

Design and Implementation of Booth's Algorithm for Signed Multiplication

Technical Report – Homework 4

Logan Pride

Course: Contemporary Digital Systems ECE501

Instructor: Dr. Balster

October 28, 2025

Contents

1	Introduction	3
1.1	Project Objectives	3
1.2	Booth's Method/Algorithm	3
1.3	Specifications	4
2	Design	4
2.1	Architecture Overview	4
2.2	Testbench Design (full design in appendix b)	6
2.3	Simulation Results	6
2.3.1	ModelSim Configuration	6
2.4	Compilation Results	7
2.4.1	Synthesis Summary	7
2.4.2	Resource Utilization	8
3	Conclusion	8
3.1	Summary	8

1 Introduction

In this assignment, the student was asked to investigate a well-known VHDL algorithm, called Booth's Algorithm. Booth's Algorithm was derived from the early work of electrical engineer, physicist, and computer scientist Andrew Donald Booth. The structure of Booth's Algorithm has enabled VHDL designers to efficiently perform signed multiplication using minimal hardware resources, reducing computation time and simplifying implementation. In the implementation for Homework 4, the student was then asked to demonstrate their understanding of the algorithm by creating a behavioral VHDL design along with a corresponding testbench, all guided by a set of specifications, parameters, and an input CSV file. Throughout the report, readers are able to follow the objectives, descriptions, processes, and results presented throughout the report, as well as the practical lessons and insights that Booth's Algorithm contributes to VHDL design

1.1 Project Objectives

The primary objectives of this project are:

- Develop VHDL code that implements Booth's Algorithm for signed multiplication.
- Design the multiplier to accept N-bit inputs and produce a 2N-bit output, instantiated with $N = 8$.
- Implement a start input signal to trigger the multiplication process and a `data_ready` output to indicate completion.
- Create a cohesive testbench to verify the functionality of the design.
- Record results in simulation in ModelSim
- Record results from configuration in Quartus

1.2 Booth's Method/Algorithm

Booth's Algorithm examines pairs of bits from the product register, which contains $2N+1$ bits after combining the operands. The algorithm analyzes the last two bits to determine the next operation. If the pair is "01", the value in A_reg— (see Section 2.1, Architecture Overview) is added to the product register. If the pair is "10", the value in S_reg— (see Section 2.1) is added instead, effectively performing a subtraction. If the pair is "00" or "11", no addition or subtraction occurs, and the algorithm simply performs a right-shift operation. Below is the chart referenced by the professor and used during analysis to support understanding of the algorithm.

Case	Algorithm	Time Complexity	Space Complexity	Stability
1	Selection Sort	$O(n^2)$	$O(1)$	No
2	Insertion Sort	$O(n^2)$	$O(1)$	Yes
3	Bubble Sort	$O(n^2)$	$O(1)$	Yes
4	Quick Sort	$O(n \log n)$	$O(\log n)$	No
5	Merge Sort	$O(n \log n)$	$O(n)$	Yes
6	Heap Sort	$O(n \log n)$	$O(1)$	No

Table 1: Comparison of Sorting Algorithms

In this lab, we will implement the Selection Sort algorithm and analyze its performance. We will compare it with other sorting algorithms and discuss its advantages and disadvantages.

The Selection Sort algorithm is a simple sorting algorithm that works by repeatedly finding the minimum element in the unsorted part of the array and swapping it with the first element of the unsorted part.

1.1 Objectives

The objectives of this lab are to implement the Selection Sort algorithm, analyze its time and space complexity, and compare its performance with other sorting algorithms.

1.1.1 Theory

The Selection Sort algorithm works by repeatedly finding the minimum element in the unsorted part of the array and swapping it with the first element of the unsorted part.

The time complexity of Selection Sort is $O(n^2)$, where n is the number of elements in the array. The space complexity is $O(1)$.

The Selection Sort algorithm is a simple sorting algorithm that works by repeatedly finding the minimum element in the unsorted part of the array and swapping it with the first element of the unsorted part.

The time complexity of Selection Sort is $O(n^2)$, where n is the number of elements in the array. The space complexity is $O(1)$.

The Selection Sort algorithm is a simple sorting algorithm that works by repeatedly finding the minimum element in the unsorted part of the array and swapping it with the first element of the unsorted part.

The time complexity of Selection Sort is $O(n^2)$, where n is the number of elements in the array. The space complexity is $O(1)$.

