# Adiabatic Approach for Low-Power Passive Near Field Communication Systems

**Maheshwari, S.**

# Adiabatic Approach for Low-Power Passive Near Field Communication Systems


University of Westminster

Sachin Maheshwari

A thesis submitted in partial fulfillment of the requirements of the University of Westminster for the degree of Doctor of Philosophy

**September 2018**

# Author's Declaration

By submitting this thesis, I hereby declare that the research work contained therein is my own, original work, that I am the sole author. No part of this thesis has been submitted in support of an application for any other degree. Where other sources of information have been used, they have been quoted. Finally, this work has been solely conducted at the Applied DSP and VLSI Research Group (ADVRG), Department of Engineering at the University of Westminster.

Sachin Maheshwari

To my family

*For their understandings, love, and support*

# Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor Dr. Viv Bartlett for his great guidance and encouragement during the first three years of my research studies. His retirement was an absolute shock for me but was gratified to have done a significant part of my research during his presence. His deep insight, understanding capability, motivation and above all, his critical thinking which has helped me in each and every step to get going when I felt stuck.

I am also indebted to the head of the Department Professor Izzet Kale, my second supervisor as well as a first supervisor ($4^{th}$ year PhD.) for giving me the opportunity to carry out my research and funding my studies at the University of Westminster. His constant support and encouragement during the course of my studies will always be appreciated and for which I am truly grateful.

I would like to thank the examination committee: Dr. Adem Coskun, my accessor, Dr. Andzrej Tarczynaski, my Chair and my supervisors for their efforts in assessing this thesis.

Also, I thank all the lecturers and staff members at the Department of Engineering among whom I spent the last four years.

I would like to thank my family for their encouragement, love, and unconditional support despite being far for them, throughout my research studies. Without them, I would have never reached this far.

My final and the most heartfelt acknowledgment go to Dr. Himadri Singh Raghav, who supported and encouraged me in the tough time.

This work was supported by Cavendish Research Scholarship whose contribution is gratefully acknowledged.

# Abstract

This thesis tackles the need of ultra-low power electronics in the power limited passive Near Field Communication (NFC) systems. One of the techniques that has proven the potential of delivering low power operation is the Adiabatic Logic Technique. However, the low power benefits of the adiabatic circuits come with the challenges due to the absence of single opinion on the most energy efficient adiabatic logic family which constitute appropriate trade-offs between computation time, area and complexity based on the circuit and the power-clocking schemes. Therefore, five energy efficient adiabatic logic families working in single-phase, 2-phase and 4-phase power-clocking schemes were chosen.

Since flip-flops are the basic building blocks of any sequential circuit and the existing flip-flops are MUX-based (having more transistors) design, therefore a novel single-phase, 2-phase and 4-phase reset based flip-flops were proposed. The performance of the multi-phase adiabatic families was evaluated and compared based on the design examples such as 2-bit ring counter, 3-bit Up-Down counter and 16-bit Cyclic Redundancy Check (CRC) circuit (benchmark circuit) based on ISO 14443-3A standard. Several trade-offs, design rules, and an appropriate range for the supply voltage scaling for multi-phase adiabatic logic are proposed.

Furthermore, based on the NFC standard (ISO 14443-3A), data is frequently encoded using Manchester coding technique before transmitting it to the reader. Therefore, if Manchester encoding can be implemented using adiabatic logic technique, energy benefits are expected. However, adiabatic implementation of Manchester encoding presents a challenge. Therefore, a novel method for implementing Manchester encoding using adiabatic logic is proposed overcoming the challenges arising due to the AC power-clock.

Other challenges that come with the dynamic nature of the adiabatic gates and the complexity of the 4-phase power-clocking scheme is in synchronizing the power-clock

phases and the time spent in designing, validation and debugging of errors. This requires a specific modelling approach to describe the adiabatic logic behaviour at the higher level of abstraction. However, describing adiabatic logic behaviour using Hardware Description Languages (HDLs) is a challenging problem due to the requirement of modelling the AC power-clock and the dual-rail inputs and outputs. Therefore, a VHDL-based modelling approach for the 4-phase adiabatic logic technique is developed for functional simulation, precise timing analysis and as an improvement over the previously described approaches.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| AL | Adiabatic Loss |
| ASK | Amplitude Shift Keying |
| ATQA | Answer to Request for Type-A |
| BPSK | Binary Phase Shift Keying |
| CAL | Clocked Adiabatic Logic |
| CLA | Carry Look Ahead |
| CPAL | Complementary Pass-transistor Adiabatic Logic |
| CRC | Cyclic Redundancy Code |
| DCVSL | Differential Cascode Voltage Switch |
| EACRL | Efficient Adiabatic Charge Recovery Logic |
| ECMA | European Computer Manufacturers Association |
| EoF | End of Frame |
| ES | Energy Saving |
| ESP | Energy Saving Percentage |
| IECRL | Improved Efficient Charge Recovery Logic |
| IEC | International Electrotechnical Commission |
| ISO | International Standard Organization |
| ITRS | International Technology Roadmap Semiconductor |
| LSB | Least Significant Bit |
| MSB | Most Significant Bit |
| NAL | Non-Adiabatic Loss |
| NFC | Near Field Communication |
| NFCIP | Near Field Communication-Interface and Protocol |
| NRZ | Non-Return-to-Zero |
| PC | Power-Clock |
| PCG | Power-Clock Generator |
| PE | Phase encoding |
| PFAL | Positive Feedback Adiabatic Logic |

| | |
|---|---|
| PVT | Process Voltage Temperature |
| RFID | Radio Frequency IDentification |
| SAK | Select Acknowledge |
| SoF | Start of Frame |
| SWC | StepWise Charging |
| UID | Unique IDentifier |

# List of Symbols

| | |
|---|---|
| $C_L$ | Load Capacitance |
| $C_{ox}$ | Gate Oxide Capacitance in a CMOS Transistor |
| $E_{AL}$ | Energy Dissipated by Adiabatic Loss |
| $E_{diss}$ | Energy Dissipated during Charging Process |
| $E_{leak}$ | Energy Dissipated by Leakage Loss |
| $E_{NAL}$ | Energy Dissipated by Non-Adiabatic Loss |
| $E_{NA}$ | Energy Dissipated by Non-Adiabatic (Conventional CMOS) |
| $E_{stored}$ | Energy Stored on the Load Capacitor |
| $E_{TD}$ | Total Energy Dissipated in Adiabatic Logic |
| $f$ | Frequency |
| $I$ | Current |
| $\bar{I}_{leak}$ | Mean Leakage Current |
| $I_D$ | Sub-threshold Current |
| $I_{D0}$ | Sub-threshold Current at zero bias |
| $L$ | Effective Length of the CMOS Transistor |
| $P$ | Power |
| $Q$ | Charge |
| $R$ | Resistance of the Charging Path |
| $R_{ON}$ | ON-Resistance of the Charging Path in a CMOS Transistor |
| $T_r$ | Ramping Time |
| $V$ | Voltage |
| $V_C$ | Voltage across the Capacitor |
| $V_T$ | Thermal Voltage |
| $V_{th}$ | Threshold Voltage of the CMOS Transistor |
| $V_R$ | Voltage across the Resistor |
| $V_{th,p}$ | Threshold Voltage of the pMOS Transistor |
| $V_{DD}$ | Supply Voltage |
| $V_{ramp}$ | Ramp Voltage charging from 0 to $V_{DD}$ |

| | |
|---|---|
| $V_{GS}$ | Gate-Source Voltage in a CMOS Transistor |
| $V_{DS}$ | Drain-Source Voltage in a CMOS Transistor |
| $W$ | Width of the CMOS Transistor |
| $\mu_o$ | Carrier Mobility |
| $\eta$ | Sub-threshold Swing Coefficient |
| $\alpha$ | Switching Activity Factor |

# 1 Introduction

## 1.1 Overview

In the last five years, the use of Near Field Communication (NFC) enabled contactless cards (tag) and handheld devices (reader) such as mobile phones have shown a tremendous increase. The increased usage in the exchange of various types of information, such as telephone numbers, pictures, MP3 files and digital authorizations for contactless payment between two NFC-enabled devices has led the designers to make low energy consumption, a very high priority.

In a passive NFC system, a reader (e.g a smartphone), and a tag communicate by means of Radio Frequency (RF) field [1]. NFC passive tags are battery-less and are powered by radio waves from the reader, thus, needs to be energy efficient due to limited power resources [2]. The increased hardware complexity due to add-on functionalities, such as security and data-storage [3] has the associated cost in terms of high energy dissipation in passive NFC tags. In addition, the increased number of retransmissions when a reader detects an error in tag-to-reader data communication [4] also causes the energy dissipation in a passive NFC system to increase [5]. The high energy dissipation of the passive tags demands the supply of high transmission power from the reader to start the communication [6] and restricts the maximum working distance from the reader [7]. Therefore, if the tag is made energy efficient it can bring large interrogation range with better accuracy without increasing the energy consumption of the reader. In addition, the power transmitted by the reader to energize the tag can also be reduced which will help to increase the energy saving and battery lifetime of the reader.

Manchester coding is one of the techniques used for encoding the data during the tag to the reader transmissions [8]-[10]. It also makes possible to detect the collisions bitwise. Therefore, energy efficient implementation of the Manchester encoding is one of the aims of this thesis. This thesis deals with the ISO/IEC 14443A standard [11]-[13] which specifies the standard protocol, commands and other parameters required for the communication between a reader and a tag.

The 2015 International Technology Roadmap Semiconductor (ITRS) shows that despite technology advances in materials such as high-K gate isolator and copper interconnect; scaling trends are barely keeping up in terms of density and are failing in terms of energy performance [14]. Consequently, an important area of research is the design and implementation of energy efficient data transmission coding and an error detection module for passive NFC system.

One of the design techniques which has the potential for low power (and in existence for more than two decades) is the "Adiabatic Logic Technique". It is one of the promising solutions at the circuit level to achieve a reduction in energy albeit at some cost in terms of performance. Adiabatic circuits use slowly changing ramp like power-clock which rises and falls linearly. This slowly changing power-clock allows approximately constant current charging/discharging and by avoiding the current surges, the circuit dissipates less energy [15], [16]. The power-clock in adiabatic circuits serves as both the power supply as well as the clock for timing the circuit operation [17]. The adiabatic circuits also make possible delivery and the storage of the energy back to the power supply during the discharging process which can be recovered using a power-clock generator. Therefore, this fact supports the argument that the adiabatic logic technique is promising and makes an attractive implementation method for the passive NFC systems instead of non-adiabatic logic design.

Adiabatic logic designs have been widely studied and various energy efficient logic families have been proposed [18]-[23]. These can be divided into two types, "Fully Adiabatic" [23] and "Quasi-Adiabatic" (also known as partial energy recovery logic) [24], [25]. The term "Fully Adiabatic Logic" refers to logic families that can theoretically operate without losses. Therefore, an important property of fully adiabatic circuits is the recovery of, in principle; all the energy supplied to the circuit thereby ideally resulting in zero dissipation. Alternatively, the term "Quasi-Adiabatic Logic"

describes the logic that operates with lower energy but involves some practical losses arising due to the threshold voltage degradation of transistors. Such losses are referred to as Non-Adiabatic Loss (NAL). Quasi-adiabatic logic circuits are designed to recover only a proportion of the delivered energy and are likely to be less complex and occupy less area than fully adiabatic designs. In this thesis, quasi-adiabatic logic is considered and for practical reasons will be referred to as adiabatic logic.

With various multi-phase adiabatic logic designs existing in the open literature showing energy-efficient operation compared to the non-adiabatic designs (conventional CMOS designs), the challenge comes from the fact that, there exist divided opinions on the adiabatic logic family as to which is the most energy efficient and constitutes an appropriate trade-off between energy efficiency, computation time, circuit complexity, The term computation time refers to the time taken by the circuit for executing one complete computation, which can be multiplication, division, addition, subtraction, square rooting, modulo operations and many more. Power-Clock Generator (PCG) complexity or power-clocking scheme complexity. Additionally, for making the decision several parameters need to be considered such as the impact of the adiabatic load as well as adiabatic logic families on the PCG, the impact of the power-clocking scheme on the computation time of the adiabatic system and on the energy dissipation of the PCG. Therefore, investigating an energy-efficient adiabatic logic working with different power-clocking schemes and finding the appropriate trade-offs between these is one of the aims of this thesis.

Another concern for the implementation of a large adiabatic system, which arises due to the complexity of the power-clocking scheme, is the lengthy design, validation and debugging times when simulating at the circuit level. This gives rise to the need for a specific rapid modelling approach such as the use of a Hardware Description Language (HDL) that can be used to depict the behavioural and functional aspect of the adiabatic logic at a higher abstraction level before the simulations at the transistor level are performed for energy measurements. The functional errors at this level can easily be detected and corrected, decreasing the overall time in design and verification significantly. Existing HDL models mostly represent the functionality aspect of the adiabatic logic gates rather than their precise behaviour which is associated with the adiabatic implementation [26]-[28]. Existing approaches use square waveform that makes the modelling for adiabatic systems the same as that for the conventional CMOS

systems. In reality, the power-clock of the adiabatic system is a trapezoidal waveform. Due to this, the behaviour of the adiabatic logic depicted by transistor level simulations (SPICE simulations) will not match that of its HDL models. However, modelling the behaviour of adiabatic logic is challenging due to the difficulty of modelling the trapezoidal power-clock. Therefore, in this thesis, an HDL-based modelling approach describing the behaviour of the 4-phase adiabatic logic technique is also developed for functional simulation. The proposed modelling method demonstrates a systematic approach for precise timing analysis and is an improvement over the previously described approaches distinctly. Additionally, capturing the exact timing errors and detecting invalid inputs and circuit operation.

## 1.2 Scope and Objectives of the Research

The objective is to exploit the energy-efficient traits of the Adiabatic Logic Technique for the implementation of the energy-efficient NFC passive systems.

- Study and understand the 4 parts of ISO 14443 and ECMA 340 standards and identify the most "power-hungry" parts of the digital processing unit (DPU) in NFC passive systems.
- Adiabatic Logic technique has proven its energy efficient traits however, which logic family presents the best trade-offs in terms of energy, area, latency and Power-Clock Generator complexity needs to be investigated.
- Designing the power-hungry sub-modules of the digital controller unit using various multi-phase (single-phase, 2-phase and 4-phase) adiabatic logic families in order to establish the trade-offs.
- Design with adiabatic logic specifically 4-phase is non-trivial and time consuming. This demands for the development of a new modelling technique (HDL) for the implementation of the adiabatic circuits for easy and fast verification.

## 1.3 Motivation

In order to tackle the need for ultra-low power operation in NFC passive tags, it is of utmost importance to the first concentrate on investigating the most energy efficient adiabatic logic family from the existing multi-phase adiabatic logic families based on

the energy efficiency, computation time and the complexity of the power-clocking scheme. Secondly, the energy efficiency of the complete adiabatic system is often degraded due to the high energy dissipation of the power-clock generator; therefore, finding the adiabatic family which delivers energy efficient operation in an adiabatic system including the Power-Clock Generator is one of the objectives of this thesis.

In a passive NFC system, when multiple passive tags are present within the working range of the reader, they transmit the data at the same time. This causes the tags collision problem leading to increased authentication time and energy consumption. Manchester coding makes possible of detecting collision bitwise. Similarly, whenever the data is transferred from the tag to the reader, Cyclic Redundancy Code (CRC) value is calculated and is appended at the end of the data stream to detect an error in the transmitted data. Thus, implementing Manchester coding and CRC as per ISO 14443-3A standard [13] using adiabatic logic technique is one of the main aims of this thesis.

Additionally, a Hardware Description Language (HDL) model that can depict the functional and behavioural aspect of the adiabatic logic as accurately as depicted by the transistor level design (SPICE simulation) is required to reduce the design and validation time in large adiabatic systems. Therefore, another goal of this thesis is to identify the shortcomings of the existing HDL models and develop a new model as an improvement over the existing models.

## 1.4 Original Contributions to Knowledge

It is believed that this work will contribute to the international academic research on adiabatic logic and its application in energy-efficient passive NFC systems. Specifically, the work done on the VHDL-based modelling will reduce the design time of the 4-phase adiabatic systems. Additionally, it helps realising (through VHDL modelling) the system which includes both the adiabatic and the conventional CMOS implementation in the same system.

The original contributions that this project has so far added to the state-of-the-art can be summed up as follows:

1. Any sequential design would require flip-flops as a memory element. To design adiabatic flip-flops with reset, resettable adiabatic buffers are required. Existing resettable flip-flops, however, are based on the 2:1 multiplexer's (MUX).

a. The proposition of novel single-phase and 2-phase resettable buffers for the flip-flop designs using; Clocked Adiabatic Logic (CAL) and Complementary Pass-transistor Adiabatic Logic (CPAL). Prior to this, the resettable flip-flops were based on 2:1 MUXs, used as one of the resettable stages. As a result, having an increased number of transistors and an extra input terminal causing energy, routing, latency, and area overhead. The work is described in Chapter 3 of this thesis and is published in the proceedings of PRIME 2016 [29].
b. The proposition of novel 4-phase resettable buffer circuits for the flip-flop design using three adiabatic logic families namely; Improved Efficient Charge Recovery Logic (IECRL), Positive Feedback Adiabatic Logic (PFAL) and Efficient Adiabatic Charge Recovery Logic (EACRL). In addition, using the proposed three resettable adiabatic flip-flops, 2-bit twisted ring counters were implemented as design examples. The resettable counter shows a maximum increment in energy consumption of 5% compared to the non-resettable counter. This work is described in Chapter 3 and is published in the proceedings of PRIME 2016 [29].
c. The design, implementation, and layout of the existing and the proposed resettable flip-flops based on the different power-clocking schemes using all the five (multi-phase) adiabatic logic families to act as a proof of concept. Compared to the existing resettable flip-flops, the proposed resettable flip-flops using; PFAL, IECRL, EACRL, and CAL show an improvement in energy consumption of approximately 14%, 3%, 10%, and 3% respectively. However, the existing resettable flip-flops implemented using CPAL shows 0.5% less energy consumption compared to the proposed resettable flip-flops. The work is described in Chapter 3 of this thesis and is published in the proceedings of ECCTD 2017 [30].

2. The trade-offs between adiabatic logic families working on single-phase, 2-phase and 4-phase power-clocking scheme in terms of energy, complexity, latency, and area are proposed. Thus, enabling the designers and researchers to use quantitative information in selecting the required power-clocking scheme and adiabatic logic families.
    a. The design and implementation of 3-bit Up-Down counter using multi-phase adiabatic logic for establishing systematic and appropriate performance trade-off in terms of complexity, energy, latency, and area. Based on the simulation results,

4-phase adiabatic logic namely; PFAL shows better performance compared to the other adiabatic logic families. This work is also described in Chapter 3 and is published in the proceedings of ECCTD 2017 [30].

3. Cyclic Redundancy Check (CRC) is one of the main components used in passive NFC systems, whenever the data is transmitted. Therefore, performance trade-offs including robustness under Process-Voltage-Temperature (PVT) variations and supply voltage scaling between multi-phase adiabatic logic families in a large adiabatic system are worthy of investigation.
    a. The 16-bit CRC was implemented as deployed in an application for the passive NFC system, using multi-phase adiabatic logic families. A generic methodology and strategy for the design of multi-phase adiabatic CRC employing single-phase, 2-phase or 4-phase power-clocking scheme was proposed. The bit-serial CRC design is modified by incorporating more functionality allowing for the use of any CRC-16 generator polynomial and any initial load values. This work is described in Chapter 4 and is published in the Elsevier Journal, Integration, The VLSI Journal [31].
    b. Impact of voltage scaling and Process Voltage Temperature (PVT) variations on multi-phase adiabatic implementations were investigated for TSMC 180nm CMOS process at 1.8V supply voltage. It was discovered that the benefit of using adiabatic logic deteriorates for supply voltages scaled less than 1.2V. Therefore, an optimal range for the supply voltage scaling was proposed for better Energy Saving Factor (ESP) and correct functionality. This work is described in Chapter 4 and is published in the Elsevier Journal, Integration, The VLSI Journal [31].
    c. When the energy dissipation of the total system comprising of the power-clock generator was considered, it was discovered that the total energy of the system employing single-phase and 2-phase adiabatic logic was approximately 3x and 2x times respectively more when compared to the 4-phase adiabatic system. Moreover, IECRL system shows the least energy consumption followed by PFAL. This work is described in Chapter 4 of this thesis and is published in Elsevier Journal, Integration, The VLSI Journal [31].

4. VHDL modelling of the Adiabatic Logic.

a. To overcome the synchronization problem arising due to the complexity of the 4-phase power-clocking scheme to reduce the design, validation and debugging time, a new method for modelling 4-phase adiabatic logic in VHDL was proposed. Shortcomings of the existing (Hardware Description Language (HDL) modelling approaches were also identified. A very close to the exact behaviour of the trapezoidal power-clock was represented by presenting all the four periods distinctively using VHDL. The verification and applicability of the modelling were done using a 2-bit ring counter and a 3-bit Up-Down Counter. This work is described in Chapter 5 of this thesis and is published in the proceedings of PATMOS 2018 [32] and the extended version of this work is in the special issue of the Elsevier Journal, Integration, the VLSI Journal [33].
b. The proposed modelling is easy and can be used for designing large complex adiabatic system, eventually reducing the amount of time needed for the design and validation of such systems. The VHDL code of the NOT/BUF gate is further enhanced by incorporating an invalid condition check in cascade logic designs. Additionally, the gate level adiabatic modelling of the primitive AND and OR gates were also done. The enhanced proposed modelling demonstrates the error of using a square waveform as a power-clock in acquiring precise timing. A more complex circuit that of a 16-bit CRC is used to show the robustness of the proposed VHDL-based modelling approach for the 4-phase adiabatic logic technique for functional and timing simulation. This work is described in Chapter 5and is also submitted to IEEE Trans. On Circuits and Systems-I [34].

5. A novel method of Manchester encoding using the adiabatic logic technique for energy minimization is proposed. First, the time period of the data bit stream is doubled such that each bit in the data bit stream occurs twice consecutively. This way the mirror image of the actual data bit stream is generated. Then the flipping of the mirror bits takes place which generates the Manchester coded bit stream ready to be sent to the reader. The adiabatic implementation is advantageous as no separate clock needs to be added to the data stream. In fact, as the input has the same frequency as that of the power-clock, the power-clock and the data can easily be recovered at the reader from the Manchester coded data stream. This work is described in Chapter 6 of this thesis and is submitted to DATE 2019 [35].

## 1.5 Research Methodology

The methodology adopted for the design of an energy efficient adiabatic implementation of the passive NFC system is briefly described below;

First, in order to use the adiabatic logic technique for designing an energy efficient passive NFC system, there is a case for examining the various adiabatic logic families. In particular, the adiabatic logic families working on multi-phase power-clocking schemes such as single-phase, 2-phase and 4-phase targeted to low energy consumption system design. Additionally, investigating the trade-offs amongst multi-phase adiabatic logic families in terms of the computation time, area and complexity.

Second, the power-clock generator in adiabatic logic is equally important as the adiabatic logic used to undertake computations and the energy benefits and performance trade-off obtained when using single-phase, 2-phase and 4-phase adiabatic systems including PCG were also established.

Third, in tandem with the establishment of trade-offs between adiabatic logic families working with different power-clocking schemes, design, and development of CRC and Manchester encoding compatible to the ISO/IEC 14443Acommunication protocol for NFC application was implemented as a necessary milestone towards the energy efficient adiabatic implementation of the passive NFC system.

Fourth, due to the complexity of synchronization in a large 4-phase adiabatic system, debugging of errors becomes difficult, thus, increasing the overall verification time. Therefore, a method for modelling adiabatic logic circuits using an industry standard hardware description language (VHDL) was carried out.

Finally, from the research work carried out, the applicability of the adiabatic logic technique for the energy efficient applications to the passive NFC system has been demonstrated.

All the work was carried out at Applied DSP and VLSI Research Group (ADVRG) of the Department of Engineering. The research group's Cadence EDA tools made it possible to simulate, model and analyze the proposed circuits.

## 1.6 Thesis Structure

Chapter 1 is the introduction chapter.

Chapter 2 presents the background on adiabatic switching principle, multi-phase power-clocking scheme and a detailed background of the adiabatic logic families working with a multi-phase power-clocking scheme.

Chapter 3 looks at five of the most energy-efficient multi-phase adiabatic logic families and explores their potential and performance trade-offs in terms of energy, throughput (computation time), complexity and area. Novel resettable adiabatic buffers for the five chosen adiabatic logic families are proposed for their application to counters and Cyclic Redundancy Check (CRC) circuits. Adiabatic flip-flops and 2-bit twisted ring counters were designed to evaluate and compare the energy efficiency of the proposed resettable buffers with the existing resettable designs [36], [37]. Additionally, a 3-bit Up-Down Counter using the five adiabatic families were designed to evaluate the performance trade-offs between these adiabatic logic families working with different power-clocking schemes.

Chapter 4 introduces the NFC protocol as depicted in the ISO/IEC standard 14443 [11]-[13] and ECMA 340 [84] for contactless cards. The design of a 16-bit CRC using the chosen adiabatic logic families discussed in chapter 2 is presented. The performance trade-offs between energy, computation time, area, power-clocking scheme, robustness under PVT variations and supply voltage scaling is investigated in this chapter. A methodology is proposed to minimize the design time and synchronization issue by implementing a CRC design which is suitable for a range of adiabatic power-clocking strategies, specifically 4-phase, 2-phase and single-phase. Additionally, the CRC design can be scaled up or down by adding or removing the CRC slices in the datapath and flip-flops in the register unit for an application other than the NFC.

Chapter 5 presents the VHDL (Very High-Speed Integrated Circuit (VHSIC) Hardware Descriptive Language) modelling of 4-phase adiabatic logic circuits in a realistic fashion. The shortcomings of the existing modelling approaches are presented, and the proposed modelling is discussed and exposed. The functional aspects of the models are verified for a variety of gates, counters and CRC designs as reported in Chapters 3 and 4. Moreover, the models are designed such that, precise timing of the computation in an

adiabatic system can be determined. The functional errors at this level of abstraction (behavioural) can be easily detected and corrected. Therefore, decreasing the overall time in the design, debugging and verification of the functionality of a complex adiabatic system before finally verifying the circuit operation using the transistor level SPICE simulation.

Chapter 6 presents a novel method of Manchester encoding using the adiabatic logic technique for energy minimization. A brief discussion of Manchester encoding followed by the design and hardware implementation using adiabatic logic technique is presented. Based on the performance trade-offs of Chapter 4 of this thesis, the proposed design was implemented using two adiabatic logic families namely; Positive Feedback Adiabatic Logic (PFAL) and Improved Efficient Charge Recovery Logic (IECRL) which are compared in terms of energy for the range of frequency variation. Furthermore, to confirm that the power-clock generator energy consumption depends on the adiabatic logic family, the energy comparison was measured including the power-clock generator designed using 2-stepwise charging circuit (SWC) and the FSM controller.

Chapter 7 presents the conclusions drawn from this research and proposes future research directions.

## 1.7 List of Publications

[SM1] **Sachin Maheshwari**, V. A. Bartlett and IzzetKale, "4-phase resettable quasi-adiabatic flip-flops and sequential circuit design," *12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, Lisbon, Portugal, pp. 1-4, June 2016.

[SM2] **Sachin Maheshwari**, V. A. Bartlett and Izzet Kale, "Adiabatic flip-flops and sequential circuit design using novel resettable adiabatic buffers," *23rd European Conference on Circuit Theory and Design (ECCTD)*, Catania, Italy, pp. 1-4, September 2017.

[SM3] **Sachin Maheshwari**, V.A.Bartlett and Izzet Kale "Energy Efficient Implementation of Multi-phase Quasi-Adiabatic Cyclic Redundancy Check in near field communication" *Integration, The VLSI Journal, Elsevier*, vol. 62, pp. 341-352, 2018. doi.org/10.1016/j.vlsi.2018.04.002

[SM4] **Sachin Maheshwari**, V.A.Bartlett and Izzet Kale "VHDL-based Modelling Approach for the Digital Simulation of 4-phase Adiabatic Logic Design" *28th International Symposium on Power and Timing Modelling, Optimization and Simulation (PATMOS)*, Costa Brava, Spain, pp. 111-117, July 2018. **(Amongst top 10 papers)**

[SM5] **Sachin Maheshwari**, V.A.Bartlett and Izzet Kale "Modelling, Simulation, and Verification of 4-phase Adiabatic Logic Design: A VHDL-Based Approach" *Integration, The VLSI Journal, Elsevier, 2019*. **(Available online)** doi.org/10.1016/j.vlsi.2019.01.007

[SM6] **Sachin Maheshwari** and Izzet Kale "Adiabatic Implementation of Manchester Encoding for Passive NFC System" *Design, Automation, and Test in Europe (DATE'19)*, Florence, Italy, pp. 1615-1618, March 2019.

[SM7] **Sachin Maheshwari**, V.A.Bartlett and Izzet Kale "VHDL-based Modelling Approach for Functional Simulation and Verification of Adiabatic Circuits" *IEEE Trans. on Circuits and Systems-I*, 2019. **(Revision Submitted)**

[SM8] **Sachin Maheshwari** and Izzet Kale "Impact of Adiabatic Logic Families on the Power-Clock Generator Energy Efficiency" *15th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, Switzerland, July 2019. **(Accepted)**

# 2 Adiabatic Logic Families and Techniques

Over the past 25 years, many energy-efficient fully-adiabatic or quasi-adiabatic logic families have been proposed as an alternative for low power circuit technique where speed is of secondary concern [15]-[25]. Though this approach has been in existence for more than two decades, its full potential remains unexplored. In this chapter, the basic principle of adiabatic switching, multi-phase power-clocking scheme, loss mechanisms and history of quasi-adiabatic logic families are discussed. To maintain the focus of the thesis, only quasi-adiabatic logic families driven by single-phase, 2-phase and 4-phase power-clocking schemes are discussed. Based on the survey of the quasi-adiabatic logic families, the five most energy efficient quasi-adiabatic logic families working on the multi-phase (single-phase, 2-phase and 4-phase) power-clocking schemes are chosen which forms the foundation of the research carried out in this thesis.

## 2.1 Introduction

Due to the increased usage of battery-less applications (e.g. smartcards) and rising energy density due to the technology shrinkage, energy-efficiency has become a major concern in the design of large systems. In the simplest conventional CMOS logic, an inverter is shown in Figure 2.1, the load capacitance ($C_L$) gets charged through the MOS transistor (P1), the output node voltage rises from 0 to supply voltage ($V_{DD}$), and $C_L V_{DD}^2$ amount of energy is supplied from the power supply [15]. Half of this energy gets dissipated in the pMOS transistor and the other half gets stored on the output load capacitance. During the high-to-low transition, the output capacitance starts discharging and the stored energy will be dissipated in the nMOS transistor (N1). So, every time when the output node discharges, it losses $\frac{1}{2} C_L V_{DD}^2$ amount of energy [15]. This loss of

energy to heat during charging and discharging happens because the transitions are abrupt: the transistor is turned on and current flows through the transistor's resistive channel to charge or discharge the capacitor. Since the energy drawn from the power supply depends on the rate at which the charges are drawn from the source, hence lowering the rate by slowly charging the output load capacitance through slowly changing AC power-clock rather than a DC, will result in less energy dissipation.

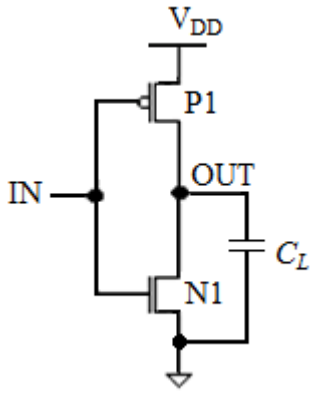



**Figure 2.1:** CMOS inverter.

One of the pioneering works was done by Teichmann [38], where the author discusses the adiabatic logic design issues, designed various arithmetic circuits, and implemented an adiabatic CORDIC-based DCT as a test vehicle to demonstrate the system-level applicability of adiabatic logic for ultra-low-power digital signal processing [39]. In [40] the behaviour of adiabatic logic circuits in weak inversion or the sub-threshold regime was analysed for a 22nm CMOS technology. Through extensive post-layout simulation, it was demonstrated that the sub-threshold adiabatic circuits can save significant energy compared with an equivalent non-adiabatic implementation. Over the years, adiabatic logic techniques have also found their applications in energy efficient power-analysis attack resilient logic designs [41]-[45]. It has been shown that by careful design and exploiting charge-sharing between the two output nodes in adiabatic logic, during the idle phase of the power clock, the circuit can be made secure for cryptographic applications [41]-[45]. Moreover, recent work has demonstrated the applicability of the adiabatic principle to adiabatic capacitive logic demonstrating the effectiveness of the technique to achieve ideally zero-power logic dissipation [46], [47].

## 2.2 Adiabatic Switching Principle

"Adiabatic" is a term of Greek origin that has spent most of its history associated with classical thermodynamics [28]. It refers to a system in which a transition occurs without energy (usually in the form of heat) being either lost to or gained from the system. In the context of electronic systems, rather than heat, the electronic charge is preserved. Thus, the adiabatic circuits would operate ideally with zero dissipation that may be approached as the logic switching is slowed down. Decreased energy consumption with increased ramping time ($T_r$) is, therefore, the defining property of adiabatic switching [24], [25].In addition, this slowly charging process gives an additional advantage of pumping the stored energy back to the power supply during the discharging process which can be recovered using an AC power-clock generator [48]-[54]. However, in order to have less dissipation, all the nodes should share the same principle of charging and discharging. The principles include; 1) never turn the switch ON when there is any potential across it; 2) never turn the switch OFF when there is current flowing through it [24]. These two rules are observed to reduce the energy dissipation by making sure that current surges do not occur and are avoided by design. In practice, these rules are applied by charging the output load capacitance of the circuit using a slowly ramp-like power-clock called trapezoidal waveform, changing from 0 to $V_{DD}$ and back to 0 and maintaining four equal Power-Clock periods called *Evaluation* (E), *Hold* (H), *Recovery* (R) and *Idle* (I).

Figure 2.2 shows the Power-Clock (PC) and the input signal, IN_H. To follow the adiabatic switching principle, IN_H should be stable when PC is evaluating. Furthermore, IN_H should start ramping down when the PC is stable (Hold). By observing these, adiabatic principles are followed, and lower energy dissipation is achieved.

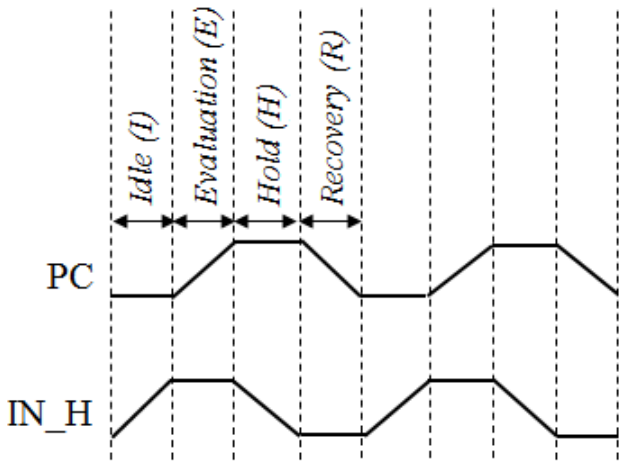



**Figure 2.2:** Relation between Power-clock (PC) and the input signal (IN_H).

Figure 2.3 shows a simple setup of the transient response using a ramp for the series *RC* circuit. $C_L$ is the load capacitance, $R$ is the resistance of the charging path, $V_{ramp}$ is the voltage source changing from 0 to $V_{DD}$ and $T_r$ is the time taken to charge the load capacitance, $C_L$. Following this, the charge that will be delivered is $Q=C_LV_{DD}$, the current drawn from the voltage source is $I = \frac{Q}{T_r}$, and the energy dissipated over time, $T_r$ in the charging path will be,

$$E_{diss} = PT_r = VIT_r = I^2RT_r$$

$$E_{diss} = \left(\frac{Q}{T_r}\right)^2 RT_r = \frac{Q^2R}{T_r}$$

$$E_{diss} = \frac{RC_L}{T_r}\left(C_LV_{DD}^2\right) \tag{2.1}$$

Likewise, the same amount of energy will be dissipated during the discharging process. Thus, the energy dissipated (Adiabatic Loss) over one cycle will be the total energy dissipated in the charging and discharging of the load capacitance.

$$E_{AL} = \frac{2RC_L}{T_r}\left(C_LV_{DD}^2\right) \tag{2.2}$$

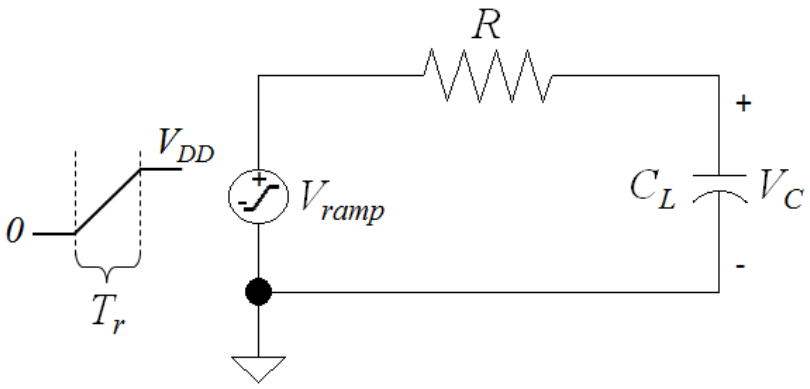



**Figure 2.3:** A simple RC series circuit.

An exact analysis of the energy dissipation can be found in [24]. From equations (2.1) and (2.2) it can be inferred that energy dissipation can be reduced, ideally to zero by choosing $T_r >> RC$, but at the expense of increased operational time. At the opposite extreme, when $T_r << RC$ the dissipation approaches that of the conventional CMOS with constant supply voltage.

On the other hand, in the conventional CMOS circuits, when $T_r$ is very small or abrupt charging and discharging, compare to the time constant of the circuit ($RC_L$), the current, $i$ in the circuit is:

$$i = C_L \frac{dV_C}{dt}$$

The voltage across the resistor will be

$$V_R = iR = RC_L \frac{dV_C}{dt}$$

From the Kirchhoff's voltage law, $V_{DD}$ equals the sum of the capacitor voltage ($V_c$) and resistor voltage ($V_R$).

$$V_{DD} = V_C + RC_L \frac{dV_C}{dt}$$

The voltage across the capacitor will be:

$$V_C = V_{DD}\left(1 - e^{\frac{-t}{RC_L}}\right) \tag{2.3}$$

And the current is given by

$$i = \frac{V_{DD}}{R}\left(1 - e^{\frac{-t}{RC_L}}\right) \tag{2.4}$$

When the load capacitor is charged through a resistance, the energy is lost in the form of heat in the resistor which is termed as non-adiabatic energy and is given by:

$$E_{NA} = \int_0^{\infty} i^2 R T_r = \frac{1}{2} C_L V_{DD}^2 \tag{2.5}$$

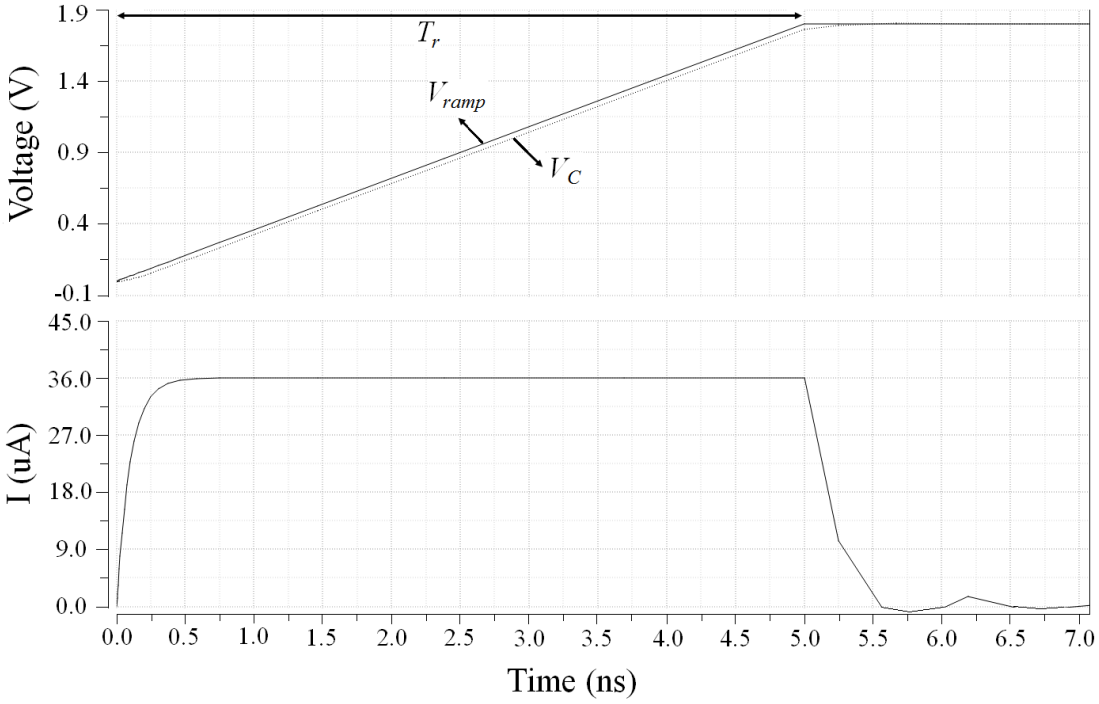
The energy stored on the load capacitor, $C_L$ is given by:

$$E_{stored} = \int_0^{V_{DD}} i V_C dt = \int_0^{V_{DD}} \left(C_L \frac{dV_C}{dt}\right) V_C dt$$

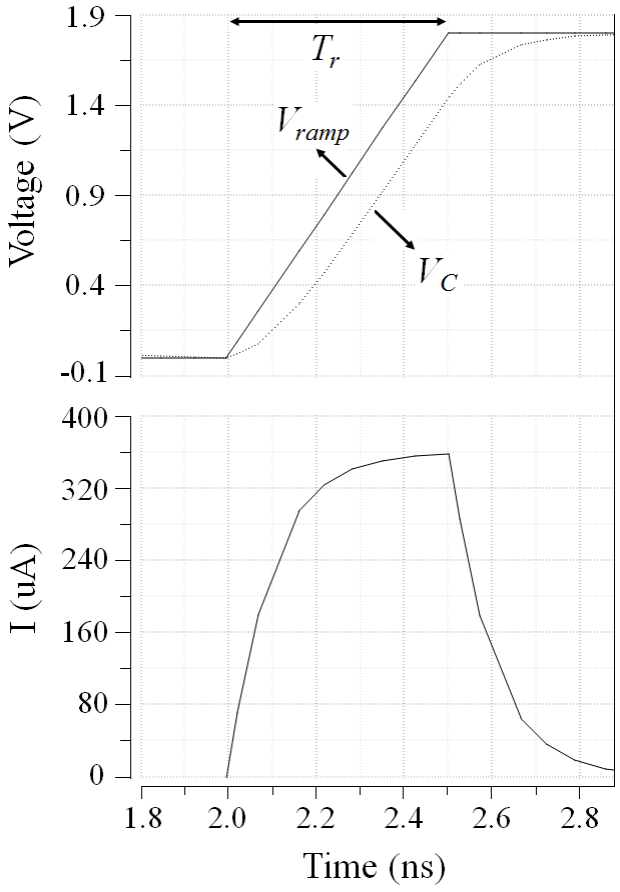
$$E_{stored} = \int_0^{V_{DD}} C_L V_C dV_C = \frac{1}{2} C_L V_{DD}^2 \tag{2.6}$$

In the conventional charging, the energy dissipated depends on the load capacitor and the supply voltage, however; in the adiabatic switching the energy dissipated is proportional to $R$. Therefore if the resistance of the charging path is decreased, the energy dissipation also decreases.

Figure 2.4 (a) and (b) show the voltage curves and peak current graphs for longer and shorter ramping time using a voltage ramp respectively. If the ramping time of the ramp voltage, $V_{ramp}$, is longer and higher than the $RC$ time constant of the circuit then the voltage, $V_C$ will track the ramp voltage with only a small dissipation as given by (2.1) and will result in smaller and constant peak current. On the other hand, for the shorter ramping times of the ramp voltage, the voltage, $V_C$ will lag the ramp voltage and will reach the supply voltage $V_{DD}$ in a characteristic exponential decay curve. The current graph in Figure 2.4 (b) is a typical exponential charging current graph for a conventional $RC$ step response whose peak current is 10 times higher than the peak current value of Figure 2.4 (a).




(a)




(b)

**Figure 2.4:**(a) Longer Ramping Time (b) Shorter (steeper) Ramping Time.

The same charging technique can also be used to discharge the logic from $V_{DD}$ back to 0. The following section discusses the power-clock requirement for multi-phase adiabatic logic families.

## 2.3 Multi-Phase Power-Clocking Scheme

Unlike static CMOS logic, due to being clock-powered, the adiabatic logic families derived from static Differential Cascode Voltage Switch (DCVSL) requires a separate power-clock supply. Depending on the adiabatic logic families, power-clocks are generated either using an inductor-based resonant circuit [48] or using a capacitor-based Step Wise Charging (SWC) circuit [49]-[54]. The SWC based power-clock generator proposed in [54] has been used in the adiabatic system simulations presented in Chapter 4 and 6 of this thesis.

Figure 2.5 shows the single-phase, 2-phase and 4-phase power-clocking schemes for multi-phase adiabatic logic families. The single-phase and 4-phase power-clocking schemes can be broken down into four equal periods namely *Evaluation (E), Hold (H), Recovery (R)* and *Idle (I)*. Whereas, 2-phase power-clock have an idle period 3 times larger than each of the evaluation, hold and recovery periods. The latency, $T_{Latency}$, for all the three power-clocking schemes is defined as the minimum time required for the first input to process to the output.

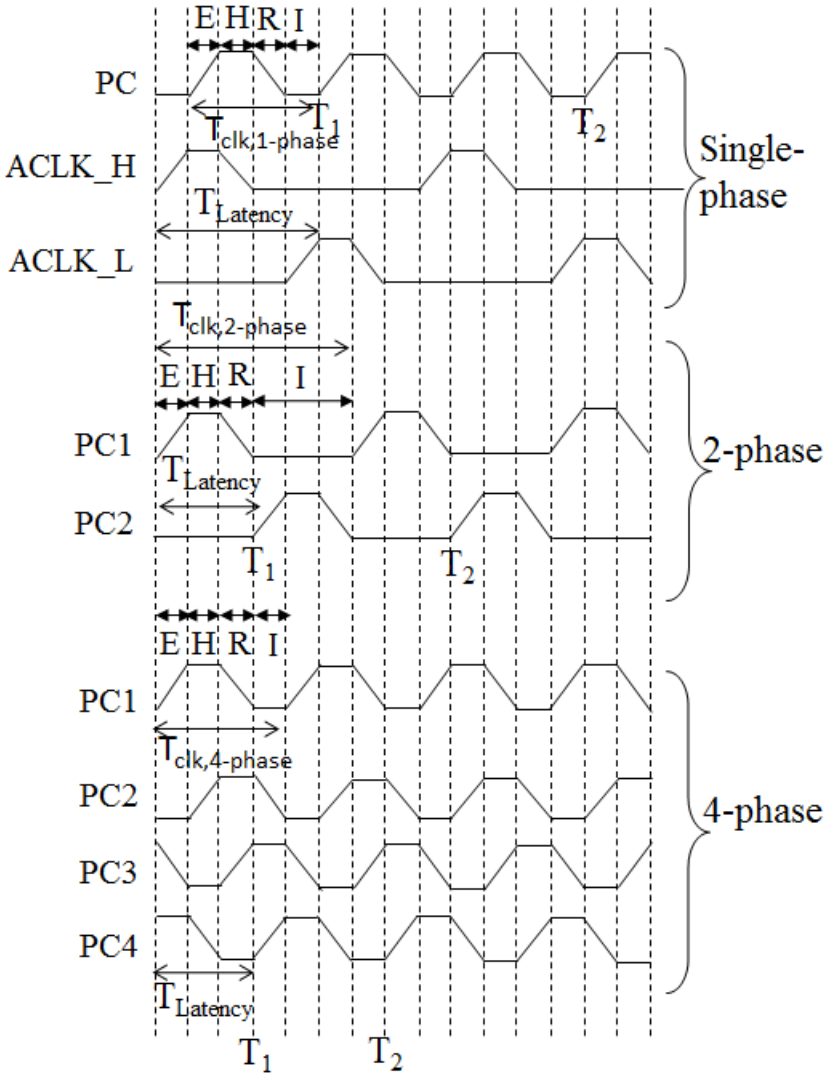



**Figure 2.5:** Comparison of single-phase, 2-phase and 4-phase power-clocking scheme.

For single-phase, power-clocking scheme signals ACLK_H and ACLK_L are used and are called the auxiliary clocks. PC is the Power-Clock. $T_{clk,1\text{-}phase}$ is the duration of one power-clock phase of the single-phase power-clocking scheme. As shown in Figure 2.5, the first output of a single-phase cascaded logic is available at time $T_1$and the second output is available at time $T_2$, which is after two power-clock phases. Thus, the output of the cascaded single-phase based sequential logic remains valid for two consecutive power-clock phases, resulting in a lower throughput and a higher latency.

Similarly, for 2-phase power-clocking scheme, PC1 and PC2 are the two phases of the power-clock. $T_{clk,2\text{-}phase}$ is the duration of one power-clock phase of the 2-phase power-clocking scheme. In the case of 2-phase power-clocking scheme, due to its longer idle

period compared to single-phase and 4-phase power-clocks, also results in lower throughput and higher latency for sequential circuits. PC1, PC2, PC3, and PC4 are the 4 phases of the 4-phase power-clocking scheme. $T_{clk,4\text{-}phase}$ is the duration of one power-clock phase of the 4-phase power-clocking scheme. Adiabatic logic using a 4-phase power-clocking scheme gives higher throughput but leads to more complex power-clock requirement compared to the single-phase and 2-phase power-clocking schemes. However, it is obvious that as the number of phase's increase the throughput and latency improves and complexity increases but its energy and area is dependent on the adiabatic logic families used. Thus, at this point, it is hard to conclude that out of the multi-phase adiabatic logic which one is the most energy efficient and has better performance in terms of energy, area, throughput, latency, robustness against PVT variations and power-clock generator complexity. However, this will be investigated and discussed in the rest of the thesis in chapters 3, 4, and 6 respectively.

## 2.4 Losses in Adiabatic Logic

In an ideal adiabatic system, losses are governed by (2.2) and are recognized as Adiabatic Loss (AL). Nevertheless, due to the shrinking device geometries into the sub-µm regime and the presence of a threshold voltage drop, $V_{th}$ drop in transistors lead to additional losses. These additional losses can govern and exhibit a lower bound for energy dissipation in an adiabatic system. For instance, due to the shrinking device geometries, leakage currents can dominate the overall energy dissipation. One of the dominant leakage currents is the so-called sub-threshold current which is expressed as [38]:

$$I_D = I_{D0} e^{\frac{V_{GS}-V_{th}}{\eta V_T}} \left(1 - e^{\frac{-V_{DS}}{V_T}}\right) \tag{2.7}$$

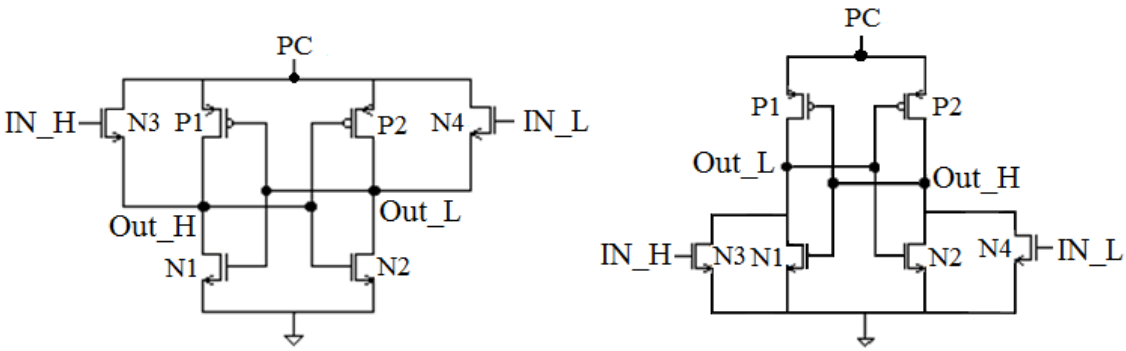
Where $I_{D0} = \frac{W \mu_0 C_{ox} V_T^2 e^{1.8}}{L}$, $V_T$ is the thermal voltage, $V_{th,}$ is the threshold voltage of the device, $V_{GS}$ and $V_{DS}$ are the gate to source and drain to source voltages, $W$ and $L$ are the effective transistor width and length, respectively. $C_{ox}$ is the gate oxide capacitance; $\mu_0$ is the carrier mobility and $\eta$ is the subthreshold swing coefficient.

For the condition when $V_{DS}$ is zero, no leakage current will flow. However, the leakage current will increase to its maximum for the values of $V_{DS}$ that are multiples of the thermal voltage. The leakage current flows from PC to ground during evaluation, hold and recovery periods and leads to dissipation of charge that cannot be recovered. All the losses due to leakage can be summarized in a mean current, $\bar{I}_{leak}$, that leads to energy consumption per cycle of:

$$E_{leak} = V_{DD}\bar{I}_{leak}\frac{1}{f} \tag{2.8}$$

Energy consumption due to leakage losses increases for lower frequencies [38]. This is because the Leakage Loss (LL) is gathered over a longer time interval (longer ramping time).

In this thesis, all the simulations have been performed using the TSMC 180nm CMOS Process, and the frequency of operation of the application is centered around 13.56MHz, therefore, leakage losses will not impact the energy dissipation significantly and thus, are not dealt with.




(a) (b)

**Figure 2.6**: NOT/BUF gate using (a) PFAL [55] (b) IECRL [22].

The NOT/BUF gate using Positive Feedback Adiabatic Logic, PFAL [55], and Improved Efficient Charge Recovery Logic, IECRL [22] are shown in Figure 2.6 (a) and (b) respectively. During the recovery period of PC, in PFAL and IECRL, the charge at one of the output nodes (following PC) is recovered until PC doesn't fall below the threshold voltage of the cross-coupled pMOS transistors (P1 and P2). This leads to a

residual charge at one of the output nodes. The residual charge is either reused in the next cycle if the inputs remain the same or is discharged to the ground if different inputs arrive. Similarly, in IECRL, (where the evaluation network is connected between the output nodes and the ground) in the evaluation period, the output node cannot instantly follow the rising PC until PC reaches the threshold voltage of the cross-coupled pMOS transistors (P1 and P2). This forces the output node to follow PC abruptly, leading to energy dissipation. All these losses/dissipations are due to the threshold voltage and lead to Non-Adiabatic Losses (NAL) which is expressed as:

$$E_{NAL} = \frac{1}{2} C V_{th,p}^2 \tag{2.9}$$

It should be noted that, unlike AL and LL, NAL is independent of the frequency of operation. AL is proportional to the frequency of operation however; LL is inversely proportional to the frequency of operation (2.8). It is also worth mentioning that, an optimum operating frequency exists for adiabatic logic families, where minimum energy dissipation per cycle at a particular frequency is observed.

The three losses in the adiabatic circuits are discussed in [38]. The overall total energy dissipation ($E_{TD}$) in adiabatic logic design is obtained by summing the effects of all the three loss mechanisms and is given by (2.10). Where $E_{AL,}$ $E_{NAL,}$ and $E_{leak}$ are mentioned in equations 2.2, 2.8 and 2.9 respectively.

$$E_{TD} = E_{AL} + E_{NAL} + E_{leak} \tag{2.10}$$

## 2.5 Quasi-Adiabatic Logic Families

Over the last 25 years, a plethora of quasi-adiabatic logic families resulting in different levels of energy saving has been proposed. Quasi-adiabatic circuits are divided into diode-based and transistor-based logic designs. Due to the high energy and area consumption of the diode-based logic, only the transistor-based logic designs are chosen for investigation in this thesis. Since the transistor-based adiabatic logic families are based on DCVSL CMOS logic, all the adiabatic logic techniques have a common structure, consisting of the cross-coupled pMOS pairs powered by the power-clock and dual-rail input and output signals. A complete adiabatic logic system consists of two major component blocks, first is the logic block and other is the charge recovery block

as shown in Figure 2.7. The charge recovery block is part of the power-clock generator as it is responsible for the recovery of charge back to the power supply during the discharging/recovery phase of the power-clock.

