Report: Introduction to VHDL

Bilkent University Electrical and Electronics Department
EE102-01 Lab 2

Semih Akkoç - 22103132

October 3, 2022

Purpose:

The intent of this lab was to study the essentials of the design and implementation of a combinational circuit on Basys3 using VHDL. Additionally, to get acquainted with combinational logic circuits that are time-independent, memoryless circuits whose outputs solely depend on the combination of inputs. .

Methodology:

This lab work primarily practiced on a real-life problem solved via a combinational circuit on your BASYS3 using VHDL. The problem chosen to be implemented was determining whether a wave is an electromagnetic wave or a mechanical wave by examining the four properties accordingly:

- Is this wave a traverse wave?
- Can this wave carry energy?
- Can this wave travel in a vacuum?
- Is this wave a compressional wave?

By checking these properties, which wave is given can be established. After establishing a problem and filling a truth table, a function that yields these methods have been discovered using canonical SOP (sum of products). Then when the functions have been simplified, they have been implemented.

Questions:

How does one specify the inputs and outputs of a module in VHDL?

In the code section entity, inputs and outputs are defined. Entity declaration forms an interface, the port statement defines the ports that are occupied and being used. Although, information about implementing the circuit is not declared in entity design, that information needs to be declared in architectural design.

How does one use a module inside another code/module? What does PORT MAP do?

The signals implement attaching design entities. Further, by using signals, the transmission of changing values can be achieved. A Port map associates the given inputs parallel to the written code architecture. Furthermore, this process gives users the ability to change the specified ports in the module to different modules.

What is a constraint file? How does it relate your code to the pins on your FPGA?

Implementing a constraint file benefits in determining and describing which components of FPGA are being utilized for the designed project. Additionally, FPGA is adaptive for the given process since the design of the FPGA can be altered by adjusting the constraint file. Therefore, if there are parts that have not been utilized, FPGA automatically disables those parts and ensures that desired components are in use.

What is the purpose of writing a testbench?

The testbench code enables users to replicate circuit design in timing diagrams and waveforms. Besides, it grants users to investigate errors in values of inputs or values of outputs depicted in the diagrams. Therefore, it answers the question of whether there is required development or is sufficient.

Design Specifications:

As stated earlier, the problem was chosen to determine whether a wave is an electromagnetic wave or a mechanical wave. In order to achieve this ambition, four inputs and two outputs have been constructed. For convenience, variable names have been assigned as follows:

```
  T: Traverse wave
  E: Capability of carrying energy
  V: Ability to traverse in a vacuum
  C: Compressional wave
EMW: Electromagnetic wave
 MW: Mechanical wave
```

Subsequently, forming a truth table revealed how functions need to be constructed. By examining the truth table in Table 1. Consecutive functions can be understood with ease.

T	E	V	C	EMW	MW
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	1
1	1	0	1	0	0
1	1	1	0	1	0
1	1	1	1	0	0

Table 1: Truth Table

Two slightly different approaches can be taken from here. Such as, one can use canonical POS (product of sums) or can benefit from canonical SOP (sum of products). Although, two approaches are valid, and after simplifying, they would yield the same result. The less cumbersome and tedious way will be using canonical SOP. Therefore after applying canonical SOP. EMW and

MW functions can be found as:

$$EMW : TEV\bar{C} \tag{1}$$

$$MW : TE\bar{V}\bar{C} + \bar{T}E\bar{V}C \equiv E\bar{V}(T + C)(\bar{C}\bar{T}) \tag{2}$$

Results:

The suggested combinational circuit has been designed via VHDL. The input variables have been initialized in the function named PORT. Later, they were used in BEGIN initializer. Inputs and outputs have been assigned at the PORT function as an instance of *STD_LOGIC* by specifying the parameter input as *in* and output as *out*. The outputs EMW and MW have been assigned to the operations of Eq.1 and Eq.2 subsequently. After coding the functions in the design source, Vivado has generated a schematic of the combinational circuit in Figure 1.

