University of Wisconsin - Madison

ECE/Comp Sci 352 Digital Systems Fundamentals

Charles R. Kime Section 2 – Fall 2001

Logic and Computer Design Fundamentals

Lecture 5 – Registers & Counters – Part 2

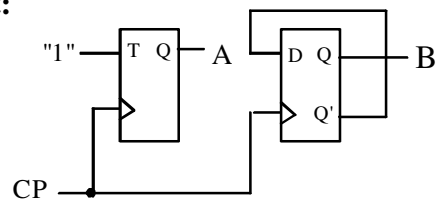
Charles Kime

Counters

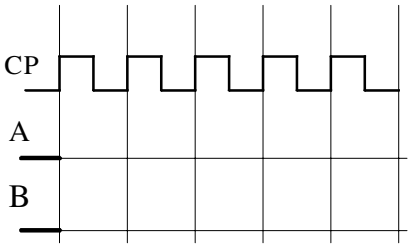
- **Counters are sequential circuits which "count" through a specific state sequence. They can count up, count down, or count through other fixed sequences. Two distinct types are in common usage:**
- **Ripple Counters**
  - **Clocks are connected to flip-flop outputs, thus not truly synchronous**
  - **Outputs are "delayed" for higher bits.**
  - **Resurgent because of low power consumption**
- **Synchronous Counters**
  - **Clocks are directly connected to the flip-flops.**
  - **Logic is used to implement the desired state sequencing.**

Counter Basics: Divide by 2

Consider the following circuit:



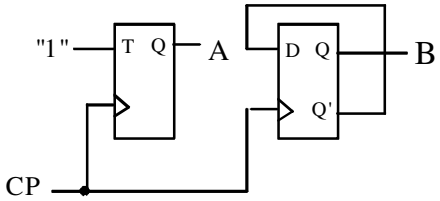
Draw Waveform A and B

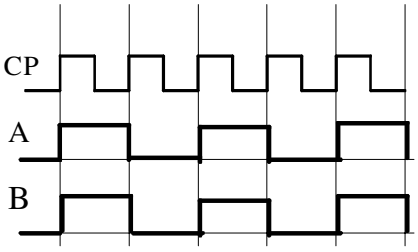


Divide Clock Frequency by 2

The waveforms look like a new clock with twice the period of CP (half the frequency).

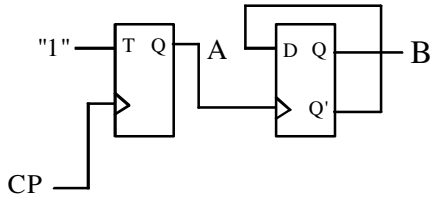
The flip-flops are said to "divide-by-2" since the frequency of the output waveform is 1/2 the frequency of clock CP.

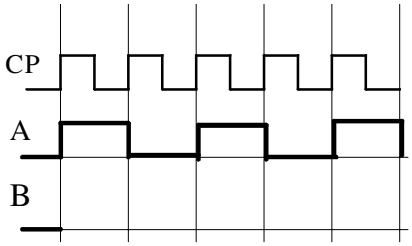




Ripple Counter

What happens now?
(note clock change on B)





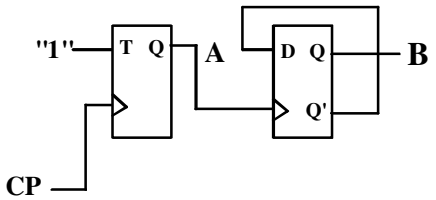
Draw the waveform for B.

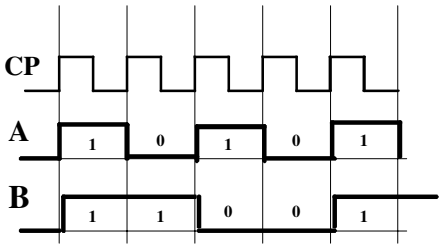
Ripple Counter (Continued)

- **Here's what happens:**
- **Note that B "divides" the frequency in A by 2 -- or doubles the period.**
- **Notice the count (B A) base 10 of:**

  3⇒2⇒1⇒0⇒3⇒2⇒1 …
- **What kind of counter is this?**

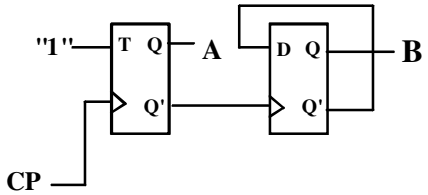
  Down Counter!

