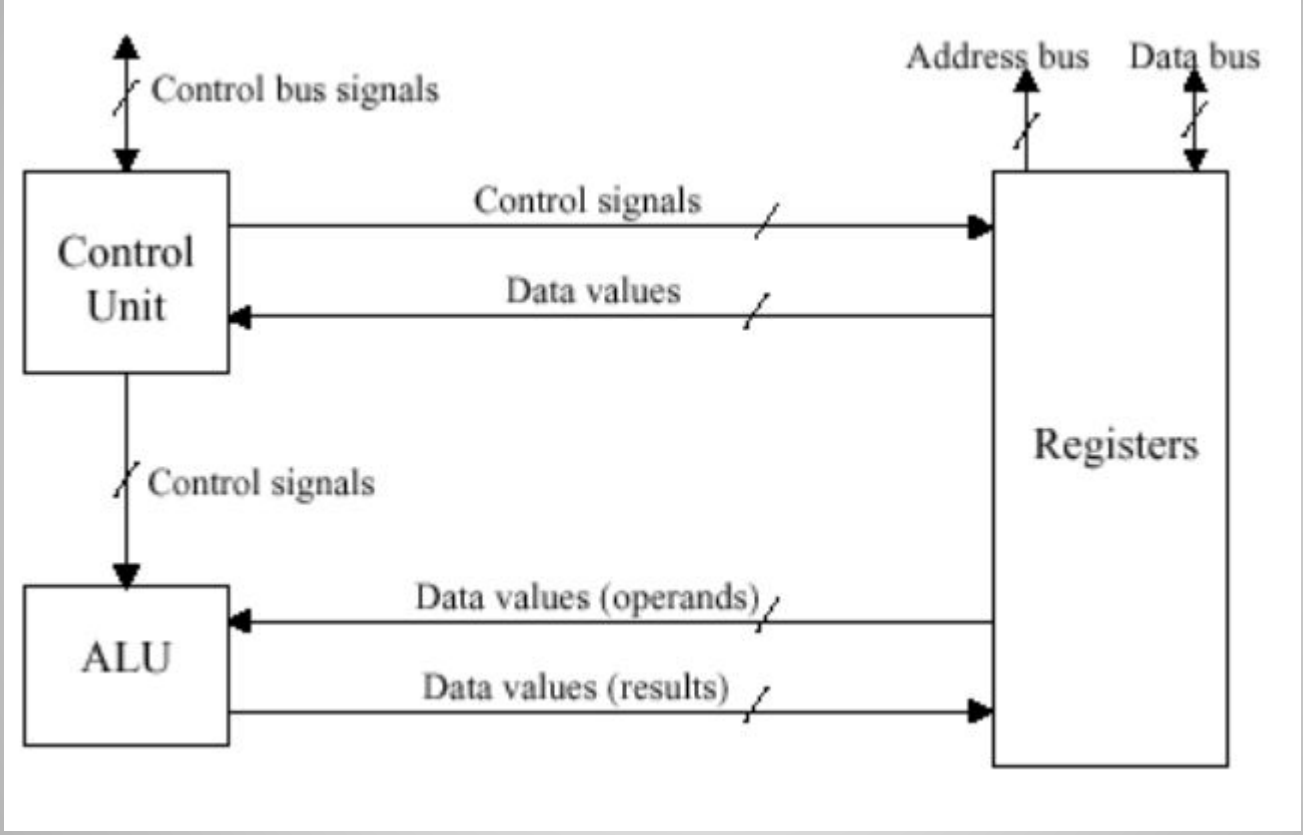
Registers and Counters

Ch.6

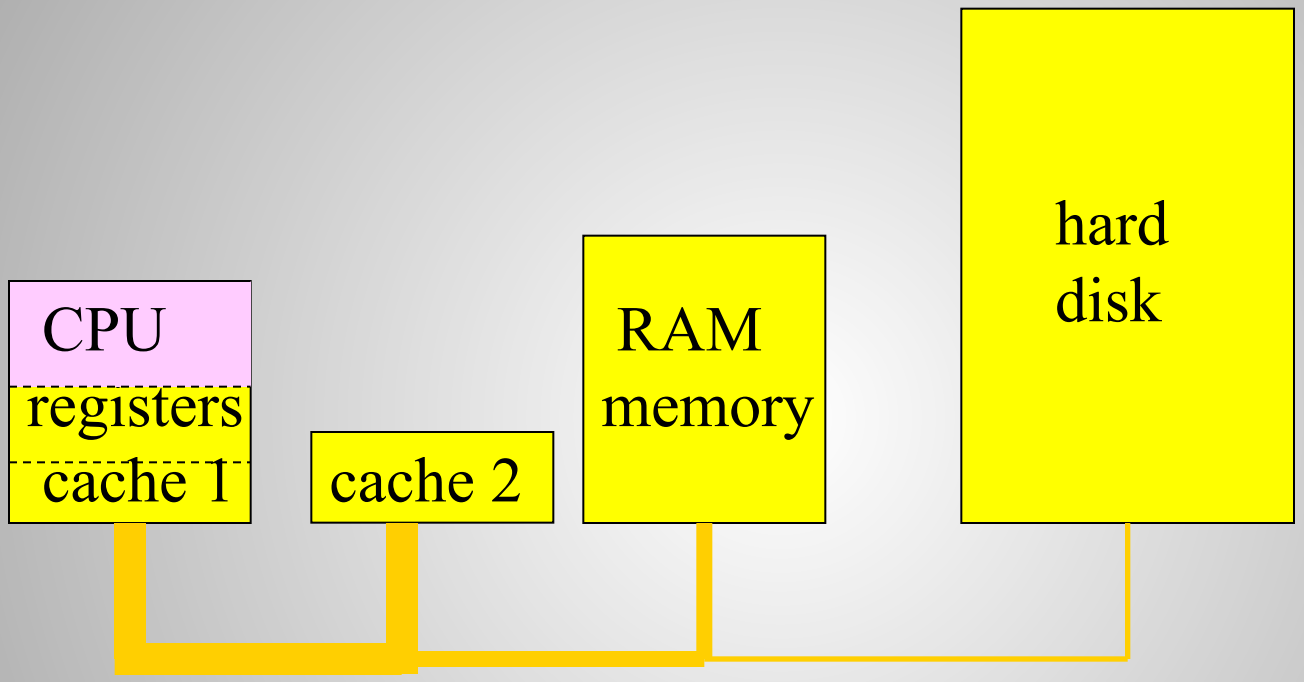
Outline

- Registers
- Parallel Data Transfer
- Parallel Addition with Accumulator
- Serial Data Transfer
- Serial Addition
- Counters
  - Ripple Counters vs Synchronous Counters
  - Binary Counter
  - BCD Counter
  - Counters for Other Sequences

Data Path vs Control unit



Memory Hierarchy



6.1 Registers

. Clocked sequential circuits
  - a group of flip-flops and combinational gates
  - connected to form a feedback path

    Flip-flops + Combinational gates

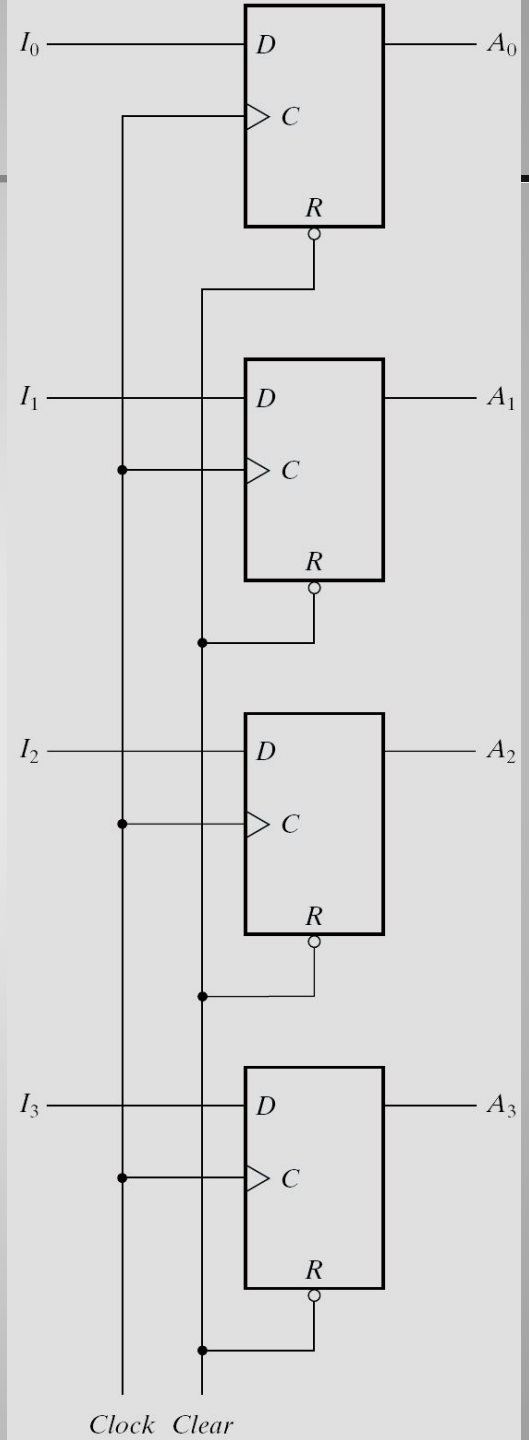
    (essential) (optional)

. Register:
  - a group of flip-flops
  - gates that determine how the information is transferred into the register

