**Lab 6**

**Logic Design using Verilog**

**Faizan Bangash**

**Section 506**

**TA: Lin Huang**

**DUE: 16 October 2018**

**Cruisectrl.v**

|  |
| --- |
| `timescale 1ns / 1ps |
|  |  |
|  | module CruiseControl( |
|  | input clock, |
|  | input reset, |
|  | input throttle, |
|  | input set, |
|  | input accel, |
|  | input coast, |
|  | input cancel, |
|  | input resume, |
|  | input brake, |
|  | output reg [7:0] speed, |
|  | output reg [7:0] cruise\_speed, |
|  | output reg cruise\_status |
|  | ); |
|  |  |
|  | // register for state |
|  | reg [1:0] state; // Current state |
|  |  |
|  |  |
|  | // flags to indicate if brake pulse or resume pulse is given |
|  | reg brake\_appl; // indicates brake pulse applied |
|  | reg resume\_appl; // indicates resume pulse applied |
|  |  |
|  |  |
|  | // Define states |
|  | localparam CRUISE\_ON = 1'b1, // Cruise Control is set |
|  | CRUISE\_OFF = 1'b0; // Cruise Control is unset ( turned off ) |
|  |  |
|  |  |
|  |  |
|  | // next state logic. |
|  | always @( posedge clock, posedge reset ) begin |
|  |  |
|  | if( reset ) begin |
|  | cruise\_speed <= 8'hzz; |
|  | state <= CRUISE\_OFF; |
|  | cruise\_status <= 1'b0; |
|  | speed <= 0; |
|  | end |
|  | else begin |
|  | case(state) |
|  |  |
|  | CRUISE\_OFF: begin |
|  | cruise\_status <= 1'b0; |
|  | // if throttle is on, increment the speed by 1 mph every cycle |
|  | if(throttle) begin |
|  | speed <= speed + 1; |
|  | // if brake pulse was applied , cancel it |
|  | if( brake\_appl != 1'b0 ) |
|  | brake\_appl <= 1'b0; |
|  | end |
|  | else begin |
|  | if(speed > 0) begin |
|  | speed <= speed - 1; |
|  | end |
|  | end |
|  |  |
|  | if (set) begin |
|  | // if set is applied and speed is greater than 45mph, set the cruise speed to |
|  | // the current speed, and change state to CRUISE\_ON |
|  | if(speed > 45) begin |
|  | cruise\_speed <= speed; |
|  | state <= CRUISE\_ON; |
|  | // if brake pulse was applied , cancel it |
|  | if( brake\_appl != 1'b0 ) begin |
|  | brake\_appl <= 1'b0; |
|  | end |
|  |  |
|  | end |
|  | end |
|  | // if brake is applied, set brake\_appl flag to zero |
|  | if ( brake ) begin |
|  | brake\_appl <= 1; |
|  | end |
|  | // if brake flag is on, decrement the speed by 2 every clock cycle |
|  | if ( brake\_appl )begin |
|  | speed <= speed - 2; |
|  | end |
|  | // if resume is applied, move to CRUISE\_ON state |
|  | if ( resume && cruise\_speed > 0 && speed > 0 ) begin |
|  | state <= CRUISE\_ON; |
|  | resume\_appl <= 1; |
|  | end |
|  |  |
|  |  |
|  |  |
|  | end |
|  |  |
|  | CRUISE\_ON : begin |
|  | cruise\_status <= 1'b1; |
|  | // if throttle is applied, increment speed by 1 mph every cycle. |
|  | // do not change cruise speed |
|  | if( throttle ) begin |
|  | speed <= speed + 1; |
|  | end |
|  |  |
|  | else begin |
|  | // if accel is applied, increment the cruise speed by 1 mph every cycle |
|  | if ( accel ) begin |
|  |  |
|  | cruise\_speed <= cruise\_speed + 1; |
|  | speed <= speed + 1; |
|  | state <= CRUISE\_ON; |
|  | // cancel resume if it was applied |
|  | if( resume\_appl == 1'b1 ) |
|  | resume\_appl <= 1'b0; |
|  |  |
|  |  |
|  | end |
|  | // if coast is applied, decrement the cruise speed by 1 mph every cycle |
|  | else if ( coast ) begin |
|  | cruise\_speed <= cruise\_speed - 1; |
|  | speed <= speed - 1; |
|  | state <= CRUISE\_ON; |
|  | // cancel resume if it was applied |
|  | if( resume\_appl == 1'b1 ) |
|  | resume\_appl <= 1'b0; |
|  |  |
|  | end |
|  | // if cancel is applied, turn off cruise control |
|  | else if ( cancel ) begin |
|  | state <= CRUISE\_OFF; |
|  | // cancel resume if it was applied |
|  | if( resume\_appl == 1'b1 ) |
|  | resume\_appl <= 1'b0; |
|  | end |
|  | // if brake is applied, turn off cruise control and decrement |
|  | // speed at 2 mph every cycle which is handled in CRUISE\_OFF state |
|  | else if ( brake ) begin |
|  |  |
|  | brake\_appl <= 1'b1; |
|  | state <= CRUISE\_OFF; |
|  |  |
|  | end |
|  | // if speed is less than cruise speed and , increase it to reach cruise speed |
|  | else if (speed < cruise\_speed && resume\_appl ) begin |
|  | speed <= speed + 1; |
|  | end |
|  | else begin |
|  | // if no input is applied, then decrease until cruise\_speed is reached |
|  | if(speed > cruise\_speed) begin |
|  | speed <= speed - 1; |
|  | end |
|  |  |
|  | end |
|  | end |
|  | end |
|  |  |
|  | endcase |
|  | end // else |
|  | end // always |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | endmodule |

