Lecture 9

Finite State machine representation of sequential circuits

The story so far:

All circuits can be represented by Boolean equations and vice versa

Sequential circuits, in which the outputs are fed back into the inputs have variables that appear on both the right and left hand side of the variables: eg

$$Q = (R \bullet P)' \quad \text{and} \quad P = (S \bullet Q)'$$

For one case R=S=1 there is no unique solution

The story so far:

For sequential circuits an interpretation of the Boolean equations cannot be made without some model of time.

A simple time delay model gives us the correct interpretation of the R-S flip flop.

However, we'd like not to worry about time if possible.

The story so far:

We can avoid worrying about time by using synchronous digital circuits

In a synchronous circuit a bank of D-type flip flops acts as a barrier between the inputs and the outputs.

The flip flops are all triggered at the same time using a common clock.

The story so far:

The equation pair:

$$R = A \cdot B + P \qquad P = (A+B)'$$

is taken to mean

$$R_t = A_{t-1} \cdot B_{t-1} + P_{t-1} \qquad P_t = (A_{t-1}+B_{t-1})'$$

thus given initial conditions we can always solve the equations

The story so far:

If we use synchronous design the only question that we need worry about is how fast the clock can go.

This will depend on how well we can minimise spikes in the combinational logic circuits.

Now read on!

Finite State Machines

In general any synchronous circuit can be modelled by a finite state machine, and vice versa.

A finite state machine in its most general form can be represented by a pair of equations:

S(t+1) = f(S(t), I(t))

O(t+1) = g(S(t), I(t))

Where S(t) is the state at time t and I(t) the input at time t, and O(t) is the output at time t.
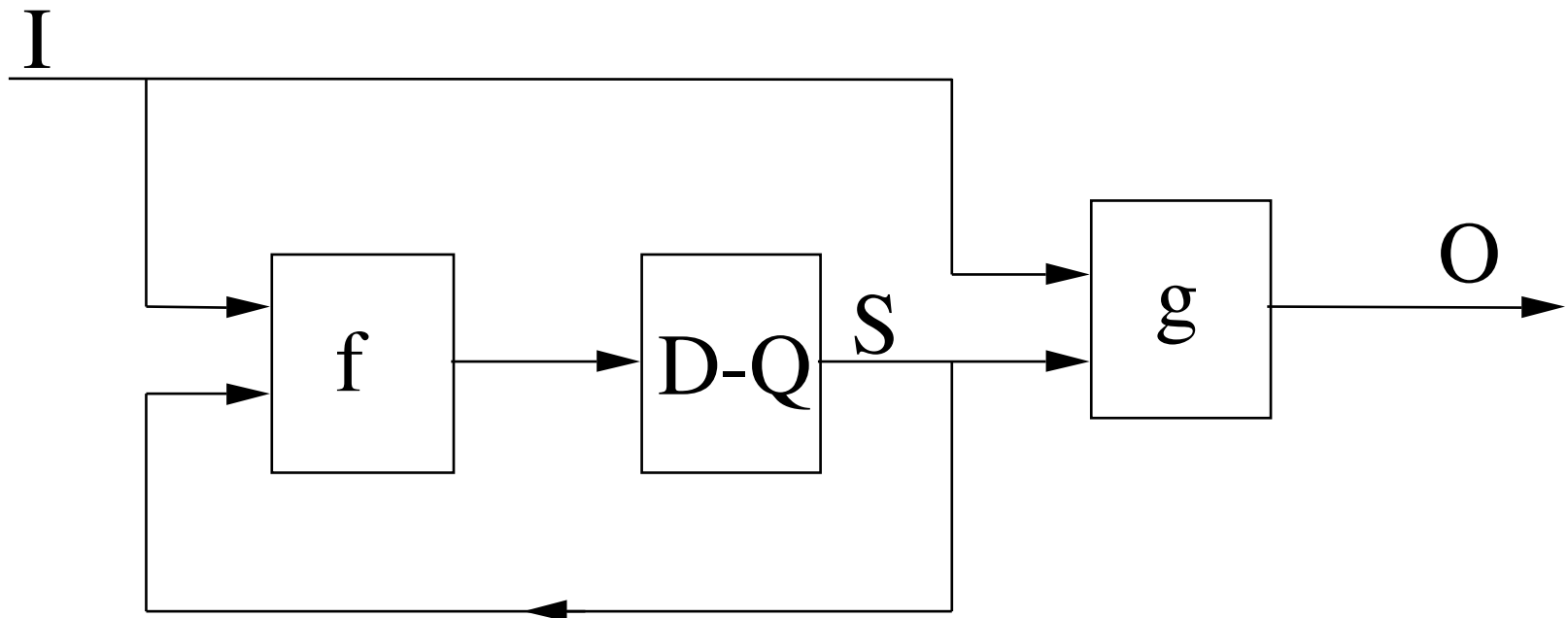
States S(t)

