Lastname: ___________________

5Digit SS:___________________

LABORATORY
Intro. to M-File

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 and we will also write M-file scripts to test a complex multiplier design that needs 3 real multiplications and 5 add operations.

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 and a complex multiplier.

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 and simulate complex multiplier systems using Simulink

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
`a=zeros(3,5);`	Creates a 3x5 matrix of zeros
`help fir1`	
`lookfor convolution`	
`xpsound`	
`x=0:127;`	
`y=sin(2*pi*x/128);`	
`v=100*rand(1,128);`	
`w=floor(x/16);`	
`f=fft(y);`	
`plot(abs(f))`	
`whos`	

Lastname: ____________________ **LABORATORY**
Intro. to M-File

5Digit SS:___________________

2. A complex multiplication of the type $(x+\mathrm{j}y)(c+\mathrm{j}s)=xc-sy+\mathrm{j}(xs+yc)=\mathrm{Re}+\mathrm{jIm}$ is a frequent used object in the DFT computation discussed in later labs. The complex multiplication in the direct form requires 4 real multiplications and 2 add operations. From an implementation standpoint, the 4 multipliers realized as array multipliers are the most resource intensive objects. Algorithms that use 3 multipliers and 5 adders are usually more resource effective. Such an algorithm works as follows:

Compute first: $a_1=c+s$, $a_2=c-s$, **and** $a_3=x-y$
then compute $m_1=c*a_3$, $m_2=y*a_2$, **and** $m_3=x*a_1$,
and finally: $re=m_2+m_1$ **and** $im=m_3-m_1$

As can be seen above, the multiplier uses 3+2=5 add and 3 multiply operations. Develop a short MatLab script that computes the output for 8 bit data and coefficient quantization by completing the following M-File:

```
%% m file to check the complex multiplier with 3*5+
x=60; y=40;
w=exp(j*pi/5);
c=round(real(w)*128);
s=round(imag(w)*128);
%%% add the equation for a1,a2,a3,m1,m2,m3, re and im here:
```

```
%%% display the results
check=(x+j*y)*w;
str= sprintf('result = %5d+j%5d => check: %5.3f+j%5.3f',...
    floor(re/128),floor(im/128),real(check),imag(check));
disp(str);
```

Save the file under the name `cmtest.m` on your **DSPwFPGAs** directory. To run the file, type its name (without the .m extension) in the MatLab prompt. Make sure to use dir *.m and cd instructions so that the MatLab prompt points to the directory where the M-file was placed. Now, use the script to complete the following test data table:

x+jy	**w=exp(jφ)=(c+js)**	**(x+jy)*w**	**c**	**s**	**c+s**	**c-s**
60+j40	φ=π/5	25+j67	104	75	179	29
70+j50	φ=π/9					
60+j40	φ=-π/9					
70-j50	φ=-π/5					

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

Lastname: ___________________ LABORATORY
Intro. to M-File

5Digit SS:__________________

Simulink Design-lab (7 points)

Follow the directions below to implement the 4 function generator and the complex multiplier.

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. From the top icon list in the **MatLab** window click on the **Simulink** icon to start **Simulink.**
4. 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. Design the Function generator systems

1. Download the file `funcgen.m` from the class webpage and put the file in the **DSPwFPGAs** folder.
2. For convenience, in **MatLab,** click on the "Current Directory" selection icon and select your **DSPwFPGAs** folder as the current directory.
3.
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:

```
clear;
%% m file to compute the 4 functions in the ML lab
x=0:2^7-1;                               %% all have length 128

y0= zeros(1,128);            %% first is the sine function
y1= zeros(1,128);            %% second is the multi step function
y2= zeros(1,128);            %% uniform random numbers
y3= zeros(1,128);            %% symmetric triangular

%%%% Plot the results and store in LUT
plot(x,y0,'k-',x,y1,'k--',x,y2,'k.',x,y3,'k-.');
legend('sine','multi step','random','triangular')
xlabel('Sample');ylabel('Amplitude');title('4 Functions generator')
axis([0 127 0 1.1*2^14]);
print -djpeg funcgen.jpg

LUT=round([y0, y1, y2, y3]);
```

5. Complete your M-file so that the required 4 functions, y0,.., y4, are plotted as shown in Fig. 1. Note that the output range is [0,2^14-1] unsigned.
Hint: For the triangular function use 3 sections: 32 values going up, 64 down, 32 up. For each section use counters like [0:31, ...]. Then scale the amplitude to full range, i.e., 14 bits.

