Low Power flip-flop Analysis and Design for use in Network Processors

by

Somasundaram Arunachalam,
B. Eng (Carleton University)

A Thesis Submitted to the
Faculty of Graduate Studies and Research
In Partial Fulfillment of the Requirements
For the Degree of

Master of Applied Science

In Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
Department of Electronics

Carleton University

August, 2010

Your file Votre référence
ISBN: 978-0-494-71511-6
Our file Notre référence
ISBN: 978-0-494-71511-6

Abstract

This thesis explores the topic of power consumption reduction of flip-flop circuits. In order to determine the impact of the low power flip-flops on circuits, common circuits are extracted by analyzing a VLSI System. The VLSI system used for analysis in this thesis is network processors. A measurement methodology is laid out in order to standardize power and performance measurements. The common circuits, 16-bit counter, 32-bit counter, 8-bit PRBS generator, and 8-bit data bus, are simulated and total power consumption is measured.

Power consumption for four existing flip-flops technologies, standard master-slave flip-flop, clock gated flip-flop, data transition look-ahead flip-flop, and clock-on-demand flip-flop, are measured and analyzed. Three flip-flop topologies, clock on demand with shared pulse generator, clock gating on demand, and clock gated data transition look-ahead flip-flop, are introduced that combine different power saving techniques to reduce the total active power consumption of the circuits they are implemented in. The proposed flip-flops share pulse-generators between many storage elements to reduce power consumption overhead. The clock signals are gated when the data input and the data output to the storage elements do not toggle. Simulations are run in IBM 65 nm CMOS technology. Utilizing clock gated data transition look-ahead flip-flop in a 16-bit counter reduces power consumption by 90% over a counter implemented with standard master slave flip-flop when counter is not being incremented and power consumption is reduced by 25% when the counter is incremented every cycle. Utilizing clock gating on demand flip-flop for an 8-bit data bus implementation reduces power consumption by 90% over data bus implemented with standard master-slave flip-flop when activity factor is 0%. However the data-bus implemented with master slave flip-flop has 4.5% lower power consumption than the data-bus implemented with clock gating on demand flip-flop.

Acknowledgments

I would like to thank everyone who helped me during the course of my research and offered me words of encouragement. I am deeply grateful to my supervisor Professor Maitham Shams for his guidance and patience. I would also like to take this opportunity to thank my managers at Cisco Systems, Sidney Allman, Shailesh Sutarwala, and Allan Silburt for their support.

I would like to express my gratitude to Nagui Mikhail, and Scott Bruce for their technical support. I am sincerely thankful to Prof. Pavan Gunupudi and the Dept of Electronics for being supportive and accommodating.

I especially would like to thank my parents, Arunachalam Subramaniam and Meenakshi Arunachalam, for their encouragement and patience throughout the course of my graduate studies.

Contents

List of Figures

List of Tables

List of Symbols

$P_{switching}$ – Switching Power

T – Clock Period

i(t) – Instantaneous Current

V_{dd} – Voltage at Supply Rail

α – Activity factor

C – Node Capacitance

$V_{threshold}$ – Threshold Voltage

RCcmax – Maximum resistance and capacitance

Abbreviations

ASIC	Application-specific integrated circuit
CMOS	Complementary Metal-Oxide Semiconductor
CoD	Clock on Demand
CGoD	Clock Gating on Demand
CRC	Cyclic Redundancy Check
CRS	Core Routing System (Cisco Systems Flagship product)
DTLA	Data Look-Ahead
DUT	Device Under Test
DV	Design Verification
flip-flop	flip-flop
GALS	Globally Asynchronous Locally Synchronous
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMIX	Internet Mix
IP	Internet Protocol
MAC	Media Access Control
MII	Media Independent Interface
MSC	Modular Service Card
NP	Network Processor
OSI	Open Systems Interconnection
PCS	Physical Coding Sublayer
PLL	Phase-locked loop
PRBS	Pseudo-Random Bit Sequence
RMON	Remote Network Monitoring
RTL	Register Transfer Language
VLSI	Very Large Scale Integration

Chapter 1 Introduction

Ever expanding internet market penetration, mobile usage, and demand for high definition video, will cause Global IP traffic to quadruple from 2009 to 2014. Annual Global IP traffic is expected to exceed 767 Exabytes (10^{18}B) per year in 2014 [1]. High speed routers capable of routing terabytes of data per second are needed to handle the IP traffic growth. Network Processors, the backbone of the high speed routers, are expected to meet the growing traffic demands in addition to supporting new features that allow better network management and higher network robustness. In order to offer these features and meet performance requirements, it is necessary to move to higher levels of integration and higher clock frequencies. This will cause power consumption to increase at a rate faster than the power reductions offered by moving to an advanced CMOS fabrication node. The power consumption of a last generation Modular Service Card, CRS-1 MSC by Cisco Systems, is 375W [2], while the power consumption of the current generation Modular Service Card, CRS-3 MSC, is 446W [3]. Even though the process technology for the current generation of the MSC cards is more advanced by two CMOS fabrication process nodes, the power consumption has increased due to increased integration, advanced feature sets, and higher performance. A solution, to reduce active power consumption without sacrificing system performance and reliability, is needed.

1.1 *Motivation for this Work*

Increasing power consumption in network processors used in an always-on environment such as core routers, demand the development of low power techniques. At the same time, these low power techniques cannot sacrifice the reliability and robustness that is demanded for operating in the mission critical environment. The expected uptime of a network core router is 99.999%. This translates to an average of 3 minutes of downtime in a year [4]. In addition, the solution should be transparent to ASIC design flow and Design Verification (DV) flow to minimize the impact on the design schedule, product deployment schedule, and product development costs. This is achieved by reducing power consumption on a commonly used circuit, which does not change the basic functionality of the circuit, but instead achieves the

power reduction necessary to meet the goals. Through this method, the changes are transparent to the RTL designers and DV engineers, and have minimal impact on schedule and cost.

Flip-flops consume 30%-60% of the total power consumption in VLSI system [5]. This includes power dissipated due to active current and leakage current. Hence, targeting flip-flops for power reduction while maintaining the desired functionality will easily achieve power savings in network processor.

1.2 *Thesis Outline*

Chapter 2 of this thesis will discuss common power reduction techniques as background information. It will go on to provide a brief overview of a Network Processor and its components in order to extract common circuits. This process allows us to understand how the flip-flops will be used in each of the circuits, and which type of flop architecture to use to reduce power consumption. Chapter 3 will provide the testing methodology and the testing environments for the flip-flops under analysis. Chapters 4 will provide the results and analysis of existing flip-flops. Three flip-flop circuits will be proposed, in Chapter 5, which combine multiple power reduction techniques to lower power consumption. Chapter 6 will summarize the thesis.

1.3 *Thesis Goals*

The thesis attempts to introduce energy-efficient flip-flop designs for use in network processors. This goal is achieved through identifying and fulfilling the following objectives.

1. Developing a test environment for flip-flop circuits as if they are operating in a network processor
2. Measure power and performance of various low-power flip-flops.
3. Suggesting new flip-flop designs to better suit the network processor environment.
4. Providing recommendations on the best flip-flops to use within the circuits tested, based on analysis of the simulation results.

All schematic and layout simulations in this work are based on an IBM CMOS 65nm design kit. The goal of the thesis is not to exhaustively characterize the flip-flops for standard cell construction, but rather to establish design methods which reduced power consumption to indicate if they can be considered low power design methods.

Chapter 2 **Power Consumption**

Power consumption mechanisms have to be well understood in order to tackle power reduction. Power consumption in ICs can be separated into two components, static power dissipation and dynamic power dissipation.

2.1 *Power Consumption Mechanism*

The general advantage of the CMOS technology over other IC technologies is that, ideally no current flows between the drain and source of transistors which are "off". However, this is not a completely accurate description. Static power is dissipated in CMOS circuits due mainly to leakage current. Leakage current is caused by conduction through "off" transistors as sub threshold voltage current, gate tunneling current, and leakage through reverse-biased diodes. The dynamic power consumption in CMOS circuits is due to two main mechanisms. The primary mechanism is the charging and discharging of various internal parasitic capacitances, and load capacitances. The secondary mechanism is the current spike that occurs when both the pull-up and pull-down networks are partially on. This is commonly referred to as the short-circuit current.

2.1.1 Leakage Power

Leakage current has been growing substantially as circuits are built in advanced process technologies that continually shrink the device geometries. Static power consumption, due to leakage, now accounts for 20 to 25 percent of the total power consumption of designs being fabricated in the most advanced process technologies [6] [21]. There are different mechanisms that cause leakage current: gate-substrate tunneling, subthreshold conduction through "off" transistors, reverse biased diodes, etc. The two biggest contributors to leakage power consumptions are gate-to-substrate tunneling and subthreshold leakage [6]. Gate to substrate leakage occurs when electrons tunnel through the gate oxide to the substrate. This current increases exponentially as the gate oxide thickness is reduced. Gate oxide thickness is reduced to increase the electric field effect through the gates. Power dissipation due to gate leakage,

gate-substrate tunneling, forms the majority of the static power consumption when gate oxide thickness is less than 20 Angstroms [7]. Subthreshold conduction occurs when the transistor is "off", when $V_{GS} < V_{threshold}$, but small amounts of current flow through the transistor due to the weak inversion layer. Subthreshold current rises exponentially with temperature and gate-to-source voltage [6]. High threshold voltage transistors will be exclusively used when implementing the flip-flops to minimize leakage power.

2.1.2 Switching Power

Switching power is the power dissipated due to charging and discharging the internal capacitances and load capacitances. The average power consumed for charging and discharging the capacitances over time T, is given by

$$P_{switching} = \frac{1}{T}\int_0^T i(t)V_{dd}\,dt \qquad \text{(Eqn 2.1)}$$

Since T refers to the clock period, 1/T can be expressed in terms of frequency. However, this needs another factor since the node does not switch every cycle. We can multiply frequency by activity factor, α, to accurately express switching power dissipation. Activity factor is the probability of a gate transitioning in a clock cycle. Maximum activity factor of a static gate on the data path is 0.5 (excluding glitches) since data can only transition once in a clock cycle. Hence, 1/T can be expressed as αf. The terms being integrated can be written in terms of total charge being delivered.

$$\int_0^T i(t)dt = CV_{dd} \qquad \text{(Eqn 2.2)}$$

The switching power can thus be rewritten as the following:

$$P_{switching} = \alpha f C V_{dd}^2 \quad \text{(Eqn 2.3)}$$

2.1.3 Short-Circuit Power

Short-circuit power consumption can consume up to 20% of the total power dissipation of CMOS VLSI systems [8]. Short-circuit power dissipation occurs when both the pull-up and pull-down network are partially ON. The short-circuit power consumption is closely related to the input slew rate and output load capacitance.

2.2 *Power Reduction Techniques*

There are many power reduction techniques that have been well researched and documented in the academic world. However, not all the techniques such as voltage reduction, asynchronous design, etc. are easily adoptable by the industry due to a disconnect between the two worlds.

2.2.1 Voltage Reduction

Analyzing the switching power equation, Eqn 2.3, it can be deduced that the act of reducing the supply voltage will have a quadratic effect in reducing average switching power consumption. Unfortunately reducing voltage has the undesired effect of reducing performance if feature size is constant. Design libraries provided by the vendors to fab-less companies contain analog and memory components which require a certain voltage level to operate across all process corners reliably. This is one of the deciding factors in setting the core voltage supply. Another important factor in setting voltage level is design margin. Network processors used in core routers have a long product lifecycle in the field and high reliability requirements. The supply voltage cannot be reduced below a certain level, as transistor performance degrades over time and will affect the reliability of the network processors [9]. For these reasons, supply voltage reduction is limited. Setting up voltage islands, shown in Figure 2-1, to reduce supply voltages of sub-modules, whose performance can be met at a lower voltage, introduces design complexities and verification effort which impact product design schedules.

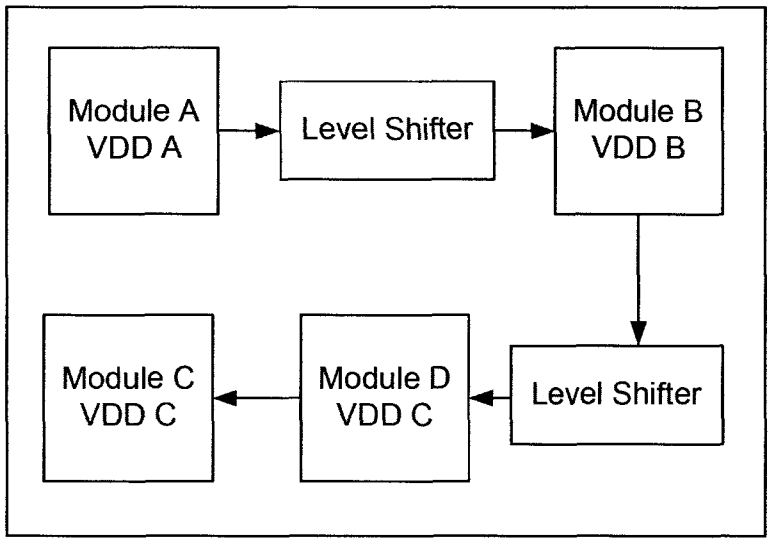

Figure 2-1: Example of usage of Voltage Islands

2.2.2 Increasing Transistor Threshold Voltage

Increasing threshold voltage can reduce leakage current, thus reducing total power consumption, but it has the adverse affect of causing performance degradation. Using high-threshold transistors on logic paths that have large timing slacks, allow for power reduction while system performance is met. Mixing transistors with different threshold voltages allows the VLSI systems to meet the performance requirement while reducing power consumption [10]. This does not create much of an impact on the design cycle or the design verification cycle of the VLSI system. All transistors used in this thesis are of the type *high threshold voltage* from the IBM 65 nm design kit. This is done to minimize leakage current.

2.2.3 Asynchronous Design

Completely asynchronous digital circuits, also referred to as clock-less circuits [11], are not on the horizon for network processors due to the requirement for supporting IP traffic and IEEE standards, i.e. IEEE 1588, flow control (binary and class-based) etc., which have knowledge of timing built-in. Flow control logic requires the network processor to pause egress traffic for a variable amount of time based on the far-end station. Lot of components in the project is re-use modules, which reduces project schedules when moving from one project to

another [12]. Completely moving to a clock-less design will force the entire re-coding and verification of all modules. This is a non-starter for many in the industry. A compromise approach often taken is the Globally Asynchronous and Locally Synchronous (GALS) method. The GALS method is when the modules in a large design are running at different frequencies, thus conserving power and still meeting system performance requirements.

2.2.4 Review of other power reduction techniques

Reducing the data transition probability reduces power consumption in digital circuits [33]. A standard master-slave flip-flop has considerable power consumption even when there is no data activity on the data input pin. This is due to the portion of the circuit that is switching when the clock is toggling [34]. Clock-gating flip-flop alleviates the problem by gating the clock when clock enable pin is de-asserted thus reducing the power consumption. Clock-gating is an effective way to reduce power consumption by reducing the activity on the clock tree. The synthesis tool identifies the clock gating signals in RTL code through analysis of the code in order to gate the flip-flops [13]. The clock-enable signal should only change when the clock signal is low for a positive edge triggered flip-flop. Hence, clock gating techniques involve the latching of the clock-enable signal in order for proper operation of the flip-flop. This only applies to clock-gating techniques that make use of the clock-enable signal. However, the power consumption is greater than standard master-slave flip-flop for any activity factor if the clock enable pin is asserted [Section 4.7]. Clock-gating techniques used in clock on demand flip-flop and data transitional look-ahead flip-flop do not have this restriction thus reducing area overhead and power consumption. Conditional flip-flop circuits, data transitional look-ahead flip-flop and clock on demand flip-flop; reduce power consumption by activating the internal clock only when a new input data arrives [34]. Data transitional look-ahead flip-flop requires an external pulse generator thus has a power penalty [34]. Clock on demand flip-flop has a power penalty for a high data activity factor [34]. Further reduction of data activity factor of the pulse generator can reduce the power consumption of the flip-flops.

Double gating [36] applies clock gating techniques separately to master latch and slave latch. Power consumption is reduced at very low levels of activity factor at the

expense of much higher area. This differs very much from the clock on demand flip-flop technique were the flip-flop is comprised of a pulse generator and a latch. Due to integration of the dual comparators with the latch, it is not possible to share clock gating with neighboring flops thus dual gating has a area and power penalty for higher activity factor. It is also not possible to take advantage of the similar temporal activity factor of circuits. The clock input capacitance is also higher due to the fan-out of the clock signal to two gating logic. This leads to the next issue of different clock input capacitances of various flip-flop architectures. To take this into account the clock tree power consumption is taken into account in the power measurements.

Conditional capture flip-flops [37] reduce power consumption by eliminating redundant transitions of the internal nodes. This in principle is similar to conditional clocking on demand flip-flop and data transition look-ahead flip-flop. The implementation is however vastly different [37]. In the implementation, the clock input signal is passed though three inverters which would increase the power consumption over clock on demand flip-flop and data transition look-ahead flip-flop even when there is no toggling of the data input signals.

2.3 *Test-bed Creation*

A network processor contains various circuits with various activity factors. It will be used as an example to extract circuits from which various test-benches will be created. The main functionalities of the network processor are to classify incoming packets, shape traffic, correctly forward packets, and collect statistics for network management and billing. A network processor consists of modules that described in the following sections.

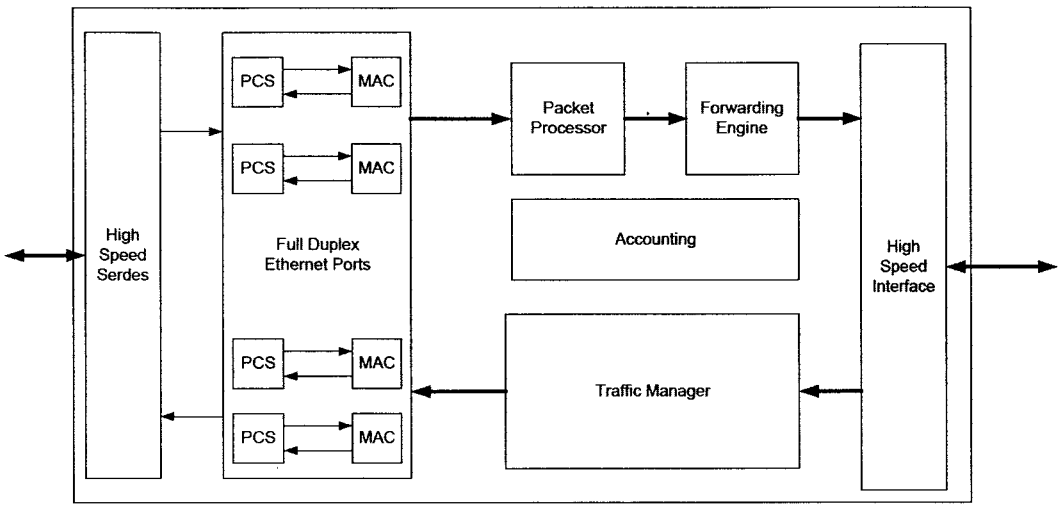

Figure 2-2: Ethernet Network Processor

2.3.1 Physical Coding Sub-Layer (PCS)

2.3.1.1 PCS Functionality

The Media Independent Interface (MII) is used to interconnect the Media Access Control (MAC), which is the layer 2 in the Open Systems Interconnection (OSI) model [14]. PCS is used to decode the data received from the layer 1 in the OSI model. In the transmit path, the PCS module is expected to scramble the data before transmitting data to the line. A side-stream scrambler in the transmitting station scrambles the incoming data [15].

2.3.1.2 PCS Data Activity Factor

Thus the data is toggling every cycle if the Ethernet port is active and the PCS clock frequency is the same as the receive clock frequency. Regardless of the packet receive and transmit rate, the data activity factor in the PCS module is expected to be high due to the scrambling and descrambling of the data. In order to simulate the typical average power consumption of flip-flops used in the PCS module, power consumption of an 8-bit Pseudo Random Bit Sequence (PRBS) generator will be measured.

2.3.2 Media Access Control (MAC)

2.3.2.1 MAC Functionality

The MAC module handles the layer 2, Data Link, functionalities of the OSI model. The complete functionalities of the MAC module are specified in the IEEE 802.3 Ethernet specification. The main functionalities of the MAC module is to be able to receive and transmit packets at the MII interface, handle flow control as specified by the IEEE802.3ae specification, handle various packet and link errors, provide Remote Network Monitor (RMON1) accounting support as specified by the Internet Ethernet Task Force (IETF). When the link is idle, no packets are being received, the MAC will receive IDLE MII codes. The Idles codes, Figure 2-3, can be identified by the data on the MII interface being 0x07 and the sideband control signal being set to 1 for every byte of IDLE code on the line. When packet data, Figure 2-4, is being sent on the line, the control signal is set to 0 and the raw packet data bytes are sent to the MAC module on the 32-bit MII interface. The MAC module can be broken down into three major sub-modules, Figure 2-5.

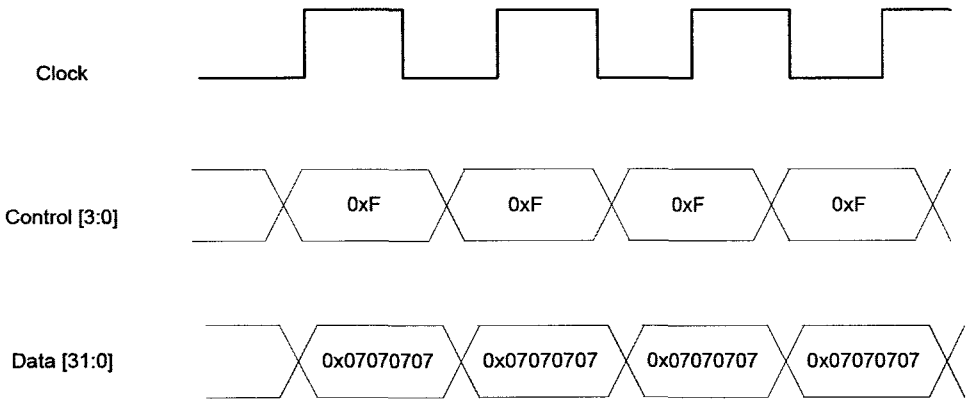

Figure 2-3: MII Idle Codes

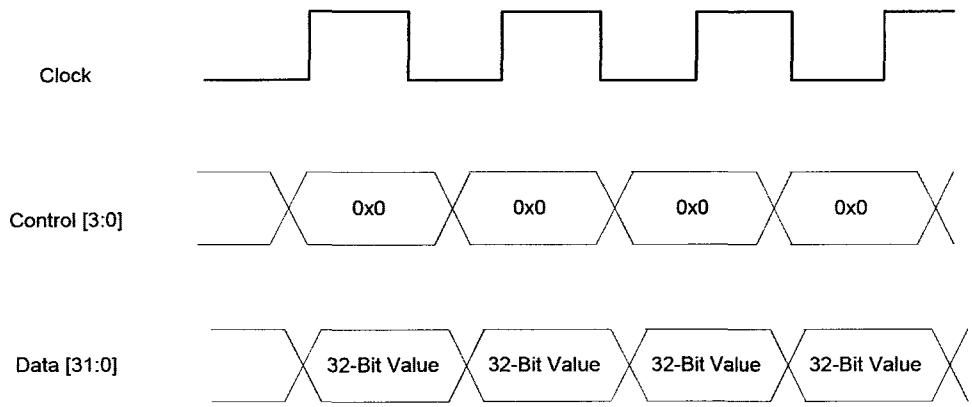

Figure 2-4: MII Data Pattern

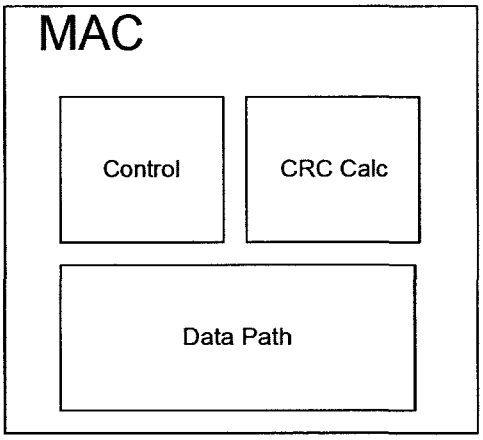

Figure 2-5: Components in MAC Module

2.3.2.2 MAC Data Activity Factor

The activity factor of the Cyclic Redundancy Check (CRC) calculator module and the data path are highly dependent on the bandwidth utilization and clock frequency. When there are no packets being sent on the line, the activity factor in these two modules is 0%. When the bandwidth utilization is high, the activity factor is dependent on the clock frequency and active data rate. A test-bed containing a data generator, with various data input activity factors will be used to simulate the power consumption of flip-flops that are present on the data path and CRC calculation module.

2.3.3 Packet Processor, Traffic Manager and Forwarding Engine Modules

The packet processor, traffic manager, and forwarding engine modules contain logic that can be classified into data path specific logic and control path specific logic. Similar to the MAC module, the data path specific logic activity factor is heavily dependent on bandwidth utilization and the control path specific logic activity factor is dependent on packet flow rate. Hence the same test-bed, data-bus test-bed, is sufficient enough to simulate the power consumption of circuits present in sub-modules of packet processor, traffic manager, and forwarding engine modules.

2.3.4 Accounting Module

The accounting module collects and stores statistics collected by all the other modules. The accounting module consists mainly of counters. The accounting module uses flip-flops to store the counter value and adders to increment the counter. A test-bed containing a 16-bit counter with a counter increment generator capable of generating different increment rates can be created to simulate the power consumption of a counter in the accounting module.

2.3.5 Test-bed Summary

There are four basic circuits extracted to be used in the test-beds. The first two circuits are counters with different widths, 16-bits and 32-bits wide. The third circuit is an 8-bit PRBS generator. The fourth and final circuit is an 8-bit data-bus, Section 3.8.

Chapter 3 **Test and Measurement Methodology**

A good methodology is required when comparing multiple characteristics of several circuits with different features and input requirements. Without a good measurement methodology, simulation results can be skewed to show some circuits with better performance under some circumstances or end up with incorrect results. In order to ensure a fair comparison of the circuits under analysis and to draw the correct conclusions, a test-bench, Figure 3-1, which provides the right stimulus and measure power consumption under the same conditions needs to be created.

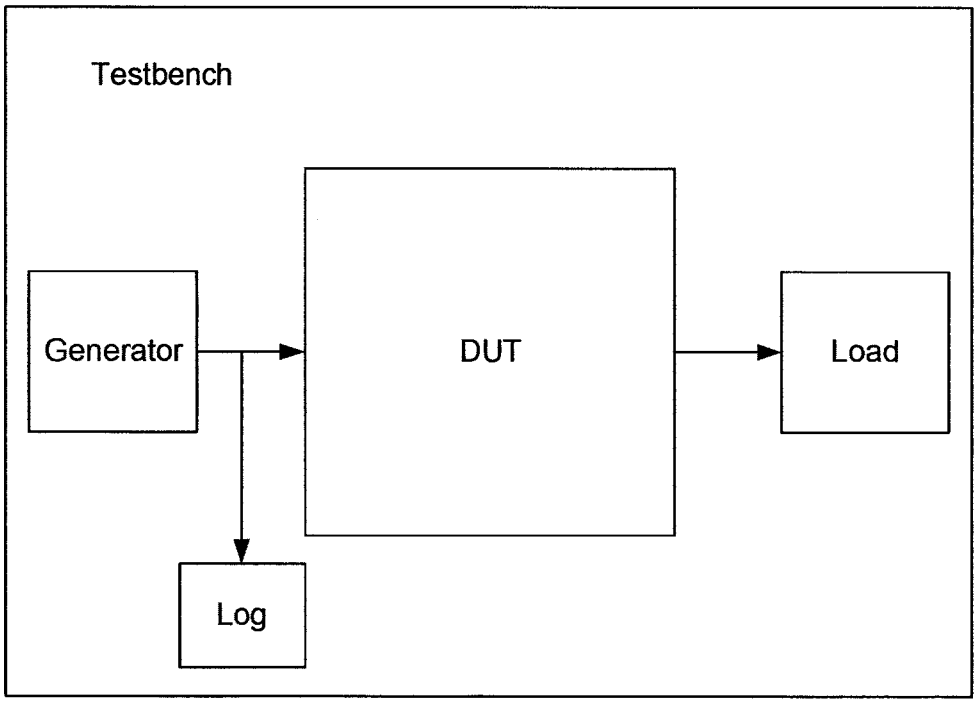

Figure 3-1: Test-bench Overview

Comparison of power measurement becomes complicated when flip-flop functionalities are different. The circuits under analysis offer different functionalities such as clock gating, data look-ahead clock gating, and explicit pulsed flip-flops. In order to draw a valid conclusion, flip-flops will need to be tested in multiple environments and their results analyzed. Even with differential analysis, there are two issues that cannot be accounted for just by running a

standalone simulation. The first of the two issues is Clock Gating since the logic needed to generate the *clock enable* signal needs to be present in the design.

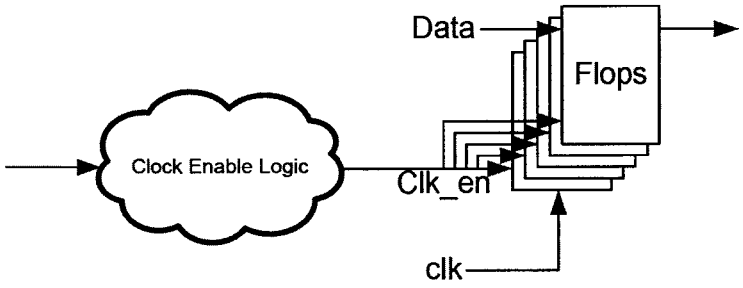

Figure 3-2: Clock Enable Logic for Clock-gated flip-flop

The "Clock Enable" signal is an input pin in the DUT, which is either high or low based on the simulation mode. The additional logic, Figure 3-2, needed to generate the *clock enable* signal needs to be factored in the analysis when deciding between different clock architectures. The clock-enable logic can be amortized over many clock-gated flip-flops thus reducing the overhead needed. This is shown in Figure 3-3. The logic used to drive the clock-enable pin can exist in the design for other functionalities and be use in addition to gate flip-flops that are idle or whose data inputs are not valid. In addition, clock enable logic is completely dependent on the specific situation and cannot be generalized for all cases. Due to these reasons, the logic needed to generate the *clock enable* signal has been excluded from the power calculations.

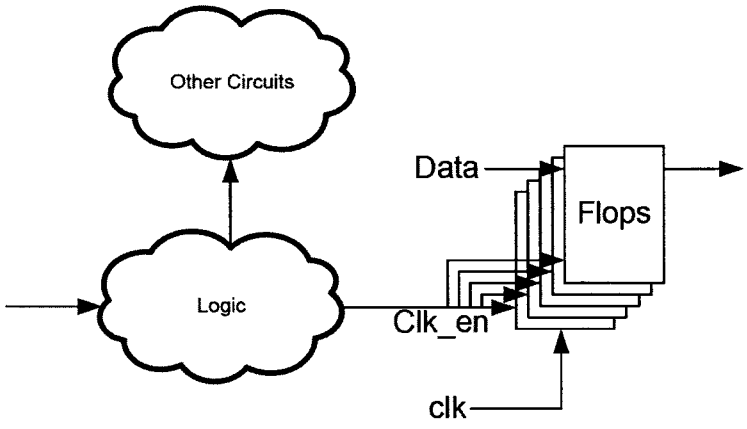

Figure 3-3: Shared Clock Enable Logic

3.1 *Creating a Test bench*

The test bench should have the following basic functionality to test a flip-flop: data generator, clock generator, Device Under Test (D.U.T), load, separate VDD rails for test bench components and D.U.T. In order to ensure that all circuits are measured in the same fashion, the D.U.T will have one or many VDD power rails. Power consumption will be measured at this power rail.

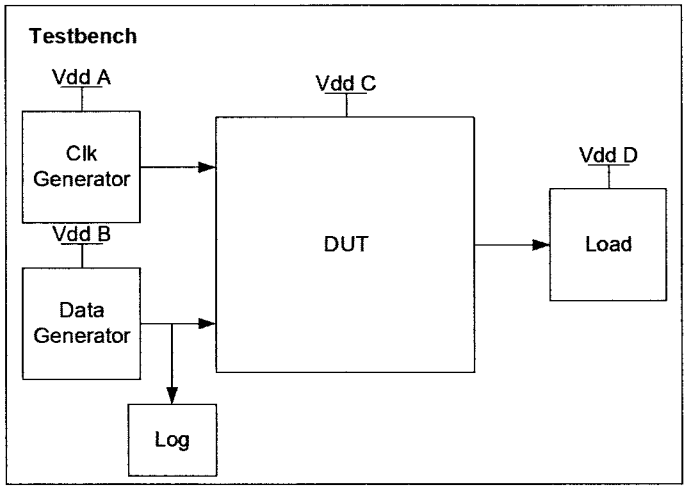

Figure 3-4: Test-bench with generators and output load

The generators output a square wave and to provide a realistic rise time and fall time for the input signals, the input signals to the DUT are passed through a chain of inverters, Figure 3-5. The output of the DUT is fed to an inverter. The power consumption of the inverters is not included in the flip-flop power consumption results.

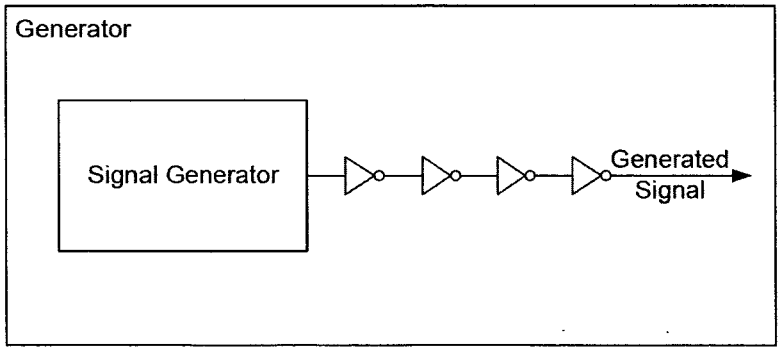

Figure 3-5: Signal generator with chain of inverters

The output of signal generator is passed through four inverters with minimum channel lengths and minimum nMOS channel width. The pMOS channel width is double the minimum size to account for the difference in electron-hole mobility. The input and output of the inverter chain are shown in Figure 3-6. The 10%-90% rise time of the signal, driven by the generator, connected to the DUT increases to 14ps.

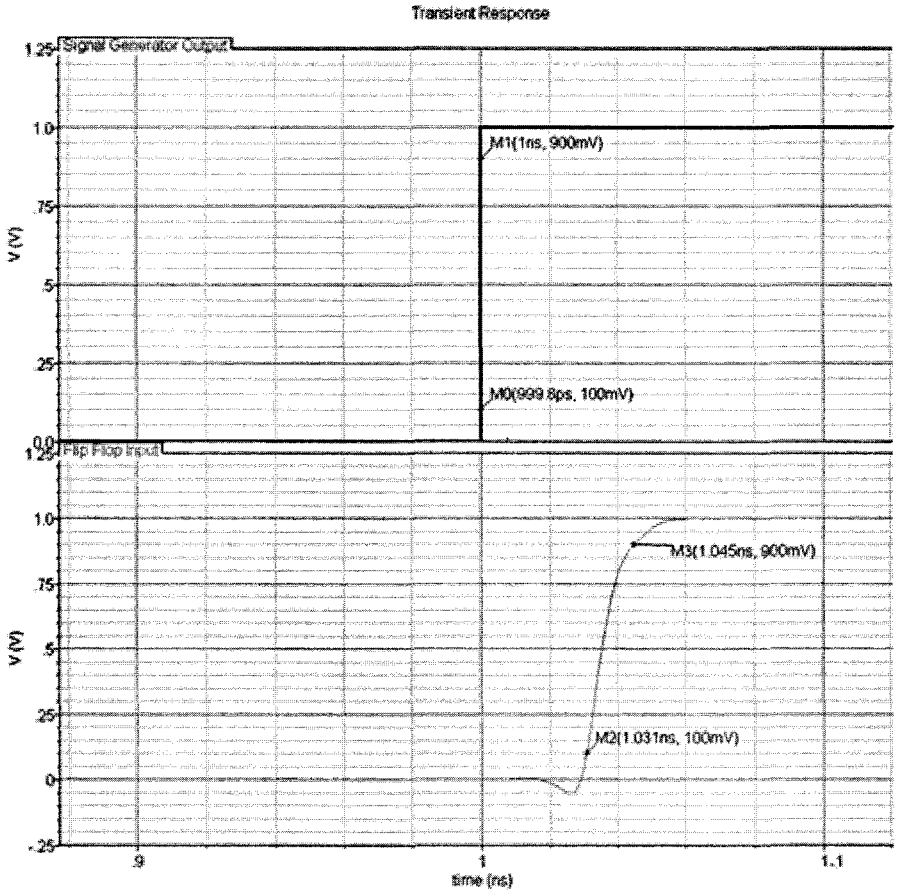

Figure 3-6: Signal Generator waveform

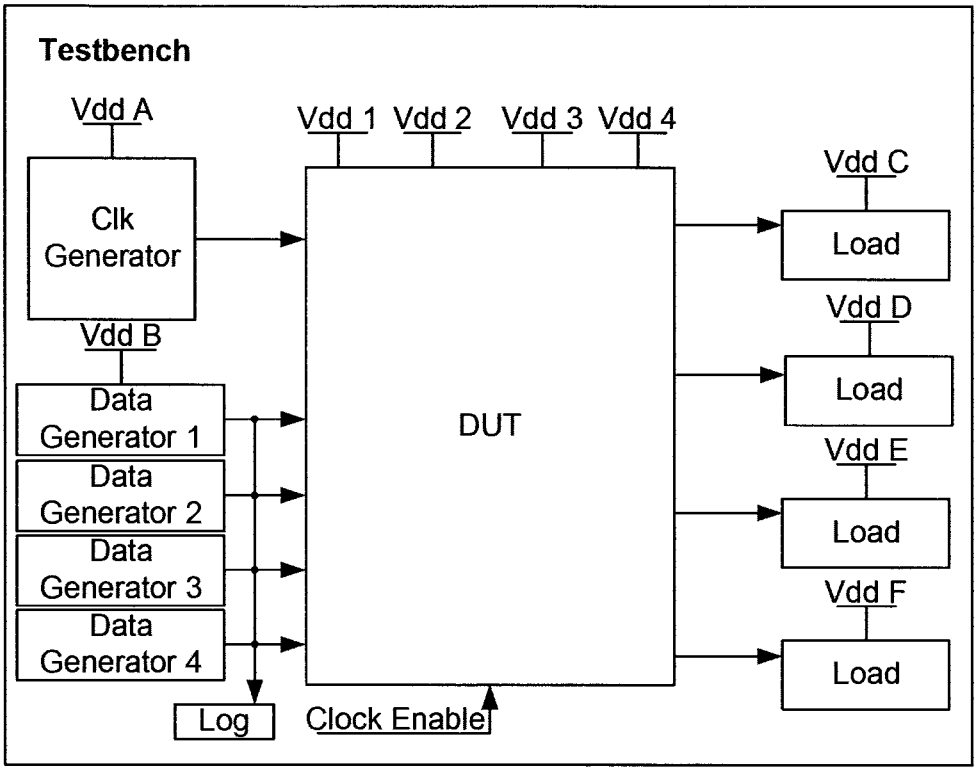

Figure 3-7: Example of a Test for a DUT which contains 4 flip-flops that support clock enable signal

The circuits being studied are not pin compatible. Some circuits require additional inputs to function correctly, example clock enable pin. The test-bench may change depending on the DUT. The test-bench, Figure 3-7, will provide additional data generators and a clock enable signal if the signals are required by the DUT to function properly.

