Topic 5.1.2 – Synchronous counters

Learning Objectives:

At the end of this topic you will be able to;

- ☑ draw a block diagram showing how D-type flip-flops can be connected to form a synchronous counter to meet a given specification;
- ☑ explain how simultaneous clocking of D-type flip-flops overcomes the limitation of ripple counters at high counting speed;
- ☑ draw the state diagram for a synchronous counter given a system specification;
- ☑ explain the significance and cause of stuck states, and describe how they can be avoided by directing unused states back into the main sequence;
- ☑ manipulate unused (don't care) states to produce simpler solutions;
- ☑ analyse and design a synchronous counter (up to 3 bits) to obtain the state diagram for the sequence it produces.

Module ET5
Electronic Systems Applications.

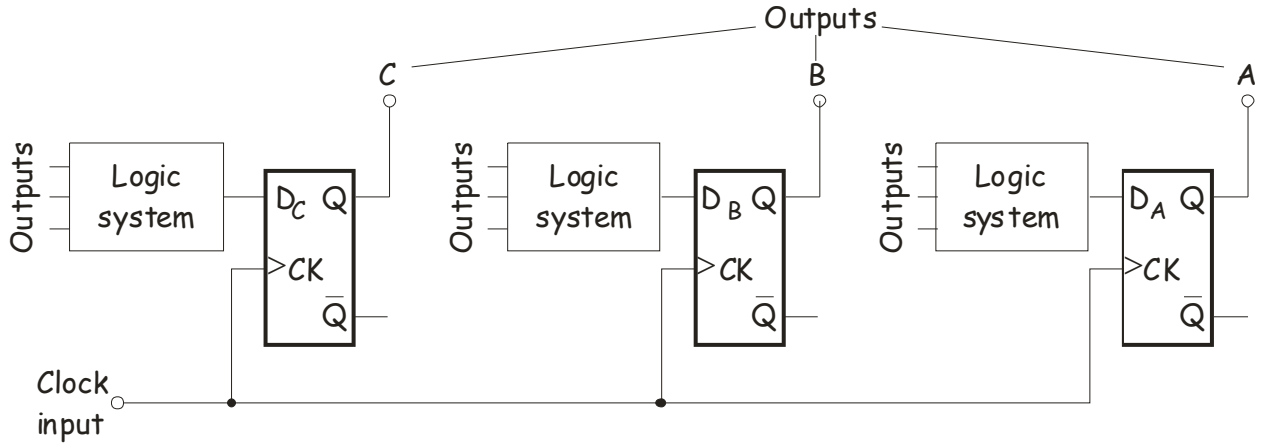
Synchronous counters

Synchronous counters differ from ripple counters in that:

- they can be designed to produce any sequence of output signals (and so are also known as sequence generators), whereas ripple counters can only count either up or down in binary;
- the clock inputs of all stages of the counter are connected together and so receive clock pulses at exactly the same time (and that is why they are called *synchronous!*);
- logic gates are used to generate appropriate signals at the data inputs of each stage.

In ripple counters, the clock signals move through the system, stage by stage, and so it takes time for the last stage to react to a pulse received at the first stage. This causes inaccuracy when the counter is counting at high speed. There is no such problem with the synchronous counter, because all stages receive the clock signal at the same time and so react at the same time.

The next diagram shows this basic structure for a 3-bit synchronous counter:



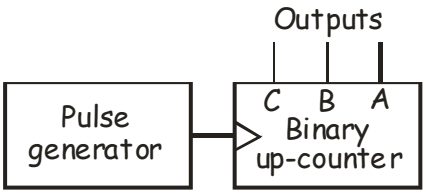
(The set and reset inputs of the D-types have been omitted to improve the clarity of the diagram.)

The syllabus expects you to be able to design a synchronous counter given the system specification, and to analyse a synchronous counter, given its circuit diagram, or the Boolean expressions linking the inputs and outputs.

1. Designing a synchronous counter:

Example 1 – A 3 bit counter

It might be a good idea to start by showing that we can make a synchronous counter behave as a normal binary up-counter, in this case counting pulses from a pulse generator.



We want the system to progress through the following sequence:

Pulse number	Outputs		
	C (m.s.b.)	B	A (l.s.b.)
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0
9	0	0	1
etc.			

The first step is to expand the truth table to show what inputs we need to generate at the D-inputs of the flip-flops in order to create this sequence.

Pulse number	Outputs			Inputs generated		
	C (m.s.b.)	B	A (l.s.b.)	D_C	D_B	D_A
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	1	1	1
7	1	1	1	0	0	0

Let's be clear about what this means. We need a logic system, attached to the data input D_C of the first flip-flop, which uses the signals $C = 0$, $B = 0$ and $A = 0$ to generate an output of logic 0, and then uses $C = 0$, $B = 0$, $A = 1$ to generate logic 0 again, and so on. Inputs D_B and D_A need their own logic systems.

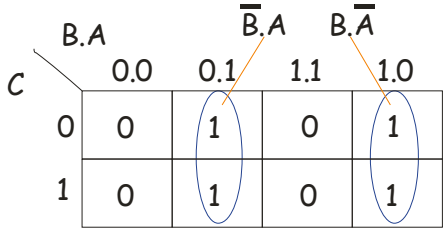
In this example, it is easiest to start with the D_A signal. Compare the D_A column and the 'A' output column in the table above. We do not need much of a logic system! D_A is always the opposite of output A. In other words:

$$D_A = \overline{A}$$

There is no need to involve outputs B and C. In fact, we don't need any logic gates at all, as the D-type flip-flop has a $\overline{Q}$ output which gives us the inverse of output A.

The other two logic systems, for the D_B and D_C input are more complicated. In general, Karnaugh maps may be needed to sort out the Boolean expressions.

For D_B:

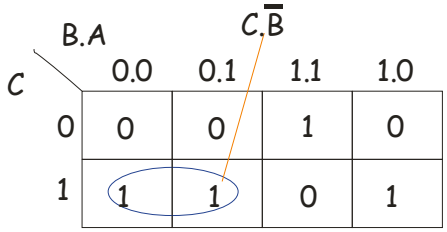


and so:

$$D_B = \overline{B}.A + B.\overline{A}$$

or, in a single logic gate: $D_B = B \oplus A$

For D_C:



and so the best we get from this map is:

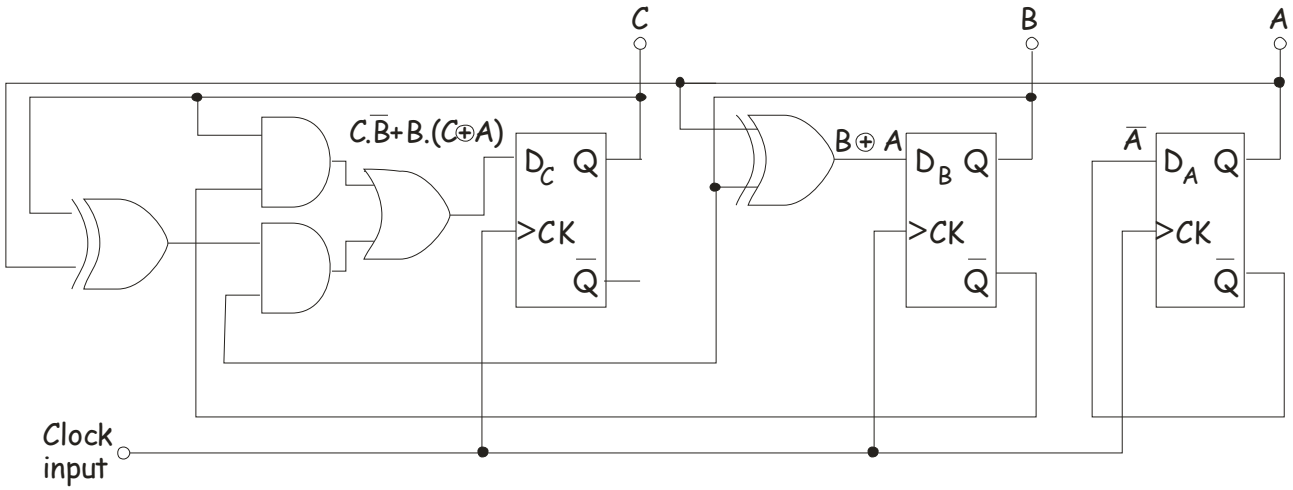
$$D_C = C.\overline{B} + \overline{C}.B.A + C.B.\overline{A}$$

Taking a factor of B from the last two terms gives:

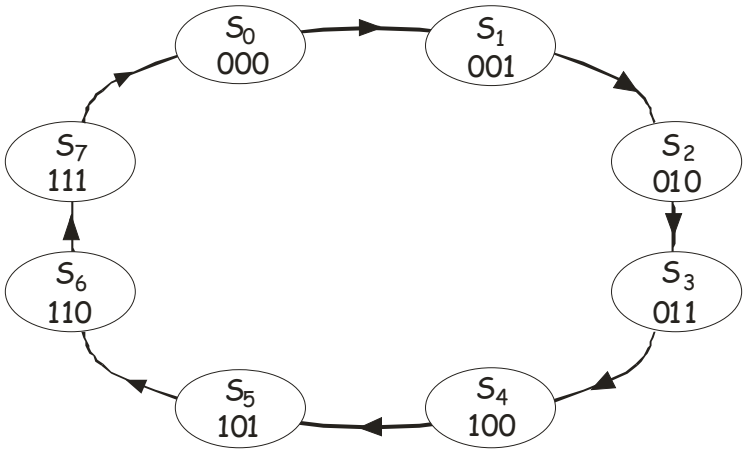
$$D_C = C.\overline{B} + B.(\overline{C}.A + C.\overline{A})$$

or: $D_C = C.\overline{B} + B.(C \oplus A)$

The circuit diagram for this system is given below.



