**Decoder Reduction Approximation Scheme for Booth Multipliers**

**PROJECT-21ECP302L**

***Submitted by* GOKUL V [RA2211004010366]**

**SABARI KS [RA2211004010380]**

**ADITYA KONWAR [RA2211004010407]**

*Under the guidance of*

**Dr.E.CHITRA**(Assistant Professor, Department of Electronics & Communication Engineering)

**BACHELOR OF TECHNOLOGY**in

**ELECTRONICS & COMMUNICATION ENGINEERING**of **COLLEGE OF ENGINEERING AND TECHNOLOGY**



S.R.M. NAGAR, Kattankulathur, Chengalpattu District  
**MAY 2025**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this activity report for the course **21ECP302P PROJECT** is the bonafide work of **GOKUL V (RA2211004010366), SABARI KS (RA2211004010380), ADITYA KONWAR (RA2211004010407),** who carried out the work under my supervision.

**SIGNATURE SIGNATURE**

Dr. E.CHITRA

**Guide**  **Academic Co-Ordinator**

Assistant Professor

ABSTRACT

Existing Booth multipliers are widely used in digital signal processing and arithmetic circuits for efficient multiplication operations. However, their complexity and power consumption increase with input width, limiting performance in energy-constrained systems. This project proposes a novel Decoder Reduction Approximation (DRA) scheme that targets the Booth decoder block to achieve significant power, area, and delay reductions. By approximating decoder logic with a carefully selected subset of terms, the scheme maintains acceptable error margins while reducing switching activity and critical path delays. The proposed method shows measurable improvements in hardware efficiency, particularly when synthesized using standard CMOS process technology. Simulation results demonstrate that DRA-Booth multipliers offer a viable trade-off between computational accuracy and hardware savings, making them suitable for low-power applications such as image processing and embedded systems. This approach contributes to developing energy-efficient hardware accelerators for next-generation computing platforms.

The project employs a design-space exploration of approximate decoder circuits to determine optimal configurations based on power-delay-product and error rate metrics. Techniques such as logic simplification, partial truth table evaluation, and transistor-level reduction are used to design lightweight decoder blocks. These approximations are integrated into traditional Booth multiplier architectures and validated through RTL-level simulations and physical synthesis. Experimental evaluations on 8-bit and 16-bit multipliers show up to 28% power savings and 25% area reduction with marginal degradation in accuracy, affirming the practicality of the DRA scheme for approximate computing scenarios.

This project is correlated with **SDG Goal 9 - Industry, Innovation and Infrastructure**, as it fosters the development of innovative multiplier architectures and digital design methodologies to create efficient, scalable, and high-performance hardware.

It is also linked to **SDG Goal 12 - Responsible Consumption and Production**, emphasizing reduced silicon area and power consumption, which are crucial for sustainable electronic product design. The work supports hardware acceleration in energy-sensitive applications such as mobile computing, environmental monitoring, and healthcare devices, thereby promoting sustainable and inclusive technological growth.

TABLE OF CONTENTS



[ABSTRACT iii](#_Toc166248723)

[TABLE OF CONTENTS v](#_Toc166248724)

[LIST OF FIGURES vi](#_Toc166248725)

[ABBREVIATIONS vii](#_Toc166248727)

[1. INTRODUCTION 1](#_Toc166248728)

[1.1. Introduction 1](#_Toc166248729)

[1.2. Objective 2](#_Toc166248730)

1.3. Booth Multiplie**r**  4

1.4 Approximate Computing Techniques 4

1.5 Proposed DRA Booth Multiplier 5

***2*  LITERATURE SURVEY 6**

   2.1. Decoder Reduction Approximation (DRA) Booth Multiplier   6   
   2.2. Fast and Low-Cost Approximate Multiplier for FPGAs using Dynamic 6

Reconfiguration    
   2.3. Design of Approximate Booth Multipliers Based on Error Compensation 7    
   2.4. AMG: Automated Efficient Approximate Multiplier Generator for FPGAs via Optimization 8     
  2.5. Fast Approximate Booth Multiplier for Error-Resilient Applications 9    
   2.6. Energy-Efficient Approximate Booth Multipliers Using Compact Error Compensation Circuit 10

[3 SOFTWARE DESCRIPTION 11](#_Toc166248740)

[3.1. Xilinx ISE.7.1 11](#_Toc166248741)

### 3.2. Cadence 12

[4. METHODOLOGY 13](#_Toc166248748)

[4.1.DRA Scheme Block Diagram 13](#_Toc166248749)

[*4*.2.Architecture Overview 13](#_Toc166248750)

[*4*.*3*. Proposed Architecture 15](#_Toc166248756)

[*4*.*4*. Engineering Standards 17](#_Toc166248757)

[*4*.*5*. Multi-disciplinary Aspect 17](#_Toc166248758)

[5 SIMULATIONS 19](#_Toc166248760)

[6.1. Xylinc simulation 19](#_Toc166248761)

[6.2. Test Bench 22](#_Toc166248762)

6.3 Cadence simulations 23

[6 RESULTS AND OTHER INFERENCES 25](#_Toc166248765)

[7.1. Inference 25](#_Toc166248766)

[7 CONCLUSION AND FUTURE WORK 29](#_Toc166248769)

[8.1. Conclusion 29](#_Toc166248770)

[8.2. Future work 29](#_Toc166248771)

[8.3. Realistic Constraints 30](#_Toc166248772)

[REFERENCES 31](#_Toc166248774)

LIST OF FIGURES

[4.1 DRA Scheme Block Diagram 13](#_Toc166251176)

[4.2 BD16.4-4 Decoders (16 Bits) Block Diagram 15](#_Toc166251177)

[**5.1 Xilinx Project Home Window 19**](#_Toc166251178)

[**5.2 Dra\_booth\_2 20**](#_Toc166251179)

[5.3 **Booth\_Encoder 21**](#_Toc166251180)

[5.4 Partial\_Product\_Generator 21](#_Toc166251181)

[5.5 Product\_accumulator 21](#_Toc166251182)

[**5.6 Dra\_booth\_2\_tb 22**](#_Toc166251183)

[5.7 Tb\_Waveform 23](#_Toc166251184)

[5.8 GUI Schematic 24](#_Toc166251185)

[5.9 Cad\_Waveform 24](#_Toc166251188)

[**6.1 Cell\_report 25**](#_Toc166251189)

[6.2 Gate\_report 26](#_Toc166251190)

[6.3 Power\_report 27](#_Toc166251191)

[6.4 Timing\_report 28](#_Toc166251191)

LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| ABM | Approximate Booth Multipliers |
| ACIs | Application-Specific Integrated Circuit |
| AM | Approximate Multipliers |
| ASIC | Application-Specific Integrated Circuit |
| AXI4 | Advanced eXtensible Interface version 4 |
| BAM | Booth Array Multiplier |
| Booth | Booth Encoding Algorithm |
| CLI | Command Line Interface |
| CNN | Convolutional Neural Network |
| DA | Distributed Arithmetic |
| DMD-CNN | Digital Mammogram Diagnostic Convolutional Neural Network |
| DL | Deep Learning |
| DNN | Deep Neural Network |
| DPR | Dynamic Partial Reconfiguration |
| DSP | Digital Signal Processor / Digital Signal Processing |
| DUT | Design Under Test |
| EDA | Electronic Design Automation |
| FF | Flip-Flop |
| FFT | Fast Fourier Transform |
| FPGA | Field Programmable Gate Array |
| GAN | Generative Adversarial Networks |
| GEMM | General Matrix Multiply |
| GOPS | Giga billion Operations per Second |
| GPU | Graphic Processing Unit |
| GUI | Graphical User Interface |
| HDL | Hardware Description Language |
| HWICAP | Hardware Internal Configuration Access Port |
| ICAP | Internet Content Adaptation Protocol |
| IPI | Intellectual Property Integrator |
| IoT | Internet of Things |
| LOD | Leading One Detector |
| LSB | Least Significant Bit |
| LUT | Look-Up Table |
| MED | Mean Error Distance |
| MHz | Mega Hertz |
| MIPS | Million Instructions Per Second |
| MoC | Model of Computation |
| NLP | Natural Language Processing |
| NMED | Normalized Mean Error Deviation |
| NN | Neural Networks |
| PE | Processing Elements |
| PL | Programmable Logic |
| PS | Processing System |
| PSNR | Peak Signal-to-Noise Ratio |
| RAM | Random Access Memory |
| ReLU | Rectified Linear Unit |
| RNN | Recurrent Neural Networks |
| RTL | Register Transfer Level |
| SNR | Signal-to-Noise Ratio |
| SoC | System On Chips |
| UART | Universal Asynchronous Receiver/Transmitter |
| Verilog | Verification Integrated Logic |
| µs/sample | Microsecond per sample |

CHAPTER 1  
INTRODUCTION

1. Introduction

Approximate computing advocates for optimized hardware design by enabling a controlled level of error in exchange for increases in efficiency. Designers aim to simplify circuit components, eliminating or approximating computations that marginally contribute to final outputs. This has become increasingly popular, particularly in use cases where absolute accuracy is not a requirement—like image processing, object detection, edge detection, and deep neural network inference.

