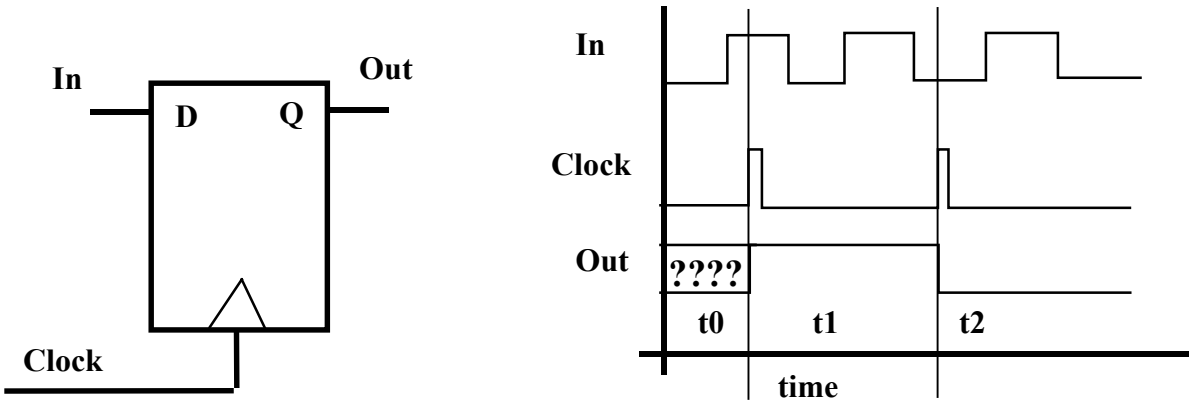
Department of Computing Course DoC 112 / Hardware
Lecture 8: Synchronous Digital Systems

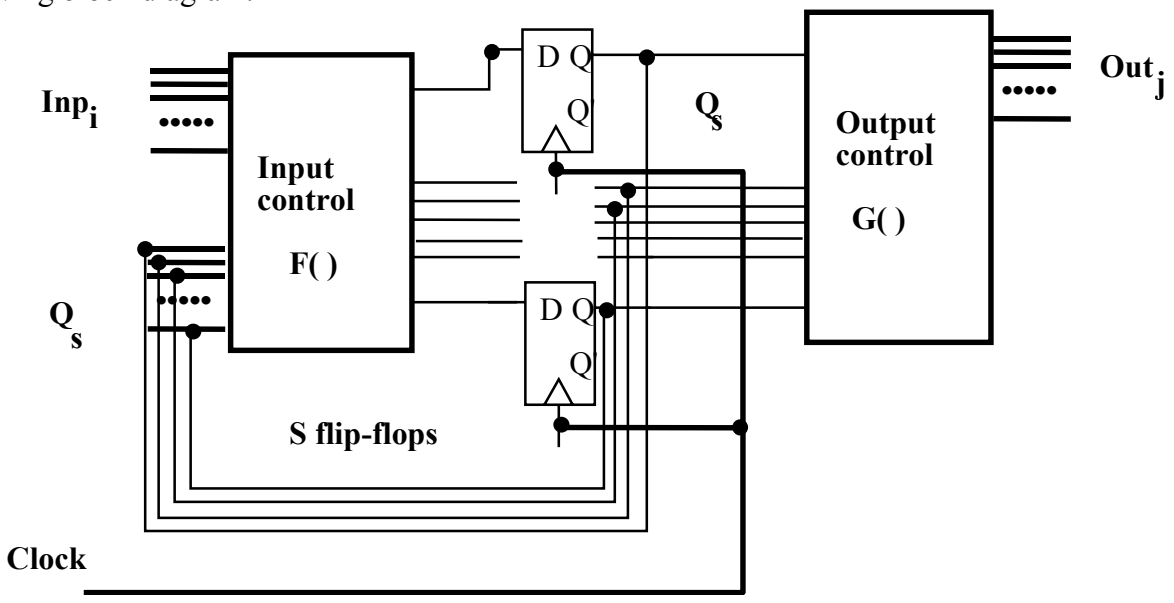
The distinguishing feature of a synchronous digital system is that it has at least one **system clock signal (SCS)**, which of course is also a digital (binary) signal. There may be other clock signals as well but all will have to be "synchronized" to the **SCS** which means that all other clock signals must have a known relationship in time to the SCS. Now we will consider simple systems which use only one clock signal, the **SCS** itself.

