# CSE100 Lab 5 – Multiply/Add with Ready/Valid

## Contents

# 1 Submission

- Due: Tuesday 11/19/2024 at the end of your lab section.
- Submit your Lab Write-Up to Gradescope.
- Submit your code to the Gradescope autograder. (You have unlimited submission attempts).

# 2 Goals

You will implement a finite state machine to control a ready/valid interface for a sequential circuit that performs a "Multiply-Add" operation. Specifically, this circuit computes `(a_i * b_i) + c_i` over several clock cycles.

# 3 Background

You are given a partially-completed sequential circuit that performs a "Multiply-Add" operation, meaning it computes `(a_i * b_i) + c_i` after a certain number of clock cycles.

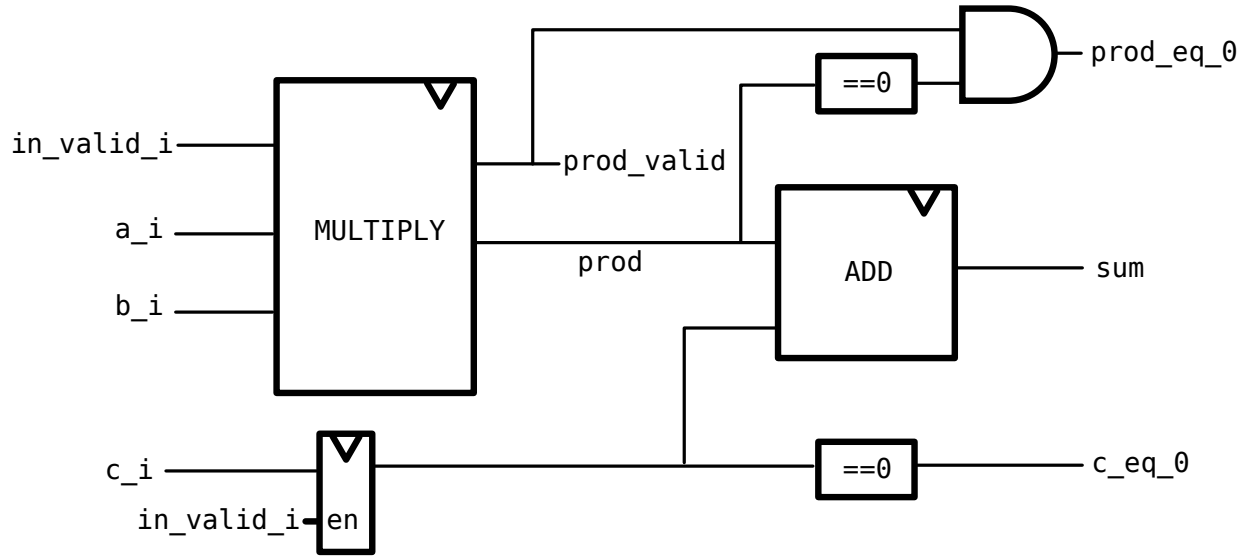
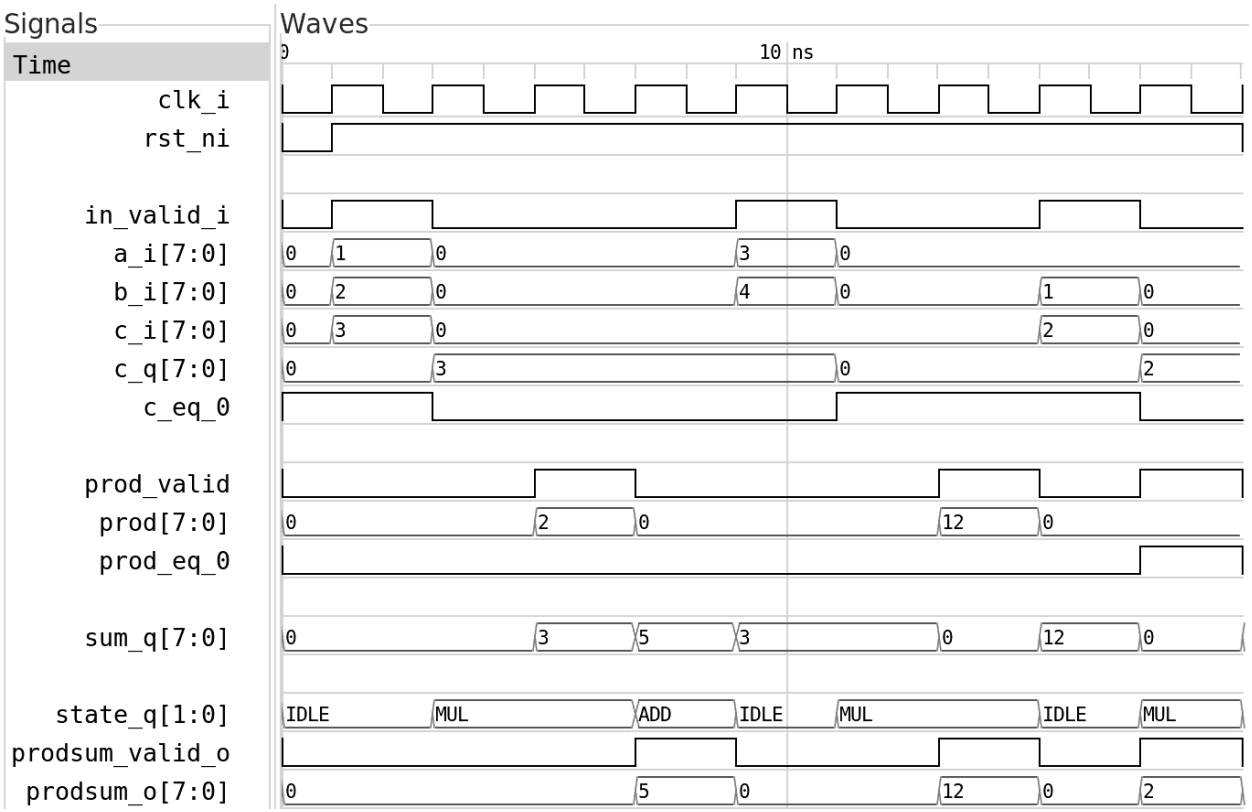



