Topic 6: Registers & Counters

-- Fahimul Haque (FHE)

* Slides are used as an aiding tool to teach in the classroom. Not every information/details that will be taught in the classes are mentioned on the slides. Hence, you are expected to follow the given textbook(s) for your course.

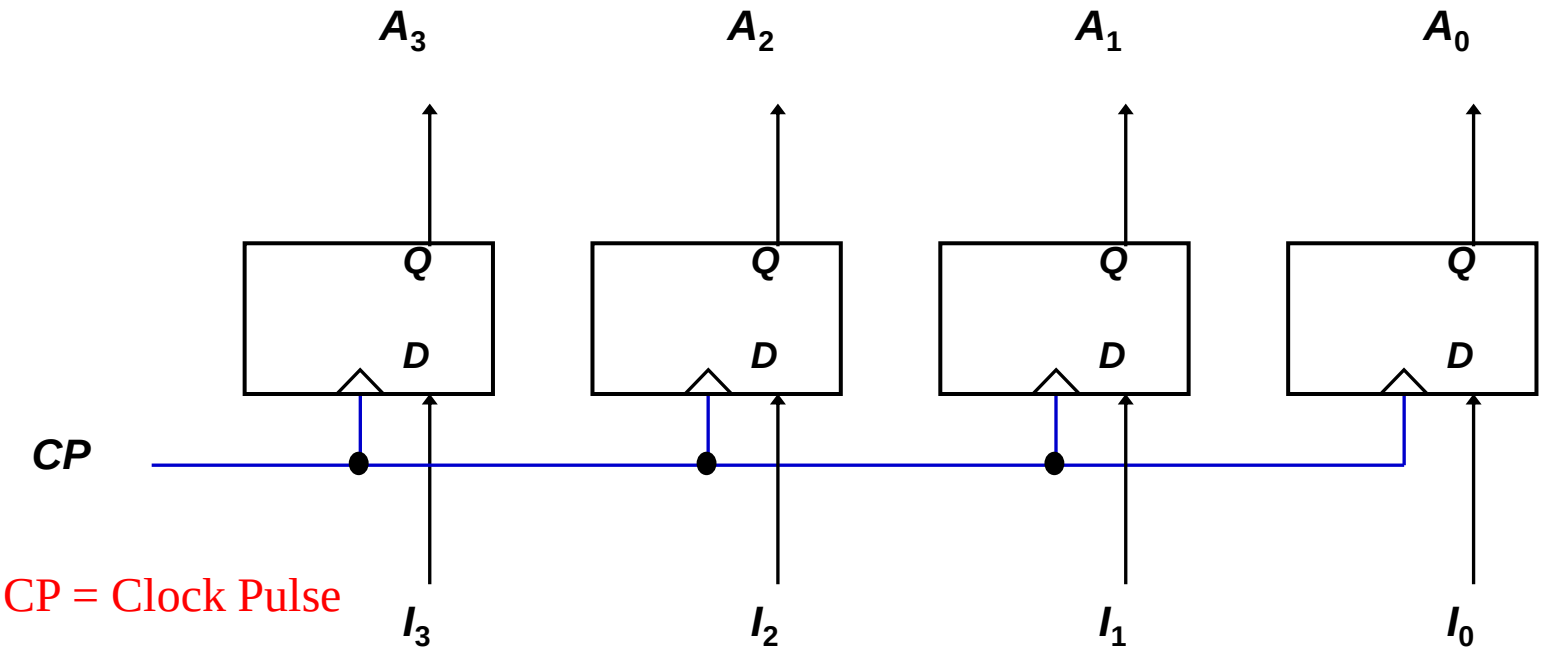
Part 1: Registers

Registers

- A *register* is a group of flip-flops, each one of which shares a common clock and is capable of storing one bit of information.
- An *n*-bit register consists of a group of *n* flip-flops capable of storing *n* bits of binary information.
- In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks.
- The flip-flops store the information while the gates control when and how new information is transferred into the register.
- Some functions of register:
  - retrieve data from register
  - store/load new data into register (serial or parallel)
  - shift the data within register (left or right)

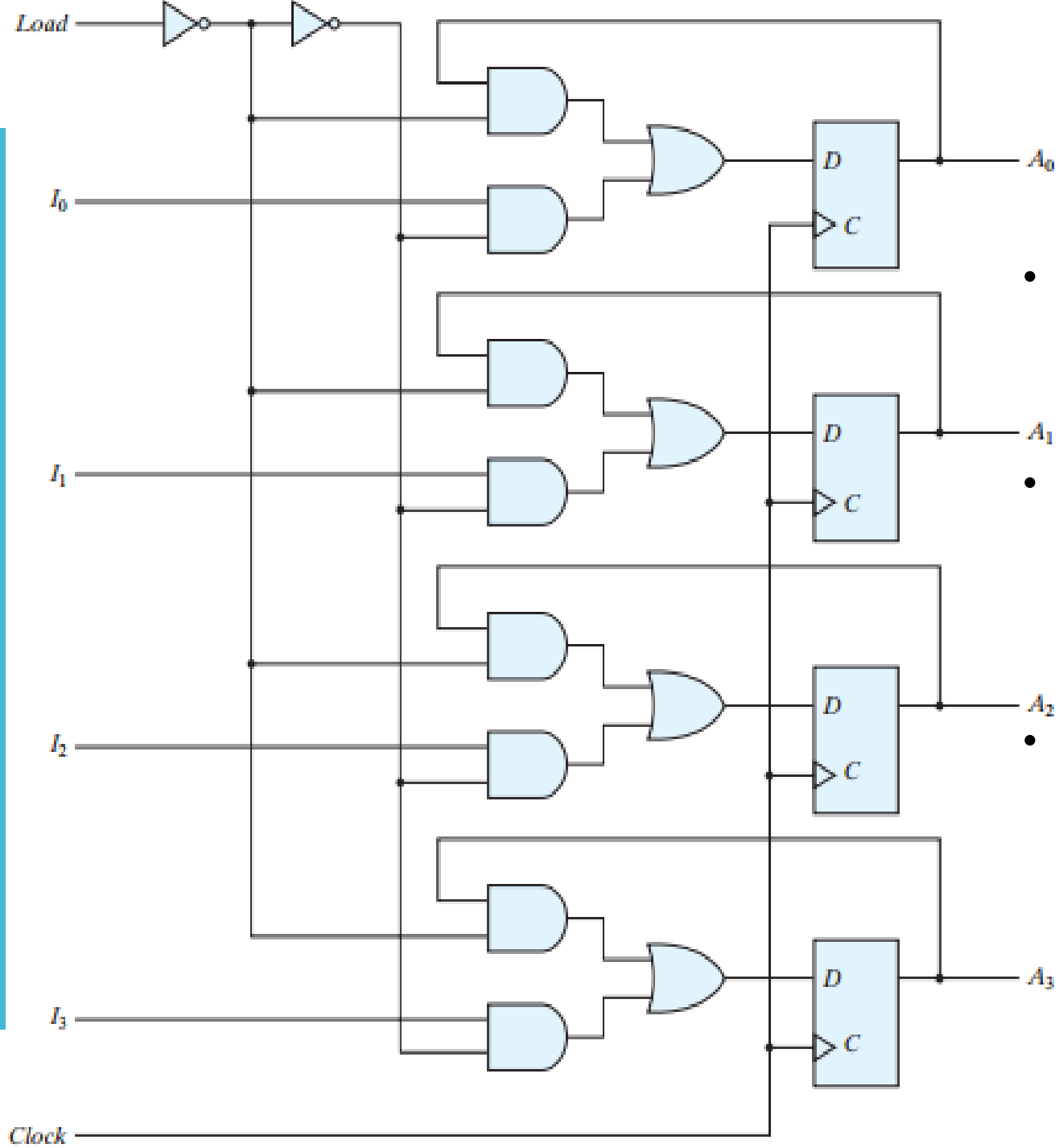
Simple Registers

- .Various types of registers are available commercially. The simplest register is one that consists of only flip-flops, without any gates.
- The transfer of new information into a register/flipflop is referred to as ***loading* or *updating*** the register/flipflop. If all the bits of the register are loaded simultaneously with a common clock pulse, we say that the loading is done ***in parallel***.
- Example: A 4-bit register. A new 4-bit data is loaded every clock cycle (the common clock input triggers all flip-flops during a positive/negative edge of a pulse).



* CP = Clock Pulse

Four-bit register with load signal

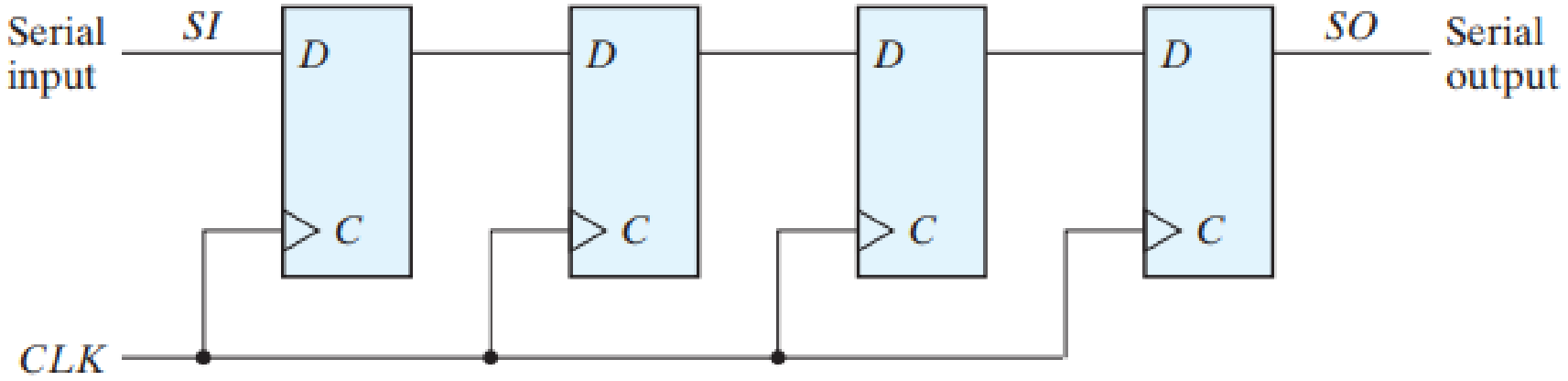


- The load input to the register determines the action to be taken with each clock pulse.
- When the load input is 1, the data at the four external inputs are transferred into the register with the next positive edge of the clock.
- When the load input is 0, the outputs of the flip-flops are connected to their respective inputs to keep the contents of the register unchanged.

Shift Registers

- Another function of a register, besides storage, is to provide for ***data movements***.
- A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction, is called a ***shift register***.
- Each *stage* (flip-flop) in a shift register represents one bit of storage, and the shifting capability of a register permits the movement of data from stage to stage within the register, or into or out of the register upon application of clock pulses.
- The logical configuration of a shift register consists of a **chain of flip-flops in cascade**, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive common clock pulses, which activate the shift of data from one stage to the next.

