2025

CS 1105

Digital Electronics & Computer Architecture

ASSIGNMENT ACTIVITY UNIT 4

SANA UR REHMAN

INSTRUCTOR: ROBERT MURRAY

BINARY ARITHMETIC CALCULATOR DESIGN: IMPLEMENTING DIGITAL MATHEMATICS THROUGH BINARY OPERATIONS

INTRODUCTION

Digital computation fundamentally relies on binary arithmetic operations that form the backbone of modern processors and calculators. Designing a binary arithmetic calculator presents an opportunity to explore how mathematical operations translate from familiar decimal representations to the binary domain that computers inherently understand. This calculator design incorporates four essential arithmetic modules => addition, subtraction, multiplication, and division, each leveraging the unique properties of binary number systems to perform efficient computations. The development of such a system demonstrates the elegant simplicity underlying complex digital operations while highlighting the practical advantages of binary arithmetic in electronic systems.

BINARY ADDER MODULE DESIGN

The foundation of binary arithmetic begins with the adder module, which employs cascaded full-adder circuits to handle multi-bit binary addition. Each full-adder processes two input bits plus a carry-in signal, producing a sum output and carry-out signal. The design utilizes XOR gates for sum generation and AND/OR gate combinations for carry propagation logic.

For an 8-bit binary adder, eight full-adders connect in series, with the carry-out of each stage feeding the carry-in of the subsequent stage. This ripple-carry configuration ensures accurate addition across all bit positions. The module accepts two 8-bit binary inputs (A and B) and produces a 9-bit result to accommodate potential overflow conditions.

Example Calculation:

- Binary A: 01101011 (107 decimal)
- Binary B: 00110110 (54 decimal)
- Result: 010100001 (161 decimal)

The carry propagation flows from least significant bit (LSB) to most significant bit (MSB), ensuring mathematical accuracy while maintaining hardware simplicity.

Binary Subtractor Module Implementation

Binary subtraction employs two's complement arithmetic, converting subtraction operations into addition problems. The subtractor module incorporates an inverter stage followed by the previously designed adder circuit. The minuend remains unchanged while the subtrahend undergoes bit inversion and receives an additional +1 through the carry-in signal, effectively implementing two's complement conversion.

This approach eliminates the need for separate subtraction hardware, demonstrating the efficiency of binary arithmetic systems. The module handles signed and unsigned operations seamlessly, with overflow detection circuits monitoring the most significant bits for arithmetic validity.

Example Calculation:

- Binary A: 10110100 (180 decimal)
- Binary B: 01001110 (78 decimal)
- Two's complement of B: 10110010
- Result: 01100110 (102 decimal)

Binary Multiplier Module Architecture

The multiplier module implements the shift-and-add algorithm, mirroring traditional long multiplication techniques adapted for binary operations. The design utilizes partial product generation through AND gate arrays, followed by accumulation using multiple adder stages arranged in a tree structure.

Each bit of the multiplier determines whether to include the corresponding shifted multiplicand in the final sum. The module employs parallel processing where possible, generating all partial products simultaneously before summation. For enhanced performance, the design incorporates Booth's algorithm optimization to reduce the number of addition operations required.

Example Calculation:

- Binary A: 00001101 (13 decimal)
- Binary B: 00001011 (11 decimal)
- Partial products: 1101, 11010, 000000, 1101000
- Result: 010001111 (143 decimal)

According to Munawar, Shabbir, and Akram (2023), modern binary multiplier implementations can achieve significant performance improvements through parallel processing architectures, reducing computation time compared to sequential approaches.

Binary Divider Module Development

Binary division presents the most complex arithmetic operation, requiring iterative subtraction and shift operations. The divider module implements the restoring division algorithm, performing conditional subtraction at each step based on the intermediate remainder sign.

The circuit maintains three registers: dividend (initially loaded with numerator), divisor, and quotient. Each iteration shifts the dividend left, subtracts the divisor, and sets the corresponding quotient bit based on the subtraction result. Negative results trigger restoration by re-adding the divisor, while positive results advance to the next iteration.

Example Calculation:

- Dividend: 01100100 (100 decimal)
- Divisor: 00001100 (12 decimal)
- Quotient: 00001000 (8 decimal)
- Remainder: 00000100 (4 decimal)

Module Integration and Organization

The calculator architecture centers on a control unit managing data flow between arithmetic modules and memory registers. Input multiplexers route operands to the appropriate arithmetic unit based on operation selection signals, while output demultiplexers direct results to display or storage registers.

