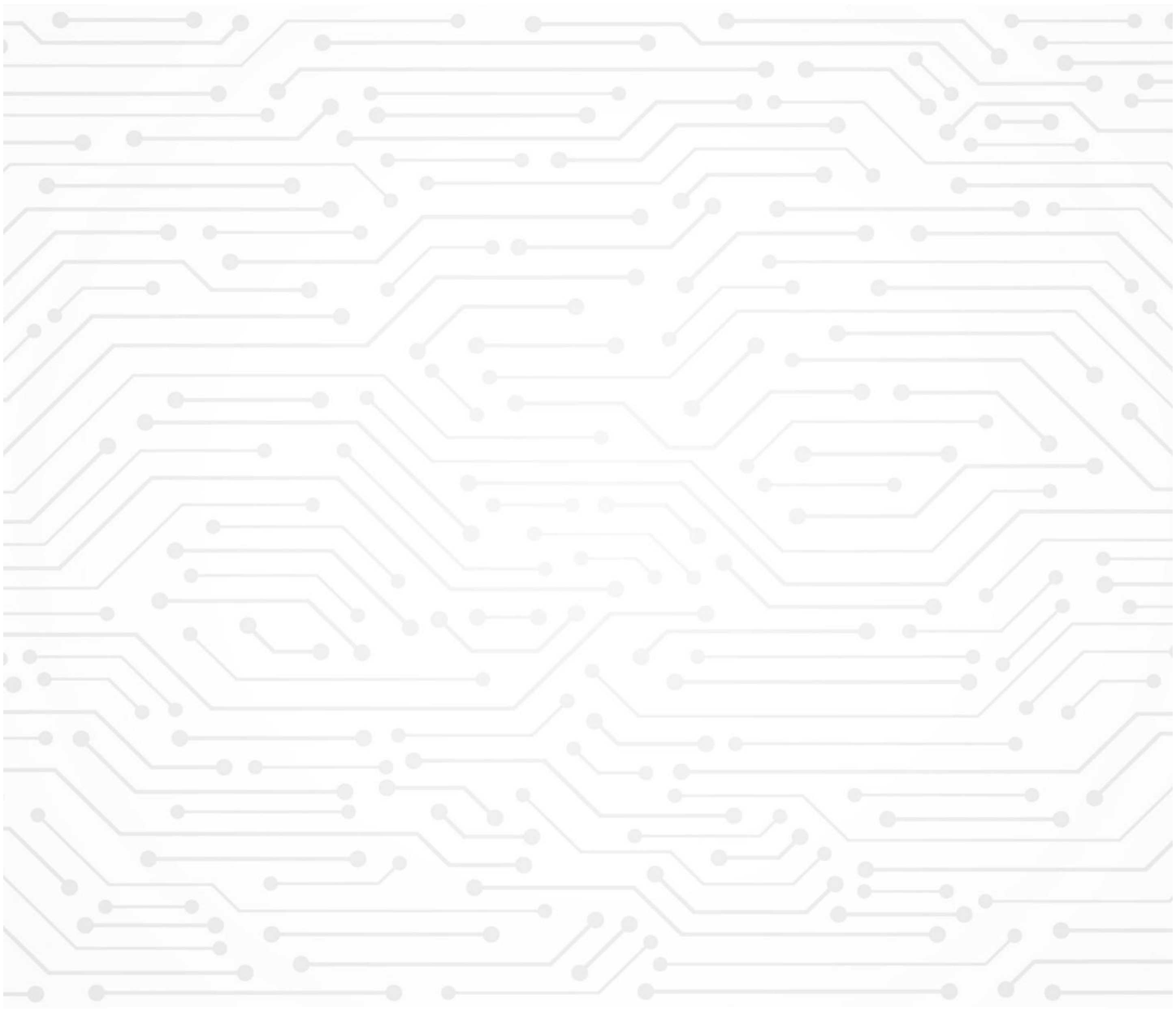

FINAL PROJECT: REAL TIME DIGITAL STOPWATCH DESIGN AND FPGA IMPLEMENTATION

Real time Digital stopwatch that has a Start/Stop button and a reset button with the application of a ripple carry adder using half adders, D-FFs, a clock divider, a BCD to 7-segment

PROJECT

Th ReaTim DigitaStopwatc implementatiowillhaveallthe conceptw learnethroughout the semestecombined

W willb abl t fus togetheth theoreticaknowledgthatw learnein clasan th practicaimplementatiothat w constructeinth labt buildsomethin withou ow knowledgandlabor.

I wa abl t accomplishi projecan gainemor expertisinDigitaLogi Designs

a) Project Description:

In this project, I will be designing and implementing a stopwatch using Boolean Expression to specify the circuit implementing Verilog in Questa and checking that the code works in an FPGA board.

