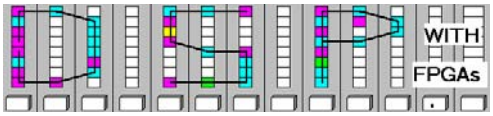
Lastname: _____

5Digit SS:_____

LABORATORY
Number Systems and Quantization



LAB : NUMBER SYSTEMS AND QUANTIZATION
(10 points)

In this lab you will be introduced to fractional number systems, quantization and IP block design. In the **pre-lab** you will compute via "pencil-and-paper" the results you later expect in your design implementation. In the **design part** you will design and simulate a 4-bit bus implementation of a signed, unsigned, and fractional number system. In the second part of the design lab a numerical controlled oscillator (NCO) using Altera's Intellectual Property (IP) block is developed. A GUI-based wizard will generate a sine wave generator similar to the one designed in lab 1.

Lab Objectives

After completing this lab you should be able to

- understand the difference of signed and unsigned numbers systems
- determine minimum and maximum values in integer and factional number systems
- compute quantization error
- design and simulate a circuit using Quartus II

Pre-lab (3 points)

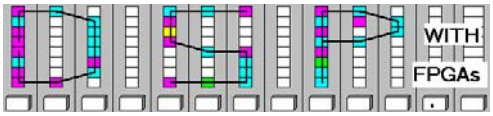
1. The data range of a *B* bit **unsigned** format is $[0...2^B-1]$.
   a. For *B*=4 determine the maximum number that can represented: X_{max} =
   b. For *B*=4 determine the minimum number that can represented: X_{min} =

2. The data range of a *B* bit **signed** format is $[-2^{B-1}... 2^{B-1}-1]$.
   a. For *B*=4 determine the maximum number that can represented: Y_{max} =
   b. For *B*=4 determine the minimum number that can represented: Y_{min} =

3. For the following **signed fractional** numbers with 1 integer and 3 fractional bits determine:
   a. the decimal weight ($+/-2^k$) for each digit in the signed 1.3 format:

   ___ . ___ ___ ___

 © 2012 Dr. Uwe Meyer-Baese



b. the largest positive number in binary format B_{max} = _____

c. the equivalent of B_{max} as decimal number D_{max} = _____

d. the smallest positive number in binary format: $B_{smallest}$ = _____

e. the equivalent of $B_{smallest}$ as decimal number $D_{smallest}$ = _____

f. the minimum negative number in binary format: B_{min} = _____

g. the equivalent of B_{min} as decimal number D_{min} = _____

4. the table values below for the 1.3 format (rounding: truncation)

Decimal Value	Binary 1.3 format	Decimal 1.3 equivalent	Quantization error (Dec. equivalent-Dec. Value)
+ 0.625	__ . __ __ __		
- 0.875	__ . __ __ __		
1/16	__ . __ __ __		
1/3	__ . __ __ __		
D_{max}=	__ . __ __ __		
D_{max}+ $D_{smallest}$ =	__ . __ __ __		

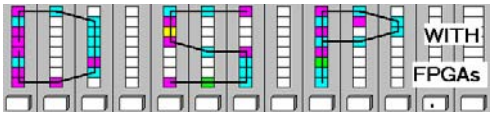
<u>Make sure your name and SS is on all pages you turn in!</u>

 © 2012 Dr. Uwe Meyer-Baese

Lastname: _____

5Digit SS:_____

LABORATORY
Number Systems and Quantization



VHDL Design-lab

Follow the directions below to implement an NCO using Altera's IP block.

.

A. Getting Started

If you are in B114 or the digital logic lab:

1. On the desktop double click on **Engineering folder**.

2. From the top icon list on the Desktop click on the **Quartus II** icon  to start **Quartus.**

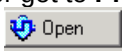
3. You should not save anything on the local hard disk. You will have to use a memory stick, or your "mapped" home directory to save the files. Create or use the folder named **DSPwFPGAs** on your mapped network or jump drive.

