University of Washington – Computer Science & Engineering

Winter 2024 Instructor: Justin Hsia 2024-03-05

CSE 369 QUIZ 3

Name: _Molly_Model____________________

Student ID Number: _1234567________

Please do not turn the page until 11:40.

Instructions

- This quiz contains 4 pages, including this cover page.
- Show scratch work for partial credit, but put your final answers in the boxes and blanks provided.
- The quiz is closed book and closed notes.
- Please silence and put away all cell phones and other mobile or noise-making devices.
- Remove all hats, headphones, and watches.
- You have 60 (+10) minutes to complete this quiz.

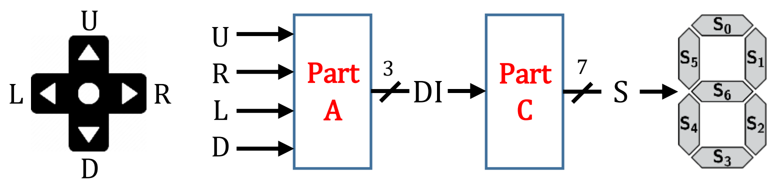
Advice

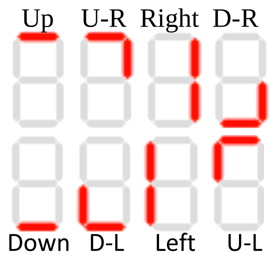
- Read questions carefully before starting. Read *all* questions first and start where you feel the most confident to maximize the use of your time.
- There may be partial credit for incomplete answers; please show your work.
- Relax. You are here to learn.

Question	Points	Score
(1) Decoders	13	13
(2) Routing Elements	10	10
(3) Shift Registers	10	10
Total:	**33**	**33**

Question 1: Decoders [13 pts]

We are building a **7-seg decoder circuit** for a **directional pad** (D-pad), which has a push-button (1 when pushed) for each of the 4 cardinal directions: U(p), R(ight), L(eft), and D(own). There is a physical restriction that at most two neighboring buttons can be pressed simultaneously (*e.g.*, $UR\overline{L}\overline{D}$ is possible but not $\overline{U}RL\overline{D}$ or $URL\overline{D}$), leading to 8 possible indicated directions on the 3-bit bus DI, numbered clockwise from Up = 0b000 to Up-Right = 0b001 around to Up-Left = 0b111.

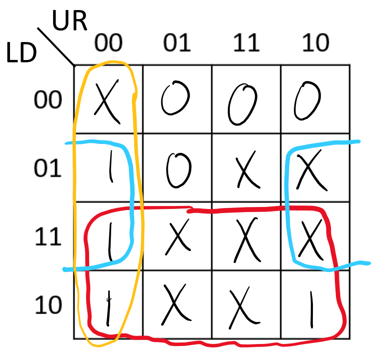




(A) Complete the truth table. [4 pt]

U	R	L	D	DI₂	DI₁	DI₀
0	0	0	0	X	X	X
0	0	0	1	1	0	0
0	0	1	0	1	1	0
0	0	1	1	1	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	1	1	0	X	X	X
0	1	1	1	X	X	X
1	0	0	0	0	0	0
1	0	0	1	X	X	X
1	0	1	0	1	1	1
1	0	1	1	X	X	X
1	1	0	0	0	0	1
1	1	0	1	X	X	X
1	1	1	0	X	X	X
1	1	1	1	X	X	X

(B) In the space below, solve for the minimal logical expression for **DI₂**. [4 pt]



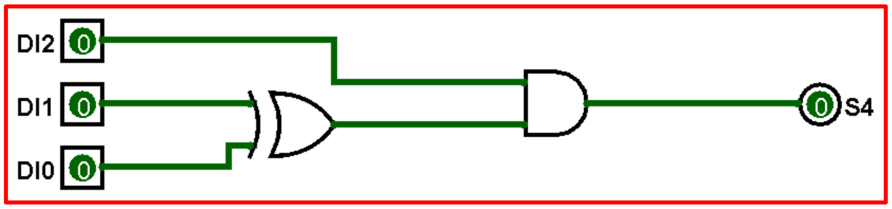
LD \ UR	00	01	11	10
00	X	0	0	0
01	1	0	X	X
11	1	X	X	X
10	1	X	X	1

$DI_2 = L + \overline{R}D$

or

$DI_2 = L + \overline{U}\overline{R}$

(C) For the 7-seg signal numbering and outputs shown above (lit/red = 1), draw the minimal logic for S_4 in terms of DI_2, DI_1, and DI_0. [3 pt]



DI₂	DI₁	DI₀	S4
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(D) *Briefly* describe what's problematic about this D-pad decoder circuit. [2 pts]