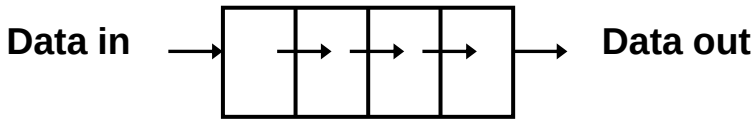
Four-bit Shift Register



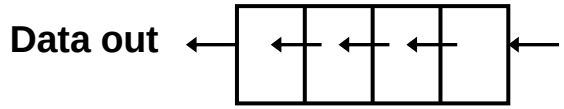
- The simplest possible shift register is one that uses only flip-flops.
- The shown shift register is **unidirectional** (left-to-right).
- The ***serial input*** determines what goes into the leftmost flip-flop during the shift.
- The ***serial output*** is taken from the output of the rightmost flip-flop.

Different Shift Register Operations

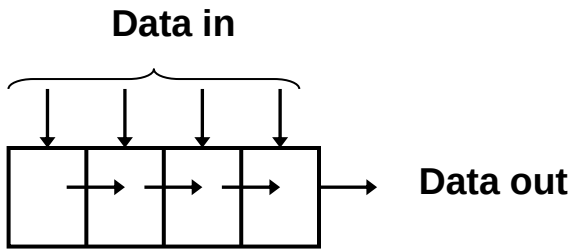
- **Basic data movement in shift registers (four bits are used for illustration).**



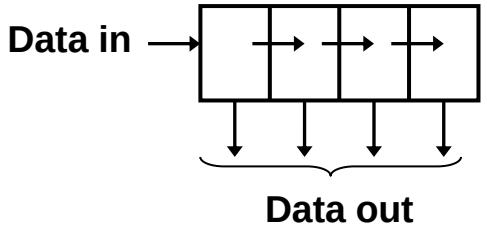
(a) Serial in/shift right/serial out



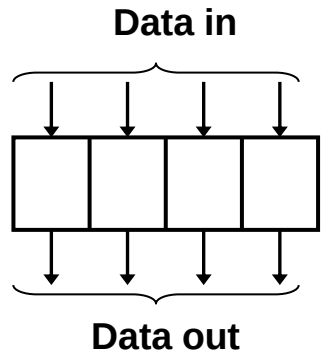
(b) Serial in/shift left/serial out



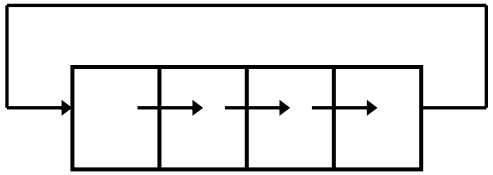
(c) Parallel in/serial out



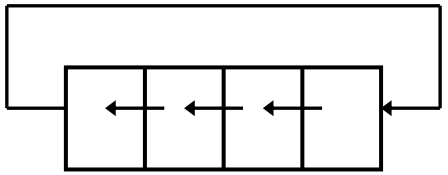
(d) Serial in/parallel out



(e) Parallel in / parallel out



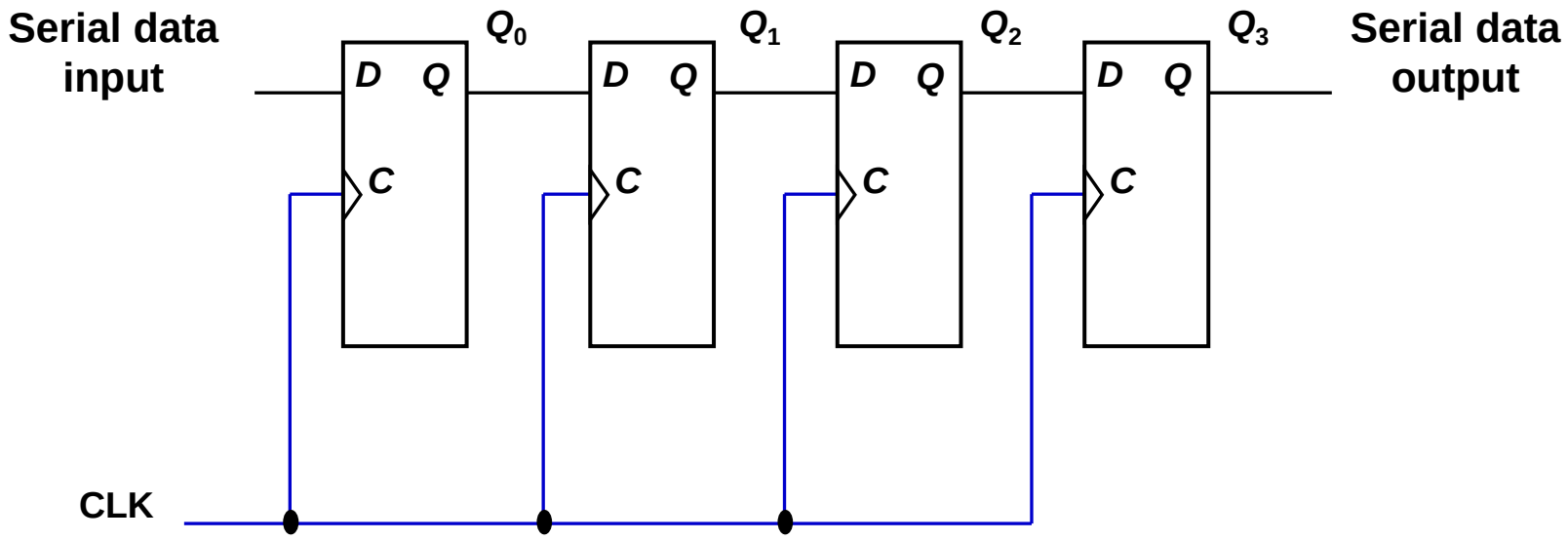
(f) Rotate right



(g) Rotate left

Serial In/Serial Out Shift Right Registers

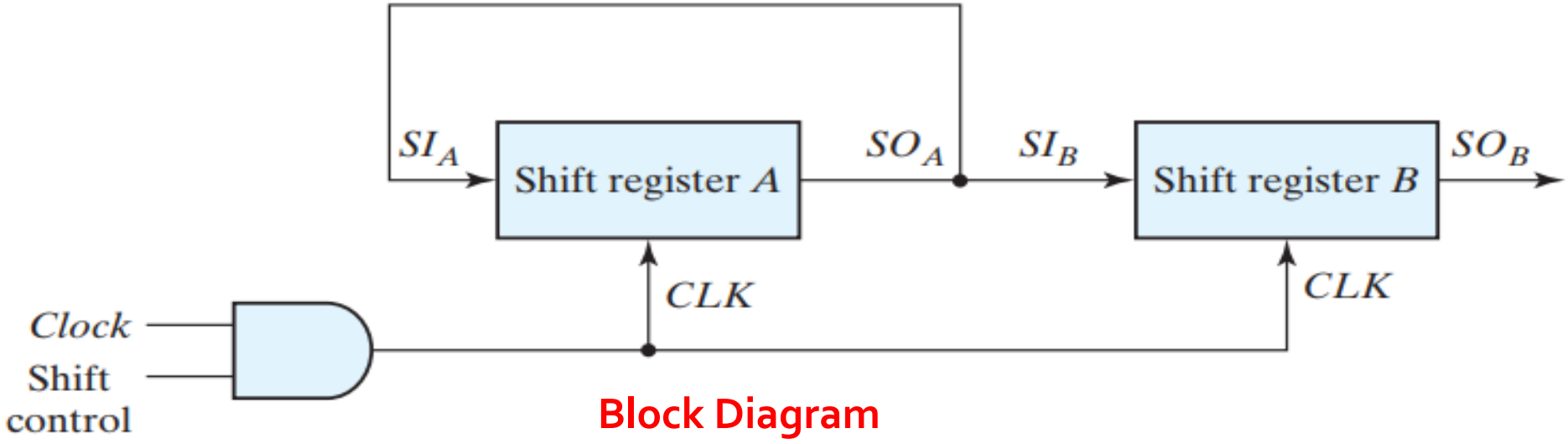
- Accepts data serially – one bit at a time – and also produces output serially.



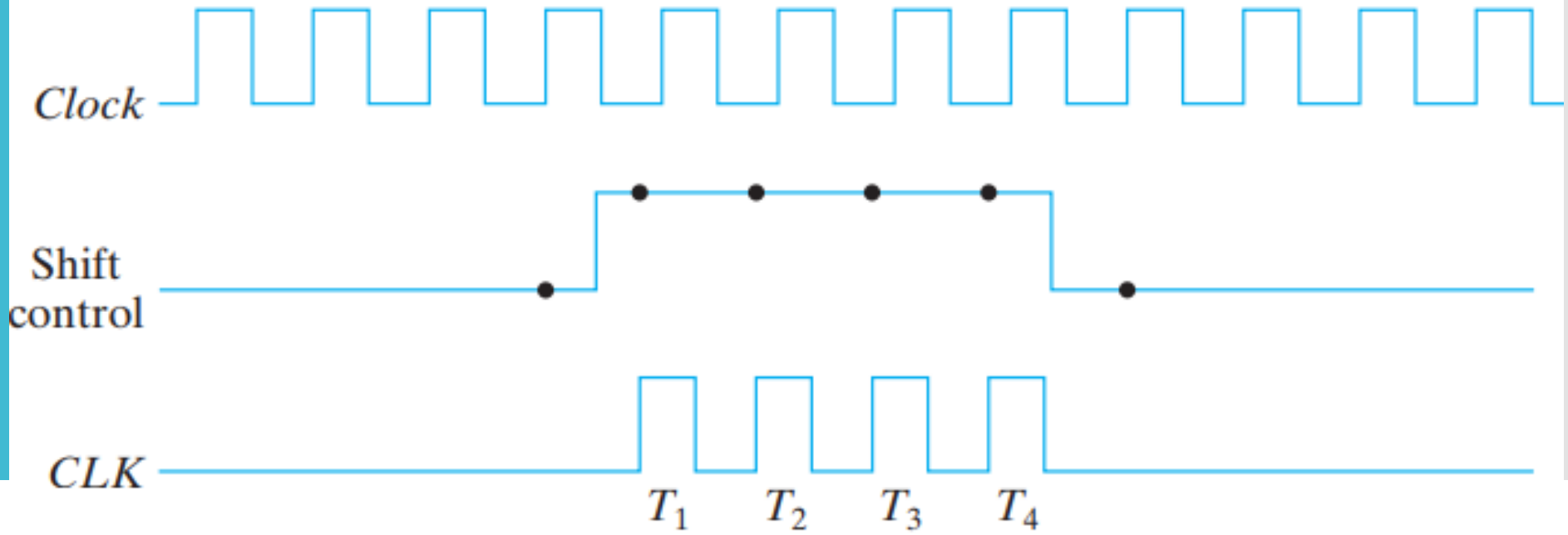
How can we make a 3-bit serial in, serial out, and left shift register?

Serial Transfer

Application: Serial transfer of data from one register to another.



