APPLICATION OF MULTIPLE – VALUED LOGIC IN FINITE FIELD ARITHMETIC

by

Taner ÇEVİK

May 2008

APPLICATION OF MULTIPLE – VALUED LOGIC IN FINITE FIELD ARITHMETIC

by

Taner ÇEVİK

A thesis submitted to

The Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

May 2008
Istanbul, Turkey

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Bekir KARLIK
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Tuğrul YANIK
Supervisor

Examining Committee Members

Assist. Prof. Dr. Tuğrul YANIK ________________________

Assoc. Prof. Dr. İhsan Ömür BUCAK ________________________

Assoc. Prof. Dr. Onur TOKER ________________________

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

Assist. Prof. Dr. Nurullah ARSLAN
Director

APPLICATION OF MULTIPLE – VALUED LOGIC IN FINITE FIELD ARITHMETIC

Taner ÇEVİK

M. S. Thesis - Computer Engineering
May 2008

Supervisor: Assist. Prof. Dr. Tuğrul YANIK

ABSTRACT

Parallel to the technological improvements, transmisson speed of data over data lines expressed as throughput has become more important than ever before. Logical circuit elements used in integrated circuits are binary systems, which cause speed constraints. Limitations at speed constrain the throughput. Although the attractive research area, circuit density has improved to some degree. The major reason for that is the limitations caused by interconnections. Consequently, research has focused on how to overcome the problems caused by interconnections. Multi-valued is an interesting approach which can satisfy our performance requirements. Mult-value logic does not just increase the amount of data carried over data lines, it also increases the speed of arithmetic operations such as addition. However, speed is not the only requirement to be considered. Power consumption and physical area requirements are other important issues to be considered. So, all of those issues have to be considered as a whole.

Galois Field arithmetic became an important topic for its use in cryptographic algorithms, such as Eliptic Curve Cryptography and AES (Advanced Encryption Standard). Recently there is some interest in fields with characteristic other than two, which brings multi-valued logic into mind.

In this thesis, we tried to collate those two important topics, multi-value logic and galois field arithmetic. Since there is no a standard methodology like binary systems it is not easy to realize all Galois Fields operations such as serial multiplication and exponantiation. In this thesis, basic addition and multiplication operations are partially implemented and realized by multi value logic. We ried to compare the complexity of a bit serial multiplier based on multi value logic gates to a binary bit serial multiplier.

Keywords: Multi-Value Logic, Multiple-Valued Logic, Galois Fields

ÇOKLU DEĞERLİ MANTIK DEVRELERİNİN SONLU CİSİM CEBİRİNDE KULLANIMI

Taner ÇEVİK

Yüksek Lisan Tezi – Bilgisayar Mühendisliği
Mayıs 2008

Tez Yöneticisi: Yrd. Doç.Dr. Tuğrul YANIK

ÖZ

Teknolojik gelişmelere paralel olarak verinin iletim hatları üzerinden iletim hızı öncesine göre çok daha büyük bir önem kazanmıştır. Entegre devrelerde kullanılan manıtksal devre elemanları ikili sistemde çalışmaktadır. Bu sebepten dolayı veri iletim hızında bir sınırlama doğmaktadır. Bu kısıtlamayı aşabilmek için çeşitli ve etkileyici çalışmalar yapılmasına rağmen, devre yoğunluğu, yani devre alanın azaltılması ancak belirli bir seviyeye gelebilmiştir. Bunun temel sebebi, karşılıklı bağlantı yollarının getirmiş olduğu sınırlamalardır. Dolayısıyla, araştırmacılar, bağlantı yollarının getirdiği kısıtlamalar ve veri yolları üzerinden nasıl daha fazla veri taşınabileceği üzerine yoğunlaşmışlardır. Bundan yola çıkarak, çok değerli mantık, bu beklentileri karşılayabilmek amacıyla ortaya atılmıştır. Çok değerli mantık sadece veri hattı üzerinden taşınabilecek veri miktarını artırmakla kalmaz. Aynı zamanda toplama gibi artimetik işlemlerin hızını da artırmaktadır. Bütün bunların yanında, hız tek başına düşünülecek ve önem verilecek konu değildir. Hıza ek olarak güç sarfiyatı, ve fiziksel devre alanı da önem verilmesi gereken diğer önemli konulardır. Dolayısıyla, bütün bu faktörleri bir bütün olarak değerlendirmek ve ele almak gerekmektedir.

Galois Fields, sonlu durum cebiri bazı kriptografik algoritmalar (ECC, AES) için büyük önem taşımakta ve temelini oluşturmaktadır. Son zamanlarda karakteristiği ikiden farklı olan alanlara olan ilgi akla çoklu mantık devrelerini getirmektedir.

Bu tezde, yukarıda bahsedilen iki konu harmanlanarak, sonlu durum cebirinde kullanılan temel aritmetik işlemlerin, çoklu mantık ile gerçeklenmesine çalışılmıştır. Çoklu mantık devrelerine dayanan bir seri çarpma devresi karmaşıklık bakımından ikili mantığa dayanan bir seri çarpma devresi ile kıyaslanmıştır.

Anahtar Kelimeler: Çoklu mantık, Sonlu Durum Cebiri

DEDICATION

To my family

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I am grateful to my adviser, Assist. Prof. Dr. Tuğrul Yanık, for giving me the opportunity to study and conduct research for constitution of this thesis. His knowledge and his logical way of thinking have been of great value for me. I appreciate his patience, understanding and encouragement throughout my thesis study.

Next, I am deeply indebted to Assoc. Prof. Dr. Onur Toker and Assoc. Prof. Dr. İhsan Ömür Bucak whose assist and stimulating suggestions were great help to me.

And huge thanks to all my friends at the Department of Electronics Engineering, especially Selahattin Nesil, Yusuf Adıbelli and Zafer Çatmakaş.

I would like to give my special thanks to my whole family whose patient love enabled me to complete this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOL/ABBREVIATION

$\bullet$	Min operator
$\boxplus$	Tsum operator
$\overline{x}$	Complement of x
${}^{a}x^{b}$	Literal of x
$x^{m}\rightarrow$	Clockwise Cycle
$x^{m}\leftarrow$	Counter-Clockwise Cycle
$+$	Max operator
$\oplus$	Modsum operator
$x\,\vert_{a}^{b}$	Upper Threshold
${}_{a}^{b}\vert\,x$	Lower Threshold
4B3T	4 (four) Binary 3 (three) Ternary
MIPS	Microprocessor without Interlocked Pipeline Stages
SOP	Sum Of Products
MOS	Metal Oxide Semiconductor
CMOS	Complementary Metal Oxide Semiconductor
CCD	Charge-Coupled Device (CCD)
MVL	Multi-Value Logic
AES	Advanced Encryption Standard
GF	Galois Fields
W	Transistor width
L	Transistor length

CHAPTER 1

INTRODUCTION

Multi-value logic is processing of arithmetic and logic operations at radixes higher than two. It has been expected to solve the problems of today's integrated circuits such as dissipation of heat, current leakage and problems that occur due to squeezing more transistors into less silicon (Omnibase Logic, 2006).

It is mentioned above that multi-value logic uses radixes higher than two. In daily life, people are used to use base ten and computers are designed to work at base two. However, sometimes working at higher radixes can be advantageous. For example, consider how to express number $(436)_{10}$ at different radixes:

Base	**Number**
16	1B4
8	664
4	12310
3	121011
2	110110100

As it is shown above, $(436)_{10}$ can be expressed with 9 digits at radix 2. If we map every digit with a single data line, this means 9 data lines. Same number is expressed with 6 digits at radix 3, 5 digits at radix 4, 3 digits at radix 8 and 3 digits at radix 16. We can realize that, multi-value logic means more information and less area (Omnibase Logic, 2006).

A sample application area where this advantage can be of used is the area of cryptography. An encryption performed with 32 digits at radix 2 could be broken with a 100 MIPS processor in 2.2 minutes. However, same encryption performed at radix 3 could be broken with same processor at 643 days. And the same encryption made at radix 4 could be broken with same processor at 175 centuries. The reason of this huge difference is that, when a brute-force attack is applied to this 32 digit information, there are 2^{32} possibilities. However, if the data is encrypted at radix 3, there are 3^{32} possibilities for brute-force attack. And if it is encrypted at radix 4, there are $4^{32} = 2^{64}$ possibilities. This numbers show the huge advantage at security if multi-value logic can be implemented (Omnibase Logic, 2006).

Ordinary comprehension at radix two is that, logic 1 means True, and logic 0 means False. However, in multi-value logic there is no such a classification. Logic 0 is just one of the states. True-False states at binary logic have no meanings at multi-value logic. In binary system, logic 1 is equivalent with arithmetic 1. This is why computers make calculations easily. However it is not convenient practically, theorically it is possible to map logic 1 with artihmetic 2. For example, during calculation of correlation properties of wireless communication, logic 0 is mapped to arithmetic -1 (Lablans, 2005-2007).

As it was mentioned before, multi-value logic is switching between discrete states at radixes higher than 2. For example, in ternary logic which means radix 3, there are three discrete states. Let assign symbolic characters ‘1’, ‘2’, ‘3’ to those discrete states. Those numbers are absolutely symbolic. They do not mean anything. This means, any other symbols can be used to represent those discrete states. Below, it is shown a signal at radix 3. As it is seen from Figure 1.1, a multi-value signal can carry more information than a binary signal:

signal x = [2 0 2 1 2 2 1 1 2]

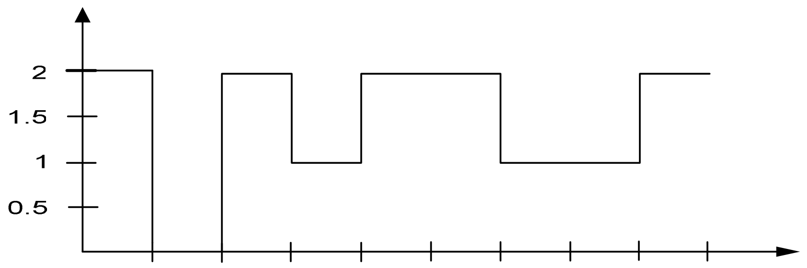


Figure 1.1 Multi-value signal representation (Lablans, 2005-2007)

Another considerable point is that, in binary system, complement of state ‘0’ is state ‘1’ and complement of state ‘1’ is state ‘0’. Because, there is no another state in binary system. However, in multi-value logic, since there are more than two states, complement of a state does not proceeded as in binary systems. A method defined in post algebra for complement operation is calculated as shown below:

For radix r, $\overline{x} = (r - 1) - x$

Ex: For radix 4:

$\overline{0} = (4 - 1) - 0 = 3$

$\overline{1} = (4 - 1) - 1 = 2$

$\overline{2} = (4 - 1) - 2 = 1$

$\overline{3} = (4 - 1) - 3 = 0$

Another interesting point is that, multi-value signals can be processed at radix 2. Most striking example about this topic is “ transmission line coding ”. “ 4B3T ” codes break data into 4 bits of pieces and after that convert those binary pieces to their 3 bits equivalences at radix 3. Data is transmitted over transmisson lines at radix 3. Recipient side, takes those 3 bits multi-value pieces of data and converts them to 4 bits of equivalences at radix 2. By this way, bandwith has been increased indisputably because 3 bits of data is enough instead of 4bits of data. If this advantage is compared by the disadvantage of encode-decode costs, it is worth (Lablans, 2005-2007).

In binary systems, 70% of circuit area is allocated to interconnection, 20% is for isolation, 10% is for circuit elements. Thus, interconnection is a very important issue. By descent of number of interconnections, number of output pins of the circuit also decresases. For reliability and strength of the circuit, minority of pins is better. Hence, there is a constraint for the number of pins on a circuit. This is called as “pinout” problem. This is another advantage of multi-value logic (Butler, 1995).

Also, using multi-value signals for point-to-point communication around 1 km can be very advantageous. Distances can be very efficient. Thus, signaling can be controlled by distributed devices where it is worth because of wiring costs (Smith, 1981).

Advantages gained by applying multi-valued logic to circuits are shown by Kameyama (Kameyama et al., 1988) at Figure 1.2:

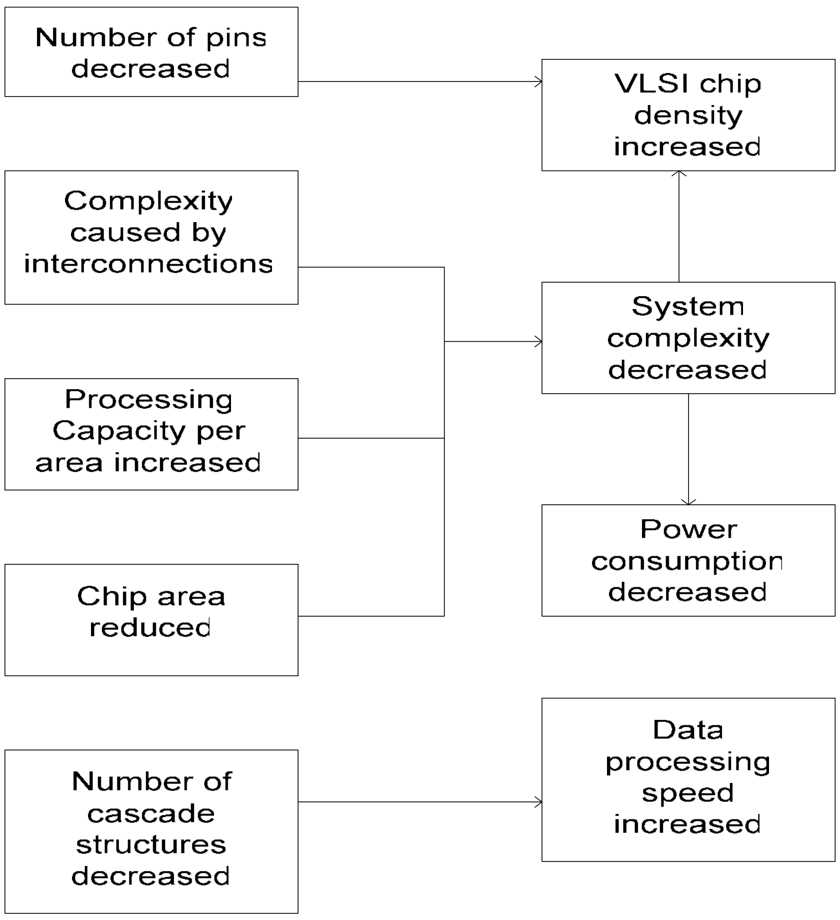

Number of pins decreased
VLSI chip density increased
Complexity caused by interconnections
System complexity decreased
Processing Capacity per area increased
Power consumption decreased
Chip area reduced
Data processing speed increased
Number of cascade structures decreased

Figure 1.2 Advantages of multi-value logic

Fuzzy Logic, one of the important topics of Artificial Intelligence, has emerged from the idea multi-value logic. Some scientists, made researches to define the importance of multi-value logic for fuzzy logic. Olmsted (Olmsted, 1998) is one of those scientists. Furthermore, Cardoli (Cardoli and Schaerf, 1996), has showed how to form a knowledge base in artificial intelligence by using multi-value logic (Nascimento, 2001).

CHAPTER 2

MULTI VALUE LOGIC

2.1 HISTORICAL EVOLUTION

Early multi-value logic circuits that draw interest were ones at radix 3. Those early radix 3 circuits were derived from their binary counterparts. Also, they were working over passive circuit elements rather than active ones. Evolution at multi-valued logic has reached to a state that commercial application areas have emerged (Smith, 1981).

If we inspect the historical evolution of multi-valued logic, we can see that the work of Post (Post, 1921) published in 1921 has contributed very much to the evolution of multi-value logic. Meanwhile, before 1965s there had been some studies about hardware such as radix 3 computer SETUN (Epstein et al., 1974) that was designed by Russians. Subsequent studies were done to provide a standard about multi-value logic and to form set of elements easy to implement and realize. Late 1960s, interest drawn to multi-value logic circuits has been raisen to a noticable level. As a result of this interest, two important studies have been presented. First one is the study done by Allen and Givone at 1968 (Allen and Givone, 1968). In this work, one of fateful elements of multi-value logic, “literal” element has been defined. Other important study belongs to Vrasenic, Lee and Smith perfomed at 1970 (Vranesic et al. 1970). In this study, other fateful element of multi-value logic, “cycle” element has been specified. Furthermore, in this study, it is shown how to apply analogue techniques to multi-value logic. Subsequent developments were specified at publications of Vranesic and Smith (Vranesic et al., 1974),(Vranesic and Smith, 1974) and later on Smith (Smith, 1976), Vranesic and Smith (Vranesic and Smith, 1977) and then Dao (Dao, 1981).

As a result of those studies, element sets have been emerged that can be applied to many technologies (Smith, 1981).

2.2 POST ALGEBRA

Functions in multi-value logic can be expressed as sum of products like in binary system. But, there is not just a single representation as it is done in binary. There are more than one representation of “individual product terms”. Representation of a function at the sum of products format changes according to the type of unary operators used. Besides that, even the unary operator set used is same, if the operators used to combine those sum-of-products terms are different, such as tsum, modsum, max, min; realization of the function differs from each other. Any multi-valued function can be represented as sum-of-products as shown below:

$f(x_1, x_2, ..., x_n) = P_1$ <sum operator> P_2 <sum operator> ... <sum operator> P_m

Every P_i expresses a product term and <sum operator> expresses an operator such as tsum, modsum, max (Jain et al., 1995).

As it is emphasized above, use of different operators provides different representations at realization of functions. Kerkhoff (Kerkhoff, 1984), McCluskey (McCluskey, 1979), Allen and Givone (Allen and Givone, 1968) achieved to realize different representations of functions by using different operators (Jain et al., 1995).

Representation of functions at SOP format is easy according to multi-value logic. As Tirumalai and Butler (Tirumalai and Butler, 1989) expressed, if min/max operators are used both, function can be realized with a minimal expression which consists of only prime implicants. If min and tsum operators are used as combining operators, for expressing the function, all of the implicants have to be considered (Jain et al., 1995).

Some fundamental functions used at multi-value logic are explained detailed below. Let R be a set consists of integer values from 0 to r-1 which are mapped into logical states of the set X that contains the logical states at radix r. F(x) is the function that maps those integer values of R to states in the set X (Jain et al., 1993).

