ELC 2137 Lab 7: Binary Coded Decimal

Spencer Stinson

October 14, 2020

Summary

In this lab experiment, we learned how to use the double dabble method to convert from hex to BCD values. This method can be implemented by using multiple instances of adders. To finish this lab, we had to implement this 11 bit BCD decoder, as well as our 7 segment design fro mthe previous lab.

Code

Listing 1: add3 code

```
‘timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/08/2020 11:42:33 AM
// Design Name:
// Module Name: add3
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module add3(
    input [3:0] num,
    output reg [3:0] numout
    );
```

```
    always @*
    if (num > 4)
        numout = num + 4'b0011 ;
    else
       numout = num ;

endmodule
```

Listing 2: double dabble 6 bit code

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/08/2020 12:26:19 PM
// Design Name:
// Module Name: dd6b
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module dd6b(
    input [5:0] B,
    output [3:0] tens,
    output [3:0] ones
    );

    wire [2:0] c1_out;
    wire [2:0] c2_out;

    assign tens[3] = 1'b0 ;

    add3 C1 (
    .num({1'b0 , B[5:3]}),
    .numout({tens[2], c1_out[2:0]})
    );

    add3 C2(
    .num({c1_out[2:0], B[2]}),
```

```
    .numout({tens[1], c2_out[2:0]})
    );

     add3 C3(
    .num({c2_out[2:0], B[1]}),
    .numout({tens[0], ones[3:1]})
    );

    assign ones[0] = B[0];
endmodule
```

Listing 3: double dabble 11 bit code

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/14/2020 05:01:45 PM
// Design Name:
// Module Name: dd11b
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module dd11b(
    input [10:0] B,
    output [3:0] ones,
    output [3:0] tens,
    output [3:0] hundreds,
    output [3:0] thousands
    );
    wire [3:0] c1o;
    wire [3:0] c2o;
    wire [3:0] c3o;
    wire [3:0] c4o;
    wire [3:0] c5o;
    wire [3:0] c6o;
    wire [3:0] c7o;
    wire [3:0] c9o;
```

```
    wire [3:0] c10o;
    wire [3:0] c11o;
    wire [3:0] c12o;
    wire [3:0] c14o;

     add3 C1 (
    .num({1'b0 , B[10:8]}),
    .numout(c1o[3:0])
    );

    add3 C2(
    .num({c1o[2:0], B[7]}),
    .numout(c2o [3:0])
    );

     add3 C3(
    .num({c2o[2:0], B[6]}),
    .numout(c3o[3:0])
    );

    add3 C4 (
    .num({c3o[2:0] , B[5]}),
    .numout(c4o[3:0])
    );

    add3 C5(
    .num({c4o[2:0], B[4]}),
    .numout(c5o[3:0])
    );

     add3 C6(
    .num({c5o[2:0], B[3]}),
    .numout(c6o[3:0])
    );

     add3 C7 (
    .num({c6o[2:0] , B[2]}),
    .numout(c7o[3:0])
    );

    add3 C8(
    .num({c7o[2:0], B[1]}),
    .numout({tens[0], ones[3:1]})
    );

     add3 C9(
    .num({0, c1o[3], c2o[3], c3o[3]}),
    .numout(c9o[3:0])
    );

     add3 C10(
    .num({c9o[2:0], c4o[3]}),
    .numout(c10o[3:0])
    );
```

```
    add3 C11(
    .num({c10o[2:0], c5o[3]}),
    .numout(c11o[3:0])
    );

    add3 C12(
    .num({c11o[2:0], c6o[3]}),
    .numout(c12o[3:0])
    );

     add3 C13(
    .num({c12o[2:0], c7o[3]}),
    .numout({hundreds[0], tens[3:1]})
    );

    add3 C14(
    .num({1'b0, c9o[3], c10o[3], c11o[3]}),
    .numout({thousands[1], c14o[2:0]})
    );

    add3 C15(
    .num({c14o[2:0], c12o[3]}),
    .numout({thousands[0],hundreds[3:1]})
    );

    assign ones[0] = B[0];
    assign thousands[3:2]= 2'b00;
endmodule
```

Listing 4: 7 segment BCD code

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/14/2020 08:30:57 PM
// Design Name:
// Module Name: sseg1_bcd
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
```

