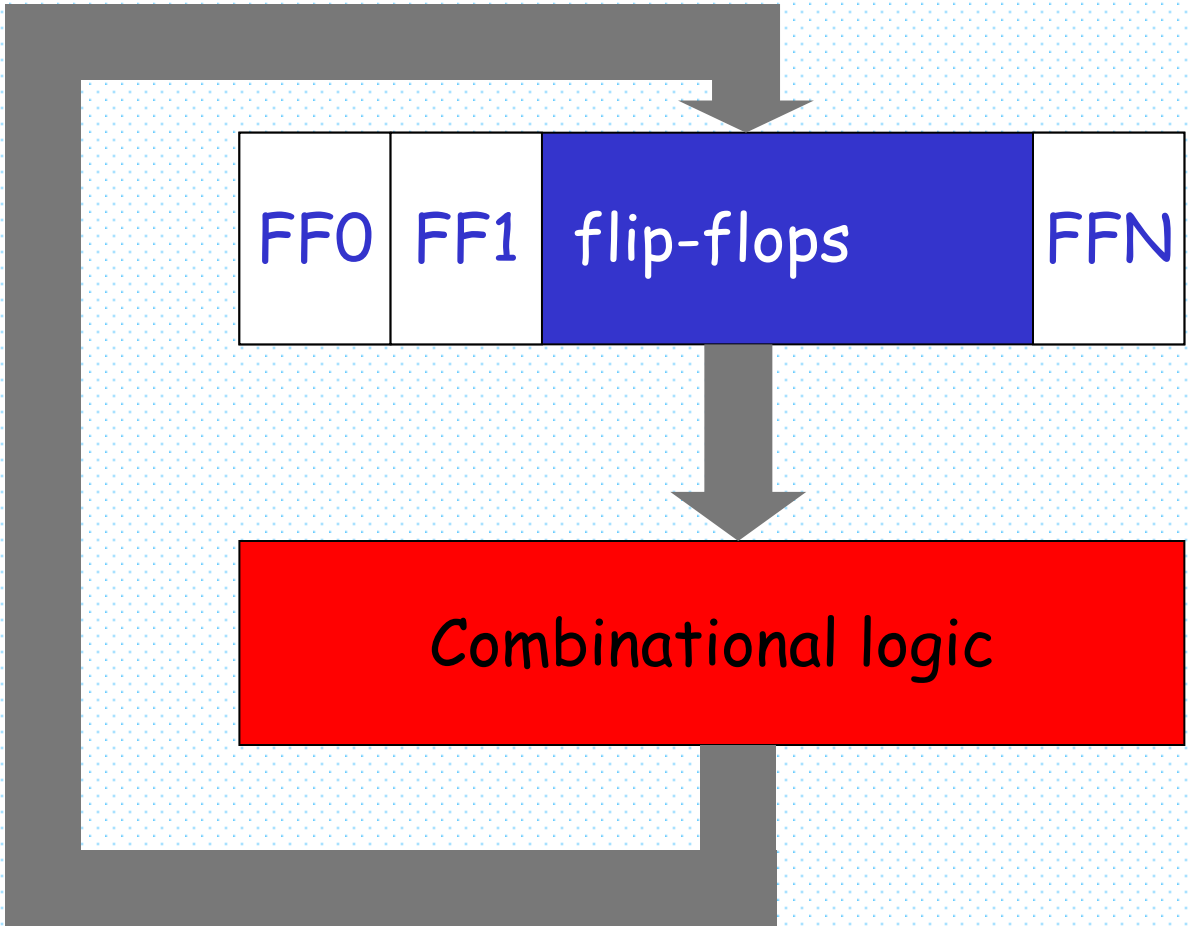
Registers & Counters

Registers

- Registers like counters are clocked sequential circuits
- A register is a group of flip-flops
  - Each flip-flop capable of storing one bit of information
  - An n-bit register
    - consists of n flip-flops
    - capable of storing n bits of information
  - besides flip-flops, a register usually contains combinational logic to perform some simple tasks
  - In summary
    - flip-flops to hold information
    - combinational logic to control the state transition

Counters

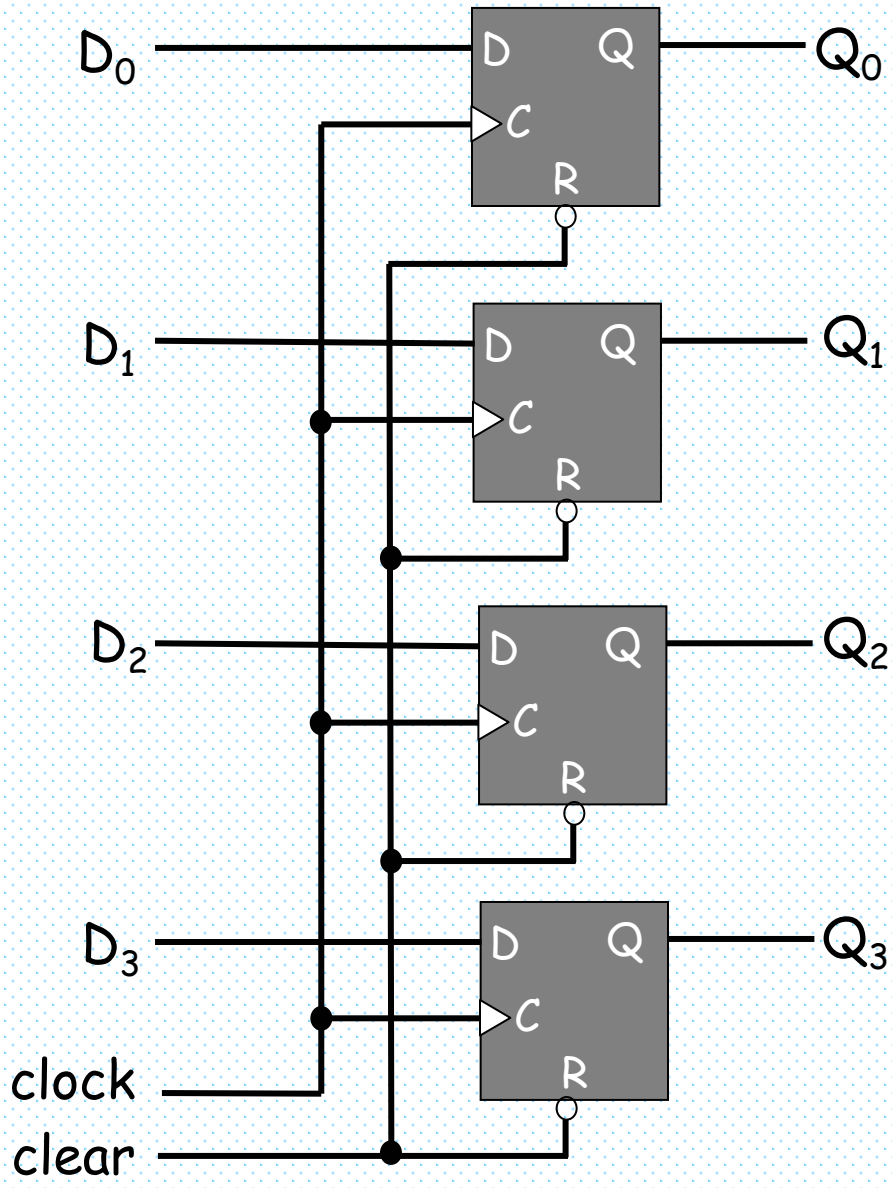
- A counter is essentially a register that goes through a predetermined sequence of states

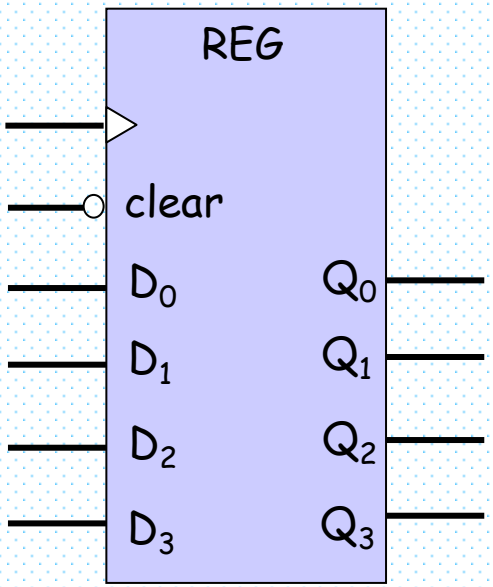


Uses of Registers and Counters

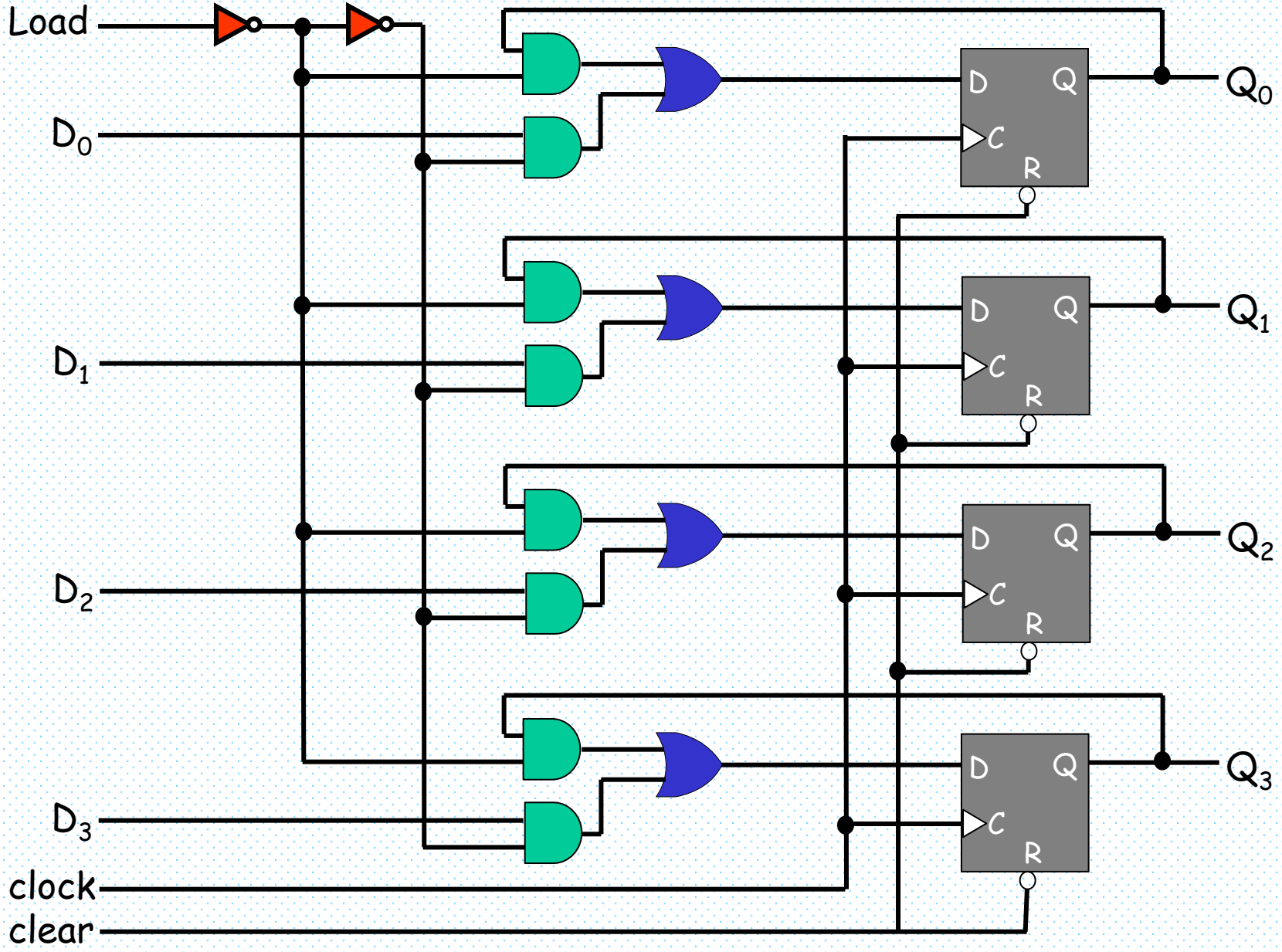
- Registers are useful for storing and manipulating information
  - internal registers in microprocessors to manipulate data
- Counters are extensively used in control logic
  - PC (program counter) in microprocessors

