

## Lab 2 – Behavioral and Dataflow VHDL

VHDL is a hardware description language. We will use this language in the laboratory to implement logic. This language will be used in several digital hardware courses that follow Switching and Logic, so it is worth the effort to learn the language.

VHDL as all hardware description languages describes what happens in a concurrent or parallel manner. All the gates in the EPM7128S are all working at the same time and in a concurrent manner. This is opposed to most software languages that describe what is to happen in a sequential or linear manner. You describe what happens first, then next and so on.

Within VHDL we can describe the logic in three different manners. These three different architectures are:

- Behavioral – describes how the output is derived from the inputs using structured statements.
- Dataflow – describes how the data flows from the inputs to the output most often using NOT, AND and OR operations.
- Structural – describes how gates are interconnected similar to schematic approach.

In this laboratory exercise you are only required to implement the VHDL using a behavioral architecture and dataflow architecture. For the interested students you may also implement the structural architecture.

VHDL has two different statements associated with every block of logic. One is the ENTITY statement, which just specified the inputs and the outputs to the block of logic. The other is the ARCHITECTURE statement, which must be associated with an ENTITY statement. As an example here is a block of logic that implements D28 discussed in the textbook:

```
ENTITY calendar IS
  PORT(
    LEAP,M8,M4,M2,M1   : IN BIT;
    D28                 : OUT BIT);
END calendar;
```

```
ARCHITECTURE dataflow OF calendar IS
  BEGIN
    D28 <= NOT LEAP AND NOT M8 AND NOT M4 AND M2 AND NOT M1;
  END dataflow;
```

Note the ENTITY statement only specifies the inputs and the outputs. While the architecture describes how the output D28 is derived from the inputs M8, M4, M2 and M1.

## ***hex7seg as VHDL Behavioral***

To complete a behavioral type implementation of logic in VHDL using Quartus II perform the following steps:

1. Create a project called behavioral with the top-level file specified as hex7seg. Place the project under your workspace directory or folder. In the past many errors have been traced to projects spanning several folders. It is suggested that each project be placed in its own folder. Do not complain about wasted time if you do not follow these suggestions.
2. Set unused pins as tri-state inputs. **Very important** so input/output pins drivers are not destroyed!
3. Create a New file of VHDL type.
4. Enter the following VHDL description in the newly created VHDL file:

```
1 -- Hexadecimal to 7 Segment Decoder for LED Display
2
3 ENTITY hex7seg IS
4
5     PORT(hex_digit      : IN  BIT_VECTOR(3 DOWNTO 0);
6          segment       : OUT BIT_VECTOR(0 TO 6));
7
8 END hex7seg;
9
10 ARCHITECTURE behavioral OF hex7seg IS
11
12 BEGIN
13     WITH hex_digit SELECT
14         -- HEX to 7 Segment Decoder for LED Display
15         -- Hex-digit is the four bit binary value to display in hexadecimal
16         segment <= "1111110" WHEN "0000",
17                   "0110000" WHEN "0001",
18                   "1101101" WHEN "0010",
19                   "1111001" WHEN "0011",
20                   "0110011" WHEN "0100",
21                   "1011011" WHEN "0101",
22                   "1011111" WHEN "0110",
23                   "1110000" WHEN "0111",
24                   "1111111" WHEN "1000",
25                   "1111011" WHEN "1001",
26                   "1110111" WHEN "1010",
27                   "0011111" WHEN "1011",
28                   "1001110" WHEN "1100",
29                   "0111101" WHEN "1101",
30                   "1001111" WHEN "1110",
31                   "1000111" WHEN "1111";
32 END behavioral;
33
```

**Listing 1 Hex7Seg Behavioral VHDL**

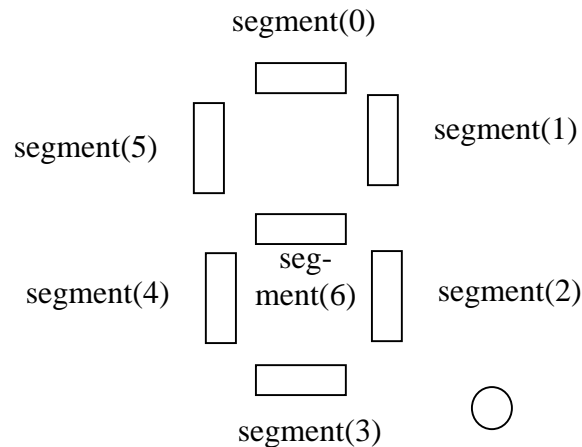
5. Save the VHDL description as *hex7seg.vhd* and perform Analysis & Synthesis until there are 0 errors and 0 warnings by correcting typographical errors. Note the entity and filename should be the same minus the file extension for the entity name.

