ELC 2137 Lab #7: Binary Coded Decimal

Sean Dickenscheidt and Sebastian Lopez

March 5, 2020

Summary

In this lab we used the double-dabble algorithm to convert hexvalues into BCD. We then explain how this algorithm can be implemented in hardware, by using structural style.

Results

Listing 1: Add3 code

```
‘timescale 1ns / 1ps
// ELC 2137 - Lab6 - 02/20/2020
// Sebastian Lopez and Sean Dickenscheidt

module add3 (
input [3:0] in,
output reg [3:0] out
);

always @*
begin
if (in >= 5)
out = in + 3;
else
out = in;
end

endmodule //add3
```

Listing 2: BCD6 code

```
‘timescale 1ns / 1ps
// ELC 2137 - Lab6 - 02/20/2020
// Sebastian Lopez and Sean Dickenscheidt

module BCD6 (
input [5:0] B,
output reg [7:0] Output6
);

wire w1, w2, w3, w4, w5, w6;
```

```
assign Output6[7] = 0;
assign Output6[0] = B[0];

add3 Bobby(
.in({1'b0, B[5:3]}),
.out({Output6[6], w1, w2, w3})
);
add3 Billy(
.in({w1, w2, w3, B[2]}),
.out({Output6[5], w4, w5, w6})
);
add3 Barron(
.in({w4, w5, w6, B[1]}),
.out({Output6[4], Output6[3], Output6[2], Output6[1]})
);

endmodule //BCD 6
```

Listing 3: BCD11 code

```
`timescale 1ns / 1ps
// ELC 2137 - Lab6 - 02/20/2020
// Sebastian Lopez and Sean Dickenscheidt

module BCD11 (
input [10:0] B,
output [13:0] Output11
);

wire w1, w2, w3, w4, w5, w6, w7, w8, w9, w10;
wire w11, w12, w13, w14, w15, w16, w17, w18, w19, w20;
wire w21, w22, w23, w24, w25, w26, w27, w28, w29, w30;
wire w31, w32, w33, w34, w35, w36, w37, w38, w39, w40;
wire w41, w42, w43, w44, w45, w46, w47;

assign Output11[0] = B[0];

add3 Bobby(
.in({1'b0, B[10:8]}),
.out({w1, w2, w3, w4})
);
add3 Billy(
.in({w2, w3, w4, B[7]}),
.out({w5, w6, w7, w8})
);
add3 Barron(
.in({w6, w7, w8, B[6]}),
.out({w9, w10, w11, w12})
);
add3 Barry(
.in({1'b0, w1, w5, w9}),
.out({w13, w14, w15, w16})
);
add3 Bart(
.in({w10, w11, w12, B[5]}),
.out({w17, w18, w19, w20})
);
add3 Blake(
.in({w14, w15, w16, w17}),
.out({w21, w22, w23, w24})
);
add3 Ben(
.in({w18, w19, w20, B[4]}),
.out({w25, w26, w27, w28})
);
add3 Brody(
.in({w22, w23, w24, w25}),
.out({w29, w30, w31, w32})
);
add3 Brett(
.in({w26, w27, w28, B[3]}),
.out({w33, w34, w35, w36})
);
add3 Sebas(
.in({1’b0, w13, w21, w29}),
.out({Output11[13], w37, w38, w39})
);
add3 Sean(
.in({w30, w31, w32, w33}),
.out({w40, w41, w42, w43})
);
add3 Steve(
.in({w34, w35, w36, B[2]}),
.out({w44, w45, w46, w47})
);
add3 Shane(
.in({w37, w38, w39, w40}),
.out({Output11[12], Output11[11], Output11[10], Output11[9]})
);
add3 Sam(
.in({w41, w42, w43, w44}),
.out({Output11[8], Output11[7], Output11[6], Output11[5]})
);
add3 Seth(
.in({w45, w46, w47, B[1]}),
.out({Output11[4], Output11[3], Output11[2], Output11[1]})
);

endmodule //BCD 11
```

