

ELE444 Advanced Electronic Design LAB3

Exercise - D/A Conversion

In this exercise you will create a project to demonstrate the functionality of the D/A converter. The datasheet for this MOSA Electronics MS6313 device can be obtained from the TREX CD Rom or the web. The following address can be used to obtain the datasheet: <http://www.mosanalog.com/datasheets/ms6313.pdf>. Peruse the datasheet to get an understanding of the operation of the device.

After learning how the 16-bit D/A converter receives its data serially over three pins, you should now try the example code below to see how a square wave is generated. You should program the following code and view the output from the D/A converter on the oscilloscope. (Note: Be sure to assign the proper pins for the D/A converter. Also, use the 27MHz clock for the clock input.)

testda.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity testda is
    PORT (
        clock: in std_logic;
        audio_lrck: out std_logic;
        audio_bck: out std_logic;
        audio_data: out std_logic);
end testda;

architecture a of testda is
    signal counter : std_logic_vector(15 downto 0);
    signal dacounter : std_logic_vector(15 downto 0);
begin
    --divide incoming clock
    process(clock)
    begin
        if(clock'event and clock = '1') then
            counter <= counter + 1;
        end if;
    end process;

    dacounter(14 downto 7) <= "11111111" when counter(12) = '1' else
        "00000000";
```

```

dacounter(15) <= '1';
dacounter(6 downto 0) <= "0000000";

with counter(4 downto 1) select
audio_data <= dacounter(15) WHEN "0000",
                                     dacounter(14) WHEN "0001",
                                     dacounter(13) WHEN "0010",
                                     dacounter(12) WHEN "0011",
                                     dacounter(11) WHEN "0100",
                                     dacounter(10) WHEN "0101",
                                     dacounter(9)  WHEN "0110",
                                     dacounter(8)  WHEN "0111",
                                     dacounter(7)  WHEN "1000",
                                     dacounter(6)  WHEN "1001",
                                     dacounter(5)  WHEN "1010",
                                     dacounter(4)  WHEN "1011",
                                     dacounter(3)  WHEN "1100",
                                     dacounter(2)  WHEN "1101",
                                     dacounter(1)  WHEN "1110",
                                     dacounter(0)  when others;

audio_lrck <= counter(5);
audio_bck  <= counter(0);
end a;

```

Does the output look as expected? What happens to the output if you change “counter(12)” in the code to “counter(8)” where “dacounter(14 downto 7)” is assigned?

Exercise - Hierarchical Design

In this exercise, an example of hierarchical design will be given. Many designs must take a hierarchical design approach to keep the project organized and easier to maintain. The example that follows creates a top level entity that adds two numbers and displays the result on the seven segment displays. Instead of explicitly stating the look-up table using a select statement, this example uses a ROM module. The ROM module could easily be replaced by an entity with the same inputs and outputs encapsulating the select statement previously used. Now, the ROM module we create can be reused multiple times by creating multiple instances with a different name and signal mapping.

First, let’s start by creating our top level entity. Type in the following code:

lab3adder.vhd

```

library ieee;
use ieee.std_logic_1164.all;

```

```

use ieee.std_logic_unsigned.all;

entity lab3adder is
  PORT(
    clock50MHz : in std_logic;
    num_one : in std_logic_vector(1 downto 0);
    num_two : in std_logic_vector(1 downto 0);
    display : out std_logic_vector(6 downto 0));
end lab3adder;

architecture a of lab3adder is
  component myrom
    PORT
      (
        address : in std_logic_vector(3 downto 0);
        clock : in std_logic;
        q : out std_logic_vector(6 downto 0)
      );
  end component;

  signal myaddress : std_logic_vector(3 downto 0);
begin

  myrom_inst : myrom PORT MAP (
    address => myaddress,
    clock => clock50MHz,
    q => display
  );

  myaddress <= ("00" & num_one) + ("00" & num_two);