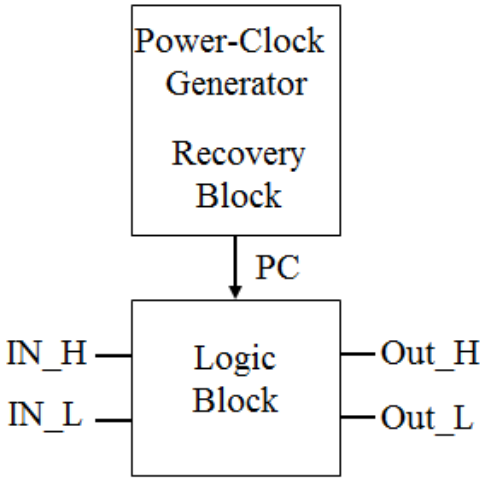



**Figure 2.7:** A basic block diagram of the adiabatic logic system.

A substantial list may not be complete, of the transistor based quasi-adiabatic logic families and their power-clocking scheme requirement is shown below in Table 2.1.

**Table 2.1:** List of transistor-based quasi-adiabatic logic families and their power-clocking schemes.

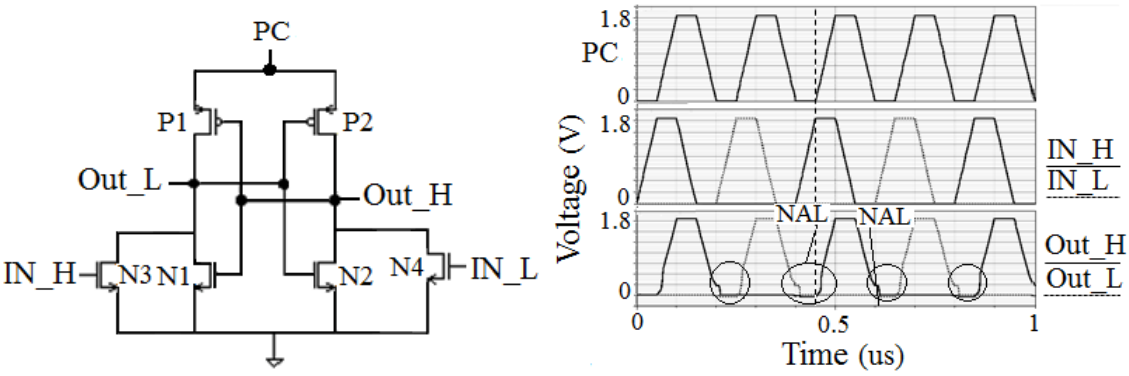
| Quasi-Adiabatic Logic Families | Power-clock |
|---|---|
| 2N2P-2N [24] | 8 |
| 2N-2N2P [25] | 4 |
| Positive Feedback Adiabatic Logic (PFAL) [55] | 4 |
| 2N-2P [56] | 4 |
| Efficient Charge Recovery Logic (ECRL) [57] | 4 |
| Clocked Adiabatic Logic (CAL) [58] | 1 |
| Complementary Adiabatic MOS Logic (CAMOS) [59] | 4 |
| Dynamic Adiabatic MOS Logic (DAMOS) [60] | 4 |
| Energy Efficient Logic (EEL) [61] | 4 |
| Pass-transistor Adiabatic Logic (PAL) [62] | 2 |
| Forward Body-bias MOS (FBMOS) [63] | 4 |
| Improved Efficient Charge Recovery Logic (IECRL) [22] | 4 |

| | |
|---|---|
| PAL-2N [64] | 4 |
| Bootstrapped NMOS Charge Recovery Logic (BNCRL) [65] | 4 |
| True Single-phase Energy recovering Logic (TSEL) [66] | 1 |
| Source Coupled Adiabatic Logic (SCAL) [67] | 1 |
| NMOS Energy Recovery Logic (NERL) [68] | 4 |
| Adiabatic Differential Cascode Voltage Switch Logic (ADCVSL) [69] | 2 |
| High Efficient energy recovery logic (HEERL) [70] | 4 |
| Efficient Adiabatic Charge Recovery Logic (EACRL) [19] | 4 |
| Improved Pass Gate Adiabatic Logic (IPGAL) [71] | 4 |
| Complementary Pass-transistor Energy Recovery Logic (CPERL) [72] | 2 |
| Complementary Pass-transistor Adiabatic Logic (CPAL) [20], [76] | 2/4 |
| Improved Positive Feedback Adiabatic Logic (IPFAL) [73] | 4 |
| Dual Transmission Gate Adiabatic Logic (DTGAL) [74] | 4 |
| Clocked Transmission Gate Adiabatic Logic (CTGAL) [75] | 2 |

Though the list in Table 2.1 does not claim to be exhaustive, it still gives a general idea of the progress of the work done on the topic of adiabatic logic techniques. Since the implementation and the distribution of multiphase power-clocking schemes require additional area, energy consumption, and increased complexity, logic families with more than 4-phases are not considered. Out of the various 4-phase adiabatic logic families reported in Table 2.1, IECRL, EACRL, and PFAL were chosen as they are considered to be the most energy efficient adiabatic logic families in the literature. However, the 4-phase power-clocking scheme complicates the design due to multiple power-clock generators. This issue can result in area, and energy overhead that could offset the advantages achieved. It should be noted that the single and 2-phase adiabatic logic designs will have less complex power-clock generator compared to that of the 4-phase adiabatic logic designs. However, due to high latency, it is not clear if the energy benefits would be obtained in comparison to 4-phase designs. Therefore, it was decided to investigate the single-phase, 2-phase and 4-phase adiabatic logic designs and to draw out the performance trade-offs between multi-phase adiabatic logic families based on energy efficiency, area, and latency/throughput and complexity. TSEL and SCAL circuits use single-phase power-clock, but the additional reference voltage and current increase the design complexity. Additionally, they are difficult to design due to the choice of reference voltage concerning the clock frequency [66], [67]. On the other

hand, because of its simple structure, CAL (Clocked Adiabatic Logic) is considered for realizing complicated circuits in the literature. In Ref. [77] it has been shown that the use of CAL for designing large adiabatic systems can save a considerable amount of energy in comparison to conventional CMOS. Thus, CAL was considered for the single-phase power-clocking scheme [58]. Many 2-phase adiabatic logic designs have been proposed in the literature however, the most energy efficient of all is CPAL due to its zero NAL at the output nodes and thus, was considered.

### *2.5.1 Improved Efficient Charge Recovery Logic (IECRL)*



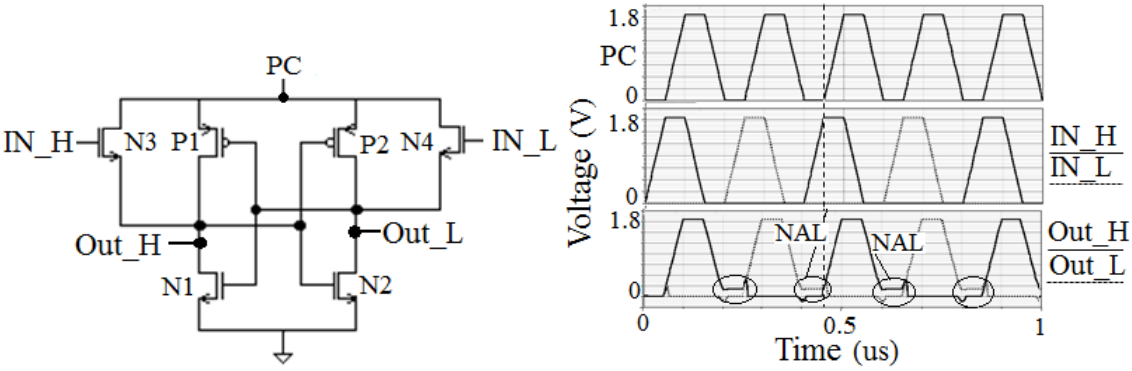

(a) (b)

**Figure 2.8:** (a) IECRL buffer [25] (b) Output Waveform.

In 1994, Denker [25] proposed a high performance improved ECRL logic family circuit shown in Figure 2.8 (a). IECRL is also called as 2N-2N2P and is an improvement over ECRL [57]. The only difference between ECRL and IECRL is that IECRL has a pair of cross-coupled nMOS transistors in addition to the cross-coupled pMOS. Thus, IECRL has cross-coupled inverters and is very similar to a standard SRAM cell. In Figure 2.8 (a), the cross-coupled inverters provide a pull-down path to ground or a non-floating node during the recovery phase, thus, reducing the coupling effect and decreasing NAL (2.9) during the recovery phase of PC. IECRL requires 4-phase power-clocks for cascaded logic and suffers from NAL during the evaluation and the recovery period of PC because of the threshold voltage degradation. During the evaluation period of the power-clock, when the input has already ramped to $V_{DD}$, the PC is still at zero voltage value (IDLE). The power-clock starts rising when the input is stable. Only when PC ramps above the threshold voltage, $|V_{th,p}|$ of the pMOS transistors, one of the output nodes starts following the power-clock causing NAL.

The basic operation and working of IECRL are described in [22]. Figure 2.8 (b) shows the operation waveforms of IERCL buffer circuit along with its NAL. Similarly, NAL occurs during the recovery period of the power-clock.

### *2.5.2 Positive Feedback Adiabatic Logic (PFAL)*

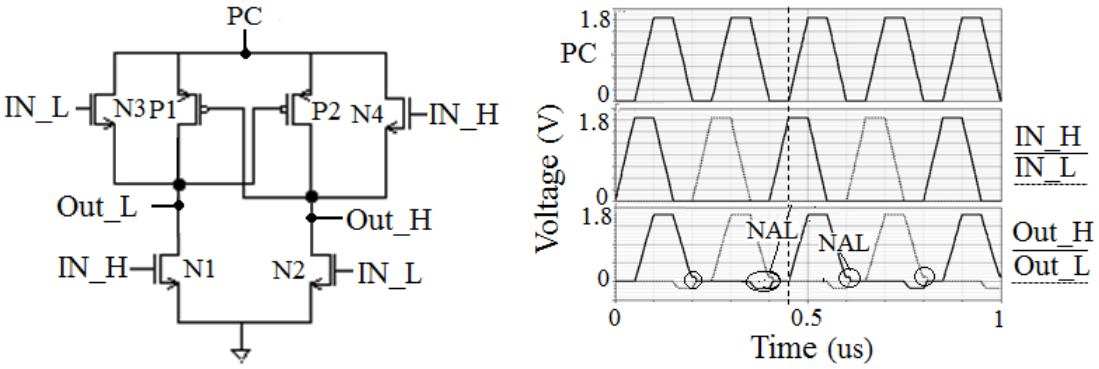



(a) (b)

**Figure 2.9:** (a) PFAL buffer [55] (b) Output Waveform.

In 1996, A. Vetuli et al. [55] proposed a new adiabatic logic family which makes use of a CMOS positive feedback amplifier. The logic is called Positive Feedback Adiabatic Logic, PFAL and requires a 4-phase power-clocking scheme for cascaded logic. PFAL is very similar to IECRL, having its evaluation tree connected between the power-clock and the output nodes. Figure 2.9 (a) shows the schematic of the PFAL buffer gate. The equivalent resistance at the two output nodes is smaller in comparison to IECRL due to the formation of transmission gate pairs between P1, N3, and P2, N4. PFAL do not suffer from NAL during the evaluation period of the power-clock because the output node follows PC through the nMOS transistors until PC is below the threshold voltage, $|V_{th,p}|$, of the pMOS transistor. However, it suffers from NAL during the recovery period when PC falls below the threshold voltage, $|V_{th,p}|$, of the pMOS transistor, leaving a residual charge on the output node. This residual charge gets discharged non-adiabatically at the start of the next cycle when the new inputs are evaluated. Figure 2.9 (b) shows the operation waveform of the PFAL buffer circuit. It can be seen that PFAL suffers from NAL only during the recovery phase of the PC only.

### 2.5.3 *Efficient Adiabatic Charge Recovery Logic (EACRL)*




(a) (b)

**Figure 2.10:** (a) EACRL buffer [19] (b) Output Waveform.

In 2001, Varga et al. [19] proposed a dual-rail energy Efficient Adiabatic Charge Recovery Logic, EACRL. This structure has an advantage over IECRL as it completely eliminates NAL during the evaluation period of PC. Similar to the IECRL logic, EACRL also requires 4-phase PC for cascaded logic. Figure 2.10 (a) shows an EACRL buffer gate. It uses a pair of cross-coupled pMOS transistors and duplicate evaluation trees; one connected between the output nodes and the ground and other connected between the PC and the output nodes. EACRL do not suffer from NAL during the evaluation period of PC. Similar to PFAL, it suffers from NAL during the recovery period when PC falls below the threshold voltage, $|V_{th,p}|$, of the pMOS transistor, leaving a charge on the output node. This charge gets discharged non-adiabatically either at the start of the next cycle when the new inputs are evaluated or is reused in the next cycle if the inputs do not change.

Figure 2.10 (b) shows the operation waveform of the EACRL buffer gate. During the recovery period of the PC, the two output nodes get coupled due to the absence of cross-coupled nMOS transistors. As a result, the output node which should be held at zero goes to a negative value. This results in the non-adiabatic dynamic loss and is called coupling effect. Figure 2.10 (b) shows NAL arising due to the threshold voltage degradation and the coupling effect.

### 2.5.4 *Complementary Pass-transistor Adiabatic Logic (CPAL)*

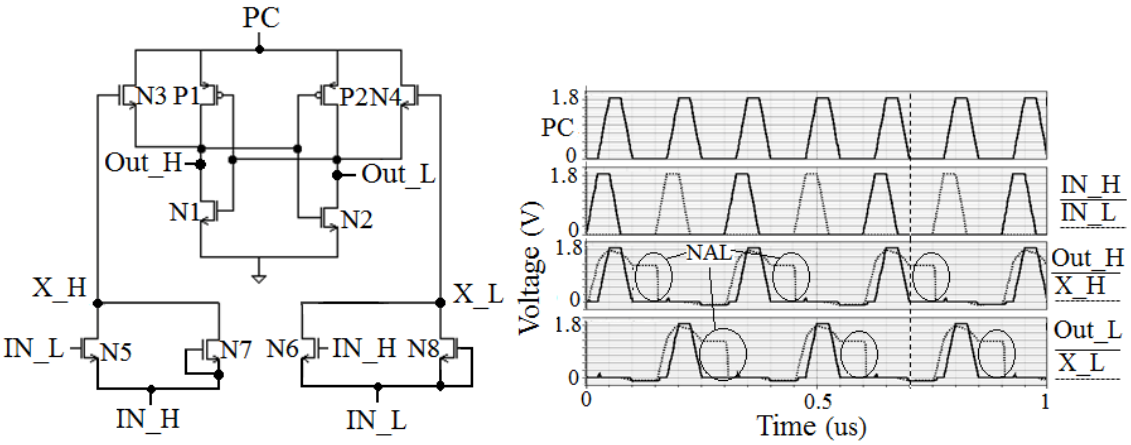



**Figure 2.11:** (a) CPAL buffer [20] (b) Output Waveform.

Pass-transistor Adiabatic Logic, PAL was first introduced by Oklobdzija and Maksimovic [62] in 1997. It uses a 2-phase power-clocking scheme for cascaded logic. The author claims that PAL outperforms the previously proposed adiabatic logic families in terms of energy consumption. The claim was based on the performance of 1600-stage PAL shift register. Then in 2003, Jaiping et al. [20] proposed a new logic family, called Complementary Pass-transistor Adiabatic Logic, CPAL which works on a 4-phase power-clocking scheme for cascaded logic. But later in 2005, the author demonstrated that the cascaded CPAL logic can be driven using a 2-phase non-overlapping power-clocking scheme [76].

Figure 2.11 (a) shows the CPAL buffer circuit which uses a PFAL buffer with the evaluation tree (N5-N8) designed using pass-transistors, connected to the gates of the nMOS pull-ups (N3-N4) also called bootstrapped transistors. Since the node X_H or X_L gets boosted, the CPAL circuit eliminates NAL at the two output nodes. The energy dissipation of the two-phase CPAL circuit includes mainly two terms: full-adiabatic energy loss (2.2) on the output nodes and non-adiabatic energy loss on internal nodes X_H (or X_L). The more detailed description of its NAL on internal nodes is analyzed in [76]. Figure 2.11 (b) shows the operation waveform of the CPAL buffer circuit and the NAL on node ‘X_H’ and ‘X_L’. An additional non-adiabatic dynamic loss occurs due to the coupling effect, where the output node (which should be at zero) goes to a negative voltage value when the PC voltage level falls below the threshold

voltage of one of the nMOS transistors N1 or N2 during the recovery phase such that it gets coupled to the node following PC.

### *2.5.5 Clocked Adiabatic Logic (CAL)*

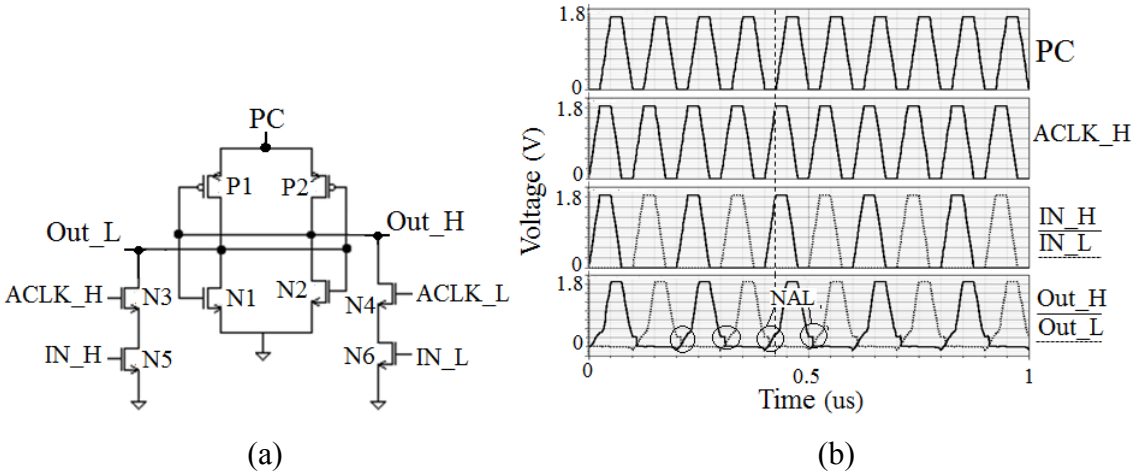



**Figure 2.12:** (a) CAL buffer [58] (b) Output Waveform.

In 1995, Maksimovic et al. [58] proposed a logic called Clocked Adiabatic Logic (CAL). It works on a single-phase power-clocking scheme. The CAL buffer gate is similar to that of 2N-2N2P but has clocked nMOS transistors (N3, N4) between the evaluation nMOS transistors (N5, N6) and the output nodes. Figure 2.12 (a) shows the schematic of the CAL buffer gate. The clocked nMOS transistors use a pair of auxiliary-clocks (ACLK_H and ACLK_L) which allows operation from a single-phase PC. A more detailed description can be found in [78]. Although this adiabatic logic family seems to have a simplified Power-Clock Generator (due to the requirement of single-phase PC) but the use of the auxiliary clock signals for cascaded logic, makes it complex and results in area and energy overhead. Figure 2.12 (b) shows the operation waveform of the CAL buffer gate. Due to the stacking of transistors at the two output nodes, it has higher NAL (compared to other adiabatic logic families discussed above) arising because of larger threshold voltage degradation.

Despite the various interesting multi-phase energy-efficient adiabatic logic families been proposed in the last 25 years, each encompassing many novel ideas and saving considerable energy compared to static CMOS, there still remains several practical implementations in the design of complex adiabatic circuits than are unexplored; i) Selection of a multi-phase adiabatic logic for an application specific design; ii) buffer

insertion for handling synchronization issue incurring area overhead; iii) latency and throughput as multiple power-clock phases require different computation time. This is because pipelining is inherent in adiabatic logic and it can only perform one logic evaluation per clock phase. Thus, every gate introduces a phase delay in propagating from input to output. This can be seen in Figure 2.5; and iv) the non-adiabatic loss compromising the energy efficiency.

## 2.6 Summary

In this chapter, the concept of the adiabatic switching principle and quasi-adiabatic logic has been introduced. Since the work on power-clock generation has been done extensively in the literature, a brief discussion of the power-clock generation through the inductor and capacitor-based circuits was introduced. The thesis deals with investigating the performance of the single, 2-phase and 4-phase adiabatic logic families. Therefore, a detailed discussion of the multi-phase power-clocking schemes for multi-phase adiabatic logic designs is presented.

In order to give the idea of the sources of energy dissipation in the adiabatic logic technique, loss mechanism is discussed. In quasi-adiabatic logic designs, primarily three losses are present. Adiabatic Losses (AL) and losses due to leakage which are called as Leakage Losses (LL) are frequency dependent. Non-Adiabatic losses (NAL) however, are not the function of frequency. The list of existing adiabatic logic families based on the multi-phase power-clocking scheme published in the last 25 years has been presented in Table 2.1. On the basis of the proven performance in the literature, five quasi-adiabatic logic families were chosen from Table 2.1. The chosen adiabatic logic families for the performance evaluation based on energy, area, computational time, and circuit and power-clock complexity are: IECRL, PFAL, EACRL, CPAL, and CAL. Out of these, IECRL, PFAL, EACRL are based on 4-phase power-clocking scheme. CPAL is based on 2-phase power-clocking scheme and CAL works with single-phase power-clocking scheme. A brief discussion of the above mentioned quasi-adiabatic logic families along with their advantages and disadvantages is presented. EACRL, for instance, requires a dual evaluation network but has zero NAL during the evaluation period of the PC and thus will be energy efficient. PFAL has reduced NAL and the equivalent resistance of the charging path however, it presents a large load to the power-clock (number of transistors connected to the PC). Similarly, CAL uses the least

complex power-clocking scheme but requires auxiliary clock for cascaded logic. Practically, it is difficult to design a high-performance system having minimum energy, latency, area, and complexity using adiabatic logic techniques, However, trade-offs can be established that enable the designer to design an application specific efficient adiabatic logic system.

# 3 Design and Evaluation of Adiabatic Resettable Buffers, Flip-flops, and Sequential Circuit Designs

In this chapter, novel resettable adiabatic buffer circuits for the design of resettable adiabatic flip-flops are proposed. The energy and area of the proposed resettable flip-flops are compared to that of the existing MUX-based and on-resettable flip-flops for each of the five chosen quasi-adiabatic logic families (discussed in Chapter 2 of this thesis). The Chapter also discusses the design of a 2-bit twisted ring and 3-bit Up-Down counter in order to extend the comparison of the five multi-phase adiabatic logic families beyond energy dissipation. The work in this chapter is based on the full-custom design with results presented for pre-layout and post-layout simulations. All the simulations were performed using the Spectre simulator in Cadence EDA tool based on TSMC 180nm CMOS process technology. The power-clock used is a trapezoidal wave, ramping from 0V to 1.8V. The transistor sizes for all the designs were set at the technology minimum ($W_n=W_p=W_{min}$=220nm, $L_n=L_p=L_{min}$=180nm).

## 3.1 Introduction

In adiabatic circuits, PC provides both the power and synchronization (clock for timing the operations of the logic gates) to each logic gate. This suggests that adiabatic circuits are implicitly pipelined where data propagate through one logic gate in each phase. A single buffer logic gate represents a latch which passes the input to the output when PC starts ramping from zero to $V_{DD.}$ Since the output follows PC, the output signal is set to zero when the PC falls from $V_{DD}$ to ground thus, a signal can never be stored. Consequently, an adiabatic D flip-flop is structured using a cascaded buffer chain. An n-

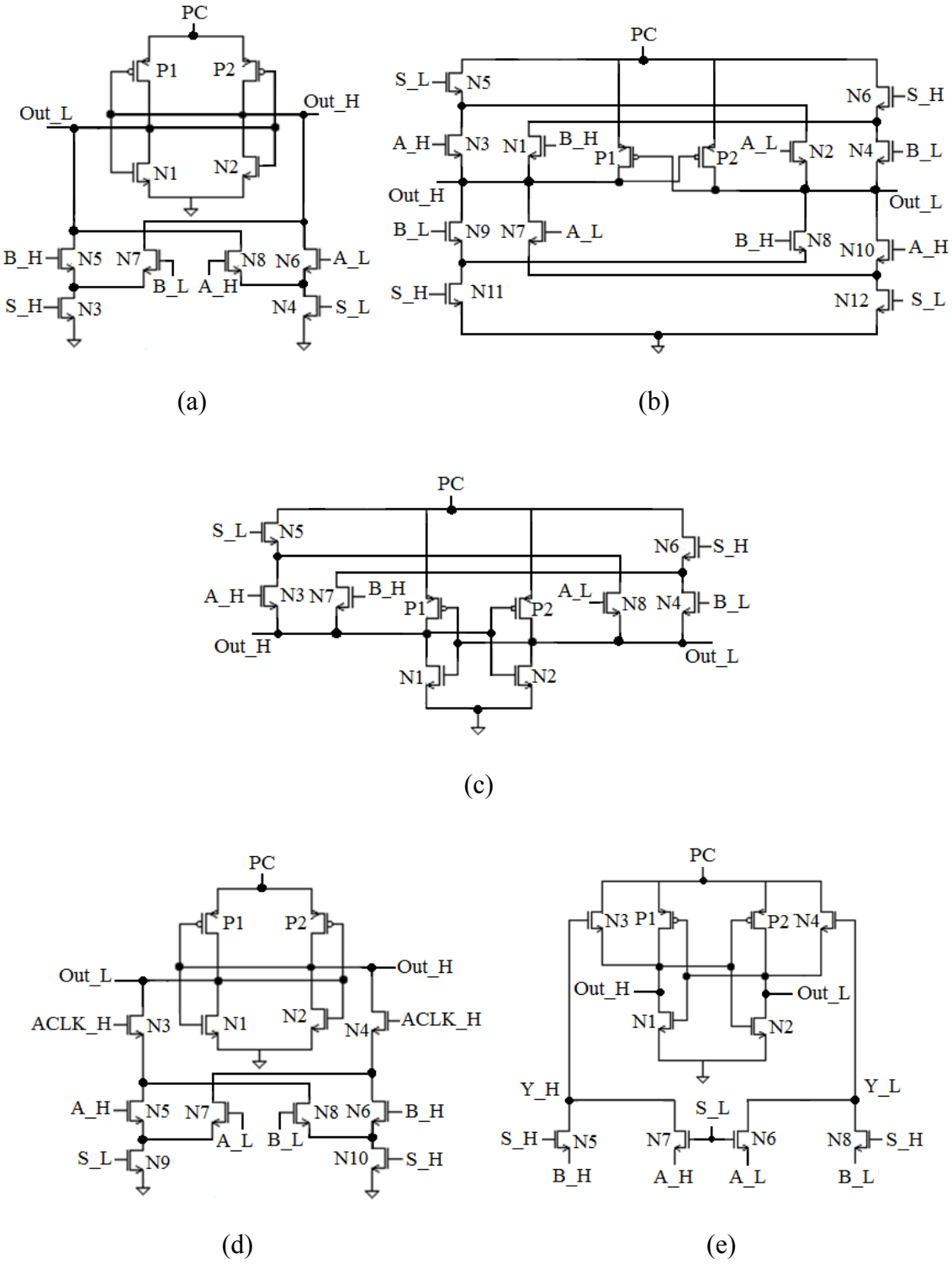



**Figure 3.1:** 2:1 MUX using (a) IECRL, (b) EACRL, (c) PFAL, (d) CAL and (e) CPAL.

phase PC will have n-stages of buffers to construct one flip-flop. The D flip-flops designed using IECRL, PFAL and EACRL use 4-phase power-clocking scheme, whereas, CPAL and CAL use 2-phase and single-phase power-clocking scheme respectively. Several adiabatic flip-flops have been reported in the literature with some

using the adiabatic MUX/NMUX as a resettable stage to provide reset terminal [36], [37], [79], [80]. Figure 3.1 shows the 2:1 MUX implementation for the five chosen adiabatic logic families.

Using the MUX as one of the stages incur area overhead due to an extra input terminal and transistor counts causing increased energy, area consumption, and layout complexity. To alleviate the problem of increased energy, area and complexity, five novel resettable buffer circuits are proposed and are used for the design of resettable flip-flops. The energy consumption of the proposed resettable flip-flops is comparable to their non-resettable counterparts, despite using more number of transistors. Additionally, they require less area compared to the MUX-based resettable flip-flops. To evaluate the performance trade-offs of five adiabatic logic families, using different power-clocking scheme, 2-bit twisted ring counter and 3-bit Up-Down counter are designed and simulated.

## 3.2 Design of Resettable Adiabatic Buffers

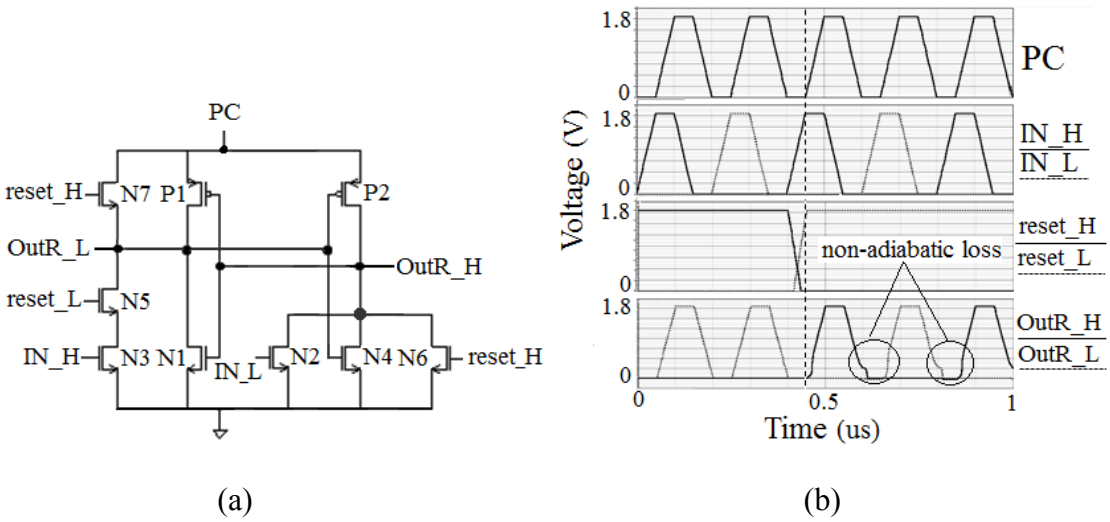
The buffer logic is the basic cell in the design of adiabatic flip-flops. In order to realize resettable adiabatic flip-flops, buffers of all the five chosen adiabatic logic families have been modified to incorporate the reset terminal for the design of resettable buffer. The modification adds several features which makes it suitable for low power applications with 10% increment in layout area compared to that of non-resettable buffers. First, the resettable buffers give low resistance output by connecting either in parallel to the pMOS cross-coupled transistor or to nMOS evaluation network or cross-coupled nMOS transistors and forming a transmission gate pair. As a result, the proposed resettable buffer circuits consume similar or less energy compared to the non-resettable buffers. Second, when in reset mode, resettable buffers help in reducing NAL by providing one of the output nodes to either connect to PC or ground.

### *3.2.1 Resettable IECRL Buffer*

The schematic of the resettable buffer gate using IECRL is shown in Figure 3.2 (a). When ‘reset_H’ signal is high (‘reset_L’ is low), transistor N6 turns ON and pulls down the node ‘Out’ to ground. Similarly, transistor N7 also turns ON and the node ‘OutR_L’ starts following PC. At the same time, ‘reset_L’ signal is low and disconnects the path between the node ‘OutR_L’ and ground. Figure 3.2 (b) shows the operation of the

resettable IECRL buffer. The transistors N6 and N7 also help in eliminating NAL in the evaluation period during the reset operation.

When the 'reset_H' signal is low ('reset_L' is high), resettable IECRL buffer works similar to that of the non-resettable buffer. The transistors N6 and N7 are turned OFF, however, due to the voltage difference between the drain and source their resistance and capacitance will be reflected at the output nodes. The decrease in resistance causes a reduction in AL, whereas, NAL at the output nodes increases as it is directly proportional to the capacitance. Due to the decreased AL and increased NAL, the overall energy dissipation increases slightly.




(a) (b)

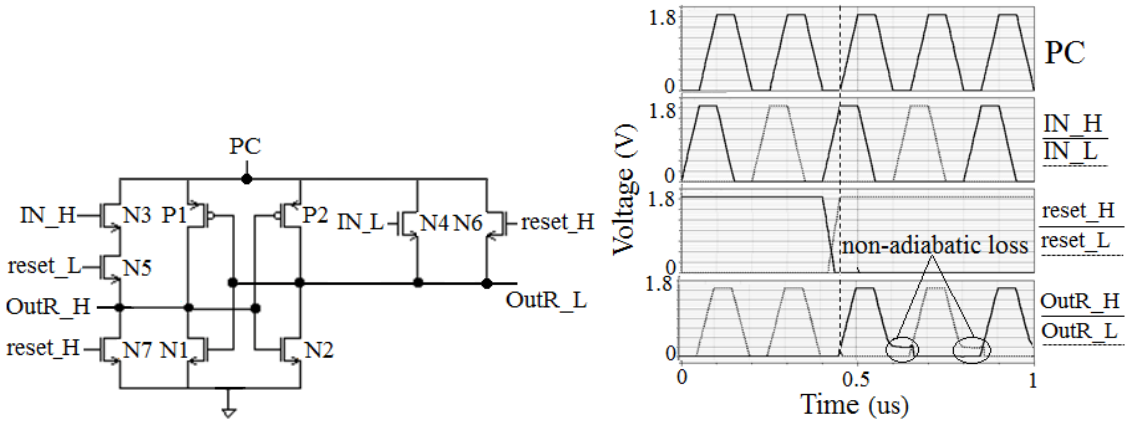
**Figure 3.2:** (a) Resettable IECRL buffer (b) Output Waveform for 10fF load.

### *3.2.2 Resettable PFAL Buffer*

The schematic and the corresponding output waveform of the resettable buffer gate using PFAL is shown in Figure 3.3 (a) and (b). PFAL resettable buffer has even lesser equivalent resistance at node 'OutR_L' during the reset and normal operation than IECRL resettable buffer. When reset signal 'reset_H' is high ('reset_L' is low), transistor N6 turns ON and node 'OutR_L' follows PC albeit not all the way to the supply voltage. Similarly, transistor N7 turns on and pulls down the node 'OutR_H' to ground. On the other hand, 'reset_L' signal is low; it disconnects the path of node 'OutR_H' from PC via transistor N3.

When the 'reset_H' signal is low ('reset_L' is high), resettable PFAL buffer works similar to its non-resettable counterpart. Like discussed above for IECRL, transistors N6

and N7 are turned OFF, however, due to the voltage difference between the drain and source their resistance and capacitance will be reflected at the output nodes causing a reduction in AL and increment in NAL, such that, the resultant energy dissipation increases slightly.



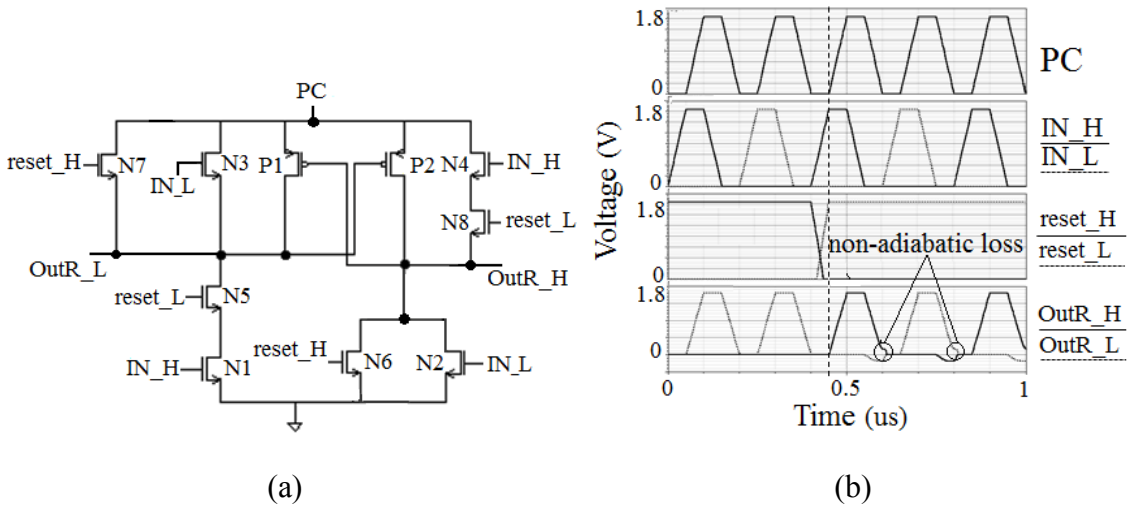

(a) (b)

**Figure 3.3:** (a) Resettable PFAL buffer (b) Output Waveform for 10fF load.

### *3.2.3 Resettable EACRL Buffer*

The schematic and the corresponding output waveform for the resettable buffer gate using EACRL is shown in Figure 3.4 (a) and (b). Since the logic is based on duplicate evaluation network, EACRL resettable buffer uses duplicate reset inputs; one connected between the output and the ground and the other connected between the input and the output. The AND/NAND implementation using EACRL is similar to the resettable buffer circuit of Figure 3.4 (a).

When ‘reset_H’ signal is high (‘reset_L’ is low), transistors N7 and N8 are turned ON and N6 and N5 are turned OFF, the path from PC to node ‘OutR_H’ and ground to node ‘OutR_L’ is cut-off. At this instant, transistors N7 and N8 help in reducing the AL by reducing the equivalent resistance at the output nodes ‘OutR_H’ and ‘OutR_L’. Due to the duplicate evaluation network, the variation in EACRL energy consumption across the ramping time is less compared to the other adiabatic logic designs. When ‘reset’ signal is low (‘reset_L’ is high), the resettable EACRL buffer works similar to that of the non-resettable EACRL buffer but reduces the output resistance at both the output nodes compared to the other four adiabatic logic families.

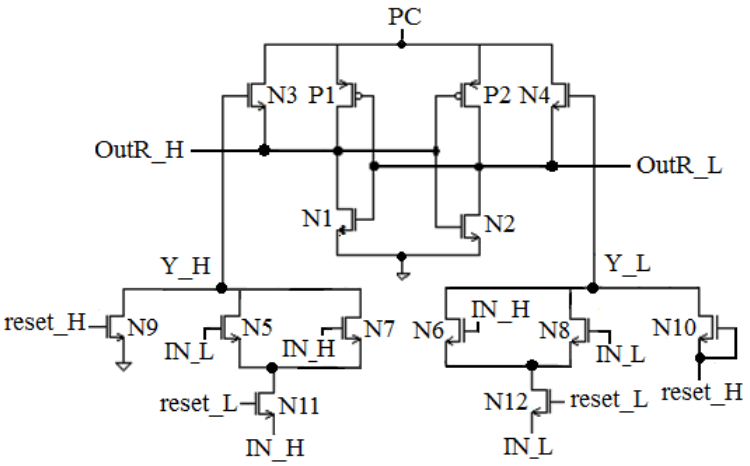



(a) (b)

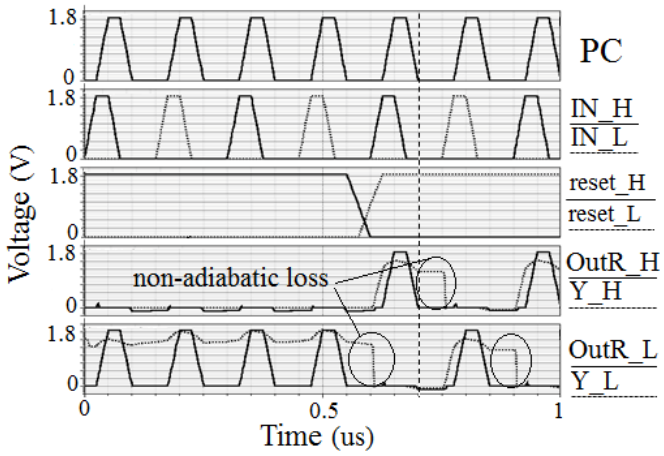
**Figure 3.4:** (a) Resettable EACRL buffer (b) Output Waveform for 10fF load.

### *3.2.4 Resettable CPAL Buffer*

The schematic and the corresponding waveform of the resettable buffer gate using CPAL is shown in Figure 3.5. (a) and (b). Since the CPAL logic is based on pass-transistors, in the resettable buffer circuit, the reset inputs are provided to the nMOS transistor N9 and N10 to pass logic '0' and '1' to the complementary intermediate nodes Y_H and Y_L. When 'reset_H' signal is high ('reset_L' low) transistor N10 turns ON and passes logic '1' ('reset_H' high) with a drop of one threshold voltage, but high enough to turn ON the transistor, N4. Thus, the node 'OutR_L' follows PC. At the same instant, transistor N9 also turns ON and the gate of transistor N3 is set at '0' voltage which switches it OFF. Because node 'OutR_L' follows PC, transistor N1 turns ON and node 'OutR_H' is pulled down to the ground. The complementary reset input terminal 'reset_L' helps in disconnecting the input 'IN_H' and complementary input 'IN_L' signals to reach to the intermediate nodes Y_H and Y_L through transistor N11 and N12, ensuring normal operation. When the 'reset_H' signal is low ('reset_L' is high), resettable CPAL buffer works similar to that of the non-resettable CPAL buffer.




(a)



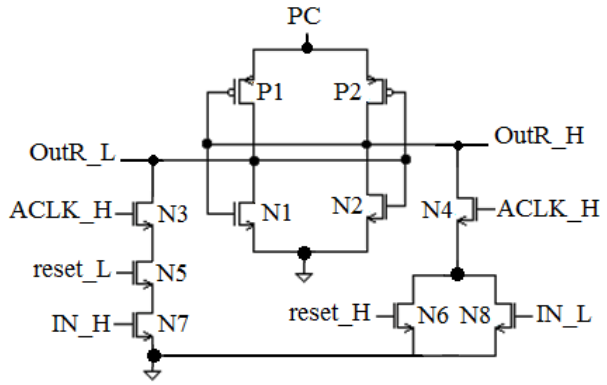

(b)

**Figure 3.5:** (a) Resettable CPAL buffer (b) Output Waveform for 10fF load.

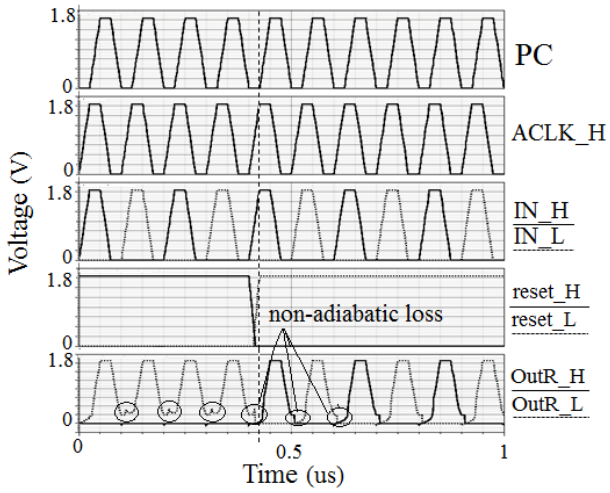
### *3.2.5 Resettable CAL Buffer*

The reset input used in the CAL buffer is an asynchronous input having priority over the auxiliary input signal, ACLK_H. Figure 3.6 (a) and (b) shows the schematic and the output waveform of the resettable CAL buffer. When 'reset_H' signal is high ('reset_L' is low), irrespective of the auxiliary clock input transistor, N4, the node 'OutR_H' is pulled down to the ground, which turns ON the transistor P1, forcing the node 'OutR_L' to follow PC. The AND/NAND adiabatic implementation using CAL is similar to the CAL resettable buffer circuit. At the same instant, the complementary reset signal 'reset_L' turns the transistor, N5 OFF, thus disconnecting the input and the output node 'OutR_L'. The resettable CAL buffer work similar to that of the non-

resettable CAL buffer when ‘reset_H’ signal is low (‘reset_L’ is high). The resistance at the ‘OutR_H’ node reduces due to the transistor, N6, causing the overall energy dissipation to increase slightly.




(a)




(b)

**Figure 3.6:** (a) Resettable CAL buffer (b) Output Waveform for 10fF load.

## 3.3 Design of Resettable Adiabatic Flip-flops

This section discusses and compares the energy performance of the proposed and the existing MUX-based resettable flip-flops and non-resettable flip-flops using five multi-phase adiabatic logic families. Because the proposed resettable buffers provide low resistance at the output nodes, the overall energy consumption is either less or comparable to their non-resettable buffer counterparts. Additionally, the energy consumption of the proposed resettable flip-flops is comparable to that of the non-resettable flip-flops and thus makes it suitable for energy-efficient applications.

Flip-flops are the inherent building blocks of all the synchronous systems. Considering the operational characteristic of the adiabatic circuits, the designs of adiabatic flip-flops should be different from that of the conventional CMOS flip-flops [81]. Adiabatic D-flip-flops can be built using a cascaded buffer chain where the input is shifted to the output through the n-stage buffer chain (depending upon the adiabatic logic style) by one clock period. Because the output of the single-phase CAL buffer follows the input with $360^{o}$ phase-lag and uses auxiliary clock signal the input 'D_H' is just shifted to the output terminal 'Q_H' through the two-stage CAL buffer chain by two clock periods. Similarly, the output of the two-phase CPAL buffer follows the input with a 180° phase lag, hence the input 'D_H' is shifted to the output terminal 'Q_H' through the two-stage CPAL buffer chain by one clock period. Additionally, considering the fact that the outputs of all the 4-phase buffers, namely; IECRL, PFAL, and EACRL follow the input with a $90^{o}$ phase-lag, the D flip-flop is constructed using 4 stages of the buffer. Due to the long idle period, 2-phase power-clocking scheme has high computation time compared to the 4-phase power-clocking scheme even for the same ramping time.

The existing resettable D flip-flops are implemented either using single-phase CAL [79] or 2-phase CPAL [37]. They use adiabatic MUX/NMUX as their second resettable stage. The major disadvantage of this is that the output from the flip-flops can only be fed to any subsequent logic after the second stage. Thus, the subsequent stage has to wait for the output for complete one cycle. This reduces the throughput of the circuit. This drawback has now been resolved by the proposed resettable adiabatic flip-flops. For all the five proposed resettable adiabatic flip-flops, the output can be fed to the subsequent logic stage even after the first stage. This increases the throughput and gives the flexibility of tapping the outputs from the required phase. Moreover, the proposed resettable flip-flops using CAL and CPAL have an advantage in terms of the area (transistor count) over the existing MUX-based resettable flip-flops using CAL and CPAL.

The first stage of the adiabatic flip-flops requires a resettable buffer and the other stages use non-resettable buffers. The structures of the 4-phase, 2-phase and single-phase resettable D flip-flops are shown in Figures 3.7, 3.8 and 3.9 respectively. It should be noted that the 4-phase power-clock, 2-phase non-overlapping power-clock and single-phase power-clock with non-overlapping auxiliary clocks 'ACLK_H' and 'ACLK_L' are shown in Figure 2.5 of Chapter 2. The reset input used in the resettable designs is an

asynchronous signal having priority over other input signals and thus must be stable before the beginning of the evaluation period of the power-clock to meet the adiabatic constraints.

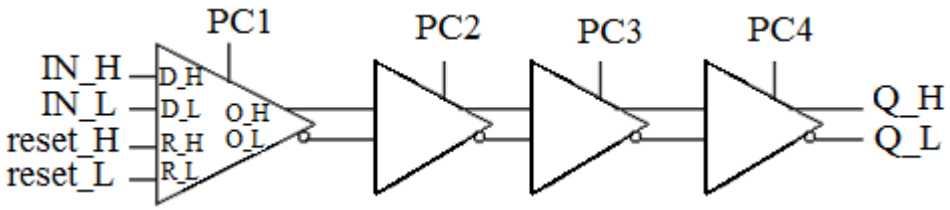



**Figure 3.7:** Resettable adiabatic flip-flop using 4-phase power-clocks.

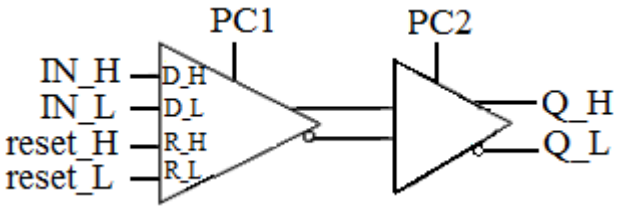



**Figure 3.8:** Resettable adiabatic flip-flop using 2-phase power-clocks.

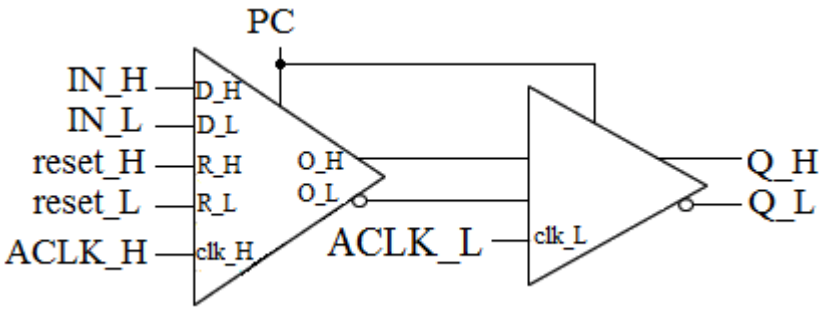
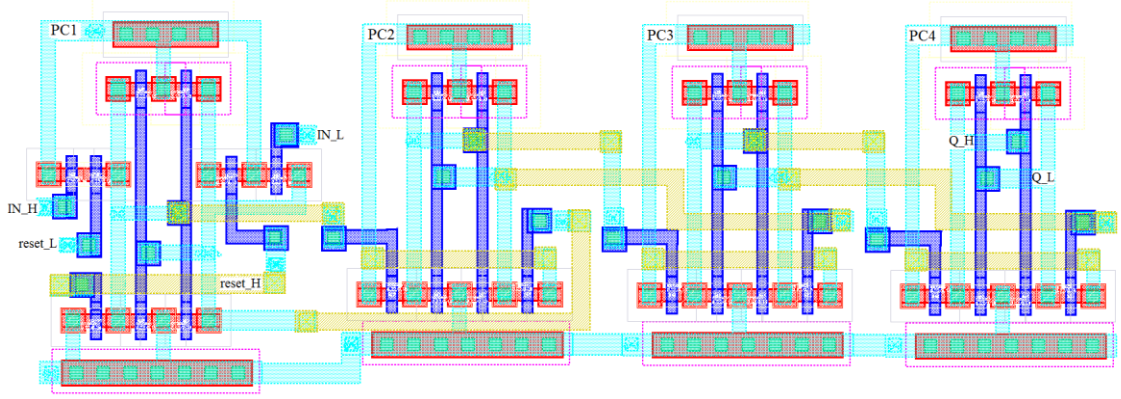
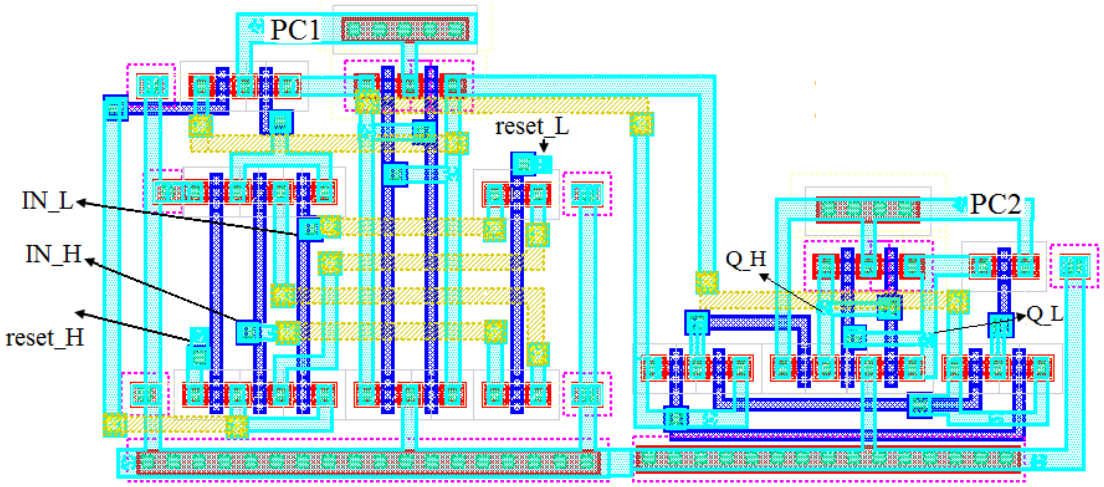



**Figure 3.9:** Resettable adiabatic flip-flop using single-phase power-clock and auxiliary clocks.

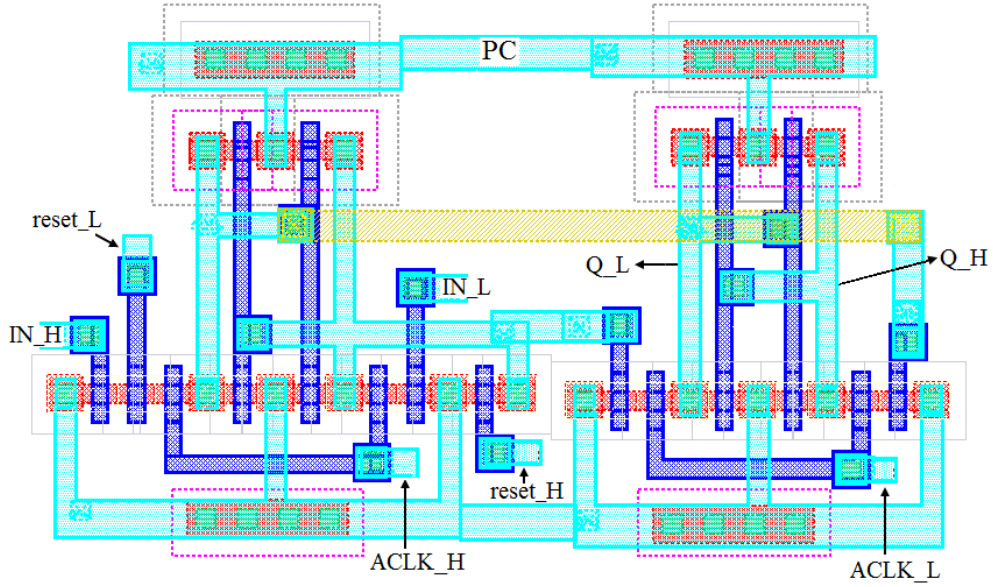
The full-custom layouts of the proposed 4-phase flip-flop using PFAL are shown in Figure 3.10 (a). The 4-phase flip-flops using IECRL and EACRL have the same number of stages. The only difference is in the layout area which is summarised in Table 3.1. The full-custom layouts of the proposed flip-flops using CPAL and CAL are shown in Figure 3.10 (b) and (c) respectively. Table 3.1 summarises the layout area used by non-resettable, existing MUX-based resettable and proposed resettable flip-flops for all the five adiabatic logic styles.




(a)




(b)




(c)

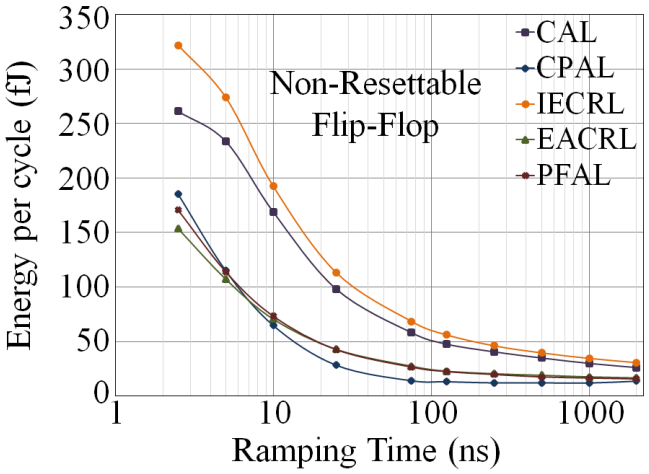
**Figure 3.10:** Proposed resettable flip-flop layouts for (a) PFAL (b) CPAL (c) CAL.

**Table 3.1:** Comparison of layout area of non-resettable, existing MUX-based resettable and proposed resettable flip-flops.

