Lecture 8

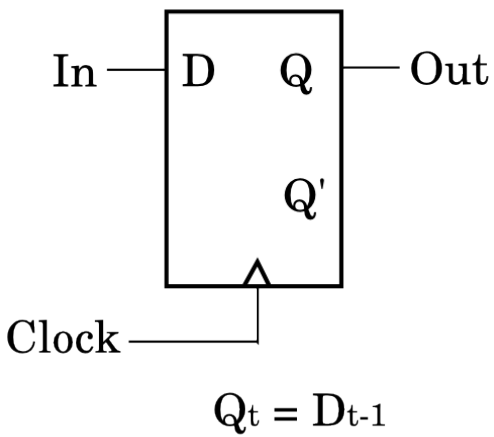
Sequential Digital Systems and Synchronous Counters

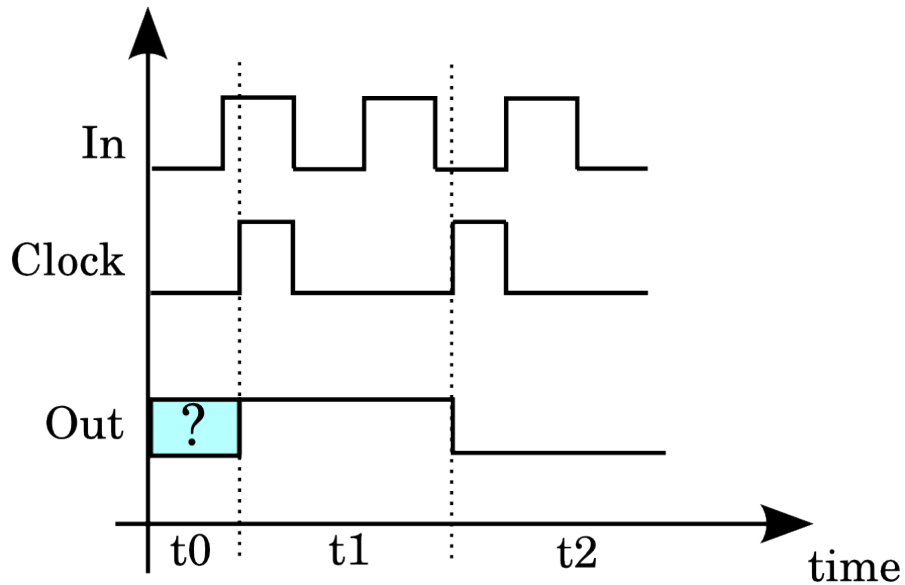
In this lecture we will:

- Introduce the main ideas of a Synchronous Digital System (SDS).
- Examine how simple flip-flops may be used to define a generic and completely general SDS.
- Examine binary counters in detail and how to design them.
- Show how to design a controlled counter which includes “don’t care” states.

The D-Q Flip-Flop

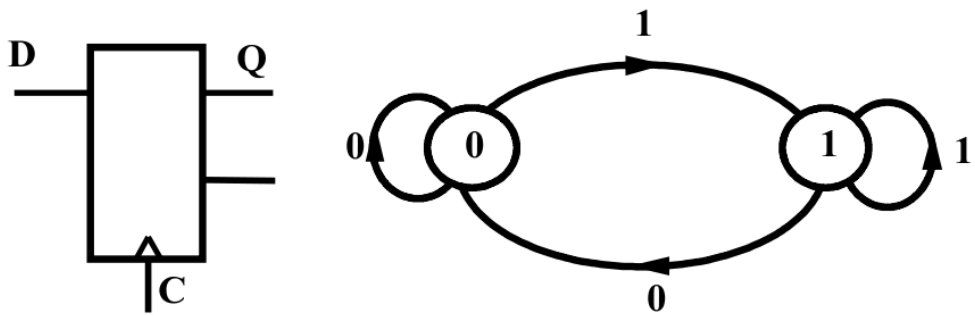
- Last lecture we saw that a flip flop is a one-bit memory
- It can be ”read” at any time
- It is ”written” when the clock input changes value
- The edge may be either rising or falling





