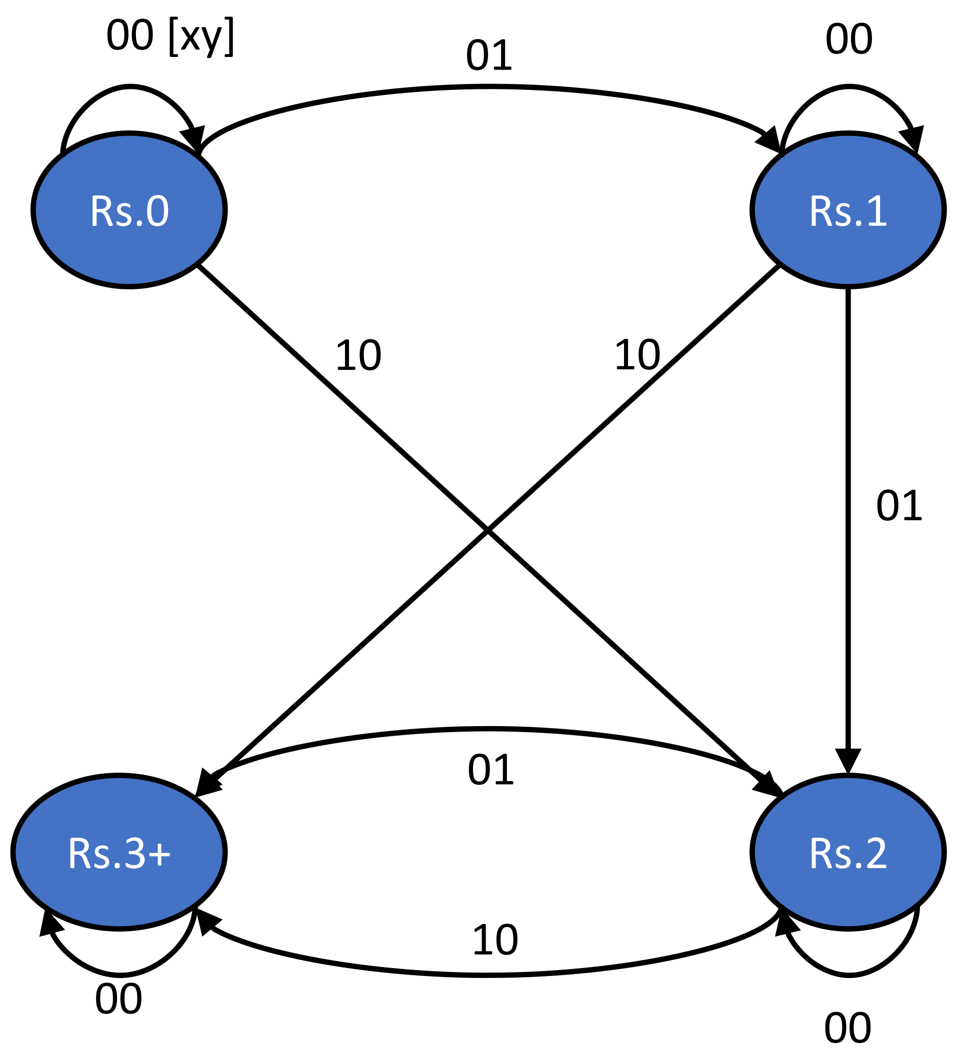
Lecture 21 – Registers and Counters 2

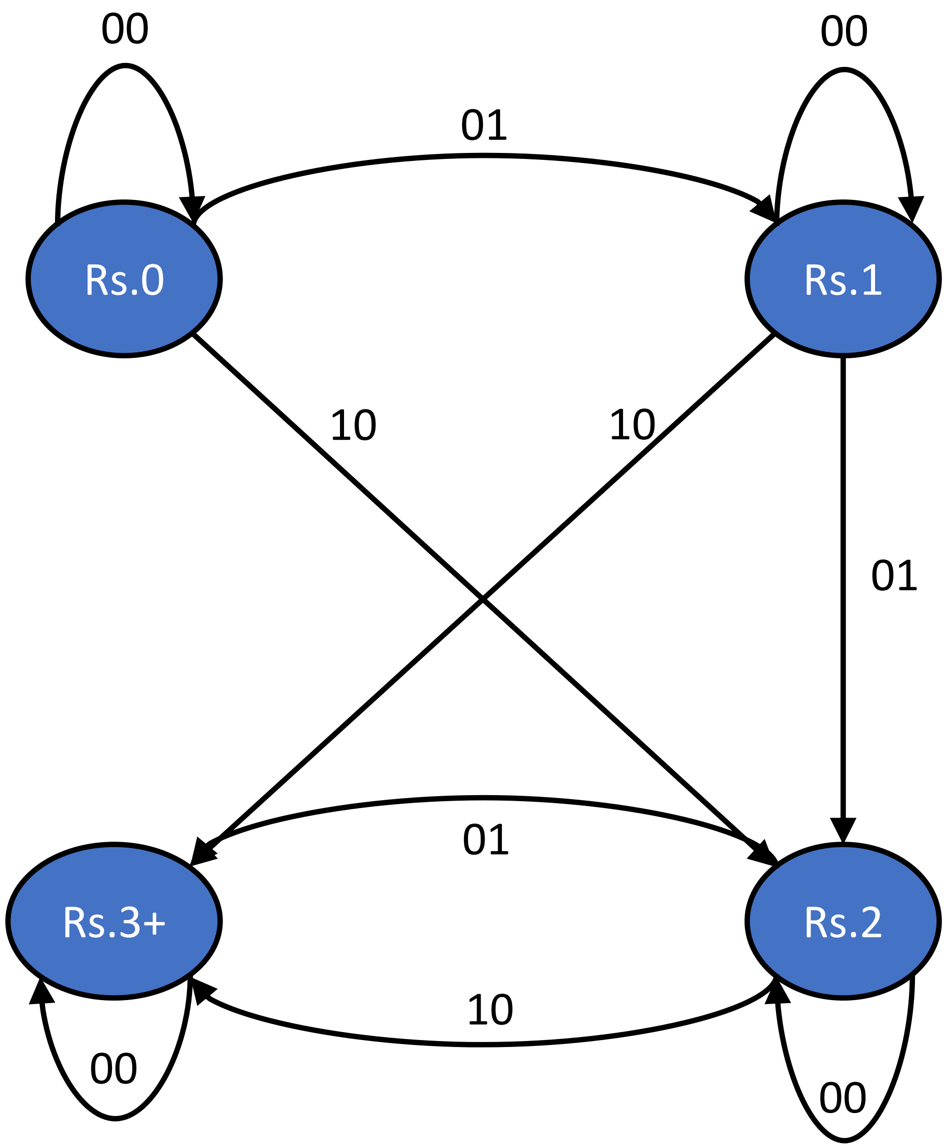
Chapter 6

Design of sequential circuits - The vending machine

- Let's say we are asked to design a circuit for a vending machine that dispenses candy for Rs. 3
- The input consists of a coin slot that can accept Rs. 1 and Rs. 2 coins
- The deposit of these coins by the user is detected by a circuit that gives out two outputs x and y
  - when Rs. 1 is inserted, y goes to one for one clock cycle
  - when Rs. 2 is inserted, x goes to one, for one clock cycle.
  - x and y are at zero by default
- Only one coin can be entered at once
- We need to design a circuit that takes x and y as inputs and outputs 1 if the sum is ≥ 3, so that the machine can dispense the candy

