"FPGA-Programming" Exercise Sheet II:

Numbers and Displays

Goal of this exercise is to learn how simple combinatorial circuits are designed that can perform binary-to-decimal number conversions and binary-coded-decimal (BCD) addition.

As target hardware always choose FPGA chip Cyclone V SoC 5CSEMA5F31C6

1) *Displaying numbers on 7-segment displays*

   In this part we with to display on the 7-segment displays $HEX1$ and $HEX0$ the values set by the switches $SW[7-0]$. Let the binary values denoted by $SW[7-4]$ and $SW[3-0]$ be displayed on $HEX1$ and $HEX0$, respectively. The circuit should be able to display the digits from 0 to 9, and should treat the binary values from 1010 to 1111 as don't cares.

   Goal of this part is to derive the logic functions for driving the 7-segment displays manually. Therefore only simple VHDL assignments should be used in which each function can be described as a boolean formula.

   Perform following steps:

   i) For each segment derive the truth table for a circuit being driven by a 4-bit vector. The circuit should be able to display the digits from 0 to 9, and should treat the binary values from $1010|_2$ to $1111|_2$ as *don't cares*. Remember the segments are *low-active*.

   ii) Derive the minimal possible logic function for the segments. Use an appropriate method such as the Karnaugh map.

   iii) Create a new Quartus project for the circuit, named **e_my_bcd7seg**.

   iv) Describe the Sub-Level Entity of the binary-coded-decimal (BCD) to 7-segment decoder, named **e_bcd7seg**.

   v) Create the VHDL Top-Level Entity, named **e_my_bcd7seg** and describe the complete circuit by using two instances of **e_bcd7seg**.

   vi) Import the necessary pin assignments.

   vii) Compile the project, download the compiled circuit and test its functionality.

2) *Binary-to-Decimal Converter*

In this part, a circuit should be designed, which converts a 4-bit binary number $V = v_3v_2v_1v_0$ into a decimal number consisting of two digits $D = d_1d_0$. Following table illustrates the functionality:

v_3	v_2	v_1	v_0	d_1	d_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	2
0	0	1	1	0	3
0	1	0	0	0	4
0	1	0	1	0	5
0	1	1	0	0	6
0	1	1	1	0	7
1	0	0	0	0	8
1	0	0	1	0	9
1	0	1	0	1	0
1	0	1	1	1	1
1	1	0	0	1	2
1	1	0	1	1	3
1	1	1	0	1	4
1	1	1	1	1	5

One part of the circuit is given in figure 2.1.