$R = \{0, 1, 2, ..., r\text{-}1\}$,

$X = \{x_0, x_1, ..., x_n\}$, $x_i \in R$,

$F(x) = R^n \longrightarrow R$

2.2.1 min:

For radix = r , $R = \{0, 1, ..., r\text{-}1\}$, $X = \{x_1, x_2, ..., x_n\}$, $x_i \in R$

$\min(x_1, x_2, ..., x_n) = x_1 \bullet x_2 \bullet ... \bullet x_n$ = minimum of $(x_1, x_2, ..., x_n)$

Ex: $r = 4$, $R = \{0, 1, 2, 3\}$,

$x_1 = 2$, $x_2 = 1$, $x_3 = 3$,

$\min(x_1, x_2, x_3) = 2 \bullet 1 \bullet 3 = 1$ (Jain et al., 1993)

2.2.2 tsum:

For radix = r , $R = \{0, 1, ..., r\text{-}1\}$, $X = \{x_1, x_2, ..., x_n\}$, $x_i \in R$

$\text{tsum}(x_1, x_2, ..., x_n) = x_1 \boxplus x_2 \boxplus .... \boxplus x_n$ = minimum of $(x1 + x2 + ... + x_n , r\text{-}1)$

Ex: $r = 5$, $R = \{0, 1, 2, 3, 4\}$,

$x_1 = 3$, $x_2 = 2$, $x_3 = 3$,

$\text{tsum}(x_1, x_2, x_3) = 3 \boxplus 2 \boxplus 3 = 4$

2.2.3 complement:

For radix = r , $R = \{0, 1, ..., r\text{-}1\}$, $X = \{x_1, x_2, ..., x_n\}$, $x_i \in R$

$\overline{x} = (r\text{-}1) - x$

Ex: $r = 5$, $R = \{0, 1, 2, 3, 4\}$,

$x = 1$,

$\overline{x} = \overline{1} = (5 - 1) - 1 = 3$

2.2.4 literal:

For radix = r , $R = \{0, 1, ..., r-1\}$, $X = \{x_1, x_2, ..., x_n\}$,

x_i, a, b Є R ,

$a \leq b$,

$$^{a}x^{b} = \begin{cases} 1 \text{ (binary high)} & \text{if } a \leq x \leq b \\ 0 \text{ (binary low)} & \text{else} \end{cases}$$

Ex: r = 4, R = {0, 1, 2, 3},

x = 2 , a = 1 , b = 2 ,

$^{a}x^{b} = {}^{1}2^{2} = 1$ (binary high)

2.2.5 clockwise cycle:

For radix = r , $R = \{0, 1, ..., r-1\}$, $X = \{x_1, x_2, ..., x_n\}$,

x_i, m Є R

$x^{m}{}_{\rightarrow} = (x + m) \bmod r$

Ex: r = 4, R = {0, 1, 2, 3},

x = 3 , m = 2 , $3^{2}{}_{\rightarrow} = (3 + 2) \bmod 4 = 1$

2.2.6 counter-clockwise cycle:

For radix = r , $R = \{0, 1, ..., r-1\}$, $X = \{x_1, x_2, ..., x_n\}$,

x_i, m Є R

$x^{m}{}_{\leftarrow} = (x - m) \bmod r$

Ex: r = 4, R = {0, 1, 2, 3},

x = 3 , m = 2 , $3^{2}{}_{\leftarrow} = (3 - 2) \bmod 4 = 1$

2.2.7 max:

For radix = r , $R = \{0, 1, ..., r-1\}$, $X = \{x_1, x_2, ..., x_n\}$,

x_i Є R

$\max(x_1, x_2, ..., x_n) = x_1 + x_2 + ... + x_n$ = maximum of $(x_1, x_2, ..., x_n)$

Ex: $r = 4$, $R = \{0, 1, 2, 3\}$,

$x_1 = 2$, $x_2 = 1$, $x_3 = 3$,

$\max(x_1, x_2, x_3) = 2 + 1 + 3 = 3$

2.2.8 modsum:

For radix = r , $R = \{0, 1, ..., r-1\}$, $X = \{x_1, x_2, ..., x_n\}$, $x_i \in R$

$\text{modsum}(x_1, x_2, ..., x_n) = x_1 \oplus x_2 \oplus \ ... \oplus x_n = (x_1 + x_2 + \ ... + x_n) \bmod r$

Ex: $r = 4$, $R = \{0, 1, 2, 3\}$,

$x_1 = 3$, $x_2 = 1$,

$x_1 \oplus x_2 = (3 + 1) \bmod 4 = 0$ (Jain et al., 1993)

Example:

$x_2 \backslash x_1$	**0**	**1**	**2**
0	2	2	2
1	2	2	
2			2

Figure 2.1 Truth table of a function

Truth table of a function is shown in Figure 2.1. Representation of the function by means of SOP is shown below (Jain et al., 1995):

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

There are many operators present in post algebra. It must be possible for all those operators to be realized in real life. Monotonic systems are the easiest multi-value sytems to be realized. General structure of monotonic systems is shown below:

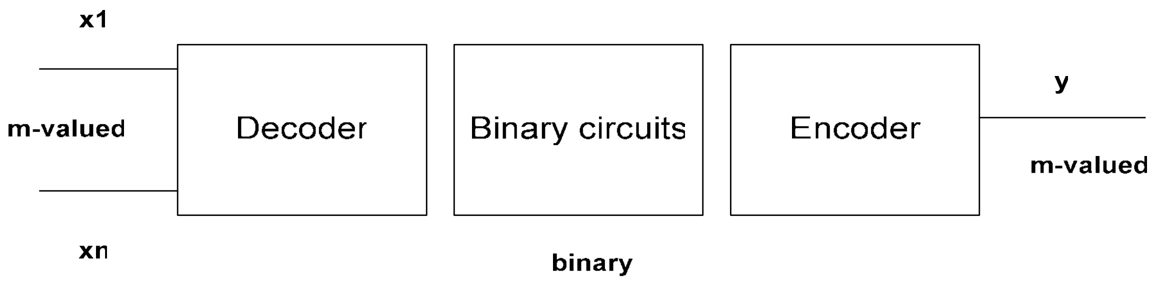


Figure 2.2 Structure Of Monotonic Systems

Although there are more than one representation of post algebra, they are isomorphic. For circuit design, monotonic systems are the easiest to realize (Etiemble and Israel, 1988).

2.3 CIRCUIT REALIZATIONS

Previously, people asked whether it would be possible to realize multi-value circuits. Depending on evolutions in microelectronic technology, number of circuit devices per unit area increased exponentially. Hence, more efficient, productive circuits are started to be produced (Etiemble and Israel, 1988).

Binary integrated circuits use boolean algebra and logical operators belong to this algebra. Binary circuits are voltage mode circuits which means use of continuous electrical signal voltage as input. 0V maps to logic 0, 5V maps to logic 1. In multi-value logic, there are more than two discrete states. In those circuits, both current and voltage signals can be used. Mapping to discrete logic states is provided by scaling of electrical signal (Etiemble and Israel, 1988).

As it is mentioned in previous chapter interconnection problem is the major problem for binary systems. Actually, with interconnection problems of binary systems has been faced at two levels. First one is the restrictive “edge” problem. Reason is that, circuit sizes grows exponentially as n^2. However, edge connection grows linearly as n. Another interconnection problem is that number of internal circuit elements grows with n^2, but the number of interconnections grows as $(n^2)!$ (Smith, 1981).

That is the point, a solution to all those problems can be to carry more data on trasmission lines. Thus, by decreasing the number of interconnections size of modules can be enlarged, number of them also can be increased. Furthermore, general purpose, public modules can be produced instead of application specific ones. In this manner, design costs can be decresaed extremeley. The reason why devices surround us work at binary system is that it is easy for circuit elements to switch between two states on-off (Smith, 1981).

In a sense, multi-value logic can be remedy for this intrconnection problem. Because, number of interconnections and pins decreases noticably. However, integrated circuit manufacturers have not concentrated on producing multi-valued circuits (Etiemble and Israel, 1988).

In multi-value logic, electric signals can be mapped more than two logical states. This way, it is possible for chip area to be reduced (Jain et al., 1993). Threshold circuit elements are one of the mostly used and important elements in multi-value logic. The reason is that, it quantizes the electrical signal in order to map to discrete logical states. Switching time of threshold element takes most of the overall latency time of this element (Hanyu et al., 1994).

Multi-value logic circuits use three types of electrical signal. Those are, current, voltage and charge. Charge-coupled devices use charge to carry data. Yamada, Fujishima, Nagasawa, and Gamou (Yamada et al., 1978) used charge coupled devices at multi-value memory design. Furthermore, Kerkhoff (Kerkhoff and Djkstra, 1979) has done some multi-value logical study and applications by using charge-coupled devices. Current, has been used especially in expressing multi value variables in I^2L circuits. Studies done by Dao (Dao, 1977), Elmasry (Elmasry, 1975) can be shown as examples. Current mode logic circuits provide the facility of adding signals easily by wiring. For radix bigger than 3, charge based circuits and current based circuits are prefered. However, especially for radix 3, voltage mode circuits are more important (Etiemble and Israel, 1974).

Multi-value logic circuits that are realized by use of different technologies such as MOS, I^2L bipolar and CCD technology have been presented in (Kameyama et al., 1988), (Etiemble and Israel, 1988), (Butler and Kerkhoff, 1988).

2.3.1 MOS Technology

There are over 100 million elements over a MOS memory. Because of the increase in the number of elements per unit area, a lot of problems have been occured. Most important ones of those problems are, cooling, managing and routing interconnections. As the number of elements increase, number of interconnections increase and those interconnections take place more than the elements itself (Etiemble and Israel, 1988).

Popularity of production of multi-value logic circuits by using CMOS technology has been increased because of use of processes that are also used at production stage of binary systems (Jain et al., 1993). CMOS multi-value logic circuits, are divided into two namely voltage-mode circuits and current-mode circuits. Both type have advantageous and disadvantageous sides over each other (Arnold, 1988). Most important advantage of voltage-mode circuits is that, more than one circuit elements are controlled by a single voltage source. This parallelism feature contributes to overcome of data redundancy. If more than one circuit elements, need the same data (signal) at the same time, by a single source, this signal can be easily transmitted to those elements by parallelism. Hence, voltage-mode circuits are mostly used for data transmission and routing operations. Important problems faced with voltage-mode circuits are physical area, addition of signals and realization of 2's complement operation.

Multi-value logic circuits that are realized by using current-mode CMOS technology, logical levels are confirmed by quantization of current signal. As an example, for logical state 0 $I_{input} = 0$ mA, logical state 1 $I_{input} = 1$ mA , logical state 2 $I_{input} = 2$ mA etc (Jain et al., 1993).

With current-mode circuits in order to implement the communication between circuit elements more data lines needed according to voltage-mode circuits. A data line must be assigned to a single point. If other circuit elements enter to this line, magnitude of this signal will be degrade to a obscure value. Besides that, addition of two signals can be realized easily by simply wiring. Also, 2'complement operation can be simply performed by converting the flow direction of current to the opposite side (Arnold, 1988).

Futhermore, current-mode circuits have the easiness of addition operation by simply wiring. However, power consumption is a serious handicap. In contrast, power consumption is very low at voltage-mode circuits. Because, there is no current flow in static state. (Syuto et al., 2000)

Current mode multi-value circuits are conformable for realizing threshold functions of binary logic. But, there is a remarkably important issue is that the result of analog addition of currents is a non level restoring value. That is to say, it falls between any two values which we found by scaling the current signal and it does not map any of the valid states. Hence, any signal generated because of noise will affect the result, and this is an undesirable situation.

DC characteristics of non level restoring and level restoring circuits are shown at Figures 2.3 and 2.4.

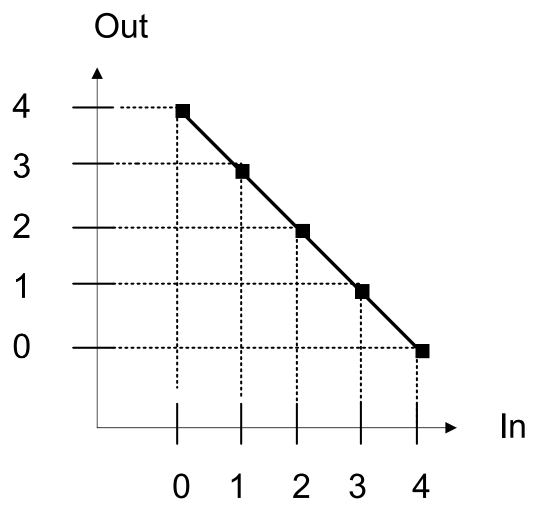


Figure 2.3 DC characteristic of non level restoring circuit

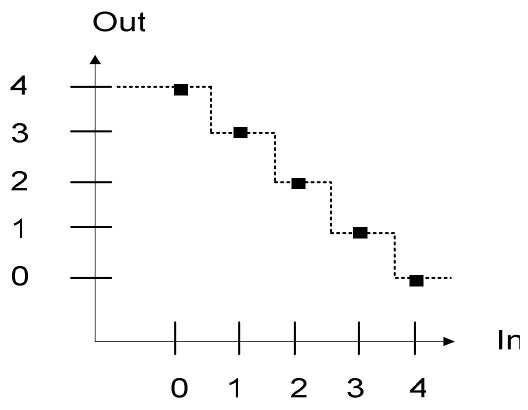


Figure 2.4 DC characteristic of level restoring circuit (Etiemble and Israel, 1988)

Optimal architecture is the use of both arhitectures in corporation. As an example, using voltage-mode structure for only routing and data transmission; and using current-mode structure at internal operations such as multplication (Arnold, 1988).

Circuits in multi-value logic which are established by using CMOS technology, can use current mode CMOS technology, voltage mode CMOS technology or both.

2.3.1.1 Basic Circuit Elements for Current Mode CMOS

There are 5 fundamental circuit elements in current mode CMOS. Those are: sum, constant, current mirror, threshold and switch.

2.3.1.1.1 Sum

Logic operation $y = x_1 + x_2 + ... + x_n$

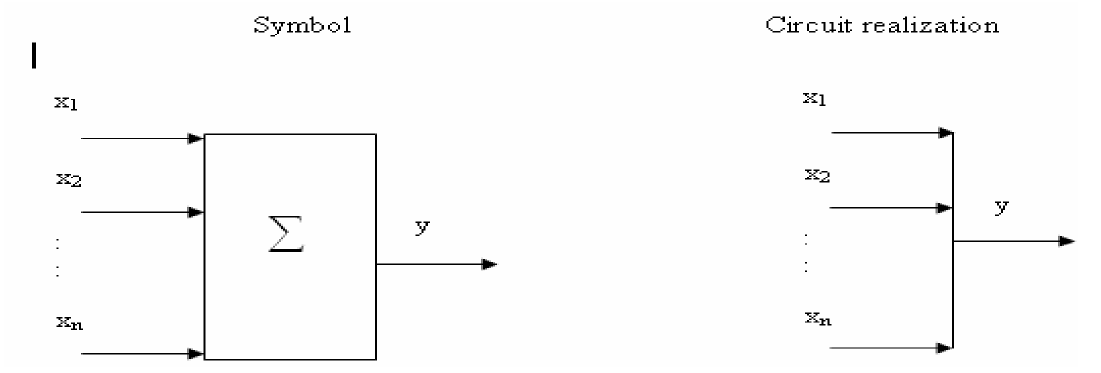


Figure 2.5 Sum Element

2.3.1.1.2 Constant

Logic operation $y = K$,

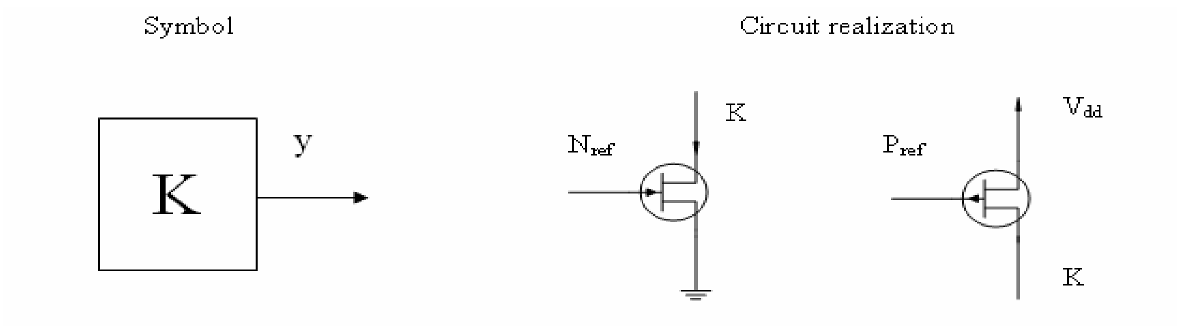


Figure 2.6 Constant element

Constant circuit element, is accomplished by using N-type or P-type transistors. If an output current K times the input current is wanted to be generated, then W/L ratio of the transistor must be equal to K (Jain et al., 1993).

2.3.1.1.3 Current mirror

Logic operation $I_{oi} = K_i \; I_{in}$ for $i = \{1, 2, ..., n\}$ K_i : constant

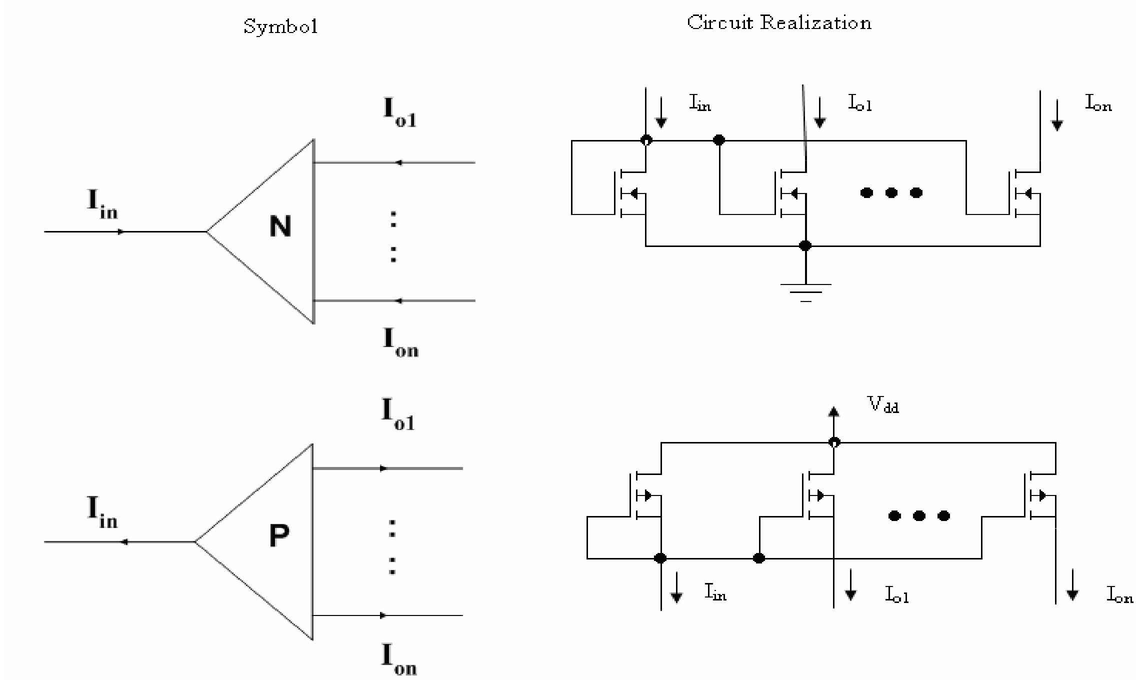


Figure 2.7 Current Mirror

Those circuits are used for copying the input current. Besides that, by arranging constant K_i, there can be generated output currents $K_i * I_{in}$ at output. There are two types of current mirrors. One of them consists of N-type transistors and the second one consists of P-type transistors. The transistor which I_{in} current flows over is named as the input transistor and the others are output transistors. W_i/L_i ratios of output transistors define the ratio of I_{oi} / I_{in}. Nevertheless, by using current mirror, direction of a current can be inverted. This method is useful for realizing subtraction operation (Jain et al., 1993).

2.3.1.1.4 Threshold

This element can be thought like a converter from current to voltage. That is to say, it is circuit element with a current input and voltage output. There are two types of threshold elements, N-type and P-type. Working principle of N-type threshold is explained below:
If the input current is smaller than the threshold current value, output voltage value becomes binary low (0V), else if input current is bigger than or equal to threshold current value, output voltage value becomes binary high (5V).

For N-type, if ($I_{th} \leq I_{in}$) then V_o = binary high else V_o = binary low.

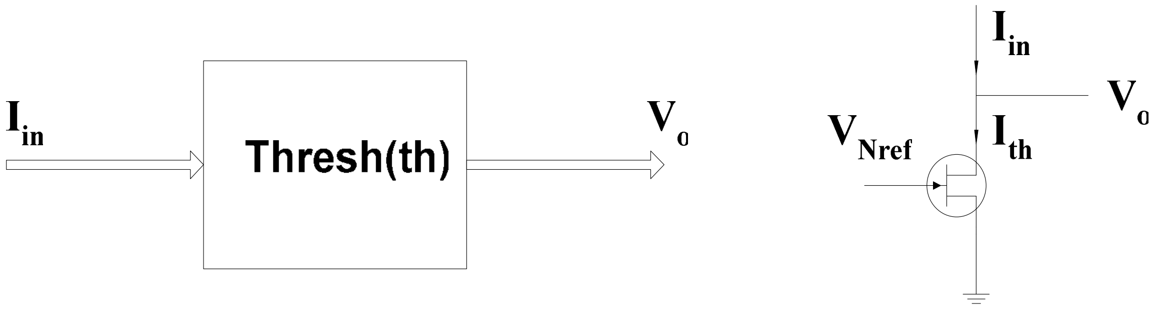


Figure 2.8 Threshold element by NMOS

Working principle of other type, P-type threshold is explained below:
If the input current is smaller than the threshold current value, output voltage value becomes binary high (5V), else if input current is bigger than or equal to threshold current value, output voltage value becomes binary low (0V).

For P-type, if ($I_{th} \leq I_{in}$) then V_o = binary low else V_o = binary high.

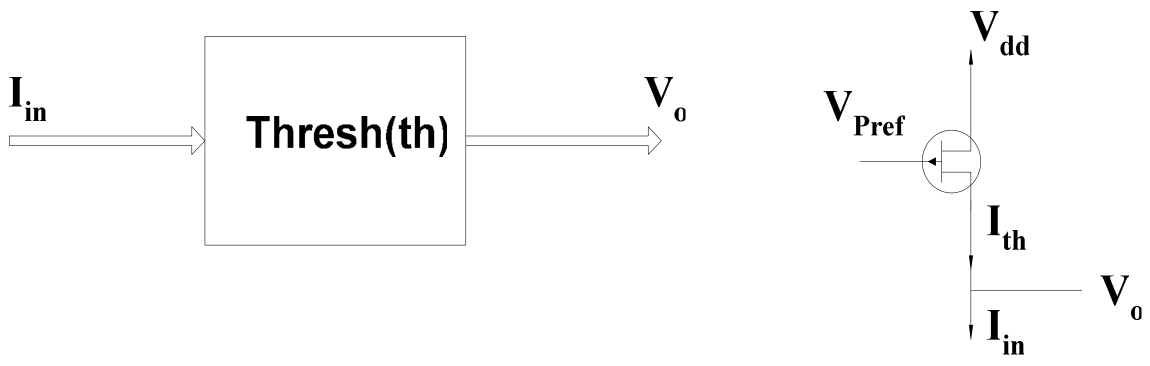


Figure 2.9 Threshold element by PMOS

I_{th} current here expresses the maximum current value that can flow over the transistor and its magnitude is defined by arranging (W / L) * I_{in} . Threshold element is used for realizing literal and cycle elements (Jain et al., 1993).

2.3.1.1.5 switch

This element, can be thought like a converter from voltage to current. That is to say, it is circuit element with a voltage input and current output. There are two types of switch elements, N-type and P-type. Working principle of N-type switch is explained below:

If a voltage level which can activate the transistor can be applied to the gate of the transistor, then transistor passes to conductive (ON) state and it permits the input current to flow to output. If voltage level applied to gate is not enough to activate transistor, then transistor passes to cut off (OFF) state and it will not permit input current to flow to output.

For N-type, if (V_{in} = binary high) then $I_o = I_{in}$ else $I_o = 0$.

