Lastname: ___________________

5Digit SS:___________________

LABORATORY
FFT

LAB FFT: INTRODUCTION TO FAST FOURIER TRANSFORM
(10 points)

In this lab you will be introduced to the design for a Fast Fourier Transform (FFT). The FFT is one of the most important DSP objects and is used not only to compute an approximation of the Fourier Transform, but also to enable fast convolution, a very time consuming filtering operation when done in the time domain.

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 an 8 point radix-2 FFT using the principle of decimation in frequency.

Lab Objectives

After completing this lab you should be able to

- Develop a radix-2 FFT and compute test data
- Understand the difference between DFT and FFT
- Design and simulate FFT using the principle of decimation in frequency using Quartus

Pre-lab (3 points)

The following figure shows the 8 point radix-2 FFT using the principle of decimation in frequency:

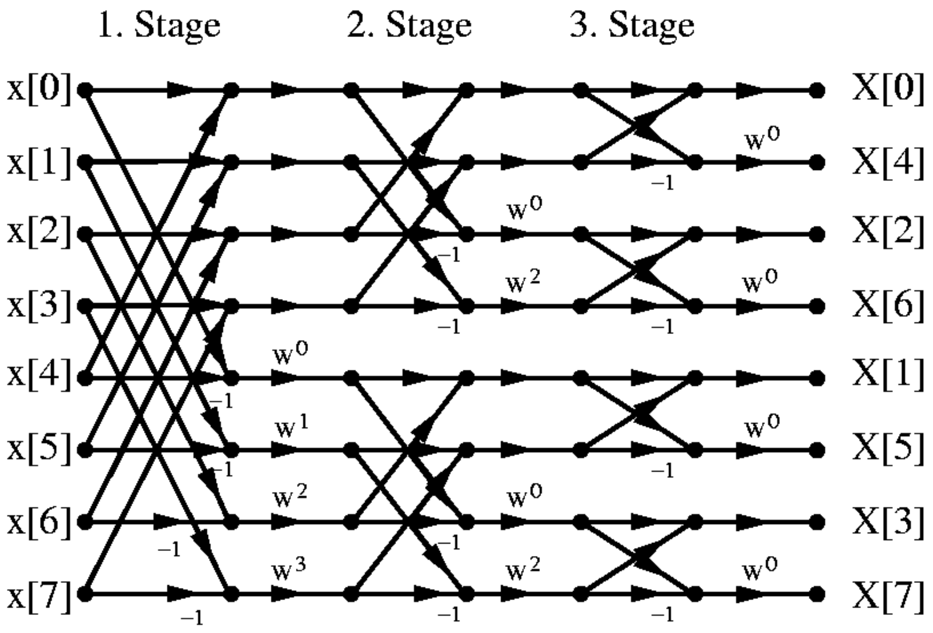


Fig. 1

Lastname: ____________________

5Digit SS:___________________

LABORATORY
FFT

1. From the signal flow graph in Fig.1, determine for each stage the number of blocks required from each type of butterfly.

	1. stage	**2. stage**	**3. stage**
Butterfly with w^0			
Butterfly with w^1			
Butterfly with w^2			
Butterfly with w^3			

2. Redraw the radix-2 signal flow graph from **Fig.1** such that the butterfly operations do not overlap and input/output are in natural rather than bit-reverse order, by completing the following figure:

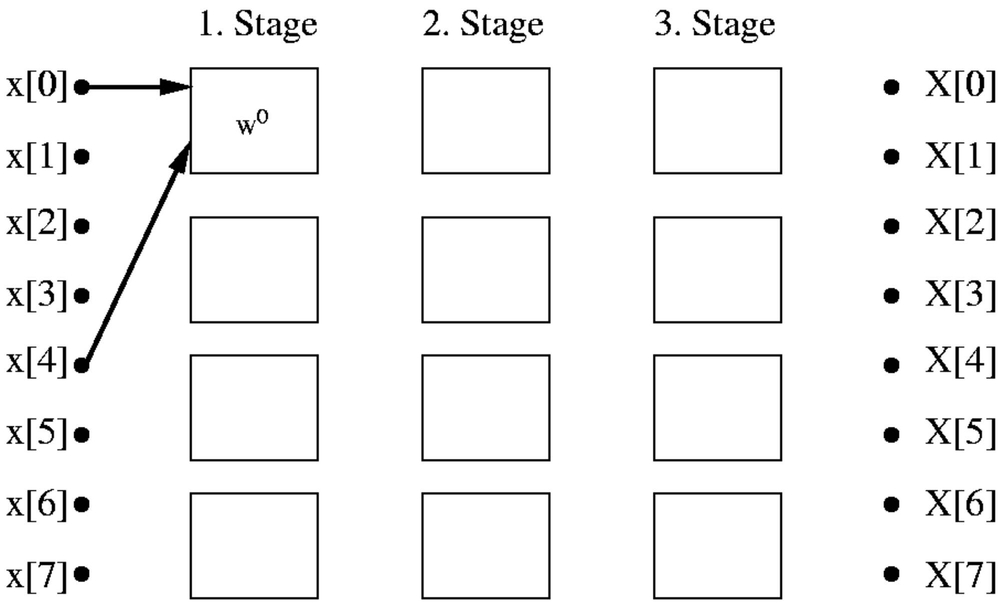


Fig. 2

3. Each butterfly $k \in [0,3]$ performs a complex value operation of the type

$$D = A+B; \quad \text{and} \quad E = (A-B)W^k$$

Complete the following table for the given input values A=100+j*110; B=- 40+j*10

	D_real	**D_imag**	**E_real**	**E_imag**
Butterfly with w^0				
Butterfly with w^1				
Butterfly with w^2				
Butterfly with w^3				

Hint: Remember **W** =exp(-j*2π/8)

4. Using MatLab, compute for x=100:100:800 the FFT, X = round(fft(x)), and complete the table:

	X[0]	**X[1]**	**X[2]**	**X[3]**	**X[4]**	**X[5]**	**X[6]**	**X[7]**
Real								
Imag								
abs(X[k])								

VHDL Design-lab (7 points)

Follow the directions below to implement the 8-point radix-2 FFT circuit.

A. Getting Started

1. On the desktop double click on **Engineering folder** and start version 9.1 of **Quartus II**. Do not use Quartus version 10 since it does not has a build-in simulator.

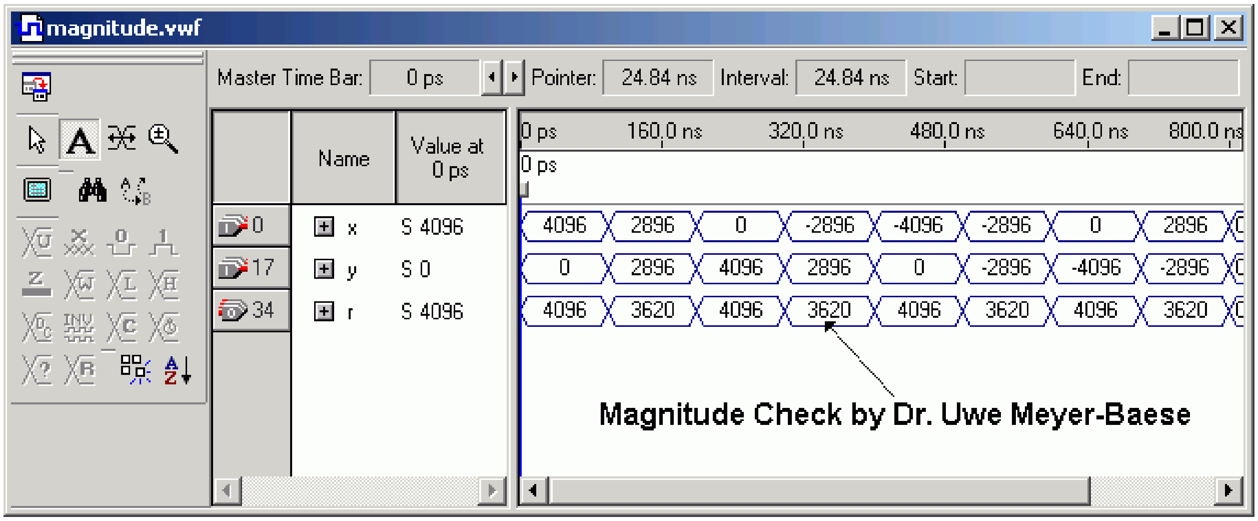
2. You should not save anything on the local hard disk. You will have to use a USB flash drive, or your “mapped” home directory to save the files. Create (if not done in a previous lab) a new folder named **DSPwFPGAs** on your mapped network drive.

