manner as a weak inverter loop, but with increased speed.

The single-to-differential converter is placed in the path of both the up and down signals, even though $\overline{D}$ and U from the converter will not be connected to the charge-pump. This symmetry equally loads both the up and down signals from the PFD and therefore the delays for these signals will be uniform from the PFD to the charge-pump.

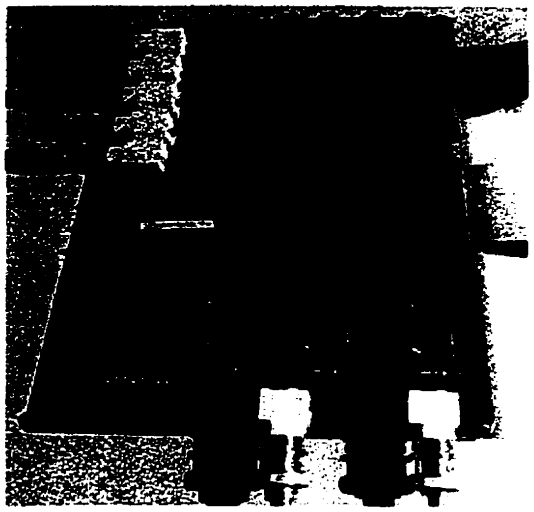
4.3 - Experimental Results

The PCM was designed and implemented with a full custom layout. The design was then manufactured using Taiwan Semiconductor Manufacturing Company's (TSMC's) 0.35μm, 3.3V CMOS process. In fact, two iterations were completed and tested. The initial iteration was fabricated as a stand alone device and the second iteration was part of a System-on-a-Chip (SoC) design. The iterative process allowed for several improvements that lead to reduced jitter and power consumption, and improved performance.

The first PCM implementation was accompanied by noise cell generation circuitry used for substrate modeling tests performed by a colleague. The noise cell generation circuitry could be turned off during tests of the PCM. The die photo is shown in Figure 4.16(a). The second iteration was part of an SoC and could be tested separately. A printed circuit board (PCB), as shown in Figure 4.16(b), was designed to improve test results.



(a)



(b)

Figure 4.16: Implementation of the PCM (a) Die Photo of First Iteration (b) Photo of the Second Iteration of the PCM on a Printed Circuit Board

4.3.1 - Tracking Jitter

Figure 4.17 shows the peak-to-peak tracking jitter, for all possible delay length settings of the VCO. A similar plot is shown for the RMS tracking jitter (or standard deviation) as well in Figure 4.18. In these plots, as in the other plots that follow, separate comparisons are made for 32, 16, and 8 delay cells (or taps), and 8, 4, and 2 cells respectively. The repetition of the 8 cells' characterization allows for easier comparison.

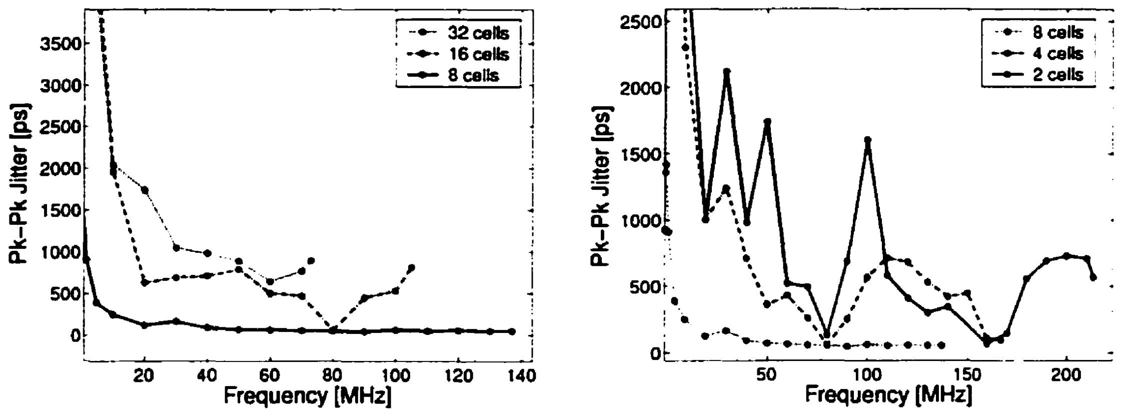


Figure 4.17: Tracking Pk-Pk Jitter [ps] plotted vs. Frequency [MHz] as a function of the number of taps in the VCO

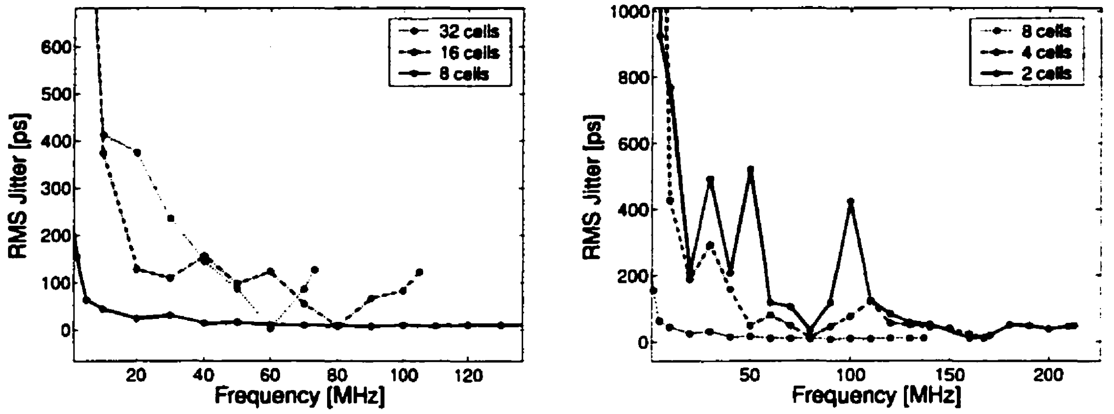


Figure 4.18: Tracking RMS Jitter [ps] plotted vs. Frequency [MHz] as a function of the number of taps in the VCO

Tracking jitter is measured by triggering the oscilloscope with the input clock once the PLL has achieved a lock, therefore measuring jitter compared to the reference clock and not the cycle to cycle jitter of the output clock. The peak-to-peak jitter is the maximum jitter exhibited in the measurement, in this case, over a period of 250 waveform acquisitions. The RMS jitter is the standard deviation over that same time period. It should be noted that the measurements were made with a reference clock that produced 32.42 ps pk-pk and 3.927 ps RMS jitter at a frequency of 50 MHz, when measured directly.

Upon observation of both the peak-to-peak and RMS tracking jitter, one may observe that the quantity of jitter is reduced as the VCO length decreases from 32 to 8 cells, and then increases as the length is shortened to 4 and then 2 delay cells. The VCO setting of 8 delay

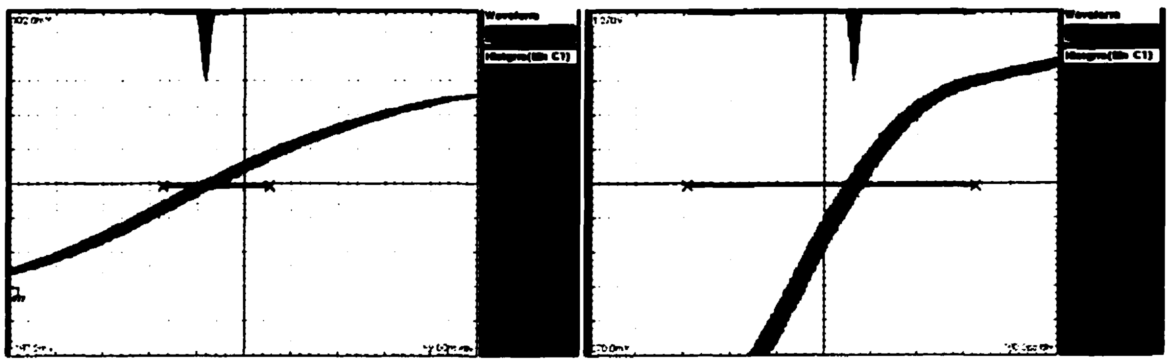

(a) (b)

Figure 4.19: Jitter Tracking Histograms (a) Reference Clock @ 50 MHz
(b) PLL Output @ 90 MHz, Length = 8 Taps

cells not only exhibits the least amount of jitter (44.00ps pk-pk, 7.234ps RMS @ 90MHz) but is also consistently low over the operating range.

Another possible observation is the increase in jitter at the extremities in the operating range. The numerical values from the tables in Appendix C demonstrates this effect more clearly. At the lower frequency levels, the jitter is high for several reasons. Primarily, mathematical modeling has shown that decreasing the tail current (I_{tail}) in the differential pair delay cell causes an increase in output jitter [43,44]. Also, for a reduced I_{tail}, and thus at lower frequencies, the rising and falling edges of the delay cells are asymmetrical and therefore degrade phase noise and jitter by increasing the $1/f^3$ corner frequency [44]. Another cause for the increased jitter can be derived from the increased gain K_O of the VCO at lower frequencies. As a result, noise on the VCO control voltage will have a greater effect on output jitter.

4.3.2 - Duty Cycle

As the operating frequency decreases and the tail current in the differential pair delay cell is reduced, Figure 4.3, the gain of the pair is also reduced. This results in a diminished output swing from the cell. When the differential signal is transformed into a single-ended signal, the high common-mode input voltage level causes the converter to exhibit an asymmetry between the high and low phases of the output clock signal, and thus a duty

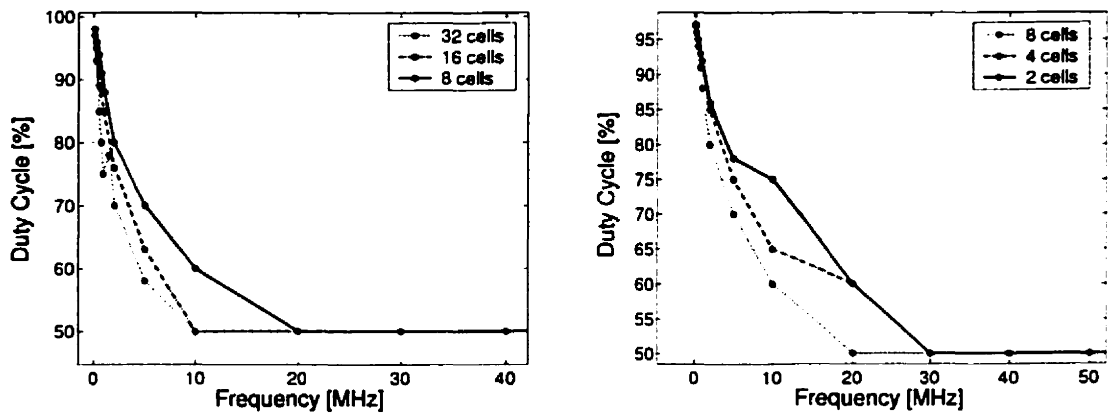


Figure 4.20: Output Duty Cycle [%] plotted vs. Frequency [MHz] as a function of the number of taps in the VCO

cycle other than 50%, as shown in Figure 4.20. By passing the output clock through a divider, the uneven duty cycle can be restored to 50%.

4.3.3 - Power Consumption

As expected, higher operating frequencies lead to increased power consumption by the PCM, as shown by the graphs in Figure 4.21. This relation implies that in order to reduce power consumption, the PCM output frequency should be limited to the speed of the highest required speed. Thus, only one clock divider needs activation if one of the clock signals runs slower, while the other divider is bypassed. By running the PLL at a high frequency and then using both output clock dividers, power is unnecessarily wasted.

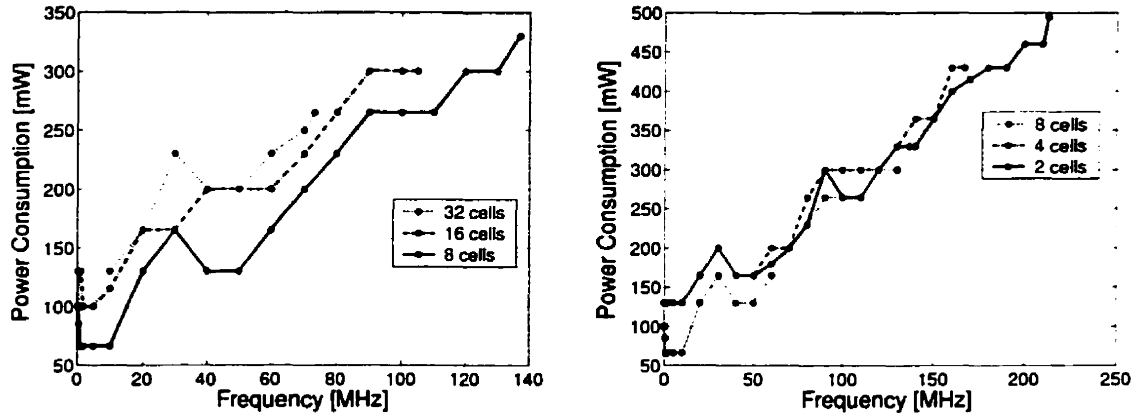


Figure 4.21: Power Consumption [mW] plotted vs. Frequency [MHz] as a function of the number of taps in the VCO

4.3.4 - Comparing Iterations

Two iterations of this PCM design were completed and tested. Both designs were similar in circuit structure, however several improvements were made in the SoC implementation. The comparisons between the iterations were made with the length set to 8 taps. These improvements lead to reduced jitter that was also more consistent over the operating range, Figure 4.22. The duty cycle was also more symmetrical at lower frequencies, Figure 4.23. At lower frequencies, the power consumption was reduced in the second iteration, but the initial design consumed less in the mid range. Another major improvement is the extension of the PLL operating range.

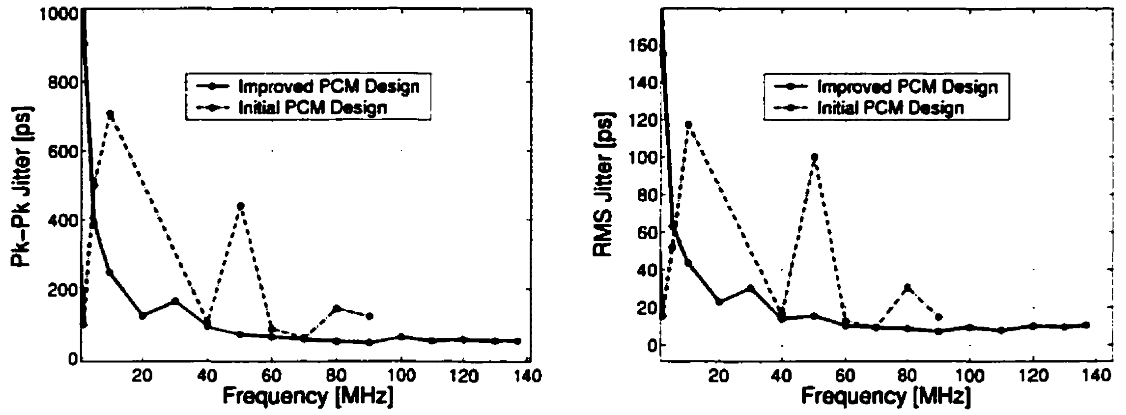


Figure 4.22: Tracking RMS and Pk-Pk Jitter [ps] plotted vs. Frequency [MHz] comparing the initial PCM design to the improved version, length = 8 taps

The improvements in the second iteration resulted from many changes. One was an improvement in the linearity of the VCO gain, K_O, and an extended operating range through transistor resizing. The charge pump and its bias circuitry was also resized in order to extend the V_C range and reduce the size of the bias resistor. Most importantly, the loop filter was changed to shift the loop bandwidth and reduce jitter. The improved filter also resulted in increased stability, where the initial design became unstable between 16MHz and 36MHz. Other frequency regions of instability were eliminated by altering the PLL loop dynamics through programmability of the charge-pump bias circuitry.

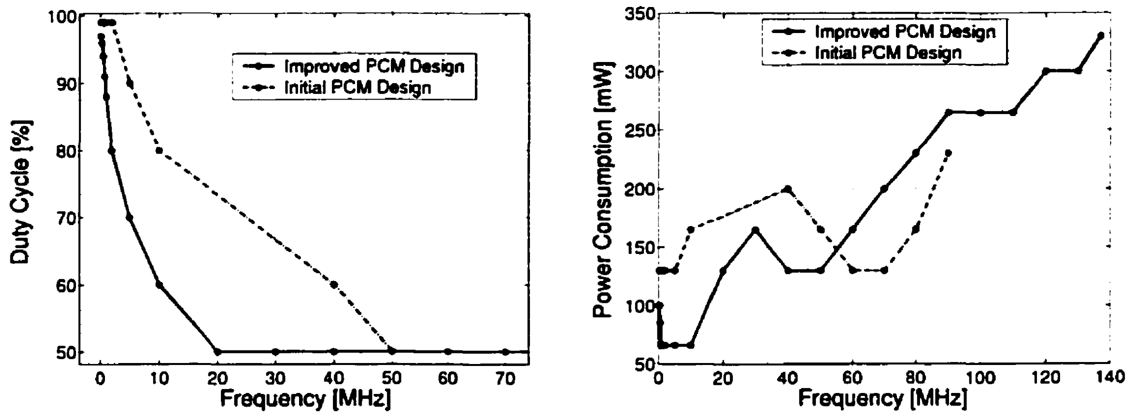


Figure 4.23: Output Duty Cycle [%] and Power Consumption [mW] plotted vs. Frequency [MHz] comparing the initial PCM design to the improved version, length = 8 taps

4.4 - Recommendations for Improvements

Further improvements are possible in the design, towards increasing the functionality, versatility and performance of the current design of the Programmable Clock Manager. One improvement is the increase in the number of possible divisions the DPCDs can produce, as described in Section 3.4. This will extend the output range by reducing the lower operating limit. By placing an extended clock divider in the feedback path of the PLL, the frequency of the input clock may be reduced. Thus, the external reference clock may be slowed and be less susceptible to board noise. Also, as the upper operating range of the PLL increases, it will not be necessary to increase the reference clock frequency. However, it should be noted that a wider range of feedback clock divisions leads to a wider variation in the stability constraint.

