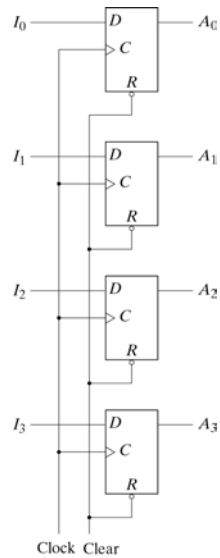


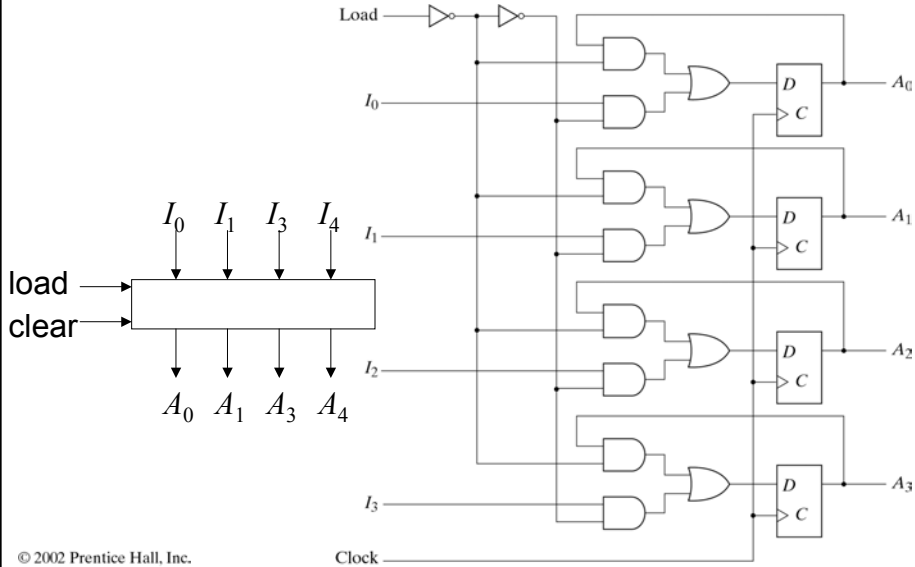
# 4-Bit Register



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 6-1 4-Bit Register

# 4-Bit Register with Separate Load



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 6-2 4-Bit Register with Parallel Load

# 4-Bit Shift Register with Serial Input

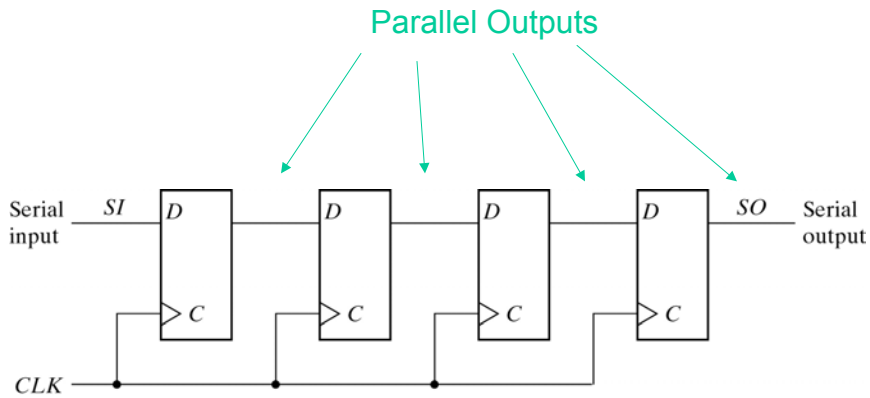
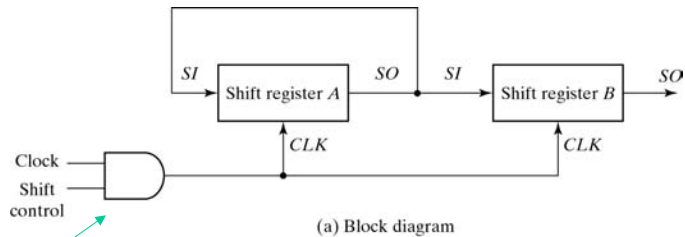


Fig. 6-3 4-Bit Shift Register

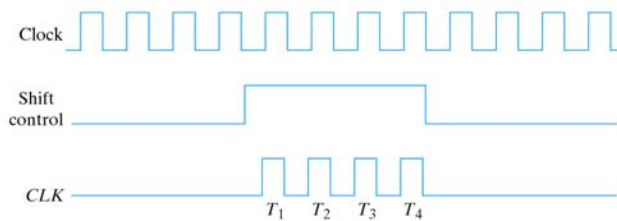
© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

