Chapter 9: Modeling Counters

Counters are a special case of finite state machines because they move linearly through their discrete states (either forward or backwards) and typically are implemented with state-encoded outputs. Due to this simplified structure and widespread use in digital systems, Verilog allows counters to be modeled using a single procedural block with arithmetic operators (i.e., + and −). This enables a more compact model and allows much wider counters to be implemented in a practical manner. This chapter will cover some of the most common techniques for modeling counters.

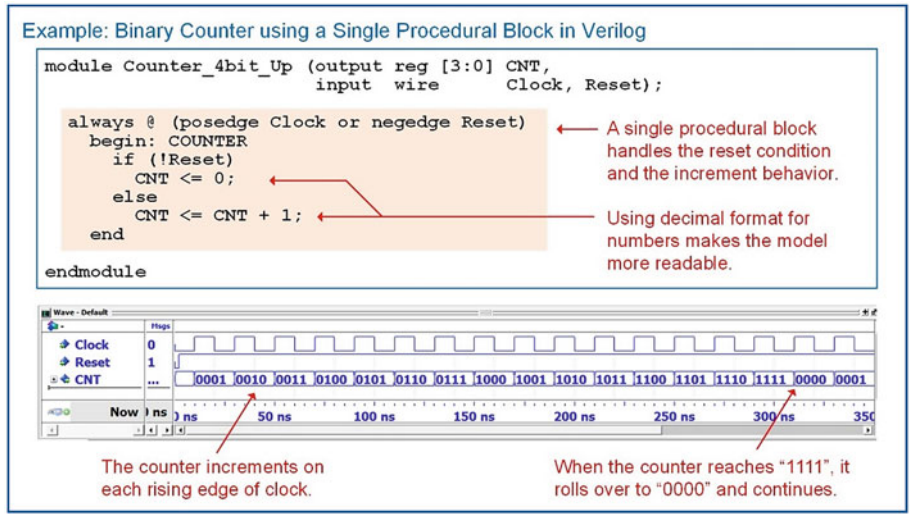
Learning Outcomes—After completing this chapter, you will be able to:

9.1 Design a behavioral model for a counter using a single procedural block.
9.2 Design a behavioral model for a counter with enable and load capability.

9.1 Modeling Counters with a Single Procedural Block

9.1.1 Counters in Verilog Using the Type reg

Let's look at how we can model a 4-bit, binary up counter with an output called *CNT*. We want to model this counter using the "+" operator to avoid having to explicitly define a state code for each state as in the three-block modeling approach to FSMs. The "+" operator works on the type reg, so the counting behavior can simply be modeled using CNT <= CNT + 1. The procedural block also needs to handle the reset condition. Both the Clock and Reset signals are listed in the sensitivity list. Within the block, an if-else statement is used to handle both the reset and increment behaviors. Example 9.1 shows the Verilog model and simulation waveform for this counter. When the counter reaches its maximum value of "1111," it rolls over to "0000" and continues counting because it is declared to only contain 4-bits.



Example 9.1
Binary counter using a single procedural block in Verilog

B. J. LaMeres, *Quick Start Guide to Verilog*, https://doi.org/10.1007/978-3-030-10552-5_9

9.1.2 Counters with Range Checking

When a counter needs to have a maximum range that is different from the maximum binary value of the count vector (i.e., $<2^n - 1$), then the procedural block needs to contain *range checking* logic. This can be modeled by inserting a nested if-else statement beneath of the else clause that handles the behavior for when the counter receives a rising clock edge. This nested if-else first checks whether the count has reached its maximum value. If it has, it is reset back to it minimum value. If it hasn't, the counter is incremented as usual. Example 9.2 shows the Verilog model and simulation waveform for a counter with a minimum count value of 0_{10} and a maximum count value of 10_{10}. This counter still requires 4-bits to be able to encode 10_{10}.