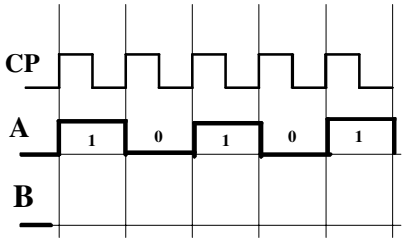




Ripple Counter (Continued)

- **Now consider this (what has changed?):**





Draw waveform B.

Ripple Counter (Continued)

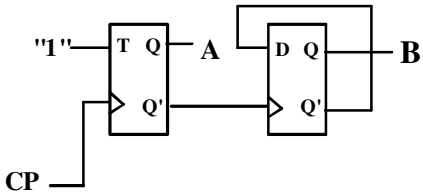
- **Here's what happens:**
- **Note that B "divides" the frequency in A by 2 -- or doubles the period.**
- **Notice the count (B A) base 10 of:**

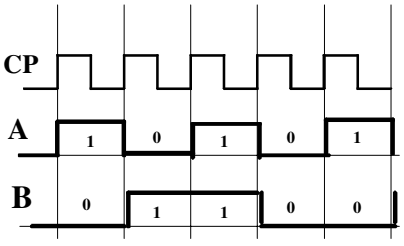
  0⇒1⇒2⇒3⇒0⇒1 …

- **What type of counter is this?**

  Up Counter!

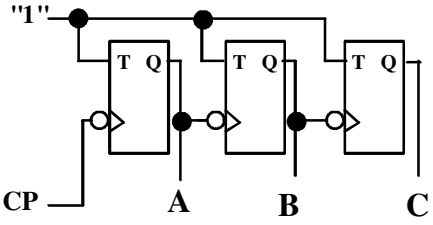
  Can also use negative edge-triggering



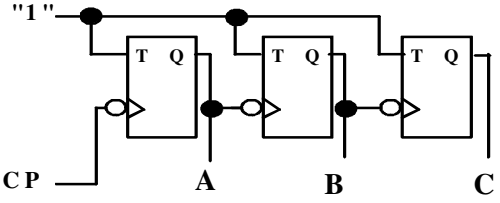


Ripple Counter (Continued)

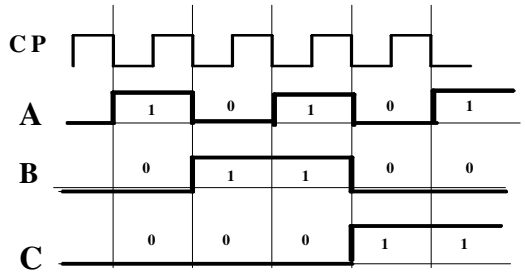
- **The circuits designed this way are called Ripple Counters because each edge sensitive transition (positive in the example's case) causes a change in the next flip-flop's state.**
- **The changes "ripple" up the chain. That is, each transition occurs after a clock to output delay from the stage before.**
- **To see this effect in detail look at the following circuit:**
- **What is the detailed waveform behavior?**



Ripple Counter (Continued)



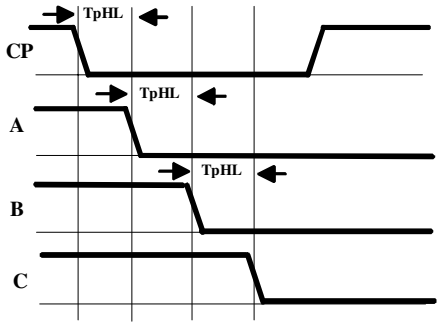
- **Here is the detailed waveform behavior:**
- **What "counts" are shown?**



Ripple Counter (Continued)

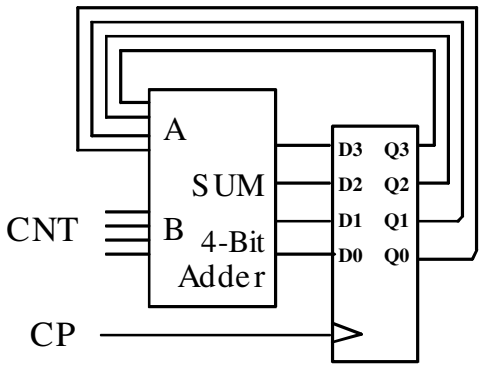
- **Starting with A=B=C = "1", equivalent to (C,B,A) = 7 base 10, the next count will increment the count to (A,B,C) = 0 base 10. Here's what happens in fine timing detail:**

