

University of Windsor

Scholarship at UWindor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

2009

Low-power CMOS rectifier and Chien search design for RFID tags

Shu-Yi Wong
University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

Recommended Citation

Wong, Shu-Yi, "Low-power CMOS rectifier and Chien search design for RFID tags" (2009). *Electronic Theses and Dissertations*. 8065.
<https://scholar.uwindsor.ca/etd/8065>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Low-Power CMOS Rectifier and Chien Search Design for RFID Tags

by

Shu-Yi Wong

A Thesis

Submitted to the Faculty of Graduate Studies through the
Department of Electrical and Computer Engineering in Partial Fulfillment
Of the Requirements for the Degree of Master of Applied Science at
The University of Windsor

Windsor, Ontario, Canada

2009

© 2009 Shu-Yi Wong



Library and Archives
Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*
ISBN: 978-0-494-57599-4
Our file *Notre référence*
ISBN: 978-0-494-57599-4

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

■♦■
Canada

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

I. Co-Authorship Declaration

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

In all cases, the primary contributions, derivations, experimental setups, data analysis and interpretation were performed by the author through the supervision of Dr. C. Chen. In addition to supervision, Dr. C. Chen provided the author with the project idea, guidance, and financial support. Dr. C. Chen has also held regular meetings with Dr. Q. M. Jonathan Wu regarding project discussion and follow-ups about the progress and outcome of this project.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Declaration of Previous Publication

This thesis includes three original papers that have been previously published / submitted for publication in peer reviewed journal and conferences, as follows:

Thesis Chapter	Publication title/full citation	Publication status*
<i>Chapter 2</i>	S.Y. Wong, C. Chen, Q.M.J. Wu, "Low Power Chien Search for BCH Decoder Using RT-Level Power Management", IEEE Transactions on Very Large Scale Integration Systems (TVLSI)	"accepted" for publication as a transaction brief
<i>Chapter 2</i>	S.Y. Wong, C. Chen and Q.M.J. Wu, "Power-Management-Based Chien Search for Low Power BCH Decoder," in Proc. 14 th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), Aug. 19-21 2009, pp. 299-302	"Published" in poster session
<i>Chapter 3, 4,5,6,7 and 8</i>	S.Y. Wong, C. Chen and Q.M.J. Wu, "Power Optimization of Multi-Stage Differential-Drive CMOS Rectifier for Passive RFID Tags"	"Submitted " to the 47 th Design Automation Conference

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Automotive sensors implemented in radio frequency identification (RFID) tags can correct data errors by using BCH (Bose-Chaudhuri-Hocquenghem) decoder, for which Chien search is a computation-intensive key step. Existing low power approaches have drastically degrading performance for multiple-bit-correcting codes. This thesis presents a novel approach of using register-transfer-level (RTL) power management in the search process, leading to significant power savings for BCH codes with higher correction capability. An example for the (255, 187, 9) BCH code has been implemented in 0.18 μm CMOS technology.

We also consider ways of conserving power for the sole power harvester on a passive tag – the rectifier. With ST CMOS 90nm technology, a three-stage differential-drive CMOS rectifier is designed by using a new transistor scaling method and a piece-wise linear matching technique. For the standard 915MHz band, simulation indicates high power conversion efficiency (PCE) of 74% and a significantly increased output power of 30.3 μW at 10 meters.

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to Dr. Chunhong Chen, who is my supervisor. He introduced me to the subject of low power design and offered me invaluable comments and encouragement throughout my two years in Windsor. His patience and his guidance, all contribute to the success of our papers. I also like to thank my co-supervisor Dr. Jonathan Wu for introducing me to the interesting area of intelligent sensors. He guided me initially and eventually led to my focus on RFID. My appreciation also goes to the thesis committee, Dr. Ziad Kobti and Dr. Rashid Rashidzadeh. The many circuit design and testing techniques are all too valuable, for these I especially like to thank Dr. Rashid. Equally appreciated are the insightful DSP lessons taught by Dr. Majid Ahmadi. Certainly, this research would not be possible without the full support of AUTO21 Network of Centres of Excellence, Canada. I also want to express my appreciation for Dr. Kemal Tepe, who gave useful comments while acting as my committee member and thanks to Dr. Narayan Kar, who chaired my defense. Many thanks go to Ms. Andria Ballo and Ms. Shelby Marchand, who always help me on the unwieldy administrative details.

I wish to especially thank my wife Connie for her unwavering support and understanding during the many hours I dedicated to achieving this milestone in my life. My heartfelt appreciation goes to my mother-in-law for her support. Many thanks will not express my gratitude to my in-law families, my sister Lucille in Hong Kong, and my brothers and sisters in Windsor. Their prayers help me through difficult times. I thank my friend Mr. Ning Chang who has provided me timely help in a lot of occasions and Mr. Otto Yung and Mr. Michael Cheung who have reminded me the importance of thanksgiving. Life is just not the same without my dear brother Kendrick and my sister-in-law Joyce. To my dear father and mother, a few thanking statements will never do the justice of your countless years of upbringing, persistent support and encouragement. This thesis was done on the very laptop computer my father gave me as a gift. At the end, the deepest thank to my Lord, the fountain of my life, the giver of my faith, the lamp that lights my path, the source of my wisdom and innovation and by whom all things consist.

TABLE OF CONTENTS

Declaration of Co-Authorship / Previous Publication.....	iii
Abstract	v
Acknowledgement.....	vi
List of Figures	x
List of Tables.....	xii
List of Abbreviations.....	xiii
Chapter 1 Introduction.....	1
1.1 General RFID Systems and UHF Passive RFID Tags.....	1
1.2 Error Correction and Chien Search.....	2
1.3 Rectifier	3
1.4 Motivation and Achievements	7
1.5 Thesis Organization	9
Chapter 2 Low Power Chien Search for BCH Decoder	11
2.1 Review of Low Power Chien Search Methods.....	11
2.2 RT-Level Power Management for Chien Search.....	11
2.2.1 Power Modeling for the Existing Methods	11
2.2.2 The Proposed Method	15
2.2.3 A Polynomial Order Reduction Algorithm	19
2.3 Circuit Implementation for the Proposed Low Power Chien Search.....	21
2.3.1 Adder for POR.....	22
2.3.2 Multiplexer and Clock-Gating.....	24
2.3.3 Layout and Area Overhead.....	24
2.3.4 Parallel Chien Search and Power Overhead.....	25
2.4 Simulation Results	26
Chapter 3 Differential-Drive CMOS Bridge Rectifier	30
3.1 Differential-Drive CMOS Half Bridge and Full Bridge Circuits	30
3.2 DC Approximation for Quasi-Steady State	31
3.3 AC model for the Half Bridge Rectifier	32
3.4 DC Analysis for the Current through the Half Bridge Rectifier.....	34

3.4.1	Forward Conduction Region	34
3.4.2	Reversed Triode Conduction.....	35
3.4.3	Partial Saturation	36
3.4.4	Full Saturation	36
3.4.5	Subthreshold Conduction	36
3.4.6	Reverse Leakage.....	37
Chapter 4 Power Efficiency Analysis		38
4.1	Power Conversion Efficiency (PCE).....	38
4.2	Maximizing PCE by 3-D Contour Analysis	40
4.3	Effect of Body Voltage to Maximum PCE.....	43
4.4	Effect of Finite Gate and Substrate Resistance to Maximum PCE.....	45
Chapter 5 Simplified Power Model.....		46
5.1	Approximating the Non-Linear Rectifier by a Linear Resistor	46
5.2	The Piece-wise Linear Model for DDCHB	50
5.3	Comparison between the Model and the Simulated Behavior.....	51
Chapter 6 Antenna Matching for Rectifier.....		52
6.1	The Difficulties of Antenna Matching for Rectifier with High PCE.....	52
6.2	Piece-Wise Linear Model for Matching	52
6.3	The Inadequacy of PCE for Rectifier Design	54
6.4	Power Utilization for the Rectifier.....	54
6.5	Determination of the Equivalent Radiative Resistance	55
6.6	Sizing of Transistors for Optimal Power Transfer	56
6.7	Matching to Simple Antenna	56
Chapter 7 Multi-Stage CMOS Differential Full-Bridge Rectifier Design		60
7.1	Topology of Multi-Stage DDCFB	60
7.2	Optimal Number of Stages	63
7.3	Coupling Capacitors and Parasitic Gate Capacitances	63
7.4	Other Parasitic Components	66
7.5	Three-Stages CMOS Differential Full-Bridge Rectifier Design	67
7.6	PCE and PU at Various Distances	68
7.7	Comparison with Prior Design Methods	70

Chapter 8 Conclusions and Future Work	71
8.1 Conclusions and Contributions.....	71
8.2 Future Work.....	72
References	73
APPENDIX A: RTL Codes for Chien Search.....	77
A.1 RTL Codes for the Conventional Chien Search.....	78
A.2 RTL Codes for the Proposed Low-Power Chien Search.....	102
A.3 MATLAB Source Codes.....	130
APPENDIX B: MATLAB Codes for DDCHB	138
B.1 Transistor Parameters Extraction	139
B.2 3-D Contour Analysis.....	148
APPENDIX C: Permission from Co-authors and IEEE/ACM	159
Permission to Use Previously Submitted/Accepted Papers	160
VITA AUCTORIS	170

LIST OF FIGURES

Figure 1.1: A Register-Transfer (RT) Level Architecture for the Chien Search.....	3
Figure 1.2: Basic Schottky Rectifier Circuit (Left), Dickson Multiplier Configuration (Right)	5
Figure 1.3: Diode-Connected NMOS (Left), PMOS (Right).....	6
Figure 2.1: Power Estimation by Determining the Bit Distance l	14
Figure 2.2: Comparison of Power Savings for Various Percentages of Un-correctable Frames	19
Figure 2.3: The Block Diagram of POR.....	23
Figure 2.4: One Bit of the POR.....	23
Figure 2.5: The Clock Gating Circuit.....	24
Figure 2.6: Core Layout of the Proposed Chien Search.....	25
Figure 2.7: Power Savings for $(255, k, t)$ BCH codes	29
Figure 3.1: Differential-Drive CMOS Half Bridge Rectifier (Left), Full Bridge (Right)	30
Figure 3.2: AC Equivalent Circuit for the Half Bridge	33
Figure 3.3: A Forwardly (Left) and a Reversely (Right) Biased MOS in Dickson Multiplier.....	36
Figure 4.1: 3-D Contour for $V_{sbn}=0, V_{sbp}=0$ and $\delta=0.82$	42
Figure 4.2: Enhanced Contrast 2-D Contour Projection for $V_{sbn}=0, V_{sbp}=0$ and $\delta=0.82$..	43
Figure 4.3: Efficiency Contour for $V_{sbp}=0, V_{sbn}=0.66V$ and $\delta=0.82$	44
Figure 4.4: Enhanced Contrast 2-D Contour Projection for $V_{sbn}=0.66V, V_{sbp}=0$ and $\delta=0.82$	44
Figure 5.1: Simulated Rectifier Current vs Fix-Resistor Modeled Current	48
Figure 5.2: Simulated Power Loss vs Fix-Resistor Modeled Power Loss	49
Figure 5.3: Piece-wise Linear Model of the Half Bridge	50
Figure 5.4: Comparison between the Simulated PCE, the Modeled PCE and δ	51
Figure 6.1: Model of the Antenna Coupled Rectifier Circuit.....	53
Figure 6.2: Equivalent Circuit of the Rectifier for Matching.....	57
Figure 6.3: Serial Equivalent Circuit of figure 6.2.....	57

Figure 6.4: L-match Network to a Dipole Antenna.....	58
Figure 7.1: A 3-Stage DDCFB	61
Figure 7.2: A Basic DDCFB Cell for Multi-Stage Design (left), the Equivalent Circuit (right).....	62
Figure 7.3: Parasitic Capacitances in the Equivalent Circuit	64
Figure 7.4: Input Driving Top Plate of C_c	66
Figure 7.5: Input Driving Bottom Plate of C_c	67
Figure 7.6: Output Current for Rectifiers Optimized for Various Distances	70

LIST OF TABLES

Table 1.1: Available Power at Various Distance for $P_{EIRP}=4W$	4
Table 1.2: Average Power Consumption for Features on State-of-the-Art Low-Power RFID Tags	4
Table 2.1: Component Power for the Conventional Circuit.....	26
Table 2.2: Component Power for the Proposed Circuit	27
Table 5.1: Parameters for a Sample Rectifier Circuit and the Model	47
Table 7.1: V_{th} at Different Body Bias Voltage	60
Table 7.2: Transistor Parameters for the 3-stage Rectifier.....	68
Table 7.3: Passive Component Values for the 3-stage Rectifier.....	68
Table 7.4: PCE and PU for the 3-Stage Rectifier.....	69

LIST OF ABBREVIATIONS

AC	Alternating Current
ACM	Association for Computing Machinery
AES	Advanced Encryption Standard
ARQ	Automatic Repeat-Requests
AWGN	Additive White Gaussian Noise
BCH	Bose-Chaudhuri-Hocquenghem
CGFM	Constant Galois Field Multiplier
CMOS	Complementary Metal Oxide Semiconductor
CRC	Cyclic Redundancy Check
DC	Direct Current
DDCFB	Differential-Drive CMOS Full-Wave Bridge Rectifier
DDCHB	Differential-Drive CMOS Half-Wave Bridge Rectifier
EEPROM	Electrically Erasable Programmable Read-Only Memory
EIRP	Equivalent Isotropic Radiated Power
EVC	External Threshold Voltage Cancelation
FCC	Federal Communication Commission
GF	Galois-Field
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
IVC	Internal Threshold Voltage Cancelation

MATLAB	Matrix Laboratory
NMOS	N-Type Metal Oxide Semiconductor
PCE	Power Conversion Efficiency
PMOS	P-Type Metal Oxide Semiconductor
POR	Polynomial Order Reduction
PU	Power Utilization
RF	Radio Frequency
RFID	Radio Frequency Identification
RT	Register Transfer
RTL	Register-Transfer-Level
SPICE	Simulation Program Integrated Circuit Emphasis
ST	STMicroelectronics
SVC	Self Threshold Voltage Cancelation
UHF	Ultra High Frequency
3-D	Three Dimensional

Chapter 1 INTRODUCTION

1.1 General RFID Systems and UHF Passive RFID Tags

RFID is the abbreviation for Radio Frequency Identification. A typical RFID system consists of readers and multiple movable devices known as tags. A reader is a stationary device that initiates communication by sending radio frequency or RF signals to tags and responses to their RF reply signals. A tag could be powered either by an external battery or entirely by the incident RF energy from the reader. The former is known as an active tag and the latter is a passive tag. In terms of power transfer, RF energy may be coupled inductively or through electromagnetic wave capture. At standard frequencies of 128kHz or 13.56MHz [1], low loss coils may be built for efficient inductive power transfer but only when the coils are very close to the reader's antenna, leading to detection range less than 1m. Application examples of this type include door entry security, electronic payment system, and monitoring of goods and books that leave a shop or library. At Ultra High Frequency (UHF) or 300MHz to 3GHz [1], when small antenna can be constructed for efficient far field operation, tags that absorb power directly from the electromagnetic field can be designed to operate beyond 1m. This is useful for supply chain management [1], warehouse management, baggage handling and electronic toll highway, just to name a few applications. However, Federal Communication Commission (FCC) and other world regulation agencies govern the use of these frequency bands and available power for RFID tags is usually at a few tens of micro-watt or less. In this sense, the frequency band of 915MHz provides a good balance of detection range and power as it has the highest power allowed by FCC amongst all the other UHF bands. If a highly efficient power extraction mechanism is developed for passive tags, more power will become available; therefore, applications are no longer restricted to simple identification tasks but more complicated functions such as intelligent sensors become possibilities. The lack of external battery in a passive tag also means a tremendous cost saving, leading to tags that are applicable everywhere. This could open doors to the possibility of low-cost ubiquitous sensor network [2].

1.2 Error Correction and Chien Search

This research considers passive RFID tags in the particular operating environment of automotive industry, where noise immunity is an important issue. Existing RFID standards focus only on identification tasks, which typically mean short transmissions of similar data for any given tag. This, however, is not true for applications such as intelligent sensors, since the involved data patterns may be longer and potentially much more complex. Therefore, error correction utilizing automatic repeat-requests (ARQ) [28] that are based on simple cyclic redundancy check (CRC) is utterly inadequate. Moreover, ARQ is also inefficient for longer data, particularly in a master-slave communication setting where a tag is always responding passively to the reader's initiation. It appears that a more elaborate error correction system is desired. To this end, forward error correction represents a potential solution.

Binary BCH (Bose-Chaudhuri-Hocquenghem) code is a proven robust and popular forward-error correcting code over Galois-field $GF(2^m)$ in the form of (n, k, t) , where $n = 2^m - 1$ is the code-word length for some positive integer m , k is the source message length, and t stands for the error-correcting capability in bits [3]. The demand for a highly efficient power extraction mechanism, as discussed in section 1.1, certainly mandates low-power methodology to be applied throughout the design of a RFID tag. If a binary BCH decoder is to be used, there is no exception. Syndrome-based decoding is the usual technique applied to linear block code, such as BCH code. The three steps of decoding are namely, syndrome computation, finding an error locator polynomial, and solving the polynomial. The first two steps have received much attention in recent research activities [4-6]. The third step of solving the error locator polynomial $\Lambda(X)$, as shown in (1.1), is commonly implemented by the Chien search:

$$\left\{ \begin{array}{l} \Lambda_t(R_i) = 1 + \sum_{i=1}^t R_i \\ \text{where } R_i = \sigma_i X^i \quad \text{and} \quad X = \alpha^z \end{array} \right. \quad (1.1)$$

Low-power Chien search design is, in fact, one of the key topics in this thesis because Y. Wu [8] shows that Chien search accounts for as much as 61% of the overall decoder power consumption. The conventional serial implementation of the Chien search was

first introduced by R. T. Chien [7]. Thereafter, it is referred to as the conventional Chien search and used as the reference throughout this thesis. In (1.1), all variables are $GF(2^m)$ m -tuples, and $GF(2^m)$ operations are assumed to use polynomial bases. The α is the primitive element of $GF(2^m)$, while σ_1 through σ_t are the coefficients of the polynomial $\Lambda(X)$, and R_i represents the registers holding results of multiplication. The Chien search involves an exhaustive linear search for all possible error positions, and an error is found when $\Lambda_t(R_i) = 0$. Figure 1.1 (solid-line portion) shows a register-transfer (RT)-level architecture of the conventional Chien search. The proposed low-power method (dotted-line portion) will be discussed in chapter 2.

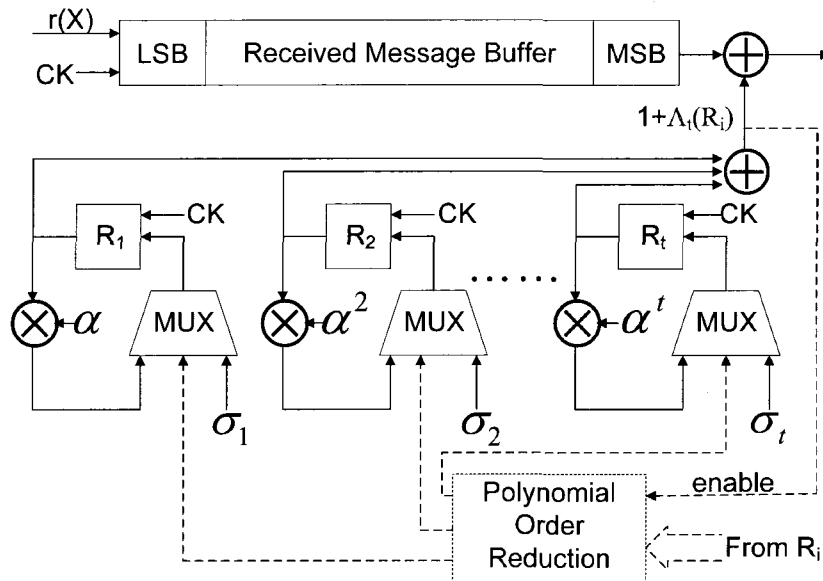


Figure 1.1: A Register-Transfer (RT) Level Architecture for the Chien Search

1.3 Rectifier

Rectifier is the front-end analog circuit on a RFID tag that converts the incident RF power in air to electrical power, by which the rest of the tag's circuits may operate. For a passive RFID tag, the rectifier circuit is especially important due to it being the sole source of power and this power, to a great extent, determines the types of application that the RFID system is capable of supporting. FCC regulates the amount of RF power in-the-air by the concept of equivalent isotropic radiative power (EIRP). This is defined as the amount of power that supplies an ideal antenna, which radiates by the same intensity in all directions. For the frequency band of 915MHz, EIRP equals 4W. By defining the term

“available power” as the power received by a resistive load when it is impedance matched to the antenna, the available power to a tag at various distances from the reader is illustrated in table 1.1. Table 1.2 shows the average power consumption of some state-of-the-art low-power features for RFID tags.

Table 1.1: Available Power at Various Distance for $P_{EIRP}=4W$

Distance (m)	Available Power (μW)
1.76	1450
3	496
5	178
10	44
20	11
30	5

Table 1.2: Average Power Consumption for Features on State-of-the-Art Low-Power RFID Tags

Features	Power Consumption
AES Cryptography @ 3.55MHz [13]	4.7 μW
1.28MHz Oscillator [14]	0.44 μW
512-bit EEPROM read [15]	8.34 μW (100% duty on memory)
512-bit EEPROM write [15]	57.7 μW (100% duty on memory)
ISO 18000-4B compatible tag [16]	1.05 μW (0.1% duty on memory)
Smart temperature sensor [17]	0.9 μW

Simple comparison of table 1.1 and 1.2 offers insights to the tightness of the power budget, which is especially the case when a tag is desired to work at great distances from the reader. For tags based on standards, such as ISO 18000-4B, advancement in low-power design in recent years has resulted in improved power consumption to around 2 μW [18]. This helps extending the operating range to almost 10 meters and beyond, while the older technology has provided a range of only 3 meters or less. There is also considerable advancement for specific type of sensor tags, such as temperature sensor. The low-power analog-to-digital converter required by these sensor has been consuming tens of micro-watts but recent effort has reduced the power consumption below 1 μW

[17]. Nevertheless, numerous other sensor applications, which utilize more complex digital functions and more elaborate memory processing, remain to be optimized power-wise. To this end, the application circuit of concern can certainly be the goal of optimization. However, a direct approach that helps releasing more power from the rectifier may be equally beneficial.

Early UHF rectifier designs use mostly Schottky diodes for their properties of low forward voltage and low reverse recovery time. A simple rectifier has the form of a peak detector shown on the left side of figure 1.2, which contains no more than a diode and a capacitor. Due to the low input voltage, more stages must be cascaded for a larger output voltage in order to support the back-end circuits. Dickson Multiplier is the usual topology for cascading, shown on the right side of figure 1.2. An important merit is power conversion efficiency (PCE), which is defined as the real power to the load divided by the real power into the rectifier. Although Schottky diode has low forward voltage, this voltage is still finite and in the order of 200mV or more, which results in a typical PCE value of only 18% [18] at the operating range of concern.

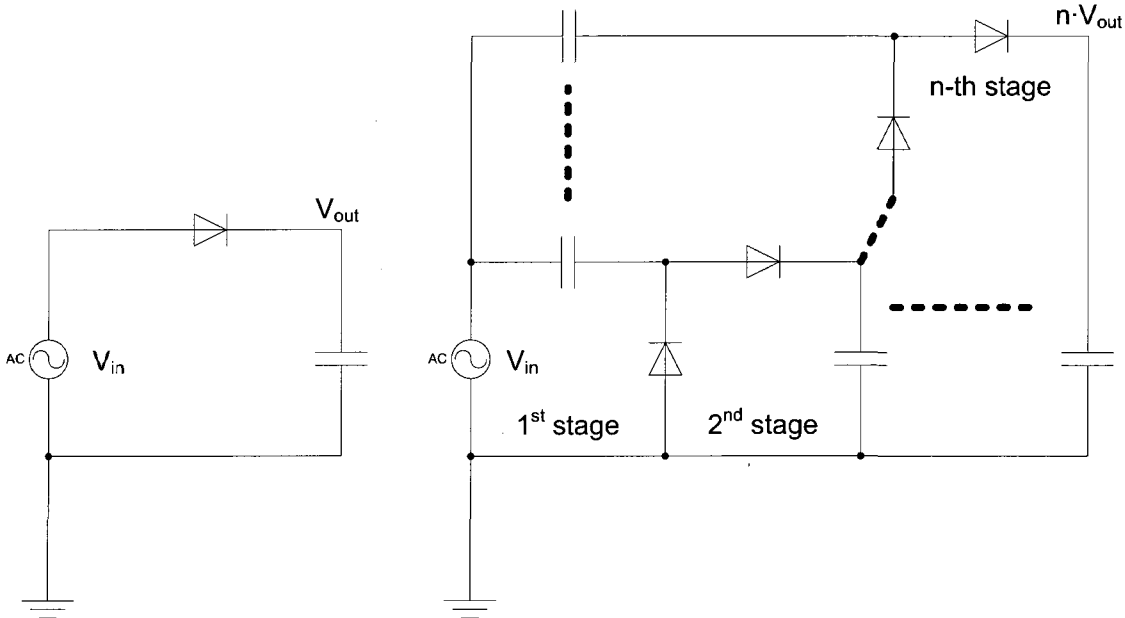


Figure 1.2: Basic Schottky Rectifier Circuit (Left), Dickson Multiplier Configuration (Right)

Extremely low cost can be achieved with standard CMOS process when RFID tags are mass produced. However, Schottky diode is not compatible with the process. A solution is to use diode-connected MOS transistors as replacements. Figure 1.3 shows the diode equivalents of both an NMOS and PMOS transistors.

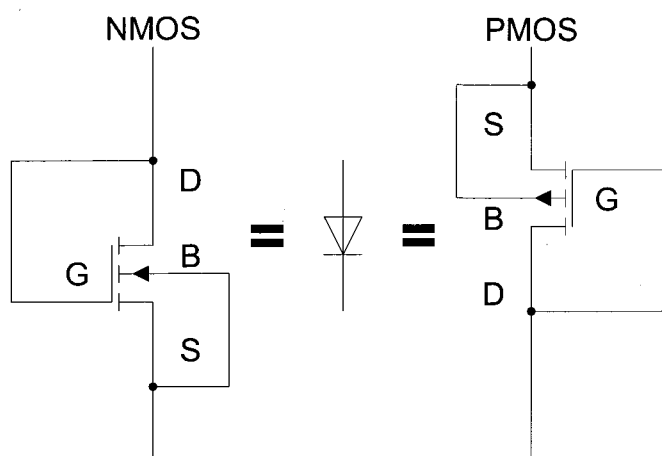


Figure 1.3: Diode-Connected NMOS (Left), PMOS (Right)

The forward voltages for these diode-connected transistor replacements are at least greater than the turn-on threshold voltages or V_{th} of the transistors. For CMOS process such as $0.5\mu\text{m}$ or $0.35\mu\text{m}$, V_{th} is often more than 0.5V . Therefore, in general, the replacement will lead to PCE inferior to that of the Schottky rectifier.

State-of-the-art rectifiers use these transistors but the problem of high V_{th} is resolved by introducing a DC bias voltage to each of the gates. This is done so that the power degradation effect due to the V_{th} is effectively cancelled. The gates may be biased by different methods. These are such as the external threshold voltage cancellation (EVC) [24], internal threshold voltage cancellation (IVC) [23] and self threshold voltage cancellation (SVC) [25]. Amongst these methods, a high PCE of 37% has been reported. To the extreme end, zero threshold transistors [20] have been used but the power efficiency is not improved because of the significant amount of reverse leakage. This, in fact, implies that V_{th} alone may not be the only limiting factor for power efficiency improvement. The minimization of reverse leakage may be just as important. To this end, a recent study of differential-drive CMOS full-wave bridge rectifier [19] points out the circuit's unique leakage behavior, which contributes to the much higher PCE of 66%.

The projected PCE, at its best, is only applicable for the particular transistor sizes and load. So, it still remains an open question whether a different set of design parameters will result in even better PCE. Furthermore, the study [19] shows results for only a single stage design but without a practical multi-stage solution required by most tags. Most studies assume body effect will degrade PCE for multi-stage CMOS transistor rectifier. This is because all N-type transistors in a rectifier circuit share the same bulk; the source voltage for each transistor is, therefore, gradually elevated over the number of stages. The V_{th} likewise increases with the source bias level. However, at least for the differential-drive CMOS full-wave bridge (DDCFB) rectifier, the PCE degradation hypothesis is not conclusive. This thesis will show, in chapter three, that the V_{th} model for PCE is an over-simplification and that the reverse leakage must be considered. A separate study [21] has investigated a three-stage DDCFB but with a PCE of only 23%. The study also implied higher PCE being unlikely due to the conflicting requirement of bandwidth and detection range. With this consideration, the thesis will tackle the problem of DDCFB power optimization from a different angle by constructing a model that combines the effects of V_{th} and the reverse leakage. Our research effort results in drastically different conclusions, as they will be developed gradually after chapter three.

1.4 Motivation and Achievements

As discussed in section 1.2 and 1.3, we are investigating the potential of more complex RFID tags for noisy automotive environment. This type of application would obviously require simultaneous high noise immunity and low-power, since a passive RFID tag draws its power entirely from the incident RF field.

The requirement of high noise immunity can potentially be tackled by a good forward error correction scheme. For this thesis, we propose BCH decoder, which is known to be robust and it is capable of correcting many random bits per data frame, depending on the particular choice of BCH code. Within a BCH decoder, the most power hungry circuit is the Chien search [8], which takes part in the last step of decoding. Our main achievement is in doubling the power savings of Chien search relative to that of the existing low-power method [8] for BCH codes with high error-correcting capabilities. In the process, we have developed a RT-level power model for explaining both the power savings for

the existing low power method and that of our proposed method. Analysis has been performed with both the standard AWGN model and a model that assume the same probability of a data frame carrying any number of errors (equal probability model). It is shown that the equal probability model, which is more easily applicable, can match the results of AWGN model closely when the BCH decoder is driven to its edge of error-correcting capability. By this newly developed power model, we have proposed our new low-power method based on RT-level power management. To enable this power management, we have also come up with the so-called Polynomial Order Reduction, which makes redundant on the fly the constant Galois field multipliers (CGFMs), key circuits in Chien search. The proposed Chien search circuit has been implemented in CMOS 0.18 μ m technology. Power has been simulated with layout extracted RC information for high accuracy and the results shows good agreement with our model.

On the other hand, we have worked on the rectifier with an attempt to increase the amount of output power by optimizing the circuit. The starting point of our investigation is differential-drive CMOS full-wave bridge (DDCFB) rectifier due to the recently reported high PCE of 66% at an input level of -12dBm [19]. Our main contributions include a full analytical analysis of DDCFB, which leads to a MATLAB based 3-D contour analysis that predicts the point of maximum PCE for a single stage DDCFB. We have also discovered that body effect does not necessarily degrade PCE and how transistors may be scaled without affecting PCE. These observations have been verified with actual SPICE simulation results with good agreement. Furthermore, we recognize that the non-linear rectifier of concern requires a different matching model. Consequently, we develop, from the results of 3-D contour analysis, a simplified piece-wise linear model for matching. It has been discovered that there exists an optimal transistor sizes for a given design range. Results from the model have been applied to the design of a three-stage DDCFB circuit in ST CMOS 90nm technology. Simulation shows PCE of 74% and an output power of 30.3 μ W at the maximum design operating range of 10m. The simulated overall system efficiency is at least 80% higher than any of the reported existing test cases [21] for the same rectifier topology.

1.5 Thesis Organization

This thesis is organized as follows.

The whole Chapter 2 is about low-power Chien search. Previous work [4-6] is first described; and it is followed by the development of RT-level power management models, both for the existing method [8] and the proposed method with or without using the standard additive white Gaussian noise (AWGN) model. The chapter then continues with the derivation of the Polynomial Order Reduction method, which enables the power savings. Immediately after that are the detail aspects of the circuit implementation with a mix of layout discussions and implications for parallel implementation. At the end of the chapter, power simulation is discussed with tabled results and graph.

Chapter 3 introduces the differential-drive CMOS full-wave bridge (DDCFB) circuit, which is subsequently simplified to differential-drive CMOS half-bridge (DDCHB) for ease of analysis. Some salient features and the relationship between the full-bridge and the half-bridge circuits are pointed out with a detail analytical analysis of the half-bridge.

The initial part of Chapter 4 explains the difficulties of both a full analytical analysis and a SPICETM simulation. This leads to the development of a MATLABTM based contour analysis that uses transistor parameters extracted from SPICETM. The results of the contour analysis are compared with the actual SPICE simulation results at a few selected instances. Both the discrepancies and the implications for body effect are then discussed.

Chapter 5 starts from the results of the contour analysis and introduces a simplified PCE model, which will help the later development of a different matching concept. The approximation is first explained and the modeled results are later compared with the actual SPICE simulated results. An equivalent circuit for the model is also explained.

Chapter 6 continues from the equivalent circuit from chapter 5. A matching model is subsequently derived. On top of the concept of PCE, the new matching model leads to the concept of power utilization (PU). Implications of the new model are then discussed. After that, method for coupling to a simple dipole antenna is explained in order that parameters for matching network may be determined for maximum power transfer.

Chapter 7 describes the design and power optimization for a three-stage differential-drive CMOS full-wave bridge (DDCFB). This chapter focuses on the practical aspects such as the optimal number of stages, use of coupling capacitors, the effect of finite gate capacitance and the non-ideal coupling capacitors. The design parameters are given at the end of the chapter with detail power simulation result. We conclude this thesis and offers directions for future work in Chapter 8.

Chapter 2 LOW POWER CHIEN SEARCH FOR BCH DECODER

2.1 Review of Low Power Chien Search Methods

The importance of power-efficient Chien search has been evident for many applications [4-6, 8-12]. One of the obvious low-power solutions [4, 5] is to disable the circuit when the decoder detects no errors. However, this is only effective for codes with small value of t when considering their error distribution (see section 2.2.2 for details). Improvement can be made by shutting down the Chien search after the last error is found. This was done in [8] where central station telecommunication applications were targeted with the average power savings of 50% for $t = 1$, and the Chien search circuit itself accounted for 61% of the total power consumption. The drawback is that as the probability of early shutdown decreases with the increasing number of errors, so does the potential power reduction. Yet another low power strategy is the parallel Chien search which trades area cost for power [6].

2.2 RT-Level Power Management for Chien Search

This section begins with a power model for both conventional Chien search and the method of [8]. This model also serves as a vehicle for understanding the potential power efficiency of the proposed power-management method to be presented later.

2.2.1 Power Modeling for the Existing Methods

From figure 1.1 (solid-line portion), a conventional Chien search with error correcting capability of t bits includes t identical stages. Each of the stages corresponds to one of the R_i terms in (1.1). Therefore, the total average power is expressed as

$$P_C = \sum_{i=1}^t [P_{cgfm}(i) + P_{reg}(i) + P_{mux}(i) + P_{add}(i)] + P_{ckt} \quad (2.1)$$

where $P_{cgfm}(i)$, $P_{reg}(i)$, $P_{mux}(i)$, $P_{add}(i)$ and P_{ckt} represent the average power consumption due to Constant Galois-Field Multipliers (CGFMs), registers, multiplexors, adders and the clock-tree, respectively, with i being the dummy variable pointing to the i -th order term of (1.1).

The total average power depicted in (2.1) is valid only for a t -th order error locator polynomial. Since the actual order of a polynomial varies with the number of errors presented in a received message, P_C is not a fixed value. Nevertheless, the total average power over all received messages can be found by using standard error distribution model such as AWGN (additive white Gaussian noise), which is to be discussed later in Section 2.2.2.