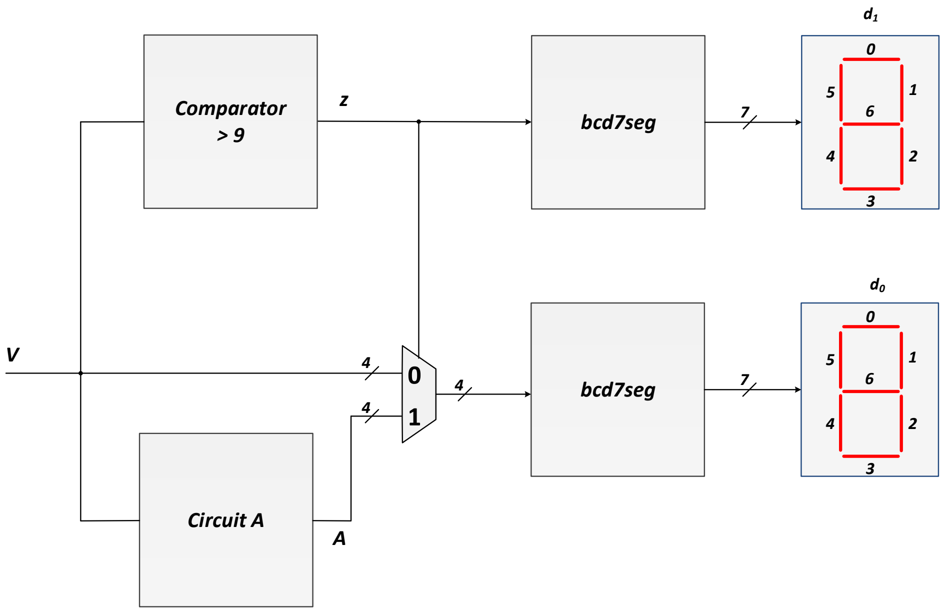

Figure 2.1: Binary-to-Decimal Converter Circuit

It includes a comparator, which checks whether the value of V is greater than 9. The output of the comparator z is used for driving the 7-segment displays. The output z can be described by a single boolean expression, depending on the input bit-vector V. The circuit furthermore includes a 4-bit wide 2-to-1 multiplexer, which has been described in Exercise Sheet I). Task of this multiplexer is to drive the lower decimal digit d_0. Therefore the value of V will be forwarded to the 7-segment decoder, if $z = 0$, otherwise the value A will be forwarded, the output of circuit A. For the description of circuit A it should be noted that input values of $V \leq 9$ do not have an affect on the functionality of the circuit, since the multiplexer will output V. If $V > 9$ the multiplexer will forward A. Therefore circuit A must yield correct values for driving d_0. As decoder use the already described VHDL Entity **e_bcd7seg**.

Goal of this part is to only use simple VHDL assignments and logic functions based on boolean expressions. The VHDL code must not include VHDL high-level syntax such as **if** or **case** statements.

Perform following steps:

i) Utilize the given table to derive the boolean expression for the output of the comparator circuit.

ii) The the description of circuit A derive the truth table and the boolean expressions for the output vector.

iii) Create a new Quartus project for the circuit, named **e_my_bin2dec**.

iv) Develop the VHDL code for the design. Use switches $SW[3-0]$ for the 4-bit binary input vector V. Use the 7-segment displays $HEX1$ and $HEX0$ to display the digits d_1 and d_0.

v) Import the necessary pin assignments.

vi) Compile the circuit. Download the compiled circuit and test its functionality by trying out all possible combination of V.

3) *Four Bit Ripple Carry Adder Circuit*

Following figure shows the circuit of a full adder, with inputs a, b, and c_i as well as outputs s and c_o.

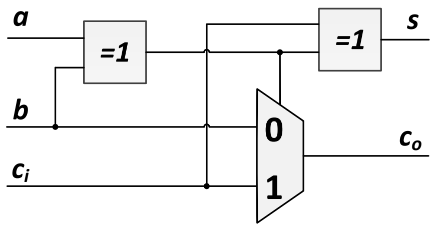

Figure 3.1: Full Adder Circuit

The symbol of a full adder circuit is given below:

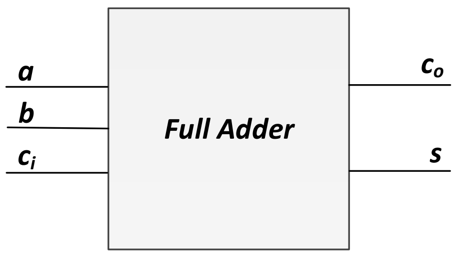

Figure 3.2: Full Adder Symbol

The truth table of a full adder is given by:

b	a	c_i	c_o	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

A 1-bit wide full adder adds the two input operands a, b as well as a potential carry input of previous digit c_i (*carry in*). The generated output values are the sum-bit s and a potential carry output for the next digit c_o (*carry out*), therefore $c_o s = a + b + c_i$.

Following figure illustrates, how four instances of this full adder circuit can be utilized to design a circuit, which adds two 4-bit numbers. This type of adder is called *Ripple Carry Adder*, since the carry signals potentially will ripple from one full adder to the next.

