Chapter-1

Introduction

A radix multiplier is a digital circuit designed to perform high-speed multiplication by processing multiple bits of the multiplier in each cycle. Unlike traditional binary (radix-2) multipliers that handle one bit at a time, radix multipliers—such as radix-4, radix-8, or radix-16—group bits to reduce the number of partial products, thereby accelerating computation. For instance, radix-4 Booth multipliers halve the number of partial products compared to radix-2, enhancing performance and efficiency. These multipliers are vital in digital signal processing, cryptography, and embedded systems where speed and power efficiency are paramount.

Radix multipliers are integral to modern digital systems, especially in applications demanding high-speed arithmetic operations like digital signal processing, cryptography, and embedded systems. Their ability to reduce computation time and power consumption makes them indispensable in designing efficient hardware architectures. As technology advances, hybrid approaches combining different radix techniques are being explored to further optimize performance and energy efficiency in complex digital circuits.

The implementation of radix multipliers, particularly using Booth's algorithm, has been extensively studied and applied in various hardware designs. For example, a study on the implementation of a radix-4 (32-bit) Booth multiplier using VHDL demonstrated significant improvements in computation time, achieving a delay of 26.32 ns, which is notably lower compared to traditional designs. Similarly, the design and comparison of high-speed radix-8 and radix-16 Booth's multipliers have shown that higher radix implementations can lead to further reductions in partial products and improved performance. These advancements highlight the importance of radix multipliers in achieving efficient and high-speed multiplication in digital systems

Chapter 2

Literature Survey

- Design and Implementation of Radix 4 Based Arithmetic Operations,
  Authors:- Saste and A. Sawant , published in 2019.
  Focuses on improving Arithmetic efficiency in Digital Systems

- Design Of High Performance Configurable Radix-4 Booth Multiplier Using Cadence Tools,
  Author:- Dr. T. Esther Rani , published in 2022.
  Designs a configurable Radix-4 Booth multiplier with a Hybrid Adder in Cadence, achieving 17.1ns delay (29% faster than Radix-2) and 1.12mW power.

- Modified Booth Encoding Radix-4 8-bit Multiplier,
  Authors:- Da Huang, Afsaneh Nassery, published in 2020.
  Implements an 8-bit Radix-4 Booth multiplier in 0.5µm CMOS, using Booth Encoder + CLA, achieving 0.2092mW power and 38-gate delay.

- A Fast Multiplier Using Modified Radix-4 Booth Algorithm With Redundant Binary Adder For Low Energy Applications ,
  Authors:- R. Mallikarjuna Sharma, K. Raju , published in 2018.
  Proposes a low-energy Radix-4 multiplier using a modified Booth algorithm and Redundant Binary Adder, simulated in Cadence (180nm) to reduce power and delay.

Chapter 3

Objectives & Problem Statement

Objectives:

- To Understanding the Radix-4 Multiplier Concept:
- To Implementation of Radix-4 Multiplier in Cadence Virtuoso
- To analyse the output waveform
- To verify with the manual calculation

Problem Statement:

The radix multiplier, while effective in handling large-scale multiplications, presents several significant challenges in its implementation. Firstly, it requires a large area on the chip due to the extensive number of logic gates and components involved in its architecture. This substantial area requirement can limit its suitability for compact or resource-constrained systems. Secondly, the high number of gates also contributes to increased power consumption, making the radix multiplier less energy-efficient, especially in battery-powered or low-power devices. Lastly, the complexity of the circuit introduces a relatively long delay time, as multiple processing stages and operations are required to complete a multiplication. These issues—large chip area, high power usage, and longer delays—collectively affect the performance and practicality of radix multipliers in certain applications.

Chapter 4

Block diagram & Implementation

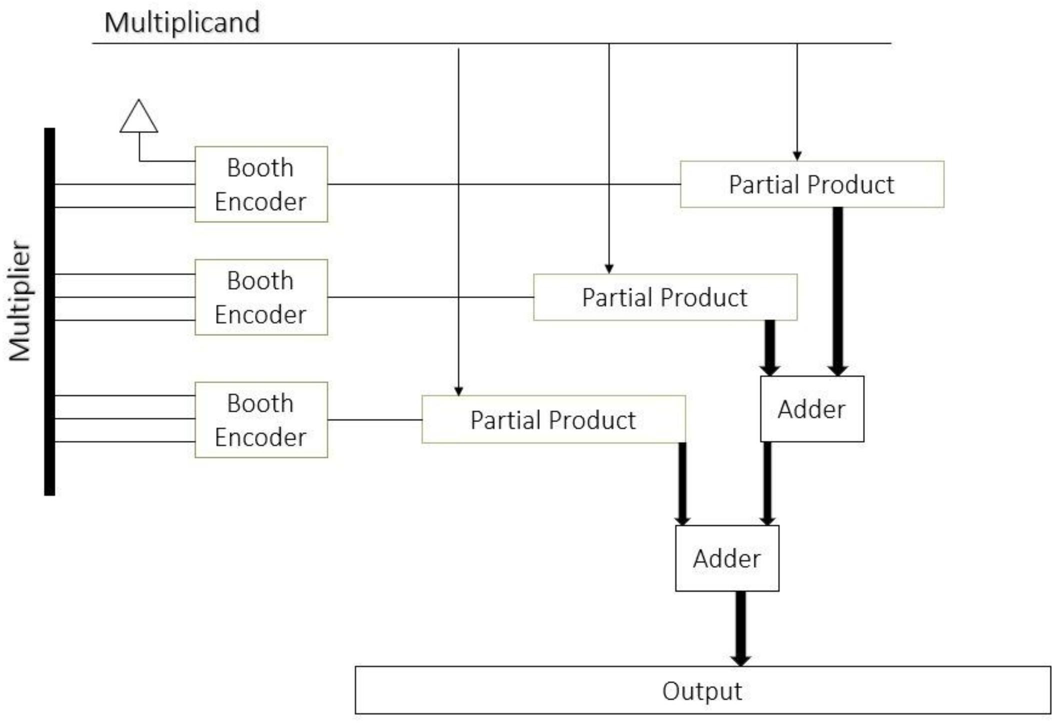
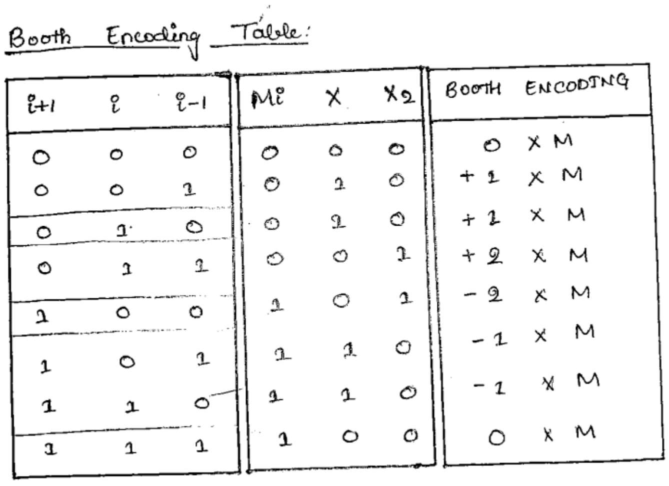


Fig. 4.1: Block-diagram of the Radix-4 Multiplier

The radix multiplier uses Booth's algorithm to perform efficient binary multiplication by encoding the multiplier into groups of bits (typically 2 for Radix-4). Each group is processed by a Booth encoder to determine the corresponding operation on the multiplicand (e.g., add, subtract, or shift). This reduces the number of partial products. These partial products are then aligned according to their bit significance and summed using a series of adders. The methodology minimizes the number of addition operations, improving speed and reducing hardware complexity, making it ideal for signed multiplication in digital systems such as DSPs and processors. The block diagram in the fig. 4.1 is explained in detail below.

