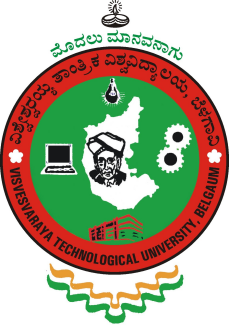
**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

Jnanasangama, Macche, Santibastwada Road, Belagavi-590018, Karnataka



**A**

**MINI PROJECT REPORT**

on

**Galois Field arithmetic**

**B - 25**

*Submitted in partial fulfillment of the requirement for the degree of*

**Bachelor of Engineering**

**in**

**Electronics & Communications Engineering**

*by*

**VISHVENDRA SINGH**

**V semester-**

**(USN: 1DS18EC156)**

**SAMEER GAUTAM**

**V semester-**

**(USN: 1DS18EC143)**

**SAURABH SINGH**

**V semester-**

**(USN: 1DS18EC144)**

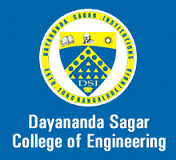
Under the

guidance

of

**Dr. A. Rajagopal**

Associate Professor, ECE Dept., DSCE, Bengaluru



**Department of Electronics & Communication Engineering**

**Dayananda Sagar College of Engineering**

(An Autonomous College affiliated to VTU Belgaum & accredited by NBA/NAAC)

Shavige Malleshwara Hills, Kumaraswamy Layout,

Banashankari, Bengaluru-560078, Karnataka

**Mini project Report Declaration**

Certified that the UG Mini project entitled, “IMPLEMENTATION OF GALOIS FIELD ARITHMETIC” has been submitted as AAT for the subject Digital System design using Verilog with Subject code-18EC5DCDSV is a bonafide work that is carried out by myself in partial fulfillment for the award of degree of Bachelor of Engineering in Electronics & Communication Engineering of the Visvesvaraya Technological University, Belagavi, Karnataka during the academic year 2019-20. I am solely responsible for all the contents that has have been presented in it.

VISHVENDRA SINGH 1DS18EC156

SAMEER GAUTAM 1DS18EC143

SAURABH SINGH 1DS18EC144

Date : 02 / 01 / 2021 Place : Bengaluru -78

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Mini project Guide

Name & Signature

Dr. A. Rajagopal

Contents

1. Abstract

2. Introduction

3. Literature survey

4. Problem statement

5. Block diagram

6. Results

7. Conclusions

8. References

**Abstract**

Galois field GF(2m) has many important applications, such as cryptography and error correcting codes. For high-speed implementation of these applications, efficient implementation of arithmetic operations in GF(2m) is important. Finite Field arithmetic is becoming increasingly a very prominent solution for calculations in many applications. Galois Field arithmetic forms the basis of BCH, Reed-Solomon and other erasure coding techniques to protect storage systems from failures. Most implementations of Galois Field arithmetic rely on multiplication tables or discrete logarithms to perform this operation. Software-based Galois field implementations are used in the reliability and security components of many storage systems. Unfortunately, multiplication and division operations over Galois fields are expensive, compared to the addition. To accelerate multiplication and division, most software Galois field implementations use pre-computed look-up tables, accepting the memory overhead associated with optimizing these operations. However, the amount of available memory constrains the size of a Galois field and leads to inconsistent performance across architectures. Typical arithmetic unit includes an adder or subtracter, multiplier and divider. Addition operation is done with one n-bit XOR operation, for multiplication operation LSB first multiplying method is used and Fermat’s little theorem for multiplicative inverse operation. These operations simulated using Verilog on Modelsim – FPGA starter edition simulator.

**Introduction**

A finite field or Galois field is a that contains a finite number of [elements](https://en.wikipedia.org/wiki/Element_(mathematics)). As with any field, a finite field is a [set](https://en.wikipedia.org/wiki/Set_(mathematics)) on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules. The most common examples of finite fields are given by the [integers mod *p*](https://en.wikipedia.org/wiki/Integers_mod_n) when *p* is a [prime number](https://en.wikipedia.org/wiki/Prime_number). The result of adding or multiplying two elements from the field is always an element in the field. One element of the field is the element zero, such that a + 0 = a for any element a in the field. One element of the field is unity, such that a • 1 = a for any element a in the field. For every element ‘a’ in the field, there is an additive inverse element -a, such that a + (- a) = 0. This allows the operation of subtraction to be defined as addition of the inverse. For every non-zero element b in the field there is a multiplicative inverse element b -1 such that b b-1= 1. This allows the operation of division to be defined as multiplication by the inverse.

