5-11-2013

Multi-State Memory and Logic Designs Using Multi-Quantum Channel Nano-FETs

Pawan Gogna
pawan.gogna@uconn.edu

Follow this and additional works at: http://digitalcommons.uconn.edu/dissertations

Recommended Citation
http://digitalcommons.uconn.edu/dissertations/121
Abstract

Multi-State Memory and Logic Designs Using Multi-Quantum

Channel Nano-FETs

By: Pawan Gogna

University of Connecticut, 2013

In this dissertation, implementation of multi value logic using a novel algebra and Spatial Wavefunction Switched Field Effect Transistor (SWSFET) has been explored. The quantum mechanical simulations, characteristics of the fabricated SWS structures are discussed. The novel device and quaternary algebra has been used to implement multi-value logic. The designs of quaternary SRAM cells, basic logic gates and arithmetic cells are presented. In addition, mixed signal architectures using SWSFET are explored. Simulations for the memory, logic and mixed signal designs are presented. BSIM3 equivalent channel models were used for SWSFET. Cadence Spectre Simulator and Advanced Design Simulator were used as the simulation tools. Quaternary to binary and binary to quaternary conversion circuits are also designed and presented. This helps the quaternary and binary circuits to co-exist on the same die.

Multi Valued Logic (MVL) has been in research for many decades. MVL offers benefits and opportunities but it has its own challenges. Quaternary SRAM using SWSFET reduces the number of transistors needed by 75%. Similar savings are shown for implementing quaternary logic when compared to implementation using CMOS based binary logic. In addition, significant reduction in gate delays is achieved by implemented logic using quaternary algebra. Also, logic
implementation using quaternary algebra leads to about 50% reduction in interconnect metal density for data signals. This is helpful in reducing the congestion in metal signal routing layers. Metal density, number of metal layers and pressure to use low resistivity materials have added to the die cost over recent years. Implementing MVL in main stream microprocessor needs further research in designs tools. In addition, the fabrication and transistor design need to be optimized to tune MVL based designs.
Multi-State Memory and Logic Designs Using Multi-Quantum Channel Nano-FETs

Pawan Gogna, M.S.

A Dissertation
Submitted in Partial Fulfillment of the Requirements of the Degree of Doctor of Philosophy @
University of Connecticut 2013
Multi-State Memory and Logic Designs Using Multi-Quantum Channel Nano-FETs

Presented by

Pawan Gogna, M.S

Major Advisor ______________________________________________________
Faquir C. Jain

Associate Advisor ______________________________________________________
John Chandy

Associate Advisor ______________________________________________________
Lei Wang

The University of Connecticut

2013
To

My Son

Paarth Gogna
Acknowledgements

First, I would like to thank almighty for always being there for me and for giving me the ability to finish this work and achieve this milestone. I would also like to extend my sincere gratitude to Dr. F. C. Jain for being my advisor and mentor. He was most gracious in the guidance and patience he offered me. My profound appreciation to the great minds of Dr. John Chandy and Dr. Lei Wang for serving on my advisory committee.

Words cannot express the gratitude I have for the support and love of my parents, Mr. Jagdish Gogna and Smt. Shashi Gogna. Thanks to my wife Dr. Mudita Gogna for her encouragements, time and enabling me in many ways to pursue this work. Special thanks to Dr. Mukesh Gogna, Dr. Praveen Gaba, and Dr. Ashok Gaba for their continuous support and help without whom this achievement was almost impossible.

In addition, I would like to extend my genuine gratitude to all my friends and anyone that I may have missed for supporting and helping me through all my work at UConn.
# Table of Contents

1 Introduction ........................................... 1

2 Multi Valued Logic ........................................... 4
  2.1 Number System ............................................. 5
  2.2 Quaternary Algebra ....................................... 6
  2.3 Quaternary Operators ................................... 7
    2.3.1 Quaternary Operator properties ................. 10
    2.3.2 Quaternary Algebra Theorems ...................... 12
    2.3.3 Maximum Number of Gate Count for Sum of Product Expressions for logic 13
    2.3.4 Maximum Gate Depth for Sum of Product Expressions for logic .......... 15
  2.4 Multi Value Logic (MVL) implementations ............... 17
    2.4.1 Ternary Logic Circuits .............................. 17
      2.4.1.1 Ternary Logic Levels ......................... 18
      2.4.1.2 Ternary Circuits .............................. 18
    2.4.2 MVL to Binary Conversion ......................... 22
      2.4.2.1 Ternary Encoders and Decoders ............... 22
      2.4.2.2 Quaternary Encoders and Decoders .......... 24
    2.4.3 MVL Circuits .................................. 30
      2.4.3.1 Flash Memories ................................ 30
      2.4.3.2 DRAM Memories ................................ 32
    2.4.4 MVL Arithmetic Designs ......................... 32
      2.4.4.1 MVL Adders .................................. 33
      2.4.4.2 MVL Multipliers .............................. 37

3 Spatial Wave Switching Field Effect Transistor (SWSFET) ............ 39
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>SWSFET: Structure and Operation</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>SWSFET: Two, Three and Four well Structures and Quantum Mechanical Simulations</td>
<td>45</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Two Well SWSFET</td>
<td>45</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Three Well SWSFET</td>
<td>48</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Four Well SWSFET</td>
<td>50</td>
</tr>
<tr>
<td>3.2.4</td>
<td>SWSFET Fabrication Processing and characteristics</td>
<td>57</td>
</tr>
<tr>
<td>3.2.4.1</td>
<td>SWSFET Characteristics</td>
<td>57</td>
</tr>
<tr>
<td>4</td>
<td>Novel Quaternary Logic</td>
<td>60</td>
</tr>
<tr>
<td>4.1</td>
<td>Novel Quaternary Logic</td>
<td>60</td>
</tr>
<tr>
<td>4.2</td>
<td>Logical NOT</td>
<td>61</td>
</tr>
<tr>
<td>4.3</td>
<td>Logical OR</td>
<td>62</td>
</tr>
<tr>
<td>4.4</td>
<td>Logical XOR</td>
<td>63</td>
</tr>
<tr>
<td>4.5</td>
<td>Logical AND</td>
<td>63</td>
</tr>
<tr>
<td>4.6</td>
<td>Arithmatic Full Adder</td>
<td>64</td>
</tr>
<tr>
<td>4.7</td>
<td>Logic Transition and Power Consumption</td>
<td>66</td>
</tr>
<tr>
<td>5</td>
<td>Quaternary Static Random Acess Memory (SRAM) design using SWSFET</td>
<td>68</td>
</tr>
<tr>
<td>5.1</td>
<td>SRAM Cell and Memory System</td>
<td>68</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Logic and Cell Architecture</td>
<td>69</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Simulations</td>
<td>72</td>
</tr>
<tr>
<td>6</td>
<td>Quaternary Logic and Arithmatic designs using SWSFET</td>
<td>75</td>
</tr>
<tr>
<td>6.1</td>
<td>Quaternary Logic</td>
<td>75</td>
</tr>
<tr>
<td>6.2</td>
<td>Logic Cells</td>
<td>76</td>
</tr>
<tr>
<td>6.2.1</td>
<td>NOT Gate</td>
<td>76</td>
</tr>
<tr>
<td>6.2.2</td>
<td>OR Gate</td>
<td>77</td>
</tr>
<tr>
<td>6.2.3</td>
<td>AND Gate</td>
<td>79</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Full Adder</td>
<td>80</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2-1: Ternary logic Inverter & AND gate with CMOS and resistors (implicit 1 level) ...... 19

Figure 2-2: Ternary logic Inverter and AND gate with CMOS and resistors (explicit 1 level)..... 20

Figure 2-3: General construction of ternary circuit with decoder and encoder [74]................. 22

Figure 2-4: Circuit diagrams for three encoding schemes [74]........................................... 23

Figure 2-5: Circuit diagrams for three decoding schemes [74]........................................... 23

Figure 2-6: Floating gate potential diagram for the conversion of quaternary to MSB output [72] .......................................................... 25

Figure 2-7: Circuit diagram for implementation of quaternary logic to binary logic – MSB using floating gate MOSFETs [72]................................................................. 25

Figure 2-8: Floating point potential diagram for conversion of quaternary to LSB output [72] ... 26

Figure 2-9: Floating point potential diagram for conversion of quaternary to LSB output Circuit Diagram for implementation of quaternary to binary logic – LSB using floating gate MOSFETs [72].................................................................................................. 27

Figure 2-10: Full circuit diagram for conversion of quaternary (4-valued) logic to binary bits using floating gate MOSFETs [72].................................................................................. 28

Figure 2-11: Binary to quaternary encoder using pass gate [73]............................................. 29

Figure 2-12: Quaternary to binary encoder (left) and XOR gate (right) using pass gate [73]...... 30

Figure 2-13: Block diagram quaternary full adder [71]. ....................................................... 34
Figure 2-14: Logic diagram of encoder circuits [71]. ................................................................. 35

Figure 2-15: Block Diagram of code generator for quaternary input X and Y [71]. .............. 35

Figure 2-16: Code generator circuit for quaternary input X and Y [71]. ................................. 36

Figure 2-17: Circuit Diagram for sum block for quaternary adder [71]. ............................... 37

Figure 2-18: Circuit Diagram for carry block for quaternary full adder [71]......................... 37

Figure 3-1: Schematic cross-section of a transport channel comprising 2 InGaAs quantum wells sandwiched between AlInAs barriers in a SWS-FET realized on an InP substrate (left) and schematic energy band diagram (right) [2]................................................................. 39

Figure 3-2: Simulated energy band diagram of a two-quantum-well SWS device showing carriers in the lower well W2 at gate voltage ................................................................. 43

Figure 3-3: Energy band diagram showing carriers shifted to the upper well W1 when the gate voltage is increased to Vg = 2.0 V ................................................................. 43

Figure 3-4: Channel charge density in each well of lower well W2 labeled “Last QW” & upper well well W1 labeled “First QW” separately and both wells together ........................................ 44

Figure 3-5: Simulated low-frequency C–V characteristic showing a peak corresponding to transfer of electrons from well W2 to well W1 ................................................................. 44

Figure 3-6: SWS-FET having an asymmetric-coupled quantum well channel. .................. 46

Figure 3-7: Schematic energy band of a coupled well SWS-FET ........................................ 46

Figure 3-8: Carriers in Si lower well W2 ........................................................................... 47

Figure 3-9: More carriers in lower well W2 than upper well W1 ....................................... 47
Figure 3-10: Carriers in Si upper well W1. ................................................................. 48

Figure 3-11: Three Ge well SWSFET ............................................................................ 49

Figure 3-12: Carrier Wavefunction located in lower most well W3 ......................... 49

Figure 3-13: Carrier Wavefunction in upper most well W1 ...................................... 50

Figure 3-14: Four well Ge well ZnSSe barrier based SWSFET ............................... 51

Figure 3-15: Four-QW SWS wave-function in W4 (Vg = -3.8V) ............................ 51

Figure 3-16: Four-QWSWS wave-function in W3(Vg = -3.5V) .............................. 52

Figure 3-17: Four-QWSWS wave-function in W2 (Vg = -3.2V) ............................ 52

Figure 3-18: Four-QW wave-function in W1 (Vg = -3.0) ........................................ 53

Figure 3-19: Charge density plot as a function of gate voltage in various quantum wells ... 53

Figure 3-20 : Four Well SWSFET with InGaAs .......................................................... 55

Figure 3-21: Wavefunctions in lower wells W4 and W3 .............................................. 55

Figure 3-22: Wavefunction in upper most well W1 .................................................... 56

Figure 3-23: Wavefunction in upper most well W1 (different barrier height of gate insulator) ... 56

Figure 3-24: SWS-MOS capacitor-voltage plot (theoretical simulation model) ........... 58

Figure 3-25: Capacitance variation due to gate voltage in fabricated 2-well InGaAs SWS-FET. 58

Figure 3-26: C-V plot for a 4-well InGaAs SWS MOS sample 2056 under inversion .......... 59

Figure 4-1 : Binary logic verses Novel Quaternary logic with two inversions ............... 61
Figure 5-1: Binary Inversion logic for SRAM using CMOS. ......................................................... 69

Figure 5-2: Quad Inversion logic for SRAM using SWSFET ......................................................... 69

Figure 5-3: Binary to quaternary conversion circuit ................................................................. 70

Figure 5-4: Quaternary to Binary conversion circuit ............................................................... 70

Figure 5-5: Two bit SRAM using CMOS devices with binary logic ........................................... 71

Figure 5-6: Two bit SRAM structure using SWSFET devices with quaternary logic ................. 71

Figure 5-7: Input voltage vs time waveform. .............................................................................. 72

Figure 5-8: Output vs time for 25nm SWSFET based cell. ......................................................... 73

Figure 6-1 : Binary logic levels with two Inversions ................................................................. 75

Figure 6-2: Quaternary logic levels with two inversions ......................................................... 75

Figure 6-3: Quaternary NOT gate implemented with SWSFET ................................................. 76

Figure 6-4: Two NOT gates implemented with binary CMOS logic ..................................... 76

Figure 6-5 : Quaternary OR gate implemented using SWSFET .............................................. 77

Figure 6-6: Two binary OR gates using CMOS ................................................................. 78

Figure 6-7 : Quaternary AND gate implemented using SWSFET ........................................ 79

Figure 6-8: Two binary AND gates using CMOS ................................................................. 79

Figure 6-9 : Two binary full adders implemented using CMOS ........................................ 82