We can represent the behaviour of this system in a **State Diagram**.



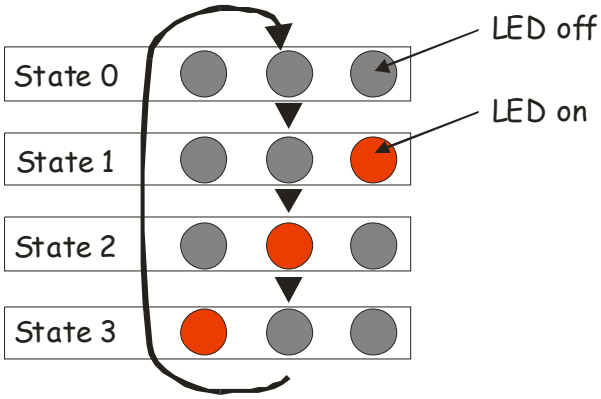
One point to make here is that when the system is powered up, it can start in any of the eight states. It does not automatically start at 000_2. However, in this system, it does not matter where it starts, as it will then count up in binary from that point, and reset when it reaches 111_2. The system could be forced to start at 000_2 by linking together the Reset inputs of the D-type flip-flops and activating them briefly when the system powers up.

This system is a little unusual in that the required sequence, called the **main sequence,** includes all possible states. In many cases, some of the possible states are unused. They can cause problems, as we will see in the next example.

Example 2 – A LED light chaser

System specification:
The behaviour of a synchronous counter is usually specified in either a truth table, a state diagram, or in visual representation of the sequence as shown in the next diagram. These are all different versions of the same information.

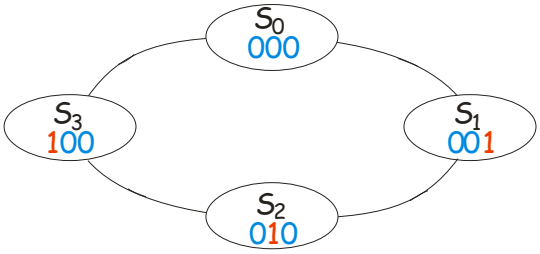
For example, suppose that you have three LEDs, A, B and C. You want to start off with all of them turned off and then turn on only one at a time in a repeating sequence. Visually, the sequence looks like:



Here is the same specification expressed in the form of a truth table.

State number	LEDs		
	C	B	A
0	Off	Off	Off
1	Off	Off	On
2	Off	On	Off
3	On	Off	Off

Finally, here is the same specification shown as a state diagram (**almost!**), assuming that a LED is on when a logic 1 signal is applied, and off when logic 0 is applied.



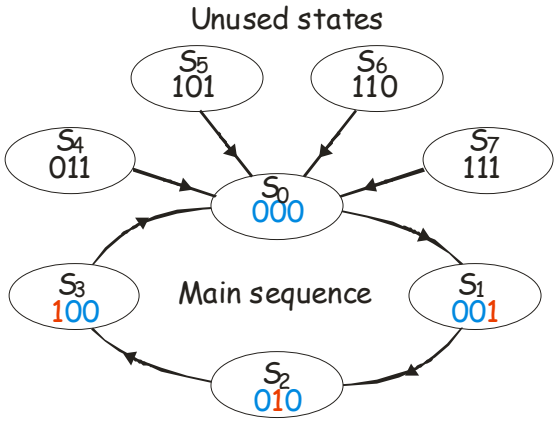
Why '**almost**'- because we need to worry about unused states!

Unused states:

The synchronous counter will drive the three LEDs in the correct sequence. It will have three digital outputs, which means that there are eight (= 2^3) possible combinations of these outputs. We are using only four of these – 000, 001, 010 and 100. This means that there are four **unused states** – 011, 101, 110 and 111.

The problem is that the system can output any of the eight possible combinations *when it is first switched on*. We have to design the counter so that even if, on power-up, it starts in an unused state, it will then progress onto the required sequence.

We complete the state diagram by showing how the system will deal with the unused states. One possible solution, but rarely the best, is to link all the unused states to state S_0, the 000 state. This is shown below:



Now, even if the system starts in an unused state, say state S_4, and outputs 011, when the first clock pulse arrives, the system moves to the S_0 state, in the main sequence. After that, the next clock pulse moves the system to the S_1 state, and after that it continues through, and is locked into, the main sequence.

The full truth table for this solution is:

	State number	Current state of LEDs			Next state of LEDs		
		C	B	A	D_C	D_B	D_A
Main sequence	0	0	0	0	0	0	1
	1	0	0	1	0	1	0
	2	0	1	0	1	0	0
	3	1	0	0	0	0	0
Unused states	4	0	1	1	0	0	0
	5	1	0	1	0	0	0
	6	1	1	0	0	0	0
	7	1	1	1	0	0	0

Notice that the table has two main columns – the **current** state of the system, and the **next** state. The headings D_C, D_B and D_A relate to the D-type flip-flops, used to build the synchronous counter.

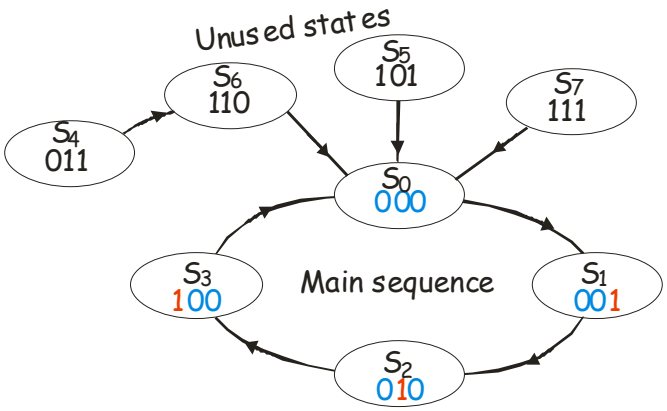
Next, we determine what logic gates are needed, and with what inputs, to supply the correct signals to the data inputs of the D-types. This is done either by inspection, or by use of Karnaugh maps, introduced in ET1, section 1.2.3.

In this case, we can do this by inspection. The Boolean expressions linking current inputs and outputs are:

$$D_C = \overline{C}.B.\overline{A}$$
$$D_B = \overline{C}.\overline{B}.A$$
$$D_A = \overline{C}.\overline{B}.\overline{A}$$

The drawback of connecting all unused states to the 000 state is that usually, this solution leads to complicated Boolean expressions and logic systems. While the expressions above are not wildly complicated, we can do better by linking the unused states in a different way. (They are unused, so we can do what we like with them as long as they lead into the main sequence.)

Here is a different state diagram for this system, though the main sequence is the same:



In this case, if the system powers up with all three LEDs switched on (the 111 state, which we have called state S_7,) then it should progress into state S_0, where all LEDs are off, and then continue through the main sequence.

The same thing happens if it powers up in state S_5 or S_6. If the system starts in state S_4, it progresses through state S_6 to the main sequence, but take two clock cycles to do so.

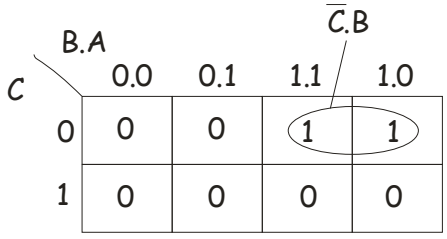
The unused states can be connected anywhere as long as they lead into the main sequence. The reason for designing the system as shown above is that it leads to simpler Boolean algebra.

