Assignment 01 – Combinatorial Logic

Documentation

Assignment Protocol

Fachhochschule Vorarlberg

ET-Dual

HDL – Hardware Description Layer

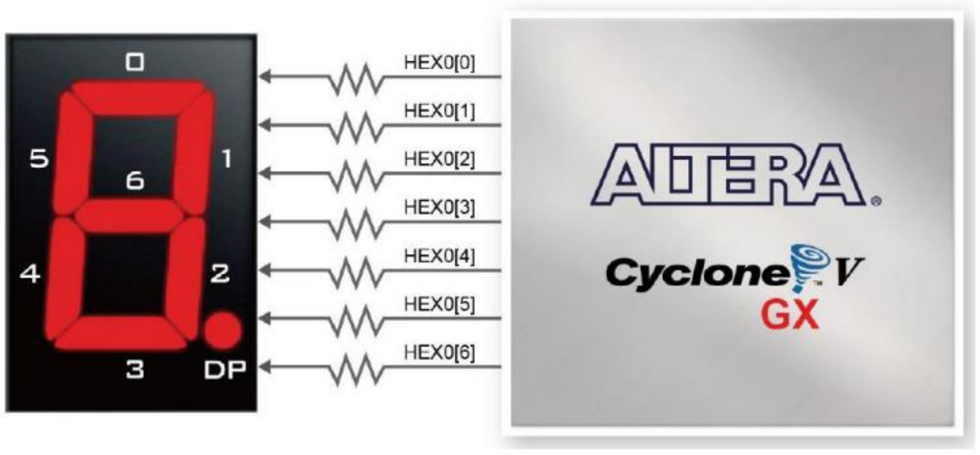


Figure 1: Seven segment display on FPGA development board (Src.: Mitterbacher)

Author:
Stefan Duenser

Dornbirn, 20/10/2021

1. Task description

The aim of the first assignment was to implement a combinatorial logic. As in the lectures, System Verilog should be used as the hardware description language.

A seven-segment display was chosen as an example for combinatorial logic. The seven LED bars of the display are controlled in a way that a decimal number from 0 to 15 is shown on the display as a hex number.

