School of Electrical and Computing

Department of Electrical and Computer Engineering

Efficient Methods for Synthesis of Multi-Valued Logic

Adib Kabir Chowdhury

This thesis is presented for the Degree of

Master of Philosophy (Electrical and Computer Engineering)

of

Curtin University

June 2014

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature: [illegible] …………………………………

(ADIB KABIR CHOWDHURY)

Date : 9th June 2014

Acknowledgements

In the name of GOD, most gracious, most merciful

When I am writing the thesis, the first person that comes to my mind is my supervisor, Professor Ashutosh Kumar Singh. His guidance, patience, technical suggestions and encouragement has always kept me motivated. My sincere and utmost gratitude and respect goes to my supervisor, who has nurtured me since my undergraduate studies. His supportive attitude and kindness has strengthened my path of research.

I would like to express my appreciation to my mentor and Co-Supervisor, Mr. Lenin Gopal who has always kept me motivated and gave me suggestions time to time. I would also like to gratefully thank my thesis committee Chairperson Associate Professor Zhuquan Zang. Apart from that, I am obliged to acknowledge my friend and research partner, Nikhil Raj. His depth in technical knowledge has indulged me to learn many things during my research work. I thank my parents, Miss Hamida Khatun and Mr. Ashadul Kabir Chowdhury for their utmost mental and financial support, my wife Nur Syazareena binti Zainuddin, who have continuously supported me and assisted in my thesis writing. Their supports have given me confidence and positive energy to overcome hurdles during any difficult times.

My sincere thankfulness to my research colleagues Shamil Mohamed, Alex Goh Kwang Leng, Balaji Palani, Omid Nabinejad and Maneesh Singh who have directly or indirectly dedicated their time or knowledge to assist in my thesis writing. Last but not least, I would like to thank Muhammad Ibrahim for his kind support, Mr. Veeramani

Shanmugam, Dr. Ekhlasur Rahman, Dr. Sujan Debnath, the postgraduate students and academic staffs in Curtin University. Last but not least, I appreciate the hard work and co-operation of Nur Afnizan Johan, Jenny S.C. Teoh and Siti Nurbaya. Their administrative support has helped me numerous times during my research period.

Furthermore, I would like to thank Prof. Clem Kuek, Prof. Marcus Man Kon Lee, Prof. Ashutosh Kumar Singh, Prof. Ir. Dr. Lau Hieng Ho and Curtin University, Sarawak Campus for providing the scholarship entitled “Curtin Sarawak Postgraduate Stipend Award 2013”. This stipend award has helped me to continue my research without any financial difficulties.

Abstract

Multi-Valued Logic (MVL) synthesis has economically revolutionized the method of designing logic functions. In the era of technological advancement of the 21^{st} century MVL becomes an alternative to our universal binary logic. MVL does not only present a theoretical foundation but also offers various experimental analysis and implementation for electronic circuitry. Representation of logic functions using binary logic requires higher number of interconnections hence causing circuits to be slower. As a branch of emerging technology, circuit simplification and size reduction is becoming a necessity in Very Large Scale Integration (VLSI) industry. MVL circuits provide a reduced chip size which performs faster and more efficiently compared to binary counterpart. These circuits demonstrate advanced dynamic and static performances over binary.

Therefore efficient MVL synthesizing techniques are desirable in the trend of electronic industry. Although many of the existing concepts have been developed and deployed, efficient algorithm for better performance in MVL synthesis still remains as a necessity. To overcome this, novel algorithms using different techniques have been proposed in this thesis. Efficient synthesis techniques for synthesizing and realizing MVL functions were developed to ensure lesser on chip interconnections, reduced delay time and faster speed. Furthermore, Artificial Intelligence (AI) incorporated MVL operators were also proposed to ensure reliable, robust and universal logic synthesis. Many sets of benchmark circuits, IC and MVL operators were used for experiments throughout the thesis. Some that were discussed are a benchmark of 50000 sequentially generated 4 valued 2 variable, benchmark of 19600 randomly generated 3 valued 2 variable, benchmark of 49998 randomly generated 4 valued 2 variable MVL functions, IC HEF4007UBP and MVL operators.

Firstly, a set of novel algebraic, postulates and logical operators was proposed to realize MVL functions. Novel High Deduction Algorithm (HDA-MVL) was represented to synthesize MVL functions. The usage of proposed MVL in logic synthesis operators revealed that a reduction of 18.00% was achieved in reducing gate count. A remarkable 33.00% reduction was observed in reducing usage of MIN operator for synthesizing MVL functions. In the worst case scenario, proposed HDA-MVL algorithm achieved 45.46% more success in reducing Product Term (PT) over evolutionary Ant Colony Optimization (ACO) algorithm. In the experiment, it is observed that overall average PT reduction was achieved by 56.88% for HDA-MVL algorithm.

Thereafter, a novel Neural Network Deployment Algorithm (NNDA-MVL) was proposed. The NNDA-MVL used the back-propagation learning capability as well as the proposed MVL operators to train. From the experimental results, it was depicted that the NNDA-MVL algorithm managed to achieve an accuracy of 99.97% for EXTENDED AND neural operator. The results also showed that the proposed NNDA-MVL induced the output delay in the range 0.01 to 0.04 seconds for the trained operators.

The advantages of NNDA-MVL algorithm was demonstrated with realization of synthesized MVL function with lesser MVL operators. Overall the algorithm showed an improvement of 52.94% for MVL MIN gate reduction and reduced MVL neural net internal link connections of 23.38% compared to existing techniques.

Alternatively a method for synthesizing Reduced MVL Networks (RMVLNs) using Non-Zero Multi-Valued Decision Diagram (NZMDD) is proposed. It is observed that reduced average PT is achieved in MVL synthesis using NZMDD. During synthesis, the proposed NZMDD algorithm reduced the number of PT by 52.62% while ACO-MVL reduced by 40.13%.

Implementation of MVL operator was accomplished by proposing a novel voltage mode MAX operator. The validation of the MAX circuit was done on enhancement mode transistors offered by IC HEF4007UBP. Proposed voltage mode MAX operator is 92.86% faster than the existing current mode MAX operator. During digital implementation, the proposed operator required 71.43% less transistor compared to existing operator.

Further experiment was conducted on a Voltage-Mode 3 Transistor (VM3T) based MAX circuit for implementation of MVL system. Simulations performed on 180nm technology revealed that proposed VM3T MAX has a lesser rising edge delay of 22.22% compared to existing CMOS NOR. It was also observed that proposed operator consumed 25.37% less power than the existing operator. MAX is a basic operator for synthesizing MVL functions. In this thesis, it was presented that optimized MAX operator can do fast realization with increased efficiency. An effective use of this operator was shown by realizing NOR gate and its comparison with standard CMOS NOR gate.

Author’s Note

JOURNAL PAPERS:

1. Adib Kabir Chowdhury, Nikhil Raj and Ashutosh Kumar Singh. “Synthesis and Reduced Logic Gate Realization of Multi-Valued Logic (MVL) Functions using Neural Network Deployment Algorithm (NNDA)”, Journal of Engineering Science and Technology. (Accepted for Publication)

2. Adib Kabir Chowdhury, Nikhil Raj and Ashutosh Kumar Singh. “Non-Zero Multi-Valued Decision Diagram (NZMDD) based Synthesis of Multi-Valued Logic (MVL) Functions”, Journal of Manufacturing Science and Technology. (Accepted for Publication)

3. Adib Kabir Chowdhury, Nikhil Raj and Ashutosh Kumar Singh. “Area Efficient MAX Operator for Multi-Valued Logic Realization”, Electronic Materials Letters. (Under Review)

4. Adib Kabir Chowdhury, Nikhil Raj and Ashutosh Kumar Singh. “A Review on Synthesis Methods and Application of Multi-Valued Logic”, Journal of Engineering Science and Technology Review. (Under Review)

CONFERENCE PAPERS:

1. Adib Kabir Chowdhury, Nikhil Raj and Ashutosh Kumar Singh. “An Analysis of Novel MVL Neural Operators Using Feed Forward Back-Propagation, Realization and Application of Logic Synthesis, International Conf. on Intelligent Network and Computing, Miri, Malaysia. (Accepted).

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACO-MVL	Ant Colony Optimization- Multi-Valued Logic
AI	Artificial Intelligence
CCD	Charged Coupled Device
CMCL	Current Mode CMOS Logic
CMOS	Complementary Metal Oxide Semiconductor
CONVERTER	MVL Logic Level Converter
CPU	Central Processing Unit
DC	Direct Cover
DD	Decision Diagram
DIP	Dual In-line Package
ECL	Emitter Coupled Logic
EVEN	Even MVL Operator
$ExAND^{C}_{number}$	EXTENDED AND prediction
$ExAND^{T}_{number}$	Total number of EXTENDED AND
EXTENDED AND	MVL AND Operator
GA	Genetic Algorithm
HDA-MVL	High Deduction Algorithm- Multi-Valued Logic
IC	Integrated Circuit
INVERTER	MVL Complement Operator
MAX	Maximum
MIN	Minimum
MIN^{C}_{number}	MIN prediction

MIN^{T}_{number}	Total number of MIN
ML	Machine Learning
MOS	Metal Oxide Semiconductor
MV	Multi-Valued
MVL	Multi-Valued Logic
NN	Neural Network
NNDA-MVL	Neural Network Development Algorithm- MVL
NN_ExAND	Neural Network EXTENDED AND
NN_MAX	Neural Network MAX
NN_MIN	Neural Network MIN
NOR	Not Or
ODC	Ordered Direct Cover
ODD	Odd MVL Operator
PLA	Programmable Logic Array
PSO	Particle Swarm Optimization
PT	Product Term
QMDD	Quantum MV decision diagram
RMVLN	Reduced Multi-Valued Logic Network
ROM	Read Only Memory
SETA	50000 Sequentially Generated Data Set
SETB	19600 Randomly Generated Data Set
SETC	49998 Randomly Generated Data Set
SETUN	Russian Computer Name
SoC	System on Chip

SoP	Sum of Product
VLSI	Very Large Scale Integration
VM	Voltage Mode
VM3T	Voltage Mode 3 Transistor
WDC	Weight Direct Cover

Glossary

a_n	Definite value 1
A	Set 1 of real values
A^c	Complement of A
b	Scalar bias
b_n	Definite value 2
B	Set 2 of real values
B^c	Complement of B
e	Exponential
E	Error
f	Function 1 of x
f_n	Sub function of f
g	Gradient
G	Function 2 of x
G_{NM}	Gauss Newton Method
h	Hessian matrix
H	Hidden layer
i	Sequence of inputs
I	Implicant representation
I_a	Partial implicant 1
I_b	Partial implicant

j	Sequence of hidden neurons
J	Jacobian matrix
k	Standard odd detection
l	Logic level
l_n	Output for a PT
$\overline{l_n}$	Complement of l_n
L_{state}	Learning rate
L_{MM}	Levenberg Marquardt Modification
m	Standard even detection
mse	Mean square error
$M_{constant}$	Momentum constant
n	Logic Constant
N_{Output}	Neuron output
p	Logic operator output
R	Set of real numbers
v_b	Level base voltage
v_{ib}	Initial base voltage value
$\overrightarrow{w_n}$	Weight vector
x	Continuous quantity
$\bar{x}$	Complement of x
x_m	Sub co-ordinate 1 of function

$\overline{x_m}$	Complement of x_m
x_n	Sub co-ordinate 2 of function
$\overline{x_n}$	Complement of x_n
$\overrightarrow{x_n}$	Column vector
X_1	Set of definite values 1
X_2	Set of definite values 2
$\overline{X}$	Complement of definite value
${}^{a}X_{n}^{b}$	Window literal
y	Radix
y_j^O	Output from output layer
y_j^H	Output of hidden layer
z	Variable length
${}^{a}\vec{x}_1^{\,b_1,b_2}$	Partial EXTENDED AND operator
φ	Transfer function
$\varphi(v)$	Hyperbolic tangent function
β	Layer of neural network
β_{input}	Total input to 1st layer
β_{hidden}	Total hidden neurons from hidden layer
β_{output}	Total output neurons from output layer
ω	Weights of neural network
ω_{ij}^H	Weights from input to hidden neurons

ω_{ij}^{O}	Weights from hidden to output neurons
β_{j}^{H}	Inputs coming from input layer
β_{j}^{O}	Input from hidden layer
Δw	Weight change
$\Delta w_{previous}$	Previous weight change

Chapter 1

Introduction

In this era of globalization and communication, electronic devices have been focused as the central role for efficient and cheap communication to be achieved. An efficient electronic device would require an effective circuit in terms of memory size, performance and the complexity of the circuit. This is where logic comes into the picture.

The concept of logic has been introduced as early as 1979 by Gottlab Frege who is often considered as the founder of logic especially in the basis of mathematics (Davis 1973). It is found by Frege that mathematics can be derived from logical principles. This is achieved by separating pure logical principle of interference which is represented by a mathematical proof (Frege 1977).

According to Clark and Reeves (1990), logic is a study which formalizes language and reasoning. Logic indicates a concept of well-developed formula which is decided to be semantic or syntactic (Gottwald 2007). It also acts as the very base of idea for computer sciences as computer science being the product of problems and methods developed using mathematical logic. Apart from being the root of computers science, logic also plays a crucial role in systems of circuits' design, relational data base systems, experts system as well as model checkers and proof of theorem.

Binary logic or also known as the Boolean logic- taking the name of its formulator, George Boole serves as one of the basics of computer science (Parkes 2002). In binary logic, the expressions are evaluated to either 1 (true) or 0 (false). This shows that expressions in binary logic have a finite number of variable which are either 1 or 0 in order to determine the truth values for specific expressions. One of the crucial applications of binary logic is to describe logic circuits mathematically (Gibson 2013). Other application includes modeling for digital circuitry (Parkes 2002).

Although binary logic is used widely in electronics and computer science, there are some limitations to it. As electronic circuits become more complex each day, circuits especially the conventional one which uses binary logic tends to be limited in certain aspects. Some of the limitations are large layout area required for the circuits with binary logic. This will then lead to increase in power consumption as well as capacity constraints (Sarif and Abd-El-Barr 2008). Other problems with usage of binary logic are limited storage of data as well as less availability of bandwidth and spectrum in cable together with wireless communications (Sarif and Abd-El-Barr 2008).

To overcome the limitations of Binary logic, Multi-Valued Logic (MVL) is often taken as an alternative to the former logic (Temel and Morgul 2002). MVL is a system which uses the variables that can take the cardinality of three and more discrete set values (Miller and Thorntorn 2008). Often used in the high-level system design, MVL specifically the multi-valued symbols represents the domain values of constants, variables, and functions (Yuan, et al. 2013).The concept of MVL is first introduced theoretically in 1920s by E.L Post (Post 1921). However, it took almost another 30 years to see application of MVL through the work of SETUN- a Russian computer

(Epstein, Frieder and Rine 1974). This is followed by a presentation by Dunderdale in 1969 on the design technique of ternary ECL circuit (Dunderdale 1969) which further focused on the application of MVL. Also in the late 1960s, there were many major attempts to formalize the process of MVL, its design of systems and minimization as well as other related issues regarding MVL (Smith 1981).

Most research done on applications of MVL follows three main directions. One of the direction lies is in escalation of binary system (Smith 1981). This approach is not new as the usage of MVL can greatly improve binary mode in terms of reliability especially by using the ternary MVL (Higuchi and Kameyama 1975). An example for this is approach is to use MVL as a mean of representation for multiple-output Binary logic function (Dubrova 1999). This is done by changing the Binary function to a single-output MVL function which the output is treated as a single MVL variable (Brayton, et al. 1996). Other example also includes application of MVL to analyze binary switching circuits giving attention to Metal Oxide Semiconductor (MOS) Very Large Scale Integration (VLSI) circuits.

Other direction of research done on MVL application is found in terms of increasing memory. By using MVL, number of lines needed to transmit huge data in parallel can be reduced (Smith 1981). This method will make betterment in terms memory development which memory arrays can be compacted. One of the examples can be observed in Read Only Memory (ROM) structures which were developed by Intel using four-valued ROMs in two of their products namely the 8087 arithmetic coprocessor and the 432-03 peripheral processor (Posa 1980). The four-valued ROM works as two-bit encoding reduces the size of ROM in order to raise the array product availability (Stark

1981). Other example of MVL application in memory design is utilized in MVL serial memory using Charged-Couple Device (CCD) (Yamada, Fujishima and Nagasawa.Koichi 1978). This produces block-random-access memory systems which makes the MVL storage cost effective due to the comparative balance cost in storage cells and signal regeneration (Terman, et al. 1981).

Apart from that, MVL is also widely applied in the communication and signaling field. The application ranges from the clear-cut connection among circuits and even gates to the more complicated channel coding (Smith 1981). Specifically discussing the communication aspect, MVL can be used to reduce the bandwidth of non-binary digital signal as well as compressing the information transmitted (Smith 1988). This has been first achieved by Motorola with the introduction of their product MC 145026/7/8 (Smith 1988). This product uses a customizable set of Complementary MOS (CMOS) chips in remote control applications. Data is encoded in four-valued units which comprise two binary digits by transmission between chips as a serial stream (Smith 1988). The MVL coding manages to add the pair of chip’s addressing range from 2’= 512 to 39 = 19,683 which proved to be a great improvement in low-cost, pin-limited Dual In-line Package (DIP) environment (Smith 1988). The concept of MVL is also used in the space vector pulse-width modulation scheme as presented by Sagar, Shiny and Baiju (2013). In this paper, MVL is utilized in the realization of computationally efficient space vector pulse-width modulation via generation of MVL algorithm (Sagar, Shiny and Baiju 2013).

Other Motorola’s product that uses MVL is the Motorola Emitter Coupled Logic (ECL) MC10194 dual line driver or receiver. It utilizes ternary signal which let full two-way operation on high-speed multiport single line bus (Etiemble 1981). Now, it is possible for any two devices on the bus communicating with each other with 3 valued-signals. The same applies to communication between one device to other remaining devices in a binary mode (Etiemble 1981; Ross 1977). In recent years, advanced applications of MVL are also made. This can be seen in the research article by Rafiev, Murphy and Yakovlev (2010) which utilizes MVL synthesis approach with security-related technique that are implemented in the devices' security such as smart card (Rafiev, Murphy and Yakovlev 2010). Song, Park and Oh (2014) also uses MVL in their research work for an Asynchronous Handshake Protocol. In this paper, the authors presented a data encoding scheme which is used to reduce the wires needed in the conventional delay-insensitive data encoding mechanisms (Song, Park and Oh 2014). Other applications of MVL includes but not limited to designing logic systems together with memory, coding of multi-level data and unique purpose digital processors (Sarif and Abd-El-Barr 2006). A general summary of MVL applications are as Figure 1.1 below.

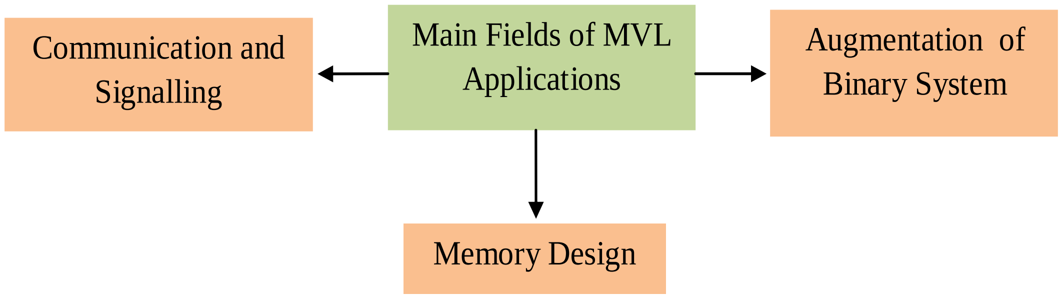

Main Fields of MVL Applications
Communication and Signalling
Augmentation of Binary System
Memory Design

Figure 1.1: Main fields of MVL applications

From many applications of MVL, it has been observed that MVL manage to super pass the conventional binary logic. The application of MVL in replacing binary logic or in augmentation to the latter reduces the need of power, improving speed and increasing the packing density of circuits especially VLSI circuits. One example which portrays reduce of energy can be observed in VLSI logic functions by using low-energy adiabatic logic circuits. The integration between the low-energy adiabatic binary and multiple valued CMOS logic is able to make a significant reduction of power consumption in VLSI chips.

As for MVL increasing the data density, this is due to the use of MVL inside VLSI chip which reduces the complexity of interconnection inside the chip. This will then lead to make the device containing the chip more efficient in holding larger amount of data in it. MVL increasing data density can also be achieved along with reduction of delay time and reduction of interconnections by embedding the multiple logics on single IC known as System on a Chip (SoC) (Romero, et al. 2014).

Next benefit of MVL is as an easy interface to binary logic. To explain further, MVL or specifically in this case, quaternary logic make the interfacing to binary logic easier as the radix $4=2^2$ allowing simple encoding or decoding of circuits. MVL is also able to boost the computational ability and is also applied in the fault diagnosis of sensor of magnetic bearings (Hu, et al. 2004). The latter is done by generalizing the binary algebra only two logic values. The application of MVL here is to measure state magnetic bearing by dispensation of signals from sensors (Hu, et al. 2004).

All the above mentioned benefits of MVL will give a significant impact on economy. The less number of interconnections used in chip and circuits contribute to less cost on the materials in producing the chips and circuits of electronic devices. This will start the chain of reaction where electronic devices available for sale at much lower price than before without sacrificing the efficiency of the said devices. A general summary of benefits of MVL is as Figure 1.2 below.

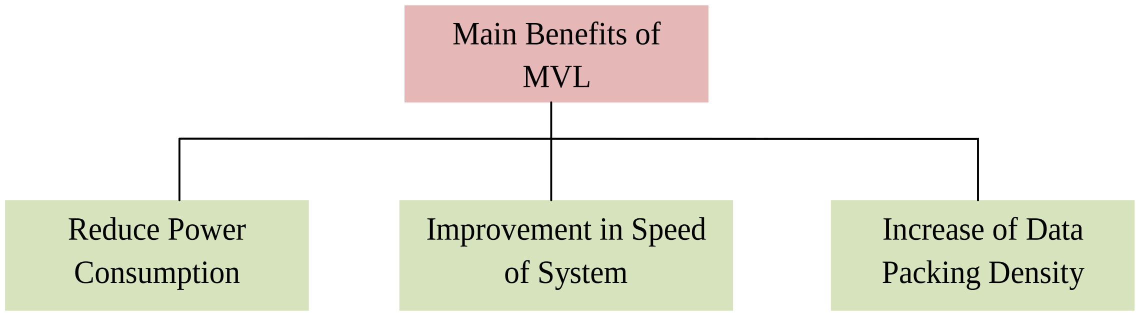

Main Benefits of MVL
Reduce Power Consumption
Improvement in Speed of System
Increase of Data Packing Density

Figure 1.2: Benefits of MVL

As MVL is crucial in many circuits and chips in electronic devices, a synthesis of the logic determines the efficiency MVL. Hence, research on MVL has mainly focused on improving MVL and its applications. Synthesis of MVL carries the meaning of a process that chose implicants to cover minterms in the said MVL function (Sarif and Abd-El-Barr 2008).

One of the research done on MVL synthesis is by using algorithm to synthesize 4-valued one-variable functions which will then be implemented using CMOS or Current Mode CMOS Logic (CMCL) circuits (Abd-El-Barr and Al-Mutawa 2001). The algorithm is introduced based on the usage of cost-table which comprises 48 functions. The algorithm will select two functions from table to create new function until all functions are synthesized (Abd-El-Barr and Al-Mutawa 2001).

Additional method of MVL synthesis includes the use of hybrid approach which is a combination of current mode and voltage mode CMOS circuits (Chang and Lee 1994). This method reduces one half of the synthesized circuit area hence making the synthesis of MVL faster (Chang and Lee 1994). A method to improve reversible logic synthesis is also proposed using Quantum Multiple-Valued Decision Diagram (QMDD) (Feinstein and Thorntorn 2009). This synthesis can reduce the required gates and yet manage to maintain minimum garbage outputs and ancillary inputs (Marinescu and Marinescu 2005). Another utilization of Multiple-Valued Decision Diagram (MDD) is as presented by Mo, Xing and Amari (2014). Here, the MDD method is used to increase the reliability analysis of a non-repairable binary-state phased-mission systems (Mo, Hing and Amari 2014). Kostolny, Kvassay and Zaitseva (2014) also utilizes MDD to represent complex system. This is because MDD manages to be processes efficiently on computer as well as taking up less storage space (Kostolny, Kvassay and Zaitseva 2014). Design of circuit can also contribute to better way of synthesizing MVL. This is shown in one of the researches which focus on the design and implementation of Integrated Circuit (IC) gates' universal set (Romero, et al. 2014).

Apart from that, the division of algebraic of MVL synthesis for MVL functions is introduced in (Wang, Lee and E. 1994). The division contains two MVL boolean properties ('identical' and 'complementary') in order to develop the new mix-algebraic division procedure (Wang, Lee and E. 1994). Heuristics approach utilizing near optimal product terms can also be used in MVL synthesis as seen in the proposed method of Particle Swarm Optimization algorithm (PSO) in Sarif and Abd-El-Barr (2008). This method is a population-based evolutionary algorithm which is also stochastic in order to solve engineering problems (Sarif and Abd-El-Barr 2008).

Next research on MVL synthesis is using the Weighted Direct Cover (WDC) and Ordered Direct Cover (Abd-El-Barr and Sarif 2007) which is also a type of heuristic approach. WDC selects different minterm or implicant based on weighted sum approach on different criteria. On the other hand, ODC selects minterm or implicant based on priority (Abd-El-Barr and Sarif 2007).

Iterative heuristics approach also plays an important role in synthesis of MVL. Among the researches done in iterative heuristic are as the papers by Yildirim, Butler and Yang (1993), Kalganova, Miller and Fogarty (1998), Tirumalai and Butler (1988) and Kalganova, Miller and Lipinitskaya (1998). Iterative heuristic explores bigger space of solution in reaching at near optimal solutions (Sarif and Abd-El-Barr 2006). Another example of iterative heuristic approach in synthesis of MVL is the Genetic Algorithm approach in which the representation of solutions in form of chromosomes strings (Sarif and Abd-El-Barr 2006).

Moving on to the next synthesis, involvement of multi-valued one and two-variable functions synthesis utilizing logic operations existing in CCDs (Abd-El-Barr, Vranesic and Zaky 1991) is discussed. As for the more current synthesis of MVL, the authors, Sarif and Abd-El-Barr (Sarif and Abd-El-Barr 2008) found a new way to synthesis MVL functions. The synthesis is done by an injection of multiple connected pseudo minterms to reduce product terms needed in order to synthesis MVL functions (Sarif and Abd-El-Barr 2008). The idea is gathered to make the function realization less complex with reduction of implicants. This will make the circuits work better (Sarif and Abd-El-Barr 2009).

In Abd-El-Barr and Sarif(2006), the Ant Colony Optimization (ACO) algorithm is proposed to synthesize MVL functions. ACO works with the ants selecting the correct minterm and implication and at the same time the ant will leave a pheromone trail as extra information for the next ant to use while making the selection (Abd-El-Barr and

Sarif 2006). Programmable Logic Array (PLA) decomposition is another efficient method to synthesize MVL. PLA decompose the system into smaller subsystem which will lead to minimization of logic network (Sasao 1988). Summary of different approaches are presented in Figure 1.3. These syntheses along with few current methods in synthesizing MVL are discussed in details in the later part of the thesis.

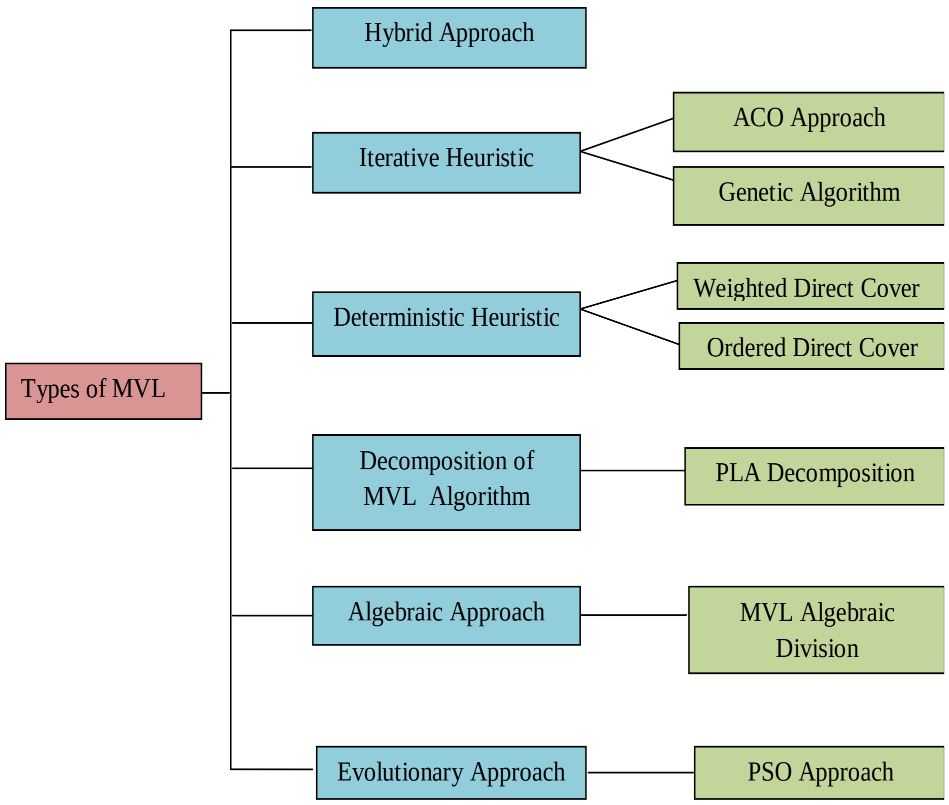


Figure 1.3: Summary of some MVL synthesis approaches

This thesis is carried out with its main objective to explore various efficient methods to synthesize MVL functions as improvements and additions to the current existing methods discussed above. To achieve the objective stated above, the thesis presents the development of several novel algorithms, postulates as well as the demonstration of reduction in gate count and lesser chip interconnections for digital design in comparison to existing techniques.

The thesis is organized as below:

Chapter 2 presents the basis of MVL operators used to synthesis MVL. These operators are used extensively for MVL synthesis throughout the thesis. The definitions and background for each of these operators are described in details. Apart from that, basic sets of benchmark circuit which are used in experiments and analysis in the later part of the thesis are explained. Introduction to HEF4007UBP integrated circuit is also made in this chapter.

In Chapter 3, a comprehensive study is made in regards to MVL synthesis techniques. The chapter starts off with a discussion on synthesis of MVL using the direct cover approach (Abd-El-Barr and Esam 2013), the iterative heuristic approach (Sarif and Abd-El-Barr 2006), the decompositional method (Files, Drechsler and Perkowski 1997), the algebraic approach (Romero, Martins and Santos 2009)and last but not least the neural network approach (Hsu, et al. 1990). Under these techniques, further researches are presented via few examples utilizing the each of the techniques discussed. Comparison among the different synthesis techniques is also made using synthesis of 5000 sequentially generated 4 valued 2 variable functions. Chapter 3 ends with an analysis of the experiment carried to compare the outcome of MVL synthesis using various techniques.

Next, Chapter 4 is outlined with explanations regarding the novel algebraic postulates and logical operators whereby both the postulates and operators are used in realizing MVL functions. Again in this chapter, definitions and the concept of all the logic operators are presented in detail. They are also proven mathematically. The chapter concludes with window lateral and short literal as MVL entity to be different from the conventional ways.

The thesis continues with Chapter 5 in which High Deduction Algorithm (HDA-MVL) is introduced as one of the way to synthesize MVL. First and foremost, definition and the concept of HDA-MVL are discussed in this chapter. This is also includes the necessary steps taken to synthesis MVL using HDA-MVL. Next, the selection of minterm and implicant is explained followed by algorithmic development of HDA-MVL. Chapter 5 also covers experimental analysis of variable functions generated using HDA-MVL alongside comparison with other methods of MVL synthesis. The chapter summarized that HDA-MVL outperforms other MVL synthesis methods.

In Chapter 6, presentation of key steps on Neural Network Deployment Algorithm (NNDA-MVL) is made. Next, neural architecture for EXTENDED AND, MIN and MAX operators are explained consecutively. This part also shows how NNDA-MVL works in order to produce trained neural operators. Then, the chapter is continued with neural and minterm representation whereby the neural operators can be used to construct neural network as well as train and representation of minterm as main component to logic reduction. Neural network operators are also analyzed before presenting the discussion on applications of trained neural MVL operators.

Chapter 7 continues the thesis with its focus on synthesis and reduced logic gate realization of MVL functions by using NNDA-MVL. Firstly, synthesis procedure using EXTENDED AND operator as an example is made followed by explanation on basic construction of NNDA algorithm. Next, analysis of neural network operators are made along with its experimental results. Comparison of MVL gate count and network link of different neural net logic operators while synthesizing MVL functions. Chapter 7 ends with conclusion of NNDA-MVL algorithm managed to reduce gate and neural net internal link compared to other existing methods.

Chapter 8 covers the topic of Non-Zero Multi-Valued Decision Diagram (NZMDD) which is taken as base for the synthesis of MVL functions. The MVL functions can be easily decomposed using decision diagrams (Files, Drechsler and Perkowski 1997). The first chapter goes to the surface of other researches done on construction of decision diagrams (Miller and Drechsler 2002) as well as synthesis of MVL using decision diagrams (Drechsler, Thornton and Wessels 2000). The chapter moves on with an effort made to reduce Product Term (PT) for synthesizing MVL which is represented as tree-like decision diagram, NZMDD which is addition to MDD (Miller and Drechsler 2002). This includes detailed explanation on Reduced Multi-Valued Logic Network (RMVLN) and the way NZMDD is used to RMVLN. After that, synthesis of MVL using NZMDD is also carried and experimented using data from (Chowdhury, Raj and Singh 2013) as well as comparing them with other synthesis techniques. Chapter 8 concludes with NZMDD algorithm outperformed ACO-MVL algorithm.

In Chapter 9, an introduction to logic gates are made. This follows with an explanation of how HDA-MVL is used to realize MVL expression with the newly proposed logic gates in this chapter. The chapter continues with representation of MVL functions as implicants which are then synthesized using EXTENDED AND operator. More details on the generation of implicants are discussed next. Also the chapter presents an elaborate study on transistor level implementation. Different MVL operators and their realizations using circuit are discussed. Then, the construction of MV logic gates are explained with the built of circuits using NMOS, PMOS transistors, resistors, diodes and comparators. Next in the chapter is an experiment to verify functionality of circuits. This verification is done using ORCAD PSpice transient analysis. The results from this experiment show that representation of MVL functions can be done by the discussed circuit elements. Lastly, Chapter 9 is summarized with comparison of the method with algebraic synthesis.

Chapter 10 of this thesis introduces two different architecture MAX operator and its application. Area efficient MAX operator for MVL realization is presented. The chapter elaborates in details about the proposed MAX circuit following definition of MAX operator and its algebraic properties. The proposed MAX circuit utilising two transistors as its base provides minimal delay. An experiment is conducted to verify the functionality of MAX using HSpice on 180nm technology. Secondly the chapter focus on the discussion of low power MAX operator design for MVL system. This second architecture MAX operator is realized in voltage mode, using limited transistors. This is followed by an application on using MAX in binary logic.

Following next is a detailed description on proposed MAX circuit which utilizes three transistors. A thorough understanding on NOR gate realization is also made. This chapter will focus on realizing the MAX operator in voltage mode with minimum number of transistors. After that, simulation results of proposed MAX circuit on HSpice on 180nm technology are presented. Chapter 11 is summarized that the proposed MAX circuit can achieved faster realisation and MAX based NOR gate operates with minimal delay at low power level.

Lastly in Chapter 11, all the findings and results of each chapter are concluded with some future directions.

Chapter 2

Preliminaries

In this chapter, fundamental basic of Multi-Valued Logic (MVL) operators are presented as most of these existing operators will be widely used for MVL synthesis throughout the presentation of thesis. Three basic sets of benchmark circuits which were used in analysis and experiments in the thesis are also discussed. Then, a brief introduction of Integrated Circuit (IC) HEF4007UBP is presented. IC HEF4007UBP was used in the physical implementation of the proposed Maximum (MAX) gate. Next, the highlight of feature sets which were investigated for the performance evaluation of different synthesis approaches is made.

2.1 Notations and Background

In Voltage Mode (VM), MVL logic levels are represented by voltage levels in terms of a base voltage. The initial base voltage value v_{ib} is set to 1.5 Volt for the experiment. Logic level l corresponded to an interval of the continuous quantity x. Hence, level 0 represents the null value and level 3 is associated with v_b = 4.5V and so on. As the logic level l changed v_b changed such that $v_b : (l)v_{ib}$ as explained in Jain, Bolton and Abd-El-Barr(1993) and Temel, Morgul and Aydin(2006).

Following the consideration of *z*-valued *y*-variable function f(x), where $x = \{x_1, x_2, \cdots, x_m\}$ and xi takes on values from $R = \{0,1,2,\cdots, y-1\}$, where "*y*" is the radix. The function f(x) is a mapping $f : R^{\,y} \Rightarrow R$. There are y^{y^m} different possible functions. If *y* = 3 and *z* = 2, then there are 3^{3^3}, meaning 19683 possible existing functions available (Jain, Bolton and Abd-El-Barr 1993; Temel and Morgul 2002).

In this thesis, a maximum of 4 valued 2 variable functions were investigated for the experiment purpose. For a maximum of 4 value, logic level *l*=3 would represent v_b = 4.5V. A logic level *l* related to an interval of the continuous quantity *x* such that (Gawande and Ladhake 2008; Jain, Bolton and Abd-El-Barr 1993).

$$x \rightarrow l : \{x \mid (l)v_{ib} \le x \le (l)v_{ib}\} \tag{2.1}$$

Definition 2.1.1- A Minimum (MIN) operator is defined as below, where a_n and b_n is a definite value and both of them are a member of set of all real values *R*. Hence $a_1, a_2, \cdots, a_n \in R$, $b_1, b_2, \cdots, b_n \in R$ and $0 \le \{a,b\} \le (y-1)$. The operation is such that $MIN(a_1, a_2, \cdots, a_n) = a_1 \bullet a_2 \bullet \cdots \bullet a_n$ and $MIN(b_1, b_2, \cdots, b_n) = b_1 \bullet b_2 \bullet \cdots \bullet b_n$ (Smith 1988; Abd-El-Barr and Al-Awami 2003). Therefore, $MIN(a,b) = a \bullet b = a \cap b$. If $X_1 \in \{a_1, a_2, \cdots, a_n\}$ and $X_2 \in \{b_1, b_2, \cdots, b_n\}$, then the representation of the MVL MIN operator is as Table 2-1 below.

Table 2-1: Tabular representation of MVL MIN operator

X_2	**MIN Function (MVL AND Gate)** X_1			
	$b_0 = 0$	$b_1 = 1$	$b_2 = 2$	$b_3 = 3$
$a_0 = 0$	0	0	0	0
$a_1 = 1$	0	1	1	1
$a_2 = 2$	0	1	2	2
$a_3 = 3$	0	1	2	3

The MIN operator can be represented by the following relation (Gawande and Ladhake 2008; Temel, Morgul and Aydin 2006):

$$MIN(a,b) = \begin{cases} a & if \quad a < b \\ b & otherwise \end{cases} \tag{2.2}$$

Definition 2.1.2- A MAX operator is defined for $a_1, a_2, \cdots, a_n \in R$, $b_1, b_2, \cdots, b_n \in R$ and $0 \le \{a,b\} \le (y-1)$, where a_n and b_n is a definite value and both of them are a member of set of all real values R (Abd-El-Barr and Al-Awami 2003). The MAX operation follows as $MAX(a_1, a_2, \cdots, a_n) = a_1 + a_2 + \cdots + a_n$ and $MAX(b_1, b_2, \cdots, b_n) = b_1 + b_2 + \cdots + b_n$. Therefore $MAX(a,b) = a + b = a \cup b$ .If $X_1 \in \{a_1, a_2, \cdots, a_n\}$ and $X_2 \in \{b_1, b_2, \cdots, b_n\}$, then representation of the MVL MAX operator is illustrated as Table 2-2. The MAX operator could be represented by the following relation:

$$MAX(a,b) = \begin{cases} a & if \quad a > b \\ b & otherwise \end{cases} \tag{2.3}$$

Table 2-2: Tabular representation of MVL MAX operator

X_2	**MAX Function (MVL OR Gate)** X_1			
	$b_0 = 0$	$b_1 = 1$	$b_2 = 2$	$b_3 = 3$
$a_0 = 0$	0	1	2	3
$a_1 = 1$	1	1	2	3
$a_2 = 2$	2	2	2	3
$a_3 = 3$	3	3	3	3

Definition 2.1.3- The complement of X or INVERTER is defined as below, where $X \in R$. Also the range of X depends upon radix y, where $0 \le X \le (y-1)$ (Nakahara, Sasao and Matsuura 2011). Table 2-3 shows the INVERTER operation.

Table 2-3: Tabular representation of MVL INVERTER operator

X	$\overline{X}$ **Inversion**
$X_0 = 0$	$\overline{X_0} = 3-0=3$
$X_1 = 1$	$\overline{X_1} = 3-1=2$
$X_2 = 2$	$\overline{X_2} = 3-2=1$
$X_3 = 3$	$\overline{X_3} = 3-3=0$

Therefore complement of X or the INVERTER can be represented as,

$$\overline{X} = y-1-X \tag{2.4}$$

2.2 Benchmark Circuits and IC HEF4007UBP

A total of 119598 benchmark circuits were used in different experiments throughout this thesis. Renowned researcher Mostafa Abd-El-Barr in his papers; Abd-El-Barr and Sarif(2006) and Sarif and Abd-El-Barr(2006) self generated 50000 sequentially generated benchmark circuits and compared them with different algorithms.

Based on his works, three different sets of benchmark circuits were generated using different methods. Data generation and computation were performed on a *Microsoft windows XP*, Intel Core2 duo 2.20GHz CPU, 0.98GB RAM machine. Generation of data and results in this thesis might vary if they are performed on different machines.

SETA (50000 Sequentially Generated Data Set) data consists of 50000 sequentially generated MVL benchmark circuits. This entire set of data was distributed in 14 different sections. Each section represents a fixed minterm- a combination of different functions. For the 19600 randomly generated data set SETB, the data was distributed in 7 sections. The distribution for both data sets is shown in the Figure 2.1 and Figure 2.2 below.

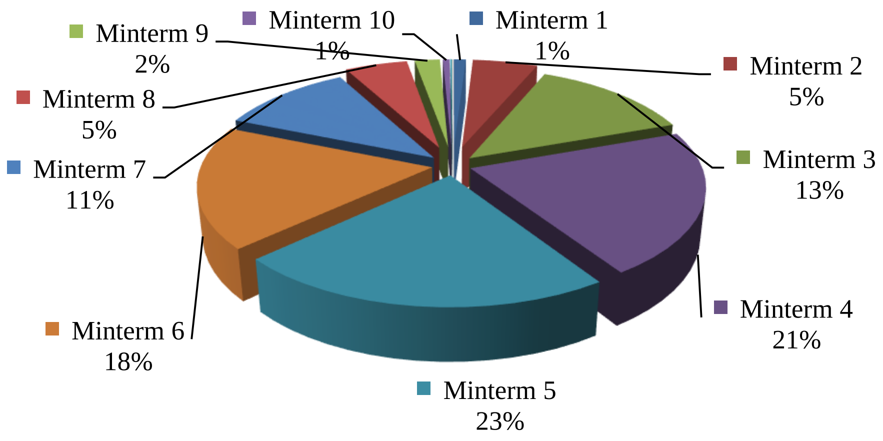


Figure 2.1: SETA-Sequentially generated benchmarks

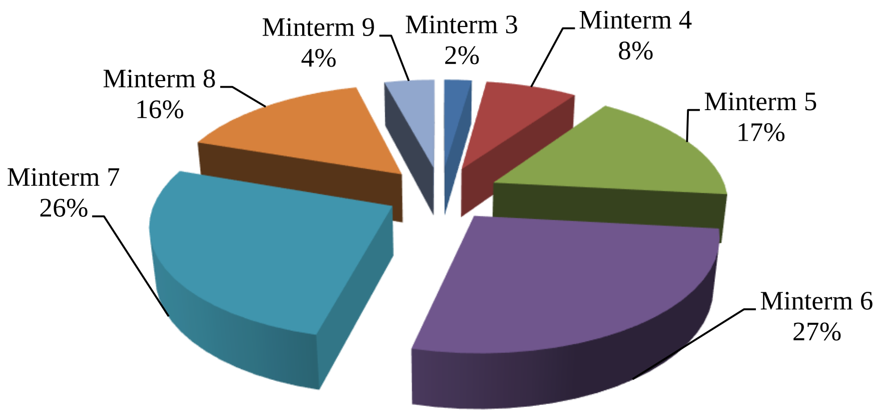


Figure 2.2: SETB-Randomly generated benchmarks

SETA was used in the analysis for HDA-MVL algorithm. It was also used to compare DC against evolutionary algorithms. SETB distributed highest 27.00% of the data for 6 minterm MVL function generation. This data set was used for the analysis of NZMDD algorithm to check the reduction in Product Term (PT). The algorithm was compared against evolutionary ACO-MVL algorithm.

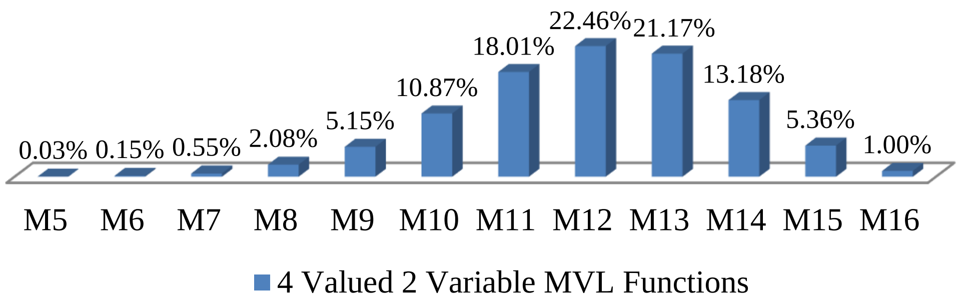


Figure 2.3: SETC-Randomly generated benchmarks

49998 randomly generated data SETC was randomly generated for a 12 section distribution. Highest 22.46% data of the set was allocated for minterm 12. As it can be observed from the Figure 2.3 shows a total of 49998 4-valued 2-varibale functions were generated.

In this thesis, a novel MAX architecture is proposed and implemented using the IC HEF4007UB. The IC is a dual complementary pair and inverter. The IC has three p-channel and three n-channel enchancement mode Metal Oxide Semiconductor (MOS) transistors. Each of the transistor's can be accessed separetly. The operator is composed of only one n-channel and one p-channel MOS transistor. Ease of accessebility to each of the different enchancement mode MOS transistor is one of the vital reason to choose this specific IC.

Throughout the thesis, this IC was used for two different experiments. One is in the area efficient logic MAX operator which was realized and tested. Another is in the analysis of the constraints such as delay, power consumption etc. which were compared against the existing architecture. The IC was operated up to a maximum of 15V which gave the opportunity to attempt various logic levels. Among many of this IC's applications are linear and high input impedance amplifiers, current drivers, high impedance buffers and oscillators (Smith 1988).

2.3 Feature Set for Different Approach Analysis

Feature set analysis is carried for all the approaches and their experiments. Feature set consists of terms which are compared against the existing architecture or algorithms. It is necessary to justify each approach by evaluating the terms in the feature set. Table 2-4 consisting of feature sets and techniques is presented as shown below.

Feature A denotes the logic expression and simulation based features. These features can be found in Romero, Martins and Santos (2009) and Abd-El-Barr and Sarif (2006) in their research articles. Simulation feature was used to confirm the logic level verification process. Logic expression and PT was analyzed to verify the reduction of the final synthesized function. A total of three features were analyzed.

