Digital Logic Circuit (SE273 – Fall 2020)

Lecture 9: Registers and Counters

Jaesok Yu, Ph.D. (jaesok.yu@dgist.ac.kr)

Assistant Professor

Department of Robotics Engineering, DGIST

Goal of this lecture

- Learn different types of clocked sequential circuits
  - Registers
  - Counters

- Learn practical implementations of registers and counters
  - Design procedure for registers or counters
  - Design different types of counters in Verilog

Registers

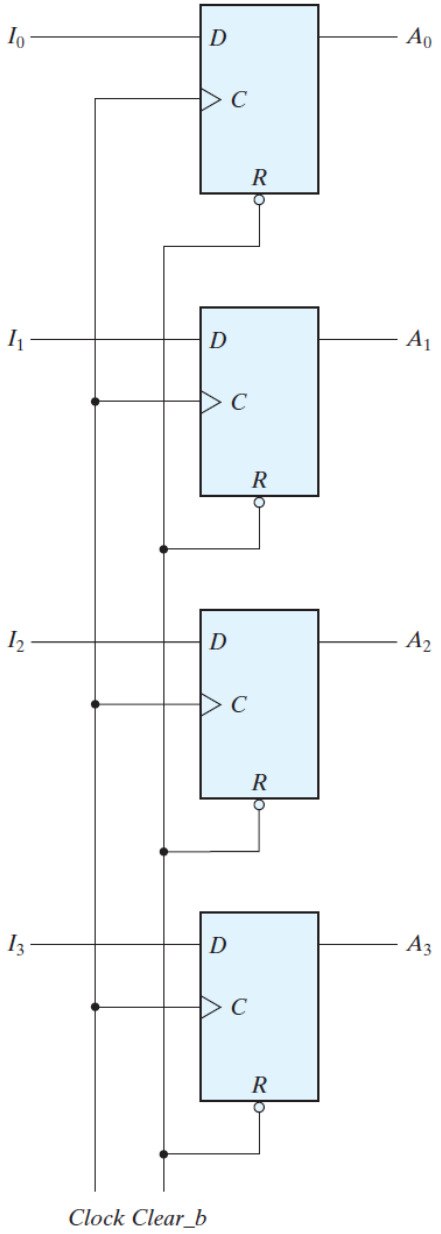
Clocked Sequential Circuit

- It consists of a group of FFs and combinational gates
  - A circuit w/ FFs is a sequential circuit even w/o combinational gates

- Circuits with FFs are usually classified by the function
  - **Register**: a group of FFs that shares a common clock
  - A n-bit register consists of a group of n FFs capable of storing n bits of binary information
  - A register may have combinational gates that perform certain data-processing

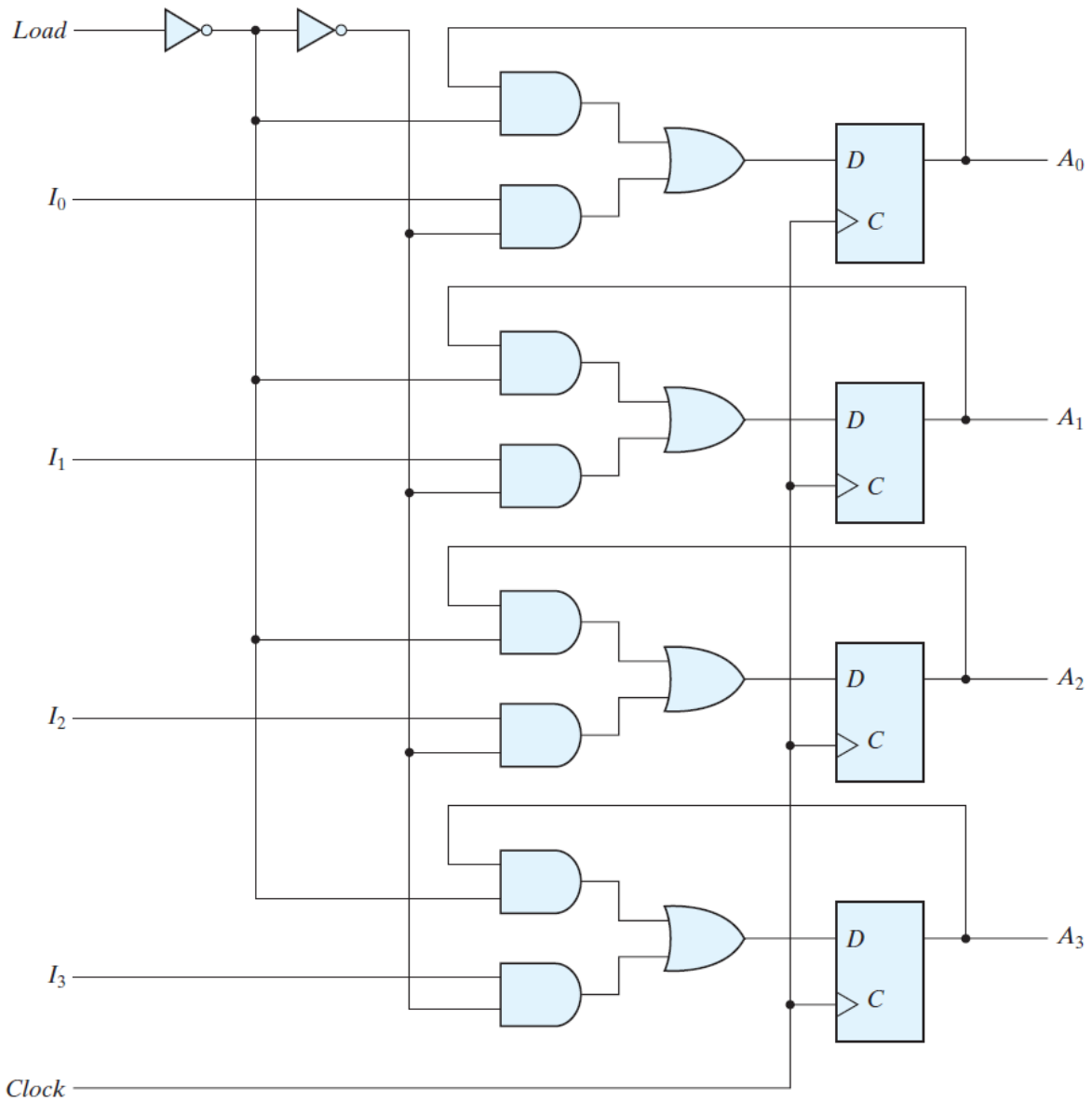
Simplest Register

- Consider a four-bit data storage register
  - It consists of only flip-flops
  - Clock triggers all FFs on **positive edge**
  - (I_3, I_2, I_1, I_0) immediately before the clock edge determines (A_3, A_2, A_1, A_0) after the clock edge
  - *Clear_b*: active-low R (reset), asynchronous reset
  - It must be at logic 1 during normal clocked operation



Register with Parallel Load

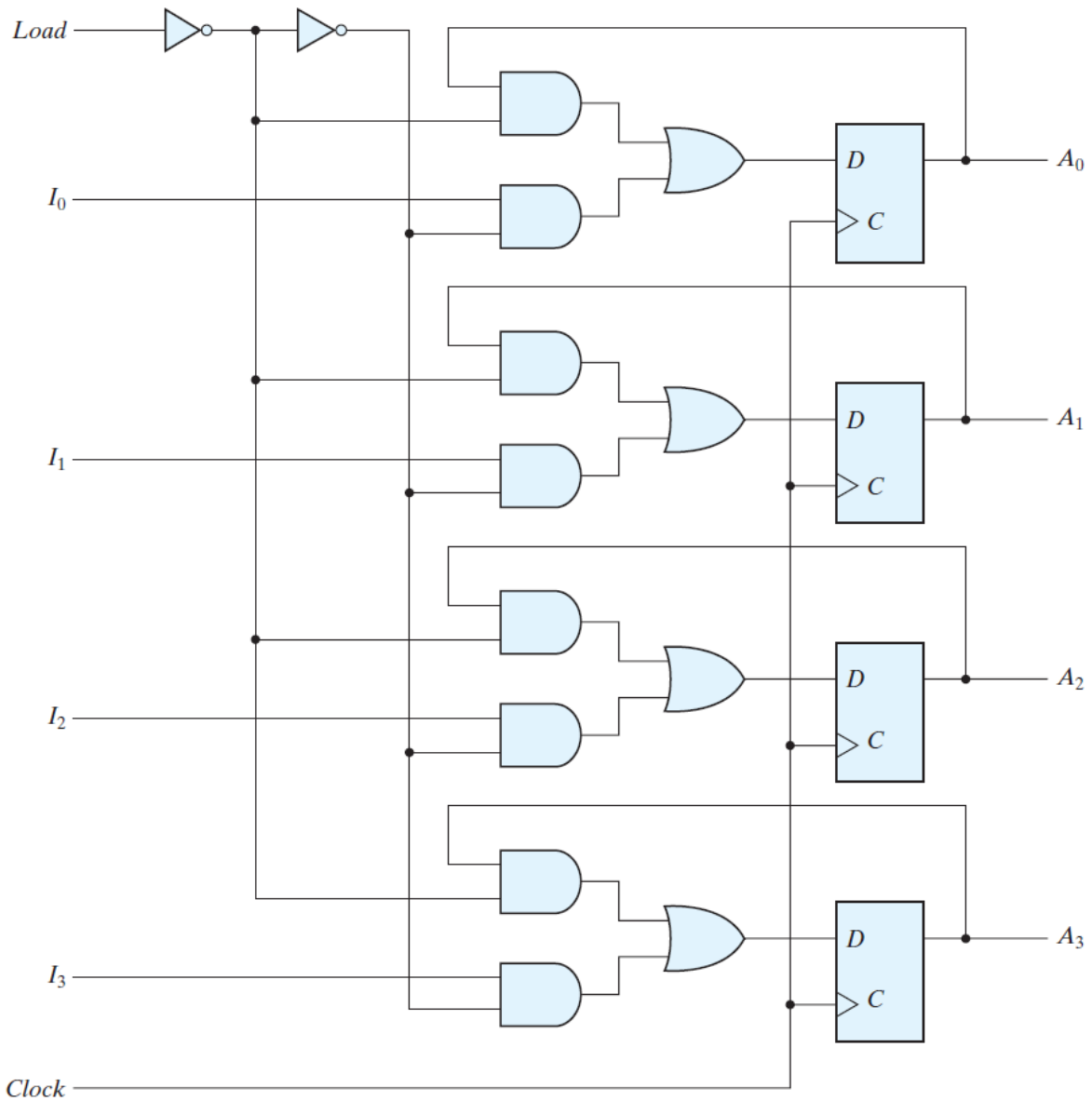
- Registers w/ parallel load are a fundamental building block in digital systems
  - Transfer of new information into a register is referred to as **loading** or **updating** the register
  - Parallel load?