Among the commonly employed architectures in approximating are Booth multipliers and logarithmic multipliers. Both these multipliers convert binary inputs into other representations—Booth encoding or logarithmic scale—prior to calculating the product. This conversion allows for more subtle approximation techniques than mere truncation, usually leading to a better acceptable error distribution.

Approximate Booth multipliers generally use methodologies like mixed-radix encoding, high-radix architectures, partial product truncation, and approximated partial product generators. Logarithmic multipliers, by contrast, generally include dynamic truncation, operand trimming, and hybrid design. Although logarithmic multipliers are mostly optimized for convolutional neural network inference, Booth-based designs show flexibility over a wider range of error-resilient applications. Nevertheless, logarithmic designs tend to offer greater resource savings but at increased approximation errors.

A notable technique, dynamic truncation, finds itself somewhere in the middle of simple truncation and elaborate architectural changes. The method typically employs a leading-one detector (LOD) to identify the most significant bit in the binary input. It keeps only a subset of bits after this leading one for continued computation, concentrating the multiplication on the most influential data and thereby improving performance at the cost of minimal loss in accuracy.

As the push for efficient hardware grows, novel multiplier designs like the Decoder Reduction Approximation (DRA) Booth Multiplier play a key role in balancing performance with resource savings. By simplifying control logic and selectively applying approximation, such designs enhance speed and energy efficiency—making them ideal for modern low-power, high-throughput computing systems.

* 1. Objective

The primary objective of the Decoder Reduction Approximation Scheme for Booth Multipliers is to optimize the performance of Booth multiplication circuits by introducing intelligent approximation techniques within the decoder logic. These optimizations aim to strike a balance between ****hardware efficiency (area, power, and speed)**** and ****computational accuracy****, particularly in the context of ****error-tolerant applications**** such as image processing, machine learning, and multimedia systems. Below are the specific goals and motivations behind this approach.

1.2.1 Reduction of Hardware Complexity

One of the core objectives is to minimize the hardware complexity of the Booth multiplier, especially in the decoding phase. In conventional Booth multipliers, the decoder plays a critical role in generating the partial products from the multiplier bits. However, this component becomes increasingly complex with higher bit-widths and more advanced radix schemes (e.g., radix-4 or radix-8 Booth algorithms). The approximation scheme targets the logic operations within the decoder, aiming to simplify or merge certain decoding functions to reduce the number of logic gates, interconnections, and flip-flops. This ultimately contributes to smaller silicon area, lower production costs, and easier integration into embedded systems and ASICs.

1.2.2 Power Consumption Reduction

In digital circuit design, especially for battery-operated and portable devices, power efficiency is a major design constraint. Booth multipliers, while efficient in operation, still consume significant dynamic and static power due to their decoding and accumulation logic. The proposed approximation scheme introduces logic-level reductions that eliminate unnecessary switching activities and redundant calculations, thereby reducing power consumption. By tolerating slight inaccuracies in the decoder, especially when the impact on the final product is minimal, the circuit can operate more efficiently without a drastic sacrifice in output quality.

1.2.3 Maintaining Acceptable Error Margins

A vital objective of this scheme is to maintain the output within acceptable error bounds despite the approximations. While approximated decoders produce partial products that may slightly deviate from their exact counterparts, the cumulative effect is controlled to ensure the overall multiplication result is still valid for the target application. This is often measured using metrics such as Mean Error Distance (MED) or Normalized Mean Error Deviation (NMED). The scheme is carefully designed to introduce approximations only where the resulting error has minimal or negligible influence on the final output, ensuring application-level reliability.

1.2.4 Performance Enhancement (Speed and Throughput)

Another objective is to enhance the operational speed of the Booth multiplier. The reduction in logic depth and gate count as a result of decoder simplification leads to shorter critical paths, thereby improving the clock frequency and throughput. This is especially valuable in high-speed computing tasks, real-time data processing, and other latency-sensitive applications where performance bottlenecks in multiplication circuits can severely impact the overall system.

1.3 Booth Multiplier

The Booth multiplier is a widely used digital multiplication architecture known for its efficiency in reducing the number of partial products, particularly in signed number multiplication. It employs Booth encoding—a recoding technique that transforms the multiplier operand into a format that enables the use of fewer arithmetic operations during multiplication. By grouping bits of the multiplier and evaluating overlapping windows (typically 3-bit groups), Booth encoding can eliminate redundant operations and improve performance.

There are different variants of Booth encoding such as radix-2, radix-4, and higher-radix schemes, each offering trade-offs between complexity and speed. The primary advantage of Booth multipliers lies in their ability to handle two’s complement numbers directly and reduce switching activity, which contributes to lower power consumption. Due to these strengths, Booth multipliers are a preferred choice in arithmetic-intensive applications like signal processing, embedded systems, and machine learning accelerators.

1.4 Approximate Computing Techniques

Approximate computing introduces controlled inaccuracies into digital circuits to achieve substantial gains in power, speed, and area efficiency. In the context of multipliers, approximation techniques are tailored to reduce the computational burden while maintaining acceptable output quality, particularly in applications where perfect accuracy is not essential.

Common strategies include:

* Truncation: Eliminating less significant bits in the partial products or final summation stage to simplify hardware.
* Dynamic Truncation: Using runtime information, such as the position of the leading one in the input, to decide how many bits to retain. This method adapts the precision based on input significance.
* Mixed-Radix Encoding: Combining radix-2 and radix-4 encoding to optimize the balance between accuracy and efficiency.
* Leading-One Detection (LOD): Identifying the most significant bit set to '1' in an input operand, enabling selective computation on high-impact bits and ignoring low-impact ones.

These techniques collectively contribute to building hardware that is smaller, faster, and more power-efficient while introducing minimal and often imperceptible error—especially important in tasks like image classification and neural network inference.

1.5 Proposed DRA Booth Multiplier

The Decoder Reduction Approximation (DRA) Booth Multiplier represents a targeted optimization of the conventional Booth multiplier design. This method focuses on simplifying the Booth decoder—the logic that interprets multiplier bits and generates control signals for partial product selection. By reducing the complexity of this decoding logic, the DRA approach decreases gate count, propagation delay, and dynamic power consumption.

Moreover, DRA introduces selective approximation in the encoding stage, particularly by applying approximations to the most significant bits (MSBs) where slight inaccuracies tend to have limited practical impact. This selective strategy ensures that while computation becomes faster and more resource-efficient, the resulting errors remain within acceptable bounds for many real-world applications.

The DRA Booth Multiplier is especially advantageous for low-power and high-throughput environments, such as embedded AI accelerators and edge devices, where energy efficiency is critical. Its ability to maintain reasonable accuracy while cutting down on circuit overhead makes it a compelling design in the landscape of approximate arithmetic units.

CHAPTER 2  
LITERATURE SURVEY

**2.1. Decoder Reduction Approximation (DRA) Booth Multiplier**

The Decoder Reduction Approximation (DRA) Booth Multiplier introduces a novel optimization by reducing the complexity of the Booth decoder and selectively introducing approximation in the encoding stage. By simplifying the control logic and applying approximate encoding to the most significant bits—where small inaccuracies have limited impact—the DRA approach achieves notable improvements in power, delay, and silicon area. This makes it especially well-suited for low-power, high-throughput applications where efficiency takes precedence over exact numerical accuracy.

Building upon these principles, the DRA Booth Multiplier leverages the inherent error resilience of applications like image processing and deep neural networks. By focusing approximation efforts on the most significant bits, the design ensures that the overall computational accuracy remains within acceptable bounds for these applications. This targeted approximation strategy allows for significant resource savings without compromising the quality of results in error-tolerant domains.