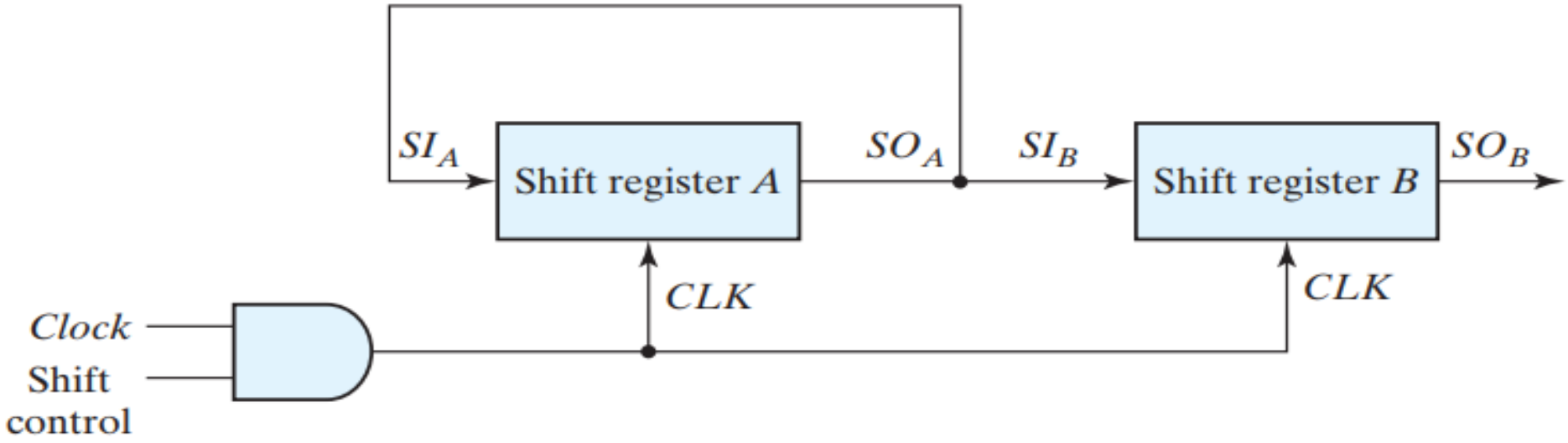
Block Diagram



Timing Diagram

Serial In/Serial Out Shift Registers

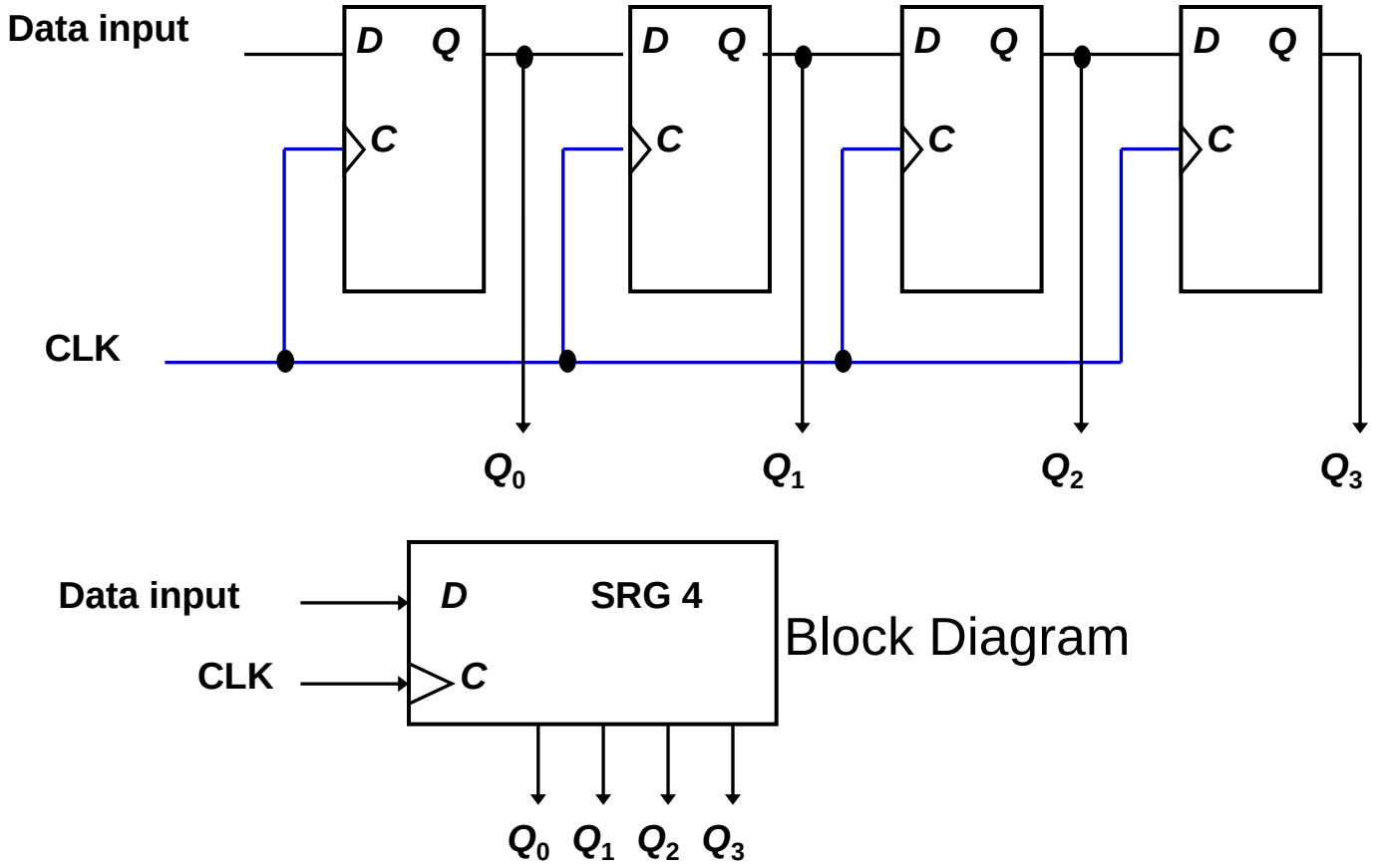
- Serial-transfer example.



Timing Pulse	Shift register *A*				Shift register *B*				Serial output of *B*
Initial value	1	0	1	1	0	0	1	0	0
After T_1	1	1	0	1	1	0	0	1	1
After T_2	1	1	1	0	1	1	0	0	0
After T_3	0	1	1	1	0	1	1	0	0
After T_4	1	0	1	1	1	0	1	1	1

Serial In/Parallel Out Shift Registers

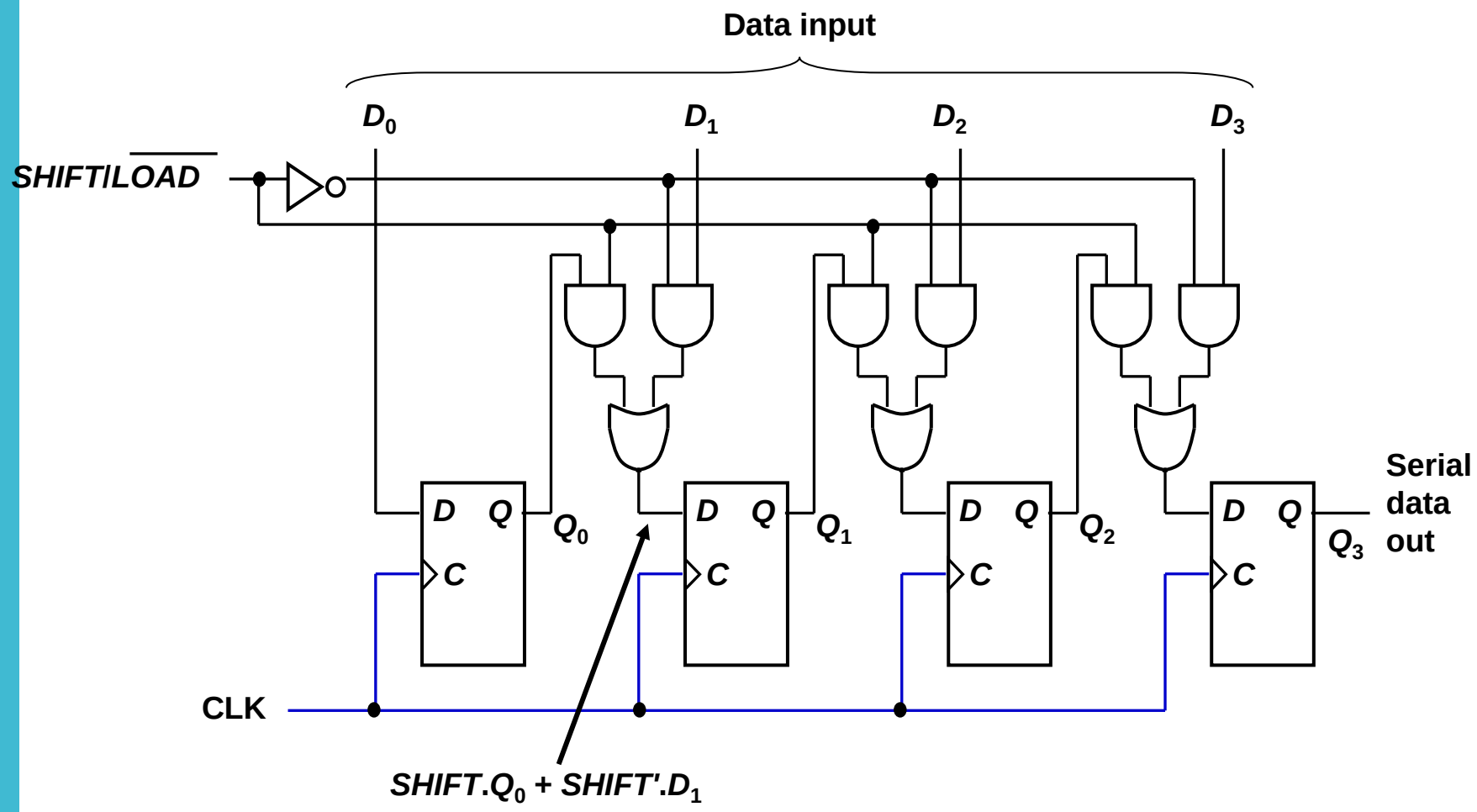
- Accepts data serially.
- Outputs of all stages are available simultaneously.