Table 2-4: Feature set for different MVL synthesis approaches

Notation	**Synthesis Techniques**	**Feature Set**	**No. of Features**
A	Postulates	• Product term (PT) • Logic Expression Reduction • Simulation	3
B	Neural Network MVL operators	• Training Time • Hidden Layer Neuron Count • Delay • Accuracy • Application	5
C	NNDA algorithm	• Simulation • Interconnected Links • Gate Count • Network Architecture	4
D	NZMDD algorithm	• Reduction in Network Size	1
E	Circuit Implementation	• Gate Input Count • Gate Count • Delay • Power Consumption • Transistor Count • Simulation • CMOS NOR	7

Feature B and C reflects the features for Neural Network (NN) approach. Gate count and interconnected links were compared for the digital circuit implementation. Training time, hidden layer neurons and accuracy were investigated for training MVL operators. Results were compared against Zheng et al. (1998) and simulations were used for operator output verification purposes. For the performance evaluation there are total of four sections – Algebraic synthesis, Decision Diagram (DD) approach, Circuit Implementation and Machine Learning (ML).

Feature D denotes the size reduction features of the decision diagram. This feature was used to check the number of nodes taken to compactly represent a MVL function. This feature also helped to analyze PT taken to synthesize each function. On the other hand, last but not least notation E features many terms which were utilized for analyzing the ideal environment for implementing MVL MAX gate.

2.4 MVL Over Binary System

MVL technology reduces the number of lines required for parallel transmission. Hence the system is able to transmit more information with less transmission line.

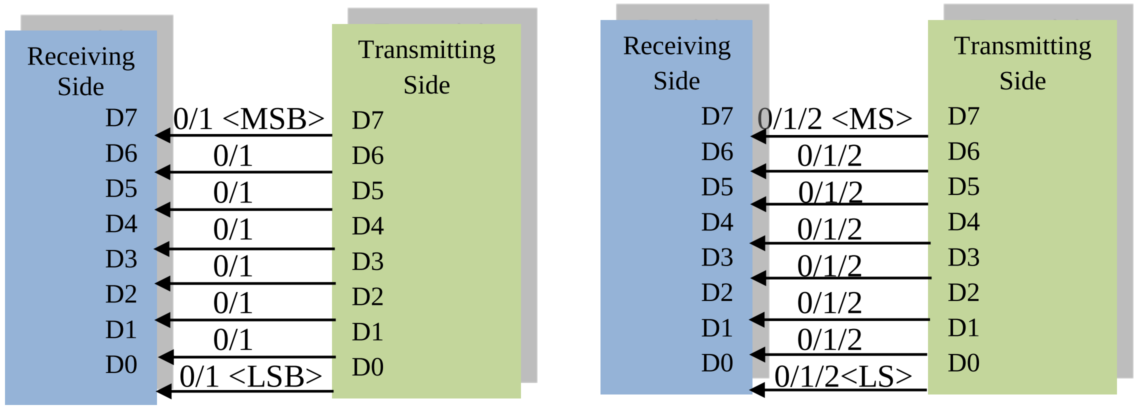


Figure 2.4: Binary Vs MVL parallel transmission

For example as shown in the Figure 2.4, in a parallel transmission scheme binary with seven data line can produce 256 combinations of information. MVL produced 64.88% more information with lesser transmission lines. MVL does not only benefit us in data transmission technology but also resolves bus connection and pin limitation problems. This futuristic technology can be also used in conjunction with binary sub-systems which will reduce wiring complexity. Lastly decreased power consumption over binary gives MVL the opportunity to become a major research field of study for the future technological advancement.

2.5 MVL Synthesis

Multi-level logic synthesis bridges all features of combinational logic, logic optimization and design for testability and verification (Brayton, et al. 1996). MVL synthesis is a process which helps accomplish desired circuit behavior which later forms into logic gate design implementation. Throughout this thesis, a set of logic operations are used for functional synthesis. The operators usually consist of MVL MIN, MAX, ODD, EVEN, INVERTER, EXTENDED AND other logic gates.

MVL synthesis optimizes logic functions up to the optimal level. Logic gates are used for the synthesis of MVL functions. A ternary function has up to 19683 possible different functions. All of these functions are essential to be optimized before logic level implementation. Higher functional optimization is achieved through lower hardware cost and efficient design architecture. The Figure 2.6 below illustrates the scope and contribution of this thesis in the process of MVL synthesis.

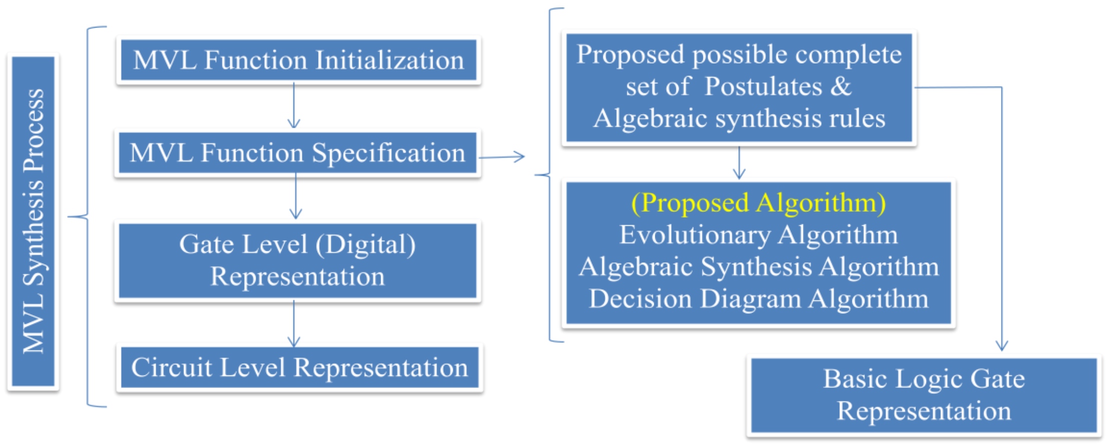

MVL Synthesis Process
MVL Function Initialization
MVL Function Specification
Gate Level (Digital) Representation
Circuit Level Representation
Proposed possible complete set of Postulates & Algebraic synthesis rules
(Proposed Algorithm)
Evolutionary Algorithm
Algebraic Synthesis Algorithm
Decision Diagram Algorithm
Basic Logic Gate Representation

Figure 2.5: Scope of research in the process of MVL synthesis

In conventional procedure, MVL functions are mapped and the minterms are extracted. Only non-zero minterms are taken under consideration to produce efficient synthesis algorithms. Various techniques are suggested by Yang and Wang (1990). In this section of the thesis, a generalized view of MVL synthesis mechanism is presented. It is assumed that 2 inputs and 1 output ternary function $G(X_1, X_2)$ represented in terms of all of its existing minterms, as shown below in Table 2-5.

Table 2-5: A 2-input 1-output ternary function $G(X_1, X_2)$

X_2	**Function** $G(X_1,X_2)$ X_1		
	$b_0=0$	$b_1=1$	$b_2=2$
$a_0=0$	1	0	0
$a_1=1$	1	0	0
$a_2=2$	2	1	1

Consider matrix G_{NS} for the non synthesized function. G_{NS} and its SoP representation is shown accordingly in the equation (2.5) and(2.6).

$$G_{NS} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 2 & 1 & 1 \end{bmatrix} \tag{2.5}$$

$$\begin{gathered} G(X_1, X_2) = 1 \bullet {}^0X_1{}^0 \bullet {}^0X_2{}^0 \; + \; 1 \bullet {}^0X_1{}^0 \bullet {}^1X_2{}^1 \; + \; 2 \bullet {}^0X_1{}^0 \bullet {}^2X_2{}^2 \; + \; 1 \bullet {}^1X_1{}^1 \bullet {}^2X_2{}^2 \; + \; 1 \bullet {}^2X_1{}^2 \bullet {}^2X_2{}^2 \\ SoP \; = \; G\left(X1, X2\right) \; = \; 10000 \; + \; 10011 \; + \; 20022 \; + \; 11122 \; + \; 12222 = 5PT \end{gathered} \tag{2.6}$$

Synthesized MVL functions are represented with lesser product terms. Non synthesized Sum of Product (SoP) required 5 PT, when synthesized expression required 3PT. Equation (2.7) shows the functional representation of synthesized $G\left(X_1, X_2\right)$. Table 2-6 highlights the minimized section of the function. The technique of minimizing the representation of these functions produces various synthesis algorithms. In this thesis various approaches for synthesizing MVL functions are discussed in detail.

$$\begin{gathered} G(X_1, X_2) = 1 \bullet {}^0X_1{}^0 \bullet {}^0X_2{}^1 \; + \; 2 \bullet {}^0X_1{}^0 \bullet {}^2X_2{}^2 \; + \; 1 \bullet {}^1X_1{}^2 \bullet {}^2X_2{}^2 \\ SoP \; = \; G\left(X1, X2\right) \; = \; 10001 \; + \; 20022 \; + \; 11222 = \; 3PT \end{gathered} \tag{2.7}$$

Table 2-6: Minimized portion of $G\left(X_1, X_2\right)$ during synthesis

X_2	**Function** $G\left(X_1, X_2\right)$ X_1		
	$b_0 = 0$	$b_1 = 1$	$b_2 = 2$
$a_0 = 0$	1	0	0
$a_1 = 1$	1	0	0
$a_2 = 2$	2	1	1

An efficient architecture of a synthesized function over original function differs during digital and circuit implementation. Design complexity, power consumption, delay, chip interconnections and number of transistors etc. plays important role in determining a better output. A comparison Table 2-7 between original and synthesized MVL function is represented as below.

Table 2-7: Difference between original and synthesized MVL function

MVL Function $G(X_1, X_2)$	**Original**	**Synthesized**
Number of Transistors	Transistor increases upon complete circuit implementation of function	Lesser transistor used to compared to original MVL function
Chip Interconnections	Higher number of interconnections needed	Synthesized function requires lesser interconnections
Logic Level Design	Complex and time consuming	Simpler and efficient
Circuit Implementation	Inefficient with respect to enormous hardware cost	Efficient with respect to lesser hardware cost

Chapter 3

Multi-Valued Logic Synthesis Techniques

3.1 Introduction

Over the last three decades, synthesis and simplification of multi-valued logic functions has become one of major research field in Integrated Circuits (IC) industry. Synthesis of Multi-Valued Logic (MVL) is one of the crucial aspects that make MVL-related systems and circuits more efficient and economical. In this chapter, a study covering types of synthesis done on MVL is discussed.

The chapter starts off with a discussion on synthesis of MVL using the direct cover approach. This approach can be divided into two-the Weighted Direct Cover (WDC) and Ordered Direct Cover (ODC) (Abd-El-Barr and Sarif 2007; Yang and Wang 1990). Next, the chapter explains in detail about Iterative Heuristic approach in synthesizing MVL. The Iterative Heuristic approach increases the possibilities to discover wider solution space (Abd-El-Barr and Sarif 2006).Some of the synthesis that utilized this approach is the Genetic Algorithm approach as discussed in Sarif and Abd-El-Barr (2006) and the Ant Colony Optimization as discussed in Abd-El-Barr and Sarif(2006).

Apart from these, a wide research is also carried in the Algebraic approach of MVL synthesis (Romero, Martins and Santos 2009). Besides that, the Neural Network Approach has also been presented as another method in MVL synthesis (Hsu, et al. 1990); Abd-El-Barr and Esam 2013; Abd-El-Barr and Sarif 2007). Lastly, comparison tabulation is made highlighting different synthesis methods of MVL. Another comparison table is also constructed to depict product terms needed for MVL functions using different types of synthesis.

3.2 MVL Synthesis Approach

There are few main categories of algorithms involved in the synthesis of MVL. According to Sarif and Abd-El-Barr(2006), the first two algorithms are the deterministic and the iterative heuristic algorithms. The other category of algorithm is the decomposition based algorithm in Al-Rabadi (2003) and Chojnacki and Jozwiak (2000). Different approaches are also observed from other literature (Allen and Givone 1968; Mathew, et al. 2008).

3.2.1 The Direct Cover approach

The deterministic algorithm is the algorithm which takes the Direct Cover (DC) approach as its basis. This algorithm focuses on selecting next minterm in order to be covered. It also chooses the proper implicant for covering selected minterm in iterative way using specific criteria state (Abd-El-Barr and Esam 2013; Yang and Wang 1990). According to Yang and Wang(1990), this step is further clarified by minimizing MVL logic truncated sum which selects both isolated minterms and implicants. Then, this will lead to the production if saturated minterm which can be reduced to a don’t care minterm to minimize the MVL logic done in the development of ND-algorithm. Here, the don’t care minterms are used as much as possible. Based on the research done by Abd-El-Barr (2011), the DC techniques to synthesize MVL are as following:

1. Choose a minterm
2. Identify a suitable implicant that covers the chosen minterm
3. Obtain a reduced function by subtracting the identified implicant from the (remaining part of the) function, and
4. Repeat steps 1 to 3 until no more minterms remains uncovered.

These steps are important especially to obtain the reduced number of products terms in order to cover a function given (M. Abd-El-Barr 2011). The DC approach can also be further classified into two approaches which can be used to combine all metrics from different minterm/implicant selection criteria. These approaches are the Weighted Direct Cover (WDC) which is used to specify weight for each selection criterion and the Ordered Direct Cover (ODC) which put the selection criteria for both minterm and implicant in order (Abd-El-Barr and Sarif 2007).

3.2.2 The iterative heuristic based approach

According toAbd-El-Barr and Sarif(2006), the iterative heuristic widens the chances to explore bigger solution set in order to arrive at near optimal solutions. One of the examples of the iterative heuristic approach is as discussed by Sarif and Abd-El-Barr(2006). According to the paper, the solutions are obtained by using the Genetic Algorithm approach (GA) where these solutions are taken as representation of strings of chromosomes. Each chromosome representing a product term which consists of some genes and each gene consist of five integer attributes. In the GA approach, the design of the parameter is determined by the length of chromosome. This is further explained by the inability for GA to find the best solution suited the chromosome. However, if the chromosome is very big in length, the GA will lose a significant amount of time in finding solution in the no-solution space. The best technique to use in this approach in a straightforward way is to use the length of the truth table as the length of chromosome. When this approach is compared to a DC approach, it is showed that the GA Approach yielded better results especially taking into consideration the number of products needed to synthesize functions (Sarif and Abd-El-Barr 2006).

Other iterative heuristic approach researched is the Ant Colony Optimization (ACO) approach. This meta-heuristic approach combines the distributed computation and autocatalysis in order to find the best solution for various number of combinatorial optimization problems. This approach is carried by using the 'ant' to find the optimal representation by selecting the right minterm and implicant (Abd-El-Barr and Sarif 2006). This approach is similar to DC approach to the extent that the best minterm is selected first followed by the best implicant for the chosen minterm. Then, the reduction of the MVL function table is reduced by removing selected implicant. However this similarity stops when the ant leaves a trail of pheromone on minterm/implicant so that the next ant will chose better based on the additional information left by the pheromone.

3.2.3 The decomposition method

Apart from the approaches mentioned above, MVL can also be optimized by using novel symbolic functional decomposition methods. The decomposition of MVL functions can lead to efficient implementation of logic circuits. This approach also manages to represent data effectively in information systems (Files, Drechsler and Perkowski 1997). This approach works by decomposing the system into few smaller subsystems hence making them easier to be synthesized. The decomposition can be implemented into various purposes. One of them is the programmable logic array (PLA) decomposition which is based on decomposing the original function into components matching to a set of interconnected PLA. As a result of this, the logic network will be minimized. Apart from that, the decomposition approach can also be implemented to reduce the space required in problems related to data representation in machine learning (Kohavi, et al. 1994).

Quaternary decision diagrams have advantage in representing and evaluating MVL functions (Sasao, Nakahara, et al. 2009). Other researchers like: Tsutomu Sasao has mathematically shown that how many variables it requires to represent Multiple-Valued incomplete specified functions (Sasao 2010). Daniel Stamate has shown that extended logic programs could be used to represent Multiple-Valued Logic systems (Stamate 2006). Arithmetic operators have also been implemented in multi-valued logic (Balasubramanian, Narayana and Chinnadurai 2005; Nakahara, Sasao and Matsuura 2011). Combinational and sequential digital circuits are proposed by Balasubramanian, Narayana and Chinnadurai(2005).

3.2.4 The algebraic approach

Another type of synthesis method is based on the Algebraic approach. This approach focuses on synthesizing the algebraic form of function. The synthesis is mainly done on the canonical form of the Sum of Extended Product (SoEP) terms, the duality and the circuit simplification procedure (Romero, Martins and Santos 2009). From the result gathered using this approach, its simplification techniques can be used to simplify various classes of MVL digital circuits. Apart from that, this Algebraic approach also allows the program tools to be implemented in order to simplify the limit imposed by algorithm complexity (Romero, Martins and Santos 2009).

For the purpose of comparison between some of the approaches to synthesize MVL, a benchmark consisting of 50000 sequentially generated 4 valued 2 variable functions has been generated and synthesized. The Figure 3.1 provides comparison of some approaches mentioned above. Average PT in regards to dissimilar numbers of MVL functions' minterms are used as benchmark. For the proposed method, MVL functions are generated sequentially not randomly. The comparison is shown below between Besslich (1986), Dueck and Miller (1987), Pomper and Armstrong(1981), Ordered Direct Cover (ODC), Weighted Direct Cover (WDC), Ant Colony Optimization (ACO), Genetic Algorithms (GA) and Revised Genetic Algorithms (RGA).

It can be observed from Figure 3.1that the worst case scenario for ACO-MVL is 13 minterm where the average PT usage is 7.18. Where as in the Figure 3.2, the overall average PT which is achieved by DC based algorithms and the Evolutionary heuristic based techniques are also shown.

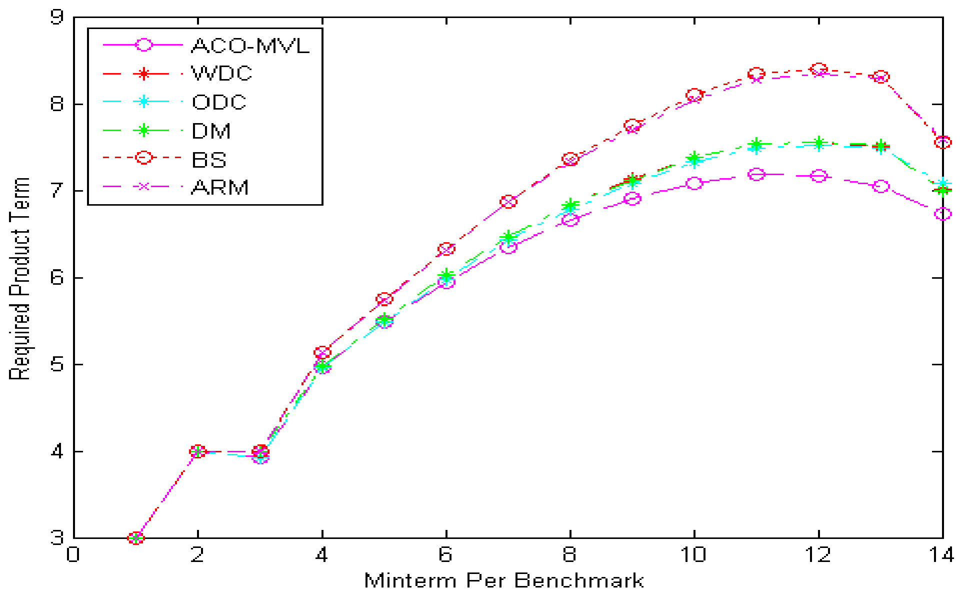


Figure 3.1: Comparison of proposed algorithm with existing evolutionary and direct cover algorithms

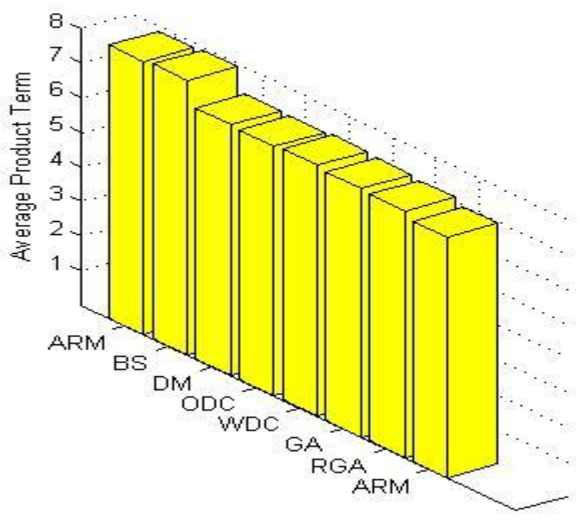


Figure 3.2: Average product term needed to synthesize MVL functions using different algorithms

The Figure 3.1 illustrates that the ACO-MVL algorithm outperforms all DC based algorithms. Among all the existing algorithms, the ACO-MVL algorithm yield better results regarding average product term required for synthesis of a given MVL function.

3.2.5 The neural network approach

The synthesis of MVL can also be done from the Neural Network approach. This approach is then can also be categorized further into few subsections. Hsu *et.al.* (1990) explained in detail about the three-valued neural logic network. In this network, the activations of nodes are limited to ordered pairs (1,0), (0,1) and (0,0) where the meanings of the third pair are UNKNOWN in the Kleene's logic and MEANINGLESS in Bochvar's logic (Hsu, et al. 1990). Besides that, the three-valued neural logic network can also be generalized into two types of networks. One is the probabilistic network which is used to predict recognition pattern and in expert system development. The other type of network is the fuzzy interference where the three-valued logic is comprehensive to handle the uncertainties in inference. The fuzzy interference network is crucial to be used in decision-making expert system (Hsu, et al. 1990).

Other research in the synthesis of MVL from the neural network approach is the error back-propagation in MVL system especially in the neural networks. The error back-propagation can derive target output pairs for each layer in the system from the global input-output data (Apostolikas and Konstantopoulos 2007). This propagation can be done by the Fuzzy Description Logic System where the system can calculate the input that is most approximate input for the fuzzy reasoned which is then used as supervised training data for adaptive learning of the first-level classifiers(Apostolikas and Konstantopoulos 2007).

Lastly, the synthesis of MVL can be carried by minimizing the multilayer feedforward of neural networks where multiple-valued multiple-threshold perceptrons are used as basic processing elements/nodes of the network (Ngom and Simovici 2004). Apart from that, construction of near minimal multi-valued neural network for computing given but arbitrary multiple-valued functions via implementation of partitioning algorithm is also discussed.

3.3 Summary

This chapter covered the types of approach in order to synthesis the MVL. The approaches of MVL synthesis discussed were the DC, iterative heuristic, decomposition method, algebraic and the neural network. These approaches were presented in details inclusive of various new techniques categorized under its specific approach.

The chapter also compared some types of synthesis discussed to measure its efficiency synthesizing the MVL. The first comparison was done using a synthesis of 5000 sequentially generated 4 valued 2 variable functions. It was shown in the first comparison that the ACO-MVL method outperformed the other types of synthesis in terms of yielding results in individual PT required for synthesis of MVL functions whereas the second comparison was done by comparing the resulting better outcome regarding average PT required for synthesis of a specific MVL function.

This chapter also concluded that Evolutionary and Combined heuristic algorithms gave benefits in terms of efficient functional synthesis. However, both of these algorithms required much computational complexity, hence more time to generate synthesized functions. Most of the existing efficient algorithms were computationally complex and required enormous amount of time (CPU time) for synthesizing MVL functions. The only draw-back of MVL algebra was the standard set of Logic rules and Universal logic gates. Existing Algebraic Synthesis methods were not efficient enough to compete against the recent evolutionary algorithms. However, with the current approaches such as ACO and GA, this was reduced significantly. On the other hand, more research should be presented to overcome these limitations.

Chapter 4

Logic Operators and Algebraic Postulates

In this chapter a set of novel algebraic postulates and logical operators has been proposed to realize Multi-Valued Logic (MVL) functions. All the postulates and operators are shown proven. Window literal or short literal as a MVL entity is defined complete differently as opposed to the conventional methods.

4.1 Introduction

MVL algebra has an important role in the synthesis of digital circuits (Lablan 2005-2011). The only draw-back of MVL algebra is that it has the standard set of logic rules and universal logic gates. Existing algebraic synthesis methods are not efficient enough to compete against the recent evolutionary algorithms. The postulates presented in this section are explained in detail. Some novel logical operators are suggested to realize MVL functions. Combining two separate window literal a minterm has been formed. Each column of a MVL functional matrix is shown as an implicant which can consist of any non-zero minterm.

The proposed postulates will result in a generalized MVL algebraic synthesis which is a unique approach to generate an efficient synthesis of MVL functions, compared to other existing algorithms or rules. The operators are essential in minimizing the hardware cost when realized. On the other hand, the discussed postulate rules help in reducing logic expressions. Throughout this thesis, many different approaches have utilized these operators and postulates to obtain better synthesis results.

4.2 Unique Multi-Valued Logic Operators

In this segment each of the proposed logic operators are presented and discussed in details. MVL operators such as ODD, EVEN, EXTENDED AND, INVERTER, EXTENDED COMPLEMENT and WINDOW LITERAL etc. are discussed along with their definitions. Below all the necessary definitions of the logic operators are explained,

Definition 4.2.1- The WINDOW LITERAL or short literal is defined as

$$ {}^{a}x^{b} = \begin{cases} b & if \quad x = a \\ 0 & otherwise \end{cases} \tag{4.1}$$

Where, $a_1, a_2, \cdots, a_n \in R$, $b_1, b_2, \cdots, b_n \in R$ and $0 \le \{a, b\} \le (y-1)$ where b is called the value of the literal. The definition of WINDOW LITERAL provides the opportunity for eliminating the usage of COMPLIMENTARY LITERAL (CL). Thus in this thesis, CL is not extensively used during the synthesis of MVL functions.

Definition 4.2.2- An EVEN operator is defined as below, where *a_n*, and *m* is a definite value. *m* is defined as a standard value for detecting an even value, where *m* = 2. *m* is a member of set of all real values *R*: $a_1, a_2, \cdots, a_n \in R$, $b_1, b_2, \cdots, b_n \in R$, $\{p, m\} \in R$ and $0 \le \{a, b\} \le (y-1)$.

The even operator detects whether value *p* is even or not. If *p* is even then $EVEN(p) = true$ and $EVEN(p) = p$ otherwise $even(p) = false$ and $EVEN(p) = 0$.

Therefore,

$$EVEN(p) = \begin{cases} p & if \quad p \bmod m = 0 \\ 0 & otherwise \end{cases} \tag{4.2}$$

And the EVEN function could be represented by the following relation:

$$EVEN\_Func(a,b) = \begin{cases} \max(EVEN(a), EVEN(b)) & if \quad EVEN(a) \,||\, EVEN(b) = true \\ 0 & otherwise \end{cases} \tag{4.3}$$

For any even input the output will be even. The purpose of the even operator is to detect any even input compared to odd input. If both even input is detected, the output is prompt to zero.

Definition 4.2.3- An ODD operator is defined as below, where a_n, *m* and *k* is a definite value. *m* is defined as a standard value for detecting an even value, where *m* = 2. *k* is defined as a standard value for detecting an odd value, where *k* = 1. *m* ,*k* is a member of set of all real values *R*: $a_1, a_2, \cdots, a_n \in R$, $b_1, b_2, \cdots, b_n \in R$, $\{p, m, k\} \in R$ and $0 \le \{a, b\} \le (y-1)$. The ODD operator detects wither value *p* is odd or not. If *p* is odd then $ODD(p) = true$ and $ODD(p) = p$ otherwise $ODD(p) = false$ and $ODD(p) = 0$.

Therefore,

$$ODD(p)=\begin{cases} p & if \quad (p-k)\geq 0 \quad and \quad (p-k)\bmod m = 0 \\ 0 & otherwise \end{cases} \tag{4.4}$$

And the ODD function could be represented by the following relation:

$$ODD\_Func(a,b)=\begin{cases} \max(ODD(a),ODD(b)) & if \quad ODD(a)\,||\,ODD(b)=true \\ 0 & otherwise \end{cases} \tag{4.5}$$

For any odd input the output will be odd. The purpose of the odd operator is to detect any odd input compared to even input. If both odd input is detected, the output is prompt to zero.

Definition 4.2.4- The EXTENDED COMPLEMENT of *x* is defined as below, where *x* a member of set of all real values *R*. Considering a MAX operation with the complement of *x,* different representation could be achieved from the EXTENDED COMPLEMENT operation. If MAX operation is performed with complement of *x* the following can be represented:

$$x+\overline{x}=\begin{cases} x & if \quad x>\overline{x} \\ \overline{x} & otherwise \end{cases} \tag{4.6}$$

If MIN operation is performed with complement of *x* the following can be represented:

$$x\bullet\overline{x}=\begin{cases} x & if \quad x<\overline{x} \\ \overline{x} & otherwise \end{cases} \tag{4.7}$$

Definition 4.2.5- Consider MIN and MAX operator is defined as below:

$$l_n = \min(a,b) = a \bullet b = a \cap b \text{ or } l_n = \max(a,b) = a + b = a \cup b \qquad (4.8)$$

Considering a 3 valued 2 variables MVL min operation, there exist a 3x3 matrix. If any l_n has a slant bonding of only 2 then there is a possible switch.

Table 4-1: 3x3 matrix of complemented l_n. arrows showing possible switch between the slant bondings

$\overline{l} = \overline{MIN(a,b)}$ a	0	1	2
b			
0	2(l_1)	2(l_2)	2(l_3)
1	2(l_4)	1(l_5)	1(l_6)
2	2(l_7)	1(l_8)	0(l_9)

Phase 1 is a given MVL function. Below in Table 4-2 an example of a 4x4 matrix for 4 valued 2 variable min operations. All 4 phases are shown by example as below:

Table 4-2: An example of 4x4 matrix MIN function

$MIN(a,b)$ a	0	1	2	3
b				
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

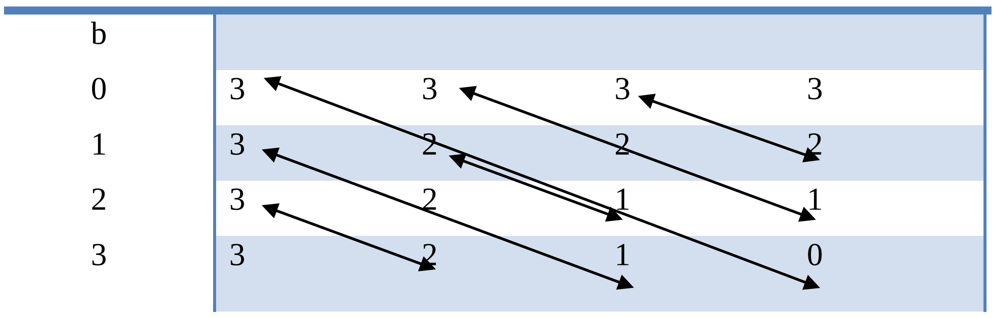
In Table 4-1, switch cannot be overlapped. For all l_n in the matrix, complement operation has to be applied before possible switching takes place. Below is a 3x3 matrix of complemented l_n. Arrows showing possible switch between the slant bondings: Phase 2 depicts by complementing all the minterms including "0" in the provided MVL function. Table 4-3 reflects on phase 2 implementation.

Table 4-3: Phase 2 of the inverse MIN or MAX operation

$\overline{l} = \overline{MIN(a,b)}$ a	0	1	2	3
b				
0	3	3	3	3
1	3	2	2	2
2	3	2	1	1
3	3	2	1	0

Table 4-4 shows Phase 3 implementation. All possible switching between the slant's bondings are shown by the arrow.

Table 4-4: Possible switching of the minterms shown by both ended arrows



Switch(l) a	0	1	2	3
b				
0	3	3	3	3
1	3	2	2	2
2	3	2	1	1
3	3	2	1	0

Phase 4 is the last phase of the inverse MIN or MAX operation. Table 4-5 represents final outcome as a max function which is obtained from a min function.

Table 4-5: Final phase 4 representing the outcome as the MAX function

$MAX(a,b)$ a	0	1	2	3
b				
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

Definition 4.2.6- An EXTENDED AND operator is defined as below:

$$ {}^{a}\vec{x}_1^{\,b_1,b_2} \bullet x_2 = \begin{cases} b_1 & if \quad x_2 = b_1 \, and \quad x_1 = a \\ b_2 & if \quad x_2 = b_2 \, and \quad x_1 = a \\ 0 & otherwise \end{cases} \tag{4.9} $$

Where $a_1, a_2, \cdots, a_n \in R$, $b_1, b_2, \cdots, b_n \in R$ and $0 \le \{a, b, x_1, x_2\} \le (y-1)$ and x_2 determines the value of the EXTENDED AND. The definition of the operator provides the opportunity of eliminating the usage of MIN operator twice in one logic expression. Thus in this thesis EXTENDED AND is extensively used during the synthesis of MVL functions.

WINDOW LITERAL or short literal is explained in ***definition 4.2.1***. Consider an implicant from a MVL function as ${}^{0}x_1^{\,1} \bullet {}^{1}x_2^{\,1} + {}^{0}x_1^{\,2} \bullet {}^{2}x_2^{\,2}$. The implicant can be used and simplified using MVL algebraic rules for the EXTENDED AND operator. Both of the WINDOW LITERALs ${}^{0}x_1^{\,1}$ and ${}^{0}x_1^{\,2}$ forms the implicant from the same column of the matrix of the MVL function. The simplified implicant can be represented as ${}^{0}\vec{x}_1^{\,1,2}({}^{1}x_2^{\,1} + {}^{2}x_2^{\,2})$. Considering coordinate $x_1 = 0$ and $x_2 = 2$ for the above mentioned implicant. Therefore the implicant would achieve the output as, ${}^{0}\vec{x}_1^{\,1,2}(0+2)$, ${}^{0}\vec{x}_1^{\,1,2} \bullet 2,\quad where\quad x_2 = 2 \therefore 2 \bullet 2 = 2$.

Definition 4.2.7- An assignment of values to variables where the following condition and rule satisfies is called a minterm. In an MVL matrix ${}^{a}x_1^{\,b}$ and ${}^{a}x_2^{\,b}$ represents the value of a minterm. ${}^{a}x_1^{\,b}$ and ${}^{a}x_2^{\,b}$ both of them are WINDOW LITERALs. WINDOW LITERALs help to form the representation of minterms with the assistance of coordinate x_1 and x_2. Table 4-6 represents a MVL function.

Table 4-6: Function f_1 representing a 3x3 matrix of a random MVL function

Function f_1	a	0	1	2
		Column 1	Column 2	Column 3
B				
0		0(l_1)	**2(l_2)**	**1(l_3)**
1		**1(l_4)**	0(l_5)	**2(l_6)**
2		**2(l_7)**	**1(l_8)**	0(l_9)

Considering coordinate $(x_1, x_2) = (2,1)$, the value of the minterm at this point is 2. This minterm is retrieved from column 2 of the MVL matrix. Since the minterm is 2, then *b* = 2. $x_1 = 2$ therefore, $a_1 = 2$ and $x_2 = 1$ therefore, $a_2 = 1$. So, representation could be ${}^{2}x_1^{\,2} \bullet {}^{1}x_2^{\,2}$. According to the ***definition 4.2.1*** of WINDOW LITERAL this could be represented as ${}^{2}x_1{}^{2} \bullet {}^{1}x_2{}^{2} = 2 \bullet 2$ or 2. Therefore an ideal minterm could be represented as min operation of two WINDOW LITERALs as ${}^{a_m}x_1{}^{b_n} \bullet {}^{a_m}x_2{}^{b_n}$, where, $a_1, a_2, \cdots, a_m \in R$, $b_1, b_2, \cdots, b_n \in R$ and $0 \le \{a, b, x_1, x_2\} \le (y-1)$ $a_m \ne a_m$ *and* $b_n = b_n$. Therefore,

$$ {}^{a_1}x_1{}^{b_1} = \begin{cases} b_1 \quad if \quad x_1 = a_1, \quad where \quad b_1 = \min term \quad value \\ \qquad 0 \quad otherwise \end{cases} \tag{4.10} $$

$$ {}^{a_2}x_2{}^{b_2} = \begin{cases} b \quad if \quad x_2 = a_2, \quad where \quad b_2 = \min term \quad value \\ \qquad 0 \quad otherwise \end{cases} \tag{4.11} $$

4.3 Algebraic Postulates

For notation purposes MVL variables are denoted as x_n or x_m. MV Logic constant are denoted with lower case alphabets such as *m*, *n* and normal alphabet such as *k*. The base of the digital representation is denoted by *y* with domain $R, R = \{0, 1, 2, \cdots, y-1\}$, where "*y*" is the radix. In MVL all identity and below mentioned postulates are represented by MIN and MAX operator. The "+" symbol always stands for the MAX operator. Mostly all the examples in this section of the thesis are performed considering *y* = 4 with domain $R \in \{0, 1, 2, 3\}$ and *y* = 3 with domain $R \in \{0, 1, 2\}$. Mainly four MVL algebraic postulates are being represented with their definitions. Distribution, Idempotent, Identity and De Morgan relations are among them. Proof of the postulates from each of the main four categories is provided within the definitions. Below four categories of postulates are presented,

Category 1: MVL Distribution is shown by the equation presented below,

$$x_n \cdot k + x_m \cdot k = k(x_n + x_m) \; x_m \cdot k + x_n \cdot k = k(x_m + x_n) \tag{4.12}$$

Category 2: MIN and MAX operators are used in the MVL Idempotent expressions. Equations representing MVL Idempotent are shown below,

$$x_n \bullet x_n = x_n \; (k \cdot x_n) \bullet (k \cdot x_n) = k \cdot x_n \tag{4.13}$$

$$\overline{x}_n \cdot \overline{x}_n = \overline{x}_n \; \overline{x}_n + \overline{x}_n = \overline{x}_n \; x_n + x_n = x_n \tag{4.14}$$

$$k \cdot x_n + k \cdot x_n = k \cdot x_n \; (k + x_n) \bullet (k + x_n) = k + x_n \tag{4.15}$$

Category 3: MVL identity for MIN and MAX operators are presented as below,

$$x + 0 = x \; \overline{x} + 0 = \overline{x} \tag{4.16}$$

$$x \bullet 0 = 0 \; \overline{x} \bullet 0 = 0 \tag{4.17}$$

Category 4: MIN, MAX operators are also used in MVL De Morgan postulates. The expressions representing the algebraic postulate is shown below,

$$\overline{\overline{x}_n + \overline{x}_m} = x_n \bullet x_m \; \overline{\overline{x}_n \bullet \overline{x}_m} = x_n + x_m \tag{4.18}$$

$$\overline{x_n + x_m} = \overline{x_n} \bullet \overline{x_m} \; \overline{x_n \bullet x_m} = \overline{x_n} + \overline{x_m} \tag{4.19}$$

Definition 4.3.1- MVL Distribution operates on WINDOW LITERAL ${}^a x^b$ and independent value *k*. The definition of WINDOW LITERAL is represented in ***definition 4.2.1***. *k* is an independent value whose range could be defined as $0 \le k \le (y-1)$ where

$k \in R$. The MIN and MAX operator (according to ***definition 2.1.1 and 2.1.2***) acts as an intermediate operator between short literal ${}^{a}x^{b}$ and *k*.

Considering y = 4, then $\{x,a,b\} \in \langle 0123 \rangle$. If a$_1$ = 2 and b$_1$ = 3, then ${}^{2}x_1{}^{3} = 3$. If a$_2$ = 1 and b$_2$ = 0, then ${}^{1}x_2{}^{0} = 0$. Now if $k = 1$, ${}^{2}x_1{}^{3} = 3$ and ${}^{1}x_2{}^{0} = 0$ then according to MVL Distribution,

$$ {}^{a}x_n{}^{b} \cdot k + {}^{a}x_m{}^{b} \cdot k = k({}^{a}x_n{}^{b} + {}^{a}x_m{}^{b}) \qquad \textit{where,} \qquad {}^{a}x_n{}^{b} \neq {}^{a}x_m{}^{b} \qquad \textit{and,} \tag{4.20}$$

$$ {}^{2}x_1{}^{3} \cdot 1 + {}^{1}x_2{}^{0} \cdot 1 = 1({}^{2}x_1{}^{3} + {}^{1}x_2{}^{0}) \tag{4.21}$$

$$ 3 \cdot 1 + 0 \cdot 1 = 1(3+0) \Rightarrow 1 + 1 = 1(3) \Rightarrow 1 = 1 \qquad [\Pr oved] \tag{4.22}$$

In the above proof, even if the ${}^{a}x_n{}^{b}$ and ${}^{a}x_m{}^{b}$ are switched their places, the same output would be achieved. In an MVL matrix, *k* could be different row of the same column. But in order to achieve the MVL Distribution law, *k* has to reside in the same column of the MVL matrix.

Definition 4.3.2- MVL Idempotent also operates on window literal ${}^{a}x^{b}$ and sometimes on independent value *k*, depending on the situation. If literal ${}^{a}x^{b}$ is related to independent value *k* with a MIN or MAX operation, then only MVL Idempotent operates on *k*. MVL Idempotent algebraic rule can only be applied if and only if ${}^{a_1}x_n{}^{b_1} = {}^{a_2}x_m{}^{b_2}$ because $n = m$ $a_1 = a_2$ *and* $b_1 = b_2$.

Considering y = 4, then $\{x,a,b\} \in \langle 0123 \rangle$. If a = 2 and b = 3, then ${}^{2}x^{3} = 3$. Now if $k = 1$ and ${}^{2}x^{3} = 3$, then according to MVL Distribution, $(k \cdot x_n) \bullet (k \cdot x_n) = k \cdot x_n$ *where,* ${}^{a}x_n{}^{b} = {}^{a}x_n{}^{b}$ *and,*

$$({}^{2}x_1^{3}\cdot 1)\bullet({}^{2}x_1^{3}\cdot 1)=1({}^{2}x_n^{3}\cdot {}^{2}x_1^{3}) \tag{4.23}$$

$$(3\cdot 1)\bullet(3\cdot 1)=1(3\cdot 3)\Rightarrow 1\bullet 1=1(3)\Rightarrow 1=1 \quad [\Pr oved] \tag{4.24}$$

Another above mentioned MVL Idempotent proof is shown below.

$$(k+\overline{x_n})\bullet(k+\overline{x_n})=k+\overline{x_n}\ where, \qquad {}^{a}x_n^{b}={}^{a}x_n^{b} \qquad and, \tag{4.25}$$

$$({}^{2}x_1^{3}+1)\bullet({}^{2}x_1^{3}+1)={}^{2}x_1^{3}+1 \tag{4.26}$$

$$(3+1)\bullet(3+1)=3+1\Rightarrow 3\bullet 3=3+1\Rightarrow 3=3 \quad [\Pr oved] \tag{4.27}$$

Definition 4.3.3- MVL Identity is only shown against the logical value “0”. Other logical values, such as (y-1)th values against MVL identity is described in the EXTENDED COMPLEMENT section of ***definition 4.2.4***. MVL identity $x\bullet 0=0$ is similar to $\overline{x}+0=\overline{x}$ and $\overline{x}+0=\overline{x}$ is similar to $x+0=x$. Hence below only two of the identities are shown.

$$x\bullet 0=0 \tag{4.28}$$

$$\overline{x}+0=\overline{x}\,;\ \overline{x}+0=\begin{cases}\overline{x} & if \quad 0\le x\le y-1\\ 0 & otherwise\end{cases} \tag{4.29}$$

Definition 4.3.4- MVL De Morgan’s algebraic postulate is the unique combination of De Morgan theorem and multi-valued logic. According to De Morgan theorem,

$$(A\cup B)^{c}=A^{c}\cup B^{c} \tag{4.30}$$

Representing conjunction with logical MIN operator and disjunction with logical MAX operator,

$$\overline{(A \bullet B)} = \overline{A} + \overline{B} \tag{4.31}$$

Now considering A and B both of them representing set of y radix multiple values, $A_1, A_2, \cdots, A_n \in R$ and $B_1, B_2, \cdots, B_n \in R$, where $0 \le \{A, B\} \le (y-1)$ .If set A is represented by x_n and set B is represented by x_m, then the following postulate could be derived,

$$\overline{\overline{x_n} + \overline{x_m}} = x_n \bullet x_m \tag{4.32}$$

Considering y = 4, then $\{x, a, b\} \in \langle 0123 \rangle$. If a_1 = 2 and b_1 = 3, then ${}^2x_1{}^3 = 3$. If a_2 = 1 and b_2 = 0, then ${}^1x_2{}^0 = 0$. Now if ${}^2x_1{}^3 = 3$ and ${}^1x_2{}^0 = 0$ then according to MVL De Morgan,

$$\overline{\overline{{}^2x_1{}^3} + \overline{{}^1x_2{}^0}} = {}^2x_1{}^3 \bullet {}^1x_2{}^0 \tag{4.33}$$

$$where, \qquad {}^ax_n{}^b \neq {}^ax_m{}^b \qquad or, \qquad {}^ax_n{}^b = {}^ax_m{}^b \tag{4.34}$$

$$\overline{\overline{3} + \overline{0}} = 3 \bullet 0 \Rightarrow \overline{0 + 3} = 3 \bullet 0 \Rightarrow \overline{3} = 3 \bullet 0 \Rightarrow 0 = 0 \qquad [\mathrm{Pr}\,oved] \tag{4.35}$$

Similarly other MVL De Morgan postulates such as, $\overline{x_n \bullet x_m} = \overline{x_n} + \overline{x_m}$, $\overline{\overline{x_n} \bullet \overline{x_m}} = x_n + x_m$ and $\overline{x_n + x_m} = \overline{x_n} \bullet \overline{x_m}$ also can be proved through substitution of MVL values.

4.4 Summary

The main drawback tends to make us realize the fact, that no universal set of logic operators or algebraic postulates has been standardized. To overcome this issue, an initiative was undertaken to generalize a set of universal logic operators and postulates. The proposed logic operators can be used in various applications. The logic operators were proven for MVL system. ODD and EVEN operator can be used for error detecting codes in MVL serial communications. EXTENDED AND operator reduced PT in each MVL function representation. Other logic operators were also essential for efficient logic synthesis. Proposed postulates were existing binary algebraic postulates which were used for better synthesized expressions in MVL system. Furthermore, MVL universal operators can be also used in different approaches for synthesizing multi-level functions.

Chapter 5

High Deduction Algorithm

In this chapter, High Deduction Algorithm (HDA-MVL) is presented to synthesize Multi-Valued Logic (MVL) functions at an optimal level. It utilizes the proposed postulate and logical operators for the synthesis and realization of MVL functions.

5.1 Introduction

Many heuristic and evolutionary algorithms had been proposed in the literature for synthesis of MVL functions. Evolutionary and combined heuristic algorithms benefits in terms of efficient functional synthesis but requires much computational complexity, thus requires extensive amount of time to generate synthesized functions. Most of the existing efficient algorithms are computationally complex and requires enormous amount of time for synthesizing.

In this section of the thesis, HDA-MVL algorithm synthesizes and extracts the expression for each MVL function. The advantages of HDA-MVL algorithm is demonstrated with reduced function representation. Furthermore these synthesized expressions are realized with digital circuit to compare the feasibility of the experiment in the later section of this thesis.

Experimental analysis of HDA-MVL algorithm is based on synthesizing a benchmark of 50000 sequentially generated 4 valued 2 variable functions. The obtained results has shown that the average number of Product Term (PT) needed to synthesize each MVL function using HDA-MVL outperforms conventional DC based heuristics and evolutionary techniques To the extent of literature done, the proposed method shows that the number of gate count in digital circuitry reduces remarkably using HDA-MVL algorithm.

5.2 HDA-MVL Algebraic Synthesis

HDA algorithm is a Direct Cover (DC) based approach. This algorithm only considers non-zero minterm for implication selection. The main steps for representing HDA techniques for synthesis of MVL functions are provided below:

(1) Choosing non-zero minterm from the function.
(2) Identify each column of the function as an effective implicant that covers all the non-zero minterms.
(3) Obtain a reduced function by applying algebraic manipulation.
(4) Repeat steps 1 to 3 until no more minterms and columns remain uncovered.