```
    //////////////////////////////////////////////////////////////////////////////////


module sseg1_bcd(
    input [15:0] sw,
    output dp,
    output [6:0] seg,
    output [3:0] an
    );
    wire [3:0] in0;
    wire [3:0] in1;
    wire [3:0] f;
    wire [3:0] g;

    wire [3:0] num;

    dd11b m11(
    .B(sw[10:0]),
    .ones(in0),
    .tens(in1),
    .hundreds(f),
    .thousands(g));

     mux4_2b mm(
     .A(in1),
     .B(in0),
     .sel(sw [15]),
     .out(num) );

    sseg_decoder ms(
    .num(num),
    .sseg(seg));

    assign an[3:2] = 2'b11;
    assign dp = 1'b1;
    assign an[0] = ~sw[15];
    assign an[1] = sw[15] ;

endmodule
```

Listing 5: testbench for add3 code

```
`timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/08/2020 11:57:16 AM
// Design Name:
// Module Name: add3_testbench
// Project Name:
```

```
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module add3_testbench();

integer  i ;

reg [3:0] num_t;
reg [3:0] numout_t;

add3 dut(
.num(num_t), .numout(numout_t)
);

initial begin
for (i=0; i <= 4'b1111; i=i+1) begin
num_t = i;
#10;
end
$finish ;
end

endmodule
```

Listing 6: testbench for double dabble 6 bit code

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/08/2020 11:57:16 AM
// Design Name:
// Module Name: add3_testbench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
```

```
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////



module dd6b_testbench_ ();
integer i;
    reg  [5:0] B_t;
    wire [3:0] tens_t;
    wire [3:0] ones_t;

dd6b dut(
.B(B_t), .tens(tens_t), .ones(ones_t)
);
initial begin
for (i=0; i <= 6'h111111; i=i+1) begin
    B_t = i;
    #10;
    end
$finish ;
end

endmodule
```

Listing 7: testbench for double dabble 11 bit

```
`timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/08/2020 11:57:16 AM
// Design Name:
// Module Name: add3_testbench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////
```

```
module dd11b_testbench();
   integer i ;

    reg [10:0] B_t;
    wire [3:0] ones_t;
    wire [3:0] tens_t;
    wire [3:0] hundreds_t;
    wire [3:0] thousands_t;

dd11b dut(
.B(B_t), .ones(ones_t), .tens(tens_t), .hundreds(hundreds_t), .thousands(
   thousands_t)
);

initial begin
for (i=0; i <= 11'b11111111111; i=i+1) begin
    B_t = i;
    #10;
    end
$finish ;
end




endmodule
```

Results

In this section, put your simulation waveforms, results tables, pictures of hardware, and any other required items.