We have looked at how **flip-flop**s are constructed and what makes them work. Now, we will ignore their internal characteristics and concentrate on their external (functional) behaviour. Again, the important thing to remember is that it is the **clock** signal which makes them work.

.



If we separate out the combinational elements of a general synchronous circuit and use only **D-type Flip-Flop**s for all its sequential elements, we can define a generic sequential system by the following block diagram:



.

As shown above, all flip-flops change at the same time according to the **Clock**. Thus, we can look at the "state" of the system during two clock pulses when all flip-flop output signals are stable by looking at the outputs of the flip-flops. We have a definition now for the **state** of a synchronous **finite-state machine**, which is another name for a synchronous digital system:

The **state** of a synchronous digital system is defined by the output signals of its flip-flops during the time when the system clock is inactive. If there are k flip-flops then there are exactly 2^k possible states of the system. The two control blocks are combinational networks; therefore, their output are strictly functions of the inputs. The behaviour of the system is defined by the transition of one state to another when a clock pulse is applied. As shown on the diagram, the output signals at any time depend only on the state of the system (collection of **1**s and **0**s of the flip-flop outputs). We can indicate these in the functional forms:

$$\mathbf{Q_s(t_{n+1}) = D_s(t_n)} \qquad \textbf{D-type Flip-Flop law}$$

$$\mathbf{D_s(t_n) = F(\ Inp_1(t_n), Inp_2(t_n), \ldots , Q_1(t_n), Q_2(t_n), \ldots)} \qquad \textbf{Combinational box F()}$$

and

$$\mathbf{Out_j(t_n) = G(Q_1(t_n), Q_2(t_n), \ldots)} \qquad \textbf{Combinational box G()}$$

Since after an active clock pulse occurred the output of each flip-flop becomes equal to its input before the clock pulse, we can also express the first equation above as:

$$\mathbf{Q_s(t_{n+1}) = F(\ Inp_1(t_n), Inp_2(t_n), \ldots , Q_1(t_n), Q_2(t_n), \ldots)}$$

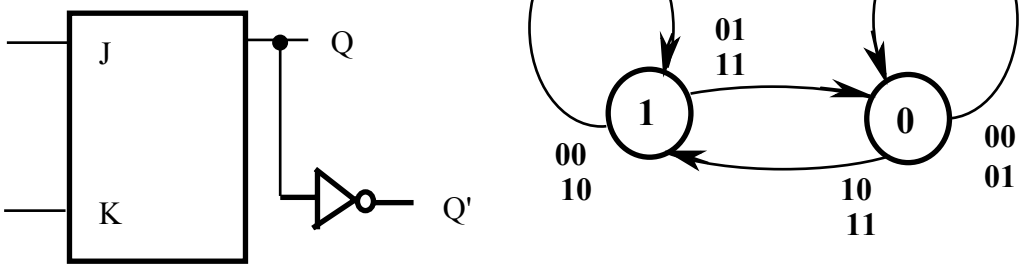
Since the collection of flip-flop outputs is defined as the **state** of the system, the last equation is called the **state transition** equation. The state transition equation completely describes the behaviour of the system for all times if the state of the system is known at an initial time and the 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**. Let us use the example of a **J-K Flip-Flop:**

The functional behaviour of the J-K flip-flop is shown in the following table:

I J	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	Togle	Q_s'(t_n)

The State transition diagram is shown below:

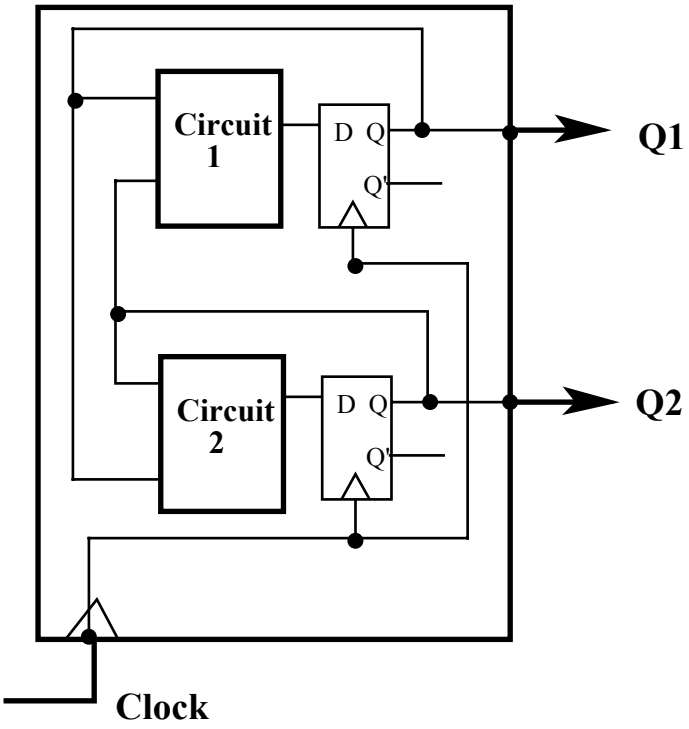


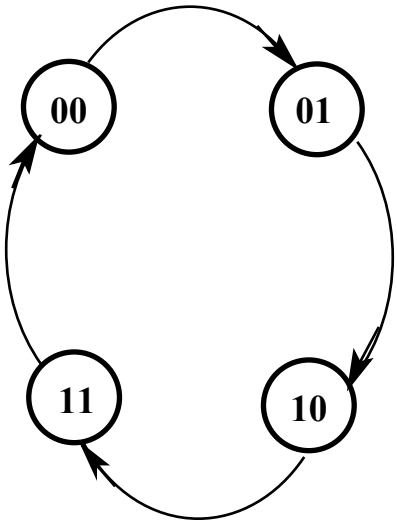
There are (as shown), two output variables (Q, Q'); however, there is only one state variable (Q), which in this case is also equal to one of the outputs. Since there is one state variable, there are two states which we can label as **State 1** and **State 2**, but more conveniently can be labelled by 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 should be four different combinations of the input values for each state. In this case, each transition occurs for two such input combinations. For example, if the flip-flop is in the **0** state, either the inputs JK = **00** (leave the output unchanged) or JK = **01** (reset the output) will leave the output state in state **0**. Similarly, either input combination **10** (set the output to **1**) or **11** (toggle the output) will change the output to become logic **1**.

Synchronous Binary Counters

The simplest synchronous digital system is the **binary counter** because it has no inputs and no combinational output block G ( ). At each clock signal 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 -> 01 -> 10 -> 11 -> 00 ->** etc. The block diagram, structure and state transition diagram of a two-bit binary counter (built with two D-type flip-flops) is shown below:





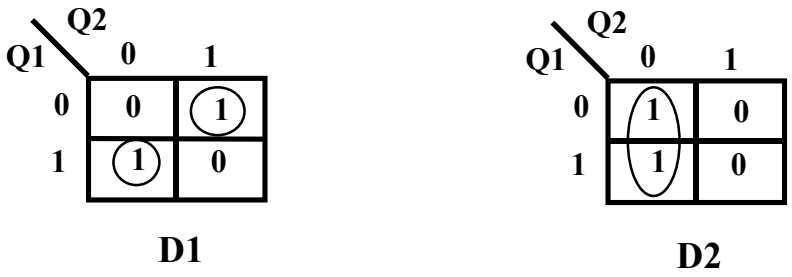
.

Now we have to design F ( ), consisting of two combinational circuits indicated by **Circuit 1** and **Circuit 2**. The first step is to construct a "truth table", since this is what we need to design a combinational circuit. For sequential systems which are designed with D-type flip-flops, this table is called the **transition table** (appropriately named) in which the "inputs" are the circuit inputs **plus** the flip-flop outputs at time t_n. The outputs are the flip-flop outputs at time t_n+1. In the case of the binary up-counter we have no independent inputs:

t_n (inputs)		t_{n+1} (outputs)	
Q1	**Q2**	**Q1**	**Q2**
0	**0**	**0**	**1**
0	**1**	**1**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**0**

.