Firstly, the algorithm converts a MVL function into a matrix. Then, the algorithm determines all the non-zero minterm from 1st column to nth column. Each column of the matrix represents an effective implicant of HDA-MVL synthesis method. Each of the implicant is formed with one effective min operation. The steps stated above are important because they not only cover an entire function but also provide lesser PT to represent it.

Assume an implicant is composed of ${}^{0}x_1{}^{1} \bullet {}^{1}x_2{}^{1} + {}^{0}x_1{}^{2} \bullet {}^{2}x_2{}^{2}$, from a MVL function. Also, assume the implicant is derived from column one. By using the algebraic law of

MVL distribution, derivation of ${}^{0}\vec{x}_1^{1,2}({}^{1}x_2^{1} + {}^{2}x_2^{2})$ is made. This expression can be further simplified using EXTENDED AND operator. After simplifying all the implicants, MAX operator operates on all the implicants to prompt the final output *y*.

5.2.1 Minterm and implicant selection

According to the key steps of HDA-MVL algorithm, the minterm selection criterion is specifically dependable on all the non-zero minterm from 1st column to nth column. Each of the columns represents an implicant, if and only if the column contains at least one non-zero minterm. Hence an implicant can consists of any non-zero minterm for a specific column. The algorithm reads each index of the matrix and determines the end of a column. Each of the minterms of a specified column consists of window literal ${}^{a}x_1^{b}$ and ${}^{a}x_2^{b}$.

Table 5-1: Function f_1 representing a 3x3 matrix of a random MVL function

Function f_1	a	0	1	2
		Column 1	Column 2	Column 3
B				
0		0(l_1)	**2(l_2)**	**1(l_3)**
1		**1(l_4)**	0(l_5)	**2(l_6)**
2		**2(l_7)**	**1(l_8)**	0(l_9)

These WINDOW LITERALs represent the coordinate of the matrix for any specific minterm. The WINDOW LITERALs ${}^{a}x_1^{b}$ and ${}^{a}x_2^{b}$ are always relating to each other forming a MIN relation between them. All the minterms for any column are related together forming a MAX relation between them. The consideration of MVL function $f_1$is made as explained in Table 5-1 above, where y = 3 and a, b are the coordinate of the minterm l_n.

In the first column of the functionf_1, bold minterm 1 and 2 generates the implicant. Similarly in the second column, bold minterm 2 and 1 and in the third column, minterm 1 and 2 generates the implicants consecutively. The following Table5-2 shows minterm representation for each coordinate of the MVL matrix and implicant $I(x_1, x_2)$ for each column.

Table 5-2: Extracting minterm and implicant from function f_1

Function f_1	x_1, x_2	y	Minterm	Implicant $I(x_1, x_2)$
Column				
1st	0, 0	0	-	
1st	0, 1	1	${}^0x_1^1 \bullet {}^1x_2^1$	${}^0x_1^1 \bullet {}^1x_2^1 + {}^0x_1^2 \bullet {}^2x_2^2$
1st	0, 2	2	${}^0x_1^2 \bullet {}^2x_2^2$	
2nd	1, 0	2	${}^1x_1^2 \bullet {}^0x_2^2$	
2nd	1,1	0	-	${}^1x_1^2 \bullet {}^0x_2^2 + {}^1x_1^1 \bullet {}^2x_2^1$
2nd	1, 2	1	${}^1x_1^1 \bullet {}^2x_2^1$	
3rd	2, 0	1	${}^2x_1^1 \bullet {}^0x_2^1$	
3rd	2,1	2	${}^2x_1^2 \bullet {}^1x_2^2$	${}^2x_1^1 \bullet {}^0x_2^1 + {}^2x_1^2 \bullet {}^1x_2^2$
3rd	2, 2	0	-	

Each of the implicant is related together forming a MAX relation between each other. ***Definition 4.2.1*** shows how EXTENDED AND operator can be used in the synthesis of

MVL functions or to be more precise, implicants. Below is an example of how EXTENDED AND operator functions on each implicant from different column.

$$f_1 = {}^0x_1^{\,1} \bullet {}^1x_2^{\,1} + {}^0x_1^{\,2} \bullet {}^2x_2^{\,2} + {}^1x_1^{\,2} \bullet {}^0x_2^{\,2} + {}^1x_1^{\,1} \bullet {}^2x_2^{\,1} + {}^2x_1^{\,1} \bullet {}^0x_2^{\,1} + {}^2x_1^{\,2} \bullet {}^1x_2^{\,2} \qquad (5.1)$$

$$Synthesized(f_1) = {}^0\vec{x}_1^{\,1,2}({}^1x_2^{\,1} + {}^2x_2^{\,2}) + {}^1\vec{x}_1^{\,2,1}({}^0x_2^{\,2} + {}^2x_2^{\,1}) + {}^2\vec{x}_1^{\,1,2}({}^0x_2^{\,1} + {}^1x_2^{\,2}) \qquad (5.2)$$

Table 5-3: Output *y* is checked using the synthesized expression from function *f*1

*f*1	x_1, x_2	Synthesized Expression	y
Col.		${}^0\vec{x}_1^{\,1,2}({}^1x_2^{\,1} + {}^2x_2^{\,2}) + {}^1\vec{x}_1^{\,2,1}({}^0x_2^{\,2} + {}^2x_2^{\,1}) + {}^2\vec{x}_1^{\,1,2}({}^0x_2^{\,1} + {}^1x_2^{\,2})$	
1st	0,0	$\langle 1,2\rangle(0+0)+\langle 0,0\rangle(2+0)+\langle 0,0\rangle(1+0) \Rightarrow \langle 1,2\rangle \bullet 0 + \langle 0,0\rangle \bullet 2 + \langle 0,0\rangle \bullet 1 \Rightarrow 0+0+0$	0
1st	0,1	$\langle 1,2\rangle(1+0)+\langle 0,0\rangle(0+0)+\langle 0,0\rangle(0+2) \Rightarrow \langle 1,2\rangle \bullet 1 + \langle 0,0\rangle \bullet 0 + \langle 0,0\rangle \bullet 2 \Rightarrow 1+0+0$	1
1st	0,2	$\langle 1,2\rangle(0+2)+\langle 0,0\rangle(0+1)+\langle 0,0\rangle(0+0) \Rightarrow \langle 1,2\rangle \bullet 2 + \langle 0,0\rangle \bullet 1 + \langle 0,0\rangle \bullet 0 \Rightarrow 2+0+0$	2
2nd	1,0	$\langle 0,0\rangle(0+0)+\langle 2,1\rangle(2+0)+\langle 0,0\rangle(1+0) \Rightarrow \langle 0,0\rangle \bullet 0 + \langle 2,1\rangle \bullet 2 + \langle 0,0\rangle \bullet 1 \Rightarrow 0+2+0$	2
2nd	1,1	$\langle 0,0\rangle(1+0)+\langle 2,1\rangle(0+0)+\langle 0,0\rangle(0+2) \Rightarrow \langle 0,0\rangle \bullet 1 + \langle 2,1\rangle \bullet 0 + \langle 0,0\rangle \bullet 2 \Rightarrow 0+0+0$	0
2nd	1,2	$\langle 0,0\rangle(0+2)+\langle 2,1\rangle(0+1)+\langle 0,0\rangle(0+0) \Rightarrow \langle 0,0\rangle \bullet 2 + \langle 2,1\rangle \bullet 1 + \langle 0,0\rangle \bullet 0 \Rightarrow 0+1+0$	1
3rd	2,0	$\langle 0,0\rangle(0+0)+\langle 0,0\rangle(2+0)+\langle 1,2\rangle(1+0) \Rightarrow \langle 0,0\rangle \bullet 0 + \langle 0,0\rangle \bullet 2 + \langle 1,2\rangle \bullet 1 \Rightarrow 0+0+1$	1
3rd	2,1	$\langle 0,0\rangle(1+0)+\langle 0,0\rangle(0+0)+\langle 1,2\rangle(0+2) \Rightarrow \langle 0,0\rangle \bullet 1 + \langle 0,0\rangle \bullet 0 + \langle 1,2\rangle \bullet 2 \Rightarrow 0+0+2$	2
3rd	2,2	$\langle 0,0\rangle(0+2)+\langle 0,0\rangle(0+1)+\langle 1,2\rangle(0+0) \Rightarrow \langle 0,0\rangle \bullet 2 + \langle 0,0\rangle \bullet 1 + \langle 1,2\rangle \bullet 0 \Rightarrow 0+0+0$	0

Table 5-3 represents the synthesized expression for each column and breaks down the expression to evaluate the output. The curly bracket "{ }" covering part of the expression in each column actually showing the active EXTENDED AND operator for the implicant. The output *y* represents each minterm from the function *f1*. After the synthesized expression has been evaluated the values are definitely matching with the output *y*, which proves that the expression is correct.

5.3 Algorithm: HDA-MVL Algebraic Synthesis

Based on the rough structure of the HDA-MVL a pseudo code has been prepared. Later this pseudo code was converted to JAVA high level programming language. Below in Figure 5.1 the entire pseudo code for the algorithm is represented.

```
Do
    Generate Sequence of all possible nth variable 2 input combinations
    Locate each index of respective sequence using array
    Check minterms in each of the sequence
    oneMinterm = checkOneMinterm(func_Sequence)
    twoMinterm = checkTwoMinterm(func_Sequence)
    .... nthMinterm = checkNthMinterm(func_Sequence)
    for each sequence
    Decomposed each PT is decomposed into → (x,y) coordinate
    Sort all PTs in the array sequence
    save  primary term sequences
    for each minterm coordinate (x,y) of primary term
    save coordinate (x,y)
```

```
check 1st term (primary term) with other terms (secondary term)

nextColumn():
for each PT of the MVL Function
foreach column match
jump to next column
save  secondary term sequencesu
for each minterm coordinate (x,y) of secondary term save coordinate
(x,y)
if secondary term = primary term
update the result
else if secondary term != primary term
gotonextColumn();
if primary term → last term
update result
ifcheckLastTerm=true
        do nothing
else ifcheckLastTerm = false
        check secondary term

do
checkLastTerm();
while(countMinTerm<finish.length())
getReducedMinterm();
        getAvgPTResults();
close file
while(full_Column != true)
end
```

Figure 5.1: HDA-MVL pseudo code

5.3.1 Calculating average PT results

The pseudo code in Figure 5.2 below generates the result of average PT. Considering an example of 500 combinations for 16 minterm. The average PT needed would be calculated based on the synthesis method.

```
getAvgPTResults():
        X denotes from minterm 16 to minterm 1 in different functions
        Y denotes limitation of functions per minterm combinations
        iffunctions_X<Y &&mintermCount==X
                function_array_X[functions_X] = h
                incrementfunctions_X}
        ifminterm counter “functions_X” reaches their limit
                checkMIN = false
        call next function to check next sequence
        nextSeq = callNextSequence(nextSeq, reference)
        do
        function_X_Total = function_X_Total + function_array_X[total_PT];
        total_PT++;
        while(total_PT<functions_X)

        function_X_Avg = (double)function_X_Total / total_PT;
        save results
```

Figure 5.2: Pseudo code generating average PT used per minterm combination of the synthesized function

5.4 Simulation Results

Benchmark digital circuit have been developed for 4 valued 2 variable MVL functions. In this section, a benchmark consisting of 50000 sequentially generated 4 valued 2 variable functions has been generated and synthesized using HDA-MVL Algorithm. Table 5-4 provides comparison of the proposed HDA-MVL and ACO-MVL. Mostafa Abd-El-Barr in his paper synthesized MVL functions of 4 valued 2 variable using evolutionary techniques (M. Abd-El-Barr 2012). He has implemented Ant Colony

Optimization (ACO-MVL) algorithm with a benchmark consisting of 50000 randomly generated 4 valued 2 variables. HDA-MVL sequentially generates MVL functions, not randomly. The comparison is shown below.

Table 5-4: Comparison of proposed algorithm with existing evolutionary and direct cover algorithms (Abd-El-Barr and Sarif 2006; Sarif and Abd-El-Barr 2006)

Min-term	**Functions (Randomly Generated)**	**ARM**	**BS**	**DM**	**ODC**	**WDC**	**ACO-MVL**	**Functions (Sequential Generated)**	**HDA-MVL (proposed)**
16	500	7.59	7.56	7.00	7.08	7.01	6.73	500	4.00
15	2679	8.29	8.30	7.51	7.48	7.50	7.05	2679	4.00
14	6589	8.35	8.40	7.56	7.51	7.55	7.16	6589	4.00
13	10585	8.27	8.35	7.54	7.49	7.54	7.18	10585	4.00
12	11230	8.04	8.09	7.38	7.33	7.38	7.08	11230	3.00
11	9003	7.70	7.75	7.12	7.08	7.13	6.90	9003	3.00
10	5434	7.32	7.36	6.83	6.78	6.83	6.66	5434	3.00
9	2575	6.87	6.87	6.47	6.43	6.47	6.35	2575	3.00
8	1038	6.30	6.32	6.02	5.97	6.02	5.93	1038	2.00
7	277	5.72	5.75	5.52	5.48	5.52	5.48	277	2.00
6	75	5.13	5.14	4.97	4.96	4.98	4.96	75	2.00
5	13	4.00	4.00	4.00	3.92	4.00	3.92	13	2.00
4	1	4.00	4.00	4.00	4.00	4.00	4.00	1	1.00
3	1	3.00	3.00	3.00	3.00	3.00	3.00	1	1.00

It was clearly observed that the proposed HDA-MVL algorithm outperformed all other techniques, regardless the number of minterms generated by each MVL function. The lowest number of product term was used by the proposed algorithm. The worst case scenario for ACO-MVL was 13 minterm where the average PT usage was 7.18.

In the same scenario the proposed HDA-MVL algorithm required on an average of 4 PT, which was almost the half of what was required by ACO-MVL. For minterm 3 and 4, both of the algorithm output similar result. It was observed that this huge difference represented a substantial improvement. HDA-MVL generated sequential sample functions were a representation of a population which consists of 4^{16} functions. A graphical comparison of the HDA-MVL against other algorithm can be overseen by the Figure5.3.

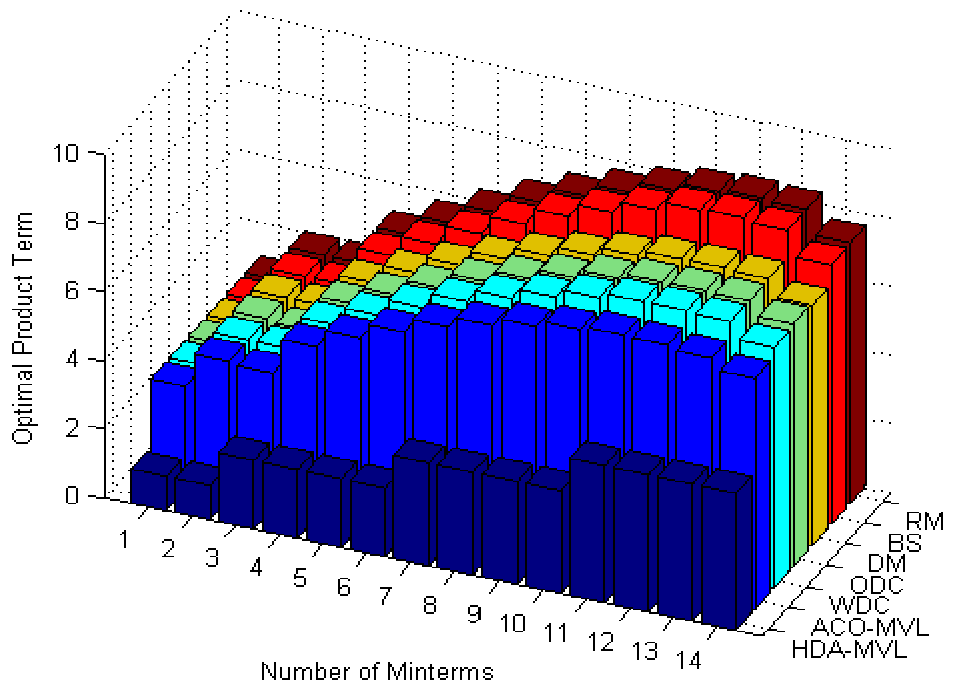


Figure 5.3: Graphical comparison of HDA-MVL algorithm and other existing algorithms, representation of PT used per minterm combination

The overall average PT achieved by DC based algorithms and the evolutionary heuristic based techniques are shown in the Table 5-5. The tabulation illustrates that the proposed HDA-MVL algorithm outperformed all DC based algorithms. Among all the existing algorithms, the proposed HDA-MVL algorithm achieved better results in terms of average product term needed to synthesize a given MVL function.

Table 5-5: Overall PT reduction using different algorithms (Sarif and Abd-El-Barr 2006; Abd-El-Barr and Sarif 2006)

Algorithm	**ARM**	**BS**	**DM**	**ODC**	**WDC**	**VGAMVL**	**RVGA-MVL**	**ACO-MVL**	**HDA-MVL (proposed)**
PT	7.89	7.94	7.25	7.20	7.25	7.18	7.13	6.96	3.00

5.5 Summary

Various DC based algorithms were proposed in the field of MVL synthesis. Even though DC based algorithm is time consuming, they are more accurate in term of retrieving logic expressions. In this chapter, HDA-MVL algorithm has been proposed which uses the postulates and logical operators to perform MVL synthesis. The results obtained using the HDA algorithm is compared using a benchmark consisting of 50K 4-valued 2-variable sequentially generated functions. The results achieved have shown that the HDA-MVL algorithm outperforms the conventional DC and evolutionary based techniques. In the experiment analysis it is observed that overall average PT reduction achieved by HDA-MVL algorithm is 56.88%. .HDA-MVL algorithm achieved 45.46% more success in reducing PT over evolutionary ACO-MVL algorithm.

Chapter 6

Multi-Valued Logic Neural Network Operators

A novel Neural Network Deployment Algorithm (NNDA-MVL) has been presented to show how this algorithm trains Multi-Valued Logic (MVL) operators. In this chapter of the thesis, main steps for NNDA is presented and shown how this algorithm benefits in producing trained neural operators. The algorithm is equipped with back-propagation learning capability and novel MVL operators.

6.1 Introduction

One of the synthesizing techniques for MVL functions is the Neural Network approach. This approach can be further categorized into few subsections. Hsu et al. explained in detail about the three-valued neural logic network (Hsu, et al. 1990). The lack of any learning algorithm for MVL networks is an important restraint to many applications.

Hence, in this section the novel NNDA-MVL algorithm focus on training MVL neural operators with higher accuracy and synthesis of MVL functions with trained operators. Firstly the algebraic synthesis is carried out to extract the synthesized expression of the MVL function by the EXTENDED AND operator. Each of the

EXTENDED AND, MIN and MAX MVL operators are trained with separate neural networks consisting 4 hidden layer neurons.

Synthesized expressions are realized with the MVL neural operators to ensure the feasibility of the experiment. To the extent of literature done, the proposed method shows that higher accuracy is achieved with lesser training period using NNDA-MVL algorithm. The proposed method depicts that MVL neural network is a novel approach to train novel MVL operators and utilize them for efficient logic synthesis as an application.

6.2 Neural Architecture for MVL Operators

The neural architecture has three basic layers. First layer of neurons is called the input layer. The second layer of neurons which takes the input is called the hidden layer. The third layer of neurons which takes the input from the hidden layers is called the output layer. Feedforward backpropagation algorithm is applied to multiple layers of neurons. The output layer and the input layer are separated by the hidden layer.

There are maximum five different inputs for the input layer. Unlike other MVL operators, EXTENDED AND can reach up to a maximum of five different inputs for the input layer. Depending on the MVL operator the number of inputs varies. The rth column of each input is represented as a column vector $\overrightarrow{x_n}$ which varies between 0, 1, 2 and 3. The column vector consists of real entities where, $x \in R, x \geq 0, 0 < n \leq 5$ and $\overrightarrow{x_n} \in \{x_0, x_1 \ldots x_4\}$. The column vector can be represented as $\overrightarrow{x_n} \in [x_0, x_1 \ldots x_4]^T$ (Matsumoto, Ueda and Nomoto 2000; Zheng, Ishizuka and Tanno 1995).

Five different input column vectors are represented as below (Ngom and Simovici 2004; Zheng, Cao and Ishizuka 1998). Each of these input vectors represent the input layer of the neural network as $\overrightarrow{x_0} = [x_0, x_1 \ldots x_r]^T; \overrightarrow{x_1} = [x_0, x_1 \ldots x_r]^T; \overrightarrow{x_2} = [x_0, x_1 \ldots x_r]^T; \overrightarrow{x_3} = [x_0, x_1 \ldots x_r]^T \, and \, \overrightarrow{x_4} = [x_0, x_1 \ldots x_r]^T$.

The weights are distributed evenly with each of the inputs. The rth column of each synaptic weight is represented as a column weight vector $\overrightarrow{w_n}$ which varies between -1 to 1. The weight vector consists of real entities where, $w \in R, -1 \le w \le 1, n \le r$ and $\overrightarrow{w_n} \in \{w_0, w_1 \ldots w_r\}$. Weight vector can be represented as below,

$$\overrightarrow{w_n} \in [w_0, w_1 \ldots w_r]^T \tag{6.1}$$

The corresponding weights for each of the inputs are multiplied with their respective inputs. The summation of the weights multiplied by the inputs could be represented as below,

$$\sum_{i=0}^{r} w_i \bullet x_i = w_0 \bullet x_0, w_1 \bullet x_1 \ldots w_r \bullet x_r \tag{6.2}$$

Where, $\overrightarrow{w_n} \bullet \overrightarrow{x_n} \in [w_0 \bullet x_0, w_1 \bullet x_1 \ldots w_r \bullet x_r]$. The bias b is scalar. The bias is added to the summation of weight and input to adjust the output from the neuron. Bias b is similar to weight, but the only difference between bias and weight is that the bias has a constant input of 1. The transfer function φ can be different in functionalities depending on the desired output. The hyperbolic tangent sigmoid transfer function $\varphi(v)$ is presented in equation (6.3).

The characteristics of tangent sigmoid function and a basic neuron unit can be observed from the Figure 6.1and 6.2 below.

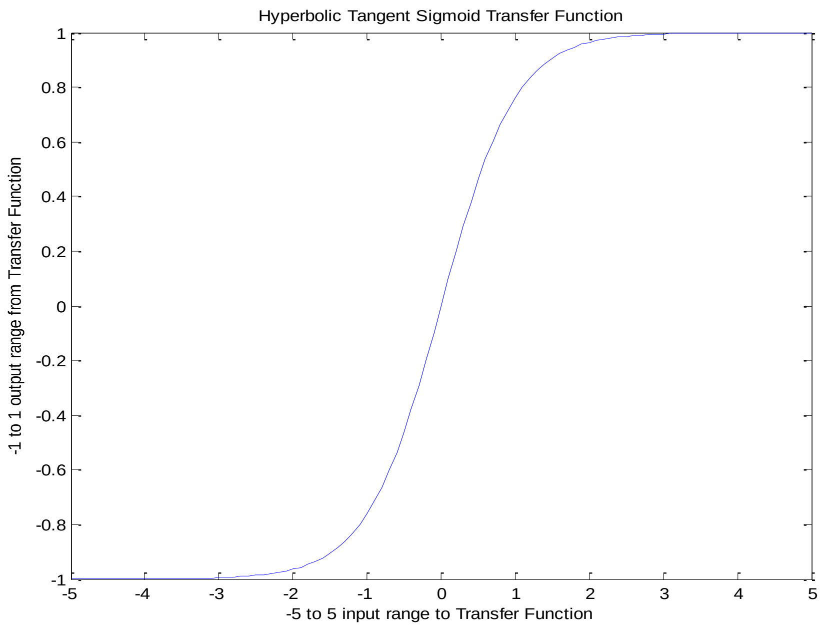


Figure 6.1: Hyperbolic tangent sigmoid function

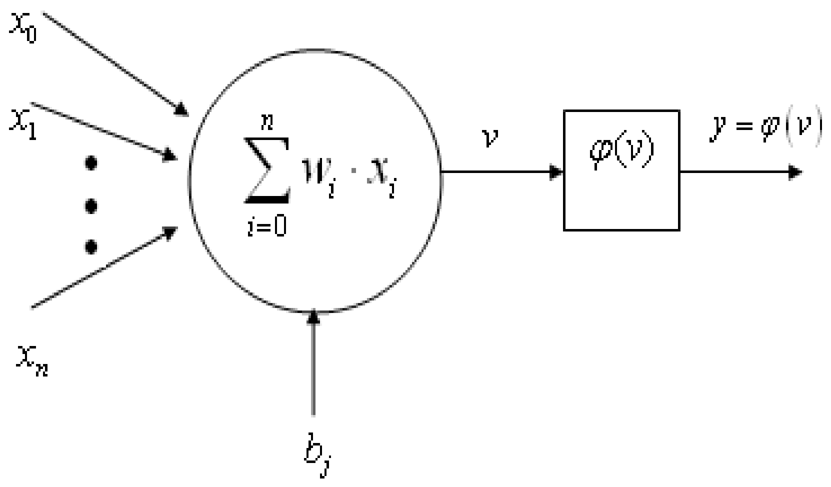


Figure 6.2: Basic neuron structure characteristics

The hyperbolic tangent sigmoid transfer function can be represented as below,

$$\varphi(v)=\frac{2}{1+e^{-2\cdot n}}-1=\frac{1-e^{-2\cdot n}}{1+e^{-2\cdot n}}=\frac{e^{2\cdot n}-1}{e^{2\cdot n}+1};\ \ [where\ \ \ n=v] \tag{6.3}$$

$\varphi(v)$ is a tangent sigmoid transfer function, where $v=\overrightarrow{w_n}\bullet\overrightarrow{x_n}+b$. The output of the neuron can be shown as below,

$$N_{Output}=\varphi\left(\overrightarrow{w_n}\bullet\overrightarrow{x_n}+b_n\right)=\varphi\left(\left[\sum_{i=0}^{r}w_i\bullet x_i\right]+b_i\right) \tag{6.4}$$

The transfer function tangent sigmoid has a very important property such that the derivative of the function can be presented as below,

$$\frac{d}{dv}\varphi(v)=\frac{2\bullet e^{2\cdot n}}{e^{2\cdot n}+1}-\frac{\left(2\bullet e^{2\cdot n}\right)\bullet\left(e^{2\cdot n}-1\right)}{\left(e^{2\cdot n}+1\right)^2}\qquad OR \tag{6.5}$$

$$\frac{d}{dv}\varphi(v)=\frac{4}{\left(e^{2\cdot n}\right)\left[\frac{1}{\left(e^{2\cdot n}+1\right)}\right]^2} \tag{6.6}$$

Hence the summarized derivative can be represented in terms of the transfer function itself as below,

$$\frac{d}{dv}\varphi(v)=1-\varphi(v)^2 \tag{6.7}$$

The output layer neuron has also the same transfer function which is hyperbolic tangent sigmoid. The network layer is represented by β. Therefore total number of inputs in the input layer can be identified by β_{input}. Similarly total number of hidden and output neurons from hidden and output layer can be identified by β_{hidden} and β_{output}. The neural network's weight is presented by ω.

As mentioned previously, all the inputs have their own corresponding weights. Therefore, weights from inputs to hidden neurons can be represented as ω_{ij}^{H}, where "*H*" represents the hidden layer, "*i*" represents the sequence of inputs, "*j*" represents the sequence of hidden neurons and $j \in [0,1\ldots\beta_{hidden}]$. b_j represents the bias associated with the respective neuron. Weights from hidden neurons to output neurons can be represented as ω_{ij}^{O}, where "*O*" represents the output layer, "*i*" represents the sequence of hidden neurons, "*j*" represents the sequence of output neurons and $j \in \left[0,1\ldots\beta_{output}\right]$ .Inputs coming from the input layer associated with respective weight and bias can be represented by the equation below,

$$\beta_j^H = \sum_{i=0}^{\beta_{input}} \left(\omega_{ij}^H \bullet x_i\right) + b_j \tag{6.8}$$

The hidden layer output for respective hidden neurons can be represented as below,

$$y_j^H = \varphi\left(\beta_j^H\right) \tag{6.9}$$

Similarly hidden layer input and output layer output can be represented as shown below,

$$\beta_j^O = \sum_{i=0}^{\beta_{hidden}} \left(\omega_{ij}^O \bullet y_i^H\right) + b_j \tag{6.10}$$

$$y_j^O = \varphi\left(\beta_j^O\right) \tag{6.11}$$

The learning function calculates the weight change Δw. The previous weight change $\Delta w_{previous}$ is stored inside learning state L_{state}. The learning rate L_{state} is initially set to 0.01 and $M_{constant}$ is set to 0.90. Both the learning rate and the momentum constant values can be defined according to the need of the learning capability of the neural net. The neuron's input vector $\overrightarrow{x_n}$, the weight vector $\overrightarrow{w_n}$ or bias b_j, error E, momentum constant $M_{cons\tan t}$ and learning rate L_{rate} are considered in equation representation. Momentum weight and bias learning function are key factors in gradient descent. The gradient can be represented as below,

$$\Delta w = M_c \bullet \Delta w_{previous} + (1 - M_c) \bullet L_{rate} \bullet w_{gradient} \qquad (6.12)$$

On the other hand, mean square error (*mse*) function is used for performance evaluation. This is the *mse* function. The network is provided with an input vector. The estimated output is determined based on the inputs. The network consists of a target. The actual output is subtracted by the expected output. This way the error is calculated. The *mse* is dependent on the subtraction value generated from the network. Assume $\hat{O}$ is the vector of predicted r output and O is the vector of actual output. The predicted output can be represented by the equation below,

$$mse = \frac{1}{r} \sum_{i=0}^{r} \left(\hat{O}_i - O_i \right)^2 \qquad (6.13)$$

The training function used in this architecture updates the weight and bias values according to Levenberg-Marquardt optimization (Demuth, Beale and Heagen 2010).

Levenberg-Marquardt backpropagation is chosen as it is a high performance supervised algorithm. This algorithm speeds up the training without computing the Hessian matrix. When the *mse* function develops the form of sum of squares, then the Hessian matrix and the gradient can be expressed as below,

$$H = J^T \bullet J \qquad and \qquad g = J^T \bullet e \tag{6.14}$$

Jacobian matrix is expressed with J. With respect to the weights $\overrightarrow{w_n}$ and biases b_j, J contains the first derivative of the network errors e, where e is the vector of network errors (Demuth, Beale and Heagen 2010). The Jacobian matrix can be shown as below,

$$J = \begin{bmatrix} \frac{\partial \cdot e_0}{\partial \cdot w_0} & \frac{\partial \cdot e_0}{\partial \cdot w_1} & \cdots & \frac{\partial \cdot e_0}{\partial \cdot w_m} \\ \frac{\partial \cdot e_1}{\partial \cdot w_0} & \frac{\partial \cdot e_1}{\partial \cdot w_1} & \cdots & \frac{\partial \cdot e_1}{\partial \cdot w_m} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial \cdot e_n}{\partial \cdot u_0} & \frac{\partial \cdot e_n}{\partial \cdot w_1} & \cdots & \frac{\partial \cdot e_n}{\partial \cdot w_m} \end{bmatrix} \tag{6.15}$$

Training weights exist in Gauss Newton method. The equation is presented as below,

$$G_{NM} = x_{r+1} = x_k - \frac{1}{\left[J^T \bullet J\right]} J^T \bullet e \tag{6.16}$$

Levenberg-Marquardt modification to Gauss Newton method is shown below,

$$L_{MM} = x_{r+1} = x_k - \frac{1}{\left[J^T \bullet J + \mu I\right]} J^T \bullet e \tag{6.17}$$

When μ is large this becomes gradient descent with a small step size. Newton's method is not only faster moreover it has more accuracy with a near minimum error. Thus μ is decreased after each successful step and it is increased only when an uncertain step increases the *mse* function. This way the *mse* is reduced for each iteration.

6.3 Neural Operator and Minterm Representation

In this section of the thesis, the functioning of neural logic operators and NNDA algorithm has been demonstrated briefly as an overview on the process of function realization. NNDA algorithm utilizes proposed MVL operators to construct neural net and train. Composed of series of trained neural networks such as, EXTENDED AND, MIN and MAX operators are used to realize the MVL functions. Minterm representation is the key to logic expression reduction. Hence, minterm representation is also discussed in brief. Trained MVL neural operators are used to represent minterms.

An assignment of values to variables where the following condition and rule satisfies is called a minterm. In an MVL matrix ${}^{a}x_1{}^{b}$ and ${}^{a}x_2{}^{b}$ represents the value of a minterm. ${}^{a}x_1{}^{b}$ and ${}^{a}x_2{}^{b}$ both of them are WINDOW LITERALs. They help to form the representation of minterms, with the assistance of coordinate x_1 and x_2. A three valued two variable MVL function is shown in Table 6-1.

Table 6-1: MVL function representing a 3x3 matrix of a random MVL function

MVL Function *f*₁	**Columns**[a]		
	0	***1***	***2***
0	0	2	1
1	1	0	2
2	2	1	0

Columns are denoted by a and Rows are denoted by b[*]

6.4 Key Steps for Neural NNDA

The key steps are the guidelines for the deterministic NNDA algorithm. Based on the steps, the entire algorithm is formed in the later section of this thesis. NNDA not only reduces the Product Term (PT) but also reduces the neural net block for representing each of the MVL functions. NNDA converts a MVL function into a matrix. Then the algorithm finds each column and their respective row's for any non-zero minterms. The algorithm determines all the non-zero minterm from 1st column to nth column. The main steps for representing NNDA techniques for synthesis of MVL functions are given as below,

1. Choose each column consisting window literal ${}^{a}x_1{}^{b}$.
2. For each column of ${}^{a}x_1{}^{b}$ find each related row consisting of window literal ${}^{a}x_2{}^{b}$
3. For each of $\left({}^{a}x_1{}^{b}, {}^{a}x_2{}^{b}\right)$ relations find the relations which consists non-zero minterm of the specified column.

4. Each column is related with other column by a MAX logic operator.
5. Realize each of the respective relations in terms of MIN and extended AND MVL operator.
6. Repeat steps 1 to 5 until no more columns and non-zero minterms remain uncovered.
7. Create separate feedforward backpropagation neural network architecture for the logic operators.
8. Train each of the neural network's with extended AND and MIN relations of the specified column.
9. Train neural network with MAX operator for relating every two columns.
10. Obtain a reduced function by applying trained neural network.
11. Repeat steps 6 to 8 until the neural net performance function "mse" is less than 0.01to obtain a directly synthesized MVL function.

Each column relations can be represented by $R_0\left({}^{a}x_{1,0}{}^{b},\left[{}^{a}x_{2,0}{}^{b},{}^{a}x_{2,1}{}^{b}\cdots{}^{a}x_{2,n}{}^{b}\right]\right)$ where R_0 represents the first column relation of the matrix, ${}^{a}x_{1,0}{}^{b}$ denotes the first column of window literal ${}^{a}x_{1}{}^{b}$, ${}^{a}x_{2,0}{}^{b}$ represents the first row of window literal ${}^{a}x_{2}{}^{b}$. For each column the respective related rows can reach up to $\left[{}^{a}x_{2,0}{}^{b},{}^{a}x_{2,1}{}^{b}\cdots{}^{a}x_{2,n}{}^{b}\right]$. Each column of the matrix represents one or more effective implicants of NNDA-MVL synthesis method. Each of the implicant is formed of different number of extended AND and MIN operators. Below the equation for predicting number of logic operator $ExAND^{C}{}_{number}$ per column,

$$ExAND^{C}{}_{number} = \begin{cases} floor\left(\dfrac{\min term_{non-zero}}{2}\right) \\ 0 \quad otherwise \end{cases} \tag{6.18}$$

Total number of the logic operator for any MVL function can be expressed by the equation below by $ExAND^{T}{}_{number}$,

$$ExAND^{T}{}_{number} = \sum_{i=0}^{n} ExAND^{i}{}_{number} \tag{6.19}$$

Below, the equation for predicting number of logic operator $MIN^{C}{}_{number}$ per column,

$$MIN^{C}{}_{number} = \begin{cases} 1 \quad if \quad \min term_{non-zero} = 1 \\ 1 \quad if \quad \min term_{non-zero} = 3 \\ \quad 0 \quad otherwise \end{cases} \tag{6.20}$$

Total number of the logic operator for any MVL function can be expressed by the equation below by $MIN^{T}{}_{number}$,

$$MIN^{T}{}_{number} = \sum_{i=0}^{n} MIN^{i}{}_{number} \tag{6.21}$$

Each EXTENDED AND operation with respect to different column is trained by a different neural network. The neural net EXTENDED AND is formed, where i is the sequence for the operator. MIN is trained by a neural network then, the neural net MIN is formed. Every two column, logic operators are combined by MAX logic operator. MAX is trained by a neural network and then the neural net MAX is formed. It is assumed that implicant from a MVL function is composed of

${}^{0}x_1^{1}\bullet{}^{1}x_2^{1}+{}^{0}x_1^{2}\bullet{}^{2}x_2^{2}+{}^{0}x_1^{3}\bullet{}^{3}x_2^{3}$. It is also assumed that the implicant is derived from the column one.

Now, the partial implicant can be represented as $I_a={}^{0}\vec{x}_1^{1,2}({}^{1}x_2^{1}+{}^{2}x_2^{2})$. The other half of the implicant can be expressed as $I_b={}^{0}x_2^{3}\bullet{}^{3}x_2^{3}$. This expression can be further simplified using EXTENDED AND operator. $I_a={}^{0}\vec{x}_1^{1,2}({}^{1}x_2^{1}+{}^{2}x_2^{2})$ is further simplified using neural EXTENDED AND and MIN operator. The neural MVL operators are later trained to realize the synthesized expressions which are the $I=I_a+I_b$ or $I=NN\_ExAND_0+NN\_MIN_0$. After simplifying all the implicants, neural MAX operator operates on all sub-implicants to prompt the final output for the consecutive columns.

6.5 Analysis of Neural Network MVL Operators

MVL operations are observed by training a specifically designed neural network. Some novel and basic MVL operators are trained by the NNDA algorithm. It has been observed that the neural network takes less than 300 iterations to fully converge to the behavior of the logic operator. A sample of 5000 benchmark training sets has been sequentially created to train each of the neural network logic operators. For all the logic operators the neural architecture remains the same. All operators are trained with additional 1K benchmark. Table 6-2 in the later part of this chapter will summarize the simulation results for all operators.

6.5.1 Neural EXTENDED AND operator

The neural network EXTENDED AND architecture consists of feedforward backpropagation. x_1, a, b_1, b_2 and x_2 parameters were used to train the neural network. The network took five inputs and predicted the possible outputs. The neural network trained itself until it fully converges to the behavior of the logic operator. The validation and training performance is as observed in Figure 6.3. The neural network EXTENDED AND was used as one of the basic components to realize and synthesizes the MVL functions. The performance of the logic operator was observed by *mse* function and it reached as low as 0.00011 within 71 epochs. The best linear fit has been observed approximately 99.97%. Among 5000 training benchmarks, a random quantity of data has been chosen for testing and validation. Floating point variables were avoided for the training purpose.

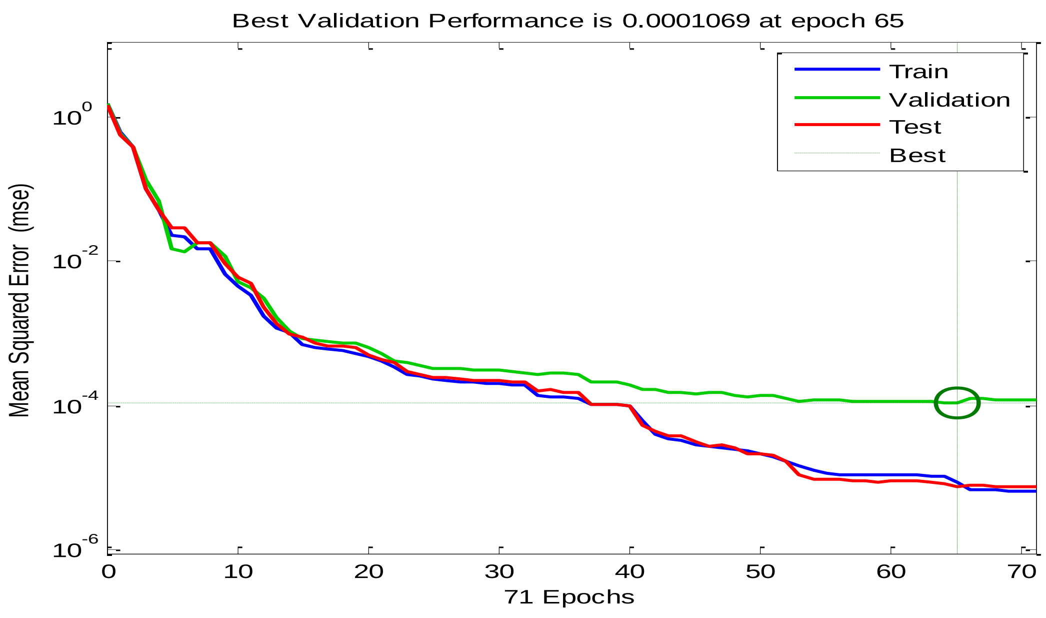


Figure 6.3: Performanceof EXTENDED AND after 71 epochs

6.5.2 Neural ODD operator

Basic neural ODD was trained with 4 valued 2 variable MVL functions. A random quantity of data was chosen for testing and validation. It was observed that the *mse* reached as low as 0.69628 within 107 epochs. The accuracy was observed to be approximately 75.97%. Floating point variables were considered for the training purpose for better linear data fit. The inputs to the network were fixed to 0-3 range. Testing performance and validation was observed as shown in Figure 6.4.

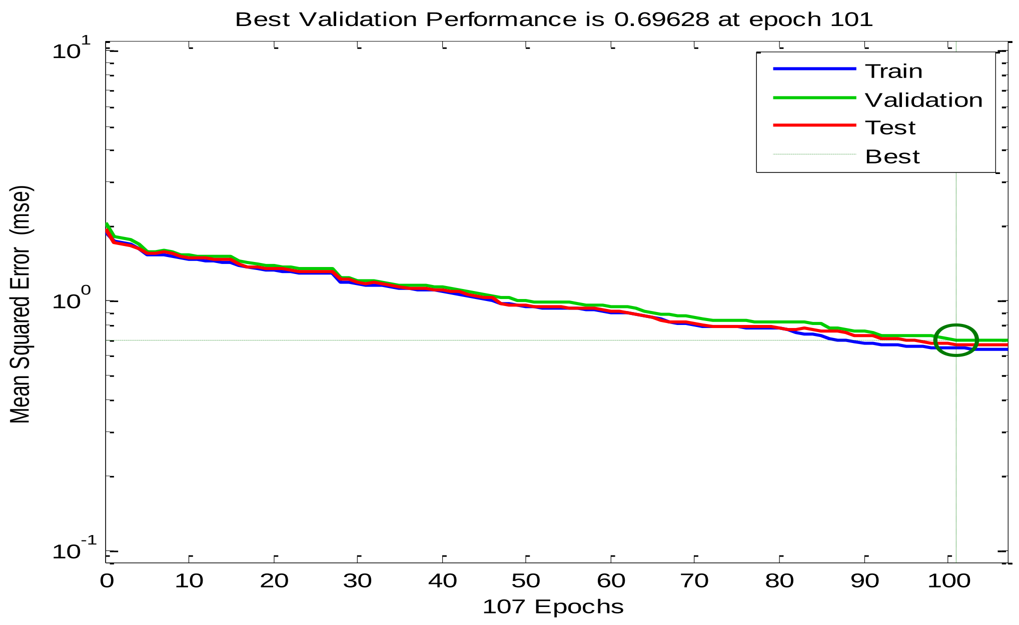


Figure 6.4: Performance of ODD after 107 epochs

6.5.3 Neural EVEN operator

The MVL-EVEN was trained similarly to ODD. The performance of the logic operator was observed by *mse* function. Figure 6.5 show the performance analysis. It was also observed that the *mse* reached as low as 0.04879 within 135 epochs. The best accuracy which was observed is 97.38%. Floating point variables were considered.

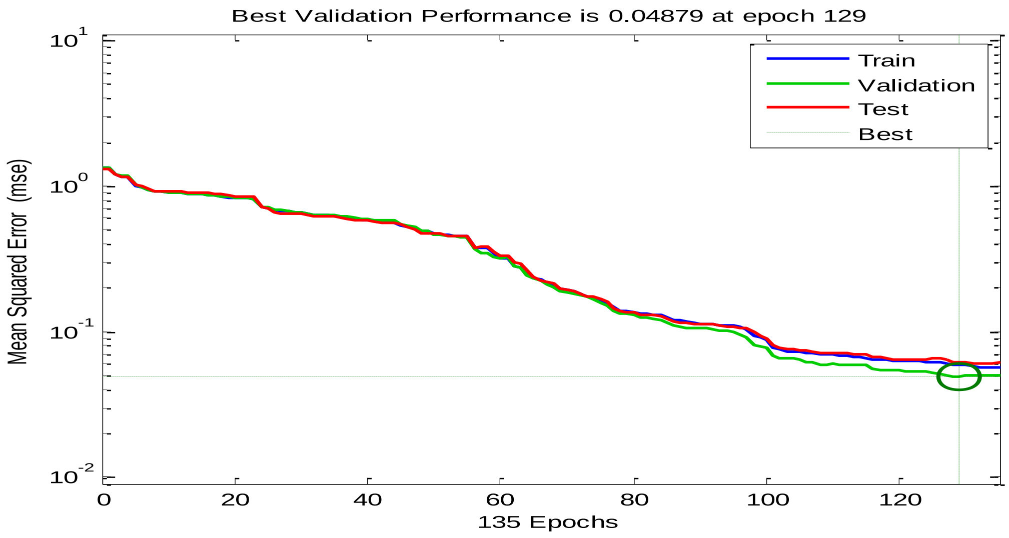


Figure 6.5: Performance of EVEN after 135 epochs

6.5.4 Neural MIN operator

The MVL MIN and MAX were trained with 4 valued 2 variables MVL. Testing performance of the extended operator is observed as Figure 6.6 below.

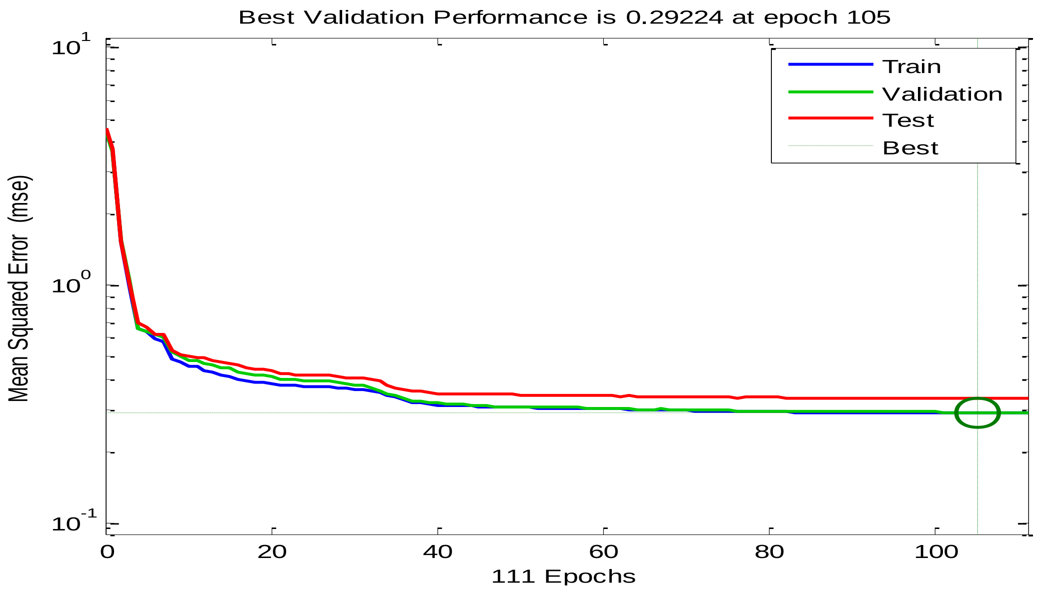


Figure 6.6: Performance of MIN after 111 epochs

From the observation, *mse* reached as low as 0.29224 within 111 epochs. The best linear fit tha has been observed was approximately 80.03%. Floating point variables were considered for the training purpose for better linear data fit. The inputs to the network was limited by 0, 1, 2 and 3.