On the other hand, we seek to provide power analysis for (2.1) at RT-level. This is possible by making two assumptions: a) the power dissipation of Chien search comes mainly from CGFMs and registers, and b) the CGFM and register for each i consume almost the same amount of power. The first assumption is valid as the power dominance of registers has been observed in similar applications [5]. The second assumption is based on the fact that all registers' average power consumption is close to one another. This is because the Chien search process is required to go through every single error position. Thus, each of the registers has its chance of holding every possible finite-field element. Although the instantaneous power consumption does depend on the finite-field element held by a particular register, its average power over one complete search cycle becomes virtually the same for all registers. When a register is holding finite-field values, the power consumption will be higher and can be modeled by the active power of a register (P_{ra}). When a register is not used because the received message contains fewer errors than t , it may hold a zero value. Even in this condition, however, the register will still consume power. We denote this power as the idle power of a register, which equals a constant η times P_{ra} where η is close to one (see section 2.4).

To gain a quantitative insight into the power savings of [8], let us consider a q -errors corrupted message word. For a (n, k, t) BCH decoder with no decoding failure, the value of q is no more than t . In other words, the Chien search may receive, from the previous stage, any q -th order error polynomial, where q may be smaller than t , the maximum

order of error polynomial that the Chien search is designed to receive. The q error bits are randomly distributed on the n -bits message word. Since the conventional Chien search always proceeds in one direction, i.e. from the most significant bit to the least significant bit, and it takes one clock cycle to progress from one bit to the next, we can introduce the concept of bit distance l ($1 \leq l \leq n-q+1$), as shown in figure 2.1, which defines the number of clock cycles between any two adjacent error bits, i.e., the x -th error bit and $(x+1)$ -th error bit, where x represents any arbitrary integer from 0 to $q-1$ (The 0-th error bit represents the beginning of a message.) We want to find the average of l , which is related to the average time for which the internal Chien search components operate. While the instantaneous power may differ with different lengths of l , we are interested in optimization of the average power. From the viewpoint of combinatorial analysis, all bits are usually assumed to have the same error rate. Therefore, for each value of l , it can be assigned the number of equally-probable error-patterns, $N_{l,x}$, which possess a bit distance l between the x -th and the $(x+1)$ -th error bit. $N_{l,x}$ is more easily determined if we define another variable b as the number of bit gaps between errors outside of x -th and $(x+1)$ -th errors (where $b = n - q + 1 - l$). We have

$$N_{l,x} = \binom{q-1+b}{b} \quad (2.2)$$

This indicates that $N_{l,x}$ is independent of x . Similarly, the mean of l is independent of x . Since [8] turns off the Chien search immediately after the final error is found, it is also important to find out the number of clock cycles (denoted by d) counted down from the final error bit. To derive the mean of both l and d from (2.2), we have (2.3) below.

$$\begin{aligned} \bar{l} &= \binom{n}{q}^{-1} \cdot \sum_{l=1}^{n-q+1} (l \cdot N_{l,x}) \\ \bar{d} &= \binom{n}{q}^{-1} \cdot \sum_{l=1}^{n-q+1} [(l-1) \cdot N_{l,q}] \end{aligned} \quad (2.3)$$

We realize that, for any given x , the sum of all $N_{l,x}$ equals all possible received message patterns with q error bits. This leads the relationship of (2.4).

Since [8] assumes an equal probability model where the likelihood of receiving q -errors corrupted frames is the same for any value of q , the average power savings of this existing method are given by (with the assumption of $\eta = 1$)

$$\bar{S}_{existing} = \frac{1}{t} \sum_{q=1}^t S_q = \frac{1}{t} \sum_{q=1}^t \frac{1 - \frac{q}{n}}{q+1} \approx \frac{1}{t} \sum_{q=1}^t \frac{1}{q+1}, \quad \text{if } n \gg t \quad (2.9)$$

However, using the AWGN model may result in unequal probabilities for q -errors corrupted frames, whose effect will be discussed in the section that follows.

2.2.2 The Proposed Method

In figure 1.1, not all CGFMs are required before the decoding reaches the final error-bit, and some of them can be made redundant and then disabled. This potentially allows additional power savings prior to the eventual power-down of the entire Chien search at the final error-bit. More specifically, (1.1) is rewritten as

$$\Lambda_t(X) = (1 + \beta_1 X)(1 + \beta_2 X) \dots (1 + \beta_t X) \quad (2.10)$$

If, for instance, all m -tuples except β_l have been found, then the only significant term will be the first one, meaning that only one CGFM is needed instead of all t CGFMs. The proposed method attempts to start saving power as soon as the first error is found by disabling the redundant CGFM with the method of clock-gating. More power can be saved with subsequent positive detections by taking out from (2.10) one polynomial factor at a time, which is equivalent to making the corresponding CGFM redundant. Finally, just prior to reaching the final error bit, most circuits except for one CGFM have been power-disabled. This represents a power-management strategy which simply rearranges operations that would otherwise consume extra power unnecessarily.

To model the proposed method, we only take into consideration the $P_{cgfm}(q)$ and $P_{reg}(q)$ in (2.1) for reasons already explained in section 2.2.1. In general, $P_{cgfm}(q)$ increases with q due to the higher circuit complexity of a finite-field element being multiplied by α^q . However, making these components constants in (2.11) (P_{cgfm} and P_{reg}) will only result in an underestimate of the average power savings for those components, since the higher q -th CGFMs and the heavier power consumers are also the first to be turned off according

to our proposed method. At RT-level where the details of circuit implementation are not available, this approximation is reasonable, particularly when the value of η is close to one, which indicates the dominance of registers' idle-mode power, as will be shown in section 2.4. Here we introduce a new variable v considering the fact that the q CGFMs are initially active for the q -th order error polynomial, but only v ($v = 1, \dots, q$) CGFMs will stay active as our proposed method progresses. If the clock gating is perfect, when v CGFMs are active, the total power (i.e., the power from both CGFMs and registers) is given by (refer to (2))

$$p(v) = v \cdot (P_{cgfm} + P_{reg}) \quad (2.11)$$

For n -bits and q -errors messages, if all q CGFMs are active initially and turned off one after another subsequently, the average power can be obtained from (2.7) and (2.11) as (for $n \gg 1$)

$$\begin{aligned} \bar{p}_{prop}(v) &= \frac{n+1}{n(v+1)} \cdot \sum_{q=1}^v p(q) \\ &\approx \frac{v}{2} \cdot (P_{cgfm} + P_{reg}) \end{aligned} \quad (2.12)$$

Combining (2.11) and (2.12) gives

$$\bar{p}_{prop}(q) \approx \frac{1}{2} p(q) \quad (2.13)$$

Based on the definition of $P_{reg}(q)$ (see section 2.2.1) and (2.11), the total power of the conventional Chien search (again, only including the first two terms in (2.1)) is approximated as

$$p_{conv}(q) = p(q) + (t-q)\eta P_{ra} \quad (2.14)$$

Unlike the conventional Chien search, the proposed method introduces q extra states, which means that the decoding time becomes $n+q$ (instead of n) clock cycles. Thus, the control logic needs to be modified, but with minimal power implication, as will be shown later in section 2.4. More importantly, the resulting Chien search circuit is clocked more often given the same number of input bits. To maintain the same output rate, the clock frequency can be increased by a factor of q/n . Since q is usually much smaller than n , the

extra power due to the q/n factor will not be significant compared to other power components, and an accurate model for this extra power is not necessary. For simplicity, it is assumed that the power consumption due to an extra state is the same as that for processing a normal error-free bit. Thus, the extra increase in power can simply be modeled by a factor of q/n . The issue of the proposed Chien search requiring different clock than that of the previous stage can be addressed by interfacing techniques or by modifying the proposed method to accept the same clock. Further discussions are beyond the scope of this thesis. With the equal-probability assumption, the average power consumption of the proposed method (again, normalized to the power of the conventional Chien search) can be written as

$$\bar{P}_{prop} = \frac{0.5 \times \sum_{q=1}^t \left[\left(1 + \frac{q}{n} \right) \cdot p(q) \right]}{\sum_{q=1}^t P_{conv}(q)} \quad (2.15)$$

where $p(q)$ and $p_{conv}(q)$ are given by (2.11) and (2.14), respectively. Generally speaking, (2.15) requires the detailed component power data from the circuit. However, the fact that the P_{ra} term is a dominant component in (2.11) allows a quick estimation of power savings by discarding other insignificant terms, leading to the final power savings of $\bar{S}_{prop} = 1 - \bar{P}_{prop}$ which is calculated approximately as

$$\bar{S}_{prop} \approx 1 - \left[\frac{6n}{3n + 2t + 1} \cdot \left(1 + \frac{t-1}{t+1} \cdot \eta \right) \right]^{-1} \quad (2.16)$$

The above expression generally promises much better results than (2.9). For instance, assuming $\eta = 1$, (2.16) produces the power savings of about 50% for $t = 1$ and 72% for $t = 9$, compared to only 21% given by (2.9) for $t = 9$ (assuming $n \gg t$).

With the AWGN model, if p_e represents the bit error rate, the word probability (p_w) of q -errors messages is given by

$$p_w(q, p_e) = \binom{n}{q} \cdot p_e^q \cdot (1 - p_e)^{n-q} \quad (2.17)$$

The average power consumption (denoted by $P_{AWGN}(t)$) of the proposed method can be modified by adjusting (2.15) as

$$P_{AWGN}(t) = \frac{\sum_{q=1}^t \left\{ \left(1 + \frac{q}{n}\right) \cdot p_w(q, p_e) \cdot p(q) \right\}}{2 \cdot \sum_{q=1}^t \{ p_w(q, p_e) \cdot p_{conv}(q) \}} \quad (2.18)$$

where $p(q)$, $p_{conv}(q)$ and $p_w(q, p_e)$ are given by (2.11), (2.14) and (2.17), respectively. To provide a fair comparison, a higher- t circuit is given a lower signal-to-noise ratio (SNR) such that the bit error rate (p_e) becomes a function of t as the probabilities of receiving uncorrectable frames are assumed to be equal for all values of t . For instance, according to (2.17), with the assumption of 0.1% of uncorrectable frames, 95% of received frames are error-free for $t = 1$ and the number drops below 50% for $t = 4$. Figure 2.2 shows the comparison of the AWGN (shown as dotted lines) and equal-probability (shown as solid lines) models during the error correction for a variety of percentages of uncorrectable frames. It can be seen from figure 2.2 that the AWGN model generally results in more power savings. However, at 5% of uncorrected frames, the power savings for both models are very close to each other (only a few percent different). Thus, the equal-probability model given by (2.9) and (2.16) can be used to provide the worst-case power estimation when the decoder is driven close to the edge of its error correcting capability.

To support the above-mentioned power management where the CGFMs are disabled with each detected error, we need a mechanism of making the CGFMs redundant. This mechanism is the *polynomial order reduction* (POR) which requires a modified Chien search, as illustrated in the dotted-line portion of figure 1.1 where an extra state is inserted for each detected error-bit. During the extra state, new values are written back to the R_i . The spared CGFMs are then disabled, one at a time, via clock gating.

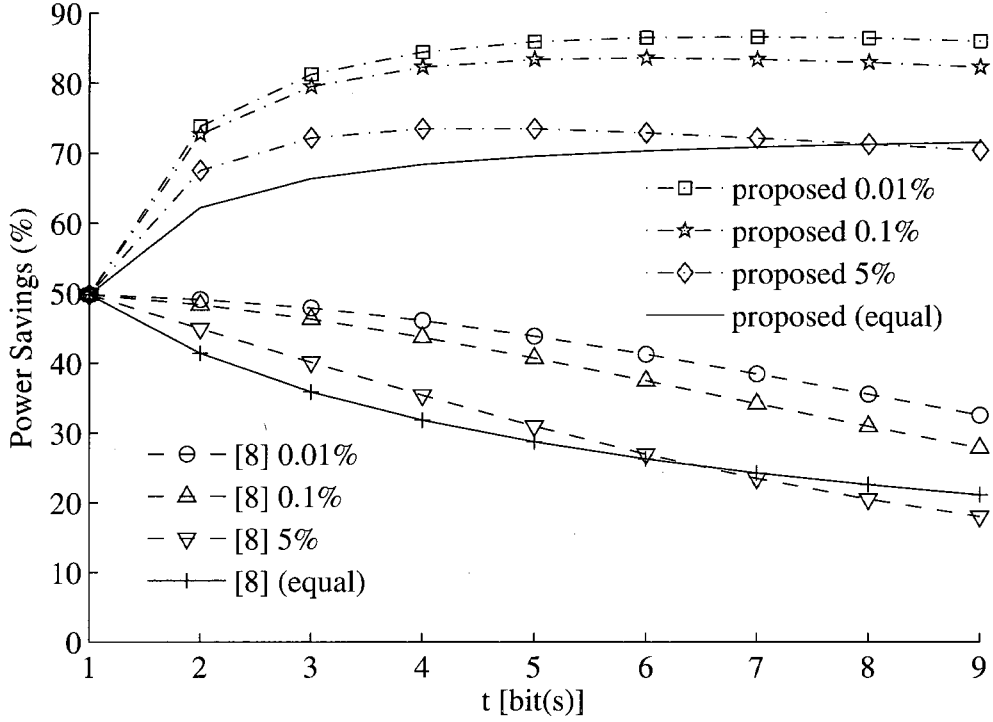


Figure 2.2: Comparison of Power Savings for Various Percentages of Un-correctable Frames

2.2.3 A Polynomial Order Reduction Algorithm

Since the proposed method seeks to reduce the order of $\Lambda_u(R_i)$, the variable i no longer represents a fixed list. To distinguish this change, a variable w ($w = 1, 2, \dots, u$) is introduced where u is the current order of the error polynomial, leading to the following:

$$\begin{cases} \Lambda_u(R_w) = 1 + \sum_{w=1}^u R_w \\ R_w = \sigma_w(\varepsilon)X^w \end{cases} \quad (2.19)$$

where the coefficients of X are functions of ε , and ε ($\varepsilon = 1, 2, \dots, t$) corresponds to the number of polynomial order reductions already performed. For instance, $\sigma_w(0)$ represents the initial value of coefficients when $u = t$, and $\sigma_w(t)$ represents zero when $u = 0$ (i.e., all roots are factored out). Assuming that the Chien search finds an error position β , it is always possible to break the expression of $\sigma_w(\varepsilon)$ into two groups of terms:

$$\sigma_w(\varepsilon) = \gamma_w(\varepsilon) + \rho_w(\varepsilon) \quad (2.20)$$

where $\gamma_w(\varepsilon)$ is the sum of all β -carrying product terms, and $\rho_w(\varepsilon)$ the sum of all product terms that are without β . Since $\gamma_w(\varepsilon)$ always contains a common factor β , what remains (after extracting the β) is the sum of product terms from $\binom{u-1}{w-1}$ combinations of any error positions except β , which is exactly the same as $\rho_{w-1}(\varepsilon)$ that also carries the sum of product terms from same combinations. Therefore, we have

$$\gamma_w(\varepsilon) = \beta \rho_{w-1}(\varepsilon) \quad (2.21)$$

The goal of POR is to find the update values R_w' for R_w . To this end, the coefficients of X , i.e., $\sigma_w(\varepsilon+1)$, are required according to (2.19). Also, the order of the polynomial is updated by the process from u to $u' = u-1$. The value of $\sigma_w(\varepsilon+1)$ is the sum of product terms from $\binom{u'}{w}$ combinations, and is given by

$$\sigma_w(\varepsilon+1) = \gamma_{w+1}(\varepsilon) \cdot \frac{1}{\beta} \quad (2.22)$$

From (2.19) and (2.22), we can express the new update value R_w' for R_w as

$$\begin{aligned} R_w' &= \sigma_w(\varepsilon+1) \cdot \left(\frac{1}{\beta}\right)^w \\ &= \gamma_{w+1}(\varepsilon) \cdot \left(\frac{1}{\beta}\right)^{w+1} + 2 \cdot \sum_{\lambda=w+1}^{u-1} \rho_\lambda(\varepsilon) \cdot \left(\frac{1}{\beta}\right)^\lambda \end{aligned} \quad (2.23)$$

Note that the second summation term in (2.23), which is intentionally inserted, equals to zero because it is performing an operation of adding two identical finite-field elements. This summation term can also be rewritten as

$$\begin{aligned} &2 \cdot \sum_{\lambda=w+1}^{u-1} \rho_\lambda(\varepsilon) \cdot \left(\frac{1}{\beta}\right)^\lambda \\ &= \sum_{\lambda=w+2}^u \rho_{\lambda-1}(\varepsilon) \cdot \left(\frac{1}{\beta}\right)^{\lambda-1} + \sum_{\lambda=w+1}^{u-1} \rho_\lambda(\varepsilon) \cdot \left(\frac{1}{\beta}\right)^\lambda + \rho_u \cdot \left(\frac{1}{\beta}\right)^u \end{aligned} \quad (2.24)$$

where the last term, which is again intentionally inserted, equals to zero. This is because ρ is the sum of all product terms without β according to the definition in (2.20), and ρ_u

belongs to the highest order coefficient where there is no product term that is without β . Substituting (2.21) into (2.24) leads to

$$\begin{aligned}
& 2 \cdot \sum_{\lambda=w+1}^{u-1} \rho_{\lambda}(\varepsilon) \cdot \left(\frac{1}{\beta}\right)^{\lambda} \\
&= \sum_{\lambda=w+2}^u \gamma_{\lambda}(\varepsilon) \cdot \left(\frac{1}{\beta}\right)^{\lambda} + \sum_{\lambda=w+1}^u \rho_{\lambda}(\varepsilon) \cdot \left(\frac{1}{\beta}\right)^{\lambda}
\end{aligned} \tag{2.25}$$

By combining (2.19), (2.20), (2.23) and (2.25), we have the following final result:

$$\begin{aligned}
R'_w &= \sum_{\lambda=w+1}^u [\gamma_{\lambda}(\varepsilon) + \rho_{\lambda}(\varepsilon)] \cdot \left(\frac{1}{\beta}\right)^{\lambda} \\
&= \sum_{\lambda=w+1}^u \sigma_{\lambda}(\varepsilon) \cdot \left(\frac{1}{\beta}\right)^{\lambda} \\
&= \sum_{\lambda=w+1}^u R_{\lambda}
\end{aligned} \tag{2.26}$$

which means that the update value for the w -th order register is simply the sum of all higher-order register values. The highest-order register should be updated with zero and become redundant. The order of $\Lambda_u(R_i)$ (2.19) decreases with the process by one at a time, eventually bringing the polynomial to zero after all errors are detected. Figure 1.1 illustrates the proposed Chien search architecture with the POR (dotted-line portion).

Alternatively, the order of (2.19) could be reduced by a regular polynomial division which requires many shift operations and additions. Instead, the POR utilizes existing features and steps of the conventional Chien search for power management purpose. For instance, there is no need for additional shifter as the searching process itself doubles as a shifter. Theoretically, there is no demand for any additional adders either, as (1.1) is satisfied by (2.26) automatically.

2.3 Circuit Implementation for the Proposed Low Power Chien Search

This section deals with some circuit design problems so that the power and area overhead can be kept at a minimum level.

2.3.1 Adder for POR

Although POR could be potentially implemented by using the existing adder without any area overhead, such an attempt would still require additional outputs from intermediate summation terms in (1.1). The dotted line portion of figure 1.1 indicates that POR needs to provide outputs for all $t-1$ CGFM stages (the t -th stage is always cleared, and thus ignored). When $t = 9$, this corresponds to 8-bit per CGFM with 8 CGFMs worth of outputs, compared to only one output that is needed in the original adder. Since this adder circuit must be driven for every cycle, higher power consumption would be expected. To minimize the negative power impact, an additional adder is used. Figure 2.3 shows the block diagram for the new POR, which has outputs for driving the multiplexers of other CGFM stages. Figure 2.4 illustrates the adder circuit for an individual bit. While this causes an increase in area, the new adder is gated by a low-load input buffer, which only activates the circuit for t out of n clock cycles while leaving it idle and minimally powered for the rest of time. The process of POR will gradually clear the registers of all CGFMs, which also bring about savings in adder circuits since its components now spend additional time being idle. Since these savings depend on specific adder implementations, they are not included in our RT-level analysis. Considering the fact that n is usually much greater than t , the extra power consumption will be minimal.

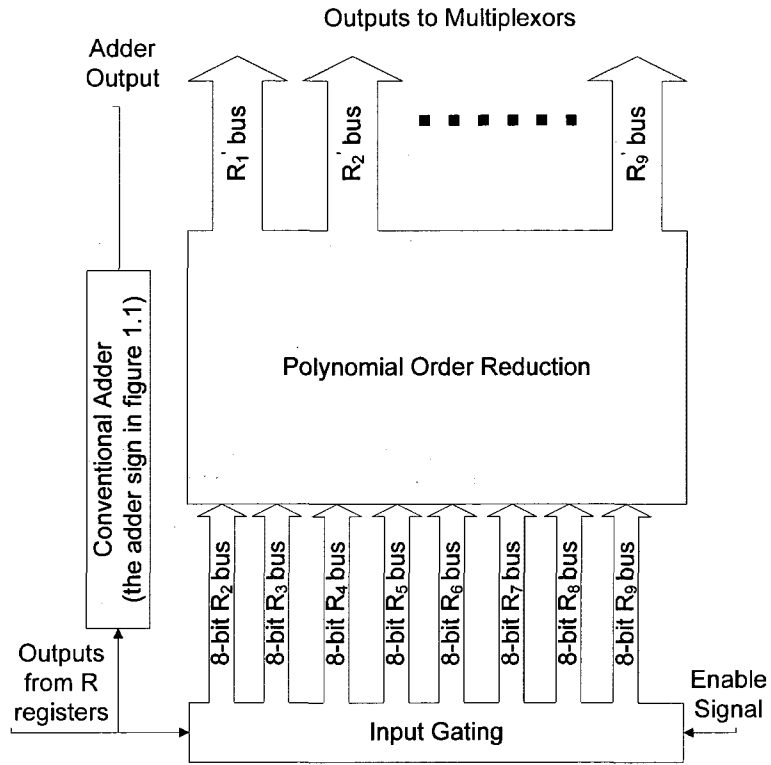


Figure 2.3: The Block Diagram of POR

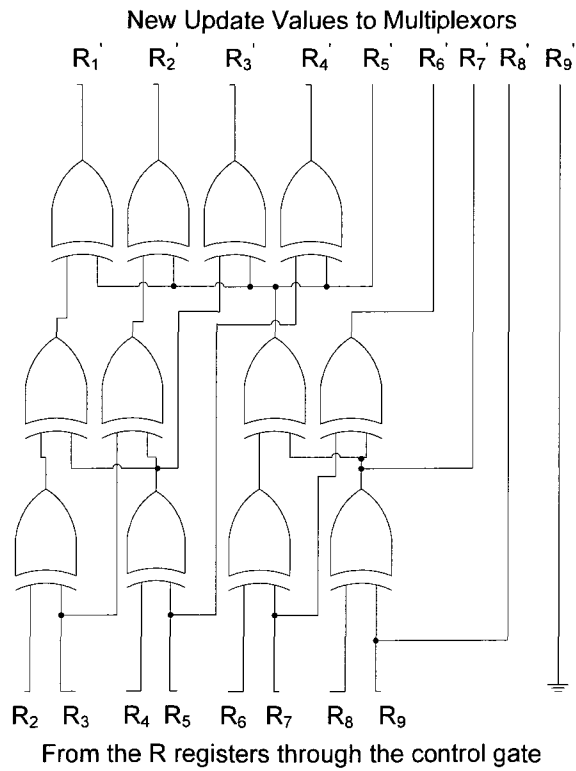


Figure 2.4: One Bit of the POR

2.3.2 Multiplexer and Clock-Gating

As can be seen in figure 1.1, each CGFM is loaded with the updated value of a register controlled by a 3-to-1 multiplexor which determines whether an initial σ value, new σ (from POR), or output of CGFM is to be transferred. The redundant CGFM registers are clock-disabled from zero-detectors at outputs of registers via Verilog descriptions for RT-level clock gating, shown in figure 2.5.

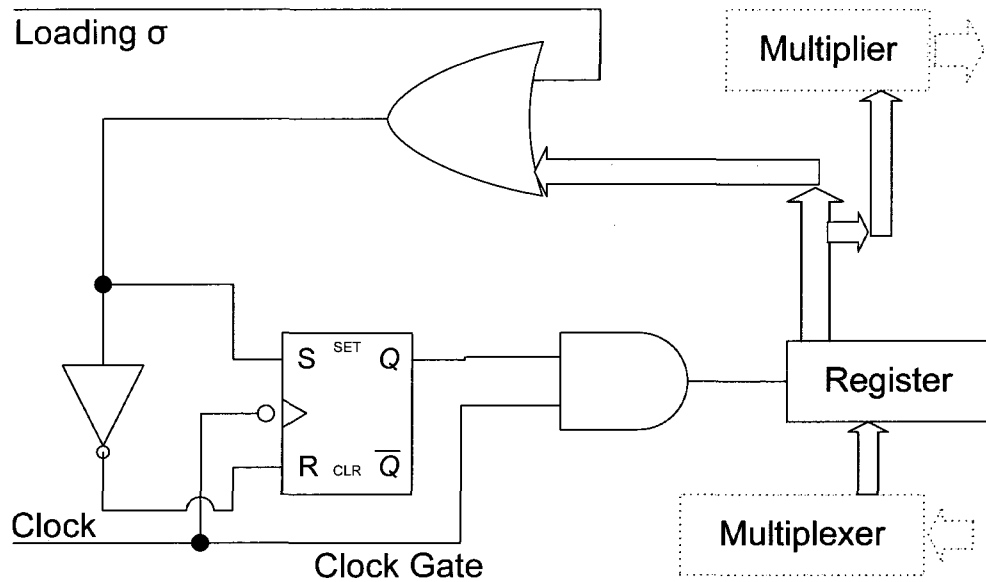


Figure 2.5: The Clock Gating Circuit

2.3.3 Layout and Area Overhead

To verify the effectiveness of our method, a (255, 187, 9) BCH code was implemented for both the conventional and proposed Chien search with CMOS $0.18\mu\text{m}$ technology. *Synopsys Design Vision* and *Cadence Encounter* were used for logic synthesis and layout design, respectively. Overall, the area of the proposed Chien search circuit increases by 24% to 0.041mm^2 due to the POR, compared to 0.033mm^2 for the conventional circuit and 106% area penalty for the two-fold parallel circuit [6]. The core layout is illustrated in figure 2.6.

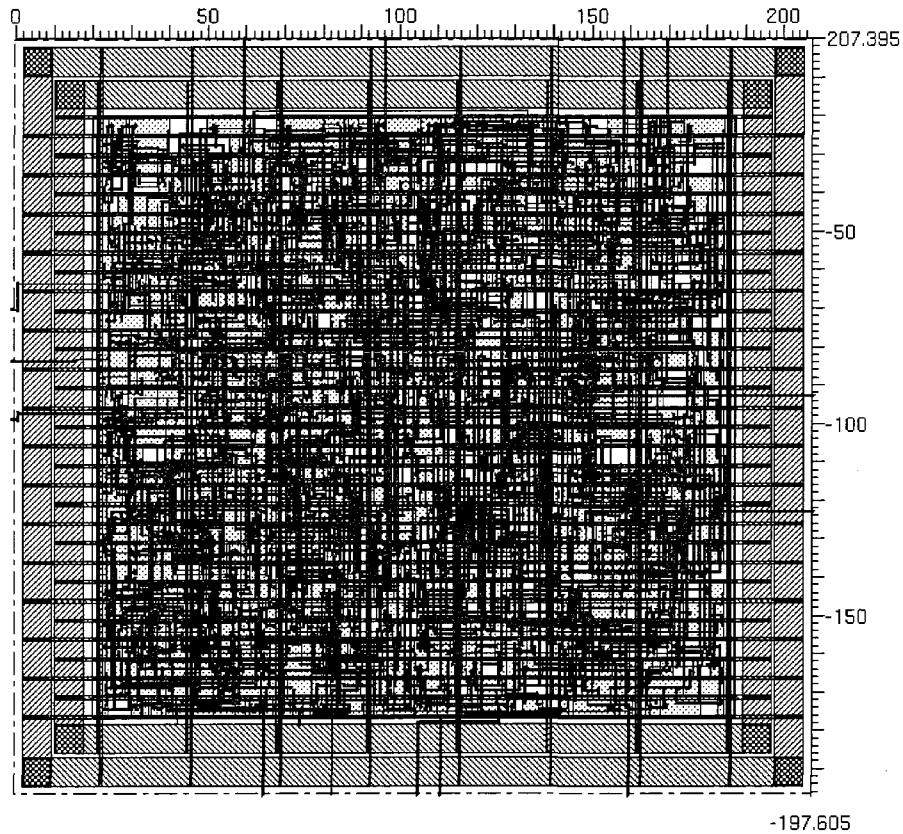


Figure 2.6: Core Layout of the Proposed Chien Search

2.3.4 Parallel Chien Search and Power Overhead

By employing similar approaches from [5], two-fold parallelism may be achieved using the proposed method with a pair of PORs, multiplexors and CGFMs, for each polynomial term of (1.1), resulting in nearly the same area overhead of 24%. Since the circuit searches twice as fast with the same number of extra states, the equal-probability power overhead (on average) is approximately t/n (see section 2.2.2), which is about 4% (instead of 2%) for the non-parallel (255, 187, 9) code. BCH decoder interleaving [5] is another way of achieving high throughput with multiple independent Chien search circuits. In this case, the proposed method can be applied directly with the same 24% area overhead and the power overhead of roughly $t/2n$ (or 2% for the above code), independent of the level of interleaving.

2.4 Simulation Results

Both conventional and proposed Chien search circuits for the (255, 187, 9) BCH code were simulated with volumes of random q -errors ($q = 1 \sim 9$) messages generated by MATLAB and sent to Device-Under-Tests via Verilog testbench. The resulting switching activities are collected for all internal circuit nodes. The power analysis uses Cadence Encounter tool with extracted RC information from layout for high accuracy.

The power consumption due to different components for both the conventional and proposed methods is shown in tables 2.1 and 2.2, respectively. The $\mu\text{W}/\text{MHz}$ values for our proposed circuit are effective values adjusted to include the effect of extra clock cycles described in section 2.2.2. It should be mentioned that only CGFMs and registers were used for deriving (2.16), and our results show that with the equal-probability model, combination of CGFMs and registers consumes only 56% of total power in the conventional Chien search. The power savings for the proposed search circuit are also plotted in figure 2.7 (top curve), from which we see 60% power reduction at $t = 9$.

Table 2.1: Component Power for the Conventional Circuit

Error-bits (q)	CGFMs & Registers ($\mu\text{W}/\text{MHz}$)	MUXs ($\mu\text{W}/\text{MHz}$)	Adders ($\mu\text{W}/\text{MHz}$)	Clock Tree ($\mu\text{W}/\text{MHz}$)
1	3.98	0.14	0.62	2.49
2	4.42	0.38	0.59	2.49
3	4.90	0.62	0.86	2.49
4	5.39	0.87	0.84	2.49
5	5.95	1.14	1.29	2.49
6	6.53	1.48	1.26	2.49
7	7.30	1.85	1.44	2.49
8	7.96	2.20	1.41	2.49
9	8.65	2.59	1.61	2.49

Table 2.2: Component Power for the Proposed Circuit

Error- bits (q)	CGFMs & Registers (μW/MHz)	MUXs & Zero- Detector (μW/MHz)	Adders (μW/MHz)	Clock Tree (μW/MHz)
1	0.51	0.14	0.43	2.42
2	0.95	0.30	0.55	2.53
3	1.39	0.47	0.67	2.64
4	1.85	0.66	0.75	2.76
5	2.39	0.89	0.91	2.88
6	2.94	1.13	1.03	3.00
7	3.48	1.36	1.18	3.11
8	4.04	1.60	1.34	3.22
9	4.67	1.89	1.51	3.34

In comparison with (2.16) (refer to the top solid curve in figure 2.2), it can be seen that the trend generally matches except for a negative discrepancy which gradually expands to 12% at $t = 9$. This is due to non-unity η (0.62) and the neglected terms in deriving (2.16). At RT-level, η is not available as it depends on particular process and circuit implementation. However, (2.16) still provides good estimation.

While simulations were not performed with the AWGN model, they would involve only a change of error distribution with a higher percentage of frames being received with fewer errors. Therefore, CGFMs and registers implemented for the higher order terms of (1.1) would spend a higher percentage of time being idle. It can be seen from figure 2.2 that the AWGN promises more power savings for the CGFMs and registers. Similarly, a relative improvement can be expected over the solid line curve of figure 2.7 for the performance of the proposed circuit under AWGN. This also applies to the total power savings (i.e., the curve with circles in figure 2.7).

In addition to power reduction from CGFMs and registers, there are also moderate power savings of 21% in the multiplexors and adders. However, the power in clock-tree (including the power for clock buffers) increases by 16%, due to the power overhead from clock-gating latches and internal clock buffers. The control logic for the proposed

search circuit only consumes small amount of power. Mainly due to the increased power consumption in clock-tree, the overall power savings of the proposed method decrease to 34% (see the curve with circles in figure 2.7). The clock buffer, which contributes much of the additional power, is process and implementation dependent. Therefore, it is not relevant for our RT-level investigation.

Although the method of [8] claimed a 50% power reduction (including the clock-tree power) for $t = 1$, the actual power savings would be much less. This is because of the fact that there is a minimum clock-tree/clock-buffer power of $0.92\mu\text{W}/\text{MHz}$, which accounts for about 6%~13% of the total power, depending on the number of error-bits. This power is also orders of magnitude higher than the leakage power which is hence ignored. The recalculated power savings of [8] are shown in figure 2.7 along with the results from the proposed method for comparison. For a low value of $t (\leq 3)$, according to section 2.2.2, the power efficiency is not as important because most frames are received without errors, and the method of [4] or [5] can be used for power improvement. When $t = 4$ for which the percentage of error-free frames drops to below 50%, the proposed method starts to outperform [8] and its advantage becomes increasingly obvious with further rise of t . At $t = 9$, in particular, the power savings of the proposed circuit nearly double that of [8]. It should be mentioned that the equal-probability results of figure 2.7 will trace closely to that of the AWGN model (refer to section 2.2.2) at an uncorrectable frame rate of 5%, but such a high rate is not typical in practical designs. Therefore, figure 2.7 just represents a worst-case scenario. More power savings can be expected with the AWGN. It should also be noted that the curves for other values of t (i.e., $t < 9$) in figure 2.7 are extrapolated based on results for the (255, 187, 9) BCH code which was implemented in this work. The basic idea behind this extrapolation is that changing the value of t mainly involves adding or removing the CGFM and adder blocks, which is a fairly linear process.