4-bit Register

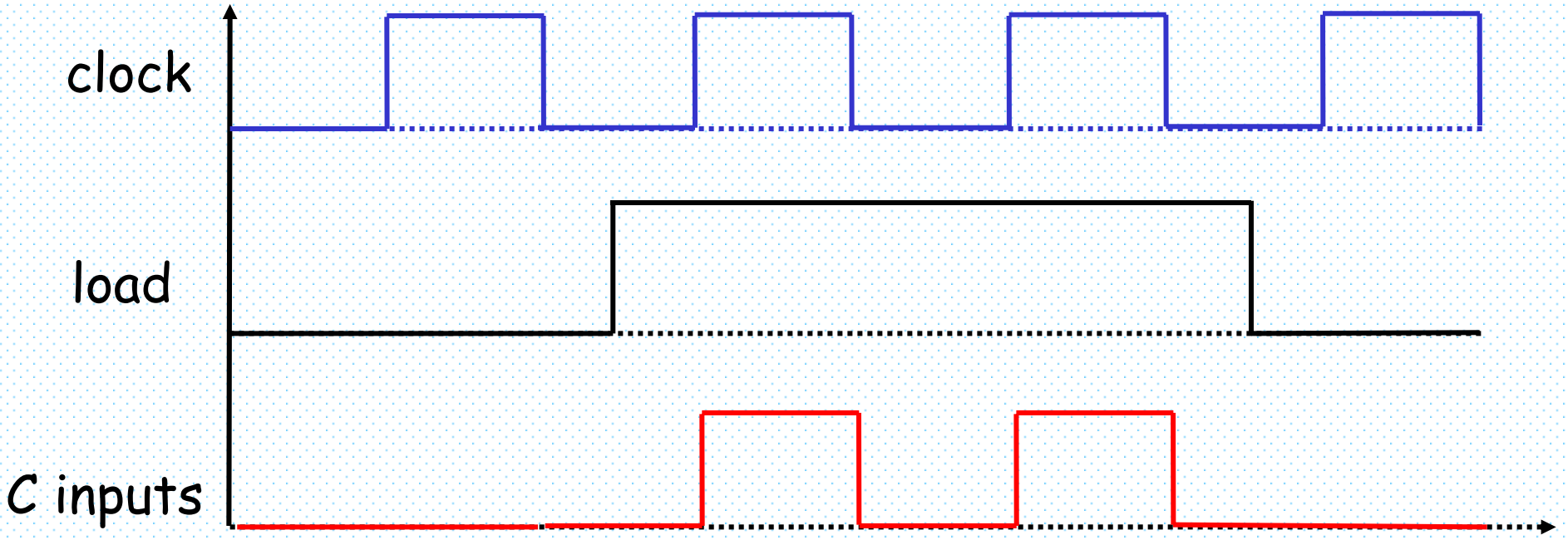




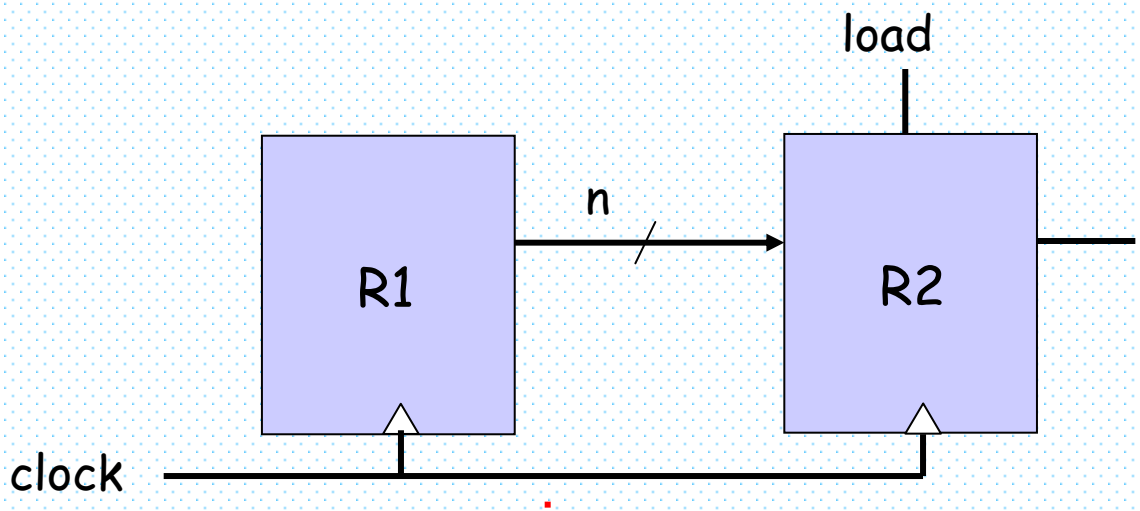
Register with Parallel Load



Loading Register



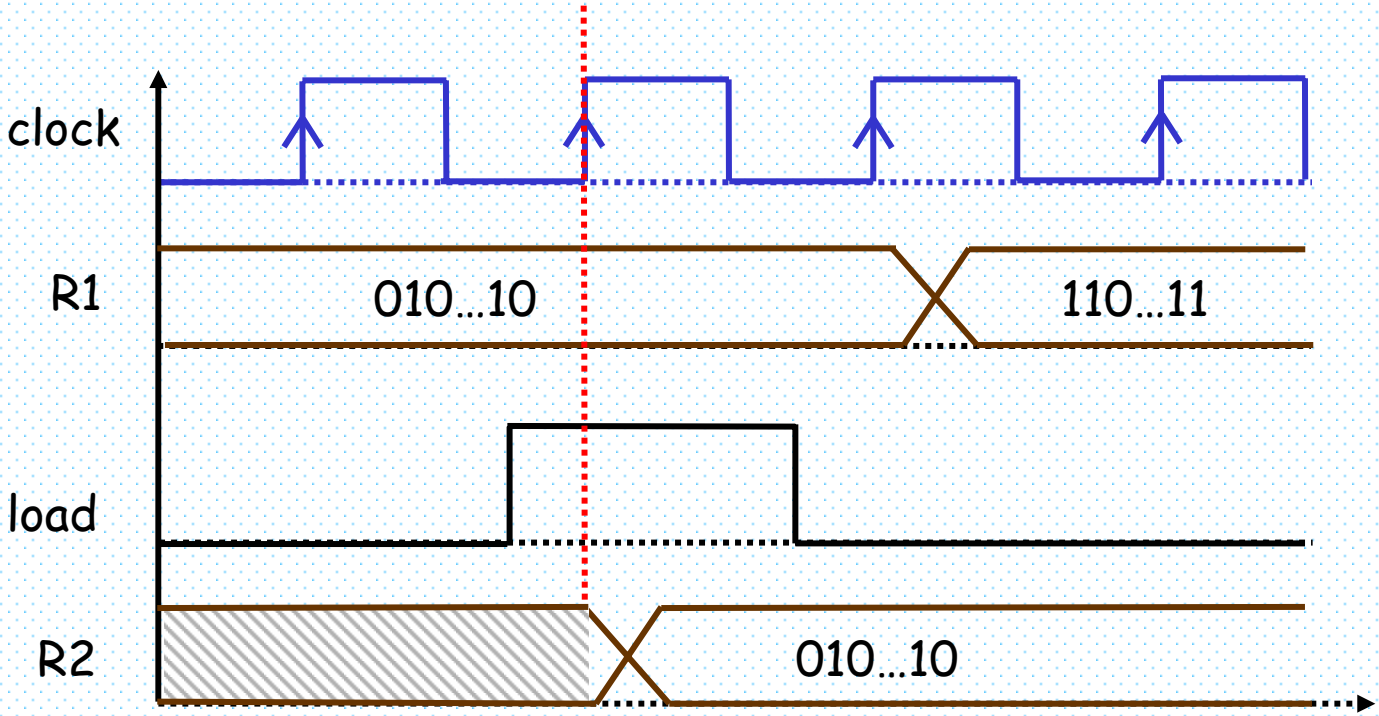
Register Transfer - 1



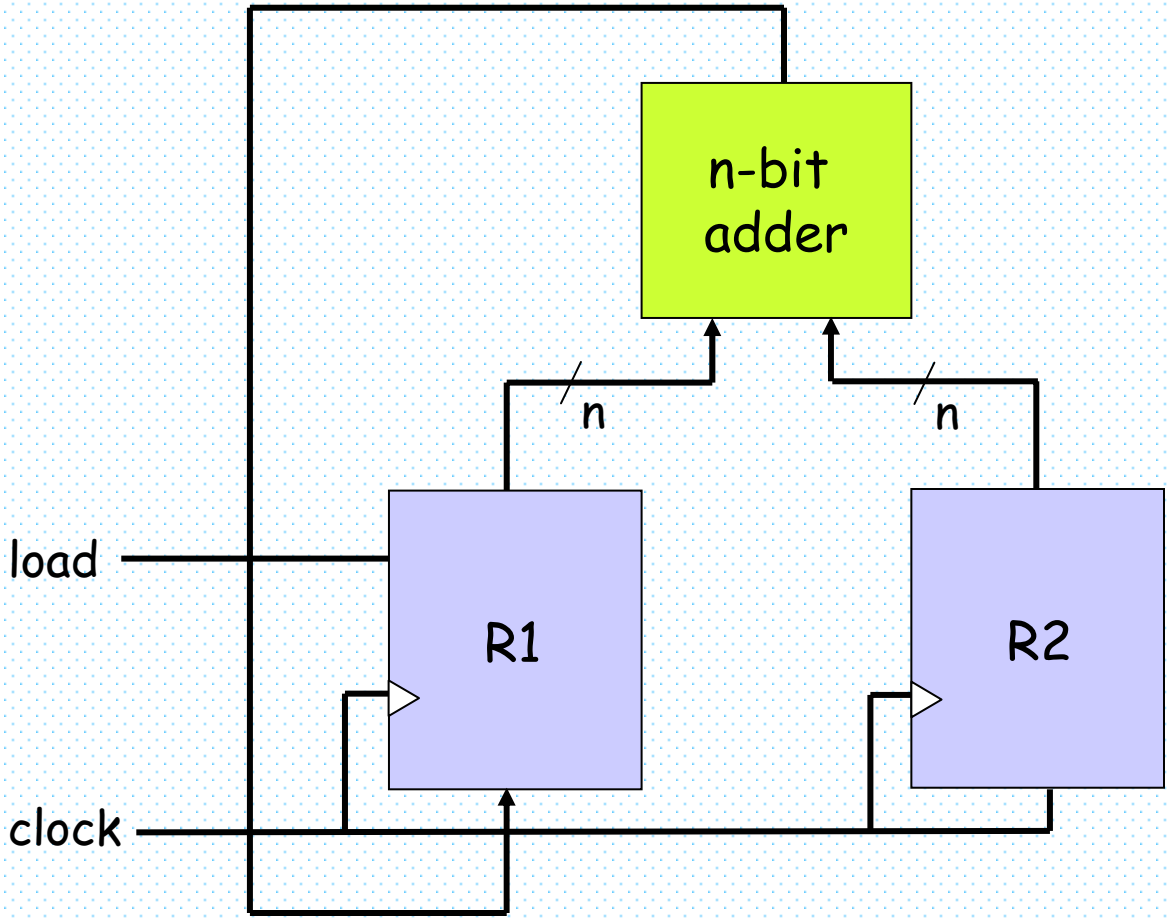
Instruction

$R2 \leftarrow R1$

Load R1 content into R2

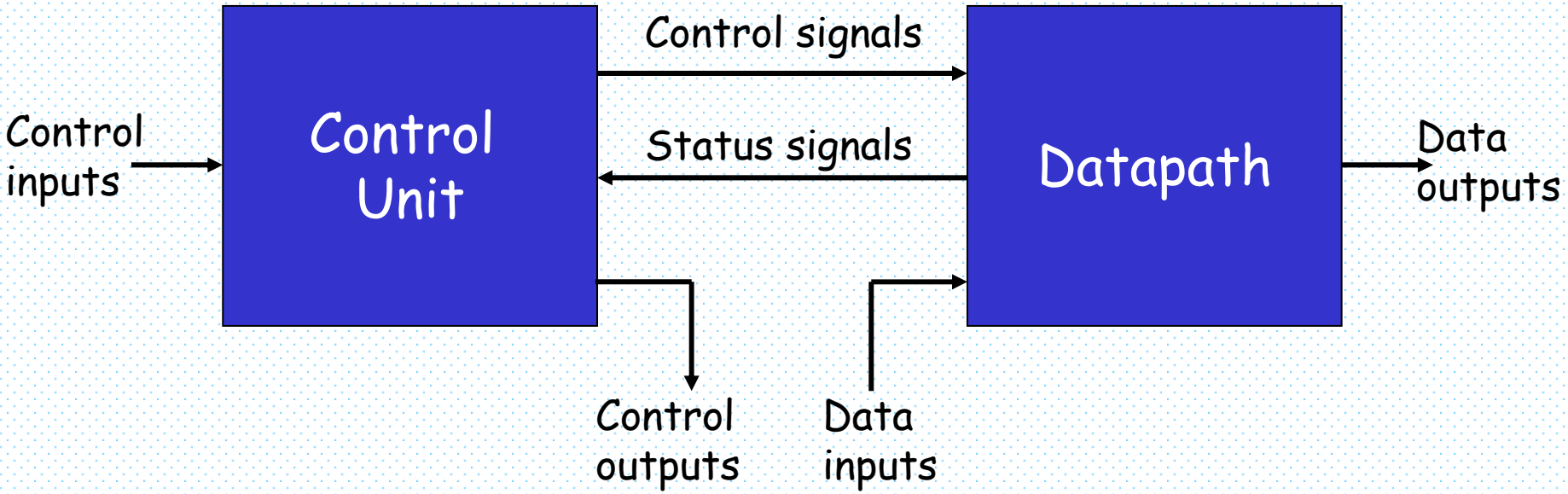


Register Transfer - 2



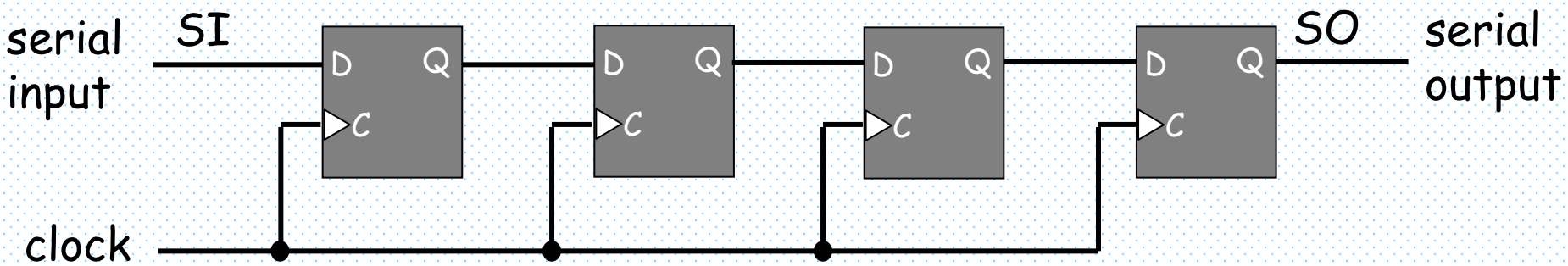
$R1 \leftarrow R1 + R2$

Datapath & Control Unit



Shift Registers

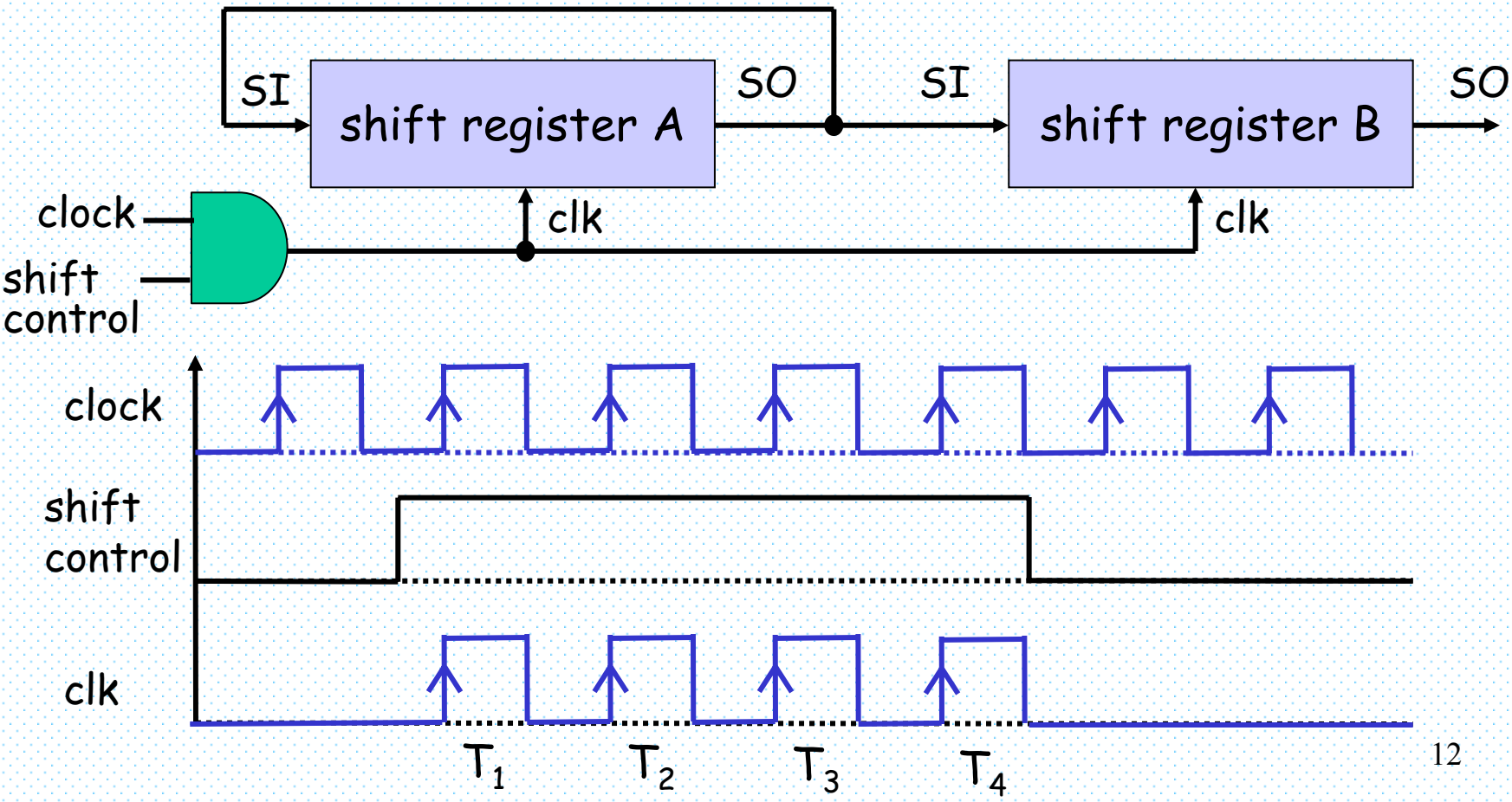
- A register capable of shifting its information in one or both directions