# Serial Transfer from A to B



(a) Block diagram

Clock Gating

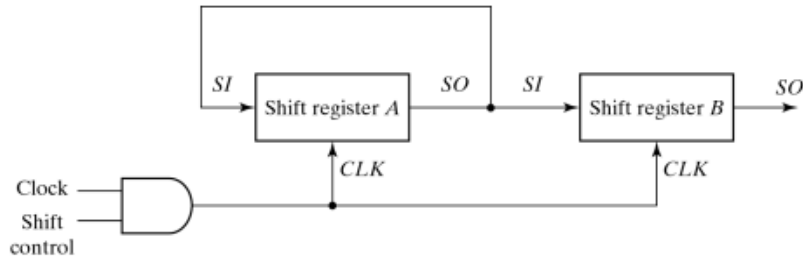


(b) Timing diagram

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 6-4 Serial Transfer from Register A to register B

## Serial Transfer from A to B



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

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

## Serial Addition Circuit

- Clear Regs
- Clock in Addend followed by augend  $2n$  clocks to load
- Clock the system  $n$  more times
- Sum is in Reg. A
- DFF holds the carry-out

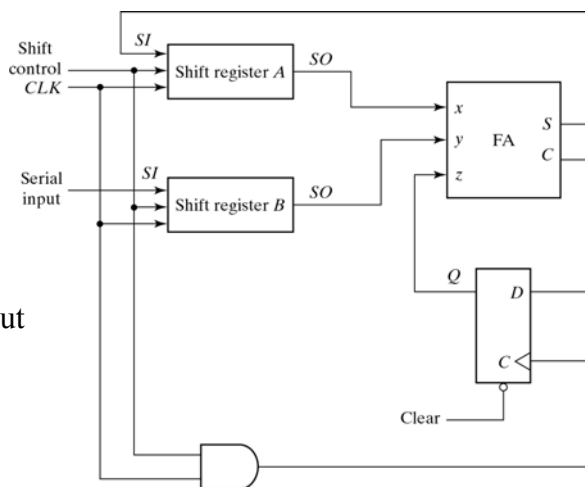


Fig. 6-5 Serial Adder

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

## Redesign Serial Adder with State Table

- $x$  and  $y$  are serial inputs (from a shift register)
- Present state is Carry-in, Next-state is Carry-out

Present State	Inputs		Next State	Output	Flip-Flop Inputs	
	$x$	$y$			$J_Q$	$K_Q$
$Q$	$x$	$y$	$Q$	$S$	$J_Q$	$K_Q$
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

$$J_Q = xy$$

$$K_Q = \bar{x}\bar{y} = \overline{x+y}$$

$$S = x \oplus y \oplus Q$$

## Redesign Serial Adder with State Table

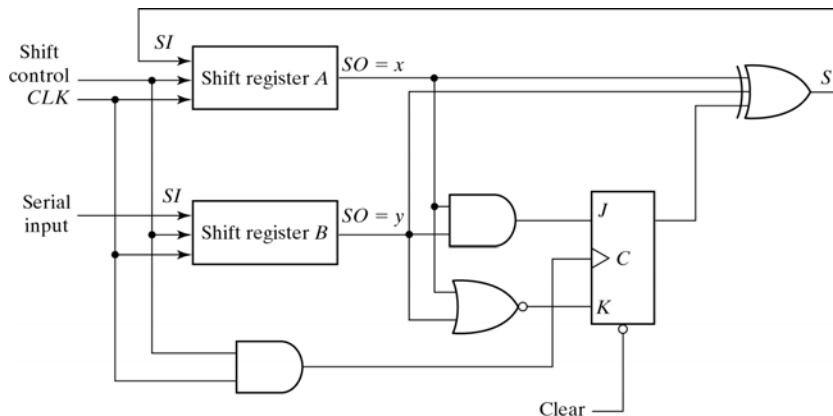
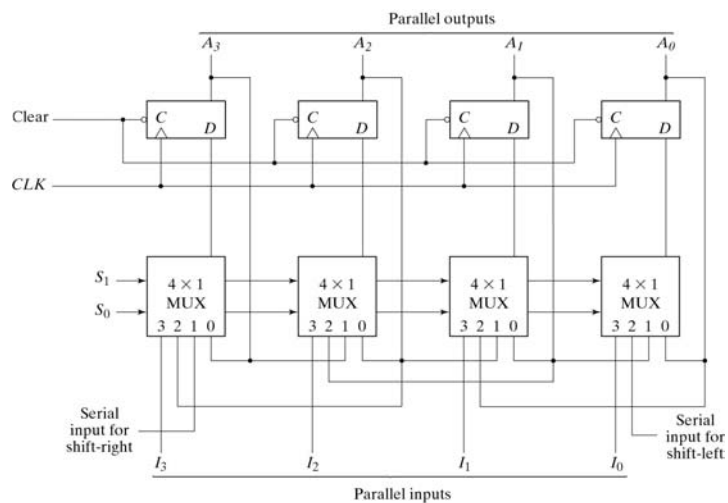