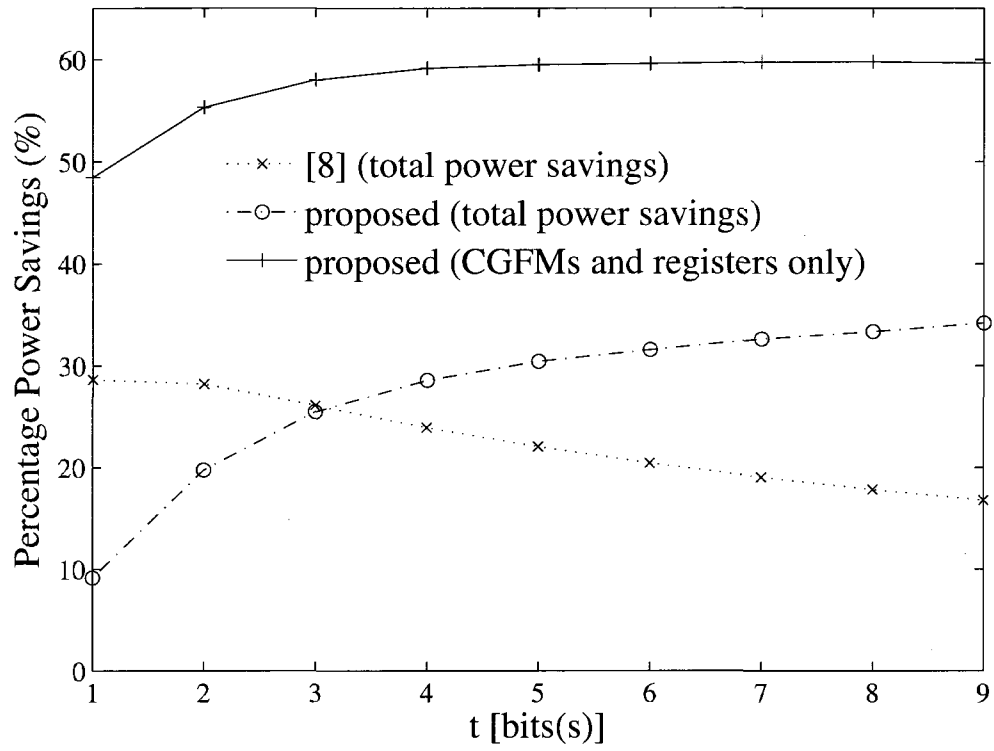


Figure 2.7: Power Savings for $(255, k, t)$ BCH codes

Chapter 3 DIFFERENTIAL-DRIVE CMOS BRIDGE RECTIFIER

From this chapter onward, the theme changes to the subject of low power rectifier design. It is reminded from chapter 1 that a rectifier is the analog front-end and the sole source of power for a passive RFID tag. Therefore, the power optimization of this vital circuit carries the utmost importance. Our particular interest is the differential-drive CMOS full-wave bridge rectifier, which we will fully explore in the subsequent chapters. In this chapter, some salient features are pointed out with the help of the analytical analysis.

3.1 Differential-Drive CMOS Half Bridge and Full Bridge Circuits

In order to find the power related optimization functions, a single stage of differential-drive CMOS bridge rectifier is studied first. Both the half bridge and the full bridge are, respectively, given in the left and the right side of figure 3.1.

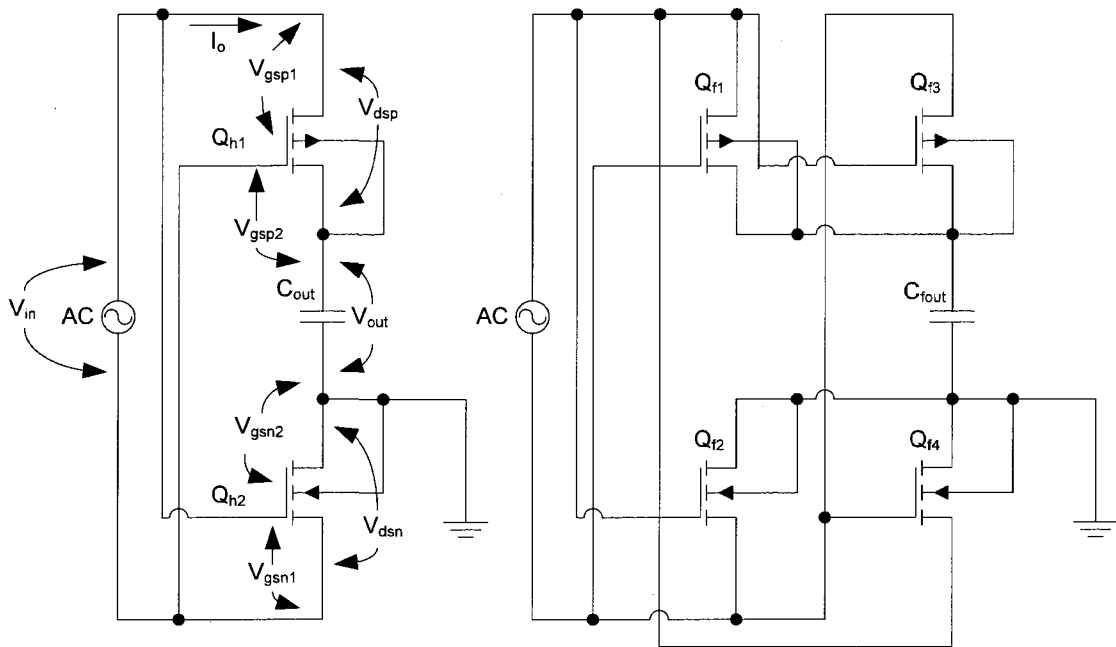


Figure 3.1: Differential-Drive CMOS Half Bridge Rectifier (Left), Full Bridge (Right)

The target circuit for optimization is the circuit on the right side of figure 3.1. It is known as differential-drive CMOS rectifier [19]. When power conversion efficiency (PCE) is defined as the ratio between the input power to the rectifier and the output power to the

load, A Sasaki et al [19] has reported a PCE of 66% when the input power level is at -12dBm. This is almost two times as much as then the state-of-the-art 37% reported by H. Nakamoto [23] with the method of IVC (see chapter 1). However, it is not clear whether the result has reached the maximum PCE achievable for the kind of circuit, and whether the high PCE can be maintained for other operating conditions such as higher load current and lower input power. Moreover, the circuit was a single stage tested with a RF power generator. A practical RFID tag demands higher output voltage from the rectifier. This requires a multiple-stages design that couples to a real antenna. S. Mandal [21] has shown a three-stage design for the same type of circuit only to achieve a PCE of 23%. Without detail analysis, it may be far too easy to bridge the two studies and conclude that body effect in the multiple-stages design caused the PCE degradation. This, however, is not our position as we investigate the true reason behind S. Mandal's [21] inferior PCE performance. In fact, the following chapters will show that high PCE performance is equally available for multi-stages design. In addition, the method for coupling the high PCE circuit to a real antenna will be revealed and such has never been being studied before. So far, all related literatures assume linear impedance model, which is only applicable when the rectifier of concern functions with low PCE [21].

3.2 DC Approximation for Quasi-Steady State

By taking either the half bridge or the full bridge circuit in figure 3.1 and then powering it by a steady high frequency AC voltage, the output voltage of the rectifier will rise from zero but eventually reaching a steady level, when the rectifier's input current and its output current to the load balances. At that point, there is no net change in voltage across the output capacitor. Furthermore, the output capacitors C_{out} and C_{fout} in figure 3.1 are usually very big relative to the AC input frequency. This means the capacitors can effectively be treated as AC short circuits. Under this circumstance, the small instantaneous change in output voltage due to the current from each AC cycle may be ignored. This creates a quasi-steady state for the output capacitor, where it can be approximated by a fixed voltage source.

The quasi-steady state allows DC analysis to be applied to the complete rectifier circuit with all parasitic capacitances being ignored by default. Returning our focus to the full-

bridge circuit of figure 3.1, we realize that Q_{f1} and Q_{f2} conduct while Q_{f3} and Q_{f4} turn off when the input voltage is above the output voltage. Likewise, when the input voltage becomes negative and has an absolute magnitude above the output voltage, Q_{f3} and Q_{f4} turn on while Q_{f1} and Q_{f2} shut off. If Q_{f1} has the same transistor parameters as Q_{f3} , and likewise for Q_{f2} and Q_{f4} , the current through the bridge will be identical during the positive half and the negative half cycle for a sinusoidal input. Therefore, the only difference between the half bridge and the full bridge is that the half bridge conducts for half of the AC cycle and drawing half as much current as the full bridge. For further simplicity, we only need to analyze a half bridge circuit.

On the left side of figure 3.1, the half bridge is defined with V_{in} , V_{out} , I_o , V_{dsn} , V_{gsn1} , V_{gsn2} , V_{dsp} , V_{gsp1} and V_{gsp2} . They represent input voltage, output voltage, output current; drain-to-source, gate-to-source and the gate-to-drain voltages for the N-type transistor; drain-to-source, gate-to-source and the gate-to-drain voltages for the P-type transistor, respectively. The gate-to-drain voltages for the N-type and P-type transistors are named as V_{gsn2} and V_{gsp2} because typical CMOS process has completely symmetrical transistor devices where drain and source are identical. So, when the rectifier is in operation, the effective gate-to-source voltages depends on the input and output voltages. We may define the operating V_{gsn} and V_{gsp} as below.

$$\begin{aligned}
 V_{gsn} &= \begin{cases} V_{gsn1} & , V_{in} \geq V_{out} \\ V_{gsn2} & , V_{in} < V_{out} \end{cases} \\
 V_{gsp} &= \begin{cases} |V_{gsp1}| & , V_{in} \geq V_{out} \\ |V_{gsp2}| & , V_{in} < V_{out} \end{cases}
 \end{aligned} \tag{3.1}$$

3.3 AC model for the Half Bridge Rectifier

Real transistors have parasitic capacitances and AC current will go through those capacitors. To model the AC current, one capacitor can be put across each pair of terminals of the transistors in figure 3.1. The capacitors, so defined, are C_{gsp1} , C_{gsp2} , C_{gbp} , C_{gsn1} , C_{gsn2} and C_{gbn} . They represent the gate-to-source, gate-to-drain and the gate-to-bulk capacitances for the P-type and N-type transistors, respectively. Figure 3.2 illustrates the half bridge AC equivalent circuits with these capacitances.

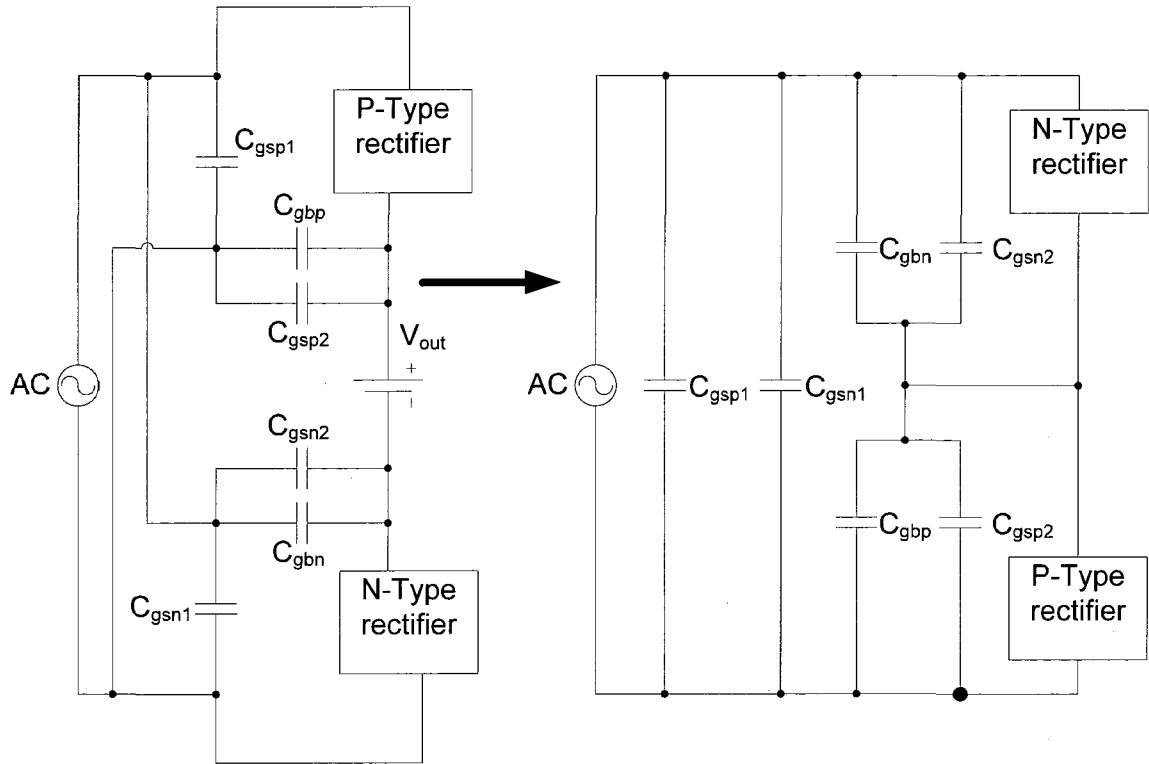


Figure 3.2: AC Equivalent Circuit for the Half Bridge

Inside the P-type and N-type rectifier boxes are the DC equivalent circuits of the transistors. As seen in figure 3.2, after some components rearrangement and ignoring V_{out} for the AC analysis, the AC equivalent circuit on the right side is obtained. In general, all capacitances are non-linear and their values depend on the biasing level. When the transistors are in cut-off region, C_{gsn1} , C_{gsn2} , C_{gsp1} and C_{gsp2} are small and they consist of mainly gate overlap capacitances. The C_{gbn} and C_{gbp} are the dominant capacitances in this case. When the transistor turns on and a strong inversion layer forms, C_{gbn} and C_{gbp} diminish but the capacitances of C_{gsn1} , C_{gsn2} , C_{gsp1} and C_{gsp2} increase [27]. Since one change in capacitance tends to cancel that of the other, the capacitive network can be roughly approximated by a fixed-value capacitor. Ideally, the capacitor is a purely reactive component, which does not consume any power. So, generally speaking, although the circuit shown in figure 3.1 is only fully characterized with AC and DC analysis, DC analysis can be applied first for gaining the most important insights. The AC effect still needs to be considered mainly because of the non-ideal properties of the capacitor. For instance the high resistivity of the poly-silicon gates and the substrate loss will manifest themselves as loss, which will be discussed in chapter 4.

3.4 DC Analysis for the Current through the Half Bridge Rectifier

The use of the quasi-steady model of the output capacitor (figure 3.1) in DC analysis allows an estimation for both the rectifier's input and output power by simply observing the properties of the output current I_o . The power estimation aspects are investigated in chapter 4. For this section, the output current I_o is studied first using level one SPICE model. The process involves zoning of the transistors' DC operating curve into multiple regions. This offers not only an understanding of I_o but useful insights towards the rectifier's operation. Prior to the delineation, an important parameter g^2 needs to be defined. This g^2 is proportional to the ratio of the channel-widths for the N-type and the P-type transistors and it is given by (3.2).

$$g^2 = \frac{\mu_n W_n}{\mu_p W_p} = \frac{\beta_n}{\beta_p} \quad (3.2)$$

In (3.2), μ_n , μ_p , W_n , W_p , β_n and β_p are the electron mobility parameters, the widths and the ratio-metric parameters for N and P type transistors, respectively. These parameters are related by (3.3),

$$\beta_n = \mu_n C_{ox} \frac{W_n}{L_n}, \quad \beta_p = \mu_p C_{ox} \frac{W_p}{L_p} \quad (3.3)$$

where C_{ox} , L_n and L_p represents the MOS effective capacitance density and the lengths for the N-type and P-type transistors, respectively. Throughout this paper, the length parameters are assumed to be the same and are, therefore, ignored. It is reminded that the transistors are completely symmetrical so that drain and source may be referred to interchangeably. The rectifier's I_o is now separated into six regions as the following.

3.4.1 Forward Conduction Region

According to the half bridge circuit in figure 3.1, the current I_{fcr} can be defined as a special case of I_o when $V_{in} \geq V_{out}$. Under this condition, $V_{in} = V_{gsp} = V_{gsn}$. It is observed that $V_{in} - V_{out} = V_{dsn} + V_{dsp}$ and therefore, $[V_{dsp}, V_{dsn}] < V_{in} - V_{out}$. If $V_{out} \geq V_{thp}$, the transistors will always operate in triode mode with the following current equations.

$$I_{fer} = \beta_p \left[(V_{in} - V_{thp}) \cdot V_{dsp} - \frac{1}{2} V_{dsp}^2 \right] \quad (3.4)$$

The unknown V_{dsp} in (3.4) is simply solved by equating the current to that of the N-type transistor using equations in (3.5)

$$I_{fer} = \beta_n \left[(V_{in} - V_{thn}) \cdot V_{dsn} - \frac{1}{2} V_{dsn}^2 \right] \quad (3.5)$$

$$V_{dsn} = V_{in} - V_{out} - V_{dsp}$$

It is noticed that the bulk connections are intentionally connected to the drain terminals, such that body effect will decrease the threshold voltages for a slightly higher forward current. The effect is, however, small and (3.4) and (3.5) still provides good estimations.

When this rectifier's forward region operation is compared with the Dickson multiplier type circuits, an advantage is seen. For instance, a transistor in the Dickson multiplier will only turn on when $V_{in} - V_{out} \geq V_{th}$. The differential-drive type rectifier will turn on when $V_{in} \geq V_{th}$. The latter is due to the fact that V_{in} is connected fully across V_{gsn} and V_{gsp} . This means differential-drive rectifier can potentially work at lower input voltage level.

3.4.2 Reversed Triode Conduction

This is when $V_{thp} \leq V_{in} < V_{out}$ and the current starts flowing in the reversed direction and V_{dsp} becomes negative. During this condition, $V_{gsn} = V_{out} + V_{dsp}$, $V_{gsp} = V_{in} - V_{dsp}$ and $||V_{dsp}, V_{dsn}|| < V_{out} - V_{in}$ such that transistors remain in triode mode and (3.4) still applies.

The transistors' bulk and source terminals are now effectively merged with negligible body effect. If the bulk terminals of figure 3.1 are connected otherwise, the reverse leakage and subthreshold current will be increased, leading to undesirable power loss.

The fact that $V_{gsn} = V_{out} + V_{dsp}$ and $V_{gsp} = V_{in} - V_{dsp}$ means that both of these voltages are decreasing quantities because $V_{dsp} \leq V_{in}$. On the contrary, during the reverse modes, V_{gsn} and V_{gsp} for the transistors in the Dickson type multiplier are always constantly biased. Figure 3.3 shows an example of one transistor in its forward and reverse mode. A constant reverse bias is highly disadvantageous for PCE because it significantly raises the subthreshold current, which is exponentially related to V_{gs} . (see section 3.4.5)

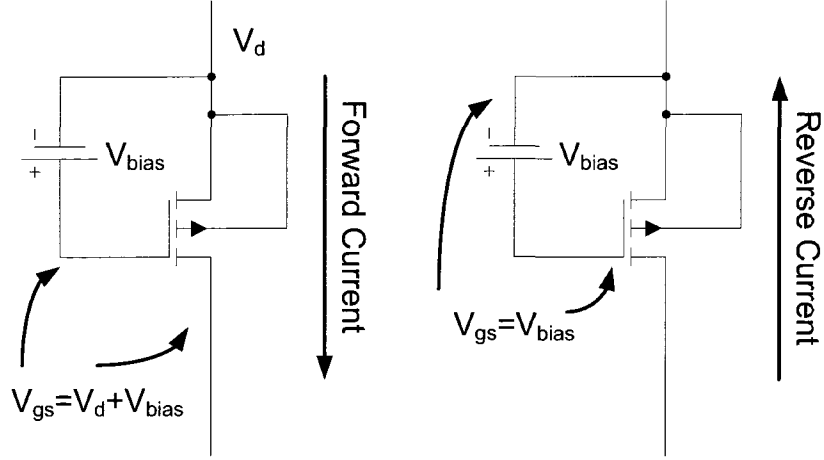


Figure 3.3: A Forwardly (Left) and a Reversely (Right) Biased MOS in Dickson Multiplier

3.4.3 Partial Saturation

The P-type transistor enters saturation region when $V_{thn} \leq V_{in} < V_{thp}$ while the N-type transistor stays in triode mode. The current during this condition is defined as the partial saturation current I_{ps} , which, for the P-type transistor, is described by (3.6).

$$I_{ps} = -\frac{1}{2} \beta_p (V_{in} - V_{thp} - V_{dsp})^2 \quad (3.6)$$

3.4.4 Full Saturation

If we define

$$v = V_{thn} + V_{thp} - V_{out} \quad (3.7)$$

, all transistors enter the saturation region when $v \leq V_{in} < V_{thn}$.

The current in this region is defined as the saturation current I_{fs} , which has an absolute magnitude that decreases with V_{in} according to (3.8).

$$I_{fs} = -\frac{1}{2} \cdot \frac{g^2}{(1+g)^2} \cdot \beta_p \cdot (V_{in} - v)^2 \quad (3.8)$$

3.4.5 Subthreshold Conduction

When V_{in} just become smaller than v , both V_{gsn} and V_{gsp} are still forwardly biased and the transistors are in the subthreshold mode of operation. The subthreshold current I_{sc} is described by (3.9)

$$I_{sc} = -\tau_p \beta_p e^{\frac{V_{gsp} - V_{sbp}}{nV_T}} \cdot \left(1 - e^{\frac{V_{dsp}}{V_T}} \right) \cdot \left(1 - \lambda_{subp} V_{dsp} \right) \quad (3.9)$$

where V_{sbp} is the source to bulk voltage, λ_{subp} is the subthreshold channel modulation parameter for the P-type transistor and τ_p is defined by (3.10) as below.

$$\tau_p = \frac{I_{sop}}{\mu_p C_{ox}} \quad (3.10)$$

In (3.10), I_{sop} is the subthreshold saturation current for the P-type transistor.

3.4.6 Reverse Leakage

When the N-type transistor is much larger than the P-type transistor, V_{dsp} becomes much greater than V_{dsn} and the former approaches the value of V_{out} , as V_{in} approaches zero, leading to a V_{gsn} that is approximately equal to zero, while V_{gsp} is approximately equal to V_{out} . Therefore, the N-type transistor will enter reverse leakage mode when $V_{in} < 0$. Likewise, when P-type transistor is much larger, the P-type transistor will first enter reverse leakage mode when $V_{in} < 0$. On the other hand, when both the N-type and the P-type transistors were sized such that V_{dsp} equals V_{dsn} , both transistors will not enter reverse leakage mode until $V_{in} < -V_{out}$. So, it is apparent that transistors will enter the reverse leakage mode anywhere when $-V_{out} \leq V_{in} \leq 0$, depending on how the individual transistor is sized. Under reverse leakage, the transistor current I_{rl} no longer depends on V_{gs} but it still depends on V_{sb} as in (3.11).

$$I_{rl} = -\tau_p \beta_p e^{\frac{V_{sbp}}{nV_T}} \cdot \left(1 - e^{\frac{V_{dsp}}{V_T}} \right) \cdot \left(1 - \lambda_{subp} V_{dsp} \right) \quad (3.11)$$

Chapter 4 POWER EFFICIENCY ANALYSIS

4.1 Power Conversion Efficiency (PCE)

With the analysis of I_o through breaking it into regions, power analysis may begin with a summary of I_o in (4.1).

$$I_o = \begin{cases} I_{fcr} & , V_{thp} \leq V_{in} \leq V_{out} \\ I_{ps} & , V_{thn} \leq V_{in} < V_{thp} \\ I_{fs} & , v \leq V_{in} < V_{thn} \\ I_{sc} & , l \leq V_{in} < v \\ I_{rl} & , V_{in} < l \end{cases} \quad (4.1)$$

In (4.1), l is defined as a voltage in the range of $-V_{out} \leq V_{in} \leq 0$ and it depends on the ratio of transistor sizes (see chapter 3, section 3.4.6). Typically, a rectifier is supplied with a sinusoidal input voltage V_{in} . By assuming $V_{in} = V \sin \theta$, where θ is an arbitrary frequency parameter in radian, the instantaneous input power (P_i) and output power (P_o) are given by the following expressions.

$$P_o = V_{out} \cdot I_o \quad (4.2)$$

$$P_i = I_o \cdot V \sin \theta \quad (4.3)$$

Power conversion efficiency (PCE) is defined as the ratio between the average output power and the average input power. By using (4.2) and (4.3), this definition becomes:

$$PCE = \frac{V_{out} \cdot \int_0^{2\pi} I_o \cdot d\theta}{V \cdot \int_0^{2\pi} I_o \sin \theta \cdot d\theta} \quad (4.4)$$

The PCE equation is defined from the half-wave bridge rectifier. However, without changing I_o , it applies equally to the full-wave bridge circuit. It is realized that the output

current simply doubles for the full-wave rectifier (see chapter 3, section 3.2). If this new current is to replace I_o in (4.4), the same doubling factor in the numerator and the denominator will cancel, resulting in the same PCE. From (3.4), (3.6), (3.8), (3.9) and (3.11), it is seen that a factor of β_p can be extracted from I_o of (4.1). Since this factor is shared by both the numerator and the denominator of (4.4), it is canceled. The implication is a PCE that depends on g^2 but not the particular values of β_p . According to (4.2), output power of the rectifier may be increased or decreased by changing either the output voltage V_{out} or I_o . In the next section, V_{out} will be shown as an important parameter for PCE; for this reason, it is assumed not changeable. This leaves the only option of changing I_o by changing β_p . The new implication, therefore, also means that as long as β_n for the N-type transistor is changed at the same time for the same g^2 of (3.2), the PCE will remain unchanged. In other words, for the same PCE, a design can be scaled regardless of the power level!

Furthermore, it is intuitive that high PCE should occur when the forward voltage drop through the transistors is small. From (4.4), this happens when $V \approx V_{out}$, where $\theta \approx \pi/2$. However, the condition would make the integral terms in the numerator and denominator identical and, therefore, cancellable. This leads to $PCE \approx V_{out}/V$, which is certainly not true for all conduction angles but, nevertheless, exemplifies the importance of the ratio and it is, in-turn defined as δ in (4.5) to be a value that ranges from 0 to 1.

$$\delta = \frac{V_{out}}{V} \quad (4.5)$$

In general, PCE depends on g^2 , δ , V_{thn} , V_{thp} , V_{out} , V_{sbp} , V_{sbn} , λ_{subp} , λ_{subn} , τ_p , τ_n and n . Amongst these parameters, λ_{subp} , λ_{subn} , τ_p , τ_n and n are intrinsic parameters, which are fixed once the process is selected. The V_{sbp} and V_{sbn} voltages modify V_{thn} and V_{thp} through body effect, which also affects I_{sc} and I_{rl} in (3.9) and (3.11). However, their effect to PCE is not as pronounced as the remaining parameters, namely g^2 , δ and V_{out} . Body effect will be discussed in section 4.3.

4.2 Maximizing PCE by 3-D Contour Analysis

It is now understood that, apart from the intrinsic transistor parameters, the value of PCE is a function of g^2 , δ and V_{out} . In addition, these variables can be constructed into a four dimensional coordinate system. Certainly, if one knows a particular set of values for the variables, one can setup a rectifier circuit in Cadence Spectre and the resulting simulated PCE should be very close to that of a real circuit under the same given operating condition. However, the question that remains is whether there exists a point of maximum PCE; if one exists, where is this point? To this end, it should be understood that the four variable dimensions present a complex problem with many possibilities. How should this optimization problem be tackled and from where? Practically speaking, if a dominant variable exists, it could be used as the starting point. We propose that δ is the most important of all and chapter 5 will prove the correctness of this hypothesis.

If there is a three dimensional body such as a mountain, it can be projected to a two dimensional plane as contour lines and with each of them signifying points around the mountain with equal height. In this traditional sense of contour analysis, a 3-dimensional object is reduced to multiple lines on a 2-dimensional coordinate plane. Since the line intervals normally represent the same height, information such as the slope of the mountain at a particular site can be calculated by measuring how close the contour lines are to each others. These slope values provide directional information for the peak search. Likewise, our PCE model is 4-dimensional but it can be projected on a 3-dimensional coordinate system as surfaces. Each of these surfaces consists of points around the 4-dimensional object with equal δ and we name it a 3-D contour. As expected, the point of maximum PCE should satisfy the following condition.

$$\frac{\partial PCE}{\partial V_{out}} = 0, \quad \frac{\partial PCE}{\partial g^2} = 0, \quad \frac{\partial PCE}{\partial \delta} = 0 \quad (4.6)$$

Another point of interest is why a 3-D contour of equal PCE is not used instead of a contour of equal δ . After all, the task at hand is to optimize PCE and the former option only seems more direct. The main reason lies in the peak search method. As it is well known, (4.6) requires that the PCE function is defined, continuous and differentiable at all values of V_{out} , g^2 and δ . The fact is that PCE is defined for these variables over a wide

range of values except for the extreme when V_{out} is very close to zero or when δ is very close to 1. The latter is easy to handle as they occur at known places where the data can just be excluded. Apart from these exceptions, for instance, we can say that a PCE value is defined for any value of δ . However, it cannot be said that δ is defined for all values of PCE. This is especially problematic as the goal is to search for the highest PCE but its value is obviously not known prior to the effort. A wrong selection of PCE values could land the optimization to where δ values do not exist.

Our target process is ST CMOS 90nm and we expect to see short channel effect coming into play. To account for these effects, we rely on extracted parameters from the BSIM3 model. We paid particular attention on the subthreshold and reverse leakage region as these are believed to limit PCE on the high end. For instance, parameters are added to (4.7) and (4.8) so that curve fitting makes sure that the data matches that of BSIM3. The extended equation for I_{sc} and I_{rl} are shown as below.

$$I_{sc(ext)} = -\tau_p \beta_p e^{\frac{V_{gsp} - V_{sbp}}{nV_T}} \cdot \left(1 - e^{\frac{V_{dsp}}{V_T}} \right) \cdot \left(1 - \lambda_{subp} V_{dsp} + \zeta_{subp} V_{dsp}^2 - \eta_{subp} V_{dsp}^3 \right) \quad (4.7)$$

$$I_{rl(ext)} = -\tau_p \beta_p e^{\frac{V_{sbp}}{nV_T}} \cdot \left(1 - e^{\frac{V_{dsp}}{V_T}} \right) \cdot \left(1 - \lambda_{subp} V_{dsp} + \zeta_{subp} V_{dsp}^2 - \eta_{subp} V_{dsp}^3 \right) \quad (4.8)$$

The mentioned modifications do not change the form of (4.4) but the system is, nevertheless too complicated for any analytic solution. Cadence Spectre can provide PCE solution for 3-D contour analysis. However, these involve a) setting different values of V_{out} , g^2 , δ ; and b) evaluating the PCE for each case. Out of the two steps, the first one may be automated to a limited extend by using the parametric simulation option. The second step, however, may only be performed manually with the Calculator function in Spectre. To understand the difficulty of this option, we first look at the variable space. For the ST 90nm process, V_{out} ranges from 0 to 1. In general, δ also ranges from 0 to 1 while the value of g^2 may approach zero but a high end of 10 should be sufficient. A high enough resolution is also required for meaningful accuracy. Assuming 100 steps for each variable, 100^3 or 1 million experiments are required. This is too much for manual operation. Therefore, MATLAB has been used for the computation instead. Figure 4.1

illustrates the resulting 3-D contour at $\delta=0.82$ when both V_{sbp} and V_{sbn} equals zero. As discussed, a different 3-D contour such as figure 4.1 comes with every new δ value. The global maximum PCE can be determined at a point, which satisfies the conditions in (4.6). In this case, this global maximum occurs at 74.6% when $V_{out}=0.28$, $g^2=0.5$ and $\delta=0.82$. Figure 4.2 is the 2-D projection of the contour with PCE showing in varies shades of gray. The gray scale has been contrast enhanced in order to show the region of maximum PCE (the brightest region). In general, it is observed that the point of maximum PCE only occurs when V_{out} is slightly higher than V_{thp} . This is reasonable since a V_{out} that is too low will force the transistor to turn on in subthreshold region, where the forward region current is small and comparable in size with the reverse conduction current. When V_{out} is too much greater than V_{thp} , the reverse triode conduction current will also grow and become comparable with the forward current. In previous literature, this latter effect is referred to as the self current limiting feature [19].

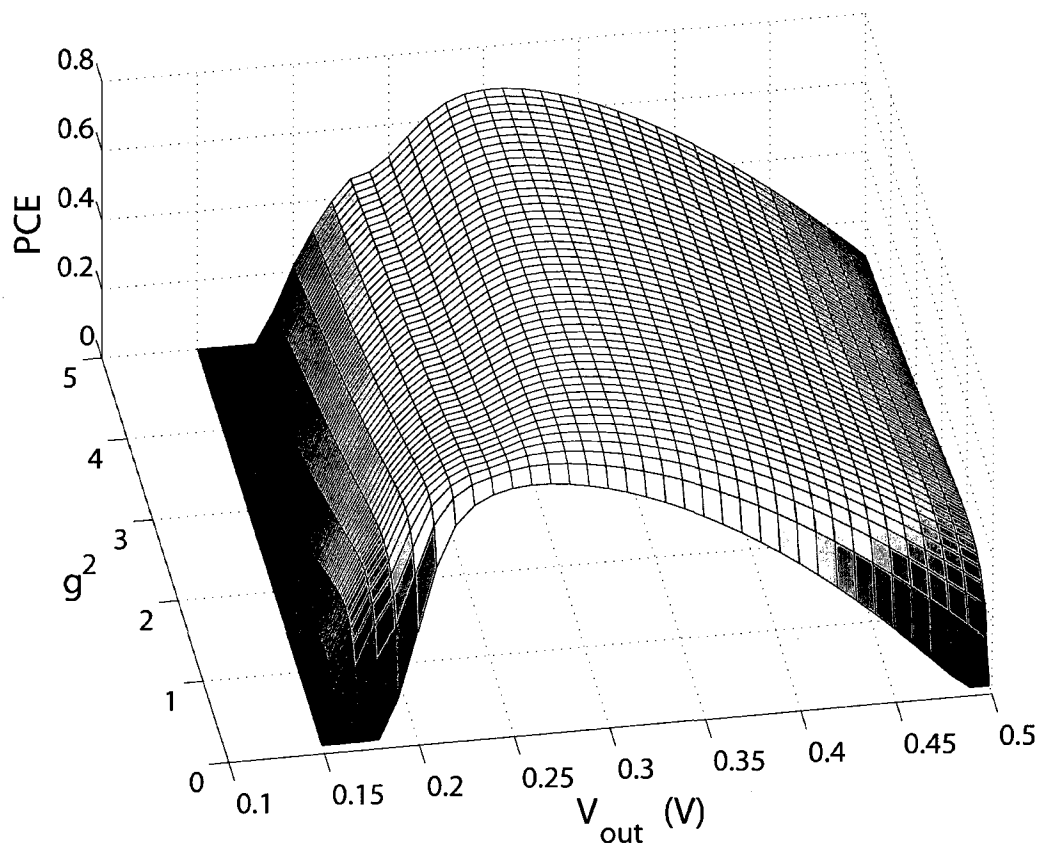


Figure 4.1: 3-D Contour for $V_{sbn}=0$, $V_{sbp}=0$ and $\delta=0.82$

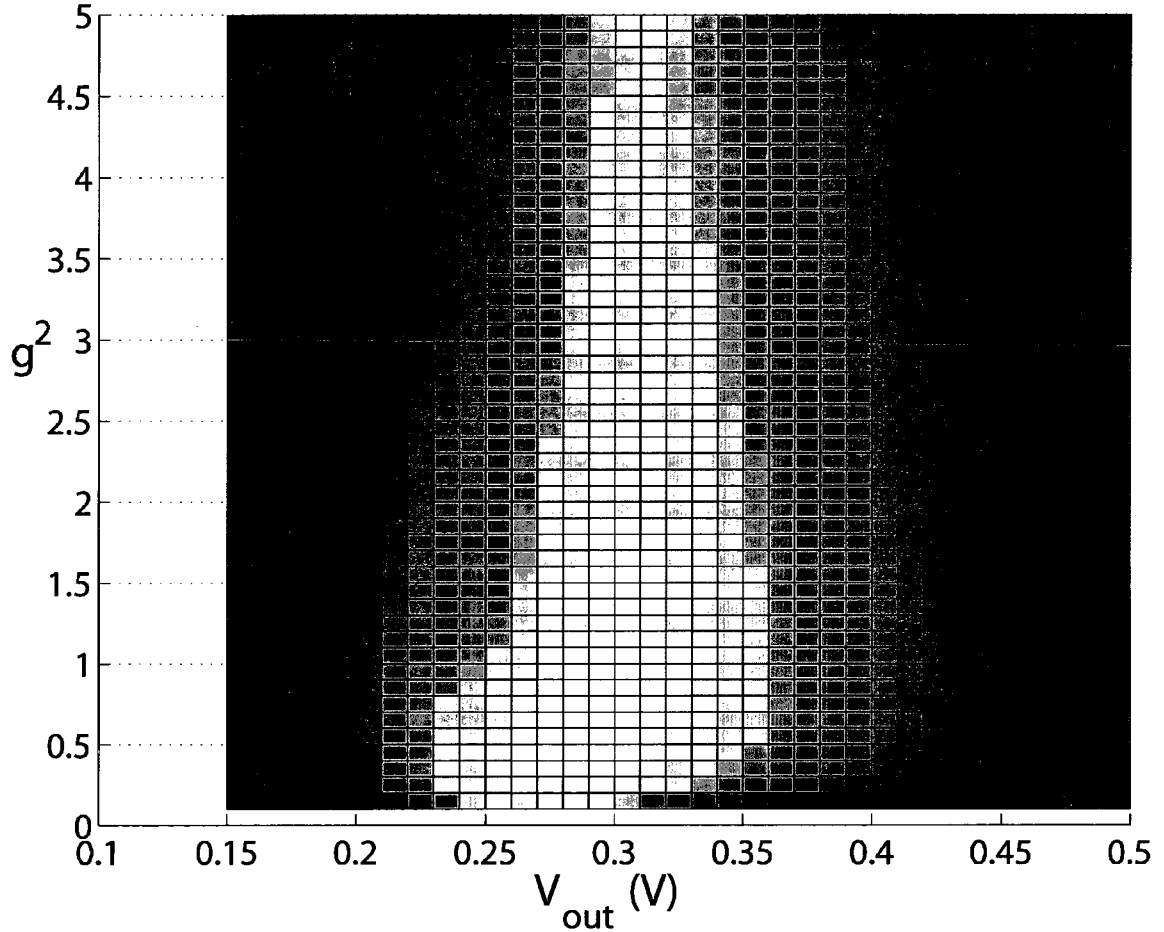


Figure 4.2: Enhanced Contrast 2-D Contour Projection for $V_{sbn}=0$, $V_{sbp}=0$ and $\delta=0.82$

4.3 Effect of Body Voltage to Maximum PCE

Traditional analysis has always identified V_{th} as the key reason for power loss in any rectifier circuit. Since a positive source-to-bulk bias will increase the threshold of an N-type transistor, it is naturally assumed that body effect will also degrade PCE, which becomes a problem especially for multi-stage rectifier design using regular CMOS transistors with non-zero V_{th} . This is because the bulk of the N-type transistors in intermediate stages will unavoidably experience an increasingly negative bias voltage. However, this assumption is not necessary valid for a differential-drive CMOS bridge rectifier, as seen in the following 3-D contour for the case when body effect is present. Likewise, figure 4.4 is showing the contrast enhanced version of a 2-D projection.

