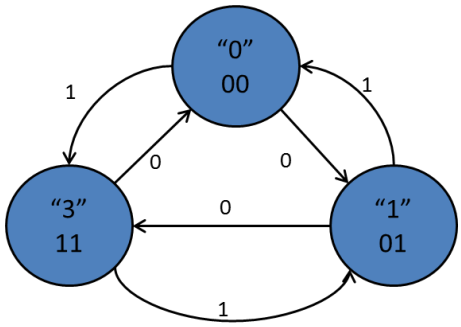
COM1006/COM1090 Devices and Networks (Autumn)
Tutorial Sheet #7: Sequential Circuits

1. Design and implement a **synchronous** custom up-and-down counter, similar to the one seen in class. The only difference is that instead of skipping state 1 in the sequence, your counter should skip state 2.

   If A=0, the counter goes through the numbers 0, 1, 3, and then starts again at 0. If A=1, the counter goes through this sequence in reverse.



   We use two flip-flops with outputs Q_1 and Q_0 to represent the counter's state Q_1Q_0, which is either "0"=00, "1"=01, or "3"=11. Choose the type of flip-flop you want to use in the remainder of this exercise: RS, D, or JK flip-flops. Then follow the design steps as shown on slides 24-29 from Lecture 8.

   a) **Step 1: truth table for state transitions**
   Complete the following truth table, showing how the next states for Q_1^+, Q_0^+ result from the current states and the input A. Note that state Q_1Q_0=10 is never reached, so we **don't care** about the values of Q_1^+,Q_0^+ in this case.

A	Q_1	Q_0	Q_1^+	Q_0^+				
0	0	0						
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						

   b) **Step 2: determine inputs for flip-flops**
   Add inputs for your flip-flops to the above truth table: depending on your choice, this is **either** J_1, K_1, J_0, K_0 **or** R_1, S_1, R_0, S_0 **or** D_1, D_0 (for D flip-flops you only need two columns). Find out how to set these inputs to store the desired next state in your flip-flops.

   c) **Step 3: design circuits for flip-flop inputs**
   Derive Boolean expressions for the above flip-flop inputs from your truth table. Simplify these expressions using Boolean algebra, logical reasoning, or the Karnaugh maps below.

Q_0 \ AQ_1	00	01	11	10
00				
01				

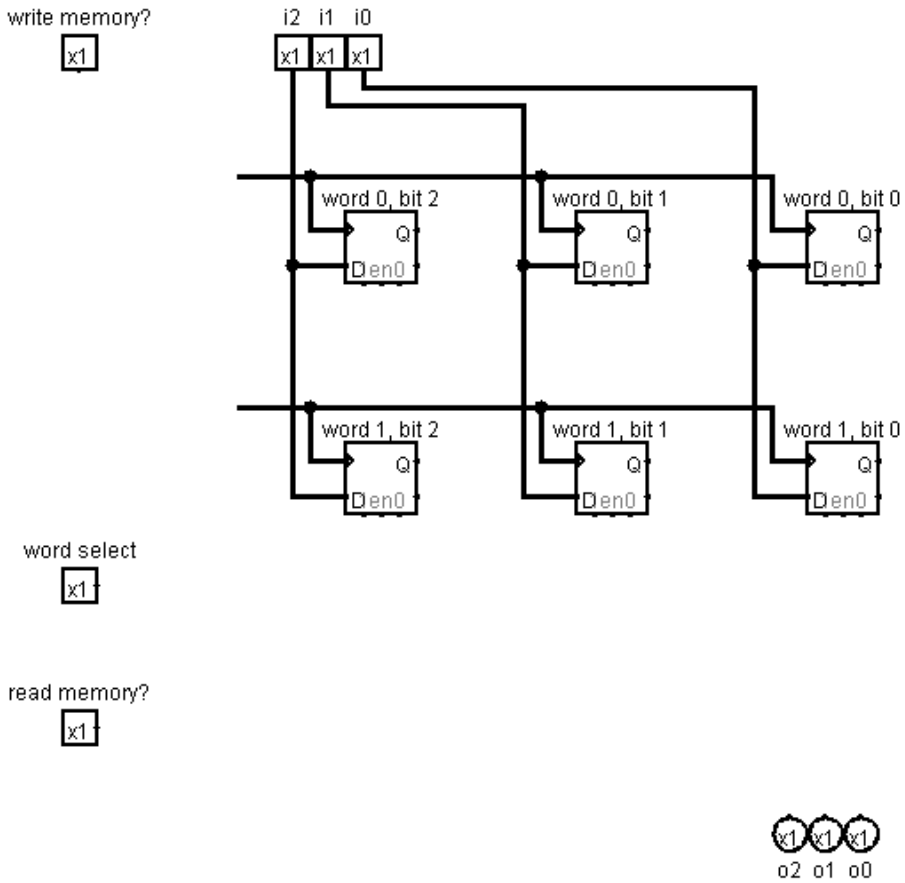
Q_0 \ AQ_1	00	01	11	10
00				
01				

Q_0 \ AQ_1	00	01	11	10
00				
01				

Q_0 \ AQ_1	00	01	11	10
00				
01				

   d) **Step 4: assemble circuits and flip-flops**
   Download one of the following stubs, depending on your choice of flip-flops:
   https://staffwww.dcs.shef.ac.uk/people/D.Sudholt/campus_only/misc/tutorial-counter-withRS-stub.circ
   https://staffwww.dcs.shef.ac.uk/people/D.Sudholt/campus_only/misc/tutorial-counter-withJK-stub.circ
   https://staffwww.dcs.shef.ac.uk/people/D.Sudholt/campus_only/misc/tutorial-counter-withD-stub.circ
   Add combinatorial logic to implement the flip-flop input signals as derived in Step 3.
   Select Simulate → Ticks Enabled to see your counter in action.

2. Let us now design a simple memory with two 3-bit words, implemented with D flip-flops as shown in the following stub, available from:
https://staffwww.dcs.shef.ac.uk/people/D.Sudholt/campus_only/misc/memory-stub.circ



We have inputs $i_2 i_1 i_0$ (coming from a bus) and outputs $o_2 o_1 o_0$ (connecting to a bus). The "word select" input determines whether we are writing to/reading from word 0 or word 1. If "write memory" is 1, the input shall be stored in the selected word. If "read memory" is 1, the selected word shall be output; otherwise outputs shall be floating. We will use demultiplexers and multiplexers as switches to address the right word.

a) Complete the wiring to the clock inputs of the flip-flops so that all flip-flops of a word trigger if and only if "write memory" is 1, and the word is being addressed. All flip-flops are triggered on high level. Use a 1-line to 2-line demultiplexer for activating the line for the selected word, based on the value of "word select".

b) Complete the wiring from the flip-flops' outputs to the output pins $o_2 o_1 o_0$. Use multiplexers for each bit position to select which line (word 0 or word 1) is forwarded to the respective output, based on the configuration of "word select".

c) Use tri-state logic to make sure that output pins are floating whenever "read memory" is 0. Either use tri-state buffers, or note that multiplexers have built-in tri-state logic and an "enable" switch.