Register with Parallel Load

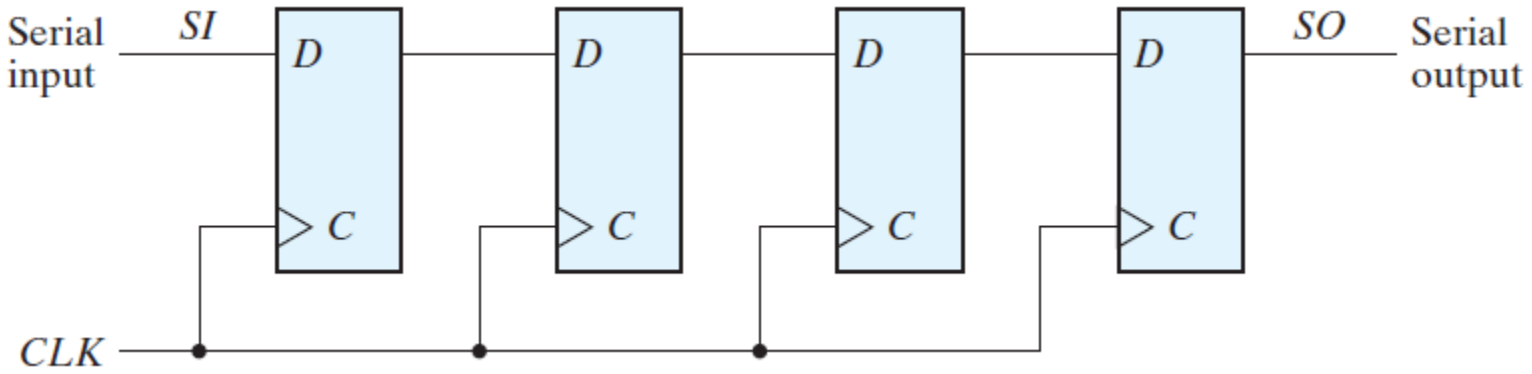
- If the contents of the register must be left unchanged…
  - Inputs must be held constant, OR
  - Clock must be inhibited from the circuit

- Inserting gates into clock path is ill advised as it may produce **uneven propagation delays** btw the master clock and the inputs of FFs



Shift Register

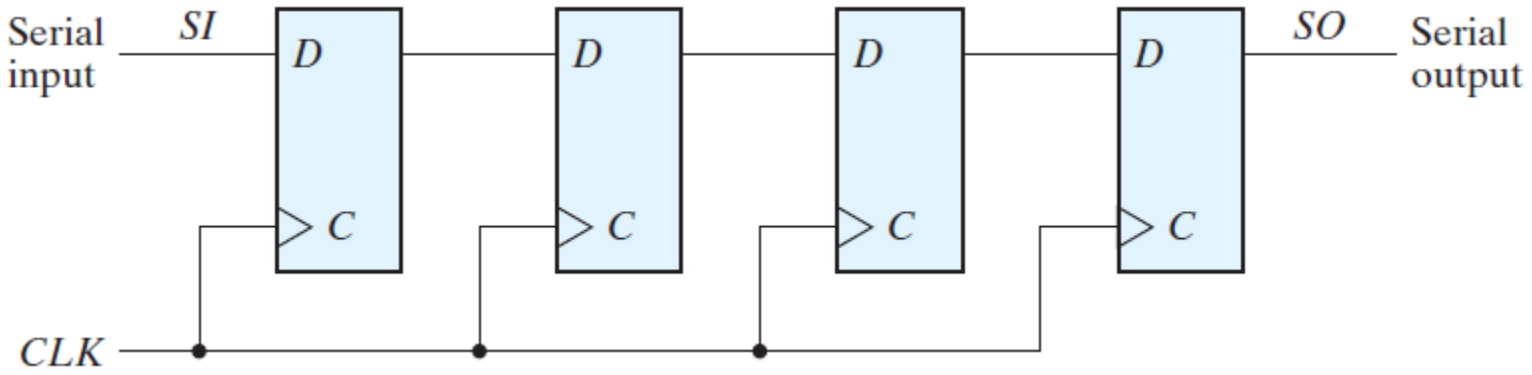
- A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction
  - Logical configuration: a chain of flip-flops in cascade
  - All FFs receive common clock pulses: activating the shift of data from one stage to the next



(Unidirectional) Shift Register

- This configuration does not support a left shift
  - The **serial output** is taken from the output of the rightmost FF
  - Sometimes it is necessary to control the shift so that it occurs only w/ certain pulses (how?)

Unidirectional (left-to-right)



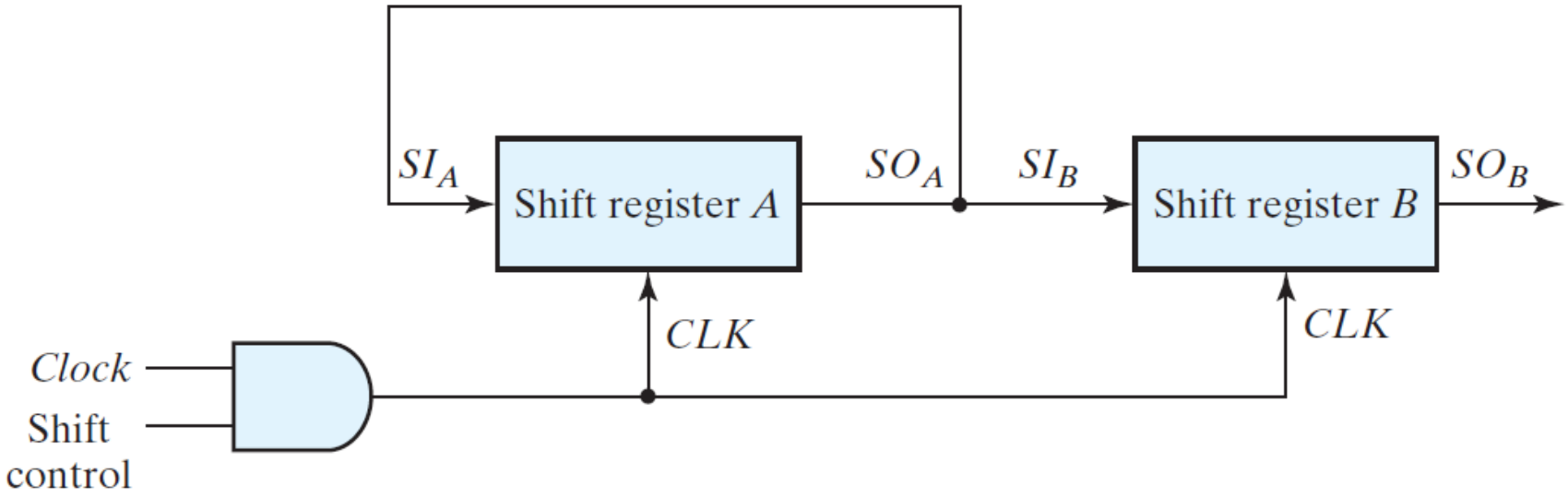
Serial Transfer

- The datapath of a digital system is said to operate in serial mode
  - When information is transferred and manipulated one bit at a time
  - Transfer by shifting a bit out of source register and into destination register

- This type of transfer is **in contrast to parallel transfer**, whereby all the bits of the register are transferred at the same time.

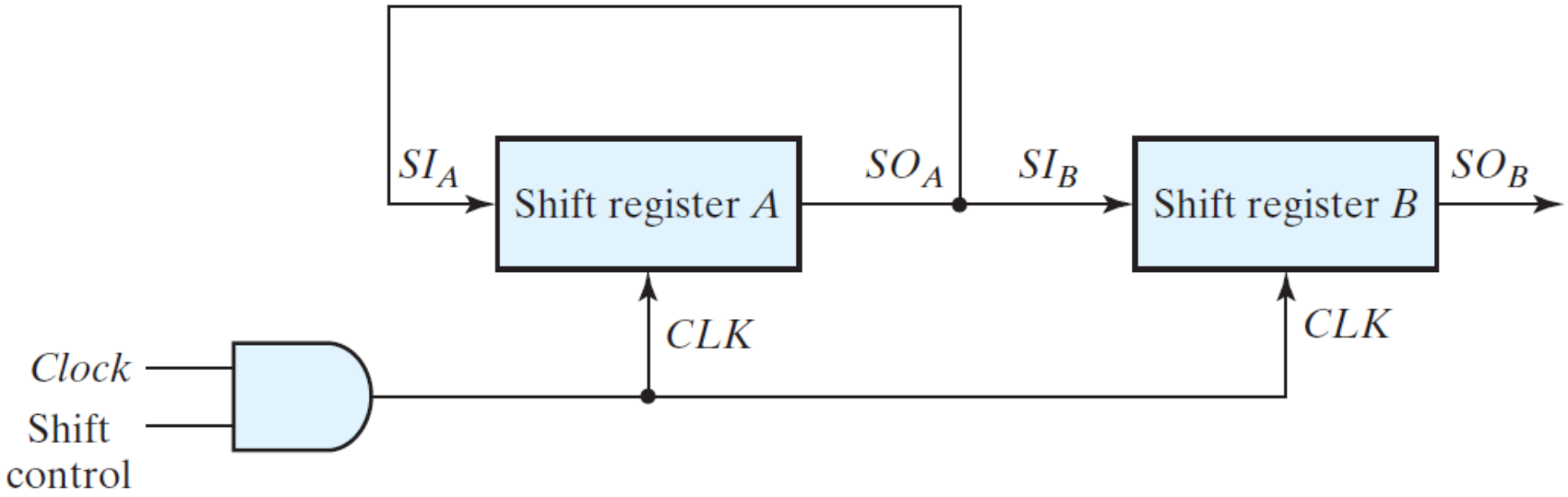
Block Diagram of Serial Transfer

- To prevent the loss of information stored in the source register
  - The information in register A is circulated back to its serial input
  - Initial content of register B is shifted out through its serial output and is lost



