# EECS 3201: Digital Logic Design Lecture 12

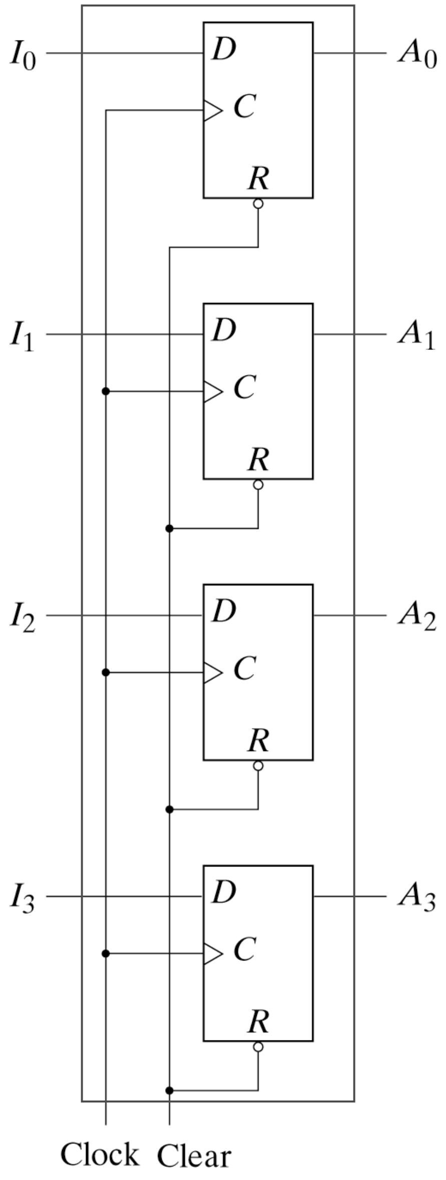
Ihab Amer, PhD, SMIEEE, P.Eng.

## Registers

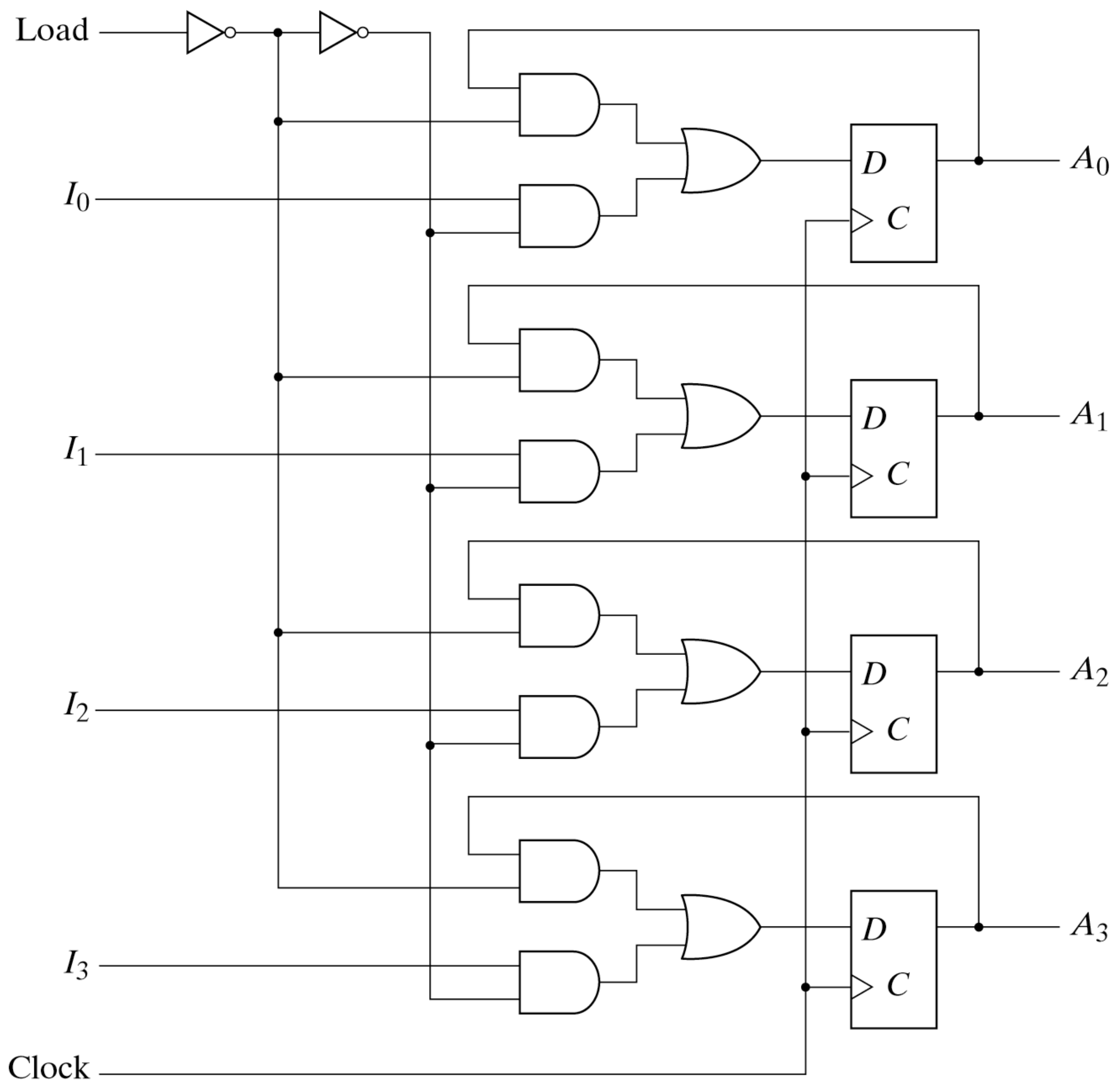
- Collections of flip-flops with special controls and logic
  - Stored values somehow related (e.g., form binary value)
  - Share clock, reset, and set lines
- Examples
  - Shift registers
  - Counters

# Four-bits Register

- Parallel load
- Clock must be inhibited from the circuit if the contents of the register is to be left unchanged (enabling gate)
- Performing logic with clock pulses inserts variable delays and may cause the system to go out of synchronism
- Solution: Direct the load control input through gates and into the FF inputs