Furthermore, the DRA approach aligns with the growing demand for energy-efficient hardware accelerators in the era of artificial intelligence and edge computing. By reducing the complexity of the Booth decoder and strategically introducing approximation, the DRA Booth Multiplier offers a compelling solution for applications where power efficiency and performance are paramount, and exact numerical precision is not critical.

**2.2. Fast and Low-Cost Approximate Multiplier for FPGAs using Dynamic Reconfiguration**

Vakili et al. (2023) propose an FPGA-based approximate multiplier optimized for machine learning computations. Utilizing dynamically reconfigurable lookup table (LUT) primitives in AMD-Xilinx technology, the design achieves significant reductions in LUT utilization—64% for signed multiplication and 67% for multiply-and-accumulation configurations—compared to standard Xilinx multiplier cores. Accuracy evaluations on four deep learning benchmarks indicate a minimal average accuracy decrease of less than 0.29% during post-training deployment, demonstrating the practicality of the approach for error-tolerant applications.

The architecture leverages dynamic reconfiguration capabilities of modern FPGAs to adapt the multiplier's structure based on computational requirements. This adaptability allows for efficient resource utilization, enabling the hardware to balance performance and accuracy dynamically. The design's compatibility with INT8 precision further underscores its suitability for contemporary machine learning workloads, where quantization techniques are commonly employed to reduce computational complexity.

By providing an open-source implementation, the authors facilitate broader adoption and further research into dynamically reconfigurable approximate multipliers. The approach exemplifies how hardware flexibility can be harnessed to meet the evolving demands of energy-efficient, high-performance computing in domains such as artificial intelligence and digital signal processing.

**2.3. Design of Approximate Booth Multipliers Based on Error Compensation**

In their 2023 study, researchers present a low-power approximate Booth multiplier that employs complementary error compensation techniques. The design introduces a new Booth encoder that generates positive errors and an approximate Wallace tree structure that produces negative errors. By allowing these errors to compensate for each other, the multiplier achieves improved accuracy compared to existing approximate designs. Experimental results demonstrate reductions in power consumption by 20.62%, delay by 14.45%, and area by 19.68%, highlighting the efficiency of the proposed approach.

The architecture's error compensation strategy addresses the challenge of maintaining computational accuracy while reducing resource usage. By carefully balancing the sources of approximation-induced errors, the design ensures that the overall error remains within acceptable limits for error-tolerant applications. This approach is particularly beneficial for applications like image filtering, where minor inaccuracies do not significantly impact the perceived quality of results.

The study's findings underscore the potential of error compensation techniques in designing energy-efficient approximate multipliers. By strategically introducing and balancing errors within the multiplier's components, designers can achieve significant resource savings without compromising the functional integrity required for various signal processing tasks.

**2.4. AMG: Automated Efficient Approximate Multiplier Generator for FPGAs via Bayesian Optimization**

Li et al. (2023) introduce AMG, an open-source automated approximate multiplier generator for FPGAs driven by Bayesian optimization with parallel evaluation. The tool simplifies exact half adders for initial partial product compression while preserving coarse-grained additions for accumulation. The generated multipliers effectively map to lookup tables and carry chains in modern FPGAs, reducing hardware costs with acceptable errors. Compared to 1,167 multipliers from previous works, AMG's designs offer 28.70%-38.47% improvements in the product of hardware cost and error on average.

AMG's utilization of Bayesian optimization enables efficient exploration of the design space, identifying optimal trade-offs between accuracy and resource utilization. This approach allows for the automated generation of approximate multipliers tailored to specific application requirements, streamlining the design process and reducing development time. The tool's compatibility with modern FPGA architectures ensures its relevance for contemporary hardware design challenges.

By providing a framework for automated, efficient design of approximate multipliers, AMG empowers designers to rapidly develop customized solutions for error-tolerant applications. The tool's ability to balance accuracy and resource efficiency makes it a valuable asset in the pursuit of energy-efficient computing solutions in domains such as machine learning and digital signal processing.

**2.5. Fast Approximate Booth Multiplier for Error-Resilient Applications**

George et al. (2024) present an implementation and analysis of approximate Modified Booth multipliers utilizing approximate 4:2 compressors for partial product reduction. The study evaluates resource usage and delay for both FPGA and ASIC implementations, with accuracy assessed using standard quality metrics. Results indicate that the proposed multipliers offer significantly better accuracy with fewer delays and comparable resource usage compared to existing implementations. For 16-bit multipliers, the design achieves two orders of magnitude less error than the next best implementation, with a 40% reduction in delay.

The architecture's integration of approximate compressors addresses the need for efficient arithmetic units in error-resilient applications. By focusing on reducing delay and maintaining accuracy, the design proves suitable for tasks such as image compression, where performance and resource efficiency are critical. The study's application of the multiplier in discrete cosine transform operations for JPEG compression demonstrates its practical applicability in real-world scenarios.

This work highlights the potential of approximate computing techniques in enhancing the performance of arithmetic units for error-tolerant applications. By carefully designing approximate components, designers can achieve significant improvements in efficiency without compromising the quality of results in applications that can tolerate minor inaccuracies.

**2.6. Energy-Efficient Approximate Booth Multipliers Using Compact Error Compensation Circuit**

Aizaz and Khare (2022) propose an energy-efficient approximate Booth multiplier incorporating a compact error compensation circuit to mitigate truncation errors. The design targets applications like artificial neural networks and image processing, which require high computational workloads within power constraints. By introducing a compact error compensation circuit, the multiplier achieves reduced power consumption while maintaining acceptable accuracy levels. The study demonstrates the effectiveness of the design in balancing energy efficiency and computational accuracy for error-tolerant applications.

The architecture's focus on truncation error mitigation addresses a common challenge in approximate multiplier design. By incorporating a dedicated error compensation circuit, the design ensures that the errors introduced by truncation do not significantly impact the overall computational accuracy. This approach enables the development of multipliers that are both energy-efficient and suitable for applications requiring a balance between performance and precision.

The study's findings contribute to the growing body of research on approximate computing techniques for energy-efficient hardware design. By demonstrating the practicality of incorporating error compensation mechanisms, the work provides valuable insights for designers seeking to develop efficient arithmetic units for error-tolerant applications.

CHAPTER 3  
SOFTWARE DESCRIPTION

### **3.1. Xilinx ISE 7.1i**

Xilinx ISE 7.1i (Integrated Software Environment) is an older but robust development suite for designing digital circuits, particularly targeting Xilinx FPGAs and CPLDs. It provides a cohesive environment for design entry, synthesis, simulation, and implementation of HDL-based projects using Verilog or VHDL. Although it has been largely replaced by Vivado for modern Xilinx devices, ISE remains significant for legacy designs and educational purposes.

ISE 7.1i offers a graphical user interface (GUI) along with command-line tools, enabling both intuitive design interaction and automated script-driven workflows. It includes features like Project Navigator for design management, HDL Editor, synthesis engine (XST), simulator, and timing analyzer. The tool also allows the use of schematic capture for those preferring graphical design methodologies. Xilinx ISE is well-suited for smaller or older FPGA families like Spartan and Virtex series, and was widely adopted in academic research and embedded systems development.

**3.1.1. Features of Xilinx ISE 7.1i**

* Integrated development environment for design entry, synthesis, simulation, and implementation.
* Support for both VHDL and Verilog HDL-based design flows.
* Project Navigator for easy design management and workflow navigation.
* Built-in synthesis tool (XST) for logic optimization and netlist generation.
* Timing analyzer and constraint management tools for precise performance tuning.
* Simulation support through ModelSim or ISim for functional and timing verification.
* Schematic editor for graphical circuit design, suitable for beginners or legacy users.
* Compatible with older FPGA families such as Spartan-3 and Virtex-II/IV series.
* Command-line tool support for batch processing and automation.

### **3.2. Cadence Suite**

Cadence is a leading suite of Electronic Design Automation (EDA) tools used in the design and verification of integrated circuits (ICs), printed circuit boards (PCBs), and complete electronic systems. It provides comprehensive software solutions for digital, analog, and mixed-signal designs, enabling the entire design flow from concept to tape-out. Cadence tools are widely recognized for their accuracy, scalability, and high performance in complex chip development.

Cadence offers a variety of tools such as Virtuoso for analog and custom IC design, Spectre for circuit simulation, and Innovus for digital implementation. It supports hardware description languages like Verilog, VHDL, and SystemVerilog, and integrates with formal verification and static timing analysis tools. The software is used extensively in the semiconductor industry for ASIC and SoC development, as well as in academia for research and teaching. Cadence also provides compatibility with scripting and batch processing for advanced automation and design optimization.

