Design Code: 4-bit up-down counter

module counter(

input clk,en,load, updown,

input[3:0] data,

Output reg[3:0] q = ‘0

);

always@(negedge clk)

if(en)

if(load)

q<=data;

else

if(updown)

q<=q+4’b1;

else

q<=q-4’b1;

always@(negedge clk)

$strobe(“time=%3t, q=%0h”,$time,q);

endmodule

* Create a testbench to verify the design and analyze the waveforms
* Testbench will have a clocking block where:
  + Load=0, data=0, updown=1, en=1 at time 0.
  + After 3 clk cycles drive data to value 3’h5
  + After 2 clk cycles make en=0
  + Wait 1 clk cycles then drive data to “z”
  + Wait 4 clk cycles, then drive bit 3 of data to “0”
  + After 4 clk cycles, drive data back to initial value of 3’h5.

module tb\_counter;

// Declare signals for the counter module

logic clk;

logic en;

logic load;

logic updown;

logic [3:0] data;

logic [3:0] q;

// Instantiate the counter module

counter uut (

.clk(clk),

.en(en),

.load(load),

.updown(updown),

.data(data),

.q(q)

);

// Clock generation

always begin

#5 clk = ~clk; // Toggle clock every 5ns (10ns period)

end

// Clocking block

clocking cb @(posedge clk);

default input #1ns output; // Default timing

input en, load, updown; // Input signals

input [3:0] data; // Data input

output q; // Data output (from UUT)

endclocking

// Test stimulus generation

initial begin

// Initialize signals

clk = 0;

en = 1;

load = 0;

updown = 1;

data = 4'b0000;

// Apply stimulus at time 0

#0;

// Initial state: Load = 0, data = 0, updown = 1, en = 1

$display("time=%3t, en=%b, load=%b, updown=%b, data=%h, q=%h", $time, en, load, updown, data, q);

// After 3 clock cycles, drive data to value 3'h5

#30;

data = 3'h5; // Change data to 5

$display("time=%3t, en=%b, load=%b, updown=%b, data=%h, q=%h", $time, en, load, updown, data, q);

// After 2 clock cycles, make en=0

#20;

en = 0; // Disable counter (en=0)

$display("time=%3t, en=%b, load=%b, updown=%b, data=%h, q=%h", $time, en, load, updown, data, q);

// Wait 1 clock cycle then drive data to 'z' (high impedance)

#10;

data = 4'bz; // Set data to high impedance (Z state)

$display("time=%3t, en=%b, load=%b, updown=%b, data=%h, q=%h", $time, en, load, updown, data, q);

// Wait 4 clock cycles, then drive bit 3 of data to 0 (clear MSB)

#40;

data[3] = 0; // Set bit 3 of data to 0

$display("time=%3t, en=%b, load=%b, updown=%b, data=%h, q=%h", $time, en, load, updown, data, q);

// After 4 clock cycles, drive data back to initial value 3'h5

#40;

data = 3'h5; // Revert data back to 3'h5

$display("time=%3t, en=%b, load=%b, updown=%b, data=%h, q=%h", $time, en, load, updown, data, q);

// End simulation

#10;

$finish;

end

// Optionally display waveforms using the $display command

initial begin

// Show waveform for the signals

$dumpfile("tb\_counter.vcd");

$dumpvars(0, tb\_counter);

end

endmodule

Testbench with random stimulus:

module tb\_counter;

// Declare signals for the counter module

logic clk;

logic en;

logic load;

logic updown;

logic [3:0] data;

logic [3:0] q;

// Instantiate the counter module

counter uut (

.clk(clk),

.en(en),

.load(load),

.updown(updown),

.data(data),

.q(q)

);

// Clock generation

always begin

#5 clk = ~clk; // Toggle clock every 5ns (10ns period)

end

// Random stimulus generation

initial begin

// Initialize signals

clk = 0;

en = 0;

load = 0;

updown = 0;

data = 4'b0000;

// Apply some initial random stimulus

#10; // Wait for a few cycles

// Randomize the stimulus for each cycle

repeat (20) begin

// Randomize the control signals

en = $random; // Random enable signal (0 or 1)

load = $random; // Random load signal (0 or 1)

updown = $random; // Random up/down control (0 or 1)

data = $random; // Random data value (4 bits)

// Apply stimulus for one cycle

#10; // Wait for 1 clock cycle

// Optionally display the stimulus and output

$display("time=%3t, en=%b, load=%b, updown=%b, data=%h, q=%h", $time, en, load, updown, data, q);

end

$finish;

end

endmodule