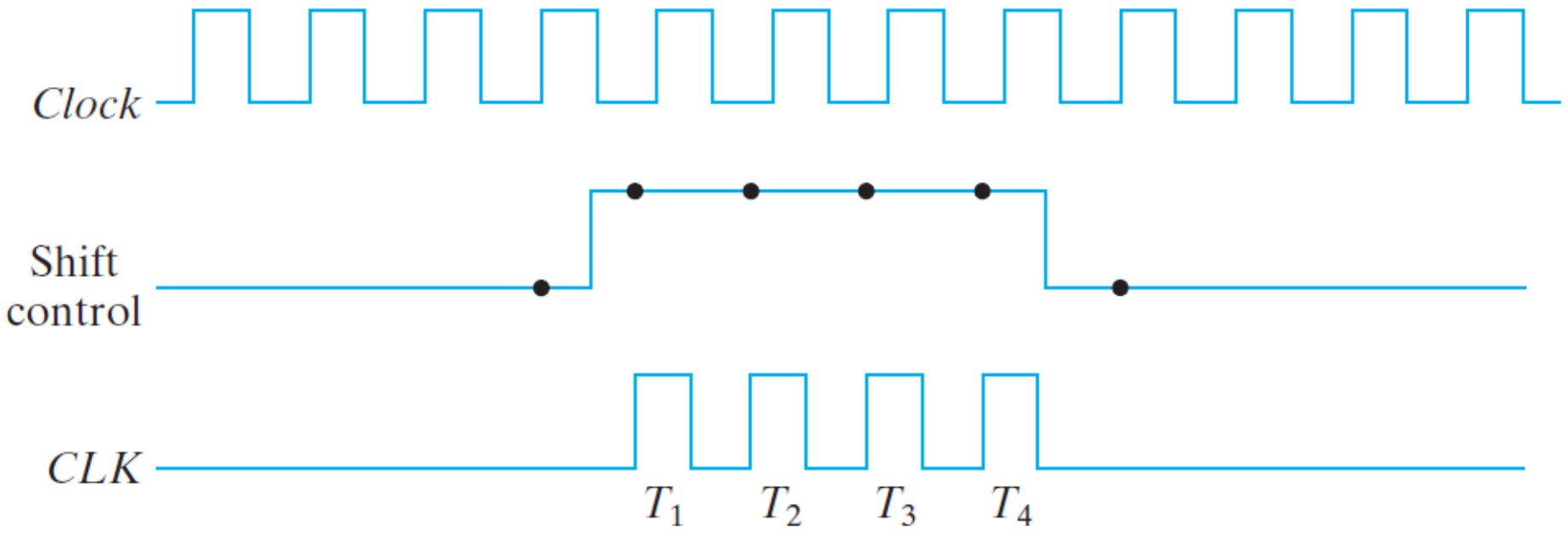
Shift Control in Serial Transfer

- The shift control input determines when and how many times the registers are shifted
  - Done with an AND gate allowing clock pulses to pass into CLK terminals
  - Note that this practice can be problematic (compromising clock path)



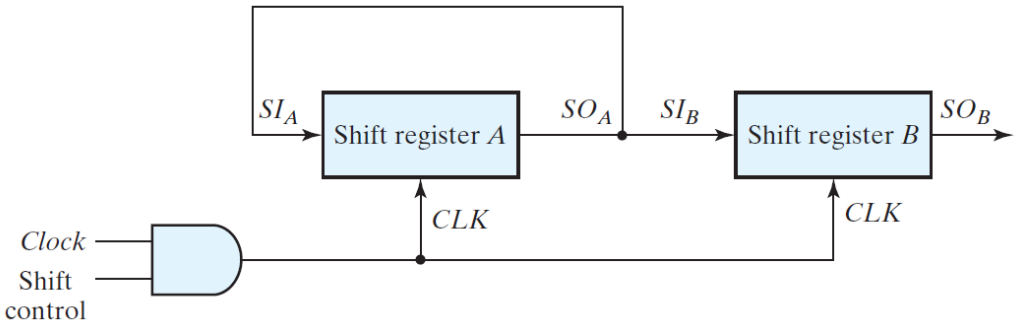
Four-bit Shift Register

- The control unit supervises the transfer for a fixed time of four clock pulses to pass an entire word
  - The shift control signal is synchronized w/ clock and changes value just after the negative edge of the clock



Four-bit Shift Register

- Assume that the binary content of A is 1011 and that of B is 0010
  - The rightmost bit of A is shifted into the leftmost bit of B
  - Also, it is circulated into the leftmost position of A
  - The previous serial output from B is lost



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

64-bits Shift Register in VHDL

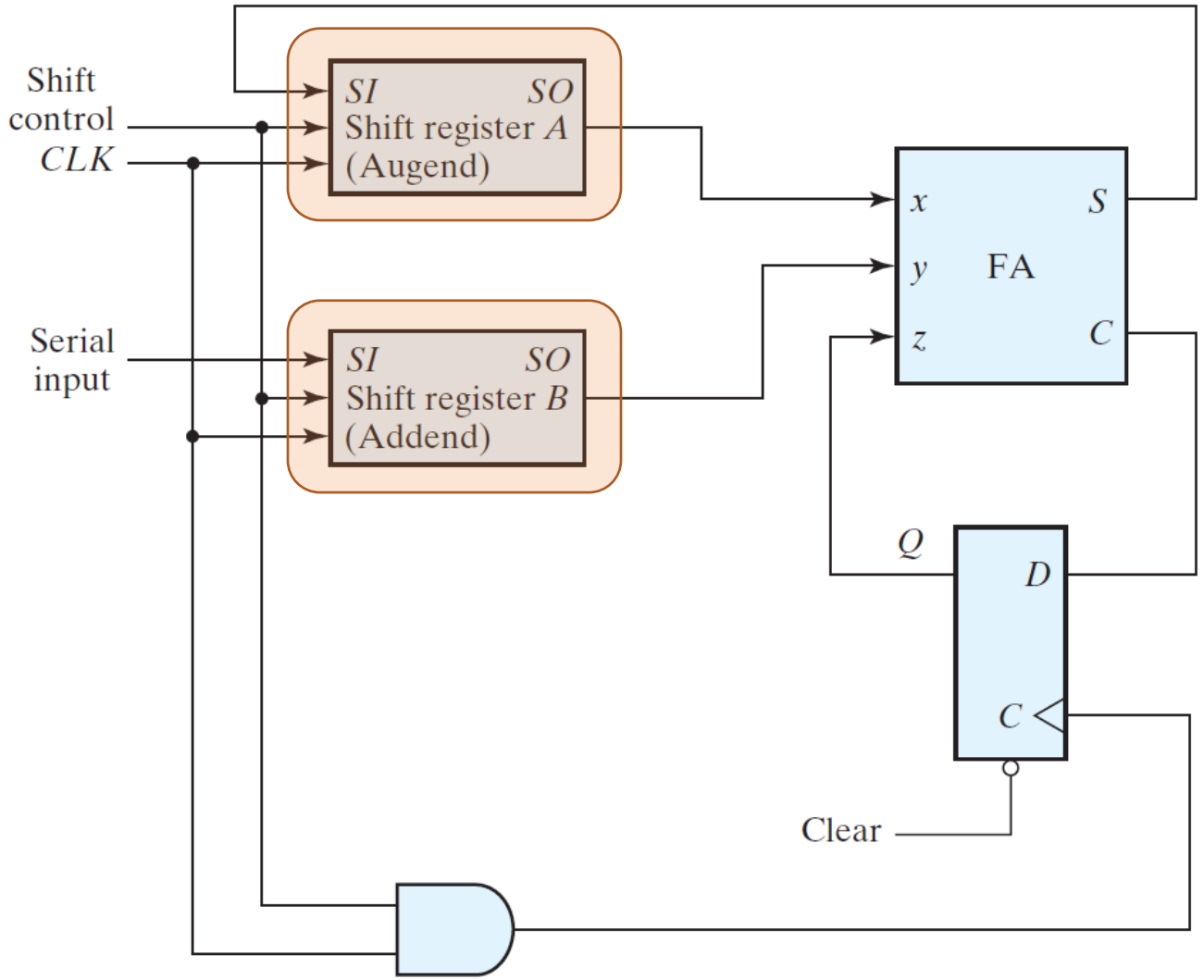
```
begin
        process (clk)
        begin
                if (rising_edge(clk)) then
                        if (enable = '1') then

                                -- Shift data by one stage; data from last stage is lost
                                sr(63 downto 1) <= sr(62 downto 0);

                                -- Load new data into the first stage
                                sr(0) <= sr_in;

                        end if;
                end if;
        end process;

        -- Capture the data from the last stage, before it is lost
        sr_out <= sr(63);
end rtl;
```

Serial Operations

- Operations in digital computers are usually done in parallel
  - For a faster mode of operation
  - Serial operations are slower as they require several clock cycles