Fig. 6-6 Second form of Serial Adder

## Universal Shift Register

- clear Control to Set Register to 0
- clock Input to Synchronize Operations
- shift-right control
- shift left control
- serial input and output lines
- parallel-load control
- n parallel output and input lines
- control signal that leaves information in register unchanged during a clock edge

## Universal Shift Register

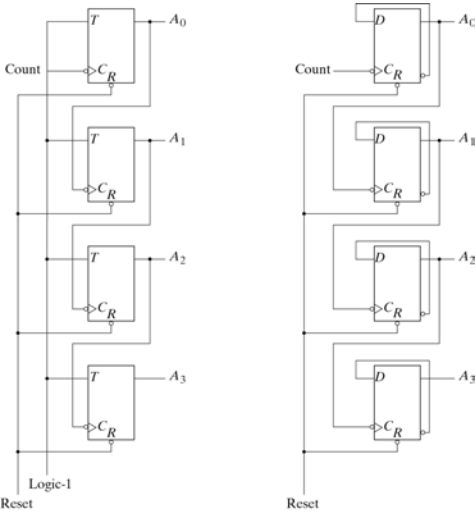


© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 6-7 4-Bit Universal Shift Register

# Ripple Counters

## Asynchronous



© 2002 Prentice Hall, Inc.  
 M. Morris Mano  
**DIGITAL DESIGN, 3e.**

(a) With T flip-flops

(b) With D flip-flops

Fig. 6-8 4-Bit Binary Ripple Counter

# BCD Counter State Diagram

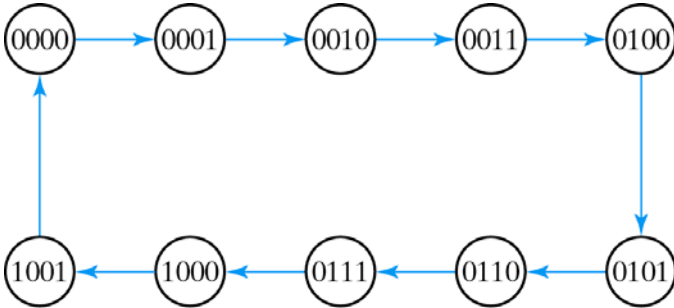


Fig. 6-9 State Diagram of a Decimal BCD-Counter

© 2002 Prentice Hall, Inc.  
 M. Morris Mano  
**DIGITAL DESIGN, 3e.**

# Ripple BCD Counter

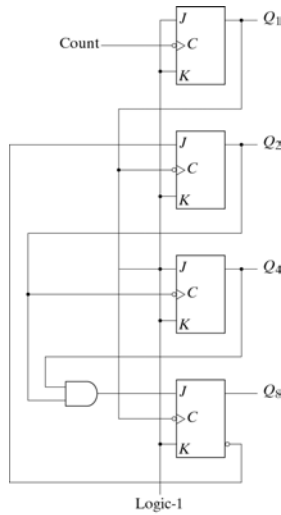


Fig. 6-10 BCD Ripple Counter

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

# Cascaded Ripple BCD Counter

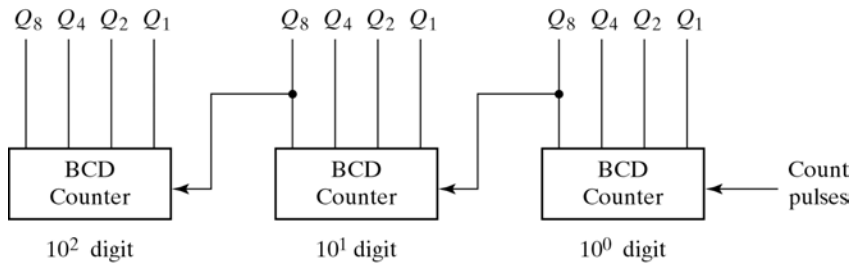
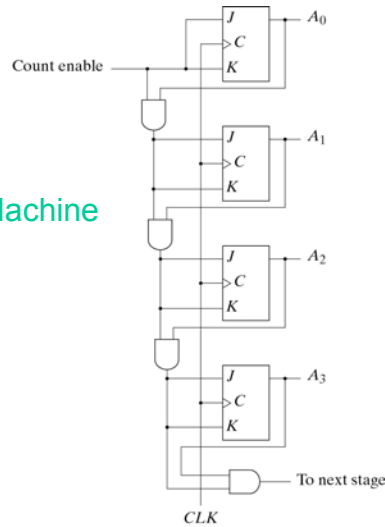