. Counter:
  - a register that goes through a predetermined sequence of states

Registers

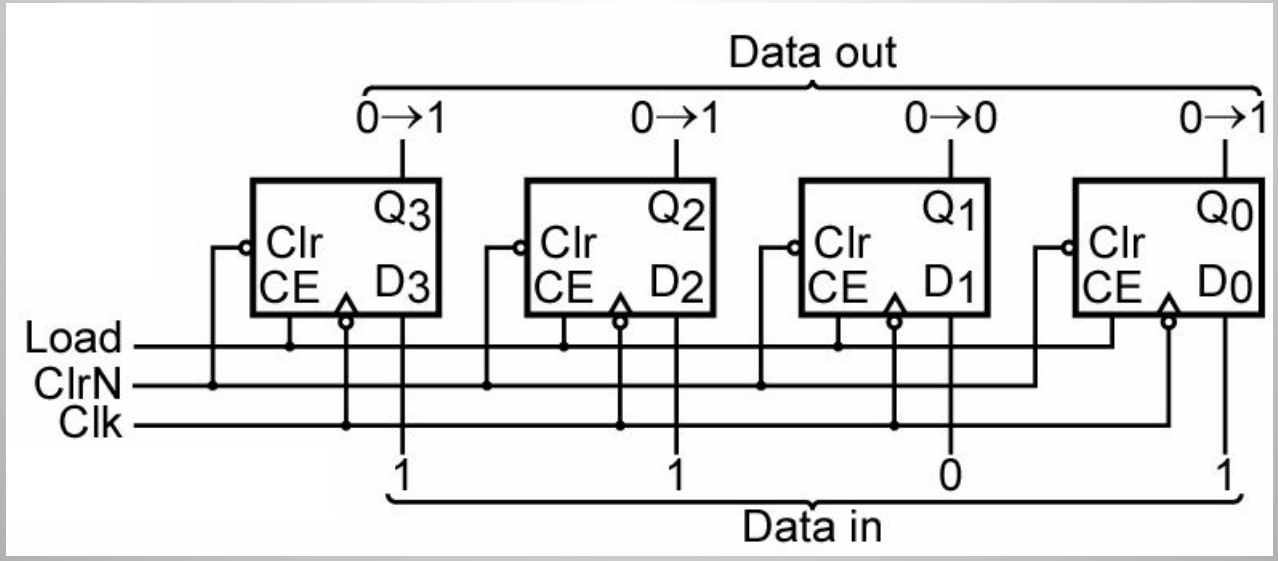
- An *n*-bit register
  - *n* flip-flops capable of storing *n* bits of binary information



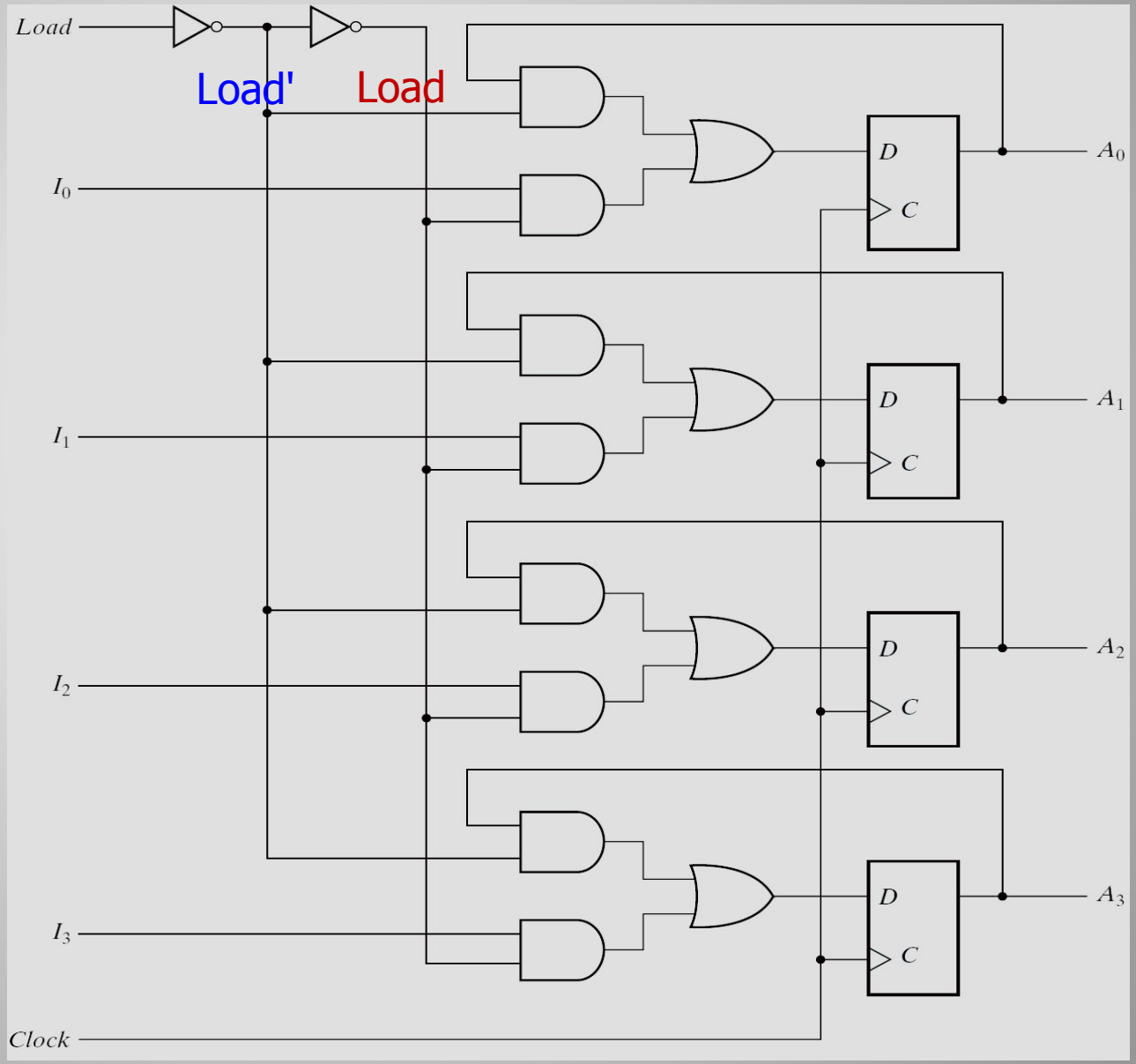
A basic 4-bit register

Registers

- To be useful, a register should keep its data until directed by a *load* control signal to read in new data
- E.g. 4-bit register built using *D* F/Fs with enable