Finite fields are algebraic structures that are used in error-correcting coding, cryptography and digital signal processing. Arithmetic in a finite field is different from standard integer arithmetic. There are a limited number of elements in the finite field; all operations performed in the finite field result in an element within that field. Finite fields are used in a variety of applications, including in classical coding theory in linear block codes such as BCH codes and Reed Solomon error correction and in cryptography algorithms such as the Rijndael encryption algorithm. All finite fields have pn elements where p is prime and n is an integer at least 1. Conversely, for every number of the form pn there is a field that size. Furthermore, all groups of a given size are isomorphic. The field with pnelements is sometimes called the Galois field with that many elements, written GF(pn). The Galois fields of order GF(p) are simply the integers mod p. For n > 1, the elements of GF(pn) are polynomials of degree n-1 with coefficients coming from GF(p). Arithmetic operations (addition, subtraction, multiplication, division) are slightly different in Galois Fields than in the real number system we are used to. This is because any operation (addition, subtraction, multiplication or division) applied in Galois fields must yield results that are elements of the Galois field only. Such a condition cannot be followed through with operations used in the real number system.

**Representation of Galois field elements (24)**

Consider the equation, **P(x) = x4 + x + 1**

A Galois Field defined by GF (24) does not have both 0 and 1 as their roots

From the above equation, it is clear that, neither 0 nor 1 is the root of the equation. So, we can say that the 4th order root of above equation lies outside of the GF (24 ) field.

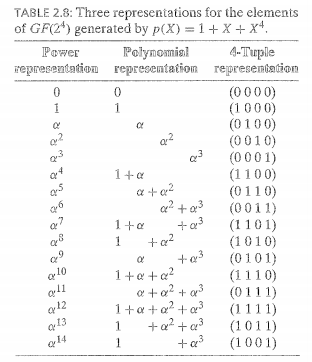
By assuming ‘α’ as one of the root of the equation,

Set P(α)=0, P (α) = α4 + α + 1 = 0 . Rearranging the equation,

The higher order field elements can be generated similarly by multiplying α to its previous power. The fifteenth power of α can be calculate as below: α15= α14. α = 1.

Here, the simplification of the fifteenth order gives 1, which is an existing element so; further powers of α will always give the existing elements.

Therefore, the field **GF (24)** has the following **16 elements**:



**Literature Survey**

There has been lot of work being done on finite field arithmetic operation. Sharma (2016) implemented S-box and inverse S-box using lookup table method for AES algorithm. Prasad (2016) proposed an improved implementation of SBOX for AES algorithm. Both these papers represent the encryption and decryption process for secure data communication over a channel. They have also analyzed the power using some power reduction techniques available in the literature that shows significant improvement in the power. Ahmad (2010) proposed a scheme of combination logic optimization over lookup table method. It will lead to fast processing on FPGA and less resource utilization. Jridi (2010) shows the process of encryption and decryption on images sent over communication channel. Two examples are provided by Hasan and Wassal and Kim and Lee. The idea of designing a finite field arithmetic unit for error-correcting coding seems to be interesting one. The hardware architectures for multiplication and inversion have been developed by a number of people over many years. Finite field arithmetic is a well-studied branch in mathematics, and hardware implementations have been known for many years. A number of early proposals can be found in Berlekamp. The first systolic architectures were proposed in the 80’s and have been further developed since. Some examples of implementations of bit-serial systolic arrays for multiplications are given by Wang and Lin, Tsai and Wang. Digital-serial systolic multipliers have been proposed by Kim, Han and Hong and Guo and Wang.

**Problem Statement**

Wireless broadband radio transmissions and computer hard drive storage are applications where the error correction is necessary, and where the demands on the coding and decoding speed are growing. If such systems are to work under more and more extreme conditions, effective error-correcting codes must be used, which in turn means that there is a need for faster arithmetic operation. There are number of ways of performing finite field arithmetic. One method is the use of software algorithms that perform the computations in a processor. That is flexible method, since the processor can normally be programmed for any finite field or field representation.

**Galois Fields Addition and subtraction:** To describe GF Addition and Subtraction, a useful way to describe the GF elements is in polynomial form. In addition to representing GF elements in index form as above, each of the GF elements may be written as a polynomial of form:

am-1xm-1 + ... +a2x2+ a1x1 + a0

where am-1...a0 take values 0 or 1. Thus, a GF element may also be described as a binary number am-1am-2...a1a0. By representing GF elements in binary form, GF addition or subtraction is implemented as the bit-by-bit exclusive-OR function of two binary numbers. Since addition and subtraction are synonymous in the exclusive-OR function of two binary numbers, minus signs in GF arithmetic may always be replaced by plus signs.

**Field generator polynomial p(x):** Also known as the primitive polynomial, this is an irreducible polynomial (with no factors) of degree m. In this project, the predetermined value form is 8. The following polynomial is used as the field generator polynomial:

x8+ x4+ x3+ x2+ 1.