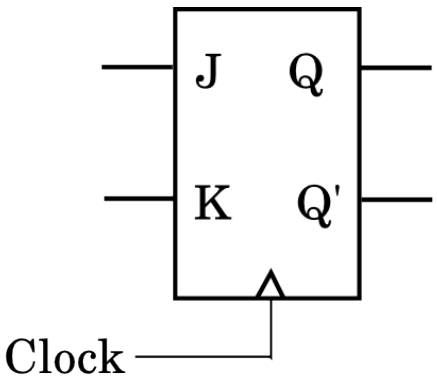
The D-Q Flip-Flop

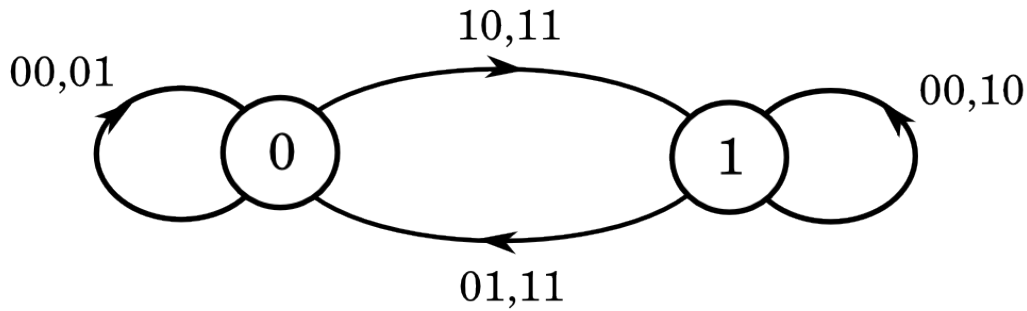
- The D-Q flip flop is a (very) simple example of a synchronous digital system.
- It can be in one of two possible states, and its change of state is controlled by a single clock signal.
- It can be represented by a finite state machine diagram.



The J-K flip-flop

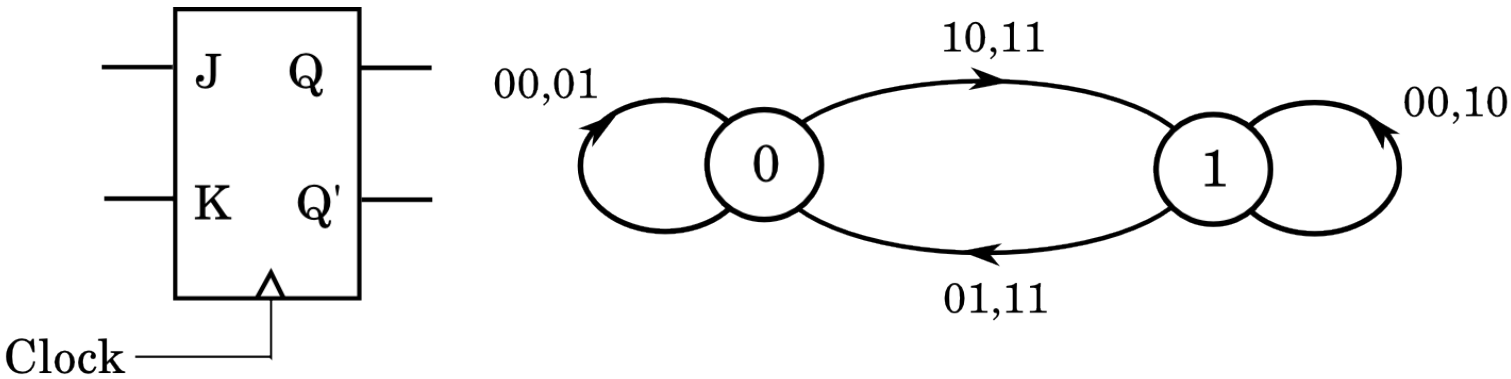
Digital designers also use the so called J-K flip-flop which has two data inputs (and of course, a clock input)





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 J-K flip-flop



- Even though the flip-flop has two inputs, it has only two states (not four).
- It has two inputs and four possible transitions

Sequential Digital Circuits

- All the different types of flip-flop are examples of sequential circuits. They go through a sequence of states controlled by the inputs.
- Another example of a sequential circuit is the counter in a digital watch. It will go through a sequence of states:

  display0 $\rightarrow$ display1 $\rightarrow$ display2 etc.

