# REAL TIME DIGITAL STOPWATCH DESIGN
# AND
# FPGA IMPLEMENTATION

Trevor Kingery

12/7/2022

U0000011047

## Objectives

1. Combine all the concepts learned during the semester to implement a real-time digital stopwatch in Verilog and programed onto an FPGA
2. Stopwatch includes a start/stop that toggles the stopwatch; must also start at 00:00 seconds

## Description

This project is a way to put an entire semester worth of knowledge into achieving a single outcome. By combining the codes used previously with the knowledge of Verilog, a clock divider and some Flip-Flops, we can create a project that successfully outputs a stopwatch on an FPGA.

## Verilog Codes

### RCA

```
module HA(X, Y, Sum, Cout);                         //Half Adder
      input X, Y;
      output Sum, Cout;

      assign Sum = X ^ Y;
      assign Cout = X & Y;
endmodule


module RCA(A, Cin, Sum, Cout);                            //RCA module
      input [3:0] A;
      input Cin;
      output [3:0] Sum;
      output Cout;
      wire c0, c1, c2;

      HA bit0(A[0], Cin, Sum[0], c0);
      HA bit1(A[1], c0, Sum[1], c1);
      HA bit2(A[2], c1, Sum[2], c2);
      HA bit3(A[3], c2, Sum[3], Cout);
endmodule
```

### BCD_Display

```
module BCD_Display(A, B, C, D, a1, b1, c1, d1, e1, f1, g1);

      input A, B, C, D;
      output a1, b1, c1, d1, e1, f1, g1;

      assign a1 = ((B & ~C & ~D) | (~A & ~B & ~C & D));
      assign b1 = ((B & ~C & D) | (B & C & ~D));
      assign c1 = (~B & C & ~D);
      assign d1 = ((B & ~C & ~D) | (~B & ~C & D) | (B & C & D));
      assign e1 = (D | (B & ~C));
      assign f1 = ((C & D) | (~A & ~B & D) | (~B & C));
      assign g1 = ((B & C & D) | (~A & ~B & ~C));
endmodule
```

### count10

```
module count10(clock, inc, reset, Count, count_eq_9);

      input clock, inc, reset;
      output [3:0] Count;
      output count_eq_9;
      wire Null, rst;
      wire [3:0] w0;
```

```
	assign count_eq_9 = (Count == 4'b1001)? 1:0;

	assign rst = ((count_eq_9 & inc) | reset);

	DFF4 Dff(w0, clock, rst, Count);

	RCA rca(Count, inc, w0, Null);
endmodule
```

**count6**

```
module count6(clock, inc, reset, Count, count_eq_5);

	input clock, inc, reset;
	output [3:0] Count;
	output count_eq_5;
	wire Null, rst;
	wire [3:0] w0;


	assign count_eq_5 = (Count == 4'b0101)? 1:0;

	assign rst = ((count_eq_5 & inc) | reset);


	DFF4 Dff(w0, clock, rst, Count);

	RCA rca(Count, inc, w0, Null);
endmodule
```