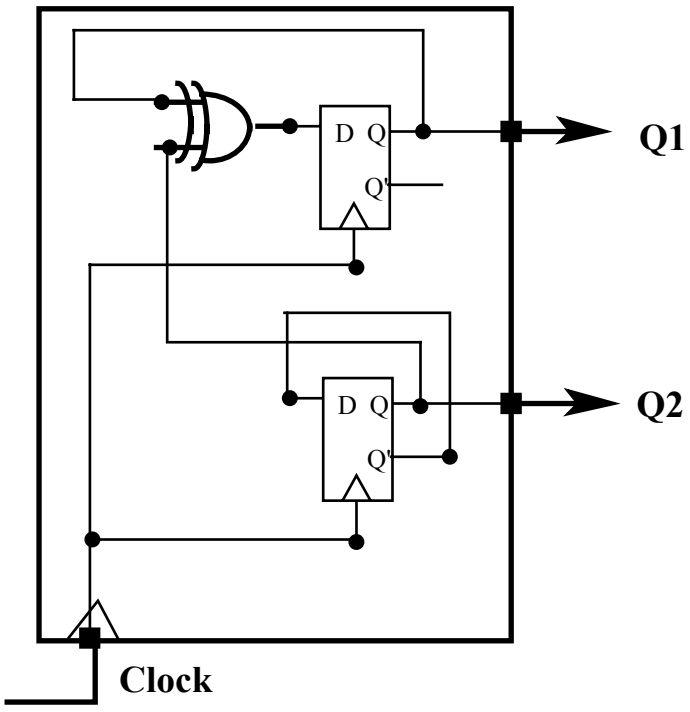
From the transition table the two K-maps are constructed and the minimised Boolean expressions are shown below:



.

$D1(t_n) = Q1(t_{n+1}) = Q1' \bullet Q2 + Q2' \bullet Q1 = Q1 \oplus Q2$ **$D2(t_n) = Q2(t_{n+1}) = Q2'$**
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, **C1**. When **C1=0** the counter counts up even numbers, i.e. **0->2->4->6->0->...,** and in binary**: 000->010->100->110->000->**.... When **C1=1** the counter counts down odd numbers; ie. **0->7->5->3->1->0->...,** and in binary**: 000->111->101->011->001->000->...** It is not important what sequence is produced if at the start the counter is in an undefined state (say, **C1=1** and the counter output is **2=010**), however, the counter should reach the **000** state sooner or later if enough number of clock pulses are applied.