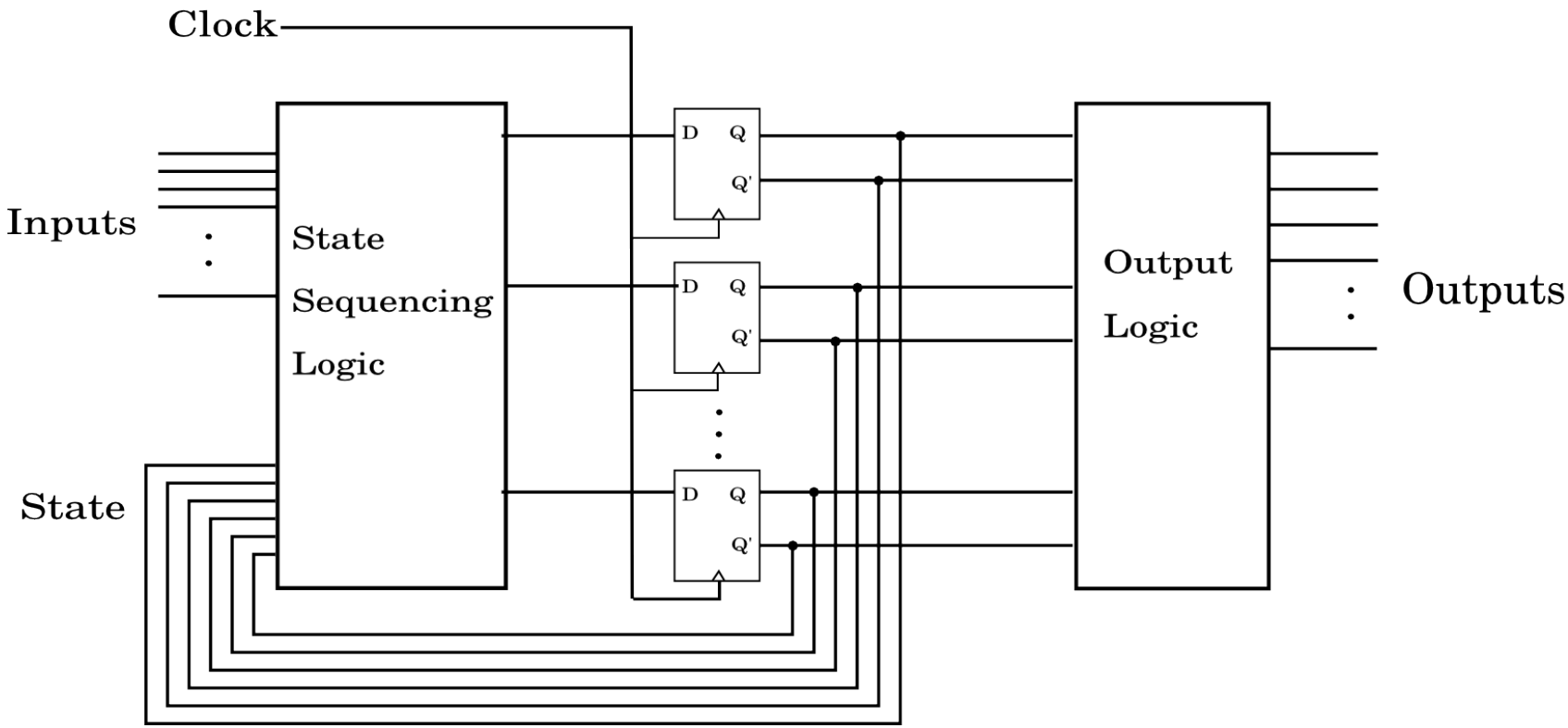
  States will change at exactly 1 second intervals

Synchronous Circuits

- Synchronous circuits are those where state changes occur at exact times controlled by a clock.
- Synchronous circuits are by definition sequential. Not all sequential circuits are synchronous.
- With the simple D type flip-flop and our knowledge of combinational digital circuits, we can construct a general model with which any synchronous circuit can be implemented!