1. **Booth Encoder**: A Booth encoder examines adjacent bits of a binary multiplier to determine whether to add, subtract, or skip the multiplicand,

optimizing signed binary multiplication. The Fig. 4.2 explains about the booth encoding of the multiplier



Booth Encoding Table:

i+1	i	i-1	Mi	X	X2	BOOTH ENCODING
0	0	0	0	0	0	0 × M
0	0	1	0	1	0	+1 × M
0	1	0	0	1	0	+1 × M
0	1	1	0	0	1	+2 × M
1	0	0	1	0	1	−2 × M
1	0	1	1	1	0	−1 × M
1	1	0	1	1	0	−1 × M
1	1	1	1	0	0	0 × M

Fig. 4.2: Booth Encoding Table

$$M_i = X_1;$$
$$X = X_0 \oplus X_{-1};$$
$$X_2 = \overline{X}_1 X_0 X_{-1} + X_1 \overline{X}_0 \overline{X}_{-1}$$

These are the equations that are used in making the schematic of the booth encoder based on the booth encoding table as mentioned in the fig. 4.2

2. **Multiplier:** Multiplier refers to how the multiplier operand (the number being multiplied) is processed to reduce the number of partial products, which improves speed and efficiency. Instead of evaluating the multiplier one bit at a time (as in the basic binary or Radix-2 approach), Radix-4 processes two bits at a time, effectively handling three bits due to the need for overlapping in Booth encoding. This encoding technique examines overlapping groups of three bits from the multiplier to determine the necessary multiples of the multiplicand (such as 0, ±1, ±2) to use in each step. The result is a significant reduction in the number of partial products roughly by half compared to Radix-2. This speeds up multiplication, especially in digital signal processors and high-speed processors. The multiplier part, therefore, includes logic to generate these overlapping groups, apply modified Booth encoding, and control the sequencing of partial product generation based on the encoded digits.

3. **Multiplicand:** In radix multipliers, the multiplicand serves as the fixed operand that is repeatedly combined with encoded segments of the multiplier to generate partial products. These partial products are then appropriately shifted and summed to produce the final multiplication result.
4. **Partial Product:** In radix multipliers, the partial product function generates intermediate results by multiplying the multiplicand with encoded segments of the multiplier, as determined by algorithms like Booth's encoding. These partial products are then aligned according to their respective weights and summed to produce the final multiplication result.

$$PP_{ij}=(Y_j X\overline{X_2}+Y_{j-1}\overline{X}X_2)\oplus M_i$$

The above mentioned equation is used to generated a single bit of partial product. Series connection of this each block along with the half adder is used in making a 10 bit Partial Product block in the Cadence.

5. **Adders:** In radix multipliers, adders are crucial for efficiently summing partial products to produce the final result. Various adder architectures are employed to optimize speed and area. 12 bit Carry look ahead adder which is created by using the three 4bit CLA adder connected in series is being used in the present multiplier. A Carry Look-Ahead Adder (CLA Adder) is a type of digital adder used in computers to quickly add two binary numbers. In regular adders like the ripple carry adder, each bit has to wait for the previous carry to be calculated before it can be added. This causes a delay, especially when adding large numbers. The CLA adder solves this problem by calculating the carry values in advance, without waiting. It does this using special logic to "look ahead" and figure out whether each bit will generate or pass on a carry. Because it does not have to wait for each bit, the CLA adder is much faster, especially in large binary additions.

Flow-Chart:

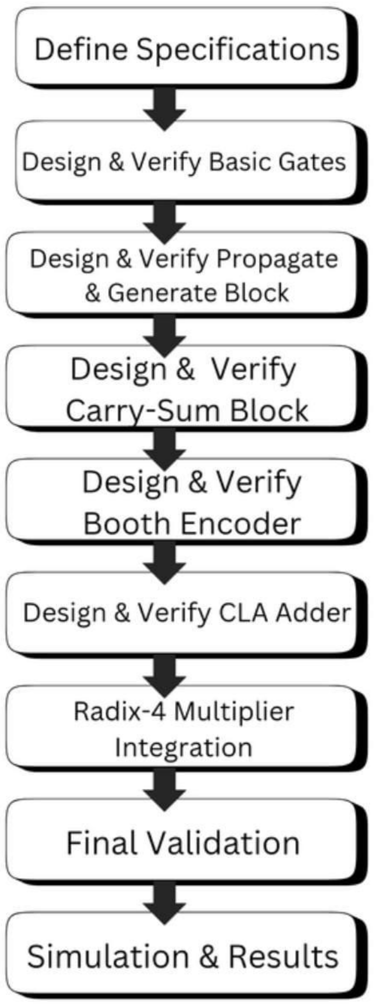

Define Specifications
Design & Verify Basic Gates
Design & Verify Propagate & Generate Block
Design & Verify Carry-Sum Block
Design & Verify Booth Encoder
Design & Verify CLA Adder
Radix-4 Multiplier Integration
Final Validation
Simulation & Results

Fig. 4.3: Flow-chart of the methodology used

This flow diagram outlines the step-by-step design and verification process for a Radix-4 Multiplier, which is a type of high-speed multiplier commonly used in digital signal processing and arithmetic logic units. Below is a paragraph-wise explanation of each block.

1. Define Specifications

This initial step involves defining the functional and performance requirements of the Radix-4 multiplier. These specifications may include the bit-width of inputs and outputs, operating frequency, area constraints, power consumption limits, and technology node. Clear specifications serve as a blueprint for all subsequent design steps.

2. Design & Verify Basic Gates

Before constructing complex blocks, basic logic gates (AND, OR, NOT, XOR, etc.) are designed and tested. Ensuring the correctness of these fundamental components is critical because they form the basis of all higher-level modules in the multiplier architecture.

3. Design & Verify Propagate & Generate Block

In this phase, the propagate and generate logic used for carry computation is designed. This is essential for fast addition operations, especially in carry-look ahead adders (CLAs) or carry-save adders used within the multiplier. This block identifies how the carries will move through the adder, a key function for reducing overall delay.

4. Design & Verify Carry-Sum Block

The carry-sum block is typically part of a carry-save adder (CSA), where partial products are reduced efficiently without immediate carry propagation. This step verifies the block that computes intermediate sums and carries, which are later resolved into the final product.

5. Design & Verify Booth Encoder

The Booth encoder is integral to Radix-4 multiplication. It reduces the number of partial products by encoding the multiplier operand using Booth's algorithm. This step designs and validates the logic that determines how to recode the multiplier bits for optimized multiplication.

6. Design & Verify CLA Adder

The carry-look ahead adder (CLA) is designed and verified next. This high-speed adder resolves the final addition of carry and sum values from the CSA or partial product stage. Its inclusion ensures rapid final summation, critical for performance.

7. Radix-4 Multiplier Integration

After individual blocks are verified, they are integrated into the complete Radix-4 multiplier system. This involves wiring the Booth encoder, partial product generators, carry-save adders, and final CLA adder together in the proper sequence.