3.2 *Defining the Device Under Test (DUT)*

The DUT incorporates the circuit that is being analyzed. In this case it will be a flip-flop or collection of flip-flops. For a flip-flop under analysis, the DUT used will be the same for power measurements and timing characterization. The figure below shows the DUT for master slave flip-flop analysis. Master slave flip-flop has 2 input ports, Data In, and Clock In. It has 1 output port, Data Out. It also has one in-out port, VDD DUT. Power consumption for the DUT, will be measured at the VDD rail, VDD DUT.

The capacitance for the node "CK" in both master slave flip-flop and clock gated flip-flop, are different. The capacitance in master slave flip-flop at node "CK" is much higher since

it is driving 3 PMOS and 3 NMOS Transistors, while the clock input capacitance for Clock-gated flip-flop is lower as it drives 1 PMOS and 1 NMOS transistor. The test-bench will contain the drivers for the DUT input signals and thus the power consumption for the drivers will not be accounted for when measuring the power consumption of the DUT. For this particular comparison, the clock-gated flip-flop power will be considerably more than the power consumption of the master slave flip-flop power consumption when the clock input pin is not gated, "Clk Enable" signal is asserted.

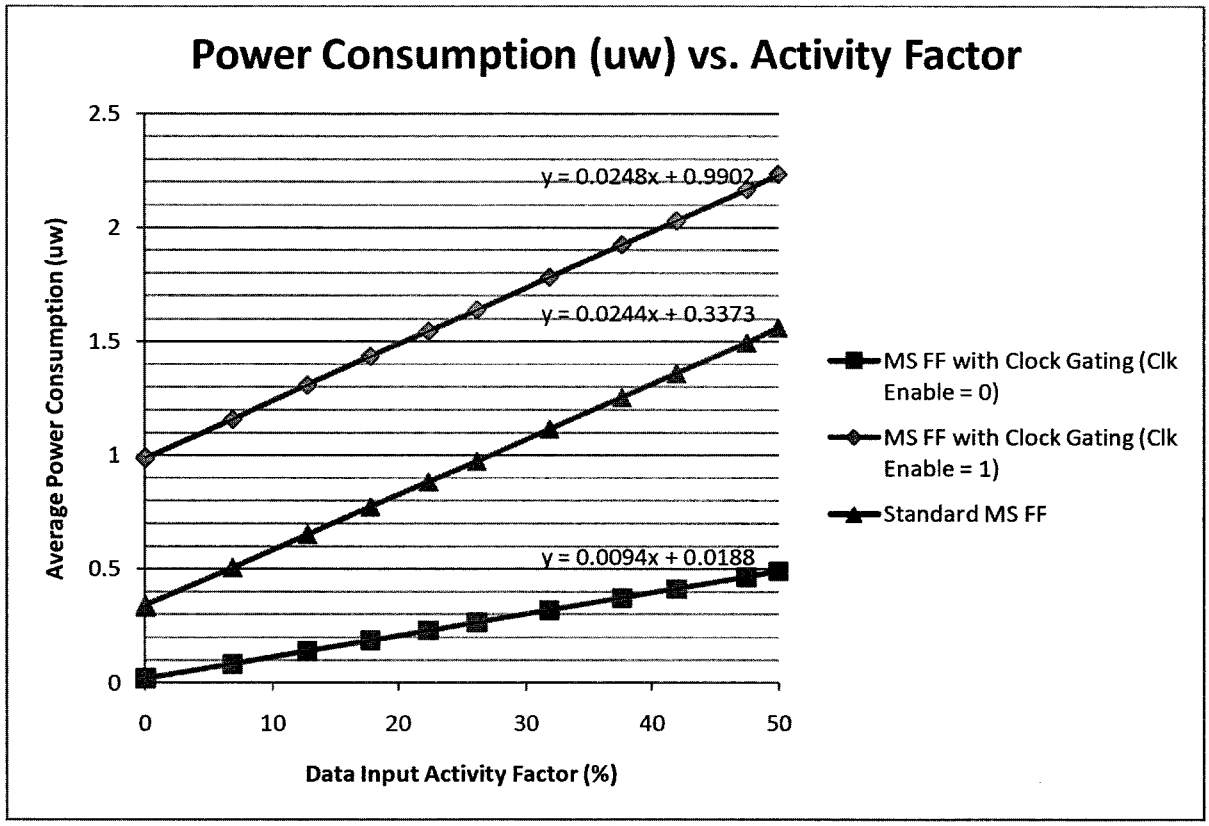

Figure 3-8: Graph of Power Consumption of MS flip-flop & Clock-gated flip-flop Vs. Activity Factor

To ensure that the power measurements for all flip-flops under analysis are under the same condition, the capacitance of the input pin has to be the same for all the DUTs. One way to achieve this is by including clock tree power consumption into the total power consumption results [23] [24].

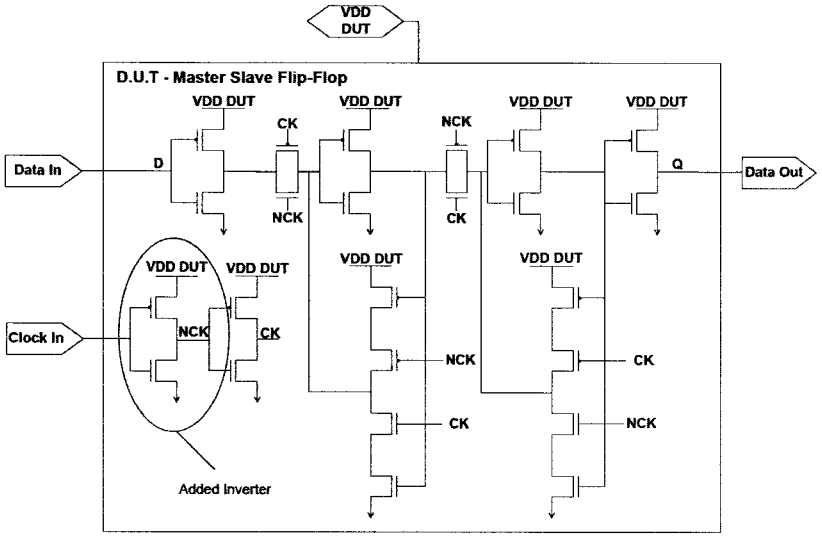

Figure 3-9: Modified Master Slave flip-flop DUT

Figure 3-10 contains the power measurement numbers for the standard master-slave flip-flop and the modified standard master-slave flip-flop which includes clock-tree power consumption.

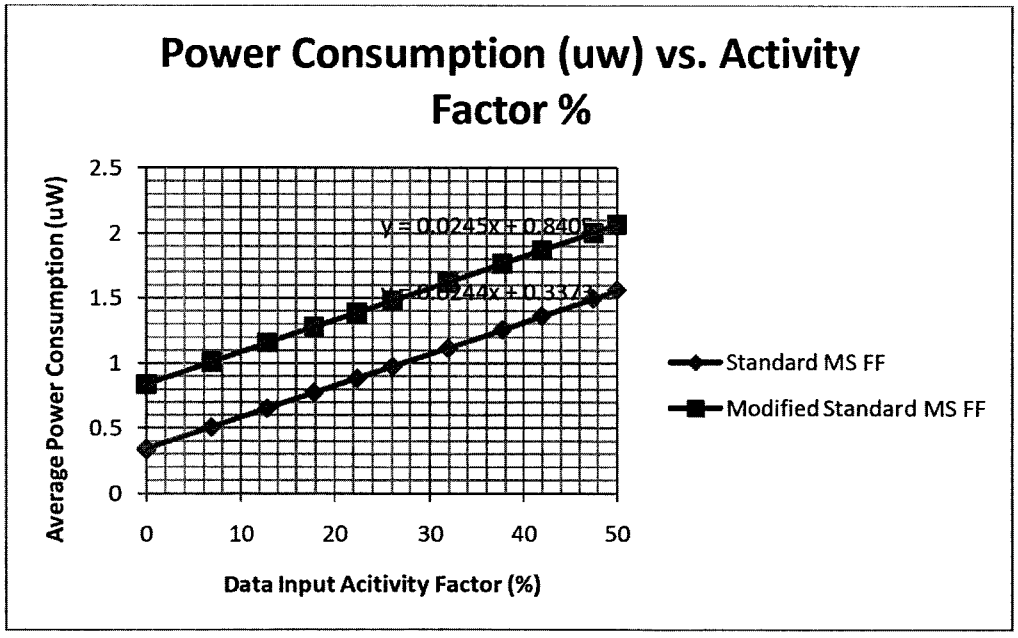

Figure 3-10: Graph of Power consumption of Modified MS flip-flop & MS flip-flop

The clock buffer was also added to the clock-gated flip-flop circuit to ensure that the input capacitance is same for all circuits under analysis. The addition of the clock tree power

consumption increases power consumption of the flip-flop by 0.2784 uw, which is obtained by calculating the power delta from the graphs in Figure 3-8 and Figure 3-10.

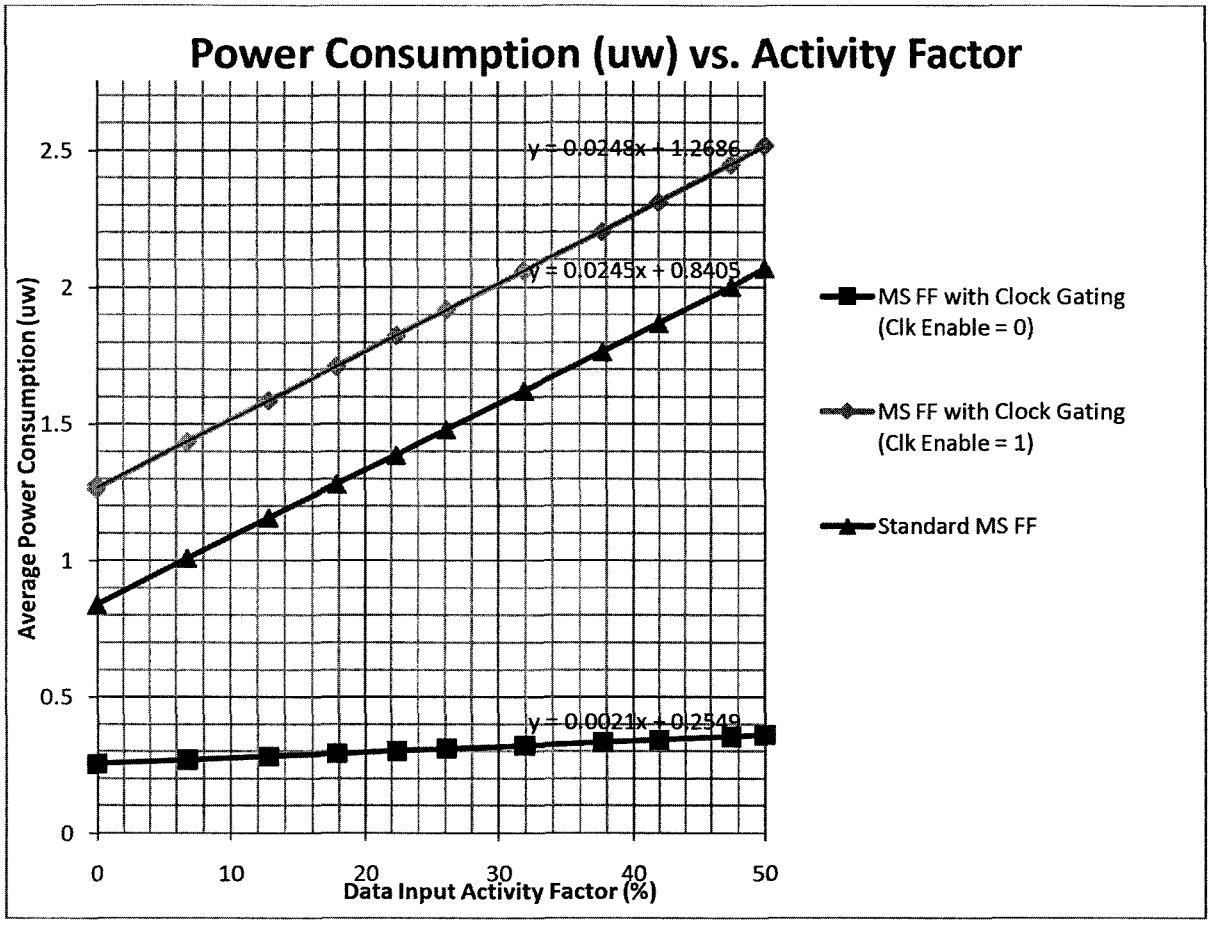

Figure 3-11: Graph of Power Consumption of MS flip-flop & Clock-gated flip-flop

The power consumption of the modified circuits is shown in Figure 3-11. The difference in power consumption of the Clock-gated flip-flop (when clock enable is asserted) and the master slave flip-flop is now 0.4281 uw which is 34.4% less than, the difference in power consumption without the addition of the clock buffer to equalize the clock input capacitance is 0.653 uw.

3.3 *Timing Characterization*

There are three timing characteristics measured for the flip-flops under study: setup time, hold time, and clock to output time. Setup time, T_{setup}, is the minimum time before the active

edge of the clock by which the data on the input pin of the flip-flop needs to settle by. If the setup time is violated, the flip-flop may enter a metastable state or capture the wrong value. The setup time in this thesis is defined as data-to-clock offset that corresponds to a 1ps increase in clock to output delay for a 1ps increase in input data transition. Measurement explanation is given in section 3.5.2.

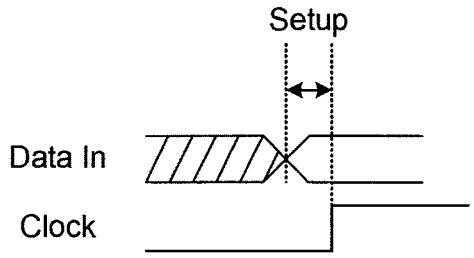

Figure 3-12: flip-flop Setup Time

Hold time, T_{hold}, is the minimum time after the active edge of the clock for which the data on the input pin of the flip-flop needs to stable for. If the hold time is violated, the flip-flop may enter a metastable state or capture a wrong value. The hold time in this thesis is defined as clock to data offset that corresponds to a 1ps increase in clock to output delay for a 1ps increase in input data transition.

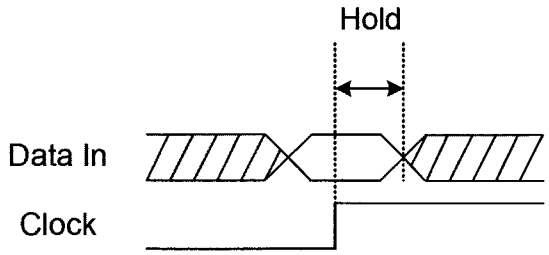

Figure 3-13: flip-flop Hold Time

Clock-to-output time, T_{co}, is the time taken for the data to propagate to the output pin after the active edge of the clock. This also referred to as the propagation delay of the flip-flop.

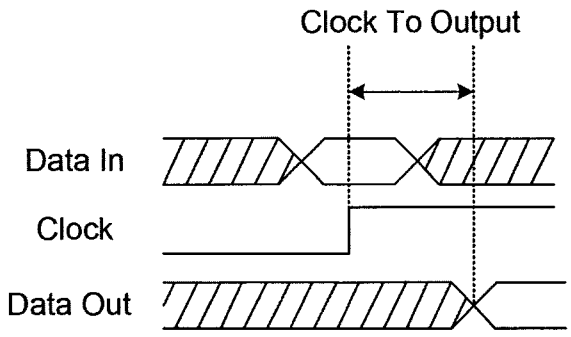

Figure 3-14: flip-flop Clock to Output Delay

Internal race immunity is the ability of the flop to tolerate clock skew between stages of flip-flops [22]. In the absence of logic gates between flip-flop stages, the propagation delay between the two stages would be the clock-to-output time. Setup time would be met but the hold time would be violated if the propagation delay is smaller than the hold time + maximum clock skew, T_{clk_skew}. The internal race immunity directly corresponds to the maximum clock skew that can be tolerated. If this number is negative, it indicates that in some cases, a buffer/chain of buffers will be needed add delay between flip-flop stages even in the absence of clock skew between flip-flop stages.

$$Internal\ Race\ Immunity = Hold\ Time - Clock\ to\ Output\ Time \quad \text{(Eqn 3.1)}$$

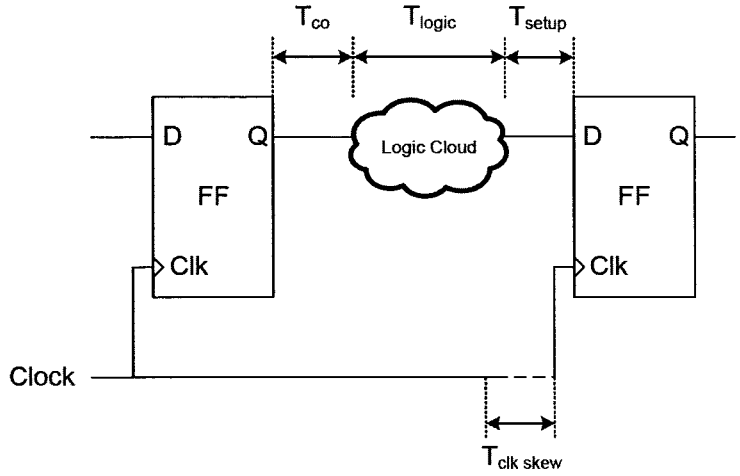

Figure 3-15: flip-flop Timing Components

The main sources of the clock skew stem from the imbalance between the clock source and the clock input pin of the flip-flop [25] [26]. The imbalance stems from the flip-flops being driven from different branches of the clock source, the differing interconnect length from a clock tree to flip-flops, capacitive coupling causing delay, clock buffers, number of fan outs on the clock tree [27].

3.4 *Providing Stimulus for DUT*

Thus all the flip-flop simulations are conducted at a single process corner case of typical-typical at room temperature. Simulation at this single corner is sufficient for analysis of the trend of power consumption of flip-flops when used in the DUT implementations. This does not compromise the goals of the thesis stated in Section 1.3. The goal of the thesis does not include the characterization of the flip-flop as a standard cell to be used in a VLSI System.

Table 3-1: Simulated Process Corners

Process Corners	Simulation
Typical-Typical	Done
Fast-Fast	Omitted
Slow-Slow	Omitted
Fast-Slow	Omitted
Slow-Fast	Omitted

3.4.1 Clock Generator

To achieve this goal, the required stimulus for power measurement would be a clock generator. The clock generator, used in all simulations for this thesis, is running at a frequency of 400 MHz and a duty cycle of 50%.

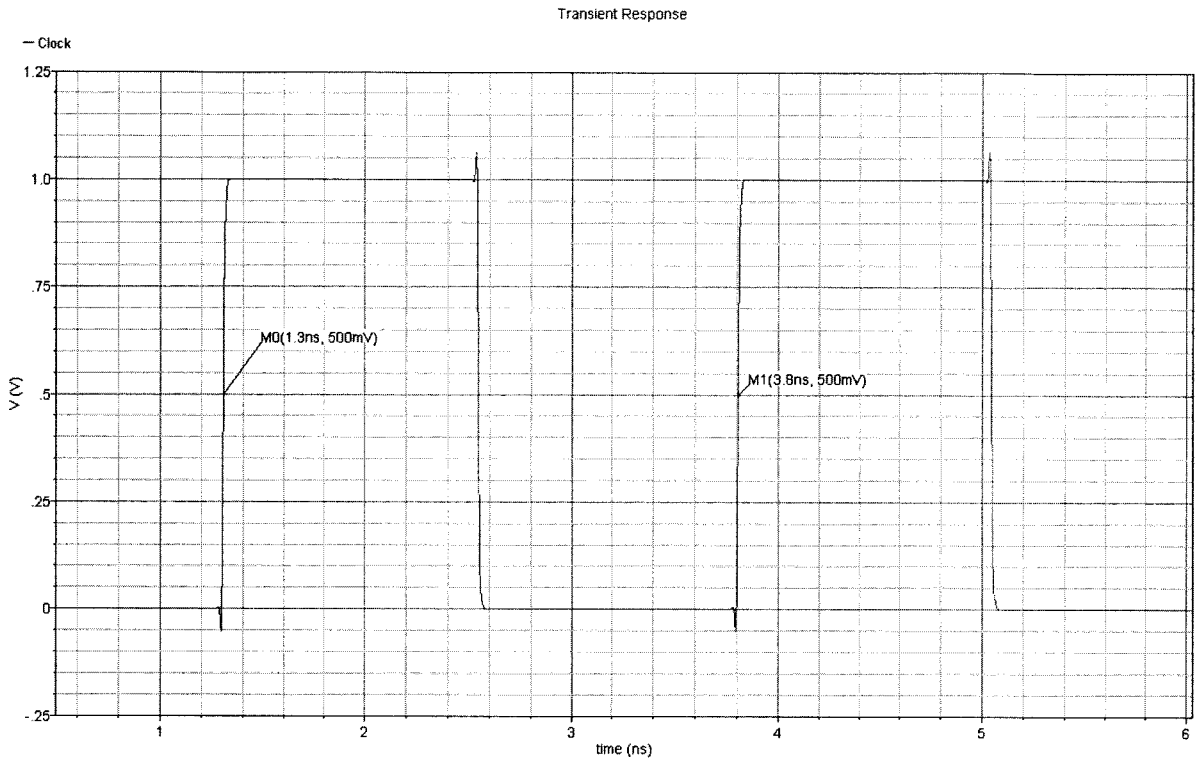

Figure 3-16: Clock Signal waveform

3.4.2 Data Generator

For power measurement, the test bench requires a data generator, shown in Figure 3-17, that can generate data patterns with different activity factors. This greatly simplifies the task of identifying data patterns for specific activity factors. The output should also be reproducible in order to achieve consistent stimulus for many different DUTs. This can be best achieved by designing a data generator [Appendix A] in VerilogA. The data generator will have different parameters to provide the different stimulus needed for the DUT power measurement. The data generator has a "SEED" parameter which is the initial seed used by the simulation tool to generate a random number between 0 and 49 with uniform distribution, every clock cycle. The generated random number is compared against the desired activity factor and if the generator random number is less than the desired activity factor, the signal, "data out", is toggled. The output data is also printed out to a log window along to a log file. A perl script [Appendix D] parses through the generated log file to get the actual data activity factor. For a given desired activity factor and a seed, the data out signal pattern is always reproducible.

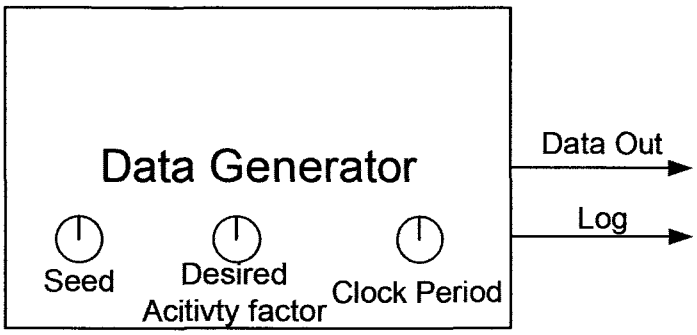

Figure 3-17: Data Generator Controls

3.4.3 Determining flip-flop timing characteristics

Flip-flop timing characteristics are found through varying the delay between the clock and data and monitoring clock-to-output propagation delay. The Data generator does not provide the stimulus to characterize the flip-flops timing parameters. Therefore a pulse generator, shown in Figure 3-18, was used to create a pulse of a certain width and occurring at a certain time to measure setup and hold time of a flip-flop.

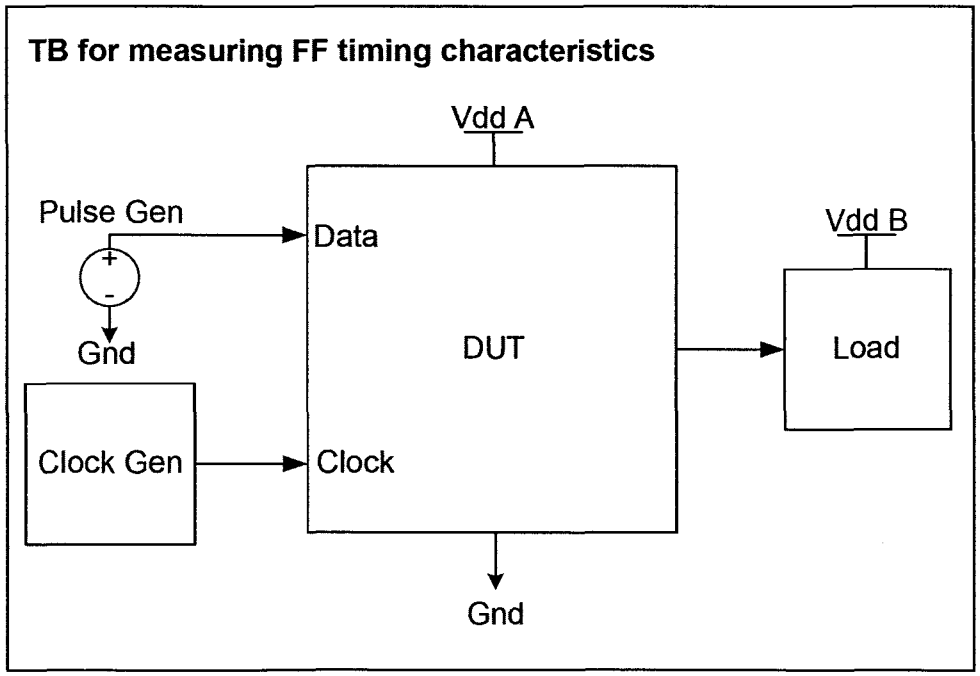

Figure 3-18: Alternate Test-bench for measuring flip-flop timing characteristics

Clock generator parameters are common for both power and timing parameter measurements for the circuits under analysis. The output of the clock-pulse generator is also passed through a chain of inverters before being fed into the DUT.

3.5 *Measuring flip-flop Performance Data*

3.5.1 Measurement of Power Consumption

Power measurement is done by averaging current throughout the duration, 500nS, of the simulation and multiplying it by the Vdd, 1V. Before the simulation is started, all the internal nodes of the flip-flop are initialized to ensure that the internal state of the flip-flop is 0. All Vdd rails are set to 1V. The VDD supply rail is connected to the storage elements and the clock tree. The VDD rail connected to the input generator is not considered while measuring power consumption. A conservative transistor model is for all simulations. All other parameters of the simulation are left to default. The output driver and loads in all circuits are identical. Pre-layout netlists are used for comparison simulations.

$$Pconsumed = \frac{1}{T}\int_0^T i(t)V_{DD}\,dt \quad \text{(Eqn 3.2)}$$

Leakage power is the average of power measured by tying the clock input to ground and tying the data input to VDD in one measurement and ground in the next measurement, shown in Figure 3-19. All internal nodes are pre-charged to either set the internal state of the flip-flop to 0 or 1. Simulation is run for 500ns and the average power consumed during the simulation is the leakage power of the flip-flop.

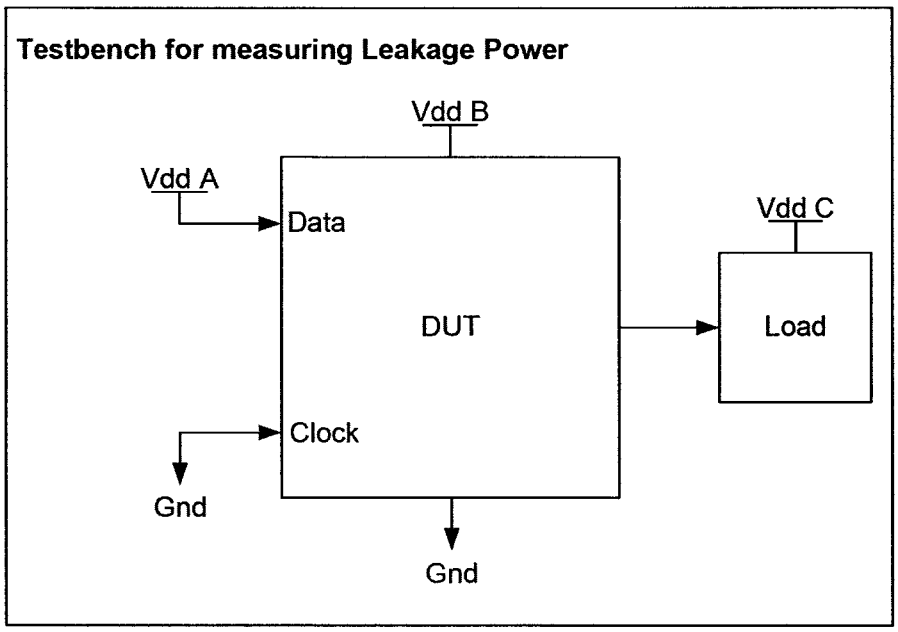

Figure 3-19: Testbench for measuring device under test leakage power

3.5.2 Flip-flop timing characteristics measurement

Setup time is sometimes defined as the absolute minimum time that the input data should settle before the clock edge for the value to be latched [32]. Sometimes setup time is defined as the data to clock offset that corresponds to an increase in clock to output propagation delay [31]. Setup time in this thesis is also defined as an increase in clock to output propagation delay. Thus, in order to find the setup time and hold time, the clock-to-output delay time needs to be measured. Parametric analysis is performed in order to find the point where the rate of change of clock-to-output delay to the setup/hold time is 1. This definition ensures that in all cases, as long as the data input meets this timing requirement, the clock to output delay of all the flip-flops do not vary by a huge margin. This greatly simplifies timing analysis.

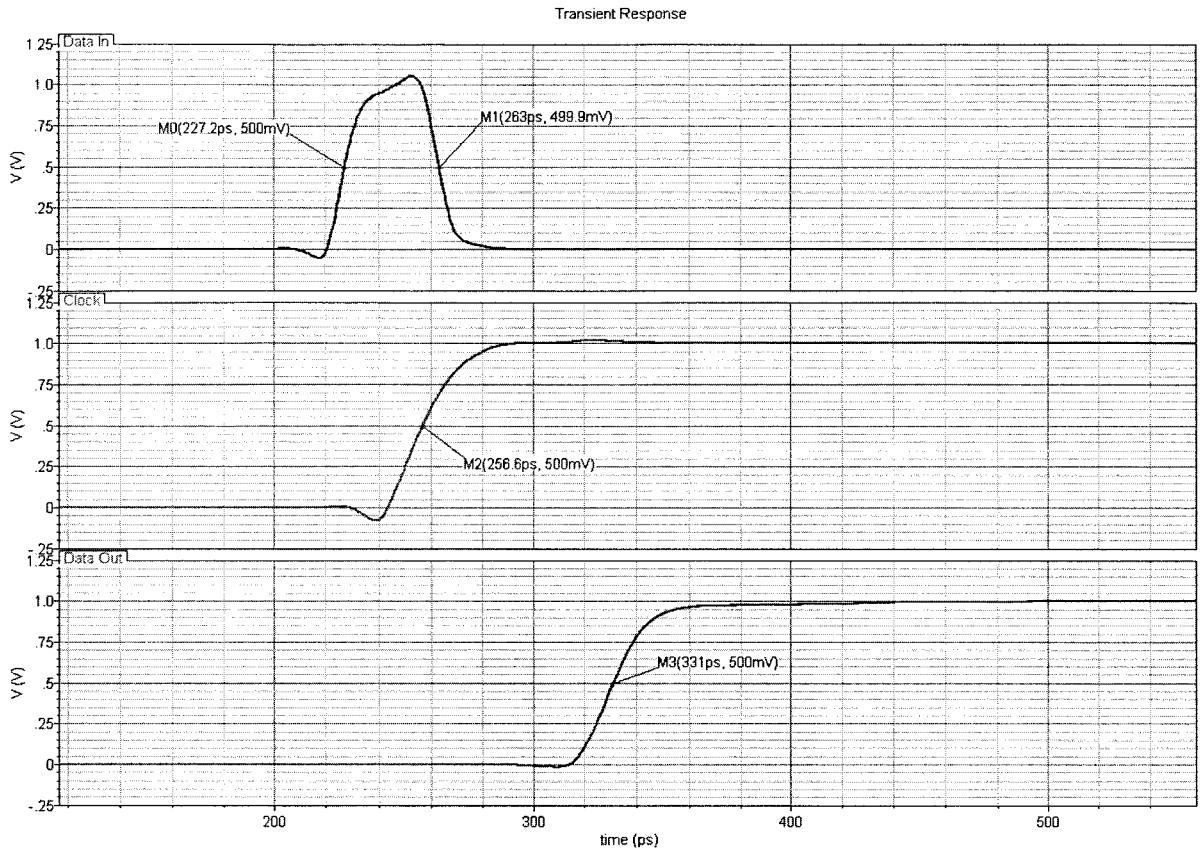

Figure 3-20: Master Slave flip-flop Setup Hold time waveform

3.5.3 Flip-flop Timing Equations

The timing parameters for the master-slave flip-flop (Figure 3-9) are given below.

$$T_{setup\_rise} = T_{Rising\ Clock\ Edge} - T_{Rising\ Data-In\ Edge} = 256.6pS - 227.2pS = 29.4pS \text{ (Eqn 3.3)}$$

$$T_{hold\_rise} = T_{Falling\ Data-In\ Edge} - T_{Rising\ Clock\ Edge} = 263pS - 256.6pS = 6.4pS \text{ (Eqn 3.4)}$$

$$T_{clock\_to\_output\_rise\_delay} = T_{Rising\ Data-Out\ Edge} - T_{Rising\ Clock\ Edge} = 74.4pS \text{ (Eqn 3.5)}$$

$$T_{race\_immunity} = T_{clock\_to\_output\_delay} - T_{hold} = 68pS \text{ (Eqn 3.6)}$$

This timing characterization methodology can be applied to the rest of the circuits under analysis without any changes.

3.6 *16-bit Counter Implementation*

The flip-flops under analysis will be used in a 16-bit counter, Figure 3-21, and the resulting power consumption will be compared. The 16-bit counter is implemented using half adder, Figure 3-22, and flip-flops. The counter will increment by 1 for every cycle the input signal, counter_enable, is asserted. There is an overflow signal which gets asserted when the counter reaches the maximum value and rolls-over to 0, when there is another request to increment.

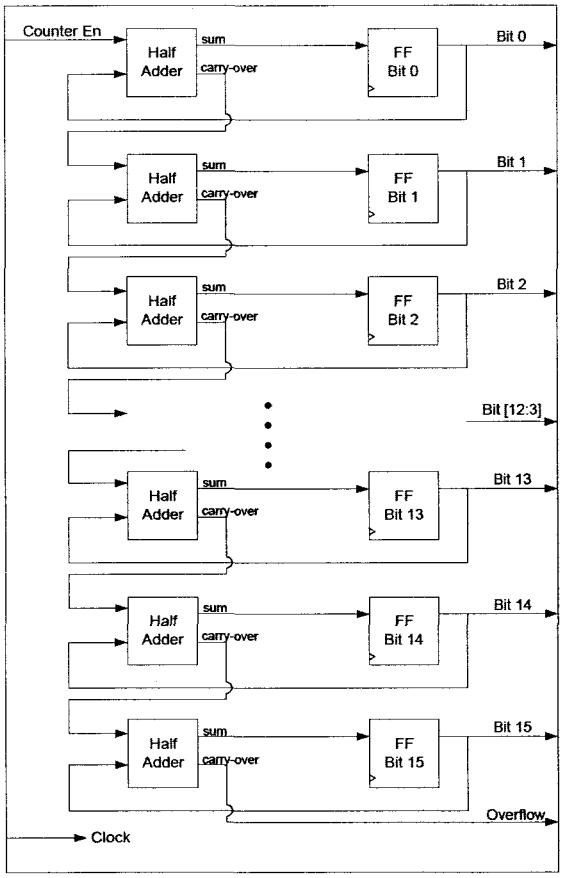

Figure 3-21: 16-bit Counter Block Diagram

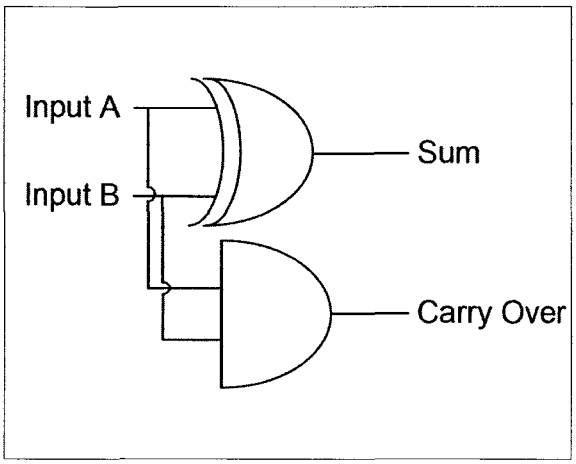

Figure 3-22: Half-Adder

Counter increment signal, "Counter_En", is generated by a Verilog-A module, enable_generator [Appendix B]. The module drives the counter increment signal based on desired assertion probability. If the assertion probability is 0, the counter increment signal is driven to 0 for the whole duration of the simulation. Setting the probability to 100 will always assert the counter increment signal for the duration of the simulation. The Clock input will be generated by a square wave generator set to output a 400MHz clock. The clock buffer circuit will be different for the circuits being tested. The circuit will be shown in the analysis of each of the flip-flops.

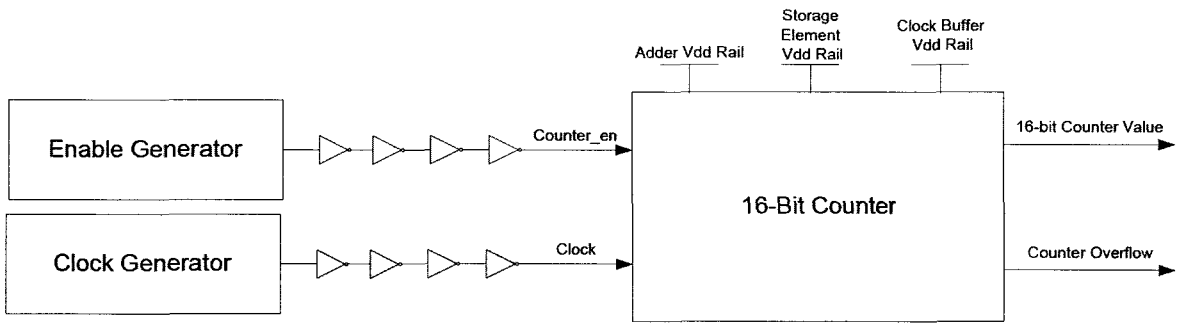

Figure 3-23: 16-bit Counter Testbench

The 16-bit counter is initialized to 0 at the beginning of the simulation. The enable generator is selected to output counter increment signals with three different probability of the signal being asserted at the beginning of the simulation: 0%, 50%, and 100%.

Each component in the 16-bit counter circuit has its own separate voltage rail in order to measure the contribution to the total power consumed. Power is calculated by averaging the current draw for the duration of the simulation at the voltage rail.

3.7 *8-bit PRBS Generator*

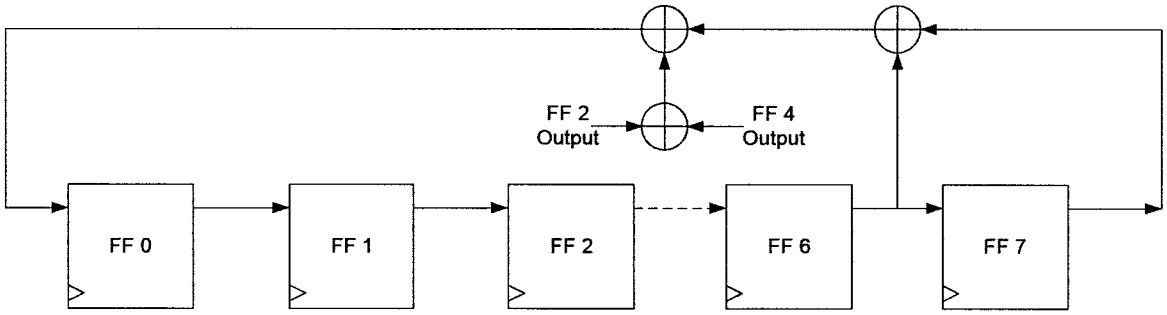

Figure 3-24: 8-bit PRBS Generator DUT

The DUT shown in Figure 3-24 is an 8-bit PRBS Generator with the polynomial, X8 +X7 +X5 +X3 +1. FF 0 is set to 1 and the rest of the flip-flops are set to 0 so the PRBS is seeded with a valid value. The test-bench, shown in Figure 3-25, contains the DUT, supply voltage rail, and a clock generator. The data input to each of the flip-flops are generated and self-contained in the DUT.

