**Clock generator code**

module square\_wavegen

(

i\_clock,

i\_enable,

i\_switch\_1,

o\_clock\_freq

);

input i\_clock;

input i\_enable;

input i\_switch\_1;

output o\_clock\_freq;

// Constants (parameters) to create the frequencies needed:

// Input clock is 100 MHz, i.e of the chosen FPGA

// Formula is: (100 MHz / 600 Hz \* 50% duty cycle) for 600hz frequency

// So for 1MHz: 100Mhz / 1Mhz\* 0.5 = 49

parameter c\_CNT\_600HZ = 83334;

parameter c\_CNT\_1MHZ = 49;

// These signals will be the counters:

reg [31:0] r\_CNT\_600HZ = 0;

reg [31:0] r\_CNT\_1MHZ = 0;

// These signals will toggle at the frequencies needed:

reg r\_TOGGLE\_600HZ = 1'b0;

reg r\_TOGGLE\_1MHZ = 1'b0;

// One bit select

reg r\_CLOCK\_SELECT;

wire w\_CLOCK\_SELECT;

begin

// All always blocks toggle a specific signal at a different frequency.

// They all run continuously even if the switches are

// not selecting their particular output.

always @ (posedge i\_clock)

begin

if (r\_CNT\_600HZ == c\_CNT\_600HZ-1) // -1, since counter starts at 0

begin

r\_TOGGLE\_600HZ <= !r\_TOGGLE\_600HZ;

r\_CNT\_600HZ <= 0;

end

else

r\_CNT\_600HZ <= r\_CNT\_600HZ + 1;

end

always @ (posedge i\_clock)

begin

if (r\_CNT\_1MHZ == c\_CNT\_1MHZ-1) // -1, since counter starts at 0

begin

r\_TOGGLE\_1MHZ <= !r\_TOGGLE\_1MHZ;

r\_CNT\_1MHZ <= 0;

end

else

r\_CNT\_1MHZ <= r\_CNT\_1MHZ + 1;

end

// Create a multiplexer based on switch inputs

always @ (\*)

begin

case (i\_switch\_1)

1'b0 : r\_CLOCK\_SELECT <= r\_TOGGLE\_600HZ;

1'b1 : r\_CLOCK\_SELECT <= r\_TOGGLE\_1MHZ;

endcase

end

assign o\_clock\_freq = r\_CLOCK\_SELECT & i\_enable;

end

endmodule

**TESTBENCH (600hz)**

module tb();

reg i\_clock,i\_enable,i\_switch\_1;

wire o\_clock\_freq;

square\_wavegen inst1 (.i\_clock(i\_clock),.i\_enable(i\_enable),.i\_switch\_1(i\_enable),.o\_clock\_freq(o\_clock\_freq));

initial

i\_clock = 1'b0;

always

#5 i\_clock = ~i\_clock;

initial

begin

i\_enable = 1; i\_switch\_1 = 0;

end

endmodule

**TESTBENCH (1.023MHz)**

module tb();

reg i\_clock,i\_enable,i\_switch\_1;

wire o\_clock\_freq;

square\_wavegen inst1 (.i\_clock(i\_clock),.i\_enable(i\_enable),.i\_switch\_1(i\_enable),.o\_clock\_freq(o\_clock\_freq));

initial

i\_clock = 1'b0;

always

#5 i\_clock = ~i\_clock;