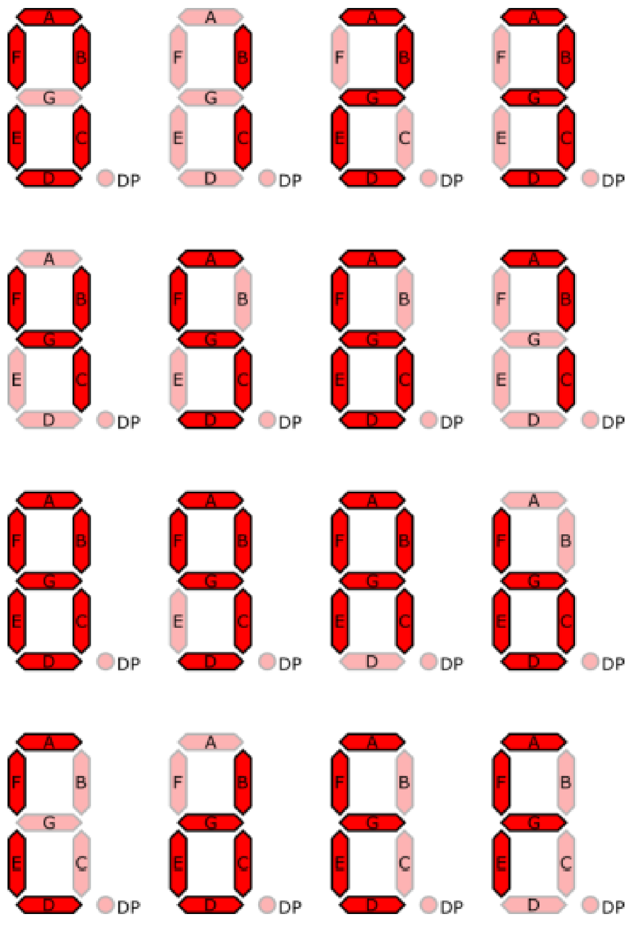
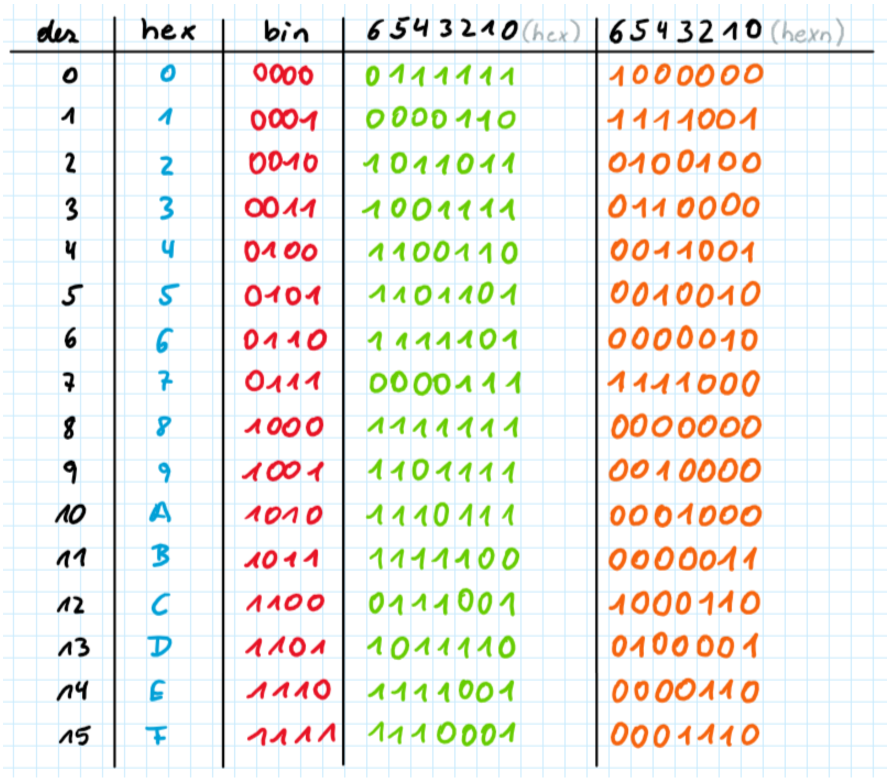


Figure 2: Display HEX values (Src.: Mitterbacher)

2. Truth table

The truth table is used to determine what value the output assumes, depending on the value applied to the input(s). The following figure shows the truth table for the seven-segment display.

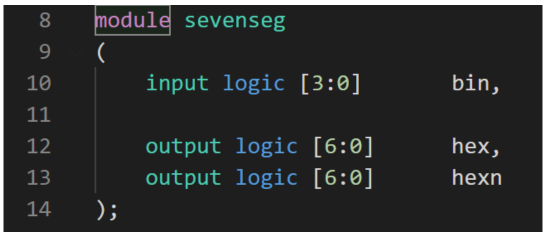


dez	hex	bin	6543210 (hex)	6543210 (hexn)
0	0	0000	0111111	1000000
1	1	0001	0000110	1111001
2	2	0010	1011011	0100100
3	3	0011	1001111	0110000
4	4	0100	1100110	0011001
5	5	0101	1101101	0010010
6	6	0110	1111101	0000010
7	7	0111	0000111	1111000
8	8	1000	1111111	0000000
9	9	1001	1101111	0010000
10	A	1010	1110111	0001000
11	B	1011	1111100	0000011
12	C	1100	0111001	1000110
13	D	1101	1011110	0100001
14	E	1110	1111001	0000110
15	F	1111	1110001	0001110

Figure 3: Truth table of the seven-segment display for hexadecimal numbers

3. Source code

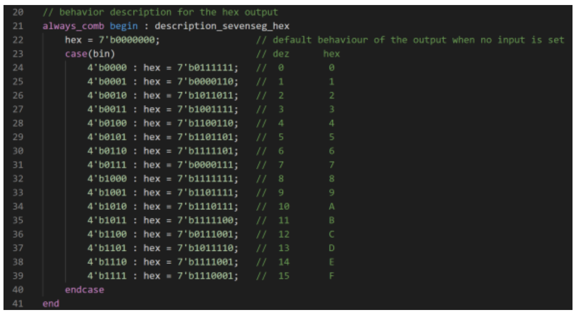
As specified in the assignment, a file "sevense.sv" was created in which the source code of the seven-segment display was written. The input and output variables were not configured individually as in the lecture but directly as a bus. The input bus *bin* consisted of four bits, the two output buses *hex* and *hexn* of 7 bits each to control the individual LED bars.



```
module sevenseg
(
    input logic [3:0]       bin,

    output logic [6:0]      hex,
    output logic [6:0]      hexn
);
```