6.5.5 Neural MAX operator

The mse reached as low as 0.86794 within 72 epochs. The observation shos that the best linear fit was 62.14%. An additional of 1000 benchmarks was used for testing the logic operator after training. Floating point variables were considered for the training purpose for better linear data fit. The inputs to the network were fixed by 0, 1, 2 and 3. Testing performance accuracy of the MAX operator is observed in Figure 6.7.

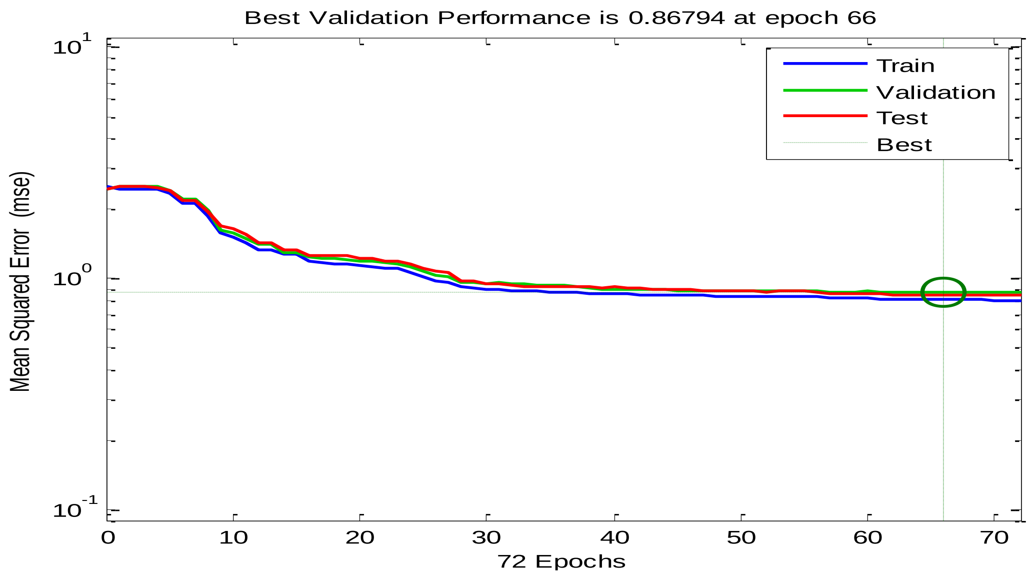


Figure 6.7: Performance of MAX after 66 epochs

Table 6-2: Post training comparison among different MVL neural net operators

MVL Operator	**Post Training Criteria**				
	Time taken to train	***Input to Output Delay***	***Inputs to NN***	***Hidden Layer Neurons***	***Accuracy (%)***
MIN	11.885	0.037	2	4	80.03%
MAX	9.654	0.019	2	4	62.14%
Extended AND	22.060	0.038	5	4	99.97%
EVEN	14.315	0.013	2	4	97.38%
ODD	9.095	0.009	2	4	75.97%
INVERTER	8.520	0.008	2	2	98.55%

6.6 Applications

The trained neural MVL operators can be utilized to realize the synthesized expressions of multi-valued logic functions. WINDOW LITERALs in each function represents the coordinate of the matrix for any specific minterm. All the minterms for any column are related together forming a MAX relation between them. The algorithm reads each index of the matrix and determines the end of a column. Considering the MVL function $f_1$from Table 6-1, where y = 3 and a, b are the coordinate of the minterm l_n. In Column 1 of the function, bold minterm 1 and 2 generates the implicant. Similarly, in column 2 bold minterm 2 and 1 and column 3 minterm 1 and 2 generates the implicants consecutively.

Each of the implicant is related together forming a MAX relation between each other. Below is an example of how EXTENDED AND operator functions on each implicant from different column.

$$f_1 = {}^0x_1^{\,1} \bullet {}^1x_2^{\,1} + {}^0x_1^{\,2} \bullet {}^2x_2^{\,2} + {}^1x_1^{\,2} \bullet {}^0x_2^{\,2} + {}^1x_1^{\,1} \bullet {}^2x_2^{\,1} + {}^2x_1^{\,1} \bullet {}^0x_2^{\,1} + {}^2x_1^{\,2} \bullet {}^1x_2^{\,2} \tag{6.22}$$

$$Synthesized(f_1) = {}^0\vec{x}_1^{\,1,2}({}^1x_2^{\,1} + {}^2x_2^{\,2}) + {}^1\vec{x}_1^{\,2,1}({}^0x_2^{\,2} + {}^2x_2^{\,1}) + {}^2\vec{x}_1^{\,1,2}({}^0x_2^{\,1} + {}^1x_2^{\,2}) \tag{6.23}$$

$$f_1\left({}^ax_1^{\,b}, {}^ax_2^{\,b}\right) = \begin{cases} {}^0\vec{x}_1^{\,1,2}(0+0) + {}^1\vec{x}_1^{\,2,1}(2+0) + {}^2\vec{x}_1^{\,1,2}(1+0) & if \quad f_1 : \xrightarrow{x_2} 0 \\ {}^0\vec{x}_1^{\,1,2}(1+0) + {}^1\vec{x}_1^{\,2,1}(0+0) + {}^2\vec{x}_1^{\,1,2}(0+2) & if \quad f_1 : \xrightarrow{x_2} 1 \\ {}^0\vec{x}_1^{\,1,2}(0+2) + {}^1\vec{x}_1^{\,2,1}(0+1) + {}^2\vec{x}_1^{\,1,2}(0+0) & if \quad f_1 : \xrightarrow{x_2} 2 \\ 0 \quad otherwise & \end{cases} \tag{6.24}$$

Each of the extended AND relations of the function f_1 can be represented as below,

$$f_{1a} = {}^0\vec{x}_1^{\,1,2}({}^1x_2^{\,1} + {}^2x_2^{\,2}) \tag{6.25}$$

$$f_{1b} = {}^1\vec{x}_1^{\,2,1}\left({}^0x_2^{\,2} + {}^2x_2^{\,1}\right) \tag{6.26}$$

$$Therefore, \qquad f_{1c} = {}^2\vec{x}_1^{\,1,2}\left({}^0x_2^{\,1} + {}^1x_2^{\,2}\right) \tag{6.27}$$

Each of these sub functions f_{1a}, f_{1b} and f_{1c} is replaced with a neural network Extended AND. The network is trained with five inputs x_1, a, b_1, b_2 and x_2. The final output y is represented with the equation below. All the neural operators are interconnected based on the final expression. The desired output has been observed with a 5 layered MVL neural network operators.

$$y = f_{1a} + f_{1b} + f_{1c} \tag{6.28}$$

6.7 Summary

MVL neural operators were used to synthesize the functions. The synthesized expressions were obtained by novel MVL neural operators. NNDA-MVL algorithm's advantages were demonstrated with the accuracy of realization for MVL neural operators. The evaluation of NNDA-MVL algorithm was based on the features such as input to output delay and accuracy achieved in training with 4 hidden neurons. In a brief, an effort of training MVL neural operators as well NNDA_MVL as an application to synthesize logic have been observed.

NNDA algorithm was utilized for generation of trained MVL neural operators. Logic functions were also synthesized by using these neural operators. As an application, the hybrid combination of MVL and NNDA resulted in efficient synthesis. The simulation results obtained using the NNDA-MVL algorithm was compared against other logic operators. The results achieved have shown that the NNDA-MVL algorithm managed to achieve an accuracy of 99.97% for extended AND neural operator. The result depicted the fastest as 0.008 seconds and the slowest as 0.038 seconds of input to output delay respectively for MVL inverter and extended AND operator.

Chapter 7

Neural Network Deployment Algorithm

7.1Introduction

Most of the efficient Multi-Valued Logic (MVL) synthesis algorithms do not have the learning capability. The lack of any learning algorithm for MVL networks is an important restraint to many applications. In the process of learning adaption also provides a degree of robustness by compensating for minor variability. It has been observed that the neural network MVL synthesis takes more hardware in terms of neuron count and interconnections (Zheng, Ishizuka and Tanno 1995). In this chapter, an effort to reduce the number of neuron in network for synthesizing MVL functions is explored.

This chapter will also tackle a representation of Neural Network Deployment Algorithm (NNDA) -MVL synthesis algorithm. This algorithm is able to realize and synthesize MVL using neural network. NNDA-MVL algorithm is combined with feed-forward backpropagation learning capability and novel MVL operators (Chowdhury, Raj and Singh 2013). In the first part of this chapter, an elaborative description of MVL operators is presented. These operators are trained and tested using NNDA for synthesis purpose. A basic construction platform for NNDA is created which serves as the basis for realizing all the MVL operators. Separate module for each gate is constructed. Each of the neural net logic operators is used for the synthesis of the logic expression. To the extent of literature done, the proposed method shows that the number of neuron count in the quaternary function networks reduces remarkably using NNDA-MVL algorithm. The proposed method is a MVL neural network synthesis which is a novel approach to generate efficient synthesis of MVL functions.

7.2 EXTENDED AND Operators in MVL Synthesis

In this section, a brief synthesis procedure is demonstrated. This is followed by a detailed discussion on construction of network architecture as a base for all neural operators. In view of EXTENDED AND operator's definition, the operator provides an opportunity for eliminating the usage of casual MIN operator twice in one expression. Thus, EXTENDED AND is extensively used during the synthesis of MVL functions in this chapter. For example, if there exist an implicant of ${}^{0}x_1{}^{1} \bullet {}^{1}x_2{}^{1} + {}^{0}x_1{}^{2} \bullet {}^{2}x_2{}^{2}$, the implicant can be used and simplified using MVL algebraic rules for EXTENDED AND and the simplified implicant can be represented as ${}^{0}\vec{x}_1{}^{1,2}({}^{1}x_2{}^{1} + {}^{2}x_2{}^{2})$.

The MVL matrix ${}^{a}x_1{}^{b}$ and ${}^{a}x_2{}^{b}$ represent the value of a minterm. Both of them are WINDOW LITERALs. This one input-output logic helps to form the representation of minterms with coordinates x_1 and x_2. An MVL function f_1 is given as [012201120]. Considering coordinate $(x_1, x_2) = (2,1) = 2$, the value of the minterm at this point is 2. Since the minterm is 2 then b = 2. $x_1 = 2$ therefore $a_1 = 2$ and $x_2 = 1$ therefore $a_2 = 1$. So

${}^{2}x_1^{2} \bullet {}^{1}x_2^{2}$ representation could be achieved. According to the definition of window literal the logic expression could be presented as ${}^{2}x_1^{2} \bullet {}^{1}x_2^{2} = 2 \bullet 2$ or 2. Therefore an ideal minterm can be presented as MIN operation of two WINDOW LITERALs as ${}^{a_m}x_1^{\,b_n} \bullet {}^{a_m}x_2^{\,b_n}$,where $a_1, a_2, \cdots, a_m \in R$, $b_1, b_2, \cdots, b_n \in R$, $0 \le \{a, b, x_1, x_2\} \le (y-1)$, $a_m \neq a_m \quad and \quad b_n = b_n$. The literal bounds are defined as below:

$$ {}^{a_1}x_1^{\,b_1} = \begin{cases} b_1 \quad if \quad x_1 = a_1, \quad where \quad b_1 = \min term \quad value \\ \qquad 0 \quad otherwise \end{cases} \tag{7.1} $$

$$ {}^{a_2}x_2^{\,b_2} = \begin{cases} b \quad if \quad x_2 = a_2, \quad where \quad b_2 = \min term \quad value \\ \qquad 0 \quad otherwise \end{cases} \tag{7.2} $$

7.3 Construction of NNDA Architecture

The construction algorithm for NNDA architecture is presented in this section. The architecture serves as a basic platform for creating different module for different logic operators. Next, a pseudo-code for NNDA algorithm is introduced. An elaborative discussion on how NNDA algorithm synthesizes MVL functions is shown in later part of the section.

7.3.1 Algorithm and procedure

The construction of the MVL neural net begins with the algorithm provided below in Figure 7.1. The algorithm initializes all the input, output and targets. Based on this procedure the NNDA algorithm would perform further processing in realization.

```
Procedure BuildNetworkArchitecture();

Each neuron output function $N_{Output} := \beta_{nNL}^{L} = \sum_{i=0}^{\beta_{input}} \left(\omega_{ij}^{L} \bullet x_i\right) + b_{nNL}$

Each neuron transfer function $\frac{2 \bullet e^{2 \cdot N_{Output}}}{e^{2 \cdot N_{Output}} + 1} - \frac{\left(2 \bullet e^{2 \cdot N_{Output}}\right) \bullet \left(e^{2 \cdot N_{Output}} - 1\right)}{\left(e^{2 \cdot N_{Output}} + 1\right)^2}$

do

Initializing input_Vector from input file

input_Vector  := input.file();

$\overrightarrow{mvl\_Input\_Vector_n} := [x_0, x_1 \ldots x_r]^T$

Initializing target_Vector from target file

target_Vector  := target.file();

$\overrightarrow{mvl\_T\arg et\_Vector_n} := [t_0, t_1 \ldots t_r]^T$

if(sizeof (input_Vector) != sizeof(target_Vector))

Create a r-layered feed-forward backpropagation network

neural_network := createNetwork (mvl_Input_Vector, mvl_Target_Vector, 4);

Set Function property and divide them

Divide vectors into three sets using random indices

neural_network.divideFunction();

        else

            do nothing;

Reinitialize the weights and bias of each layer
```

```
neural_network := init (neural_network);
**while** (sizeof (input_Vector) != sizeof(target_Vector))
set expected_MSE = 0.01
define $mse = \frac{1}{r}\sum_{i=0}^{r}\left(\overset{\wedge}{O_i} - O_i\right)^2$
**do**
Train the neural_network using $x_{r+1} = x_k - \frac{1}{\left[J^T \bullet J + \mu I\right]} J^T \bullet e$
[neural_net, tr]:=train(neural_net, input_Vector, target_Vector);
calculate(MSE);
**while**(MSE>expected_MSE)
```

Figure 7.1: The procedure which builds the MVL neural architecture

NNDA not only reduces the product term but also reduces the neural net block for representing each of the MVL functions. NNDA converts a MVL function into a matrix. Then the algorithm finds each column and their respective rows for any non-zero minterms. The algorithm determines all the non-zero minterm from 1st column to nth column.

Each column of the matrix represents one or more effective implicants of NNDA-MVL synthesis method. Each of the implicant is formed from different number of EXTENDED AND and MIN operators. MIN logic operator is only trained once during the synthesis. MIN is also a universal operator and it does not change depending on the column sequence. MIN is trained by a neural network and then, the NN_MIN_i is formed. Every two column logic operators are combined by the logic operator MAX.

Similar to the MIN operator, the MAX logic operator is also trained once during the synthesis as it is a universal operator and it does not change depending on the column sequence. MAX is trained by a neural network and which will from the NN_MAX_i.

As mentioned in Chowdhury and Singh (2014), an implicant is composed of ${}^0x_1{}^1 \bullet {}^1x_2{}^1 + {}^0x_1{}^2 \bullet {}^2x_2{}^2 + {}^0x_1{}^3 \bullet {}^3x_2{}^3$, from a MVL function. Also assume the implicant is derived from column one. Now the partial implicant can be represented as $I_a = {}^0\vec{x}_1^{1,2}({}^1x_2{}^1 + {}^2x_2{}^2)$. The other half of the implicant can be expressed as $I_b = {}^0x_2{}^3 \bullet {}^3x_2{}^3$ This Expression could be further simplified using EXTENDED AND operator. $I_a = {}^0\vec{x}_1^{1,2}({}^1x_2{}^1 + {}^2x_2{}^2)$ is simplified using EXTENDED AND operator. Therefore $I = I_a + I_b$ or $I = NN\_ExAND_0 + NN\_MIN_0$.

After simplifying all the implicants, MAX operator operates on all the sub-implicants to prompt the final output for the 1st column. Based on the NNDA, a pseudo code has been prepared. Below the entire pseudo code for the algorithm is represented in Figure 7.2.

```
Do
Generate Sequence of all possible 4 valued 2 variable combinations
Locate each Column of respective window literal using array
track_Term=0;
        for each column $C_i : i < m$

Find each related column consisting ${}^a x_1^b$
```

```
        for each row $R_j : j < n$
                Find each related row consisting ${}^{a}x_2^{\,b}$
                Check minterms in each of the relations
$R_j = \left({}^{a}x_1^{\,b}, {}^{a}x_2^{\,b}\right)$
                if($\left({}^{a}x_1^{\,b}, {}^{a}x_2^{\,b}\right) \neq 0$)
$minterm_j = \min termCheck(R_i);$
$track\left[track\_Term\right] = \min term_j;$
getLiteral[track_Term]=getWindowLiteral(track[track_Term],$R_j$,$C_i$);
track_Term++;
                else
                        exit loop;
NN_MIN = BuildNetworkArchitecture();
NN_MAX = BuildNetworkArchitecture();
NN_ExAND = BuildNetworkArchitecture();
        Get NN_MIN and NN_MAX using track_Term;
        reset track_Term=0;
        getReducedMinterm();
        getAvgPTResults();
while( $minterm_j \neq null$ )
end


getWindowLiteral(Term[],row,column):
```

```
for each column find ${}^{a}x_{1}{}^{b}$
        for each row find ${}^{a}x_{2}{}^{b}$
                get(a,b,$x_2$)
                value1 = getValue(a,b,$x_2$);
x2 = get(a,b,$x_2$);
return x2;
x1 = get(a,b,$x_1$);
value2 = getValue(a,b,$x_2$);
return x1;
```

Figure 7.2: NNDA-MVL pseudo code

7.4 Simulation Analysis

MVL operations were analysed by specifically designed NNDA algorithm. MVL operators were trained by the algorithm. It was observed that the neural network took less than 300 iterations to fully converge to the behaviour of the logic operator. A sample of 5000 benchmark training sets was sequentially created to train each of the neural network system. For all the logic operators, the basic neural architecture remained the same.

7.4.1 EXTENDED AND neural network

EXTENDED AND architecture consisted of feedforward backpropagation and five inputs. The inputs x_1, a, b_1, b_2 and x_2 were used to train the neural network. The network took the inputs and predicted the possible outputs. The construction of neural network section explained the details on how the basic architecture of the network was created.

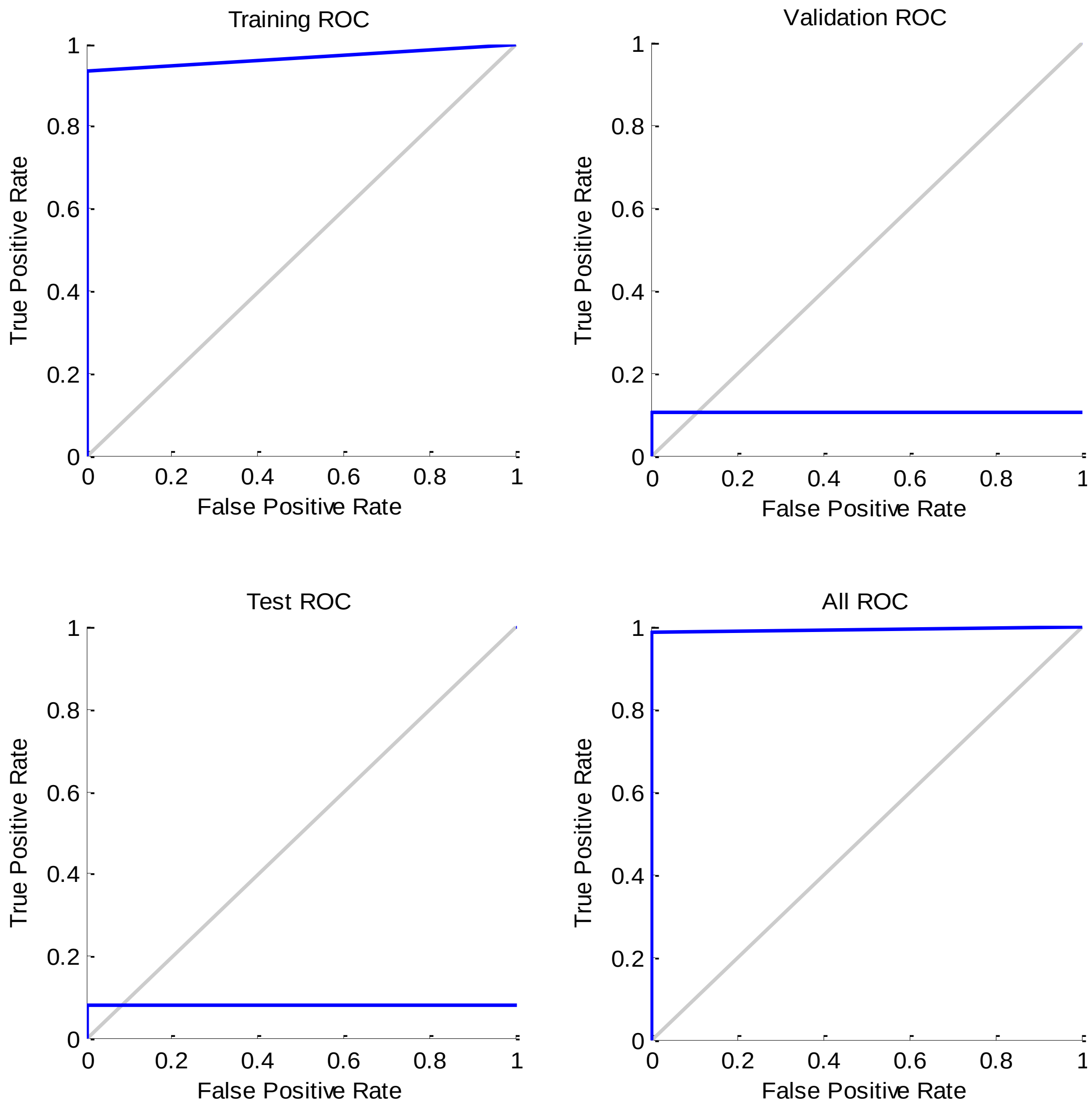


Figure 7.3: ROC observation for EXTENDED AND

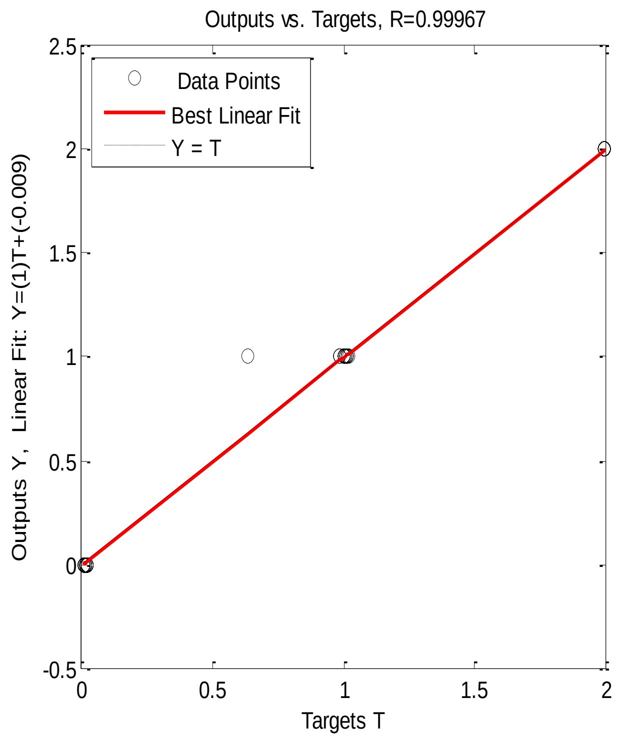


Figure 7.4: Accuracy of EXTENDED AND operation

The neural network trained itself until it fully converged to the behaviour of the logic operator. The operator was used as one of the basic component to realize and synthesize the MVL functions. The neural operator was trained with 3 valued 2 variable MVL functions. The training, validation and test Receiver Operating Characteristics (ROC) can be observed in Figure 7.3 and 7.4 above. The more the values are leaning towards the left axis and closer to "1", the better the classification stands. ROC was used to check the quality of the classifiers.

It was observed that the performance mean square error (*mse)* reached as low as 0.00011 within 71 epochs. The best linear fit has been observed approximately 99.97%. Among training benchmarks, a random quantity of data was chosen for testing and validation. An additional of 1K benchmarks was used for testing the logic operator after training. Floating point variables were avoided for the training purpose. The inputs to the network were limited by 0, 1 and 2 values. The testing performance and post training accuracy is shown in Figure 7.4. Random weights were initialized with every input to the network. A total of 20 weights which were associated with inputs can be observed from Figure 7.5. After the training, it was observed that within 71 epochs the gradient reached 0.00020 and 6 validation checks had commenced. The gradient analysis and the validation check can be observed in the Figure 7.6.

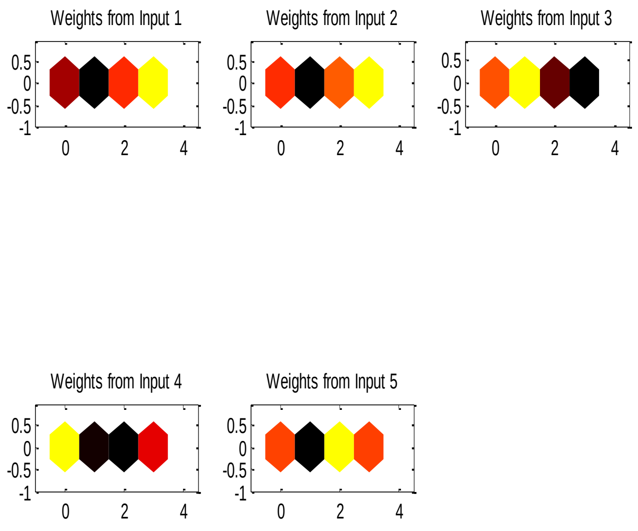


Figure 7.5: Weight initialization for x_1, x_2, b_1, b_2, a

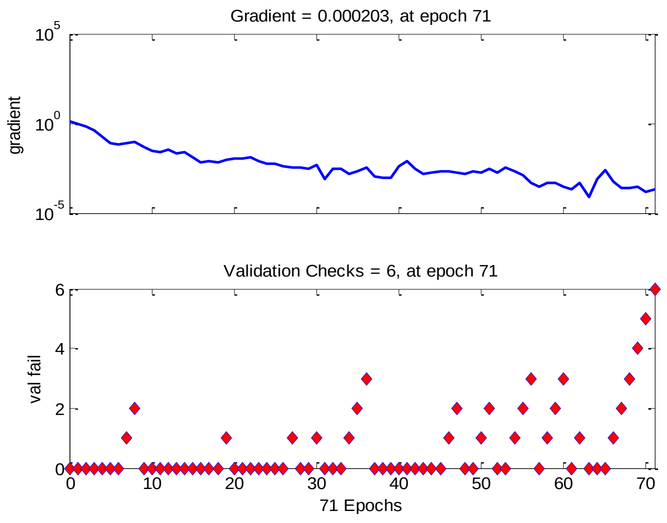


Figure 7.6: Gradient and validation after 71epochs

7.4.2 ODD neural network

Basic neural ODD was trained with 4 valued 2 variable benchmarks. Among all benchmarks, a random quantity of data was chosen for testing and validation. The *mse* reaches as low as 0.69628 within 101 epochs. The best linear fit was observed approximately 75.97%. An additional of 1K benchmarks was used for testing the logic operator after training. Floating point variables were considered for the training purpose and better linear fit data. The inputs to the network were limited by 0, 1, 2 and 3. Weight initialization, post training accuracy and gradient analysis were as observed in Figure 7.7, 7.8 and 7.9. It was observed that within 107 epochs the gradient reached 0.10650 and 6 validation checks was commenced.

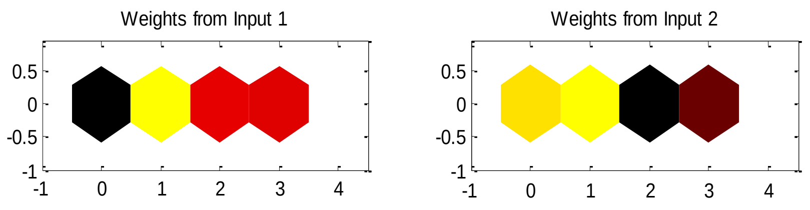


Figure 7.7: Weight initialization for x_1, x_2

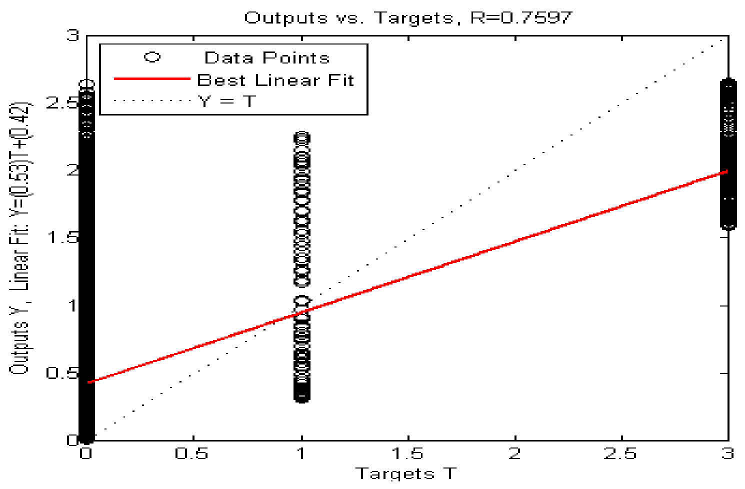


Figure 7.8: Accuracy of ODD operation

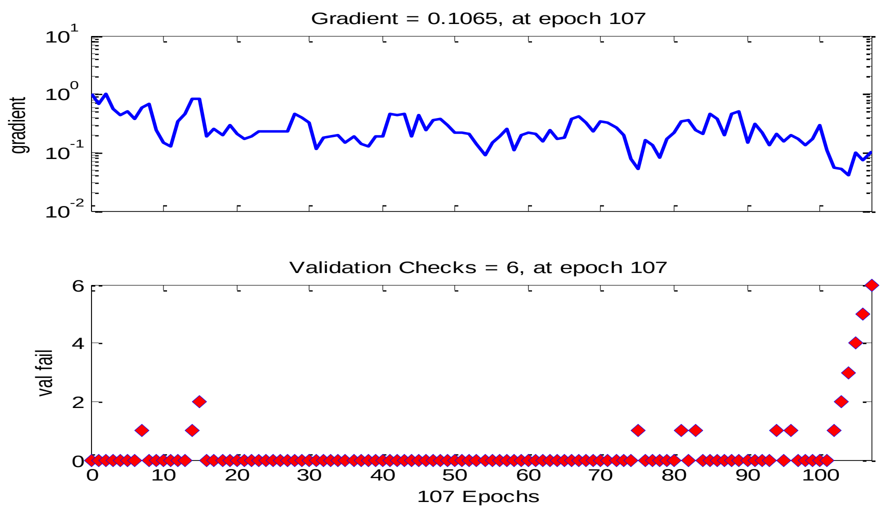


Figure 7.9: Gradient and validation after 107 epochs

7.4.3 EVEN neural network

MVL-EVEN was trained with 4 valued (0-3) benchmarks. Additional benchmark was used for post-training evaluation. Floating point variables were considered for the training purpose for better linear data fit. The best linear fit that was observed is to be approximately 97.38%. The *mse* reached as low as 0.04879 within 129 epochs. Total of 8 weight initialization, testing performance and gradient analysis of the EVEN operator can be observed in Figure 7.10, 7.11 and 7.12.

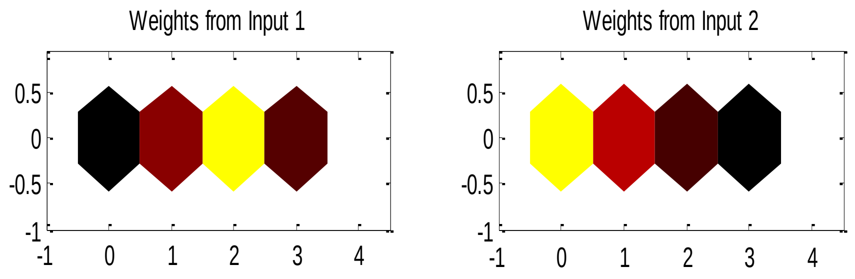


Figure 7.10: Weight initialization for x_1, x_2

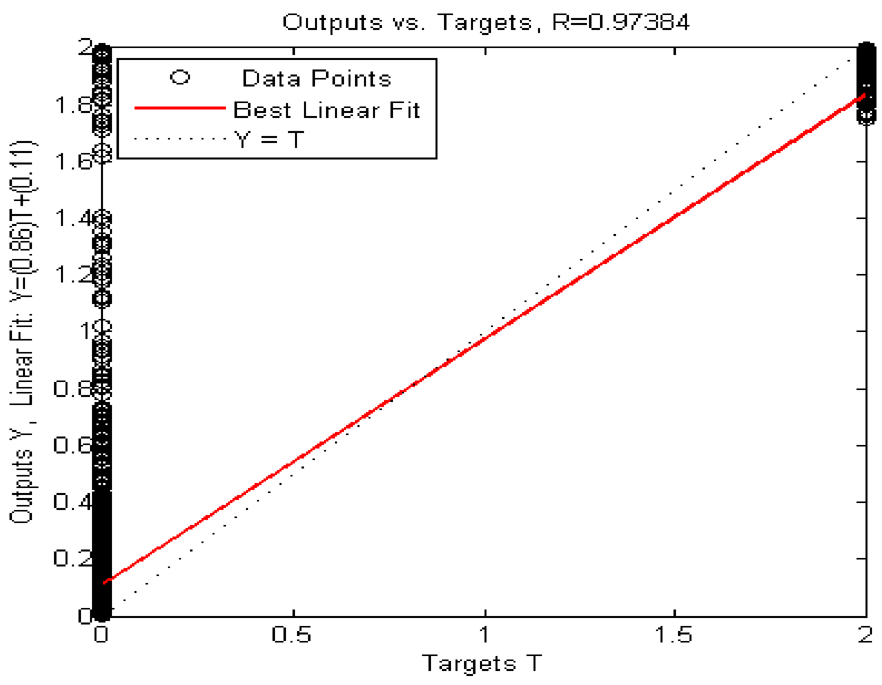


Figure 7.11: Accuracy of EVEN Operation

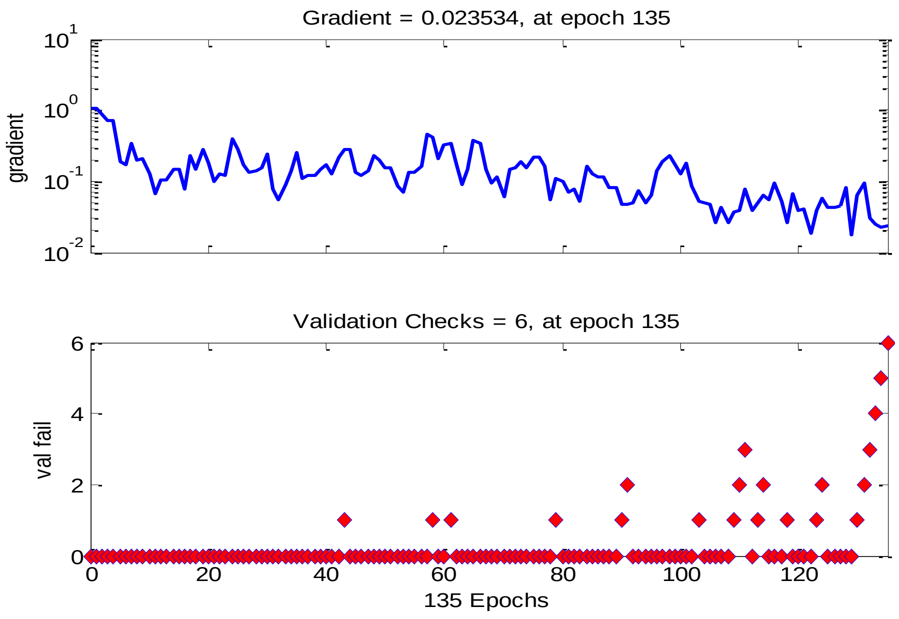


Figure 7.12: Gradient and validation after 135 epochs

7.4.4 MIN and MAX neural network

The MIN and MAX operators were trained with 5K 4-valued benchmarks. 1K benchmark was used for testing the logic operators. Floating point variables were considered for better linear data fit. The *mse* was measured up to 0.29224 for MIN and 0.86794 for MAX consecutively within 105 and 66 epochs. MIN obtained the best linear fit of 80.03% compared to 62.14% of MAX. For MAX, the gradient has reached 0.01201 within 111 epochs. On the other hand, MIN has reached 0.03418 within 72 epochs. On both cases, validation check is 6. Weight initialization, testing performance and gradient analysis of the MIN and MAX operator can be observed in Figure 7.13, 7.14, 7.15, 7.16 and 7.17.

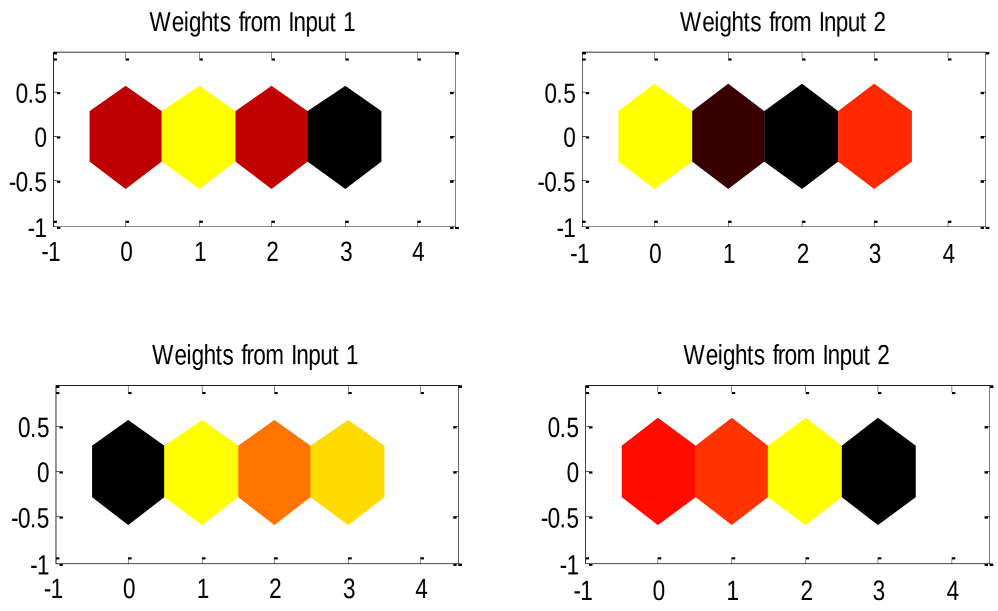


Figure 7.13: Weight initialization of MIN and MAX operator

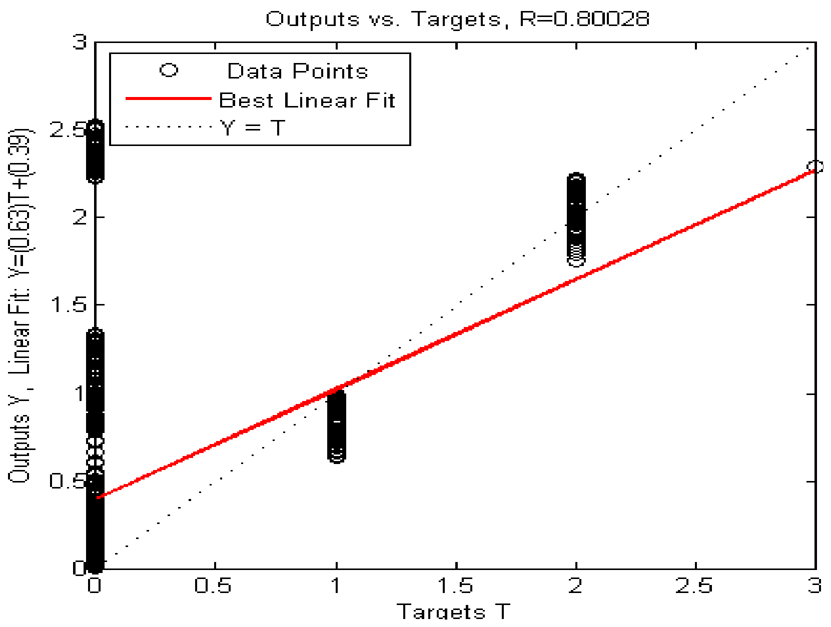


Figure 7.14: Accuracy of MIN operation

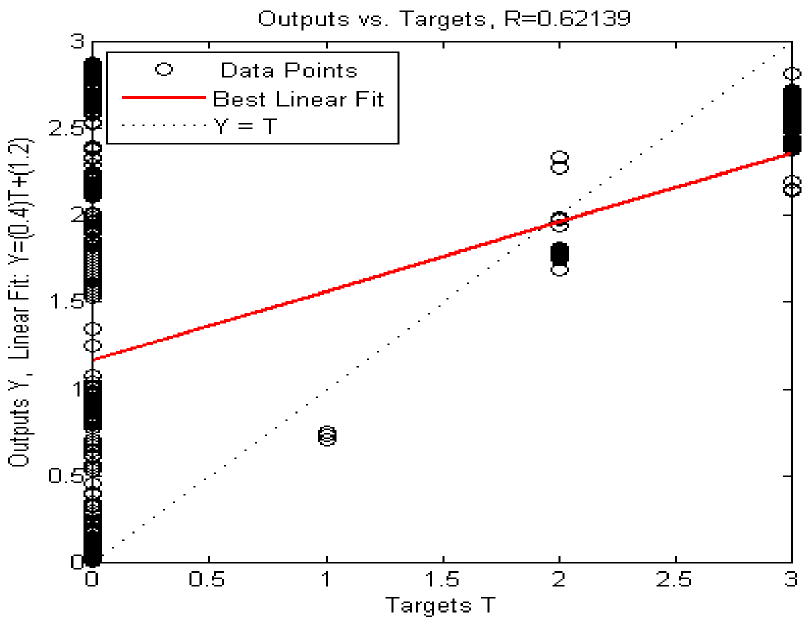


Figure 7.15: Accuracy of MAX operation

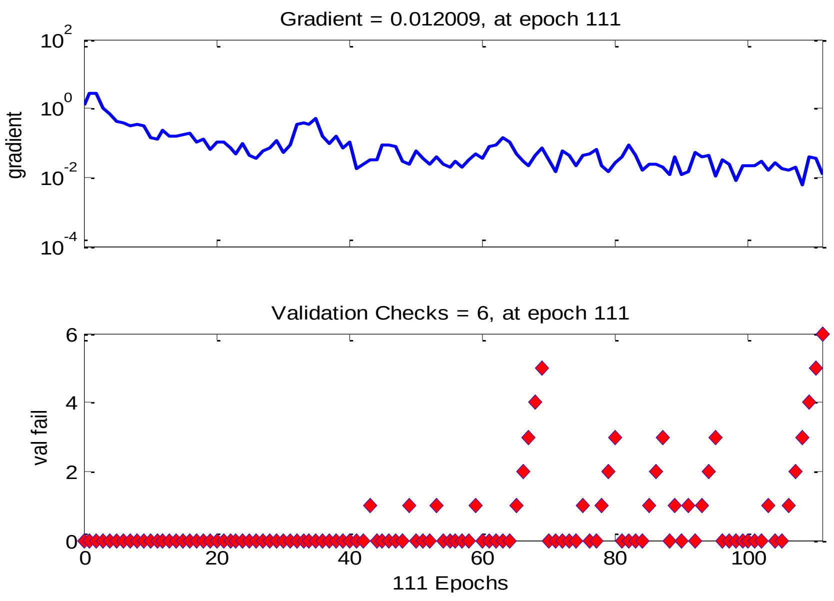


Figure 7.16: Gradient and validation analysis for neural MIN operator

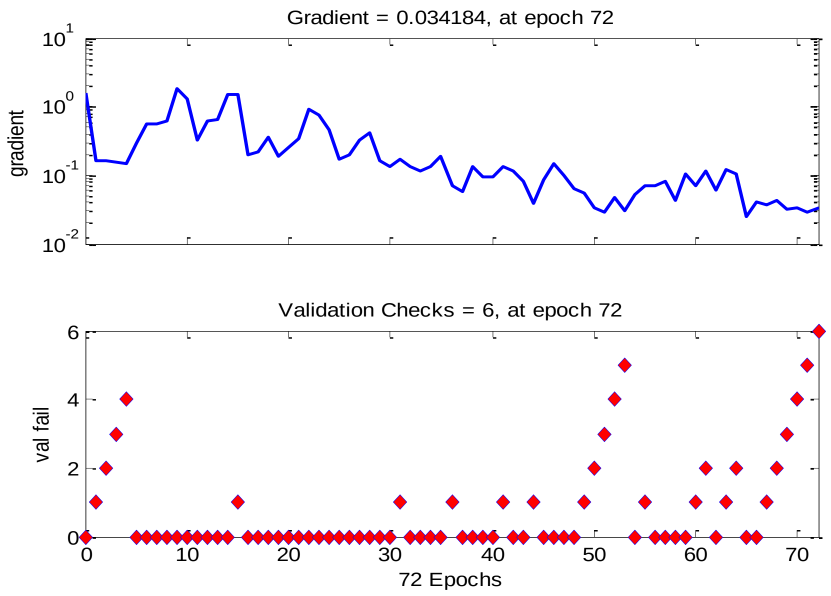


Figure 7.17: Gradient and validation analysis of MAX neural operator

7.5 Discussion and Comparisons

MVL gate count and network link count is observed when different neural net logic operators are interconnected to realize synthesized MVL functions. Figure 7.16 illustrates the entire reduced neural network for realizing the quaternary function in the paper by Jain, Bolton and Abd-El-Barr (1993). Table 7-1 shows NNDA-MVL algorithm has managed to reduce logic operators and interconnected link in the neural network.

Table 7-1: Post training comparison among different MVL neural net operators

	Zheng, Cao and Ishizuka (1998)	**NNDA-MVL**	**Improved Reduction Achievement (%)**
*MVL MIN Operator	17	8	52.941%
MVL MAX Operator	13	13	00.00
Interconnected Links	77	59	23.377%

*In Table 7-1 MVL MIN operator consists of MIN and EXTENDED AND operator.

The reduced neural network incorporates lesser MVL neural operators for synthesizing a logic expression. The network also reduces the number of internal connections among the logic operators. For some MVL operators the network provides an efficient and fast output. A total of logic operator is used for the network architecture. 2 NN_MIN , 6 NN_ExtAND, 16 $NN\_CONVERTER$ and 13 NN_MAX .

