



ASSIGNMENT 01
COMBINATORIAL LOGIC

EMBEDDED SYSTEMS 3

FACHHOCHSCHULE VORARLBERG

MASTER MECHATRONICS

EINGEREICHT BEI

DR. ANDRÈ MITTERBACHER

VORGELEGT VON

ROMAN PASSLER

DORNBIRN, 13.12.2017

Inhaltsverzeichnis

| | |
|--|------------|
| Abbildungsverzeichnis | II |
| Tabellenverzeichnis | III |
| Listings | IV |
| 1 Siebensegmentanzeige | 1 |
| 1.1 Einleitung | 1 |
| 1.2 Implementierung | 1 |
| 1.3 Test Bench | 2 |
| 1.4 Simulationsscript | 4 |
| 1.5 Transkript und Waveform Window | 5 |
| 1.6 Vor- und Nachteile der Implementierung | 8 |

Abbildungsverzeichnis

| | | |
|-----|---------------------------|---|
| 1.1 | Waveform Window | 5 |
|-----|---------------------------|---|

Tabellenverzeichnis

Listings

| | | |
|-----|--|---|
| 1.1 | Implementierung | 1 |
| 1.2 | Testbench für die Siebensegmentanzeige | 2 |
| 1.3 | Simulationsscript | 4 |
| 1.4 | Commandline Output A | 6 |
| 1.5 | Commandline Output B | 6 |
| 1.6 | Commandline Output C | 7 |
| 1.7 | Commandline Output D | 7 |

1 Siebensegmentanzeige

1.1 Einleitung

In der ersten Aufgabe soll eine Siebensegmentanzeige mit Verilog implementiert werden. Die Anforderungen sind:

- die Binärzahl korrekt den Ausgängen zuweisen („hex“).
- „hex_n“ ist die negierte Version von „hex“¹.
- Die Zahlen 10 bis 15 nicht vergessen!

Zusätzlich zu den Anforderungen muss die Implementierung getestet und verifiziert werden. Dies bedeutet, dass alle Eingänge stimuliert und alle Signale im „Wave Window“ angezeigt werden. Außerdem sollte der aktuelle Zustand der Ein- und Ausgänge als Lookup-Tabelle in der Modelsim-Konsole („Transkript-Fenster“) angezeigt werden.

1.2 Implementierung

Für die Implementierung wurde die Verhaltensbeschreibung verwendet, da die „Gate-Primitive“ nicht empfohlen wurde und die kontinuierliche Zuweisung zu umständlich wäre. Der Code kann in Listing 1.1 nachgelesen werden.

```
1  /*-----
2  Project: Seven Segment
3  Purpose: Implement a Seven segment BCD
4  Author: rpa2306
5  Version: 00, 11.12.2017
6
7      A
8      #-----#
9      |         |
10     F         B
11     |         |
12     G         |
13     |         |
14     E         C
15     |         |
16     |         |
17     #-----#
18      D
19  -----*/
20
21 module sevenseg(
```

¹Ist hilfreich für „active low“ Anzeigen.

```

22 // Describe the IO of the module
23     input  logic    [3:0] bin,
24     output logic    [6:0] hex,
25     output logic    [6:0] hex_n
26     // no comma after the last entry!
27 );
28
29
30
31 always_comb begin
32     case (bin)
33         4'h0 : hex = 7'b011_1111; // 0
34         4'h1 : hex = 7'b000_0110; // 1
35         4'h2 : hex = 7'b101_1011; // 2
36         4'h3 : hex = 7'b100_1111; // 3
37         4'h4 : hex = 7'b110_0110; // 4
38         4'h5 : hex = 7'b110_1101; // 5
39         4'h6 : hex = 7'b111_1101; // 6
40         4'h7 : hex = 7'b000_0111; // 7
41         4'h8 : hex = 7'b111_1111; // 8
42         4'h9 : hex = 7'b110_1111; // 9
43         4'ha : hex = 7'b111_0111; // A
44         4'hb : hex = 7'b111_1100; // B
45         4'hc : hex = 7'b011_1001; // C
46         4'hd : hex = 7'b101_1110; // D
47         4'he : hex = 7'b111_1001; // E
48         4'hf : hex = 7'b111_0001; // F
49         default : hex = 7'b000_000;
50     endcase
51     assign hex_n = ~hex;
52 end
53
54 endmodule