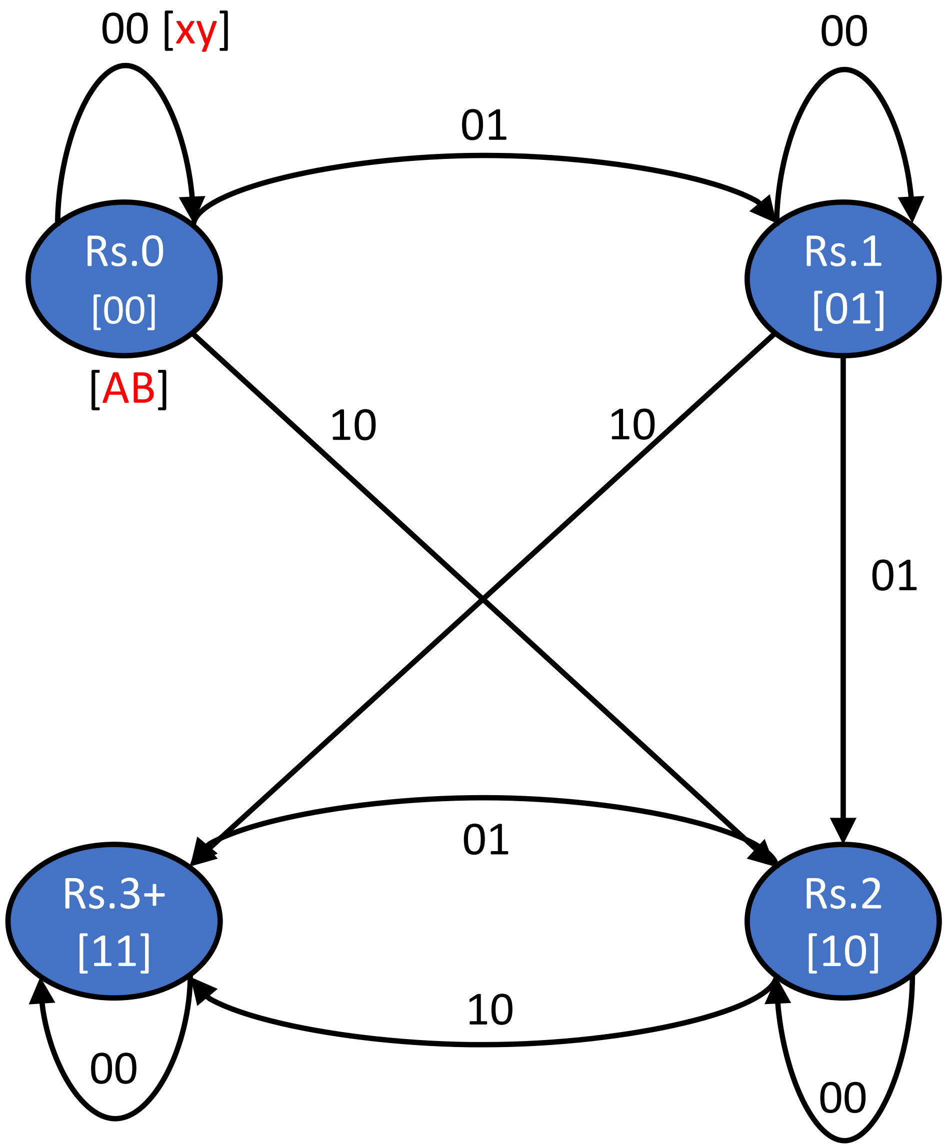


The vending machine



The vending machine

A	B	x	y	A(t+1)	B(t+1)	Z (output)
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	x	x	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	X	X	0
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	X	x	0
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	x	x	1



The vending machine

A(t+1)

AB \ xy	00	01	11	10
00	m_0 0	m_1 0	m_3 X	m_2 1
01	m_4 0	m_5 1	m_7 X	m_6 1
11	m_{12} 1	m_{13} 1	m_{15} X	m_{14} 1
10	m_8 1	m_9 1	m_{11} X	m_{10} 1

x (columns 11, 10); y (columns 01, 11); A (rows 11, 10); B (rows 01, 11)

$$A(t+1) = A + x + By$$

The vending machine

B(t+1)

AB \ xy	00	01	11	10
00	m_0: 0	m_1: 1	m_3: X	m_2: 0
01	m_4: 1	m_5: 0	m_7: X	m_6: 1
11	m_{12}: 1	m_{13}: 1	m_{15}: X	m_{14}: 1
10	m_8: 0	m_9: 1	m_{11}: X	m_{10}: 1