The truth table for the full design is now:

	State number	Current state of LEDs			Next state of LEDs		
		C	B	A	D_C	D_B	D_A
Main sequence	0	0	0	0	0	0	1
	1	0	0	1	0	1	0
	2	0	1	0	1	0	0
	3	1	0	0	0	0	0
Unused states	4	0	1	1	1	1	0
	5	1	0	1	0	0	0
	6	1	1	0	0	0	0
	7	1	1	1	0	0	0

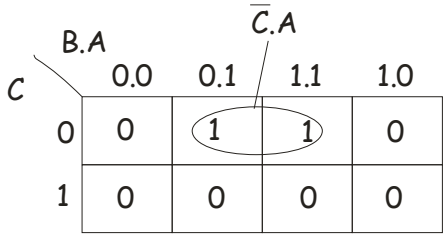
We will use Karnaugh maps to sort out the algebra:

for D_C:



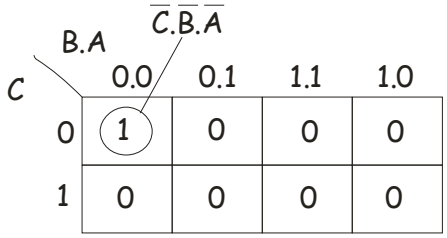
Hence $D_C = \overline{C}.B$;

for D_B:



Hence $D_B = \overline{C}.A$;

for D_A:



Hence $D_A = \overline{C}.\overline{B}.\overline{A}$ (or $\overline{C+B+A}$);

You see that we have two options for D_A. We will choose the second as it is easier to generate. However, either solution is acceptable!

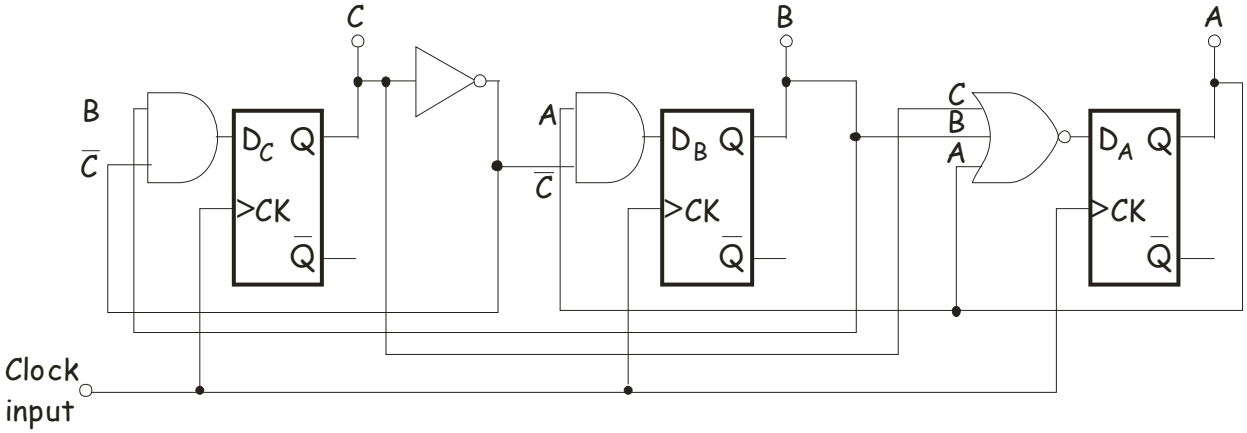
The Boolean relationships between inputs and outputs are then:

$$D_C = \overline{C}.B$$
$$D_B = \overline{C}.A$$
$$D_A = \overline{C+B+A}$$

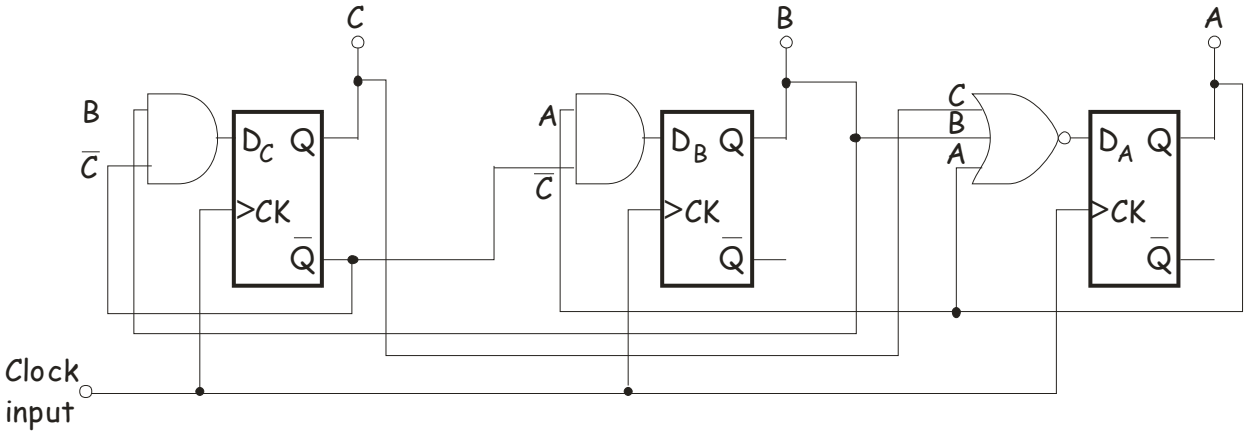
Notice that these are a bit simpler than the expressions produced on page 8, when all unused states were connected directly to S_0.

Finally, we can now draw the circuit diagram for this synchronous counter. However, there are several ways to do this!

It could look like:



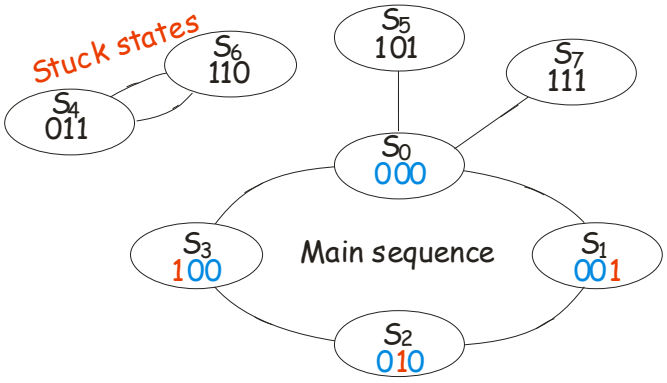
The next version is better, because it uses fewer logic gates (= fewer ICs = cheaper and more reliable,) by generating the $\overline{C}$ signal from the $\overline{Q}$ output:



Stuck states:

Stuck states are unused states that do not progress into the main sequence.

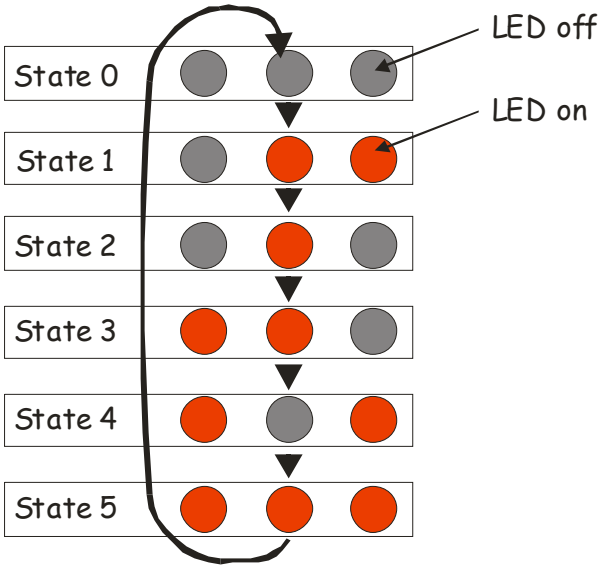
Careless design of a system can lead to a situation where, on power-up, the system locks in an unused state, and never progresses to the main sequence. The next state diagram shows the same main sequence that was used earlier but with a different, and deadly, arrangement of unused states:



Now, when you switch on the system, there is a chance that it starts in either state S_4 or S_6. If so, as the three D-type clock inputs receive pulses, the outputs simply alternate between the 011 and 110 states. They never reach the main sequence. These are known as **stuck states**. Remember, these are only a problem on power-up. Once the system reaches the main sequence, it continues to cycle around the states in the main sequence.

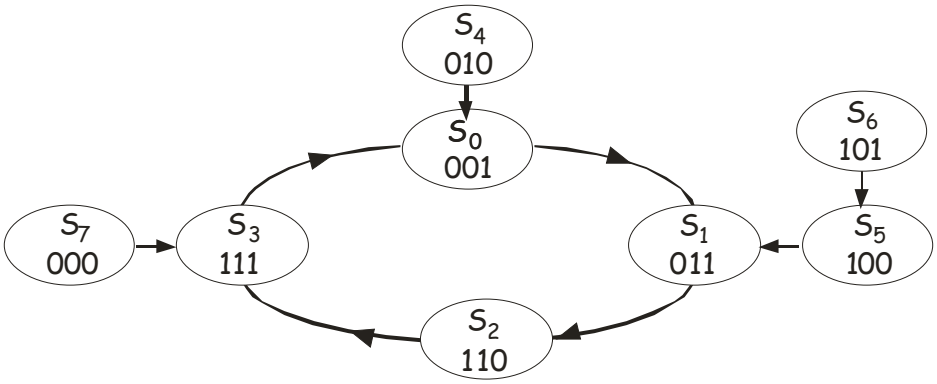
Exercise 1 (Solutions are given at the end of this topic)

1. Three LEDs are switched on and off in the following sequence:



Draw a state diagram for this system, taking care to avoid stuck states.