Synchronous Digital Circuits

The general form of a synchronous digital circuit is:



Note that both the State Sequencing (Input) Logic and the Output Logic are combinatorial circuits.

Synchronous Digital Circuits

- The state of the circuit can be thought of as a binary number whose bits are stored on the D-Q flip-flops.
- The number of inputs and outputs depends on the circuit specification.
- The outputs depend only on the state the circuit is in.

$$O_j = G_j(Q_1, Q_2 \cdots Q_n)$$

- The next state depends on the current state and the circuit inputs

$$D_k = F_k(I_1, I_2 \cdots I_n, Q_1, Q_2 \cdots Q_n)$$

$$Q_k(t+1) = D_k(t)$$

Synchronous Binary Counters

- We will look at binary counters as an example of synchronous sequential circuit design.
- Simple binary counters have just a clock - no inputs.
- Their output is a repeating sequence of binary numbers. The state flip-flop outputs can be used directly as outputs (there is no output logic).

Synchronous Binary Counters

For a two-bit counter, there are four states:

0 (00), 1 (01), 2 (10), and 3 (11).

There are 6 complete counting sequences:

$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$ etc

$0 \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 0$ etc

$0 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 0$ etc

$0 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 0$ etc

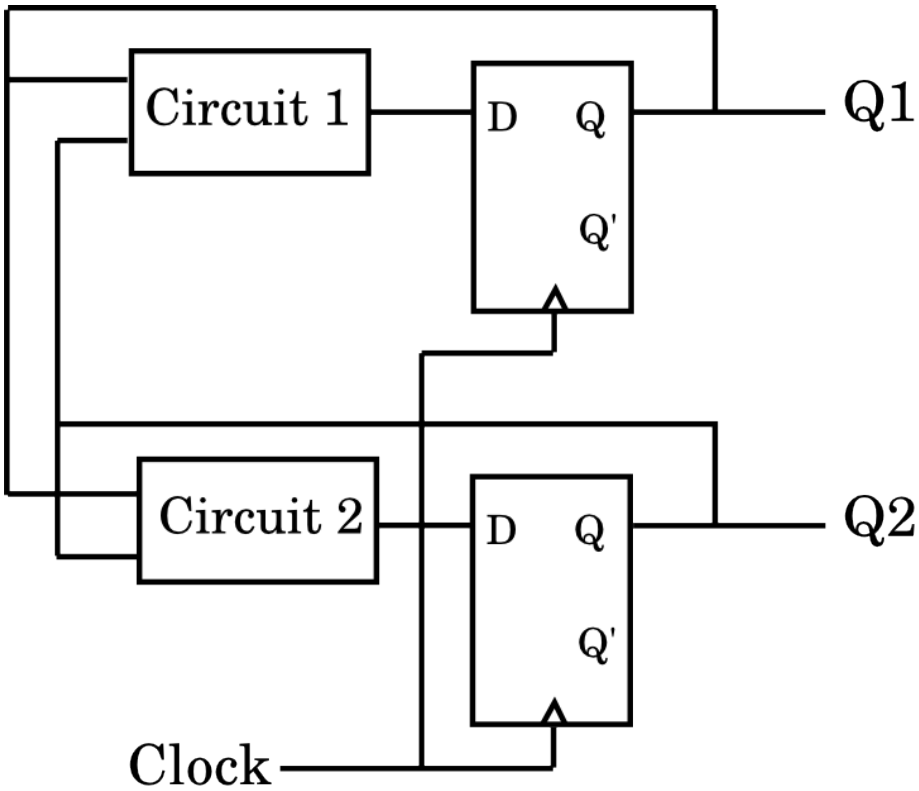
$0 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 0$ etc