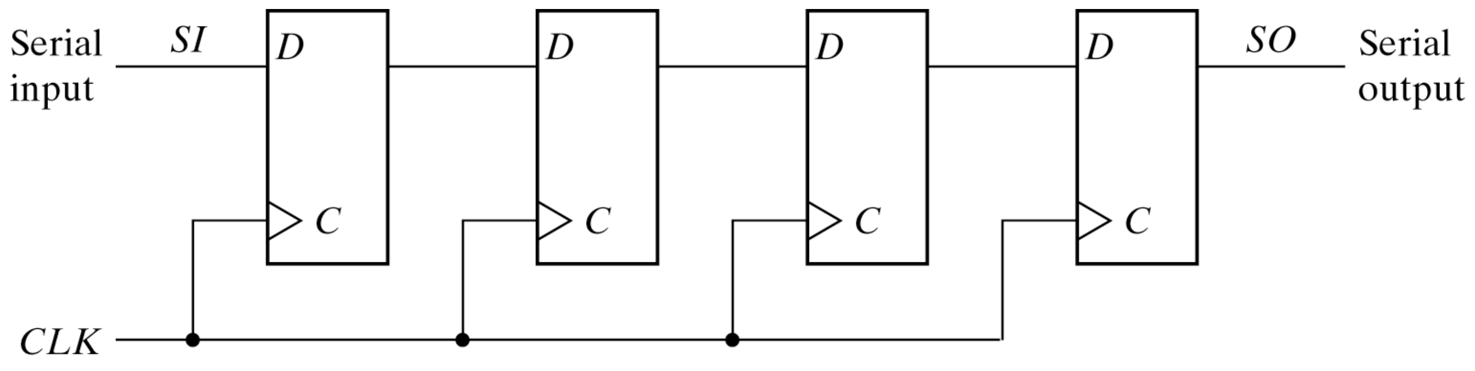


# Registers with Parallel Load



## Shift Registers

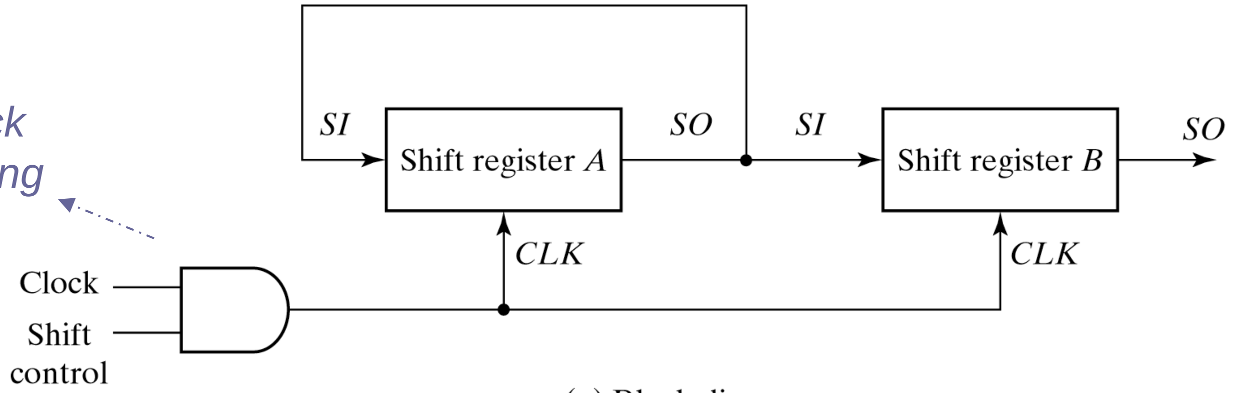
- A Register capable of shifting its content in one *or* both directions is called a *shift register*. It has many applications such as *serial transfer* and *serial addition*



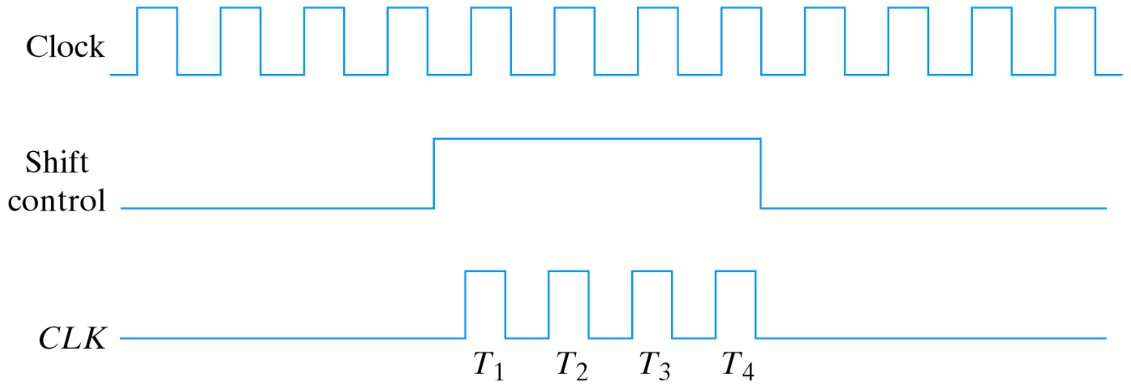
Serial-In/Serial-Out 4-bit shift register

