Indian Institute of Technology Tirupati Department of Electrical Engineering

VLSI Circuits for Signal Processing

(EE5037) Instructor: Dr. Vikramkumar Pudi

PROJECT

Student Name: Venkata Saket Ram Goteti
Roll No. EE19B043

Student Name: M Suchithra
Roll No. EE19B047

Student Name: Surya Prathap Tyagi
Roll No. EE19B049

JPEG COMPRESSION:

JPEG Stands for Joint Photographic Expert Group

It was proposed by Nasir Ahmed in 1972. Who is an Indian and American electrical engineer and computer scientist.

JPEG uses transform coding, it is largely based on the following observations:

- Fact 1: The lower spatial frequency components contain more information than the high-frequency components.
- Fact 2: The human eyes are more immune to the loss of higher spatial frequency components than of lower frequency components

(a) Desert (Before Compression)

(b) Desert (After Compression)

Steps involved in JPEG Compression:

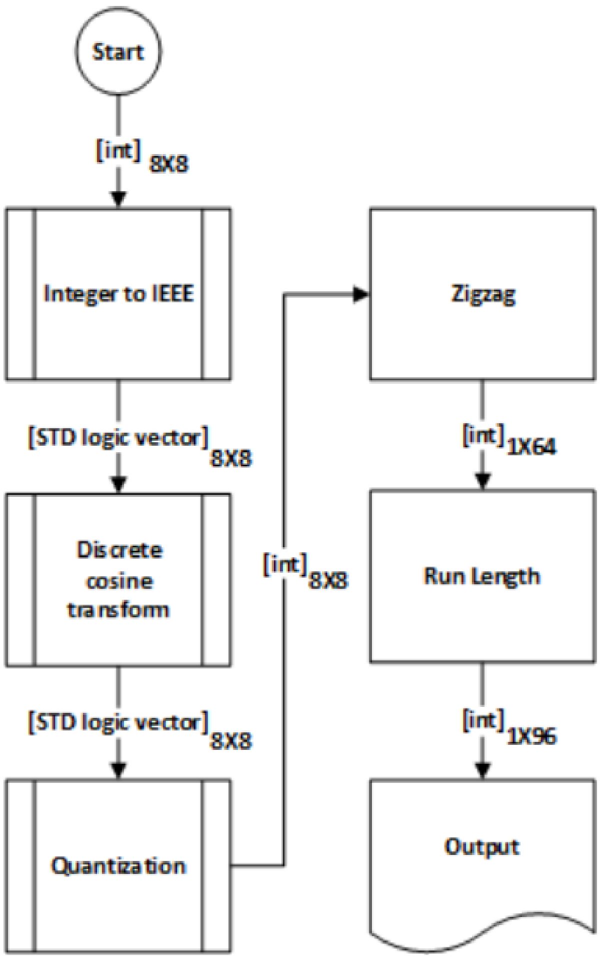


Image data :

```
.1001001    11001001    11001001    11001001    11001001    11000000    11000110    11000110
.1001001    11001001    11001001    11001001    11001001    11001000    11001001    11000100
.1001001    11000110    11001001    11001101    11001101    11001100    11001001    11000000
.1001001    11001101    11010110    11100000    11010111    11010110    11001001    11000000
.1001001    11011100    11101000    11101001    11100101    11100001    11001001    10111011
.1010001    11100000    11011011    11010001    11010110    11101001    11001101    11000000
.1001010    11011011    11100000    11101000    11100000    11100001    11001101    11000100
.1011000    11011011    11100100    11100100    11100000    11100110    11001101    11000011
```