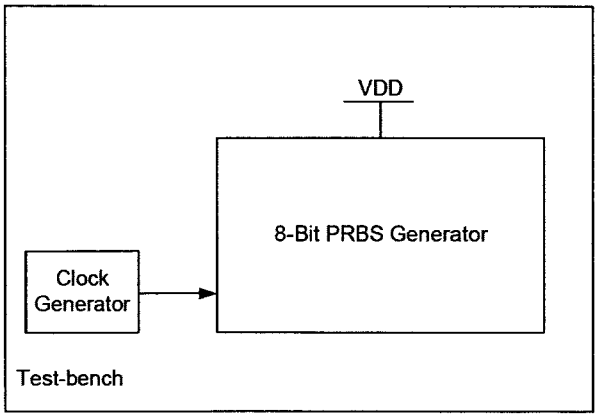

Figure 3-25: 8-bit PRBS Test-bench

3.8 *Data Bus Test-bench*

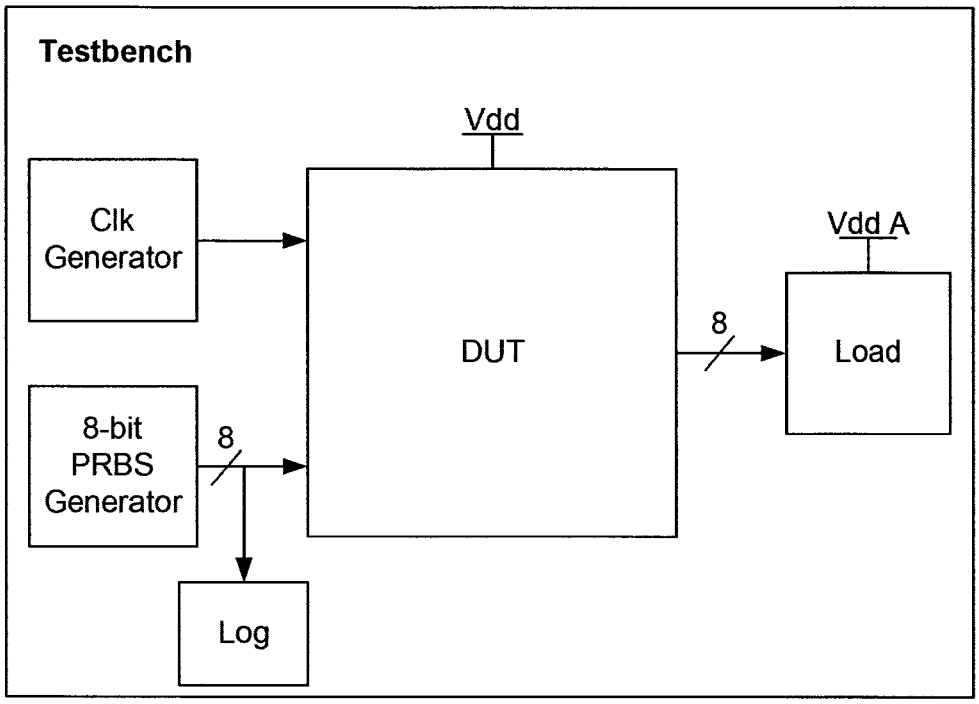

Figure 3-26: 8-bit Data Bus Test-bench

The Data Bus test-bench, shown in Figure 3-26, is used to test the power consumption of the data bus under different activity factors. The 8-bit data bus is a collection of eight flip-flops with no other logic present in the DUT. The input to the DUT is provided by an 8-bit PRBS generator [Appendix C] designed in Verilog-A. The output of the PRBS generator is passed through a chain of inverters. The PRBS generator is intended to provide a random input sequence and the initial value is set to 1, in order to generate a valid sequence. The PRBS generator outputs 0s when the initial value is set to 0.

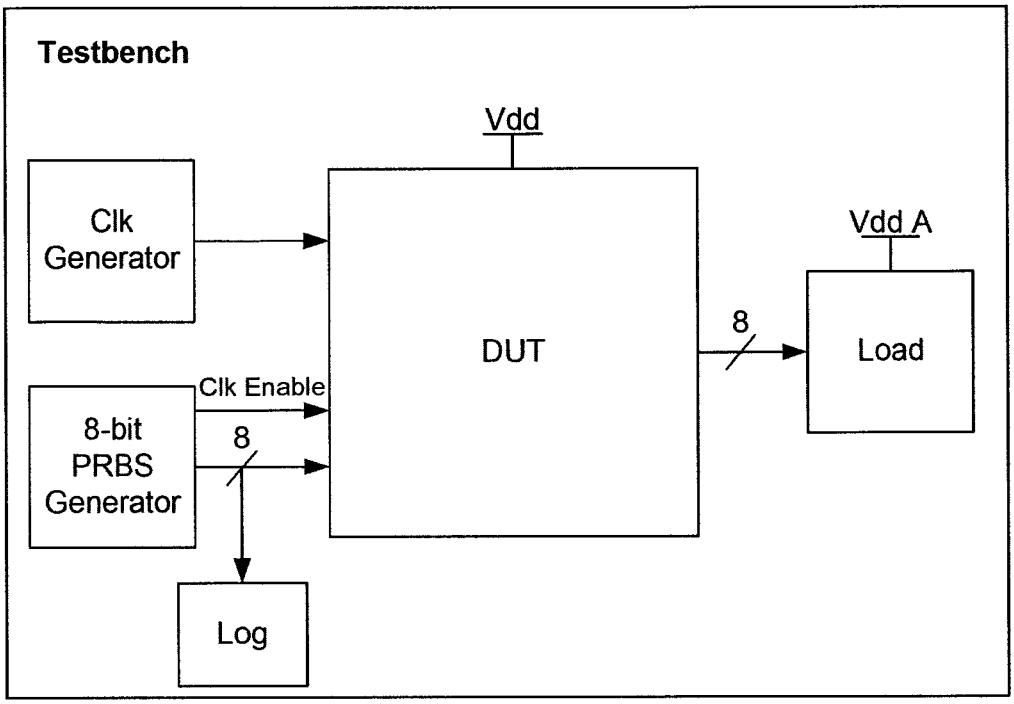

Figure 3-27: Alternate 8-bit Data Bus Test-bench with Clk Enable

The alternate 8-bit Data Bus Test-bench generates 8-bit data PRBS pattern and a "Clk Enable" signal, shown in Figure 3-27. This is needed to test DUTs which support the clock gating feature. While the PRBS generator is not producing a new sequence, the "Clk Enable" signal is de-asserted. The "Clk Enable" signal will be de-asserted for 50% of the simulation when the PRBS generator activity factor is set to 25%.

Chapter 4 **Existing flip-flop Architectures**

The standard master slave flip-flop, sometimes referred to as the conventional D flip-flop, is one of the most widely used flip-flops in VLSI Systems design [18]. The standard master-slave flip-flop is a well understood dual latch, master latch and slave latch, design. The standard master-slave flip-flop consumes a lot of power even when the data input to the flip-flop may not be toggling or valid.

4.1 ***Standard Master-Slave flip-flop Design***

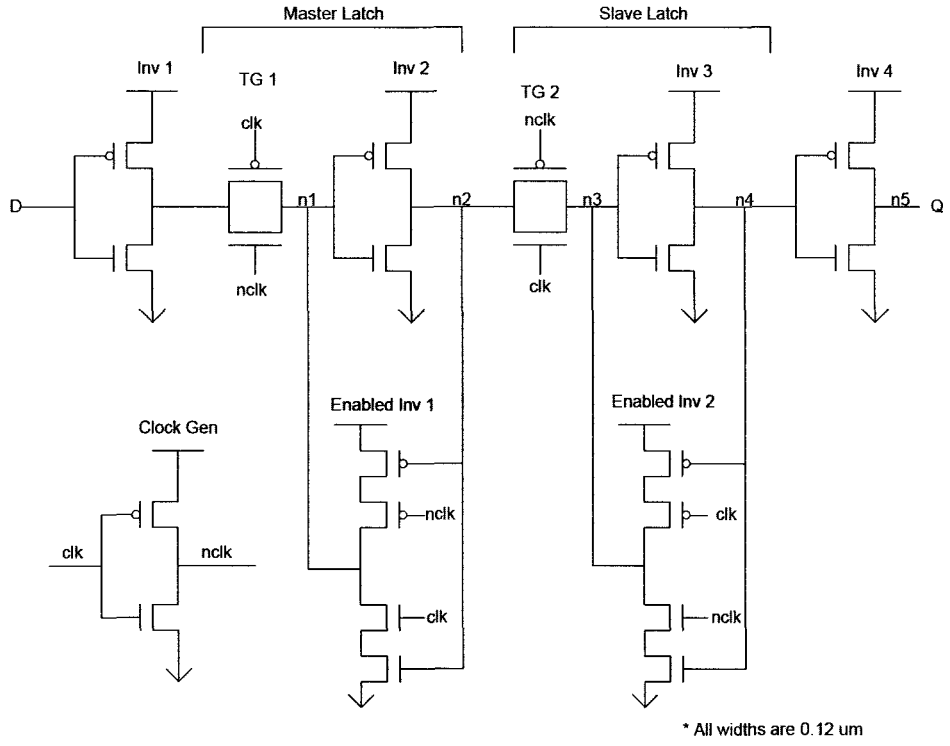

Figure 4-1: Standard Master-Slave Flip Flop Schematic

The function of the circuit, in Figure 4-1 standard master slave flip-flop, is a D flip-flop. This circuit latches the input data at the rising edge of the clock input signal, "Clk" input pin. The circuit is comprised of a master latch, slave latch, output driver, and a clock buffer. The master latch, slave latch, and the output driver circuits are collectively referred to as the storage

element. The master latch is transparent, illustrated in Figure 4-3, when the clock, signal "clk", is de-asserted. The master latch is not transparent when the clock is asserted as illustrated in Figure 4-4. The slave latch however is transparent when the clock signal is asserted. However, it is not transparent when clock is de-asserted. The master-slave flip-flop operates as positive edge triggered flip-flop in the current configuration of "clk" and "nclk" wiring. Switching the wiring produces a negative edge triggered flip-flop. The relationship between the input and output signals are shown in Figure 4-2. Switched capacitance can be reduced by minimizing transistor sizes thus saving power consumption [35]. Thus minimum sized transistors are used where-ever possible.

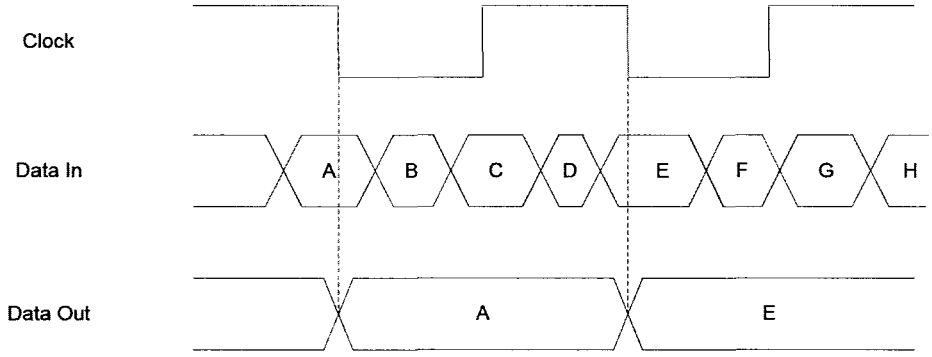

Figure 4-2: Relationship between flip-flop Data In and Data Out

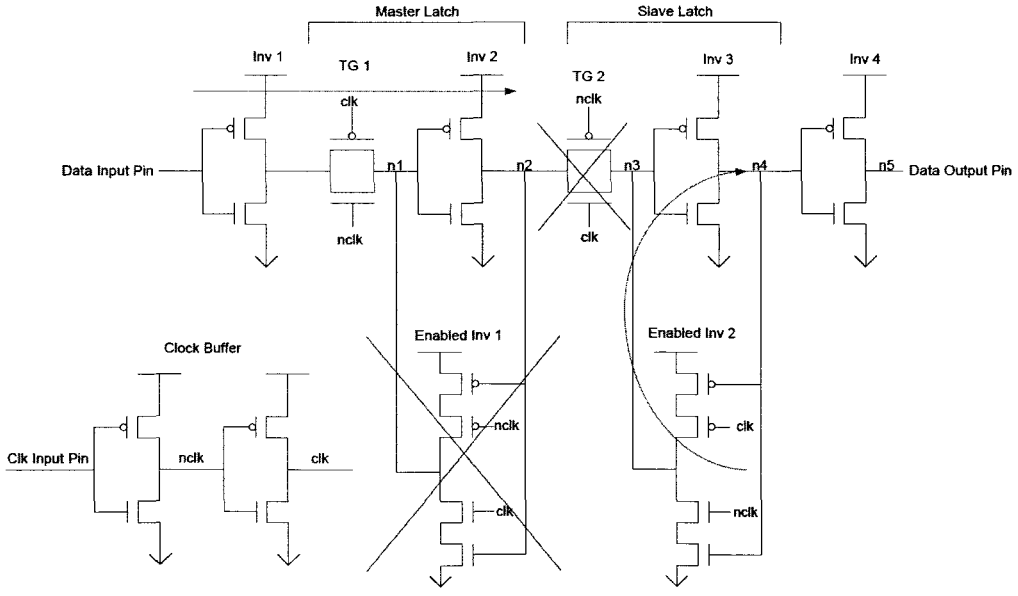

Figure 4-3: Standard Master Slave flip-flop data path when clk is de-asserted

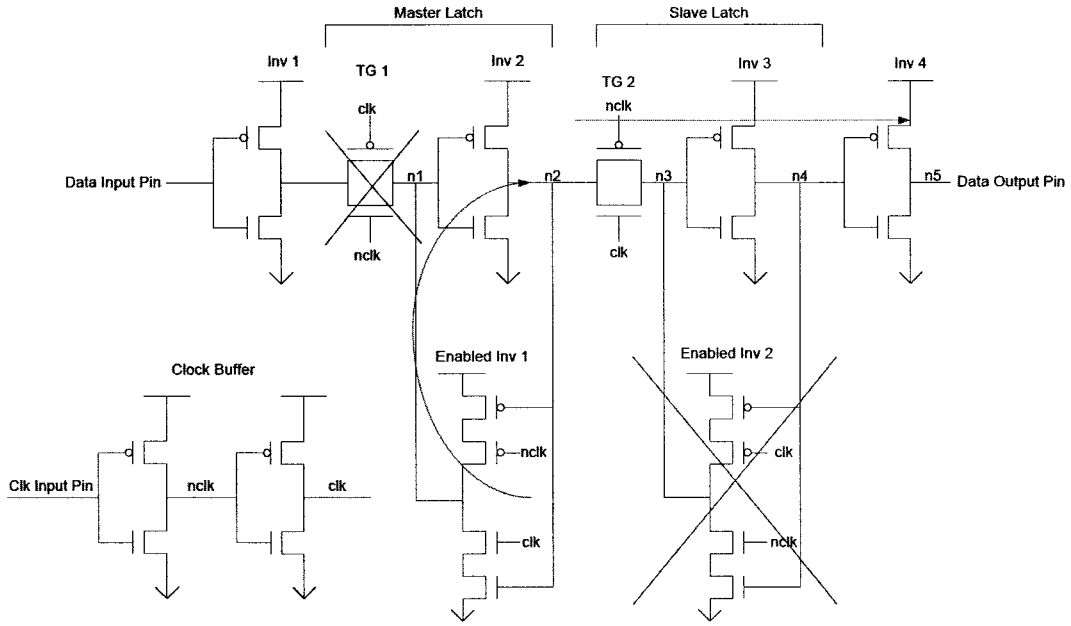

Figure 4-4: Standard Master Slave flip-flop data path when clk is high

The clock buffer is comprised of two inverters. The input to the clock buffer is the "Clk Input Pin", the output of the first inverter is the signal "nclk", and the output of the second inverter is the signal "clk". This configuration provides us with a positive edge triggered flip-flop.

4.1.1 Clock Gate flip-flop Design

Clock gated flip-flops allows the gating of the clocks at a fine level of granularity. Portions of circuits which are inactive can thus be deactivated resulting in power reduction in the clock circuits internal to the flip-flop [19].

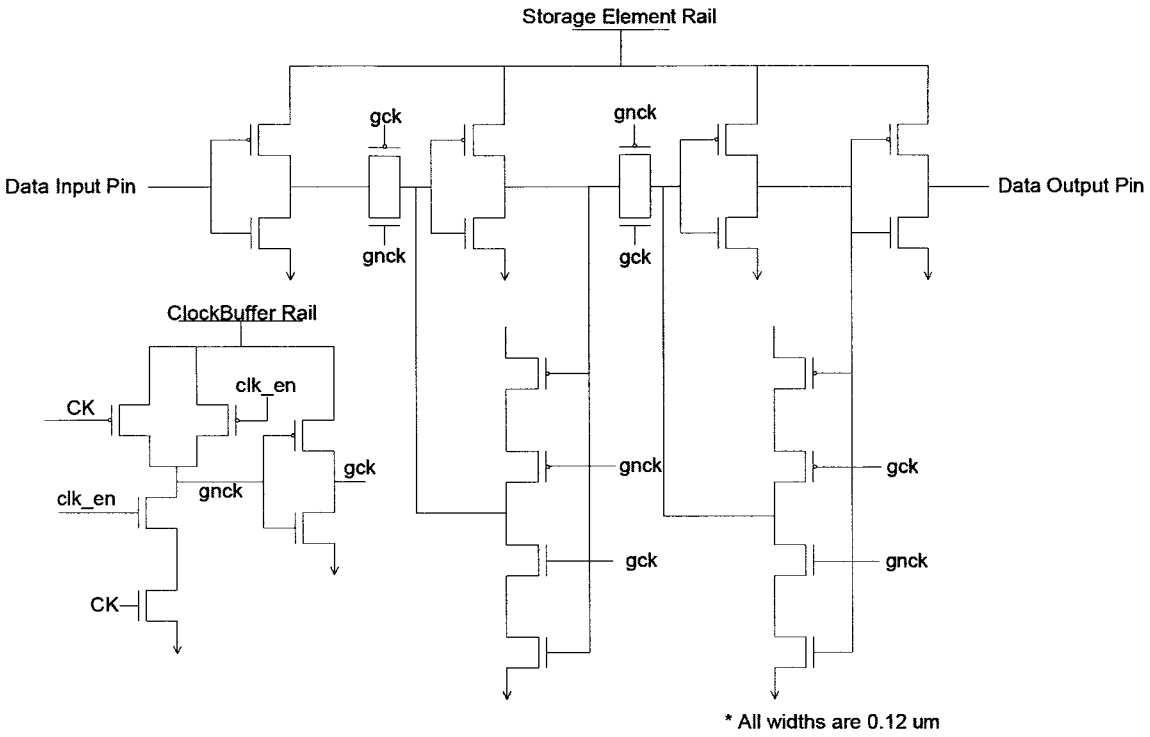

Figure 4-5: Clock-gated Master Slave flip-flop Schematic

The function of the circuit, shown in Figure 4-5: clock-gated master slave flip-flop is a D flip-flop with the added ability to gate the clock input pin based on the "clk en" signal. The circuit is comprised of a master latch, slave latch, output driver, clock gate, and a clock buffer. When the clock enable signal, clk en, is asserted, the circuit functionally behaves the same as a master slave flip-flop. When the clock enable signal de-asserted, the flip-flop does not latch the input data signal, "Data Input", at the rising clock edge, but drives the load with the data stored previously at the last rising edge of the clock when clk en was asserted. The relationship between "Data Input Pin", "clk en", and "Data Output Pin" is shown in Figure 4-6.

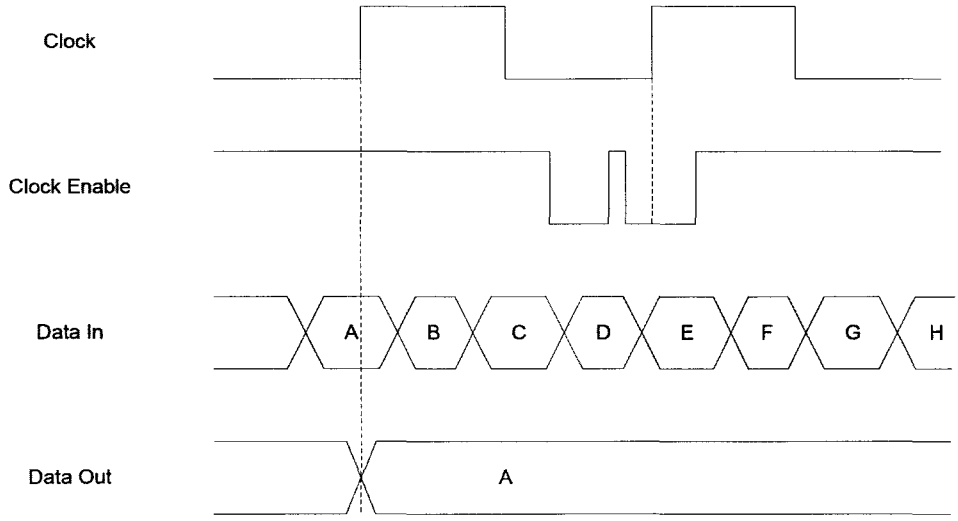

Figure 4-6: Data-In to Data-Out relation for Clock-gated flip-flop

4.1.2 Clock-gated flip-flop Operation

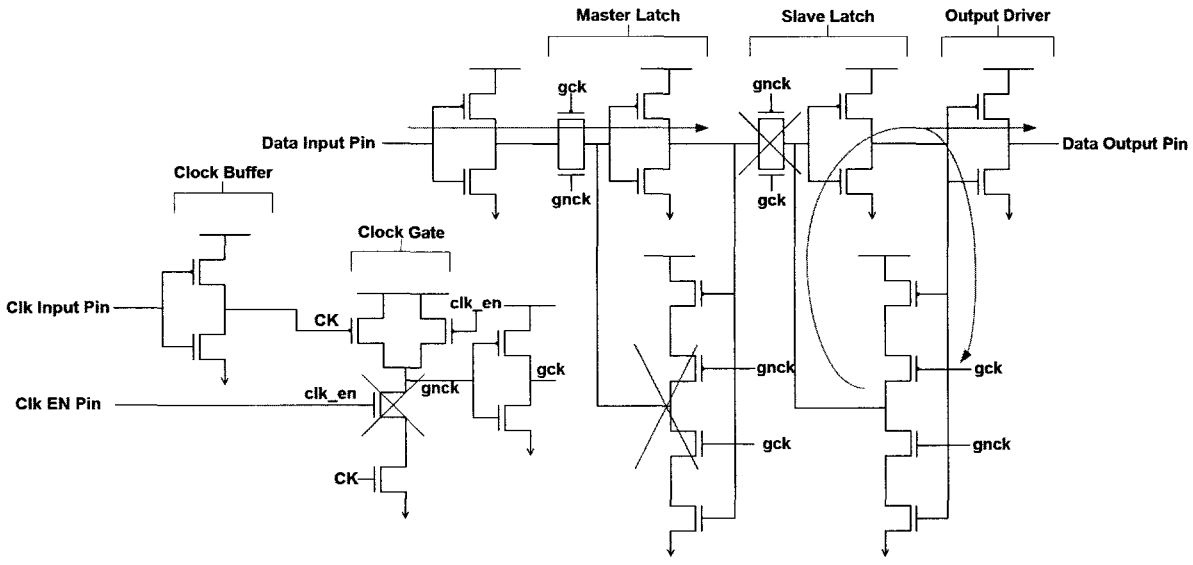

Figure 4-7: Clock-gated Master Slave flip-flop when ClkEN is de-asserted

When the "clk en" signal is de-asserted, master latch is transparent but the slave latch is not transparent. When "clk en" is low the first transmission gate, of the master latch, is transparent. This makes output of the master latch dependant on the input signal. However the transmission gate, of the slave latch, is not transparent; hence there is a feedback loop that stores the data while driving the data output pin. This is illustrated in Figure 4-7

4.1.3 DTLA flip-flop Design

Data transition look-ahead (DTLA) flip-flop reduces power consumption by gating the internal clock signal. Power consumed is reduced in accordance with the data transition probability. This is due to internal clock signal being gated when there are no data transitions [20].

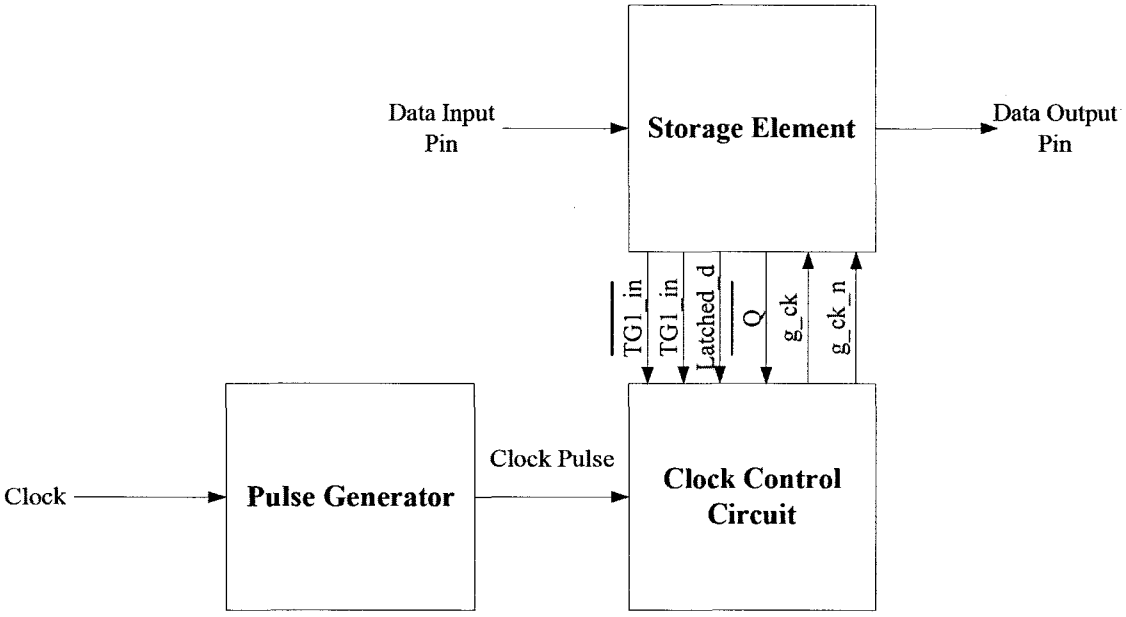

Figure 4-8: Data Transition Look Ahead flip-flop Schematic

The function of the circuit shown in Figure 4-8, DTLA flip-flop, is a D flip-flop with the added ability to gate the clock input pin based on the Data input value and Data output value. The DTLA flip-flop is based on the pulsed flip-flop. A pulse is generated either at the positive or negative edge of the clock which causes the latch to be briefly transparent and capture the input data. The generated pulse to the latch is gated if the Data input value and the Data output values are the same. The circuit is comprised of a dynamic latch, a static latch, output driver, clock control circuit, and a clock pulse generator. The dynamic latch should not be confused with dynamic digital circuits belonging to Domino Logic Family or Nora Logic Family. The overall circuit functionally behaves the same as a negative edge triggered master slave flip-flop. The data input is latched at the falling edge of the clock.

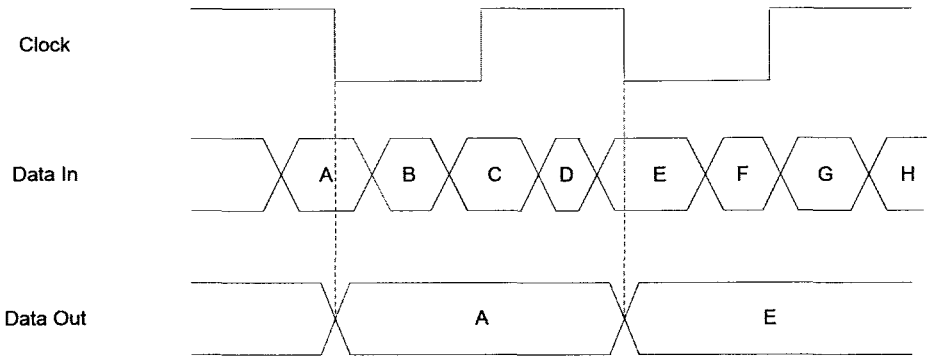

Figure 4-9 : Data-In to Data-Out relation for Clock-gated flip-flop

The specific implementation of the DTLA flip-flop in this report captures data at the falling edge of the clock. Data input is sampled every clock cycle. The clock pulse generator circuit, shown in Figure 4-10, generates a short clock pulse at the negative edge of the clock input.

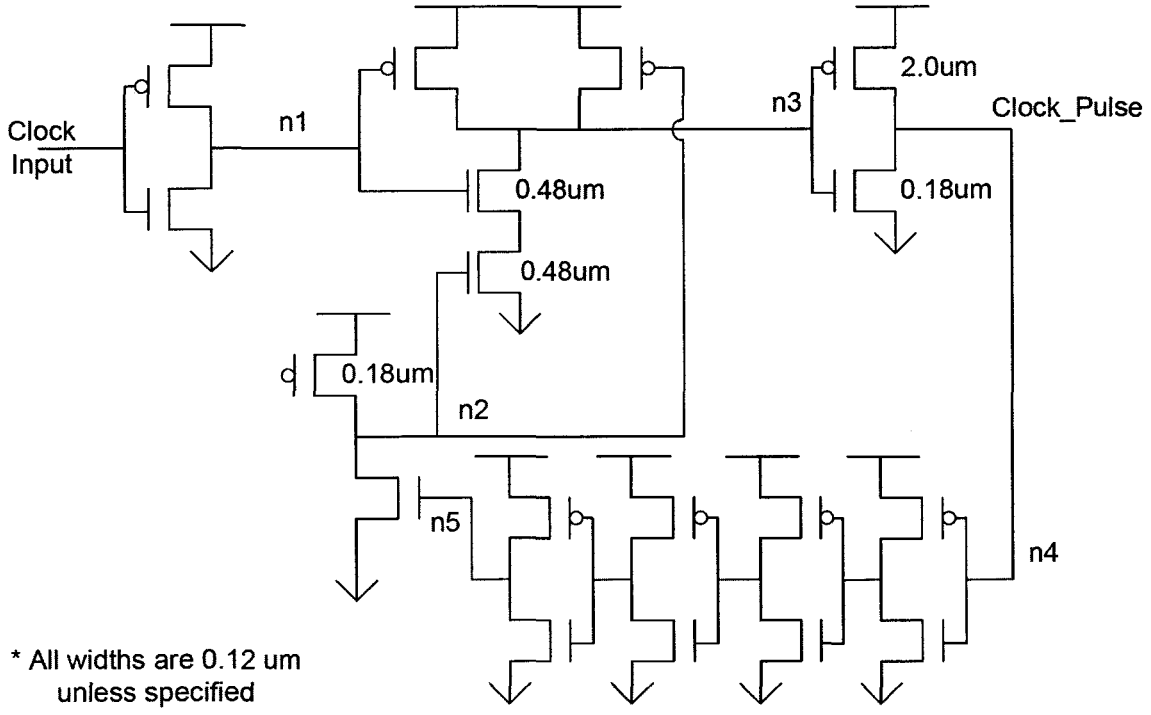

Figure 4-10: Clock Pulse Generator

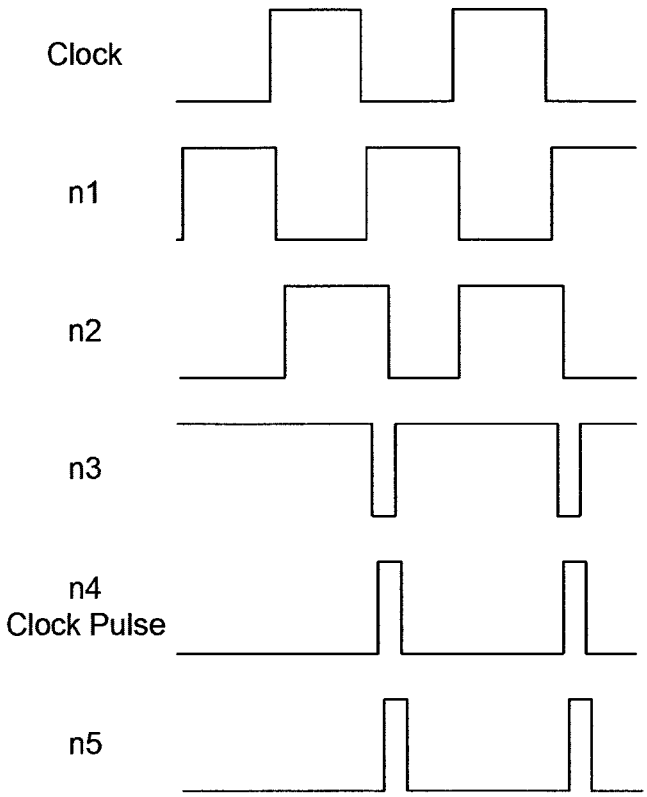

Figure 4-11: Pulse Generator timing waveform

Node, n1, is the inversion of the clock input signal. When Clock is asserted, n1 is de-asserted. This causes m1 to charge node, n2. At this instant, the input to the NAND gate is 0 & 1, which causes the NAND gate to charge node, n3. When Clock is de-asserted, n1 is asserted. At this moment, node n2 is not driven since the input to m1 is 1 and input to m2 is 0. The inputs to the NAND gates are 1 &1, which causes the NAND gate to discharge node, n3. An inverter delay later node n4 is charged which also happens to be the signal Clock Pulse. Four inverter delays later, node n5 is asserted. This discharges the node n2, which causes node n3 to be charged. Node n4 is discharged an inverter delay later thus producing a short clock pulse.

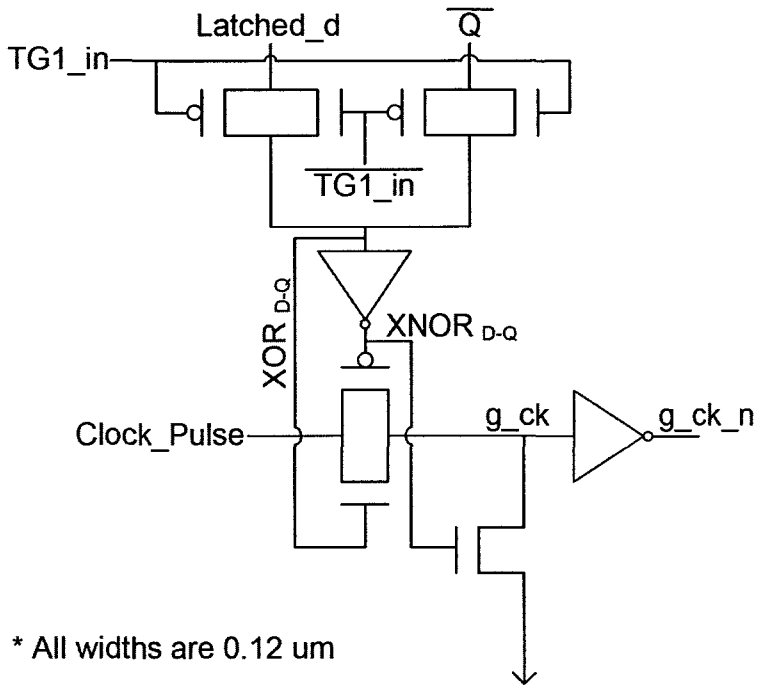

Figure 4-12: DTLA Clock Control Circuit

The internal storage element signals are used to drive an xnor circuit built using complimentary pass transistor logic, shown in Figure 4-12. The output of the xnor circuit is equivalent to the xor of Data input and Data output signals. The output of the circuit is used to drive an "AND" gate (built using complimentary pass transistor logic). The inputs to the "AND" gate are the clock pulse signal and outputs of the xnor circuit. When signal XOR_{D-Q} is de-asserted, the Clock Pulse signal is gated. The outputs of the clock control circuit remain unchanged. When the signal XOR_{D-Q} is asserted, the clock pulse signal is allowed to propagate to the outputs of the circuit.

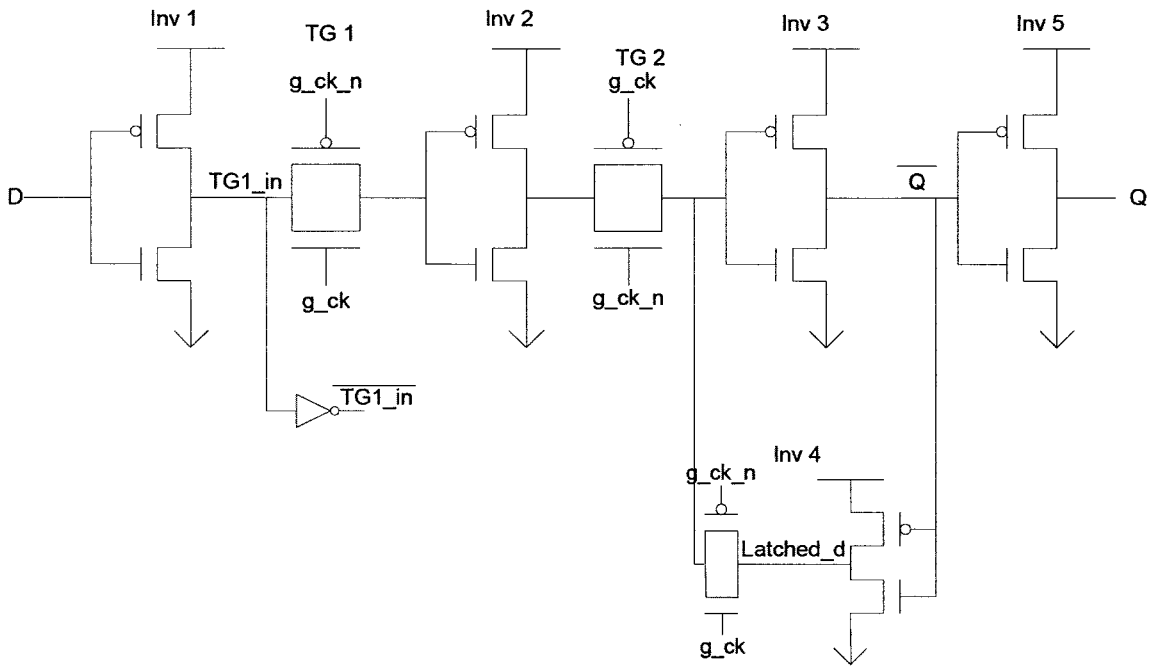

Figure 4-13: DTLA flip-flop Storage Element

The storage element is composed of two latch stages. The first stage is a dynamic latch and the second stage is the slave latch. The dynamic latch is not absolutely necessary for the operation of this flip-flop. It is added to improve the hold timing characteristics of the flip-flop. Without the dynamic latch, the hold time of the flip-flop will increase [20] as shown in Appendix I. The difference in setup time is negligible but the difference in hold time is 44 ps, on average.

4.1.4 Clock on Demand flip-flop Design

Clock on demand flip-flop reduces power consumption over standard master-slave flip-flop by gating the clock input signal when the data input does not toggle [31]. Clock gating is integrated into the pulse generator thus it should have lower power consumption than DTLA flip-flop, since a clock pulse is only generated if the data stored in the storage element differs from the data input.