8. Final Validation

This stage validates the entire integrated design. Functional and timing simulations are run to ensure the multiplier performs accurately under all input conditions and meets specified timing requirements.

9. Simulation & Results

The final step involves extensive simulation using test benches and input vectors. Performance metrics like propagation delay, area, and power consumption are analyzed. The results determine if the design meets the initial specifications or requires optimization.

Chapter-5

Software tools Used

Software:

Fig. 5.1: Cadence Virtuoso logo

The Software we have used in this project is Cadence Virtuoso.
Cadence Virtuoso is a leading software platform used for custom IC (integrated circuit) design, particularly in analog, mixed-signal, RF, and custom digital circuits. It provides a comprehensive suite of tools for schematic capture, layout design, simulation, and verification. Engineers use Virtuoso to create highly accurate and complex chip layouts, ensuring optimal performance and manufacturability. Its integration with simulation tools like Spectre enables precise electrical analysis. Virtuoso supports advanced process technologies, automation features, and collaborative workflows, making it essential in the semiconductor industry for designing and validating chips used in mobile devices, automotive systems, IoT, and high-performance computing applications.

Chapter-6

Photographs of the circuit & Simulation Results

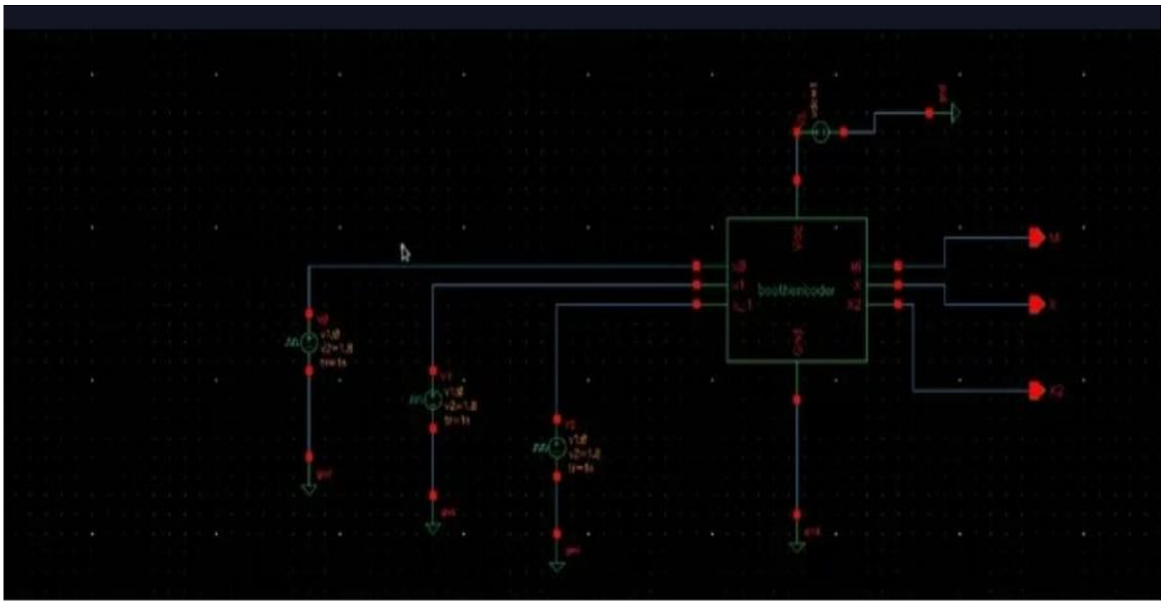


Fig. 6.1: Booth encoder

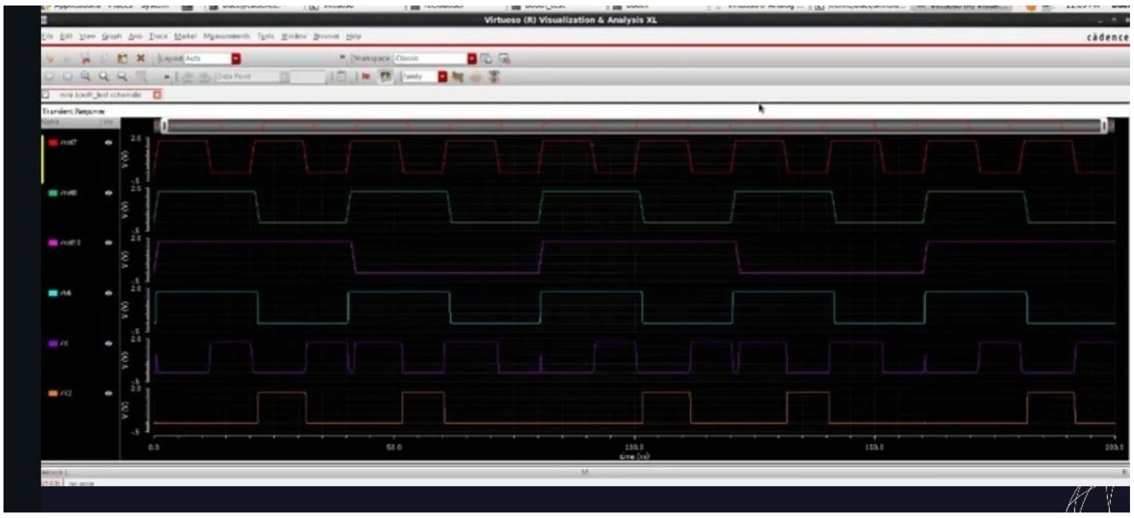


Fig. 6.2: Waveform of Booth encoder

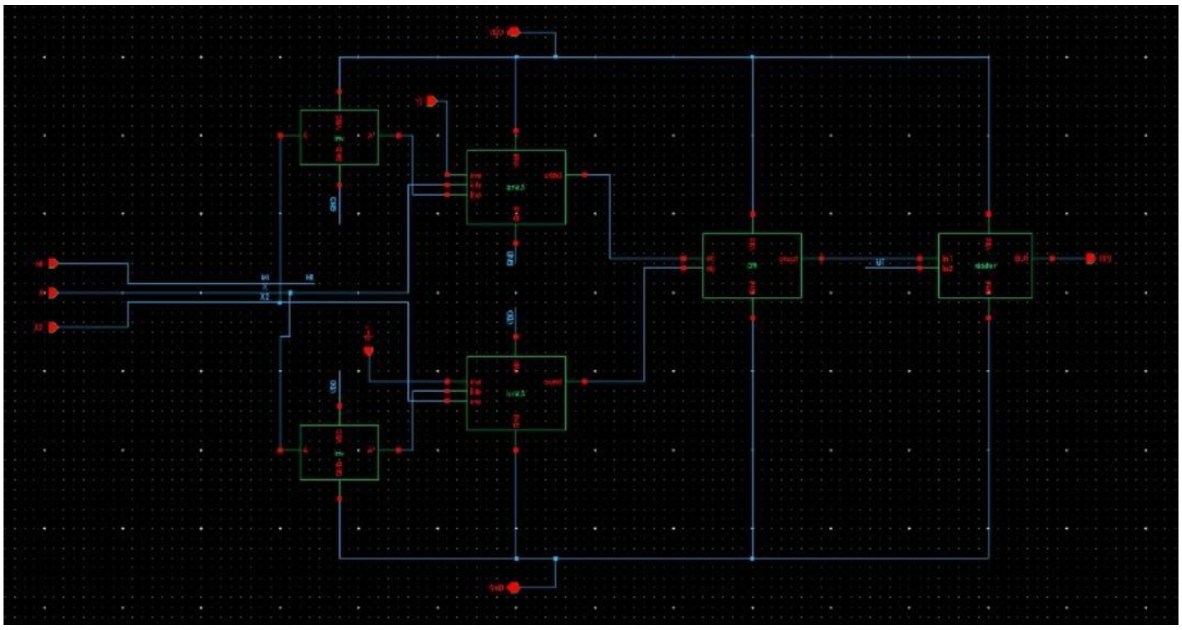


Fig. 6.3: Partial Product Generator for single bit

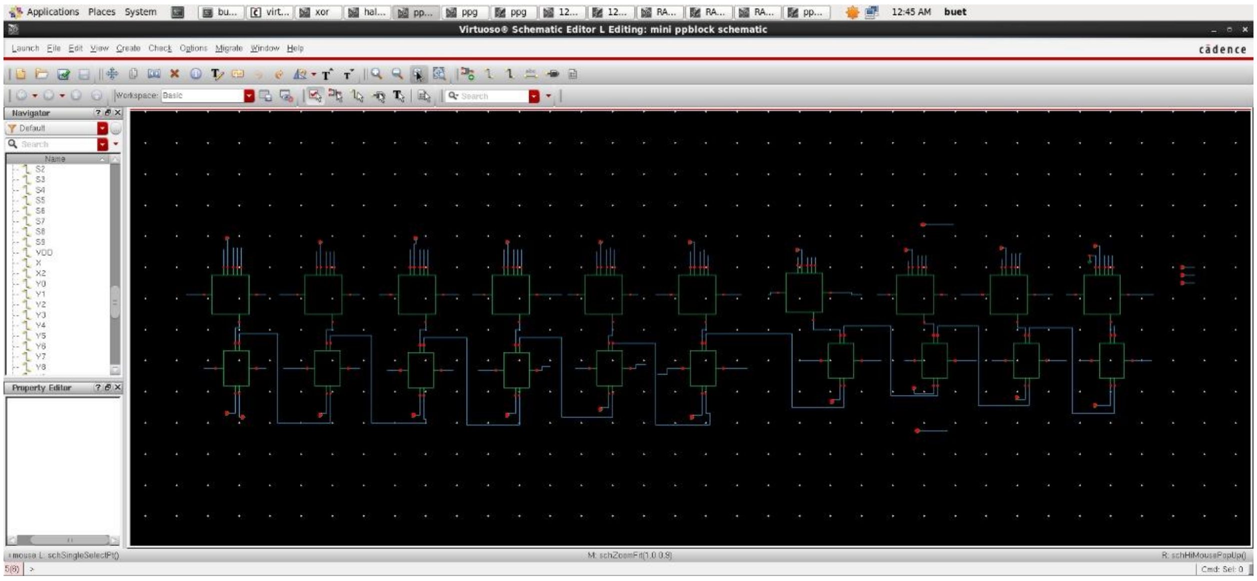

Applications Places System
Virtuoso® Schematic Editor L Editing: mini ppblock schematic
Launch File Edit View Create Check Options Migrate Window Help
cādence
Workspace: Basic
Navigator
Property Editor