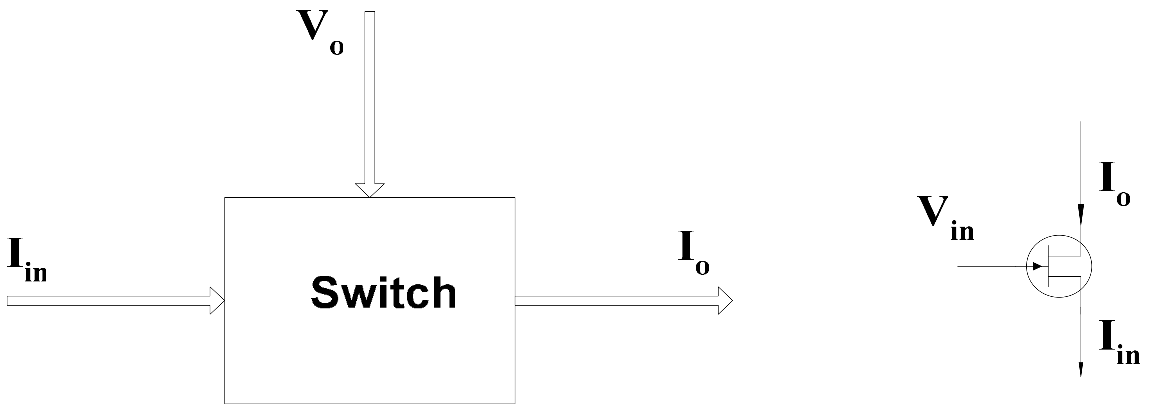


Figure 2.10 Switch element by NMOS

Working principle of P-type switch is explained below:
If a voltage level of binary low which is applied to the gate of the transistor, then transistor passes to conductive (ON) state and it permits the input current to flow to output. If voltage level applied to gate is binary high, then transistor passes to cut off (OFF) state and it will not permit input current to flow to output (Jain et al., 1993).

For P-type, if (V_{in} = binary low) then $I_o = I_{in}$ else $I_o = 0$.

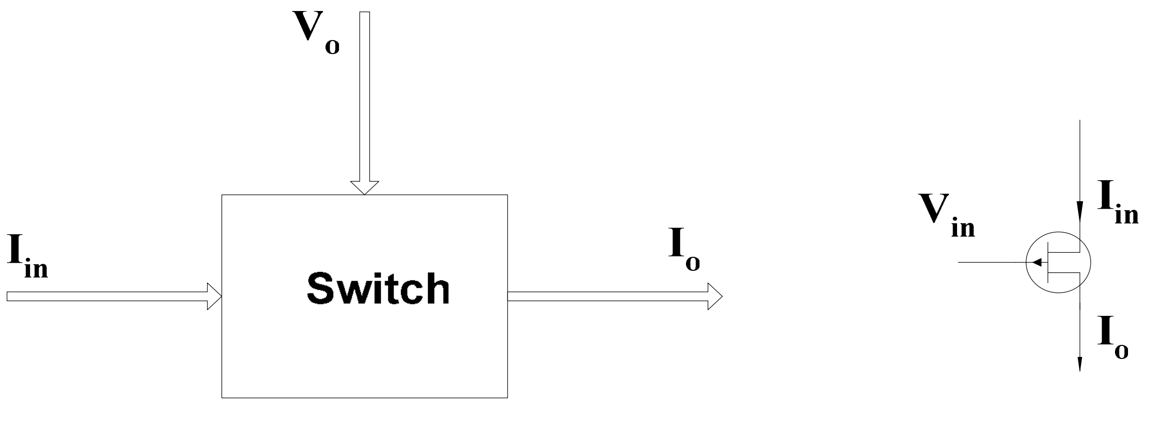


Figure 2.11 Switch element by PMOS

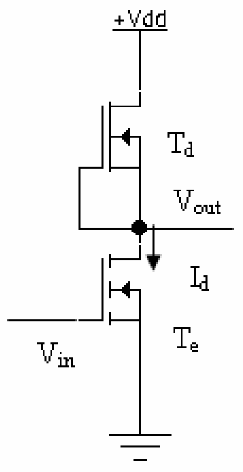
Previously, MOS technology employed only PMOS transistors. Later on, NMOS transistors have started to be used. Today, both NMOS and PMOS transistors are used.

CMOS technology superseded NMOS technology because of low power consumption. NMOS technology basically use N-type enhancement mode and N-type depletion mode transistors. CMOS technology use N-type enhancement and P-type enhancement transistors.

2.3.1.2 Sample Circuit Elements For Voltage Mode CMOS

In voltage mode CMOS, there are circuit elements such as inverter, NAND, NOR, etc.

2.3.1.2.1 NMOS inverter



Fundamental formulas of a MOS transistor are shown below:

If $V_{GS} - V_T < 0$:

$$I_D = 0$$

If $0 < V_{GS} - V_T < V_{DS}$:

$$I_D = (0.5*\mu_n*C_{ox}) * (W/L)* (V_{GS}-V_T)^2$$

If $V_{GS} - V_T > V_{DS}$:

$$I_D = (0.5*\mu_n*C_{ox}) * (W/L)* (V_{DS})[2*(V_{GS}-V_T)-V_{DS}]$$

Figure 2.12 NMOS inverter

For realizing desired logic states, parameters that must be adjusted are threshold voltage values or W/L ratio of transistors (Etiemble and Israel, 1988).

2.3.1.2.2 Four-valued Encoder

Below is shown a circuit realization of a 4-valued encoder circuit. X, Y, Z values are the binary logic values that map to the input voltage values of the circuit. M is the multi-value logic state that maps to the output voltage value of the circuit.

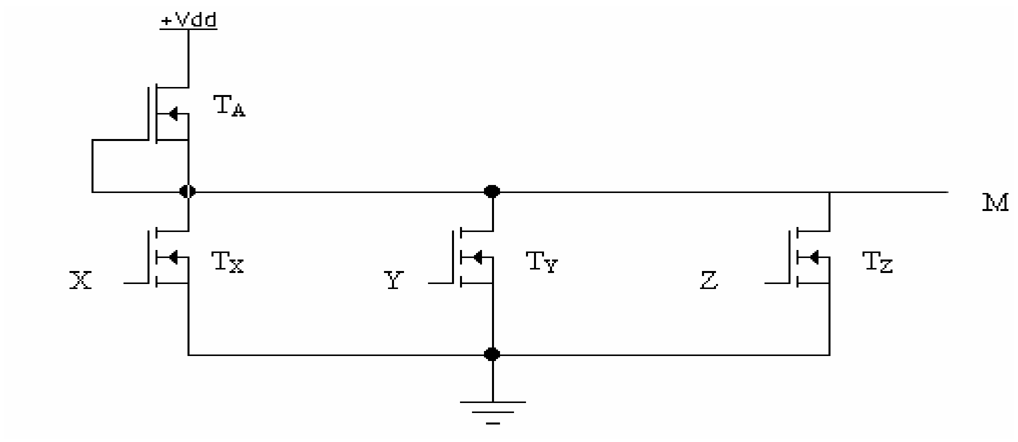
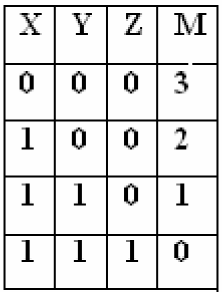


Figure 2.13 Four-valued NMOS Encoder

Table 2.1 Truth table of Four-valued NMOS Encoder



X	Y	Z	M
0	0	0	3
1	0	0	2
1	1	0	1
1	1	1	0

Let's say, input voltage levels map to logic levels X=0, Y=1 and Z=0, are applied to transistors T_X, T_Y, T_Z. In this situation, T_X is at cut-off state, T_Y is at saturation state, and T_Z is at saturation state. Hence, output voltage level is determined by the ratio $(W_A/L_A) / (W_Y/L_Y)$. If we give another example, let's consider input voltage levels map to logic levels X=1, Y=0 and Z=1, are applied to transistors T_X, T_Y, T_Z. In this situation, T_X is at saturation state, T_Y is at cut-off state, and T_Z is at saturation state. Hence, output voltage level is determined by the ratio $(W_A/L_A) / [(W_X/L_X) + (W_Z/L_Z)]$. (Etiemble and Israel, 1988).

2.3.1.3 Sample Realization Of Some Operators By MOS Transistors

2.3.1.3.1 Upper Threshold

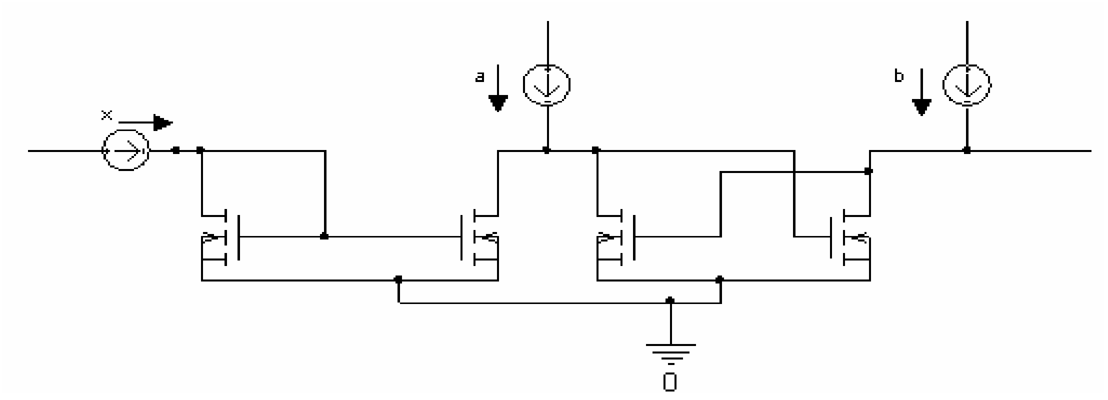


Figure 2.14 Upper Threshold Circuit (Morgul and Temel, 2005)

$$x\,|_{a}^{b} = \begin{cases} b & \text{if } a \le x \\ 0 & \text{else} \end{cases}$$

2.3.1.3.2 Lower Threshold

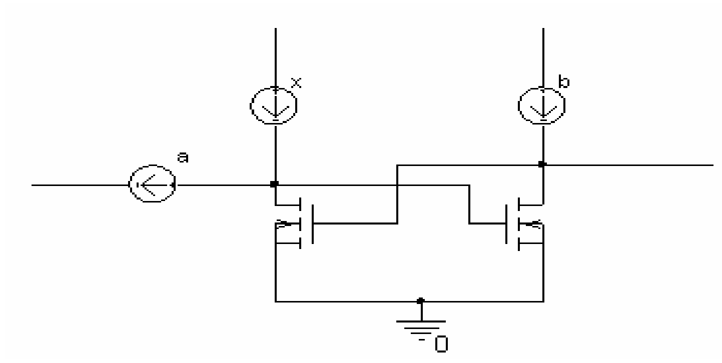


Figure 2.15 Lower Threshold Circuit (Temel and Morgul, 2002)

$${}_{a}^{b}|\,x = \begin{cases} b & \text{if } x \le a \\ 0 & \text{else} \end{cases}$$

2.3.1.3.3 Literal

Literal element can be realized by using lower threshold model shown at figure 2.16.

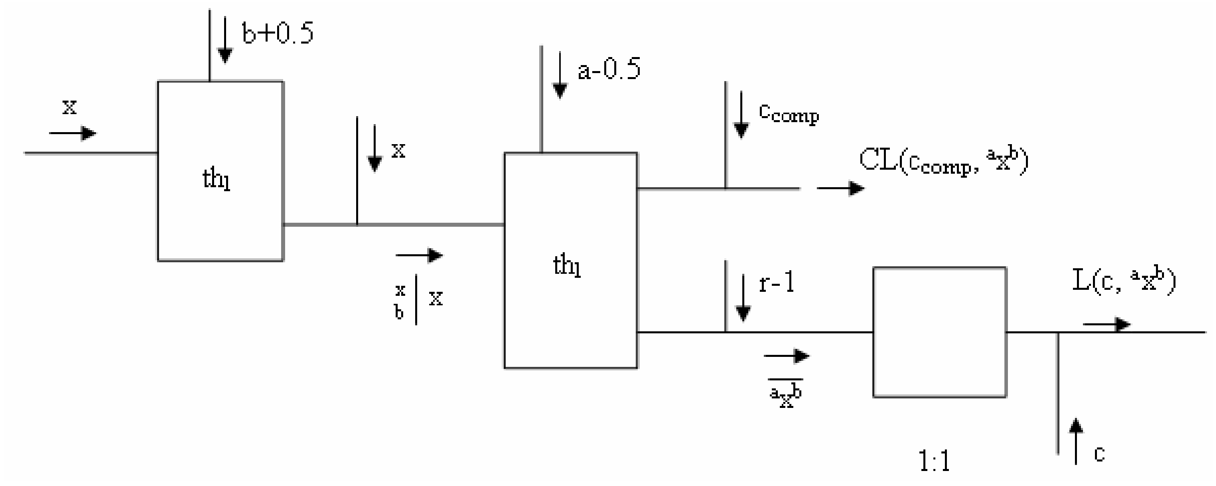


Figure 2.16 Literal 1 (Temel and Morgul, 2005)

Another way of realizing literal element can be accomplished by using lower threshold model shown at figure 2.17 and upper threshold model in corporation.

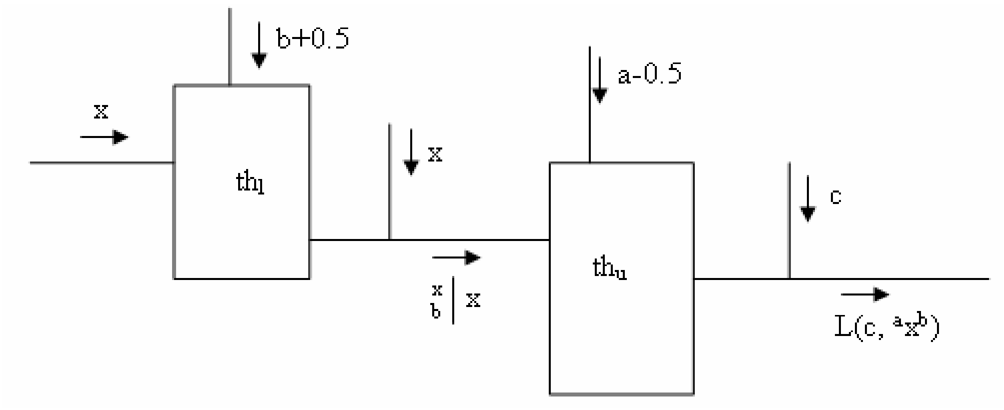


Figure 2.17 Literal 2

Literal element realized by MOS transistors are shown below:

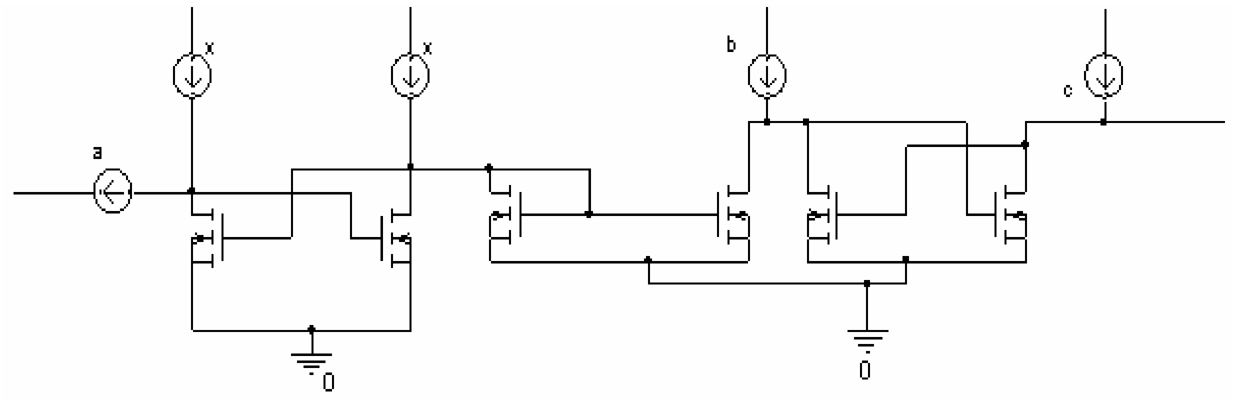


Figure 2.18 Literal Circuit

$$L(c, {}^{a}x^{b}) = \begin{cases} c & \text{if } b \leq x \leq a \\ 0 & \text{else} \end{cases}$$

2.3.2 Bipolar Technology

Another important technology used in multi-value logic is bipolar technology. While realizing current mode multi-value circuits, especially simple ones, bipolar technology is very advantageous. Some of those basic circuits implemented by using bipolar technology are shown below:

2.3.2.1 Current Mirror

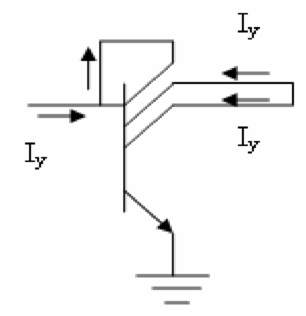


Figure 2.19 Current Mirror Circuit

2.3.2.2 Sum

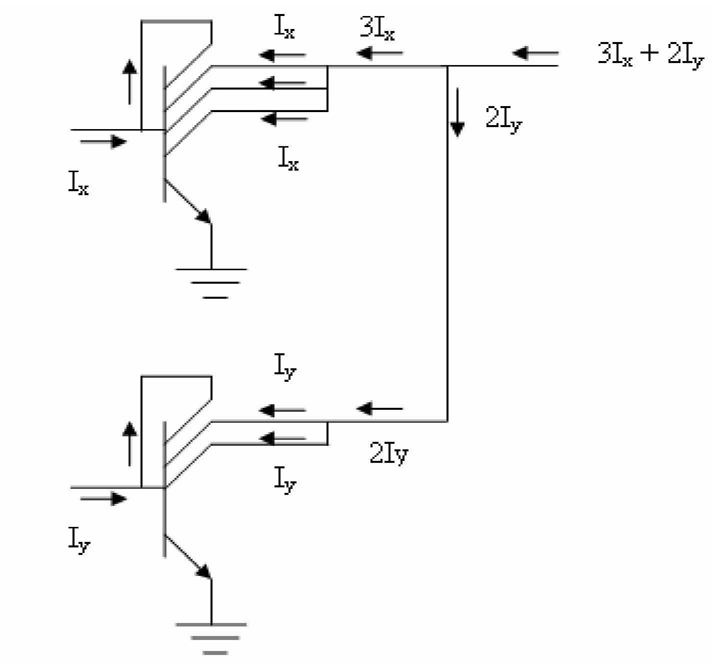


Figure 2.20 Linear Summation Circuit

2.3.2.3 Threshold

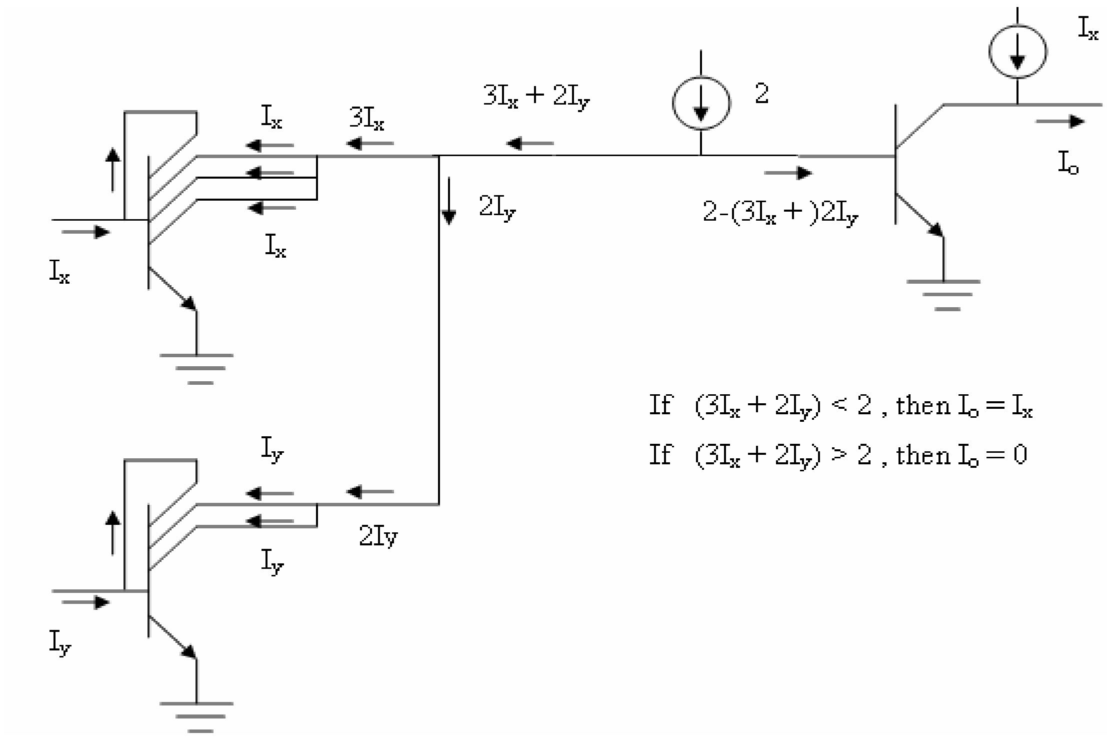


Figure 2.21 Threshold Circuit (Etiemble and Israel, 1988)

Using bipolar technology during realizing multi-value current circuits is more advantageous than using MOS technology. Because, bipolar transistors are current controlled devices , in contrast, MOS transistors are voltage controlled (Etiemble and Israel,1988).