  - Flip-flops in cascade



- The current state can be output in n clock cycles

Serial Mode

- A digital system is said to operate in serial mode when information is transferred and manipulated one bit a time.



Serial Transfer

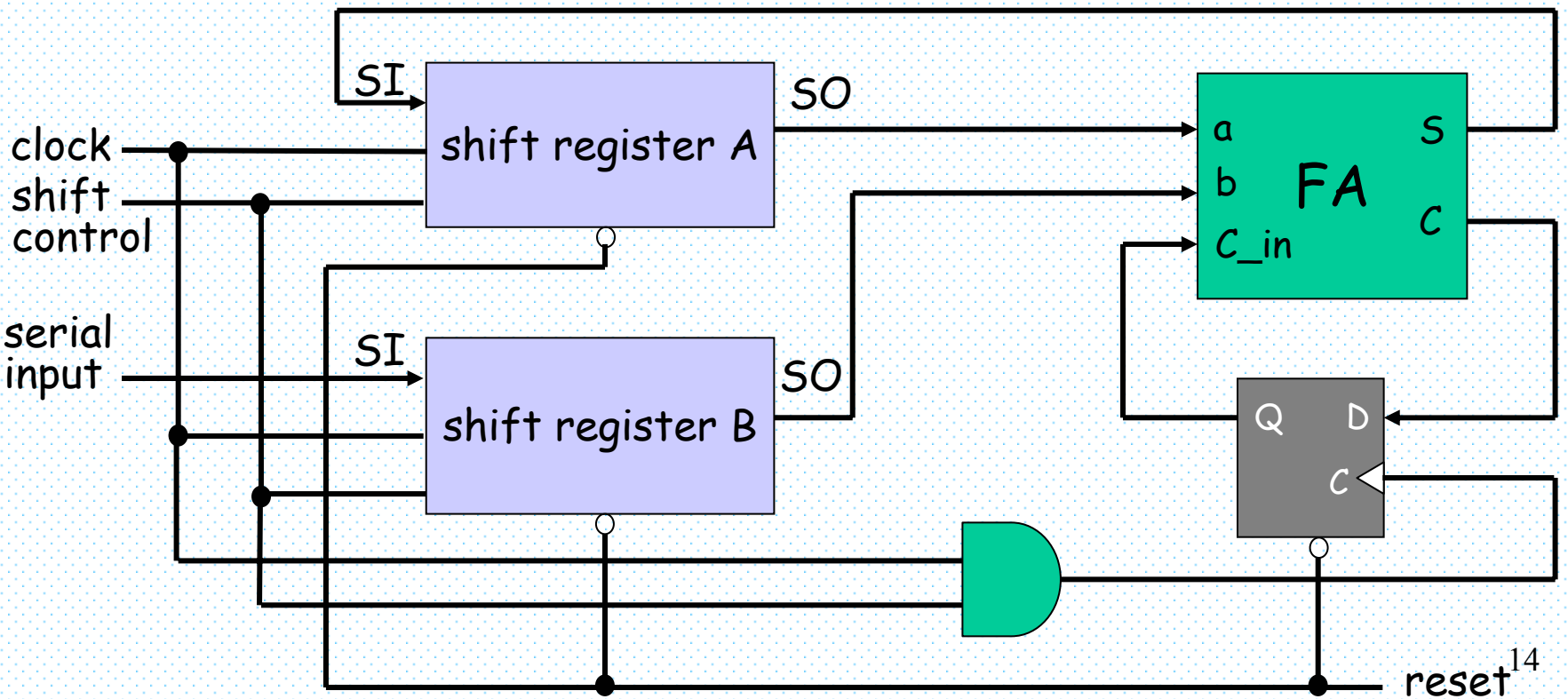
- Suppose we have two 4-bit shift registers

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

$B \leftarrow A$

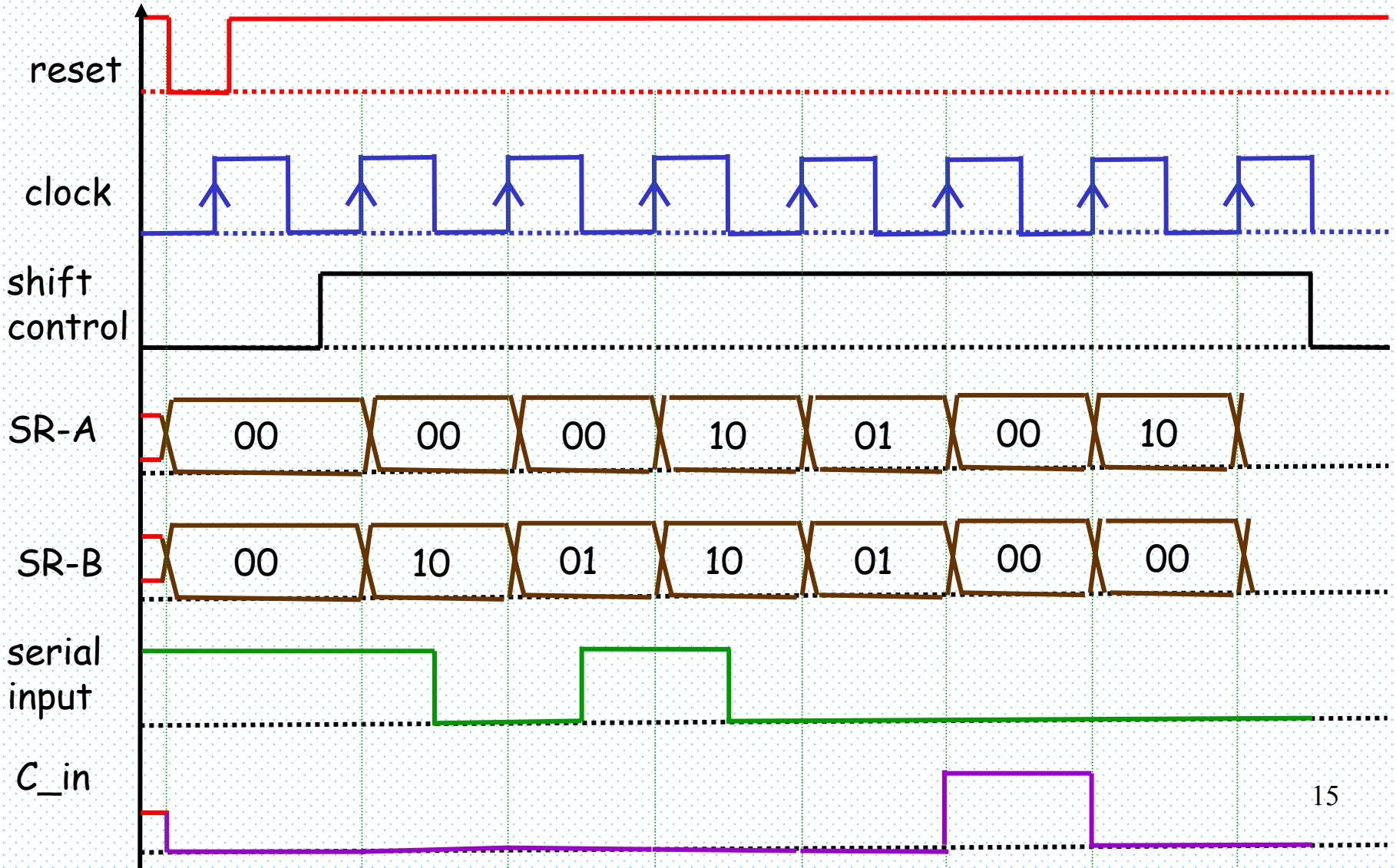
Serial Addition

- In digital computers, operations are usually executed in parallel, since it is faster
- Serial mode is sometimes preferred since it requires less equipment