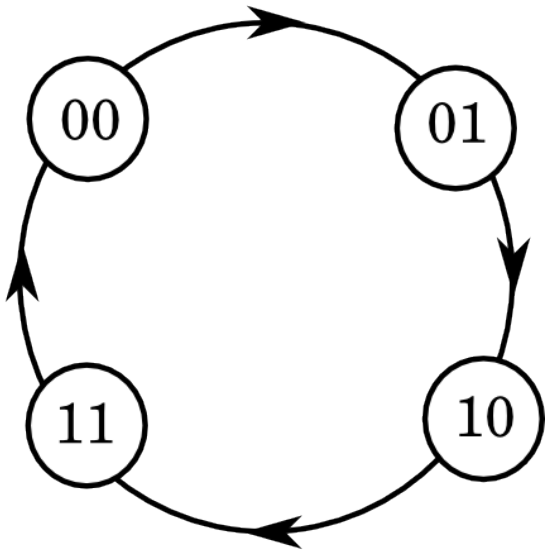
$0 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 0$ etc

And more if not all the states are used

A Two-Bit Binary Up Counter

The sequence is $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$ etc





We need to design "Circuit 1" and "Circuit 2"

A Two-Bit Binary Up Counter

- The first step is to construct the truth table of a synchronous circuit - its Transition Table.
- The transition table shows the state output values after the clock pulse (next) as a function of the input and state output values before the clock pulse (now).
- Since for a D type flip-flop the output (Q) after the clock pulse is equal to the input (D) before the clock pulse, the transition table becomes a simple input/output truth table.

A Two-Bit Binary Up Counter

The state transition table and Karnaugh Maps:

NOW		NEXT	
Q_1	Q_2	D_1	D_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Q1 \ Q2	0	1
0	0	1
1	1	0

D1 (Q1(next))

Q1 \ Q2	0	1
0	1	0
1	1	0

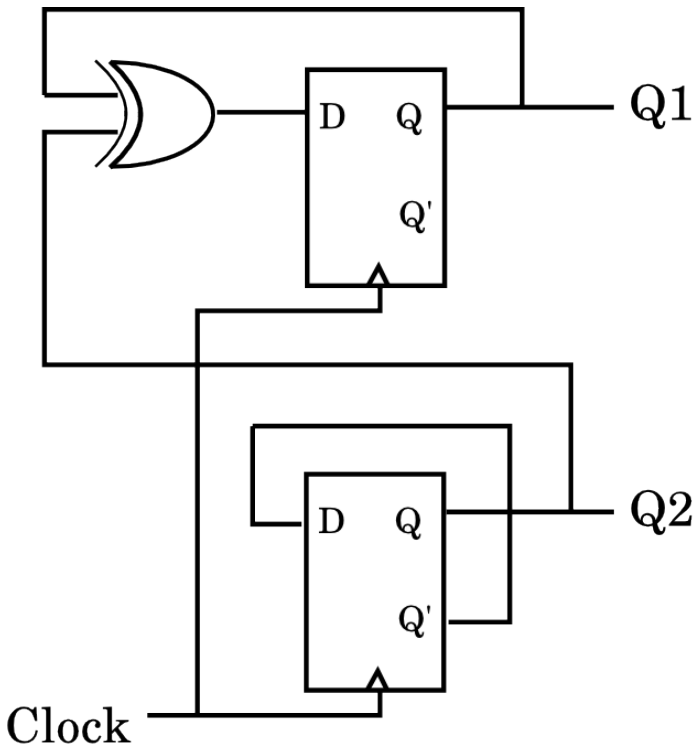
D2 (Q2(next))