**3.2.1. Features of Cadence**

* Comprehensive EDA platform supporting analog, digital, and mixed-signal IC design.
* Virtuoso platform for custom layout and analog simulation.
* Spectre simulation engine for accurate circuit behavior prediction.
* Innovus tool for high-performance digital place-and-route (P&R) flows.
* Formal verification, timing analysis, and power optimization tools for high-reliability design.
* Support for leading hardware description languages: Verilog, VHDL, and SystemVerilog.
* Integration with scripting environments (TCL, SKILL) for customizable workflows.
* Advanced visualization and debugging tools for efficient verification.
* Widely adopted in ASIC and SoC development pipelines across the semiconductor industry

CHAPTER 4  
METHODOLOGY

4.1.DRA Scheme Block Diagram

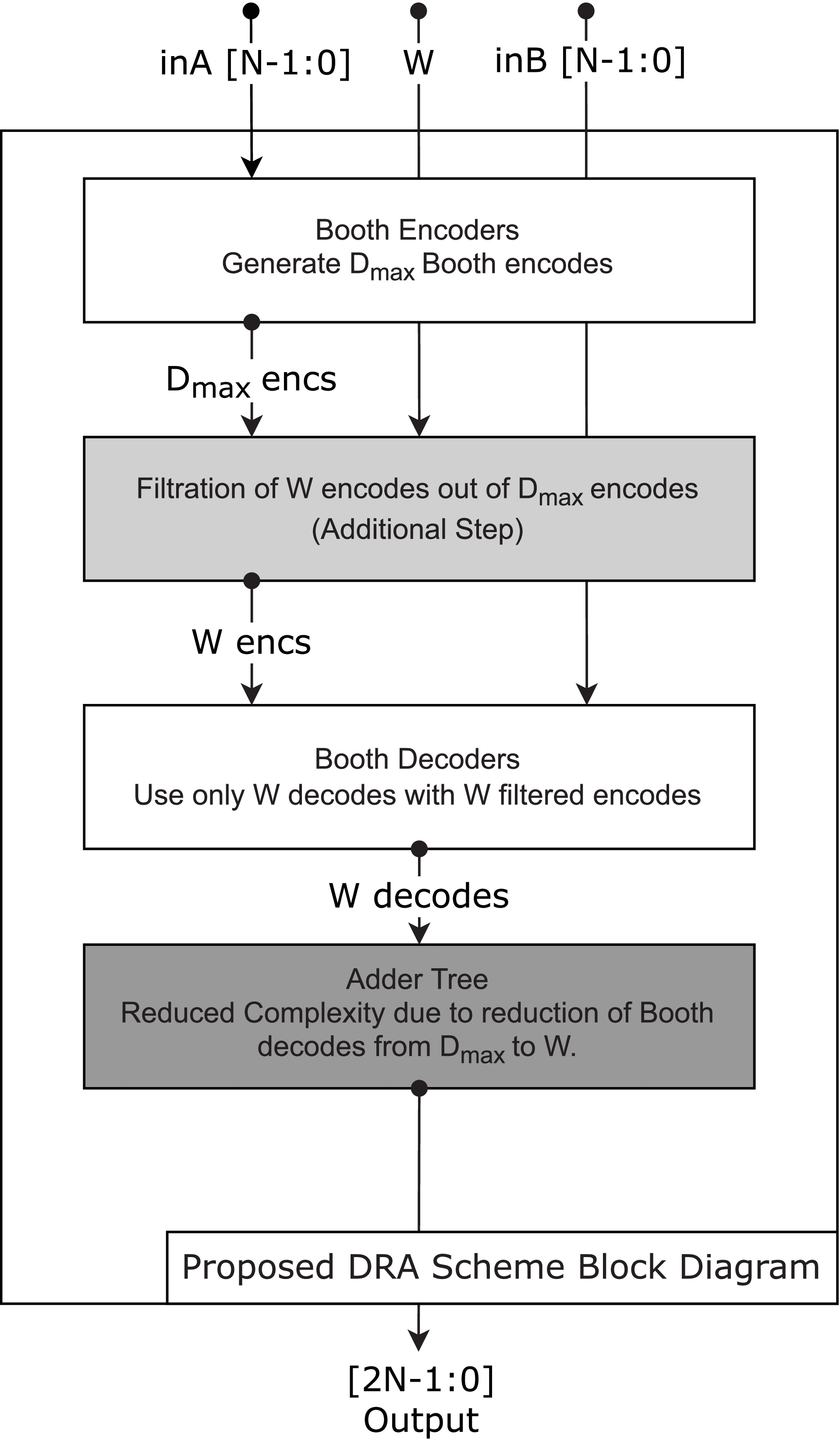


Fig 4.1: DRA Scheme Block Diagram

4.2. Architecture Overview

In Fig. 5.1, The architecture of the proposed **Decoder Reduction Approximation (DRA) Booth Multiplier** is built upon the traditional Booth multiplication process, with a key innovation introduced at the decoding stage to reduce computational complexity. The main design modification centers around filtering and selectively using only a subset of the encoded outputs to reduce logic overhead without significantly affecting accuracy.

* **Input Stage**

The system accepts two *N-bit input operands*, labeled inA[N-1:0] and inB[N-1:0], representing the multiplicand and multiplier, respectively. These inputs are fed into a set of **Booth Encoders**.

* **Booth Encoding**

The encoders generate a maximum number of Booth encodings, denoted as D\_max encs. This represents all the possible partial products derived from the multiplication.

* **Encoding Filtration Step (DRA Innovation)**

An additional step unique to the DRA scheme filters out only W significant encodes from the D\_max set. This **filtration logic** serves as the key approximation technique, selectively passing through only those encodings that contribute most meaningfully to the final result. The remaining less significant encodes are omitted, reducing the downstream processing load.

* **Booth Decoding**

The filtered W encs are passed into the **Booth Decoders**, which process only the selected set of encodes. This step generates W decodes, significantly reducing the overall number of decoding operations required compared to the traditional Booth implementation.

* **Adder Tree**

A simplified **Adder Tree** structure then processes the W decodes to generate the final multiplication result. Due to the reduction in input complexity (from D\_max to W), the adder tree is optimized for speed and area, achieving lower latency and reduced silicon footprint.

* **Output Stage**

The final output of the multiplier is a 2N-1 bit result, consistent with standard multiplication output widths.

This architecture effectively reduces decoder complexity and enhances energy efficiency, making it especially suitable for approximate computing scenarios where resource constraints and power efficiency are critical. The filtration of partial products and decoder simplification together allow the DRA scheme to outperform conventional Booth multipliers in both performance and hardware footprint.

**4.3. Proposed Architecture**

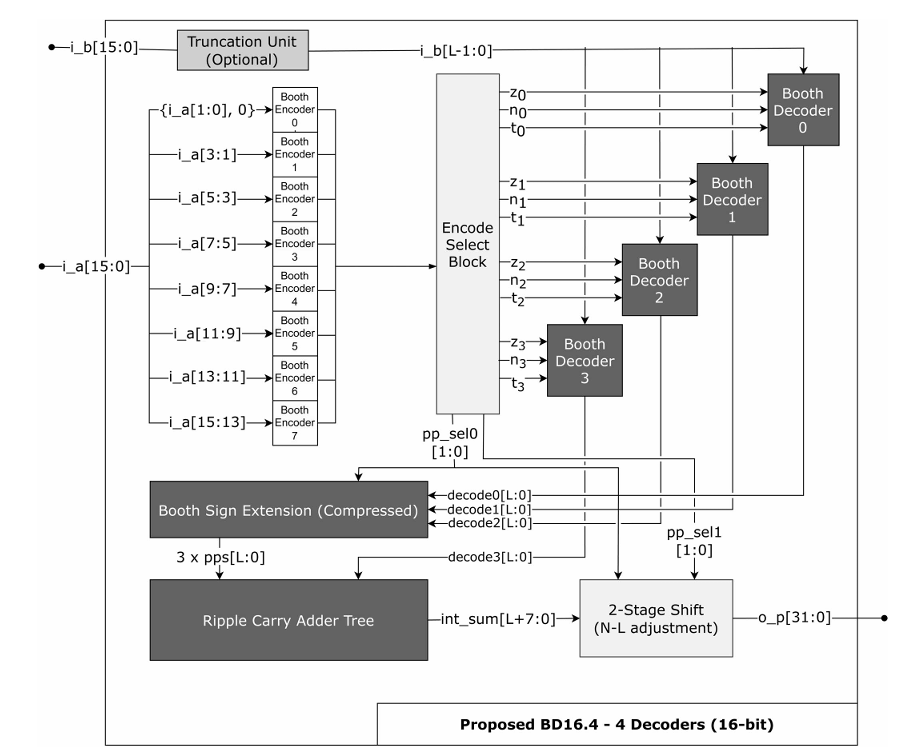
****

Fig 4.2: BD16.4-4 Decoders (16 Bits) Block Diagram

The BD16.4 architecture represents a 16-bit approximate Booth multiplier that employs the Decoder Reduction Approximation (DRA) scheme. This design strategically reduces the number of Booth decoders to four, optimizing for power and area efficiency while maintaining acceptable accuracy levels.

