VLSI IMPLEMENTATION OF LCC REED SOLMON DECODER

DISSERTATION-II

Submitted in partial fulfillment of the
Requirement for the award of the
Degree of

MASTERS OF TECHNOLOGY
IN
Electronics & Communication Engineering

By
Remalli Dinesh
(11006538)

Under the Guidance of
Mr. Sandeep Bansal
Assistant Professor
ECE

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA (DISTT. KAPURTHALA), PUNJAB

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINNERING
LOVELY PROFESSIONAL UNIVERSITY
Phagwara-144002, Punjab (India)
APRIL 2015

PAC form

ABSTRACT

In this report, we design LCC decoders for Reed-Solomon (RS) codes. The complete Syndrome generator for RS(128,K_x) and RS(64,K_y) is the first effort in published research for Reed Solomon codes. The known decoding techniques for Reed-Solomon codes are customized to obtain a technique which is suitable for VLSI implementation. The chip architecture of key building block of VLSI RS decoder system are subsequently analyzed for its thorough understanding.

In 0.18µm CMOS technology, the preferred Power and delay was analysed for all the section of LCC Reed-Solomon Decoder which was suitable for scalable VLSI architecture of RS decoding technique. The experimental work start with the design and analysis of PTL AND gate with power and Delay analysis. After while, all the basic circuits was designed in Cadence Virtuoso using the 45 nm technology library of Cadence which collaborate help in designing of main module of Reed-Solomon decoder. The design of RS decoder is designed for three level of design techniques such as CMOS, PTL and GDI techniques. The Power, delay and the PDP values for each and every sub-module is being analyzed and reduce the complexity of the hardware as move from CMOS to PTL to GDI in terms of the transistor count while there are several other trade-off affected with this design flow. The RS codes is constrained to deep space communication, with the growth in VLSI technology, the scientific interest for incorporating RS-decoders in large volume applications rise permanently.

CERTIFICATE

This is to certify that **Remalli Dinesh (11006538)** has completed M.Tech dissertation titled "**VLSI Implementation of LCC Reed Solomon Decoder**" under my guidance and supervision. To the best of my knowledge, the present work is the result of his original investigation and study. No part of the dissertation has ever been submitted for any other degree or diploma.

The dissertation is fit for the submission and the partial fulfillment of the conditions for the award of M.Tech Electronics and Communication Engineer.

Date: _______________

Mr. Sandeep Bansal
Assistant Professor
Department of ECE
Lovely Professional University

ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude to **Mr. Sandeep Bansal** who gave his heart whelming full support in the compilation of this dissertation-II with his valuable suggestions and encouragement to go ahead in all the time of the dissertation-II.

I am also thankful to **Prof. Bhupinder Verma**, HOS, School of Electronics Engineering, School of Electrical And Electronics Engineering for providing us with the infrastructure in carrying more interesting the Research.

I would also like to thank other research scholars and faculty members, School of Electronics And Communication Engineering for their kind support during this work.

At last but not the least my gratitude towards my parents, I would also like to thank my friends for the strength that keep me standing and for the hope that keep me believing that this dissertation will be possible.

REMALLI DINESH
11006538

DECLARATION

I hereby declare that the dissertation entitled, "**VLSI Implementation of LCC Reed Solomon Decoder"** submitted for the M.Tech Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date:_______________

Remalli Dinesh
11006538

PUBLICATIONS

[1].Remalli Dinesh and Sandeep Bansal, "Design and Formulative Analysis of VLSI Syndrome Generator for RS(128, K_x) and RS(64, K_y)". IJCA - International Journal of Computer Applications, Vol. 92, No. 8, pp. 17- 21, April 2014.

[2].Remalli Dinesh, Sandeep Bansal et al., "High Efficiency Low Complexity Chase Architecture for Reed-Solomon Decoder of RS(255, K)". International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE), Vol.1, No. 4, March 2015.

[3].Remalli Dinesh et al. "A Comparative Analysis of Low Power High Speed 1-Bit Full Adder from 28 to 10 Transistor using 0.18μm Technology". IEEE- ICKCE, (Accepted for Publication) and submitted to IEEE-EDAS Bangalore.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABELS

CHAPTER 1
INTRODUCTION

In this thesis we design and implement the Low Complexity Chase (LCC) Reed-Solomon (RS) Decoder having its architecture designed in Cadence using 45nm CMOS technology, PTL logic and GDI Technique. Here we brief about the history of Reed-Solomon and LCC algorithms.

1.1 Background: RS Code

The central problem in the information transmission is efficient and reliable data transmission from source to destination. When you are using any source of information also in space exploration mission it is essential use of information theory. Reed-Solomon (RS) codes are a kind of algebraic FEC codes and it were pioneered by Irving S. Reed along with Gustave Solomon in 1960 [1]. Both of them explain a proficient way of making codes with the intention of detecting and correcting several random symbol errors. As a result of addition of‘t’ check symbols in the data word, the RS code can find out some arrangement of t amount of erroneous symbols, or rectify it up to symbols t/2. Since in the erasure code, there is limit up to‘t’ erasures can be able to rectify so that it can locate and also correct the pattern of errors as well as erasures.

Reed-Solomon (RS) codes have established well-known applications in data accumulation and communication due to their straightforward decoding and their potential to fix bursts of errors. The first RS codes decoding algorithm develop by Peterson [2]. Afterward Berlekamp [3] and Massey [4] shorten the algorithm by screening that the decoding problem is corresponding to locate the shortest linear feedback shift register (LFSR) which generates a set of given sequence. The algorithm is known as Berlekamp-Massey algorithm (BMA) [1]. From the time since when BMA has been discovered, much effort has been done in the improvement of Reed-Solomon hard decision decoding algorithms. The algorithm which was mostly used for the purpose of error location polynomial was Euclid Algorithm [5]. Berlekamp and Welch invented an algorithm which precludes the syndrome computation, foremost step in the formerly proposed decoding algorithms. Each and every previous algorithm can rectify the errors upto half of the smallest possible distance i.e. d_{min} of given codewords. There has not been any progress on decoding performance since 45 years after the beginning of RS codes.

A RS code has been preferred by both the ESA and NASA as the external code in a coding scheme. The RS code is represented by RS(N, N-32) of 8-bit symbols block code capable of correcting 16 symbol errors. A shortened code is created when N<255 which was enviable for some applications.

The first VLSI implementation of decoder has been done by Liu which used 40 VLSI chip pieces with [5] 100 support chips. The efficiency of communication channels has an extremely prominent effect of RS error correction codes [6]. RS codes are systematic codes which left the data unaltered instead append the parity symbols of the stream of data and they are created through encoding the information stream by a code generator. Also data of a n-bit symbol is found to be factor of every 2^n elements in the Galois Field ($GF(2^n)$) [6]. The RS codes taken into account code generator polynomial such that it can reproduce codes within the encoder as well as to locate the error and rectify it inside the decoder. The RS(n, k) codes can determine and rectify errors upto (k-n)/2 error symbols inside a codeword having length of k-symbol. Afterward the codes are being transmitted in the communication channel for the receiver terminal. The received codeword are corrupted with the incursion of noise and other surrounding issues inside the channel. The intention of the decoder is to locate the values and locations of the errors reside in the received codeword. The intricacy of the RS codec lays in a decoder. Effective algorithms as well as VLSI architectures advancement are necessary to design a architecture of high speed decoder. While new algorithms with the architectures are still invent to put down the complexity problem of decoder section with increase in its speed [6].

1.2 Syndrome Generator

It is an amount of how faraway the received code word from the one which was transmitted [5]. RS code block is represents as [5]:

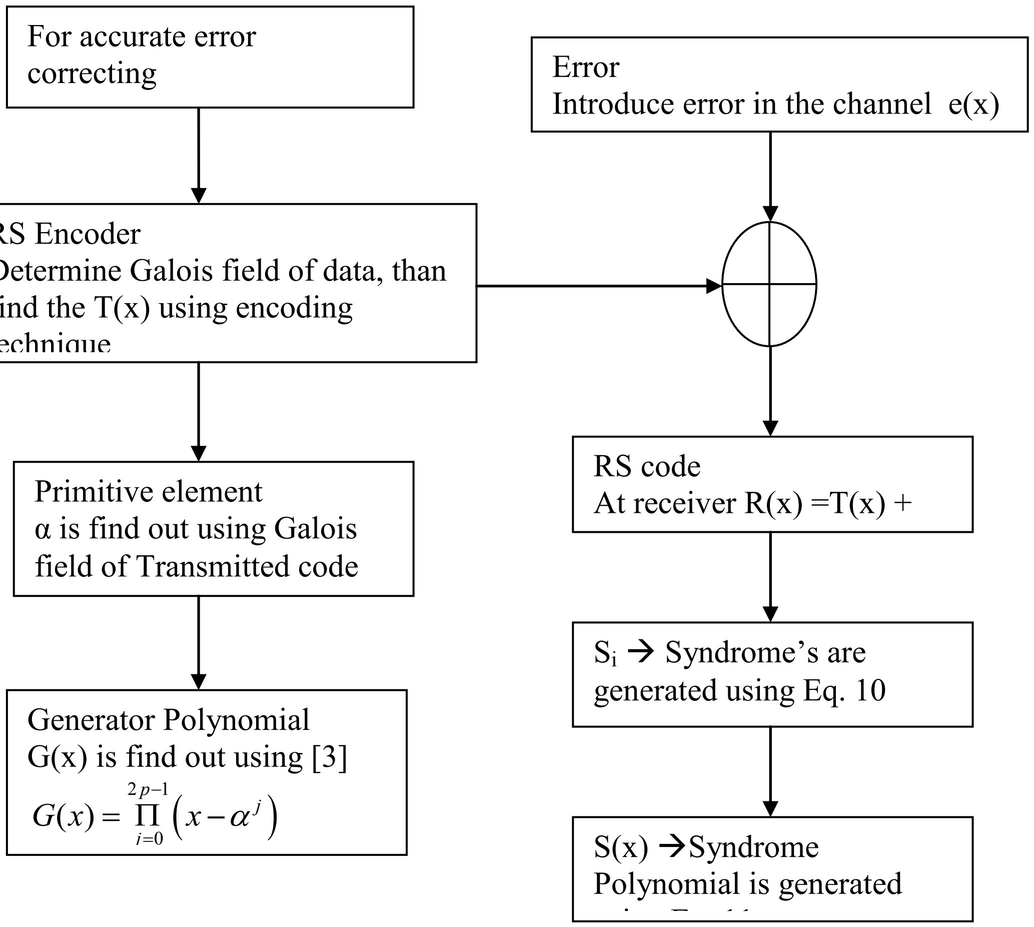
$$C(x) = x^{32}M(x) + M(x)\,|\,G(x)\,| \tag{1}$$

Every suitable code block is a factor of the generator polynomial which is represented as G(x) [5] ,

$$G(x) = \prod_{i=0}^{2t-1} (x - \alpha^{i}) = \sum_{j=0}^{2t} G_{j} x^{j} \tag{2}$$

Where, α is Galois field primitive element.

Galois field was defined as an algebraic field which has limited number of members. Galois fields include 2^{m} members are utilized in error-control and are denoted by GF(2^{m}). since m is an integer having value lies between 1 and 16. The primitive element of the GF(2^{m}) is a cyclic generator value of the group of some nonzero elements of GF(2^{m}). It means that every nonzero factor of the field can be conveyed as the primitive factor raised to integer power.



For accurate error correcting
Error
Introduce error in the channel e(x)
RS Encoder
Determine Galois field of data, than find the T(x) using encoding technique
Primitive element
α is find out using Galois field of Transmitted code
RS code
At receiver R(x) =T(x) +
Si → Syndrome's are generated using Eq. 10
Generator Polynomial
G(x) is find out using [3]
$G(x) = \prod_{i=0}^{2p-1} \left(x - \alpha^{j}\right)$
S(x) →Syndrome
Polynomial is generated

Fig.1.1. Block diagram depicts flow from RS Encoder to Syndrome Polynomial Generation

Galois field can be determined by an irreducible polynomial, P(x) [5]:

$$P(x) = x^8 + x^7 + x^2 + x + 1 \tag{3}$$

And during transmission, the code undergoes changes due to noise present in the channel. Therefore the received polynomial declared as [5]:

$$R(x) = C(x) + E(x) \tag{4}$$

Where, E(x) is a representation for error polynomial.

Syndrome polynomial is expressed as [5],

$$S(x) = R(x) \bmod G(x) \tag{5}$$

We can also expressed syndrome polynomial as [5],

$$S_k = \sum_{i=0}^{n-1} r_i b^{i(k+s)} \tag{6}$$

Where, $0 \le k \le 2t-1$

Another expression of syndrome polynomial was [5]

$$S(x) = \sum_{k=0}^{2t-1} S_k x^k \tag{7}$$

For any RS(n, k) code, α^js $(0 \le j \le (k-1))$ indicate the possible error locations. We can confirmed the received polynomial is suitable codeword only when each and every syndrome S_i $(1 \le i \le 16)$ are zero [9].

1.3 Chase Decoding

In the year 1968, the known first decoding solution for the RS codes was invented by Berlekamp [1]. There are algorithms such as HDD which are able to achieve decoding of a particular vector and ensure that it has error not more than $d_{min}/2$, whereas d_{min} is considered to be the least distance of the incoming code. As in modern communication, still different procedures of Berlekamp real model are in operation even decades passed after its generation. Despite a bulk of research has already done but the proficiency of RS decoding upto its limit of $d_{min}/2$ exceeds through Sudan [10] work. Although there found to be trade off in performance gain and complexity exponential and they are inversely proportional with respect to the decoder expansion of radius. And the Sudan's research was legalized by Koetter and Vardy in the year 2000 [10] and it can be achieve with the introduction of reliability information and that helps in achieving better coding gain with respect to favorable complexity. And all these techniques of decoding RS codes are ahead of $d_{min}/2$, and Chase in 1972 [10] had designed a method of decoding which are found to be the nonspecific for the type of code and that permits for any existing HDD procedure to raise its decoding radius.

In the conventional HDD procedure, a hard decision vector known as Maximum a- Posteriori (MAP) is utilized for providing input vector to a decoding algorithm [3]. Whereas in the Chase method of decoding, to generate a particular set of test vector it tend to use its reliability information. For the decoding of codes beyond the limit of $d_{min}/2$ with consideration of regular techniques for decoding, it need to have vector which comes within the decoding radius for the algorithm which is in presently in operation [10]. However, it will enhance the performance but at the cost of linear advancement of complexity depends on cardinality of test-set.

In the presented work, we being consider decoding of RS codes through Chase technique of using test-vector in sets which are same in every manner but with a small count of indices. With the introduction of point-by-point interpolation algorithm through which a procedure is presented to show up the similarity for the demise of complexity of an starting decoding step of interpolation.

1.4 LCC Algorithm

In this the η unreliable positions of the code are chosen by multiplicity assignment according to the reliability information given at the input whereas η is considered to be a positive integer.

Since every η positions is allocate with the two interpolation points such as: (α_j, β_j) and $(\alpha_j, \beta_j^{'})$, and anyone point (α_j, β_j) is assumed to be allocate to remaining n–η positions of code. And for the determination of mapping encoding the field element α_j is being used whereas β_j and $\beta_j^{'}$ symbolizes the hard decision along with symbol for jth code position which is most likely respectively. Also the multiplicity of every interpolation point is found to be one. For every position of code only one interpolation point is need to make the test vector while the overall test vector is 2^{η} and there is only two participating points for every single η most unreliable code positions.

CHAPTER 2
REVIEW OF LITERATURE

Kuang Yung Liu et al. 1984 [8] endeavor to adapt for the known decoding procedures in order to gain for a recursive along with repetitive process which was appropriate for the VLSI implementation as well as pipelining. In this the author demonstrated about the symbol-slice logic architecture which is apt for the VLSI implementation of RS codes.

Since the author estimated, if bit-serial as well as bit-parallel processes are work for the syndrome generator along with for the LFSR synthesis chips respectively. After that both the chips then utilizes almost 4000 number of MOS gates and the throughput of this decoding system is approximately 4Mbit/s. The author claimed that because of size and power benefits in the VLSI decoding system and the speed of decoding can easily raised up to a certain level using distributed processing scheme.

Gary K. Maki et al. 1986 [5] presented the VLSI implementation of decoder which accepts data at the rates of 80Mbit/s. To make it possible a total of 7 chips are used and operate using the symbol clock while the system clock for chip is already set and nearly 1.65 billion GF operations/sec are attained with this chip set. The author reviewed various papers from the literature , in one such paper it required 40 VLSI design chips with 100 support chips also operating at 2.5Mbit/s rate. It uses systolic arrays while the author presented a group of custom architectures utilized by each module.

The author design steps followed the sequence as Syndrome generator, Euclid divide/multiply, Polynomial solver and finally error correction. The decoder presented can be able of rectifying the errors up to 16 symbol having 80Mbit/s data rate.

Hung-Wei Chen et al. 1995 [11] reckoned about the VLSI based architecture for RS decoder with the erasure function and via modified Euclid's algorithm for solving the key equation in order to diminish the hardware. The correction ability is found to be 20bytes/block whereas block length ranges from 96 to 255 bytes and the complexity of Hardware depends on 2t. The author used erasure function in his system and it is defined as an error whose position is known but not magnitude. With erasure, RS(N,K) having distance d=2t+u+1 can rectify 'u' erasures and t errors. The author choose Euclid’s algorithm for solving key equation because it is easy to understand than BMA, easy to execute, always congregate for a zero remainder in a limited count of the steps, most importantly hardware complexity is not rise when t is increased.

Wolfgang Wilhelm et al. 1999 [12] presented a VLSI architecture for RS decoding technique which is scalable w.r.t the throughput rate and it is achieved through systematic time-sharing technique. Using this, a new multiplexed architecture has been created to produce key equation as well as to implement the field divisions finite.
An RS(n,n-2t) is defined over a certain finite field GF(2^m), 2t parity symbols is include in each code of length n. By Chien search, error location is revealed which is time-domain multiplexed along with Forney algorithm utilized to correct the code error by adding the quotient to the received symbol. Decoder can be extended further for supporting erasures corrections, in which correction capability can be increased by marking the position of detected erasures without changing the parity symbols. The advantage is that erasure need one parity symbol instead of two.

Dong-Sun Kim et al. 1999 [15] discussed about the digital coded system is used presently for transmission of information and also for the data storage that founds to be similar in both of them. The outcome of the storage or the channel varies from the available input as they are sensitive for errors that produce some results from the affected transmission. Due to this, RS codes are being used in various fields. The author of this paper shows the VLSI design of RS decoder consist of the enhancement of architecture supporting parallel as well as pipeline way of processing, so as to improve its speed and minimize its power through its design. He also analyzed the decoding algorithm of RS codes for proficient pipeline and parallel architectures.

Anh Dinh et al. 2004 [6] deduce the high speed implementation of RS(255,239) decoder by using 180nm CMOS technology with 8-error correcting capability. The decoder uses the concept of division free algorithm and the modified BMA along with key equation solver, a terminated process in Chien search. The complexity is low, also due to low latency of inversion circuits and power-sum it boost-up the speed of the decoder and chip area is 1.5mm^2 and data processing rate is greater than 1Gbits/s.

Richard Huynh et al. 2009 [14] proposed that in most of the present communication system have vastly adopted the RS codes due to its proficient performance. In this paper, author also describes some new ways for detecting the error in syndrome block in a RS decoder. Instead of having whether the codeword is right it is acceptable to calculate only few syndrome which was less than the half of the available syndromes. The algorithm of error detection in the syndrome block need not require any sort of modification for a basic implementation of hardware for the coefficient of syndrome computation. It helps in demising of the computational complexity of a syndrome block which results in reducing the required power.

W. Zhang et al. 2012 [13] proposed that ASD of the RS codes is able to avail efficient gain in terms of coding as compared to the HDD along with the complexity in polynomial. When look for the another ASD based algorithm the LCC founds to be minimum computational complexity along with 2^{η} test vectors and also same or a better gain. In order to minimize the interpolation latency, number of interpolators in the pipeline manner can be recommended by the LCC decoding but there is some trade-off in terms of power consumption as well as in area that is not supposed to preferred in the pipeline decoder. The solution of this is to assume a serial LCC decoder proposed by author having low complexity for the RS(458, 410) code on the GF of 2^{10}.

Kunal et al. 2012 [19] present the Gate Diffusion Input (GDI) techniques. As it is fast and suitable for low power circuits with the less amount of transistor as compared with the conventional CMOS based design along with presently existing PTL techniques. This technique allows reuse of proficient design algorithm which are depends on Shannon expansion method. That results in generation of combinational circuits through this GDI in the VLSI industry without using any sophisticated library. So this makes the GDI to be an extra benefit when compared with CMOS as well as PTL logic.

CHAPTER 3
PRESENT WORK

3.1 Inference of Report

In this report, the research work has been presented for the VLSI design of the Reed-Solomon codes which were having very significant applications in deep space telecommunications. The authors in various papers shows a traverse approach to system design, through dealings among the algorithm design and core architecture and circuit accomplishment, can capitulate the most significant upgradation in design intricacy.

At the ASD algorithm level, LCC is the propose subsection that comes after various algorithms which has been applied on the RS codes for decoding, S_i like BMA, Euclid and various erasures techniques. At VLSI level, the number of MOS chips embedded in design of system with complexity, power and area considerations are the compensation of VLSI decoding systems with all these speed can be raised with the advancement in CMOS technology. The data rates ascend from 4Mbits/s to 1.28Gbits/s

Some authors support their work by showing the Verilog implementation for system design and C coding to obtain the results of various mathematical concepts. The Cadence virtuoso designing of the RS codes decoding system at various CMOS technology level gives various result of improvement in the field of complexity reduction. The optimization of transistor or MOS chips is the major concern in the growing technology in the design of various decoding systems with applied algorithms.

The CD Player is only a startup application. The commercial market/world is apt for rising mobile, while on the same side demanding reliable, fast access to sales, marketing, also accounting information. Unluckily the mobile channel is revolting environment for communication; with deep-fades has an eternal existence. RS codes are the only best solution; no any other error control technique can match its reliability performance. The optical channel gives another group of problems. Shot noise, dispersive, noisy medium line of sight systems, generating noise bursts are best taken care off by RS codes. As we can see optical fibers in high-speed multiprocessors, so RS codes are also worked here. In more advanced technology percept, occasional deep-space probe, RS codes will keep on to be working for force communication systems.

3.2 Problem Statement

The Parameters such as transistor count, power consumption, delay and power-delay product are need to be analyzed in Cadence Virtuoso using 45nm technology with three logic styles such as CMOS, PTL and GDI for the various segments of RS decoder separately and unite all to make a SOC of RS decoder block. The sub- Architecture blocks of LCC based RS decoder are shown in the following section below.