# Serial Transfer

Clock Gating



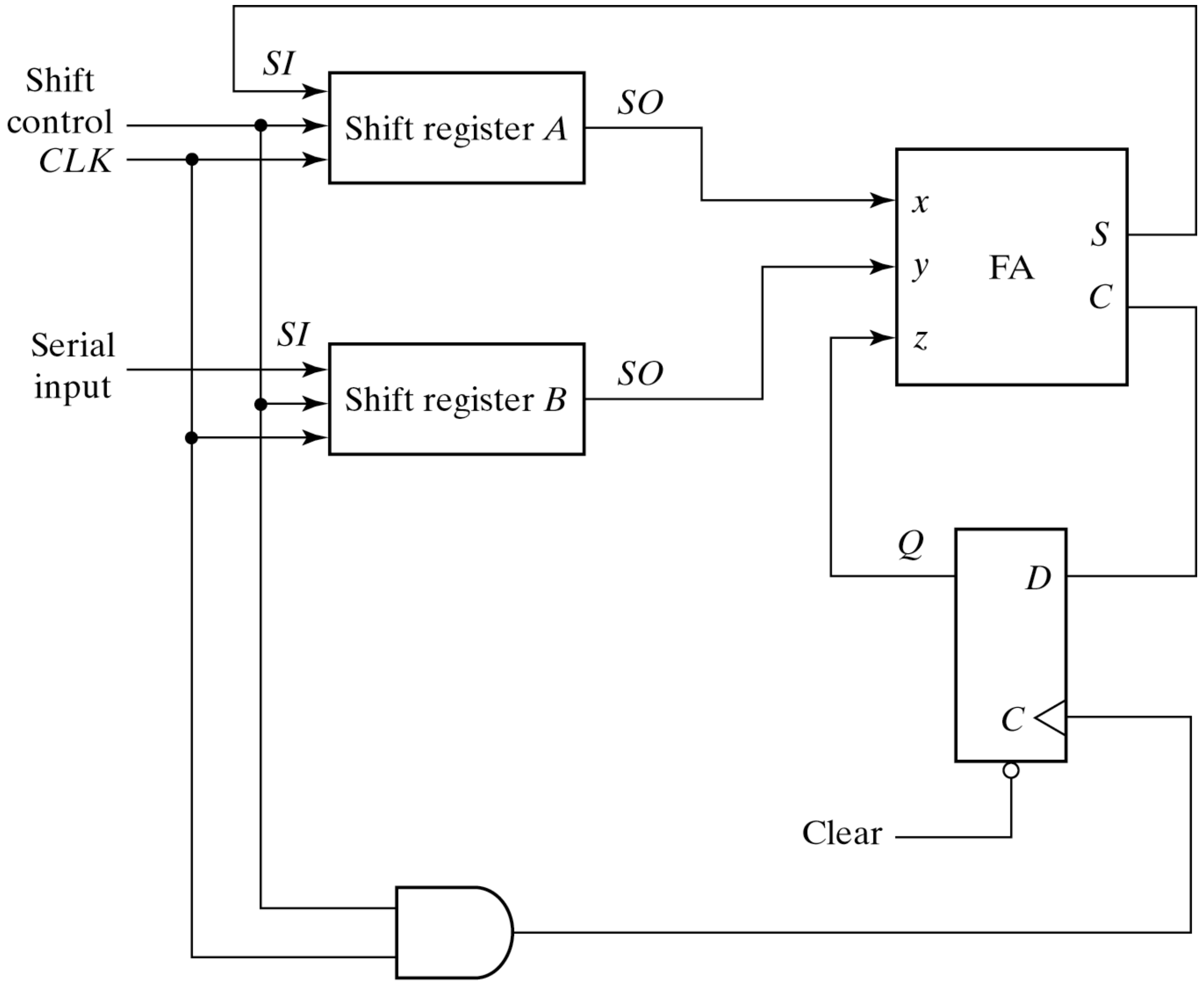
(a) Block diagram



(b) Timing diagram

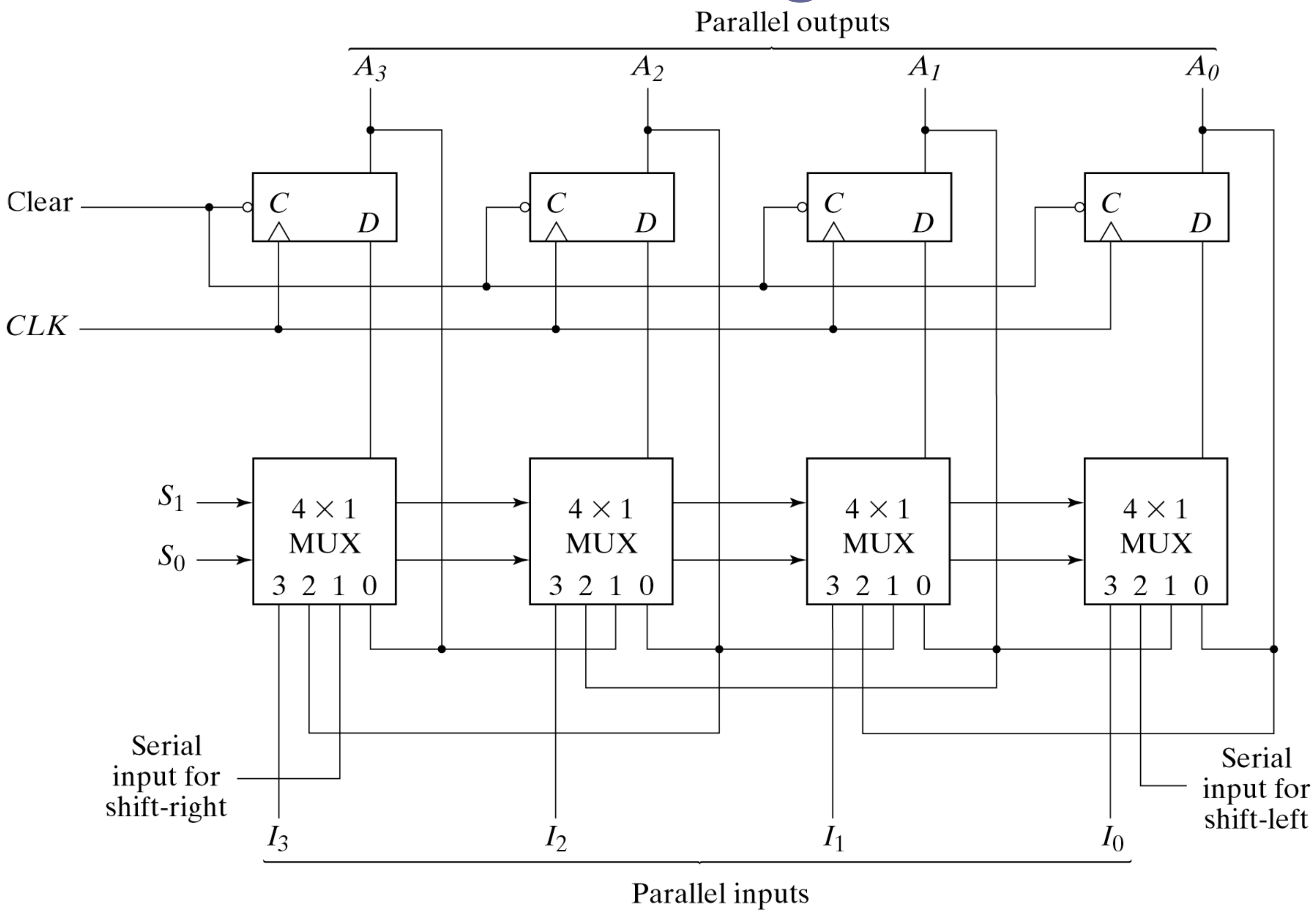
Serial Transfer from Register *A* to register *B*

# Serial Adder



**Refer to the text-book for the *design* of another form of a serial adder using JK FF**

# Universal Shift Register



# HDL for U Shift Register

| Mode Control | | Register Operation |
|---|---|---|
| $S_1$ | $S_0$ | |
| 0 | 0 | No Change |
| 0 | 1 | Shift Right |
| 1 | 0 | Shift Left |
| 1 | 1 | Parallel Load |

**Behavioral Description**

```
module shftreg (s1,s0,Pin,lfin,rtin,A,CLK,Clr);
  input s1,s0;                //Select inputs
  input lfin, rtin;           //Serial inputs
  input CLK,Clr;              //Clock and Clear
  input [3:0] Pin;            //Parallel input
  output [3:0] A;             //Register output
  reg [3:0] A;
  always @ (posedge CLK or negedge Clr)
  if (~Clr) A = 4'b0000;
  else
      case ({s1,s0})
       2'b00: A = A;                  //No change
       2'b01: A = {rtin,A[3:1]};  //Shift right
       2'b10: A = {A[2:0],lfin};  //Shift left
       2'b11: A = Pin;               //Parallel load input
      endcase
endmodule
```