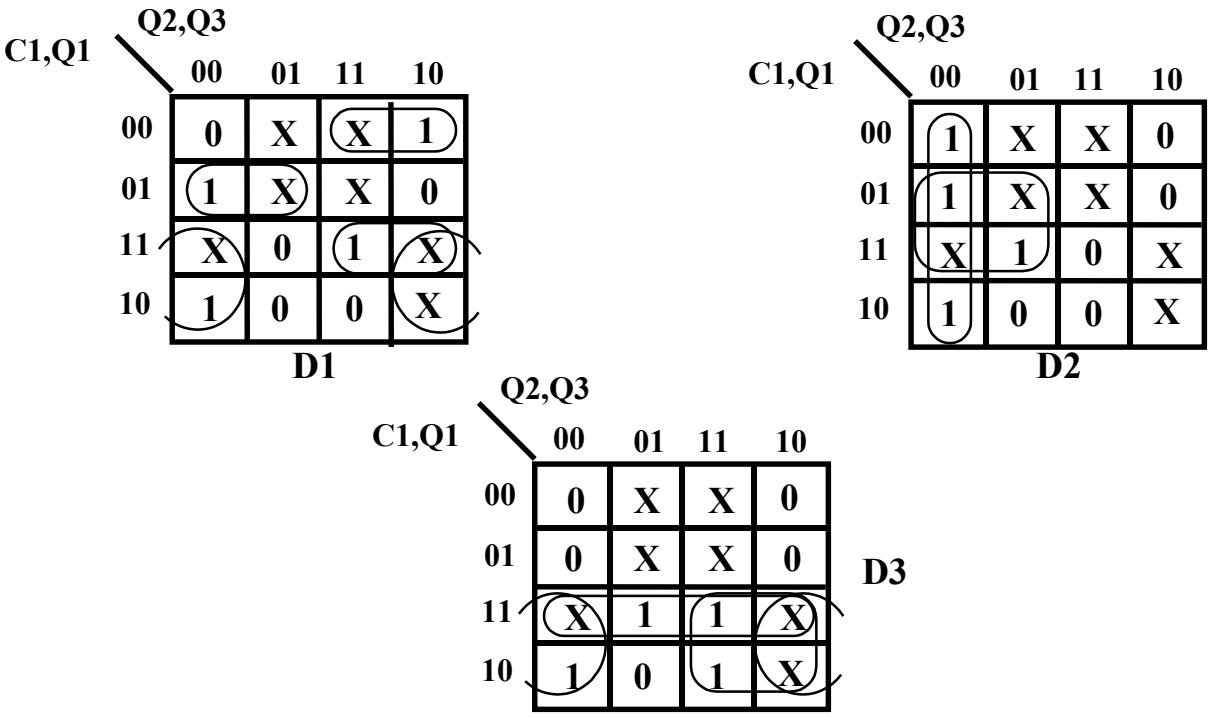
Step 1.

The transition table is produced. The don't care outputs **X** indicate that the state is not part of the counting defined sequence:

C1	Q1	Q2	Q3	D1	D2	D3
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

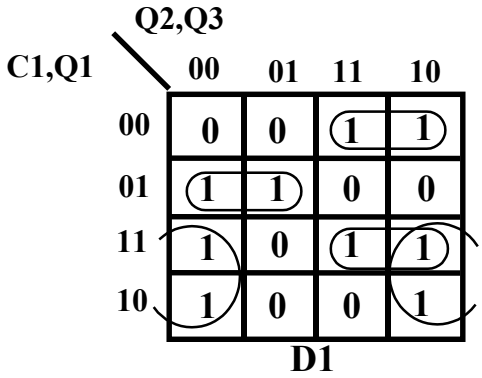
Step 2.

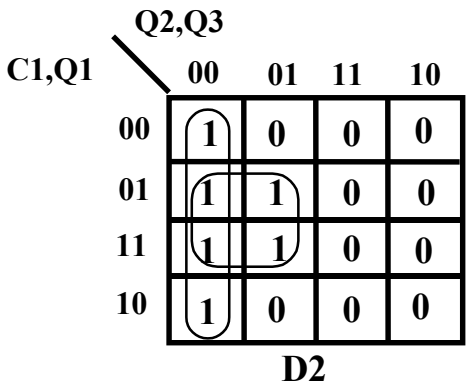
The K-maps and the suggested minimisation is indicated:

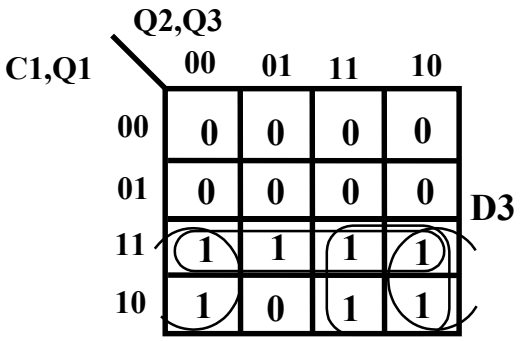


Step 3.

We now enter the true values of the outputs into the K-map instead of the "don't care" indicators. We can do this now since the indicated minimisation assigns each **X** to either a **0** or to a **1**.