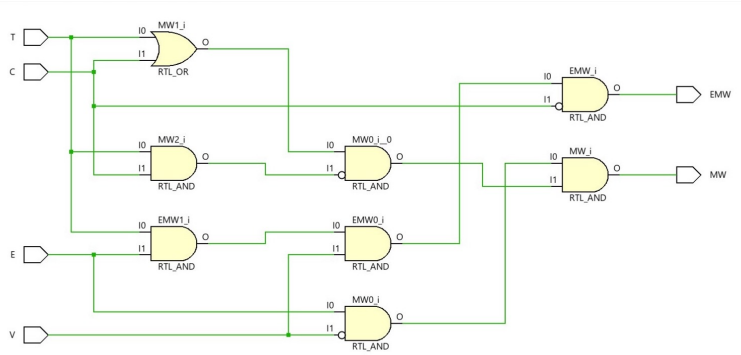


Figure 1: Schematic of circuit.

In furtherance of simulating the signals as a waveform, test-bench code has been written. This process initiated the simulation. By simulating the results of the truth table as a timing diagram in Figures 2a and 2b.

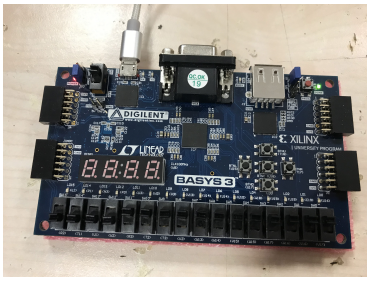
(a) 0101 state. (b) 1110 state.

Figure 2: Timing diagram.

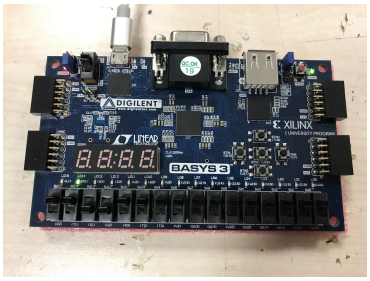
In favor of assigning inputs to switches and outputs to LED indicators on the Basys3. The constraint file has been written. Inputs and outputs have been assigned as follows:

```
   T: Switch R2
   E: Switch T1
   V: Switch U2
   C: Switch W2
 EMW: LED L1
  MW: LED P1
```

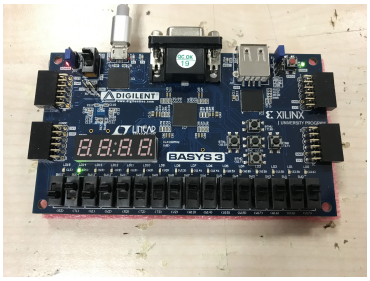
Finally, codes and designs that have been written on Vivado have been imported to Basys3. Results can be seen in Figures 3a, 3b, 3c, and 3d.



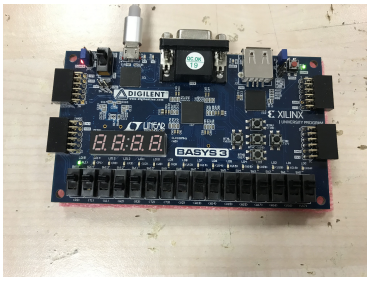
(a) 0000 state.



(b) 1100 state.



(c) 0101 state.



(d) 1110 state.

Figure 3: Design on Basys3.

Conclusion:

The ambition of this lab was to get acquainted with the fundamentals of the design and implementation of a combinational circuit on BASYS3 using VHDL and to get familiar with combinational logic circuits that are time-independent, memoryless circuits whose outputs solely depend on the

combination of inputs. The outcomes of the lab were consistent with the anticipated results.