As its name suggests, p(x) is used to construct all the field elements. It does so by using the fact that α is a root of p(x), i.e. p(α) = 0. Thus,

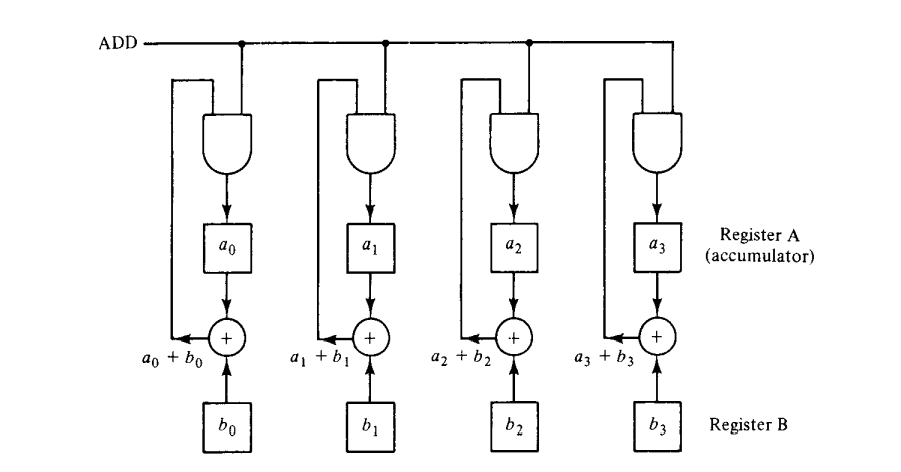
α8+ α 4+ α 3+ α 2+ 1 = 0

or      α8= α 4+ α 3+ α 2+ 1\*

\*Minus signs are interchangeable with plus signs in GF arithmetic.

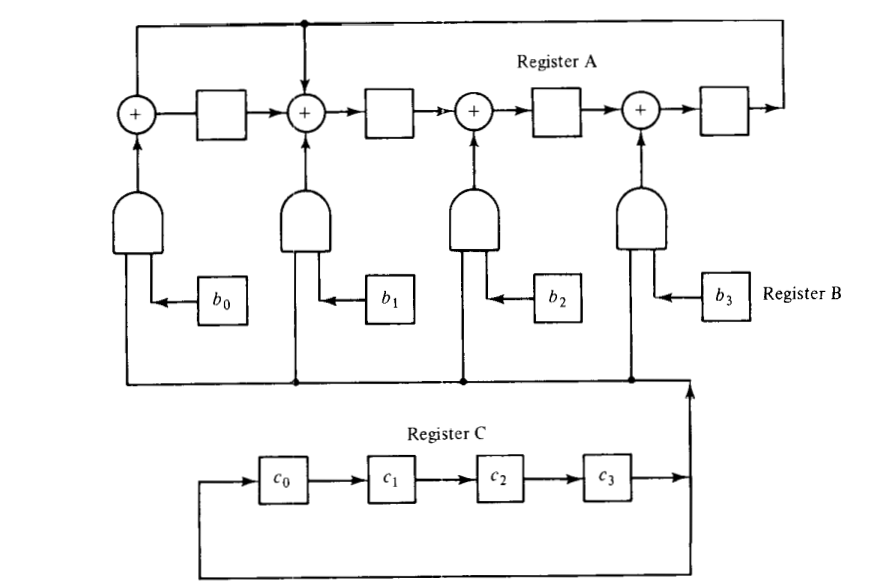
**Galois Fields Multiplication:** Multiplication in Galois field is the product modulo of p(x). Represent numbers you want to multiply in polynomial form. Perform normal multiplication with the polynomials. Divide this product by p(x), remainder of this division is the GF product.

**BLOCK DIAGRAM**

ADDITION OF GALOIS FIELD:

In GF(2m), to add two fields elements, we simply add their vector representation (XOR operation). The resultant vector is then the vector representation of the sum of the two field elements.

MULTIPLICATION OF GALOIS FIELD:



In GF (24), to multiply two arbitrary field elements. Let β and γ be two elements in GF (24).

Express these two elements in polynomial form:

β = b0 + b1α + b2α2 + b3α3 γ = c0 + c1α + c2α2 + c3α3

Then the product βγ can be expressed in the following form:

βγ = (((c3β)α + c2β)α + c1β)α + c0β

This product can be carried out with the following steps:

1. Multiply c3β by α and add the product to c2β.

2. Multiply (c3β)α + c2β by α and add the product to c1β.

3. Multiply ((c3β)α + c2β)α + c1β by α and add the product to c0β.

(15,7) BCH Decoder (Syndrome calculation):

The decoding algorithm for BCH codes consists of three major steps :-

* Calculate the syndrome value Si , i=1,2,….,2t from the received word, r(x).
* Determine the error location polynomial s*(x)*
* Find the roots of *s(x)* and then correct the errors

1. GALOIS FIELD ADDITION:

TO ADD TWO GALOIS FIELD ELEMENTS:

α7 + α13 :

α7  and α13 in GF (24),

In vector representation we can write:

α7 => 1101

α13 => 1011

So, α7 + α13 will be,

1101 + 1011 = 0110 [XOR-Operation]

which is the vector representation of α5.