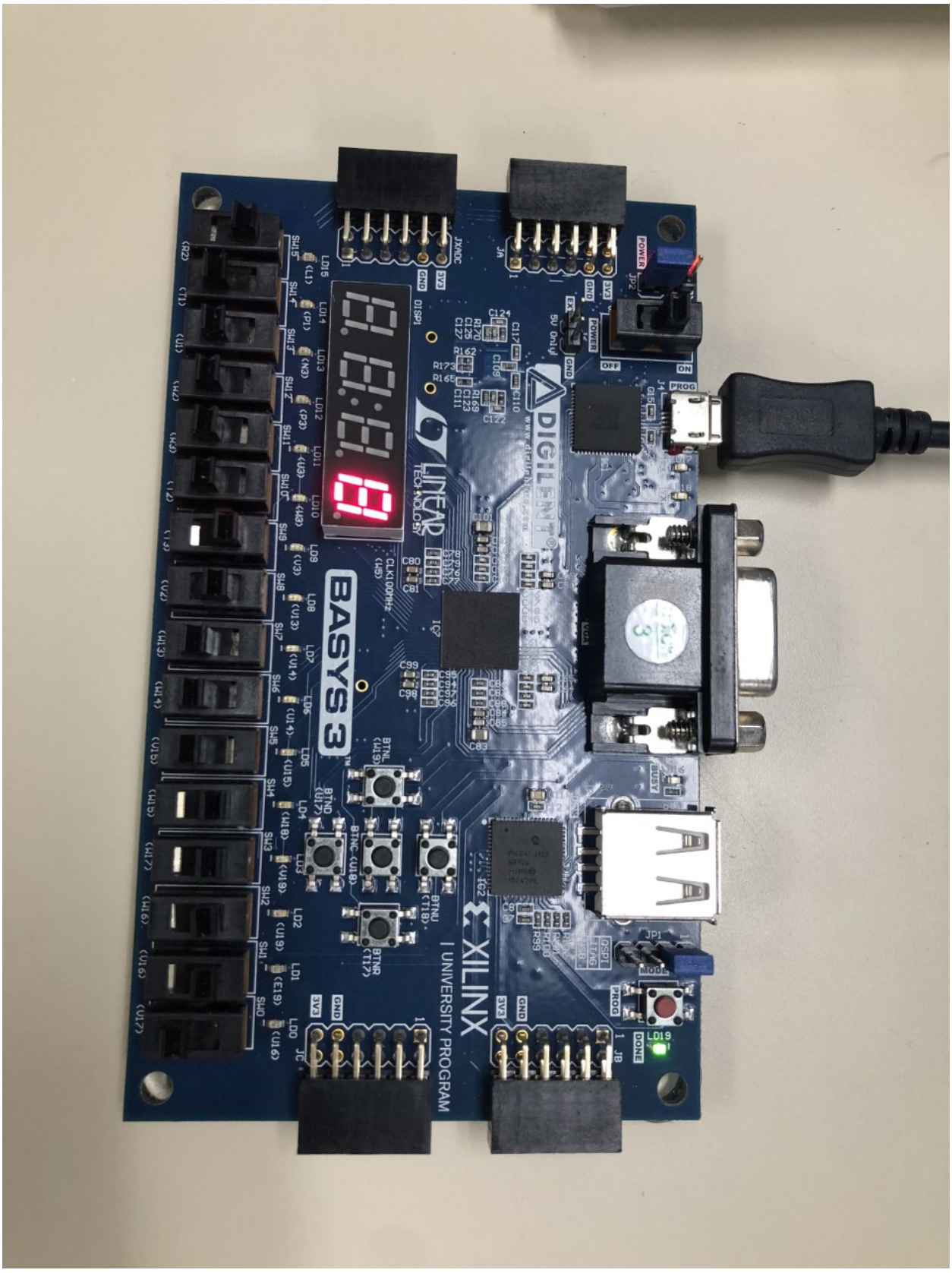


Figure 1: Basys board running 7 segment on first display

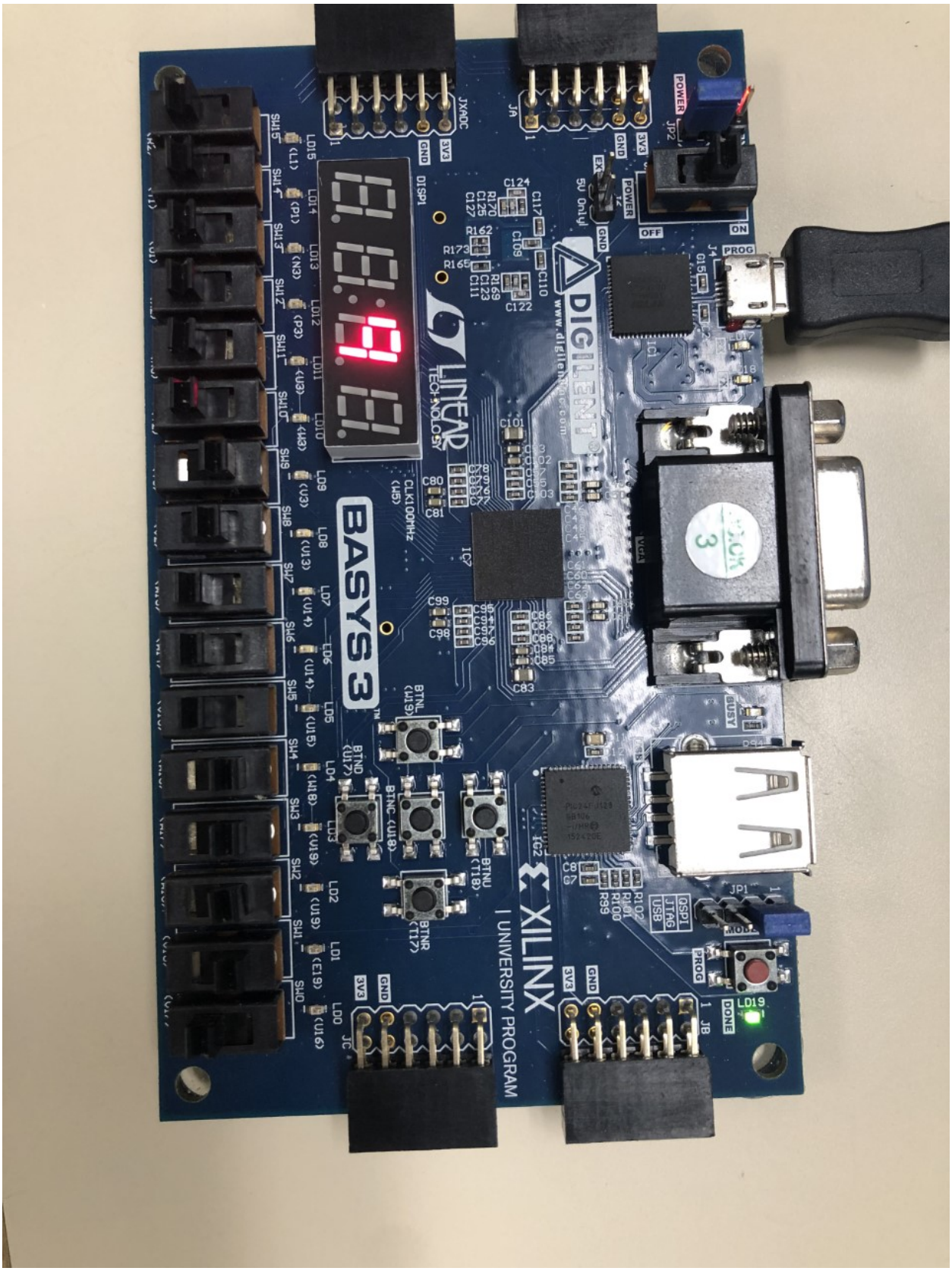


Figure 2: Basys board running 7 segment on second display