2. Here is the state diagram for a synchronous counter.



Complete the following table for this counter:

State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	1			
1						
2						
3						
4						
5						
6						
7						

Manipulating unused (don't care) states

Example 1:

Consider the following design problem.
In an industrial process, a pump is used to fill a drum. Once it is partly full, a motor rotates it to mix the contents, and continues to do so after the pump is switched off. Then the motor is turned off, and a valve opens to empty the drum. Finally, the system switches off all the devices and the sequence is repeated. (This example may not be very realistic, but it makes the point that synchronous counters can control a variety of output devices, through suitable interfaces.)

We can use a synchronous counter to control this process, providing all the steps last for the same amount of time (which will be the period of the clock signal.) To design the counter, we first of all turn the description of the sequence of events into a truth table to show the main sequence. We assume that a logic 1 signal turns a device on, and a logic 0 turns it off.

Make sure that you are happy that the contents of the table match the description above

	State number	Current state			Next state		
		Pump C	Motor B	Valve A	D_C	D_B	D_A
Main sequence	0	0	0	0	1	0	0
	1	1	0	0	1	1	0
	2	1	1	0	0	1	0
	3	0	1	0	0	0	1
	4	0	0	1	0	0	0
Unused states	5	0	1	1	X	X	X
	6	1	0	1	X	X	X
	7	1	1	1	X	X	X

X = Don't care

Next, we work what logic gates are needed by looking at the Boolean relationships between outputs and inputs, ignoring the unused states. These are also called 'don't care' states, because it doesn't matter where they go (as long as they lead into the main sequence.)

Once we decide what the relationships are, we then use them to determine the fate of the three unused states.

1. Look at the D_B column. It is identical to the *C* column. We will go for this straightforward relationship:

$$D_B = C$$

2. The D_C column has logic 1 entries in the rows only where B and *A* are both logic 0. We can specify this relationship as:

$$D_C = \overline{B} \,.\, \overline{A}$$

3. The D_A column has a logic 1 entry in State 3. This is the only time in the main sequence that B = 1 and *C* = 0. We can specify this relationship as:

$$D_A = \overline{C} \,.\, B$$

Another viewpoint sees that State 3 is where *C* =0 and B = 1 and *A* = 0. This would result in the relationship as

$$D_A = \overline{C} \,.\, B \,.\, \overline{A}$$

This is electronically more complicated, requiring a 3-input AND gate rather than a 2-input one. We will use the first relationship.

We obtained these relationships by inspection. It is always worth checking them using Karnaugh maps. We use this approach later, in example 2.

Now that we have the Boolean expressions, we can return to the question of the unused states. These Boolean expressions will govern what happens to the unused states, as well as to the main sequence. The unused states are:
011, 101 and 111.

We apply the Boolean expressions, obtained above, to these to see what state each leads into.

State number	Current state			Next state		
	Pump *C*	Motor B	Valve *A*	D_C	D_B	D_A
5	**0**	**1**	**1**	0	0	1

State number	Current state			Next state		
	Pump *C*	Motor B	Valve *A*	D_C	D_B	D_A
6	**1**	**0**	**1**	0	1	0

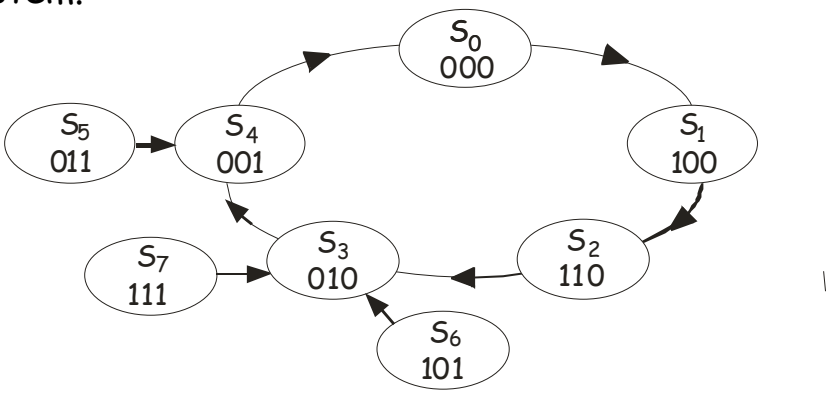
State number	Current state			Next state		
	Pump *C*	Motor B	Valve *A*	D_C	D_B	D_A
7	**1**	**1**	**1**	0	1	0

Be clear about what we have just done! We first decided on the set of logic gates needed to produce the main sequence. We have just looked at what these gates will do when the system powers up into an unused state. It is vital that these lead into the main sequence, that they are not stuck states, in other words.

Including the results for the unused states, the full truth table for the control system is:

State number	Current state			Next state		
	Pump C	Motor B	Valve A	D_C	D_B	D_A
0	0	0	0	1	0	0
1	1	0	0	1	1	0
2	1	1	0	0	1	0
3	0	1	0	0	0	1
4	0	0	1	0	0	0
5	0	1	1	0	0	1
6	1	0	1	0	1	0
7	1	1	1	0	1	0

It is vital that these unused states do not form stuck states. To check whether this will happen, we now use this table to draw the state diagram for this control system.

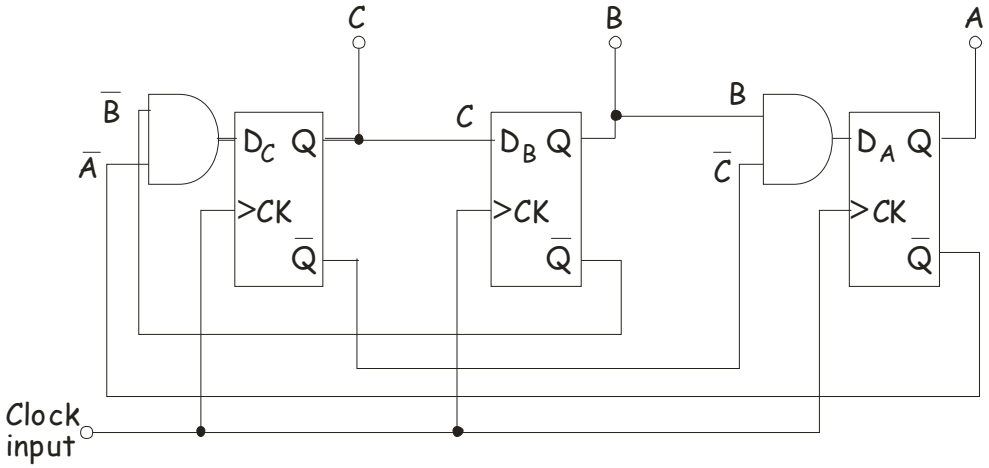


As you can see, the unused states all lead into the main sequence. If the control system powered up in an unused state, it would progress onto the main sequence in the next clock cycle.

A different question:
Is it a problem that that the main sequence can start at any point? It's not an issue that concerns us at the moment! We are simply using this example to show that synchronous counters can control devices other than LEDs!

However, to ensure that the sequence always starts in the 000 state, the reset pins of the D-types can be linked to a 'Start Process' power switch.

The circuit diagram for this solution is:



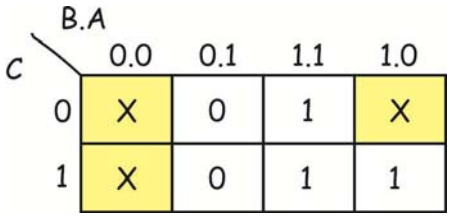
Example 2:

The application does not concern us. This system could be controlling a set of lights, or the machinery in an industrial manufacturing process. It does not matter. We are going to focus on how to handle the unused or 'don't care' states.

