Spring 2015 Week 6 Module 30

Digital Circuits and Systems

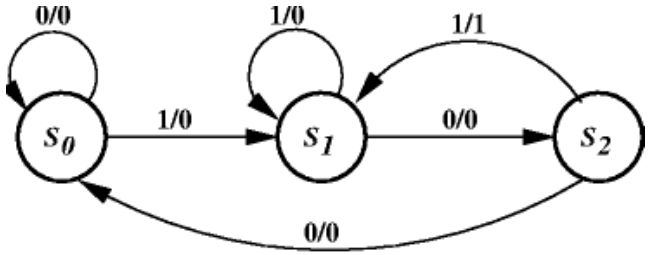
State Machines 2: Design Problems

*Shankar Balachandran**
Associate Professor, CSE Department
Indian Institute of Technology Madras

*Currently a Visiting Professor at IIT Bombay

Example 1:

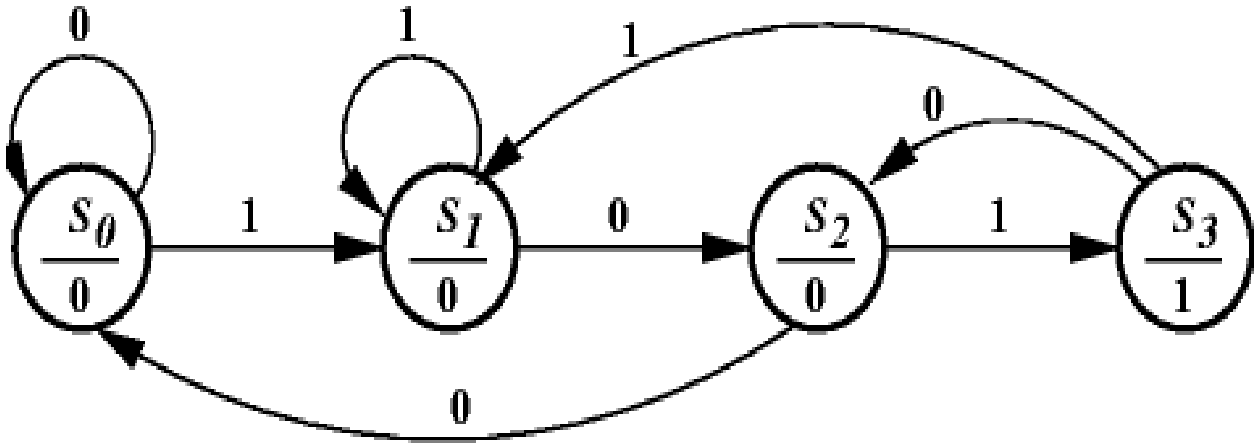
- Design a sequential circuit that produces “1” on its output if it detects the sequence “101” on it’s input. The detector should keep checking for the appropriate sequence and should not reset to the initial state after it has recognized the sequence.



P.S. Q_1Q_0	N.S. ($Q_1^*Q_0^*$)		Out (z)	
	x=0	x=1	x=0	x=1
s_0 (00)	s_0 (00)	s_1 (01)	0	0
s_1 (01)	s_2 (10)	s_1 (01)	0	0
s_2 (10)	s_0 (00)	s_1 (01)	0	1