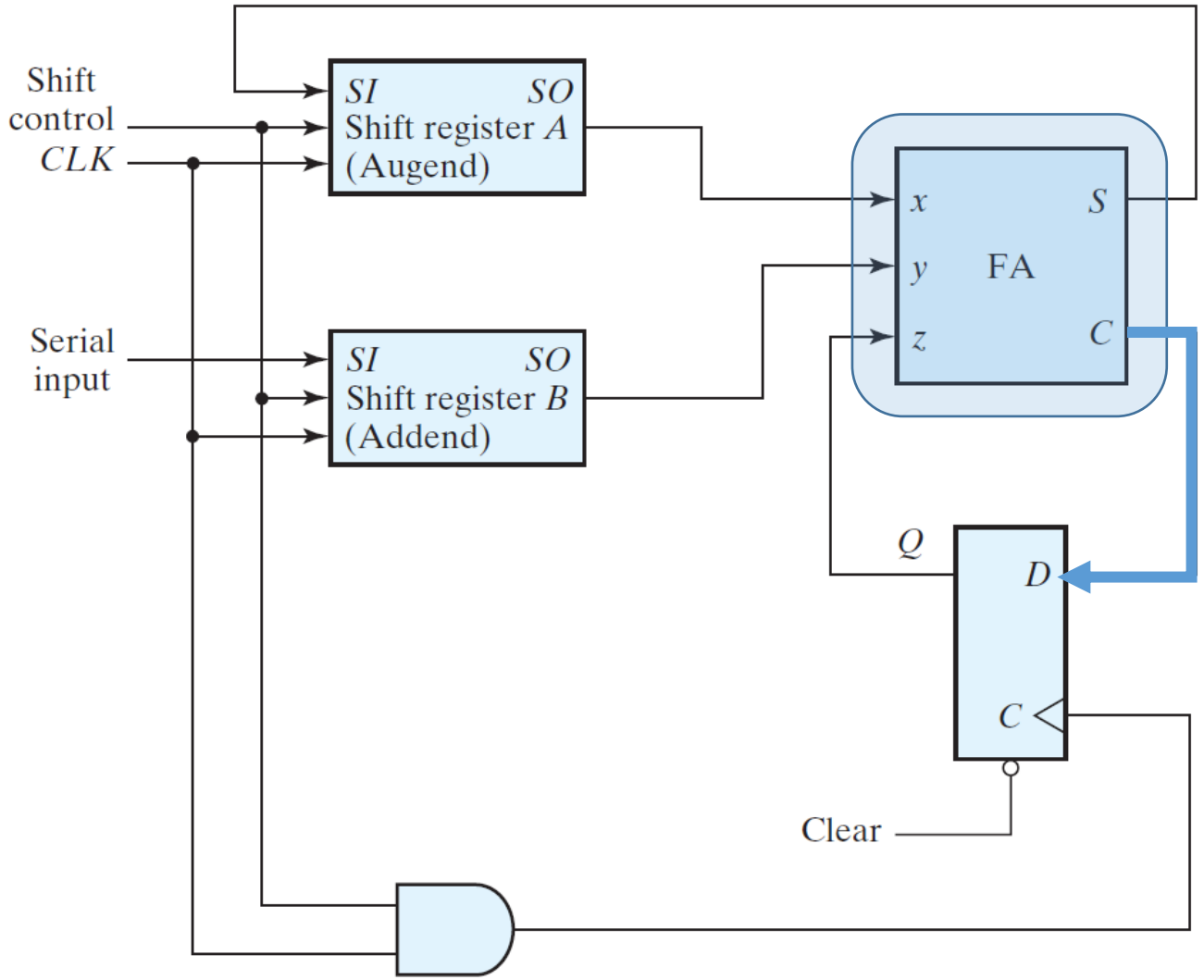
Serial Addition

- Let's look at a serial adder for an example
  - Two binary numbers are added serially
  - These operands are stored in two **shift registers**



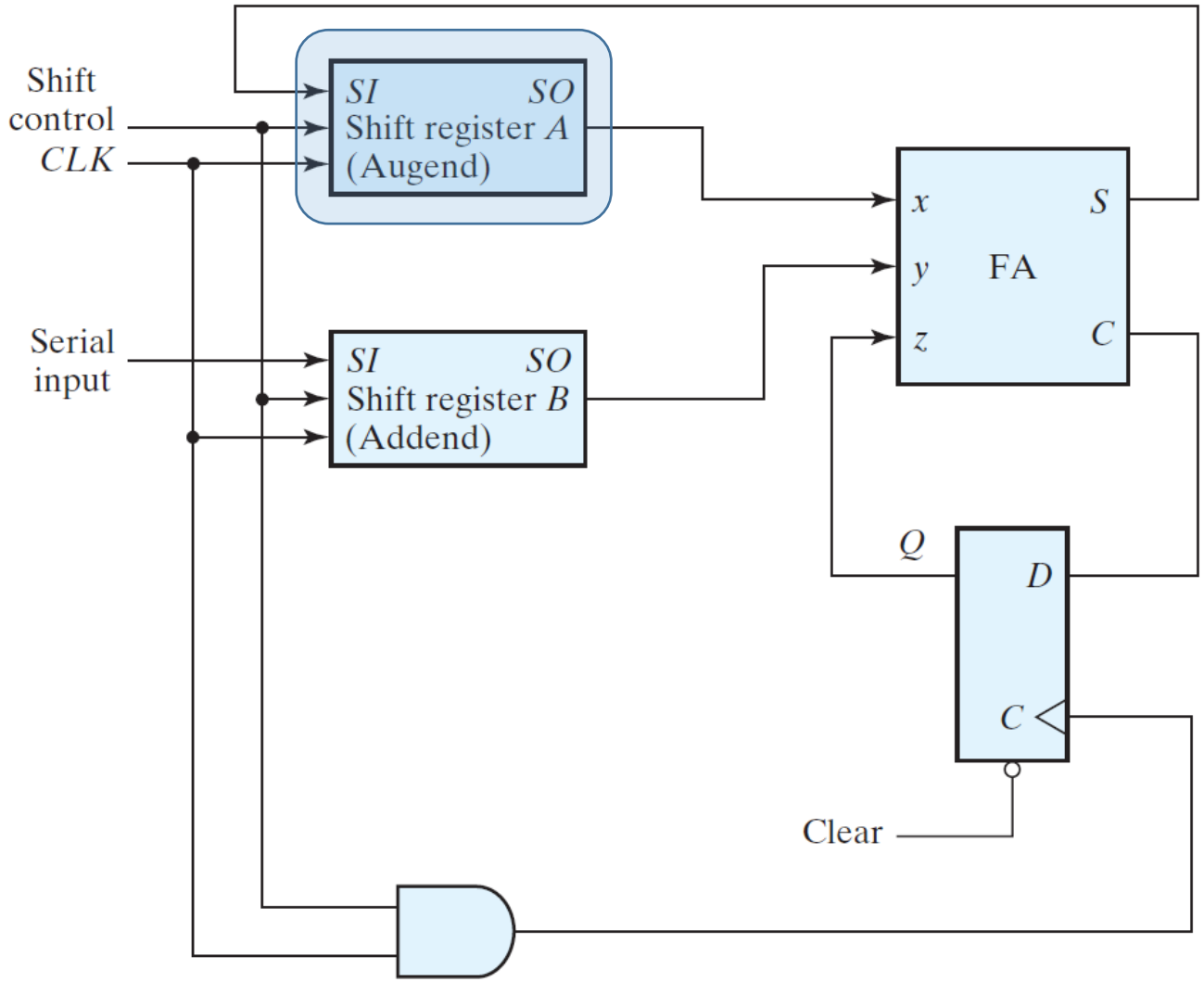
Serial Addition

- Beginning w/ the least significant pair of bits
  - A single full adder (FA) adds one pair at a time
  - The carry out is transferred to a D FF, then used as a carry input for the next pair



Serial Addition

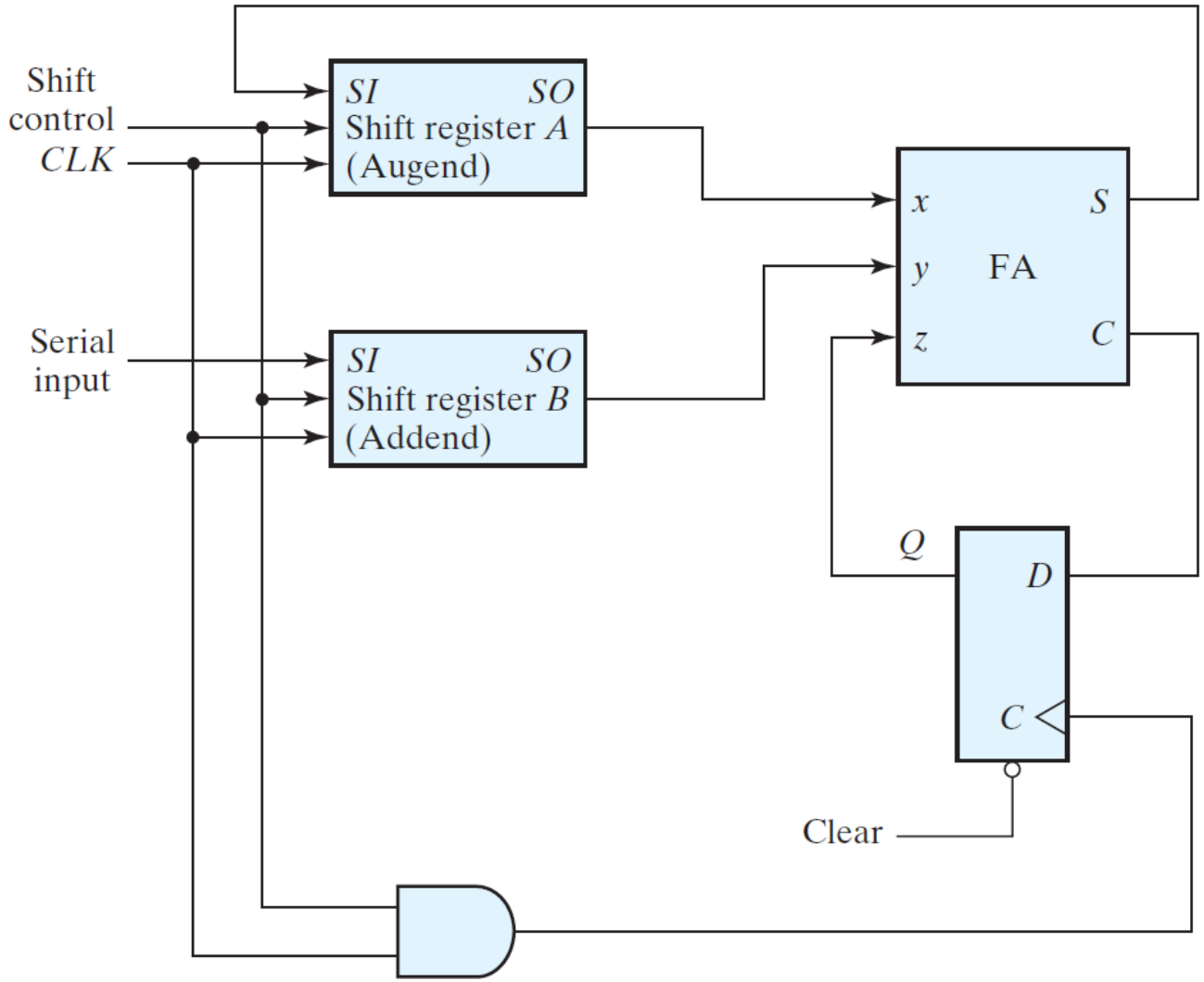
- Beginning w/ the least significant pair of bits
  - A single full adder (FA) adds one pair at a time
  - The carry out is transferred to a D FF, then used as a carry input for the next pair
  - By shifting sum into A while the bits of A are shifted out, we can **save the additional output register**



Serial Addition

- The shift control enables both registers and D FF
  - After enabled, a bit pair is shifted once to the right
  - The sum bit from S enters the leftmost flip-flop of A
  - The output carry is transfer-red into flip-flop *Q*

Until the shift control is disabled!