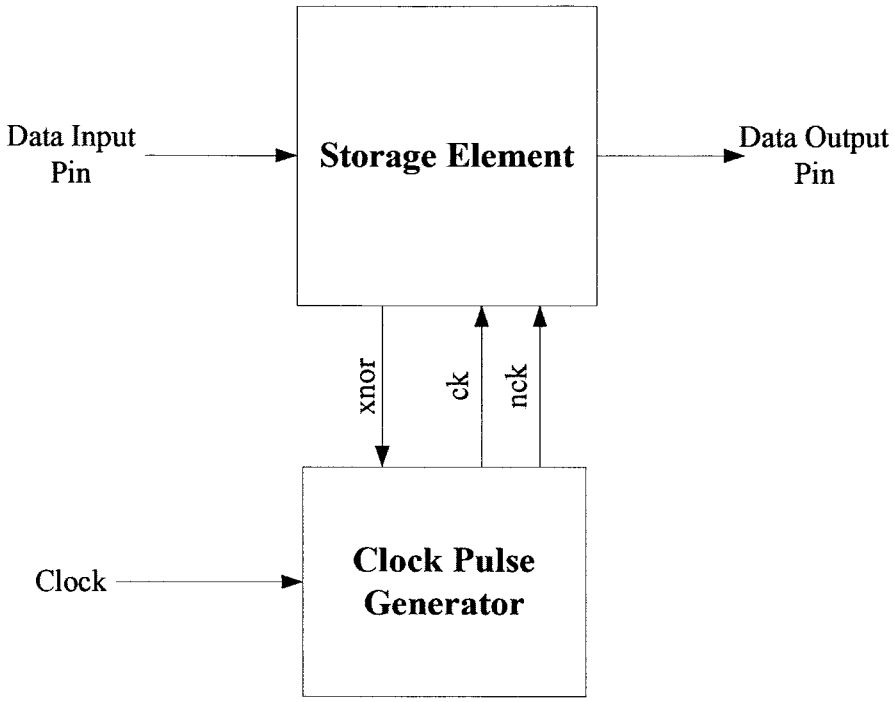

Figure 4-14: Clock on Demand flip-flop Schematic

The function of the circuit shown in Figure 4-14, Clock on Demand (CoD) flip-flop, is a D flip-flop with the added ability to gate the clock input pin based on the Data input value and Data output value. The CoD flip-flop is based on the Pulse-Triggered Latch. A pulse is generated either at the positive or negative edge of the clock which causes a latch to be briefly transparent and capture the input data. The difference between the CoD flip-flop and a pulse-triggered latch is that the generated pulse to the storage element is gated if the Data input value and the Data output values are the same. The circuit is comprised of a static latch, output driver, xnor circuit, and a clock pulse generator. The Storage Element, Figure 4-14, consists of latch, output driver, and a xnor circuit. The overall circuit functionally behaves the same as a negative edge triggered master slave flip-flop. The data input is latched at the falling edge of the clock. The CoD flip-flop implemented for this thesis captures data at the falling edge of the clock. Data is latched at the falling edge of every clock cycle.

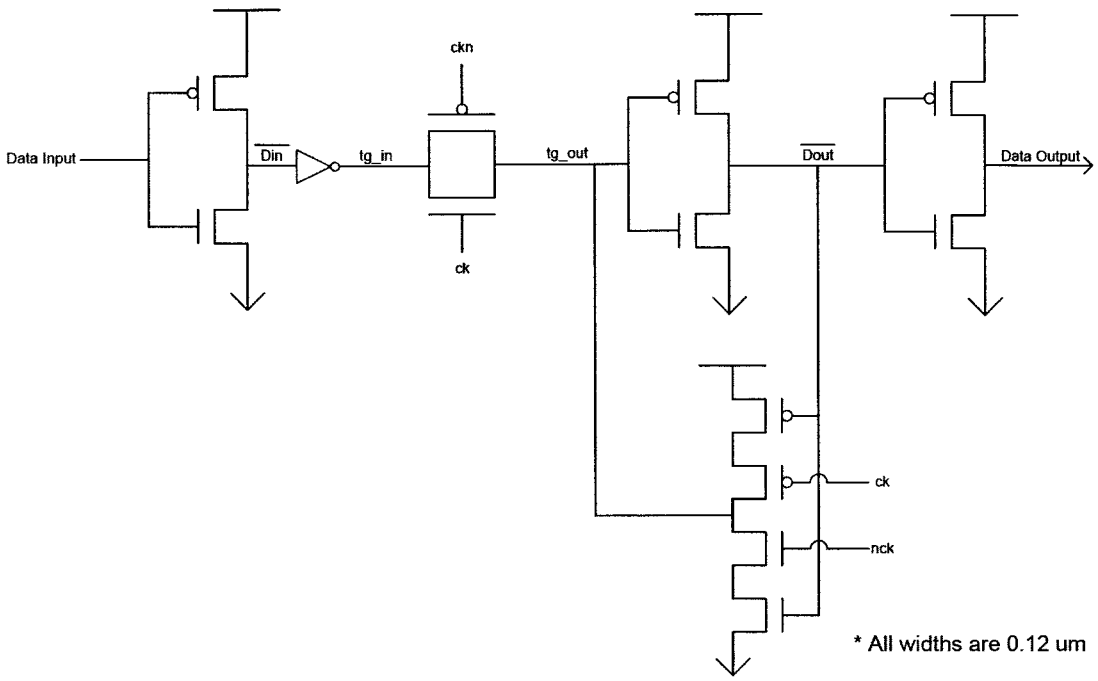

Figure 4-15: CoD Latch and Output Driver

The storage element is contains a single stage latch, xnor circuit for data comparison, and an output driver. The latch is transparent when the clock pulse signal, "ck", is asserted. The xnor circuit is built using complementary pass transistor logic to reduce area and power consumption. The internal storage element signals are used to drive the xnor circuit, Figure 4-16. The output of the xnor circuit is equivalent to the xor of Data input and Data output signals. The output of the circuit is used to drive the clock pulse generation circuit.

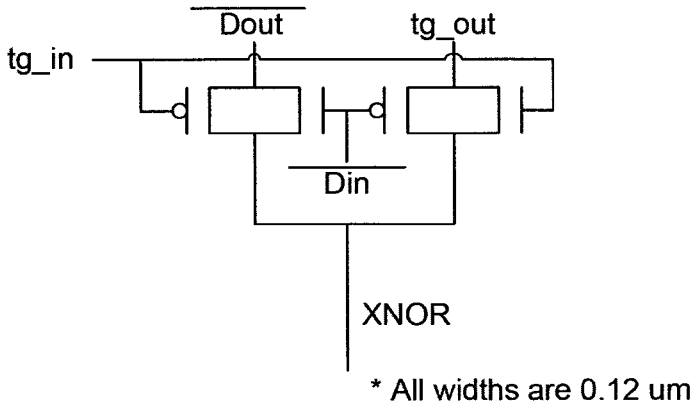

Figure 4-16: Clock on Demand XNOR Circuit

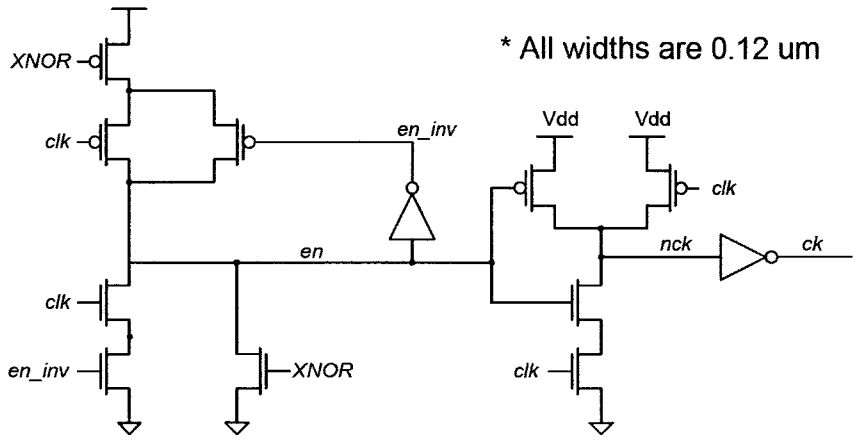

Figure 4-17: Clock Pulse Generator

The clock pulse generator circuit, shown in Figure 4-17, generates a short clock pulse at the negative edge of the clock input if the data input is different from the latched data. The signal, "XNOR", is de-asserted when data input and the data stored in the storage element are different. The signal, "en", is asserted when both signals, "Clk" and "XNOR", are de-asserted. This acts as a clock enable, in the clock gating circuit. When the data-input is latched, "XNOR" signal gets asserted which discharges the signal "en". This gates the clock causing the latch to no longer be transparent. Due to this mechanism of using xnor to de-assert the clock pulse signal, the CoD flip-flop's hold time is very close to its propagation delay resulting in extremely low race immunity.

4.2 *16-Bit Counter Power Consumption*

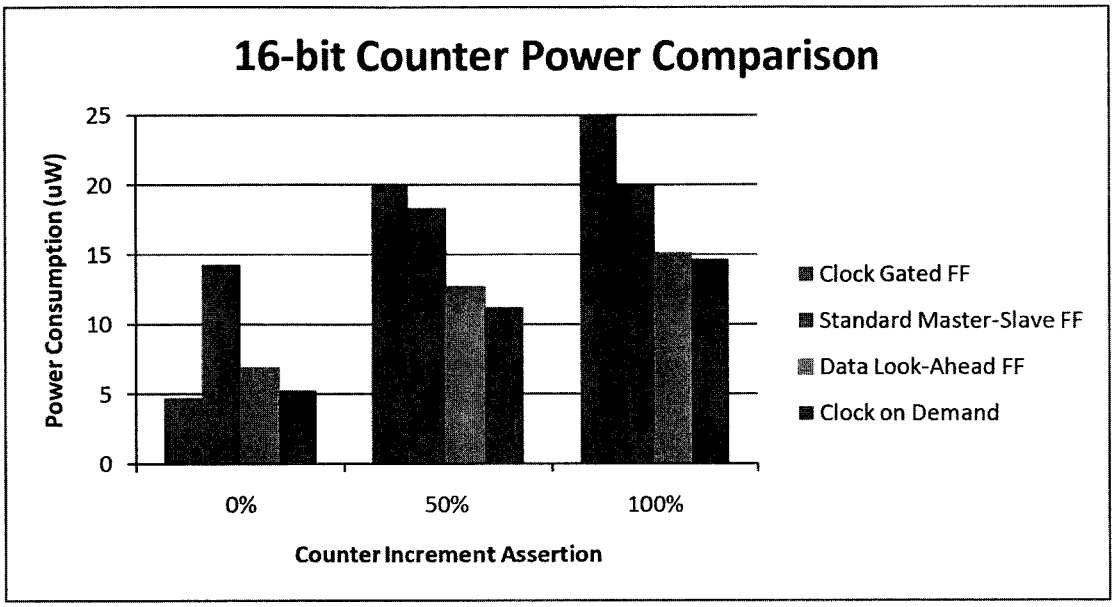

Figure 4-18: 16-bit Counter Power Consumption Comparison

The 16-bit counter implemented with CoD flip-flop has lower power consumption than all other counter implementations when the counter increment rate is higher than 0%. Only the 16-bit counter implemented with clock gated flip-flop has lower power consumption, by 24%, than the 16-bit Counter implemented with CoD flip-flop when the counter increment rate is 0%. The difference in power consumption is attributed to the higher internal clock capacitance in the CoD flip-flop. The reason for the pronounced non-linearity in the power consumption measurement of the counter implemented with clock gated flip-flop is due to the clock-gating of the 16 flip-flops with the increment signal. Clock signal to the counter is not gated when counter increment signal is asserted. When counter increment signal is de-asserted, the power consumption of a single clock-gated flip-flop is represented by the "Clock-gated flip-flop (clk_en =0)" line in Figure 4-19. When counter increment signal is always asserted, the power consumption of a single clock-gated flip-flop is represented by the "Clock-gated flip-flop (clk_en =1)" line in Figure 4-19.

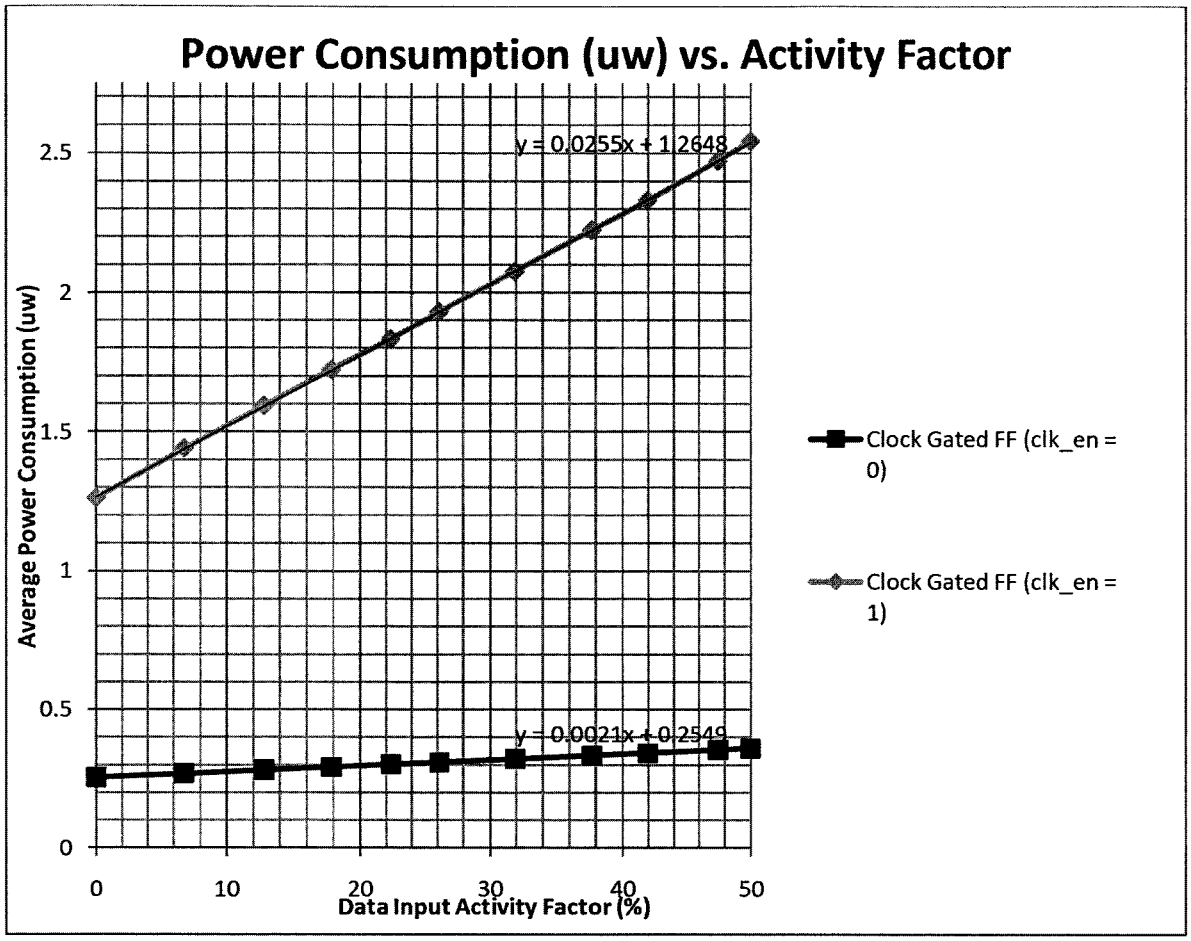

Figure 4-19 : Clock-gated flip-flop Power Consumption vs. Data Activity Factor

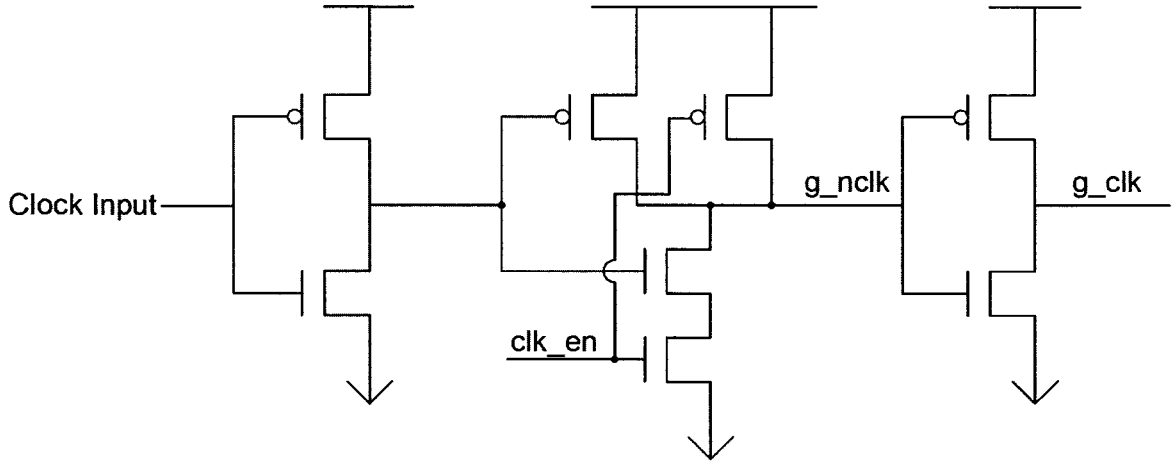

Figure 4-20: Clock-gated flip-flop Clock Component Activity Factor

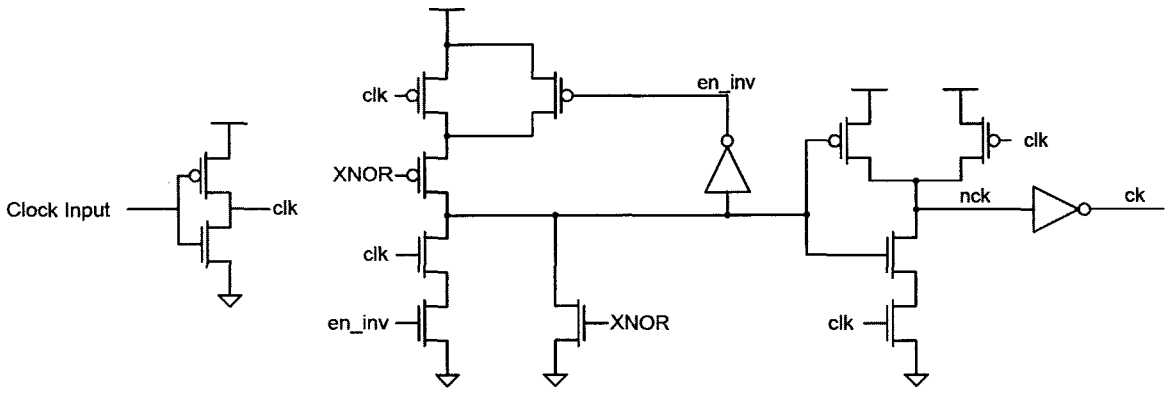

Figure 4-21: CoD flip-flop Clock Component Activity Factor

In the case of clock-gated flip-flop, apart from the input inverter, only two gates are being charged and discharged every cycle when the "clk_en" signal is de-asserted. In the case of CoD flip-flop six gates are being charged and discharged every cycle when the "XNOR" is asserted, in other words when data input does not toggle. Hence when counter increment rate is 0%, the clock gated flip-flop counter implementation manages to have lower power consumption.

4.3 *32-bit Counter Power Consumption*

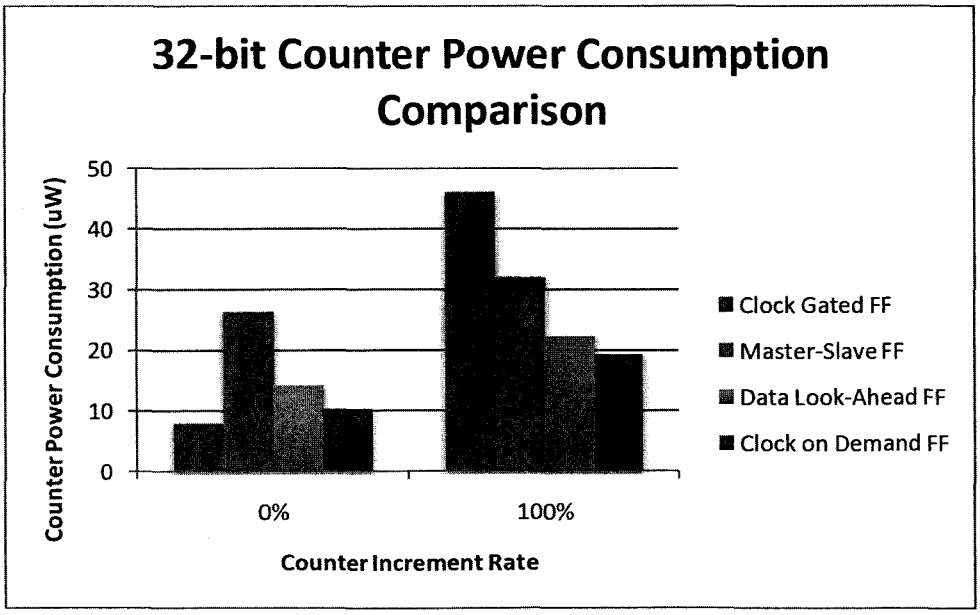

Figure 4-22: 32-bit Counter Power Consumption Comparison

The 16-bit counter power consumption trend continues with 32-bit counter implementation. 32-bit counter implemented with clock-gated flip-flop consumes less power than 32-bit counter implemented with CoD flip-flop. 32-bit counter implemented with CoD flip-flop consumes less power in the all other scenarios. The average activity factor of the counter bits are reduced when the counter width is increased thus helping the counter implemented with CoD flip-flop reduce power consumptions over the other implementations. Even though CoD flip-flop and DTLA flip-flop use the same principle to reduce power consumption, the power consumption of the counter flip-flop implemented with CoD flip-flop is lower than the counter implemented with DTLA flip-flop due to the pulse generator power consumption overhead.

4.4 *Timing Characteristics*

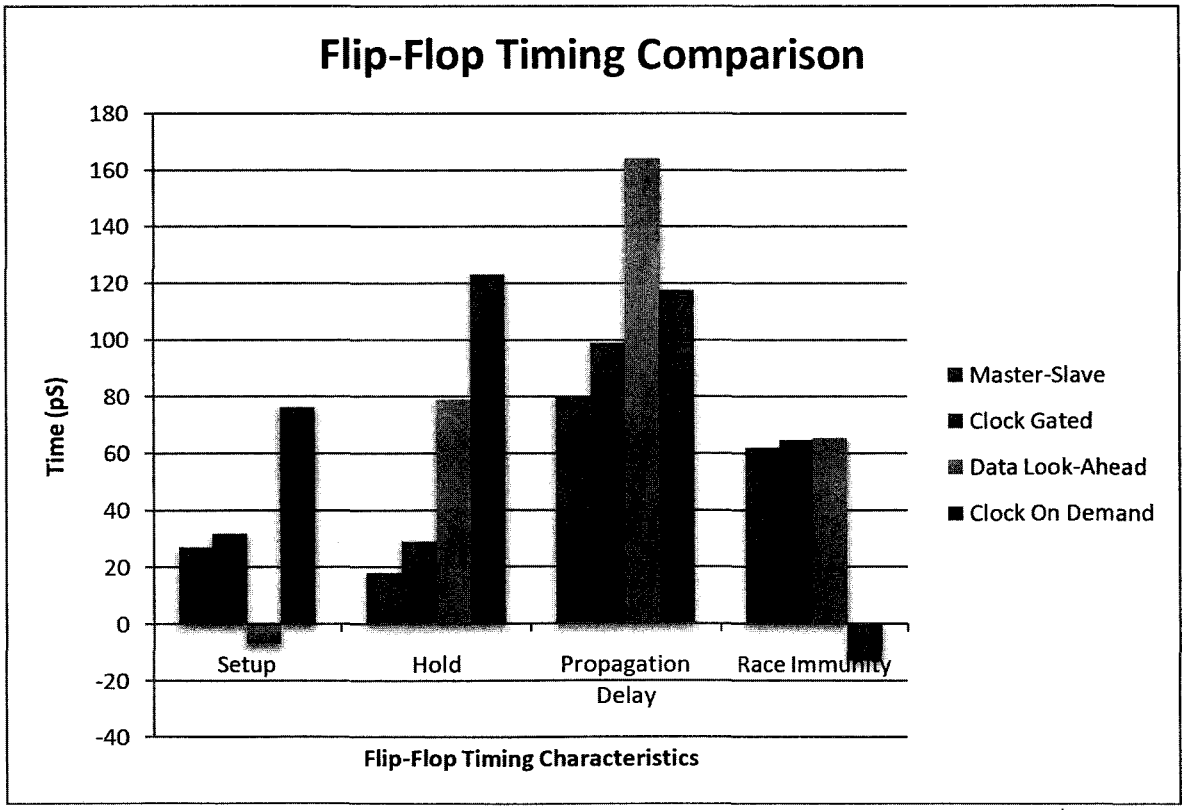

Figure 4-23: Flip-flop Timing Characteristics Comparison

CoD flip-flop's setup time is longer than other flip-flops due to the serial nature of generating the clock pulse. In order for the clock pulse to be generated, "EN" signal needs to be asserted. The "XNOR" signal needs to be asserted in order to generate the "EN" signal. The worst case hold time is longer than the propagation delay resulting in low race immunity for the CoD flip-flop implementation. This will lead to a hold time violation in the CoD flip-flop. The reason the 16-bit and 32-bit counter implementations did not exhibit hold time violations are due to the half-adder logic between each stages of flip-flop, which added delay. Implementing a pseudo random shift register will clearly show that the CoD flip-flop will cause the circuit to fail due to hold time violation.

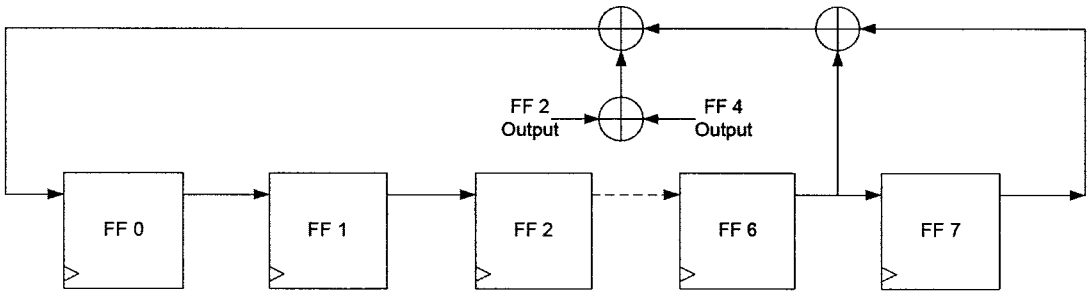

Figure 4-24: 8-bit PRBS implementation

FF 1's hold time is violated as the data from flip-flop 0 falls before the hold time can be met. Adding inverters between the flip-flops added enough delay to satisfy the hold time requirements of the clock on demand flip-flops.

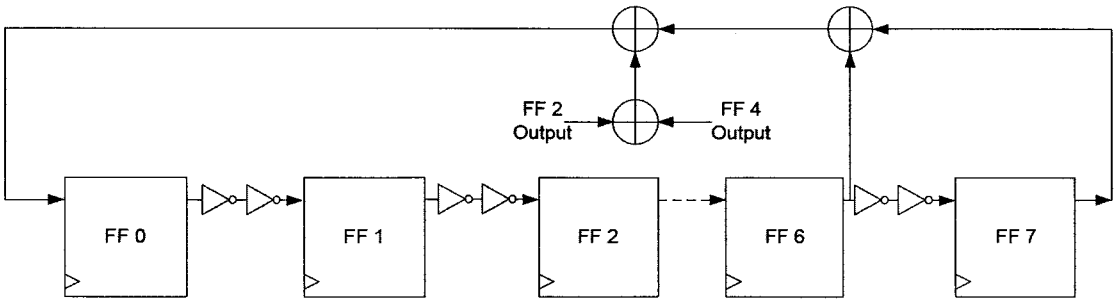

Figure 4-25: PRBS Implementation with inverters between stages

The data is correctly latched with the addition of the inverters between the flop stages. The added delay in the logic is enough for the data to be correctly captured by the flip-flops.

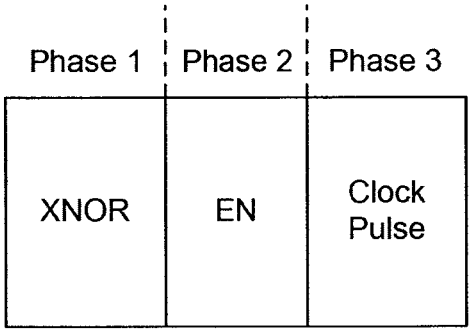

Figure 4-26: 3 Phases in asserting and de-asserting Clock Pulse

The order of events which leads to assertion and de-assertion of clock pulse is shown is shown in Figure 4-26. The inputs to the xnor circuits are based on the latched data and the input data to the flip-flop. When the inputs to the flip-flop circuit toggles, it causes the output of the xnor circuit to be asserted. This causes the enable, "en" signal in the pulse generator, to be asserted if the clock input signal is de-asserted. The clocking signal to storage element is asserted when clock signal rises. The output of the xnor circuit de-asserts when the latched data and the input data are the same. This then leads to the enable, "en" signal in the pulse generator, to be de-asserted. This gates the clocking signal to the storage element.

The long hold-time is due to the de-assertion of the clock pulse from the output of the xnor circuit. The order of events in asserting or de-asserting clock pulse is shown in Figure 4-26. Hold-time can be greatly reduced if the pulse generation is de-asserted with a different method, Figure 5-1. This is explained in section 5.1.

4.5 *PRBS Power Consumption*

PRBS generator implemented with CoD flip-flop has higher power consumption than implementations with standard master-Slave flip-flop and clock gated flip-flop due to high activity factor of the per bit activity factor of the 8-bit PRBS generator. Some of the power advantage over the DTLA flip-flop is lost due to the addition of the inverters between flip-flop stages which adds to the PRBS generator power consumption.

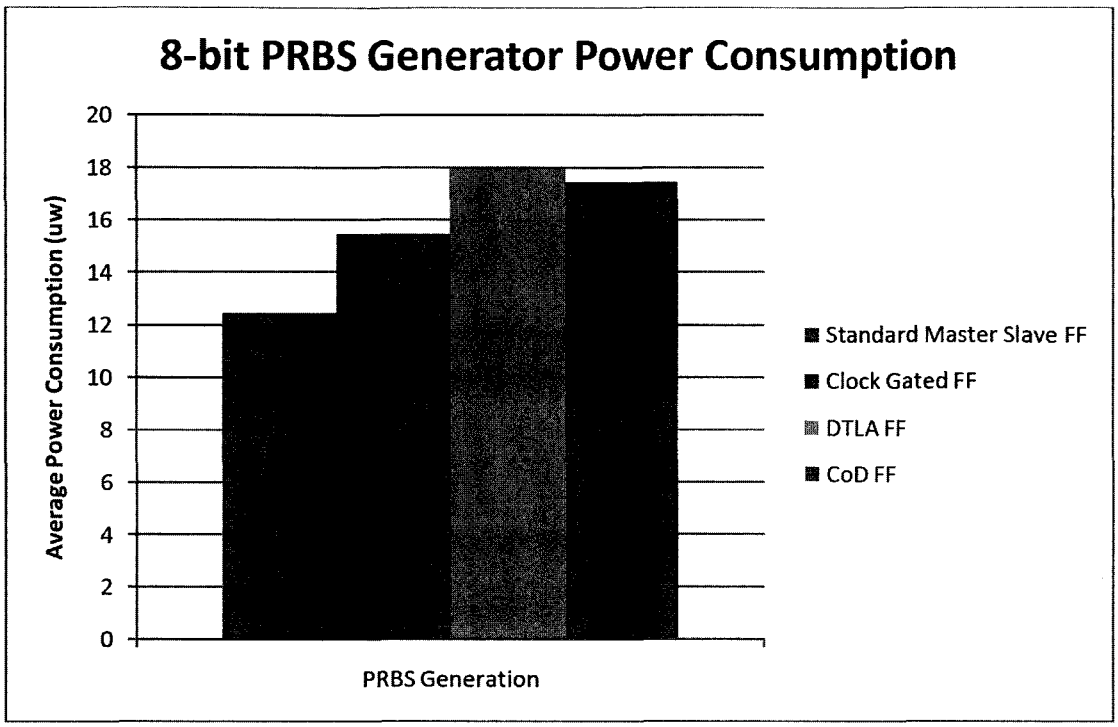

Figure 4-27: 8-bit PRBS Power Consumption comparison

4.6 *Data Bus Power Consumption*

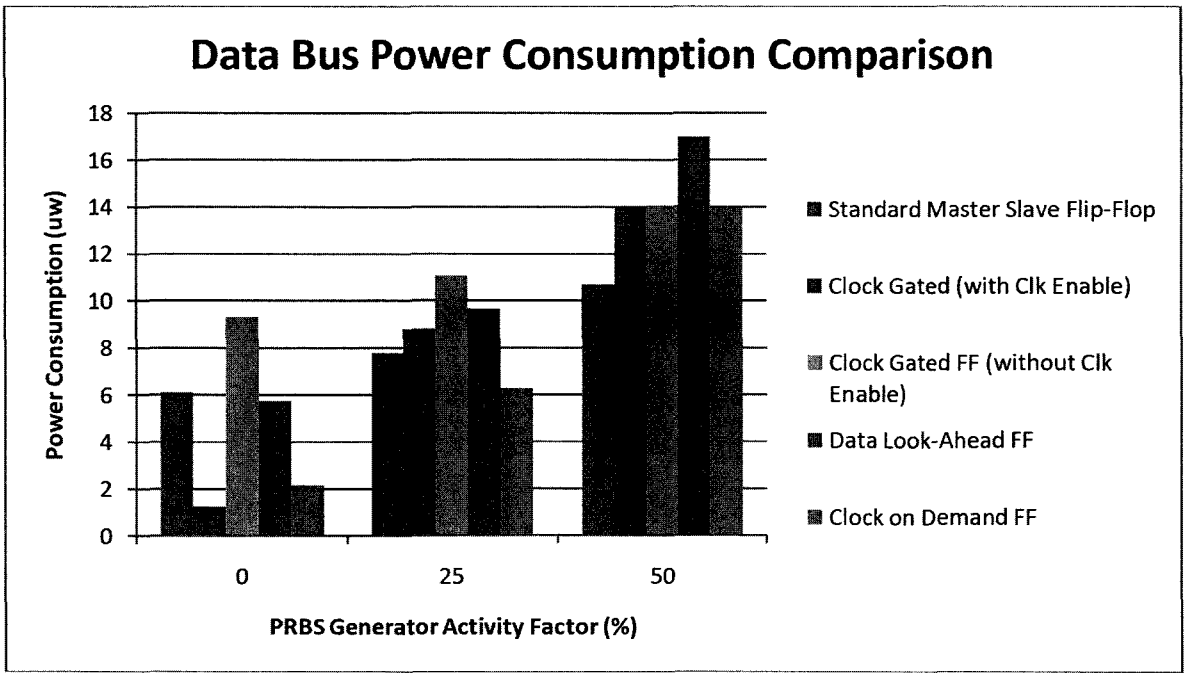

Figure 4-28: Data Bus Power Consumption Comparison

Data bus implemented with CoD flip-flop has lower power consumption than data bus implemented with DTLA flip-flop due to the gating of the generation of the clock pulse signal at an earlier stage. Power consumption is lower than standard master-slave flip-flop for PRBS generation activity factor of 0% and 25%. For these activity factors, the average per bit activity factors are 0% and 12.2% which are lower than the 12.95% which is the data activity factor for which both the flip-flops consume equal amounts of power. The non-linearity of the power consumption for data-bus implemented with clock gated ff is explained in section 4.2.

Chapter 5 **Proposed flip-flops**

In this section, three flip-flops that reduce power consumption and fix the shortcomings in earlier circuits are proposed. The first two flip-flops, clock on demand flip-flop with shared pulse generator and clock gating on demand flip-flop, are based on the CoD flip-flop. The two proposed flip-flops reduce the CoD flip-flop hold time and increase the race immunity of the CoD flip-flop. They also reduce power consumption under certain conditions when compared to the CoD flip-flop. The third flip-flop, clock-gated DTLA flip-flop, is based on the DTLA flip-flop. Clock-gated DTLA flip-flop improves on the DTLA flip-flop through the addition of clock gating functionality to the pulse generator without greatly increasing the power consumption or the area.

5.1 *CoD Pulse generator with hold-fix*

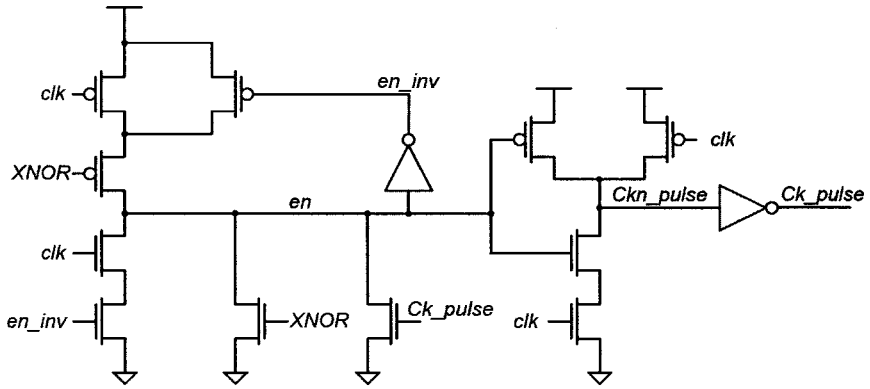

Figure 5-1: CoD Pulse Generation with Hold Fix Circuit

Reducing hold time will improve CoD flip-flop's race immunity and eliminate the need for adding inverters between flip-flop stages. The mechanism that leads to this issue is explained in section 4.4. The "en" signal is currently de-asserted when "XNOR" signal asserts. To reduce hold time, the signal "en" is de-asserted right after the clock pulse is generated. The generated pulse itself is used to de-assert the "en" signal, which greatly reduces the CoD flip-flop's hold time. In order to use the clock pulse to de-assert the "en" signal, an nmos transistor is added to the pull-down circuit of the "en" generation circuit. The width of the inverter

generating the clock pulse signal is increased to drive the additional transistor and widen the clock pulse signal.

5.1.1 Timing Characteristics Comparison

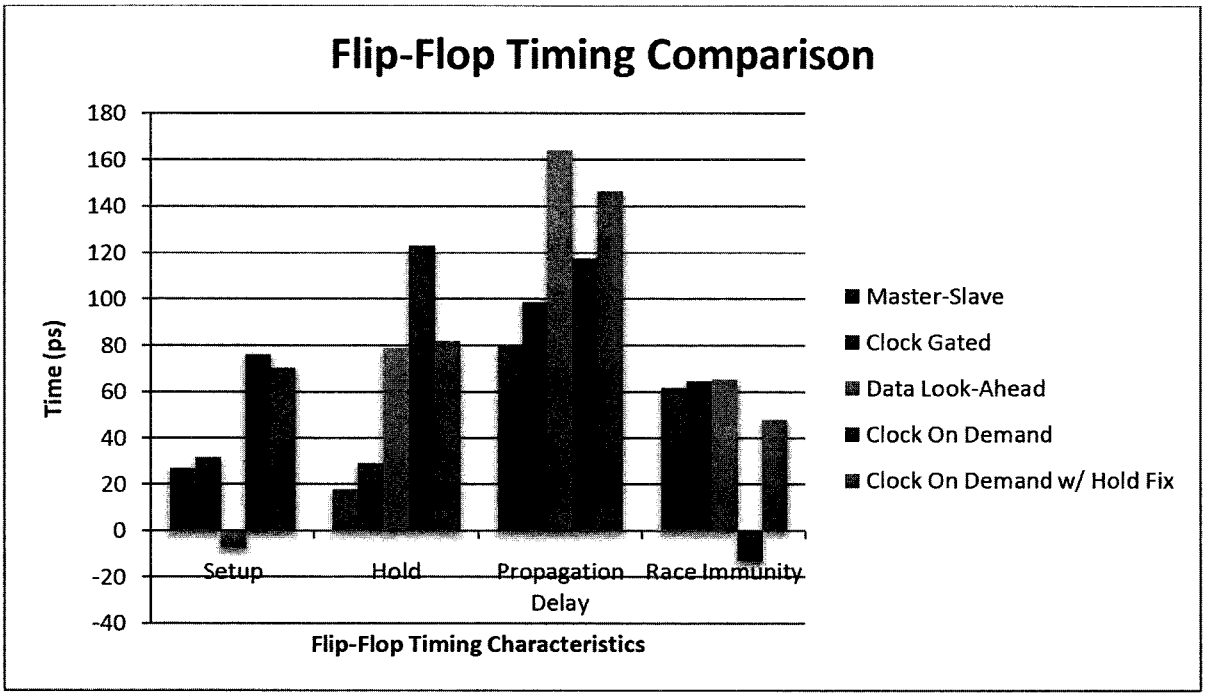

Figure 5-2: flip-flop Timing Comparison

The worst-case setup time was reduced by 5.9ps. This is attributed to the increased clock-pulse drive strength. Hold time is reduced by 41.3ps due to the hold fix addition to the circuit. Propagation delay is increased by 28.9ps due to the reduced clock pulse width. Worst-case race immunity is increased by 60.7 ps due to decreased hold time and increased propagation delay.

