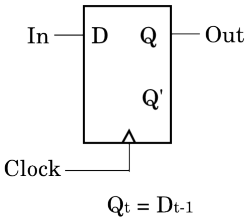
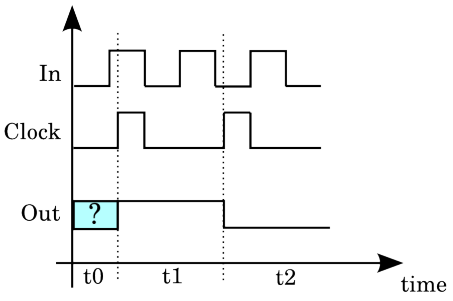
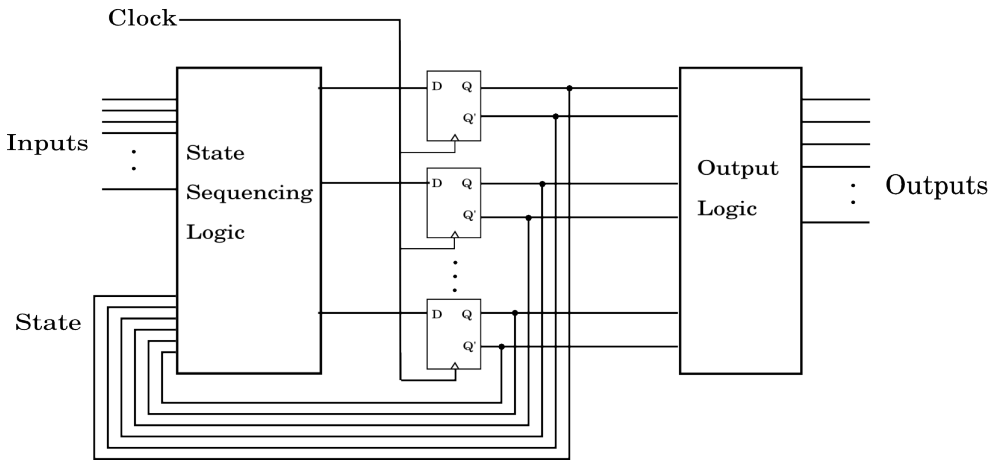
Lecture 8: Synchronous Digital Systems

The distinguishing feature of a synchronous digital system is that the circuit only changes in response to a system clock. For example, consider the edge triggered flip-flop which we discussed earlier in the course. Its key property is that the output changes to the input value only on receit of a clock pulse. Depending on how we design the flip flop, the change may occur on either the rising clock edge or the falling clock edge. In a big circuit, there may be other clock signals as well but all will have to be "synchronized" to the system clock and any other clock signal must have a known relationship in time to the system clock.





A sequential circuit is one that goes through a sequence of states. For example, we could use a sequential circuit to control a traffic light system, or a washing machine programmer. There are many examples of sequential circuits in the real world, and the most complext of them is the central processing unit of a digital computer. Our focus in this lecture and the next, is the design of synchronous sequential digital circuits. We will make use of a generic synchronous sequential system which is shown in the following block diagram:



All flip-flops are connected to the same clock, and therefore all change at the same time. The "state" of the system is defined as the Q outputs of the flip flops, and is stable between the clock pulses. The state is therefore a binary number of the form $Q_1Q_2Q_3..Q_n$. If there are n flip-flops then there are exactly 2^n possible states of the system. The state sequencing logic and output logic blocks are combinational circuits and their outputs are strictly functions of the inputs. Like all combinational logic circuits they suffer from transport delays, and may have spikes on the output in response to changing inputs.

The behaviour of the system is defined by the transition of one state to another which occurs when a clock pulse is applied. As shown above, the output signals at any time depend only on the state of the system (defined by the flip-flop outputs). The next state depends on both the state and the input. We can indicate these in the functional forms:

$Q_s(t_{n+1}) = D_s(t_n)$ The D type flip flop law

$D_s(t_n) = F(I_1(t_n), I_2(t_n), ..., Q_1(t_n), Q_2(t_n), ...)$ State Sequencing Logic