Figure 4: Input and output bus



```
// behavior description for the hex output
always_comb begin : description_sevenseg_hex
    hex = 7'b0000000;                   // default behaviour of the output when no input is set
    case(bin)                           // dez      hex
        4'b0000 : hex = 7'b0111111;     //  0        0
        4'b0001 : hex = 7'b0000110;     //  1        1
        4'b0010 : hex = 7'b1011011;     //  2        2
        4'b0011 : hex = 7'b1001111;     //  3        3
        4'b0100 : hex = 7'b1100110;     //  4        4
        4'b0101 : hex = 7'b1101101;     //  5        5
        4'b0110 : hex = 7'b1111101;     //  6        6
        4'b0111 : hex = 7'b0000111;     //  7        7
        4'b1000 : hex = 7'b1111111;     //  8        8
        4'b1001 : hex = 7'b1101111;     //  9        9
        4'b1010 : hex = 7'b1110111;     //  10       A
        4'b1011 : hex = 7'b1111100;     //  11       B
        4'b1100 : hex = 7'b0111001;     //  12       C
        4'b1101 : hex = 7'b1011110;     //  13       D
        4'b1110 : hex = 7'b1111001;     //  14       E
        4'b1111 : hex = 7'b1110001;     //  15       F
    endcase
end
```

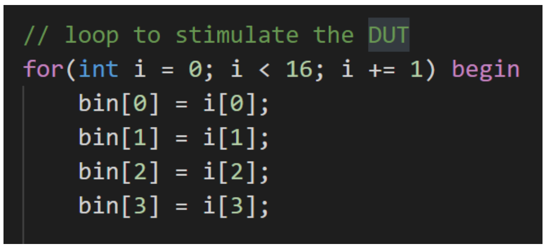
Figure 5: Binary output (hex) in relation to the input (bin)
- from truth table

As can be seen in Fig. 5 above, the task was implemented with the behaviour description *`always_comb`*. A case is used to map all possible input states to the output. This kind of implementation is very close to the truth table and very clear. With this method, a time-consuming setting up of a Boolean function via a KV diagram is not necessary. In addition, the program is very easy to read.

4. Testbench

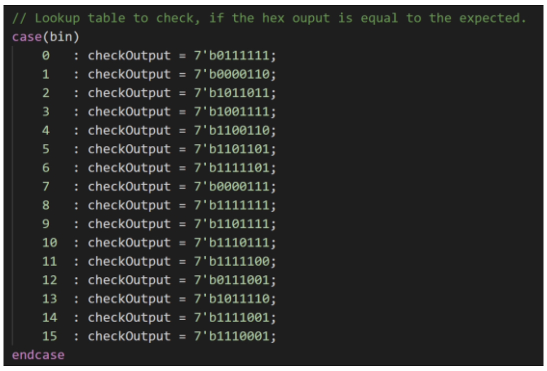
So far, only code has been generated, but it cannot be verified yet. With the testbench real pins are assigned to the inputs and outputs of the FPGA.

To check the seven-segment display, each input state must be simulated and then it has to be checked if the correct bit order is generated or if the bits for the display of the respective HEX number are driven incorrectly. As can be seen in Fig. 6, this so-called "stimulation" was realized via a for-loop. The input state is continuously changed in the loop.



```
// loop to stimulate the DUT
for(int i = 0; i < 16; i += 1) begin
    bin[0] = i[0];
    bin[1] = i[1];
    bin[2] = i[2];
    bin[3] = i[3];
```