S(t) can be one of a finite number of states

We noted that the flip-flop has only two states defined by Q=0 and Q=1

In a digital circuit state changes are caused by a clock pulse given to the D-Q flip flop(s)
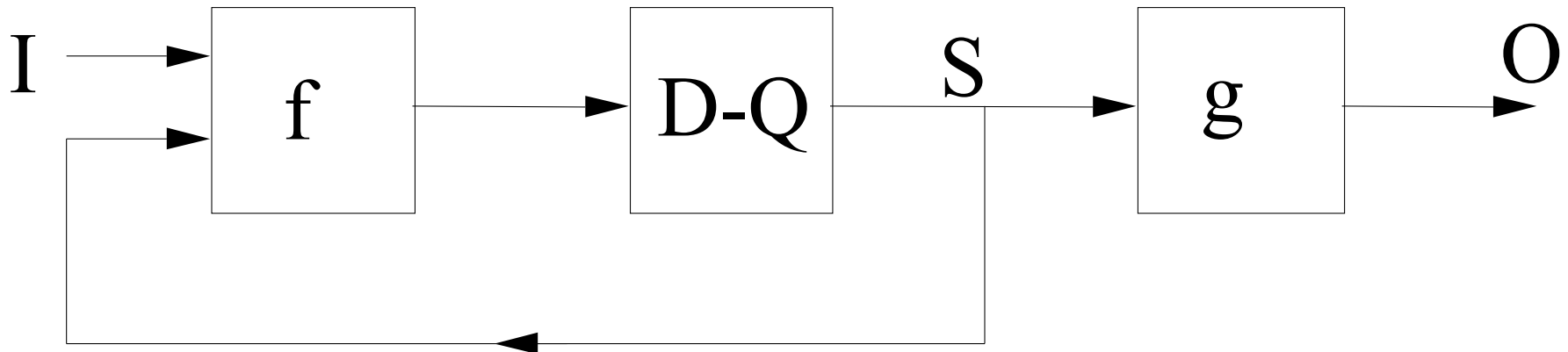
The Mealey machine

One form of finite state machine is called the Mealey machine, and is represented thus:

The Moore Machine

The Moore machine (seen last lecture) is similar but does not have a connection between the inputs and the output logic
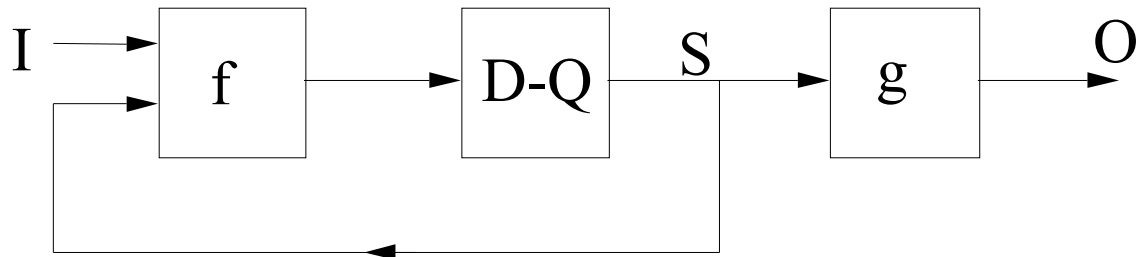
The Moore Machine 2

f is a combinational circuit which is called the state sequencing (or input) logic

g is a combinational circuit which is called the output (or decode) logic

The D-Q may be one or many flip flops, and their outputs represent the state

I → | f | → | D-Q | —S→ | g | → O

Why use finite state machines

The finite state machines give us a way specifying exactly what a circuit should do.

Having designed the finite state machine we can apply a design methodology to create the actual circuit.

FSMs play a crucial role in computer science as a modelling tool.