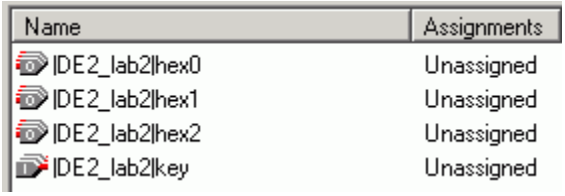
B. Simulating Signals using Unsigned, Signed and Fractional Radix System

1. Download the file `DE2_lab2.zip` from the class webpage and put the file in the **DSPwFPGAs** folder. Unzip the file and you should see the following files: `DE2_lab2.vhd, DE2_lab2.qsf, DE2_lab2.qpf, audio_pll.vhd, audio_pll.qip, audio_pll.cmp, i2c_config.vhd, i2c_controller.vhd, and audio_dac.vhd.`

2. Use **File->OpenProject->**`DE2_lab2.qpf` to open the project file in **Quartus II**. Load the top level file `DE2_lab2.vhd` (i.e., **File->Open->**`DEC_lab1.vhd`). You should see after a moment a small design that connects the input `key` to the outputs `hex0, hex1` and `hex2`. Complete the Header with your name and date.

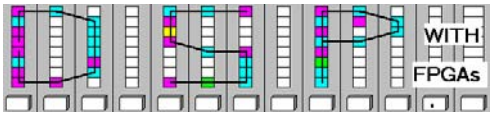
3. Compile this design in **Quartus II** by pressing the  icon, hitting **Ctrl+L**, or select **Processing->Start Compilation**.

4. To open the simulator got to **Processing->Simulator Tool**. The **Simulator Too**l window will pop up. Now select open  in the lower right corner. A window called `Waveform1.vwf` will pop up. Now double click in the white section under Name to open the **Insert Node or Bus** window. Click on the **Node Finder...** button. In the **Node Finder** use as **Filter: Pins: all** then press the **List** button. Now select the following nodes.



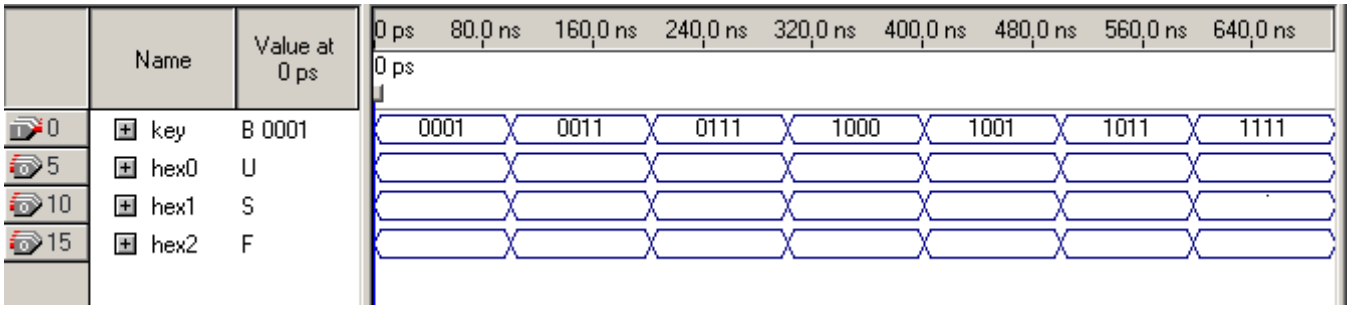
Using **CTRL+Mouse** left click to select in the **Nodes Found** section these signal and then use the  button to transfer these nodes to the **Selected Nodes** window. Then click ok and these signals should now be seen in the `Waveform1.vwf` window. Select `key` with the mouse and

 © 2012 Dr. Uwe Meyer-Baese



move if to the first row. Double click the Bus icons left to the name and set the **Node Properties** windows opens. Set the **Radix** for `key` to **Binary**; `hex0` to **Unsigned Decimal;** the node `hex1` to **Signed Decimal**; and the node `hex2` to **Fractional**.

