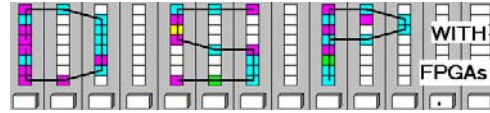


Lastname: \_\_\_\_\_

**LABORATORY  
M-File**



5Digit SS: \_\_\_\_\_

---

**LAB M-FILE: INTRODUCTION TO MATLAB M-FILE SCRIPTS  
(10 points)**

---

In this lab you will be introduced to the MatLab M-file coding. MatLab is a powerful DSP interpreter, that allows you quickly and efficiently to develop LUT and test bench data for your FPGA design. We will in this lab extend our function generator from Lab 1 with additional test functions.

In the **pre-lab** you will compute with “pencil-and-paper” the results you later expect in your design implementation. In the **design part** you will design a function generator for 4 different functions.

---

**Lab Objectives**

After completing this lab you should be able to

- Write simple M-file scripts to define tables and to use predefined functions.
- Use the MatLab help, demo and function library
- Design, simulate, and run a custom function generator using Quartus II on an FPGA board.

---

**Pre-lab (3 points)**

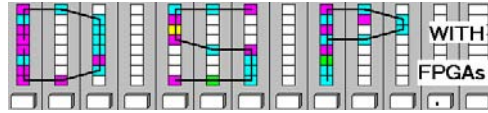
**Note that MatLab can be accessed from any computer in the college.**

1. MatLab works like a very powerful pocket calculator and has a wide selection of predefined functions. You can easily define data, and process, manipulate, or plot these data. Using the MatLab prompt determine the operation of the following MatLab instructions:

MatLab instruction	Short description
<code>a=zeros(3,5);</code>	Creates a 3x5 matrix of zeros
<code>help fir1</code>	
<code>lookfor convolution</code>	
<code>xpsound</code>	
<code>x=0:255;</code>	
<code>y=sin(2*pi*x/256);</code>	
<code>v=100*(rand(1,256)-0.5);</code>	
<code>w(round(256*rand(1,10)))=100;</code>	
<code>f=fft(y);</code>	
<code>plot(abs(f))</code>	
<code>whos</code>	

Lastname: \_\_\_\_\_

**LABORATORY  
M-File**



5Digit SS: \_\_\_\_\_

---

**VHDL Design-lab (7 points)**


Follow the directions below to implement the 4 function generator.

**A. Getting Started**

If you are in B114 or the digital logic lab:

1. On the desktop double click on **Engineering folder**.
2. Double click on **MatLab** icon  to start **MatLab**.
3. Use your **DSPwFPGAs** folder to save your designs. Never save your files to the local drive, use your **network drive** or a **USB drive** instead.

**B. Develop the Function generator MIF files**

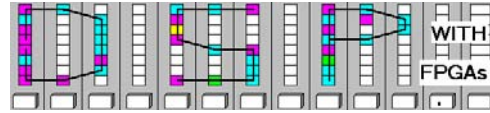
1. Within your **DSPwFPGAs** folder make a new folder called **DE2\_lab4**.
2. Download the file `funcgen.m` from the class webpage and put the file in the **DE2\_lab4** folder.
3. For convenience, in **MatLab**, click on the “Current Directory” selection icon  and select your **DE2\_lab4** folder as the current directory.
4. The files in the **DSPwFPGAs** folder are now visible in the upper left **MatLab** window. Double click on the `funcgen.m` file and you should see after a moment the incomplete M-file:

```
%% m file to compute the 4 functions in the M-File lab
clear; y=zeros(4,256);
x=0:255;                                %% all have length 256
y(1,:)= zeros(1,256);                  %% first is the 1 kHz sine function
y(2,:)= zeros(1,256);                  %% second is the random noise function
y(3,:)= zeros(1,256);                  %% impulse noise
y(4,:)= zeros(1,256);                  %% 16 th harmonic, i.e., 16 kHz sine

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plot the 4 functions %%%%%%%%%%%%%%%
for k=1:4
    subplot(2,2,k)
    stairs(x,y(k,:), 'k-');
    switch k
        case 1, title('lut0:1 KHz sine');
        case 2, title('lut1:Random noise');
        case 3, title('lut2:Impulse noise');
        otherwise title('lut3:16th harmonic, i.e., 16 kHz sine');
    end
    axis([0 255 1.5*min(y(k,:)) 1.5*max(y(k,:))]);
    str=sprintf('lut%d.mif',k-1); fid=fopen(str,'w');
    fprintf(fid,'depth= 256;\r\nwidth = 16;\r\n');
```