$$D_1 = \bar{x}Q_0$$
$$D_0 = x$$
$$z = xQ_1$$

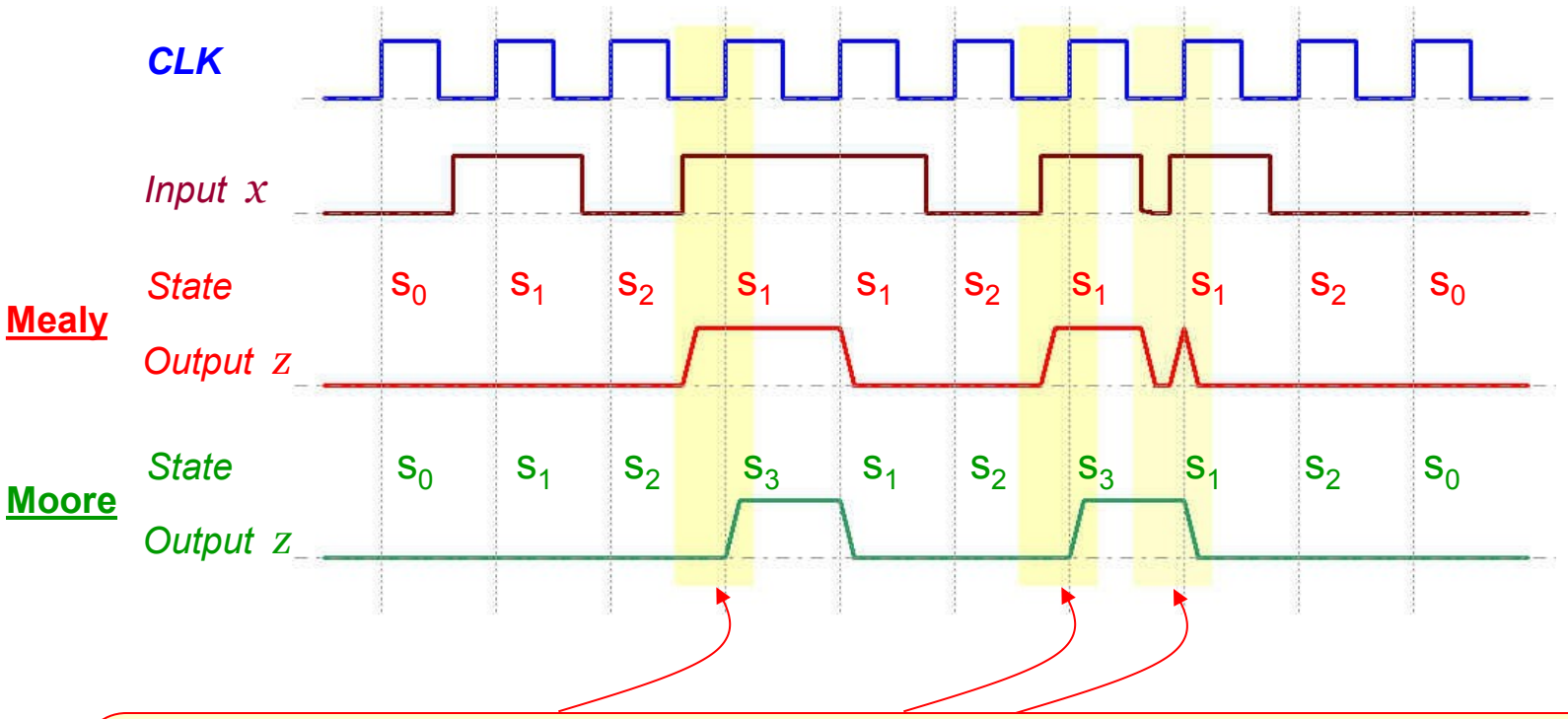
Moore Style



P.S. Q_1Q_0	N.S. ($Q_1^*Q_0^*$)		Out (z)
	x=0	x=1	
s_0 (00)	s_0 (00)	s_1 (01)	0
s_1 (01)	s_2 (10)	s_1 (01)	0
s_2 (10)	s_0 (00)	s_3 (11)	0
s_3 (11)	s_2 (10)	s_1 (01)	1