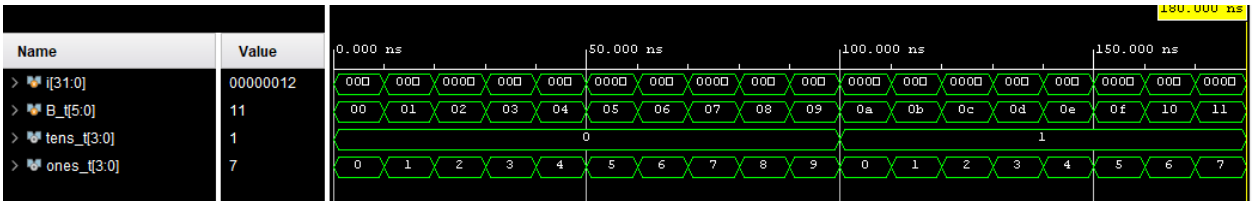


Figure 3: waveform for 6-bit

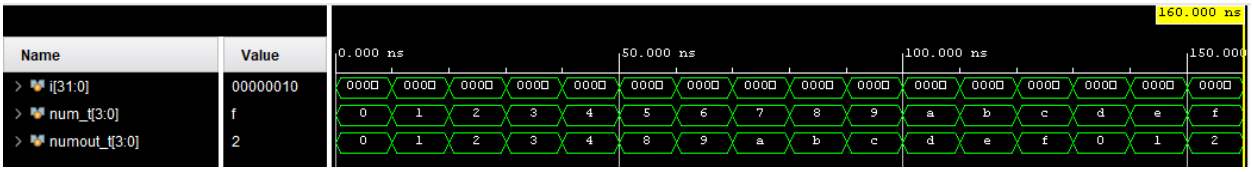


Figure 4: waveform for add3

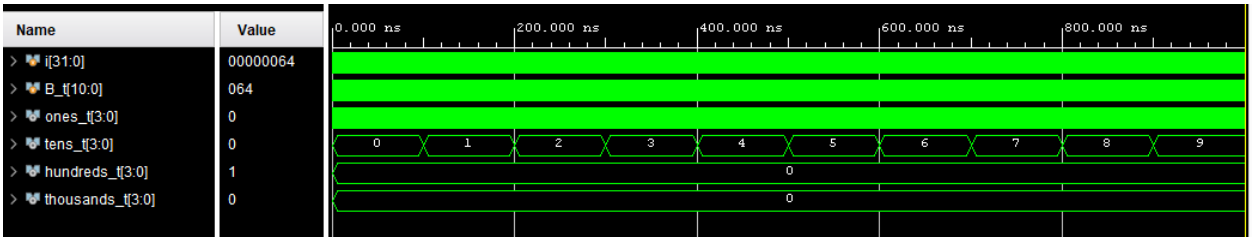
