FINAL PROJECT:
REAL-TIME DIGITAL STOPWATCH DESIGN AND FPGA IMPLEMENTATION

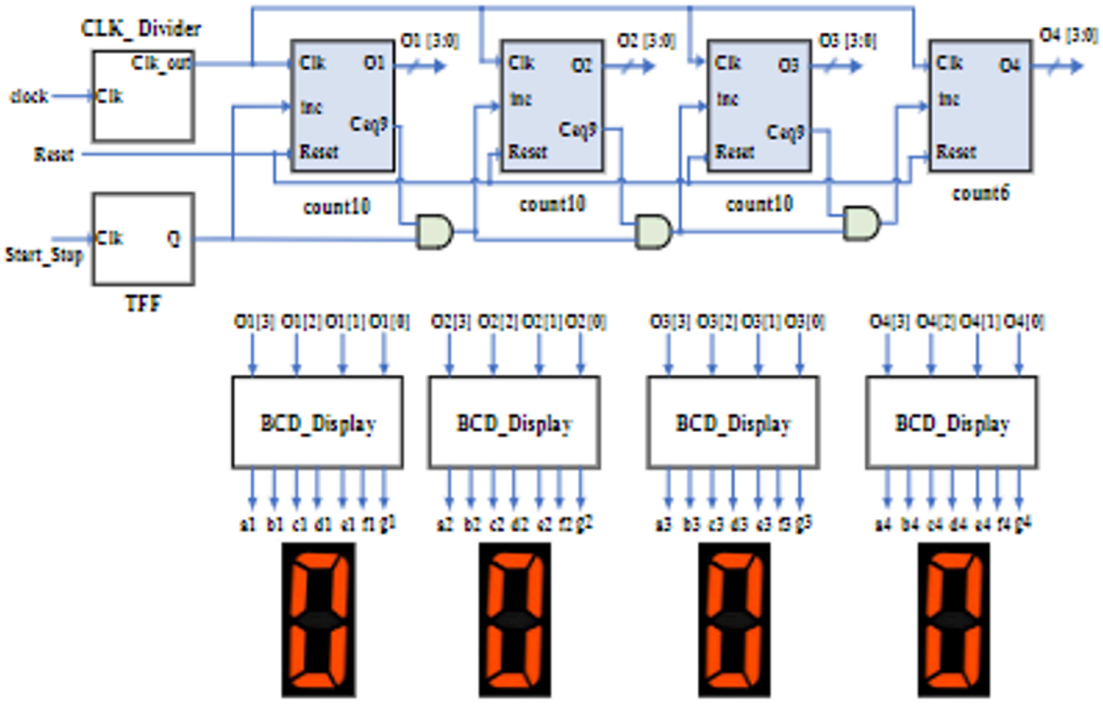
Author: Paul Patullo (ID: 1216)
Florida Polytechnic University
Digital Logic Design Lab (Ist-1058) Section 1
April 24, 2023

B. In this Project, I will be providing the Verilog code that allows a stop watch to start, stop and reset from 0- 59.99 seconds.

