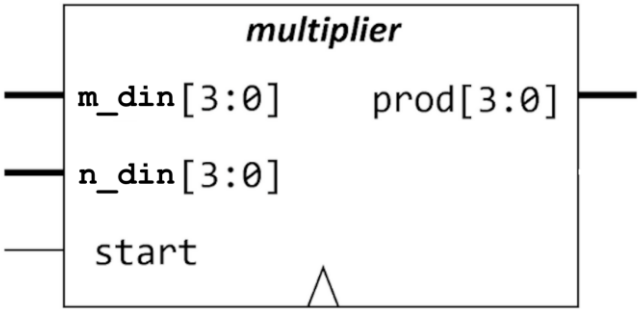
Name(s): Aaron Wang
CSE 20221 Logic Design
HW09: Design of a Hardware Multiplier

Overview

In HW03, we examined the implementation of the shift-and-add algorithm for multiplication as an albaCore assembly language program. In this assignment, we examine another approach as a custom hardware implementation using a datapath and FSM controller derived from a high-level state machine. The system should multiply two 4-bit unsigned integers `m_din, n_din` and produce a 4-bit result `prod` (you do not need to worry about overflow).



Setup

Here is a list of each circuit you will work with in this assignment, its description, and its state of completion.

Circuit	Description	State of Completion
`mult_datapath`	Multiplier datapath	From scratch
`|--> load_register`**X**	Registers in `mult_datapath` Replace **X** with the correct number (e.g., `load_register2` if 2 bit registers are needed in the datapath)	Use the circuit from class examples (see `load_register4` in `L20_timer_hlsm_complete`) - expand/shrink as needed
`mult_controller`	Multiplier FSM (connects state register and combinational logic)	From scratch
`|--> fsm_state_reg`**Y**	State register for multiplier FSM Replace **Y** with the correct number (e.g., `fsm_state_reg2` if 2 bits are needed to encode FSM states)	Use the circuit from class examples (see `fsm_state_reg` in `L20_timer_hlsm_complete`) - expand/shrink as needed
`|--> mult_comb`	Combinational logic for multiplier FSM	Input truth table and generate using Combinational Analysis
`mult_proc`	Multiplier processor (connects datapath and controller)	From scratch

High-Level State Machine

The HLSM for the system is specified by the table below:

Inputs: `m_din[3:0], n_din[3:0], start`
Outputs: `prod[3:0]`
Internal variables: `m[3:0], n[3:0], mask[3:0], i[2:0]`

State	Action	Transition
`WAIT`		`if (start) goto INIT` `else goto WAIT`

`INIT`	`prod ← 0`
	`i ← 4`
	`mask ← 1`
	`m ← m_din`
	`n ← n_din`
		`goto LOOP`

`LOOP`		`if (i == 0) goto FINISH`
		`else if ((mask & n) == 0) goto SHIFT`
		`else goto ADD_PROD`

