University of Washington – Computer Science & Engineering

Winter 2023 Instructor: Chris Thachuk 2023-03-07

CSE 369 QUIZ 3

Name: _____ Perry_Perfect _____

UWNetID: _____perryp_____

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 50 minutes to complete this quiz.

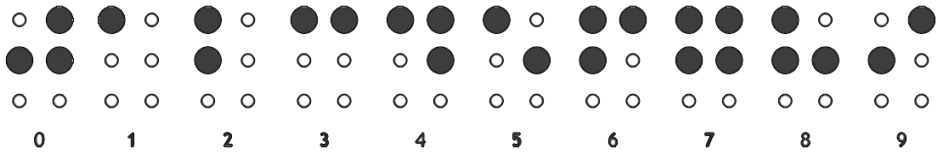
Advice

- 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) Building Blocks	12	12
(2) Shift Registers	13	13
(3) Fredkin gates (Routing elements)	7	7
Total:	**32**	**32**

Question 1: Building Blocks [12 pts]

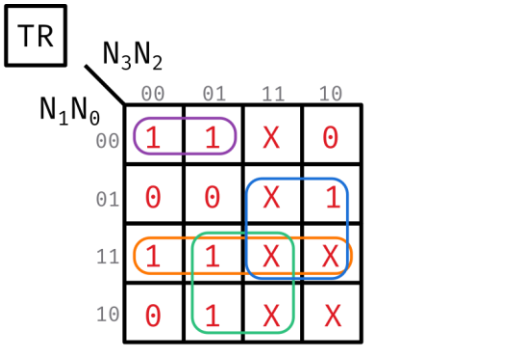
We want to build a **binary-to-braille decoder circuit**. Braille is represented by 6 dots, as shown below. Circles that are black/white represent LEDs that are on (1)/off (0), respectively. The binary-coded digit is given in bits N_3–N_0.



Implement the *simplest* two-level logic for an *enabled* circuit below for the **top-right dot (TR)**. You may use any 1- to 3-input logic gates discussed in the class.

- The dot should always be off (0) if `Enable` = 0
- Your truth table *will be graded*.

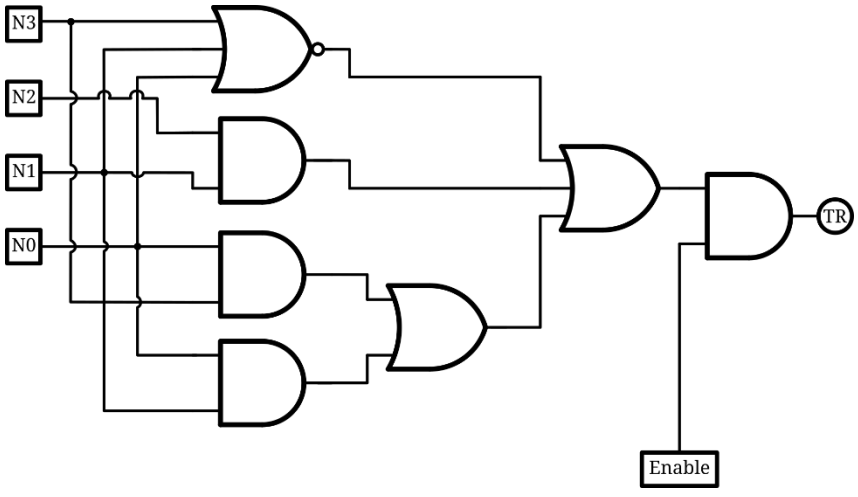
N_3	N_2	N_1	N_0	TR
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



$$TR = \overline{N_3}\,\overline{N_1}\,\overline{N_0} + N_2N_1 + N_1N_0 + N_3N_0$$

