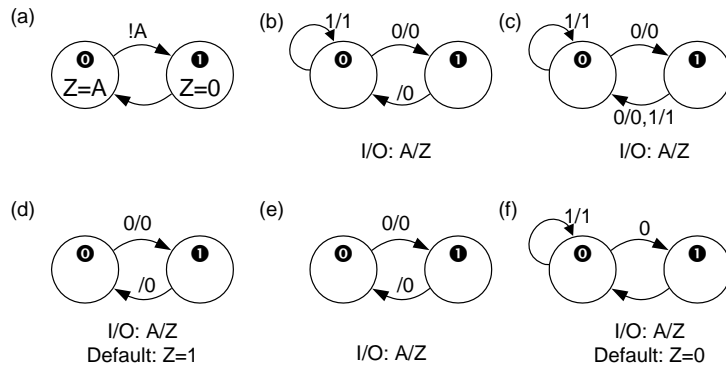# E2.11/ISE2.22 – Digital Electronics II

Problem Sheet 4

(Question ratings: A=Easy, …, E=Hard. All students should do questions rated A, B or C as a minimum)

1B. Say which of the following state diagrams denote the same state machine as version (a). Where an arrow is marked 0/1, for example, it means when A=0, the output Z will be 1 and the transition will be taken at the next CLOCK rising edge.


(a) !A Z=A Z=0

(b) 1/1 0/0 /0 I/O: A/Z

(c) 1/1 0/0 0/0,1/1 I/O: A/Z

(d) 0/0 /0 I/O: A/Z Default: Z=1

(e) 0/0 /0 I/O: A/Z

(f) 1/1 0 I/O: A/Z Default: Z=0

2C. The state diagram and input waveforms of a state machine are shown below. All input and state transitions occur shortly after the clock rising edge. Complete the timing diagram by indicating the value of the state during each clock cycle and by drawing the waveform of X. The initial state is 0 as shown.


Clock ↑ A B State 0

11/1 0 00 1 10 2 01 00

I/O Signals: A,B/X Default: X=0
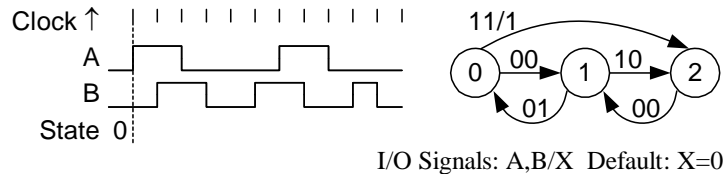
3B. A synchronous state machine has its state represented by the 2-bit number S1:0 and has a single input signal DIR. The current state is stored in a D-type register whose input NS1:0 is defined by: $NS1 = S0 \oplus DIR$ and $NS0 = \overline{S1 \oplus DIR}$. Draw the state diagram for the state machine.

4C. Draw the state diagram for a state machine whose output goes high when the input is high for four or more clock cycles. As shown in the timing diagram, the output should go high during the fourth clock cycle and remain high so long as the input does. Input and state transitions occur shortly after the clock rising edge.


Clock ↑ IN OUT
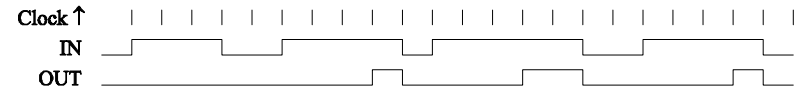
5D. Draw the state diagram for a state machine whose output goes high during the clock cycle following the reception of the input sequence 1011010. The trigger sequences can overlap as in the example below. Indicate the sequence of states followed by your design.for the input sequence given below.


Clock ↑ IN 1 0 1 1 0 1 0 1 0 1 1 0 1 0 OUT
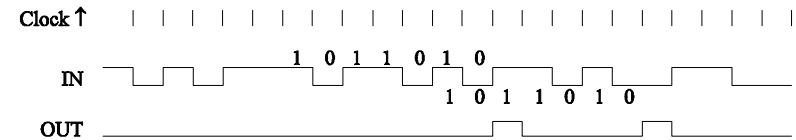
6C. A counter is required that follows the sequence 1, 2, 3, 1, 2, 3, …. Design a state machine to follow this sequence using D-type flipflops and as few gates as possible. You should ensure that the counter will reach the desired sequence regardless of its initial state.