5. Save the waveform file in the project directory under the name `DE2_lab2.vwf`. Set **Edit->End Time** to 700 ns. Now use the **View->Fit in Window** to see the full 700 ns simulation. Select the `key` waveform in the 0 to 100 ns range and set the binary value to 0001. Continue this for the 100-200 ns next section with value 0011, etc. Your wave form file should looks finally as follows:



6. Now go back to the **Simulator Tool** window. Run the **Generate Functional Simulation Netlist**, select **Overwrite simulation input file with simulation results** and deselect **Automatically add pins to simulation output waveforms**. Finally hit the **Start** button to run the simulation. Answer with **Yes** to overwrite the current simulation results.

7. Fill in the simulation results above for `hex0, hex1`, and `hex2`. Circle the largest positive number for all three **Radix** systems, and compare to the data from the Prelab.

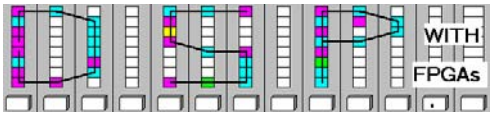
C. Generate the NCO using the MegaWizard Plug-In Manager

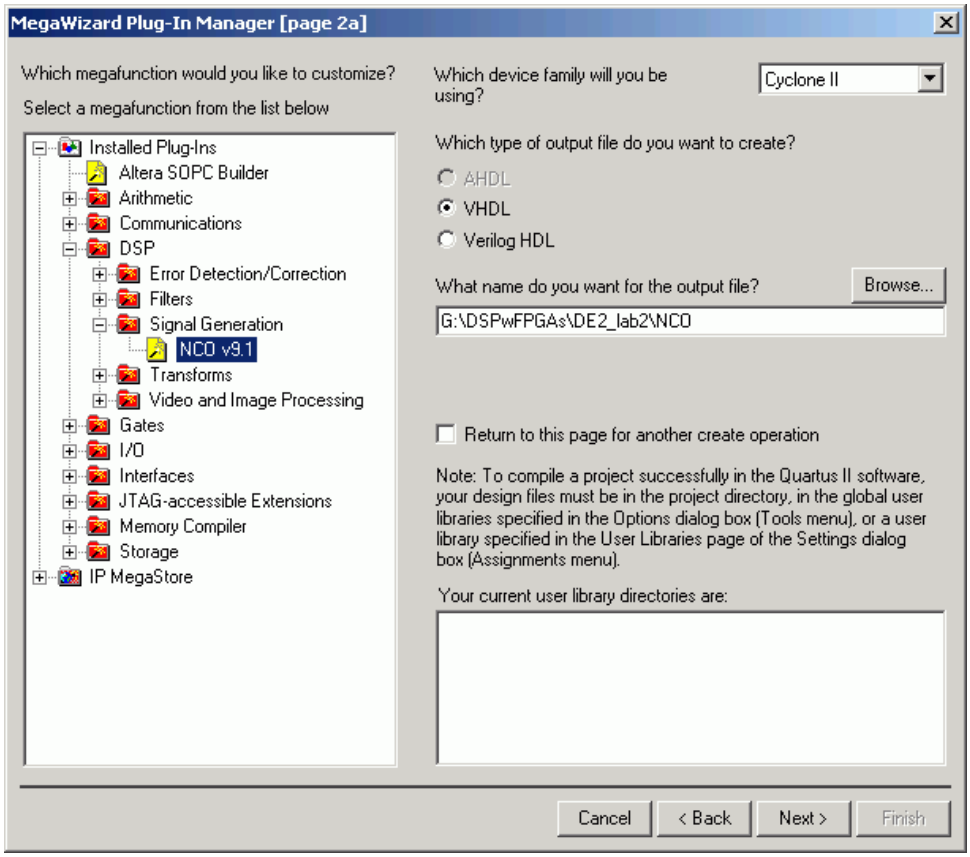
1. Use **File->OpenProject->**`DE2_lab2.qpf` to open the project file in **Quartus II**. Load the top level file `DE2_lab2.vhd` (i.e., **File-> Open->**`DEC_lab1.vhd`). You should see after a moment the small design.