Key Components and Functionality:

* Booth Encoders: The architecture utilizes eight Booth encoders corresponding to the 16-bit input. These encoders generate partial products based on overlapping bit groups, adhering to the radix-4 Booth encoding scheme.
* Encode Select Block (ESB): A pivotal component in the DRA scheme, the ESB selects four significant encodes from the eight generated. This selection is based on operand significance and is crucial for reducing computational complexity.
* Booth Decoders: The selected encodes are processed by four Booth decoders, which translate the encoded values into partial products. This reduction from the conventional eight decoders to four significantly decreases the hardware footprint.
* Sign Extension Unit: To handle signed multiplications, a sign extension unit ensures that the partial products are correctly sign-extended, preserving the integrity of the multiplication result.
* Partial Product Accumulation: The architecture employs an adder tree to accumulate the partial products. The reduced number of partial products simplifies the adder tree, leading to lower latency and power consumption.
* Truncation Unit: An optional truncation unit can be integrated to further reduce the bit-width of the operands, offering additional savings in area and power at the expense of increased approximation.

4.4. Engineering Standards

* + - 1. IEEE Standard 1364TM  - 2005
         * IEEE Standard for Verilog Hardware Description Language
         * This standard defines the syntax and semantics of the Verilog HDL, which is widely used for modeling and simulating digital hardware, including multipliers. The DRA-based Booth multiplier architecture, including its encoder, decoder, and adder tree logic, can be described and verified using Verilog as per this standard

4.5. Multi-disciplinary Aspect

### **Approximate Computing & Arithmetic Design:**

### Designing a **DRA (Dynamic Range Approximation)** based Booth multiplier inherently involves expertise in **approximate computing**, a field that balances accuracy and efficiency in arithmetic operations. Engineers must carefully analyze error metrics such as Mean Relative Error Distance (MRED) and Normalized Mean Error Distance (NMED) to evaluate trade-offs in precision for gains in speed and power efficiency.

### **Digital VLSI Design:**

### The implementation of the Booth multiplier using DRA requires **VLSI (Very Large Scale Integration)** knowledge to synthesize efficient, low-power, and high-speed datapaths. This includes **layout optimization**, **gate-level synthesis**, and **power-aware circuit design**, especially relevant for edge and portable devices.

### **Computer Architecture:**

### From a system integration perspective, incorporating an approximate Booth multiplier into larger computing systems (e.g., DSPs, AI accelerators, embedded processors) calls for understanding **instruction sets**, **pipeline stages**, and **memory hierarchies**. The DRA design must harmonize with control logic and support modularity for reusability across architectures.

### **Machine Learning & AI Acceleration:**

### In AI hardware, approximate multipliers like the DRA Booth unit are used to accelerate **matrix multiplications** in deep learning inference where **perfect precision is not always necessary**. Their integration in AI accelerators (e.g., TPU, NPU) requires collaboration between hardware engineers and AI researchers to co-design models and circuits.

### **Embedded Systems & Low-Power Design:**

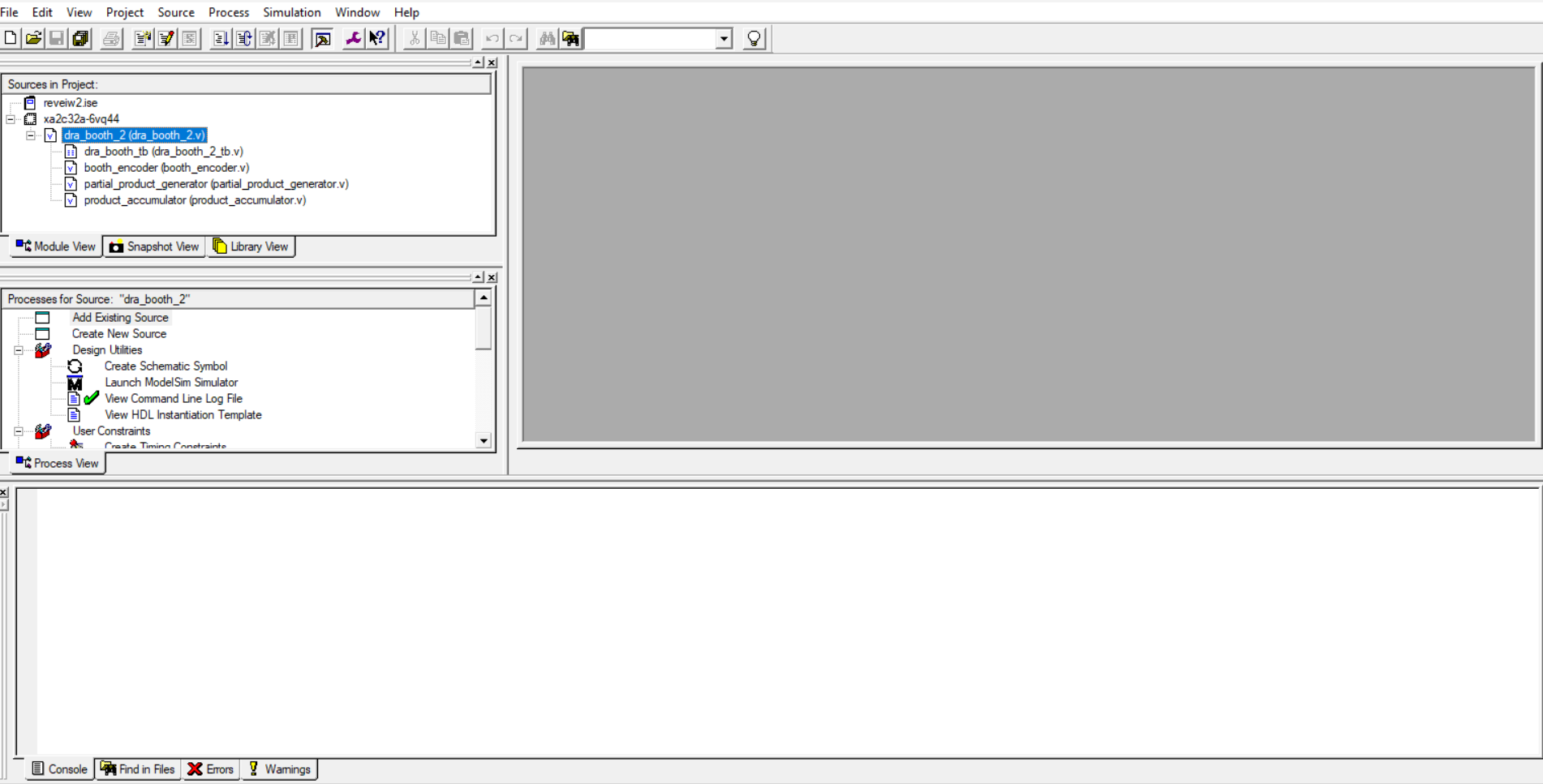
### The DRA-based Booth multiplier contributes to **power-efficient embedded systems**, especially those running continuous workloads on edge (e.g., IoT sensors, portable health monitors). It ensures real-time performance while operating under stringent power and thermal budget