1. GALOIS FIELD MULTIPLICATION:

TO MULTIPLY TWO GALOIS FIELDS ELEMENTS:

α5 + α6 :

α5  and α6 in GF (24),

In vector representation we can write:

α5 => 0110

α6 => 0011

So, (α5) \* (α6 ­) will be,

α11 =>0111.

1. SYNDROME CALCULATION:

For R(x)=1+x­­­­8,

S = [S1 S2 S3 S4]

Substitute α1, α2­, α3, α4 in R(x) for Syndrome:

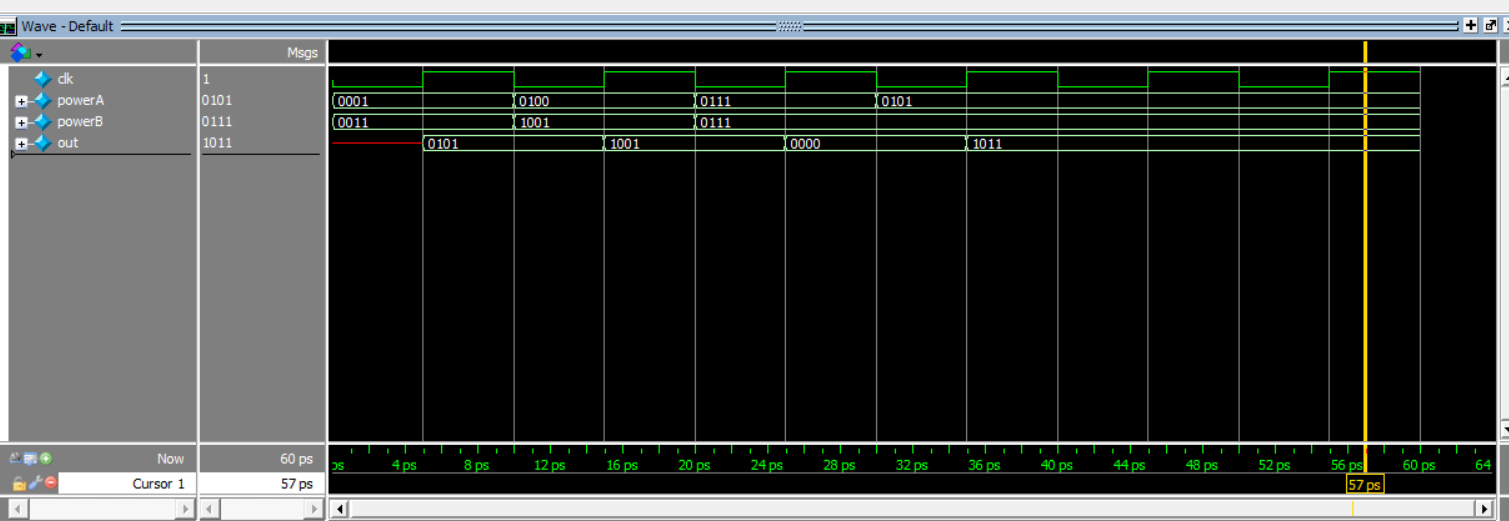
S = [α2, α4­, α7, α8]

**Results**

In this project, simulation and synthesis results of Galois field arithmetic is discussed. The unit has been simulated using Modelsim – FPGA starter edition. Synthesis was carried out using RTL schematic viewer.

GALOIS FIELD ADDITION:

Waveform:



OUTPUT (COMMAND WINDOW):