$Q_s(t_{n+1}) = F(I_1(t_n), I_2(t_n), ..., Q_1(t_n), Q_2(t_n), ...)$

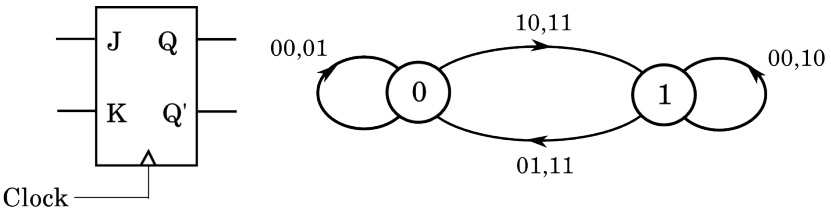
$Out_j(t_n) = G(Q_1(t_n), Q_2(t_n), ...)$ Output Logic

Since the collection of flip-flop outputs is defined as the state of the system, the third equation above is called the state transition equation. The state transition equation completely describes the behaviour of the system for all times if the state of the system is known at an initial time and the inputs are known for both initial and all later times. The output of the system is a direct (combinational) function of its state and does not influence the behaviour of the system, only its outputs. The advantage of the state transition description is that it has a convenient graphical representation: the state transition diagram (also called the finite state machine diagram).

Let us look at the example of a "J-K" Flip-Flop, which is a simple synchronous circuit. The functional behaviour of the J-K flip-flop is shown in the following table:

J	K	Function	Output $Q_s(t_{n+1})$
0	0	No Change	$Q_s(t_n)$
0	1	Reset	0
1	0	Set	1
1	1	Toggle	$Q_s(t_n)$

The symbol and state transition diagram are::

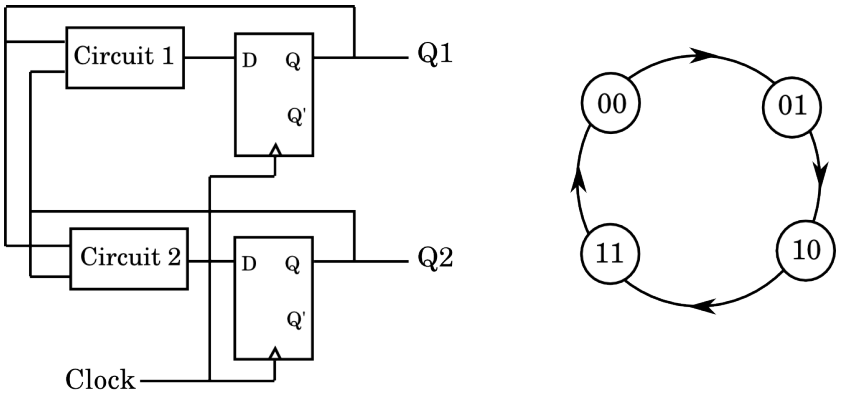


Although there appear to be two output variables (Q, Q'), there is only one state variable Q, which in this case is also one of the outputs. Since there is one state variable, there are two states, which can be conveniently labelled by the value of Q (i.e. binary 0 or binary 1). The labels of the two states appear in the two circles.

The arrows represent the state transitions and for two inputs (JK) there are four different combinations of the input values for each state. In this case, each state transition occurs for two input combinations. For example, if the flip-flop is in state 0, either the inputs JK = 00 (leave the output unchanged) or JK = 01 (reset the output) will leave the circuit state 0. Similarly, either input combination 10 (set the output to 1) or 11 (toggle the output) will change the state to 1.

Synchronous Binary Counters

The binary counter is an example of a simple synchronous digital circuit. It has no inputs and no combinational output logic circuit. At each clock pulse the counter takes up a new state and thus goes through a specific count sequence. We shall design now a binary up-counter with two outputs which goes through the sequence: $00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00 \rightarrow$etc. The block diagram, structure and state transition diagram of a two-bit binary counter (built with two D-type flip-flops) is of the following form:

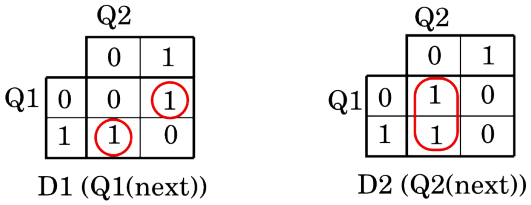


We have to design the state sequencing logic which consists of the two combinational circuits labelled Circuit 1 and Circuit 2. The first step is to construct truth tables for the circuits. The inputs are made up of the current state, and the outputs will define the next state. These outputs form the D inputs to the flip flops.