**Complete Stopwatch** *Entire code written together is on last pages

```
module Stopwatch(clock, reset, start_stop, a1, b1, c1, d1, e1, f1, g1, a2,
b2, c2, d2, e2, f2, g2, a3, b3, c3, d3, e3, f3, g3, a4, b4, c4, d4, e4, f4,
g4);
	input clock, reset, start_stop;
	output a1, b1, c1, d1, e1, f1, g1, a2, b2, c2, d2, e2, f2, g2, a3, b3,
c3, d3, e3, f3, g3, a4, b4, c4, d4, e4, f4, g4;
	wire clk_out, q_out, an1, an2, an3;
	wire [3:0] out0;
	wire [3:0] out1;
	wire [3:0] out2;
	wire [3:0] out3;

	//clk_divider(clock, rst, clk_out);
	//TFF0(data, clk, reset, q);

	TFF0 toggle(1, start_stop, 0, q_out);

	clk_divider clk(clock, 0, clk_out);

	//count10(clock, inc, reset, Count, count_eq_9);
	//count6(clock, inc, reset, Count, count_eq_5);
	count10 bit0(clk_out, q_out, reset, out0, count_eq_9_0);
	assign an1 = (q_out & count_eq_9_0);
	count10 bit1(clk_out, an1, reset, out1, count_eq_9_1);
	assign an2 = (an1 & count_eq_9_1);
	count10 bit2(clk_out, an2, reset, out2, count_eq_9_2);
	assign an3 = (an2 & count_eq_9_2);
```

```
      count6 bit3(clk_out, an3, reset, out3, count_eq_5);

      BCD_Display dis0(out0[3], out0[2], out0[1], out0[0], a1, b1, c1, d1,
e1, f1, g1);
      BCD_Display dis1(out1[3], out1[2], out1[1], out1[0], a2, b2, c2, d2,
e2, f2, g2);
      BCD_Display dis2(out2[3], out2[2], out2[1], out2[0], a3, b3, c3, d3,
e3, f3, g3);
      BCD_Display dis3(out3[3], out3[2], out3[1], out3[0], a4, b4, c4, d4,
e4, f4, g4);

endmodule
```

## Conclusion

I was able to successfully implement the entire code and get the intended, working stopwatch. This project was a struggle at times, while scanning through my code looking for the small little details that were causing problems, but overall it was interesting to work on.