Lastname: \_\_\_\_\_

**LABORATORY  
M-File**



5Digit SS: \_\_\_\_\_

```
fprintf(fid,'address_radix = dec;\r\n\data_radix = dec;\r\n');  
fprintf(fid,'content \r\n\begin\r\n');  
xy=[x; y(k,:)];  
fprintf(fid,'%d : %d;\r\n',xy);  
fprintf(fid,'end;\r\n');  
fclose('all');  
end;  
print -djpeg90 fungen.jpg
```

5. Complete your M-file for  $y(1,:)$ ,  $y(2,:)$ ,  $y(3,:)$ , and  $y(4,:)$  so that the required 4 functions, are plotted as shown in Fig. 1. Consult the prelab examples for help!

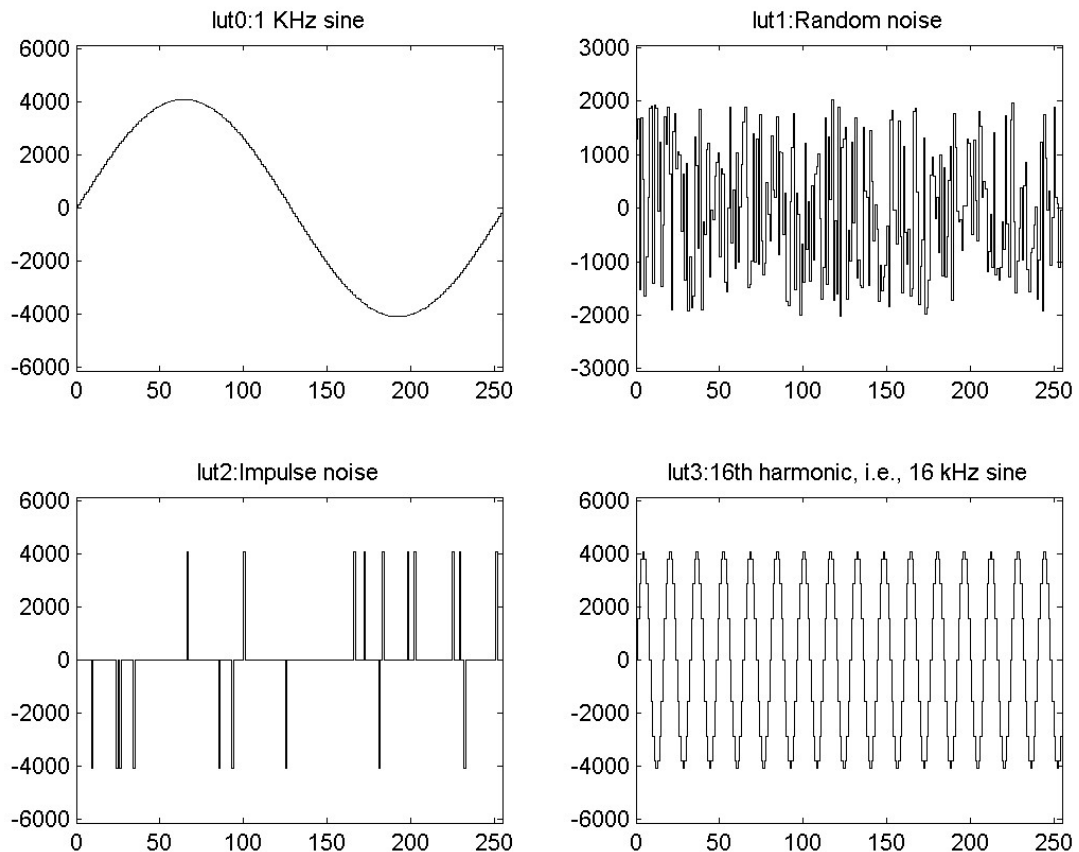
**lut0** is a sine wave with one period and amplitude  $2^{12}$ .