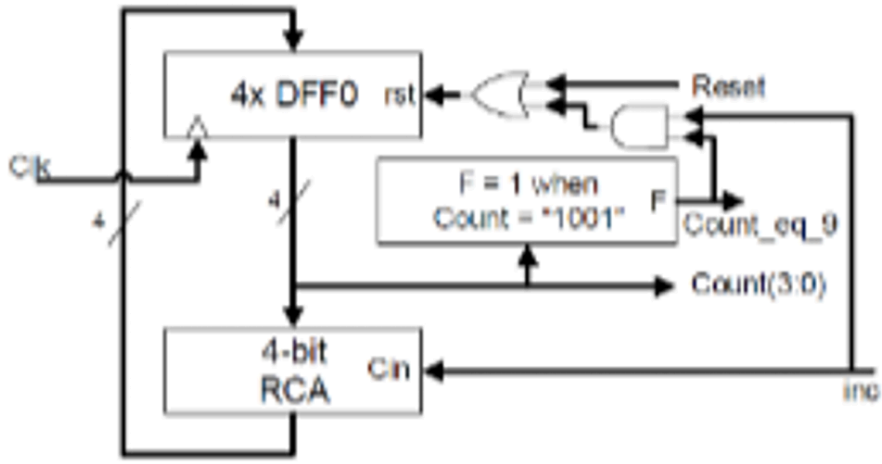
Experimental Equipment

Verilog HDL, Intel Quartus Prime FPGA edition

Fig. 1. Stopwatch Block Diagram

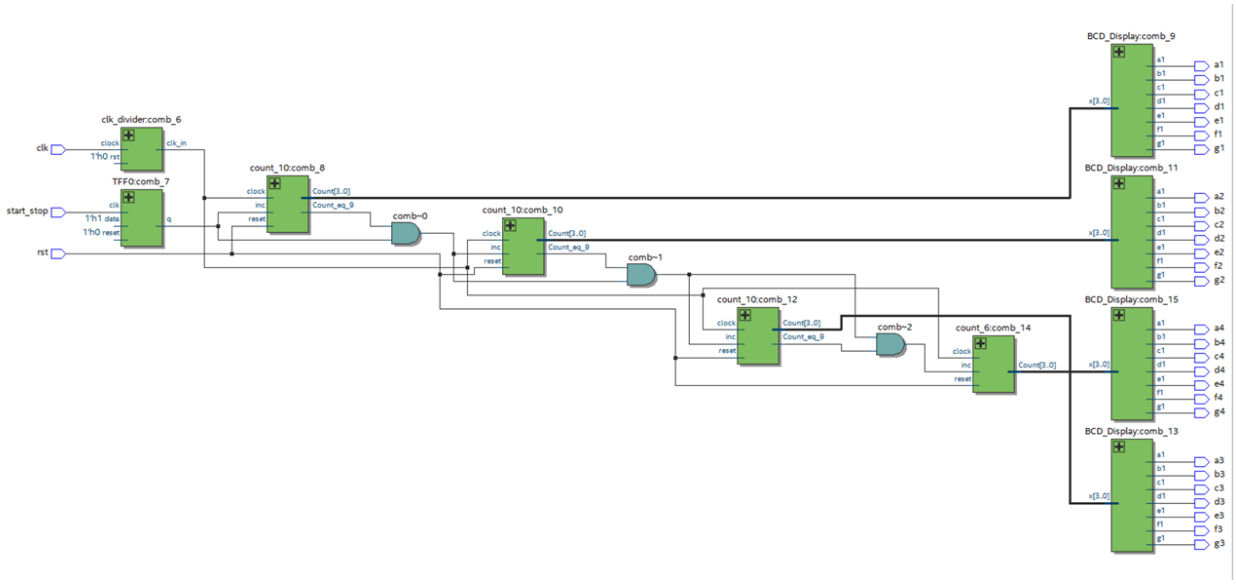


Include the **count10** module code in your report.



Count 10 diagram^^

Final RTL Viewer:



D. **PROJECTPROGRAM:**

//TFlipFlop

```
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;  // Reset Q to 0 when reset
        end else if (data) begin
                 q <= !q;  // Toggle Q on rising edge of clock if data input is high
End

Endmodule
```

//DFlipFlop

```
module DFF0(data_in,clock,reset, data_out);
input data_in;
input clock,reset;
output reg data_out;
always@(posedge clock)
        Begin
                if(reset)
                        data_out<=1'b0; // Reset output to 0 when reset signal
                Else
                        data_out<=data_in;
end
Endmodule
```

//Clock Divider

```
data_out<=data_in;
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; // Complementing the clock divider output

assign clk_in = clkdiv[18]; // Output the last bit of the clock divider as the clock input

Endmodule
```

//Create Half Adder

```
module Half_Adder(x, y, Sum, Carry);
input x, y;
output Sum, Carry;
xor(Sum, x, y); // Sum is XOR of inputs
and(Carry, x, y); // Carry is AND of inputs
Endmodule
```

//Create RCA Module using HA

```
module RCA(x, y, output1);
input [3:0] x;
input y;
output [3:0]output1;

wire s1, s2, s3, s4;

Half_Adder H_Adder1(x[0], y, output1[0], s1);
Half_Adder H_Adder2(x[1], s1, output1[1], s2);
Half_Adder H_Adder3(x[2], s2, output1[2], s3);
Half_Adder H_Adder4(x[3], s3, output1[3], s4);
Endmodule
```

// Create 4bit DFF Module(DFF4)

```
module DFF4(inp, rst, clk, out);
input [3:0]inp;
input rst, clk;
output [3:0]out;

DFF0 fourBit1(inp[0],clk,rst, out[0]);
DFF0 fourBit2(inp[1],clk,rst, out[1]);
DFF0 fourBit3(inp[2],clk,rst, out[2]);
DFF0 fourBit4(inp[3],clk,rst, out[3]);
Endmodule
```

//Create Count 10 module using RCA and DFF4 Component

```
module count_10(clock, inc, reset, Count_eq_9, Count);
input clock, inc, reset;
output Count_eq_9;
output [3:0]Count;
wire [3:0]a3;
wire a1;
wire RorI;
wire F;
assign Count_eq_9 = (Count == 4'b1001)? 1: 0; // Set Count_eq_9 high if Count is 9
assign F = Count_eq_9;   // Set F high if Count_eq_9 is high
and(RorI, inc, F);  // Perform AND operation between inc and F
or(a1, reset, RorI); // Perform OR operation between reset and RorI
DFF4 FourDFF(a3, a1, clock, Count);
RCA RCABit(Count,inc,a3);
Endmodule
```