Consider the following truth table with the unused or 'don't care' states marked with an X

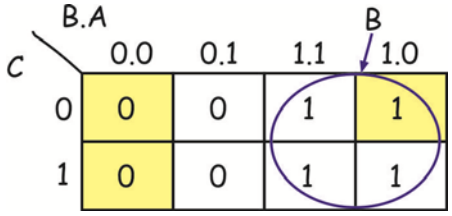
State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	1	0	1	1
1	0	1	1	1	1	0
2	1	1	0	1	1	1
3	1	1	1	1	0	1
4	1	0	1	0	0	1
5	0	0	0	X	X	X
6	1	0	0	X	X	X
7	0	1	0	X	X	X

Here is the Karnaugh map for D_C:



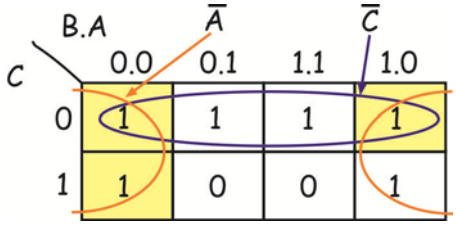
Notice that three of the boxes contain 'don't care' states. We can choose to make these either logic 0 or logic 1, in order to make the Boolean algebra and the electronics easier to implement. In this case we will make the 'don't care' state in the top right hand corner into a 1 and make the other two 'don't cares' into 0's

The map becomes



giving $D_C = B$

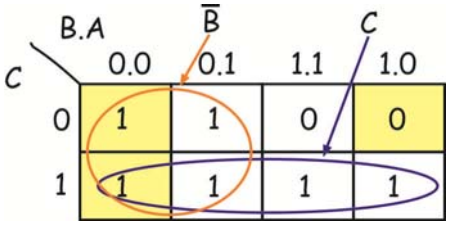
In the same way, here is the Karnaugh map for D_B:



In this case we have converted all the 'don't care's into logic 1's to give:

$$D_B = \overline{C} + \overline{A}$$

Finally, here is the Karnaugh map for D_A:



In this case we have converted the 'don't care' state in the top right hand corner into logic 0 and the other two into logic 1 to give:

$$D_A = C + \overline{B}$$

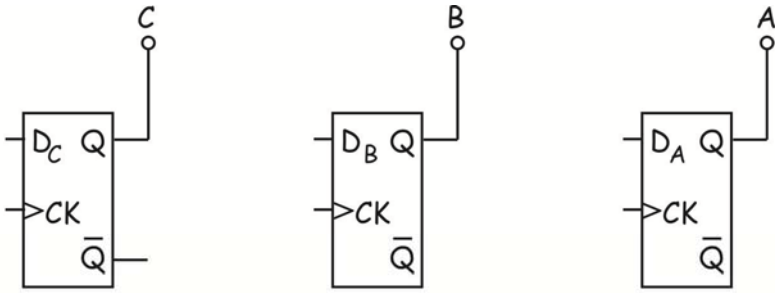
We must then apply the Boolean expressions for D_A, D_B and D_C to find the corresponding 'Next state' and complete that column of the truth table to make sure that there are no stuck states.

State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
5	0	0	0	0	1	1
6	1	0	0	0	1	1
7	0	1	0	1	1	0

Now complete the state diagram and circuit diagram for this system:

State diagram

Circuit diagram



Clock input

Which method do I use?

Although the first method is more direct it requires you to spot a Boolean expression from a column of 0's and 1's.
The second method takes longer but has a more visual approach.
Choose whichever method works for you!

Exercise 2: (Solutions are given at the end of this topic)

1. A sequence generator is needed to provide signals for a lighting display control system.

   The next table gives the main sequence of signals C, B and A.

State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	0			
1	1	1	1			
2	0	1	0			
3	1	0	1			
4	0	1	1			

   States 5, 6 and 7 are unused. The states they progress into have been chosen to simplify the system

State number	C	B	A	D_C	D_B	D_A
5	0	0	1	0	1	1
6	1	0	0	1	1	1
7	1	1	0	1	1	1

   (a) Complete the first table to show the inputs D_C, D_B and D_A needed to generate the sequence.

   (b) Determine Boolean expressions for D_C, D_B and D_A in terms of outputs C, B and A. (Simplify the expressions as much as possible, using the rules of Boolean algebra or Karnaugh maps.)

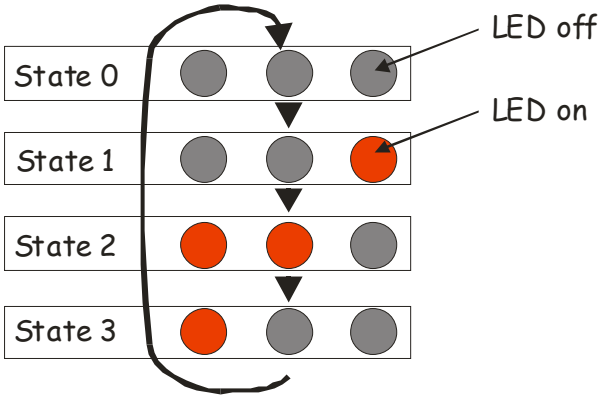
   D_C = ……

   D_B = ……

   D_A = ……

   (c) Complete the circuit diagram for this system by adding:
   - correct clock connections,
   - appropriate logic gates correctly connected.

2. Back to controlling LEDs!
   Design a synchronous counter that will produce the following 'light chaser' effect:



   Your final design should include a truth table and state diagram showing both main sequence and unused states, and the circuit daigram.

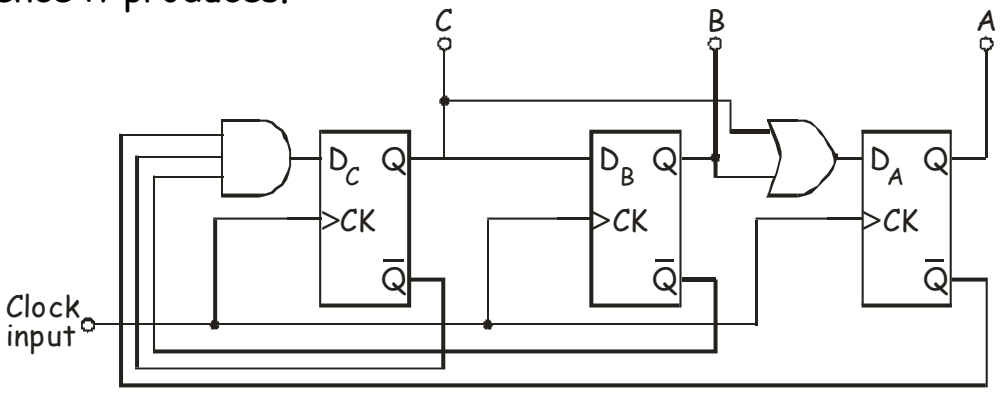
3. Design a synchronous counter that will count up in binary from 000_2 to 100_2, and then, on the next clock pulse, reset to 000_2.

   Again, your final design should include a truth table and state diagram showing both main sequence and unused states, and the circuit diagram.

Analysing a synchronous counter:

Example 1

Here is the circuit diagram for a synchronous counter. The task is to find out what sequence it produces.



First of all, we write down Boolean expressions for the inputs D_C, D_B and D_A in terms of the outputs C, B and A.

In this case:

$$D_C = \overline{C}.\overline{B}.\overline{A}$$

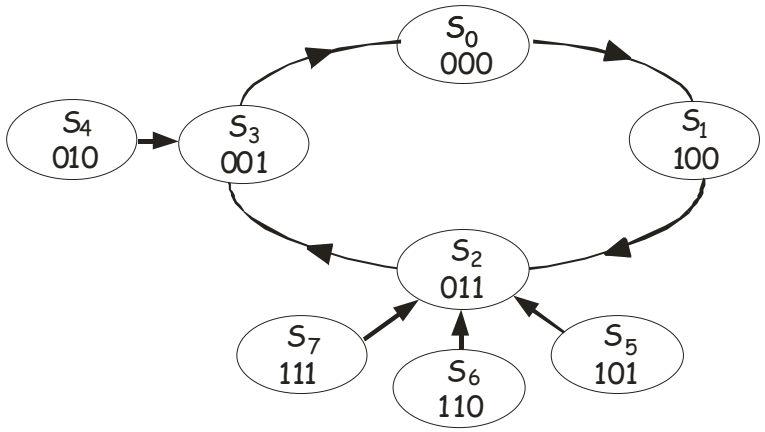
$$D_B = C$$

$$D_A = C + B$$

Next we use these relationships to complete the truth table to show the sequence of output states produced by this system. Unless we know one of the states in the main sequence, we start the table at the 000 state, and see what happens.