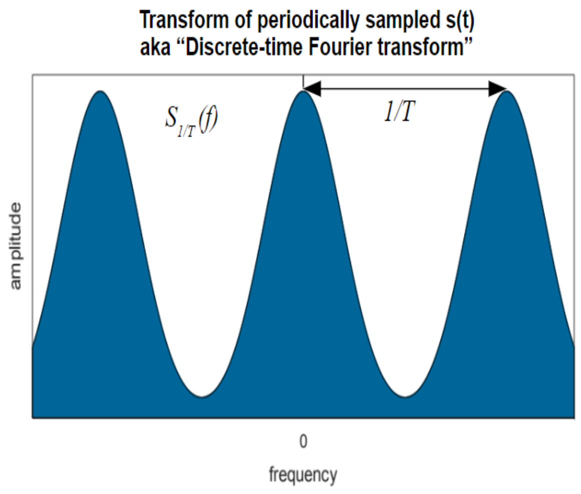
Image data file converted using matlab:

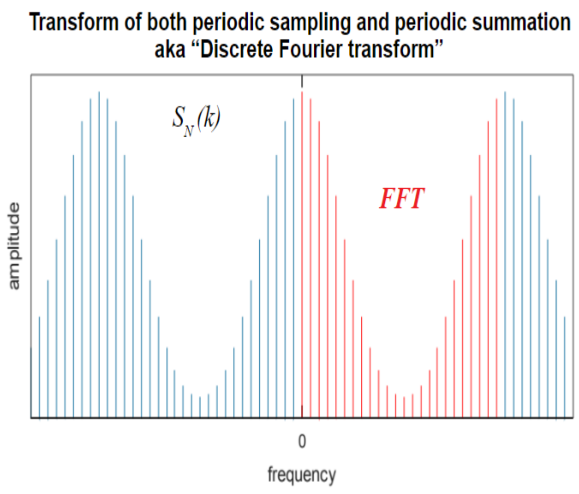
$0 < \text{Pixel Value} < 255 \Rightarrow [-128\ 127]$

We send one pixel for each clock cycle and perform DCT, Quantisation and Zig Zag.

Discrete Cosine Transform