**Entire Code**

```
module HA(X, Y, Sum, Cout);                         //Half Adder
      input X, Y;
      output Sum, Cout;

      assign Sum = X ^ Y;
      assign Cout = X & Y;
endmodule


module RCA(A, Cin, Sum, Cout);                            //RCA module
      input [3:0] A;
      input Cin;
      output [3:0] Sum;
      output Cout;
      wire c0, c1, c2;

      HA bit0(A[0], Cin, Sum[0], c0);
      HA bit1(A[1], c0, Sum[1], c1);
      HA bit2(A[2], c1, Sum[2], c2);
      HA bit3(A[3], c2, Sum[3], Cout);
endmodule



module BCD_Display(A, B, C, D, a1, b1, c1, d1, e1, f1, g1);
      //BCD Display

            input A, B, C, D;
            output a1, b1, c1, d1, e1, f1, g1;

            assign a1 = ((B & ~C & ~D) | (~A & ~B & ~C & D));
            assign b1 = ((B & ~C & D) | (B & C & ~D));
            assign c1 = (~B & C & ~D);
            assign d1 = ((B & ~C & ~D) | (~B & ~C & D) | (B & C & D));
            assign e1 = (D | (B & ~C));
            assign f1 = ((C & D) | (~A & ~B & D) | (~B & C));
            assign g1 = ((B & C & D) | (~A & ~B & ~C));
endmodule



module DFF0(data_in,clock,reset, data_out);                                 //D
Flip Flop
input data_in;
input clock,reset;

output reg data_out;

always@(posedge clock)
      begin
            if(reset)
                  data_out<=1'b0;
            else
                  data_out<=data_in;
      end

endmodule
```

```
//TFF0(data, clk, reset, q)
module TFF0 (
data  , // Data Input
clk   , // Clock Input
reset , // Reset input
q       // Q output
);
//-----------Input Ports---------------
input data, clk, reset ;
//-----------Output Ports---------------
output q;
//------------Internal Variables--------
reg q;
//-------------Code Starts Here---------
always @ ( posedge clk or posedge reset)
if (reset) begin
  q <= 1'b0;
end else if (data) begin
  q <= !q;
end
endmodule



module clk_divider(clock, rst, clk_in);
      input clock, rst;
      output clk_in;

      wire [18:0] din;
      wire [18:0] clkdiv;

DFF0 dff_inst0(
    .data_in(din[0]),
       .clock(clock),
       .reset(rst),
    .data_out(clkdiv[0])
);

genvar i;
generate
for (i = 1; i < 19; i=i+1)
      begin : dff_gen_label
             DFF0 dff_inst (
                    .data_in (din[i]),
                    .clock(clkdiv[i-1]),
                    .reset(rst),
                    .data_out(clkdiv[i])
             );
             end
endgenerate

assign din = ~clkdiv;

assign clk_in = clkdiv[18];

endmodule




module DFF4(in, clk, reset, out);
      //DFF Register
            input [3:0] in;
            input clk, reset;
```

```
            output [3:0] out;

            DFF0 DFF_0(in[0], clk, reset, out[0]);
            DFF0 DFF_1(in[1], clk, reset, out[1]);
            DFF0 DFF_2(in[2], clk, reset, out[2]);
            DFF0 DFF_3(in[3], clk, reset, out[3]);

endmodule



module count10(clock, inc, reset, Count, count_eq_9);
      //Count10 module
      input clock, inc, reset;
      output [3:0] Count;
      output count_eq_9;
      wire Null, rst;
      wire [3:0] w0;


      assign count_eq_9 = (Count == 4'b1001)? 1:0;

      assign rst = ((count_eq_9 & inc) | reset);
      //reset for DFF4 using built in and external input

      //DFF4(in, clk, reset, out);
      DFF4 Dff(w0, clock, rst, Count);

      //RCA(A, Cin, Sum, Cout);
      RCA rca(Count, inc, w0, Null);
endmodule



module count6(clock, inc, reset, Count, count_eq_5);
      //Count10 module
      input clock, inc, reset;
      output [3:0] Count;
      output count_eq_5;
      wire Null, rst;
      wire [3:0] w0;


      assign count_eq_5 = (Count == 4'b0101)? 1:0;

      assign rst = ((count_eq_5 & inc) | reset);
      //reset for DFF4 using built in and external input

      //DFF4(in, clk, reset, out);
      DFF4 Dff(w0, clock, rst, Count);

      //RCA(A, Cin, Sum, Cout);
      RCA rca(Count, inc, w0, Null);
endmodule



module Stopwatch(clock, reset, start_stop, a1, b1, c1, d1, e1, f1, g1, a2,
b2, c2, d2, e2, f2, g2, a3, b3, c3, d3, e3, f3, g3, a4, b4, c4, d4, e4, f4,
g4);
      input clock, reset, start_stop;
      output a1, b1, c1, d1, e1, f1, g1, a2, b2, c2, d2, e2, f2, g2, a3, b3,
c3, d3, e3, f3, g3, a4, b4, c4, d4, e4, f4, g4;
```

```
      wire clk_out, q_out, an1, an2, an3;
      wire [3:0] out0;
      wire [3:0] out1;
      wire [3:0] out2;
      wire [3:0] out3;

      //clk_divider(clock, rst, clk_out);
      //TFF0(data, clk, reset, q);

      TFF0 toggle(1, start_stop, 0, q_out);

      clk_divider clk(clock, 0, clk_out);

      //count10(clock, inc, reset, Count, count_eq_9);
      //count6(clock, inc, reset, Count, count_eq_5);
      count10 bit0(clk_out, q_out, reset, out0, count_eq_9_0);
      assign an1 = (q_out & count_eq_9_0);
      count10 bit1(clk_out, an1, reset, out1, count_eq_9_1);
      assign an2 = (an1 & count_eq_9_1);
      count10 bit2(clk_out, an2, reset, out2, count_eq_9_2);
      assign an3 = (an2 & count_eq_9_2);
      count6 bit3(clk_out, an3, reset, out3, count_eq_5);

      BCD_Display dis0(out0[3], out0[2], out0[1], out0[0], a1, b1, c1, d1,
e1, f1, g1);
      BCD_Display dis1(out1[3], out1[2], out1[1], out1[0], a2, b2, c2, d2,
e2, f2, g2);
      BCD_Display dis2(out2[3], out2[2], out2[1], out2[0], a3, b3, c3, d3,
e3, f3, g3);
      BCD_Display dis3(out3[3], out3[2], out3[1], out3[0], a4, b4, c4, d4,
e4, f4, g4);

endmodule
```