Example: Binary Counter with Range Checking in Verilog

```
module Counter_4bit_Up (output reg [3:0] CNT,
                        input  wire      Clock, Reset);

  always @ (posedge Clock or negedge Reset)
    begin: COUNTER
      if (!Reset)
        CNT <= 0;
      else
        if (CNT == 10)
          CNT <= 0;
        else
          CNT <= CNT + 1;
    end

endmodule
```

A nested if-else statement checks if the counter has reached its maximum value. If it has, it is reset back to zero. If it hasn't, it increments.

Once the counter reaches 10, it is set back to 0. In this waveform, the radix of the counter is formatted as unsigned decimal.

Example 9.2
Binary counter with range checking in Verilog

Concept Check

CC9.1 If a counter is modeled using only one procedural block in Verilog, is it still a finite state machine? Why or why not?

(A) Yes. It is just a special case of a FSM that can easily be modeled using one block. Synthesizers will recognize the single block model as a FSM.

(B) No. Using only one block will synthesize into combinational logic. Without the ability to store a state, it is not a finite state machine.

9.2 Counter with Enables and Loads

9.2.1 Modeling Counters with Enables

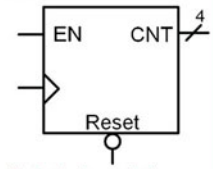
Including an *enable* in a counter is a common technique to prevent the counter from running continuously. When the enable is asserted, the counter will increment on the rising edge of the clock as usual. When the enable is de-asserted, the counter will simply hold its last value. Enable lines are synchronous, meaning that they are only evaluated on the rising edge of the clock. As such, they are modeled using a nested if-else statement within the main if-else statement checking for a rising edge of the clock. Example 9.3 shows an example model for a 4-bit counter with enable.

Example: Binary Counter with Enable in Verilog

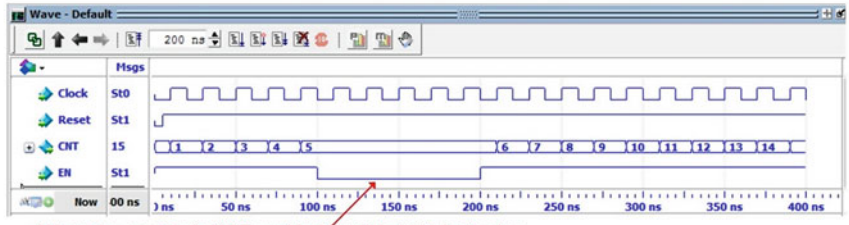
```
module Counter_4bit_Up (output reg [3:0] CNT,
                        input  wire      Clock, Reset, EN);

  always @ (posedge Clock or negedge Reset)
    begin: COUNTER
      if (!Reset)
        CNT <= 0;
      else
        if (EN)
          CNT <= CNT + 1;
    end

endmodule
```



The EN is synchronous to the clock, so its logic is nested beneath the portion of the main if-else clause that handles the behavior when the counter receives a rising edge of clock.



When the counter is NOT enabled, it will hold its last value.

Example 9.3
Binary counter with enable in Verilog

9.2.2 Modeling Counters with Loads

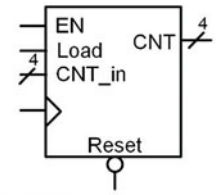
A counter with a *load* has the ability to set the counter to a specified value. The specified value is provided on an input port (i.e., CNT_in) with the same width as the counter output (CNT). A synchronous load input signal (i.e., Load) is used to indicate when the counter should set its value to the value present on CNT_in. Example 9.4 shows an example model for a 4-bit counter with load capability.