Sequential Circuit Design Methodology

1. Determine the number of states required

2. Determine the state transitions

3. Choose the way in which the states will be represented

4. Express the state sequencing logic as Boolean equations and minimise using Karnaugh Maps

5. Express the output logic as Boolean Equations and minimise

A design problem

1

Sequential Counter

0
1
2
3
4
5
6
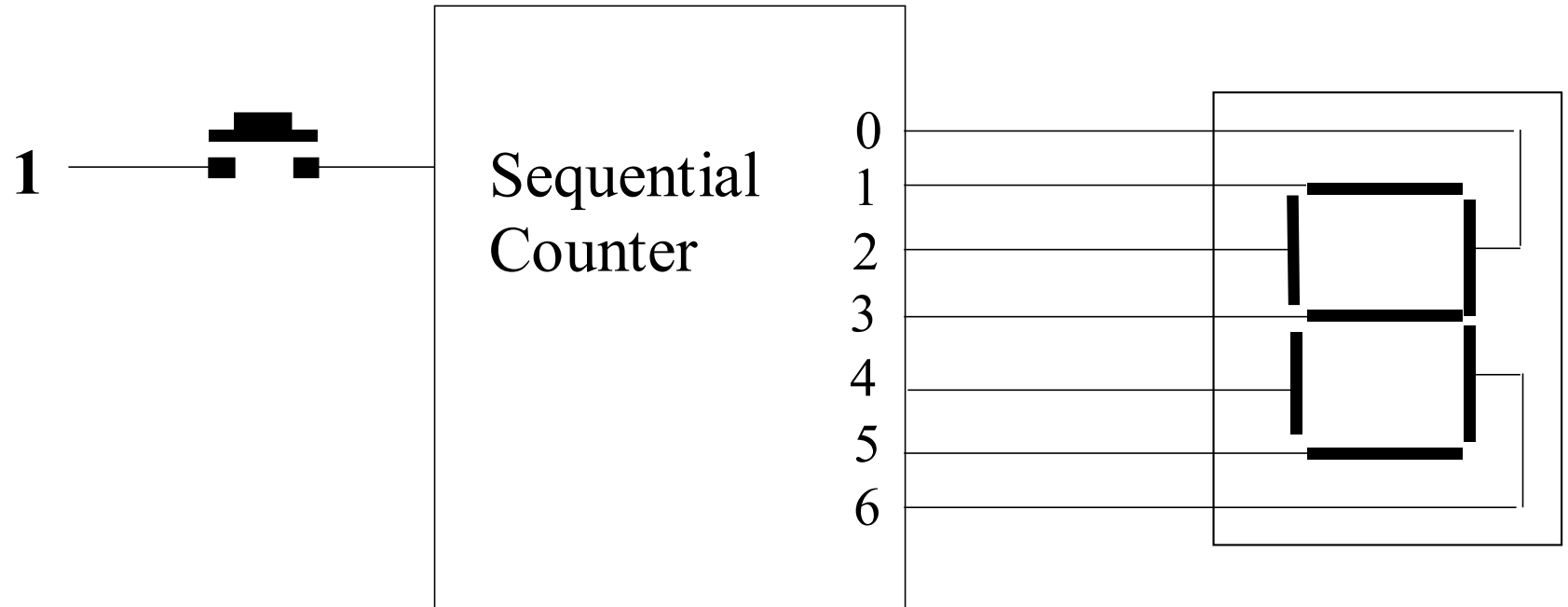
Determining the number of states

We will consider the simple case where the counter counts from 0 to 5 before returning to 0.

(as might be found in a digital clock)

Clearly we will need six states being:

"0" displayed

"1" dispalyed

etc

Specifying the states and output logic

We can give the states names, and specify what output we want from the circuit:

State	Description	Output
0	0 displayed	$A = \{1,1,1,0,1,1,1\}$
1	1 displayed	$B = \{1,0,0,0,0,0,1\}$
etc		

We now can draw the states as circles

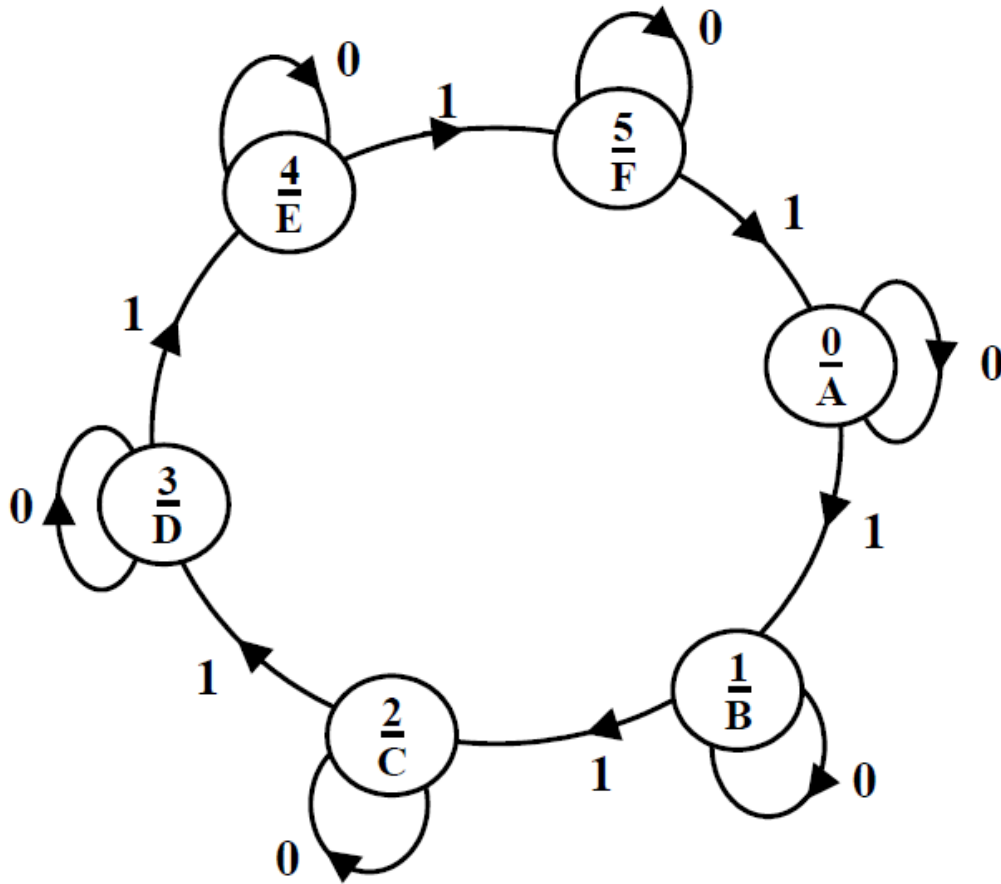As we are designing a Moore machine, we write both the name and the outputs in the state circles:

$\frac{5}{F}$

$\frac{4}{E}$

$\frac{0}{A}$

$\frac{3}{D}$

$\frac{1}{B}$

$\frac{2}{C}$
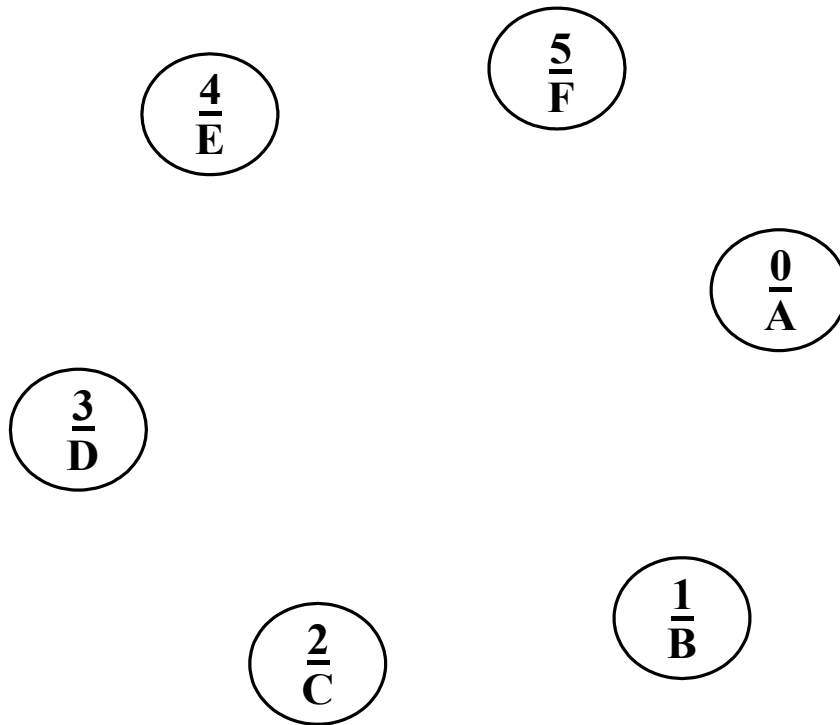
Output Vectors

A = {1,1,1,0,1,1,1}
B = {1,0,0,0,0,0,1}
etc.

Next we specify the state transitions

If the input is 0, stay in the same state.

If the input is 1, go on to the next state.

State representation

We now have to decide how we are to represent each state:

One flip flop per state would be an expensive possibility.

Three flip flops can be though of as a three bit memory, and therefore could represent eight states.

The values Q2,Q1 and Q0 define the state.

The State Assignment Problem

Three flip flops can represent eight binary numbers, but we must now choose which six of the eight possibilities we use to represent our states. We might choose:

Flip Flop Outputs	State
0 0 0	0
0 0 1	5
0 1 0	unused
0 1 1	4
1 0 0	3
1 0 1	unused
1 1 0	1
1 1 1	2