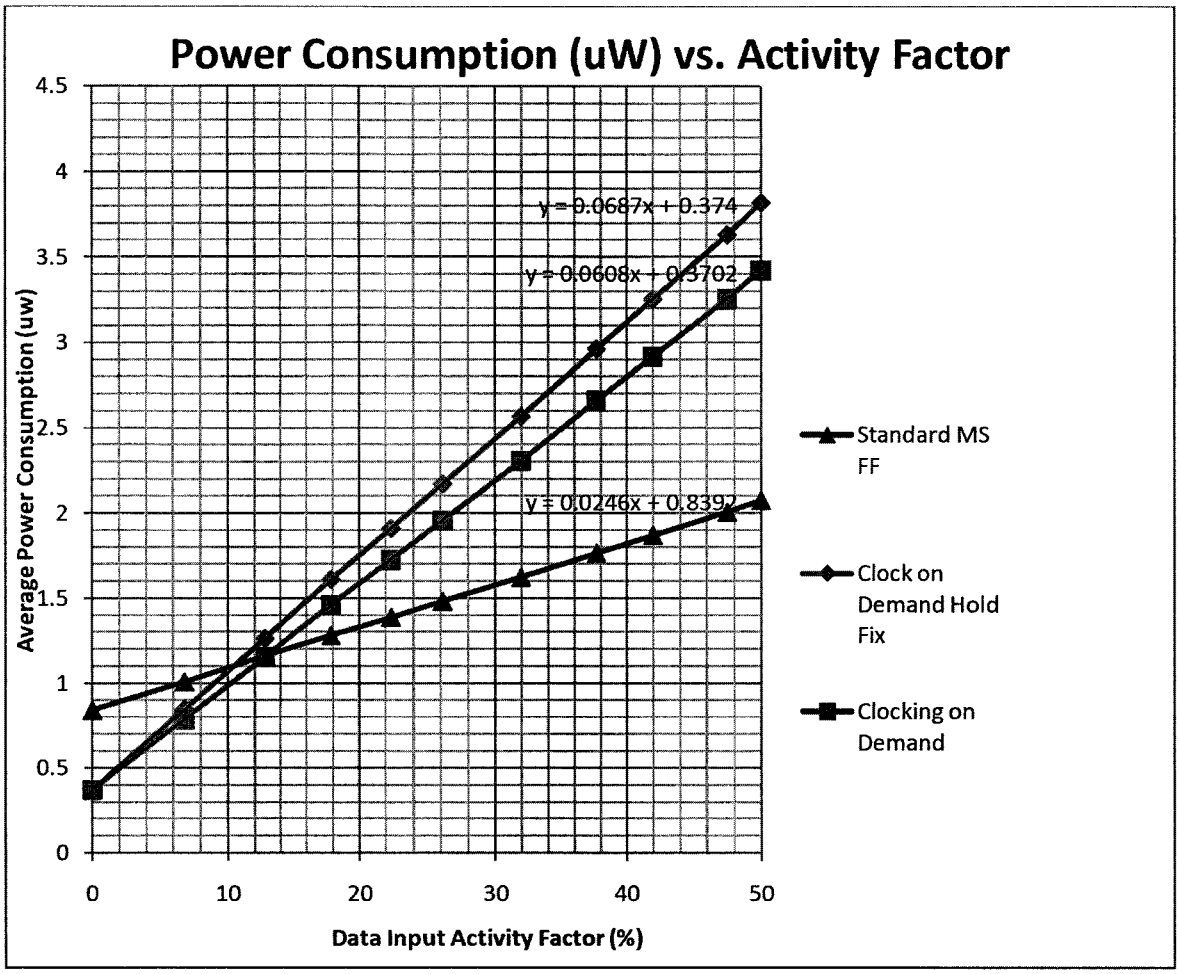

Figure 5-3: flip-flop power consumption comparison

Fixing the hold time by de-asserting "en" signal when clock pulse is asserted, adds to the pulse generator power consumption, shown in Figure 5-3. This causes the clock on demand flip-flop to consume more power than the standard master-slave flip-flop when the data input activity factor is higher than 10.5% as opposed to 12.95% without the hold-fix circuit. This fix is necessary in order to improve race immunity and preserve flip-flop functionality.

5.2 *Shared Pulse Generator in CoD flip-flop*

The pulse generator in CoD flip-flop consumes lower power for low data activity factor when compared to standard master slave flip-flop. The power consumption for the CoD flip-flop can be lowered for the flip-flops by sharing the pulse generator. This reduces the clock

input capacitance and the total overall capacitance of the implementation. Due to the sharing of the pulse generator, a pulse needs to be generated if the data input to any one of the n storage element toggles. Thus this flip-flop style should not be implemented in designs with mismatched temporal activity as it will reduce the power savings, as shown in Figure 5-15.

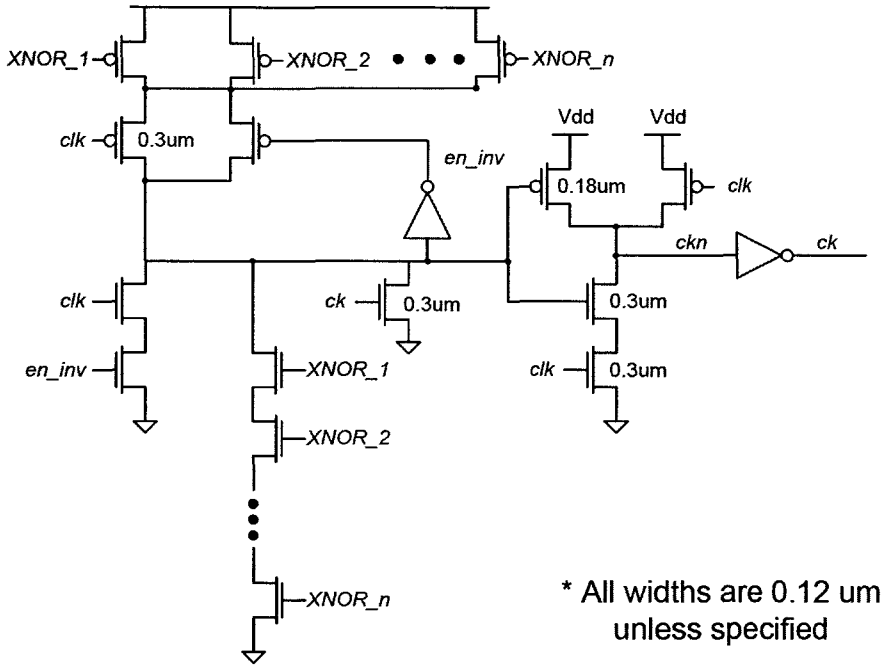

Figure 5-4: Shared Pulse Generator Implementation

5.2.1 CoD flip-flop with Shared Pulse Generator - Power Consumption

The 16-bit counter implemented with CoD flip-flop with shared pulse generator has much lower power consumption than other implementations for 0% counter increment rate. However the power savings are quickly diminished for increasing counter activity factor. Power consumption is still lower than the other implementation for 50% counter increment rate. At the counter increment rate of 100%, the implementation consumes slightly more power than 16-bit counter implemented with CoD flip-flop. The power consumption increase is due to increased clock pulse generator activity factor exceeds the power savings brought by amortization of the pulse generator.

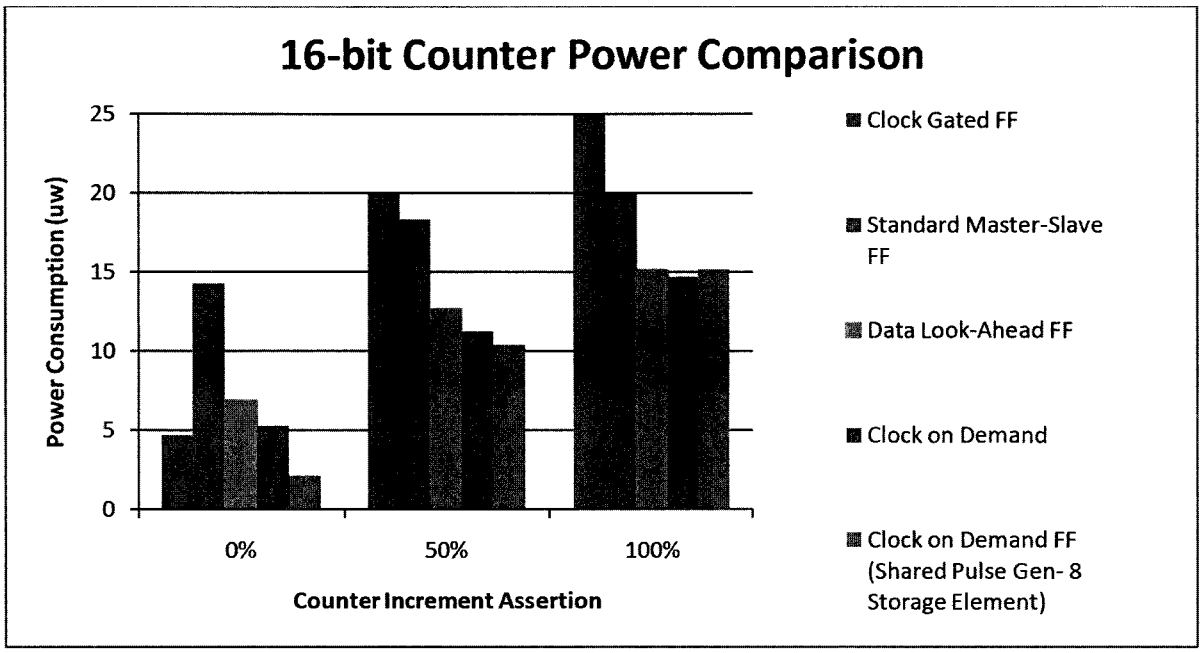

Figure 5-5: 16-bit Counter Power Consumption Comparison

At 0% counter increment rate, the shared pulse generator helps reduce the total power consumption by 59% over the 16-bit counter implemented with regular clock on demand flip-flop. The power savings is reduced as the counter increment rate increases. This is due to a change in data input to any of the storage element resulting in a clock pulse generated to all the storage elements. As the counter width increases, the power consumption will be lowered due to lower overall activity factor.

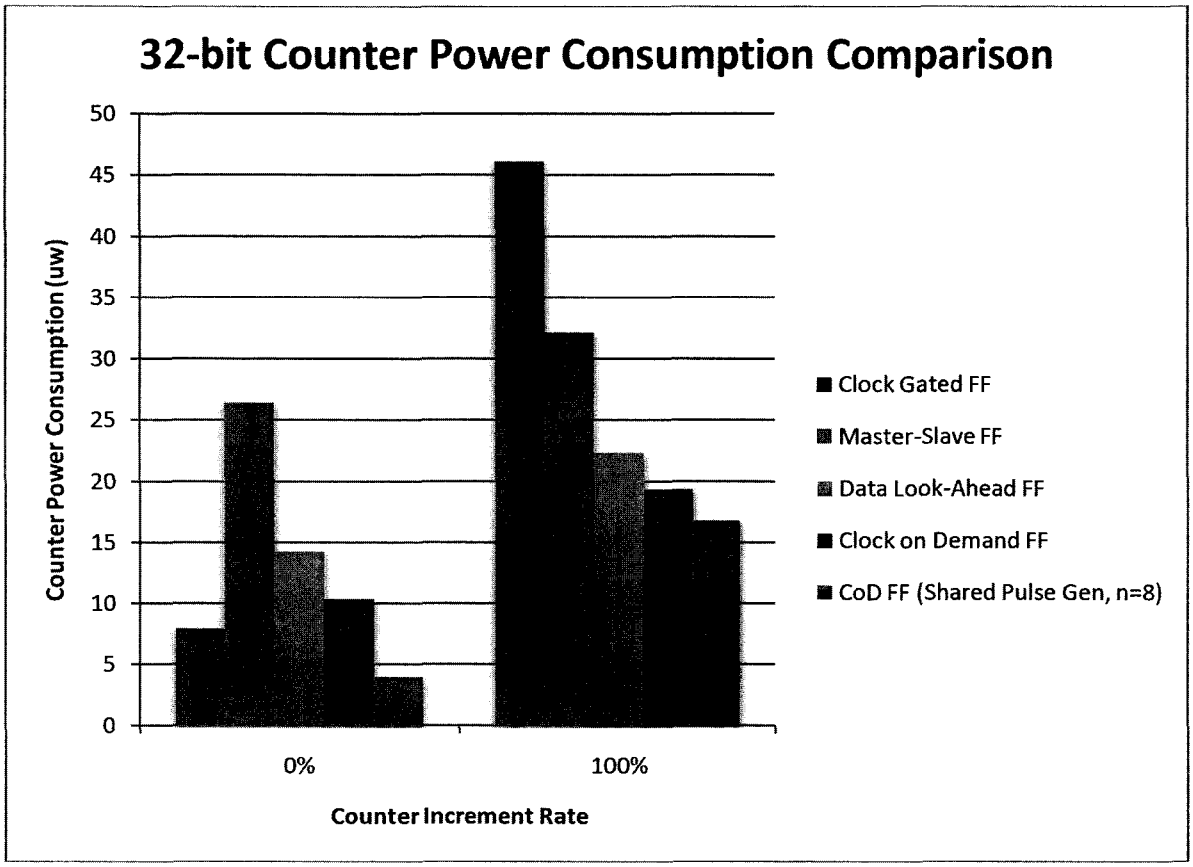

Figure 5-6: 32-bit Power Consumption Comparison

The power consumption trend continues with 32-bit counter simulation. 32-bit counter implemented with CoD flip-flops with shared clock pulse generator displays much lower power consumption than the other implementations. As the counter increment rate increases, the power saving diminishes over counters implemented using data look-ahead flip-flop and regular CoD flip-flop. This is due to the increased average activity factor of the generated clock pulse. Power consumption is lower than the other implementations for all counter increment rates, due to the reduced activity factor of the flip-flops used in the upper bits of the counter.

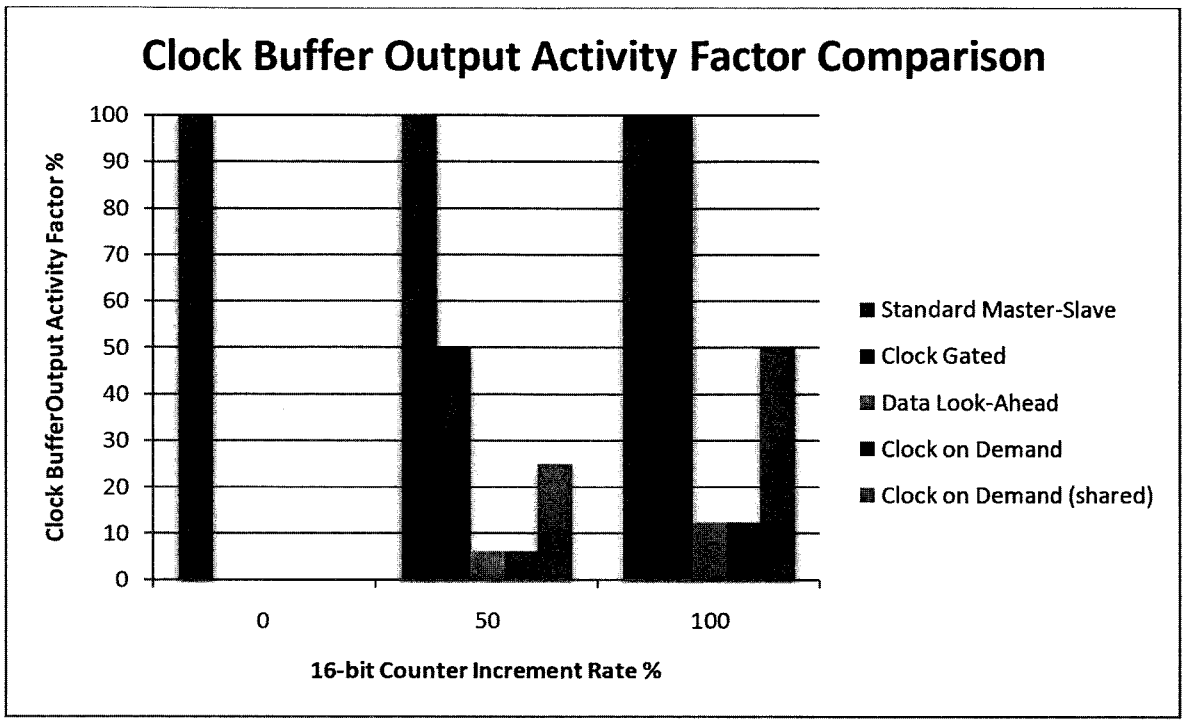

Figure 5-7: Clock Buffer Output activity factor in 32-bit Counter implementations

The CoD flip-flop shared clock pulse generator's activity factor increases quickly with increasing counter increment rate. This is due to the shared pulse generator toggling every cycle due to the changing least significant bit in the counter. Clock on demand flip-flop (shared pulse generator) exhibits low power consumption, for counter increment rate of 50%, even though it has a higher clock buffer output activity factor due to amortization of the pulse generator. This flip-flop style is more suited for circuits with low activity factor and similar temporal activity. Adding a clock enable/disable signal will reduce clock pulse generator's activity factor and will thus reduce power consumption in the clock on demand flip-flop for specific logic implementations. These logic implementations will however have to support "Clock Enable" signal generation logic and a toggling data-bus while the "Clock Enable" signal is de-asserted.

5.3 *Clock Gating on Demand flip-flop Design*

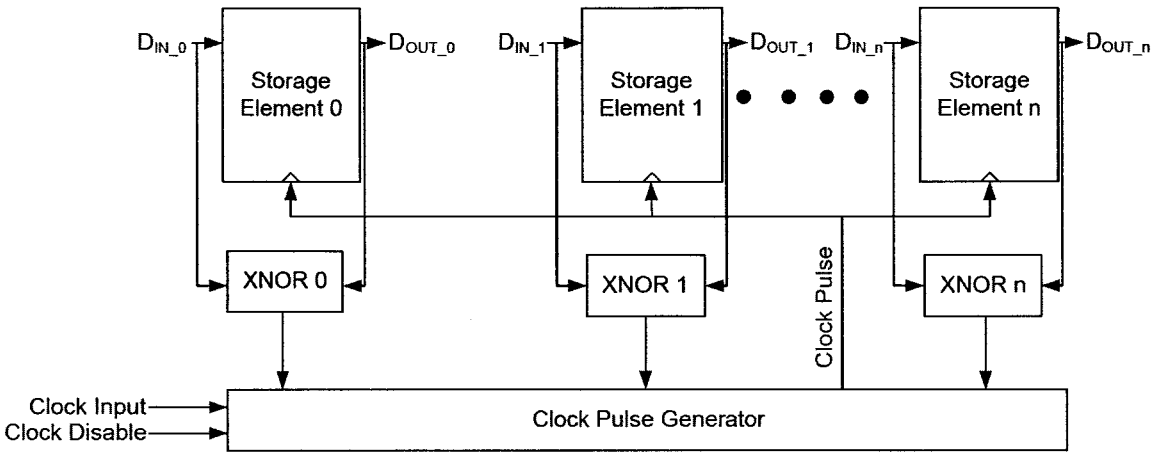

Figure 5-8: Clock Gating on Demand Overview

Clock Gating on Demand (CGoD) flip-flop improves on CoD flip-flop by fixing the hold-time of the CoD flip-flop, sharing the pulse generator and gating the pulse generator additionally when the "Clock Disable" signal is asserted. When the "Clock Disable" signal is asserted the XNOR signals are not looked at and regardless of the toggling of the data input signals, the clock pulse signal is not generated.

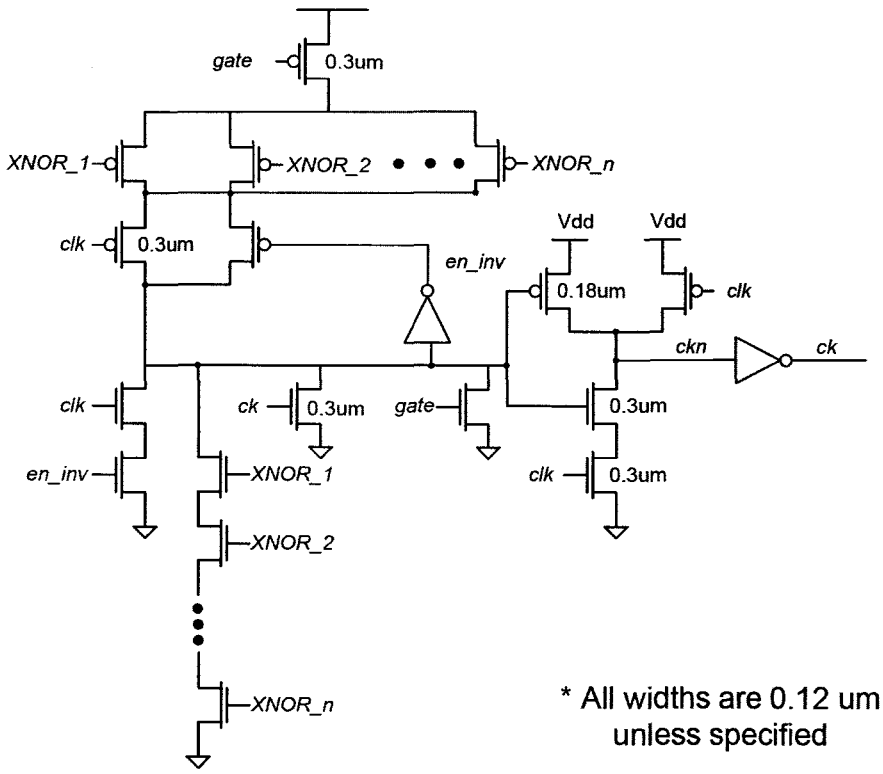

Figure 5-9: CGoD flip-flop Pulse Generator

In order to support clock gating, two additional transistors are added to the pulse generator with hold-fix. When "clk_disable" is asserted, the data input-output comparator results are ignored. When the "clk_disable" signal is de-asserted, the circuit will generate a clock pulse either the rising edge of the clock or the falling edge of the clock, based on the point of reference, when the data input to at least one of the n storage elements have toggled.

5.3.1 CGoD flip-flop - 16-bit Counter Power Consumption

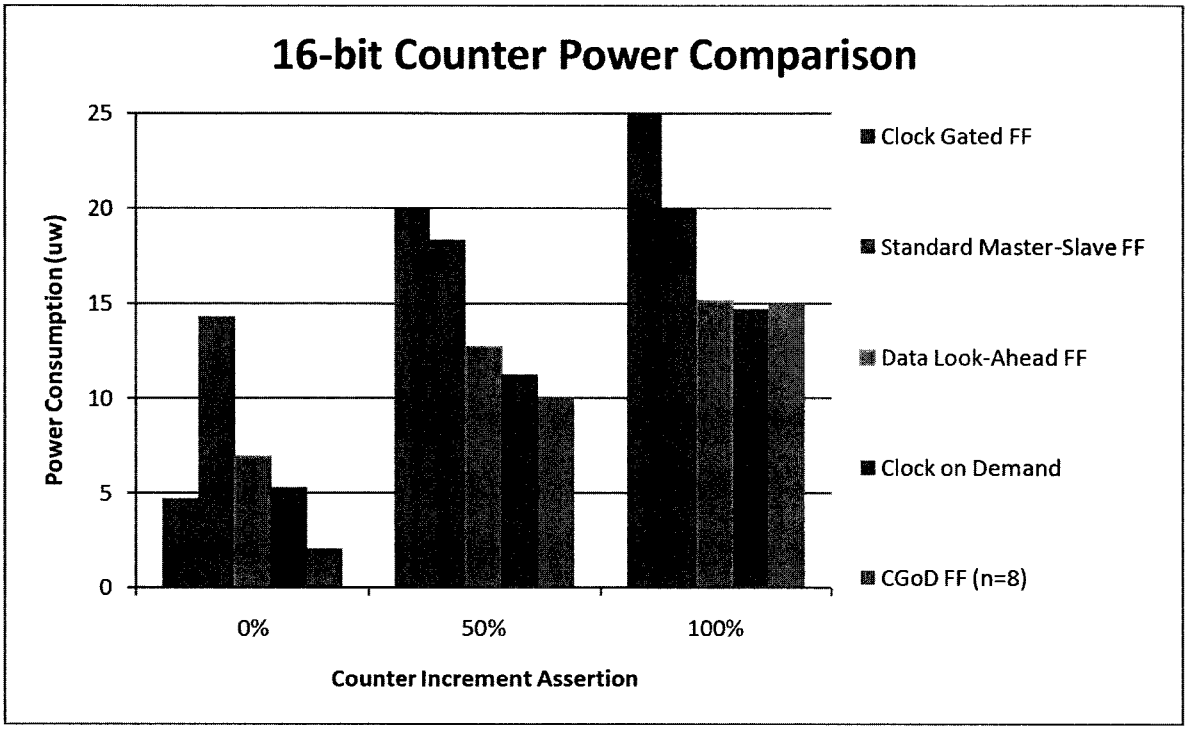

Figure 5-10: 16-bit Counter Power Consumption Comparison

Power consumption over the other implementations is a lot less for lower activity factors. Power consumption of the counter implementation is very similar to the clock on demand flip-flop with shared pulse generator. This is because the clock gating has no effect since the data inputs are not toggling whenever the clock is gated. Thus for all subsequent simulations, the CGoD flip-flop is not expected provide power savings over the clock on demand flip-flop with shared pulse generator due to the lack of a test bed which toggles data when clock is gated.

5.3.2 CGoD flip-flop - 32-bit Counter Power Consumption

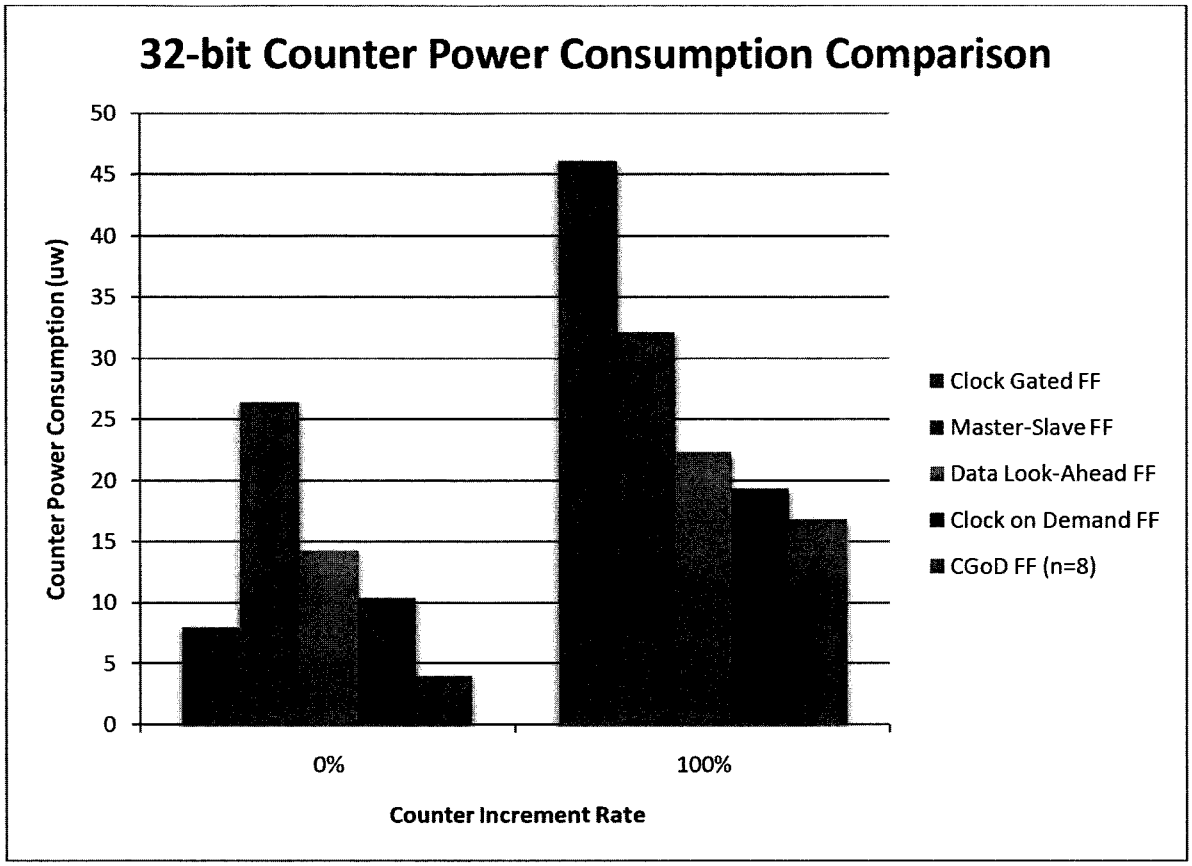

Figure 5-11: 32-bit Counter Power Consumption Comparison

The power consumption trend continues with 32-bit power consumption simulation. The 32-bit counter implemented with CGoD flip-flop is markedly lower than other implementations but it is similar to the CoD flip-flop with shared pulse generator.

5.3.3 CGoD flip-flop - PRBS Generator Power Consumption

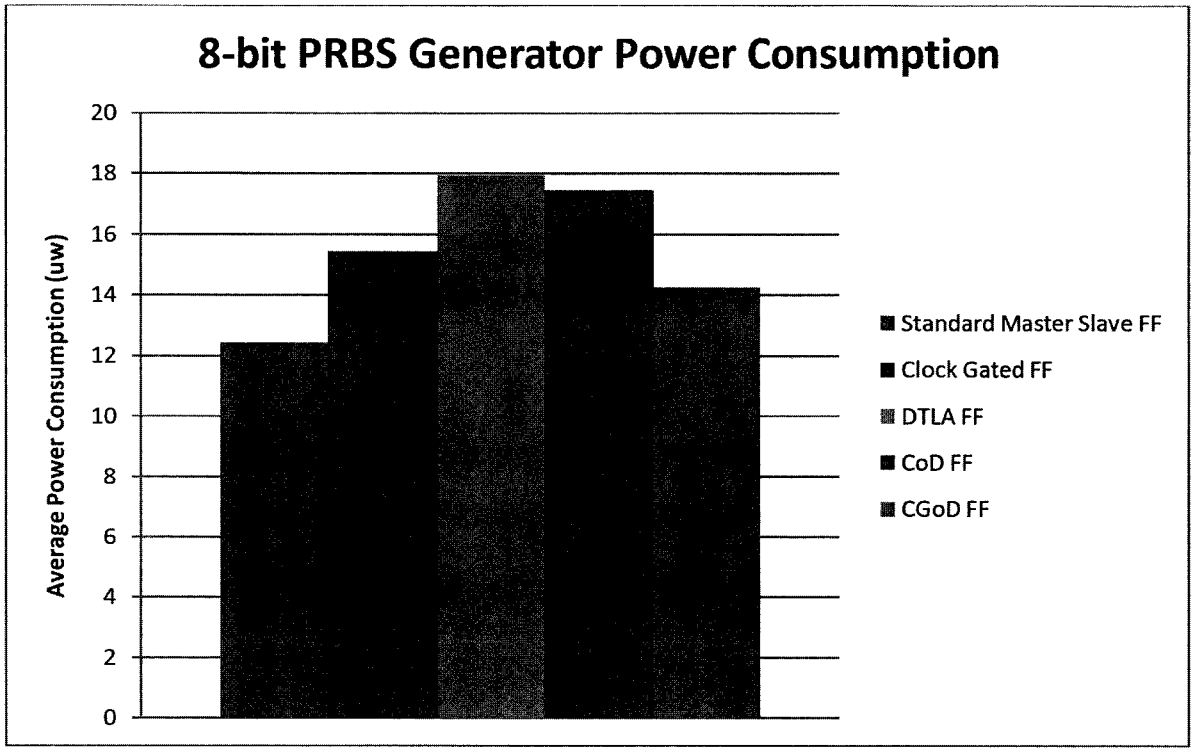

Figure 5-12: 8-bit PRBS Generator Power Consumption Comparison

Power consumption of the PRBS generator implemented with CGoD flip-flop is lower than most other implementations with the exception of the standard master slave flip-flop implementation. The power savings is due to the amortization of the power consumed by pulse generator. In any given clock cycle one or more bits are toggling, due to the nature of the PRBS pattern generator which is built to generate new patterns every clock cycle. Thus, more than one pulse generator is generating a clock pulse every cycle in the PRBS generator implemented in CoD flip-flop. Thus this power consumption is reduced by sharing the pulse generator between multiple flip-flops. The power consumption of the PRBS generator implemented with CGoD flip-flop is higher than the one implemented with standard master-slave flip-flop due to non-similar temporal activity. The power savings over all implementations will only be achieved if the shared storage elements have similar temporal activity factor. The data patterns A and B shown in Figure 5-13 and Figure 5-14 have the same per-bit activity factor. In both the data patterns, A & B, each of the bits comprising the eight bit data bus toggle once in the eight

clock cycle window. In data pattern A, each of the bits toggle at different clock cycle but in data pattern B, the bits toggle in the same clock cycle. Power consumption for both data pattern should be the same if data bus was implemented with CoD flip-flop. Power consumption for data pattern A should be higher and power consumption for data pattern B should be a lot lower if data bus was implemented with CGoD flip-flop. The difference is due to a clock pulse being generated for every cycle for data pattern A and one clock pulse generated in eight cycles for data pattern B. The data bus in a network processor resembles data pattern B more closely than data pattern A.

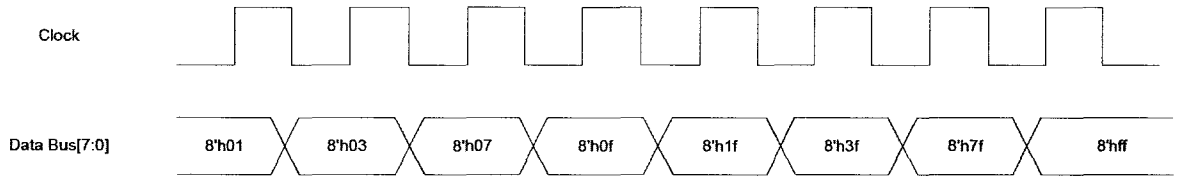

Figure 5-13: Data Pattern A with different temporal activity

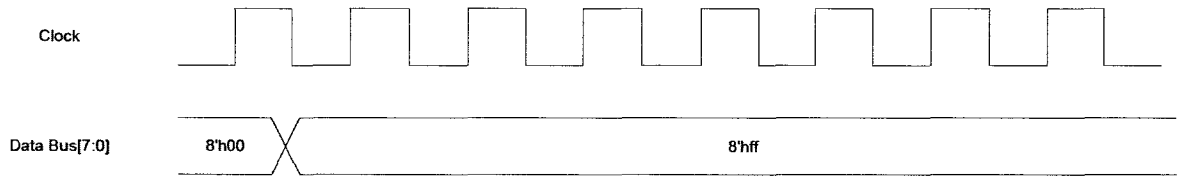

Figure 5-14: Data Pattern B with same temporal activity

This trend can be shown by simulating two 8-bit data-buses with different temporal activity factors but the same per bit activity factor. Each of the data-bus will be implemented with CoD flip-flop and CoD flip-flop with shared pulse generator. Two patterns are driven into each of the two DUTs with the simulation lasting eight clock cycles. All the storage elements in the DUTs are initialized to 0 for both the simulations.

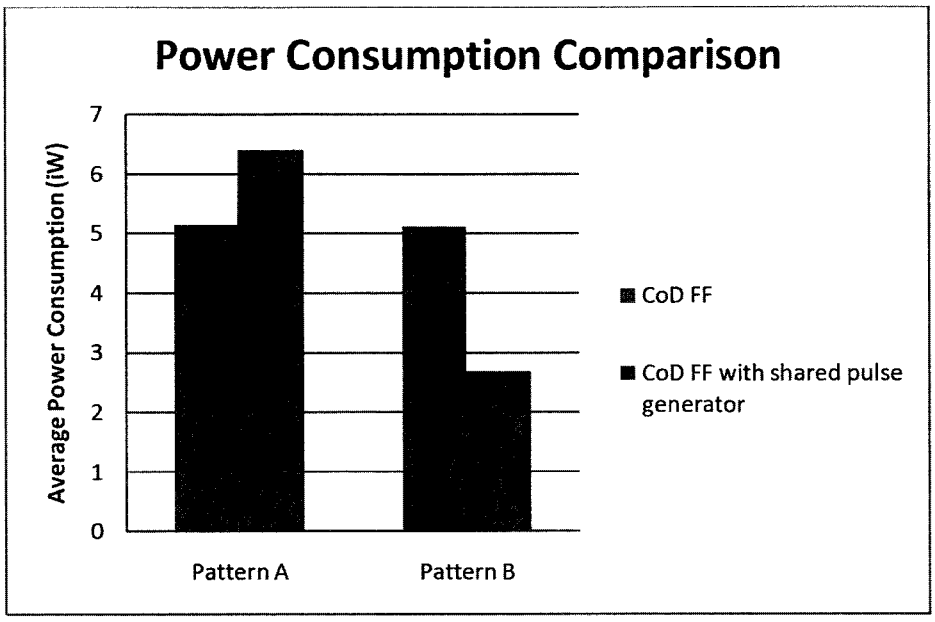

Figure 5-15: Effect of Temporal Activity on Power Consumption

The simulations results illustrate that temporal activity greatly affects the power consumption of CoD flip-flop with shared pulse generator and CGoD. For different patterns of input data, which have the same per bit activity factor, the power consumption for the 8-bit data bus implemented with CoD flip-flop with shared pulse generator differs by 3.73uw. The power consumed for data pattern A, different temporal activity, is 139% higher than the power consumed for data pattern B, same temporal activity.