Another improvement is possible in the design of the VCO. The transmission gates, used to multiplex clock signals from the various delay cells should be replaced by a multi-level multiplexer in order to reduce capacitive loading and increase the maximum operating frequency by reducing delay. Also in the VCO design, the differential-to-single ended converter could be implemented by a circuit composed of two opposite NMOS differential amplifiers driving two PMOS common-source amplifiers connected by an NMOS current mirror [39]. This will result in a 50% duty cycle, regardless of the VCO control voltage, because the correct common-mode input voltage level will be received.

The PCM should also have circuitry to indicate that the feedback signal has locked onto the reference clock signal. This signal can be used to control clock gating circuitry to prevent malfunctions in the synchronous circuitry when perturbations occur in the PLL or reference clock. Also, when the PLL is locking onto a new frequency, there is a possibility of overshoot. This overshoot may exceed the operational capability of the synchronous system being driven. A similar situation may occur if an incorrect setting of the PCM registers leads to an operating frequency that also exceeds the maximum tolerance. To prevent these failures, the lock circuitry can be modified to be reset when V_C exceeds a particular voltage for a given length.

Other possible improvements include increased linearity of the VCO gain, K_O, over the entire operating frequency range. This leads to less variation in the stability constraint and thus a simpler PLL design in terms of loop dynamics. Also, if K_O is reduced, then there is more flexibility in choosing design parameters, such as damping factor and lock time [26,45,42], while easily meeting the stability constraints. Jitter is also lower when K_O is decreased [46,47]. However, by reducing K_O, the operating range is also reduced for the possible range of voltages attainable by V_C. As the supply voltage is reduced, the range of V_C is reduced, and the problem is compounded. If a PLL can be designed to take advantage of the various operating ranges produced by varying the VCO length, without requiring external intervention, then the PCM can have an increased range at lower supply voltages and reduced jitter. Such a PLL is the focus of the next chapter.

4.5 - Summary

This chapter described the development of a Programmable Clock Manager capable of dynamically producing a variety of output frequencies. This design lends well to the demands and requirements of Dynamic Clock Management. The components within the PCM were chosen based on the requirements of the clock generation circuitry. DPCDs were used for the clock dividers so that the outputs would not exhibit glitches. The PLL was designed for a wide operating range and reduced jitter. The PCM design was implemented and tested for functionality and performance. There are several improvements that can be made to the initial designs, one is the Range Shifting Phase-Locked Loop.

Chapter 5 - Range Shifting Phase-Locked Loop

For the purposes of dynamic clock management, the clock manager must be on-the-fly programmable, capable of producing a wide range of frequencies, have low power consumption, and generate minimal jitter. Having the Phase-Locked Loop (PLL) exhibit these characteristics is beneficial considering the PLL is the major component of clock managers. As Integrated Circuit (IC) design continues to advance, thus requiring more stringent requirements from the clock distribution network, the clock generation circuitry must also develop so as to offer increased flexibility and operational capabilities. This circuitry must offer a wider range of frequencies with less jitter. The Range Shifting Phase-Locked Loop (RSPLL) is a novel architecture capable of offering these desirable properties.

5.1 - PLL Design Concerns

As System-on-a-Chip (SoC) designs increase in density and more independent sub-systems are included in the same IC, the range of required frequencies supplied by clock managers is extended. Dynamic clock management requires a clock manager capable of attaining the maximum operating frequency for a system, in addition to lower frequencies during times of reduced demand. As the upper limit of the operating frequency continues

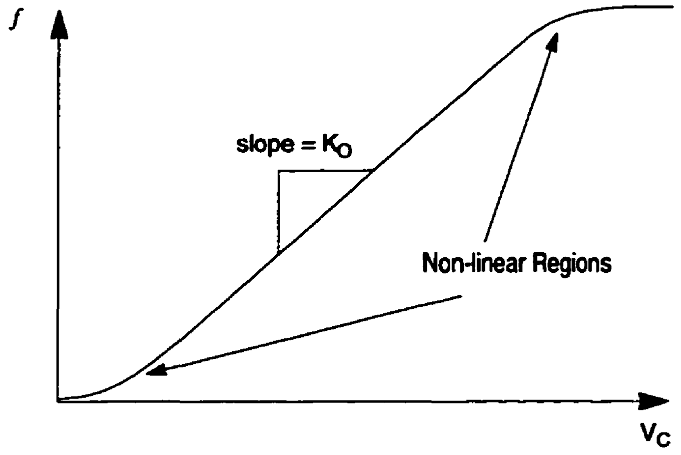


Figure 5.1: Frequency vs. Control Voltage: Standard VCO

to rise, the frequency range changes proportionally. A wide operating range causes the Voltage Controlled Oscillator (VCO) to have a large K_O, defined as the *VCO gain* in radians/second/volt. VCO gain is also represented as the slope of the line in Figure 5.1. For a given voltage range, the line becomes steeper as the upper limit of *f* increases, shown in Figure 5.2(a). As the supply voltage level (V_{DD}) drops, the range of the VCO control voltage (V_C) is also reduced. Therefore, K_O must be increased to maintain the same frequency range, as shown in Figure 5.2(b).

5.1.1 - Meeting PLL Specifications

A large VCO gain creates difficulties and reduces flexibility in meeting PLL specifications. Design to ensure stability is increasingly problematic, in addition to other factors such as damping, pull-in time, noise performance and filter coefficients. To demonstrate the difficulties in choosing design factors when K_O is large, the stability and damping factor will be examined. For a second-order approximation of a PLL system, the *stability limit* is given by [42,38]:

$$\omega_{Ref} > K_O I_P R \tag{5.1}$$

Also, as derived in Appendix A, the *damping factor* is represented by:

$$\zeta = \frac{RC}{2}\sqrt{\frac{K_O I_P}{2\pi C}} \tag{5.2}$$

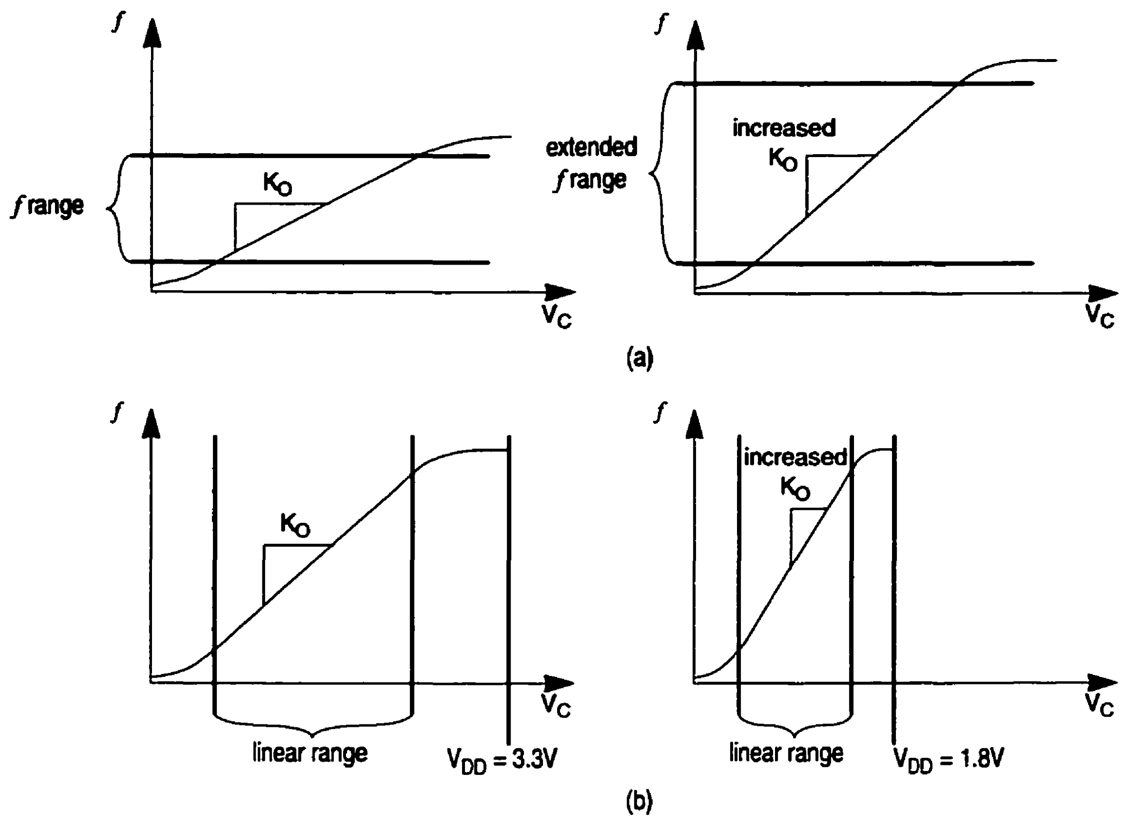


Figure 5.2: Factors Leading to an Increased K_O: (a) Increase in Frequency Range (b) Decrease in Supply Voltage (V_{DD})

In order to prevent an overshoot of the lock value that may exceed operational capabilities of the system, as shown in Figure 2.10, the damping factor is forced to be greater than one. However, considering a feedback division N, the ζ will vary with $1/(\sqrt{N})$. As a result, if all of the other design parameters are held constant, ζ will decrease as N increases and thus the damping factor will most likely become less than one.

Without considering the effects of N, attempting to design a PLL with a high K_O creates difficulties in choosing the remaining design parameters. By observing the stability constraint, one should notice that a high K_O requires the product of I_P and R to be small, Equation (5.1). Reducing the size of R allows its absolute value to be more susceptible to variations in the manufacturing process. On the other hand, decreasing I_P will cause noise effects to have a greater influence on the charge-pump circuitry.

When attempting to obtain an appropriate ζ, a high K_O with a low I_P and R will require C to be very large to compensate, Equation (5.2). Also, with the possibility of a high value for N, C will have to be even greater.

Also, when pursuing an extended operating frequency range, the VCO frequency vs. voltage characteristic may have non-linear gain characteristics through regions near the extremes of the operating range, as shown in Figure 5.1. This further complicates design modeling and could result in possible instabilities or unexpected characteristics.

5.1.2 - Increased Jitter

An additional concern is derived from a noisy V_C signal, which leads to jitter at the output of the VCO [48,46,47]. Say, for example, a noisy peak-to-peak 10mV signal at V_C is measured to directly cause 100ps of jitter at the output of the VCO for a given K_O. In order to maintain a constant frequency range as V_{DD} is halved, K_O would have to at least be doubled. This will cause that same 10mV of peak-to-peak noise to generate 200ps of jitter.

5.2 - RSPLL Architecture

The RSPLL architecture, shown in Figure 5.3, provides a wide operating range while maintaining the benefits of a VCO with a low gain (K_O) and linear operation. Also, the design is better suited for system-level programmability. This is achieved with a VCO capable of shifting operating ranges by automatically changing its length. The PLL that was designed and implemented is similar to the PLL designed for the Programmable Clock Manager (PCM). Therefore, many of the components, such as the Phase-Frequency Detector (PFD), Charge-Pump (CP), loop filter structure and VCO delay cells, are the same as the devices presented in Section 4.2.

There are many ways of implementing the RSPLL. A charge-pump PLL was used as the basic building block, however the design is not limited to this type of PLL and may be implemented in various other forms. The main distinctions between the RSPLL and conventional PLLs are the variable length VCO, the length control for the VCO, the way the variable length VCO is incorporated into the PLL, and the applications of the RSPLL.

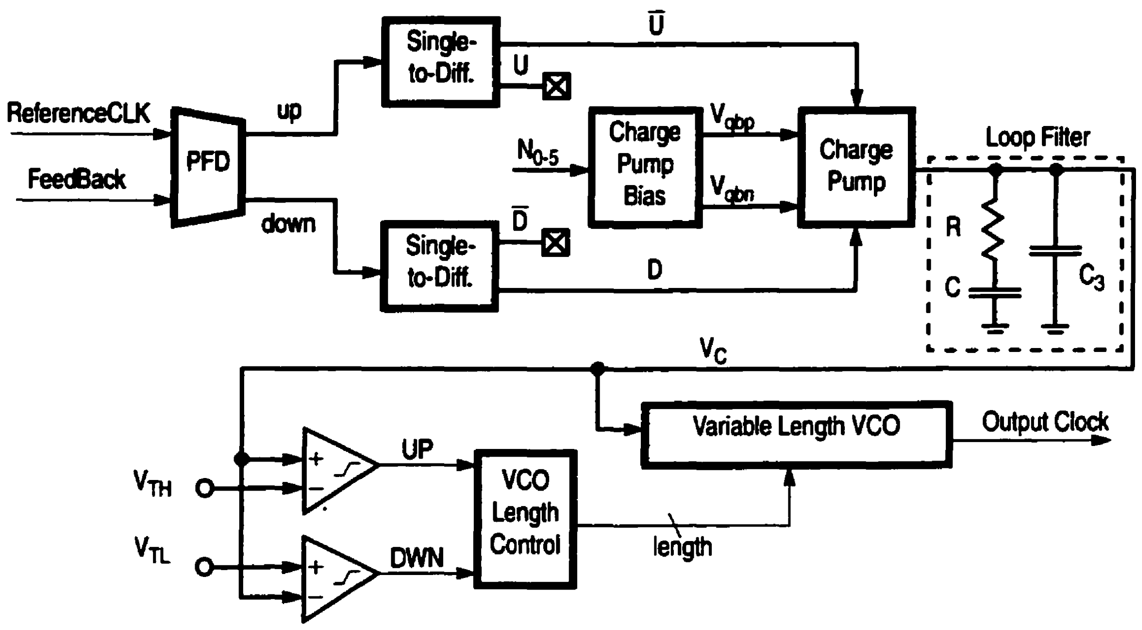


Figure 5.3: Proposed PLL Architecture

The VCO is implemented with 16 delay cells and the following discussion adhered to this particular implementation. However, the number of cells is not limited and can be varied to meet desired specifications.

5.2.1 - Reduced K_O with a Variable Length VCO

The output frequency range of the VCO is dependent on its length, or number of active delay cells. The angular frequency of the VCO in its linear mode is:

$$\omega_{OSC} = m \cdot \omega_{OC} + K_O \cdot V_C \tag{5.3}$$

$$m = 16/x \tag{5.4}$$

where ω_{OC} is the VCO center frequency when all 16 delay cells are active, and x is the number of active delay cells. Changing the length affects the value of m, thus creating a shift in the linear operating range along the y-axis, as shown in Figure 5.4. The VCO is designed to keep K_O constant and therefore independent of x.

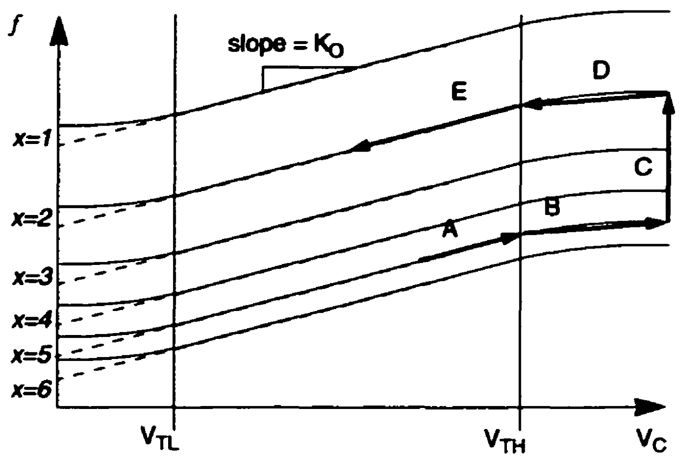


Figure 5.4: Frequency vs. Control Voltage: Output of VCO for Varying Lengths

System operation is consistent with traditional PLLs when:

$$V_{TL} < V_C < V_{TH} \tag{5.5}$$

and therefore, within this range, can be modeled as a conventional charge pump PLL. Outside this range, the coarse control loop is activated to automatically determine the length necessary to return operation to the linear region. In contrast, a standard clock manager PLL would need external intervention. As the required VCO length setting changes due to process variations, external control is inherently difficult and prone to errors.

5.2.2 - RSPLL Operation

Figure 5.5 shows an example of the RSPLL timing diagram where the control voltage (V_C) exceeds the upper threshold voltage (V_{TH}). This causes the length controller to decrease the number of active VCO delay cells, x and consequently increment m. The result is a shift in the VCO operating range until the lock frequency is within the bounds set in Equation (5.5). In a similar manner, the operating range can be shifted down when V_C falls below V_{TL}. Thus the overall range of the VCO is expanded without increasing K_O while ensuring that the output frequency remains in the linear region of the VCO.