Fig. 6.4: Partial Product Block

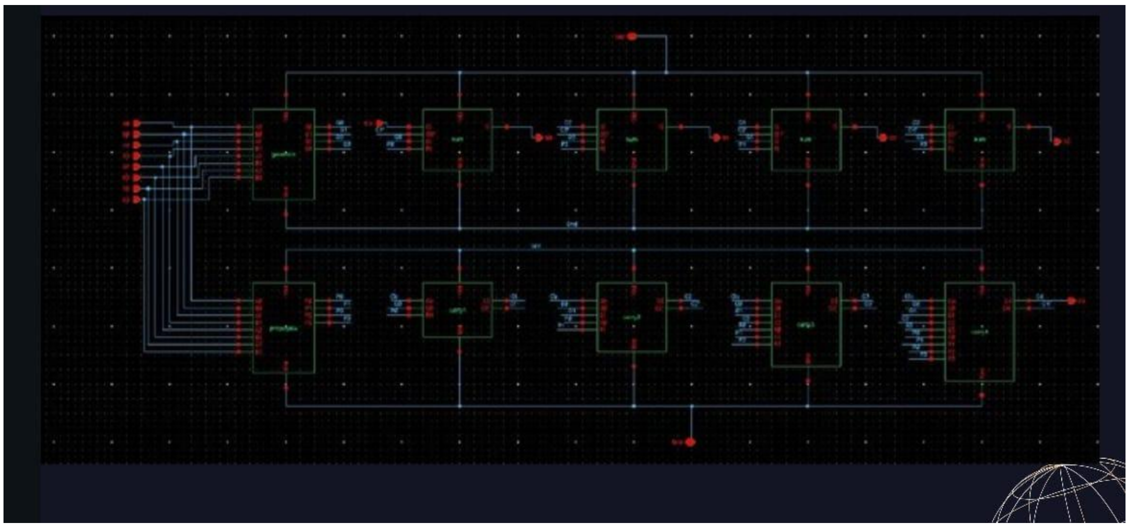


Fig. 6.5: CLA adder

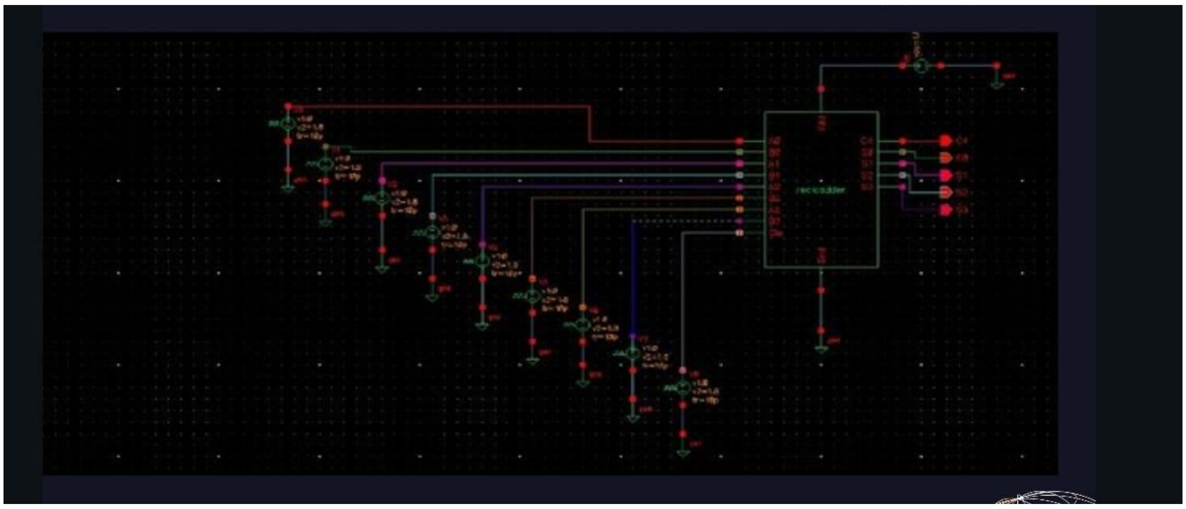


Fig. 6.6: Symbol of the CLA adder

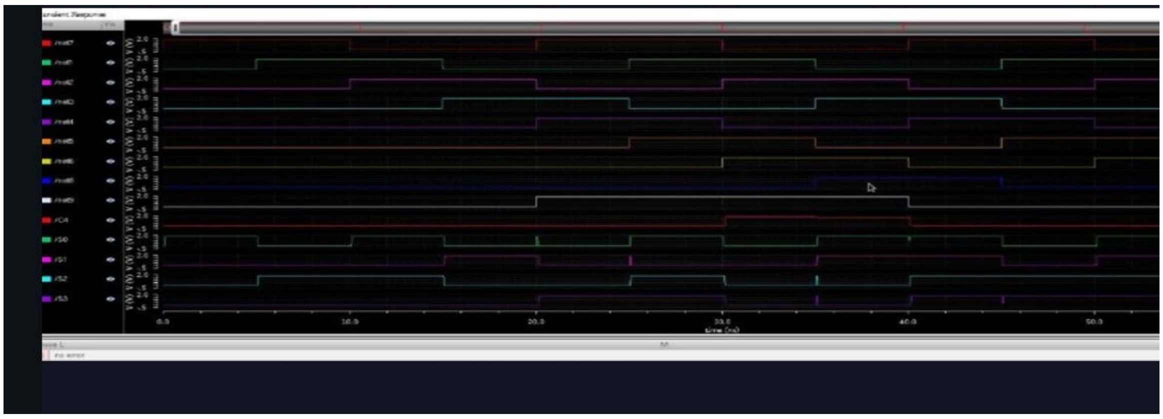


Fig. 6.7: CLA adder waveform

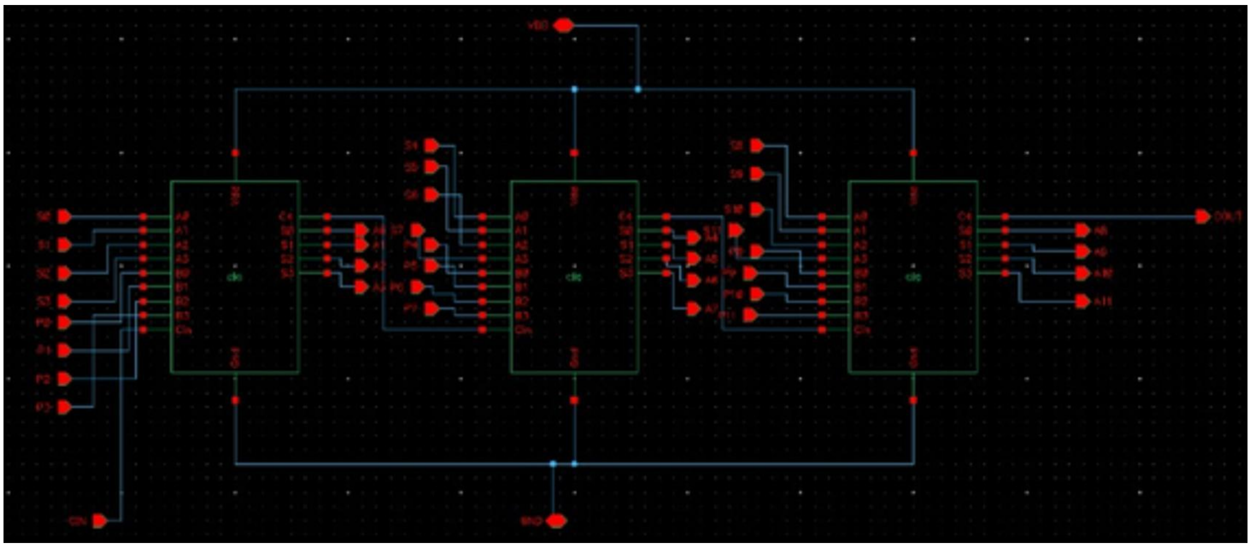


Fig. 6.8: 12 bit adder block

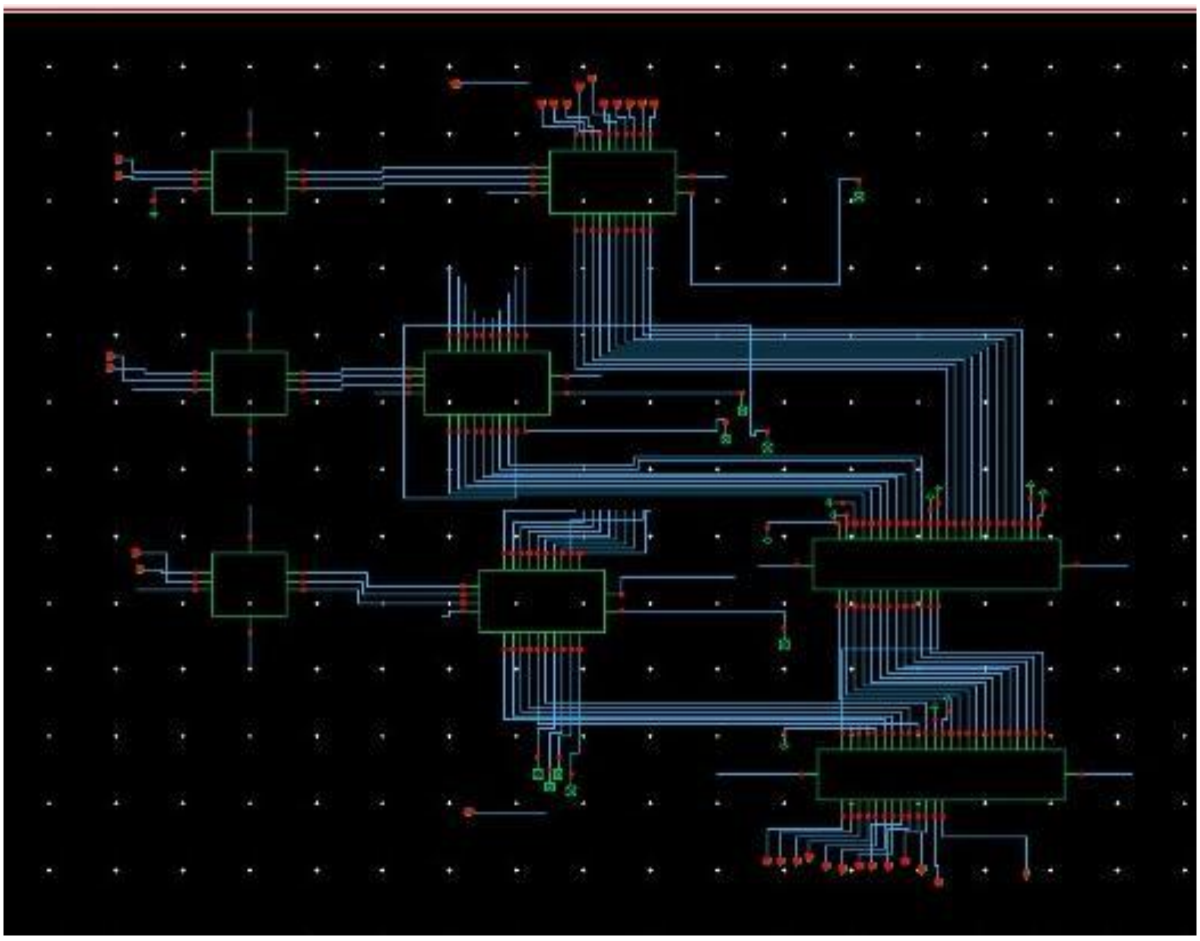


Fig. 6.9: Radix 4 Multiplier

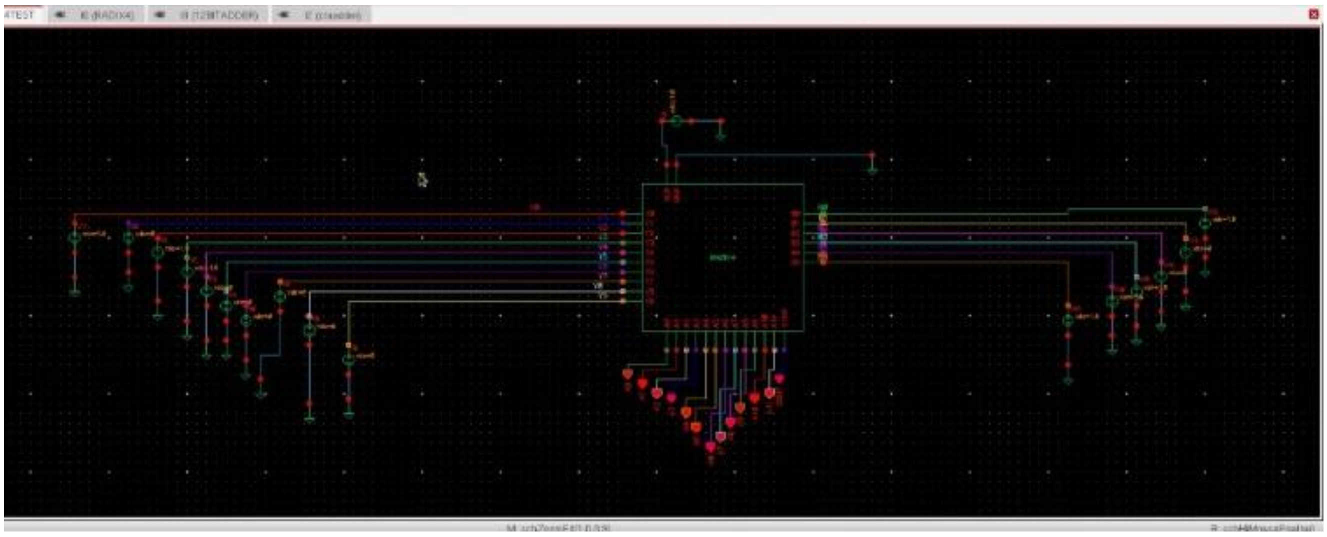


Fig. 6.10: Radix 4 Multiplier test circuit

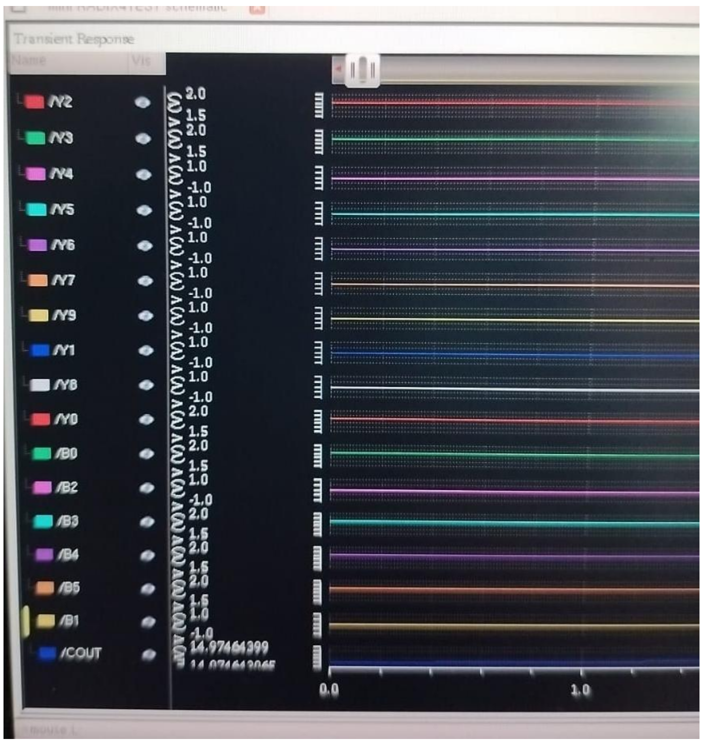


Fig. 6.11: Radix 4 Multiplier input waveform

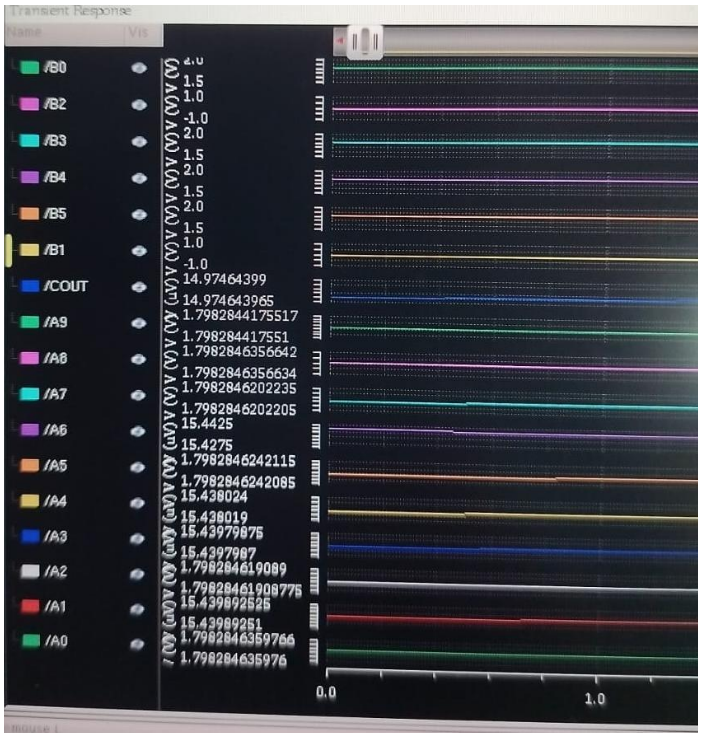


Fig. 6.12: Radix-4 Multiplier output waveform

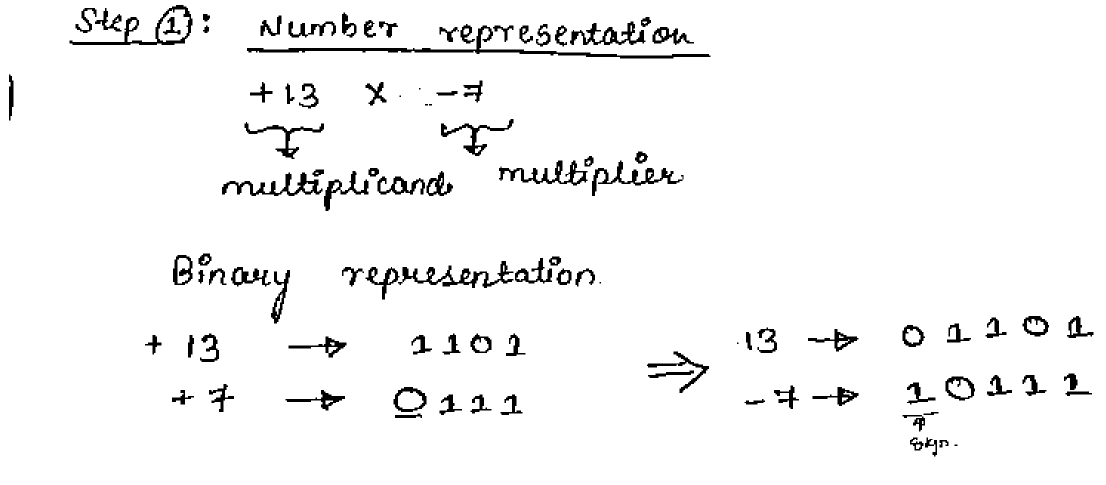
Chapter-7

Results and Discussions