Example: Serial Addition

- A and B are 2-bit shift registers



Designing Serial Adder - 1

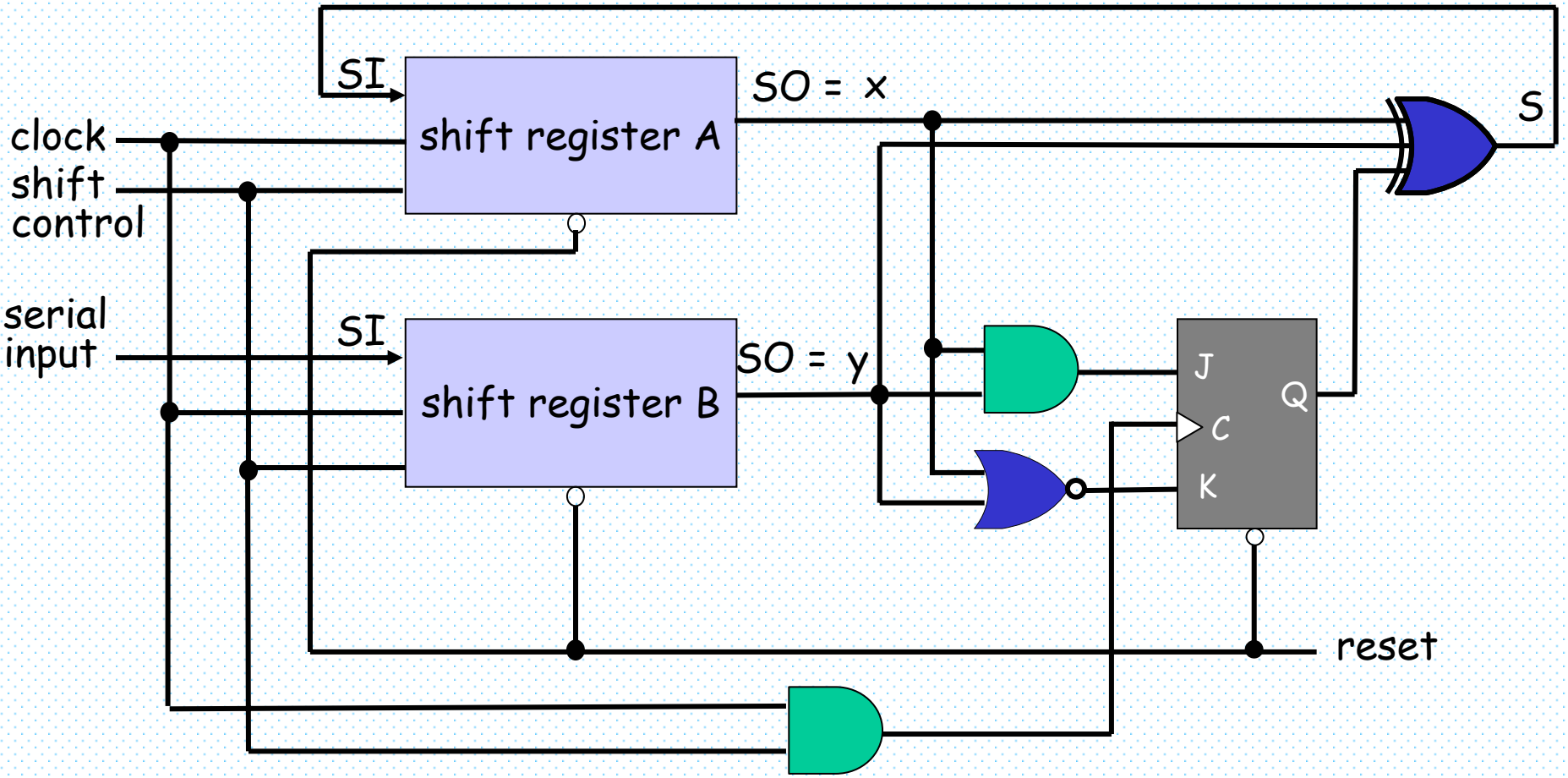
$$Q(t+1) = JQ' + K'Q$$

Present state	Inputs		Next state	Output	Flip-flop inputs	
Q	x	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

$J_Q = xy$ $K_Q = x'y' = (x + y)'$ $S = x \oplus y \oplus Q$

Designing Serial Adder - 2

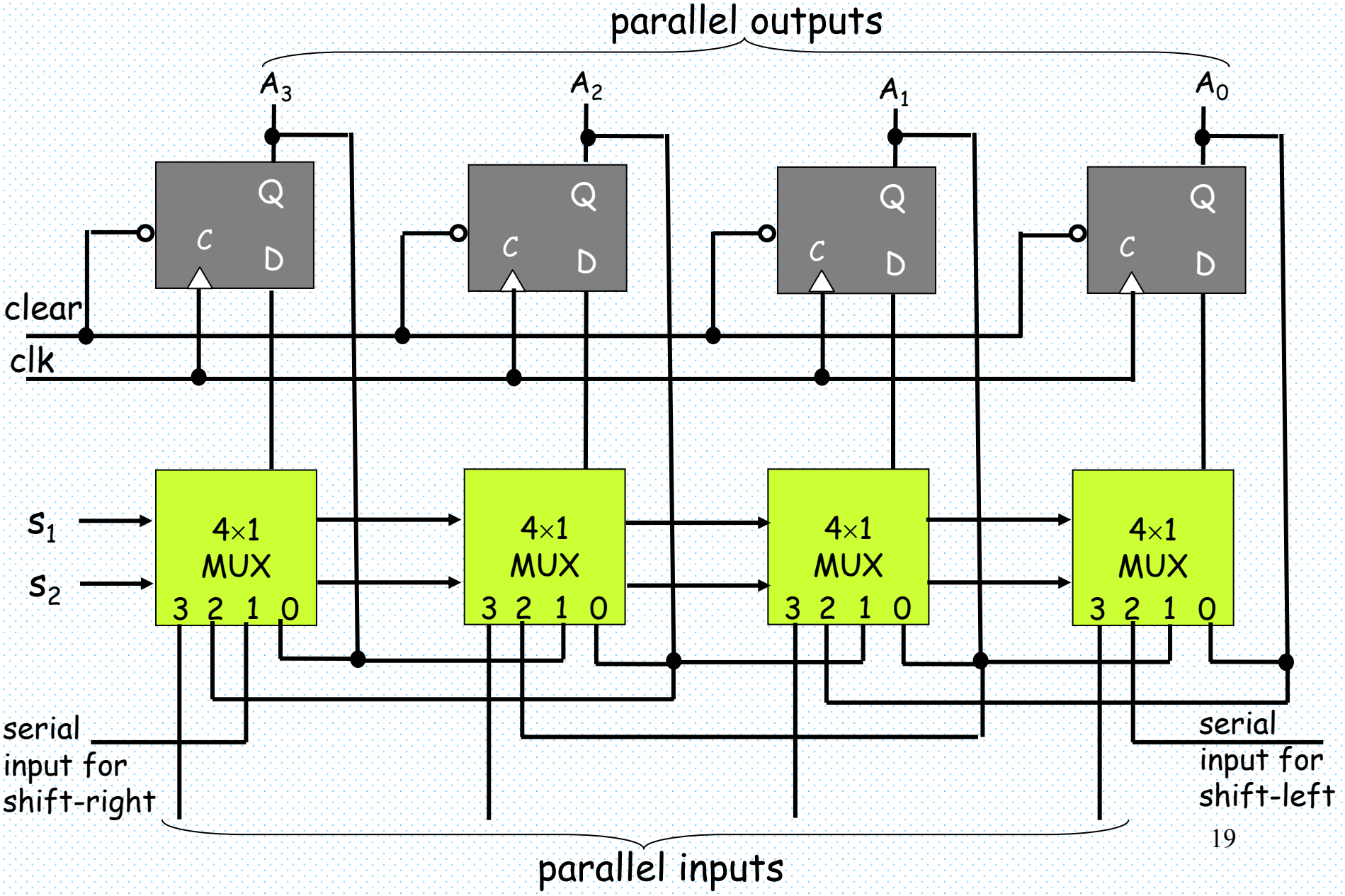
$J_Q = xy$ $K_Q = x'y' = (x + y)'$ $S = x \oplus y \oplus Q$



Universal Shift Register

- Capabilities:
  1. A *clear* control to set the register to 0.
  2. A *clock* input
  3. A *shift-right* control
  4. A *shift-left* control
  5. n input lines
  6. A *parallel-load* control
  7. n parallel output lines
  8. A *shift-control*

Universal Shift Register



Universal Shift Register

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

Counters

- A counter is basically a register that goes through a prescribed sequence of states upon the application of input pulses
  - input pulses are usually clock pulses
- Example: n-bit binary counter
  - count in binary from 0 to 2^n-1
- Classification
  1. Ripple counters
     - flip-flop output transition serves as *the* pulse to trigger other flip-flops
  2. Synchronous counters
     - flip-flops receive the same common clock as *the* pulse