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

# Synchronous Binary Counter

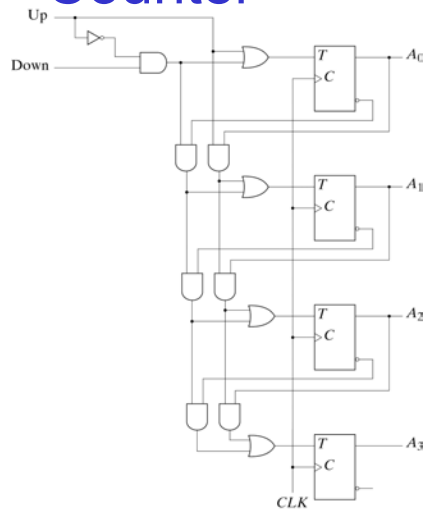
Use Normal Design Process for Moore Machine



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 6-12 4-Bit Synchronous Binary Counter

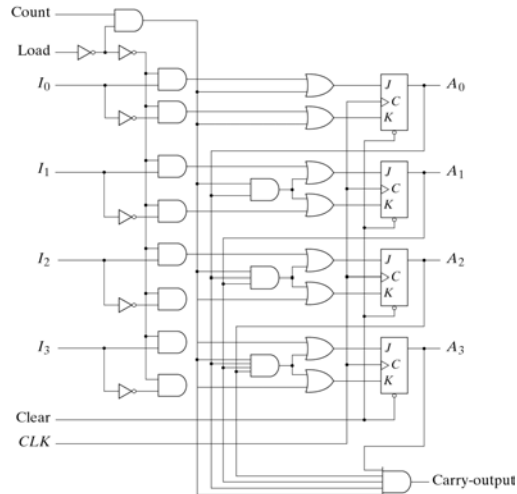
# Synchronous Up/Down Binary Counter



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 6-13 4-Bit Up-Down Binary Counter

# 4-bit Counter with Parallel Load



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 6-14 4-Bit Binary Counter with Parallel Load

# BCD Counters with Parallel Load

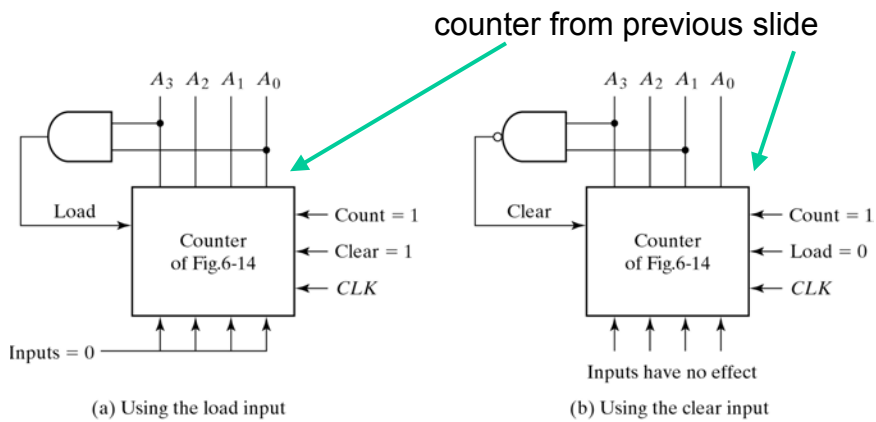
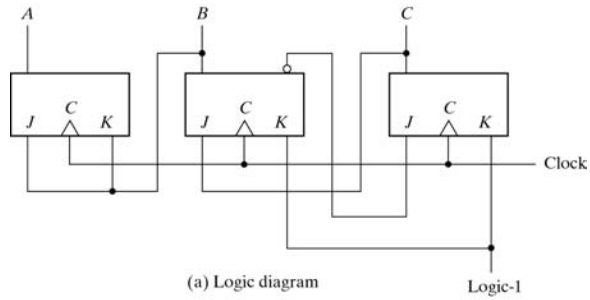


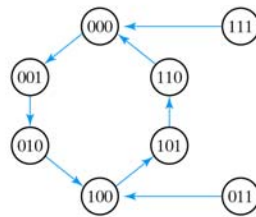
Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