initial

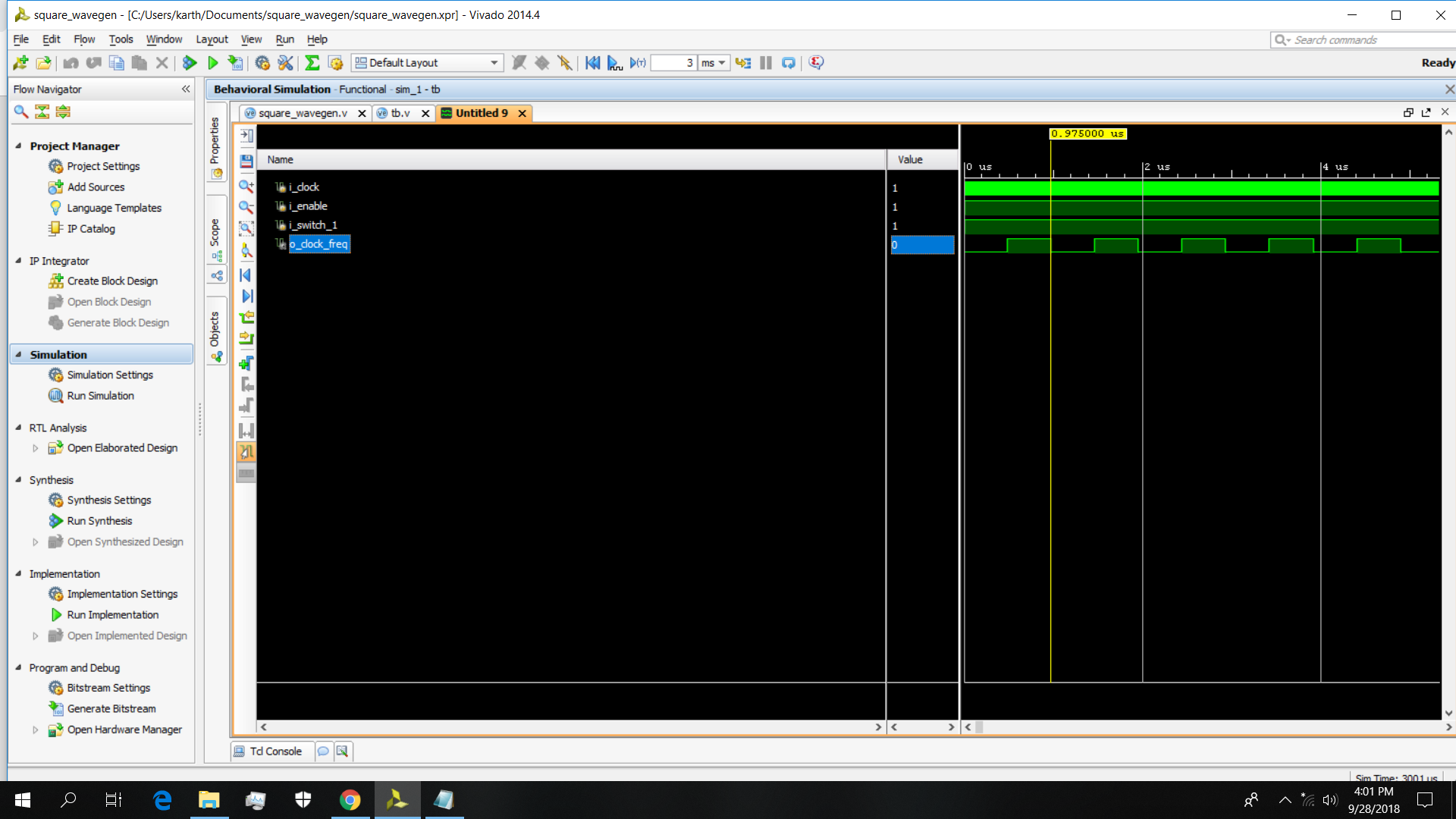
begin

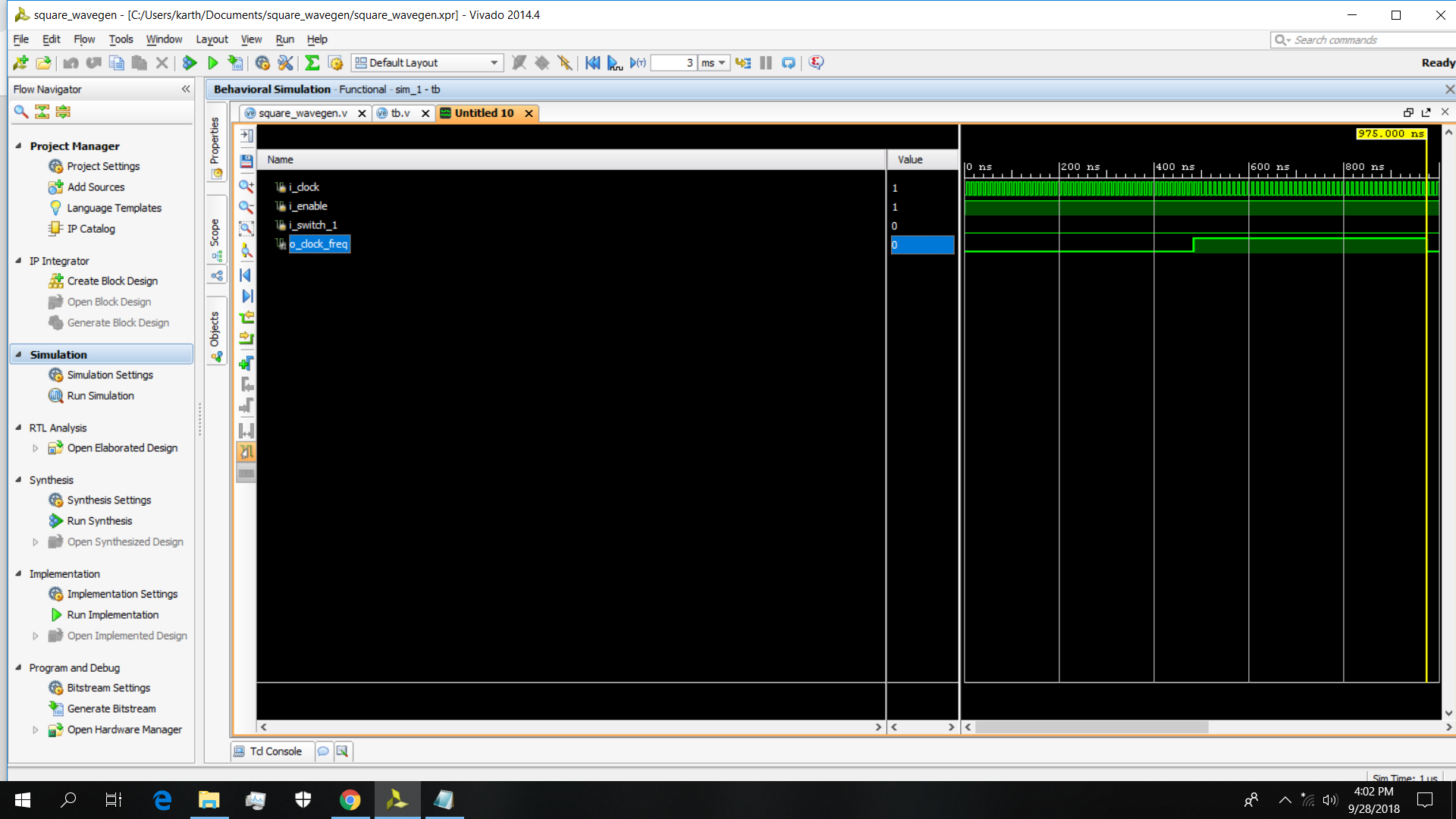
i\_enable = 1; i\_switch\_1 = 1;

end

endmodule

**SIMULATION CAPTURE**





**RTL DESIGN**