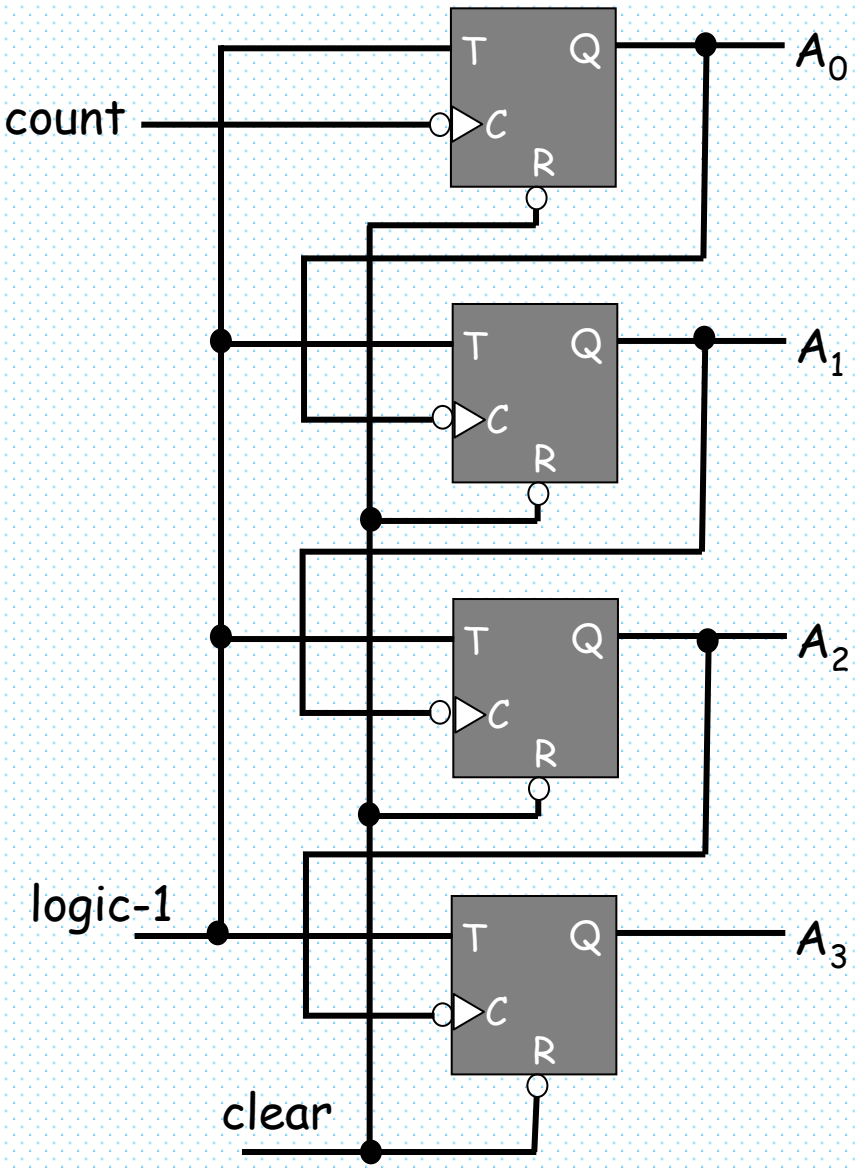
Binary Ripple Counter

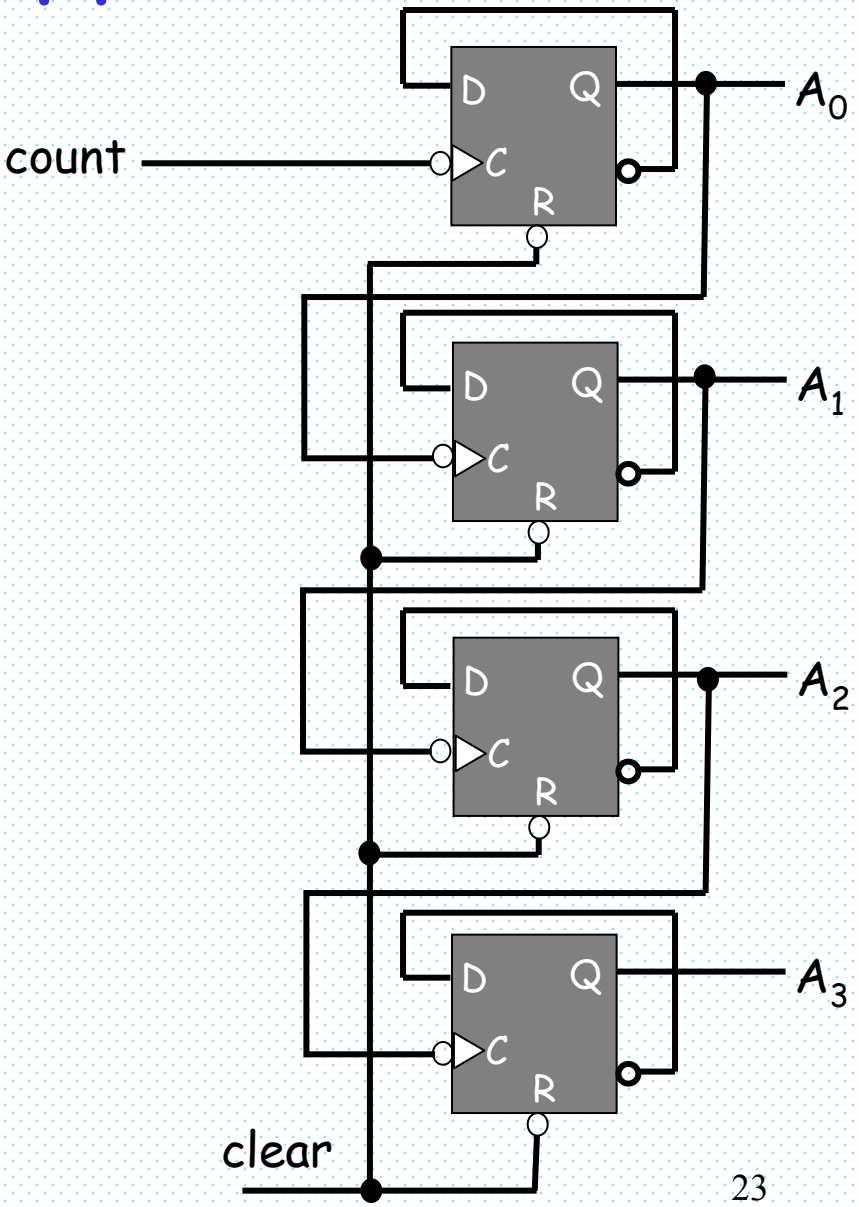
3-bit binary ripple counter

0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
0	0	0	0

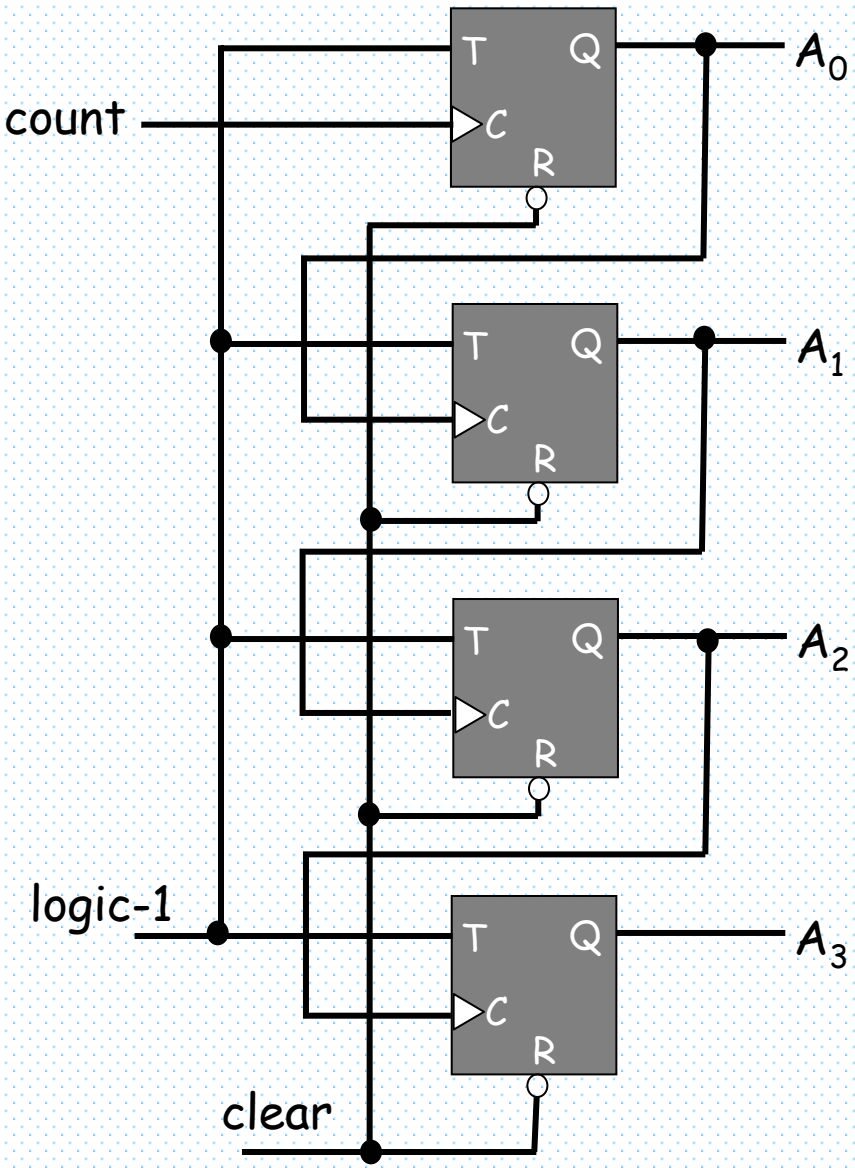
- Idea:
  - to connect the output of one flip-flop to the C input of the next high-order flip-flop
- We need "complementing" flip-flops
  - We can use T flip-flops to obtain complementing flip-flops or
  - JK flip-flops with its inputs are tied together or
  - D flip-flops with complement output connected to the D input.

4-bit Binary Ripple Counter





4-bit Binary Ripple Counter

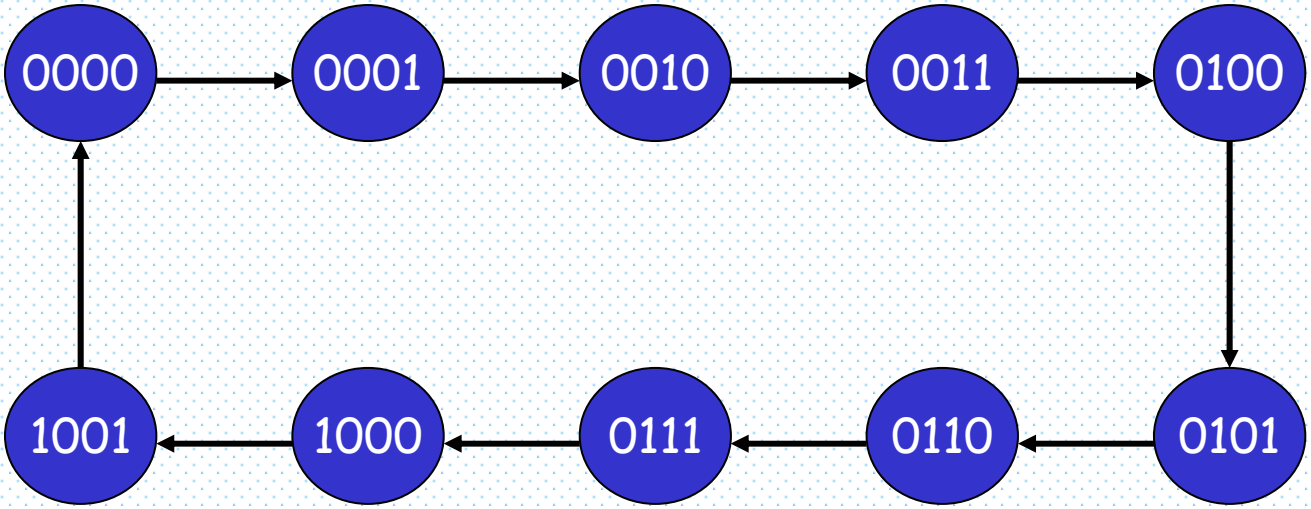


- Suppose the current state is `1100`
- What is the next state?
- A_0 = 1 (0 → 1)
- A_1 = 1 (0 → 1)
- A_2 = 0 (1→ 0)
- A_3 = 1
- next state: `1011`