Figure 6-10: Quaternary full adder implemented using SWSFET ........................................ 82
Figure 6-11: Quaternary Latch implanted using SWSFET ...................................................... 83

Figure 6-12: Quaternary SWSFET based SRAM cell ............................................................... 84

Figure 6-13: Two CMOS based binary logic SRAM cells equivalent to one quaternary cell .... 85

Figure 6-14: Binary logic to quaternary logic conversion circuit ............................................. 86

Figure 6-15: Quaternary logic to binary logic conversion circuit .......................................... 87

Figure 6-16: SWSFET based two input quaternary AND gate .................................................. 88

Figure 6-17: SWSFET based two input quaternary OR gate .................................................... 88

Figure 6-18: Simulation results for SWSFET based quaternary inverter using BSIM equivalent channel models. .............................................................................................................. 89

Figure 6-19: Simulation results for SWSFET based quaternary inverter using BSIM equivalent channel models. .............................................................................................................. 89

Figure 7-1: SWSFET switch cap based based two-bit digital to analog converter ..................... 92

Figure 7-2: SWSFET switch cap based based four-bit digital to analog converter ................. 92

Figure 7-3: Fast Digital to Analog Converter (DAC) designs using SWSFET and switch capacitor architectures .................................................................................................................. 93

Figure 7-4: Simulation results quaternary algebra and SWSFET based four bit (two quidt) digital to analog converter architecture presented in figure 7-2 ......................................................... 94
List of Tables

Table 2-1: Quaternary Operators ................................................................. 7
Table 2-2: Quaternary Operators with Quaternary Operands .................... 8
Table 2-3: Single-variable function truth tables ........................................ 21
Table 2-4: Two-variable function truth tables ........................................... 21
Table 3-1: Parameters used in the simulation of wavefunctions in Ge quantum wells* ........ 54
Table 4-1: Truth table implementing NOT Gate ......................................... 61
Table 4-2: Truth table implementing OR Gate ........................................... 62
Table 4-3: Truth table implementing XOR Gate ........................................ 63
Table 4-4: Truth table implementing AND gate ........................................... 64
Table 4-5: Truth table implementing two bit Full adder ............................ 65
Table 5-1: Comparison of SWSFETs vs. CMOS SRAM cell ....................... 74
Table 6-1 : Truth table implementing NOT Gate. ..................................... 77
Table 6-2 : Truth table implementing OR Gate. ....................................... 78
Table 6-3: Truth table implementing AND gate ........................................... 80
Table 6-4: Truth table implementing two bit Full adder ............................ 81
Table 6-5: Truth table for a latch ............................................................... 83
Table 6-6 : Transistor count per two bits for digital building blocks ............. 85
1 Introduction

Binary logic has been used for electronic designs for many decades. The challenges of the binary logic have inspired the researchers for research in multi-valued logic (MVL). For example, with binary logic designs, chip interconnects occupy so much area that it limits the performance of the VLSI chips [1]. Binary algebra also takes more digits to represent any number when compared to any higher radix algebra [4]. Going to Multi valued logic may have some efficient solutions for these challenges. Use of multi value logic quaternary logic for interconnect is definitely a good way to reduce the data related metal interconnect density to half. In addition quaternary logic algebra inherently processes the information for efficiently and reduced the number of transistors required to compute a calculation when compared to processing in binary logic. In quaternary logic the levels are represented by 0, 1, 2, 3. Significant research has been done on multi valued logic and quaternary logic in past few decades [1,4]. Researchers have explored the suitable algebra for logic and arithmetic, optimized the logic algorithms, and developed the test algorithms for multi valued logic. Though so much work has been put into research we still do not see much of implementations in MVL. One of the reasons for this is availability of suitable transistors to implement all these logic and arithmetic functions offered by the MVL research.

Spatial Wavefunction Switched Field Effect Transistor (SWSFET) was introduced by Jain et al [2,3]. The transistor offers some unique features and characteristics that are not available in CMOS or BJT. The salient feature is that it offers multiple vertically stacked channels. Also the location carrier concentration among these channels could be controlled via the gate voltage in these transistors. As we will see in the following chapters, these features of SWSFET help enable multi value logic in voltage mode. It’s important to mention that voltage mode is important part of the implementation. One of the reasons that CMOS became a most
popular method of implementing digital design was that it offered voltage mode implementation for the binary logic. Voltage mode design implementation offers the low static power. This means lower power density on the die and also the lower cost for thermal solutions for these VLSI chips.

In the following chapters we will discuss details of the SWSFET feature, work done in MVL and our MVL designs. Chapter 2 elaborates lots of research done towards multi values logic in general, and ternary and quaternary logic in specific. Also it discusses the methods, algebra and some of the previous work done towards implementation of these different algebras. The advantages, arithmetic block designs and test methodologies are touched upon.

Chapter 3 introduces SWSFET design along with quantum simulations. The design features, device topologies for multiple designs are discussed. This chapter lays the foundation of quaternary logic design implementations. Also the characteristics of fabricated SWS structures are presented to demonstrate the spatial wavefunction switching (SWS) behavior of the device.

Chapter 4 introduces the novel quaternary logic and algebra to harvest the capabilities of SWSFETs. The algebra used was design to keep easy equivalence to binary algebra. This makes it easy to convert from binary to quaternary and vice versa. The easy conversion to binary enables the co-existence of binary and quaternary circuits on the same die. Some circuits may be easier and more beneficial to implement in quaternary logic while the other may be better implemented in binary logic. As an example, one such scenario is implementing cache in quaternary logic while keeping the core design in binary. Cache occupies over 50% area in performance microprocessors. The implementation of cache in quaternary logic could save up to 75% of this over half the die area. Easy conversions make the hybrid design implementations possible.

Chapter 5 presents the basic logic design to implement MVL inverter and SRAM cell. The design schematics, simulations and advantages over the binary logic CMOS using design discussed.
Chapter 6 introduced more general basic MVL logic and arithmetic designs using SWSFETs. Design of basic gates AND, OR, NOT, XOR and Latch are presented. Also full adder design is introduced. The simulations for these basic gates are carried out and the results are discussed. The advantages of the MVL logic and arithmetic operations in MVL are summarized.

Chapter 7 presents the mixed signal logic design using SWSFET. The chapter introduces novel designs of digital to analog converters. SWSFET features offer the fast and compact design architectures of these MVL digital to analog converters. The digital signals in these designs were used as MVL levels or quaternary levels to make the design more efficient.

Finally, this thesis concludes with summary and future directions.
2 Multi Valued Logic

Multi-value logic (MVL) has been in research for some time now [1]. Various researchers have explored the algebra, circuit techniques and benefits [4]. While the computers mostly use binary algebra and computation methodology, the world around us is multi valued. Many practical problems could use more efficient solutions using multi-valued logic. Audio and video signals a few very common forms of data being processed digitally. This data is inherently multi valued and would be processed in an efficient way using MVL algebra. In addition, MVL has many other useful applications. Broadly these could be divided into two groups. The first one relates to solving Boolean problems in an efficient manner. For example, Berkley’s tool for verification and synthesis VIS uses the approach to convert multiple Boolean outputs to single MVL output and run the VIS efficiently [21]. The second group includes the problems that use the more than two level signals for designing electronic circuits. These circuits include multi-values logic, memories, FPGA, arithmetic circuits etc. MVL also offers compaction of the interconnect signals along with the several potential opportunities for the improvement of current VLSI circuit designs [26]. In addition to this, arithmetic operations using MVL makes the VLSI circuits perform efficiently with lower number of gates and delay stages [5]. For example using four or higher number of levels for an adder reduces the number of ripple carries to less than half when compared to implementation using binary circuit [70,71]. This could result in increasing the speed of electronic designs. Even though MVL presents all these potential advantages, use of MVL is limited by the circuits that could be implemented using the available process and circuit technologies that allow MVL designs. In addition, these implementations have to be compatible with the currently-in-use binary technologies [74]. In this chapter we review some of the theories and practical work available on multi-value logic.
2.1 Number System

The computing systems mostly use digital technology and use discrete symbols to represent information. Present day digital technology mostly uses binary system with just two discrete levels 0 and 1. This may be helpful in the computing circuit implementation but the real world around us is not binary. For that reason the optimal solutions for the real world problem may not exist in the binary space. For example, the arithmetic calculations would be close to our experience if carried out in decimal number system. Also, any practical audio and video signals have many levels and need to be represented as such. Processing of these signals using MVL would not only cut the MVL to binary conversion and vice versa but any mathematical processing would also be more efficient [26]. In addition, in many engineering problems we do not have only ON of OFF state; we need HIGHZ state to be practical. This section gives a brief overview of how the numbers could be represented in MVL as compared to binary system.

A number system is a convention used to represent a value. If the system represents only the positive values, it is called the unbalanced system [31,32]. And if it represents both positive and negative values, it is called balanced or signed system [29,30]. Unbalanced system is the most common and represents 0, 1, 2, 3, 4 …. N-2, N-1 while the balanced system represents -(N-1), -(N-2) …. -4, -3, -2, -1, 0, 1, 2, 3, 4 …. N-2, N-1. In a number system, a string of digits \(x_0, x_1, \ldots, x_{m-1}\) over a set of \(n\) values represents the following number.

\[x_0 + n^1 x_1 + n^2 x_2 + \ldots + n^{m-1} x_{m-1}\]

Where ‘\(n\)’ is called the base of the number system, for binary the base \(n = 2\) and for ternary and quaternary systems the base is equal to 3 and 4 respectively. In addition, an unbalanced ternary system could have \(x_i\) value belonging to \{0, 1, 2\} while a balanced ternary system will have the \(x_i\) belonging to \{-1, 0, 1\}. And for quaternary system \(x_i\) belongs to \{0, 1, 2, 3\}. Boolean algebra based digital systems only two levels are available. This inherently makes
the binary system as unbalanced [11, 12] but the multiple techniques could be used to represent negative numbers in the binary system. These techniques are called signed magnitude, 1’s complement and 2’s complement. In signed magnitude, an additional digit is used to represent the sign. To represent a negative number in 1’s complement each bit is subtracted from 1. To represent a negative number using 2’s complement, each bit is subtracted from 1 and then 1 is added to the resulting number. Alternatively, in a balanced ternary system, a negative number could be represented by changing 1s to -1 in a number while keeping the 0 unchanged. This negates the need to need for additional hardware for subtraction over the addition. Additional advantage of the balanced number system is that in such a system, rounding and truncation are identical. This is not possible in binary system.

2.2 Quaternary Algebra

Quaternary logic has multiple advantages over the binary logic [5, 7, 15, 18, 19, 32]. While the binary and quaternary numbers can easily find equivalence among other system. It takes only half the digital to represent a number in quaternary system when compared to binary. This makes the quaternary system especially attractive for memory implementation [6]. Similarly, quaternary system needs only half the number of signals to carry the same amount of data. This section goes over some basic quaternary operators and functions available in literature.

As we discussed in the previously, quaternary algebra includes set of operators and set of values {0, 1, 2, 3} for any value proposition. Each Quaternary digit could very simply be represented as binary equivalent [0 = 00, 1 = 01, 2 = 10 and 3 =11]. A quaternary digit is called qudit. This property of quaternary number system makes it attractive for practical use in computational circuits along with binary circuits. Also note that, when the binary-equivalents interchange their positions while keeping the quaternary state unchanged, the states are symmetrical. If this is not the case, then those are asymmetrical. Note that {1,2} are asymmetrical
while \{0, 3\} are symmetrical. Several physical and theoretical variants of quaternary logic are available in the literature [4][14][16][17][19]. Here a comprehensive and coherent set of quaternary algebra is presented, which could be used to implement all binary equivalent functions.

2.3 Quaternary Operators

The quaternary operators could be broadly classified into two groups. The first group includes the fundamental operators, which are sufficient to completely define the quaternary algebra. These operators could also be used to derive other operators. The second group includes the functional operators. These operators are the combinations of two or more fundamental operators. These operators could be used to express any arbitrary quaternary function. Table 2.1 gives the fundamental and functional quaternary operators [5].

<table>
<thead>
<tr>
<th>Quaternary Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fundamental Operators</strong></td>
</tr>
<tr>
<td>AND, OR, NOT, BitSwap</td>
</tr>
</tbody>
</table>

As discussed previously, a quaternary digit could be represented in terms of two binary digits packed together.

\[
A = \langle a_1, a_0 \rangle \equiv 2 \times a_1 + a_0 \quad \ldots 2.1
\]
Here \(a_0\) and \(a_1\) are the binary constituent bits of quaternary digit \(A\). Also the right side of the Eq 2.1 is decimal equivalent of the quaternary digit. With the above notion fundamental binary dyadic operators could be presented the form of bitwise binary operator

\[
F(A,B) = F(\langle a_1, a_0 \rangle, \langle b_1, b_0 \rangle) = \langle f(a_1, b_1), f(a_0, b_0) \rangle \quad \cdots 2.2
\]

In 2.2, \(F\) stands for quaternary operator while \(f\) stands for corresponding binary operator. The notation is called the packed-binary representation for quaternary numbers. The fundamental and functional operators are below in table 2.2 [5]

<table>
<thead>
<tr>
<th>Operands</th>
<th>A</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>NOT</td>
<td>(\bar{A})</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Outward Inverter</td>
<td>!A</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BitSwap</td>
<td>(~A)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inward inverter</td>
<td>A'</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AND</td>
<td>A.B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>OR</td>
<td>A+B</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>XOR</td>
<td>A(\oplus) B</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Equality</td>
<td>E(A,B)</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>MIN</td>
<td>A.B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MAX</td>
<td>A + B</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

In addition to the operators defined in table 2.2, A new derivative operator could be realized when an operator is followed by another. For example a bitswap NOR (NOR followed by
a bitswap) or bitswap XOR (XOR followed by a bitswap) could be defined as a derivative operators. A set of unique symbols could be defined for each of the operators in table 2.2 for circuit representation.