$$D_1 = \bar{x} Q_0 + x Q_1 \overline{Q_0}$$
$$D_0 = x$$
$$z = Q_1 Q_0$$

Example (contd.): Output Waveforms



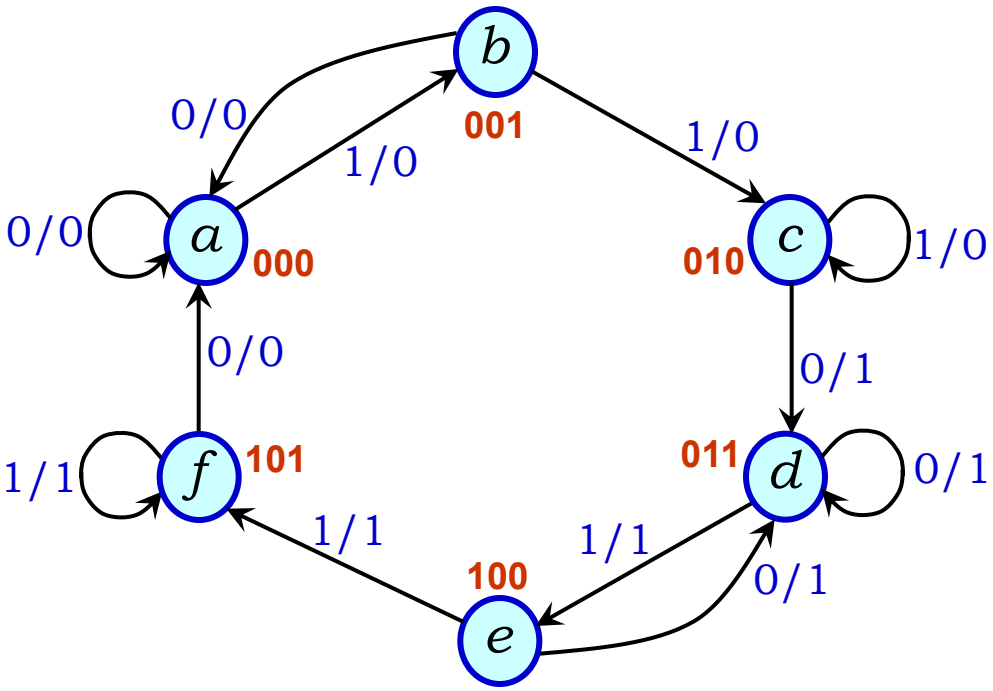
Mealy outputs may change when an input changes (i.e., not necessarily on a clock edge).
⇨output may have glitches. This problem can be solved by making Mealy inputs synchronous.

Moore outputs only change on clock edges since they depend only on the present state.
⇨Moore outputs may be delayed w.r.t. the corresponding outputs in a Mealy implementation.

Example 2:

A sequential circuit has one input and one output. When input sequence "110" occurs the output becomes 1 and remains 1 until the sequence "110" occurs again in which case the output returns to 0. The output remains 0 until "110" occurs a third time, etc.

State Diagram:



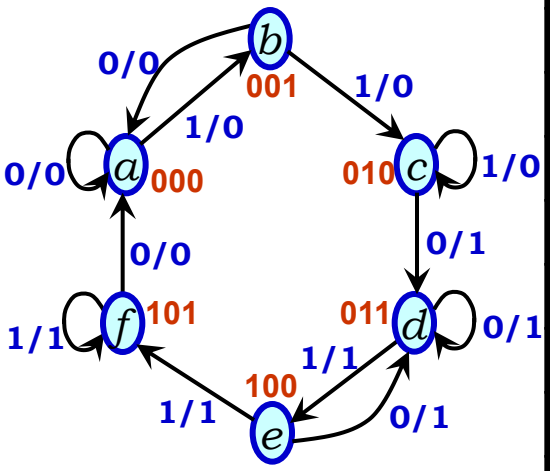
State Encoding:

6 states ⇨ 3-bit encoding

a	000
b	001
c	010
d	011
e	100
f	101

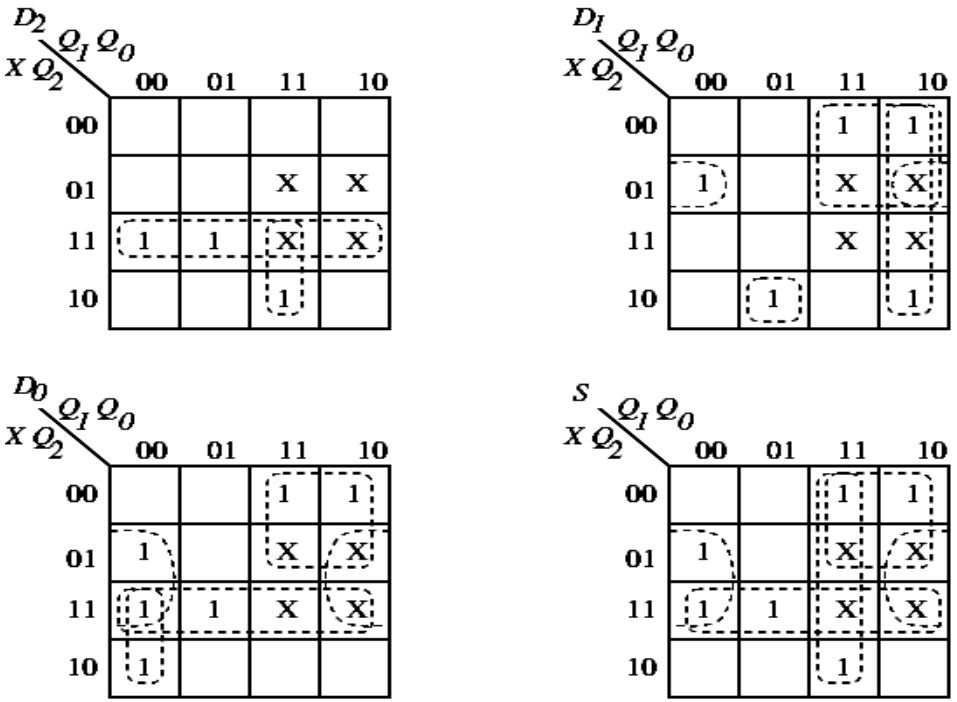
3-bit encoding ⇨ 3 flip-flops

Excitation table using D Flip-flops:



Input (X)	Present State			Next State			Flip-flop Inputs			Output (S)
	Q_2	Q_1	Q_0	Q_2*	Q_1*	Q_0*	D_2	D_1	D_0	
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	1	0	1	1	1
1	0	1	0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	0	1	1	1
1	0	1	1	1	0	0	1	0	0	1
0	1	0	0	0	1	1	0	1	1	1
1	1	0	0	1	0	1	1	0	1	1
0	1	0	1	0	0	0	0	0	0	0
1	1	0	1	1	0	1	1	0	1	1

Excitation and output logic functions:



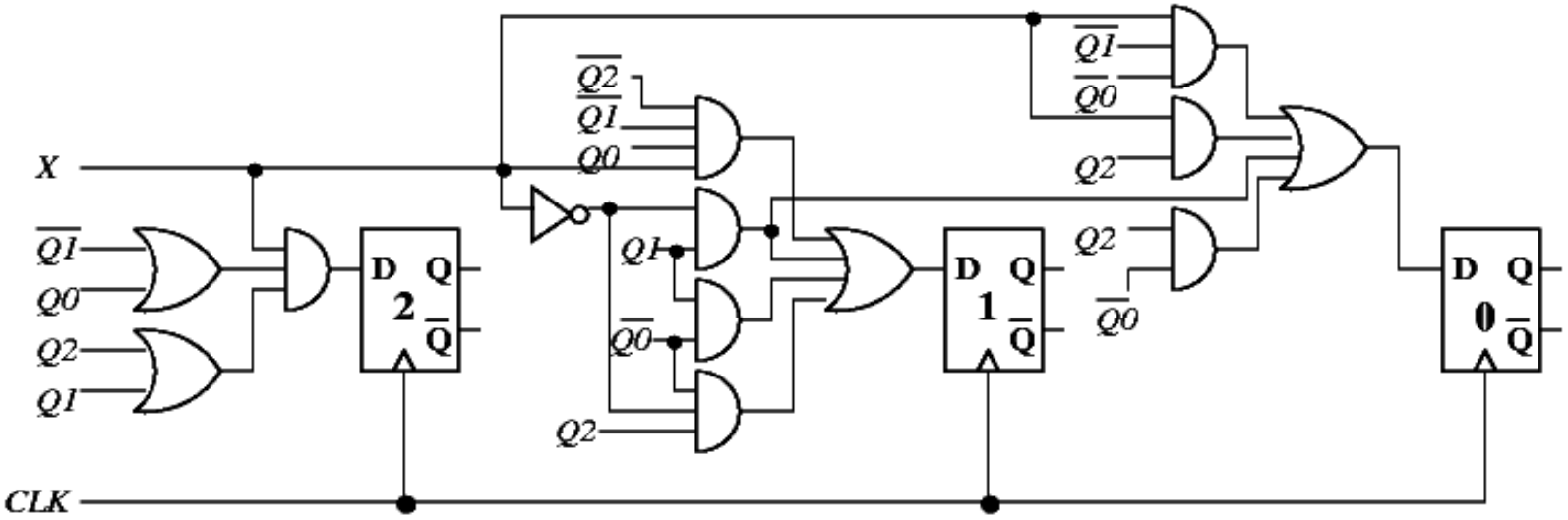
$$D_2 = XQ_2 + XQ_1Q_0$$

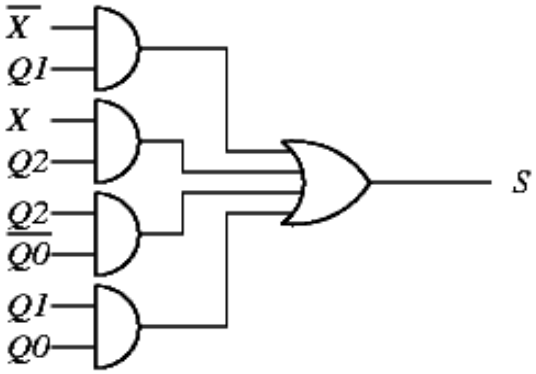
$$D_1 = X\overline{Q_2}\ \overline{Q_1}Q_0 + \overline{X}Q_1 + Q_1\overline{Q_0} + \overline{X}Q_2\overline{Q_0}$$

$$D_0 = \overline{X}Q_1 + XQ_2 + Q_2\overline{Q_0} + X\overline{Q_1}\ \overline{Q_0}$$

$$S = \overline{X}Q_1 + XQ_2 + Q_2\overline{Q_0} + Q_1Q_0$$

Circuit Implementation





$$D_2 = XQ_2 + XQ_1Q_0$$

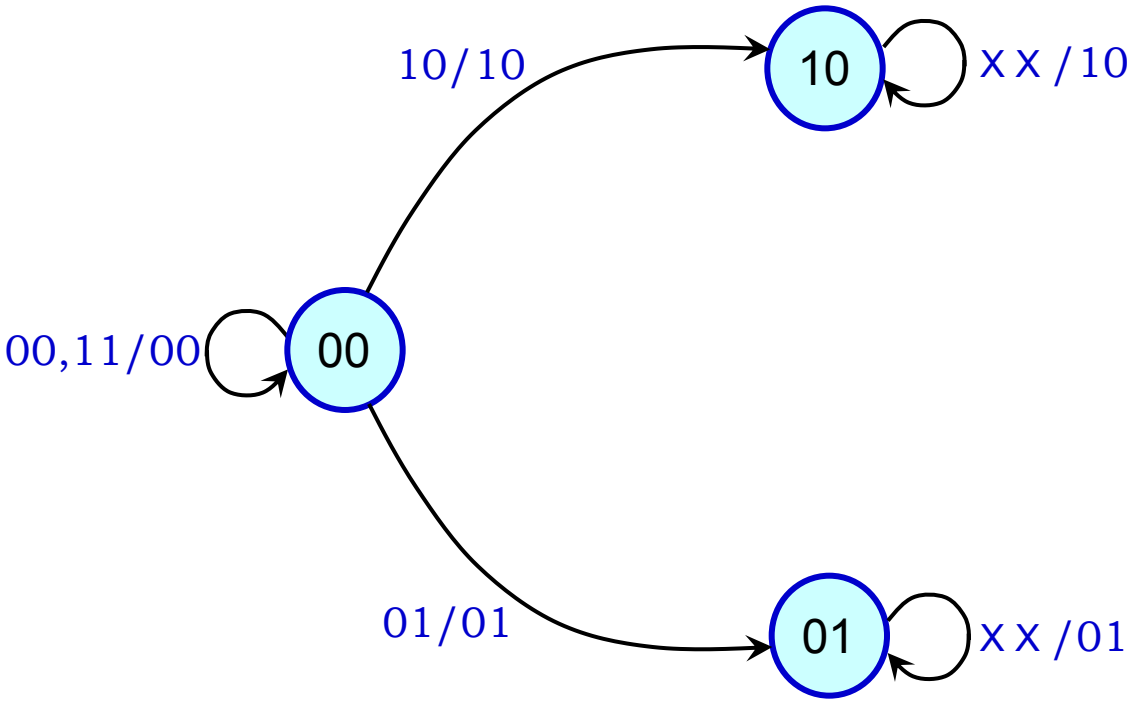
$$D_1 = X\overline{Q_2}\ \overline{Q_1}Q_0 + \overline{X}Q_1 + Q_1\overline{Q_0} + \overline{X}Q_2\overline{Q_0}$$