Figure 6: Stimulation of the DUT with a for-loop

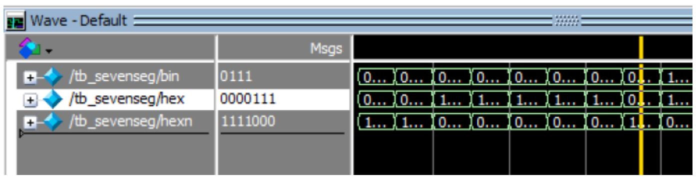


```
// Lookup table to check, if the hex ouput is equal to the expected.
case(bin)
    0   : checkOutput = 7'b0111111;
    1   : checkOutput = 7'b0000110;
    2   : checkOutput = 7'b1011011;
    3   : checkOutput = 7'b1001111;
    4   : checkOutput = 7'b1100110;
    5   : checkOutput = 7'b1101101;
    6   : checkOutput = 7'b1111101;
    7   : checkOutput = 7'b0000111;
    8   : checkOutput = 7'b1111111;
    9   : checkOutput = 7'b1101111;
    10  : checkOutput = 7'b1110111;
    11  : checkOutput = 7'b1111100;
    12  : checkOutput = 7'b0111001;
    13  : checkOutput = 7'b1011110;
    14  : checkOutput = 7'b1111001;
    15  : checkOutput = 7'b1110001;
endcase
```

Figure 7: Lookup table for output (hex) to expected output comparision

5. Verification

The "ModelSim" program can then be used to simulate the test bench. The outputs *`hex`* and *`hexn`* as well as the input *`bin`* are displayed. A graphical representation (Fig. 9) clearly shows which bits are "high" or "low" at the input and at the two outputs at which point in time.



```
Wave - Default
                      Msgs
/tb_sevenseg/bin      0111
/tb_sevenseg/hex      0000111
/tb_sevenseg/hexn     1111000
```

Figure 8: ModelSim Simulation of the output/input - example with number 7

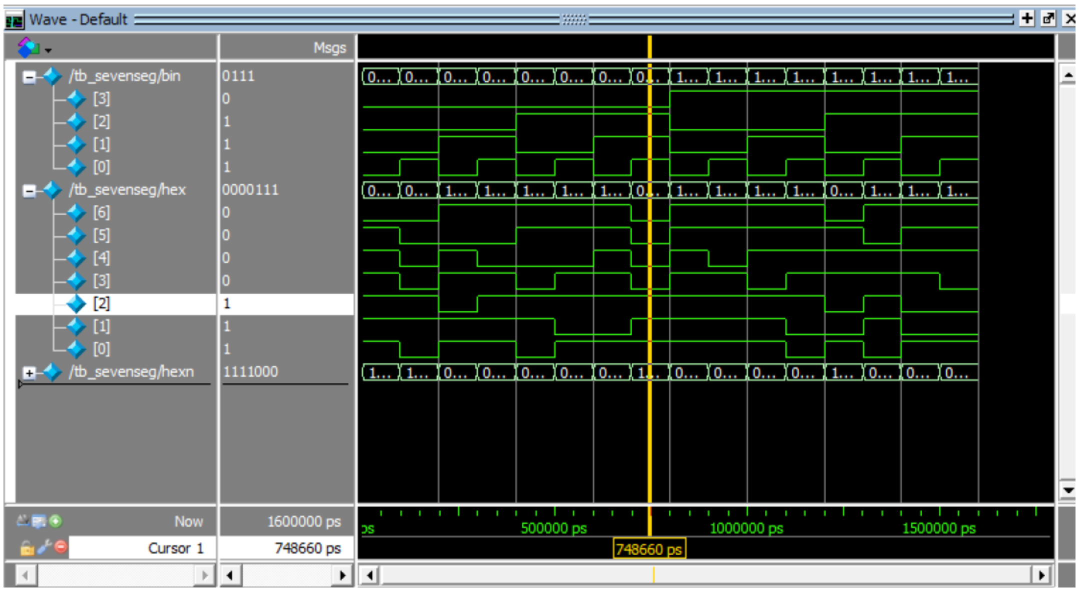
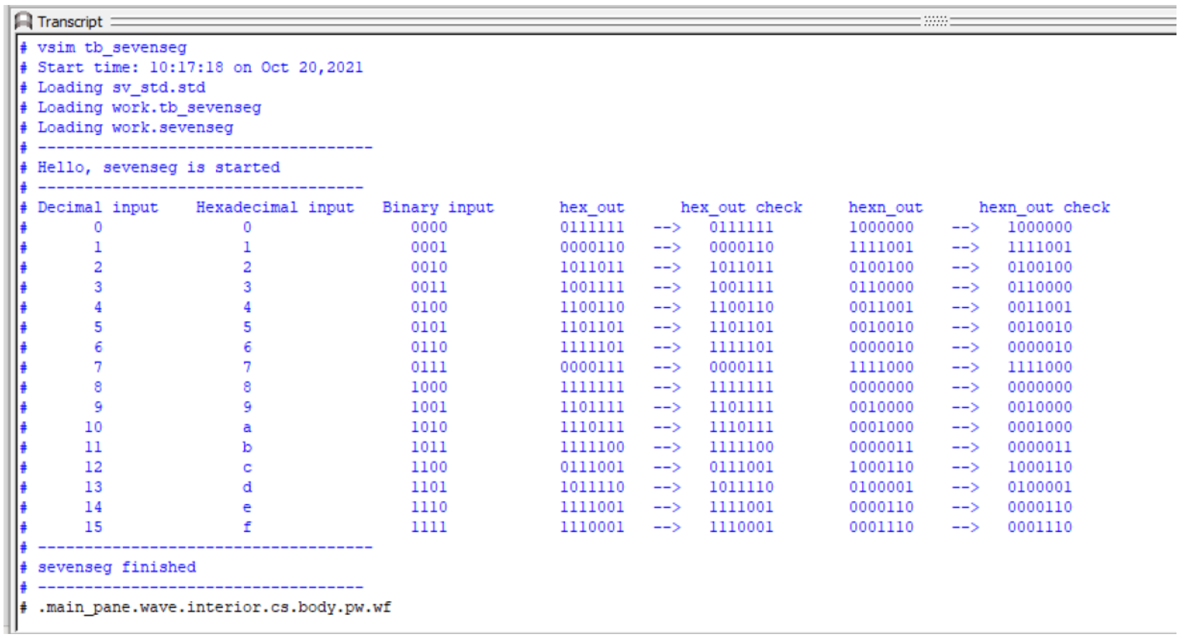