DCT converts the information contained in a block(8x8) of pixels from spatial domain to the frquency domain.





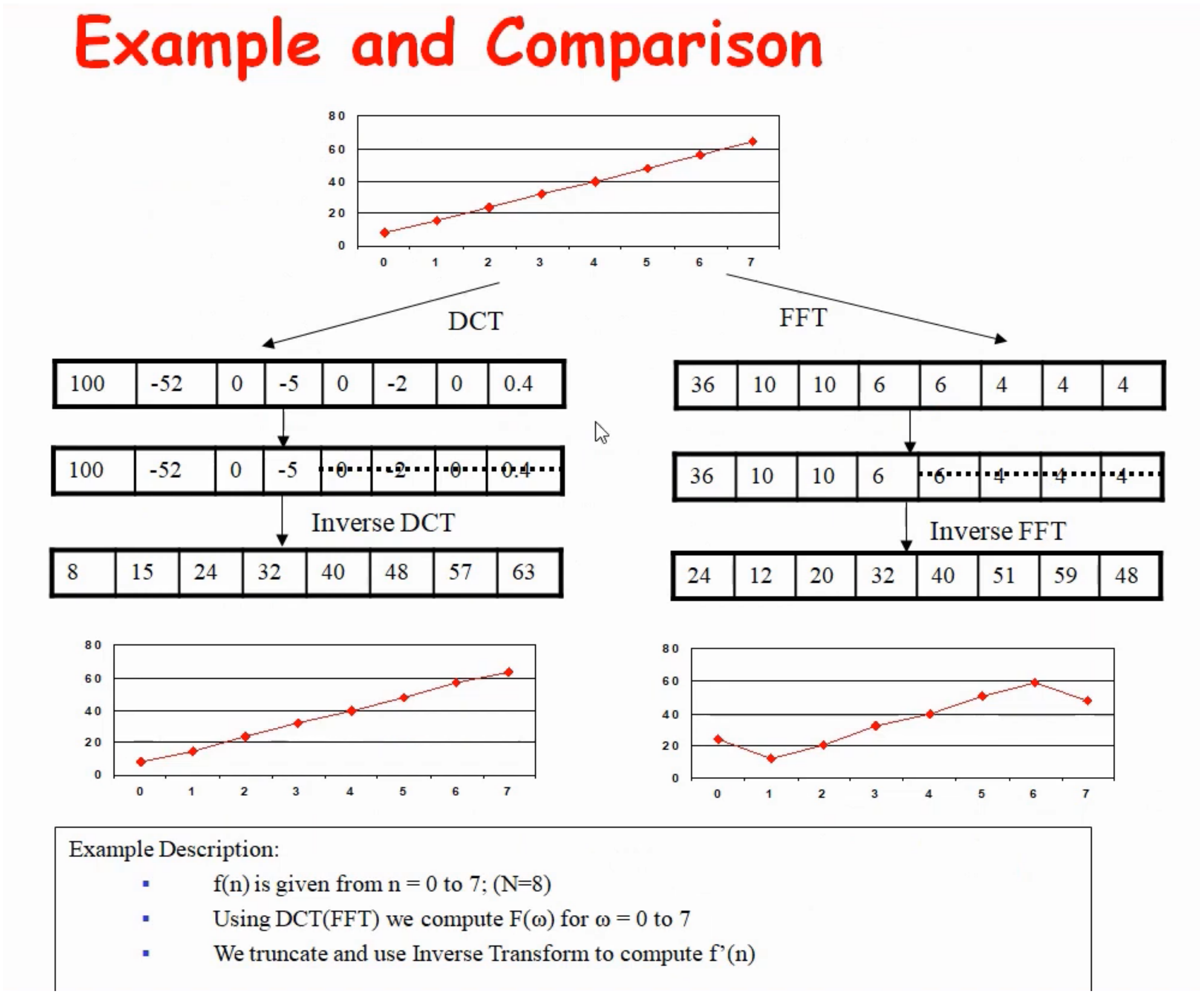
- $$DCT(\mathbf{u},\mathbf{v}) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7}P(x,y) \cdot cos\left[\frac{(2x+1)u\pi}{16}\right] cos\left[\frac{(2x+1)v\pi}{16}\right]$$