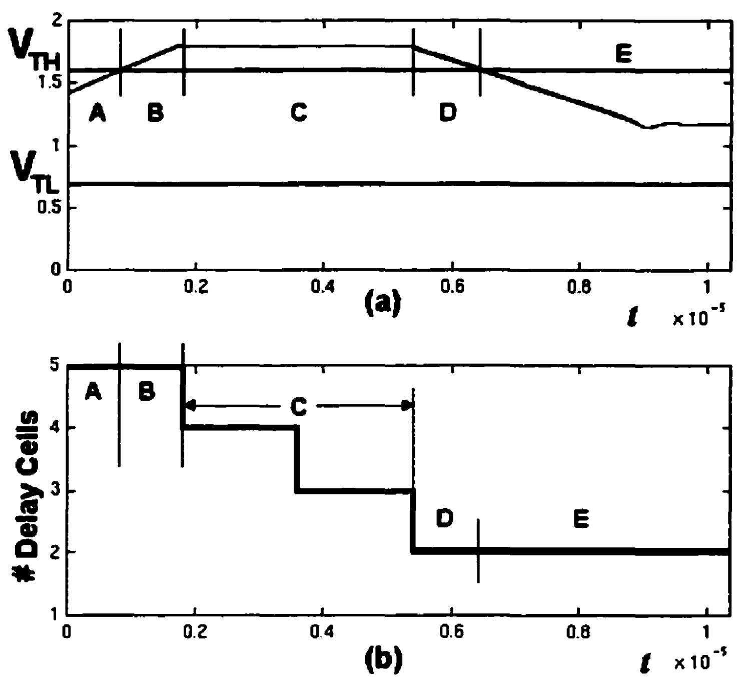


Figure 5.5: (a)V_C vs. Time (b)Length vs. Time

5.3 - Loop Dynamics: Conditions for Range Shifting

Two parallel control loops can compromise the overall system stability and lead to oscillations in the output frequency. When modeling the PLL in the locked state, the length is invariant and thus *m* can be held constant. As a result, the Laplace transform of $\omega_{OSC}(t)$ becomes:

$$\Theta_{OSC}(s) = K_O V_C(s)/s \tag{5.6}$$

which is the transfer function for a conventional charge pump VCO [45,26]. Therefore, in the locked state, a basic charge-pump analysis is applicable to this PLL [42]. A second-order analysis is presented in Appendix A.

When V_C exceeds the threshold limits, the coarse control loop is introduced to the model. To ensure stability, the length must not change too quickly or unnecessarily. For example, the single loop response can be approximated as a second-order system defined by the transfer function:

$$H(s) = K\omega_1 \cdot \frac{1 + \frac{s}{\omega_1}}{s^2 + sK + K\omega_1} \tag{5.7}$$

Consider the case where this second-order system has a normalized underdamped step response shown in Figure 5.6. The response will exhibit an overshoot of the final value followed by a ripple until it settles into lock. The points of interest on the figure are the rise time (t_r) and the time to the peak overshoot value (t_p), defined as:

$$t_r \approx \frac{2.2}{\omega_n\sqrt{1 - 2\zeta^2 + \sqrt{2 - 4\zeta^2 + 4\zeta^4}}} = \left.\frac{2.2}{\omega_n}\right|_{\zeta = \frac{1}{\sqrt{2}}} \tag{5.8}$$

$$t_p = \frac{\pi}{\omega_n\sqrt{1 - \zeta^2}} \tag{5.9}$$

$$\zeta = \frac{R}{2}\sqrt{\frac{K_O I_p C}{2\pi}} \tag{5.10}$$

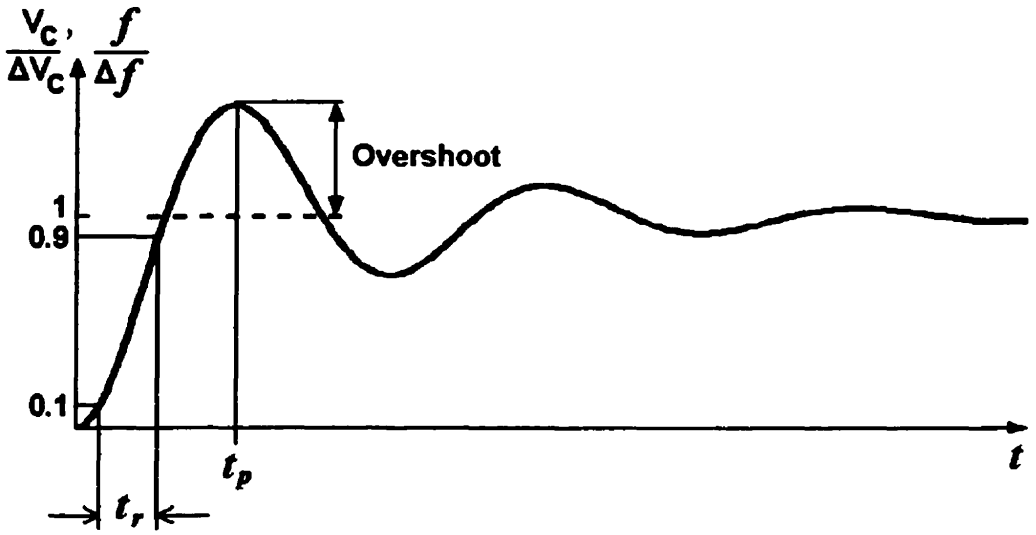


Figure 5.6: Step Response of Underdamped Second-Order System

where ω_n is the natural frequency of the system, ζ is the damping factor, R and C are from the loop filter, and I_P is the bias current of the charge pump [42,49]. When ζ is less than 1 (critical damping occurs for $\zeta = 1/\sqrt{2}$), there will be an overshoot of the lock frequency. When this frequency corresponds to a V_C near V_{TH} or V_{TL}, the overshoot will cause V_C to surpass the threshold. The first two values of t where the step response crosses unity are:

$$t_1 = \frac{\ln(-\zeta + \sqrt{\zeta^2 - 1})}{\omega_n \sqrt{\zeta - 1}} \qquad (5.11)$$

$$t_2 = \frac{\ln(-\zeta + \sqrt{\zeta^2 - 1})}{\omega_n \sqrt{\zeta - 1}} + \frac{\pi}{\omega_n \sqrt{1 - \zeta^2}} \qquad (5.12)$$

The time difference between these points:

$$t_2 - t_1 = \left(\frac{\ln(-\zeta + \sqrt{\zeta^2 - 1})}{\omega_n \sqrt{\zeta - 1}} + \frac{\pi}{\omega_n \sqrt{1 - \zeta^2}} \right) - \left(\frac{\ln(-\zeta + \sqrt{\zeta^2 - 1})}{\omega_n \sqrt{\zeta - 1}} \right) = t_p \qquad (5.13)$$

shows that $(t_2 - t_1)$ is equal to t_P, given by Equation (5.9).

5.3.1 - Activating the Coarse Control Loop

Before executing a length change, a delay which exceeds the worst case value of t_P should be introduced to ensure against undesired shifts in operating range. If, after the delay, V_C is still above V_{TH} (region **B**, Figure 5.5) or below V_{TL}, then it can be assumed that the final locked value of V_C will also be beyond the threshold. Therefore, a different VCO length will be better suited for generating the desired frequency and a change will be necessary. Conversely, if V_C moves back across the threshold before a delay of t_P, then it can be assumed that the final value of V_C will be within the inequality given in Equation (5.5).

5.3.2 - Deactivating the Coarse Control Loop

When a large jump in operating frequency is necessary, multiple length changes are executed. In this case, V_C will remain above the threshold. After the appropriate delay, to ensure there is not an overshoot (region **B**, Figure 5.5), the VCO is adjusted to the proper operating range (region **C**, Figure 5.5). When this range is set, V_C will proceed toward the linear region as the primary control loop begins acquisition (region **D**, Figure 5.5). The fine control loop must be given enough time to bring V_C below V_{TH} before the coarse control loop conducts an additional shift in operating range. The length controller must wait between steps for a period longer than t_r, Equation (5.8), to prevent undesired additional decreases in length. If too many length changes occur by not waiting for enough time, V_C will most likely oscillate between thresholds.

5.3.3 - Coarse Loop Induced Instability

The next consideration focuses on instabilities due to oscillations between two nonlinear regions, $V_C > V_{TH}$ and $V_C < V_{TL}$. Instabilities would occur if the length change causes the VCO to converge towards the static operating point in the alternative non-linear region. As demonstrated in Figure 5.7, this will happen only if there is an absence of overlap between neighboring linear regions. Through design of the VCO delay elements and the choice of threshold voltages, this overlap is easily obtainable.

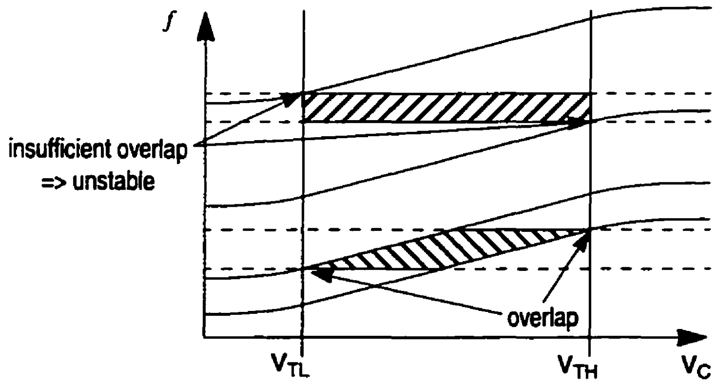


Figure 5.7: f vs. V_C: Frequency Range Overlap

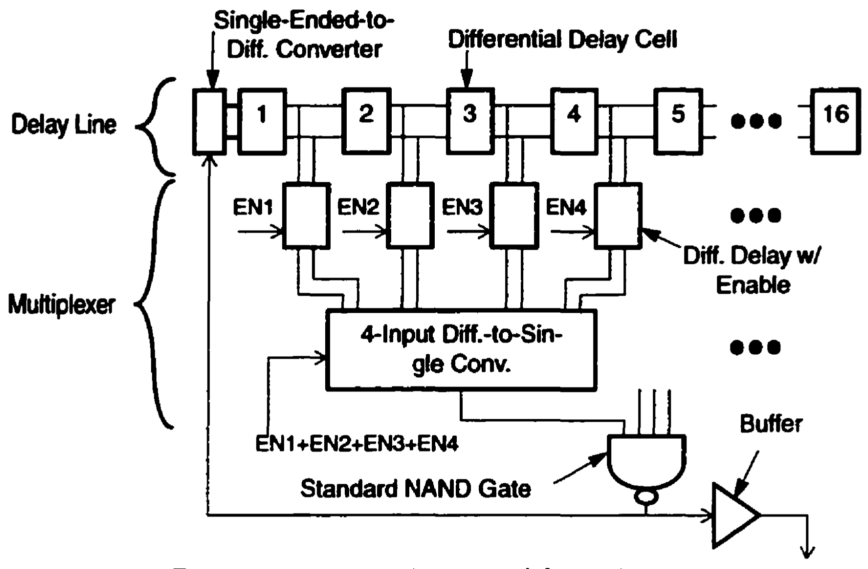


Figure 5.8: Variable Length VCO Architecture

5.4 - Variable Length VCO Implementation

The VCO, shown in Figure 5.8, is comprised of 16 delay cells and a multiplexer which dynamically selects the appropriate length. To reduce jitter, a differential pair, Figure 5.9(a), was selected due to its low sensitivity to power supply and substrate noise [38,39]. The delay cell contains a source coupled pair with PMOS diodes used for clamping the output voltage to provide a fixed common mode and swing without the need for a replica bias circuit [38].

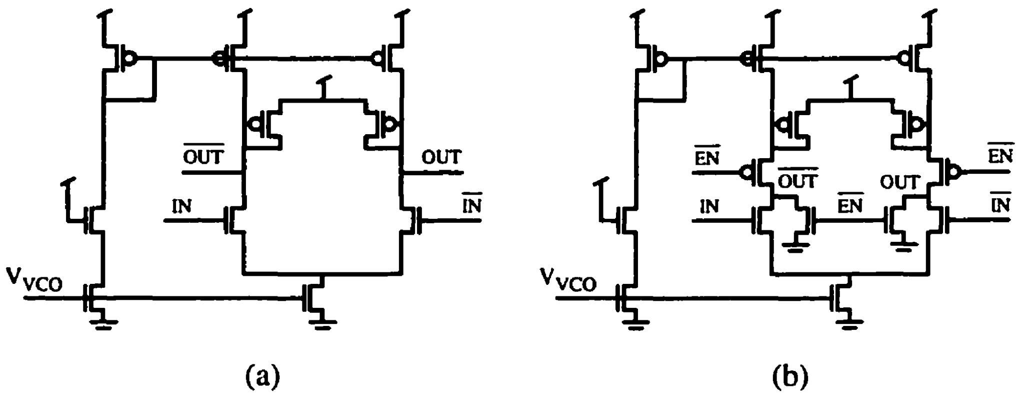


Figure 5.9: Differential Delay Cell: (a) Basic (b) with Enable

The multiplexer selects the desired differential signal and converts it to a single-ended digital signal. For the same reasons as in the delay cell, a differential pair was selected for most of the logic elements in the multiplexer. A circuit structure similar to the delay cell is used so its delay is also varied based on control voltage. The differential delay cell with an enable signal (EN), shown in Figure 5.9(b), operates in the same manner as the standard delay cell of Figure 5.9(a) when EN is low. However, when EN is high, both differential output signals are low. In this way, cells that are not enabled cannot influence their corresponding NMOS pair in the four input differential-to-single-ended converter, as shown in Figure 5.10.

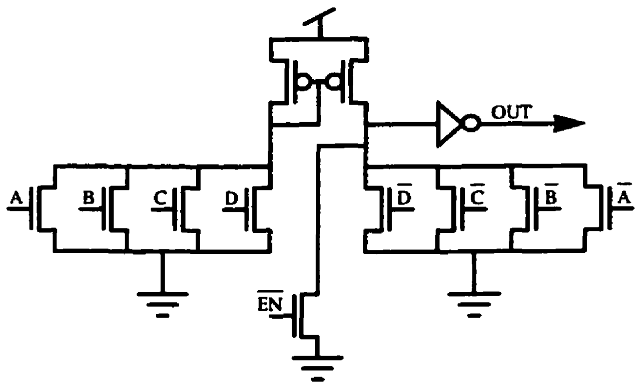


Figure 5.10: Four Input Differential-to-Single Converter

The VCO delay cells, including the enabled differential gates, are grouped into four by the differential-to-single-ended converter. When the chosen path is not contained in a particular grouping, all of the differential gates are disabled and the output of the differential-to-single ended converter is tied high. As a result, only a traditional four input NAND gate is required to pass the clock signal. Also, deactivation of unused logic reduces unnecessary switching, thus resulting in a reduction of overall power consumption. The signal completes the oscillator loop by returning to the input of the delay line through a single-to-differential ended converter, as shown in Figure 4.15.

5.5 - Length Controller

The length of the VCO is selected by the Length Controller in a manner that prevents glitches on the output clock. The length is stored in the controller by employing a daisy chain of 16 cells. Each cell produces an enable signal (*EN*) that corresponds to a path, and also a differential delay cell, in the VCO. The length controller cell is comprised of three Muller C-elements [50] with weak inverter loops to hold the state. C-elements were chosen because they are compact and asynchronous. They are quick, glitch-free and consume less area than the equivalent implementation of flip-flops and logic. The structure of each cell is shown in Figure 5.11.

In addition to *EN*, each cell in the length controller has secondary signals *Y* and *Z* for use within the length controller. Signal *Y* is for use only within the individual cell (*x*), however *Z* and *EN* are used by adjacent cells (*x*+1 and *x*-1). The *UP* and *DWN* signals are generated by comparators who compare the control voltage to the appropriate threshold voltages. *UP* being set when V_C exceeds V_{TH}, and conversely *DWN* when V_C drops below V_{TL}. The purpose of *Cnt* is to indicate the passing the of the appropriate delays, as discussed in Section 5.3. In this design of the RSPLL, the delays are generated by a programmable counter activated by the assertion of *UP* or *DWN*. In order to ensure a glitchless clock signal from the VCO, the length is changed by an increase or decrease of only one cell per transition. In addition, the enable signals to the VCO multiplexer are overlapped, so a path is not disabled before the new path is fully functional.

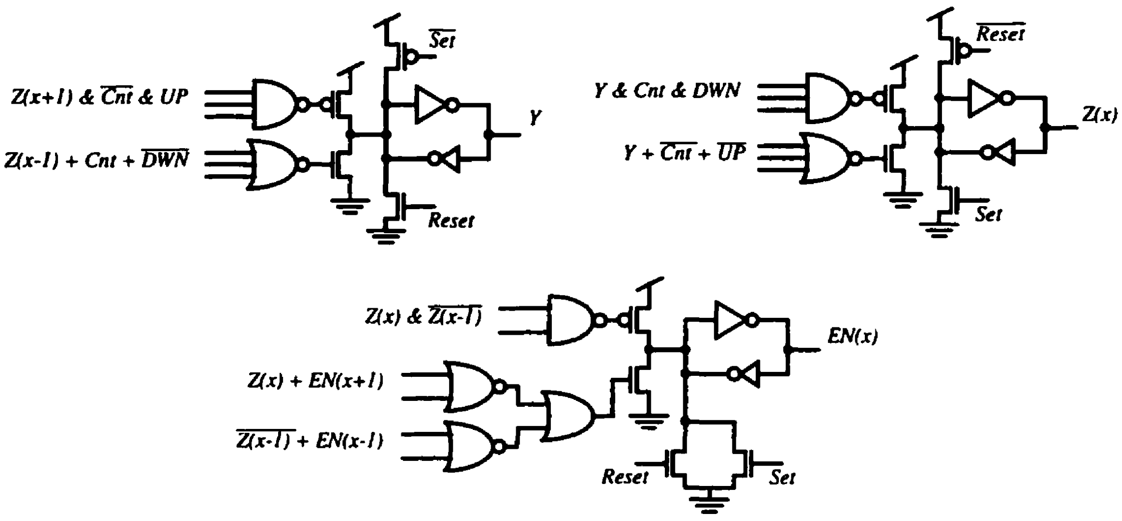

$\overline{Set}$
$Z(x+1)$ & $\overline{Cnt}$ & UP
$Z(x-1) + Cnt + \overline{DWN}$
Y
Reset
$\overline{Reset}$
Y & Cnt & DWN
$Y + \overline{Cnt} + \overline{UP}$
$Z(x)$
Set
$Z(x)$ & $\overline{Z(x-1)}$
$Z(x) + EN(x+1)$
$\overline{Z(x-1)} + EN(x-1)$
$EN(x)$
Reset
Set

Figure 5.11: A Single Cell in the Length Controller Comprised of Three Muller C-Elements

Upon initial start-up, all of the signals are deactivated. As the control voltage from the charge pump surpasses the upper threshold voltage to achieve lock, *UP* is asserted and the counter is activated. After a delay, *Cnt* is set only if V_C still exceeds V_{TH}. In this case, *Z*(16) and *EN*(16) is asserted, the counter resets but continues to count, and the VCO begins to oscillate at full length and in the lowest frequency range.