Problem Break

If we wanted our counter to count from 0 to 9 rather than from 0 to 6 how many states would it need?
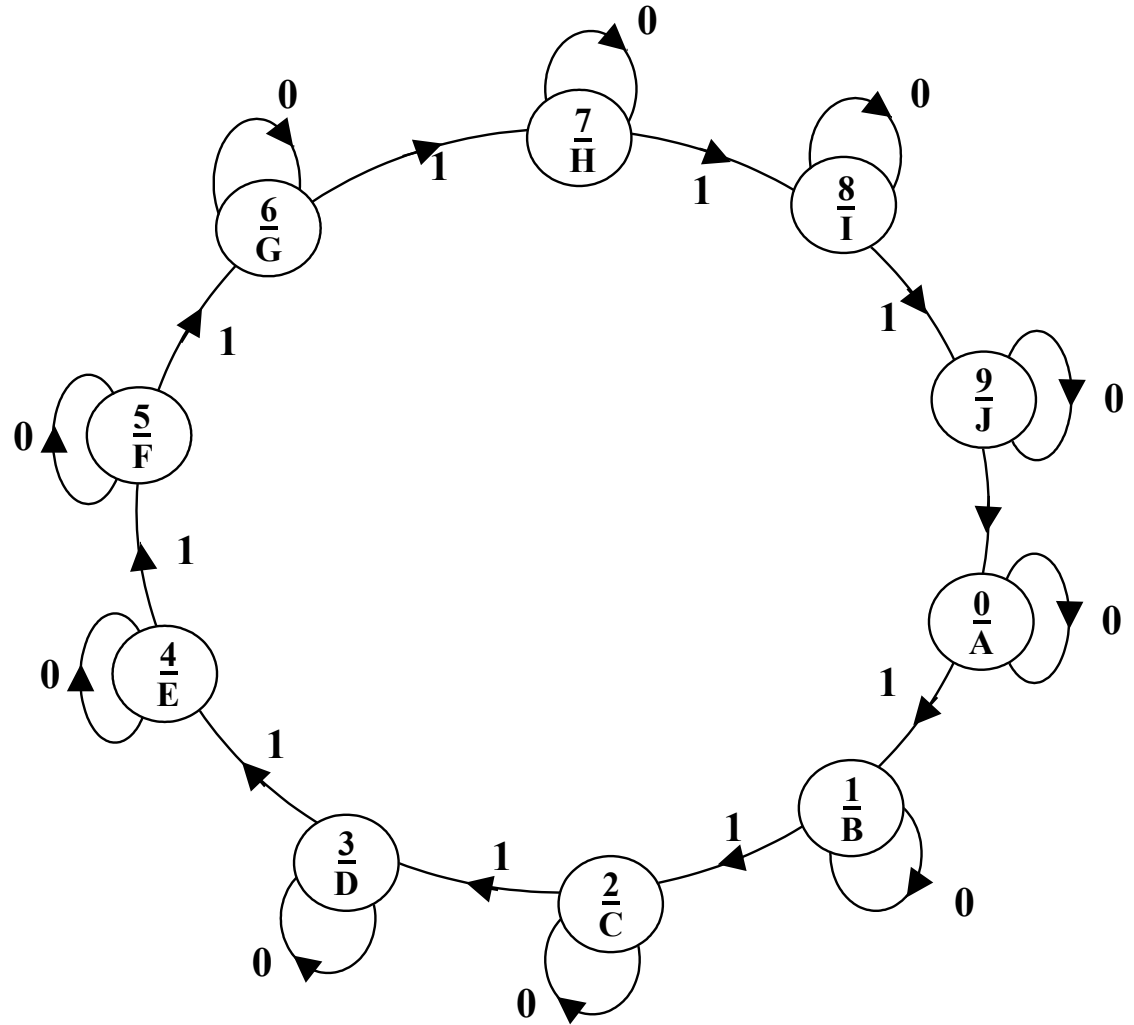
What would the finite state machine look like?

How many flip flops would we need to represent the states?

Problem Break

10 States

4 flip flops

Choose a state assignment

Let's make the obvious choice and allocate flip flop outputs to the states as follows:

Flip Flop Outputs	State	Minterm
0 0 0	0	Q2'•Q1'•Q0'
0 0 1	1	Q2'•Q1'•Q0
0 1 0	2	Q2'•Q1•Q0'
0 1 1	3	Q2'•Q1•Q0
1 0 0	4	Q2•Q1'•Q0'
1 0 1	5	Q2•Q1'•Q0
1 1 0	unused	
1 1 1	unused	

States and Minterms

Each state can now be represented by a Boolean Equation. The variable Si is 1 when the circuit is in state i, otherwise it is 0.

$S_0 = Q2' \cdot Q1' \cdot Q0'$

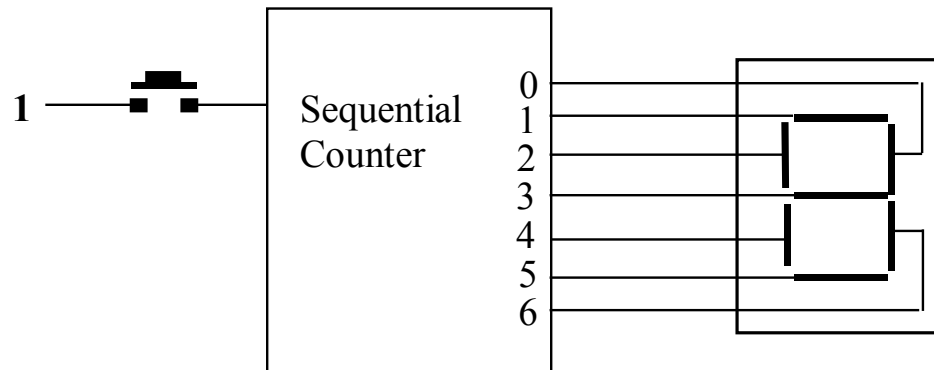$S_1 = Q2' \cdot Q1' \cdot Q0$

$S_2 = Q2' \cdot Q1 \cdot Q0'$

&c.

Output Logic

Our output logic is now completely specified, for example we want the segment 1 to be lit when displaying 0,2,3,5 but not 1 or 4, so:

O1 = S0 + S2 + S3 + S5

O1 = Q2'•Q1'•Q0' + Q2'•Q1'•Q0 + Q2'•Q1•Q0 + Q2•Q1'•Q0

Designing the state sequencing logic

We obtain the specification of the state sequencing logic from the finite state machine representation.