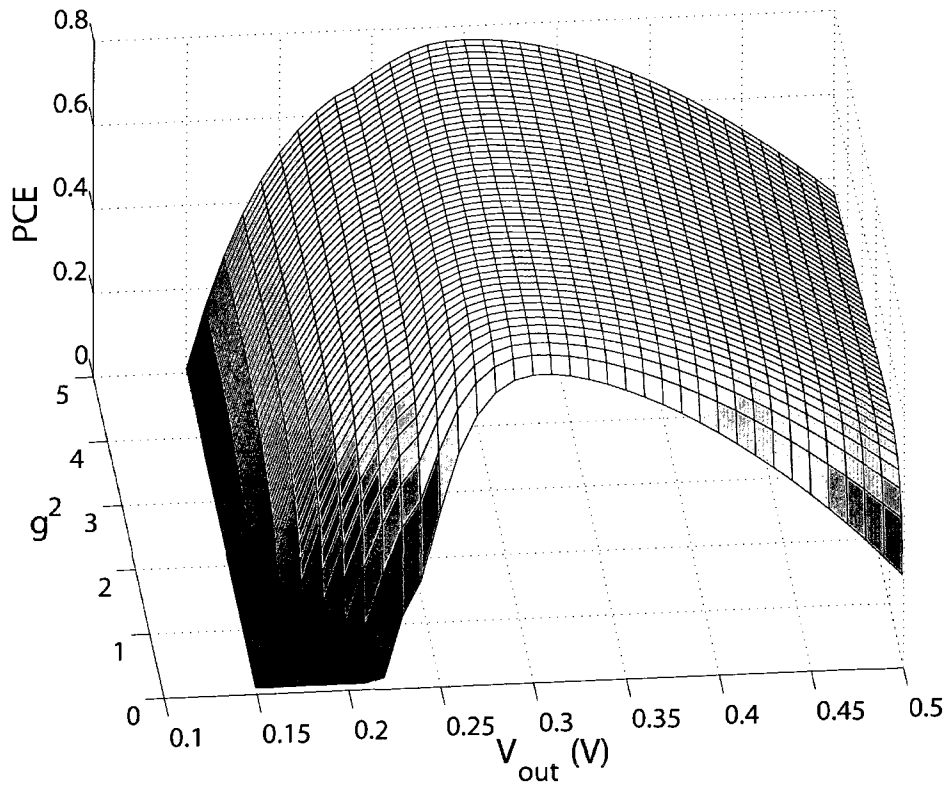


Figure 4.3: Efficiency Contour for $V_{sbp}=0$, $V_{sbn}=0.66$ V and $\delta=0.82$

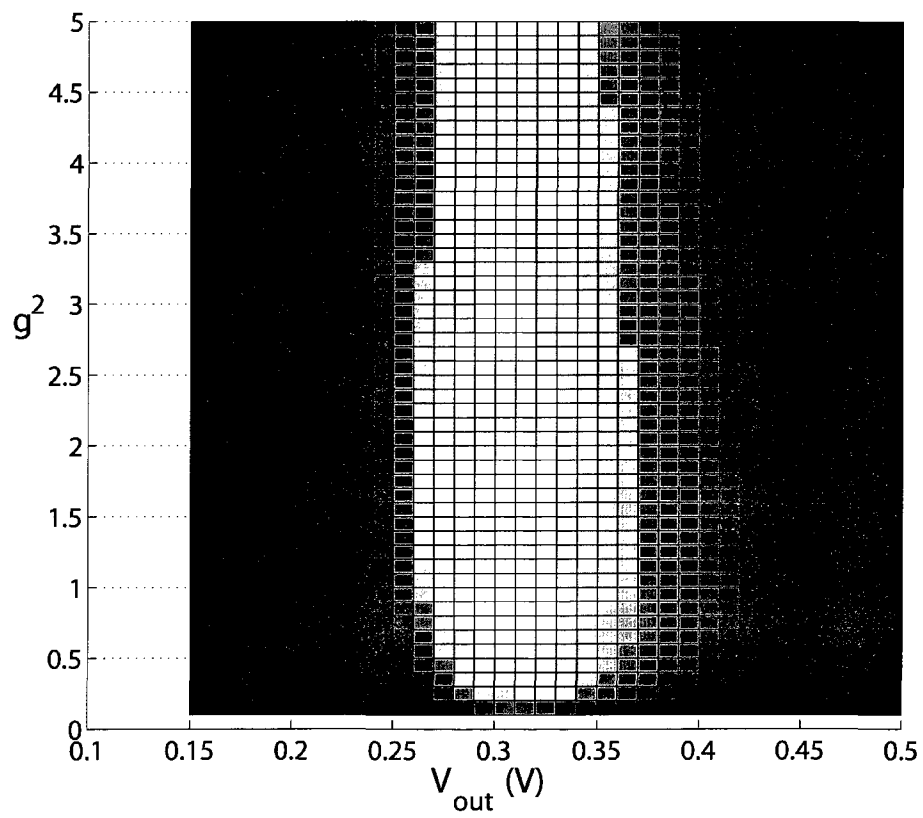


Figure 4.4: Enhanced Contrast 2-D Contour Projection for $V_{sbn}=0.66$ V, $V_{sbp}=0$ and $\delta=0.82$

By referencing figure 4.2 and 4.4, we observe that the point of maximum PCE has simply shifted to $V_{out}=0.3$ and $g^2=1.2$, at a higher value of 78.8%. This may seem counter-intuitive at first but we recognize that higher source-to-bulk voltages will suppress both the subthreshold and reverse leakage current (I_{sc} and I_{rl}), leading to a small PCE improvement as these current are still not dominant when comparing to current of the other regions.

4.4 Effect of Finite Gate and Substrate Resistance to Maximum PCE

When using a 915MHz source, we need to consider the AC equivalent circuit as discussed in chapter 3 section 3.3. Cadence Spectre simulation of the two previous cases show slightly less PCE at 72.4% and 74.3%, comparing to 74.6% and 78.8% in figure 4.1 and 4.3, respectively. This is reasonable as the 3-D contour analysis is a DC analysis, which does not take into account the finite resistance due to the poly-silicon gate and the substrate of the transistors. These in-series resistance can manifest itself as power loss. To try making up for the loss, the input signal amplitude needs to be increased, therefore, causing the reduction in δ . In addition, V_{out} also increases because more power transferred in the forward conduction should partially offset the negative impact of increased loss. The effect can be quantitatively estimated as chapter 6 will detail.

Chapter 5 SIMPLIFIED POWER MODEL

5.1 Approximating the Non-Linear Rectifier by a Linear Resistor

As it is apparent from the analysis of chapter 4, PCE only becomes the highest when $V_{out} \geq V_{thp}$. The transistors enter triode region during this condition and behave resistively. Although the transistors still behave non-linearly in a strict sense, they may be modeled sufficiently by a linear fix-value resistor. The following paragraphs will show that such an approximated model is possible. The model offers not only useful insights but allows the development of a design methodology for appropriate matching, which will be detailed in chapter 6.

The 3-D contour analysis in chapter 4 helps determining the global maximum PCE at a unique set of V_{out} , g^2 and δ values. For a half bridge, we may then guess at one transistor's size and adjust the other one for meeting the g^2 criteria. The resulting circuit can be simulated with Cadence Spectre so that the exact average output current is determined. Depending on the required load current, the transistors can be linearly scaled for reaching the target, knowing that the scaling will not affect PCE as long as g^2 is kept constant. The problem now is how the impedance characteristic for the rectifier may be obtained. Knowing the impedance characteristic is important for many reasons. To the least, it provides information for the design of matching networks. However, due to the complexity of the rectifier and transistor models, analysis based on level one SPICE model does not provide accurate answer. Therefore, the usual approach is to measure impedance by experiment. This is counter-productive for the design of matching since the circuit may also demand very specific impedance from the rectifier in order that maximum power transfer will occur at the desired operating range. Indeed, chapter 6 will show that this requirement of the rectifier's impedance is very specific. So, the method of matching the known impedance of a rectifier to the antenna will not automatically yield the most desired performance. We clearly need a rule for sizing transistors so that impedance characteristic is designed rather than shot in the dark.

To begin explaining our model, rectifier's impedance in the usual sense should be understood first. Simply put, a rectifier is reduced to a complex but linear impedance. The impedance consists of a combination of parasitic capacitance and the average on-region resistances of the transistors. This model is valid when the rectifier's conduction angle is so large that the switching discontinuity may be ignored. However, conduction angle is only large when PCE is also low as section 5.3 will prove. For a rectifier operating with high PCE, the conduction angle is too small for the switching discontinuity to be ignored and the usual impedance model does not apply. Our solution is simple, instead of reducing the rectifier to a linear impedance, we reduce it to a piece-wise linear circuit, such that the problem of switching discontinuity is addressed. The piece-wise linear model replaces the transistors by a switch, which turns on when $V_{in} \geq V_{thp}$; the impedance characteristics of the transistors are summarized by a fixed valued resistor that connects to the switch in-series. For a half bridge rectifier that outputs a certain size of average output current, it is always possible to find for the piece-wise linear model a resistor value, which gives the same current.

Instead of deriving a general abstract mathematical formulation for proving the model's validity, illustrations through an example may offer more insight. Figure 5.1 compares one such fix resistor example with a real half bridge rectifier when both circuits are driven by a full cycle of sinusoidal input signal. The parameters for the circuit and the model are shown in table 5.1.

Table 5.1: Parameters for a Sample Rectifier Circuit and the Model

Circuit and Model Parameters	Values
δ	0.77
g^2	1.4
V_{out}	0.33V
V_{thn}	0.2806V
V_{thp}	0.2785V
P-Type transistor width	31.35 μ m
All transistor lengths	0.1 μ m
Effective Resistance for the Model	579.4 Ω

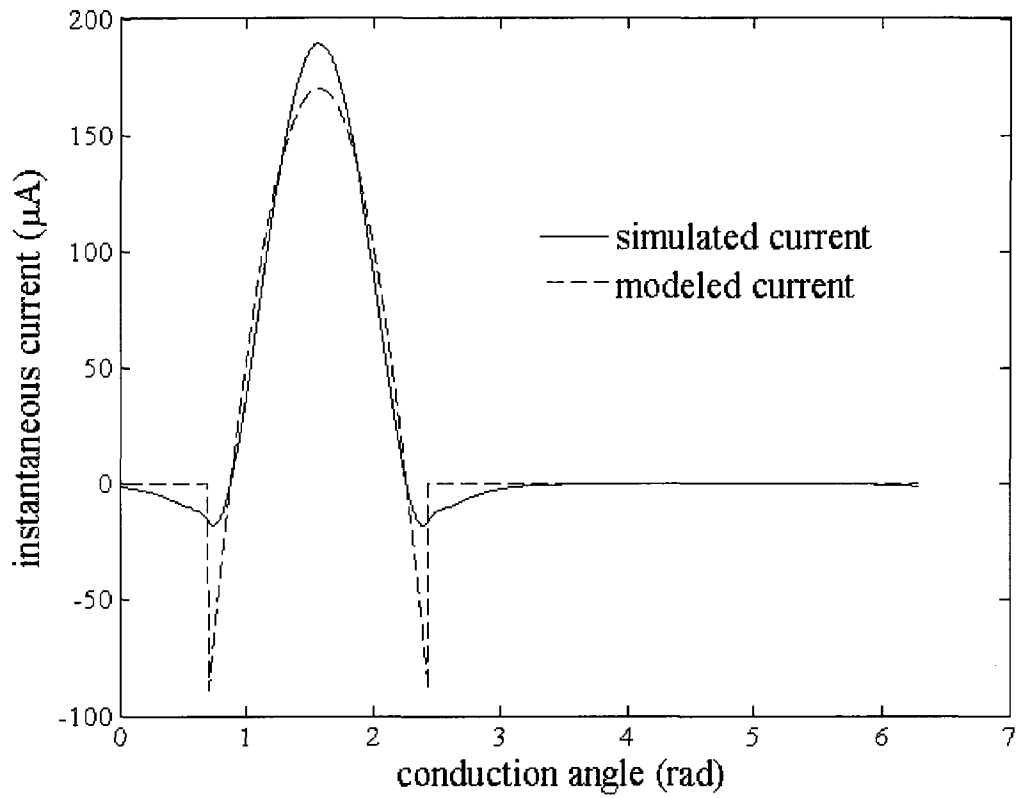


Figure 5.1: Simulated Rectifier Current vs Fix-Resistor Modeled Current

Both the rectifier and the model output an average current of exactly $22.4\mu\text{A}$. From figure 5.1, it is first observed that the reversed conduction current eventually settles to a negligibly small value, such that only half of the cycle needs to be considered. Out of this half cycle, the fix-resistor matches the real transistors very well 25% of the time. Close to the peak, where the rectifier current is the highest, the model underestimates by 10%. This means less than 10% of discrepancy in current for 20% of the half cycle time. The peak reverse current for the model is also much greater because the transistors only start to turn on but still with high resistances and 10% of time is spent in this region. This leaves more than 50% of time in the subthreshold and reverse leakage region, where the fixed resistor model is completely turned off. The observation so far amounts to how good the turned-on transistors are modeled by a fix resistor for matching purpose. The picture, however, is not complete without the power also being analyzed. Figure 5.2 is a comparison of the instantaneous power waveforms for the same example.

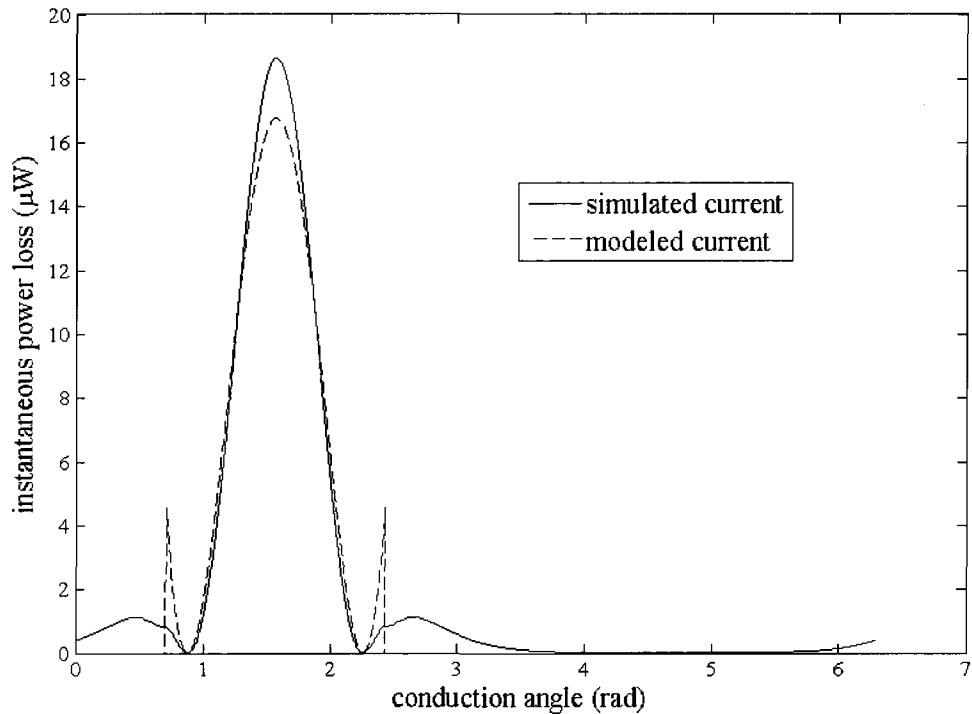


Figure 5.2: Simulated Power Loss vs Fix-Resistor Modeled Power Loss

It is observed from figure 5.2 that over 87% of the power loss is due to the forward conduction region. Therefore, matching is a more important concern for this region. In this sense, the 10% peak current discrepancy observed in figure 5.1 is a reasonable sacrifice for the modeling simplicity. In terms of power, there is less than 3% discrepancy in the forward conduction region but 69% error in the reverse region. To address this error, we add an in-parallel resistor (loss element) to the model that we have discussed so far, resulting in figure 5.3, the final piece-wise linear model for differential-drive CMOS half-wave bridge rectifier (DDCHB). It should be understood that the actual loss element is non-linear in nature and the loss resistor, therefore, does not really exist. However, it can be used as a fudge factor for changing the model enough so that the experimental data for high PCE will fit. In fact, although this model is derived from DC analysis, it can capture the AC effects by means of the loss element (see Chapter 7). Regardless, this is an important step that enables design. To see how well this scheme works, in the next section, the model will be compared with actual simulated PCE values.

5.2 The Piece-wise Linear Model for DDCHB

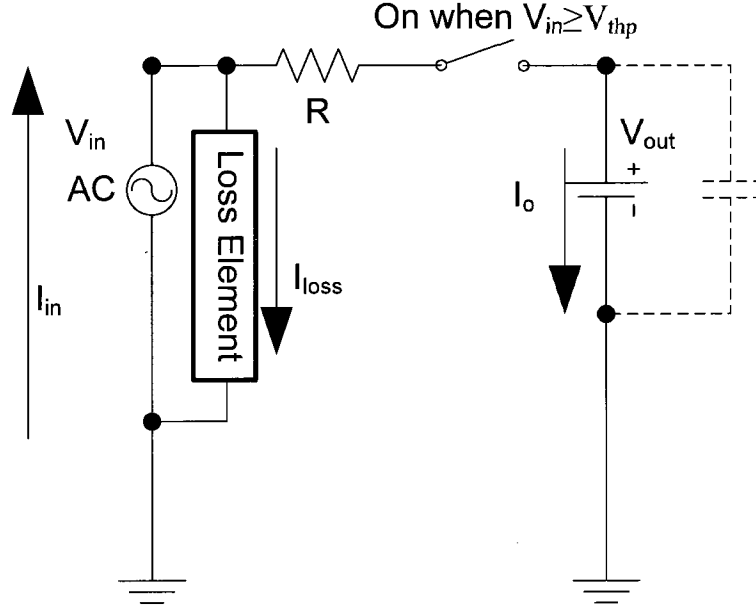


Figure 5.3: Piece-wise Linear Model of the Half Bridge

If V_{in} is a sinusoidal source of the form $V \sin \theta$, we may derive the expression of average output current I_o as below,

$$\begin{aligned}
 I_o &= \frac{1}{2\pi} \int_{\sin^{-1} \chi \delta}^{\pi - \sin^{-1} \chi \delta} \frac{V \sin \theta - V_{out}}{R} d\theta \\
 &= \frac{V}{2\pi R} \left[2\sqrt{1 - (\chi \delta)^2} - \delta \pi + 2\delta \sin^{-1}(\chi \delta) \right]
 \end{aligned} \tag{5.1}$$

where R is the equivalent resistance and $\chi = V_{thp}/V_{out}$. If the loss element is given by R_{loss} , the expression of PCE can be derived by using (4.4) and (5.1) as below.

$$PCE_{(fix-res)} = \delta^2 \cdot \frac{\frac{2}{\delta} \sqrt{1 - (\chi \delta)^2} - \pi + 2 \sin^{-1}(\chi \delta)}{\frac{\pi}{2} - \sin^{-1}(\chi \delta) + \delta \cdot (\chi - 2) \cdot \sqrt{1 - (\chi \delta)^2} + \frac{\pi R}{R_{loss}}} \tag{5.2}$$

5.3 Comparison between the Model and the Simulated Behavior

Since δ has been suggested in chapter 4 as an important parameter for PCE, figure 5.4 gives a comparison between δ , the PCE of the piece-wise linear model and the actual Cadence Spectre simulation result.

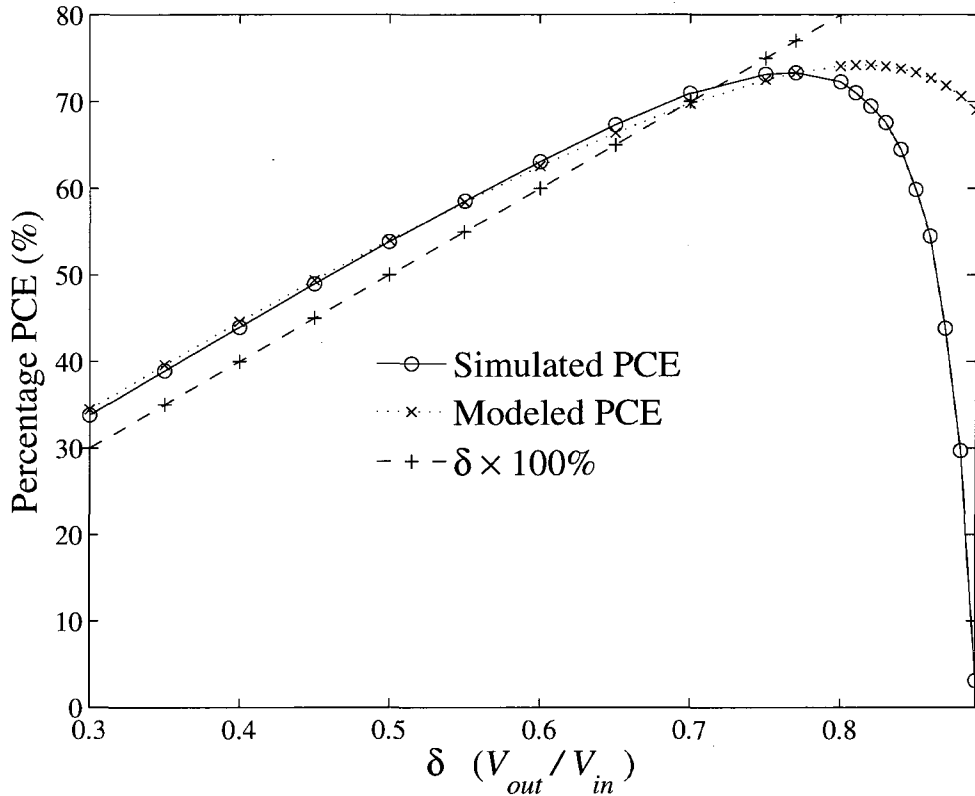


Figure 5.4: Comparison between the Simulated PCE, the Modeled PCE and δ

It is seen that the model PCE matches the simulated PCE almost exactly from a low PCE value all the way up to the maximum PCE of 73.3%, before it deviates. It is understood that, at high value of δ , the subthreshold current (I_{sc}) and the reverse leakage (I_{rl}) become increasingly dominant. The fact that there is a large discrepancy also illustrates the highly non-linear nature of this leakage current, which cannot be accurately model by a linear resistor. Nevertheless, our goal is to achieve the maximum PCE, where the effect of non-linearity is not yet dominant, such that the piece-wise linear model is sufficiently accurate. On the other hand, it is noticed that δ closely follows the actual PCE curve with only less than 5% error until getting near the point of maximum PCE. This shows that PCE is roughly predicted by δ , especially when PCE is low.

Chapter 6 ANTENNA MATCHING FOR RECTIFIER

6.1 The Difficulties of Antenna Matching for Rectifier with High PCE

As the previous chapters have already implied, the concept of matching for a rectifier with high PCE having a known impedance is not as simple as designing a matching network for just making sure that the transformed rectifier's impedance is the same as that of the antenna. At least for the DDCHB or DDCFB type of rectifiers, maximum PCE only occurs at a particular combination of V_{out} and δ values. If matching is for the impedance, the δ condition may not be met and the PCE will suffer. On the other hand, if matching is for the voltage, conditions for high PCE are met but maximum power transfer may not occur. For applications that are more power demanding, the design goal should not merely focus on optimization of operating range for small output power measured in single digit unit of μW . To this end, a high PCE is certainly important but not the only design criteria. It is useful only when the rectifier is also matched to its source for the maximum power transfer. Therefore, it becomes a valid question to ask whether the power and the PCE can be optimized at the same time. To compound the problem, the rectifier is now described by a piece-wise linear model (see chapter 5) and the traditional matching theory may not apply. To be specific, it has always been assumed that maximum power transfer occur only when impedances match, a condition of the source and the load impedances being complex conjugates to each other. The following analysis will gradually unfold the fact that the new matching is quite different from the matching of traditional sense.

6.2 Piece-Wise Linear Model for Matching

By applying the piece-wise linear model, we may consider the problem of matching with the circuit in figure 6.1, where R_r is the equivalent radiative resistance from the antenna and U_0 is the amplitude of the sinusoidal source, where $\theta = \omega t$. The right-side of figure 6.1 is the equivalent circuit for piece-wise linear model on the left. If we further define k as the ratio of voltages in (6.1) and σ as the ratio of resistance in (6.2), such that:

$$k = \left(1 + \frac{R_r}{R_{loss}}\right) \cdot \frac{V_{out}}{U_0} \quad (6.1)$$

$$\sigma = \frac{R \cdot (R_r + R_{loss})}{R_r \cdot R_{loss}} \quad (6.2)$$

,we may then derive an expression for P_r , the power to V_{out} , as below.

$$\begin{aligned} P_r &= \frac{1}{2\pi} \int_{\sin^{-1}(\chi k)}^{\pi - \sin^{-1}(\chi k)} V_{out} \cdot \frac{U_0 \sin \theta - V_{out}}{R_r // R_{loss} + R} d\theta \\ &= \frac{U_0^2}{8R_r} \cdot \left(\frac{R_{loss}}{R_{loss} + R_r}\right) \cdot \frac{8k}{(1 + \sigma) \cdot \pi} \left[\sqrt{1 - (\chi k)^2} + k \sin^{-1}(\chi k) - \frac{k\pi}{2} \right] \end{aligned} \quad (6.3)$$

The first term of (6.3) is the available power (P_a) from the antenna and it is known to be proportional to G , the antenna gain [1], as below.

$$P_a = \frac{U_0^2}{8R_r} = \frac{P_{EIRP}}{4\pi r^2} \cdot \frac{\lambda^2}{4\pi} \cdot G \quad (6.4)$$

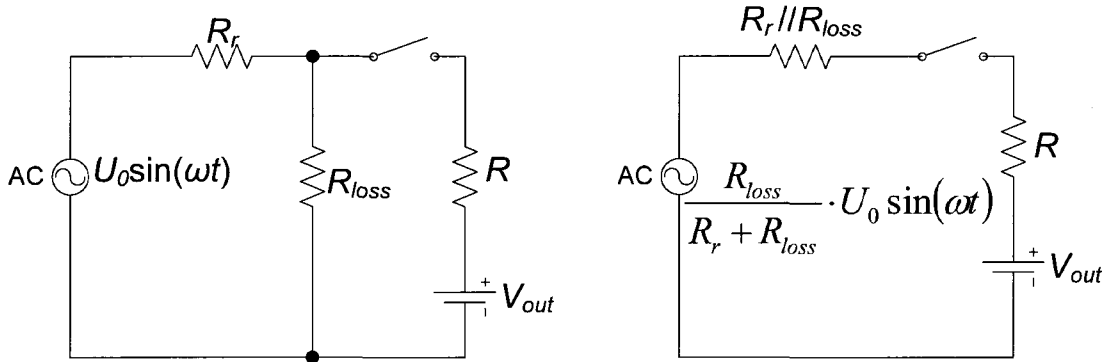


Figure 6.1: Model of the Antenna Coupled Rectifier Circuit

Alternatively, (6.4) can be rearranged for U_0 as in (6.5). U_0 is a fictitious voltage source obtained for the antenna by imagining that the antenna and its ambient RF field are indeed as figure 6.1.

$$U_0 = \frac{\lambda}{2\pi r} \cdot \sqrt{2P_{EIRP} \cdot G \cdot R_r} \quad (6.5)$$

In (6.5), λ and P_{EIRP} are fixed by standards. In our case, they are 327.87 milli-meter and 4W respectively for the 915MHz band in North America. G depends on the particular antenna. In this thesis, we assume a simple dipole antenna with $G=1.64$. The design operating range would set the r value; a list of P_a at various r is given in table 1.1. As such, the only unknown is the equivalent radiative resistance R_r .

6.3 The Inadequacy of PCE for Rectifier Design

U_0 in (6.5) gives an idea of the required antenna source voltage for achieving the maximum PCE. For instance, if it is given the pair of V_{out} and δ values for the maximum PCE obtained from the 3-D contour analysis, the value of U_0 must also be at least equal to or greater than V_{out}/δ . R_r , in turn, should be sized for allowing a certain size of average input power to the rectifier, which equals the average output power to the load divided by PCE. Although this method of design guarantees high PCE, the power through R_r may not be optimal. Depending on the choice of U_0 , too much power could end-up being reflected by R_r (radiative impedance is not dissipative). In fact, most of the power could be reflected, leaving only a small portion for the rectifier. This shows the limitation of PCE as a design parameter. A better design metric is power utilization (PU), which describes how much of the available power from the antenna can be transferred to the load (or V_{out}). In this context, available power means the maximum amount of power obtainable from the antenna with a matched linear load. As an example, PU=1 means that all of the available power from the antenna is delivered to the load. For a rectifier, PU can be very low in value but the maximum happens when 100% of the available power is delivered to the rectifier; at this state, if the rectifier also operates with the maximum PCE, the PU will equal PCE.

6.4 Power Utilization for the Rectifier

By defining PU_{half} as the power utilization of a half bridge rectifier, we form the expression of (6.6) by dividing (6.3) with P_a .

$$PU_{half} = \left(\frac{R_{loss}}{R_{loss} + R_r} \right) \cdot \frac{8k}{(1 + \sigma) \cdot \pi} \left[\sqrt{1 - (\chi k)^2} + k \sin^{-1}(\chi k) - \frac{k\pi}{2} \right] \quad (6.6)$$

Chapter 3 section 3.2 has already detailed the difference between a full bridge and a half

bridge. The power utilization for a full bridge (PU_{full}) is easily derived in (6.7).

$$PU_{full} = 2 \times PU_{half} \quad (6.7)$$

PU so defined is equivalent to the system efficiency used by S. Mandal et al. [21] but we have provided a design method instead of using it only as a measured performance merit. Now, the value of k that maximizes (6.6) can be found by taking the partial derivative of (6.6) with respect to k and then equaling it to zero, resulting in (6.8).

$$\sqrt{1-(\chi k)^2} + 2k \sin^{-1}(\chi k) - k\pi + (1-\chi) \cdot \frac{\chi k^2}{\sqrt{1-(\chi k)^2}} = 0 \quad (6.8)$$

For the special case of $\chi=1$, k equals 0.3942 and we define this special k value as k_{max} . If $\sigma \approx 0$ and $R_r \ll R_{loss}$, PU_{full} will be approximately equal to 0.9226. In the next section, we will show that σ must carry a finite value. This means PU_{full} will always be smaller than 0.9226 even when matching is ideal. This also implies a very specific value of U_0 for maximum power transfer.

6.5 Determination of the Equivalent Radiative Resistance

Since the goal is to maximize power transfer, a necessary condition is to maximize PCE. The 3-D contour analysis from chapter 4 has already shown the existence of an optimal PCE at a unique pair of V_{out} and δ values. Consequently, the voltage across the in-series network of R and V_{out} in figure 6.1 is known and it equals V_{out}/δ . Strictly speaking, the ratio is only valid for a sinusoidal input, which is not the case in figure 6.1 due to the switching distortion. The ratio actually decreases for a wider conduction angle but the decrease should be negligible for a narrow-band antenna and matching system. So, assuming that a rectifier input voltage of V_{out}/δ still yields the maximum PCE, when the AC source in figure 6.1 reaches its maximum amplitude, the current that flows through R and V_{out} should establish exactly V_{out}/δ . Therefore, the relationship between R and R_r for maximized PCE is obtained in (6.9) by solving (6.1) and (6.2) with k_{max} .

$$\sigma_{match} \approx \frac{k_{max}(1-\delta_{max})}{\delta_{max} - k_{max}} \quad (6.9)$$

In (6.9), k_{max} and δ_{max} signifies the k for maximum power transfer and the δ for optimal PCE. Therefore, σ_{match} represents the optimal resistance ratio for achieving k_{max} . It is seen that δ_{max} is always smaller than one. Hence, σ_{match} carries a finite value and it can never equal to zero. The implication is a PU_{full} , which is always smaller than 0.9226. Once σ_{match} is determined, R_r is simply found in (6.10) by substituting (6.9) into (6.2).

$$R_r = \frac{R \cdot R_{loss}}{R_{loss} \cdot \sigma_{match} - R} \quad (6.10)$$

6.6 Sizing of Transistors for Optimal Power Transfer

It is noticed that (6.10) still contains the unknown R . In order to find R , we need first to determine the value of P_r . In (6.3), since R_r is usually far smaller than R_{loss} , the second product term may be dropped with few errors. After substituting k_{max} and (6.9) into (6.3), we get (6.11).

$$P_r \approx P_a \cdot \frac{8k_{max}}{(1 + \sigma_{match}) \cdot \pi} \left[\sqrt{1 - (\chi k_{max})^2} + k_{max} \sin^{-1}(\chi k_{max}) - \frac{k_{max} \pi}{2} \right] \quad (6.11)$$

P_r may also be determined from (5.1) by using the relationship of $V=V_{out}/\delta_{max}$.

$$P_r = \frac{V_{out}^2}{2\pi\delta_{max}R} \left[2\sqrt{1 - (\chi\delta_{max})^2} - \delta_{max}\pi + 2\delta_{max} \sin^{-1}(\chi\delta_{max}) \right] \quad (6.12)$$

By equating (6.11) and (6.12), a unique R value is determined. It is reminded that (5.1) is based on our piece-wise linear model of the CMOS bridge rectifier. Therefore, the transistors can be sized freely until I_o matches to that of (5.1); this is as long as g^2 , V_{out} and δ are kept to their optimal values according to the results of the 3-D contour analysis. The process implies, for any given target operating range, there exists an optimal transistor size, where the power transfer is maximized.