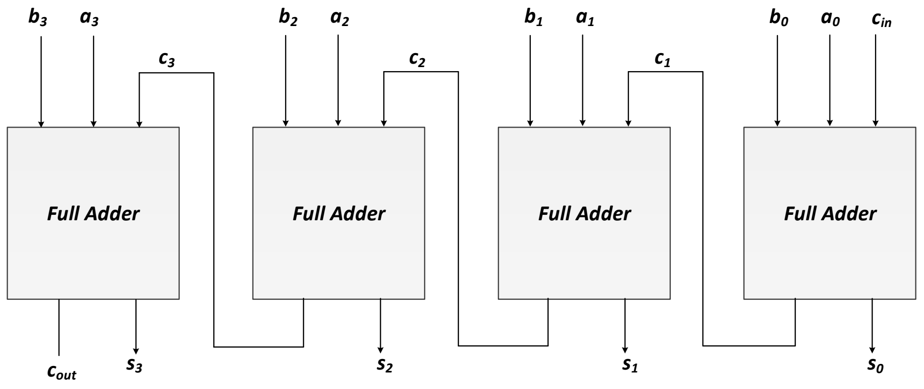

Figure 3.3: 4-Bit Ripple Carry Adder

Develop a VHDL code, which implements a 4-bit wide Ripple Carry Adder. Perform following steps:

i) Based on the table above derive the boolean expressions for both of the outputs of a full adder.

ii) Create a new Quartus project for the Ripple Carry Adder circuit, named **e_my_ripplecarryadder**.

iii) Write a VHDL Sub-Level Entity for the full adder circuit, named **e_fulladder** and a Top-Level VHDL Entity (**e_my_ripplecarryadder**) which includes four instances of the full adder component.

iv) Use switches $SW[7-4]$ for the input vector A, switches $SW[3-0]$ for input vector B. Use switch $SW[8]$ for the carry input bit c_{in}. Connect the outputs of the circuit c_{out} and S with $LEDR[4-0]$.

v) Import the necessary pin assignments.

vi) Compile the circuit, download it and test its functionality for different values of A, B and c_{in}.

4) *Addition of two BCD numbers using boolean expressions*

For this part a circuit should be designed, which can add two binary coded decimal (BCD) numbers. The inputs of the circuit are two 4-bit wide BCD numbers X and Y, as well as a carry input bit c_{in}. When these inputs are added, the result will be a 5-bit binary number. But the result is to be displayed on the 7-segment displays as a two-digit BCD sum S_1S_0. For a sum equal to zero $S_1S_0 = 00$ should be displayed, for a sum equal to ten $S_1S_0 = 10$ should be displayed and so on. Further, note that the inputs X and Y are assumed to be decimal digits, which means that the largest sum that needs to be handled by this circuit is $S_1S_0 = 9 + 9 + 1 = 19$.

Perform following steps:

i) Create a new Quartus project for the circuit, named **e_my_BCDadder**. Use for 4-bit wide adder circuit of part 3) to produce a four-bit sum and carry output for the operation $X + Y + c_{in}$.

ii) A proper way to work out the design of the circuit is to first make it handle sums $X+Y+c_{in} \leq 15$. With these values, the circuit of part 2) can be used to convert the 4-bit sum into the two decimal digits S_1S_0. Then, once this is working, modify the design to handle values of $15 < (X+Y) \leq 19$. One way to do this is to still use the circuit of part 2), but to modify its outputs before attaching them to the 7-segment display to make the necessary adjustments when the sum from the adder exceeds 15. The VHDL code only should contain simple assignments and boolean expressions, not VHDL high-level expressions such as **if** and **case** statements, should be used for this part.

iii) Use switches $SW[7-4]$ for input X, switches $SW[3-0]$ for input Y. Use switch $SW[8]$ for the carry input c_{in}. Connect the resulting 4-bit sum and the carry output bit with $LEDR[4-0]$. Display the BCD values of X and Y on the 7-segment displays $HEX5$ and $HEX3$, and display the result S_1S_0 on $HEX1$ and $HEX0$.

iv) Since the circuit handles only BCD digits, check for the cases when the input X or Y is greater than nine. If this occurs, indicate an error by turning on $LEDR[9]$.

v) Import the necessary pin assignments and compile the design. Analyse the circuit with the **RTL Viewer**.

vi) Download the compiled circuit and test the functionality for different values of X, Y and c_{in}.