If after another predetermined delay and *UP* is still asserted, *Cnt* will be set again. At this time *Z*(15) and *EN*(15) will be asserted. The signal *EN*(16) will now be reset, only after *EN*(15) has been fully asserted to ensure that an overlap exists. With the change in active enable signals comes a change in length of the VCO and thus a change in output frequency range. Since the length transition is only to an adjacent cell, the difference in each output is only equal to the delay of one cell. Therefore, it is likely that both cells are in the same state when the output from an adjacent cell is used to create the loop of the VCO. If one or both of the cells are in transition, the overlap of the enable signals causes the final output of the four input differential-to-single converter to be an average of the input signals. Another precaution to remove glitches during a length change is to activate the length controller at the rising edge of the oscillator output clock. If a change in the enable signals occurs during the duty cycle of the output clock, then the length will be changed only when both adjacent cells are in the same high state.

The length controller continues operating in such an manner until the desired output frequency range has been reached. If at this time the control voltage drops below the upper threshold voltage, then *UP* will no longer be set and the delay counters are reset. The length controller thus maintains the current setting and the RSPLL obtains lock in the current frequency range until such time as a different output frequency is desired. The reverse procedure occurs when the control voltage drops below the lower threshold voltage in order to obtain lock in a lower output frequency range. Since the length controller is capable of selecting the appropriate length independent from outside control, the required interaction between the clock manager and external control circuitry is greatly reduced. However, the controller may be programmed to an initial value through the set signals to the C-elements.

5.6 - Additional Design Considerations

5.6.1 - VCO Design Considerations

When designing the VCO, there is importance in ensuring that K_O is linear for a wide range. This is obtainable through proper transistor sizing and circuit choice. Also, for a variable length application, K_O should be consistent for all possible lengths. Achieving this consistency can be difficult. If the VCO is comprised of equal delay cells and K_O varies, then the delay cells can be progressively sized to have an increased delay for the same control voltage.

If the VCO cannot be designed to achieve a consistent K_O, then the charge-pump bias control circuitry, Figure 4.6(a), can be manipulated to alter the value of the charge-pump current I_P. This allows the manipulation of the PLL model parameters so they remain consistent while K_O is variant. The PLL can be simulated and analyzed to determine the various values of K_O that may occur and design circuitry to automatically choose the appropriate settings of the charge-pump bias, given the length set by the controller, to maintain the desired PLL characteristics.

5.6.2 - Design Using Logical Effort

In order to maximize speed and minimize delay in critical paths of the circuit, the method of *Logical Effort* was employed [50]. With the goal of achieving a maximum operating frequency of several GHz, the critical path in the clock dividers and VCO multiplexer was identified and reduced using logical effort. This method was also used to match buffer delays for complementary signals.

The method of logical effort is a mathematical method for estimating delay in a CMOS circuit. It is capable of comparing delay estimates for different logic structures, specifying the proper number of logic stages on a path and determining the best transistor sizes for the logic gates. Since the method is easy to use, it is useful for evaluating alternatives during early stages of design.

5.7 - Clock Manager Implementation

A clock manager incorporating the RSPLL was designed and implemented in the 1.8V 0.18μm CMOS process by TSMC. The design is shown in Figure 5.12 and the die photograph is in Figure 5.13. The RSPLL can replace standard PLLs in existing clock manager designs with limited additional changes to the overall design of the clock manager or system in which it is used. A system designer may wish to take advantage of the ability to pre-determine and set the desired length, but this additional computation circuitry is unnecessary due to the fact that the length controller is capable of operating in an automated fashion. The settings for the divider in the feedback path may be reprogrammed on-the-fly without requiring any additional computation to set the proper length of the VCO associated with the new operating range.

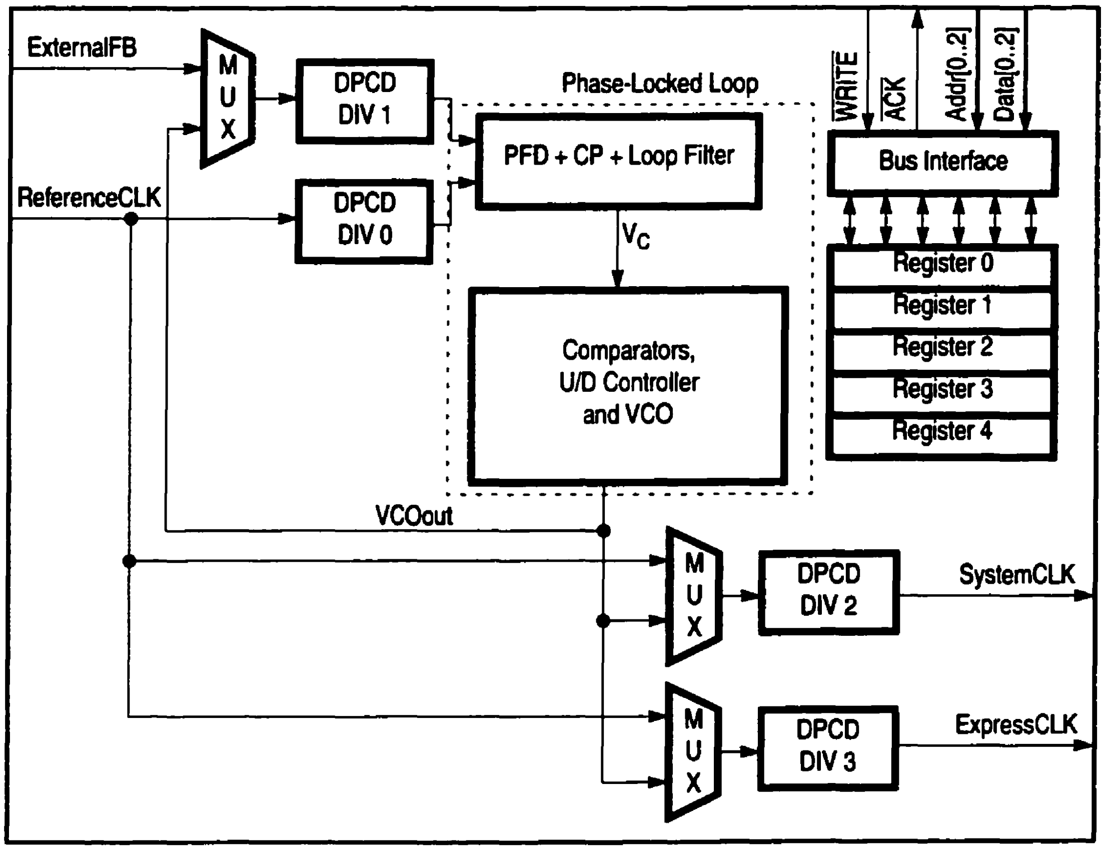


Figure 5.12: Programmable Clock Manager With RSPLL

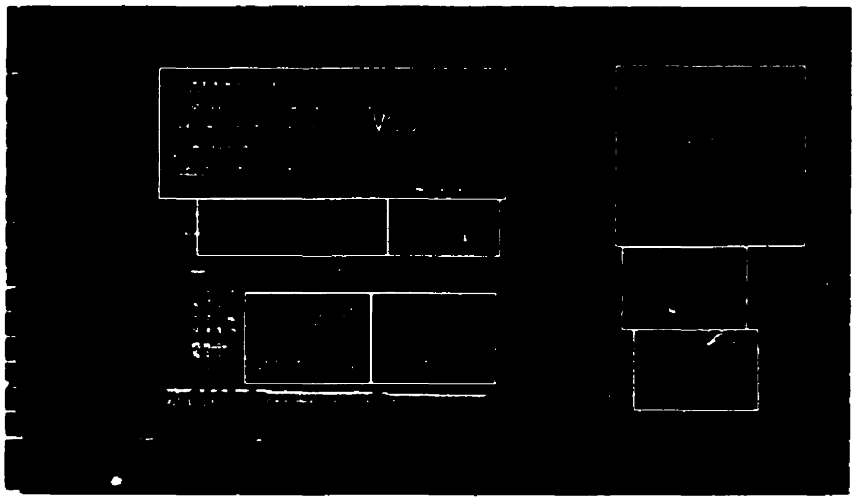


Figure 5.13: Die Photograph of Clock Manager With RSPLL Implementation

Before computing the necessary values for the charge-pump gain and the filter function, the VCO must be characterized, as shown in Figure 5.14. As can be seen from the graph, K_O is designed to be consistent between all lengths in the linear operating range. V_{TH} and V_{TL} were chosen, 1.7V and 0.9V respectively, to take advantage of the linear range while ensuring a slight overlap between the frequency ranges for adjacent lengths. With this VCO, the operating range of the PLL is simulated to effectively operate from 1.5MHz to 1.08GHz. The PLL can operate below 1.5MHz, but jitter becomes greatly increased and the duty cycle is no longer 50%, similar to the PLL described in the previous chapter. Also, one may notice that the slope of the graph increases greatly as the control voltage falls below 0.9V, which is more pronounced when the VCO length is reduced. This region of operation is thus avoided to prevent instability and drastic changes to the PLL design parameters that occurs due to an increase in K_O.

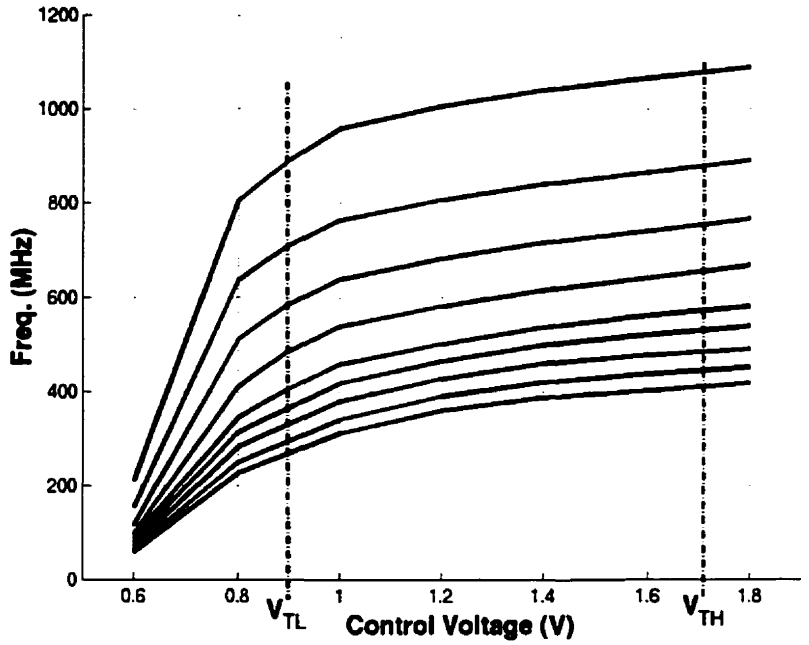


Figure 5.14: V_C vs. Frequency for 1-9 Delay Cells

The value for K_O was estimated from the results of the extracted simulations, Figure 5.14. The parameters for the PLL were chosen and the design was then completed and implemented. The extracted simulations show that the PLL including the length controller is fully functional and properly locks on to the appropriate frequency, as shown in Figure 5.15. The frequency of ReferenceCLK was set to 5 MHz and the feedback divider was programmed to a division value of 15. SystemCLK is shown to lock on to the frequency of 75 MHz, as desired. Initially, the VCO is in a reset state and thus does not oscillate. The control voltage charges until it surpasses the threshold and meets the supply voltage level. By surpassing V_{TH} of 1.7V, the length controller is activated and waits to ensure against an overshoot. After the appropriate delay, the length controller sets the VCO to the longest length of 16 taps (delay cells). This event is identified by the commencement of oscillation from SystemCLK. After two additional length changes, the control voltage begins to lock on to the new frequency. Once V_C passes V_{TH} on the way down, the length controller is deactivated and the current length of 14 taps is maintained.

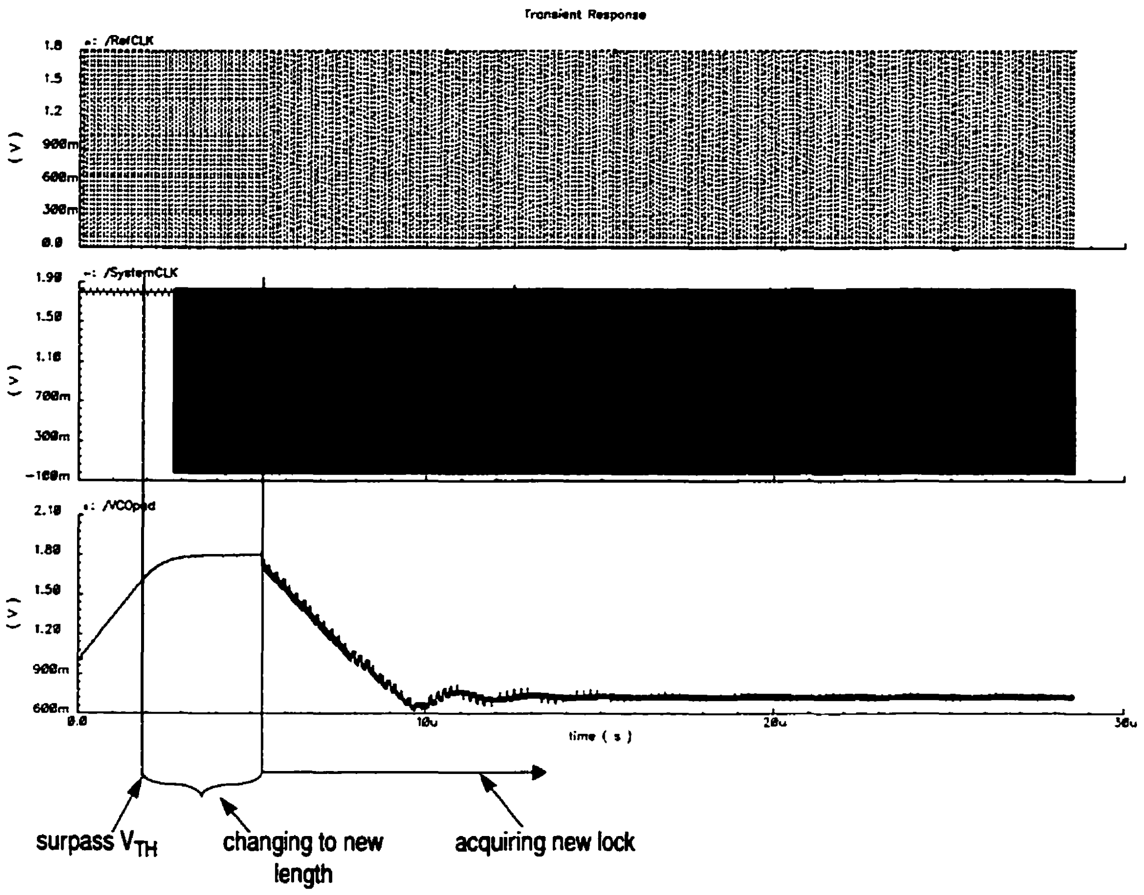


Figure 5.15: Simulation of the RSPLL Locking to a Frequency of 75 MHz. The Reference Frequency is 5 MHz and the Feedback Multiplication Factor is 15.

Magnified versions of Figure 5.15 can be seen in Appendix E.

5.7.1 - Experimental Results

Due to a layout error, only simulation results for the implementation of the RSPLL are available. Because of the lack of a poly-poly capacitor in TSMC's 0.18μm process, an interesting high capacitance structure [51] was tried so as to reduce the area necessary for the filter. The capacitor is structured in a form similar to the fingered layout used to match transistors in analog layout. The structure takes advantage of not only the capacitance between adjacent metal layers in layout, but also the parasitic capacitance between two parallel wires on the same metal layer.

Unfortunately, due to the high level of complexity of the capacitor structure and because the layout of the capacitive structure was performed by a colleague, an error occurred where a short was created. Due to this short, the loop filter became useless and the PLL did not function properly. An attempt to correct the error with a Focused Ion Beam resulted in damage to the tail transistors in the delay cells, thus preventing operation of the VCO. With the error corrected in the layout, the extracted simulations demonstrate operation as expected. Another colleague will pursue another fabrication of the RSPLL with a corrected version of the loop filter.

With the delay cell structure and VCO ranges similar to the implementation discussed in Chapter 4, assumptions can me made about the performance of the RSPLL. Since the RSPLL was implemented in a more advanced process, it is conceivable that the performance will improve upon the SoC implementation of the Programmable Clock Manager. The power consumption is simulated to be approximately 8.9mW @ 330 MHz.