x

y

A

B

$$B(t+1) = (B + y + Ax)(B' + y' + A)$$

The vending machine

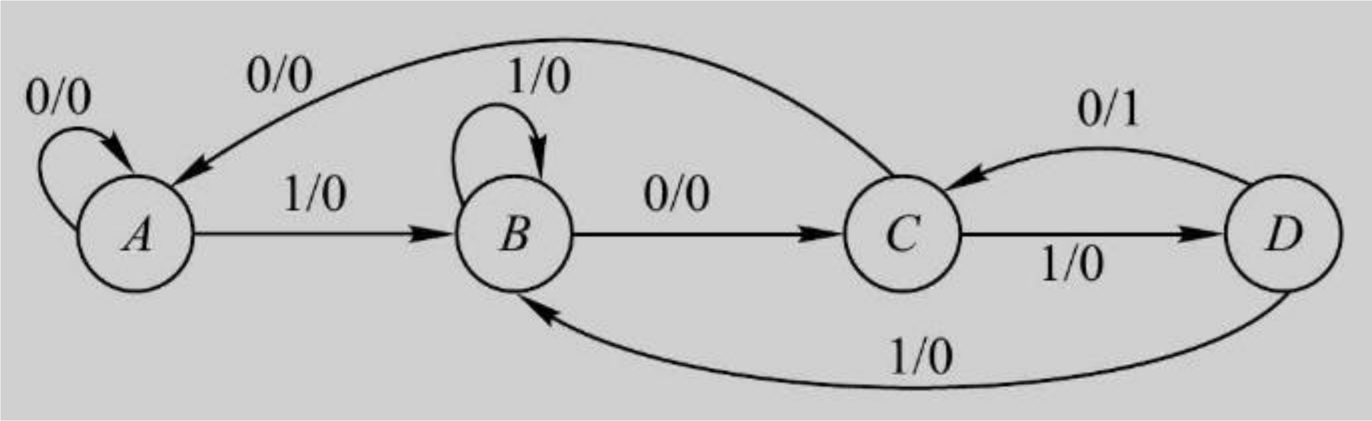
$$A(t+1) = A + x + By$$

$$B(t+1) = (B + y + Ax)(B' + y' + A)$$

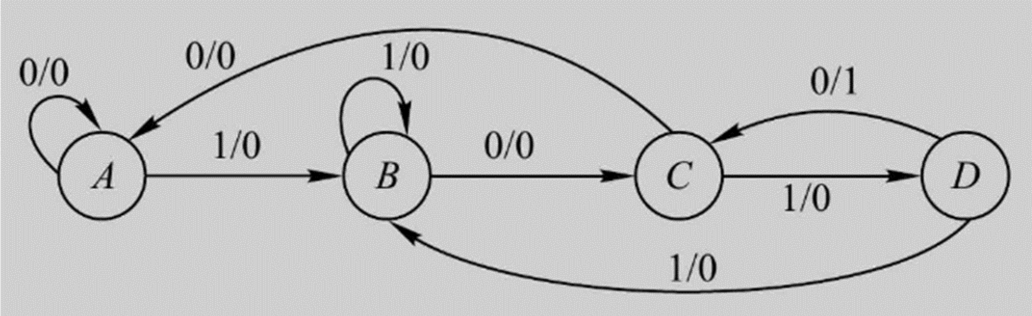
$$z = AB$$

Sequential circuit - pattern detector

- A sequence/pattern detector to detect a sequence of 1010. Output should go to high when the sequence is detected.
  - Eg: input x … 1 0 1 0 1 0 1 0 1 0 0 0 0 …
    output y … 0 0 0 1 0 1 0 1 0 1 0 0 0 …



Sequential circuit - design

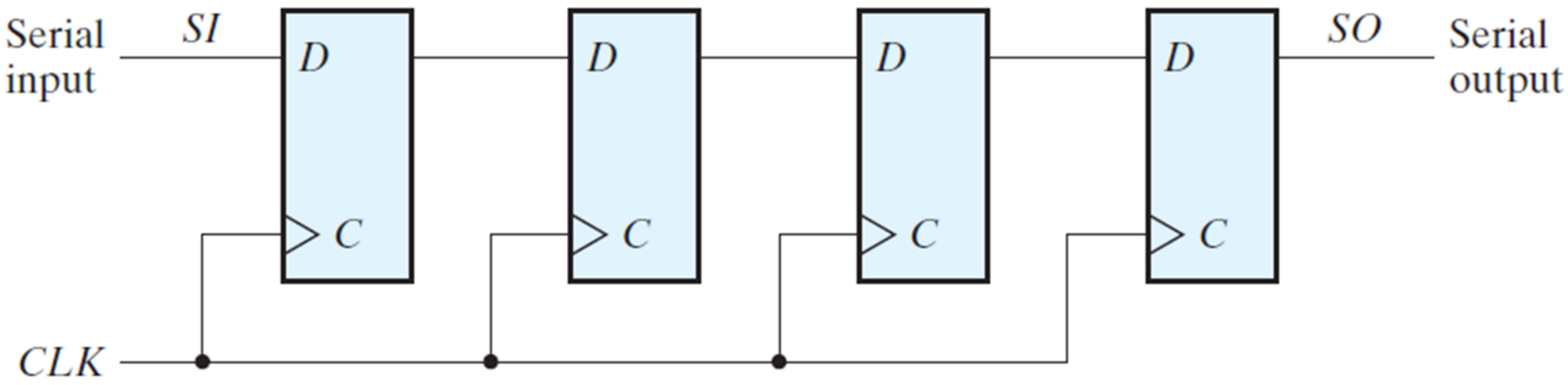


A → 00
B → 01
C → 10
D → 11

Present state		Input	Next state		Output
Q_1	Q_0	X	Q_1(t+1)	Q_0(t+1)	Y(t)
0	0	0	0	0	0
0	1	0	1	0	0
1	0	0	0	0	0
1	1	0	1	0	1
0	0	1	0	1	0
0	1	1	0	1	0
1	0	1	1	1	0
1	1	1	0	1	0

Universal register

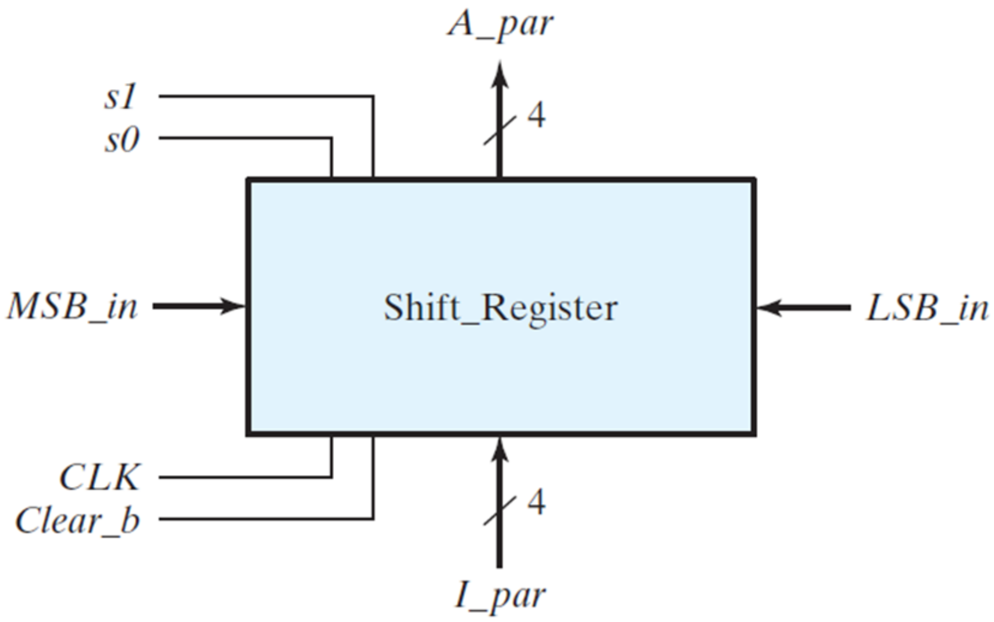
- If the flip-flop outputs of a shift register are accessible, then information entered serially by shifting can be taken out in parallel from the outputs of the flip-flops
- If a parallel load capability is added to a shift register, then data entered in parallel can be taken out in serial fashion by shifting the data stored in the register
- Some shift registers provide the necessary input and output terminals for parallel transfer
- They may also have both shift-right and shift-left capabilities