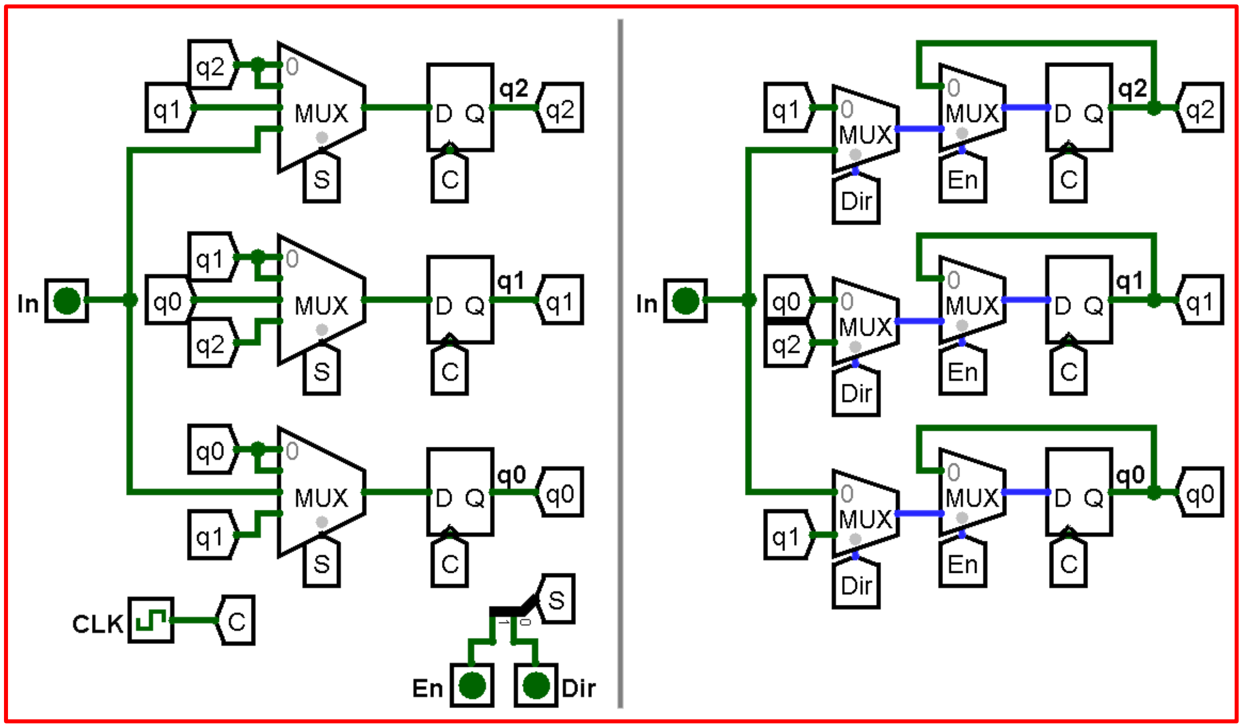
No buttons pressed ($\overline{U}\overline{R}\overline{L}\overline{D}$) is a don't care so it will be a potentially unknown output and, regardless of what it is, it will indicate a direction when nothing is being pressed. Should add a Valid output signal to this system.

Question 2: Routing Elements [10 pts]

We are creating an *enabled* 3-bit bidirectional shifter (can shift in both directions), which takes input bits **En** (short for Enable), **Dir**, and **In**. When enabled, we shift the current bits either to the right (*i.e.*, into the less significant bit) when **Dir** = 1 or to the left when **Dir** = 0 and shift in **In**. When not enabled, the state bits remain the same.

(A) Draw the circuit diagram below using *logic gates* and *routing elements* discussed in class. Assume the clock inputs are connected properly for you. You may use multiple copies of a signal name (*e.g.*, `q2`, `q1`, `q0`), which are assumed connected to the same net/wire. [5 pt]

Two possible solutions shown below:



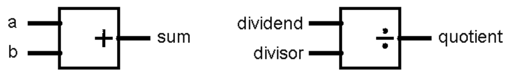
(B) In the Verilog test bench below, fill in the blanks to indicate how the state of our bidirectional shifter updates. [5 pts]