# Lock-out Prevention



(a) Logic diagram

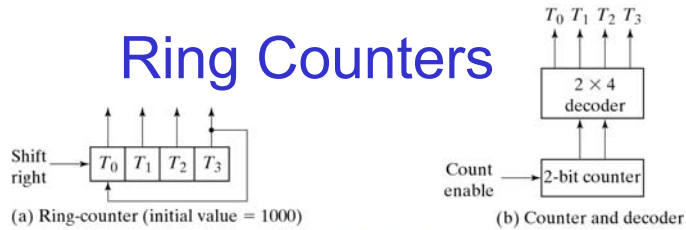


(b) State diagram

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

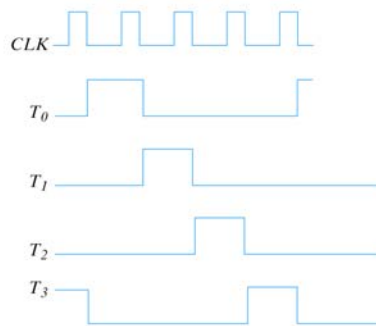
Fig. 6-16 Counter with Unused States

# Ring Counters



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

(b) Counter and decoder

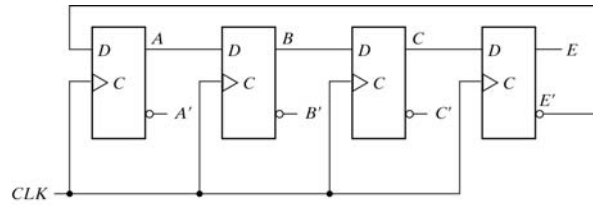


(c) Sequence of four timing signals

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 6-17 Generation of Timing Signals

# Johnson Counter



(a) Four-stage switch-tail ring counter

Sequence number	Flip-flop outputs				AND gate required for output
	A	B	C	E	
1	0	0	0	0	$A'E'$
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	$C'E$

(b) Count sequence and required decoding

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 6-18 Construction of a Johnson Counter

# Verilog for Universal Shift Register

```
//HDL Example 6-1
//-----
//Behavioral description of
//Universal shift register
// Fig. 6-7 and Table 6-3
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
```

## Verilog for Universal Shift Register Structural Description

```
//HDL Example 6-2
//-----
//Structural description of
//Universal shift register(see Fig.6-7)
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
```

## Verilog for Universal Shift Register Structural Description (cont)

```
//One stage of shift register
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
```

## Verilog for Synchronous Counter

```
//HDL Example 6-3
//-----
//Binary counter with parallel load
//See Figure 6-14 and Table 6-6
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
```

## Verilog for Ripple Counter

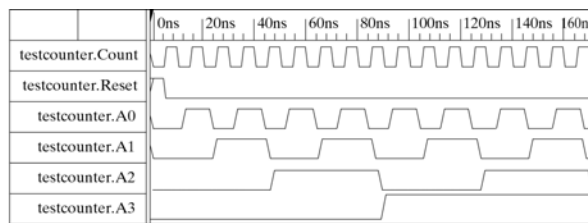
```
//HDL Example 6-4
//-----
//Ripple counter (See Fig. 6-8(b))
module ripplecounter (A0,A1,A2,A3,Count,Reset);
  output A0,A1,A2,A3;
  input Count,Reset;
  //Instantiate complementing flip-flop
  CF F0 (A0,Count,Reset);
  CF F1 (A1,A0,Reset);
  CF F2 (A2,A1,Reset);
  CF F3 (A3,A2,Reset);
endmodule

//Complementing flip-flop with delay
//Input to D flip-flop = Q'
module CF (Q,CLK,Reset);
  output Q;
  input CLK,Reset;
  reg Q;
  always @ (negedge CLK or posedge Reset)
    if (Reset) Q = 1'b0;
    else Q = #2 (~Q);    // Delay of 2 time units
endmodule
```

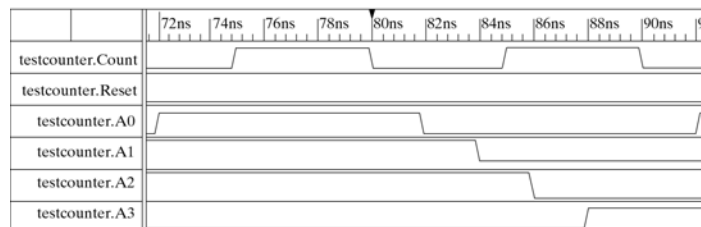
## Verilog for Ripple Counter Testbench

```
//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
```

## Ripple Counter Simulation Results



(a) From 0 to 170 ns



(b) From 70 to 92 ns