Block Diagram

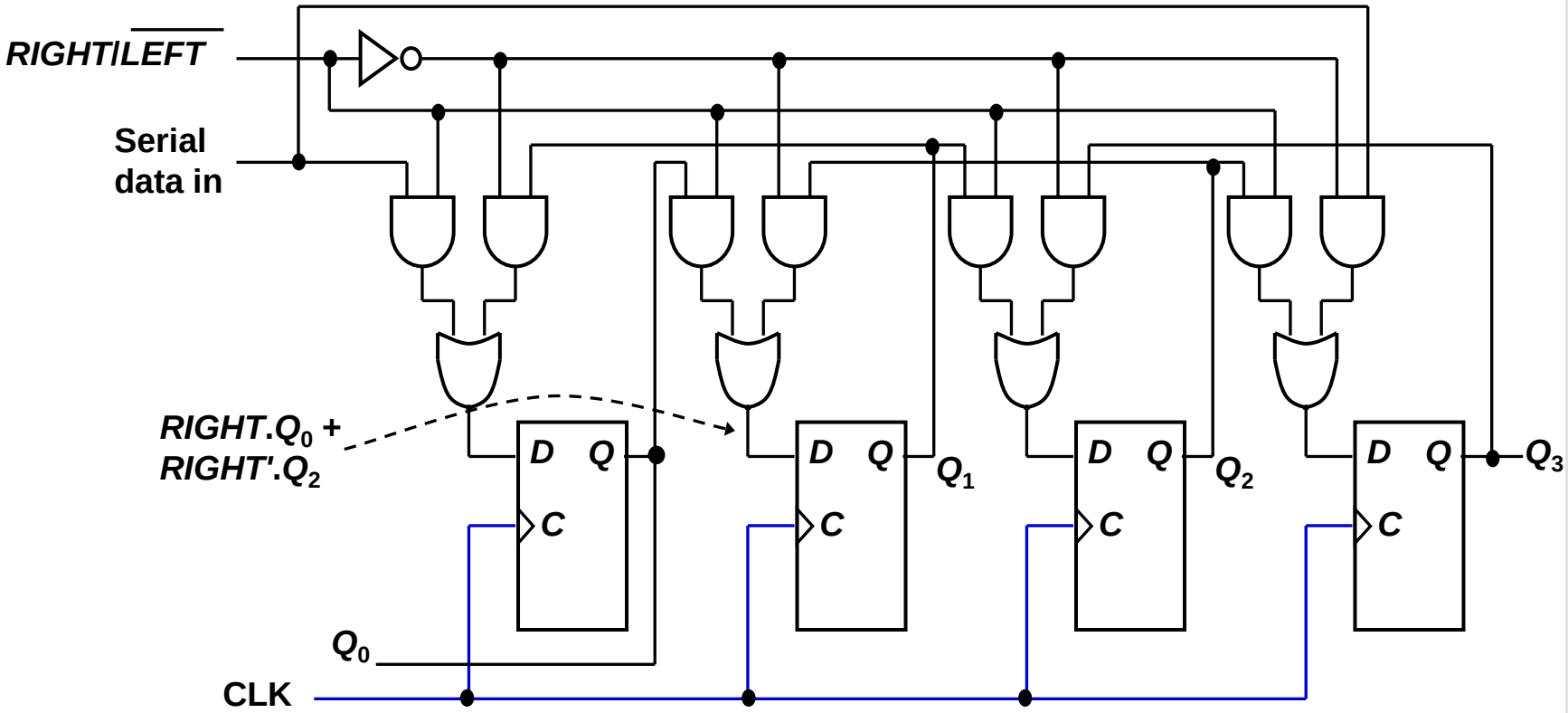
Parallel In/Serial Out Shift Registers

- Bits are entered simultaneously, but output is serial.



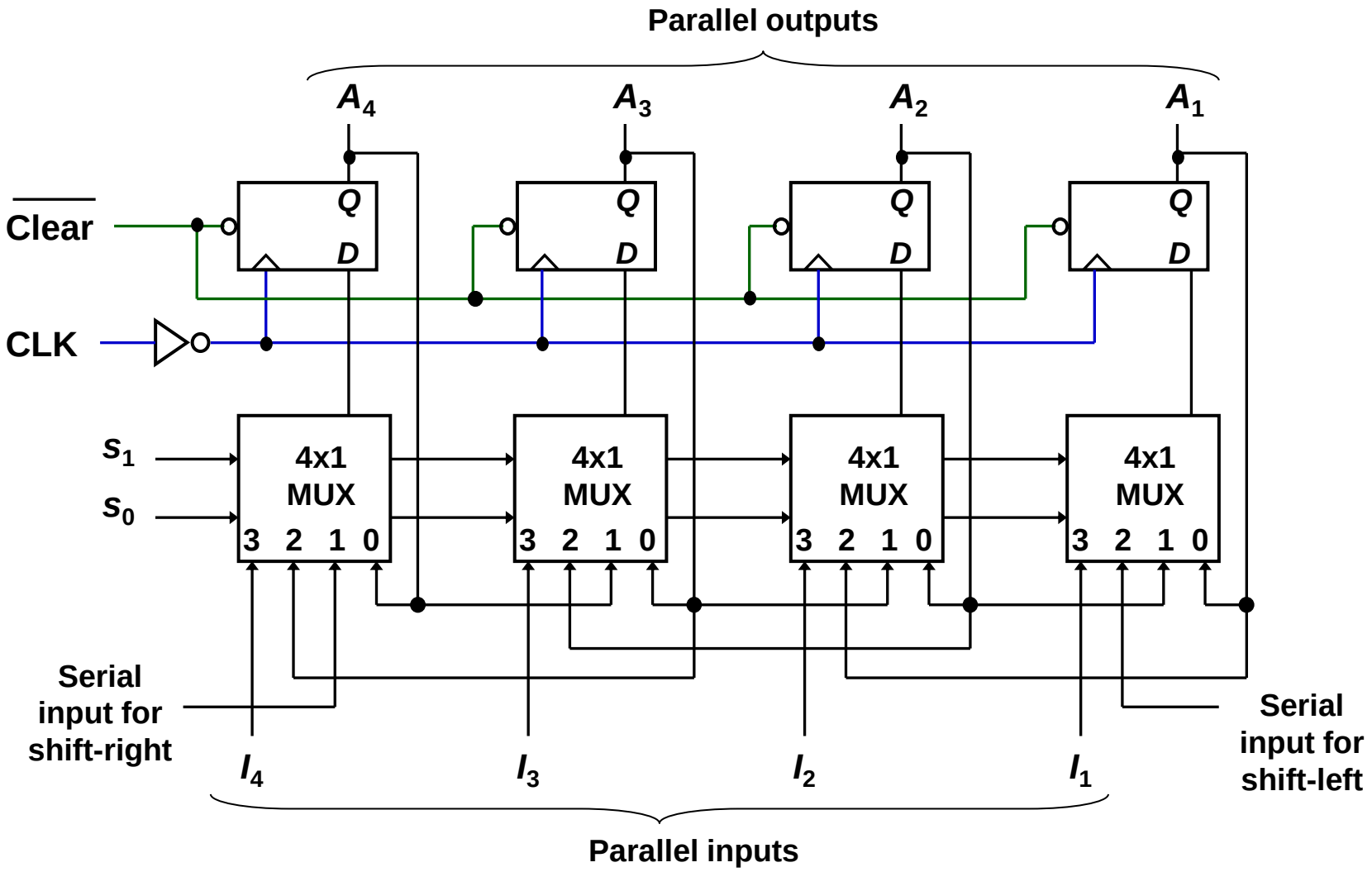
Bidirectional Shift Registers

- Data can be shifted either left or right, using a control line *RIGHT/LEFT* (or simply *RIGHT*) to indicate the direction.



Bidirectional Shift Registers

- 4-bit bidirectional shift register with parallel load.



Bidirectional Shift Registers

- 4-bit bidirectional shift register with parallel load.

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

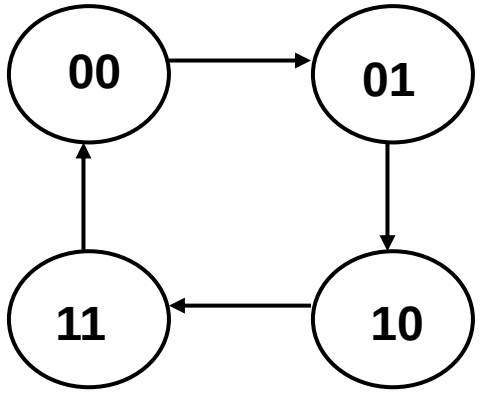
Part 2:Counters

Counters

- Counters are circuits that cycle through a specified number of states.
- Two types of counters:
  - synchronous (parallel) counters
  - asynchronous (ripple) counters
- Ripple counters allow some flip-flop outputs to be used as a source of clock for other flip-flops.
- Synchronous counters apply the same clock to all flip-flops.

Synchronous (Parallel) Counters

- Synchronous (parallel) counters: the flip-flops are clocked at the same time by a common clock pulse.
- We can design these counters using the sequential logic design process.
- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).



Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

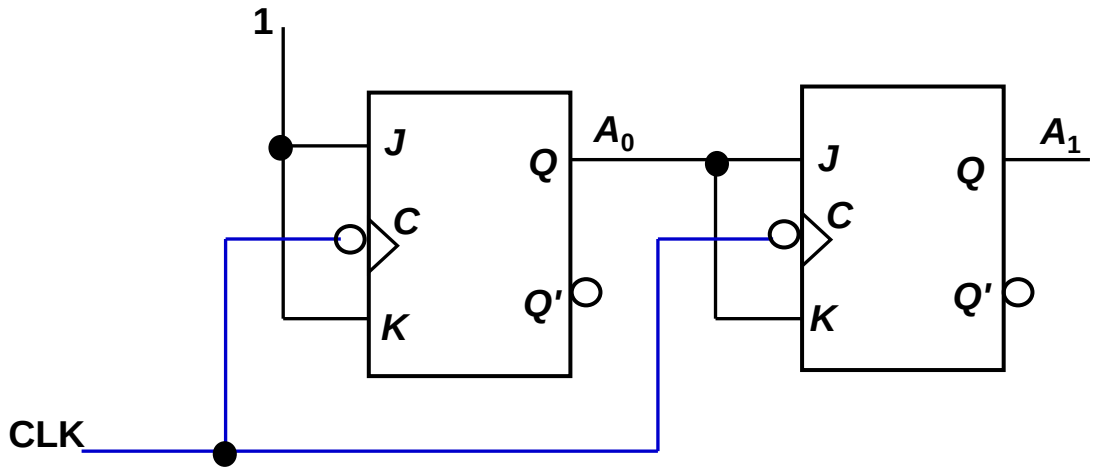
Synchronous (Parallel) Counters

- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).

Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

$TA_1 = A_0$

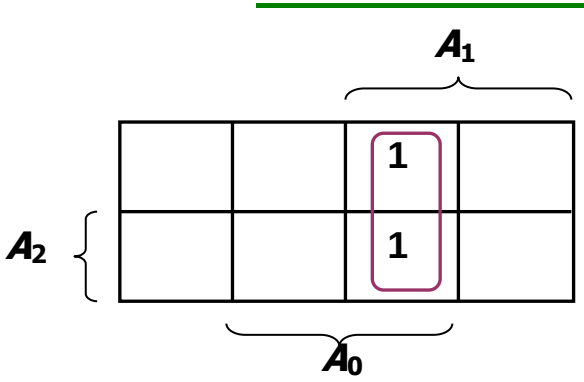
$TA_0 = 1$



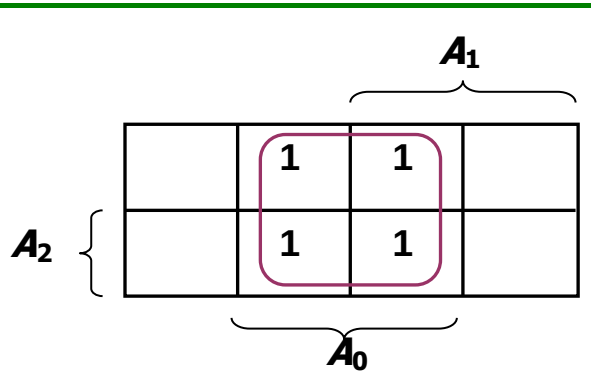
Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J, K inputs).