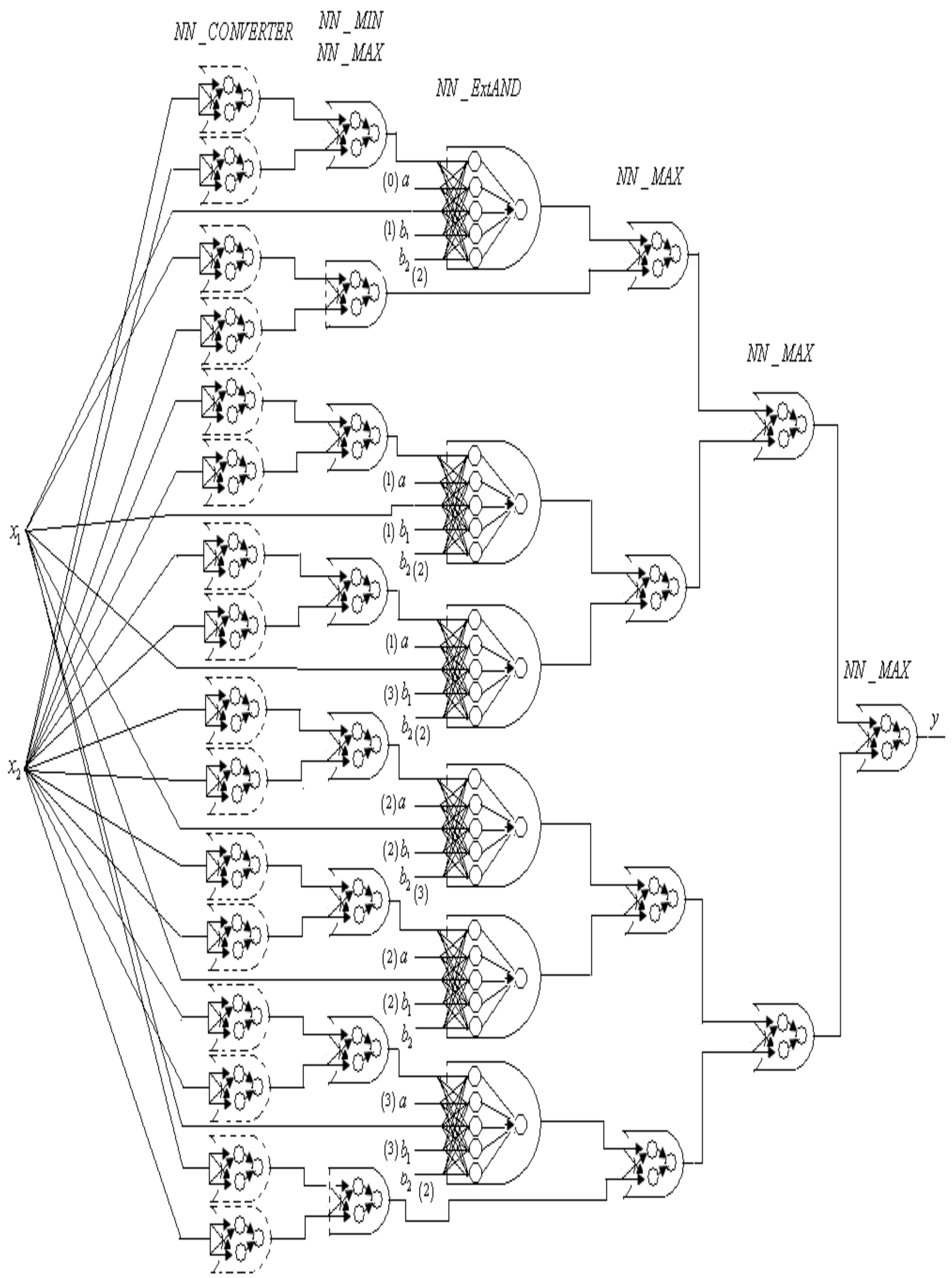


Figure 7.18: Networks for a synthesized MVL function

A comparative analysis is carried out for different MVL neural network operators in this section of the chapter. Some specific criteria such as, CPU time taken to train, input to output delay, number of inputs, hidden neurons and accuracy. INVERTER not only takes the least amount of time 8.52 seconds to train, but also it has the least transition delay of 0.00842 seconds. The highest accuracy of 99.97% is observed by EXTENDED AND neural operator. Among MIN, MAX, EVEN and ODD operators, ODD not only takes the least amount of time of 9.09 seconds to train but also has the least input-output delay of 0.00988 seconds. EVEN has the highest accuracy of 97.38%.

7.6 Summary

In this chapter, an evolutionary technique for synthesizing MVL functions using NNDA was presented. The algorithm was combined with back-propagation learning capability and novel MVL operators. The advantages of NNDA-MVL algorithm was also demonstrated with realization of synthesized MVL function by lesser number of MVL operators. The comparison against NNDA-MVL algorithm was based on the reduction of MVL gate count, network link count, network input to output delay and accuracy achieved in training. As a summarisation, an effort of reduced network size was observed for synthesized MVL functions in this chapter. Overall the algorithm showed an improvement of 52.94% for MVL MIN gate reduction and reduced MVL neural net internal link connections of 23.38% compared to existing techniques.

Chapter 8

Non-Zero Multi-Valued Decision Diagram

8.1 Introduction

There are several researches of Multi-Valued Logic (MVL) synthesis based on Decision Diagrams (DD). According to Files, Drechsler and Perkowski (1997), the MVL functions can be functionally decomposed using the decision diagrams. DDs can also be used to represent and manipulate MVL functions. The DD packages as discussed in the chapter by Drechsler, Jankovic and Stankovic (1999) are based on recursive synthesis operations. The author discussed an approach which can be easily utilised to prototype the MVL DD packages. Drechsler, Thornton and Wessels (2000) utilised the decision diagram for the synthesis of MVL specifically MVL Networks (MVLN) by developing an edge-mapping approach for MVLN. The resulted circuits have linear size with respect to the initial DDs (Drechsler, Thornton and Wessels 2000).

Miller and Drechsler (2002) examined the constructions of Multi-Valued Decision Diagrams (MDD). The evaluation is achieved by comparing the recursive MIN and MAX as primitive operations in MDD. The authors also used the cyclic negations and complements as MDD edge operations to reduce the MDD node count. Jiang, Matic and

Brayton (2003) studied the optimum functional evaluation problem for a multi valued relation. The Generalized Co-factoring Diagram (GCD) proposed in this paper is tested on multi valued relations which can be used in logic simulation, software synthesis for embedded control applications and functional decomposition in logic synthesis (Jiang, Matic and Brayton 2003). It has also been observed that many existing algorithm requires more hardware compared to evolutionary algorithms (Sarif and Abd-El-Barr 2006).

In this chapter, an effort to reduce the Product Term (PT) for synthesizing MVL functions is studied. It represents a tree-like decision diagram, Non Zero MDD **(**NZMDD) which is an extension of the existing MDD. Firstly, the MVL function is represented as the Reduced MVLN (RMVLN). RMVLN is reduced to only two rows in height. Next, RMVLN is mapped to NZMDD for synthesis process. The synthesized compact form of NZMDD is evaluated as MVL expressions. MVL gates such as, WINDOW LITERAL, EXTENDED AND, MIN and MAX is used to realize the logic expressions. Two sets of randomly generated benchmark circuits are generated to evaluate the experimental results. First set initially consist of 1 million randomly generated MVL functions, out of which 19600 circuits are used to represent 3-valued MVL synthesis. 4-valued 49998 MVL benchmark circuits are used for synthesis and comparison against ACO-MVL algorithm. This algorithm is the latest research finding on the evolutionary algorithm which synthesizes MVL functions. Therefore, ACO-MVL algorithm is chosen to be compared against the proposed NZMDD algorithm. Among all the evolutionary algorithms, the ACO-MVL exhibits the best PT reduction rate. Hence, ACO-MVL algorithm is compared against the proposed NZMDD algorithm to justify the improvement in PT reduction.

8.2 Details of RMVLN and NZMDD

In this section, a description of nodal logic gates is presented. A detailed elaboration of RMVLN is introduced and how NZMDD is used to synthesize from RMVLN is also discussed. The structure of NZMDD and its working are elaborately explained. A brief sight of MVL expression is presented to show how these are used to form the circuits.

8.2.1 Model of RMVLN and its nodal logic gate

Multi-Valued Logic (MVL) functions are represented as RMVLN. RMVLNs are formed as a directed acyclic graph which consists of vertices V and edges E. The graph is represented as $C = (V, E)$. One of the basic cell's which comprises of an Initial Input (II) or Initial Output (IO) is labeled with each of the vertex $v \in V$. There exists a set of fundamental cells in RMVLN which consists of WINDOW LITERAL, EXTENDED AND, MIN and MAX logic gates. If an output of a cell related with u is connected to the input of another cell related with v, then there exists an edge $E = (u, v)$ from vertex u to v. The vertex information about their input and output is embedded in every edge. The first node of the network is labeled as II since it has no incoming edge. Nodes that have no outgoing edges are labeled as IO. The basic cells interact with each other through edge and later help forming logic expressions. The network has a maximum of two rows of nodes. Incoming and outgoing edge of a node is considered to decide the form of fundamental cell and their inputs.

In a RMVLN the II values are determined from an ordered finite set S , where $S=\{0,\ldots,k-1\}$ and k denotes the logic levels. Terminals $T=\{T_0,T_{1,}\ldots,T_t\}$ are available for each of the II. Terminal values are determined by the ordered finite set T_s, where $T_s=\{1,\ldots,k-1\}$. The terminal values for initial input x are defined as $T_t(x_i)=\{1,\ldots,k-1\}$.

A WINDOW LITERAL consisting of bounds $\{a,b\}$ *and* $(a,b)\in S$, $[0\le a\le b]<k$ has one input and output. As proposed in (Abd-El-Barr and Sarif 2006) for a given input x with bounds $\{a,b\}$ the literal or short literal is defined as:

$$^{a}x^{\,b}=\begin{cases} b & if \quad x=a \\ 0 & otherwise \end{cases} \tag{8.1}$$

Where $a_1,a_2,\cdots,a_n\in R$, $b_1,b_2,\cdots,b_n\in R$ and $0\le\{a,b\}\le(k-1)$ and b is called the value of the literal. In an MVL function $^{a}x_1^{\,b}$ and $^{a}x_2^{\,b}$ represents the value of a minterm. Window literal helps to form the representation of non-zero minterms, with the assistance of coordinate (x_1,x_2). Considering coordinate $(x_1,x_2)=(2,1)=2$ the minterm is retrieved from 3rd column 2nd row of the function. Since the minterm is 2, then b = 2. $x_1=2$ therefore $a_1=2$ and $x_2=1$ therefore $a_2=1$. Hence the literal could be represented as $^{2}x_1^{\,2}\bullet{}^{1}x_2^{\,2}=2\bullet2=2$. Therefore an ideal minterm could be represented as MIN operation of two WINDOW LITERALS as $^{a_m}x_1^{\,b_n}\bullet{}^{a_m}x_2^{\,b_n}$ where $a_m\neq a_m,b_n=b_n$ and:

$$^{a_1}x_1{}^{b_1} = \begin{cases} b_1 \quad if \quad x_1 = a_1, \quad where \quad b_1 = \min term \quad value \\ \quad\quad 0 \quad otherwise \end{cases} \tag{8.2}$$

$$^{a_2}x_2{}^{b_2} = \begin{cases} b \quad if \quad x_2 = a_2, \quad where \quad b_2 = \min term \quad value \\ \quad\quad 0 \quad otherwise \end{cases} \tag{8.3}$$

Consideration of given input $^{a}\vec{x}_1{}^{b_1,b_2}$ *and* x_2 of EXTENDED AND operator as proposed by Chowdhury, Raj and Singh(2013) is defined as below.

$$^{a}\vec{x}_1{}^{b_1,b_2} \bullet x_2 = \begin{cases} b_1 \quad if \quad x_2 = b_1 \, and \quad x_1 = a \\ b_2 \quad if \quad x_2 = b_2 \, and \quad x_1 = a \\ \quad 0 \quad otherwise \end{cases} \tag{8.4}$$

Where $0 \le \{a, b, x_1, x_2\} \le (k-1)$, x_2 determines the value of the operator if an implicant from a MVL function as $^{0}x_1{}^{1} \bullet {}^{1}x_2{}^{1} + {}^{0}x_1{}^{2} \bullet {}^{2}x_2{}^{2}$ exist. The implicant can be simplified using EXTENDED AND operator. The simplified implicant can be represented as $^{0}\vec{x}_1{}^{1,2}({}^{1}x_2{}^{1} + {}^{2}x_2{}^{2})$. Considering coordinate $x_1 = 0$ and $x_2 = 2$, the implicant achieves $^{0}\vec{x}_1{}^{1,2}(0+2) = {}^{0}\vec{x}_1{}^{1,2} \bullet 2, \quad where \quad x_2 = 2$ and the output is 2.

8.2.2 NZMDD generation using RMVLN

Functions ranging in $f : \{1, \ldots, k-1\}^n \rightarrow \{1, \ldots, k-1\}$ represents NZMDD. This decision diagram only maps the non-zero minterms in the multi-valued logic network. Hence, the edges from a node which are relevant to the non-zero minterms are considered and mapped to the RMVLN. From root to terminal, the nodes are defined from a set

$N=\left\{{}_0N_0,{}_0N_{1,}\ldots,{}_rN_n\right\}$ where $r=\{0,1\}$. The node contains fundamental cells. Based on the incoming and outgoing edges, the cell is selected.

Each internal node has minimum one outgoing and maximum $k-1$outgoing edges, where for a particular edge $e_x \rightarrow T_t\left(x_i\right)=\{1,\ldots,k-1\}$. Each ${}_rN_n$has an if-else clause to determine the flow of incoming edges. The flow leads to the non-zero terminal of the decision diagram. NZMDD does not contain vertices with isomorphic sub-graphs or with all successors pointing to the same node. On all such paths of this DD the variables are not encountered in the same order. Hence, NZMDD is reduced but not ordered. The remaining of this paper focus on reduced NZMDD.

The function $f(x_1,x_2)=[012111000]$is a two variable three valued vector presented as shown Table 8-1. Figure 8.1 exhibits a diagram of a reduced and ordered MDD of the same function and Figure 8.2 represents diagram of reduced NZMDD.

Table 8-1: Matrix of function $f(x_1,x_2)$

Functionf_1	0	1	2
	Column 1	Column 2	Column 3
b		A	
0	0	1	0
1	1	1	0
2	2	1	0

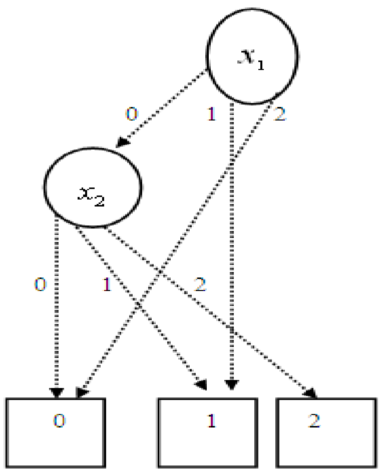


Figure 8.1: Ordered-reduced MDD

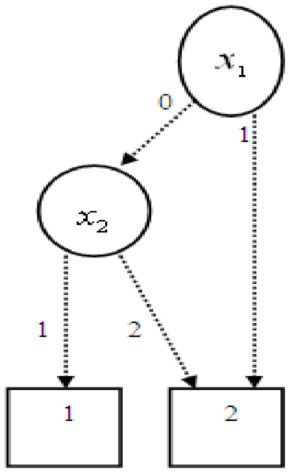


Figure 8.2: Reduced NZMDD

A directed acyclic RMVLN can be constructed based on function $f(x)$ for which a corresponding NZMDD can be generated as below:

I. Nodes are labeled with II and IO. Terminal nodes ${}_0N_0, {}_0N_{1,} \ldots, {}_rN_n$ are created, where the value of N is $N \in T_t(x_i) = \{1, \ldots, k-1\}$.

II. In reduced RMVLN for each of the II a vertex or variable is created in the NZMDD. The node checks all non-zero minterm $\sum_{i=0}^{n} f(x_1^i, x_2^i)$.

III. Based on the nodal clause the outgoing edge points to the terminal nodes. The logic gates of the reduced RMVLN are visited in a topological order. Each of the logic gates has its corresponding NZMDD operation which is carried out.

The two rows of reduced RMVLN assure a complete visit of all the nodes in a topological manner. This also ensures that all input edges are known before it is evaluated through logic gate expression. After visiting all the nodes the $IO's$ of the NZMDD is evaluated.

In Figure 8.3 an example of a nodal simulation of NZMDD is shown through logic gates. The logic gates consists of EXTENDED AND and MVL MAX gate. The input to gate corresponds to the outgoing edge of x_1 and outgoing edge of x_2. Input to the MAX gate depends on the repeated incident edge of x_2. The output edges of the gates correspond to the nodal relation that is represented by the NZMDD.

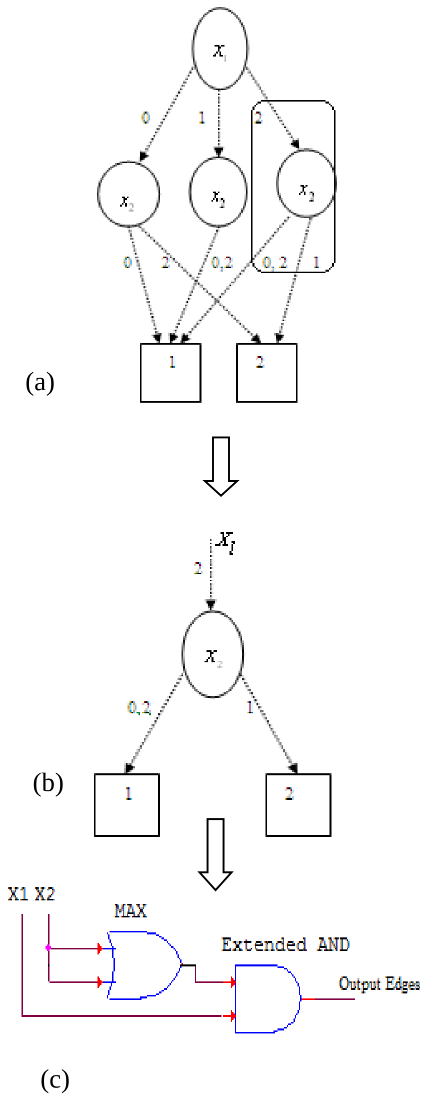


Figure 8.3: A nodal simulation of NZMDD using MAX and EXTENDED AND

8.3 NZMDD Synthesis of MVL Benchmarks

Consider an NZMDD representing a k valued two variable function which is initially provided for the synthesis purpose. The function $f(x)$ is the cumulative set of all sub-functions of the main $f(x)=\{Z_0(x)+Z_1(x)+\ldots+Z_n(x)\}$. Each of the sub-function's representing intermediate nodes of NZMDD. Root node and Terminal nodes are not considered as sub-functions of the DD. Each $Z_n(x_2)$ consist of variable x_2 with input edges from x_1 and outgoing edges from the node are mapped to a set of logic gates. The incoming edges and outgoing edges of $Z_n(x_2)$ is represented as $e_{a_0}, e_{a_1}, \ldots, e_{a_n}$ and $e_{x_0}, e_{x_1}, \ldots, e_{x_n}$. If the function being computed is $f(x)$, then the number of logic gates require can be represented by $\left(\sum_{i=0}^{n} Z(x_i)\right)\cdot\left(\sum_{j=0}^{m} e(x_j)\right)$. All the sub-functional output edges are input to a MAX gate. The output from the MAX gate is the final output of the function $f(x)$. In order to compute the functions, the basic gates used are EXTENDED AND, MIN, MAX and WINDOW LITERAL. It is assumed that each intermediate node is referring to a fundamental logic cell, where the inputs are from incoming edge $e_{a_0}, e_{a_1}, \ldots, e_{a_n}$, outgoing edge $e_{x_0}, e_{x_1}, \ldots, e_{x_n}$ and output is terminal node $T_t(x_i)=\{1,\ldots,k-1\}$.Each incidental repeating outgoing edge of intermediate node of NZMDD is translated to EXTENDED AND and MAX gate. Other outgoing non-incidental edges are translated into MIN gate. Output edges of EXTENDED AND and MIN serves as the final input for MAX2 gate. The MAX2 gate produces the final value for the specified sub-function.

As shown in Figure 8.4 below, three inputs to the MAX1 gate originate from the repeating outgoing edge of x_2. The predecessor edge of x_2 and output from MAX1 gate serves as the input to the EXTENDED AND. Non incidental outgoing edge for x_2 and predecessor edge serves as the input for MIN gate.

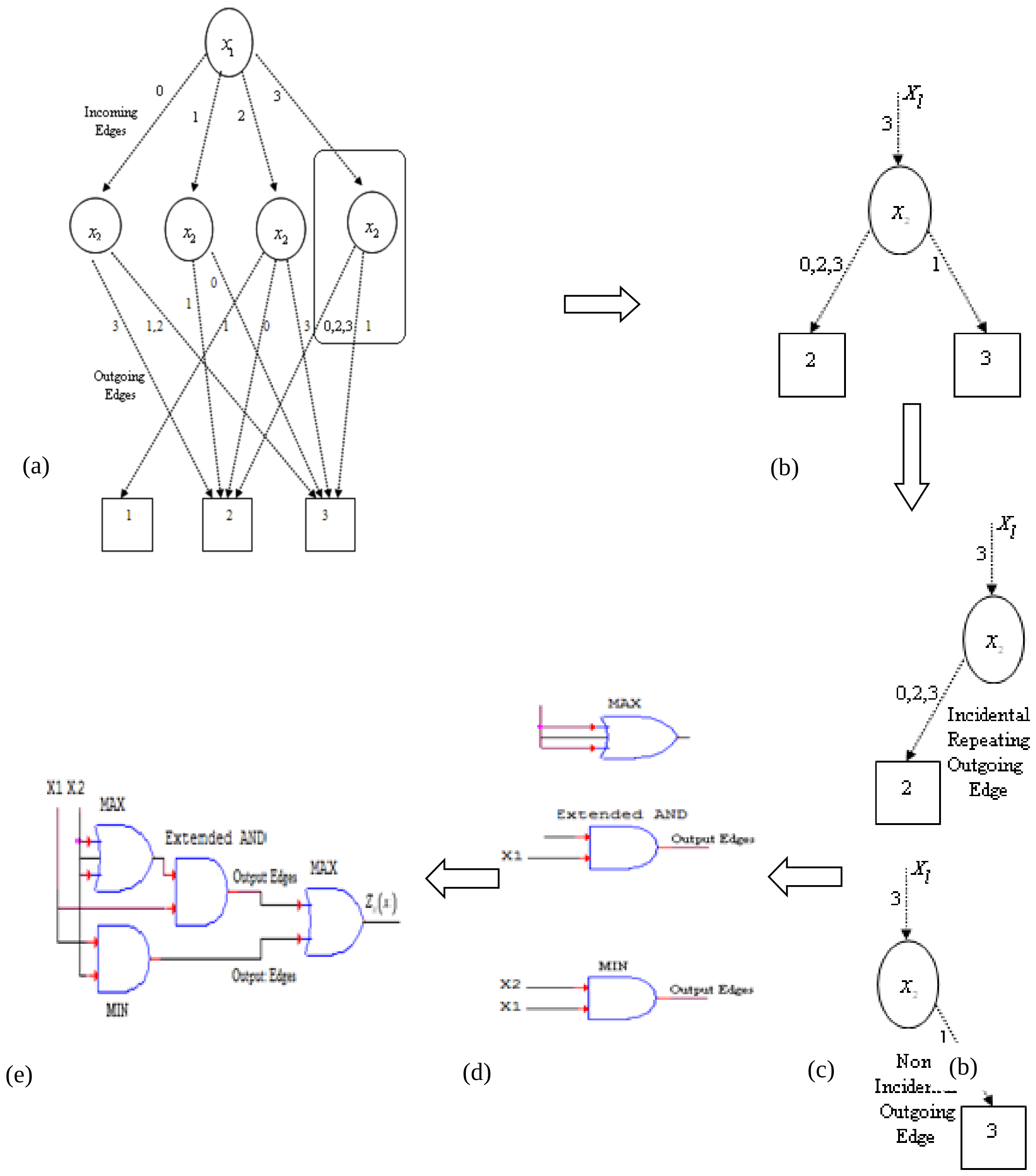


Figure 8.4: A 4 valued 2 variable MVLN, its sub-functional synthesis to MVL gates

The circuit portion in Figure 8.4(d) and 8.4(e) shows the relevant inputs and gates related to outgoing edge e_{x_2} *and* e_{x_3}, predecessor edge e_{a_3} and intermediate node $Z_3(x_2)$. Considering the NZMDD in Figure 8.4 if a realization for node $Z_3(x_2)$ is to be constructed, incoming edge e_{a_3} and outgoing edge e_{x_2}, e_{x_3} are needed. Redundant single input-output WINDOW LITERAL gates can be replaced with a single pass. MAX gates which determine the final output of the function $f(x)$ can be shared across all the output edges of the sub-functional nodes.

For a k valued logic level each intermediate node in the NZMDD can accommodate at max $k-1$ outgoing edges, $\frac{k}{2}$ incidental repeating outgoing edges. A worst case scenario may appear when at least one outgoing edge is incidental repeating. In these circumstance the total number of logic gates required to express a sub-functional intermediate node can be observed by $2\cdot\left(\sum_{i=0}^{n} e_{x_i}\right)$ and $2\cdot\left(\sum_{i=0}^{n} e_{x_i}\right)+1$. Using EXTENDED AND in all the outgoing edges decreases the number of MIN gates but increases the number of MAX gate. The best outcome has been achieved when at most one of the edge has utilized EXTENDED AND operator.

8.4 Experimental Analysis

An NZMDD analyzer was constructed to generate experimental results using 19600 and 49998 benchmark circuits. This benchmark generating method was also used by Sarif and Abd-El-Barr (2006), Abd-El-Barr and Sarif (2006) and Chowdhury, Raj and Singh (2013). Two sets of experimental datum were generated using the analyzer. The first and second set consecutively consisted of 19600 and 49998 randomly generated MVL functions. The first set benchmarks were 3 valued 2 variable functions, while the second set consisted of 4 valued 2 variable MVL functions. The benchmark did not contain any sequential circuits.

All measurements were performed on a *Microsoft windows XP*, Intel Core2 duo 2.20GHz CPU, 0.98GB RAM machine. The 3 valued combinational benchmarks were used to observe the average number of product term (PT) needed to synthesize each MVL function provided in Table 8-2. The number of minterms the function contains is shown in the first column. The proposed NZMDD column denotes the average PT needed to synthesize the group of functions.

Table 8-2: Average PT used in 3-valued MVL benchmarks using NZMDD method

	Generated	**NZMDD (proposed)**
Minterm	**MVL Functions**	**Avg. PT**
9	799	5.214
8	3067	4.968
7	5178	4.645
6	5319	4.273
5	3338	3.823
4	1461	3.261
3	438	2.625

Chowdhury, Raj and Singh (2013) had experimented on synthesizing MVL functions of 4 valued 2 variable using evolutionary techniques. It can be clearly observed that evolutionary algorithm had outperformed all other previous synthesis techniques. Considering the experimental results shown in Table 8-3, it is observed that the proposed method synthesizes an average 10.5 minterm 4-valued MVL functions for an average of 4.975 PT. ACO-MVL required an average of 6.286 PT for the same experiment. The synthesis using NZMDD reduced the number of PT by 52.62%, while ACO-MVL reduced PT by 40.13%. Hence the proposed NZMDD synthesis method outperformed ACO-MVL algorithm by 12.49% of improvement. Further reduction in synthesis could be achieved if the NZMDD could be reduced to its optimal size.

Table 8-3: PT reduction using NZMDD over ACO-MVL for 4-valued MVL functions

Minterm	**MVL Functions**	**ACO-MVL**	**NZMDD (proposed)**
16	500	6.730	6.316
15	2679	7.054	6.124
14	6589	7.163	5.852
13	10585	7.182	5.719
12	11230	7.086	5.483
11	9003	6.904	5.210
10	5434	6.660	4.782
9	2575	6.356	4.626
8	1038	5.934	4.429
7	277	5.480	4.094
6	75	4.960	3.987
5	13	3.923	3.077

As NZMDD managed to reduce PT better than ACO-MVL and other synthesis techniques, this will result in the reduction of logic gates used. With smaller number of logic gates, there will be less hardware involves in the logic circuit realization. In return, reduction of input to output delay as well as the economic cost involving the use of extra hardware can be achieved with the use of NZMDD method in synthesizing MVL.

8.5 Summary

Decision Diagrams (DD) was known to be one of the effective ways to synthesize MVL. In this chapter, a presentation and evaluation of the tree-like decision diagram have been made. NZMDD was considered as an extension to MDD, which synthesizes reduced MVL functional networks. The synthesized MVL functions were realized with MVL gates. The compact form of NZMDD reduced the size of logic expression, which in turn reduced the size of the circuit.

The results obtained using the NZMDD was compared using randomly generated 49998 non-sequential benchmark circuits against ACO-MVL evolutionary algorithm. The results achieved have shown that the NZMDD outperformed the existing evolutionary ACO-MVL algorithm in the paper by Chowdhury, Raj and Singh (2013). The result depicted an improvement of 12.49% for reduction in PT for synthesizing 4-valued MVL functions compared to ACO-MVL algorithm.

Chapter 9

Implementation and Realization at Gate and Transistor Level

In this chapter of the thesis, Multi-Valued Logic (MVL) gates are represented using traditional logic gate elements. The logic gates are used to realize MVL functions. Furthermore realization of MVL operators are performed by using N-channel Metal Oxide Semiconductor (NMOS), P-channel Metal Oxide Semiconductor (PMOS), transistors, resistors, diodes and comparators.

9.1 Introduction

Logic gate representation of synthesized MVL function is important, as it provides a clear view of hardware requirement. The logic gates differ in their functionality over conventional binary gates. It is observed that MVL realization requires lesser logic gates compared to the existing MVL architecture. For MVL systems, there are various logical operators, mainly MIN, MAX and EXTENDED AND. These logical operators operate on the WINDOW LITERALS which represent the minterm of MVL functions, or in other words it operates on the coordinates of the minterm of a MVL matrix. Operator MIN signifies the Product Term (PT) in function representation. Hence, an initiative was undertaken to reduce MIN operator in function realization, by substituting EXTENDED AND operator.

In this chapter, synthesized MVL expressions from High Deduction Algorithm–MVL (HDA-MVL) algorithm are used for realization using proposed logic gates. Each row of MVL function represents an implicant. Each of these implicants are synthesized using EXTENDED AND operator. A three valued two variable function can generate up to maximum three implicants. MVL functions and their implicants are presented before converting to gate level. Each implicant and its relevant gate expression are carefully scrutinized before merging all of them with MAX gate. Realization reveals better result compared to existing algebraic synthesis techniques.

Furthermore, MVL Function realization is shown achievable by implementing logic gates in transistor level. During implementation voltage mode architecture is considered. In voltage mode, MVL levels are represented by voltage levels in terms of a base voltage value. The initial base voltage value, v_b is set to 1.5 Volt for the experimental analysis. Logic level *l* corresponds to an interval of the continuous quantity *x*. Hence, level 0 represents the null value and level 3 is associated with v_b = 4.5V and so on.

9.2 Logic Circuit Presentation at Gate Level

Most of the logic operators requires two inputs and prompts one output. Exceptional operators such as EVEN, ODD and INVERTER requires only one input and output. Throughout this thesis, in various MVL function realizations these logic operator elements are used. Romero, et al.(2009) in the research article presented basic MVL operators and utilized them for realization. Inspired by the paper, an initiative was undertaken to present similar approach for the logic circuit elements. Extended and novel logic operators are introduced to reduce gate count during the synthesis procedure as an improvement to the MVL operators presented by Romero, et al. (2009). Some of the basic logic circuit elements of these logical operators are shown below in Figure 9.1.

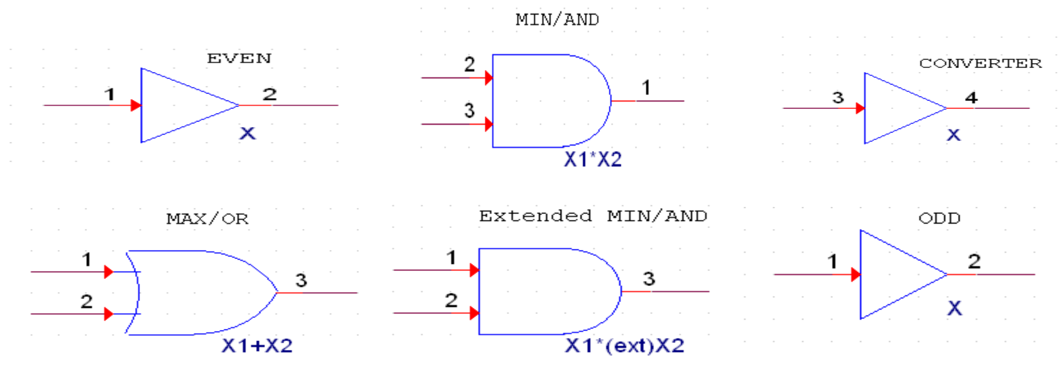


Figure 9.1: Representation of essential logic circuit elements

Romero, et al. (2014) in his paper represented MVL functions of 4 valued 2 variable using algebraic synthesis methodology. Below in Table 9-1, a MV logic function *G(a₁,a₂)* and logic circuit representation of function *G(a₁,a₂)* from his paper is elaborated.

Table 9-1: Example of a MVL function f_1

Function f_1	0	1	2
	Column 1	Column 2	Column 3
b		a	
0	0	2	1
1	1	0	2
2	2	1	0

Synthesized MVL functions are represented with the logic circuit elements. These circuit elements are realized at transistor level to observe the realization and behaviour. The synthesized functional expression for the above function is represented as below. A series of EXTENDED AND is expressed in the form of SoP. Table 9-2 represents $G(a_1, a_2)$

$$Synthesized(f_1) = {}^{0}\vec{x}_1^{\,1,2}({}^{1}x_2^{\,1} + {}^{2}x_2^{\,2}) + {}^{1}\vec{x}_1^{\,2,1}({}^{0}x_2^{\,2} + {}^{2}x_2^{\,1}) + {}^{2}\vec{x}_1^{\,1,2}({}^{0}x_2^{\,1} + {}^{1}x_2^{\,2}) \quad (9.1)$$

Table 9-2: Example of a MVL function $G(a_1, a_2)$ from (Romero, Martins and Santos 2009)

Function $G(a_1,a_2)$	0	1	2	3
a_2	a_1			
0	1	0	0	0
1	0	3	2	1
2	0	2	3	0
3	0	0	0	0

MVL function f_1 was realized with the pre-defined logic operators as shown below in Figure 9.2. The entire system had two inputs and one output. Intermediate or successor operators were the first layer of operators which initiated inputs to the second layer. The second layer consisted of OR logic operator. The output from the OR operator and another input went into EXTENDED AND operator.

The other input was always generated from input x_2. A total of three EXTENDED AND operator generated three outputs. Each of these operators represented an implicant from the main function. Three of the outputs from the operators were fed into a three input MAX operator. The MAX operator generated the final output of the entire MVL function. Instead of a three input MAX operator, a two input MAX operator could be also considered. If a two input MAX was considered, the amount of MAX operator in the design would increase by 20%. The successor or first layer operators occupied 46.15% out of total operators in the design. On the other hand, 23.08% of the MIN operators and 30.77% MAX operators were used in the entire design of the MVL function. Existing MVL logic circuit implementation of the function *$G(a_1, a_2)$* has been shown in Figure 9.3.

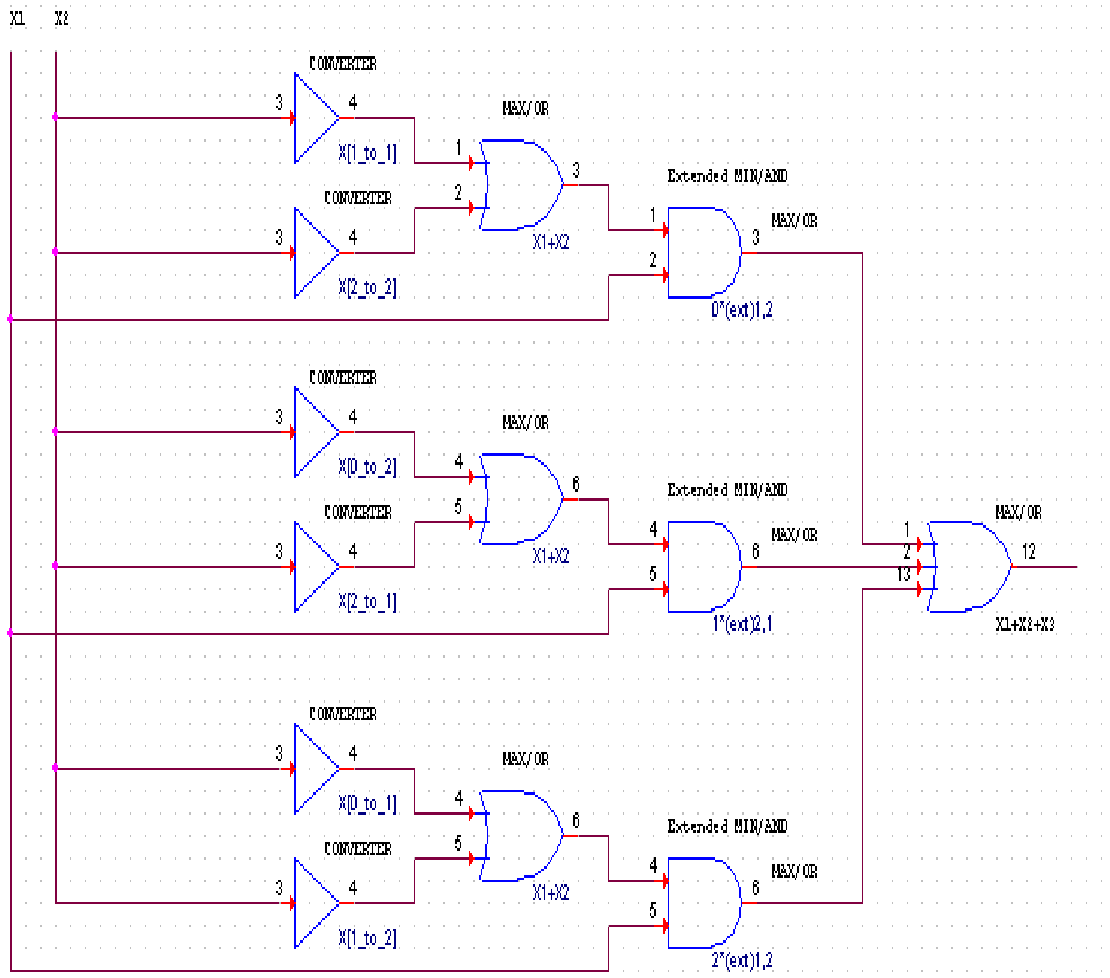


Figure 9.2: Realization of function f_1 using logic circuit design

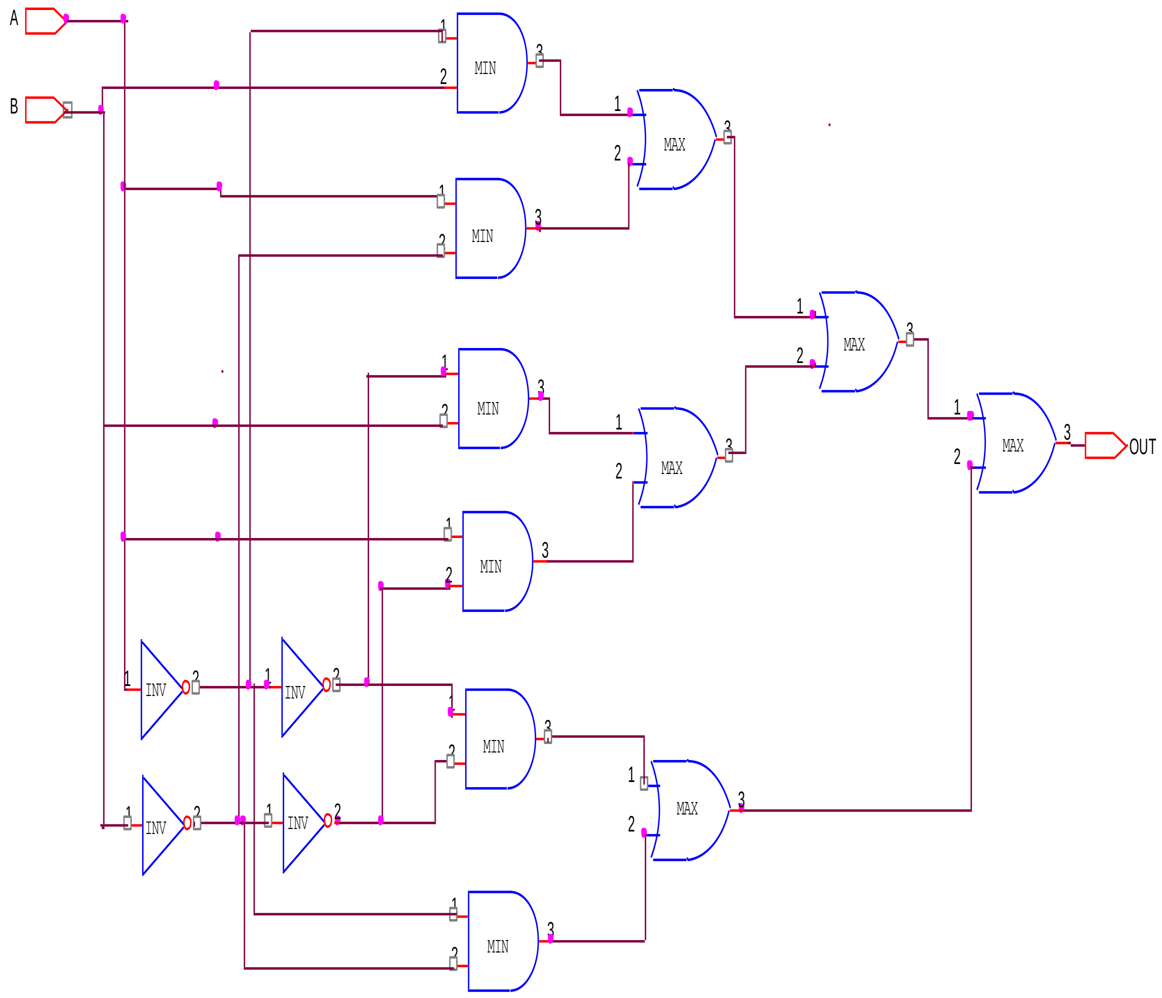


Figure 9.3: MVL circuit implementation according to Romero, Martins and Santos (2009)

The circuit representation clearly revealed that six MIN operator was used along with five MAX operators. Number of inputs were two and number of intermediate operators or successor operators were four. HDA-MVL algorithm syntheseized the *$G(a_1, a_2)$* function, after synthesizing MVL functional expression was retrieved. The logic representation of the proposed methodology is shown as Figure 9.4 below.

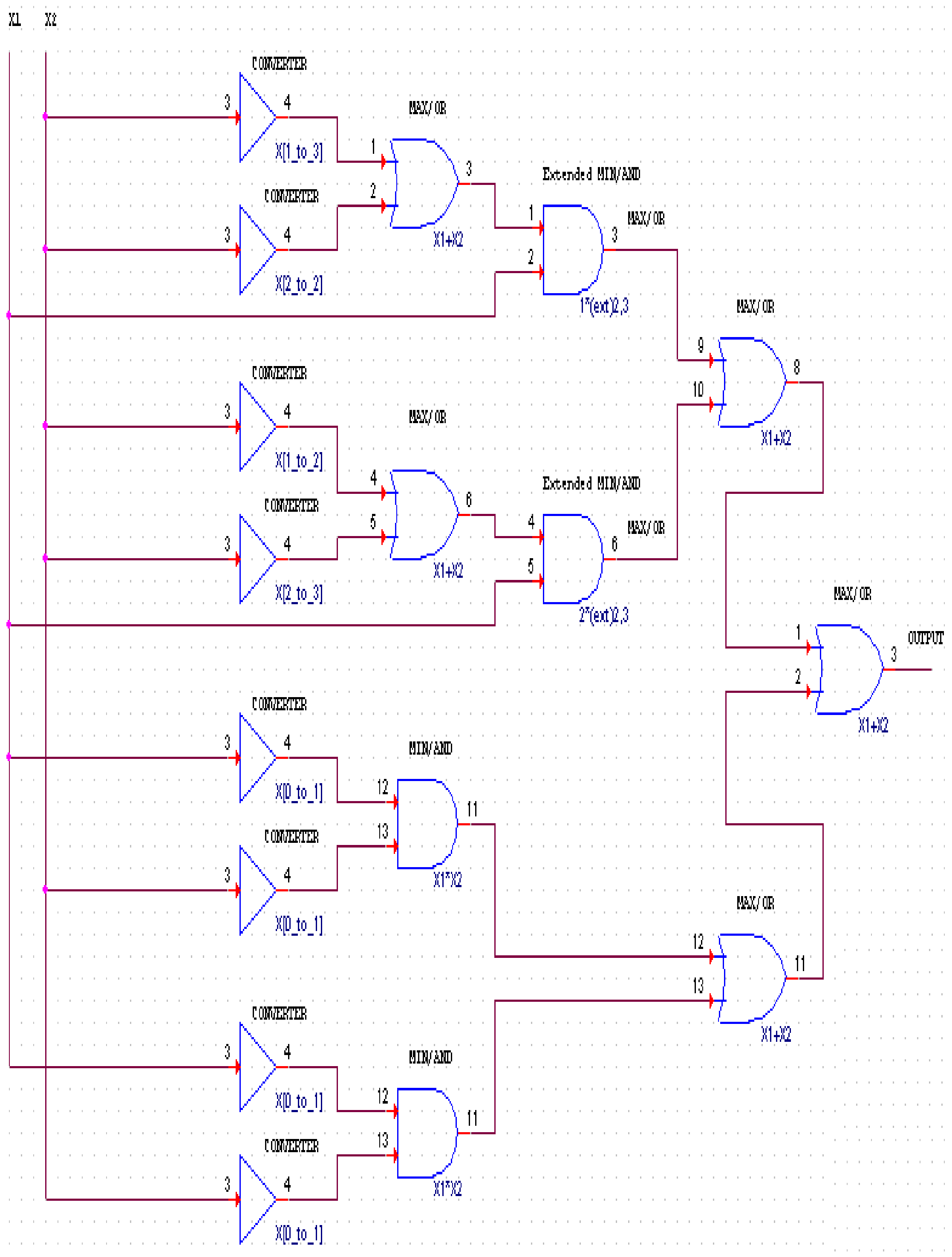


Figure 9.4: Logic circuit representation according to the proposed method

9.3 MVL Operator Realization at Transistor Level

In this section, different MVL operators are widely discussed and realized on transistor level. Level 3 models of NMOS and PMOS transistors are used to build the converter logic. Threshold voltage V_{t_0} =0.8V is set for NMOS transistor whereas V_{t_0} =-0.9V is set for PMOS transistor. Diode-switch model is used to represent some of the circuits. Circuits like EXTENDED AND is a combination of both Diode-Switch (DS) and Metal Oxide Semiconductor (MOS).

A maximum of 4 valued 2 variable functions is investigated for experimental purpose. For a maximum of 4 value, logic level *l*=0 will represents v_b = 0V, *l*=1 will represents v_b = 1.5V, *l*=2 will represents v_b = 3V and *l*=3 will represents v_b = 4.5V. All the experiments are conducted in ORCAD CAPTURE CIS Lite tool. Simulation results are investigated to show the expected output of each MVL operator. ODD, EVEN, MIN, MAX, differential comparator, INVERTER, CONVERTER and EXTENDED AND MVL circuits are also used in the experimental analysis. ORCAD Capture tool is used to design the circuits with level 3 transistor parameters.

9.3.1 MIN circuit

Design of this converter needs a circuit that gives maximum outputs of two voltages. Two diodes (D8-D9) are used as switches. The diode with the lowest input voltage will turns on. Considering D8 has the lowest voltage, it turns on its diode raising the voltage at the output and the other diode D9 is turned off. Construction of the circuit is needed to replace the ground with a voltage which is more than the input voltage can become (Max, Min, Average Circuits 2010). The MIN circuit is shown in Figure 9.5.