- Binary count-down counter

BCD Ripple Counter

- State diagram

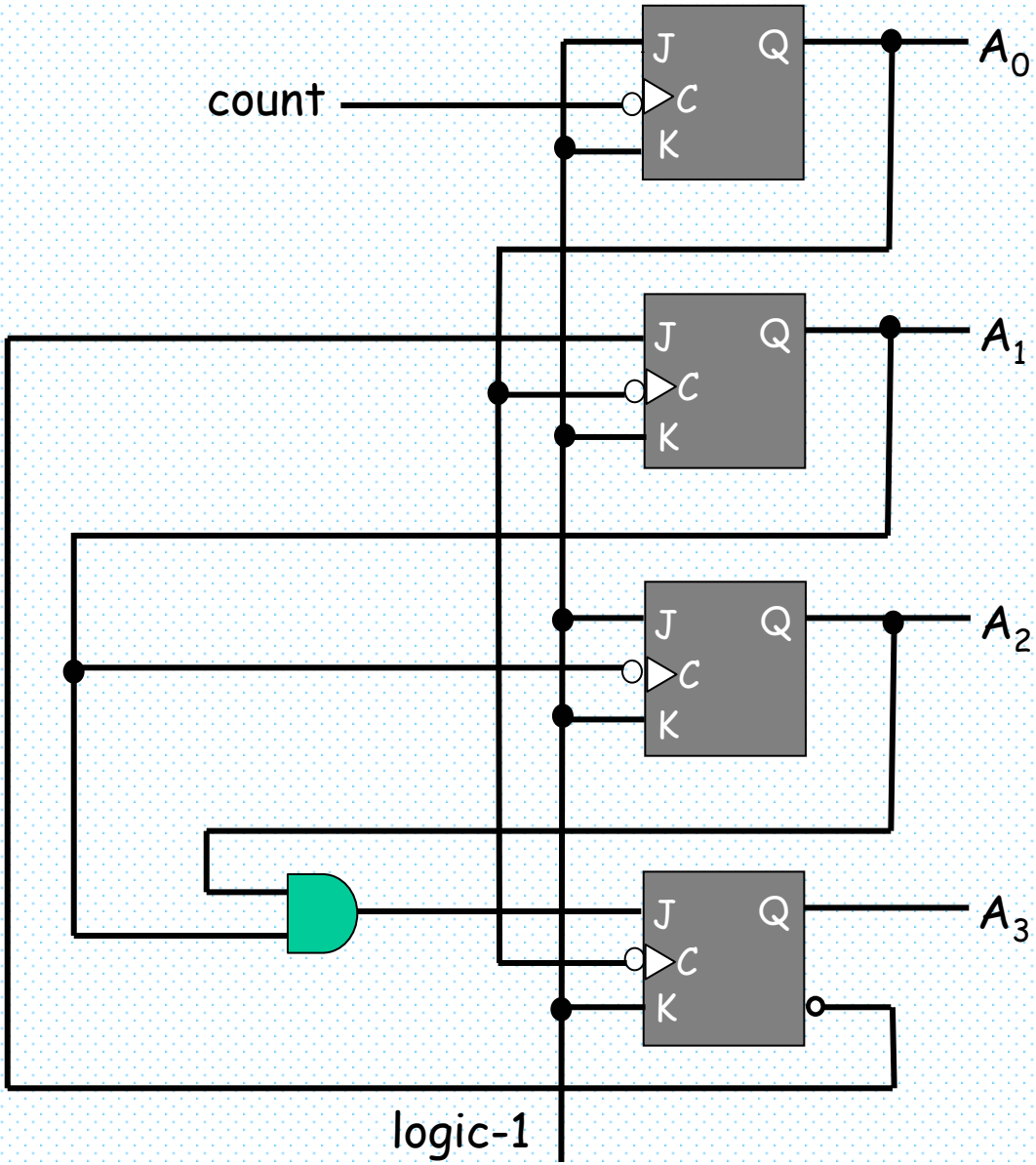


BCD Ripple Counter

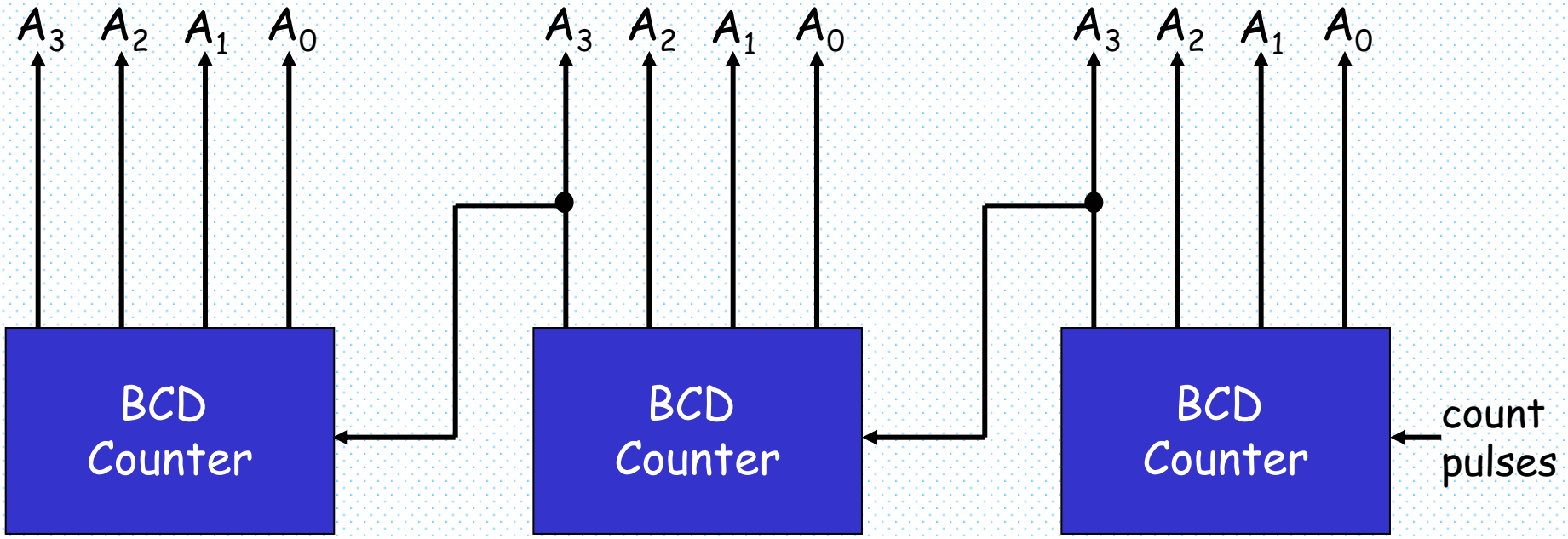
- State transitions

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
0	0	0	0

BCD Ripple Counter with JK FFs



Multi-digit BCD Counter

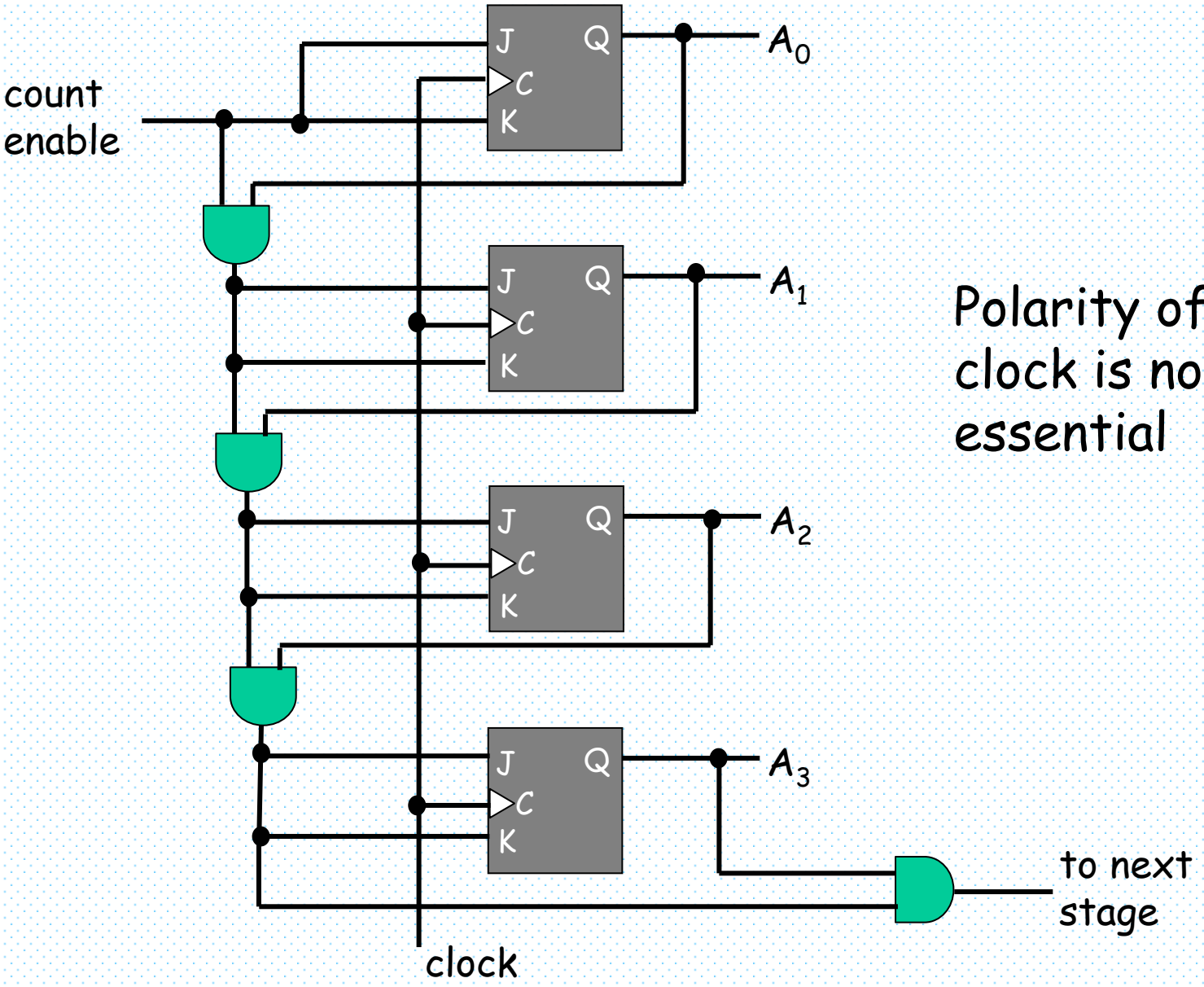


3-digit BCD counter

Synchronous Counters

- There is a common clock
  - that triggers all flip-flops simultaneously
  - If T = 0 or J = K = 0 the flip-flop does not change state.
  - If T = 1 or J = K = 1 the flip-flop does change state.
- Design procedure is so simple
  - no need for going through sequential logic design process
  - A_0 is always complemented
  - A_1 is complemented when $A_0 = 1$
  - A_2 is complemented when $A_0 = 1$ and $A_1 = 1$
  - so on

4-bit Binary Synchronous Counter

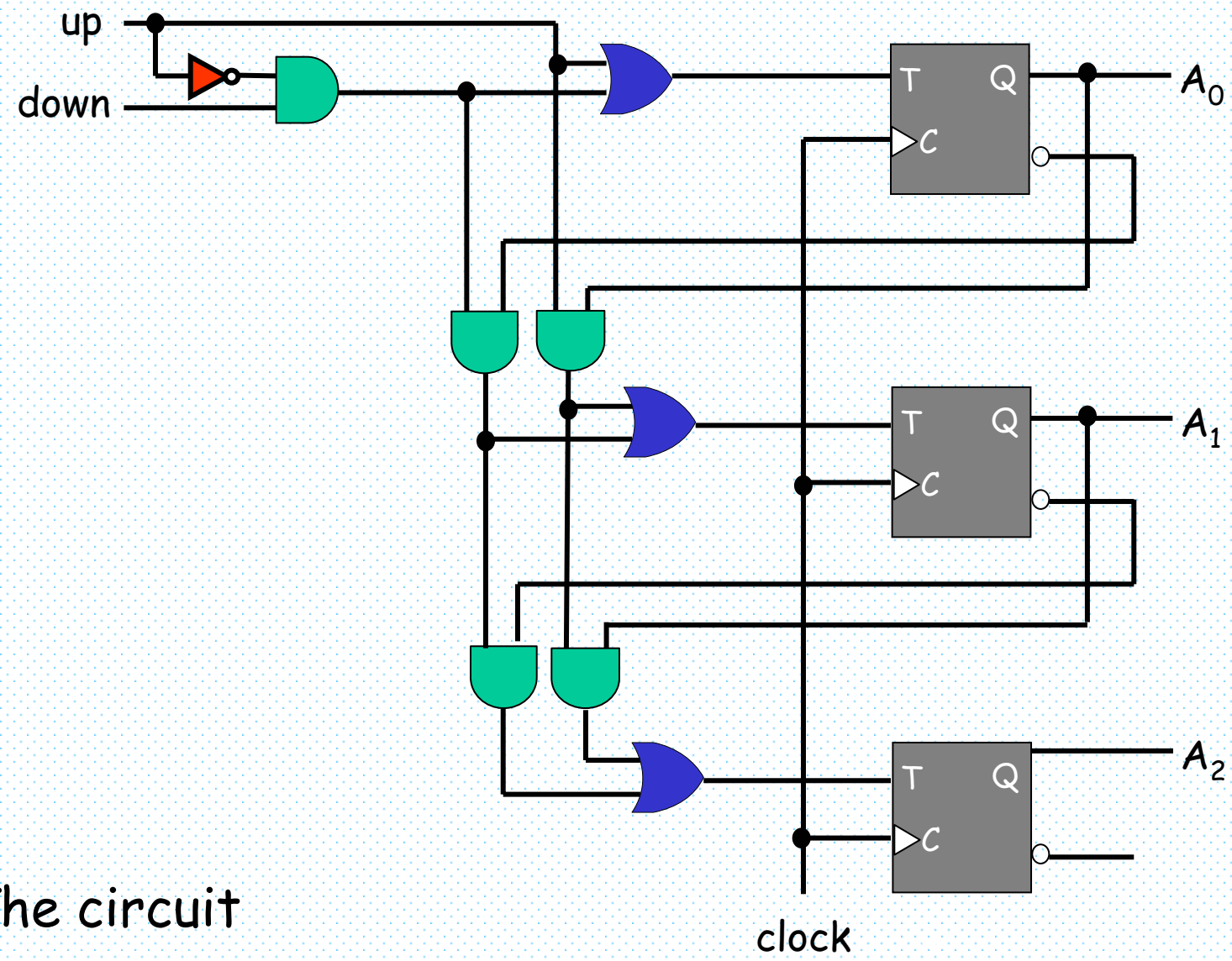


Polarity of the clock is not essential

Up-Down Binary Counter

- When counting downward
  - the least significant bit is always complemented (with each clock pulse)
  - A bit in any other position is complemented if all lower significant bits are equal to 0.
  - For example: 0100
    - Next state: 0011
  - For example: 1100
    - Next state: 1011

