

# Implementation of 2D fft in Verilog using inbuilt 1-D fft ip in Vivado.

B Sathiya Naraayanan IMT2020534  
Ayyappa Koppuravuri IMT2020555

## ALGOGRITHM

- The input is a 2-D matrix A.
- 1-D FFT is performed on all the rows the input matrix and matrix B is formed.
- 1-D FFT is again performed on the columns of obtained matrix B to get the desired output output matrix.

## Algo check in matlab

```
inp = [  
    1,1,1,1,1,1,1,1;  
    2,2,2,2,2,2,2,2;  
    3,3,3,3,3,3,3,3;  
    4,4,4,4,4,4,4,4;  
    5,5,5,5,5,5,5,5;  
    6,6,6,6,6,6,6,6;  
    7,7,7,7,7,7,7,7;  
    8,8,8,8,8,8,8,8;  
];  
  
tmp = zeros(8,8);  
  
for i = 1 : 1 : 8  
    tmp(:,i) = fft(inp(:,i));  
end  
out = zeros(8,8);  
for i = 1 : 1 : 8  
    out(i,:) = fft(tmp(i,:));  
end  
  
out1 = fft2(inp);  
  
disp(inp)  
disp('out = ')  
disp(out)  
disp('out1 = ')  
disp(out1)
```

## The outputs for above code

```
>> fft_test
    1      1      1      1      1      1      1      1
    2      2      2      2      2      2      2      2
    3      3      3      3      3      3      3      3
    4      4      4      4      4      4      4      4
    5      5      5      5      5      5      5      5
    6      6      6      6      6      6      6      6
    7      7      7      7      7      7      7      7
    8      8      8      8      8      8      8      8

out =
    1.0e+02 *

    2.8800 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 + 0.7725i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 + 0.3200i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 + 0.1325i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 - 0.1325i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 - 0.3200i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 - 0.7725i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i

out1 =
    1.0e+02 *

    2.8800 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 + 0.7725i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 + 0.3200i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 + 0.1325i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 - 0.1325i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 - 0.3200i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   -0.3200 - 0.7725i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
```

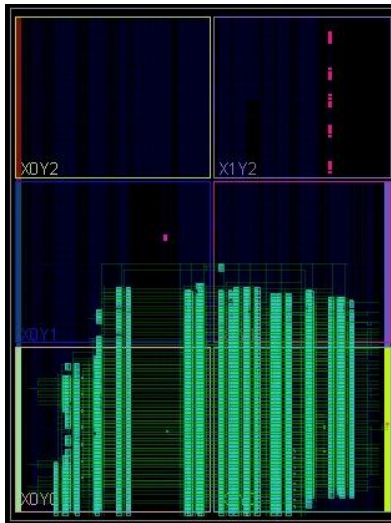
Here out is obtained using the above algorithms, where as out1 is obtained from the inbuilt fft2 function available on matlab.

The Algo validation is successful and done in matlab.

## Explanation of Input/output Flow in the code.

- We made a .sv file(fft\_1d.sv) to use the inbuilt FFT ip for solving 1-D fft.
- We made another .sv file(fft\_2d.sv) that takes a matrix as input and apply 1-D fft on all the rows and on the columns of the resulting matrix using fft\_1D module above.
- Then wrapper module is used to give inputs to fft\_2D module and obtain the results from it and display on ILA.
- The outputs are displayed on ILA using an 1-D array. Each column values of the resulting matrix is passed to the 1-D array so that all the columns are displayed on ILA using less resources.

## FLOORPLAN



## RESOURCE UTILIZATION

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	Block RAM Tile (50)	DSPs (90)	Bonded IOB (106)	BUFGCTRL (32)	BSCANE2 (4)
▼ N Wrapper	5474	13332	662	514	3030	4305	1169	16	8	1	2	1
> 🛠️ dbg_hub (dbg_hub)	444	727	0	0	211	420	24	0	0	0	1	1
> 🛠️ dut (FFT_2D)	3496	9157	594	514	2109	2936	560	1	8	0	0	0
> 🛠️ your_instance_name (ila_0)	1534	3448	68	0	780	949	585	15	0	0	0	0

## TIMING ANALYSIS

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.710 ns	Worst Hold Slack (WHS): 0.021 ns	Worst Pulse Width Slack (WPWS): 3.750 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 30864	Total Number of Endpoints: 30848	Total Number of Endpoints: 15477

All user specified timing constraints are met.