CHAPTER 3

FINITE FIELDS

A field is a set of elements which has the following properties; closure, associativity, commutativity, distributivity and presence of additive and multiplicative inverses. If a field contains finite number of elements it is a finite field. Finite fields are very significant especially for cryptography and are applied to many areas in cryptography such as cryptographic algorithms like AES and Elliptic Curve crptography that are constructed over finite field arithmetic (Stallings, 2006).

Definition 1: p is a prime and n is a positive integer,
order of a finite field is the number of elements in the field which is shown as p^n .

Definition 2: fields with the order p can be defined by arithmetic over modulus p (mod p).

Definition 3: modular arithmetic, is the process of mapping an integer to an integer in the set {0, 1, ..., n-1} for an integer n.

Definition 4: fields with the order p^n and n>1, can be defined by polynomial arithmetic.

For an element set to be a field, it must possess some specifications.
Let consider set $X = \{x_1, x_2, ..., x_n\}$, $i, j, k, l, m < n \in Z^+$

Specification 1: If an addition operation is applied over two elements of the set X, and if the result is one of the elements of the same set X, then this set X possesses the closure property over addition.
If $x_k + x_l = x_m \in X \rightarrow X$ is closure under addition.

Specification 2: If $(x_k + x_l) + x_m = x_k + (x_m + x_n)$ $\rightarrow$ X is associative under addition.

Specification 3: If for all elements of set X,
$x_i + y = x_i$ and $y \in X$ $\rightarrow$ y is the identity element over addition and set X has the property of additive identity.

Specification 4: If for each x_i in the set X,
$x_i + y_i = 0$ and $y_i \in X$ $\rightarrow$ $y_i = -x_i$ and set X has the property of presence of additive inverse.

Specification 5: If for every element x_i in the set X,
$x_i + y_i = y_i + x_i$ $\rightarrow$ set X has the property of additive comutativity.

Specification 6: If a multiplication operation is applied over two elements of set X, and if the result is one of the elements of the same set X, then this set X Possesses the closure property over multiplication.

If $x_k * + x_l = x_m \in X$ $\rightarrow$ X is closure under multiplication.

Specification 7: If $(x_k * x_l) * x_m = x_k * (x_l * x_m)$ $\rightarrow$ X is associative under multiplication.

Specification 8: If $x_k * (x_l + x_m) = x_k * x_l + x_k * x_m$ $\rightarrow$ X has the distributive property.

Specification 9: If for every element x_i in the set X,
$x_i * y_i = y_i * x_i$ $\rightarrow$ set X has the property of multiplicative comutativity.

Specification 10: If for all elements of set X,
$x_i * y = x_i$ and $y \in X \rightarrow$ y is the identity element over multiplication and set X has the property of multiplicative identity (Stallings, 2006).

Specification 11: If for every element x_i in the set X,
$xi * yi = 0$ and $xi = 0$ or $yi = 0 \rightarrow$ there is no element in set X that can be divided by 0.

Specification 12: If for each x_i in the set X,
$x_i * y_i = 1$ and $y_i \in X \rightarrow y_i = (x_i)^{-1}$ and set X has the property of presence of multiplicative inverse.

After all those specifications, we can show rational numbers and real numbers as an example for fields. However, set of all integers is not a field because this set does not have the property of presence of multiplicative inverse (Stallings, 2006).

Ex: $3 * x = 1 \rightarrow x = 3^{-1} = 1/3$ is not an element of integers.

3.1 FINITE FIELDS OF THE FORM GF(p)

Finite fields are mentioned as Galois Fields to honor Evariste Galois. GF(p^n) conveys the finite field that has the order p^n which shows the number of elements in this set. When n=1, structure differs from the structure when n > 1.

Below, there are shown basic operations over GF(2):

Table 3.1 Addition Operation for GF(2)

+	0	1
0	0	1
1	1	0

Table 3.2 Multiplication Operation for GF(2)

*	0	1
0	0	0
1	0	1

Table 3.3 Inverse Operation for GF(2)

x	-(x) Additive inverse	$(x)^{-1}$ Multiplicative inverse
0	0	- (Not defined)
1	1	0

As it can be realized from the tables above, addition operation maps to logical XOR operation and multiplication operation maps to logical AND operation because those operations are binary. This means, radix 2 (Stallings, 2006).

As it is mentioned at definition 2, fields with the order p can be defined by using arithmetic modulus p. Tables represented below show operations of GF(7). Those tables are generated over mod 7.

Table 3.4 Addition Operation for GF(7)

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Table 3.5 Multiplication Operation for GF(7)

*	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Table 3.6 Inverse Operation for GF(2)

x	-(x) Additive inverse	$(x)^{-1}$ Multiplicative inverse
0	0	- (Not defined)
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

3.2 POLYNOMIAL ARITHMETIC

Before carrying on finite fields of type GF(p^n), it will be better to remember some basic operations of GF($p=2$).

3.2.1 Addition

Let $f(x) = a_n*x^n + a_{n-1}*x^{n-1} + \dots + a_0*x^0$

$g(x) = b_n*x^n + b_{n-1}*x^{n-1} + \dots + b_0*x^0$

$f(x) + g(x) = (a_n + b_n) * x^n + (a_{n-1} + b_{n-1}) * x^{n-1} + \dots + (a_0 + b_0) * x^0$

Ex: $f(x) = x^2 + 1$, $g(x) = x + 1$

$$f(x) + g(x) = x^2 + x + \underbrace{\boxed{(1+1)}}_{\downarrow \; 2 \,=\, 0 \pmod 2}$$

$f(x) + g(x) = x^2 + x$

3.2.2 Multiplication

Let $f(x) = a_n*x^n + a_{n-1}*x^{n-1} + \dots + a_0*x^0$

$g(x) = b_n*x^n + b_{n-1}*x^{n-1} + \dots + b_0*x^0$

$$\begin{aligned} f(x) * g(x) = {} & (a_n * b_n) * x^{2n} + (a_n * b_{n-1}) * x^{2n-1} + \dots + (a_n * b_0) * x^n \\ & + (a_{n-1} * b_n) * x^{2n-1} + (a_{n-1} * b_{n-1}) * x^{2n-2} + \dots + (a_{n-1} * b_0) * x^{n-1} \\ & + \dots\dots \\ & + (a_0 * b_n) * x^n + (a_0 * b_{n-1}) * x^{n-1} + \dots + (a_0 * b_0) * x^0 \end{aligned}$$

Ex: $f(x) = x^2 + 1$, $g(x) = x^2 + 1$

$$f(x) * g(x) = x^4 + \underbrace{\boxed{2}}_{\downarrow \; 2 \,=\, 0 \pmod 2}x^2 + 1 \quad \Rightarrow \quad f(x) * g(x) = x^4 + 1$$

3.2.3 Division

$$\frac{f(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)}$$

$f(x) = q(x) * g(x) + r(x)$

Ex: $f(x) = x^3 + x + 1$ $g(x) = x^2 + x$
$f(x) = (x^2 + x)(x + 1) + 1$ (Stallings, 2006)

3.3 IRREDUCIBLE POLYNOMIALS

If a polynomial f(x) can not be factorized to polynomials defined in the same field and with lower degree than it, then this polynomial is called an irreducible polynomial.

Ex: $f(x) = x^4 + 1$ defined in GF(2), is a reducible polynomial because, we can factorize f(x) as $f(x) = x^4 + 1 = (x + 1)(x^3 + x^2 + x + 1)$.

Ex: $f(x) = x^3 + x + 1$ is an irreducible polynomial because it can not be factorized to polynomials defined in the same field (Stallings, 2006).

3.4 FINITE FIELDS OF THE FORM GF(p^n)

As it is mentioned before, order of a finite field must be p^n . Here, p is a prime, n is a positive integer. However, when order is p^n, arithmetic operations made at modulus p^n do not form a field. Because of this reason, instead of doing arithmetic at modulus p^n, for GF(p^n), polynomial arithmetic will be done at modulus irreducible polynomial.

Cryptographic algorithms, generally perform arithmetic operations on integers. If there is a division operation performed on integers, then an arithmetic has to be applied which results a field (Stallings, 2006).

Let’s consider a cryptographic algorithm that performs 8-bit encryption that performs division operation on integers. We can represent numbers between 0-255 by 8 bits. But, 256 is not a prime and result of arithmetic at modulus 256 is not a field. Biggest prime smaller than 256 is 251. Hence, result of arithmetic at modulus 251 is a field but this time characters that can be represented with 8-bits and have integer equivalence between 251 and 256 will not be able to expressed at this arithmetic. Thus, for GF (2^8), at modulus irreducible polynomial, by using polynomial arithmetic that forms a field, arithmetic operations have to be performed. (Stallings, 2006)

3.4.1 Constructing Finite Fields

3.4.1.1 GF(3^2)

For GF(9) = GF(3^2) , degrees of polynomials defined in the field can be at most 1 and coefficients can be at most 2. Hence, since we chose GF(3^2) as an example, polynomials defined in the field are:

0	1	2
x	$x + 1$	$x + 2$
$2x$	$2x + 1$	$2x + 2$

As it is mentioned before, there are $p^n = 3^2 = 9$ elements in the field. (order = 9)

To generate a p^n type galois field, firstly a convenient irreducible polynomial has to be found and field must be derived from this irreducible polynomial. For large fields, it can be difficult to find an irreducible polynomial and there are some procedures for this purpose. However, for small fields, there is no need for those procedures. Brute force method can be applied (Department of Mathematical Sciences University of Colorado Denver).

Let’s consider GF(3^2):

Irreducible Polynomial Candidates are:

x^2	$x^2 + 1$	$x^2 + 2$
$x^2 + x$	$x^2 + x + 1$	$x^2 + x + 2$
$x^2 + 2x$	$x^2 + 2x + 1$	$x^2 + 2x + 2$

In order to find irreducible polynomials among candidates above, a few methods can be applied. First one is trying to factorize each of them by x. Hence, x^2, $x^2 + x$, $x^2 + 2x$ can be eliminated because those ones can be factorized by x. After that, since p = 3, values of x can be 0, 1, 2. As an example, if those values are put instead of x at polynomial $x^2 + 2$:

$0^2 + 2 = 2$ $1^2 + 2 = 0$ $2^2 + 2 = 0$

Results show, $x^2 + 2$ can be factorized as $(x + 1)(x + 2)$, so this polynomial has to be eliminated too. Last method is to eliminate polynomials generated from root squares like

$(x + 1)(x + 1)$, $(x + 2)(x + 2)$. By this method, $x^2 + 2x +1$ and $x^2 + x +1$ are eliminated. At the end, three polynomials will stay not eliminated which are irreducible ones. Hence, all elements of GF(9) will be generated by using one of those irreducible polynomials. Since multiplicative group of a field is cyclic, a generator of this cyclic group has to be defined first.

Primitive element is one of the roots of irreducible polynomial. But it should be paid attention that not every root of irreducible polynomial is a primitive element.

Let's consider $x^2 +1$,

$x^2 = -1 \Rightarrow -1 = 2 \bmod 3$

- ⇨ $x^2 = 2$
- ⇨ $x^1 = x$
- ⇨ $x^2 = 2$
- ⇨ $x^3 = x^2 * x = 2x$
- ⇨ $x^4 = x^2 * x^2 = 2 * 2 = 4 \bmod 3 = 1$

As it is seen above, order = 4 and not all of the elements in the field could be generated. Hence, irreducible polynomial is not convenient.

Let's consider $x^2 + x + 2$,

$x^2 = - x - 2 \Rightarrow -1 = 2 \bmod 3$, $-2 = 1 \bmod 3$

- ⇨ $x^2 = 2x + 1$

- ⇨ $x^1 = x$
- ⇨ $x^2 = 2x + 1$
- ⇨ $x^3 = x^2 * x = (2x + 1) * x = 2x^2 + x = 2 * (2x + 1) + x = 4x + 2 + x = 2x + 2$
- ⇨ $x^4 = (2x + 1) * (2x + 1) = 4x^2 + 4x + 1 = x^2 + x +1 = 3x + 2 = 2$
- ⇨ $x^5 = x^4 * x = 2x$
- ⇨ $x^6 = x^4 * x^2 = 2 * (2x + 1) = 4x + 2 = x + 2$
- ⇨ $x^7 = x^4 * x^3 = 2 * (2x + 2) = 4x + 4 = x + 1$
- ⇨ $x^8 = x^4 * x^4 = 2 * 2 = 4 = 1$

As it is seen, all of the polynomials in the field are generated and the order = 9, so this irreducible polynomial is convenient. (Department of Mathematical Sciences University of Colorado Denver)

3.4.1.2 GF(2^3)

For GF(8) = GF(2^3) , degrees of polynomials defined in the field can be at most 2 and coefficients can be at most 1. Hence, GF(2^3) is chosen as an example, polynomials defined in the field are:

0	1	x	$x +1$
x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x +1$

As it is mentioned before, there are $p^n = 2^3 = 8$ elements in the field. (order = 8)

Let's consider GF(2^3):

Irreducible Polynomial Candidates are:

$x^3 + 1$ $\quad$ $x^3 + x + 1$ $\quad$ $x^3 + x^2 + 1$ $\quad$ $x^3 + x^2 + x + 1$

Since p = 2, values of x can be 0, 1. If those values are put instead of x at polynomial $x^3 + 1$:

$0^3 + 1 = 1$ $\quad$ $1^3 + 1 = 0$

Results show, this polynomial has to be eliminated too. Two polynomials will stay not eliminated which are irreducible ones. Those are $x^3 + x + 1$, $x^3 + x^2 + 1$. Hence, all elements of GF(8) will be generated by using one of those irreducible polynomials.

Let's consider $x^3 + x + 1$,

$x^3 = - x -1$ => $-1 = 1 \bmod 2$ => $x^3 = x + 1$

- ⇨ $x^1 = x$
- ⇨ $x^2 = x^2$
- ⇨ $x^3 = x + 1$
- ⇨ $x^4 = x^3 * x = (x + 1) * x = x^2 + x$
- ⇨ $x^5 = x^3 * x^2 = (x + 1) * (x^2) = x^3 + x^2 = x^2 + x + 1$
- ⇨ $x^6 = x^3 * x^3 = (x + 1) * (x + 1) = x^2 + 2x + 1 = x^2 + 1$
- ⇨ $x^7 = x^6 * x = (x^2 + 1) * (x) = x^3 + x = x + x + 1 = 2x + 1 = 1$

As it is seen above, order = 8 and all of the elements in the field can be generated. Hence, irreducible polynomial is convenient.

Let's consider $x^3 + x^2 + 1$

$x^3 = - x - 1$ => $-1 = 1 \bmod 2$ => $x^3 = x^2 + 1$

- ⇨ $x^1 = x$
- ⇨ $x^2 = x^2$
- ⇨ $x^3 = x^2 + 1$
- ⇨ $x^4 = x^3 * x = (x^2 + 1) * (x) = x^3 + x = x^2 + x + 1$
- ⇨ $x^5 = x^4 * x = (x^2 + x + 1) *(x) = x^3 + x^2 + x = x^2 + 1 + x^2 + x = x + 1$
- ⇨ $x^6 = x^5 * x = (x + 1) * (x) = x^2 + x$
- ⇨ $x^7 = x^6 * x = (x^2 + x) * (x) = x^3 + x^2 = x^2 + 1 + x^2 = 1$

As you can see, all of the polynomials in the field are generated and the order = 8, so this irreducible polynomial is convenient (Department of Mathematical Sciences University of Colorado Denver).

3.4.1.3 Polynomial Arithmetic at GF(8)

Irreducible polynomial = $x^8 + x^4 + x^3 + x + 1$

$$x^8 = - x^4 - x^3 - x - 1$$

$$x^8 = x^4 + x^3 + x + 1$$

3.4.1.3.1 Addition

$f(x) = x^7 + x^5 + x^3 + x + 1$

$g(x) = x^5 + x^3 + x^2$

$f(x) + g(x) = x^7 + 2x^5 + 2x^3 + x^2 + x + 1 = x^7 + x^2 + x + 1$

Since we work at GF(2^3) this means binary, we can show operation above as XOR with bit representation:

$$\begin{array}{lll} f(x) = x^7 + x^5 + x^3 + x + 1 & = & 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1 \\ g(x) = x^5 + x^3 + x^2 & = & 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0 \\ f(x) + g(x) & = & \underline{\oplus\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1} \quad = \quad x^7 + x^2 + x + 1 \end{array}$$

3.4.1.3.2 Multiplication

$f(x) = x^4 + x + 1$

$g(x) = x^7 + x^2 + 1$

$f(x) * g(x) = (x^4 + x + 1) * (x^7 + x^2 + 1) = x^{11} + x^6 + x^4 + x^8 + x^3 + x + x^7 + x^2 + 1$

$$x^{11} = x^8 * x^3 = (x^4 + x^3 + x + 1) * x^3 = x^7 + x^6 + x^4 + x^3$$

$$x^8 = x^4 + x^3 + x + 1$$

$f(x) * g(x) = x^7 + x^6 + x^4 + x^3 + x^6 + x^4 + x^4 + x^3 + x + 1 + x^3 + x + x^7 + x^2 + 1$

$= x^4 + x^3 + x^2$

3.5 MULTI VALUE LOGIC APPLICATION OF BASIC GALOIS FIELDS OPERATIONS FOR GF(3)

3.5.1 Addition

Table 3.7 Addition For GF(3)

+ x y	**0**	**1**	**2**
0	0	1	2
1	1	2	0
2	2	0	1

According to Table 3.7 solution of (x + y) can be expressed by means of cycle operator or by literal operator.

Realization with cycle operator:

$$x + y = x^{y}_{\rightarrow}$$

Realization with literal operator:

$$x + y = x^{\{1,2\}}y^{\{0\}} + x^{\{2\}}y^{\{0\}} + x^{\{0,1\}}y^{\{1\}} + x^{\{1\}}y^{\{1\}} + x^{\{2\}}y^{\{2\}} + x^{\{0\}}y^{\{2\}}$$

3.5.1.1 Circuit Realization of Addition With Literal Operator

$$x + y = x^{\{1,2\}}y^{\{0\}} + x^{\{2\}}y^{\{0\}} + x^{\{0,1\}}y^{\{1\}} + x^{\{1\}}y^{\{1\}} + x^{\{2\}}y^{\{2\}} + x^{\{0\}}y^{\{2\}}$$

Let x = 2 , y = 1,

$x^{\{1,2\}}y^{\{0\}}$ = Io1 = 1 • 0 = 0

$x^{\{2\}}y^{\{0\}}$ = Io2 = 1 • 0 = 0

$x^{\{0,1\}}y^{\{1\}}$ = Io3 = 0 • 1 = 0

$x^{\{1\}}y^{\{1\}}$ = Io4 = 0 • 1 = 0

$x^{\{2\}}y^{\{2\}} = Io5 = 1 \bullet 0 = 0$

$x^{\{0\}}y^{\{2\}} = Io6 = 0 \bullet 0 = 0$

$x + y = 2 + 1 = 3 \bmod 3 = 0$, $Io1 + I\,o2 + Io3 + I\,o4 + Io5 + I\,o6 = 0$

For x = 2 , y = 1,

$x^{\{1,2\}}y^{\{0\}} = Io1 = 1.854\ fA = 0$

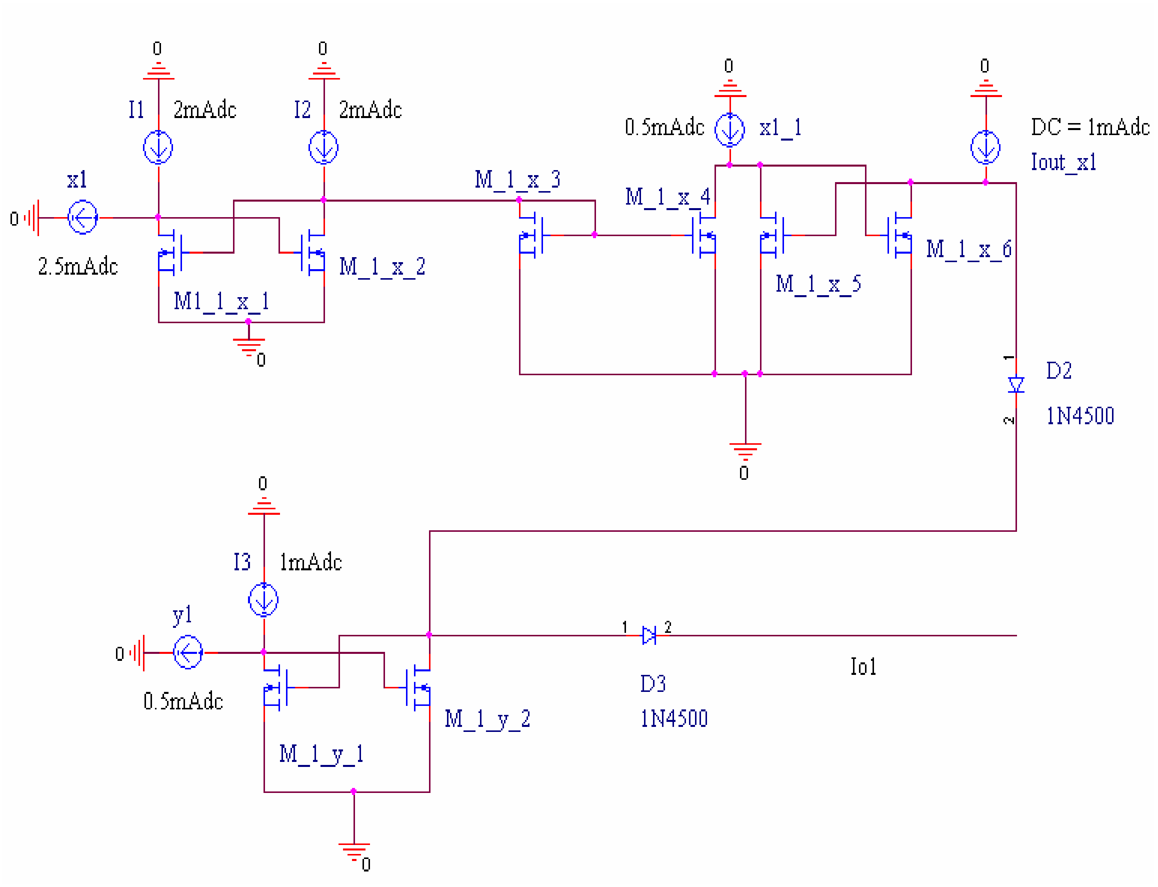


Figure 3.1 MOS realization of $x^{\{1,2\}}y^{\{0\}}$

Transistor parameters :

