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
- Dual-buffered input operand registers allows the CPU to load new data while a current MAC operation is still in progress.

System Implementation

MAC Architecture

Our MAC implementation will feature:

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.

Configurable Accumulation Modes:

- Standard mode: acc += A×B

- Clear-and-multiply mode: acc = A×B

- Saturating accumulation to prevent overflow

Status Register:

- Provides operation status including overflow flag and ready signal

Operation Latency:

- Exactly 2 clock cycles from start signal to result availability

Fixed-Point Number Format

Our MAC peripheral will implement the following fixed-point format:

- **Input Format**: Q7.0 (8-bit signed integers)
- 1 sign bit + 7 magnitude bits
- Range: -128 to +127

- This format maximizes the dynamic range while maintaining compatibility with the 8-bit CPU architecture

- **Accumulator Format**: Q15.0 (16-bit signed integers)

- 1 sign bit + 15 magnitude bits
- Range: -32,768 to +32,767

- Provides sufficient headroom for accumulating multiple products without overflow

- **Saturation Logic**: When enabled, results exceeding the 16-bit range will saturate at the maximum/minimum values rather than wrapping around

CPU Interface Protocol

The MAC is memory-mapped into the CPU’ s I/O address space, allowing the CPU to control it using standard load/store instructions. No special bus handshake or DMA is used; instead, the CPU uses a simple polling mechanism to coordinate with the peripheral:

Handshake Mechanism:

- The CPU initiates operations by writing to the control register with the start bit set

- The MAC peripheral asserts the ready signal when computation is complete
- The CPU can poll the ready signal by reading the status register (bit 1 of 0xF5)
- No interrupt mechanism is implemented; the CPU must poll the ready signal

Data Transfer:

- Standard memory-mapped writes for input data and control
- Standard memory-mapped reads for results and status

Timing Requirements:

- The CPU must wait for the ready signal before outputting results
- The CPU can load new data while an operation is in progress
- The CPU must set the start bit to begin a new operation

Memory-Mapped Interface

Our memory map design uses five addresses:

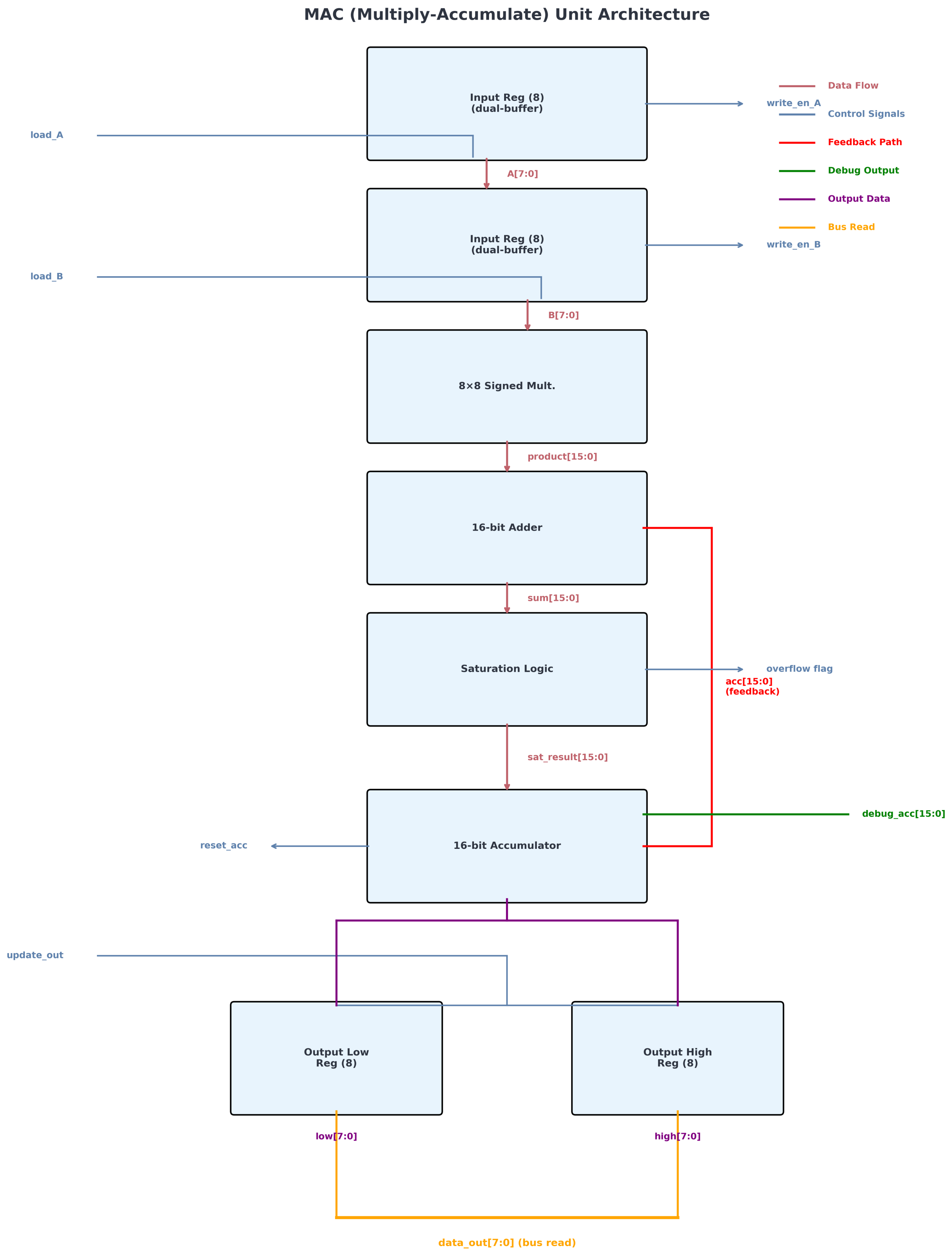
- 0xF0: Input Register A (write-only)
- 0xF1: Input Register B (write-only)
- 0xF2: Control Register (write-only, bit 0: start, bit 1: reset, bit 2: mode select)
- 0xF3: Result Low Byte (read-only)
- 0xF4: Result High Byte (read-only)
- 0xF5: Result Status (read-only bit 0: overflow, bit 1: ready)