It could be proved that of the operators listed in table 2.2, the following three sets of operators could be used to realize any arbitrary quaternary function.

i. AND, OR, NOT, Bitswap

ii. AND, OR, Equality

iii. MIN, MAX, Equality

Next, the properties of quaternary algebra and above operators are reviewed. These properties help express and manipulate the complex algebraic functions.

1. Closure:

For all dyadic operators, \( F(A,B) \subseteq \{0, 1, 2, 3\} \), as evident from definition. For every unary operator \( G(A) \subseteq \{0, 1, 2, 3\} \)

2. Complement:

For unary operator NOT the following properties hold true
i. \( A + \overline{A} = \langle a_1 + \overline{a_1}, a_0 + \overline{a_0} \rangle = \langle 1, 1 \rangle = 3 \)

ii. \( A \cdot \overline{A} = \langle a_1 \cdot \overline{a_1}, a_0 \cdot \overline{a_0} \rangle = \langle 0, 0 \rangle = 0 \)

3. Associativity:

i. \( A + (B + C) = \langle a_1 + (b_1 + c_1), a_0 + (b_0 + c_0) \rangle = \langle (a_1 + b_1) + c_1, (a_0 + b_0) + c_0 \rangle = (A + B) + C \)

ii. \( A \cdot (B \cdot C) = \langle a_1 \cdot (b_1 \cdot c_1), a_0 \cdot (b_0 \cdot c_0) \rangle = \langle (a_1 \cdot b_1) \cdot c_1, (a_0 \cdot b_0) \cdot c_0 \rangle = (A \cdot B) \cdot C \)
4. **Commutativity:**
   
i. \( A + B = (a_1 + b_1, a_0 + b_0) = (b_1 + a_1, b_0 + a_0) = B + A \)

   ii. \( A.B = (a_1, b_1, a_0, b_0) = (b_1, a_1, b_0, a_0) = B.A \)

5. **Distributivity:**
   
i. \( A + (B.C) = (a_1 + (b_1.c_1), a_0 + (b_0.c_0)) =
   
   \( ((a_1 + b_1). (a_1 + c_1), (a_0 + b_0). (a_0 + c_0)) = (A + B). (A + C) \)

   ii. \( A.(B + C) = (a_1, (b_1 + c_1), a_0, (b_0 + c_0)) =
   
   \( ((a_1.b_1) + (a_1.c_1), (a_0.b_0) + (a_0.c_0)) = (A.B) + (A.C) \)

6. **Boundedness:**
   
i. \( A + 0 = (a_1 + 0, a_0 + 0) = (a_1, a_0) = A \)

   ii. \( A.3 = (a_1.1, a_0.1) = (a_1, a_0) = A \)

   iii. \( A + 3 = (a_1 + 1, a_0 + 1) = (1,1) = 3 \)

   iv. \( A.0 = (a_1.0, a_0.0) = (0,0) = 0 \)

**2.3.1.1 Quaternary Operator properties**

Next the properties of quaternary operators are reviewed. Here ten such properties are discussed.

1. **Bitswap operator distributes itself over AND & OR operators as per the following.**
   
i. \( \sim (A + B) = \sim (a_1 + b_1, a_0 + b_0) = \sim (a_1, a_0) + \sim (b_1, b_0) = \sim A + \sim B \)

   ii. \( \sim (A.B) = \sim (a_1, b_1, a_0, b_0) = \sim (a_1, a_0). \sim (b_1, b_0) = \sim A. \sim B \)

2. **NOT obeys the De Morgan’s law, when applied to the output of OR or AND gates.**
   
i. \( \overline{A+B} = \overline{(a_1 + b_1, a_0 + b_0)} = \overline{(a_1. b_1, a_0. b_0)} = \overline{A}. \overline{B} \)

   ii. \( \overline{A.B} = \overline{(a_1, b_1, a_0, b_0)} = \overline{(a_1. + b_1, a_0 + b_0)} = \overline{A} + \overline{B} \)
3. Outward inverter also obeys the De Morgan’s law, when applied to the output of OR or AND gates.

   i. \(! (A + B) = ! (a_1 + b_1, a_0 + b_0) = (\overline{a_1 + b_1}, \overline{a_0 + b_0}) = (\overline{a_1}, \overline{a_0}).(\overline{b_1}, \overline{b_0}) = ! A！B\)

   ii. \(! (A \cdot B) = ! (a_1 \cdot b_1, a_0 \cdot b_0) = (\overline{a_1 \cdot b_1}, \overline{a_0 \cdot b_0}) = (\overline{a_1}, \overline{a_0}).(\overline{b_1}, \overline{b_0}) = ! A！B\)

4. There is no compact expression that can be used to express the distribution of inward inverter over AND or OR operators.

   i. \((A + B)' = ((a_1 + b_1, b_0 + a_0))' = (a_1 + b_1, a_1 + b_1)'

   ii. \((A \cdot B)' = ((a_1 \cdot b_1, b_0 \cdot a_0))' = (a_1 \cdot b_1, a_1 \cdot b_1)'

The above two expressions cannot not be expressed in the form similar to \(\langle f(a_1 \cdot b_1), f(a_0 \cdot b_0) \rangle\), this means inward inversion following AND or OR operation cannot be expanded to any algebraic expression.

5. The order of inward inverter and NOT can be reversed

   \((\overline{A})' = (\overline{a_1}, \overline{a_0})' = (a_1, a_1) = (\overline{a_1}, a_1) = (\overline{A})\)

6. The order of outward inverter and NOT can be reversed

   \(! (\overline{A}) = ! (\overline{a_1}, a_0) = (a_1, a_0) = (\overline{a_1}, a_0) = (! A)\)

7. The order of bitswap and NOT can be altered

   \(\sim (\overline{A}) = \sim (\overline{a_1}, a_0) = (a_0, a_1) = ! (a_0, a_1) = (\sim A)\)

8. The order of bitswap and inward inverter can be altered under certain condition, not generally.

   i. \((\sim A)' = ((a_0, a_1))' = (a_0, a_0)\)

   ii. \((\sim A)' = (\overline{a_0}, a_1) = (a_1, a_1)\)

This means that \((\sim A)' = (\sim A)\) if and only if \(a_1 = \overline{a_0}\) that is if \(A\) is asymmetric.
9. The order of bitswap and outward inverter can be altered under certain conditions, not generally.

   i. \( \sim (!A) = \sim (\overline{a_1, a_1}) = (\overline{a_1, a_1}) \)
   
   ii. \( ! (\sim A) = ! (a_0, a_1) = (\overline{a_0, a_0}) \)

   This means, \( \sim (!A) = ! (\sim A) \) if and only if \( a_1 = a_0 \), that is if \( A \) is symmetric.

10. The order of inward and outward inverter can never be reversed under any condition.

   i. \( (\overline{a, a})' = (\overline{\overline{a_1, a_1}})' = (a_1, a_1) \)
   
   ii. \( ! (A') = ! (\overline{a_1, a_1}) = (a_1, a_1) \)

   This means, \( (\overline{a, a})' \neq !(A') \) under any condition.

### 2.3.1.2 Quaternary Algebra Theorems

The quaternary algebra and operator properties discussed in this section could be applied to derive theorems of quaternary algebra. These theorems are helpful in simplifying the complex algebraic expressions.

1. **Law of Idempotency**
   \[
   A + A = A, \quad A.A = A
   \]

2. **Law of Absorption**
   \[
   A + (A.B) = A, \quad A. (A+B) + A
   \]

3. **Law of Identity**
   \[
   A + B = A, \quad A.B = A; \text{ for } A = B
   \]

4. **Law of complements with NOT**
   \[
   A + B = 3, \quad A.B = 0; \text{ for } A = \overline{B}
   \]
5. **Law of Involution with NOT and bitswap**

\[ A = \overline{A}, \quad A = \sim (\sim A) \]

6. **Law of Elimination with NOT**

\[ X + \overline{X}.Y = X + Y, \quad X (\overline{X} + Y) = X.Y \]

7. **Law of Consensus with NOT**

\[ X. Y + \overline{X}.Y + Y.Z = X.Y + \overline{X}.Z \]

\[ (X + Y). (\overline{X} + Z). (Y + Z) = (X + Y). (\overline{X} + Z) \]

8. **Law of Interchange with NOT**

\[ (X. Y) + (\overline{X}.Z) = (X + Y). (\overline{X} + Z) \]

\[ (X + Y). (\overline{X} + Z) = (X. Z) + (\overline{X}.Y) \]

It can be easily shown that any arbitrary logic function can be implemented using sum of products. Sum of products could be represented in a vectorized or non-vectorized form.

The expression below is the combination of vectorized and non-vectorized forms.

\[ \sum X.Y \equiv X_1.Y_1 + X_2.Y_2 + X_3.Y_3 + \ldots + X_n.Y_n \]

The right side of the above equation is non-vectorized. Looking at the left side of the equation, AND operation is non-vectorized while the OR operation is vectorized.

### 2.3.1.3 Maximum Number of Gate Count for Sum of Product Expressions for logic

It was discussed previously that any arbitrary function could be implemented using the sum of products expression. This holds true in quaternary logic and algebra. This section discusses the number of maximum gates and gate depth required to evaluate the sum of products expression to implement any arbitrary function.

If the maximum number of inputs on an AND/OR gate is \( v \), then the maximum number of gates required to compute AND/OR of \( n \) propositions is \[ \left\lceil \frac{n-1}{v-1} \right\rceil \]
To prove this theorem, let’s assume that we have to compute AND of \( n \) proposition. Also let’s assume that maximum number of inputs on an AND gate is \( v \) and that \( n \) is larger than \( v \). Now the number of gates used will be minimum if not more than one gate is used with less than \( v \) inputs. The gate-delay would be minimum if the inputs are divided and processed in parallel using AND gates. If there are \( x_0 \) such gates processing exactly \( v \) number of inputs, it gives the following –

\[
v x_0 + r_0 = n
\]

here,

\[
x_0 = \left\lfloor \frac{n}{v} \right\rfloor, \quad r_0 = n - v \left\lfloor \frac{n}{v} \right\rfloor
\]

The above equations give \( x_0 \) gates with \( x_0 \) outputs and \( r_0 \) propositions left in case \( n \) is not divisible by \( v \). This makes the next level with \( x_0 + r_0 \) propositions. Assuming that this level has \( x_1 \) gates and \( r_1 \) propositions, the following holds for this level-

\[
x_0 = v x_1 + r_1 - r_0
\]

and so on, in general the following can be written for \( m^{th} \) level,

\[
x_{m-1} = v x_m + r_m - r_{m-1}
\]

Assuming that the gate tree stops with the \( (m+1)^{th} \) level as the last level with only one AND gate. In this condition, the number of propositions are less than or equal to \( v \). Let’s assume there are \( \delta < v \) unused inputs of AND gate in this level. This gives –

\[
x_m = v - \delta - r_m
\]
Summing of above three equations gives –

\[
\sum_{i=0}^{m} x_i = v \sum_{i=0}^{m} x_i + v - \delta - r_0
\]

Simplifying the above equation, it becomes –

\[
\sum_{i=1}^{m} x_i = \frac{x_0 - v + \delta + r_0}{v - 1} = \frac{\lfloor \frac{n}{v} \rfloor - v + \delta + n - v \lfloor \frac{n}{v} \rfloor}{v - 1}
\]

Hence the total number of gates comes to –

\[
N = 1 + x_0 + \sum_{i=1}^{m} x_i = \frac{\lfloor \frac{n}{v} \rfloor + \lfloor n - v \rfloor \lfloor \frac{n}{v} \rfloor - 1}{v - 1} = \frac{n - 1}{v - 1} + \frac{\delta}{v - 1}
\]

Here \(\delta\) is only used to account for unused inputs so that the right and the left sides of the equation above equate. Given this the number of gates needed for this implementation can be given as –

\[
N = \left\lfloor \frac{n - 1}{v - 1} \right\rfloor
\]

Note that the number of unused inputs \(\delta\) does not have to reside only in the last level. The above equation holds true regardless of the level that has the unused inputs.

2.3.1.4 Maximum Gate Depth for Sum of Product Expressions for logic

After calculating the maximum gate count needed for implementing a logic function, this section focuses on computing the maximum gate depth needed to compute AND/OR of \(n\) propositions with the AND/OR gates with maximum \(v\) number of inputs. In general case like this, it is possible to compute AND/OR with \(\left\lfloor \log_v n \right\rfloor\) depth of operators.
To prove this theorem, let’s look at the logic tree. It can be easily seen that in a logic tree to implement the AND/OR gate would reduce by a factor or if \( n = v^k \). For this and the gate depth of \( d \), it can be shown that –

\[
d = k
\]

In case \( n = v^{k+1} \)

\[
d = k + 1
\]

For the above two cases, it can be seen that the \( d \) can be given by the following –

\[
d = \log_v n
\]

Let’s consider the event for which \( d \) lies between \( k \) and \( k+1 \), this would mean that that following would hold true –

\[
v^k < n < v^{k+1}
\]

In this case \( n \) can be expressed as –

\[
n = a_k \cdot v^k + a_{k-1} \cdot v^{k-1} + \ldots + a_0
\]

Here \( a_i \) belongs to \{0, 1, 2, 3…….\( v-1 \}\} and atleast two of the \( a_i \)’s including \( a_k \) must be non-zero. This gives us –

\[
v - a_i \geq 1
\]

From the last two equations, it can be determined that AND of \( n \) propositions can be calculated within \( k+1 \) gate levels. Now, let’s review the cases with \( n = v^k \), \( v^k < n < v^{k+1} \) and \( n = v^{k+1} \), it can be concluded that the gate depth can be given by the following equation –

\[
d = \lceil \log_v n \rceil
\]
2.4 Multi Value Logic (MVL) implementations

The last section reviewed MVL algebra, operators along with some properties and theorems. This section goes over some of work related to MVL implementation. Ternary logic circuit along with circuits for conversion to binary and vice versa are discussed. Also quaternary to binary conversion circuits, binary to quaternary conversion circuit, performance quaternary and MVL adders, MVL multipliers, MVL flash memories and MVL DRAM memories are discusses.