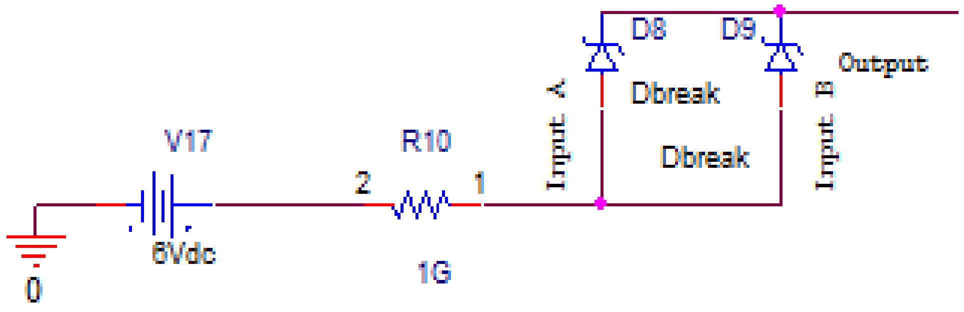


Figure 9.5: Circuit realization of a MIN logic operator (Gawande and Ladhake 2008; Max, Min, Average Circuits 2010)

9.3.2 Differential comparator circuit

The schematic of comparator is shown in Figure 9.6. The differential comparator comprises of MOS transistors M1-M8 which compares the level-shifted input A with the fixed reference voltage. The reference voltage differs from circuit to circuit as different realization of voltage level requires different reference voltage.

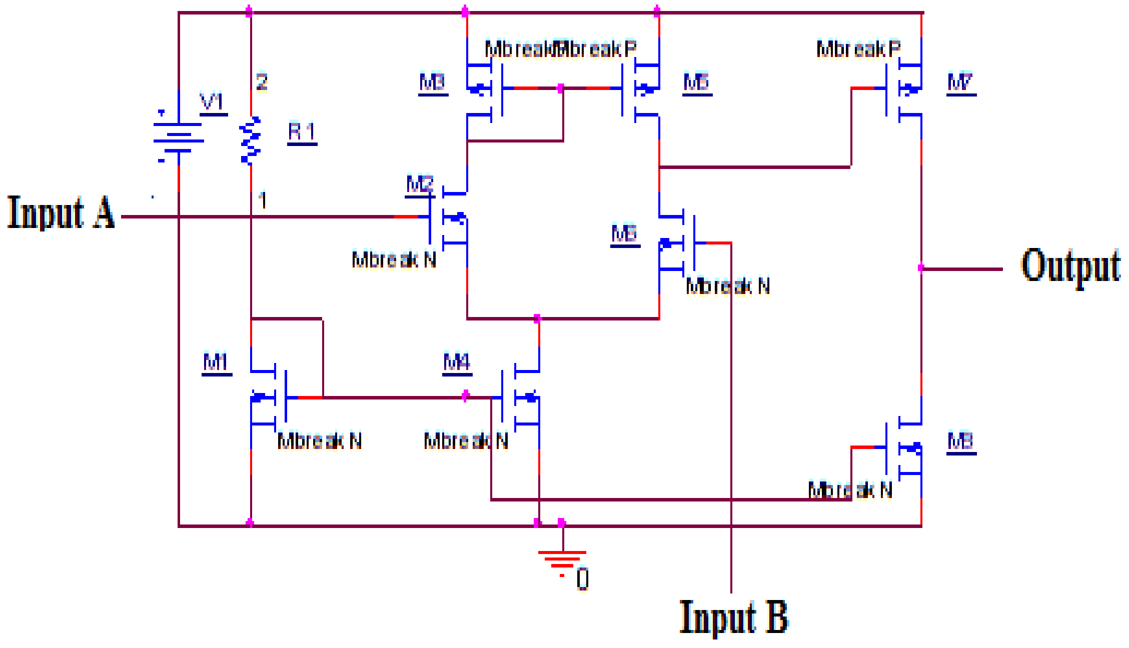


Figure 9.6: Differential comparator

For realizing ${}^{1}x_{1}^{1}$ WINDOW LITERAL the reference voltage is set to 0.73 volt which is set for logic 1, i.e. 1.5 volt. The output of comparator is fed to MIN circuit. The other input of the MIN circuit is the level shifted input A. The level shifting of a signal occurs due to the presence of diode. Voltage source V1 is the supply of the comparator. Input A is the positive terminal and Input B is the negative terminal of the comparator.

9.3.3 CONVERTER circuit

The CONVERTER is capable of converting a specific voltage to a different voltage level. This CONVERTER actually represents the output (voltage) of a WINDOW LITERAL of a MVL expression. Considering ${}^{2}x_{1}^{1}$, if the input is 2 then the output is 1. Since in MVL voltage mode circuitry logic levels are represented with voltage, then l=2 would represent v_b = 3V and l=1 would represent v_b = 1.5V. That means the CONVERTER for this specific WINDOW LITERAL will output 1.5V, given the condition that the input voltage level is 3V. For all other voltage level, the output is 0V. ***Definition 4.2.1*** explains how a WINDOW LITEERAL steps up to an output. A CONVERTER is a representation of a WINDOW LITERAL.

Conversion: 0V to (0V, 1.5V, 3V, 4.5V) output

The circuit for 0V to 1.5V conversion is shown in Figure 9.7. VS1.txt generates the amount of voltage input to be supplied to the circuit to check the circuit output. The VDC is set to 1.5V to achieve the output of 1.5V which is of logic level 1.

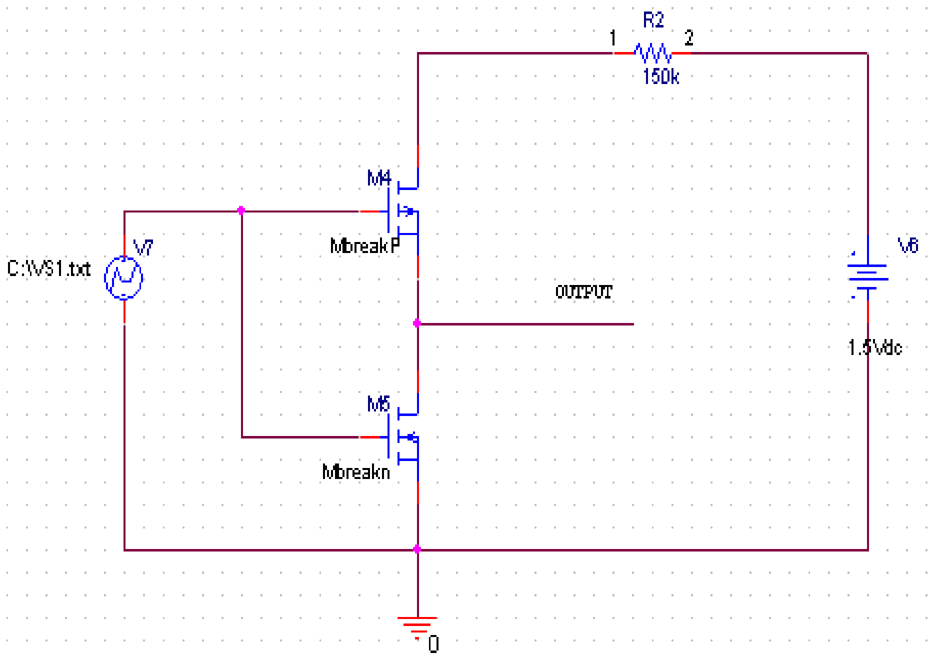


Figure 9.7: A typical circuit realization of 0V to 1.5V CONVERTER

The CONVERTER is designed for ${}^{0}x_{1}^{1}$ WINDOW LITERAL. The circuit only outputs 1.5V when the input voltage is 0V. For any other level of voltage input between 0-5V the output is always 0V. Two transistors (M4-M5) are used to control the output voltage level of the CONVERTER. If input is non zero, transistor M4 is switched off and M5 is switched on, hence ground potential switch the output to 0V. If the input is zero volts, M4 is switched on and 1.5V is drawn to the output from voltage source V6.

Conversion: 1.5V to (0V, 1.5V, 3V, 4.5V) output

The CONVERTER schematic is shown in Figure 9.8. VS1.txt generates the amount of voltage input to be supplied to the circuit to check the circuit output. V13 is the Direct Cover (DC) supply voltage of 5 volt of the comparator.

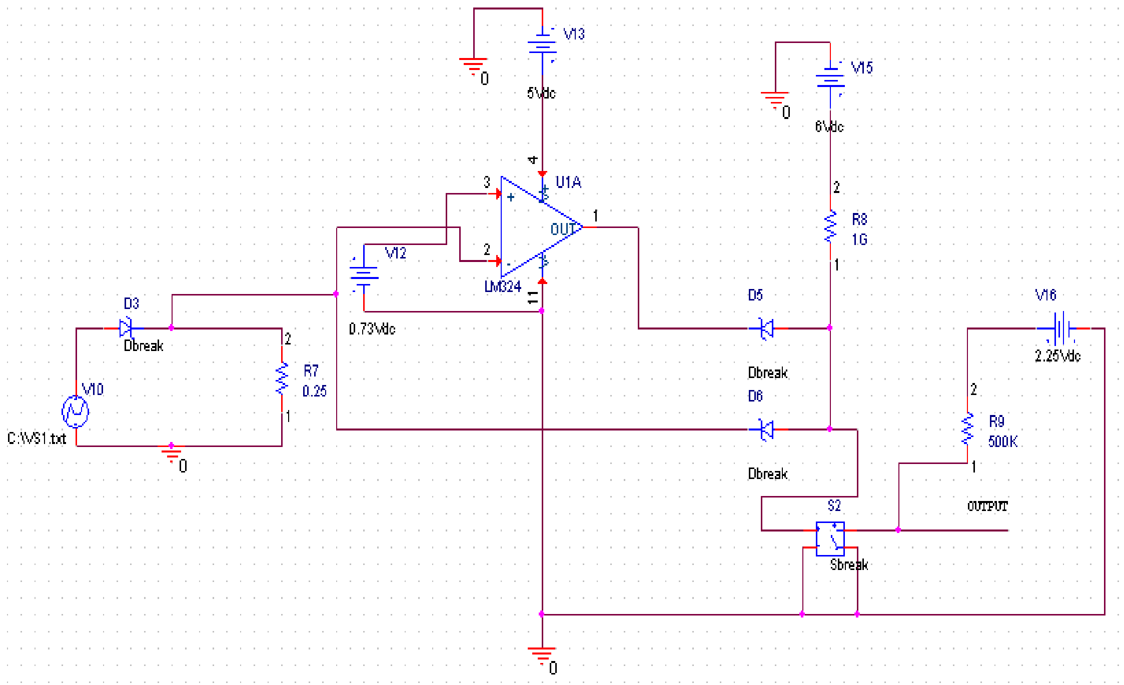


Figure 9.8: Circuit realization of a 1.5V to 1.5V CONVERTER

V15 is the supply for the MIN circuit set to 6 volt. A switch is used to control the final output of the circuit. The Sbreak VSWITCH model parameters are set as, Roff=1e6, Ron=1.0, Voff=0.73 and Von=0.0. The V16 VDC is set to 2.25V to achieve the output of 1.5V which is of logic level 1. V12 VDC = 0.73V determines the 1.5V input peak. There exists a 2.25V decrement or increment per logic level representation. For example, to represent *l*=2, the VDC needs to be set as 4.5V. The converter is designed for ${}^{1}x_{1}^{1}$ WINDOW LITERAL. The circuit only outputs 1.5V when the input voltage is 1.5V. For any other level of voltage input ranges between 0-5V the output is always 0V.

Conversion: 3V to (0V, 1.5V, 3V, 4.5V) output

The schematic for this conversion realization is shown in Figure 9.9. The CONVERTER's V12 is set to DC voltage of 0.73V which determines the 3V input peak. This V12 voltage supply is the only entity that differs between the 2nd and 3rdCONVERTER.

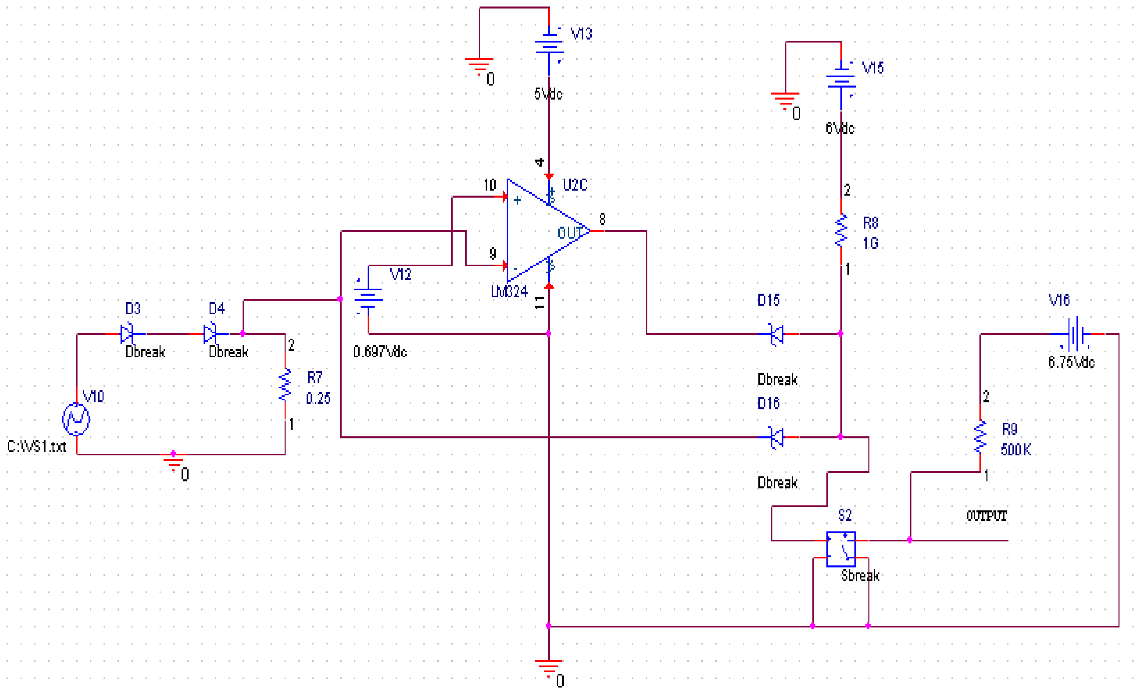


Figure 9.9: Circuit realization of a 3V to 4.5V CONVERTER

VS1.txt generates the amount of Voltage input to be supplied to the circuit to check the circuit output. V13 and V15 have the same supply voltage. A switch is used to control the final output of the circuit. The Sbreak VSWITCH model parameters are set as, Roff=1e6, Ron=1.0, Voff=0.73 and Von=0.0. The V16 VDC is set to 6.75V to achieve the output of 4.5V which is of logic level 3. There exists a 2.25V decrement or increment per logic level representation. For example, to represent *l*=2, the VDC needs to be set as 4.5V. The CONVERTER is designed for ${}^{2}x_1^{3}$ WINDOW LITERAL. The circuit only outputs 4.5V when the input voltage is 3V. For any other level of voltage input ranges between 0-5V the output is always 0V.

Conversion: 4.5V to (0V, 1.5V, 3V, 4.5V) output

Figure 9.10 shows the converter schematic. Like the earlier convertors discussed, this CONVERTER as well VS1.txt text file generates the amount of voltage input to be supplied to the circuit to check the circuit output. A switch is used to control the final output of the circuit. The Sbreak VSWITCH model parameters are set as, Roff=1e6, Ron=1.0, Voff=0.156 and Von=0.0. The V9 VDC is set to 4.5V to achieve the output of 4.5V which is of logic level 3. The converter is designed for ${}^{3}x_1^{3}$ window literal. The circuit only outputs 4.5V when the input voltage is 4.5V. For any other level of voltage input ranges between 0-5V the output is always 0V.

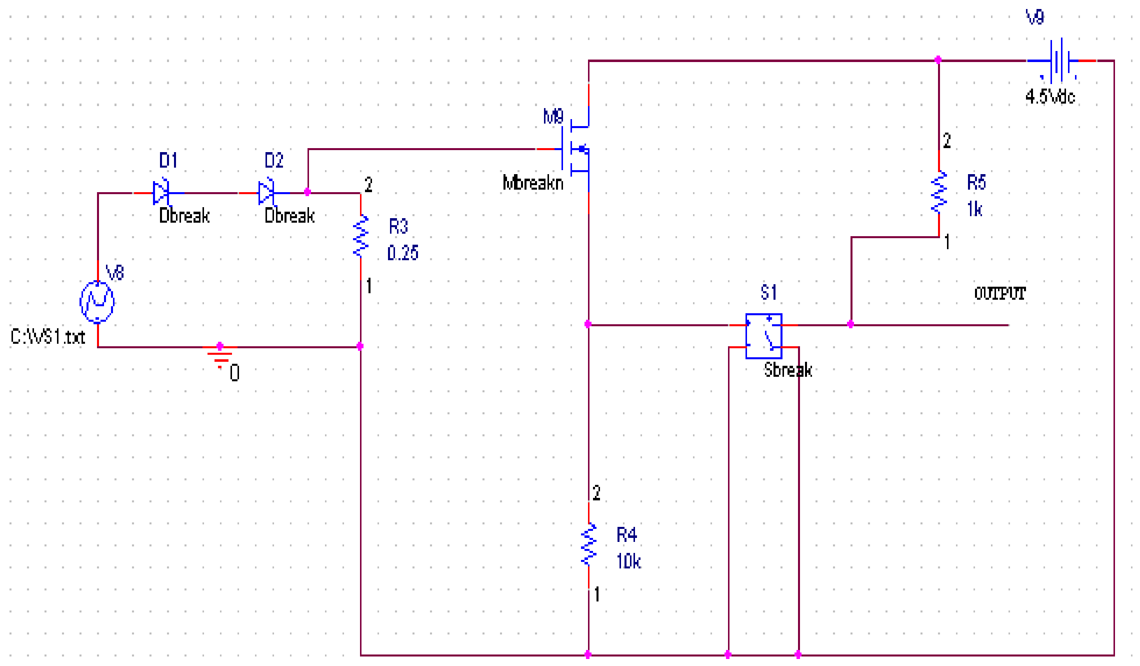


Figure 9.10: Circuit realization of a 4.5V to 4.5V CONVERTER

9.3.4 ODD and EVEN logic operator

The circuit is shown in Figure 9.11 which comprises of four diodes, two comparators, two min circuit, and CMOS OR gate to select the max number. The logic level is divided into 4 voltage levels where logic 0 is 0 volt, logic1 is 1.5 volt, logic 2 is 3 volt, and logic 3 is 4.5 volt. The diodes D1-D2 acts as level shifter which decreases the voltage level of input A in order to be compared to the reference voltage.

The reference voltage is set as such to depend on the reference which the circuit performs the logic level selection. The 1st comparator comprises of Metal Oxide Semiconductor (MOS) transistors M1-M8 which compares thelevel-shifted input A with the fixed refrence volatge 0.69 volt which is set for logic 2, i.e. 3 volt. The output of comparator is fed to MIN circuit in which other input is the level shifted input A. Depending on the MIN circuit input combination, the switch S1 gives the output of required logic according to the value of voltage V7. The voltage V7 is the desired voltage source of output.

The full sechematic of EVEN logic operator is illustrated as in Figure 9.11. Similarly, the diodes D3-D4 acts as level shifter for input B which acts as input to 2nd comparator made of transistors M9-M16. The 2nd comparator compares the level shifted input B to the fixed refrence voltage of 0.69 volt. The output of 2nd comparator is fed to MIN circuit and switch S2 outputs the required logic depending on V10. These two switch ouputs is fed back to standard CMOS OR gate made of M17-M22 which performs the required EVEN function, i.e. gives output for any of the input comprising the logic two an even number. For ODD logic operator only the CONVERTER block is changed.

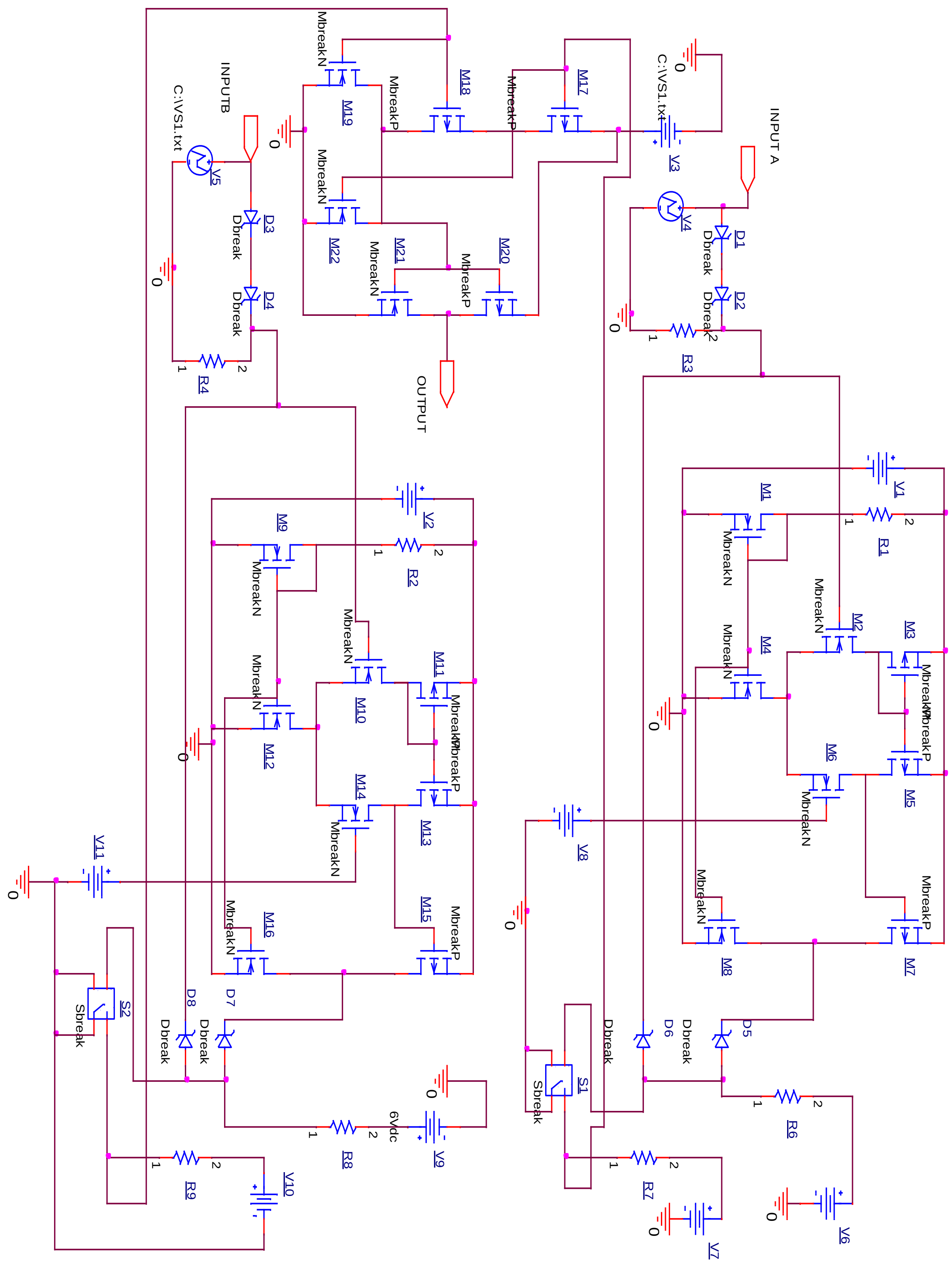


Figure 9.11: EVEN logic operator

9.3.5 EXTENDED AND Operator

The schematic to perform EXTENDED AND function is shown in Figure 9.12 where the MOS transistors M1 and M2 checks the condition for x_1 to satify 'a' which in this case is assumed to be 0 volt. Once the condition is satisfied, the output checks the condition for x_2. For x_2 variable, the S_1 and S_2 switch alternatively as per the condition gets true. The logic behind EXTENDED AND operation can be observed in (Nakahara, Sasao and Matsuura 2011). The switch S_1 and S_2 is set to value b_1 and b_2 respectively.

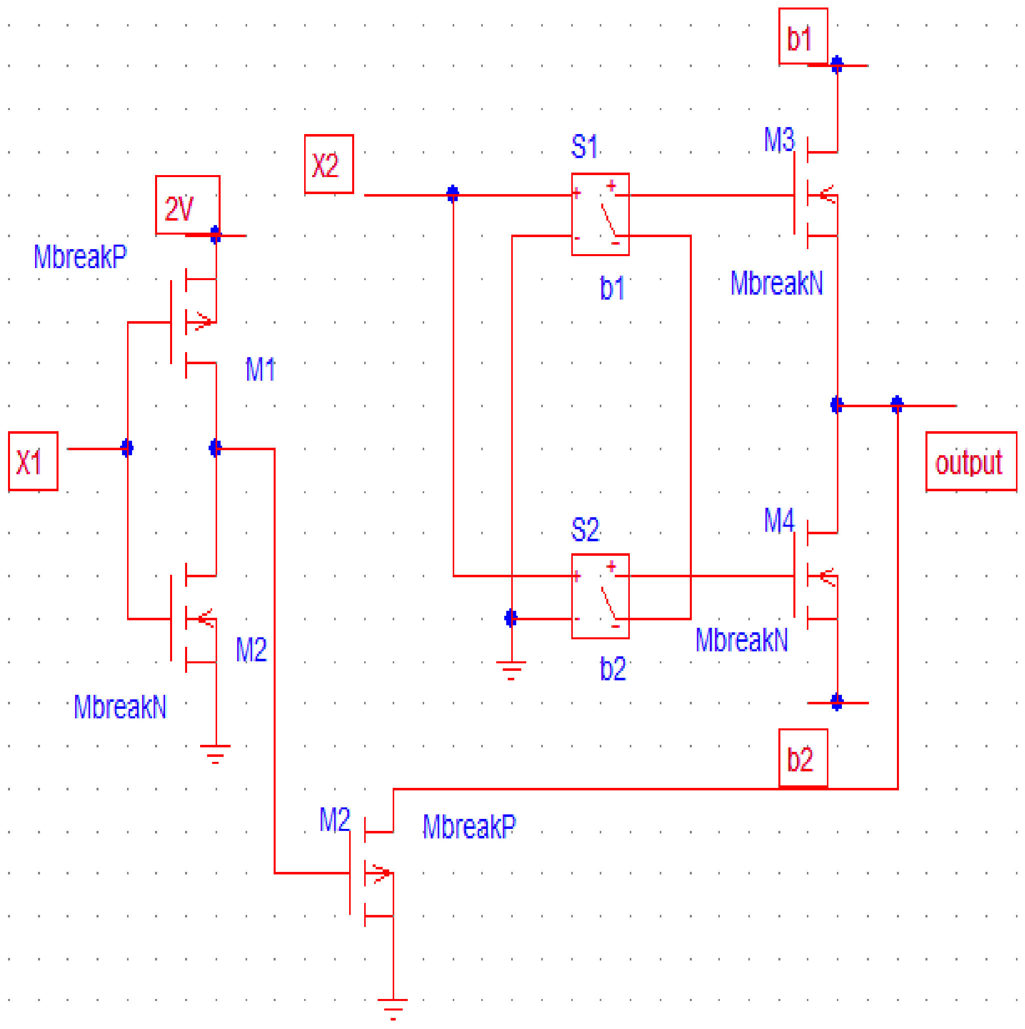


Figure 9.12: EXTENDED AND schematic for ${}^{0}\vec{x}_1^{\,1,5,3} \bullet x_2$

9.4 Simulation Results

ORCAD PSpice transient analysis simulation verified the functionality of the circuits discussed in the previous sections. All simulations used Cadence ORCAD Pspice MOS level 3 model and parameters. Below, simulation of a 0V to 1.5V CONVERTER is shown in Figure 9.13.

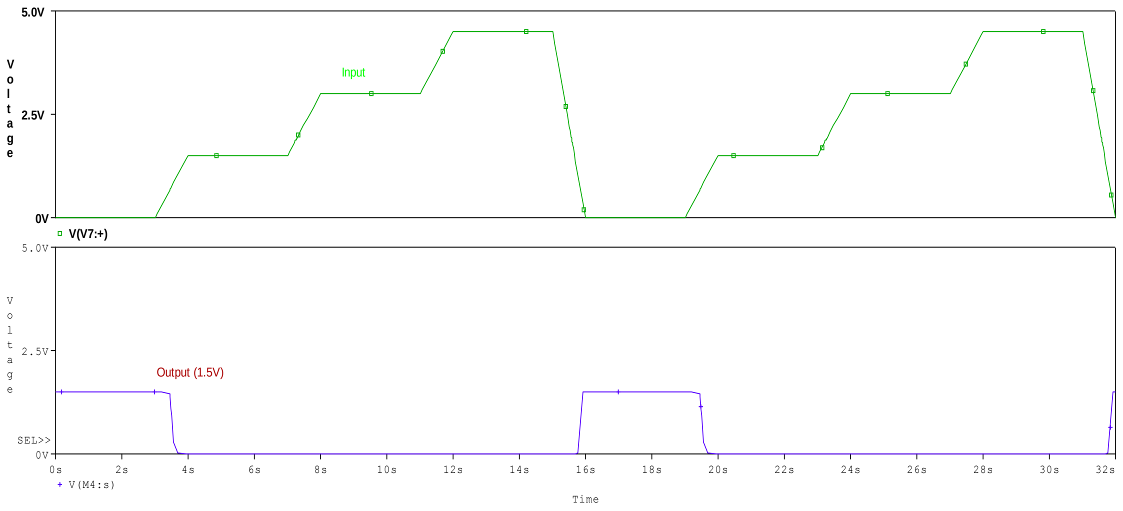


Figure 9.13: Transient analysis of the circuit realization of ${}^{0}x_{1}^{1}$ window literal

Simulation of a 1.5V to 1.5V CONVERTER is shown in Figure 9.14.

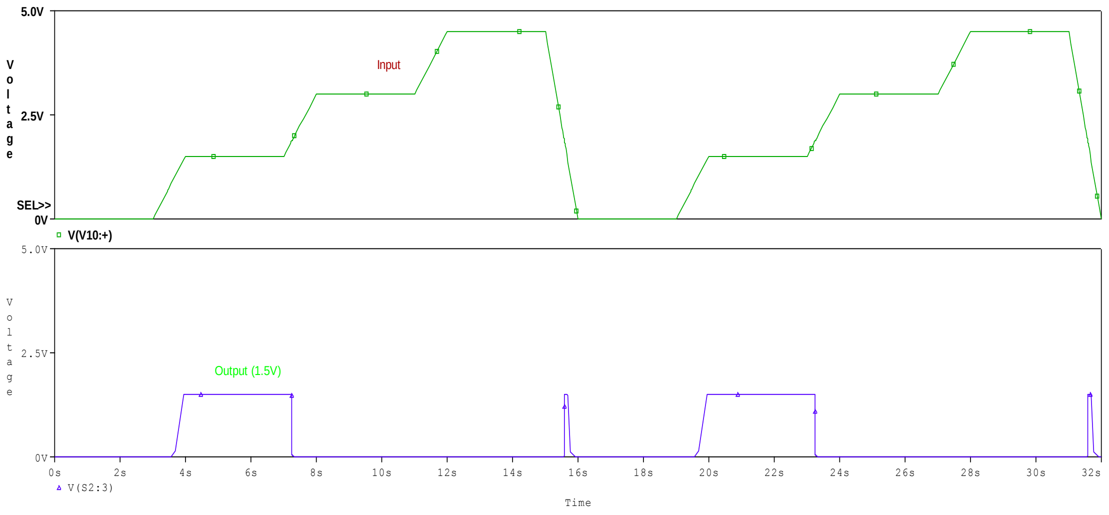


Figure 9.14: Transient analysis of the circuit realization of ${}^{1}x_{1}^{1}$ WINDOW LITERAL

Below a 3V to 4.5V and 4.5V to 4.5V CONVERTER simulation results is presented as a transient analysis in Figure 9.15 and Figure 9.16.

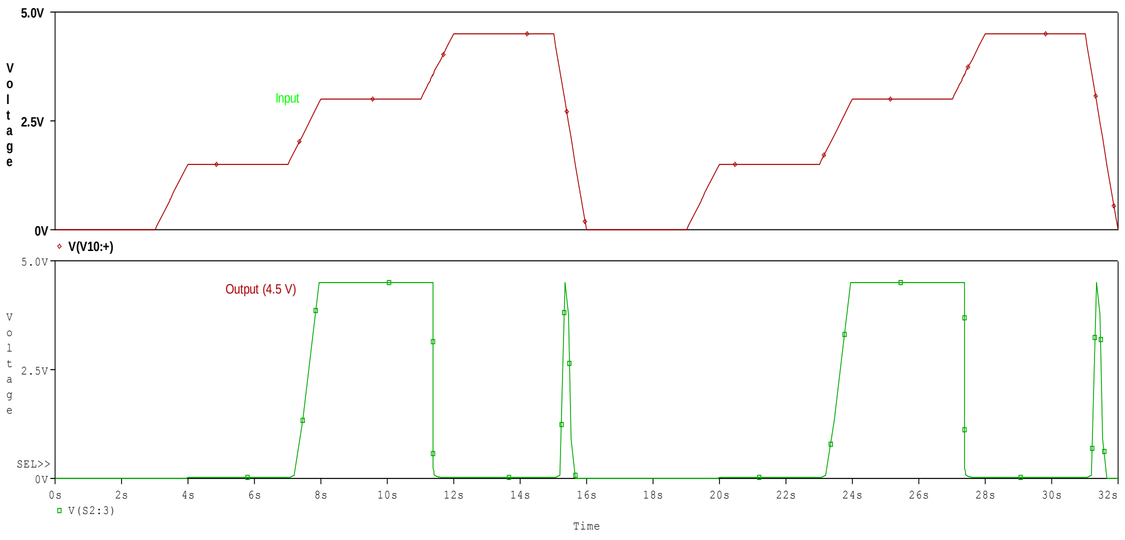


Figure 9.15: Transient analysis of the circuit realization of ${}^{2}x_{1}^{3}$ WINDOW LITERAL

The following simulation represents 4.5V to 4.5V CONVERTER,

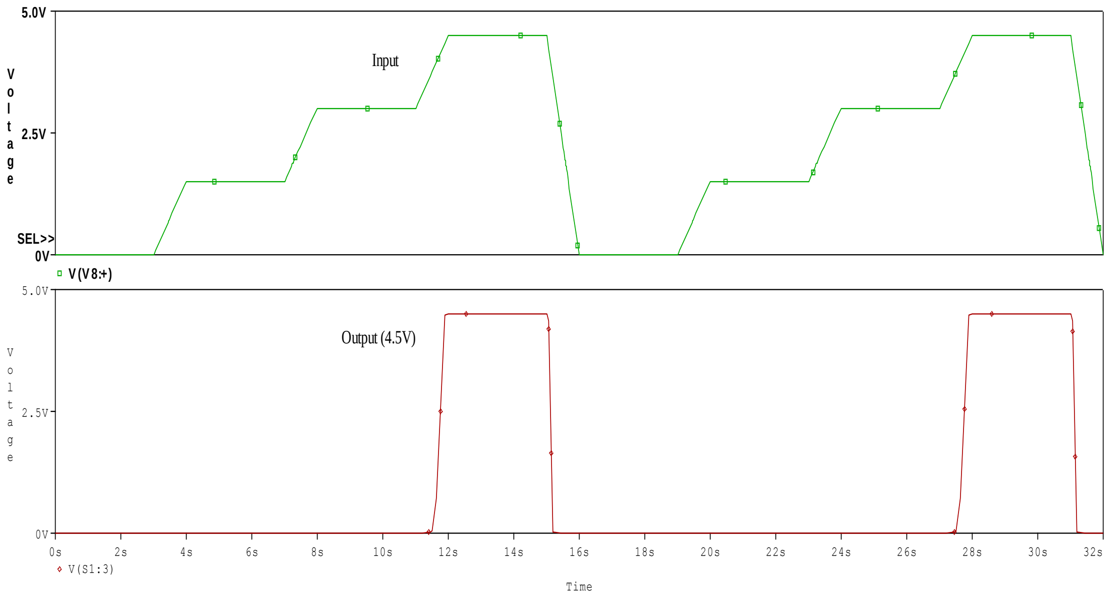


Figure 9.16: Transient analysis of the circuit realization of ${}^{3}x_{1}^{3}$ WINDOW LITERAL

Below, simulation of EXTENDED AND logical operator is presented. It was assumed that the variable 'a' and 'x_1' was taken as 0 volt. The value of b_1 and b_2 was set as 1.5V and 3V respectively. Depending on x_2 values, the switch S_1 and S_2 were activated and the output was switched between b_1 and b_2.

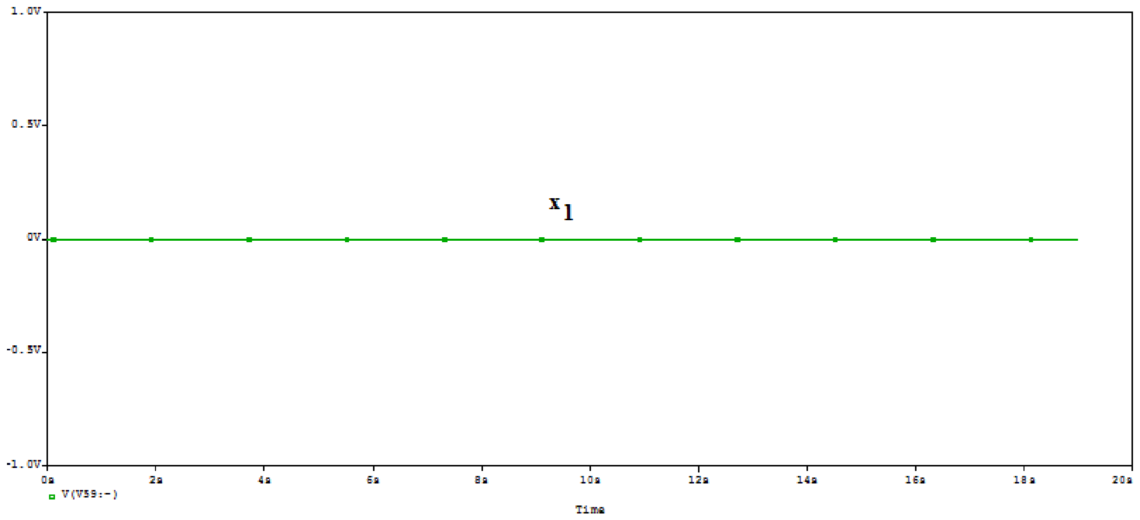


Figure 9.17: Transient analysis of the circuit realization of EXTENDED AND logic operator (Variable x_1)

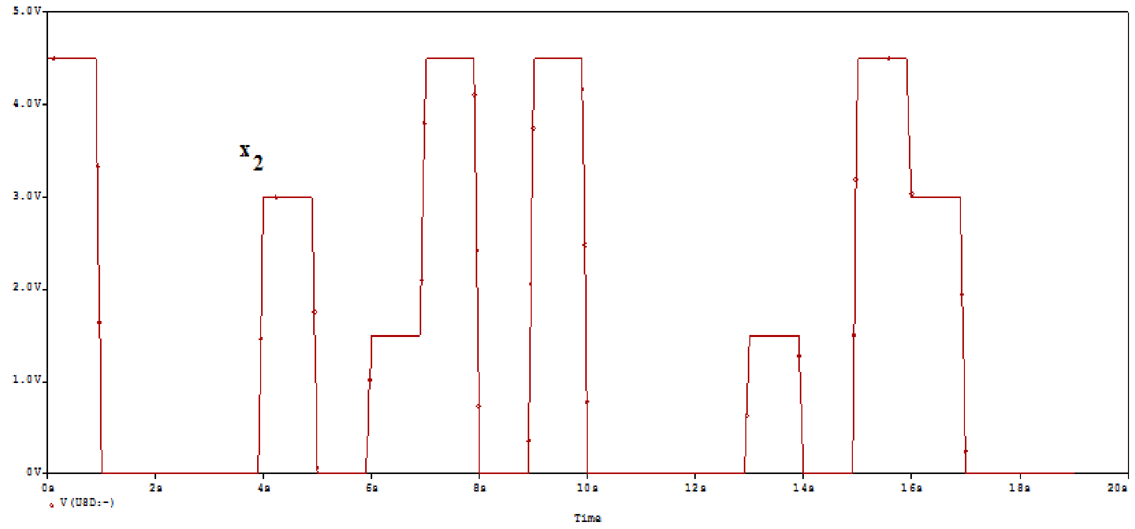


Figure 9.18: Transient analysis of the circuit realization of EXTENDED AND operator (Variable x_2)

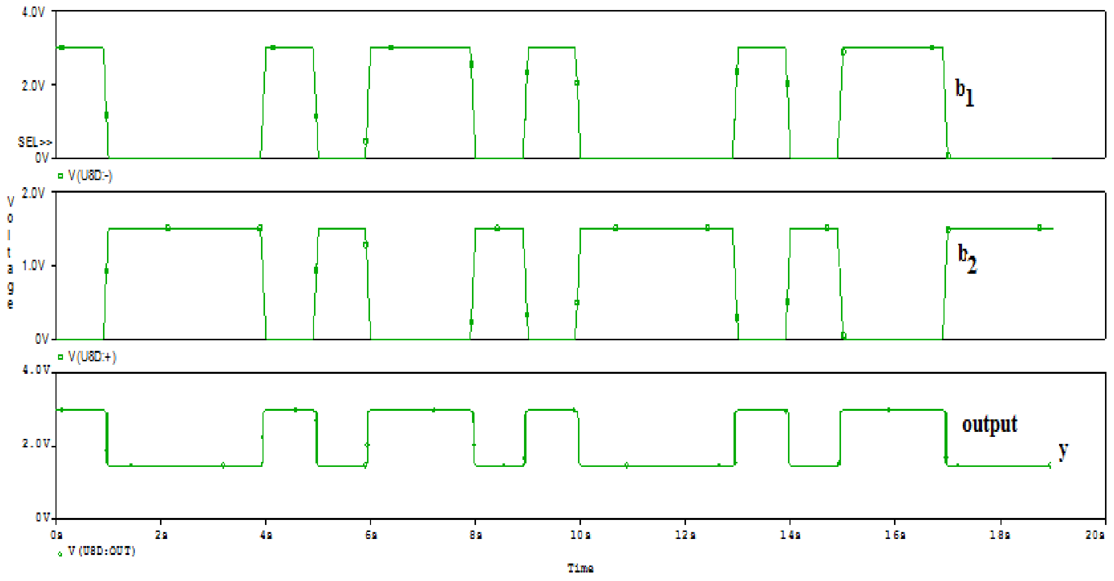


Figure 9.19: Transient analysis of the circuit realization of EXTENDED AND logic operator (Var: b_1,b_2 and y)

Considering the proposed circuit elements, the design had higher successor operator compared to the existing technique. According to $G(a_1,a_2)$, Figure 9.3 and Figure 9.4 establish different scenarios. The design ensured its stability in reducing MIN operator. Table 9-3 and 9-4 below shows an elaborative comparision on logic operator usage between the two methods.

Table 9-3: Operator usage during logic circuit representation of function $G(a_1,a_2)$

	In Design Implementation		Proposed Method against Romero, Martins and Santos (2009)	
	Proposed	Romero, Martins and Santos (2009)	Incline	Decline
MIN Operator	4	6	66.00%	-
MAX Operator	5	5	0.00%	0.00%
Successor Operator	8	4	-	50.00%
Interconnected Lines	27	27	0.00%	0.00%

Table 9-4: Comparison of work by Romero, Martins and Santos (2009) and proposed method

	Romero, Martins and Santos (2009)	Proposed Algebraic Method	Improvement (%)
Gate Input Count	2	2	0.00%
Gate Count (MIN/MAX)	11	9	18.00%

As explained above, novel logic operators are proposed then implemented using diode, NMOS and PMOS transistors. Upon implementation, the behaviour for the respective logic gate is observed through simulation result. The simulation results reveal that the output from logic circuit has less distortion and represents accurate logic depending on respective voltage level. The simulation results verify the characteristic of the logic circuits. The logic circuits are then further utilized for realizing logic expressions. Similar strategy is observed by Romero, Martins and Santos (2009).

In their research, Romero, Martins and Santos (2009) realized particular logic expression with MIN/MAX logic gates. The proposed EXTENDED MIN operator reduces the MIN gate count by 18% for similar example. Lesser logic gate count indicates a reduction of hardware utilization for the logic realization. Hence, this will minimize the cost for hardware used in realizing MVL functions at the same time increasing the efficiency in the realization process. As a practical implementation, the MVL operators can be further utilized to form arithmetic and logic unit.

9.5 Summary

Based on the experimental results it had been observed that the number of MIN gate decreased according to the proposed methodology. The logic circuit representation revealed that four MIN operator was used along with five MAX operator. An overall reduction of 33.00% was achieved for the MIN operator.Improvement declined in terms of successor operator. No difference was observed in case of interconnected links and MAX operator.Finally it was concluded that the number of gate count in logic circuitry remarkably reduced when compared to the exiting technique.

Also an initiative was taken to implement all the basic MVL operators at transistor level. Basic NMOS and PMOS transistor were used to implement the circuits. Simulated results and circuit implementation of all these basic digital circuit elements of logical operators indicated that, synthesized MVL functions could be represented by these circuit elements. It was observed that the simulated results were providing near to optimal output. Hence, the logic operator circuits could be practically implemented and had many beneficial applications.

Chapter 10

Area Efficient and Low Power MAX Operator and Its Application

10.1 Introduction

Research has shown that current-mode Multi-Valued Logic (MVL) circuit implementation reduces power consumption (Temel, Morgul and Aydin 2006; Temel and Morgul 2002). If the MVL functions could be synthesized close to optimal level, it will reduce the cost of circuit implementation. As of the most recent researches done, the main aspect of MVL is to reduce bus connection and produce efficient circuit level logic component. MVL synthesis and realization is mainly observed through logic gates such as MIN and MAX.

However, MVL has proven its performance over binary operation in the last few decades. The realization of MVL functions using CMOS architectures still provides to be a challenging aspect for researchers. Though many circuits has been proposed in literature by Jain, Bolton and Abd-El-Barr (1993); Temel, Morgul and Aydin (2006); Thoidis, et al. (1998) and Da Silva, Boudinov and Carro (2006) to perform MVL operations, the low power still remains as motivation to urge researchers in searching for better alternatives. Device minimization for low power computing is a general trend followed by researchers. Yet compared to voltage mode, some of the realization is done by current-mode circuits. However, realizations using current-mode circuits are faster in speed but consume more power. In voltage-mode circuits, the information is transferred by voltage levels (Gawande and Ladhake 2008; Thoidis, et al. 1998; Da Silva, Boudinov and Carro 2006).

This chapter will focus on realizing the MAX operator in voltage mode with two different architectures. The proposed area efficient MAX operator shows its advantage in reduced delay and transistor count. Apart from that, comparison of NOR gate logic has been done using proposed MAX and standard Complementary Metal Oxide Semiconductor (CMOS) logic. The chapter is divided into few sub-sections as follows. The second part of the chapter covers the area efficient proposed MAX circuit using two transistors. Third part covers second architecture of MAX operator using three transistors. Its application as a NOR gate realization also presented in detail. Next section presents experimental results of the area efficient MAX operator. Fifth section covers the simulation results for realization of NOR logic gate. Lastly, a chapter summary is made.

10.2 Proposed Area Efficient MAX Operator

MVL levels are represented by voltage levels in terms of a base voltage in the voltage mode circuits. In the first section of this chapter, an experiment is conducted where the initial base voltage value v_{ib} is set to 1 Volt. Logic level l corresponds to an interval of the continuous quantity X.

Hence, level 0 represents the null value and level 10 is associated with $v_b\left(=10V\right)$ and so on. As the logic level l changes, V_b changes such that,

$$v_b : (l)v_{ib} \tag{10.1}$$

Also for the experimental analysis, a maximum of 11 valued 2 variable functions are investigated using a novel voltage-mode MAX circuit which is realized using only two MOS transistors. For a maximum of 11 value, logic level l=10 would represent $v_b = 10V$. A logic level l relates to an interval of the continuous quantity x.