Dual-Buffered Operation

The dual-buffered input design enables efficient pipelined operation:

1. While the MAC unit is processing the current operation, new values can still 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

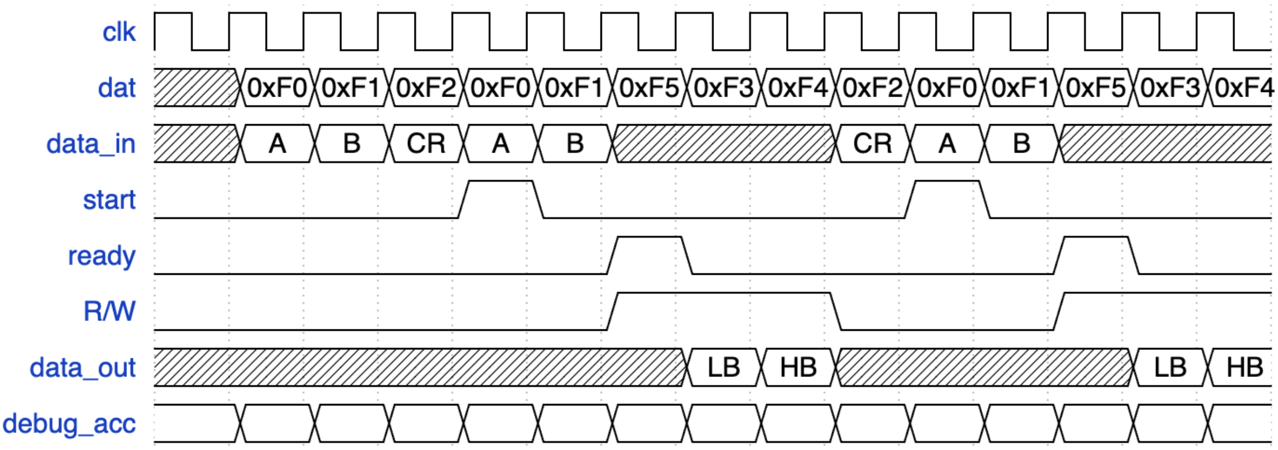


I/O Definition Table

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
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:



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 status byte from register 0xF5
7. CPU reads result low byte from register 0xF3
8. CPU reads result high byte from register 0xF4

Test Plan

Key Function Tests

Core MAC Operation

- Test basic multiplication with positive and negative inputs
- Confirm correct result reading sequence (low byte then high byte)
- Verify 2-cycle latency from start signal to ready signal

Accumulation Modes

- Verify standard accumulation mode (acc += A×B)
- Test clear-and-multiply mode (acc = A×B)
- Confirm saturation behavior with large values

Control and Status

- Validate ready signal assertion timing
- Test overflow detection with boundary values
- Verify reset functionality

Test Implementation

- Simple python test focusing on key functions