Listing 4: sseg1 code

```
‘timescale 1ns / 1ps
// ELC 2137 - Lab6 - 02/20/2020
// Sebastian Lopez and Sean Dickenscheidt

module sseg1_BCD (
input [15:0] sw,
output [3:0] an,
output [6:0] seg,
output dp
);
wire [13:0] sseg1_wire;
wire [1:0] thousands;
wire [3:0] hundreds, tens, ones;
wire [3:0] out1;

BCD11 sseg1(
.B(sw[10:0]),
.Output11(sseg1_wire)
);

assign thousands = sseg1_wire[13:12];
assign hundreds = sseg1_wire[11:8];
assign tens = sseg1_wire[7:4];
assign ones = sseg1_wire[3:0];

mux2_4b Test1(
.in0(ones), .in1(tens), .sel(sw[15]),
.out(out1)
);

sseg_decoder Test2(
.num(out1),
.sseg(seg)
);

assign an[1] = sw[15];
assign an[0] = sw[15];
assign an[3:2] = 2’b11;
assign dp = 1;

endmodule //sseg1_BCD
```

Listing 5: Add3 test code

```
‘timescale 1ns / 1ps
// ELC 2137 - Lab6 - 02/20/2020
// Sebastian Lopez and Sean Dickenscheidt

module add3_test ();

reg [3:0] in;
wire [3:0] out;

integer i;

add3 Bobby(
.in(in),
.out(out)
);
```

```
initial
begin

for (i =0; i <= 4’b1111; i = i + 1)
begin
in = i;
#10;
end

$finish;

end
endmodule //add3_test
```

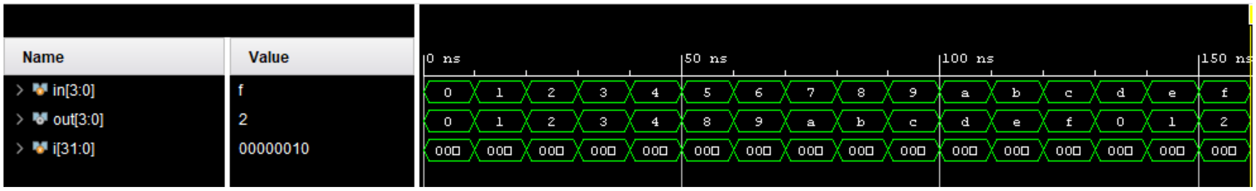
Listing 6: BCD6 test code

```
‘timescale 1ns / 1ps
// ELC 2137 - Lab6 - 02/20/2020
// Sebastian Lopez and Sean Dickenscheidt

module BCD6_test ();

reg [5:0] B;
wire [7:0] Output6;

integer i;

BCD6 test1(
.B(B),
.Output6(Output6)
);

initial
begin

for (i =0; i <= 6’b111111; i = i + 1)
begin
B = i;
#10;
end

$finish;

end

endmodule //BCD6_Test
```

Listing 7: BCD11 test code

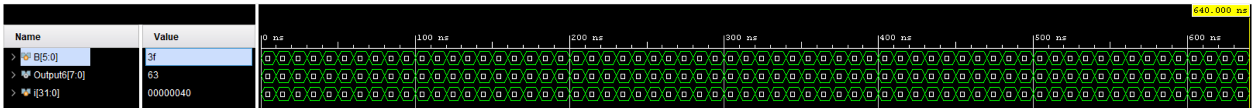
```
‘timescale 1ns / 1ps
// ELC 2137 - Lab6 - 02/20/2020
// Sebastian Lopez and Sean Dickenscheidt

module BCD11_test ();
```

```
reg [10:0] B;
wire [13:0] Output11;

integer i;

BCD11 test1(
.B(B),
.Output11(Output11)
);

initial
begin

for (i =0; i <= 11'b11111111111; i = i + 1)
begin
B = i;
#10;
end

$finish;

end

endmodule //BCD11_Test
```