**lut1** is an uniform random noise in the range  $[-2^{11}, 2^{11}]$ .

**lut2** is an impulse noise with 20 impulses with values  $\pm 2^{12}$ .

**lut3** is a sine wave with 16 periods and amplitude  $2^{12}$ .

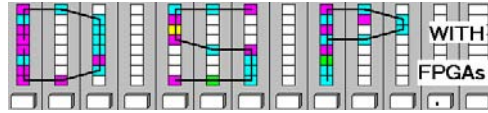
All luts have 256 values and are stored in four different MIF files, lut0.mif, ..., lut3.mif.



**Fig. 1:** Plot of 4 different functions.

Lastname: \_\_\_\_\_

**LABORATORY  
M-File**



5Digit SS: \_\_\_\_\_

**C. Completion of the Function Generator VHDL Design**

1. Copy from your DE2\_lab1 folder the following files: DE2\_lab1.vhd, DE2\_lab1.qsf, DE2\_lab1.qpf, audio\_pll.vhd, audio\_pll.qip, audio\_pll.cmp, i2c\_config.vhd, i2c\_controller.vhd, and audio\_dac.vhd.
2. Rename the three text files DE2\_lab1.vhd, DE2\_lab1.qsf, DE2\_lab1.qpf, to lab4 and replace lab1 with lab4 in all three files. Alternative you can compose a new project using all HDL files, the DE2 device EP2C35F672C6, and load the pin assignment file DE2\_pin\_small.csv using **Assignments-> Import Assignments ...**
3. Then start Quartus II and load the project DE2\_lab4.
4. Make the following changes in the top level file DE2\_lab4.vhd:

The accumulator increment for the *simulation* should implement a length 64 cycle, i.e.,  
`accu <= accu + CONV_STD_LOGIC_VECTOR(67108864,32);`

Keep the 32 bit accumulator size but connect the 8 bits to the MSBs, i.e.,  
`accu_msbs <= accu(31 DOWNTO 24);`

Duplicate and adjust the `lpm_rom` component instantiation as follows:

```
rom0: lpm_rom
GENERIC MAP ( LPM_WIDTH => 16, LPM_WIDTHAD => 8, LPM_FILE => "lut0.mif")
PORT MAP(address=>accu_msbs, inclock=>clock_50, outclock=>clock_50, q=>lut0);
```

Then enumerate the labels with `rom0...rom3` and change the `LPM_FILE` name to "lut0.mif"..."lut3.mif".

5. To implement the next part it is recommended to use a **PROCESS** and four **IF** statements. Build a 16 bit **VARIABLE** called `sum` that is controlled by the four switches `SW0...SW3`. Use the following behavior

<code>sw(0)='1' -&gt; Add lut0, i.e.1 kHz sine to variable sum</code>	<code>sw(0)='0' -&gt; keep sum unchanged</code>
<code>sw(1)='1' -&gt; Add lut1, i.e., random noise to sum</code>	<code>sw(1)='0' -&gt; keep sum unchanged</code>
<code>sw(2)='1' -&gt; Add lut2, i.e., impulse noise to sum</code>	<code>sw(2)='0' -&gt; keep sum unchanged</code>
<code>sw(3)='1' -&gt; Add lut3, i.e.,16 kHz sine to variable sum</code>	<code>sw(3)='0' -&gt; keep sum unchanged</code>