State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	0	1	0	0
1	1	0	0	0	1	1
2	0	1	1	0	0	1
3	0	0	1	0	0	0
4	0	1	0	0	0	1
5	1	0	1	0	1	1
6	1	1	0	0	1	1
7	1	1	1	0	1	1

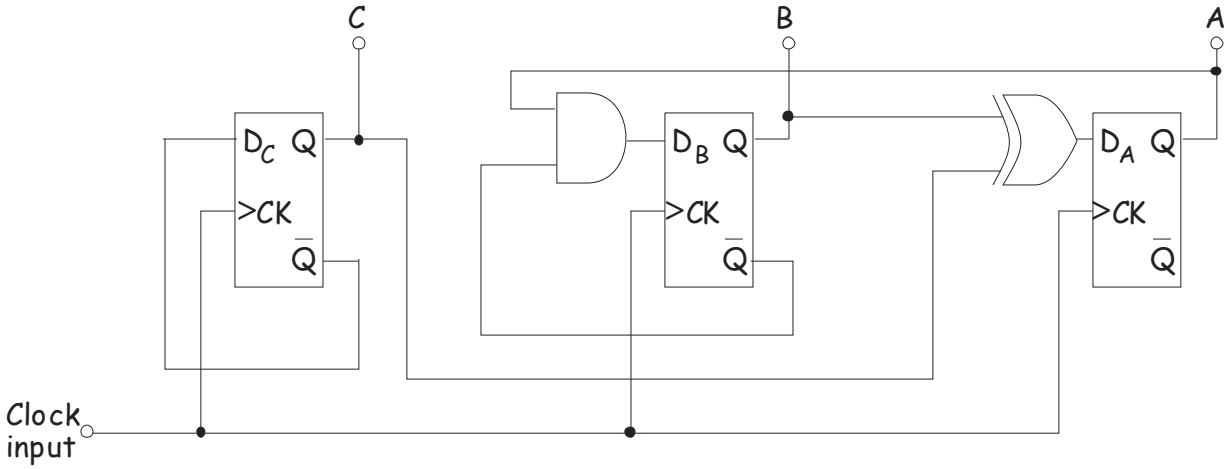
In this case, we were lucky. The 000 state was part of the main sequence. We know when the main sequence comes to an end because it returns to the first state we looked at. Altogether there are four states in the main sequence, and four unused states. We need to check whether any of these are stuck states. To do this, we use the truth table to create the state diagram.



We can now see that the system functions without any risk of stuck states.

Example 2

Here is the circuit diagram for another sequence generator.



The first step is to deduce the Boolean expressions linking inputs and outputs. By inspecting the circuit diagram, these are seen to be:

$$D_C = \overline{C}$$

$$D_B = \overline{B} \cdot A$$

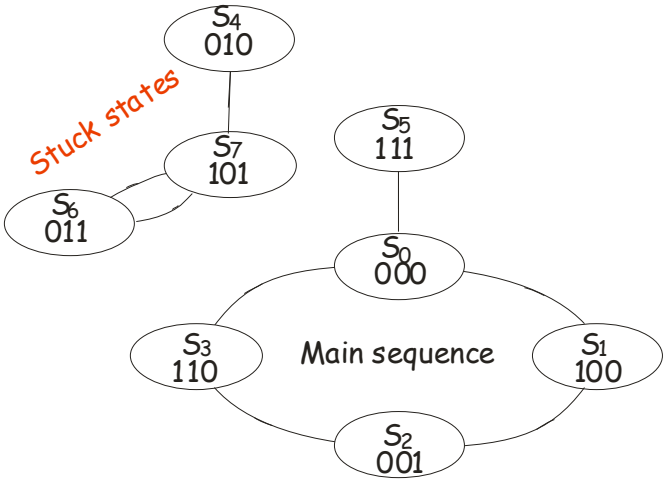
$$D_A = C \oplus B$$

The next step is to use these Boolean expressions to complete the truth table for this system. It does not matter which state we start with.

State	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	0	1	0	0
1	1	0	0	0	0	1
2	0	0	1	1	1	0
3	1	1	0	0	0	0
4	0	1	0	1	0	1
5	1	1	1	0	0	0
6	0	1	1	1	0	1
7	1	0	1	0	1	1

You can see that the first four states make up a sequence, presumably the main sequence. States 4, 5, 6 and 7 are, we assume, the unused states.

Now use the truth table to draw the state diagram for this system:

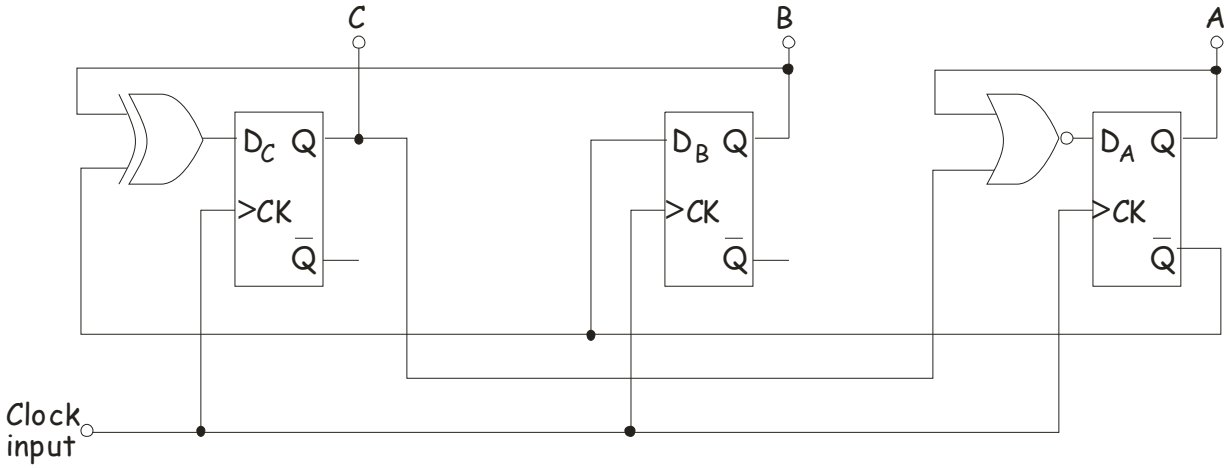


You should be able to spot a major problem with the unused states. Only one, S_5, leads into the main sequence. If the system powers up in either S_4, S_6 or S_7, then it can never progress into the main sequence. These are stuck states!

Exercise 3: (Solutions are given at the end of this topic)

Analyse the sequence produced by the following synchronous counter by:

- obtaining the Boolean expressions linking the inputs and outputs;
- completing the truth table;
- drawing the state diagram, including the unused states.



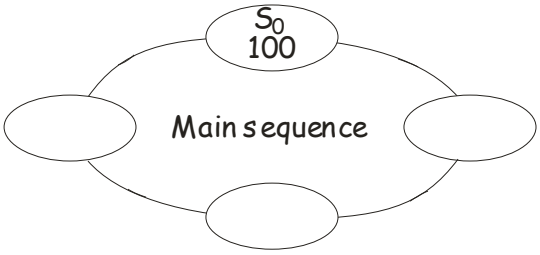
Boolean expressions:

D_C = ... D_B = ... D_A = ...

Truth table: *(Hint – the first state (100) is part of the main sequence. You should find that there are four states altogether in the main sequence.)*

State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	1	0	0			
1						
2						
3						
4						
5						
6						
7						

State diagram: *(Don't forget to add the unused states!)*



Module ET5
Electronic Systems Applications.

Practice Exam Questions:

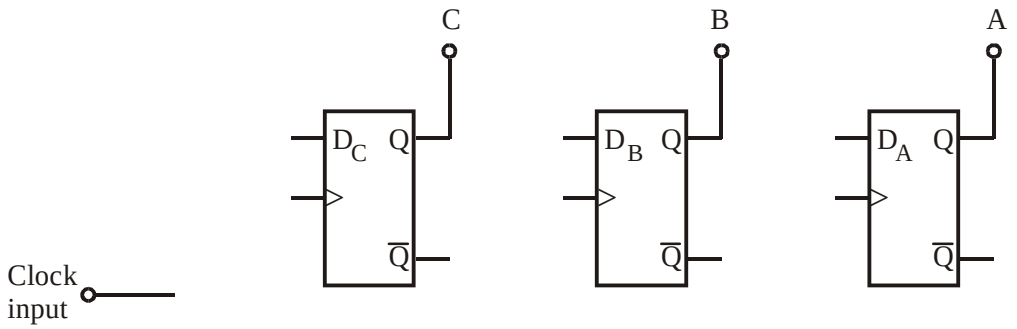
1. A sequence generator is governed by the following Boolean equations:

$$D_A = \overline{A}$$

$$D_B = B \oplus A$$

$$D_C = \overline{B \oplus A}$$

(a) Complete the circuit diagram for this sequence generator by adding:
(i) correct clock connections for the three D-type flip-flops,
(ii) logic gates to provide the required input signals for the D-type flip-flops. [4]



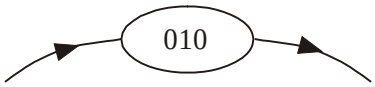
(b) Complete the following truth table to show the sequence of states produced by this system. You should find that the sequence contains only four states. [3]

State	C	B	A	D_C	D_B	D_A
0	0	1	0			
1						
2						
3						

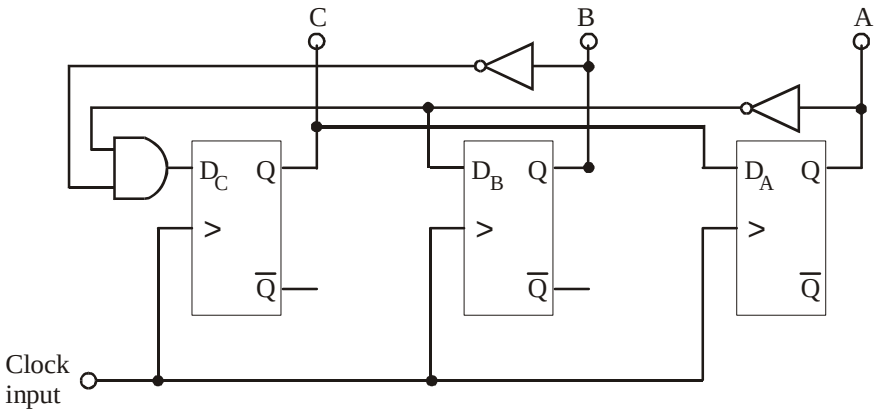
(c) Identify the four unused states and for each one, show into which state the unused state will lead. [4]

Unused state			Leads into		
C	B	A	D_C	D_B	D_A

(d) Hence draw the state diagram for this sequence generator. [2]



2. A student designs a light-chaser effect, based on a synchronous counter, for a model car. Part of the circuit diagram is shown below.