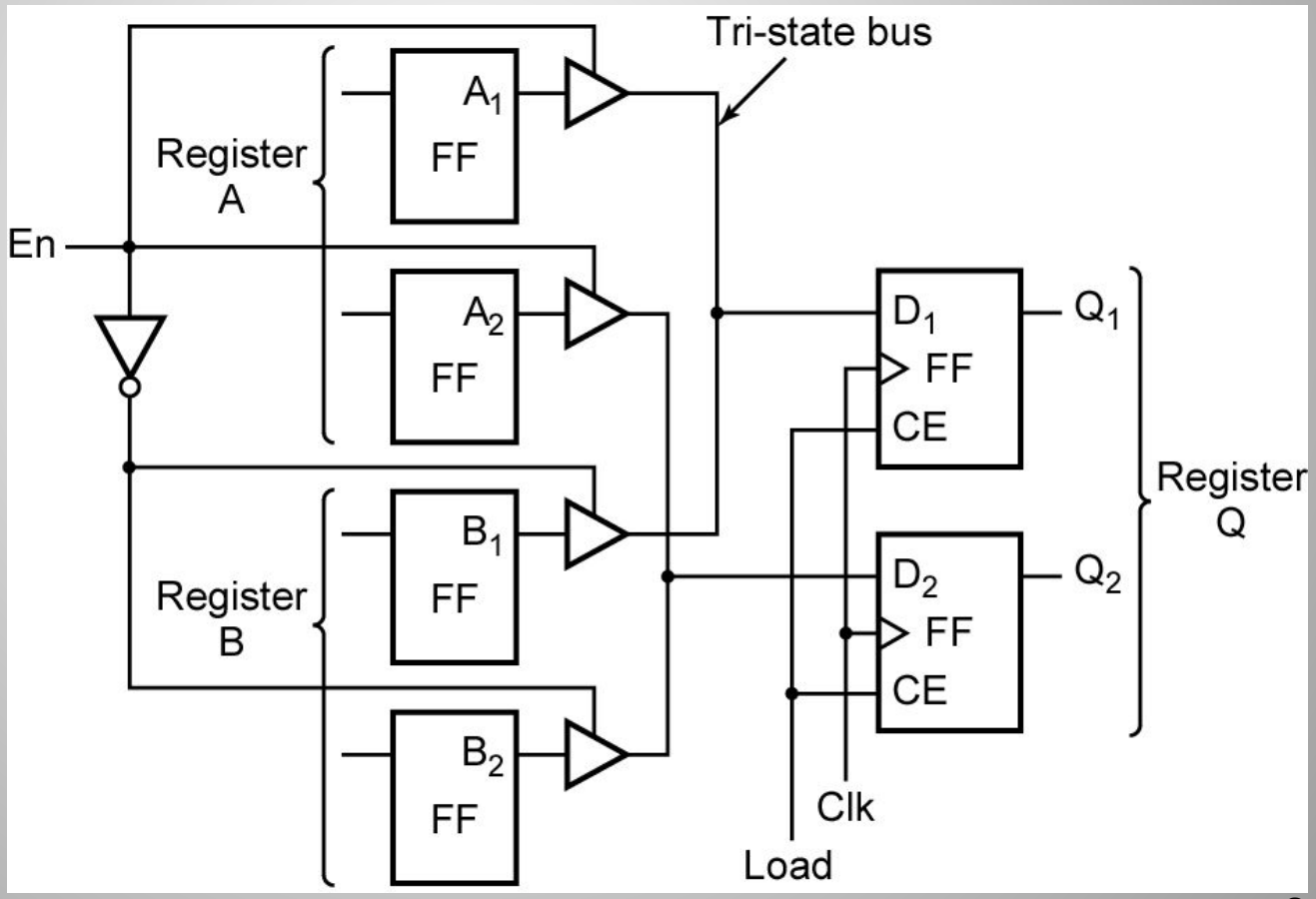


4-bit register with load control



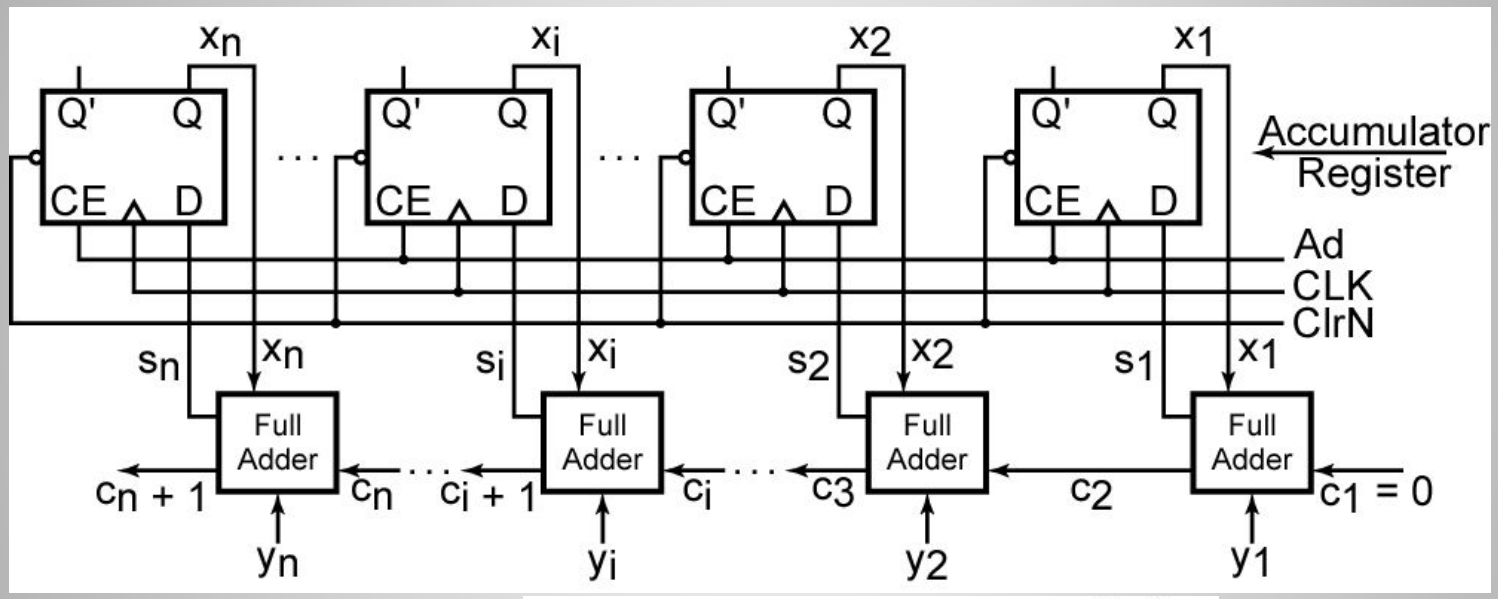
Register Transfer

. Describe what can be done between registers A, B, Q.

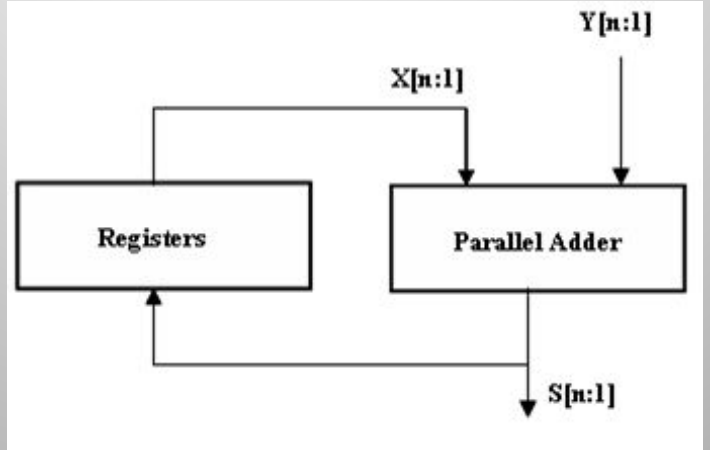


Parallel Adder with Accumulator

. Describe the operation of the circuit below. X<=X+Y;



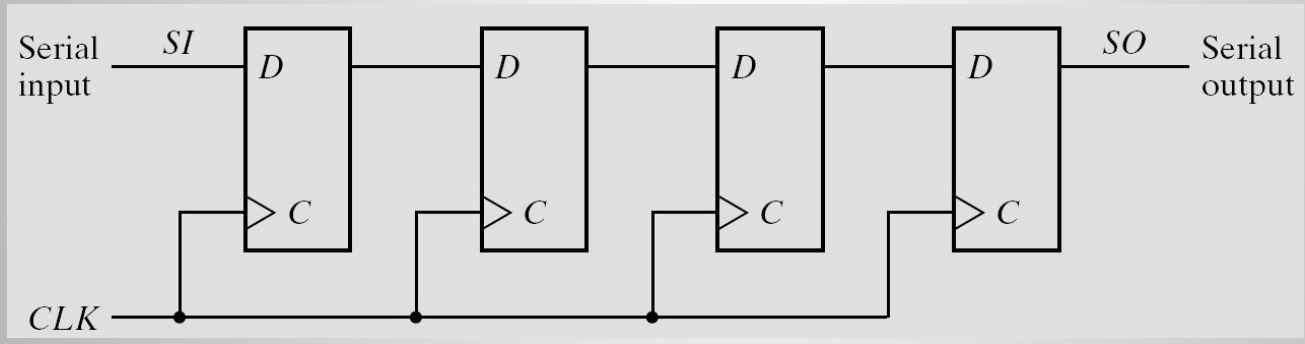
Equivalent to:



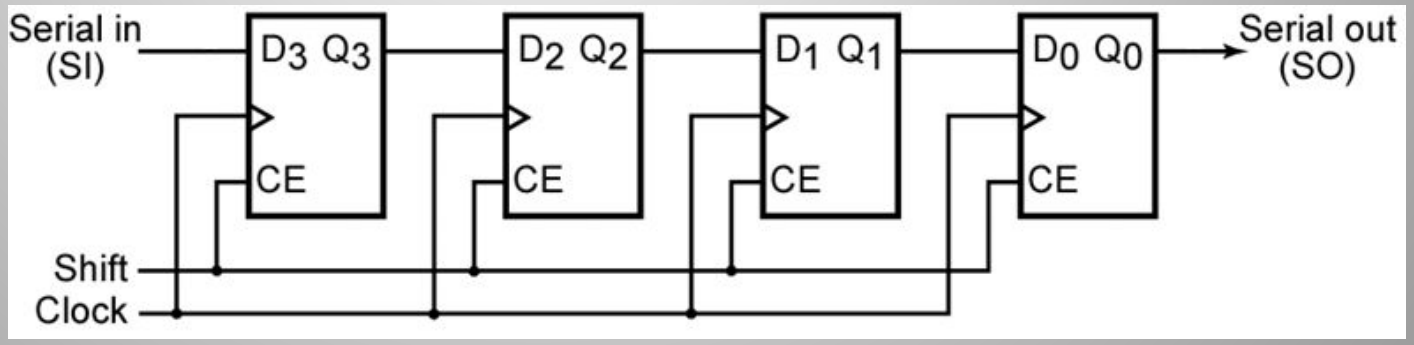
Shift Registers

. Shift registers
  - Can shift its data in one or both directions

. E.g. 4-bit serial-in, serial-out right shift registers



With shift control:

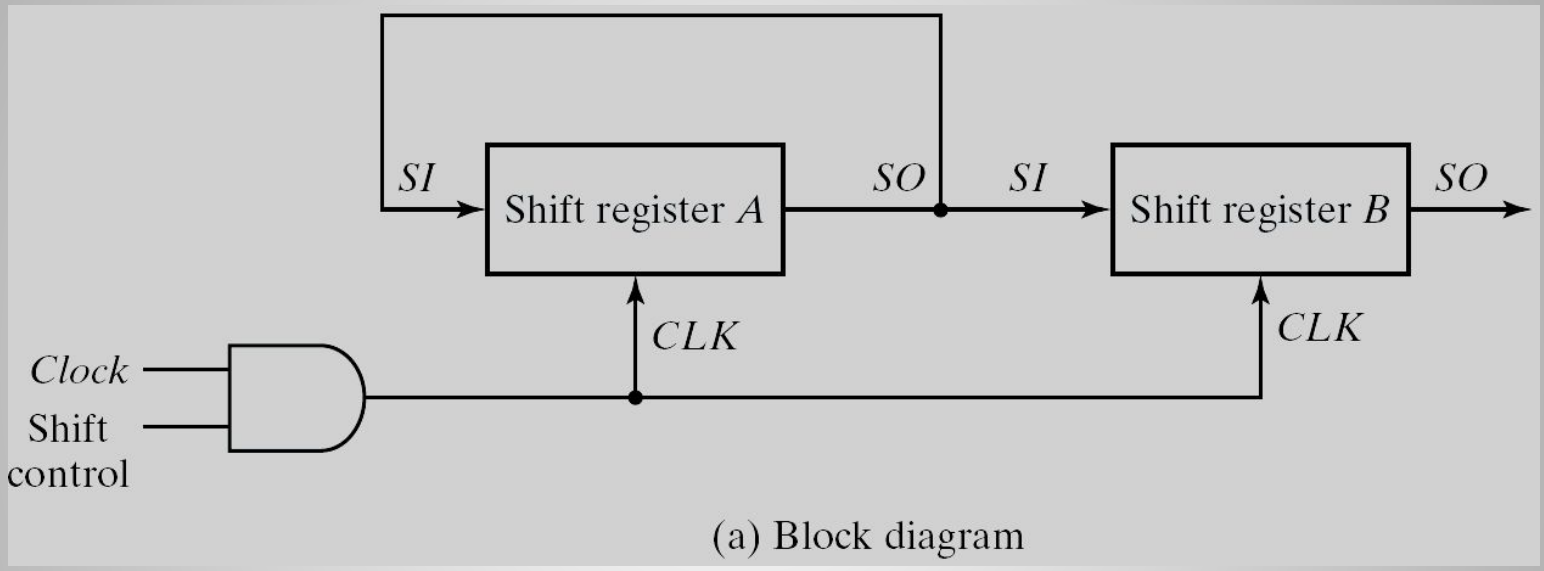


Data Transfer

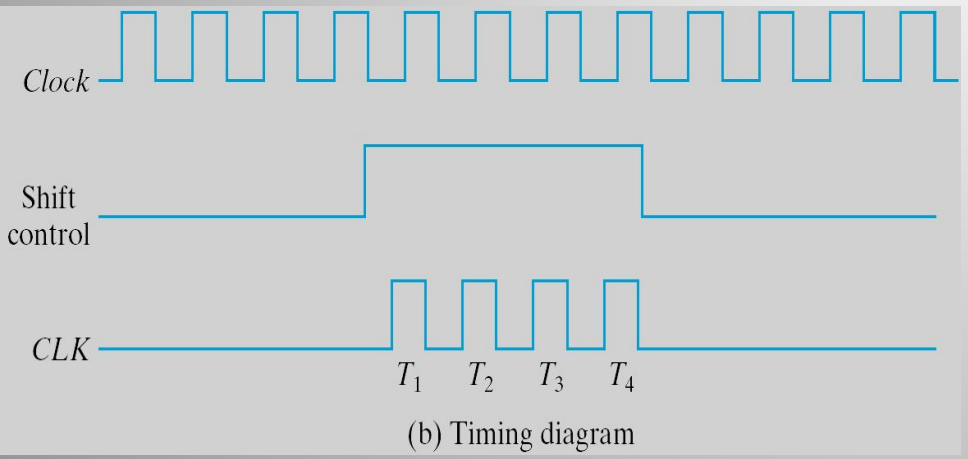
- Serial transfer vs. Parallel transfer
  - Serial transfer
    - Information is transferred one bit at a time
    - Shifts the bits out of a source register into a destination register
  - Parallel transfer
    - All the bits of a register are transferred at the same time

Serial Transfer

. E.g. Serial transfer from reg A to reg B



(a) Block diagram



(b) Timing diagram

Table 6.1
Serial-Transfer Example

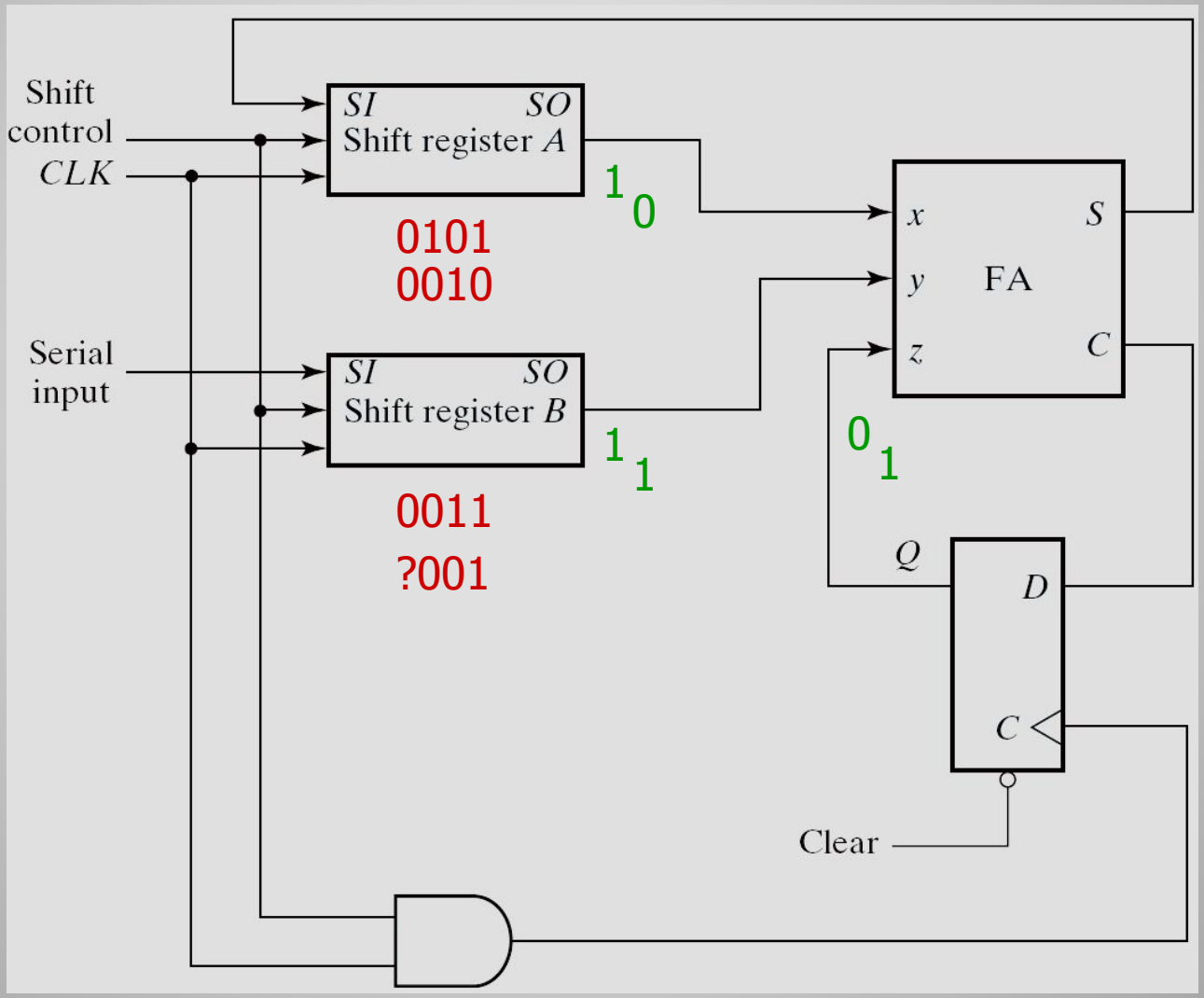
Timing Pulse	Shift Register *A*				Shift Register *B*			
Initial value	1	0	1	1	0	0	1	0
After T_1	1	1	0	1	1	0	0	1
After T_2	1	1	1	0	1	1	0	0
After T_3	0	1	1	1	0	1	1	0
After T_4	1	0	1	1	1	0	1	1

Serial Adder

- Serial adder
  - Adds two numbers *serially with a single full adder and a carry flip-flop* starting at the least significant bit (c.f. *n* full adders in an *n*-bit parallel adder)
  - Takes *n* clock cycles to add two *n*-bit numbers
  - It's smaller but slower than a parallel adder

Serial addition using D flip-flop for storing carry

$A \leftarrow A+B$



Types of Shift Registers

- Unidirectional shift registers
- Bidirectional shift registers
- Universal shift registers
  - have both direction shifts & parallel load/access capabilities

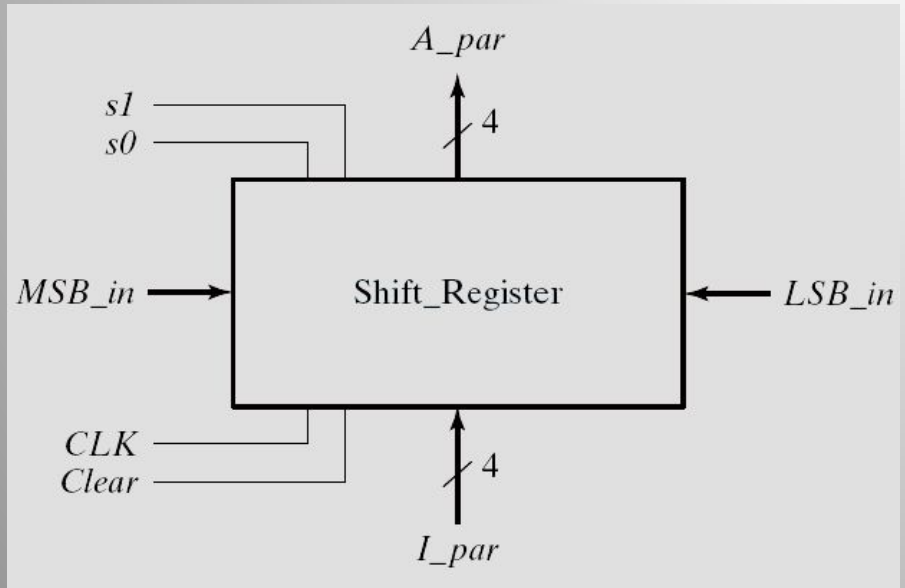
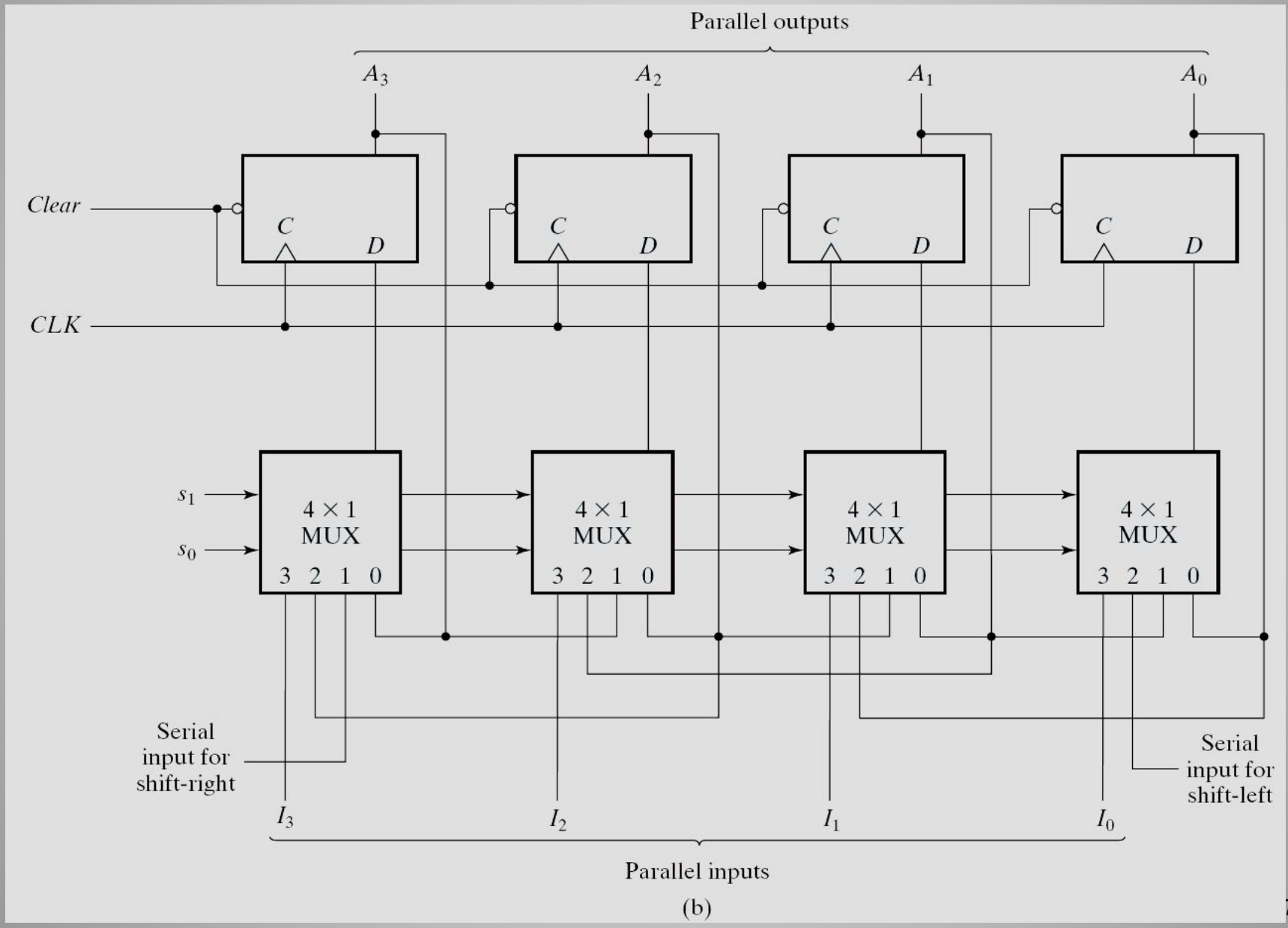