Serial Adder vs. Parallel Adder

- The parallel adder
  - Uses registers w/ parallel load
  - # of FAs = # of bits to be added

- The serial adder
  - Uses shift registers
  - Requires only one FA + **carry flip-flop**

Sequential circuit can be described by a state table

State Table for Serial Adder

- Two shift registers are available to store the binary numbers to be added serially
  - The serial outputs from the registers are designated by x and y

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

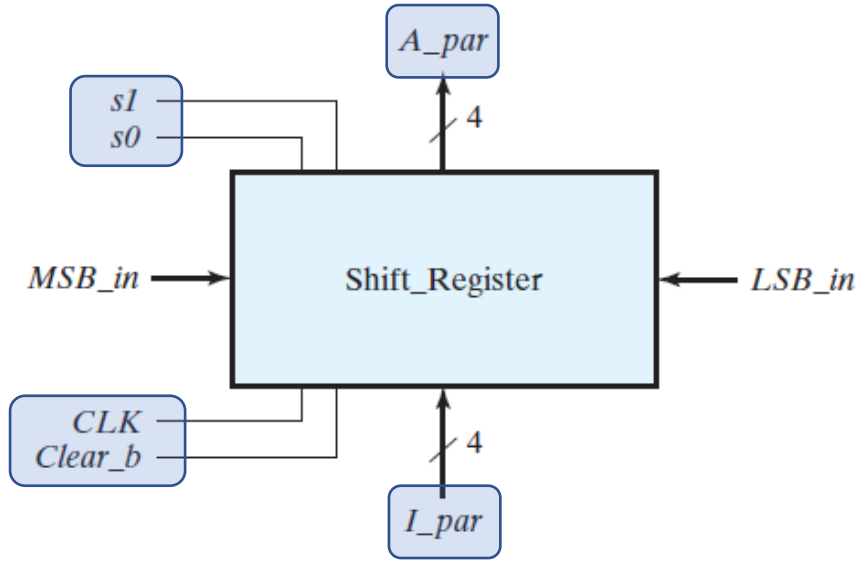
State Table for Serial Adder

- Two inputs x and y provide a pair of significant bits, an output S generates the sum bit, and FF Q for storing the carry
  - The present state Q is the input to carry-in (added together w/ x and y)

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

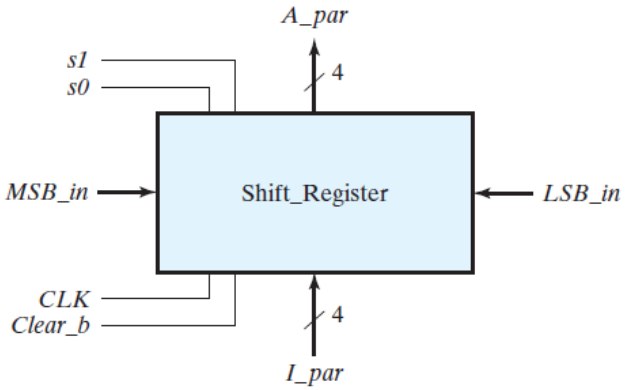
Universal Shift Register

- Some shift registers provide necessary input and output terminals for parallel transfer
  - If FF **outputs of a shift register are accessible**, then information entered serially by shifting can be **taken out in parallel**
  - If a **parallel load capability** is available in a shift register, then data entered in parallel can be **taken out in serial** fashion

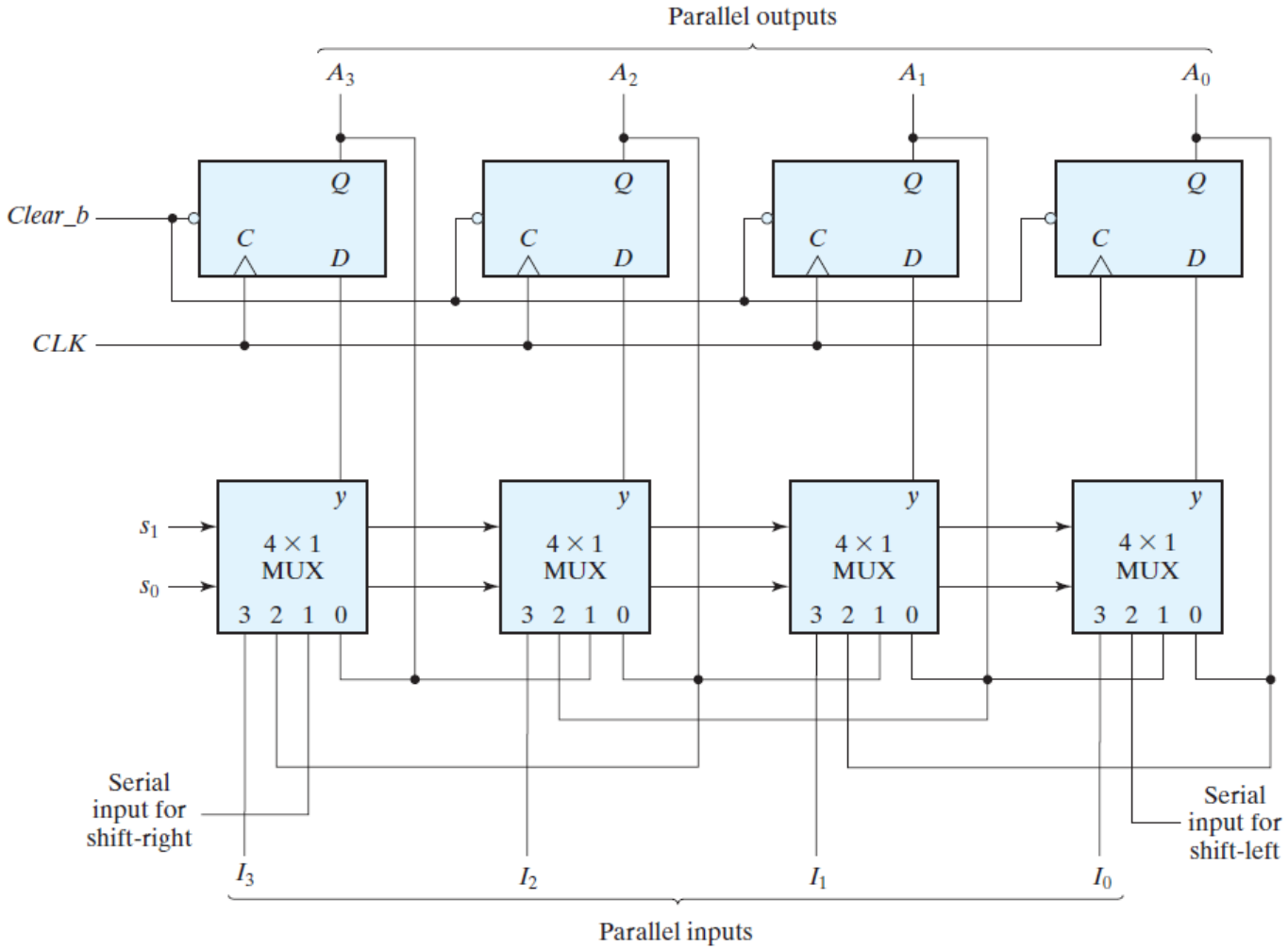


4-bit Universal Shift Register

- Universal: bidirectional shift + parallel-load



Mode Control		
s_1	s_0	Register Operation



Registers

Clocked Sequential Circuit

- It consists of a group of FFs and combinational gates
  - A circuit w/ FFs is a sequential circuit even w/o combinational gates

- Circuits with FFs are usually classified by the function
  - **Counter**: a register that goes into a predetermined sequence of binary states
  - Gates are connected in a way to produce the prescribed sequence of states
  - Counters are a special type of register