Table 3.8 Transistor parameters for circuit $x^{\{1,2\}}y^{\{0\}}$

Transistor name	**Model**	**W**	**L**
M_1_x_1	MbreakN	0.36u	1.08u
M_1_x_2	MbreakN	1.08u	0.36u
M_1_x_3	MbreakN	1.6u	1u
M_1_x_4	MbreakN	1.6u	1u
M_1_x_5	MbreakN	4u	1u
M_1_x_6	MbreakN	1u	4u
M_1_y_1	MbreakN	0.36u	1.08u
M_1_y_2	MbreakN	1.08u	0.36u

$x^{\{2\}}y^{\{0\}}$ = Io2 = 1.831 fA = 0

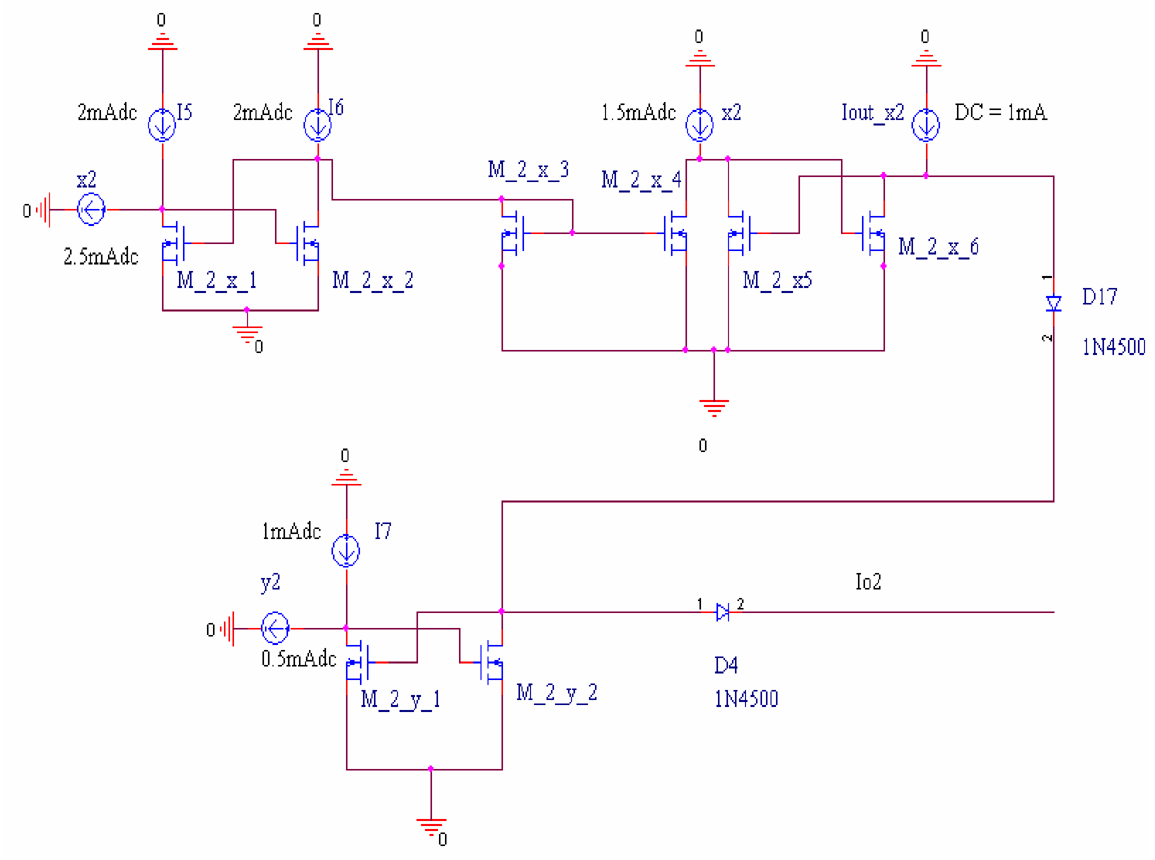


Figure 3.2 MOS realization of $x^{\{2\}}y^{\{0\}}$

Transistor parameters :

Table 3.9 Transistor parameters for circuit $x^{\{2\}}y^{\{0\}}$

Transistor name	**Model**	**W**	**L**
M_2_x_1	MbreakN	0.36u	1.08u
M_2_x_2	MbreakN	1.08u	0.36u
M_2_x_3	MbreakN	1.6u	1u
M_2_x_4	MbreakN	1.6u	1u
M_2_x_5	MbreakN	4u	1u
M_2_x_6	MbreakN	1u	4u
M_2_y_1	MbreakN	0.36u	1.08u
M_2_y_2	MbreakN	1.08u	0.36u

$x^{\{0,1\}}y^{\{1\}}$ = Io3 = 1.850 fA =

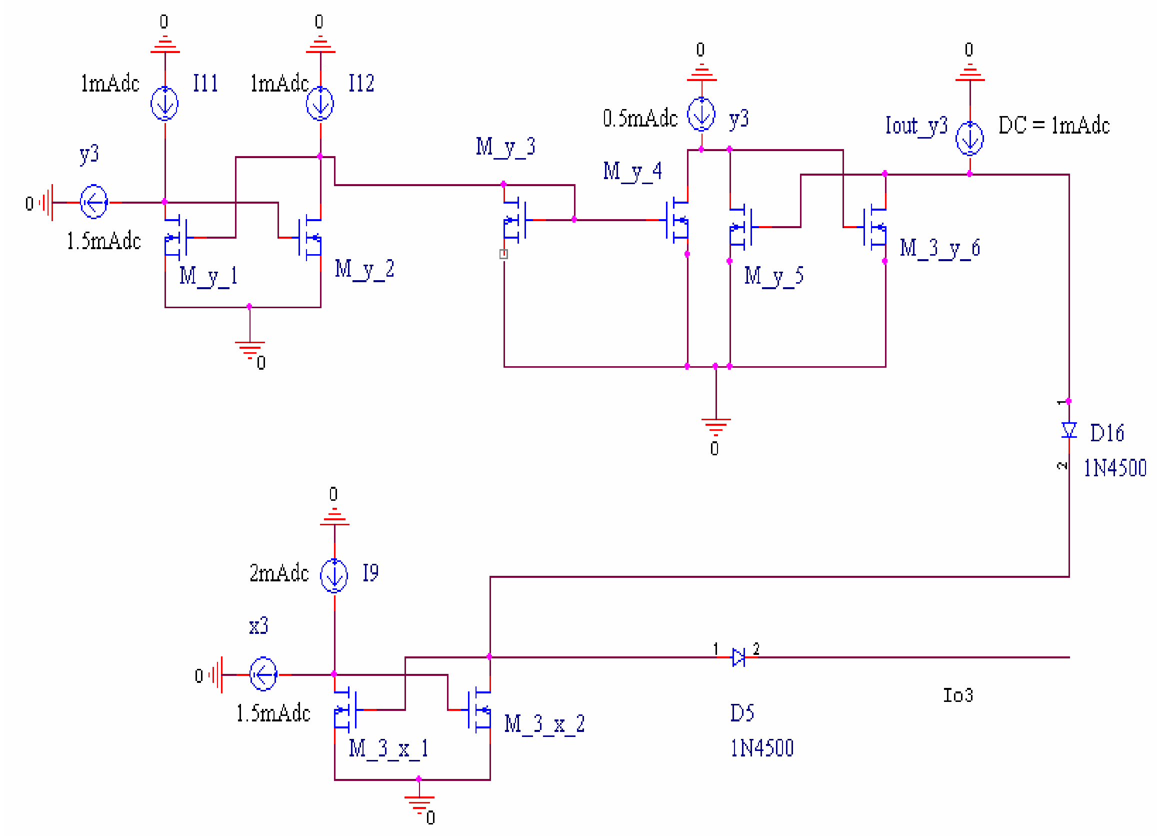


Figure 3.3 MOS realization of $x^{\{0,1\}}y^{\{1\}}$

Transistor parameters :

Table 3.10 Transistor parameters for circuit $x^{\{0,1\}}y^{\{1\}}$

Transistor name	**Model**	**W**	**L**
M_y_1	MbreakN	0.36u	1.08u
M_y_2	MbreakN	1.08u	0.36u
M_y_3	MbreakN	1.6u	1u
M_y_4	MbreakN	1.6u	1u
M_y_5	MbreakN	4u	1u
M_y_6	MbreakN	1u	4u
M_3_x_1	MbreakN	0.36u	1.08u
M_3_x_2	MbreakN	1.08u	0.36u

$x^{\{1\}}y^{\{1\}}$ = Io4 = 3.534e-18 A = 0

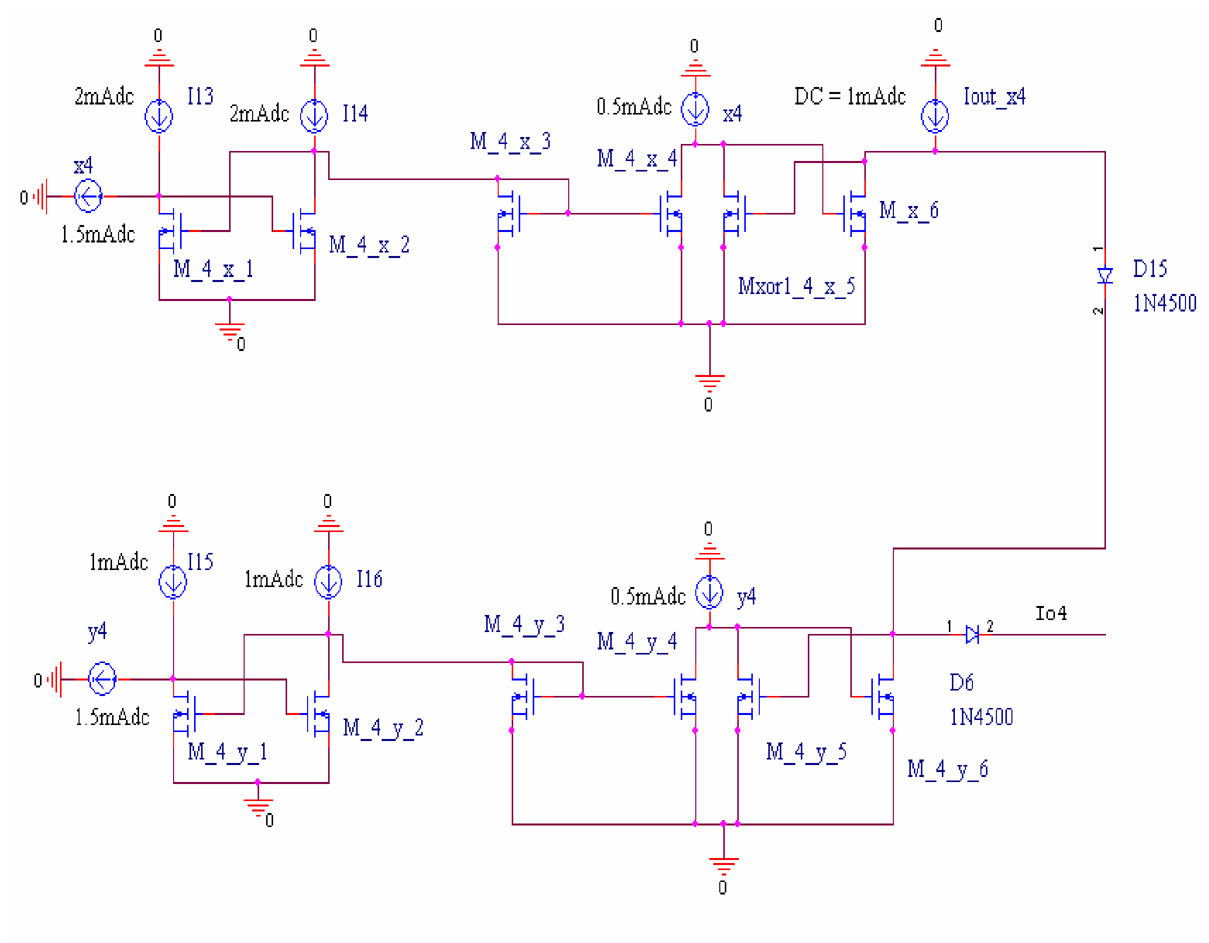


Figure 3.4 MOS realization of $x^{\{1\}}y^{\{1\}}$

Transistor parameters :

Table 3.11 Transistor parameters for circuit $x^{\{1\}}y^{\{1\}}$

Transistor name	**Model**	**W**	**L**
M_4_x_1	MbreakN	0.36u	1.08u
M_4_x_2	MbreakN	1.08u	0.36u
M_4_x_3	MbreakN	1.6u	1u
M_4_x_4	MbreakN	1.6u	1u
M_4_x_5	MbreakN	4u	1u
M_4_x_6	MbreakN	1u	4u
M_4_y_1	MbreakN	0.36u	1.08u
M_4_y_2	MbreakN	1.08u	0.36u
M_4_y_3	MbreakN	1.6u	1u
M_4_y_4	MbreakN	1.6u	1u
M_4_y_5	MbreakN	4u	1u
M_4_y_6	MbreakN	1u	4u

$x^{\{2\}}y^{\{2\}}$ = Io5 = 21.97 fA =
0

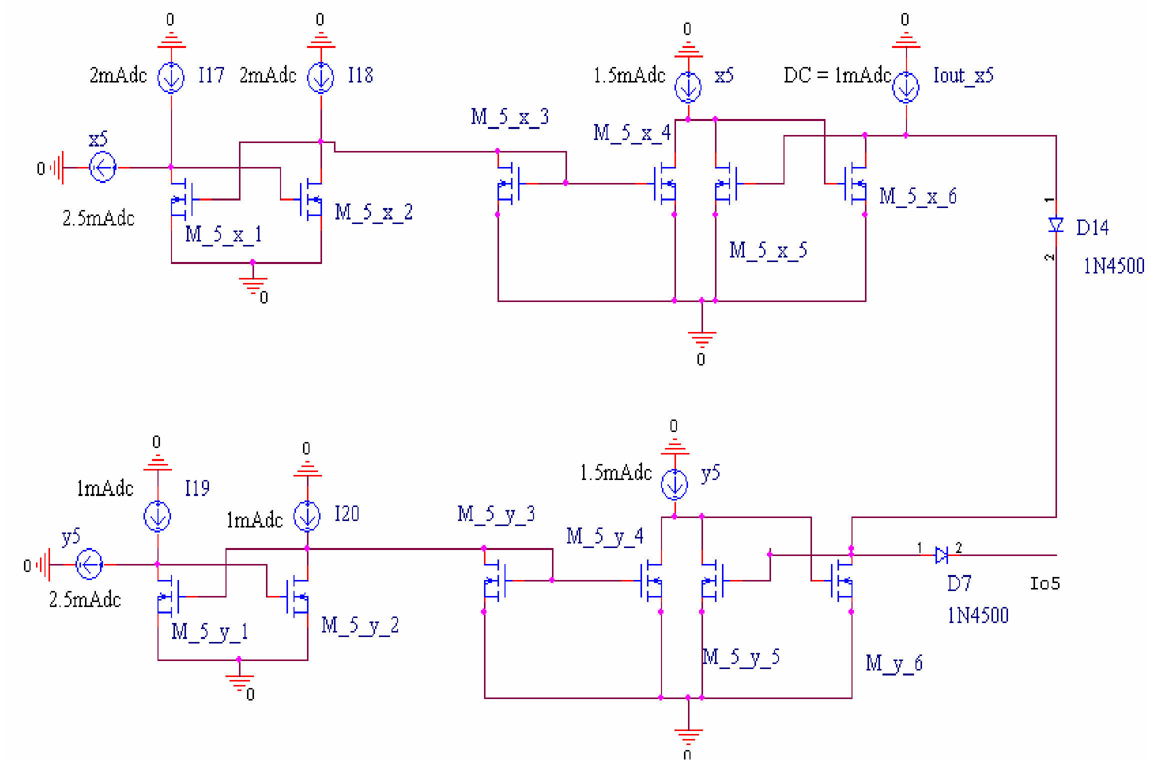


Figure 3.5 MOS realization of $x^{\{2\}}y^{\{2\}}$

Transistor parameters :

Table 3.12 Transistor parameters for circuit $x^{\{2\}}y^{\{2\}}$

Transistor name	**Model**	**W**	**L**
M_5_x_1	MbreakN	0.36u	1.08u
M_5_x_2	MbreakN	1.08u	0.36u
M_5_x_3	MbreakN	1.6u	1u
M_5_x_4	MbreakN	1.6u	1u
M_5_x_5	MbreakN	4u	1u
M_5_x_6	MbreakN	1u	4u
M_5_y_1	MbreakN	0.36u	1.08u
M_5_y_2	MbreakN	1.08u	0.36u
M_5_y_3	MbreakN	1.6u	1u
M_5_y_4	MbreakN	1.6u	1u
M_5_y_5	MbreakN	4u	1u
M_5_y_6	MbreakN	1u	4u

$x^{\{0\}}y^{\{2\}}$ = Io6 = 1.861 fA = 0

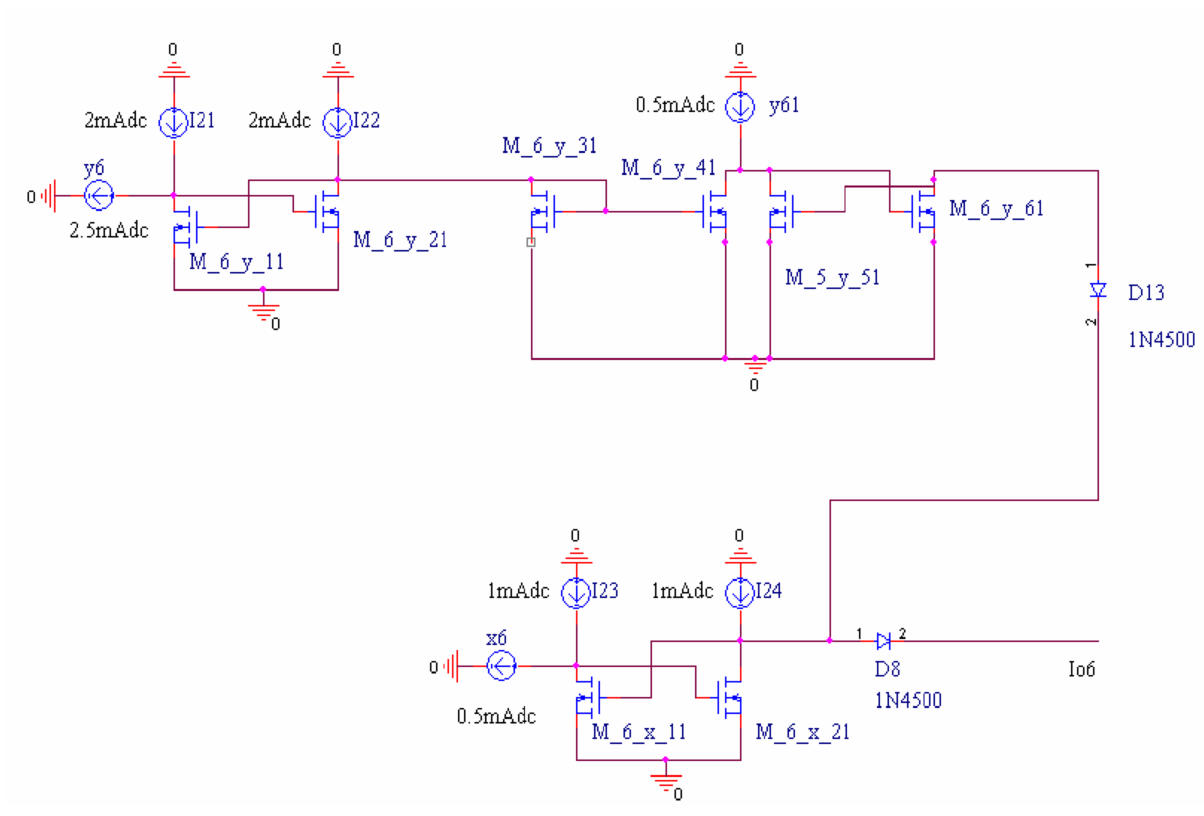


Figure 3.6 MOS realization of $x^{\{0\}}y^{\{2\}}$

Transistor parameters :