6.7 Matching to Simple Antenna

We know that, in general, the rectifier circuit is not linear and the switching discontinuity will generate prominent 3rd and 5th harmonics. However, when we are only concerned

about the power transfer behavior during the rectifier's on time, the first principle time domain analysis mentioned in this chapter and in chapter 5 provides accurate results. When it comes to impedance matching, the piece-wise linear model allows the rectifier to be analyzed in a single frequency as if it is a completely linear circuit, given in figure 6.2.

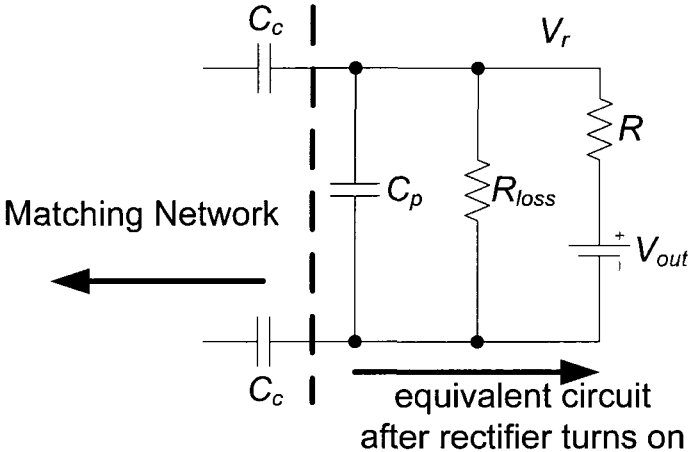


Figure 6.2: Equivalent Circuit of the Rectifier for Matching

In figure 6.2, C_p and C_c represent, respectively, the parasitic capacitance and the coupling capacitor of the rectifier. The detail of these components is explained in chapter 7. Regardless, by defining Q_2 in (6.13), we can convert figure 6.2 into an AC serial equivalent circuit in figure 6.3.

$$Q_2 = \omega C_p \cdot (R_{loss} // R) \tag{6.13}$$

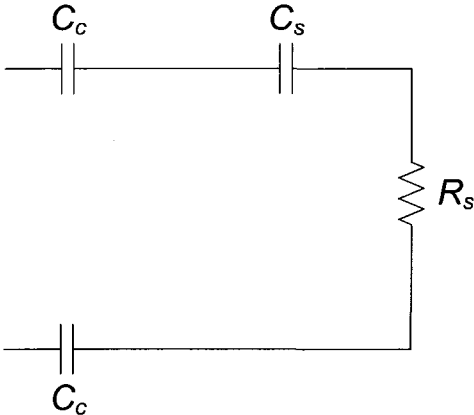


Figure 6.3: Serial Equivalent Circuit of figure 6.2

In figure (6.2), C_s and R_s are given by (6.14) and (6.15).

$$C_s = \frac{1 + Q_2^2}{Q_2^2} \cdot C_p \quad (6.14)$$

$$R_s = \frac{R // R_{loss}}{1 + Q_2^2} \quad (6.15)$$

Finally, the overall Q of the serial equivalent circuit can be found and the value of the equivalent parallel resistance and capacitance can also be determined as the following.

$$Q = \frac{1}{\omega \cdot \left(\frac{1}{2} C_c // C_s \right) \cdot R_s} \quad (6.16)$$

$$C_{eff} = \frac{Q^2}{1 + Q^2} \cdot \left(\frac{1}{2} C_c // C_s \right) \quad (6.17)$$

$$R_{eff} = (1 + Q^2) \cdot R_s \quad (6.18)$$

The effective parallel resistance value (R_{eff}) should be substituted back to (6.10) for the design value of R_r . According to the procedure in section 6.5, for an operating range of 10m, the value of equivalent resistance is about 300Ω , while $R_r \approx 1.5k\Omega$. This is completely non-intuitive but the arrangement would result in close to maximum power transfer. A typical antenna such as simple dipole has much lower radiative impedance when comparing to the design R_r . To address this problem, we can use a simple L-match for transforming the low impedance to R_r , as it is illustrated in figure 6.4.

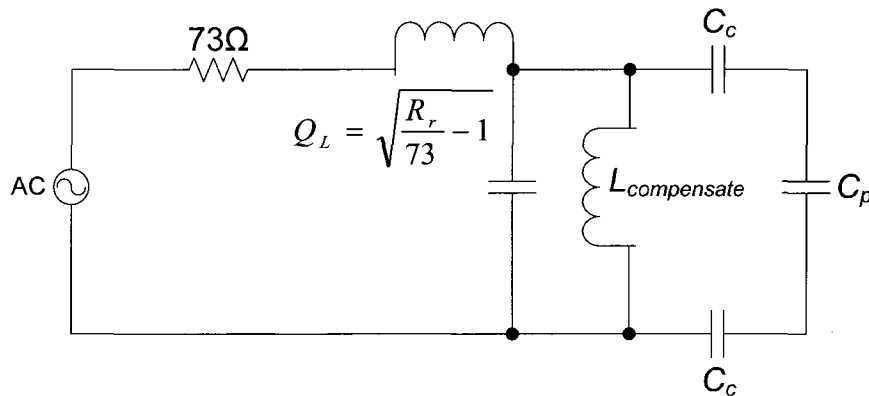


Figure 6.4: L-match Network to a Dipole Antenna

In figure 6.4, Q_L is the quality factor for the L-match while $L_{compensate}$ is the inductance for cancelling the reactive component of the rectifier circuit. This is done so that the power utilization, which assumes no reactive current, will not be affected. It is important to realize that $L_{compensate}$ is for compensating $0.5C_d/C_p$ but not the effective capacitance (C_{eff}) in (6.17). Chapter 7 will show that this is appropriate. In general, it is not possible to use one linear inductor for complete cancelation of the reactive current because the capacitance is non-linear. The leftover component, however, is small due to the small conduction angle for high efficiency operation and it will not significantly undermine power utilization.

Chapter 7 MULTI-STAGE CMOS DIFFERENTIAL FULL-BRIDGE RECTIFIER DESIGN

7.1 Topology of Multi-Stage DDCFB

The design method discussed so far is for the single stage differential-drive CMOS half-wave (DDCHB) and the full-wave bridge rectifier (DDCFB). However, as chapter 4 has revealed, optimal PCE is fixed at a particular set of V_{out} , g^2 and δ values. The value of V_{out} should also be slightly greater than the P-type transistors' threshold voltage (V_{thp}) but not too much greater. This voltage is in the order of 0.3V for the ST CMOS 90nm process. Table 7.1 gives a list of threshold voltages at varies source-to-bulk bias levels.

Table 7.1: V_{th} at Different Body Bias Voltage

V_{sb} (V)	V_{thn} (V)	V_{thp} (V)
0	0.2237	0.2785
0.2	0.2434	0.2991
0.3	0.2523	0.3088
0.33	0.2549	0.3117
0.36	0.2574	0.3145
0.6	0.2763	0.3364
0.66	0.2806	0.3417
0.72	0.2849	0.3469

The low V_{out} of a single stage design is, therefore, insufficient for most application. Fortunately, multiple stages can be cascaded for increasing the output voltage. One topology based on cascading DDCFB is shown in figure 7.1.

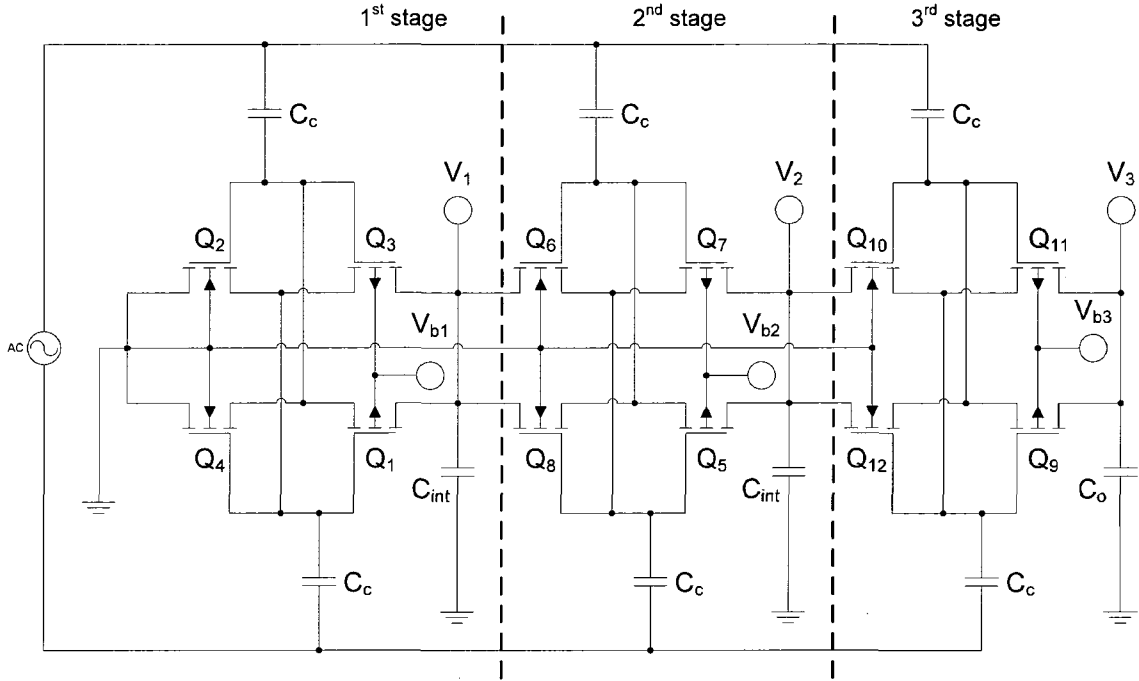


Figure 7.1: A 3-Stage DDCFB

This is a three-stage design with stage boundaries indicated by the dotted line. It is possible to repeat the basic cell for even higher output voltage. The Q s with odd suffixes are P-type transistors and those with even suffixes are N-type transistors. C_{int} represents the charge reservoirs of the intermediate stages and they are assumed to have the same value. C_o is the output capacitor that connects to the load and it may carry a bigger value than C_{int} . The capacitance size of C_o depends on the transient current requirement of the load. It is noticed that all bulk connections of the N-type transistors are wired to the same ground due to the process but there is a degree of freedom for setting the bulk voltages (V_{b1} , V_{b2} and V_{b3}) for the P-type transistors because of the separate N-wells. This freedom can be utilized for a design advantage to be discussed in the next section. For illustrating how the output voltage is increased, a basic DDCFB cell and its equivalent circuit is shown in figure 7.2.

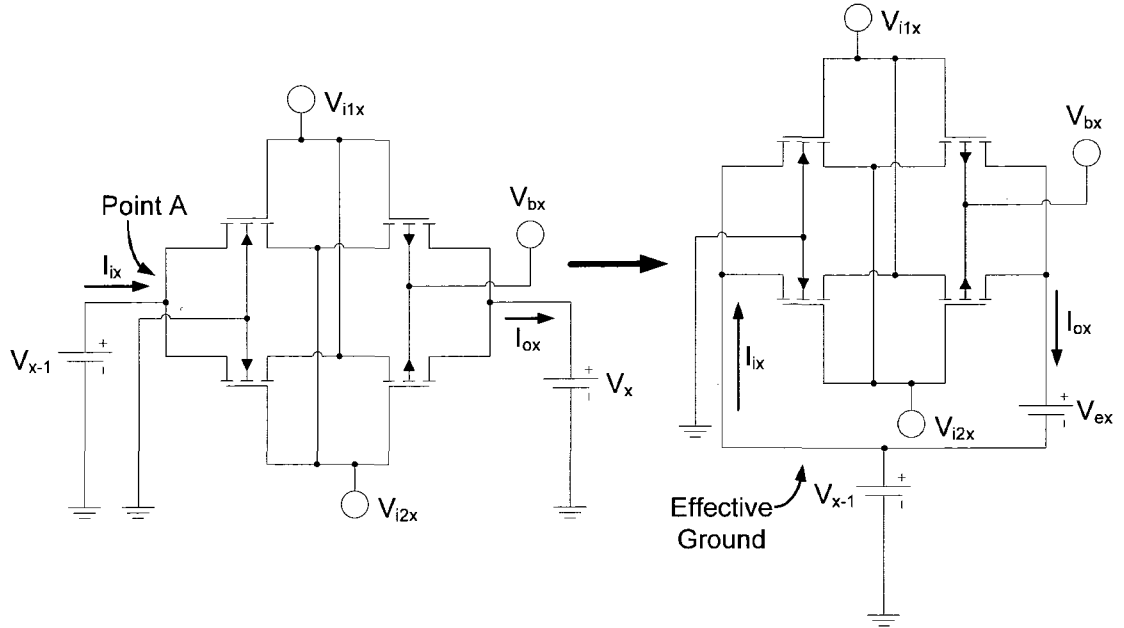


Figure 7.2: A Basic DDCFB Cell for Multi-Stage Design (left), the Equivalent Circuit (right)

In figure 7.2, the x suffix represents the stage number. By the quasi-steady state DC model introduced in chapter 3, the output capacitors are replaced by the voltage sources V_x and V_{x-1} , which represent the output voltages of the current stage and that of the previous stage, respectively. V_{i1x} and V_{i2x} are the input voltage terminals. I_{ix} and I_{ox} are the current driven by the $(x-1)$ -th stage to the x -th stage, and the output current of the x -th stage, respectively. A comparison between the left diagram of figure 7.2 and figure 3.1 shows the only difference between the two. It lies in V_{x-1} , which is connected to point A or the drain ends of the N-type transistors. For the single stage DDCFB, point A is always connected to ground. Since it has already been shown in chapter 3 that DC analysis is the most critical for the understanding of DDCFB and DDCHB, DC analysis is used here too. With this assumption, it is clear that, during the steady state, I_{ox} equals I_{ix} . If V_x is separated into two voltages, such that $V_x = V_{ex} + V_{x-1}$. It is seen that I_{ox} will flow through the V_{x-1} at the output branch while I_{ix} , which is equal but opposite to I_{ox} , will flow through the V_{x-1} source at point A. Hence, the two V_{x-1} voltages are combined in the equivalent circuit on the right side of figure 7.2, where V_{x-1} becomes the new effective ground and the potentials of V_{i1x} and V_{i2x} are raised accordingly. It is for this reason that the C_c coupling capacitors are added in figure 7.1. If V_{ex} equals V_1 for all stages, it is straightforward to show that $V_x = x \cdot V_1$.

7.2 Optimal Number of Stages

Chapter 6 section 6.4 and section 6.5 have implied that all stages should have the same V_{out} . This is in order that k_{max} , and therefore, the optimal power matching condition can be maintained for all stages. Due to these reasons, the optimal number of stages (OT) is approximately given by (7.1).

$$OT = \frac{\text{desired total output voltage}}{V_{thp} + \Delta} \quad (7.1)$$

Δ in (7.1) is an additional offset that V_{out} of individual stages must operate above V_{thp} for achieving the optimal PCE. The value of Δ is empirically observed from chapter 4 to be about 0.01-0.03V.

Assuming a Δ of 0.02V and the desired total output voltage of 1V, when $V_{sb}=0$ and $V_{thp}=0.2785V$, OT should equal 3.35, which unfortunately is not an integer. If OT is rounded to three stages with V_{out} equals 0.33V per stage, V_{thp} needs to be increased in order for keeping PCE. This can be accomplished by modifying the body bias of the P-Type transistors. While referencing figure 7.1, this strategy involves connecting V_{b1} to V_2 and then V_{b2} to V_3 . Since the 4-th stage does not exist, V_{b3} is simply connected to V_3 . With this arrangement, according to table 7.1, the V_{thp} for both the first and second stage should equal 0.3117V. The source-to-bulk voltages for the P-type transistors of the 3-rd stage remain zero but that of the N-type transistors will be biased to 0.66V, which forces the V_{thn} to be above V_{thp} at a value of 0.2806V; the higher voltage means that the bridge circuit will turn on at V_{thn} instead of V_{thp} . However, this is still much lower than 0.33V when comparing to the other stages and the PCE will not be optimal. Fortunately, the 3-D contour close to the optimal point of PCE is rather flat (see chapter 4) and the raised source-to-bulk voltages also ensure that significant PCE compromise is unlikely.

7.3 Coupling Capacitors and Parasitic Gate Capacitances

The power matching criteria (chapter 6) requires the ratio of the effective output voltage of each stage and the effective input amplitude be kept very close to k_{max} . The effective output voltages are kept the same by making sure of a constant V_{ex} through all stages. Since the coupling capacitors (C_c) are also a part of the input network, they should all

have the same value. These capacitors will form capacitive dividers with the parasitic gate capacitances (C_p) of the transistors as in figure 7.3. (R_{loss} is negligible)

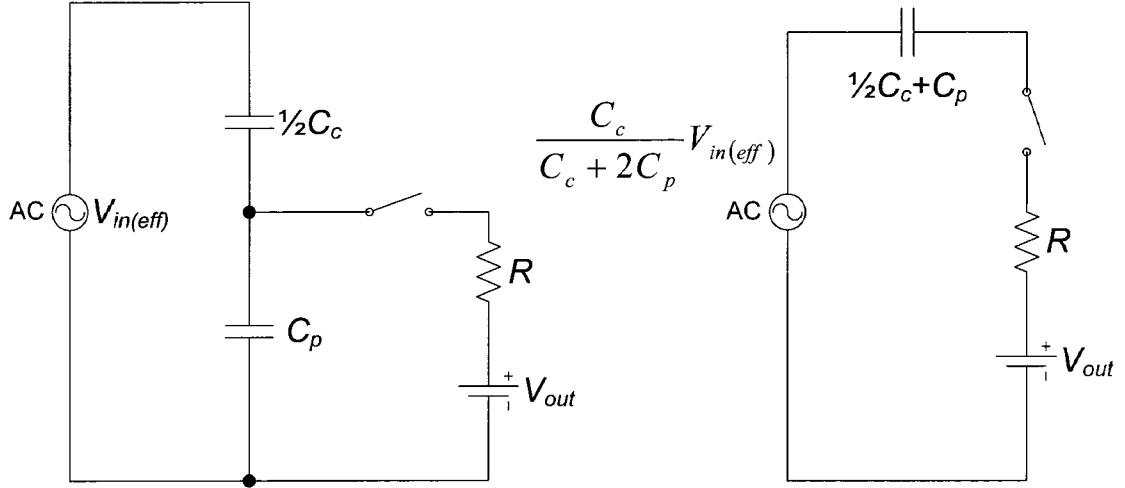


Figure 7.3: Parasitic Capacitances in the Equivalent Circuit

$V_{in(eff)}$ is introduced because the bridge rectifier (R , V_{out} and the switch) will see a voltage different from that of the antenna. Due to the switching discontinuity, the circuit cannot be analyzed directly with the usual impedance model but the equivalent circuit on the right side of figure 7.3 can still be formed. This arrangement is invariant to the effect of switching and it allows straightforward evaluation of initial conditions because the equivalent capacitances only store charges as a result of switching, unlike the actual capacitors, which also store charges from $V_{in(eff)}$.

The first important thing to know is the voltage seen by the rectifier, across C_p on the left circuit of figure 7.3. The δ ratio between this voltage and V_{out} must be maintained for maximum PCE. To find this voltage, we define the instantaneous current through the R branch as $i(t)$. So, $i(t)$ has the form of (7.2).

$$i(t) = K_1 \sin(\omega t + \phi + \alpha) + K_2 e^{-\frac{2t}{R(C_c + 2C_p)}} \quad (7.2)$$

This derivation assumes the input voltage from the antenna has the form of $V_{in(eff)} \cdot \sin(\omega t)$, where ω is an arbitrary frequency constant. The time parameter (t) for $i(t)$ has also been normalized to zero when the switch of figure 7.3 turns on. Therefore, ϕ defines the phase of a sinusoidal cycle when the bridge turns on and α defines the additional phase shift

due to switching. If we further define the initial voltage of the capacitor at the on-set of switching as V_c , the parameters of (7.2) are then given by the following.

$$\phi = \sin^{-1} \frac{(V_{thp} - V_c) \cdot (C_c + 2C_p)}{V_{in(eff)} \cdot C_c} \quad (7.3)$$

$$\alpha = \frac{\pi}{2} - \tan^{-1} \left[\frac{\omega R \cdot (C_c + 2C_p)}{2} \right] \quad (7.4)$$

$$K_1 = V_{in(eff)} \cdot \frac{\omega C_c}{\sqrt{4 + [\omega R \cdot (C_c + 2C_p)]^2}} \quad (7.5)$$

$$K_2 = \frac{V_{thp} - V_{out}}{R} - K_1 \sin(\phi + \alpha) \quad (7.6)$$

If we define the average current to the load as $I_{load(total)}$ and the operating frequency as f , where $f = \omega/2\pi$, we may determine the value of V_c by first determine V_{dt} , the voltage drop across the equivalent capacitance after every half cycle of charge loss due to switching.

$$V_{dt} = \frac{I_{load(total)}}{2f \cdot \left(\frac{1}{2} C_c + C_p \right)} \quad (7.7)$$

For a full-wave bridge with half-bridge components that charge the equivalent capacitor in different direction every half cycle, the initial voltage at the on-set of switching (V_c) simply becomes the average value in (7.8).

$$V_c = \frac{V_{dt}}{2} \quad (7.8)$$

By substituting (7.3~7.8) into (7.2), we may then find the instantaneous voltage as $R \cdot i(t) + V_{out}$. The amplitude of this voltage, defined as V_r in (7.9) turns out to be approximately equal to the no-load voltage, except for a phase shift difference.

$$V_r \approx V_{in(eff)} \cdot \frac{C_c}{C_c + 2C_p} \quad (7.9)$$

7.4 Other Parasitic Components

The most significant parasitic loss effect comes from the finite gate resistance of the transistors and the substrate loss (see figure 3.2). This could be modeled by the quality factor of the gate-to-source and gate-to-bulk capacitances. Once the quality factors are found, the equivalent in-parallel resistances are determined and lumped with the equivalent loss resistance R_{loss} . Practically, the resistance due to the poly-silicon gates can be reduced significantly by introducing sufficient number of gate fingers. Another point of cautious is the selection of driving plate for the coupling capacitor C_c . The top plate driving method, where the input signal is connected to the top plate of C_c , is first considered in figure 7.4.

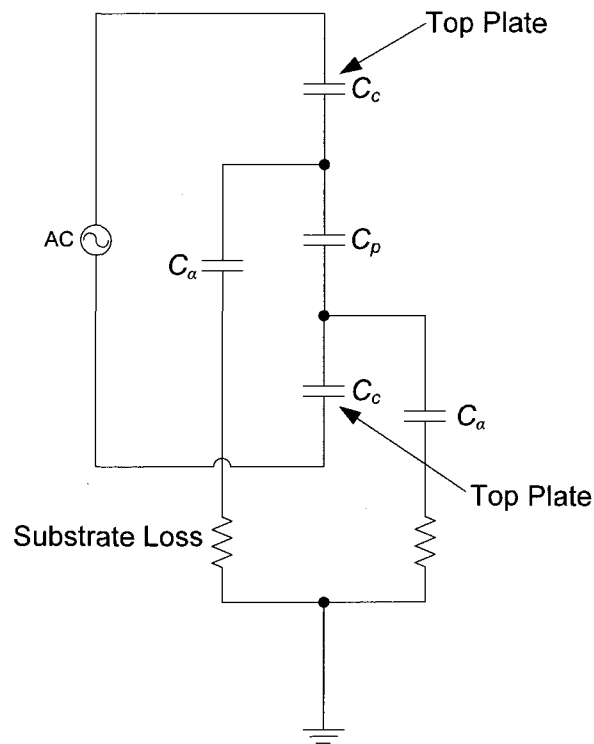


Figure 7.4: Input Driving Top Plate of C_c

In figure 7.4, C_α represents the bottom plate coupling to the substrate. In this case, C_α capacitors are directly across C_p , which reduces V_r according to (7.9). This is not desirable as Q_L (see figure 6.4) must be increased for compensating the voltage loss and the bandwidth is curtailed. Figure 7.5 shows the other case when the bottom plate is driven instead.

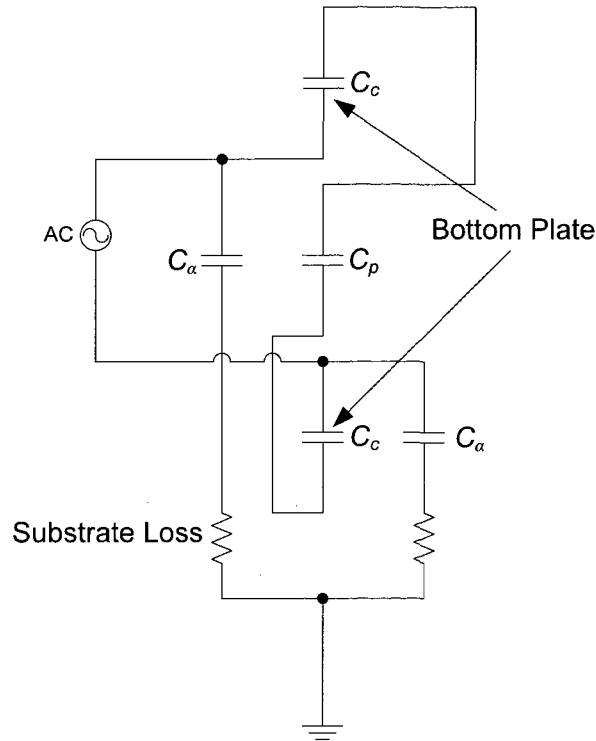


Figure 7.5: Input Driving Bottom Plate of C_c

The negative side-effect does not exist in this case because the C_α capacitors are in parallel with the source. In both case, however, the substrate loss would decrease the PCE of the rectifier circuit. Since the value of C_α is proportional to the size of C_c , either small C_c or capacitors with low C_α values should be used. The MIM type capacitor supported by ST 90nm technology is ideal for this purpose as it uses a lot less area than regular capacitors for achieving the same capacitance.

7.5 Three-Stages CMOS Differential Full-Bridge Rectifier Design

By applying the design method in chapter 6 and the multi-stage adjustment of this chapter, a three-stage DDCFB is designed. So far, we have not considered bandwidth but it suffices saying that it is not an issue. Our rectifier circuit, while being designed for a 10m range has a Q_L (see figure 6.4) value of 4.5, giving a bandwidth in excess of 200MHz, which is considered quite ample when similar applications [21, 22] demand about 100MHz. Following is a list of component values used in our three stage differential bridge rectifier design.

Table 7.2: Transistor Parameters for the 3-stage Rectifier

	Stage 1	Stage 2	Stage 3
W_p (μm)	40.2	40.87	31.35
$p_fingers$	120	122	95
W_n (μm)	9.9	12.92	16.75
$n_fingers$	30	38	50

Table 7.3: Passive Component Values for the 3-stage Rectifier

	Values
L_L (nH)	57.34
C_L (pF)	0.503
$L_{compensate}$ (nH)	165
C_c (pF)	1.2
C_{output} (pF)	1.2

In table 7.2, $p_fingers$ and $n_fingers$ are the number of fingers for the gate contacts. By referencing figure 6.4, L_L and C_L in table 7.3 are L-match component values. C_{output} is the output capacitor for each stage (see figure 7.1). L_L , C_L and $L_{compensate}$ can be embedded with the antenna design and not being part of the IC. All transistors lengths are $0.1\mu\text{m}$.

7.6 PCE and PU at Various Distances

We have simulated the 3-stage rectifier with Cadence Spectre, using design parameters given in table 7.2 and 7.3. In order to simulate PCE and power utilization (PU) at different distance, U_0 values (see figure 6.1) are calculated for each level of available power according to (6.4), with a dipole antenna and the P_{EIRP} of 4W. Table 7.4 summarizes the result.

Table 7.4: PCE and PU for the 3-Stage Rectifier

r (m)	P_a (μ W)	P_a (dBm)	PU	PCE	I_o (μ A)
1	4459.47	6.49	0.20	0.52	900.7
3	495.50	-3.05	0.47	0.65	237.4
5	178.38	-7.49	0.64	0.70	115.3
7	91.01	-10.41	0.72	0.73	65.95
9	55.06	-12.59	0.71	0.74	39.48
10	44.59	-13.51	0.68	0.74	30.61
11	36.86	-14.34	0.63	0.73	23.43
12	30.97	-15.09	0.56	0.72	17.6
13	26.39	-15.79	0.48	0.69	12.81
14	22.75	-16.43	0.38	0.63	8.80
15	19.82	-17.03	0.27	0.56	5.45
16	17.42	-17.59	0.14	0.39	2.55

All results are measured at output voltage of 0.99V. At 10m, PCE is at the maximum of 74%, which is more than three times higher than the existing [21]. The power utilization (PU) is at 68% while output power is 30.3 μ W, which is at least 50 times that of [24] under comparable conditions. This is a three-stage rectifier design but the PCE is still higher than the single stage design of the similar type [19]. At an output power level of 2 μ W, our rectifier can operate as far as 16m, at a P_a level of -17.6dBm. This is equivalent to a power-up threshold of 17.4 μ W. However, our three-stage rectifier is optimized for 10m. Based on our design procedure, a smaller transistor size may be used at 16m for optimized PU and a performance similar to that of table 7.4, such as 68%, can be expected while existing method only yields 25% [21]. It is noticed that the maximum PU occurs at 7m instead of the design 10m. This could be explained by the imperfect piece-wise linear model (see chapter 5), where the modeled resistance is slightly higher

at the conduction peaks when comparing to the resistance of the real transistors. This causes a slight mismatch and the voltage across the rectifier drops accordingly. Since the k ratio now deviates from the ideal k_{max} value, PU becomes less than optimal. However, table 7.4 shows that at 10m, it is only 4% away from the optimal PU. This proves that our design method, which is based on the piece-wise linear approximation of the rectifier, is highly effective in optimizing the power for circuits of this type.

7.7 Comparison with Prior Design Methods

Previous design method by S. Mandal et al [21] focuses on the optimization of threshold power, defined as the input power for an output of $2\mu\text{W}$, where the curve intersects the horizontal axis in figure 7.6. In this sense, our method matches the prior observation that smaller parasitic capacitance lead to lower threshold power. Our method differs in finding the highest output power at a design distance, where only one curve appears at the top. The crossed and dotted-line in figure 7.6 corresponds to the experimental data for the 10m rectifier. It matches the model (solid line) at low distances, only to deviate after 10m, when the non-linear components at high δ values starts to dominate (see 5.3).

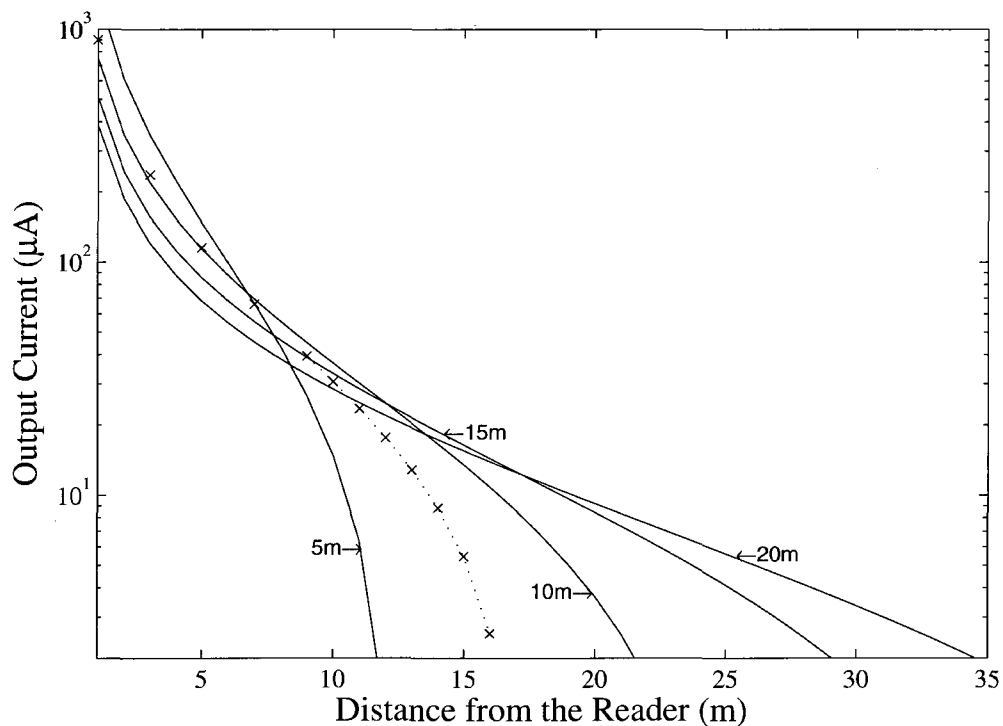


Figure 7.6: Output Current for Rectifiers Optimized for Various Distances

Chapter 8 CONCLUSIONS AND FUTURE WORK

8.1 Conclusions and Contributions

In the process of investigating more power demanding RFID sensor applications for the noisy automotive environment, we seek both the power reduction strategies for circuits and the efficiency enhancement techniques for the power source. The former leads to our proposed power optimization method for Chien search. The novelty of our approach lies in a power efficient factorization algorithm, which allows fine-grain RT-level power management. Circuits for (255, 187, 9) binary BCH code have been synthesized with 0.18 μ m CMOS technology. Simulation shows overall average power savings of 34% at $t = 9$, which double the amount achievable for the method of [8]. Thus, the proposed method is suitable for portable digital video and data storage applications where highly correction-capable BCH codes are used.

The latter direction of efficiency enhancement for the power harvesting process brings us to the rectifier circuit. We have presented a full strategy for the analysis, design and optimization for the differential-drive CMOS full-wave and half-wave bridge rectifier circuit. A piece-wise linear approximation of the rectifier circuit has been investigated. Our work shows that there is an optimal transistor size for a given design range in order that the rectifier may operate efficiently. In addition, we have shown that, contrary to common believe, an increased bulk biasing level, which increases the intensity of body effect, can be strategically utilized for enhancing PCE.

To prove our concept, a three-stage rectifier circuit has been designed for an operating frequency of 915MHz, using ST CMOS 90nm process. Relative to existing design [21], simulation results indicate PCE of 74%, which is at least three times higher. Output power level of 30.3 μ W is also 50 times that of 0.6 μ W in [24], at the same distance, regardless of the significant level of source-to-bulk bias.

8.2 Future Work

A low power RFID tag for sensor applications include all typical tag components namely rectifier, modulator, demodulator, decoder, encoder, state-machine, power control, oscillator and memory. We have studied the design theory for a type of rectifier circuit known as differential-drive CMOS rectifier and simulation results have shown drastic improvement in terms of power conversion efficiency and the amount of output current. However, a circuit has yet to be implemented on silicon level. Furthermore, efficiency is only optimized at one distance. A self reconfigurable rectifier circuit may solve the problem but this is left for future study. In addition, since we have investigated BCH decoder as a potential solution for error-correction in noisy environment, the integration of this circuit becomes a concern. In this sense, not only Chien search, which is the subject of this thesis but other parts of the decoder also need to be brought in. Certainly, the rest of the typical tag components still require low power design work or optimization studies.