I designed some of the components for the stopwatch with sequential logic using flip flops, ripple carry adder, clock divider, among others. The FPGA board should display a counter from 0 to 9 and then reset back to 0 and repeat on a loop until either the start switch if off or the reset switch is on (on this part of the lab, the reset button will only pause the count).

b) Complete Verilog code of the following module: RCA, BCD_Display, count10, and count6

```
 //RCA code
module RCA(A,Cin,Sum,Cout);
//Input ports
input [3:0]A;
input Cin;
//Output port
output [3:0]Sum;
output Cout;
//Wires wire
[2:0]C;
//Expression
HalfAdder HA1(.A (A[0]), .B (Cin), .Sum (Sum[0]), .Carry(C[0]));
HalfAdder HA2(.A (A[1]), .B (C[0]), .Sum (Sum[1]), .Carry(C[1]));
HalfAdder HA3(.A (A[2]), .B (C[1]), .Sum (Sum[2]), .Carry(C[2]));
HalfAdder HA4(.A (A[3]), .B (C[2]), .Sum (Sum[3]), .Carry(Cout));
Endmodule

//BCD_Display
module BCDto7seg(A ,B, C, D, a, b, c, d, e, f, g);
//Input ports
input A; input
B; input C;
input D;
//Output port
output a;
output b;
output c;
output d;
output e;
output f;
output g;
//Expression
assign a =
~A&~C&(B^D)
; assign b =
B&(C^D);
assign c =
C&~D&~B;
assign d = ~A&(B|D)&(~C|D)&(B|~C)&(~B|C|~D);
assign e = (B|D)&(~C|D); assign f =
~A&(~B|C)&(~B|D)&(C|D); assign g =
~A&~(B^C)&(~B|D);
endmodule

//count 10
module Count10(clk, inc, rst, Count, count_eq_9);
input clk; input
rst;//global
input inc;

output [3:0]Count;
output count_eq_9;

wire dummy; wire
[3:0]RCAout; wire
a;
wire reset;

assign count_eq_9 = (Count == 4'b1001)? 1 : 0; assign a =
count_eq_9&inc;
assign reset = a|rst;

RCA RCA1(.A (Count), .Cin (inc), .Sum (RCAout), .Cout(dummy));
DFF4 DF1(.A (RCAout), .clock (clk), .reset (reset), .Count(Count));

endmodule

//count 6
module Count6(clk, inc, rst, Count);
input clk; input
rst;//global input
inc;
//assign rs;

output [3:0]Count;

wire [3:0]RCAout;
wire a; wire and1;
wire reset;
wire dummy;

assign F = (Count == 4'b0101)? 1 : 0; assign
and1 = F&inc;
assign reset = and1|rst;

RCA RCA1(.A (Count), .Cin (inc), .Sum (RCAout), .Cout(dummy));
DFF4 DF1(.A (RCAout), .clock (clk), .reset (reset), .Count(Count));

endmodule
```

c) Verilog code of the complete STOWATCH module

```
module Counter(clock, reset, start, 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; input
reset; input start;
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; wire [3:0]O1; wire [3:0]O2; wire [3:0]O3; wire [3:0]O4; wire count_eq_91;//ask wire
count_eq_92; wire count_eq_93;
wire q; wire
and1; wire
and2; wire
and3;

TFF0 TFF1(.data(1), .clk(start), .reset(0), .q(q));

clk_divider clk_dvider1(.clock (clock), .rst (0), .clk_out (clk_out));

Count10 Count101(.clk (clk_out), .inc (q), .rst (reset), .Count(O1), .count_eq_9(count_eq_91));

BCDto7seg BCD27Seg1(.A (O1[3]), .B (O1[2]), .C (O1[1]), .D(O1[0]), .a(a1), .b(b1), .c(c1), .d(d1), .e(e1),
.f(f1), .g(g1));

assign and1 = count_eq_91&q;
Count10 Count102(.clk (clk_out), .inc (and1), .rst (reset), .Count(O2), .count_eq_9(count_eq_92));

BCDto7seg BCD27Seg2(.A(O2[3]), .B(O2[2]), .C(O2[1]), .D(O2[0]), .a(a2), .b(b2), .c(c2), .d(d2), .e(e2),
.f(f2), .g(g2));

assign and2 = count_eq_92&and1;
Count10 Count103(.clk (clk_out), .inc (and2), .rst (reset), .Count(O3), .count_eq_9(count_eq_93));

BCDto7seg BCD27Seg3(.A(O3[3]), .B(O3[2]), .C(O3[1]), .D(O3[0]), .a(a3), .b(b3), .c(c3), .d(d3), .e(e3),
.f(f3), .g(g3));

assign and3 = count_eq_93&and2;
Count6 Count64(.clk (clk_out), .inc (and3), .rst (reset), .Count(O4));

BCDto7seg BCD27Seg4(.A (O4[3]), .B (O4[2]), .C (O4[1]), .D(O4[0]), .a(a4), .b(b4), .c(c4), .d(d4), .e(e4),
.f(f4), .g(g4));


endmodule
```

c) Project comment

In this project we can appreciate the implementation of, if not all, most of the material that we went over in the course. From the base, we used digital and number systems for the counting of the clock. Next, we used Boolean algebra to get the minimizations of the logic behind the BCD to 7-segment converter. We also used logic gates to tie in the Boolean functions that we found from the k-maps. Finally, we designed a sequential circuit with the use of all above mentioned plus sequential MSI logic circuit and synchronous state machines. I have learned to design combinational circuits using Boolean algebra to minimize and optimize them. We simulated and tested sequential digital circuits and now I understand the relationship between abstract logic characterizations and practical electrical implementations.