5) *Addition of two BCD numbers using high-level syntax*

In part 4) a VHDL code for a BCD adder using simple assignments and boolean expressions has been designed. A different approach for describing the adder in VHDL code is to specify an algorithm like the one represented by the following pseudo-code.

```
1  T_0 = A + B + c_0;
2  if(T_0) > 9) then
3     Z_0 = 10;
4     c_1 = 1;
5  else
6     Z_0 = 0;
7     c_1 = 0;
8  end if
9  S_0 = T_0 - Z_0;
10 S_1 = c_1;
```

It is reasonably straightforward to see what circuit could be used to implement this pseudo-code. Lines 1 and 9 represent adders, lines 2-8 correspond to multiplexers, and testing for the condition $T_0 > 9$ requires comparators.

Note that the subtraction operation (line 9) could also be performed by an addition operation using two's complement representation. The intent of this part is, to examine the effects of relying more on the VHDL compiler to design the circuit by using VHDL expressions such as **if-else**, **case-when** along with the VHDL operators **>**, **+**, **-**. Perform the following steps:

i) Create a new Quartus project for the circuit, named **e_my_BCDadder2**.
ii) Describe the 7-segment decoder (**e_bcd7seg**) by using a *combinatorial process* and a **case** statement.
iii) Include all functions of **std_logic_unsigned** from the **ieee** library and describe the circuit **e_my_BCDadder2** using high-level syntax. For the implementation of the pseudo-code use a *combinatorial process* and the **if** statement.
iv) Implement two different solutions the line 9 of the pseudo-code. One should use subtraction (**a_my_BCDadder2_1**), one addition using two's complement representation (**a_my_BCDadder2_2**).
v) Use switches $SW[7-4]$ for input A, switches $SW[3-0]$ for input B- Use switch $SW[8]$ for the carry input bit c_{in}. The value of A should be displayed on the 7-segment display $HEX5$, while B should be displayed on $HEX3$. Display the BCD sum S_1S_0 on $HEX1$ and $HEX0$.
vi) Import the necessary pin assignments and compile the circuit.
vii) Use the **RTL Viewer** to analyse the circuit. Compare the synthesis with the one of part 4).
viii) Download the compiled circuit and test its functionality with different values of A, B and c_{in}.
ix) Also test *selective signal assignment* and *conditional signal assignment* syntax for the description of the 7-segment decoder **e_bcd7seg**.

6) 6-*bit binary number to two-digit decimal number converting circuit*

Design a circuit, which converts a 6-bit binary number into a decimal number consisting of 2 BCD digits. Use switches $SW[5-0]$ as binary input vector and the 7-segment displays $HEX1$ and $HEX0$ for displaying the decimal number. Following figure illustrates the converting circuit.

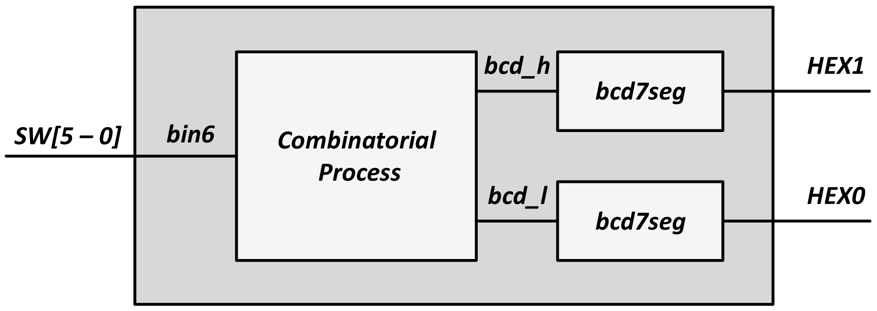

Figure 6.1: 6-bit binary number to two-digit decimal number converting circuit

Again, use VHDL high-level syntax, such as *combinatorial processes*, **case** and **if** statements. Perform following steps:

i) Create a new Quartus project for the circuit, named **e_my_bin2dec**.

ii) For the description of the 7-segment decoder (**e_bec7seg**) use one of part 5).

iii) Describe the VHDL Top-Level entity (**e_my_bin2dec**) using high-level syntax.

iv) Import the necessary pin assignments.

v) Compile the project, download the compiled circuit and test its functionality.