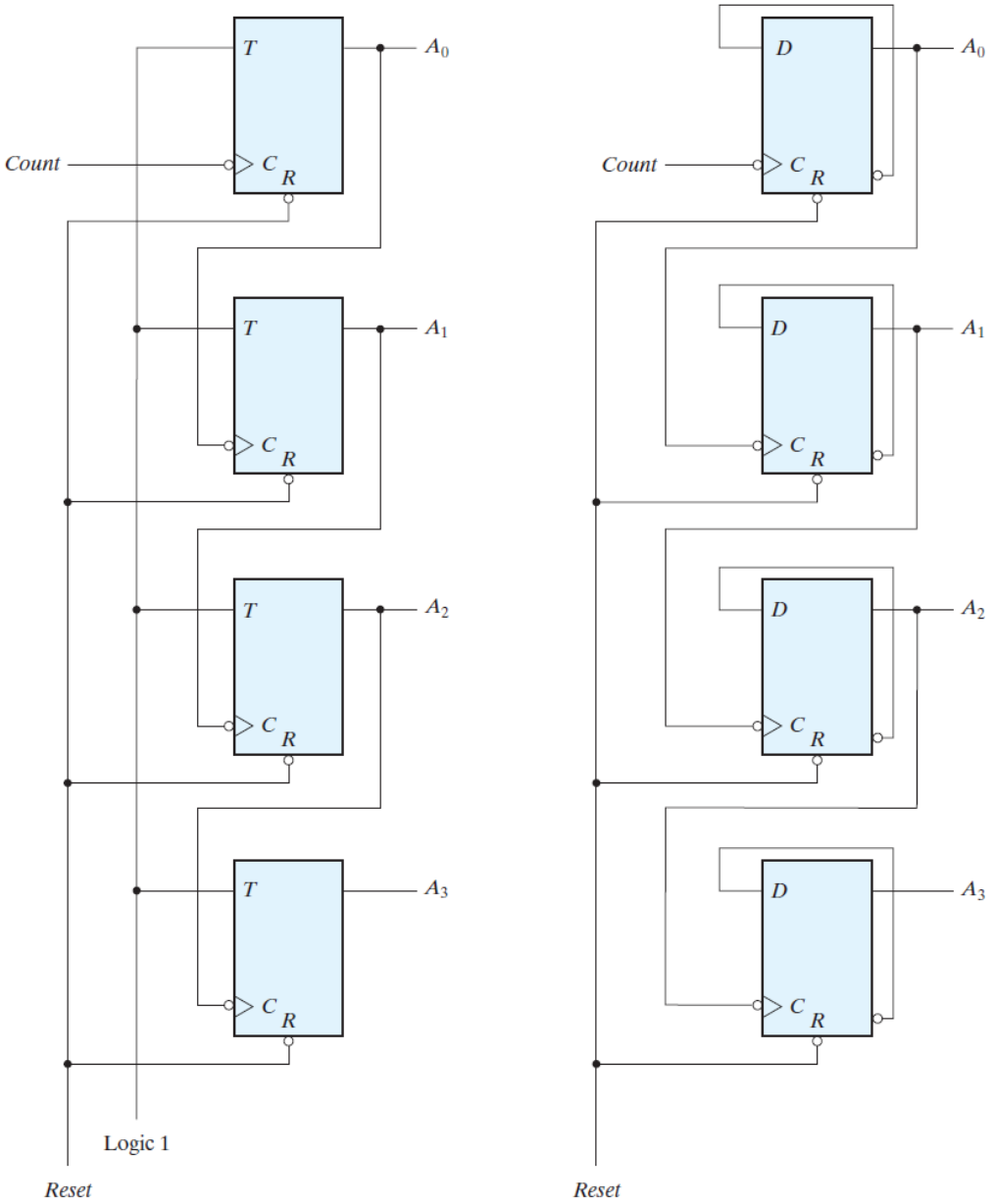
Binary Counter

- A counter that follows the binary number sequence
  - ‘n’ flip-flops can count in binary from 0 to 2^n-1

- There are two categories
  - Ripple counters
  - Synchronous counters

Binary Ripple Counter

- It consists of a series connection of complementing FFs
  - With output of a FF connected to C input of the next higher-order FF
  - The FF holding the least significant bit receives the incoming count pulses.

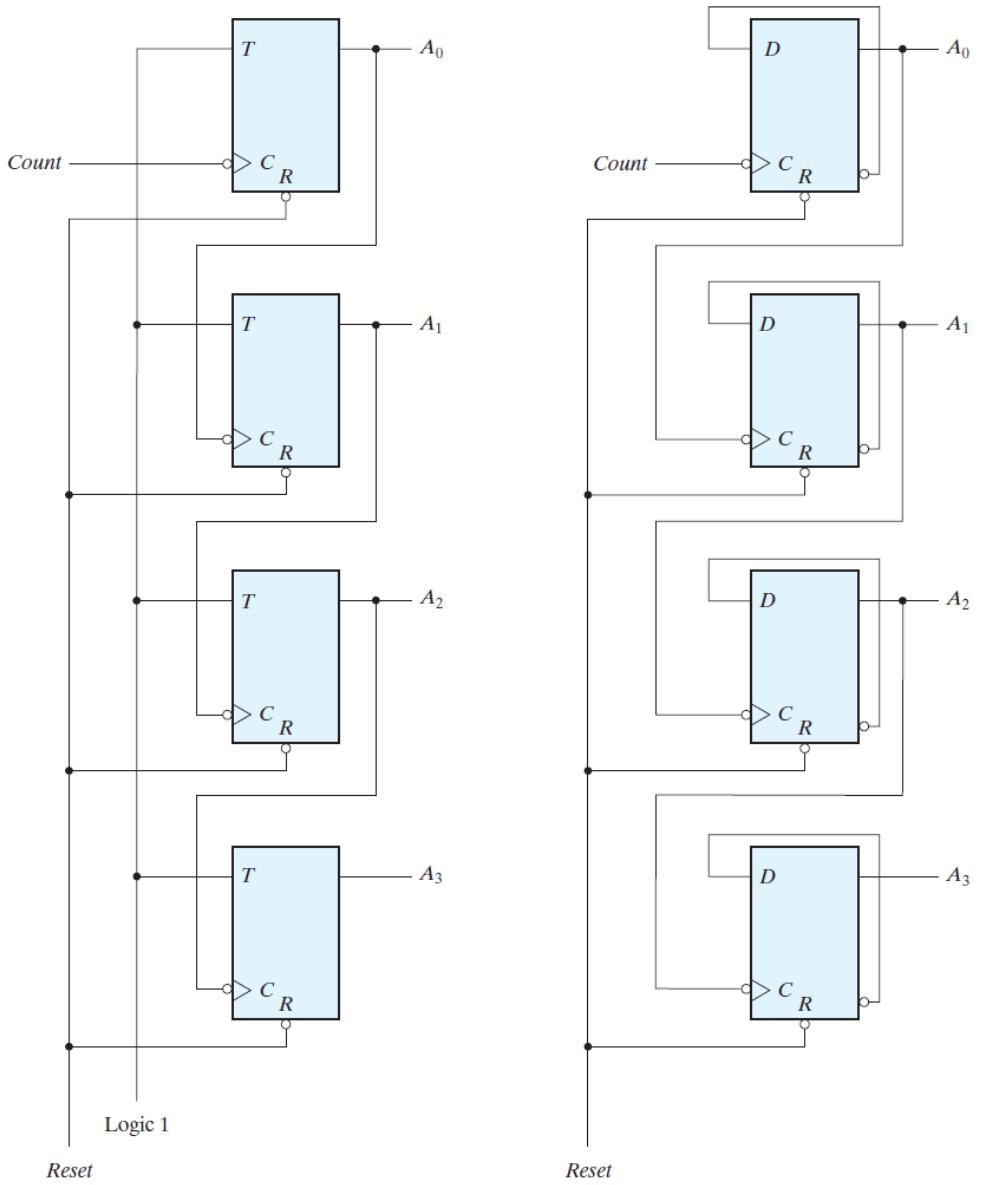


Binary Ripple Counter

- Let's look at first nine binary numbers for its operation
  - The count starts with binary 0 and increments by 1 with each count pulse input

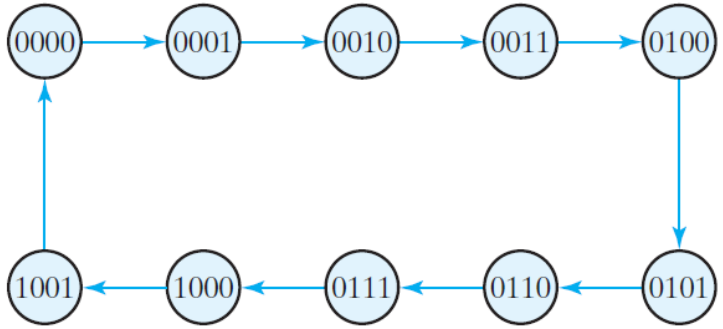
A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

How to design a countdown counter?



Binary Ripple Counter

- A decimal counter follows a sequence of 10 states and returns to 0 after the count of 9
  - It must have at least four FFs to represent each decimal digit

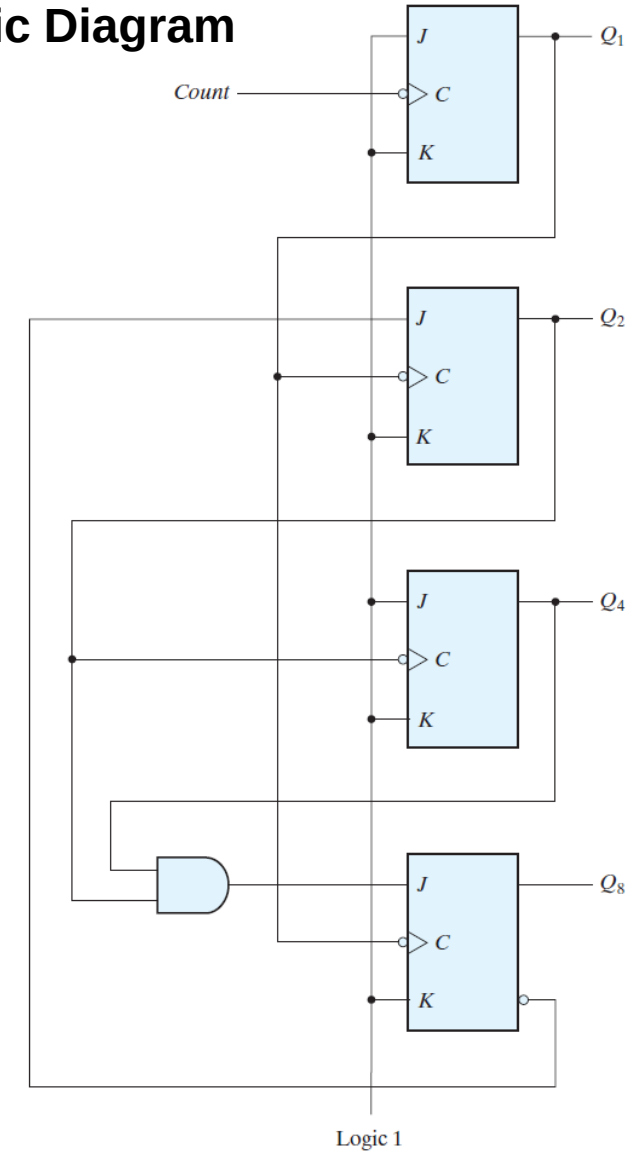


State Diagram

JK Flip-Flop			
J	***K***	***Q*(*t* + 1)**	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

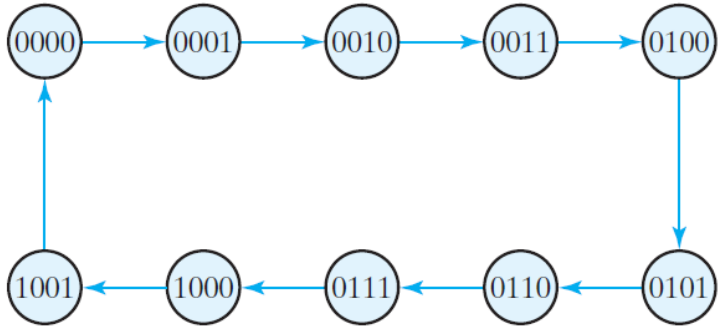
Characteristic Table

Logic Diagram



Binary Ripple Counter

- Signals that affect the FF transition depend on the way they change from 1 to 0