2.4.1 Ternary Logic Circuits

Multi-Value logic has been in research for decades. A significant amount of research work has been done for ternary and quaternary logic. This section discusses some of the work done on ternary logic. The researchers [74] have pointed out many benefits of using ternary logic as MVL. “(a) since 3 is the smallest radix higher than binary, ternary functions and circuits have the simpler form and construction. They can be studied and discussed easily, yet they still display the characteristics of multi-valued elements. (b) As a measure of the cost or complexity of multi-valued circuits, the product of the radix and the number of signals has been proposed. Since 3 is the digit nearest to $e = 2.718$, ternary circuits will be more economical according to this measure. (c) if balanced ternary logic (1,0,-1) is used, the same hardware may be used for addition and for subtraction. (d) since 3 is not an integral power of 2, research on ternary logic may disclose design techniques that are overlooked in the study of binary or quaternary logic.” Besides these reasons, there were obvious benefits which includes that using MVL logic the chip interconnects could transmit more information than in case of binary designs. Number of pin outs from the chip could be reduced since the more information would be carried per pin. Serial IOs could carry more information and hence would lead to higher speed of serial IOs. In addition, any higher level algebra would process information more efficiently when compared to binary. This could mean smaller and compact circuit designs for same computing functions. Though there have been
many publications on ternary designs, this section focuses on one such work with algebra and related circuit components.

2.4.1.1 Ternary Logic Levels

Ternary logic consists of three logic levels belonging to the set \( \{0, 1, 2\} \). In an electronic circuit this could be represented by three voltage levels, namely low, mid and high voltage levels. Practically, the circuits need to detect two different threshold voltages to identify these levels. Hence one way to design the ternary circuits is to have the two different thresholds in the circuits and then functional output performs as per the ternary algebra presented below.

Some of the ternary algebraic functions are given below in table 2.3 [74] and table 2.4 [74]. Table 2.3 gives the truth table with one-variable functions while the table 2.4 gives the truth table of two-variable functions.

2.4.1.2 Ternary Circuits

Ternary circuit designs have been demonstrated by researchers using CMOS transistors and resistors. The design of ternary logic circuits needs two threshold levels to implement three states of ternary logic. This could be accomplished by using PMOS or NMOS transistor with appropriately skewed threshold voltages. PMOS and NMOS threshold could be designed such that, there exists three regions with first region of only PMOS ON (or OFF), second region of only NMOS ON (or OFF) and third region with both NMOS and PMOS ON (or OFF). This method could be used to make basic ternary logic gates. Figure 2.1 and 2.2 [74] gives the design of ternary inverter and AND gate using this method. While designs in figure 2.1 used the resistors as a voltage divider to give state 1, the designs in figure 2.2 shows explicit “1” state level supply connected to the output node via a resistor. In figure 2.1 the PMOS threshold \( T_P = -0.5 \) and NMOS threshold \( T_N = 0.5 \). For figure 2.2, the PMOS threshold \( T_P = -1.5 \) and NMOS threshold
\( T_N = 1.5 \). The similarity to CMOS logic gates can be clearly perceived. This similarity makes the circuits ingenious and attractive.

Figure 2-1: Ternary logic Inverter & AND gate with CMOS and resistors (implicit 1 level)
To process the ternary logic using binary components [75] the designs of decoders and encoders was proposed. This helps get rid of the resistor components in the ternary circuits proposed in figure 2.1 and figure 2.2. In the next section, a design of binary to ternary and ternary to binary conversion circuits is presented.
Table 2-3: Single-variable function truth tables

<table>
<thead>
<tr>
<th>x</th>
<th>( \bar{x} )</th>
<th>( 0 \cdot x )</th>
<th>( 1 \cdot x )</th>
<th>( 2 \cdot x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2-4: Two-variable function truth tables

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a . b</th>
<th>!( a . b )</th>
<th>a + b</th>
<th>!( a + b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
2.4.2 MVL to Binary Conversion

In vast amounts of research on MVL it is widely prevalent view that MVL and binary circuits need to work side by side. While MVL may be able to implement most of the usable logic functions, it is possible that better methodologies or circuit designs could be available in binary domain. This may happen for a variety of reason. For example, optimized designed may be available in binary which have not been yet researched in a particular MVL design. This makes it crucial that the designs to convert binary logic signals to MVL signals and MVL signals to binary logic signals be available. This section discusses some of the work and available methods to implement these conversions.

2.4.2.1 Ternary Encoders and Decoders

For processing ternary data sometimes it’s convenient to convert the ternary signal into binary and processes it in binary domain. After the processing the signal could be converted back to ternary signal. Figure 2.3 presents the general block diagram for this scheme [74]. For encoding a binary to a ternary signal, multiple encoding schemes could be used. Figure 2.4 [74] gives three circuit diagrams for one such encoding scheme used in [74]

Figure 2-3: General construction of ternary circuit with decoder and encoder [74].
Figure 2-4: Circuit diagrams for three encoding schemes [74]

Figure 2-5: Circuit diagrams for three decoding schemes [74]
To convert the ternary to binary the circuit diagrams of the required decoders are presented in Figure 2.5 [74]. The decoders use multi-threshold MOS transistors.

### 2.4.2.2 Quaternary Encoders and Decoders

Quaternary logic has the advantage that it could be mapped easily to two binary bits. The number of levels in quaternary digit (0, 1, 2 and 3) is exactly the same that could be represented by a two digit binary number (00, 01, 10 and 11). Researchers have picked up specific mapping as per convenience for their research and methodologies. Several methods have been proposed for binary to quaternary and vice versa converters [76, 78, 79, 80]. This section touches upon a few of these methods.

#### 2.4.2.2.1 Quaternary to Binary Convertor using multiple-input floating gate MOSFETs

A novel quaternary to binary convertor was presented in [72]. This paper breaks down the circuit response needed for LSB and MSB extraction from a quaternary digit. This section will go over the simplified methodology used using floating gate MOSFETs. Figure 2.6 and 2.7 [72] give the required characteristics and circuit diagram for extracting MSB from a quaternary digit. Vdd for this design is 3V and the circuit gives MSB to be “1” for Vin > 1.45 V.
Figure 2-6: Floating gate potential diagram for the conversion of quaternary to MSB output [72]

Figure 2-7: Circuit diagram for implementation of quaternary logic to binary logic – MSB using floating gate MOSFETs [72]
In the similar fashion Figure 2.8 [72] gives the floating point potential diagram (FPD) for extracting LSB from the quaternary signal. The circuit design implementation for this extraction is presented in figure 2.9 [72]. It can be noticed that some parts of the circuits used for extracting LSB and MSB are common. The commonality between the circuits was exploited and full circuit diagram for conversion quaternary digit to binary bits using floating gate MOSFETs is shown in figure 2.10 [72].
Figure 2-9: Floating point potential diagram for conversion of quaternary to LSB output Circuit Diagram for implementation of quaternary to binary logic – LSB using floating gate MOSFETs [72]
2.4.2.2 Binary to Quaternary Convertor using Pass gates

Multi supply source based convertor using pass gates was presented in [73]. The design uses eight transistors for full binary to quaternary conversion. Figure 2.11 gives the circuit diagram of the conversion circuit [73]. Looking closely it can be seen that the design employs...
pass gates for LSB voltage selection and then MSB selects the final voltage level to be passed on as the quaternary equivalent of the two binary bits. Also, figure 2.12 gives the circuit diagram for quaternary to binary circuit design using a similar method [73]. The design was simulated for 0.13μm process technology using SPICE simulator and performed functionally well at 500MHz.

Figure 2-11: Binary to quaternary encoder using pass gate [73]
Figure 2-12: Quaternary to binary encoder (left) and XOR gate (right) using pass gate [73]

2.4.3 MVL Circuits

Last section introduced MVL algebra and conversion circuits. This section gives an overview of some of the work available for MVL components in memory, logic & arithmetic etc.

2.4.3.1 Flash Memories

Multiple-valued logic has been used in DRAM [66, 67], optical [63] and Flash [52, 53, 54, 55, 56, 57] memory designs. DRAM and Flash memories have had the greatest commercial success. This section discusses these two technologies here. [64] and [65] gives the overview of CMOS-related MVL memory technologies.
Flash memory is a multiple-write non-volatile Electrically Erasable Programmable Read Only Memory (EEPROM). Flash memory has long retention time for the programmed value regardless of the electrical power or voltage presence. The memory is arranged in chunks for erasing and programming. The block size could range from 8Kbit to 1Mbit. Memory storage cell is usually a transistor with a floating gate. The floating poly-silicon gate has the capacity to store charge. The amount of charge presence in the floating gate determines the state of a bit. Since the memory cell is a single transistor, very small area of silicon is used to store one bit of data. This makes it a very low cost memory. These features make flash memory very attractive for mobile applications, most smart phones and tablets today use flash memory for storage. Historically the increase in density and reduction in cost for flash memory came from scaling in process technology. In 1990’s Intel started the research to start using MVL for flash. Using MVL for memory would pack more bits per memory cell. In other words each bit would use only a fraction of a transistor. As a result, in 1997 Intel introduced StrataFlash for storing two bits per memory cell. The MVL memory cell was 5% larger and used the same supply voltage. The write and erase performance remained equivalent but the read performance decreased by 20%. The endurance cycles specification dropped by an order but it was in the acceptable range and the change in performance was very well justified by the cost reduction. The success of MVL flash encouraged Intel for plans to scale the designs in future process technologies [34]. This innovation was followed by multiple announcements by companies like Samsung, Hitachi, Mitsubishi and NEC for flash using MVL [57].

The success of above four level or 2 bit memory per cell was such a success that industry started thinking about packing even more bits per cell and store the analog levels in the memory cell and using the digital to analog converters to create these levels before storing in the memory cell. Invox Technology demonstrated that it could use digital to analog converter with NOW flash memory to store three or four bits. The related device could hold 256 levels of analog voltages
per memory cell [58]. Another company, Information Storage Devices also announced analog based MVL flash memory with the capability to store 256 levels per memory cell.

2.4.3.2 DRAM Memories

DRAM memory loses its content when the power is turned off, in other words DRAM is volatile. In a DRAM cell a capacitor and a transistor are used to store the charge. The quantity of the charge determines the voltage level or the bit value stored in a DRAM cell. For a two level cell, the amount of signal charge is about one half the maximum charge that could stored in the cell, but this ratio drops to one sixth for a four level cell [66]. To keep the same signal circuitry, the DRAM cell needs to have three times higher charge capacity when compared to binary or two level implementation. The density and capacity improvements in DRAM chips came from the process technology scaling and improvement in call structures and circuits. This resulted in 50% growth in memory size generation over generation. The result was the first 1GBit DRAM chip in 1995. One company NEC decided to accelerate the growth rate by using MVL in DRAM chips and in 1997 NEC introduced first 4Gbit four-level DRAM. This was the densest DRAM chip at the time [67]. NEC used high dielectric materials for capacitor dielectrics to keep the performance up. The chip was designed using 0.15μm CMOS technology that used 2.0 – 2.0V for power supply. Also, NEC used four level sensing and restoring operations to keep the speed high as well as to reduce the area needed for sense circuits. The chip was capable of 1GB/sec read mode data rate with 64 bit parallel operation.

2.4.4 MVL Arithmetic Designs

This section gives the overview of designs of arithmetic block using MVL. Adder and multiplier blocks have been demonstrated using the MVL designs and higher radix number
systems and algebra [27, 28, 30, 31, 32, 68, 70]. The inherent efficiency of high radix number system makes the MVL designs more efficient when compare to binary based designs.

### 2.4.4.1 MVL Adders

Ternary or quaternary adders designs have been demonstrated using various MVL designs. A three valued full adder was presented in [30]. The design used multiple–valued literal circuits and takes advantage of the resistance characteristics of RTDs to realize a compact gated transfer function. The design used current mode configured MOS transistors. The addition is achieved by summing the current from two or more of these MOS transistors. This architecture uses the redundant balanced number representations for the design. This representation allows addition without carry propagation or carry look ahead. A similar CMOS based redundant binary adder requires 56 transistors, while the implementation in [30] used 13 MOS transistor with one RTD.

Another design based on CMOS current-mode implementation is presented in [70]. This design uses differential logic circuits with dual-rail complementary input. Also, a balanced redundant number representation was employed to give fast implementation. The design was simulated SPICE simulator and was designed using 0.8μm CMOS process technology. Another design with module 7 and residue number system was design using 147 transistors [27]. Also, some of the previous versions of these two adders were presented in [31].

