Chapter 6

Registers and Counters

Chapter 6

- A register is a group of flip-flops each id capable of storing one bit of information.
- A counter is a register that go through a predetermined sequence of states.

4-Bit Register

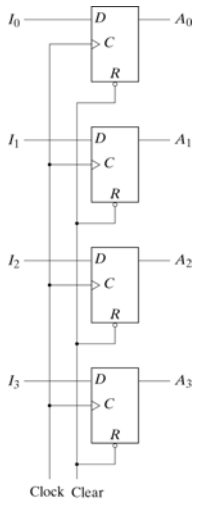


Fig. 6-1 4-Bit Register

- When the clock is applied to the input, it loads data.
- If we want to inhibit the clock, one way is to and it with a control signal.
- Inserting gates in the clock path produces un even clock delay to the different gates.
- To fully synchronize the system, we have to ensure that all flip-flops are triggered simultaneously

4-bit Register With Parallel Load

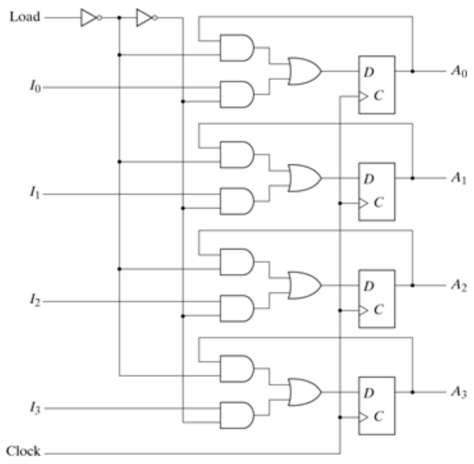


Fig. 6-2 4-Bit Register with Parallel Load

4-bit Shift Register

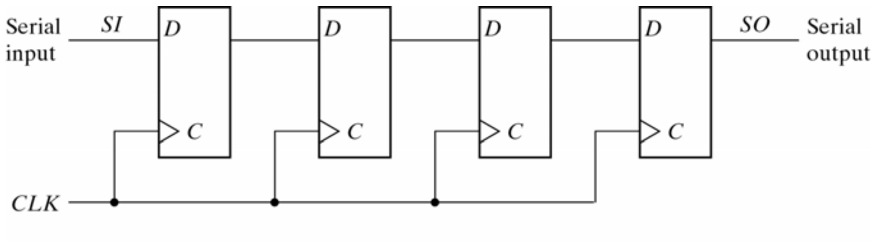
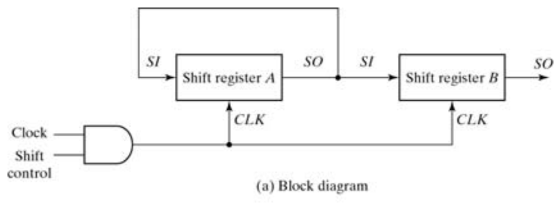
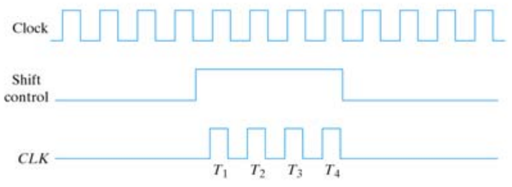