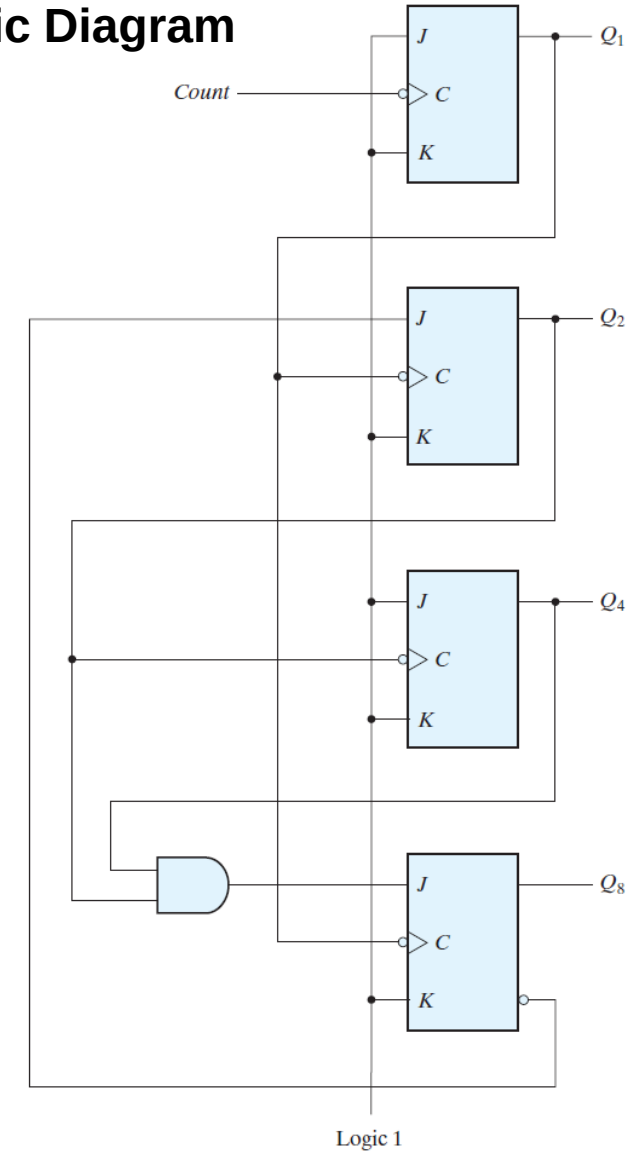


State Diagram

JK Flip-Flop			
J	***K***	***Q*(*t* + 1)**	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

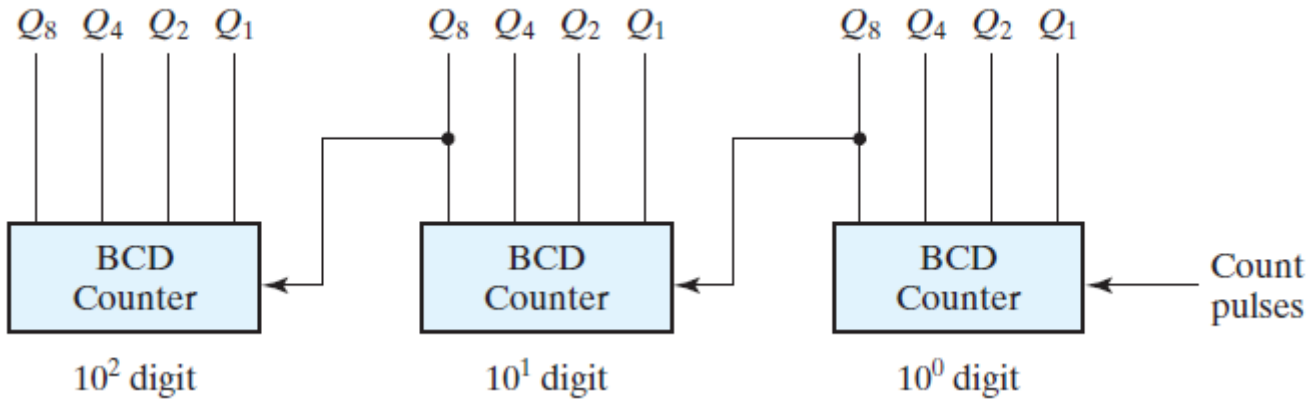
Characteristic Table

Logic Diagram



3-Decade Decimal BCD 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 Q8 of previous decade

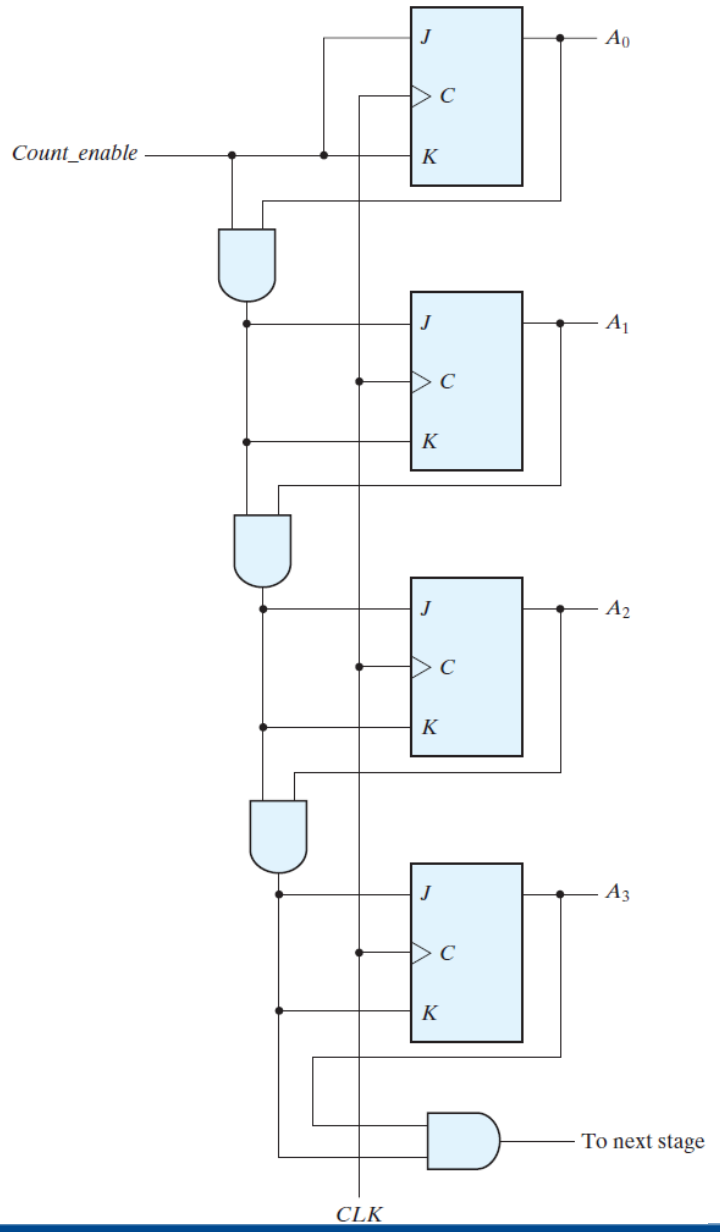


Synchronous Counters

- Clock pulses are applied to the inputs of all flip-flops in synchronous counters
  - The decision whether a flip-flop is to be complemented is determined from the values of the data inputs
  - If T = 0 or J = K = 0, the flip-flop does not change state
  - If T = 1 or J = K = 1, the flip-flop complements

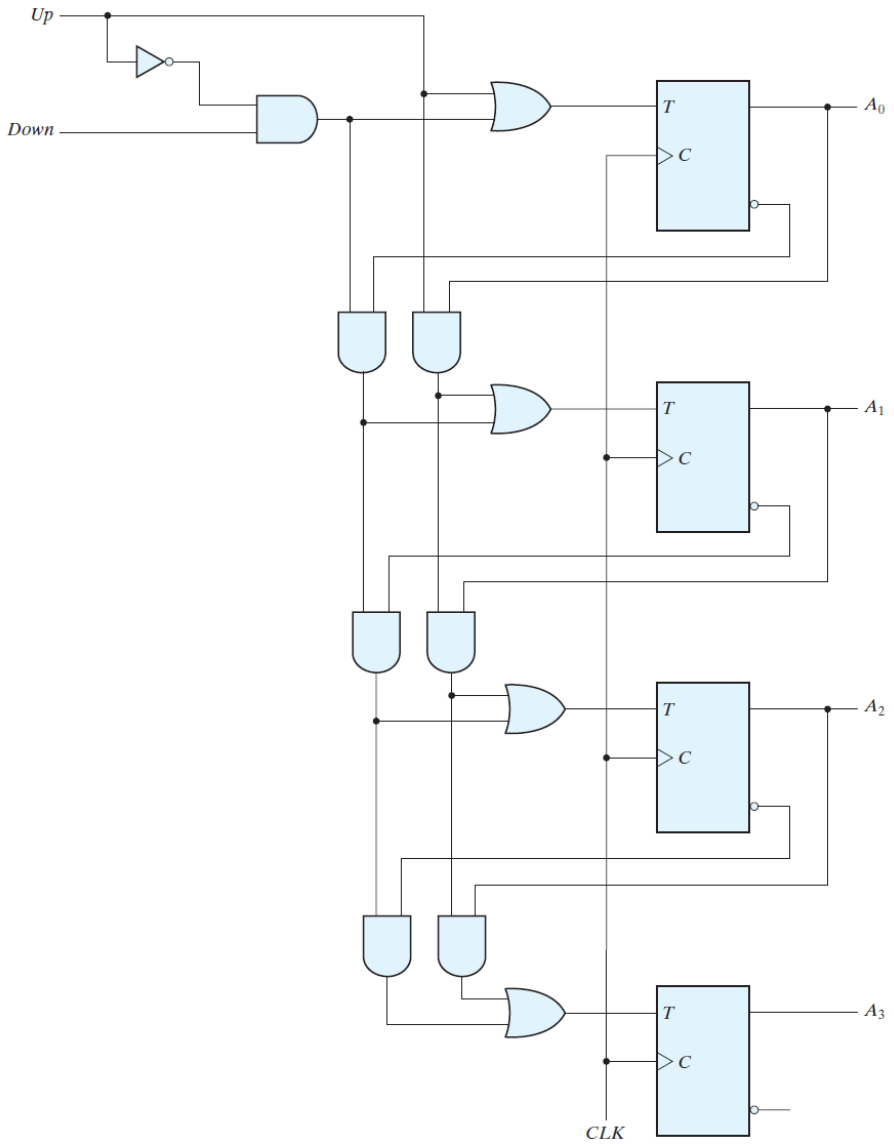
Synchronous Binary Counters

- The least significant position is comple-mented w/ every pulse
- A flip-flop in any other position is **complemented** when **all the bits** in the **lower** significant **positions are equal to 1**



