Fixed-Point MAC Peripheral Design Proposal

Design Specification

Our team proposes to design a fixed-point Multiply-Accumulate (MAC) peripheral specifically optimized for the 8-bit RISC CPU architecture. This peripheral will enhance the computational capabilities of the base CPU by providing hardware acceleration for multiplication and accumulation operations.

Key Design Features:

- Implementation of an 8×8→16-bit fixed-point MAC unit with **multi-cycle operation**
- Custom memory-mapped interface with optimized register layout
- Configurable accumulator behavior with saturation protection
- Status flags for operation completion and overflow detection
- Debug output ports for real-time monitoring

System Implementation

MAC Architecture

Our MAC implementation will feature:

1. **Enhanced Input Buffering**: Dual-buffered input registers that enable pipelined operation by allowing new operands to be loaded while the current multiplication is still in progress. This allows back-to-back MAC operations with minimal latency between operations.

2. **Configurable Accumulation Modes**:

3. Standard mode: acc += A×B
4. Clear-and-multiply mode: acc = A×B

5. Saturating accumulation to prevent overflow

6. **Status Register**: Provides operation status including overflow flag and ready signal

7. **Operation Latency**: Exactly 2 clock cycles from start signal to result availability

Memory-Mapped Interface

Our memory map design uses five addresses:

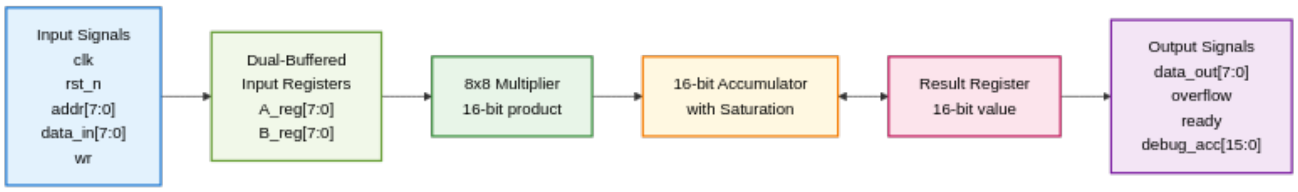
- 0xF0: Input Register A (write-only)
- 0xF1: Input Register B (write-only)
- 0xF2: Control Register (bit 0: start, bit 1: reset, bit 2: mode select)
- 0xF3: Result Low Byte (read-only)
- 0xF4: Result High Byte and Status Flags (read-only, bits 0-6: high byte bits 6-0, bit 7: overflow)

Dual-Buffered Operation

The dual-buffered input design enables efficient pipelined operation:

1. While the MAC unit is processing the current operation, new A and B values can be written to the input registers
2. Once the current operation completes (ready signal asserts), setting the start bit will begin the next operation immediately
3. This allows for continuous operation with a throughput of one MAC operation every 2 cycles after the initial latency