6. The last missing part is a multiplexer that allows displaying the sine wave with and without noise added. To implement this part use the **PROCESS** just created for `sum`, but add a `rising_edge` control to create an output register `dac` controlled by clock `DACLRCK` and a two level multiplexer according to the following I/O behavior. (Note: a similar mux was designed in lab 2 part E.3)

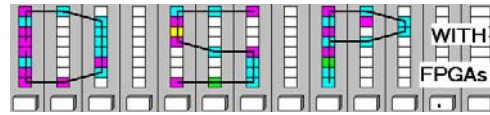
<b>key(0)</b>	<b>key(1)</b>	<b>dac &lt;=</b>	<b>Comment</b>
0	-	<code>(OTHERS =&gt; '0')</code>	Reset dac
1	0	<code>lut0</code>	Display 1 kHz sine
1	1	<code>sum</code>	Display output variable sum

7. For the simulation connect `accu_msbs` to `ledr[7..0]` and `sum` to `ledr[15..8]`, i.e.,

```
ledr(7 DOWNTO 0) <= accu_msbs;
```

Lastname: \_\_\_\_\_

## LABORATORY M-File



5Digit SS: \_\_\_\_\_

```
ledr(15 DOWNTO 8) <= sum(15 DOWNTO 8);
```

Note since `sum` is a **VARIABLE** the second assignment needs to be placed within the **PROCESS**.

8. The overall **RTL viewer** display with annotations of the circuit parts is shown below.

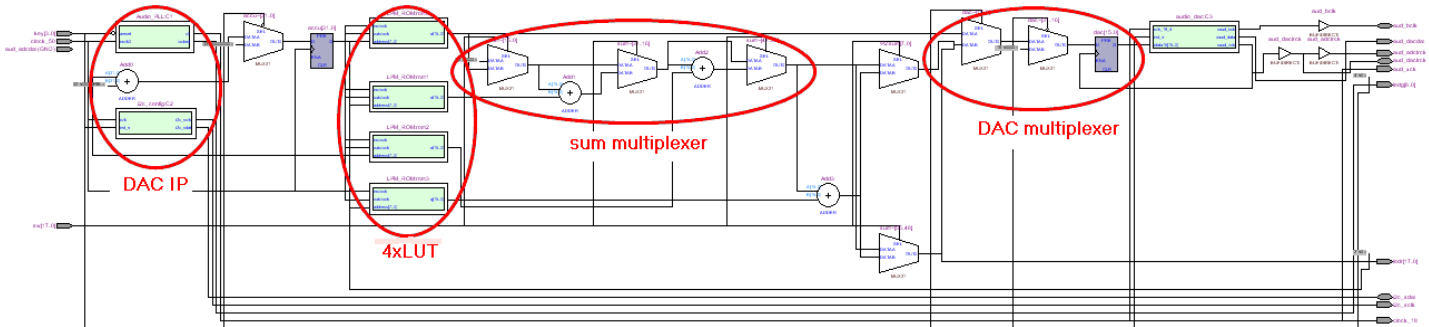


Fig. 2: RTL viewer of complete lab 4 design for verification.

### D. Simulate the Design using the Waveform Display

1. To run a simulation follow the usual procedure you used in the FPLD Quartus labs to do a simulation. As a reminder you may also consult lab 1 part E.
2. From the ENTITY ports import via the **Node Finder...** only `clock_50`, `key`, `sw`, and `ledr`. Use **Ungroup** to remove 3 bits `ledr(17...14)` from `ledr`. Now use **Group** of the bits `ledr(13...8)` to form the signal `sum`; also use **Group** of the bits `ledr(7...0)` to form `accu`. 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 select the `lut` to be displayed one at a time. The simulation should show first the base sine signal, then random noise, impulse noise and finally the 16<sup>th</sup> harmonic. The waveform for `sum` and `accu` should be **Analog** with **Style Step** and **height 10**. Match the simulation shown below. Finally add your electronic signature before printing the simulation in "landscape" format, please.