`ADD_PROD`	`prod ← prod + m`	`goto SHIFT`

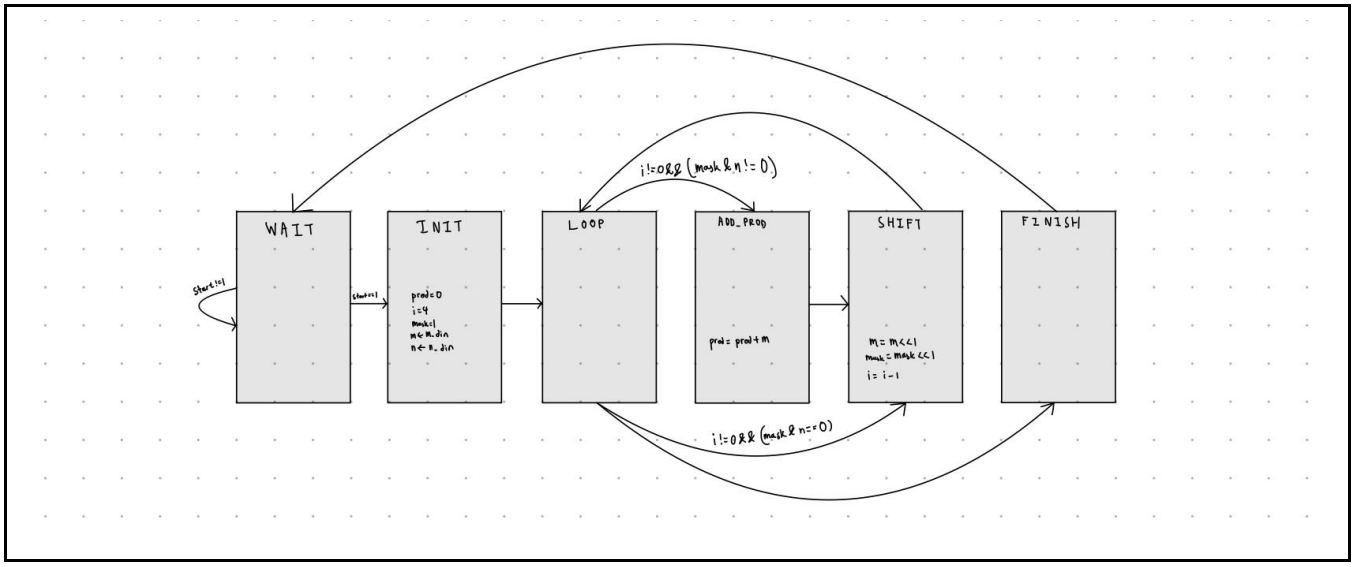
`SHIFT`	`m ← m << 1`
	`mask ← mask << 1`
	`i ← i - 1`
		`goto LOOP`

`FINISH`		`goto WAIT`

Problem 0 - HLSM Diagram

Convert the HLSM table on the last page to an equivalent HLSM diagram showing states as circles, transitions as arrows with labels, etc. The diagram should match the table 1:1 as in our examples in class. You may draw this by hand or using any digital tool (Google Slides or PowerPoint works well). Make sure your drawing is neat and easy to read.

HLSM diagram drawing



Problem 1 - Stage Table

Complete the stage table based on the HLSM table. (Add rows as necessary.)
(We did this for the *Timer* example on page 9 of these notes)

Destination (Load Register)	Source(s)	Control Signal(s)
`prod`	`0:0` `1:prod+m`	`en_prod` `sel_prod`
`i`	`0:4` `1:i-1`	`en_i` `sel_i`
`mask`	`0:1` `1:mask<<1`	`en_i` `sel_i`
`m`	`0:m_din` `1:m<<1`	`en_m` `sel_m`
`n`	`n_din`	`en_n`

Problem 2 - Condition Table

Complete the conditions table based on the HLSM table. (Add rows as necessary.)
(We did this for the *Timer* example on page 9 of these notes)

Condition	Flag
`start`	`start`
`i==0`	`finish`
`i!=0 && m&n==0`	`shift`

Problem 3 - FSM Table

Complete the FSM table using the control signals and conditions from the stage table and conditions table. Suggestion: copy the HLSM table and modify it with the control signals and flags.
(We did this for the *Timer* example on page 13 of these notes)

State	Action(s)	Transition(s)
`WAIT`		`if (start) goto INIT` `else goto WAIT`

`INIT`	`en_prod = 1`
	`en_i = 1`
	`en_mask = 1`
	`en_m = 1`
	`en_n = 1`
		`goto LOOP`

`LOOP`		`if (finish) goto FINISH`
		`else if (shift) goto SHIFT`
		`else goto ADD_PROD`

`ADD_PROD`	`en_prod = 1` `sel_prod = 1`	`goto SHIFT`

`SHIFT`	`en_m = 1`
	`sel_m = 1`
	`en_mask = 1`
	`sel_mask = 1`
	`en_i = 1`
	`sel_i = 1`
		`goto LOOP`