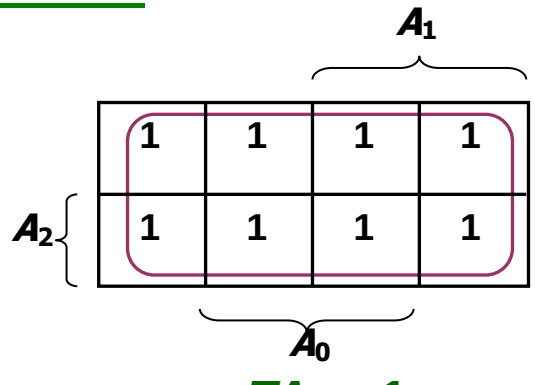
Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2^+	A_1^+	A_0^+	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



$TA_2 = A_1.A_0$



$TA_1 = A_0$

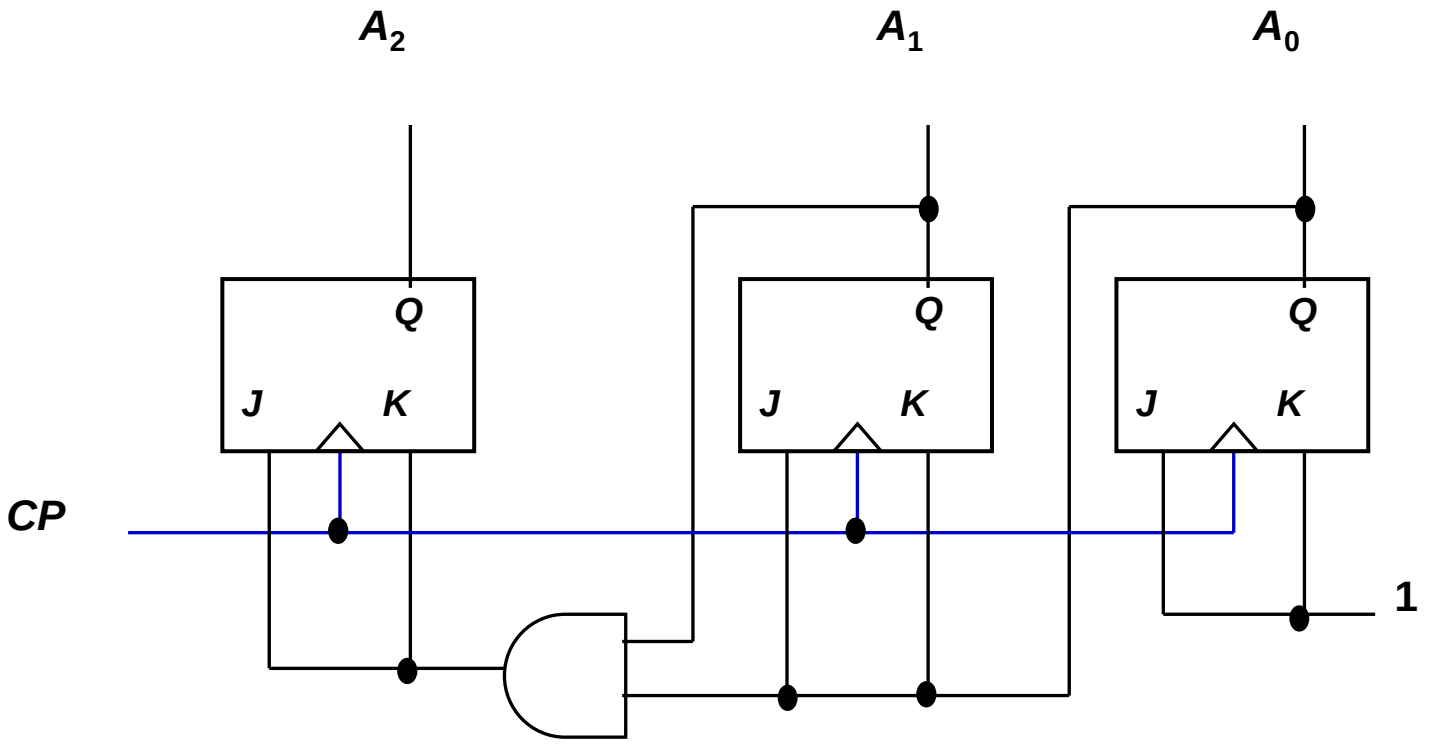


$TA_0 = 1$

Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (continued).

$TA_2 = A_1.A_0$ $TA_1 = A_0$ $TA_0 = 1$



Synchronous (Parallel) Counters

- Note that in a binary counter, the n^{th} bit (shown underlined) is always complemented whenever

  $\underline{0}11\ldots11 \rightarrow \underline{1}00\ldots00$

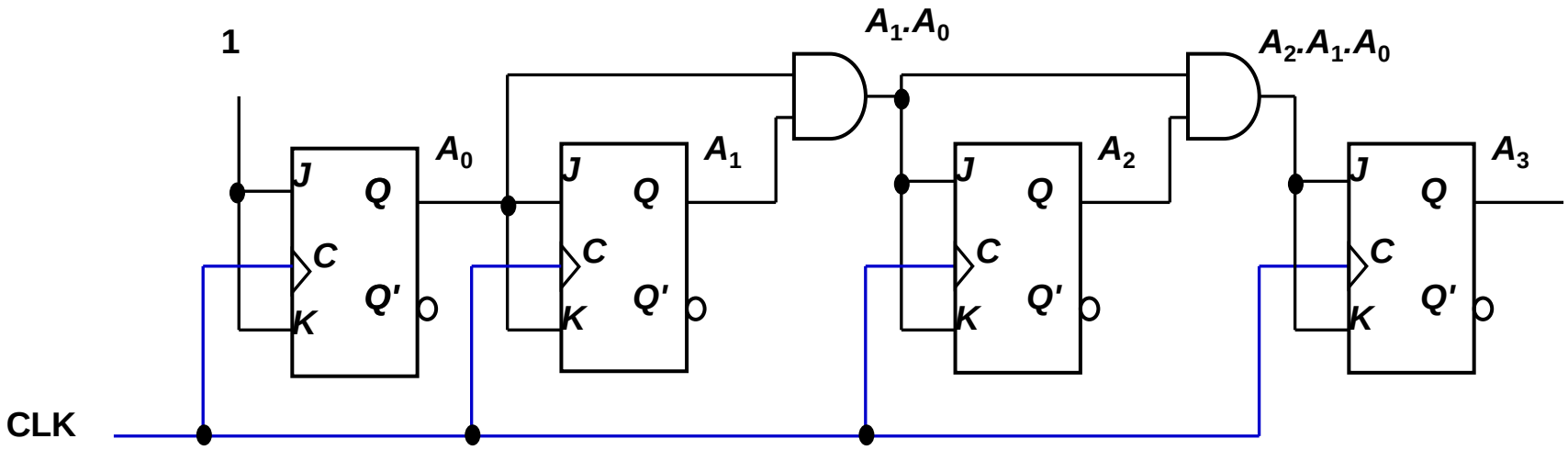
  or $\underline{1}11\ldots11 \rightarrow \underline{0}00\ldots00$

- Hence, X_n is complemented whenever
  $X_{n-1}X_{n-2} \ldots X_1X_0 = 11\ldots11.$
- As a result, if T flip-flops are used, then
  $TX_n = X_{n-1} \cdot X_{n-2} \cdot \ldots \cdot X_1 \cdot X_0$

Synchronous (Parallel) Counters

- Example: 4-bit synchronous binary counter.

$$TA_3 = A_2 \,.\, A_1 \,.\, A_0$$
$$TA_2 = A_1 \,.\, A_0$$
$$TA_1 = A_0$$
$$TA_0 = 1$$



Synchronous (Parallel) Counters

- Example: Synchronous decade/BCD counter.

Clock pulse	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 (recycle)	0	0	0	0

$T_0 = 1$

$T_1 = Q_3'.Q_0$

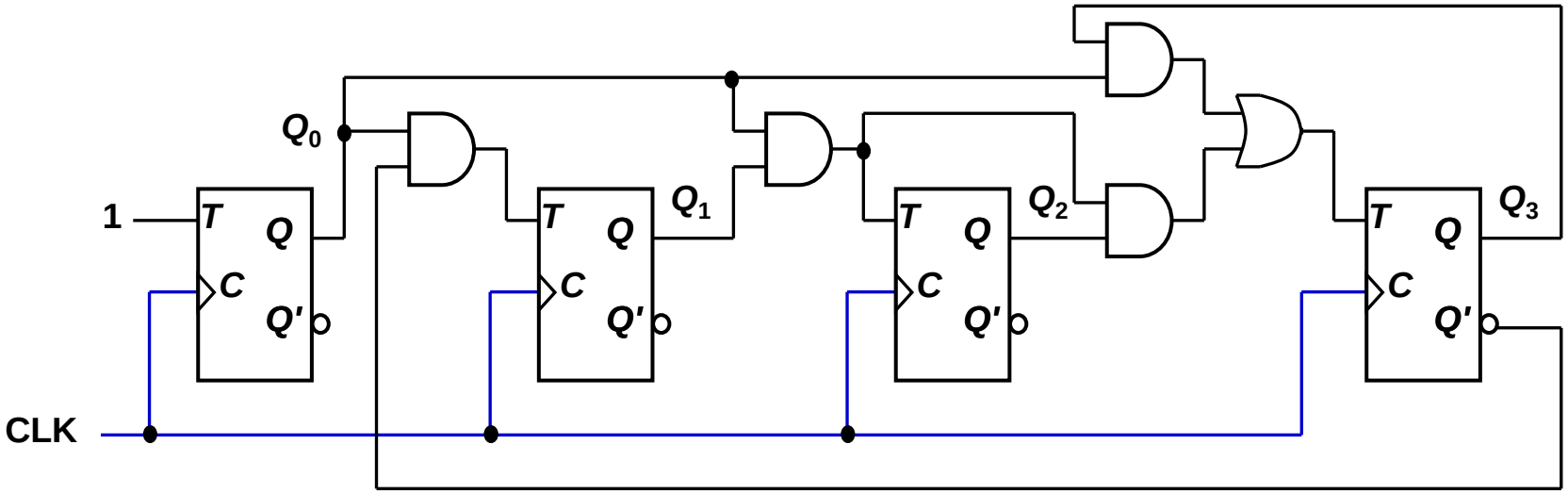
$T_2 = Q_1.Q_0$

$T_3 = Q_2.Q_1.Q_0 + Q_3.Q_0$

Synchronous (Parallel) Counters

- Example: Synchronous decade/BCD counter (continued).

$$T_0 = 1$$
$$T_1 = Q_3'.Q_0$$
$$T_2 = Q_1.Q_0$$
$$T_3 = Q_2.Q_1.Q_0 + Q_3.Q_0$$



Up/Down Synchronous Counters

- Up/down synchronous counter: a *bidirectional* counter that is capable of counting either up or down.
- An input (control) line *Up*/$\overline{Down}$ (or simply *Up*) specifies the direction of counting.
  - *Up*/$\overline{Down}$ = 1 $\rightarrow$ Count upward
  - *Up*/$\overline{Down}$ = 0 $\rightarrow$ Count downward

Up/Down Synchronous Counters

- Example: A 3-bit up/down synchronous binary counter.