Universal register

- The most general shift register has the following capabilities:

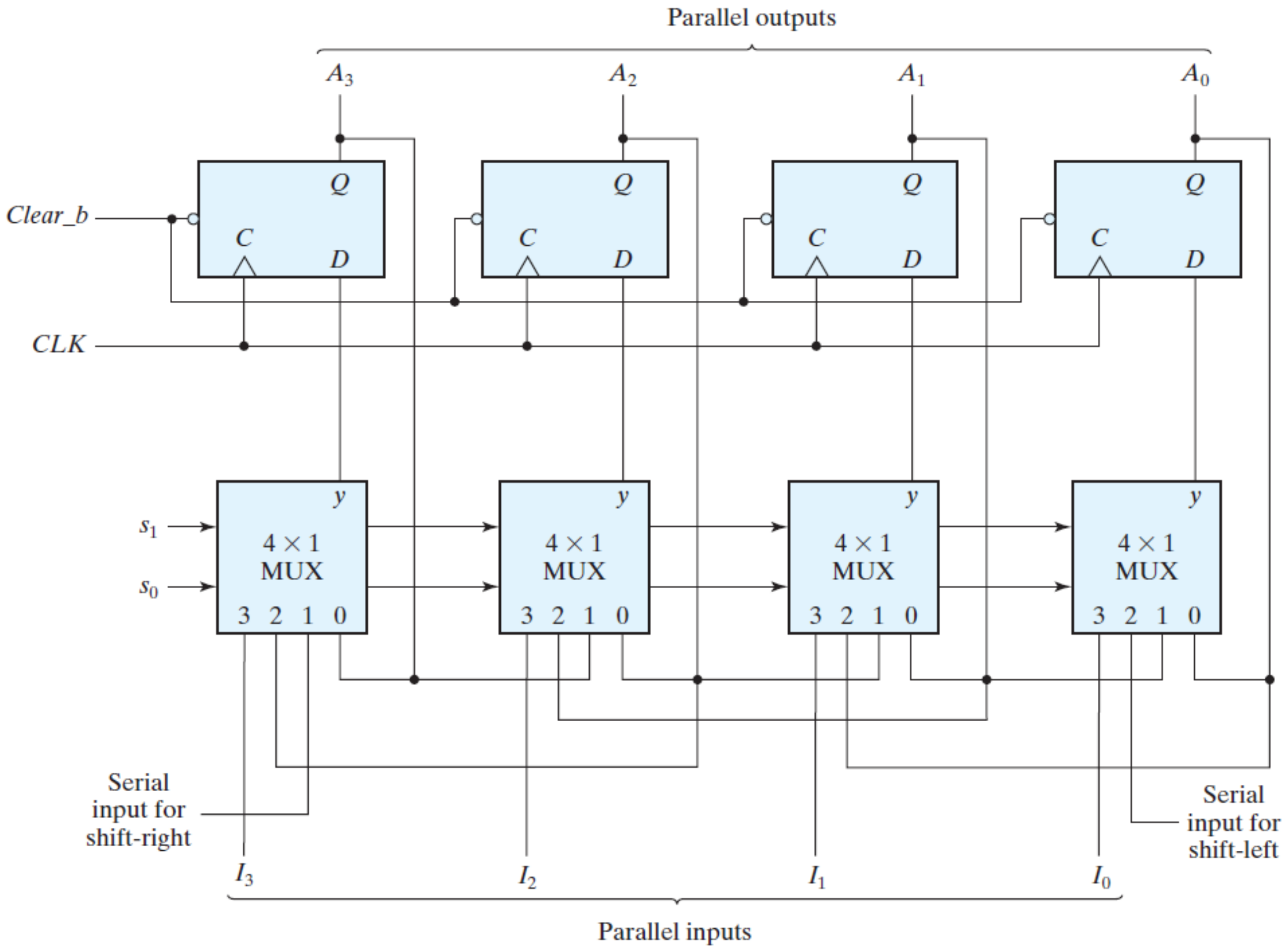
1. A *clear* control to clear the register to 0
2. A *clock* input to synchronize the operations
3. A *shift-right* control to enable the shift-right operation and the *serial input* and *output* lines associated with the shift right
4. A *shift-left* control to enable the shift-left operation and the *serial input* and *output* lines associated with the shift left
5. A *parallel-load* control to enable a parallel transfer and the *n* input lines associated with the parallel transfer
6. *n* parallel output lines
7. A control state that leaves the information in the register unchanged in response to the clock



Mode Control		
s_1	s_0	**Register Operation**
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