5.3.4 CGoD flip-flop - Data-Bus Power Consumption

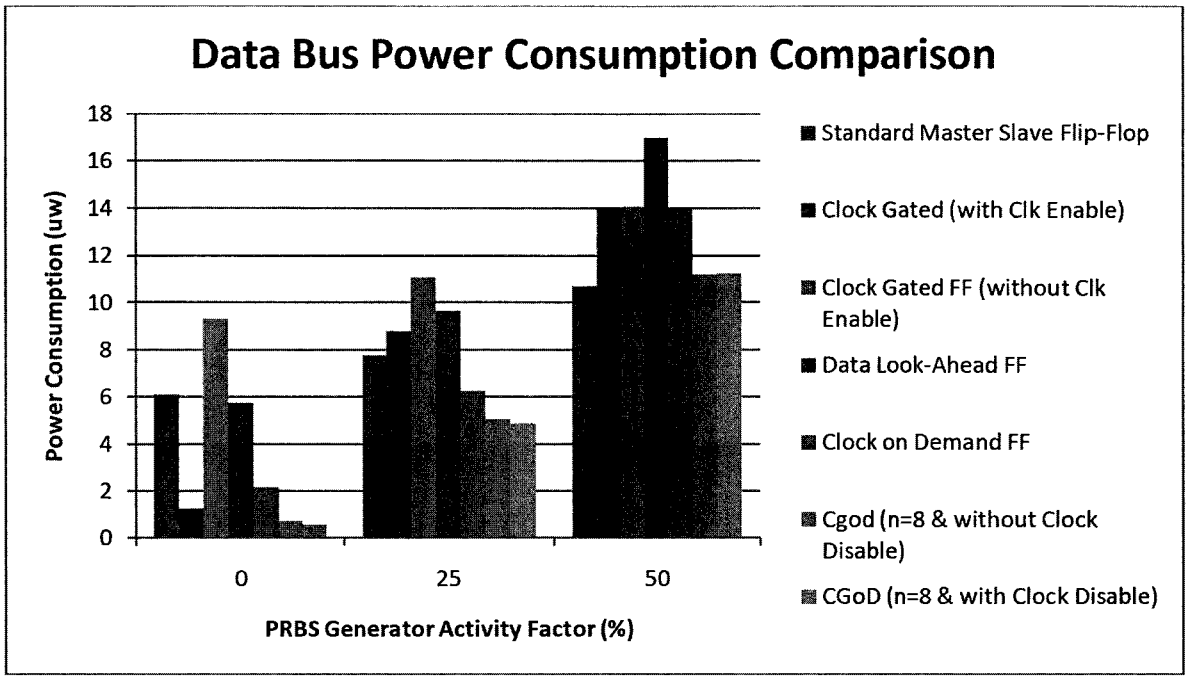

Figure 5-16: Data Bus Power Consumption

Power consumption of the 8-bit data bus implemented with CGoD is much lower than other implementations for low data input activity factors. The ability to gate the clock does not provide huge power savings due to the data inputs not toggling when the "Clock Disable" signal is asserted.

5.3.5 Design: Clock-gated Data Look-Ahead flip-flop

The DTLA flip-flop analyzed in section 7, has a disadvantage when compared to CoD flip-flop due to the power consumption overhead of the pulse generator. This is because the clock pulse signal is generated regardless of the data input toggling status. In the CoD flip-flop, the clock pulse is only generated when the data input has toggled and is different from the stored value. Power consumption can be greatly reduced by gating the clock pulse generator through a "Clock Enable" signal.

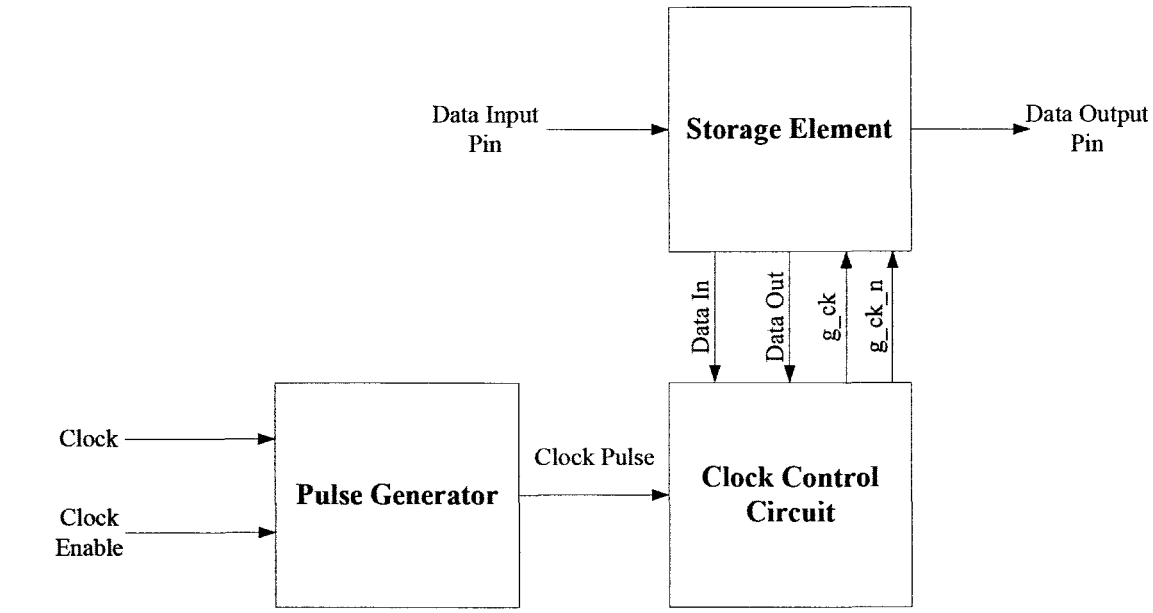

Figure 5-17: Clock-gated DTLA flip-flop overview

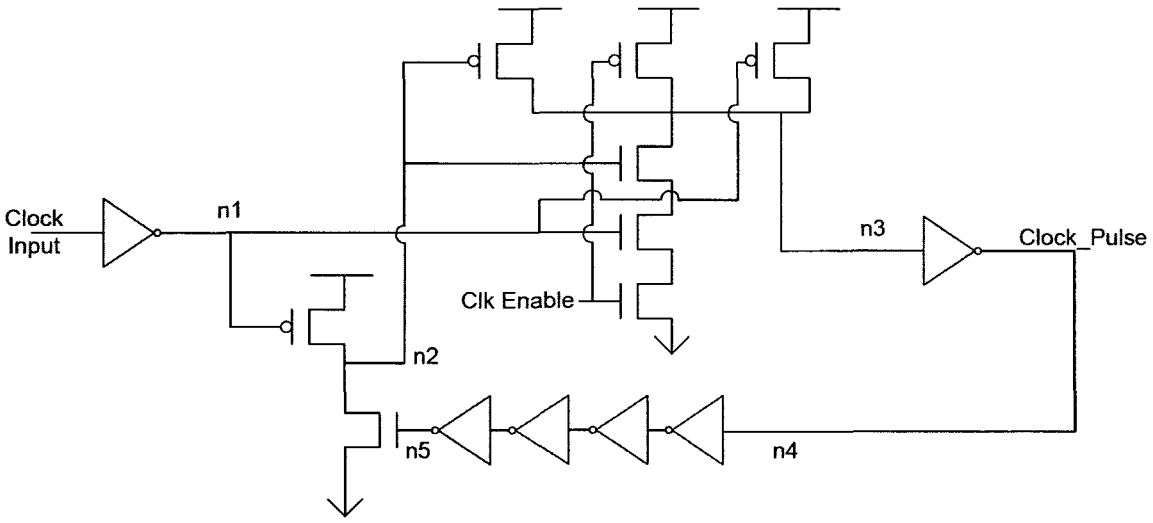

Figure 5-18: Clock-gated DTLA flip-flop Pulse Generation

5.3.6 Clock Gate DTLA flip-flop - 16-bit Counter Power Consumption

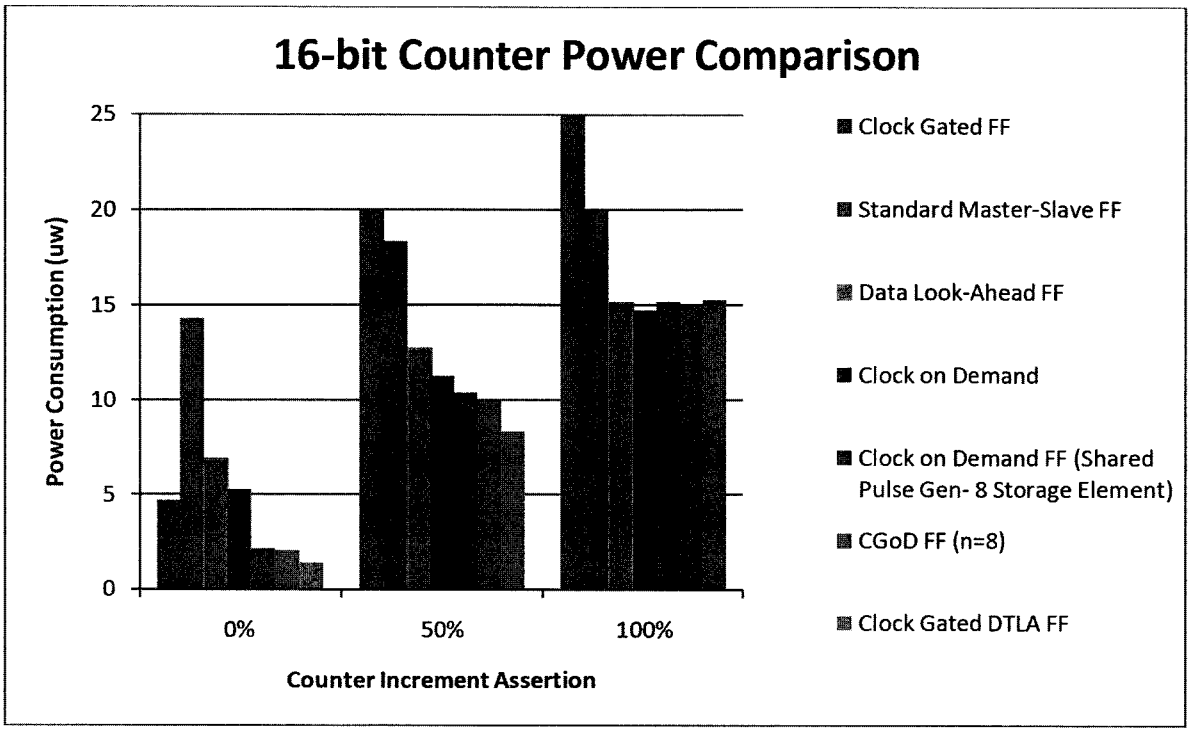

Figure 5-19: 16-bit Counter Power Consumption Comparison

16-bit counter implemented with clock-gated DTLA flip-flop has the lowest power consumption among all implementations when the counter increment rate is 0% or 50%. The power consumption when the counter increment rate is 100% is comparable to other flip-flops except for standard master-slave flip-flop and clock-gated flip-flop implementations. Power consumption is much lower than these two implementations, 23.765% lower than 16-bit counter implemented with standard master-slave flip-flop and 32.9% lower than counter implemented with clock-gated flip-flop. The power savings can be attributed to reduction in pulse generation power consumption through the use of clock gating. Hence at 100% counter increment rate, the clock gating power reduction advantage is lost, since the clock is not gated for the duration of the simulation.

5.3.7 Clock-gated DTLA flip-flop - 32-bit Counter Power Consumption

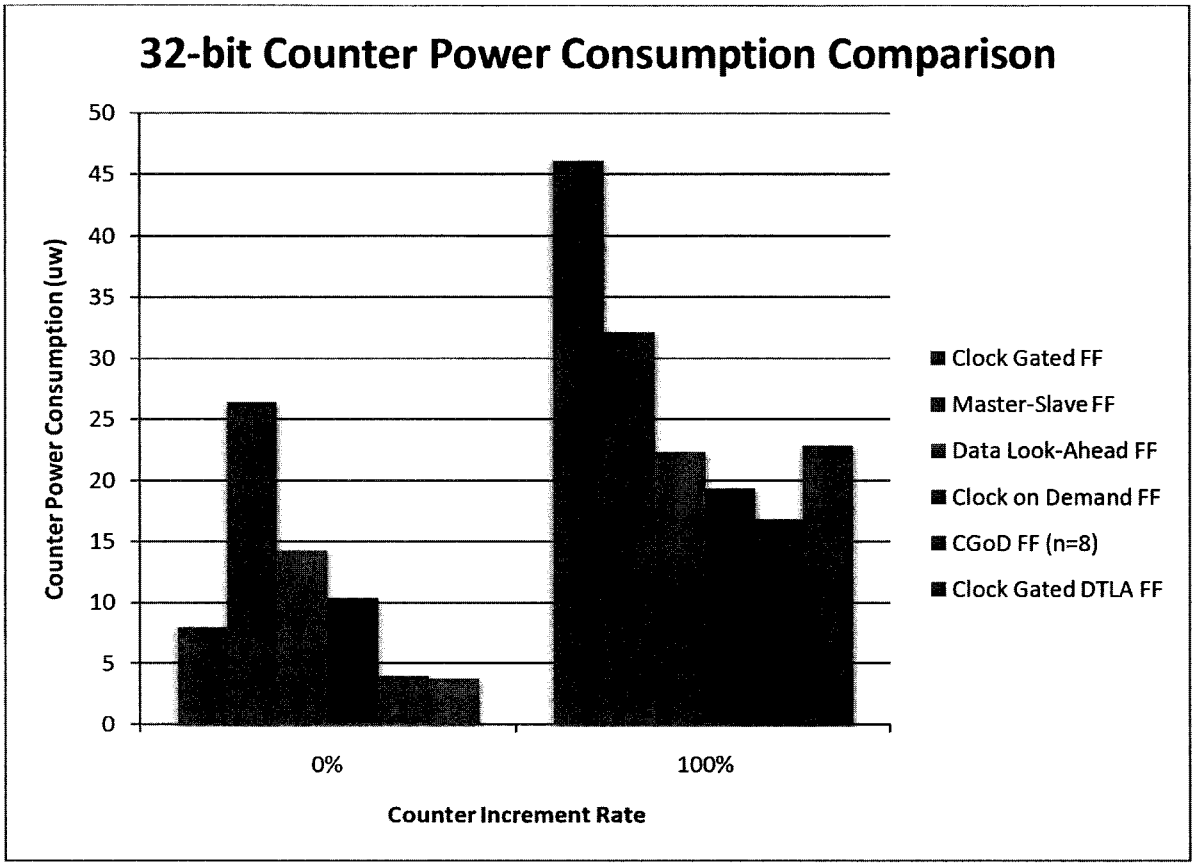

Figure 5-20: 32-bit Counter Power Consumption Comparison

Counter implemented with clock-gated DTLA flip-flop continues to exhibit low power consumption when counter increment rate is 0%. For counter increment rate of 100%, the power saving advantage is lost due to the power consumption overhead in generating a clock pulse signal for the storage elements that store the upper 16 bits of the counter which do not increment frequently. This is because of the sharing of the clock enable signal between the two pulse generators. A finer granularity of clock enable signal generation will reduce power consumption significantly.

5.3.8 Clock-gated DTLA flip-flop - PRBS Generator Power Consumption

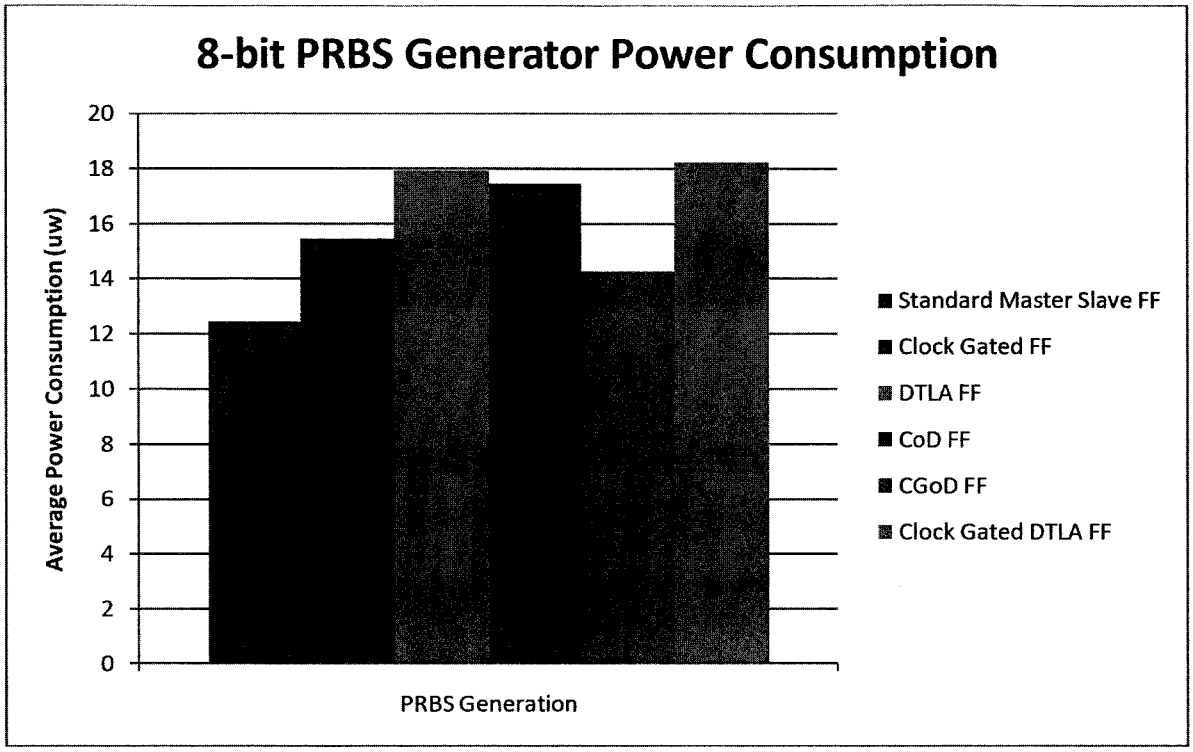

Figure 5-21: 8-bit PRBS Generator Power Consumption Comparison

PRBS generator implemented with clock-gated DTLA flip-flop exhibits high power consumption since the power saving advantages of clock gating through "Clock Disable" and data look-ahead are lost.

5.3.9 Clock-gated DTLA flip-flop - Data-Bus Power Consumption

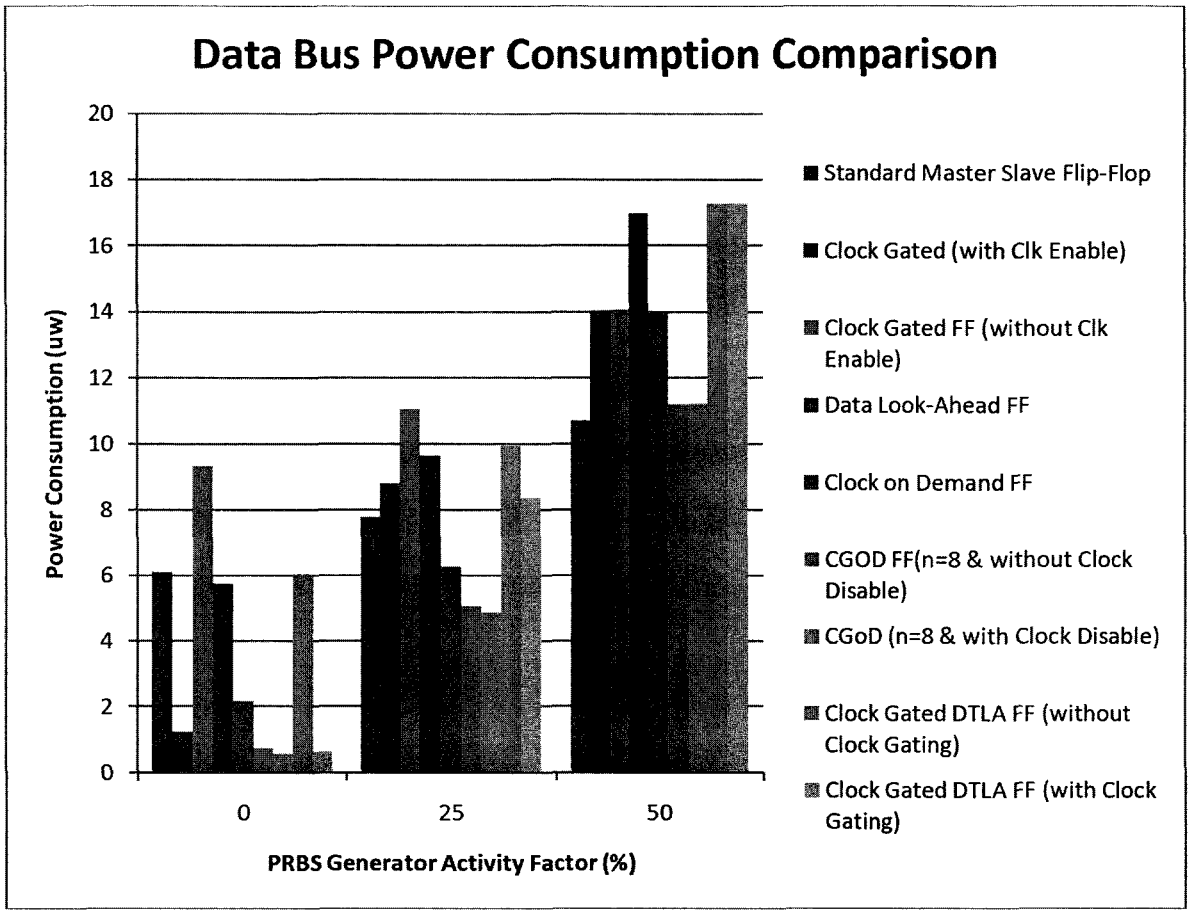

Figure 5-22: Data Bus Power Consumption Comparison

Data-bus implemented with clock-gated DTLA flip-flop has low power consumption when both the clock gating techniques are taken advantage off. While the clock is gated for majority of the duration and the data activity factor is low, the power consumption of the DTLA flip-flop is significantly lower. The power consumption of the DTLA flip-flop jumps significantly when the average data activity factor increases.

5.4 *Flip-flop Usage Recommendation*

5.4.1 Flip-flops to be used in modules with high activity factor

Standard master-slave flip-flop should be used exclusively for flip-flops with high data activity factor and for scenarios where the chance of clock signal being disabled is low. Clock-gated flip-flops should not be used due to high power consumption when the clock signal is enabled. The data input to the flip-flops are usually valid every cycle due to the nature of Ethernet specification, unless the PCS is running at a much higher frequency than needed to handle data at line rate, which leads to a lot idle cycles as a result of over-clocking. Reducing power consumption while a port is disabled or a module is disabled should not be done through clocking gating at the flip-flop level granularity. Clock root gating should be done for these scenarios to reduce power consumption.

5.4.2 Flip-flops to be used in modules with low per-bit data activity factor

Modules that have a low per-bit data input activity factor, and a low clock enable assertion rate but not similar temporal activity, should use clock-gated DTLA FFs. CoD flip-flop's with shared pulse generator should be used for low per-bit data activity factor with similar temporal activity factor, but in the case of high clock enable assertion rate or when clock enable assertion logic is not present. In the case of presence of clock enable logic and low clock enable assertion rate, the CGoD ff should be used.

5.4.3 Flip-flops to be used in counters

Small counters in the presence of clock enable logic should be implemented with clock-gated DTLA FFs. For all other counters should be implemented with CoD flip-flop with shared pulse generator should be utilized.

5.5 *Clock-gated DTLA flip-flop Pulse Generator Layout*

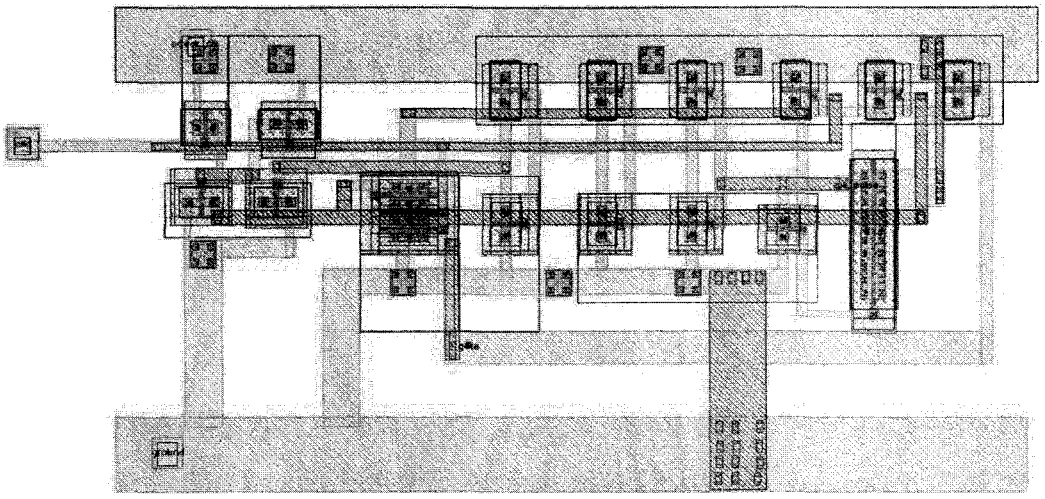

Figure 5-23: Clock-gated DTLA flip-flop Layout

A layout, Figure 5-23, of the pulse generator circuit in the clock-gated DTLA flip-flop was created using the 65nM IBM-ST kit. The main purpose of this exercise is to prove that the circuit functionally operates after layout. DRC and LVS were run to ensure that the pulse generator layout was correct. Layout extraction was done with extraction method set to 'RCcmax' for simulation.

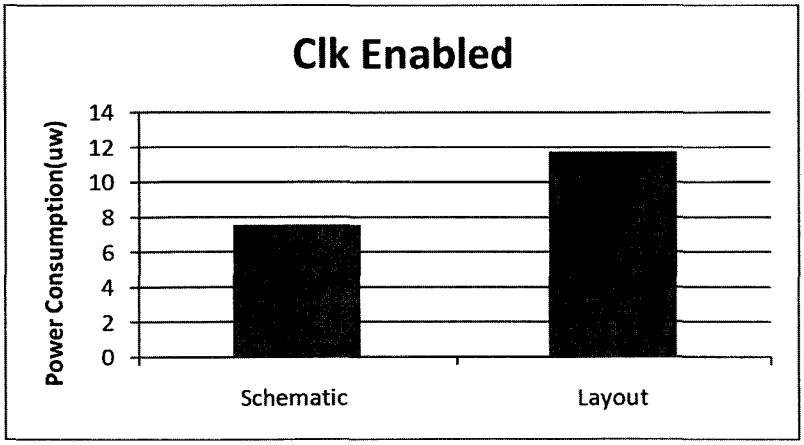

Figure 5-24: Clock-gated DTLA Power Consumption Comparison (Clk Enabled)

Power consumption obtained from the post layout simulation was 4.2 uw higher than the schematic simulation when the clock is enabled. The power consumption can be attributed to higher internal capacitances and using RCcmax extraction option contributed to the power increase. A better layout of the pulse generator would have reduced the power consumption increase.

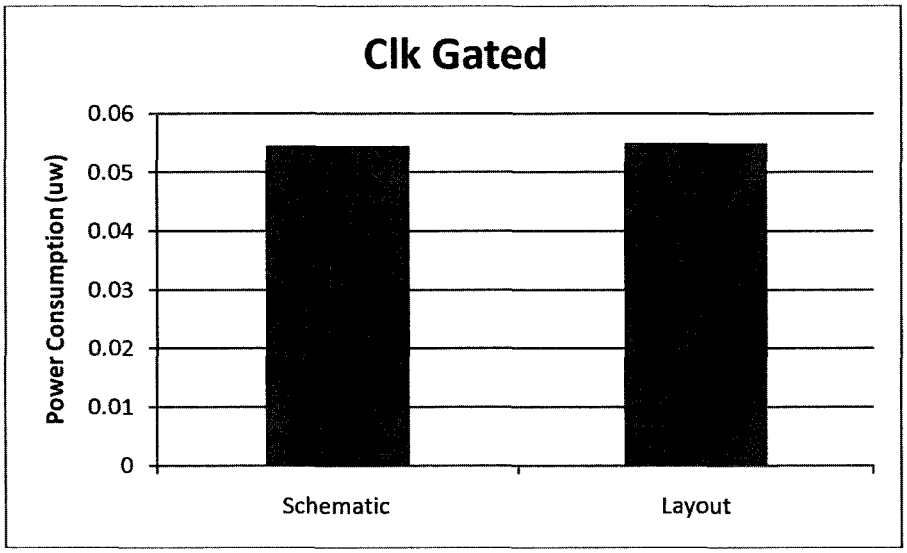

Figure 5-25: Clock-gated DTLA Power Consumption Comparison (Clk Disabled)

Power consumption from the post layout simulation matched closely to the schematic simulation when clock is gated. The difference in power consumption is 0.456 nw. The similarity in the power consumption when clock is gated and the disparity in power consumption when clock is not gated implies that the majority of the power savings are through reducing the toggling of the internal signals whenever possible.

Chapter 6 **Conclusions**

Power consumption is one of the most important issues in VLSI design [29] [30]. The increasing integration of different functionalities into a design to form Systems on Chip has caused power consumption to become a crucial factor in determining a feature list. The methods of power reduction looked at in this thesis are flip-flop design and selective flip-flop placement based on activity factor.

In this thesis, the power consumption of four published flip-flop architectures is investigated under various scenarios, Chapter 4. Three flip-flops are proposed with the focus being on reducing the power consumption under various scenarios in Chapter 5. Simulations for multiple test-beds with different stimuli are run to determine the best use of these new flip-flops in reducing overall power consumption of VLSI systems.

6.1 *Summary of Work in this Thesis*

In this thesis, various test-benches are created following the network processor activity factor analysis in order to obtain a better understanding of the impact of flip-flops on the power consumption. This will lead to a better recommendation of the flip-flop choice for various scenarios to reduce total power consumption of the network processor.

A detailed test methodology for building test-beds, running simulations, measuring power consumption, and measuring flip-flop timing characteristics is presented. This will ensure accurate results for the devices being tested.

Four flip-flop architectures, (1) standard master slave flip-flop, (2) clock-gated flip-flop, (3) data transition look-ahead flip-flop, and (4) clock on demand flip-flop, are simulated and their results were analyzed. Based on the analysis of their design and results from the simulation, three new low power flip-flops, (1) clock on demand flip-flop with shared pulse generator, (2) clock gating on demand flip-flop, and (3) clock gated data transition look-ahead flip-flop have been presented. The effects of clock gating and data

input activity factor have been studied. Simulation and measurement results show that clock gating through the comparison of data input and data output, clock gating through clock enable/clock disable signal, and usage of an explicit pulse generator can be combined to optimally reduce power consumption.

Measurement results of the three proposed flip-flops demonstrated that the reduction in system power consumption can be achieved by leveraging low data activity factor or high clock disable assertion through the selective usage of the three flip-flop designs in addition to standard master-slave flip-flop [28]. Finally, flip-flops are recommended based on the power measurement results. Up to 90% of the power consumption can be reduced when clock gated data-transition look-ahead flip-flops are used to implement counters, and clock gating on demand flip-flops are used to implement a data-bus.

Modules with high activity factor should utilize standard master-slave flip-flops. Modules that have a low per-bit data input activity factor, and a low clock enable assertion rate but not similar temporal activity, should use clock-gated DTLA FFs. CoD flip-flop's with shared pulse generator should be used for low per-bit data activity factor with similar temporal activity factor, but in the case of high clock enable assertion rate or when clock enable assertion logic is not present. In the case of presence of clock enable logic and low clock enable assertion rate, the CGoD flip-flop should be used. Counters with clock enable logic should be implemented with clock-gated DTLA FFs. Counters without a clock enable logic should be implemented with CoD flip-flop with shared pulse generator.

6.2 *Thesis Contributions*

The primary contributions of this thesis work are:

1. Created multiple environments for characterizing flip-flop power consumption.
2. Created a methodology for measuring flip-flop power consumption.
3. Proposed three low-power flip-flop architectures, clock on demand flip-flop with shared pulse generator, clock gating on demand flip-flop, clock-gated data transition look-ahead flip-flop.

4. Recommended scenarios for the usage of the flip-flops in order to reduce power consumption of VLSI systems.

6.3 *Future Work*

Several topics relating to complete layout of the proposed flip-flops and the associated parasitic capacitances have not been looked at. The effect of parasitic capacitances and the optimum number of storage elements that can share a pulse generator need to be analyzed. Hence, one of the possible areas where this work could be extended would involve a post layout simulation of all the proposed flip-flops.

The proposed flip-flops presented in this thesis reduce power consumption when the data input activity factor is low. The next step could be to selectively implement these flip-flops in existing designs and compare power consumption results with the original designs. There is a necessity of CAD tools to be developed that can process simulation results and automatically select the optimum flip-flop to use based on the activity factor obtained from the simulations. This would greatly speed up the process of flip-flop selection and optimize a design for power reduction.

References

[1] Cisco Systems, "Cisco Visual Networking Index: Forecast and Methodology, 2009-2014," *Cisco Systems White paper*, pp. 1–2, June 2010.

[2] Cisco Systems, "Cisco CRS-1 Carrier Routing System 16-Slot Line Card Chassis System Description," *Cisco Systems White paper*, Chapter5-pg7, [Aug. 27, 2010]

[3] Cisco Systems, "Cisco CRS-3 Modular Services Card Data Sheet," Cisco Systems Data Sheet, pp. 4, [Aug. 27, 2010]

[4] Cisco Systems, High Availability initiative, "http://www.cisco.com/warp/public/cc/pd/rt/7500/prodlit/haibd_ov.pdf" [online], pp 1, [Aug. 27, 2010]

[5] P. Zhao and Z. Wang, "Low Power Design of VLSI Circuits and Systems," *Proceedings of IEEE 8th International conference on ASIC*, pp. 17-20, 2009.

[6] A. Sanyal, A. Rastogi, Wei Chen, and S. Kundu, "An Efficient Technique or Leakage Current Estimation in Nanoscaled CMOS Circuits Incorporating Self-Loading Effects," *proceedings of IEEE transaction on Computers*, vol. 59, no. 7, pp. 922-932, July 2010.

[7] H. Rahman and C. Chakrabarti, "A leakage estimation and reduction technique for scaled CMOS logic circuits considering gate-leakage," *proceedings of the International Symposium on Circuits and Systems Circuits and Systems,* vol. 2, pp. 297-300, May 2004.

[8] K. Nose and T. Sakurai, "Analysis and future trend of short-circuit power," *Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 9, pp. 1023-1030, Sep 2000.

[9] S. V. Kumar, C. H. Kim, and S .S. Sapatnekar, "Adaptive Techniques for Overcoming Performance Degradation Due to Aging in CMOS Circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 99, pp. 1-12, December 2009.

[10] D.T. Blaauw, A. Dharchoudhury, R. Panda, S. Sirichotiyakul, C. Oh, and T. Edwards, "Emerging power management tools for processor design," *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 143-148, 1998

[11] A.J. Martin, and M. Nystrom, "Asynchronous Techniques for System-on-Chip Design," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1089-1120, June 2006

[12] R. Saleh, S. Wilton, S. Mirabbasi, A. Hu, M. Greenstreet, G. Lemieux, P.P. Pande, C. Grecu, and A. Ivanov, "System-on-Chip: Reuse and Integration," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1050-1069, June 2006

[13] C. Juanjuan, W. Xing, J. Yunhian, and Qiang Z., "Improve clock gating through power-optimal enable function selection", *12th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, pp. 30-33, April 2009

[14] IEEE Standards Association, "IEEE 802.3 Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and Physical Layer specifications," page 11, 2008

[15] IEEE Standards Association, "IEEE 802.3 Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and Physical Layer specifications," page 444, 2008

[16] Akamai Technologies "How will the Internet Scale", 2008, Available: http://www.akamai.com/html/perspectives/whitepapers_content.html [online] 2008, [Aug. 27, 2010]

[17] Spirent Communications, "Internet Mix (IMIX) Journal", *Test Methodology Journal,* Available: http://gospirent.com/whitepaper/IMIX%20Test%20Methodolgy%20Journal.pdf [online] March 2006, [Aug. 27, 2010]

[18] H.Kawaguci, and T. Sakurai, "A reduced clock-swing flip-flop(RCSFF) for 63% power reduction," *IEEE Journal of Solid-Stage Circuits*, vol. 33, no. 5, pp. 807-811, May 1998

[19] T. Lang, E. Musoll, and J. Cortadella, "Individual flip-flops with Gated Clocks for Low Power Datapaths," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 44, no. 6, pp. 507-516, June 1997

[20] M. Nogawa, and Y. Ohtmo, "A Data-Transition Look-Ahead DFF Circuit for Statistical Reduction in Power Consumption," *IEEE Journal of Solid-Stage Circuits*, vol. 33, no. 5, pp. 702-706, May 1998

[21] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits", *IEEE J. Solid-State Circuits*, vol. SC-19, pp. 1984 .

[22] C.K. Teh, M. Hamada, T. Fujita, H. Hara, N. Ikumi, Y. Oowaki, "Conditional Data Mapping flip-flops for Low-Power and High-Performance Systems," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, issue 12, pp 1379-1383, 2006

[23] M. W. Allam, "New Methodologies for Low-Power High-Performance Ditial VLSI Design," Ph.D thesis, University of Waterloo, Waterloo, ON, Canada, pp 87, 2000

[24] D. Liu, C. Svensson, "Power Consumption Estimation in CMOS VLSI Chips," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 7, pp 663-670 , June 1994

[25] J.L. Neves, and E. G. Friedman, "Optimal Clock Skew Scheduling Tolerant to Process Variations," *Design Automation Conference Proceedings, 33rd*, pp 623-628, Jun 1996

[26] J.L. Neves, and E. G. Friedman, "Design Methodology for Synthesizing Clock Distribution Networks Exploiting Nonzero Localized Clock Skew," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 4, no. 2, pp 623-628, Jun 1996

[27] B. Liu, B. Andrew, X. Xu, J. Hu, and G. Venkataraman, "A Global Minimum Clock Distribution Network Augmentation Algorithm for Guaranteed Clock Skew Yield," *Design Automation Conference*, pp 24-31, Jan 2007

[28] T. Baumann, J. Berthold, T. Niedermeier, J. Dienstuhl, D. Schmitt-Landsiedel, C. Pacha, "Performance improvement of embedded low-power microprocessor cores by selective flip flop replacement," *European Solid State Circuits Conference*, pp 308-311, Sept 2007

[29] G. Palumbo, F. Pappalardo, S. Sannella, "Evaluation on power reduction applying gated clock approaches," *IEEE International symposium on Circuits and Systems*, vol. 4, pp IV85-IV88, Aug 2002

[30] J. N. Kozhaya, L. A. Bakir, "An electrically robust method for placing power gating switches in voltage islands," *IEEE Conference on Custom Integrated Circuits*, pp 321-324, 2004

[31] D. Markovic, B. Nikolic, R. W. Brodersen, "Analysis and Design of Low-Energy flip-flops," *International Symposium on Low Power Electronics and Design*, pp 52-55, Aug 2001

[32] J. M. Rabaey, A. Chandrakasan, B. Nikolic, *Digital Integrated Circuits – A Design Perspective*, 2nd ed. Prentice Hall Electronics and VLSI Series, pp 335, 2003

[33] A.P. Chandrakasan, R. Allmon, A. Stratakos, R. W. Brodersen, "Design of Portable Systems," *Proceedings of the IEEE Custom Integrated Circuits Conference*, Issue 1, pp 259-266, May 1994

[34] M. Hamada, T. Terazawa, T. Higashi, S. Kitabayashi, S. Mita, Y. Watanabe, M. Ashino, H. Hara, T. Kuroda, "flip-flop Selection Technique for Power-Delay Trade-Off," *IEEE International Solid-State Circuits Conference*, pp 270-271, 1999

[35] R. Zimmermann and W. Fichtner, "Low-Power Logic Styles: CMOS Versus Pass-Transistor Logic," *IEEE Journal of Solid-State Circuits*, vol.32, no.7, pp 1079-1090, July 2007

[36] A.G.M. Strollo, E. Napoli, D. De Caro, "New Clock-Gating Techniques for Low-Power Flip-flops," *The International Symposium on Low Power Electronics and Design*, pp 114-119, 2000

[37] B.S. Kong, S.S. Kim, Y.H. Jun, "Conditional-capture flip-flop for statistical power reduction," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 8, pp 1263-1271, 2001.