Table 3.13 Transistor parameters for circuit $x^{\{0\}}y^{\{2\}}$

Transistor name	**Model**	**W**	**L**
M_6_y_11	MbreakN	0.36u	1.08u
M_6_y_21	MbreakN	1.08u	0.36u
M_6_y_31	MbreakN	1.6u	1u
M_6_y_41	MbreakN	1.6u	1u
M_6_y_51	MbreakN	4u	1u
M_6_y_61	MbreakN	1u	4u
M_6_x_11	MbreakN	0.36u	1.08u
M_6_x_21	MbreakN	1.08u	0.36u

3.5.1.2 Circuit Realization of Addition With Lower Threshold Operator

In chapter two, lower threshold operator was described. If this operator is used instead of literal operator, number of transistors used in the circuit and complexity of the circuit decreases.

$$ {}^{b}_{a}|\, x = \begin{cases} b & \text{if } x \le a \\ 0 & \text{else} \end{cases} $$

According to the formula above, addition operation performed by using lower threshold circuit is shown below:

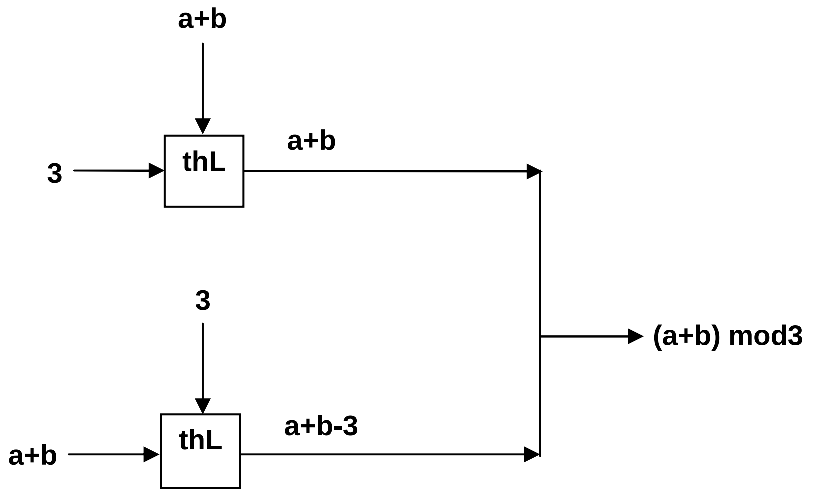


Figure 3.7 Schematic of Realization of Addition With Lower Threshold Operator

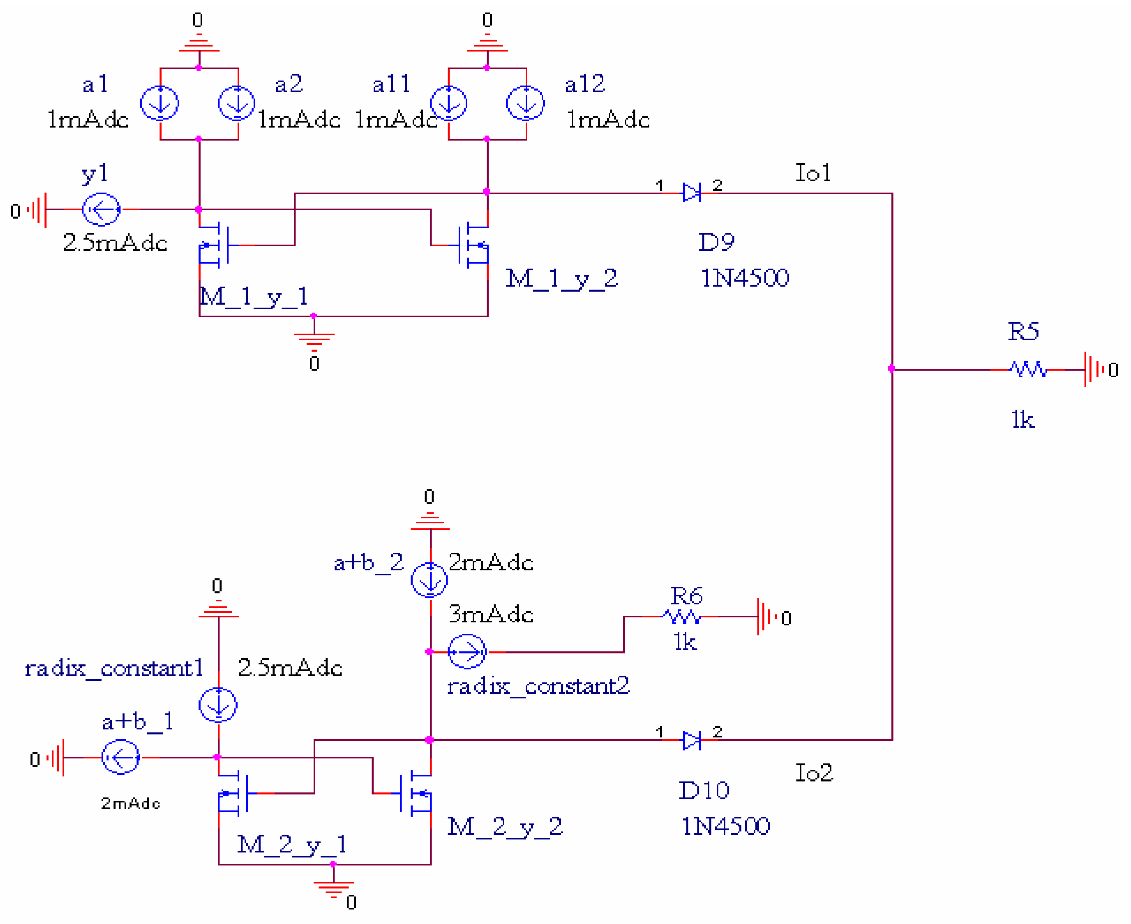


Figure 3.8 Circuit Realization of Addition With Lower Threshold Operator

Table 3.14 Transistor parameters for Realization of Addition With Lower Threshold Operator

Transistor name	**Model**	**W**	**L**
M_1_y_1	MbreakN	0.36u	1.08u
M_1_y_2	MbreakN	1.08u	0.36u
M_2_y_1	MbreakN	0.36u	1.08u
M_2_y_2	MbreakN	1.08u	0.36u

3.5.2 Multiplication

Table 3.15 Multiplication For GF(3)

* x / y	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	1

According to Table 3.14 solution of (x * y) can be expressed by means of literal operator:

$$x * y = x^{\{1,2\}}y^{\{1,2\}} + x^{\{2\}}y^{\{1\}} + x^{\{1\}}y^{\{2\}}$$

Let x = 2 , y = 1,

$x^{\{1,2\}}y^{\{1,2\}} = Io1 = 1 \bullet 1 = 1$

$x^{\{2\}}y^{\{1\}} = Io2 = 1 \bullet 1 = 1$

$x^{\{1\}}y^{\{2\}} = Io3 = 0 \bullet 0 = 0$

x * y = 2 * 1 = 2 mod 3 = 2 , Io1 + I o2 + Io3 = 1 + 1 + 0 = 2

Let x = 1 , y = 1,

$x^{\{1,2\}}y^{\{1,2\}}$ = Io1 = 1 • 1 = 1

$x^{\{2\}}y^{\{1\}}$ = Io2 = 0 • 1 = 0

$x^{\{1\}}y^{\{2\}}$ = Io3 = 1 • 0 = 0

x * y = 1 * 1 = 1 mod 3 = 1 , Io1 + I o2 + Io3 = 1 + 0 + 0 = 1

3.5.2.1 Circuit Realization of Multiplication With Literal Operator

$x * y = x^{\{1,2\}}y^{\{1,2\}} + x^{\{2\}}y^{\{1\}} + x^{\{1\}}y^{\{2\}}$

For x = 2 , y =1,

$x^{\{1,2\}}y^{\{1,2\}}$ = Io1 = 993.8 μA = 1

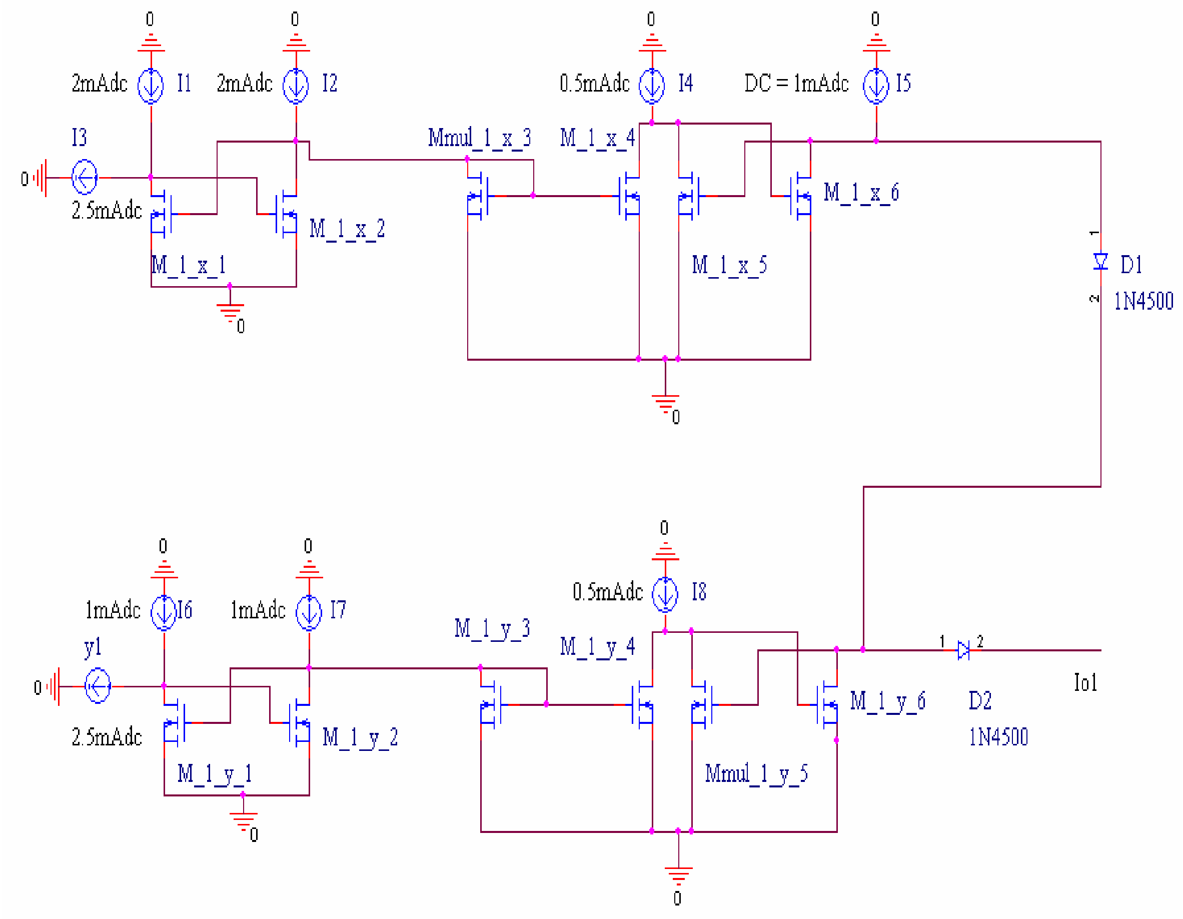


Figure 3.9 MOS realization of $x^{\{1,2\}}y^{\{1,2\}}$

Transistor parameters :

Table 3.16 Transistor parameters for circuit $x^{\{1,2\}}y^{\{1,2\}}$

Transistor name	Model	W	L
M_1_x_1	MbreakN	0.36u	1.08u
M_1_x_2	MbreakN	1.08u	0.36u
M_1_x_3	MbreakN	1.6u	1u
M_1_x_4	MbreakN	1.6u	1u
M_1_x_5	MbreakN	4u	1u
M_1_x_6	MbreakN	1u	4u
M_1_y_1	MbreakN	0.36u	1.08u
M_1_y_2	MbreakN	1.08u	0.36u
M_1_y_3	MbreakN	1.6u	1u
Ml_1_y_4	MbreakN	1.6u	1u
M_1_y_5	MbreakN	4u	1u
M_1_y_6	MbreakN	1u	4u

$x^{\{2\}}y^{\{1\}}$ = Io2 = 963 μA = 1

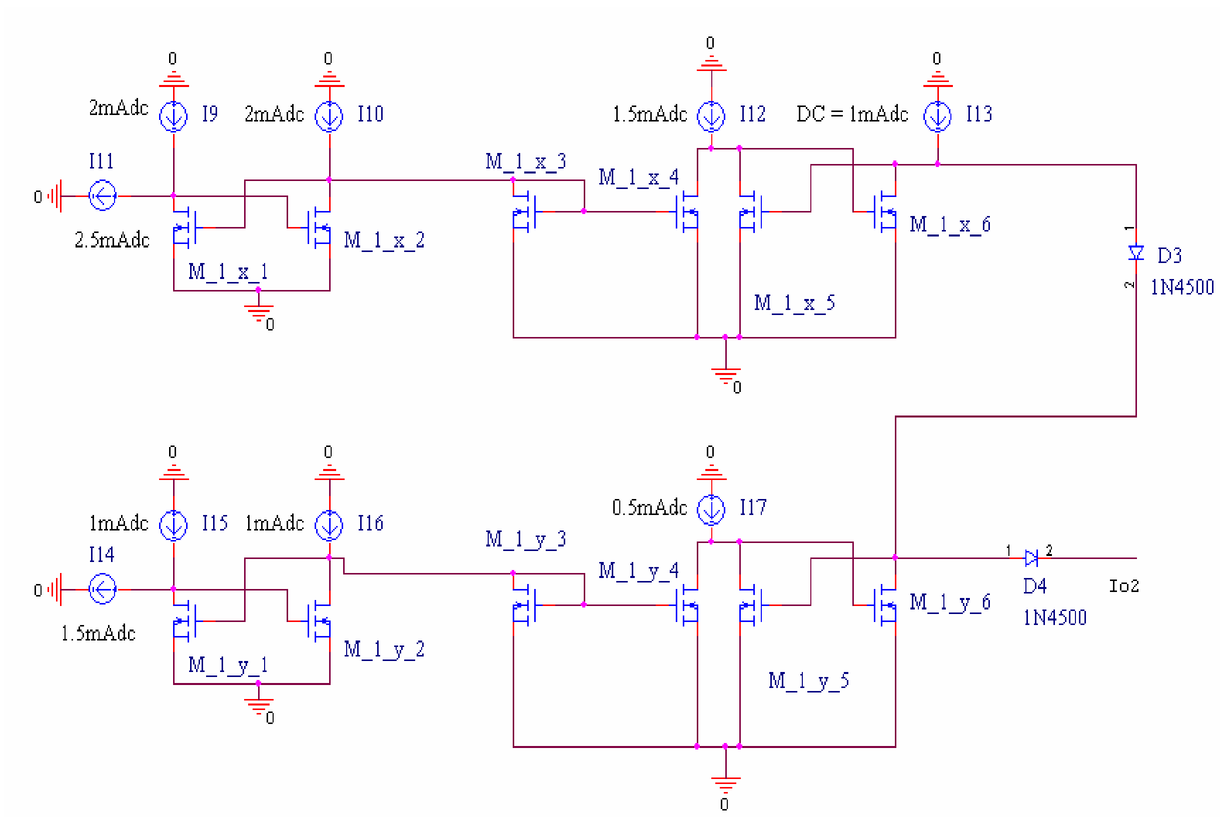


Figure 3.10 MOS realization of $x^{\{2\}}y^{\{1\}}$

Transistor parameters :

Table 3.17 Transistor parameters for circuit $x^{\{2\}}y^{\{1\}}$

Transistor name	**Model**	**W**	**L**
M_1_x_1	MbreakN	0.36u	1.08u
M_1_x_2	MbreakN	1.08u	0.36u
M_1_x_3	MbreakN	1.6u	1u
M_1_x_4	MbreakN	1.6u	1u
M_1_x_5	MbreakN	4u	1u
M_1_x_6	MbreakN	1u	4u
M_1_y_1	MbreakN	0.36u	1.08u
M_1_y_2	MbreakN	1.08u	0.36u
M_1_y_3	MbreakN	1.6u	1u
M_1_y_4	MbreakN	1.6u	1u
M_1_y_5	MbreakN	4u	1u
M_1_y_6	MbreakN	1u	4u

$x^{\{1\}}y^{\{2\}}$ = Io3 = 5.159 nA = 0

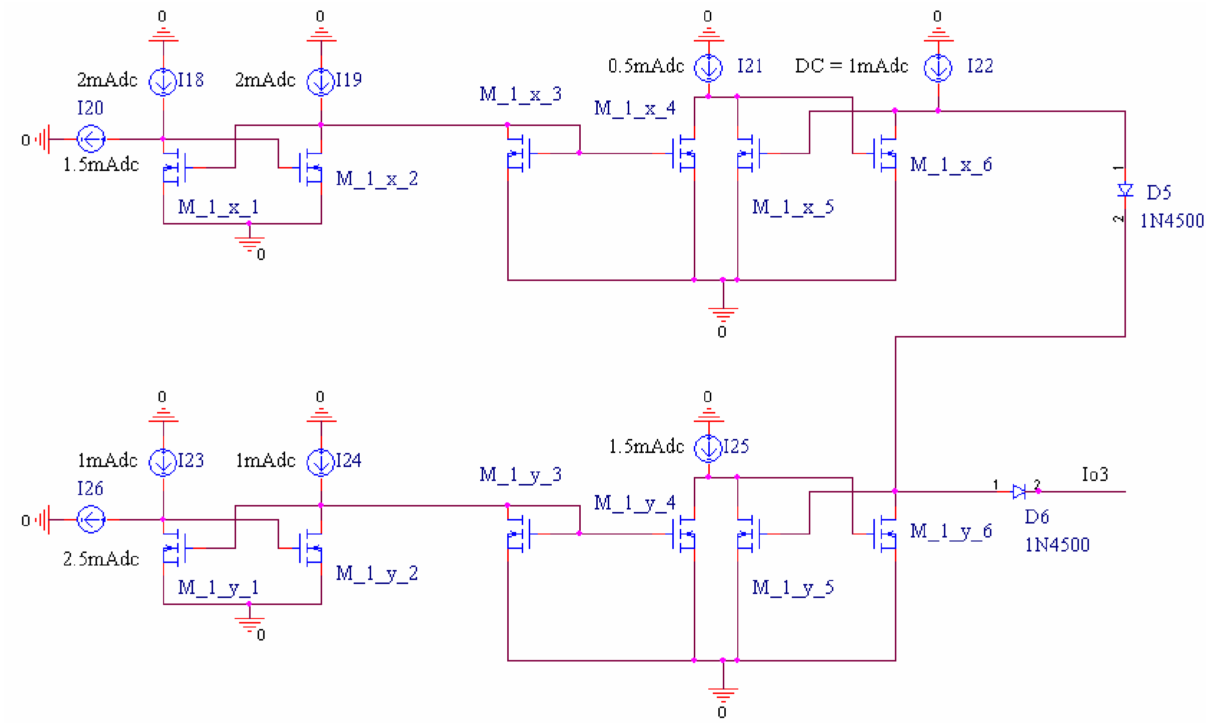


Figure 3.11 MOS realization of $x^{\{1\}}y^{\{2\}}$

Transistor parameters :

Table 3.18 Transistor parameters for circuit $x^{\{1\}}y^{\{2\}}$

Transistor name	**Model**	**W**	**L**
Mmul_3_x_1	MbreakN	0.36u	1.08u
Mmul_3_x_2	MbreakN	1.08u	0.36u
Mmul_3_x_3	MbreakN	1.6u	1u
Mmul_3_x_4	MbreakN	1.6u	1u
Mmul_3_x_5	MbreakN	4u	1u
Mmul_3_x_6	MbreakN	1u	4u
Mmul_3_y_1	MbreakN	0.36u	1.08u
Mmul_3_y_2	MbreakN	1.08u	0.36u
Mmul_3_y_3	MbreakN	1.6u	1u
Mmul_3_y_4	MbreakN	1.6u	1u
Mmul_3_y_5	MbreakN	4u	1u
Mmul_3_y_6	MbreakN	1u	4u

3.5.3 Least-Significant Element First Multiplier

Let assume A , B and C represent polynomials and F(x) represent the irreducible polynomial in GF(3^m). Least-Significant Element First multiplier is a method in which processing of the digits of polynomial B starts from the least significant and continues to the most significant one. The result of multiplicaiton is calculated according to the formula below:

$C \equiv A * B \bmod F(x)$

$\equiv [B_0 A + B_1(Ax^D \bmod F(x)) + B_2 (Ax^D x^D \bmod F(x))$

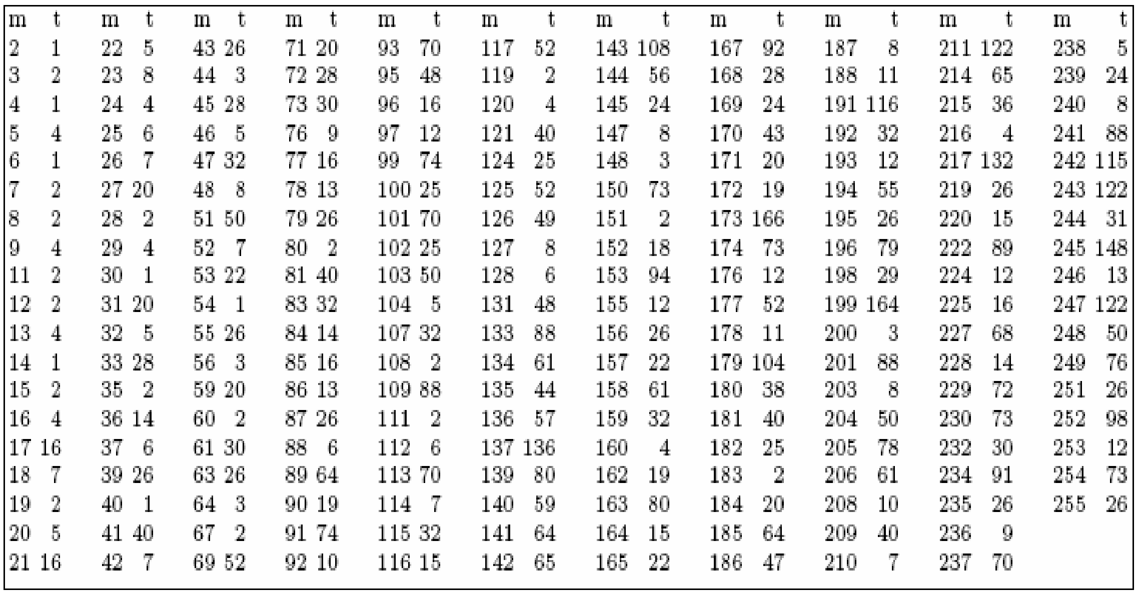
$+ \ldots + B_{d-1} (Ax^{D(d-2)} x^D \bmod F(x))] \bmod F(x)$

The Algorithm of the Least-Significant Element First multiplier is shown below (Guajardo et al., 2006):

```
C = 0
for i = 0 to m-1 do
    C = $b_i * A + C$
    A = A*x mod F(x)
end for
Return (C)
```

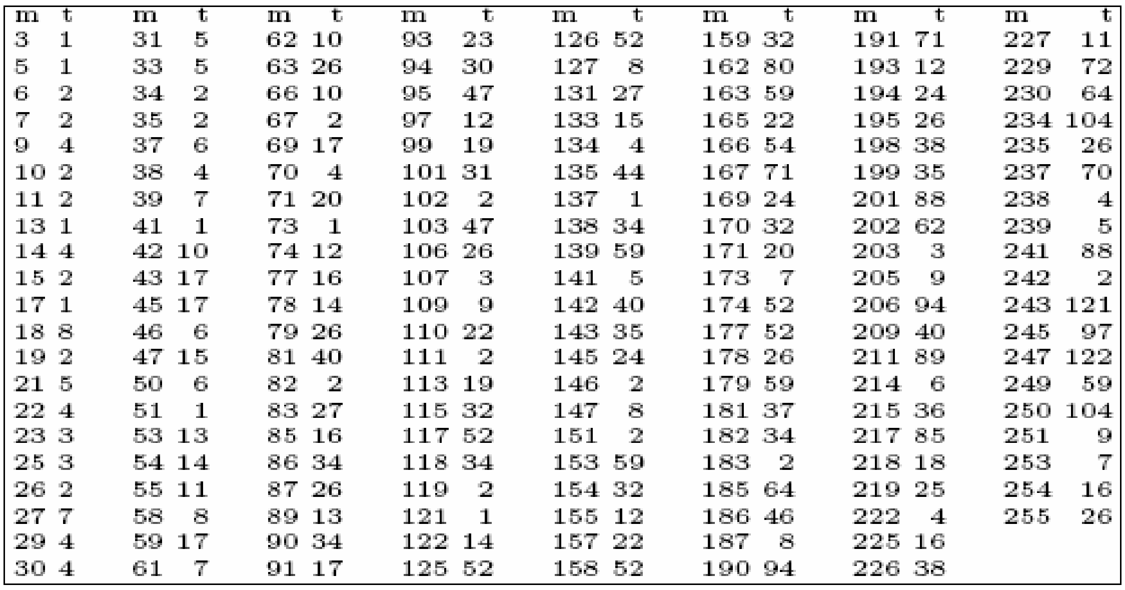
By selecting irreducible polynomial from trinomials listed in the tables (Guajardo et al., 2006) shown below, we can construct a simple architecture for realizing the algorithm above:

Table 3.19 Irreducible trinomials of the form $x^m + x^t + 2$ over GF(3), $2 \leq m \leq 255$



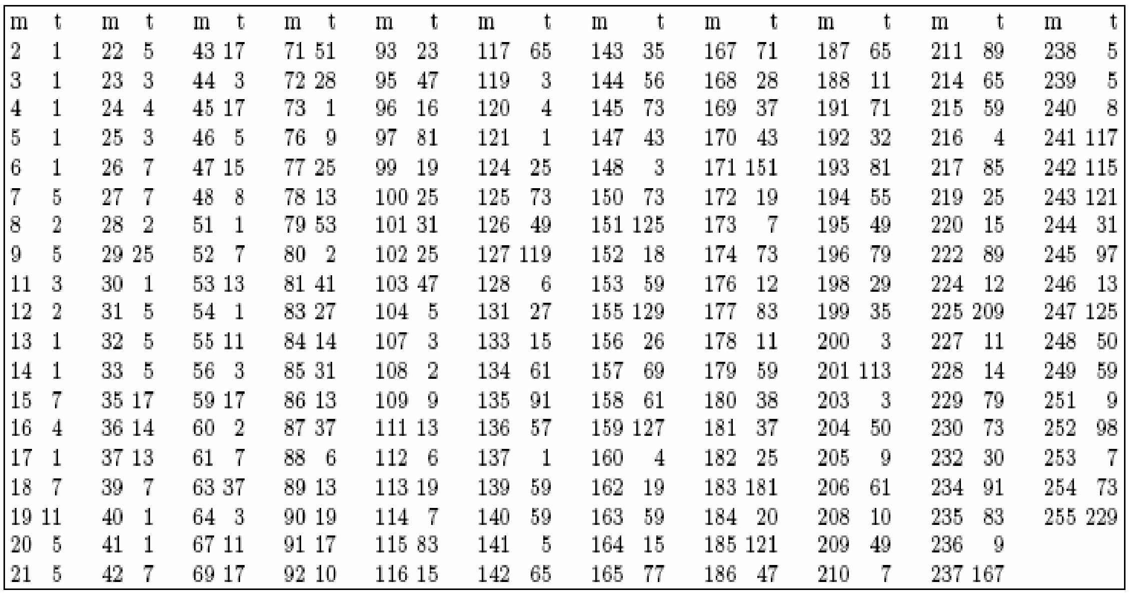
m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t
2	1	22	5	43	26	71	20	93	70	117	52	143	108	167	92	187	8	211	122	238	5
3	2	23	8	44	3	72	28	95	48	119	2	144	56	168	28	188	11	214	65	239	24
4	1	24	4	45	28	73	30	96	16	120	4	145	24	169	24	191	116	215	36	240	8
5	4	25	6	46	5	76	9	97	12	121	40	147	8	170	43	192	32	216	4	241	88
6	1	26	7	47	32	77	16	99	74	124	25	148	3	171	20	193	12	217	132	242	115
7	2	27	20	48	8	78	13	100	25	125	52	150	73	172	19	194	55	219	26	243	122
8	2	28	2	51	50	79	26	101	70	126	49	151	2	173	166	195	26	220	15	244	31
9	4	29	4	52	7	80	2	102	25	127	8	152	18	174	73	196	79	222	89	245	148
11	2	30	1	53	22	81	40	103	50	128	6	153	94	176	12	198	29	224	12	246	13
12	2	31	20	54	1	83	32	104	5	131	48	155	12	177	52	199	164	225	16	247	122
13	4	32	5	55	26	84	14	107	32	133	88	156	26	178	11	200	3	227	68	248	50
14	1	33	28	56	3	85	16	108	2	134	61	157	22	179	104	201	88	228	14	249	76
15	2	35	2	59	20	86	13	109	88	135	44	158	61	180	38	203	8	229	72	251	26
16	4	36	14	60	2	87	26	111	2	136	57	159	32	181	40	204	50	230	73	252	98
17	16	37	6	61	30	88	6	112	6	137	136	160	4	182	25	205	78	232	30	253	12
18	7	39	26	63	26	89	64	113	70	139	80	162	19	183	2	206	61	234	91	254	73
19	2	40	1	64	3	90	19	114	7	140	59	163	80	184	20	208	10	235	26	255	26
20	5	41	40	67	2	91	74	115	32	141	64	164	15	185	64	209	40	236	9		
21	16	42	7	69	52	92	10	116	15	142	65	165	22	186	47	210	7	237	70		

Table 3.20 Irreducible trinomials of the form $x^m + 2x^t + 1$ over GF(3), $3 \leq m \leq 255$



m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t
3	1	31	5	62	10	93	23	126	52	159	32	191	71	227	11
5	1	33	5	63	26	94	30	127	8	162	80	193	12	229	72
6	2	34	2	66	10	95	47	131	27	163	59	194	24	230	64
7	2	35	2	67	2	97	12	133	15	165	22	195	26	234	104
9	4	37	6	69	17	99	19	134	4	166	54	198	38	235	26
10	2	38	4	70	4	101	31	135	44	167	71	199	35	237	70
11	2	39	7	71	20	102	2	137	1	169	24	201	88	238	4
13	1	41	1	73	1	103	47	138	34	170	32	202	62	239	5
14	4	42	10	74	12	106	26	139	59	171	20	203	3	241	88
15	2	43	17	77	16	107	3	141	5	173	7	205	9	242	2
17	1	45	17	78	14	109	9	142	40	174	52	206	94	243	121
18	8	46	6	79	26	110	22	143	35	177	52	209	40	245	97
19	2	47	15	81	40	111	2	145	24	178	26	211	89	247	122
21	5	50	6	82	2	113	19	146	2	179	59	214	6	249	59
22	4	51	1	83	27	115	32	147	8	181	37	215	36	250	104
23	3	53	13	85	16	117	52	151	2	182	34	217	85	251	9
25	3	54	14	86	34	118	34	153	59	183	2	218	18	253	7
26	2	55	11	87	26	119	2	154	32	185	64	219	25	254	16
27	7	58	8	89	13	121	1	155	12	186	46	222	4	255	26
29	4	59	17	90	34	122	14	157	22	187	8	225	16		
30	4	61	7	91	17	125	52	158	52	190	94	226	38		

Table 3.21 Irreducible trinomials of the form $x^m + 2x^t + 2$ over GF(3), $2 \leq m \leq 255$



m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t
2	1	22	5	43	17	71	51	93	23	117	65	143	35	167	71	187	65	211	89	238	5
3	1	23	3	44	3	72	28	95	47	119	3	144	56	168	28	188	11	214	65	239	5
4	1	24	4	45	17	73	1	96	16	120	4	145	73	169	37	191	71	215	59	240	8
5	1	25	3	46	5	76	9	97	81	121	1	147	43	170	43	192	32	216	4	241	117
6	1	26	7	47	15	77	25	99	19	124	25	148	3	171	151	193	81	217	85	242	115
7	5	27	7	48	8	78	13	100	25	125	73	150	73	172	19	194	55	219	25	243	121
8	2	28	2	51	1	79	53	101	31	126	49	151	125	173	7	195	49	220	15	244	31
9	5	29	25	52	7	80	2	102	25	127	119	152	18	174	73	196	79	222	89	245	97
11	3	30	1	53	13	81	41	103	47	128	6	153	59	176	12	198	29	224	12	246	13
12	2	31	5	54	1	83	27	104	5	131	27	155	129	177	83	199	35	225	209	247	125
13	1	32	5	55	11	84	14	107	3	133	15	156	26	178	11	200	3	227	11	248	50
14	1	33	5	56	3	85	31	108	2	134	61	157	69	179	59	201	113	228	14	249	59
15	7	35	17	59	17	86	13	109	9	135	91	158	61	180	38	203	3	229	79	251	9
16	4	36	14	60	2	87	37	111	13	136	57	159	127	181	37	204	50	230	73	252	98
17	1	37	13	61	7	88	6	112	6	137	1	160	4	182	25	205	9	232	30	253	7
18	7	39	7	63	37	89	13	113	19	139	59	162	19	183	181	206	61	234	91	254	73
19	11	40	1	64	3	90	19	114	7	140	59	163	59	184	20	208	10	235	83	255	229
20	5	41	1	67	11	91	17	115	83	141	5	164	15	185	121	209	49	236	9		
21	5	42	7	69	17	92	10	116	15	142	65	165	77	186	47	210	7	237	167		

Ex: $A = 2x + 1$, $B = x + 2$,

irreducible polynomial = $x^2 + x + 2$ $\quad$ => $x^2 = 2x + 1$

$C = A * B \bmod F(x) = (2x + 1) * (x + 2) = 2x^2 + 4x + x + 2$

$= 2x^2 + 5x + 2 = 2x^2 + 2x + 2$

$= 2 * (2x + 1) + 2x + 2$

$= 6x + 4 = 1$

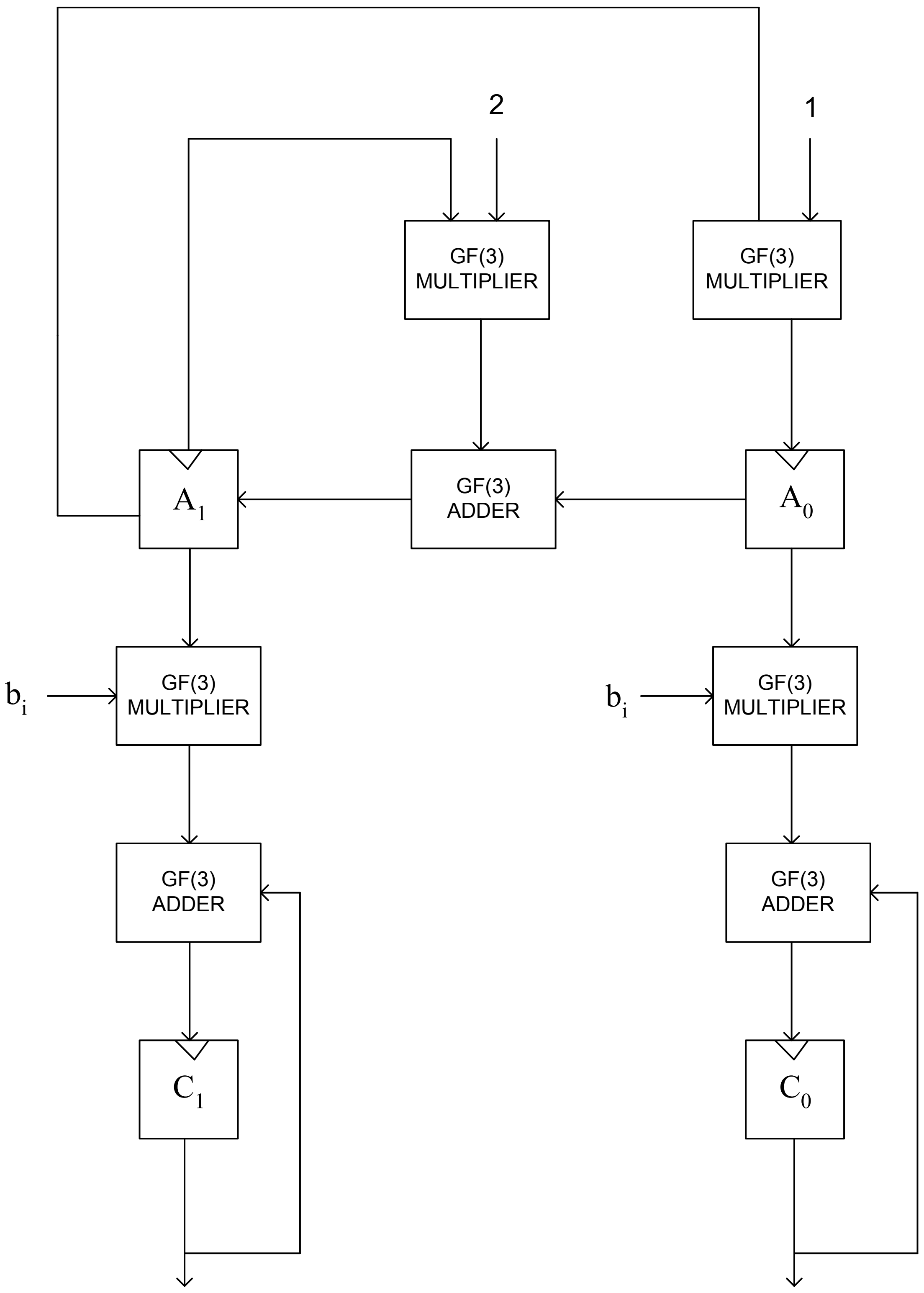


Figure 3.12 LSE multiplier architecture over GF(3^2)

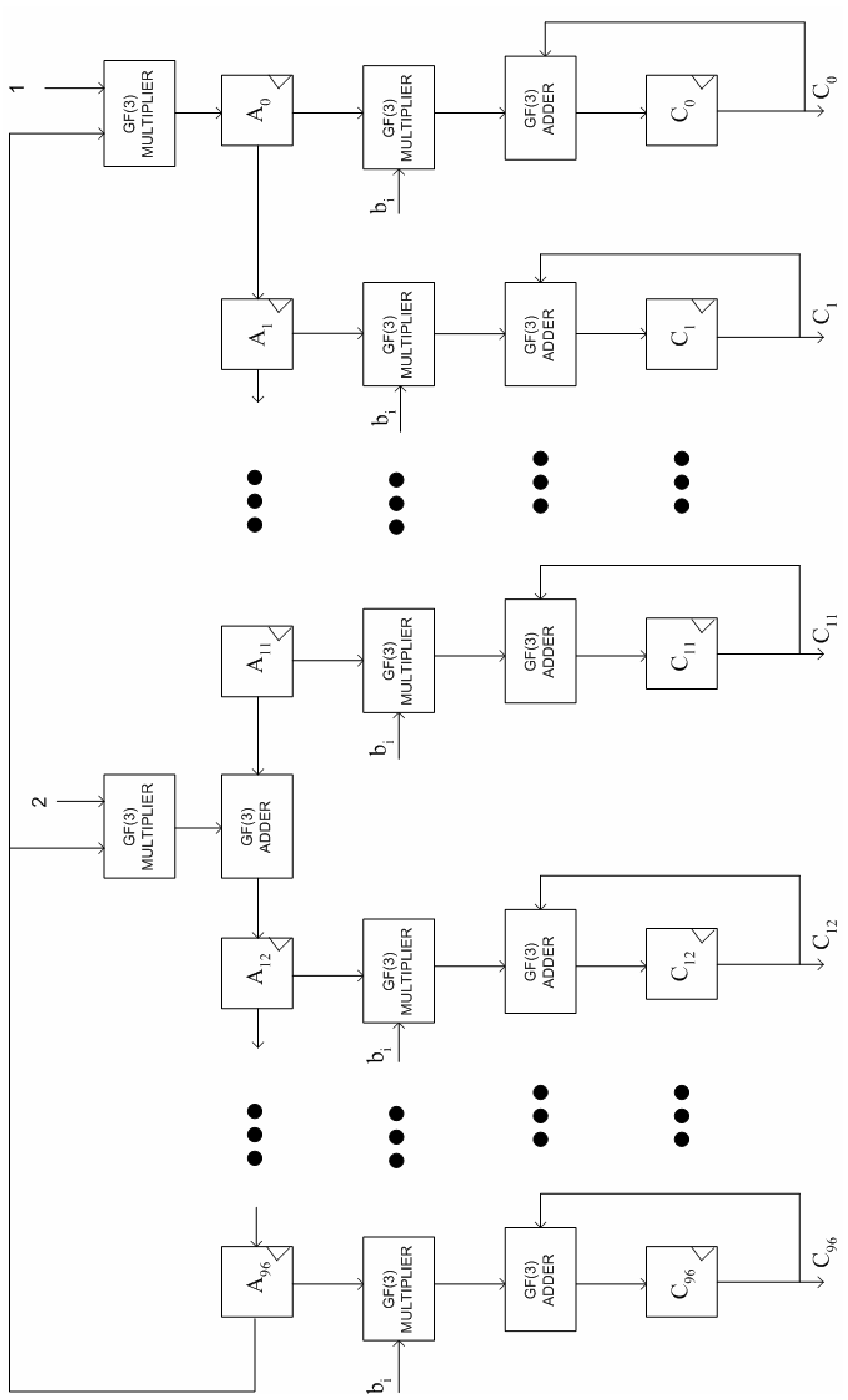


Figure 3.13 LSE multiplier architecture GF(3^{97})