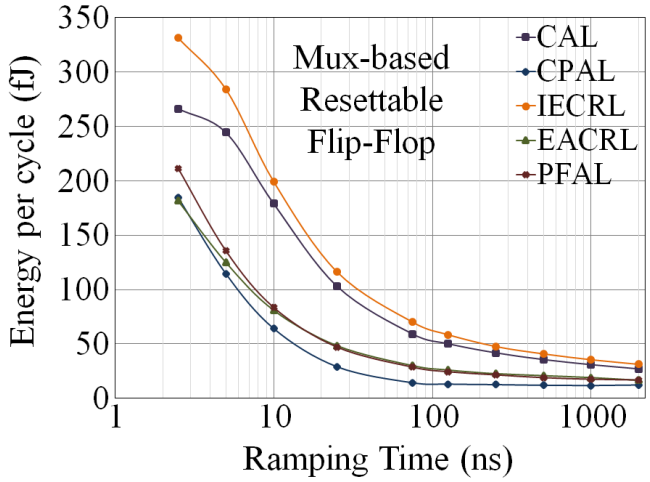
| **Adiabatic Logic Families** | **Area (µm)$^2$** | | |
|---|---|---|---|
| | ***Non-Resettable Flip-flop*** | ***Existing MUX-based Resettable Flip-Flop*** | ***Proposed Resettable Flip-flop*** |
| CPAL | 5.68 x 17.28 | 6.44 x 17.32 | 8.99 x 18.40 |
| CAL | 6.44 x 9.68 | 7.71 x 12.48 | 6.44 x 10.91 |
| PFAL | 6.46 x 18.75 | 7.72 x 23.16 | 6.98 x 19.29 |
| EACRL | 6.07 x 20.89 | 8.24 x 30.75 | 6.97 x 24.75 |
| IECRL | 6.24 x 17.68 | 7.51 x 22.35 | 6.56 x 18.73 |

From Table 3.1, it can be seen that the proposed resettable flip-flops consume less area as compared to the existing MUX-based design for all the adiabatic logic families, except CPAL. Because both the CPAL buffer and MUX use the same number of transistors, thus, the area of the MUX-based CPAL flip-flop is less. However, both the MUX-based resettable and non-resettable flip-flops use the same number of transistors, but the area of former is larger than that of the later as the former uses an extra input pin requiring extra routing space.

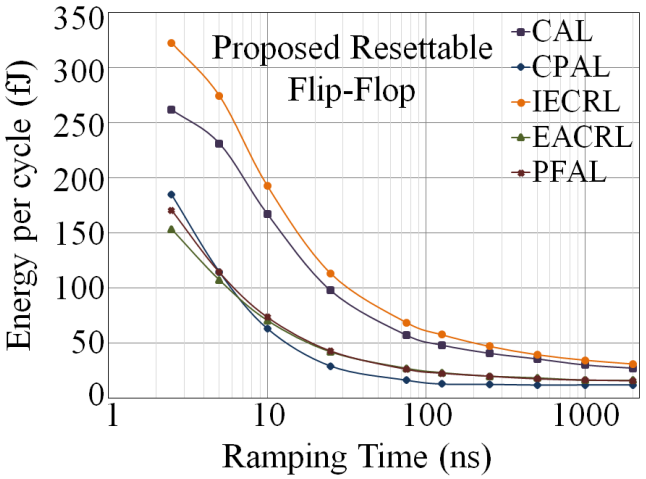
The energy consumption is measured per clock-cycle and each of the adiabatic flip-flops is compared at ramping times ranging from 2.5ns to 2000ns under an output load capacitance of 100fF. The energy consumption of the flip-flops was measured through simulations for the periodic sequence of "101010….", thereby, giving the maximum average energy consumed per cycle. Figure 3.11 (a), (b) and (c) illustrate the relationship between pre-layout energy consumption and ramping time for non-resettable, existing MUX-based and proposed resettable flip-flops respectively. It can be seen that the energy consumption of the proposed flip-flops is much less compared to the existing MUX-based design. The MUX-based flip-flops using EACRL and PFAL consume approximately 16%, more energy, whereas IECRL and CAL consume approximately 3% to 5% more energy when compared to non-resettable counterparts. The energy consumption of the MUX-based CPAL is similar to that of the non-resettable flip-flop for the entire range of ramping time with an increment of approximately 0.5%.




(a)



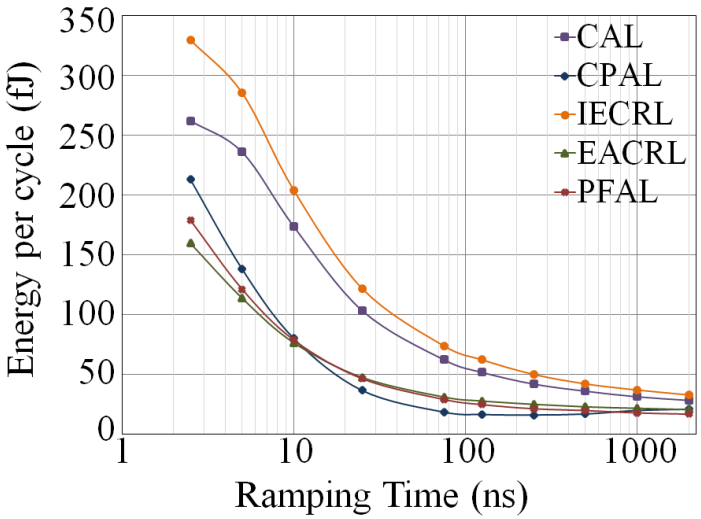

(b)



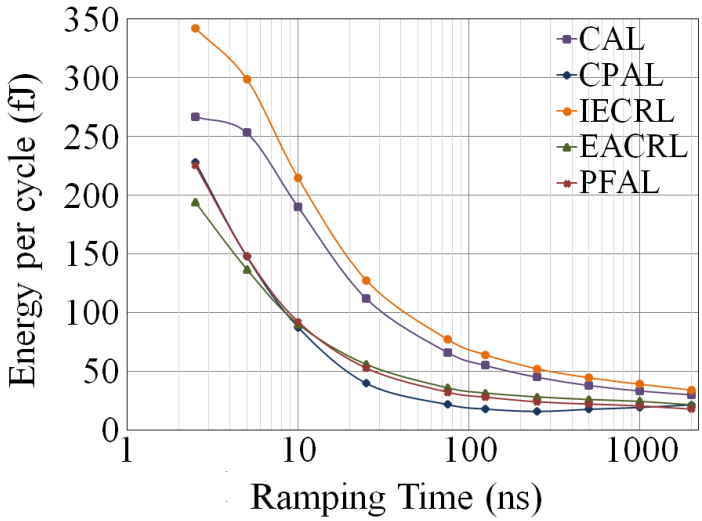

(c)

**Figure 3.11:** Pre-layout energy consumption versus ramping time of (a) Non-resettable (b) Existing MUX-based (c) Proposed resettable flip-flops.

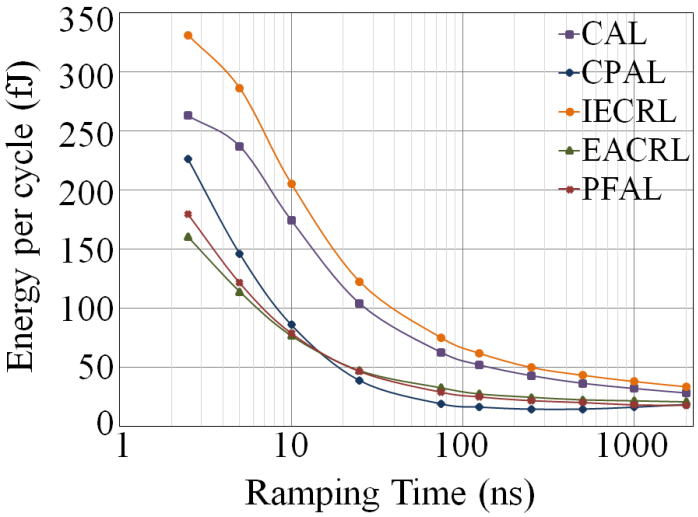
From Figure 3.11 (c), the proposed resettable flip-flops using PFAL, CPAL, CAL, and IECRL consume approximately 0.5% to 1% more energy compared to the non-resettable flip-flops. On the other hand, due to the decrease in the output resistance of the proposed buffer using EACRL, its energy consumption shows a decrement of approximately 6% compared to the non-resettable flip-flop. Flip-flop using CPAL consumes minimum energy at longer ramping times, however, as the ramping time is made shorter its energy dissipation increases. The post-layout simulation results in Figure 3.12 (a), (b) and (c) show a similar trend as shown by the pre-layout simulations. The difference is in terms of the increased energy consumption due to the addition of the parasitic resistance and capacitance of the routing metal layers.




(a)



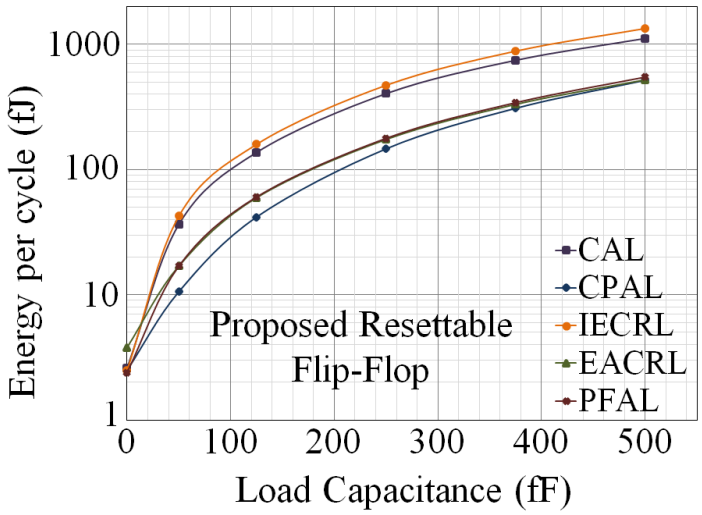

(b)



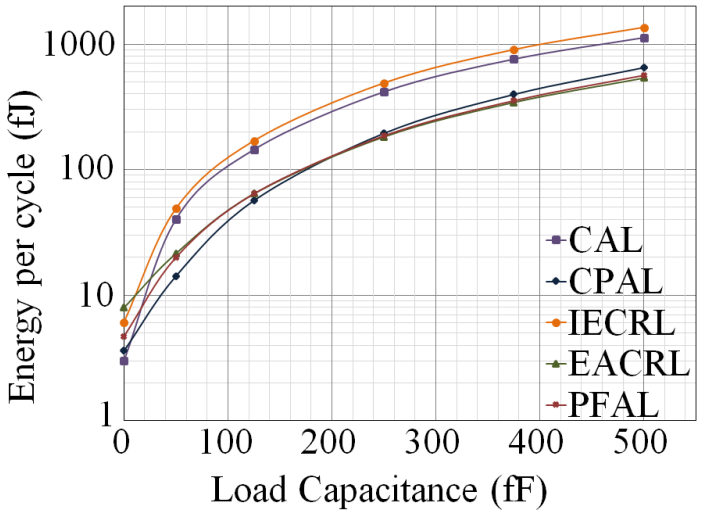

(c)

**Figure 3.12:** Post-layout energy consumption (per cycle) versus ramping time of flip-flops (a) Non-resettable (b) Existing MUX-based (c) Proposed resettable.

Figure 3.13 (a) and (b) show the effect of loading on energy consumption of the proposed resettable flip-flops at the ramping time of 25ns. From Figure. 3.13 (a), the energy consumption of IECRL and CAL is maximum due to NAL present during both the evaluation and the recovery period. However, EACRL and PFAL consume the same amount of energy for the load capacitance value higher than 50fF, as both suffer from NAL only during the recovery period. Since, EACRL also suffers from energy loss due to the absence of cross-coupled nMOS transistors causing higher coupling loss at the output nodes, its energy at zero capacitive load is highest. The advantage of having zero NAL at the output nodes makes CPAL consume the least energy at smaller values of load capacitances but as the load capacitance increases to 500fF, its energy consumption becomes almost similar to that of PFAL and EACRL. Although flip-flop using CPAL is able to work up to the ramping time of 2.5ns, the output nodes of the flip-flop are not charged up fast enough resulting in the voltage difference between the output node and PC. Hence, the CPAL circuit dissipates more energy. Similarly, due to the addition of parasitics after the post-layout simulation, CPAL consumes the least energy until the load capacitance value of 160fF. Beyond that, EACRL and PFAL consume the least energy as can be seen from Figure 3.13 (b).




(a)




(b)

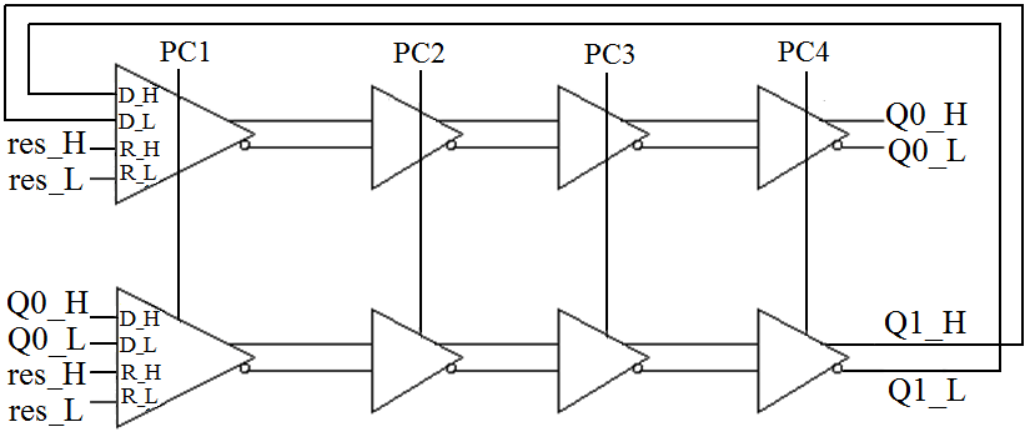
**Figure 3.13:** Energy per cycle under different load capacitances at the ramping times of 25ns (a) pre-layout (b) post-layout.

In order to compare the performance of the non-resettable and resettable adiabatic flip-flops, 2-bit twisted ring counter was designed using each of the five adiabatic logic families and was evaluated in the next section.

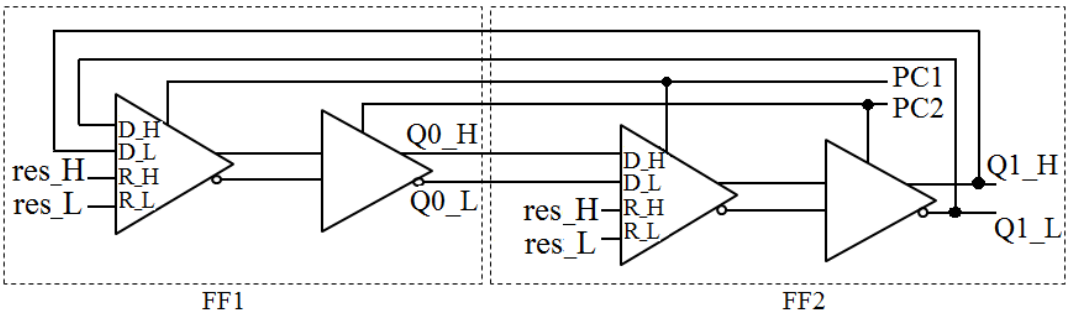
## 3.4 Design of 2-bit Twisted Ring Counters using Adiabatic Logic

The twisted ring counter is able to self-initialize from an all-zeros state and does not have any external inputs (only flip-flops no logic gates), and therefore, was used as a vehicle for comparing five non-resettable adiabatic flip-flops. Commonly, a resettable

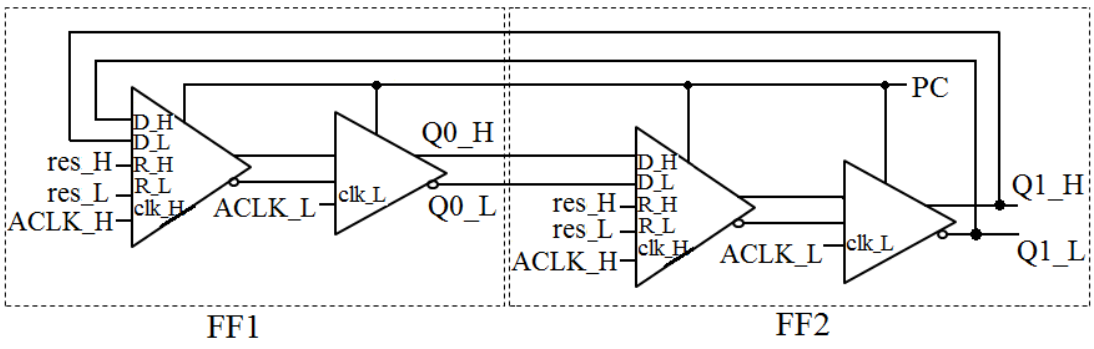
flip-flop is needed as it forces the logic into a known state at the beginning of the simulation time. The 2-bit twisted ring counter consisting of two D flip-flops implemented using 4-phase, 2-phase and single-phase adiabatic logic designs are shown in Figure 3.14 (a), (b) and (c) respectively. The outputs, 'Q0_H', and 'Q1_H' represent the Least Significant Bit (LSB) and the Most Significant Bit (MSB) respectively.




(a)



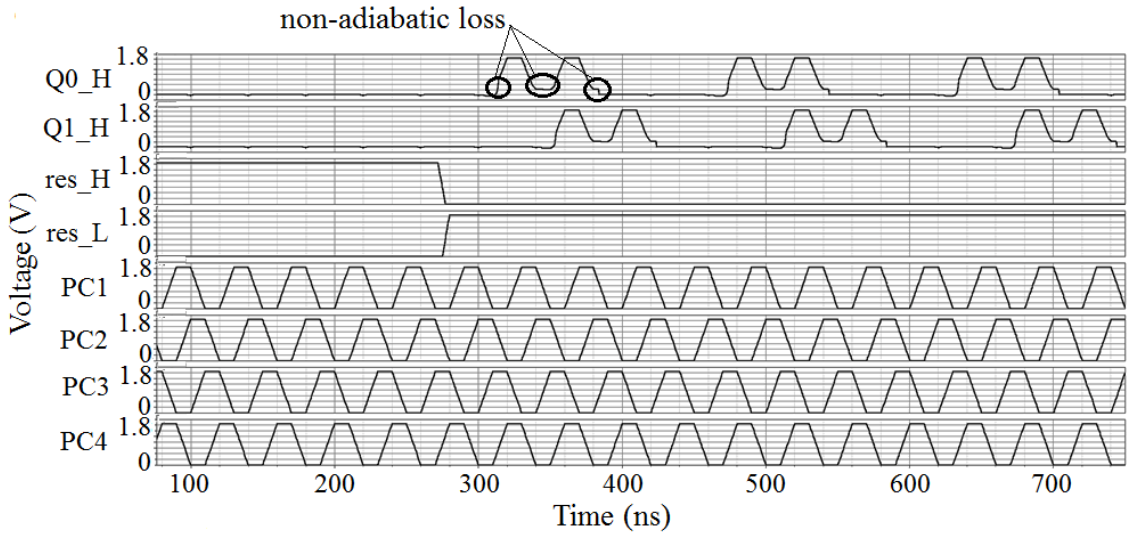

(b)



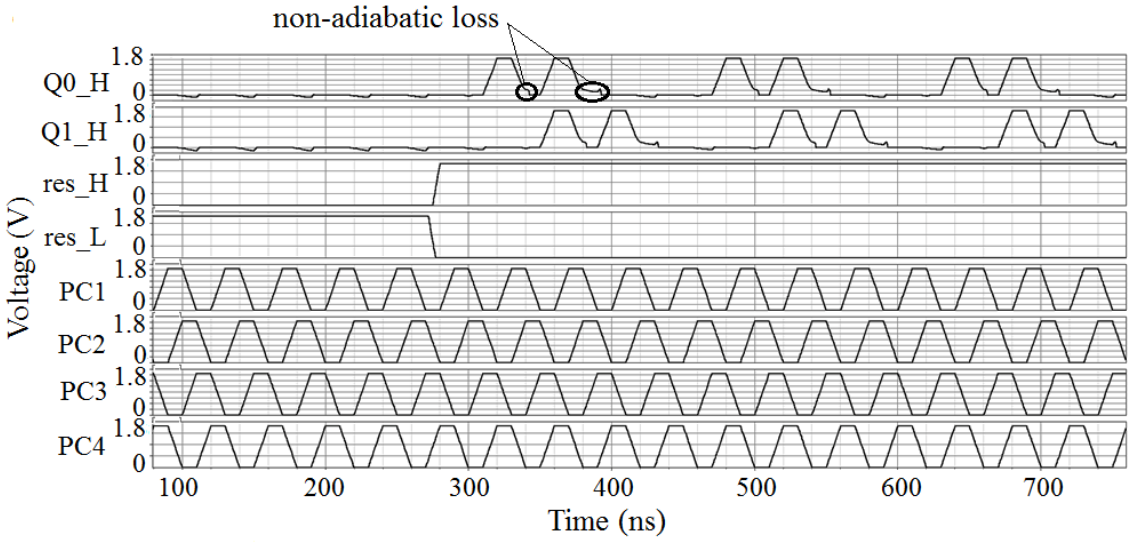

(c)

**Figure 3.14:** 2-bit resettable twisted ring counter using adiabatic logic with power-clocking scheme (a) 4-phase(b) 2-phase (c) Single phase.

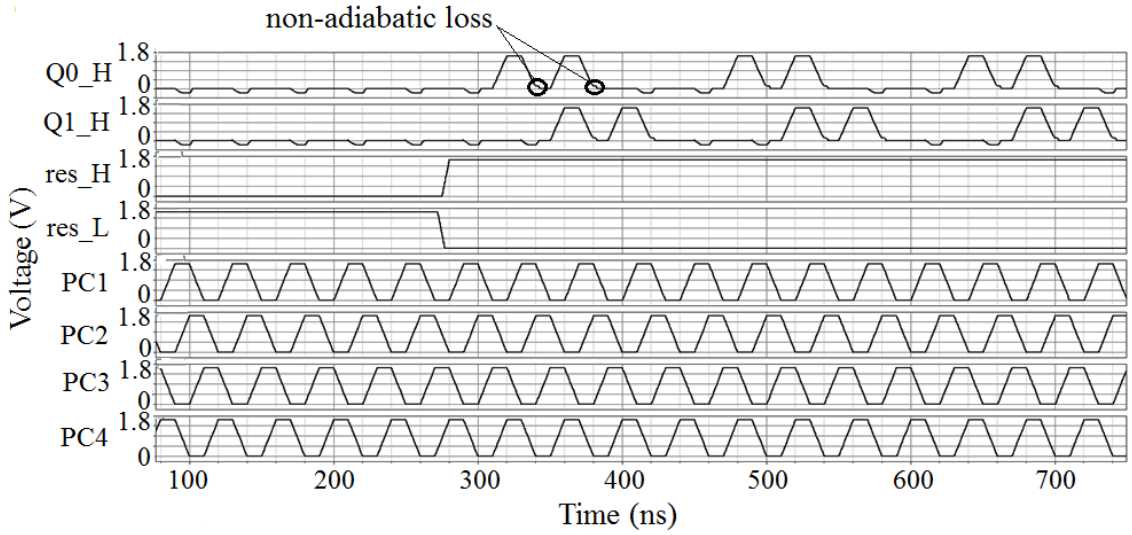
When the reset signal ‘res_H’ is logic ‘1’, the output of the 2-bit twisted ring counter ‘Q0_H’ and ‘Q1_H’ goes to zero and the complement outputs, ‘Q0_L’ and ‘Q1_L’ follow the power-clock. The outputs of the 2-bit twisted ring counter using five adiabatic logic families along with the signals, reset, resetb and power-clock are shown in Figure 3.15 (a), (b), (c), (d) and (e). It can be seen that the outputs of the 2-bit twisted ring counter implemented using IECRL, PFAL, EACRL, and CAL suffer from NAL. The region encircled shows this NAL arising due to threshold voltage degradation on one of the outputs. On the other hand, CPAL doesn’t show NAL on the output nodes but has a non-adiabatic dynamic loss due to the coupling effect on the low-level output node (output node not following the power-clock). The flip-flop design using CAL works on single-phase power-clock however, due to the use of auxiliary clock inputs, the cascaded logic becomes complex. Also, from Figure 3.15 (d), the output of CAL implementation shows that each stage of the twisted ring counter is valid for two power-clock periods, hence large latency for sequential circuit design.



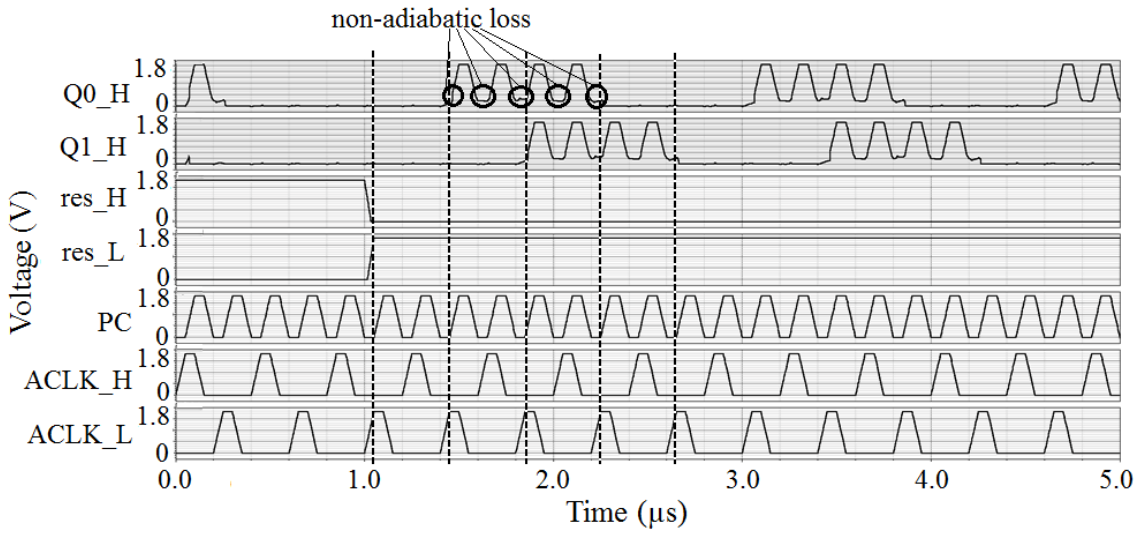

(a)


non-adiabatic loss
Q0_H
Q1_H
res_H
res_L
PC1
PC2
PC3
PC4
1.8
0
Voltage (V)
100
200
300
400
500
600
700
Time (ns)

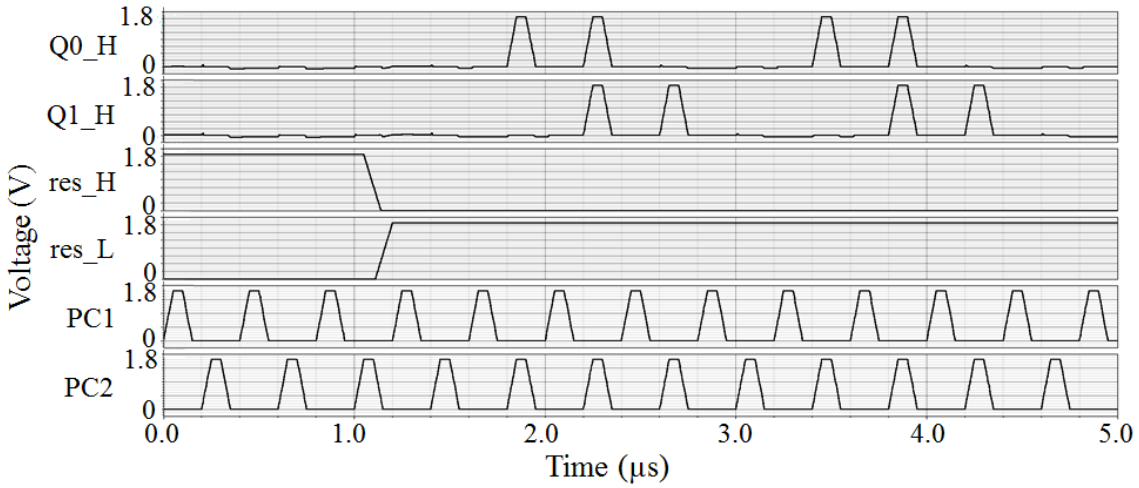
(b)


non-adiabatic loss
Q0_H
Q1_H
res_H
res_L
PC1
PC2
PC3
PC4
1.8
0
Voltage (V)
100
200
300
400
500
600
700
Time (ns)

(c)


non-adiabatic loss
Q0_H
Q1_H
res_H
res_L
PC
ACLK_H
ACLK_L
1.8
0
Voltage (V)
0.0
1.0
2.0
3.0
4.0
5.0
Time (µs)

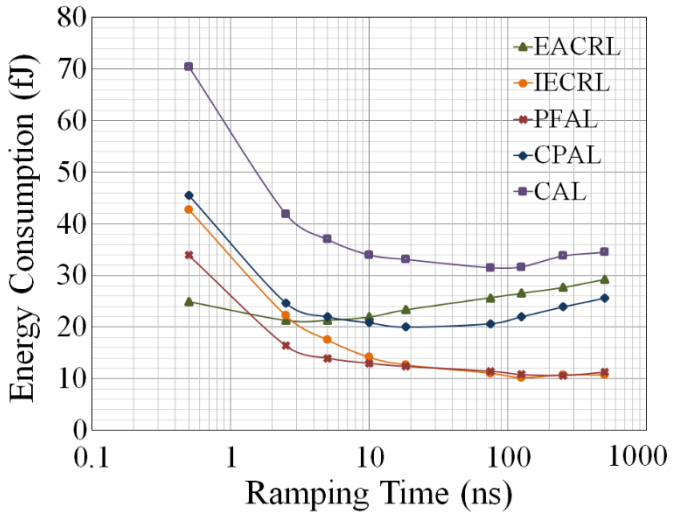
(d)



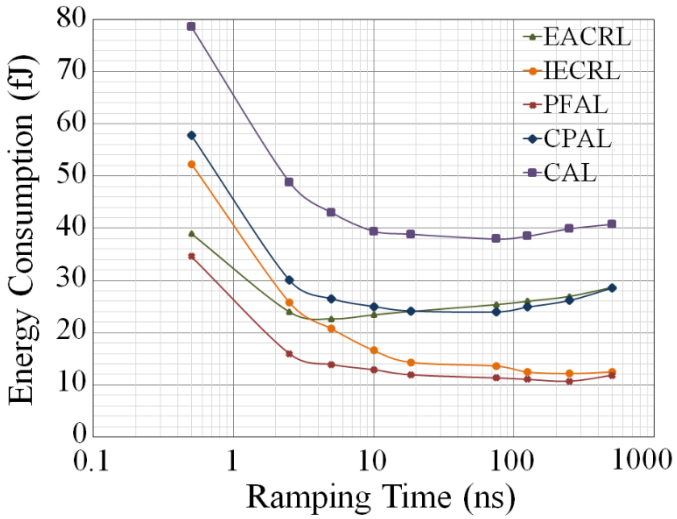

(e)

**Figure 3.15:** Output waveforms of 2-bit resettable twisted ring counter using (a) IECRL, (b) PFAL, (c) EACRL, (d) CAL, (e) CPAL.

The 2-bit non-resettable and resettable twisted ring counter circuits using the five chosen adiabatic logic families were also designed to verify if the trend of energy consumption remains the same for the larger circuit at various ramping times. The energy consumed by 2-bit twisted ring counters was measured over its four states under zero external load capacitance. The energy consumption of the non-resettable and the resettable 2-bit twisted ring counter for EACRL, IECRL, PFAL, CPAL, and CAL are shown in Figures 3.16 (a) and (b) respectively. Because of the decreased output resistance, the difference in the energy of the resettable twisted ring counter using EACRL and PFAL compared to that of the non-resettable counter are approximately 5% and 2% respectively for a ramping time longer than 10ns. Whereas, the energy consumption of the 2-bit resettable twisted ring counter using IECRL, CAL and CPAL are approximately 18% more than their non-resettable counterparts. The energy performance of CAL is worst for both the non-resettable and the resettable counters, though not much variation in its energy consumption is observed at shorter ramping times. Similarly, the non-resettable and the resettable counters using EACRL show steady variations in energy consumption for ramping times ranging from 2.5ns to 500ns. Overall, the PFAL based design of both the non-resettable and the resettable counters show the approximately 2% difference in energy consumption at all ramping times.




(a)




(b)

**Figure 3.16:** Pre-layout energy consumption per cycle of 2-bit twisted ring counter (a) non-resettable (b) resettable.

To further evaluate and compare the performance of the five multi-phase adiabatic logic families, a 3-bit Up-Down counter was used as an application example.

## 3.5 Design of 3-bit Up-Down Counters using Adiabatic Logic

In the past, various examples like 16-bit Carry Look Ahead (CLA) [55], 8-bit multiplier [21], mode-10 counter [37] and [80], etc. have been demonstrated to show the comparison between different adiabatic logic families and the conventional CMOS design in terms of energy efficiency. In [82], [83], the authors have discussed the

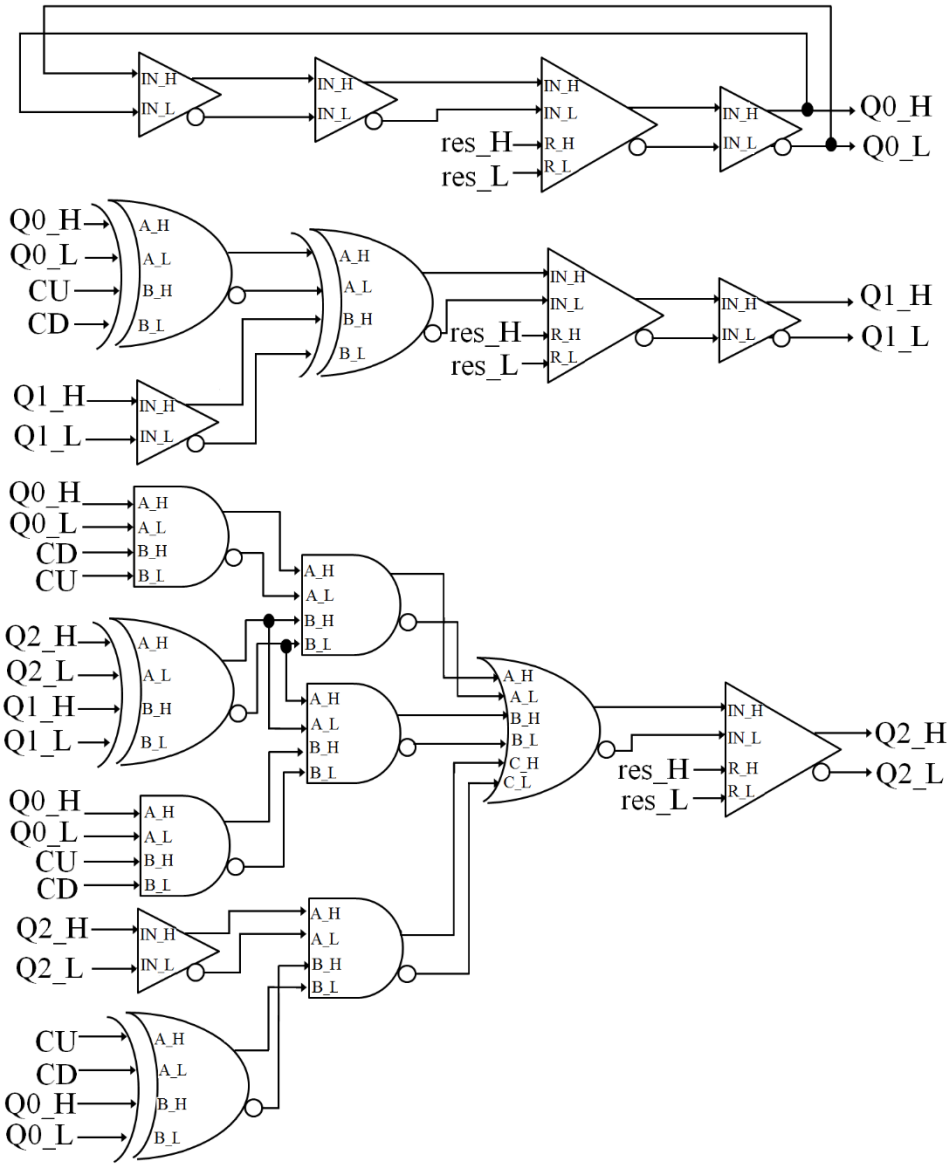
performance issues of adiabatic logic design in comparison to conventional CMOS design but lacked to give a comparison which encompasses performance issues amongst multi-phase adiabatic logic families. Despite the authors claim in [84] that, single and 2-phase adiabatic logic namely CAL and CPAL reduces the latency by half and cut down the number of buffers required significantly, this is not universally true instead it is a design specific scenario. In order to suggest appropriate performance trade-offs, a 3-bit Up-Down counter is designed and simulated using each of the five energy-efficient adiabatic logic families. Figure 3.17 (a) shows the design used for single-phase and 2-phase adiabatic logic families whereas, Figure 3.17 (b) is the design used for 4-phase adiabatic logic families. The counter starts counting up or down depending on 'UD' input signal when reset is low. It counts down when the 'CU' signal is high ('CD' signal low) and counts up 'CU' signal is low ('CD' signal is high). The Boolean expressions for $Q0\_H^{n+1} - Q2\_H^{n+1}$ are given by;

$$D0\_H = Q0\_H^{n+1} = res\_L.(Q0\_H)^n$$

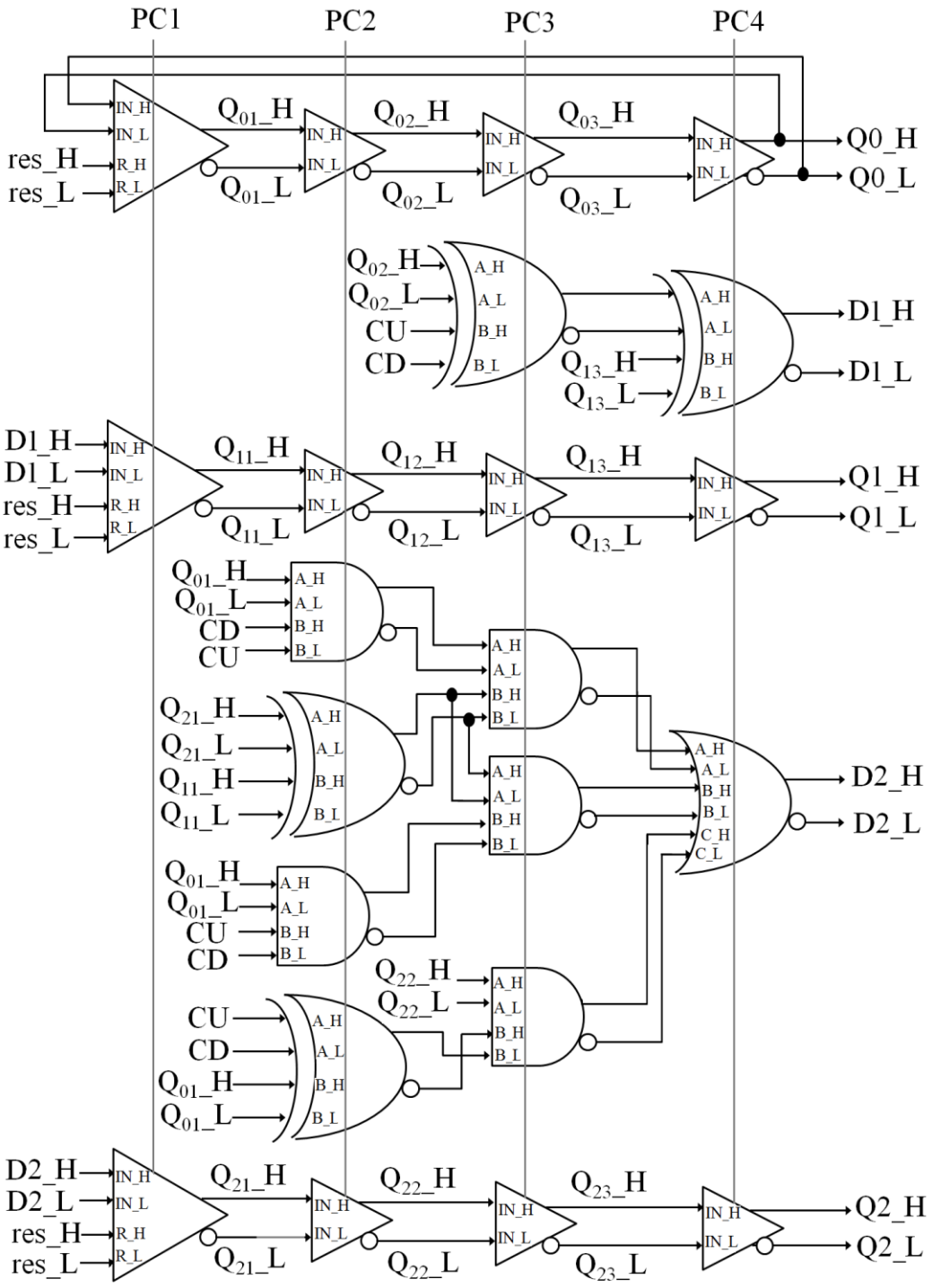
$$D1\_H = Q1\_H^{n+1} = res\_L.[Q1\_H^n \oplus (Q0\_H^n \oplus CU)] \quad (3.1)$$

$$D2\_H = Q2\_H^{n+1} = res\_L.[Q0\_H^n.CD.(Q1\_H^n \oplus Q2\_H^n) + \overline{Q0\_L^n}.CU.\overline{(Q1\_H^n \oplus Q2\_H^n)} + Q2\_H^n.\overline{(CU \oplus Q0\_H^n)}]$$

As can be seen from Figure 3.17 (b) and equation (3.1), to implement a function D2_H a minimum of 3 cascade levels are required. In the case of a single-phase, 2-phase and 4-phase designs; 3 power-clock periods that is 12*Tr*, 1.5 power-clock period that is 9*Tr* and 3/4 power-clock period that is 3*Tr* are required respectively. For synchronizing the LSB bit of the counter output, 'Q0_H' with 'Q1_H' and 'Q2_H' output bits in the single and 2-phase design, two buffers are added, whereas, in 4-phase designs, the correct intermediate signals from the D flip-flops are used as inputs to the XOR/XNOR and AND/NAND gate. As seen in the previous sections, both single-phase and 2-phase designs have high latency. The structures of their Up-Down counter are different compared to 4-phase design, in order to save the area and synchronization buffers. The first two and the first three stages of 'Q1_H' and 'Q2_H' respectively of the single-phase and 2-phase counter are realized using the combinational logic function thus, saving 4 synchronization buffers circuits.


IN_H
IN_L
res_H
res_L
R_H
R_L
Q0_H
Q0_L
A_H
A_L
B_H
B_L
CU
CD
Q1_H
Q1_L
C_H
C_L
Q2_H
Q2_L

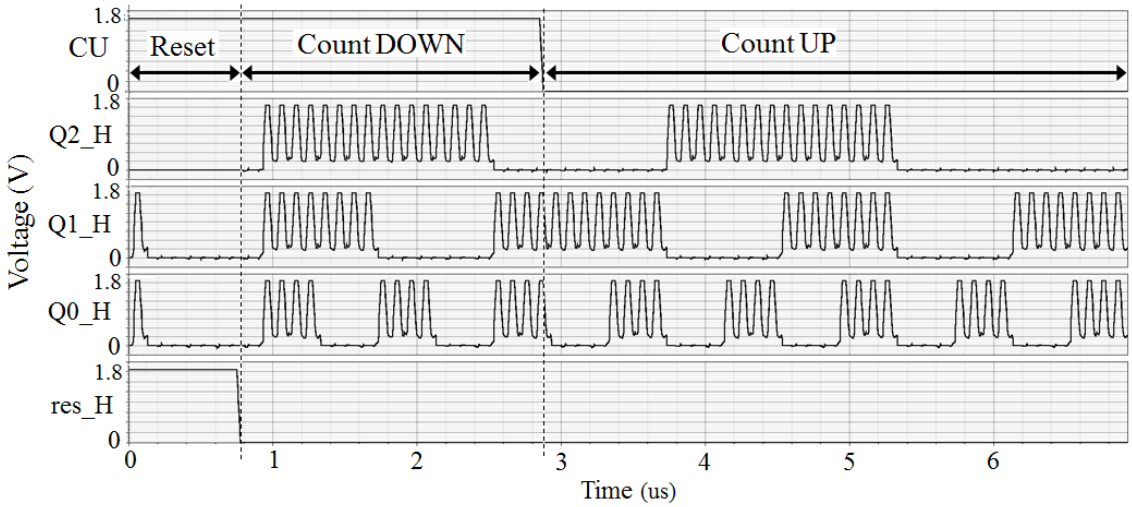
(a)



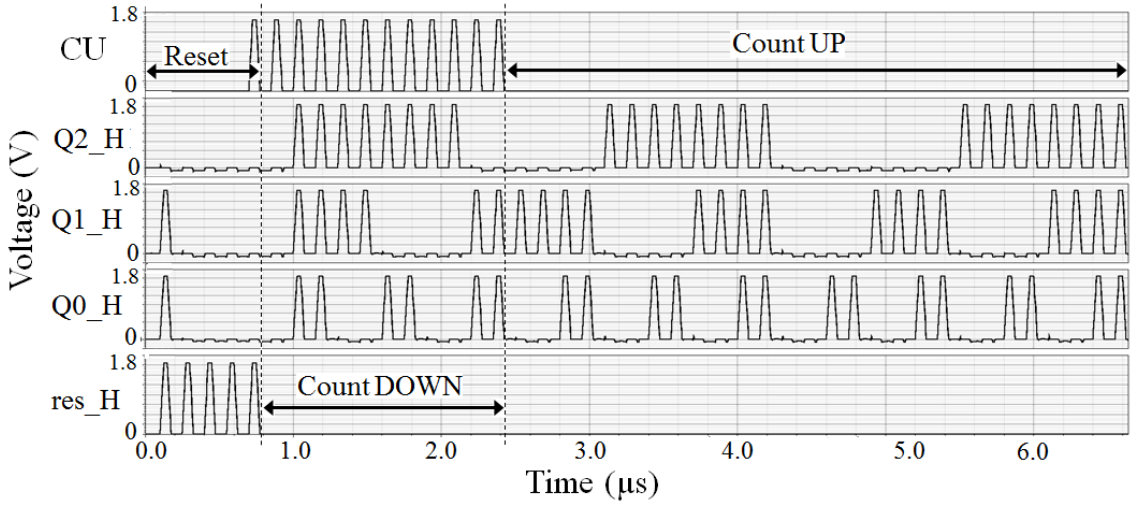

(b)

**Figure 3.17:** Up-Down counter design for (a) single-phase and 2-phase (b) 4-phase.

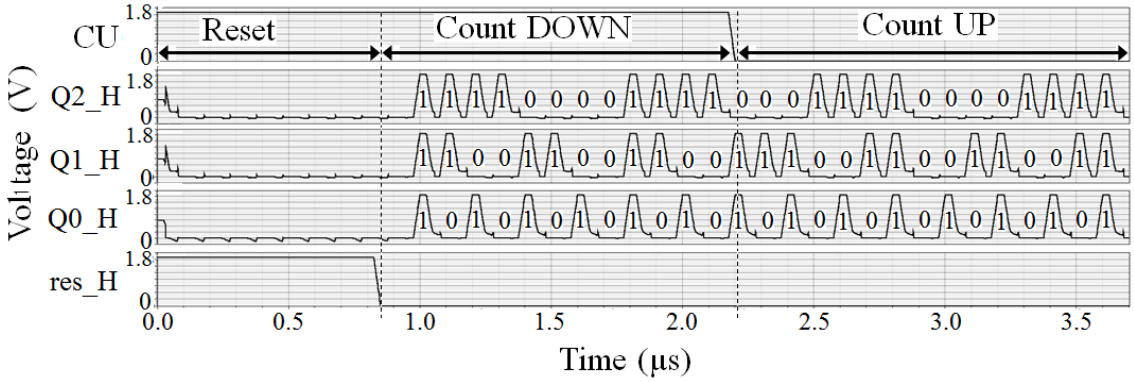
The outputs of the Up-Down counter for single-phase, 2-phase and 4-phase adiabatic logic designs are shown in Figure 3.18 (a), (b) and (c) respectively. The reset signal 'res_H' is an asynchronous signal having priority over all other signals.




(a)




(b)




(c)

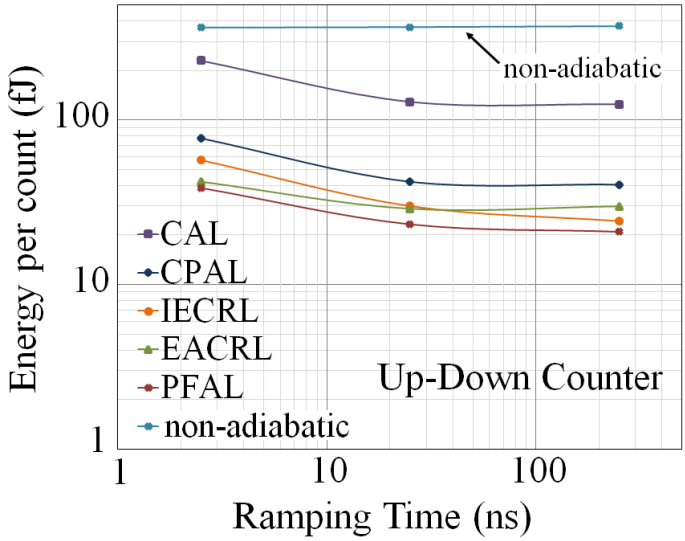
**Figure 3.18:** Up-Down counter outputs for (a) Single-phase (b) 2-phase (c) 4-phase.

The energy consumption of the counter is averaged over fifteen counts, counting from seven down to zero and back to seven. Figure 3.19 (a) shows the average energy consumption per count of the five adiabatic logic designs and non-adiabatic (conventional CMOS) for an Up-Down counter under a load capacitance of 10fF at ramping times ranging from 2.5ns to 250ns. Because the CAL and IECRL logic designs have an evaluation network connected between the output and the ground, they also suffer from NAL in the evaluation period of the power-clock apart from the recovery period. As the ramping time becomes shorter, the AL combined with NAL makes the output node lagging behind the power-clock thus increasing the energy dissipation. SinceNAL is directly proportional to the node capacitance and the threshold voltage; different adiabatic logic families have different NAL. As the ramping time becomes longer, the leakage loss dominates both AL and NAL, whereas, NAL dominate AL [85].

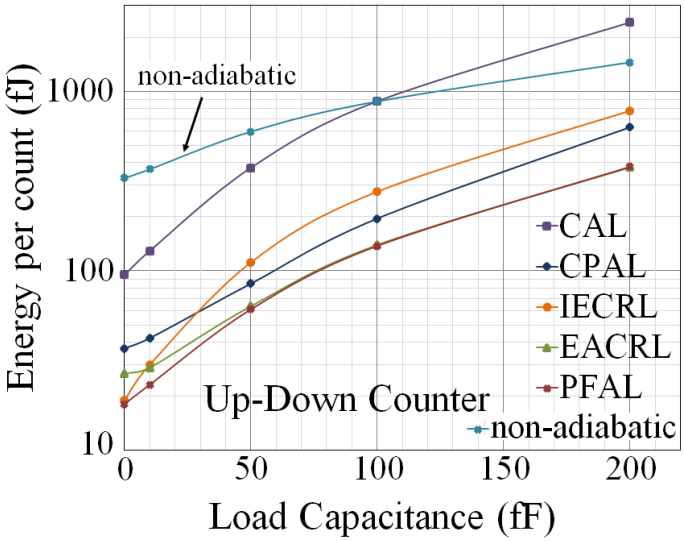
Due to the fact that each state of the CAL design takes four power-clock cycles, the energy performance of CAL counter is worst at all the ramping times as shown in Figure 3.19 (a). Similarly, the CPAL design, which takes two power-clock cycles for each count exhibits the second worst energy. In the case of the EACRL logic, since it has dual evaluation network (more number of transistors), the leakage losses dominate over AL and NAL at longer ramping times and thus increased energy compared to IECRL. The PFAL counter design shows the minimum energy at all the ramping times. IECRL and PFAL have the same number of transistor counts; however, as the former have higher output resistance and NAL it consumes more energy at the shorter ramping time. But as the ramping time is increased, its energy consumption drastically decreases and becomes lesser than that of EACRL at a ramping time longer than 30ns.

Figure 3.19 (b) shows the effect of loading on energy consumption of the Up-Down counter at the ramping time of 25ns. Though CAL is less complex, however, due to the low throughput of CAL sequential design, its energy is worst and even crosses the energy dissipation of the non-adiabatic at higher capacitance value which is mainly due to its high NAL. In Figure 3.19 (b), the increase and decrease in the energy dissipation of IECRL and EACRL respectively at 10fF load capacitance is because of the higher NAL of IECRL. On the other hand, at load capacitance higher than 100fF, the energy consumption of EACRL is exactly similar to that of PFAL. This is because as the load capacitance value increases, the effective load at the output nodes will mainly comprise of the load capacitance rather than its internal load capacitance. Both PFAL and

EACRL have a similar NAL however, due to more number of transistors in EACRL, it consumes more energy due to its dual evaluation network giving higher internal node capacitance at the output nodes compared to the other logic designs at lower load capacitance values.




(a)




(b)

**Figure 3.19:** Energy consumption versus (a) Ramping time (b) Load capacitance.

## 3.6 Performance Results

Based on the simulation results of flip-flop design, 2-bit twisted ring counter and 3-bit Up-Down counter the comparison of five adiabatic logic techniques are tabulated in

Table 3.2. The energy consumption in Table 3.2 is measured for ramping time 25ns at zero load capacitance value.

**Table 3.2:** Comparison of area and energy across the range of sequential circuit designs.

| **Adiabatic Logic Families** | **Proposed Resettable Flip-Flops** | | **2-bit resettable Twisted Ring Counter** | | **3-bit Up-Down counter** | |
|---|---|---|---|---|---|---|
| | **Area: No. of Transistors** | **Energy per cycle (fJ)** | **Area: No. of Transistors** | **Energy per state (fJ)** | **Area: No. of Transistors** | **Energy per count (fJ)** |
| IECRL | 27 | 2.50 | 54 | 3.55 | 189 | 19.10 |
| PFAL | 27 | 2.40 | 54 | 2.95 | 189 | 18.13 |
| EACRL | 28 | 3.80 | 56 | 6.13 | 222 | 26.88 |
| CPAL | 24 | 2.50 | 48 | 6.00 | 238 | 37.02 |
| CAL | 18 | 2.60 | 36 | 9.63 | 212 | 95.00 |

Table 3.2 shows that the CAL logic uses least transistors for designing sequential designs comprising only of buffer logic gates. However, for a more complex sequential logic (3-bit counter) comprising of combinational circuits and buffer, its area increases. On the other hand, the EACRL logic uses maximum transistors for the design of less complex sequential circuits which do not employ combinational logic gates, whereas, CPAL uses the maximum transistors for the design of 3-bit counter. However, for a 3-bit counter, design EACRL energy consumption is approximately 71% and 27% less compared to single-phase and 2-phase adiabatic logic respectively. CAL uses the least complex power-clock network, however, due to the auxiliary clock inputs, it has high latency and low throughput, therefore consuming maximum energy compared to the 2-phase and 4-phase adiabatic sequential logic designs. Similarly, the CPAL logic uses less complex power-clocking scheme, however, due to its high circuit complexity, it uses more transistors for designing the complex sequential circuits. All the 4-phase adiabatic designs have high complexity due to the power-clocking scheme, however, due to the complex evaluation network of the EACRL, its complexity and area requirement is maximum. Overall, out of the five adiabatic logic families, PFAL and IECRL prove to be a better choice in terms of energy consumption, area, and circuit

complexity (single evaluation network). Based on the simulation results tabulated in Table 3.2, the performance of the five adiabatic logic families in terms of Complexity and computation time is tabulated in Table 3.3.