6. Assign the pins locations using the Assignment Editor. See Table 1 given below for the signal and pin number associations.

**Table 1 Hex7Seg I/O Pin Assignments**

Pin	Type	Name	PLDT-2	Pin	Type	Name	PLDT-2
34	Input	hex_digit[3]	S1-1	58	Output	segment[0]	DS1-A
33	Input	hex_digit[2]	S1-2	60	Output	segment[1]	DS1-B
36	Input	hex_digit[1]	S1-3	61	Output	segment[2]	DS1-C
35	Input	hex_digit[0]	S1-4	63	Output	segment[3]	DS1-D
				64	Output	segment[4]	DS1-E
				65	Output	segment[5]	DS1-F
				67	Output	segment[6]	DS1-G

7. Perform a full compile from Analysis & Synthesis through Timing Analysis. There should be 0 errors and 0 warnings. Correct the problem if any errors or warnings are displayed.
8. Create a Vector Waveform File and save it as *hex7seg.vwf*. The Vector Waveform File should include all 16 combinations of the four inputs in 100 ns steps.
9. Set the End Time to 1600 ns under the Edit menu and then perform a functional simulation. Check the simulation for correctness. The sequence of outputs should display 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, and F.



10. Program the EPM7128SLC84-15 on the PLDT-2 board and exercise the DIP switch SK-1 while observing seven-segment display DS-1. **Make sure the J3 jumper is spanning the header pins.** Note toward the edge of the board is high or “1” and toward the center of the board is low or “0”. Record the results in the Table 2 truth table. Color in each segment that is on.
11. You may find it convenient to print the *hex7seg.vhd* file, the pin assignments screen and the simulation waveform at this time for use in the laboratory report.

**Table 2 Hexadecimal to 7-Segment Truth Table**

S1-1	S1-2	S1-3	S1-4	DS-1	S1-1	S1-2	S1-3	S1-4	DS-1
0	0	0	0		0	1	0	0	
0	0	0	1		0	1	0	1	
0	0	1	0		0	1	1	0	
0	0	1	1		0	1	1	1	

S1-1	S1-2	S1-3	S1-4	DS-1	S1-1	S1-2	S1-3	S1-4	DS-1
1	0	0	0		1	1	0	0	
1	0	0	1		1	1	0	1	
1	0	1	0		1	1	1	0	
1	0	1	1		1	1	1	1	

**Discussion of Listing 1.**

The ENTITY statement specifies the four inputs and the seven outputs. The four inputs can be recognized by the IN keyword and the seven outputs can be recognized by the OUT keyword. Both the input and outputs signals are of type BIT\_VECTOR. Type BIT signals may only have the values of 0 or 1 for both synthesis and simulation. Both BIT\_VECTORS are treated in blocks by placing double quotes around the binary values.

The architecture fills in the “black box” of the entity statement. The architecture name is behavioral in this listing. The architecture must be associated with an entity, which is hex7seg in this listing. The signals segment(0) to segment(6) must have values assigned to them. The <= operator is used to make this assignment. The WITH-SELECT statement is a behavioral description of the segments. It is behavioral in that the output segments are only specified in terms of the input bits, hex\_digit. As an example, the output segments are assigned the value “1110000” when the input hex\_digit is “0111”. Note the WITH-SELECT statement looks like a truth table with the outputs on the left and the inputs on the right.

***Dec7Seg as VHDL Dataflow***

To complete a dataflow type implementation of logic in VHDL using Quartus II perform the following steps:

1. Create a project called dataflow with the top-level file specified as Dec7Seg. Place the project under your LAB\_DAY directory or folder.
2. Set unused pins as tri-state inputs. **Very important** so input/output pins drivers are not destroyed!
3. Create a New file of VHDL type
4. Enter the following VHDL description:

```

1 ENTITY Dec7Seg IS
2   PORT (
3     D   : IN   BIT_VECTOR(3 DOWNTO 0);
4     S   : OUT  BIT_VECTOR(0 TO 6));
5 END Dec7Seg;
6
7 ARCHITECTURE dataflow OF Dec7Seg IS
8 BEGIN
9   S(0) <= (D(1) AND D(0)) OR (D(2) AND NOT D(1) AND D(0)) OR
10  (NOT D(2) AND NOT D(0)) OR (D(3));
11  S(1) <= (NOT D(2) AND D(0)) OR (D(1) AND D(0)) OR
12  (NOT D(1) AND NOT D(0)) OR (NOT D(2) AND NOT D(0));
13  S(2) <= (NOT D(2) AND D(0)) OR (D(1) AND D(0)) OR
14  (NOT D(1) AND NOT D(0)) OR (D(2) AND NOT D(0)) OR
15  (D(2) AND NOT D(1) AND D(0));
16  S(3) <= (D(1) AND NOT D(0)) OR (NOT D(2) AND D(1)) OR
17  (D(2) AND NOT D(1) AND D(0)) OR (NOT D(2) AND NOT D(0));
18  S(4) <= (D(1) AND NOT D(0)) OR (NOT D(2) AND NOT D(0));
19  S(5) <= (NOT D(1) AND NOT D(0)) OR (D(2) AND NOT D(0)) OR
20  (D(2) AND NOT D(1) AND D(0)) OR (D(3));
21  S(6) <= (NOT D(2) AND D(1)) OR (D(2) AND NOT D(0)) OR
22  (D(2) AND NOT D(1) AND D(0)) OR (D(3));
23 END dataflow;
24