7B. State the circumstances under which an input to a state machine should be passed through a register before going to the logic that generates the next-state bits.

State the circumstances under which an output from a state machine should be passed through a register before being used elsewhere in a circuit.

8C. Two possible numberings for a state machine are shown below. Explain why it is essential for the input to be synchronised with the clock in one case but not in the other.


0 0 1 1 2 3

1 0 0 1 2 3

9C. Show that for one of the state machines shown below it is possible to renumber the states to avoid output glitches but that this is not possible for the other one. Assume the input is synchronized with the clock.
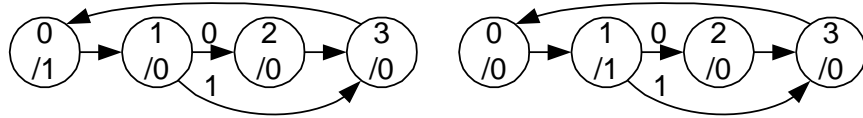

10C. Construct the state diagram for a state machine that emits a single pulse on each rising edge of its input and a double pulse on each falling edge as shown below. Each output pulse should last exactly one clock cycle. Assume that the input signal has been synchronized with the clock rising edge. How does your design react to an input signal that goes low for less than four clock cycles?
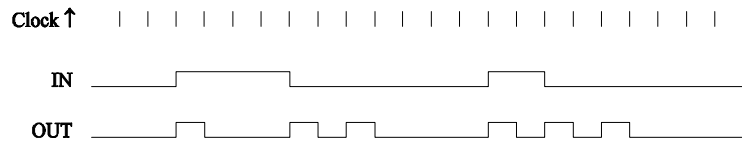
11C. In the state machine illustrated below, the contents of the logic block are defined by:
$NS1 = S1 \oplus S0$, $NS0 = PIN + S1 + \overline{S0}$, $NOUT = \overline{PIN} \cdot S0 + S1 \cdot S0$ which gives the state diagram shown. Transitions of the input signal IN occur on the *falling* edge of the clock. Complete the timing diagram by indicating the sequence of states and the signals PIN, NOUT and OUT.

I/O Signals: PIN/POUT
Default: NOUT=0
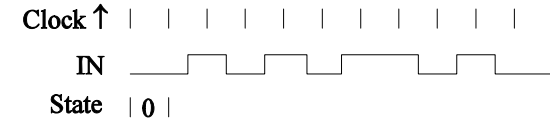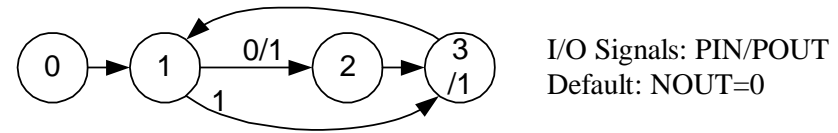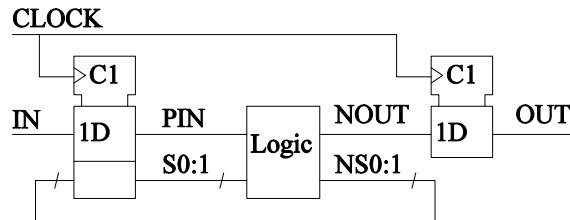
12D. In the notes (page 3.35) the "noise pulse eliminator" is designed as a Moore machine and introduces a two-cycle delay in the output. Show that if it is designed as a Mealy machine it will only require three states and will introduce only one cycle delay as shown (all transitions occur on the rising clock edge):
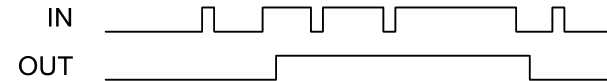
Design the state machine and give Boolean expressions for the outputs of the logic block..