**Table 3.3:** Comparison of complexity and throughput of the multi-phase adiabatic logic designs.

| Multi-phase Adiabatic Logic Design | Circuit Complexity | Power-clock Complexity | Computation Time |
|---|---|---|---|
| IECRL | Low | High | High |
| PFAL | Low | High | High |
| EACRL | Very High | High | High |
| CPAL | High | Medium | Medium |
| CAL | Medium | Low | Low |

## 3.7 Summary

This chapter explored the design of new resettable buffers. The resettable buffers are implemented for five multi-phase adiabatic logic families namely, IECRL, PFAL, EACRL, CPAL and CAL. The proposed novel adiabatic resettable buffers are used for the design of the resettable flip-flops. The performance of the proposed resettable flip-flops is compared to that of the existing MUX-based resettable adiabatic flip-flops. The proposed resettable flip-flops lead to decreased energy and area consumption compared to the existing MUX-based resettable adiabatic flip-flops. Using the proposed resettable flip-flops, a 2-bit twisted ring counter was designed using five adiabatic logic families as the design example for the performance evaluation. Overall, the PFAL based design of both the non-resettable and the resettable counters shows the minimum (approximately 2%) difference in energy consumption at all the ramping times.

Since the twisted ring counter does not contain any combination of logic thus, in order to facilitate the performance of the multi-phase adiabatic logic design, 3-bit Up-Down counter using five multi-phase adiabatic logic families were designed. The CAL logic design is worst in performance based on computation time, area (transistor count) and the energy consumption, however, its complexity (in terms of power-clocking scheme)

is lowest compared to the 2-phase and 4-phase power-clocking scheme. Similarly, the energy and area efficiency of CPAL decreases drastically for a more complex sequential circuit design (3-bit counter).

# 4 Design and Performance Trade-offs of Multi-phase Adiabatic Implementation of CRC Algorithm for NFC Application

Cyclic Redundancy Check (CRC) is ubiquitously common in all the communication protocol as it is an efficient way of detecting errors. In this chapter, the use of quasi-adiabatic logic techniques in implementing a 16-bit CRC design compatible with the ISO/IEC 14443A-3 [13] communication protocol for low energy Near Field Communication (NFC) application is presented. As the performance trade-offs of multi-phase quasi-adiabatic logic designs have already been evaluated in the previous chapter. This chapter re-investigates it by including robustness against Process Voltage Temperature (PVT) variations for the implementing CRC design using multi-phase adiabatic logic. A design methodology is proposed to minimize the design time and synchronization issue by implementing a CRC design which is suitable for a range of adiabatic clocking strategies, specifically 4-phase, 2-phase, and single phase. The CRC design is programmable for applications other than CRC due to its loadable initial value and CRC-16 generator polynomial. In addition, a system level implementation of CRC using adiabatic logic design including Power-Clock Generator (PCG) for different power-clocking strategies is implemented and compared based on energy consumption. In the end, the comparison of the multiphase adiabatic implementations and non-adiabatic implementation (conventional CMOS) is performed in terms of energy benefits, throughput, latency, complexity, robustness, and area.

## 4.1 Introduction

CRC is widely used in all data-communication, transmission and memory devices as a powerful method for detecting errors. One of the traditional hardware solutions for the CRC calculation is a bit-serial approach using a Linear Feedback Shift Register (LFSR) consisting of XOR gates and flip-flops [86], as shown in Figure 4.1. $G_1$, $G_2$,…$G_{n-1}$ are the generator polynomial, M(x) is the message and $CRC_0$ to $CRC_{n-1}$ are the calculated CRC values. The bit-serial approach has a low throughput since every n-bit data word requires n clock-cycles to calculate the CRC value. Depending on the application, a generator polynomial is used which gives a high probability of error detection [87]. For very high-speed data transmission, researchers have proposed numerous hardware and software-based CRC implementations. These include a parallel software implementation based on look-up algorithms [88] and hardware implementations based on z-transforms [89], matrix formulation [90]-[92] and pipelining [93]. These parallel approaches focus mainly on fast error detection when processing large data messages. These parallel approaches are mainly used for accessing storage devices when the data message is parallel or in the case when the fast data transfer rate is required such as in case of the fiber optic in a local area network. Software solutions have several drawbacks, for instance, they are slow, occupy processor resources, and requires ROM storage for the lookup table. Nevertheless, in all the references cited above, nothing has been mentioned about the energy consumption.

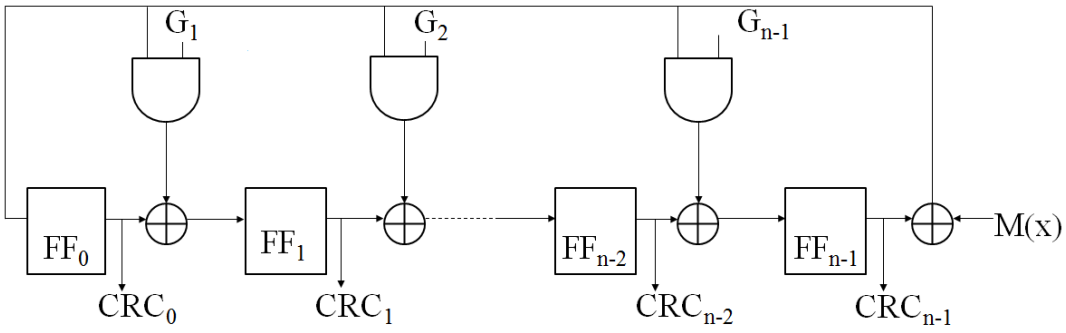



**Figure 4.1:** A bitwise serial LFSR for n-bit CRC generator.

In the literature, researchers have mostly demonstrated the low energy benefits of adiabatic implementations of counters [29], multiplexers and arithmetic units [82]. There exist very few papers [77], [39] which demonstrate the benefits of the adiabatic logic technique in a complex adiabatic circuit which also includes a power-clock

generator. It should be noted that adiabatic circuit/core combined with the power-clock generator is/will be referred as the adiabatic system. In literature, most of the papers demonstrated the energy benefits of various adiabatic designs/logic families over the non-adiabatic designs, however, there exists no work which compares the performance of the multi-phase adiabatic logic designs based on the adiabatic system architecture (array-based or iterative), energy consumption (with and without PCG), computation time, area, robustness (PVT variation), circuit and power-clock complexity. Practically, it is difficult to design an efficient adiabatic system however, trade-offs between energy, throughput, area, robustness, and complexity can be established that enables the designer to design efficient adiabatic logic systems.

Architectures can be designed either using an array-based approach or the iterative approach. An array-based approach consumes a large area due to the duplication of logic used in each stage. This is relaxed in the iterative approach, however, at the expense of high complexity due to synchronization problems. The majority of the designs follow the array-based approach, like a logarithmic signal processor using a single-phase power-clock [77] and a CORDIC based DCT using a 4-phase power-clock [39] as it is easy to synchronize the gates in the array-based approaches. There also exists designs which are iterative in nature like counters, CRC, etc. where the output is fed back to the input. For designing these systems, the designer needs to have a perfect understanding of the power-clock synchronization. Since adiabatic logic operates in the mid-frequency range favoring low-data-rate communication systems, the timing was never an issue. Because of the multi-phase adiabatic logic techniques, the area consumption, synchronization, and complexity have always been the challenges for adiabatic circuit designs. In a system design using an iterative approach where the control signal is used to trigger multiple blocks or modules like the counter unit, datapath unit, a register unit, etc, the design becomes tedious and cumbersome because of the synchronization problem. However, it saves a large amount of area and energy. Furthermore, if the iterative systems are designed properly, there is always a chance of reducing the synchronization buffers with the adiabatic logic gates.

## 4.2 ISO/IEC and ECMA

The international organization for standardization widely known as ISO is an international standard-setting body composed of representatives from various national

standards organizations. NFC system has been standardized by a number of globally accepted standard bodies. The first Radio Frequency (RF) Near Field Communication (NFC) standard was ECMA 340 [94], based on the Air Interface of ISO/IEC 14443A and JIS X6319-4.ECMA 340 was approved as the ISO/IEC 18092 standard [95]. In parallel major credit card companies (Europay, Mastercard, and Visa) have introduced the payment standard EMVCo based on ISO/IEC 14443A and ISO/IEC 14443B. Within the NFC Forum, both groups harmonized the air interfaces. They are named NFC-A (ISO/IEC 14443 A based), NFC-B (ISO/IEC 14443 B based) and NFC-F (FeliCa-based) [96].

The ISO/IEC 14443 standard is a four-part international standard for contactless smart cards operating at 13.56 MHz in close proximity (~10cm) with a reader antenna [11]-[13], [95]. This ISO standard describes the modulation and transmission protocols between the card and the reader to create interoperability for the contact-less smart card products. The ISO 14443 standard defines a protocol stack from the radio layer up to a command protocol as shown in Figure 4.2 (a). There are two versions of the radio layer ISO 14443-2 [12], with different modulation and bit encoding methods. These versions are known as the type A and type B versions of the ISO 14443 standard. Similarly, ISO 14443 specifies two versions of the packet framing and low-level protocol part such as initialization and anti-collision (ISO 14443-3) [13]. The topmost layer of the ISO protocol stack defines a command interface (ISO 14443-4) for transferring information.


(a) (b)

**Figure 4.2:** (a) ISO 14443 protocol stack (b) New command Protocol.

A new command protocol, NFCIP-1 [94] which replaces the topmost part of the stack of Figure 4.2 (a) is shown in Figure 4.2 (b). The peer-to-peer communication between two NFC devices is made possible by mechanisms defined in the Near Field Communication-Interface and Protocol specification, NFCIP-1. This key NFC

specification is also known as ECMA-340 [94] and ISO 18092standard [95]. NFCIP-1 includes two communication modes which allow an NFC device to communicate with other NFC devices in a peer-to-peer manner as well as with NFCIP-1 based NFC tags. The more in-depth details of the NFC specification can be found in [11]-[13] and [94]-[96].

## 4.3 Application of CRC in NFC

NFC is the emerging RF technology for short-range wireless communication that exchanges data between a reader, such as a phone or a sensor and a target such as another reader or a microchip embedded in a device. NFC is compatible with the most existing Radio Frequency Identification (RFID) and contactless smartcard system as it is an evolution of RFID and smartcard technology, however, its architecture is different in principle. RFID and contactless smartcards have a reader/tag structure. An NFC device can be both a reader (NFC-enabled device) and a target (NFC tag).

Two communication modes are supported by the NFC device; active and passive communication mode. The Frame Format of the data transmission in the NFC protocol of ISO/IEC 14443-3 and ECMA 340 standard are shown in Figure 4.3 contains 5 fields namely; preamble, SYNC, length, payload, and 16-bit CRC to check errors [94], [95].

| Preamble (48-bit min.) | SYNC (16-bit) | Length (8-bit) | Payload | CRC (16-bit) |
|---|---|---|---|---|

**Figure 4.3:** NFC Frame Format [94].

The process of encoding and decoding is carried out by a sixteen-stage cyclic register with an appropriate feedback and is based on ITU-T Recommendation V.41 [97] and the circuit is shown in Figure 4.1.CRC specifications for multiple bit-rate as per the standard for NFC Type-A [13] is tabulated in Table 4.1. The CRC frame is a function of k data bits which consist of all the data bits in the frame excluding the parity bits, the start of frame bits (Sof), end of frame bits (Eof) and a CRC bit itself. Since the data is encoded in bytes, the number of bits k; is a multiple of 8. For checking errors, two CRC bytes are sent in the standard frame [94] after the data bytes and before the Eof bits. The CRC calculation is cyclic which incorporates the current CRC value of the data (MSB first) and the CRC value of the previous data bytes. For large data blocks, the CRC

value from the preceding data byte is used as the starting value for the subsequent data byte. The LFSR carries a bit by bit multiplication in the Galois Field 2 (GF2) modulo. The division is then performed through shifting and feedback into the LFSR so that the result (CRC) is the value of the register once the whole message has been processed.

**Table 4.1:** CRC specification as given in ISO/IEC 14443 standard for NFC Type-A.

| Bit rates | Length | Polynomial | Pre-set value |
|---|---|---|---|
| 106 kbps (*fc*/128) | 16 bits | $x^{16} + x^{12} + x^5 + 1$ | '6363' |
| 212 kbps (*fc*/64) | 16 bits | $x^{16} + x^{12} + x^5 + 1$ | '0000' |
| 424 kbps (*fc*/32) | 16 bits | $x^{16} + x^{12} + x^5 + 1$ | '0000' |

The CRC calculation is cyclic, which incorporates the current CRC value of the data (MSB first) and the CRC value of the previous data bytes. Let M(x), G(x), Q(x) and R(x) represent the message polynomial, generator polynomial, quotient polynomial, and remainder polynomial respectively. The message, M(x) is a k-bit payload which is operated upon to form an n-k bit CRC detection block, where n is the length of the complete block. The algorithm for the CRC calculation for NFC is described in the following steps.

Step 1: The original k-bit payload, M(x) is multiplied by $x^{n-k}$ to shift the data and the pre-set value is appended.

Step 2: The result is then divided by the generator polynomial G(x) to form the quotient Q(x) and remainder R(x).

Step 3: The transmission polynomial T(x) is formed by appending the payload, M(x) and the remainder, R(x).

Step 4: At the receiver, the CRC calculation on the transmitted block, T(x) is done to check for errors in the transmission.

Step 5: After the transmission, the received message is processed with step 1 to 2 albeit with the received message replacing M(x). If the remainder, R(x) produced is zero, the transmission is assumed to be error-free.

The more detailed description of CRC algorithms is specified in [98]. The appending shall be done so that the bit ordering does not change. For example, as specified in

Annex B of ISO/IEC 14443-3 [13] and Annex A of ECMA-340 NFCIP-1 [94], the following message bit stream shown in Figure 4.4 produces the CRC value R(x) of the register 0xCF26 (least significant bits to most significant bit). The modified CRC algorithm for NFC application is given in Appendix A.

| | 1st data byte | 2nd data byte | 1st CRC byte | 2nd CRC byte | |
|---|---|---|---|---|---|
| Sof | 01001000 | 00101100 | 01100100 | 11110011 | Eof |
| | '0x12' | '0x34' | '0x26' | '0xCF' | |

**Figure 4.4:** Message stream and its corresponding CRC value.

## 4.4 Design Methodology

A modification in conventional LFSR which fulfills the criteria for an ISO/IEC-14443 standard is presented. Using the conventional CRC only a single bit-rate with an initial value of zeros can be loaded whereas; the proposed design is valid for a multiple data bit-rate and every initial load value. The proposed CRC design using adiabatic logic also has the flexibility to be used for different power-clocking schemes (single-phase, 2-phase and 4-phase) without modifying the design. Although, CRC is implemented for NFC-A application it can be easily modified to accommodate different CRC applications like mobile networks, Ethernet, USB, high-level data link control, etc [87]. A wide range of generator polynomials is presented in [88] along with their applications. With the proposed strategy, an n-bit CRC can be implemented by replicating n "slices" of circuitry. This approach enables CRCs of every number of bits to be readily created, thus decreases design time and synchronization issues [83].

An n-bit CRC is designed using n-blocks of CRC slices in the datapath. Each block of CRC slices has four logic gates connected in a cascade manner. Out of the n-blocks, n-1 are identical having the same logic gates connected in the same order. However, the Least Significant Bit (LSB) of the CRC slice has the position of the XOR gate different than that of the identical blocks. This is due to the synchronization of the feedback signal with the input message bits. A single block slice requires three stages or phases of the power-clock but due to the iterative nature of the CRC implementation, the number of stages should either be a multiple of two in the case of single-phase (because of auxiliary clock signals) and 2-phase designs or a multiple of four in the case of 4-

phase logic designs. Thus, a buffer is added in each slice to have an even number of stages for correct synchronization and functionality. Each slice in the CRC datapath implemented using 4-phase adiabatic logic takes one power-clock cycle, whereas single-phase and 2-phase designs take four and two power-clock cycles respectively. The controller generates the control signals for the CRC design. The CRC starts the computation when the signals 'New message' and 'R_count_H' are logic '1'. The input message, M(x) is provided to the CRC datapath serially using a multiplexer used as a test circuitry for the CRC design. To synchronize the CRC computation and the input message, the counter outputs act as the select lines for this multiplexer which provides serial input to the CRC datapath and the register unit.

The speedup technique is used as described in [89] to increase the throughput. The buffers in the counter are replaced with the functional logic gates (AND/OR/XOR). Thus, the throughput and latency of 4-phase designs are improved by ½ of the power-clock cycle whereas, in the case of 2-phase and single-phase CRC design, an improvement of one power-clock cycle and two power-clock cycles respectively is achieved. In addition, the technique also reduces the buffers required for synchronization by four in the counter unit.

For a message word-length of 16 bits, the 16-bit CRC datapath requires 64 power-clock cycles using a single-phase power-clocking scheme, whereas, 32 and 16 power-clock cycles are required for 2-phase and 4-phase adiabatic logic respectively. In general, for the message word-length of k-bits, an n-bit CRC datapath requires 4k, 2k and k power-clock cycles for single-phase, 2-phase and 4-phase adiabatic logic designs respectively. Where k is always greater than or equal to n. Since the presented work is in accordance to ISO/IEC 14443 standard for NFC, a 16-bit CRC is designed based on the methodology and strategy used in describing n-bit CRC. The CRC is implemented using all the five adiabatic logic families and tested for its functionality and robustness against PVT variations. All the components including the multiplexer (providing input serially) are designed using adiabatic logic.

## 4.5 Hardware Implementation of 16-bit CRC using Adiabatic Logic

The typical CRC is implemented using Linear Feedback Shift Register (LFSR) having serial input message data bits. A block diagram of the 16-bit CRC design is shown in Figure 4.5. All the adiabatic logic designs have differential input and output signals,

however, for the simplicity and better understanding, complementary signals are not shown in Figure 4.5. The complementary signals are denoted by '_L' added at the end of all the signals used denoting active low signals. The advantage of the proposed architecture is that it can be used for multi-phase power-clocking scheme design. The designer only has to replace the unit with the specific adiabatic logic style with an appropriate power-clocking scheme. The main part of the CRC design is its datapath which is responsible for computing the CRC value. The 16-bit input message (M(x)) is provided to the CRC datapath through a 16:1 multiplexer at every count of the counter. To be consistent with the protocol, MSB is the first bit transmitted as shown in Figure 4.5. Since each block in the datapath has a latency of 4 power-clock phases, a delay cell is added at the output of the 16:1 multiplexer to synchronize the final CRC values from the CRC datapath and the input message, M(x). Finally, the final CRC value gets appended at the end of the message bits using a 32-bit register unit. The functionality of the proposed CRC architecture is verified by taking the example specified in Annex B of the ISO/IEC 14443-3 protocol [13].

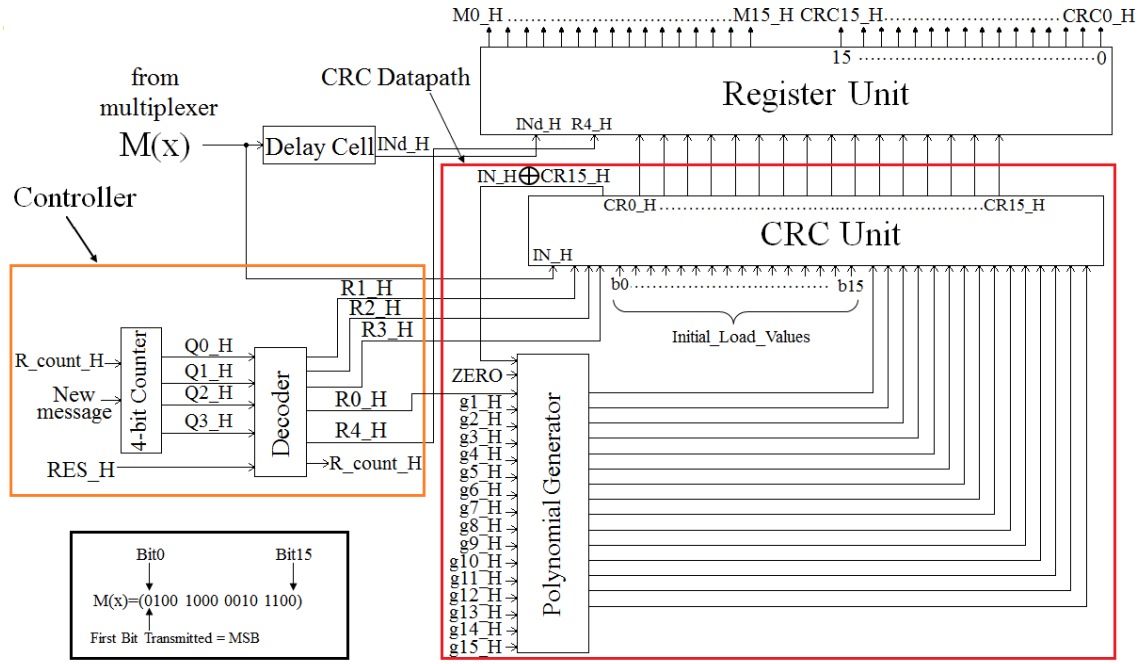



**Figure 4.5:** Block diagram of the 16-bit CRC Design and its message, M(x) format.

CRC is initialized using the reset input 'RES_H' which clears the CRC unit, the register unit, the controller unit, resets the counter and load the pre-set value '0x6363'. When 'RES_H' signal is set low and the 'new message' bit is logic '1', the counter starts

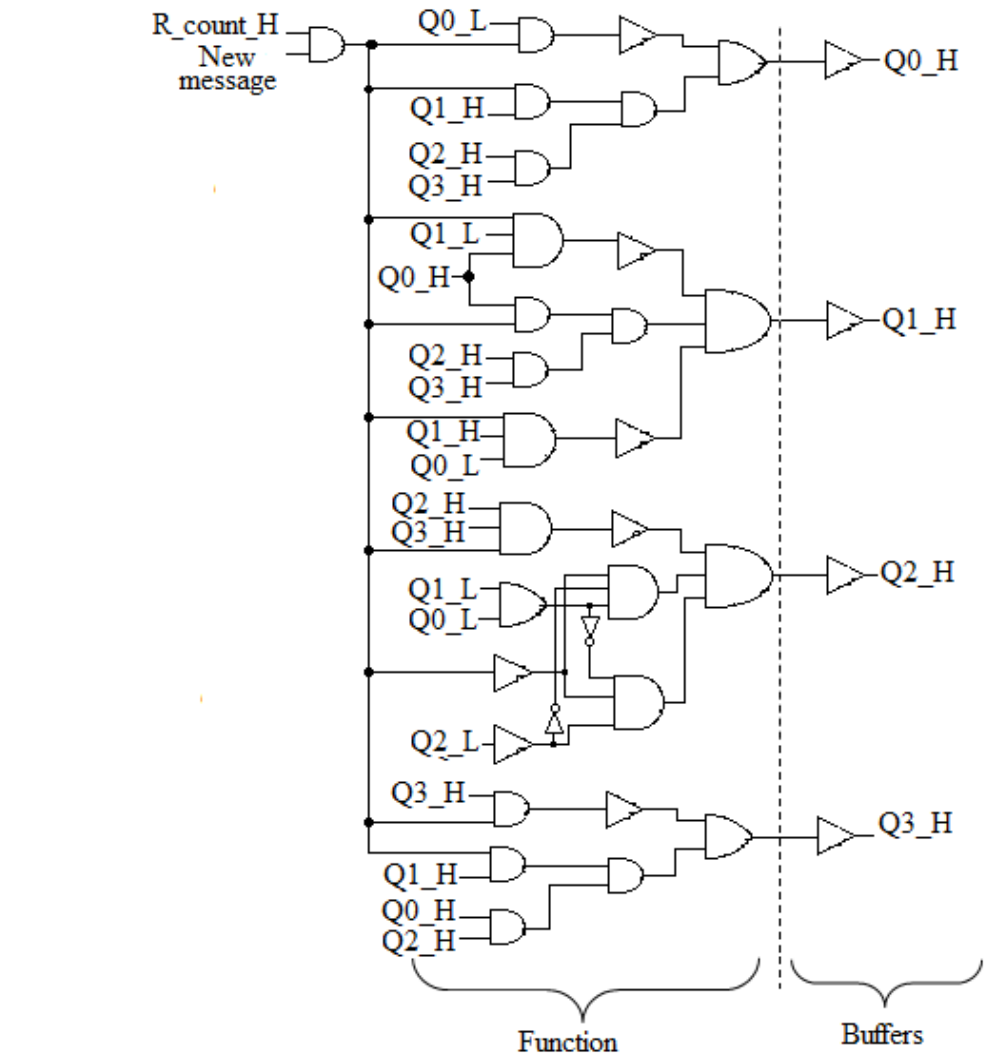
counting. With every count, the message (M(x)) is serially sent to the CRC datapath for the computation. At the same time, the message bit is stored in the register unit after a delay of 1 power-clock cycle. The final CRC value is calculated when the last message bit is sent to the datapath, and the counter reaches the value '1111'. Then the calculated final CRC value from the datapath gets appended to the register unit with the message while the counter returns to the value '0000'. The appended CRC value and the message word are retained during the wait period in the specially designed register unit, while the values in the CRC datapath are cleared to zero. The wait period lasts for two power-clock cycles and after that, the counter starts counting again automatically allowing the CRC to re-calculate its value. To calculate the new CRC value either the new message bits or the generator polynomial along with the load values can be provided during the wait period.

The CRC design has a number of advantages. Firstly, it can be used for different power-clocking schemes. Secondly, all the control signals remain the same for multi-phase adiabatic logic designs. Thus, the designer has only to pick the required adiabatic logic family and replace the gates with their chosen adiabatic logic family gates saving design time and eliminating synchronization issues. Thirdly, the use of a polynomial generator unit and initial load value makes it reusable for other applications of 16-bit CRC.

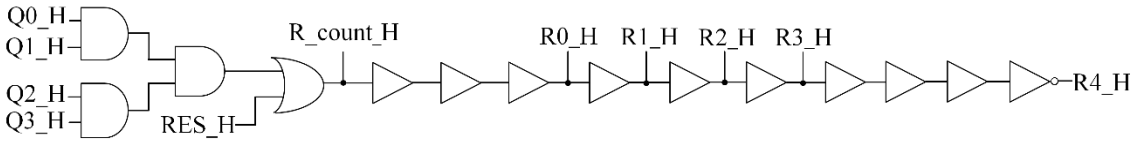
In order to have the reusable CRC design for multi-phase clocking scheme and for applications other than NFC, the implementation has associated hardware cost. Firstly, the generator polynomial unit incurs an area overhead of twelve 2-input AND gates and twelve 2:1 multiplexer. Secondly, for the CRC designs using multi-phase adiabatic logic, the register unit of the single-phase and 2-phase implementations use approximately 50% more buffers.

The controller comprises a counter which generates the states and a decoder (combinational logic) that generates the synchronization signals for the CRC. The counter is designed using D flip-flops. It has two inputs, 'R_count_H' (coming from the decoder) and the 'New message'. The 'New message' input is an active high external input. Initially, it is zero when the counter is in the reset state. The counter starts counting when both the 'New message' and 'R_count_H' signal values are logic '1'. In

### *4.5.1 Controller (Counter and Decoder)*




(a)




(b)

**Figure 4.6:** Controller (a) 4-bit Counter (b) Decoder.

general, the adiabatic D-flip-flop is structured using a cascaded buffer chain, but in this case, the buffers are replaced with the logic gates (AND/OR/XOR) which saves exactly twelve buffer gates. For the test purpose, the 16-bit new message is provided to the CRC datapath using 16:1 multiplexer. Figure4.6 (a) shows the functional part and the

synchronization buffers used in the 4-bit counter. The inputs 'Q0_L', 'Q1_L', and 'Q2_L' are the complementary signals of 'Q0_H', 'Q1_H', and 'Q2_H' which are not shown for simplicity.

The outputs of the counter 'Q0_H', 'Q1_H', 'Q2_H, and 'Q3_H' are the inputs to the decoder along with the external reset input 'RES_H' as shown in Figure 4.6 (b). The decoder automatically provides activation reset signals ('R_count_H', 'R0_H', 'R1_H', 'R2_H') to the counter and the datapath. The signal, 'R0_H' is the input to the AND gate in generator polynomial bit blocks of the CRC datapath. Whereas, the signal, 'R3_H' is the select input for the 2:1 multiplexer in the CRC bit blocks which selects the initial load value when logic '1'. The new generator polynomial along with the load values can be provided during the wait period. The signal, 'R4_H' is an inverted signal of 'R3_H' which is delayed by four buffer gates. It is used as a wait signal to the register unit that generates a wait period of two power-clock cycles. The decoder performs three tasks; firstly, it generates a retain signal which helps to retain the final CRC value in the register unit. Secondly, it reset the CRC datapath, the counter unit, and the register unit before the computation begins and after the final CRC value is computed. Lastly, the buffers in the decoder serve the purpose of synchronizing the decoder output with different units of the CRC design for correct calculation of the CRC value.

The use of the signals from the decoder makes the CRC design to calculate the CRC value continuously until the counter reaches the value '1111'. Because each bit blocks in the CRC unit is having four logic gates connected in the cascade manner, the implementation of the controller remains fixed for all the power-clocking schemes.

### *4.5.2 CRC Datapath*

The CRC design is based on the serial LFSR design [86] which has been modified in accordance with the specification outlined in ISO 14443-3 type A protocol. The CRC datapath consists of the CRC unit and the generator polynomial unit. The CRC unit computes the CRC value based on the generator polynomial (g1_H…..g15_H). The generator polynomial, G(x) for NFC applications, is $x^{16}+x^{12}+x^5+1$. A wide range of generator polynomials is presented in [89] along with their applications. Since the binary value of the MSB and LSB of the generator polynomial is always one, the polynomial generator unit consists of fifteen 2-input AND gates each followed by 2:1

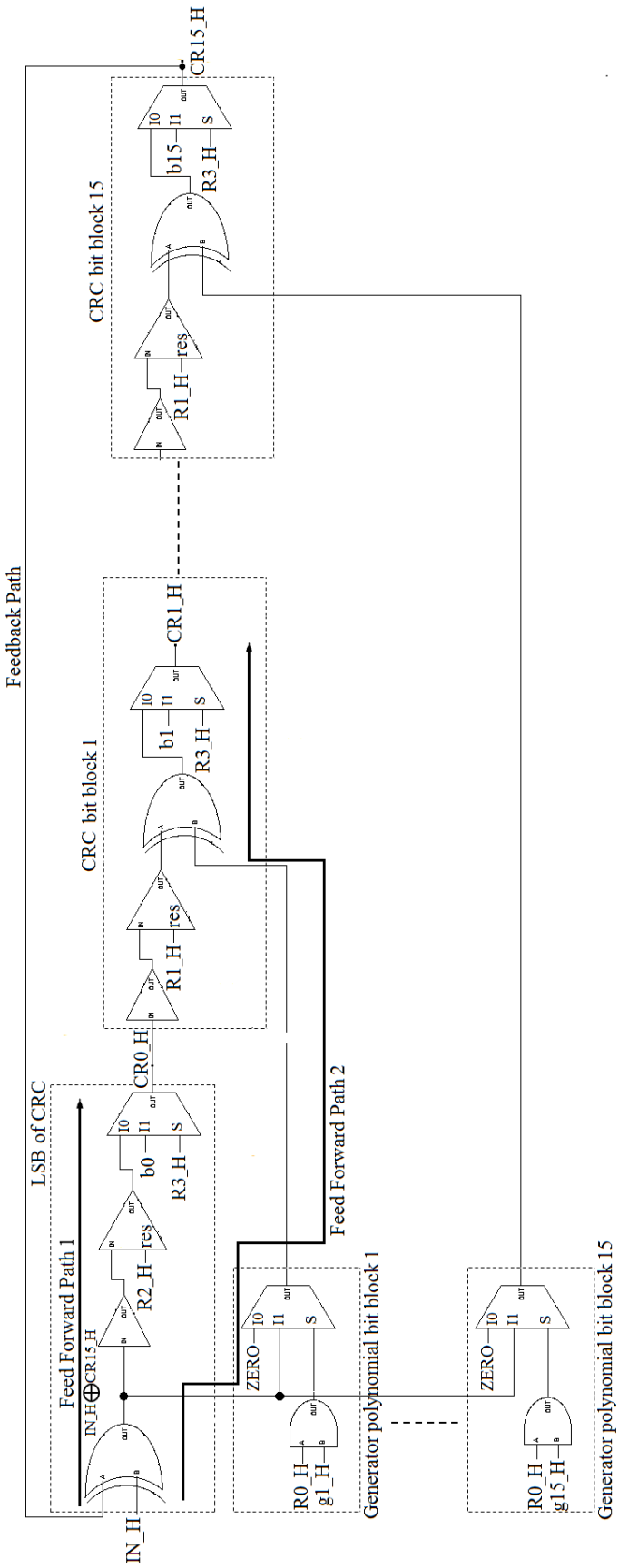



**Figure 4.7:** CRC Datapath

multiplexers. The hex value '0x8810' corresponds to G(x) ('g1_H', 'g2_H', ……., 'g15_H') is fed along with the reset signal, 'R0_H'. The output of the AND gate triggers the multiplexer to select either a zero or the XOR function of the input message bit with the MSB bit of the CRC Unit ('CR15_H') as shown in Figure 4.7. The outputs from the generator polynomial bit blocks are fed into the XOR gates of the respective CRC bit blocks.

A 16-bit CRC has sixteen-bit blocks with one LSB bit block and fifteen identical blocks (1 to 15) as shown in Figure 4.7. Each identical block uses four logic gates which incorporate a synchronization buffer, a resettable buffer for resetting the datapath, XOR gate for generator polynomial representation and 2:1 multiplexer for initial bit loading for different bit-rates (b0, b1,…., b15). The initial load value (0x6363) is loaded in the CRC datapath during reset operation when 'R3_H' signal is logic '1'. Two different resettable signals, 'R1_H' and 'R2_H' are used to synchronize the CRC unit due to the different position of the resettable buffers in the CRC bit blocks and the LSB of CRC. The design can be reused either for a higher bit or for lower bit CRC depending on the application by adding the identical CRC bit blocks or by eliminating it. Figure 4.7 shows two feedforward paths and a feedback path. Both the feedforward paths comprise of four cascade gates. Since the feedforward path 2 has a fixed latency of four logic gates (two XOR gates and two MUXs), a buffer is added in the feedforward path 1 for synchronization. Thus, the n-bit CRC datapath implementation has a fixed overhead of n-buffer logic gates due to the synchronization.

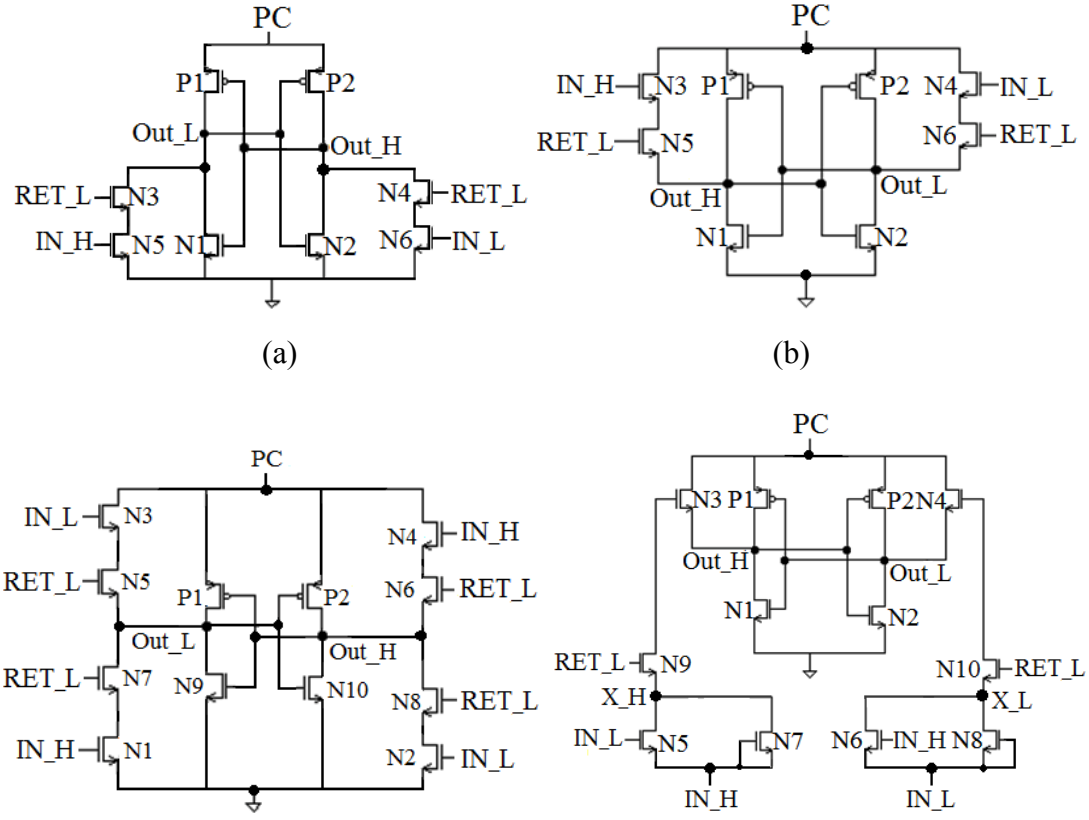
The same concept is applied for CRC implementation using single-phase and 2-phase power-clocking scheme. Thus, all the multi-phase logic designs use the same design with the same signals as shown in Figure 4.5. The overhead in terms of synchronization in implementing the datapath of $2^n$ bit CRC is $2^n$ buffer gates.

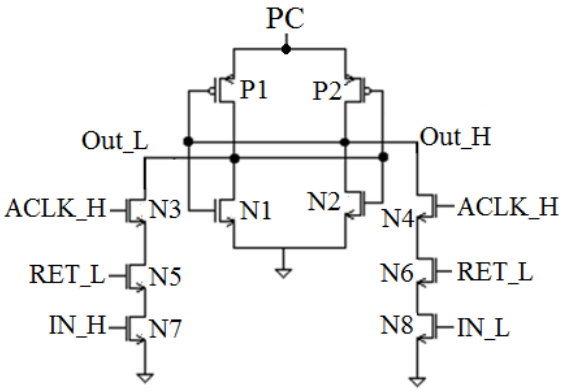
### *4.5.3 Register Unit*

The CRC value is appended to a message bit stream in the register unit. Typically, a message bit stream is stalled using a delay cell comprises of four adiabatic buffer gates to synchronize it with the CRC value which has a latency of four gates. A single-bit register comprises of four buffer logic stages connected in a cascade manner. The first three stages consist of a buffer logic (shown in Chapter 3 of this thesis) and the last stage consists of a novel retain buffer logic. Figure 4.8 shows the retain buffer logic

circuits for all the five adiabatic logic families. The 'RET_L' is an active low input. It performs a function of retaining the final CRC value using the wait signal, 'R4_H' from the decoder. As soon as the computation is over, the 'RET_L' input signal is logic '0', cutting-off the two output nodes from the power-clock and the ground respectively. Thus, the logic value gets retained because of the cross-coupled nMOS and pMOS transistors.

In the case of EACRL logic (having dual-evaluation network), duplicate retain transistors were insufficient. It is because the logic suffers from the coupling effect due to the absence of nMOS cross-coupled transistors where both the output nodes get coupled when 'RET_L' input goes low. Thus, two extra cross-coupled nMOS transistors, N9 and N10 are used as shown in Figure 4.8 (c). The cross-coupled transistors pair P1, N9, and P2, N10 reduces the coupling effect and helps in providing the complementary output signals at the two output nodes. Conventionally, to construct a 1-bit register using a single-phase and 2-phase adiabatic logic, two buffer stages are required (see Chapter 3 of this thesis). However, due to the synchronization issue and using the design for the multi-phase power-clocking scheme, the number of stages used in a single bit register of CRC is twice the conventional case.






(e)

**Figure 4.8:** Adiabatic retain logic (a) IECRL (b) PFAL (c) EACRL (d) CPAL (e) CAL.

## 4.6 Simulation Results

For meaningful simulations and to compare CRC implementation using different adiabatic logic designs, the transistor sizes were set to the technology minimum for high energy efficiency [90]. The simulations were done using Spectre simulator in Cadence EDA tool based on TSMC 180nm CMOS process technology at 'Typical-typical (TT) process corner.

For a single-phase and 4-phase adiabatic logic designs, each power-clock is generated using the trapezoidal wave, ramping from 0 to $V_{DD}$, having an equal duration of *Evaluation (E), Hold (H), Recovery (R) and Idle (I)* periods as shown in Figure 2.5 of Chapter 2 of this thesis. The ramping time ($T_r$) of the power-clock is one-quarter of the power-clock time-period ($T_{CLK,1\text{-phase/4-phase}}$). In the case of 2-phase clocking scheme, due to the non-overlapping requirement of the power-clock the *Idle period* is three times that of each *Evaluation*, *Hold* or *Recovery period*. Hence the ramping time ($T_r$) of the 2-phase power-clock is one-sixth of the power-clock time-period ($T_{CLK,2\text{-phase}}$). Because the adiabatic and non-adiabatic designs do not share the same ramping time, the clock frequency of the non-adiabatic implementation is chosen such that its frequency of operation is same as that of an adiabatic implementation keeping the rise time and fall time constant across the chosen frequency range. For example, for a ramping time of 2.5ns, the time period of one power-clock cycle is 10ns thus, the clock period for the non-adiabatic implementation is taken as 10ns with constant rising and falling time of 10ps. To measure the energy dissipation and avoiding excessive data dependencies, the average energy per computation was measured for ten random message input

combinations. It was measured at various frequencies ranging from 1MHz to 100MHz, load capacitances, supply voltage scaling and PVT variations for all the five adiabatic and non-adiabatic CRC implementations. Also, the computation time in terms of power-clock-cycles for various message word-lengths was extrapolated. In the end, a comparison at the adiabatic system level including PCG and between adiabatic logic families was performed and energy saving percentage was calculated for each of them.

### *4.6.1 Impact of Frequency on Energy Dissipation*

The energy per computation at varying power-clock frequencies was measured for an output load capacitance of 10fF connected at the output of the register unit. Here, the energy per computation implies to the energy dissipated in one complete computation (i.e. generating the final CRC value after all the message bits have been sent). Figure 4.9 shows that the energy of all the adiabatic implementations outperforms the non-adiabatic implementation and show significant energy benefits compared to conventional CMOS. Energy Saving (*ES*) is calculated and is defined as the difference in the energy consumption of non-adiabatic and adiabatic implementations divided by the energy dissipation of the non-adiabatic implementation. The formula for "Energy Saving Percentage" (*ESP*) is given by (4.1)

$$ESP = \frac{E_{NA} - E_{TD}}{E_{NA}} \times 100 \qquad (4.1)$$

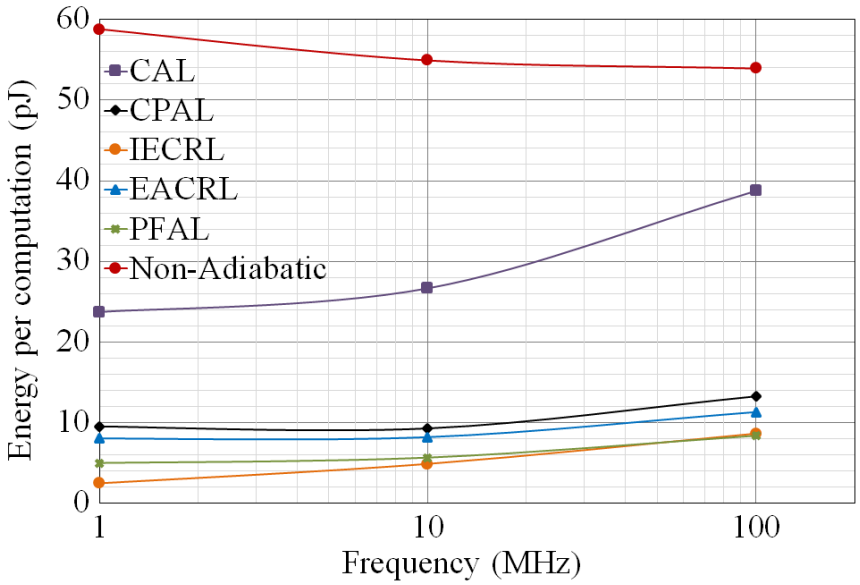



**Figure 4.9:** The energy per computation of the 16-bit CRC for a 16-bit message length at varying frequency

In the calculation of the energy saving, the energy dissipation of PCG is not included. Out of the five adiabatic logic designs, PFAL exhibits the maximum ESP of approximately 84.5% at 100MHz, whereas, at 10MHz and 1MHz frequencies, IECRL implementation exhibits the maximum ESP of approximately 91% and 96% respectively. The energy consumption per computation and the ESP of the five adiabatic logic families at frequencies simulated are reported in Table 4.2.

**Table 4.2:** Energy per computation for adiabatic logic families and non-adiabatic implementation at the frequencies simulated.

| **CRC Implementation** | | ***Frequency (MHz)*** | | |
|---|---|---|---|---|
| | | ***1*** | ***10*** | ***100*** |
| Non-adiabatic | Energy (pJ) | 58.81 | 54.93 | 53.93 |
| CAL | Energy | 23.72 | 26.65 | 38.74 |
| | ESP | 59.67 | 51.48 | 28.17 |
| CPAL | Energy | 9.51 | 9.27 | 13.25 |
| | ESP | 83.83 | 83.13 | 75.43 |
| IECRL | Energy | 2.46 | 4.87 | 8.63 |
| | ESP | 95.83 | 91.13 | 83.99 |
| EACRL | Energy | 8.02 | 8.19 | 11.30 |
| | ESP | 86.36 | 85.10 | 79.05 |
| PFAL | Energy | 4.98 | 5.65 | 8.37 |
| | ESP | 91.54 | 89.71 | 84.48 |

The single-phase, CAL design is least beneficial in comparison to the other adiabatic implementations. Unlike the adiabatic logic using 2-phase and 4-phase power clocking schemes, in a single-phase cascaded CAL logic, the inputs from the previous stages always have the same phase as the power-clock, except with a small delay. As the wait signal, 'R4_H' is connected to the 'RET_L', the input of 32 retain transistors, and the propagation delay increases as the power-clock speed is increased (shorter ramping time). As a result, the input reads the wrong value which gets propagated to the register outputs. Hence, for a shorter ramping time (higher frequency), the sizing of the logic gate generating the 'R4_H' signal in CAL controller was done leading to increased energy dissipation. On the other hand, at the simulated frequencies, the IECRL design shows the minimum energy per computation at frequencies lower than 25MHz approximately whereas, PFAL consumes the minimum energy above 25MHz.

### *4.6.2 Impact of Load Capacitance on Energy Dissipation*

Figure 4.10 shows energy per computation at varying load capacitances at 10MHz. It can be seen that the variations in the energy dissipation of CAL and the IECRL logic with load variation are steeper as compared to the rest of the logic designs presented. At load capacitance greater than 60fF, IECRL crosses the energy dissipation of CPAL and becomes the second worst (after CAL). Out of the five adiabatic logic designs, the CAL implementation consumes the maximum energy. It is also worth mentioning that the non-adiabatic design outperforms the CAL logic at load capacitance values greater than 100fF. On the other hand, PFAL consumes the least energy at load capacitance values greater than 20fF. However, the advantage of the low energy consumption of the 2-phase CPAL logic (due to zero NAL at the two output nodes) diminishes mainly because of the high computation time incurred by the CRC datapath.

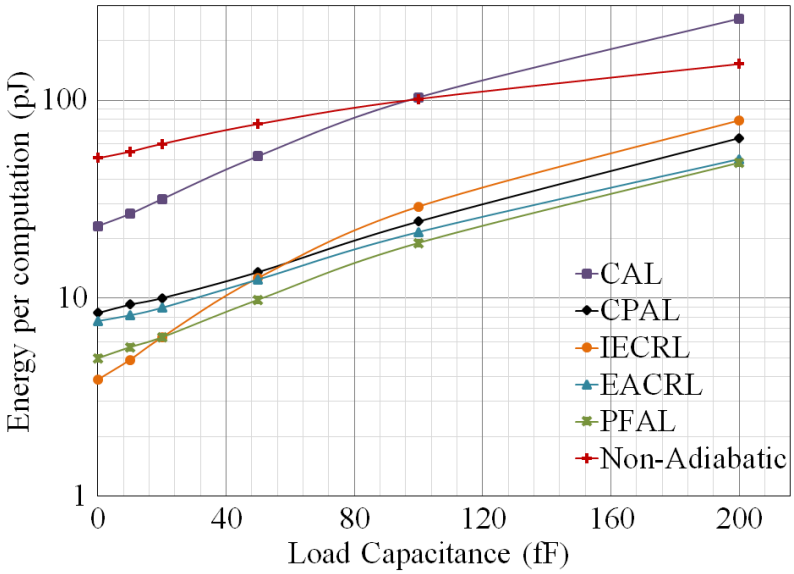



**Figure 4.10:** The energy per computation at varying load capacitances.

Considering the EACRL design, it dissipates more energy in comparison to PFAL and IECRL at lower capacitive load as shown in Figure 4.10. However, as the load increases beyond 50fF, the advantage of zero NAL in the evaluation phase overpowers its disadvantages of higher input/output node capacitances (due to dual evaluation logic) and the coupling effect. Thus, it dissipates less energy than that of IECRL at higher capacitive loading. In addition, when compared to PFAL, due to more number of transistors, EACRL consumes approximately 55% more energy at zero load capacitance. But at 200fF, the load capacitance dominates the internal node capacitance of EACRL

and consequently, the difference in the energy dissipation of PFAL and EACRL reduces. EACRL dissipates approximately 4.3% more energy than PFAL.

### *4.6.3 Impact of Supply Voltage Scaling on Energy Dissipation*

Energy in both adiabatic and non-adiabatic implementations can be reduced by supply voltage scaling according to the quadratic dependence of the energy dissipation on the supply voltage (2.2) and (2.6).

However, in adiabatic logic, reducing $V_{DD}$ also increases the ON-resistance, $R_{ON}$, of the transistor in the charging path (4.2), thus increases the energy dissipation [38]. Hence, the energy benefits of the reduced supply voltage in adiabatic circuits are less.

$$R_{ON} = \frac{1}{K\left(V_{GS} - V_{th}\right)} \tag{4.2}$$

Where $K = \frac{\mu\, C_{ox} W}{L}$. As long as $V_{DD}$ is above than $V_{th}$, the energy dissipation in an adiabatic logic is given by

$$E_{AL} = \frac{C_L^2 V_{DD}}{KT_r}\left(\frac{V_{DD}}{V_{DD} - V_{th}}\right)$$

$$E_{AL} = \frac{C_L^2 V_{DD}}{KT_r}\left(1 + \frac{V_{th}}{V_{DD}}\right) \tag{4.3}$$

Assuming negligible NAL and leakage, $E_{TD} = E_{AL}$ then substituting (2.6) and (4.3) in (4.1), the effect of voltage scaling on *ES* in an adiabatic circuit can be derived (4.4).

$$ES = 1 - \frac{\beta}{V_{DD}}\left(1 + \frac{V_{th}}{V_{DD}}\right) \tag{4.4}$$

Where; $\beta = \frac{2C_L}{\alpha K T_r}$

Figure 4.11 shows the effect of voltage scaling on energy per computation for five adiabatic and non-adiabatic CRC implementations at 10MHz and at 10fF load

capacitance. From (4.4) and Figure 4.12, it can be seen that the adiabatic techniques largely suffered from voltage scaling in terms of ESP and functionality. PFAL and IECRL show a similar reduction in *ESP* as the voltage is scaled down, except the fact that the former fails to deliver the correct functionality at 0.6V (voltage closer to the threshold voltage). Also, due to the higher voltage drop of pass transistors in CPAL, it malfunctions at 1V and less. Thus, it makes CPAL highly vulnerable logic at lower voltages. As expected, CAL shows minimum ESP and goes below zero, approximately 5% at 0.6V. This implies that the energy dissipation of the non-adiabatic implementation becomes less than that of CAL design.

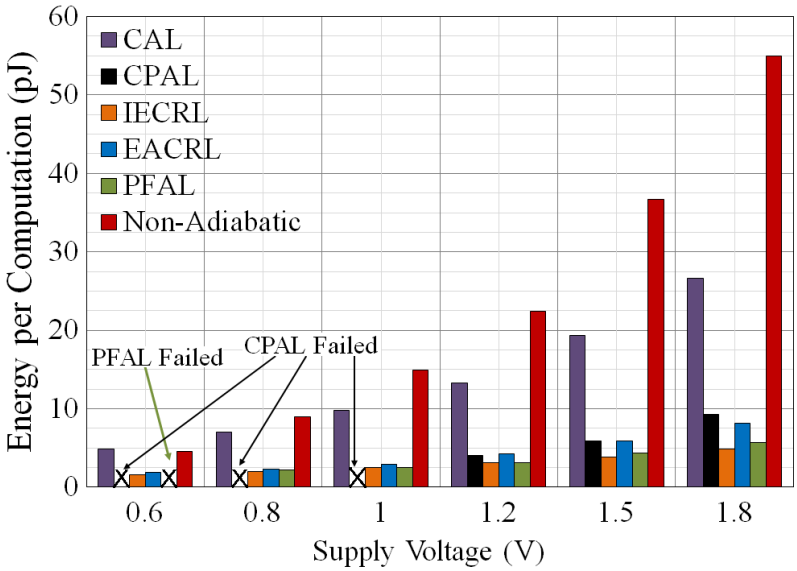



**Figure 4.11:** Energy per computation at the varying supply voltage.

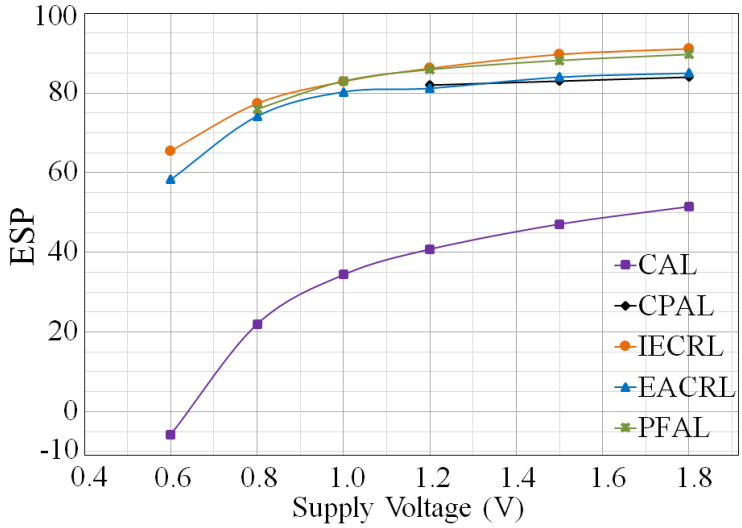



**Figure 4.12:** ESP at the varying supply voltage.

It can be summed up that, *ESP* of adiabatic logic designs shows a steeper response at supply voltage less than 1.2V. In addition, the reduction in supply voltage will also degrade the noise margin both in non-adiabatic and adiabatic implementations. Thus, for the adiabatic logic families, an optimal range for the supply voltage scaling is proposed. It is named as "Adiabatic Voltage Scaling Range" for better ESP and proper functionality and is stated as;

$$V_{DD} \geq 2V_{th} \tag{4.5}$$

### *4.6.4 Impact of Process Voltage Temperature (PVT) variation on the Energy Dissipation*

The robustness of the CRC design using adiabatic logic against PVT variations is investigated by running the PVT analysis in Analog Design Environment (ADE). All the CRC implementations were simulated for five corners to ensure correct operation. Figure 4.13 shows the energy per computation measured for the adiabatic and non-adiabatic designs at 10MHz and 10fF load capacitance.

Temperature plays an important role in the energy dissipation of the adiabatic circuit due to the dependency of on adiabatic energy dissipation on $R_{ON}$. The increase in temperature causes $R_{ON}$ to increase, causing the adiabatic logic to dissipate more at a higher temperature. The worst-case energy dissipation was measured for the Fast-Fast (FF) process corner at a 1.98V supply voltage and 100ºC temperature. Similarly, for the best case, slow-slow (SS), 1.62V and 0ºC were considered. Whereas for the skewed corners slow-fast (SF), and fast-slow (FS), the designs were simulated for 1.62V and 100ºC temperature giving energy dissipation close to the SS corner and for the FS corner 1.98V and 0ºC, close to the FF corner. For typical-typical corner (TT), 1.8V and 27ºC temperature is the default value.

In SF corner, the CAL implementation malfunctions, therefore its energy dissipation is not measured. On the other hand, CPAL design shows large variations in the energy consumption at extreme corners (FF and SS) compared to the other adiabatic logic designs presented. However, out of the five adiabatic CRC implementations, PFAL and EACRL show constant ESP approximately 90% and 85% respectively at all process corner. Whereas IECRL shows ESP of 85% at FS corner and 91% at rest of the four process corners.