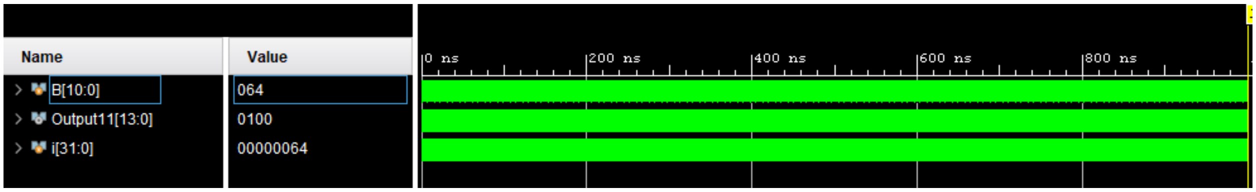
Name	Value
in[3:0]	f
out[3:0]	2
i[31:0]	00000010

Figure 1: Add3



Name	Value
B[5:0]	3f
Output6[7:0]	63
i[31:0]	00000040

Figure 2: BCD6



Name	Value
B[10:0]	064
Output11[13:0]	0100
i[31:0]	00000064

Figure 3: BCD11

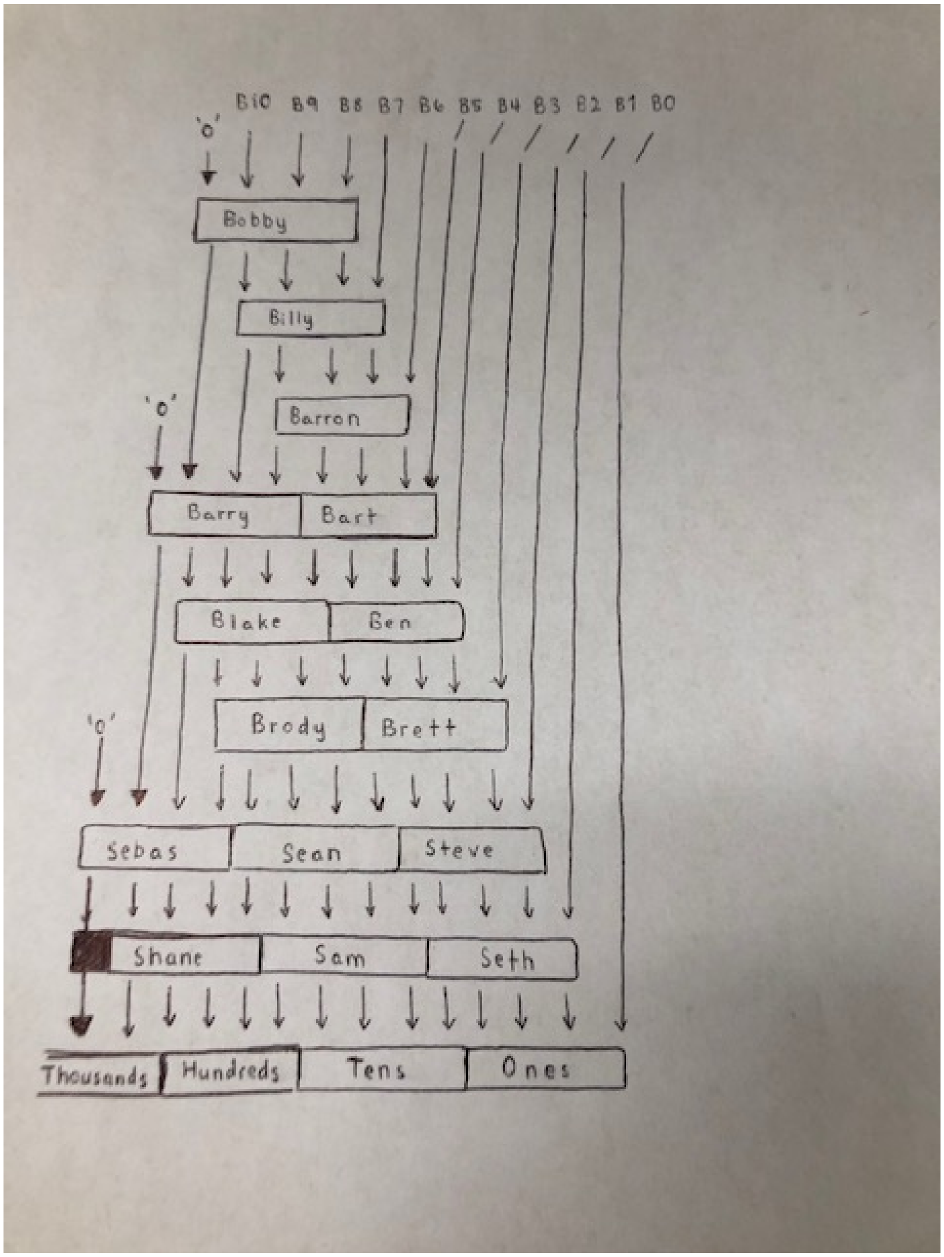