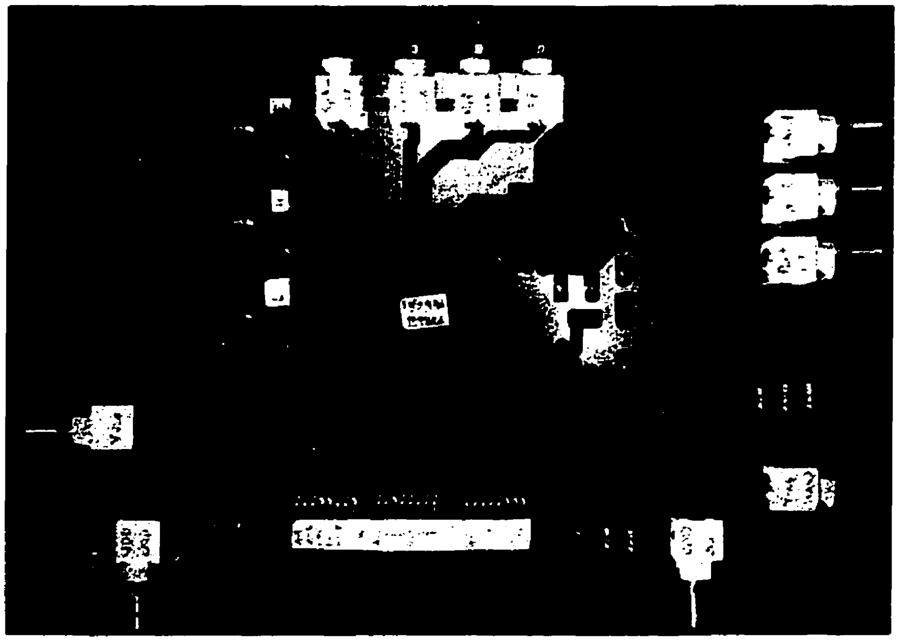


Figure 5.16: PCB Layout to Test the RSPLL Implementation

5.8 - Summary

As the requirement of a wider operating range becomes increasingly important for SoC designs, clock managers are required to produce an extensive range of clock bandwidths with minimal jitter. The frequency range shifting PLL contains a VCO with an increased operating range, a linear operating region, reduced VCO gain, and minimal jitter. This circuit can replace PLLs in existing clock managers to decrease software or control complexity.

Chapter 6 - SoC Implementation

As complexity and integration of System-on-a-Chip (SoC) circuits continue to increase, the creation of efficient SoC research platforms have become a necessity to explore large scale system design issues. The Semiconductor Industry Association (SIA) predicts that integrated circuits will have up to 10 000 separate clock signals by 2005, and that value will increase to 100 000 by 2010. The focus of the Managed Clock System-on-Chip (MCSoC) project is the creation of high quality clock management schemes and a comprehensive testbed for exploring various applications of clock management.

In order to incorporate components developed in previous chapters and to optimize functionality from a research standpoint, a system design was chosen to consist of a simple processor with memory and communication subsystems. MCSoC maximizes the ability of multiple subsystems to operate independently, at varying clock rates. Such a system will provide a medium to test and verify processor controlled dynamic clock management for low power purposes, before incorporating them into larger SoCs.

6.1 - MCSoC Organization

MCSoC is comprised of five main subsystems, interconnected by an asynchronous bus, as in Figure 6.1. An integrated processor interfaces with the bus through the Processor Bus Controller (PBC). Also included in the SoC is a memory core, a Universal Serial Bus (USB) host controller and a Programmable Clock Manager (PCM). The second iteration

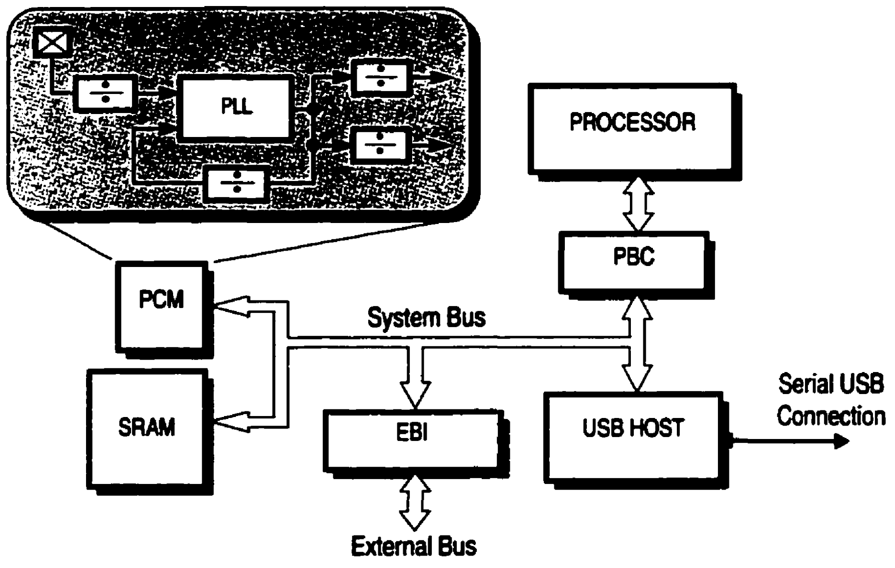


Figure 6.1: System Level Overview of MCSoC

of the PCM created in Chapter 4 is incorporated into MCSoC. The system clock from the PCM drives the clock distribution network for the processor, and the express clock drives the USB host.

The processor is based on the Programmable Integrated Controller (PIC) from Microchip Technology Inc. [52]. This compact processor has been redesigned for MCSoC. The PBC was created as a separate component from the processor to enable independent control, thus increasing its versatility. To complement the memory available in the processor, and for embedded memory research, MCSoC includes a memory core comprised of Static Random Access Memory (SRAM). The SRAM can be tested and operated independently from the other subsystems within MCSoC.

For the purposes of external communication, a USB host controller has been incorporated into the SoC. The USB protocol was chosen due to its multiple operating speeds. MCSoC is intended to act as a low power device, possibly powered through the attachment to the USB bus.

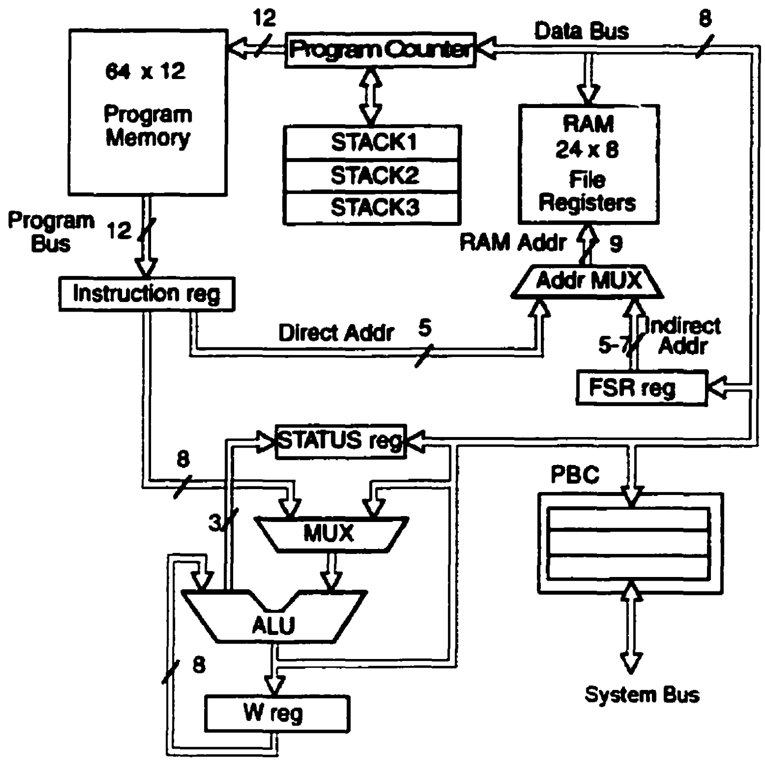


Figure 6.2: Processor Architecture

6.2 - Processor Design

MCSoC's 8-bit programmable microcontroller, shown in Figure 6.2, is an area efficient Reduced Instruction Set Computer (RISC) architecture. The 12-bit instruction set provides 27 single word instructions with a simple opcode structure. In this Harvard architecture processor, program and data are accessed through separate internal buses.

The microprocessor executes instructions in two pipeline stages. Almost all instructions require only one CPU cycle to execute. However, the branching operations can consume up to two CPU cycles. The power management instructions, modes and circuitry in the original PIC have been redefined in favour of the dynamic clock management feature offered by the PCM. The PCM is designed to be complete and easy to use, and thus does not require additional instructions to be added to the PIC's existing set or additional hardware alterations in the SoC design.

Since the processor is incorporated into a larger SoC design, the PIC's serial Input/Output (I/O) communication, necessary for stand-alone operation, has been replaced by a new internal system bus interface. The PBC provides a more flexible and expandable interface, and the capability to perform Direct Memory Access (DMA). The DMA transfers occur between the system bus and the program memory, or the bus and processor registers. The processor can continue operation during these transfers without providing additional control past the initial set-up. The PCB is also used as a simple debugging interface that improves coverage and accessibility.

The program memory is in the form of an SRAM containing 64 cells, each 12 bits in width. The bootstrap code is built into the program memory upon start-up and can be overwritten in order to increase available operating space. With every reset of the processor, the bootstrap code is initialized in the program memory. This code requests the initiation of a DMA transfer from an address of an external device into the program memory. This procedure loads the operating program upon start-up, and also allows the program to be updated during operation.

6.3 - System Bus and External Communication

During normal operation, the PBC assumes the role of a bus master. The processor programs the PBC to fetch and transfer data through the asynchronous internal system bus. The system bus uses pull-up resistors to force the bus lines to logical '1' when not in use, as in Figure 6.3. Also, the PBC uses open-drain connections to the system bus. As a

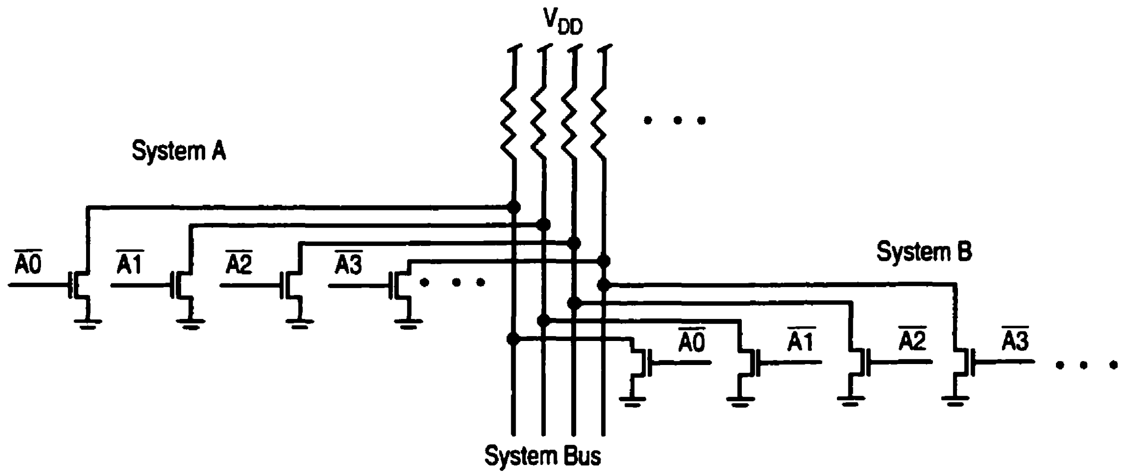


Figure 6.3: Connections to System Bus With Pull-up Resistors to Maintain a Logical '1' and Pulled Down by NMOS Transistors

result, other devices can have control of the bus when the PBC is in the reset or disabled state.

MCSoC may communicate to external devices through two ports, the USB host and the External Bus Interface (EBI), as shown in Figure 6.1. The USB host is intended to be the main device for this purpose, with the EBI as a secondary choice but the main access for debugging. The USB protocol uses serial communication with handshaking. The address queried during bootstrapping for obtaining the operating program is found on a device addressed by the USB host. The communication protocol for the USB host is generated in hardware, but general control and timing is delegated by the processor. There are several control registers in the USB host that are updated and changed by the processor, thus transferring much of the control to software. This increases flexibility in the design by allowing changes or improvements to the protocol to be more easily implemented, and reduces the amount of dedicated hardware required.

The internal system bus is connected to bi-directional I/O pins through the EBI. There are three functions performed by the EBI. The first is to allow external devices to have control over the system bus. Therefore, the PBC must be in a reset state to avoid conflicts. This function is desirable during testing, so that the PCM, SRAM and USB host can be tested individually without interference from, or having to program the processor and PBC. Secondly, the EBI can be used as a connection to external devices through memory mapping to external control registers. The final function is observation of the interactions between subsystems on the internal bus.

6.4 - Development

The MCSoC project was a team effort between five people. In the initial stages, Field Programmable Gate Arrays (FPGAs) were used extensively, followed by the development of custom designed *pilot chips*. The lessons learned and ideas developed were amalgamated into a final design.

FPGAs are a very attractive platform for design, prototyping, and debugging of digital blocks. In the MCSoC project, they were used to develop and confirm digital system design ideas before silicon fabrication. As discussed in Chapter 3, the development and confirmation of the theories behind dynamic clock management were performed in FPGAs. Also, the processor core was initially developed and implemented in an FPGA.

FPGAs cannot be used to design and characterize analog and time-critical digital components. Thus, the development plan included the design and manufacture of pilot chips. These smaller designs ensure proper operation of the more complex components without wasting silicon real estate through multiple attempts at a large SoC design. The benefit of having pilot chips was readily observed from a reduction in jitter between the pilot PCM and the second iteration implemented in the final SoC.

Two such pilot chips were used in this project, with the additional objective of facilitating substrate coupling experiments and modeling performed by other students working on the project. Also, the pilot chips provided the means to improve the clock management PLL design. The first PCM design was fabricated in the 3.3V 0.35μm CMOS technology and design consumed an area of 3mm by 3mm with I/O pads and buffers.

6.5 - PCM Incorporation Into the SoC

Aside from changing the bus interface to be compatible with the system bus, few alterations to the PCM design in Chapter 4 are necessary for incorporation into MCSoC. The PCM and the appropriate software make MCSoC capable of performing dynamic clock management. With the addition of the Range Shifting Phase-Locked Loop (RSPLL) design discussed in Chapter 5, the PCM becomes a powerful tool for reducing power consumption in SoCs based on demand.

6.5.1 - PCM Positioning Within MCSoC

When amalgamating the PCM with an SoC, issues regarding placement and connectivity become a concern. Skew within a clock distribution network can be reduced through layout techniques such as a symmetric H-tree distribution network [20,53,54]. The origin of the H-tree distribution is the at the center of the IC. Therefore, placing the clock generation circuitry as close to this central location may be the most beneficial for reducing overall skew. However, with the PLL being an analog component, adequate shielding will be necessary to remove the large amount of substrate noise generated by the surrounding digital circuitry [55,56]. In the case of MCSoC, the PCM was placed in the corner of the IC to reduce the substrate noise effects on the PLL. Guard bands [55,56,57] were placed between the PCM and the digital circuitry, in addition to extra bands around the more sensitive components of the PLL.

6.5.2 - Driving the Clock Domains

Before assigning the output clocks from the PCM, MCSoC requires partitioning into separate clock domains. Since an SoC is a compilation of multiple independent systems, there are natural boundaries that can be used to create separate clock domains. In the case of MCSoC, the processor and PBC operates as an independent entity from the USB host. Therefore, the processor and PBC forms one clock domain and a second clock domain drives the USB host.

The system clock was chosen to drive the clock distribution network for the processor and the PBC. The express clock drives the USB host since the operating frequency for the host is usually less than that of the processor. As discussed in Section 4.1, the system clock output multiplexes between the reference clock, the output of the Voltage Controlled Oscillator (VCO), and a divided version of the output of the VCO. However, the express clock output can divide the reference clock or the output of the VCO. Therefore, the express clock is capable of producing much lower frequencies than the system clock when the VCO is programmed to produce high frequencies.

6.5.3 - The Feedback Signal

The feedback for the PCM was taken from a point within the domain of the system clock, which drives the processor. This point was chosen so that the quantity of skew is reduced within the system clock domain, thus improving the operation of the processor. However, if the PCM is programmed to divide the system clock, and the feedback signal for the PLL comes from the system clock domain, then frequency multiplication will occur instead of the expected division. Therefore, if a frequency division is desired in the system clock domain, then the feedback signal must originate from a loop within the PCM, or a multiplexer must be inserted so the feedback may come from another clock domain.

6.5.4 - Control

The PCM settings are determined by software in the processor and set through the system bus. The flexibility inherent in software allows various speed setting algorithms [25] to be explored in MCSoC. Control logic is discussed in Section 3.3.2. In the case of MCSoC, some methods of control may be determined by the software based on the functions being performed. There may be certain instances where the software designer has prior knowledge of how long the execution of a function may take and what speed is necessary to execute that function within the time constraints of the overall program. Therefore, the designer may purposefully call a function to set a specified PCM setting. Otherwise, another function may be periodically called to assess the current demand and change the PCM settings based on the results of the assessment. Demand may be measured by the quantity of time spent in the idle state, or executing the `nop` (no operation) command.

6.6 - Testing, Verification and Implementation

MCSoC included many circuit alterations and additions in order to facilitate the debugging process. In some cases, design for debugging allowed near complete functional coverage. The major entry for testing is through the EBI. By using the bi-directional bus connections, an external device can test each subsystem within MCSoC. Each subsystem also has individual reset lines, thus allowing all other components to be in a reset state when conducting tests on one specifically.