Recent noteworthy design of high performance is presented in [71]. Figure 2.13 gives the block diagram for this quaternary adder. X and Y are the two quaternary inputs and Cin is the carry in. The inputs X and Y get encoded through the encoder and code-generator blocks. Figures 2.14 - 16 [71] gives the block diagram and logic circuits for encoders and code generators. The output of these code generators along with carry-in signal serve as input to the
sum block and carry out block for the adder. Figures 2.17 and Figures 2.18 [71] give the circuits for the sum and carry out blocks.

This quaternary adder employs unique encoding scheme which reduces the requirement for complex hardware. This in turns enable higher performance of the quaternary full adder. For a design using 180nm technology, the adder reportedly consumed 91.25\,\mu\text{W}. Also the circuit designed required a total of 113 transistors. It was claimed that the adder design required less number of transistors and demonstrated higher performance when compared to some of the other designs [81, 83, 84].

![Block diagram quaternary full adder](image)

Figure 2-13: Block diagram quaternary full adder [71].
Figure 2-14: Logic diagram of encoder circuits [71].

Figure 2-15: Block Diagram of code generator for quaternary input X and Y [71].
Figure 2-16: Code generator circuit for quaternary input X and Y [71].
Figure 2-17: Circuit Diagram for sum block for quaternary adder [71].

Figure 2-18: Circuit Diagram for carry block for quaternary full adder [71].

2.4.4.2 MVL Multipliers

Many MVL successful multiplier designs have been demonstrated over the years. This section will review a few of the notable multiplier designs. A very convincing demonstration with success of MVL multipliers is presented in [29], the design used current more MOS circuits. The dual-rail source coupled logic was used and redundant balanced number representation was employed. Using these techniques give a small signal-voltage swing and provide constant driving current. For normalized power dissipation, the performance of this 54 x 54 bit multiplier was 1.4
times faster when compared to corresponding binary implementations. Another multiplier design with 4 x 4bit multiplication capability was presented in [32]. The design used redundant number system for direction generation of partial product. Also, it used a redundant quaternary adder to add the two partial products without carry propagation. The last level converts the redundant numbers to non-redundant number by quaternary carry look-ahead adder. Even more MVL multipliers are reported in [68], [27] and [28]. These designs are quaternary multiplier, modulo-7 multiplier and modulo m multiplier respectively.

As this chapter discussed, there has been significant components of VLSI has been designed using multi-value logic (MVL). Some of the products have even been commercialized. MVL still faces challenges to be integrated into main stream micro processors. Some of the challenges lie in availability of design tools and algorithms that could easily comprehend MVL design and produce the design databases correctly. There has been some research done in this area but significant work need to be done in this area. Also, we need a design environment, process and devices which could provide most of the MVL components in one package. It is not cost efficient to integrate various different technologies and design environments to bring in MVL technology into main stream processor designs.
3  Spatial Wave Switching Field Effect Transistor (SWSFET)

This chapter covers the concept and working of spatial wavefunction switched Field Effect Transistor (SWSFET). SWSFET was invented by Jain et al. [2][3]. Many kinds of FETs have been under study in recent years. These include FETs using carbon nanotubes, graphene nano wire Si and zSiGe FETs, and InGaAs FETs using self-aligned HfO$_2$/TaN gate stacks [85]. In addition Quantum Dot gate FETs with vertical and lateral structures have been investigated [87], [88], [89]. All these devices employ only one channel. Spatial wave switched FET introduced the idea of multiple stacked transport channels. The channels are vertically stacked and the electron wave function can switch from one channel to another with change in gate voltage.

![Schematic cross-section of a transport channel comprising os two InGaAs quantum wells sandwiched between AlInAs barriers in a SWS-FET realized on an InP substrate (left) and Schematic energy band diagram (right) [2].]
Figure 3.1 [2] shows the basic topology of two quantum well SWSFET. The wells are of GaInAs and barrier layers are composed of AlInAs. The energy band diagram of the structure is also shown in figure 3.1. Change in gate voltage locate the carriers in different wells and hence activate the different channels. This action gives this device very unique switching properties. The sections ahead discuss these properties in details. These properties make SWSFET useful in numerous practical applications. We will explore some of these applications in the later chapters.

3.1 SWSFET: Structure and Operation

Figure 3.1 shows the cross sectional view and energy band diagram of a two well spatial wave switched FET (SWSFET). This InGaAs based SWSFET was designed on P type InP substrate and has metal-oxide-semiconductor (MOS) structure. This SWSFET structure presents two wells & two barriers and hence offers two transport channels. Of the two wells the thickness of the upper well W1 is 2 nm while that of lower well W2 is 5 nm. These wells are separated by AlInAs barriers B1 and B2 with respective thickness of 1.5 nm and 100 nm. The well and barriers layers are grown over p-InGaAs layer, which is grown in InP substrate.

The simulations for the SWSFET with the charge density in different channels are presented in figures 3.2 and 3.3 [2]. Figure 3.2 shows the high charge density in lower well at the gate voltage of -3.0V. This well is much thicker than the upper well. As the gate voltage is raised to -2.0V the charge density shifts to upper well, while the density in lower well drops. The absolute gate voltage could be different based on the reference used in the device. The design of the SWSFET needs to provide a distinct gate voltage ranges for which the carriers could be located in upper or lower wells. This feature of the device plays the key role in the manifestation of SWS effect.

The self-consistent solution to Poisson and Schrödinger equations give the two-dimensional charge density and distribution in the coupled quantum well channels [2,3]. The
following equations were used for computing the energy band diagram for the wells. Equation (1) below gives the Poissons equation in terms of electric potential field and charge.

\[ \nabla.(\varepsilon \nabla \phi) = q\left(n_{QM} + n - N_D^+ + N_A^- - p\right) \quad \ldots \ 3.1 \]

Where \( \phi \) is the electrostatic potential,

\( n_{QM} \) is the two dimensional electron gas in the quantum well,

\( n \) is the three dimensional electron concentration,

\( p \) is the three dimensional hole concentration,

\( N_D^+ \) is the ionized donor concentration and

\( N_A^- \) is the ionized acceptor concentration.

Here, to solve equation (1) we need to find the wavefunctions and bound-state energies to determine the two-dimensional charge contributions. The two-dimensional charge contribution is given by Eq (2).

\[ n_{QM} = \sum n \frac{kTm^*}{\pi\hbar^2} \Theta(E_F - E_n) \ln \left[1 + \exp \left(\frac{E_F - E_n}{kT}\right)\right] |\psi_n|^2 \quad \ldots \ 3.2 \]

Where, \( E_F \) is the Fermi Level,

\( E_n \) and \( \psi_n \) are the bound states, and

\( \Theta \) is the Heaviside step function.

We can determine \( E_n \) and \( \psi_n \) are using the Schrödinger equations given below.
\[
\frac{\hbar^2}{2} \nabla \left( \frac{1}{m^*} \nabla \psi_n \right) + (E_n - V)\psi_n = 0 \quad \ldots 3.3
\]

Where, \( V \) can be determined from the conduction band of the structure. This can be calculated by equation 3.1. The solution of these equations determines the energy band diagram for the SWSFET structure. The calculated energy band diagram is presented in figure 3.2 and 3.3 [2]. Fig 3.2 [2] shows the computed solution for gate voltage of -3.0 volts. At this gate voltage lower well has high charge density while the charge density upper well is low. Fig 3.3[2] presents the solution at higher gate voltage of -2.0 volts. In this solution the upper well has higher charge density than the lower well. The charge-densities in upper well, the lower well and total charge densities are plotted in figure 3.4[2]. The solution for 2D electron gas in quantum well also helps compute the \( I_d \) vs. \( V_d \) and other characteristics for the device. Fig 3.5[2] presents the C-V characteristics for such device. It shows a peak corresponding to the peak charge density in lower well. With higher gate voltage the peak charge density in lower well diminishes and so does the related capacitance. This effect can be seen [2] in the figure 3.5.
Figure 3-2: Simulated energy band diagram of a two-quantum-well SWS device showing carriers in the lower well W2 at gate voltage $V_g = -3.0$ V

Figure 3-3: Energy band diagram showing carriers shifted to the upper well W1 when the gate voltage is increased to $V_g = -2.0$ V
Figure 3-4: Channel charge density in each well of lower well W2 labeled ‘‘Last QW’’ & upper well well W1 labeled ‘‘First QW’’ separately and both wells together.

Figure 3-5: Simulated low-frequency C–V characteristic showing a peak corresponding to transfer of electrons from well W2 to well W1
3.2 SWSFET: Two, Three and Four well Structures and Quantum Mechanical Simulations

In the last section we discussed the basic idea, structure and operation of a SWSFET using a two well SWSFET. This section examines three and four well structures in addition to two well. Also we will discuss the related quantum mechanical simulation and the energy band diagrams for such structures. In the later chapter we will see how the three and four well structure can be used for the practical and efficient applications in digital designs.

3.2.1 Two Well SWSFET

Figure 3.6 [2] shows the structure of a typical Spatial Wavefunction Switching (SWS) FET having two Si quantum well W1 and W2. These two wells serve as two channels. The energy band diagram is shown in Figure 3.7 [2]. This is an asymmetric coupled quantum well transport channel FET. As shown in figure 3.9, the electrons are first confined in the lower channel or well W2. As the gate voltage is raise above the threshold (V_{th2}), the electron wavefunction is spatially switched to the upper channel (well W1 – as shown in figure 3.8), resulting in changes in V-I and C-V characteristics. Figures 3.8 – 3.10 [2] show quantum mechanical simulations of a spatial wavefunction switched (SWS) Si-SiGe FET [89] having two coupled wells in the transport channel.
Figure 3-6: SWS-FET having an asymmetric-coupled quantum well channel.

\[
qV_G = -q(V_{TH1} + \Delta V_G)
\]

Figure 3-7: Schematic energy band of a coupled well SWS-FET.
Figure 3-8: Carriers in Si lower well W2

Figure 3-9: More carriers in lower well W2 than upper well W1.
3.2.2 Three Well SWSFET

Here SWSFET using three Ge wells and ZnSe barriers on Si is discusses. The quantum well-barrier configuration is optimized to obtain high contrast switching so that the cross talk between adjacent channels is minimized. Quantum Simulation of a three-coupled well SWS structure demonstrates a superior confinement in three wells as illustrated in Figs. 3.11 - 3.13 [2]. The control of barrier heights as well as confinement of carriers is significantly improved in the Ge quantum well channels. In this three-coupled well structure there is a higher degree of confinement of carriers within a well. When the gate voltage is changed from a $V_g = -4.0V$ to $-3.0V$, the carriers abruptly transfer from Well W3 to the uppermost well W1. Note that the gate voltage can be adjusted by changing: (1) gate insulator thickness, (2) well and barrier material composition and thickness, and (3) strain in the wells.
Figure 3-11: Three Ge well SWSFET

Figure 3-12: Carrier Wavefunction located in lower most well W3
3.2.3 Four Well SWSFET

This section presents four Quantum well Ge-ZnSe and InGaAs-AlInAs SWSFETs. Fig. 3.14 shows a four-well Ge (well)-ZnSe-ZnMgSSe (II-VI barrier) structure. In the quantum simulations, the wavefunction spatial locations can be seen predominantly in one of the four wells (W4 to W1) as a function of gate voltage (-4 to -3 V). The magnitude of threshold variation can be adjusted by device parameters. Figures 3.15 – 3.18 [2] give the results from the quantum simulations with four different gate voltage. As clear from these simulations, the carrier wave function moves in steps from lower most well W4 to upper most well W1 with the rise in gate voltage. Also figure 3.19 [2] shows the charge density plot as a function of gate voltage in the four quantum wells. Also table 3.1 [2] gives the parameter values used for quantum simulations presented in figures 3.15 through 3.19.
Figure 3-14: Four well Ge well ZnSSe barrier based SWSFET.

Figure 3-15: Four-QW SWS wave-function in W4 (Vg= -3.8V)
Figure 3-16: Four-QW SWS wave-function in W3 ($V_g = -3.5V$)

Figure 3-17: Four-QW SWS wave-function in W2 ($V_g = -3.2V$).
Figure 3-18: Four-QW SWS wave-function in W1 (Vg= -3.0)

Figure 3-19: Charge density plot as a function of gate voltage in various quantum wells.
Device structure optimization would enhance the contrast of wavefunction confinement in distinct wells (e.g. W4, W3 etc.). Some of the parameters in table 3.1 can be used to optimize the device parameters as required by the device design.