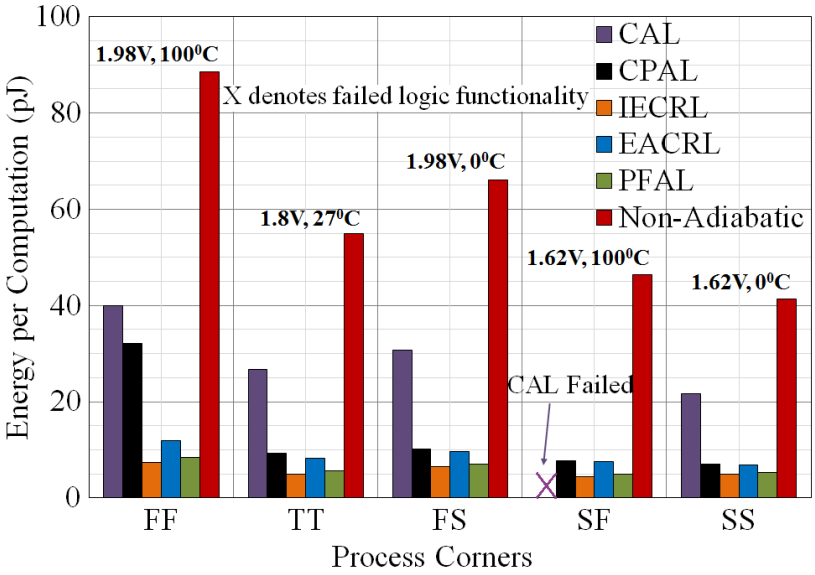



**Figure 4.13:** Energy per computation at five process corners.

### *4.6.5 Impact of Message Word-Length on Computation Time*

The datapath of the CRC for all the 4-phase and 2-phase 16-bit CRC designs, take 64 power-clock phases for the computation of 16-bit message word-length. An additional seven phases, four for the counter and three for 16:1 multiplexer are required for the message bits to arrive at the input of the CRC datapath. Another four phases are required by the CRC value to be appended with the message word in the register unit. Thus, the total of 75 power-clock phases equivalent to 18.75 power-clock cycles is required by the 4-phase designs for CRC computation. Whereas, for the 2-phase design, 37.5 power-clock cycles are required for the complete computation. Although the single-phase design has the lowest power-clock complexity, however, it requires 75 power-clock cycles in total. Therefore, resulting in the lowest throughput and highest energy dissipation.

The non-adiabatic design requires 18 clock-cycles, approximately 3/4th less as compared to that required by the 4-phase adiabatic logic designs. This is because the adiabatic implementation of the multiplexer test circuit requires three power-clock phases whereas non-adiabatic requires none. Figure 4.14 shows the extrapolated result of the computation time at varying message word-length using the multi-phase power-clock designs and the non-adiabatic design for the 16-bit CRC code.

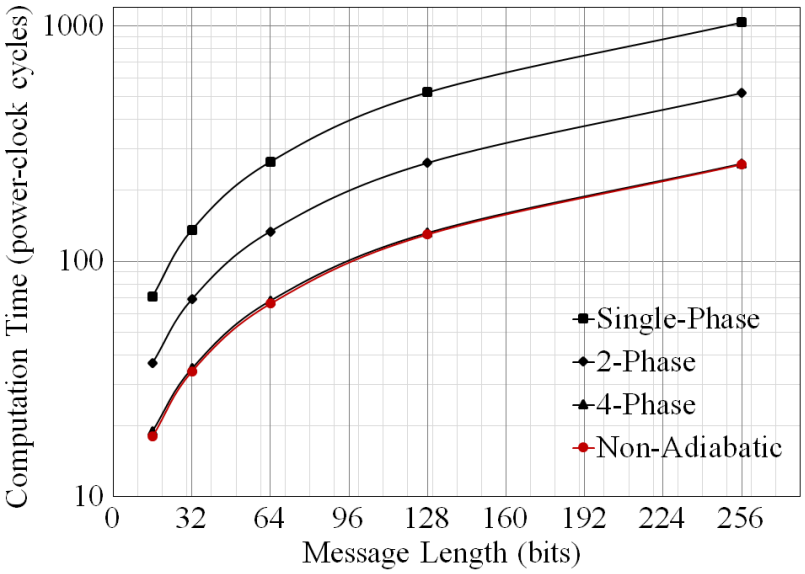



**Figure 4.14:** Computation time versus message bit length.

### *4.6.6 Power-Clock Generation (PCG)*

Unlike static CMOS logic, adiabatic circuits are powered from the clock, requiring a separate "power-clock" supply. PCG will consume a significant amount of the energy (analogous to the clock generation in conventional CMOS). It is important to bear in mind that PCG will be able to supply considerably more circuitry than the CRC presented here. Nevertheless, it is appropriate to consider its energy too, which is often neglected in adiabatic papers present in the open literature. Here, a 4-phase PCG based on StepWise Charging (SWC) circuit is used, as found in [51]. The complete adiabatic system was designed which comprises of the power-clock generator and the adiabatic core.

The adiabatic core contains the CRC. The required power-clock phases come from PCG. Single-phase, 2-phase, and 4-phase power-clock generators were designed using 2-step charging circuit. To generate 2-phase power-clock two 2-step charging circuits were required. Similarly, for 4-phase power-clock, four 2-step charging circuits were required. For a single-phase, only one 2-step charging circuit was required and the auxiliary clocks were supplied using a trapezoidal power source. What also has to keep in mind that generating power-clock of the same ramping time for 2-phase and single/4-phase clocking scheme, the power-clock frequency is different.

The simulations were performed for a ramping time of 25ns for the power-clocks with supply-voltage 1.8V $V_{DD}$ and 10fF capacitive load attached to the output of the adiabatic core. The reference CLK for generating the power-clock frequency of 25ns ramping time (10MHz) was taken to be 40MHz and 60MHz for single/4-phase and 2-phase clocking schemes respectively. The frequency of operation for non-adiabatic was taken to be 10MHz. The value of the tank capacitance used in the 2-stepwise charging circuit of the single-phase, 2-phase, and 4-phase PCG was 5pF. In the 2-step-charging circuit, keeping the length of the switches minimum, the width of the switches were taken based on the logic families. For a single phase and 2-phase designs, the width of the pMOS and nMOS was chosen to be 1u and 0.5u respectively whereas for PFAL and IECRL, the width was taken as 0.25u for all the transistors. In the case of EACRL, due to its dual evaluation network, the pMOS width was taken as 4u and nMOS width was 2u.

**Table 4.3:** Energy dissipation per computation by an adiabatic system (including PCG) and the non-adiabatic design.

| Logic Design Styles | $E_{PCG}$(pJ) | $E_{TOTAL\ SYSTEM}$(pJ) |
|---|---|---|
| Non-Adiabatic | -- | 54.93 |
| CPAL | 101.03 | 107.27 |
| CAL | 113.55 | 134.82 |
| PFAL | 44.17 | 48.53 |
| EACRL | 48.39 | 59.74 |
| IECRL | 29.36 | 36.93 |

Table 4.3 reports the energy consumed by the adiabatic system (including PCG) and the non-adiabatic design for computing the CRC value. In comparison to the non-adiabatic design, only PFAL and IECRL show a decrease in energy dissipation. It is also worth mentioning that the energy consumption of the signal generator for SWC has not been considered. In addition, the energy dissipation of the adiabatic system can be made lower by using step charging circuits with more than 2-steps [49]. It has not been possible to include the clock distribution overhead for non-adiabatic as the figures can be misleading and not reflect reality. Moreover, it will very much be design and layout specific, dependent, how well they are optimised and the tools used. However, the comparison between adiabatic and non-adiabatic in Table 4.3 reported an unfavorable outcome for the adiabatic circuit since the dissipation of the clock generator and

distribution network present in almost all the non-adiabatic circuits are not considered, it is clear from Table 4.3 that IECRL is superior to non-adiabatic regardless of the clock distribution network.

Based on the simulation results for a 16-bit message word-length for 16-bit CRC, the performance trade-offs of the multi-phase adiabatic logic design is tabulated in Table 4.4. The only difference in the structure of PFAL and IECRL logic is in the connection of the evaluation network. They both have the same and a minimum number of transistor counts. On the other hand, the CPAL logic design uses approximately 40% more transistors compared to PFAL and IECRL whereas, CAL and EACRL design consume 25% and 20% more transistors respectively. This increase of CPAL transistor counts is because of the twice the number of buffers needed in the register unit due to the synchronization issue.

**Table 4.4:** Performance trade-offs between multi-phase adiabatic 16-bit CRC implementation for a 16-bit message word-length.

| **Adiabatic Logic Families** | **Area (in terms of transistor counts)** | **Robustness against PVT Variations** | **Computation Time (power-clock cycles)** | **Circuit Complexity** | **Power-clock Complexity** | **$E_{TOTAL\ SYSTEM}$ (pJ)** |
|---|---|---|---|---|---|---|
| CPAL | 3012 | Medium | 75 | High | Medium | 107.27 |
| CAL | 2696 | Low | 37.50 | Medium | Low | 134.82 |
| PFAL | 2150 | High | 18.75 | Low | High | 48.53 |
| EACRL | 2582 | High | 18.75 | High | High | 59.74 |
| IECRL | 2150 | High | 18.75 | Low | High | 36.93 |

The impact of increased message word-length is more on the computation time (throughput) of single-phase and 2-phase designs rather than the 4-phase design. The area is mostly incurred by the register unit rather than the other CRC components, therefore, the impact of increased message word-length is not much on the area of the CRC design for all the five adiabatic logic designs. Since the CRC datapath implementation requires four cascade logic for a single bit CRC bit-slice, the advantage of single phase (CAL) and 2-phase (CPAL) designs in terms of transistor count and throughput diminishes. It can be seen that the 4-phase schemes are more efficient in terms of area and throughput. They also show high robustness against PVT variations.

The power-clock complexity depends on the number of SWC circuits needed to generate the required power-clock phase and the area utilized by the controller circuitry. A single-phase PCG requires one SWC circuits and two flip-flops and two 2-inputs logic gates for the controller. Whereas the 2-phase power-clock generator requires two SWC circuits, three flip-flops, and nineteen 2-input logic gates. On the other hand, all 4-phase power-clock generator is designed using four SWC circuits, two flip-flops, and eight 2-inputs and single-input logic gates.

## 4.7 Summary

This chapter presents the exhaustive survey of single-phase, two-phase and four-phase adiabatic logic families based on the 16-bit adiabatic implementation of CRC for NFC application. A methodology for selecting generically "efficient" design is based on achieving optimum trade-offs between energy, area, computation time, robustness against PVT variations, supply voltage scaling, power-clocking scheme, and power-clock generator complexity.

The 4-phase adiabatic logic designs outperform the single-phase and 2-phase adiabatic logic designs. The CAL complexity is lowest due to the use of single-phase power-clocking scheme, however, its performance is worst based on computation time, throughput, latency, robustness, and energy dissipation. Even though the three 4-phase adiabatic logic designs have high complexity due to the 4-phase power-clock requirement, they show high robustness against PVT variations and energy efficiency compared to the single-phase and 2-phase designs. The 4-phase EACRL has the highest area and energy due to the complex evaluation network compared to IECRL and PFAL. On the other hand, IECRL dissipates more energy at higher capacitance load and less energy at lower capacitance load when compared to PFAL.

Energy saving deteriorates when PCG is considered. The results show that only IECRL consume less energy compared to the non-adiabatic design (without considering the energy dissipation of the clock drivers, clock distribution network and clock generator). The system energy comparison in Table 4.3 and performance comparison between adiabatic logic techniques in Table 4.4 will enable the designers to use quantitative information in selecting the required n-phase adiabatic logic to design an effective feedback system.

# 5 VHDL Modelling for Timing Characterization

Functionality obtained in previous chapters is based on SPICE simulations of transistor-level circuits using Spectre simulator in Cadence. From these, certain inferences were drawn about the synchronization of power-clock phases for correct operation and the time spent in debugging errors in a large adiabatic system. Therefore, if the dual-rail 4-phase adiabatic logic can be modeled using Hardware Description Languages (HDL), time for design, functional verification and error debugging can be significantly reduced. This is the main motivation of this chapter. In this chapter, (VHSIC Hardware Description Language) VHDL based models for the simulations of the dual-rail4-phase adiabatic logic technique are presented. The functional aspects of the models are verified for the 4-phase adiabatic circuit designs used in Chapters 3 and 4. Moreover, the models are designed such that, precise timings of the computation in the 4-phase adiabatic system can be determined. This feature is included as a secondary objective of the VHDL modelling.

## 5.1 Introduction

The verification of the functionality and the low energy traits of adiabatic logic in comparison to non-adiabatic logic is generally performed using the SPICE simulations at the transistor level. But as the size and complexity of the adiabatic system increases, the amount of time required in designing and validating the design increases. Additionally, due to the complexity of synchronizing the power-clock phases, debugging of errors becomes difficult and time-consuming. This gives rise to a need for specific modelling approach that can be used to describe the adiabatic logic behaviour at a higher level of abstraction before the simulations at the transistor level are performed for energy measurements. Such a model would allow functional errors to be detected

and corrected, decreasing the overall time in designing and verifying the functionality of complex adiabatic systems. Moreover, as both the adiabatic and non-adiabatic uses the same process technology and can be fabricated on the same wafer, the precise modelling of both the logic with proper interfacing [100] can save a considerable amount of time arising due to the synchronization complexity. The designing of adiabatic circuits requires much more efforts in contrast to the non-adiabatic logic for which well-developed tools exist. The major difference between the two is that the adiabatic logic designs use slowly changing ac power-clock supply instead of dc power-supply.

The use of adiabatic logic techniques instead of non-adiabatic logic (conventional CMOS) design can decrease the energy consumption of a large system considerably [54]. Based on the performance comparison results of adiabatic logic techniques presented in Chapter 4, the VHDL modelling for the 4-phase adiabatic logic has been done. VHDL is valid and is efficiently used for signal levels '0' and '1' having zero rises and fall time ideally. However, in adiabatic logic, due to the dual-rail encoding of inputs and outputs and the multi-phase clocking scheme, the waveforms are more complex. In addition to the logic '1' and logic '0', the adiabatic power-clock supply uses two transition levels where the power-clock is a ramp. The transition from logic '0' to logic '1' known as charging/evaluation period and transition from logic '1' to logic '0' known as discharging/recovery period. The 4-phase power-clock is modeled such that all the four periods share the same time. The more detailed description of the power-clocking scheme is given in Chapter 2 of this thesis.

In the literature, few research papers exist that details the modelling of adiabatic logic using HDL. Not much attention has been given to the higher-level simulation of the adiabatic logic designs due to the complex power-clock generation requirements. According to the authors best knowledge and literature review, the first modelling of adiabatic logic was done by M. Vollmer and J. Gotze in 2005 [26]. They described a systolic array of CORDIC devices using adiabatic logic modeled in VHDL. Their work included the description of the adiabatic logic block but did not model the dual-rail behaviour and used one global clock net instead of 4-phase power-clock for cascade designs. A year later, Laszlo Varga et.al. [27] described two-level pipelining scheduling of adiabatic logic using integer linear programming formulation and a heuristic scheduling. The authors presented the VHDL description for functional simulation of the synthesized adiabatic datapath together with the non-adiabatic part of the digital

system. This approach focused mainly on producing a pipeline schedule of the power-clock behaviour of the adiabatic logic but did not model the power-clock and used the single-rail encoding of the adiabatic logic. In 2010, David John Willingham in his Ph.D. thesis [28] reported Asynchrobatic Logic in Verilog, an industry standard HDL. First, the author demonstrated the idea in a single-rail scheme and then extended it to dual-rail, which was found to be missing in Vollmer and Laszlo's modelling. Though the dual-rail implementation proves to be advantageous in detecting invalid circuit operations, the author did not model the power-clock in HDL, instead uses square waveform changing from logic '1' to '0' and vice versa.

The main drawback of all the existing approaches is that none have shown the actual representation of an adiabatic logic technique [24] by representing all the four periods namely; evaluation, hold, recovery and idle of the power-clock in HDL. Instead, the power-clock is represented as a square waveform with only two logic levels (logic '1' and logic '0') like that of the non-adiabatic logic. Here, the logic '1' corresponds to the hold period and logic '0' corresponds to the idle period. The remaining two periods, which are ramp; one changing from 0 to $V_{DD}$ corresponds to the evaluation and other from $V_{DD}$ to 0 corresponding to the recovery period, have been skipped or merged with hold and idle periods. Ideally, all the four periods should share the same time, for correctly representing the adiabatic waveform/power-clock and follow the adiabatic principles. Thus, unlike SPICE-level simulation, when the inputs and power-clock are in the same phase then the output will still be valid using the existing modelling approaches. This invalid operation will lead to a wrong functionality and will be difficult and time-consuming to be detected errors in a large circuit. The error in the encoding of HDL using the existing approach is given in the next section.

Therefore, in this study, VHDL-based modelling for the 4-phase adiabatic logic technique is developed for functional simulation. It represents the 4-phase power-clocking scheme and includes a systematic approach for precise timing analysis. This is the novel contribution which captures the exact timing behaviour and detects the circuit's invalid operation by checking the generated complementary outputs. The modelling includes the dual-rail representation of the input/output signals. The four periods of the power-clock explicitly defined as a function in VHDL. The conceptual block diagram for an adiabatic NOT/BUF gate is given in Figure 5.1. The power-clock generator block comprises of two flip-flops working as a 2-bit counter. The input to the

power-clock generator is the clock signal (CLK). The four states of the counter are assigned to the four periods of the power-clock. The adiabatic conversion block comprises of a 2-bit counter and depending on the pulse input levels (IN_H, IN_L) the adiabatic outputs (A_H, A_L) are generated. The adiabatic core is a NOT/BUF gate generating both the complementary outputs (Out_H, Out_L).

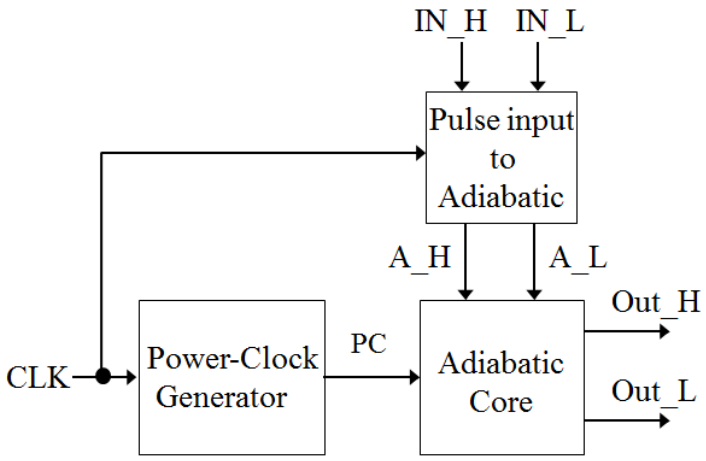



**Figure 5.1:** A conceptual block diagram of an adiabatic system where the adiabatic core is a NOT/BUF gate.

## 5.2 Encoding of HDL Models

The most difficult part of modelling the adiabatic logic using conventional HDLs is that these languages are made entirely for encoding two logic levels (‘0’ and ‘1’) and is either ‘level’ or ‘edge’ sensitive. In order to define a cell library with HDL functional models of the adiabatic cells which can be used with conventional HDL simulators in the design of a large adiabatic circuit, power-clock of the adiabatic logic needs to be suitably encoded for all the four periods (Figure 2.2 in Chapter 2 of this thesis). In the literature, the voltage-level encoding style for adiabatic logic has been used which is similar to the non-adiabatic logic designs. With the adiabatic logic, having trapezoidal power-supply, the gates operating during a specific period must follow the adiabatic principle. In addition, each adiabatic logic gates such as MUX, AND/NAND, OR/NOR and XOR/XNOR must be sequential that combines the logic functionality with storage capability, therefore, requires a different encoding approach. In this work, two encoding styles are discussed i.e. previously used voltage-level encoding style and the proposed multi-level event-based encoding style.

### *5.2.1 Voltage-level Event-based Approach*

So far voltage-level event-based approach is commonly used for encoding the behaviour of the power-clock in adiabatic logic. It uses two logic levels to represent the four periods of the adiabatic power-clock. The logic '1' of the power-clock signal represents Evaluation (E) and Hold (H) period of the trapezoidal power-clock, whereas, the logic '0' represents the Recovery (R) and Idle (I) period of the trapezoidal waveform. Figure 5.2 shows the voltage-level encoding style. The adiabatic logic modeled using this style uses the clock signal as the power-clock.

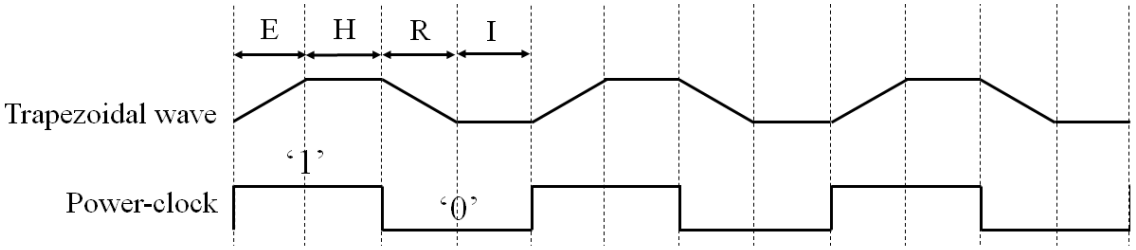



**Figure 5.2:** Voltage level event-based encoding.

Although the author [36] included the checking of the invalid states on the positive-edge of the power-clock, the drawback of this style still exists. Here the output does not follow the power-clock; rather it is a function of the input being processed (which is not the case with adiabatic circuits). For example, in an adiabatic buffer gate, when the inputs are valid (logic '1') during the positive-edge of the clock, the complementary output nodes follow the complementary inputs. However, in the case of SPICE simulation, one of the outputs follows the power-clock depending on the input being processed whereas its complementary node is discharged to ground. This difference has been removed in the proposed encoding scheme by making sure that the outputs follow the power-clock, not the inputs. Moreover, as stated above, voltage-level encoding for adiabatic power-clock doesn't follow the adiabatic principle for cascade logic. Thus, the adiabatic logic design can malfunction if either the PC is delayed or the input arrives early such that the power-clock rising edge aligns with the falling edge of the input.

### *5.2.2 Multi-level Event-based Approach*

In the proposed approach, the hold and the idle periods of the power-clock are represented as logic '1' and logic '0' similar to that of the voltage-level encoding.

Whereas, the evaluation and the recovery period are encoded with an intermediate state marked as 'X', for the duration of the ramp period. This approach is not straightforward, as apart from generating the power-clock which has three logic levels, the adiabatic inputs must also be generated with three logic levels for proper functionality and timing analysis. The trapezoidal power-clock is modeled as three logic levels as shown in Figure 5.3. The four periods of the power-clock are defined as an edge function in VHDL which is aggregated into a package named "Adiabatic_signal". The package is shared between different VHDL models to develop the cell library of the basic adiabatic logic gates. The approach can also be easily used for single-phase and 2-phase adiabatic logic techniques, although it will not be straightforward in the case of 2-phase adiabatic logic due to its long idle period.

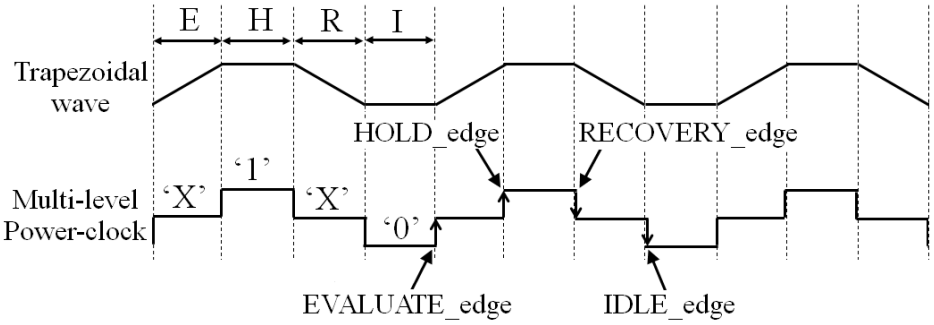



**Figure 5.3:** Multi-level event-based encoding.

## 5.3 HDL Modelling of 4-phase Adiabatic Logic Technique

VHDL is used to model the 4-phase adiabatic logic technique to capture the circuit description. One of the advantages of modelling is that the design can be simulated with logic simulators and can be interfaced with the non-adiabatic logic designs for energy efficiency. Generally, the circuit behaviour and the timing extracted from SPICE simulation are used to develop the VHDL models. First, the trapezoidal power-clock used at the transistor level is encoded as a multi-level in standard logic (shown in Figure 5.4) to capture the behaviour of adiabatic logic. This is followed by the gate-level modelling and interconnection modelling (pulse input to adiabatic conversion). Other than simulating the circuit behaviour in HDL, the main objective is to measure the computation time of the circuit so that for a large system, the throughput can easily be calculated. Moreover, like SPICE simulation, the proposed modelling approach for adiabatic logic detects invalid complementary inputs by checking on to the

complementary outputs. Also, the conventional CMOS circuits required for the power-clock and adiabatic input generation are similar to that of the controller used to generate signals for PCG [101].

The method used is presented as follows.

1) Modelling the behaviour of the trapezoidal AC power-clock.

2) Generation of dual-rail adiabatic signals from dual-rail pulse input.

3) Developing a VHDL model library.

4) Modelling invalid inputs.

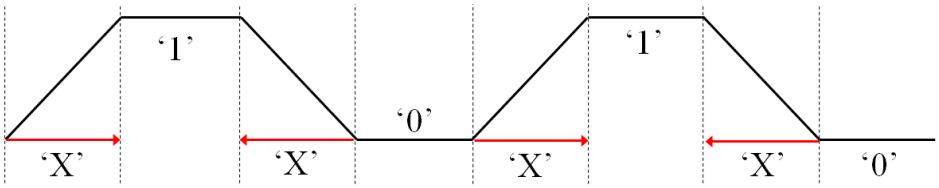



**Figure 5.4:** Encoding trapezoidal waveform in standard logic.

### *5.3.1 Modelling Trapezoidal AC Power-clock*

To realize the adiabatic power-clock in standard logic using a multi-level approach, as depicted in Figure 5.3, four states are required. Each state is encoded based on the voltage level needed. The four states can be easily generated using two flip-flops counting from "00" to "11".For simplicity, the counter output is forced externally as the input to the power-clock generator block using the clock signal 'CLK' as a two-bit number, generating four states. Figure 5.5 shows the circuit simulation of the power-clock for a time period of 200ns. The VHDL code for power-clock generation is shown below in Listing 5.1:

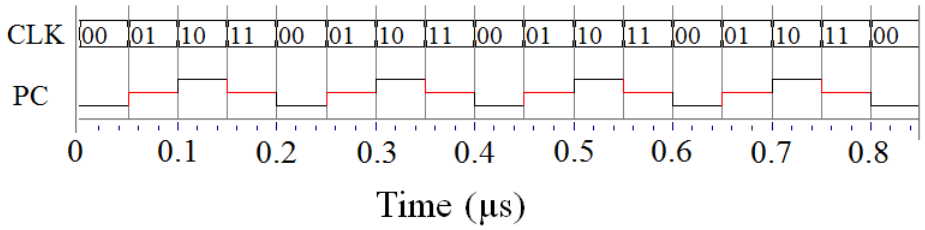



**Figure 5.5:** HDL simulation for generating a power-clock signal.

```
1. LIBRARY IEEE;
2. USE IEEE.STD_LOGIC_1164.ALL;
3. USE IEEE.STD_LOGIC_ARITH.ALL;

4. ENTITY GENERATE_ADIABATIC_CLOCK IS
5. port (CLK: in std_logic_vector (1 downto 0); PC : out
   std_logic);
6. END ENTITY GENERATE_ADIABATIC_CLOCK;

7. Architecture Behavioural of GENERATE_ADIABATIC_CLOCK IS

8. Begin
9. Process (CLK) IS
10. Begin
11.   if CLK ="00" then           // IDLE PERIOD //
12.    PC<='0';
13.   elsif CLK = "01" then       // EVALUATE PERIOD //
14.    PC<='X';
15.   elsif CLK = "10" then       // HOLD PERIOD //
16.    PC<='1';
17.   elsif CLK = "11" then       // RECOVERY PERIOD //
18.    PC<='X';
19.   End if;
20.End Process;
21.End Architecture Behavioural;
```

**Listing 5.1:** VHDL code for a power-clock generation.

### *5.3.2 Generating Dual-rail Adiabatic Signals from Dual-rail Pulse Input.*

One of the key requirements for adiabatic logic to perform correctly is the generation of the adiabatic inputs using a multi-level encoding approach. In adiabatic logic, the input must be stable (hold period) during the evaluation period of the power-clock. This behaviour is captured in the proposed modelling by having the input to arrive one phase before the power-clock such that more realistic modelling representing the adiabatic logic is realized. Similar to the power-clock generator block, the pulse input to adiabatic conversion block consists of the two-bit clock signal 'CLK' forced externally to generate four states. Depending on the dual-rail signals it's equivalent dual-rail adiabatic outputs are generated. Figure 5.6 shows the VHDL simulations for the generation of adiabatic input signals. The adiabatic inputs can be generated for any power-clock phase required by simply assigning the states to the 'CLK' signal. The VHDL code for the same is shown in Listing 5.2. It also shows the modelling of invalid pulse inputs to its equivalent adiabatic outputs.

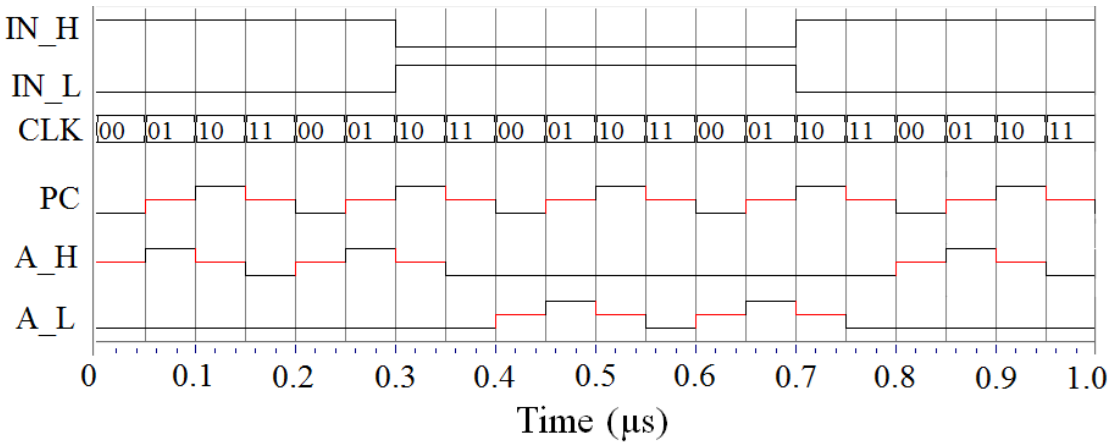



**Figure 5.6:** VHDL simulation for generating the adiabatic input signals.

```
1.LIBRARY IEEE;
2.USE IEEE.STD_LOGIC_1164.ALL;
3.USE IEEE.STD_LOGIC_ARITH.ALL;

4.ENTITY PULSE_INP_to_ADIABATIC IS
5.port (IN_H, IN_L : in std_logic; CLK: in std_logic_vector(1
  downto 0);A_H, A_L : out std_logic);
6.END ENTITY DC_TO_ADIABATIC;

7.Architecture Behavioural of PULSE_INP_to_ADIABATIC is
8.Begin
9.  Process (CLK) is
10.       Begin

11.  if CLK ="00" then
12.     if IN_H ='1' and IN_L ='0' then
13.         A_H<='X';
14.         A_L<='0';
15.     elsif IN_L ='1' and IN_H ='0' then
16.         A_H<='0';
17.         A_L<='X';
18.     elsif IN_H ='1' and IN_L ='1' then  //Invalid State//
19.         A_H<='X';
20.         A_L<='X';
21.     elsif IN_H ='0' and IN_L ='0' then  //Invalid State//
22.         A_H<='0';
23.         A_L<='0';
24.     End if;

25.  elsif CLK = "01" then
26.     if IN_H ='1' and IN_L ='0' then
27.      A_H<='1';
28.      A_L<='0';
29.   elsif IN_L ='1' and IN_H='0' then
30.         A_H<='0';
31.         A_L<='1';
```

```
32.   elsif IN_H ='1' and IN_L ='1' then      //Invalid State//
33.         A_H<='1';
34.         A_L<='1';
35.    elsif IN_H ='0' and IN_L ='0' then    //Invalid State//
36.      A_H<='0';
37.      A_L<='0';
38.   End if;

39.   elsif CLK = "10" then
40.    if IN_H ='1' and IN_L ='0' then
41.      A_H<='X';
42.      A_L<='0';
43.   elsif IN_L ='1' and INP_H='0' then
44.      A_H<='0';
45.      A_L<='X';
46.   elsif IN_H ='1' and IN_L ='1' then    //Invalid State//
47.      A_H<='X';
48.      A_L<='X';
49.   elsif IN_H ='0' and IN_L ='0' then    //Invalid State//
50.      A_H<='0';
51.      A_L<='0';
52.   End if;

53.   elsif CLK = "11" then
54.   A<='0';
55.   Ab<='0';
56.   End if;
57.   End Process;
58.   End Architecture behavioural;
```

**Listing 5.2:** VHDL code for converting DC inputs to adiabatic inputs.

### *5.3.3 Developing a VHDL Model Library*

To model the adiabatic logic gates, VHDL primitives are compared to equivalent adiabatic gates based on the multi-level encoding approach. Table 5.1 shows the truth table of the two basic primitives AND and OR. The not gate is not shown as behavioural modelling of the NOT/BUF adiabatic gate has been done. In Table 5.1 and 5.2, the outputs in red indicate the one that is not matched with the adiabatic logic modelling. The proposed modelling uses ‘x’ and ‘z’ as intermediate and invalid states respectively. Thus, the operation involving either of them with ‘1’ and ‘z’ produces an invalid output ‘z’, marked in red for the adiabatic logic modelling. In addition, the OR operation of the adiabatic logic modelling involving ‘z’ with ‘0’ produces an invalid output marked with ‘z’ in Table 5.2. Table 5.2 is used to write a user-defined primitive for AND and OR as a function in VHDL. The functions utilize case statement control structure and are named ‘Aand’ and ‘Aor’ in Adiabatic_2INP_GATES package body.

Their VHDL codes are shown in Listings 5.3. The lines 12-21 represent the Aor and lines 27-38 represents the Aand.

**Table 5.1:** Basic logic gates AND and OR.

| AND | 0 | 1 | x | z |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | x | x |
| x | 0 | x | x | x |
| z | 0 | x | x | x |

| OR | 0 | 1 | x | z |
|---|---|---|---|---|
| 0 | 0 | 1 | x | x |
| 1 | 1 | 1 | 1 | 1 |
| x | x | 1 | x | x |
| z | x | 1 | x | x |

**Table 5.2:** Basic logic gates AND and OR for adiabatic logic modelling.

| Aand | 0 | 1 | x | z |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | z |
| 1 | 0 | 1 | z | z |
| x | 0 | z | x | z |
| z | z | z | z | z |

| Aor | 0 | 1 | x | z |
|---|---|---|---|---|
| 0 | 0 | 1 | x | z |
| 1 | 1 | 1 | z | z |
| x | x | z | x | z |
| z | z | z | z | z |

```
1.LIBRARY IEEE;
2.USE IEEE.STD_LOGIC_1164.ALL;
3.PACKAGE Adiabatic_2INP_GATES IS
4.FUNCTION Aor  (L, R : std_ulogic) RETURN UX01Z;
5.FUNCTION Aand (L, R: std_ulogic) RETURN UX01Z;
6.END Adiabatic_2INP_GATES;

7.PACKAGE BODY Adiabatic_2INP_GATES IS

                //User Defined Adiabatic OR//

8. FUNCTION Aor (L, R : std_ulogic) RETURN UX01Z is
9.    VARIABLE sel: std_logic_vector (1 downto 0);
10.      begin
11.   sel:= L&R;
12.      case sel is
13.        when "00"=> return '0';
14.        when "01"=> return '1';
15.        when "10"=> return '1';
16.        when "11"=> return '1';
17.        when "0X"=> return 'X';
18.        when "X0"=> return 'X';
19.        when "XX"=> return 'X';
20.        when others=> return 'Z';
21.    end case;
22. END FUNCTION;

                // User Defined Adiabatic AND//

23. FUNCTION AandL, R : std_ulogic) RETURN UX01Z is
24.         VARIABLE sel: std_logic_vector (1 downto 0);
25.      begin
26.        sel:= L&R;
27.        case sel is
28.     when "0Z"=> return '0';
29.     when "Z0"=> return '0';
30.     when "00"=> return '0';
31.     when "10"=> return '0';
32.     when "01"=> return '0';
33.     when "0X"=> return '0';
34.     when "X0"=> return '0';
35.     when "XX"=> return 'X';
36.     when "11"=> return '1';
37.     when others=> return 'Z';
38. end case;
39.   END FUNCTION;
40.End PACKAGE BODY Adiabatic_2INP_GATES;
```

**Listing 5.3:** User-defined primitives as functions in a user-defined package

In addition, the logic level simulation and timing verification of the adiabatic logic circuits with standard tools is possible only after defining the functions for power-clock periods namely; EVALUATE_edge, HOLD_edge, RECOVERY_edge and the

IDLE_edge in the package that is used by all the adiabatic HDL description files by placing the 'USE' directive in the program. The VHDL package defining the power-clock periods are shown in Listing 5.4. The function for EVALUATE_edge is represented by the lines 10-13, HOLD_edge by the lines 14-17, RECOVERY_edge by the lines 18-21 and lines 22-25 represent the IDLE_edge of the power-clock.

```
1. LIBRARY IEEE;
2. USE IEEE.STD_LOGIC_1164.ALL;
3. PACKAGE Adiabatic_signal IS
4. FUNCTION EVALUATE_edge(SIGNAL s :std_ulogic)
   RETURNBOOLEAN;
5. FUNCTION HOLD_edge(SIGNAL s : std_ulogic) RETURN BOOLEAN;
6. FUNCTION RECOVERY_edge(SIGNAL s : std_ulogic) RETURN
   BOOLEAN;
7. FUNCTION IDLE_edge(SIGNAL s : std_ulogic) RETURN BOOLEAN;
8. END Adiabatic_signal;

9. PACKAGE BODY Adiabatic_signal IS

10. FUNCTION EVALUATE_edge(SIGNAL s : std_ulogic) RETURN
    BOOLEAN is
11. Begin
12. RETURN (s'EVENT AND(To_X01(s) = 'X') AND
    (To_X01(s'LAST_VALUE)= '0'));
13. END FUNCTION;

14. FUNCTION HOLD_edge(SIGNAL s :std_ulogic) RETURN BOOLEAN is
15. Begin
16. RETURN (s'EVENT AND (To_X01(s) = '1') AND
    (To_X01(s'LAST_VALUE)= 'X'));
17. END FUNCTION;

18. FUNCTION RECOVERY_edge(SIGNAL s : std_ulogic) RETURN
    BOOLEAN is
19. Begin
20. RETURN (s'EVENT AND (To_X01(s) = 'X') AND
    (To_X01(s'LAST_VALUE)= '1'));
21. END FUNCTION;

22. FUNCTION IDLE_edge(SIGNAL s :std_ulogic) RETURN BOOLEAN is
23. Begin
24. RETURN (s'EVENT AND (To_X01(s) = '0') AND
    (To_X01(s'LAST_VALUE)= 'X'));
25. END FUNCTION;
26. End PACKAGE BODY Adiabatic_signal;
```

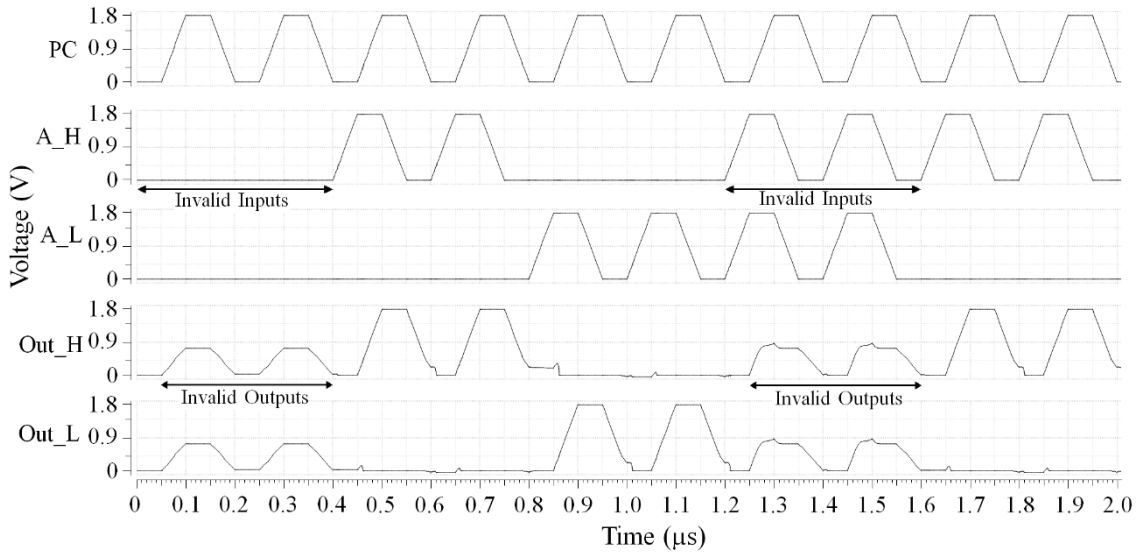
**Listing 5.4:** User-defined four power-clock periods as functions in a package.

Then the development of VHDL model for the NOT/BUF adiabatic gate is collectively done using the power-clock generation, package defining the four periods of the adiabatic signal and pulse input to adiabatic inputs conversion. The VHDL code for the NOT/BUF adiabatic gate is given in Listing 5.5. The package is defined in line 4. Line 6 defines the input/output ports of the gate and line 15 describes the signals similar to the wires in a schematic used to interconnect the components. Line 17-18 defines the component instantiation for the adiabatic power-clock and adiabatic input generation. The behaviour of the adiabatic NOT/BUF gate is captured in lines 19-73. Apart from checking the invalid input condition in each of the four periods, an invalid state is also checked for the NOT/BUF gate for cascade designs in lines 69-71. The four periods of the power- clock in Listing 5.5 are defined as a level sensitive signals due to the use in cascade logic designs, otherwise, the stages ahead of the first will be stuck at logic '0'. The output waveform using the SPICE simulation and the proposed modelling is shown in Figure 5.7 (a) and (b) respectively. The VHDL simulation shows the precise timing similar to the SPICE simulation.

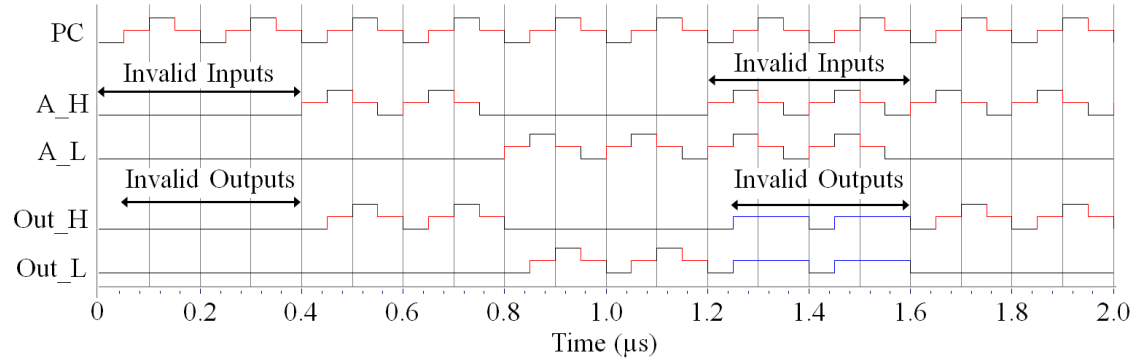
```
1.  LIBRARY IEEE;
2.  USE IEEE.STD_LOGIC_1164.ALL;
3.  USE IEEE.STD_LOGIC_ARITH.ALL;
4.  USE work.Adiabatic_signal.all;    //Package Definition//

5.  ENTITY Proposed_Buf IS
6.  port (IN_H, IN_L: in std_logic; CLK: in std_logic_vector(1
    downto 0); Out_H, Out_L: out std_logic);
7.  END ENTITY Proposed_Buf;

8.  Architecture Behavioural of Proposed_Buf IS
9.   Component DC_TO_ADIABATIC
10.  port (IN_H, IN_L : in std_logic; CLK : in std_logic_vector(1
     downto 0);  A_H, A_L : out std_logic);
11.  End Component;

12.  Component GENERATE_ADIABATIC_CLOCK
13.  port(CLK: in std_logic_vector (1 downto 0); PC : out
     std_logic);
14.  End Component;

15. Signal A_H, A_L, PC :std_logic;
16. Begin
17.  INPUT1: DC_TO_ADIABATIC port map(IN_H,IN_L,CLK,A_H,A_L);
18.  CLK1: GENERATE_ADIABATIC_CLOCK  port map(CLK, PC);
19. Process (PC, A_H, A_L) is
20.  Begin
                    // IDLE PERIOD //
21.   if PC='0'  then
22.     Out_H <=PC;
23.     Out_L<= PC;
```

```
                          // EVALUATION PERIOD //
24.   elsif PC='X' and HOLD_edge (A_H) and HOLD_edge (A_L)
      then//Invalid State//
25.     Out_H<='Z';
26.     Out_L<='Z';
27.   elsif PC='X' and HOLD_edge (A_H) then
28.     Out_H <= PC;
29.     Out_L<='0';
30.   elsif PC='X' and HOLD_edge (A_L) then
31.     Out_H <= '0';
32.     Out_L<=PC;
33.   elsif PC='X' and RECOVERY_edge (A_H) then
34.     Out_H<='Z';
35.     Out_L<='Z';
36.   elsif PC='X' and RECOVERY_edge (A_L) then
37.     Out_H<='Z';
38.     Out_L<='Z';
                          // HOLD PERIOD //
39.   elsif PC='1' and RECOVERY_edge (A_H) and RECOVERY_edge (A_L)
      then //Invalid State//
40.     Out_H<='Z';
41.     Out_L<='Z';
42.   elsif PC='1' and RECOVERY_edge (A_H) then
43.     Out_H<= PC;
44.     Out_L<='0';
45.   elsif PC='1' and RECOVERY_edge (A_L) then
46.    Out_H<= '0';
47.    Out_L<=PC;
48.   elsif PC='1' and IDLE_edge (A_H) then
49.     Out_H<='Z';
50.     Out_L<='Z';
51.   elsif PC='1' and IDLE_edge (A_L) then
52.     Out_H<='Z';
53.     Out_L<='Z';
                          // RECOVERY PERIOD //
54.   elsif PC='X' and IDLE_edge (A_H) and IDLE_edge (A_L)
      then                          //Invalid State//
55.    Out_H<='Z';
56.    Out_L<='Z';
57.   elsif PC='X' and IDLE_edge (A_H) then
58.    Out_H<= PC;
59.    Out_L<='0';
60.   elsif PC='X' and IDLE_edge (A_L) then
61.    Out_H<= '0';
62.    Out_L<=PC;
63.   elsif PC='X' and EVALUATE_edge (A_H) then
64.    Out_H<='Z';
65.    Out_L<='Z';
66.   elsif PC='X' and EVALUATE_edge (A_L) then
67.    Out_H<='Z';
68.    Out_L<='Z';
                           // INVALID STATE //
69.   elsif A_H='Z' and A_L='Z' then
70.      Out_H<='Z';
71.      Out_L<='Z'
72.   End if;
73.  End Process;
74. End Architecture Behavioural;
```

**Listing 5.5:** VHDL code for adiabatic NOT/BUF gate.




(a)




(b)

**Figure 5.7:** Simulation results for NOT/BUF gate (a) SPICE (b) VHDL.

### *5.3.4 Modelling Invalid Complementary Inputs*

The operation of the adiabatic logic gates, although conceptually simple, can be somewhat complex to model accurately. This is due to the two cross-coupled inverters forming a latch, which retains the last value stored at the output. For example: if both the complementary inputs are logic '0' (invalid states), the complementary outputs will retain the last value stored. That is if the last value is logic '1' and logic '0' on the two output nodes 'Out_H' and 'Out_L' respectively then the same value will be retained. This can be seen in Figure 5.8. This, invalid input in a large circuit will be difficult to debug, especially in the case when functionally, logic '1'and '0' is expected on the two output nodes. In addition, this invalid circuit operation will lead to high energy consumption, due to the non-adiabatic losses. On the other hand, if the complementary inputs are invalid by being at logic '1', the complementary output nodes will be charged

through the pMOS transistor (which follows the power-clock)and at the same instant, the nMOS transistor will be discharging the output nodes to ground. Therefore, the output nodes will settle at some intermediate value as can be seen from Figure 5.7 (a) and 5.8.

The proposed HDL model can easily identify both these classes or invalid inputs by ensuring the complementary output nodes to be at high impedance state ‘z’, when invalid inputs are logic ‘1’ consistent with the SPICE simulation Whereas when the invalid inputs are logic ‘0’, the complementary output nodes are also at logic ‘0’. The above invalid operations are shown in Figure 5.7 (b). This helps in identifying the value of the invalid inputs clearly.

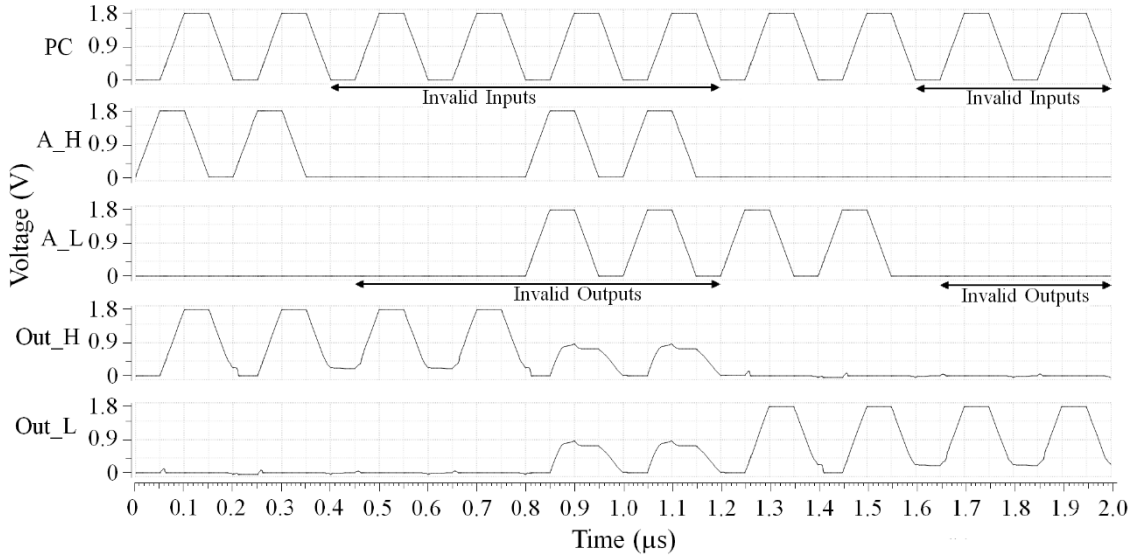



**Figure 5.8:** SPICE simulation for PFAL NOT/BUF gate showing invalid outputs.

For all the other adiabatic logic gates such as AND/NAND, OR/NOR, XOR/XNOR and MUX/DeMUX, the functional behaviour can be described similar to the adiabatic NOT/BUF gate. However, the modelling of the above logic gates is performed by combining the functional part and the adiabatic NOT/BUF. This is because the modelling is done considering the dual-rail inputs thus, the state checking complexity of the HDL behavioural increases for an increased number of inputs due to the complementary inputs. The NOT/BUF gate used helps in following the adiabatic principle and identifying the invalid complementary inputs while synchronizing the outputs for correct timing characterization.

Figure 5.9 shows how the dual-rail 2-input AND/NAND gate, is conceptualized as a logic function combined with a NOT/BUF gate for timing characterization in VHDL. The collection of all the logic gates described in VHDL formed the cell library.