Figure 9: Output verification in bitwise display - example with number 7

In addition, the current bit sequence at the input and output can be output in the console window. A lookup table (Fig. 7) can then be used to compare whether the output value corresponds to the expected input value or not.

The output on the console is done in System Verilog via the "$"-character command which is comparable with "printf" in C.



```
Transcript
# vsim tb_sevenseg
# Start time: 10:17:18 on Oct 20,2021
# Loading sv_std.std
# Loading work.tb_sevenseg
# Loading work.sevenseg
# ------------------------------------
# Hello, sevenseg is started
# -----------------------------------
# Decimal input    Hexadecimal input   Binary input       hex_out      hex_out check      hexn_out      hexn_out check
#         0                0             0000            0111111  -->  0111111           1000000  -->   1000000
#         1                1             0001            0000110  -->  0000110           1111001  -->   1111001
#         2                2             0010            1011011  -->  1011011           0100100  -->   0100100
#         3                3             0011            1001111  -->  1001111           0110000  -->   0110000
#         4                4             0100            1100110  -->  1100110           0011001  -->   0011001
#         5                5             0101            1101101  -->  1101101           0010010  -->   0010010
#         6                6             0110            1111101  -->  1111101           0000010  -->   0000010
#         7                7             0111            0000111  -->  0000111           1111000  -->   1111000
#         8                8             1000            1111111  -->  1111111           0000000  -->   0000000
#         9                9             1001            1101111  -->  1101111           0010000  -->   0010000
#        10                a             1010            1110111  -->  1110111           0001000  -->   0001000
#        11                b             1011            1111100  -->  1111100           0000011  -->   0000011
#        12                c             1100            0111001  -->  0111001           1000110  -->   1000110
#        13                d             1101            1011110  -->  1011110           0100001  -->   0100001
#        14                e             1110            1111001  -->  1111001           0000110  -->   0000110
#        15                f             1111            1110001  -->  1110001           0001110  -->   0001110
# ------------------------------------
# sevenseg finished
# -----------------------------------
# .main_pane.wave.interior.cs.body.pw.wf
```

Figure 10: Console output