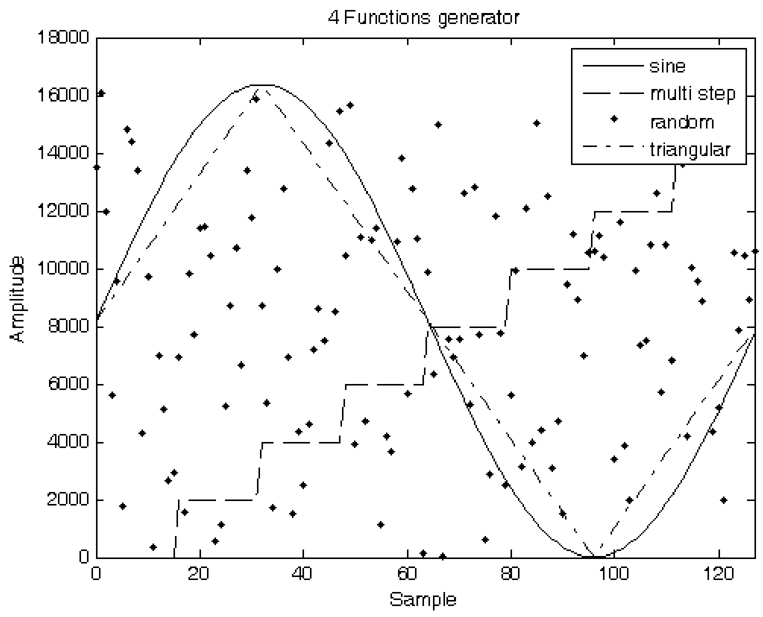


Fig. 1: Plot of 4 different functions.

6. From the **Simulink Library Browser** window, open the design you created for lab 1 and save it under a different name, `lab4.mdl`.

7. Rearrange your design to match the changes show in Fig. 2. Add in your design file the **Binary to Seven Segment**, **Bus Conversion**, **Seven Segment Display 1, Repeating Sequence Stair** and **GND** blocks for the LEDs to your design as shown in Fig. 2.

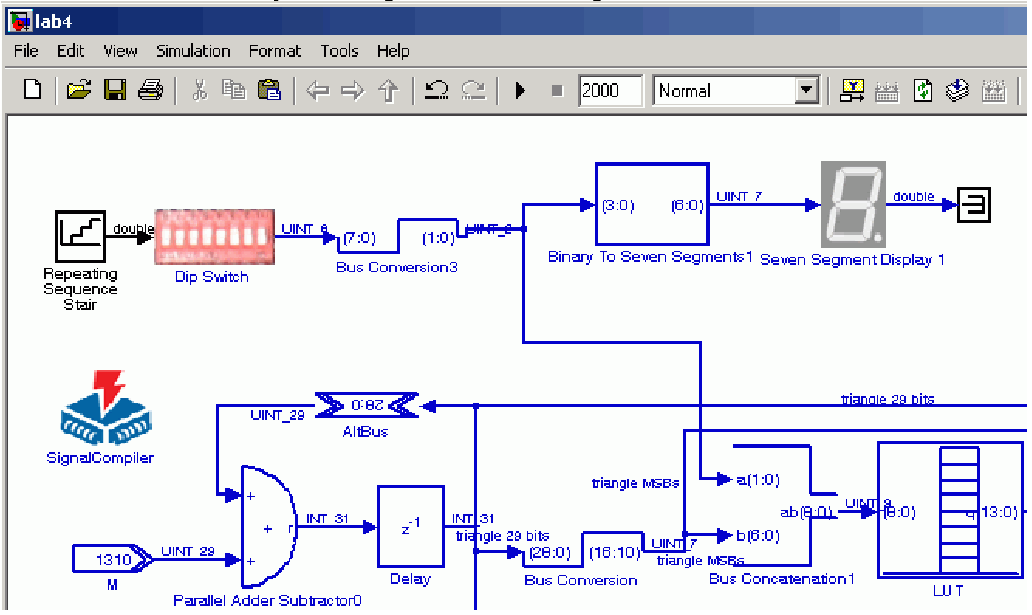
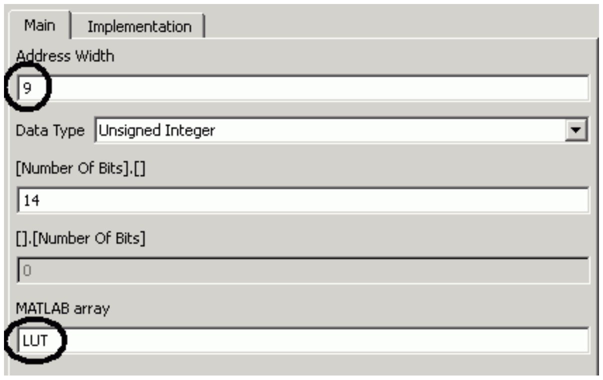


Fig. 2: Changes made to `lab1.mdl`.