B. Design the magnitude circuit

1. For an absolute value of a complex number x+jy the equation $r=\sqrt{(x^2+y^2)}$ needs to be computed. An approximation to this task should be designed next. Generate the new Quartus project `magnitude.`

2. Design a `magnitude.vhd` circuit using the following approximation: $r=\max(x,y)+\min(x,y)/4$. This approximation is bias free and has an error of less than 10%. Design the circuit with inputs x and y, and output r . All data should be in the signed 16-bit integer format; do not use any flip-flops.

3. Match the following simulation for a testbench:



magnitude.vwf

Master Time Bar: 0 ps Pointer: 24.84 ns Interval: 24.84 ns Start: End:

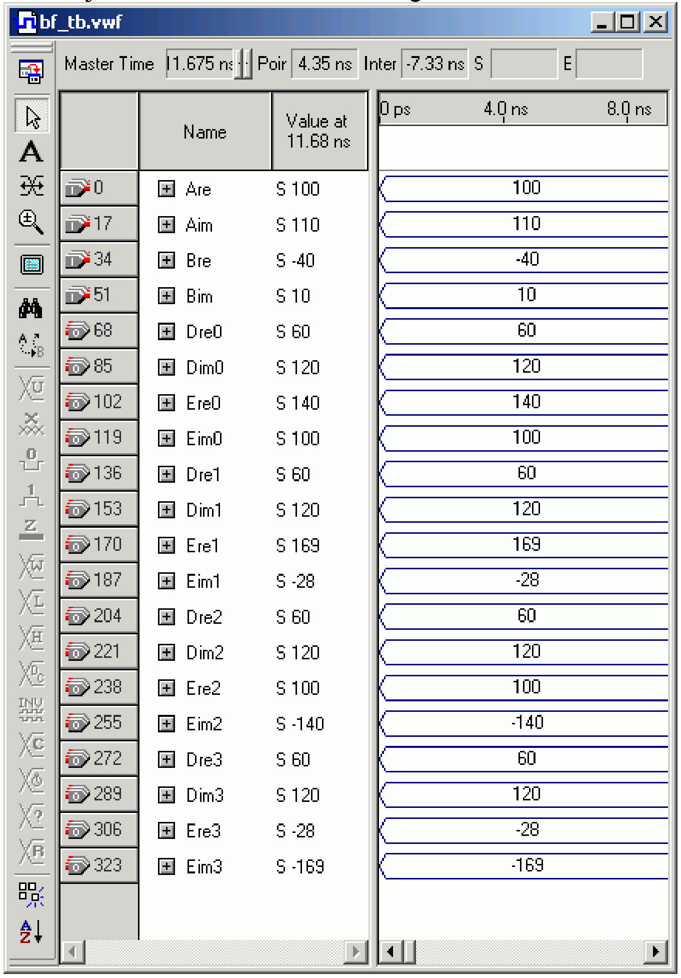
	Name	Value at 0 ps	0 ps							
0	x	S 4096	4096	2896	0	-2896	-4096	-2896	0	2896
17	y	S 0	0	2896	4096	2896	0	-2896	-4096	-2896
34	r	S 4096	4096	3620	4096	3620	4096	3620	4096	3620

Magnitude Check by Dr. Uwe Meyer-Baese

C. Design the W1 and W3 Butterflies

4. Download the files `bf0.vhd` and `bf1.vhd` from the class webpage and place it in your **DSPwFPGAs** folder. These are the butterfly designs for w^0=1 and w^1=-jπ/4. Now design the w^2 butterfly `bf2.vhd` using the w^0 butterfly and the w^3 butterfly `bf3.vhd` using w^1.