$$D_0 = \overline{X}Q_1 + XQ_2 + Q_2\overline{Q_0} + X\overline{Q_1}\ \overline{Q_0}$$

$$S = \overline{X}Q_1 + XQ_2 + Q_2\overline{Q_0} + Q_1Q_0$$

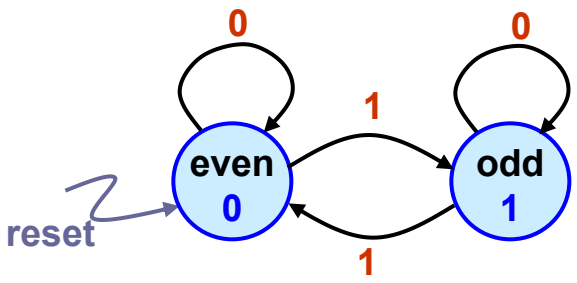
Example 3:

Design a sequential comparator circuit that is to determine which of the two multi-bit numbers, A and B, of equal length is larger. Inputs are supplied in MSB first fashion.



Example 4: Moore Machine

- Design a bit-serial *odd parity checker*: It counts the number of 1's in a bit-serial input stream and asserts its output when the input stream contains an odd number of 1's.



P.S. Q_0	N.S. (Q_0*)		Out (z)
	x=0	x=1	
even (0)	even (0)	odd (1)	0
odd (1)	odd (1)	even (0)	1