- **The clock to output delay t_{PHL} causes an increasing delay from clock edge for each stage transition.**
- **Thus, the count "ripples" from least to most significant bit.**
- **For *n* bits, total worst case delay is *n* tPHL.**



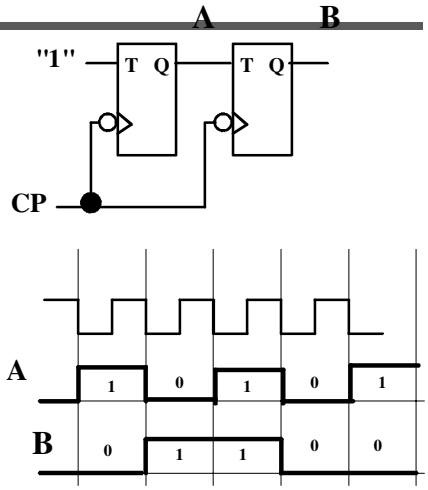
Synchronous Counters

- **In order to eliminate the "ripple" effect, we will use a common clock for each flip-flop and a combinatorial circuit to generate the next state.**
- **One way to generate a counter is with an Adder/Register circuit:⇒**
- **Here "CNT" is the count constant: 0000 is HOLD, 1111 is DOWN and 0001 is UP.**
- **Note potential logic simplification due to constant applied to B.**

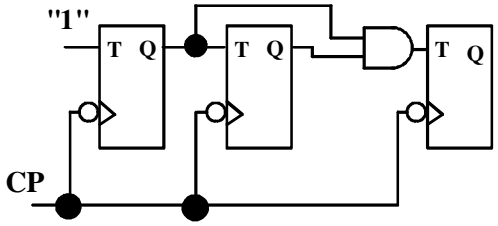


Synchronous Counters (Continued)

- A simple 2-bit synchronous counter can be made with two "T" Flip-Flops:
- Note that the Q from the first stage enables the second stage to toggle.
- Does it count "up" or "down"?
- How do you extend this to three stages?



Synchronous Counters (Continued)



- The concept can be extended to multiple stages. For binary "UP" counters, the upper stages toggle when ALL of the lower stages are at the value "1" and the clock occurs.
- By "ANDing" in a count enable signal to each "T" input, we can produce a "HOLD" count signal.

Synchronous Counters – Serial Gating

- **When a two-input AND gate is used for each stage of the counter with a "ripple-like" carry, this is referred to as serial gating.**
- **As the size of the counter increases the delay through the combinational logic increases roughly in proportion to *n,* the number of stages.**

Synchronous Counters – Parallel Gating

- **When a multiple-input (>2) AND gates are used for each stage of the counter with logic dedicated to each stage or to a group of stages, this is referred to as parallel gating. It resembles carry lookahead in an adder.**
- **As the size of the counter increases the delay through the combinational logic increases roughly in proportion to *n/m,* the number of stages/the group size.**

Design: Synchronous BCD

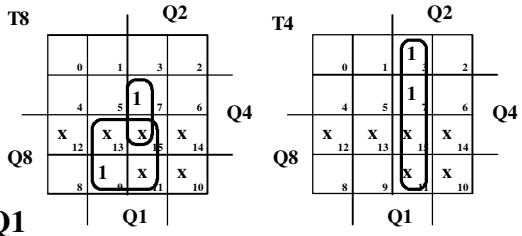
- We can use the sequential logic model to design a synchronous BCD counter with T flip-flops. Below is the State Table.

Current State Q8 Q4 Q2 Q1	Next State Q8 Q4 Q2 Q1	T-FF Excitation T8 T4 T2 T1
0 0 0 0	0 0 0 1	0 0 0 1
0 0 0 1	0 0 1 0	0 0 1 1
0 0 1 0	0 0 1 1	0 0 0 1
0 0 1 1	0 1 0 0	0 1 1 1
0 1 0 0	0 1 0 1	0 0 0 1
0 1 0 1	0 1 1 0	0 0 1 1
0 1 1 0	0 1 1 1	0 0 0 1
0 1 1 1	1 0 0 0	1 1 1 1
1 0 0 0	1 0 0 1	0 0 0 1
1 0 0 1	0 0 0 0	1 0 0 1

- Don't care states have been left out here.