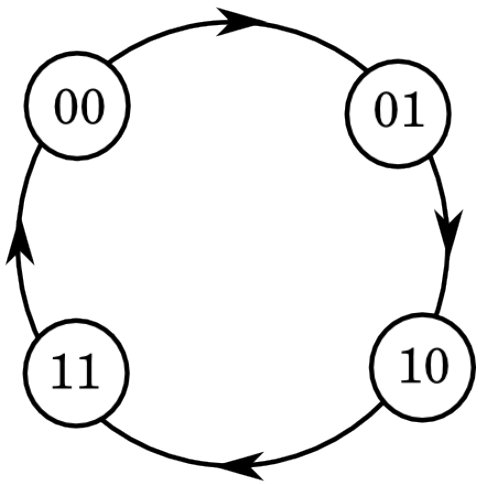
$$D_1 = Q_1' \bullet Q_2 + Q_1 \bullet Q_2' = Q_1 \oplus Q_2$$

$$D_2 = Q_2'$$

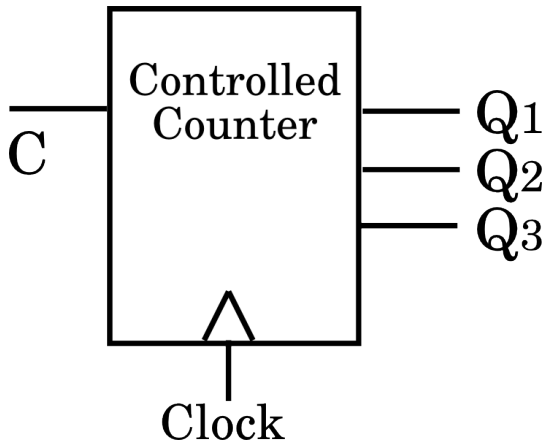
A Two-Bit Binary Up Counter

The final circuit:





A Three bit Controlled Counter



For the second example we will design a controlled three bit counter which has "don't care" states.

- When input C=0 the counter counts up even numbers: $000 \rightarrow 010 \rightarrow 100 \rightarrow 110 \rightarrow 000 \rightarrow$ etc
- When input C=1 the counter counts down odd numbers: $000- \rightarrow 111 \rightarrow 101 \rightarrow 011 \rightarrow 001 \rightarrow 000$

Problem Time

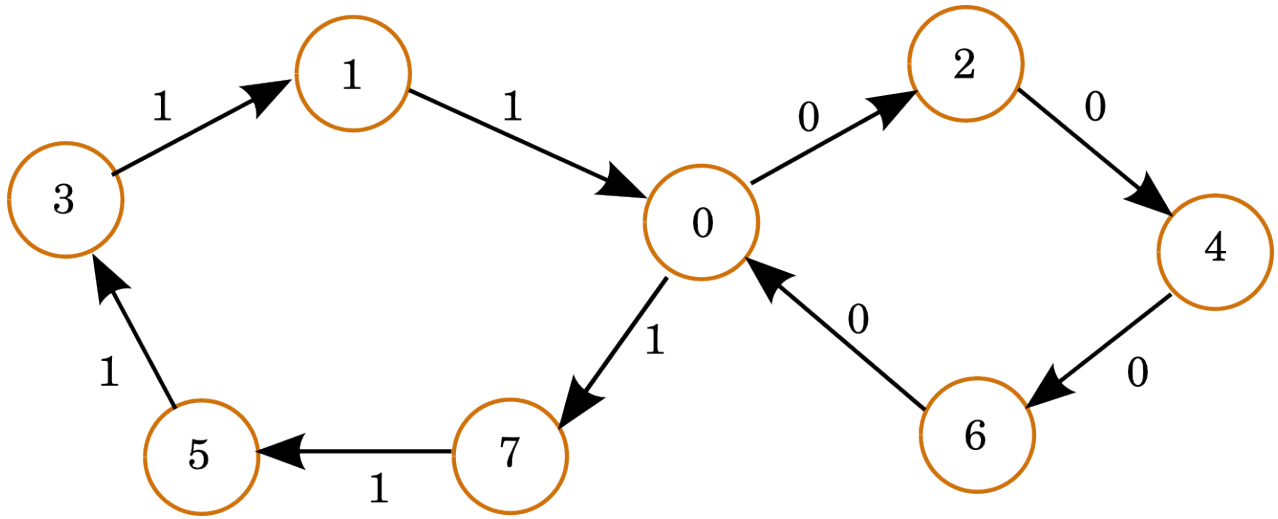
What does the state transition diagram look like?

