8x8 Signed Serial-Parallel Multiplier (SPM)

Mennatallah Zaher Farida Elanany Mahinour Abdelgawad

May 21, 2025

1 Introduction

This report presents the design and implementation of a sequential 8-bit signed Serial-Parallel Multiplier (SPM) on the Basys 3 FPGA board using the Artix 7 FPGA. The SPM multiplies two signed 8-bit binary numbers, with one operand processed serially (bit-by-bit) and the other loaded in parallel. This approach balances hardware efficiency and operational speed, making it ideal for resource-constrained FPGA applications.

The system features robust control logic for managing serial input, parallel loading, calculation sequencing, and output display. The user interface leverages toggle switches, push buttons, LEDs, and 7-segment displays for input, control, and result visualization.

2 System Overview

2.1 User Interface

- **Switches (SW15–SW0):**
  - SW7–SW0: Multiplier (signed 8 bits)
  - SW15–SW8: Multiplicand (signed 8 bits)
- **Push Buttons:**
  - BTNC: Start multiplication
  - BTNL/BTNR: Scroll product digits on display
- **LEDs:**
  - LD0: Indicates end of multiplication
- **7-Segment Displays:**

- Leftmost: Product sign
- Right three: Product digits (scrollable for 5-digit results)

3 Module Descriptions

Module	Description
Top Module	Main module that integrates all components and interfaces with the FPGA pins. Handles the overall system operation and user interface.
SPM Controller (FSM)	Manages the multiplication process through states: IDLE, LOAD, MULTIPLY, and DONE. Controls the timing of operations and signals completion.
Serial-Parallel Multiplier Core	Performs the actual multiplication by processing the multiplier serially and the multiplicand in parallel. Handles sign correction for signed operands.
HalfAdder	Basic building block that performs addition of two bits, producing sum and carry outputs. Used in constructing more complex adders.
FullAdder	Adds three bits (including carry-in), producing sum and carry-out. Used extensively in the arithmetic operations of the multiplier.
Adder4Bit	Four-bit ripple carry adder constructed from full adders. Used for multi-bit addition operations in the multiplier.
CarrySaveAdder	Specialized adder that reduces three binary numbers to two, optimizing the partial product accumulation process.
Binary-to-BCD Converter	Converts the 16-bit binary product to Binary-Coded Decimal format for display on the 7-segment displays. Implements the Double Dabble algorithm.

Continued on next page

Module	Description
Seven-Segment Display Driver	Controls the 7-segment displays, including multiplexing, digit selection, and sign display. Supports scrolling for viewing all digits.
DFF	D Flip-Flop module used for storing state and synchronizing signals throughout the design.
Debouncer	Eliminates switch bounce from button inputs to ensure clean, single transitions.
PushButton_Detector	Detects button presses and generates clean control signals for the system.
Rising_Edge	Detects rising edges in signals, used for triggering state transitions on button presses.
Clock_Divider	Generates slower clock signals from the main system clock for display multiplexing and timing operations.
Synchronizer	Prevents metastability by synchronizing asynchronous inputs to the system clock domain.

4 Implementation Challenges and Solutions

During the development process, we encountered several challenges that required careful debugging and redesign. The table below summarizes the key issues and their resolutions:

Module	Issue	Solution
Binary-to-BCD Converter	Incorrect conversion for negative numbers	Implemented absolute value calculation before conversion and handled sign separately
SPM Core	Sign extension errors causing incorrect results for negative numbers	Revised the sign handling logic to properly extend sign bits

Continued on next page

Module	Issue	Solution
Debouncer	Button presses occasionally missed or registered multiple times	Adjusted debounce timing parameters and improved the state machine logic
Display Controller	Scrolling functionality not working correctly	Redesigned the digit selection logic and fixed buffer management
Clock Divider	Display flickering due to improper timing	Adjusted clock division ratio and improved synchronization
FSM Controller	State transitions occasionally skipped	Added additional synchronization and improved state encoding

5 Development Process and Simulation Results

Our development process followed an iterative approach, with each module being tested individually before integration. The table below outlines the key simulation milestones:

Simulation	Date	Status	Notes
Initial Basic Modules	May 5, 2025	Passed	HalfAdder, FullAdder, and DFF modules verified
Adder Components	May 7, 2025	Failed	CarrySaveAdder had timing issues with carry propagation
Adder Components (Revised)	May 8, 2025	Passed	Fixed carry chain and verified correct operation
SPM Core	May 10, 2025	Failed	Sign handling issues for negative operands
SPM Core (Revised)	May 12, 2025	Passed	Corrected sign extension and verified with test cases
Binary-to-BCD	May 14, 2025	Failed	Conversion errors for large numbers

Continued on next page

Simulation	Date	Status	Notes
Binary-to-BCD (Revised)	May 15, 2025	Passed	Implemented proper Double Dabble algorithm
Display Controller	May 16, 2025	Failed	Scrolling logic had indexing errors
Display Controller (Revised)	May 17, 2025	Passed	Fixed scrolling logic and verified display operation
Full System	May 19, 2025	Failed	Timing issues between modules caused occasional errors
Full System (Final)	May 20, 2025	Passed	Added proper synchronization between modules

6 Team Member Contributions

6.1 Mennatallah Zaher

- Designed and implemented the SPM Controller (FSM)
- Developed the Binary-to-BCD converter module
- Created the project documentation and block diagrams
- Led the integration testing and debugging efforts

6.2 Farida Elanany

- Worked on SPM development and integrated it into the top module
- Designed the adder components (HalfAdder, FullAdder, Adder4Bit)
- Created comprehensive test benches for verification
- Handled sign correction logic for signed multiplication
- Performed FPGA testing and created the constraint files

6.3 Mahinour Abdelgawad

- Developed the Seven-Segment Display Driver

- Contributed to the final display and sign logic integration
- Implemented the user interface components (debouncer, button detector)
- Created the clock divider and synchronizer modules
- Managed the GitHub repository and version control

7 Conclusion

This project successfully demonstrates a complete, modular, and well-documented implementation of an 8x8 signed Serial-Parallel Multiplier on FPGA. The design achieves a balance between hardware efficiency and operational speed, with a user-friendly interface and robust output display. Through careful module design and extensive testing, we were able to overcome several challenges related to timing, synchronization, and sign handling.

The modular structure and clear documentation support maintainability and future enhancements. The project demonstrates our understanding of digital design principles, FPGA implementation techniques, and effective teamwork.

8 Future Improvements

For future iterations of this project, we would consider the following enhancements:

- Implementing a more efficient multiplication algorithm such as Booth's algorithm
- Adding a memory component to store and recall previous calculations
- Enhancing the user interface with additional display modes
- Optimizing the design for reduced power consumption

9 Appendix

9.1 Block Diagram

Block Diagram Link: `https://drive.google.com/file/d/SPM.drawio`
The complete block diagram is available at the Google Drive link above.

9.2 References

1. Parhami, B. (2010). Computer Arithmetic: Algorithms and Hardware Designs. Oxford University Press.

2. Xilinx. (2018). Artix-7 FPGA Family Data Sheet.

3. Digilent. (2016). Basys 3 FPGA Board Reference Manual.

4. Anthropic. (2025). Claude 3.7 Sonnet AI model. Used for document formatting and report structure optimization.

5. OpenAI. (2024). GPT-4 AI model. Used for reference during initial project planning.