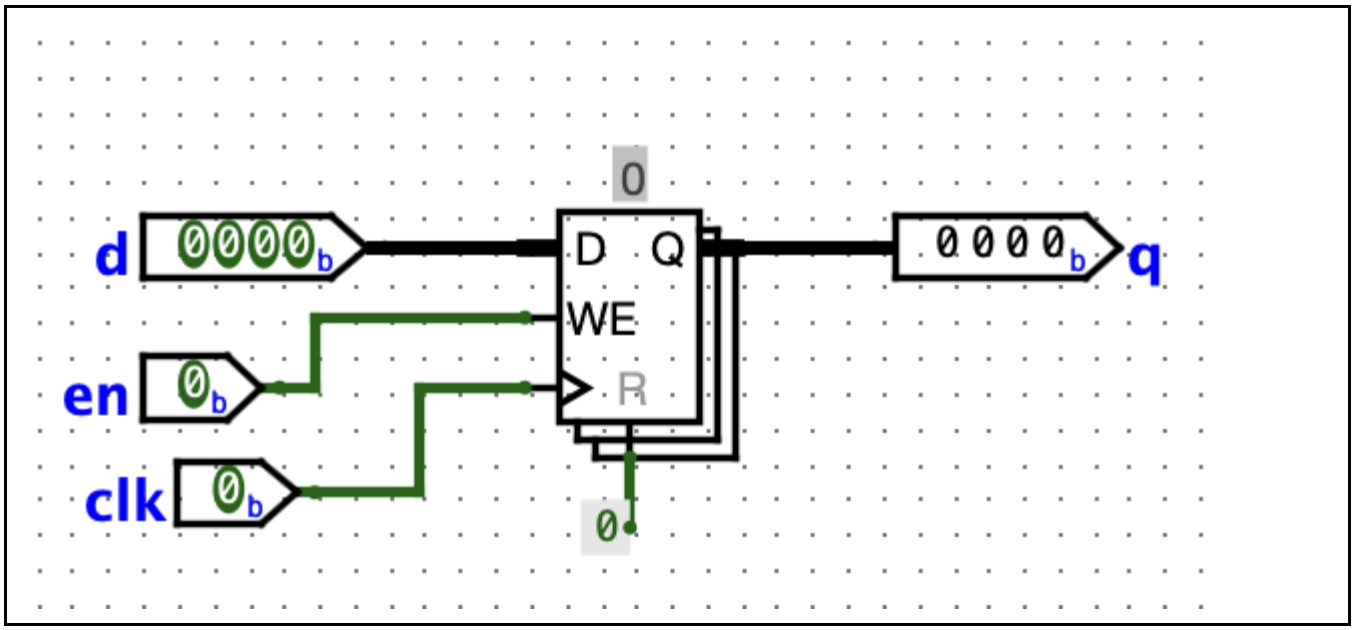
`FINISH`		`goto WAIT`

Problem 4 - Datapath and Sub-circuits

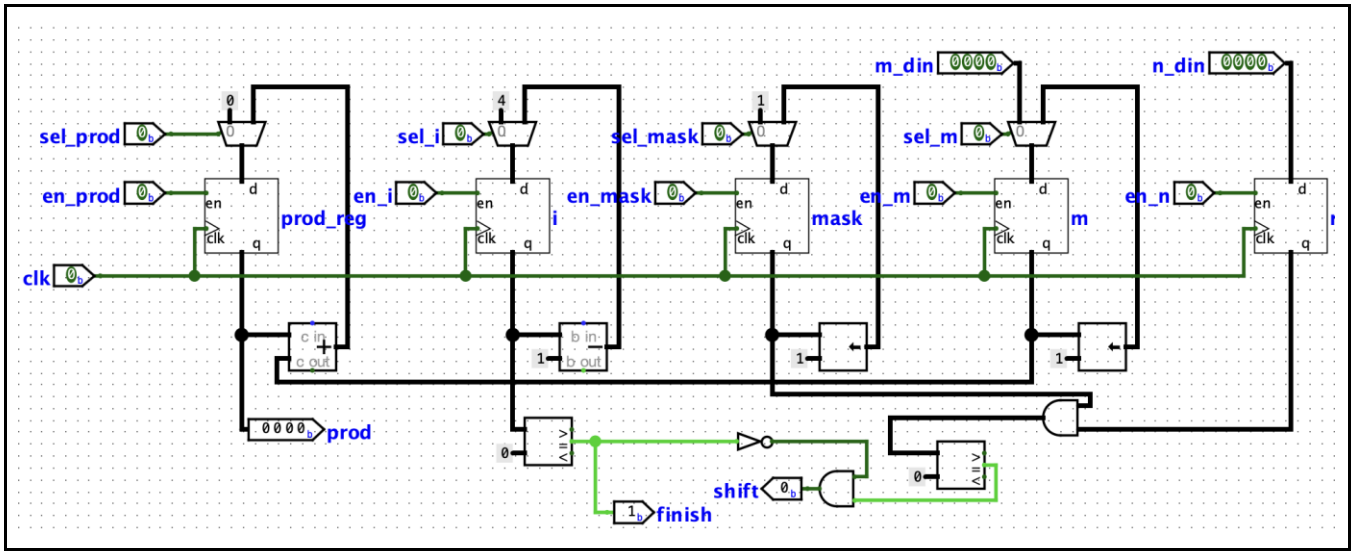
Complete the load register and the `mult_datapath` circuits based on your tables from Problem 1 and Problem 2.

(We did this for the *Timer* example on page 12 of these notes, and then in Logisim-evolution)