Synchronous BCD (Continued)

- Use K-Maps to minimize the next state function:



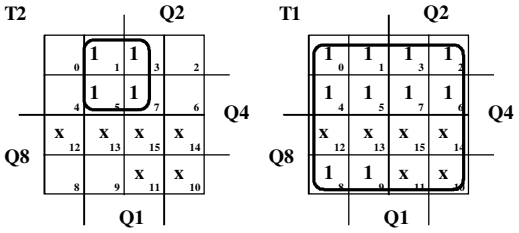
T8 = Q8•Q1 + Q4•Q2•Q1

T4 = Q2•Q1

T2 = Q8'•Q1

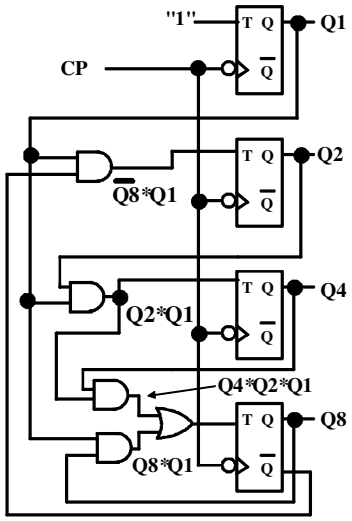
T1 = "1"

Note: Don't Cares are included here.



Synchronous BCD (Continued)

- **The minimized circuit:**



Synchronous BCD (Continued)

What about the Don't Cares now?. <u>ALL</u> Next States are now specified!!

Current State Q8 Q4 Q2 Q1	Next State Q8 Q4 Q2 Q1	T-FF Excitation T8 T4 T2 T1
0 0 0 0	0 0 0 1	0 0 0 1
0 0 0 1	0 0 1 0	0 0 1 1
• • •	• • •	• • •
1 0 0 0	1 0 0 1	0 0 0 1
1 0 0 1	0 0 0 0	1 0 0 1
1 0 1 0	? ? ? ?	0 0 0 1
1 0 1 1	? ? ? ?	1 1 0 1
1 1 0 0	? ? ? ?	0 0 0 1
1 1 0 1	? ? ? ?	1 0 0 1
1 1 1 0	? ? ? ?	0 0 0 1
1 1 1 1	? ? ? ?	1 1 0 1

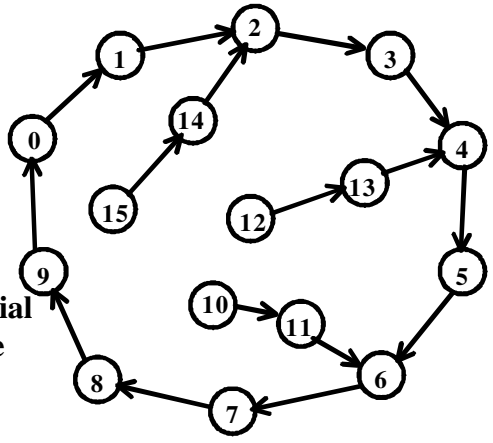
Synchronous BCD (Continued)

- **Don't care states have now been specified by the logic.**

Current State Q8 Q4 Q2 Q1	Next State Q8 Q4 Q2 Q1	T-FF Excitation T8 T4 T2 T1
0 0 0 0	0 0 0 1	0 0 0 1
0 0 0 1	0 0 1 0	0 0 1 1
• • •	• • •	• • •
1 0 0 0	1 0 0 1	0 0 0 1
1 0 0 1	0 0 0 0	1 0 0 1
1 0 1 0	1 0 1 1	0 0 0 1
1 0 1 1	0 1 1 0	1 1 0 1
1 1 0 0	1 1 0 1	0 0 0 1
1 1 0 1	0 1 0 0	1 0 0 1
1 1 1 0	1 1 1 1	0 0 0 1
1 1 1 1	0 0 1 0	1 1 0 1

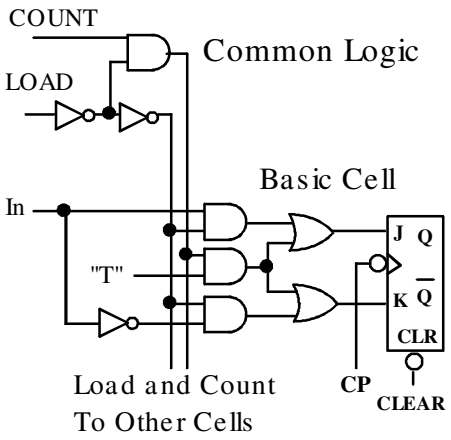
Synchronous BCD (Continued)