Using *D* flip-flops

$$D_0 = x \oplus Q_0$$
$$z = Q_0$$

Using *T* flip-flops

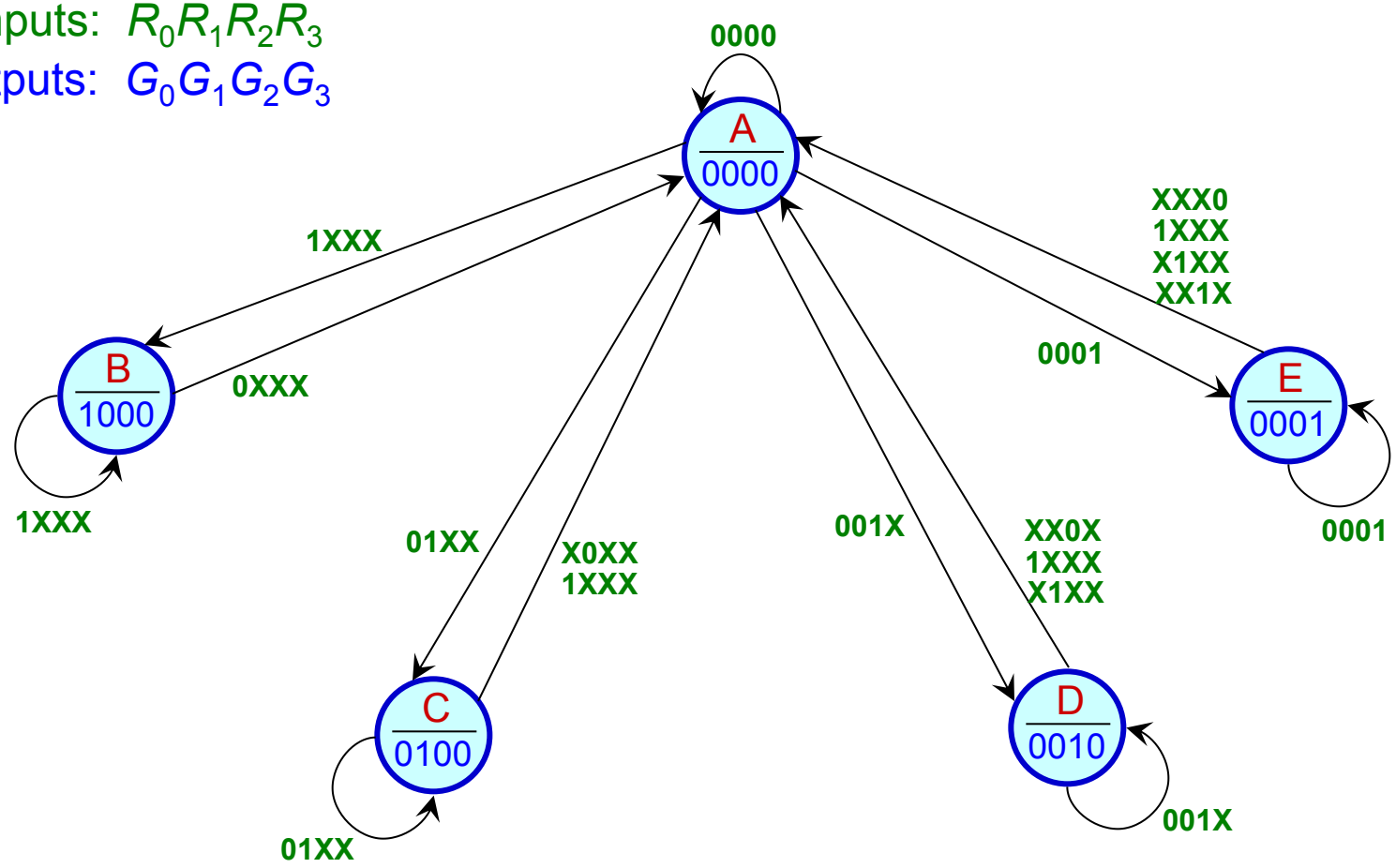
$$T_0 = x$$
$$z = Q_0$$

Example: Moore Machine

- A bus controller, that receives requests on separate lines, R_0 to R_3 from 4 devices to use the bus. It has four outputs, G_0 to G_3, only one of which is 1 (indicating which device is granted control for that clock period). The lowest number device has the highest priority. A higher priority device can preempt the bus. Assume that, before servicing any pending request, the controller remains idle for one clock period.
  - The bus controller has five states:
    - A: idle, no device is using the bus
    - B: device 0 is using the bus
    - C: device 1 is using the bus
    - D: device 2 is using the bus
    - E: device 3 is using the bus

Moore state machine for the bus controller example.