load_registerX circuit - remember, X should be replaced with a number (see table above)



mult_datapath circuit

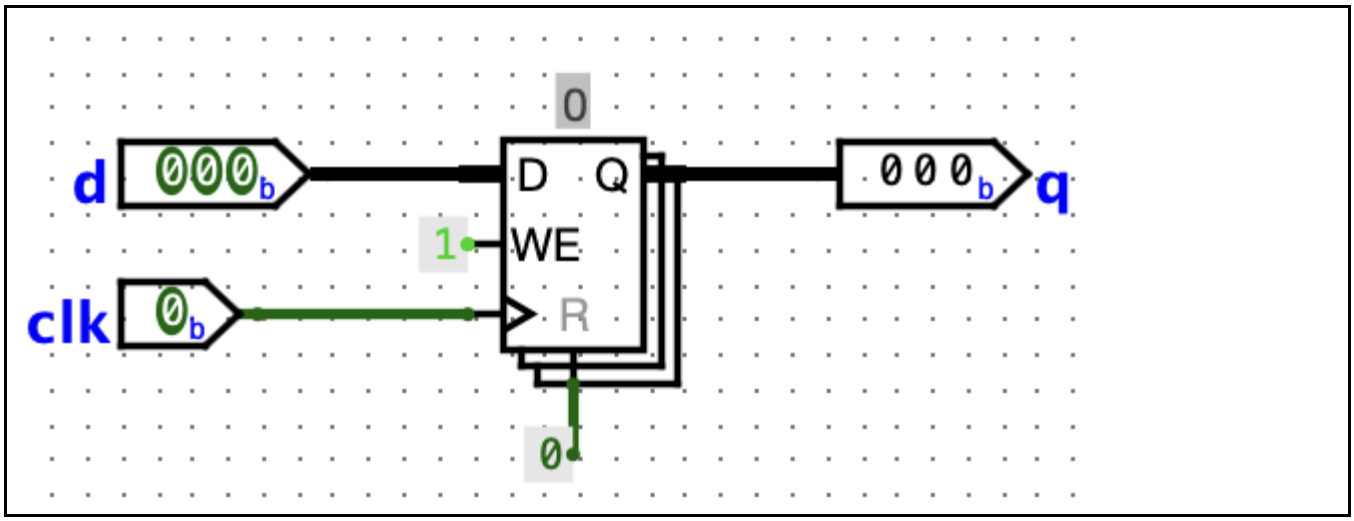


Problem 5 - Controller and Sub-circuits

Complete the state register, combinational logic (`mult_comb`), and `mult_controller` circuit based on your FSM table from Problem 3.