```

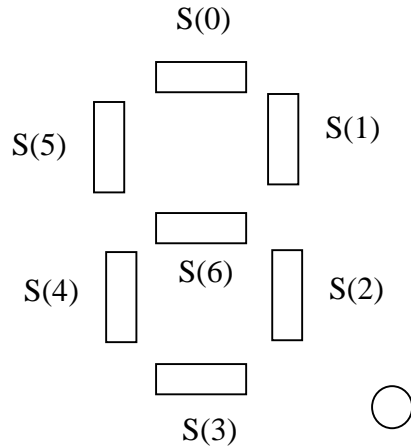
**Listing 2 Dec7Seg Dataflow VHDL**

5. Save the VHDL description as *dec7seg.vhd* and perform Analysis & Synthesis until there are 0 errors and 0 warnings by correcting typographical errors.
6. Assign the pins locations using the Assignment Editor. See Table 3 given below for the signal and pin number associations.

**Table 3 Dec7Seg I/O Pin Assignments**

Pin	Type	Name	PLDT-2	Pin	Type	Name	PLDT-2
34	Input	D[3]	S1-1	69	Output	S[0]	DS2-A
33	Input	D[2]	S1-2	70	Output	S[1]	DS2-B
36	Input	D[1]	S1-3	73	Output	S[2]	DS2-C
35	Input	D[0]	S1-4	74	Output	S[3]	DS2-D
				76	Output	S[4]	DS2-E
				75	Output	S[5]	DS2-F
				77	Output	S[6]	DS2-G

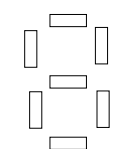
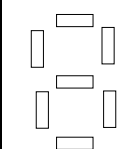
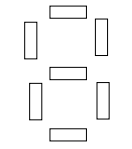
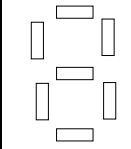
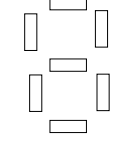
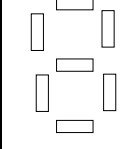
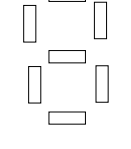
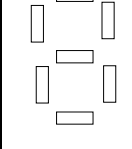
7. Perform a full compile from Analysis & Synthesis through Timing Analysis. There should be 0 errors and 0 warnings. Correct the problem if any errors or warnings are displayed.
8. Create a Vector Waveform File and save as *dec7seg.vwf*. The Vector Waveform File should include all 16 combinations of the four inputs in 100 ns steps.
9. Perform a functional simulation with an ending time of 1600 ns. Check the first ten steps of the simulation for correctness. The sequence of the outputs should display 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.



10. Program the EPM7128SLC84-15 on the PLDT-2 board and exercise the DIP switch SK-1 while observing seven-segment display DS-2. Make sure the J3 jumper spans the header pins. Note toward the edge of the board is high or “1” and toward the center of the board is low or “0”. Record the results in the Table 4 truth table. Color in each segment that is on.

**Table 4 Decimal to 7-Segment Truth Table**

S1-1	S1-2	S1-3	S1-4	DS-1	S1-1	S1-2	S1-3	S1-4	DS-1
0	0	0	0		0	1	0	0	
0	0	0	1		0	1	0	1	
0	0	1	0		0	1	1	0	
0	0	1	1		0	1	1	1	

S1-1	S1-2	S1-3	S1-4	DS-1	S1-1	S1-2	S1-3	S1-4	DS-1
1	0	0	0		1	1	0	0	
1	0	0	1		1	1	0	1	
1	0	1	0		1	1	1	0	
1	0	1	1		1	1	1	1	

11. You may find it convenient to print the dec7seg.vhd file, the pin assignments screen and the simulation waveform at this time for use in the laboratory report.