# Structural Description

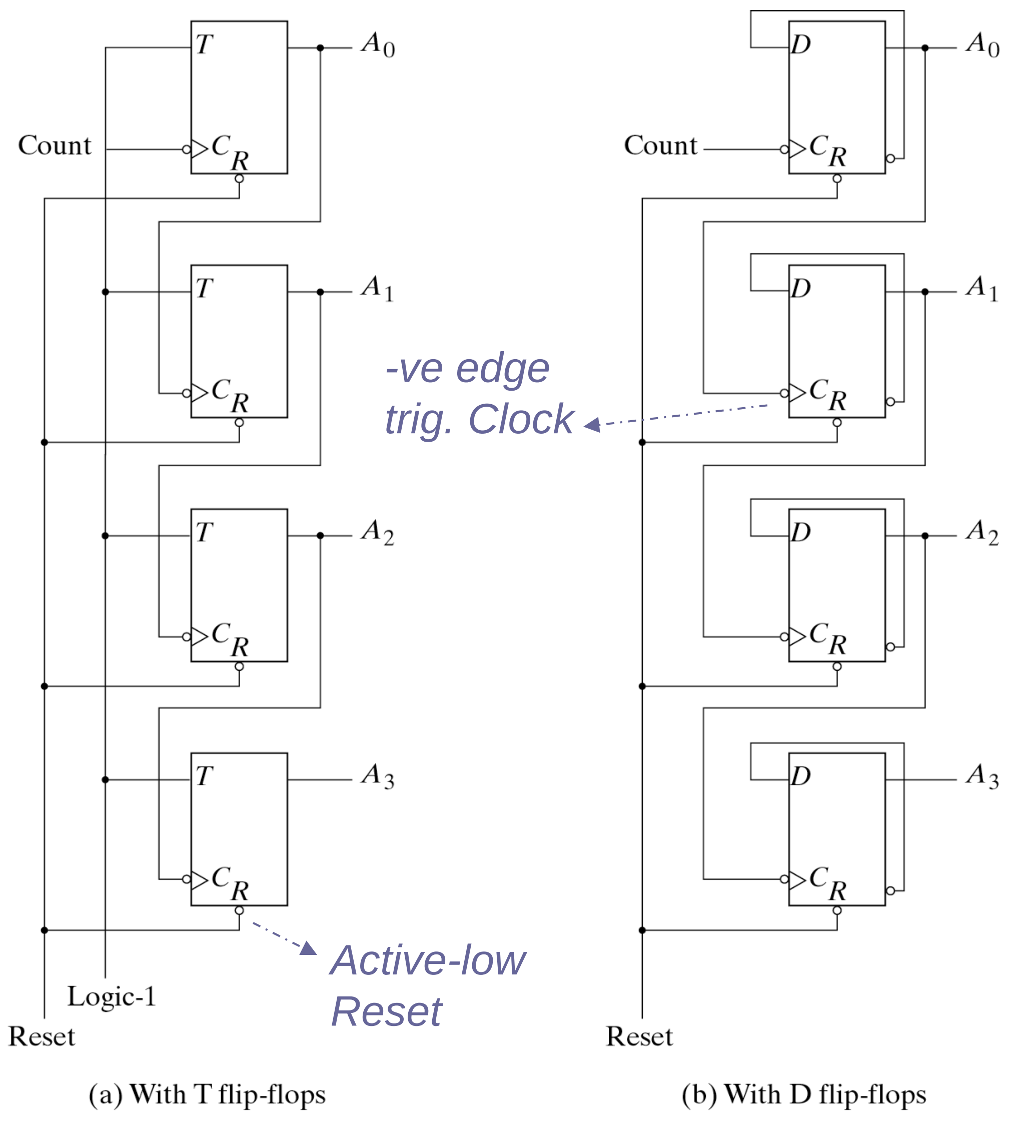
```
module SHFTREG
(I,select,lfin,rtin,A,CLK,Clr);
   input [3:0] I;            //Parallel input
   input [1:0] select;       //Mode select
   input lfin,rtin,CLK,Clr;  //Serial
                             //inputs,clock,clear
   output [3:0] A;           //Parallel output
 //Instantiate the four stages
   stage ST0
(A[0],A[1],lfin,I[0],A[0],select,CLK,Clr);
   stage ST1
(A[1],A[2],A[0],I[1],A[1],select,CLK,Clr);
   stage ST2
(A[2],A[3],A[1],I[2],A[2],select,CLK,Clr);
   stage ST3
(A[3],rtin,A[2],I[3],A[3],select,CLK,Clr);
endmodule
```

```
module stage(i0,i1,i2,i3,Q,select,CLK,Clr);
   input i0,i1,i2,i3,CLK,Clr;
   input [1:0] select;
   output Q;
   reg Q;
   reg D;
   //4x1 multiplexer
   always @ (i0 or i1 or i2 or i3 or select)
       case (select)
         2'b00: D = i0;
         2'b01: D = i1;
         2'b10: D = i2;
         2'b11: D = i3;
       endcase
   //D flip-flop
   always @ (posedge CLK or negedge Clr)
       if (~Clr) Q = 1'b0;
       else Q = D;
endmodule
```

## Counters

- A register that goes through a prescribed sequence of states upon the application of input pulses is called a counter
- Examples of counters are ripple and synchronous counters

# Ripple Counters



4-Bit Binary Ripple Counter

**Sequence of States**

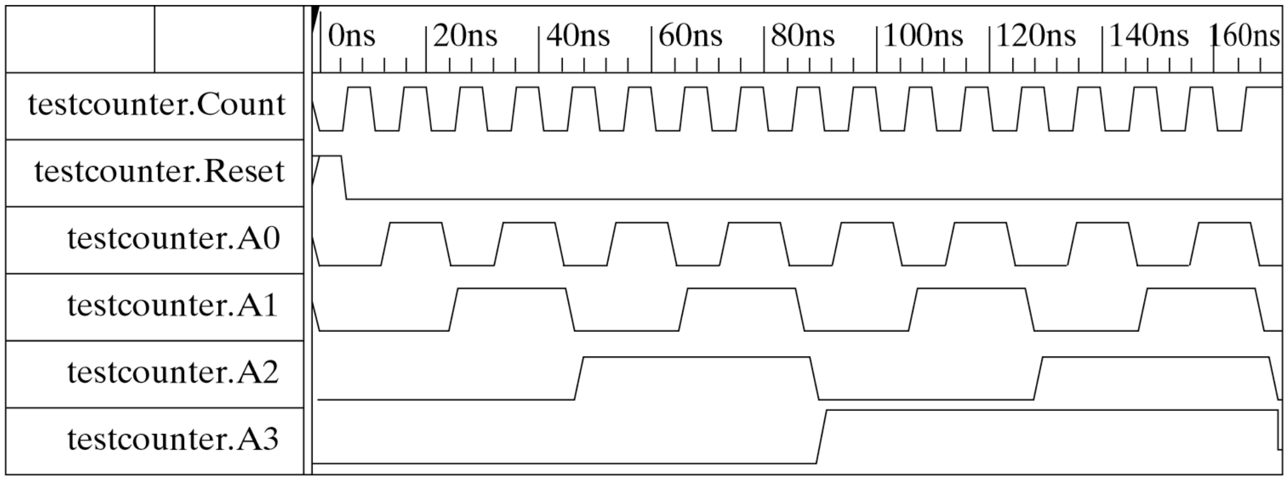
| $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