The system employs a common data bus architecture with tri-state buffers enabling module isolation when inactive. Clock signals synchronize operations across modules, ensuring data integrity during multi-cycle operations like multiplication and division. Status flags monitor overflow, zero result, and sign conditions, providing essential feedback for program control.

The integration strategy prioritizes modularity, allowing independent testing and optimization of individual arithmetic units while maintaining system-level coherence through standardized interfaces.

Advantages and Challenges of Binary Arithmetic

Binary arithmetic offers substantial advantages in digital implementation, primarily stemming from the direct correspondence between binary digits and electronic switch states. Hardware complexity reduces significantly compared to decimal arithmetic, as each digit requires only two states rather than ten. This simplification enables faster switching speeds, reduced power consumption, and improved reliability through noise immunity.

Binary operations exhibit mathematical elegance through consistent algorithms across all bit positions, facilitating parallel processing implementations. Error detection and correction mechanisms integrate naturally into binary systems through parity checking and redundant encoding schemes.

However, binary arithmetic presents challenges in human interpretation, requiring conversion between number systems for practical use. Precision limitations in fixed-width binary representations can introduce rounding errors, particularly in division operations. As noted by Padmanabhan et al. (2022), binary arithmetic systems require careful consideration of word length and scaling factors to maintain numerical accuracy in practical applications.

Significance in Understanding Number Systems

Designing a binary arithmetic calculator promotes profound understanding of fundamental mathematical concepts underlying digital computation. The process reveals how abstract mathematical operations translate into concrete hardware implementations, bridging theoretical knowledge with practical engineering applications.

The calculator demonstrates the universality of mathematical principles across different number systems while highlighting the specific advantages binary representation offers for electronic implementation. Furthermore, the design process emphasizes the importance of modular thinking in complex system development, showcasing how individual components combine to create sophisticated computational capabilities.

Conclusion

The binary arithmetic calculator represents a comprehensive exploration of digital computation principles, integrating fundamental arithmetic operations within a cohesive hardware architecture. The modular design approach facilitates understanding of individual operations while demonstrating their collective integration within larger computational systems. Through practical implementation of binary adders, subtractors, multipliers, and dividers, the calculator showcases the efficiency and versatility of binary arithmetic in solving mathematical problems. The project ultimately reinforces the significance of binary number systems in modern digital technology while providing valuable insights into the mathematical foundations underlying computer architecture and digital system design.

References

Munawar, M., Shabbir, Z., & Akram, M. (2023). *Area, delay, and energy-efficient full Dadda multiplier*. arXiv. https://doi.org/10.48550/arXiv.2307.05677

Padmanabhan, K. K., & others. (2022). *High-speed grouping and decomposition multiplier for improved parallel performance*. Electronics, 11(24), 4202. https://doi.org/10.3390/electronics11244202

Wordcount: 1049

Binary Arithmetic Calculator Visualization

BINARY ADDITION

First Number: 107

Second Number: 54

CALCULATE ADDITION

Addition Result

A (Binary)	B (Binary)	Result (Binary)
01101011	00110110	010100001
0 1 1 0 1 0 1 1	0 0 1 1 0 1 1 0	0 1 0 1 0 0 0 0 1

Step-by-step Addition:
107 (decimal) = 01101011 (binary)
54 (decimal) = 00110110 (binary)
Sum = 161 (decimal) = 010100001 (binary)

Carry propagation flows from right to left (LSB to MSB)

BINARY SUBTRACTION (TWO'S COMPLEMENT)

Minuend: 180

Subtrahend: 78

CALCULATE SUBTRACTION

Subtraction Result (Using Two's Complement)

A (Binary)	-B (Two's Complement)	Result (Binary)
10110100	10110010	01100110
1 0 1 1 0 1 0 0	1 0 1 1 0 0 1 0	0 1 1 0 0 1 1 0

Step-by-step Subtraction (Two's Complement):
180 - 78 = 180 + (-78)
180 (decimal) = 10110100 (binary)
78 (decimal) = 01001110 (binary)
Invert bits of 78: 10110001
Add 1: 10110010 (two's complement of 78)
Result = 102 (decimal) = 01100110 (binary)