Notation N0 is read as "The next state is S0"

So we can write for each state an equation of the form:

$$N0 = S5 \cdot I + S0 \cdot I'$$

$$N1 = S0 \cdot I + S1 \cdot I'$$

Finding the D values

The next state must be form the D inputs of the three flip flops, so that it is transferred to the Q outputs on the next clock pulse.

We can find Boolean equations for the D inputs by looking at the state allocation table.

Q2 is 1 when the state is 4 or 5 thus:

$D2 = N4 + N5$

$D1 = N2 + N3$

$D0 = N1 + N3 + N5$

Q2 Q1 Q0	State
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5

Substitute for the variables Ni

D1 = N2 + N3 = S1•I + S2•I' + S2•I + S3•I'

Substitute for the Si variables

D1 = Q2'•Q1'•Q0•I + Q2'•Q1•Q0'•I' +
　　　　　　Q2'•Q1•Q0'•I + Q2'•Q1•Q0•I'

This is the boolean equation for D1, and D2 and
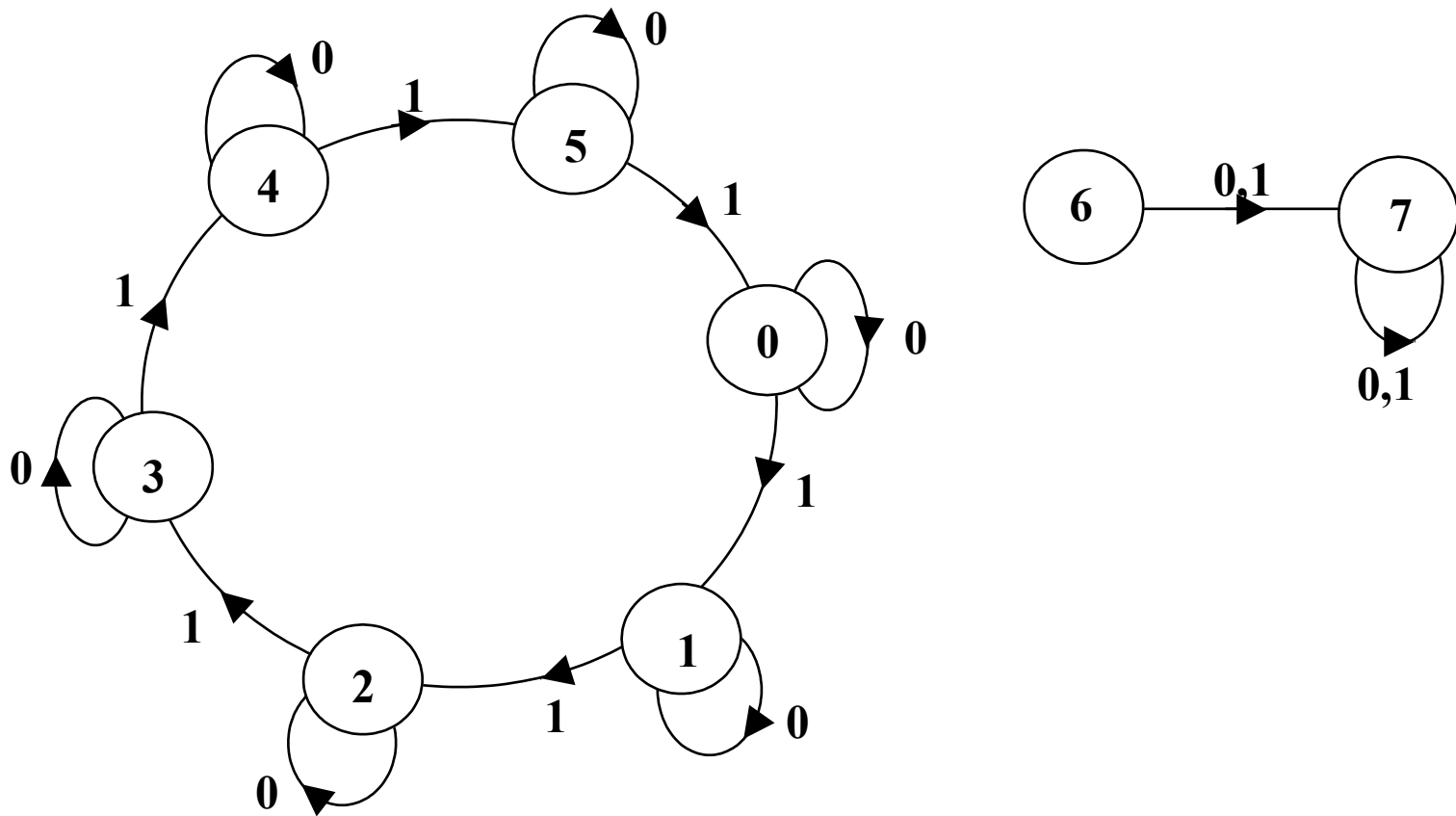　D0 can be found similarly

Unused States

The unused states were designated don't care states.

To get the minimum representation we will need to include them as don't cares in our Karnaugh maps.