#### Discussion of Listing 2.

As before the ENTITY statement specifies the four inputs and the seven outputs. The four inputs can be recognized by the IN keyword and the seven outputs can be recognized by the OUT keyword. Both the input and outputs signals are of type BIT\_VECTOR. The input signals have been shortened to D and the outputs to S to reduce the typing in the architecture statement.

The architecture name is dataflow in this listing. The architecture must be associated with an entity, which is dec7seg in this listing. The signals S(0) to S(6) must have values assigned to them. The <= operator is used to make this assignment. Each element of S is a function of the inputs D(3), D(2), D(1) and D(0). These inputs are interconnected with the NOT, AND and OR operators are used in VHDL. Note that each group of ANDs is enclosed in parenthesis. This is because in VHDL, AND and OR have the same precedence. NOT has a higher precedence than either AND or OR. Without the parenthesis the equation

$$S(4) <= ((D1) \text{ AND NOT } D(0)) \text{ OR } (\text{NOT } D(2) \text{ AND NOT } D(0));$$

would read as:

$$S(4) \leq (D1) \text{ AND NOT } D(0) \text{ OR NOT } D(2) \text{ AND NOT } D(0);$$

This would evaluate as:

$$S(4) \leq (((D1) \text{ AND NOT } D(0)) \text{ OR NOT } D(2)) \text{ AND NOT } D(0);$$

### **Report Format**

This laboratory exercise requires an informal report. The report should contain the ten items listed below.

1. Title page (see page 11)
2. VHDL listing of Hex7Seg.vhd
3. Pin assignment editor output for Hex7Seg
4. Simulation Output waveform showing all 16 conditions
5. Table 2 Hexadecimal to 7-Segment Truth Table
6. VHDL listing for Dec7Seg.vhd
7. Pin assignment editor output for Dec7Seg
8. Simulation output waveforms for Dec7Seg showing all 10 conditions
9. Table 4 Decimal to 7-Segment Truth Table
- 10. Convert Figure 2 in the Calendar handout on the course web page to a behavioral type VHDL listing. You don't need to enter your VHDL into Quartus II. Use the keyword OTHERS for the WHEN case that does not include the encoding of January through December. As an example "0000" WHEN OTHERS;**

For your convenience, the table has been duplicated and enhanced below. The encoding don't care encoding of 0000 -, 1101 -, and 11-- - may be taken care of by the OTHERS case. All months have been expanded to the leap year and non-leap year case.

Month	Leap	D28	D29	D30	D31	Month	Leap	D28	D29	D30	D31
0000	-	-	-	-	-	0111	0	0	0	0	1
0001	0	0	0	0	1	0111	1	0	0	0	1
0001	1	0	0	0	1	1000	0	0	0	0	1
0010	0	1	0	0	0	1000	1	0	0	0	1
0010	1	0	1	0	0	1001	0	0	0	1	0
0011	0	0	0	0	1	1001	1	0	0	1	0
0011	1	0	0	0	1	1010	0	0	0	0	1
0100	0	0	0	1	0	1010	1	0	0	0	1
0100	1	0	0	1	0	1011	0	0	0	1	0
0101	0	0	0	0	1	1011	1	0	0	1	0
0101	1	0	0	0	1	1100	0	0	0	0	1
0110	0	0	0	1	0	1100	1	0	0	0	1
0110	1	0	0	1	0	1101	-	-	-	-	-
						111-	-	-	-	-	-

**Done?**

Things that you might consider doing on your own are:

1. Modify Hex7Seg so that the sequence is 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, P, and blank. Then perform a complete implementation to include pin assignments, simulation, and program/verify on the PLDT-2 system.
2. Investigate the use of 7447 device in the others/maxplus2 library.

## Lab 2 – Behavioral and Dataflow VHDL

Group Member #1 \_\_\_\_\_  
Please Print

Group Member #2 \_\_\_\_\_  
Please Print

Group Member #3 \_\_\_\_\_  
Please Print

Lab Period: \_\_\_\_\_  
Monday ... Friday

Lab Station: \_\_\_\_\_  
1 through 12

### Check List:

√	Description	Score
	VHDL listing of Hex7Seg.vhd	
	Pin assignment editor output for Hex7Seg	
	Simulation Output for Hex7Seg	
	Table 2 Hexadecimal to 7-Segment Truth Table	
	VHDL listing for Dec7Seg.vhd	
	Pin assignment editor output for Dec7Seg	
	Simulation output waveforms for Dec7Seg	
	Table 4 Decimal to 7-Segment Truth Table	
	Calendar Behavioral VHDL	
	<b>Total</b>	

*The work presented is the sole work of the stated group members. Representing the work of others as your own is plagiarism and is punishable by failure of the course for the slightest infraction.*

Casey Bakula/Yang Wang