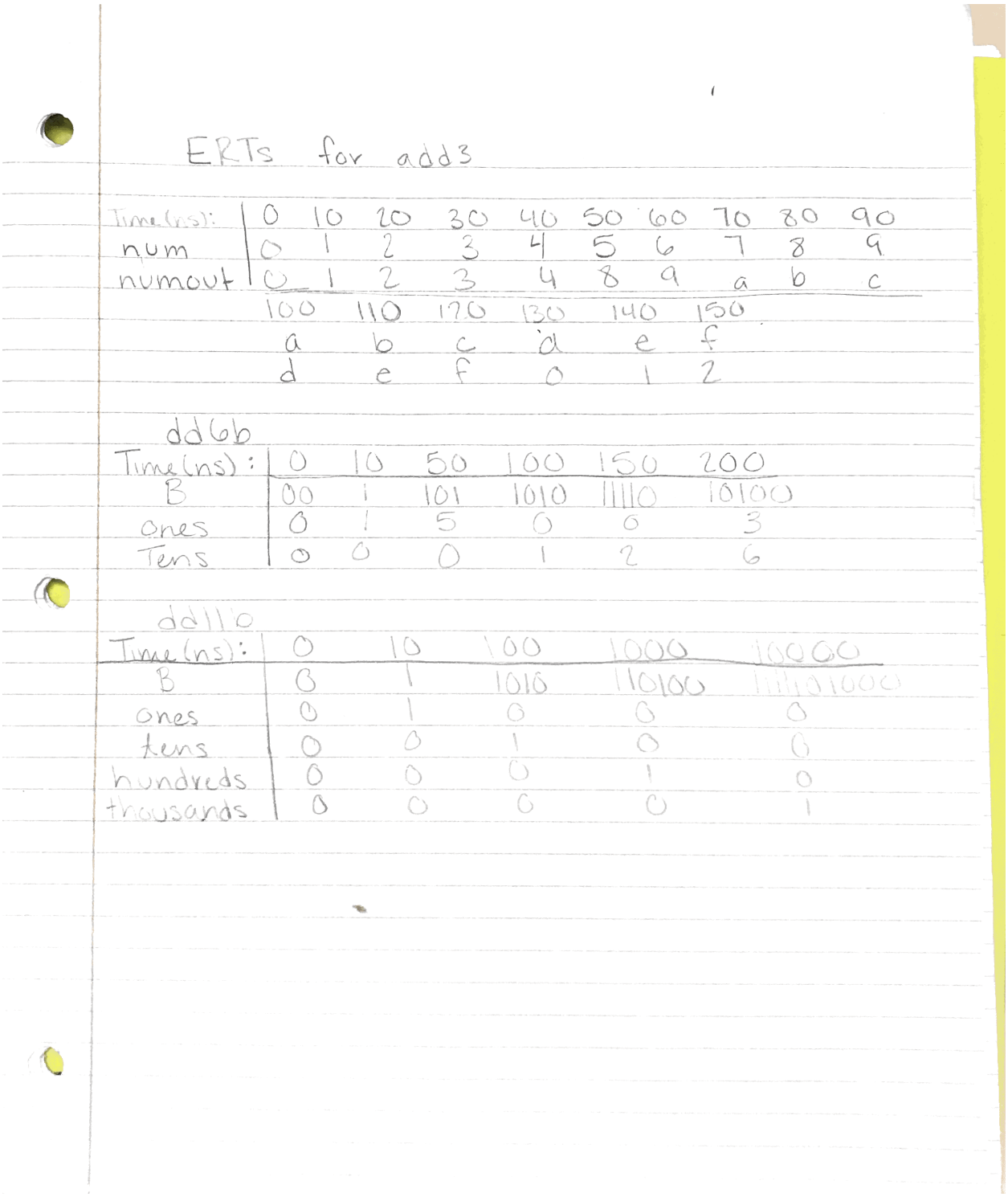


Figure 5: waveform for 11-bit



ERTs for add3

Time(ns):	0	10	20	30	40	50	60	70	80	90
num	0	1	2	3	4	5	6	7	8	9
numout	0	1	2	3	4	8	9	a	b	c