```

Listing 1.1: Implementierung

1.3 Test Bench

Das Testkonzept ist eine for-Schleife, die abhängig von der Bit-Länge bis zum Maximum iteriert, siehe Codezeile 44 in Listing 1.3. Die Iterationsvariable wird in jedem Iterationsschritt „bin“ zugewiesen. Dabei wird die tatsächliche Anzahl der Ein- und Ausgänge in binärer Darstellung im „Transcript“ der Modelsim-Software dargestellt. Darüber hinaus wird die Siebensegmentanzeige im „Transcript“ simuliert, siehe Codezeile 51 bis 91.

```

1  /*-----
2  Project: Seven Segment
3  Purpose: Implement a Seven segment BCD
4  Author:  rpa2306
5  Version: 00, 12.12.2017
6  -----*/
7
8  'define HORIZONTALA "\t#-----#"
9
10 module tb_sevenseg();
11
12 /*

```

```

13 * (1) Create wires to connect the DUT
14 * Like footprints for an IC on a PCB
15 */
16 logic [3:0] bin;
17 logic [6:0] hex;
18 logic [6:0] hex_n;
19 /*
20 * (2) Create an instance of the DUT
21 */
22
23 sevenseg DUT
24 // name of the moule name of this instance of the module
25 (
26 // IO connection syntax:
27 // .pin name (track name),
28 // .bin (bin),
29 // .hex (hex),
30 // .hex_n (hex_n)
31 // .
32 );
33
34 /*
35 * (3) Create stimuli for all inputs
36 */
37
38 initial begin
39 // All initial blocks are started at simulation time = 0. Only used in
40 // the test bench
41 // Execution is line by line -> it's software!!!
42 $display("-----");
43 $display(" tb_sevenseg started ");
44 $display("-----");
45 for (int i = 0; i < $bits(bin) << 2; i++) begin //
46 bin = i;
47 #100ns; // wait for 100 ns
48 $display("");
49 $display("bin\t hex\t hex_n");
50 $display("%b\t %b\t %b\n", bin, hex, hex_n);
51 /* the next lines are only for nice displaying*/
52 if (hex[0] ) begin //
53 $display("%s", 'HORIZONTALA);
54 end else begin
55 if(hex[5]) begin
56 $write("\t# ");
57 end else begin
58 $write("\t ");
59 end
60 if(hex[1]) begin
61 $write("#");
62 end
63 $display("");
64 end
65 for (int j = 0; j < 2; j++) begin
66 for (int n = 0; n < 4; n++) begin
67 if (hex[4] && j==1 || hex[5] && j==0) begin
68 $write("\t|");
69 end else begin
70 $write("\t ");
71 end
72 $write(" ");
73 if (hex[1] && j==0 || hex[2] && j==1) begin

```



```
73         $display("|");
74     end else begin
75         $display("");
76     end
77 end
78 if (hex[6] && j==0 || hex[3] && j==1) begin
79     $display("%s", 'HORIZONTALA);
80 end else begin
81     if (hex[4] && j==1 || hex[5] && j==0) begin
82         $write("\t#");
83     end else begin
84         $write("\t");
85     end
86     if (hex[1] && j==0 || hex[2] && j==1) begin
87         $write("#");
88     end
89     $display("");
90 end
91 end //
92 end
93 $display("-----");
94 $display("    sb_sevenseg finished    ");
95 $display("-----");
96 end
97
98 endmodule
```

Listing 1.2: Testbench für die Siebensegmentanzeige

1.4 Simulationsscript

In Listing 1.3 ist das Simulationsscript dargestellt. Es beinhaltet dieselben Befehle wie in der letzten Lehrveranstaltung, natürlich angepasst an die Siebensegmentanzeige.

```
1 # Create simulation environment
2 vlib work
3 vmap work work
4
5 # Compile desing files -> use file names
6 vlog ../src/sevenseg.sv
7 # Compile the test bench
8 vlog tb_sevenseg.sv
9 # Init simulation -> use module name
10 vsim tb_sevenseg
11 # -r recursive
12 log -r *
13 do wave_tb_sevenseg.tcl
14
15 # Run simulation
16 run -all
17 # run 100us
18 # Show results
19 view wave
```

Listing 1.3: Simulationsscript

1.5 Transkript und Waveform Window

In Abbildung 1.1 können die Ausgänge „hex“ und „hex_n“ im Waveform Window überprüft werden. In Listings 1.4 bis 1.7 ist der Commandline Output abgebildet, welcher ebenfalls zeigt, dass der Test erfolgreich war.