Clock pulse	*Up*	Q_2	Q_1	Q_0	*Down*
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	

$TQ_0 = 1$

$TQ_1 = (Q_0.Up) + (Q_0'.Up')$

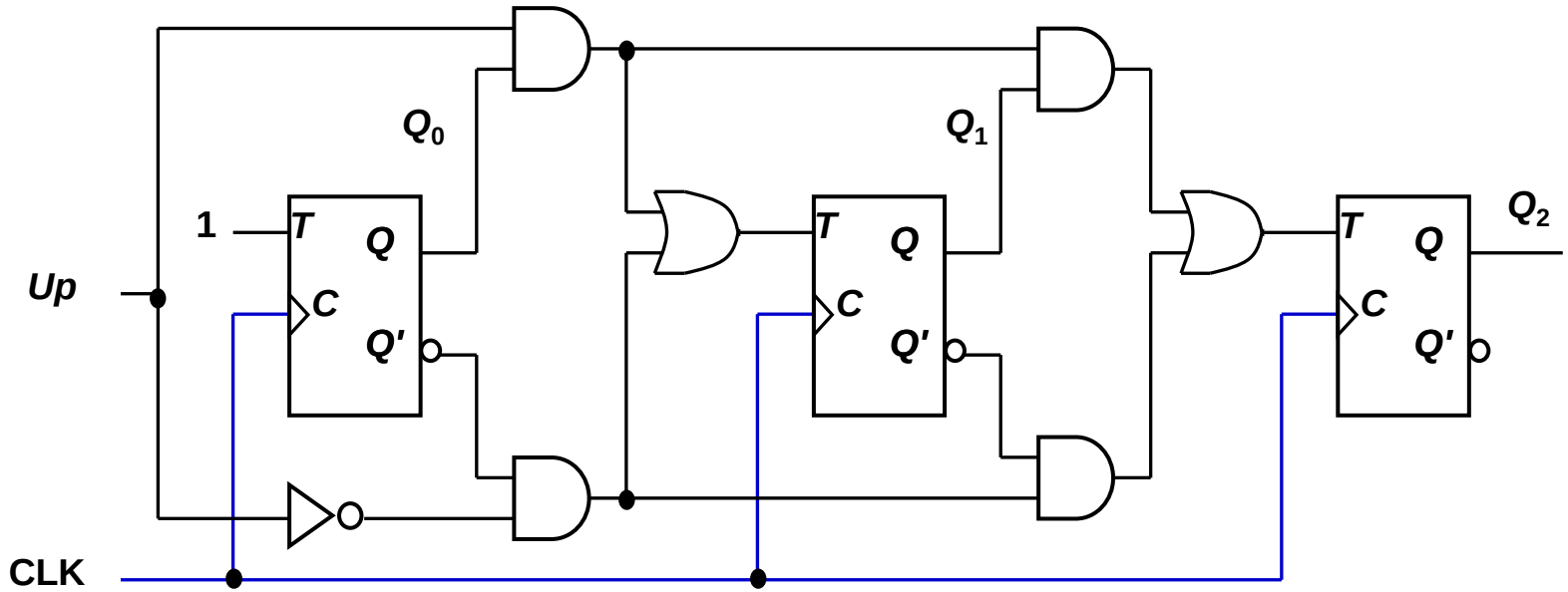
$TQ_2 = (Q_0.Q_1.Up) + (Q_0'.Q_1'.Up')$

Up counter	Down counter
$TQ_0 = 1$	$TQ_0 = 1$
$TQ_1 = Q_0$	$TQ_1 = Q_0'$
$TQ_2 = Q_0.Q_1$	$TQ_2 = Q_0'.Q_1'$

Up/Down Synchronous Counters

- Example: A 3-bit up/down synchronous binary counter (continued).

$$TQ_0 = 1$$
$$TQ_1 = (Q_0.Up) + (Q_0'.Up')$$
$$TQ_2 = (Q_0.Q_1.Up) + (Q_0'.Q_1'.Up')$$

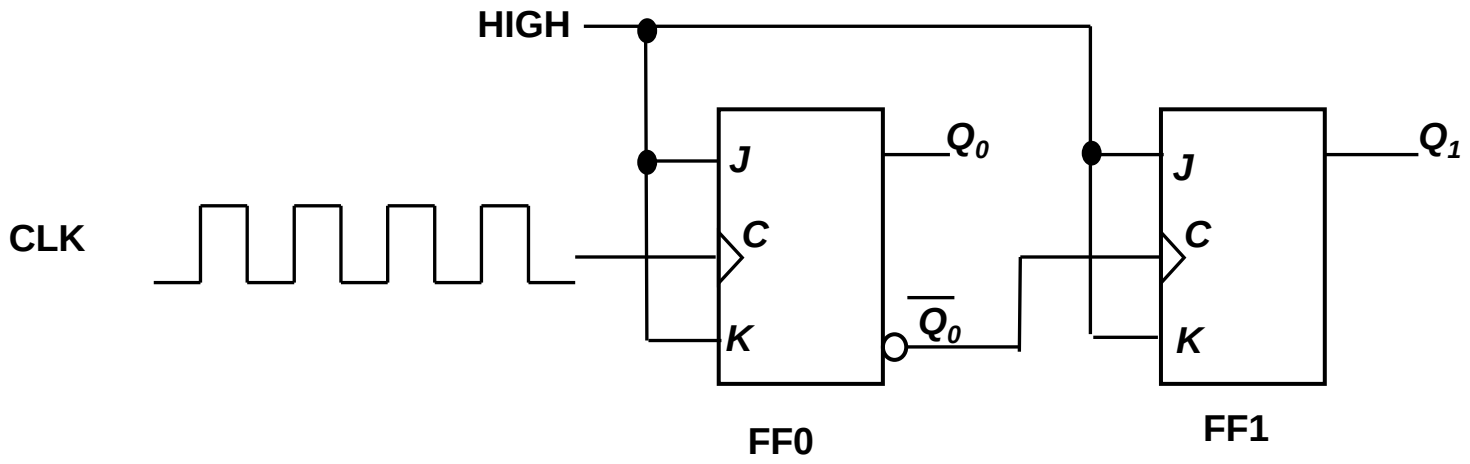


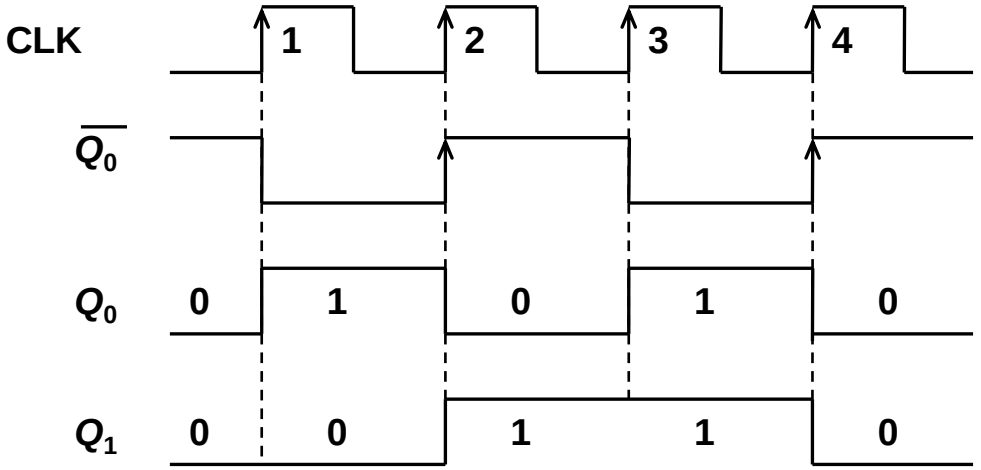
Asynchronous (Ripple) Counters

- Asynchronous counters: the flip-flops do not change states at exactly the same time as they do not have a common clock pulse.
- Also known as ripple counters, as the input clock pulse "ripples" through the counter – cumulative delay is a drawback.
- This counter is also a *frequency divider*.

Asynchronous (Ripple) Counters

- Example: 2-bit ripple binary counter.
- Output of one flip-flop is connected to the clock input of the next more-significant flip-flop.



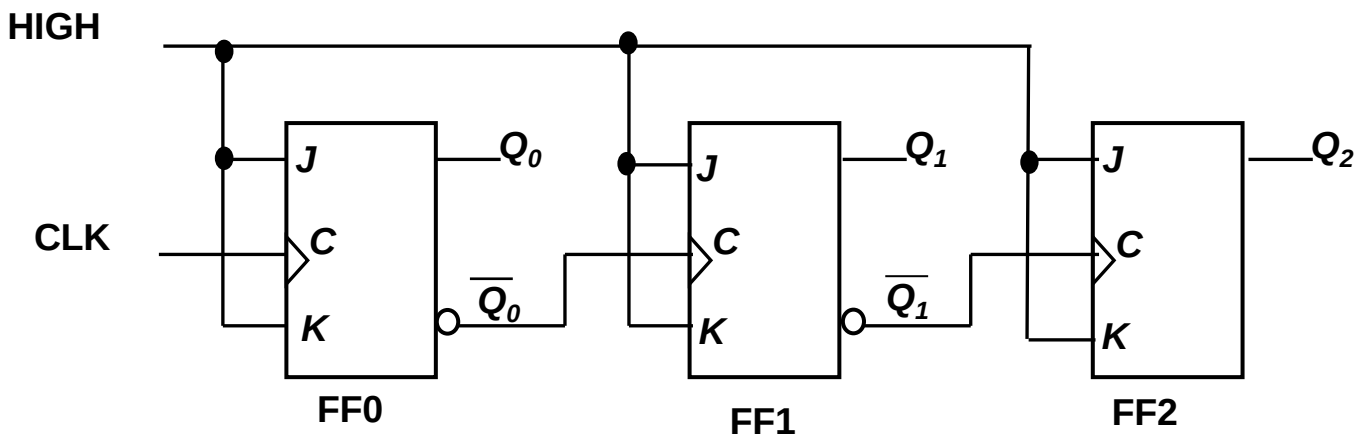


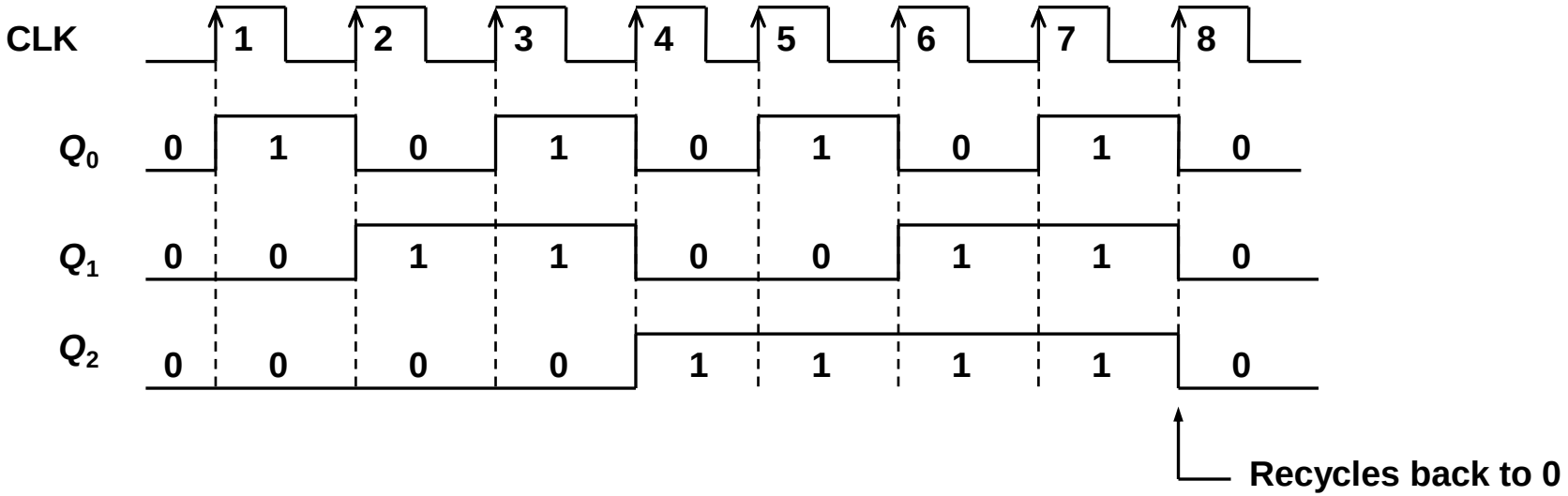
Timing diagram

$00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00$ ...