Table 6.3
Function Table for the Register of Fig. 6.7

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

An implementation of a 4-bit universal shift register



Counters

- Counters are special sequential circuits that sequence though some prescribed set of outputs.
- Examples:
  - Binary counter:

    000⟶001⟶010⟶011⟶100⟶101⟶110⟶111

    ⟶000⟶001⟶…
  - BCD counter:

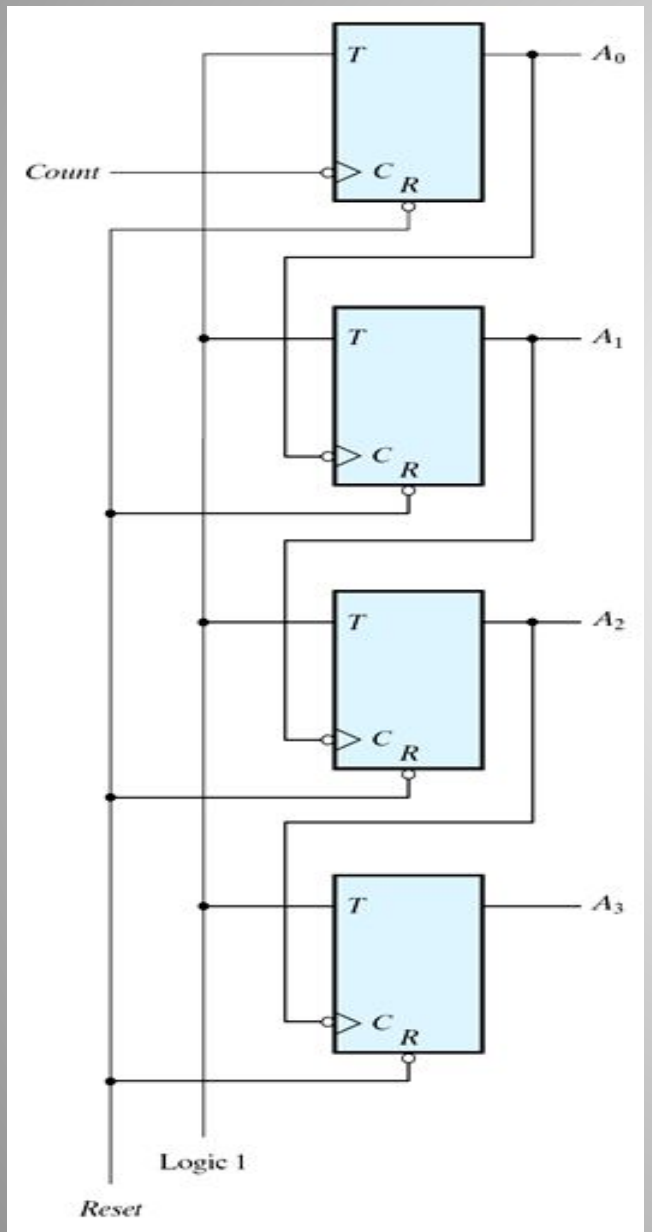
    0000⟶0001⟶0010⟶…⟶1001

    ⟶0000⟶0001⟶…

Ripple Counters vs Synchronous Counters

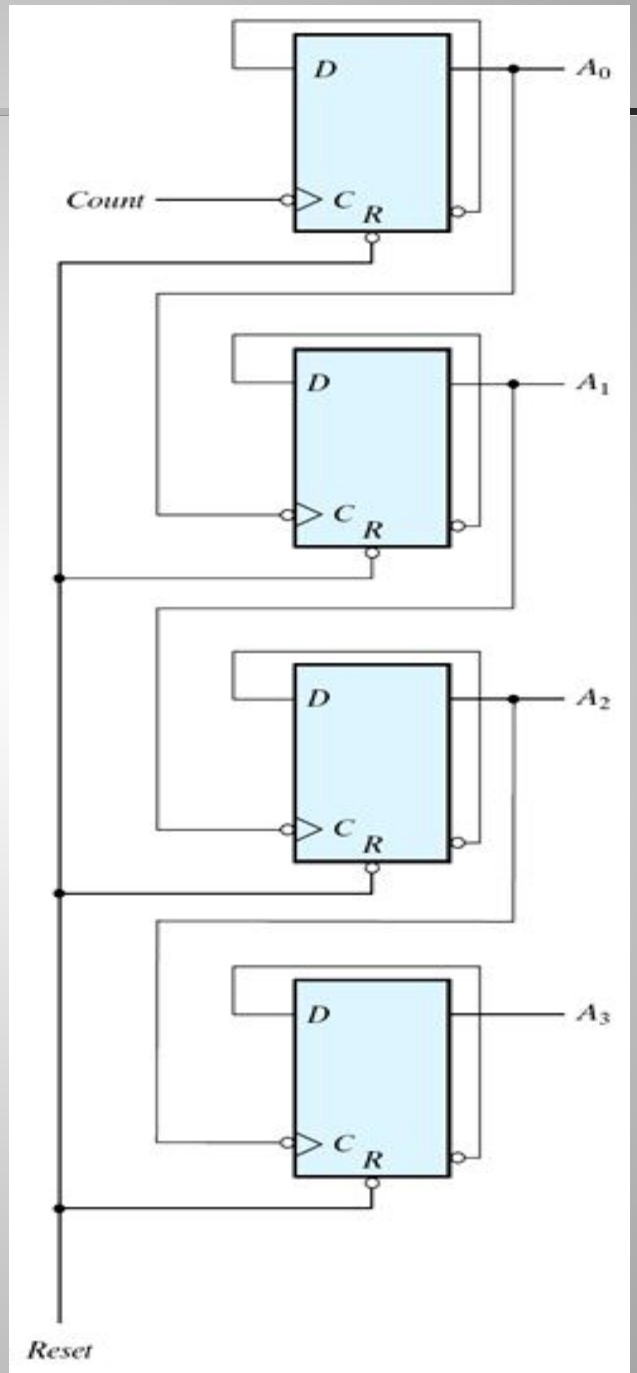
- Counters are classified into two board categories according to the way they are clocked.
- *Ripple counter*
  - the first flip-flop is clocked by the external clock pulse and then each successive flip-flop is clocked by the output of the preceding flip-flop
  - i.e., an asynchronous sequential circuit
- *Synchronous counter*
  - all flip-flops are clocked simultaneously by the external clock pulse.

Binary Ripple Counter



- Count sequence: 0000, 0001, 0010, …, 1111, 0000, 0001, …
- This is a *ripple counter* because when the count is changed, the i+1th bit only changes after the i^{th} bit has changed. (The flip-flops' *outputs don't change synchronously* together.)
- Note that every times A_i goes from 1 to 0, it toggles A_{i+1}.

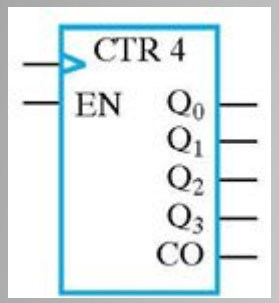
4-bit binary ripple counter with D flip-flops