Appendices:

Code 1: (WAVES_DES.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity WAVES_DES is
    Port ( T : in STD_LOGIC;
           E : in STD_LOGIC;
           V : in STD_LOGIC;
           C : in STD_LOGIC;
           EMW : out STD_LOGIC;
           MW : out STD_LOGIC);
end WAVES_DES;

architecture Behavioral of WAVES_DES is

begin
EMW  <= (T and E and V and (not C));
MW <= E and (not V) and ((T or C) and ( not (T and C)));

end Behavioral;
```

Code 2: (WAVES_TEST.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity WAVES_TEST is
--  Port ( );
end WAVES_TEST;

architecture Behavioral of WAVES_TEST is
component WAVES_DES
port(T : IN STD_LOGIC;
    E : IN STD_LOGIC;
    V : IN STD_LOGIC;
    C : IN STD_LOGIC;
    EMW : OUT STD_LOGIC;
    MW : OUT STD_LOGIC);
end component;
SIGNAL T : STD_LOGIC;
SIGNAL E : STD_LOGIC;
SIGNAL V : STD_LOGIC;
SIGNAL C : STD_LOGIC;
SIGNAL EMW : STD_LOGIC;
SIGNAL MW : STD_LOGIC;


begin
UUT: WAVES_DES PORT MAP(
T => T,
E => E,
V => V,
C => C,
EMW => EMW,
```

```
MW => MW
);
WAVES_TEST : PROCESS
BEGIN

wait for 10 ns;
T<='0';E<='0';V<='0';C<='0';
wait for 10 ns;
T<='0';E<='0';V<='0';C<='1';
wait for 10 ns;
T<='0';E<='0';V<='1';C<='0';
wait for 10 ns;
T<='0';E<='0';V<='1';C<='1';
wait for 10 ns;
T<='0';E<='1';V<='0';C<='0';
wait for 10 ns;
T<='0';E<='1';V<='0';C<='1';
wait for 10 ns;
T<='0';E<='1';V<='1';C<='0';
wait for 10 ns;
T<='0';E<='1';V<='1';C<='1';
wait for 10 ns;
T<='1';E<='0';V<='0';C<='0';
wait for 10 ns;
T<='1';E<='0';V<='0';C<='1';
wait for 10 ns;
T<='1';E<='0';V<='1';C<='0';
wait for 10 ns;
T<='1';E<='0';V<='1';C<='1';
wait for 10 ns;
T<='1';E<='1';V<='0';C<='0';
wait for 10 ns;
T<='1'; E<='1'; V<='0'; C<='1';
wait for 10 ns;
T<='1'; E<='1'; V<='1'; C<='0';
wait for 10 ns;
T<='1'; E<='1'; V<='1'; C<='1';

END PROCESS;

end Behavioral;
```

Code 3: (WAVES_CONST.xdc)

```
set_property PACKAGE_PIN R2 [get_ports {T}]
    set_property IOSTANDARD LVCMOS33 [get_ports {T}]
#set_property PACKAGE_PIN L1 [get_ports {T}]
#    set_property IOSTANDARD LVCMOS33 [get_ports {T}]

set_property PACKAGE_PIN T1 [get_ports {E}]
    set_property IOSTANDARD LVCMOS33 [get_ports {E}]
#set_property PACKAGE_PIN P1 [get_ports {E}]
#    set_property IOSTANDARD LVCMOS33 [get_ports {E}]

set_property PACKAGE_PIN U1 [get_ports {V}]
    set_property IOSTANDARD LVCMOS33 [get_ports {V}]
#set_property PACKAGE_PIN N3 [get_ports {V}]
#    set_property IOSTANDARD LVCMOS33 [get_ports {V}]

set_property PACKAGE_PIN W2 [get_ports {C}]
    set_property IOSTANDARD LVCMOS33 [get_ports {C}]
#set_property PACKAGE_PIN P3 [get_ports {C}]
#    set_property IOSTANDARD LVCMOS33 [get_ports {C}]

set_property PACKAGE_PIN L1 [get_ports {EMW}]
    set_property IOSTANDARD LVCMOS33 [get_ports {EMW}]

set_property PACKAGE_PIN P1 [get_ports {MW}]
    set_property IOSTANDARD LVCMOS33 [get_ports {MW}]
```

References:

- https://digilent.com/reference/learn/programmable-logic/tutorials/basys-3-programming-guide/start
- https://www.xilinx.com/support/documents/university/Vivado-Teaching/HDL-Design/2015x/Verilog/docs-pdf/Vivado_tutorial.pdf
- https://github.com/CankutBoraTuncer/Bilkent-EEE102-Labs