Fig. 6-3 4-Bit Shift Register

Serial Transfer



(a) Block diagram



(b) Timing diagram

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

Serial addition

Initially A holds the augend, B holds the addend. Carry bit is 0

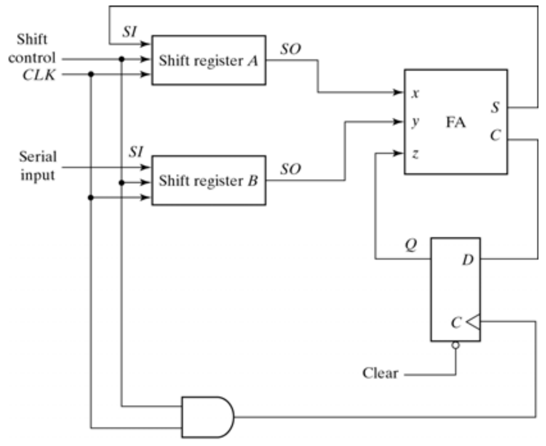


Fig. 6-5 Serial Adder

Universal Shift Register

S_1	S_2	OP
0	0	NOP
0	1	Sh R
1	0	Sh L
1	1	Par load

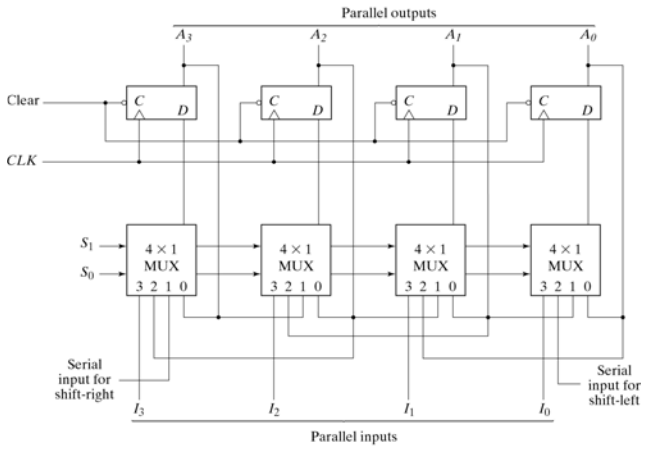


Fig. 6-7 4-Bit Universal Shift Register

Counters

- Ripple counters: the flip-flop output transition serves as a source for triggering other flip-flops (C input is triggered by flip-flop output rather than a common clock).
- Asynchronous counters: The CLK input of all flip-flops receive common clock. Transitions are triggered by combinatorial logic of other flip-flop outputs

Binary Ripple Counters

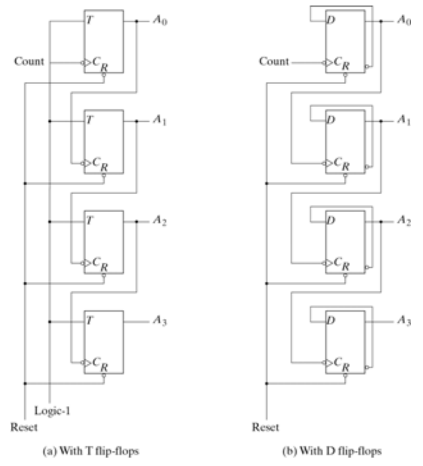


Fig. 6-8 4-Bit Binary Ripple Counter

BCD Ripple Counter

$$\begin{tikzcd}
0000 \arrow[r] & 0001 \arrow[r] & 0010 \arrow[r] & 0011 \arrow[r] & 0100 \arrow[d] \\
1001 \arrow[u] & 1000 \arrow[l] & 0111 \arrow[l] & 0110 \arrow[l] & 0101 \arrow[l]
\end{tikzcd}$$

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

BCD Ripple Counter

Q_1 changes state with every clock pulse

Q_2 complements every time Q1 goes from 1 to 0 as long as Q_8=0, when Q_8 =1, Q2 remains at 0

Q_4 complements every time Q_2 goes from 1->0

Q_8 remains at 0 as long as Q_2 or Q_4 is 0. When both Q_2 and Q4 is 1, Q_8 complements when Q1 goes from 1->0. Q_8 is cleared on the next transition of Q_1

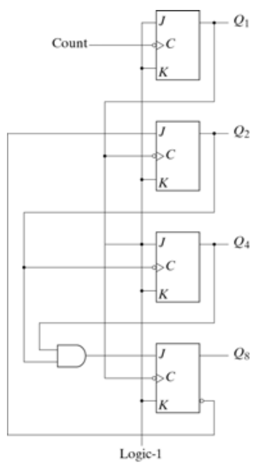


Fig. 6-10 BCD Ripple Counter

Q_8 Q_4 Q_2 Q_1
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

Three Decades Decimal BCD Counter

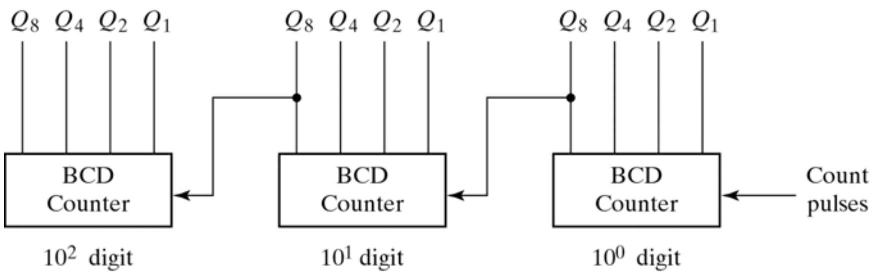


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

The input to the second and third stage comes from Q_8 to the previous stage. When Q_8 goes from 1 o zero, that is if Q_8 goes to 0, it triggers the count in the higher stage while its own count goes to zero

Synchronous Counters

•Clock pulses are applied to all flip-flops

•The least significant bit is complemented every clock cycle.