6.6.1 - Test Plan

For the purposes of efficiency, a test plan was used to organize efforts, share access to the device and test equipment, and to ensure incremental debugging. The debugging process began with the system bus and PCM, followed by the SRAM, then the processor and PBC.

The USB is tested last due to its complex serial communication protocol, and because the EBI can be used for external communication.

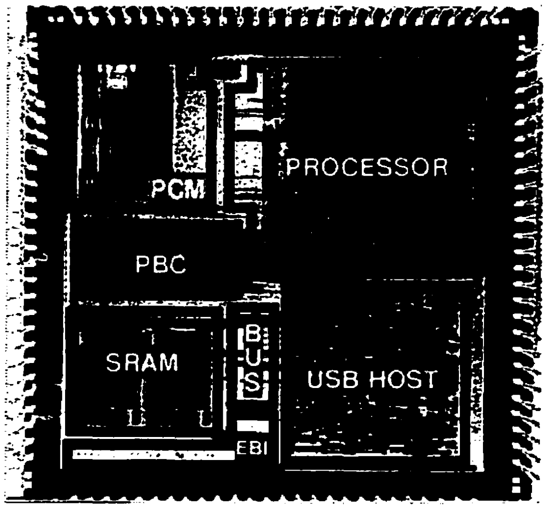
6.6.2 - The IEEE 488 and VXI Bus Measurement Setup

The test plan included the development of C code used for controlling the HP1401 VXI bus mainframe, the primary test device for communicating with the MCSoC system bus through the EBI. The VXI bus is capable of generating and reading several digital and analog signals. For the purposes of MCSoC, the VXI bus used 32 bi-directional digital pins and seven digital control pins. Although the VXI bus generators and analyzer modules are limited to an operational speed of 20MHz, testing was possible due to the full handshake protocol of the system bus. Additional analog test equipment is connected through an IEEE 488 bus, and in early stages, the 50 MHz HP16500B logic analyzer was used together with its signal generator modules.

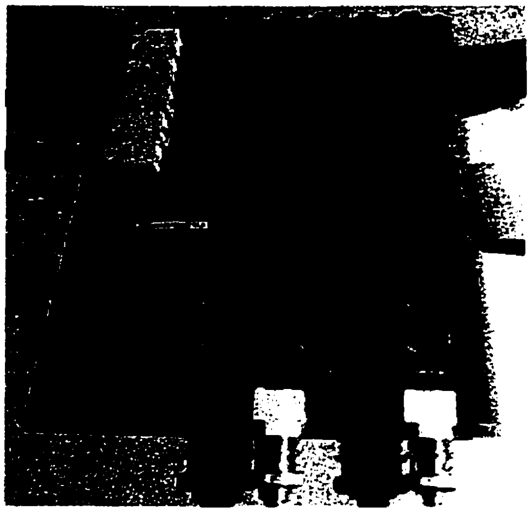
The test process was established in advance so all desired options could be built into the C code during its development. This resulted in an efficient hierarchical code structure that is easily maintained and understood by successive designers. The test code was written so that the specific functions for controlling the test equipment were transparent to designers when setting up programs to test their components. Designers need to concern themselves only with high level functional calls for bus operations such as master write, slave read and observation. With the hardware functionality of MCSoC confirmed, a C program is used to automatically load the assembly level code from a text file into the processor by merely running the executable and specifying the file on the command line.

6.6.3 - Printed Circuit Board Design

A Printed Circuit Board (PCB), shown in Figure 6.4(b), was created to preserve the integrity of the power, ground, clock and digital signals, and to provide a platform for interconnecting the multiple buffers to the MCSoC IC and test equipment connectors. Initial tests, using a generic test head and HP16500B logic analyzers, were conducted to ensure functionality before proceeding with the design and construction of the PCM. Many tri-state buffers are needed on the PCB due to the bidirectional connections of the



(a)



(b)

Figure 6.4: Implementation of MCSoC (a) Die Photo (b) PCB Test Board

EBI and VXI bus. Buffers are also needed to step down the 5V signal of the VXI bus to the 3.3V operating voltage of the MCSoC IC.

6.6.4 - Implementation

The final MCSoC chip was fabricated in TSMC's 3.3V 0.35mm CMOS process. The design occupies 4mm by 4mm, including the I/O pads and buffers. The layout organization is shown in the die photo, Figure 6.4(a). The test results of MCSoC's PCM can be found in Section 4.3.

Table 3: Contributions to MCSoC

Task	Participant/Designer
Supervisor	Prof. Zeljko Zilic
Team Lead	Ian Brynjolfson
FPGA Implementation of Processor	Jean-Samuel Chenard and Hsin Yun Yao
Adaptation and High Level Architectural Design of MCSoC Processor	Ian Brynjolfson and Yanai Danan
MCSoC Processor Implementation	Yanai Danan
USB Host Design	Ian Brynjolfson

Table 3: Contributions to MCSoC

Task	Participant/Designer
Embedded SRAM Design	Boris Polianskikh
PCM Design	Ian Brynjolfson
Noise Modeling and Analysis	Henry H. Y. Chan
Test Plan	Ian Brynjolfson
VXI Code Implementation	Yanai Danan
PCB Design	Ian Brynjolfson
MCSoC Testing	Ian Brynjolfson and Yanai Danan

6.7 - Summary

The SoC implementation of dynamic clock management, and the circuitry to generate the appropriate clock signals, is implemented in the MCSoC project. This platform eases research in the areas of large scale development and the confirmation of dynamic clock management ideas. With a simple generic processor, embedded SRAM, communication systems, and the PCM, MCSoC is has been optimized for functionality from a research standpoint. The design also maximizes the ability of multiple subsystems to operate independently, at varying clock rates.

Currently, MCSoC is clock configurable by software access to PCM control registers. Further developments in software will continue to build additional operations, including new design for testability modes. The capabilities of MCSoC will be used to testbench clock rate scheduling algorithms that determine operating speed based on demand for the purposes of power reduction. Further software explorations will include scheduling of power management and system debug utilities. Hardware research will continue into successive clock management and embedded memory designs, as well as the substrate reduction methods. These future developments will be carried out by students furthering the work presented in this thesis.

Chapter 7 - Conclusion

To combat the increasing demands for reduced power consumption, we have investigated an interesting method of reducing power consumption without paying the price in performance. This thesis explored dynamic clock management as a potentially powerful system-level technique that varies clock rate based on demand, thus taking advantage of the relationship between clock speed and power consumption. We developed clock generation circuitry essential for dynamic frequency synthesis. These clock managers enable dynamic clock management with limited alterations to existing systems.

We presented a clock divider capable of changing frequency on-the-fly, as demanded by dynamic clock management. The Dynamic Programmable Clock Divider (DPCD) is capable of changing frequencies without exhibiting glitches or presenting an inconsistent duty cycle. The DPCD can easily replace the dividers in existing clock manager designs, thus permitting dynamic clock management. The DPCD was implemented in a custom designed clock manager and also in the user logic of a Field Programmable Gate Array (FPGA). We therefore demonstrated that dynamic clock management is possible in existing FPGAs by simply including the DPCD and some control logic.

Another circuit that can be added to the present design of clock managers is the Range Shifting Phase-Locked Loop (RSPLL). This variation to the traditional design of a Phase-Locked Loop (PLL) reduces output jitter and increases the flexibility in choosing design parameters. These improvements are achieved by a reduction in the Voltage Controlled

Oscillator (VCO) gain while still maintaining a wide operating range. The RSPLL is capable of varying its output clock's frequency operating range, by changing the length of the delay line in its VCO.

The RSPLL also simplifies the design of clock managers, by offering a wide operating range and reduced jitter. These benefits do not come at the expense of increased external control. Since the RSPLL is self-contained with automated length control, the main system control circuitry responsible for programming the clock manager does not have to be concerned with possible process or temperature variations that may alter PLL operation.

The PCM design with an RSPLL and DPCDs is therefore a powerful, scalable, complete and easy to use component that may be added to a clock distribution network without additional infrastructure support. Also, to perform dynamic clock management, the control software does not need the special instructions to be included in the processor's instruction set. These benefits are shown by our exploration into the Managed Clock System-on-a-Chip (MCSoC) platform.

This thesis has developed the infrastructure needed in the use of dynamic clock management for the purposes of reducing energy consumption in Integrated Circuits (ICs). The clock management circuitry presented here can be used in existing system designs to enable dynamic clock management. The PCM with an RSPLL and DPCDs is not limited to dynamic clock management for low power applications, but can be used for all clock management applications. With the circuitry presented in this thesis, the next step in dynamic clock management research is the further development towards solving the system level issues.

Appendix A - Calculating PLL Design Parameters

The models used for classical Phase-Locked Loop (PLL) analysis are based on assumptions regarding the behavior of the loop and the component hardware. The loop is assumed to be frequency clocked and operates in continuous time. Concerning the hardware, the Voltage Controlled Oscillator (VCO) signal source is an ideal integrator and the phase detector is a linear adder. The s-domain representation used in the analysis of the PLL is shown in Figure A.1.

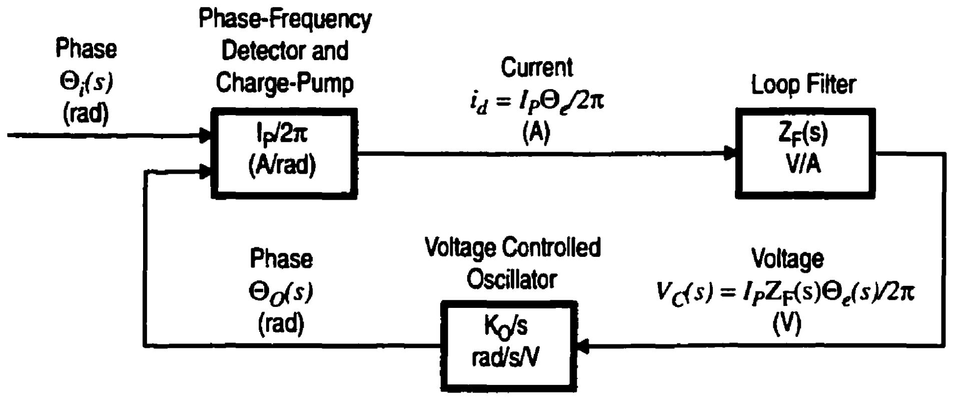


Figure A.1: s-Domain Representation of a PLL

Using the charge-pump model depicted in Figure A.2(a), a pump current I_P is given to the filter impedance Z_F for a time t_p on each cycle. Since each cycle has a duration of $2\pi/\omega_i$ seconds, then the average error current over a cycle is derived as:

$$t_p = |\Theta_e|/\omega_i \tag{A.1}$$

$$i_d = I_P\Theta_e/2\pi \tag{A.2}$$

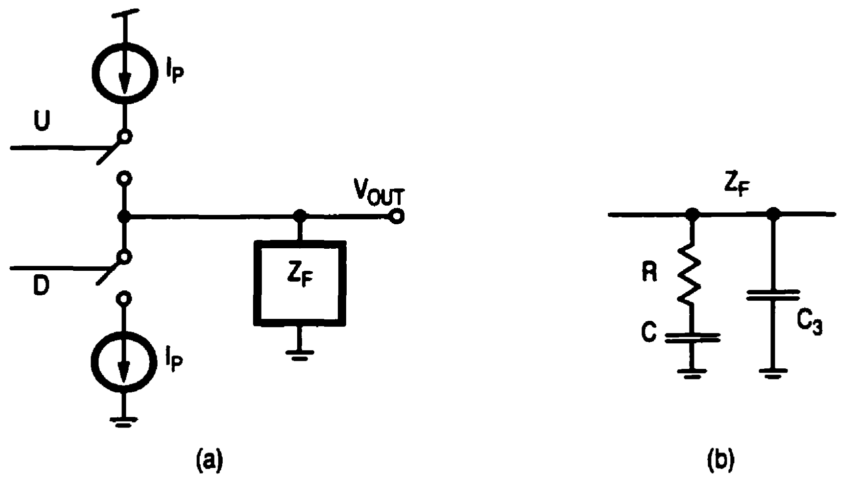


Figure A.2: (a) Charge Pump Model Used in PLL Analysis
(b) Filter for Third Order Loop

This i_d is also the error current averaged over many cycles, provided that both inputs are periodic. The control voltage for the VCO is given by:

$$V_C(s) = I_d(s)Z_F(s) = I_P Z_F \Theta_e(s)/2\pi \tag{A.3}$$

where $I_d(s)$ is the Laplace transform of $i_d(t)$, and similarly for the other symbols. The only condition for which these transfer functions are applicable occur when the loop is in a locked state. This is due to the non-linearities that occur when the loop is out of lock. For a locked loop, the VCO phase is given by:

$$\Theta_O(s) = K_O V_C(s)/s \tag{A.4}$$

where K_O is the VCO gain in radians/second/volt.

The role of the PFD is to determine the phase error Θ_e between the input Θ_i and feedback (or VCO output) Θ_O clock signals. This error is represented by:

$$\Theta_e(s) = \Theta_i(s) - \Theta_O(s) \tag{A.5}$$

The previous expressions lead to the loop transfer function:

$$\frac{\Theta_O(s)}{\Theta_i(s)} = \frac{K_O I_P Z_F(s)}{2\pi s + K_O I_P Z_F(s)} = H(s) \tag{A.6}$$

H(s) can also be derived from the open loop transfer function given by:

$$G(s) = \frac{I_P}{2\pi} \cdot Z_F(s) \cdot \frac{K_O}{s} \tag{A.7}$$

which then leads to Equation (A.6) by substituting Equation (A.7) in the following equation:

$$H(s) = \frac{G(s)}{1 + G(s)} \tag{A.8}$$

With the establishment of the loop transfer function, the loop filter function can be substituted in for $Z_F(s)$. For the loop filter in Figure A.2(b), the filter impedance is:

$$\begin{aligned} Z_F(s) &= \left(R + \frac{1}{sC}\right) \cdot \frac{\frac{1}{sC_3}}{R + \frac{1}{sC} + \frac{1}{sC_3}} \\ &= \frac{R + \frac{1}{sC}}{1 + sC_3R + \frac{C_3}{C}} \end{aligned} \tag{A.9}$$

With the assumption that $C \gg C_3$, then $Z_F(s)$ reduces to:

$$Z_F(s) = \frac{1}{sC} \cdot \frac{1 + sRC}{1 + sRC_3} \tag{A.10}$$

The open loop transfer function becomes:

$$G(s) = \frac{I_P}{2\pi} \cdot \frac{1}{sC} \cdot \frac{1 + sRC}{1 + sRC_3} \cdot \frac{K_O}{s} \tag{A.11}$$

This third-order system has two poles at $\omega = 0$, a zero at $\omega_1 = 1/RC$, and another pole at $\omega_3 = 1/RC_3$.

Through substitution of Equation (A.10) into Equation (A.6), *H(s)* becomes:

$$H(s) = \frac{1}{RC}\cdot\frac{K_O I_P R}{2\pi}\cdot\frac{1+sRC}{s^3 RC_3 + s^2 + s\frac{K_O I_P R}{2\pi} + \frac{1}{RC}\cdot\frac{K_O I_P R}{2\pi}} \tag{A.12}$$

A simplification can be made using the following definition of loop gain:

$$K = \frac{I_P K_O R}{2\pi} \tag{A.13}$$

Therefore, using the definition for *K*, ω_1 and ω_3, *H(s)* can be represented by:

$$H(s) = K\omega_1\cdot\frac{1+\frac{s}{\omega_1}}{\frac{s^3}{\omega_3} + s^2 + sK + K\omega_1} \tag{A.14}$$

By making the assumption that $\omega_3 \gg \omega_1$, then s^3/ω_3 can be simplified out of Equation (A.14), therefore being reduced to a second-order system. In circuit and control theory, common practice dictates that the denominator of a transfer function should be written in the *normalized form*: $s^2 + 2\zeta\omega_n s + \omega_n^{\ 2}$, where ω_n is the natural frequency and ζ is the damping factor. By applying the normalized form to Equation (A.14), the equations for ω_n and ζ become:

$$\omega_n = \sqrt{\frac{I_P K_O}{2\pi C}} \tag{A.15}$$

$$\zeta = \frac{1}{2\omega_1}\sqrt{\frac{I_P K_O}{2\pi C}} = \frac{\omega_n}{2\omega_1} \tag{A.16}$$

Critical damping occurs when $\zeta > 1/\sqrt{2}$, but only when $\zeta > 1$ does the transfer function not exhibit an overshoot of the final lock value, as shown in Figure 2.10. There are also several useful relations:

$$K = 2\zeta\omega_n \tag{A.17}$$

$$\frac{K}{\omega_1} = 4\zeta^2 \tag{A.18}$$

$$K\omega_1 = \omega_n{}^2 \tag{A.19}$$

Any two of the three parameters completely define the linearized, time-averaged behavior of the PLL. However, stability is influenced by an additional parameter, the reference frequency. This PLL model implies stability with the following inequality [42,38]:

$$\omega_{Ref} > K_O I_P R \tag{0.1}$$

where ω_{Ref} is the reference signal in radians/second.

In the case where the feedback signal is divided by N in order to perform multiplication, the natural frequency and damping factor become:

$$\omega_n = \sqrt{\frac{I_P K_O}{2\pi C N}} \tag{A.20}$$

$$\zeta = \frac{1}{2\omega_1}\sqrt{\frac{I_P K_O}{2\pi C N}} \tag{A.21}$$

This implies that the damping factor is influenced by the division value of N. Higher values of N imply a decrease in the damping factor. Including N in the stability model creates a new inequality:

$$\omega_{Ref} > \frac{K_O I_P R}{N} \tag{0.2}$$

which implies that increasing the division factor N actually improves the stability of the system if the other design factors remain constant.