#0 clk=0, powerA=0001, powerB=0011, out=xxxx

#5 clk=1, powerA=0001, powerB=0011, out=0101

#10 clk=0, powerA=0100, powerB=1001, out=0101

#15 clk=1, powerA=0100, powerB=1001, out=1001

#20 clk=0, powerA=0111, powerB=0111, out=1001

#25 clk=1, powerA=0111, powerB=0111, out=0000

#30 clk=0, powerA=0101, powerB=0111, out=0000

#35 clk=1, powerA=0101, powerB=0111, out=1011

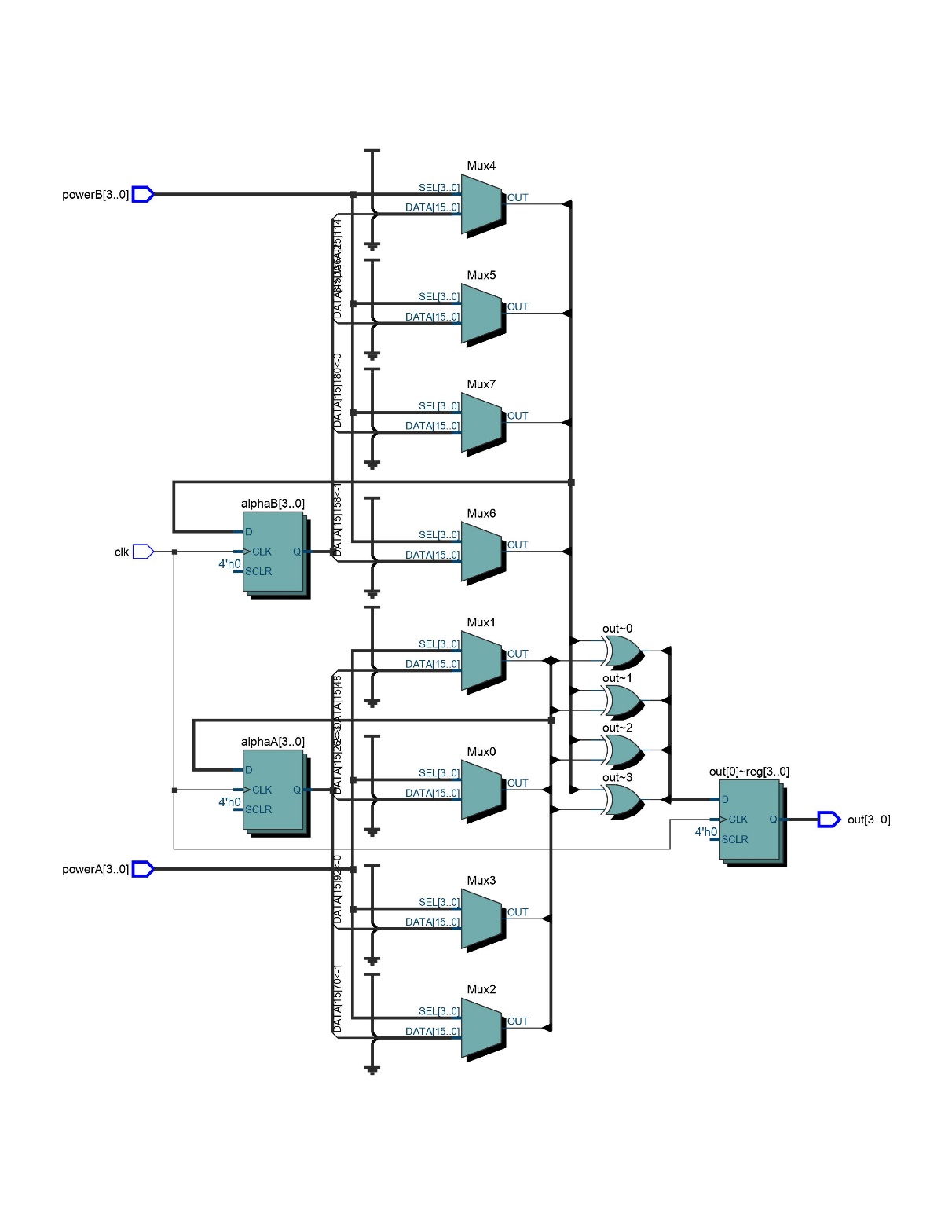
#40 clk=0, powerA=0101, powerB=0111, out=1011

#45 clk=1, powerA=0101, powerB=0111, out=1011

#50 clk=0, powerA=0101, powerB=0111, out=1011

#55 clk=1, powerA=0101, powerB=0111, out=1011

RTL SCHEMATIC:



IO REPORT:

*Control Signals:*

INPUT CLK

INPUT [3:0]powerA, powerB

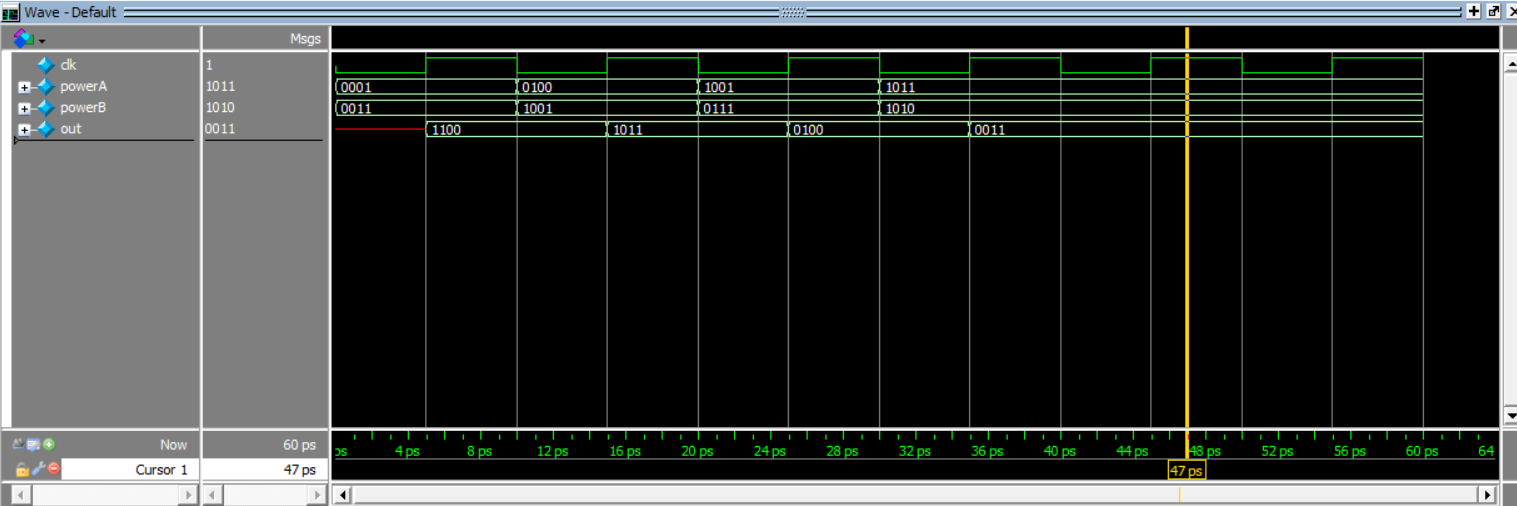
OUTPUT [3:0]out

*Two reg variables:*

4-bit “alphaA” , 4-bit “alphaB”

GALOIS FIELD MULTIPLICATION:

Waveform:



OUTPUT (COMMAND WINDOW):

#0 clk=0, powerA=0001, powerB=0011, out=xxxx

#5 clk=1, powerA=0001, powerB=0011, out=1100

#10 clk=0, powerA=0100, powerB=1001, out=1100

#15 clk=1, powerA=0100, powerB=1001, out=1011

#20 clk=0, powerA=1001, powerB=0111, out=1011

#25 clk=1, powerA=1001, powerB=0111, out=0100

#30 clk=0, powerA=1011, powerB=1010, out=0100

#35 clk=1, powerA=1011, powerB=1010, out=0011

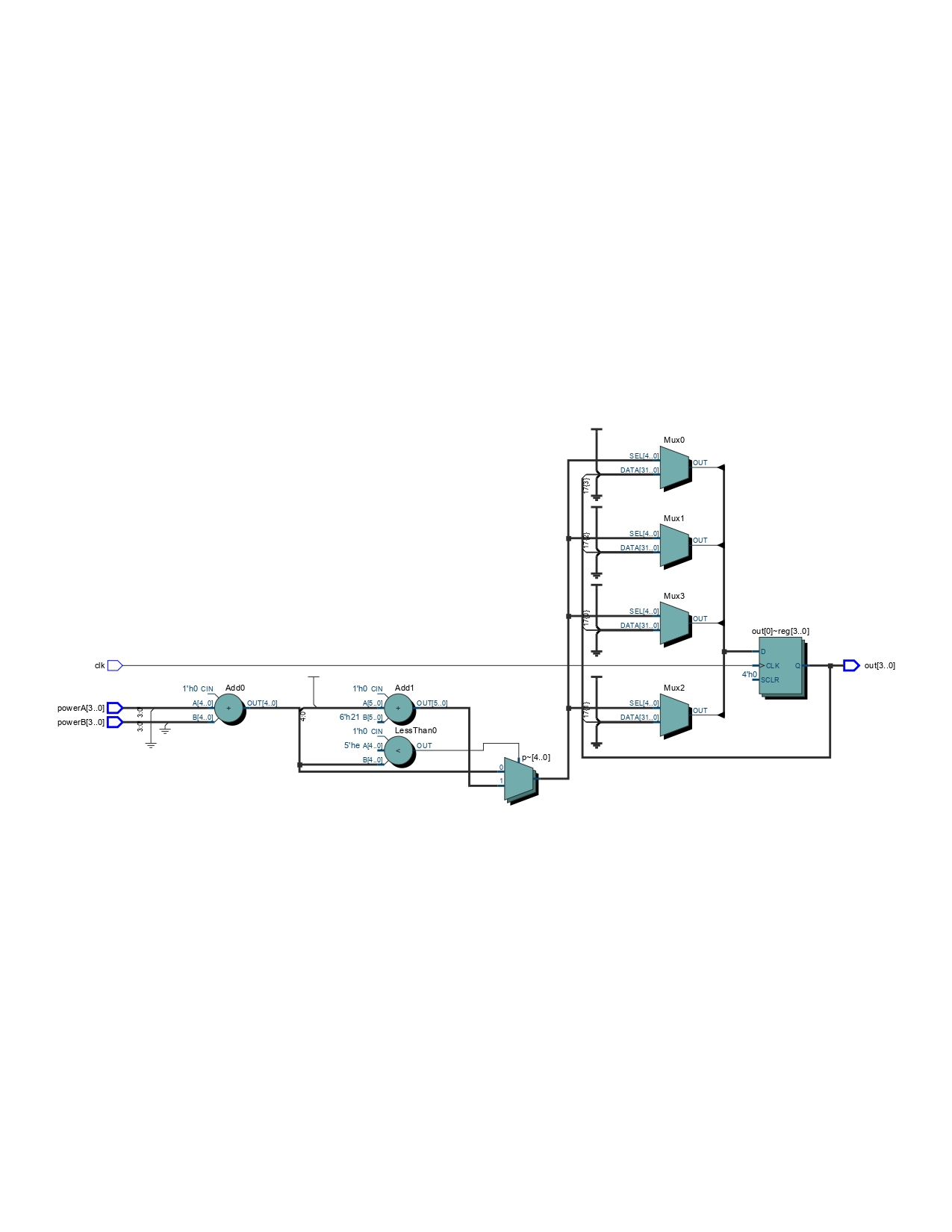
#40 clk=0, powerA=1011, powerB=1010, out=0011

