ECL333	DIGITAL SIGNAL PROCESSING LABORATORY	CATEGORY	L	T	P	CREDIT
		PCC	0	0	3	2

Preamble:

- The following experiments are designed to make the student do real time DSP computing.
- Dedicated DSP hardware (such as TI or Analog Devices development/evaluation boards) will be used for realization.

Prerequisites:

- ECT 303 Digital Signal Processing
- EST 102 Programming in C

Course Outcomes: The student will be able to

CO 1	Simulate digital signals.
CO 2	verify the properties of DFT computationally
CO 3	Familiarize the DSP hardware and interface with computer.
CO 4	Implement LTI systems with linear convolution.
CO 5	Implement FFT and IFFT and use it on real time signals.
CO 6	Implement FIR low pass filter.
CO 7	Implement real time LTI systems with block convolution and FFT.

Mapping of Course Outcomes with Program Outcomes

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12
CO1	3	3	1	2	3	0	0	0	3	0	0	1
CO2	3	3	1	2	3	0	0	0	3	0	0	1
CO3	3	3	3	2	3	0	0	0	3	1	0	1
CO4	3	3	1	2	3	0	0	0	3	0	0	1
CO5	3	3	1	1	3	0	0	0	0	0	0	1
CO6	3	3	1	1	3	0	0	0	0	0	0	1
CO7	3	3	1	3	3	0	0	0	3	3	0	0

Assessment Pattern

Mark Distribution:

Total Mark	CIE	ESE
150	50	100

Continuous Internal Evaluation Pattern:

Each experiment will be evaluated out of 50 credits continuously as

Attribute	Mark
Attendance	15
Continuous assessment	30
Internal Test (Immediately before the second series test)	30

End Semester Examination Pattern: The following guidelines should be followed regarding award of marks

Attribute	Mark
Preliminary work	15
Implementing the work/ Conducting the experiment	10
Performance, result and inference (usage of equipments and trouble shooting)	25
Viva voce	20
Record	5

Course Level Assessment Questions

CO1-Simulation of Signals

1. Write a Python/MATLAB/SCILAB function to generate a rectangular pulse.
2. Write a Python/MATLAB/SCILAB function to generate a triangular pulse.

CO2-Verfication of the Properties of DFT

1. Write a Python/MATLAB/SCILAB function to compute the N-point DFT

matrix and plot its real and imaginary parts.

2. Write a Python/MATLAB/SCILAB function to verify Parseval's theorem for $N = 1024$.

CO3-Familarization of DSP Hardware

1. Write a C function to control the output LEDs with input switches.
2. Write a C function to connect the analog input port to the output port and test with a microphone.

CO4-LTI System with Linear Convolution

1. Write a function to compute the linear convolution and download to the hardware target and test with some signals.

CO5-FFT Computation

1. Write and download a function to compute N point FFT to the DSP hardware target and test it on real time signal.
2. Write a C function to compute IFFT with FFT function and test in on DSP hardware.

CO6-Implementation of FIR Filter

1. Design and implement an FIR low pass filter for a cut off frequency of 0.1π and test it with an AF signal generator.

CO7-LTI Systems by Block Convolution

1. Implement an overlap add block convolution for speech signals on DSP target.

List of Experiments

(All experiments are mandatory.)

Experiment 1. Simulation of Signals Simulate the following signals using Python/ Scilab/MATLAB.

1. Unit impulse signal
2. Unit pulse signal
3. Unit ramp signal
4. Bipolar pulse
5. Triangular signal

Experiment 2. Verification of the Properties of DFT

- Generate and appreciate a DFT matrix.
  1. Write a function that returns the N point DFT matrix $\mathbf{V_N}$ for a given N.
  2. Plot its real and imaginary parts of $\mathbf{V_N}$ as images using *matshow* or *imshow* commands (in Python) for $N = 16$, $N = 64$ and $N = 1024$
  3. Compute the DFTs of 16 point, 64 point and 1024 point random sequences using the above matrices.
  4. Observe the time of computations for $N = 2^{\gamma}$ for $2 \gamma 18$ (You may use the *time* module in Python).
  5. Use some iterations to plot the times of computation against γ. Plot and understand this curve. Plot the times of computation for the *fft* function over this curve and appreciate the computational saving with FFT.
- Circular Convolution.
  1. Write a python function *circcon.py* that returns the circular convoluton of an N_1 point sequence and an N_2 point sequence given at the input. The easiest way is to convert a linear convolution into circular convolution with $N = max(N_1, N_2)$.
- Parseval's Theorem
  For the complex random sequences $x_1[n]$ and $x_2[n]$,

$$\sum_{n=0}^{N-1} x_1[n]x_2^*[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_1[k]X_2^*[k]$$