Appendix A **Data Generator Verilog-A Code**

```
// VerilogA for flops, data, veriloga
`include "constants.vams"
`include "disciplines.vams"

module data(data_out);
output [7:0] data_out;
electrical [7:0] data_out;
parameter integer activity_factor = 0;
integer x;
integer rand_data[7:0];
integer rand_data_d[7:0];
parameter real start = 1n;
parameter real period = 2.5n;
integer seed;
integer iterator;

analog begin
  @(initial_step) begin
    rand_data[0] = 1;
    rand_data[1] = 0;
    rand_data[2] = 0;
    rand_data[3] = 0;
    rand_data[4] = 0;
    rand_data[5] = 0;
    rand_data[6] = 0;
    rand_data[7] = 0;
    seed=0;
  end
  @(timer(start, period)) begin
    x = abs($random(seed) % 50);
    if (x<activity_factor) begin
       for (iterator=0;iterator<8;iterator=iterator+1) begin
         rand_data_d[iterator] = rand_data[iterator];
       end
       for (iterator=0;iterator<7;iterator=iterator+1) begin
         rand_data[iterator+1]=rand_data_d[iterator];
       end
       rand_data[0] = rand_data_d[7] ^ rand_data_d[6] ^ rand_data_d[4] ^ rand_data_d[2];
    end
    else begin
      for (iterator=0;iterator<8;iterator=iterator+1) begin
        rand_data[iterator] = rand_data[iterator];
      end
```

```
   end
   $strobe ("Generated Data is %b , %g\t", rand_data[0], $abstime);
  end
  //V(data_out) <+ transition(rand_data,0.2n,0.2n);
  V(data_out[0]) <+ rand_data[0];
  V(data_out[1]) <+ rand_data[1];
  V(data_out[2]) <+ rand_data[2];
  V(data_out[3]) <+ rand_data[3];
  V(data_out[4]) <+ rand_data[4];
  V(data_out[5]) <+ rand_data[5];
  V(data_out[6]) <+ rand_data[6];
  V(data_out[7]) <+ rand_data[7];
end

endmodule
```

Appendix B **Enable Generator Verilog-A Code**

```
// VerilogA for flops, data, veriloga
`include "constants.vams"
`include "disciplines.vams"

module enable_generator(data_out);
output data_out;
electrical data_out;
parameter integer enable_asserted = 100;
integer x;
integer rand_data;
parameter real start = 1n;
parameter real period = 2.5n;
integer seed;

analog begin
  @(initial_step) begin
    rand_data = 0;
    seed=0;
  end
  @(timer(start, period)) begin
    x = abs($random(seed) % 100);
    if (x<enable_asserted) begin
      rand_data=1;
    end
    else begin
      rand_data = 0;
    end
    $strobe ("Generated Enable Value is %b , %g\t", rand_data, $abstime);
  end
  //V(data_out) <+ transition(rand_data,0.2n,0.2n);
  V(data_out) <+ rand_data;

end

endmodule
```

Appendix C **PRBS Generator Verilog-A Code**

```
// VerilogA for flops, prbs, veriloga
`include "constants.vams"
`include "disciplines.vams"

module prbs(data_out);
output [7:0] data_out;
electrical [7:0] data_out;
parameter integer activity_factor = 0;
integer x;
integer rand_data[7:0];
integer rand_data_d[7:0];
parameter real start = 1n;
parameter real period = 2.5n;
integer seed;
integer iterator;

analog begin
 @(initial_step) begin
  rand_data[0] = 1;
  rand_data[1] = 0;
  rand_data[2] = 0;
  rand_data[3] = 0;
  rand_data[4] = 0;
  rand_data[5] = 0;
  rand_data[6] = 0;
  rand_data[7] = 0;
  seed=0;
 end
 @(timer(start, period)) begin
  x = abs($random(seed) % 50);
  if (x<activity_factor) begin
     for (iterator=0;iterator<8;iterator=iterator+1) begin
       rand_data_d[iterator] = rand_data[iterator];
     end
     for (iterator=0;iterator<7;iterator=iterator+1) begin
       rand_data[iterator+1]=rand_data_d[iterator];
     end
     rand_data[0] = rand_data_d[7] ^ rand_data_d[6] ^ rand_data_d[4] ^ rand_data_d[2];
  end
  else begin
   for (iterator=0;iterator<8;iterator=iterator+1) begin
     rand_data[iterator] = rand_data[iterator];
   end
  end
```

```
   $strobe ("Generated Data is %b , %g\t", rand_data[0], $abstime);
  end
  //V(data_out) <+ transition(rand_data,0.2n,0.2n);
  V(data_out[0]) <+ rand_data[0];
  V(data_out[1]) <+ rand_data[1];
  V(data_out[2]) <+ rand_data[2];
  V(data_out[3]) <+ rand_data[3];
  V(data_out[4]) <+ rand_data[4];
  V(data_out[5]) <+ rand_data[5];
  V(data_out[6]) <+ rand_data[6];
  V(data_out[7]) <+ rand_data[7];
end

endmodule

// VerilogA for flops, prbs_with_enable, veriloga
`include "constants.vams"
`include "disciplines.vams"

module prbs_with_enable(data_out,enable);
output [7:0] data_out;
output       enable;
electrical [7:0] data_out;
electrical       enable;
parameter integer activity_factor = 0;
integer x;
integer rand_data[7:0];
integer rand_data_d[7:0];
parameter real start = 1n;
parameter real period = 2.5n;
integer seed;
integer iterator;
integer enable_d;

analog begin
  @(initial_step) begin
    enable_d    = 0;
    rand_data[0] = 1;
    rand_data[1] = 0;
    rand_data[2] = 0;
    rand_data[3] = 0;
    rand_data[4] = 0;
    rand_data[5] = 0;
    rand_data[6] = 0;
    rand_data[7] = 0;
    seed=0;
```

```
 end
 @(timer(start, period)) begin
  x = abs($random(seed) % 50);
  if (x<activity_factor) begin
     enable_d = 1;
     for (iterator=0;iterator<8;iterator=iterator+1) begin
       rand_data_d[iterator] = rand_data[iterator];
     end
     for (iterator=0;iterator<7;iterator=iterator+1) begin
       rand_data[iterator+1]=rand_data_d[iterator];
     end
     rand_data[0] = rand_data_d[7] ^ rand_data_d[6] ^ rand_data_d[4] ^ rand_data_d[2];
  end
  else begin
    enable_d = 0;
    for (iterator=0;iterator<8;iterator=iterator+1) begin
      rand_data[iterator] = rand_data[iterator];
    end
  end
  $strobe ("Generated Data is %b , %g\t", rand_data[0], $abstime);
 end
 //V(data_out) <+ transition(rand_data,0.2n,0.2n);
 V(data_out[0]) <+ rand_data[0];
 V(data_out[1]) <+ rand_data[1];
 V(data_out[2]) <+ rand_data[2];
 V(data_out[3]) <+ rand_data[3];
 V(data_out[4]) <+ rand_data[4];
 V(data_out[5]) <+ rand_data[5];
 V(data_out[6]) <+ rand_data[6];
 V(data_out[7]) <+ rand_data[7];
 V(enable)      <+ enable_d;
end

endmodule
```

Appendix D **Log Parse - Perl Script**

```
#!/usr/local/bin/perl5.6 -w

use FindBin;
use lib $FindBin::Bin;
use Storable qw(nstore dclone retrieve);
use File::Basename;
use Getopt::Long;

   $infile = "gated_counter_50pct_log.log";
   open (IN, "<$infile") or die "Can't open $infile :$!\n";
   my @text_array = ();
   $generated_data = 0;
   while ($text = <IN>) {
    if ($text =~ /Generated/) {
      $generated_data++;
      @line = split(/ /,$text);
      push (@text_array,$line[4]);
    }
   }
   close IN;
   $enable = 0;
   for ($index=0; $index<@text_array; $index++) {
    if ($text_array[$index] != 1) {
      $enable++;
    }
   }
   $array_size = @text_array;
   $enable_factor = $enable/(@text_array);
   print "Enable factor $enable div $array_size is $enable_factor\n";
```

Appendix E **Counter Model – Perl Script**

```
#!/usr/local/bin/perl5.6 -w

use FindBin;
use lib $FindBin::Bin;
use Storable qw(nstore dclone retrieve);
use File::Basename;
use Getopt::Long;

   $infile = "gated_counter_50pct_log.log";
   open (IN, "<$infile") or die "Can't open $infile :$!\n";
   my @text_array = ();
   $generated_data = 0;
   while ($text = <IN>) {
    if ($text =~ /Generated/) {
     $generated_data++;
     @line = split(/ /,$text);
     push (@text_array,$line[4]);
    }
   }
   close IN;
   $enable = 0;
   for ($index=0; $index<@text_array; $index++) {
    if ($text_array[$index] == 1) {
     $enable++;
    }
   }
   $array_size = @text_array;
   $enable_factor = $enable/(@text_array);
   print "Enable factor $enable div $array_size is $enable_factor\n";

   my @counter=();
   my @counter_factor=();
   my $average_af = 0;
   for ($counter_index=0;$counter_index<16;$counter_index++) {
    $counter[$counter_index] = 0;
    $counter_factor[$counter_index] = 0;
   }
   for ($index=0; $index<$array_size; $index++) {
    if ($text_array[$index] == 1){
    #if ($text_array[$index] == 1 || $text_array[$index]==0) {
     $counter_factor[0]++;
     if($counter[0] == 0) {
```

```
 $counter[0] = 1;
}
else {
 $counter[0] = 0;
 $counter_factor[1]++;
 if($counter[1] == 0) {
  $counter[1] = 1;
 }
 else {
  $counter[1] = 0;
  $counter_factor[2]++;
  if($counter[2] == 0) {
   $counter[2] = 1;
  }
  else {
   $counter[2] = 0;
   $counter_factor[3]++;
   if($counter[3] == 0) {
    $counter[3] = 1;
   }
   else {
    $counter[3] = 0;
    $counter_factor[4]++;
    if($counter[4] == 0) {
     $counter[4] = 1;
    }
    else {
    $counter[4] = 0;
    $counter_factor[5]++;
    if($counter[5] == 0) {
    $counter[5] = 1;
    }
  else {
  $counter[5] = 0;
  $counter_factor[6]++;
  if($counter[6] == 0) {
   $counter[6] = 1;
  }
  else {
  $counter[6] = 0;
  $counter_factor[7]++;
  if($counter[7] == 0) {
   $counter[7] = 1;
  }
  else {
  $counter[7] = 0;
```

```
$counter_factor[8]++;
if($counter[8] == 0) {
 $counter[8] = 1;
}
else {
$counter[8] = 0;
$counter_factor[9]++;
if($counter[9] == 0) {
 $counter[9] = 1;
}
else {
$counter[9] = 0;
$counter_factor[10]++;
if($counter[10] == 0) {
 $counter[10] = 1;
}
else {
$counter[10] = 0;
$counter_factor[11]++;
if($counter[11] == 0) {
 $counter[11] = 1;
}
else {
$counter[11] = 0;
$counter_factor[12]++;
if($counter[12] == 0) {
 $counter[12] = 1;
}
else {
$counter[12] = 0;
$counter_factor[13]++;
if($counter[13] == 0) {
 $counter[13] = 1;
}
else {
$counter[13] = 0;
$counter_factor[14]++;
if($counter[14] == 0) {
 $counter[14] = 1;
}
else {
$counter[14] = 0;
$counter_factor[15]++;
if($counter[15] == 0) {
 $counter[15] = 1;
}
```

```
        else {
        $counter[15] = 0;
        }

        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
    }
for ($counter_index=0;$counter_index<16;$counter_index++) {
  print "Flop $counter_index -> $counter_factor[$counter_index]\n";
  $average_af = $average_af + $counter_factor[$counter_index];
}
$average_af = $average_af / 16;
$average_af = $average_af / 400;
print "Average flip-flop activity factor is $average_af\n";
```

Clock Activity Factor Calculation – Perl Script

```
#!/usr/local/bin/perl5.6 -w

use FindBin;
use lib $FindBin::Bin;
use Storable qw(nstore dclone retrieve);
use File::Basename;
use Getopt::Long;

   $infile = "gated_counter_50pct_log.log";
   open (IN, "<$infile") or die "Can't open $infile :$!\n";
   my @text_array = ();
   $generated_data = 0;
   while ($text = <IN>) {
    if ($text =~ /Generated/) {
      $generated_data++;
      @line = split(/ /,$text);
      push (@text_array,$line[4]);
    }
   }
   close IN;
   $enable = 0;
   for ($index=0; $index<@text_array; $index++) {
    if ($text_array[$index] == 1) {
      $enable++;
    }
   }
   $array_size = @text_array;
   $enable_factor = $enable/(@text_array);
   print "Enable factor $enable div $array_size is $enable_factor\n";

   my @counter=();
   my @counter_factor=();
   my $average_af = 0;
   my $clock_gated_af=0;
   my $data_look_ahead_af = 0;
   my $cod_shared_af = 0;

   for ($counter_index=0;$counter_index<16;$counter_index++) {
    $counter[$counter_index] = 0;
    $counter_factor[$counter_index] = 0;
   }
   for ($index=0; $index<$array_size; $index++) {
    ###if ($text_array[$index] == 1){  ### uncomment if you want to use activity factor from
log file
```

```
if(1){
$clock_gated_af = $clock_gated_af + (2 * 16);
 $cod_shared_af = $cod_shared_af + (2*8);
 $data_look_ahead_af = $data_look_ahead_af + 2;
 $counter_factor[0]++;
 if($counter[0] == 0) {
  $counter[0] = 1;
 }
 else {
  $data_look_ahead_af = $data_look_ahead_af + 2;
  $counter[0] = 0;
  $counter_factor[1]++;
  if($counter[1] == 0) {
   $counter[1] = 1;
  }
  else {
   $data_look_ahead_af = $data_look_ahead_af + 2;
   $counter[1] = 0;
   $counter_factor[2]++;
   if($counter[2] == 0) {
    $counter[2] = 1;
   }
   else {
    $data_look_ahead_af = $data_look_ahead_af + 2;
    $counter[2] = 0;
    $counter_factor[3]++;
    if($counter[3] == 0) {
     $counter[3] = 1;
    }
    else {
     $data_look_ahead_af = $data_look_ahead_af + 2;
     $counter[3] = 0;
     $counter_factor[4]++;
     if($counter[4] == 0) {
      $counter[4] = 1;
     }
     else {
     $data_look_ahead_af = $data_look_ahead_af + 2;
     $counter[4] = 0;
     $counter_factor[5]++;
     if($counter[5] == 0) {
     $counter[5] = 1;
     }
    else {
    $data_look_ahead_af = $data_look_ahead_af + 2;
    $counter[5] = 0;
```

```
$counter_factor[6]++;
if($counter[6] == 0) {
 $counter[6] = 1;
}
else {
$data_look_ahead_af = $data_look_ahead_af + 2;
$counter[6] = 0;
$counter_factor[7]++;
if($counter[7] == 0) {
 $counter[7] = 1;
}
else {
$data_look_ahead_af = $data_look_ahead_af + 2;
$cod_shared_af = $cod_shared_af + (2*8);
$counter[7] = 0;
$counter_factor[8]++;
if($counter[8] == 0) {
 $counter[8] = 1;
}
else {
$data_look_ahead_af = $data_look_ahead_af + 2;
$counter[8] = 0;
$counter_factor[9]++;
if($counter[9] == 0) {
 $counter[9] = 1;
}
else {
$data_look_ahead_af = $data_look_ahead_af + 2;
$counter[9] = 0;
$counter_factor[10]++;
if($counter[10] == 0) {
 $counter[10] = 1;
}
else {
$data_look_ahead_af = $data_look_ahead_af + 2;
$counter[10] = 0;
$counter_factor[11]++;
if($counter[11] == 0) {
 $counter[11] = 1;
}
else {
$data_look_ahead_af = $data_look_ahead_af + 2;
$counter[11] = 0;
$counter_factor[12]++;
if($counter[12] == 0) {
 $counter[12] = 1;
```

```
        }
        else {
        $data_look_ahead_af = $data_look_ahead_af + 2;
        $counter[12] = 0;
        $counter_factor[13]++;
        if($counter[13] == 0) {
         $counter[13] = 1;
        }
        else {
        $data_look_ahead_af = $data_look_ahead_af + 2;
        $counter[13] = 0;
        $counter_factor[14]++;
        if($counter[14] == 0) {
         $counter[14] = 1;
        }
        else {
        $data_look_ahead_af = $data_look_ahead_af + 2;
        $counter[14] = 0;
        $counter_factor[15]++;
        if($counter[15] == 0) {
         $counter[15] = 1;
        }
        else {
        $data_look_ahead_af = $data_look_ahead_af + 2;
        $counter[15] = 0;
        }

        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
        }
    }
for ($counter_index=0;$counter_index<16;$counter_index++) {
 print "Flop $counter_index -> $counter_factor[$counter_index]\n";
```

```
  $average_af = $average_af + $counter_factor[$counter_index];
}
$average_af = $average_af / 16;
$average_af = $average_af / 400;
$clock_gated_af=$clock_gated_af/16;
$clock_gated_af=$clock_gated_af/400;
$data_look_ahead_af=$data_look_ahead_af/16;
$data_look_ahead_af=$data_look_ahead_af/400;
$cod_shared_af=$cod_shared_af/16;
$cod_shared_af=$cod_shared_af/400;


print "Average flip-flop activity factor is $average_af\n";
print "Average Clock-Gated Activity Factor is $clock_gated_af\n";
print "Average DTLA Activity Factor is $data_look_ahead_af\n";
print "Average Cod Shared Activity Factor is $cod_shared_af\n";
```

Appendix F Half-Adder Power Consumption

Table F-1: Half-Adder Power Consumption

Input A	Input B	Power Consumed(nw)
0->0	0->0	68.36
0->0	0->1	290.3
0->0	1->0	549.6
0->0	1->1	66.68
0->1	0->0	409.4
0->1	0->1	668.5
0->1	1->0	509.8
0->1	1->1	830.9
1->0	0->0	613.8
1->0	0->1	418.9
1->0	1->0	1123
1->0	1->1	1008
1->1	0->0	60.37
1->1	0->1	825.7
1->1	1->0	1162
1->1	1->1	92.54

Appendix G Standard Master Slave flip-flop Results

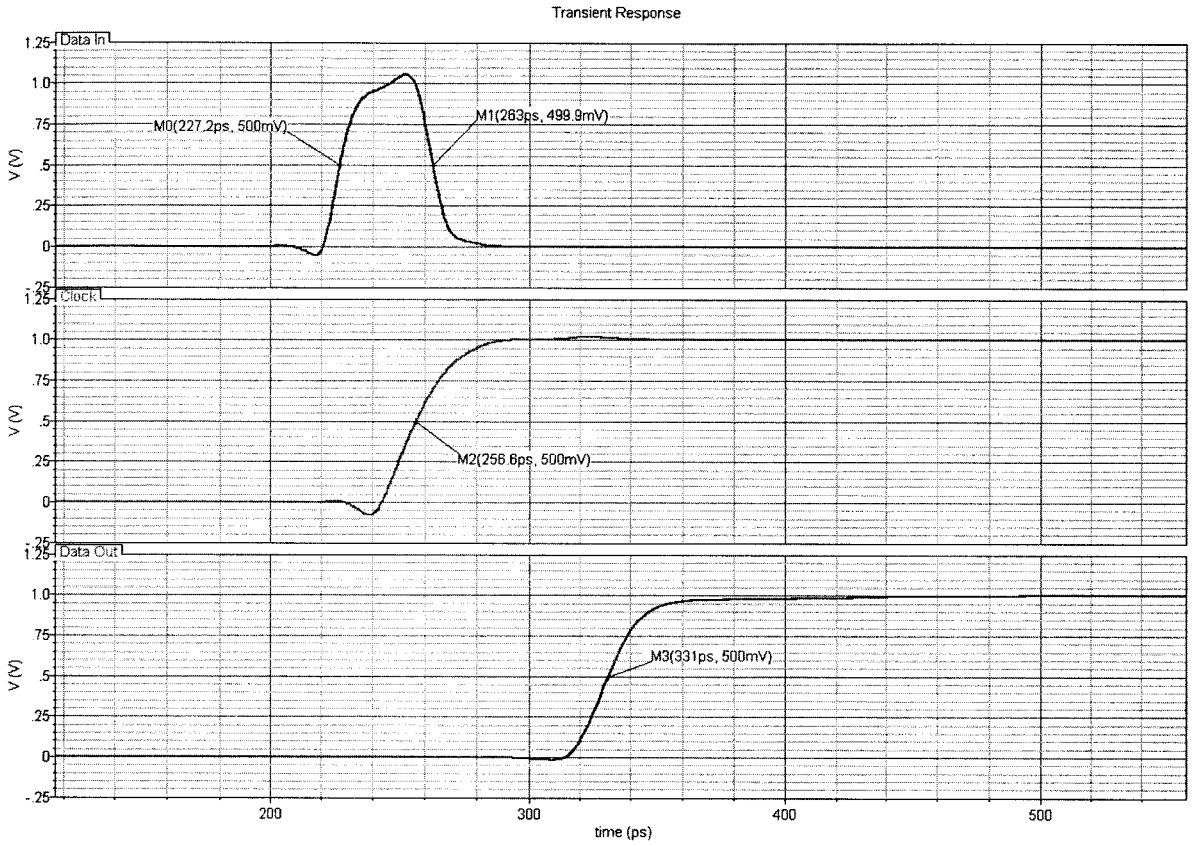

Figure G-1: Standard Master Slave flip-flop Rising Setup-Hold waveform

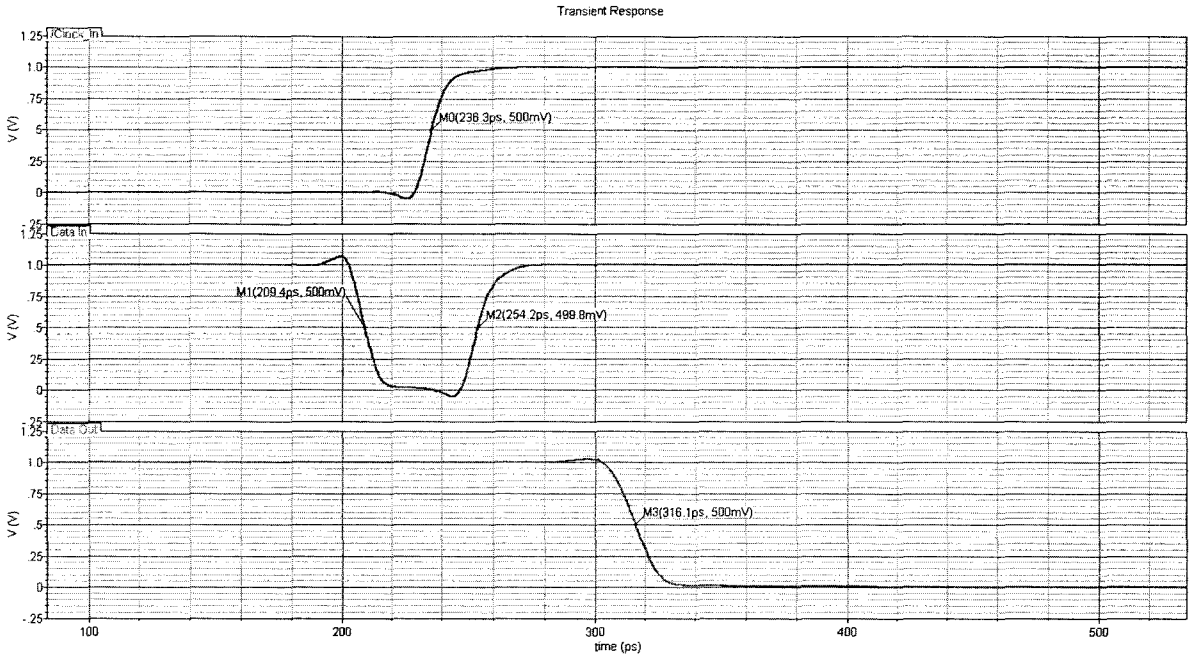

Figure G-2: Standard Master Slave flip-flop Falling Setup-Hold waveform

Table G-1: Master-Slave flip-flop timing characteristics

flip-flop Timing Characteristics	Time
Setup 0-1 Time	29.4 ps
Setup 1-0 Time	26.9 ps
Hold 0-1 Time	6.4 ps
Hold 1-0 Time	17.9 ps
Propagation Delay 0-1	74.4 ps
Propagation Delay 1-0	79.8 ps
flip-flop Race Immunity 0-1	68 ps
flip-flop Race Immunity 1-0	61.9 ps

Table G-G-2: Master Slave flip-flop Power Consumption

Activity Factor	Clock Buffer Power Consumption	Storage element Power Consumption	Total Power Consumption	Clock Buffer power as % of Total Power
0	0.8288	0.009668	0.838468	98.846
6.784	0.8248	0.1808	1.0056	82.020
12.814	0.8291	0.3328	1.1619	71.3572

17.839	0.8195	0.4594	1.2789	64.078
22.362	0.8177	0.5681	1.3858	59.005
26.131	0.8153	0.6646	1.4799	55.091
31.91	0.8131	0.8076	1.6207	50.169
37.688	0.8098	0.9544	1.7642	45.901
41.96	0.8083	1.06	1.8683	43.263
47.487	0.8057	1.198	2.0037	40.210
50	0.8039	1.269	2.0729	38.781

Appendix H Clock-gated flip-flop Simulation Results

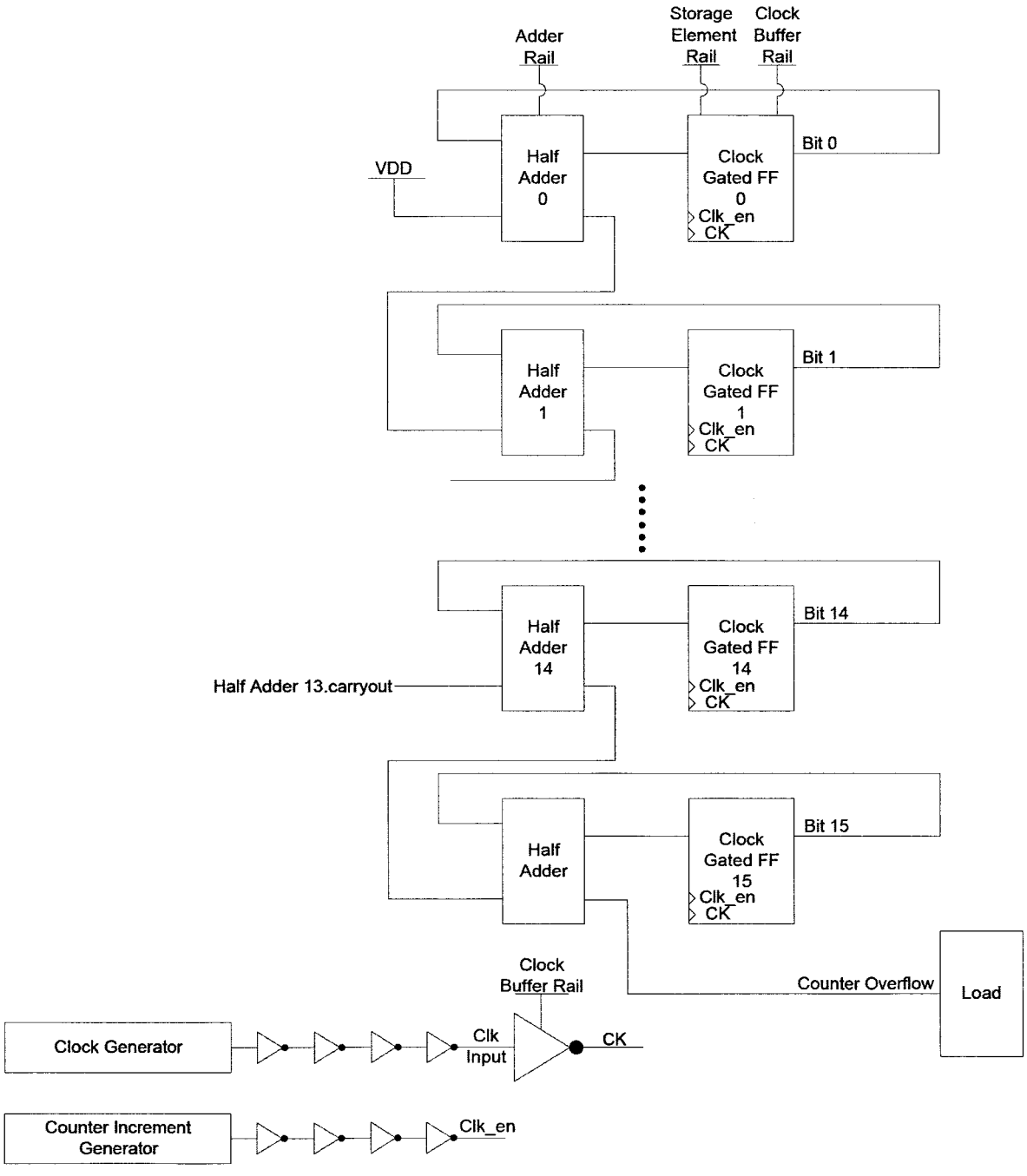

Figure H-1: 16-Bit Counter Test-bed for Clock Gate flip-flop

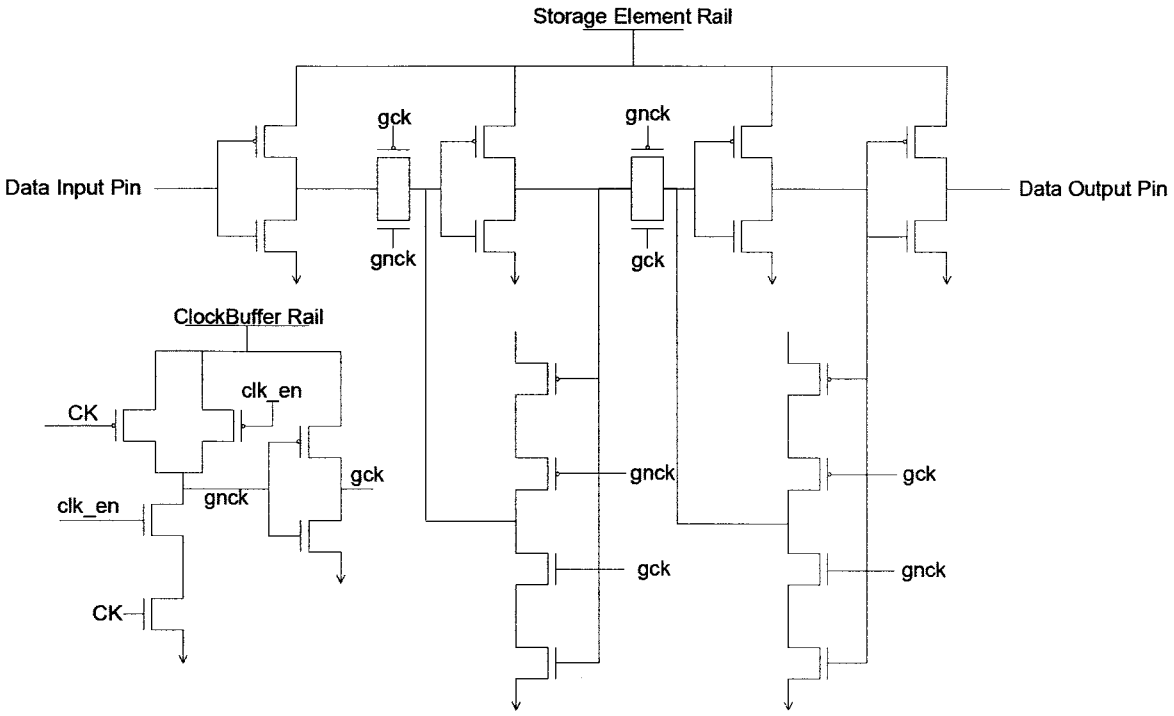

Figure H-2: Clock-Gated flip-flop used in Counter test-bed

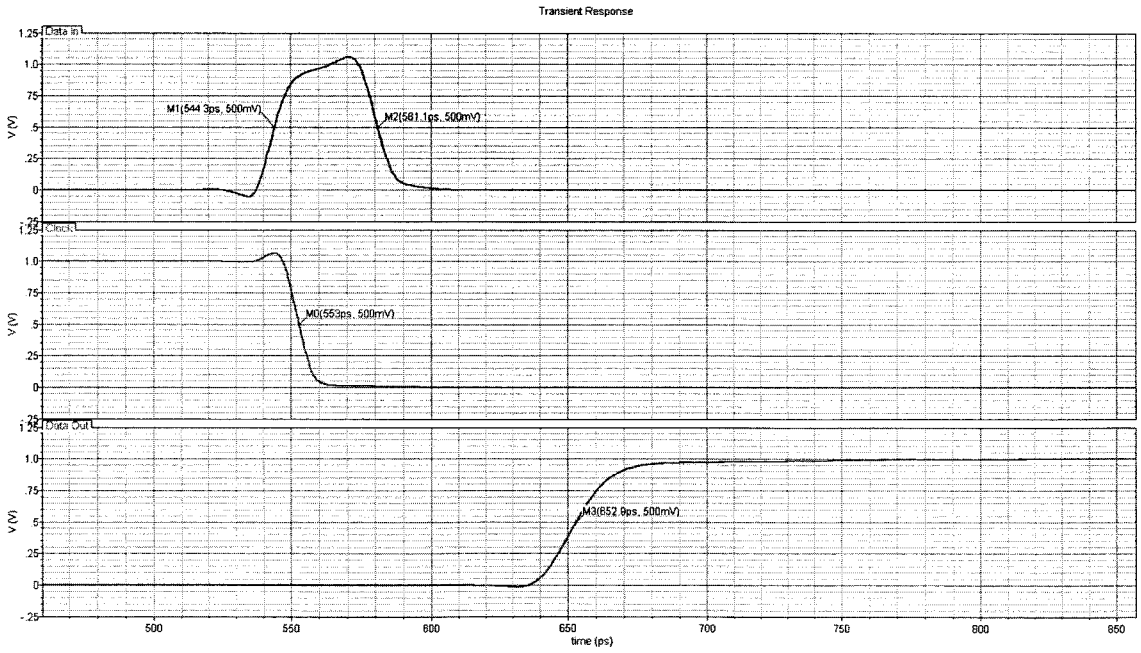

Figure H-3: Clock-gated flip-flop Data Rising Setup-Hold Waveform

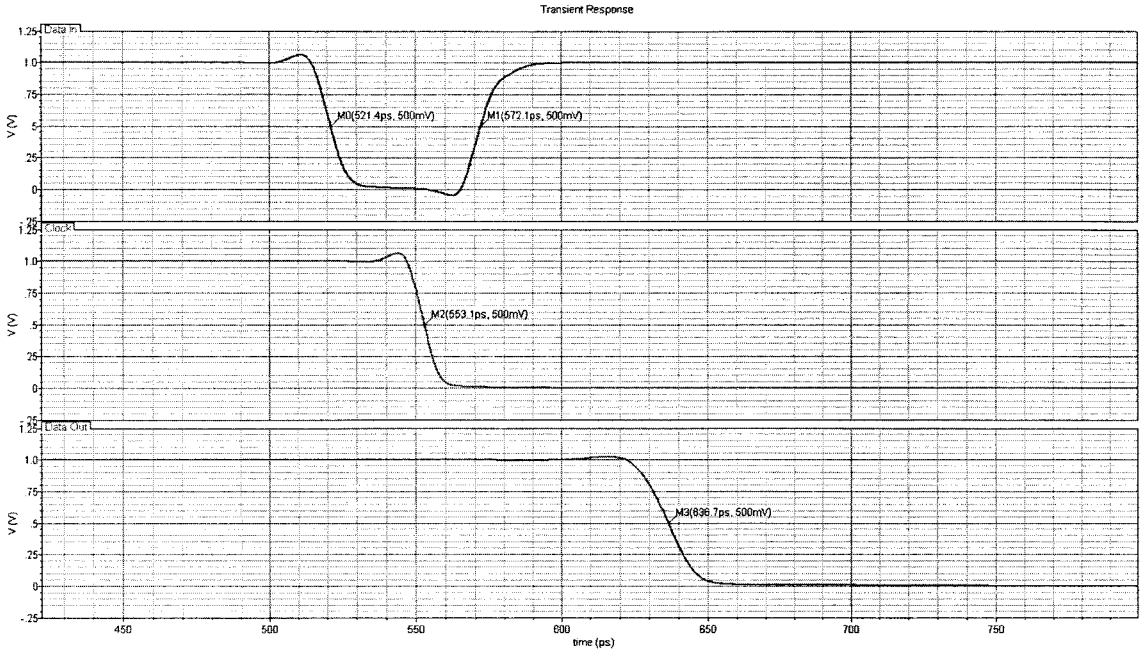

Figure H-4: Clock-gated flip-flop Data Falling Setup-Hold Waveform

Table H-1: Clock-gated flip-flop Timing Characteristics

flip-flop Timing Characteristics	Time
Setup 0-1 Time	8.7 ps
Setup 1-0 Time	31.7 ps
Hold 0-1 Time	28.1 ps
Hold 1-0 Time	19.0 ps
Propagation Delay 0-1	99.9 ps
Propagation Delay 1-0	83.6 ps
flip-flop Race Immunity 0-1	71.8 ps
flip-flop Race Immunity 1-0	64.6 ps

Table H-2: Clock-gated flip-flop Power consumption (Clock is enabled)

Activity Factor	Storage Element (uw)	Clock Buffer (uw)	Total Power (uw)	Clock Buffer Power as % of Total Power
0	0.009	1.254	1.263	99.32
6.784	0.187	1.253	1.440	87.03
12.814	0.343	1.250	1.593	78.45
17.839	0.474	1.249	1.723	72.50
22.362	0.586	1.247	1.833	68.05
26.131	0.685	1.245	1.930	64.50
31.91	0.833	1.242	2.075	59.87
37.688	0.984	1.239	2.223	55.74

41.96	1.092	1.238	2.330	53.13
47.487	1.235	1.237	2.472	50.04
50	1.308	1.234	2.542	48.54

Table H-3: Clock-gated flip-flop Power consumption (Clock is disabled)

Activity Factor	Storage Element (uw)	Clock Buffer (uw)	Total Power (uw)	Clock Buffer Power as % of Total Power
0	0.008	0.247	0.255	96.85
6.784	0.022	0.247	0.269	91.90
12.814	0.034	0.247	0.282	87.77
17.839	0.045	0.247	0.292	84.61
22.362	0.054	0.247	0.302	81.96
26.131	0.063	0.247	0.310	79.72
31.91	0.074	0.247	0.322	76.86
37.688	0.087	0.247	0.334	73.94
41.96	0.096	0.247	0.343	72.13
47.487	0.107	0.247	0.354	69.78
50	0.113	0.247	0.361	68.54