end;

```

Now, a ROM module should be created to provide the code for the referenced component in your top level entity. To do this, we will use the MegaWizard Plug-In Manager. Start the wizard by clicking Tools->MegaWizard Plug-In Manager from the menu at the top. Next, select the option Create a new custom megafunction variation and click Next. Using the following pictures as guides, complete the wizard's steps.

The last step specifies which files to create with the wizard. The myrom.cmp file gives you an example of defining a component to use in the design. Once you have a component defined inside your VHDL source file, you can create instances of the "black box". An example of how to create an instance is given in the file myrom_inst.vhd. Basically, a name is given to the instance and which inputs and outputs of the "black box" maps to the signals of the current entity.

Using any text editor outside of Quartus, create a myrom.mif file to hold the contents of the ROM. The following shows the contents to type into the file. This should be modified to include the other seven segment values to display. Only the first three values are given.

myrom.mif

```
WIDTH=7 ;
DEPTH=16 ;

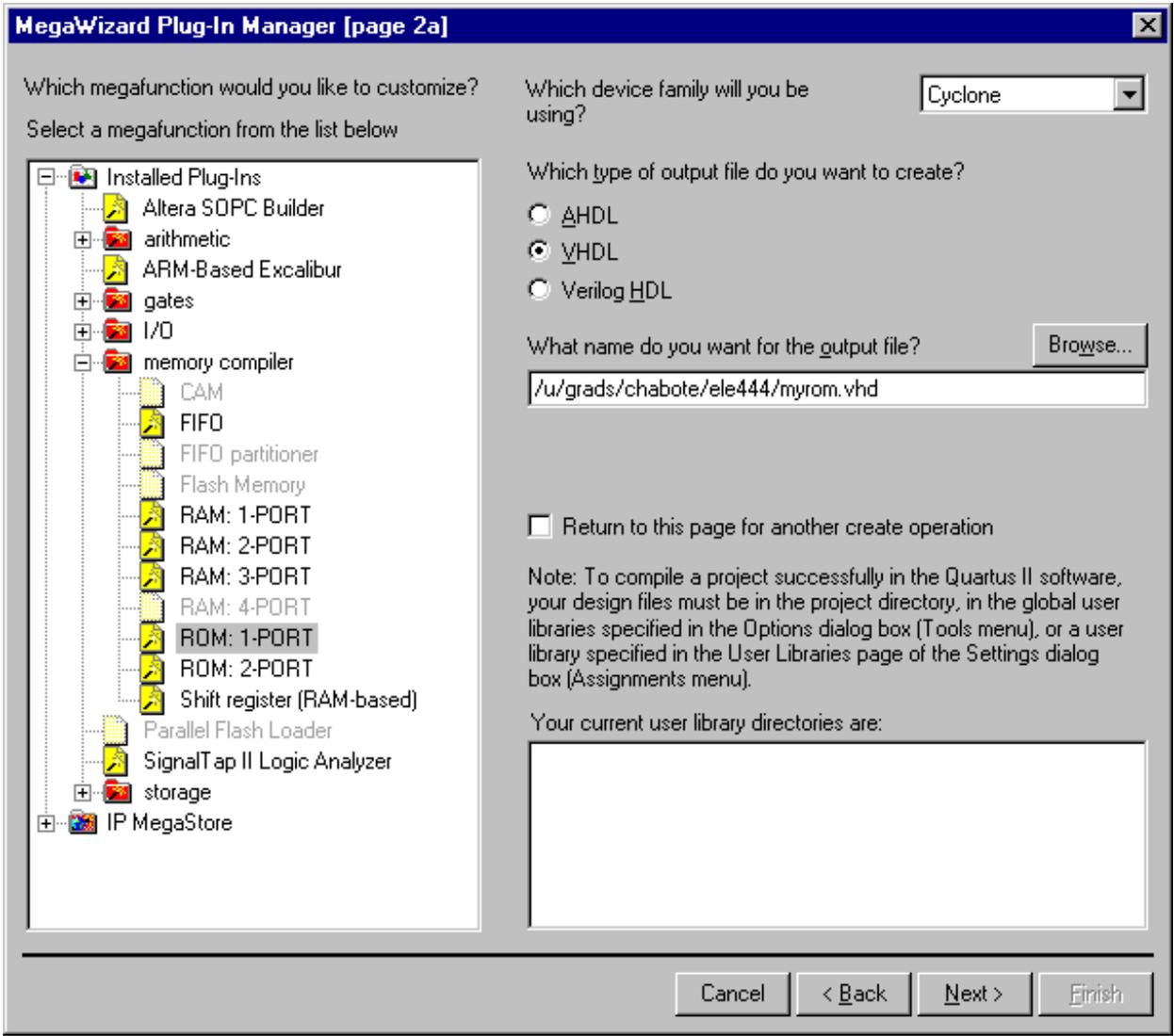
ADDRESS_RADIX=HEX ;
DATA_RADIX=HEX ;