The output waveform of a Radix-4 multiplier provides valuable insight into the functional correctness, timing behavior, and performance of the design. Typically, the waveform will show input signals (such as the multiplicand and multiplier), control signals (like clock and reset), and the final output product. By analyzing the waveform, we can verify whether the output product matches the expected value for given inputs. If the partial product generation and addition stages are working correctly, the output will stabilize to the correct result within a few clock cycles, depending on the implementation. The vdc is provided as input for the radix 4 multiplier for both multiplicand and for the booth encoder. The voltage for the high input is gives as 1.8V and the voltage for the low input is given as 0V for the symbol of the Radix-4 Multiplier. The output is obtained with the minimal losses as shown in the waveform.

Manual Calculation:

Step 1: The decimal number is represented in binary form as shown in fig. 7.1



Step ①: Number representation

$\underbrace{+13}_{\text{multiplicand}} \times \underbrace{-7}_{\text{multiplier}}$

Binary representation

$+13 \rightarrow 1101$

$+7 \rightarrow \underline{0}111$

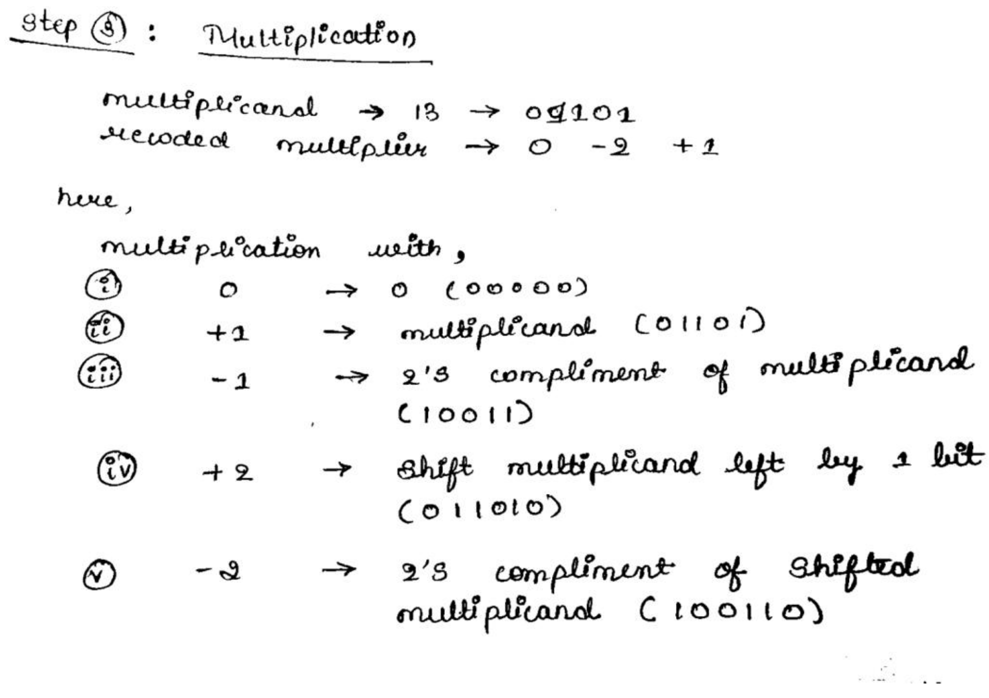
$\Rightarrow$

$13 \rightarrow 01101$

$-7 \rightarrow \underset{\text{sign}}{\underline{1}}0111$

Fig. 7.1: Number Representation in Binary

The Fig. 7.2 explains about the functions for every recoded multiplier



Step ③ : Multiplication

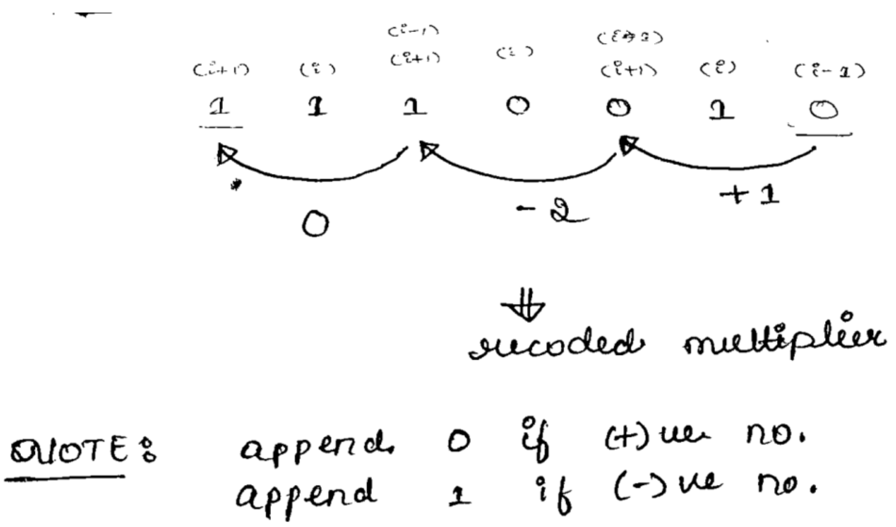