Appendix I **Data Look Ahead flip-flop**

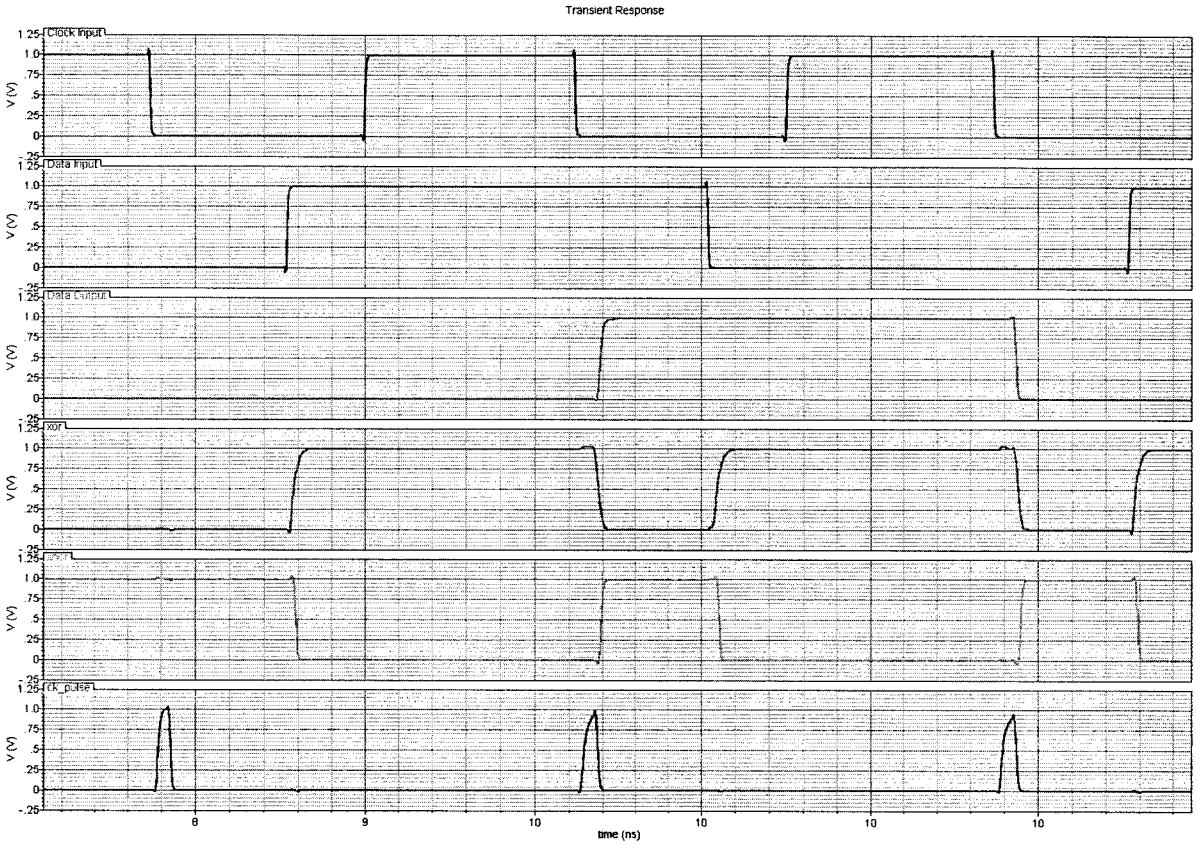

Figure I-1: DTLA internal waveform

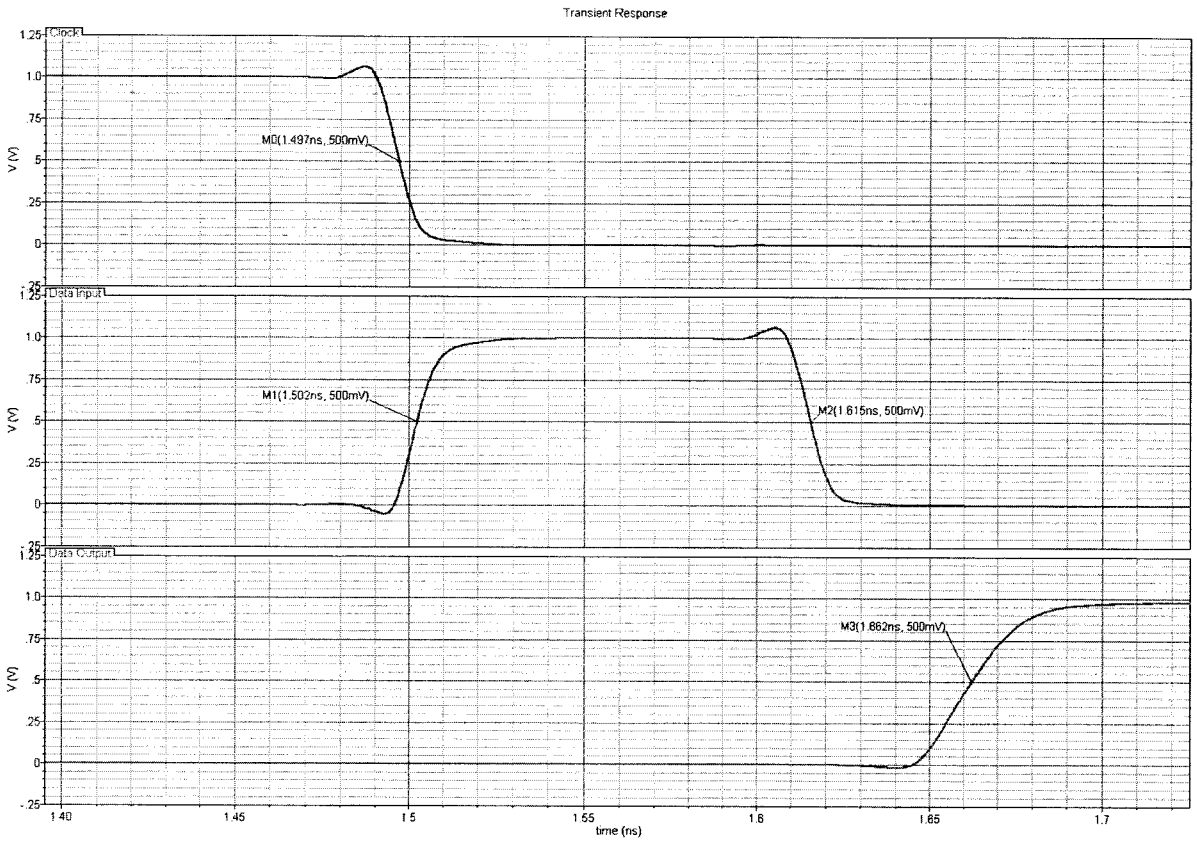

Figure I-2: DTLA flip-flop - No Dyn Latch Rise Setup/Hold

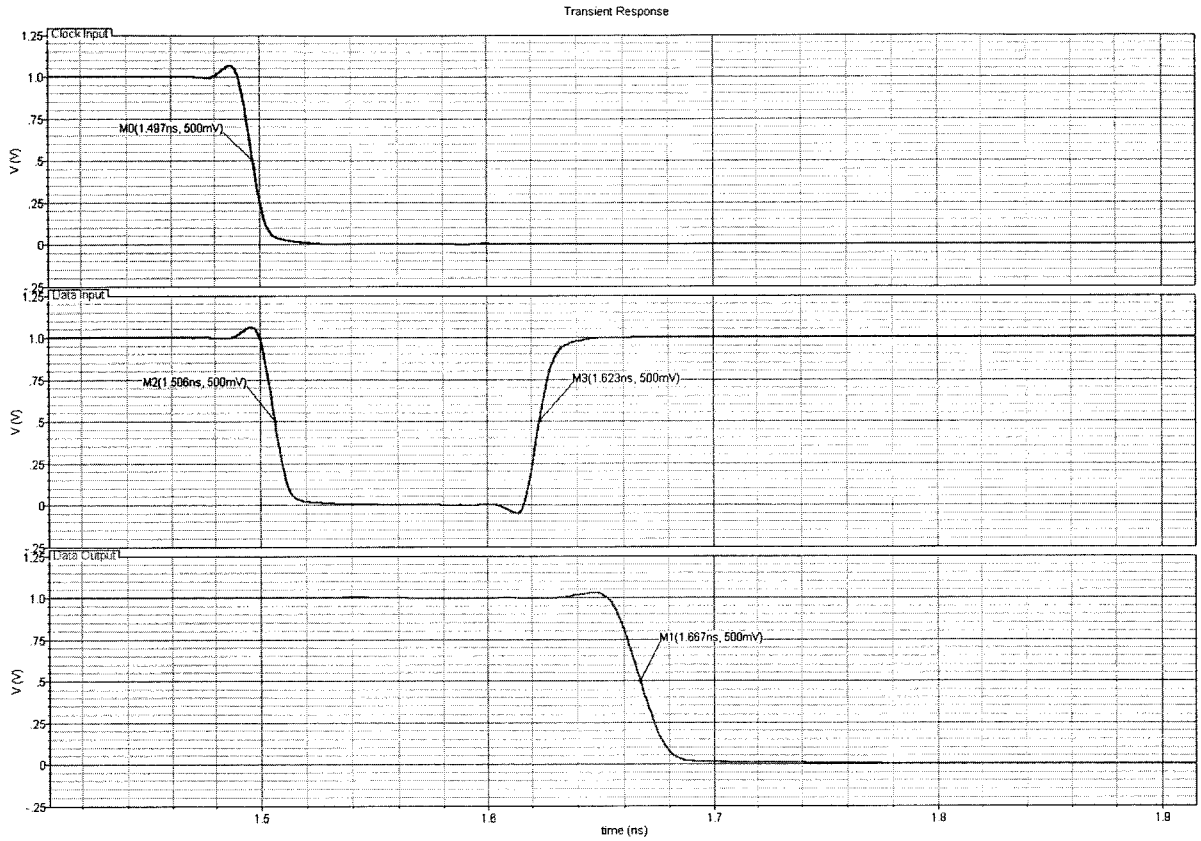

Figure I-3: DTLA flip-flop - No Dyn Latch Fall Setup/Hold

Table I-1: DTLA flip-flop (No Dynamic Latch) Timing Characteristics

flip-flop Timing Characteristics	**Time**
Setup 0-1 Time	-5 ps
Setup 1-0 Time	-9 ps
Hold 0-1 Time	118 ps
Hold 1-0 Time	126 ps
Propagation Delay 0-1	170 ps
Propagation Delay 1-0	165 ps
flip-flop Race Immunity 0-1	52 ps
flip-flop Race Immunity 1-0	39 ps

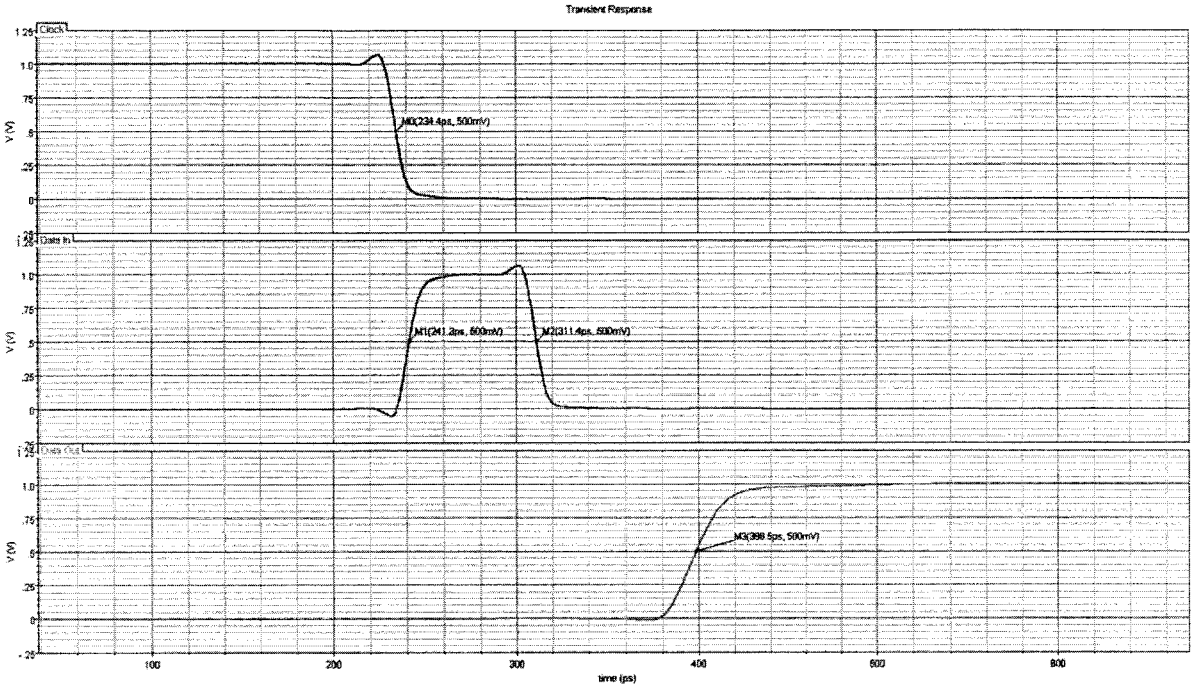

Figure I-4: DLTA flip-flop Rise Setup/Hold Time

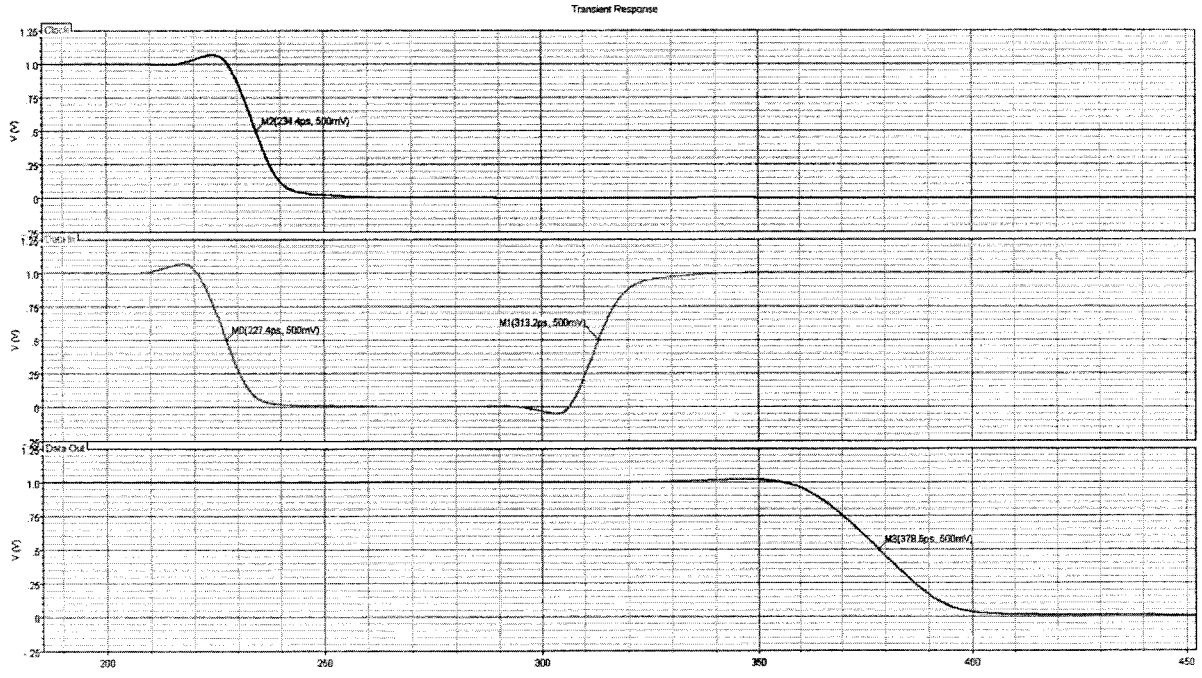

Figure I-5: DLTA flip-flop Fall Setup/Hold Time

Table I-2: Data Look Ahead flip-flop Timing Characteristics

flip-flop Timing Characteristics	**Time**
Setup 0-1 Time	-6.9 ps
Setup 1-0 Time	7 ps
Hold 0-1 Time	77 ps
Hold 1-0 Time	78.8 ps
Propagation Delay 0-1	164.1 ps
Propagation Delay 1-0	144.1 ps
flip-flop Race Immunity 0-1	87.1 ps
flip-flop Race Immunity 1-0	65.6 ps

Table I-3: Power Consumption for Data Look Ahead flip-flop

Activity Factor	Storage Element (uw)	Clock Buffer (uw)	Total Power (uw)
0	0.01	2.39	2.40
6.784	0.01	2.39	2.41
12.814	0.24	2.54	2.79
17.839	0.45	2.68	3.12
22.362	0.63	2.79	3.42
26.131	0.78	2.88	3.67
31.91	0.91	2.97	3.88
37.688	1.11	3.10	4.21
41.96	1.31	3.22	4.54
47.487	1.46	3.32	4.78
50	1.65	3.44	5.10

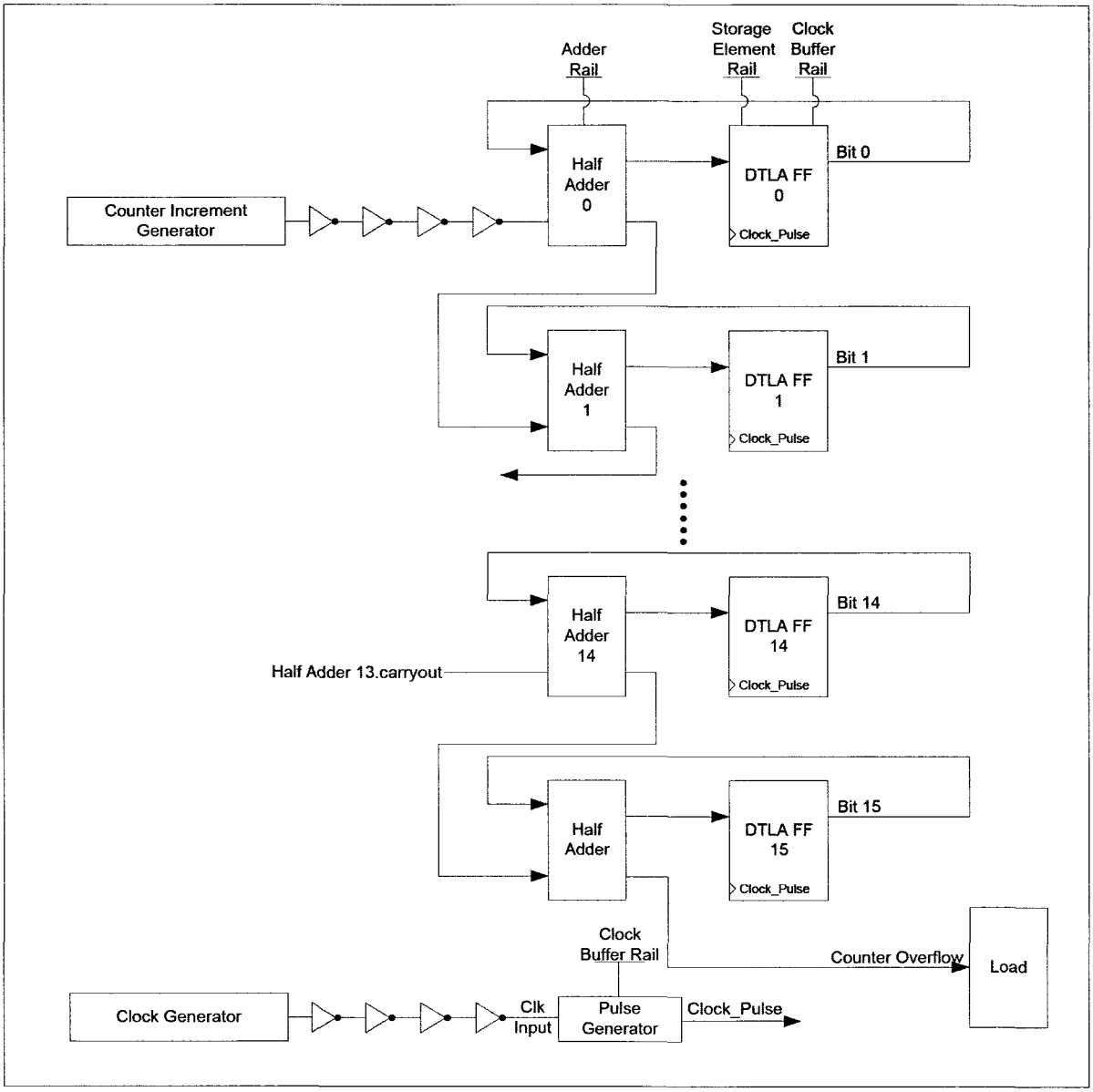

Figure I-6: DTLA Counter Test-bench

Appendix J Clock on Demand flip-flop

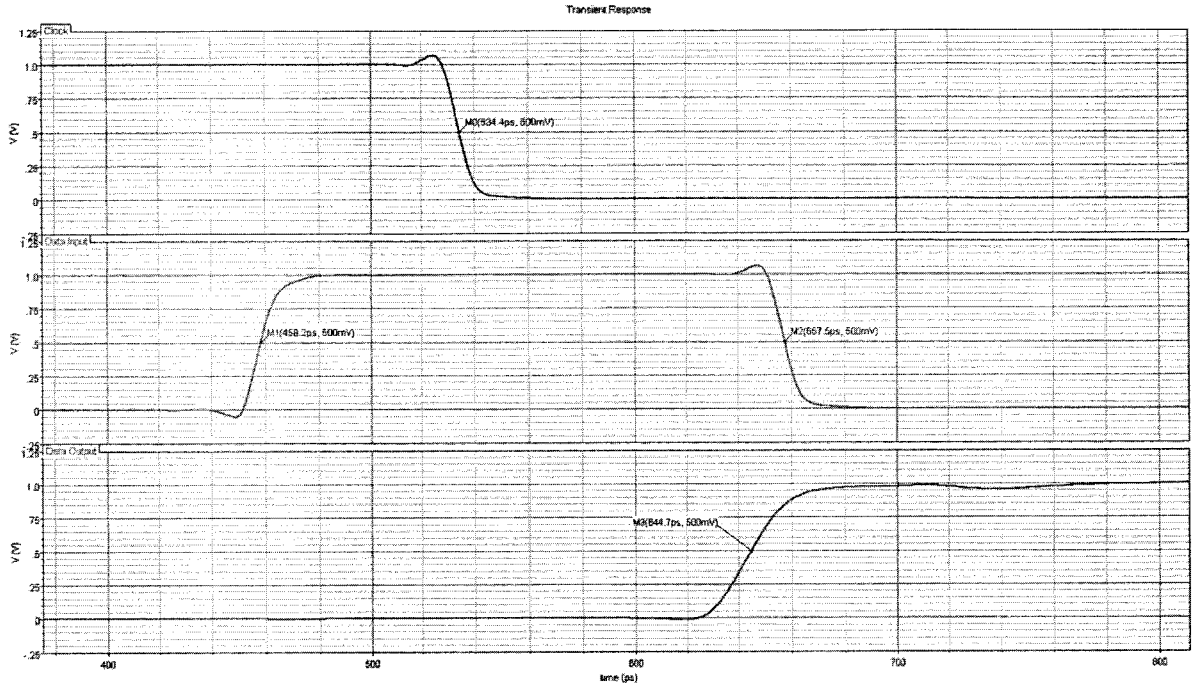

Figure J-1: CoD flip-flop Rise Setup/Hold waveform

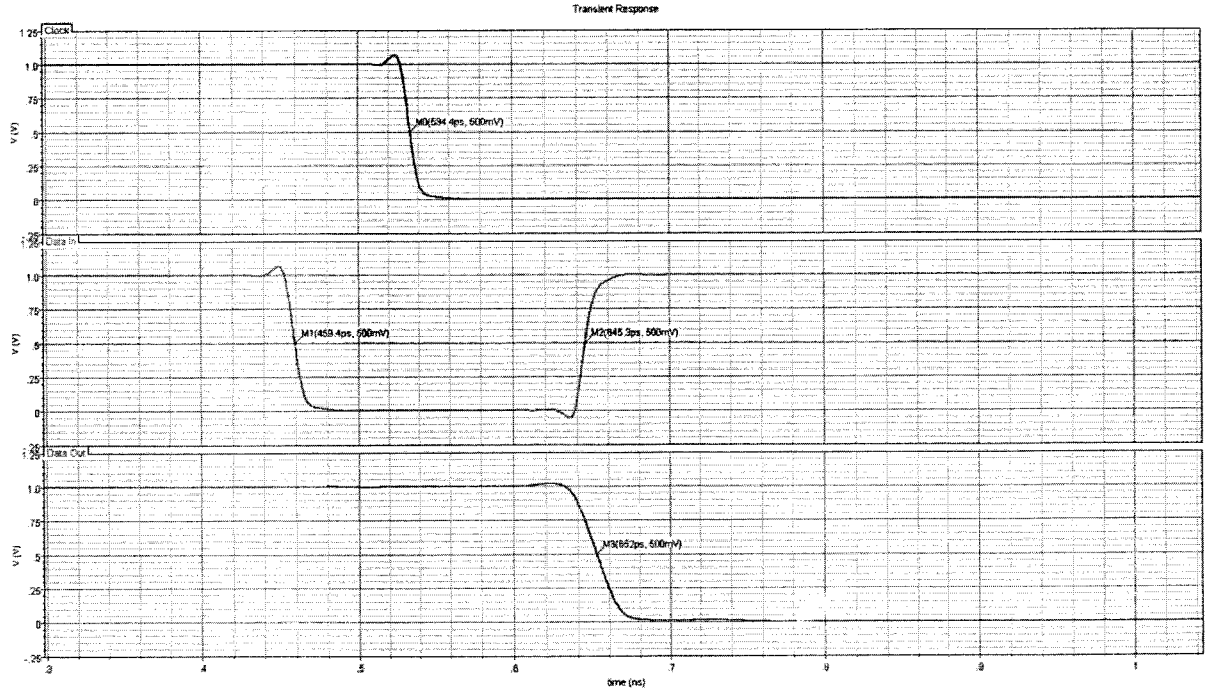

Figure J-2: CoD flip-flop Rise Setup/Hold waveform

Table J-1: CoD flip-flop Timing Characteristics

flip-flop Timing Characteristics	**Time**
Setup 0-1 Time	76.2 ps
Setup 1-0 Time	75.0 ps
Hold 0-1 Time	123.1 ps
Hold 1-0 Time	110.9 ps
Propagation Delay 0-1	110.3 ps
Propagation Delay 1-0	117.6 ps
flip-flop Race Immunity 0-1	-12.8 ps
flip-flop Race Immunity 1-0	6.7 ps

Table J-2: CoD flip-flop Power Consumption Measurements

Activity Factor	Storage Element (uw)	Clock Buffer (uw)	Total Power (uw)
0	0.361	0.008787	0.361
6.784	0.5673	0.2177	0.5673
12.814	0.7512	0.4017	0.7512
17.839	0.9042	0.5551	0.9042
22.362	1.037	0.6859	1.037
26.131	1.152	0.8022	1.152
31.91	1.328	0.9763	1.328
37.688	1.503	1.154	1.503
41.96	1.634	1.282	1.634
47.487	1.802	1.45	1.802
50	1.885	1.536	1.885

Appendix K **Clock on Demand flip-flop (with Hold Fix)**

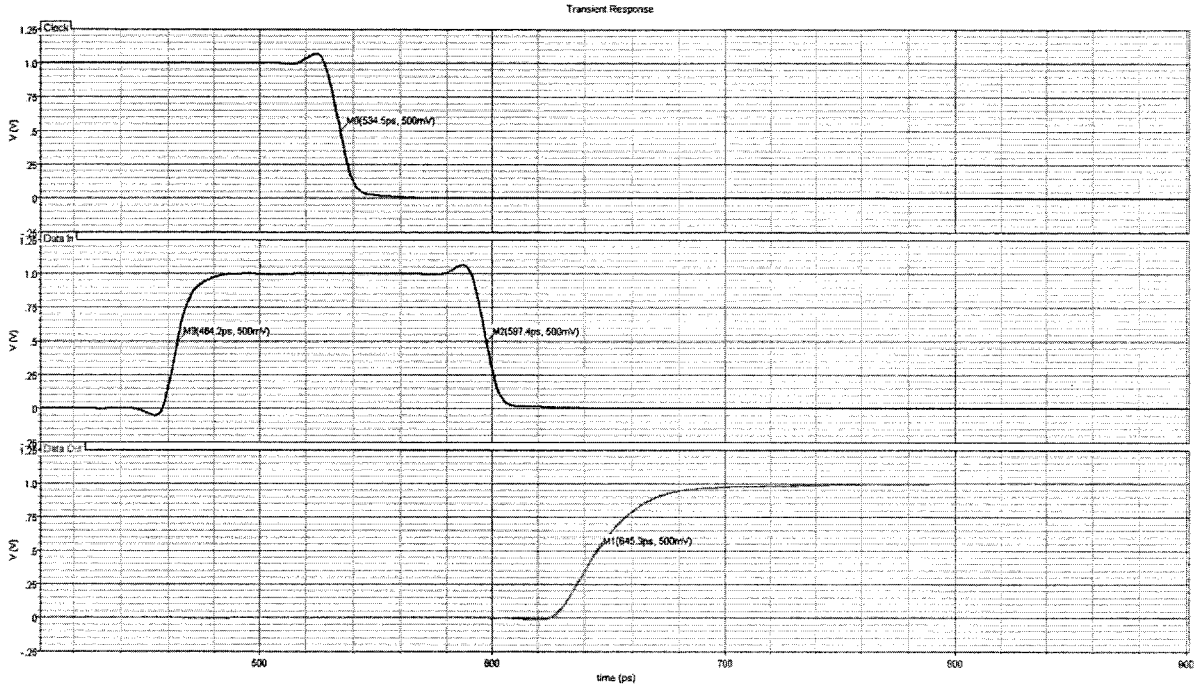

Figure K-1: CoD flip-flop (w Hold Fix) Rise Setup/Hold waveform

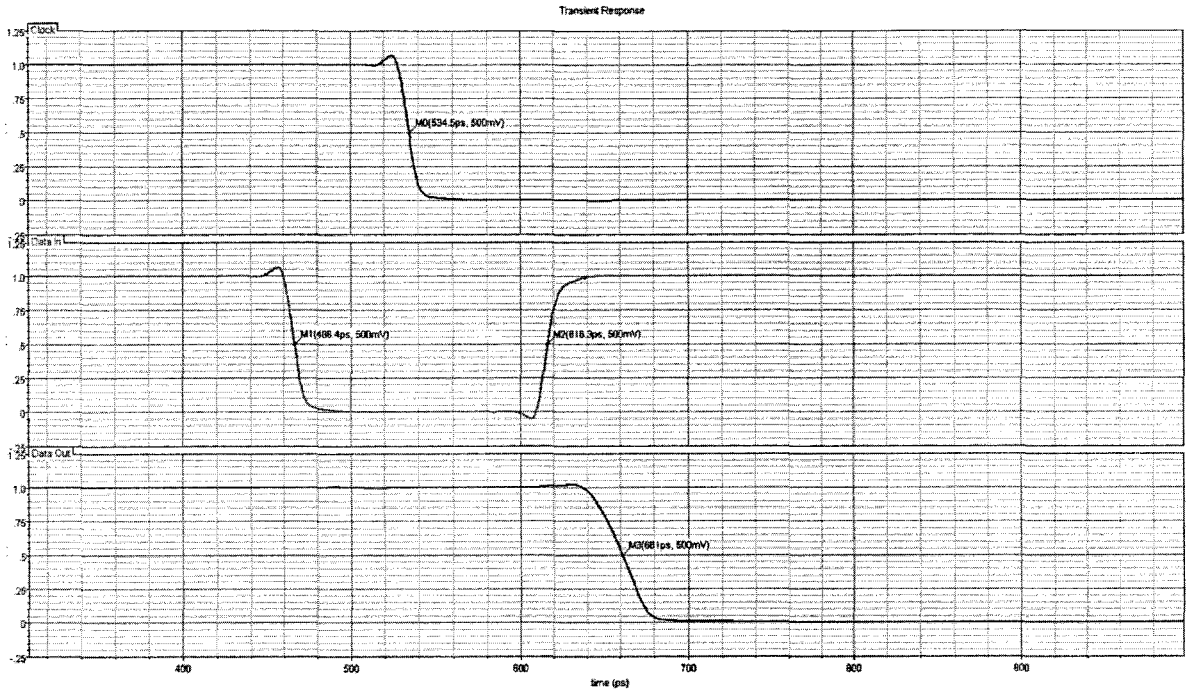

Figure K-2: CoD flip-flop (w Hold Fix) Fall Setup/Hold waveform

Table K-1: CoD flip-flop Timing Characteristics

flip-flop Timing Characteristics	Time
Setup 0-1 Time	70.3 ps
Setup 1-0 Time	68.1 ps
Hold 0-1 Time	62.9 ps
Hold 1-0 Time	81.8 ps
Propagation Delay 0-1	110.8 ps
Propagation Delay 1-0	146.5 ps
flip-flop Race Immunity 0-1	47.9 ps
flip-flop Race Immunity 1-0	64.7 ps

Table K-2: CoD flip-flop Power Consumption Measurement Results

Activity Factor	Storage Element (uw)	Clock Buffer (uw)	Total Power (uw)
0	0.35	0.01	0.36
6.784	0.62	0.21	0.84
12.814	0.86	0.39	1.26
17.839	1.05	0.55	1.60
22.362	1.22	0.68	1.90

26.131	1.37	0.7986	2.1686
31.91	1.591	0.9726	2.5636
37.688	1.81	1.15	2.96
41.96	1.973	1.278	3.251
47.487	2.181	1.446	3.627
50	2.284	1.532	3.816

Appendix L PERL PRBS activity factor model

```
#!/usr/local/bin/perl5.6 -w

use FindBin;
use lib $FindBin::Bin;
use Storable qw(nstore dclone retrieve);
use File::Basename;
use Getopt::Long;

   my @prbs=();
   my @prbs_old=();
   my @prbs_factor=();
   my $average_af = 0;
   for ($prbs_index=0;$prbs_index<8;$prbs_index++) {
    $prbs[$prbs_index] = 0;
    $prbs_old[$prbs_index] = 0;
    $prbs_factor[$prbs_index] = 0;
   }
   $prbs[0] = 1;
   $prbs_old[0] = 1;
   for ($index=0; $index<40; $index++) {
     $prbs[7] = $prbs_old[6];
     $prbs[6] = $prbs_old[5];
     $prbs[5] = $prbs_old[4];
     $prbs[4] = $prbs_old[3];
     $prbs[3] = $prbs_old[2];
     $prbs[2] = $prbs_old[1];
     $prbs[1] = $prbs_old[0];
     $prbs[0] = $prbs_old[2] ^ $prbs_old[4] ^ $prbs_old[6] ^ $prbs_old[7];
     for ($i=0; $i<8; $i++){
       if($prbs[$i] != $prbs_old[$i]) {
         $prbs_factor[$i]++;
       }
       $prbs_old[$i] = $prbs[$i];
     }
   }
   for ($prbs_index=0;$prbs_index<8;$prbs_index++) {
    print "Flop $prbs_index -> $prbs_factor[$prbs_index]\n";
    $average_af = $average_af + $prbs_factor[$prbs_index];
   }
   $average_af = $average_af / 8;
   $average_af = $average_af / 80;
   print "Average flip-flop activity factor is $average_af\n";
```

Appendix M **Leakage Results**

Table M-1: Leakage Power Consumption Measurement Results (Per bit)

FlipFlop	Leakage - Clock Disabled (nw)	Leakage - Clock Enabled (nw)
Standard MS	N/A	10.15
Clock-gated	11.75	11.36
DTLA	N/A	25.1
DTLA shared	N/A	17.7875
CoD	N/A	15.72
GoD	10.52375	11.31
DTLA Gated	16.5125	18.7625

Appendix N flip-flop Sizing

Table N-1: Per Storage Bit flip-flop Sizing Comparison

	Normalized PMOS Transistors (*min* Size)	Normalized NMOS Transistors (*min* Size)
Standard MS	12	12
Clock-gated	14	14
DTLA	23	24
DTLA shared (per Storage Element)	17.14588	16.9375
CoD	16	17
GoD (per Storage Element)	9.2925	9.33
DTLA Gated (per Storage Element)	17.27088	17.8125

Appendix O **Post Layout Simulation Graph**

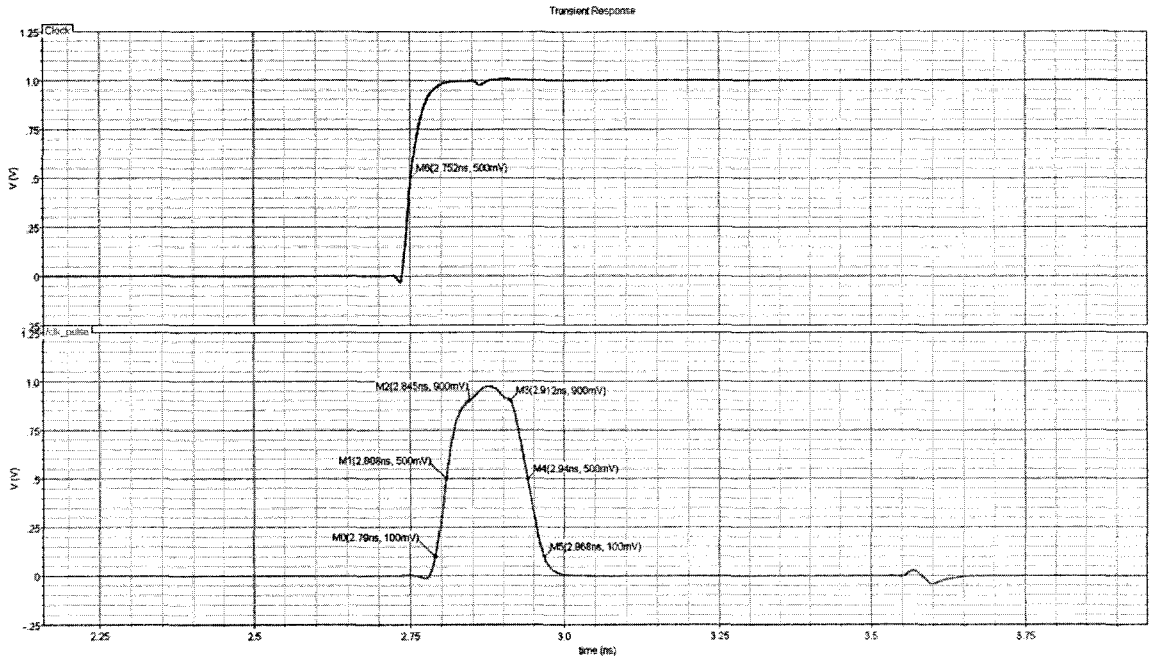

Figure O-1: DTLA flip-flop Pulse Generator Simulation Graph

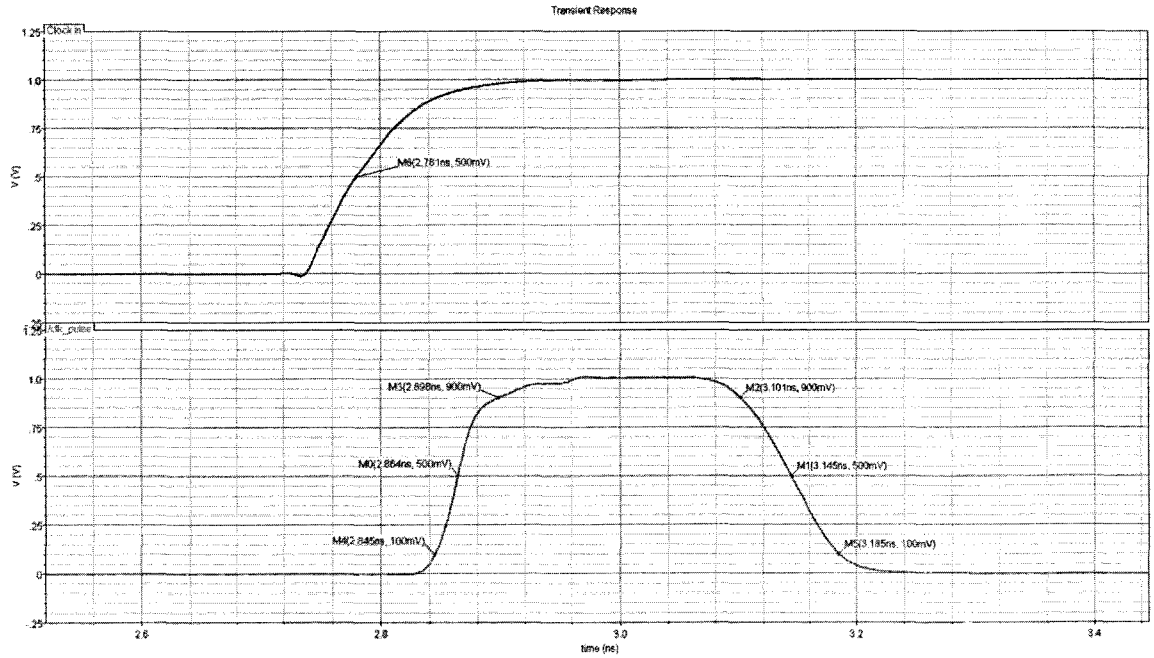

Figure O-2: DTLA flip-flop Pulse Generator Post Layout Simulation Graph