University of Western Ontario, Computer Science Department
CS3350B, Computer Organization

Assignment 2 **Due: February 16, 2023**

General Instructions: This assignment consists of 4 pages, 5 exercises, and is marked out of 80. For any question involving calculations you must provide your workings. You may collaborate with other students in the class in the sense of general strategies to solve the problems. But each assignment and the answers within are to be solely individual work and completed independently. Any plagiarism found will be taken seriously and may result in a mark of 0 on this assignment, removal from the course, or more serious consequences.

Submission Instructions: The answers to this assignment are to be submitted to Gradescope. Ideally, the answers are to be typed. At the very least, clearly *scanned* copies (no photographs) of hand-written work. If the person correcting your assignment is unable to easily read or interpret your answer then it may be marked as incorrect without the possibility of remarking.

Useful Facts:

Logism is a good tool for drawing circuits.

draw.io is a good tool for drawing any kind of diagram, including circuits.

You may also consider using OneNote, Photoshop, etc. to draw circuits

$$1\ GHz = 1 \times 10^9\ Hz$$

$$1\ byte = 8\ bits$$

$$1\ Kbyte\ (KB) = 1024\ bytes$$

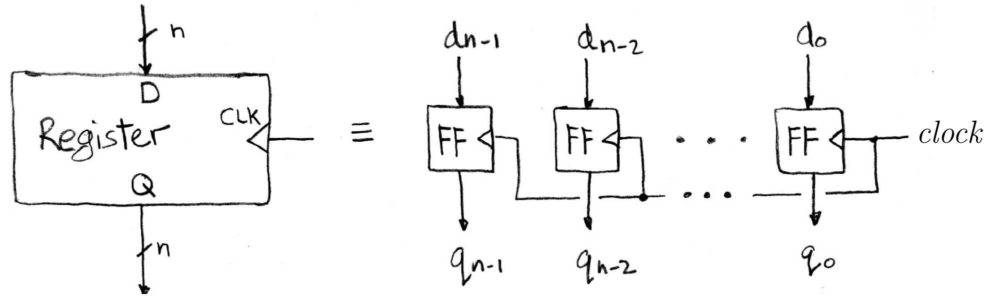
Recall that $\overline{XY} \equiv \overline{X}\,\overline{Y} \equiv \overline{X} \cdot \overline{Y}$

Exercise 1. [6 marks] Re-write the following Boolean expression using only NAND operations. Show your step-by-step process. You do not need to simplify.

$$\overline{(\overline{p} + q) \cdot r}$$

Exercise 2. [10 Marks] Prove that NOR is functionally complete. You should give formulas as well as truth tables to support those formulas. You can use the symbol ↓ for NOR.

Exercise 3. [10 marks] Draw a 4-bit Serial In, Serial Out register using SR flip-flops. For example, the below diagram represents a Parallel In, Parallel Out n-bit register using D flip-flops.



Exercise 4. [20 marks] In this exercise you are to construct an ALU with the following specification:

- The ALU takes 3 different 32-bit inputs: A, B, C; and has one 32-bit output: D.
- The ALU supports 8 different operations:
  - A + B (addition)
  - A - B (subtraction)
  - MAX(A,B,C)
  - MIN(A,B,C)
  - A & B (bitwise AND)
  - B & C (bitwise AND)
  - A | B (bitwise AOR)
  - B | C (bitwise OR)

To construct this ALU you will follow a modular design. Assume you have an unlimited number of the following gates and combinational circuits at your disposal, but no others:

- 2-arity AND gate,
- 2-arity OR gate,
- NOT gate,
- 2-way MUX,
- bit-wise AND circuit: it takes two 32-bit inputs and has a 32-bit output which is the bit-wise AND of its inputs.
- bit-wise OR circuit: it takes two 32-bit inputs and has a 32-bit output which is the bit-wise OR of its inputs.
- 32-bit ADDER: it takes two 32-bit inputs, a 1-bit control signal, and has a 32-bit output. If the control signal is 0, the circuit outputs the sum of its inputs. If the control signal is 1, the circuit outputs the difference of its inputs. The inputs and outputs are assumed to be 32-bit two's complement numbers.
- COMAPRATOR: it takes two 32-bit inputs, a 1-bit control signal, and has a 32-bit output. If the control signal is 0, it outputs the minimum of its two inputs. If the control signal is 1, it outputs the maximum of its two inputs. The inputs and outputs are assumed to be 32-bit two's complement numbers.

(a) Give the *specification* of the ALU by determining the number of control signals needed and then:

(*i*) present a table specifying the control signal values and the operation the ALU performs given those control signal values;

(*ii*) draw a schematic diagram (Lec. 7, pg. 5) for your ALU; include bit-widths and labels.

(b) Using a modular design and the previously stated circuits and gates, draw a circuit which fulfills the specification of your ALU given in part **(a)**. (*Hint:* you can have multiple input wires with the same label; you can have control signal "stubs"; see Lecture 7, page 25.)

Exercise 5. [10 + 8 + 8 + 8 = 34 marks] In this exercise we will explore designing an integrated circuit for use within a microwave. Not the whole thing (that's too complex), but just a small piece of it.

Consider that the microwave already has a circuit which controls the timer and the magnetron (the thing that produces the microwaves). Your goal is to design a simple circuit which controls the turntable. The turntable is powered by a gear box which takes two input bits:

- R. If $R \equiv 1$, the gear box rotates. If $R \equiv 0$ the gear box does not rotate.
- C. If $C \equiv 1$ and $R \equiv 1$, the gearbox turns *clockwise*. If $C \equiv 0$ and $R \equiv 1$, the gearbox turns *counterclockwise*.

The manufacturer of this microwave requests a synchronous circuit which will control the gearbox to **alternate between clockwise and counterclockwise rotation each time the microwave is turned on**. The circuit to be designed will receive a single input bit I from the circuit controlling the magnetron.

- When $I \equiv 1$, the turntable should rotate.
- When $I \equiv 0$, the turntable should not rotate.

To guide your design process, complete parts (a) to (d) below.

(a) Draw a Finite State Machine which describes this state circuit. Since this circuit has 4 possible combinations of outputs, it should have 4 states. Be sure to include the inputs and outputs in the FSM. Order the outputs as RC. Hence, a transition in the FSM will be labelled as I/RC. The change in direction should be triggered by the microwave *turning on*.

(b) Create a truth table to describe the inputs, outputs, and transitions of your FSM. That is, create a truth table describing the combinational logic of your eventual state circuit. Since there are 4 states, you must have 2 bits of "input" to encode the current state, and 2 bits of "output" to encode the next state. (*Hint:* your table should have 8 rows and 7 columns.)

(c) Give the DNF for each output of the combinational part of your state circuit (there are 4 of them; recall that the "next state" is part of the combinational logic output). Then, simplify each DNF into a reduced sum of products. For this particular question, you do not need to specify the Boolean law being applied at each step of the simplification.

(d) Draw a circuit which implements the combinational logic of your turntable controller circuit. That is, draw a circuit implementing your 4 simplified formulas from part **(c)**. Use gates with *arity* at most 2. For clarity (and to avoid crossing too many lines), you may draw 4 independent circuits, one for each output bit.