Inputs (t_n)		Outputs (t_{n+1})	
Q_1	Q_2	D_1	D_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

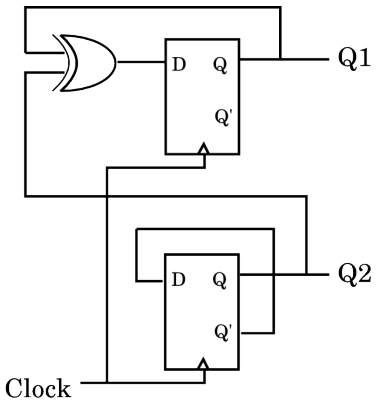
For sequential systems designed with D-type flip-flops, this table is called (appropriately) the transition table. In general, both the flip-flop outputs at time t_n and the circuit inputs at time t_n make up the left column. The outputs are the D inputs to the flip-flops which will become their Q outputs at time t_{n+1}. In the case of the binary up-counter we have no inputs:

From the transition table the Karnaugh maps for the two circuits are constructed and the minimised Boolean expressions are found:



$D_1(t_n) = Q_1(t_{n+1}) = Q_1' \cdot Q_2 + Q_2' \cdot Q_1 = Q_1 \oplus Q_2$ and $D_2(t_n) = Q_1(t_{n+1}) = Q_2'$
where $\oplus$ is the XOR operator.

Since the outputs of Circuit 1 and Circuit 2 are connected to the D inputs of the flip-flops, the above results can be directly applied to the circuit because for a D-type flip-flop $Q(t_{n+1}) = D(t_n)$. Consequently, the finalised circuit is shown below:

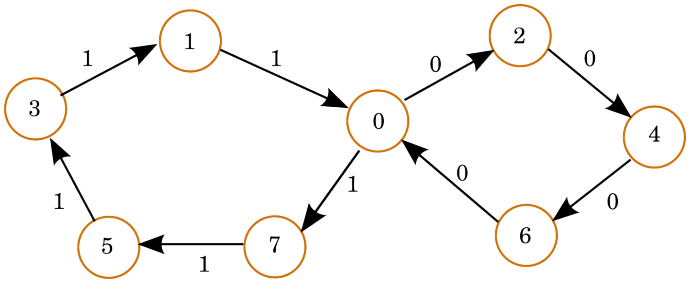


Design of a controlled 3-bit counter with don't care states

We are given the following description of a synchronous sequential circuit to design:

A three-bit binary counter has one control input, C. When $C = 0$ the counter counts up even numbers, i.e. $0 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 0 \rightarrow$ etc, or in binary: $000 \rightarrow 010 \rightarrow 100 \rightarrow 110 \rightarrow 000 \rightarrow$ etc. When $C = 1$ the counter counts down odd numbers: $0 \rightarrow 7 \rightarrow 5 \rightarrow 3 \rightarrow 1 \rightarrow 0 \rightarrow 7$ etc, or in binary: $000 \rightarrow 111 \rightarrow 101 \rightarrow 011 \rightarrow 001 \rightarrow 000 \rightarrow 111$ etc. The counter may start in an undefined state (say, $C = 1$ and the counter output is $2 = 010$). The sequence it goes through at start up is not important, but it must reach the 000 state from any starting state.

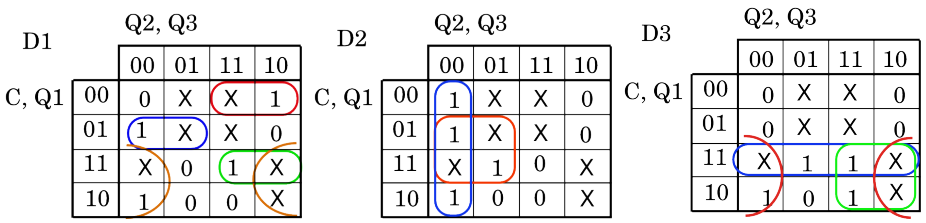
Step 1: The finite state machine diagram is constructed from the wordy specification. For the moment we only include the transitions that are given in the specification.