//Count 6 display

```
module count_6(clock, inc, reset, Count_eq_9, Count);
input clock, inc, reset;
output Count_eq_9;
output [3:0]Count;

wire [3:0]a3;
wire a1;
wire RorI;
wire F;
assign Count_eq_9 = (Count == 4'b0101)? 1: 0; // Set Count_eq_9 high if Count is 5
assign F = Count_eq_9;  // Set F high if Count_eq_9 is high

and(RorI, inc, F);
or(a1, reset, RorI);
DFF4 FourDFF(a3, a1, clock, Count);
RCA RCABit(Count,inc,a3);
Endmodule
```

// Create BCD_DISPLAY

```
module BCD_Display(x, a1, b1, c1, d1, e1, f1, g1);
input [3:0] x;
output a1, b1, c1, d1, e1, f1, g1;
assign a = x[3]; // Assign the BCD segments
assign b = x[2];
assign c = x[1];
```

```
assign d = x[0];
                                                    // BCD to 7-segment conversion
assign a1 = (b & ~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 = (b & ~c) | d;
assign f1 = (~b & c) | (~a & ~b & d) | (c & d);
assign g1 = (~a & ~b & ~c) | (b & c & d);
Endmodule
```

// Create the final Countermodule using clock divider, BCD_display, and Count10 Module

```
module SingleDisplay(clock, sw0, sw1, a1, b1, c1, d1, e1, f1, g1);
input clock, sw0, sw1;
output a1, b1, c1, d1, e1, f1, g1;

wire x1;
wire [3:0]q1;

clk_divider(clock, sw0, x1);
count_10(x1, sw1, sw0, Count_eq_9, q1);
BCD_Display(q1, a1, b1, c1, d1, e1, f1, g1);

Endmodule

module StopWatchDisplay(clk, rst, 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 clk, rst, start_stop;
output a1, b1, c1, d1, e1, f1, g1,          // 7-segment display outputs for first digit
a2, b2, c2, d2, e2, f2, g2,                 // 7-segment display outputs for second digit
a3, b3, c3, d3, e3, f3, g3,                 // 7-segment display outputs for third digit
a4, b4, c4, d4, e4, f4, g4;                 // 7-segment display outputs for fourth digit

wire x1;          // Clock divider output
wire [3:0]ceqOut; // Count_eq_9 for each digit
wire [3:0]o1;    // Output for first digit
wire [3:0]o2;    // Output for second digit
wire [3:0]o3;    // Output for third digit
wire [3:0]o4;    // Output for fourth digit
```

```
wire [3:0]Count_eq_9;              // Count_eq_9 for each digit
and(ceqOut[1], Count_eq_9[0], ceqOut[0]);       // Generate Count_eq_9 for second digit
and(ceqOut[2], Count_eq_9[1],ceqOut[1]);
and(ceqOut[3], Count_eq_9[2],ceqOut[2]);

clk_divider(clk, 0, x1);

TFF0 (1, start_stop, 0,ceqOut[0]);                  // Toggle flip-flop

count_10(x1, ceqOut[0], rst, Count_eq_9[0], o1);          // Count to 9 for first digit
BCD_Display(o1, a1, b1, c1, d1, e1, f1, g1);              // Display first digit

count_10(x1, ceqOut[1], rst, Count_eq_9[1], o2);          // Count to 9 for second digit
BCD_Display(o2, a2, b2, c2, d2, e2, f2, g2);              // Display second digit

count_10(x1, ceqOut[2], rst, Count_eq_9[2], o3);          // Count to 9 for third digit
BCD_Display(o3, a3, b3, c3, d3, e3, f3, g3);              // Display third digit

count_6(x1, ceqOut[3], rst, Count_eq_9[3], o4);           // Count to 5 for fourth digit
BCD_Display(o4, a4, b4, c4, d4, e4, f4, g4);              // Display fourth digit

Endmodule
```

E. I learned programming the functions of electronic components is broken down into functions that work hand in hand together which becomes an achievable goal. While programming one function, and incorporating that with another, one missed semicolon causes the entirety of the program to crash without notice. Having a design plan allows for the structure to come into place while tackling the challenge is manageable. I did enjoy this project very much having combined hardware and software components to seamlessly operate.

Conclusion:

F. We used Quartus to break down the stopwatch module into diagrams and from there, we made each component from small to large and kept on linking the parts together until our final goal was achieved. It did take time outside of class to work on, but the sight of progress being made was a great sign of relief knowing that the project was coming together.