Chapter 6 Registers and Counter

- The flip-flops are essential component in clocked sequential circuits.

- Circuits that include flip-flops are usually classified by the function they perform. Two such circuits are registers and counters.

- An $n$-bit register consists of a group of $n$ flip-flops capable of storing $n$ bits of binary information.
6-1 Registers

- In its broadest definition, a register consists of a group of flip-flops and gates that effect their transition.
  - The flip-flops hold the binary information.
  - The gates determine how the information is transferred into the register.
- Counters are a special type of register.
- A counter goes through a predetermined sequence of states.
6-1 Registers

- Fig 6-1 shows a register constructed with four D-type flipflops.
- “Clock” triggers all flipflops on the positive edge of each pulse.
- “Clear” is useful for clearing the register to all 0’s prior to its clocked operation.
Register with Parallel Load

- A clock edge applied to the C inputs of the register of Fig. 6-1 will load all four inputs in parallel.

- For synchronism, it is advisable to control the operation of the register with the D inputs rather than controlling the clock in the C inputs of the flip-flops.

- A 4-bit register with a load control input that is directed through gates and into the D inputs of the flip-flops is shown in Fig. 6-2.
Register with Parallel Load

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

- When the load input is 1, the data in the four inputs are transferred into the register with next positive edge of the clock.
- When the load input is 0, the outputs of the flip-flops are connected to their respective inputs.
- The feedback connection from output to input is necessary because the D flip-flops does not have a “no change” condition.
A register capable of shifting its binary information in one or both directions is called a shift register.

All flip-flops receive common clock pulses, which activate the shift from one stage to the next.

The simplest possible shift register is one that uses only flip-flops, as shown in Fig. 6-3.
Shift Registers

Fig. 6-3 4-Bit Shift Register
Shift Registers

- Each clock pulse shifts the contents of the register one bit position to the 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.
Serial Transfer

- A digital system is said to operate in a serial mode when information is transferred and manipulated one bit at a time.
- This in contrast to parallel transfer where all the bits of the register are transferred at the same time.
- The serial transfer is done with shift registers, as shown in the block diagram of Fig. 6-4(a).
Serial Transfer

(a) Block diagram
Serial Transfer

- To prevent the loss of information stored in the source register, the information in register A is made to circulate by connecting the serial output to its serial input.

- The shift control input determines when and how many times the registers are shifted. This is done with an AND gate that allows clock pulses to pass into the CLK terminals only when the shift control is active. [Fig. 6-4(a)].
Serial Transfer

(b) Timing diagram

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

- The shift control signal is synchronized with the clock and changes value just after the negative edge of the clock.

- Each rising edge of the pulse causes a shift in both registers. The fourth pulse changes the shift control to 0 and the shift registers are disabled.
# Serial Transfer

## Table 6-1

Serial-Transfer Example

<table>
<thead>
<tr>
<th>Timing Pulse</th>
<th>Shift Register A</th>
<th>Shift Register B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial value</td>
<td>1 0 1 1</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>After T1</td>
<td>1 1 0 1</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>After T2</td>
<td>1 1 1 0</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>After T3</td>
<td>0 1 1 1</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>After T4</td>
<td>1 0 1 1</td>
<td>1 0 1 1</td>
</tr>
</tbody>
</table>
Serial Transfer

- In the parallel mode, information is available from all bits can be transferred simultaneously during one clock pulse.
- In the serial mode, the registers have a single serial input and a single serial output. The information is transferred one bit at a time while the registers are shifted in the same direction.
Serial Addition

- Operations in digital computers are usually done in parallel because this is a faster mode of operation.
- Serial operations are slower, but have the advantage of requiring less equipment.
- The two binary numbers to be added serially are stored in two shift registers.
- Bits are added one pair at a time through a single full adder. [Fig. 6-5]
Fig. 6-5  Serial Adder
Serial Addition

- By shifting the sum into A while the bits of A are shifted out, it is possible to use one register for storing both the augend and sum bits.
- The carry out of the full adder is transferred to a D flip-flop.
- The output of the D flip-flop is then used as carry input for the next pair of significant bits.
Serial Addition

- To show that serial operations can be designed by means of sequential circuit procedure, we will redesign the serial adder using a state table.

- The serial outputs from registers are designated by $x$ and $y$.

- The sequential circuit proper has two inputs, $x$ and $y$, that provide a pair of significant bits, an output $S$ that generates the sum bit, and flip-flop $Q$ for storing the carry. [Table. 6-2]
## Serial Addition

Table 6-2
State Table for serial Adder

<table>
<thead>
<tr>
<th>Present State</th>
<th>Inputs</th>
<th>Next State</th>
<th>Output</th>
<th>Flip-Flop Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>X</td>
<td>y</td>
<td>Q</td>
<td>S</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Serial Addition

- The two flip-flop input equations and the output equation can be simplified by means of map to obtain
  - $JQ = xy$
  - $KQ = x'y' = (x+y)'$
  - $S = x \oplus y \oplus Q$