10.2.1 MAX Circuit Analysis

Following the definition of MAX operator and its algebraic property, the proposed MAX circuit is shown in Figure 10.1. It uses one PMOS (MP1) and one NMOS (MN1) transistor. Under normal condition, the PMOS is ON when its source-to-gate voltage $\left(V_{SGP}\right)$ is greater than threshold voltage $\left(\left|V_{TP}\right|\right)$ and in similar fashion NMOS require its gate-to-source voltage $\left(V_{GSN}\right)$ greater than threshold voltage $\left(V_{TN}\right)$. The MP1 transistor source terminal is connected to output node while the drain and gate are tied together and is connected to supply rail V_A.

Similarly, for MN1 the drain terminal is connected to output node whereas the gate and source terminals are tied together to supply rail V_B. Since, the MN1 has $\left(V_{GSN} = 0V\right)$, it is OFF throughout the circuit operation. The only possibility arises by the bulk potential to create the channel for MN1 to be ON. The body terminals of the respective transistors are tied to their source terminals to avoid threshold variation effect caused by bulk-to-source (V_{SB}) effect.

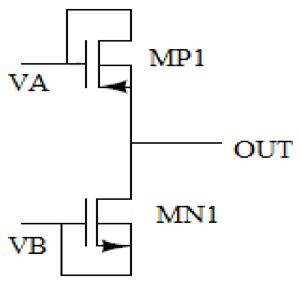


Figure 10.1: Proposed MAX circuit

Throughout the circuit operation, the output switches to the voltage according to (10.1). The expression for the output voltage is

$$V_{OUT} = \begin{cases} V_A - V_{DS,PMOS} \\ or \\ V_B + V_{DS,NMOS} \end{cases} \tag{10.2}$$

The circuit uses two power supplies V_A and V_B as input which can have different voltage levels and the output is switched to maximum supply with slight drop in voltage level according to (10.1). When V_A is at 0 potential, V_{SGP} do not get enough potential to overcome threshold voltage V_{TP} and MP1 is OFF. When V_A rises to some potential and V_B is maintained at 0 potential, the V_{SGP} turns to negative voltage which makes MP1 to remain in OFF condition. When V_A and V_B both rises to some potential, this condition still does not affect the mode of transistors. The only case when both get condition: ON and the saturation is moved in is when V_A is maintained at 0 potential whereas V_B

acquire some potential. Under this condition, V_{SGP} makes MP1 to be in saturation and through bulk potential MN1 is also tuned in saturation region. The output switches to maximum supply of circuit.

10.2.2 Experimental Results

The circuit simulation was performed on ORCAD CAPTURE CIS LITE Tool. The dimension of MN1 and MP1 were set to their minimum value $(W/L)_{1,2} = 0.24/0.24$. Figure 10.2 shows the simulation results. The output follows the maximum of input with an offset of near to 0.6V. Figure 10.3 shows the delay calculation which was about 100ps. These enhanced parameters suited best for complex computation. Furthermore, the experimental validation was done using enhancement mode PMOS (MP1) and NMOS (MN1) transistors of IC HEF4007UBP as shown in Figure 10.4.

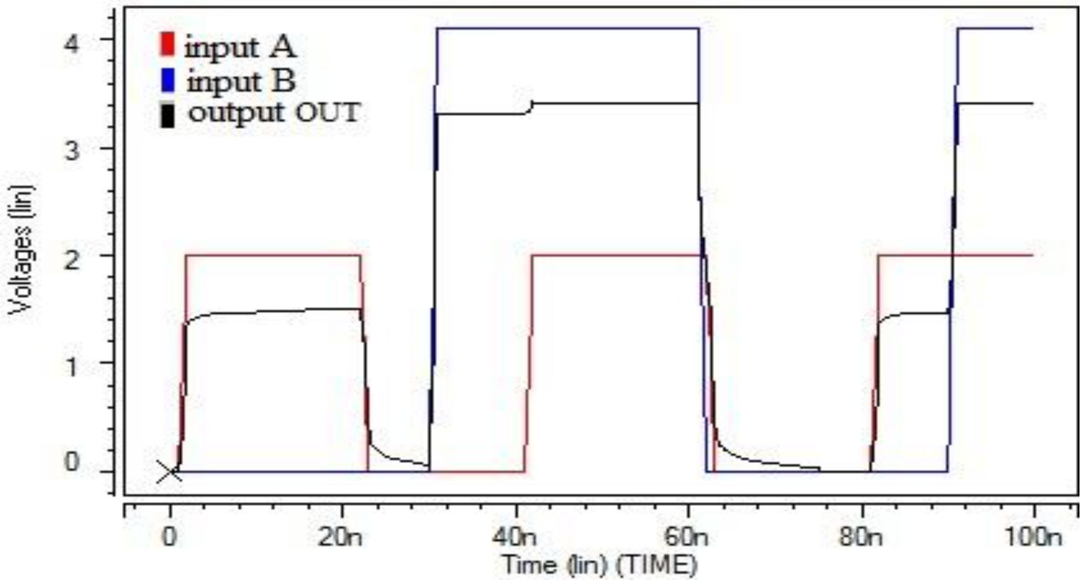


Figure 10.2: Transient simulation of proposed MAX

The PMOS source, gate, drain and bulk terminals corresponded to Pin no. 2, 3, 1 and 14 respectively. For NMOS transistor, the source, gate, drain terminals were Pin number 9, 10, 12 and 7 respectively. Pin number 7 was taken as common ground terminal.

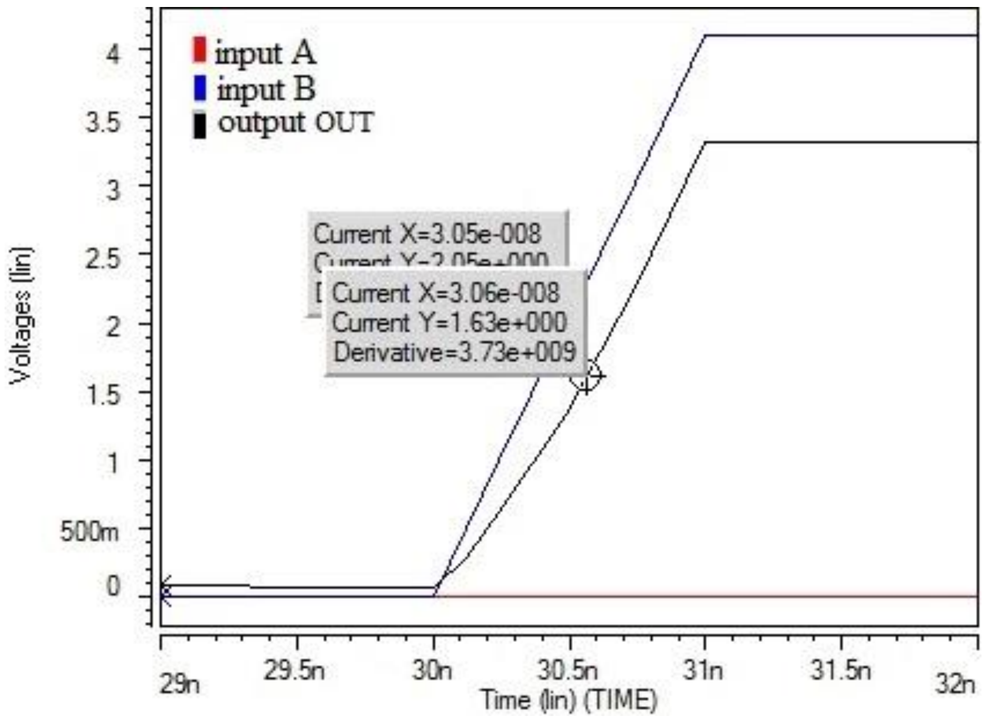


Figure10.3: Delay calculation for MAX operator

The supply input voltage for V_A and V_B ranged from 0V to 10V. Table 10-1 shows the measured voltage fluctuation in output. Figure 10.4 illustrates IC HEF4007UBP.

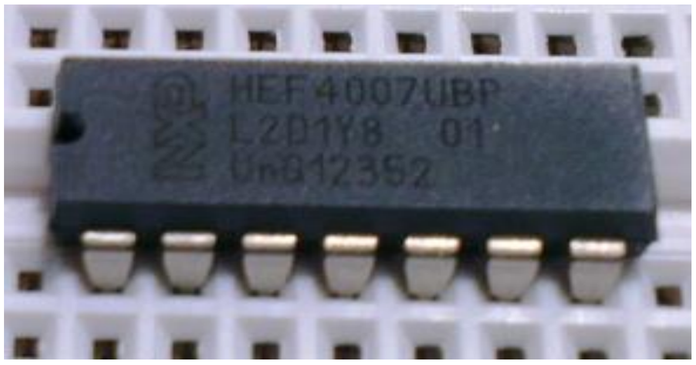


Figure 10.4: IC HEF4007UBP

Table 10-1: Output voltage under DC supply to MAX operator

Input V_A (Volt)	Input V_B (Volt)	Output (Volt)
0	10	9.330
1	9	8.950
2	8	7.530
3	7	6.460
4	6	5.370
5	5	4.780
6	4	5.360
7	3	6.560
8	2	7.270
9	1	8.680
10	0	9.890
10	10	9.950

Table 10-2 shows the comparison of proposed circuit against existing MAX circuit by Temel and Morgul (2004). This research article is one of the few detailed works that focus on logic operators in transistor level. In the article, Temel and Morgul (2004) previously proposed the MAX circuit in current mode. However, an initiative is taken to propose the MAX circuit in voltage mode which exhibits the improvement in transistor count against existing technique. The proposed work proved to be a better reduction in transistor count thus saving effective area. Due to reduced transistors, the less parasitic capacitance offered by transistors made the circuit faster.

Table 10-2: Comparison to existing MAX circuit

Ref.	Mode	Transistor count	Delay(nS)
Temel and Morgul (2004)	Current	7	1.400
Proposed work	Voltage	2	0.100

10.3 Proposed 3-Transistor MAX Operator

Considering the definition of MAX operator and its algebraic property, Figure 10.5 shows the proposed circuit. The circuit utilizes two PMOS (MP1 and MP2) and one NMOS (MN1) transistor.

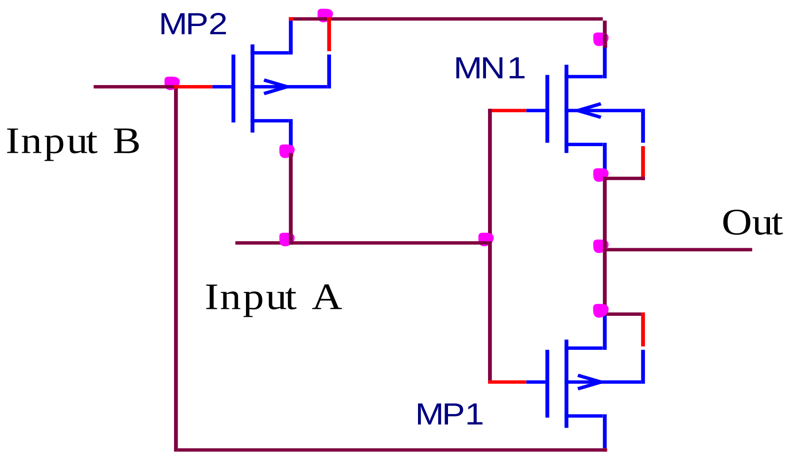


Figure 10.5: Proposed MAX circuit

Normally, the PMOS is ON when its source-to-gate voltage (V_{SGP}) is greater than threshold voltage $(|V_{TP}|)$. Same goes for the NMOS as it requires its gate-to-source voltage (V_{GSN}) greater than threshold voltage (V_{TN}). Connection between MP1 transistor source terminal and the output node is made. As for the MP2 transistor, drain is connected to its gate. The gates of MP1 and MN1 are used for signal input A. The other signal input B is connected to gate of MP2. The source terminal of MP2 and drain terminal of MN1 are tied together. The body terminals of the respective transistors are tied to their source terminals to avoid threshold variation effect caused by bulk-to-source (V_{SB}) effect.

10.3.1 Application: MVL NOR logic realization

The circuit behaves as a binary NOR logic gate. Even though the underlying concept of the architecture is based on MVL, the circuit can be implemented in binary system with reduced delay. The MVL MAX operator consists of MP1, MN1 and MN2. An additional inverter (MN2 and MP3) is added to MAX to form MAX NOR operator.

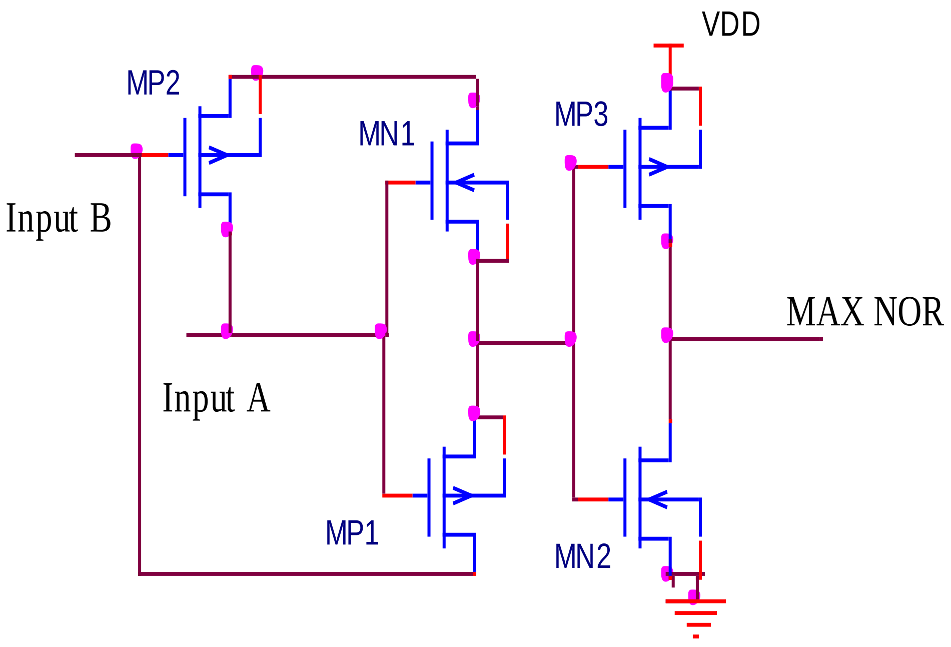


Figure 10.6: NOR gate realization using proposed MAX circuit

The circuit uses two power supplies input A and input B. The mode of operation of each of these transistors is determined by the voltage difference between these input signals. Depending on the magnitude of whichever signal input is higher, the output track the input. Further using the MAX circuit, a NOR gate realization is done as shown in Figure 10.6. It is also compared with the conventional CMOS NOR gate.

10.3.2 Simulation Results

The proposed MAX circuit simulation was performed on ORCAD CAPTURE CIS LITE Tool. The W/L ratio of MOS transistors used in all circuits was kept at its minimal value which was 0.24u/0.24u.

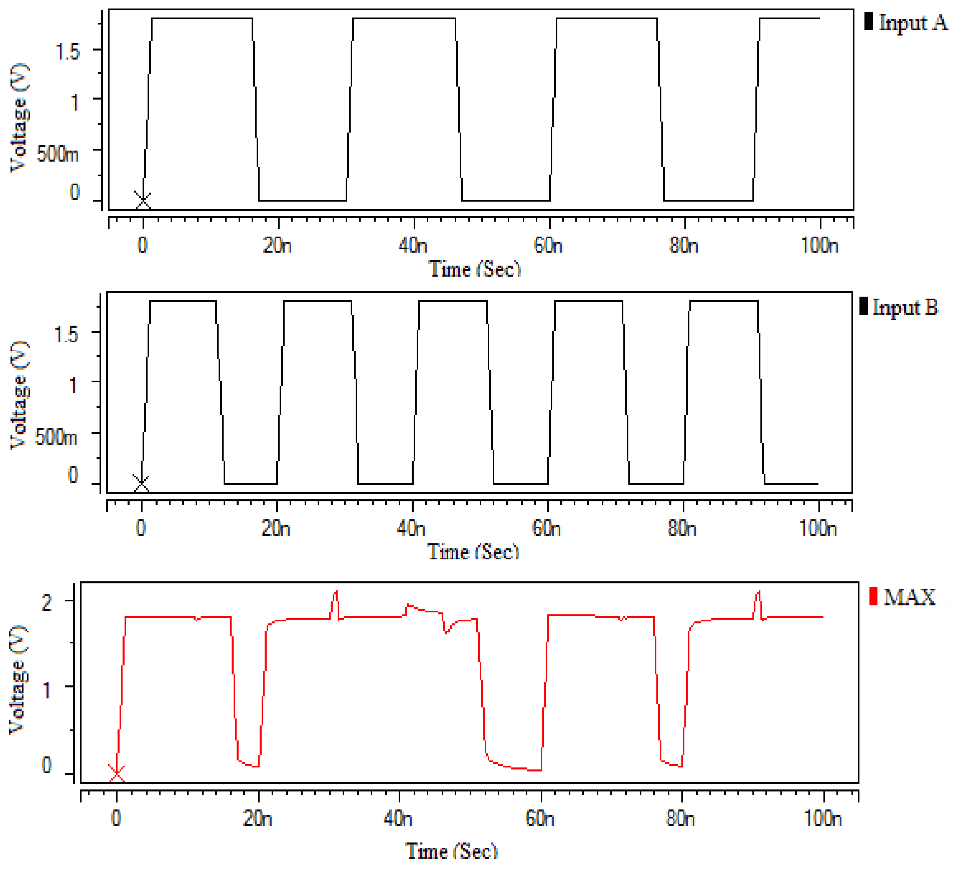


Figure 10.7: Simulation result of proposed MAX operator

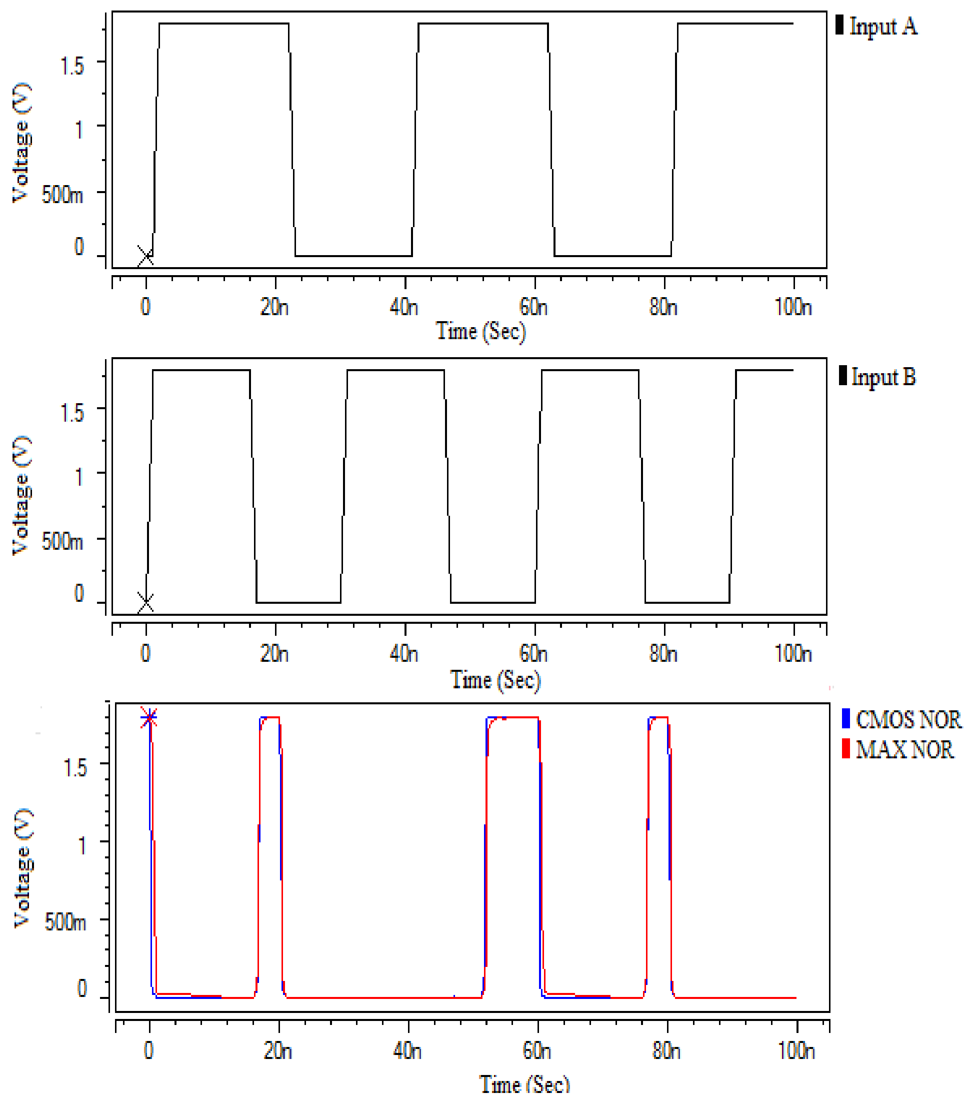


Figure 10.8: Simulation result of NOR gate using CMOS and proposed MAX operator

Figure 10.7 and Figure 10.8 respectively show the simulation results of Figure 10.5 and Figure 10.6 respectively. In Figure 10.4, the NOR gate realization done using proposed MAX operator was compared with conventional CMOS NOR gate. To evaluate the advantage of using MAX based NOR does gate, Figure 10.9 show where a single pulse was analyzed for delay response. From the plot below, it has become quite clear that the MAX based realization resulted in minimal delay and requirement of less current extraction from supply voltage. This made the circuit to reduce power consumption by 50 percent. The comparison results for delay and power is shown in Table 10-3.

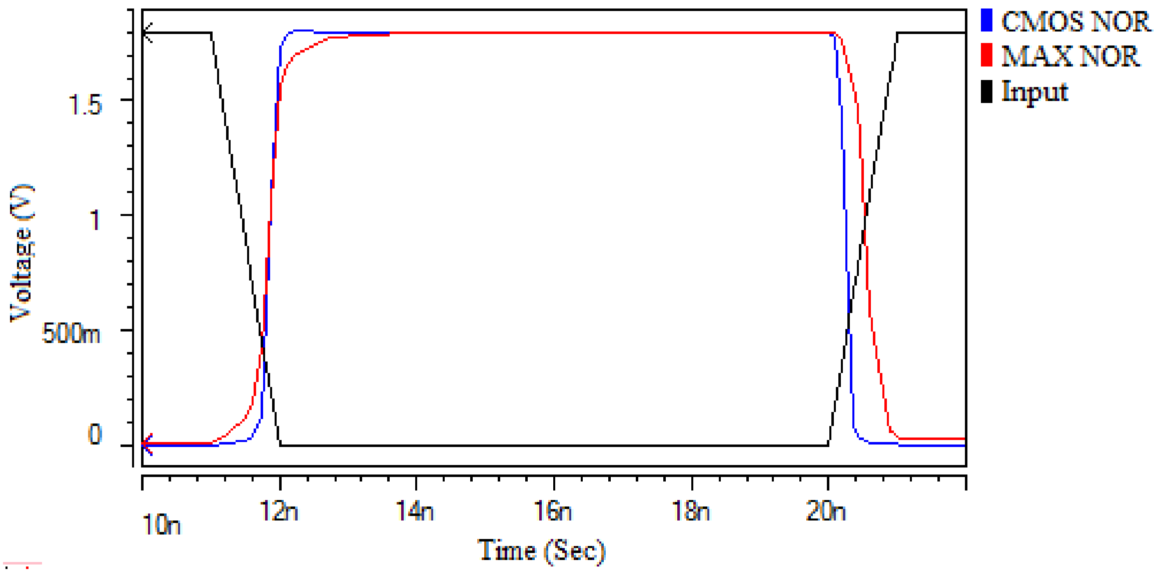


Figure 10.9: Analysis of delay for NOR gate using CMOS and proposed MAX operator

Table 10-3: Performance comparison of NOR gate

Logic	Transistor count	Rising Delay (nS)	Falling Delay (nS)	Power (uW)
CMOS	4	0.450	0.280	13.400
Proposed MAX based	5	0.350	0.004	10.000

As shown above, the proposed MAX based realization works better in terms of reducing both rising delay and falling delay. This will contribute to saving of effective area. Apart from that, the reduction of transistors which utilized parasitic capacitance is also achieved. Hence, the operation of the circuits become more efficient and less time consuming. The proposed MAX based realization also gives resulted in less requirement for current extraction from the supply voltage when compared to the use of

CMOS. It is shown that CMOS requires more power than the proposed MAX based realization. Less power used by the circuit will reduce the economic cost needed.

10.4 Summary

A novel voltage-mode MAX circuit for implementation of Multi-Valued Logic (MVL) system was presented in this chapter. The proposed circuit was based on two transistors. The proposed MAX operated for wide range of supply voltage. The novelty of circuit lay in realizing the MAX operation by only two transistors with minimal delay of 100pS. After verifying the functionality of MAX, the further experimental validation of proposed circuit was done on enhancement mode transistors offered by IC HEF4007UBP. It showed that the use of less transistor count was not only effective in terms of area saving but also increased the speed of circuit due to less presence of parasitic capacitances. Since MAX is a basic operator for synthesizing MVL function, the optimized MAX operator could do fast realization with increased efficiency.

Furthermore, a voltage-mode three transistor based MAX circuit for implementation of MVL system was proposed in this section. The proposed MAX operated at very low power consumption ranging in micro watts. To evaluate MAX performance, a NOR gate realization was done and compared to standard CMOS NOR gate. The simulation result confirmed that the MAX based NOR gate operated with minimal delay at low power level.

Chapter 11

Conclusion

This chapter gives overall conclusions to the thesis. It also discusses the findings of the thesis as well as future research directions related to the synthesis of Multi-Valued Logic (MVL).

Chapter 11 can be divided into two sections. The first section will highlight the summaries made in each chapter of this thesis. On the other hand, the second section will discuss future research directions.

11.1 Summary of Thesis

In the 21st century of technological advancement, MVL is often considered as an alternative to the binary logic system. MVL system has numerous advantages over the previous one. The MVL application ranges from connection among circuits to complicated channel coding, communication and signaling field, huge data transmission, memory design and cost effective storage system etc. To ensure better efficiency and reliability, MVL is used as a transitional subs system in binary logic.

The major fact behind the success of MVL system is the minimization of hardware cost for efficient logic design. Synthesis algorithms are built to retrieve efficient logic expressions. Logic design entirely depends upon the logic expression. Hardware cost depends upon the efficiency of the logic design. Hence, MVL synthesis algorithms are important and reflect the hardware cost for a given system. There were many major attempts to enhance the process of MVL synthesis design and its minimization.

Existing techniques such as evolutionary algorithms produces the finest result but lacks in algorithmic minimalism, computation time etc. Taken under consideration the aforementioned issues, this research involved implementing different synthesis approaches. Nevertheless of the capability of MVL system, in any of its application synthesis algorithm plays an important role. Synthesis algorithms tend to minimize the hardware representation before physical implementation. In this thesis, Chapter 1 gave an overview of the MVL system, its past, present and recent research activities, benefits and others. Chapter 2 briefly outlined the basic MVL operators and their algebraic representation.

In chapter 3, a comprehensive literature review on different MVL synthesis techniques was presented. Several comparisons were also presented to discuss different synthesis techniques. Research gap between latest research works were investigated. This managed to provide the opportunity to realize the fact that there was no universal set of MVL operators and postulates.

Chapter 4 gave an overview of a set of novel algebraic postulates and logical operators. These operators and postulates were proposed to realize MVL functions and their synthesis. Proofs for each postulate were provided in each definition. Examples were presented to elaborate the efficiency of using these operators and postulates.

In chapter 5, a novel synthesis algorithm entitled High Deduction Algorithm (HDA-MVL) was presented. A total of 50000 sequentially generated 4-valued 2-variable MVL functions were generated for experimental analysis. Proposed MVL operators were used in logic function synthesis. Experiments revealed that a reduction of 18.00% was achieved in reducing the gate count. It was observed that overall average Product Term (PT) reduction achieved by HDA-MVL algorithm is 56.88%.A remarkable 33.00% reduction was observed in reducing the usage of MIN operator for synthesizing MVL functions. In the worst case scenario proposed HDA-MVL algorithm achieved 45.46% more success in reducing PT over evolutionary ACO-MVL algorithm.

Furthermore in chapter 6, Neural Network Deployment Algorithm (NNDA-MVL) had been proposed. Main algorithmic steps were presented. NNDA-MVL is combined with back-propagation learning capability. NNDA-MVL utilized the proposed MVL operators to train. The results achieved showed that the NNDA-MVL algorithm managed to achieve an accuracy of 99.97% for EXTENDED AND neural operator. Experiments also depicted the fastest 0.008 and slowest 0.038 seconds of input to output delay for trained operators. 5000 randomly generated sample benchmarks were generated for the training purpose.

In chapter 7, the entire pseudo code was provided for the NNDA-MVL algorithm. The advantages of NNDA-MVL algorithm was demonstrated with realization of synthesized MVL function with lesser MVL operators. Overall the algorithm showed an improvement of 52.94% for MVL MIN gate reduction and reduced MVL neural net internal link connections of 23.38% compared to existing techniques.

A tree-type decision diagram based approach was investigated in chapter 8. In this chapter, an alternative approach synthesizing RMVLNs using NZMDD was proposed. A total of 49998 randomly generated 4-valued 2-variable benchmarks were generated for experimental purpose. Apart from that, another set of approximately 20K benchmarks were generated. It was observed that reduced average PT was achieved in MVL synthesis using NZMDD.During Synthesis NZMDD reduced the number of PT by 52.62% while ACO-MVL reduced by 40.13%.

In chapter 9, all logic operators were realized with transistors. ORCAD CAPTURE PSpice software was used to design architecture for each operators. Simulation results had shown logic level realization. Finally, a comparison is shown with existing algebraic synthesis to show the efficiency of the process. Realization revealed better result compared to existing algebraic synthesis techniques. In this section of the thesis, as a continuation of the logic operator, realization MV logic gates were constructed by building circuits with NMOS, PMOS transistors, resistors, diodes and comparators. The basic electronic elements that were used to build each of the circuit to represent the logical operator had differed from operator to operator.

Further experiment was conducted on a VM2T and VM3T based MAX circuit in chapter 10. A novel voltage mode MAX operator was designed and physical implemented. Digital implementation of the proposed operator required only 28.57% transistors compared to existing architecture. The validation of the MAX circuit was done on enhancement mode transistors offered by IC HEF4007UBP. Proposed voltage mode MAX operator performs better in terms of delay reduction. On the contrary, the simulations results confirmed that proposed VM3T MAX had a lesser rising edge delay of 22.22% compared to existing CMOS NOR. It was also observed that proposed operator consumes 25.37% less power than the existing operator. In this chapter, it was presented that optimized MAX operator could do fast realization with increased efficiency. An effective use of this operator was shown by realizing NOR gate and its comparison with standard CMOS NOR gate.

11.2 Future Research Directions

Day by day MVL is expanding towards becoming a next generation technology. Yet, many more different approaches can be investigated to produce better and efficient synthesis algorithms. Area efficient MAX operator can be further investigated to optimize power efficiency. ODD and EVEN logic operators can be further utilized in applications such as, MVL wireless signal processing system. Architecture 2: low power MAX operator can be tested using latest HSpice technology to verify its significance and usage in the electronic industry. Switching effect of the proposed architecture can be further investigated to optimize its rising and falling delay.

In Information Technology (IT) industry, corporations such as, IBM and Intel have developed memory and processor using MVL technology. Yet this efficient technology is considered costly to reach the door step of every common people. Research initiative can be focused to reduce the production cost for individual MVL operators and devices. Also, the proposed novel MVL operators can be thoroughly scrutinized to find its implication in the computing industry. As a practical implementation, MVL circuitry can be incorporated as an intermediate subsystem to binary counterpart. Most evolutionary algorithms derived from real life activities. Approaches such as, shortest path detection, Food source vs. Ant attraction, Tree growth vs. Sunlight direction, Jelly fish neural architecture are among the many examples of future research directions that can be introduced to obtain simple and efficient synthesis techniques.

Bibliography

Abd-El-Barr, M. "Evolutionary Techniques in Synthesis of Multiple-Valued Logic Functions." *International Journal of New Computer Architectures and Their Applications* 2 , no. 3 (2012): 410-421.

—. "Hybrid Fuzzy Direct Cover Algorithm for Synthesis of Multiple-Valued Logic Functions." *International Journal of Computer Science Issues* 8, no. 2 (2011): 158-166.

Abd-El-Barr, M., and A., Khan. Esam. "Improved Direct Cover Heuristic Algorithms for Synthesis of Multiple-Valued Logic Functions." *International Journal of Electronics* (Taylor & Francis) 101, no. 2 (2013): 1-16. doi:10.1080/00207217.2013.780296

Abd-El-Barr, M., and B.A.B. Sarif. "Synthesis of MVL Functions - Part II: The Ant Colony Optimization Approach." *International Conference on Microelectronics.* IEEE Conference Publications, 2006. 158-161. doi: 10.1109/ICM.2006.373291

—. "Weighted and Ordered Direct Cover Algorithms for Minimization of MVL Functions." *The 37th International Symposium on Multi-Valued Logic*. IEEE Conference Publications, 2007. 48. doi: 10.1109/ISMVL.2007.60

Abd-El-Barr, M., and L. Al-Awami. "Analysis of Direct Cover Algorithms for Minimization of MVL Functions." *Proceedings of the 15th International Conference on Microelectronics.* IEEE Conference Publications, 2003. 308-312. doi: 10.1109/ ICM.2003.1287819

Abd-El-Barr, M., and M. Al-Mutawa. "A New Improved Cost-Table-Based Technique for Synthesis of 4-Valued Unary Functions Implemented using Current-Mode CMOS Circuits." *Proceedings of the 31st IEEE International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 2001. 15-20. doi: 10.1109/ISMVL.2001.924549

Abd-El-Barr, M., Z.G. Vranesic, and S.G. Zaky. "Algorithmic synthesis of MVL Functions for CCD implementation." *IEEE Transactions on Computer* 40, no. 8 (1991): 977-986. doi: 10.1109/12.83641

Allen, C.M., and Donald D. Givone. "A Minimization Technique for Multiple-Valued Logic Systems." *IEEE Transactions on Computers* (IEEE Jounals and Magazines) C17, no. 2 (1968): 182-184. doi: 10.1109/TC.1968.227407

Al-Rabadi, A.N. "Iterative Symmetry Indices Decomposition for Ternary Logic Synthesis in Three-Dimensional Space." *Proceedings of 33rd International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 2003. 139-145. doi: 10.1109/ISMVL.2003.1201398

Apostolikas, G., and S. Konstantopoulos. "Error Back-Propagation in Multi-valued Logic Systems." *International Conference on Computational Intelligence and Multimedia Applications.* IEEE Conference Publications, 2007. 207-213. doi: 10.1109/ICCIMA.2007.362

Balasubramanian, P., M.R.L. Narayana, and R. Chinnadurai. "Design of Combinational Logic Digital Circuits using a Mixed Logic Synthesis Method." *Proceedings of the IEEE Symposium on Emerging Technologies.* IEEE Conference Publications, 2005. 289-294. doi: 10.1109/ICET.2005.1558896

Besslich, P.W. "Heuristic Minimization of MVL functions: A Direct Cover Approach." *IEEE Transactions on Computer* (IEEE Journals and Magazines) C-35, no. 2 (1986): 134-144. doi: 10.1109/TC.1986.1676731

Brayton, R.K., et al. "VIS: A System for Verification and Synthesis." *Proceedings of the 8th International Conference on Computer Aided Verification.* Springer Lecture Notes in Computer Science, 1996. 428-432.

Chang, Yeong-Jar, and Chung Len Lee. "Synthesis of multi-variable MVL functions using hybrid mode CMOS logic." *Proceedings of the 24th International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 1994. 35-41. doi: 10.1109/

ISMVL.1994.302222

Chojnacki, A., and L. Jozwiak. "Multi-Valued Sub-Function Encoding in Functional Decomposition Based on Information Relationships Measures." *Proceedings of 30th*

IEEE International Symposium on Multiple-Valued Logic. IEEE Conference Publications, 2000. 83-90. doi: 10.1109/ISMVL.2000.848604

Chowdhury, A.K, and A.K. Singh. "An Analysis of Novel MVL Neural Operators using Feed Forward Back-Propagation, Realization and Applications of Logic Synthesis." *Manuscript submitted for publication.* 2014.

Chowdhury, A.K, N. Raj, and A.K. Singh. "A Novel High Deduction Algorithm for Synthesizing Multiple-Valued Logic and Circuit." *Manuscript submitted for publication, 2013.*

Clarke, Michael, and Steve Reeves. *Logic for Computer Science.* London: Addison-Wesley Publishers Ltd, 1990.

Da Silva, R.C.G., H. Boudinov, and L. Carro. "A Novel Voltage-Mode CMOS Quaternary Logic Design." *IEEE Transactions on Electron Devices* (IEEE Journals and Magazines) 53, no. 6 (2006): 1480-1483. doi: 10.1109/TED.2006.874751

Davis, Martin. "Hilbert's Tenth Problem is Unsolvable." *The American Mathematical Monthly* 80, no. 3 (1973): 233-269.

Demuth, H., M. Beale, and M. Heagen. *Neural Network Toolbox 6: User's Guide.* 2010. http://www.manualslib.com/manual/392845/Matlab-Neural-Network-Toolbox-6.html?page=3#manual (accessed September 23, 2013).

Drechsler, R., D. Jankovic, and R.S. Stankovic. "Generic Implementation of DD Packages in MVL." *Proceedings of 25th EUROMICRO Conference.* IEEE Conference Publications, 1999. 352-359. doi: 10.1109/EURMIC.1999.794491

Drechsler, R., M. Thornton, and D. Wessels. "MDD-Based Synthesis of Multi-Valued Logic Networks." *The 30th IEEE International Symposium on Multi-Valued Logic.* IEEE Conference Publications, 2000. 41-46. doi: 10.1109/ISMVL.2000.848598

Dubrova, Elena. "Multiple-Valued Logic in VLSI: Challenges and Opportunities." *Proceedings of NORCHIP*. 1999. 340-350.

Dueck, G., and D.M. Miller. "A Direct Cover MVL Minimization Using the Truncated Sum." *Proceeding of The 17th International Symposium on Multi-Valued Logic*. IEEE Conference and Publications, 1987. 221-227.

Dunderdale, H. "Current-mode circuits for ternary-logic realisation." *Electronic Letters* 5, no. 23 (1969): 575-577. doi: 10.1049/el:19690433

Epstein, George, Gideon Frieder, and David C. Rine. "The Development of Multiple-Valued Logic as Related to Computer Science." *Computer* 7, no. 9 (1974): 20-32. doi: 10.1109/MC.1974.6323304

Etiemble, D. "Multivalued Integrated Circuits for Signal Transmission." *Proceedings of COMPCON*. 1981, 1981. 205-208.

Feinstein, D.Y., and M.A. Thorntorn. "On the Guidance of Reversible Logic Synthesis by Dynamic Variable Reordering." *Proceedings of 39th International Symposium on Multiple-Valued Logic*. IEEE Conference Publications, 2009. 132-138. doi: 10.1109/ISMVL.2009.31

Files, C., R. Drechsler, and M.A Perkowski. "Functional Decomposition of MVL Functions using Multi-Valued Decision Diagrams." *Proceedings of the 27th International Symposium on Multiple-Valued Logic*. IEEE Conference Publications, 1997. 27-32. doi: 10.1109/ISMVL.1997.601370

Frege, Gottlab. "A Formula Language,Modeled upon that of Arithmethic for Pure Thought." In *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, by Jean Van Heijenoort, 10-76. Harvard University Press, 1977.

Gawande, A.D., and S. A. Ladhake. "Constraints In the Design of CMOS MVL Circuits." *Proceedings of the 7th WSEAS International Conference on Instrumentation, Measurement, Circuits and Systems*. 2008. 108-113.

Gibson, James R. *Electronic Logic Circuits*. New York: Taylor and Francis, 2013.

Gottwald, Siegfried. "Many-Valued Logics." In *Philosophy of Logic*, edited by Dov M. Gabbay, Paul Thagard, John Woods and Dale Jacquette, 675-722. Oxford: North-Holland, 2007.

Higuchi, T., and M. Kameyama. "Ternary Logic System based on T-gate." *Proceedings of the 5th International Symposium on Multi-Valued Logic.* IEEE Conference Publications, 1975. 290-304.

Hsu, Loke-Soo, H.-H. Teh, S.-C. Chan, and Kia-Fock Loe. "Multi-Valued Neural Logic Networks." *Proceedings of the 12th International Symposium on Multiple-Valued Logic.* IEEE Conference Publications , 1990. 426-432. doi: 10.1109/ISMVL.1990.122658

Hu, Ye-Fa, Shao-Ping Ku, Zu-De Zhao, and Yi-Xin Su. "Multi-Valued Logic and its Application in the Fault Diagnosis of the Sensors of Magnetic Bearings." *Proceedings of 2004 International Conference on Machine Learning and Cybernetics.* IEEE Conference Publications, 2004. 2458-2462. doi: 10.1109/ICMLC.2004.1382216

Jain, A. K., R. J. Bolton, and M. Abd-El-Barr. "CMOS Multiple-Valued Logic Design. I. Circuit Implementation." *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* (IEEE Journals and Magazines) 40, no. 8 (1993): 503-514. doi: 10.1109/81.242320

Jain, A. K., R. J. Bolton, and M. Abd-El-Barr. "CMOS Multiple-Valued Logic Design. II. Function realization." *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* (IEEE Journals and Magazines) 40, no. 8 (1993): 515-522. doi: 10.1109/81.242321

Jiang, Yunjian, S. Matic, and R.K. Brayton. "Generalized Cofactoring for Logic Function Evaluation." *Proceedings of Design Automation Conference.* IEEE Conference and Publications, 2003. 155-158. doi: 10.1109/DAC.2003.1218924

Kalganova, T., T.J Miller, and N. Lipinitskaya. "Multiple-Valued Combinational Circuits Synthesized using Evolvable Hardware Approach." *Proceedings of the 7th*

Workshop on Post-Binary Ultra Large Scale Integration Systems in Association with ISMVL'98. Fukuoka: IEEE Conference Publications, 1998.

Kalganova, T., T.J Miller, and T. Fogarty. "Some Aspects of an Evolvable Hardware Approach for Multiple-Valued Combinational Circuit Design." *Proceedings of Second International Conference on Evolvable System: From Biology to Hardware.* Springer Berlin Hidelberg, 1998. 78-89.

Kohavi, R., G. John, R. Long, D. Manley, and K. Pfleger. "MLC++: A Machine Learning Library in C++." *Proceedings of the 6th International Conference on Tools with Artificial Intelligence.* IEEE Conference Publications, 1994. 740-743. doi: 10.1109/TAI.1994.346412

Kostolny, J., Kvassay, M., & Zaitseva, E. (2014). Analysis of Algorithms for Computation of Direct Partial Logic Derivatives in Multiple-Valued Decision Diagrams. *9th International Conference on Availability, Reliability and Security (ARES)*, (pp. 356-361). Fribourg. doi:10.1109/ARES.2014.54

Lablan, P. *Multi-Valued Logic.* 2005-2011. http://archive.is/rIyG (accessed July 26, 2013).

Marinescu, Dan C., and Gabriela M. Marinescu. *Approaching Quantum Computing.* Prentice Hall, 2005.

Mathew, J., H. Rahaman, A.K. Singh, A.M. Jabir, and D.K. Pradhan. "A Galois Field Based Logic Synthesis Approach with Testability." *21st International Conference on VLSI Design.* Hyderabad: IEEE Conference Publications, 2008. 629-634. doi: 10.1109/VLSI.2008.88

Matsumoto, M., Y. Ueda, and I. Nomoto. "The synthesis of multiple-valued logic circuits using local-excitation-type." *Proceedings of the 30th IEEE International Symposium on ISMVL.* Oregon: IEEE, 2000. 21-26. doi: 10.1109/ISMVL.2000.848595

"Max, Min, Average Circuits." *Bison Academy.* December 2010. http://www.bisonacademy.com/ECE321/Lectures/10%20Max%20Min%20Average.pdf (accessed August 1, 2013).

Miller, D.M., and R. Drechsler. "On the Construction of Multiple-Valued Decision Diagrams." *Proceedings 32nd IEEE International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 2002. 245-253. doi: 10.1109/ISMVL.2002.1011095

Miller, Michael D., and Mitchell A. Thorntorn. *Multiple Valued Logic: Concepts and Representations.* Morgan and Claypool Publishers, 2008.

Mo, Y., Hing, L., & Amari, S. (2014). A Multiple-Valued Decision Diagram Based Method for Efficient Reliability Analysis of Non-Repairable Phased-Mission Systems. *IEEE Transcation on Reliability, 63*(1), 320-330. doi:10.1109/TR.2014.2299497

Nakahara, H., T. Sasao, and M. Matsuura. "A Comparison of Heterogeneous Multi-valued Decision Diagram Machines for Multiple-Output Logic Functions." *The 41st IEEE International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 2011. 125-130. doi: 10.1109/ISMVL.2011.15

Ngom, A., and D.A. Simovici. "Evolutionary strategy for learning multiple-valued logic functions." *Proceedings of the 34th International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 2004. 154-160. doi: 10.1109/ISMVL.2004.1319935

Parkes, Alan. *Introduction to Languages, Machines and Logic: Computable Languages, Abstract Machines and Formal Logic.* London: Springer-Verlag, 2002.

PHILIPS. "HEF4007UB Gates: Dual Complementary Pair and Inverter." *NXP Semiconductors.* January 1995. http://www.nxp.com/documents/data_sheet/HEF4007UB_CNV.pdf (accessed November 1, 2013).

Pomper, G., and J.A. Armstrong. "Representation of Multiple Valued Functions using the Direct Cover Method." *IEEE Transcations on Computers* (IEEE Journals and Magazines), 1981: 674-679.

Posa, J.G. "Four State Cell Doubles ROM Bit Capacity." *Electron*, 1980: 39.

Post, Emel L. "Introduction to a General Theory of Elementary Propositions." *American Journal of Mathematics* 43, no. 3 (1921): 163-185.

Rafiev, A., Murphy, J., & Yakovlev, A. (2010). Secure Design Flow for Asynchronous Multi-Valued Logic Circuits. *40th IEEE International Symposium on Multi-Valued Logic*, (pp. 264-269). Barcelona. doi:10.1109/ISMVL.2010.56

Romero, M. E. R., E. M. Martins, and R. R. Santos. "Multiple Valued Logic Algebra for the Synthesis of Digital Circuits." *The 39th International Symposium on Multiple-Valued Logic*. IEEE Conference Publications, 2009. 262-267. doi: 10.1109/ISMVL.2009.45

Romero, M.E, E. Mazina Martins, R. Ribiera dos Santos, and M.E.D. Gonzales. "Universal Set of CMOS Gates for the Synthesis of Multiple Valued Logic Digital Circuits." *IEEE Transactiosn on Circuits and Systems I: Regular Papers* (IEEE Journals and Magazines) 61, no. 3 (2014): 736-749. doi: 10.1109/TCSI.2013.2284187

Ross, C.W. "Reducing System Interconnections with Multivalued Logic." *Electron*, 1977: 122-124.

Sagar, P., Shiny, G., & Baiju, M. (2013). Space Vector based Pulse Width Modulation Scheme for Multilevel Inverters Using the Concept of Multi-Valued Logic. *10th IEEE International Conference on Power Electronics and Drive Systems (PEDS)*, (pp. 1360-1365). Kitakyushu. doi:10.1109/PEDS.2013.6527231

Sarif, B.A.B, and M. Abd-El-Barr. "Functional Synthesis using Discrete Particle Swarm Optimization." *Swarm Intelligence Symposium.* IEEE Conference Publications, 2008. 1-8. doi: 10.1109/SIS.2008.4668306

—. "Minterm Injection Technique for Synthesis of Multiple- Valued Logic Functions." *Proceedings of IASTED International Conference on Circuits and Systems.* ACTA Press, 2008. 61-65.