$$NOR(N_3,N_1,N_0)+AND(N_2,N_1)+AND(N_1,N_0)+AND(N_3,N_0)$$

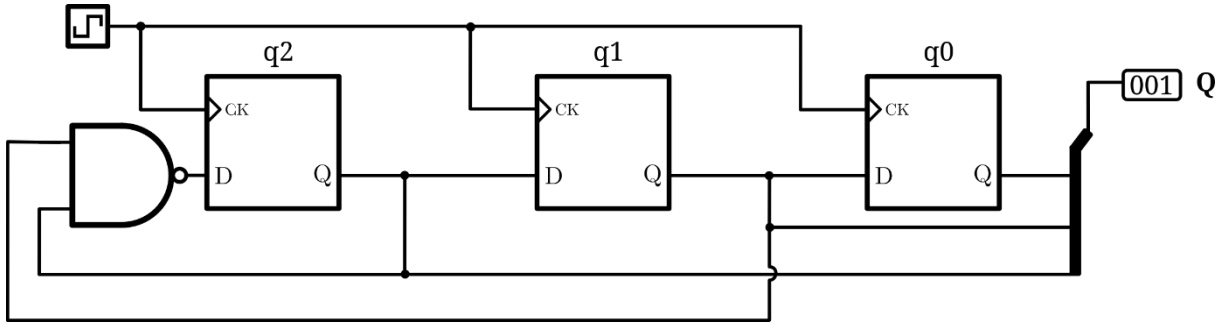
Simplifications of above also accepted.



Question 2: Shift Registers [13 pts]

We are designing a 3-bit LFSR as a pseudo-random number generator, but we only have a 2-input NAND gate available. Note that this LFSR shifts bit RIGHT.

(A) Assuming we start in state **001**, and bits **q2** and **q1** are the inputs to the NAND gate (as wired below) what sequence of values for **Q** will be generated prior to a value being repeated? What is the length of the cycle we end up in? [5 pt]



Values of **Q**:	001, _100, 110, 011, 101_____
Cycle length:	__3__

(B) In a redesign of (A) which bits (*e.g.,* **q2**, **q1**, **q0**) would you wire as inputs to the NAND gate to yield the longest cycle possible if you do not have control over the initial value of **Q**? [4 pt]

Using q1 and q0 yields the longest cycle when the shift register can begin with any value (*i.e.,* when you cannot depend on a particular initial value).

Inputs to 2-input NAND gate: __q1___ , __q0___

(C) Complete the Verilog code below to realize an *enabled* version of a 3-bit LFSR that shifts RIGHT and yields the longest cycle when you have a single 2-input NAND gate and control over the initial value (*i.e.,* it can be reset to value 000). [4 pt]

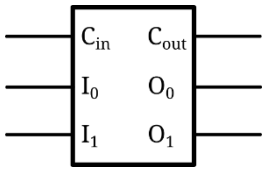
```
module LFSR (Q, enable, reset, clk);
  input  logic enable, reset, clk;
  output logic [2:0] Q;

  always_ff @(posedge clk)
    if ( reset )
      Q <= 3'b000;
    else if ( enable )
      Q <= { _~(Q[0]&Q[0])_, Q[2], ____Q[1]____ };

endmodule
```

Beginning from value 000, using Q[0] for both inputs to the NAND gate yields a cycle of length 6.

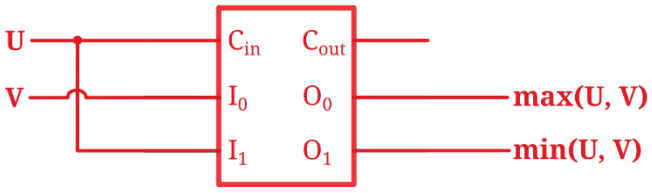
Question 3: Fredkin gates [7 pts]

A **Fredkin gate** is a routing element that has application in *reversible* computing. It is also referred to as a "controlled swap"; see its truth table below to understand why (*hint*: check how the output values relate to the inputs based on the "control bit" C_{in}).

C_{in}	I_1	I_0	C_{out}	O_1	O_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1



The maximum and minimum of two elements, **U** and **V**, can be determined with a single Fredkin gate.



Using only Fredkin gates, draw (and label with inputs outputs) a minimal size circuit that takes three input bits, A, B, and C, and outputs them in sorted, ascending order to outputs X, Y and Z. (*e.g.,* input: 0 1 0, output: 0 0 1).