Asynchronous (Ripple) Counters

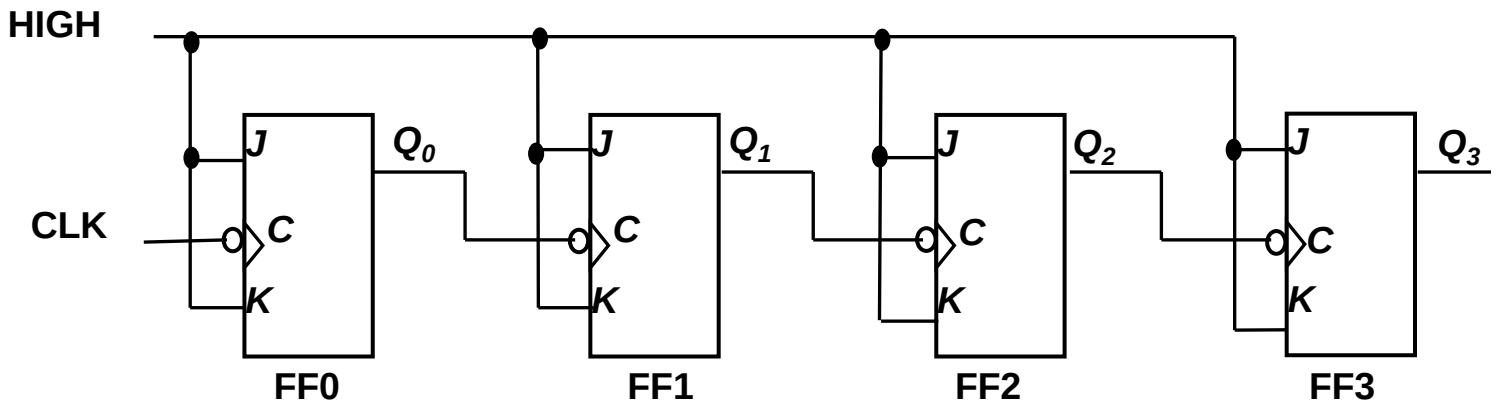
- Example: 3-bit ripple binary counter (positive-edge triggered).

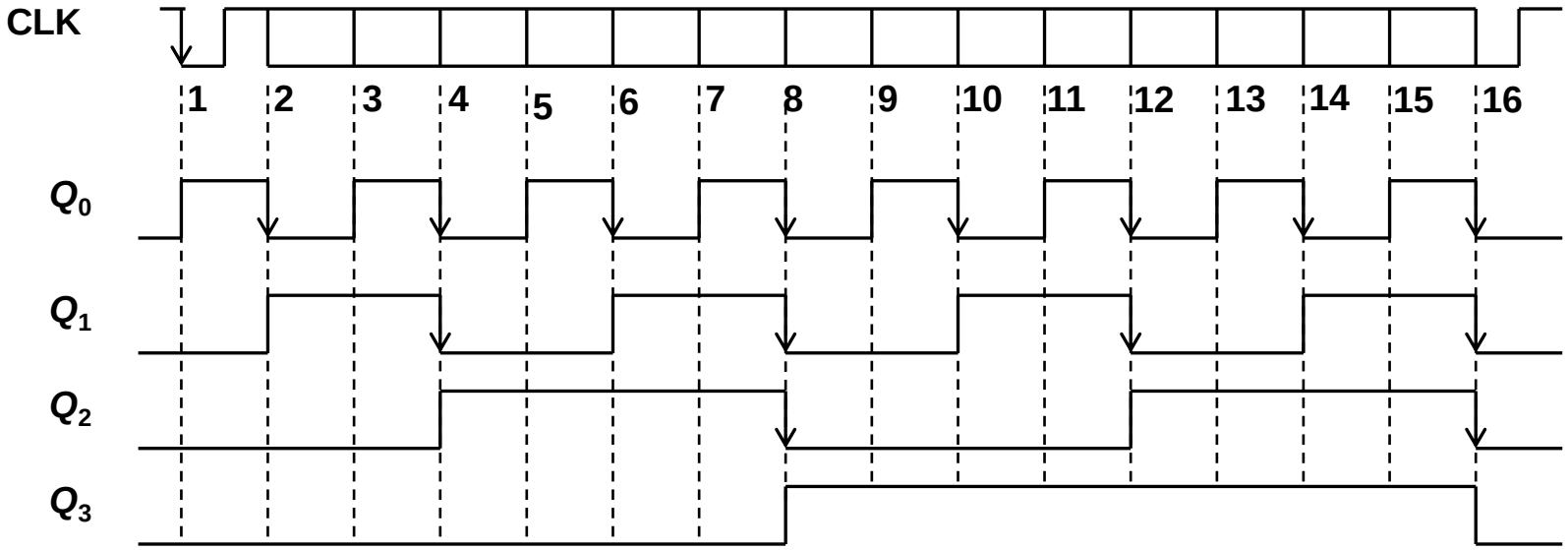




Asynchronous (Ripple) Counters

- Example: 4-bit ripple binary counter (negative-edge triggered).

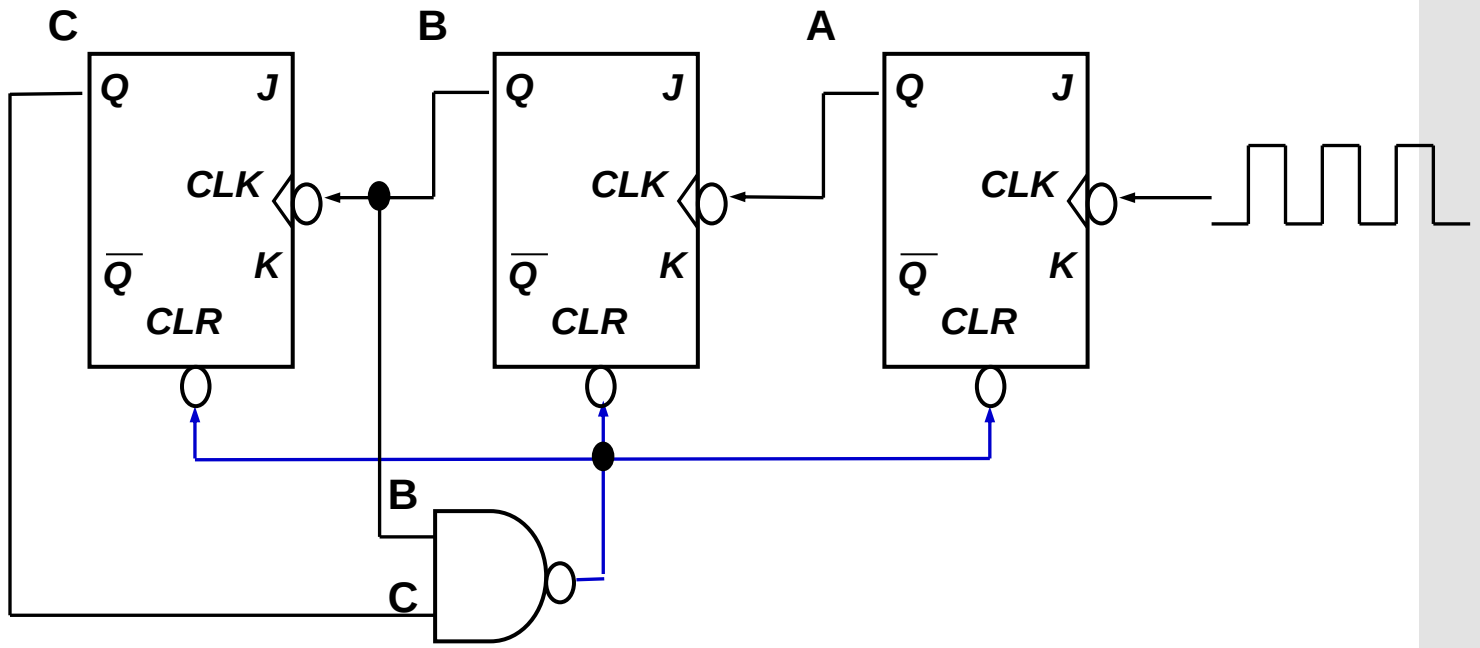




Asyn. Counters with MOD (modulus) no. < 2^n

- States may be skipped resulting in a truncated sequence.
- Technique: force counter to *recycle before going through all of the states* in the binary sequence.
- Example: Given the following circuit, determine the counting sequence (and hence the modulus no.)