- The circuit diagram is shown in [Fig. 6-6]
Serial Addition

Fig. 6-6 Second form of Serial Adder
Universal Shift Register

- A *clear* control to clear the register to 0.
- A *clock* input to synchronize the operations.
- A *shift-right* control to enable the shift operation and the *serial input* and *output* lines associated with the shift right.
- A *shift-left* control to enable the shift operation and the *serial input* and *output* lines associated with the shift left.
Universal Shift Register

- A parallel-load control to enable a parallel transfer and the $n$ input lines associated with the parallel transfer.
- $n$ parallel output lines.
- A control state that leaves the information in the register unchanged in the presence of the clock.
- If the register has both shifts and parallel load capabilities, it is referred to as a universal shift register.
Fig. 6-7 4-Bit Universal Shift Register
# Universal Shift Register

Table 6-3
Function Table for the Register of Fig. 6-7

<table>
<thead>
<tr>
<th>Mode Control</th>
<th>Register Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S&lt;sub&gt;1&lt;/sub&gt;</td>
<td>S&lt;sub&gt;0&lt;/sub&gt;</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Universal Shift Register

- Shift registers are often used to interface digital system situated remotely from each other.
- If the distance is far, it will be expensive to use $n$ lines to transmit the $n$ bits in parallel.
- Transmitter performs a parallel-to-serial conversion of data and the receiver does a serial-to-parallel conversion.
A register that goes through a prescribed sequence of states upon the application of input pulse is called a counter.

A counter that follows the binary number sequence is called a binary counter.

Counters are available in two categories
  - Ripple counters
  - Synchronous counters
Binary Ripple Counter

- The output of each flip-flop is connected to the C input of the next flip-flop in sequence.
- The flip-flop holding the last significant bit receives the incoming count pulse.
- A complementing flip-flop can be obtained from:
  - JK flip-flop with the J and K inputs tied together.
  - T flip-flop
  - D flip-flop with the complement output connected to the D input. [Fig. 6-8]
Fig. 6-8  4-Bit Binary Ripple Counter

(a) With T flip-flops

(b) With D flip-flops
Binary Ripple Counter

Table 6-3
Function Table for the Register of Fig. 6-7

<table>
<thead>
<tr>
<th>A₃</th>
<th>A₂</th>
<th>A₁</th>
<th>A₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
BCD Ripple Counter

- A decimal counter follows a sequence of ten states and returns to 0 after the count of 9.
- This is similar to a binary counter, except that the state after 1001 is 0000.
- The operation of the counter can be explained by a list of conditions for flip-flop transitions.
Fig. 6-9 State Diagram of a Decimal BCD-Counter
BCD Ripple Counter

The four outputs are designated by the letter symbol Q with a numeric subscript equal to the binary weight of the corresponding bit in the BCD code.
BCD Ripple Counter

- The BCD counter of [Fig. 6-9] is a decade counter.
- To count in decimal from 0 to 999, we need a three-decade counter. [Fig. 6-11]
- Multiple decade counters can be constructed by connecting BCD counters in cascade, one for each decade.
BCD Ripple Counter

**Fig. 6-11** Block Diagram of a Three-Decade Decimal BCD Counter
6-4 Synchronous Counters

- Synchronous counters are different from ripple counters in that clock pulses are applied to the inputs of all flip-flops.
- A common clock triggers all flip-flops simultaneously rather than one at a time in succession as in a ripple counter.
The design of a synchronous binary counter is so simple that it is not need to go through a sequential logic design process.

Synchronous binary counters have a regular pattern and can be constructed with complementing flip-flop and gates.

Fig. 6-12 4-Bit Synchronous Binary Counter
Up-Down Binary Counter

- The two operations can be combined in one circuit to form a counter capable of counting up or down.
- It has an up control input and down control input.
BCD Counter

- Because of the return to 0 after a count of 9, a BCD counter does not have a regular pattern as in a straight binary count.

- To derive the circuit of a BCD synchronous counter, it is necessary to go through a sequential circuit design procedure.
# BCD Counter

## Table 6-5
State Table for BCD Counter

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
<th>Flip-Flop inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q8  Q4  Q2  Q1</td>
<td>Q8  Q4  Q2  Q1</td>
<td>Y</td>
<td>TQ8  TQ4  TQ2  TQ1</td>
</tr>
<tr>
<td>0  0  0  0</td>
<td>0  0  0  1</td>
<td>0</td>
<td>0  0  0  1</td>
</tr>
<tr>
<td>0  0  0  1</td>
<td>0  0  1  0</td>
<td>0</td>
<td>0  0  1  1</td>
</tr>
<tr>
<td>0  0  1  0</td>
<td>0  0  1  1</td>
<td>0</td>
<td>0  0  0  1</td>
</tr>
<tr>
<td>0  0  1  1</td>
<td>0  1  0  0</td>
<td>0</td>
<td>0  1  1  1</td>
</tr>
<tr>
<td>0  1  0  0</td>
<td>0  1  0  1</td>
<td>0</td>
<td>0  0  0  1</td>
</tr>
<tr>
<td>0  1  0  1</td>
<td>0  1  1  0</td>
<td>0</td>
<td>0  0  1  1</td>
</tr>
<tr>
<td>0  1  1  0</td>
<td>0  1  1  1</td>
<td>0</td>
<td>0  0  0  1</td>
</tr>
<tr>
<td>0  1  1  1</td>
<td>1  0  0  0</td>
<td>0</td>
<td>1  1  1  1</td>
</tr>
<tr>
<td>1  0  0  0</td>
<td>1  0  0  1</td>
<td>0</td>
<td>0  0  0  1</td>
</tr>
<tr>
<td>1  0  0  1</td>
<td>0  0  0  0</td>
<td>1</td>
<td>0  0  0  1</td>
</tr>
</tbody>
</table>
BCD Counter