Step 2: The transition table is produced by tracing through the finite state machine diagram. The don't care outputs X indicate that the state is not part of the counting defined sequence:

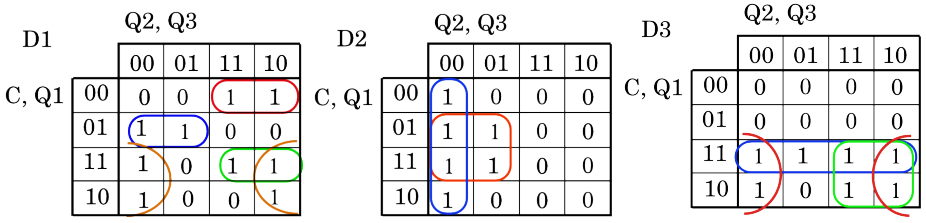
C	Q_1	Q_2	Q_3	D_1	D_2	D_3
0	0	0	0	0	1	0
0	0	0	1	X	X	X
0	0	1	0	1	0	0
0	0	1	1	X	X	X
0	1	0	0	1	1	0
0	1	0	1	X	X	X
0	1	1	0	0	0	0
0	1	1	1	X	X	X
1	0	0	0	1	1	1
1	0	0	1	0	0	0
1	0	1	0	X	X	X
1	0	1	1	0	0	1
1	1	0	0	X	X	X
1	1	0	1	0	1	1
1	1	1	0	X	X	X
1	1	1	1	1	0	1

Step 3: The entries in the transition table are copied into Karnaugh maps and the minimised Boolean equations are found for each D input.



D1 = C'·Q1'·Q2+C'·Q1·Q2'+C·Q3'+C·Q1·Q2
D2 = Q2'·Q3'+Q1·Q2'
D3 = C·Q2+C·Q3'+C·Q1

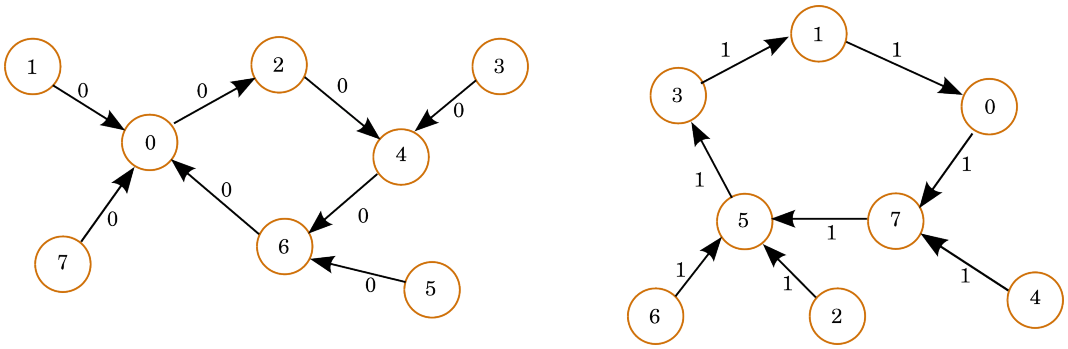
Step 4: The true values for the don't cares can now be entered into the Karnaugh maps. Any don't care within a circle will be a 1 and any don't care outside all the circles will be a 0.



Step 5: The correct transition table, without don't cares, can now be constructed from the three Karnaugh maps. The states are indicated by their equivalent digit.

C	Q_1	Q_2	Q_3	D_1	D_2	D_3	$S(t_n)$	$S(t_{n+1})$
0	0	0	0	0	1	0	0	2
0	0	0	1	0	0	0	1	0
0	0	1	0	1	0	0	2	4
0	0	1	1	1	0	0	3	4
0	1	0	0	1	1	0	4	6
0	1	0	1	1	1	0	5	6
0	1	1	0	0	0	0	6	0
0	1	1	1	0	0	0	7	0
1	0	0	0	1	1	1	0	7
1	0	0	1	0	0	0	1	0
1	0	1	0	1	0	1	2	5
1	0	1	1	0	0	1	3	1
1	1	0	0	1	1	1	4	7
1	1	0	1	0	1	1	5	3
1	1	1	0	1	0	1	6	5
1	1	1	1	1	0	1	7	5

In order to see whether the counter will operate properly from every starting state and input value, the complete finite state machine diagram can be constructed from the table. For clarity it can be divided into two parts, one for the input $C = 0$ and one for $C = 1$.