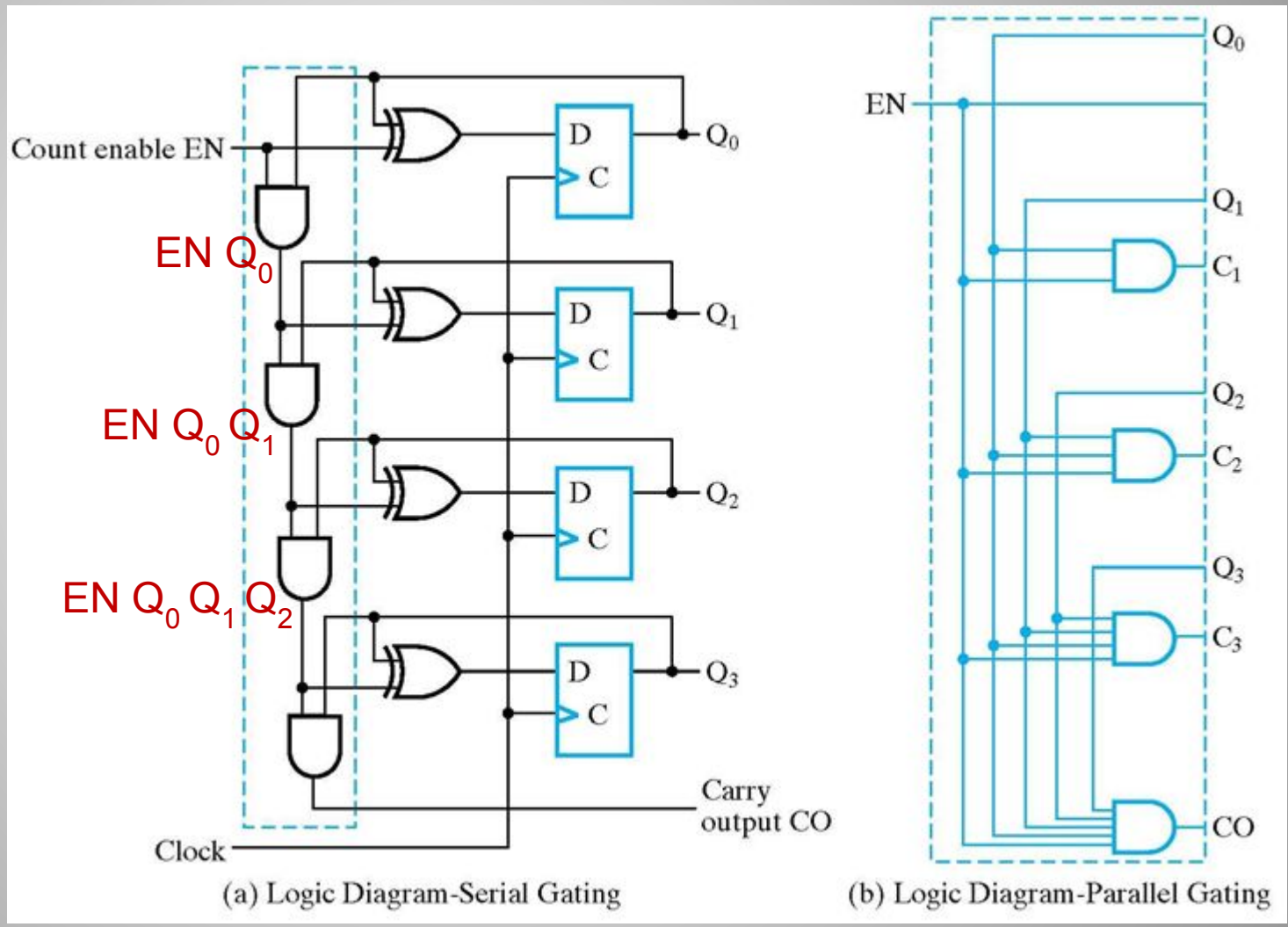
Synchronous Binary Counters

- Synchronous counter
  - A common clock triggers all flip-flops simultaneously
- Binary counter's counting sequence e.g. 0000, 0001, 0010, …, 1111, 0000, 0001, …
  - Q_0 is toggled every clock cycle
  - Q_i needs to be toggled in the next state if Q_0 to Q_{i-1} are all one in the present state (e.g. 0011 -> 0100)

Synchronous Binary Counter Design using D Flip-flops

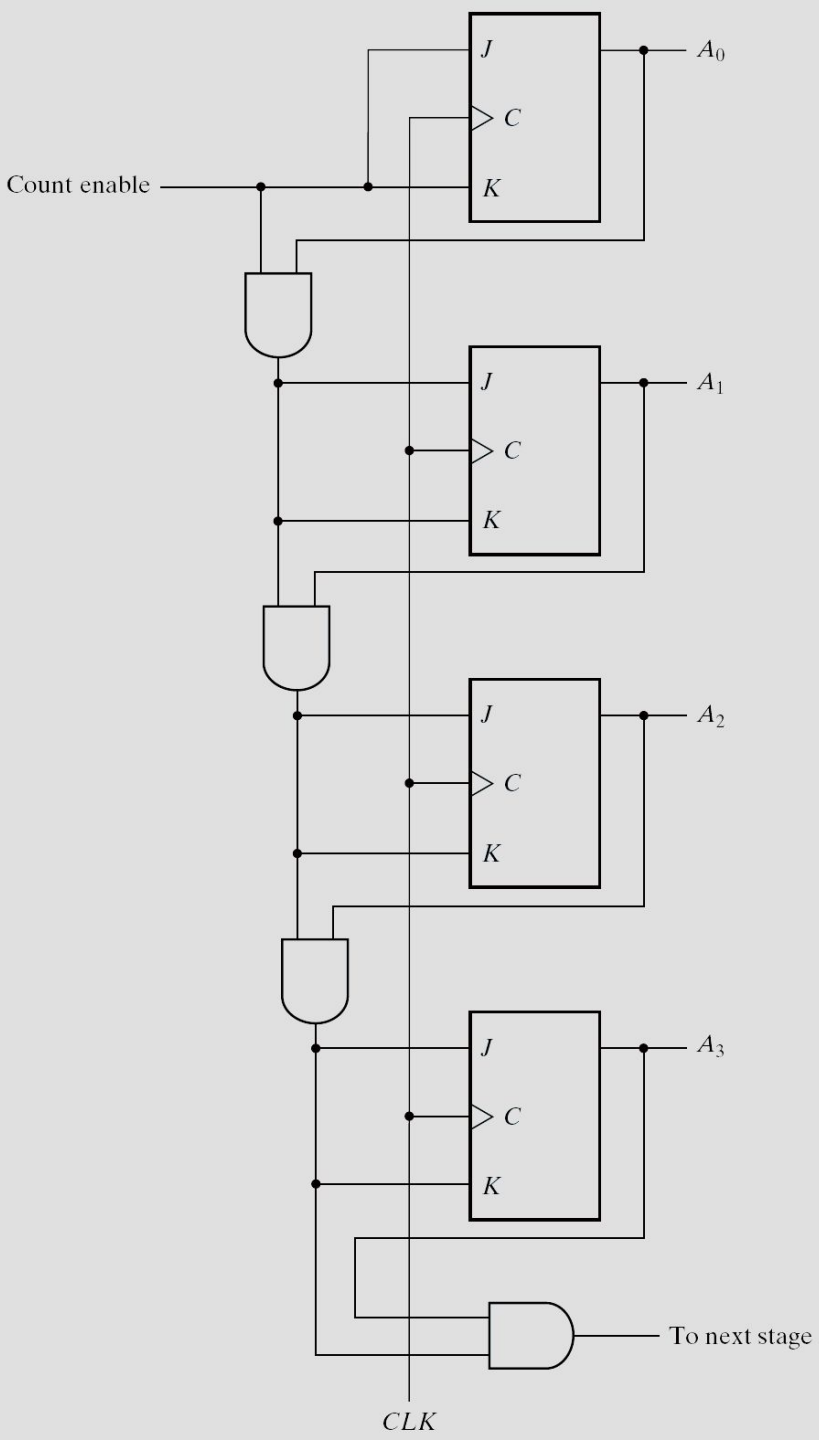


- with *count enable EN* and *carry output CO*



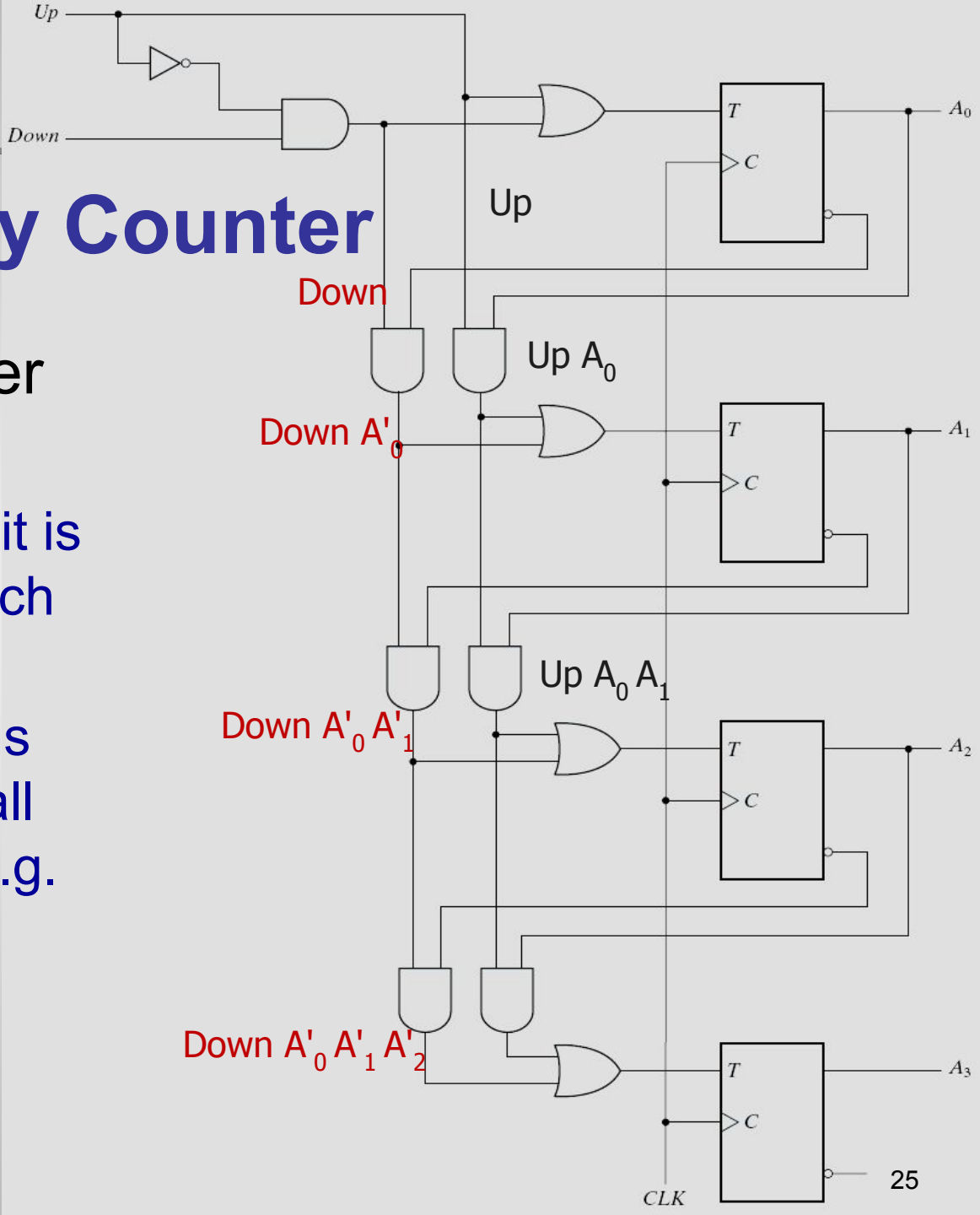
A 4-bit synchronous binary counter using JK flip-flops