(We looked at this for the *Timer* example on page 15 of these notes)

fsm_state_regY circuit - remember, Y should be replaced with a number (see table above)



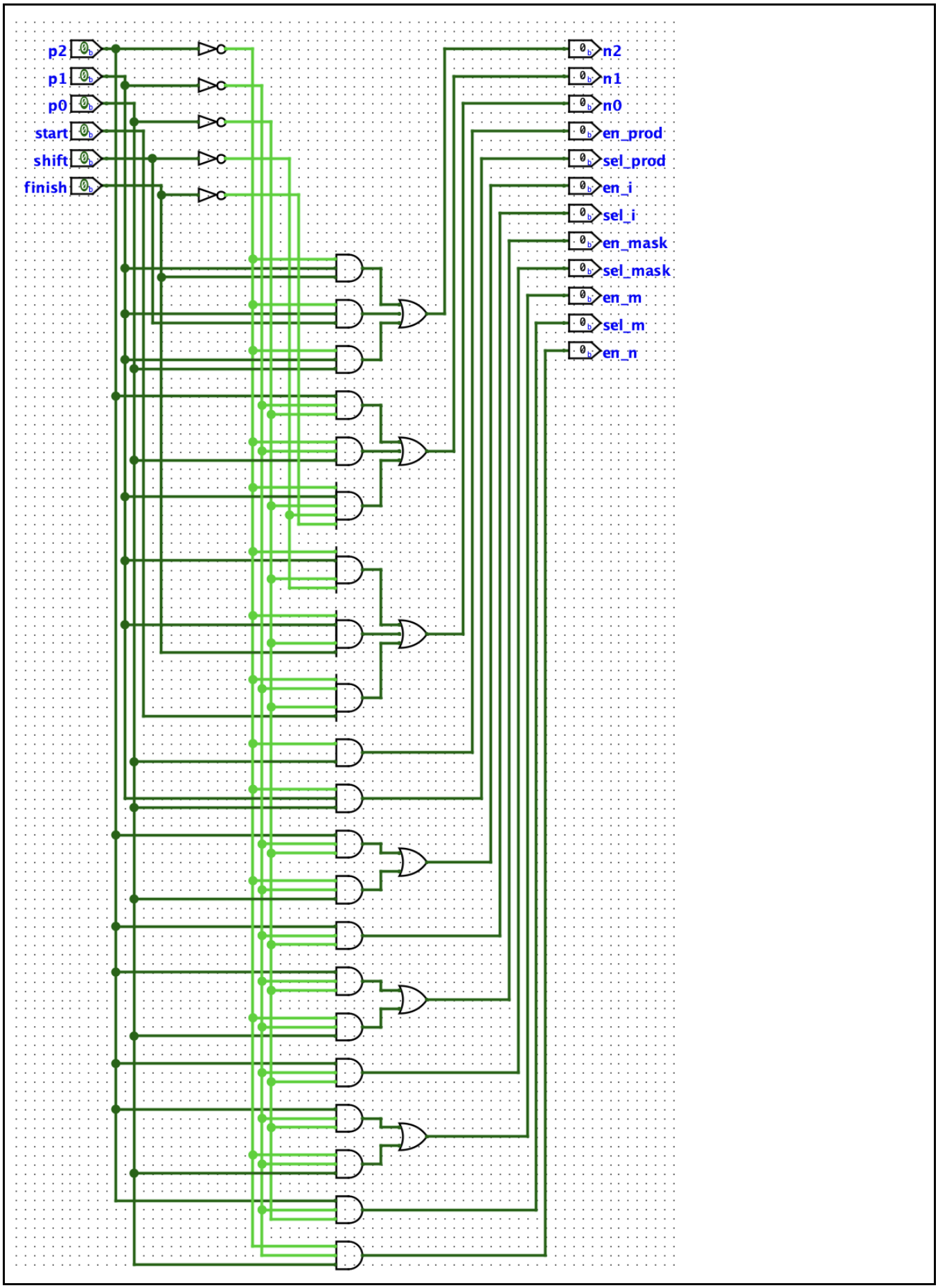
Export your FSM truth table to a file and copy the file contents here.