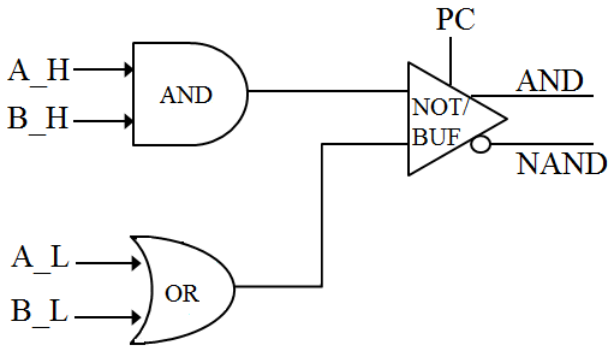
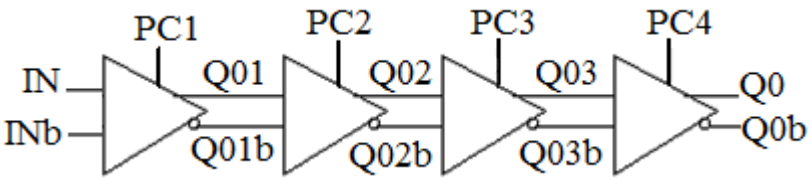



**Figure 5.9:** Conceptualization of 2-input AND/NAND gate for timing characterization.

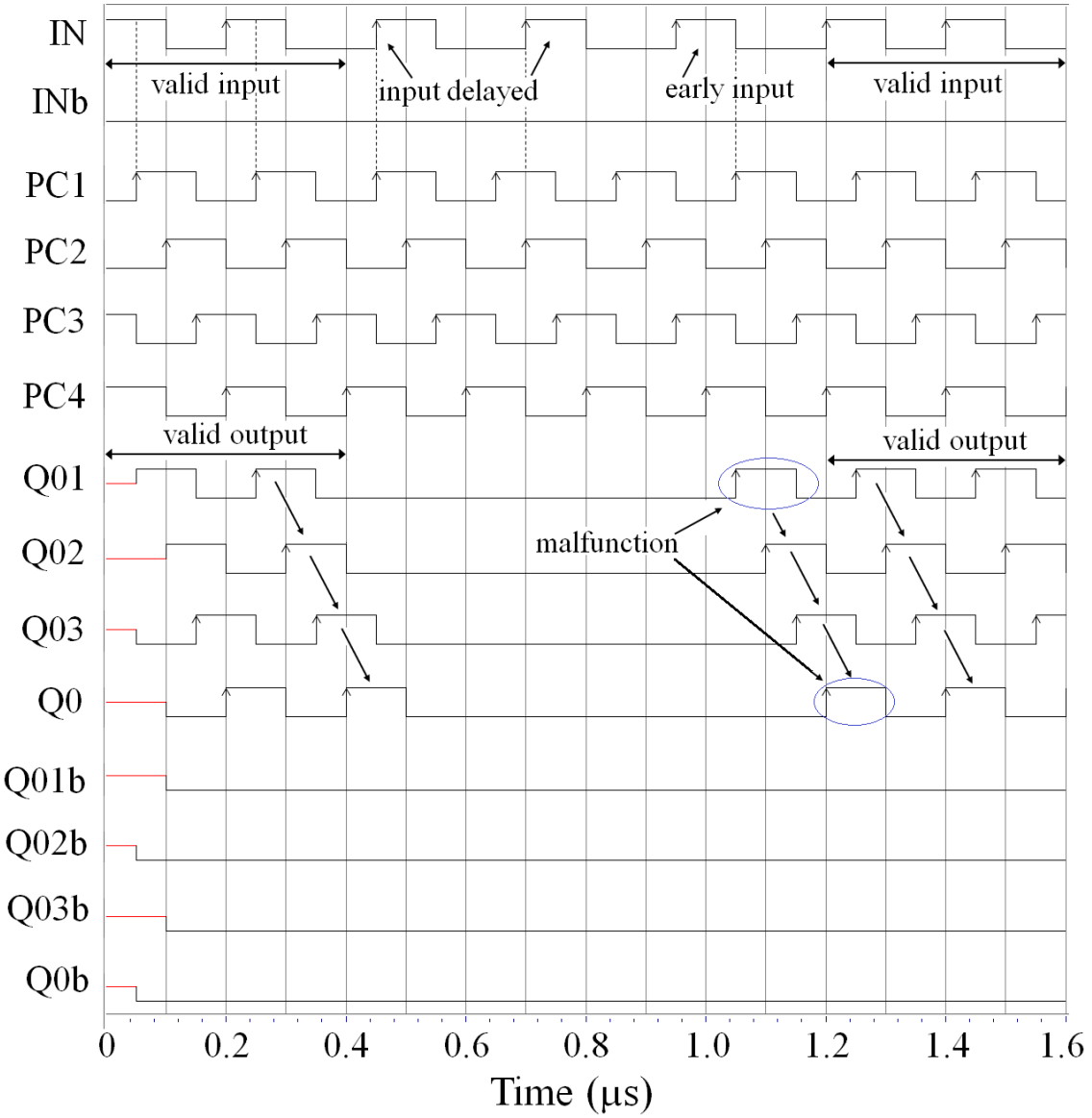
## 5.4 Error in Modelling of Existing Approach

As stated in section 5.2, voltage-level encoding can result in circuit malfunctioning due to the violation of the adiabatic principle. As a result, the circuit will fail to maintain precise timing with that of the simulated waveform at the transistor level. Thus, to calibrate our proposed modelling, in case, if either the input or the power-clock arrives early or gets delayed the two output nodes should discharge to ground, identifying an invalid input has occurred and the modelling follows the adiabatic principle. Figure 10 (a) shows the cascade buffer chain designed using PFAL NOT/BUF gate connected in cascade. The gate working in power-clock phase 1 (PC1) produces the first stage output denoted as ‘Q01_H’ and ‘Q01_L’. The fourth stage works in power-clock phase 4 (PC4) and produces the final stage outputs denoted by ‘Q0_H’ and ‘Q0_L’. Figure 10 (b) shows the condition of an early arrival and delayed input for the existing modelling using square-waveform. It can be seen in Figure 10 (b) that for the delayed input condition, the outputs follow the adiabatic principle by generating logic ‘0’ for the existing approach. However, when the input arrives early, the output follows the power-clock, thus violating the adiabatic principle. Therefore, in the existing approach, a timing window exists between the input and the power-clock for correct circuit and timing operation. The same condition can occur if the power-clock is either delayed or arrives early. In a small circuit such errors can be easily detected manually, but for a large complex circuit, detection of such errors will be time-consuming and very difficult

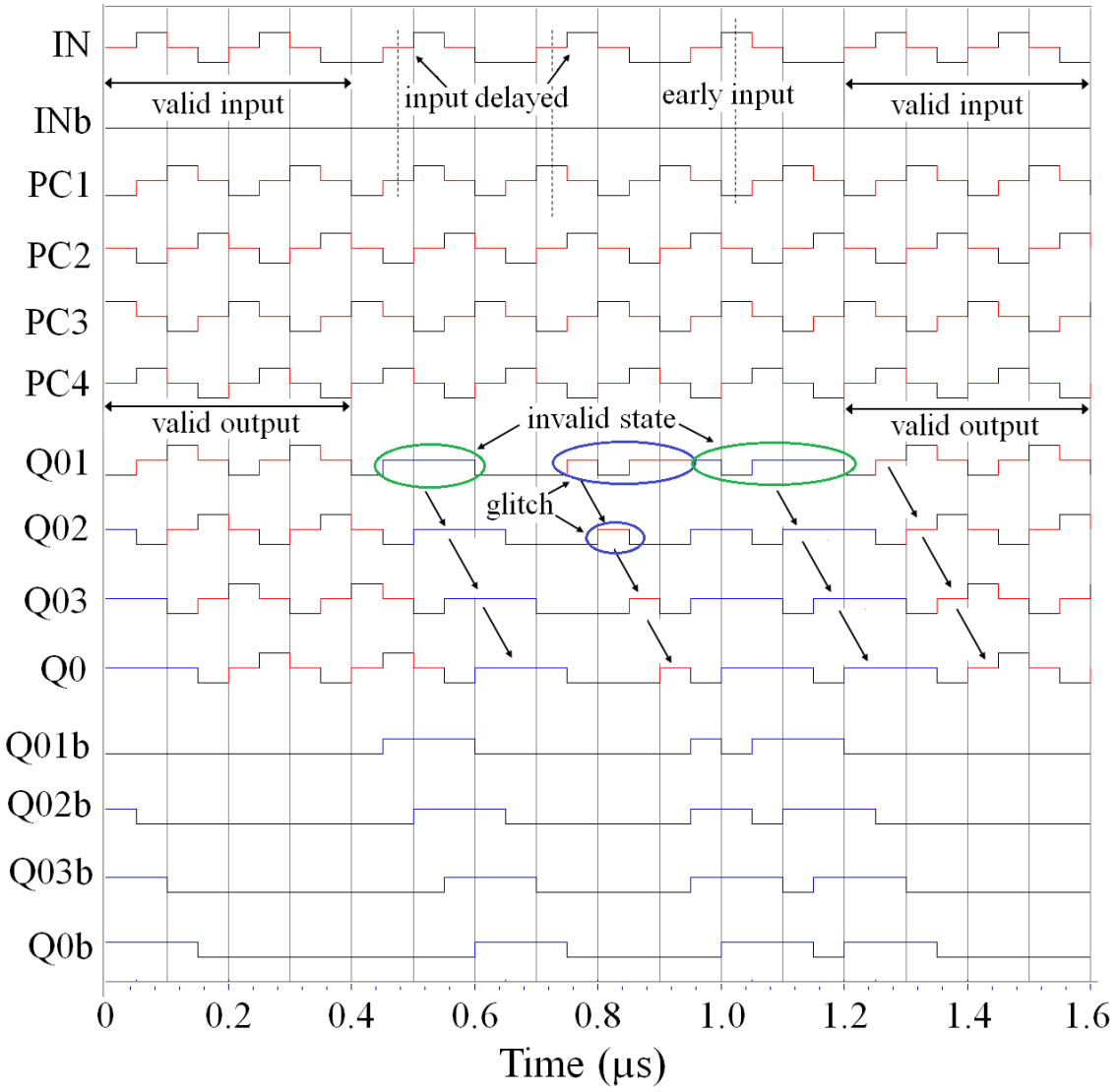
to debug. In addition, the circuit will fail to maintain precise timing that agrees with that of the SPICE simulated waveform.




(a)




(b)




(c)

**Figure 5.10:** Schematic of the cascade buffer chain. (b) Simulated waveforms of input timing variations for the existing approach using square-waveform. (c) Simulated waveform using the proposed approach.

To overcome the drawbacks highlighted earlier on in the paper and identify the invalid inputs correctly, the power-clock of the adiabatic logic is encoded for all the four periods. It can be seen from Figure 10 (c) that the proposed modelling approach will fail if the wrong input signal or the power-clock (delayed or arrived early) is supplied. This gate generation failure will be similar to that of the SPICE simulation.

The proposed modelling approach is much more precise, however, it generates a glitch for the delayed input condition, which reduces as it is passed through a cascade NOT/BUF gates which can be seen in Figure 10 (c). The glitch arises due to the signal

'X' being used for encoding both the ramps (evaluation and recovery period). It can however be removed if two different signals such as 'U' and 'X' are used for encoding the two ramps. However, this glitch is insufficient to cause any functionality and timing error which the existing logic exhibits. The simple 'X' was used for simplicity and it works well with the timing requirement.

## 5.5 Simulation Results

Using the cell library, a 2-bit ring counter and a 3-bit Up-Down counter structural models were successfully verified. The circuit functionality and timing verification are done using HDL Designer from Mentor Graphic. The time period of the power-clock is taken as 100ns, having an equal time period for the four periods of the power-clock i.e. 25ns each. The 4-phase adiabatic logic family used for the SPICE simulation is PFAL. The simulation setup for the SPICE analysis is similar to that of the VHDL so that the uniformity and comparability are maintained across both the simulations. The VHDL codes for the above two designs are included in Appendix B of this thesis, whereas the VHDL simulations alongside with the SPICE level simulations are presented for the ring counter and Up-Down counter.

The structural level of abstraction is used which combines the components to form a large adiabatic system. In order to work in a cascade manner, first, the 4-phase power-clock is generated each having a 90$^{o}$ phase difference between them. Figure 5.11 shows the output waveform of the 4-phase power-clock generation. The VHDL code for the same is given in Appendix B.

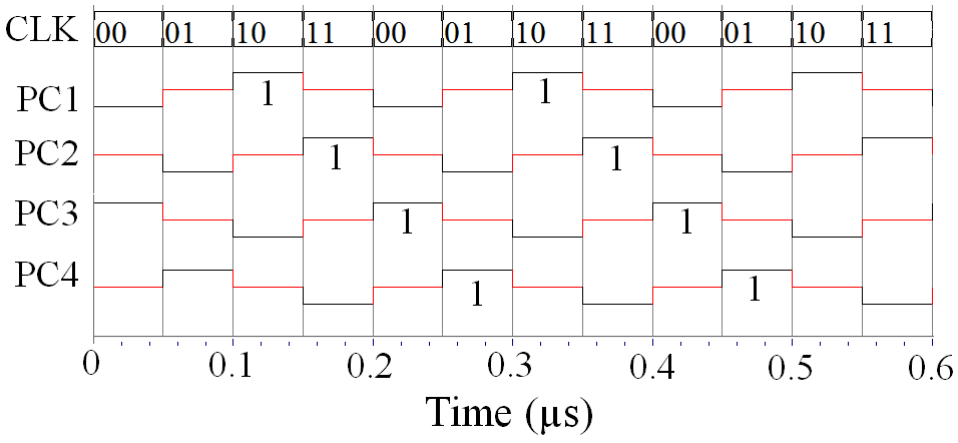
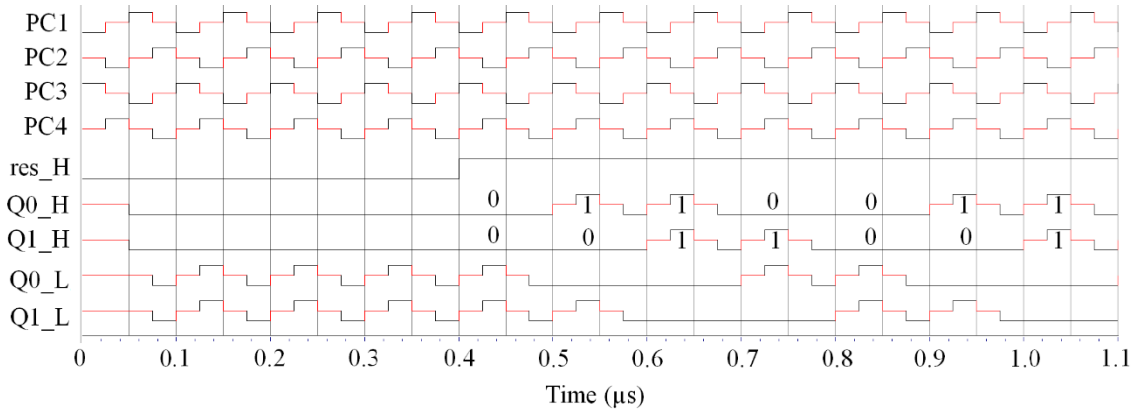


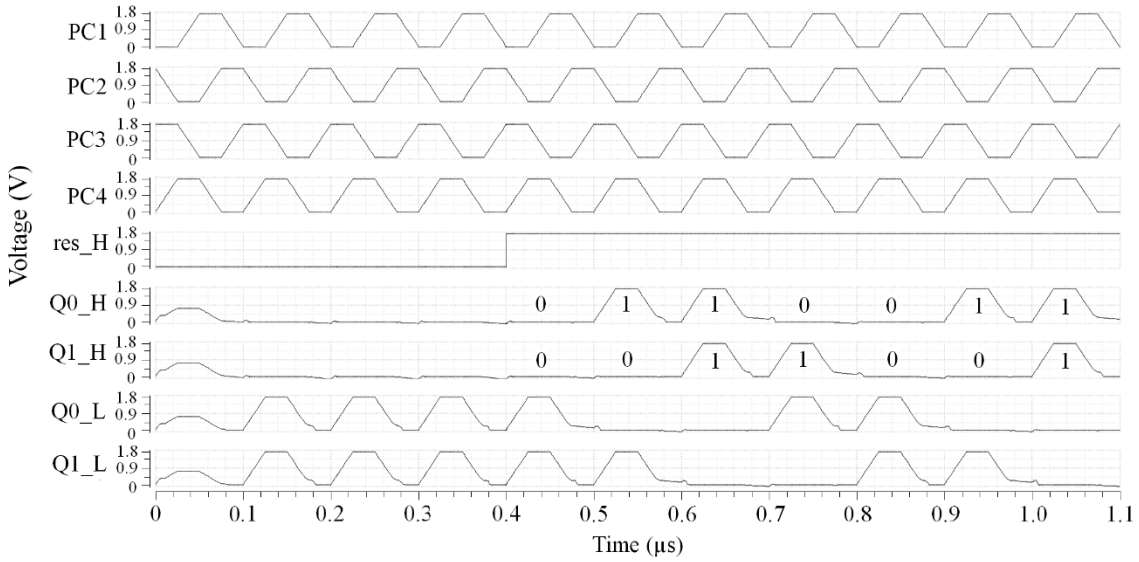

**Figure 5.11:** 4-phase power-clock.

A single-rail and a dual-rail resettable input NOT/BUF gate have been designed. The single-rail is simple which is used only when the resettable buffer is working in power-clock phase 1 (PC1). Here the reset input 'res_H' has not been converted from pulse input to adiabatic input. When 'res' is at logic '0', the counter outputs (Q0_H and Q1_H) are at logic '0', whereas the complementary outputs (Q0_L and Q1_L) follows the power-clock depending on the dual-rail inputs. For allowing state checking and detecting invalid circuit operation, the dual-rail resettable NOT/BUF gate is also designed. The VHDL codes for both are given in appendix B.

There is no variation in the timing of the VHDL simulation to that of the SPICE (circuit) simulation of the 2-bit ring counter. The two output waveforms are shown in Figure 5.12 (a) and (b).




(a)




(b)

**Figure 5.12:** 2-bit ring counter output waveforms (a) VHDL (b) SPICE.

As the ring counter does not employ any combination logic, a 3-bit Up-Down counter was also modeled. Based on the circuit diagram of the 4-phase Up-Down counter in Chapter 3 of this thesis, the VHDL code is written which is given in the Appendix B. Here, the dual-rail resettable NOT/BUF gate is used. The block diagram is given in Figure 5.13 showing the inputs and the 4-phase power-clocks to the adiabatic core which are being generated by the pulse input to the adiabatic conversion block and the 4-phase power-clock generator respectively. Figure 5.14 (a) and (b) shows the VHDL simulation waveforms alongside the SPICE simulation waveform for 3-bit Up-Down counter. The counter design shows the accuracy of the modelling in terms of timing and the representation of the adiabatic logic technique.

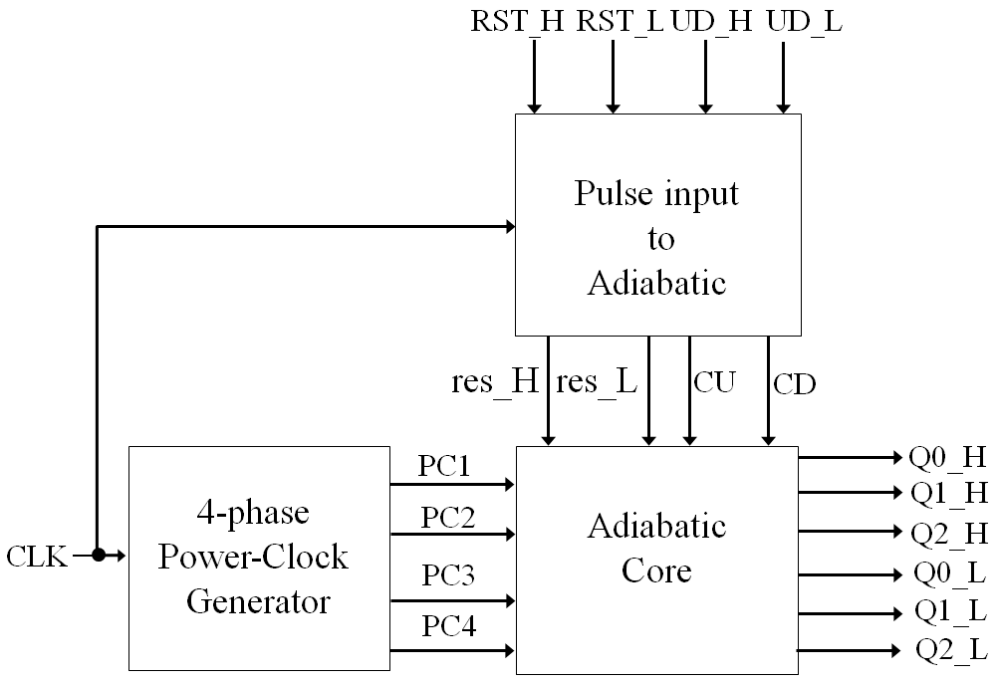
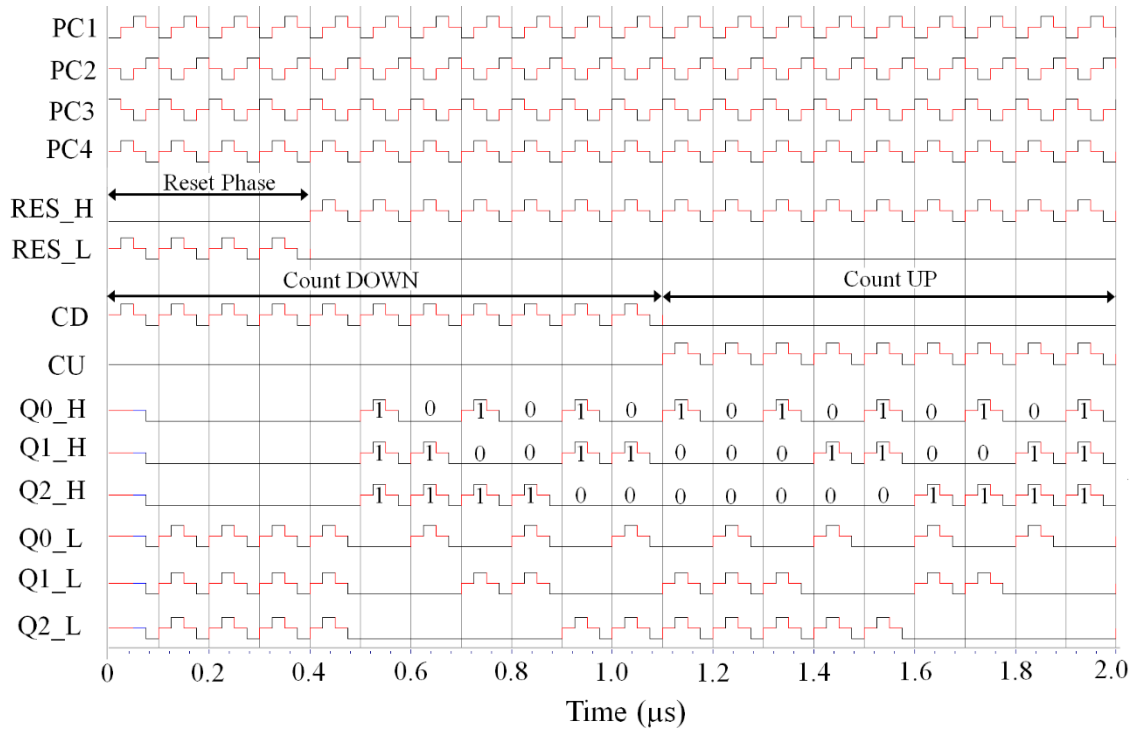
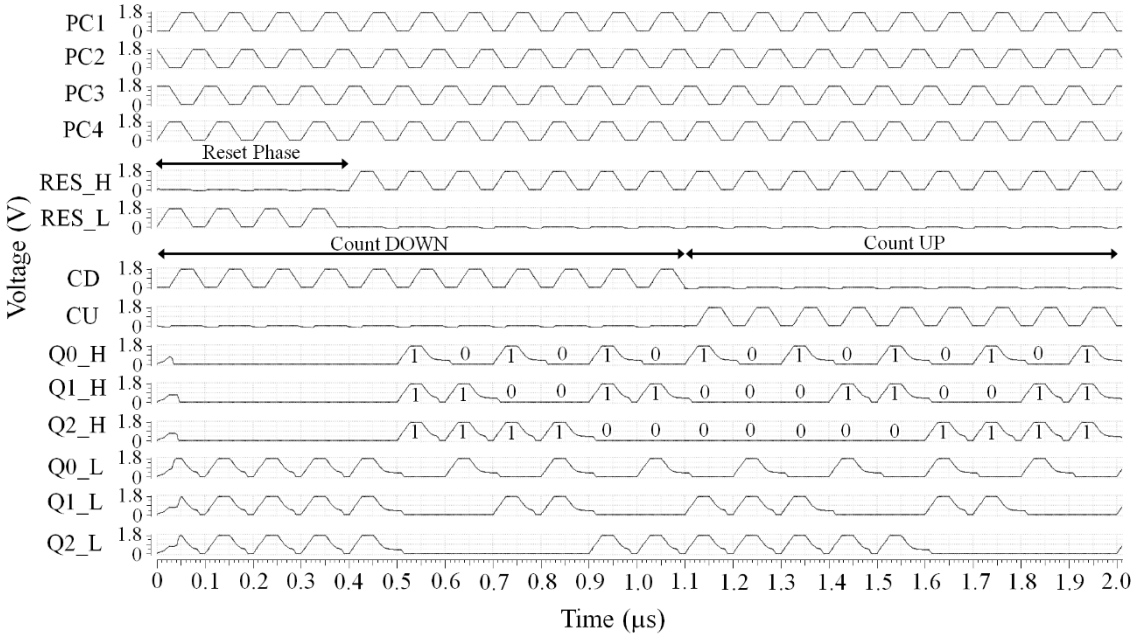



**Figure 5.13:** Block diagram of the 3-bit Up-Down counter.




(a)



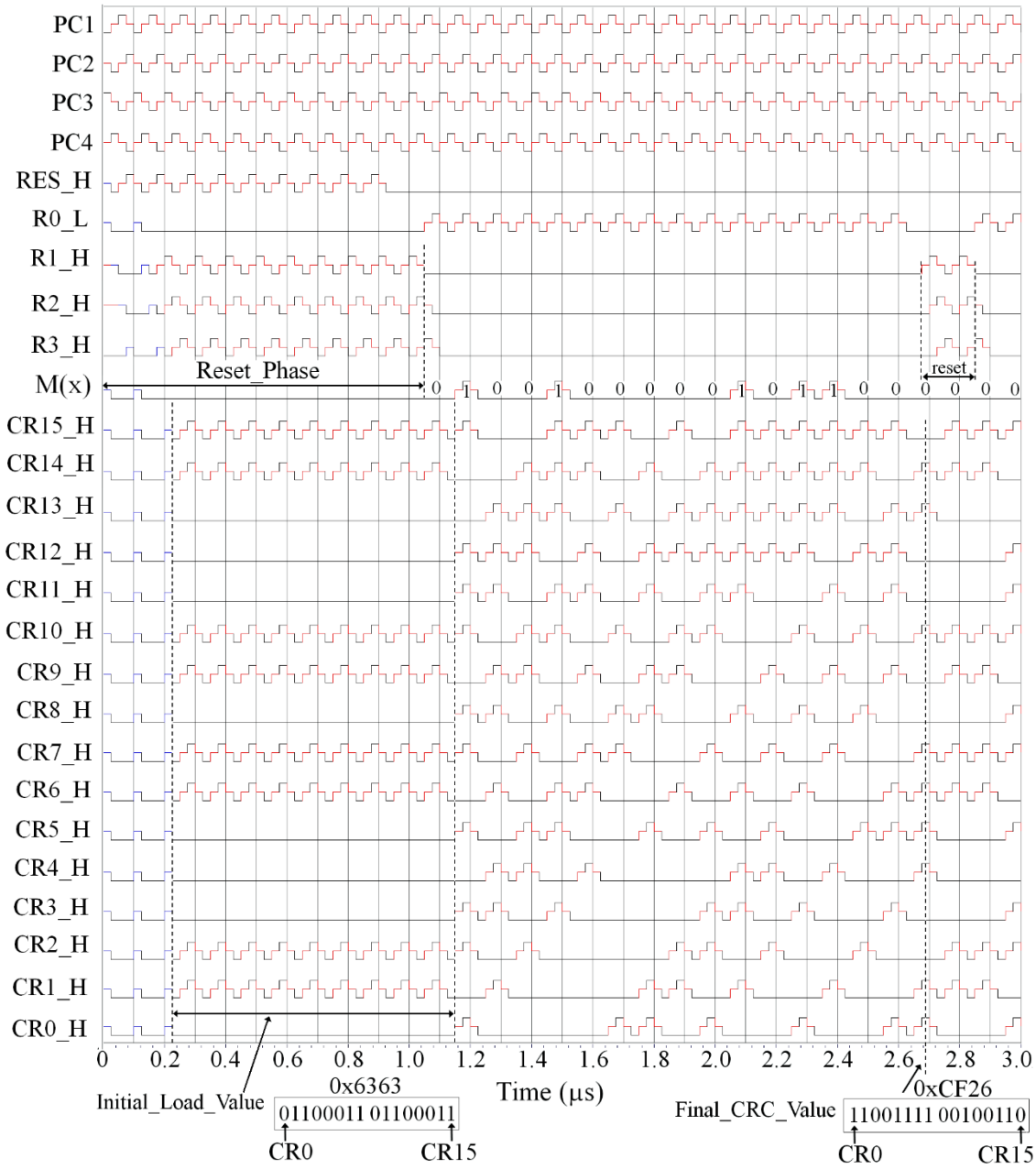

(b)

**Figure 5.14:** 3-bit Up-Down counter output waveforms (a) VHDL (b) SPICE.

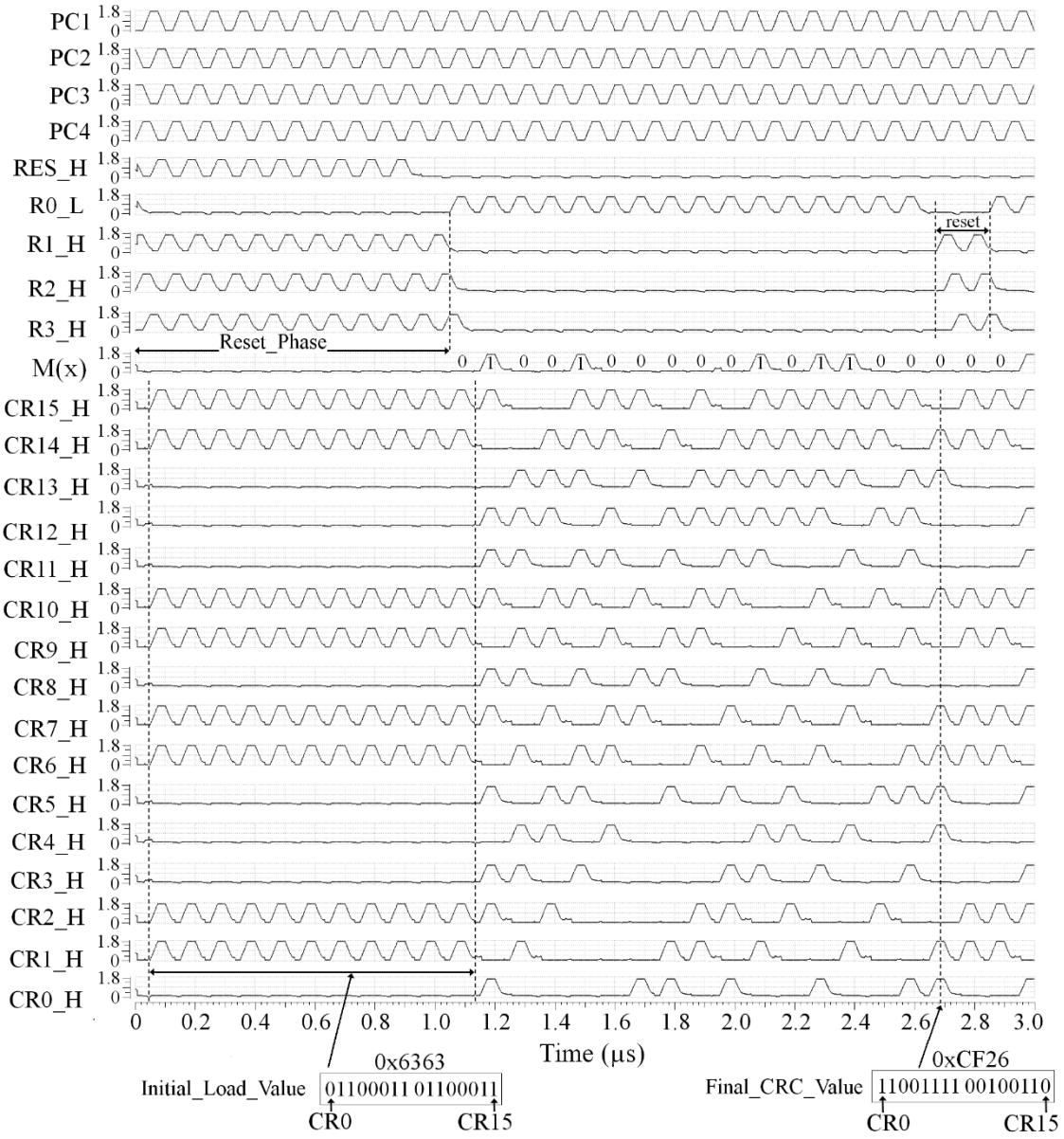
To further demonstrate the applicability of the cell library in the structural modelling of adiabatic VLSI circuits, a 16-bit CRC for the 16-bit message word length designed in Chapter 4 of this thesis was also modeled and simulated successfully.

The CRC is initialized using the reset input 'RES_H' which resets the counter to “0000” state and load the pre-set value “0x6363” to the CRC datapath. When ‘RES_L’ signal is set ‘0’, CRC starts the computation. The controller signal ‘R0_L’ is the complementary signal of ‘R0_H’, used to select the required generator polynomial. The signals ‘R1_H’ and ‘R2_H’ from the controller are the inputs to the CRC datapath acting as the reset signals whereas, the signal ‘R3_H’ serves as the select signal which loads the pre-set value to the CRC datapath during the reset operation. The CRC value is calculated when the last message bit is sent, and the counter reaches “1111” state. Then the calculated CRC value from the datapath and the message bits get appended to the register unit while the counter returns to “0000” state. The CRC value in the CRC datapath is cleared and loaded with the pre-set value. The reset period lasts for two power-clock cycles and after that, the counter starts counting again automatically allowing the CRC to re-calculate its value.

Figure 5.15 (a) and (b) shows the VHDL and the SPICE simulation waveforms for 16-bit CRC. Both the simulation was run on the same platform with the same resources. The SPICE simulation using the high performance spectre simulator XPS-MS takes 117% longer than the VHDL ModelSim simulator. The VHDL simulation result shows the precise time modelling when compared to the SPICE result. The only difference is the delay gained by the VHDL implementation at the start of the simulation. This is because, in VHDL modelling, the pulse inputs are converted to the adiabatic inputs which are then passed through the buffer gates for generating inputs for the cascade logic working in the respective power-clock phases. Whereas, in transistor level design the inputs are given based on the requirement of the power-clock input phase for the cascade logic. If the inputs in transistor level design are processed similar to the VHDL design, then both the simulations will have the same initial delay at the start of the simulation.


PC1
PC2
PC3
PC4
RES_H
R0_L
R1_H
R2_H
R3_H
M(x)
Reset_Phase
reset
0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
CR15_H
CR14_H
CR13_H
CR12_H
CR11_H
CR10_H
CR9_H
CR8_H
CR7_H
CR6_H
CR5_H
CR4_H
CR3_H
CR2_H
CR1_H
CR0_H
0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
Time (µs)
Initial_Load_Value
0x6363
01100011 01100011
CR0
CR15
Final_CRC_Value
0xCF26
11001111 00100110
CR0
CR15

(a)




(b)

**Figure 5.15:** 16-bit CRC waveforms output waveform (a) VHDL (b) SPICE.

It can be concluded for the VHDL implementation of the 16-bit CRC that the modelling approach presented in this chapter shows the possibility of an efficient design approach for timing characterization of a high-end complex adiabatic system.

## 5.6 Summary

This chapter discusses the existing approaches for modelling the adiabatic logic technique. In particular, the shortcomings of the existing VHDL modelling approaches are identified and as a solution, a new multi-level event-based approach is proposed.

The proposed HDL modelling of adiabatic logic circuits shows that the precise timing as that of the SPICE simulation can be achieved. The modelling of the 4-phase adiabatic logic technique includes the generation of dual-rail adiabatic signals from dual-rail pulse input, developing VHDL model library with basic adiabatic primitive AND and OR gates (‘Aand’ and ‘Aor’) and modelling invalid complementary inputs. The exact behaviour of the trapezoidal power-clock is also represented by presenting all the four periods distinctively using VHDL.

For the verification and applicability of the proposed approach, 2-bit ring counter, 3-bit Up-Down counter, and ISO 14443 benchmark circuit, 16-bit CRC are modeled. The simulation results confirm the precise timing of VHDL modelling. The proposed modelling is easy and can be used for designing a large complex system, eventually reducing the amount of time needed for design validation. However, whilst HDL simulation is essential for early functional check and error detection, the use of SPICE simulation is still required to measure energy consumption. The novel use of all the four edges of a power-clock has enhanced the robustness and reliability of the proposed modelling for the design of the large complex adiabatic system.

# 6 Manchester Coding using Adiabatic Logic Technique

Energy plays an important role in NFC passive tags as they are powered by radio waves from the reader. The ISO/IEC 14443 [12] standard utilizes Manchester coding for the data transmission from the passive tag to the reader for NFC type-A passive communications. This chapter proposes a novel method of Manchester encoding using the adiabatic logic technique for energy minimization. The chapter also discusses the challenge associated with the implementation of Manchester encoding using adiabatic logic. The design is implemented by generating the replica bits of the actual data bit and then flipping the replica bits, for generating the Manchester coded bits. Based on the performance trade-offs discussed in Chapter 4 of this thesis, the proposed design was implemented using two adiabatic logic families namely; Positive Feedback Adiabatic Logic (PFAL) and Improved Efficient Charge Recovery Logic (IECRL) which are compared in terms of energy dissipation for the range of frequencies. Furthermore, to investigate the impact of adiabatic logic family on the power-clock generator the energy dissipation of the complete adiabatic system was measured including the power-clock generator designed using 2-StepWise Charging circuit and a controller generating control signals is considered.

## 6.1 Introduction

The energy cost in passive NFC system consists of first initializing the communication (powering the tag) and then exchange of information wirelessly between the reader and the tag [5]. When the tag comes near the reader, it initiates the communication and the

data through modulation is transmitted between the reader and the tag [102]. In order to reduce the error rate and improve data efficiency, the data is encoded before being modulated. Different encoding techniques are used for the data to be transmitted from the reader to the tag and vice-versa. Miller coding is used to transmit data from the reader to the tag, whereas, Manchester coding is used for the data transfer from the tag to the reader [8]-[10]. A dynamic binary search algorithm [103] is used to initialize and select an NFC-A tag, where Manchester coding makes it possible to detect collision bitwise as per ISO 14443-3A standard [13]. Cyclic Redundancy Code (CRC) which is added to the end of the transmitted data was done and presented in Chapter 4 of this thesis. In this chapter, adiabatic implementation of the Manchester coding is presented.

## 6.2 Initialization and Anti-collision (ISO/IEC 14443-3A)

When a tag comes within the working field of the reader, the communication between the tag and the reader is first established. Two scenarios can occur, first it may happen that the reader is already communicating with another tag, secondly, multiple tags are present simultaneously in the working field of the reader. A way must be provided to allow interference-free communication with a single tag. Establishing communication between a tag and a reader and the anti-collision methods to be used for selecting one tag from multiple are described in part 3 of ISO/IEC 14443 [13]. The standard contains specifications for two types of tags, Tag-A and Tag-B. Each type uses different coding schemes and anti-collision methods. Anti-collision method for multiple tag identification is divided into two main algorithms: Binary tree-based deterministic for Type-A tag [103] and Aloha-based probabilistic for the Type-B tag [104]. Additionally, Type-A tag uses Manchester coding before sending the data for load modulation using Amplitude Shift Keying (ASK), whereas, Type-B uses Non-Return to Zero (NRZ) coding of data which is sent to the reader after load modulation using Binary Phase Shift Keying (BPSK). Table 6.1 summarizes the modulation schemes, coding and anti-collision method for these two types of tags.

In this thesis, an NFC type-A tag has been considered. A variant of the binary search algorithm which is a dynamic binary search algorithm [103] is used to initialize and select Type-A tag. When the reader gets the acknowledge command, Answer to Request type-A (ATQA), from the tag, it recognizes that one tag is present within the reader's field. It then initiates the anti-collision procedure which allows reading the type-A

unique identifier (UID) by transmitting the SELECT command. If the reader determines the complete UID, it transmits a SELECT command with the UID transmitted by the tag. The tag with the corresponding UID confirms this command by transmitting a SELECT acknowledge (SAK).

**Table 6.1:** Modulation, Coding and Anti-collision method for ISO 14443-3 [13].

| NFC passive tag | Modulation scheme | Coding Method | Anti-collision method |
|---|---|---|---|
| Type-A | Amplitude Shift Keying (ASK) | Manchester | Binary search |
| Type-B | Binary Phase Shift Keying (BPSK) | Non-Return to Zero (NRZ) | Aloha |

However, if two or more tags are located in the reader's field, they react simultaneously to the reader's SELECT command and starts sending their UIDs and the reader will receive the data superimposed on each other. If the reader detects the collision, it responds by transmitting bit-oriented anti-collision frames [13]. The bit-oriented anti-collision frame is divided into two parts. The valid bits before the collision is the first part. After the collision have been detected the reader sends back this part of the UID followed by a '1' bit. Only the tag whose part of the UID that matches with that of the data transmitted by the reader will send the remaining bits to the reader which forms the second part of the anti-collision frame. Thus, the amount of bit transmission is reduced as the selected tag sends only the bits after the collision has occurred. In general, the dynamic binary search algorithm decreases the collision bits step by step to achieve the goal of identifying the conflicting tags. This improves the communication quantity and reduces the communication time. The detailed anti-collision algorithm is given in part 3 of the ISO/IEC 14443A standard and the example is given in Annex A of the same [13].

Detecting a collision bit position in the reader is the foundation of the binary search algorithm. In this algorithm, the data from the tags are encoded using Manchester encoding. It detects the collision by the fact that the superimposition will cause the carrier to be modulated by the subcarrier for the full duration of one or more of the bit interval. For example, consider two 8-bit ID numbers 11100101 and 10101000 that have different bits at the second, fifth, sixth and eighth place from left to right, which the reader can not determine clearly the signal on these four bits, so the received signals

become 1?10??0?. Where, ? represents an uncertainty which results in collision bit at the second, third, sixth and eighth places. The conventional and the proposed Manchester encoding is discussed in the next section. Figure 6.1 shows the collision of two-bit sequences with Manchester coding (Type A).

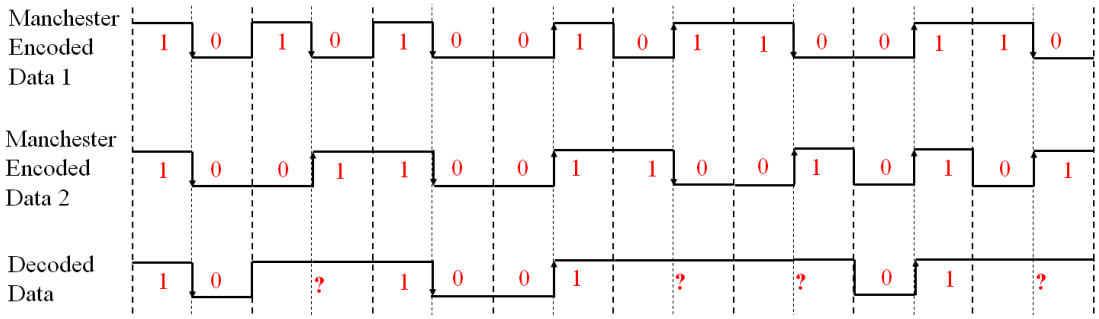



**Figure 6.1:** Collision of two Manchester encoded bitstream.

## 6.3 Adiabatic Implementation of Manchester Encoding

Coding techniques define how accurately, efficiently and robustly a message is constructed from the data that needs to be communicated. Manchester code (also known as Phase Encoding (PE) or bi-phase) is a line code in which the encoding of each data bit has at least one transition and occupies the same time [9]. The idea is to have multiple transitions in the data bitstream for a long sequence of '1' and '0' data. Manchester encoding is based on the synchronous clock encoding technique and provides a means of adding the data rate clock to the message [9]. It uses "two-level state changes" to represent 0 and 1. Logic '0' is indicated by a '0' to '1' transition and logic '1' is indicated by a '1' to '0' transition. Bi-phase Manchester code is one, where logic '1' is encoded as the transition from '0' to '1', and vice versa. Manchester code reports an error when the state doesn't change in a clock cycle, indicating the collision of data. So, when the reader receives bits from the tags and some bits do not change, the reader can know which bits are conflicting with each other. The waveform for a Manchester encoded bit stream '11100101' is shown in Figure6.2. The simplest way of implementing the Manchester encoding requires an exclusive NOR (XNOR) function between the clock (CLK) and the data bit stream, whereas, exclusive OR (XOR) for bi-phase Manchester encoding.

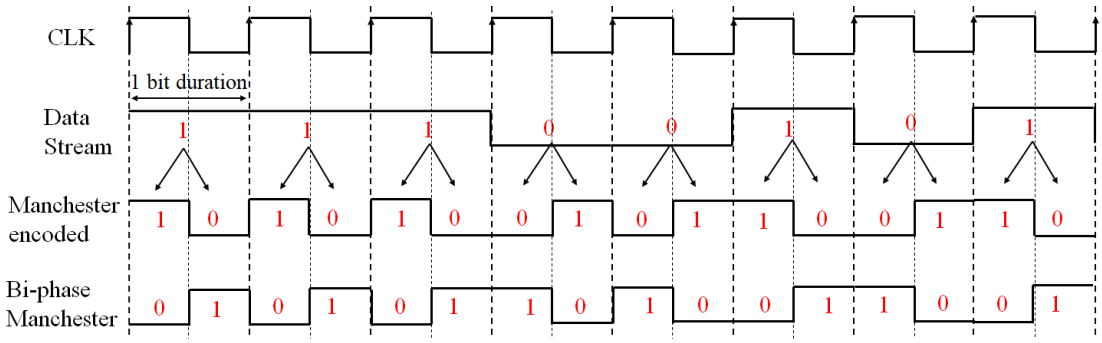



**Figure 6.2:** Manchester encoding waveform for multiple data bits.

Manchester coding using the adiabatic logic technique is challenging due to the following reasons;

1) In the adiabatic logic technique, the input and the power-clock both have the same time-period (frequency), having a $90^{o}$ phase difference. In addition, the output follows the power-clock depending on the input.

2) If the two different power-clock frequencies are used in the same circuit then it may be necessary to generate them separately. This will add to the complexity of the power-clock generator incurring overhead in terms of energy consumption and area of the complete adiabatic system.

3) The use of two different frequencies will violate the adiabatic principle and cause the energy dissipation to increase. This is specifically due to the fact that the output following the power-clock.

Therefore, for the adiabatic implementation, a new method and hardware are required for encoding the data bit stream such that the long strings of 1s and 0s are avoided. One of the advantages of the adiabatic implementation is that no separate power-clock needs to be added to the data bit stream. In fact, as the input has the same time period as that of the power-clock, the clock and the data can easily be recovered at the receiver side from the Manchester coded data. Figure 6.3 shows the relationship between the input and PC for PFAL NOT/BUF gate.

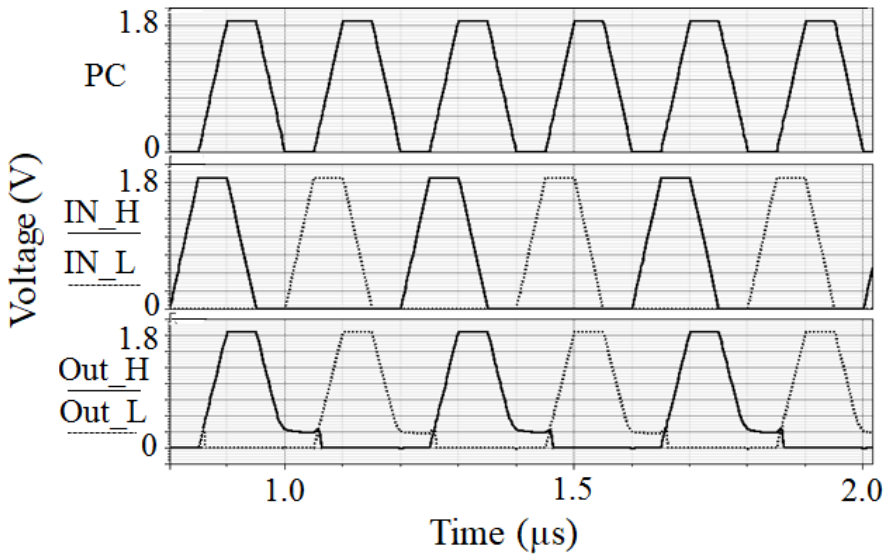



**Figure 6.3:** Relationship between PC, input and output waveforms in the adiabatic logic technique.

Since the adiabatic logic has the constraints due to PC and the input having the same time period, in the proposed method the time-period of the data bit stream is doubled such that each bit in the data bit stream occurs twice consecutively. This way the replica image of the actual data bit stream is created. After this, the flipping of the replica bits takes place which generates the Manchester coded bit stream ready to be sent to the reader. This complete process ensures that there are no consecutive '1' and '0' in the transmitted coded data bit stream. Figure 6.4 shows the Manchester encoding using the proposed method. The encoding of the data bit stream is described as follows;

Step 1: Data bit stream stored in the register implemented using adiabatic logic.

Step 2: Using a 2-state counter, each bit in the data bit stream from the register is read twice in consecutive power-clock cycles, such that each bit from the actual data stream is replicated and forms a bit pair. Each pair in the data stream contains the actual and the replica bits. The second bit of each pair in the replicated bitstream is the replica bit.

Step 3: The complement of the replicated bit stream is generated.

Step4: The replicated bit stream and its complement are multiplexed to form Manchester coded data bit stream.

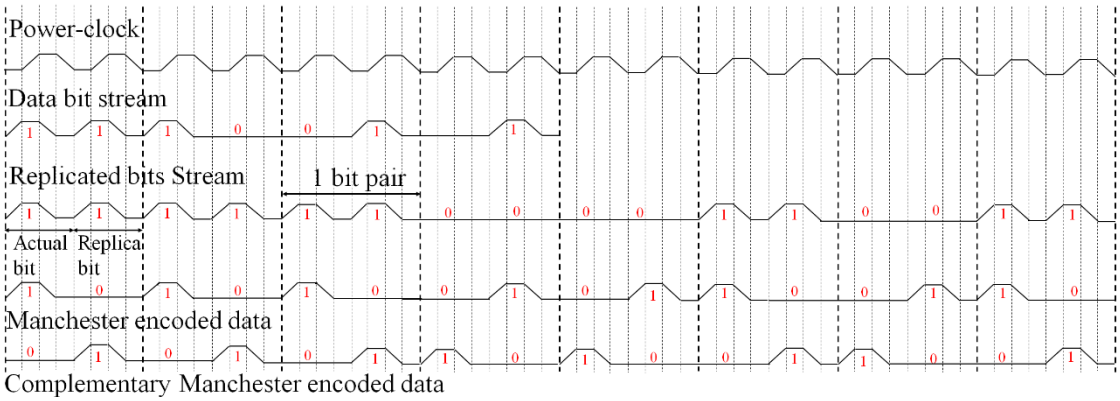



**Figure 6.4:** Manchester encoding waveform using adiabatic logic for multiple bits.

From Figure 6.4, it can be seen that every single data bit is encoded into an actual and replica bit. Manchester code using adiabatic logic can be defined as a code in which the encoding of each bit of double length occurs in pairs for two power-clock cycles and has exactly one transition either from '1' to '0' or vice-versa. So, when the reader receives Manchester coded bits from multiple tags and some bits do not change in a bit pair then the collision exists in that particular bit pair. Figure 6.5 shows the collision when two Manchester coded data using the proposed method arrives in the reader for identifying the collision bits.

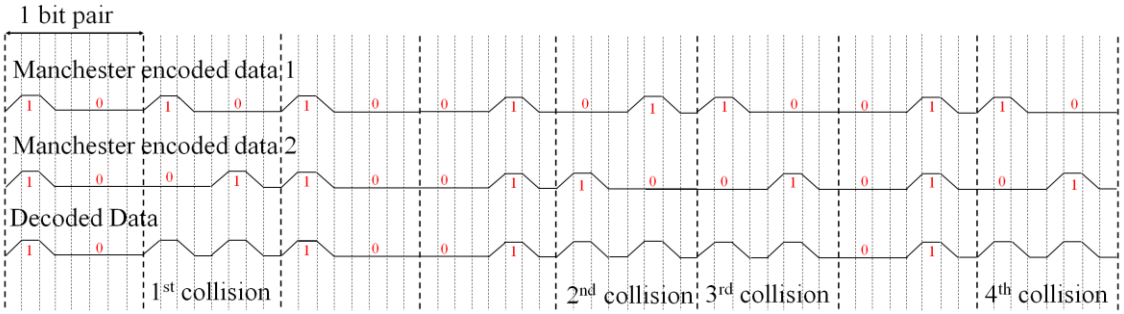
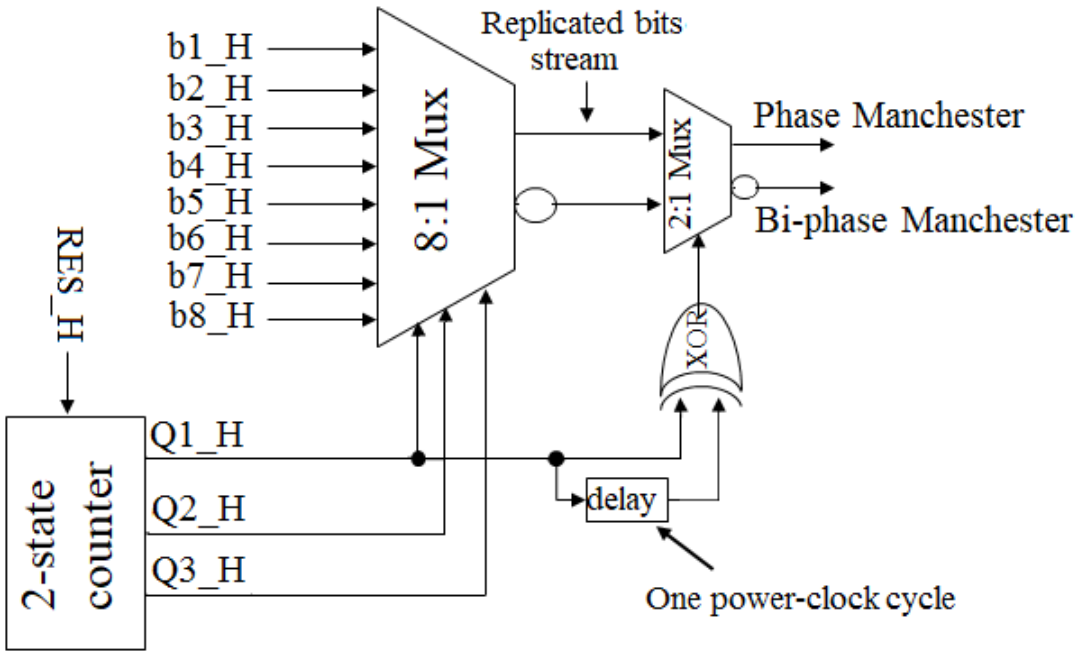


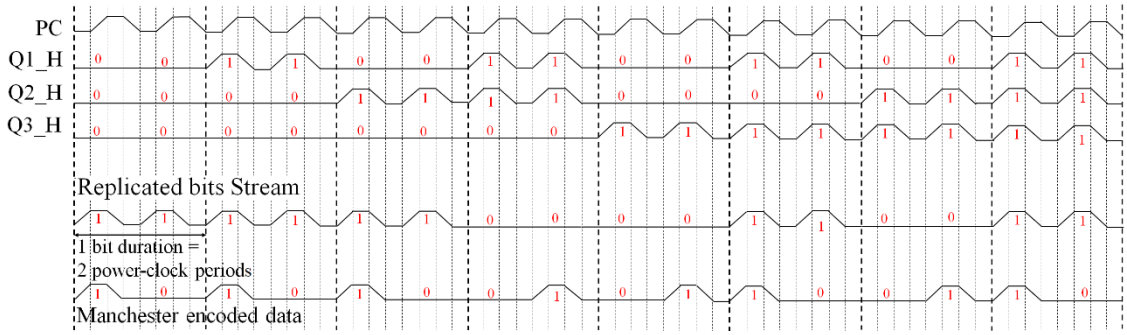

**Figure 6.5:** Collision of two Manchester encoded data using the proposed method.

The hardware requirement for the proposed Manchester encoding using adiabatic logic is as simple as the conventional encoding method which requires XOR gates. However, the proposed method requires a few more logic gates for replicating the bit stream and multiplexing it to generate the Manchester encoded data. Unlike conventional CMOS logic, the adiabatic logic generates complementary signals which give the advantage of generating both phase and bi-phase Manchester encoded data at the same time.

The adiabatic design of the Manchester coding for the 8-bit data stream is shown in Figure 6.6 (a), whereas, Figure 6.6 (b) shows the corresponding output waveforms of the 2-state counter, replicated bits stream, and Manchester coded signal. The input bits stream b1_H -- b8_H is taken as '11100101' as an example.




(a)




(b)

**Figure 6.6:** Proposed Manchester encoding (a) Circuit diagram (b) Output waveforms.

For generating the replica bits, the proposed encoding method uses a two-state counter. A two-state counter is one whose count remains in the same state for two consecutive power-clock cycles. Figure 6.7 shows the circuit for the two-state counter. The counter counts from "000" state to "111". The output of the counter is the select input to the 8:1

multiplexer which sends each bit to 2:1 multiplexer serially. 8:1 multiplexer is designed using two 4:1 and one 2:1 multiplexer. The circuit design for 2:1 multiplexer is depicted in Figure 3.1 in Chapter 3 of this thesis. After this, the replicated data bit stream and its complementary data stream are multiplexed. The Manchester encoded signal is generated by sequentially turning the switch ON and OFF between the replicated and its complementary data bit streams. The select signal to the 2:1 Multiplexer is generated by the XOR operation between the LSB bit of the counter and its delayed bit (by one power-clock cycle) to maintain synchronization between the bit stream and the counter. The delay of one power-clock cycle is generated by connecting four buffer gates in cascade. The circuit diagram is depicted in Figure 3.7 in Chapter 3 of this thesis.