3.2.1 Erasure Architecture

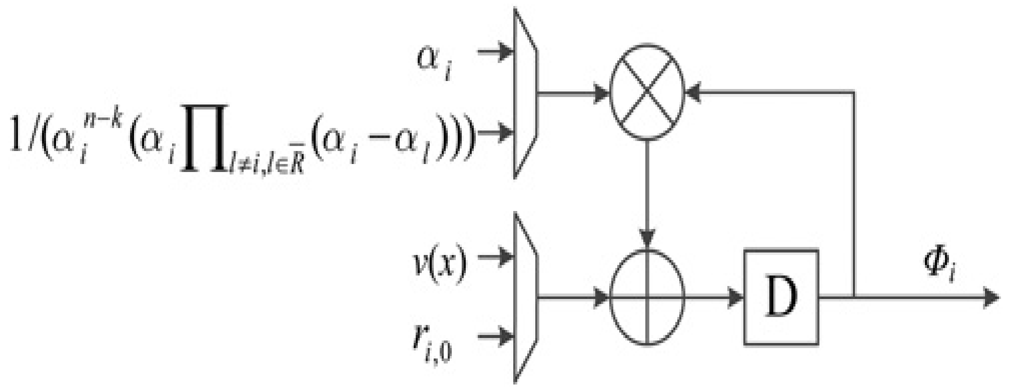

α_i
$1/(\alpha_i^{n-k}(\alpha_i\prod_{l\neq i,l\in\bar{R}}(\alpha_i-\alpha_l)))$
$v(x)$
$r_{i,0}$
D
Φ_i

Fig.3.1 Block diagram of Erasure Magnitude Computation [13]

3.2.2 Syndrome Computation Architecture

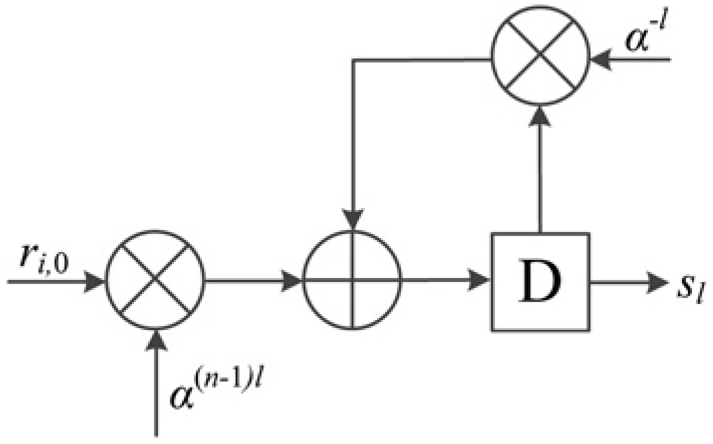


Fig.3.2. Block Diagram of Syndrome Computation [13]

3.2.3 Inverse Computation Architecture

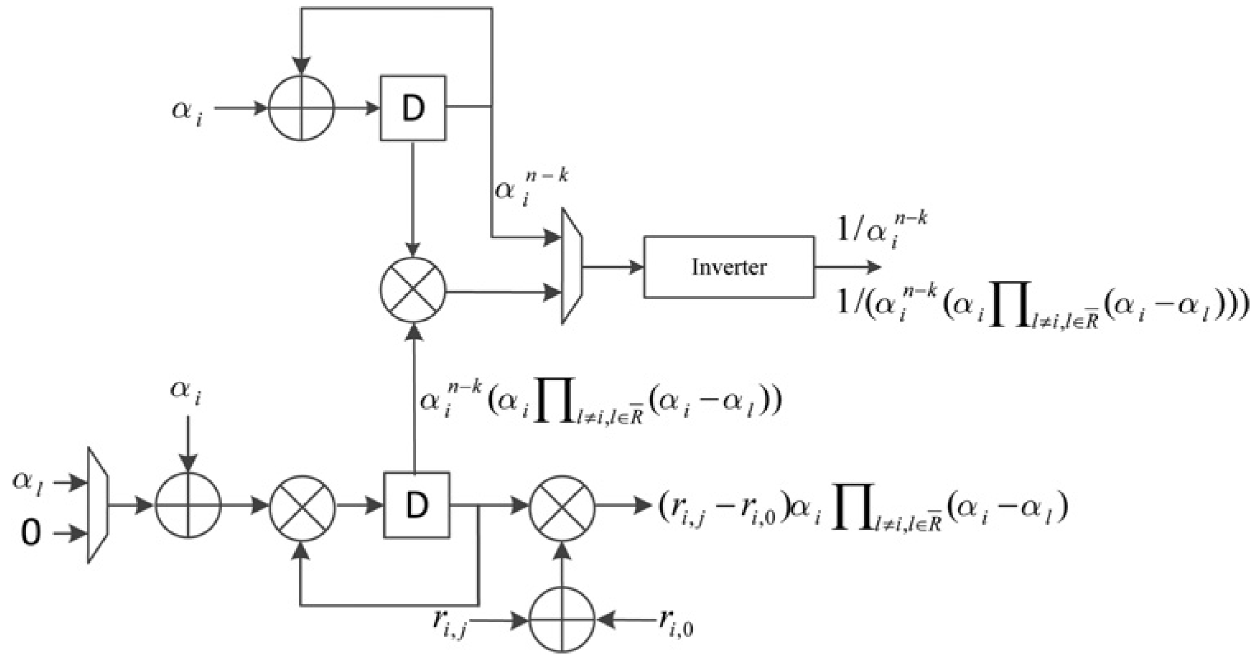

α_i
D
α_i^{n-k}
Inverter
$1/\alpha_i^{n-k}$
$1/(\alpha_i^{n-k}(\alpha_i\prod_{l\neq i,l\in\bar{R}}(\alpha_i-\alpha_l)))$
α_i
$\alpha_i^{n-k}(\alpha_i\prod_{l\neq i,l\in\bar{R}}(\alpha_i-\alpha_l))$
α_l
0
D
$(r_{i,j}-r_{i,0})\alpha_i\prod_{l\neq i,l\in\bar{R}}(\alpha_i-\alpha_l)$
$r_{i,j}$
$r_{i,0}$

Fig.3.3. Block Diagram of Inverse Computation [13]

3.2.4 Polynomial Selection Architecture

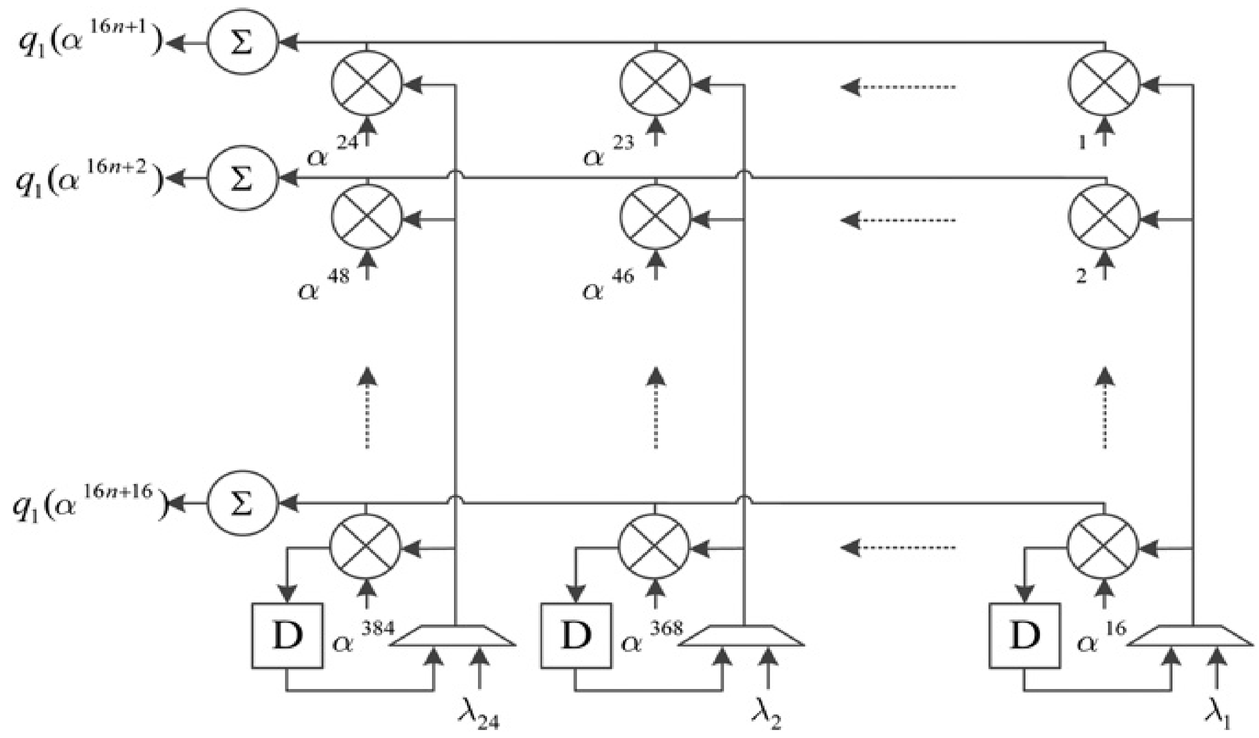


Fig.3.4. Block Diagram of Polynomial Selection [13]

3.2.5 Error Computation Architecture

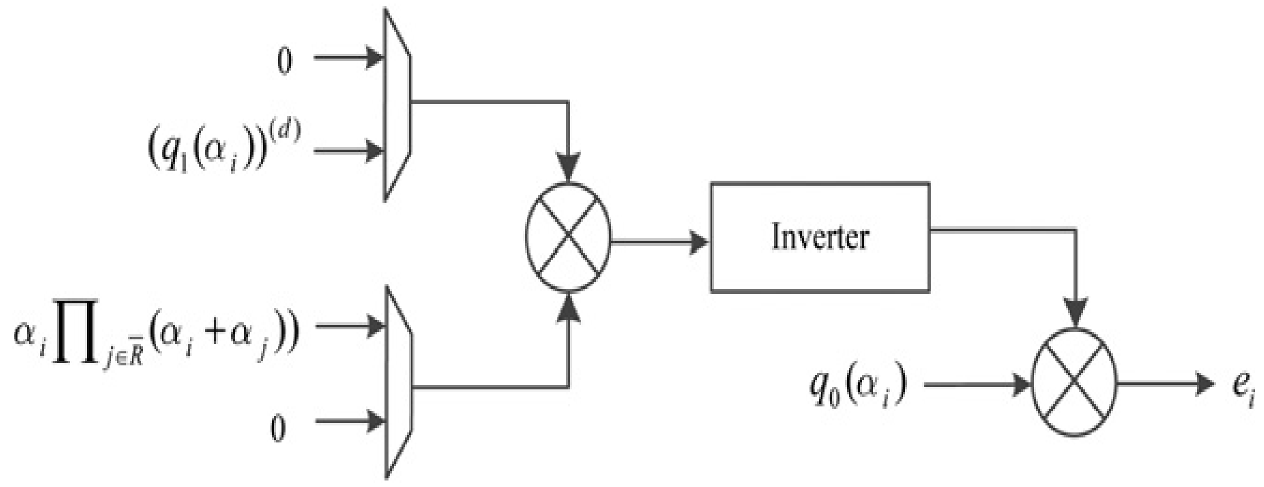


Fig.3.5 Block diagram for Error e_i Computation [13]

3.3 Constraint Used

The constraint which defines the system model used in RS code can be express with the following constraints and notation [4]:

m → the number of bits/symbol which lies in the range $3 \le m \le 16$.

n → the number of symbols/codeword which lies in range $3 \le n \le 2^{\wedge}(m-1)$.

p → the number of error correctable capability symbol errors.

w → the number of words used by the Reed Solomon Encoder to encode before transmission the data sequences.

2p → the number of parity check symbols in the transmitted codeword.

k → the number of symbols/message in the transmitted codeword .

C(x) → the illustration of code block of the range of N-1 polynomial.

R(x) → the received polynomial received at the receiver.

T(x) → the transmitted polynomial at the transmitter.

G(x) → the generator polynomial.

S(x) → the Syndrome Polynomial.

3.4 ASD Decoding Algorithm

It takes into account an RS (n, k) code which would create over GF (2^q). It is executed in three steps namely multiplicity assignment, interpolation and factorization. The function of multiplicity assignment was to decide the interpolation points and interpolation locate a bivariate polynomial $Q(x, y)$ having minimum (1, k-1) weighted degree crossing each interpolation point. After that factorization calculates all factors for $Q(x, y)$ which is represented in the form of $y - f(x)$. The implementation process for ASD decoder is shown in Fig.5.1. [16].

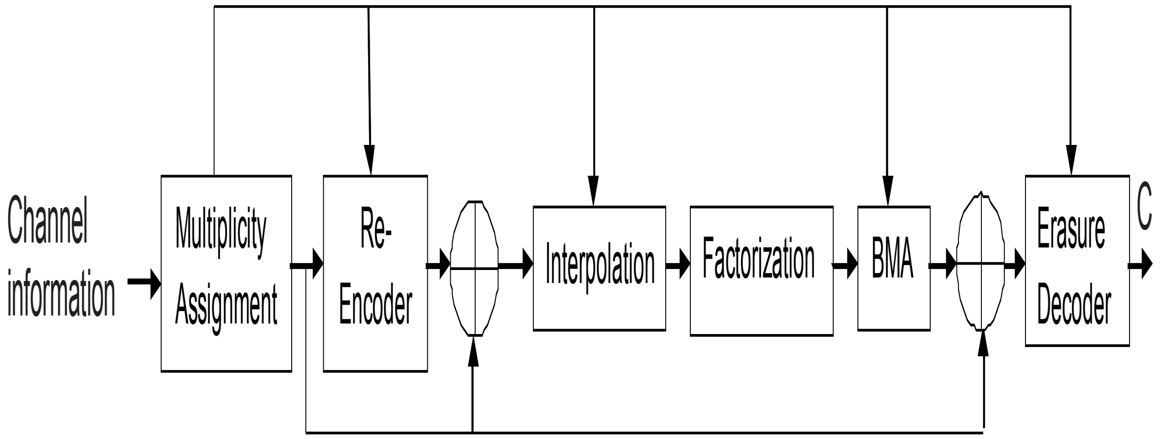


Fig.3.6. ASD Decoder Block

In Fig.5.1, the re-encoder is designed to locate the codeword ϕ which has the similar symbols as in received word r in most consistent k code position, which constituent a set R. Coordinate transformation also applies to the interpolation points. Accordingly, the decoding can be applied on $\phi + r$ and those points which have code locations in R was consider for interpolation process [16]. The outcome of factorization process used as syndromes in Berlekamp – Massey algorithm (BMA) for retrieving of errors in code position in R. Then, a new erasure decoding is making functional for recovering of complete codeword [16].

3.5 Re-Encoded LCC Decoding Technique

The LCC algorithm of ASD has multiplicity assignment one of its feature where (α_j, β_j) and $(\alpha_j, \beta_j^{'})$ are the points which are allocate to each of η least consistent code position. In this field element is represented as α_j which was encoded in evaluation map encoding and the hard-decision symbol is shown by β_j whereas $\beta_j^{'}$ was second jth most expected symbol for code position. The polynomial $Q(x, y)$ which was found through interpolation can be solved through $k^{\cdot\cdot}$ algorithm [17], which initiate with $Q^{(0)}(x, y) = 1$ and $Q^{(1)}(x, y) = y$ for LCC decoding having high rate codes. The Block diagram of LCC decoder is shown in Fig. 3.7 [17].

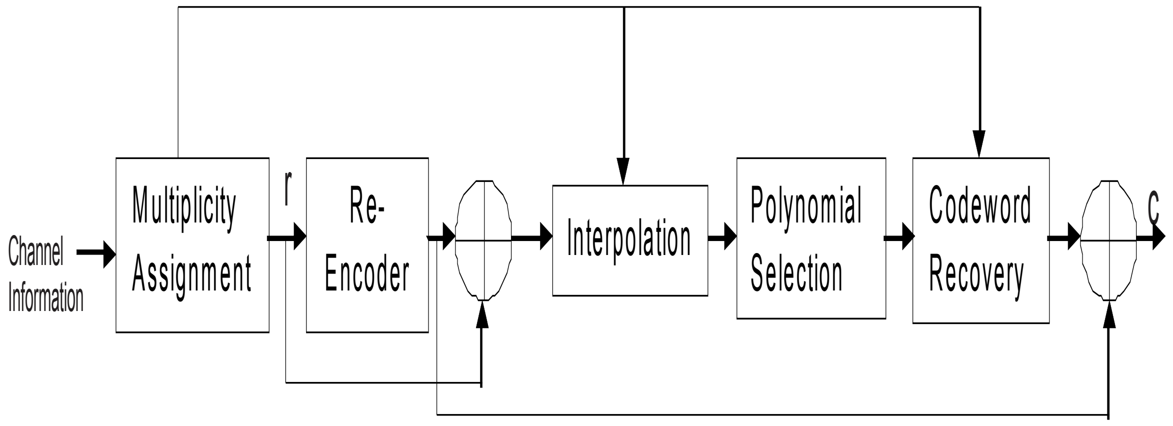


Fig.3.7. Block Diagram of Re-Encoded LCC Decoder

Now, developing a $Gr\ddot{}$ basis by allowing iteratively updating of two polynomials so that it can pass an additional point at a time. Then the polynomial found to be lowest weighted degree was considered to be the least weighted degree among all the present polynomial. Hence, lowest weighted degree polynomial from the most recent iteration was the required interpolation output. Then factorization determines all the factors related to $Q(x, y)$ in the form such as $y - f(x)$ with constituent degree of $(f(x)) < k$ and each $f(x)$ in the list are equivalent to a message polynomial [17].

3.6 Transformed LCC Decoding with Re-Encoding

In order to make simpler the interpolation in algebraic soft decision (ASD), the re-encoding and coordinate transformation algorithm can be applied on it. Let indicate the arrangement of majority of k reliable code position by R in r. The erasure decoding applied in re-encoding to obtain the codeword ϕ. Now, the decoding is carried on [18]:

$$\overline{r} = r + \phi \tag{1}$$

Let us assume the error vector e is being adjoin with the codeword and it can be represented as [18]:

$$r = c + e \tag{2}$$

Also, we can obtain another codeword using the similar error vector which can be represented as

[18]:

$$\overline{r} = c + \phi + e = \overline{c} + e \qquad (3)$$

In addition for $i \in R$,

$$\overline{r}_i = r_i + \phi_i = 0 \qquad (4)$$

So the interpolation process can be applied on various code positions which were available in R and it can be applied on rest of the code position which was accessible only in $n-k$ code position. Therefore, the polynomials which were pre-computed and factor of these polynomial can be computed as [18]:

$$v(x) = \Pi_{i \in R}(x + \alpha_i) \qquad (5)$$

The length of polynomial can be further decrease up to k by taken out the factor using the process of coordinate transformation.

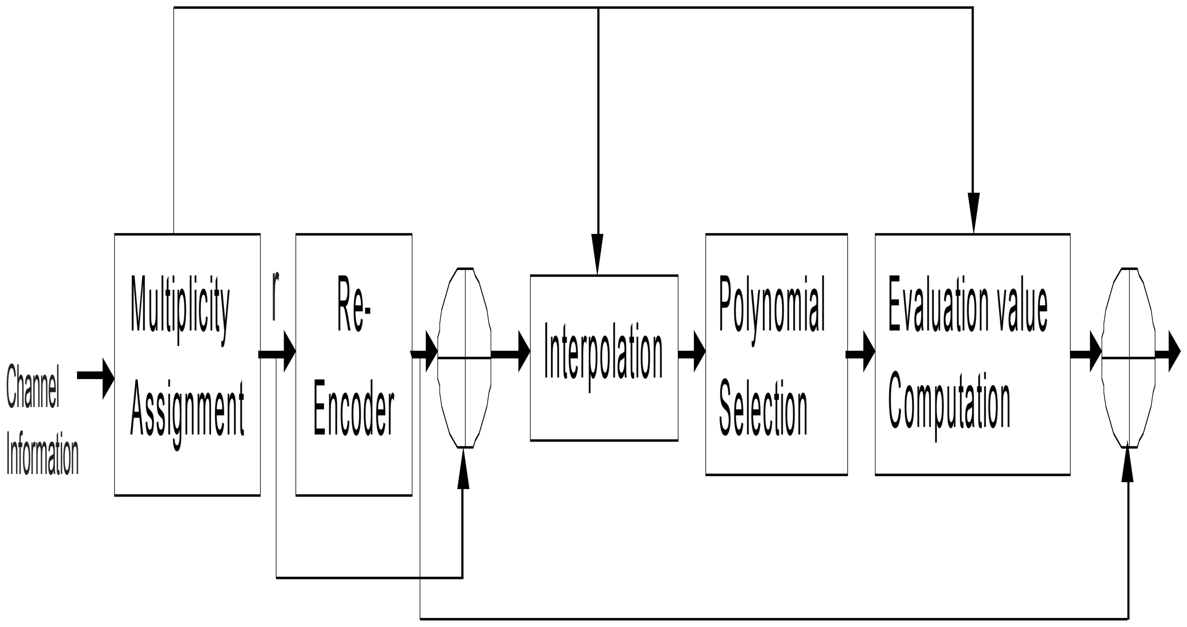


Fig.3.8. Block Diagram of transformed LCC Decoding

One approach can be applied using $q_1(x)$ and $q_0(x)$ such that errors can be locate in code position of R while using Chien search and Forney's algorithm. After the errors in R are rectified so to avoid the complexity originate from factorization, the erasure decoding procedure can be applied to retrieve the $n-k$ symbols in $\overline{R}$. Instead to gain the message polynomial $\overline{f}(x)$ of $\overline{c}$

proceed with the multiplication of $v(x)$ back to the interpolation output.

3.7 Pass-Transistor Logic

Primary inputs able to drive gate terminals along with source-drain terminals. It requires lower switching energy to charge up a node and it is due to reduced voltage swing.

The output terminal/node get charge from 0 to $V_{dd} - V_{th}$ while the energy require for this to charge from the power supply source is given by $C_L \cdot V_{dd}(V_{dd} - V_{th})$. When the consumed switching power is getting lower, it may try to consume static power when the output terminal is HIGH.

Since there are circuits that can be differential with complementary inputs as well as outputs are available. And these type of logics need extra circuitry to work on and those can be complex gates like XOR gates, MUX as well as ADDERS. We use Complementary Pass Transistor Logic (CPL) which is a static gate and it is because output can be connected to Vdd or Gnd with a path having low-resistance.

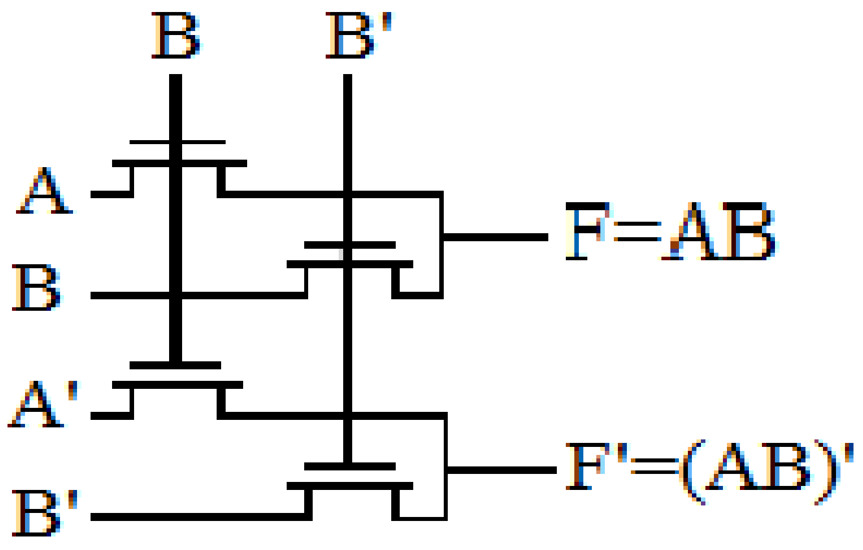


Fig.3.9. CPTL AND/ NAND gate

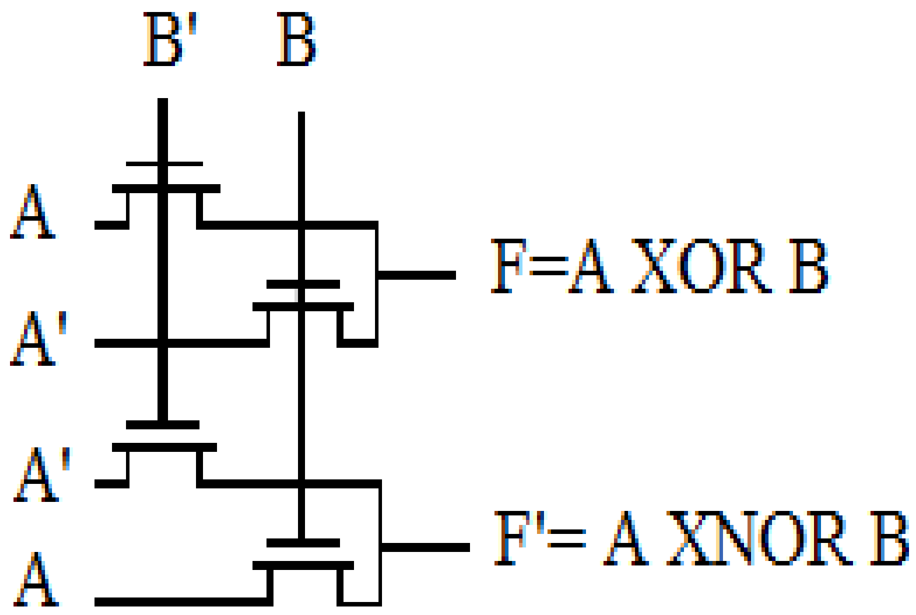


Fig.3.10. CPTL XOR/ XNOR gate

Advantage of PTL in contrast to CMOS design:

- Due to the small node capacitances, it has high speed.
- When look into account the number of transistor than the power dissipation is found to be low.
- Due to low area, the interconnection effects are also lower.