Appendix B - PCM Register Configuration

The register map for the PCM included in MCSoC.

6 registers X 8 Bits each

```
Register0 - Divider0 Programming
  Bits[3:0] 4-Bit Divider, DIV0, Value.
            This value enables the input clock to immediately
            be divided by a value from 1 to 8. A 0 value
            (the default) indicates that DIV0 is bypassed
            (no division).

  Bits[6:4] Reserved

  Bit7      DIV0 Reset Bit
            DIV0 may not be reset by GSRN depending on the value
            of register 4, bit 6. This bit may be set to 1 to
            reset DIV0 to its default value.


Register1 - Divider1 Programming
  Bits[3:0] 4-Bit Divider, DIV1, Value.
            This value enables the feedback clock to immediately
            be divided by a value from 1 to 8. A 0 value
            (the default) indicates that DIV1 is bypassed
            (no division).

  Bits[6:4] Reserved

  Bit7      DIV1 Reset Bit
            DIV1 may not be reset by GSRN depending on the value
            of register 4, bit 6. This bit may be set to 1 to
            reset DIV1 to its default value.


Register2 - Divider2 Programming
  Bits[3:0] 4-Bit Divider, DIV2, Value.
            This value enables the delay line output clock driven
            into SystemCLK to immediately be divided by a value
            from 1 to 8. A 0 value (the default) indicates that
            DIV2 is bypassed (no division).
```

Bits[6:4] Reserved

Bit7 DIV2 Reset Bit
DIV2 may not be reset by GSRN depending on the value of register 4, bit 6. This bit may be set to 1 to reset DIV2 to its default value.

Register3 - Divider3 Programming
Bits[3:0] 4-Bit Divider, DIV3, Value.
This value enables the delay line output clock driven into ExpressCLK to immediately be divided by a value from 1 to 8. A 0 value (the default) indicates that DIV2 is bypassed (no division).

Bits[6:4] Reserved

Bit7 DIV3 Reset Bit
DIV3 may not be reset by GSRN depending on the value of register 4, bit 6. This bit may be set to 1 to reset DIV3 to its default value.

Register3 - PLL Biasing/Reset Logic
Bits[5:0] PLL Charge Pump Bias Value
All 0 = Full Bias Current Flow
One 1 = 4/5 Current Flow
Two 1 = 3/5 Current Flow
Three 1 = 2/5 Current Flow
Four 1 = 1/5 Current Flow
All 1 = No Current Flow

Bit6 PCM GSRN Enable Bit
0 = normal GSRN operation
1 = GSRN has no effect on PCM logic, so clock processing will not be interrupted by a chip reset.
Default is 0.

Bit7 PCM Reset
A value of 1 resets all PCM logic.

Register4 - PCM Control Programming/Clock Selection
Bit0 PLL Phase Detector Feedback Input Selection Bit
0 = Feedback from programmable delay line output.
1 = feedback signal from routing/FeedbackCLK
Default is 0.

Bits[2:1]	SystemCLK Output Source Selector.
	00: PCM input clock, bypass path through PCM
	01: delay line output
	10: divided (DIV2) delay line output
	11: reserved
	Default is 00.
Bits[4:3]	ExpressCLK Output Source Selector.
	00: PCM input clock, bypass path through PCM
	01: delay line output
	10: reserved
	11: reserved
	Default is 00.
Bits[7:5]	VCO Length
	000: 32 Delay Cells
	001: 16 Delay Cells
	010: 8 Delay Cells
	011: 4 Delay Cells
	100: 2 Delay Cells
	101: 1 Delay Cell

Appendix C - Jitter Measurements

The following jitter measurements were made using a Tektronix TDS 8000 Digital Sampling Oscilloscope. 250 waveform acquisitions were made for each measurement. The input clock signals were generated by an HP 81130A 400/660 MHz Pulse-/Pattern Generator, which exhibited 32.42 ps pk-pk and 3.927 ps RMS jitter at a frequency of 50 MHz.

Length = 32 cells				
Frequency [MHz]	Tracking Jitter		Apx. Duty Cycle [%]	Power [mW]
	rms [ps]	pkpk [ps]		
0.100	5 304	18 200	97	130
0.250	4 767	16 800	93	130
0.500	5 983	20 400	85	130
0.750	7 319	28 600	80	100
0.900	7 915	34 000	75	100
1.50	4 150	17 700	78	100
2.00	3 707	14 600	70	100
5.00	895.6	4 000	58	100
10.00	412.2	2 050	50	130
20.00	374.6	1 740	50	165
30.00	235.7	1 048	50	230
40.00	145.0	980.0	50	200
50.00	85.33	888.0	50	200
60.00	1.5.5	644.0	50	230
70.00	85.23	768.0	50	250
73.00	125.9	896.0	50	265
55	76.87	544.0	50	200

Length = 16 cells				
Frequency [MHz]	Tracking Jitter		Apx. Duty Cycle [%]	Power [mW]
	rms [ps]	pkpk [ps]		
0.100	5 444	19 100	98	130
0.250	3 389	9 400	95	130
0.500	4 145	12 800	89	130
1.00	5 107	19 000	85	130
2.00	3 043	14 100	76	100
5.00	985.3	4 500	63	100
10.00	373.9	1 960	50	115
20.00	127.8	630.0	50	165
30.00	108.6	690.0	50	165
40.00	155.6	710.0	50	200
50.00	97.20	790.0	50	200
60.00	122.8	500.0	50	200
70.00	53.65	470.0	50	230
80.00	7.515	56.64	50	265
90.00	66.48	448.0	50	300
100.0	82.20	532.0	50	300
105.0	121.9	816.0	50	300
81.00	10.42	48.00	50	265

Length = 8 cells				
Frequency [MHz]	Tracking Jitter		Apx. Duty Cycle [%]	Power [mW]
	rms [ps]	pkpk [ps]		
0.100	189.9	930.0	97	100
0.250	163.4	920.0	96	100
0.500	530.5	3 080	94	85
0.750	215.1	1 360	91	66
1.00	217.2	1 420	88	66
2.00	154.9	910.0	80	66
5.00	63.02	388.0	70	66
10.00	43.72	248.0	60	66
20.00	23.09	122.0	50	130
30.00	30.35	164.0	50	165
40.00	14.12	90.00	50	130
50.00	15.45	68.00	50	130
60.00	10.26	62.00	50	165
70.00	9.283	54.00	50	200
80.00	8.663	48.00	50	230
90.00	7.234	44.00	50	265
100.0	9.426	60.00	50	265
110.0	7.868	50.00	50	265
120.0	10.15	54.00	50	300
130.0	9.469	50.00	50	300
137.0	10.45	50.00	50	330

Length = 4 cells				
Frequency [MHz]	Tracking Jitter		Apx. Duty Cycle [%]	Power [mW]
	rms [ps]	pkpk [ps]		
0.100	4 395	13 700	99	130
0.250	4 766	13 200	97	130
0.500	2 451	7 200	95	130
0.750	3 140	10 800	93	130
1.00	3 170	13 100	92	130
2.00	3 110	13 100	85	130
5.00	1 749	5 900	75	130
10.00	426.9	2 300	65	130
20.00	187.7	1 000	60	165
30.00	292.3	1 240	50	200
40.00	158.9	710.0	50	165
50.00	47.95	360.0	50	165
60.00	79.57	428.0	50	200
70.00	47.29	256.0	50	200
80.00	11.75	52.0	50	265
90.00	44.92	252.0	50	300
100.0	76.16	568.0	50	300
110.0	121.7	712.0	50	300
120.0	55.47	680.0	50	300
130.0	50.12	528.0	50	330
140.0	43.11	420.0	50	365
150.0	39.64	444.0	50	365
160.0	22.48	108.0	50	430
167.0	10.84	92.0	50	430

Length = 2 cells				
Frequency [MHz]	**Tracking Jitter**		**Apx. Duty Cycle [%]**	**Power [mW]**
	rms [ps]	**pkpk [ps]**		
1.00	3 353	11 100	92	130
2.00	3 200	9 800	86	130
5.00	925.0	4 100	78	130
10.00	768.2	3 160	75	130
20.00	211.3	1 000	60	165
30.00	491.6	2 120	50	200
40.00	206.0	980.0	50	165
50.00	520.2	1 740	50	165
60.00	117.5	520.0	50	180
70.00	104.7	492.0	50	200
80.00	33.93	128.0	50	230
90.00	117.6	688.0	50	300
100.0	423.0	1 600	50	265
110.0	124.2	580.0	50	265
120.0	83.44	408.0	50	300
130.0	56.43	296.0	50	330
140.0	51.16	344.0	50	330
160.0	11.21	64.00	50	400
170.0	19.41	142.0	50	415
180.0	50.05	556.0	50	430
190.0	48.17	692.0	50	430
200.0	37.63	728.0	50	460
210.0	45.13	708.0	50	460
213.0	47.85	568.0	50	495

This table shows the jitter measurements for the first iteration of the Programmable Clock Manager. The length is set to 8 cells and use for comparison against the second iteration.

Length = 8 cells (first PCM)				
Frequency [MHz]	**Tracking Jitter**		**Apx. Duty Cycle [%]**	**Power [mW]**
	rms [ps]	**pkpk [ps]**		
0.100	15.27	88.00	99	130
0.250	14.44	84.00	99	130
0.500	14.96	88.00	99	130
0.750	15.28	90.00	99	130
1.00	57.44	320.0	99	130
2.00	15.32	98.00	99	130
5.00	51.99	500.0	90	130
10.00	117.5	708.0	80	165
20.00	---------- Unstable ----------			
30.00	---------- Unstable ----------			
40.00	16.50	100.0	60	200
50.00	100.2	440.0	50	165
60.00	12.80	84.00	50	130
70.00	9.466	58.00	50	130
80.00	30.74	142.0	50	165
90.00	14.92	120.0	50	230
93.0	16.92	108.0	50	230

Appendix D - Clock Divider Simulations

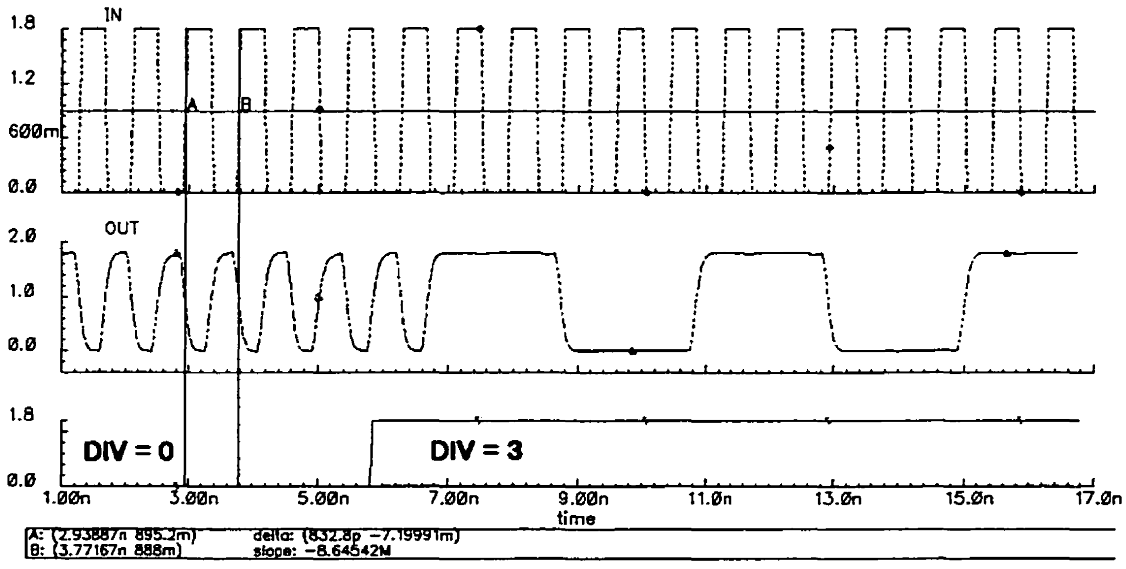


Figure D.1: Simulation of Clock Divider Operation, Manufactured in the 0.18μm 1.8V CMOS TSMC Process, Input Clock at a Frequency of 1.2GHz, The Output is Divided by Zero then Divided by Three.

Appendix E - RSPLL Simulations

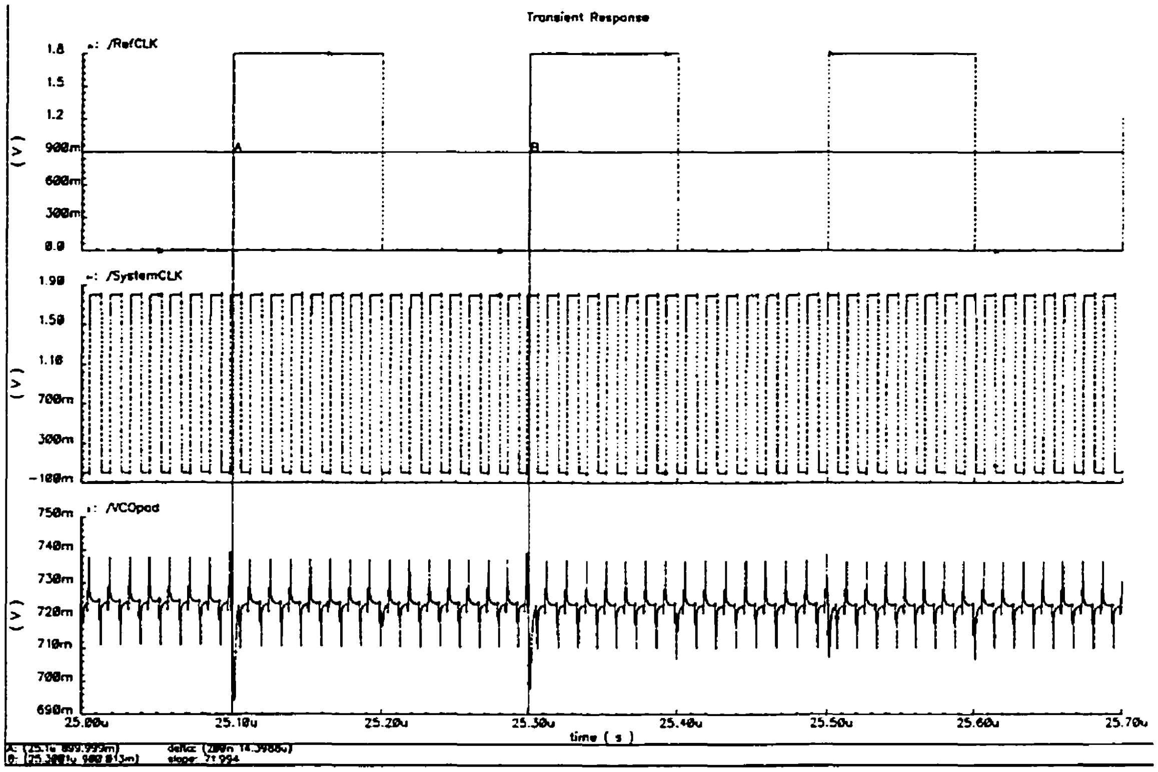
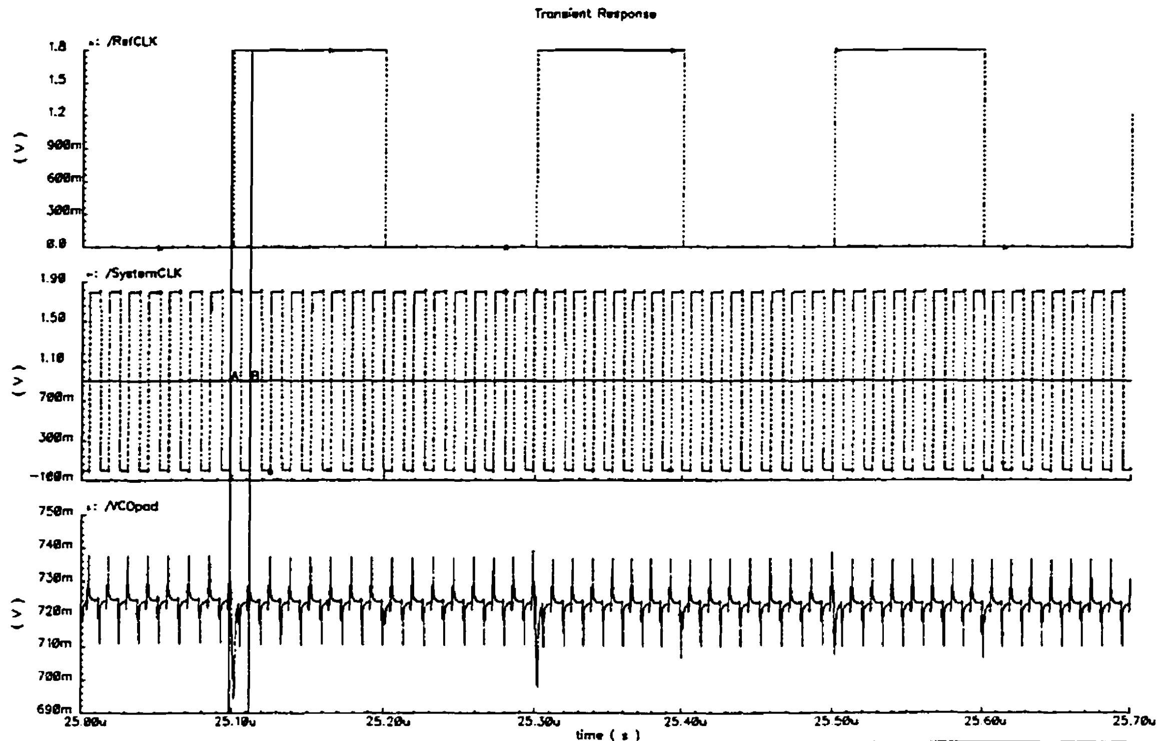


Figure E.1: Magnified View of ReferenceCLK, SystemCLK and the Control Voltage. ReferenceCLK is Measured to be 200 ns, Corresponding to 5 MHz.



```
A: (25.0984u 903.896m)    delta: (13.3467n 19.196u)
B: (25.1118u 903.916m)    slope: 1.43827K
```

Figure E.2: Magnified View of ReferenceCLK, SystemCLK and the Control Voltage. SystemCLK is Measured to be 13.3467 ns, Corresponding to ~75 MHz.