Up-Down Binary Counter



- The circuit

Synchronous BCD Counter

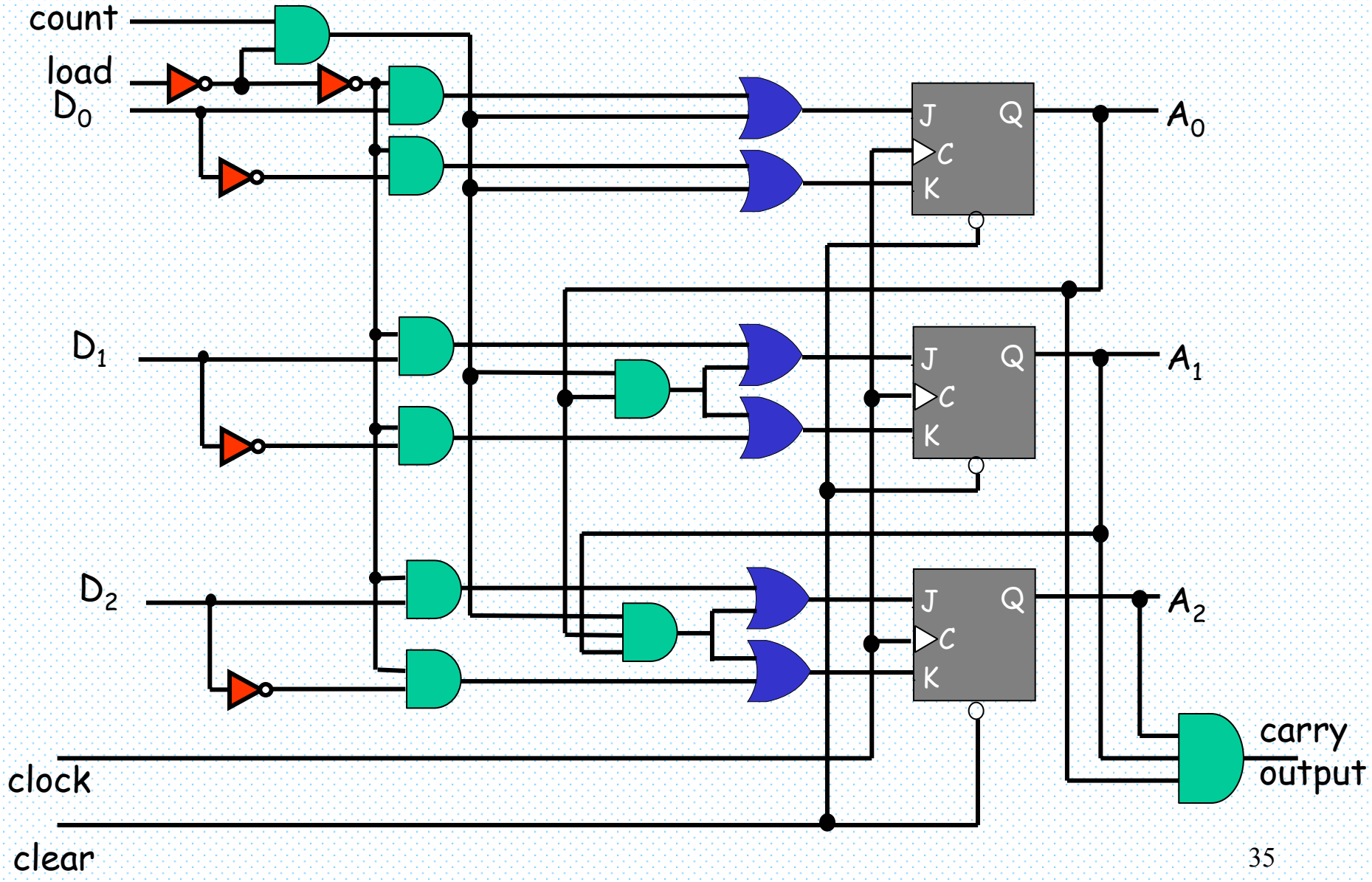
- Better to apply the sequential circuit design procedure

Present state				Next state				output	Flip-Flop inputs			
A_8	A_4	A_2	A_1	A_8	A_4	A_2	A_1	y	T_8	T_4	T_2	T_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

Synchronous BCD Counter

- The flip-flop input equations
  - $T_1 = 1$
  - $T_2 = A_8' A_1$
  - $T_4 = A_2 A_1$
  - $T_8 = A_8 A_1 + A_4 A_2 A_1$
- Output equation
  - $y = A_8 A_1$
- Cost
  - Four T flip-flops
  - four 2-input AND gates
  - one OR gate
  - one inverter

Binary Counter with Parallel Load



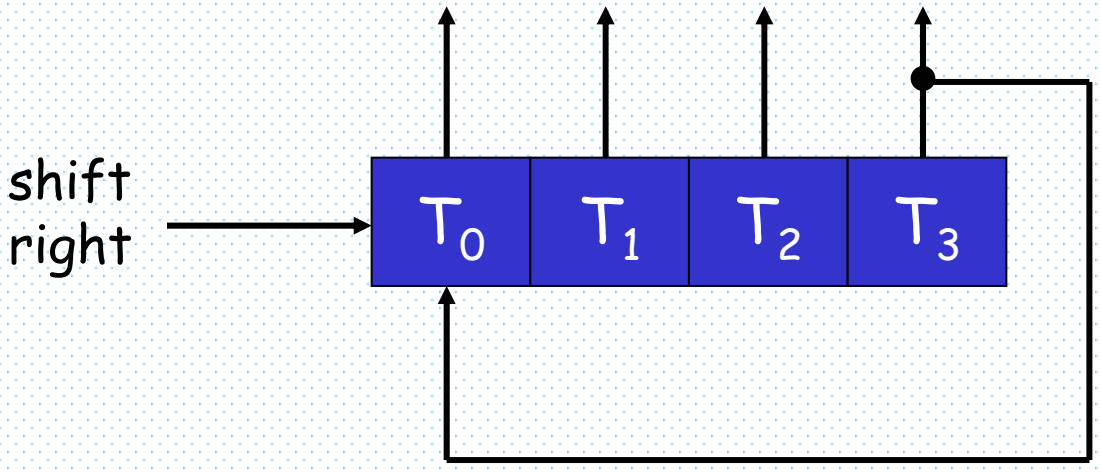
Binary Counter with Parallel Load

Function Table

clear	clock	load	Count	Function
0	X	X	X	clear to 0
1	↑	1	X	load inputs
1	↑	0	1	count up
1	↑	0	0	no change

Other Counters

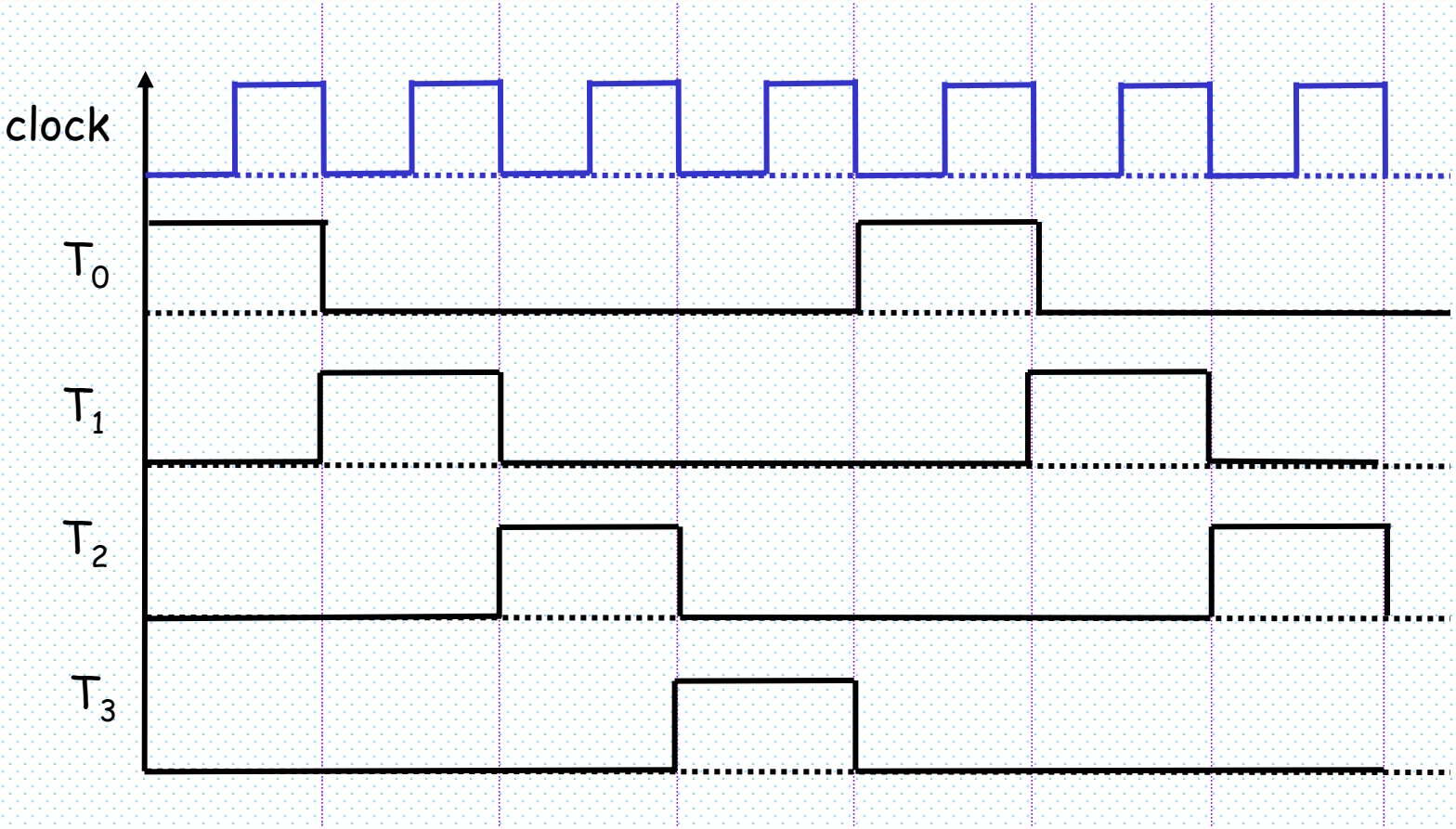
- Ring Counter
  - Timing signals control the sequence of operations in a digital system
  - A ring counter is a circular shift register with only one flip-flop being set at any particular time, all others are cleared.



initial value
1000

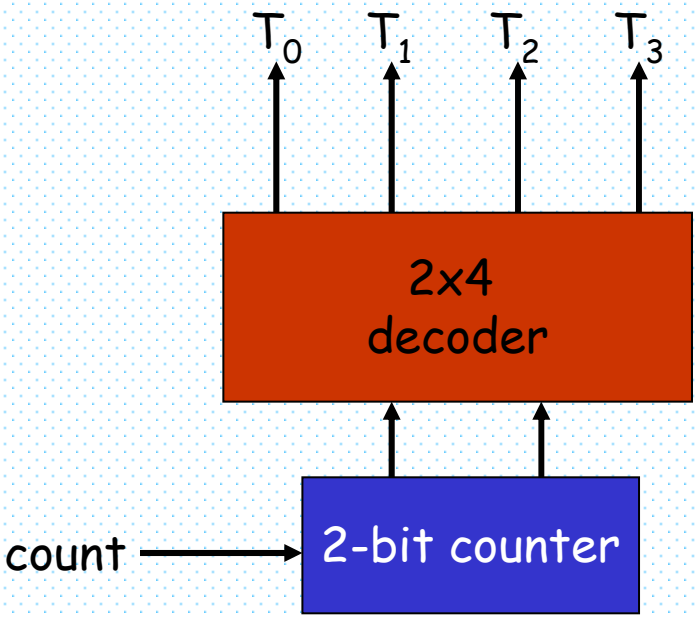
Ring Counter

- Sequence of timing signals



Ring Counter

- To generate 2^n timing signals,
  - we need a shift register with 2^n flip-flops
- or, we can construct the ring counter with a binary counter and a decoder

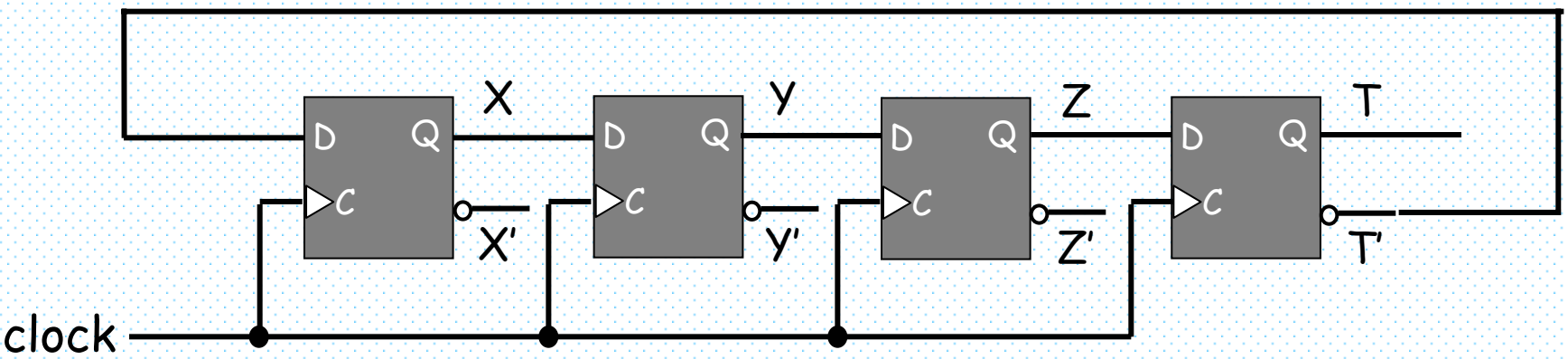


Cost:
- 2 flip-flop
- 2-to-4 line decoder

Cost in general case:
- n flip-flops
- n-to-2^n line decoder
  - 2^n n-input AND gates