3.8 Gate-Diffusion Input (GDI) Technique

The GDI cell consists of three inputs namely:

G means common gate input of nMOS and pMOS.

P means input to source/drain of pMOS.

N means input to source/drain of nMOS.

In this the bulk of nMOS is connected to input N while bulk of pMOS is connected to input P such that it will randomly biased in contrast with the CMOS inverter [19].

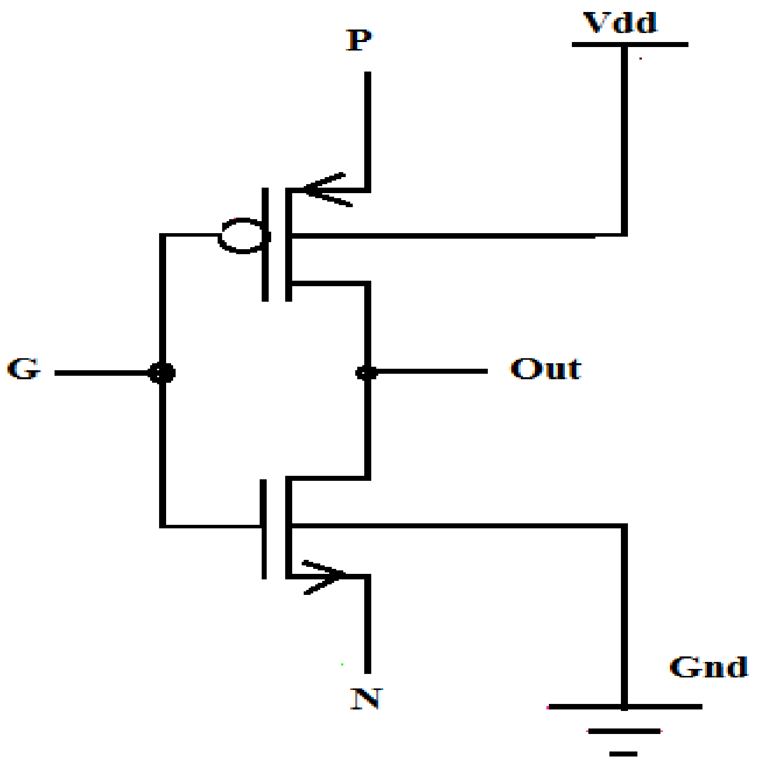


Fig.3.11. Basic GDI Cell [19]

It can be observed that a number of functions can be implemented using this GDI technique.

TABLE 3.1 Basic functions using GDI cell

N	P	G	OUT	FUNCTION
0	1	A	A’	INVERTER
0	B	A	A’B	F1
B	1	A	A’+B	F2
1	B	A	A+B	OR
B	0	A	AB	AND
C	B	A	A’B+AC	MUX
B’	B	A	A’B+B’A	XOR
B	B’	A	AB+A’B’	XNOR

As for an example, MUX design is consider as complex design needs 8-12 transistors when using conventional CMOS but that can be designed with GDI using only two transistors.