Figure 1: Schematic

Components:

- **Multiply Unit** takes in `a_i`, `b_i`, and `in_valid_i`. After 1 or 2 clock cycles, the product `prod` is generated, and `prod_valid` becomes active.
- **Add Unit** outputs the sum after 1 cycle.

You will need to finish this "Multiply-Add" unit according to the behavior denoted in the following waveform:




Usually, the circuit state should go from `IDLE` to `MUL` to `ADD` to `IDLE`. However, there are 2 exceptions:

- If `prod` is zero, the addition should be skipped and `prodsum_o` should be immediately set to `c_q`.
- Similarly, if `c_q` is zero, the addition should be skipped and `prodsum_o` should be immediately set to `prod`.

Note: While the provided waveform screenshot does not show the "ready" signals, your implementation must correctly manage them to ensure proper timing. The ready signals should prevent multiple simultaneous inputs and coordinate data flow correctly.

# 4 Lab

1. Complete the `always_comb` block marked as "`TODO`" in the `muladd` module.
2. Simulate your design using commands specified in the `"README.md"`. Submit to the autograder until you get 100%.
3. Complete the write-up questions.

# 5 Write-Up

The following questions can be hand drawn or drawn digitally (https://inkscape.org/release, https://draw.io/).

1. Draw a bubble diagram for the state machine. Denote the inputs required to transition into the next state.

2. Draw the MUX tree for `in_ready_o`.
3. Draw the MUX tree for `prodsum_o`.
4. Draw the MUX tree for `prodsum_valid_o`.
5. Draw the MUX tree for `state_d`.