This can cause problems since we will allocate the don't cares real values in the final design.

A possible (disasterous) outcome of using don't cares

Dealing with unused states

If we use don't cares in our design (and we should do so as a first try) we must check what happens to the unused states.

If they form an isolated machine, we can either:

See if a simple modification will fix the problem without increasing the size too much

Explicitly include the unused states in our finite state machine, and re-do the design

Explicitly Allocated States
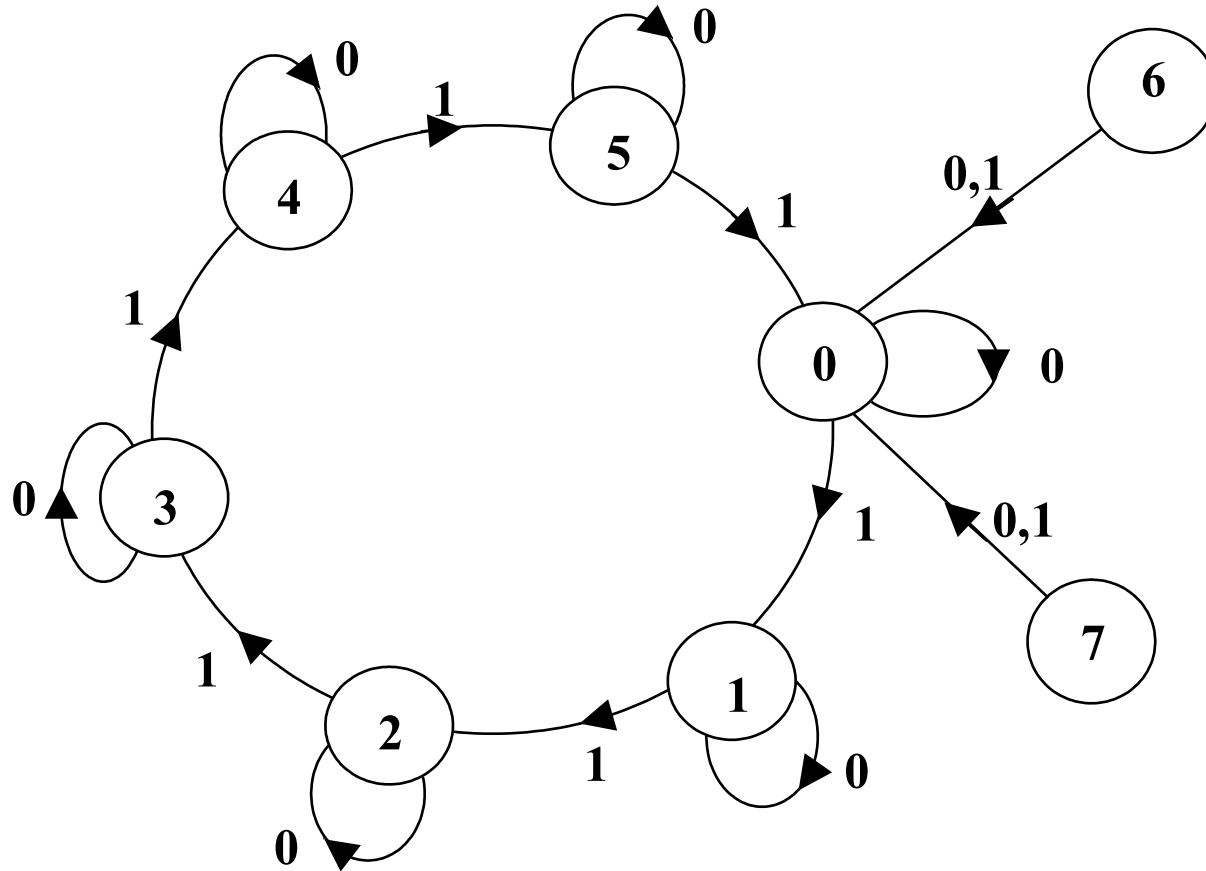
Choosing the best state assignment

If the number of states is small, then we can try all possible allocations

The number of ways we could allocate the eight possibilities to the six states is 56

Fortunately we can eliminate many of these

Isomorphic Assignments

Since flip-flops always have complementary outputs (Q and Q') The same circuit will result from complementing each column of our assignment