3.9 Design Using CMOS Logic Style

3.9.1 Inverter

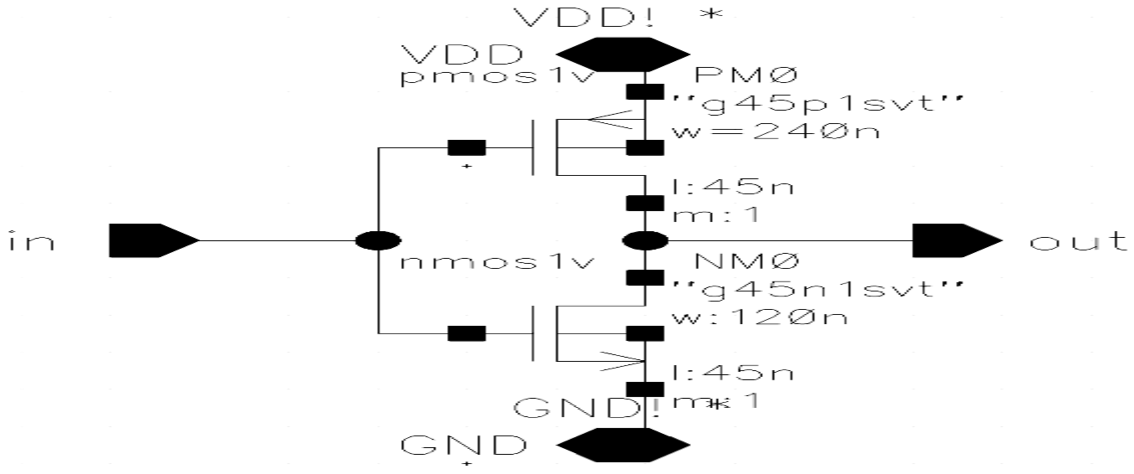


Fig.3.12 CMOS based Inverter

3.9.2 NOR gate

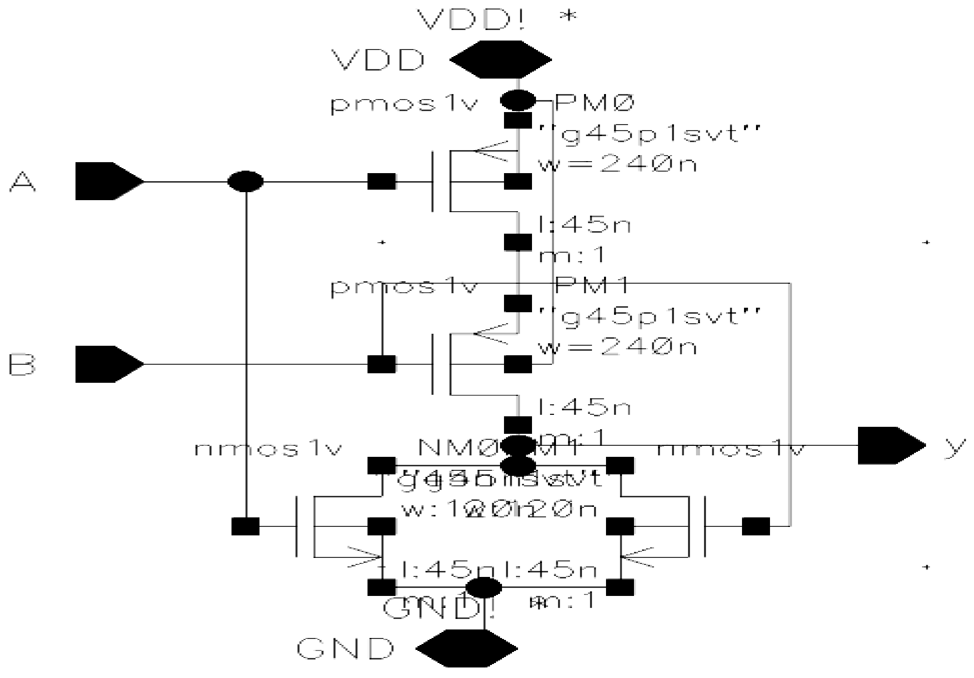


Fig.3.13 CMOS based NOR gate

3.9.3 NAND gate

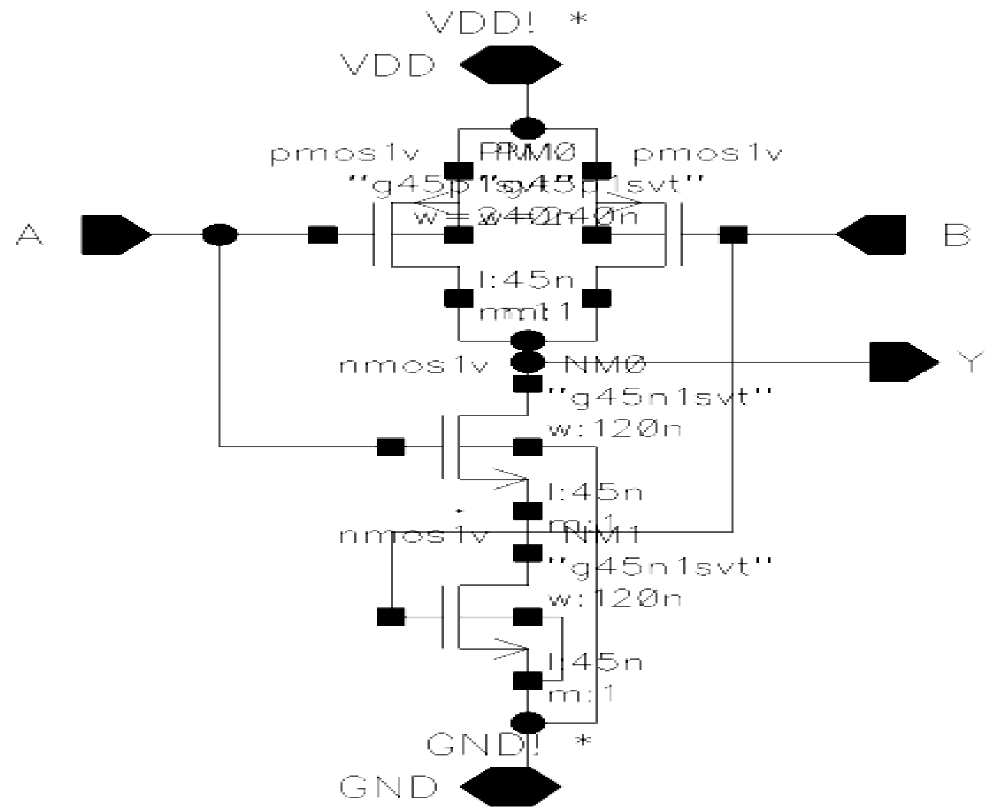


Fig.3.14 CMOS based NAND gate

3.9.4 AND gate

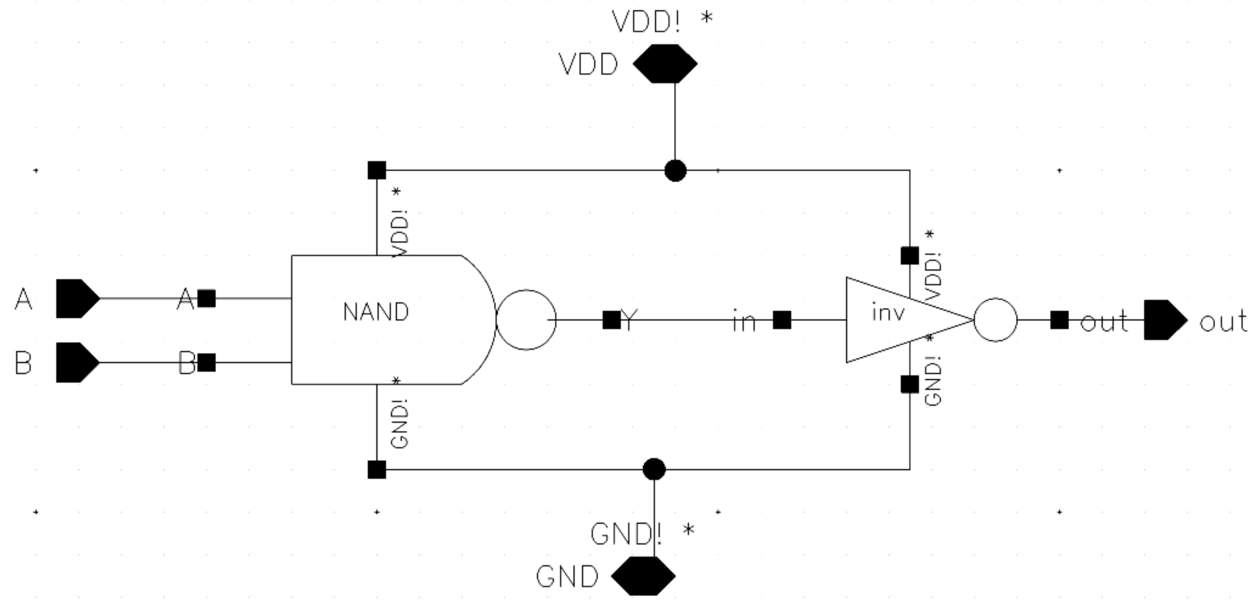


Fig.3.15 CMOS based AND gate

3.9.5 XOR gate

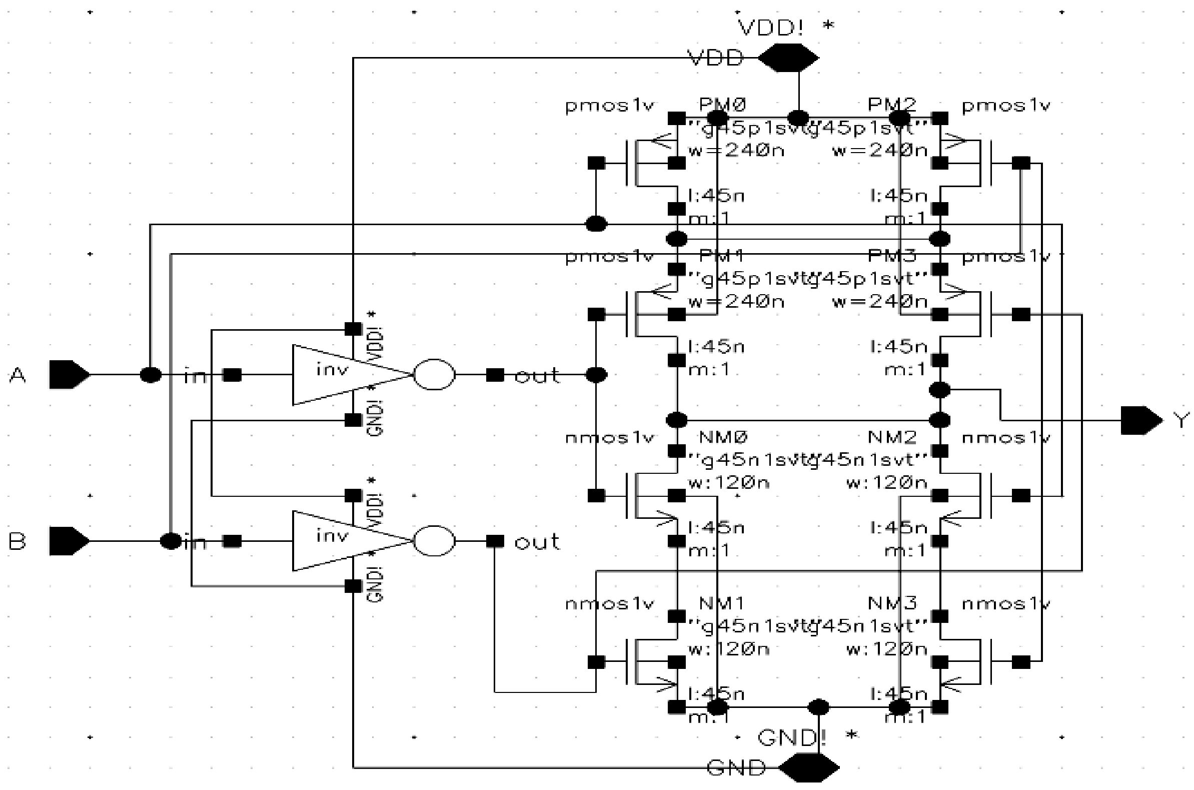


Fig.3.16 CMOS based XOR gate

3.9.6 MUX 2:1

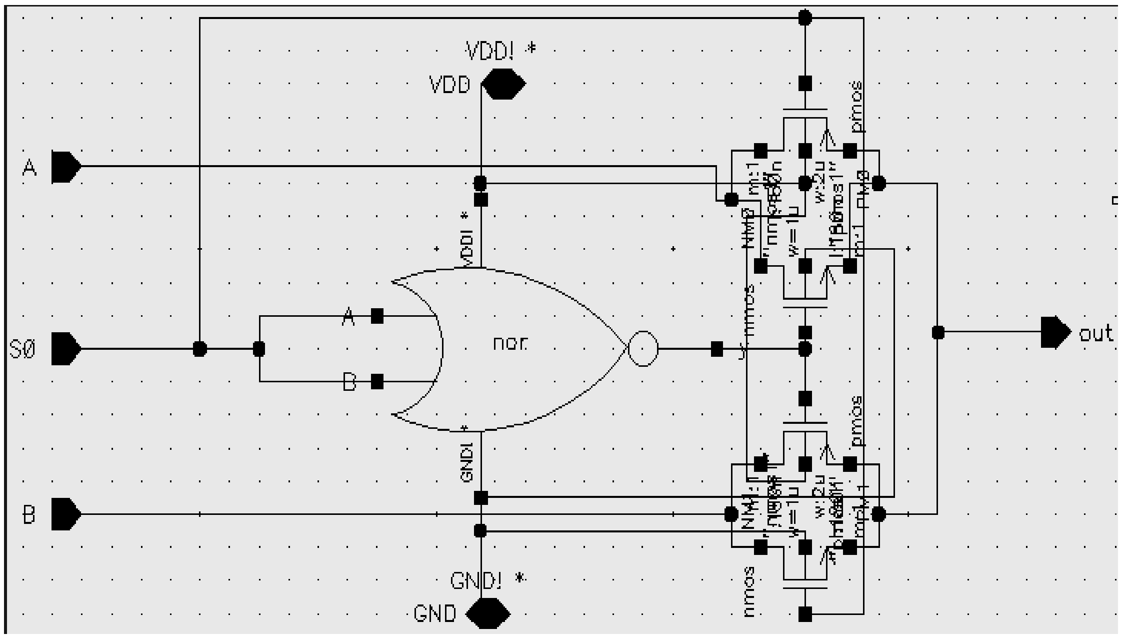


Fig.3.17 CMOS based MUX 2:1

3.10 Design Using PTL Logic Style

3.10.1 NOR gate

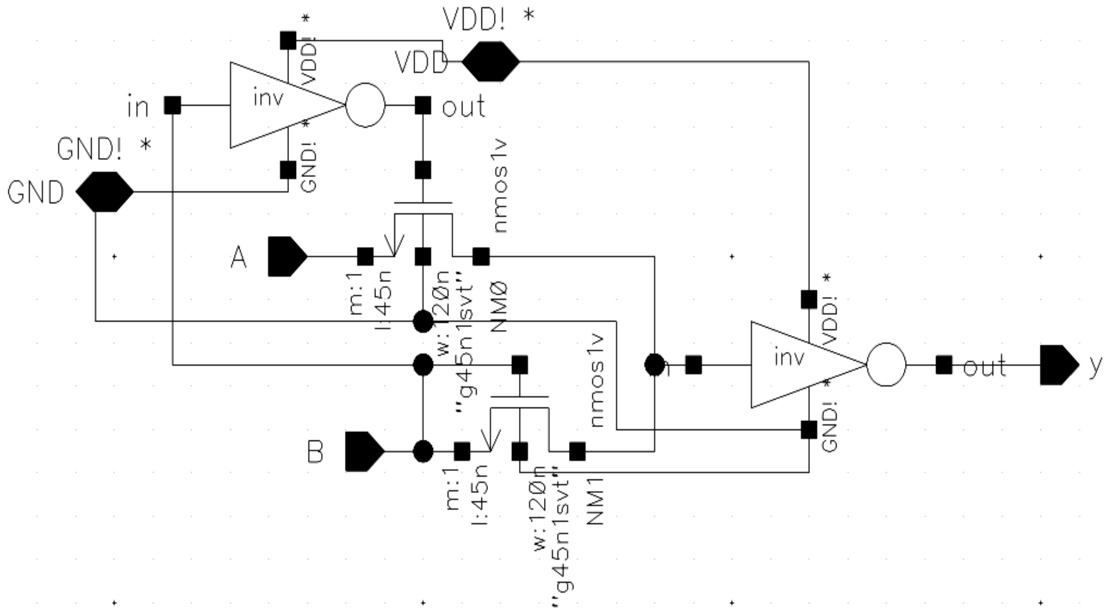


Fig.3.18 PTL based NOR gate

3.10.2 NAND gate

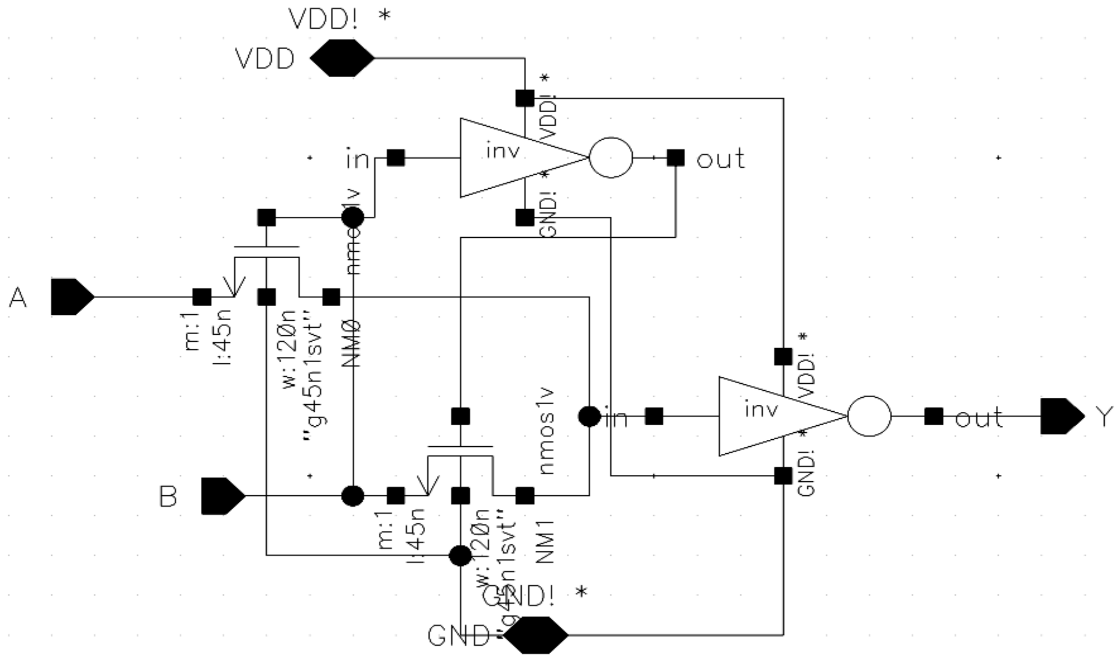


Fig.3.19 PTL based NAND gate

3.10.3 AND gate

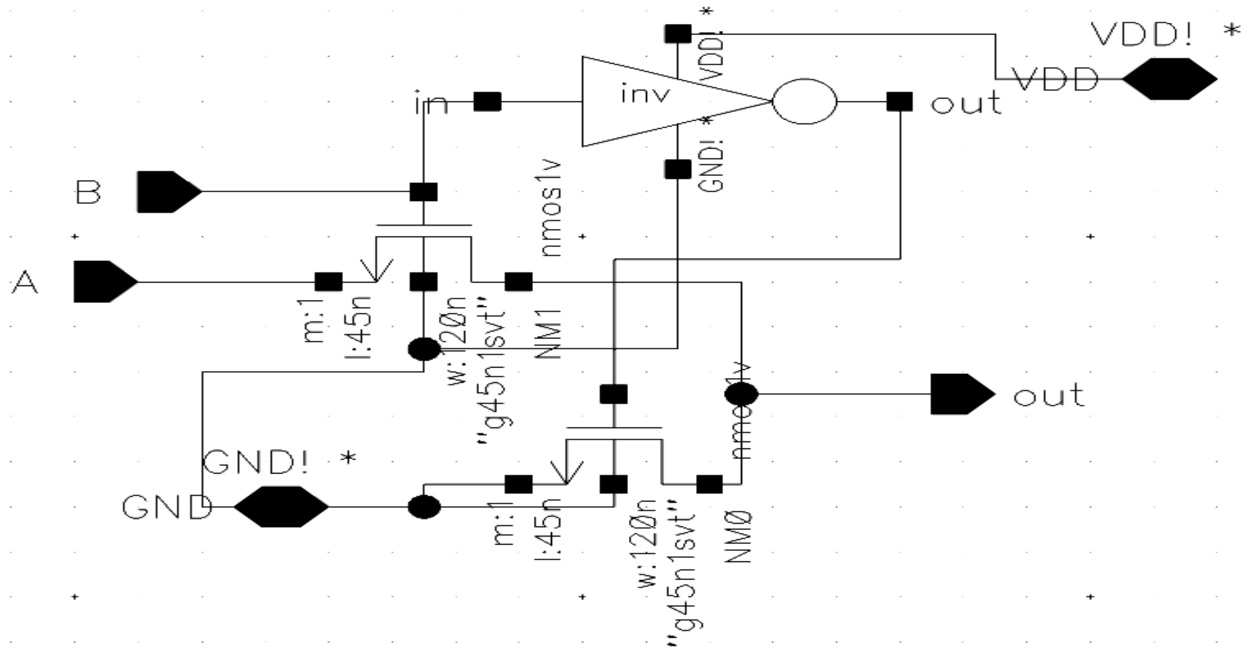


Fig.3.20 PTL based AND gate

3.10.4 XOR gate

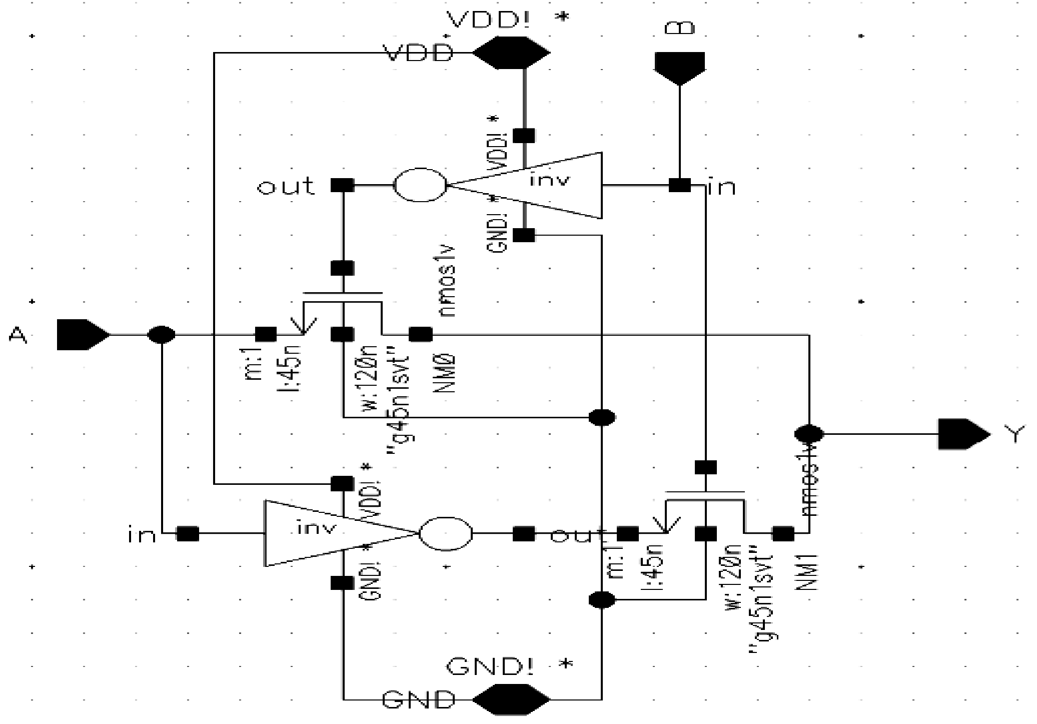


Fig.3.21 PTL based XOR gate

3.11 Design Using GDI Logic Style

3.11.1 OR gate

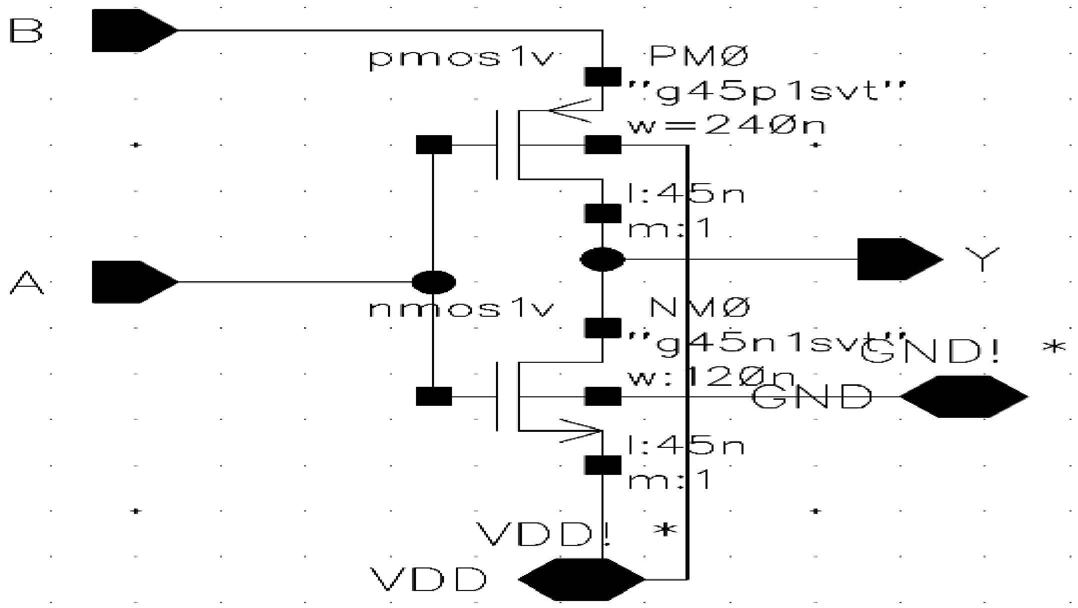


Fig.3.22 GDI based OR gate

3.11.2 AND gate

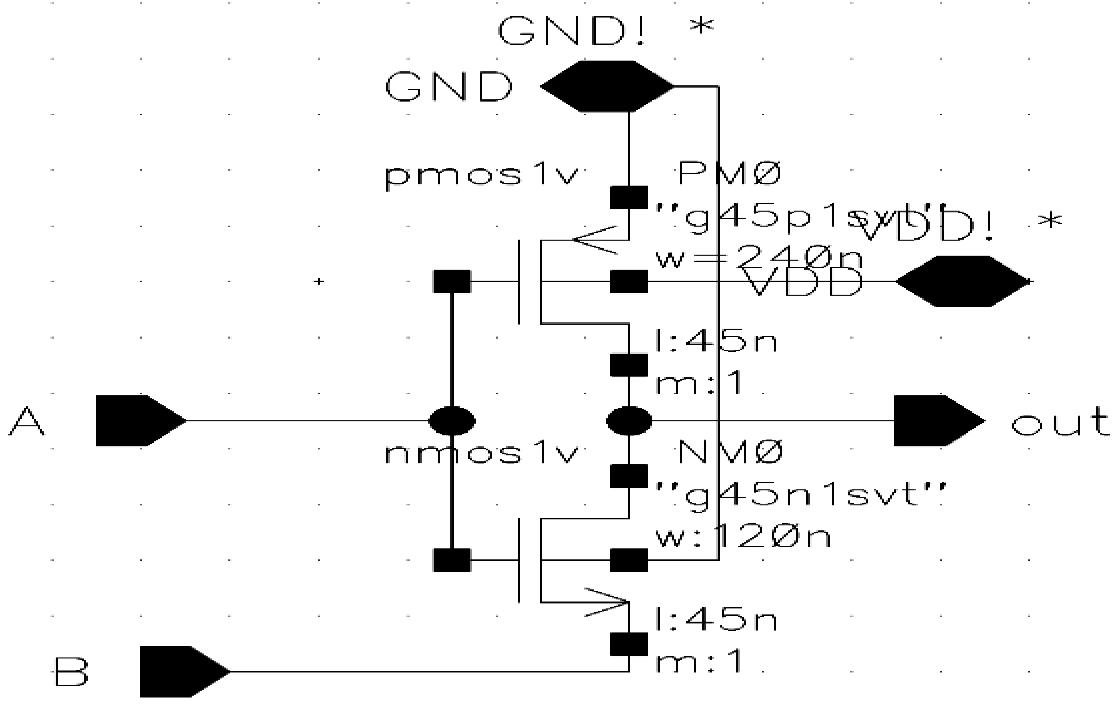


Fig.3.23 GDI based AND gate

3.11.3 NAND gate

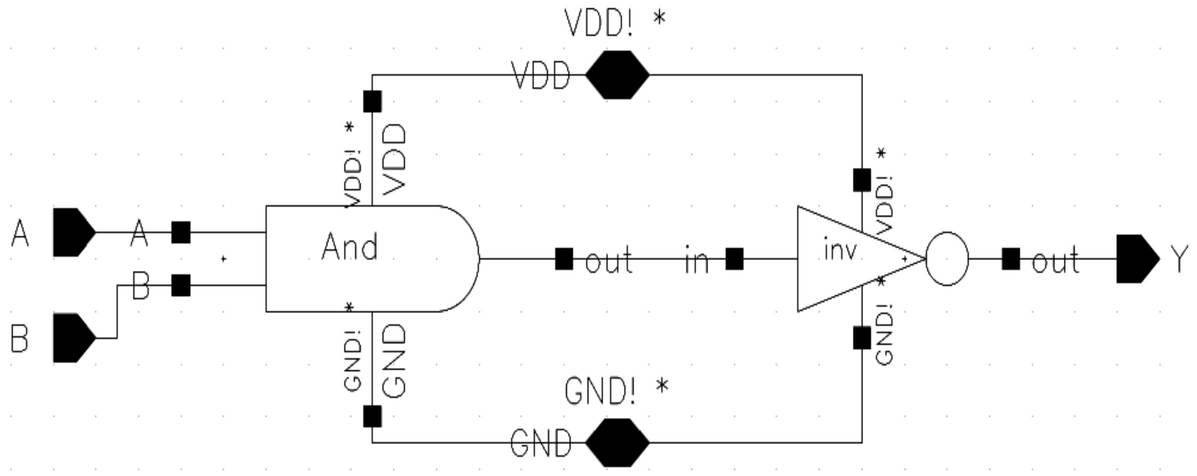


Fig.3.24 GDI based NAND gate

3.11.4 XOR gate

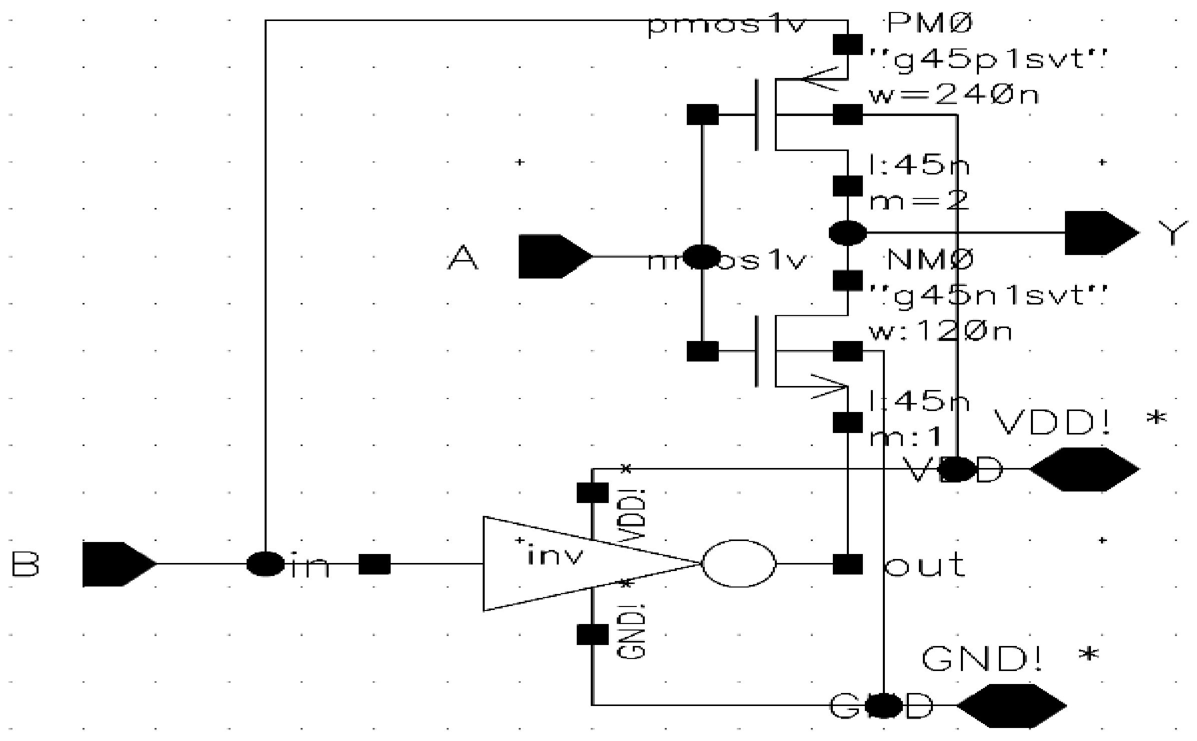


Fig.3.25 GDI based XOR gate

3.11.5 MUX 2:1

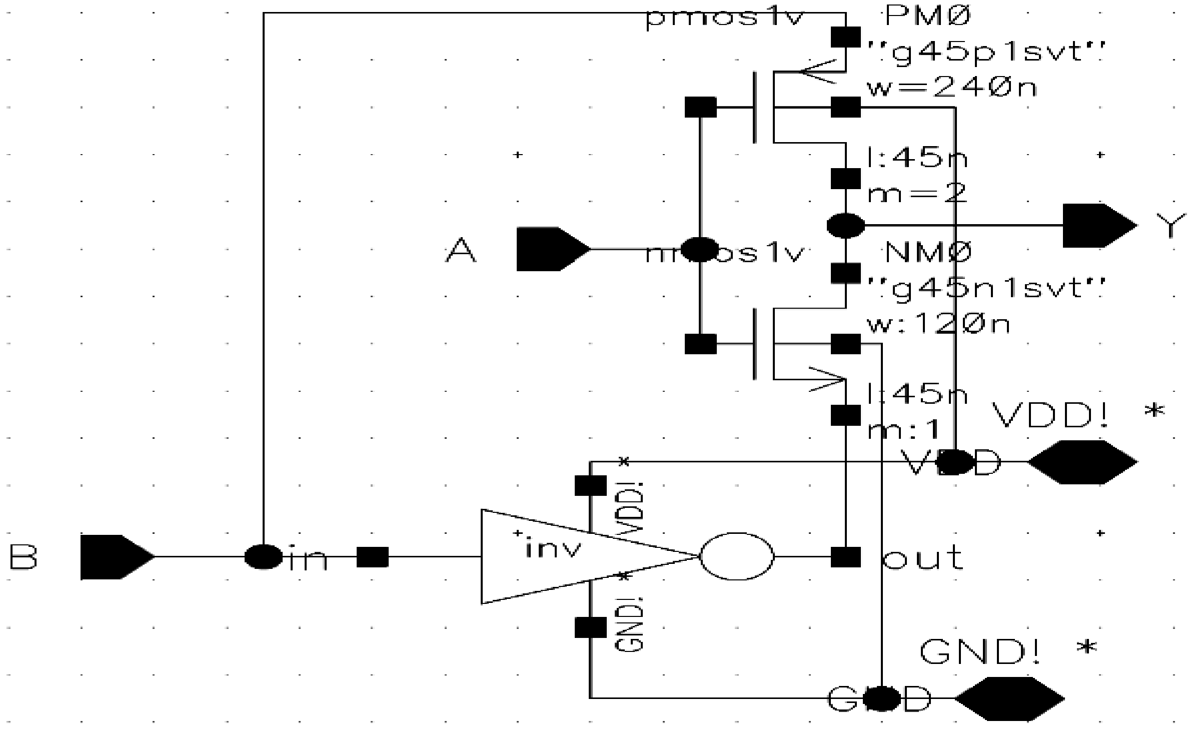


Fig.3.26 GDI based MUX 2:1

3.12 Design of Components

In this section number of components was designed using the symbols of various gates and combinational circuits of the above Logic styles.