1. Generate two random complex sequences of say 5000 values.
2. Prove the theorem for these signals.

Experiment 3. Familarization of DSP Hardware

1. Familiarization of the code composer studio (in the case of TI hard- ware) or Visual DSP (in the case of Analog Devices hardware) or any equivalent cross compiler for DSP programming.
2. Familiarization of the analog and digital input and output ports of the DSP board.
3. Generation and cross compilation and execution of the C code to con- nect the input digital switches to the output LEDs.
4. Generation and cross compilation and execution of the C code to con- nect the input analog port to the output. Connect a microphone, speak into it and observe the output electrical signal on a DSO and store it.
5. Document the work.

Experiment 4. Linear convolution

1. Write a C function for the linear convolution of two arrays.
2. The arrays may be kept in different files and downloaded to the DSP hardware.
3. Store the result as a file and observe the output.
4. Document the work.

Experiment 5. FFT of signals

1. Write a C function for N - point FFT.
2. Connect a precision signal generator and apply 1 mV , 1 kHz sinusoid at the analog port.
3. Apply the FFT on the input signal with appropriate window size and observe the result.
4. Connect microphone to the analog port and read in real time speech.
5. Observe and store the FFT values.
6. Document the work.

Experiment 6. IFFT with FFT

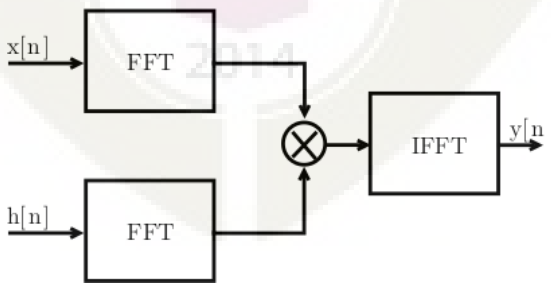
1. Use the FFT function in the previous experiment to compute the IFFT of the input signal.
2. Apply IFFT on the stored FFT values from the previous experiments and observe the reconstruction.
3. Document the work.

Experiment 7. FIR low pass filter

1. Use Python/scilab to implement the FIR filter response $h[n] = \frac{sin(\omega_c n)}{\pi n}$ for a filter size $N = 50$, $\omega_c = 0.1\pi$ and $\omega_c = 0.3\pi$.
2. Realize the hamming($w_H[n]$) and kaiser ($w_K[n]$) windows.
3. Compute $h[n]w[n]$ in both cases and store as file.
4. Observe the low pass response in the simulator.
5. Download the filter on to the DSP target board and test with 1 mV sinusoid from a signal generator connected to the analog port.
6. Test the operation of the filters with speech signals.
7. Document the work.

Experiment 8. Overlap Save Block Convolution

1. Use the file of filter coefficients From the previos experiment.
2. Realize the system shown below for the input speech signal $x[n]$.



3. Segment the signal values into blocks of length $N = 2000$. Pad the last

block with zeros, if necessary.

4. Implement the *overlap save* block convolution method
5. Document the work.

Experiment 9. Overlap Add Block Convolution

1. Use the file of filter coefficients from the previous experiment.
2. Realize the system shown in the previous experiment for the input speech signal $x[n]$.
3. Segment the signal values into blocks of length $N = 2000$. Pad the last block with zeros, if necessary.
4. Implement the *overlap add* block convolution method
5. Document the work.

Schedule of Experiments: Every experiment should be completed in three hours.

Textbooks

1. Vinay K. Ingle, John G. Proakis, “Digital Signal Processing Using MATLAB.”
2. Allen B. Downey, “Think DSP: Digital Signal Processing using Python.”
3. Rulph Chassaing, “DSP Applications Using C and the TMS320C6x DSK (Topics in Digital Signal Processing)”