3.5.4 Complexity Estimates for GF(3^m) Multiplication and Comparison with GF(2^n)

The field $GF(2^{151})$ offers security comparable to that of $GF(3^{97})$ for cryptosystems based on the elliptic curve discrete logarithm problem. Circuit realizations of fundamental arithmetic operators for GF(3) are presented before. In order to make a comparison between the hardware implementations of GF(2^{151}) and GF(3^{97}) let's remind MOS realizations of the fundamental binary elements and represent the hardware implementation of polynomial multiplication for GF(2^{151}):

AND:

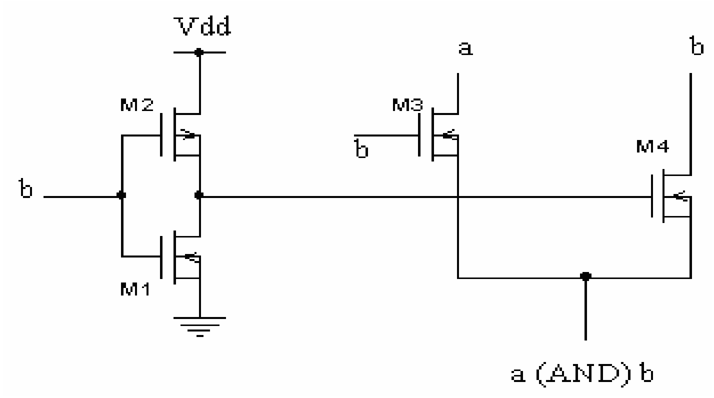


Figure 3.14 MOS realization of AND gate

XOR:

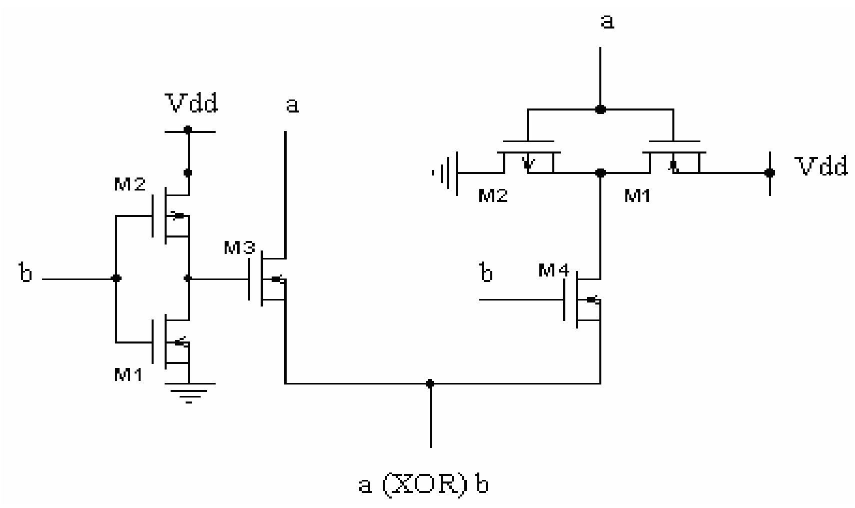


Figure 3.15 MOS realization of XOR gate

NAND:

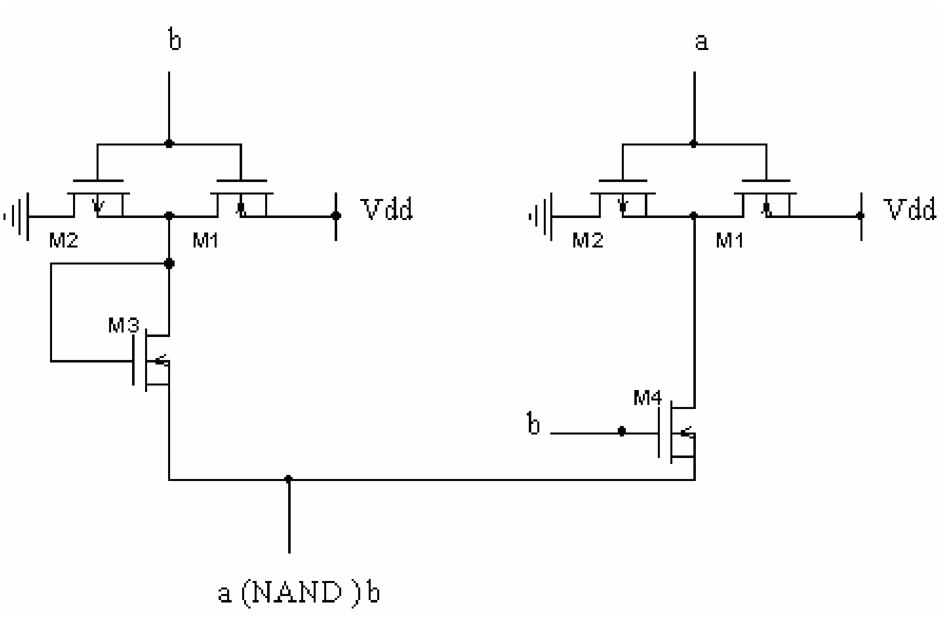


Figure 3.16 MOS realization of NAND gate

D-LATCH:

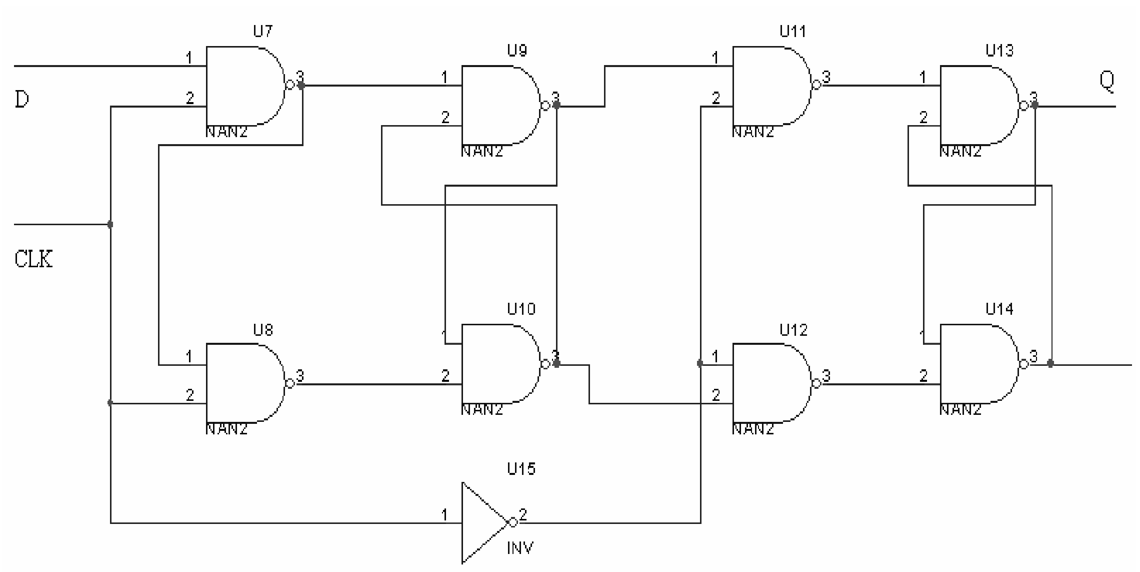


Figure 3.17 D-LATCH with NAND gates

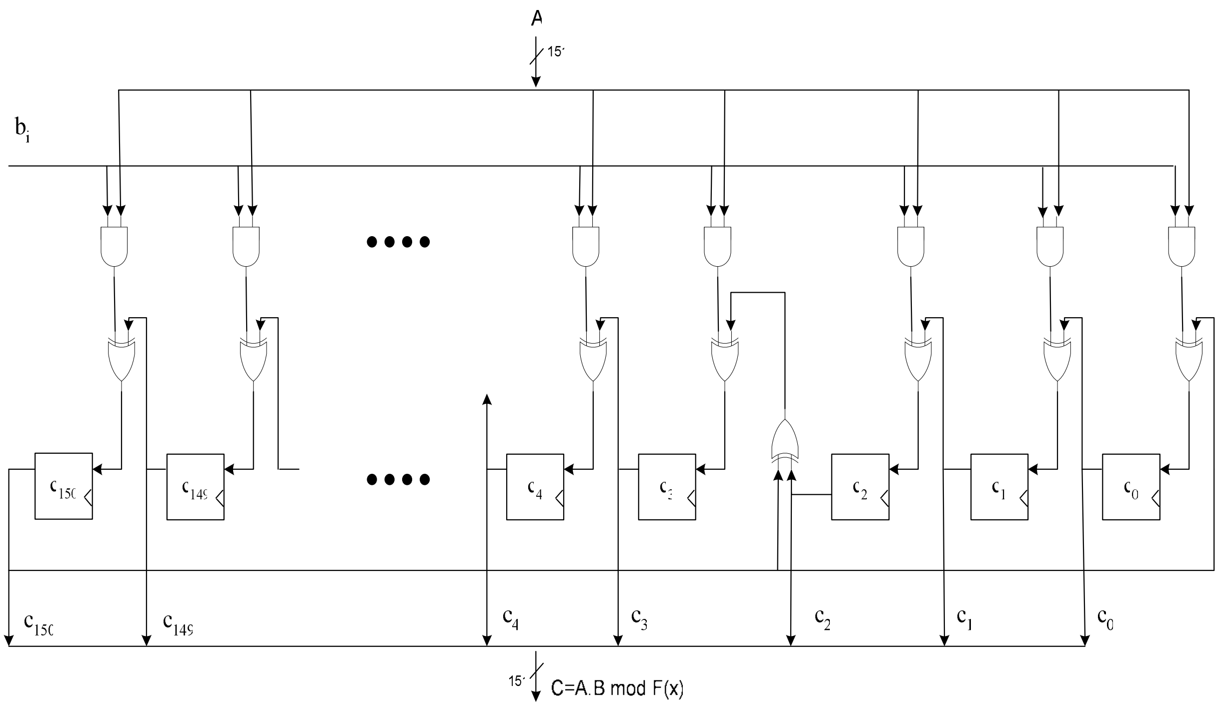


Figure 3.18 Hardware Implementation of polynomial multiplication for GF(2^{151})

Complexity estimates of the circuit given above is represented at Table 3.22:

Table 3.22 Complexity Estimates for GF(2^{151})

Circuit	Number Of Transistors	Number Of Interconnections
AND GATE	4	14
XOR GATE	6	21
D LATCH	50	184
TOTAL FOR GF(2^{151}) : 152 XOR + 151 AND + 151 D	9066	42156

According to the structures of the main components of our architectures above we can make a complexity generalization for GF(3^m). Before that for GF(3^2) and GF(3^{97}) complexity estimates are represented below:

Table 3.23 Complexity Estimates for GF(3^{97})

Circuit	Number Of Transistors	Number Of Interconnections
GF(3) ADDER	4	38
GF(3) MULTIPLIER	36	165
MULTI-VALUE LATCH	27	63
TOTAL FOR GF(3^2) : 3 ADD. + 4 MUL. + 4 LAT.	264	1026
TOTAL FOR GF(3^{97}) : 98 ADD. + 99 MUL. + 97 LAT.	6575	26170

General formula for total number of components, transistors and interconnections for a LSE multiplier over GF(3^m) can be formulized as shown below (Current mirrors are not considered):

Total number of adders: $m + 1$

Total number of multipliers : $m + 2$

Total number of latches: $2m$

Total number of transistors estimated: $(m+1)*(4) + (m+2)*(36) + (2m)*(27) = 94m+76$

Total number of interconnections estimated: $(m+1)*(38) + (m+2)*(165) + (2m)*(63)$

$$= 329m + 368$$

As it is seen from the tables 3.22 and 3.23, number of transistors and interconnections used for GF(2^{151}) , is more than used for GF(3^{97}). Hence, since those two fields GF(2^{151}) and GF(3^{97}) are comparable, for cryptographic applications it can be better to work at GF(3^m) instead of GF(2^n).

CHAPTER 4

CONCLUSIONS

People, always tried to have multi-value and binary logic compete with each other. Always tried to prove ascendancies of them to each other by comparing those two technologies. If the main purpose is comparison, binary logic is the indisputable winner of the competition. On the other hand, instead of making comparisons, using multi-value logic and binary logic in corporation, will generate much more useful and efficient results.

Multi-value logic, reduces the number of elements used in integrated circuits, hence reduces the complexity indisputably. One of the most important reason of that is the accomplishment of linear sum operation by simply adding inputs with wiring. Circuits that use binary logic and multi-value logic incorporation, shows better performance.

In this thesis, definition and fundamental concepts of multi-value logic are introduced. Besides that, fundamental operators and circuit realizations of those operators are shown. Literal and threshold operators are most effective ones between them.

As it is explained before, if multi-value logic can be applied to security mechanisms such as Galois Fields which is very important for some cryptographic algorithms, attack risks can be reduced. We tried to perform basic arithmetic operations of Galois Fields, by using multi-value logic. Addition operation is performed with two different circuits. As you can see, second one is much more simpler than the first circuit. But, because of multi-value truth table of multiplication operation, it could not be possible to realize multiplication operation in a different way.

At the end of previous chapter, it is shown how to perform serial multiplication at Galois Fields(3^m). In the block diagram of the circuit, ‘*’ and ‘+’ and boxes expresses the circuits realized by literal and threshold elements.

In this thesis, serial multiplication at GF(3^m) is performed at system level. Every discrete component of the architecture is realized at the circuit level. For future work, this system level block diagram will be realized at circuit level as a whole.

REFERENCES

Allen, C.M., Givone, D.D., “A minimization technique for multi-valued logic systems”, IEEE Trans. On Computers, Volume C-17, pp.182-184, February 1968.

Andrews, M., ”An Systolic SBNR Adaptive Signal Processor”, IEEE Transactions on Circuits and Systems, Vol. CAS-33, No. 2, pp. 230-238, February 1986.

Arnold, B.K., “Unique Techniques for a Reliable SBNR Architecture”, Proceedings 1988 IEEE Aerospace Applications Conference, Park City, Utah, February 1988.

Arnold, B.K., “Current/voltage hybrid multiplier”, IEEE Electronics Letters, Volume 24, Issue 14, pp. 860 – 862, 7 July 1988.

Arnold, B.K., “Foundational elements and functional analysis of multi-valued logic”, Aerospace Applications Conference, 1989. Digest., 1989 IEEE, pp. 12, 12-17 Feb. 1989.

Butler, J.T., “Multple-valued logic Examining its use in ultra-high speed computation ”, IEEE Potentials, APRIL/MAY 1995.

Butler, J.T., Kerkhoff, H.G., “Multiple-valued CCD circuits,” IEEE Computer, vol. 21, pp. 58-69, Apr. 1988.

Cardoli, M., Schaerf, M., “On the Complexity of Entailment in Propositional Multivalued Logics”, Annals of Marhematics and Artificial Intelligence”, Roma, Italy,1996.

Dao, T.T., "Recent multivalued circuits," in Proc. COMPCON, San Francisco, pp. 194-203, Jan. 1981.

Dao, T.T., “Threshold I^2L and its application in binary symmetric functions and multivalued logic”, IEEE J.Solid-State Circuits, pp. 463-475, Oct. 1977.

Department of Mathematical Sciences College of Liberal Arts and Sciences University of Colorado Denver, “Introduction to Finite Fields”. http://www.math.cudenver.edu/~wcherowi/courses/finflds.html

Elmasry, M.I., "Folded-collector integrated injection logic," IEEE J. Solid-State Circuits, vol. SC-10, pp. 644-647, Oct. 1975.

Etiemble, D. and Israel, M., "A new concept of ternary logic elements," in Proc. 4th Int. Symp. Multiple-Valued Logic, pp. 437-548, May 1974.

Etiemble, D., Israel, M., “Comparison of binary and multivalued ICs according to VLSI criteria”, IEEE Computer Vol. 21, pp. 28 – 42, April 1988.

Epstein, G., Frieder, G., Rine, D.C., "The development of multiple valued logic as related to computer science," Computer, vol. 7, pp. 20-32, September 1974.

Guajardo, J., Güneysu, T., Kumar, S.S., Paar, C., Pelzl, J., “Efficient Hardware Implementation of Finite Fields with Applications to Cryptography ”, Acta Applicandae Mathematicae, Vol. 93, No. 1-3, pp. 75-118, September 2006.

Hanyu, T., Mochizuki, A., Kameyama, M., “Multiple-valued Current-Mode MOS Integrated Circuits Based on Dual-Rail Source-Coupled Logic”, Proceedings of the Twenty-Fourth International Symposiumon Multiple-Valued Logic, Boston, MA, USA, 25-27 May 1994, pp. 19-26, May 1994.

Jain, A.K., Abd-El-Barr, M.H., Bolton, R.J., “Current-Mode CMOS Multiple-Valued Logic Function Realization Using a Direct Cover Algorithm”, IEEE, 1995.

Jain, A.K., Bolton, R.J., Abd-El-Barr, M.H., “CMOS Multiple-Valued Logic Design-Part I: Circuit Implementation”, IEEE Transactions On Circuits And Systems -I: Fundamental Theory And Applications, Vol. 40, pp. 503-514, August 1993.

James, D., “VLSI-MVL implemenmtation of a fast arithmetic cell with SBNR”, IEEE Aerospace Applications Conference, February, 1987.

Kameyama, M., Kawahito, S., Higuchi, T., “A multiplier chip and multiple-valued bidirectional current-mode logic circuits,” IEEE Computer, vol. 21, pp. 43-56, Apr. 1988.

Kerkhoff, H.G., “Theory, Design and Applications of Digital Charge-Coupled Devices”, PhD Thesis, twente University of Technology, Enschede, The Netherlands, April 1988.

Kerkhoff, H.G., Dijkstra, H., "The application of CCD's in multiple-valued logic," in Proc. 5th Int. Conf. Charge-Coupled Devices, Edinburgh, pp. 304-309, Sept. 1979.

Lablans, P., “Multi-Valued Logic (MVL) in Multi-State Switching”, 2005-2007. http://www.multivaluelogic.com

McCluskey, E.J., “Logic design of multivalued I^2L logic circuits”, IEEE Trans. On Computers, vol. C-28, pp. 546-559, August 1979.

Morales, O.J., “An SBNR Floating-Point Convention”, Proceedings 1988 IEEE Region V Conference, Colorado Springs, Colorado, pp. 6-10, March 1988.

Morgul, A., Temel, T., “Current-mode level restoration: circuit for multi-valued logic”, IEEE Electronics Letters Volume 41, pp. 230 – 231, March 2005.

Nascimento, L.P., “An Automated Tool for Analysis and Design of MVL Digital Circuits”, Integrated Circuits and Systems Design, 2001, 14th Symposium pp. 52-57, 10-15 September 2001.

Olmsted, D.D., “The History and Principles of Multivalued Logic”, 1998.

Omnibase Logic, “Multiple-Valued Logic: Do the Math”, 2006. http://www.omnibaselogic.com/business/tech/mvl/math.html

Post, E.L., "Introduction to a general theory of elementary propositions", American Journal of Mathematics, vol. 43, pp. 163-185, 1921.

Smith, K.C., "Circuits for multiple-valued logic-A tutorial and appreciation", in Proc. 6th Int. Symp. Multiple-Valued Logic, pp. 30-43 , May 1976.

Smith, K.C., “The Prospects for Multivalued Logic: A Technology and Applications View”, IEEE Transactions on Computers, Vol. C-30, No. 9, September 1981.

Stallings, W., “Cryptography and Network Security”, Prentice Hall, Upper Saddle River, 2006.

Syuto, M., Shen, J., Tanno, K., Ishizuka, O., “Multi-Input Variable-Threshold Circuits for Multi-Valued Logic Functions”, Proceedings of the 30th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2000), pp. 27, 2000.

Takagi, N., Yaasura, H. And Yajima, S., “High Speed VLSI Multiplicaiton Algorithm with a Redundant Binary Addition Tree”, IEEE Trans. On Comp., Vol. C-34, No.9, pp.789-796, September 1985.

Temel, T., Morgul, A., “Implementation of multi-valued logic, simultaneous literal operations with full CMOS current-mode threshold circuits”, IEEE Electronics Letters Volume 38, pp. 160 - 161, February 2002.

Tirumalai, P.P., Butler, J.T., “Prime and non-prime implicants in the minimization of multiple-valued logic functions”, in Proceedings of 19th International Symposium on Multiple-Valued Logic (IS-MVL), (Ghangzhou, China), pp. 272-279, May 1989.

Vranesic, Z.G., Smith, K.C., Druzeta, A., "Electronic implementation of multi-valued logic networks," in Proc. 4th Int. Symp. Multiple-Valued Logic, Morgantown, WV, pp. 59-77, May 1974.

Vranesic, Z.G., Smith, K.C., "Engineering aspects of multivalued logic systems," Computer, vol. 7, pp. 34-41, Sept. 1974.

Vranesic, Z.G., Smith, K.C., "Electronic circuits for multivalued digital systems," in Computer Science and Multiple-Valued Logic, D. Rine, Ed. Amsterdam, The Netherlands: North-Holland, pp.397-419, 1977.

Vranesic, Z.G., Lee, E.S., Smith, K.C., "A many-valued algebra for switching systems", IEEE Trans. Comput., vol. C-19, pp. 964-971, 1970.

Yamada, M., Fujishima, K., Nagasawa, K., Gamou, Y., "A new multilevel storage structure for high density CCD memory," IEEE J. Solid-State Circuits, vol. SC-13, pp. 688-693, Oct. 1978.