The last but not the least is the sensor application, which may require not just a state-machine but a full micro-controller engine with more memory. Traditionally, memory has been a big source of power dissipation and low power design techniques, when applied to the core memory architecture and charge pump design should help making the integration effort feasible. The design of the micro-controller may involve the use of innovative low power method not otherwise require in common applications, when considering the extreme low power condition. While the processing unit is important, sensor data may only be effectively communicated with non-standard RFID protocols, which possibly demand not just engineering of the protocol but a more complex and reliable addressing and collision detection scheme.

REFERENCES

- [1] K. Finkenzeller, "RFID Handbook-2nd edition," John Wiley and Sons Ltd., 2003
- [2] W. Weber, C. Braun, R. Glaser, Y. Gsottberger, M. Halik, S. Jung, H. Klauk, C. Lauterbach, G. Schmid, X. Shi, T.F. Sturm, G. Stromberg, U. Zschieschang, "Ambient Intelligence – Key Technologies in the Information Age," *2003 IEEE International Electron Devices Meeting (IEDM'03)*, Dec. 2003, pp.111-118
- [3] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Chapter 6, Prentice Hall, 1983, ISBN 0-13-283796-X
- [4] H. Chang, C. Lin and C. Lee, "A Low Power Reed Solomon Decoder for STM-16 Optical Communications," *Proc. of IEEE Asia-Pacific Conf. on ASIC*, Aug. 2002, pp. 351-354.
- [5] L. Song, M. Yu and M. S. Shaffer, "10- and 40-Gb/s Forward Error Correction Devices for Optical Communications," *IEEE J. of Solid-State Circuits*, vol. 37, no. 11, Nov. 2002, pp. 1565-1573.
- [6] A. Raghupathy and K. J. R. Liu, "Algorithm-Based Low-Power/High-Speed Reed-Solomon Decoder Design," *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, vol. 47, no. 11, Nov. 2000, pp. 1254-1270.
- [7] R. T. Chien, "Cyclic Decoding Procedure for the Bose-Chaudhuri -Hocquenghem Codes," *IEEE Trans. Information Theory*, IT-10, October 1964, pp. 357-363.
- [8] Y. Wu, "Low Power Decoding of BCH Codes," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2, May 2004, pp. II-369-372.
- [9] W. Liu, J. Rho and W. Sung, "Low-Power High-Throughput BCH Error Correction VLSI Design for Multi-Level Cell NAND Flash Memories," *Proc. of IEEE Workshop on Signal Processing Systems (SIPS)*, Oct. 2006, pp. 303-308.

- [10] Z. Yu, "Low Power Finite Field Multiplication and Division in Reconfigurable Reed-Solomon Codec," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS), May 2002*, vol. 5, pp V-785-788.
- [11] W. P. L. Soon and W. C. Wong, "Robust H.263 Portable Video Coding," *Proc. of Information, Communications and Signal Processing*, September 1997, vol. 1, pp. 306-310.
- [12] J. Cho and W. Sung, "Software Implementation of Chien Search Process for Strong BCH Codes," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS), May 2008*, pp. 1842-1845.
- [13] A.S.W. Man, E.S. Zhang, V.K.N. Lau, C.Y. Tsui and H.C. Luong, "Low Power VLSI Design for a RFID Passive Tag baseband System Enhanced with an AES Cryptography Engine," *1st Annual RFID Eurasia*, Sept. 2007, pp. 1-6
- [14] F. Cilek, K. Seemann, D. Brenk, J. Essel, J. Heidrich, R. Weigel, G. Holweg, "Ultra Low Power Oscillator for UHF RFID Transponder," *2008 IEEE International Frequency Control Symposium*, May 2008, pp. 418-421
- [15] J.H. Lee, G.H. Lim, J.H. Kim, M.H. Park, K.H. Jim, J.W. Cha, P.B. Ha, Y.J. Gang, Y.H. Kim, "A Low-Power 512-Bit EEPROM Design for UHF RFID Tag Chips," *Proc. of the 7th International Conference on Computational Science (ICCS 2007)*, Part IV, Beijing, China, pp. 721-724
- [16] V. Pillai, H. Heinrich, D. Dieska, P.V. Nikitin, R. Martinez and K.V.S. Rao, "An Ultra-Low-Power Long Range Battery/Passive RFID Tag for UHF and Microwave Bands with a Current Consumption of 700nA at 1.5V," *IEEE Trans. on Circuits and Systems-I:Regular Papers*, Vol. 54, No.7, July 2007, pp. 1500-1512
- [17] S. Zhou, N. Wu, "A Novel Ultra Low Power Temperature Sensor for UHF RFID Tag Chip," *2007 IEEE Asian Solid-State Circuits Conference (ASSCC'07)*, Nov. 2007, pp. 464-467
- [18] U. Karthans and M. Fischer, "Fully Integrated Passive UHF RFID Transponder IC with 16.7 μ W Minimum RF Input Power," *IEEE Journal of Solid-State Circuits*, Vol. 38, No. 10, October 2003, pp. 1602-1608

- [19] A. Sasaki, K. Kotani, T. Ito, "Differential-Drive CMOS Rectifier for UHF RFIDs with 66% PCE at -12dBm Input," *2008 IEEE Asian Solid-State Circuits Conference (ASSCC'08)*, Nov. 2008, pp. 105-108
- [20] J. Yi, W. Ki, C. Tsui, "Analysis and Design Strategy of UHF Micro-Power CMOS Rectifiers for Micro-Sensor and RFID Applications," *IEEE Trans. on Circuits and Systems-I: Regular Papers*, Vol. 54, No.1, January 2007, pp.153-166
- [21] S. Mandal, R. Sarpeshkar, "Low-Power CMOS Rectifier Design for RFID Applications," *IEEE Trans. on Circuits and Systems-I:Regular Papers*, Vol. 54, No.6, June 2007, pp. 1177-1188
- [22] A. Shameli, A. Safarian, A. Rofougaran, et al., "Power Harvester Design for Passive UHF RFID Tag Using a Voltage Boosting Technique," *IEEE Trans. on Microwave Theory and Techniques*, Vol. 55, Issue 6, Part 1, June 2007, pp.1089-1097
- [23] H. Nakamoto, D. Yamazaki, T. Yamamoto, H. Kurata, S. Yamada, K. Mukaida et al., "A Passive UHF RF Identification CMOS Tag IC Using Ferroelectric RAM in 0.35- μ m Technology," *IEEE Journal of Solid-State Circuits*, Vol. 42, No.1, January 2007, pp. 101-110
- [24] T. Umeda, H. Yoshida, S. Sekine, Y.Fujita, et al., "A 950-MHz Rectifier Circuit for Sensor Network Tags With 10-m Distance," *IEEE Journal of Solid-State Circuits*, Vol. 41, No. 1, January 2006, pp. 35-41
- [25] K. Kotani, T. Ito, "High Efficiency CMOS Rectifier Circuit with Self-Vth-Cancellation and Power Regulation Functions for UHF RFIDs," *2007 IEEE Asian Solid-State Circuits Conference (ASSCC'07)*, Nov. 07, pp. 119-122
- [26] C. Ma, C. Zhang, Z. Wang, "Power Analysis for the MOS AC/DC Rectifier of Passive RFID Transponders," *2006 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS'06)*, Dec. 06, pp.1350-1353
- [27] Edward S. Yang, "Microelectronics Devices," McGraw-Hill Book Company, 1988, ISBN 0-07-072238-2

- [28] J. Inwhee, "A Low-Power Hybrid ARQ Scheme for the RFID System," *Proc. of 2nd International High Performance Computing and Communications Conference (HPCC 2006)*, Lecture Notes in Computer Science Vol.4208, 13-15 Sept. 2006, Munich, Germany, pp. 535-541
- [29] S.Y. Wong, C. Chen and Q.M.J. Wu, "Power-Management-Based Chien Search for Low Power BCH Decoder," in Proc. 14th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), Aug. 19-21 2009, pp. 299-302, © 2009 Association for Computing Machinery, Inc. Reprinted by Permission.
- [30] © 2009 IEEE. Reprinted, with permission, from IEEE Transactions on Very Large Scale Integration Systems (TVLSI), "Low Power Chien Search for BCH Decoder Using RT-Level Power Management, S.Y. Wong, C. Chen and Q.M. J. Wu

APPENDIX A: RTL CODES FOR CHIEN SEARCH

In this appendix, both the RTL codes for the conventional Chien search and that of the proposed low power Chien search have been recorded. The RTL codes were written in Verilog and their functionality were verified with the respective test bench codes also written in Verilog. The Verilog testbenches require simulation random data source formatted as BCH code words. The code words are generated by MATLAB and the files are included at the end of this appendix. Following are the organization of the files.

Conventional Chien Search:

chien_search.v -- main source code
errornot.v -- conventional adder
malpha.v ~ malpha9.v -- constant Galois field multipliers
rectest.v -- testbench

Proposed Low-Power Chien Search:

lp3chien.v -- main source code
lp3error.v -- low power adder + POR
lpalpha.v ~ lpalpha9.v -- constant Galois field multipliers
lp3test.v -- testbench

MATLAB Source Codes:

retestvec.m -- generate simdata.dat for the Verilog testbenches
syndrome.m -- syndrome calculation, called by retestvec.m
berlekamp.m -- the algorithm for generating error polynomials
read_h.m -- generate parity matrix for (255,187,9) BCH code

A.1 RTL Codes for the Conventional Chien Search

chien_search.v:

```
//The regular Chien Search
//Main State Machine

module chien_search (rst,load,mout,data_en,epoly,CK,mess_in);

    // I_O ports definition
    input      rst;          //reset signal from the Berlekamp stage
    input      CK;          //clock input from the Berlekamp stage
    input      mess_in;     //the most significant bit of the input
message
    input [1:9] epoly;      //bit-serial inputs for the error
polynomial
    output     load;        //loading signal for the error
polynomial
    output     mout;        //error corrected output data
stream
    output     data_en;     //data message bit enable signal

    // signal type definitions
    wire       rst,sysclk,mess_in,fec;
    wire [1:9] epoly;
    reg        load,mout,data_en,load_dis;

    // signals from the "malphax" to "errornot"
    wire [0:7] m1,m2,m3,m4,m5,m6,m7,m8,m9;

    // state counters
    reg [2:0]  iscounter;   //multipliers internal states
    reg [1:0]  cstate;      //Chien Search state counter
    reg [7:0]  mcounter;    //message bit counter

    // State constants of the Chien Search
    parameter  reset_state=2'b00,          //state after power-up
or reset
```

```

        fstate=2'b01,           //process the first bit
        ostate=2'b11,         //process other bits
        lsigma_state=2'b10;    //the state for loading
error polynomial

// error correct the mess_in with output from errornot
always @ (posedge CK)
    if (rst==1)
        mout <= 0;
    else
        if (data_en==1)
            mout <= fec ^ mess_in;
        else
            mout <= mout;

// multiplier internal state machine
always @ (posedge CK)
    if ((rst==1)|| (load_dis==1))
        iscounter <=3'b000;
    else
        case (iscounter)
            3'b000:    iscounter<=3'b001;
            3'b001:    iscounter<=3'b011;
            3'b011:    iscounter<=3'b010;
            3'b010:    iscounter<=3'b110;
            3'b110:    iscounter<=3'b111;
            3'b111:    iscounter<=3'b101;
            3'b101:    iscounter<=3'b100;
            3'b100:    iscounter<=3'b000;
            default:    iscounter<=3'b000;
        endcase

// message bit counting state machine
always @ (posedge CK)
    if (rst==1)
        mcounter <= 8'h00;
    else

```

```

        if (((iscounter <= 3'b001) && ((cstate ==
fstate)|| (cstate == ostate))) || ((cstate==lsigma_state) && (iscounter
== 3'b000)))
            `include "gray_code.v"
        else
            mcounter <= mcounter;

// Chien Search State Machine
always @ (posedge CK)
    begin
        if (rst==1)
            begin
                load <= 0;
                data_en <= 0;
                load_dis <= 0;
                cstate <= reset_state;
            end
        else
            case (cstate)
                reset_state:
                    begin
                        data_en <= 0;
                        load_dis <= 0;
                        load <= load;
                        cstate <= cstate;
                        if (iscounter==3'b100)
                            if (load==0)
                                load <= 1;
                            else
                                begin
                                    cstate <=
fstate;
                                    load <= 0;
                                end
                            end
                    end

                fstate:
                    begin

```

```

data_en <= 0;
load <= 0;
load_dis <= load_dis;
cstate <= cstate;
if (iscounter==3'b000)
    begin
        cstate <= ostate;
        data_en <= 1;
        load_dis <= 1;
    end
end
end

ostate:
begin
    data_en <= 1;
    load_dis <= 1;
    load <= 0;
    cstate <= cstate;
    if (mcounter==8'h81)
        begin
            load <= 1;
            load_dis <= 0;
            cstate <=
lsigma_state;
        end
    end
end

lsigma_state:
begin
    data_en <= 0;
    load_dis <= 0;
    load <= 1;
    cstate <= lsigma_state;
    if (iscounter==3'b100)
        begin
            load <= 0;
            cstate <= fstate;
        end
    end
end

```



```

end

default:    cstate <= reset_state;
endcase

end

//instantiating the mutipliers
malpha ma1 (
    .sigma_in(epoly[1]),
    .product_out(m1),
    .load(load),
    .clk(CK)
);

malpha2 ma2 (
    .sigma_in(epoly[2]),
    .product_out(m2),
    .load(load),
    .clk(CK)
);

malpha3 ma3 (
    .sigma_in(epoly[3]),
    .product_out(m3),
    .load(load),
    .clk(CK)
);

malpha4 ma4 (
    .sigma_in(epoly[4]),
    .product_out(m4),
    .load(load),
    .clk(CK)
);

malpha5 ma5 (
    .sigma_in(epoly[5]),
    .product_out(m5),

```

```

        .load(load),
        .clk(CK)
    );

malpha6 ma6 (
    .sigma_in(epoly[6]),
    .product_out(m6),
    .load(load),
    .clk(CK)
);

malpha7 ma7 (
    .sigma_in(epoly[7]),
    .product_out(m7),
    .load(load),
    .clk(CK)
);

malpha8 ma8 (
    .sigma_in(epoly[8]),
    .product_out(m8),
    .load(load),
    .clk(CK)
);

malpha9 ma9 (
    .sigma_in(epoly[9]),
    .product_out(m9),
    .load(load),
    .clk(CK)
);

//instantiating the sum of product device for error correction
errornot errorc (
    .m1(m1),
    .m2(m2),
    .m3(m3),
    .m4(m4),

```

```

        .m5(m5),
        .m6(m6),
        .m7(m7),
        .m8(m8),
        .m9(m9),
        .corrt(fec)
    );
endmodule

```

errornot.v:

```

//The module for summing all product terms together
//for determining the status of a bit, whether it
//is received in error or not
module errornot (m1,m2,m3,m4,m5,m6,m7,m8,m9,corrt);

    // I_O port definitions
    input [0:7] m1,m2,m3,m4,m5,m6,m7,m8,m9; //products from the
mutipliers
    output          corrt; //correction true
or not

    // sign type definitions
    wire [0:7] m1,m2,m3,m4,m5,m6,m7,m8,m9;
    reg          corrt;
    reg [0:7] sb; //contains the sum of all
terms

    // variable used
    integer          sloop; //for iterating
all the bits

    always @ (m1 or m2 or m3 or m4 or m5 or m6 or m7 or m8 or m9)

        begin

            sb[0]=(((m1[0]^m2[0])^(m3[0]^m4[0]))^((m5[0]^m6[0])^(m7[0]^m8[0])
))^ (m9[0]^1'b1);

```

```

        for (sloop=1; sloop<8; sloop=sloop+1)

            sb[sloop]=(((m1[sloop]^m2[sloop])^(m3[sloop]^m4[sloop]))^(m5[sloop]^m6[sloop])^(m7[sloop]^m8[sloop]))^m9[sloop];
            corrt = ~|sb;
        end
    endmodule

```

malpha.v:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this multiplier multiplies an 8-bit number "sigma" by alpha
module malpha (sigma_in, product_out, load, clk);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal
    output [0:7] product_out; //8-bit parallel product

output

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

    // main process of multiplication
    always @ (posedge clk)
        if (load==1)
            begin
                S <= (S>>1);
                S[0] <= sigma_in;
            end
        else

```

```

        begin
            S[5:7] <= S[4:6];
            S[4] <= S[3]^S[7];
            S[3] <= S[2]^S[7];
            S[2] <= S[1]^S[7];
            S[1] <= S[0];
            S[0] <= S[7];
        end
endmodule

```

malpha2:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha^2
module malpha2 (sigma_in, product_out, load, clk);
    // I_O port definitions
    input      sigma_in; //bit-serial input for the sigma
    input      load;     //load enable signal for sigma
    input      clk;     //input clock signal
    output [0:7] product_out; //8-bit parallel product
output

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

    // main process of mutiplication
    always @ (posedge clk)
        if (load==1)
            begin
                S <= (S>>1);
            end
endmodule

```

```

                S[0] <= sigma_in;
            end
        else
            begin
                S[7] <= S[5];
                S[6] <= S[4];
                S[5] <= S[3]^S[7];
                S[4] <= S[2]^S[6]^S[7];
                S[3] <= S[1]^S[6]^S[7];
                S[2] <= S[0]^S[6];
                S[1] <= S[7];
                S[0] <= S[6];
            end
        endmodule

```

malpha3:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this multiplier multiplies an 8-bit number "sigma" by alpha^3
module malpha3 (sigma_in, product_out, load, clk);
    // I_O port definitions
    input      sigma_in; //bit-serial input for the sigma
    input      load;     //load enable signal for sigma
    input      clk;      //input clock signal
    output [0:7] product_out; //8-bit parallel product
output

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

```

```

// main process of mutiplication
always @ (posedge clk)
    if (load==1)
        begin
            S <= (S>>1);
            S[0] <= sigma_in;
        end
    else
        begin
            S[7] <= S[4];
            S[6] <= S[3]^S[7];
            S[5] <= S[2]^S[6]^S[7];
            S[4] <= S[1]^S[5]^S[6]^S[7];
            S[3] <= S[0]^S[5]^S[6];
            S[2] <= S[5]^S[7];
            S[1] <= S[6];
            S[0] <= S[5];
        end
endmodule

```

malpha4:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha^4
module malpha4 (sigma_in, product_out, load, clk);
    // I_O port definitions
    input      sigma_in; //bit-serial input for the sigma
    input      load;     //load enable signal for sigma
    input      clk;      //input clock signal
    output [0:7] product_out; //8-bit parallel product
output

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;

```

```

reg [0:7] S;

// connecting the product register to output
assign product_out=S;

// main process of mutiplication
always @ (posedge clk)
    if (load==1)
        begin
            S <= (S>>1);
            S[0] <= sigma_in;
        end
    else
        begin
            S[7] <= S[3]^S[7];
            S[6] <= S[2]^S[6]^S[7];
            S[5] <= S[1]^S[5]^S[6]^S[7];
            S[4] <= S[0]^S[4]^S[5]^S[6];
            S[3] <= S[4]^S[5]^S[7];
            S[2] <= S[4]^S[6];
            S[1] <= S[5];
            S[0] <= S[4];
        end
endmodule

```

malpha5:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha^5
module malpha5 (sigma_in, product_out, load, clk);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal

```



```

        output [0:7]      product_out;      //8-bit parallel product
output

// signal type definitions
wire      sigma,load,clk;
wire [0:7] product_out;
reg [0:7]  S;

// connecting the product register to output
assign product_out=S;

// main process of mutiplication
always @ (posedge clk)
    if (load==1)
        begin
            S <= (S>>1);
            S[0] <= sigma_in;
        end
    else
        begin
            S[7] <= S[2]^S[6]^S[7];
            S[6] <= S[1]^S[5]^S[6]^S[7];
            S[5] <= S[0]^S[4]^S[5]^S[6];
            S[4] <= S[3]^S[4]^S[5];
            S[3] <= S[3]^S[4]^S[6]^S[7];
            S[2] <= S[3]^S[5]^S[7];
            S[1] <= S[4];
            S[0] <= S[3]^S[7];
        end
endmodule

```

malpha6:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha^6

```

```

module malpha6 (sigma_in, product_out, load, clk);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal
    output [0:7] product_out; //8-bit parallel product
output

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

    // main process of mutiplication
    always @ (posedge clk)
        if (load==1)
            begin
                S <= (S>>1);
                S[0] <= sigma_in;
            end
        else
            begin
                S[7] <= S[1]^S[5]^S[6]^S[7];
                S[6] <= S[0]^S[4]^S[5]^S[6];
                S[5] <= S[3]^S[4]^S[5];
                S[4] <= S[2]^S[3]^S[4];
                S[3] <= S[2]^S[3]^S[5]^S[6];
                S[2] <= S[2]^S[4]^S[6]^S[7];
                S[1] <= S[3]^S[7];
                S[0] <= S[2]^S[6]^S[7];
            end
        end
endmodule

```

malpha7:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this multiplier multiplies an 8-bit number "sigma" by alpha^7
module malpha7 (sigma_in, product_out, load, clk);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal
    output [0:7] product_out; //8-bit parallel product
output

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

    // main process of multiplication
    always @ (posedge clk)
        if (load==1)
            begin
                S <= (S>>1);
                S[0] <= sigma_in;
            end
        else
            begin
                S[7] <= S[0]^S[4]^S[5]^S[6];
                S[6] <= S[3]^S[4]^S[5];
                S[5] <= S[2]^S[3]^S[4];
                S[4] <= S[1]^S[2]^S[3]^S[7];
                S[3] <= S[1]^S[2]^S[4]^S[5];
                S[2] <= S[1]^S[3]^S[5]^S[6];
                S[1] <= S[2]^S[6]^S[7];
                S[0] <= S[1]^S[5]^S[6]^S[7];
            end
endmodule

```

```

                                end
endmodule

malpha8:

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha^8
module malpha8 (sigma_in, product_out, load, clk);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal
    output [0:7]  product_out;    //8-bit parallel product
output

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    reg [0:7]  S;

    // connecting the product register to output
    assign product_out=S;

    // main process of mutiplication
    always @ (posedge clk)
        if (load==1)
            begin
                S <= (S>>1);
                S[0] <= sigma_in;
            end
        else
            begin
                S[7] <= S[3]^S[4]^S[5];
                S[6] <= S[2]^S[3]^S[4];
                S[5] <= S[1]^S[2]^S[3]^S[7];
                S[4] <= S[0]^S[1]^S[2]^S[6];
            end
    end
endmodule

```

```

        S[3] <= S[0]^S[1]^S[3]^S[4];
        S[2] <= S[0]^S[2]^S[4]^S[5]^S[7];
        S[1] <= S[1]^S[5]^S[6]^S[7];
        S[0] <= S[0]^S[4]^S[5]^S[6];
    end
endmodule

```

malpha9:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this multiplier multiplies an 8-bit number "sigma" by alpha^9
module malpha9 (sigma_in, product_out, load, clk);
    // I_O port definitions
    input      sigma_in; //bit-serial input for the sigma
    input      load;     //load enable signal for sigma
    input      clk;      //input clock signal
    output [0:7] product_out; //8-bit parallel product
output

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

    // main process of multiplication
    always @ (posedge clk)
        if (load==1)
            begin
                S <= (S>>1);
                S[0] <= sigma_in;
            end
        else
            begin

```

```

        S[7] <= S[2]^S[3]^S[4];
        S[6] <= S[1]^S[2]^S[3]^S[7];
        S[5] <= S[0]^S[1]^S[2]^S[6];
        S[4] <= S[0]^S[1]^S[5];
        S[3] <= S[0]^S[2]^S[3]^S[7];
        S[2] <= S[1]^S[3]^S[4]^S[6]^S[7];
        S[1] <= S[0]^S[4]^S[5]^S[6];
        S[0] <= S[3]^S[4]^S[5];
    end
endmodule

```

rectest.v:

```

//Testbench for the regular Chien Search Engine
`timescale 1ns/10ps
module rectest;

    // Test signals definitions
    reg        rst;           //system reset signal
    reg        sysclk;       //system clock signal
    reg [3:0]  errnum;       //register containing the number of
error bits per block
    reg [0:7]  sigma1,sigma2; //registers for the error
polynomial
    reg [0:7]  sigma3,sigma4;
    reg [0:7]  sigma5,sigma6;
    reg [0:7]  sigma7,sigma8;
    reg [0:7]  sigma9;
    wire       load;         //data enable for the sigma terms
    wire       m2c;         //streaming data to the Chien Search
    wire       c2t;         //corrected data stream from the Chien
Search
    wire       data_en;     //data enable for the streaming data
input
    wire [1:9] epoly;

    // Source Registers
    reg [0:186] orig_mess; //holding the error-free original data

```

```

        reg [0:254] source_mess;      //holding the data to the Chien
Search
        reg [0:254] corrat_mess;     //holding the error corrected
message from Chien Search

        // Other Registers used by the tester only
        reg [0:254] temp_source;     //buffers for transient data to the
Chien Search
        reg [0:186] temp_orig; //and to the tester
        reg [0:7]  temp1,temp2;     //temp buffers used during test
array data transfer

        // Source Test Vector array specific variables
        integer      index;          //index to the current block
of test vectors
        integer      dloop;          //data fetch loop variables
        integer      bc;
        integer      simdone;        //indicate the completion of a
simulation
        integer      lastbit;        //indicate readiness of a new block
        integer      afreset;        //is this right after reset?
        parameter    slist=65*50;    //length of the test vectors
        reg [0:7]    sim_data [0:slist-1]; //test vector array

        // connect source_mess "epoly" to the Chien Search Engine
        assign m2c=source_mess[254];
        assign
epoly={sigma1[7],sigma2[7],sigma3[7],sigma4[7],sigma5[7],sigma6[7],sigm
a7[7],sigma8[7],sigma9[7]};

        //Initializing the Testbench
        //by filling the test vector array first
        //and generate a reset pulse in the meantime for initializing the
Chien Search Engine
        initial
            begin
                rst=1;
                $readmemb("simdata.dat",sim_data);

```

```

        #84  rst=0;
        $dumpfile ("chien_search.dump");
        $dumpvars(1,recserh);
    end

//reseting the array pointer and starting the system clock
initial
    begin
        index=0;
        simdone=0;
        sysclk=0;
        lastbit=0;
        afreset=1;
        bc=0;
        forever
            #10  sysclk = ~sysclk;
        end

//Define the running tasks
//load the next test vector
always @ (posedge load)
    if (index*65>=slist)
        simdone=1;
    else
        begin
            //fetch the next original message for
verification
            for (dloop=0; dloop<23 ;dloop=dloop+1)

                {temp_orig[dloop*8],temp_orig[dloop*8+1],temp_orig[dloop*8+2],tem
p_orig[dloop*8+3],temp_orig[dloop*8+4],temp_orig[dloop*8+5],temp_orig[d
loop*8+6],temp_orig[dloop*8+7]}=sim_data[index*65+dloop];
                temp1=sim_data[index*65+23];
                temp_orig[184:186] = temp1[0:2];

            //fetch the number of errors for the next block
            errnum <= temp1[4:7];
        end
    end

```



```

        //fetch the next error polynomial
        sigma1=sim_data[index*65+24];
        sigma2=sim_data[index*65+25];
        sigma3=sim_data[index*65+26];
        sigma4=sim_data[index*65+27];
        sigma5=sim_data[index*65+28];
        sigma6=sim_data[index*65+29];
        sigma7=sim_data[index*65+30];
        sigma8=sim_data[index*65+31];
        sigma9=sim_data[index*65+32];

        //fetch the next corrupted message block
        for (dloop=0; dloop<31; dloop=dloop+1)

            {temp_source[dloop*8],temp_source[dloop*8+1],temp_source[dloop*8+
2],temp_source[dloop*8+3],temp_source[dloop*8+4],temp_source[dloop*8+5]
,temp_source[dloop*8+6],temp_source[dloop*8+7]}=sim_data[index*65+33+dloop];

            temp2=sim_data[index*65+64];
            temp_source[248:254]=temp2[0:6];

            //go to the next block
            index <= index+1;
            $display("fetch test vector %d",index);
        end

//storing output from the Chien Search Engine
always @ (posedge sysclk)
    if (data_en==1)
        begin
            bc <= bc+1;
            corrat_mess[0] <= c2t;
            corrat_mess[1:254] <=corrat_mess[0:253];
            if (load==1)
                lastbit<=2;
        end
    else
        begin

```

```

corrat_mess <= corrat_mess;
if (lastbit==2)
    begin
        corrat_mess[0] <= c2t;
        corrat_mess[1:254] <=
corrat_mess[0:253];
        lastbit<=1;
    end
else if (lastbit==1)
    begin
        lastbit<=0;
        orig_mess<=temp_orig;
        source_mess<=temp_source;
        //if (corrat_mess[68:254] !=
orig_mess)
        //    begin
            $display("block bit
count=%d",bc);
            bc <= 0;
            $display("error
position");

            $display("%b",orig_mess^corrat_mess[68:254]);
            //    end
            if (simdone==1)
                begin
                    $dumpoff;
                    $display("vector
index=%d, list length=%d ",index,slist);
                    $display("simulation
completed!");
                    $finish;
                end
            end
    end
else
    if ((afreset==1)&&(load==1))
        begin
            orig_mess<=temp_orig;

```

```

        source_mess<=temp_source;
        afreset<=0;
    end

    end

//load the error polynomial at the beginning of every message bit
always @ (posedge sysclk)
    if (load==1)
        begin
            sigma1 <= (sigma1>>1);
            sigma2 <= (sigma2>>1);
            sigma3 <= (sigma3>>1);
            sigma4 <= (sigma4>>1);
            sigma5 <= (sigma5>>1);
            sigma6 <= (sigma6>>1);
            sigma7 <= (sigma7>>1);
            sigma8 <= (sigma8>>1);
            sigma9 <= (sigma9>>1);
        end
    else
        begin
            sigma1 <= sigma1;
            sigma2 <= sigma2;
            sigma3 <= sigma3;
            sigma4 <= sigma4;
            sigma5 <= sigma5;
            sigma6 <= sigma6;
            sigma7 <= sigma7;
            sigma8 <= sigma8;
            sigma9 <= sigma9;
        end

//streaming data to the Chien Search Engine
always @ (posedge sysclk)
    if (data_en==1)
        source_mess <= (source_mess>>1);

```

```
//instantiating the Chien Search Engine
chien_search recserh (
    .rst(rst),
    .load(load),
    .mout(c2t),
    .data_en(data_en),
    .epoly(epoly),
    .CK(sysclk),
    .mess_in(m2c)
);
endmodule
```

A.2 RTL Codes for the Proposed Low-Power Chien Search

lp3chien.v:

```
//The Low power Chien Search
//Main State Machine

module lp3chien (rst,load,mout,data_en,epoly,CK,mess_in);

    // I_O ports definition
    input      rst;          //reset signal from the Berlekamp stage
    input      CK;          //clock input from the Berlekamp stage
    input      mess_in;     //the most significant bit of the input
message
    input [1:9] epoly;      //bit-serial inputs for the error
polynomial
    output     load;        //loading signal for the error
polynomial
    output     mout;        //error corrected output data
stream
    output     data_en;     //data message bit enable signal

    // signal type definitions
    wire       rst,sysclk,mess_in,cec;
    wire [1:9] epoly;
    reg        load,mout,data_en,load_dis;
    wire       pload;
    reg        mout_dis;

    // signals from the "lpalphax" to "lperror" and vice versa
    wire [0:7] m1,m2,m3,m4,m5,m6,m7,m8,m9;
    wire [0:7] to1,to2,to3,to4,to5,to6,to7;

    // state counters
    reg [2:0]  iscounter;   //multipliers internal states
    reg [1:0]  cstate;      //Chien Search state counter
    reg [7:0]  mcounter;    //message bit counter
```

```

// State constants of the Chien Search
parameter reset_state=2'b00, //state after power-up
or reset
        fstate=2'b01, //process the first bit
        ostate=2'b11, //process other bits
        lsigma_state=2'b10; //the state for loading
error polynomial

assign pload=((cstate==lsigma_state)|| (cstate==fstate)) ? 0:fec;

// error correct the mess_in with output from errornot
always @ (posedge CK)
    if (rst==1)
        mout <= 0;
    else
        if ((data_en==1) && (mout_dis==0))
            mout <= fec ^ mess_in;
        else
            mout <= mout;

// multiplier internal state machine
always @ (posedge CK)
    if ((rst==1)|| (load_dis==1))
        iscounter <=3'b000;
    else
        case (iscounter)
            3'b000: iscounter<=3'b001;
            3'b001: iscounter<=3'b011;
            3'b011: iscounter<=3'b010;
            3'b010: iscounter<=3'b110;
            3'b110: iscounter<=3'b111;
            3'b111: iscounter<=3'b101;
            3'b101: iscounter<=3'b100;
            3'b100: iscounter<=3'b000;
            default: iscounter<=3'b000;
        endcase

```

```

// message bit counting state machine
always @ (posedge CK)
    if (rst==1)
        mcounter <= 8'h00;
    else
        if ((iscounter == 3'b000) && (cstate != reset_state)
&& (pload==0))
            `include "gray_code.v"
        else
            mcounter <= mcounter;

// Chien Search State Machine
always @ (posedge CK)
    begin
        if (rst==1)
            begin
                load <= 0;
                data_en <= 0;
                load_dis <= 0;
                mout_dis <= 0;
                cstate <= reset_state;
            end
        else
            case (cstate)
                reset_state:
                    begin
                        data_en <= 0;
                        load_dis <= 0;
                        load <= load;
                        cstate <= cstate;
                        mout_dis <= 0;
                        if (iscounter==3'b100)
                            if (load==0)
                                load <= 1;
                            else
                                begin
                                    cstate <=
fstate;

```

```

load <= 0;
load_dis <=
1;

end

end

fstate:
begin
data_en <= 1;
load <= 0;
load_dis <= 1;
mout_dis <= 0;
cstate <= ostate;
end

ostate:
begin
data_en <= ~fec;
load_dis <= 1;
load <= 0;
cstate <= cstate;
mout_dis <= fec;
if (mcounter==8'h81)
begin
load <= ~fec;
load_dis <= fec;
if (fec==0)
cstate <=
lsigma_state;
end

end

lsigma_state:
begin
data_en <= 0;
load_dis <= 0;
load <= 1;
mout_dis <= 0;

```



```

        cstate <= lsigma_state;
        if (iscounter==3'b100)
            begin
                load <= 0;
                load_dis <= 1;
                cstate <= fstate;
            end
        end

        end

        default:    cstate <= reset_state;
    endcase

end

//instantiating the mutipliers
lalpha ma1 (
    .sigma_in(epoly[1]),
    .product_out(m1),
    .load(load),
    .clk(CK),
    .pdata(to1),
    .pload(pload)
);

lalpha2 ma2 (
    .sigma_in(epoly[2]),
    .product_out(m2),
    .load(load),
    .clk(CK),
    .pdata(to2),
    .pload(pload)
);

lalpha3 ma3 (
    .sigma_in(epoly[3]),
    .product_out(m3),
    .load(load),
    .clk(CK),
    .pdata(to3),