Figure 4: BCD11

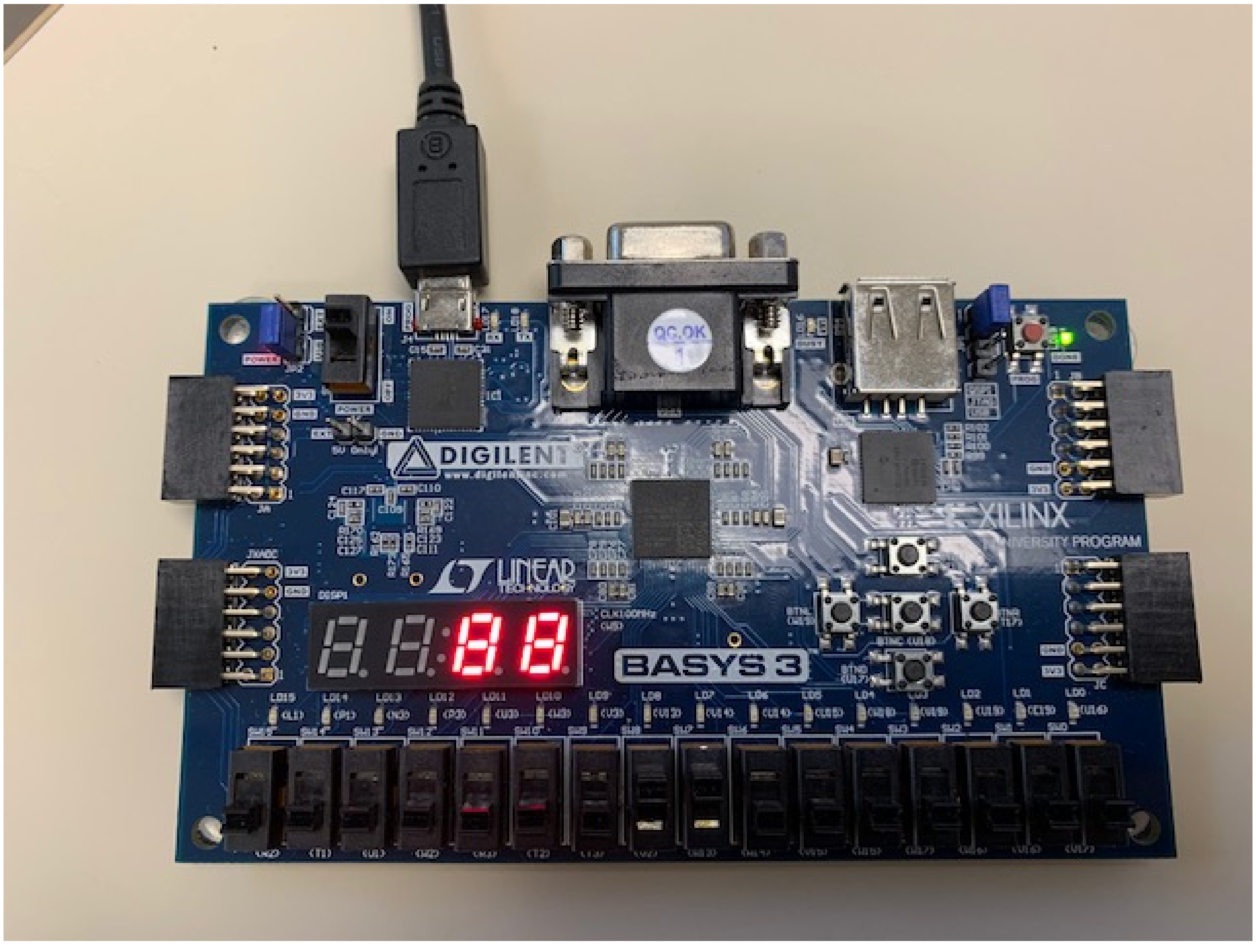


Figure 5: 88 Value on Board

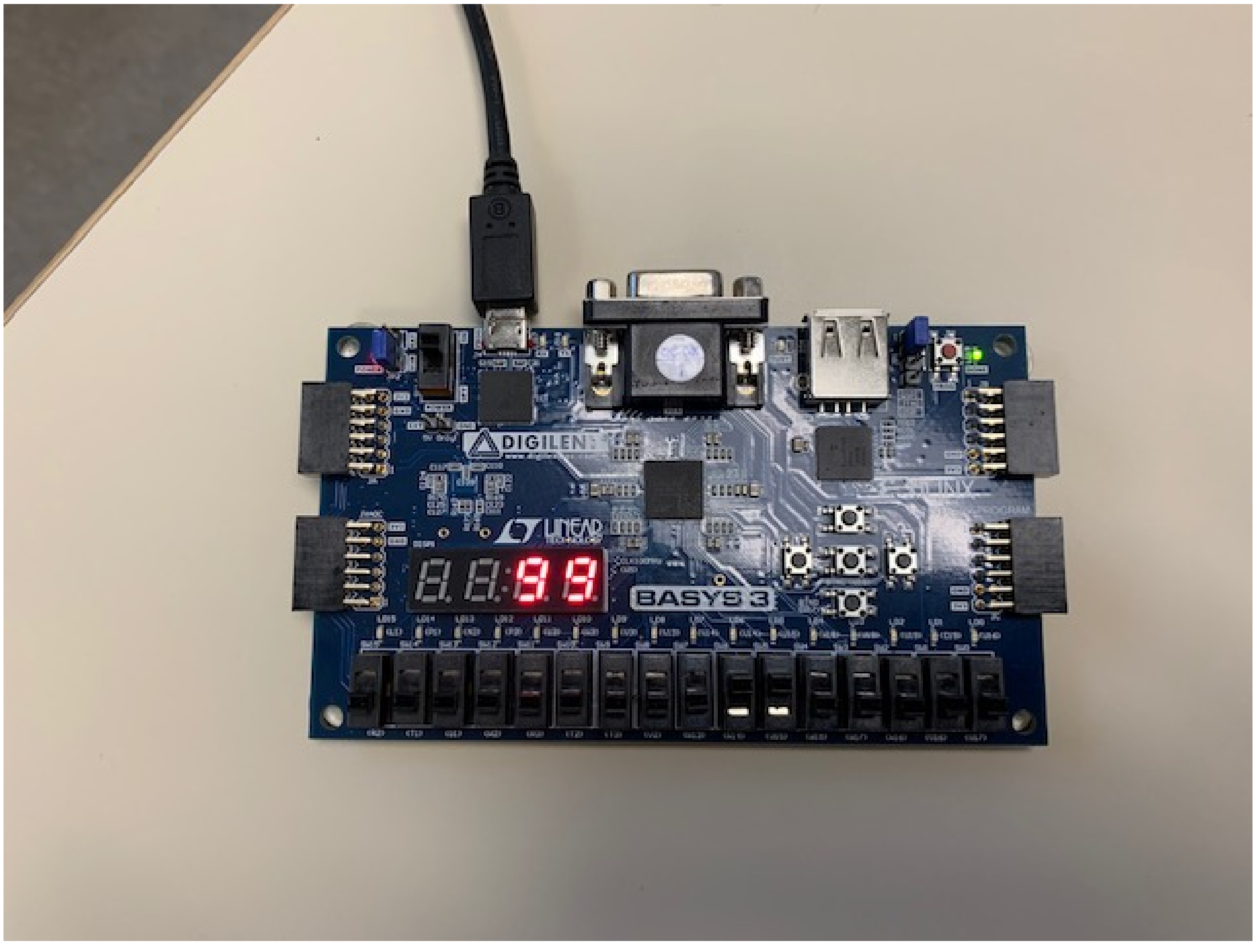


Figure 6: 99 Value on Board