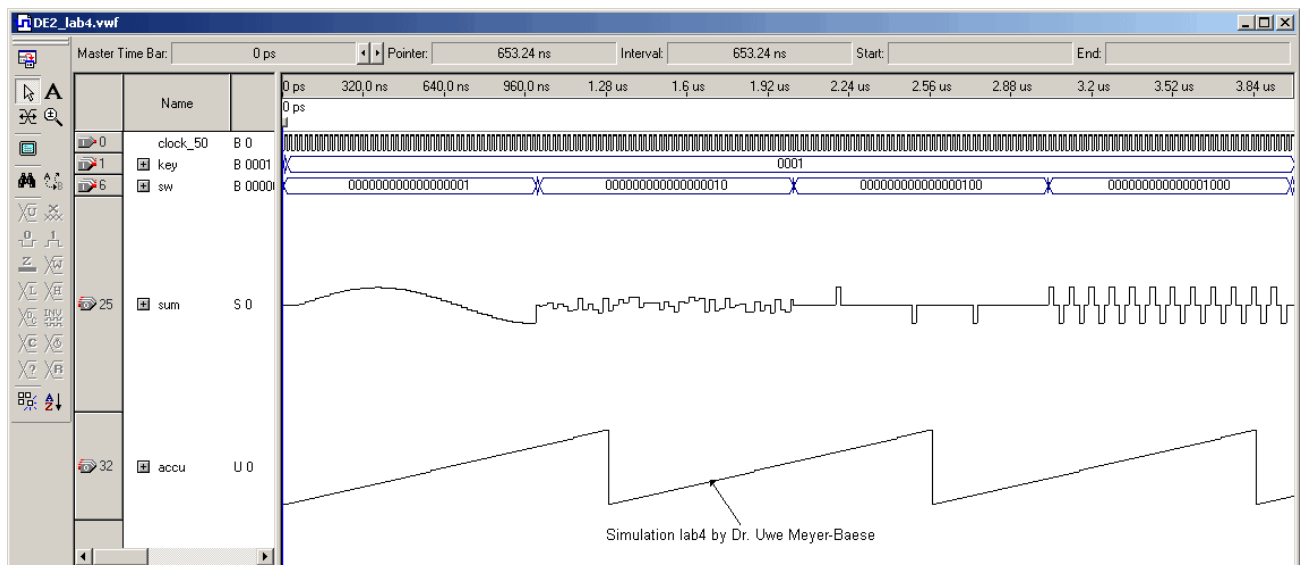
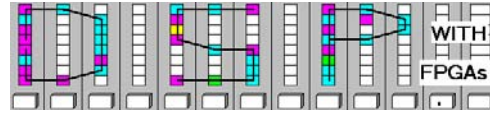


Fig. 3: Simulation of the 4 waveform types.

Lastname: \_\_\_\_\_

**LABORATORY  
M-File**



5Digit SS: \_\_\_\_\_

**E. Full Compile, Download and Observing the FPGA board**

- 1) To run the design on the DE2 we need to adjust to a 1 kHz output signal. The accumulator increment for the *implementation* should be changed as follows  
`accu <= accu + CONV_STD_LOGIC_VECTOR(85899,32);`
- 2) Now run a full compilation. 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

Enter these resource and performance data also in your DE2\_lab4 VHDL header file.

- 3) Once you have completed and compiling your model, you can now download it to the board. Follow the board programming step as in lab 1 or 2.
- 4) Using your headset *or* the oscilloscope to report briefly your observations. Alternative, you may also use the PCs remote access of the oscilloscope to print the scope plot.

SW value	Observation
SW0=1 all other SW=0	
SW1=1 all other SW=0	
SW2=1 all other SW=0	
SW3=1 all other SW=0	

Remember SW is '1' when "north", i.e., turned towards the LCD.

**F. Deliverables:**

1. Solve the problems of the pre-lab. (3 points).
2. Complete this report, print the `funken.jpg` file, the VHDL file `DE2_lab4.vhd` (with Resource Data listed), and the simulation file as in Fig. 3 (7 points).

**Make sure your name and SS is on all pages you turn in!**