# HDL for ripple counter

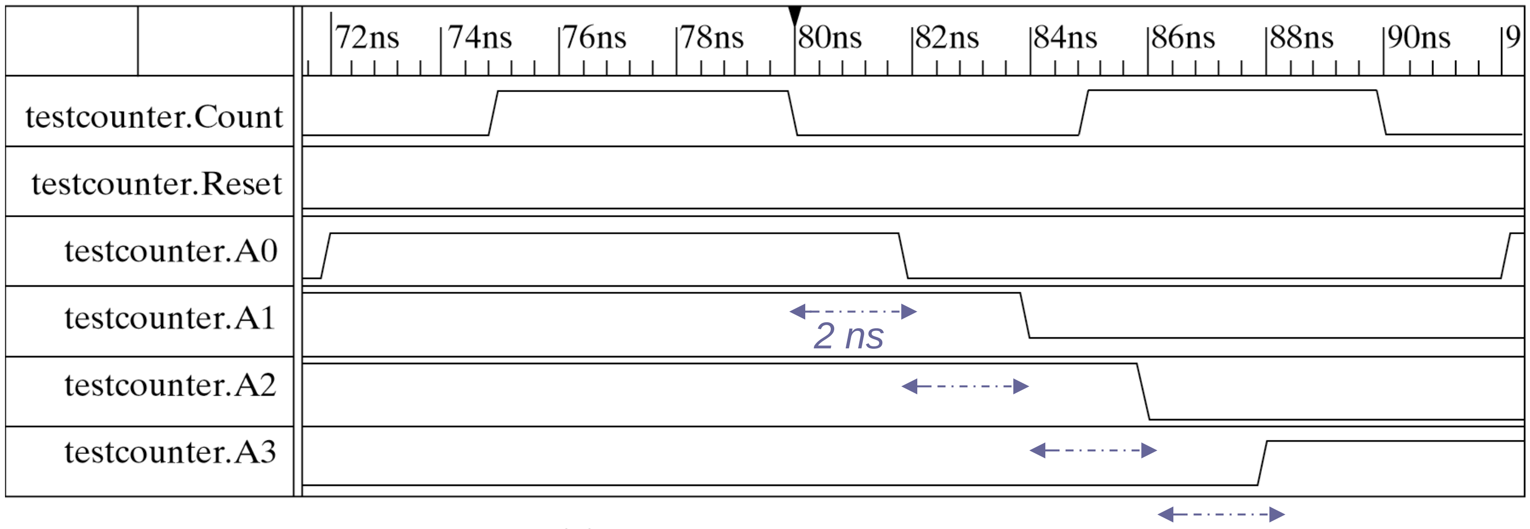
```
module ripplecounter (A0,A1,A2,A3,Count,Reset);
   output A0,A1,A2,A3;
   input Count,Reset;
//Instantiate complementing flip-flop
   CF F0 (A0,Count,Reset);
   CF F1 (A1,A0,Reset);
   CF F2 (A2,A1,Reset);
   CF F3 (A3,A2,Reset);
endmodule
//Complementing flip-flop with delay
//Input to D flip-flop = Q'
module CF (Q,CLK,Reset);
   output Q;
   input CLK,Reset;   Active-low         Active-high
   reg Q;             Clock              Reset
   always @ (negedge CLK or posedge Reset)
      if (Reset) Q = 1'b0;
      else  Q = #2 (~Q);     // Delay of 2 time units
endmodule
```

```
//Stimulus for testing ripple counter
module testcounter;
   reg Count;
   reg Reset;
   wire A0,A1,A2,A3;
//Instantiate ripple counter
   ripplecounter RC
(A0,A1,A2,A3,Count,Reset);
always
    #5 Count = ~Count;
initial
 begin
     Count = 1'b0;
     Reset = 1'b1;
   #4 Reset = 1'b0;
   #165 $finish;
 end
endmodule
```