13E. Data is recorded onto floppy disks in a Modified form of Frequency Modulation known as MFM in which each bit of data is recorded as a pair of bits on the disk.. A logical 1 is always recorded as 01 whereas a logical 0 is recorded as either 10 or 00 according to whether the previous bit was 0 or 1. This recording scheme ensures that successive 1's on the disk are separated by between one and three 0's. The timing diagram shows the input (DATA) and the recorded bit-pairs (DISK) for the sequence 101001110001.

Design the state diagram of a state machine which converts an input stream of data bits into the bit-pairs that must be recorded onto the disk as shown in the timing diagram below. Each DATA bit lasts for two clock cycles and all state and input transitions occur shortly after the clock's rising edge. If possible, your design should function correctly regardless of its initial state.
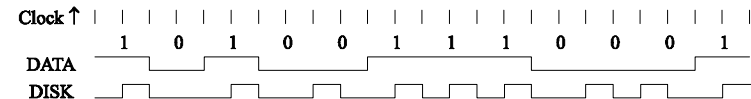
E2.11/ISE2.22 – Digital Electronics II

Solution Sheet 4

(Question ratings: A=Easy, …, E=Hard. All students should do questions rated A, B or C as a minimum)

1B. In comparing state diagrams, you should first check the transitions and then check that the outputs are the same in each state. It can be seen that the transitions are the same for all the versions. However the outputs are incorrect in version (c) and version (e).

When output are marked on arrows they refer to the state from which the arrows originate. In version (c) therefore, state 1 has an output Z=A instead of Z=0 as it should be.

In version (e), the output is not specified for the case A=1 in state 0. It must either be specified by default as in version (d) or else explicitly as in version (b).

2C. You should first determine the state sequence. The transitions depend on the value of A and B immediately *before* the Clock↑ edge. A common mistake is to use the values *after* the edge.

Note that X is only ever high in state 0 and then only if A and B are high. A common mistake is to make X high in state 2 rather than state 0: remember that outputs on transition arrows refer to the preceding state.
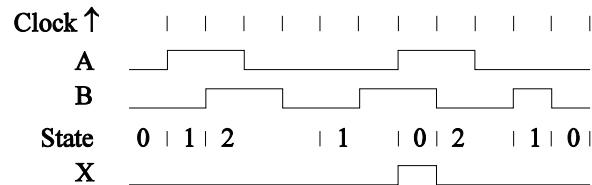

3B. This represents a 2-bit bidirectional counter whose counting sequence has only one bit changing at a time. This unit-distance property means that you can decode the outputs without risk of glitches.
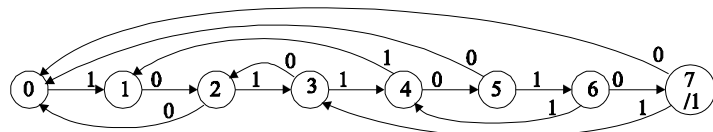
DIR	S1	S0	NS1	NS0
0	0	0	0	1
0	0	1	1	1
0	1	0	0	0
0	1	1	1	0

