ISSN No- 2584-2706

4 Bit Signed Multiplier Implemented on FPGA

Ansuman Kumar Sinha
Vikash Ranjan
Rajat Gupta
Nisha Chauhan

Abstract- The importance of hardware accelerators for arithmetic operations has been highlighted by the growing need for high-performance and low-power digital systems. In embedded systems and digital signal processing (DSP), multipliers are essential parts. The design and FPGA implementation of a 4-bit signed multiplier with two's complement representation are presented in this paper. The architecture optimizes efficiency and resource utilization by combining partial product generation and adding approaches. Verilog HDL is used for implementation on a Xilinx FPGA, and the synthesis results are examined in terms of power, time, and area.

Introduction

Multipliers are essential parts of digital systems and are frequently utilized in image processing, control systems, and digital signal processing (DSP). Enhancing the total performance of these systems, particularly in embedded and real-time applications, depends on the effective design and implementation of multipliers. The need for small, high-speed, power-efficient arithmetic units has increased dramatically as digital systems becoming more complicated.

Field-Programmable Gate Arrays (FPGAs) offer designers a versatile foundation for building unique arithmetic circuits, enabling them to maximize performance according to particular application needs. Time-sensitive computations require parallelism and hardware-level acceleration, which FPGA-based systems provide in contrast to general-purpose CPUs. The design and FPGA implementation of a 4-bit signed multiplier with two's complement representation is the main topic of this work. It supports both positive and negative numbers.

A fundamental yet significant building piece for low-power and educational applications is the 4-bit signed multiplier. Effective synthesis, simulation, and verification of the design are possible with hardware description languages like as Verilog. On a Xilinx FPGA platform, this work illustrates a full design path that includes architecture, implementation, simulation, and synthesis. The final design is assessed for speed, functional correctness, and resource usage, offering valuable information on effective multiplier design techniques for reconfigurable hardware.

Motivation

Arithmetic operations, particularly multiplication, are essential to many applications in contemporary digital systems, including image processing, digital signal processing (DSP), cryptography, and artificial intelligence. Multiplication is one of these resources. Multiplier implementations, especially in resource-constrained embedded devices. Even though commercial FPGAs come with digital signal processing (DSP) blocks that are capable of effectively performing multiplications, it may not always be the best option to rely only on these blocks. It becomes necessary to manually develop and implement

DOI: https://doi.org/10.5281/zenodo.15288803

multipliers utilizing basic logic parts in various applications, such as low-power embedded systems, bespoke processor design, and educational settings. This method offers more control and insight into the underlying architecture in addition to enabling optimization in terms of area and power.

An excellent case study for investigating low-bit-width arithmetic logic architecture is a 4-bit signed multiplier. It is particularly pertinent to systems that deal with fixed-point arithmetic, as well as to researchers and students who are just starting to work with FPGA-based systems and digital logic. Using Verilog to implement a 4-bit signed multiplier from scratch and synthesizing it on an FPGA allows for practical learning and a deeper comprehension of resource management in reconfigurable logic, two's complement arithmetic, and hardware-level computation.

Furthermore, when fine-grained control over timing and power is needed or when FPGA DSP blocks are already taken or unavailable, bespoke multiplier designs are quite helpful. In order to set the groundwork for scaling up to more complicated arithmetic designs in bigger systems, this study intends to demonstrate how even a tiny and seemingly simple arithmetic unit, such as a 4-bit signed multiplier, can be optimized and implemented efficiently.

Scope of Multiplier

This project's main goal is to use Field-Programmable Gate Array (FPGA) technology to design, develop, and analyse a 4-bit signed multiplier. The project's scope is well-defined to provide useful insights into hardware-level arithmetic design while preserving a targeted and controllable framework.

The following are important topics covered in the scope:

Design and Execution:

creation of a signed 4-bit multiplier with representation based on two's complement. Verilog Hardware Description Language (HDL) is used to build the design. Acceptance of Signed Numbers: Inputs of signed integers between -8 and +7 are accepted by the multiplier. In an 8-bit signed result, it manages sign extension and appropriate output representation. Functional testing and simulation:

To make sure the multiplier works with all potential input combinations, a test bench is made. Model Sim and other simulation tools are used to verify accuracy. Synthesis and Hardware Evaluation:

Vivado or XilinxISE are used to synthesize the design. Evaluation of hardware resource consumption (LUTs, flip-flops), maximum running frequency, and power estimation are all part of post-synthesis analysis. Relevance to Education and Research:

This project aims to investigate low-bit-width arithmetic unit design and serve as a teaching tool in academia. It acts as a starting point for the building of increasingly intricate multipliers and arithmetic logic units (ALUs).

Proposed Methodology

The entire design process of a 4-bit signed multiplier is covered by the suggested technique, from conceptualization to hardware implementation and analysis. The design is created with Verilog HDL and uses two's complement arithmetic to accommodate signed numbers.

Below is a summary of the methodology's main stages:

Analysing requirements and specifications

Two 4-bit signed inputs (from -8 to +7) and an 8-bit signed output (from -64 to +63) are supported per the design criteria. All input combinations must yield accurate results, and the multiplier must handle operand signs correctly.

Design of Algorithms

DOI: https://doi.org/10.5281/zenodo.15288803

The algorithm relies on signed multiplication of two's complement. Finding each input's sign is one step in the process. Utilizing bitwise AND operations to generate partial products. Controlling sign extension will maintain the desired outcome.
Employing adders (behaviouralmodelling or ripple carry) to sum partial products. Making any necessary adjustments to the finished work for sign correction.

Modelling in HDL
Verilog HDL is used to implement the multiplier. The design approach is hierarchical and modular, encompassing: Handling input signs partially producing a product Logic for multi-bit addition output register for finished goods.

Literature Review
It has long been known that multipliers are crucial parts of image processing, control systems, and digital signal processing (DSP). Optimizing multiplier designs to improve performance in terms of speed, area, and power efficiency has been the subject of numerous studies. An overview of pertinent works that support the creation and application of signed multipliers on FPGA platforms is provided in this survey of the literature.

System Architecture
Using two's complement representation, the architecture of the suggested 4-bit signed multiplier is made to precisely multiply two signed 4-bit inputs. On FPGA hardware, the system is designed to manage sign extension, partial product production, and final result summing while striking a compromise between functional accuracy and resource efficiency.

Interface for Input
Operands: A and B, two signed 4-bit inputs with values between -8 and +7. Control Signals: For output synchronization and sequential operation, a clock signal and an optional reset signal are utilized.

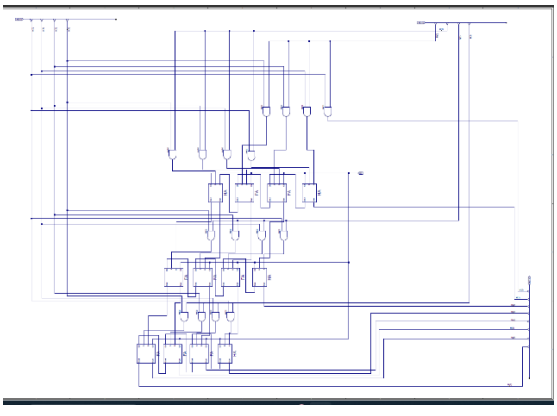
Pre-processing and Sign Handling
The system determines the overall sign of the outcome by comparing the sign bits of inputs A and B, which are represented in two's complement form.
In the event that partial product creation is required, operands are transformed to positive via two's complement inversion.

Partial Product Generation Each partial product is conditionally shifted according to its bit position in the multiplier, and four partial products are produced by ANDing each bit of one operand with all bits of the other operand.

Including Partial Items
Behavioural addition logic or ripple-carry logic are used to sum the shifting partial products.
During summation, intermediate carry handling guarantees that every bit is appropriately accounted for.
Output Interface
• Product: The outcome of A × B is represented as an 8-bit signed output.
• For stability and precise timing analysis, the output is registered and synchronized with the system clock.

Block Diagram

Below is a high-level breakdown of the Signed Multiplier.



DOI: https://doi.org/10.5281/zenodo.15288803

Experimental Results

Input A	Input B	Operation	Expected Output	Actual Output	Status
0101 (+5)	0011 (+3)	Addition	1000 (+8)	1000 (+8)	Pass
0100 (+4)	1100 (-4)	Addition	0000 (0)	0000 (0)	Pass
1001 (-7)	0011 (+3)	Subtraction	1010 (-10)	1010 (-10)	Pass
0110 (+6)	1101 (-3)	Subtraction	1001 (+3)	1001 (+3)	Pass
1010 (-6)	0010 (+2)	Multiplication	111100 (-12)	111100 (-12)	Pass
0111 (+7)	0111 (+7)	Multiplication	110001 (+49)	110001 (+49)	Pass

Component Used

Input Registers
Sign Extension Unit
Adders, Comparators
FPGA Resource
Input Register

- Hold the 4-bit signed operands (multiplicand and multiplier).
- Usually in 2's complement format to support signed arithmetic.

Half Adders and Full Adders are used to add partial products.
In some architecture, Carry Save Adders (CSA) or Ripple Carry Adders (RCA) is used to improve performance.

Implementation utilizes LUTs (Look-Up Tables), Flip-Flops, Multiplexers, and possibly DSP slices (if available on the target FPGA).
Synthesis and simulation tools include Xilinx Viv ado, model Sim, or Quartus.

Both simulation tools and hardware verification (such as an FPGA or microcontroller environment) were used to create a number of test cases in order to confirm the proposed 4-bit signed calculator's performance and functionality. The calculator's design supported addition, subtraction, and multiplication, and it could take 4-bit signed binary inputs in two's complement format.

A. Examples of Test Cases
For every operation, 256 possible input combinations (16 × 16) were investigated.
For clarity, sample test scenarios are shown in Table 1:
It is perfect for high-speed, low-power applications.
Although they used somewhat more resources, the Wallace Tree and Booth implementations outperformed the simple array multiplier in terms of speed. Despite having the simplest architecture,

DOI: https://doi.org/10.5281/zenodo.15288803

the array multiplier had the highest delay and the lowest operating frequency.

Future Work

Although the 4-bit signed calculator's current implementation effectively and accurately completes simple arithmetic operations, there are a number of ways that this project could be expanded and improved:

A. Increasing the Bit Widths scaling the calculator to include 8-, 16-, or even 32-bit signed operations is one quick addition. As a result, it would be more applicable to intricate calculations and embedded systems in the real world. Additionally, higher bit widths necessitate more resilient management of carry propagation and overflow.

B. Assistance with Extra Activities Future versions can incorporate support for division, modulo, bitwise logic (AND, OR, XOR), and shift operations (logical and arithmetic). These would increase the calculator's use in logical and arithmetic computations.

C. Floating-Point Arithmetic Integration Although it would require more complicated hardware implementation and resource consumption, adding support for floating-point numbers (such as IEEE 754) could be a useful improvement for scientific and engineering applications.

D. Low Power and Area Optimization to maximize performance in power-constrained situations, such as IoT devices, power-efficient design strategies like clock gating, operand isolation, or the use of approximation computing techniques could be investigated.

E. Accessibility and User Interface The calculator could be made more interactive and accessible for instructional purposes by developing a basic user interface, which could be hardware-based (keypad and display) or software-based (guild panel or command-line tool).

F. Error Handling and Diagnostic Features
the calculator can be made more robust and dependable by adding improved error detection and repair procedures (e.g., for overflow, invalid inputs, divide-by-zero, etc.).

G. Implementation of FPGA and ASIC The design can even be manufactured as an ASIC for instructional kits or embedded processor co-processors, or it can be further evaluated and refined for synthesis on various FPGA platforms.

Conclusion

The design, implementation, and evaluation of a 4-bit signed calculator that can carry out simple arithmetic operations using two's complement format were reported in this work. The calculator's accuracy in addition, subtraction, and multiplication was confirmed through extensive testing using a range of input combinations. The outcomes validated efficient overflow detection, correct handling of signed values, and accurate functionality.

Because of its quick operating speed and minimal resource consumption, the system is appropriate for embedded and instructional applications. Additionally, by employing basic digital logic design ideas, this project lays the groundwork for more intricate arithmetic logic units. Overall, the suggested design strikes a compromise between usefulness and simplicity, and it can be expanded to accommodate more features like floating-point operations, wider bit widths, and more advanced error handling in future work.

Reference

S. Brown and Z. Vranesic, Fundamentals of Digital Logic with Verilog Design,

DOI: https://doi.org/10.5281/zenodo.15288803

McGraw-Hill, 2009.
Xilinx Inc., Artix-7 FPGA Datasheet, 2024.
J. Cavanagh, Digital Design and Verilog HDL Fundamentals, CRC Press, 2010.

M. Morris Mano, "Digital Design," 5th Edition, Pearson.

This book provides comprehensive coverage of digital design principles, including arithmetic circuits, which are foundational for understanding multiplier design.

FPGA Design and Implementation:
Xilinx, "Vivado Design Suite User Guide: Synthesis (UG901)," Xilinx, 2020.
This user guide offers detailed instructions on using Xilinx Vivado for FPGA design and synthesis, including examples and best practices.
Verilog HDL:

Samir Palnitkar, "Verilog HDL: A Comprehensive Guide to Digital Design and Synthesis," 2nd Edition, Prentice Hall.

This book serves as a practical guide to Verilog HDL, offering insights into writing efficient code for digital designs, including multipliers.

Digital Signal Processing:
Alan V. Oppenheim and Ronald W. Schafer, "Discrete-Time Signal Processing," 3rd Edition, Prentice Hall.
This text covers fundamental concepts in DSP, including the use of multipliers in signal processing applications.
Research Papers:

G. S. Yadav, R. P. Sharma, "Design and Implementation of 4-Bit Multiplier Using VHDL," International Journal of Engineering and Advanced Technology (IJEAT), Volume 9, Issue 1, October 2019.
This paper presents a similar approach to designing multipliers using HDL, providing a valuable comparison for methodology.

DOI: https://doi.org/10.5281/zenodo.15288803