- When input C=0 the counter counts up even numbers: $000 \rightarrow 010 \rightarrow 100 \rightarrow 110 \rightarrow 000 \rightarrow$ etc
- When input C=1 the counter counts down odd numbers: $000- \rightarrow 111 \rightarrow 101 \rightarrow 011 \rightarrow 001 \rightarrow 000$

Problem Time

What does the state transition diagram look like?

- When input C=0 the counter counts up even numbers: $000 \rightarrow 010 \rightarrow 100 \rightarrow 110 \rightarrow 000 \rightarrow$ etc
- When input C=1 the counter counts down odd numbers: $000- \rightarrow 111 \rightarrow 101 \rightarrow 011 \rightarrow 001 \rightarrow 000$



A Three bit Controlled Counter

In the specification we have state transitions that are unknown. We can deal with these in two different ways:

1. Design a safe circuit:
   We can specify the unknown state transitions ourselves to ensure that the circuit operates correctly. For example we could specify that in state 3 with input 0 the next state is 0.
2. Design the smallest possible circuit:
   We treat all unknown states and transitions as "don't cares" in order to have a good chance of getting the smallest possible design. However we then must check that the final circuit works correctly in all instances.

For the time being we will adopt the second more risky strategy.

A Three bit Controlled Counter

The transition table is generated from the finite state machine diagram.

The unspecified states are marked by X to indicate "don't care" values.

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

A Three bit Controlled Counter

Karnaugh Map Minimisation

D1

C, Q1 \ Q2, Q3	00	01	11	10
00	0	X	X	1
01	1	X	X	0
11	X	0	1	X
10	1	0	0	X

$$D_1 = C' \cdot Q_1' \cdot Q_2 + C' \cdot Q_1 \cdot Q_2' + C \cdot Q_3' + C \cdot Q_1 \cdot Q_2$$

D2

C, Q1 \ Q2, Q3	00	01	11	10
00	1	X	X	0
01	1	X	X	0
11	X	1	0	X
10	1	0	0	X

$$D_2 = Q_2' \cdot Q_3' + Q_1 \cdot Q_2'$$

D3

C, Q1 \ Q2, Q3	00	01	11	10
00	0	X	X	0
01	0	X	X	0
11	X	1	1	X
10	1	0	1	X

$$D_3 = C \cdot Q_2 + C \cdot Q_3' + C \cdot Q_1$$

What happens to the Don't Care values?

When we draw circles on a Karnaugh map we specify what values the "don't cares" will take in the implementation:

1. If a "don't care" is inside any circle it will take the value 1.
2. If a "don't care" is outside all the circles it will take the value 0.

Using these two rules we can determine what will happen in our final implementation.

A Three bit Controlled Counter

The behaviour of the circuit designed:

D1

C, Q1 \ Q2, Q3	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	0	1	1
10	1	0	0	1

$$D_1 = C' \cdot Q_1' \cdot Q_2 + C' \cdot Q_1 \cdot Q_2' + C \cdot Q_3' + C \cdot Q_1 \cdot Q_2$$

