Nang Thiri Wutyi

20113

11/5/2024

**Verilog Lab 5**

1. **DFF Module**

`timescale 1ns/100ps

module dff(input D\_i, clock\_i, output reg Q\_o);

always @(posedge clock\_i)

Q\_o <= D\_i;

endmodule

1. **Latch with XOR logic**

`timescale 1ns/100ps

module xor3 (input a\_i, b\_i, c\_i, clock\_i, output reg Q\_o);

always @(posedge clock\_i) begin

if (a\_i)

Q\_o <= c\_i;

else

Q\_o <= b\_i;

end

endmodule

1. **Original Counter Module**

module counter (input clock, input [3:0] in, input latch, input dec, output zero);

reg [3:0] value;

always @(posedge clock) begin

if (latch)

value <= in;

else if (dec && !zero)

value <= value - 1'b1;

end

assign zero = (value == 4'b0000);

endmodule

**Modified Counter Module**

// counter\_modified.sv

module counter\_modified (

input clock,

input [3:0] in,

input latch,

input dec,

output zero

);

reg [3:0] value;

always @(posedge clock) begin

if (latch)

value <= in; // Load initial value when latch is high

else if (dec && !zero)

value <= value - 2'b10; // Decrement by 2 if dec is high and zero flag is not set

end

assign zero = (value == 4'b0000 || value == 4'b0001); // Set zero flag high at 0000 or 0001

endmodule

**Modified Testbench**

// counter\_modified\_tb.sv

`timescale 1ns/100ps

module counter\_modified\_tb;

reg clock;

reg latch;

reg dec;

reg [3:0] in;

wire zero;

// Instantiate the modified counter

counter\_modified uut (

.clock(clock),

.in(in),

.latch(latch),

.dec(dec),

.zero(zero)

);

// Generate clock signal with a period of 10 time units

initial begin

clock = 0;

forever #5 clock = ~clock;

end

// Test sequence

initial begin

// Initialize inputs

in = 4'b0101; // Starting value

latch = 1; // Load initial value

dec = 0; // Decrement disabled initially

#10;

// Start decrementing by setting latch to 0 and dec to 1

latch = 0;

dec = 1;

// Monitor the value and zero flag for a few cycles

repeat (10) begin

#10; // Wait for one clock cycle

$display("Time: %0t | Value: %b | Zero Flag: %b", $time, uut.value, zero);

end

$stop; // Stop simulation

end

endmodule

**Results**

**2024-11-06 05:24:25 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out**

**warning: Some design elements have no explicit time unit and/or**

**: time precision. This may cause confusing timing results.**

**: Affected design elements are:**

**: -- module counter\_modified declared here: design.sv:2**

**Time: 200 | Value: 0011 | Zero Flag: 0**

**Time: 300 | Value: 0001 | Zero Flag: 1**

**Time: 400 | Value: 0001 | Zero Flag: 1**

**Time: 500 | Value: 0001 | Zero Flag: 1**

**Time: 600 | Value: 0001 | Zero Flag: 1**

**Time: 700 | Value: 0001 | Zero Flag: 1**

**Time: 800 | Value: 0001 | Zero Flag: 1**

**Time: 900 | Value: 0001 | Zero Flag: 1**

**Time: 1000 | Value: 0001 | Zero Flag: 1**

**Time: 1100 | Value: 0001 | Zero Flag: 1**

**testbench.sv:44: $stop called at 1100 (100ps)**

**\*\* VVP Stop(0) \*\***

**\*\* Flushing output streams.**

**\*\* Current simulation time is 1100 ticks.**

**Execution interrupted or reached maximum runtime.**

**Exit code expected: 0, received: 137**

**Done**

#### **Race Condition Handling**

To avoid race conditions in sequential designs, the following guidelines were implemented:

* **Nonblocking Assignments for Sequential Logic**: Used nonblocking (<=) for sequential operations.
* **Blocking Assignments for Combinational Logic**: Applied blocking (=) for purely combinational always blocks.
* **Consistent Assignment Practice**: Avoided mixing blocking and nonblocking assignments within the same always block.
* **Single Assignment Block for Each Variable**: Ensured that each variable is assigned only in one always block.
* **Avoiding #0 Delays**: #0 delays were not used to prevent timing issues.

1. **XOR Circuit Implementation**

`timescale 1ns/100ps

module xor3 (input B\_i, D\_i, sel\_i, clock, output reg E\_o);

always @(posedge clock) begin

case (sel\_i)

0: E\_o <= D\_i ^ B\_i;

1: E\_o <= B\_i;

endcase

end

endmodule

**Testbench**

`timescale 1ns/100ps

module xor3\_tb;

reg B\_i, D\_i, sel\_i, clock;

wire E\_o;

xor3 uut (.B\_i(B\_i), .D\_i(D\_i), .sel\_i(sel\_i), .clock(clock), .E\_o(E\_o));

initial begin

clock = 0;

forever #5 clock = ~clock;

end

initial begin

// Test cases

B\_i = 0; D\_i = 1; sel\_i = 0; #10;

B\_i = 1; D\_i = 1; sel\_i = 1; #10;

B\_i = 0; D\_i = 0; sel\_i = 0; #10;

$stop;

end

endmodule