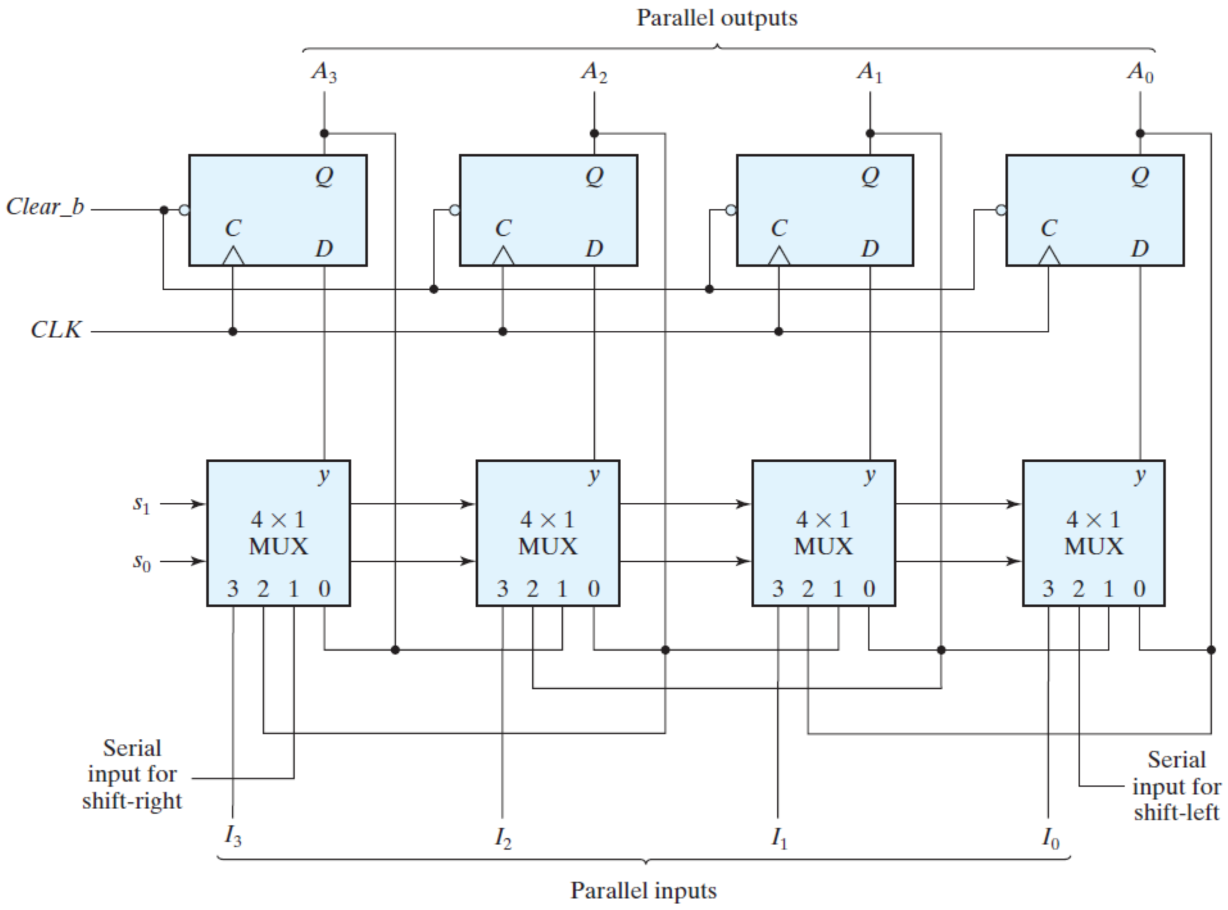
Universal register

- The circuit consists of four *D* flip-flops and four multiplexers
- The four multiplexers have two common selection inputs s_1 and s_0
- The selection inputs control the mode of operation of the register according to the function required
- The output of the MUXes are applied to the inputs of the FFs which controls the next state of the FF



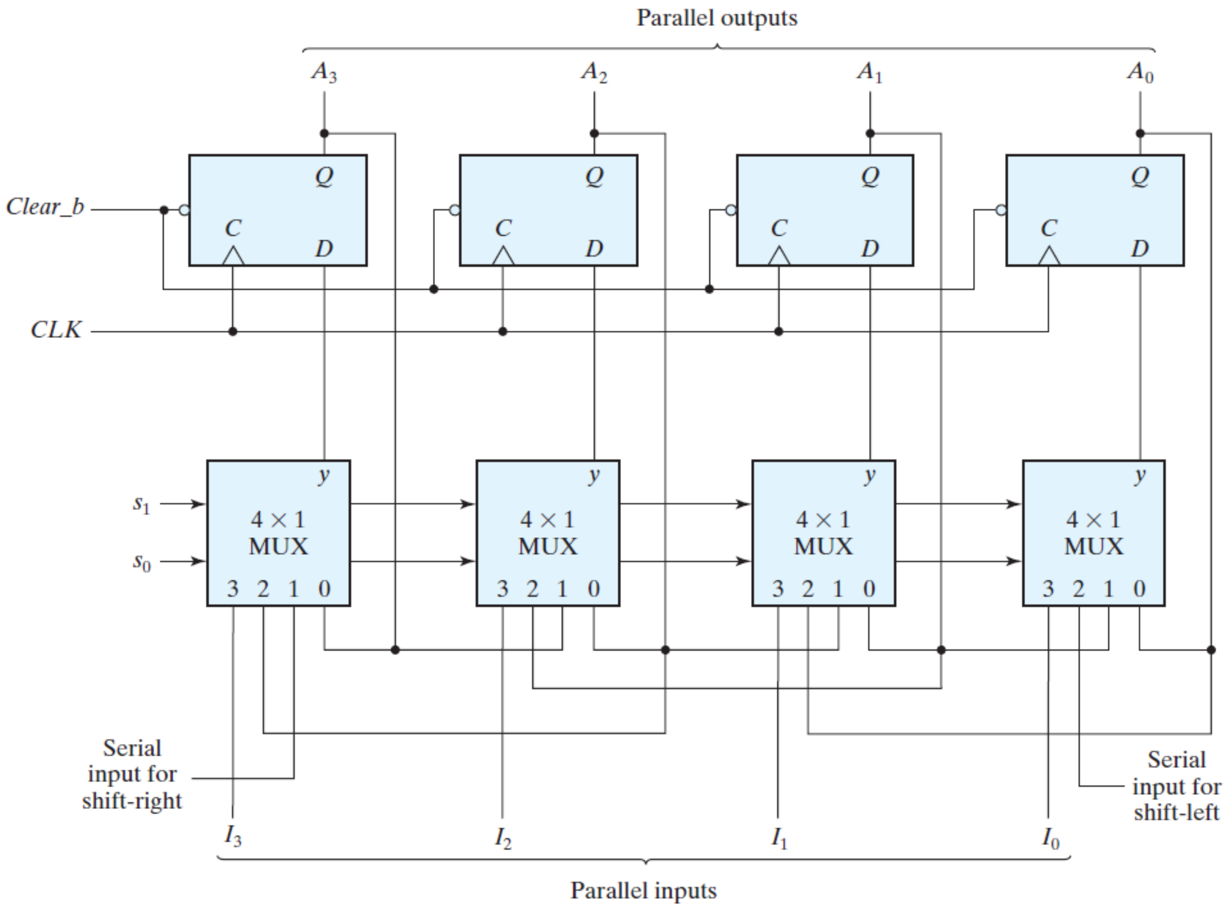
Universal register

- When $s_1 s_0$ = 00, the present value of the register is applied to the *D* inputs of the flip-flops
- This condition forms a path from the output of each flip-flop into the input of the same flip-flop, so that the output recirculates to the input in this mode of operation
- The next clock edge transfers into each flip-flop the binary value it held previously, and no change of state occurs