Cruisectrl testbench code

|  |
| --- |
| `timescale 1ns / 1ps |
|  |  |
|  |  |
|  | module CruiseControl\_tb(); |
|  |  |
|  | reg clock; |
|  | reg reset; |
|  | wire state; |
|  |  |
|  | reg throttle, set, accel; |
|  | reg coast, cancel, resume, brake; |
|  | wire [7:0] speed; |
|  | wire [7:0] cruise\_speed; |
|  | wire cruise\_status; |
|  |  |
|  | // instantiate cruise control module |
|  | CruiseControl cc( .clock(clock), |
|  | .reset(reset), |
|  | .throttle(throttle), |
|  | .set(set), |
|  | .accel(accel), |
|  | .coast(coast), |
|  | .cancel(cancel), |
|  | .resume(resume), |
|  | .brake(brake), |
|  | .speed(speed), |
|  | .cruise\_speed(cruise\_speed), |
|  | .cruise\_status(cruise\_status) ); |
|  |  |
|  | // monitor state |
|  | assign state = cc.state; |
|  | // define clock |
|  | always #50 clock = ~clock; |
|  |  |
|  | // drive test stimuli |
|  | initial begin |
|  | $monitor($time,"speed = %d", speed); |
|  | // initialize clock |
|  | clock = 0; |
|  | // reset the design |
|  | reset = 0; |
|  | #150 reset = 1; |
|  | #50 reset = 0; |
|  | // 1) increase the speed to 30 mph using throttle |
|  | #10; |
|  | throttle = 1; |
|  | wait(speed == 30); |
|  | // try to set the cruise control using set |
|  | @(negedge clock); |
|  | #25; |
|  | set = 1; |
|  | #50; |
|  | set = 0; |
|  | // turn throttle off |
|  | throttle = 0; |
|  | // wait until speed is 20mph |
|  | wait(speed == 20); |
|  | // now turn throttle on |
|  | throttle = 1; |
|  | // wait until speed is 50 mph |
|  | wait(speed == 50); |
|  | //try to set the cruise control speed at this point, it should work |
|  | @( negedge clock); |
|  | set = 1; |
|  | #100; |
|  | set = 0; |
|  | // continue to increase the speed until 60mph |
|  | wait (speed == 60); |
|  | #100; |
|  | // take throttle off at this point |
|  | throttle = 0; |
|  | // wait till speed drops until 50 |
|  | wait(speed == 50); |
|  | // cruise for 5 clock cycles at this cruising speed |
|  | repeat(5) @(negedge clock) |
|  | #100; |
|  | // now apply brake |
|  | #50; |
|  | brake = 1; |
|  | #100; |
|  | brake = 0; |
|  | wait(speed == 30); |
|  | #100; |
|  | // provide resume pulse |
|  | # 50 |
|  | resume = 1; |
|  | #100; |
|  | resume = 0; |
|  | wait (speed == 50); |
|  | repeat(5) @(negedge clock); |
|  | #100; |
|  | // give five consecutive accel pulses |
|  | repeat(5) begin |
|  | @(negedge clock); |
|  | #25; |
|  | accel = 1; |
|  | #50; |
|  | accel = 0; |
|  | end |
|  | #100; |
|  |  |
|  | // give five consecutive coast pulses |
|  | repeat(5) begin |
|  | @(negedge clock); |
|  | #25; |
|  | coast = 1; |
|  | #50; |
|  | coast = 0; |
|  | end |
|  | #100; |
|  | repeat(5) |
|  | #100; |
|  | // apply cance |
|  | @(negedge clock); |
|  | #25 cancel = 1; |
|  | #50 cancel = 0; |
|  | // wait until speed goes to zero |
|  | wait(speed == 0); |
|  | #100; |
|  | $finish; |
|  |  |
|  | end |
|  |  |
|  | endmodule |