3.12.1 1-bit Multiplier

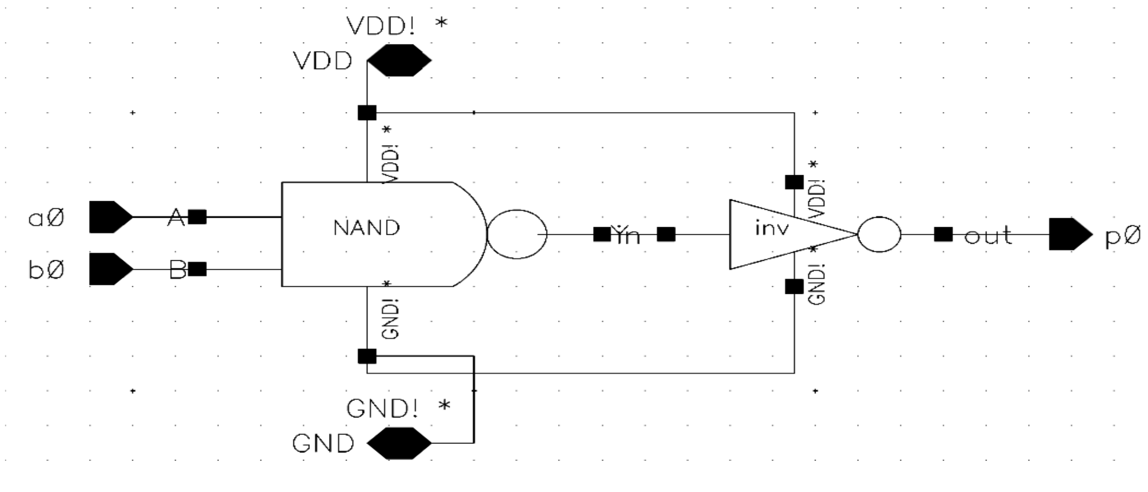


Fig.3.27 Circuit of 1-bit Multiplier

3.12.2 2-bit Multiplier

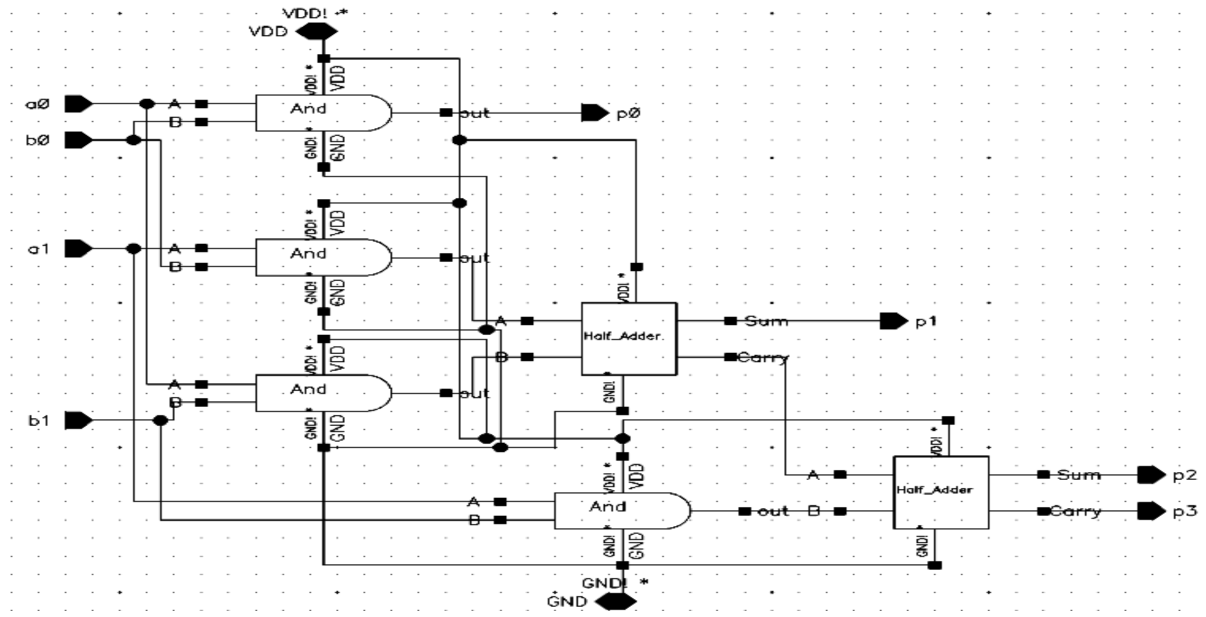


Fig.3.28 Circuit of 2-bit Multiplier

3.12.3 Half Adder

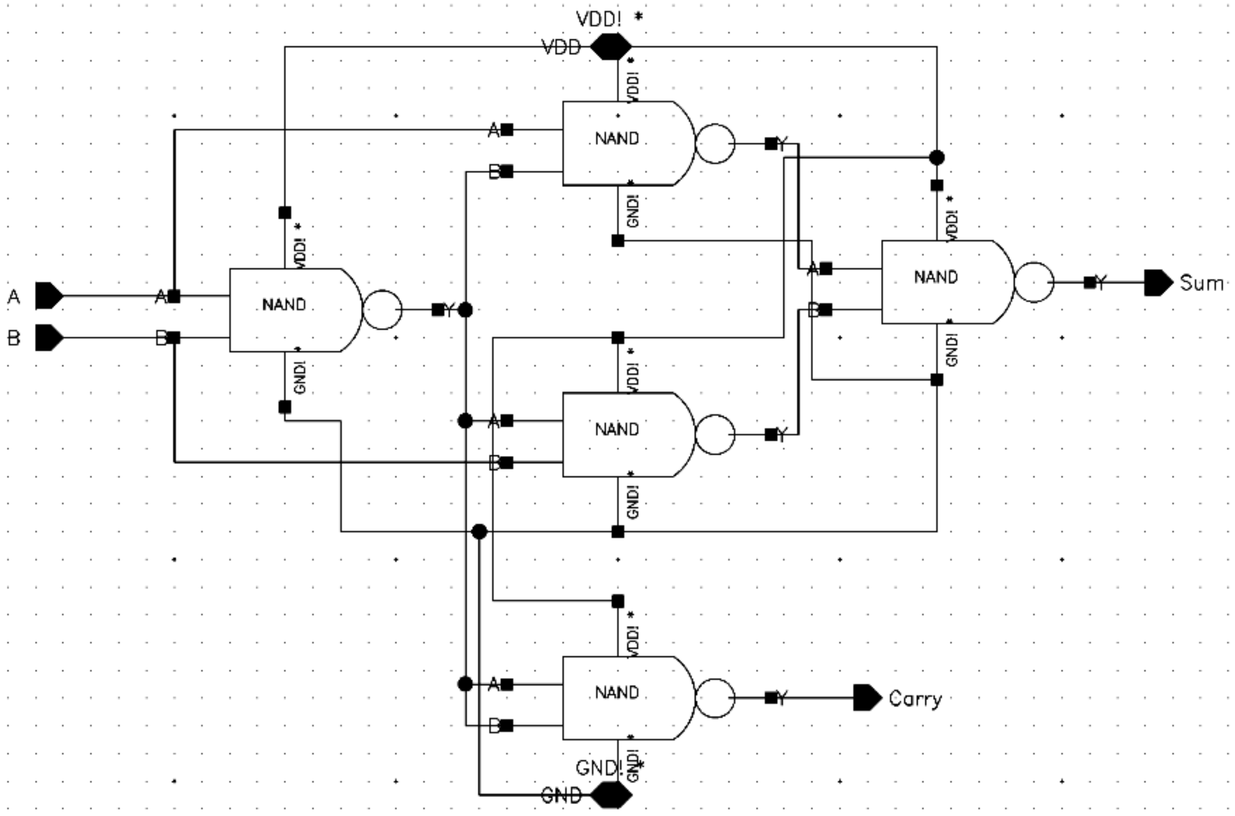


Fig.3.29 Circuit of Half Adder

3.12.4 Full Adder

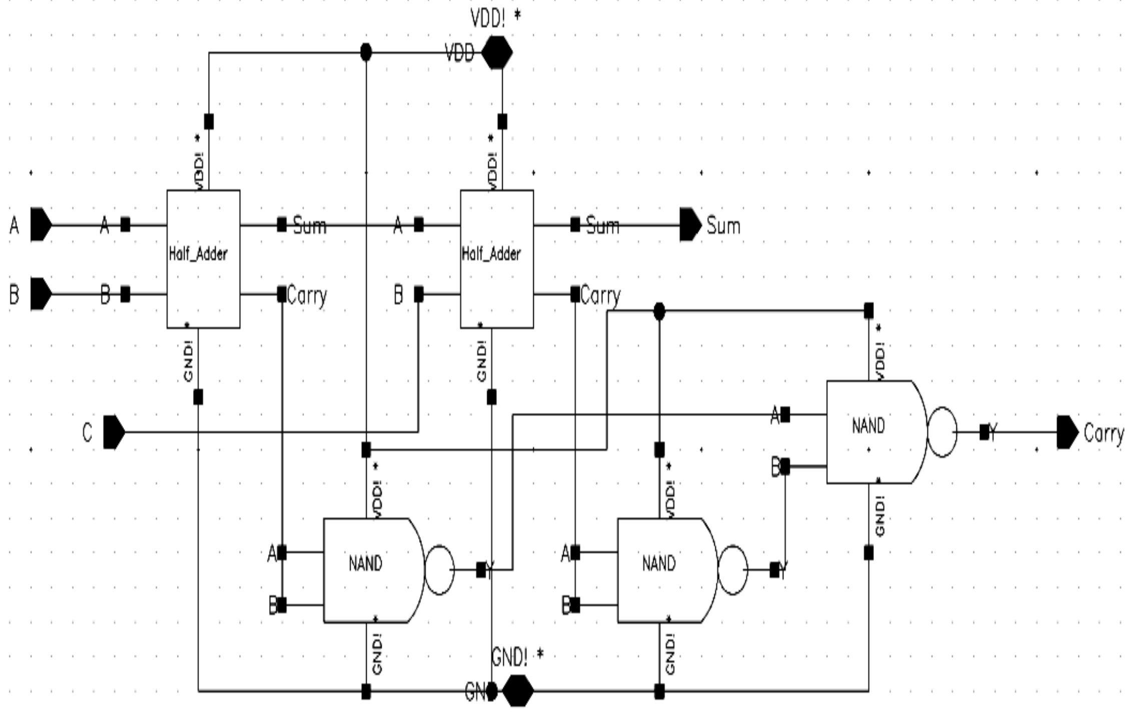


Fig.3.30 Circuit of Full Adder

3.12.5 Half Subtractor

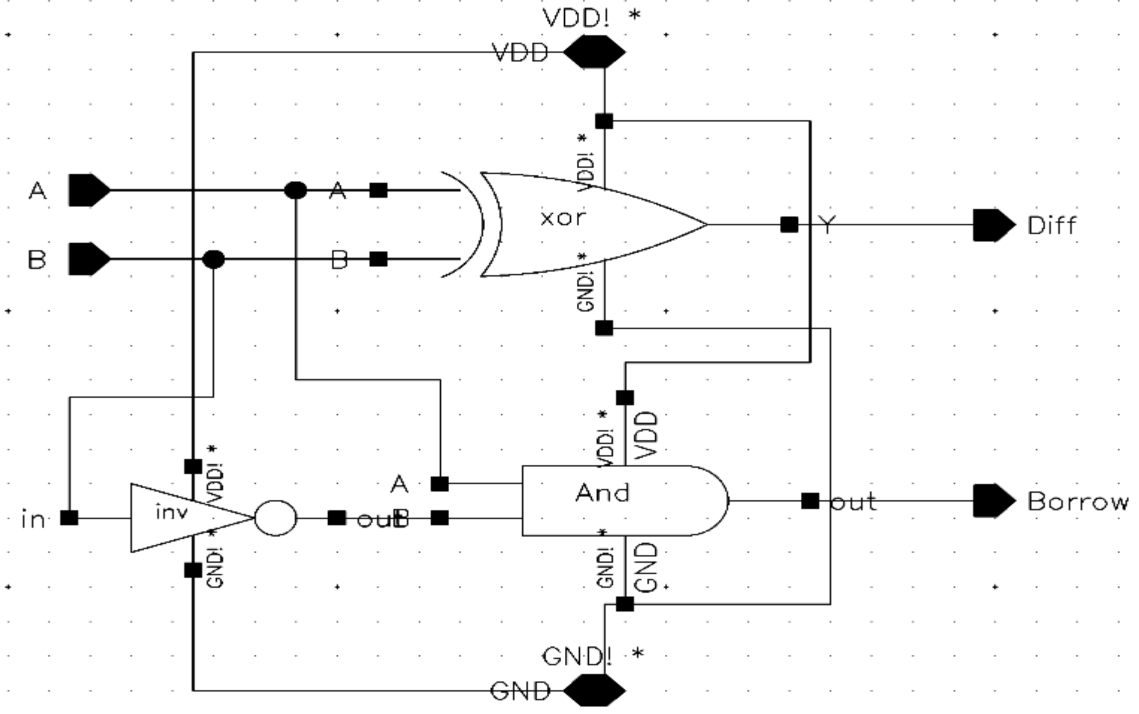


Fig.3.31 Circuit of Half Subtractor

3.12.6 Full Subtractor

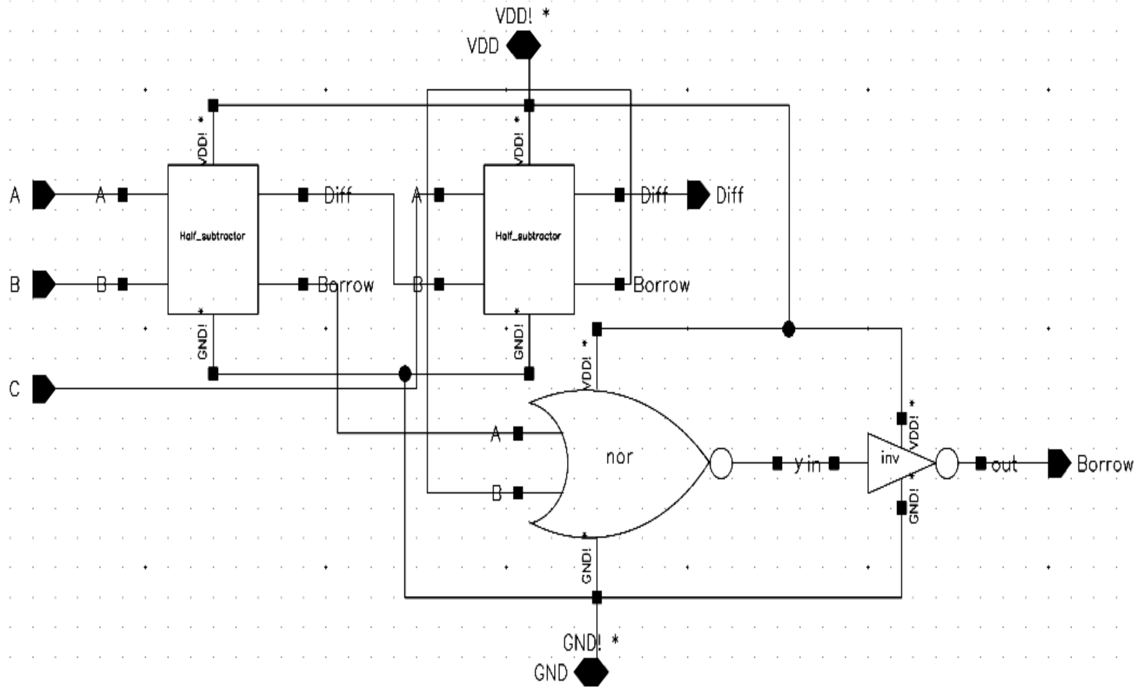


Fig.3.32 Circuit of Full Subtractor

3.12.7 D Flip-Flop

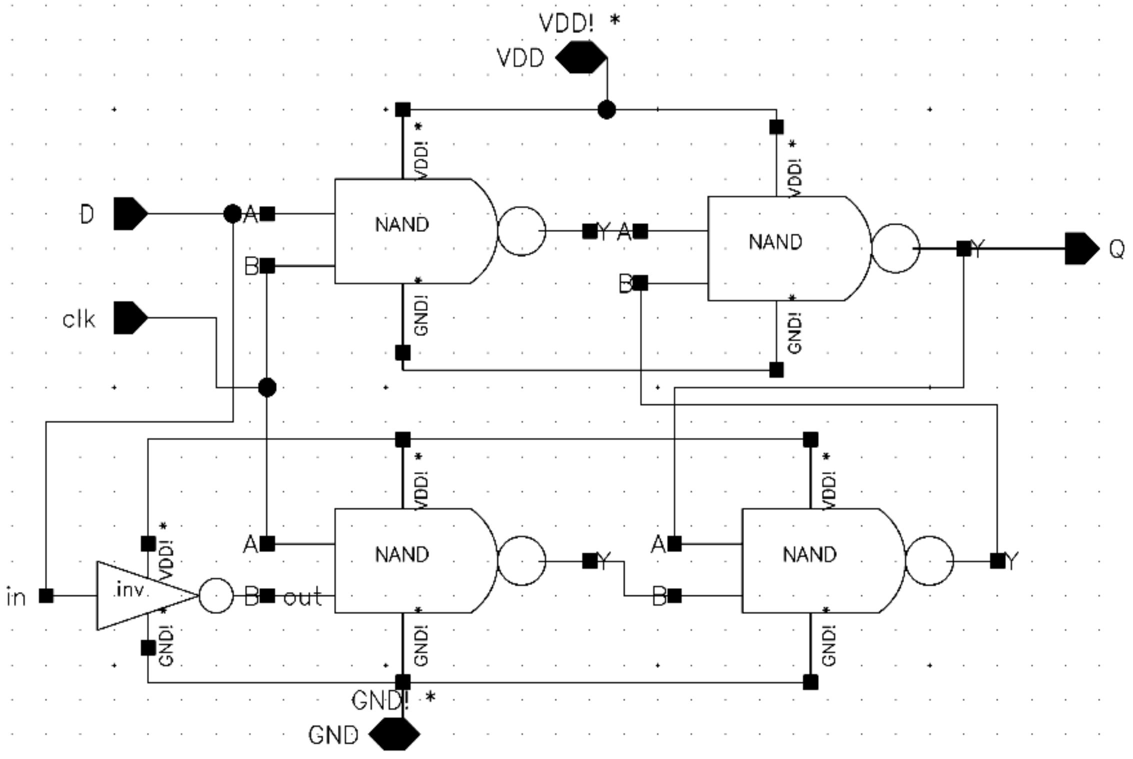


Fig.3.33 Circuit of D-FF

3.12.8 Shift Register

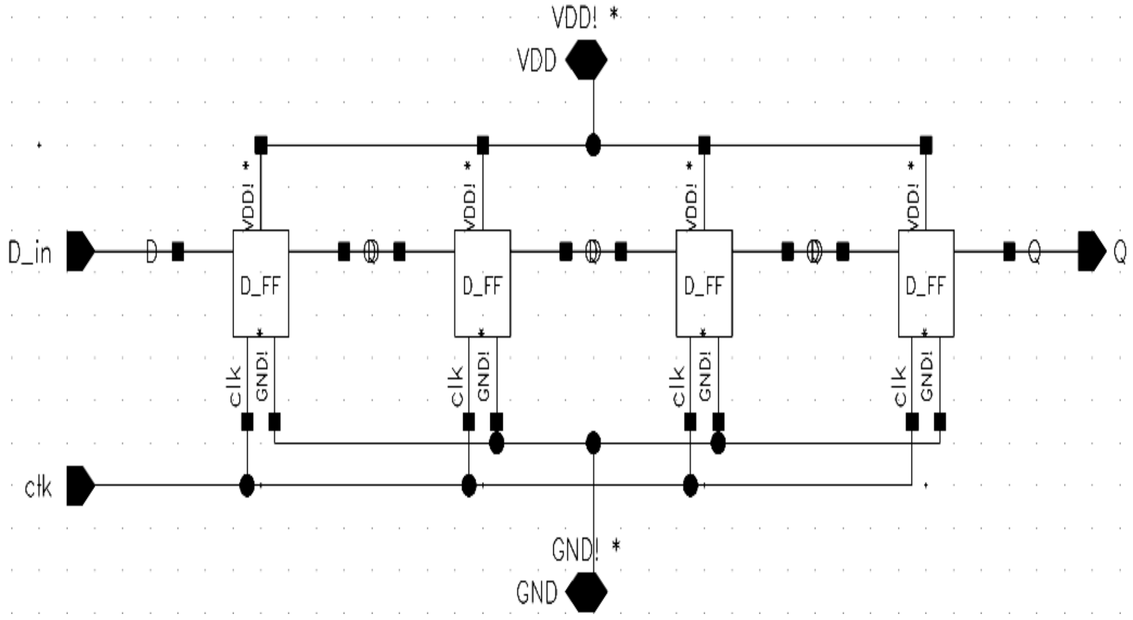


Fig.3.34 Circuit of Shift Register

3.13 Sections of Reed-Solomon Decoder

3.13.1 Syndrome Computation

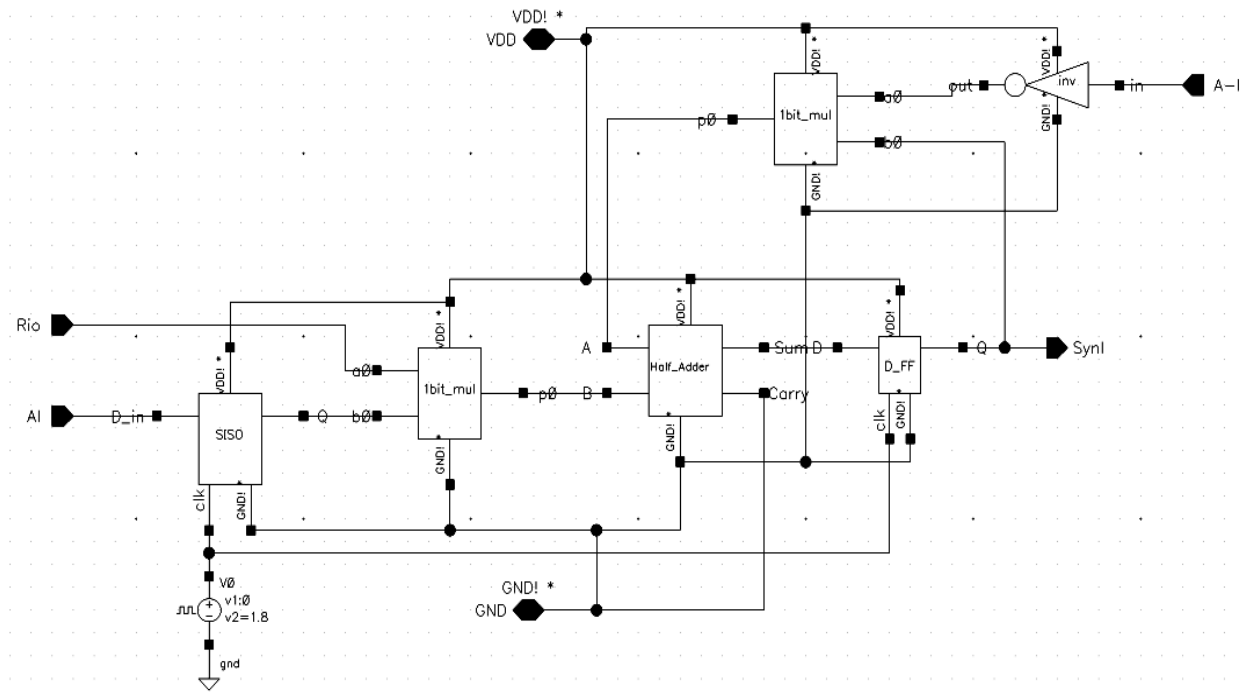


Fig.3.35 Schematic of Syndrome Computation

3.13.2 Erasure Magnitude Computation

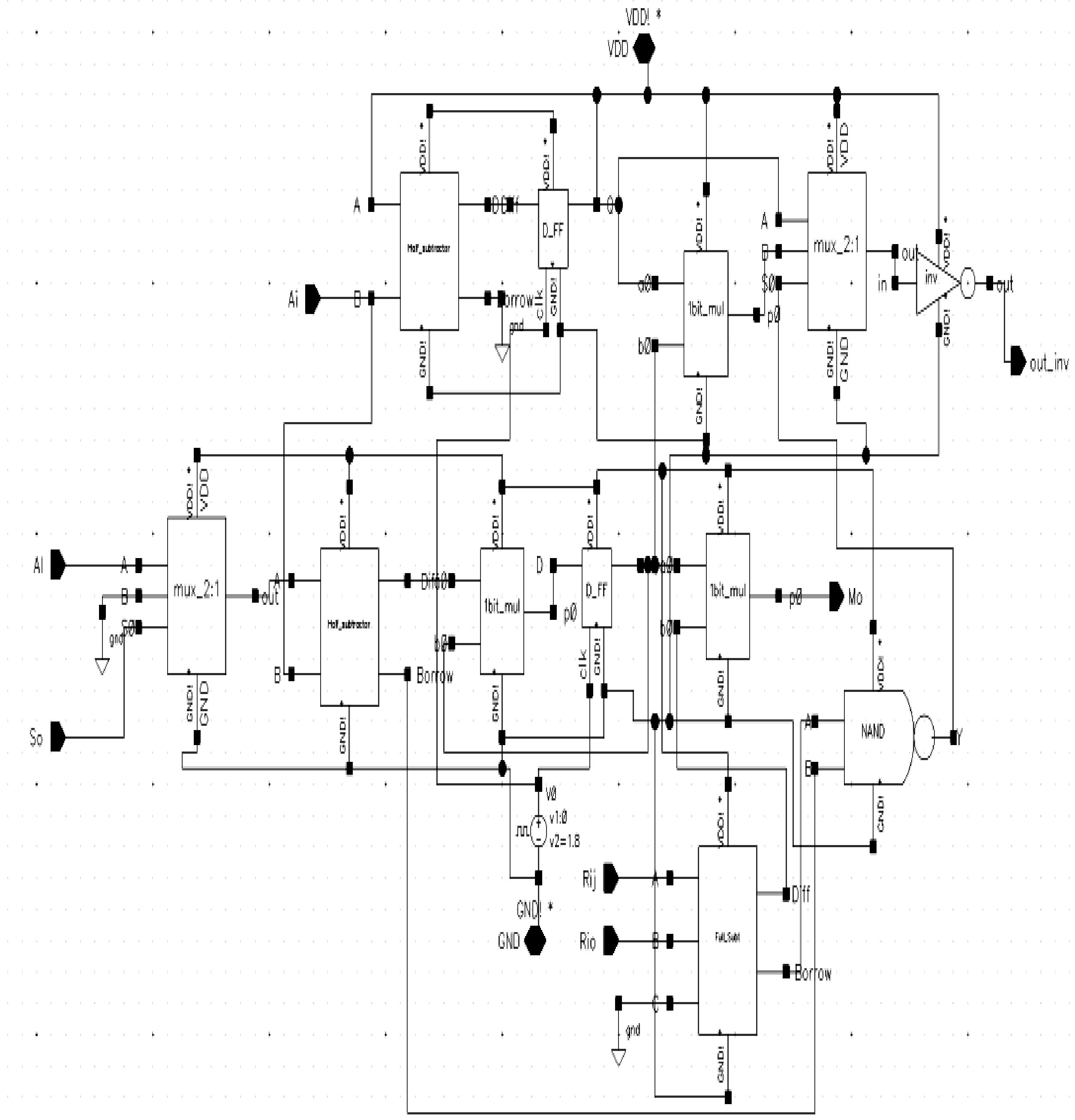


Fig.3.36 Schematic of Erasure Magnitude Computation

3.13.3 Inverse Computation

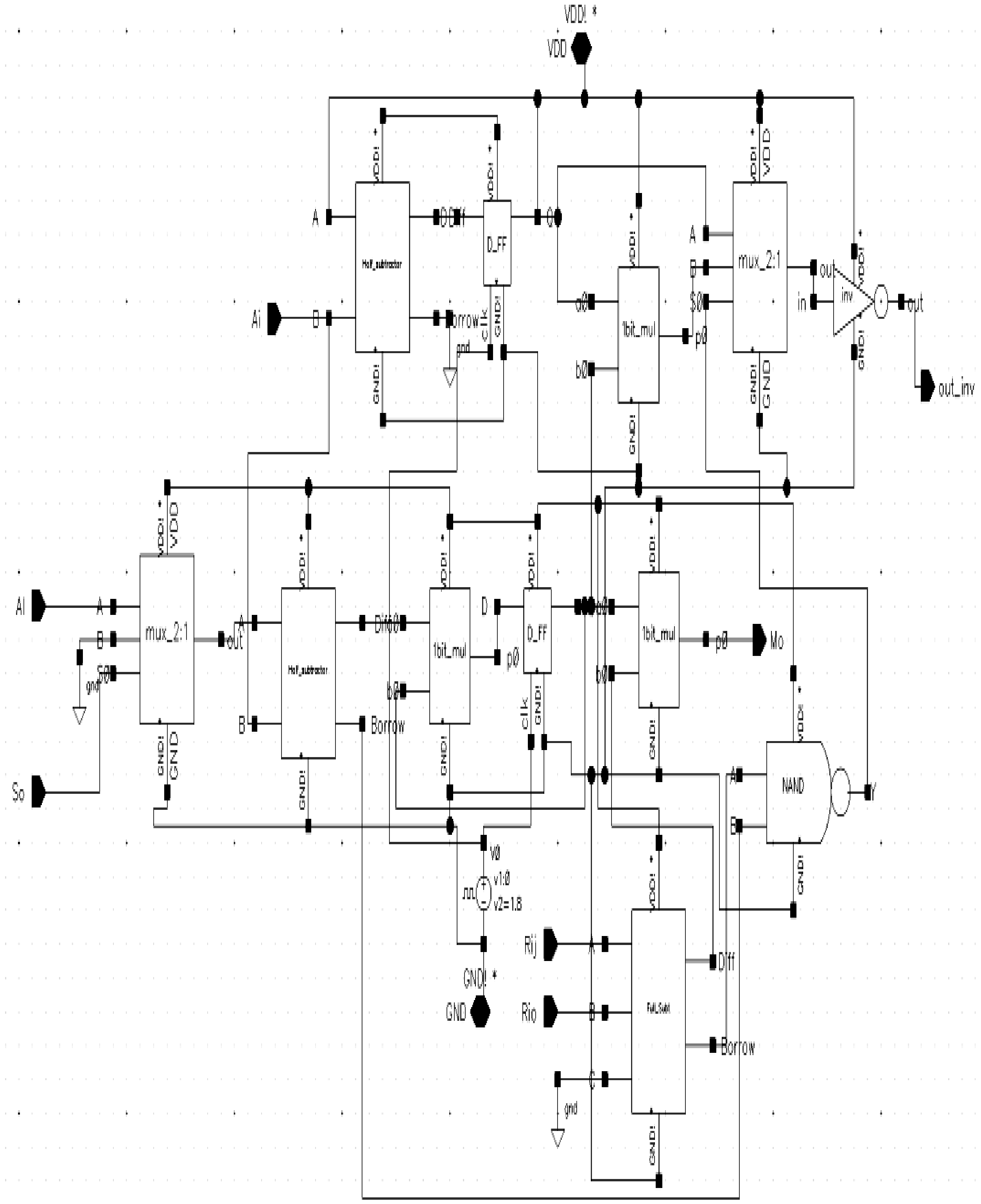


Fig.3.37 Schematic of Inverse Computation

3.13.4 Polynomial Selection

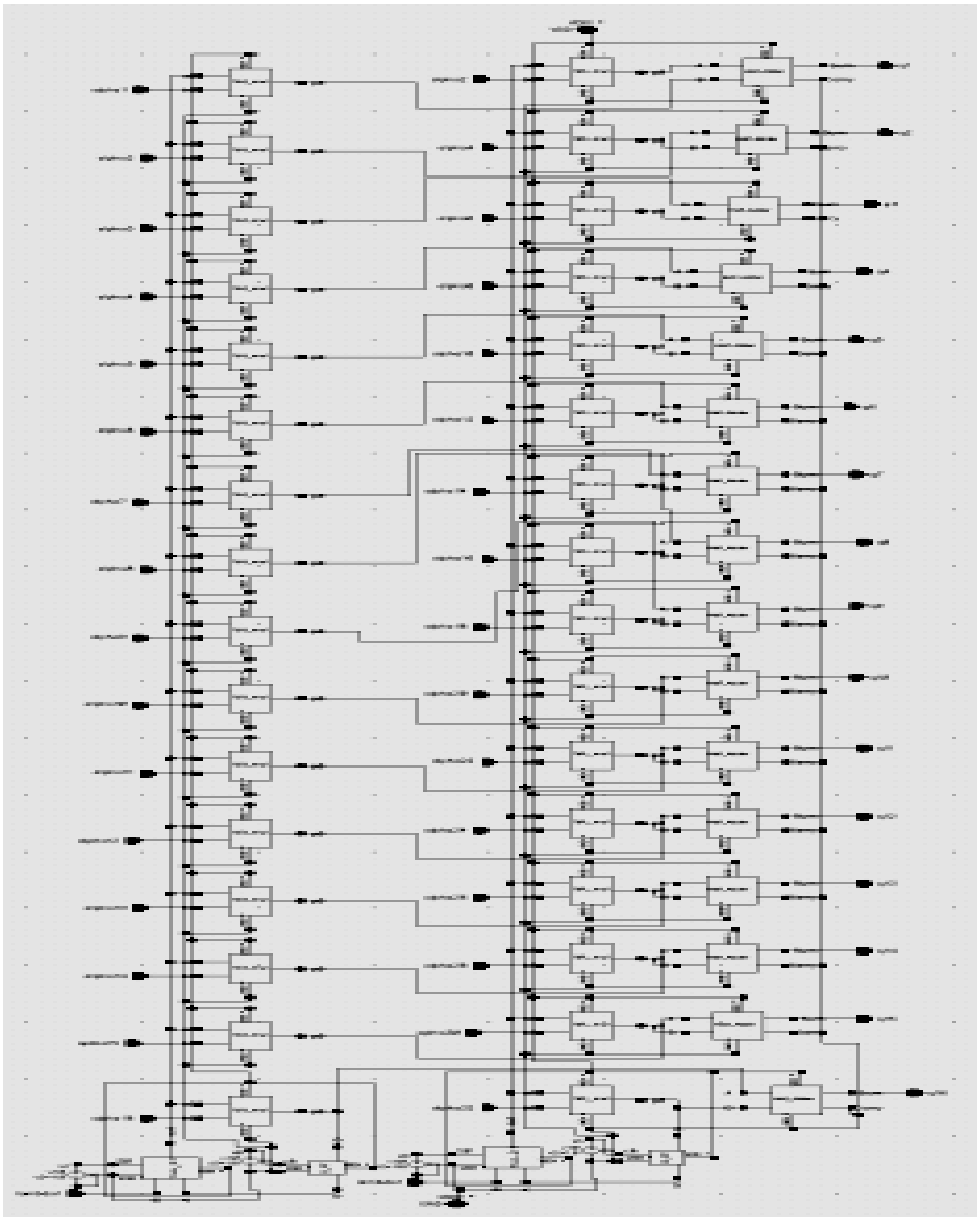


Fig.3.38 Schematic of Polynomial Selection

3.13.5 Error Computation

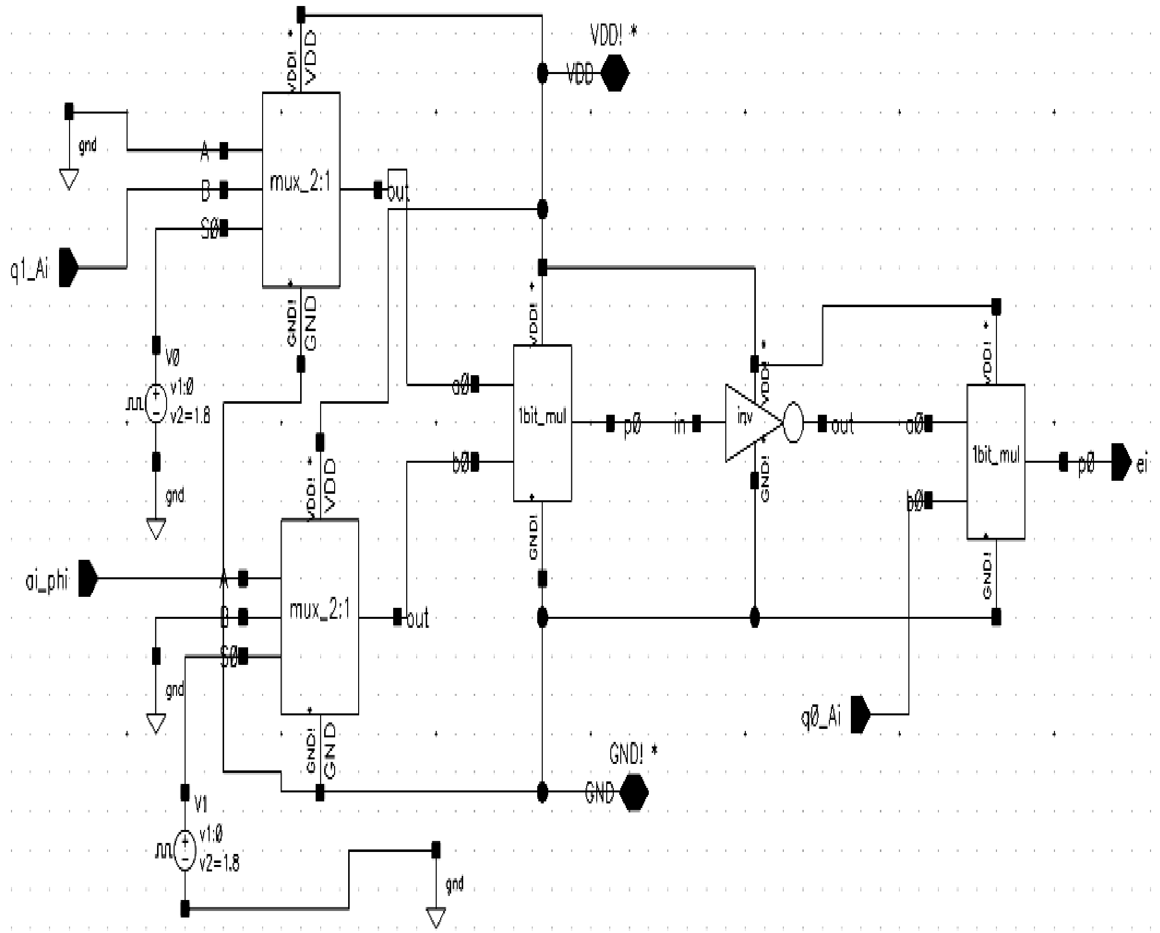


Fig.3.39 Schematic of Error Computation

3.14 Final Schematic of Reed-Solomon Decoder

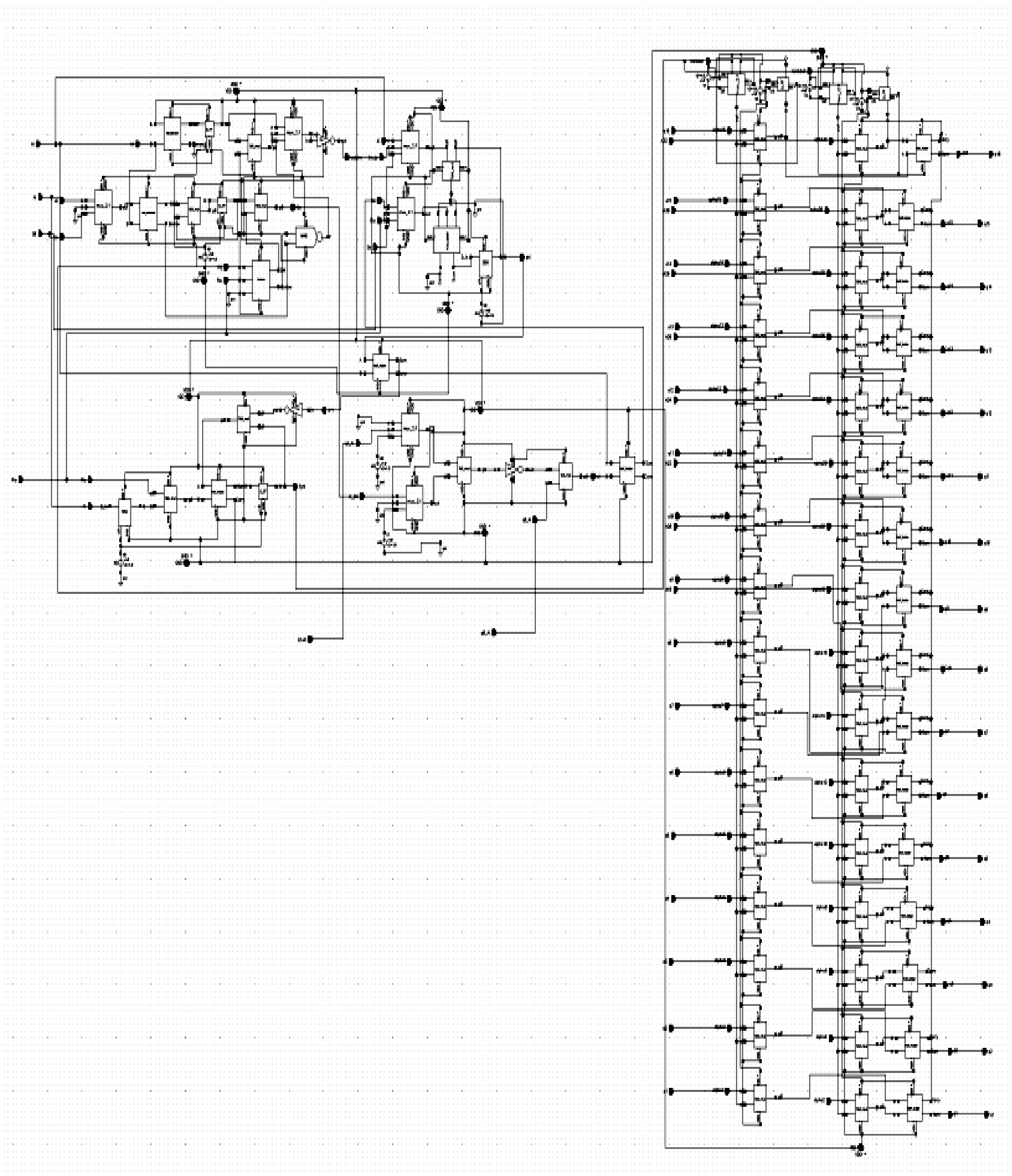


Fig.3.40 Schematic of Reed-Solomon Decoder

3.15 Reed-Solomon Decoder SOC using GDI

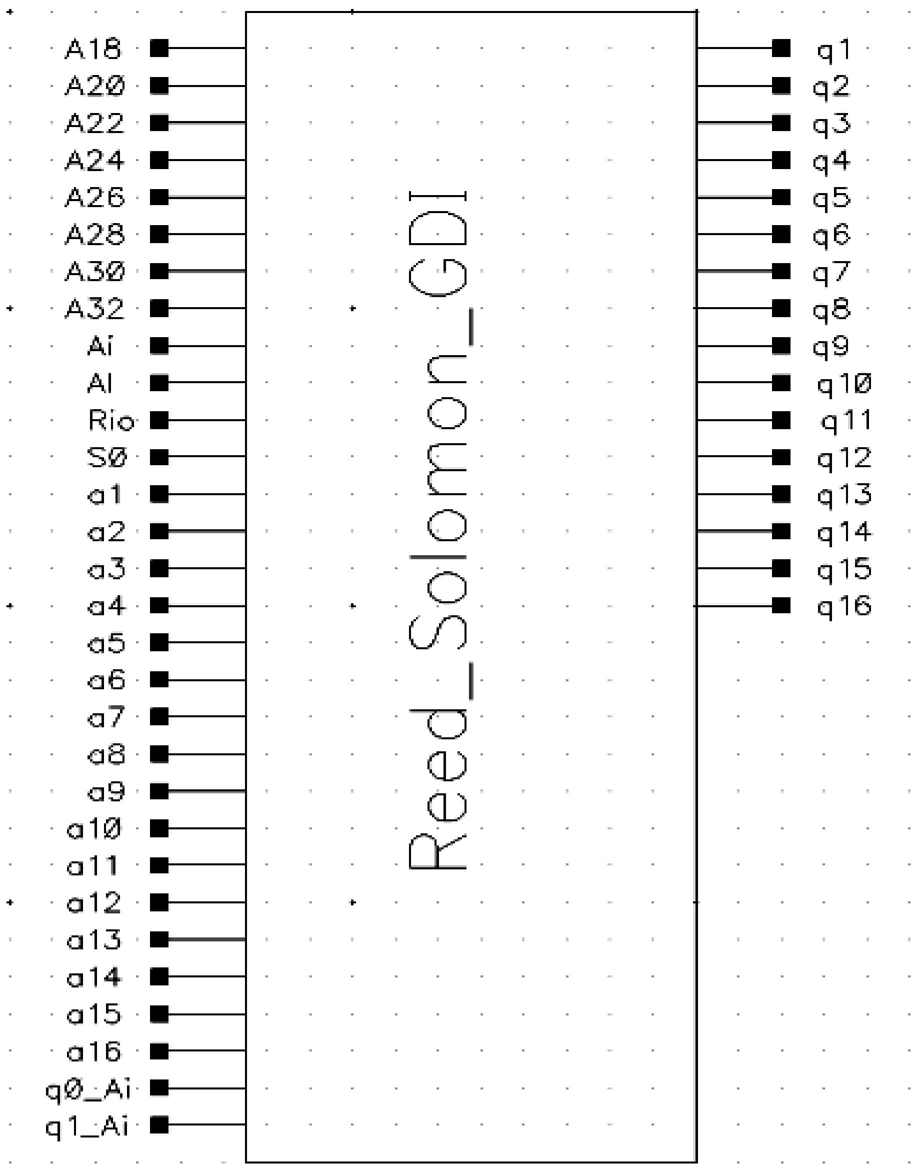


Fig.3.41 GDI based SOC of Reed-Solomon Decoder

3.16 Reed-Solomon Decoder SOC using PTL

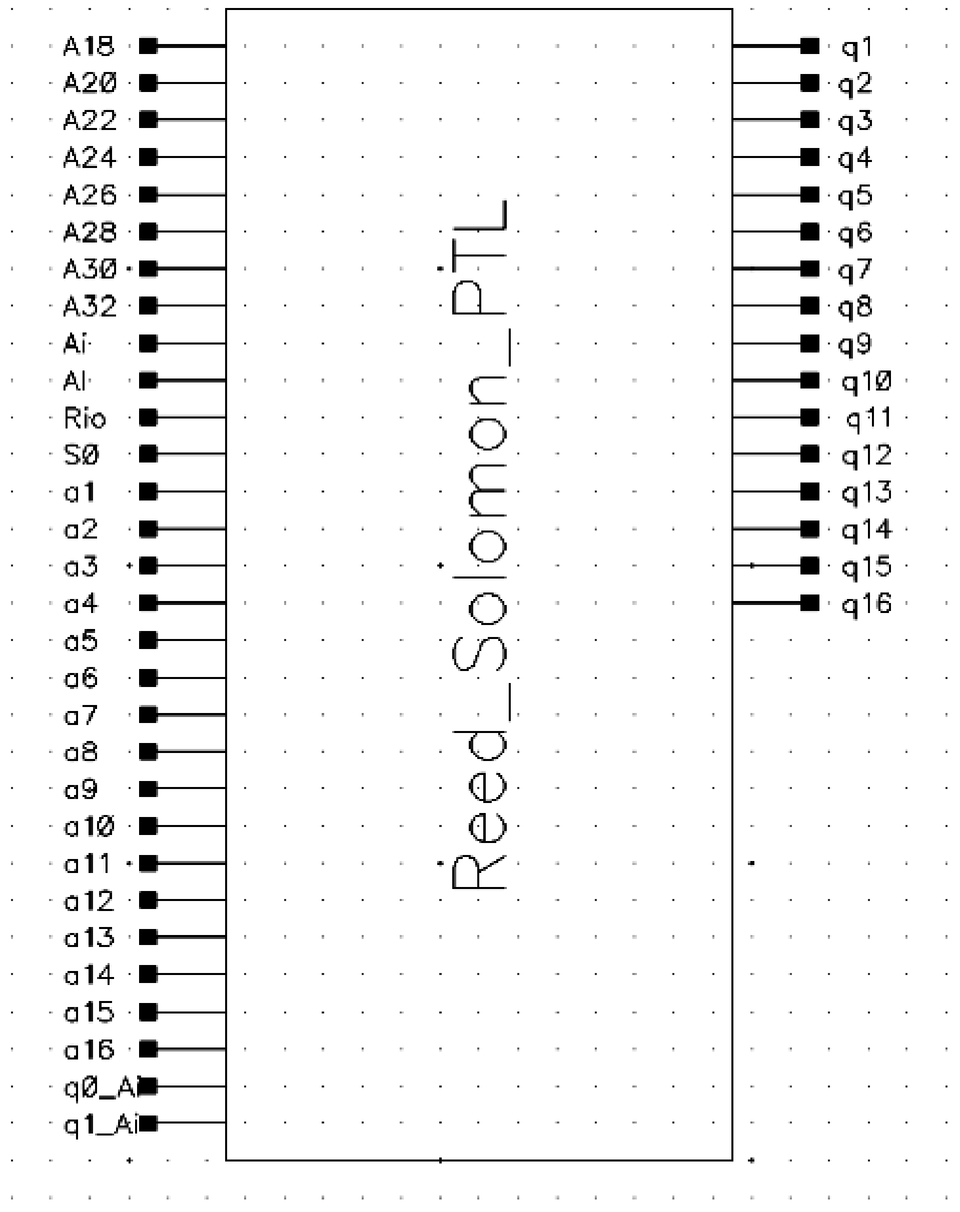


Fig.3.42 PTL based SOC of Reed-Solomon Decoder

CHAPTER 4
RESULT AND DISCUSSION

4.1 Graphs for Power Analysis of Parts of Reed Solomon decoder

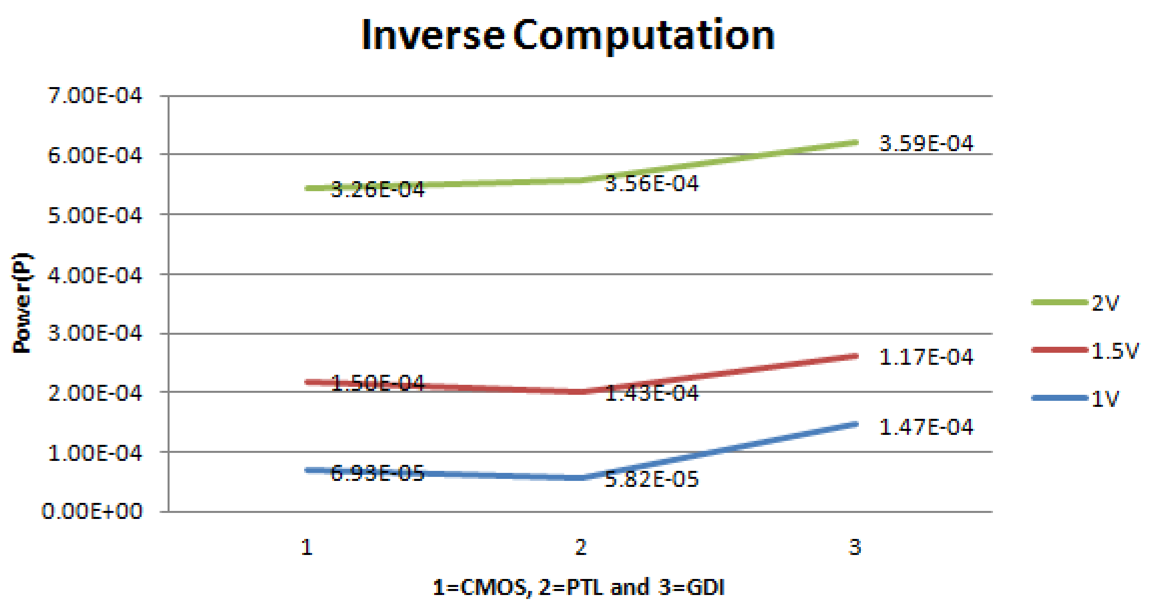


Fig.4.1 Power Graph for Inverse Computation for all three logic styles

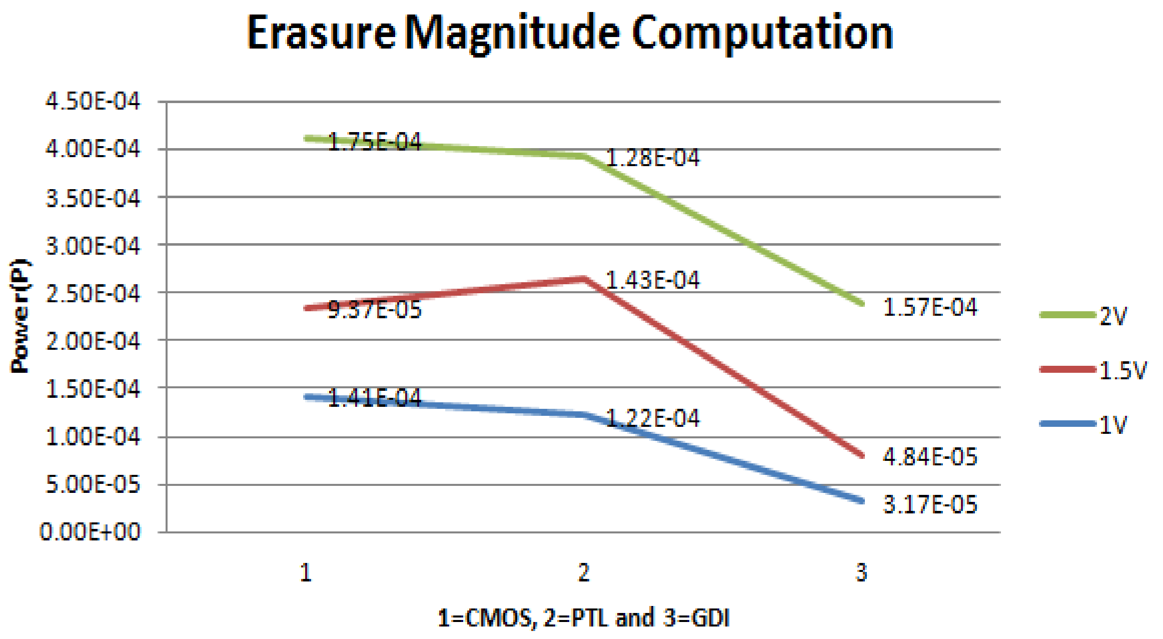


Fig.4.2 Power Graph for Erasure Magnitude Computation for all three logic styles

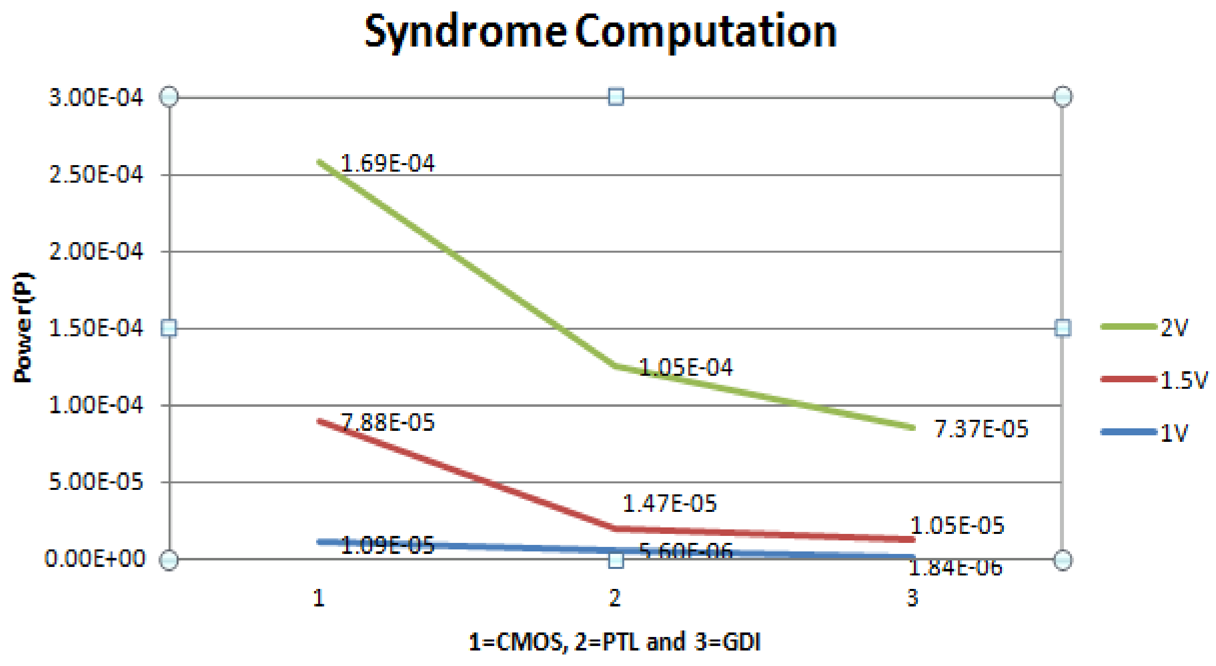


Fig.4.3 Power Graph for Syndrome Computation for all three logic styles

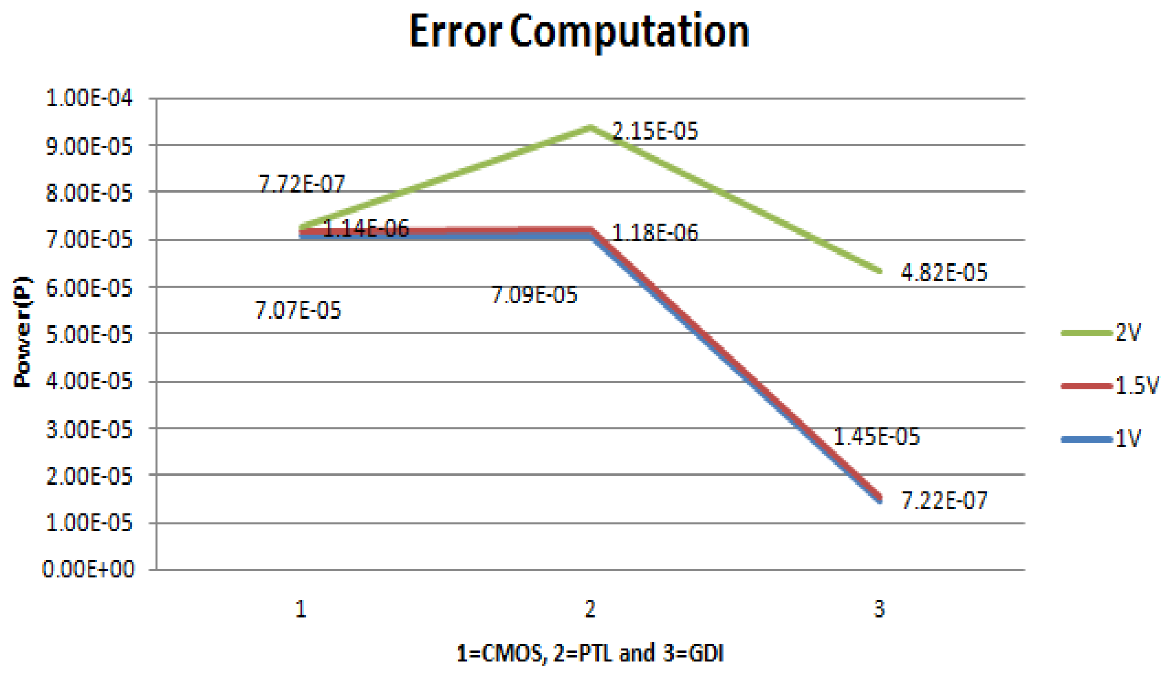


Fig.4.4 Power Graph for Error Computation for all three logic styles

4.2 Graphs for Delay Analysis of Parts of Reed Solomon decoder

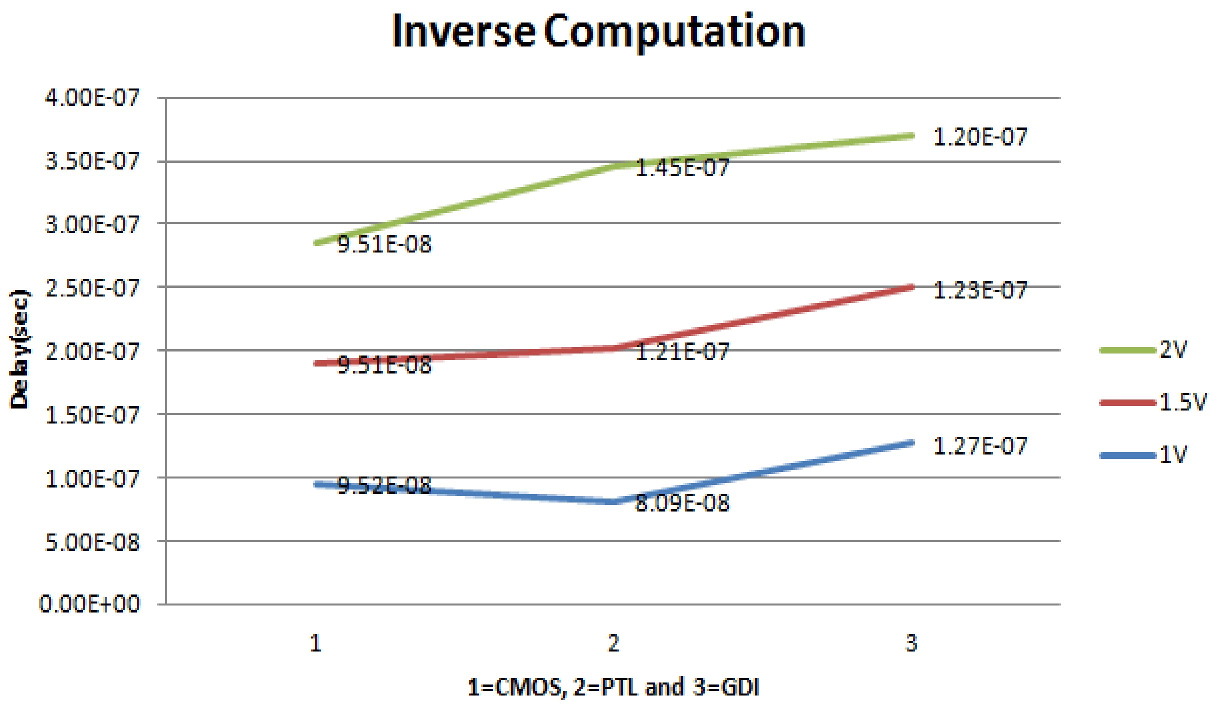


Fig.4.5 Delay Graph for Inverse Computation for all three logic styles

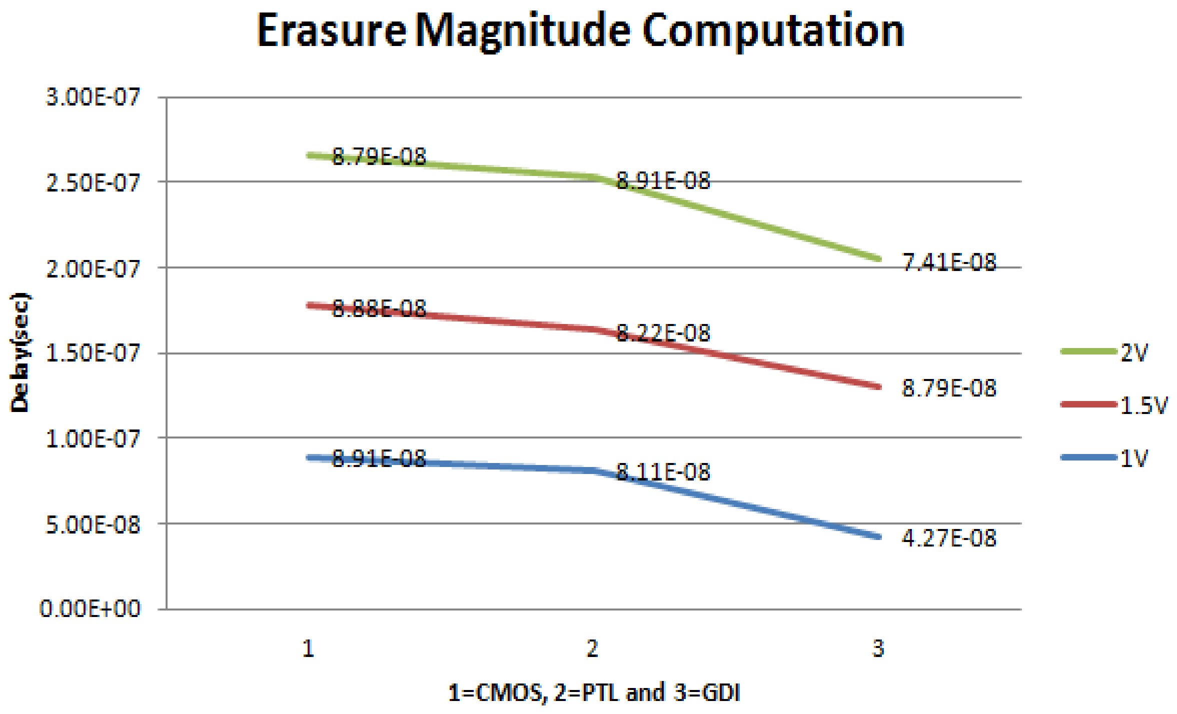


Fig.4.6 Delay Graph for Erasure Magnitude Computation for all three logic styles

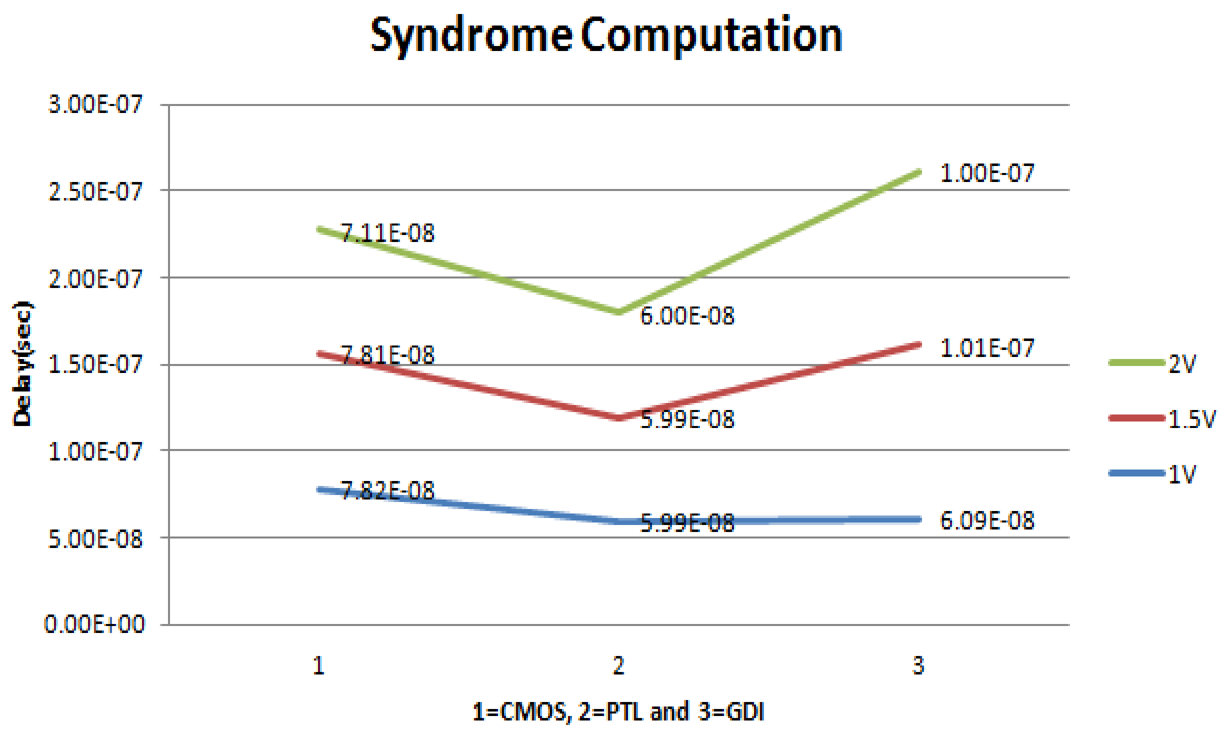


Fig.4.7 Delay Graph for Syndrome Computation for all three logic styles

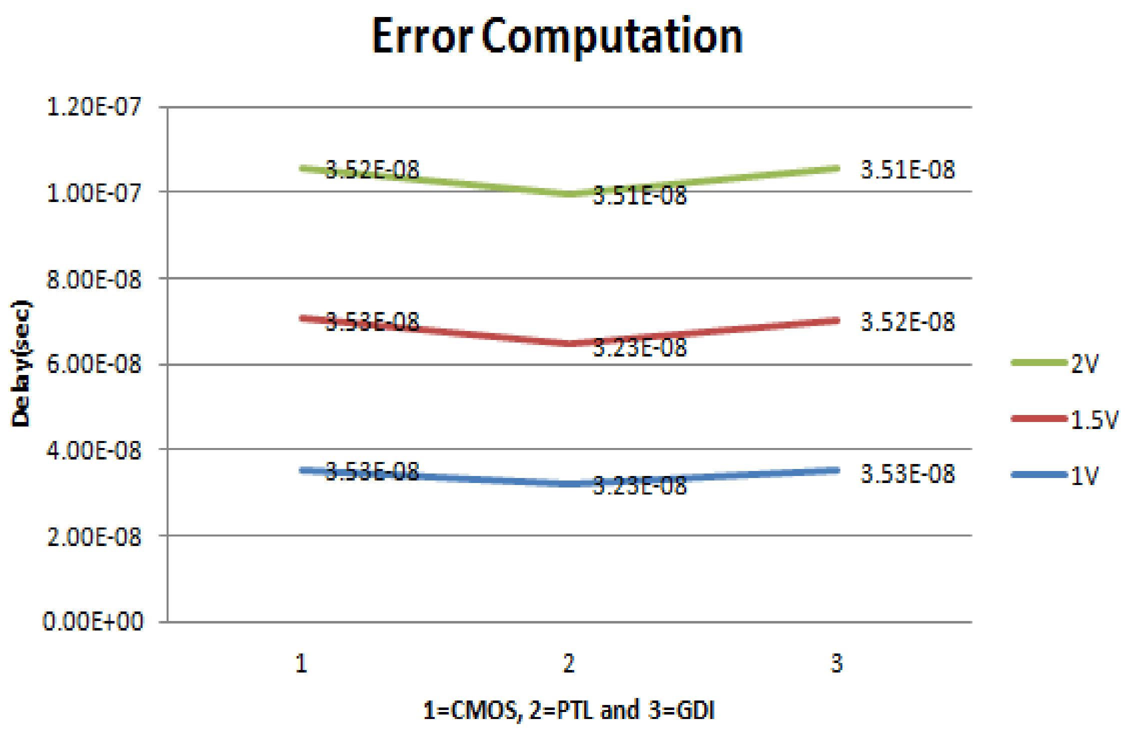


Fig.4.8 Delay Graph for Error Computation for all three logic styles

4.3 Graphs for PDP Analysis of Parts of Reed Solomon decoder

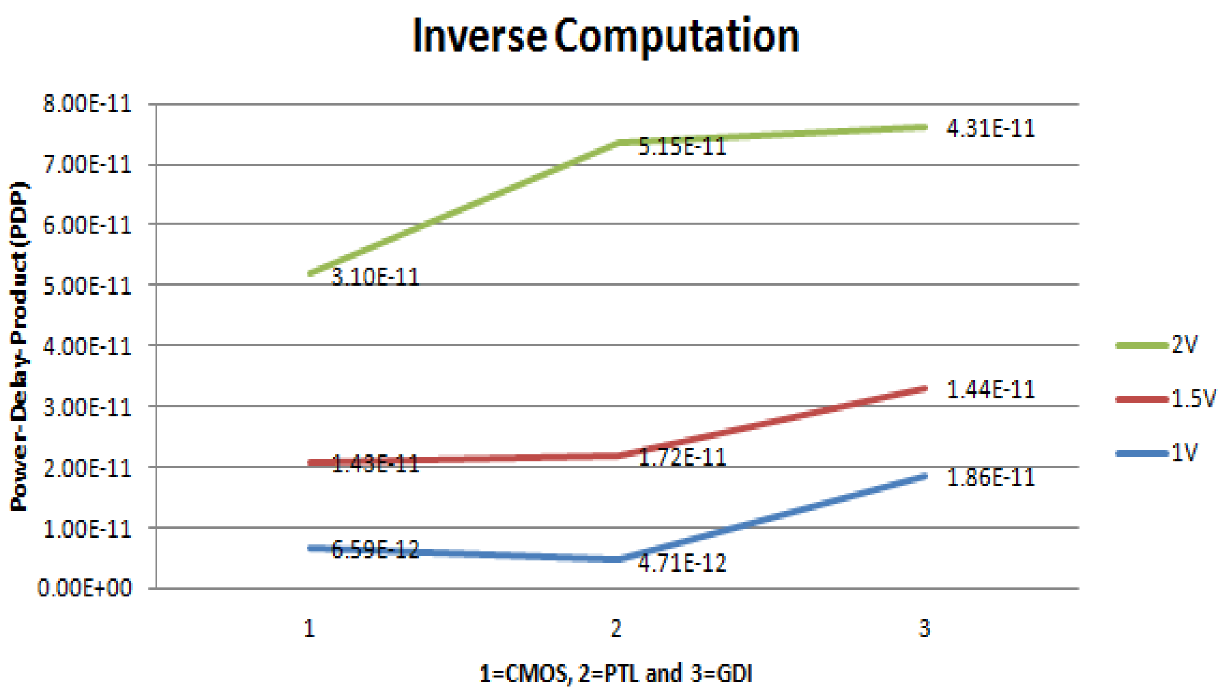


Fig.4.9 PDP Graph for Inverse Computation for all three logic styles

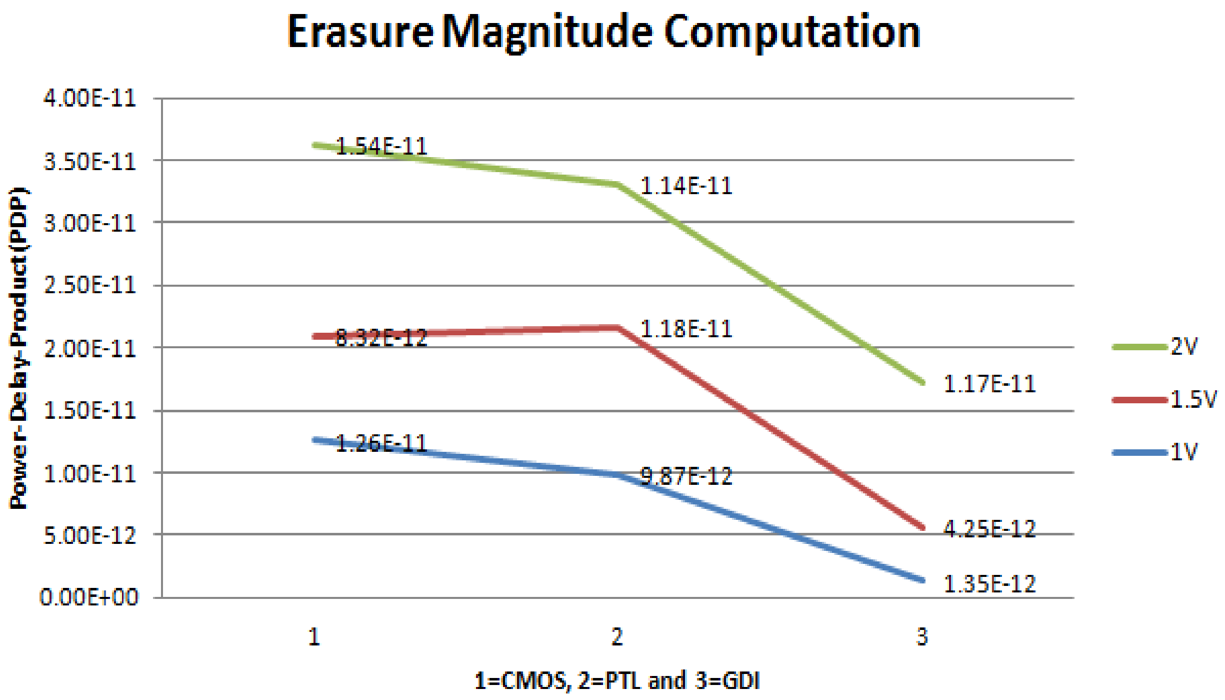


Fig.4.10 PDP Graph for Erasure Magnitude Computation for all three logic styles

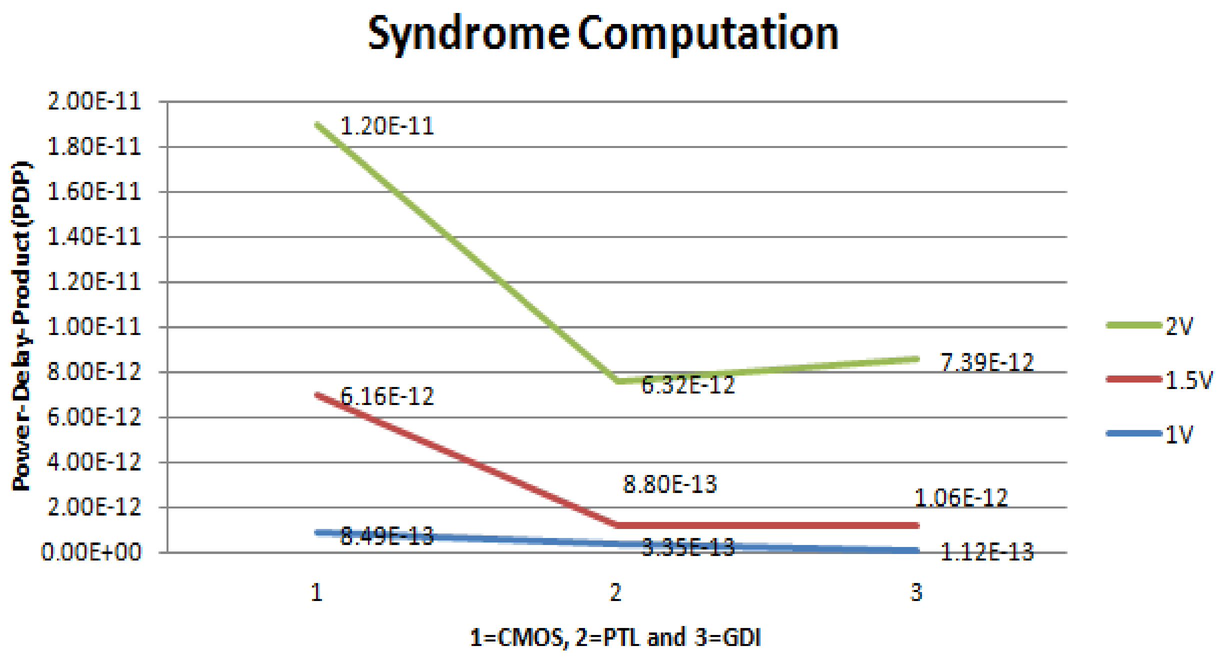


Fig.4.11 PDP Graph for Syndrome Computation for all three logic styles

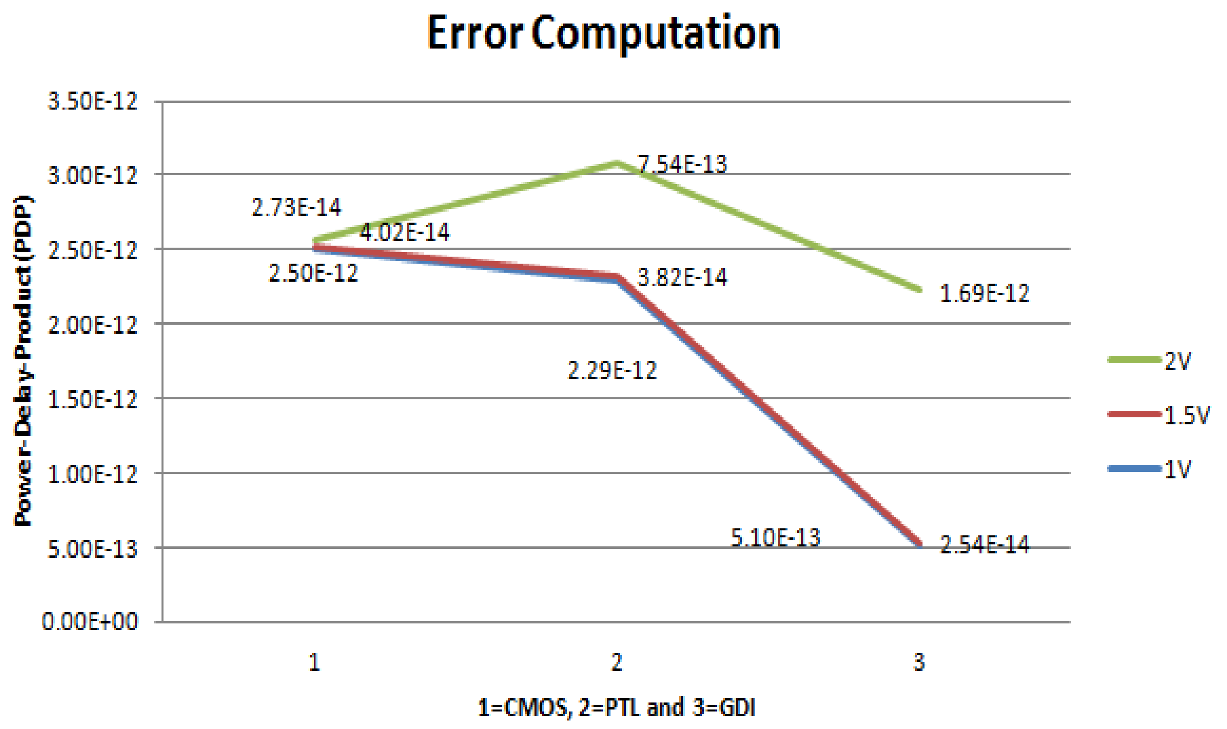


Fig.4.12 PDP Graph for Error Computation for all three logic styles

4.4 Power Analysis

4.4.1 CMOS Logic style

TABLE 4.1 Power values for CMOS logic Style

Sub-Circuits	2V	1.5V	1V
INV	2.90E-07	4.05E-08	4.78E-09
NOR	1.46E-07	3.80E-08	1.52E-08
NAND	3.92E-07	4.91E-08	2.89E-08
XOR	9.58E-07	1.82E-07	4.98E-08
AND	5.45E-07	1.49E-07	5.66E-08
HALF_ADDER	1.61E-06	5.75E-07	2.22E-07
FULL_ADDER	3.36E-06	1.36E-06	5.40E-07
HALF_SUBTRACTOR	1.58E-06	4.25E-07	1.50E-07
FULL_SUBTRACTOR	3.44E-06	9.67E-07	3.60E-07
D_FF	1.04E-06	3.44E-07	1.36E-07
MUL_1B	5.45E-07	1.49E-07	5.66E-08
MUL_2B	3.37E-06	1.25E-06	4.87E-07