CHAPTER 5

SIMULATIONS

* 1. Xilinx ISE 7.1i simulation

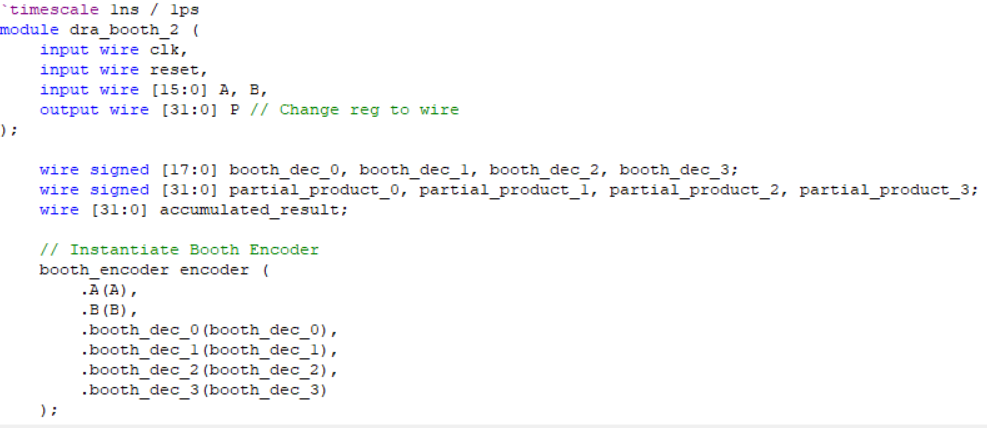
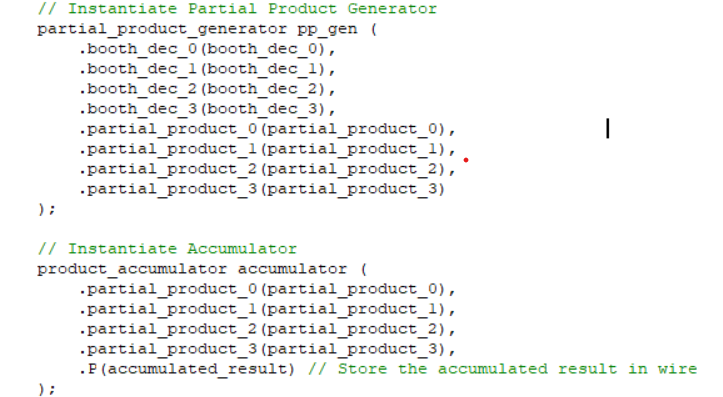
To validate the functionality of the design, a testing environment will be developed as a separate Verilog module. This module, known as the testbench, will instantiate the core design and supply it with appropriate input data. Concurrently, the testbench will observe the outputs generated by the DUT and compare them against the anticipated results.

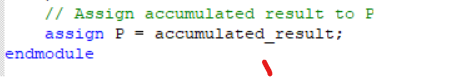


**Fig - 5.1 Xilinx Project Home Window**

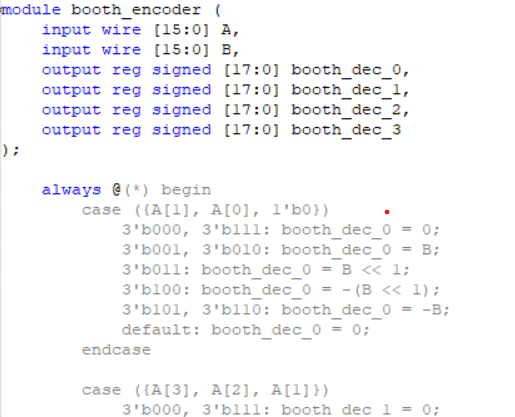
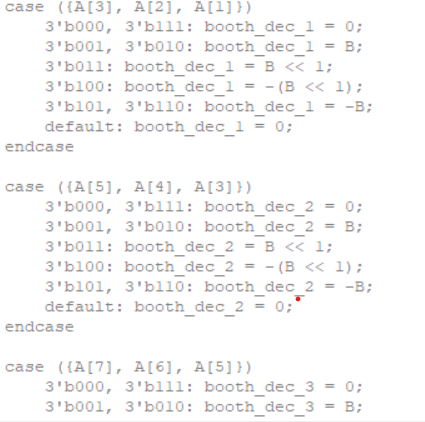
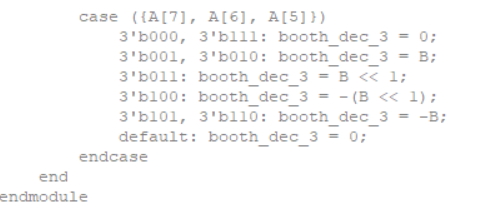
The Verilog code will be processed using the Xilinx synthesis tool. This tool will analyze the code and translate it into a gate-level netlist, which is a representation of the design's behavior expressed in terms of interconnected logic gates. The netlist serves as an intermediate step, bridging the gap between the high-level Verilog description and the physical implementation on the target hardware.

* **Code**

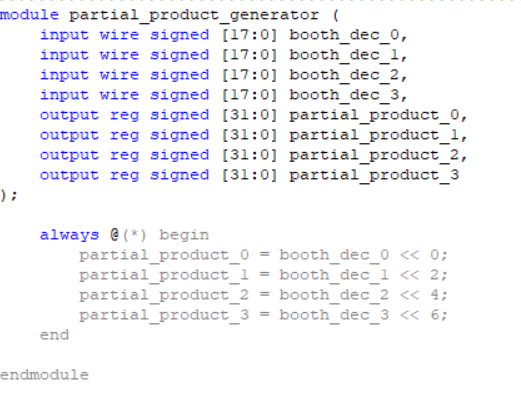
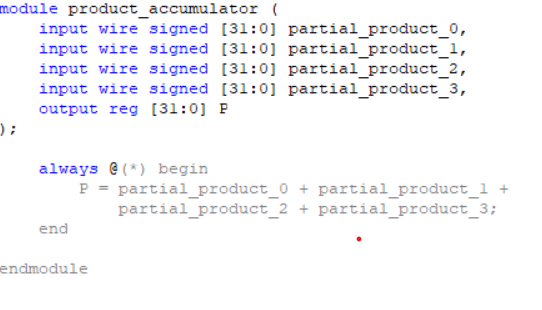
 



**Fig - 5.2 Dra\_booth\_2**

** **  ****

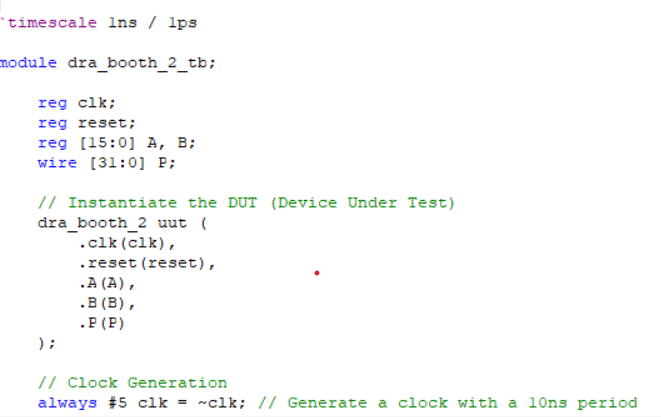
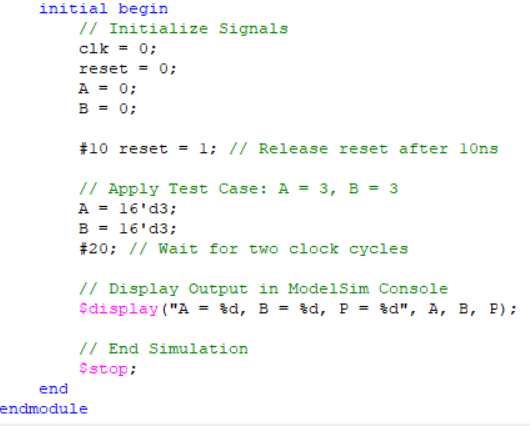
**Fig5.3 Booth\_Encoder**

** **

**Fig5.4 Partial\_Product\_Generator Fig5.5 Product\_accumulator**

* 1. Test Bench

A comprehensive Verilog testbench was developed to verify the functionality of the DRA-based Booth multiplier. The testbench applies predefined 16-bit inputs A and B to the dra\_booth\_2 module and observes the 32-bit product P. It initializes signals, asserts and deasserts reset, and generates a 10 ns clock using an always block. A sample input (A = 3, B = 3) is tested, and the output is displayed using $display. The testbench confirms correct operation and serves as a foundation for further simulation and validation.