5. Generate a new project `bf_tb`. Develop a testbench `bf_tb.vhd` using the data from the Prelab i.e., A=100+j*110; B=-40+j*10 and match the following simulation for a testbench:



bf_tb.vwf

Master Time 11.675 ns Poir 4.35 ns Inter -7.33 ns S E

	Name	Value at 11.68 ns	0 ps 4.0 ns 8.0 ns
0	Are	S 100	100
17	Aim	S 110	110
34	Bre	S -40	-40
51	Bim	S 10	10
68	Dre0	S 60	60
85	Dim0	S 120	120
102	Ere0	S 140	140
119	Eim0	S 100	100
136	Dre1	S 60	60
153	Dim1	S 120	120
170	Ere1	S 169	169
187	Eim1	S -28	-28
204	Dre2	S 60	60
221	Dim2	S 120	120
238	Ere2	S 100	100
255	Eim2	S -140	-140
272	Dre3	S 60	60
289	Dim3	S 120	120
306	Ere3	S -28	-28
323	Eim3	S -169	-169

D. Completion of the 8-point radix-2 FFT

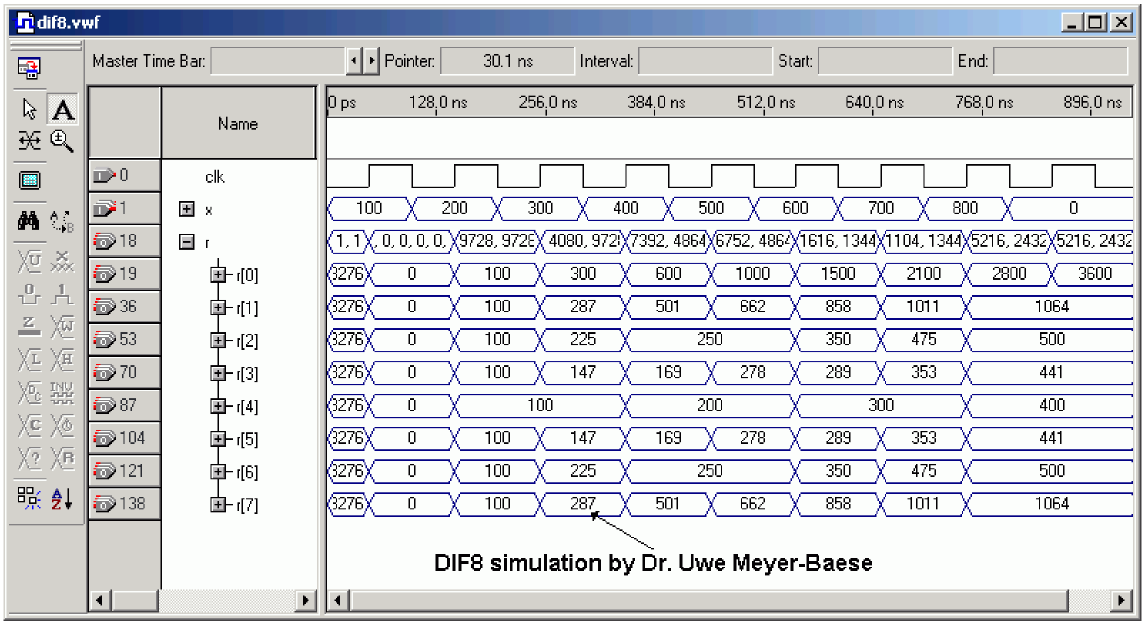
1. Generate a new project `dif8`. Develop a design `dif8.vhd` using the signal flow graph you developed in the pre-lab. Your `dif8.vhd` design should have input `clk` and `x` and output a magnitude vector `r` with 8 elements.

2. The imaginary input to the FFT is zero. The real input x should be put in a shift register such as the shift register we have used in the direct form FIR filter. Your output `r` should also be using registers. Do not use any registers for the butterfly or magnitude computations.

3. Copy and paste the necessary subsystems components and then wire them as determined in the pre-lab. You need 12 butterfly components and 8 magnitude component instantiations.

4. Simulate your design and compare the results shown by the provided scopes with the data you computed for the pre-lab for x=100:100:800.

5. Try to match the following simulation:



DIF8 simulation by Dr. Uwe Meyer-Baese

6. 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 `dif8.vhd` header file.