MUX_2:1	3.63E-07	2.71E-07	5.25E-05
SHIFT_REGISTER	3.25E-06	1.19E-09	4.82E-07
ARCH_INV_COMP	3.26E-04	1.50E-04	4.93E-05
ERASURE_MAG	1.75E-04	9.37E-05	1.41E-04
SYNDROME_COMP	1.69E-04	4.88E-05	1.09E-05
ERROR	1.14E-06	4.72E-07	4.07E-05

4.4.2 PTL Logic style

TABLE 4.2 Power values for PTL logic Style

Sub-Circuits	**2V**	**1.5V**	**1V**
INV	2.90E-07	4.05E-08	4.78E-09
NOR	5.10E-06	9.91E-08	4.23E-08
NAND	5.41E-06	1.39E-07	4.18E-08
XOR	4.62E-07	1.23E-07	3.75E-08
AND	3.74E-07	8.32E-08	3.27E-08
HALF_ADDER	1.46E-05	1.51E-06	3.80E-07
FULL_ADDER	2.61E-05	3.37E-06	8.64E-07
HALF_SUBTRACTOR	1.04E-06	1.71E-07	3.98E-08
FULL_SUBTRACTOR	3.26E-05	8.99E-06	4.38E-07
D_FF	4.57E-06	8.71E-07	2.66E-07
MUL_1B	5.66E-06	2.17E-07	8.23E-08
MUL_2B	1.83E-04	4.03E-06	4.97E-07
MUX_2:1	9.17E-06	4.26E-07	5.26E-05
SHIFT_REGISTER	2.55E-05	3.13E-06	4.71E-07
ARCH_INV_COMP	3.56E-04	1.43E-04	5.82E-05
ERASURE_MAG	1.28E-04	1.43E-04	1.22E-04
SYNDROME_COMP	1.05E-04	1.47E-05	5.60E-06
ERROR	2.15E-05	1.18E-06	4.09E-05

4.4.3 GDI Logic style

TABLE 4.3 Power values for GDI logic Style

Sub-Circuits	2V	1.5V	1V
INV	2.90E-07	4.05E-08	4.78E-09
NOR	1.46E-07	3.80E-08	1.52E-08
NAND	5.71E-06	2.16E-07	5.89E-08
XOR	4.21E-07	2.07E-07	1.43E-05
AND	3.78E-09	4.66E-09	4.20E-09
HALF_ADDER	1.71E-05	1.82E-06	3.53E-07
FULL_ADDER	2.22E-05	4.69E-06	8.67E-07
HALF_SUBTRACTOR	4.49E-07	1.72E-07	1.47E-05
FULL_SUBTRACTOR	2.16E-05	1.02E-06	3.15E-05
D_FF	1.19E-05	4.77E-07	2.39E-07
MUL_1B	5.66E-06	2.17E-07	8.23E-08
MUL_2B	8.44E-05	1.04E-08	5.49E-07
MUX_2:1	4.05E-08	5.47E-08	1.46E-05
SHIFT_REGISTER	2.51E-05	2.91E-06	4.62E-07
ARCH_INV_COMP	3.59E-04	1.17E-04	1.47E-04
ERASURE_MAG	1.57E-04	4.84E-05	3.17E-05
SYNDROME_COMP	4.37E-05	1.05E-05	1.84E-06
ERROR	4.82E-05	4.22E-07	1.45E-05

4.5 Delay Analysis

4.5.1 CMOS Logic style

TABLE 4.4 Delay values for CMOS logic Style

Sub-Circuits	**2V**	**1.5V**	**1V**
INV	1.14E-11	1.19E-11	2.67E-11
NOR	2.00E-08	2.00E-08	2.00E-08
NAND	2.00E-08	2.00E-08	2.00E-08
XOR	9.66E-09	9.59E-09	2.01E-08
AND	2.41E-10	2.94E-10	3.60E-10
HALF_ADDER	4.13E-11	4.75E-11	4.52E-11
FULL_ADDER	9.90E-09	9.88E-09	9.84E-09
HALF_SUBTRACTOR	2.20E-11	2.20E-11	1.02E-08
FULL_SUBTRACTOR	9.88E-09	3.98E-08	3.96E-08
D_FF	3.01E-08	3.02E-08	9.77E-09
MUL_1B	1.98E-08	1.97E-08	1.97E-08
MUL_2B	3.96E-08	3.95E-08	3.94E-08
MUX_2:1	2.97E-08	2.96E-08	2.95E-08
SHIFT_REGISTER	2.03E-08	2.04E-08	2.07E-08
ARCH_INV_COMP	9.51E-08	9.51E-08	9.52E-08
ERASURE_MAG	8.79E-08	8.88E-08	8.91E-08
SYNDROME_COMP	4.11E-08	4.81E-08	4.82E-08
ERROR	3.52E-08	3.53E-08	3.53E-08

4.5.2 PTL Logic style

TABLE 4.5 Delay values for PTL logic Style

Sub-Circuits	2V	1.5V	1V
INV	3.24E-13	5.06E-12	2.17E-11
NOR	9.86E-09	9.86E-09	9.80E-09
NAND	9.83E-09	9.81E-09	4.00E-08
XOR	2.00E-08	2.00E-08	5.01E-08
AND	1.01E-08	4.02E-08	4.03E-08
HALF_ADDER	1.99E-08	1.99E-08	2.05E-08
FULL_ADDER	1.00E-08	1.00E-08	1.01E-08
HALF_SUBTRACTOR	4.01E-08	4.00E-08	4.00E-08
FULL_SUBTRACTOR	3.00E-08	4.03E-08	4.03E-08
D_FF	1.00E-08	1.01E-08	1.01E-08
MUL_1B	2.41E-10	2.94E-10	3.60E-10
MUL_2B	3.96E-08	3.95E-08	3.94E-08
MUX_2:1	3.01E-08	3.02E-08	3.02E-08
SHIFT_REGISTER	2.00E-08	5.00E-11	5.97E-11
ARCH_INV_COMP	1.45E-07	1.21E-07	8.09E-08
ERASURE_MAG	8.91E-08	8.22E-08	8.11E-08
SYNDROME_COMP	4.00E-08	5.99E-08	5.99E-08
ERROR	3.51E-08	3.23E-08	3.23E-08

4.5.3 GDI Logic style

TABLE 4.6 Delay values for GDI logic Style

Sub-Circuits	**2V**	**1.5V**	**1V**
INV	3.24E-13	5.06E-12	2.17E-11
NOR	2.00E-08	2.00E-08	2.00E-08
NAND	9.84E-09	9.83E-09	9.80E-09
XOR	5.01E-08	5.01E-08	5.02E-08
AND	4.02E-08	4.03E-08	4.03E-08
HALF_ADDER	5.01E-08	5.01E-08	3.87E-08
FULL_ADDER	9.00E-08	9.00E-08	4.00E-08
HALF_SUBTRACTOR	9.04E-08	9.05E-08	9.05E-08
FULL_SUBTRACTOR	4.03E-08	4.04E-08	4.05E-08
D_FF	9.98E-09	1.00E-08	1.02E-08
MUL_1B	4.02E-08	4.02E-08	4.03E-08
MUL_2B	8.05E-08	8.06E-08	8.06E-08
MUX_2:1	4.98E-08	4.99E-08	5.00E-08
SHIFT_REGISTER	2.02E-08	2.63E-10	1.45E-07
ARCH_INV_COMP	1.20E-07	1.23E-07	1.27E-07
ERASURE_MAG	4.41E-08	8.79E-08	4.27E-08
SYNDROME_COMP	1.00E-07	1.01E-07	4.09E-08
ERROR	3.51E-08	3.52E-08	3.53E-08