CONTENT BEGIN
    00 : 40 ;
    01 : 4F ;
    02 : 28 ;
    [03..0F] : 00 ;
END ;
```

Realize the design on the development board to verify that the adder functions properly.

Assignment

Aside from generating a square wave with the D/A converter, it is easy to create a sawtooth wave. Your task is to create a sawtooth waveform using the skills acquired from the exercises. Demonstrate the functionality of your circuit to the TA.



MegaWizard Plug-In Manager - ROM: 1-PORT [page 3 of 6]

Currently selected device family: **Cyclone**

Family supports LPM_ROM only in backward-compatibility mode. Altera recommends using ALTSYNCRAM wizard.

How wide should the 'q' output bus be? **7** bits

How many 7-bit words of memory? **16** words

What should the RAM block type be?

Auto
 M512
 M4K
 M-RAM
 LDCs
 Options...

Set the maximum block depth to **Auto** words

What clocking method would you like to use?

Single clock
 Dual clock: use separate 'input' and 'output' clocks

Resource Estimate
1 M4K

Documentation... Cancel < Back Next > Finish

MegaWizard Plug-In Manager - ROM: 1-PORT [page 4 of 6]

Which ports should be registered?

'data' input port
 'address' input port
 'q' output port

Create one clock enable signal for each clock signal. All registered ports are controlled by the enable signal(s). More Options...

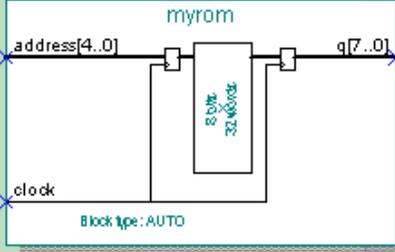
Create a byte enable port
 What is the width of a byte for byte enable? **8** bits

Create an 'aclr' asynchronous clear for the registered ports. More Options...

Resource Estimate
1 M4K

Documentation... Cancel < Back Next > Finish

MegaWizard Plug-In Manager - ROM: 1-PORT [page 5 of 6]



Do you want to specify the initial content of the memory?

No, leave it blank

Initialize memory content data to XXX...X on power-up in simulation

Yes, use this file for the memory content data
 (You can use a Hexadecimal (Intel-format) File [.hex] or a Memory Initialization File [.mif])

Browse...

File name:

The initial content file should conform to which port's dimensions?

Allow In-System Memory Content Editor to capture and update content independently of the system clock

The 'Instance ID' of this ROM is:

Resource Estimate
1 M4K

Documentation... Cancel < Back Next > Finish

MegaWizard Plug-In Manager - ROM: 1-PORT [page 6 of 6] -- Summary



When the 'Finish' button is pressed, the MegaWizard Plug-In Manager will create the checked files in the following list. You may choose to include or exclude a file by checking or unchecking its corresponding checkbox, respectively. The state of checkboxes will be remembered for the next MegaWizard Plug-In Manager session.

The MegaWizard Plug-In Manager will create these files in the directory:
/u/grads/chabote/ele444/

File	Description
<input checked="" type="checkbox"/> myrom.vhd	Variation file
<input type="checkbox"/> myrom.inc	AHDL Include file
<input checked="" type="checkbox"/> myrom.cmp	VHDL Component declaration file
<input type="checkbox"/> myrom.bsf	Quartus symbol file
<input checked="" type="checkbox"/> myrom_inst.vhd	Instantiation template file

Documentation... Cancel < Back Next > Finish