C. Completion of the Function Generator design

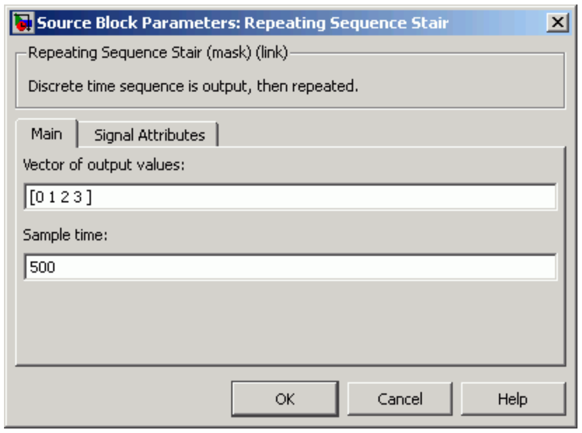
1. On the menu bar, go to **Format**, **Port/Signal Display**. Make sure **Port Data Types** is checked. This way, Simulink will show the data types leaving each block.

2. Double click the **LUT** block and enter the new **Address Width** 9 and under **MATLAB array** your new variable LUT. Note: you need to run your `funcgen.m` script to use the LUT data each time you start MatLab new.



Address Width: 9
Data Type: Unsigned Integer
[Number Of Bits].[]: 14
[].[Number Of Bits]: 0
MATLAB array: LUT

3. Complete your design as shown in Fig. 2. The **Repeating Sequence Stair** block can be found in the **Simulink->Sources** subdirectory. Double click the **Repeating Sequence Stair** block and enter the data shown below:



Source Block Parameters: Repeating Sequence Stair

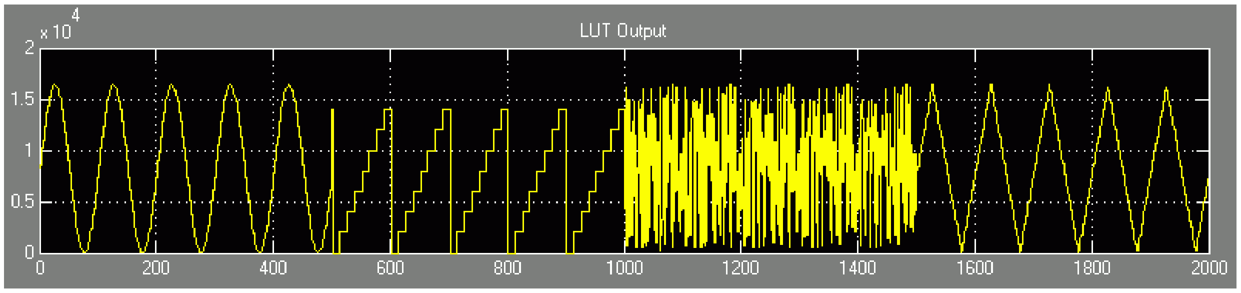
Repeating Sequence Stair (mask) (link)

Discrete time sequence is output, then repeated.

Vector of output values: [0 1 2 3]

Sample time: 500

4. Verify the design via Simulink simulation. Change the Simulation stop time to 2000 and the accumulator increment value to M=$F_{out}/F_{in}*2^{17}$=1e6/100e6*2^{17}=1310 so that all four set of functions are viewed and the output frequency is 1 MHz. (Print out the simulation waveform for LUT scope.) The simulation should look as follows:



5. Next double-click the **SignalCompiler** block. Note That you can only compile the design on a PC that has a board, i.e. has a DSP builder license enabled.

6. The **SignalCompiler** block should pop up. Select **Use Board Block to Specify Device**. Then click on the **Compile** button. (Note: This may take a couple of minutes.)

7. Click the **Program** button. Watch the output of the D2A output with an oscilloscope or the 7-segment LEDs in case you do not have an oscilloscope connected to the board. Change the bits 1 and 2 of the **DIP switch** so that you can see all 4 functions on the oscilloscope.