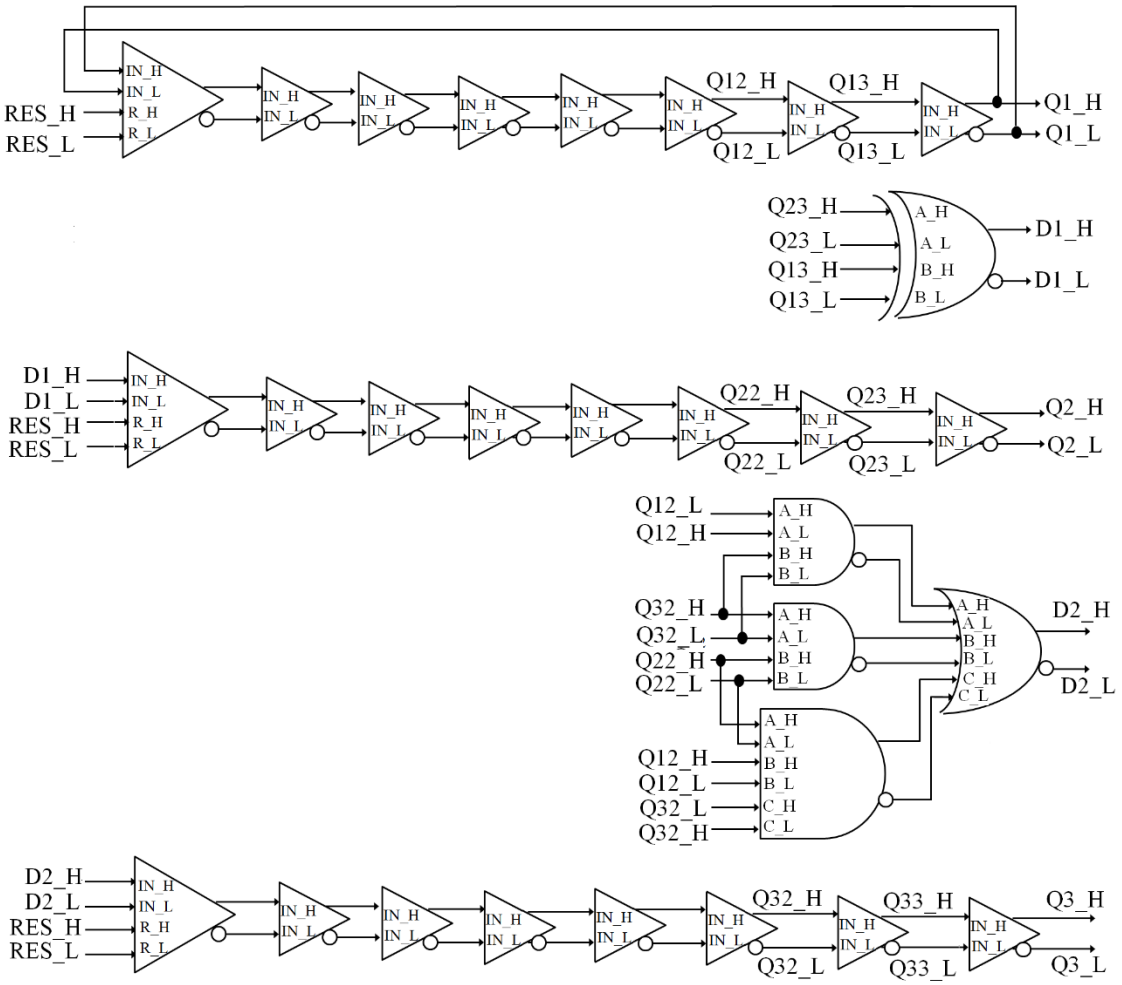



**Figure 6.7:** 2-state counter.

The only disadvantage of the proposed method is the increase in the number of effective transitions and an effective doubling of the bit duration to two power-clock cycles. This

doubles the computation time, however, it is expected to be energy efficient due to the adiabatic implementation. To the author's best knowledge, this is the first time Manchester encoding using adiabatic logic technique is proposed and implemented.

## 6.4 Simulation Results

Based on the performance results presented in previous Chapters 3 and 4, the 4-phase adiabatic logic families have been chosen. Specifically, the 4-phase adiabatic logic families used for the SPICE simulation are PFAL and IECRL. The transistor sizes for both were set to the technology minimum. The simulations were performed with the Spectre simulator using the Cadence EDA tool in a 'typical-typical', TT process corner using TSMC 180nm CMOS process at 1.8V power supply.

Since the implementation of Manchester encoding is a part of initialization and anti-collision of the ISO standard 14443-3 [13], the energy dissipation was measured as the energy per cycle for 40 data bits. The energy measurement is taken at various PC frequencies ranging from 1MHz to 100MHz for a load capacitance of 10fF. Figure 6.8 shows the simulation waveform for PFAL at 10MHz PC frequency for 40 bits data stream. The waveform shows the 2-state counter, replicated data bit stream and the phase and bi-phase Manchester encoded data along with 4-phase power-clock. When the 'RES' is logic '1' the complete system is at zero states. The system starts the computation when the 'RES' signal goes to logic '0'.

The comparison of energy per power-clock cycle of PFAL and IECRL is shown in Table 6.2. It can be seen that IECRL implementation has the lowest energy compared to PFAL. The increment in PFAL energy is not significant from 1 MHz to100MHz in comparison to that of IECRL. Though, Table 6.2 shows IECRL consumes minimum energy, however, as stated in Chapter 2 under section 2.5 that increase in load capacitance will increase its energy consumption due to the non-adiabatic losses occurring during both the evaluation period and the recovery period [30], [31].

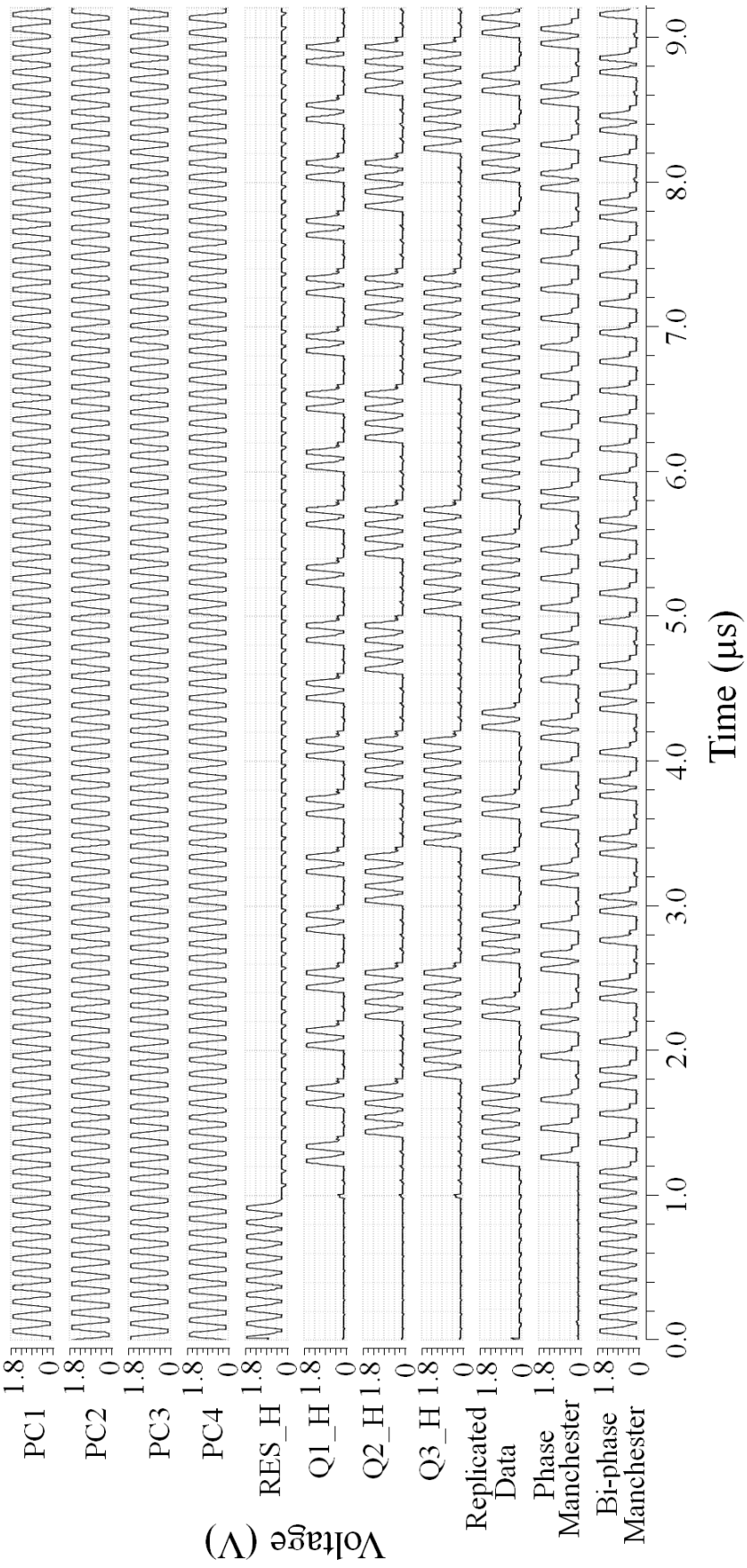



**Figure 6.8:** SPICE simulation waveform for the proposed Manchester encoding.

**Table 6.2:** Comparison of Energy per power-clock cycle of PFAL and IECRL.

| | **Energy Consumption per power-clock cycle (fJ)** | | |
|---|---|---|---|
| **Frequency (MHz)** | 1 | 10 | 100 |
| PFAL | 253.30 | 253.78 | 310.38 |
| IERCL | 119.98 | 119.46 | 304.77 |

Figure 6.9 shows the graph for the two adiabatic logic families at varying load capacitance at 10MHz PC frequency. It is observed that IECRL show a steeper response in energy consumption as the load capacitance increases in comparison to PFAL. Nevertheless, the above comparison is inconclusive without taking consideration of the power clock generator.

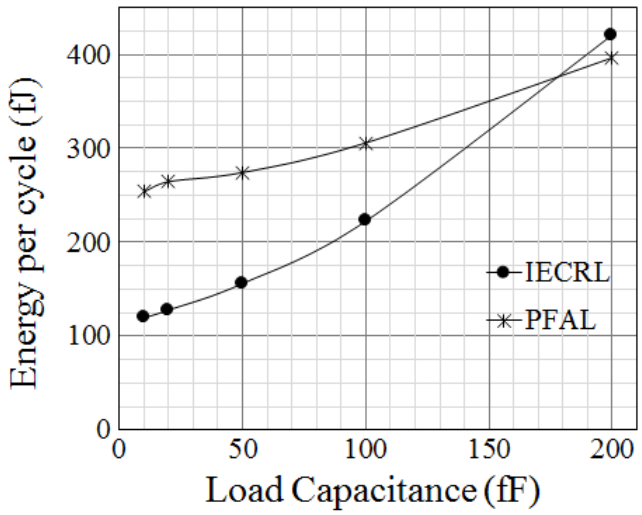



**Figure 6.9:** Energy consumption per power-clock cycle vs load capacitance.

As stated in Chapter 4, the Power Clock Generator (PCG) circuitry is an important part of an adiabatic logic design. It accounts for a significant amount of energy compared to the adiabatic design as it is designed using conventional CMOS logic. PCG used for the simulation is a 4-phase PCG using SWC circuit. The clock frequency (CLK) is taken as 40MHz generating a power-clock frequency of 10MHz (ramping time of 25ns), supply-voltage is 1.8V and 10fF capacitive load attached to the output of the adiabatic core. The complete adiabatic system designed comprises the power-clock generator and the adiabatic core is shown in Figure 6.10. PCG is implemented using a 2-step charging circuit.

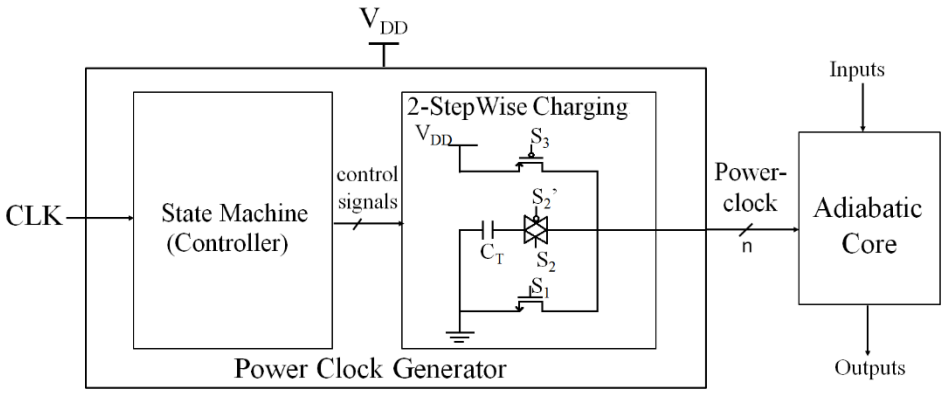



**Figure 6.10:** Complete Adiabatic System with 4-phase PCG using SWC circuit.

The tank capacitance ($C_T$) chosen for both the logic families is 5pF. The aspect ratio of the SWC circuit for both the logic families is taken to be the same for fair comparison and evaluation of the two adiabatic logic families. The adiabatic core is the Manchester encoding circuit.

**Table 6.3:** Comparison of energy per power-clock cycle of the adiabatic system using PFAL and IECRL.

| | **Energy Consumption per power-clock cycle (pJ)** | | |
|---|---|---|---|
| | Controller | SWC+core | Total |
| PFAL | 1.040 | 1.524 | 2.565 |
| IERCL | 1.002 | 1.098 | 2.10 |

Table 6.3 reports the energy consumed by the adiabatic system including the power-clock generator for encoding the data bit stream into Manchester code. From Figure6.10, it can be seen that the adiabatic core is the load to the SWC, hence the energy is measured for the controller and the SWC by measuring the current at the supply voltage ($V_{DD}$) separately. It is also worth noting that, that the energy consumption of the controller for the SWC is almost constant for all system size [54] for the fixed power-clock cycle. More importantly, the energy consumption of the SWC for the PFAL family is approximately 40% more in comparison to IECRL. This is because the evaluation network of the PFAL family being connected between the power-clock and the output as depicted in Figure 2.6 (a) of Chapter 2. This connection adds on the extra capacitance of a minimum of two nMOS transistors (drain terminal) connected to

the power-clock in addition to the two pMOS transistors. As a result, PFAL presents a large load capacitance to SWC. On the other hand, IECRL has its evaluation network connected between the output nodes and the ground, as a result, PC is connected only to the two pMOS transistors, as depicted in Figure 2.6 (b). Therefore, IECRL shows a decrease in capacitance value compared to PFAL, hence shows a reduction in energy dissipation of SWC.

## 6.5 Summary

This Chapter gives a brief discussion of the initialization and anti-collision protocol based on ISO/IEC 14443A standard. Specifically, the use of Manchester encoding for data transmission is discussed. It is one of the important parts in wireless data communication and is used in the anti-collision algorithm to increase the data efficiency which aids in identifying the collision bit in the reader after transmission. In addition, challenges associated with the implementation of Manchester encoding using adiabatic logic technique is discussed. Furthermore, a novel method of encoding data bit stream into Manchester coding using a 4-phase adiabatic logic technique is proposed. The proposed implementation generates phase and bi-phase Manchester encoding data simultaneously, as depicted in Figure 6.2, due to the dual-rail adiabatic logic design. Since the power-clock and the input have the same time period, the adiabatic implementation has an advantage over conventional CMOS method, i.e. it does not require a clock signal to be exclusively XORed with the data bit stream. At the receiver side, the decoding of the received Manchester coded data takes place, where the actual data and the clock is decoded and will be checked for collision. The only disadvantage the proposed method has is that it doubles the effective bit duration. However, as the working frequency of NFC technology is centered around 13.56MHz, the proposed method can easily be used for such applications. From the simulation results, it is concluded that IECRL consumes less energy compared to PFAL at 10fF load capacitance. If the load capacitance increases the IECRL adiabatic core energy will increase but it is anticipated that the overall system energy will not be more compared to PFAL due to the extra capacitance (a minimum of two pMOS transistors).

# 7 Conclusion and Future Work

This Chapter summarises the achievements of this research work in a reflective manner and provides the author's recommendations for further work relating to the adiabatic approach.

## 7.1 Research Summary

The main motivation of this thesis was to exploit the energy efficient traits of the adiabatic logic technique to deliver the ultra-low power operation for NFC applications. Chapter 2 reported in this thesis builds the foundation on the general background of the adiabatic logic technique and several adiabatic logic families working on single-phase, 2-phase and 4-phase power-clocking schemes. Due to the divided opinions on the most energy efficient adiabatic logic family which also constitute an appropriate trade-off between computation time, circuit complexity and power-clocking scheme complexity, five of the most energy efficient adiabatic logic families namely; CAL, CPAL, IECRL, EACRL and PFAL were chosen for further research and scrutiny which has formed the basis of this research.

Due to the requirement of resettable buffers in sequential logic designs, novel resettable buffers for the five chosen adiabatic logic families were designed. The proposed novel adiabatic resettable buffers used for the design and layout implementation of resettable flip-flops shows a reduction in energy and layout area consumption compared to the existing MUX-based resettable adiabatic flip-flops. Additionally, it has been shown through the design of 2-bit twisted ring counter and 3-bit Up-Down counter that PFAL and IECRL driven by a 4-phase power-clocking scheme constitute appropriate

performance trade-offs between energy consumption, computation time, power-clocking scheme and circuit complexity. This has been discussed in Chapter 3 of this thesis.

To enable future designers and researchers in this subject area to use quantitative information on selecting the required power-clocking scheme and robust adiabatic logic family, a 16-bit CRC test circuit was implemented using the five chosen adiabatic logic families as an application and system component of an NFC tag. A methodology and strategy for CRC implementation using adiabatic logic working on single-phase, 2-phase and 4-phase power-clocking scheme was proposed. A modification in the bit-serial CRC design was also done by incorporating more functionality which allows for the use of any CRC-16 generator polynomial and any initial load values. Significant differences in functionality and robustness under voltage scaling and PVT variations among multiphase adiabatic implementations were discovered. Moreover, considering the supply voltage scaling, it has been shown that the benefit of using adiabatic logic deteriorates for supply voltage less than 1.2V. Therefore, a functional range for the supply voltage scaling is proposed for better Energy Saving Factor (ESP) and correct functionality. Finally, based on the performance trade-offs between energy dissipation, computation time, area, robustness under PVT variation, supply-voltage scaling and power-clock generator complexity in a large adiabatic system, it was concluded that IECRL shows the best performance results followed by PFAL. This has been described in Chapter 4 of this thesis.

To overcome the problem of excessive design time and difficulty of identifying and debugging the errors in a large adiabatic system arising due to the complexity of the 4-phase power-clocking scheme, a new modelling approach using VHDL for a 4-phase dual-rail adiabatic logic family was proposed. The shortcomings of the existing modelling approaches are reported. The proposed modelling provides the solution to the shortcomings of the existing modelling approaches. The modelling of the 4-phase adiabatic logic technique comprises of the generation of dual-rail adiabatic signals from dual-rail pulse input, developing VHDL model library and modelling invalid complementary inputs. The accurate behaviour of the trapezoidal power-clock was represented by presenting all the four periods distinctively using VHDL. For the verification and applicability of the proposed approach, 2-bit ring counter, 3-bit Up-Down counter, and 16-bit CRC circuits were modeled. The simulation results confirm

the precise timings of VHDL modelling. The proposed modelling is easy and can be used for designing large complex adiabatic systems, ultimately reducing the amount of time needed for design validation. This aspect of the work was covered in Chapter 5 of this thesis. The method is also applicable for single phase and 2-phase power-clocking scheme.

Finally, a novel method of encoding data bits into Manchester coding using the adiabatic logic technique is presented. The proposed implementation generates phase Manchester, as well as bi-phase Manchester, encoded data simultaneously due to the dual-rail adiabatic logic design. In the adiabatic logic technique, since the power-clock and the input have the same frequency, the adiabatic implementation has an advantage over conventional CMOS method, that is, it does not require a clock signal to be exclusively XORed with the data bits. At the receiver side, the decoding of the received Manchester coded data take place, where the actual data bit stream and the clock are decoded and will be checked for collision. The only disadvantage the proposed method has is that it doubles the bit duration which makes the proposed method applicable for application working at low frequency. From the simulation results, it is concluded that at an adiabatic system designed using IECRL family consumes 40% less energy compared to PFAL at 10fF load capacitance. This aspect of the work is discussed in Chapter 6 of this thesis.

## 7.2 Novelty Contributions (listed in the order of significance)

The contents of this section are in order of significance, unlike Chapter 1 Section 1.4, which is in chronological order.

1. VHDL modelling of the Adiabatic Logic.

   To overcome the synchronization problem arising due to the complexity of the 4-phase power-clocking scheme and to reduce the design, validation and debugging time, a new method for modelling 4-phase adiabatic logic in VHDL was proposed. This will enable the designers and researchers to design and validate the adiabatic design in a short span of time ahead of actually designing the transistor level schematic. Additionally, the existing modelling approaches model the adiabatic circuits using square shaped power-clock (as is done for modelling the

conventional CMOS circuits) instead of trapezoidal power-clock and therefore, fails to follow the adiabatic principles.

a. Shortcomings of the existing (Hardware Description Language (HDL) modelling approaches were identified. A very close to the exact behaviour of the trapezoidal power-clock was represented by modelling all the four periods distinctively using VHDL. The verification and applicability of the modelling were verified using a 2-bit ring counter and a 3-bit Up-Down counter.

b. The proposed modelling is easy and can be used for designing large complex adiabatic systems, eventually reducing the amount of time needed for the design and validation of such systems. The VHDL code of the NOT/BUF gate is further enhanced by incorporating an invalid condition check in cascade logic designs. Additionally, the gate level adiabatic modelling of the primitive AND and OR gates were also done. The enhanced proposed modelling demonstrates the error due to the use of a square power-clock in acquiring precise timing. A more complex circuit such as a 16-bit CRC is used to show the robustness of the proposed VHDL-based modelling approach for the 4-phase adiabatic logic technique for functional and timing simulations.

2. A novel method of Manchester encoding using the adiabatic logic technique for energy minimization is proposed. First, the time period of the data bit stream is doubled such that each bit in the data bit stream occurs twice consecutively. This way the mirror image of the actual data bit stream is generated. Then the flipping of the mirror bits takes place which generates the Manchester coded bit stream ready to be sent to the reader. The adiabatic implementation is advantageous as no separate clock needs to be added to the data stream. In fact, as the input has the same frequency as that of the power-clock, the power-clock and the data can easily be recovered at the reader from the Manchester coded data stream. This is the first time in the literature where adiabatic implementation of the Manchester encoding is done for the energy efficient implementation of the NFC applications.

3. Cyclic Redundancy Check (CRC) is one of the main components used in passive NFC systems, whenever the data is transmitted. Therefore, for the implementation of the energy efficient NFC systems, performance trade-offs including robustness against Process-Voltage-Temperature (PVT) variations and supply voltage scaling between multi-phase adiabatic logic families in a large adiabatic system are worthy of investigation. This provides quantitative information to the designers and

researchers if the supply voltage scaling benefits the performance of the adiabatic systems. Furthermore, for the completeness of the evaluation and trade-offs proposition, it is important to investigate the performance of the multiphase adiabatic logic families in the presence of the Power-Clock Generator. This will help the designers to choose the appropriate adiabatic logic family.

a. The 16-bit CRC was implemented as deployed in an application for the passive NFC system, using multi-phase adiabatic logic families. A generic methodology and strategy for the design of multi-phase adiabatic CRC employing single-phase, 2-phase or 4-phase power-clocking scheme was proposed. Additionally, the bit-serial CRC design is modified by incorporating more functionality allowing for the use of any CRC-16 generator polynomial and any initial load values.
b. Impact of voltage scaling and Process Voltage Temperature (PVT) variations on multi-phase adiabatic implementations were investigated for TSMC 180nm CMOS process at 1.8V supply voltage. It was discovered that the benefit of using adiabatic logic deteriorates for supply voltages scaled less than 1.2V. Therefore, an optimal range for the supply voltage scaling was proposed for better Energy Saving Factor (ESP) and correct functionality.
c. When the energy dissipation of the total system comprising of the power-clock generator was considered, it was discovered that the total energy of the system employing single-phase and 2-phase adiabatic logic was approximately 3x and 2x times respectively more when compared to the 4-phase adiabatic system. Moreover, IECRL system shows the least energy consumption followed by PFAL. Any sequential design would require flip-flops as a memory element.

4. The trade-offs between adiabatic logic families working on single-phase, 2-phase and 4-phase power-clocking scheme in terms of energy, complexity, throughput, and area are proposed. Thus, enabling the designers and researchers to use quantitative information on selecting the required power-clocking scheme and adiabatic logic families.

a. The design and implementation of 3-bit Up-Down counter using multi-phase adiabatic logic for establishing systematic and appropriate performance trade-offs in terms of complexity, energy, throughput, and area. Based on the simulation results, 4-phase adiabatic logic namely; PFAL shows better performance compared to the other adiabatic logic families.

5. To design adiabatic flip-flops with reset, resettable adiabatic buffers are required. Existing resettable flip-flops, however, are based on the 2:1 multiplexer's (MUX).

   a. The proposition of novel single-phase, 2-phase and 4-phase resettable buffers for the design of flip-flops using; Clocked Adiabatic Logic (CAL), Complementary Pass-transistor Adiabatic Logic (CPAL), Improved Efficient Charge Recovery Logic (IECRL), Positive Feedback Adiabatic Logic (PFAL) and Efficient Adiabatic Charge Recovery Logic (EACRL). Prior to this, the resettable flip-flops were based on 2:1 MUXs, used as one of the resettable stages. As a result, having an increased number of transistors and an extra input terminal causing energy, routing, latency, and area overhead.
   b. The design, implementation, and the layout of the existing and the proposed resettable flip-flops based on the different power-clocking schemes using all the five(multi-phase) adiabatic logic families to act as a proof of concept. In Compared to the existing resettable flip-flops, the proposed resettable flip-flops using; PFAL, IECRL, EACRL, and CAL show an improvement in energy consumption of approximately 14%, 3%, 10%, and 3% respectively. However, the existing resettable flip-flops implemented using CPAL shows 0.5% less energy consumption compared to the proposed resettable flip-flops.

## 7.3 Future Work

The adiabatic logic technique can be used in many applications where power consumption is the critical importance rather than speed due to its energy efficient operation. Other than the application mentioned in this thesis, it can also be considered for applications such as medical devices. The author would like to make the following recommendations for future work.

### *7.3.1 Development of new adiabatic logic with high energy efficiency to the power-clock generator.*

The overall energy saving deteriorates when the power-clock generator is considered. The energy dissipation of the power-clock generator comprises of the energy consumed by the controller and stepwise charging circuit. Based on the literature review and the work presented in Chapter 3 of this thesis, it was concluded that PFAL is the most energy-efficient adiabatic logic design approach. However, when the power-clock

generator was considered, it was found that IECRL consumes the least energy out of the five adiabatic logic designs [31]. This is because of the energy consumed by the StepWise Charging (SWC) circuit. In PFAL, the evaluation network is connected between the power-clock and the output, whereas, in IECRL it is connected between the output and ground. Hence, when the power-clock is supplied through SWC, an extra capacitance due to the evaluation network connection in PFAL increases the energy consumption of the system including the power-clock generator. Therefore, a new 4-phase adiabatic logic which doesn't load the power-clock generator and at the same time has less Non-Adiabatic Losses (NAL) is worthy of investigation.

### *7.3.2 Development of CAD Tools*

The design of conventional CMOS is easily facilitated by the existence commercially available CAD tools. The design of adiabatic systems is time-consuming and difficult due to the complexity of the power-clocking scheme specifically, the 4-phase power-clocking scheme. The work presented in Chapter 5 includes the functional simulation of the adiabatic logic for detecting errors before designing a large adiabatic system at the transistor level and reduces the time consumed in identifying and debugging of errors. In order to expedite the design of large complex adiabatic systems, the development of CAD tool for logic synthesis and automatic routing is essential.

### *7.3.3 Development of Adiabatic System in Deep Sub-micron Technology*

There exist a couple of research papers that have investigated adiabatic logic at near-threshold [97], [98] and sub-threshold [89] for deep sub-micron CMOS technology. The work in the papers describe the adiabatic flip-flops and sequential circuits only. It would be worthwhile to investigate and compare the energy, area, and performance of the large adiabatic systems like communication protocol in NFC or arithmetic unit used in cryptosystem for smartcard applications(below 45nm), including the power-clock generator to give a more realistic and objective measure and plot the way for the future.

### *7.3.4 Development of the Complete Initialization and Anti-collision for NFC-A using the Adiabatic Logic Technique*

Passive tags used for NFC application have a high cost because of the increased hardware complexity, which includes security for the transaction and data-storage circuitry for storing information. This causes an increase in the tag energy requirements

as they need more energy for powering-up the circuitry, processing and transmit data messages [108], [5]. Moreover, when collisions occur due to multiple tags in the reader's proximity it results in wastage of bandwidth, energy and increased identification time of the passive system due to the increase in re-transmission. Anti-collision protocol is the main part of the digital processing unit in the NFC tag. The Anti-collision algorithms are proposed to reduce the collisions [9], [104] such that the number of re-transmissions is reduced. There are few energy-based performance evaluations of the anti-collision protocol for passive RFID systems that can be found in the literature [105]-[109], [5] [6]. However, the use of lower-power techniques for implementation of the initialization and anti-collision protocol for passive NFC system remains unexplored.

# References


[1] K.Finkenzeller, "RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification", 2nd Ed, Wiley, 2003.

[2] P. C. Kocher, J. Jaffe, and B. Jun, “White Paper: Near Field Communication (NFC) Technology and Measurements,” Rohde & Schwarz NFC technology and measurements, 2011.

[3] T. Plos, M. Hutter, M. Feldhofer, M. Stiglic, and F. Cavaliere, "Security-Enabled Near-Field Communication Tag with Flexible Architecture Supporting Asymmetric Cryptography," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 21, No. 11, pp. 1965-1974, Nov. 2013.

[4] X. Deng, M. Rong, T. Liu, Y. Yuan and D. Yu, "Segmented Cyclic Redundancy Check: A Data Protection Scheme for Fast Reading RFID Tag's Memory," *IEEE Wireless Communications and Networking Conference*, pp. 1576-1581, 2008.

[5] F. Hessar and S. Roy, "Energy-Based Performance Evaluation of Passive EPC Gen 2 Class 1 RFID Systems," *IEEE Transactions on Communications*, Vol. 61, No. 4, pp. 1337-1348, 2013.

[6] Feng Zhou, Chunhong Chen, Dawei Jin, Chenling Huang, and Hao Min, "Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems," *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, pp. 357-362, 2004.

[7] He Yan, Hu jianyun, Li Qiang and Min Hao, "Design of low-power baseband-processor for RFID tag," *International Symposium on Applications and the Internet Workshops (SAINTW'06)*, pp. 4 pp.-63, 2006.

[8] Mustapha Djeddou, Rafik Khelladi and Mustapha Benssalah “Improved RFID anti-collision algorithm” *International Journal of Electronics and Communications*, vol 67, No. 3, pp. 256-262, 2013

[9] Y. Hung, M. Kuo, C. Tung, and S. Shieh, "High-Speed CMOS Chip Design for Manchester and Miller Encoder," 5th*International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Kyoto, pp. 538-541, 2009.

[10] Y. Yongkang, C. Chunsheng, Z. Tuanfeng, L. Xiang and H. Liping, "Improvement on RFID-based Binary Anti-collision Algorithm," *2012 International Conference on Computer Science and Service System*, Nanjing, pp. 515-518, 2012.

[11] Identification cards - Contactless integrated circuit(s) cards – Proximity cards - Part 1: Physical characteristics, ISO/IEC Std. FCD 14443-1, 2007.

[12] Identification cards - Contactless integrated circuit(s) cards – Proximity cards - Part 2: Radio frequency power and signal interface, ISO/IEC Std. FDIS 14443-2, 2009.

[13] Identification cards - Contactless integrated circuit(s) cards – Proximity cards - Part 3: Initialization and anticollision, ISO/IEC Std. FCD 14443-3, 2008.

[14] International technology roadmap for semiconductors (ITRS): Executive Summary, 2015.

[15] J. G. Koller and W. C. Athas, "Adiabatic Switching, Low Energy Computing, and the Physics of Storing and Erasing Information", *Proc. of the 2nd Workshop on Physics and Computation*, pp. 267-270, 1992.

[16] R. T. Hinman and M. F. Schlecht, "Recovered Energy Logic – A Highly Efficient Alternative to Today's Logic Circuits", *24th Annual IEEE Power Electronics Specialists Conf.*, pp. 17-26, 1993.

[17] W. C. Athas, N. Tzartzanis, L. J. Svensson, L. Peterson, H. Li, P. Wang, and W.-C. Liu, "AC-1: a clock-powered microprocessor," *Proc. of the International Symposium on Low Power Electronics and Design*, pp. 328-333, 1997.

[18] Joonho Lim, Kipaek Kwon, and Soo-Ik Chae, "Reversible energy recovery logic circuit without non-adiabatic energy loss," *Electronics Letters*, vol. 34, no. 4, pp. 344-346, 1998.

[19] L. Varga, F. Kovács and G. Hosszú, "An Efficient Adiabatic Charge-Recovery Logic," *Proc. of the IEEE South East Conf.*, pp. 17-20, 2001.

[20] Hu Jianping, Cen Lizhang and Liu Xiao, “A new type of low-power adiabatic circuit with complementary pass-transistor logic,” *5th International Conf. on ASIC*, pp. 1235-1238, 2003.

[21] A. Blotti, and R. Saletti, “Ultralow-power adiabatic circuit semi-custom design”, *IEEE Trans. on VLSI Systems*, vol. 12, no. 11, pp. 1248-1253, 2004.

[22] F. Liu and K. T. Lau, “Improved structure for efficient charge recovery logic”, *Electronics Letters*, vol. 34, no. 18, pp. 1731-1732, 1998.

[23] Saed G. Younis, “Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic,” MIT, Ph.D. Thesis, 1994.

[24] W.C. Athas, L. Svensson, J.G. Koller, N.Tzartzanis and E.Y.Chou, “Low-power Digital Systems Based on Adiabatic-switching Principles,” *IEEE Trans. on VLSI Systems*, vol. 2, no. 4, pp. 398-407, 1994.

[25] J. S. Denker, “A review of adiabatic computing,” *IEEE Symposium on Low Power Electronics – Digest of Technical Papers*, pp. 94-97, 1994.

[26] M. Vollmer, and J. Götze, "An adiabatic architecture for linear signal processing", *Advances in Radio Science*, pp. 325–329, 2005.

[27] L. Varga, G. Hosszu and F. Kovacs, "Two-level Pipeline Scheduling of Adiabatic Logic," *29th International Seminar on Electronics Technology*, pp. 390-394, 2006.

[28] David John WILLINGHAM, “Asynchrobatic logic for low-power VLSI design,” Ph.D. Thesis, 2010.

[29] Sachin Maheshwari, V. A. Bartlett and Izzet Kale, "4-phase resettable quasi-adiabatic flip-flops and sequential circuit design," 12thConference on Ph.D. Research in Microelectronics and Electronics (PRIME), Lisbon, Portugal, pp. 1-4, 2016.

[30] Sachin Maheshwari, V. A. Bartlett and Izzet Kale, "Adiabatic flip-flops and sequential circuit design using novel resettable adiabatic buffers," 23rdEuropean Conference on Circuit Theory and Design (ECCTD), Catania, Italy, pp. 1-4, 2017.

[31] Sachin Maheshwari, V.A.Bartlett and Izzet Kale “Energy Efficient Implementation of Multi-phase Quasi-Adiabatic Cyclic Redundancy Check in near field communication” Integration, The VLSI Journal, Elsevier, vol. 62, pp. 341-352, 2018.

[32] Sachin Maheshwari, V.A.Bartlett and Izzet Kale "VHDL-based Modelling Approach for the Digital Simulation of 4-phase Adiabatic Logic Design" 28thInternational Symposium on Power and Timing Modelling, Optimization and Simulation (PATMOS), Costa Brava, Spain, pp. 111-117,2018. (Amongst top 10 papers)

[33] Sachin Maheshwari, V.A.Bartlett and Izzet Kale "Modelling, Simulation, and Verification of 4-phase Adiabatic Logic Design: A VHDL-Based Approach" Integration, The VLSI Journal, Elsevier. (Available online) doi.org/10.1016/j.vlsi.2019.01.007

[34] Sachin Maheshwari, V.A.Bartlett and Izzet Kale "VHDL-based Modelling Approach for Functional Simulation and Verification of Adiabatic Circuits" IEEE Trans. on Circuits and Systems-I. (Revision Submitted)

[35] Sachin Maheshwari and Izzet Kale "Adiabatic Implementation of Manchester Encoding for Passive NFC System" Design, Automation, and Test in Europe (DATE'19), Florence, Italy, 25th – 29th March, 2019.

[36] Haiyan Ni and Jianping Hu, "Near-threshold Sequential Circuits using Improved Clocked Adiabatic Logic in 45nm CMOS Processes", *54th Midwest Symposium on Circuits and Systems*, pp. 1-4, 2011.

[37] Zheming Xin, Jianping Hu, and Qi Chen, "Adiabatic Two-Phase CPAL Flip-Flops Operating on Near-Threshold and Super-Threshold Regions", *Procedia Environmental Science, Elsevier*, vol. 11, pp. 339-345, 2011.

[38] Ph. Teichmann, "Adiabatic logic: Future Trend and System Level Perspective", Springer Series in Advanced Microelectronics, 34, 2012.

[39] Ph. Teichmann, M. Vollmer, J. Fischer, B. Heyne, J. Götze, D. Schmitt-Landsiedel, "Saving potentials of Adiab. Logic on system level: A CORDIC-based adiabatic DCT" *12th International Symposium on Integrated Circuits*, pp. 105-108, 2009.

[40] M. Chanda, S. Jain, S. De, and C. K. Sarkar, "Implementation of Subthreshold Adiabatic Logic for Ultralow-Power Application," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 12, pp. 2782-2790, Dec. 2015.

[41] Himadri Singh Raghav and Izzet Kale, “A Balanced Power Analysis Attack Resilient Adiabatic Logic using Single Charge Sharing Transistor”, Integration, The VLSI Journal, Elsevier, 2019. https://doi.org/10.1016/j.vlsi.2018.07.010.

[42] Himadri Singh Raghav, Viv A. Bartlett, and Izzet Kale, “Investigating the Effectiveness of Without Charge-Sharing Quasi-Adiabatic Logic for Energy Efficient and Secure Cryptographic Implementations”, Microelectronics Journal, Elsevier, vol. 76, 2018, pp. 8-21.

[43] S. D. Kumar, H. Thapliyal, A. Mohammad, and S. K. Perumalla, “Design exploration of a symmetric pass gate adiabatic logic for energy-efficient and secure hardware,” Integration, the VLSI Journal, vol. 58, pp. 369-377, 2016.

[44] C. Monteiro, Y. Takahashi, and T. Sekine, “DPA Resistance of charge sharing symmetric adiabatic logic,” in Proc. of IEEE ISCAS’13, pp. 2581–2584, 2013.

[45] B. B. P. and V. S. K. Bhaaskaran, "Positive Feedback Symmetric Adiabatic Logic Against Differential Power Attack," 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), Pune, pp. 149-154, 2018.

[46] G. Pillonnet, H. Fanet, and S. Houri, "Adiabatic capacitive logic: A paradigm for low-power logic," IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-4, 2017.

[47] A. Galisultanov, Y. Perrin, H. Fanet, L. Hutin and G. Pillonnet, "Compact MEMS modeling to design full adder in Capacitive Adiabatic Logic," *48th European Solid-State Device Research Conference (ESSDERC)*, Dresden, pp. 174-177, 2018.

[48] Athas, W. C., Koller, J. G., and Svensson, L, “An Energy-Efficient CMOS Line Driver Using Adiabatic Switching," *Proc. of IEEE Great Lakes Symposium on VLSI*, 1994.

[49] L. Svensson, J. Koller, “Driving a capacitive load without discharging fCV^2”, *Proc. of the IEEE Symposium on Low-Power Electronics*, pp. 100-103, 1994.

[50] Himadri Singh Raghav, V.A.Bartlett and Izzet Kale “Investigation of stepwise charging circuits for a power-clock generation in Adiabatic Logic”, *12th International Conf. on Ph.D. Research in Microelectronics and Electronics*, pp.1-4, 2016.

[51] Himadri Singh Raghav, V.A.Bartlett and Izzet Kale “Energy efficiency of 2-step charging power-clock for adiabatic logic”, *26th International Workshop on Power and Timing Modelling, Optimisation and Simulation*, pp. 176-182, 2016.

[52] Nicolas Jeanniot, Aida Todri-Sanial, Pascal Nouet, Gaël Pillonnet, Hervé Fanet, “Investigation of the power-clock network impact on adiabatic logic”, *IEEE 20th Workshop on Signal and Power Integrity*, pp. 176-182, 2016.

[53] A. Khorami and M. Sharifkhani, "An Efficient Fast Switching Procedure for Stepwise Capacitor Chargers," *IEEE Transactions on VLSI Systems*, vol. 25, no. 2, pp. 705-713, 2017.

[54] Himadri Singh Raghav, “Adiabatic Circuits for Power-Constrained Cryptographic Computations,” Ph.D. Thesis, 2018.

[55] A. Vetuli, S. D. Pascoli, and L. M. Reyneri, “Positive feedback in adiabatic logic”, *Electronics Letters*, vol. 32, no. 20, pp. 1867-1869, 1996.

[56] A. Kramer, J. S. Denker, B. Flower, and J. Moroney, “2nd Order adiabatic computation with 2n-2p and 2n-2n2p logic circuits,” *Proc. of IEEE Symposium Low Power Design*, pp. 191-196, 1995.

[57] Y. Moon, D.K. Jeong, “An efficient charge recovery logic circuit”, *IEEE Journal of Solid-State Circuits*, vol. 31, no. 4, pp. 514-522, 1995.

[58] D. Maksimoviü and V. G. Oklobdžija, “Integrated Power Clock Generators for Low Energy Logic”, *Annual IEEE Power Electronics Specialists Conf.*, pp. 61-67, 1995.

[59] V. K. De and J. D. Meindl, “Complementary Adiabatic and Fully Adiabatic MOS Logic Families for Gigascale Integration” *43rd IEEE International Solid-State Circuits Conf. – Digest of Technical Papers*, pp. 300-301, 1996.

[60] V. K. De and J. D. Meindl, “A dynamic energy recycling logic family for ultra-low-power giga scale integration (GSI), *Proc. of the International Symposium on Low Power Electronics and Design*, pp. 371-375, 1996.

[61] C. C. Yeh, J. H. Lou, and J. B. Kuo, “1.5V CMOS full-swing energy-efficient logic (EEL) circuit suitable for low-voltage and low-power VLSI applications”, *Electronics Letters*, vol. 33, no. 16, pp. 1375-1376, 1997.

[62] V. G. Oklobd˘zija, D. Maksimovi´c, and F. Lin "Pass-Transistor Adiabatic Logic Using Single Power-Clock Supply" *IEEE Trans. on Circuits and Systems—II*, vol. 44, no. 10, pp. 842-846, 1997.

[63] K. Kioi, H. Kotaki, "Forward body-bias MOS (FBMOS) dual rail logic using an adiabatic charging technique with sub -0.6V operation", *Electronics Letters*, vol. 33, no. 14, pp. 1200-1201, 1997.

[64] F. Liu and K. T. Lau, "Pass-transistor adiabatic logic with NMOS pull-down configuration", *Electronics Letters*, vol. 34, no. 8, pp. 739-741, 1998.

[65] Seung-Moon Yoo and Sung-Mo (Steve) Kang, "A Bootstrapped NMOS Charge Recovery Logic", *Proc. of the Great Lakes Symposium on VLSI*, pp. 30-33, 1998.

[66] S. Kim and M. C. Papaefthymiou, "True single-phase energy recovering logic for low-power, high-speed VLSI", *Proc. of the International Symposium on Low Power Electronics and Design*, pp. 167-172, 1998.

[67] S. Kim and M. C. Papaefthymiou, "Single-phase Source-Coupled Adiabatic Logic", *Proc. of the International Symposium on Low Power Electronics and Design*, pp. 97-99, 1999.

[68] C. Kim, S. M. Yoo, and S. M. Kang, "Low-power adiabatic computing with NMOS energy recovery logic", *Electronics Letters*, vol. 36, no. 16, pp. 1349-1350, 2000.

[69] D. Suvakovic and C. Salama, "Two-phase non-overlapping clock adiabatic differential cascode voltage switch logic", *IEEE International Solid-State Circuits Conference – Digest of Technical Papers*, pp. 364-365, 2000.

[70] Dai Hongyu, Zhou Runde, and Ge Yuanqing, "High Efficient energy recovery logic circuit for adiabatic computing", *Proc. of the 4th International Conf. on ASIC*, pp. 858-861, 2001.

[71] L. Varga, F. Kovacs, and G. Hosszu, "An Improved Pass-Gate Adiabatic Logic", *Proc. of 14th Annual International ASIC/SOC Conf.*, pp. 208-211, 2001.

[72] R. C. Chang, P.-C. Hung and I.-H. Wang, "Complementary pass-transistor energy recovery logic for low-power applications", *IEEE Proc. on Computers and Digital Techniques*, vol. 149, no. 4, pp. 146- 151, 2002.

[73] J. Fischer, E. Amirante, A. Bargagli-Stoffi, and D. Schmitt- Landsiedel, “Improving the positive feedback adiabatic logic family”, *Advances in Radio Science*, vol. 2, pp. 221–225, 2004.

[74] Jianping Hu, Xien Ye, Yinshui Xia, “A new dual transmission gate adiabatic logic and design of an 8×8-bit multiplier for low-power DSP," *Proc. of 7th International Conf. on ICSP*, pp.559-562, 2004.

[75] Wang Peng-jun, Yu Jun-jun and Xu Jian “Design of Clocked Transmission Gate Adiabatic Logic Circuit Based on the 3ECEAC”, *IEEE Asia Pacific Conf. on Circuits and Systems*, pp.431-434, 2006.

[76] Yangbo Wu, Huiying Dong, Yi Wang, and Jianping Hu, "Low-power adiabatic sequential circuits using two-phase power-clock supply," *International Conf. on ASIC*, pp. 185-188, 2005.

[77] G. Yemiscioglu, & P. Lee, “Very-large-scale integration implementation of a 16-bit clocked adiabatic logic logarithmic signal processor”, *IET Computers & Digital Techniques*, vol. 9, pp. 239–247, 2015.

[78] D. Maksimovic´, V. G. Oklobdžija, B. Nikolic´, and K. W. Current “Clocked CMOS Adiabatic Logic with Integrated Single-Phase Power-Clock Supply”, *IEEE Transactions on VLSI Systems*, vol. 8, no. 4, pp.460-463, 2000.

[79] Haiyan Ni and Jianping Hu, “Single-Phase Adiabatic Flip-Flops and Sequential Circuits with Power-Gating Scheme”, *IEEE 8th International Conf. on ASICON*, pp. 1-4, 2009.

[80] Weiqiang Zhang, Dong Zhou, Xuanyan Hu and Jianping Hu “The Implementations of Adiabatic Flip-Flops and Sequential Circuits with Power-Gating Schemes”, *51st Midwest Symposium on Circuits and Systems*, pp. 767-770, 2008.

[81] J. M. Rabaey, A. Chandrakasan and B. Nikolic, Digital Integrated Circuits: A Design Perspective, Second Edition, Prentice Hall. 2003.

[82] M.C. Knapp, P. J. Kindlmann, M. C. Papaefthymiou, “Design and Evaluating of Adiabatic Arithmetic Units”, *Analog Integrated Circuits and Signal Processing*, vol. 14, pp. 71-79, 1997.

[83] Ph. Teichmann, J. Fischer, F. Chouard, and D. Schmitt-Landsiedel "Design issues of arithmetic structures in adiabatic logic", *Advances in Radio Science*, vol. 5, pp. 291-295, 2007.

[84] K. W. Ng and K.T. Lau, "Improved PAL-2N Logic with Complementary Pass-Transistor Logic Evaluation Tree", *Microelectronics Journal, Elsevier*, vol. 31, pp. 55-59, 2000.

[85] E. Amirante, A. Bargagli-Stoffi, J. Fischer, G. Iannaccone, and D. Schmitt-Landsiedel, "Variations of the Power Dissipation in Adiabatic Logic Gates," *11th International Workshop on Power and Timing Modelling, Optimisation and Simulation*, pp. 9.1.1–9.1.10, 2001.

[86] J. Pang, K. Andrews, and Leigh Torgerson, Clarification for CCSDS CRC-16 Computation Algorithm, Section 332M-Communication Networks, 2006.

[87] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," *International Conf. Dependable Systems and Networks (DSN*), pp. 145–154, 2004.

[88] T. V. Ramabadran and S.S. Gaitonde, "A Tutorial on CRC Computation," *IEEE Micro*, vol. 8, no. 4, pp. 62-75, 1988.

[89] G. Albertengo and R. Sisto, "Parallel CRC Generation," *IEEE Micro*, vol. 10, no. 5, pp. 63–71, 1990.

[90] G. Campobello, G. Patane, and M. Russo, "Parallel CRC Realization," *IEEE Trans. on Computers*, vol. 52, no. 1, pp. 1312–1319, 2003.

[91] C. Kennedy and Arash Reyhani-Masoleh, "High-Speed Parallel CRC Circuits" *42nd Asilomar Conf. on Signals Systems and Computers*, pp. 1823-1829, 2008.

[92] C. E. Kennedy and Mehran Mozaffari-Kermani, "Generalized Parallel CRC Computation on FPGA" *28th Canadian Conf. on Electrical and Computer Engineering*, pp107-113, 2015.

[93] M. Walma, "Pipelined Cyclic Redundancy Check (CRC) Calculation," *16th IEEE International Conf. on Computer Communications and Networks*, pp. 365–370, 2007.

[94] Standard ECMA-340 - Near Field Communication-Interface and Protocol (NFCIP-1), 3rd edition, 2013.

[95] ISO/IEC 18092 Information Technology-Telecommunications and information exchange between systems-Near Field Communication-Interface and Protocol

[96] NFC forum. Available: http://members.nfc-forum.org/specs/spec_list/

[97] Telecommunication Standardization Sector of ITU-T V.41- Code-Independent Error Control system, 1993.

[98] W. W. Peterson, and D. T. Brown, "Cyclic Codes for Error Detection", *Proc. of the IRE*, vol. 49, no.1, pp. 228-235, 1961.

[99] J. Fischer, E. Amirante, F. Randazzo, G. Iannaccone, and D. Schmitt-Landsiedel, "Reduction of the energy consumption in adiabatic gates by optimal transistor sizing", *International Workshop on Power and Timing Modelling, Optimisation and Simulation*, pp. 309-318, 2003.

[100] J. Fischer, E. Amirante, A. Bargagli-Stoffi and D. Schmitt-Landsiedel, "Adiabatic circuits: converter for static CMOS signals", *Advances in Radio Science*, pp. 274–251, 2003.

[101] S. Nakata, R. Honda, H. Makino, H. Morimura and Y. Matsuda, "Energy dissipation reduction during adiabatic charging and discharging with controlled inductor current," *IEEE 55th International Midwest Symposium on Circuits and Systems*, pp. 1068-1071, 2012.

[102] S. Morris and A. Lefley, "A 90nm CMOS 13.56MHz NFC transceiver," *2009 IEEE Asian Solid-State Circuits Conference*, Taipei, 2009, pp. 25-28.

[103] Ziqiang Yu and Xianqiang Liu, "Improvement of Dynamic Binary Search Algorithm used in an RFID system," *Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference*, Harbin, pp. 1046-1049, 2011.

[104] D. K. Klair, K. Chin and R. Raad, "A Survey and Tutorial of RFID Anti-Collision Protocols," in *IEEE Communications Surveys & Tutorials*, vol. 12, no. 3, pp. 400-421, Third Quarter 2010.

[105] D. K. Klair, K.-W. Chin, and R. Raad, "An investigation into the energy efficiency of pure and slotted aloha based RFID anti-collision protocols," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM'07)*, pp. 1–4, 2007.

[106] L. Qiu, Z. Huang, S. Zhang and W. Wang, "Location-aware anti-collision protocol for energy efficient passive RFID system," *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 255-260, 2014.

[107] V. Namboodiri and L. Gao, "Energy-Aware Tag Anticollision Protocols for RFID Systems," *IEEE Transactions on Mobile Computing*, Vol. 9, No. 1, pp. 44-59, 2010.

[108] H. Landaluce, A. Perallos, E. Onieva, L. Arjona and L. Bengtsson, "An Energy and Identification Time Decreasing Procedure for Memoryless RFID Tag Anticollision Protocols," *IEEE Transactions on Wireless Communications*, Vol. 15, No. 6, pp. 4234-4247, 2016.

[109] Feng Zhou, Dawei Jin, Chenling Huang, Min Hao, "Optimize the power consumption of passive electronic tags for anti-collision schemes," *5th International Conference on ASIC*, pp. 1213-1217, 2003.

# Appendix A: C Code for Cyclic Redundancy Check (CRC)

This appendix gives the C code for 16-bit CRC described in Chapter 4 of this thesis.

## A.16-bit CRC Algorithm for NFC application

```
CRC Algorithm (msgbit, NM, Load_value, Reset, Gen_pol, crcout)
if (Reset = '1' and NM = '1') then
   count = "0000";
   crctemp[15 : 0] = Load_value [15 : 0];
   L='0';
Else
  For (i=0; i<N; i++)
    crctemp(0)n+1 = [L. Load_value(0) + (L'(msgbit[N-1-⊕
i]crctemp(N-1)n))]Reset'
    For (j=1; j< N; j++)
     crctemp(j)n+1 ={[L. Load_value(j)+L'.crctemp(j-    ⊕
     1)n][Gen_pol(j). (msgbit(N-1⊕i)crctemp(N-1)n)]} Reset'
  End for
      count = count + '1'
    End for
```

```
End if
  if (count="FFFF") then
      crcout = crc_temp;
      rc_temp = Load_value (15 downto 0);
      L= '1'
      count = "0000"
  Else
      crcout="0000000000000000";
  End if
```

# Appendix B: VHDL Code for Adiabatic Logic Technique

This appendix gives the VHDL modelling of 4-phase adiabatic logic technique that is described in Chapter 5 of this thesis.