4.6 Power-Delay-Product (PDP) Analysis

4.6.1 CMOS Logic style

TABLE 4.7 PDP values for CMOS logic Style

Sub-Circuits	**2V**	**1.5V**	**1V**
INV	3.30E-18	4.80E-19	1.81E-19
NOR	2.92E-15	4.60E-16	3.04E-16
NAND	4.85E-15	1.58E-15	5.77E-16
XOR	9.25E-15	1.75E-15	1.00E-15
AND	1.31E-16	4.39E-17	2.04E-17
HALF_ADDER	4.65E-17	2.73E-17	1.66E-17
FULL_ADDER	3.33E-14	1.34E-14	5.32E-15
HALF_SUBTRACTOR	3.46E-17	9.32E-18	1.52E-15
FULL_SUBTRACTOR	3.39E-14	3.84E-14	1.43E-14
D_FF	3.14E-14	1.04E-14	1.32E-15
MUL_1B	1.08E-14	2.95E-15	1.11E-15
MUL_2B	1.34E-13	4.95E-14	1.92E-14
MUX_2:1	1.08E-14	8.04E-15	1.55E-12
SHIFT_REGISTER	4.61E-14	2.43E-17	9.95E-15
ARCH_INV_COMP	3.10E-11	1.43E-11	4.59E-12
ERASURE_MAG	1.54E-11	8.32E-12	1.26E-11
SYNDROME_COMP	1.20E-11	4.16E-12	8.49E-13
ERROR	4.02E-14	2.73E-14	2.50E-12

4.6.2 PTL Logic style

TABLE 4.8 PDP values for PTL logic Style

Sub-circuits	**2V**	**1.5V**	**1V**
INV	9.39E-20	2.05E-19	1.47E-19
NOR	5.03E-14	9.77E-16	4.15E-16
NAND	5.32E-14	1.37E-15	1.67E-15
XOR	1.32E-14	2.45E-15	1.88E-15
AND	3.76E-15	3.35E-15	1.32E-15
HALF_ADDER	2.92E-13	3.00E-14	4.81E-15
FULL_ADDER	2.61E-13	3.38E-14	8.68E-15
HALF_SUBTRACTOR	4.26E-14	4.82E-15	1.59E-15
FULL_SUBTRACTOR	9.76E-13	3.62E-13	2.57E-14
D_FF	4.60E-14	8.79E-15	2.68E-15
MUL_1B	1.36E-15	4.39E-17	2.96E-17
MUL_2B	4.26E-12	2.38E-13	2.74E-14
MUX_2:1	2.76E-13	1.28E-14	1.59E-12
SHIFT_REGISTER	5.11E-13	1.56E-16	4.01E-17
ARCH_INV_COMP	5.15E-11	1.72E-11	4.71E-12
ERASURE_MAG	1.14E-11	1.18E-11	9.87E-12
SYNDROME_COMP	4.32E-12	8.80E-13	3.35E-13
ERROR	4.54E-13	3.82E-14	2.29E-12

4.6.3 GDI Logic style

TABLE 4.9 PDP values for GDI logic Style

Sub-Circuits	**2V**	**1.5V**	**1V**
INV	9.39E-20	2.05E-19	1.47E-19
NOR	2.92E-15	4.60E-16	3.04E-16
NAND	5.62E-14	2.12E-15	5.77E-16
XOR	2.11E-14	1.04E-14	4.16E-13
AND	1.52E-16	1.88E-16	2.50E-16
HALF_ADDER	8.55E-13	9.14E-14	1.37E-14
FULL_ADDER	1.99E-12	4.22E-13	4.07E-14
HALF_SUBTRACTOR	5.87E-14	1.55E-14	1.33E-12
FULL_SUBTRACTOR	1.52E-12	4.20E-14	2.22E-12
D_FF	1.19E-13	4.80E-15	2.44E-15
MUL_1B	2.27E-13	8.74E-15	3.31E-15
MUL_2B	4.79E-12	8.39E-16	4.43E-14
MUX_2:1	2.02E-15	2.73E-15	4.31E-13
SHIFT_REGISTER	5.06E-13	4.65E-16	4.86E-14
ARCH_INV_COMP	4.31E-11	1.44E-11	1.86E-11
ERASURE_MAG	1.17E-11	4.25E-12	1.35E-12
SYNDROME_COMP	4.39E-12	1.06E-12	1.12E-13
ERROR	1.69E-12	2.54E-14	5.10E-13

4.7 Discussion

After all the work and analysis the power consumption using GDI technology was found to be less than the CMOS and PTL Logic styles while the delay found higher in PTL as compared to CMOS and GDI. The transistor count was found less in GDI as compared to CMOS and PTL. Thus, VLSI Design using GDI Logic level was found efficient in the designing of Reed Solomon Decoder.

CHAPTER 5
CONCLUSION

5.1 Conclusion

Since various architectures for the LCC have been depicted, therefore there is not an exact solution to a question that which LCC decoding technique was more proficient with the alteration of various decoding parameters. The diverse η along with code rates simultaneously resolve error rectification capability. In LCC decoder, combination of enhanced rate RS code with smaller η may consist of the related performance as in larger η along with lower code rate. Also, the lesser t will minimize the complexity in hardware whereas the greater η raise the amount of the test vectors in a vividly manner. Since if we increase η by 1, it will cause the test vectors to be doubled and in return which may cause to twice the hardware in order to maintain them and trigger more latency. Conversely, gain in the t will amounts to restricted hardware.

A detailed and reviewed work on LCC implementation of RS decoder has been seen through different forms of decoder designs; selection of test vectors and after the interpolation is applied only on the specific selected test vector. In addition proficient architecture of interpolation, polynomial selection and evaluation, erasure computation was developed. Although the reduction in various technology parameters was observed with an increase in η. Compared to past approaches, considerable area reduction and efficiency enhancement has been achieved without any loss of throughput. The latency of the RS decoder can be reduced with the help of pipelining and syndrome computation. Future work will concentrate on further advancement in technology and improvement in code recovery through various technology based parameters.