—. "Synthesis of MVL Functions - Part I: The Genetic Algorithm Approach." *International Conference on Microelectronics.* IEEE Conference Publications, 2006. 154-157. doi: 10.1109/ICM.2006.373290

—. "The Use of Multiple Connected Pseudo Minterms in the Synthesis of MVL Functions." *39th International Symposium on Multi-Valued Logic.* IEEE Conference Publications, 2009. 145-150. doi: 10.1109/ISMVL.2009.56

Sasao, T. "Multiple-Valued Input Index Generation Functions: Optimization by Linear Transformation." *42nd IEEE International Symposium on Multiple-Valued Logic (ISMVL).* Victoria: IEEE, 2012. 185-190. doi: 10.1109/ISMVL.2009.35

Sasao, T. "Multiple-Valued Logic and Optimization of Programmable Logic Arrays." *Computer* 21, no. 4 (1988): 71-80. doi: 10.1109/ISMVL.2012.21

—. "On the Numbers of Variables to Represent Multi-Valued Incompletely Specified Functions." *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD).* IEEE Conference Publications, 2010. 420-423. doi: 10.1109/2.52

Sasao, T., H. Nakahara, M. Matsuura, Y. Kawamura, and J.T Butler. "A Quaternary Decision Diagram Machine and the Optimization of its Code." *39th International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 2009. 362-369. doi: 10.1109/DSD.2010.9

Smith, K.C. "A Multiple Valued Logic: A Tutorial and Appreciation." *IEEE Transactions on Computer* (IEEE Journals and Magazines) 21, no. 4 (1988): 17-27. doi: 10.1109/2.48

Smith, K.C. "The Prospects for Multivalued Logic: A Technology and Applications View." *IEEE Transactions on Computer* (IEEE Journals and Magazines) C-30, no. 9 (1981): 619-634. doi: 10.1109/TC.1981.1675860

Song, S.-G., Park, S.-M., & Oh, M.-H. (2014). A New Data Encoding Scheme using Multi-Valued Logic for an Asynchronous Handshake Protocol. *18th IEEE International Symposium on Consumer Electronics (ISCE 2014),* (pp. 1-2). Jeju Island. doi:10.1109/ISCE.2014.6884392

Stamate, D. "Assumption based multi-valued semantics for extended logic programs." *36th International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 2006. 10. doi: 10.1109/ISMVL.2006.13

Stark, M. "Two bits per cell ROM." *Proceedings of COMPCON*. 1981. 209-216.

Temel, T, and A. Morgul. "Implementation of Multi-Valued Logic Gates using Full Current Mode CMOS Circuit." *Analog Integrated Circuit and Signal Processing*, 2004: 191-204.

Temel, T., A. Morgul, and N. Aydin. "Signed Higher-Radix Full-Adder Algorithm and Implementation with Current-Mode Multi-Valued Logic Circuits." *IEE Proceedings Circuits, Devices and Systems.* 153, no. 5 (2006): 489-496. doi: 10.1049/ip-cds:20045096

Temel, T., and A. Morgul. "Implementation of Multi-Valued Logic Gates using Full Current-Mode CMOS Circuits." *International Symposium on Circuits and Systems.* IEEE Conference Publications, 2002. I-881- I-884. doi: 10.1109/ISCAS.2002.1009982

Terman, L.M, Y.S Yee, Merril R.B, L.G Heller, and M.B. Pettigrew. "CCD Memory using Multilevel Storage." *IEEE Journal of Solid-State Circuits* (IEEE Journals and Magazines) 16, no. 5 (1981): 472-478. doi: 10.1109/JSSC.1981.1051625

Thoidis, I., D. Soudris, I. Karafyllidis, S. Christoforidis, and A. Thanailakis. "Quaternary voltage-mode CMOS circuits for multiple-valued logic." *IEE Proceedings* *of Circuits, Devices and Systems.* IET Journals & Magazines, 1998. 71-77. doi: 10.1049/ip-cds:19981763

Tirumalai, P., and J.T. Butler. "Analysis of Minimization Algorithms for Multiple-Valued Programmable Logic Arrays." *Proceedings of the Eighteenth International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 1988. 226-236. doi: 10.1109/ISMVL.1988.5178

Wang, Min Hui, Chung Len Lee, and Chen Jwu E. "Algebraic Division for Multilevel Logic Synthesis of Multi-Valued Logic Circuits." *Proceedings of the 24th International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 1994. 44-51. doi:10.1109/ISMVL.1994.302221

X. Xiang. "PSpice Student 9.1 Tutorial." *Digital-Analog Hybrid Circuit Coursework.* May 2010. http://wenku.baidu.com/view/bf6b4637ee06eff9aef807a8 (accessed July 3, 2013).

Yamada, M., K. Fujishima, and Gamou, Y. Nagasawa.Koichi. "A New Multilevel Storage Structure for High Density CCD Memory." *IEEE Journal of Solid-State Circuits* (IEEE Journals and Magazines) 13, no. 5 (1978): 688-693. doi: 10.1109/JSSC.1978.1051120

Yang, C., and Y.M. Wang. "A Neighborhood Decoupling Algorithm for Truncated Sum Minimization." *Proceedings of the 12th International Symposium on Multiple-Valued Logic.* IEEE Conference Publications, 1990. 153-160. doi: 10.1109/ISMVL.1990.122611

Yildirim, C., J.T Butler, and C. Yang. "Multiple-valued PLA Minimization by Concurrent Multiple and Mixed Simulated Annealing." *Proceedings of The 23rd International Symposium on Multi-Valued Logic.* IEEE Conference Publications, 1993. 17-23. doi: 10.1109/ISMVL.1993.289587

Yuan, K.-L., Kuo, C.-Y., Jiang, J.-H. R., & Li, M.-Y. (2013). Encoding Multi-Valued Functions for Symmetry. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD),* (pp. 771-778). San Jose. doi:10.1109/ICCAD.2013.6691201

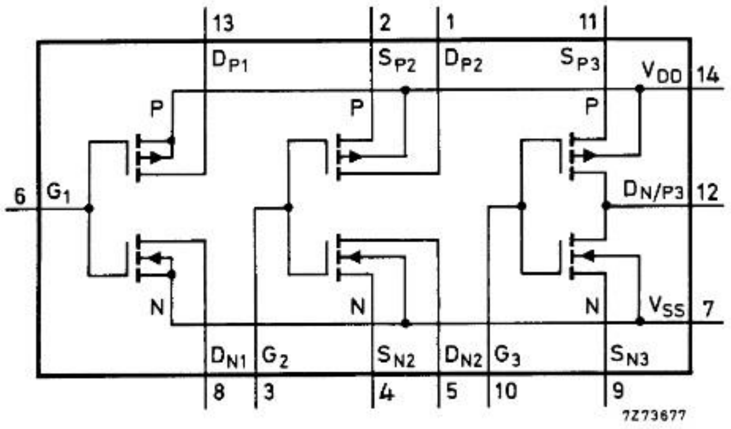
Zheng, T., O. Ishizuka, and K. Tanno. "A Learning Multiple-Valued Logic Networks based on Backpropagation." *Proceedings of the 25th International Symposium on Multi-Valued Logic*. IEEE Conferenec Publications, 1995. 270-275. doi: 10.1109/ISMVL.1995.513542

Zheng, T., Qi-Ping Cao, and O. Ishizuka. "A Learning Multiple-Valued Logic Network: Algebra, Algorithm, and Applications." *IEEE Transactions on Computers* (IEEE Journals and Magazines) 47, no. 2 (1998): 247-251.

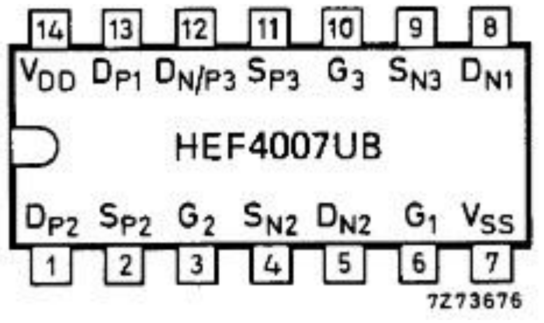
Appendices

APPENDIX A – SCHEMATIC AND PIN CONNECTION OF IC HEF4007UB

The HEF4007UB is a dual complementary pair and an inverter with access to each device. It has three N-Channel and three P-Channel enhancement mode MOS transistors. In the diagram below it can be observed that SP2 and SP3source connections to 2ndand 3rd P-Channel transistors. DP1 and DP2drain connections from the 1stand 2nd P-Channel transistors.DN1 and DN2drains connections from the 1stand 2nd N-Channel transistors. On the other hand, SN2 and SN3source connections to the 2ndand 3rd N-Channel transistors. DN/P3 common connection to the 3rd P-Channel and N-Channel transistor drains. G1to G3 gate connections to N-Channel and P-Channel of the three transistor pairs. The schematic diagram of the IC is presented as below.



The pin diagram of the IC is shown as below.



All data taken from (PHILIPS 1995)

APPENDIX B – CHAPTER 6 TRAINING AND TEST RESULTS

A. The table below generates the output of the neural EXTENDED AND operator. Total 5000 sample benchmarks were fed to the system for training. Only 178 outputs are shown in the table below.

Index	Input A	Input B	Input C	Input D	Input E	Output
1.	2	1	1	1	1	0
2.	3	1	3	2.94	2	0
3.	1	2	2	1	2	1
4.	3	1	1	3	2	3
5.	2	3	3	3	3	0
6.	3	2	1.32	1	1	0
7.	2	2.85	2	3	3	0
8.	1	1	1	1	3	1
9.	2	3	3.59	3	3	0
10.	3	1	1	3	2	3
11.	2	2	2	2	1	2
12.	2	3	3	2	3.41	2
13.	2	1	1	2	1.32	2
14.	3	1	1	3	1	3
15.	1	3	3	1	0	1
16.	0	1.33	2	0	0	0
17.	3	3	3	3	1	3
18.	0	2	0	3	3	0
19.	2	3	0	1	3.41	0
20.	2	1	1	2	1	2
21.	1.33	2	1	1	2	0
22.	2.6	2	3	3	3	0
23.	3	1	2	1	3.41	0
24.	3.52	2.97	2	3	1	0
25.	3	1	1	3	3	3
26.	1	3	3	1	1	1
27.	1.35	2	1	1	1	0
28.	2	2	2.99	1	2.75	0
29.	1	3	0	0	3.51	0
30.	0	3	2	3	0	0

31.	2	2	2	2	2	2
32.	1	1	1.32	2	0	0
33.	2	1	1	2	2	2
34.	3	1	1	3	2	3
35.	2.52	3.44	3	1	2	0
36.	2	3	3	2	0	2
37.	3	1	1	3	3.42	3
38.	3.58	0	1	3	0	0
39.	1	1	1	1	2	1
40.	3	1	1	3	2	3
41.	2	3	1	2	1	0
42.	2	1	1	2	3	2
43.	1	1	1	1	3	1
44.	1	2	3.53	2	3	0
45.	3.49	3	2	1	2	0
46.	2	3	3	2	1	2
47.	3	1	1	3	2	3
48.	3	3	3	3	1	3
49.	3.37	3	3	1	0	0
50.	3	0	2.86	2	3	0
51.	1	1	1	1	0	1
52.	3	3	3	3	3	3
53.	3	2	2	3	3.56	3
54.	1	1	1	1	3	1
55.	2	2	2	2	3	2
56.	3	2	2	3	1	3
57.	2	3	3.45	0	3.6	0
58.	3	1	2	2	2.92	0
59.	1	2	2	1	3	1
60.	1	1	1	2	0	0
61.	3	3	3	3	3	3
62.	2	2	2	2	3	2
63.	2	2	2	3	2	0
64.	1	2.68	3.42	0	3	0
65.	3	3.55	0	1	2	0
66.	2	2	2	2	3	2
67.	0	2	1.32	3	1	0
68.	1	2	2	1	1	1
69.	1	1	1	1	1.34	1
70.	1	2	2	1	1.31	1
71.	1	3	3	1	0	1

72.	1.32	0	0	3	1.3	0
73.	2	3	3	2	1	2
74.	2	3	3	2	1	2
75.	3	3	3	3	0	3
76.	3	2	2	3	1	3
77.	2	1	1	2	2.55	2
78.	1	2	2	1	0	1
79.	1	1	1	1	0	1
80.	1	1	1	1	1.34	1
81.	2	1.3	1.3	2	2	2
82.	1	1	1	1	1	1
83.	3	3	1	1	1.35	0
84.	1	3	3	1	2	1
85.	3	3	3	3	2	3
86.	3	3	3	3	3	3
87.	2	1	1	2	1	2
88.	2	1	1	2	1	2
89.	2.7	0	2	2	2	0
90.	2	2	2	2	3	2
91.	2	2	2	2	0	2
92.	0	1	3	1	0	0
93.	3	1	1	3	3	3
94.	2	0	0	2	1	2
95.	1	3	3	1	1	1
96.	1	2	2	1	1	1
97.	1	1	1	1	1	1
98.	3	1	1	3	1.33	3
99.	1	1	1	1.32	1	0
100.	1	2	2	1	2	1
101.	2	3	3	2	1	2
102.	3	3	3	3	3	3
103.	1	1	1	1	0	1
104.	1	3	1	1	1	0
105.	2	0	0	2	1	2
106.	1	3	3	1	3	1
107.	1	2	2	1	2	1
108.	2	0	0	2	2	2
109.	2	3	3	2	2	2
110.	2	2	2	2	2	2
111.	2	3	3	2	2	2
112.	1	2	2	1	3	1

113.	3	1	1	3	0	3
114.	3	3	3	3	1	3
115.	3	3	3	3	2	3
116.	1	2	2	1	3	1
117.	3	3	3	3	2.81	3
118.	2	3	3	2	2.92	2
119.	3	3	3	3	2	3
120.	2	3	3	2	1	2
121.	1	0	0	1	2	1
122.	1	2	2	1	2	1
123.	2.76	2.77	1	1	2	0
124.	2	1	1	2	2	2
125.	2	1	1	2	2	2
126.	1	1	1	1	2	1
127.	3	3	3	3	2	3
128.	1	2	2	1	0	1
129.	1	3	3	1	3	1
130.	3	3	3	3	2	3
131.	1	1	1	1	2.55	1
132.	2	2	2	2	1	2
133.	1	3	3	1	3	1
134.	2	2	2	2	1	2
135.	1	2	2	1	2	1
136.	1	0	0	1	3	1
137.	2	2	2	2	3	2
138.	1	2	2	1	3	1
139.	1	2	2	1	1	1
140.	3	3	3	3	3	3
141.	3	3	3	3	3.54	3
142.	1	2	2	1	3	1
143.	1	2	2	1	3	1
144.	3	3	3	3	2	3
145.	2	1	1	2	1	2
146.	2	3	3	2	2	2
147.	3	2	2	3	1.29	3
148.	3	3	3	3	0	3
149.	2	2	2	2	0	2
150.	3	2	2	3	2	3
151.	1	1	1	1	1	1
152.	1	1	1	1	3	1
153.	1	1	1	1	2	1

154.	3	3	3	3	0	3
155.	1	3	3	1	3	1
156.	1	3	3	1	1	1
157.	3	3	3	3	1	3
158.	2	1	1	2	3	2
159.	3	3	3	3	1	3
160.	1	1	1	1	1	1
161.	3	3	3	3	3	3
162.	1	1	1	1	1	1
163.	1	2	2	1	1.29	1
164.	3	3	3	0	0	0
165.	3	3	3	3	3	3
166.	2	3	3	2	3.53	2
167.	1	1	1	1	3	1
168.	1	1	1	1	2.58	1
169.	3	3	3	3	3.36	3
170.	1	1	1	1	3	1
171.	2	1	1	2	1.35	2
172.	2	1	1	2	2	2
173.	1	3	3	1	0	1
174.	2	3	3	2	2	2
175.	3	3	3	3	3	3
176.	1	3	3	1	1	1
177.	2	3	3	2	1	2
178.	1	1	1	1	2.9	1

B. The table below represents partial training set of 5K sample set. Trend of training data set for neural operator EXTENDED AND is shown in the table below. The table has five inputs and one target output.

Index	Input A	Input B	Input C	Input D	Input E	Target
1.	0	0	1	0	1	1
2.	0	0	2	0	2	2
3.	0	1	0	0	1	1
4.	0	1	1	0	1	1
5.	0	1	2	0	1	1
6.	0	1	2	0	2	2
7.	0	2	0	0	2	2

8.	0	2	1	0	1	1
9.	0	2	1	0	2	2
10.	0	2	2	0	2	2
11.	1	0	1	1	1	1
12.	1	0	2	1	2	2
13.	1	1	0	1	1	1
14.	1	1	1	1	1	1
15.	1	1	2	1	1	1
16.	1	1	2	1	2	2
17.	1	2	0	1	2	2
18.	1	2	1	1	1	1
19.	1	2	1	1	2	2
20.	1	2	2	1	2	2
21.	2	0	1	2	1	1
22.	2	0	2	2	2	2
23.	2	1	0	2	1	1
24.	2	1	1	2	1	1
25.	2	1	2	2	1	1
26.	2	1	2	2	2	2
27.	2	2	0	2	2	2
28.	2	2	1	2	1	1
29.	2	2	1	2	2	2
30.	2	2	2	2	2	2

C. Training and testing data set for approximately 240 samples are presented in the table below. This partial set of data was part of the experimental analysis in chapter 6

	ODD Operator Test			**ODD Operator Train**			**Even Operator**		
Index	**Input A**	**Input B**	**Output**	**Input A**	**Input B**	**Target**	**Input A**	**Input B**	**Output**
1.	3	2	3	1	1.32	1	1	1.33	0
2.	2	1	1	2	2.06	0	1.32	0	0
3.	2	2	0	1.23	1.34	0	0	0	0
4.	1	1	1	0	1	1	1	2	2
5.	2	3	3	1	2	1	3	3	0
6.	2	3	3	2.87	1	1	3	0	0
7.	2.87	3	3	1	2.59	1	1	0	0

8.	0	1	1	2.76	3	3	0	1.32	0
9.	1	0	1	1.54	1	1	1	3	0
10.	2	2	0	1	1.24	1	3	3	0
11.	3	1	3	1	3.44	0	3	3	0
12.	1	0	1	0.47	0	0	0	3	0
13.	1	3.46	0	1.13	1.15	0	2	1	2
14.	3	1	3	1.15	1	1	2	2	2
15.	2	0	0	1	0	1	3	3	0
16.	1	2	1	1	1.9	1	2	3	2
17.	3	3	3	1	2	1	1	3	0
18.	3	3	3	3.33	3.27	0	2	2	2
19.	1	0	1	2	1.27	0	2	2	2
20.	2	0	0	3	3.02	0	3	3	0
21.	0	2	0	2.49	1.22	0	1	3.59	0
22.	1	0	1	3.18	1	0	1.35	2	2
23.	3	2	3	2.39	1.27	0	1	1	0
24.	2	3	3	3.35	2.64	0	0	2	2
25.	3	3	3	3	2	3	1	3.51	0
26.	3.3	3	0	1	0.48	1	2	1	2
27.	3	0	3	0	2.91	0	0	2	2
28.	3	2.96	3	3.36	1.18	0	1	1	0
29.	2.67	3	3	3	0.47	3	0	3	0
30.	2	1	1	2.81	2	0	3	3	0
31.	2	2	0	1	3.18	0	1	0	0
32.	1	3	3	2.94	1.23	0	3	3	0
33.	1	2	1	2.83	2	0	3	2	2
34.	2	0	0	1	0	1	1.29	2	2
35.	1	1	1	1	1.63	1	1	1	0
36.	3	0	3	2.82	1	1	3	3	0
37.	3	3	3	1	1	1	1	1	0
38.	2	3	3	0	1.24	0	1	3	0
39.	2	2.72	0	3	3	3	1	3	0
40.	2	3	3	3	2.84	3	1	3	0
41.	1	0	1	3	2.93	3	3	3	0
42.	3.35	2	0	1.16	1.26	0	1	2	2
43.	3	2	3	1	3.44	0	2	1	2
44.	3	3	3	3.03	1.23	0	1	2	2
45.	3.42	1	0	2	3	3	1	3	0
46.	1	3	3	2.32	2.86	0	1	1	0
47.	1	1	1	1	1	1	2	0	2
48.	1	3	3	2	0.47	0	3	2	2

49.	1	2	1	3.02	1.25	0	3	0	0
50.	3	3	3	1.28	2	0	2.83	2.57	0
51.	3	2	3	1	2.83	1	0	1.33	0
52.	1	1	1	2.28	3	3	1	3	0
53.	1.3	1	1	3	1	3	1	2	2
54.	1	2	1	2	1.28	0	1	3	0
55.	3.54	2.56	0	3	2.62	3	3	3	0
56.	1	2	1	1.66	1.26	0	1	2.61	0
57.	1.33	2	0	1.16	2.84	0	3	2	2
58.	1	3	3	1.6	0	0	1	3	0
59.	1.32	0	0	3	1.13	3	2	3	2
60.	0	2	0	2	0.48	0	1.29	1	0
61.	1	1	1	0.47	2	0	2	3	2
62.	3	3	3	1	2.72	1	1	0	0
63.	1	3	3	3.58	3.55	0	0	3	0
64.	3	2	3	1	2	1	1	3	0
65.	0	3	3	2.58	3	3	3	1	0
66.	3	2	3	2.76	1.13	0	1	1	0
67.	2	1.3	0	3	1	3	3	1	0
68.	1	3	3	1	1	1	3	1.32	0
69.	1	2	1	1	1	1	3	3	0
70.	3	0	3	1.17	2.87	0	0	3	0
71.	1	2	1	2.91	1.89	0	0	1	0
72.	2	0	0	2	1	1	2	3	2
73.	0	0	0	3	1	3	1	3	0
74.	3	2	3	0	3.43	0	2	0	2
75.	0	2	0	1.82	0.48	0	3.49	3	0
76.	3	2	3	1.6	2	0	1	3.58	0
77.	1	2	1	1.26	1.16	0	1	2	2
78.	1	3	3	1.33	2	0	3	3	0
79.	1.28	2.64	0	2.12	3.03	0	1	3	0
80.	1	3	3	2	3	3	2.91	2	2
81.	1.33	3	3	2.85	3	3	2	1	2
82.	2	3	3	2.85	2	0	1	2	2
83.	1	1.35	1	1	2.21	1	0	2	2
84.	0	1	1	3	0	3	1	2.69	0
85.	3	1	3	3.26	2	0	2	1	2
86.	0	2	0	2	2	0	2	3	2
87.	3	1	3	1.25	3	3	3.52	1.35	0
88.	1	2	1	2.54	1	1	1	2	2
89.	1	1	1	2.52	3	3	3	3.36	0

90.	0	3.57	0	2	1	1	1	1	0
91.	3.35	3	0	3	2.79	3	2	0	2
92.	1	1.28	1	2	3	3	2.98	3	0
93.	1	1	1	2.49	1.28	0	3	2	2
94.	3	0	3	3	1.19	3	2	2	2
95.	1	2	1	2.48	3	3	3	2	2
96.	1	0	1	2	3.03	0	2	0	2
97.	2	2	0	1	1.18	1	2	2.99	2
98.	2	3.4	0	1.3	0.46	0	2	1	2
99.	3	1	3	3.16	3.02	0	2	2	2
100.	1	2	1	2.16	2.79	0	0	2	2
101.	3	1	3	1.13	1.27	0	1	1	0
102.	1.31	2.61	0	2.39	2.88	0	1	3	0
103.	1	3	3	0	1.24	0	2	0	2
104.	3	1	3	0.47	2	0	3	1	0
105.	3.53	2.84	0	1.13	1.34	0	2	3	2
106.	1	1	1	3	2.85	3	3	3	0
107.	0	1	1	2.24	0.46	0	2	1.32	2
108.	1	3	3	2	1	1	3	2	2
109.	1	1	1	3	0	3	1	2	2
110.	1	3	3	1.17	2	0	2.62	1	0
111.	0	1	1	2.91	2	0	3	3	0
112.	2	3	3	0	1	1	1	3	0
113.	2	2	0	3.49	3.38	0	1	2	2
114.	2	1	1	1.17	2.77	0	3	2	2
115.	2	2	0	1	1	1	2	1	2
116.	1	2	1	1.21	1.15	0	0	1	0
117.	2	0	0	0.47	1	1	2	3.39	0
118.	2	3.48	0	3.18	3	0	2	3	2
119.	1	0	1	3	1	3	2	3.36	0
120.	1	2.96	1	2.42	2	0	3	2	2
121.	2.98	0	0	1.27	1	1	3	2	2
122.	3	0	3	1	3.56	0	1	2.58	0
123.	3	2	3	2.49	2.04	0	2	1	2
124.	3	2	3	2.21	0.47	0	2	1	2
125.	3	1	3	2.31	1	1	2	3.33	0
126.	1	3.54	0	0	1	1	1	3	0
127.	2	3	3	1	2	1	0	2	2
128.	1	3	3	0.48	2.94	0	2	2	2
129.	2	1	1	3.14	1.15	0	2	1	2
130.	1.35	3	3	0.46	0	0	3	1	0

131.	2	2	0	2.72	1	1	0	1	0
132.	1	1.34	1	1	3	3	3	1	0
133.	1	1	1	1.74	0.48	0	3	1	0
134.	2.76	1.33	0	2.24	1	1	0	0	0
135.	3.31	1	0	0.48	1.24	0	3	2	2
136.	1.35	1	1	2	2.74	0	3	2	2
137.	3	3	3	1.72	1	1	2	3	2
138.	1	0	1	1.28	3.1	0	1.33	1	0
139.	1	1	1	2	2	0	1	2	2
140.	2	0	0	0	1.26	0	3	2	2
141.	1	2	1	1.17	2	0	1.34	3	0
142.	3	2	3	2.7	1.19	0	3	3	0
143.	2	1	1	3.04	3.6	0	0	1	0
144.	2	3.37	0	1	1	1	1	0	0
145.	3	1	3	2	1	1	3.4	3	0
146.	0	2	0	1	3	3	3.5	1	0
147.	2	2	0	3.07	1.75	0	2.96	0	0
148.	0	3.51	0	2	1.95	0	1.34	3	0
149.	1.3	1.29	0	0.46	2.73	0	2	0	2
150.	1.32	3	3	1.34	3	3	2	1	2
151.	1	1	1	2.5	2.97	0	3	0	0
152.	1	3	3	2	0	0	2	3.53	0
153.	1.29	0	0	2.78	1	1	1	2	2
154.	2	3	3	3.52	0.46	0	1	2	2
155.	0	3.43	0	1	2	1	3	1	0
156.	2	3	3	1.24	1	1	3	3	0
157.	2	1	1	0.47	3	3	3	2	2
158.	1	3	3	3	1	3	3	1	0
159.	3	2	3	1.95	2	0	3	0	0
160.	3	2.68	3	2	1.22	0	2	3	2
161.	1	2	1	0.48	3	3	1	2	2
162.	3	3	3	1.23	3	3	2	3	2
163.	3.57	1	0	1.35	0.48	0	0	2.52	0
164.	0	1	1	1.55	2.27	0	2	1	2
165.	1.28	2	0	2	3	3	3	3	0
166.	3	1	3	1	1.16	1	2	1	2
167.	3	1	3	2	3	3	0	3	0
168.	1	2.98	1	2	3.13	0	1	1	0
169.	3	2	3	2.63	0.45	0	1.33	2	2
170.	2	2	0	1.31	3.48	0	3	1	0
171.	2	1	1	3.46	2.93	0	2	2	2

172.	3	1.33	3	1.27	1.17	0	3	0	0
173.	1	1	1	1	1	1	1.3	3	0
174.	1	1	1	2	2	0	1	2	2
175.	1	3	3	0.46	1	1	0	3	0
176.	2.7	3	3	0.47	3.54	0	2	1	2
177.	3.43	1	0	0.46	1	1	2	1	2
178.	3	3	3	1	3.45	0	2.5	0	0
179.	3	1	3	1.21	1.16	0	1	2.83	0
180.	1.32	3	3	1	1.21	1	0	3	0
181.	1	1	1	2.57	2	0	3.5	1	0
182.	2	3	3	0.46	3	3	2.84	0	0
183.	3	3	3	0.47	0	0	1	1	0
184.	0	2	0	1	0	1	2	1	2
185.	1	3	3	1.52	2	0	1.34	3	0
186.	1	2	1	2.38	1.63	0	3.37	0	0
187.	0	1	1	1.24	0.45	0	3	3	0
188.	3.42	1.33	0	2.76	0	0	3	2	2
189.	3.37	2	0	2	1	1	3.31	0	0
190.	2	3	3	1.29	2.46	0	3	3	0
191.	1	0	1	1.29	3	3	3	2	2
192.	0	1	1	2.97	1	1	3.51	1	0
193.	3	3	3	1	3	3	3	2	2
194.	2	1	1	1.57	3	3	3	3	0
195.	3.42	1.29	0	3	1.2	3	2	0	2
196.	3	3	3	3	1	3	2	1	2
197.	2.82	0	0	2.83	3	3	1	3	0
198.	1	1	1	1.33	2.92	0	1	2	2
199.	3	1	3	0	1.34	0	0	1	0
200.	3	2	3	1.34	3	3	3	2	2
201.	2	2	0	3.04	1	0	3	0	0
202.	3	1.3	3	1.14	1.3	0	3	2	2
203.	1	3.4	0	2	0.47	0	1	3	0
204.	3.59	3	0	2	2	0	0	2	2
205.	3	0	3	1.13	2	0	2	3	2
206.	3	2.95	3	3.44	1	0	1	1	0
207.	2	1	1	0.47	0.45	0	0	2	2
208.	3	0	3	1	2	1	2	2	2
209.	3	3	3	1	3	3	3	3	0
210.	1	1	1	3.17	3	0	3	3	0
211.	3.46	2	0	1.57	0.46	0	2.77	1	0
212.	1	1	1	3.14	3	0	1	3.34	0

213.	3	1	3	3	2.5	3	0	1	0
214.	3	3	3	3.27	1	0	1	2	2
215.	3	2	3	1.79	1.29	0	1	0	0
216.	3	2	3	1.23	2	0	3	2	2
217.	1	1.28	1	2.16	3.38	0	1	3.58	0
218.	0	1	1	3.55	1	0	0	2	2
219.	2	3	3	2	2	0	3	3	0
220.	2	3	3	2	1.2	0	2	2.55	2
221.	1	2.94	1	1.98	2.01	0	3	1	0
222.	0	1	1	2.3	2.7	0	1	3	0
223.	0	0	0	0.46	3.2	0	3	2	2
224.	3	3	3	0	2.46	0	2	0	2
225.	1	3	3	2	3	3	1	2	2
226.	3	2	3	1.24	1.26	0	2	1	2
227.	1.33	1	1	3.17	0.46	0	2	0	2
228.	3	1	3	1	2	1	2	2	2
229.	2	3	3	1.25	3.12	0	2	2	2
230.	1	1	1	0.45	2	0	2	1	2
231.	0	2	0	2.8	3	3	2	3	2
232.	2	3.41	0	2	0.47	0	2	1	2
233.	0	2.67	0	3.59	0.46	0	0	1	0
234.	1	1	1	1.18	2	0	1	3.55	0
235.	1	0	1	2	0.48	0	1	3	0
236.	1	3	3	1.14	2.36	0	1	1	0
237.	2.63	2	0	1.18	2.05	0	2	2	2
238.	3.54	1.3	0	2	2.73	0	1	3	0
239.	2	1	1	2	1.19	0	2	2	2
240.	3	1	3	2.78	3.42	0	3.42	1	0
241.	1	1	1	3	1.17	3	2	1	2
242.	2	2.59	0	1.79	2	0	2.96	1.3	0
243.	1	3	3	0.48	2	0	1	1	0

APPENDIX C – NZMDD BENCHMARK CIRCUITS (3-VALUED & 4-VALUED 2-VARIABLE FUNCTIONS)

The Table display 216 sample benchmark circuits. Both 3-valued and 4-valued benchmarks are presented side by side. In chapter 8, a total of 69,598 benchmark circuits were generated for investigation. The table provides a glimpse of what was analyzed.

Index	Benchmarks (3-Valued 2-Variable)	NonZero Minterm	Traditional Synthesis (PT Reduction)	Proposed Synthesis (PT Reduction)	Benchmarks (4-Valued 2-Variable)	Traditional Synthesis (PT Reduction)	Traditional Synthesis (PT Reduction)	Proposed Synthesis (PT Reduction)
1.	212221020	7	6	5	1233323033232323	15	13	9
2.	222112221	9	5	5	3130213023010232	12	16	10
3.	200201010	4	7	4	3120212302311310	13	16	11
4.	111102021	7	7	5	2131031313323033	14	13	9
5.	212122102	8	7	6	3011331021103303	12	11	7
6.	000201012	4	7	4	0030231132321232	13	13	9
7.	220100210	5	7	4	0300233320322131	12	11	8
8.	210021102	6	9	6	3322311122332113	16	9	9
9.	112101211	8	6	5	3200112323003333	12	10	8
10.	101102000	4	6	3	3332020023313202	12	11	8
11.	202122022	7	6	4	1032101013203122	12	15	9
12.	010101002	4	6	3	3313033002112032	12	10	7
13.	221022221	8	6	5	1202333331333123	15	10	8
14.	202111000	5	4	2	2310231222012013	13	14	11
15.	110200212	6	6	4	3313330303111313	14	11	7
16.	210121101	7	7	5	3321111313301011	14	10	8
17.	000202210	4	6	3	3112313201313130	14	15	10
18.	222020201	6	6	4	2012323332321110	14	11	7
19.	121012102	7	8	6	1003310022330221	11	11	8
20.	222202010	6	5	3	1333331230023331	14	10	9
21.	102122211	8	7	6	2123303231202103	13	16	11
22.	202222022	7	5	3	1210333003021220	11	13	7
23.	220210001	5	7	4	3112033333303233	14	9	7
24.	022202112	7	6	4	3203131220302100	11	14	9
25.	101120101	6	7	4	0321321330023003	11	12	9
26.	001110002	4	6	3	2133210332133002	13	13	11
27.	101012021	6	8	5	2123131203330203	13	14	9
28.	012202012	6	8	5	0123102333001133	12	12	9
29.	210212222	8	6	5	3010321133112223	14	11	9
30.	201220212	7	7	5	3233300312101031	12	11	7

31.	100101020	4	6	3	0100100230233330	9	10	6
32.	220120020	5	7	4	0223032002320011	10	12	7
33.	120120110	6	8	5	3002303233133332	13	11	8
34.	021221221	8	7	6	0331222211311333	15	8	7
35.	002122202	6	6	4	3302333313312010	13	10	7
36.	220110020	5	6	3	1201313031212311	14	14	10
37.	222101020	6	5	3	1130200302111333	12	11	8
38.	122001220	6	6	4	3333311020120331	13	10	7
39.	012020200	4	7	4	1110012031132333	13	9	7
40.	221012221	8	7	6	1230332313312323	15	12	9
41.	111102020	6	6	4	0213010023320130	10	11	8
42.	202011212	7	6	4	1313213210133213	15	14	10
43.	210222210	7	7	5	1323312302000023	11	12	9
44.	201102120	6	9	6	2233232003011033	12	13	8
45.	011021120	6	8	5	0203131033202313	12	15	9
46.	212222102	8	6	5	2323333232232303	15	12	8
47.	222212010	7	5	4	3033133310001133	12	8	6
48.	121211102	8	7	6	0301302113320030	10	13	9
49.	122200201	6	7	5	1323232031302233	14	14	9
50.	012200221	6	7	5	3200011333321133	13	10	8
51.	202002002	4	6	3	1212102031311313	14	16	8
52.	122222201	8	6	5	3013231132312313	15	14	11
53.	102222212	8	6	5	3223003123030231	12	13	9
54.	210101120	6	8	5	3133203010333133	13	11	8
55.	200101202	5	6	3	1120311301323131	14	13	9
56.	201122011	7	7	5	2133333321031330	14	11	9
57.	202221121	8	6	5	0113333330231132	14	10	8
58.	010220011	5	6	3	0111021101203332	12	10	7
59.	222202112	8	5	4	2013033332303010	11	14	8
60.	122012111	8	6	5	0221033211331232	14	12	9
61.	122101212	8	6	5	3312032301132011	13	13	9
62.	202100110	5	6	3	2033331303301320	12	11	8
63.	002002110	4	6	3	0203213211221031	13	12	9
64.	002000111	4	4	2	1132331303311031	14	11	9
65.	022122121	8	6	5	0232222203212223	14	11	8
66.	101102212	7	7	5	3303333333033120	13	9	6
67.	211110012	7	7	5	2332331333133313	16	8	8
68.	111202010	6	5	3	0130313330033203	11	10	7
69.	221200010	5	6	4	2313331023113330	14	12	9
70.	221221010	7	6	5	3333333330233132	15	9	7
71.	220000112	5	5	3	3113231133301113	15	9	8
72.	011202221	7	6	4	3203101132313333	14	10	7
73.	102101110	6	7	4	2201031123032010	11	14	8
74.	212222011	8	5	4	2133322131302322	15	12	10
75.	112212200	7	6	5	3211303312201213	14	12	9
76.	112112102	8	7	6	0333033103031312	12	13	6
77.	212200112	7	6	5	0213330232101320	12	15	11
78.	110020210	5	7	4	3033003203333200	10	10	6
79.	100100112	5	6	4	3313132123231002	14	12	9
80.	222121102	8	6	5	3231322310311033	14	12	9
81.	112212021	8	7	6	0301303301331313	12	13	7
82.	202012111	7	6	4	0103302313311133	13	11	8
83.	210100201	5	8	5	3033300313333333	13	7	5

84.	022022022	6	6	3	0320031203232001	10	14	9
85.	220212122	8	6	5	1223133000130332	12	12	9
86.	211110021	7	7	5	3232003333301013	12	12	6
87.	200110020	4	6	3	1133030333133031	13	12	6
88.	020002110	4	6	3	3311032322303033	13	11	7
89.	220010022	5	6	3	3303333320213123	14	10	7
90.	101002022	5	6	3	3203301000203123	10	12	8
91.	122112021	8	7	6	3133311021333023	14	11	9
92.	220200000	3	5	2	1332133320103102	13	13	10
93.	202111112	8	5	4	2330131032330231	13	13	9
94.	101201201	6	8	5	0232103021303323	12	14	9
95.	011021212	7	7	5	3231222303313133	15	11	9
96.	021101222	7	6	4	0321300113222303	12	14	10
97.	201111120	7	7	5	2321110031311303	13	14	8
98.	222212112	9	5	5	0212003333101203	11	13	8
99.	121212122	9	6	6	2023300330031332	11	11	7
100.	220220101	6	6	3	0123103321223033	13	11	8
101.	202022211	7	6	4	3111301133130310	13	10	8
102.	100212101	6	6	4	1120131130311113	14	11	8
103.	010101010	4	6	3	3313233310331010	13	11	6
104.	210000220	4	6	3	0213010332330333	12	12	8
105.	210100202	5	7	4	0111010310300010	8	12	6
106.	011122122	8	6	5	0333313013313033	13	10	6
107.	122000201	5	6	4	2230321031323313	14	13	10
108.	222012111	8	5	4	1033233032130001	11	11	8
109.	122022020	6	6	4	0001203302011031	9	12	7
110.	212211000	6	5	4	3203221220033333	13	9	7
111.	200220011	5	6	3	0303001212133312	12	14	10
112.	010122101	6	6	4	0332311003333033	12	10	6
113.	112001202	6	6	4	2311033331320323	14	13	9
114.	210120002	5	8	5	1232001301010332	11	14	7
115.	101001121	6	6	4	0303033323200203	10	14	7
116.	220101022	6	6	3	3312333231222311	16	11	11
117.	112111212	9	5	5	2312133333011202	14	12	9
118.	221220110	7	6	4	3333101133331003	13	7	5
119.	200012021	5	8	5	0003313310133220	11	11	7
120.	122022221	8	6	5	3333330021300233	12	10	7
121.	211212010	7	6	5	1321220323333002	13	11	9
122.	200102122	6	7	5	1333111302213210	14	11	9
123.	020000120	3	6	3	3332032032002011	11	11	8
124.	000122102	5	6	4	3310013131322203	13	14	9
125.	100220100	4	6	3	3113333330320311	14	10	7
126.	010202020	4	6	3	2311111130323113	15	10	8
127.	111010110	6	5	3	3333033233312323	15	10	7
128.	010210120	5	8	5	3110020213130132	12	15	7
129.	220110021	6	7	4	2313320111033311	14	13	10
130.	111212101	8	5	4	1333333133123312	16	10	10
131.	002200021	4	7	4	3312110001333203	12	11	8
132.	112112222	9	5	5	1300311032333113	13	10	8
133.	220222101	7	5	3	2223010232033303	12	11	7
134.	000211220	5	5	3	1333203303233223	14	11	8
135.	121120011	7	7	5	0301132223133033	13	13	9
136.	110022201	6	7	4	0032131233310223	13	12	9

137.	221220112	8	6	5	2321010033230332	12	11	8
138.	100110002	4	6	3	3321021113330233	14	11	9
139.	201020222	6	6	4	3312001311301111	13	10	8
140.	101011110	6	6	3	3313030011300332	11	10	7
141.	211020021	6	7	5	0230333112303101	12	13	9
142.	211021222	8	6	5	3232311223232213	16	14	10
143.	022000200	3	5	2	1330223233330233	14	9	7
144.	100202222	6	5	3	3310302100330233	11	12	8
145.	020012022	5	7	4	2031232323121320	14	15	11
146.	122021022	7	7	5	3313133333312131	16	10	9
147.	222122101	8	5	4	3310033313002101	11	12	7
148.	210220211	7	7	5	3100121333323020	12	13	9
149.	112201021	7	8	6	0320222120313012	12	13	10
150.	120211112	8	7	6	3303333133201330	13	10	7
151.	021212212	8	7	6	3132333121331330	15	12	10
152.	122022102	7	7	5	3133103230311221	14	12	9
153.	212110211	8	6	5	2032333033323232	14	11	7
154.	222002221	7	5	4	1131230333301200	12	11	7
155.	221212120	8	7	6	0031323200320110	10	12	7
156.	220012221	7	7	5	3303031301032023	11	14	7
157.	121002012	6	7	5	3033312300233232	13	12	8
158.	221210212	8	7	6	1103333120233303	13	11	7
159.	021111202	7	6	4	3013332233000320	11	10	7
160.	100210222	6	6	4	0313322213301232	14	13	9
161.	102112222	8	6	5	2313133020310022	12	13	9
162.	000020121	4	5	3	3233330120331101	13	10	7
163.	201120020	5	8	5	2011023200322033	11	13	8
164.	211212000	6	5	4	1323321202221010	13	14	7
165.	201112011	7	7	5	1133221123002132	14	10	9
166.	202020112	6	6	4	3033003313003333	11	8	5
167.	222102122	8	6	5	0112302001030333	10	13	7
168.	101200202	5	6	3	0313312110002113	12	13	9
169.	220000220	4	5	2	3301223232222300	13	10	8
170.	201020001	4	7	4	2023331323010323	13	14	9
171.	120002020	4	7	4	2101030321233010	11	16	7
172.	221022222	8	5	4	2002200332333011	11	10	7
173.	122002000	4	5	3	1131333233333333	16	6	6
174.	020110211	6	6	4	2333111013321013	14	11	8
175.	111111020	7	4	3	2013333322321213	15	11	9
176.	222212110	8	5	4	3110102230002210	10	11	7
177.	111210000	5	5	3	3232313202010331	13	15	9
178.	112011112	8	6	5	3333301133120113	14	10	8
179.	222121022	8	5	4	3321333131333330	15	9	8
180.	111222201	8	5	4	2230033323233220	13	12	7
181.	021100012	5	8	5	3302021031300000	8	11	6
182.	120102111	7	7	5	1120201013331320	12	13	9
183.	222221222	9	4	4	0301120312330313	12	15	10
184.	022001100	4	6	3	1012303330030122	11	11	6
185.	220222201	7	6	4	0002233312121233	13	11	8
186.	211102022	7	7	5	3220233323202233	14	11	8
187.	122202021	7	7	5	3001332223322323	14	11	8
188.	010200220	4	6	3	1331033330033133	13	8	6
189.	122100020	5	6	4	3333021100200311	11	9	6

190.	001210021	5	8	5	3310123033131031	13	12	9
191.	222210221	8	6	5	1203133333322330	14	11	9
192.	022201221	7	7	5	2010302023103333	11	13	8
193.	220202200	5	6	3	2233210223130133	14	12	9
194.	120212122	8	7	6	3322333121221103	15	9	8
195.	012010101	5	7	4	2132023230322312	14	14	10
196.	120120001	5	8	5	3323001113310132	13	10	8
197.	201211220	7	7	5	0313313333131123	15	11	9
198.	111002010	5	5	3	3310333131131320	14	11	9
199.	202022120	6	7	4	3323212333233233	16	10	9
200.	111121221	9	5	5	2332033333013312	14	10	8
201.	221100101	6	6	4	0001333013222133	12	10	8
202.	200000112	4	5	3	3232333322200001	12	9	5
203.	201022010	5	7	4	1133312133000011	12	10	7
204.	121220221	8	6	5	3022333311323032	14	11	8
205.	202012212	7	7	5	3123313033331012	14	12	8
206.	202011012	6	7	4	0122331002113302	12	12	8
207.	202102021	6	8	5	2110133132331233	15	10	9
208.	200220220	5	6	3	1120011133103331	13	10	7
209.	022011122	7	6	4	3233123213200200	12	12	9
210.	200021112	6	7	5	2333202132130223	14	12	9
211.	212101121	8	6	5	3123010332010110	11	13	9
212.	110202221	7	6	4	3311322330133331	15	9	8
213.	011122100	6	6	4	1232213121320321	15	15	12
214.	112211222	9	5	5	3213002032130022	11	10	8
215.	110222201	7	6	4	0021132333103103	12	13	9
216.	211200112	7	6	5	3322331213311031	15	10	9

APPENDIX D – SPICE NMOS AND PMOS LEVEL 3 MODEL PARAMETER

A. N-channel MOS - 1um CMOS TECHNOLOGY FILE

```
* 1 um Level 3 models
*
* Don't forget the .options scale=1u if using an Lmin of 1
* 1<Ldrawn<200 10<Wdrawn<10000 Vdd=5V
.MODEL Mbreakn NMOS LEVEL = 3
+ TOX = 200E-10 NSUB = 1E17 GAMMA = 0.5
+ PHI = 0.7 VTO = 0.8 DELTA = 3.0
+ UO = 650 ETA = 3.0E-6 THETA = 0.1
+ KP = 120E-6 VMAX = 1E5 KAPPA = 0.3
+ RSH = 0 NFS = 1E12 TPG = 1
+ XJ = 500E-9 LD = 100E-9
+ CGDO = 200E-12 CGSO = 200E-12 CGBO = 1E-10
+ CJ = 400E-6 PB = 1 MJ = 0.5
+ CJSW = 300E-12 MJSW = 0.5
```

B. P-channel MOS - 1um CMOS TECHNOLOGY FILE

```
* 1 um Level 3 models
*
* Don't forget the .options scale=1u if using an Lmin of 1
* 1<Ldrawn<200 10<Wdrawn<10000 Vdd=5V
.MODEL Mbreakp PMOS LEVEL = 3
```

```
+ TOX = 200E-10 NSUB = 1E17 GAMMA = 0.6

+ PHI = 0.7 VTO = -0.9 DELTA = 0.1

+ UO = 250 ETA = 0 THETA = 0.1

+ KP = 40E-6 VMAX = 5E4 KAPPA = 1

+ RSH = 0 NFS = 1E12 TPG = -1

+ XJ = 500E-9 LD = 100E-9

+ CGDO = 200E-12 CGSO = 200E-12 CGBO = 1E-10

+ CJ = 400E-6 PB = 1 MJ = 0.5

+ CJSW = 300E-12 MJSW = 0.5
```

Model Parameters taken from (Xioing 2010)