- **What does the <u>complete</u> state diagram look like?**



- **How does the sequential machine get into some of the states?**

Counter with Parallel Load



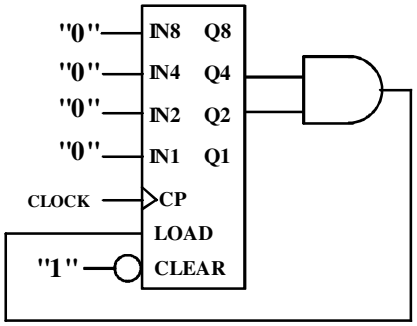
- **If we replace the T flip-flop with a JK flip-flop and some other logic, we can introduce a Hold function and a Parallel Load function.**
- **This basic cell replaces the T-FFs from before.**
- **Note that the "T" input can be used as a normal T-FF if LOAD is low and COUNT is high. The clock, CP, and CLEAR lines are tied to all flip-flops.**

Counting Modulo N

- **The Load feature can be used to preset the counter synchronously on command.**
- **The Clear feature can asynchronously reset the counter to zero. (This can lead to counts which are present only a very short time).**
- **By detecting a "terminal" count of N-1 in a Modulo-N count sequence, we can synchronously load in "zero" to start over.**
- **By detecting a "terminal" count of N in a Modulo-N count sequence, we can clear the count asynchronously to "zero" to start over.**
- **Alternatively, we can detect the "all-ones" terminal count and use load to preset a count of the maximum count value minus (N-1).**

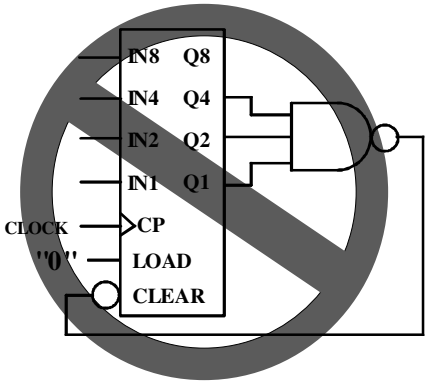
Counting Modulo 7

- **A synchronous 4-bit binary counter with a synchronous load and an asynchronous clear is to be used to make a Modulo 7 counter.**
- **Use the Load feature to detect the count "6" and load in "zero". This gives a count of 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0.... etc.**



Counting Modulo 7, Asyn.Clear

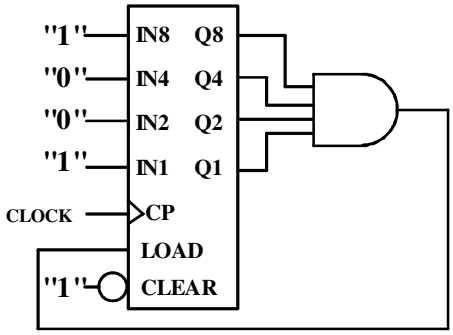
- **A synchronous 4-bit binary counter with a synchronous load and an asynchronous clear is to be used to make a Modulo 7 counter.**
- **Use the Clear feature to detect the count "7" and clear the count to "zero". This gives a count of 0, 1, 2, 3, 4, 5, 6, 7(short)⇒0, 1, 2, 3, 4, 5, 6, 7(short)⇒0, etc.**



- **DON'T DO THIS! Referred to as a "suicide" counter!**

Counting Modulo 7, Preset 9

- A synchronous, 4-bit binary counter with a synchronous load and an asynchronous clear is to be used to make a Modulo 7 counter.
- Use the Load feature to preset the count to "9" when you detect the count "15". This gives a count of 9, 10, 11, 12, 13, 14, 15, 9, 10, 11, 12, 13, 14, 15, 0, ... etc.
- Sometimes the "Detect 15" is built in to the counter.



Timing Sequences

- For digital systems, useful to generate a multi-phase sequence of timing signals to perform different functions at different intervals. There are many ways to do this.
- Here are a few that start with a clock and use a "counter" to divide the clock into separate phases.

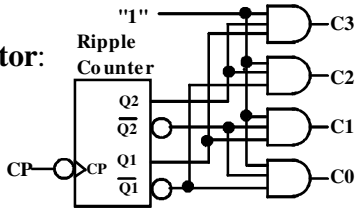
Counter/Decoder - connect the output of a counter to a decoder.