See the Panopto recording "HW09 Truth Table Tips" for suggestions on how to speed up the truth table creation (tl;dw: use vim or another text editor's find and replace).

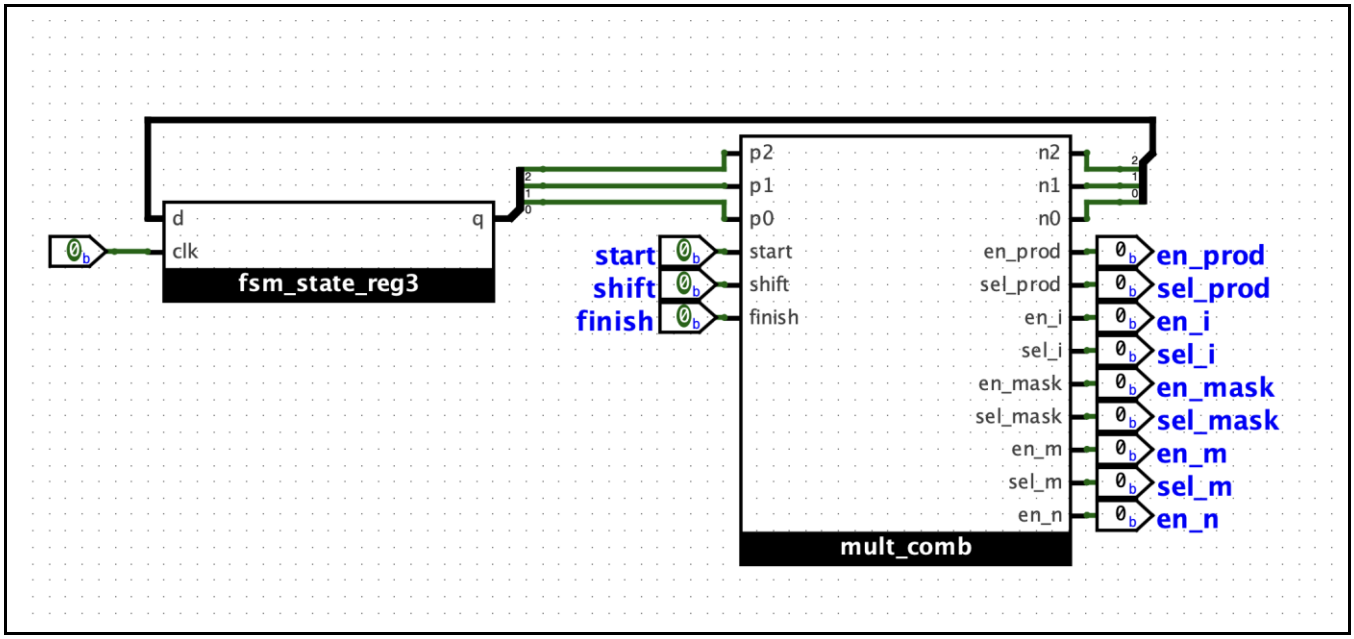
mult_comb truth table

```
p2 p1 p0 start shift finish | n2 n1 n0 en_prod sel_prod en_i sel_i en_mask sel_mask en_m sel_m en_n
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
0  0  0  0     x     x      | 0  0  0  0       0        0    0     0       0        0    0     0
0  0  0  1     x     x      | 0  0  1  0       0        0    0     0       0        0    0     0

0  0  1  x     x     x      | 0  1  0  1       0        1    0     1       0        1    0     1

0  1  0  x     x     1      | 1  0  1  0       0        0    0     0       0        0    0     0
0  1  0  x     1     0      | 1  0  0  0       0        0    0     0       0        0    0     0
0  1  0  x     0     0      | 0  1  1  0       0        0    0     0       0        0    0     0
|
0  1  1  x     x     x      | 1  0  0  1       1        0    0     0       0        0    0     0

1  0  0  x     x     x      | 0  1  0  0       0        1    1     1       1        1    1     0

1  0  1  x     x     x      | 0  0  0  0       0        0    0     0       0        0    0     0

1  1  0  x     x     x      | 0  0  0  0       0        0    0     0       0        0    0     0
1  1  1  x     x     x      | 0  0  0  0       0        0    0     0       0        0    0     0
```