100	110	120	130	140	150
a	b	c	d	e	f
d	e	f	0	1	2

dd6b

Time(ns):	0	10	50	100	150	200
B	00	1	101	1010	11110	10100
ones	0	1	5	0	6	3
Tens	0	0	0	1	2	6

dd110

Time(ns):	0	10	100	1000	10000
B	0	1	1010	1110100	10011010000
ones	0	1	0	0	0
tens	0	0	1	0	0
hundreds	0	0	0	1	0
thousands	0	0	0	0	1

Figure 6: Estimated Results Table

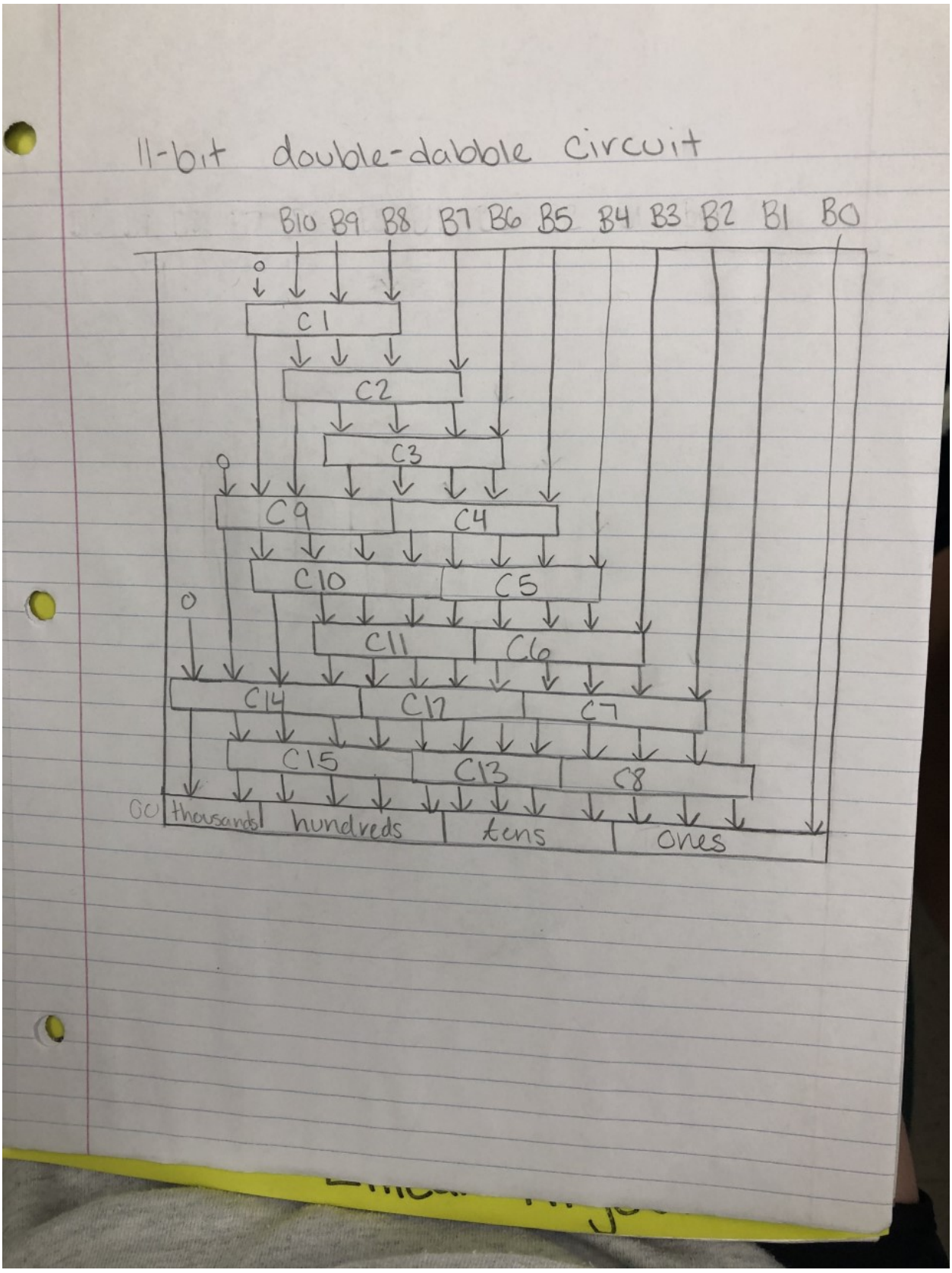


Figure 7: 11 bit circuit drawing