# Simulation Output



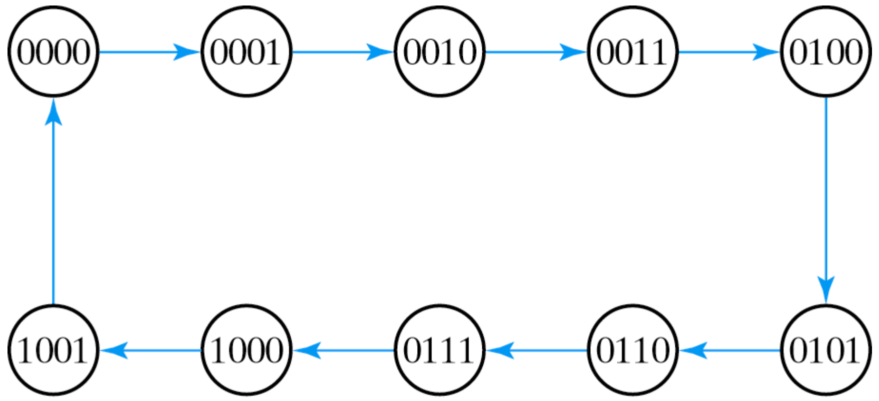
(a) From 0 to 170 ns



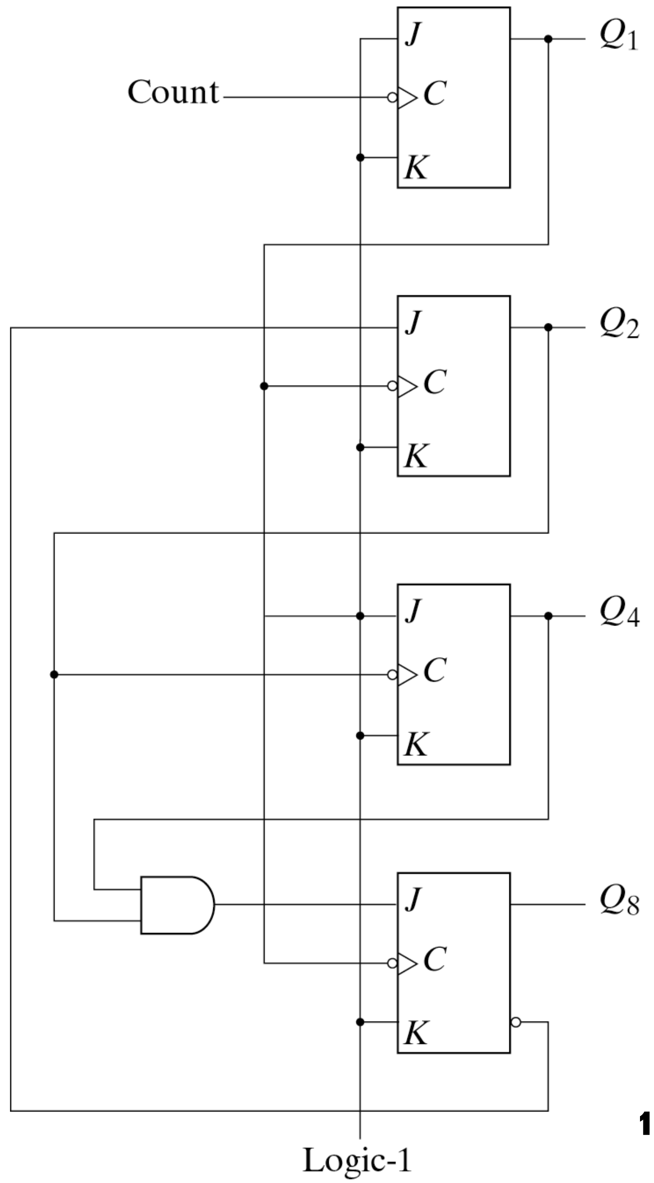
(b) From 70 to 92 ns

Simulation Output of HDL Example 6-4

# BCD Ripple Counter *(self-study)*

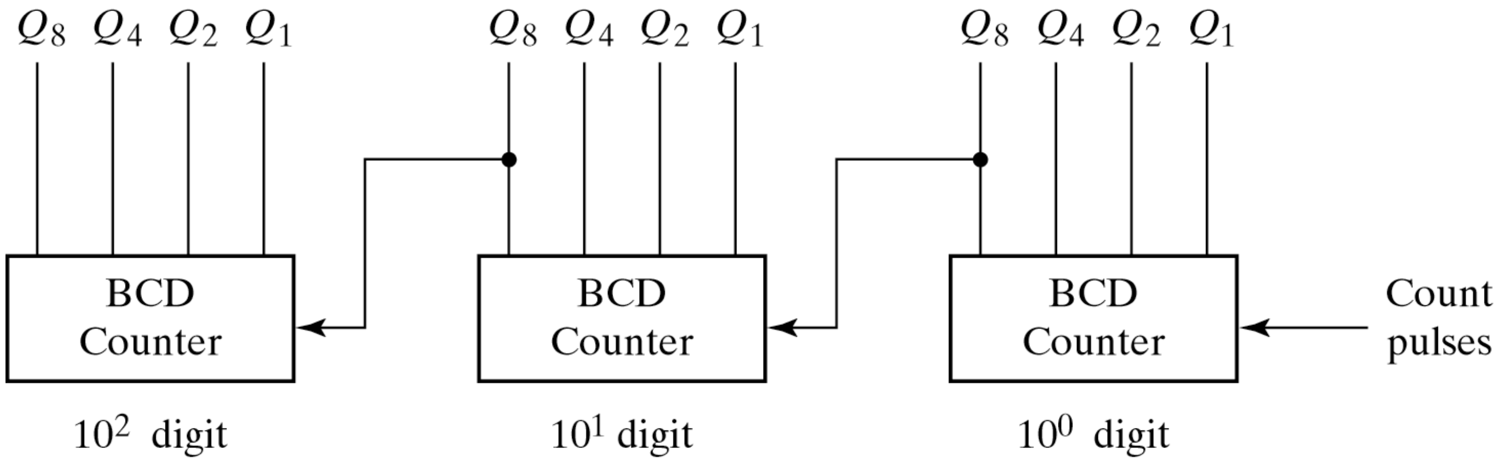


State Diagram of a Decimal BCD-Counter

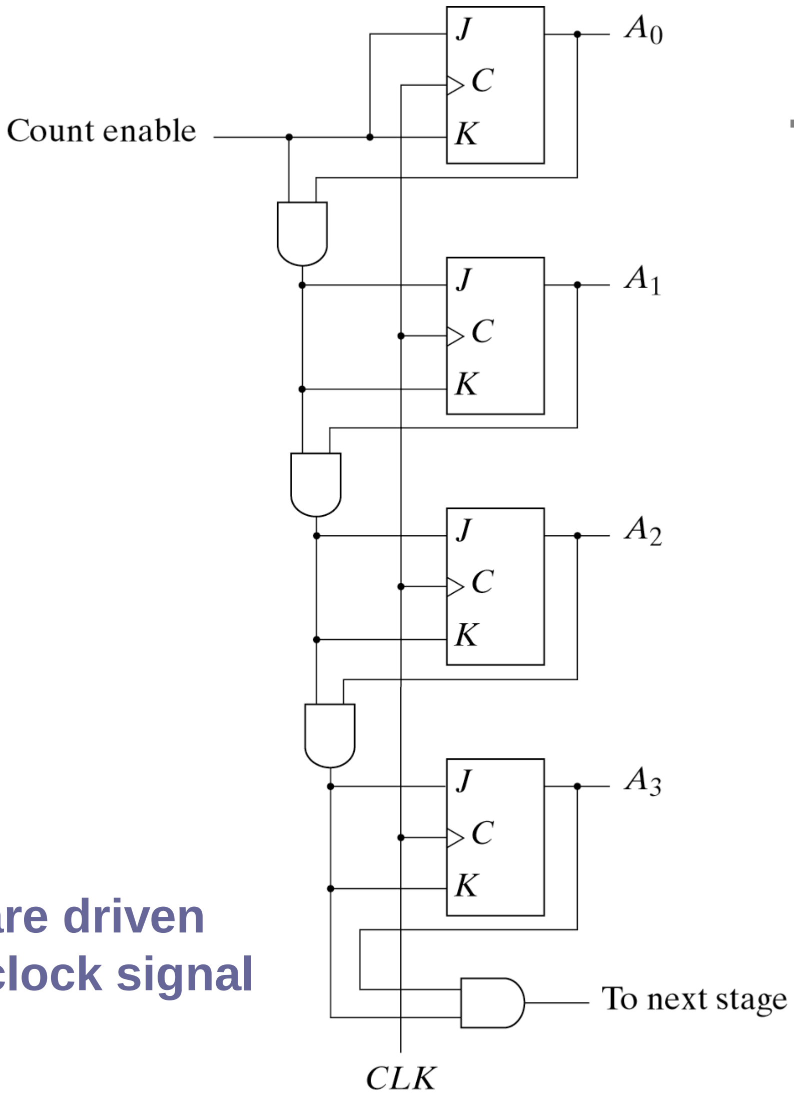


- $Q_1$ changes after each clock pulse
- As long as $Q_8$ is 0, $Q_2$ complements each time $Q_1$ goes from 1 to 0
- $Q_2$ remains at 0 when $Q_8$ is 1
- $Q_4$ complements each time $Q_2$ goes from 1 to 0
- $Q_8$ remains at 0 as long as $Q_2$ or $Q_4$ is 0
- When both $Q_2$ and $Q_4$ are 1, $Q_8$ complements when $Q_1$ goes from 1 to 0
- $Q_8$ is cleared on the next -ve transition of $Q_1$

# Three-Decade Decimal BCD Counter



# Synchronous Counters



4-Bit Synchronous Binary Counter

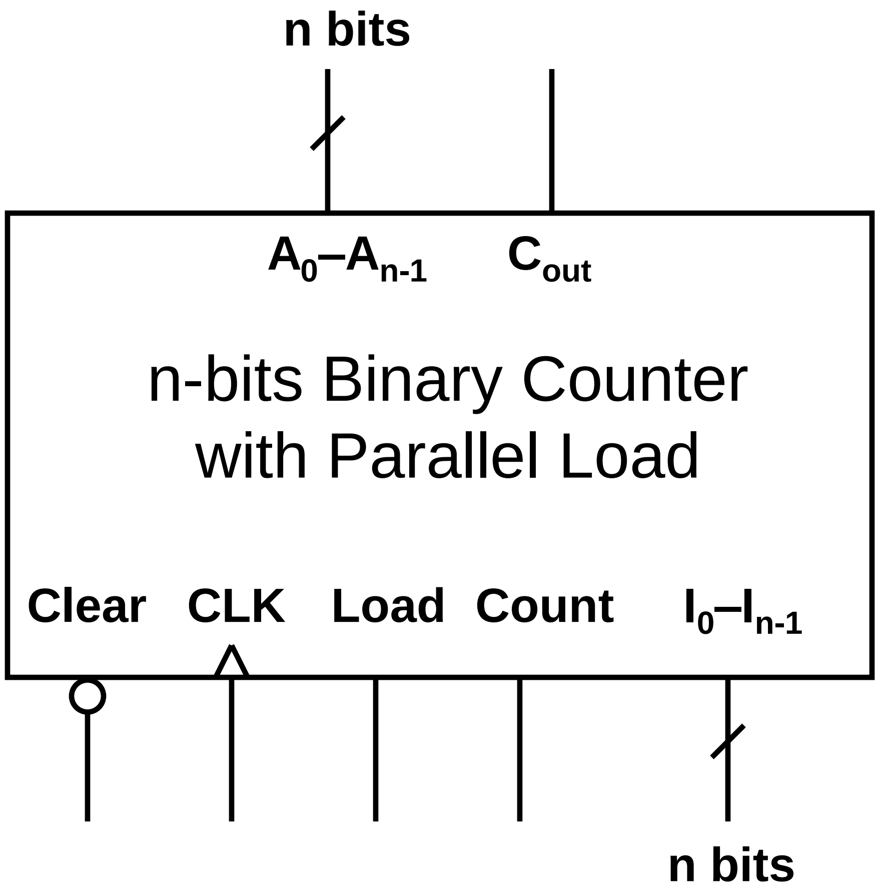
**All flip flops are driven by the same clock signal**