REFERENCES

[1] G.Reed and I. Solomon, (1960)“Polynomial codes over certain finite fields”. Journal of the society for Industrial and Applied Mathematics, VOL.8, pp.300-304.

[2] W.Peterson, (1960) “Encoding and error correction procedures for bose-chaudhuri codes”. IRE transactions on Information theory, pp.459-470.

[3] E. Berlekamp and L.welch, (1986) “Error correction for algebrain block codes”. US patent 4 633 470.

[4] J. Massey, (1968) “Shift register synthesis and bch decoding”. IEEE Transaction on Information Theory, pp.122-127.

[5] Gary K. Maki, Patrick A. Owsley, Kelly B. Cameron and Jack Venbrux, (1986), “VLSI REED SOLOMON DECODER DESIGN”. University of Idaho, Moscow.

[6] Anh Dinh and Daniel Tang, (2004) “Design of High-Speed (255,239) RS Decoder using 0.18 M CMOS”. IEEE, May, pp.2171-2174.

[7] Wolfgang Wilhelm, Andre Kaufmann and Tobias G. Noll, (1998) “A New Scalable VLSI Architecture for Reed Solomon Decoders”. IEEE CUSTOM INTEGRATED CIRCUITS CONFERENCE, pp.13-16.

[8] KUANG YUNG LIU, (1984) “Architecture for VLSI Design of Reed – Solomon Decoders ”. IEEE Transactions on Computers, Feb, VOl. C-33.

[9] Hanho Lee, Meng-Lin yu and Leilei Song, (2000) “VLSI DESIGN OF REED -SOLOMON DECODER ARCHITECTURES ” . IEEE International Symposium on Circuits and Systems, May.

[10] Jason Bellorado and Aleksander Kavcic, (2006) “A Low – Complexity Method for Chase Type Decoding of Reed-Solomon Codes”. IEEE, ISIT, July, pp. 2037-2041.

[11] Hung-Wei Chen, Jiin-Chuan Wu, Gwo-Sheng Huang, Ji-Chien Lee and Shin-Shi Chang, (1995), “A New Vlsi Architecture of Reed-Solomon Decoder with Erasure Function”. IEEE.

[12] Wolfgang Wilhelm, (1999) “A new scalable VLSI architecture for Reed-Solomon Decoders”. IEEE Journal of Solid-State Circuits, March, Vol. 34, No. 3.

[13] W. Zhang, J. Wang, H. Wang, Y.Y.Liu, Z. Jiang, and S.Q. Wu. (2012) “Low-power high-efficiency architecture for low-complexity chase soft-decision Reed-solomon decoding”. IET Communication; Vol. 6, No. 17, August, pp. 3046-3052.

[14] Richard Huynh, GE Ning and YANG HuaZhong, (2009) “A Low Power Error Detection in the Syndrome Calculator Block for Reed-Solomon Codes : RS(204,188)”. Aug, Vol.14, No. 4.

[15] Dong-Sun Kim, Jong-Chan Choi and Duck-Jin Chung, (1999) “Implementation of High Speed Reed-Solomon Decoder”. IEEE.

[16] Xinmiao Zhang, “High-Speed VLSI Architecture for Low-Complexity Chase Soft-decision Reed-Solomon Decoding”. IEEE.

[17] Xinmiao Zhang, and Yu Zheng. (2012) “Systematically Re-encoded Algebraic Soft-Decision Reed-Solomon Decoder”. IEEE TRANSACTIONS on Circuits and Systems; Vol. 59, No.6, June, pp. 376-380.

[18] Xinmiao Zhang, and Yu Zheng. (2011) “Efficient Codeword Recovery Architecture for Low-Complexity Chase Reed-Solomon Decoding”. National Science Foundation, IEEE, May, pp.497-499.

[19] Kunal and Nidhi Kedia, (2012), “GDI Technique: A Power-Efficient Method for Digital Circuits”. International Journal of Advanced Electrical and Electronics Engineering, Vol. 1, No. 3, pp.87-93.

APPENDIX

LIST OF ABBREVATIONS

B:

BMA**:** Berlekamp-Massey Algorithm

C:

CMOS: Complementary Metal Oxide Semiconductor

CD: Compact Disc

D:

DRAM: Dynamic Random Access Memory

E:

ESA: European Space Agency

F:

FEC**:** Forward Error Correction

FH: Frequency Hopping

G:

GF: Galois Field

I:

IEEE: Institute of Electrical and Electronics Engineers

L:

LFSR: Linear Feedback Shift Register

LCC: Low Complexity Chase

M:

MOS: Metal Oxide Semiconductor

N:

NASA: National Aeronautic Space Agency

R:

RS: Reed-Solomon

S:

SS: Spread Spectrum

V:

VLSI: Very Large Scale Integration