Universal register

- When s_1s_0 = 01, terminal 1 of the multiplexer inputs has a path to the *D* inputs of the flip-flops
- This causes a shift-right operation, with the serial input transferred into flip-flop A_3
- When s_1s_0 = 10, a shift-left operation results, with the other serial input going into flip-flop A_0
- Finally, when s_1s_0 = 11, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock edge

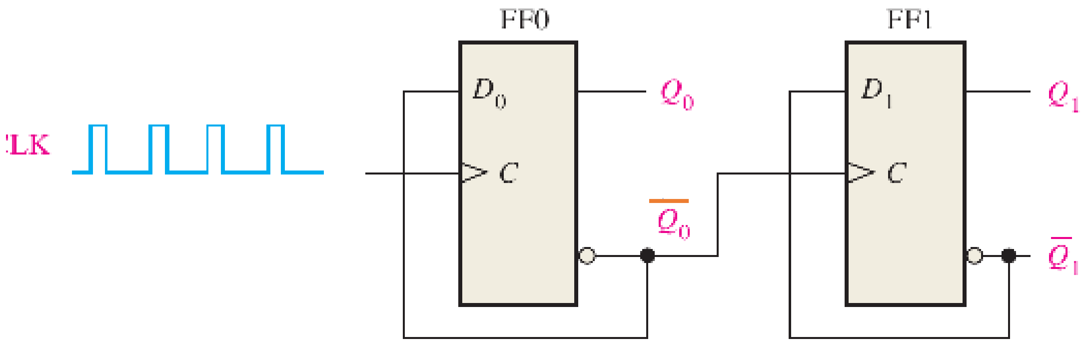


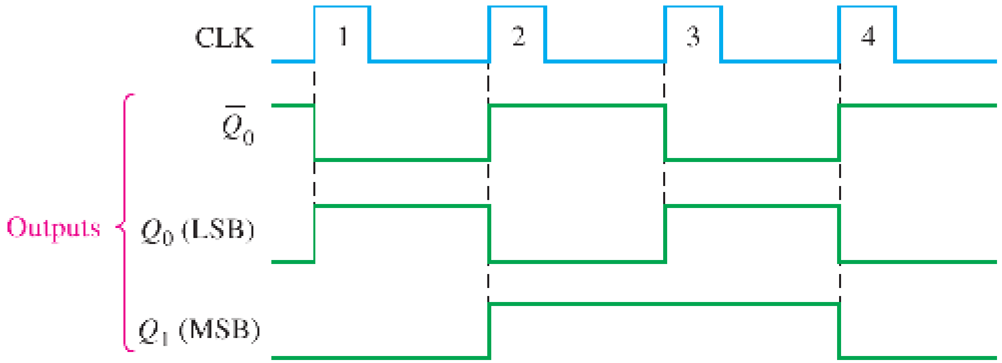
Counters

- A register that goes through a prescribed sequence of states upon the application of input pulses is called a *counter*
- The input pulses may be clock pulses, or they may originate from some external source and may occur at a fixed interval of time or at random
- The sequence of states may follow the binary number sequence or any other sequence of states
- Counters are available in two categories: ripple counters and synchronous counters
- In a ripple counter, a flip-flop output transition serves as a source for triggering other flip-flops
- In a synchronous counter, the C inputs of all flip-flops receive the common clock

Ripple counter - binary

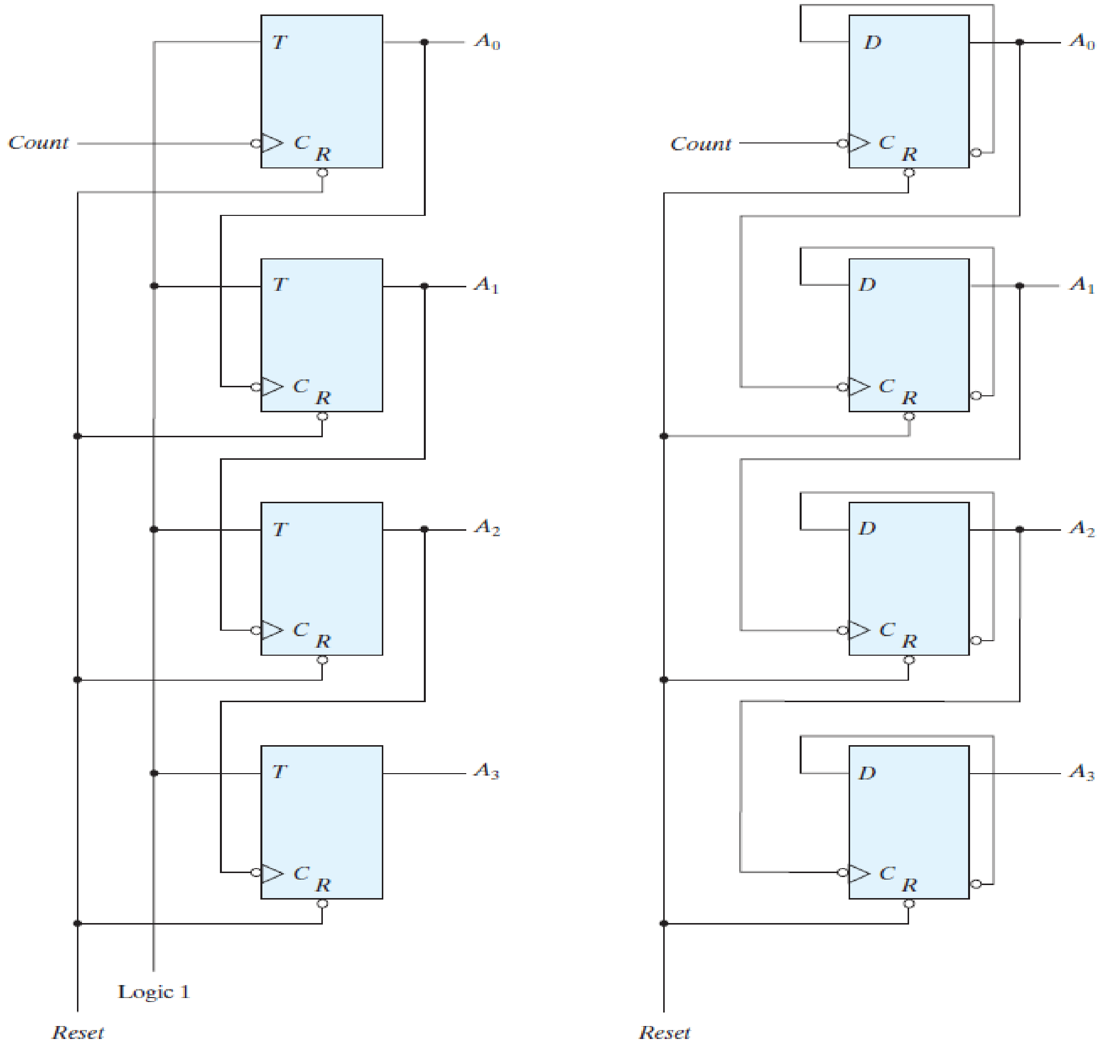
- A binary ripple counter consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the *C* input of the next higher order flip-flop
- The flip-flop holding the least significant bit receives the incoming count pulses
- A complementing flip-flop can be obtained from a *JK* flip-flop with the *J* and *K* inputs tied together or from a *T* flip-flop
- Another possibility is to use a *D* flip-flop with the complement output connected to the *D* input
- In this way, the *D* input is always the complement of the present state, and the next clock pulse will cause the flip-flop to complement