(a) The circuit can be simplified without changing its performance.
Explain how to modify the circuit so that the NOT gates are not needed. [1]

……..

……..

……..

(b) Write down Boolean expressions for the inputs D_A, D_B and D_C in terms of the outputs A, B and C. [3]

D_A = …….

D_B = …….

D_C = …….

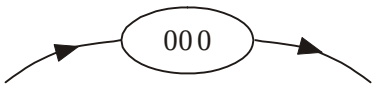
(c) Use these Boolean expressions to complete the table, showing the sequence of output states that this system will generate. You should find there are only **three** states in the sequence. [2]

State	A	B	C	D_A	D_B	D_C
0	0	0	0			
1						
2				0	0	0

(d) Complete the next table by listing the unused states, and by working out the state that each one progresses into. [5]

	Current Outputs			Next Outputs		
State	A	B	C	A	B	C
3						
4						
5						
6						
7						

(e) Hence draw the state diagram for this system [2]



(f) There is a serious defect with the design of this sequence generator.

(i) Explain what this defect is, and why it might cause a problem. [2]

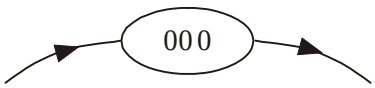
……..

……..

……..

(ii) Redraw the state diagram showing how to overcome this defect, without changing the main sequence.

[1]



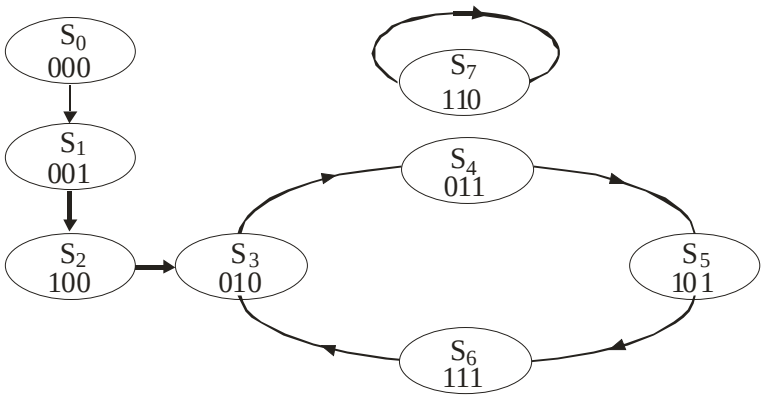
3. *(a)* What makes synchronous counters more suitable than ripple counters for counting high frequency pulses? [1]

……..

……..

……..

(b) Here is the state diagram for a synchronous counter.



(i) Identify the main sequence for this counter by listing the states that it contains [1]

……..

(ii) Identify any stuck states. [1]

……..

(iii) Why is it important to avoid stuck states when designing synchronous counters? [1]

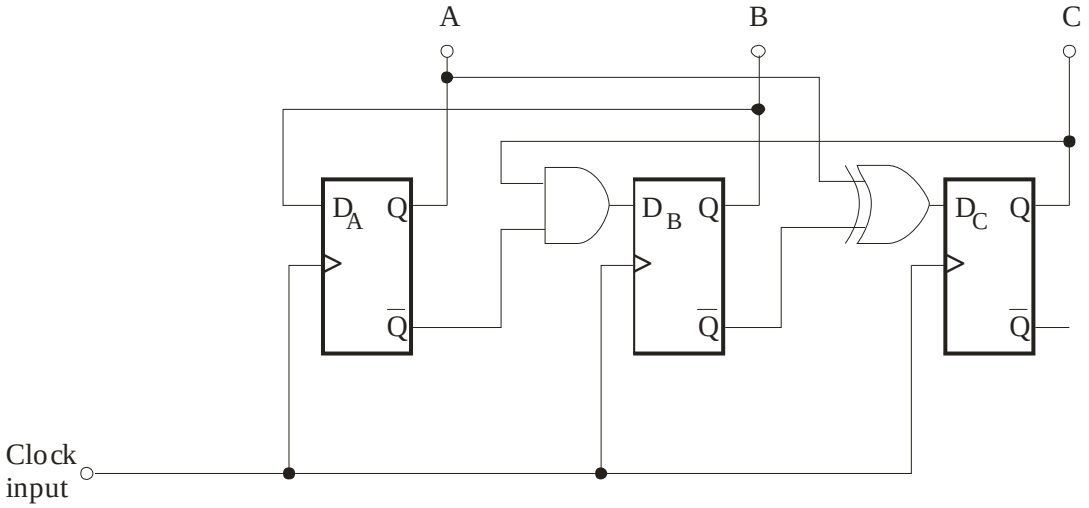
……..

……..

……..

……..

4. The circuit diagram for a synchronous counter is shown below.



(a) Give the Boolean expressions for the inputs D_A, D_B and D_C in terms of the outputs A, B and C. [3]

D_A = ……..

D_B = ……..

D_C = ……..

(b) Complete the table to show the sequence of states produced by the synchronous counter.
*You should find that there are **five** different states in the sequence.* [5]

State	A	B	C	D_A	D_B	D_C
0	0	0	0			
1						
2						
3						
4						

(c) Complete the following table by:

(i) identifying any unused states,

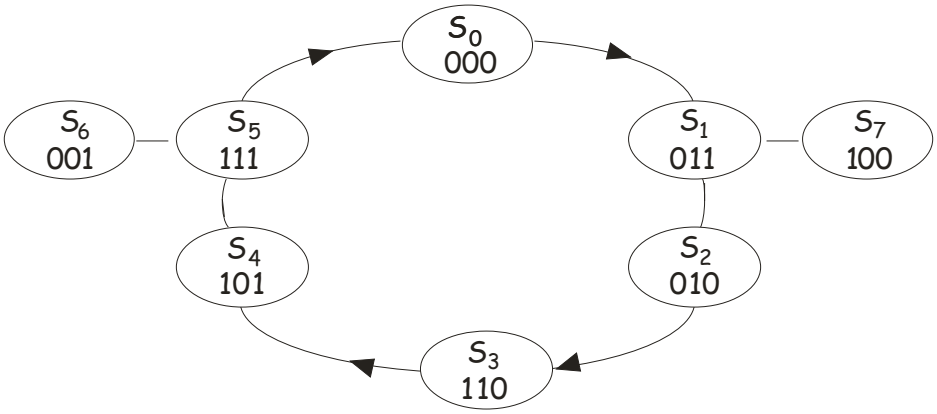
(ii) determining the values of D_A, D_B and D_C which they produce, [4]

Unused state			Produces		
A	B	C	D_A	D_B	D_C

Solutions to Exercises:

Exercise 1:

1.



This is not the only correct solution. The main sequence must be that shown in the diagram, but the unused states, S_6 and S_7, can be connected in any arrangement that leads to the main sequence.

2

	State	Current state			Next state		
		C	B	A	D_C	D_B	D_A
Main sequence	0	0	0	1	0	1	1
	1	0	1	1	1	1	0
	2	1	1	0	1	1	1
	3	1	1	1	0	0	1
Unused states	4	0	1	0	0	0	1
	5	1	0	0	0	1	1
	6	1	0	1	1	0	0
	7	0	0	0	1	1	1

Notice that within the main sequence, the 'Next state' in one row becomes the 'Current state' in the next row. This is not true for the unused states.