- The flip flop input equations can be simplified by means of maps. The simplified functions are
  - $T_{Q1} = 1$
  - $T_{Q2} = Q_8'Q_1$
  - $T_{Q4} = Q_2Q_1$
  - $T_{Q8} = Q_8Q_1 + Q_4Q_2Q_1$
  - $y = Q_8Q_1$

- The circuit can be easily drawn with four T flip-flops, five AND gates, and one OR gate.
Binary Counter with Parallel Load

- Counters employed in digital systems quite often require a parallel load capability for transferring an initial binary number into the counter prior to count operation.
- The input load control when equal to 1 disables the count operation and causes a transfer of data from the four data inputs into the four flip-flops [Fig. 6-14]
Fig. 6-14  4-Bit Binary Counter with Parallel Load
## Binary Counter with Parallel Load

### Table 6-6
Function Table for the Counter of Fig. 6-14

<table>
<thead>
<tr>
<th>Clear</th>
<th>CLK</th>
<th>Load</th>
<th>Count</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Clear to 0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>X</td>
<td>Load inputs</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>1</td>
<td>Count next binary state</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
<td>No change</td>
</tr>
</tbody>
</table>
Binary Counter with Parallel Load

- A counter with parallel load can be used to generate any desire count sequence.
- [Fig.6-15] shows two ways in which a counter with parallel load is used to generate the BCD count.
Binary Counter with Parallel Load

(a) Using the load input
(b) Using the clear input

Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load
6-5 Other Counters

- Counters can be designed to generate any desired sequence of states.
- Counters are used to generate timing signals to control the sequence of operations in a digital system.
- Counters can be constructed also by means of shift registers.
Counter with Unused States

- Once the circuit is designed and constructed, outside interference may cause the circuit to enter one of the unused state.

- If the unused states are treated as don’t-care conditions, then once the circuit is designed, it must be investigated to determine the effect of the unused states.

- The next state from an unused state can be determined from the analysis of the circuit after it is design.
## Counter with Unused States

### Table 6-7

State Table for Counter

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next state</th>
<th>Flip-Flop Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Counter with Unused States

- The count has a repeated sequence of six states.
- The simplified equations are:
  - $J_A = B$  $K_A = B$
  - $J_B = C$  $K_B = 1$
  - $J_C = B'$  $K_C = 1$
- The logic diagram and state diagram is shown in [Fig. 6-16]
Fig. 6-16 Counter with Unused States
Ring Counter

- A ring counter is a circular shift register with only one flip-flop being set at any particular time, all others are cleared.

- The single bit is shifted from one flip-flop to the next to produce the sequence of timing signals. [Fig. 6-17(a)] [Fig. 6-17(c)]

- The decoder shown in [Fig. 6-17(b)] decodes the four states of the counter and generates the required sequence of timing signals.
(a) Ring-counter (initial value = 1000)

(b) Counter and decoder

(c) Sequence of four timing signals

Fig. 6-17 Generation of Timing Signals
Johnson Counter

- Generate the timing signals with a combination of a shift register and a decoder, which is called a *Johnson counter*.

- The number of states can be double if the shift register is connect as a *switch-tail* ring counter. [Fig. 6-18(a)]

- Starting from a cleared state, the switch-tail ring counter goes through a sequence of eight states, as shown in [Fig. 6-18(b)].
(a) Four-stage switch-tail ring counter

<table>
<thead>
<tr>
<th>Sequence number</th>
<th>Flip-flop outputs</th>
<th>AND gate required for output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Count sequence and required decoding

Fig. 6-18 Construction of a Johnson Counter
Johnson Counter

- A Johnson counter is a k-bit switch-tail ring counter with $2^k$ decoding gates to provide outputs for $2^k$ timing signals.
- The decoding of a k-bit switch-tail ring counter to obtain $2^k$ timing signals follows a regular pattern.
- Johnson counters can be constructed for any number of timing sequences.
6-6 HDL for Registers and Counters

- Registers and counters can be describe in HDL at either the behavioral or the structural level.
- The various components are instantiated to form a hierarchical description of the design similar to a representation of a logic diagram.
// 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' // No change
else
  case ({s1, s0}) // No change
    2’b00: A = A; // Shift right
    2’b01: A = {rtin, A[3:1]}; // Shift left
    2’b10: A = {A[2:0], lfin}; // Parallel load input
    2’b11: A = Pin;
  endcase
endmodule
// 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
Shift Register

// 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
Synchronous Counter

// 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
Ripple Counter

// 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

KtuQbank
// 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
Fig. 6-19  Simulation Output of HDL Example 6-4