- $$DCT(\mathbf{u},\mathbf{v}) = C \cdot P \cdot C^T$$

- $$C(\mathbf{u},\mathbf{v}) = \begin{cases} \sqrt{\frac{2}{N}} & , u = 1 \\ \sqrt{\frac{2}{N}} cos\left[\frac{(u-1)(2v-1)\pi}{2N}\right] & , u \neq 1 \end{cases}$$

- $$\boxed{C=} \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}_{8\times 8}$$

Comaprision between FFT and DCT:



Example and Comparison

100	-52	0	-5	0	-2	0	0.4
100	-52	0	-5	~~0~~	~~-2~~	~~0~~	~~0.4~~
8	15	24	32	40	48	57	63

36	10	10	6	6	4	4	4
36	10	10	6	~~6~~	~~4~~	~~4~~	~~4~~
24	12	20	32	40	51	59	48

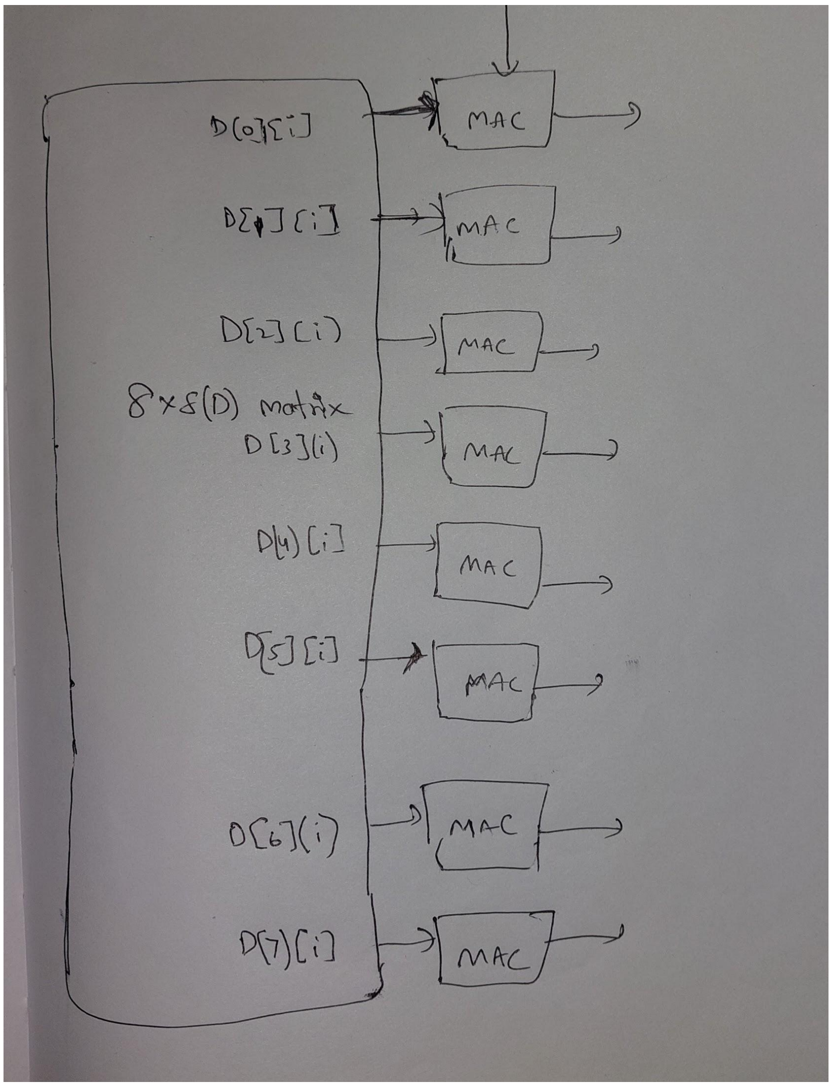
Example Description:

- f(n) is given from n = 0 to 7; (N=8)
- Using DCT(FFT) we compute F(ω) for ω = 0 to 7
- We truncate and use Inverse Transform to compute f'(n)

We can clearly observe that DCT is producing exact graph as f(n) whereas that is not the case with FFTSo, we consider DCT for JPEG Compression.

Architecture for XD^T

We have considered the input to come serially, one at a time for each clock cycle. The input pixel will be given to 8 MACs in the following way