(Exercise: Construct an 8-bit synchronous binary counter using two instances of the 4-bit binary counter shown)



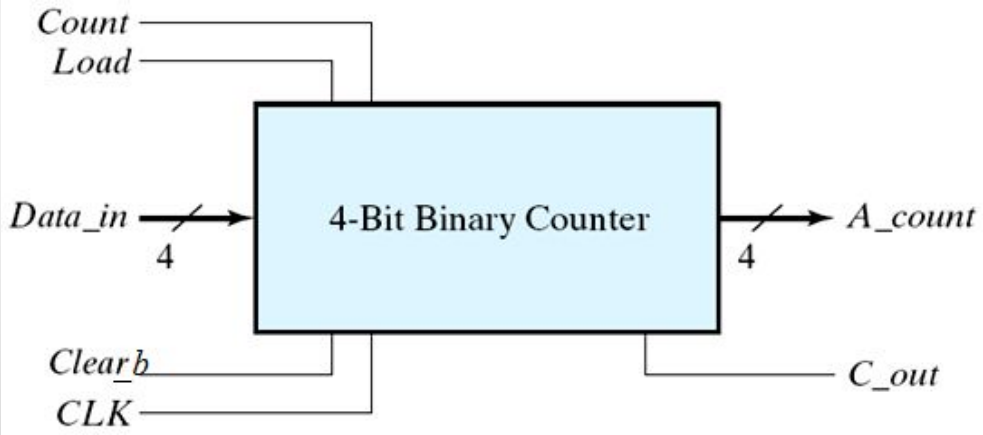
Up-Down Binary Counter

- For a down counter
  - Counts downward
  - Least significant bit is complemented each time
  - Other bit position is complemented if all lower bits are 0 (e.g. 1000→0111)
- A 4-bit up-down binary counter is shown:

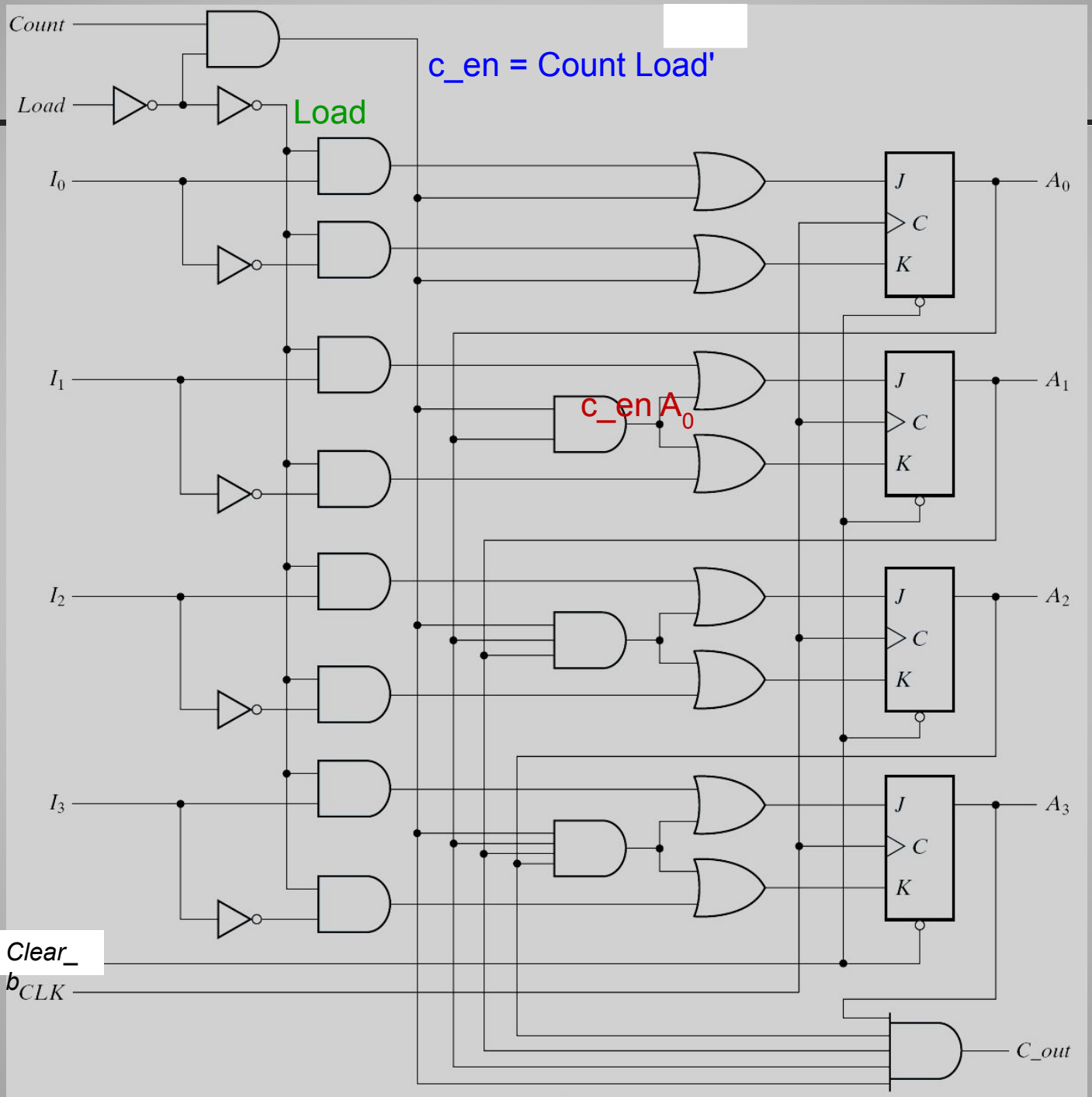


Binary Counter with Parallel Load

- It is desirable if we may load an initial count and/or clear a counter
- E.g. 4-bit binary counter w/ parallel load and active-low clear

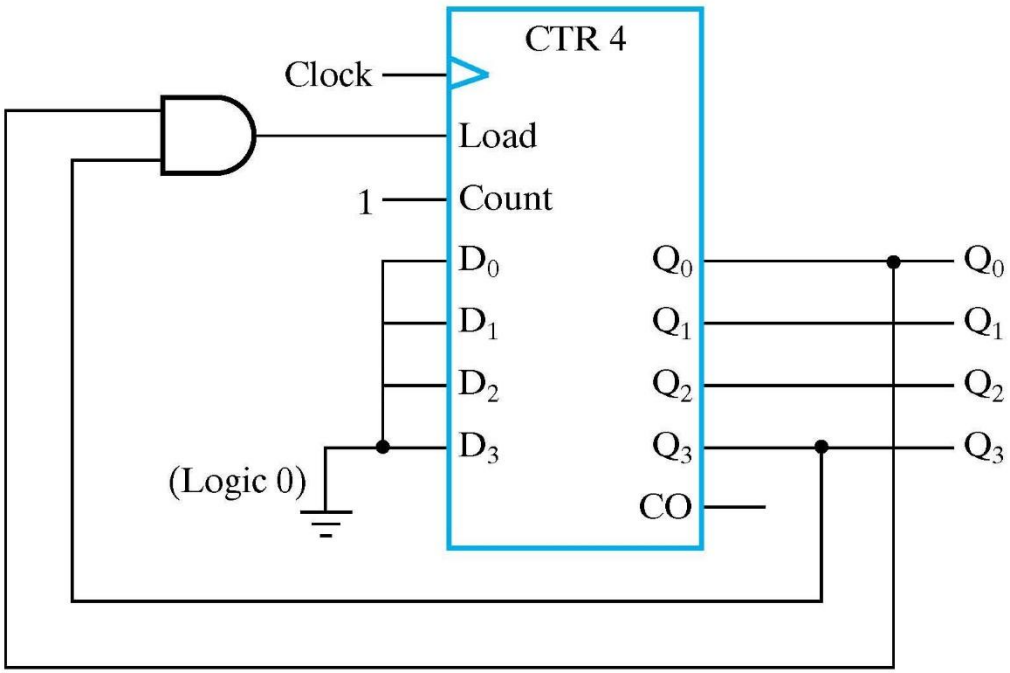


Clear_b	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change


Count
Load
c_en = Count Load'
Load
I_0
I_1
I_2
I_3
c_en A_0
J
C
K
A_0
A_1
A_2
A_3
Clear_b
CLK
C_out

BCD Counter

- Counts from binary 0 to 9
  (0000 → 0001 → … → 1001 → 0000 → …)
- Can be implemented using a 4-bit binary counter:



Q. Can you explain how it works?

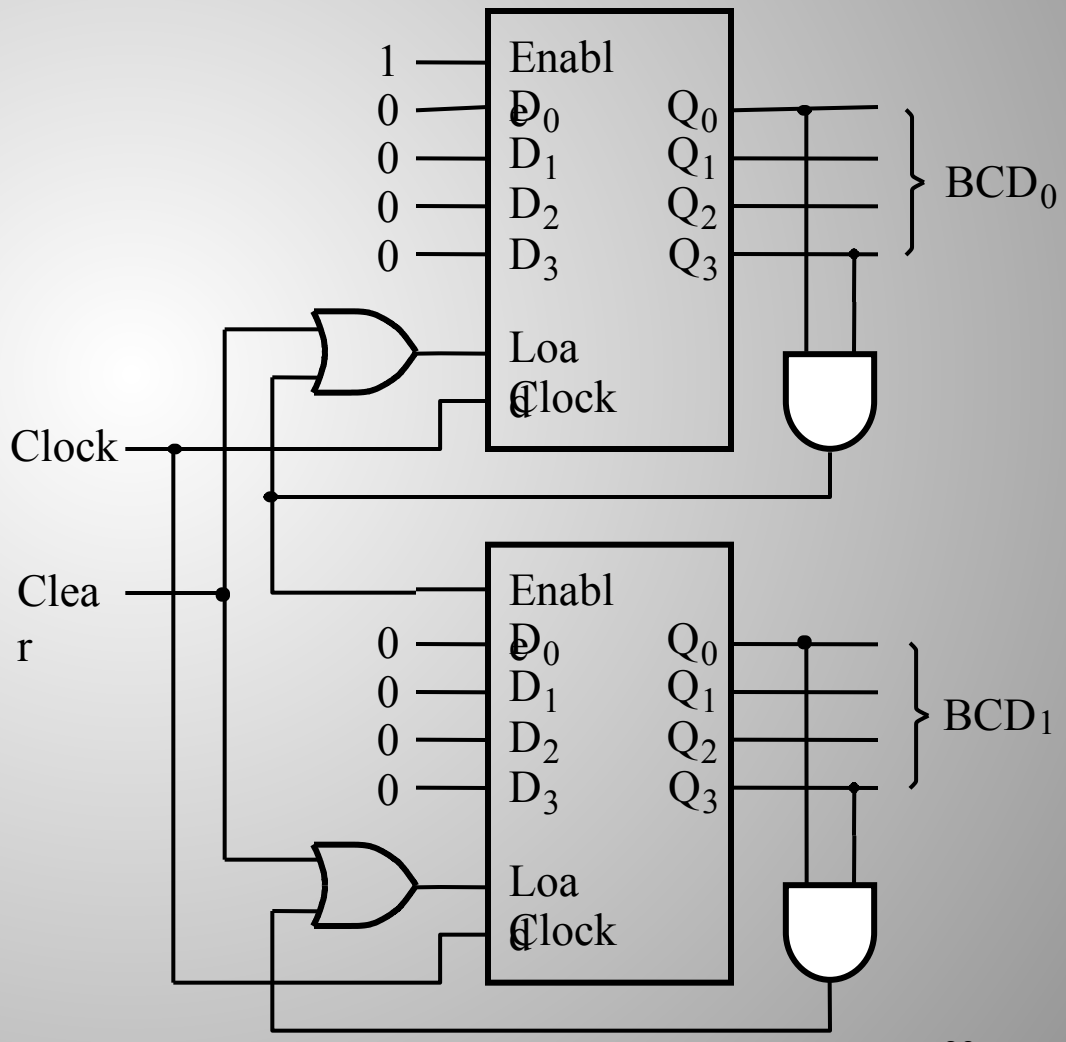
BCD Counter

- Alternatively, we can construct a BCD counter from scratch following the standard procedure.

Exercise: Finish the design of a BCD counter using D flip-flops.

Example

. Construct a 2-digit BCD counter using two 4-bit binary counters



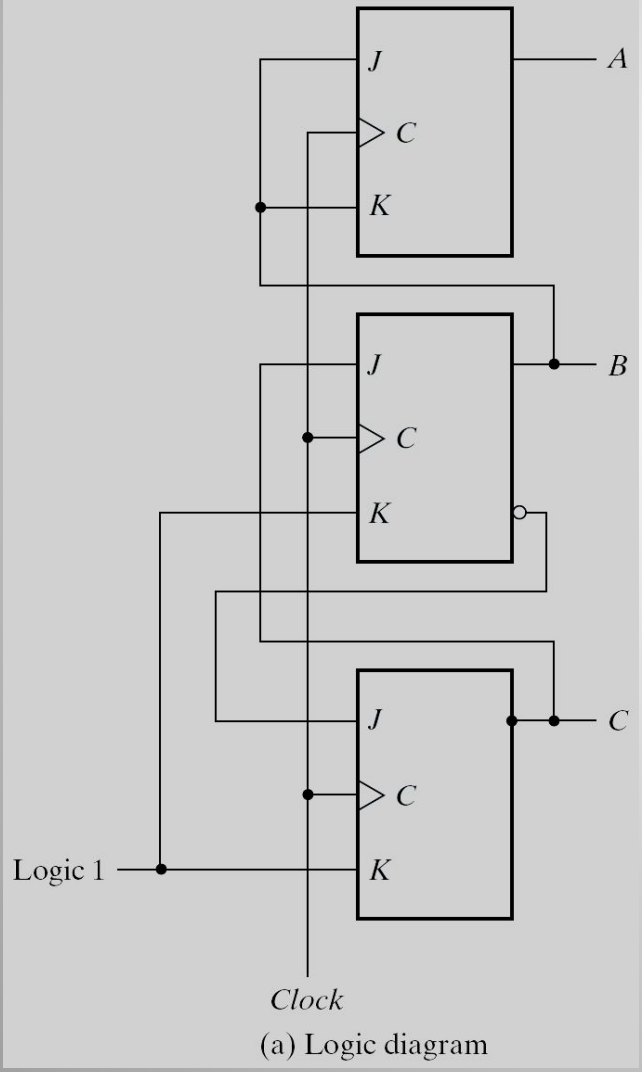
Counter for Arbitrary Count Sequence

- Constructed by following standard procedure for sequential circuit design.
- E.g. Construct a counter that counts in the sequence 000, 001, 010, 100, 101, 110 repeatedly.

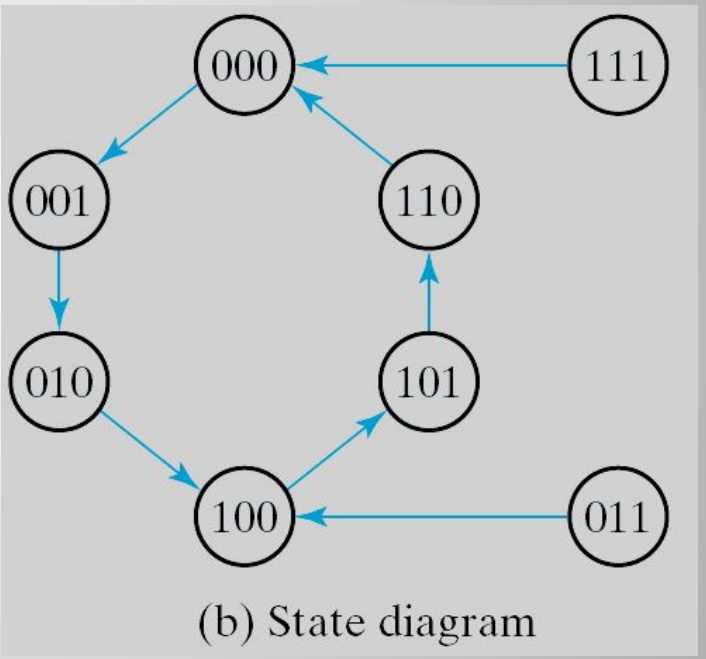
Table 6.7
State Table for Counter

Present State			Next State			Flip-Flop Inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

States 011 and 111 are unused.
If we simplify assuming don't care conditions for their next states, we get:



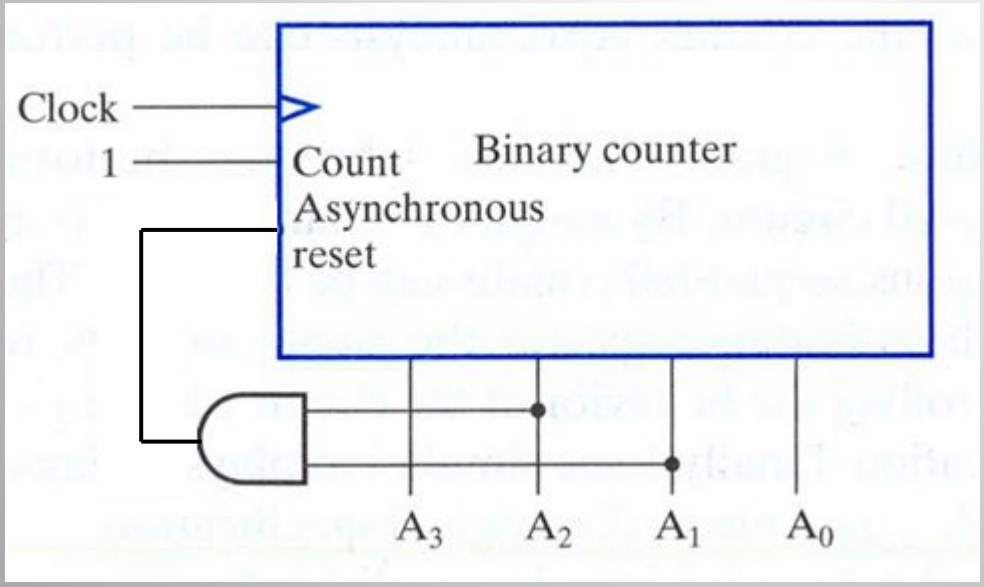
(a) Logic diagram



(b) State diagram

Synchronization Pitfall

- A student designed a counter which is supposed to count 0,1,2,3,4,5,6,0,…
- What is the potential problem of this design?



This is an asynchronous circuit

```
// 4-bit Left Shift Register with Reset

module srg_4_r_v (CLK, RESET, SI, Q,SO);
   input CLK, RESET, SI;
   output [3:0] Q;
   output SO;

reg [3:0] Q;

assign SO = Q[3];

always@(posedge CLK or posedge RESET)
begin
   if (RESET)
      Q <= 4'b0000;
   else
      Q <= {Q[2:0], SI};
end
endmodule
```

```
// 4-bit Binary Counter with Reset

module count_4_r_v (CLK, RESET, EN, Q, CO);
   input CLK, RESET, EN;
   output [3:0] Q;
   output CO;

reg [3:0] Q;

assign CO = (count == 4'b1111 && EN == 1'b1) ? 1 : 0;
always@(posedge CLK or posedge RESET)
   begin
   if (RESET)
     Q <= 4'b0000;
   else if (EN)
     Q <= Q + 4'b0001;
   end
endmodule
```