mult_comb circuit


p2
p1
p0
start
shift
finish
n2
n1
n0
en_prod
sel_prod
en_i
sel_i
en_mask
sel_mask
en_m
sel_m
en_n

mult_controller circuit

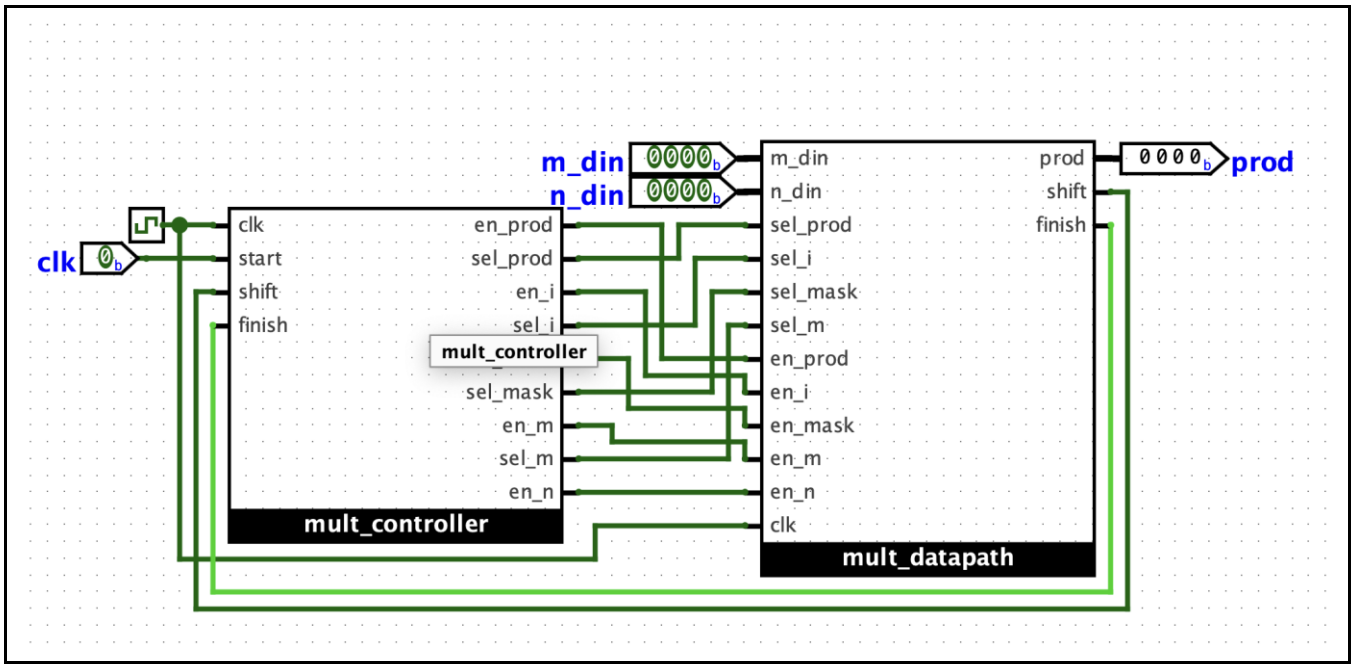


Problem 6 - Processor Circuit

Complete the `mult_processor` circuit by connecting the datapath and controller.