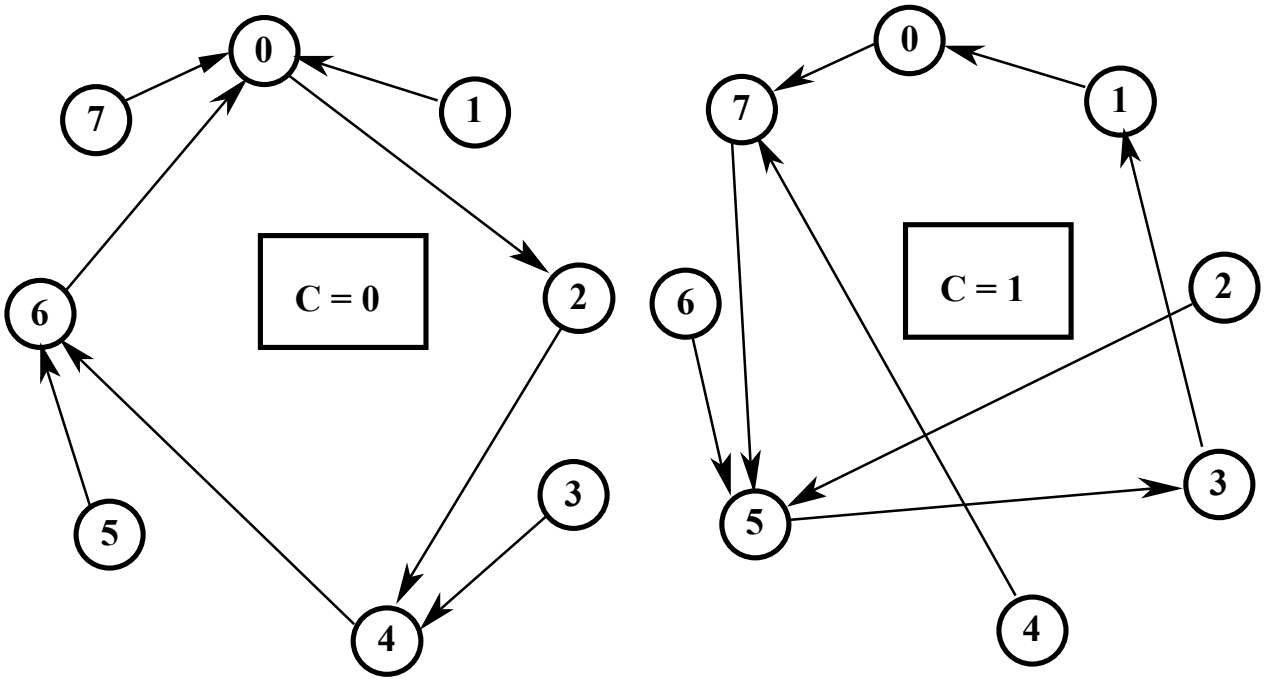


Step 4.

We now produce the correct transition table without don't cares indicating the binary states with their octal numerical equivalent **0-7.**

C1	Q1	Q2	Q3	D1	D2	D3	$S(t_n)$	$S(t_{n+1})$
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 state transition diagram is more handy which can be constructed from the table, as hown on the next page:



After closer examination we see that the required specifications are satisfied because for either control input case the counter will eventually reach **State 0**.

Step 5.

Build the circuit. Here we will assume that any basic gate (**AND, OR, NAND, NOR, XOR, XNOR, Inverter**) can be used. From the K-maps we have:

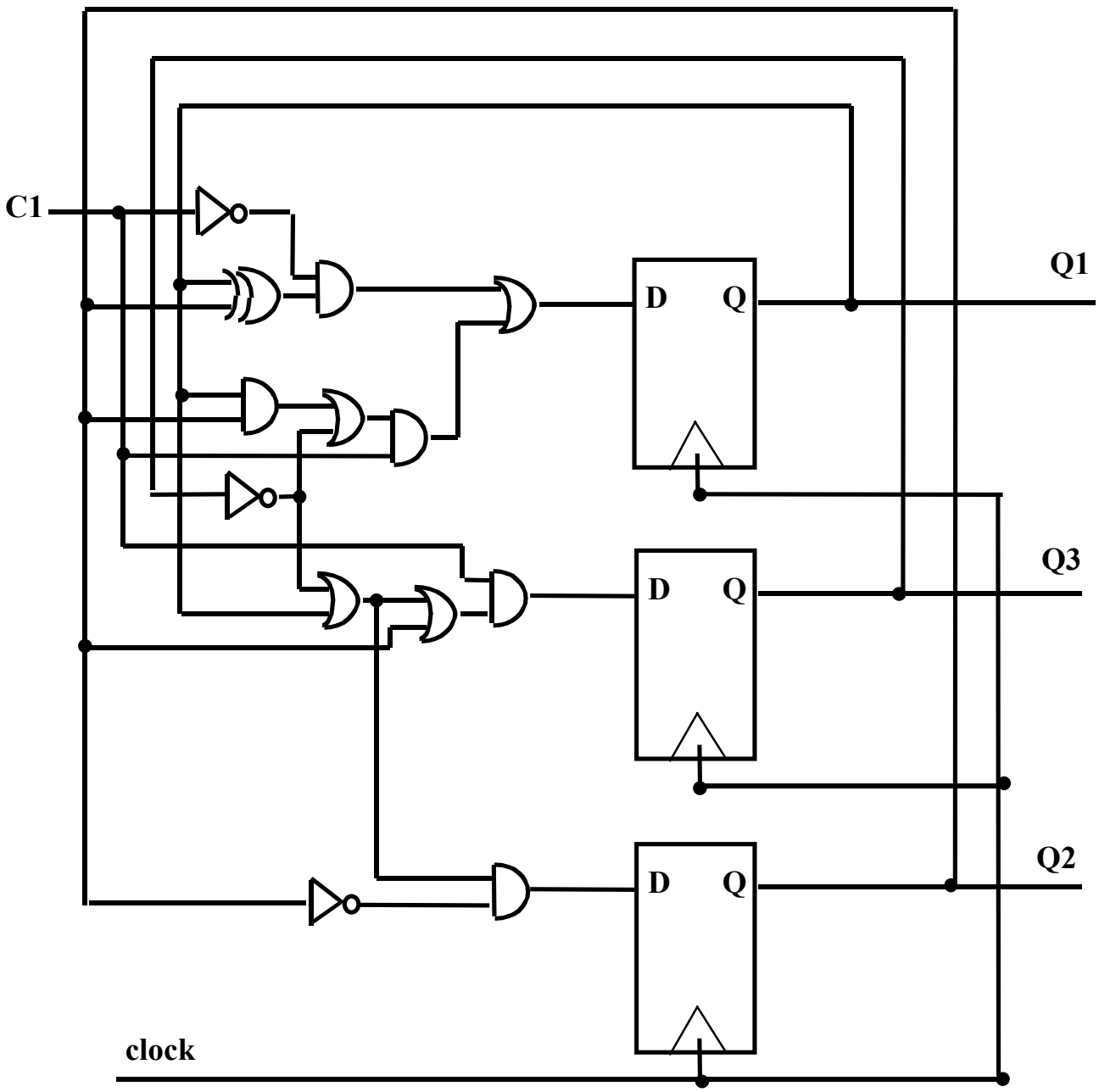
$$D1 = C1'\bullet Q1'\bullet Q2 + C1'\bullet Q1\bullet Q2' + C1\bullet Q1\bullet Q2 + C1\bullet Q3'$$
$$= C1'\bullet(Q1\oplus Q2) + C1\bullet(Q1\bullet Q2 + Q3')$$

$$D2 = Q2'\bullet Q3' + Q1\bullet Q2'$$
$$= Q2'\bullet(Q1 + Q3')$$

$$D3 = C1\bullet Q1 + C1\bullet Q3' + C1\bullet Q2$$
$$= C1\bullet(Q1 + Q2 + Q3')$$
$$= C1\bullet((Q1 + Q3') + Q2)$$

We have bracketed terms which appear in more than one Boolean expression. There is savings to be made by these expressions since only one set of gates have to be used.

The actual circuit is shown on the next page.


C1
D
Q
Q1
D
Q
Q3
D
Q
Q2
clock