8. Show the lab TA your working FPGA board design.

9. Determine the number of logic elements, M4Ks, and the minimum slack of the device:

   Logic Cells = _________________

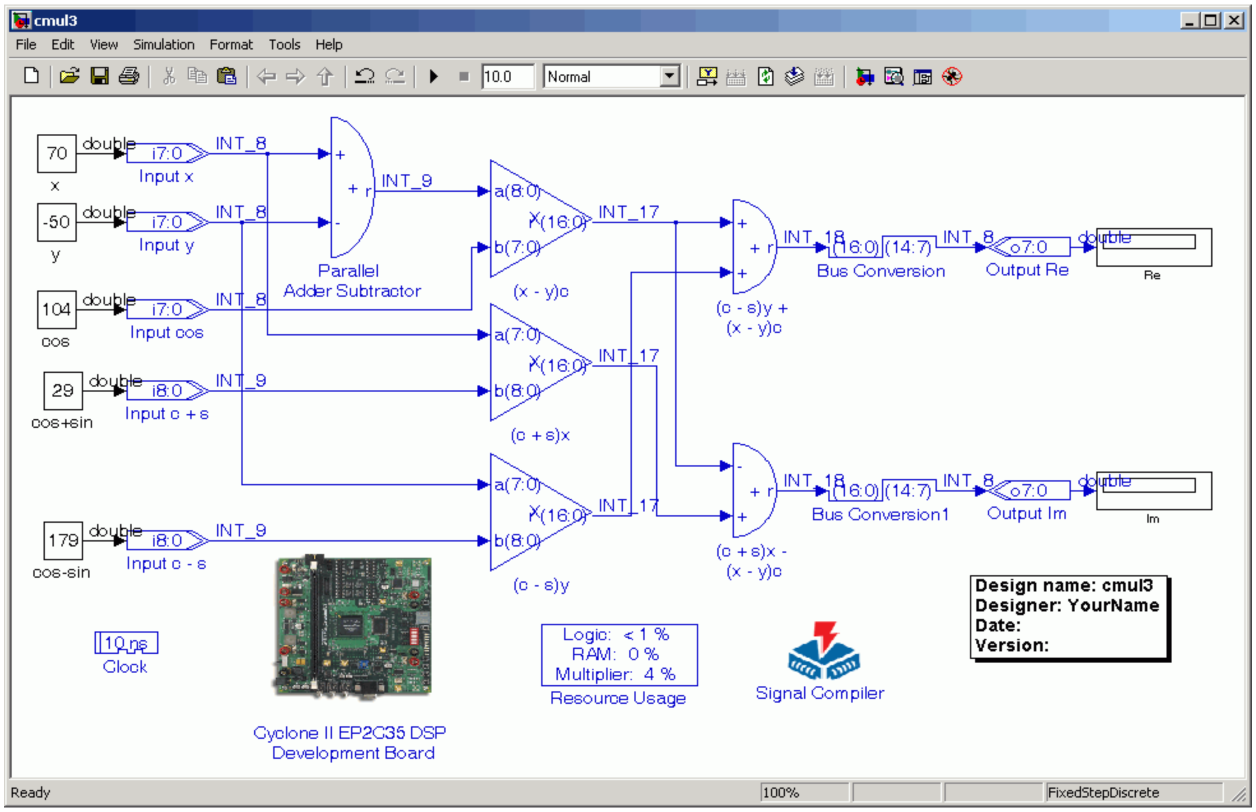
   M4Ks = _________________

   Slack = _________________

   from the report files or the **Resource Usage** block. Finally, complete the author field for Lab 4.

D. Completion of the Complex multiplier Design

1. Create a new model and complete the design of the complex multiplier to match Fig. 3.



Design name: cmul3
Designer: YourName
Date:
Version:

Fig. 3: Complex multiplier using three real multiplications.

2. Verify the design via Simulink simulation for the 4 different test data computed in the prelab and complete the following table using your Simulink simulation results:

x+jy	w=exp(jφ)=(c+js)	Re	Im
60+j40	φ=π/5		
70+j50	φ=π/9		
60+j40	φ=-π/9		
70-j50	φ=-π/5		

3. Compile the design using **Signal Compiler** and determine

   Logic Cells = ______________

   DSP 9x9 = ______________

   from the report files or the **Resource Usage** block.

E. Deliverables:

1. Solve the problems of the pre-lab. (3 points).
2. Print the 2 MDF files and the Simulink simulation (7 points).

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