1	0	0	1	0
1	0	1	0	0
1	1	0	1	1
1	1	1	0	1

4C. Since the output must go high during the fourth clock cycle in response to the value in that cycle, we must have a Mealy machine: a Moore machine would insert too much delay. If IN=1 during the current cycle then we want OUT=1 if the previous three (or more) cycles had IN=1. We therefore need to remember how many of the previous cycles had IN=1: 0, 1, 2 or ≥3. We therefore need four states. Note that an unavoidable glitch possibility exists if IN goes high for three clock cycles; this can only be eliminated reliably by delaying the output for an extra cycle.
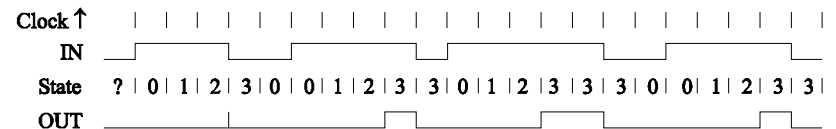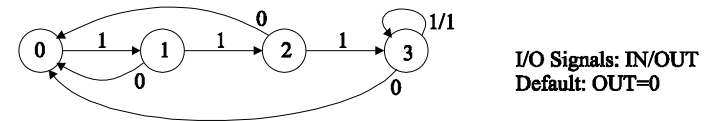
5D. The previous question could be regarded as recognising the sequence 1111. This question is pretty similar but with a different pattern to recognise. There are two significant differences. Firstly, when an input bit does not conform to the required sequence, we cannot always just branch back to state 0; the last few bits of the rejected input sequence may be the first few bits of the correct one. Secondly, the output must go high during the cycle *following* the trigger sequence; this requires an extra state at the end and allows us to use a Moore machine.
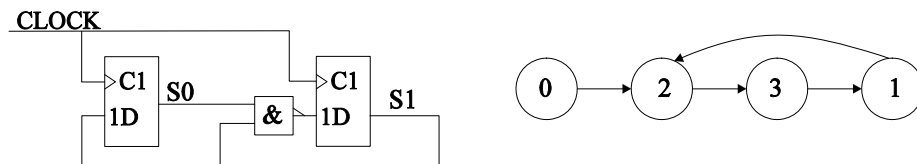

I/O Signals: IN/OUT   Default: OUT=0
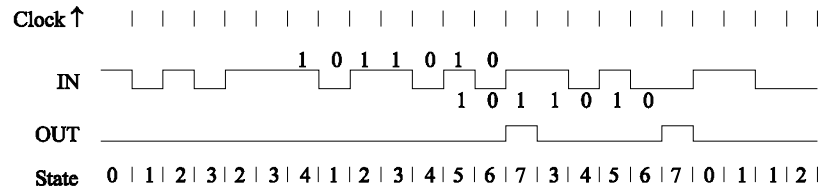
6C. The following table therefore lists both the value of the next state (NS1:0):

S1	S0	NS1	NS0
0	0	X	X
0	1	1	0
1	1	0	1
1	0	1	1

Choosing the "don't care" entries to simplify the expressions we get:

D flipflop inputs: $NS1 = \overline{S1} + \overline{S0}$ $NS0 = S1$

If our flipflops possess set inputs, we don't require the gate. We can avoid state 0 either by forcing S0 high whenever S1 is low or by forcing S1 high whenever S0 is low. The former approach does not work because the set input to S0 will not be released in time when the state machine goes from state 1 to state 2. We can redraw our table with the assumption that S1 is forced high whenever S0 is low; this means that the S1 flipflop's data input can be "don't care" whenever NS0 is equal to 0:

S1	S0	NS1	NS0
0	0	X	X
0	1	X	0
1	1	0	1
1	0	1	1

Choosing the "don't care" entries to simplify the expressions we get:

D flipflop inputs: $NS1 = \overline{S0}$ $NS0 = S1$

7B. Any inputs that are not already synchronized with the clock must be passed through a register before going to the next-state logic. The only exception to this is if the level of a particular input only ever selects between two states whose state numbers differ in a single bit position.

Any output that is prone to glitches must be passed through a register before being connected to a clock, set or reset input of a subsequent circuit either directly or via combinational logic. Another way of putting this is that a glitch-prone output should not be connected to a glitch-sensitive input.

If ROM or RAM is used for the combinational logic then all outputs are glitch-prone. If hazard-free combinational logic is used then glitches are possible if an input depends on two or more inputs/state-bits that can change simultaneously and if any of their 2^n possible combinations would cause the output to change. Another way of looking at this is that since absolute simultaneity is impossible, any inputs to the combinational logic that change together might in fact change in any conceivable sequence; an output is glitch-prone if any of these sequences would cause it to change.

8C. In the rightmost diagram, the input signal selects between states 1 and 2 (01 and 10 in binary); it thus causes both state bits to change. If the input changes just before the clock edge, both inputs to the state register will be changing within the setup-hold window and may end up taking any value. In particular the circuit may end up in state 3 whence it returneth not.