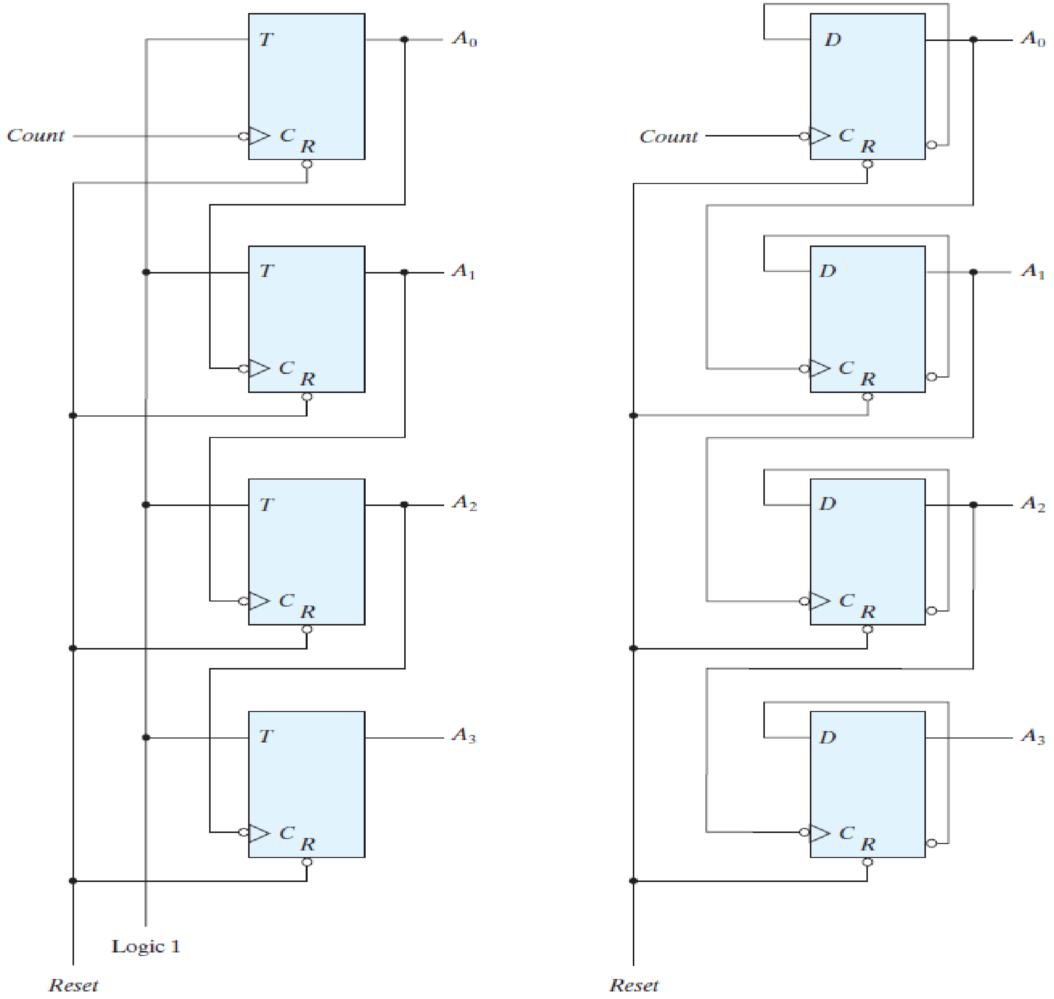
Ripple counter - binary

- The count starts with binary 0 and increments by 1 with each count pulse input
- After the count of 15, the counter goes back to 0 to repeat the count
- The least significant bit, A_0, is complemented with each count pulse input
- Every time that A_0 goes from 1 to 0, it complements A_1 and so on…