Synchronous Up-Down Binary Counters

- Countdown counter
  - The least significant position is comple-mented w/ each pulse
  - A flip-flop in any other position is **complemented** when **all the bits** in the **lower** significant **positions are equal to 0**



Let's combine up counter with down counter

BCD Counter

- It counts binary-coded decimal from 0000 to 1001 and back to 0000
  - As returning to 0 after a count of 9, it does not have a regular pattern
  - Thus, we need to go through a **sequential design procedure**

State Table

Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_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

Design of a BCD Counter

- The flip-flop input equations can be simplified by means of maps
  - The unused states for minterms 10 to 15 are taken as don't-care terms
  - The circuit consists of four T flip-flops, five AND gates, and one OR gate

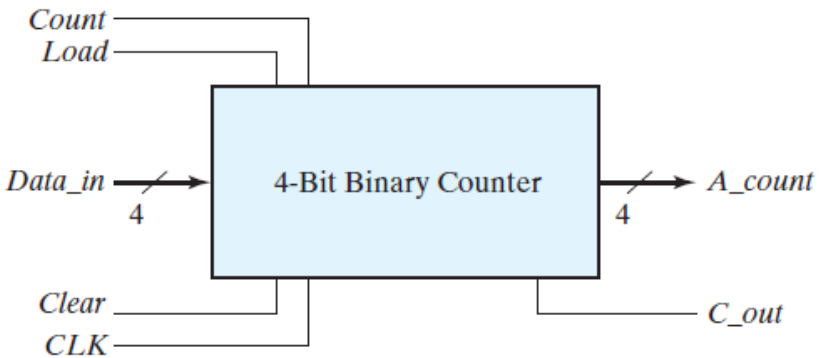
State Table

Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_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

$$T_{Q1} = 1$$
$$T_{Q2} = Q'_8 Q_1$$
$$T_{Q4} = Q_2 Q_1$$
$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$y = Q_8 Q_1$$

Binary Counter with Parallel Load

- Counters quite often require a parallel-load capability
  - Ex) transferring an initial binary number into the counter prior to its operation
  - When '**Load=1**', the input load control disables the count operation and causes a transfer of data
  - If **both control inputs are 0**, clock pulses do not change the state of the register

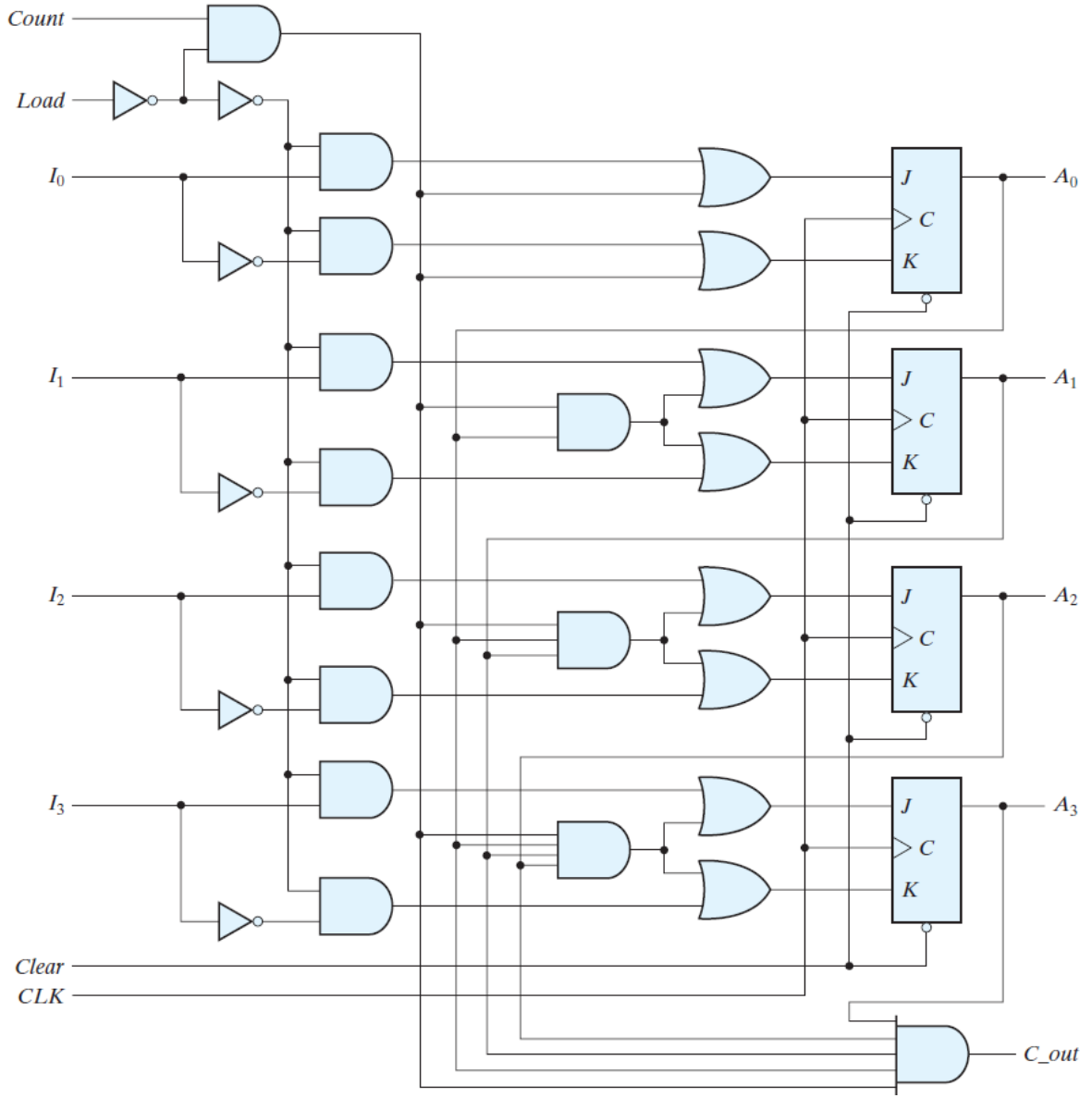


Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

Binary Counter with Parallel Load

- The ‘C_out = 1’ if all the flip-flops are equal to 1 while the count input is enabled
  - Expand the counter w/ more than 4 bits
  - Speed of the counter is increased when the carry is generated directly from the outputs of **all four** flip-flops (why?)

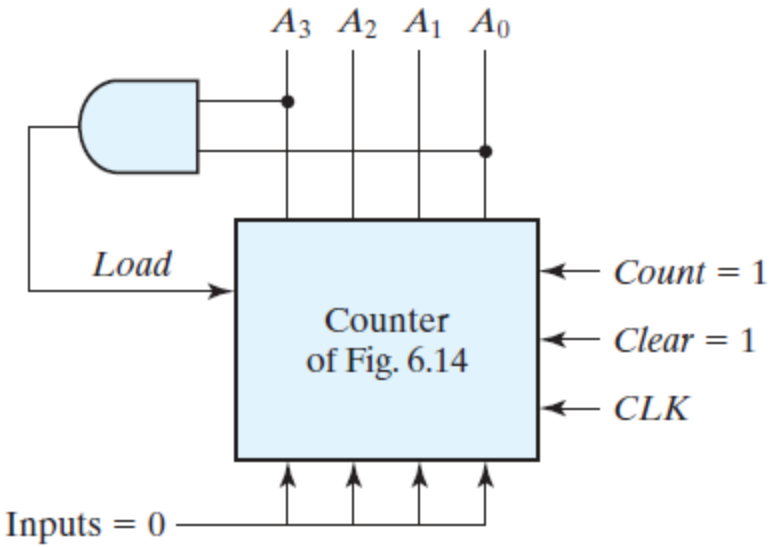
Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change



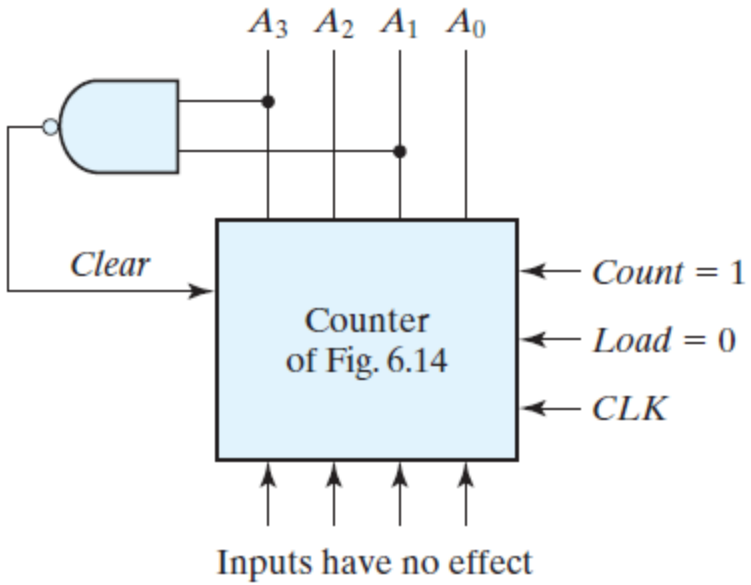
BCD Counter Using a Counter w/ Parallel Load

- Using ‘Load’ input, we detect state 1001 (clears on next clock edge)
- Using ‘Clear’ input, count 1010 is detected and clears immediately

Using the ‘Load’ Input



Using the ‘Clear’ Input



Counter in VHDL

```
begin
        process (clk)
        begin
                if (rising_edge(clk)) then
                        if reset = '1' then
                                -- Reset the counter to 0
                                cnt <= (others => ‘0’);
                        elsif enable = '1' then
                                -- Increment the counter if counting is enabled
                                cnt <= cnt + ‘1’;
                        end if;
                end if;

                -- Output the current count
                q <= cnt;
        end process;

end rtl;
```

Summary

- Learned different types of clocked sequential circuits
  - Registers
  - Counters

- Learned how to implement of registers and counters
  - Design procedure for registers or counters
  - Design different types of counters in Verilog