```

```

        .pload(pload)
    );

lpalpha4 ma4 (
    .sigma_in(epoly[4]),
    .product_out(m4),
    .load(load),
    .clk(CK),
    .pdata(to4),
    .pload(pload)
);

lpalpha5 ma5 (
    .sigma_in(epoly[5]),
    .product_out(m5),
    .load(load),
    .clk(CK),
    .pdata(to5),
    .pload(pload)
);

lpalpha6 ma6 (
    .sigma_in(epoly[6]),
    .product_out(m6),
    .load(load),
    .clk(CK),
    .pdata(to6),
    .pload(pload)
);

lpalpha7 ma7 (
    .sigma_in(epoly[7]),
    .product_out(m7),
    .load(load),
    .clk(CK),
    .pdata(to7),
    .pload(pload)
);

```

```

lpalpha8 ma8 (
    .sigma_in(epoly[8]),
    .product_out(m8),
    .load(load),
    .clk(CK),
    .pdata(m9),
    .pload(pload)
);

```

```

lpalpha9 ma9 (
    .sigma_in(epoly[9]),
    .product_out(m9),
    .load(load),
    .clk(CK),
    .pdata(8'h00),
    .pload(pload)
);

```

//instantiating the sum of product device for error correction

```

lp3error errorc (
    .m1(m1),
    .m2(m2),
    .m3(m3),
    .m4(m4),
    .m5(m5),
    .m6(m6),
    .m7(m7),
    .m8(m8),
    .m9(m9),
    .corrt(fec),
    .to1(to1),
    .to2(to2),
    .to3(to3),
    .to4(to4),
    .to5(to5),
    .to6(to6),
    .to7(to7)
);

```

```

);
endmodule

```

lp3error.v:

```

//The module for summing all product terms together
//for determining the status of a bit, whether it
//is received in error or not
module lp3error
(m1,m2,m3,m4,m5,m6,m7,m8,m9,corrt,to1,to2,to3,to4,to5,to6,to7);

    // I_O port definitions
    input [0:7] m1,m2,m3,m4,m5,m6,m7,m8,m9; //products from the
multipliers
    output corrt; //correction true
or not
    output [0:7] to1,to2,to3,to4,to5,to6,to7; //outputs for
polynomial order reduction

    // sign type definitions
    reg [0:7] to1,to2,to3,to4,to5,to6,to7,to8,to9;
    reg [0:7] ts1,ts2,ts3,ts4,ts5,ts6,ts7;
    reg corrt;
    reg [0:7] sb; //contains the sum of all
terms

    // variable used
    integer sloop; //for iterating
all the bits

    always @ (m1 or m2 or m3 or m4 or m5 or m6 or m7 or m8 or m9 or
to1 or to2 or to3 or to4 or to5 or to6 or to7 or sb or corrt)
        begin
            ts7[0]=m9[0]^m8[0];
            ts6[0]=ts7[0]^m7[0];
            ts5[0]=ts6[0]^m6[0];
            ts4[0]=ts5[0]^m5[0];

```

```

ts3[0]=ts4[0]^m4[0];
ts2[0]=ts3[0]^m3[0];
ts1[0]=ts2[0]^m2[0];
sb[0]=m1[0]^(~ts1[0]);
for (sloop=1; sloop<8; sloop=sloop+1)
    begin
        ts7[sloop]=m9[sloop]^m8[sloop];
        ts6[sloop]=ts7[sloop]^m7[sloop];
        ts5[sloop]=ts6[sloop]^m6[sloop];
        ts4[sloop]=ts5[sloop]^m5[sloop];
        ts3[sloop]=ts4[sloop]^m4[sloop];
        ts2[sloop]=ts3[sloop]^m3[sloop];
        ts1[sloop]=ts2[sloop]^m2[sloop];
        sb[sloop]=ts1[sloop]^m1[sloop];
    end
corrt = ~|sb;

if (corrt==0)
    begin
        to1=8'h00;
        to2=8'h00;
        to3=8'h00;
        to4=8'h00;
        to5=8'h00;
        to6=8'h00;
        to7=8'h00;
    end
else
    begin
        to1=ts1;
        to2=ts2;
        to3=ts3;
        to4=ts4;
        to5=ts5;
        to6=ts6;
        to7=ts7;
    end
end
end

```

```
endmodule
```

lpalpha.v:

```
//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this multiplier multiplies an 8-bit number "sigma" by alpha
module lpalpha (sigma_in, product_out, load, clk, pdata, pload);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal
    output [0:7] product_out;    //8-bit parallel product

    output
    input [0:7] pdata;        //8-bit parallel data input
    input      pload;        //parallel data load signal

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    wire [0:7] pdata;
    wire      pload;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

    // main process of multiplication
    always @ (posedge clk)
        if (load==1)//&&(pload==0))
            begin
                S <= (S>>1);
                S[0] <= sigma_in;
            end
        else if ((pload==1)&&(load==0))
            S <= pdata;
```

```

        else if ((pload==0)&&(load==0))
            begin
                S[5:7] <= S[4:6];
                S[4] <= S[3]^S[7];
                S[3] <= S[2]^S[7];
                S[2] <= S[1]^S[7];
                S[1] <= S[0];
                S[0] <= S[7];
            end
        else
            S <= S;
        endmodule

```

lalpha2.v:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha^2
module lalpha2 (sigma_in, product_out, load, clk, pdata, pload);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal
    output [0:7] product_out;    //8-bit parallel product
output
    input [0:7] pdata;      //8-bit parallel data input
    input      pload;       //parallel data load signal

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    wire [0:7] pdata;
    wire      pload;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

```

```

// main process of mutiplication
always @ (posedge clk)
    if (load==1)//&&(pload==0))
        begin
            S <= (S>>1);
            S[0] <= sigma_in;
        end
    else if ((pload==1)&&(load==0))
        S <= pdata;
    else if ((pload==0)&&(load==0))
        begin
            S[7] <= S[5];
            S[6] <= S[4];
            S[5] <= S[3]^S[7];
            S[4] <= S[2]^S[6]^S[7];
            S[3] <= S[1]^S[6]^S[7];
            S[2] <= S[0]^S[6];
            S[1] <= S[7];
            S[0] <= S[6];
        end
    else
        S <= S;
endmodule

```

lalpha3.v:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha^3
module lalpha3 (sigma_in, product_out, load, clk, pdata, pload);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal

```



```

        output [0:7]      product_out;      //8-bit parallel product
output
        input [0:7] pdata;                  //8-bit parallel data input
        input      pload;                  //parallel data load signal

// signal type definitions
wire      sigma,load,clk;
wire [0:7] product_out;
wire [0:7] pdata;
wire      pload;
reg [0:7]  S;

// connecting the product register to output
assign product_out=S;

// main process of mutiplication
always @ (posedge clk)
    if (load==1)//&&(pload==0))
        begin
            S <= (S>>1);
            S[0] <= sigma_in;
        end
    else if ((pload==1)&&(load==0))
        S <= pdata;
    else if ((pload==0)&&(load==0))
        begin
            S[7] <= S[4];
            S[6] <= S[3]^S[7];
            S[5] <= S[2]^S[6]^S[7];
            S[4] <= S[1]^S[5]^S[6]^S[7];
            S[3] <= S[0]^S[5]^S[6];
            S[2] <= S[5]^S[7];
            S[1] <= S[6];
            S[0] <= S[5];
        end
    else
        S <= S;
endmodule

```

lalpha4.v:

```
//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this multiplier multiplies an 8-bit number "sigma" by alpha^4
module lalpha4 (sigma_in, product_out, load, clk, pdata, pload);
    // I_O port definitions
    input      sigma_in; //bit-serial input for the sigma
    input      load;     //load enable signal for sigma
    input      clk;      //input clock signal
    output [0:7] product_out; //8-bit parallel product

output
    input [0:7] pdata; //8-bit parallel data input
    input      pload; //parallel data load signal

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    wire [0:7] pdata;
    wire      pload;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

    // main process of multiplication
    always @ (posedge clk)
        if (load==1)//&&(pload==0))
            begin
                S <= (S>>1);
                S[0] <= sigma_in;
            end
        else if ((pload==1)&&(load==0))
            S <= pdata;
        else if ((pload==0)&&(load==0))
```

```

begin
    S[7] <= S[3]^S[7];
    S[6] <= S[2]^S[6]^S[7];
    S[5] <= S[1]^S[5]^S[6]^S[7];
    S[4] <= S[0]^S[4]^S[5]^S[6];
    S[3] <= S[4]^S[5]^S[7];
    S[2] <= S[4]^S[6];
    S[1] <= S[5];
    S[0] <= S[4];
end
else
    S <= S;
endmodule

```

lpalpha5.v:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha^5
module lpalpha5 (sigma_in, product_out, load, clk, pdata, pload);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal
    output [0:7] product_out;    //8-bit parallel product
output
    input [0:7] pdata;      //8-bit parallel data input
    input      pload;       //parallel data load signal

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    wire [0:7] pdata;
    wire      pload;
    reg [0:7] S;

    // connecting the product register to output

```

```

assign product_out=S;

// main process of mutiplication
always @ (posedge clk)
    if (load==1)//&&(pload==0)
        begin
            S <= (S>>1);
            S[0] <= sigma_in;
        end
    else if ((pload==1)&&(load==0))
        S <= pdata;
    else if ((pload==0)&&(load==0))
        begin
            S[7] <= S[2]^S[6]^S[7];
            S[6] <= S[1]^S[5]^S[6]^S[7];
            S[5] <= S[0]^S[4]^S[5]^S[6];
            S[4] <= S[3]^S[4]^S[5];
            S[3] <= S[3]^S[4]^S[6]^S[7];
            S[2] <= S[3]^S[5]^S[7];
            S[1] <= S[4];
            S[0] <= S[3]^S[7];
        end
    else
        S <= S;
endmodule

```

lalpha6.v:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha^6
module lalpha6 (sigma_in, product_out, load, clk, pdata, pload);
    // I_O port definitions
    input      sigma_in;    //bit-serial input for the sigma
    input      load;        //load enable signal for sigma
    input      clk;         //input clock signal

```

```

        output [0:7]      product_out;      //8-bit parallel product
output
        input [0:7] pdata;                  //8-bit parallel data input
        input      pload;                   //parallel data load signal

// signal type definitions
wire      sigma,load,clk;
wire [0:7] product_out;
wire [0:7] pdata;
wire      pload;
reg [0:7]  S;

// connecting the product register to output
assign product_out=S;

// main process of mutiplication
always @ (posedge clk)
    if (load==1)//&&(pload==0))
        begin
            S <= (S>>1);
            S[0] <= sigma_in;
        end
    else if ((pload==1)&&(load==0))
        S <= pdata;
    else if ((pload==0)&&(load==0))
        begin
            S[7] <= S[1]^S[5]^S[6]^S[7];
            S[6] <= S[0]^S[4]^S[5]^S[6];
            S[5] <= S[3]^S[4]^S[5];
            S[4] <= S[2]^S[3]^S[4];
            S[3] <= S[2]^S[3]^S[5]^S[6];
            S[2] <= S[2]^S[4]^S[6]^S[7];
            S[1] <= S[3]^S[7];
            S[0] <= S[2]^S[6]^S[7];
        end
    else
        S <= S;
endmodule

```

lalpha7.v:

```
//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this multiplier multiplies an 8-bit number "sigma" by alpha ^7
module lalpha7 (sigma_in, product_out, load, clk, pdata, pload);
    // I_O port definitions
    input      sigma_in; //bit-serial input for the sigma
    input      load;     //load enable signal for sigma
    input      clk;      //input clock signal
    output [0:7] product_out; //8-bit parallel product
output
    input [0:7] pdata; //8-bit parallel data input
    input      pload; //parallel data load signal

    // signal type definitions
    wire      sigma,load,clk;
    wire [0:7] product_out;
    wire [0:7] pdata;
    wire      pload;
    reg [0:7] S;

    // connecting the product register to output
    assign product_out=S;

    // main process of multiplication
    always @ (posedge clk)
        if (load==1)//&&(pload==0))
            begin
                S <= (S>>1);
                S[0] <= sigma_in;
            end
        else if ((pload==1)&&(load==0))
            S <= pdata;
        else if ((pload==0)&&(load==0))
```

```

begin
    S[7] <= S[0]^S[4]^S[5]^S[6];
    S[6] <= S[3]^S[4]^S[5];
    S[5] <= S[2]^S[3]^S[4];
    S[4] <= S[1]^S[2]^S[3]^S[7];
    S[3] <= S[1]^S[2]^S[4]^S[5];
    S[2] <= S[1]^S[3]^S[5]^S[6];
    S[1] <= S[2]^S[6]^S[7];
    S[0] <= S[1]^S[5]^S[6]^S[7];
end
else
    S <= S;
endmodule

```

lalpha8.v:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this multiplier multiplies an 8-bit number "sigma" by alpha ^8
module lalpha8 (sigma_in, product_out, load, clk, pdata, pload);
    // I_O port definitions
    input    sigma_in;    //bit-serial input for the sigma
    input    load;        //load enable signal for sigma
    input    clk;         //input clock signal
    output [0:7]    product_out;    //8-bit parallel product
output
    input [0:7] pdata;    //8-bit parallel data input
    input    pload;      //parallel data load signal

    // signal type definitions
    wire    sigma,load,clk;
    wire [0:7]    product_out;
    wire [0:7]    pdata;
    wire    pload;
    reg [0:7]    S;

    // connecting the product register to output

```

```

assign product_out=S;

// main process of mutiplication
always @ (posedge clk)
    if (load==1)//&&(pload==0))
        begin
            S <= (S>>1);
            S[0] <= sigma_in;
        end
    else if ((pload==1)&&(load==0))
        S <= pdata;
    else if ((pload==0)&&(load==0))
        begin
            S[7] <= S[3]^S[4]^S[5];
            S[6] <= S[2]^S[3]^S[4];
            S[5] <= S[1]^S[2]^S[3]^S[7];
            S[4] <= S[0]^S[1]^S[2]^S[6];
            S[3] <= S[0]^S[1]^S[3]^S[4];
            S[2] <= S[0]^S[2]^S[4]^S[5]^S[7];
            S[1] <= S[1]^S[5]^S[6]^S[7];
            S[0] <= S[0]^S[4]^S[5]^S[6];
        end
    else
        S <= S;
endmodule

```

lalpha9.v:

```

//regular bit-serial Finite Field multiplier
//for use in BCH decoder in the Chien Search stage
//for GF(2^8), p(alpha)=1 + X^2 + X^3 + X^4 + X^8 =0
//this mutiplier mutiplies an 8-bit number "sigma" by alpha ^9
module lalpha9 (sigma_in, product_out, load, clk, pdata, pload);
    // I_O port definitions
    input      sigma_in; //bit-serial input for the sigma
    input      load;     //load enable signal for sigma
    input      clk;      //input clock signal

```



```

        output [0:7]      product_out;      //8-bit parallel product
output
        input [0:7] pdata;                  //8-bit parallel data input
        input      pload;                  //parallel data load signal

// signal type definitions
wire      sigma,load,clk;
wire [0:7] product_out;
wire [0:7] pdata;
wire      pload;
reg [0:7]  S;

// connecting the product register to output
assign product_out=S;

// main process of mutiplication
always @ (posedge clk)
    if (load==1)//&&(pload==0))
        begin
            S <= (S>>1);
            S[0] <= sigma_in;
        end
    else if ((pload==1)&&(load==0))
        S <= pdata;
    else if ((pload==0)&&(load==0))
        begin
            S[7] <= S[2]^S[3]^S[4];
            S[6] <= S[1]^S[2]^S[3]^S[7];
            S[5] <= S[0]^S[1]^S[2]^S[6];
            S[4] <= S[0]^S[1]^S[5];
            S[3] <= S[0]^S[2]^S[3]^S[7];
            S[2] <= S[1]^S[3]^S[4]^S[6]^S[7];
            S[1] <= S[0]^S[4]^S[5]^S[6];
            S[0] <= S[3]^S[4]^S[5];
        end
    else
        S <= S;
endmodule

```

lp3test.v

```
//Testbench for the regular Chien Search Engine
`timescale 1ns/10ps
module lp3test;

    // Test signals definitions
    reg        rst;          //system reset signal
    reg        sysclk;       //system clock signal
    reg [3:0]  errnum;       //register containing the number of
error bits per block
    reg [0:7]  sigma1,sigma2; //registers for the error
polynomial
    reg [0:7]  sigma3,sigma4;
    reg [0:7]  sigma5,sigma6;
    reg [0:7]  sigma7,sigma8;
    reg [0:7]  sigma9;
    wire       load;        //data enable for the sigma terms
    wire       m2c;         //streaming data to the Chien Search
    wire       c2t;         //corrected data stream from the Chien
Search
    wire       data_en;     //data enable for the streaming data
input
    wire [1:9] epoly;

    // Source Registers
    reg [0:186] orig_mess; //holding the error-free original data
    reg [0:254] source_mess; //holding the data to the Chien
Search
    reg [0:254] corrat_mess; //holding the error corrected
message from Chien Search

    // Other Registers used by the tester only
    reg [0:254] temp_source; //buffers for transient data to the
Chien Search
    reg [0:186] temp_orig; //and to the tester
```

```

        reg [0:7]  temp1,temp2;          //temp buffers used during test
array data transfer

        // Source Test Vector array specific variables
        integer      index;              //index to the current block
of test vectors
        integer      dloop;              //data fetch loop variables
        integer      bc;
        integer      simdone;           //indicate the completion of a
simulation
        integer      lastbit;           //indicate readiness of a new block
        integer      afreset;           //is this right after reset?
        parameter    slist=65*50;       //length of the test vectors
        reg [0:7]    sim_data [0:slist-1]; //test vector array

        // connect source_mess "epoly" to the Chien Search Engine
        assign m2c=source_mess[254];
        assign
epoly={sigma1[7],sigma2[7],sigma3[7],sigma4[7],sigma5[7],sigma6[7],sigma
a7[7],sigma8[7],sigma9[7]};

        //Initializing the Testbench
        //by filling the test vector array first
        //and generate a reset pulse in the meantime for initializing the
Chien Search Engine
        initial
            begin
                rst=1;
                $readmemb("simdata.dat",sim_data);
                #84  rst=0;
                $dumpfile("lp3chien.dump");
                $dumpvars(1,lpserh);
            end

        //reseting the array pointer and starting the system clock
        initial
            begin
                index=0;

```

```

        simdone=0;
        sysclk=0;
        lastbit=0;
        afreset=1;
        bc=0;
        forever
            #10    sysclk = ~sysclk;
        end

//Define the running tasks
//load the next test vector
always @ (posedge load)
    if (index*65>=slist)
        simdone=1;
    else
        begin
            //fetch the next original message for
verification
            for (dloop=0; dloop<23 ;dloop=dloop+1)

                {temp_orig[dloop*8],temp_orig[dloop*8+1],temp_orig[dloop*8+2],tem
p_orig[dloop*8+3],temp_orig[dloop*8+4],temp_orig[dloop*8+5],temp_orig[d
loop*8+6],temp_orig[dloop*8+7]}=sim_data[index*65+dloop];
                temp1=sim_data[index*65+23];
                temp_orig[184:186] = temp1[0:2];

            //fetch the number of errors for the next block
            errnum <= temp1[4:7];

            //fetch the next error polynomial
            sigma1=sim_data[index*65+24];
            sigma2=sim_data[index*65+25];
            sigma3=sim_data[index*65+26];
            sigma4=sim_data[index*65+27];
            sigma5=sim_data[index*65+28];
            sigma6=sim_data[index*65+29];
            sigma7=sim_data[index*65+30];
            sigma8=sim_data[index*65+31];

```

```

        sigma9=sim_data[index*65+32];

        //fetch the next corrupted message block
        for (dloop=0; dloop<31; dloop=dloop+1)

            {temp_source[dloop*8],temp_source[dloop*8+1],temp_source[dloop*8+
2],temp_source[dloop*8+3],temp_source[dloop*8+4],temp_source[dloop*8+5]
,temp_source[dloop*8+6],temp_source[dloop*8+7]}=sim_data[index*65+33+dloop];

            temp2=sim_data[index*65+64];
            temp_source[248:254]=temp2[0:6];

            //go to the next block
            index <= index+1;
            $display("fetch test vector %d",index);
        end

//storing output from the Chien Search Engine
always @ (posedge sysclk)
    if (data_en==1)
        begin
            bc <= bc+1;
            corrat_mess[0] <= c2t;
            corrat_mess[1:254] <=corrat_mess[0:253];
            if (load==1)
                lastbit<=2;
        end
    else
        begin
            corrat_mess <= corrat_mess;
            if (lastbit==2)
                begin
                    corrat_mess[0] <= c2t;
                    corrat_mess[1:254] <=
corrat_mess[0:253];

                    lastbit<=1;
                end
            else if (lastbit==1)

```

```

begin
    lastbit<=0;
    orig_mess<=temp_orig;
    source_mess<=temp_source;
    bc <= 0;
    if (corrat_mess[68:254] !=
orig_mess)
        begin
            $display("block bit
count=%d",bc);
            $display("error
positions");

            $display("%b",orig_mess^corrat_mess[68:254]);
            end
            if (simdone==1)
                begin
                    $dumpoff;
                    $display("vector
index=%d, list length=%d ",index,slist);
                    $display("simulation
completed!");
                    $finish;
                end
            end
        else
            if ((afreset==1)&&(load==1))
                begin
                    orig_mess<=temp_orig;
                    source_mess<=temp_source;
                    afreset<=0;
                end
            end
        end
end
end

```

```

//load the error polynomial at the beginning of every message bit
always @ (posedge sysclk)

```

```

if (load==1)
    begin
        sigma1 <= (sigma1>>1);
        sigma2 <= (sigma2>>1);
        sigma3 <= (sigma3>>1);
        sigma4 <= (sigma4>>1);
        sigma5 <= (sigma5>>1);
        sigma6 <= (sigma6>>1);
        sigma7 <= (sigma7>>1);
        sigma8 <= (sigma8>>1);
        sigma9 <= (sigma9>>1);
    end
else
    begin
        sigma1 <= sigma1;
        sigma2 <= sigma2;
        sigma3 <= sigma3;
        sigma4 <= sigma4;
        sigma5 <= sigma5;
        sigma6 <= sigma6;
        sigma7 <= sigma7;
        sigma8 <= sigma8;
        sigma9 <= sigma9;
    end

//streaming data to the Chien Search Engine
always @ (posedge sysclk)
    if (data_en==1)
        source_mess <= (source_mess>>1);

//instantiating the Chien Search Engine
lp3chien lpserh (
    .rst(rst),
    .load(load),
    .mout(c2t),
    .data_en(data_en),
    .epoly(epoly),
    .CK(sysclk),

```

```
        .mess_in(m2c)
    );
endmodule
```


A.3 MATLAB Source Codes

retestvec.m:

```
%generate test vectors for testing the hardware Chien Search
%for 88-541 Low Power CMOS Design
%we will first generate a (255,187) random code messages
%corrupt the message with 9 uniformly distributed random error bits
%then decode the message upto the generation of error polynomial

%define the blocks of test vectors
%each block is 65 bytes
block=200;

%create the test vector file
fid=fopen('simdata.dat','w');

%iterating the test vectors
for loop=1:1:block
    %initializing the error positions
    epos1=0;
    epos2=0;
    epos3=0;

    %generate the 187 bits source message
    u=rand(1,187);
    u=(u>0.5);
    m=encoder(u);

    %introduce nine random error bits
    while ((epos1==epos2) || (epos1==epos3) || (epos1==epos4) || ...
           (epos1==epos5) || (epos1==epos6) || (epos1==epos7) || ...
           (epos1==epos8) || (epos2==epos9) || (epos2==epos3) || ...
           (epos2==epos4) || (epos2==epos5) || (epos2==epos6) || ...
           (epos2==epos7) || (epos2==epos8) || (epos2==epos9) || ...
           (epos3==epos4) || (epos3==epos5) || (epos3==epos6) || ...
           (epos3==epos7) || (epos3==epos8) || (epos3==epos9) || ...
```

```

        (epos4==epos5) || (epos4==epos6) || (epos4==epos7) || ...
        (epos4==epos8) || (epos4==epos9) || (epos5==epos6) || ...
        (epos5==epos7) || (epos5==epos8) || (epos5==epos9) || ...
        (epos6==epos7) || (epos6==epos8) || (epos6==epos9) || ...
        (epos7==epos8) || (epos7==epos9) || (epos8==epos9))
epos1=round(rand(1)*186)+1;
epos2=round(rand(1)*186)+1;
epos3=round(rand(1)*186)+1;
epos4=round(rand(1)*186)+1;
epos5=round(rand(1)*186)+1;
epos6=round(rand(1)*186)+1;
epos7=round(rand(1)*186)+1;
epos8=round(rand(1)*186)+1;
epos9=round(rand(1)*186)+1;
end
m(epos1)=xor(m(epos1),1);
m(epos2)=xor(m(epos2),1);
m(epos3)=xor(m(epos3),1);
m(epos4)=xor(m(epos4),1);
m(epos5)=xor(m(epos5),1);
m(epos6)=xor(m(epos6),1);
m(epos7)=xor(m(epos7),1);
m(epos8)=xor(m(epos8),1);
m(epos9)=xor(m(epos9),1);

%generate the error polynomial
S=syndrome(m);
perror=berlekamp(S);

%write the original message and error number
for dloop=0:1:22
    fprintf(fid,'%d%d%d%d%d%d%d%d\n',u(dloop*8+1:dloop*8+8));
end
fprintf(fid,'%d%d%d%d%d%d%d%d\n',[u(23*8+1:23*8+3), [0 1 0 0 1]]);

%write the error polynomial
for dloop=1:1:9
    fprintf(fid,'%d%d%d%d%d%d%d%d\n',perror(dloop,:));

```

```

end

%write the corrupted message
for dloop=0:1:30
    fprintf(fid, '%d%d%d%d%d%d%d%d\n', m(dloop*8+1:dloop*8+8));
end
fprintf(fid, '%d%d%d%d%d%d%d%d\n', [m(31*8+1:31*8+7), 0]);
end
fclose('all');

```

syndrome.m:

```

%syndrome calculation for GF(2^8) (255,187) BCH code
function S=syndrome(m)

%define syndrome and parity matrix
S=zeros(18,8);
H=zeros(255,8,18);
gm=2.*ones(1,8);

%read the parity matrix
H=read_h();

%form syndrome matrix
for sloop=1:1:18
    S(sloop,:)=mod(m*H(:,:,sloop),gm);
end

```

berlekamp.m

```

%Berlekamp's Algorithm for finding the error polynomial
function perror=berlekamp(S)

%get information from syndrome
[row,m]=size(S);

```

```

t=row/2;

%define variables used by the algorithm
perror=zeros(t,m);           %keeping the final value of the polynomial
sigma=zeros(t,m,t);         %intermediate steps for the polynomial
mu=0;                        %iteration counter
pmu=0;                       %"going back" counter
d=zeros(t,m);               %mu discrepancy
l=zeros(t,1);               %polynomial order
pd=zeros(t,1);              %for order determination
gtemp=zeros(1,8);
ptemp=zeros(t,m);

%initialize entries for mu=1
perror(1,:)=S(1,:);
sigma(1,:,1)=S(1,:);
if (sum(S(1,:))~=0)
    l(1)=1;
    pd(1)=1;
else
    l(1)=0;
    pd(1)=2;
end
d(1,:)=xor(S(3,:),gf_multiply(S(2,:),sigma(1,:,1)));

%finding the error polynomial
for mu=1:1:8
    pmu=mu;
    sigma(:,:,mu+1)=sigma(:,:,mu);
    if (sum(d(mu,:))~=0)
        if pmu==1
            pmu=0;
            %what if d(0)==0, then go back further to pmu=-0.5
            %and set X^2(mu-pmu)=ptemp
            if (sum(S(1,:))==0)
                pmu=-0.5;
                l(mu+1)=3;
            else

```

```

        l(mu+1)=2;
    end
    ptemp(l(mu+1),:)= [1 0 0 0 0 0 0 0];
    %find dmμ*(1/dμ)
    if pmu== -0.5
        gtemp=d(1,:);
    else
        gtemp=gf_multiply(d(1,:),gf_invert(S(1,:)));
    end
    %find sigma for the mu+1==2 entry
    for sl=1:1:l(mu+1)
        sigma(sl,:,mu+1)=xor(sigma(sl,:,mu+1),...
            gf_multiply(gtemp,ptemp(sl,:)));
    end
    %update "2mu-1mu" entry
    pd(mu+1)=2*(mu+1)-l(mu+1);
    %find the mu discrepancy

d(mu+1,:)=xor(xor(S(2*mu+3,:),gf_multiply(sigma(mu,:,mu+1),...
S(2*mu+2,:))),gf_multiply(sigma(mu+1,:,mu+1),S(2*mu+1,:)));
    else
        %find the last pmu entry with dmμ~0
        pmu=pmu-1;
        tp_pmu=pmu;
        while (tp_pmu>1)
            while (sum(d(pmu,:))==0)
                pmu=pmu-1;
                tp_pmu=pmu;
            end
            if ((pd(tp_pmu-1)>pd(pmu)) && (sum(d(tp_pmu-1,:))~0))
                pmu=tp_pmu-1;
            end
            tp_pmu=tp_pmu-1;
        end
        if ((pmu==1) && (sum(d(pmu,:))==0))
            if (sum(S(1,:))~0)
                pmu=0;
            end
        end
    end

```

```

        else
            pmu=-0.5;
        end
    end

end

%calculate lmu , X^2(mu-pmu)=ptemp and "dmu*(1/dp)"
if pmu==0
    l(mu+1)=2*(mu-pmu);
    ptemp(l(mu+1),:)= [1 0 0 0 0 0 0 0];
    gtemp=gf_multiply(d(mu,:),gf_invert(S(1,:)));
elseif pmu==-0.5
    l(mu+1)=2*(mu-pmu);
    ptemp(l(mu+1),:)= [1 0 0 0 0 0 0 0];
    gtemp=d(mu,:);
else
    l(mu+1)=2*(mu-pmu)+l(pmu);
    ptemp(2*(mu-pmu),:)= [1 0 0 0 0 0 0 0];
    ptemp((2*(mu-pmu)+1):l(mu+1),:)=sigma(1:l(pmu),:,pmu);
    gtemp=gf_multiply(d(mu,:),gf_invert(d(pmu,:)));
end

end

%incase of error, display what it is
if (l(mu+1)>9)
    disp(['mu=' num2str(mu)]);
    disp(['pmu=' num2str(pmu)]);
    disp(['l(mu+1)=' num2str(l(mu+1))]);
    disp(['l(pmu)=' num2str(l(pmu))]);
    for jloop=1:1:(mu+1)
        disp(['l(' num2str(jloop) ')=' num2str(l(jloop))]);
    end
    for jloop=1:1:(mu+1)
        disp(['pd(' num2str(jloop) ')='
num2str(pd(jloop))]);
    end
end

end

%find sigma
for sl=1:1:l(mu+1)

```

```

        sigma(sl, :, mu+1)=xor(sigma(sl, :, mu+1), ...
            gf_multiply(gtemp, ptemp(sl, :)));
    end
    pd(mu+1)=2*(mu+1)-l(mu+1);
    if (mu>7) break; end
    gtemp=S(2*mu+3, :);
    for dl=1:1:l(mu+1)
        gtemp=xor(gtemp, gf_multiply(sigma(dl, :, mu+1), ...
            S((2*mu+3-dl), :)));
    end
    d(mu+1, :)=gtemp;
end
else
    l(mu+1)=l(mu);
    pd(mu+1)=2*(mu+1)-l(mu+1);
    if (mu>7) break; end
    gtemp=S(2*mu+3, :);
    for dl=1:1:l(mu+1)
        gtemp=xor(gtemp, gf_multiply(sigma(dl, :, mu+1), ...
            S((2*mu+3-dl), :)));
    end
    d(mu+1, :)=gtemp;
end
end
perror=sigma(:, :, t);

```

read_h.m:

```

%read the parity matrix
function H=read_h()

%define the parity matrix
H=zeros(255,8,18);

%form parity matrix from h*.txt files
for ssloop=1:1:18
    eval(['fid=fopen(''h' int2str(ssloop) '.txt','r');']);

```

```
for loop=1:1:255
    [temp,count]=fscanf(fid,'%1d',8);
    H(loop,:ssloop)=temp';
end
fclose(fid);
end
```


APPENDIX B: MATLAB CODES FOR DDCHB

This appendix contains MATLAB source codes for both the SPICE parameters extraction of the transistors and the generation of 3-D contours for the determination of the point of maximum power conversion efficiency. Following are the list of files and some brief description of their functions.

Parameters Extraction:

necaltrans.m -- extract normal region parameters for the ST 90nm transistors
 input: np90_*.mat where * represents source-to-bulk bias
 output: t90p.mat
cal leaks.m -- extract subthreshold parameters for the ST 90nm transistors
 input: t90p.mat, np90_*.mat, leak90_*.mat (leakage data from Cadence)
 output: t90sp.mat

3-D Contour Analysis:

recon.m -- plot the 3-D contour for one δ value at a time
 calls: ffa.m, fra.m
ffa.m -- find average current and power when $V_{in} \geq V_{out}$
 calls: pfrwd1.m
fra.m -- find average current and power when $V_{in} < V_{out}$
 calls: prevrs1.m
pfrwd1.m -- find instantaneous V_{dsp} and I_o when $V_{in} \geq V_{out}$
 input: t90p.mat
 calls: ftriode.m, fpsub_nt.m, fnpsub.m
prevrs1.m -- prepare data for calling subcat1.m
 input: t90p.mat
 calls: subcat1.m
ftriode.m -- find V/I in the triode region
fpsub_nt.m -- find V/I when the P-type transistor enters subthreshold region
fnpsub.m -- find V/I when both transistors enter subthreshold region
subcat1.m -- find instantaneous V_{dsp} and I_o when $V_{in} < V_{out}$
 input: t90p.mat, t90sp.mat

B.1 Transistor Parameters Extraction

ncaltrans.m:

```
%estimate beta values for 90nm transistors
%beta=0.5*uCox*W/L    (W&L are drawn width and length)
%
%Drawn dimensions:
%NFET: W=31.825um, L=0.1um,    fingers=95 (0.333um per finger)
%PFET: W=31.825um, L=0.1um,    fingers=95 (0.333um per finger)
%
%Threshold voltages VTLIN (measured at Id=12.73uA, vds=25mV)
%NFET:
%Vsb=0V,    Vtlin=0.2237V
%Vsb=0.2V,  Vtlin=0.2434V
%Vsb=0.3V,  Vtlin=0.2523V
%Vsb=0.33V, Vtlin=0.2549V
%Vsb=0.36V, Vtlin=0.2574V
%Vsb=0.6V,  Vtlin=0.2763V
%Vsb=0.66V, Vtlin=0.2806V
%Vsb=0.72V, Vtlin=0.2849V
%
%PFET:
%Vsb=0V,    Vtlin=0.2785V
%Vsb=0.2V,  Vtlin=0.2991V
%Vsb=0.3V,  Vtlin=0.3088V
%Vsb=0.33V, Vtlin=0.3117V
%Vsb=0.36V, Vtlin=0.3145V
%Vsb=0.6V,  Vtlin=0.3364V
%Vsb=0.66V, Vtlin=0.3417V
%Vsb=0.72V, Vtlin=0.3469V

%load data for transistors
%load 'np90_0.mat' %for 90nm transistor Vsb=0mv
%load 'np90_2.mat' %for 90nm transistor Vsb=200mV
%load 'np90_3.mat' %for 90nm transistor Vsb=300mV
load 'np90_3_3.mat' %for 90nm transistor Vsb=330mV
```

```

%load 'np90_3_6.mat' %for 90nm transistor Vsb=360mV
%load 'np90_6.mat' %for 90nm transistor Vsb=600mV
%load 'np90_6_6.mat' %for 90nm transistor Vsb=660mV
%load 'np90_7_2.mat' %for 90nm transistor Vsb=720mV

%initialized threshold voltages
%vth_n=0.2237; %for Vsb=0mV
%vth_p=0.2785;

%vth_n=0.2434; %for Vsb=200mV
%vth_p=0.2991;