•The flip-flop in any position is complemented when all the bits in the lower significant positions are equal to 1.

•The flip-flops trigger on the positive, the polarity of the clock is not essential here as it was in the ripple counter case.

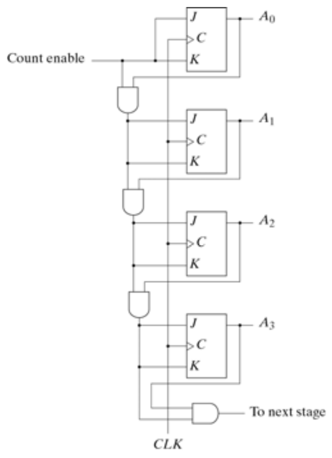


Fig. 6-12 4-Bit Synchronous Binary Counter

Up-Down Binary Counter

- •The bit in the least significant bit is complemented every clock cycle.
- •For count down, a bit in any position is complemented if all the bits in the lower positions are 0's
- •**BCD counters could be implemented using the techniques we learnt in the previous chapter**

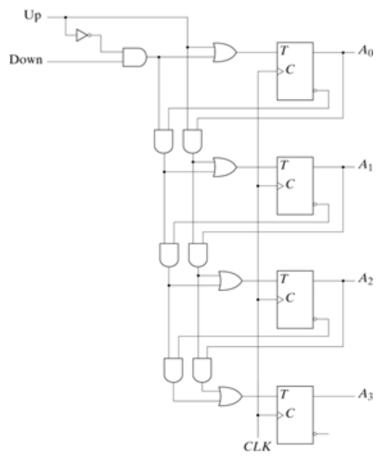
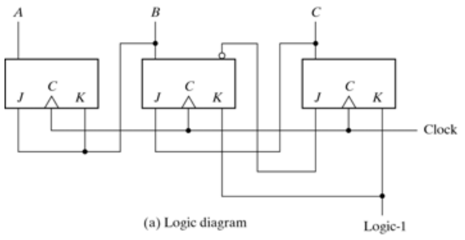


Fig. 6-13 4-Bit Up-Down Binary Counter

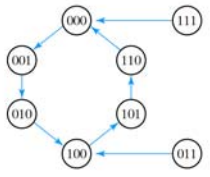
Counters with unused states

Analyze the circuit to know what will be the effect of the system being in one of the unused states due to error.

If the system goes eventually to the correct counting sequence, it is a **self correcting counter**



(a) Logic diagram



(b) State diagram

Fig. 6-16 Counter with Unused States

Ring Counter

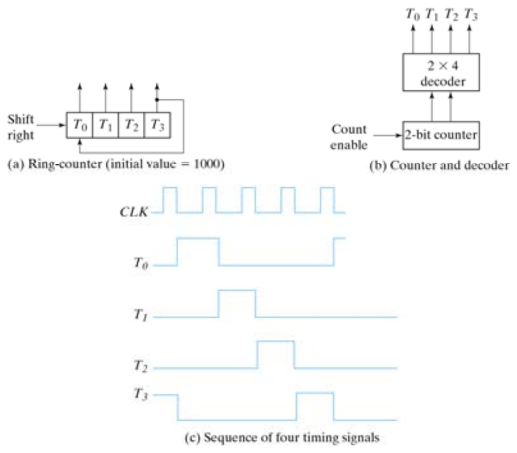
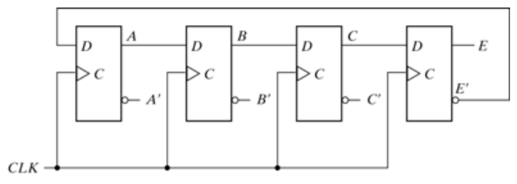
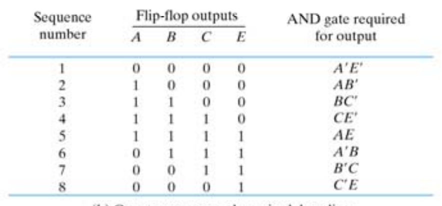


Fig. 6-17 Generation of Timing Signals

Johnson Counter



(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	$A'E'$
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

Fig. 6-18 Construction of a Johnson Counter

Example

```
//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
```

FALL 2011 CSE3201

Example

- //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

- //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

FALL 2011 CSE3201

Example

```
•  //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
```