## B1. 4-Phase Power-Clock Generation

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY FOUR_PHASE_PCLK IS
port (CLK: in std_logic_vector (1 downto 0); PC1, PC2, PC3,
PC4:out std_logic);
END ENTITY FOUR_PHASE_PCLK;

Architecture behavioural of FOUR_PHASE_PCLK is

BEGIN
Process (CLK) is
BEGIN
if CLK ="00" then
    PC1<='0';
    PC2<='X';
    PC3<='1';
    PC4<='X';
elsif CLK = "01" then
    PC1<='X';
    PC2<='0';
    PC3<='X';
    PC4<='1';
elsif CLK = "10" then
    PC1<='1';
    PC2<='X';
    PC3<='0';
    PC4<='X';
```

```
elsif CLK = "11" then
    PC1<='X';
    PC2<='1';
    PC3<='X';
    PC4<='0';
End if;
End process;
End Architecture behavioural;
```

---

## B2. Adiabatic Logic Gates

**----------Single-rail Resettable Buffer**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE work.Adiabatic_signal.all;

ENTITY Proposed_Reset_Buf IS
port (A_H, A_L, PC, RST : in std_logic; Out_H, Out_L : out
std_logic);
END ENTITY Proposed_Reset_Buf;

Architecture behavioural of Proposed_Reset_Buf is
BEGIN

Process (RST, PC, A_H, A_L) is

BEGIN
if RST = '0' then                // RESET condition //
        Out_H <= '0';
        Out_L <= PC;
else
                              // IDLE PERIOD //
if PC='0'  then
        Out_H <= '0';
        Out_L <= '0';
elsif PC='0' and EVALUATE_edge (A_H) and EVALUATE_edge(A_L) then
        Out_H <='Z';
        Out_L <='Z';
                              // EVALUATE PERIOD //
elsif PC='X' and HOLD_edge (A_H) and HOLD_edge(A_L) then
        Out_H <='Z';
        Out_L <='Z';
elsif PC='X' and HOLD_edge (A_H) then
        Out_H <= PC;
        Out_L <='0';
elsif PC='X' and HOLD_edge (A_L) then
        Out_H <= '0';
        Out_L <= PC;
```

```
                        // HOLD PERIOD //
elsif  PC='1'  and  RECOVERY_edge  (A_H)  and  RECOVERY_edge  (A_L)
then
        Out_H <='Z';
        Out_L <='Z';
elsif PC='1' and RECOVERY_edge (A_H) then
        Out_H <= PC;
        Out_L <='0';
elsif PC='1' and RECOVERY_edge (A_L) then
        Out_H <= '0';
        Out_L <= PC;
                        // RECOVERY PERIOD //
elsif PC='X' and IDLE_edge (A_H) and IDLE_edge (A_L) then
        Out_H <='Z';
        Out_L <='Z';
elsif PC='X' and IDLE_edge (A_H) then
        Out_H <= PC;
        Out_L <='0';
elsif PC='X' and IDLE_edge (A_L) then
        Out_H <= '0';
        Out_L <= PC;
    End if;
  End if;
End Process;
End Architecture behavioural;
```

**----------Dual-rail Resettable Buffer**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE work. Adiabatic_2INP_GATES.all;   // User Defined Adiabatic
Logic Gates //

ENTITY Proposed_ Proposed_dual_reset_buf IS
port (A_H, Res_H, A_L, Res_L, PC : in std_logic; Out_H, Out_L :
out std_logic);
END ENTITY Proposed_ Proposed_dual_reset_buf;

Architecture behavioural of Proposed_dual_reset_buf is

Component Proposed_Buf
port (A_H, A_L : in std_logic; PC : in std_logic; Out_H, Out_L :
out std_logic);
End Component;

SIGNAL Z_H, Z_L: std_logic;

BEGIN

C01: Proposed_Buf port map (Z_H, Z_L, PC, Out_H, Out_L);

     Z_H <= Aand (A_H,Res_H);
     Z_L <=Aor (A_L,Res_L);
```

```
End Architecture behavioural;
```

**----------2-input AND/NAND**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE work. Adiabatic_2INP_GATES.all;   // User Defined Adiabatic
Logic Gates //

ENTITY Proposed_ANDNAND2 IS
port (A_H, B_H, A_L, B_L, PC: in std_logic; AND2, NAND2: out
std_logic);
END ENTITY Proposed_ANDNAND2;

Architecture behavioural of Proposed_ANDNAND2 is

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

SIGNAL Z_H, Z_L: std_logic;

BEGIN

C01: Proposed_Buf port map (Z_H, Z_L, PC, AND2, NAND2);

     Z_H <= Aand (A_H, B_H);
     Z_L <= Aor (A_L, B_L);

End Architecture behavioural;
```

**----------2-input OR/NOR**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE  work.Adiabatic_2INP_GATES.all;  //User  Defined  Adiabatic
Logic Gates //

ENTITY Proposed_ORNOR2 IS
port (A_H, B_H, A_L, B_L, PC: in std_logic; OR2, NOR2: out
std_logic);
END ENTITY Proposed_ORNOR2;

Architecture behavioural of Proposed_ORNOR2 is

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

SIGNAL Z_H, Z_L: std_logic;
```

```
BEGIN

C01: Proposed_Buf port map (Z_H, Z_L, PC, OR2, NOR2);

     Z_H <= Aor (A_H, B_H);
     Z_L <= Aand (A_L, B_L);

End Architecture behavioural;
```

**----------3-input AND/NAND**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE  work.Adiabatic_2INP_GATES.all;  //User Defined Adiabatic
Logic Gates //

ENTITY Proposed_ANDNAND3 IS
port (A_H, B_H, C_H, A_L, B_L, C_L, PC: in std_logic; AND3,
NAND3: out std_logic);
END ENTITY Proposed_ANDNAND3;

Architecture behavioural of Proposed_ANDNAND3 is

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

     SIGNAL Z_H, Z_L: std_logic;

BEGIN

C01: Proposed_Buf port map (Z_H, Z_L, PC, AND3, NAND3);

     Z_H <= Aand (Aand (A_H, B_H), C_H);
     Zb<=Aor (Aor (A_L, B_L), C_L);

End Architecture behavioural;
```

**----------3-input OR/NOR**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE  work.Adiabatic_2INP_GATES.all;  //User Defined Adiabatic
Logic Gates //

ENTITY Proposed_ORNOR3 IS
port (A_H, B_H, C_H, A_L, B_L, C_L, PC: in std_logic; OR3, NOR3:
out std_logic);
END ENTITY Proposed_ORNOR3;
```

```
Architecture behavioural of Proposed_ORNOR3 is

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

      SIGNAL Z_H, Z_L: std_logic;

BEGIN

C01: Proposed_Buf port map (Z_H, Z_L, PC, OR3, NOR3);

      Z_H <=Aor (Aor (A_H, B_H), C_H);
      Z_L <=Aand (Aand (A_L, B_L), C_L);

End Architecture behavioural;
```

**----------2-input XOR/XNOR**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE  work.Adiabatic_2INP_GATES.all;  //User  Defined  Adiabatic
Logic Gates //

ENTITY Proposed_XORXNOR2 IS
port (A_H, B_H, A_L, B_L, PC: in std_logic; XOR2, XNOR2: out
std_logic);
END ENTITY Proposed_XORXNOR2;

Architecture behavioural of Proposed_XORXNOR2 is

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

   Signal Z_H, Z_L: std_logic;
Begin

C01: Proposed_Buf port map (Z_H, Z_L, PC, XOR2, XNOR2);

      Z <= Aor (Aand (A_L, B_H), Aand (A_H, B_L));
      Zb <= Aor (Aand (A_L, B_L), Aand (A_H, B_H));

End Architecture behavioural;
```

---

## B3. 2-bit Adiabatic Ring-counter

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY Ring_Counter_2bit is
port (CLK: in std_logic_vector (1 downto 0); RES: in std_logic;
Q0_H, Q0_L, Q1_H, Q1_L: inout std_logic);
END ENTITY Ring_Counter_2bit;

Architecture Structural of Ring_Counter_2bit IS

Component FOUR_PHASE_PC
port (CLK: in std_logic_vector (1 downto 0); PC1, PC2, PC3, PC4:
out std_logic);
End Component;

Component Proposed_Reset_Buf
port (A_H, A_L, PC: in std_logic; RST: in std_logic; Out_H,
Out_L: out std_logic);
End Component;

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

SIGNAL PC1, PC2, PC3, PC4: std_logic;
SIGNAL Q01_H, Q01_L, Q02_H, Q02_L, Q03_H, Q03_L, Q11_H, Q11_L,
Q12_H, Q12_L, Q13_H, Q13_L: std_logic;

BEGIN

CLK1: FOUR_PHASE_PCLK port map (CLK, PC1, PC2, PC3, PC4);
n0: Proposed_Reset_Buf port map (Q1_L, Q1_H, PC1, RES, Q01_H,
Q01_L);
n1: Proposed_Buf port map (Q01_H, Q01_L, PC2, Q02_H, Q02_L);
n2: Proposed_Buf port map (Q02_H, Q02_L, PC3, Q03_H, Q03_L);
n3: Proposed_Buf port map (Q03_H, Q03_L, PC4, Q0_H, Q0_L);
n4: Proposed_Reset_Buf port map (Q0_H, Q0_L, PC1, RES, Q11_H,
Q11_L);
n5: Proposed_Buf port map (Q11_H, Q11_L, PC2, Q12_H, Q12_L);
n6: Proposed_Buf port map (Q12_H, Q12_L, PC3, Q13_H, Q13_L);
n7: Proposed_Buf port map (Q13_H, Q13_L, PC4, Q1_H, Q1_L);

END Architecture structural;
```

---

## B4. 3-bit Up-Down counter

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY Proposed_UP_DOWN_Counter3 is
port (CLK: in std_logic_vector(1 downto 0); RST_H, RST_L, UD_H,
UD_L: in std_logic; Q0_H, Q0_L, Q1_H, Q1_L, Q2_H,Q2_L: inout
std_logic);
END ENTITY Proposed_UP_DOWN_Counter3;

Architecture structural of Proposed_UP_DOWN_Counter3 is

Component DC_TO_ADIABATIC
port (INP_H, INP_L: in std_logic; CLK: in std_logic_vector(1
downto 0); A_H, A_L: out std_logic);
End Component;

Component FOUR_PHASE_PC
port (CLK: in std_logic_vector(1 downto 0); PC1, PC2, PC3, PC4:
out std_logic);
End Component;

Component Proposed_dual_reset_buf
port (A_H, A_L, RES_H, RES_L, PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

Component Proposed_ANDNAND2
port (A_H, B_H, A_L, B_L, PC: in std_logic; AND2, NAND2: out
std_logic);
End Component Proposed_ANDNAND2;

Component Proposed_XORXNOR2
port (A_H, B_H, A_L, B_L, PC: in std_logic; XOR2, XNOR2: out
std_logic);
End Component Proposed_XORXNOR2;
```

```
Component Proposed_ORNOR3
port (A_H, B_H, C_H, A_L, B_L, C_L, PC: in std_logic; OR3, NOR3:
out std_logic);
End Component Proposed_ORNOR3;

SIGNAL RES_H, RES_L: std_logic;
SIGNAL PC1, PC2, PC3, PC4, CU, CD: std_logic;
SIGNAL Q01_H, Q01_L, Q02_H, Q02_L, Q03_H, Q03_L: std_logic;
SIGNAL Q110_H, Q110_L, Q111_H, Q111_L, Q112_H, Q112_L, Q113_H,
Q113_L, Q11_H, Q11_L, Q12_H, Q12_L, Q13_H, Q13_L: std_logic;
SIGNAL Q210_H,Q210_L, Q211_H,Q211_L, Q212_H,Q212_L, Q213_H,
Q213_L, Q220_H, Q220_L, Q221_H, Q221_L, Q222_H, Q222_L, Q223_H,
Q223_L, Q21_H, Q21_L, Q22_H, Q22_L, Q23_H, Q23_L: std_logic;

BEGIN

CLK1: FOUR_PHASE_PCLK port map (CLK, PC1, PC2, PC3, PC4);
INPUT1: DC_TO_ADIABATIC port map (UD_H, UD_L, CLK, CD, CU);
INPUT2: DC_TO_ADIABATIC port map (RST_H, RST_L, CLK, RES_H,
RES_L);

C01: Proposed_dual_reset_buf port map (Q0_L, Q0_H, RES_H, RES_L,
PC1, Q01_H, Q01_L);
C02: Proposed_Buf port map (Q01_H, Q01_L, PC2, Q02_H, Q02_L);
C03: Proposed_Buf port map (Q02_H, Q02_L, PC3, Q03_H, Q03_L);
C04: Proposed_Buf port map (Q03_H, Q03_L, PC4, Q0_H, Q0_L);
C110: Proposed_Buf port map (CD, CU, PC1, Q110_H, Q110_L);
C111: Proposed_Buf port map (Q110_H, Q110_L, PC2, Q111_H,
Q111_L);

C112: Proposed_XORXNOR2 port map (Q02_H, Q111_H, Q02_L, Q111_L,
PC3, Q112_H, Q112_L);
C113: Proposed_XORXNOR2 port map (Q112_H, Q13_H, Q112_L, Q13_L,
PC4, Q113_H, Q113_L);
C11: Proposed_dual_reset_buf port map (Q113_H, Q113_L, RES_H,
RES_L, PC1, Q11_H, Q11_L);
C12: Proposed_Buf port map (Q11_H, Q11_L, PC2, Q12_H, Q12_L);
C13: Proposed_Buf port map (Q12_H, Q12_L, PC3, Q13_H, Q13_L);
C14: Proposed_Buf port map (Q13_H, Q13_L, PC4, Q1_H, Q1_L);

C210: Proposed_XORXNOR2 port map (Q11_H, Q21_H, Q11_L, Q21_L,
PC2, Q210_H, Q210_L);
C211: Proposed_ANDNAND2 port map (Q01_H, Q110_L, Q01_L, Q110_H,
PC2, Q211_H, Q211_L);
C212: Proposed_ANDNAND2 port map (Q01_L, Q110_H, Q01_H, Q110_L,
PC2, Q212_H, Q212_L);
C213: Proposed_XORXNOR2 port map (Q110_H, Q01_H, Q110_L, Q01_L,
PC2, Q213_H, Q213_L);
```

```
C220: Proposed_ANDNAND2 port map (Q210_H, Q211_H, Q210_L, Q211_L,
PC3, Q220_H, Q220_L);
C221: Proposed_ANDNAND2 port map (Q210_L, Q212_H, Q210_H, Q212_L,
PC3, Q221_H, Q221_L);
C222: Proposed_ANDNAND2 port map (Q213_L, Q22_H, Q213_H, Q22_L,
PC3, Q222_H, Q222_L);
C223:  Proposed_ORNOR3 port map (Q220_H, Q221_H, Q222_H, Q220_L,
Q221_L, Q222_L, PC4, Q223_H, Q223_L);
C21:  Proposed_dual_reset_buf port map (Q223_H, Q223_L, RES_H,
RES_L, PC1, Q21_H, Q21_L);
C22: Proposed_Buf port map (Q21_H, Q21_L, PC2, Q22_H, Q22_L);
C23: Proposed_Buf port map (Q22_H, Q22_L, PC3, Q23_H, Q23_L);
C24: Proposed_Buf port map (Q23_H, Q23_L, PC4, Q2_H, Q2_L);

END Architecture structural;
```

---

## B5. 16-bit CRC for 16-bit message word

**----------DC pulse train to Adiabatic for PC3**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY DC_TO_ADIABATIC_PC3_INP IS
port (INP_H, INP_L: in std_logic; CLK: in std_logic_vector(1
downto 0); A_H, A_L: out std_logic);
END ENTITY DC_TO_ADIABATIC_PC3_INP;

Architecture behavioural of DC_TO_ADIABATIC_PC3_INP is

BEGIN

Process (CLK, INP_H, INP_L) is

  BEGIN

if CLK ="00" then
  if INP_H ='1' and INP_L ='0' then
           A_H <='X';
           A_L <='0';
elsif INP_L ='1' and INP_H ='0' then
           A_H <='0';
           A_L <='X';
End if;

Elsif CLK = "01" then
           A_H <='0';
```

```
            A_L <='0';

elsif CLK = "10" then
if INP_H ='1' and INP_L ='0' then
            A_H <='X';
            A_L <='0';
      elsif INP_L ='1' and INP_H='0' then
            A_H <='0';
            A_L <='X';
End if;

elsif CLK = "11" then
if INP_H ='1' and INP_L ='0' then
            A_H <='1';
            A_L <='0';
      elsif INP_L ='1' and INP_H='0' then
            A_H <='0';
            A_L <='1';
End if;

End if;
End Process;
End Architecture behavioural;
```

**----------4-bit Counter**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY Counter_4_bit is
port (PC1, PC2, PC3, PC4, R_count_H, R_count_L, NM_H, NM_L: in
std_logic; Q0_3_H, Q0_3_L, Q1_3_H, Q1_3_L, Q0_H, Q0_L, Q1_H,
Q1_L, Q2_H, Q2_L, Q3_H, Q3_L: inout std_logic);
END ENTITY Counter_4_bit;

Architecture structural of Counter_4_bit is

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

Component Proposed_ANDNAND2
port (A_H, B_H, A_L, B_L, PC: in std_logic; AND2, NAND2: out
std_logic);
End Component Proposed_ANDNAND2;

Component Proposed_ORNOR2
port (A_H, B_H, A_L, B_L, PC: in std_logic; OR2, NOR2: out
std_logic);
End Component Proposed_ORNOR2;
```

```
Component Proposed_ORNOR3
port (A_H, B_H, C_H, A_L, B_L, C_L, PC: in std_logic; OR3, NOR3:
out std_logic);
End Component Proposed_ORNOR3;

Component Proposed_ANDNAND3
port (A_H, B_H, C_H, A_L, B_L, C_L, PC: in std_logic; AND3,
NAND3: out std_logic);
End Component Proposed_ANDNAND2;

SIGNAL RST_H, RST_L: std_logic;
SIGNAL Q01_H, Q01_L, Q02_H, Q02_L, Q03_H, Q03_L, Q04_H, Q04_L,
Q05_H, Q05_L: std_logic;
SIGNAL Q11_H, Q11_L, Q12_H, Q12_L, Q13_H, Q13_L, Q14_H, Q14_L,
Q15_H, Q15_L, Q16_H, Q16_L, Q17_H, Q17_L: std_logic;
SIGNAL Q21_H, Q21_L, Q22_H, Q22_L, Q23_H, Q23_L, Q24, Q24b, Q25,
Q25_L, Q26_H, Q26_L, Q27_H, Q27_L, Q2_3_H, Q2_3_L: std_logic;
SIGNAL Q31_H, Q31_L, Q32_H, Q32_L, Q33_H, Q33_L, Q34_H, Q34_L,
Q35_H, Q35_L, Q3_3_H, Q3_3_L: std_logic;

BEGIN

C00: Proposed_ANDNAND2 port map (R_count_H, NM_H, R_count_L,
NM_L, PC4, RST_H, RST_L);
C01: Proposed_ANDNAND2 port map (RST_H, Q0_L, RST_L, Q0_H, PC1,
Q01_H, Q01_L);
C02: Proposed_ANDNAND2 port map (RST_H, Q1_H, RST_L, Q1_L, PC1,
Q02_H, Q02_L);
C03: Proposed_ANDNAND2 port map (Q2_H, Q3_H, Q2_L, Q3_L, PC1,
Q03_H, Q03_L);
C04: Proposed_Buf port map (Q01_H, Q01_L, PC2, Q04_H, Q04_L);
C05: Proposed_ANDNAND2 port map (Q02_H, Q03_H, Q02_L, Q03_L, PC2,
Q05_H, Q05_L);
C06: Proposed_ORNOR2 port map (Q04_H, Q05_H, Q04_L, Q05_L, PC3,
Q0_3_H, Q0_3_L);
C07: Proposed_Buf port map (Q0_3_H, Q0_3_L, PC4, Q0_H, Q0_L);

C10: Proposed_ANDNAND3 port map (RST_H, Q1_L, Q0_H, RST_L, Q1_H,
Q0_L, PC1, Q11_H, Q11_L);
C11: Proposed_ANDNAND2 port map (Q0_H, RST_H, Q0_L, RST_L, PC1,
Q12_H, Q12_L);
C12: Proposed_ANDNAND2 port map (Q2_H, Q3_H, Q2_L, Q3_L, PC1,
Q13_H, Q13_L);
C13: Proposed_ANDNAND3 port map (RST_H, Q1_H, Q0_L, RST_L, Q1_L,
Q0_H, PC1, Q14_H, Q14_L);
C14: Proposed_Buf port map (Q11_H, Q11_L, PC2, Q15_H, Q15_L);
C15: Proposed_ANDNAND2 port map (Q12_H, Q13_H, Q12_L, Q13_L, PC2,
Q16_H, Q16_L);
```

```
C16: Proposed_Buf port map (Q14_H, Q14_L, PC2, Q17_H, Q17_L);
C17: Proposed_ORNOR3 port map (Q15_H, Q16_H, Q17_H, Q15_L, Q16_L,
Q17_L, PC3, Q1_3_H, Q1_3_L);
C18: Proposed_Buf port map (Q1_3_H, Q1_3_L, PC4, Q1_H, Q1_L);

C20: Proposed_ANDNAND3 port map (RST_H, Q2_H, Q3_H, RST_L, Q2_L,
Q3_L, PC1, Q21_H, Q21_L);
C21: Proposed_ORNOR2 port map (Q1_L, Q0_L, Q1_H, Q0_H, PC1,
Q22_H, Q22_L);
C22: Proposed_Buf port map (RST_H, RST_L, PC1, Q23_H, Q23_L);
C23: Proposed_Buf port map (Q2_L, Q2_H, PC1, Q24_H, Q24_L);
C24: Proposed_Buf port map (Q21_H, Q21_L, PC2, Q25_H, Q25_L);
C25: Proposed_ANDNAND3 port map (Q22_H, Q23_H, Q24_L, Q22_L,
Q23_L, Q24_H, PC2, Q26_H, Q26_L);
C26: Proposed_ANDNAND3 port map (Q23_H, Q22_L, Q24_H, Q23_L,
Q22_H, Q24_L, PC2, Q27_H, Q27_L);
C27: Proposed_ORNOR3 port map (Q25_H, Q26_H, Q27_H, Q25_L, Q26_L,
Q27_L, PC3, Q2_3_H, Q2_3_L);
C28: Proposed_Buf port map (Q2_3_H, Q2_3_L, PC4, Q2_H, Q2_L);

C30: Proposed_ANDNAND2 port map (RST_H, Q3_H, RST_L, Q3_L, PC1,
Q31_H, Q31_L);
C31: Proposed_ANDNAND2 port map (RST_H, Q1_H, RST_L, Q1_L, PC1,
Q32_H, Q32_L);
C32: Proposed_ANDNAND2 port map (Q0_H, Q2_H, Q0_L, Q2_L, PC1,
Q33_H, Q33_L);
C33: Proposed_Buf port map (Q31_H, Q31_L, PC2, Q34_H, Q34_L);
C34: Proposed_ANDNAND2 port map (Q32_H, Q33_H, Q32_L, Q33_L, PC2,
Q35_H, Q35_L);
C35: Proposed_ORNOR2 port map (Q34_H, Q35_H, Q34_L, Q35_L, PC3,
Q3_3_H, Q3_3_L);
C36: Proposed_Buf port map (Q3_3_H, Q3_3_L, PC4, Q3_H, Q3_L);

END Architecture structural;
```

**----------Controller**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY Controller is
port (PC1, PC2, PC3, PC4, NM_H, NM_L, RESET_H, RESET_L: in
std_logic; Q0_3_H, Q0_3_L, Q1_3_H, Q1_3_L, Q2_H, Q2_L, Q3_H,
Q3_L, R_count_H, R_count_L, R0_H, R0_L, R1_H, R1_L, R2_H, R2_L,
R3_H, R3_L: inout std_logic; R4_H, R4_L: out std_logic);
END ENTITY Controller;

Architecture Structural of Controller is
```

```
Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

Component Proposed_ANDNAND2
port (A_H, B_H, A_L, B_L, PC: in std_logic; AND2, NAND2: out
std_logic);
End Component Proposed_ANDNAND2;

Component Proposed_ORNOR2
port (A_H, B_H, A_L, B_L, PC: in std_logic; OR2, NOR2: out
std_logic);
End Component Proposed_ORNOR2;

Component Counter_4_bit IS
port (PC1, PC2, PC3, PC4, R_count_H, R_count_L, NM_H, NM_L: in
std_logic; Q0_3_H, Q0_3_L, Q1_3_H, Q1_3_L, Q0_H, Q0_L, Q1_H,
Q1_L, Q2_H, Q2_L, Q3_H, Q3_L: inout std_logic);
End ComponentCounter_4_bit;

SIGNAL RX_H, RX1_H, RX2_H, RX3_H, RX4_H, RX_L, RX1_L, RX2_L,
RX3_L, RX4_L: std_logic;
SIGNAL Q0_H, Q0_L, Q1_H, Q1_L: std_logic;
SIGNAL CTR1_H, CTR1_L, CTR2_H, CTR2_L, CTR3_H, CTR3_L: std_logic;

BEGIN

count01: Counter_4_bit port map (PC1, PC2, PC3, PC4, R_count_L,
R_count_H ,NM_L, NM_H, Q0_3_H, Q0_3_L, Q1_3_H, Q1_3_L, Q0_H,
Q0_L, Q1_H, Q1_L, Q2_H, Q2_L, Q3_H, Q3_L);

C01: Proposed_ANDNAND2 port map (Q0_H, Q1_H, Q0_L, Q1_L, PC1,
CTR1_H, CTR1_L);
C02: Proposed_ANDNAND2 port map (Q2_H, Q3_H, Q2_L, Q3_L, PC1,
CTR2_H, CTR2_L);
C03: Proposed_ANDNAND2 port map(CTR1_H, CTR2_H, CTR1_L, CTR2_L,
PC2, CTR3_H, CTR3_L);
C21: Proposed_ORNOR2 port map(CTR3_H, RESET_H, CTR3_L, RESET_L,
PC3, R_count_H, R_count_L);

B01: Proposed_Buf port map (R_count_H, R_count_L, PC4, RX_H,
RX_L);
B02: Proposed_Buf port map (RX_H, RX_L, PC1, RX1_H, RX1_L);
B03: Proposed_Buf port map (RX1_H, RX1_L, PC2, R0_H, R0_L);
B04: Proposed_Buf port map (R0_H, R0_L, PC3, R1_H, R1_L);
B05: Proposed_Buf port map (R1_H, R1_L, PC4, R2_H, R2_L);
B06: Proposed_Buf port map (R2_H, R2_L, PC1, R3_H, R3_L);
```

```
B07: Proposed_Buf port map (R3_H, R3_L, PC2, RX2_H, RX2_L);
B08: Proposed_Buf port map (RX2_H, RX2_L, PC3, RX3_H, RX3_L);
B09: Proposed_Buf port map (RX3_H, RX3_L, PC4, RX4_H, RX4_L);
B10: Proposed_Buf port map (RX4_H, RX4_L, PC1, R4_H, R4_L);

END Architecture structural;
```

**----------2:1 Multiplexer**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE work.Adiabatic_2INP_GATES.all;//User Defined Adiabatic Logic
Gates //

ENTITY Proposed_2_1_MUX IS
port (A_H, B_H, A_L, B_L, S_H, S_L, PC: in std_logic; Out_H,
Out_L: out std_logic);
END ENTITY Proposed_2_1_MUX;

ARCHITECTURE Structural OF Proposed_2_1_MUX IS

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

SIGNAL X_H, X_L: std_logic;

BEGIN

C01: Proposed_Buf port map (X_H, X_L, PC, Out_H, Out_L);

     X_H <= Aor (Aand (S_L, A_H), Aand (S_H, B_H));
     X_L <= Aand (Aor (S_L, B_L), Aor (S_H, A_L));

END ARCHITECTURE Structural;
```

**----------4:1 Multiplexer**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE  work.Adiabatic_2INP_GATES.all;  //User  Defined  Adiabatic
Logic Gates //

ENTITY Proposed_4_1_MUX IS
port (A_H, B_H, C_H, D_H, A_L, B_L, C_L, D_L, S0_H, S0_L, S1_H,
S1_L, PC: in std_logic; Out_H, Out_L: out std_logic);
END ENTITY Proposed_4_1_MUX;
```

```
ARCHITECTURE Structural OF Proposed_4_1_MUX IS

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

SIGNAL X_H, X_L: std_logic;
BEGIN

C01: Proposed_Buf port map (X_H, X_L, PC, Out_H, Out_L);

X_H <= Aor (Aor (Aand (Aand (S0_L, S1_L), A_H), Aand (Aand (S0_H,
S1_L), B_H)), Aor (Aand (Aand (S0_L, S1_H), C_H), Aand (Aand
(S0_H, S1_H), D_H)));

X_L <= Aand (Aand (Aor (Aor(S0_H, S1_H), A_H), Aor (Aor (S0_L,
S1_H), B_L)), Aand (Aor (Aor (S0_H, S1_L), C_L), Aor (Aor (S0_L,
S1_L), D_L)));

END ARCHITECTURE Structural;
```

**----------8:1 Multiplexer**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY Proposed_8_1_MUX IS
port (A_H, B_H, C_H, D_H, E_H, F_H, G_H, H_H, A_L, B_L, C_L, D_L,
E_L, F_L, G_L, H_L, S0_H, S1_H, S2_H, S0_L, S1_L, S2_L, PC1, PC2:
in std_logic; Out_H, Out_L: out std_logic);
END ENTITY Proposed_8_1_MUX;

ARCHITECTURE Structural OF Proposed_8_1_MUX IS

Component Proposed_4_1_MUX
port (A_H, B_H, C_H, D_H, A_L, B_L, C_L, D_L, S0_H, S1_H, S0_L,
S1_L, PC: in std_logic; Out_H, Out_L: out std_logic);
End Component;

Component Proposed_2_1_MUX
port (A_H, B_H, A_L, B_L, S_H, S_L, PC: in std_logic; Out_H,
Out_L: out std_logic);
End Component;

SIGNAL T_H, T_L, U_H, U_L: std_logic;
```

```
BEGIN

MUX1: Proposed_4_1_MUX port map (A_H, B_H, C_H, D_H, A_L, B_L,
C_L, D_L, S0_H, S1_H, S0_L, S1_L, PC1, T_H, T_L);
MUX2: Proposed_4_1_MUX port map (E_H, F_H, G_H, H_H, E_L, F_L,
G_L, H_L, S0_H, S1_H, S0_L, S1_L, PC1, U_H, U_L);
MUX3: Proposed_2_1_MUX port map (T_H, U_H, T_L, U_L, S2_H, S2_L,
PC2, Out_H, Out_L);

END ARCHITECTURE Structural;
```

**----------16:1 Multiplexer**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY Proposed_16_1_MUX IS
port (A_H, B_H, C_H, D_H, E_H, F_H, G_H, H_H, I_H, J_H, K_H, L_H,
M_H, N_H, O_H, P_H, A_L, B_L, C_L, D_L, E_L, F_L, G_L, H_L, I_L,
J_L, K_L, L_L, M_L, N_L, O_L, P_L, S0_H, S1_H, S2_H, S3_H, S0_L,
S1_L, S2_L, S3_L, PC1, PC2, PC3: in std_logic; Out_H, Out_L: out
std_logic);
END ENTITY Proposed_16_1_MUX;

ARCHITECTURE Structural OF Proposed_16_1_MUX IS

Component Proposed_8_1_MUX
port (A_H, B_H, C_H, D_H, E_H, F_H, G_H, H_H, A_L, B_L, C_L, D_L,
E_L, F_L, G_L, H_L, S0_H, S1_H, S2_H, S0_L, S1_L, S2_L, PC1, PC2:
in std_logic; Out_H, Out_L: out std_logic);
End Component;

Component Proposed_2_1_MUX
port (A_H, B_H, A_L, B_L, S_H, S_L, PC: in std_logic; Out_H,
Out_L: out std_logic);
End Component;

SIGNAL X_H, V_H, W_H, X_L, V_L, W_L: std_logic;
BEGIN

MUX1: Proposed_8_1_MUX port map (A_H, B_H, C_H, D_H, E_H, F_H,
G_H, H_H, A_L, B_L, C_L, D_L, E_L, F_L, G_L, H_L, S0_H, S1_H,
S2_H, S0_L, S1_L, S2_L, PC1, PC2, V_H, V_L);
MUX2: Proposed_8_1_MUX port map (I_H, J_H, K_H, L_H, M_H, N_H,
O_H, P_H, I_L, J_L, K_L, L_L, M_L, N_L, O_L, P_L, S0_H, S1_H,
S2_H, S0_L, S1_L, S2_L, PC1, PC2, W_H, W_L);
```

```
MUX3: Proposed_2_1_MUX port map (V_H, W_H, V_L, W_L, S3_H, S3_L,
PC3, Out_H, Out_L);

END ARCHITECTURE Structural;
```

**----------Controller + MUX**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY controller_MUX IS
port (A_H, B_H, C_H, D_H, E_H, F_H, G_H, H_H, I_H, J_H, K_H, L_H,
M_H, N_H, O_H, P_H, A_L, B_L, C_L, D_L, E_L, F_L, G_L, H_L, I_L,
J_L, K_L, L_L, M_L, N_L, O_L, P_L, PC1, PC2, PC3, PC4, NM_H,
NM_L, RESET_H, RESET_L: in std_logic; R_count_H, R_count_L, R0_H,
R0_L, R1_H, R1_L, R2_H, R2_L, R3_H, R3_L: inout std_logic;
Msg_IN_L, Msg_IN_L, R4_H, R4_L: out std_logic);
END ENTITY controller_MUX;

ARCHITECTURE structural OF controller_MUX IS

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

Component Controller
port (PC1, PC2, PC3, PC4, NM_H, NM_L, RESET_H, RESET_L: in
std_logic; Q0_3_H, Q0_3_L, Q1_3_H, Q1_3_L, Q2_H, Q2_L, Q3_H,
Q3_L, R_count_H, R_count_L, R0_H, R0_L, R1_H, R1_L, R2_H, R2_L,
R3_H, R3_L: inout std_logic; R4_H, R4_L: out std_logic);
End Component Controller;

Component Proposed_16_1_MUX
port (A_H, B_H, C_H, D_H, E_H, F_H, G_H, H_H, I_H, J_H, K_H, L_H,
M_H, N_H, O_H, P_H, A_L, B_L, C_L, D_L, E_L, F_L, G_L, H_L, I_L,
J_L, K_L, L_L, M_L, N_L, O_L, P_L, S0_H, S1_H, S2_H, S3_H, S0_L,
S1_L, S2_L, S3_L, PC1, PC2, PC3: in std_logic; Out_H, Out_L: out
std_logic);
End Component Proposed_16_1_MUX;

SIGNAL Q0_3_H, Q1_3_H, Q2_H, Q3_1_H, Q0_3_L, Q1_3_L, Q2_L,
Q3_1_L, Q3_H, Q3_L: std_logic;

BEGIN

B01: Proposed_Buf port map (Q3_H, Q3_L, PC1, Q3_1_L, Q3_1_L);
```

```
controller01: Controller port map (PC1, PC2, PC3, PC4, NM_H,
NM_L, RESET_H, RESET_L, Q0_3_H, Q0_3_L, Q1_3_H, Q1_3_L, Q2_H,
Q2_L, Q3_H, Q3_L, R_count_H, R_count_L, R0_H, R0_L, R1_H, R1_L,
R2_H, R2_L, R3_H, R3_L, R4_H, R4_L);
MUX1: Proposed_16_1_MUX port map (A_H, B_H, C_H, D_H, E_H, F_H,
G_H, H_H, I_H, J_H, K_H, L_H, M_H, N_H, O_H, P_H, A_L, B_L, C_L,
D_L, E_L, F_L, G_L, H_L, I_L, J_L, K_L, L_L, M_L, N_L, O_L, P_L,
Q0_3_H, Q1_3_H, Q2_H, Q3_1_H, Q0_3_L, Q1_3_L, Q2_L, Q3_1_L, PC4,
PC1, PC2, Msg_IN_H, Msg_IN_L);

END ARCHITECTUREstructural;
```

**----------CRC_Generator_Polynomial_bit_block**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY CRC_Generator_Polynomial_bit_block IS
port (CR0_H, CR0_L, PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, L0_H, L0_L, G1_H, G1_L, CR15_xor_IN_H, CR15_XOR_IN_L,
ZERO, ONE: in std_logic; CR1_H, CR1_L: out std_logic);
END ENTITY CRC_Generator_Polynomial_bit_block;

ARCHITECTURE structural OF CRC_Generator_Polynomial_bit_block IS

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

Component Proposed_dual_reset_buf
port (A_H, Res_H, A_L, Res_L, PC : in std_logic; Out_H, Out_L :
out std_logic);
End Component;

Component Proposed_ANDNAND2
port (A_H, B_H, A_L, B_L, PC: in std_logic; AND2, NAND2: out
std_logic);
End Component Proposed_ANDNAND2;

Component Proposed_XORXNOR2
port (A_H, B_H, A_L, B_L, PC: in std_logic; XOR2, XNOR2: out
std_logic);
End Component Proposed_XORXNOR2;

Component Proposed_2_1_MUX
port (A_H, B_H, A_L, B_L, S_H, S_L, PC: in std_logic; Out_H,
Out_L: out std_logic);
```

```
End Component;

SIGNAL CD1_H, CD2_H, CD3_H, CD1_L, CD2_L, CD3_L: std_logic;
SIGNAL SEL_GP1_H, SEL_GP1_L, GP1_H, GP1_L: std_logic;
SIGNAL ZERO1, ZERO2, ZERO3, ONE1, ONE2, ONE3, Load0_H, Load0_L:
std_logic;

BEGIN

B001: Proposed_Buf port map (ZERO, ONE, PC1, ZERO1, ONE1);
B002: Proposed_Buf port map (ZERO1, ONE1, PC2, ZERO2, ONE2);
B003: Proposed_Buf port map (ZERO2, ONE2, PC3, ZERO3, ONE3);

A01: Proposed_ANDNAND2 port map (R0_L, G1_H, R0_H, G1_L, PC3,
SEL_GP1_H, SEL_GP1_L);
MUX1: Proposed_2_1_MUX port map (ZERO3, CR15_xor_IN_H, ONE3,
CR15_xor_IN_L, SEL_GP1_H, SEL_GP1_L, PC4, GP1_H, GP1_L);
B004: Proposed_Buf port map (CR0_H, CR0_L, PC3, CD1_H, CD1_L);
RB01: Proposed_dual_reset_buf port map (CD1_H, CD1_L, R1_L, R1_H,
PC4, CD2_H, CD2_L);
X01: Proposed_XORXNOR2 port map (CD2_H, GP1_H, CD2_L, GP1_L, PC1,
CD3_H, CD3_L);
MUX2: Proposed_2_1_MUXport map (CD3_H, Load0_H, CD3_L, Load0_L,
R3_H, R3_L, PC2, CR1_H, CR1_L);

END ARCHITECTURE structural;
```

**----------CRC_Datapath**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITY CRC_Datapath IS
port (MSG_IN_H, MSG_IN_L, PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H,
R1_L, R2_H, R2_L, R3_H, R3_L: in std_logic; Load_H, Load_L: in
std_logic_vector(15 downto 0); G_H, G_L : in std_logic_vector(15
downto 1); ZERO, ONE : in std_logic; CR_H, CR_L: inout
std_logic_vector(15 downto 0));
END ENTITY CRC_Datapath;

ARCHITECTURE Structural OF CRC_Datapath IS

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;
```

```
Component Proposed_dual_reset_buf
port (A_H, Res_H, A_L, Res_L, PC : in std_logic; Out_H, Out_L :
out std_logic);
End Component;

Component Proposed_XORXNOR2
port (A_H, B_H, A_L, B_L, PC: in std_logic; XOR2, XNOR2: out
std_logic);
End Component Proposed_XORXNOR2;

Component Proposed_2_1_MUX
port (A_H, B_H, A_L, B_L, S_H, S_L, PC: in std_logic; Out_H,
Out_L: out std_logic);
End Component;

Component CRC_Generator_Polynomial_bit_block
port (CR0_H, CR0_L, PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, L0_H, L0_L, G1_H, G1_L, CR15_xor_IN_H, CR15_XOR_IN_L,
ZERO, ONE: in std_logic; CR1_H, CR1_L: out std_logic);
End Component;

SIGNAL L_H, L_L, CR15_xor_IN_H, CD11_H, CD12_H, CR15_xor_IN_L,
CD11_L, CD12_L: std_logic;

BEGIN

BF01: Proposed_Buf port map (Load_H(0),Load_L(0), PC1, L_H, L_L);
XN01: Proposed_XORXNOR2 port map (CR_H(15), MSG_IN_H, CR_L(15),
MSG_IN_L, PC3, CR15_xor_IN_H, CR15_xor_IN_L);
BF17: Proposed_Buf port map (CR15_xor_IN_H, CR15_xor_IN_L, PC4,
CD11_H, CD11_L);
RB01: Proposed_dual_reset_buf port map (CD11_H, CD11_L, R2_L,
R2_H, PC1, CD12_H, CD12_L);
MUX1: Proposed_2_1_MUX port map (CD12_H, L_H, CD12_L, L_L, R3_H,
R3_L, PC2, CR_H(0), CR_L(0));
CRC_Gen_Poly_block1: CRC_Generator_Polynomial_bit_block port map
(CR_H (0), CR_L(0), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, Load_H(1), Load_L(1), G_H(1), G_L(1), CR15_xor_IN_H,
CR15_XOR_IN_L, ZERO, ONE, CR_H(1), CR_L(1));
CRC_Gen_Poly_block2: CRC_Generator_Polynomial_bit_block port map
(CR_H(1), CR_L(1), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, Load_H(2), Load_L(2), G_H(2), G_L(2), CR15_xor_IN_H,
CR15_XOR_IN_L, ZERO, ONE, CR_H(2), CR_L(2));
CRC_Gen_Poly_block3: CRC_Generator_Polynomial_bit_block port map
(CR_H(2), CR_L(2), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, Load_H(3), Load_L(3), G_H(3), G_L(3), CR15_xor_IN_H,
CR15_XOR_IN_L, ZERO, ONE, CR_H(3), CR_L(3));
CRC_Gen_Poly_block4: CRC_Generator_Polynomial_bit_block port map
(CR_H(3), CR_L(3), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
```

```
R3_H, R3_L, Load_H(4), Load_L(4), G_H(4), G_L(4), CR15_xor_IN_H,
CR15_XOR_IN_L, ZERO, ONE, CR_H(4), CR_L(4));
CRC_Gen_Poly_block5: CRC_Generator_Polynomial_bit_block port map
(CR_H(4), CR_L(4), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, Load_H(5), Load_L(5), G_H(5), G_L(5), CR15_xor_IN_H,
CR15_XOR_IN_L, ZERO, ONE, CR_H(5), CR_L(5));
CRC_Gen_Poly_block6: CRC_Generator_Polynomial_bit_block port map
(CR_H(5), CR_L(5), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, Load_H(6), Load_L(6), G_H(6), G_L(6), CR15_xor_IN_H,
CR15_XOR_IN_L, ZERO, ONE, CR_H(6), CR_L(6));
CRC_Gen_Poly_block7: CRC_Generator_Polynomial_bit_block port map
(CR_H(6), CR_L(6), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, Load_H(7), Load_L(7), G_H(7), G_L(7), CR15_xor_IN_H,
CR15_XOR_IN_L, ZERO, ONE, CR_H(7), CR_L(7));
CRC_Gen_Poly_block8: CRC_Generator_Polynomial_bit_block port map
(CR_H(7), CR_L(7), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, Load_H(8), Load_L(8), G_H(8), G_L(8), CR15_xor_IN_H,
CR15_XOR_IN_L, ZERO, ONE, CR_H(8), CR_L(8));
CRC_Gen_Poly_block9: CRC_Generator_Polynomial_bit_block port map
(CR_H(8), CR_L(8), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H, R1_L,
R3_H, R3_L, Load_H(9), Load_L(9), G_H(9), G_L(9), CR15_xor_IN_H,
CR15_XOR_IN_L, ZERO, ONE, CR_H(9), CR_L(9));
CRC_Gen_Poly_block10: CRC_Generator_Polynomial_bit_block port
map (CR_H(9), CR_L(9), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H,
R1_L, R3_H, R3_L, Load_H(10), Load_L(10), G_H(10), G_L(10),
CR15_xor_IN_H, CR15_XOR_IN_L, ZERO, ONE, CR_H(10), CR_L(10));
CRC_Gen_Poly_block11: CRC_Generator_Polynomial_bit_block port
map (CR_H(10), CR_L(10), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H,
R1_L, R3_H, R3_L, Load_H(11), Load_L(11), G_H(11), G_L(11),
CR15_xor_IN_H, CR15_XOR_IN_L, ZERO, ONE, CR_H(11), CR_L(11));
CRC_Gen_Poly_block12: CRC_Generator_Polynomial_bit_block port
map (CR_H(11), CR_L(11), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H,
R1_L, R3_H, R3_L, Load(12), Load_L(12), G_H(12), G_L(12),
CR15_xor_IN_H, CR15_XOR_IN_L, ZERO, ONE, CR_H(12), CR_L(12));
CRC_Gen_Poly_block13: CRC_Generator_Polynomial_bit_block port
map (CR_H(12), CR_L(12), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H,
R1_L, R3_H, R3_L, Load_H(13), Load_L(13), G(13), Gb(13),
CR15_xor_IN_H, CR15_XOR_IN_L, ZERO, ONE, CR_H(13), CR_L(13));
CRC_Gen_Poly_block14: CRC_Generator_Polynomial_bit_block port
map (CR_H(13), CR_L(13), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H,
R1_L, R3_H, R3_L, Load(14), Loadb(14), G_H(14), G_L(14),
CR15_xor_IN_H, CR15_XOR_IN_L, ZERO, ONE, CR_H(14), CR_L(14));
CRC_Gen_Poly_block15: CRC_Generator_Polynomial_bit_block port
map (CR_H(14), CR_L(14), PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H,
R1_L, R3_H, R3_L, Load_H(15), Load_L(15), G_H(15), G_L(15),
CR15_xor_IN_H, CR15_XOR_IN_L, ZERO, ONE, CR_H(15), CR_L(15));

END ARCHITECTURE Structural;
```

**----------16-bit CRC for 16-bit message word**

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

ENTITYMUX_Controller_CRC_Datapath IS
port (IN1_H, IN2_H, IN3_H, IN4_H, IN5_H, IN6_H, IN7_H, IN8_H,
IN9_H, IN10_H, IN11_H, IN12_H, IN13_H, IN14_H, IN15_H, IN16_H,
IN1_L, IN2_L, IN3_L, IN4_L, IN5_L, IN6_L, IN7_L, IN8_L, IN9_L,
IN10_L, IN11_L, IN12_L, IN13_L, IN14_L, IN15_L, IN16_L: in
std_logic; CLOCK : in std_logic_vector(1 downto 0); NM_H, NM_L,
RESET_H, RESET_L: in std_logic; R_count_H, R_count_L, R0_H, R0_L,
R1_H, R1_L, R2_H, R2_L, R3_H, R3_L, R4_H, R4_L, Msg_IN_H,
Msg_IN_L: inout std_logic; LV_H, LV_L, GP1_H, GP1_L : in
std_logic_vector(15 downto 1);
ZERO1, ONE1: in std_logic; CR_H, CR_L: inout std_logic_vector(15
downto 0));
END ENTITY MUX_Controller_CRC_Datapath;

ARCHITECTURE Structural OF MUX_Controller_CRC_Datapath IS

Component DC_TO_ADIABATIC
port (INP_H, INP_L: in std_logic; CLK: in std_logic_vector(1
downto 0); A_H, A_L: out std_logic);
End Component;

Component FOUR_PHASE_PC
port (CLK: in std_logic_vector(1 downto 0); PC1, PC2, PC3, PC4 :
out std_logic);
End Component;

Component Proposed_Buf
port (A_H, A_L: in std_logic; PC: in std_logic; Out_H, Out_L:
out std_logic);
End Component;

Component DC_TO_ADIABATIC_PC3_INP
port (INP_H, INP_L: in std_logic; CLK: in std_logic_vector(1
downto 0); A_H, A_L: out std_logic);
End Component;

Component controller_MUX
port (A_H, B_H, C_H, D_H, E_H, F_H, G_H, H_H, I_H, J_H, K_H, L_H,
M_H, N_H, O_H, P_H, A_L, B_L, C_L, D_L, E_L, F_L, G_L, H_L, I_L,
J_L, K_L, L_L, M_L, N_L, O_L, P_L, PC1, PC2, PC3, PC4, NM_H,
NM_L, RESET_H, RESET_L: in std_logic; R_count_H, R_count_L, R0_H,
R0_L, R1_H, R1_L, R2_H, R2_L, R3_H, R3_L, R4_H, R4_L, Msg_IN_H,
Msg_IN_L: inout std_logic);
```

```
END COMPONENT controller_MUX;

Component CRC_Datapath
port (MSG_IN_H, MSG_IN_L, PC1, PC2, PC3, PC4, R0_H, R0_L, R1_H,
R1_L, R2_H, R2_L, R3_H, R3_L: in std_logic; Load_H, Load_L, G_H,
G_L: in std_logic_vector(15 downto 1); ZERO, ONE : in std_logic;
CR_H, CR_L: inout std_logic_vector(15 downto 0));
END COMPONENT CRC_Datapath;

SIGNAL A_H, B_H, C_H, D_H, E_H, F_H, G_H, H_H, I_H, J_H, K_H,
L_H, M_H, N_H, O_H, P_H, A_L, B_L, C_L, D_L, E_L, F_L, G_L, H_L,
I_L, J_L, K_L, L_L, M_L, N_L, O_L, P_L: std_logic;
SIGNAL A1_H, B1_H, C1_H, D1_H, E1_H, F1_H, G1_H, H1_H, I1_H,
J1_H, K1_H, L1_H, M1_H, N1_H, O1_H, P1_H, A1_L, B1_L, C1_L, D1_L,
E1_L, F1_L, G1_L, H1_L, I1_L, J1_L, K1_L, L1_L, M1_L, N1_L, O1_L,
P1_L: std_logic;
SIGNAL  ZERO, ONE, NM11_H, NM11_L, NM12_H, NM12_L, NM1_H, NM2_H,
NM1_L,  NM2_L,  RES1_H,  RES1_L,  RES2_H,  RES2_L,  RES_H,  RES_L:
std_logic;
SIGNAL GP_H, GP_L: std_logic_vector (15 downto 1);
SIGNAL Load_H, Load_L: std_logic_vector (15 downto 0);
SIGNAL PC1, PC2, PC3, PC4: std_logic;

BEGIN

CLK1: FOUR_PHASE_PCLK port map (CLOCK, PC1, PC2, PC3, PC4);

INPUT1: DC_TO_ADIABATIC_PC3_INP port map (IN1_H, IN1_L, CLOCK,
A1_H, A1_L);
INPUT2: DC_TO_ADIABATIC_PC3_INP port map (IN2_H, IN2_L, CLOCK,
B1_H, B1_L);
INPUT3: DC_TO_ADIABATIC_PC3_INP port map (IN3_H, IN3_L, CLOCK,
C1_H, C1_L);
INPUT4: DC_TO_ADIABATIC_PC3_INP port map (IN4_H, IN4_L, CLOCK,
D1_H, D1_L);
INPUT5: DC_TO_ADIABATIC_PC3_INP port map (IN5_H, IN5_L, CLOCK,
E1_H, E1_L);
INPUT6: DC_TO_ADIABATIC_PC3_INP port map (IN6_H, IN6_L, CLOCK,
F1_H, F1_L);
INPUT7: DC_TO_ADIABATIC_PC3_INP port map (IN7_H, IN7_L, CLOCK,
G1_H, G1_L);
INPUT8: DC_TO_ADIABATIC_PC3_INP port map (IN8_H, IN8_L, CLOCK,
H1_H, H1_L);
INPUT9: DC_TO_ADIABATIC_PC3_INP port map (IN9_H, IN9_L, CLOCK,
I1_H, I1_L);
INPUT10: DC_TO_ADIABATIC_PC3_INP port map (IN10_H, IN10_L, CLOCK,
J1_H, J1_L);
INPUT11: DC_TO_ADIABATIC_PC3_INP port map (IN11_H, IN11_L, CLOCK,
K1_H, K1_L);
```

```
INPUT12: DC_TO_ADIABATIC_PC3_INP port map (IN12_H, IN12_L, CLOCK,
L1_H, L1_L);
INPUT13: DC_TO_ADIABATIC_PC3_INP port map (IN13_H, IN13_L, CLOCK,
M1_H, M1_L);
INPUT14: DC_TO_ADIABATIC_PC3_INP port map (IN14_H, IN14_L, CLOCK,
N1_H, N1_L);
INPUT15: DC_TO_ADIABATIC_PC3_INP port map (IN15_H, IN15_L, CLOCK,
O1_H, O1_L);
INPUT16: DC_TO_ADIABATIC_PC3_INP port map (IN16_H, IN16_L, CLOCK,
P1_H, P1_L);
BF01: Proposed_Buf port map (A1_H, A1_L, PC3, A_H, A_L);
BF02: Proposed_Buf port map (B1_H, B1_L, PC3, B_H, B_L);
BF03: Proposed_Buf port map (C1_H, C1_L, PC3, C_H, C_L);
BF04: Proposed_Buf port map (D1_H, D1_L, PC3, D_H, D_L);
BF05: Proposed_Buf port map (E1_H, E1_L, PC3, E_H, E_L);
BF06: Proposed_Buf port map (F1_H, F1_L, PC3, F_H, F_L);
BF07: Proposed_Buf port map (G1_H, G1_L, PC3, G_H, G_L);
BF08: Proposed_Buf port map (H1_H, H1_L, PC3, H_H, H_L);
BF09: Proposed_Buf port map (I1_H, I1_L, PC3, I_H, I_L);
BF010: Proposed_Buf port map (J1_H, J1_L, PC3, J_H, J_L);
BF011: Proposed_Buf port map (K1_H, K1_L, PC3, K_H, K_L);
BF012: Proposed_Buf port map (L1_H, L1_L, PC3, L_H, L_L);
BF013: Proposed_Buf port map (M1_H, M1_L, PC3, M_H, M_L);
BF014: Proposed_Buf port map (N1_H, N1_L, PC3, N_H, N_L);
BF015: Proposed_Buf port map (O1_H, O1_L, PC3, O_H, O_L);
BF016: Proposed_Buf port map (P1_H, P1_L, PC3, P_H, P_L);

INPUT17: DC_TO_ADIABATIC port map (RESET_H, RESET_L, CLOCK,
RES1_H, RES1_L);
INPUT18: DC_TO_ADIABATIC port map (NM_H, NM_L, CLOCK, NM1_H,
NM1_L);

BF017: Proposed_Buf port map (RES1_H, RES1_L, PC1, RES2_H,
RES2_L);
BF018: Proposed_Buf port map (RES2_H, RES2_L, PC2, RES_H, RES_L);
BF019: Proposed_Buf port map (NM1_H, NM1_L, PC1, NM11_H, NM11_L);
BF020: Proposed_Buf port map (NM11_H, NM11_L, PC2, NM12_H,
NM12_L);
BF021: Proposed_Buf port map (NM12_H, NM12_L, PC3, NM2_H, NM2_L);

INPUT19: DC_TO_ADIABATIC port map (ZERO1, ONE1, CLOCK, ZERO,
ONE);
INPUT24: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(1), GP1_L(1),
CLOCK, GP_H(1), GP_L(1));
INPUT25: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(2), GP1_L(2),
CLOCK, GP_H(2), GP_L(2));
INPUT26: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(3), GP1_L(3),
CLOCK, GP_H(3), GP_L(3));
```

```
INPUT27: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(4), GP1_L(4),
CLOCK, GP_H(4), GP_L(4));
INPUT28: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(5), GP1_L(5),
CLOCK, GP_H(5), GP_L(5));
INPUT29: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(6), GP1_L(6),
CLOCK, GP_H(6), GP_L(6));
INPUT30: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(7), GP1_L(7),
CLOCK, GP_H(7), GP_L(7));
INPUT31: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(8), GP1_L(8),
CLOCK, GP_H(8), GP_L(8));
INPUT32: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(9), GP1_L(9),
CLOCK, GP_H(9), GP_L(9));
INPUT33: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(10), GP1_L(10),
CLOCK, GP_H(10), GP_L(10));
INPUT34: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(11), GP1_L(11),
CLOCK, GP_H(11), GP_L(11));
INPUT35: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(12), GP1_L(12),
CLOCK, GP_H(12), GP_L(12));
INPUT36: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(13), GP1_L(13),
CLOCK, GP_H(13), GP_L(13));
INPUT37: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(14), GP1_L(14),
CLOCK, GP_H(14), GP_L(14));
INPUT38: DC_TO_ADIABATIC_PC3_INP port map (GP1_H(15), GP1_L(15),
CLOCK, GP_H(15), GP_L(15));

INPUT40: DC_TO_ADIABATIC port map (LV_H(0), LV_L(0), CLOCK,
Load_H(0), Load_L(0));
INPUT41: DC_TO_ADIABATIC port map (LV_H(1), LV_L(1), CLOCK,
Load_H(1), Load_L(1));
INPUT42: DC_TO_ADIABATIC port map (LV_H(2), LV_L(2), CLOCK,
Load_H(2), Load_L(2));
INPUT43: DC_TO_ADIABATIC port map (LV_H(3), LV_L(3), CLOCK,
Load_H(3), Load_L(3));
INPUT44: DC_TO_ADIABATIC port map (LV_H(4), LV_L(4), CLOCK,
Load_H(4), Load_L(4));
INPUT45: DC_TO_ADIABATIC port map (LV_H(5), LV_L(5), CLOCK,
Load_H(5), Load_L(5));
INPUT46: DC_TO_ADIABATIC port map (LV_H(6), LV_L(6), CLOCK,
Load_H(6), Load_L(6));
INPUT47: DC_TO_ADIABATIC port map (LV_H(7), LV_L(7), CLOCK,
Load_H(7), Load_L(7));
INPUT48: DC_TO_ADIABATIC port map (LV_H(8), LV_L(8), CLOCK,
Load_H(8), Load_L(8));
INPUT49: DC_TO_ADIABATIC port map (LV_H(9), LV_L(9), CLOCK,
Load_H(9), Load_L(9));
INPUT50: DC_TO_ADIABATIC port map (LV_H(10), LV_L(10), CLOCK,
Load_H(10), Load_L(10));
INPUT51: DC_TO_ADIABATIC port map (LV_H(11), LV_L(11), CLOCK,
Load_H(11), Load_L(11));
```

```
INPUT52: DC_TO_ADIABATIC port map (LV_H(12), LV_L(12), CLOCK,
Load_H(12), Load_L(12));
INPUT53: DC_TO_ADIABATIC port map (LV_H(13), LV_L(13), CLOCK,
Load_H(13), Load_L(13));
INPUT54: DC_TO_ADIABATIC port map (LV_H(14), LV_L(14), CLOCK,
Load_H(14), Load_L(14));
INPUT55: DC_TO_ADIABATIC port map (LV_H(15), LV_L(15), CLOCK,
Load_H(15), Load_L(15));

Cont_MUX1: controller_MUX port map (A_H, B_H, C_H, D_H, E_H, F_H,
G_H, H_H, I_H, J_H, K_H, L_H, M_H, N_H, O_H, P_H, A_L, B_L, C_L,
D_L, E_L, F_L, G_L, H_L, I_L, J_L, K_L, L_L, M_L, N_L, O_L, P_L,
PC1, PC2, PC3, PC4, NM2_H, NM2_L, RES_H, RES_L, R_count_H,
R_count_L, R0_H, R0_L, R1_H, R1_L, R2_H, R2_L, R3_H, R3_L, R4_H,
R4_L, Msg_IN_H, Msg_IN_L);
CRC_Dpath1: CRC_Datapath port map (MSG_IN_L, MSG_IN_L, PC1, PC2,
PC3, PC4, R0_H, R0_L, R1_H, R1_L, R2_H, R2_L, R3_H, R3_L, Load_H,
Load_L, GP_H, GP_L, ZERO, ONE, CR_H, CR_L);

END ARCHITECTURE Structural;
```