2. To generate the core start the Wizard via **Tools->MegaWizard Plug-In Manager ...** Select **Create a new custom Megafunction variation** then click 

. In the page 2a select **DSP->Signal Generation->NCO v9.1** as Megafunction. As file type select **VHDL** and as name `NCO`.

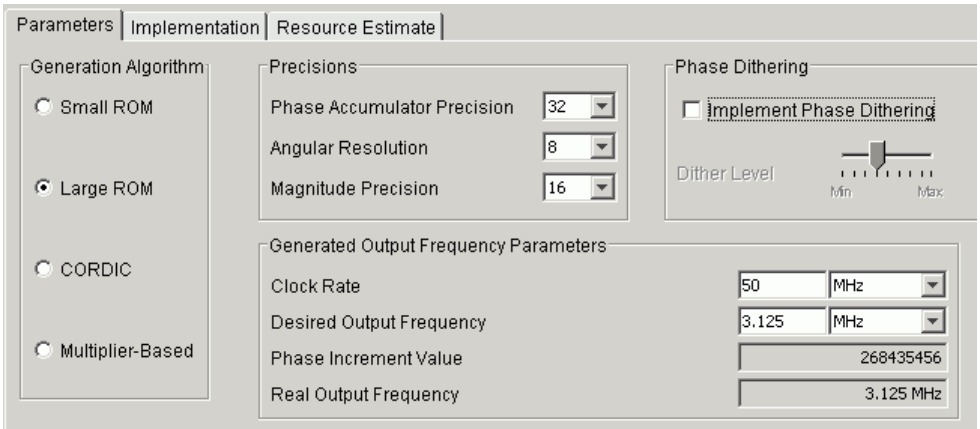
 © 2012 Dr. Uwe Meyer-Baese





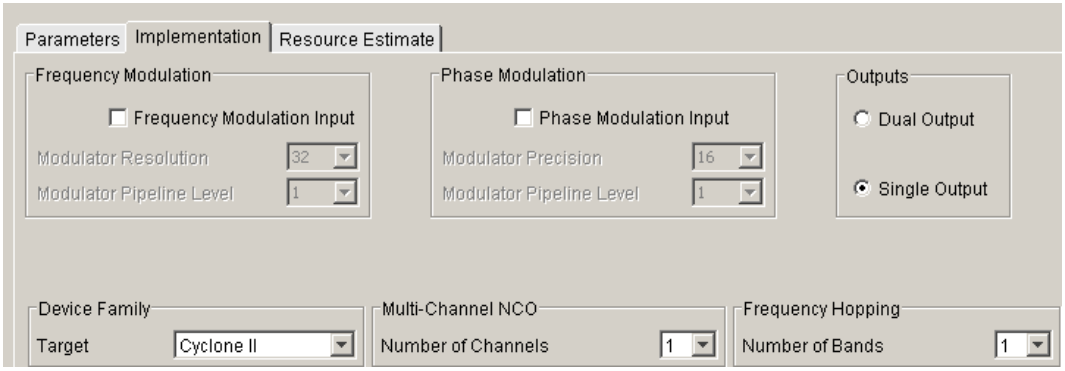
3. Then click 

and the IP ToolBench  will load. You may want to read the **About** and the **Documentation** first. Next click on **Step1: Parameterize** and make the following setting in the first **Parameters** windows of the core.