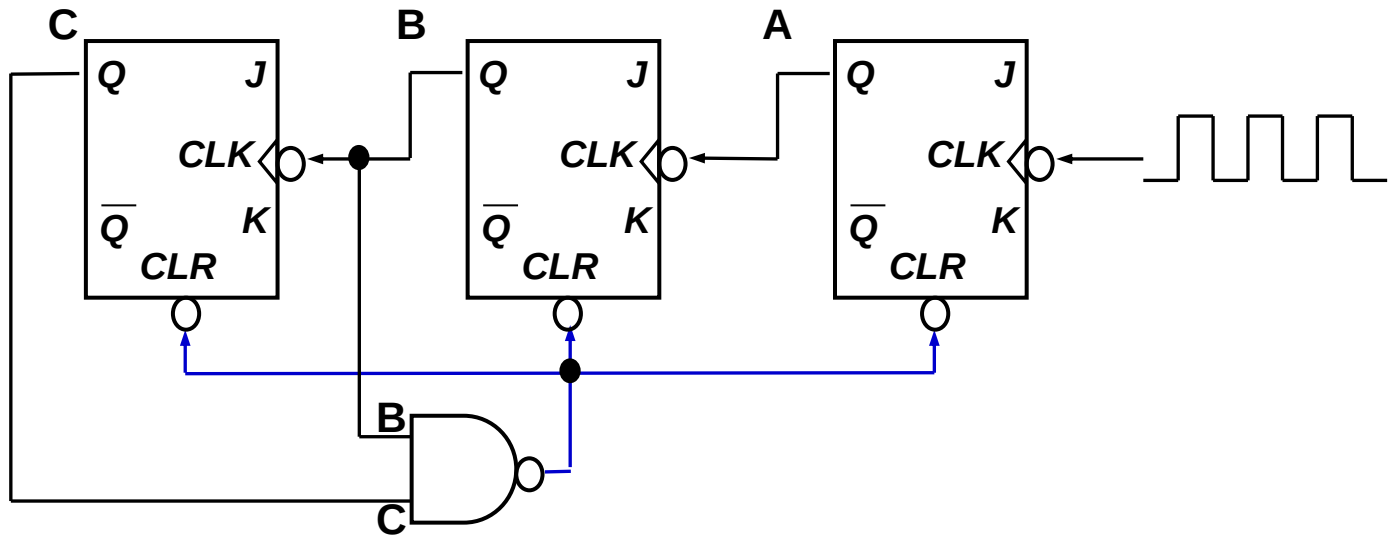
All *J*, *K* inputs are 1 (HIGH)

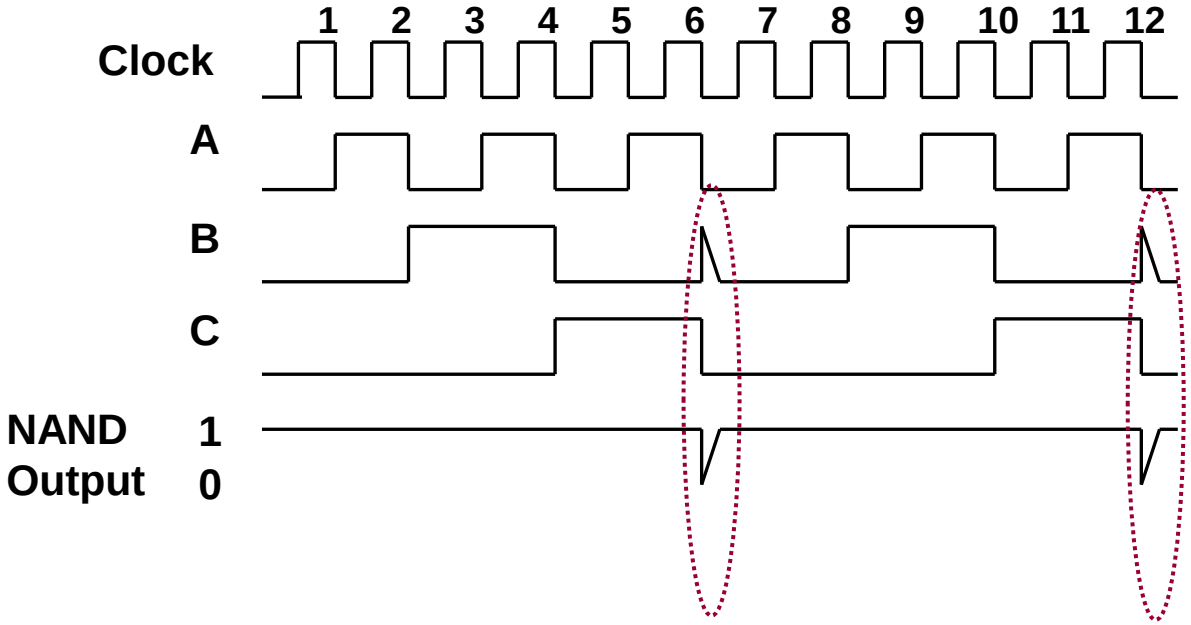


Asyn. Counters with MOD no. $< 2^n$

- Example (continued):

All *J*, *K* inputs are 1 (HIGH).

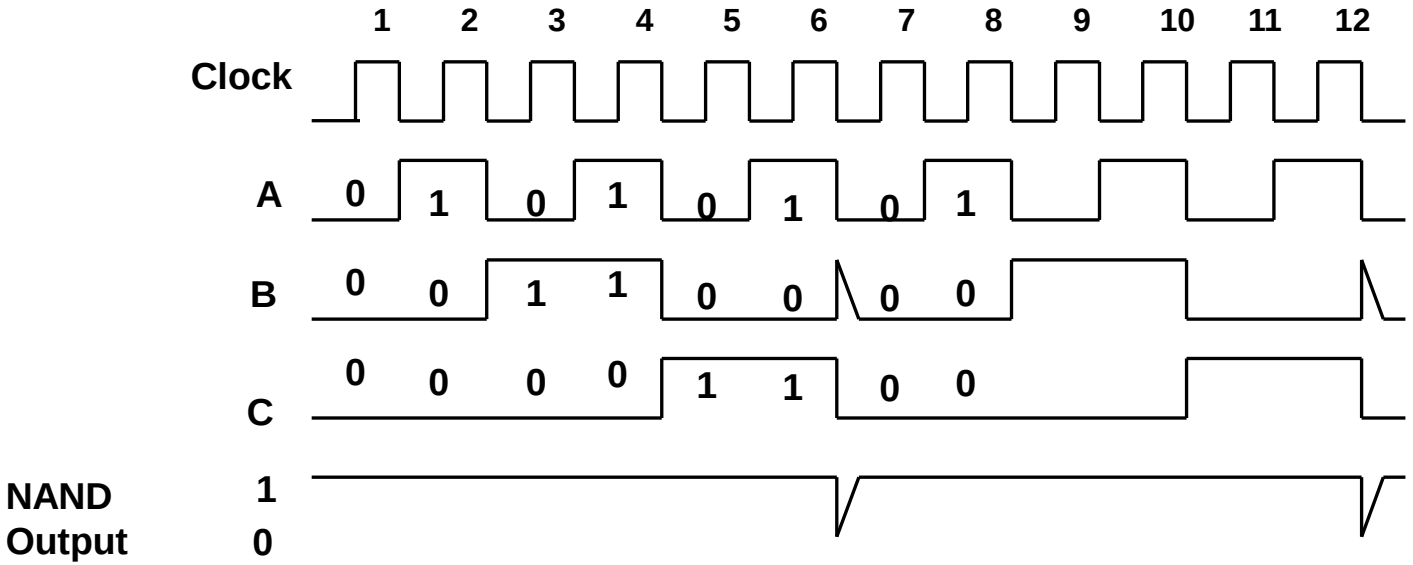


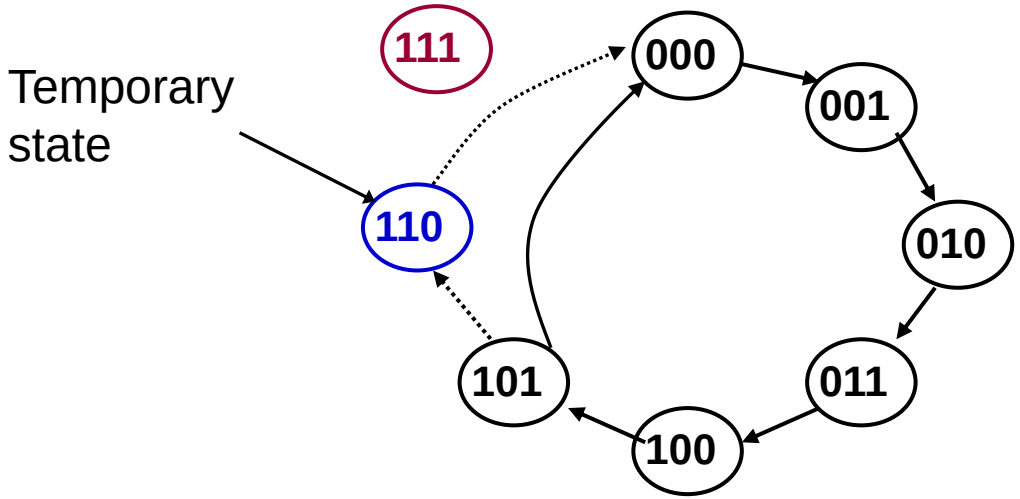


MOD-6 counter produced by clearing (a MOD-8 binary counter) when count of six (110) occurs.

- Example (continued): Counting sequence of circuit (in CBA order).

Asyn. Counters with MOD no. $< 2^n$





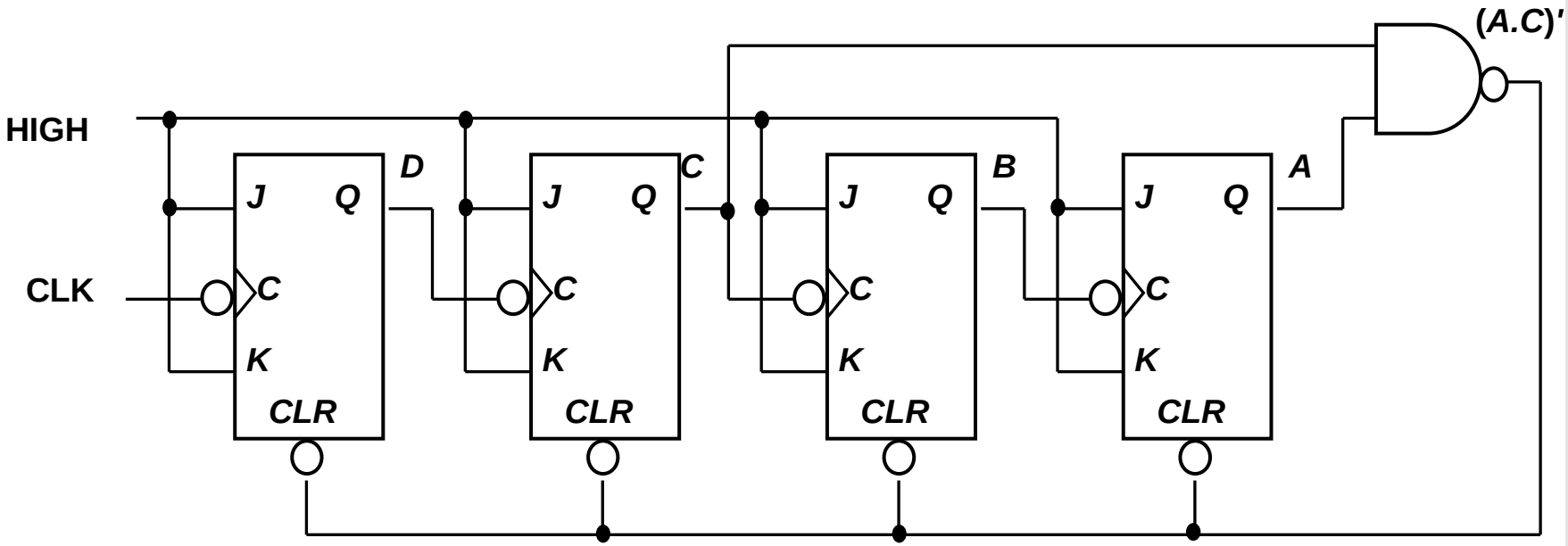
Counter is a MOD-6 counter.

Asyn. Counters with MOD no. $< 2^n$

- *Exercise:* How to construct an asynchronous MOD-5 counter? MOD-7 counter? MOD-12 counter?

Asyn. Counters with MOD no. $< 2^n$

- Decade counters (or BCD counters) are counters with 10 states (modulus-10) in their sequence. They are commonly used in daily life (e.g.: utility meters, odometers, etc.).
- Design an asynchronous decade counter.



Asyn. Counters with MOD no. $< 2^n$

- Asynchronous decade/BCD counter (continued).