In the leftmost state diagram, the input signal selects between states 0 and 2 (00 and 10 in binary). The LSB is 0 in both cases and so the state machine cannot possibly go to any state other than these two.

9C. We can assume without any loss of generality that the state in which the output is high is numbered 3 (we can always invert one or both of the state bits to make this true). The output will therefore be generated by means of an AND gate. The AND gate output will be prone to glitches if its two inputs (S0 and S1) ever change in opposite directions simultaneously; this is because there will always be a chance that they may briefly be high together. S0 and S1 change simultaneously if and only if we ever have a transition from 1 to 2 (01 to 10) or from 2 to 1 (10 to 01).

In the leftmost state diagram, the three states in which the output is 0 are all mutually connected and so whichever two of them are numbered 1 and 2, there must be a transition joining them and hence a possibility of a glitch.

In the rightmost state diagram we can number the states from left to right 2,3,1,0. There is now no direct link between 1 and 2 and no glitch is possible on the output. In general, any numbering scheme will work in which the second and fourth states from the left have numbers adding up to 3.

Note that even with the leftmost diagram, we can suppress the potential glitch by numbering the states 3,1,2,0 from the left and then inserting a delay in the S1 output of the state register. This delay will ensure that S1 effectively changes later than S0 and that the dangerous transition follows the sequence $1 \rightarrow 0 \rightarrow 2$ without any possibility of passing through state 3.

Note too that we can also make the left diagram work by numbering the states 4, 1, 2, 3 from the left and ensuring that the output is high only in state 4 (out of the possible 8 states). This now requires 3 state bits which is inefficient but none of the transitions between states 1, 2 and 3 can generate a high output since the most significant state bit will remain low.

10C. The basic diagram is shown below; note that we *must* use a Mealy machine in order to get zero delay between IN and OUT. The only two points of difficulty is what to do if the input goes high in the middle of the double pulse sequence and whether we wish to ensure that consecutive pulses are separated by at least one clock cycle.
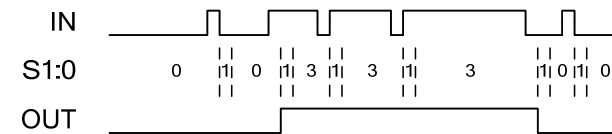

I/O Signals: IN/OUT
Defaults: OUT=0
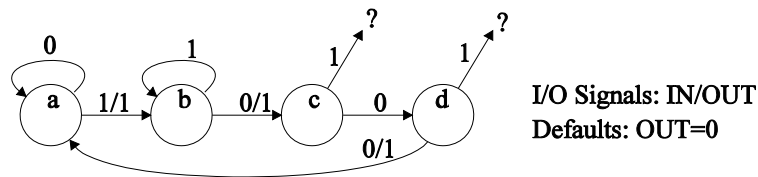
The following diagram ensures that pulses are distinct (by the addition of states e and f) and abandons pulse sequences when another input transition occurs:

I/O Signals: IN/OUT Default: OUT=0
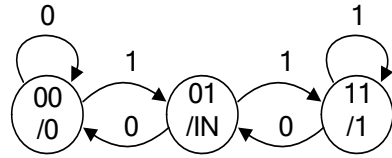
11C.