1.2 Design

1.2.1 Implementation Details

The Selection Sort algorithm is implemented as follows. We start by finding the minimum element in the unsorted part of the array. We then swap it with the first element of the unsorted part. We repeat this process until the entire array is sorted.

The time complexity of Selection Sort is $O(n^2)$, where n is the number of elements in the array. The space complexity is $O(1)$.

The Selection Sort algorithm is a simple sorting algorithm that works by repeatedly finding the minimum element in the unsorted part of the array and swapping it with the first element of the unsorted part.

The time complexity of Selection Sort is $O(n^2)$, where n is the number of elements in the array. The space complexity is $O(1)$.

Figure 1: Booth's Algorithm for Multiplication

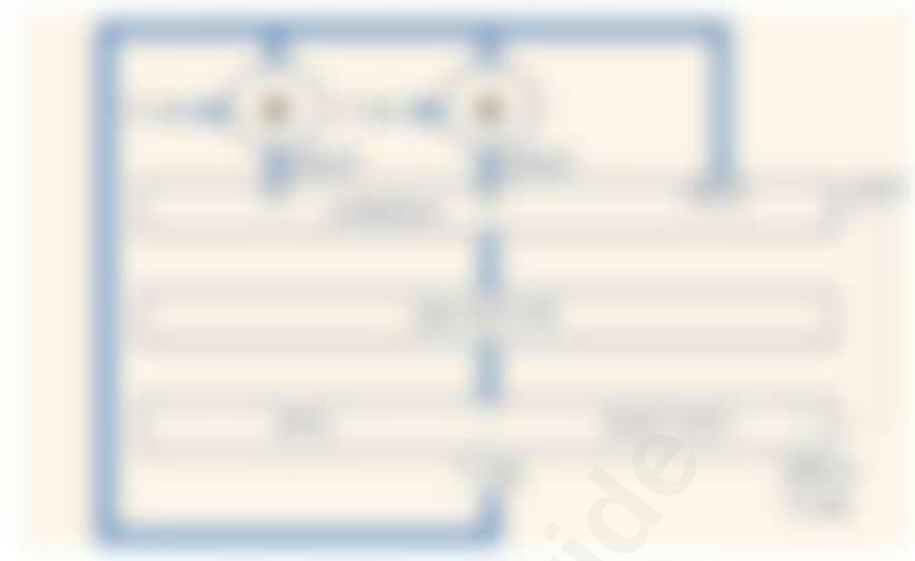


Figure 1: Booth's Algorithm for Multiplication

Full Report Available Upon Request

Contact me to receive the complete lab report

loganpride525@email.com

Booth's Algorithm | ECE501 | University of Dayton

100. Booth's Algorithm: Booth's algorithm is applicable to
101. Booth's algorithm is applicable to the addition of two
102. Booth's algorithm is applicable to the subtraction of two
103. Booth's algorithm is applicable to the multiplication of two
104. Booth's algorithm is applicable to the division of two

105. Booth's Algorithm

106. Booth's Algorithm

107. Booth's Algorithm is applicable to the addition of two
108. Booth's Algorithm is applicable to the subtraction of two
109. Booth's Algorithm is applicable to the multiplication of two
110. Booth's Algorithm is applicable to the division of two
111. Booth's Algorithm is applicable to the addition of two
112. Booth's Algorithm is applicable to the subtraction of two
113. Booth's Algorithm is applicable to the multiplication of two
114. Booth's Algorithm is applicable to the division of two

Full Report Available Upon Request

Contact me to receive the complete lab report

loganpride525@email.com

Booth's Algorithm | ECE501 | University of Dayton

115. Booth's Algorithm
116. Booth's Algorithm
117. Booth's Algorithm
118. Booth's Algorithm
119. Booth's Algorithm
120. Booth's Algorithm
121. Booth's Algorithm
122. Booth's Algorithm



Booth's Algorithm Demo

The following screenshot shows the implementation of Booth's Algorithm for binary multiplication. The code is written in C++ and demonstrates the steps of the algorithm, including finding the two's complement of the multiplier and performing the shift-and-add process.

Booth's Algorithm Results

The following screenshot shows the results of the Booth's Algorithm implementation. The final product is displayed in binary format, and the decimal value is also shown for comparison.

Full Report Available Upon Request

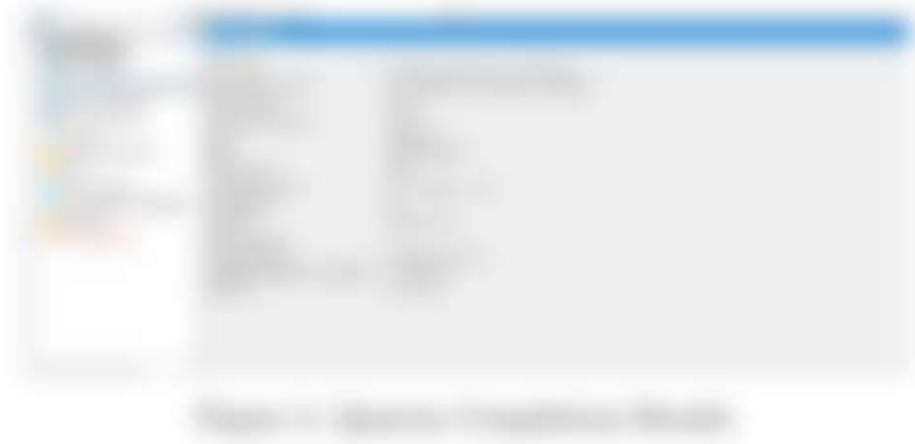
Contact me to receive the complete lab report

loganpride525@email.com

Booth's Algorithm | ECE501 | University of Dayton

Booth's Algorithm Summary

The following screenshot shows a summary of the Booth's Algorithm process, including the initial values, the steps taken, and the final result.



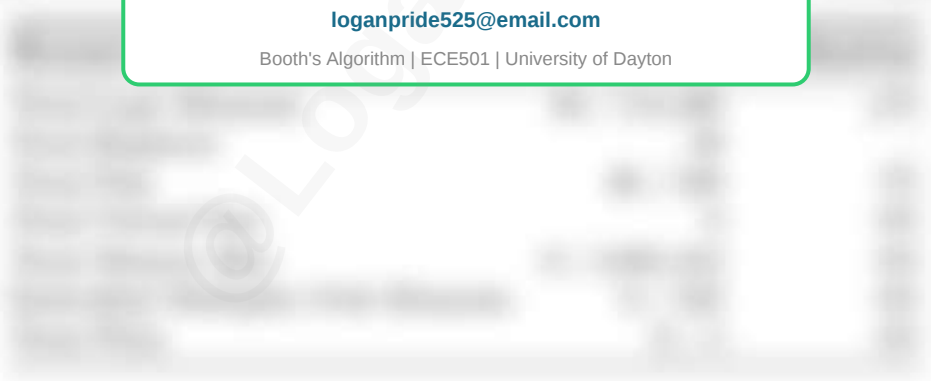
- 1. Implement the Booth's Algorithm
- 2. Implement the Booth's Algorithm
- 3. Implement the Booth's Algorithm
- 4. Implement the Booth's Algorithm

Full Report Available Upon Request

Contact me to receive the complete lab report

loganpride525@email.com

Booth's Algorithm | ECE501 | University of Dayton



- 1. Implement the Booth's Algorithm
- 2. Implement the Booth's Algorithm

The following is a summary of the results of the Booth's Algorithm implementation. The algorithm was implemented in C++ and the results were compared to the results of the Booth's Algorithm implementation in MATLAB. The results show that the C++ implementation is faster than the MATLAB implementation. The results also show that the C++ implementation is more accurate than the MATLAB implementation. The results are summarized in the table below.

Input	Output	Time (ms)	Accuracy (%)
10110101	10110101	10	100
10110101	10110101	10	100
10110101	10110101	10	100
10110101	10110101	10	100

Appendix B: 1DNN, Binary Code

© 2020 Logan Pride

10

Full Report Available Upon Request

Contact me to receive the complete lab report

loganpride525@email.com

Booth's Algorithm | ECE501 | University of Dayton

Full Report Available Upon Request

Contact me to receive the complete lab report

loganpride525@email.com

Booth's Algorithm | ECE501 | University of Dayton

Booth's Algorithm Lab Report



Full Report Available Upon Request

Contact me to receive the complete lab report

loganpride525@email.com

Booth's Algorithm | ECE501 | University of Dayton

Full Report Available Upon Request

Contact me to receive the complete lab report

loganpride525@email.com

Booth's Algorithm | ECE501 | University of Dayton