#45 clk=1, powerA=1011, powerB=1010, out=0011

#50 clk=0, powerA=1011, powerB=1010, out=0011

#55 clk=1, powerA=1011, powerB=1010, out=0011

RTL SCHEMATIC:



IO REPORT:

*Control Signals:*

*INPUT CLK*

*INPUT powerA, powerB*

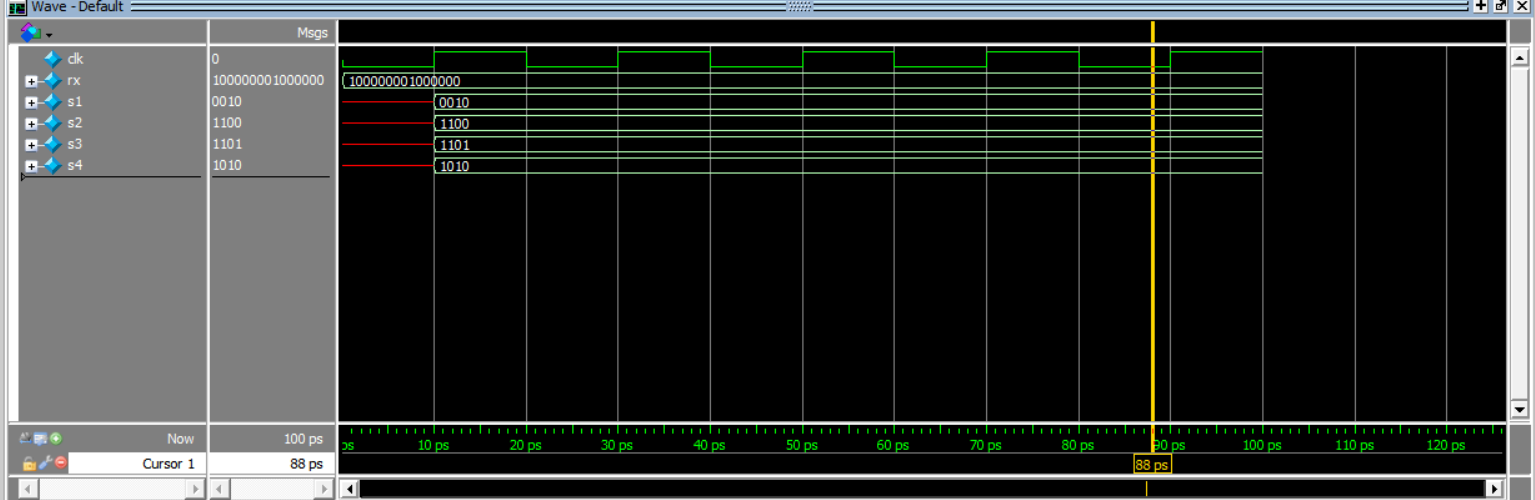
*OUTPUT out*

*reg variable:*

*5-bit “p”*

SYNDROME CALCULATION:

Waveform:

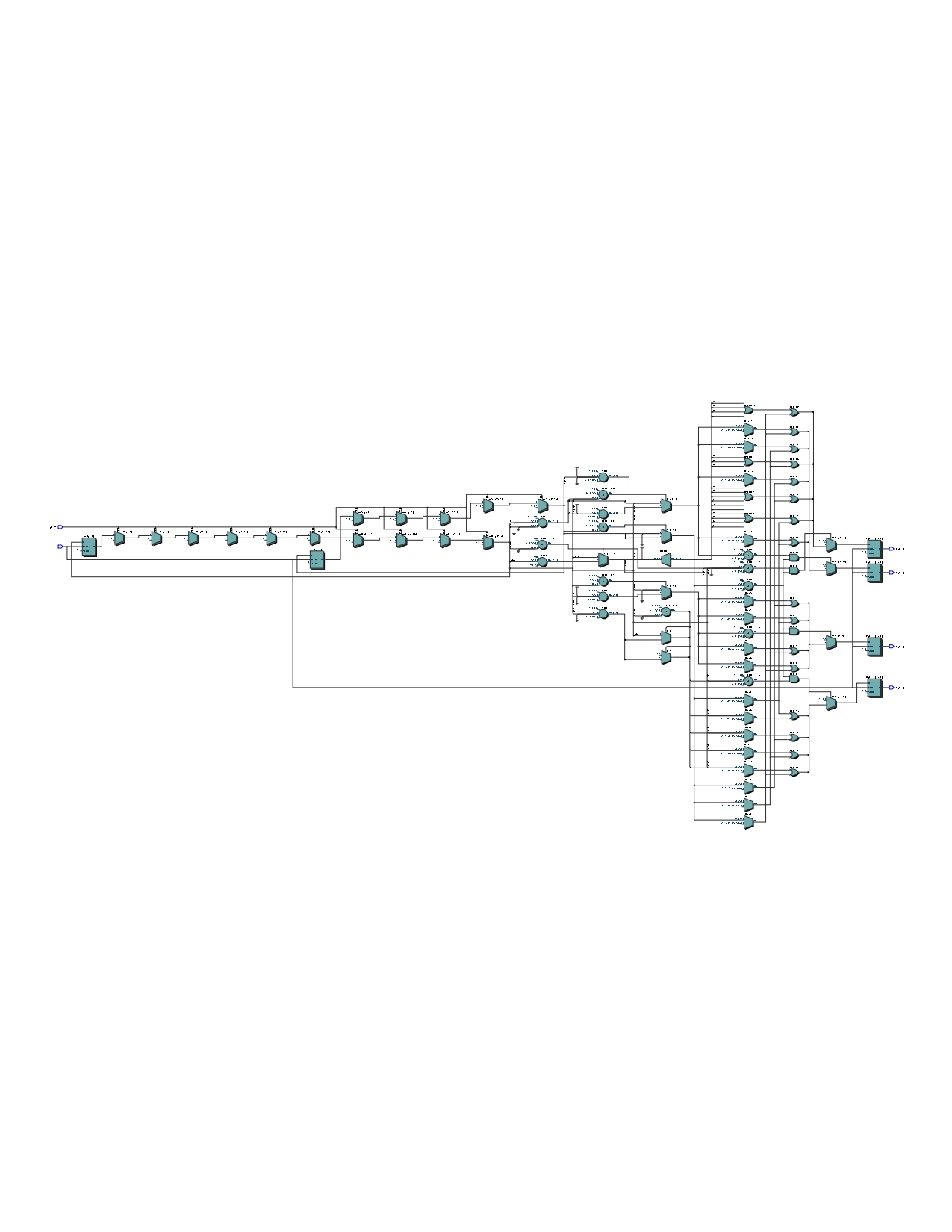


OUTPUT (COMMAND WINDOW):

#0 rx=100000001000000, s1=xxxx, s2=xxxx, s3=xxxx, s4=xxxx

#10 rx=100000001000000, s1=0010, s2=1100, s3=1101, s4=1010

RTL SCHEMATIC:



**APPLICATIONS**

The real life applications of Galois Field Arithmetic are squaring of circle, trisection of most angles and duplication of a cube using a straightedge and compass. They require substantially less than the full content of Galois Theory, but certainly are consequences of it. Most of the quintic equations cannot be solved with radicals and need Galois Field Arithmetic to be used.

Galois Field theory helps us understand finite fields which have numerous real-life applications in cryptography coding theory and combinatorial designs. Galois theory is a beautiful product of human thought and has been used to intrigue and excite thousands of young minds, ultimately leading some of them to develop other mathematical theories that had all sorts of real-life applications.

**CONCLUSION**

In this project, we discuss an efficient finite field arithmetic unit in binary finite field GF (24) which is used in most of the encoding and decoding applications to compute arithmetic operations which are essential for error detection and correction. This project covers the history of Galois field and mathematical equations associated with it. Design of Arithmetic Unit is discussed here which performs mathematical operations like addition, multiplication and syndrome calculation (first step for decoding).

**REFERENCES**

[1] Shu Lin, and Daniel J. Costello, Jr., “Error Control Coding”, Prentice-Hall, New Jersey, 1983.

[2] A.G.WassalM.A.Hasan. VLSI algorithms, architectures andimplementation of a versatile GF (2 m ). IEEE Transactions on computers, Vol. 49 No.10, 2000.

[3] Dong-Ho Lee Ju-Hynn Kim. A compact finite field processor over GF(2m) for elliptic curve cryptography. IEEE, 2002.

[4] LakhendraKumar, Dr. K. L. Sudha,” Implementation of Galois Field Arithmetic Unit on FPGA”, INDIA, Vol. 2, Issue 6, June 2014.

[5] Jung-Lung Lin Chin-Liang Wang. Systolic array implementation of multipliers for finite fields GF (2 m ). IEEE Transactions on Circuits and Systems, Vol. 38, No. 7, 1991.