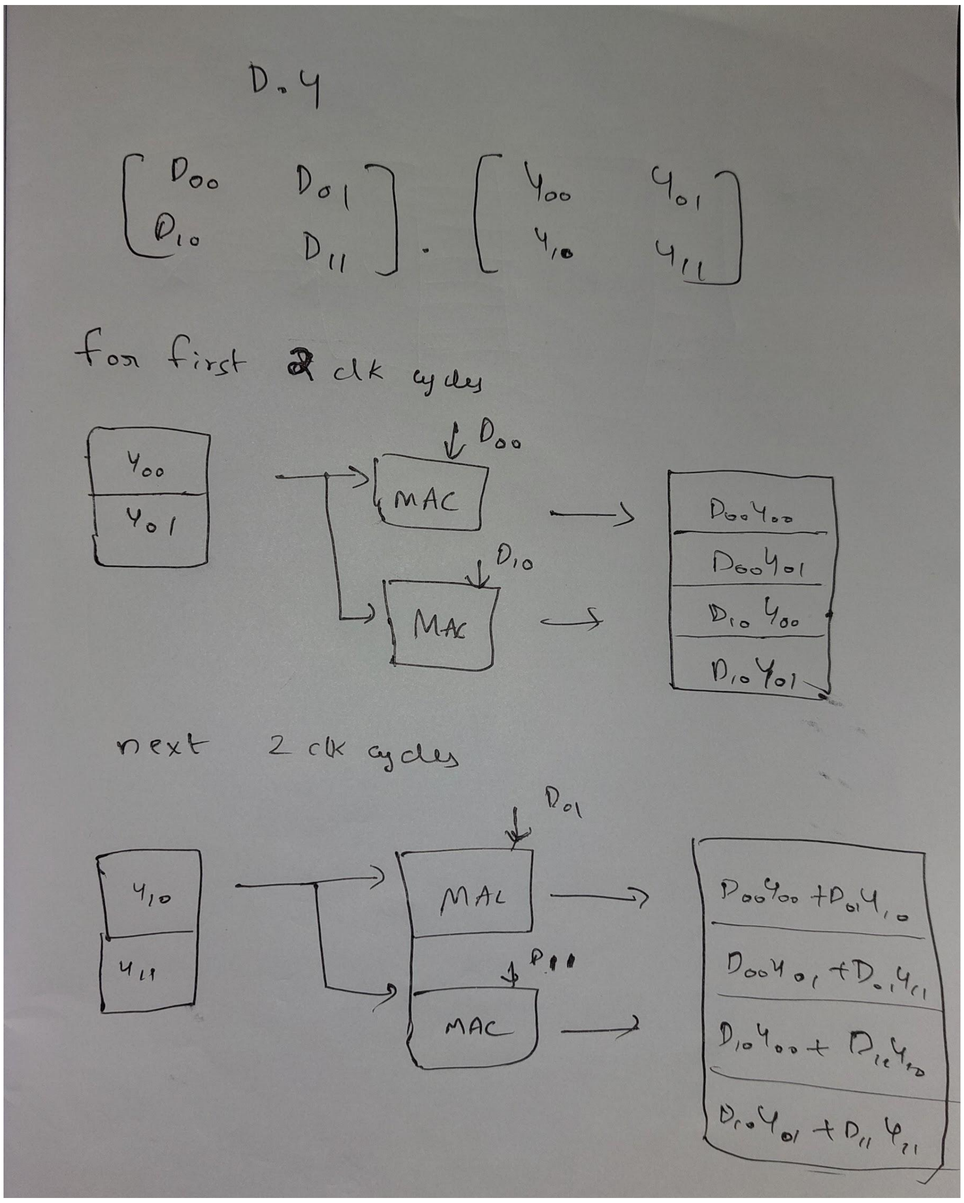


Where it go from 0 to 7 and back to 0 every time it sees a positive edge at the clock.

The output of this module comes after 8 clock cycles and stays for 8 clock cycles. The output is nothing but each row of matrix XDT.

Architecture for DY:

Each row of the XDT matrix is fed to this module. So input changes every 8 Clock cycles. Here is the architecture for a 2x2 matrix case.



Hence for an 8x8 matrix, using only 8 MACs and 64 registers, We can implement DY matrix multiplication. The output appears after 64 clock cycles. After 64 clock cycles, The first Column of the DY matrix appears, and the next clock cycle, the second column, and so on.

Quantization:

Quantization is the process of reducing the number of bits needed to store an integer value by reducing the precision of the integer.

Quantisation is a scalar division. In a scalar division, we only need to divide each number in the DCT matrix by the corresponding number in the quantization matrix

$$Q_{DCT} = round\left(\frac{DCT(u,v)}{Q(u,v)}\right)$$

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

We have used a non-uniform Quantisation table

Proposed Architecture for Quantization:

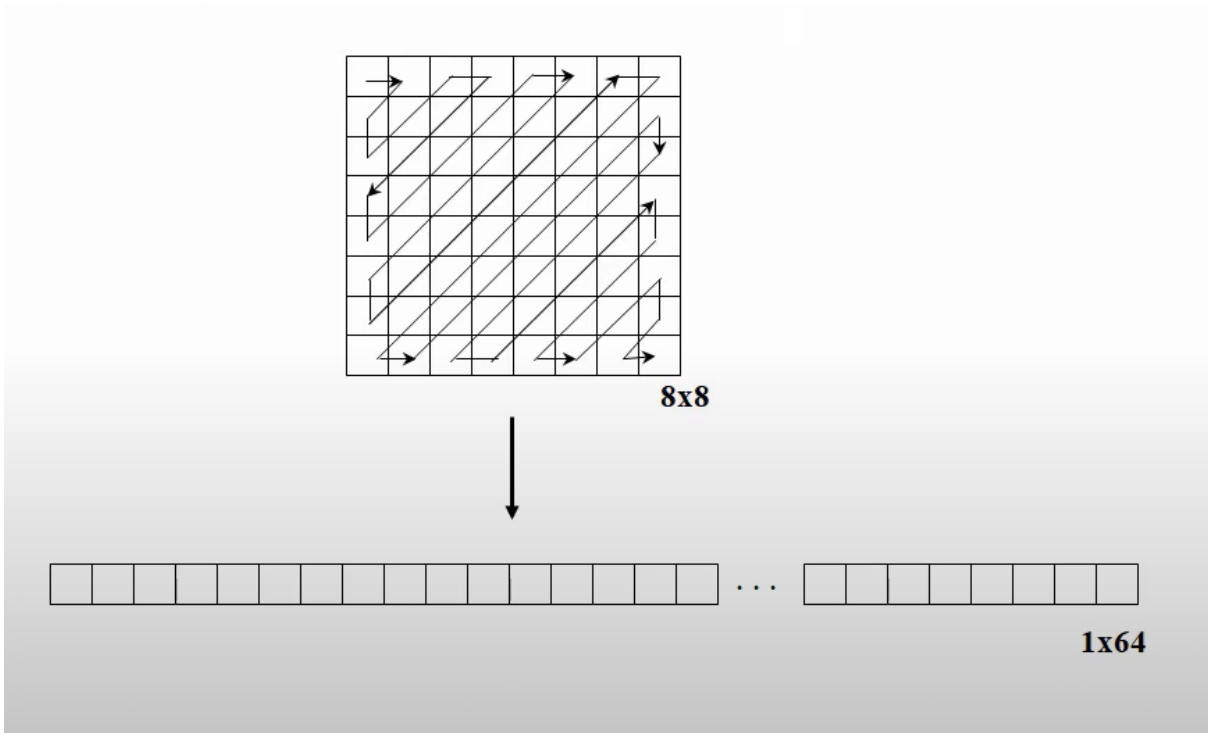
$$Q_{DCT} = (1/Q(u,v)).*(DCT(u,v))$$

The input of the Quantization matrix is a column vector of the DXDT matrix. Each clock cycle, a new column comes as an input.

We first compare the input with the corresponding element of the Q matrix, and if it is greater than that, we multiply it with 1/Q or make it zero otherwise.

Zig-Zag Scan:

We do Zig-Zag Scan to group low-frequency coefficients on top of the vector and high-frequency coefficients at the bottom.



Maps 8x8 to a 1x64 vector.

Simulation Results:

We were successful in writing each individual module, namely XDT, DY, Quantisation, and zigzag. Each Individual module was taking inputs accordingly to how the output of the previous module was. Each individual module was working correctly when simulated accordingly. But when we tried to integrate the modules together, some of the outputs of the DY matrix turned out to be wrong and there was some timing mistake in the way we structured our integration code. We sat down for hours trying to fix the problem but failed. Unfortunately, the deadline was reached and limited time with a busy internship schedule didn't help our case. But the effort we put in was never less than 100 percent. Bearing in mind that we are new to HDL Coding and had limited time, we are hoping that you will be liberal with grading. We always put a lot of effort in this course, whether it was the assignments or this project. We don't want to leave this project incomplete. After/during our internship, we will find time to complete this project.

Reference Paper:

JPEG Image Compression using the Discrete Cosine Transform: An Overview, Applications, and Hardware Implementation Ahmad Shawahna, Md. Enamul Haque, and Alaaeldin Amin Department of Computer Engineering King Fahd University of Petroleum and Minerals, Dhahran.

Other links:

- https://www.youtube.com/c/bhanuprakashreddy
- https://en.wikipedia.org/wiki/Discrete_Fourier_transform#/media/File:Fourier_transform,_Fourier_series,_DTFT,_DFT.svg