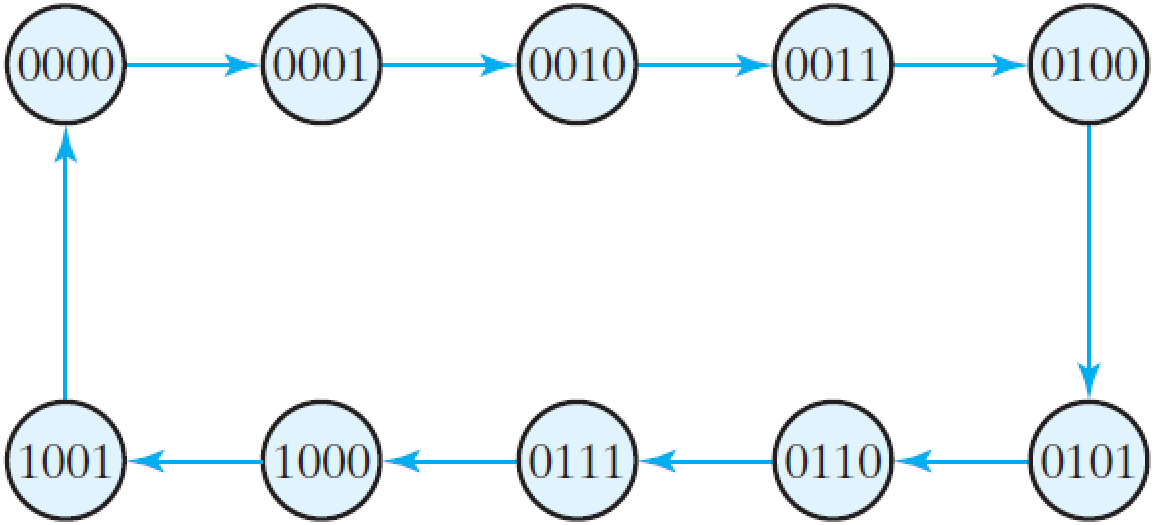
Ripple counter - binary

- For example, consider the transition from count 0011 to 0100
- A_0 is complemented with the count pulse
- Since A_0 goes from 1 to 0, it triggers A_1 and complements it
- As a result, A_1 goes from 1 to 0, which in turn complements A_2, changing it from 0 to 1
- A_2 does not trigger A_3, because A_2 produces a positive transition, and the flip-flop responds only to negative transitions



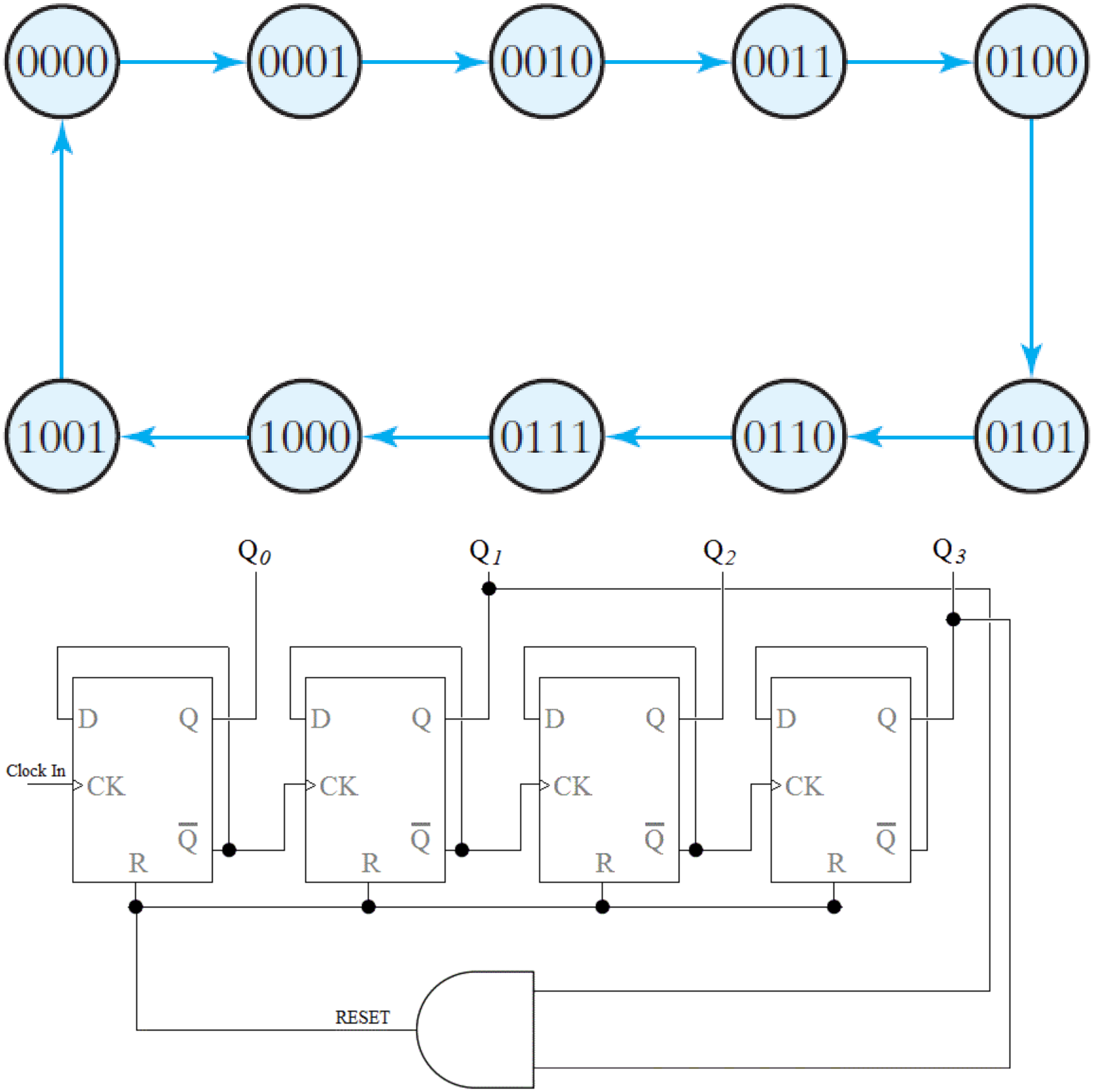
Ripple counter - BCD

- A decimal counter follows a sequence of 10 states and returns to 0 after the count of 9
- Such a counter must have at least four flip-flops to represent each decimal digit, since a decimal digit is represented by a binary code with at least four bits
- The sequence of states in a decimal counter is dictated by the binary code used to represent a decimal digit
- A decimal counter is similar to a binary counter, except that the state after 1001 (the code for decimal digit 9) is 0000 (the code for decimal digit 0)



Ripple counter - BCD

- We can obtain a decade counter by clearing all the flip-flops as soon as the state 1010 is obtained
- This can be done with the asynchronous input of CLR
- The condition for 1010 is checked by ANDing Q_3 and Q_1
- This is a very commonly used counter for making clocks/timer circuits



Ripple counter - BCD

- A decade counter counts from 0 to 9
- To count in decimal from 0 to 99, we need a two-decade counter
- To count from 0 to 999, we need a three-decade counter
- Multiple decade counters can be constructed by connecting BCD counters in cascade, one for each decade
- The inputs to the second and third decades come from Q_8 of the previous decade
- When Q_8 in one decade goes from 1 to 0, it triggers the count for the next higher order decade while its own decade goes from 9 to 0