BINARY MULTIPLICATION

Multiplicand	Multiplier	
13	11	CALCULATE MULTIPLICATION

Multiplication Result (Shift-and-Add Method)

A (Binary)	B (Binary)	Result (Binary)
00001101	00001011	0000000010001111
0 0 0 0 1 1 0 1	0 0 0 0 1 0 1 1	0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1

Step-by-step Multiplication (Shift-and-Add):
13 (decimal) = 00001101 (binary)
11 (decimal) = 00001011 (binary)

Partial Products:
00001101 << 0 = 00001101
00001101 << 1 = 000011010
00001101 << 3 = 00001101000

Final Result = 143 (decimal) = 0000000010001111 (binary)

BINARY DIVISION

Dividend	Divisor	
100	12	CALCULATE DIVISION

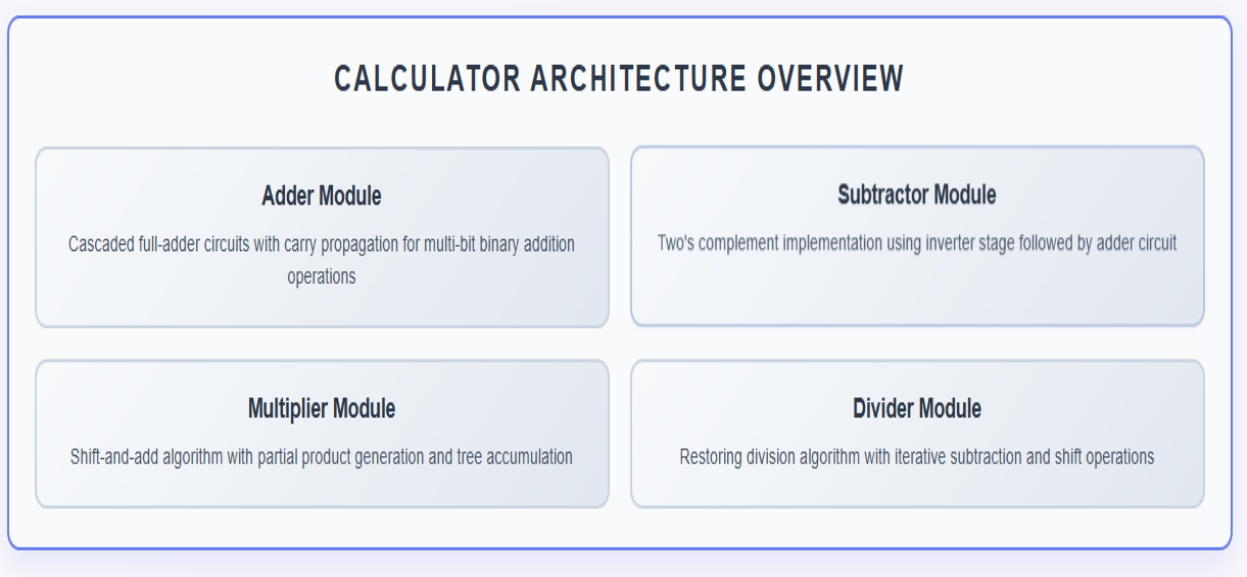
Division Result (Restoring Division Algorithm)

Dividend (Binary)	Quotient (Binary)	Remainder (Binary)
01100100	00001000	00000100
0 1 1 0 0 1 0 0	0 0 0 0 1 0 0 0	0 0 0 0 0 1 0 0

Step-by-step Division (Restoring Algorithm):
100 ÷ 12
Dividend: 100 (decimal) = 01100100 (binary)
Divisor: 12 (decimal) = 00001100 (binary)

Using iterative subtraction and shift operations:
Quotient: 8 (decimal) = 00001000 (binary)
Remainder: 4 (decimal) = 00000100 (binary)

Verification: 8 × 12 + 4 = 100 = 100 ✓


CALCULATOR ARCHITECTURE OVERVIEW
Adder Module
Cascaded full-adder circuits with carry propagation for multi-bit binary addition operations
Subtractor Module
Two's complement implementation using inverter stage followed by adder circuit
Multiplier Module
Shift-and-add algorithm with partial product generation and tree accumulation
Divider Module
Restoring division algorithm with iterative subtraction and shift operations