multiplicand → 13 → 01101
recoded multiplier → 0 -2 +1

here,
multiplication with,

(i) 0 → 0 (00000)
(ii) +1 → multiplicand (01101)
(iii) -1 → 2's compliment of multiplicand (10011)
(iv) +2 → Shift multiplicand left by 1 bit (011010)
(v) -2 → 2's compliment of shifted multiplicand (100110)

Fig. 7.2: Booth encoding explanation

Step 2: The multiplier is converted into the recoded multiplier as shown in the fig. 7.3



NOTE: append 0 if (+)ve no.
append 1 if (-)ve no.

Fig. 7.3: Recoded Multiplier

Step 3: Calculations of the Partial Products is shown in the fig. 7.4

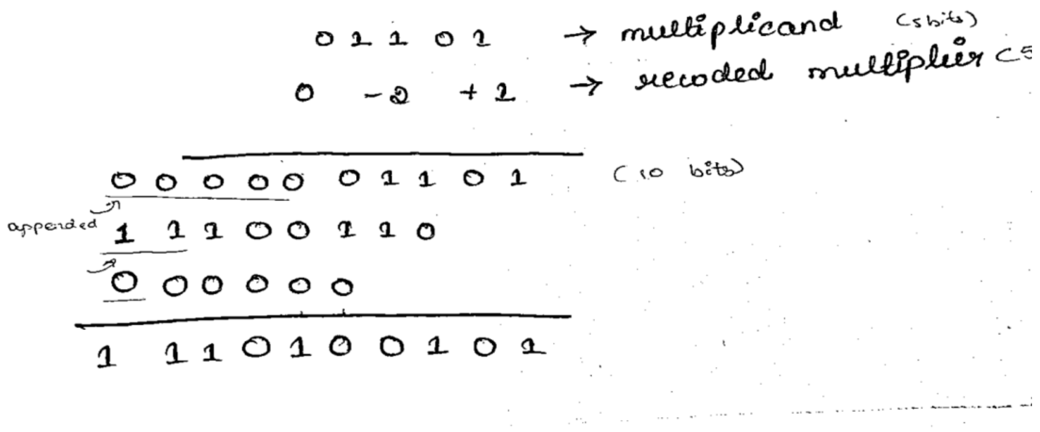

0 1 1 0 1 → multiplicand (5 bits)
0 -2 +1 → recoded multiplier (5
0 0 0 0 0 0 1 1 0 1 (10 bits)
appended 1 1 1 0 0 1 1 0
0 0 0 0 0 0
1 1 1 0 1 0 0 1 0 1