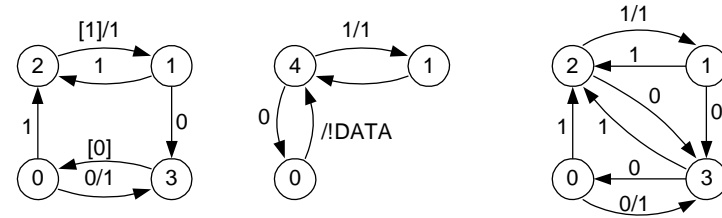I/O Signals: PIN/NOUT
Default: NOUT=0
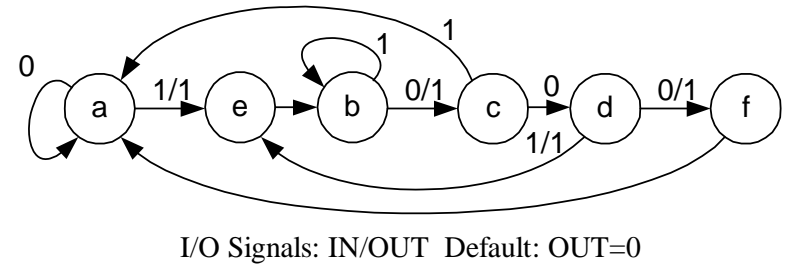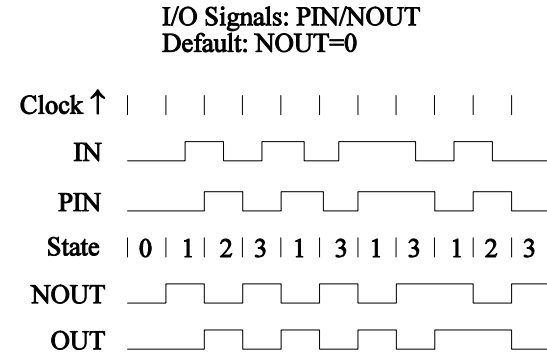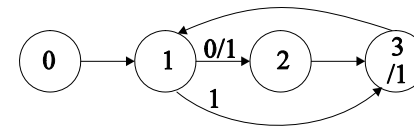
12D. We design the state diagram in the same way as in the lecture notes. Now however, the output in state 1 has to depend on IN (and therefore the circuit must be a Mealy machine); this is illustrated in the first two occurrences of state 1 in the timing diagrm.

We can now draw the state diagram:

0   1

1   1   1
00   01   11
/0   0 /IN   0 /1

State Numbers: S1,S0
Inputs/Outputs: IN/OUT

We now make a Karnaugh map for the three outputs: NS1,NS0/OUT. The last row of the K-map is all "don't care".

S1	S0	IN=0	IN=1
0	0	00/0	01/0
0	1	00/0	11/1
1	1	01/1	11/1
1	0	X**X/X**	XX/X

Choosing the "don't care" entries to simplify the expressions we get:

NS1=S0.IN, NS0=S1+IN, OUT=S1 + S0.IN = S1 + NS1

The X's in the final row of the state table are bold where these expressions are true. Note that the final row always branches to state 1 and so we can never get trapped in state 2.

13E.   The starting point is the leftmost state diagram. Here states 0 and 1 are occupied during the first half of each input bit and 2 and 3 are occupied during the second half. Since the input data should not change between the two halves, we do not really need to check its value in states 2 and 3; I have therefore put the branch conditions in brackets. The top two states correspond to DATA=1 while the lower two correspond to DATA=0.

[1]/1   1/1   1/1
2   1   1   4   1   2   1   1
            0   0
1   0   0   1 1   0
0 0/1 3   0 /!DATA   0   0   3
[0]   0/1

I/O Signals: DATA/DISK; DEFAULT: DISK=0

If we assume that DATA is correct in states 2 and 3, we can merge them to form state 4 as in the second diagram. A better solution is to branch 2→3 or 3→2 if DATA is incorrect as in the third diagram. This will resynchronise the circuit and make if function correctly regardless of its initial state. The figure shows several examples of this.

Clock ↑ | | | | | | | | | | | | | | | | | | | | | | | | |

DATA   1   0   1   0   0   1   1   1   0   0   0   1

DISK

State |1/0| 2 | 1 | 3 | 0 | 2 | 1 | 3 | 0 | 3 | 0 | 2 | 1 | 2 | 1 | 2 | 1 | 3 | 0 | 3 | 0 | 3 | 0 | 2 |

Synchronization | 2 | 1 | 2 | 3 | ... | 3 | 0 | 3 | 2 | ... | 0 | 2 | ...