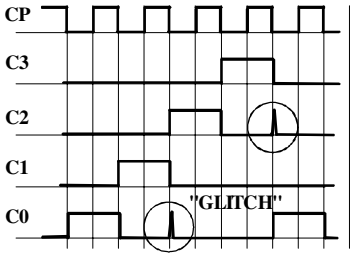
Ring Counter - shift a single pulse down a chain of flip-flops connected as a shift register. For "N" phases, use "N" flip-flops. There can be "unwanted" states if not initialized properly.

Johnson Counter - (Switch-Tail Ring) uses an "N" bit shift register and decoders to produce 2*N phases. There can be "unwanted" states if not initialized properly.

Counter Decoder Example

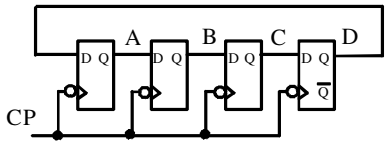
- **Here is one variant of a counter-decoder multi-phase clock generator:**
- **Note the "Glitches" due to the ripple count. A synchronous counter would have been better than the ripple counter.**
- **Even with a synchronous counter, glitches are possible although reduced in duration.**
- **Method OK if "glitches" not a problem, for example, if signals are used as enables in a positive edge-triggered system.**



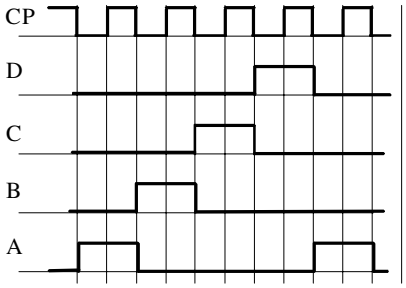


Ring Counter

- **A ring counter is just a shift register with feedback. Assuming the initial state ABCD is 1000 we get the timing diagram shown:**
- **The state must be initialized to operate correctly. Use logic to assure this.**
- **Same circuit can be used with positive edge-triggering.**

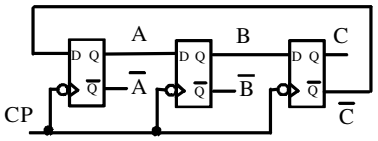


IF ABCD = 1000 At Start:

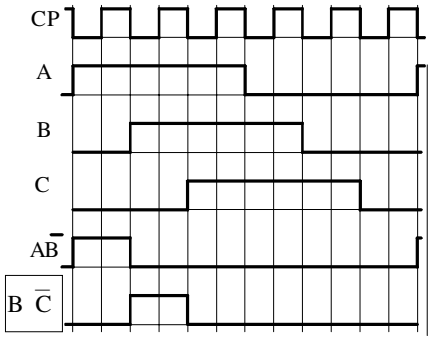


Johnson Counter (Switch-Tail)

- **A Johnson counter is also a shift register with feedback. Assuming the initial state ABC is 000 we get the timing diagram shown:**
- **The state must be initialized to operate correctly. Use logic to assure this.**



IF ABCD = 000 At Start:



Verilog for Registers and Counters

- **Register – same as flip-flop except multiple bits:** `reg[3:0] Q;`

```
input[3:0] D;
always@(posedge CLK or posedge RESET)
begin
      if (RESET)  Q <= 4'b0000;
      else        Q <= D;
end
```

- **Shift Register – use concatenate:**

```
Q <= {Q[2:0], SI};
```

- **Counter – use increment/decrement:**

```
count <= count + 1; or count <= count - 1
```

Verilog Description of Left Shift Register

```
// 4-bit Shift Register
   with Reset
// (See Figure 5-3)

module srg_4_r_v (CLK,
   RESET, SI, Q,SO);
    input CLK, RESET, SI;
    output [3:0] Q;
    output SO;
   reg [3:0] Q;
   assign SO = Q[3];

always@(posedge CLK or
      posedge RESET)
begin
   if (RESET)
      Q <= 4'b0000;
   else
      Q <= {Q[2:0], SI};
end
endmodule
```


Verilog Description of Binary Counter

```
// 4-bit Binary Counter with
   Reset
// (See Figure 5-10)

module count_4_r_v (CLK,
   RESET, EN, Q, CO);
    input CLK, RESET, EN;
    output [3:0] Q;
    output CO;

reg [3:0] count;
assign Q = count;
assign CO = (count == 4'b1111
   && EN == 1'b1) ? 1 : 0;

always@(posedge CLK or
   posedge RESET)
    begin
    if (RESET)
       count <= 4'b0;
    else if (EN)
          count <= count + 1;
    end
endmodule
```