Fig. 7.4: Partial Product Generation and output

Step 4: Verification of the obtained output as shown in fig. 7.5

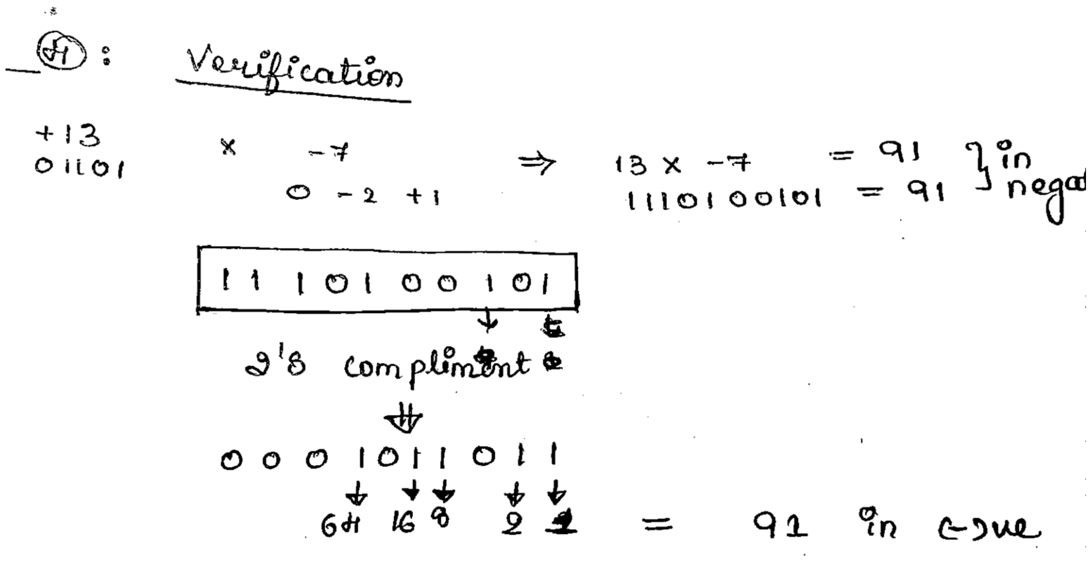


Fig. 7.5: Verification of the Output

Chapter-8

Applications, Advantages, Outcome and Limitations

Applications:

1. **Digital Signal Processing (DSP)** – Used in filters, FFTs, and other DSP algorithms.
2. **Graphics Processing Units (GPUs)** – For fast arithmetic in image rendering.
3. **Embedded Systems** – Where area and power optimization is crucial.
4. **Microprocessors/ALUs** – Speeds up multiplication in CPUs.
5. **Communication Systems** – Used in encoding/decoding and modulation/demodulation.
6. **Control Systems** – For real-time mathematical computations.

Advantages:

1. **Faster Computation** – Reduces the number of partial products (especially in Radix-4 or higher).
2. **Efficient for Signed Numbers** – Handles 2's complement arithmetic directly.
3. **Lower Hardware Complexity** – Compared to full-width multipliers.
4. **Power Efficient** – Fewer switching operations due to fewer add/subtract stages.
5. **Scalable** – Can be extended to higher radix versions (e.g., Radix-8) for further speed-up.

Outcome:

- A radix multiplier produces the product of two binary numbers faster and with fewer resources than basic multipliers, making it suitable for high-speed arithmetic applications in VLSI and real-time systems.

Limitations:

1. **Complex Booth Encoding Logic** – Higher radix values increase control logic complexity.
2. **More Preprocessing Time** – Encoding and decoding stages take time.
3. **Design Overhead** – Difficult to design and optimize for very high radix systems.
4. **Not Ideal for Small Bit-widths** – Simpler multipliers may outperform for low-bit operations.

Chapter-9

Conclusions

The radix multiplier, particularly when implemented using Booth's algorithm (e.g., Radix-4, Radix-8), plays a critical role in modern digital arithmetic circuits. It provides an efficient and optimized method for performing multiplication, which is one of the most fundamental operations in computing systems. The radix-based approach significantly improves performance by reducing the number of partial products generated during the multiplication process. This reduction directly translates into faster computation time and lower power consumption—key metrics for high-performance and embedded system designs.

Radix multipliers are especially useful in systems requiring frequent and rapid arithmetic operations such as digital signal processing (DSP), image and video processing, cryptography, and real-time control applications. By efficiently handling signed binary numbers in two's complement format, they simplify the design complexity while maintaining high computational accuracy.

One of the primary advantages of higher-radix multipliers is their ability to minimize the number of addition and subtraction operations. This makes them well-suited for VLSI implementation where area, speed, and power constraints are critical. However, this efficiency comes at the cost of increased encoding logic complexity, which must be carefully designed to avoid timing bottlenecks and errors.

Despite the limitations, the outcome of using radix multipliers is generally positive. They strike a practical balance between speed, hardware complexity, and power usage, making them favorable over conventional shift-and-add multipliers for medium to large bit-width operations.

In conclusion, radix multipliers offer a high-performance solution for binary multiplication in both general-purpose processors and application-specific integrated circuits (ASICs). As computational demands continue to rise in modern electronics, radix multipliers will remain a cornerstone of efficient arithmetic logic unit (ALU) design, enabling faster, smaller, and more power-efficient digital systems. Their scalability and adaptability ensure their relevance in next-generation processor and hardware accelerator designs.

References:

- S. S. Saste and A. G. Sawant, “Design and Implementation of Radix 4 Based Multiplication on FPGA,” International Journal of Engineering Research & Technology (IJERT), vol. 5, no. 9, pp. 1–5, Sep. 2016.
  Available: https://www.ijert.org/design-and-implementation-of-radix-4-based-multiplication-on-fpga

- T. Esther Rani, “Design of High Performance Configurable Radix-4 Booth Multiplier Using Cadence Tools,” CVR Journal of Science and Technology, vol. 6, pp. 66–74, Jun. 2014.
  Available:https://cvr.ac.in/ojs/index.php/cvracin/article/view/505

- D. Huang and A. Nassery, “Modified Booth Encoding Radix-4 8-bit Multiplier,” Duke University, 2008.
  Available:https://people.ee.duke.edu/~jmorizio/ece261/F08/projects/MULT.pdf

- R. M. Sharma and K. Raju, “A Fast Multiplier Using Modified Radix-4 Booth Algorithm With Redundant Binary Adder for Low Energy Applications,” IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), vol. 8, no. 4, pp. 15–17, Jul.–Aug. 2018.
  Available:https://www.iosrjournals.org/iosr-jvlsi/papers/vol8-issue4/Version-1/F0804015157.pdf