Johnson Counter

- A k-bit ring counter can generate k distinguishable states
- The number of states can be doubled if the shift register is connected as a switch-tail ring counter



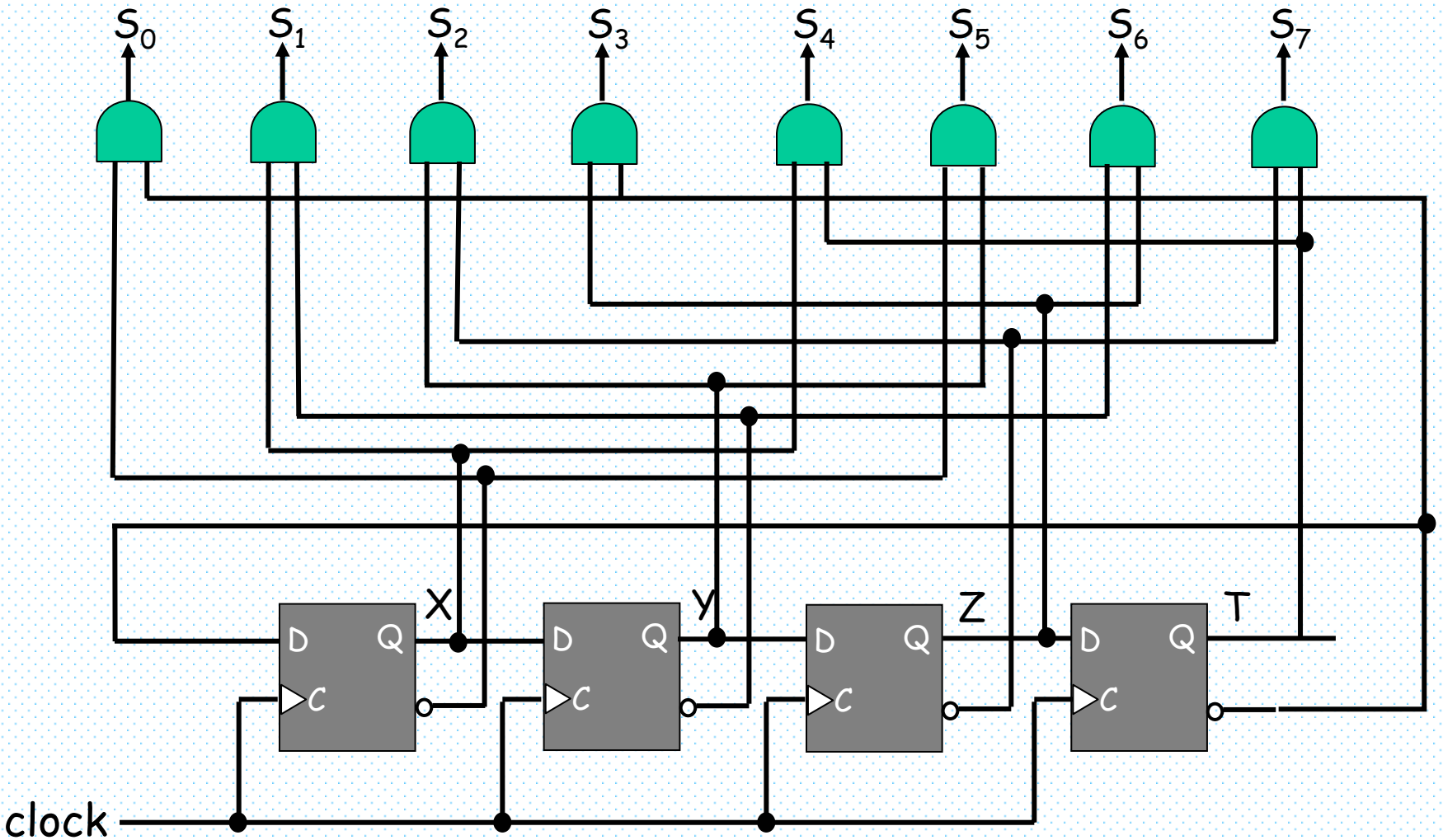
Johnson Counter

- Count sequence and required decoding

sequence number	Flip-flop outputs				Output
	X	Y	Z	T	
1	0	0	0	0	X'T'
2	1	0	0	0	XY'
3	1	1	0	0	YZ'
4	1	1	1	0	ZT'
5	1	1	1	1	XT
6	0	1	1	1	X'Y
7	0	0	1	1	Y'Z
8	0	0	0	1	Z'T

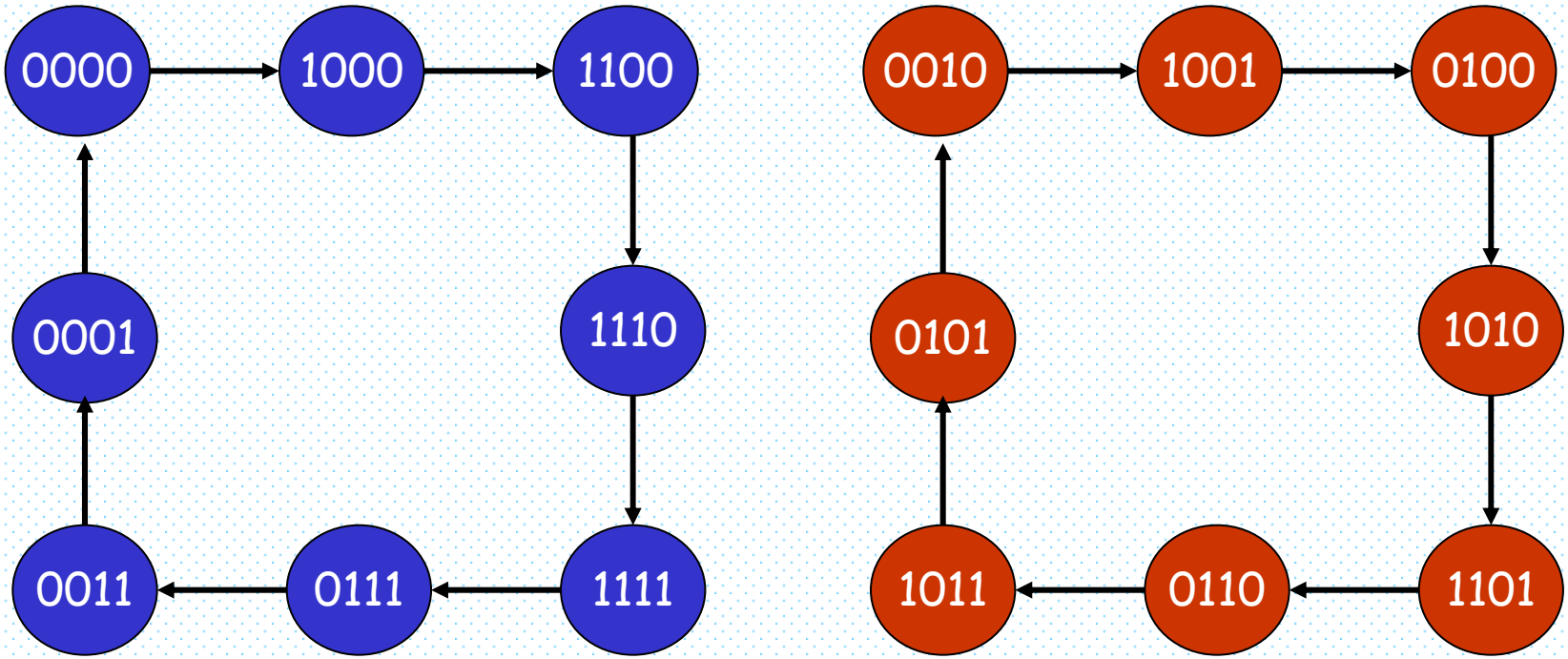
Johnson Counter

- Decoding circuit



Unused States in Counters

- 4-bit Johnson counter



Johnson Counter

Inputs				Outputs			
X	Y	Z	T	X	Y	Z	T
0	0	0	0	1	0	0	0
1	0	0	0	1	1	0	0
1	1	0	0	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1
0	1	1	1	0	0	1	1
0	0	1	1	0	0	0	1
0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	1
1	1	0	1	0	1	1	0
0	1	1	0	1	0	1	1
1	0	1	1	0	1	0	1
0	1	0	1	0	0	0	0
0	0	1	0	1	0	0	1
1	0	0	1	0	1	0	0
0	1	0	0	1	0	0	0

K-Maps

XY \ ZT	00	01	11	10
00	1			1
01	1			1
11	1			1
10	1			1

X = T'

XY \ ZT	00	01	11	10
00				
01				
11	1	1	1	1
10	1	1	1	1

Y = X

XY \ ZT	00	01	11	10
00				
01			1	1
11	1	1	1	1
10				

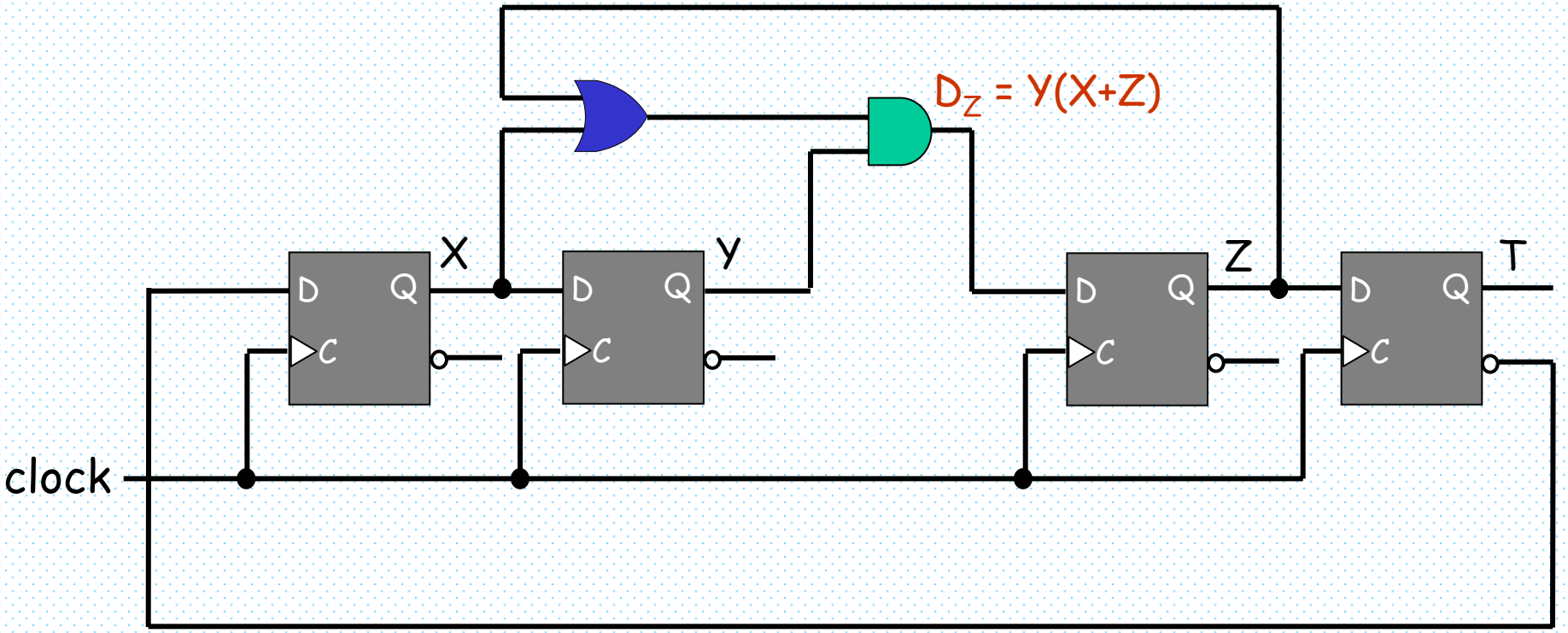
Z = XY + YZ

XY \ ZT	00	01	11	10
00			1	1
01			1	1
11			1	1
10			1	1

T = Z

Unused States in Counters

- Remedy



Unused States in Counters

- State diagram