E. DE2 Implementation of the FFT Design

1. Download the MatLab file `fun4fft.m` and `seg7_bar.vhd` from the class webpage and place it in your **DSPwFPGAs** folder. Run the `fun4fft.m` script in MatLab. It will generate the sine LUTs for 0, 6, 12, and 18 kHz.

2. Start Quartus II and build a new project called `DE2_lab7`. As device select `EP2C35F672C6` and load the pin assignment file `DE2_pin_small.csv` (from lab5 folder or BlackBoard) into Quartus II using **Assignments-> Import Assignments ...**

3. Copy from your DE2_lab5 folder into the DE2_lab7 folder the following files: `DE2_lab5.vhd`, `audio_pll.vhd`, `audio_pll.qip`, `audio_pll.cmp`, `i2c_config.vhd`, `i2c_controller.vhd`, and `audio_dac.vhd`. Rename the file `DE2_lab5.vhd` to `DE2_lab7.vhd` and change the entity name to `DE2_lab7`.

4. Remove the fir component from DE2_lab7.vhd and define in the signal section and instantiate dif8 and seg7_BAR as a component into the DE2_lab7.vhd as follows:

```
C4: dif8 PORT MAP(clk=>DACLRCK, x=>x, r=>r);

R0: seg7_BAR PORT MAP (idig => CONV_STD_LOGIC_VECTOR(r(0)/4096,4), oseg => hex7);
R1: seg7_BAR PORT MAP (idig => CONV_STD_LOGIC_VECTOR(r(1)/4096,4), oseg => hex6);
R2: seg7_BAR PORT MAP (idig => CONV_STD_LOGIC_VECTOR(r(2)/4096,4), oseg => hex5);
R3: seg7_BAR PORT MAP (idig => CONV_STD_LOGIC_VECTOR(r(3)/4096,4), oseg => hex4);
R4: seg7_BAR PORT MAP (idig => CONV_STD_LOGIC_VECTOR(r(4)/4096,4), oseg => hex3);
R5: seg7_BAR PORT MAP (idig => CONV_STD_LOGIC_VECTOR(r(5)/4096,4), oseg => hex2);
R6: seg7_BAR PORT MAP (idig => CONV_STD_LOGIC_VECTOR(r(6)/4096,4), oseg => hex1);
R7: seg7_BAR PORT MAP (idig => CONV_STD_LOGIC_VECTOR(r(7)/4096,4), oseg => hex0);
```

Define the required signals in the correct width. For convenience copy the definitions for S16 data type from the iir_order1.vhd or fir.vhd file and place it in the signal definition section of DE2_lab7.vhd. In the port list add hex2,...,hex7 to the available **OUT** ports.

5. Keep key(0) as global reset for the registers and the functionality of SW0…SW3 as in lab4. The signal dac should always be a register controlled by DACLRCK with key(0) as synchronous reset. Connect dac to the sum to observe the FFT input with the scope/headset.

6. Now run a full compilation. From the **Compilation Report** and the **Classic Timing Analyzer Tool** (**Processing** menu) for clock_50 determine

   Total logic elements = _____________

   Embedded Multiplier 9-bit elements = _____________

   Total memory bits = _____________

   Registered Performance = _____________ MHz

Enter these resource and performance data in your DE2_lab7 VHDL header file.

7. Now download the DE2_lab7.sop file to the board and observe with oscilloscope or head set. The hex0-7 display shows the FFT magnitude values in form of a “bar” graph. The letter _,u,o,..8 are used. Complete the table below with your observations. For the HEX 7-segments specify the displayed letter. For the scope specify the function and frequency.

SW3	SW2	SW1	SW0	HEX7	HEX6	HEX5	HEX4	HEX3	HEX2	HEX1	HEX0	Scope
0	0	0	1									
0	0	1	0									
0	1	0	0									
1	0	0	0									
1	1	1	1									

F. Deliverables:

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

2. Complete this report. Print the VHDL files magnitude.vhd, bf2.vhd, bf3.vhd, dif8.vhd and DE2_lab7.vhd and the 3 simulations from magnitude, bf_tb and dif8 (7 points).

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