%vth_n=0.2523; %for Vsb=300mV
%vth_p=0.3088;

vth_n=0.2549; %for Vsb=330mV
vth_p=0.3117;

%vth_n=0.2574; %for Vsb=360mV
%vth_p=0.3145;

%vth_n=0.2763; %for Vsb=600mV
%vth_p=0.3364;

%vth_n=0.2806; %for Vsb=660mV
%vth_p=0.3417;

%vth_n=0.2849; %for Vsb=720mV
%vth_p=0.3469;

%initialize current when Vgs=Vth
%dimensional parameters (in um)
Isthn=spline(vgs,nid,vth_n);
Isthp=spline(vgs,pid,vth_p);
Wn=31.825;
Ln=0.1;
Wp=31.825;
Lp=0.1;

```

```

%setup program variables
alpha=1/100;
alpha2=alpha^2;
beta_n=zeros(100,1);
beta_p=zeros(100,1);
nssd=zeros(100,1);
pssd=zeros(100,1);

%calculate all possible beta and vth values from all data
for cloop=101:-1:2
    if vgs(cloop-1)>vth_n
        beta_n(cloop-1)=((sqrt(nid(cloop))-sqrt(nid(cloop-
1))))^2)/alpha2;
    else
        beta_n(cloop-1)=beta_n(cloop);
    end
    if vgs(cloop-1)>vth_p
        beta_p(cloop-1)=((sqrt(pid(cloop))-sqrt(pid(cloop-
1))))^2)/alpha2;
    else
        beta_p(cloop-1)=beta_p(cloop);
    end
end

%extrapolate all data points based on each pair of beta & vth value
%then compare with the real curve and calculate the SSD
%Id data points with vgs<vth are excluded
for bvloop=1:1:100
    Id_n=beta_n(bvloop).*((vgs-vth_n).^2);
    Id_n=(vgs<vth_n).*Isthn+(vgs>=vth_n).*Id_n;
    nssd(bvloop)=sum((Id_n-max(nid,Isthn.*ones(101,1))).^2);

    Id_p=beta_p(bvloop).*((vgs-vth_p).^2)+Isthp;
    Id_p=(vgs<vth_p).*Isthp+(vgs>=vth_p).*Id_p;
    pssd(bvloop)=sum((Id_p-max(pid,Isthp.*ones(101,1))).^2);
end

```

```

%find the estimated beta and vth pair
[dummy, nestimate]=min(nssd);
[dummy, pestimate]=min(pssd);
uncox=2*beta_n(nestimate)*Ln/Wn/1.1942;
upcox=2*beta_p(pestimate)*Lp/Wp/1.1942;
%load 't90fp.mat'
display(uncox);
display(Isthn);
display(vth_n);

display(upcox);
display(Isthp);
display(vth_p);
save 't90p.mat' uncox Isthn vth_n upcox Isthp vth_p

%calculate the estimated Id based on the beta and vth pair
%set Id=0 for all points with vgs<vth
Id_n=(0.5*uncox*Wn/Ln).*((vgs-
vth_n).^2);%.*(1+lamdf_n.*vgs+zetaf_n.*vgs.^2);
Id_n=(vgs>=vth_n).*Id_n;
Id_p=(0.5*upcox*Wp/Lp).*((vgs-
vth_p).^2);%.*(1+lamdf_p.*vgs+zetaf_p.*vgs.^2);
Id_p=(vgs>=vth_p).*Id_p;

%plot the result
plot(vgs,nid,'-+k',vgs,Id_n,':or',vgs,pid,'-+b',vgs,Id_p,'- .og');

```

calleaks.m:

```

%estimate parameters for transistors
%load 'leak.mat'      %90nm W=0.12 L=0.1
%load 'leak90_0.mat'  %90nm W=31.825 L=0.1 Vsb=0V
%load 'leak90_2.mat'  %90nm W=31.825 L=0.1 Vsb=200mV
%load 'leak90_3.mat'  %90nm W=31.825 L=0.1 Vsb=300mV
load 'leak90_3_3.mat'  %90nm W=31.825 L=0.1 Vsb=330mV
%load 'leak90_3_6.mat' %90nm W=31.825 L=0.1 Vsb=360mV
%load 'leak90_6.mat'  %90nm W=31.825 L=0.1 Vsb=600mV
%load 'leak90_6_6.mat' %90nm W=31.825 L=0.1 Vsb=660mV

```

```
%load 'leak90_7_2.mat'      %90nm W=31.825 L=0.1 Vsb=720mV
```

```
%load forward region data
```

```
%load 'np90_0.mat'
```

```
%load 'np90_2.mat'
```

```
%load 'np90_3.mat'
```

```
load 'np90_3_3.mat'
```

```
%load 'np90_3_6.mat'
```

```
%load 'np90_6.mat'
```

```
%load 'np90_6_6.mat'
```

```
%load 'np90_7_2.mat'
```

```
%load active region transistor parameters
```

```
load 't90p.mat'
```

```
%Wn=0.12;
```

```
%Ln=0.1;
```

```
%Wp=0.12;
```

```
%Lp=0.1;
```

```
Wn=31.825;
```

```
Ln=0.1;
```

```
Wp=31.825;
```

```
Lp=0.1;
```

```
k=1.3806503e-23;
```

```
q=1.60217646e-19;
```

```
T=27+273.15;
```

```
vt=k*T/q;
```

```
k_n=zeros(100,1);
```

```
gamma_n=zeros(100,1);
```

```
k_p=zeros(100,1);
```

```
gamma_p=zeros(100,1);
```

```
nssd=zeros(100,1);
```

```
pssd=zeros(100,1);
```

```
enleak=zeros(101,1);
```

```
epleak=zeros(101,1);
```

```

%calculate all possible k and gamma values from all data
for cloop=2:1:101
    rho=(vds(cloop-1)-vds(cloop-1)*exp(-1*vds(cloop-1)/vt))/...
        (vds(cloop)-vds(cloop)*exp(-1*vds(cloop)/vt));
    k_n(cloop-1)=(nleak(cloop-1)-rho*nleak(cloop))/...
        (rho*exp(-1*vds(cloop)/vt)-rho-exp(-1*vds(cloop-1)/vt)+1);
    gamma_n(cloop-1)=(nleak(cloop)/k_n(cloop-1)+exp(-1*vds(cloop)/vt)-
1)...
        /(vds(cloop)-vds(cloop)*exp(-1*vds(cloop)/vt));

    k_p(cloop-1)=(pleak(cloop-1)-rho*pleak(cloop))/...
        (rho*exp(-1*vds(cloop)/vt)-rho-exp(-1*vds(cloop-1)/vt)+1);
    gamma_p(cloop-1)=(pleak(cloop)/k_p(cloop-1)+exp(-1*vds(cloop)/vt)-
1)...
        /(vds(cloop)-vds(cloop)*exp(-1*vds(cloop)/vt));
end

%extrapolate all data points based on each pair of k & gamma value
%then compare with the real curve and calculate the SSD
for bvloop=1:1:20
    nls=0;
    pls=0;
    for eloop=1:1:101
        ileak_n=k_n(bvloop)*(1-exp(-
1*vds(eloop)/vt))*(1+gamma_n(bvloop)...
            *vds(eloop));
        nls=nls+(ileak_n-nleak(eloop))^2;

        ileak_p=k_p(bvloop)*(1-exp(-
1*vds(eloop)/vt))*(1+gamma_p(bvloop)...
            *vds(eloop));
        pls=pls+(ileak_p-pleak(eloop))^2;
    end
    nssd(bvloop)=nls;
    pssd(bvloop)=pls;
end

```

```

%find the estimated beta and vth pair
[dummy, nestimate]=min(nssd);
[dummy, pestimate]=min(pssd);
ison=k_n(nestimate)*Ln/Wn;
lamdn=gamma_n(nestimate);
isop=k_p(pestimate)*Lp/Wp;
lamdp=gamma_p(pestimate);
display(ison);
display(lamdn);
display(isop);
display(lamdp);

enl=k_n(nestimate)*(1-exp(-1*0.5/vt))*(1+gamma_n(nestimate)*0.5);
epl=k_p(pestimate)*(1-exp(-1*0.5/vt))*(1+gamma_p(pestimate)*0.5);

zeta_n=(nleak(51)-enl)/(k_n(nestimate)*(1-exp(-1*vds(51)/vt))...
*vds(51)^2);
zeta_p=(pleak(51)-epl)/(k_p(pestimate)*(1-exp(-1*vds(51)/vt))...
*vds(51)^2);

hnl=k_n(nestimate)*(1-exp(-1/vt))*(1+gamma_n(nestimate)+zeta_n);
hpl=k_p(pestimate)*(1-exp(-1/vt))*(1+gamma_p(pestimate)+zeta_p);

eta_n=(nleak(101)-hnl)/(k_n(nestimate)*(1-exp(-1/vt)));
eta_p=(pleak(101)-hpl)/(k_p(pestimate)*(1-exp(-1/vt)));

%calculate the estimated Id based on the beta and vth pair
for etloop=1:1:101
    enleak(etloop)=k_n(nestimate)*(1-exp(-1*vds(etloop)/vt))*...
        (1+gamma_n(nestimate)*vds(etloop)+zeta_n*vds(etloop)^2+...
        eta_n*vds(etloop)^3);
    epleak(etloop)=k_p(pestimate)*(1-exp(-1*vds(etloop)/vt))*...
        (1+gamma_p(pestimate)*vds(etloop)+zeta_p*vds(etloop)^2+...
        eta_p*vds(etloop)^3);
end

display(zeta_n);
display(zeta_p);

```



```

display(eta_n);
display(eta_p);

%find n parameters for the leakage
Isthn=spline(vgs,nid,vth_n);
Isthp=spline(vgs,pid,vth_p);
ni=0;
nf=3;

ilf=0;
while(abs(Isthn-ilf)>eps)
    n_nt=(ni+nf)/2;
    ilf=ison*Wn/Ln*exp(vth_n/(n_nt*vt))*(1-exp(-1*vth_n/vt))*...
        (1+lamd_n*vth_n+zeta_n*vth_n^2+eta_n*vth_n^3);
    if ilf>Isthn
        ni=n_nt;
    elseif ilf<Isthn
        nf=n_nt;
    else
        break;
    end
end

ni=0;
nf=3;

ilf=0;
while(abs(Isthp-ilf)>eps)
    n_pt=(ni+nf)/2;
    ilf=isop*Wp/Lp*exp(vth_p/(n_pt*vt))*(1-exp(-1*vth_p/vt))*...
        (1+lamd_p*vth_p+zeta_p*vth_p^2+eta_p*vth_p^3);
    if ilf>Isthp
        ni=n_pt;
    elseif ilf<Isthp
        nf=n_pt;
    else
        break;
    end
end

```

```
end

display(n_nt);
display(n_pt);

save 't90sp.mat' ison lamdn zeta_n eta_n n_nt isop lamdp zeta_p eta_p
n_pt

%plot the result
plot(vds,nleak,'-+k',vds,enleak,':or');
figure;
plot(vds,pleak,'-+b',vds,epleak,'-.og');
```

B.2 3-D Contour Analysis

recon.m:

```
%find the rectifier contour for 90nm CMOS
%for  $g^2=0.4$  to  $g^2=8$ 

%define contour variables
voir=0.82;           %output to input voltage ratio
glist=0.1:0.1:5;     %g2 ratio
vlist=0.15:0.01:0.5; %output voltage array
vi=vlist./voir;      %max input voltage array
gl=length(glist);
vl=length(vlist);
inpower=zeros(gl,vl);
outpower=zeros(gl,vl);
current=zeros(gl,vl);
[X,Y]=meshgrid(vlist,glist);
count=0;
tcount=length(glist)*length(vlist);

tic
%find the efficiency contour
for gloop=1:1:gl
    for vloop=1:1:vl
        [ileak,rp]=fra(vi(vloop),vlist(vloop),glist(gloop));
        [ipa,fp]=ffa(vi(vloop),vlist(vloop),glist(gloop));

        inpower(gloop,vloop)=fp+rp;
        current(gloop,vloop)=ipa+ileak;
        outpower(gloop,vloop)=current(gloop,vloop)*vlist(vloop);

        count=count+1;
        if ((floor(count/10)*10)-count)==0
            clc;
            disp(['process ' int2str(count) ' of ' int2str(tcount)...
                ' datapoints']);
        end
    end
end
```

```

                disp('');
            end
        end
    end
end
if ((floor(count/10)*10)-count)~=0
    clc;
    disp(['process ' int2str(count) ' of ' int2str(tcount)...
        ' datapoints']);
    disp('');
end
toc
outpower=max(outpower,zeros(gl,vl));
eff=outpower./inpower;

%plot the contour
surf(X,Y,eff);
figure(2);
surf(X,Y,current);
ceff=max(max(current));
peff=max(max(eff));
display(ceff);
display(peff);
figure(3);
plot(glist,eff(:,1),'-r',glist,eff(:,7),'-b',glist,eff(:,21),'-g');

```

ffa.m:

```

%find forward average current and power
function [avc,avp]=ffa(vmax,vout,g2)

%define resolution
step=100;

%determine input voltage array
thetai=asin(vout/vmax);
thetaf=pi-thetai;
ar=(pi-2*thetai)/(2*pi);

```

```

vin=vmax.*sin(thetai:(thetaf-thetai)/step:thetaf);

%determine the P-transistor forward current and Vds
[vdsp,ip]=pfrwd1(vin,vout,g2);

%find average current
avc=sum((ip(1:step)+ip(2:step+1))./2)/step*ar;

%find average input power
power=vin.*ip;
avp=sum((power(1:step)+power(2:step+1))./2)/step*ar;

```

fra.m:

```

%find reverse average current and power
function [avc,avp]=fra(vmax,vout,g2)

%define resolution
step=500;

%determine input voltage array
thetai=pi-asin(vout/vmax);
thetaf=3*pi-thetai;
ar=(pi+2*(pi-1*thetai))/(2*pi);
vin=vmax.*sin(thetai:(thetaf-thetai)/step:thetaf);

%determine the P-transistor forward current and Vds
[vdsp,ip]=prevrsl(vin,vout,g2);

%find average current
avc=sum((ip(1:step)+ip(2:step+1))./2)/step*ar;

%find average input power
power=vin.*ip;
avp=sum((power(1:step)+power(2:step+1))./2)/step*ar;

```

pfrwd1.m:

```

%solve the quadratic forward current equation for 90nm CMOS
%g2=beta_n/beta_p, 90nmfor=current per [upCoxWp/Lp(1+g2)]
function [vdsp,ip]=pfrwd1(vin,vout,g2)

%load 90nm transistor parameters
load 't90p.mat'
tmp=27;

%create return array
fs=length(vin);
vdsp=zeros(1,fs);
ip=zeros(1,fs);

%define forward conduction regions
%triode region: vin >= vout && vin >= vthp
m1=((vin>=vout)&(vin>=vth_p));

%region 2: P-FET enters subthreshold region with N-FET still in triode
%vin >= vout && vthp > vin >= vthn
m2=((vin<vth_p)&(vin>=vth_n)&(vin<vth_p));

%region 3: both FETs enters subthreshold region
%vthn > vin >= vout
m3=((vin>=vout)&(vin<vth_n));

%process the array
for aloop=1:1:fs
    if m1(aloop)==1
        Id(aloop)=ftriode(vin(aloop),vout,g2,tmp);
    elseif m2(aloop)==1
        Id(aloop)=fsub_nt(vin(aloop),vout,g2,tmp);
    elseif m3(aloop)==1
        Id(aloop)=fnsub(vin(aloop),vout,g2,tmp);
    end
end
end

```

prevrs1.m:

```
%solve the reverse current equation for 90nm CMOS
%g2=beta_n/beta_p, 90nmfor=current per [upCoxWp/Lp(1+g2)]
function [vdsp,ip]=prevrs1(vin,vout,g2)

%define 90nm transistor parameters
load 't90p.mat'
tmp=27;

%create return array
rs=length(vin);
vdsp=zeros(1,rs);
ip=zeros(1,rs);

%define reverse conduction regions
%combine all regions except for vin=vout (process by the forward eq.)
mask=(vin<vout);

%process the array
for aloop=1:1:rs
    if mask(aloop)==1
        [vdsp(aloop),ip(aloop)]=subcat1(vin(aloop),vout,g2,tmp);
    end
end

vdsp=-1.*vdsp;
ip=-1.*ip;
```

ftriode.m:

```
%The function that calculates the forward triode region current
%of the P-FET in 90nm
%for individual input voltage
function [vds,Id]=ftriode(vin,vout,g2,temp)

%thermal temperature
```

```

vt=(1.3806504e-23)*(temp+273.15)/1.602176487e-19;

%subthreshold parameters
load 't90p.mat'
load 't90sp.mat'

vdsi=0;
vdsf=vin-vout;
Idn=0;
Idp=1;

while(abs(Idn-Idp)>eps)
    vds=(vdsi+vdsf)/2;
    vdsn=vin-vout-vds;
    if vds<=vin-vth_p
        Idp=(isop/(upcox*(1+g2)))*exp(vth_p/(n_pt*vt))*...
            (1-exp((-
1/vt)*vds))* (1+lamp* vds+zeta_p*vds^2+eta_p*vds^3)+...
            1/(1+g2)*((vin-vth_p)*vds-0.5*vds^2);
    else
        Idp=(isop/(upcox*(1+g2)))*exp(vth_p/(n_pt*vt))*...
            (1-exp((-
1/vt)*vds))* (1+lamp* vds+zeta_p*vds^2+eta_p*vds^3)+...
            0.5/(1+g2)*((vin-vth_p)^2);
    end
    if vdsn<=vin-vth_n
        Idn=(ison*g2/(uncox*(1+g2)))*exp(vth_n/(n_nt*vt))*...
            (1-exp((-1/vt)*vdsn))* (1+lamd* vdsn+zeta_n*vdsn^2+...
            eta_n*vdsn^3)+g2/(1+g2)*((vin-vth_n)*vdsn-0.5*vdsn^2);
    else
        Idn=(ison*g2/(uncox*(1+g2)))*exp(vth_n/(n_nt*vt))*...
            (1-exp((-1/vt)*vdsn))* (1+lamd* vdsn+zeta_n*vdsn^2+...
            eta_n*vdsn^3)+0.5*g2/(1+g2)*((vin-vth_n)^2);
    end

    if Idp<Idn
        vdsi=vds;
    elseif Idp>Idn

```



```

        vdsf=vds;
    else
        break;
    end

    if (vdsi==0) && (vdsf==0)
        break;
    end

    if abs(vdsi-vdsf)<eps
        Idp=(Idp+Idn)/2;
        break;
    end
end
end
Id=Idp;

```

fpsub_nt.m:

```

%The function that calculates the forward triode region current
%of the P-FET in 90nm
%for individual input voltage
function [vds,Id]=fpsub_nt(vin,vout,g2,temp)

%thermal temperature
vt=(1.3806504e-23)*(temp+273.15)/1.602176487e-19;

%subthreshold parameters
load 't90p.mat'
load 't90sp.mat'

vdsi=0;
vdsf=vin-vout;
Idn=0;
Idp=1;

while(abs(Idn-Idp)>eps)
    vds=(vdsi+vdsf)/2;

```

```

vdsn=vin-vout-vds;
Idp=(isop/(upcox*(1+g2)))*exp(vin/(n_pt*vt))*...
    (1-exp((-1/vt)*vds))*(1+lamp*vds+zeta_p*vds^2+eta_p*vds^3);

if vdsn<vin-vth_n
    Idn=(ison*g2/(uncox*(1+g2)))*exp(vth_n/(n_nt*vt))*...
        (1-exp((-1/vt)*vdsn))*(1+lamd*n*vdsn+zeta_n*vdsn^2+...
        eta_n*vdsn^3)+g2/(1+g2)*((vin-vth_n)*vdsn-0.5*vdsn^2);
else
    Idn=(ison*g2/(uncox*(1+g2)))*exp(vth_n/(n_nt*vt))*...
        (1-exp((-1/vt)*vdsn))*(1+lamd*n*vdsn+zeta_n*vdsn^2+...
        eta_n*vdsn^3)+0.5*g2/(1+g2)*((vin-vth_n)^2);
end

if Idp<Idn
    vdsi=vds;
elseif Idp>Idn
    vdsf=vds;
else
    break;
end

if abs(vdsf-vdsi)<eps
    Idp=(Idp+Idn)/2;
    break;
end
end
Id=Idp;

```

fnpsub.m:

```

%The function that calculates the forward triode region current
%of the P-FET in 90nm
%for individual input voltage
function [vds,Id]=fnpsub(vin,vout,g2,temp)

%thermal temperature

```

```

vt=(1.3806504e-23)*(temp+273.15)/1.602176487e-19;

%subthreshold parameters
load 't90p.mat'
load 't90sp.mat'

vdsi=0;
vdsf=vin-vout;
Idn=0;
Idp=1;

while(abs(Idn-Idp)>eps)
    vds=(vdsi+vdsf)/2;
    vdsn=vin-vout-vds;
    Idp=(isop/(upcox*(1+g2)))*exp(vin/(n_pt*vt))*...
        (1-exp((-1/vt)*vds))*(1+lamp*vds+zeta_p*vds^2+eta_p*vds^3);

    Idn=(ison*g2/(uncox*(1+g2)))*exp(vin/(n_nt*vt))*...
        (1-exp((-
1/vt)*vdsn))*(1+lamd*n*vdsn+zeta_n*vdsn^2+eta_n*vdsn^3);

    if Idp<Idn
        vdsi=vds;
    elseif Idp>Idn
        vdsf=vds;
    else
        break;
    end
end
Id=Idp;

```

subcat1.m:

```

%The function that calculates the reverse conduction region current
%of the P-FET in 90nm
%for individual input voltage
function [vds,Id]=subcat1(vin,vout,g2,temp)

```

```

%thermal temperature
vt=(1.3806504e-23)*(temp+273.15)/1.602176487e-19;

%subthreshold parameters
load 't90p.mat'
load 't90sp.mat'

vdsi=0;
vdsf=vout-vin;
Idn=0;
Idp=1;

while(abs(Idn-Idp)>eps)
    vds=(vdsi+vdsf)/2;
    vdsn=vout-vin-vds;
    vgsd=vin+vds;
    vgsn=vout-vds;
    if vgsd>=vth_p
        if vds<=vgsd-vth_p
            Idp=(isop/(upcox*(1+g2)))*exp(vth_p/(n_pt*vt))*...
                (1-exp((-1/vt)*vds))*(1+lamp*vds+zeta_p*vds^2+...
                eta_p*vds^3)+1/(1+g2)*((vgsd-vth_p)*vds-0.5*vds^2);
        else
            Idp=(isop/(upcox*(1+g2)))*exp(vth_p/(n_pt*vt))*...
                (1-exp((-1/vt)*vds))*(1+lamp*vds+zeta_p*vds^2+...
                eta_p*vds^3)+0.5/(1+g2)*((vgsd-vth_p)^2);
        end
    elseif vgsd>0
        Idp=(isop/(upcox*(1+g2)))*exp(vgsd/(n_pt*vt))*...
            (1-exp((-
1/vt)*vds))*(1+lamp*vds+zeta_p*vds^2+eta_p*vds^3);
    else
        Idp=(isop/(upcox*(1+g2)))*(1-exp((-1/vt)*vds))*...
            (1+lamp*vds+zeta_p*vds^2+eta_p*vds^3);
    end

    if vgsn>=vth_n
        if vdsn<=vgsn-vth_n

```

```

        Idn=(ison*g2/(uncox*(1+g2)))*exp(vth_n/(n_nt*vt))*...
            (1-exp((-1/vt)*vdsn))*(1+lamd_n*vdsn+zeta_n*vdsn^2+...
            eta_n*vdsn^3)+g2/(1+g2)*((vgsn-vth_n)*vdsn-0.5*vdsn^2);
    else
        Idn=(ison*g2/(uncox*(1+g2)))*exp(vth_n/(n_nt*vt))*...
            (1-exp((-1/vt)*vdsn))*(1+lamd_n*vdsn+zeta_n*vdsn^2+...
            eta_n*vdsn^3)+0.5*g2/(1+g2)*((vgsn-vth_n)^2);
    end
elseif vgsn>0
    Idn=(ison*g2/(uncox*(1+g2)))*exp(vgsn/(n_nt*vt))*...
        (1-exp((-1/vt)*vdsn))*(1+lamd_n*vdsn+zeta_n*vdsn^2+...
        eta_n*vdsn^3);
else
    Idn=(ison*g2/(uncox*(1+g2)))*(1-exp((-1/vt)*vdsn))*...
        (1+lamd_n*vdsn+zeta_n*vdsn^2+eta_n*vdsn^3);
end

if Idp<Idn
    vdsi=vds;
elseif Idp>Idn
    vdsf=vds;
else
    break;
end
if abs(vdsi-vdsf)<eps
    Idp=(Idn+Idp)/2;
    break;
end
end
Id=Idp;

```

APPENDIX C: PERMISSION FROM CO-AUTHORS AND IEEE/ACM

In this section a letter that permits the author of this thesis to use papers that were co-written by Dr. C. Chen and Dr. Q.M. Jonathan Wu is attached.

Also attached are email correspondences that permit the author to reuse IEEE/ACM-Copyrighted material.

Permission to Use Previously Submitted/Accepted Papers

I Dr. Chunhong Chen and I Dr. Q.M. Jonathan Wu give permission to Mr. Shu-Yi Wong to include the following papers into his master's thesis.

- 1) Paper published in poster session of ISLPED'09, May 2009, San Francisco, California, USA, entitled:

S.Y. Wong, C. Chen and Q.M.J. Wu, "Power-Management-Based Chien Search for Low Power BCH Decoder," in Proc. 14th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), Aug. 19-21 2009, pp. 299-302

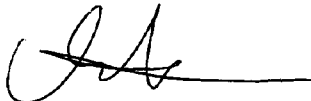
- 2) Paper accepted as a transaction brief for publication to IEEE Trans. on Very Large Scale Integration Systems, entitled:

S.Y. Wong, C. Chen, Q.M.J. Wu, "Low Power Chien Search for BCH Decoder Using RT-Level Power Management", IEEE Transactions on Very Large Scale Integration Systems (TVLSI)

- 3) Paper submitted to the 47th Design Automation Conference, entitled:

S.Y. Wong, C. Chen and Q.M.J. Wu, "Power Optimization of Multi-Stage Differential-Drive CMOS Rectifier for Passive RFID Tags"

Sincerely,



Dr. Chunhong Chen



Dr. Q.M. Jonathan Wu

Comments/Response to Case ID: 006A37E5

ReplyTo: Copyrights@ieee.org

From: Jacqueline Hansson Date: 11/11/2009

Subject: Re: ***[Possible Send To: "Wong Shu-Yi"
UCE]*** Permission <wong11j@uwindsor.ca>
to reuse paper for
Thesis

cc:

Dear Shu-Yi Wong :

We are happy to grant you this permission to reprint this IEEE copyrighted paper in your thesis and, if you wish, have it placed on your university's website. We have only two requirements that must be satisfied before we can consider this permission final:

(1) The following copyright/credit notice must appear prominently either on the first page of the reprinted material or prominently in the references of the reprinted paper, with the appropriate details filled in:

© [year] IEEE. Reprinted, with permission, from [IEEE publication title, paper title, and author names].

(2) Additionally, if your thesis is to appear on the university's website, the following message should be displayed either at the beginning of the credits or in an appropriate and prominent place on the website:

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Windsor's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By

choosing to view this material, you agree to all provisions of the copyright laws protecting it.

If applicable, University Microfilms, Inc. or ProQuest may supply single copies of the dissertation.

Sincerely,

Jacqueline Hansson

© ©

IEEE Intellectual Property Rights Office
445 Hoes Lane
Piscataway, NJ 08855-1331 USA
+1 732 562 3966 (phone)
+1 732 562 1746 (fax)

IEEE-- Fostering technological innovation
and excellence for the benefit of humanity.

© ©

Dear Sir/Madam,

I am requesting a written permission for the following paper to be included into my Master's Thesis. The paper is written by me under the supervision of my professor, Dr. Chunhong Chen and was accepted as a transaction brief for the IEEE Transactions on Very Large Scale Integration Systems in 2009, (TVLSI-00159-2009).

The information of the paper is described below:

- 1) Author's name: Shu-Yi Wong, Chunhong Chen, and Q.M. Jonathan Wu.
- 2) Paper title: Low Power Chien Search for BCH Decoder Using RT-Level Power Management
- 3) IEEE publication title: S. Y. Wong, C. Chen, Q. M. J. Wu, "Low Power Chien Search for BCH Decoder Using RT-Level Power Management," IEEE Transactions on Very Large Integration Systems (TVLSI), Transaction Brief

I would like to reuse all portions of the paper into my thesis and I would appreciate your response as soon as possible but, in any case, please reply no later than the end of November, 2009.

Thank You,

Sincerely,

Shu-Yi Wong
Master's Candidate

University of Windsor
401 Sunset Avenue, Windsor, Ontario
Canada. N9B 3P4
Tel: (519) 253-3000

Email: wong11j@uwindsor.ca

**ASSOCIATION FOR COMPUTING MACHINERY, INC. LICENSE
TERMS AND CONDITIONS**

Nov 14, 2009

This is a License Agreement between Shu-Yi Wong ("You") and Association for Computing Machinery, Inc. ("Association for Computing Machinery, Inc.") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Association for Computing Machinery, Inc., and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	2306280420142
License date	Nov 11, 2009
Licensed content publisher	Association for Computing Machinery, Inc.
Licensed content publication	Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design
Licensed content title	Power-management-based Chien search for low power BCH decoder
Licensed content author	Shu-Yi Wong, et al
Licensed content date	Aug 19, 2009
Type of Use	I don't see my intended use
Requestor type	Author of this ACM article

Order reference number

Special Requirements Request for reprinting in Master's Thesis. ACM publication title: S. Y. Wong, C. Chen, Q. M. J. Wu, "Power-Management- Based Chien Search for Low Power BCH Decoder," in Proceedings of ISLPED'09, Poster Session, pp.299-302.

Total 0.00 USD

Terms and Conditions

Rightslink Terms and Conditions for ACM Material

1. The publisher for this copyrighted material is Association for Computing Machinery, Inc. (ACM). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at).
2. ACM reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.
3. ACM hereby grants to licensee a non-exclusive license to use or republish this material in secondary works (especially for commercial distribution) with the stipulation that consent of the lead author has been obtained independently. Licenses are for one-time use in a single edition of the work, only with a maximum distribution equal to the number that you identified in the licensing process. Any additional form of republication must be specified according to the terms included at the time of licensing.
4. Any form of republication or redistribution must be used within 180 days from the date stated on the license and any electronic posting is limited to a period of six months unless an extended term is selected during the licensing process. Separate subsidiary and

subsequent republication licenses must be purchased to redistribute copyrighted material on an extranet. These licenses may be exercised anywhere in the world.

5. Licensee may not alter or modify the material in any manner (except that you may use, within the scope of the license granted, one or more excerpts from the copyrighted material, provided that the process of excerpting does not alter the meaning of the material or in any way reflect negatively on the publisher or any writer of the material).

6. Licensee must include the following copyright and permission notice in connection with any reproduction of the licensed material: "[Citation] © YEAR Association for Computing Machinery, Inc. Reprinted by permission." Include the article DOI as a link to the definitive version in the ACM Digital Library. Example: Charles, L. "How to Improve Digital Rights Management," Communications of the ACM, Vol. 51:12, © 2008 ACM, Inc. <http://doi.acm.org/10.1145/nnnnnn.nnnnnn> (where nnnnnn.nnnnnn is replaced by the actual number).

7. Translation of the material in any language requires an explicit license identified during the licensing process. Due to the error-prone nature of language translations, Licensee must include the following copyright and permission notice and disclaimer in connection with any reproduction of the licensed material in translation: "This translation is a derivative of ACM-copyrighted material. ACM did not prepare this translation and does not guarantee that it is an accurate copy of the originally published work. The original intellectual property contained in this work remains the property of ACM."

8. You may exercise the rights licensed immediately upon issuance of the license at the end of the licensing transaction, provided that you have disclosed complete and accurate details of your proposed use. No license is finally effective unless and until full payment is received from you (either by CCC or ACM) as provided in CCC's Billing and Payment terms and conditions.

9. If full payment is not received within 90 days from the grant of license transaction, then any license preliminarily granted shall be deemed automatically revoked and shall

be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted.

10. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.

11. ACM makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its Billing and Payment terms and conditions for this licensing transaction.

12. You hereby indemnify and agree to hold harmless ACM and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

13. This license is personal to the requestor and may not be sublicensed, assigned, or transferred by you to any other person without publisher's written permission.

14. This license may not be amended except in a writing signed by both parties (or, in the case of ACM, by CCC on its behalf).

15. ACM hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and ACM (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

16. This license transaction shall be governed by and construed in accordance with the laws of New York State. You hereby agree to submit to the jurisdiction of the federal and state courts located in New York for purposes of resolving any disputes that may arise in connection with this licensing transaction.

17. There are additional terms and conditions, established by Copyright Clearance Center, Inc. ("CCC") as the administrator of this licensing service that relate to billing and payment for licenses provided through this service. Those terms and conditions apply to each transaction as if they were restated here. As a user of this service, you agreed to those terms and conditions at the time that you established your account, and you may see them again at any time at <http://myaccount.copyright.com>

Special Terms: Gratis permission is granted as an author-retained right. This includes permission to post an author-prepared version of this article on the Dept. of Electrical and Computer Engineering web site and the University library reserves. Other document delivery requires separate license from the Copyright Clearance Center. Additional republication permission requires a separate licence and fee(s).

Gratis licenses (referencing \$0 in the Total field) are free. Please retain this printable license for your reference. No payment is required.

From: "Wong Shu-Yi" <wong11j@uwindsor.ca>
Subject: Request for permission to reuse paper in Thesis
Date: Wed, 11 Nov 2009 11:40:45 -0500
To: Permissions@acm.org
Cc: cotton@acm.org

Dear Sir/Madam,

I am requesting a written permission for the following paper to be included into my Master's Thesis. The paper is written by me under the supervision of my professor, Dr. Chunhong Chen. It was presented during the 14th ACM/IEEE International Symposium on Low Power Electronics and Design, held in San Francisco, California, USA, Aug 19-21, 2009 and it was subsequently published in the conference proceedings.

The information of the paper is described below:

1) Author's name: Shu-Yi Wong, Chunhong Chen, and Q.M. Jonathan Wu.

2) Paper title: Power-Management-Based Chien Search for Low Power BCH Decoder

3) ACM publication title: S. Y. Wong, C. Chen, Q. M. J. Wu, "Power-Management-Based Chien Search for Low Power BCH Decoder," in Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED'09), Poster Session, Aug 19-21, 2009, San Francisco, California, USA, pp.299-302.

I would like to reuse all portions of the paper into my thesis and I would appreciate your response as soon as possible but, in any case, please reply no later than the end of November, 2009.

Thank You,

Sincerely,

Shu-Yi Wong

Master's Candidate

University of Windsor
401 Sunset Avenue, Windsor, Ontario
Canada. N9B 3P4
Tel: (519) 253-3000
Email: wong11j@uwindsor.ca

VITA AUCTORIS

Shu-Yi (Jack) Wong was born in 1970 in Chiayi, Taiwan and immigrated to Canada in 1987. He graduated from Northview Heights Secondary School in 1989. From there he went on to the University of Waterloo where he obtained a B.A.Sc. in Electrical Engineering in 1994. Afterwards, Jack had worked in the industry and with his last involvement being a Senior Hardware Engineer with Tyco Safety Products Canada Ltd. in 2007. He is currently a candidate for the Master of Applied Science degree in Electrical Engineering at the University of Windsor and hopes to graduate in Fall 2009.