Table 3-1: Parameters used in the simulation of wavefunctions in Ge quantum wells*.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Thick(um)</th>
<th>$\chi$(eV)</th>
<th>$E_g$(eV)</th>
<th>$m_e$</th>
<th>$m_h$</th>
<th>$\varepsilon_r$</th>
<th>$N_d$</th>
<th>$N_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZnMgSSe</td>
<td>0.0050</td>
<td>0.90</td>
<td>4.50</td>
<td>0.13</td>
<td>0.38</td>
<td>8.0</td>
<td>0.0e00</td>
<td>0.0e00</td>
</tr>
<tr>
<td>Ge</td>
<td>0.0014</td>
<td>4.1</td>
<td>0.67</td>
<td>0.082</td>
<td>0.28</td>
<td>16</td>
<td>0.0e00</td>
<td>1.0e13</td>
</tr>
<tr>
<td>ZnMgSSe Bar</td>
<td>0.0026</td>
<td>3.5</td>
<td>4.50</td>
<td>0.13</td>
<td>0.38</td>
<td>8.0</td>
<td>0.0e00</td>
<td>1.0e15</td>
</tr>
<tr>
<td>Ge</td>
<td>0.0015</td>
<td>4.11**</td>
<td>0.67</td>
<td>0.082</td>
<td>0.28</td>
<td>16</td>
<td>0.0e00</td>
<td>1.0e13</td>
</tr>
<tr>
<td>ZnMgSSe Bar</td>
<td>0.0030</td>
<td>3.5</td>
<td>4.50</td>
<td>0.13</td>
<td>0.38</td>
<td>8.0</td>
<td>0.0e00</td>
<td>1.0e15</td>
</tr>
<tr>
<td>Ge</td>
<td>0.0025</td>
<td>4.12**</td>
<td>0.67</td>
<td>0.082</td>
<td>0.28</td>
<td>16</td>
<td>0.0e00</td>
<td>1.0e13</td>
</tr>
<tr>
<td>ZnMgSse Bar</td>
<td>0.0025</td>
<td>3.5</td>
<td>4.50</td>
<td>0.13</td>
<td>0.38</td>
<td>8.0</td>
<td>0.0e00</td>
<td>1.0e15</td>
</tr>
<tr>
<td>Ge</td>
<td>0.0030</td>
<td>4.13**</td>
<td>0.67</td>
<td>0.082</td>
<td>0.28</td>
<td>16</td>
<td>0.0e00</td>
<td>1.0e13</td>
</tr>
<tr>
<td>ZnMgSse Bar</td>
<td>0.0200</td>
<td>3.5</td>
<td>4.50</td>
<td>0.13</td>
<td>0.38</td>
<td>8.0</td>
<td>0.0e00</td>
<td>1.0e15</td>
</tr>
<tr>
<td>ZnSe</td>
<td>0.1000</td>
<td>4.0</td>
<td>2.67</td>
<td>0.13</td>
<td>0.38</td>
<td>8.0</td>
<td>0.0e00</td>
<td>1.0e15</td>
</tr>
<tr>
<td>SiGe buffer</td>
<td>0.4000</td>
<td>3.9</td>
<td>0.89</td>
<td>0.19</td>
<td>0.49</td>
<td>11.9</td>
<td>0.0e00</td>
<td>1.0e16</td>
</tr>
<tr>
<td>Si* substrate</td>
<td>1.0</td>
<td>4.15</td>
<td>1.12</td>
<td>0.19</td>
<td>0.49</td>
<td>11.9</td>
<td>0.0e00</td>
<td>1.0e16</td>
</tr>
</tbody>
</table>

* ($\chi$ is electron affinity, $E_g$ the bandgap, $m_e$ and $m_h$ are the electron and hole masses, $\varepsilon_r$ the dielectric constant, $N_d$ and $N_a$ are donor and acceptor concentrations).

Next, we examine an InGaAs based four well SWSFET structure with four separate sources connected to the associated transport channels. Figure 3.20 [2] shows the cross-sectional schematic of an InGaAs (well) and AlInGaAs barrier system realized on an InP substrate. Figures 3.21 through 3.23 [2] give the associated quantum mechanical simulations for carrier wavefunction placement at various gate voltages. SWS action can be seen with the movement of carriers from lower wells to the upper most well. The SWSFET parameters could be optimized to obtain altered carrier concentration for these gate voltages.
Figure 3-20: Four Well SWSFET with InGaAs

Figure 3-21: Wavefunctions in lower wells W4 and W3.
Figure 3-22: Wavefunction in upper most well W1.

Figure 3-23: Wavefunction in upper most well W1 (different barrier height of gate insulator).
3.2.4 SWSFET Fabrication Processing and characteristics

This section discusses the fabrication process for SWSFET. Also, the section explores some of the fabricated samples and related characteristics. SWSFETs have been designed and simulated in multiple materials. For this discussion here, first we will take InGaAs-AllnAs well-barrier structure. InGaAs-AllnAs based four well SWSFET structure is presented in figure 3.20 earlier in this chapter.

This SWSFET growth is carried out on $p$-InP wafer. Layers of InGaAs-AllnAs are grown over the InP wafer using the metal organic chemical vapor deposition (MOCVD) techniques. After the well and the barrier layers are in place, selective regrowth of an $n^+$-InGaAs is done to form the source and drain regions. Then the gate area is opened and II-VI gate insulators are grown on it. This growth also uses MOCVD methods. [2] uses ZnSe-ZnS-ZnMgS-ZnS-ZnSe stack of multiple layers for gate. For one of the runs, the gate layers were (1) ZnSe buffer at 505 degree Celsius for 0.5 min (2) ZnS at 333 degree Celsius for 1 min, (3) ZnMgS at 333 degree Celsius for 3 minutes and (4) ZnSe at 333 degree Celsius for 2 minutes.

3.2.4.1 SWSFET Characteristics

SWSFET structure gives distinct C-V characteristics with peaks for the gate voltage regions where the carriers are concentrated in different quantum wells. Figure-3.24 [2] gives the simulated CV characteristics for a three well SWS structure. Figure-3.25 [6] presents the measured C-V characteristics on a two well InGaAs-AllnAs SWS sample. Also figure-3.26 shows the C-V measurements for a four well InGaAs SWS MOS sample under inversion.
Figure 3-24: SWS-MOS capacitor-voltage plot (theoretical simulation model).

Figure 3-25: Capacitance variation due to gate voltage in fabricated 2-well InGaAs SWS-FET.
Figure 3-26: C-V plot for a 4-well InGaAs SWS MOS sample 2056 under inversion.

This chapter reviewed multiple SWSFET designs and discussed the quantum mechanical simulations. SWS action could be seen in simulation of two well, three well and four well designs. When the wells are connected to different sources or drains, a unique channel switching FET is the result. Also the fabrication of SWSFET and C-V characteristics were discussed, these measurements confirmed the SWS action and carrier wavefunction movement as predicted by the quantum mechanical simulations. In the chapters ahead, we will discuss the applications of SWSFET as valuable device for enabling multiple-value logic based memories, logic and mixed signal design components.
4 Novel Quaternary Logic

In chapter 3, we explored SWSFET designs, quantum mechanical simulation along with fabrication process and characteristics. As discussed, SWSFET designs offer multiple channels this feature opens a new paradigm for design applications. Many innovative designs could potentially be implemented for electronic circuits using SWSFET. This chapter discusses the novel logic that could be implemented using the multiple channels offered by SWSFETs. The motivation of for the novel logic was to present a logic which could take advantage of multiple channels offered by SWSFETs. The boundary conditions for the novel logic were that it should be compatible with existing CMOS binary logic and could be implemented with reasonable number of channels available in SWSFETs. As will be clear in the following sections and chapters, the maximum number of channels needed in SWSFET for this logic is four. When the SWSFET threshold voltages are skewed, the logic could also be implemented with three-channel SWSFETs. The total number of transistors used to implement a given logic cell would be different in the two cases above. The following sections in this chapter describe the logic levels, contrast the new logic & algebra with binary logic and truth tables for some basic logic gates.

4.1 Novel Quaternary Logic

The quaternary logic by definition is a four level logic as compared to binary logic which a two level logic. The challenge is to define meaningful logic and arithmetic operations which could be practically implemented. Figure 4.1 shows the conventional binary logic levels along with the new quaternary logic levels [6][7]. The quaternary logic is designed with two bit binary equivalence in mind. Which means four quaternary levels 0,1,2 and 3 essentially represent 00, 01, 10 and 11 states for two binary bits. This is also shown in the figure 4.1 below. Also figure 4.1 below gives inversion or logic NOT operator for binary and quaternary logics. Note that quaternary logic NOT operations on quaternary and corresponding two bit binary bits are in
agreement with each other. Therefore at any node in an electronic circuit a quaternary logic could be converted to binary levels and vice-versa, given the availability of right number of binary bits.

![Diagram](image)

**Figure 4-1**: Binary logic verses Novel Quaternary logic with two inversions.

The rest of this chapter elaborates on the definition of logic and arithmetic operations using the above defined quaternary logic.

### 4.2 Logical NOT

Quaternary logical NOT is shown in figure 4.1 above. The truth table for the NOT gate is presented in table 4.1 below [7]. Note the inversion of two binary bits in the NOT operation.

<table>
<thead>
<tr>
<th>S/ N</th>
<th>A(A₁A₂)</th>
<th>NOT A = Y(Y₁Y₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0(00)</td>
<td>3(11)</td>
</tr>
<tr>
<td>2</td>
<td>1(01)</td>
<td>2(10)</td>
</tr>
<tr>
<td>3</td>
<td>2(10)</td>
<td>1(01)</td>
</tr>
<tr>
<td>4</td>
<td>3(11)</td>
<td>0(00)</td>
</tr>
</tbody>
</table>

The quaternary logical operation is intuitive in this case, but as will be seen in the following sections it may not be same even for some other basic logic gates.
4.3 Logical OR

Table 4.2 gives the truth table for quaternary logical OR operation [7]. While the output for logical NOT operation was intuitive, it can be seen by examining the table 4.2 below that it is not the case for logical OR operation.

Table 4-2: Truth table implementing OR Gate

<table>
<thead>
<tr>
<th>S/ N</th>
<th>A(A₁A₂)</th>
<th>B(B₁B₂)</th>
<th>A OR B = Y(Y₁Y₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0(00)</td>
<td>0(00)</td>
<td>0(00)</td>
</tr>
<tr>
<td>2</td>
<td>0(00)</td>
<td>1(01)</td>
<td>1(01)</td>
</tr>
<tr>
<td>3</td>
<td>0(00)</td>
<td>2(10)</td>
<td>2(10)</td>
</tr>
<tr>
<td>4</td>
<td>0(00)</td>
<td>3(11)</td>
<td>3(11)</td>
</tr>
<tr>
<td>5</td>
<td>1(01)</td>
<td>0(00)</td>
<td>1(01)</td>
</tr>
<tr>
<td>6</td>
<td>1(01)</td>
<td>1(01)</td>
<td>1(01)</td>
</tr>
<tr>
<td>7</td>
<td>1(01)</td>
<td>2(10)</td>
<td>3(11)</td>
</tr>
<tr>
<td>8</td>
<td>1(01)</td>
<td>3(11)</td>
<td>3(11)</td>
</tr>
<tr>
<td>9</td>
<td>2(10)</td>
<td>0(00)</td>
<td>2(10)</td>
</tr>
<tr>
<td>10</td>
<td>2(10)</td>
<td>1(01)</td>
<td>3(11)</td>
</tr>
<tr>
<td>11</td>
<td>2(10)</td>
<td>2(10)</td>
<td>2(10)</td>
</tr>
<tr>
<td>12</td>
<td>2(10)</td>
<td>3(11)</td>
<td>3(11)</td>
</tr>
<tr>
<td>13</td>
<td>3(11)</td>
<td>0(00)</td>
<td>3(11)</td>
</tr>
<tr>
<td>14</td>
<td>3(11)</td>
<td>1(01)</td>
<td>3(11)</td>
</tr>
<tr>
<td>15</td>
<td>3(11)</td>
<td>2(10)</td>
<td>3(11)</td>
</tr>
<tr>
<td>16</td>
<td>3(11)</td>
<td>3(11)</td>
<td>3(11)</td>
</tr>
</tbody>
</table>

We will also see in the later chapters that the quaternary logic along with enhancement capabilities of SWSFET offer savings of up to 75% in transistor count for implementing an equivalent of binary two bit OR circuit as compared to CMOS implementation.
4.4 Logical XOR

The truth table for quaternary logical XOR operation is presented in table 4.3 [7]. The quaternary output of the logical XOR is not intuitive but the binary equivalence can be clearly seen in the last column of table 4.3.

<table>
<thead>
<tr>
<th>S/N</th>
<th>A(A₁,A₂)</th>
<th>B(B₁,B₂)</th>
<th>A XOR B = Y(Y₁,Y₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0(00)</td>
<td>0(00)</td>
<td>0(00)</td>
</tr>
<tr>
<td>2</td>
<td>0(00)</td>
<td>1(01)</td>
<td>1(01)</td>
</tr>
<tr>
<td>3</td>
<td>0(00)</td>
<td>2(10)</td>
<td>2(10)</td>
</tr>
<tr>
<td>4</td>
<td>0(00)</td>
<td>3(11)</td>
<td>3(11)</td>
</tr>
<tr>
<td>5</td>
<td>1(01)</td>
<td>0(00)</td>
<td>1(01)</td>
</tr>
<tr>
<td>6</td>
<td>1(01)</td>
<td>1(01)</td>
<td>0(00)</td>
</tr>
<tr>
<td>7</td>
<td>1(01)</td>
<td>2(10)</td>
<td>3(11)</td>
</tr>
<tr>
<td>8</td>
<td>1(01)</td>
<td>3(11)</td>
<td>2(10)</td>
</tr>
<tr>
<td>9</td>
<td>2(10)</td>
<td>0(00)</td>
<td>2(10)</td>
</tr>
<tr>
<td>10</td>
<td>2(10)</td>
<td>1(01)</td>
<td>3(11)</td>
</tr>
<tr>
<td>11</td>
<td>2(10)</td>
<td>2(10)</td>
<td>0(00)</td>
</tr>
<tr>
<td>12</td>
<td>2(10)</td>
<td>3(11)</td>
<td>1(01)</td>
</tr>
<tr>
<td>13</td>
<td>3(11)</td>
<td>0(00)</td>
<td>3(11)</td>
</tr>
<tr>
<td>14</td>
<td>3(11)</td>
<td>1(01)</td>
<td>2(10)</td>
</tr>
<tr>
<td>15</td>
<td>3(11)</td>
<td>2(10)</td>
<td>1(01)</td>
</tr>
<tr>
<td>16</td>
<td>3(11)</td>
<td>3(11)</td>
<td>0(00)</td>
</tr>
</tbody>
</table>

4.5 Logical AND

Quaternary logical AND operation truth table is presented in table 4.4 [7]. The output of this operation is relatively easily correlated with the input values. For input level on A equal to 0,
the output is always 0. For input level on A equal to 1, the output level are 1 in case B is 1 or 3, else it is 0. Similar pattern exists for A equal to 2. For A equal to 3, the output equals B. The implementation of quaternary AND gate using SWSFET is shown in later chapters. The simplicity of the output makes it relatively easier to implement this case using SWSFETs.