Appendix F - PCB Layout Schematics

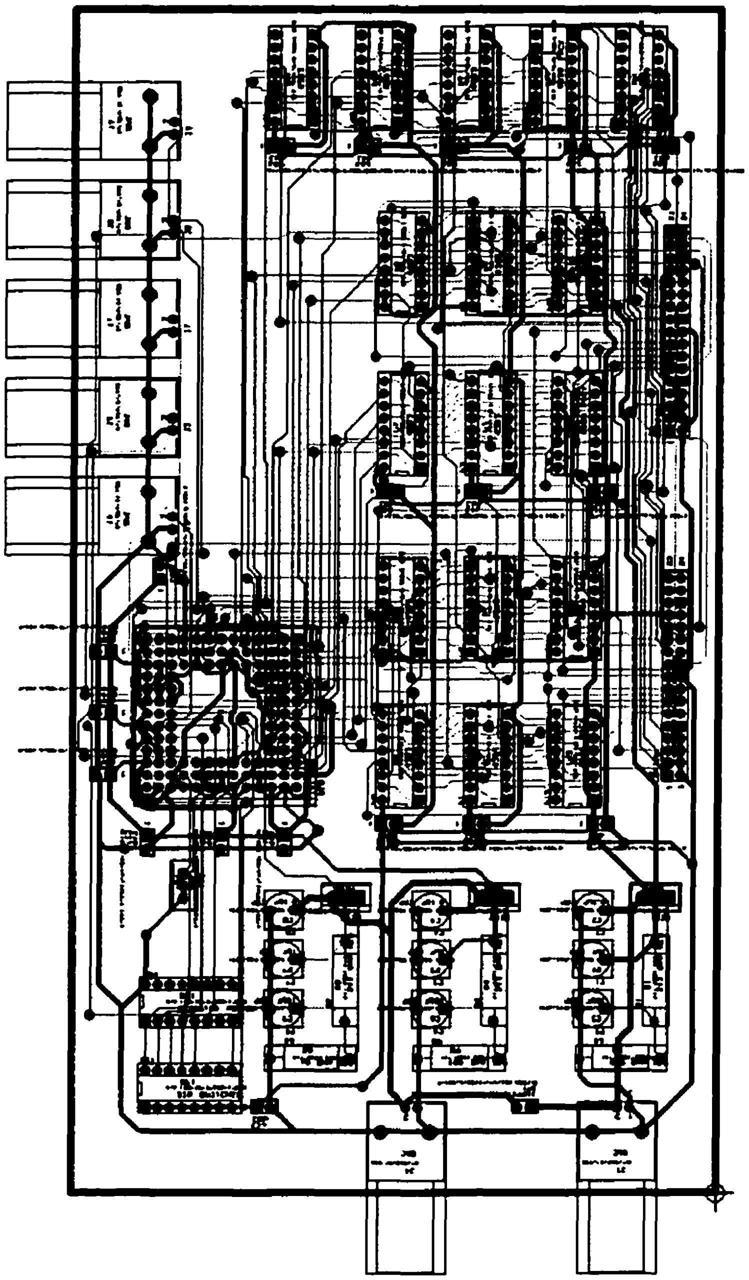


Figure F.1: Printed Circuit Board Layout Schematic for MCSoC

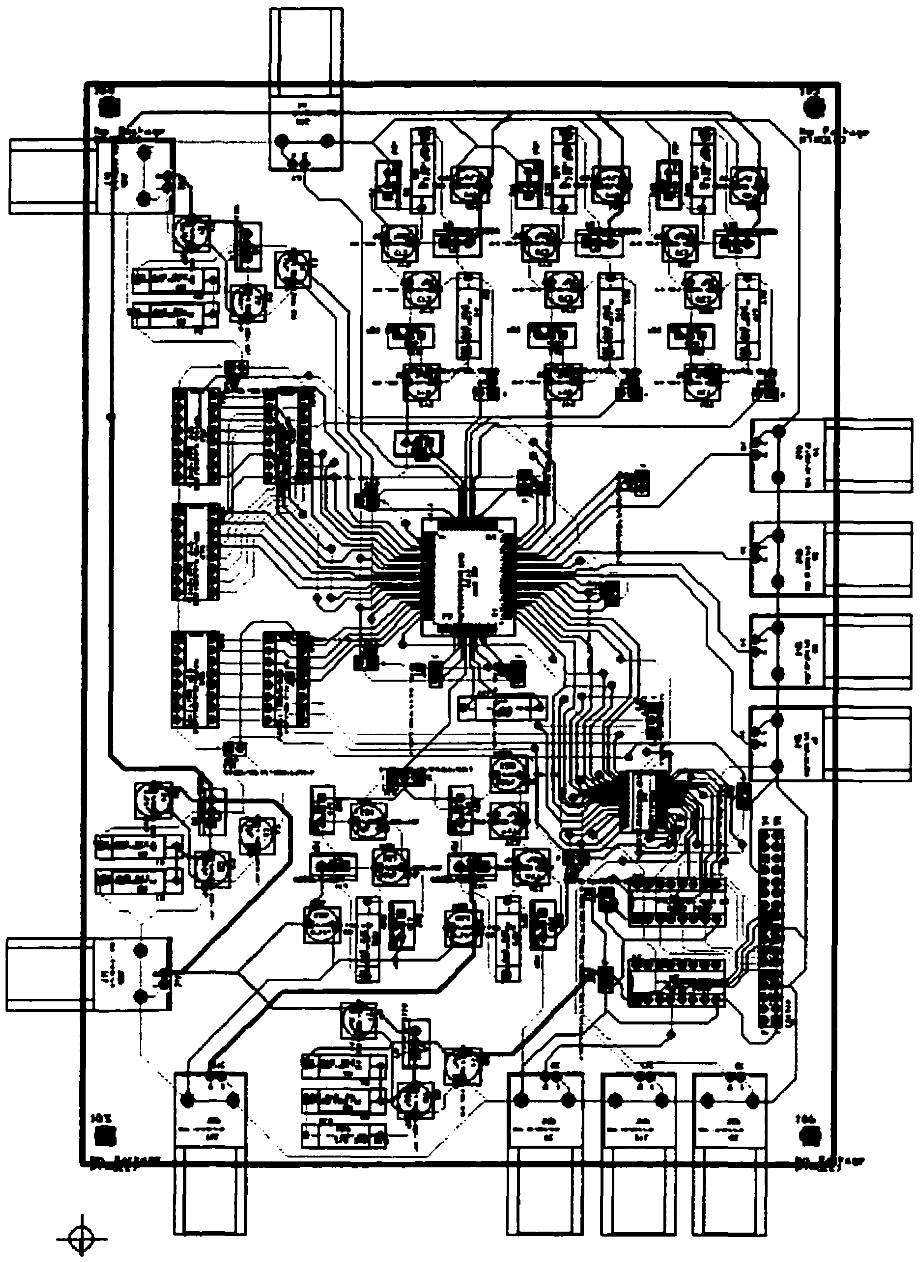


Figure F.2: Printed Circuit Board Layout Schematic for RSPLL

References

[1] I. Brynjolfson and Z. Zilic, "Dynamic clock management for low power applications in FPGAs," *Proceedings of IEEE Custom Integrated Circuits Conference*, May 2000.

[2] I. Brynjolfson and Z. Zilic, "A new PLL design for clock management applications," *Proceedings of IEEE International Symposium of Circuits and Systems*, May 2001.

[3] I. Brynjolfson and Z. Zilic, "Dynamic Programmable Clock Divider", U.S. Patent Pending, June 2000.

[4] I. Brynjolfson and Z. Zilic, "Range Shifting Phase-Locked Loop", Report of Invention, McGill University, Feb. 2001.

[5] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design - Circuits and Systems*, Kluwer Academic Publishers, Boston, 1995.

[6] J. B. Kuo and J.-H. Lou, *Low-Voltage CMOS VLSI Circuits*, John Wiley and Sons, Inc., New York, 1999.

[7] S.S. Rofail, and K.-S. Yeo, *Low-Voltage, Low-Power Digital BiCMOS Circuits*, Prentice Hall PTR, Upper Saddle River, NJ, 2000.

[8] *High-Performance System Design - Circuits and Logic*, V.G. Oklobdzija (Ed.), IEEE Press, New York, 1999.

[9] M.S. Elrabaa, I.A. Abu-Khater and M.I. Elmasry, *Advanced Low-Power Digital Circuit Techniques*, Kluwer Academic Publishers, Boston, 1997.

[10] J.M. Rabaey and M. Pedram, *Low Power Design Methodologies*, Kluwer Academic Publishers, Boston, 1996.

[11] G.K. Yeap, *Practical Low Power Digital VLSI Design*, Kluwer Academic Publishers, Boston, 1998.

[12] S. Iman and M. Pedram, *Logic Synthesis for Low Power VLSI Designs*, Kluwer Academic Publishers, Boston, 1997.

[13] A. Raghunathan, N.K. Jha, S. Dey, *High-Level Power Analysis and Optimization*, Kluwer Academic Publishers, Boston, 1998.

[14] S. H. Gunther, F. Binns and D. M. Carmean, J.C. Hall, "Managing the impact of increasing microprocessor power consumption," *Intel Technology Journal Q1, 2001*, Intel Corporation, Santa Clara, CA, 2001.

[15] Motorola Inc., *Motorola MPC755 and MPC745 PowerPC Microprocessors*, Fact Sheet, May 22, 2000.

[16] A. Klaiber, *The Technology Behind Crusoe Processors*, Transmeta Corporation, Santa Clara, CA, January 2000.

[17] D. Dobberpuhl and R. Witek, "A 200MHz 64b dual-issue CMOS microprocessor," *Proceedings IEEE International Solid-State Circuits Conference*, pp. 106-107, 1992.

[18] J.C. Monteiro, "Power optimization using dynamic power management," *Proceedings of IEEE Integrated Circuits and Systems Symposium*, 1998.

[19] V. Krishna, N. Ranganathan and N. Vijaykrishnan, "Energy efficient datapath synthesis using dynamic frequency clocking and multiple voltages," *Proceedings of 12th International Conference on VLSI Design*, pp. 440-445, Jan. 1999.

[20] E.G. Friedman, "Introduction to clock distribution networks in VLSI circuits and systems," *Clock Distribution Networks in VLSI Circuits and Systems*, E.G. Friedman (Ed.), IEEE Press, New Jersey, pp. 1-36, 1995.

[21] N. Ranganathan, N. Vijaykrishnan and N. Bhavanishankar, "A linear array processor with dynamic frequency clock for image processing applications," *IEEE Transactions of Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 435-445, Aug. 1998.

[22] S. Hauck, "Asynchronous Design Methodologies: An Overview," *Proceedings of the IEEE*, vol. 83, no. 1, pp. 69-93, Jan. 1995.

[23] M. Olivieri, A. Trifiletti and A. De Gloria, "A low-power microcontroller with on-chip self-tuning digital clock-generator for variable-load applications," *IEEE International Conference on Computer Design*, pp. 233-240, Oct. 1999.

[24] T. Pering, T Burd and R. Broderson, "The simulation and evaluation of dynamic voltage scaling algorithms," *Proceedings of International Symposium on Low-Power Electronics and Devices*, Monterey, pp. 76-81, Aug. 1998.

[25] K. Govil, E. Chan and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU," *Proceedings of 1st ACM International Conference on Mobile Computing and Networking*, 1995.

[26] Roland E. Best, *Phase-Locked Loops - Design, Simulations & Applications*, Mcgraw-Hill, New York, 1997.

[27] E. Kusse and J. Rabaey, "Low-energy embedded FPGA structures," *1998 International Symposium of Low Power Electronics and Design*, pp. 155-160, Aug. 1996.

[28] V. Betz and J. Rose, "Circuit design, transistor sizing and wire layout of FPGA interconnect," *IEEE 1999 Custom Integrated Circuits Conference*, pp. 171-174, May 1999.

[29] V. George, H. Zhang and J. Rabaey, "The design of a low energy FPGA," *1999 International Symposium on Low Power Electronics and Design. Proceedings*, pp. 188-193, Aug. 1999.

[30] J. Cong, L. Hei, C. Koh and Z. Pan, "Global interconnect sizing and spacing with consideration of coupling capacitance," *IEEE/ACM International Conference on Computer Aided Design*, pp. 628-633, Nov. 1997.

[31] E. Kusse and J. Rabaey, "Low-energy embedded FPGA structures," *Proceedings of 1998 International Symposium on Low Power Electronics and Design*, pp. 155 -160, 1998.

[32] Xilinx Inc., *Using the Virtex Delay-Locked Loop (XAPP132 Version 1.31)*, Advanced Application Note, October 21, 1998.

[33] Altera Corporation, *Using the CLockLock & ClockBoost Features in APEX Devices*, Application Note 115 (ver 1.0), May 1999.

[34] Lucent Technologies, *ORCA Series 3C and 3T FPGAs*, Data Sheet, June 1999.

[35] L.R. Albu, B.K. Britton, W.-B. Leung, R.G. Stuby, Jr., J.A. Thompson and Z. Zilic, "Programmable clock manager for a programmable logic device that can generate at least two different output clocks," U.S. Patent 6 028 463, Feb. 2000.

[36] L.R. Albu, B.K. Britton, W.-B. Leung, R.G. Stuby, Jr., J.A. Thompson and Z. Zilic, "Programmable clock manager for a programmable logic device that can implement delay-locked loop functions," U.S. Patent 6 043 677, March 2000.

[37] W.J. Dally and J.W. Poulton, *Digital Systems Engineering*, Cambridge University Press, Cambridge, 1998.

[38] P. Larsson, "A 2-1600MHz CMOS clock recovery PLL with low-Vdd capability," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 12, pp. 1951-1960, Dec. 1999.

[39] J. G. Maneatis, "Low-jitter process-independent DLL and PLL based on self-biased techniques," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, November 1996.

[40] A. Hajimiri and T. H. Lee, "A general theory of phase noise in electrical oscillators," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 179-195, Feb. 1998.

[41] K. Iravani, F. Saleh, D. Lee, P. Fung, P. Ta, and G. Miller, "Clock and data recovery for 1.25 Gb/s Ethernet transceiver in 0.35 mm CMOS," *Proceedings of the Custom Integrated Circuits Conference*, pp. 261 - 264, 1998.

[42] F.M. Gardner, "Charge-pump phase-lock loops," *IEEE Transactions on Communication*, vol VOM-28, pp. 1949-1858, November 1980.

[43] J. A. McNeill, "Jitter in ring oscillators," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, June 1997.

[44] A. Hajimiri, S. Limotyrakis and T. H Lee, "Jitter and phase noise in ring oscillators," *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 790-804, June 1999.

[45] F. M. Gardner, *Phaselock Techniques*, 2nd ed., Wiley, New York, 1979.

[46] A. Kral, F. Behbahani and A.A. Abidi, "RF-CMOS oscillators with switched tuning," *IEEE Custom Integrated Circuits Conference*, 26.1.1, May 1998.

[47] B. Razavi, "A study of phase noise in CMOS oscillators," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 3, pp.331-343, March 1996.

[48] T. Yasuda, "High-speed wide-locking range VCO with frequency calibration," IEEE *International Symposium on Circuits and Systems*, III-45, May 2000.

[49] B. C. Kuo, *Automatic Control Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[50] I. Sutherland, B. Sproull and D. Harris, *Logical Effort*, Morgan Kaufmann, San Francisco, 1999.

[51] O. E. Akcasu, "High capacitance structures in a semiconductor device," U.S. Patent 5 208 725, May 1993.

[52] Microchip Technologies Inc., *PIC12C5XX 8-Pin, 8 Bit CMOS Microcontrollers*, Data Sheet DS40139E, 1999.

[53] H.B. Bakoglu, J.T. Walker and J.D. Meindl, "A symmetric clock-distribution tree and optimized high-speed interconnections for reduced clock skew in ULSI and WSI circuits," *Proceedings of IEEE International Conference on Computer Design*, pp. 118-122, Oct. 1986.

[54] E.G. Friedman, "Clock distribution design in VLSI circuits - an overview," *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 475-1480, May 1993.

[55] X. Aragones, J.L. Gonzalez and A. Rubio, *Analysis and Solutions for Switching Noise Coupling in Mixed-Signal ICs*, Kluwer Academic Publishers, Boston, 1999.

[56] N.K. Verghese, T.J. Schmerbeck and D.J. Allstot, *Simulation Techniques and Solutions for Mixed-Signal Coupling in Integrated Circuits*, Kluwer Academic Publishers, Boston, 1995.

[57] H.-M. Chen, M.-H. Wu, B.C. Liau, L. Chang and C.-F. Wu, "The study of substrate noise and noise-rejection-efficiency of guard-ring in monolithic integrated circuits," *IEEE International Symposium on Electromagnetic Capability*, vol. 1, pp. 123-128, 2000.