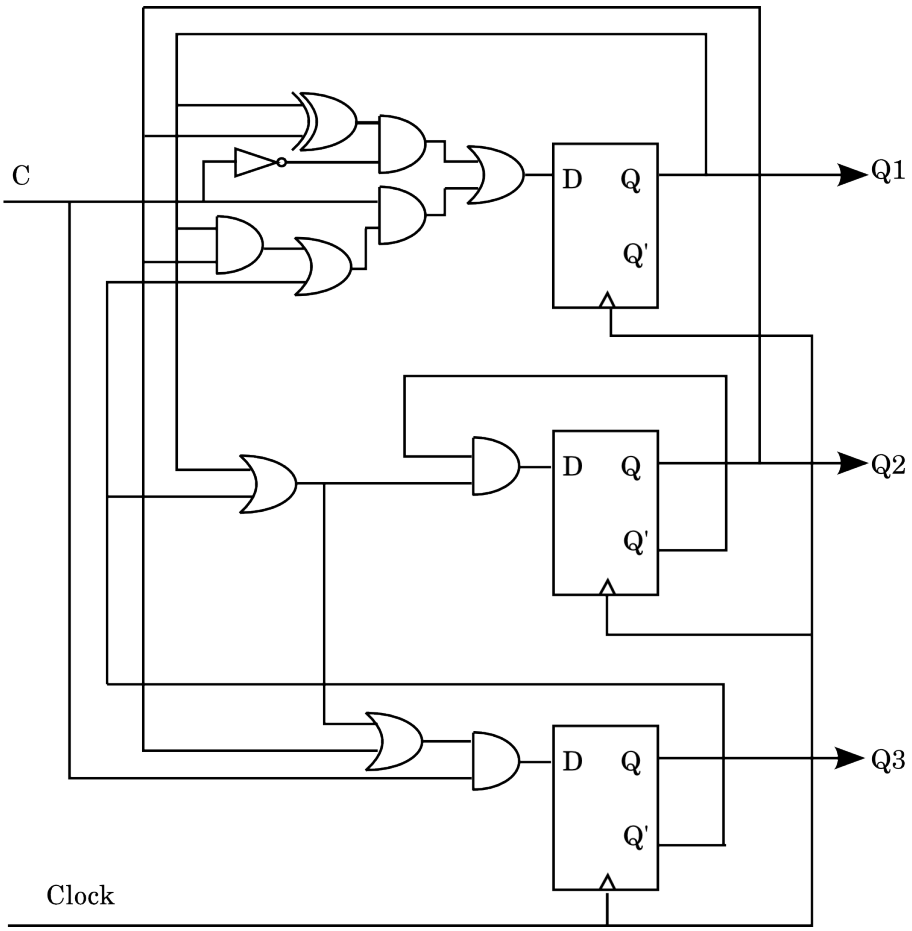


We see that the required specifications are satisfied because for either control input, the counter will eventually reach state 0 regardlesss of it's initial state.

Step 6: the circuit can now be built. Here we will assume that any basic gate (AND, OR, NAND, NOR, XOR, XNOR, Inverter) can be used. From the Karnaugh maps we have:

$$
\begin{aligned}
D_1 &= C' \cdot Q_1' \cdot Q_2 + C' \cdot Q_1 \cdot Q_2' + C \cdot Q_1 \cdot Q_2 + C \cdot Q_3' \\
&= C' \cdot Q_1 \oplus Q_2 + C \cdot (Q_1 \cdot Q_2 + Q_3') \\
D_2 &= Q_2' \cdot Q_3' + Q_1 \cdot Q_2' \\
&= Q_2' \cdot [Q_1 + Q_3'] \\
D_3 &= C \cdot Q_1 + C \cdot Q_3' + C \cdot Q_2 \\
&= C \cdot (Q_1 + Q_2 + Q_3') \\
&= C \cdot ([Q_1 + Q_3'] + Q_2)
\end{aligned}
$$

The square brackets indicate a term which appears in more than one Boolean expression. There are savings to be made by noting these expressions since only one set of gates needs to be used to implement them. The final circuit is:


C
D
Q
Q'
Q1
D
Q
Q'
Q2
D
Q
Q'
Q3
Clock