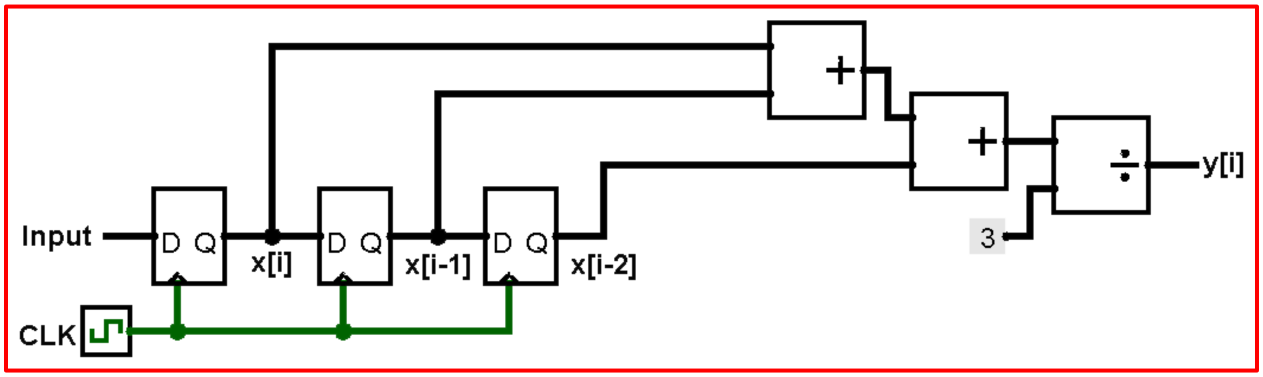
```
initial begin                                   // state: q2q1q0
   En <= 0; Dir <= 0; In <= 0;                  // state:  010
   @(posedge clk);  En <= 1;    // state: 010 (disabled)
                                // state: 100 (<< in 0)
   @(posedge clk);  {Dir, In} <= 2'd3;
   @(posedge clk);  In <= 0;    // state: 110 (>> in 1)
   @(posedge clk);  En <= 0;    // state: 011 (>> in 0)
   @(posedge clk);  $stop();    // state: 011 (disabled)
end
```

Question 3: Shift Registers [10 pts]

A **boxcar** (or moving average) **filter** computes the average of the last k samples and can be used to smooth out sudden input spikes (*i.e.*, it's a low-pass filter).

(A) Draw a circuit below that computes the moving average of the last 3 inputs, *i.e.*, $k = 3$ and $\mathbf{y}_i=(\mathbf{x}_i+\mathbf{x}_{i-1}+\mathbf{x}_{i-2})/3$, for a signal **x**, where $\mathbf{x}_{i-1}$ is the value of $\mathbf{x}_i$ from the previous clock cycle and so on. The given register is there for synchronization; you should *not* connect anything directly to `Input`. You can use registers, constants, and any number of the following logic blocks: [6 pts]

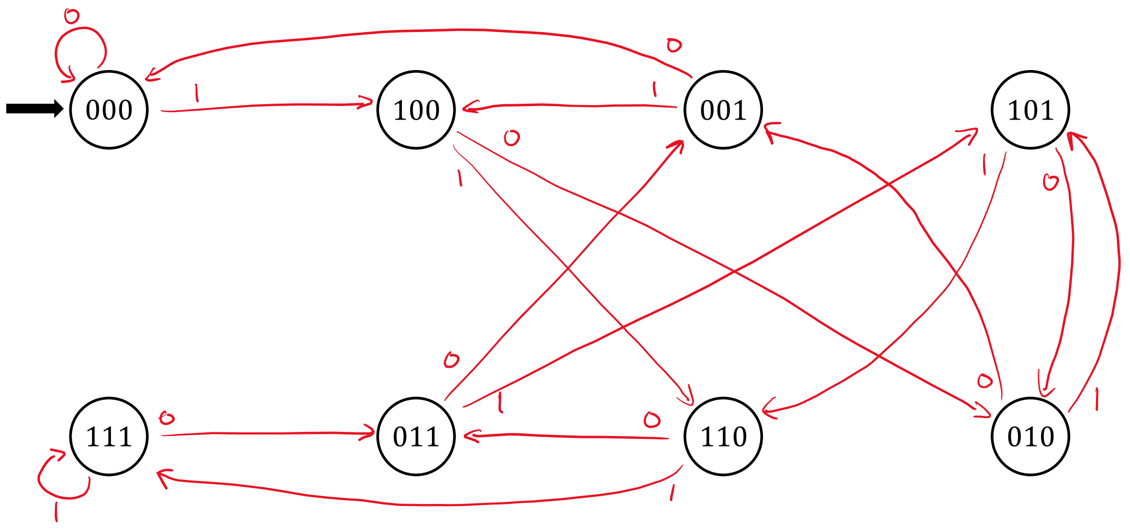




An alternative solution would be to divide all signals first before summing them together (5 logic blocks total).

(B) The time delay portion of the boxcar filter (*i.e.*, computing $\mathbf{x}_i$, $\mathbf{x}_{i-1}$, and $\mathbf{x}_{i-2}$) can be considered a shift register! Assuming 1-bit signals and using $\{\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{x}_{i-2}\}$ as our state bits and `Input` as our input, complete the corresponding Moore machine: the state names correspond to the output state bits, so only the value of `Input` should be shown on the transition arrows. [4 pts]

Right shifting!


000
100
001
101
111
011
110
010
0
1