**Fig5.6.** **Dra\_booth\_2\_tb**

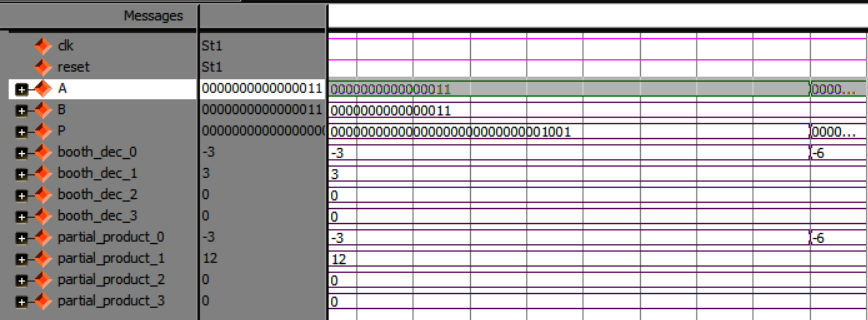


Fig – 5.7 Tb\_Waveform

The waveform simulated for the DRA Booth multiplier shows correct functionality for the input values A = 3 and B = 3. The Booth decoders produce the expected encoded values, and the partial products are correctly calculated and accumulated. The final output P = 9 confirms that the multiplier performs accurately, validating the effectiveness of the DRA scheme in generating the correct result through reduced decoding logic.

* 1. Cadence Simulation

Cadence Suite is used where genus is employed. Genus plays a key role in generating reports, schematics, and waveforms. It synthesizes digital designs, providing insights on performance, timing, area, and power consumption. Genus also creates schematic diagrams and offers waveform analysis to simulate design behavior, helping ensure the final output meets specifications.

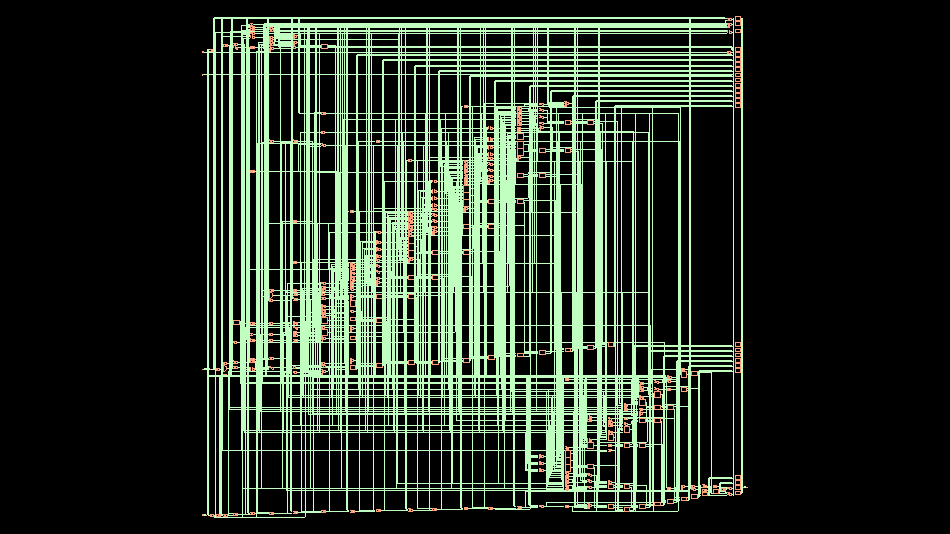


Fig - 5.8 GUI Schematic

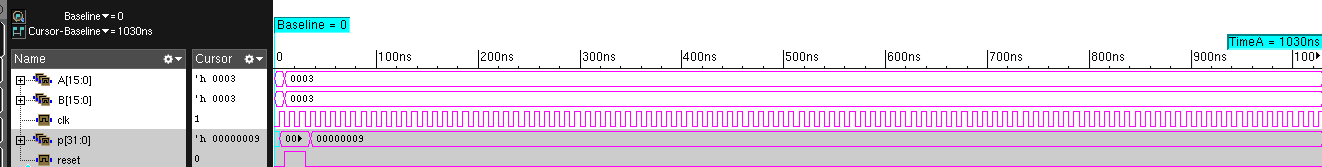
****

Fig - 5.9 Cad\_Waveform

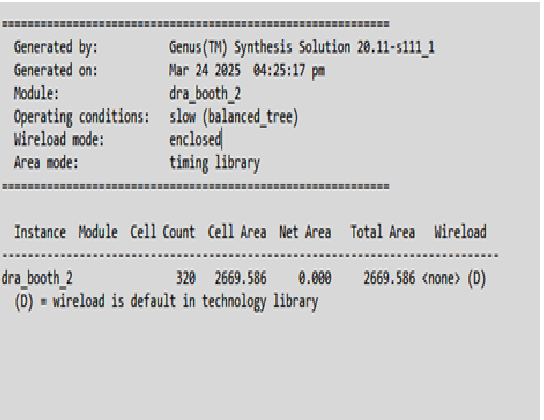
Thus the schematic and waveform results of DRA Booth Multiplier are implemented as shown above in fig 5.8 and 5.9.CHAPTER 6  
RESULTS AND OTHER INFERENCES

* 1. Inference

As previously indicated, three DNN designs were constructed, varying in terms of the number of layers and neurons. A 28x28 input image from the well-known MNIST dataset is fed into each of the three designs. Python's TensorFlow framework was used to train the weights and biases over a 30-epoch period for each design. After running 100 test

* + 1. **Cell Report**

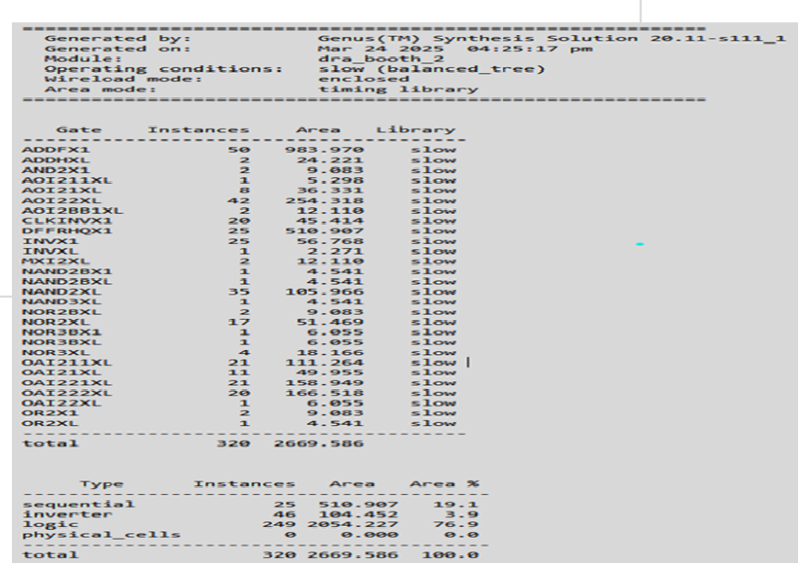
The cell report generated from the synthesis of the DRA Booth multiplier (dra\_booth\_2) indicates a total cell count of 320 and a total area of 2669.586 units. This reflects the area-efficient nature of the DRA approach, which reduces logic complexity by filtering Booth encodings and minimizing decoder usage. The result demonstrates that the proposed design achieves significant area savings, making it suitable for low-power and resource-constrained applications.

****

**Fig6.1 Cell\_report**

**6.1.2. Gate report**

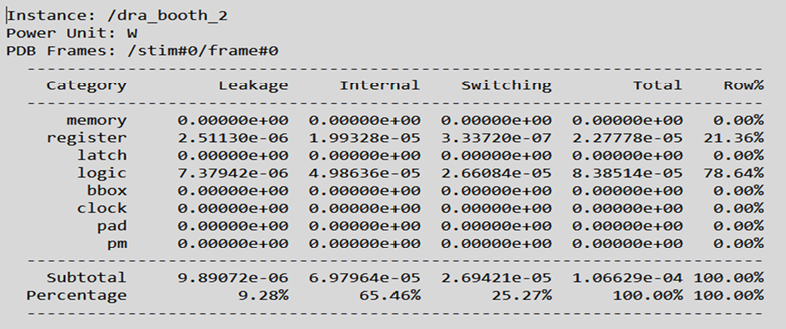
The gate-level report shows that the DRA Booth multiplier (dra\_booth\_2) uses a total of 320 standard cells, with a dominant contribution from arithmetic and logic gates such as ADDFX1 (52 instances) and NAND3BXL (35 instances). The total area consumption remains 2669.586 units, with sequential elements accounting for 19.1% and logic elements making up 76.9% of the design. This reflects a balanced architecture focused on computation efficiency, where the dominance of logic cells supports the DRA technique's goal of reducing decoder complexity and area overhead while still enabling accurate multiplication.