You are not required to make a custom appearance for the datapath and controller, but try to keep the wiring as clean as possible.

mult_processor circuit

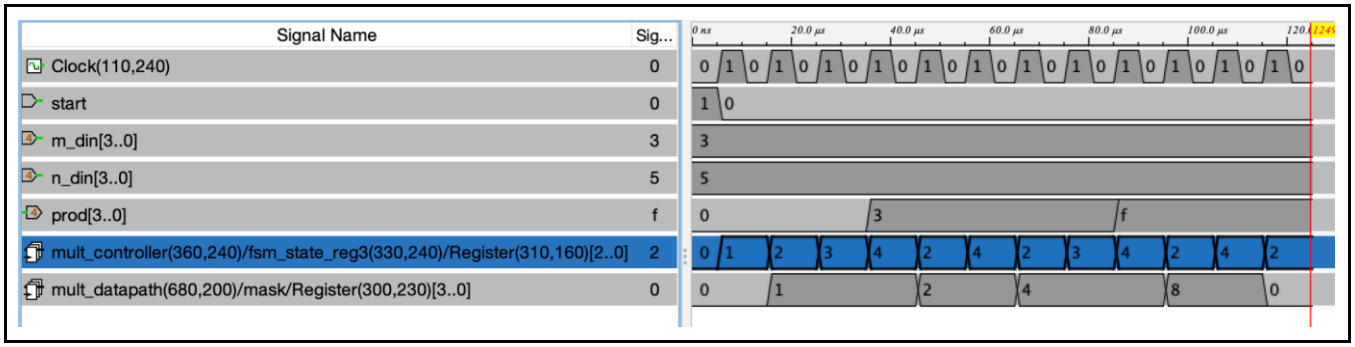


Problem 7 - Processor Timing Diagram and Dropbox Upload

Test `mult_processor` for the case where `m_din = 3, n_din = 5`. You will generate a timing diagram showing all inputs and outputs for `mult_processor`. Also include the current state in `mult_state_reg` (the register's value) and the `mask` register from the datapath.

Suggestion: you may need to do some debugging, so add signals as needed to your timing diagram, debug as necessary until everything is working properly, and then delete the extra signals before taking a screenshot of your timing diagram

Timing diagram showing mult_processor works correctly for `m_din = 3, n_din = 5`
Make sure signal names are visible and your screenshot is zoomed enough to read each value (if needed, you can take multiple screenshots and list them sequentially in the box)



Please upload a copy of your .circ file(s) to one group member's student dropbox. List the dropbox full, absolute path here – from within the directory where you placed the .circ file(s), use this command to get the ***absolute*** *path:* `pwd -P`

```
/escnfs/courses/fa24-cse-20221.01/dropbox/awang27/HW09
```

Deliverable Checklist

HLSM Diagram (5 points)

	Deliverable	Points
1	Convert the HLSM table to a diagram	5

Stage Table (10 points)

	Deliverable	Points
2	Complete the stage table	10

Condition Table (5 points)

	Deliverable	Points
3	Complete the condition table	5

FSM Table (10 points)

	Deliverable	Points
4	Complete the FSM Table	10

Datapath and Sub-circuits (17 points)

	Deliverable	Points
5.1	Load register	2
5.2	Datapath	15

Controller and Sub-circuits (17 points)

	Deliverable	Points
6.1	FSM state register	2
6.2	FSM combinational logic truth table	5
6.3	FSM combinational logic circuit	5
6.4	Full controller circuit	5

Processor (10 points)

	Deliverable	Points
7	Processor circuit (controller connected to datapath)	10

Processor Timing Diagram and Dropbox Path (11 points)

	Deliverable	Points
8.1	Processor timing diagram showing 3 * 5 = 15	10
8.2	*Absolute path* to .circ file(s) in dropbox	1