D2

C, Q1 \ Q2, Q3	00	01	11	10
00	1	0	0	0
01	1	1	0	0
11	1	1	0	0
10	1	0	0	0

$$D_2 = Q_2' \cdot Q_3' + Q_1 \cdot Q_2'$$

D3

C, Q1 \ Q2, Q3	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	0	1	1

$$D_3 = C \cdot Q_2 + C \cdot Q_3' + C \cdot Q_1$$

A Three bit Controlled Counter

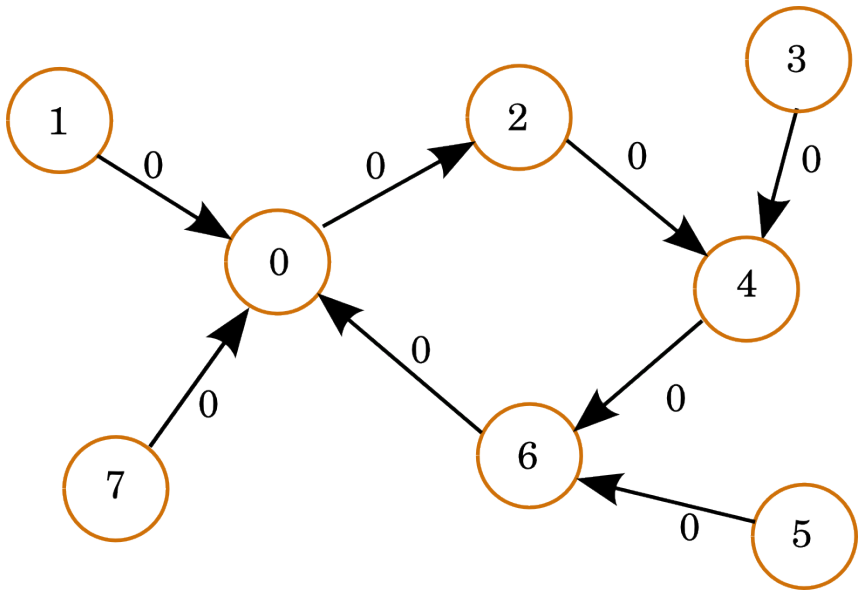
The don't care values are copied back into the transition table.

The unspecified state transitions can now be found.

C	Q_1	Q_2	Q_3	D_1	D_2	D_3	S_n	S_{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

A Three bit Controlled Counter

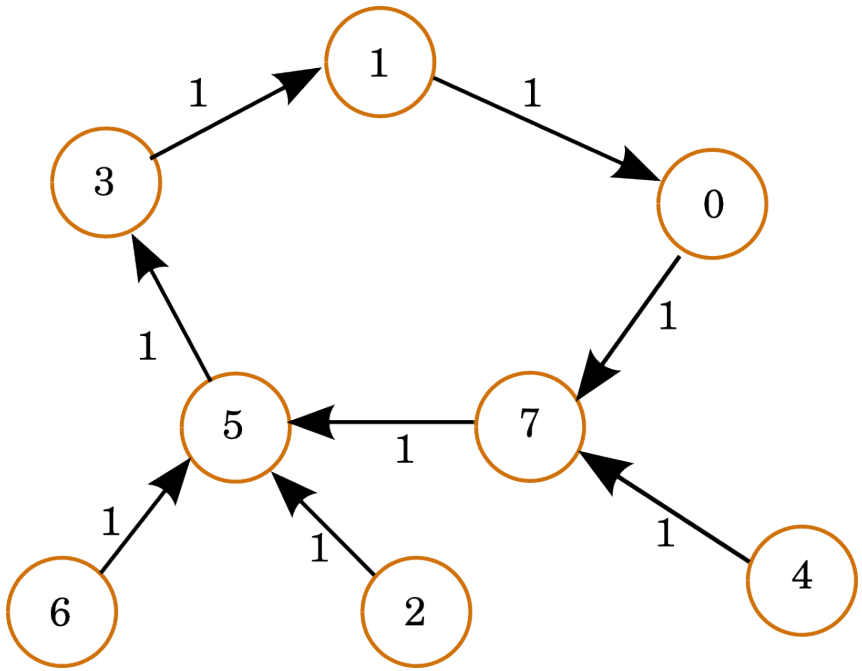
The final finite state machine diagram for C=0:



C	S_n (Now)	S_{n+1} (Next)
0	0	2
0	1	0
0	2	4
0	3	4
0	4	6
0	5	6
0	6	0
0	7	0

A Three bit Controlled Counter

The final finite state machine diagram for C=1:



C	S_n (Now)	S_{n+1} (Next)
1	0	7
1	1	0
1	2	5
1	3	1
1	4	7
1	5	3
1	6	5
1	7	5

A Three bit Controlled Counter

Finally we build the circuit from the Boolean equations found during the design:

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

Note that common terms only need to be committed to hardware once!

A Three bit Controlled Counter - Final Circuit