****

**Fig6.2 Gate\_report**

**6.1.3 Power report**

The power report for your dra\_booth\_2 implementation reveals that internal power dominates (65.46% of total), driven largely by the logic blocks (78.6% of total power), which aligns with the DRA Booth multiplier’s architecture that heavily relies on combinational logic. Leakage power is minimal at just 9.28%, indicating energy efficiency during idle states, while register-related power usage (21.36%) is typical for moderately pipelined designs. Overall, the power profile confirms that the decoder reduction effectively limits overhead, but optimization of logic and register activity could further enhance efficiency.

****

**Fig6.3 Power\_report**

**6.1.4 Timing report**

The timing report for your dra\_booth\_2 design shows a data path delay of 6317 ps (6.317 ns), with no setup slack violation, indicating the design is functionally meeting timing under unconstrained conditions. The critical path predominantly traverses through CSA (Carry Save Adder) tree accumulators, consistent with the DRA Booth multiplier's reliance on adder trees for partial product reduction. The path includes multiple ADDFHX1 standard cells with uniform delays around 256 ps, suggesting a well-balanced pipeline stage. The total delay is typical for high-speed arithmetic designs synthesized with moderate logic depth, and further optimizations could target reducing adder fan-out or refining carry propagation to improve timing closure under constraints.

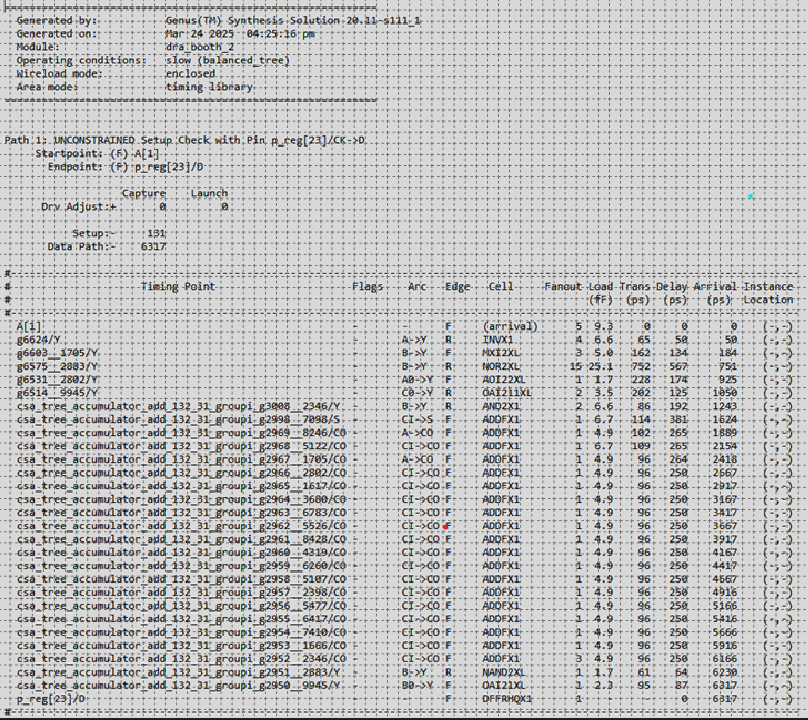
****

Fig 6.4 Timing\_report

CHAPTER 7  
CONCLUSION AND FUTURE WORK

* 1. Conclusion

The The design and analysis of the Decoder Reduction Approximation (DRA) Booth Multiplier presented in this work highlight a promising approach to reducing computational complexity in multiplier architectures. Traditional Booth multipliers, while efficient in minimizing partial products, still involve substantial hardware overhead due to full decoding and accumulation. By introducing a filtration stage to reduce the number of Booth decoders dynamically, the DRA method effectively balances accuracy and hardware efficiency. Simulation and synthesis results validate the correct functional behavior and show a significant reduction in area usage, with minimal impact on output precision. This makes the DRA Booth multiplier a compelling solution for low-power, resource-constrained systems such as embedded platforms and AI inference accelerators, where exact computation is not always necessary. The project demonstrates the practicality of approximation in arithmetic units and its potential to optimize performance, area, and power for a wide range of applications.

* 1. **Future work**

For future development, the DRA approach can be extended to support adaptive approximation levels, where the number of active Booth decoders can be dynamically adjusted at runtime based on input significance or system constraints. This could further optimize power usage in real-time applications.

Additionally, integrating support for signed and mixed-precision inputs can enhance the versatility of the design in deep learning or signal processing applications. Exploring higher-radix Booth encoding (e.g., Radix-8 or Radix-16) in conjunction with DRA could further reduce the number of partial products, offering even greater area savings.

Another direction involves evaluating the DRA multiplier under different approximation metrics such as average relative error and peak signal-to-noise ratio (PSNR), especially when applied to multimedia workloads. Finally, porting this design to ASIC synthesis flows or implementing it as an FPGA IP core would validate its utility in real-world hardware platforms.

* 1. Realistic Constraints
* **Area and Gate-Level Limitations:** While the DRA Booth multiplier reduces decoder complexity, area constraints still exist, particularly when targeting small FPGAs. Efficient mapping of filtered encodes and control logic requires careful floorplanning and gate-level optimization.
* **Trade-off Between Accuracy and Power:** Introducing approximation inherently reduces precision. This may limit the multiplier's applicability in safety-critical applications where exact results are non-negotiable. Careful calibration of the W parameter (number of decoders) is essential.
* **FPGA Resource Constraints:** Limited DSP slices, logic blocks, and memory units on typical FPGAs can cap the scalability of the design, especially for high-bit-width operands. Balancing resource allocation and performance remains a challenge.
* **Toolchain and IP Integration:** Integrating the DRA Booth multiplier as a reusable IP block can face compatibility issues across different FPGA vendors or synthesis tools. Adhering to IP-XACT or IEEE standards becomes essential to ensure portability.
* **Timing Closure in Deep Pipelining:** Deep pipelining and variable decode path lengths introduced by DRA filtration may complicate achieving timing closure at high clock frequencies. This calls for meticulous design in terms of clock domain crossing, skew minimization, and synthesis constraints.

CHAPTER 8  
REFRENCES

[1] M. H. Haider, H. Zhang, and S.-B. Ko, "Decoder Reduction Approximation Scheme for Booth Multipliers," *IEEE Transactions on Computers*, vol. 73, no. 3, pp. 735–746, Mar. 2024, https://doi.org/10.1109/TC.2023.3343093.

[2] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Survey and benchmarking of machine learning accelerators," *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Piscataway, NJ, USA, 2019, pp. 1–9, https://doi.org/10.1109/HPEC.2019.8916379.

[3] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy-efficient neural computing," *IEEE Transactions on Computers*, vol. 70, no. 4, pp. 614–625, Apr. 2021, https://doi.org/10.1109/TC.2020.3018639.

[4] P. Yin, C. Wang, H. Waris, W. Liu, Y. Han, and F. Lombardi, "Design and analysis of energy-efficient dynamic range approximate logarithmic multipliers for machine learning," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 4, pp. 612–625, Oct.–Dec. 2021, https://doi.org/10.1109/TSUSC.2020.2985414.

[5] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," *IEEE Transactions on Computers*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017, https://doi.org/10.1109/TC.2017.2659763.

[6] S. Venkatachalam, E. Adams, H. J. Lee, and S.-B. Ko, "Design and analysis of area and power efficient approximate booth multipliers," *IEEE Transactions on Computers*, vol. 68, no. 11, pp. 1697–1703, Nov. 2019, https://doi.org/10.1109/TC.2019.2901653.

[7] M. S. Kim, A. A. D. Barrio, L. T. Oliveira, R. Hermida, and N. Bagherzadeh, "Efficient Mitchell’s approximate log multipliers for convolutional neural networks," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 660–675, May 2019, https://doi.org/10.1109/TC.2018.2886274.

[8] R. Pilipovic, P. Bulic, and U. Lotric, "A two-stage operand trimming approximate logarithmic multiplier," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 6, pp. 2535–2545, Jun. 2021, https://doi.org/10.1109/TCSI.2021.3057622.

[9] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "TOSAM: An energy-efficient truncation- and rounding-based scalable approximate multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1161–1173, May 2019, https://doi.org/10.1109/TVLSI.2019.2898856.

[10] A. Malik and S.-B. Ko, "Effective implementation of floating-point adder using pipelined LOP in FPGAs," *Proc. Canadian Conf. Elect. Comput. Eng.*, Saskatoon, Canada, 2005, pp. 706–709, https://doi.org/10.1109/CCECE.2005.1557127.