Block Diagram



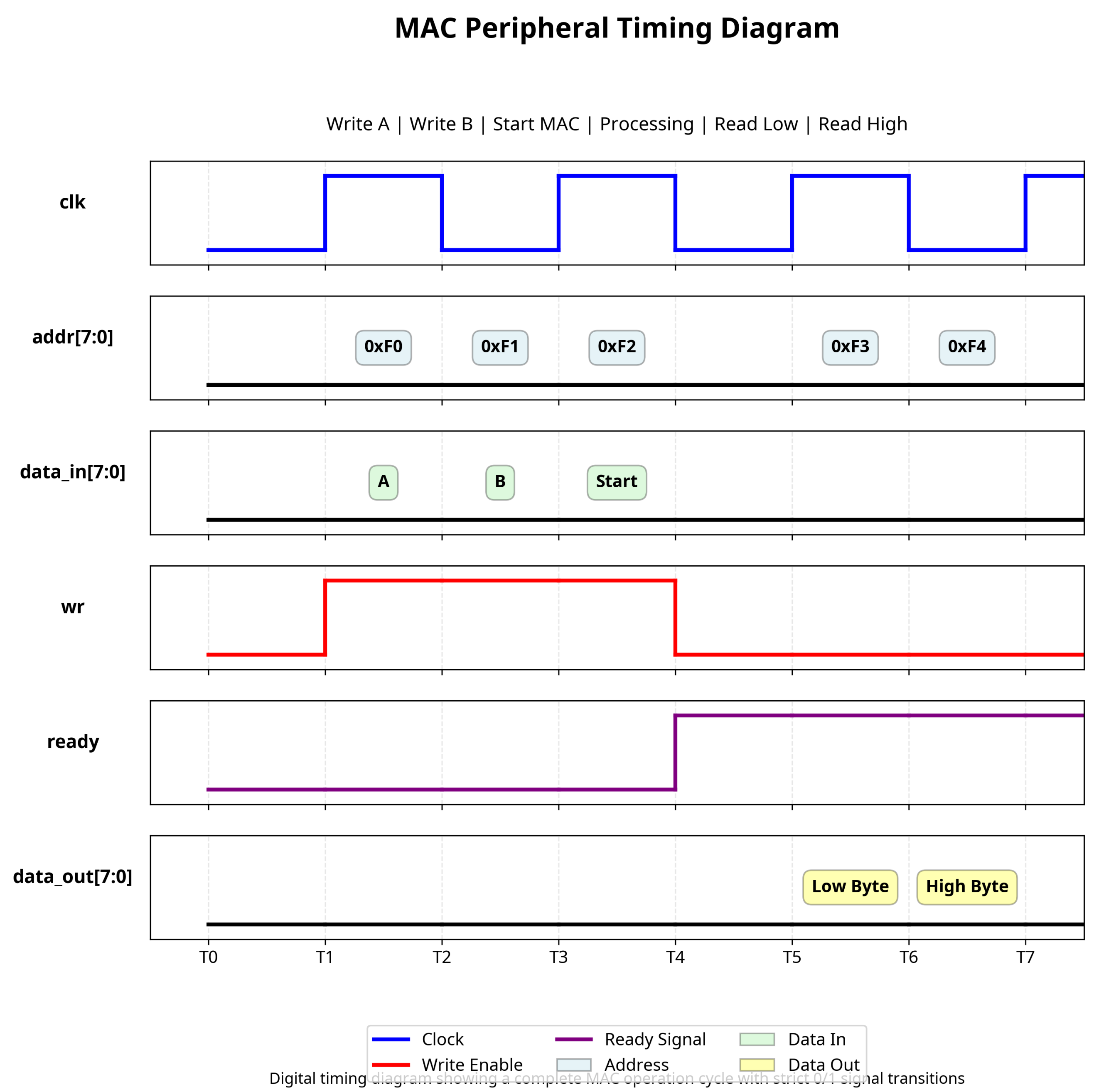
I/O Definition Table

Signal Name	Direction	Width (bits)	Description
clk	Input	1	System clock (max 50MHz)
rst_n	Input	1	Active-low global reset from TinyTapeout
addr	Input	8	Address bus

Signal Name	Direction	Width (bits)	Description
data_in	Input	8	Data input bus
wr	Input	1	Write enable
data_out	Output	8	Data output bus
overflow	Output	1	Overflow indicator
ready	Output	1	Result ready flag (asserts when computation is complete)
debug_acc	Output	16	Debug: Current accumulator value

Timing Diagram

The following timing diagram illustrates the key time events in our MAC peripheral design:



Digital timing diagram showing a complete MAC operation cycle with strict 0/1 signal transitions

The diagram shows a complete MAC operation cycle: 1. CPU writes operand A to register 0xF0 2. CPU writes operand B to register 0xF1 3. CPU writes to control register 0xF2 with start bit set 4. MAC unit processes the operation (multiplication and accumulation) over exactly 2 clock cycles 5. Ready signal asserts when result is available 6. CPU reads result low byte from register 0xF3 7. CPU reads result high byte from register 0xF4

Test Plan

Test Environment

- RTL simulation using Verilog testbench
- FPGA implementation on TinyTapeout platform

Test Approach

1. **Functional Testing**
2. Verify register read/write operations
3. Test basic multiplication with various input combinations
4. Validate all accumulation modes

5. Check overflow detection and saturation behavior

6. **Performance Verification**

7. Measure operation latency at 50MHz clock (confirm 2-cycle latency)
8. Verify timing constraints are met

9. Confirm proper ready signal assertion after computation

10. **Integration Testing**

11. Test CPU interface with memory-mapped access patterns
12. Verify proper handling of back-to-back operations
13. Test operation pipelining with dual-buffered inputs to achieve one operation every 2 cycles

Test Implementation

- Automated test script will generate test vectors and verify results
- Hardware validation using UART interface to control inputs and monitor outputs
- Results will be logged and compared against expected values