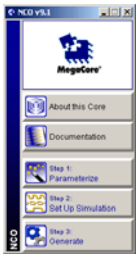
 © 2012 Dr. Uwe Meyer-Baese

4. Click on the **Implementation** tab and select single output, i.e.



5. Complete the following table using the data shown in **Resource Estimate**, i.e., third tab.

Parameter	**Estimated Value**
Number of LEs	
Number of Memory Bits	
Number of M4Ks	
Number of 9-bit DSP Elements	

6. Now go back to the NCO v9.1 ToolBench window 

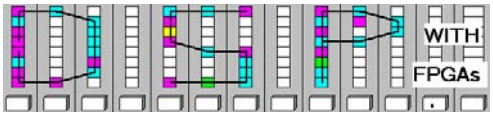
Skip **Step 2** and click on **Step 3: Generate.**, Several output files are generated including a `NCO.html` file that gives an overview over the generated files. Study the short description given in the html file. Finally click **EXIT** to exit the ToolBench.

D. Integrating the NCO into your Design

1. Use **File->OpenProject->**`DE2_lab2.qpf` to open the project file in **Quartus II**. Load the top level file `DE2_lab2.vhd` (i.e., **File-> Open->**`DEC_lab1.vhd`). You should see after a moment a small design. Delete the three lines that connect the input `key` to the outputs `hex0, hex1` and `hex2` from the architecture.

2. Open the file `NCO.cmp` in **Quartus II**. Copy the component description into the section between **ARCHITECTURE** and the first **BEGIN** where the other components are also listed. Instantiate the NCO within the architecture and make the following **PORT MAP** associations.

 © 2012 Dr. Uwe Meyer-Baese




```
C0: NCO PORT MAP (
   phi_inc_i  =>  CONV_STD_LOGIC_VECTOR(268435456,32), --i.e. 2^32/16
   fsin_o  =>  lut, clk  =>  clock_50, reset_n  =>  key(0),
   clken  =>  '1', out_valid  =>  ledg(4));
```

4. Now connect the key to the green leds, and combine the sw and NCO output `lut` using an **AND** gate to implement different quantization in the output waveform, i.e.,

```
ledr <= sw(15 DOWNTO 0) AND lut;
ledg(3 DOWNTO 0) <= key;
```

5. Run a full compilation, by either pressing the  icon, selecting **Processing->Start Compilation,** or hitting **Ctrl+L**. From the report file and the **Classic Timing Analyzer Tool** (**Processing** menu) determine

   Total logic elements = _____

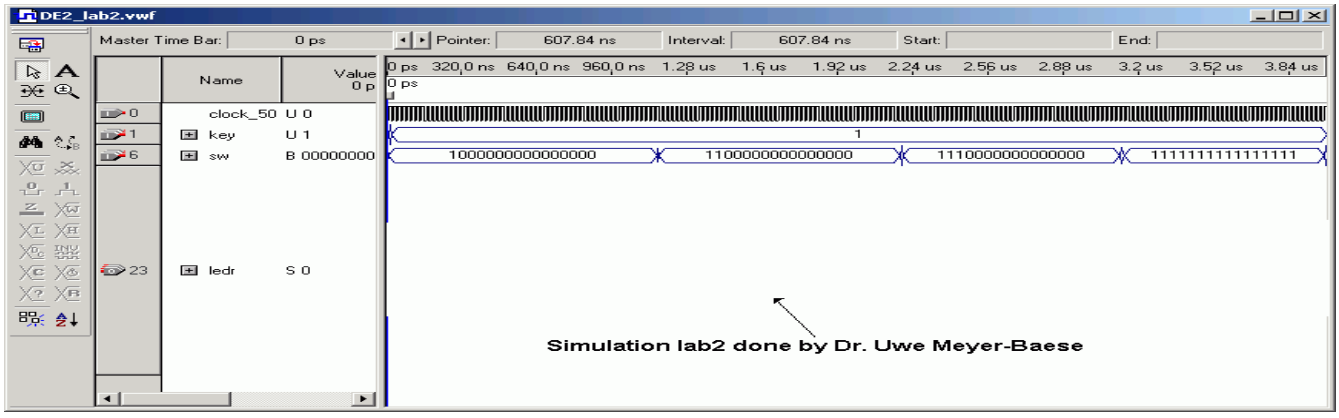
   Embedded Multiplier 9-bit elements = _____