Table 4-4: Truth table implementing AND gate

<table>
<thead>
<tr>
<th>A(A_1A_2)</th>
<th>B(B_1B_2)</th>
<th>A AND B =</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0(00)</td>
<td>0(00)</td>
</tr>
<tr>
<td>2</td>
<td>0(00)</td>
<td>1(01)</td>
</tr>
<tr>
<td>3</td>
<td>0(00)</td>
<td>2(10)</td>
</tr>
<tr>
<td>4</td>
<td>0(00)</td>
<td>3(11)</td>
</tr>
<tr>
<td>5</td>
<td>1(01)</td>
<td>0(00)</td>
</tr>
<tr>
<td>6</td>
<td>1(01)</td>
<td>1(01)</td>
</tr>
<tr>
<td>7</td>
<td>1(01)</td>
<td>2(10)</td>
</tr>
<tr>
<td>8</td>
<td>1(01)</td>
<td>3(11)</td>
</tr>
<tr>
<td>9</td>
<td>2(10)</td>
<td>0(00)</td>
</tr>
<tr>
<td>10</td>
<td>2(10)</td>
<td>1(01)</td>
</tr>
<tr>
<td>11</td>
<td>2(10)</td>
<td>2(10)</td>
</tr>
<tr>
<td>12</td>
<td>2(10)</td>
<td>3(11)</td>
</tr>
<tr>
<td>13</td>
<td>3(11)</td>
<td>0(00)</td>
</tr>
<tr>
<td>14</td>
<td>3(11)</td>
<td>1(01)</td>
</tr>
<tr>
<td>15</td>
<td>3(11)</td>
<td>2(10)</td>
</tr>
<tr>
<td>16</td>
<td>3(11)</td>
<td>3(11)</td>
</tr>
</tbody>
</table>

4.6 Arithmetic Full Adder

The previous sections of this chapter presented the definition of logical operations for the novel quaternary logic. This section presented some basic arithmetic operations using the quaternary logic. A later chapter also shows the implementation of the quaternary arithmetic
blocks using SWSFETs. Table 4.5 below presents the truth table for full adder for the quaternary logic [7]. As in the case of logical operations, the output of the full adder shows equivalence to the binary bit for the same operation. Though not presented in this chapter, the truth table for half adder and multiplier can also be generated in the same manner.

<table>
<thead>
<tr>
<th>Carry-in</th>
<th>A</th>
<th>A'</th>
<th>B</th>
<th>Sum</th>
<th>Carry-Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0(00)</td>
<td>0(00)</td>
<td>0(00)</td>
<td>0(00)</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0(00)</td>
<td>0(00)</td>
<td>1(01)</td>
<td>1(01)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0(00)</td>
<td>0(00)</td>
<td>2(10)</td>
<td>2(10)</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0(00)</td>
<td>0(00)</td>
<td>3(11)</td>
<td>3(11)</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1(01)</td>
<td>1(01)</td>
<td>0(00)</td>
<td>1(01)</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1(01)</td>
<td>1(01)</td>
<td>1(01)</td>
<td>2(10)</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1(01)</td>
<td>1(01)</td>
<td>2(10)</td>
<td>3(11)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1(01)</td>
<td>1(01)</td>
<td>3(11)</td>
<td>0(00)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>2(10)</td>
<td>2(10)</td>
<td>0(00)</td>
<td>2(10)</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>2(10)</td>
<td>2(10)</td>
<td>1(01)</td>
<td>3(11)</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>2(10)</td>
<td>2(10)</td>
<td>2(10)</td>
<td>0(00)</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>2(10)</td>
<td>2(10)</td>
<td>3(11)</td>
<td>1(01)</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>3(11)</td>
<td>3(11)</td>
<td>0(00)</td>
<td>3(11)</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>3(11)</td>
<td>3(11)</td>
<td>1(01)</td>
<td>0(00)</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>3(11)</td>
<td>3(11)</td>
<td>2(10)</td>
<td>1(01)</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>3(11)</td>
<td>3(11)</td>
<td>3(11)</td>
<td>2(10)</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>0(00)</td>
<td>1(01)</td>
<td>0(00)</td>
<td>1(01)</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0(00)</td>
<td>1(01)</td>
<td>1(01)</td>
<td>2(10)</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>0(00)</td>
<td>1(01)</td>
<td>2(10)</td>
<td>3(11)</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0(00)</td>
<td>1(01)</td>
<td>3(11)</td>
<td>0(00)</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1(01)</td>
<td>2(10)</td>
<td>0(00)</td>
<td>2(10)</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1(01)</td>
<td>2(10)</td>
<td>1(01)</td>
<td>3(11)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>1(01)</td>
<td>2(10)</td>
<td>2(10)</td>
<td>0(00)</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1(01)</td>
<td>2(10)</td>
<td>3(11)</td>
<td>1(01)</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>2(10)</td>
<td>3(11)</td>
<td>0(00)</td>
<td>3(11)</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>2(10)</td>
<td>3(11)</td>
<td>1(01)</td>
<td>0(00)</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>2(10)</td>
<td>3(11)</td>
<td>2(10)</td>
<td>1(01)</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>2(10)</td>
<td>3(11)</td>
<td>3(11)</td>
<td>2(10)</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>3(11)</td>
<td>0(00)</td>
<td>0(00)</td>
<td>0(00)</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>3(11)</td>
<td>0(00)</td>
<td>1(01)</td>
<td>1(01)</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>3(11)</td>
<td>0(00)</td>
<td>2(10)</td>
<td>2(10)</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>3(11)</td>
<td>0(00)</td>
<td>3(11)</td>
<td>3(11)</td>
</tr>
</tbody>
</table>

In later chapters the implementation of quaternary full adder is presented using SWSFETs. The implementation is based on the truth table in table 4.5 above.

### 4.7 Logic Transition and Power Consumption

Quaternary logic offers less transition density for logical operations. The lower the transition density, lower the dynamic power consumption in a digital logic circuit. Comparing between binary and quaternary OR gates, the output transition probability for a binary two input OR gate is 25% with equal weights to two inputs. For quaternary logic the corresponding transitions density is 18.5%. The quaternary logic circuit offers over 25% lower transition density and hence about 25% lower dynamic-power in this case. Comparison of binary and quaternary AND also gives the same numbers. In the case of inverter quaternary logic has higher transition density than the binary case.
Proposed Quaternary logic

Total possibilities for set of inputs from current to next = 16 x 16

Number of transitions from:

State 3 = 7
State 2 = 13
State 1 = 13
State 0 = 15

Transition density = (number of transitions) / Total Possibilities

= (7 + 13 + 13 + 15) / 16x16

= 0.185 = 18.5%

Binary logic

Total possibilities for set of inputs from current to next = 4 x 4

Number of transitions from:

State 1 = 3
State 0 = 1

Transition density = (number of transitions) / Total Possibilities

= (3+1) / 4x4

= 0.25 = 25%

The calculation for transition densities in case of AND gate is presented above. In addition, high radix algebra offers efficient implementation of an algebraic calculation. This makes quaternary algebra a good candidate for future processor designs in VLSI.
5 Quaternary Static Random Access Memory (SRAM) design using SWSFET

This chapter discusses first of the circuit application using SWSFETs and quaternary logic presented in this thesis. Performance microprocessors are pushing the limit on number of transistors used along with the scaling down the transistor sizes. Majority of the die area is consumed by the cache on the die [8][106]. Higher and higher cache is needed to enhance the performance but larger die sizes present technical and cost challenges. This chapter introduces a multi-bit SRAM cell using SWSFET. Spatial Wavefunction Switched Field-Effect Transistors (SWSFETs) introduced by Jain et al. was discussed in chapter 3. The motivation here is to take advantage of stacked up multiple bits on a single SRAM cell without multiplying the transistor count. In the following sections we present the quaternary SRAM designs employing the use of SWSFET, simulations of the proposed designs and comparisons with binary logic CMOS based implementation. Also the advantages over the binary implementations are discussed.

5.1 SRAM Cell and Memory System

SWSFETs offer multiple applications related to electronic circuits and networking. Here we discuss the application for multi-bit SRAM memory. SWSFETs offer multiple channels as compared to a single channel in CMOS. SWSFET can be designed such that multiple channels in SWSFET could be activated ‘one hot’ or in pairs. This section discusses the four channel SWSFET design with one hot channel activation [6]. In the design, the carriers are mostly concentrated in W1, W2, W3 or W4 between a set of gate thresholds. For example, in well W3 when gate voltage $V_g$ is at $V_{th-31}$ the distribution of carriers places half the carriers in W3. As the gate voltage is increased over $V_{th-32}$ most carriers move into W3. Thus channel W3 is expected to be ‘one hot’ between $V_{th-31}$ and $V_{th-32}$. With the further increase in gate voltage above the threshold $V_{th-21}$ half the carriers move into W2 and then above $V_{th-22}$ most carriers are located in W2 and so on. Similarly a
set of threshold voltages are assigned to all the wells. For maximum noise immunity and performance the sets of thresholds for one channel have minimum overlap with another channel. Several architectures were investigated for SWSFET based memory cells. In the following sections, we are discussing 2-bit SRAM cell using a four channel SWSFET.

5.1.1 Logic and Cell Architecture

Quaternary logic levels and inverter transfer function presented in Fig. 5-2 are developed. Figures 5.3 and 5.4 show the conversion circuits developed to convert binary to quaternary logic and vice versa.

Figure 5-1: Binary Inversion logic for SRAM using CMOS.

Figure 5-2: Quad Inversion logic for SRAM using SWSFET
Conventional CMOS SRAM cell architecture is shown in Fig. 5.5. Fig. 5.6 shows a 2-bit SRAM cell using two 4-channel SWSFETs. SWSFET based design employs back-to-back connected invertors to form a single memory cell. Each SWSFET acts as quaternary inverter by itself. The output of the inverter is fed to the second inverter to generate the original level and thus to create a quaternary buffer. The buffer output is fed back to the input and the memory cell is created.
Figure 5-5: Two bit SRAM using CMOS devices with binary logic.

Figure 5-6: Two bit SRAM structure using SWSFET devices with quaternary logic.
5.1.2 Simulations

BSIM equivalent models were used for the SWSFET channels. From the simulations presented earlier in chapter 3 [2][3][9], it was demonstrated that carrier mobility increases from lower to upper channels. This was taken into account while modeling the channels. The simulation results, using Advanced Design System (ADS) software (BSIM 4.6 Version) for 25 nm channel length SWSFETs, are shown in Fig. 5.7 and Fig 5.8. Fig. 5.7 gives the input waveform to SWSFET SRAM cell and fig 5.8 gives the waveform at the inverting node in the cell. In the simulation SWS-FET channels are modeled by four conventional transistors channels (each having a different threshold voltage per SWS characteristics). That is, only one channel is turned ON and other transport channels are OFF. The levels show the agreement with quaternary inversion logic as presented in figure 5.2.

![Graph](image_url)  
Figure 5-7: Input voltage vs. time waveform.
Figure 5-8: Output vs. time for 25nm SWSFET based cell.

The higher mobility in upper channels compared to the lower channels introduces the asymmetry in the SWS inverter performance. The falling edges are slower than the rising edge. The fastest transition is noted from logic 2 to 3 while the slowest transition is 3 to 0. The maximum speed of the writes is limited by 3 to 0 transitions. Adaptive hardware algorithms to look ahead the write data and optimize the write performance may be helpful for performance enhancement.

Table 5.1 [6] [7] compares the SWSFET based two bit SRAM cell with CMOS based SRAM cell. SWSFET uses 75% fewer number of devices. For same feature size this translates to 75% less die area. Also metal interconnects carry two bit data, thus the needed metal density for data lanes is also reduced by a factor of 2.
This chapter presented one of the applications of the SWSFET in circuit design applications. This application in multi valued logic memory is an important one. As we noticed in the chapter 2, multi valued logic memory is one of the few applications of MVL which got commercialized. In this case also, the potential saving are very significant as SRAM uses over 50% of die area in performance microprocessor and SWSFET based technology could potentially save 75% of that area. The next chapters of this thesis will explore the SWSFET use for logic and mixed signal applications.

<table>
<thead>
<tr>
<th></th>
<th>CMOS</th>
<th>SWSFETs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device count per bits</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>R/W bits per operation on 8 wide</td>
<td>8 bits</td>
<td>16 bits</td>
</tr>
<tr>
<td>Metal routing needed for data access</td>
<td>2X</td>
<td>1X</td>
</tr>
</tbody>
</table>
6 Quaternary Logic and Arithmatic designs using SWSFET

After the discussion on the simple designs of SWSFET based quaternary SRAM cells in the last chapter, this chapter explores the designs of basic logic and arithmetic building blocks in quaternary logic [7]. The SWSFET is used to implement these blocks. Also the comparison between the quaternary implementation and CMOS based binary implementation is discussed. In addition, simulations proving the functionality of these designs are also presented.