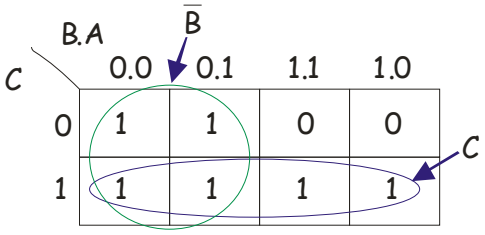
Exercise 2

1. (a)

State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	0	1	1	1
1	1	1	1	0	1	0
2	0	1	0	1	0	1
3	1	0	1	0	1	1
4	0	1	1	0	0	0

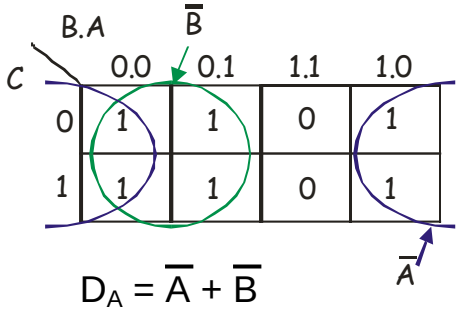
(b) $D_C = \overline{A}$

Here is the Karnaugh map for D_B:



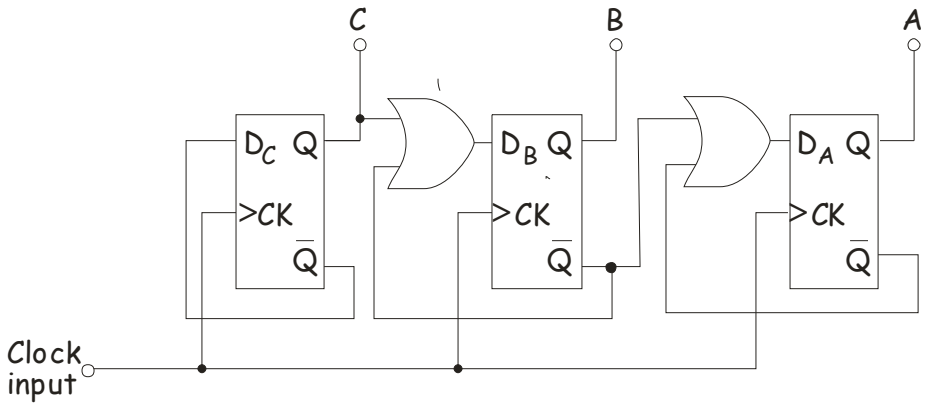
Hence $D_B = C + \overline{B}$

and for D_A:



giving the result $D_A = \overline{A} + \overline{B}$

(c) The circuit diagram for this system:



2. The truth table, first of all with 'Don't care' states:

State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	0	0	0	1
1	0	0	1	1	1	0
2	1	1	0	1	0	0
3	1	0	0	0	0	0
4	0	1	0	X	X	X
5	0	1	1	X	X	X
6	1	0	1	X	X	X
7	1	1	1	X	X	X

The following Boolean expressions work for the main sequence:

$D_C = A + B$

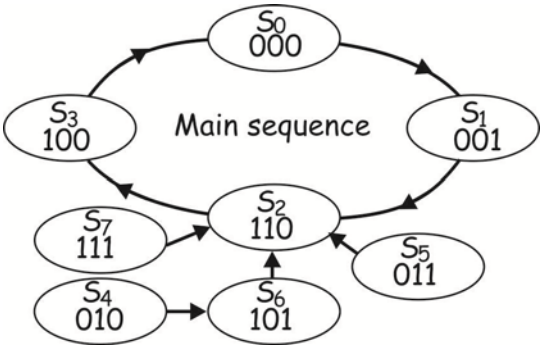
$D_B = A$

$D_A = \overline{A + C}$

Using these to complete the truth table:

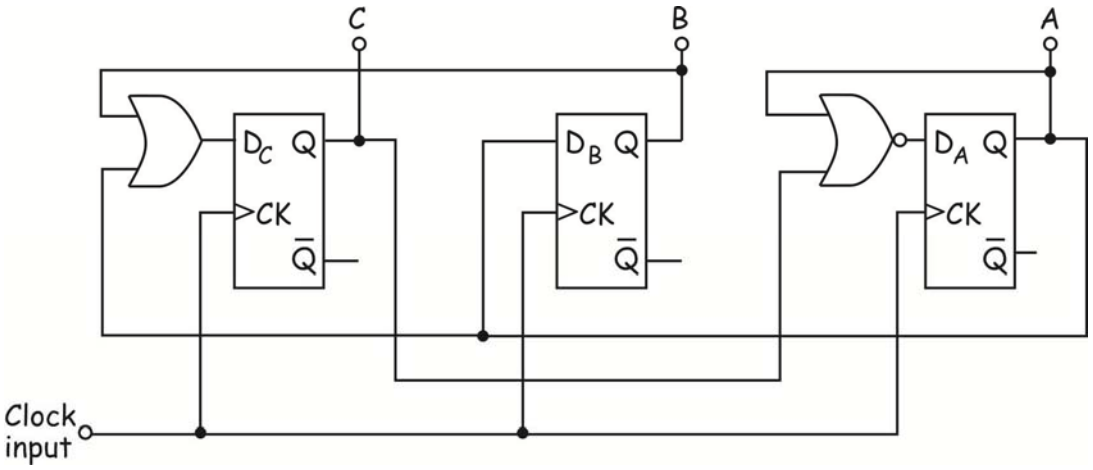
State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	0	0	0	1
1	0	0	1	1	1	0
2	1	1	0	1	0	0
3	1	0	0	0	0	0
4	0	1	0	1	0	1
5	0	1	1	1	1	0
6	1	0	1	1	1	0
7	1	1	1	1	1	0

Hence the state diagram:



Notice that there are no stuck states.

Finally, the circuit diagram:



3. The truth table with 'don't care' states:

State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	0	0	0
5	1	0	1	X	X	X
6	1	1	0	X	X	X
7	1	1	1	X	X	X

The main sequence gives rise to the following Boolean expressions:

$$D_C = B.A$$

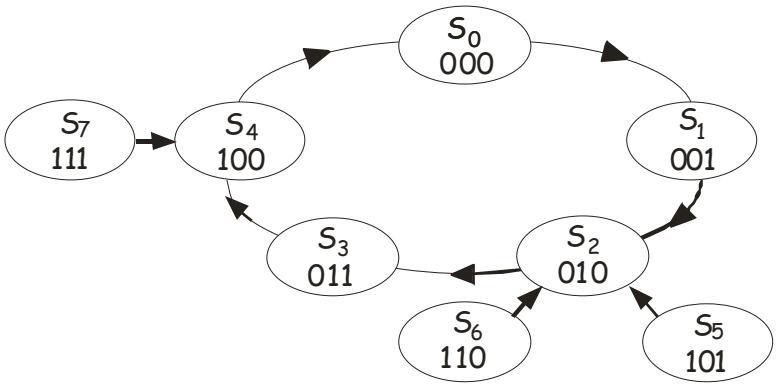
$$D_B = B \oplus A$$

$$D_A = \overline{C + A}$$

Using these, the complete truth table is:

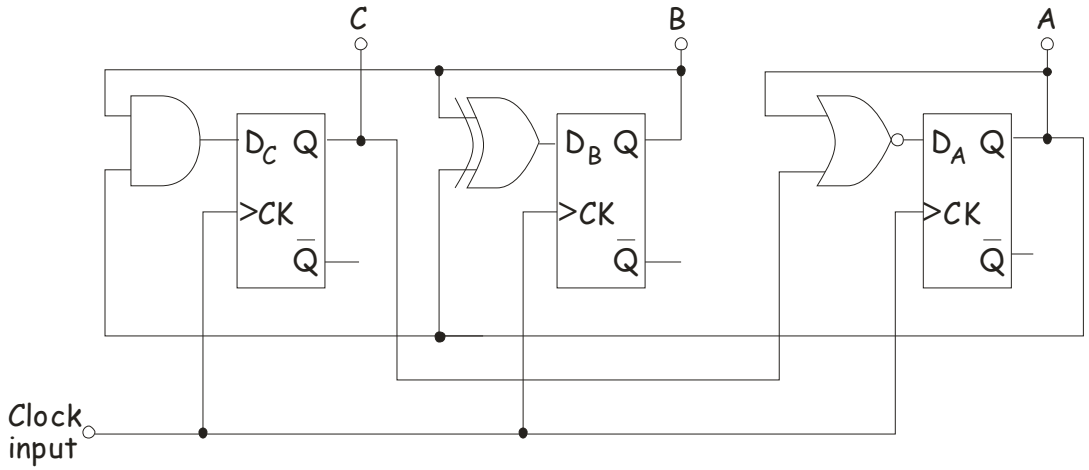
State number	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	0	0	0
5	1	0	1	0	1	0
6	1	1	0	0	1	0
7	1	1	1	1	0	0

The state diagram:



Again, notice that there are no stuck states.

The circuit diagram:



Exercise 3:

Boolean expressions:

$$D_C = \overline{A} \oplus B$$

$$D_B = \overline{A}$$

$$D_A = \overline{A + C}$$

Truth table:

State	Current state			Next state		
	C	B	A	D_C	D_B	D_A
0	1	0	0	1	1	0
1	1	1	0	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	0	0	0	1	1	1
5	1	1	1	1	0	0
6	0	0	1	0	0	0
7	1	0	1	0	0	0

State diagram: