University of Washington – Computer Science & Engineering
Winter 2022 Instructor: Justin Hsia 2022-03-08

CSE 369 QUIZ 3

Name: _Perry_Perfect_____

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.

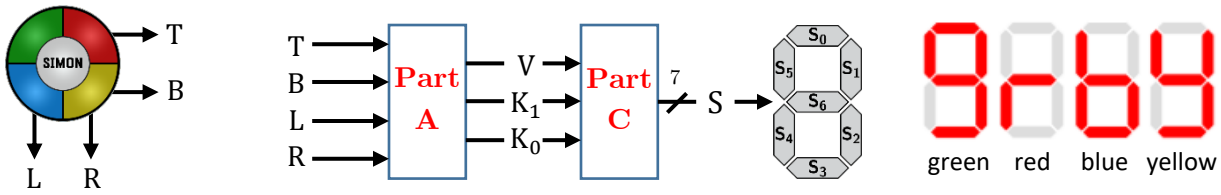
Advice

- 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	12	12
(2) Routing Elements	11	11
(3) Cryptography	9	9
Total:	32	32

Question 1: Decoders [12 pts]

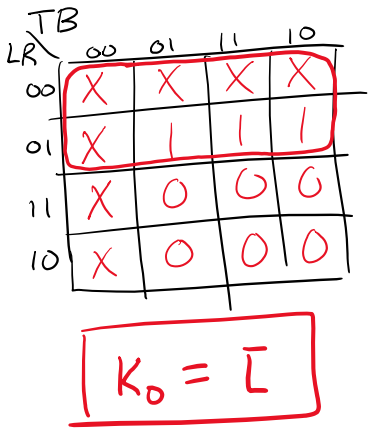
We are building a 7-seg decoder circuit for Simon, a color-based memory game with 4 buttons. The signals T(op), B(ottom), L(eft), and R(right) are high (1) only if a button in the corresponding column or row is being pushed. The 2-bit bus K represents which color is recognized (green = 00, red = 01, blue = 10, yellow = 11). The Valid signal (V) is high when at least one *button* is being pressed.



(A) Complete the truth table. We give priority to T and L. [4 pt]

T	B	L	R	V	K_1	K_0
0	0	0	0	0	X	X
0	0	0	1	0	X	X
0	0	1	0	0	X	X
0	0	1	1	0	X	X
0	1	0	0	0	X	X
0	1	0	1	1	1	1
0	1	1	0	1	1	0
0	1	1	1	1	1	0
1	0	0	0	0	X	X
1	0	0	1	1	0	1
1	0	1	0	1	0	0
1	0	1	1	1	0	0
1	1	0	0	0	X	X
1	1	0	1	1	0	1
1	1	1	0	1	0	0
1	1	1	1	1	0	0

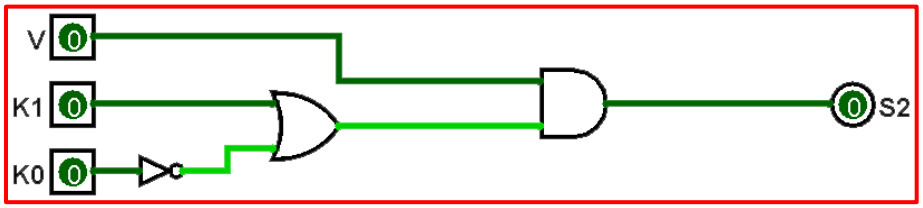
(B) In the space below, solve for the minimal logical expression for $\mathbf{K_0}$. [4 pt]



(C) For the 7-seg signal numbering and outputs shown above (lit/red = 1), draw the minimal logic for $\mathbf{S_2}$ in terms of V, K_1, and K_0. All signals should be off when V = 0. [4 pt]

K_1	K_0	S_2
0	0	1
0	1	0
1	0	1
1	1	1

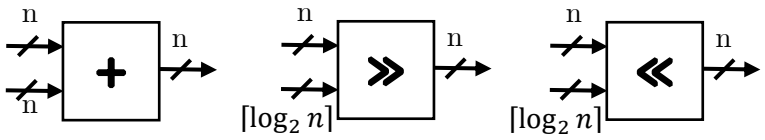
$S_2 = K_1 + \bar{K_0}$

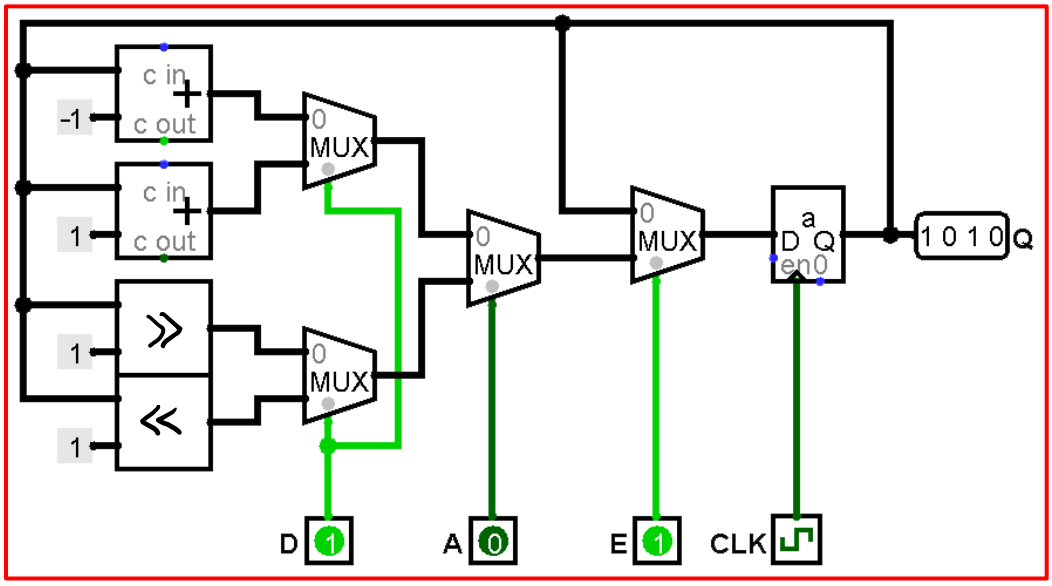


Question 2: Routing Elements [11 pts]

We are creating a sequential circuit with 1-bit inputs `E` (enable), `A` (action), and `D` (direction) and n-bit output `Q`. When not enabled, `Q` stays constant, otherwise, the circuit will either count (`A=0`) or shift (`A=1`) each cycle. `D=0` indicates to *decrement* when counting or *right-shift* when shifting and `D=1` indicates the opposites. The circuit always shifts in a `0` bit.

(A) Draw out the circuit below. You can freely use registers, constants, 2:1 MUXes, and the following logic blocks. Make sure you label the corresponding selector bits for ports of routing elements. [8 pt]





(B) Now assume that we instantiate our circuit with $n = 3$. In the Verilog testbench below, fill in the blanks to indicate how the output of our sequential circuit updates. [3 pt]

```
initial begin
                    D <= 1; A <= 0; E <= 1;   // Q:  000
   @(posedge clk);          A <= 1;           // Q: _001_ ( + 1 )
   @(posedge clk);                   E <= 0;  // Q: _010_ ( <<1 )
   @(posedge clk);  D <= 0; A <= 0; E <= 1;   // Q: _010_ (  Ø  )
   @(posedge clk);          A <= 1;           // Q: _001_ ( - 1 )
   @(posedge clk);          A <= 0;           // Q: _000_ ( >>1 )
   @(posedge clk);  $stop();                  // Q: _111_ ( - 1 )
end
```

Question 3: Cryptography [9 pts]

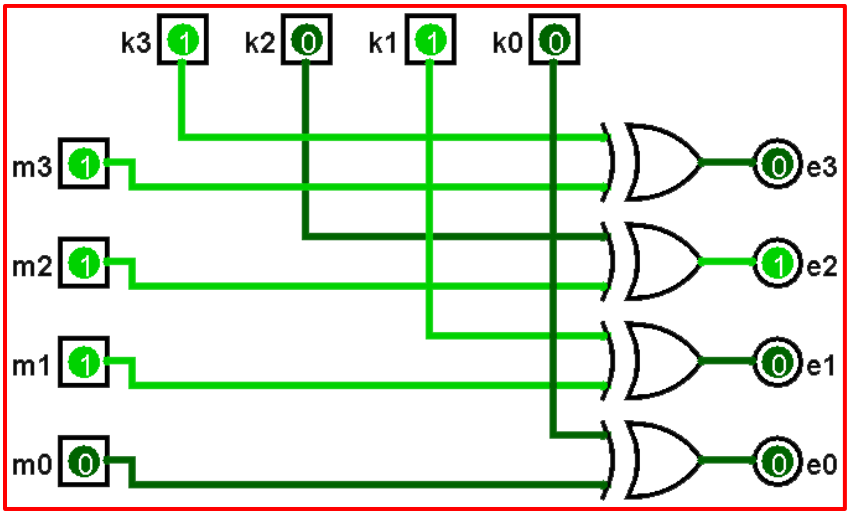
In cryptography, we wish to encode a message to apparent nonsense in a *reversible* manner so that the intended recipient can decode it and recover the original message. We can build a simple encoder using logic gates and a special "key"!

Example: With the message `0b1110`, and key `0b1010`, we get the encrypted message `0b0100`, from which we can recover the original message using the same key.

(A) (Circle one) Which type of gate will allow us to reversibly encrypt and decrypt? [1 pt]

AND NAND NOR OR XNOR (XOR)

(B) Below, implement a 4-bit encryption circuit that computes the encrypted message `e` from the original message `m` and key `k`. You may only use a *single type of 2-input logic gate*. [3 pt]



(C) Assume $t_{\mathsf{NOT}} = 10$ ns, $t_{\mathsf{AND}} = t_{\mathsf{OR}} = 25$ ns, and $t_{\mathsf{XOR}} = 40$ ns. Now we want to implement a decryption circuit that reverses the encryption. How much slower, if at all, would this decryption circuit be than the encryption circuit from Part B? [2 pt]

The decryption circuit is identical to the encryption circuit! __0__ ns

(D) Outline (in writing) a possible solution to handling messages of any length (*e.g.*, ones that are much longer than the key), assuming that we only have one instance of the encryption circuit (*i.e.*, we can't spawn extra circuitry on the fly). [3 pt]

One possibility would be to use the encryption circuit along with shift registers, encrypting n-bits at a time. You would need to shift n bits of the message, the key, and the result every clock cycle. The result would need to either be sent serially or written into a large enough buffer.