   Total memory bits = _____

   Registered Performance = _____ MHz

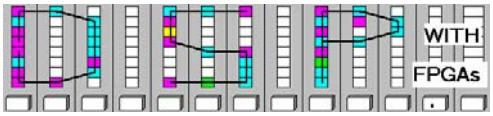
   Do these data match the estimation from the core generation in C.5? (Yes/No)
   If not which data did not match? _____

6. Now start the simulator (details see part B), delete any nodes shown from part B, and import using the **Node Finder ...** the nodes of `clock_50, key, sw` and `ledr` (using the bus nodes not the single bits). Use **Ungroup** and **Group** to remove the `sw(16)` and `sw(17)` from sw. Set **Edit->End Time** to 4000 ns. Set `clock_50` period to 20 ns. `key` to 0 for the first clock cycle and 1 after that. sw is used to change the bits of the sine wave, in other words, implementing different sine quantization error. Turning on `sw(15)=1` only gives 1 bit output precision; `sw(15)=1+sw(14)=1` gives two bit precision etc. The values of sw should change every 1000 ns according to the following pattern.



Start the functional simulation after generating the netlist and then change the `ledr` waveform to **Analog** with **Style Step** and **height 10**. Print the completed waveform including your e-signature.

 © 2012 Dr. Uwe Meyer-Baese



E. Running the NCO Design on the DE2

1. Use **File->OpenProject->**`DE2_lab2.qpf` to open the project file in **Quartus II**. Load the top level file `DE2_lab2.vhd` (i.e., **File-> Open->**`DEC_lab1.vhd`). Instantiate the three components that are required to use the DAC, i.e.,

```
C1: audio_pll PORT MAP(areset=> NOT key(0), inclk0 => clock_50,
                       c0 => clock_18, locked => locked);

C2: i2c_config PORT MAP(iclk => clock_50, irst_n => key(0), --  Host Side
       i2c_sclk => i2c_sclk, i2c_sdat => i2c_sdat);--  I2C Side

C3: audio_dac PORT MAP( --  Audio control incoming
    idata16 => dac, iclk_18_4 => clock_18, irst_n => key(0),
    --  Audio side outgoing
    oaud_bck => aud_bclk, oaud_data => aud_dacdat, oaud_lrck => daclrck);
```

   make the necessary wire connection as follows

```
aud_xck     <= clock_18;    -- 18 MHz clock for audio CODEC
aud_daclrck <= daclrck;     -- Connect left/right signal to CODEC
aud_adclrck <= aud_daclrck; -- Use same left/right signal for DAC and ADC
```

2. To set a 1 kHz output signal we solve $M=F_{out}*2^{32}/F_{in}=85899$ and we need to change in the NCO component instantiation the phase increment to:
   `phi_inc_i  =>  CONV_STD_LOGIC_VECTOR(85899,32), -- Dec value for 1 kHz`

3. The last missing part is a multiplexer that allows displaying the sine wave with and without quantization. To implement this part write an additional **PROCESS** with output register `dac` controlled by clock `DACLRCK` and a two level multiplexer according to the following I/O behavior.

key(0)	**key(1)**	**dac <=**	**Comment**
0	-	(OTHERS => '0')	Reset dac
1	0	lut	display sine without quantization
1	1	sw(15 DOWNTO 0) **AND** lut	Display with quantization

4. After full compilation (**CTRL+L**) download the `DE2_lab2.sop` to the DE2 board using the programmer. Enter the resource and performance data in your `DE2_lab2.vhd` header file.

5. Check the Oscilloscope to see how many bits are needed for a good quality sine wave (use `key(1)` to compare).

   Bits needed for perfect sine wave display _____

6. Listen via headset. How many bits are required to hear a "perfect" sine signal?

   Bits needed for perfect sine wave sound _____

D. Deliverables:

1. Solve the problems from the pre-lab. (3 points).

2. Complete this report; print the VHDL file that includes the NCO component and the resource data; print the waveform from part D.5 for the 4 different quantization. (7 points)

 © 2012 Dr. Uwe Meyer-Baese