Abbildung 1.1: Waveform Window
Quelle: eigene Ausarbeitung

```

1 # bin    hex hex_n
2 # 0000    0111111 1000000
3 #
4 #   #-----#
5 #   |         |
6 #   |         |
7 #   |         |
8 #   |         |
9 #   #         #
10 #   |         |
11 #   |         |
12 #   |         |
13 #   |         |
14 #   #-----#
15 #
16 # bin    hex hex_n
17 # 0001    0000110 1111001
18 #
19 #             #
20 #             |
21 #             |
22 #             |
23 #             |
24 #             #
25 #             |
26 #             |
27 #             |
28 #             |
29 #             #
30 #
31 # bin    hex hex_n
32 # 0010    1011011 0100100
33 #
34 #   #-----#
35 #   |         |
36 #   |         |
37 #   |         |
38 #   |         |
39 #   #-----#
40 #   |         |
41 #   |         |
42 #   |         |
43 #   |         |
44 #   #-----#
45 #
46 # bin    hex hex_n
47 # 0011    1001111 0110000
48 #
49 #   #-----#
50 #   |         |
51 #   |         |
52 #   |         |
53 #   |         |
54 #   #-----#
55 #   |         |
56 #   |         |
57 #   |         |
58 #   |         |
59 #   #-----#

```

Listing 1.4: Commandline Output A

```

1 # bin    hex hex_n
2 # 0100    1100110 0011001
3 #
4 #   #         #
5 #   |         |
6 #   |         |
7 #   |         |
8 #   |         |
9 #   #-----#
10 #   |         |
11 #   |         |
12 #   |         |
13 #   |         |
14 #           #
15 #
16 # bin    hex hex_n
17 # 0101    1101101 0010010
18 #
19 #   #-----#
20 #   |         |
21 #   |         |
22 #   |         |
23 #   |         |
24 #   #-----#
25 #   |         |
26 #   |         |
27 #   |         |
28 #   |         |
29 #   #-----#
30 #
31 # bin    hex hex_n
32 # 0110    1111101 0000010
33 #
34 #   #-----#
35 #   |         |
36 #   |         |
37 #   |         |
38 #   |         |
39 #   #-----#
40 #   |         |
41 #   |         |
42 #   |         |
43 #   |         |
44 #   #-----#
45 #
46 # bin    hex hex_n
47 # 0111    0000111 1111000
48 #
49 #   #-----#
50 #   |         |
51 #   |         |
52 #   |         |
53 #   |         |
54 #           #
55 #   |         |
56 #   |         |
57 #   |         |
58 #   |         |
59 #           #

```

Listing 1.5: Commandline Output B

```

1 # bin    hex hex_n
2 # 1000    1111111 0000000
3 #
4 #   #-----#
5 #   |         |
6 #   |         |
7 #   |         |
8 #   |         |
9 #   #-----#
10 #   |         |
11 #   |         |
12 #   |         |
13 #   |         |
14 #   #-----#
15 #
16 # bin    hex hex_n
17 # 1001    1101111 0010000
18 #
19 #   #-----#
20 #   |         |
21 #   |         |
22 #   |         |
23 #   |         |
24 #   #-----#
25 #   |         |
26 #   |         |
27 #   |         |
28 #   |         |
29 #   #-----#
30 #
31 # bin    hex hex_n
32 # 1010    1110111 0001000
33 #
34 #   #-----#
35 #   |         |
36 #   |         |
37 #   |         |
38 #   |         |
39 #   #-----#
40 #   |         |
41 #   |         |
42 #   |         |
43 #   |         |
44 #   #   #
45 #
46 # bin    hex hex_n
47 # 1011    1111100 0000011
48 #
49 #   #
50 #   |
51 #   |
52 #   |
53 #   |
54 #   #-----#
55 #   |         |
56 #   |         |
57 #   |         |
58 #   |         |
59 #   #-----#

```

Listing 1.6: Commandline Output C

```

1 # bin    hex hex_n
2 # 1100    0111001 1000110
3 #
4 #   #-----#
5 #   |         |
6 #   |         |
7 #   |         |
8 #   |         |
9 #   #
10 #   |
11 #   |
12 #   |
13 #   |
14 #   #-----#
15 #
16 # bin    hex hex_n
17 # 1101    1011110 0100001
18 #
19 #               #
20 #               |
21 #               |
22 #               |
23 #               |
24 #   #-----#
25 #   |         |
26 #   |         |
27 #   |         |
28 #   |         |
29 #   #-----#
30 #
31 # bin    hex hex_n
32 # 1110    1111001 0000110
33 #
34 #   #-----#
35 #   |         |
36 #   |         |
37 #   |         |
38 #   |         |
39 #   #-----#
40 #   |         |
41 #   |         |
42 #   |         |
43 #   |         |
44 #   #-----#
45 #
46 # bin    hex hex_n
47 # 1111    1110001 0001110
48 #
49 #   #-----#
50 #   |         |
51 #   |         |
52 #   |         |
53 #   |         |
54 #   #-----#
55 #   |         |
56 #   |         |
57 #   |         |
58 #   |         |
59 #   #

```

Listing 1.7: Commandline Output D

1.6 Vor- und Nachteile der Implementierung

Folgend sind die Vor- und Nachteile der Implementierung gelistet:

Vorteile

- Keine Counter
- einfach

Nachteile

- Mögliche Verzögerungen, da die Implementierung asynchron ist.