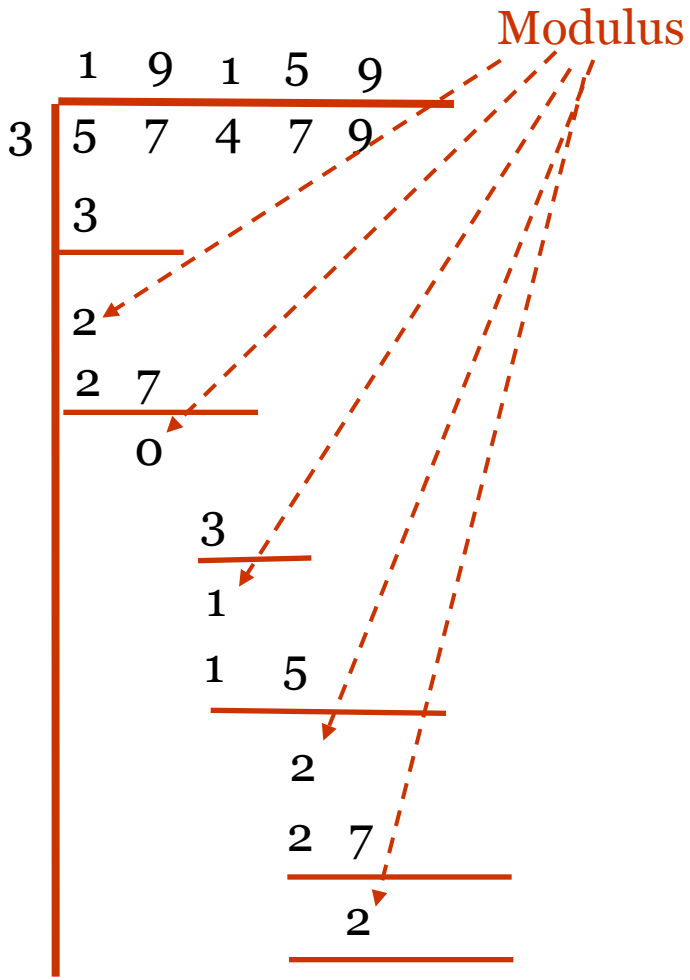
Inputs: $R_0R_1R_2R_3$
Outputs: $G_0G_1G_2G_3$



Read on your own

Example – Modulo 3 in Binary

- Analogous to Decimal - Use long division



- Take one digit at a time
  - Start at the most significant digit
- Calculate remainder
- Shift in another digit
  - The previous modulus moves to "tens" position
  - Shifted digit gets added
- Calculate remainder

- Repeat until all digits are done

Binary Modulus

- Take one bit at a time (MSB first)
- Calculate remainder m
  - Three possible values – 0,1,2
- Shift in the next bit
  - Shifting results in doubling the previous modulus value
  - New digit gets added to this doubled value

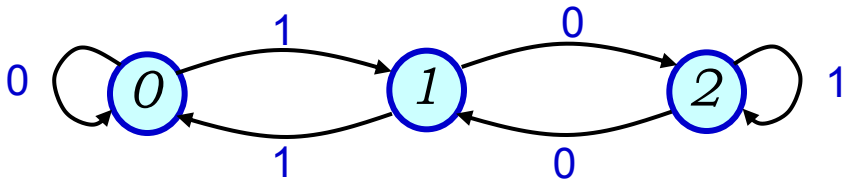
$$m' = \begin{cases} 2m+1; \text{if next bit is } 1 \\ 2m+0; \text{if next bit is } 0 \end{cases}$$

- Perform modulus of the resultant value

$$m = 2m' \bmod 3$$

State Table and State Diagram

Present Modulus	Next	
	$x = 0$	$x = 1$
0	0	1
1	2	0
2	1	2



Present Modulus	Next	
	$x = 0$	$x = 1$
00	00	01
01	10	00
10	01	10

Implementation with T Flip-Flops

Input	Present State		Next State		Flip-flop Inputs	
x	Q_1	Q_0	Q_1^*	Q_0^*	T_1	T_0
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	1	1	1
1	0	0	0	1	0	1
1	0	1	0	0	0	1
1	1	0	1	0	0	0

$$T_1 = \bar{x}(Q_1 + Q_o)$$

$$T_0 = (x \oplus Q_1) + Q_o$$

Do it Yourself

1. Design a Gray code counter using JK flip-flops.
2. Design a 3 bit up/down counter using T flip-flops. The count direction is determined by input D (D=0 means count down).
3. Design a sequence detector with one input *X* and one output *Z*. The detector should recognize the input sequence “101”. The detector should keep checking for the appropriate sequence and should not reset to the initial state after it has recognized the sequence.

End of Week 6: Module 30

Thank You