**Sequence of States**

| $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|
| 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 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

# Counter with Parallel Load

| Clear | CLK | Load | Count | Function |
|---|---|---|---|---|
| 0 | X | X | X | Clear to 0 |
| 1 | ↑ | 1 | X | Load inputs |
| 1 | ↑ | 0 | 1 | Up-Count |
| 1 | ↑ | 0 | 0 | No Change |

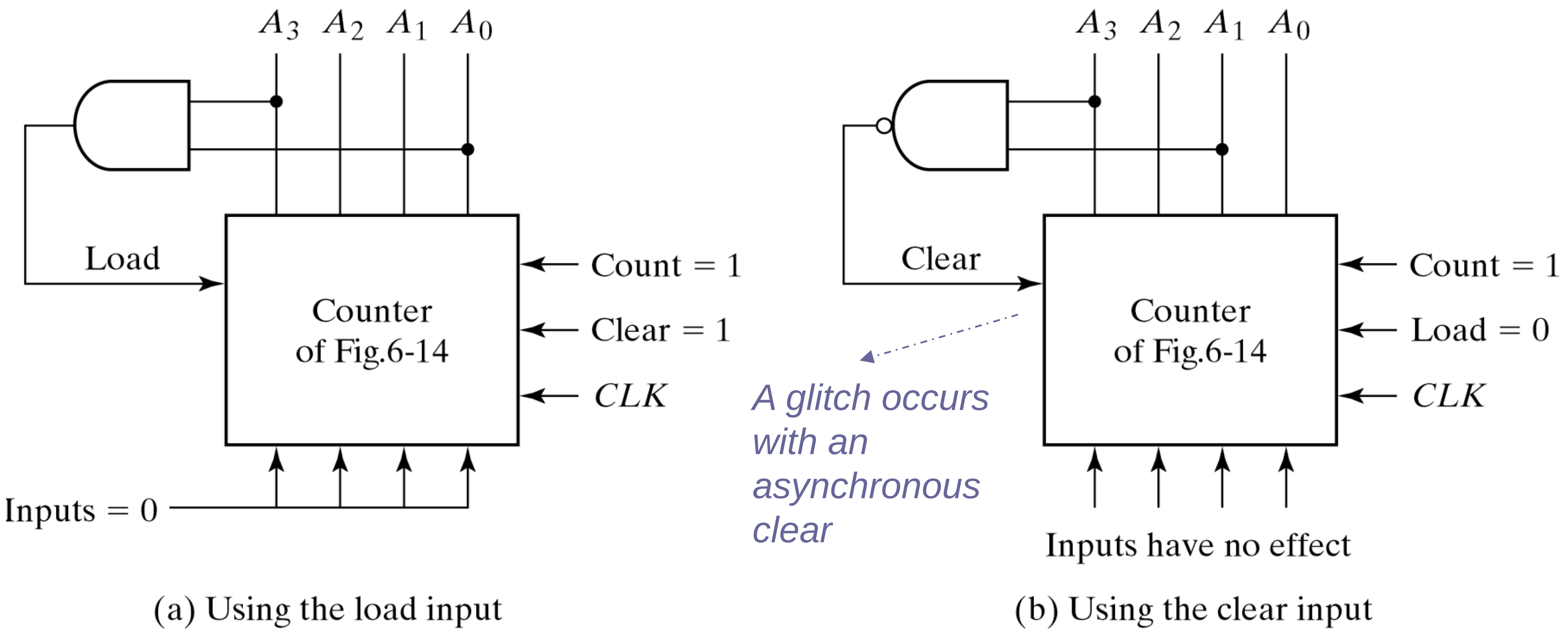


*Refer to the textbook for the circuit diagram*

## HDL for Binary Counter with Parallel Load

```
module counter (Count,Load,IN,CLK,Clr,A,CO);
  input Count,Load,CLK,Clr;
  input [3:0] IN;                 //Data input
  output CO;                       //Output carry
  output [3:0] A;                 //Data output
  reg [3:0] A;
  assign CO = Count & ~Load & (A == 4'b1111);
  always @ (posedge CLK or negedge Clr)
    if (~Clr) A = 4'b0000;
    else if (Load)  A = IN;
    else if (Count) A = A + 1'b1;
    else A = A;                 // no change, default condition
endmodule
```

# BCD Counter using Counter with Parallel Load



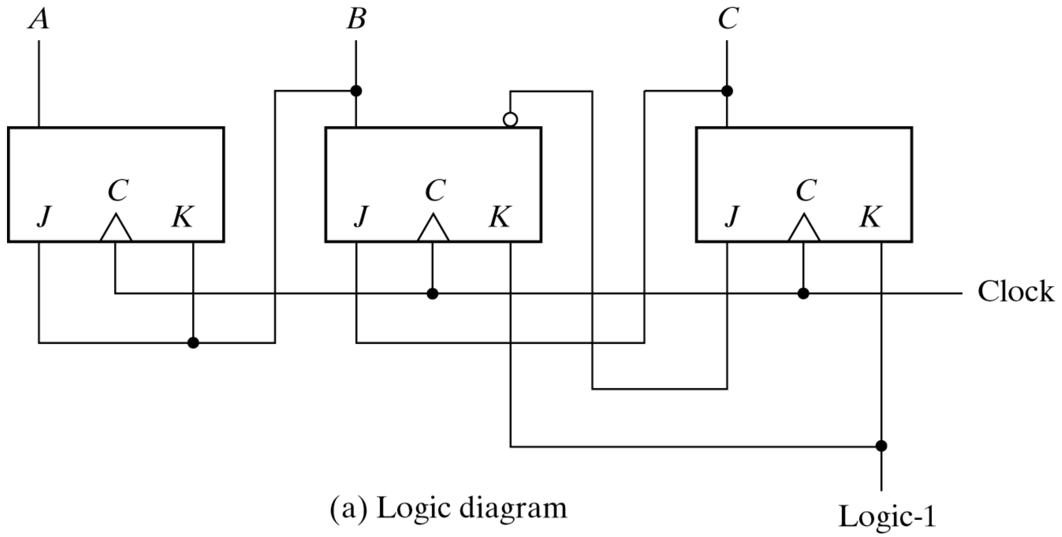
*A glitch occurs with an asynchronous clear*

Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

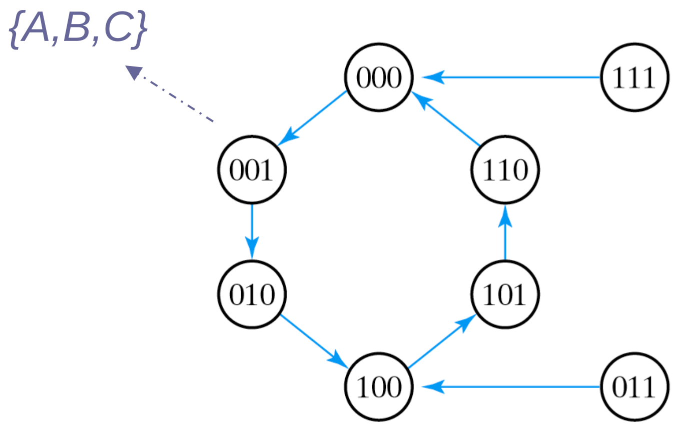
# Counter with Unused States

**Self Correcting Counter**

If it happens to be in one of the unused states, eventually reaches the normal count sequences after one or more clock pulses



(a) Logic diagram

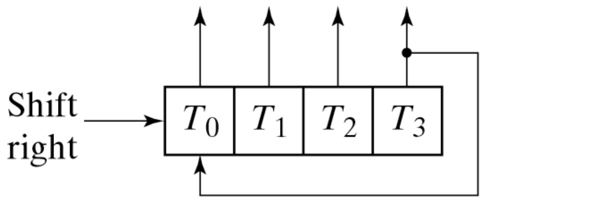


(b) State diagram

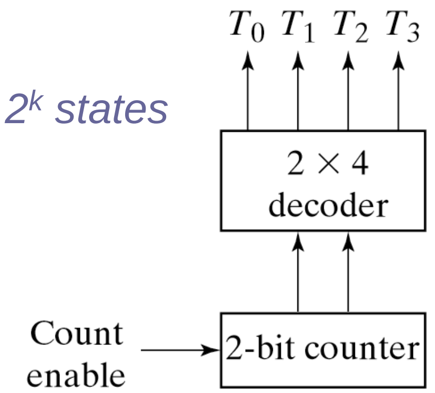
# Generating Timing Signals

*k FFs ⟶ k states*

*k FFs ⟶ 2^k states*

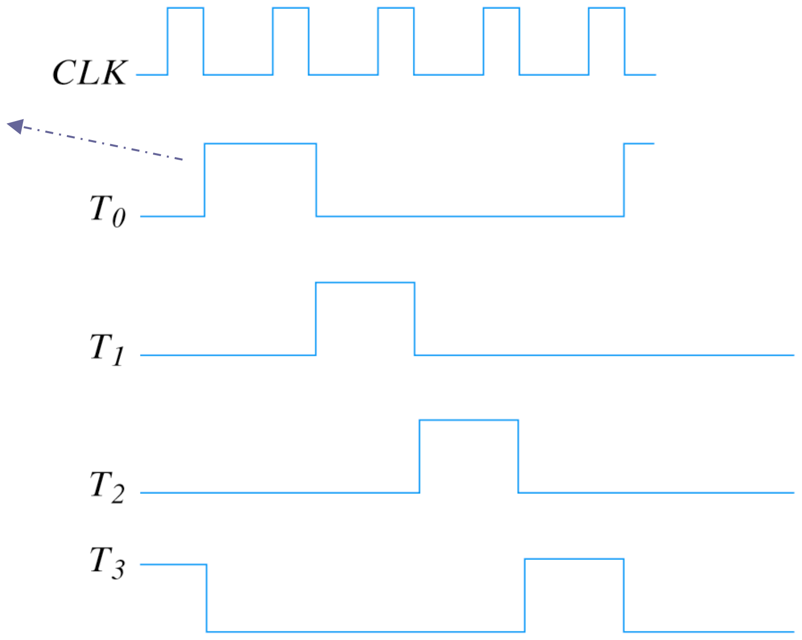


(a) Ring-counter (initial value = 1000)



(b) Counter and decoder

*Assuming –ve edge-triggering*

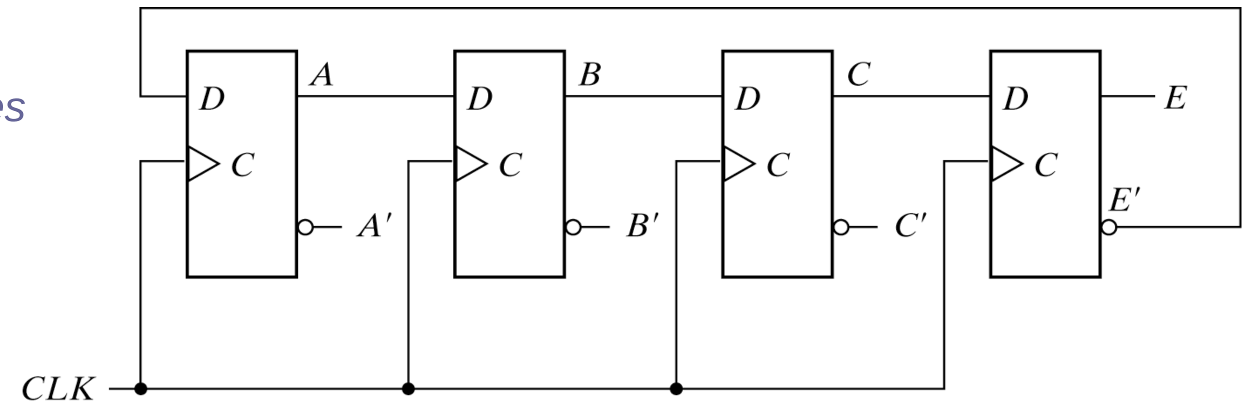


(c) Sequence of four timing signals

*Next lecture we will emphasize why would this be important*

# Johnson Counter

*k FFs ⟶ 2k states*



(a) Four-stage switch-tail ring counter

| Sequence number | Flip-flop outputs A | B | C | E | AND gate required for output | |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | $A'E'$ | $\dashrightarrow T_0$ |
| 2 | 1 | 0 | 0 | 0 | $AB'$ | |
| 3 | 1 | 1 | 0 | 0 | $BC'$ | |
| 4 | 1 | 1 | 1 | 0 | $CE'$ | |
| 5 | 1 | 1 | 1 | 1 | $AE$ | |
| 6 | 0 | 1 | 1 | 1 | $A'B$ | |
| 7 | 0 | 0 | 1 | 1 | $B'C$ | |
| 8 | 0 | 0 | 0 | 1 | $C'E$ | $\dashrightarrow T_7$ |

(b) Count sequence and required decoding

Construction of a Johnson Counter

## Corresponding Chapter in Textbook

- Chapter 6 *(entire chapter)*

## References

- Digital Design, M. Morris, Mano
- http://bawankule.com/verilogfaq/files/jhld099401.pdf