Example: Binary Counter with Load in Verilog

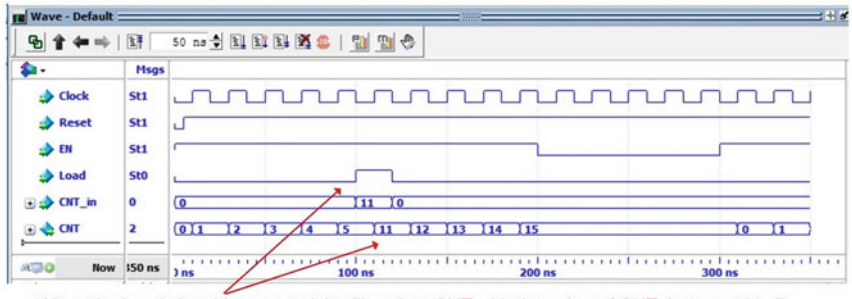
```
module Counter_4bit_Up (output reg [3:0]  CNT,
                        input  wire       Clock, Reset, EN, Load,
                        input  wire [3:0] CNT_in);

  always @ (posedge Clock or negedge Reset)
    begin: COUNTER
      if (!Reset)
        CNT <= 0;
      else
        if (EN)
          if (Load)
            CNT <= CNT_in;
          else
            CNT <= CNT + 1;
    end

endmodule
```



A nested if-else statement is used to load CNT with CNT_in when the Load signal is asserted and the counter receives a rising edge of clock.



When the Load signal is asserted, it will update CNT with the value of CNT_in (e.g., "11_{10}").

Example 9.4
Binary counter with load in Verilog

Concept Check

CC9.2 If a counter is modeled using only one procedural block in Verilog, is it still a finite state machine? Why or why not?

(A) Yes. It is just a special case of a FSM that can easily be modeled using one block. Synthesizers will recognize the single block model as a FSM.

(B) No. Using only one block will synthesize into combinational logic. Without the ability to store a state, it is not a finite state machine.

Summary

- Counters are a special type of finite state machine that can be modeled using a single procedural block. Only the clock and reset signals are listed in the sensitivity list of the counter block.
- Registers are modeled in Verilog in a similar manner to a D-flip-flop with a synchronous enable. The only difference is that the inputs and outputs are vectors.
- Register Transfer Level, or RTL, modeling provides a higher level of abstraction for moving and manipulating vectors of data in a synchronous manner.

Exercise Problems

Section 9.1: Modeling Counters with a Single Procedural Block

9.1.1 Design a Verilog behavioral model for a 16-bit, binary up counter using a single procedural block. The block diagram for the port definition is shown in Fig. 9.1.

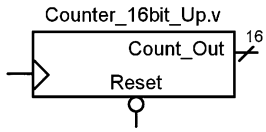


Fig. 9.1
16-Bit Binary Up Counter Block Diagram

9.1.2 Design a Verilog behavioral model for a 16-bit, binary up counter with range checking using a single procedural block. The block diagram for the port definition is shown in Fig. 9.1. Your counter should count up to 60,000 and then start over at 0.

Section 9.2: Counters with Enables and Loads

9.2.1 Design a Verilog behavioral model for a 16-bit, binary up counter with enable using a single procedural block. The block diagram for the port definition is shown in Fig. 9.2.

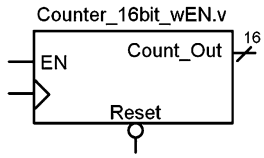


Fig. 9.2
16-Bit Binary Up Counter with Enable Block Diagram

9.2.2 Design a Verilog behavioral model for a 16-bit, binary up counter with enable and load using a single procedural block. The block diagram for the port definition is shown in Fig. 9.3.

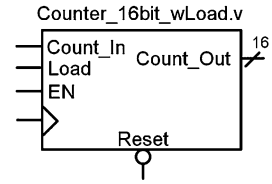


Fig. 9.3
16-Bit Binary Up Counter with Load Block Diagram

9.2.3 Design a Verilog behavioral model for a 16-bit, binary up/down counter using a single procedural block. The block diagram for the port definition is shown in Fig. 9.4. When Up = 1, the counter will increment. When Up = 0, the counter will decrement.

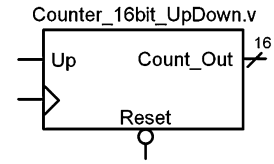


Fig. 9.4
16-Bit Binary Up/Down Counter Block Diagram