6.1 Quaternary Logic

Figure 6.1 and Figure 6.2 recaptures the logic levels for the proposed quaternary levels with the binary equivalents. These logic levels will be used in this chapter for the design all logic and arithmetic blocks.

![Binary logic levels with two Inversions](image1)

Figure 6-1 : Binary logic levels with two Inversions

![Quaternary logic levels with two inversions](image2)

Figure 6-2: Quaternary logic levels with two inversions
6.2 Logic Cells

This section presents quaternary logic building blocks along with implemented truth tables for the logic levels presented above.

6.2.1 NOT Gate

The NOT gate implementations with SWSFET and CMOS are shown in figures 6.3 and figure 6.4. The related truth table is presented in table 6.1. [7] Two bit NOT gate implementation with CMOS binary logic requires four transistors as compared to one SWSFET using quaternary logic.

![Figure 6-3: Quaternary NOT gate implemented with SWSFET](image)

![Figure 6-4: Two NOT gates implemented with binary CMOS logic.](image)
### Table 6-1: Truth table implementing NOT Gate.

<table>
<thead>
<tr>
<th>#</th>
<th>A (A₁, A₂)</th>
<th>Y (Y₁, Y₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 (00)</td>
<td>3 (11)</td>
</tr>
<tr>
<td>2</td>
<td>1 (01)</td>
<td>2 (10)</td>
</tr>
<tr>
<td>3</td>
<td>2 (10)</td>
<td>1 (01)</td>
</tr>
<tr>
<td>4</td>
<td>3 (11)</td>
<td>0 (00)</td>
</tr>
</tbody>
</table>

### 6.2.2 OR Gate

Two bit OR gate implementations using SWSFET with quaternary logic and CMOS with binary logic are presented in figure 6.5 and figure 6.6. The truth table for the implemented OR-gates is presented in table 6.2 [7]. Two bit OR gate implementation in CMOS based binary logic requires twelve transistors as compared to three transistors required in SWSFET based implementation with quaternary logic.

![Quaternary OR gate implemented using SWSFET](image_url)

Figure 6-5: Quaternary OR gate implemented using SWSFET
Figure 6-6: Two binary OR gates using CMOS.

Table 6-2: Truth table implementing OR Gate.

<table>
<thead>
<tr>
<th>S/N</th>
<th>A(A₁A₂)</th>
<th>B(B₁B₂)</th>
<th>A OR B = Y(Y₁Y₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>2</td>
<td>00</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>3</td>
<td>00</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>00</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>01</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>6</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>7</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>01</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>11</td>
<td>00</td>
<td>11</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>16</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>
6.2.3 AND Gate

The AND gate implementations using SWSFET with quaternary logic and CMOS with binary logic are shown in figure 6.7 and figure 6.8. The related truth table is presented in table 6.3 [7]. Two bit AND gate implementation in CMOS based binary logic requires twelve transistors as compared to three transistors required SWSFET based implementation with quaternary logic.

Figure 6-7: Quaternary AND gate implemented using SWSFET

Figure 6-8: Two binary AND gates using CMOS.
Table 6-3: Truth table implementing AND gate

<table>
<thead>
<tr>
<th></th>
<th>( A(A_1A_2) )</th>
<th>( B(B_1B_2) )</th>
<th>( A \text{ AND } B = Y(Y_1Y_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>2</td>
<td>00</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>3</td>
<td>00</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>4</td>
<td>00</td>
<td>11</td>
<td>00</td>
</tr>
<tr>
<td>5</td>
<td>01</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>6</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>7</td>
<td>01</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>8</td>
<td>01</td>
<td>11</td>
<td>01</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>11</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

6.2.4 Full Adder

Arithmetic blocks are important and integral part of the computing processors. Here quaternary full adder using SWSFET is introduced. Figures 6.9 and 6.10 show full adder implanted using SWSFET with quaternary logic and CMOS logic with binary logic respectively. The truth table for the implemented full adder is presented in table 6.4 [7]. A simple full adder implementation in CMOS with binary logic for two bits requires eighty transistors as compared to eleven required for the adder using SWSFET with quaternary logic here. Also the longest delay path in SWSFET implementation is three transistors as while in CMOS based implementation it is five gates as shown clear from figure 6.10.
Table 6-4: Truth table implementing two bit Full adder.

<table>
<thead>
<tr>
<th>Carry-in</th>
<th>A</th>
<th>A’</th>
<th>B</th>
<th>Sum</th>
<th>Carry-Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>00</td>
<td>00</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>00</td>
<td>00</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>01</td>
<td>01</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>01</td>
<td>01</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>01</td>
<td>01</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>10</td>
<td>11</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>00</td>
<td>01</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>00</td>
<td>01</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>00</td>
<td>01</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>01</td>
<td>10</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>01</td>
<td>10</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>01</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>01</td>
<td>10</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>11</td>
<td>00</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>11</td>
<td>00</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>11</td>
<td>00</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>11</td>
<td>00</td>
<td>11</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 6-9: Two binary full adders implemented using CMOS

Figure 6-10: Quaternary full adder implemented using SWSFET
6.2.5 Quaternary Latch

All the digital logic could be implemented using NOT, AND & OR gates. An additional building block for digital processors is a latch or a flip flop. A quaternary latch implementation using SWSFET is shown in figure 6.11. The related truth table is presented in table 6.5 [7]. A similar two bit latch implementation for two bits using CMOS binary logic would require about twelve transistors as compared to four required using SWSFET with quaternary logic.

Figure 6-11: Quaternary Latch implanted using SWSFET.

<table>
<thead>
<tr>
<th>#</th>
<th>Clock</th>
<th>D-Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>D-Out</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Data-In</td>
</tr>
</tbody>
</table>

Table 6-5 : Truth table for a latch
6.2.6 SRAM Cell

In the previous chapter design of SRAM cell was discussed. Here the design is recaptured for comparison perspectives. SRAM is an important part of microprocessors today with over 50% die area assigned to SRAM for caches. Figure 6.12 shows the SRAM cell implementations using SWSFET with quaternary logic. Figure 6.13 gives the equivalent CMOS implementation with binary logic. Two bit SRAM cell implementation in CMOS requires eight transistors as compared to two transistors required using SWSFET [6] [7].

![Figure 6-12: Quaternary SWSFET based SRAM cell.](image-url)
This section discussed the basic building blocks of a digital micro processor. Also some comparison between the quaternary and binary implementation is presented. Table 6.6 [7] gives the summary of transistor count comparison for implementations using SWSFET vs. CMOS for digital building blocks.

<table>
<thead>
<tr>
<th>S/N</th>
<th>Cell</th>
<th>CMOS Count</th>
<th>SWSFET Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NOT</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>AND</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>OR</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Full Adder</td>
<td>80</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>SRAM</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Latch</td>
<td>16</td>
<td>4</td>
</tr>
</tbody>
</table>

### 6.2.7 Integration with Binary logic

It is important for a MVL design to co-exist with binary logic on the same die. This is especially important if it is decided to take advantage of high value quaternary designs only. For example a chip could use quaternary SRAM while using binary based designs for the rest of the
logic on the die. Figure 6.14 and figure 6.15 give binary to quaternary and quaternary to binary conversion circuits respectively[6, 7]. These convertor designs could be employed with the SWSFET based logic blocks discussed in this chapter.

Figure 6-14: Binary logic to quaternary logic conversion circuit
6.3 Simulations

The SWSFET based basic quaternary gates (AND, OR & NOT) were simulated using the BSIM equivalent channel models. Figures 6.16 and 6.17 [7] give the simulation results for quaternary AND gate, OR gate and NOT gate. These results are in agreement with corresponding truth tables presented in the previous section. Figure 6.18 and Figure 6.19 [6][7] give the simulation results for quaternary inverter. The simulations were carried out using Cadence Spectre simulation tool.
Figure 6-16: SWSFET based two input quaternary AND gate

Figure 6-17: SWSFET based two input quaternary OR gate.
Figure 6-18: Simulation results for SWSFET based quaternary inverter using BSIM equivalent channel models.

Figure 6-19: Simulation results for SWSFET based quaternary inverter using BSIM equivalent channel models.
The basic logic gates, sequential elements and arithmetic blocks are the building blocks that cover most of the VLSI designs. The goal of the work here has been to design MVL based blocks to deliver major design components for a VLSI chip. It is understood that some of the mixed signal and analog blocks may have challenges for conversion to SWSFET based designs. The next chapter explores some mixed signal designs. The rest of the analog and mixed signal designs are proposed for future work.
7 Mixed Signal Circuit using MVL

The previous chapter covered the design of digital infrastructure for VLSI in quaternary MVL space. There are many components on a VLSI die that uses analog or mixed signal designs. These components cover a small area on a typical microprocessor but nonetheless are very important part of a microprocessor. This chapter touches on some of the designs in mixed signal architectures space. Specifically this chapter discusses digital to analog converter architecture using SWSFET and quaternary algebra. The simulation results are presented for one of the designs.

7.1 Digital to Analog Converters using SWSFET

This section introduces the application of SWSFET for mixed signal applications. We present switch-cap architecture based fast digital to analog converters using SWSFETs. These architectures take advantage of SWSFETs unique feature of controllable multiple vertically-stacked-channels. Figures 7.1, 7.2 and 7.3 [10] [11] show the schematics for the design of two bit, four bit and eight bit convertors. All three designs convert the incoming digital signal to analog output within one clock cycle. Two bit DAC runs continuously, while the four-bit and eight-bit DAC uses two non overlapping clocks \( \phi_1 \) and \( \phi_2 \). The architecture of the design gives the DAC very low latency. Also the digital signal traverses only one SWSFET, this makes the design very fast, limited by the bandwidth \( f_T \) of the SWSFET and cap size. These converters would be an attractive option for application with low latency and high speed applications. Figure 7.4 [10] gives the simulation results for four-bit digital to analog converter presented in figure 7.2. The design was simulated using Cadence Spectre Simulator tool with SWSFET channels modeled as BSIM equivalent channels. Two Quaternary signals were used as inputs to the digital to analog convertor. The input signals are labeled as LSB and MSB waveforms. As clear from figure 7.4,
each of these inputs is a four level quaternary waveform. The waveform labeled as “output” represents the analog output of the converter.

Figure 7-1: SWSFET switch cap based 2-bit digital to analog converter.

Figure 7-2: SWSFET switch cap based 4-bit digital to analog converter
Figure 7-3: Fast 8 bit Digital to Analog Converter (DAC) design using SWSFET and switch capacitor architecture.
Figure 7-4: Simulation results for quaternary algebra and SWSFET based 4-bit (two qudit) digital to analog converter architecture presented in figure 7-2.

SWSFET based digital to analog converter (DAC) was designed and simulated. It is understood that a DAC is only one mixed signal design components, but the goal of this chapter to introduce SWSFET to mixed signal space. Many such applications and designs can be explored and are encouraged to pursued as future work.
8 Conclusion

8.1 Static Random Access Memory, logic, arithmetic and mixed signal cells

In this dissertation, SWSFET based static random access memory (SRAM), logic blocks, arithmetic block and digital to analog convertors have been investigated. The SWSFET designs, quantum simulations and characteristics are presented. SWSFET offers some unique characteristics which are not offered by any other FET so far. A novel quaternary MVL algebra and logic was proposed to take advantage of SWSFET features. Using this novel algebra with SWSFET, the designs of quaternary SRAM, quaternary logic gates, quaternary adders, quaternary sequential latches and some mixed signal blocks are presented. Also, simulations for many of these designs were carried out and the results are presented.

The work in this thesis lays a basic ecosystem of building block components for designing a VLSI chip using SWSFET and quaternary logic. Since all possible design components cannot be explored, the architectures and designs of quaternary to binary and binary to quaternary convertors is given. This enables the quaternary and binary designs to co-exist on the same die. One such scenario may be to use quaternary SRAM for cache while keeping the rest of the design in binary. This could be in fact a very practical application, since whole quaternary tool suite is required to implement all the blocks in quaternary MVL on a die. Caches on the other hand consume majority of the die space in performance-microprocessors but have very repetitive design.

The SWSFET based building blocks presented cover wide area of the digital logic. Still, there are many challenges that need to be addressed to take the technology move further. The next section makes some suggestions about the future work that may be helpful in progress the quaternary based SWSFET technology further.
8.2 Suggestions for Future Work

Enabling any technology has its challenges at every step. SWSFET also has its challenges at every level. This section makes some suggestion for future work to resolve those challenges.

1-D quantum mechanical models are used for SWSFET simulations presented in this thesis. The 2-D and 3-D modeling of SWSFET could help uncover and fix any device issue if exists. Fabrication of SWSFET also has its own challenges. Specially connecting multiple drains or sources to a few nanometer thick channels presents practical challenges. This challenge needs further work to develop such processes which could reliably deliver separate sources feeding into the vertically stacked transport channels. Also the exact I-V characteristics could be measured with the physical device with the availability of four channels/sources SWSFET. The device parameters could be further tuned to enhance the contrast of carrier densities between the quantum wells. The high contrast ratio is important to improve the performance of the quaternary logic gates and SWSFET based designs in general.
9 References


[52] G. Atwood et al., “Inter StrataFlash memory technology overview”, 1999 Intel Corporation


[91] Yasushi YUMINAKA and Masaaki OKUI, "Efficient Data Transmission using Multiple-Valued Pulse-Position Modulation" 2012 IEEE 42nd International Symposium on Multiple-Valued Logic