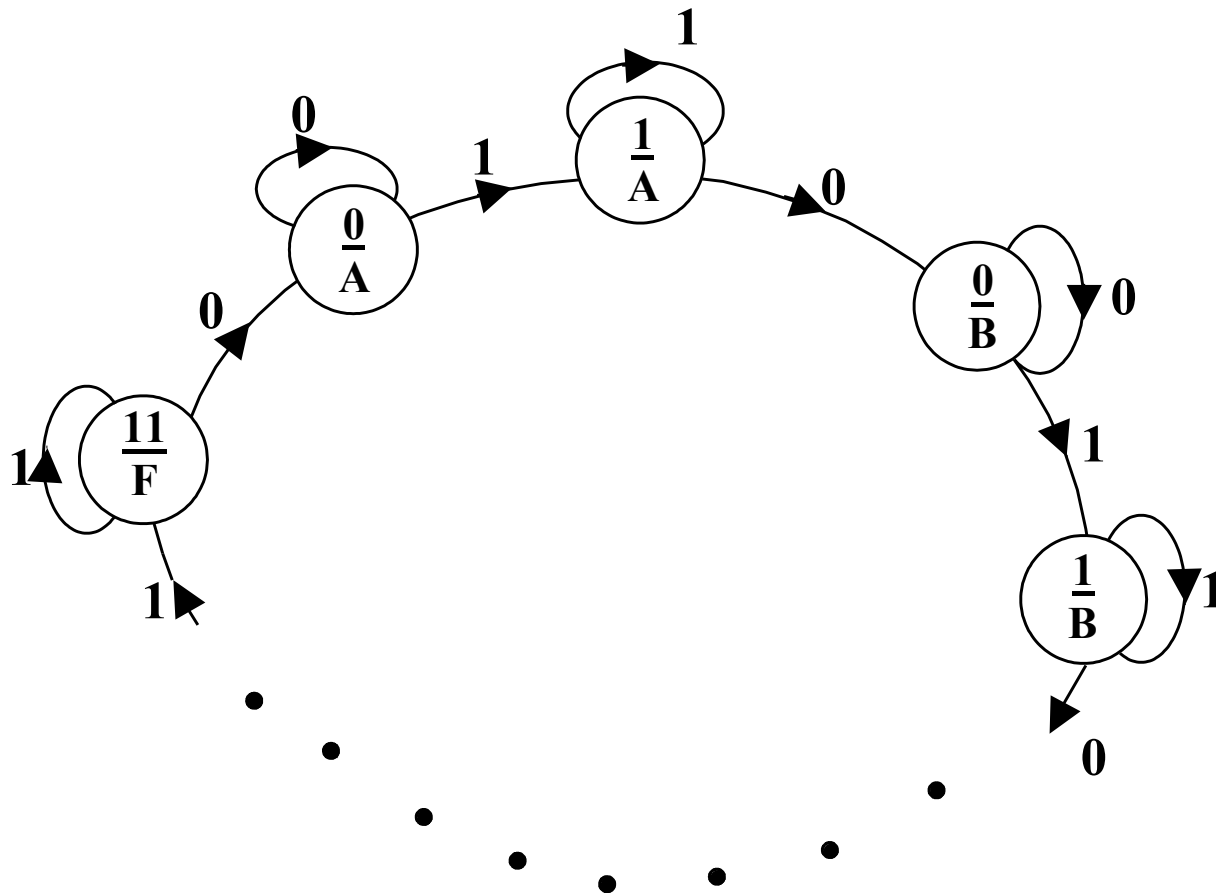| State	Assignment	Isomorphs
0	000	100 010 110
1	001	101 011 111
2	010	110 000 100
3	011	111 001 101
4	100	000 110 010
5	101	001 111 011

These isomorphs are created by:

Invert Column 1
Invert Column 2
Invert Columns 1 and 2

Making an edge triggered counter

# State Allocation for the edge triggered circuit

This circuit has now 12 states and we will need to use 4 flip flops to represent them.

The number of possible state assignments goes up to 1820, so exhaustive search is not a possibility.

Heuristic Rules for State Allocation

There are two rules that should give good Karnaugh map minimisations for the sequencing logic.

First, all those states that have the same next state for the same input should be given adjacent state assignments
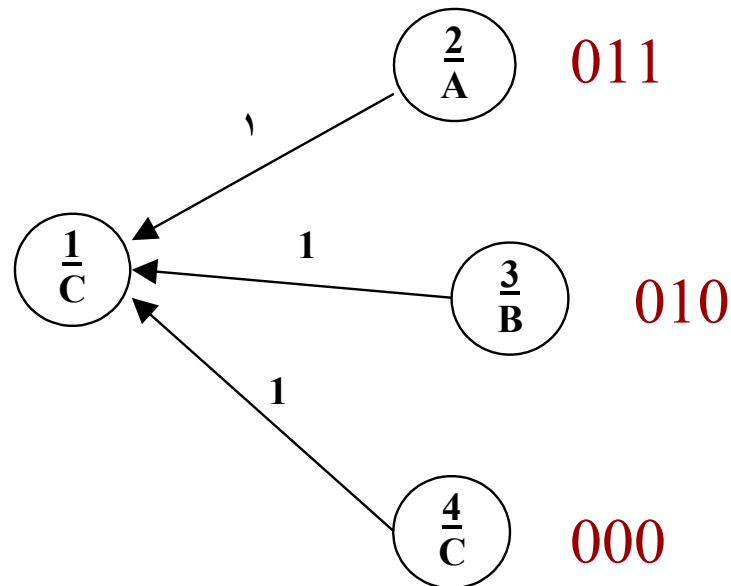
Heuristic Rules for State Allocation

Second, the next states of a state produced by applying adjacent input conditions should be given adjacent state assignments.