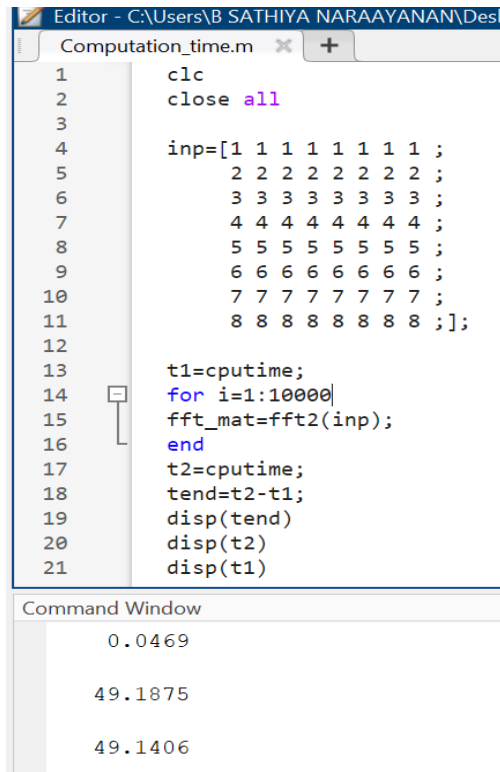
## COMPARISON OF EXECUTION TIME

The execution time in FPGA.

- No of clock cycle consumed for 1-D fft =102
- Time for one clock cycle =10 ns
- Total number of 1-D FFT for 2-D fft on 8x8 input =8+8 =16

The execution time in FPGA is (No of clk cycles for each 1-D fft)\*(clkperiod)\*(no of cycles) = 16.32 micro seconds.

## Extecutio time in matlab



The screenshot shows the MATLAB Editor window with a file named 'Computation\_time.m'. The code defines an 8x8 input matrix 'inp' and measures the execution time of an 8-point FFT using 'cputime' and 'fft2'. The Command Window displays the results of the timing measurements.

```
1      clc
2      close all
3
4      inp=[1 1 1 1 1 1 1 1 ;
5           2 2 2 2 2 2 2 2 ;
6           3 3 3 3 3 3 3 3 ;
7           4 4 4 4 4 4 4 4 ;
8           5 5 5 5 5 5 5 5 ;
9           6 6 6 6 6 6 6 6 ;
10          7 7 7 7 7 7 7 7 ;
11          8 8 8 8 8 8 8 8 ;];
12
13      t1=cputime;
14      for i=1:10000
15          fft_mat=fft2(inp);
16      end
17      t2=cputime;
18      tend=t2-t1;
19      disp(tend)
20      disp(t2)
21      disp(t1)
```

Command Window

```
0.0469
49.1875
49.1406
```

The execution time in case of matlab was 4.69 microseconds.

**Observation :** Matlab is performing better than Verilog implementation.

## Input 1

[illegible]

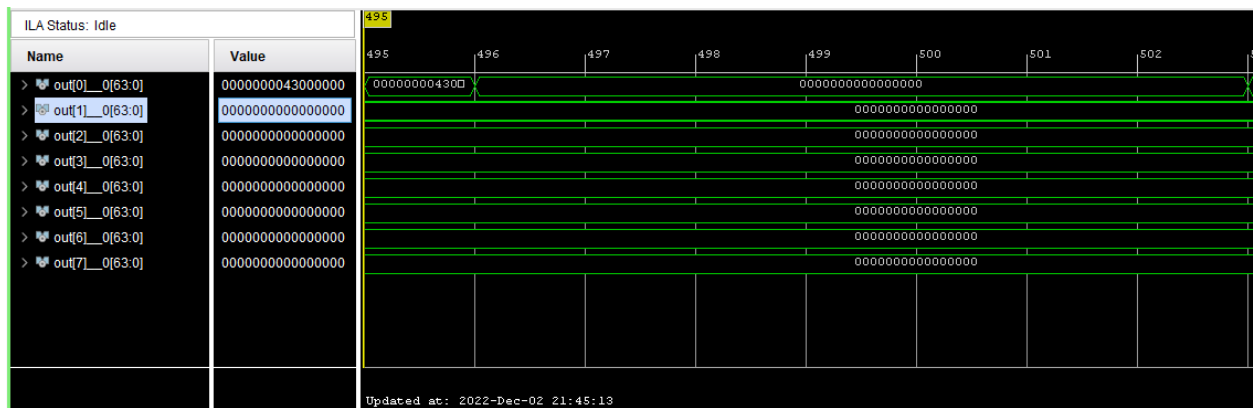
### Output observed in matlab

[illegible]

### Output of Matlab in a tabular form

[illegible]

Output observed on ILA



Output of ILA in a tabular form

128	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

**Test case 1 :** Only Real values are there in the outputs. The output obtained in ILA is matched with output obtained from matlab.

## Input 2

```
inp = [  
    1,1,1,1,1,1,1,1;  
    0,0,0,0,0,0,0,0;  
    0,0,0,0,0,0,0,0;  
    0,0,0,0,0,0,0,0;  
    1,1,1,1,1,1,1,1;  
    0,0,0,0,0,0,0,0;  
    0,0,0,0,0,0,0,0;  
    0,0,0,0,0,0,0,0;  
];
```

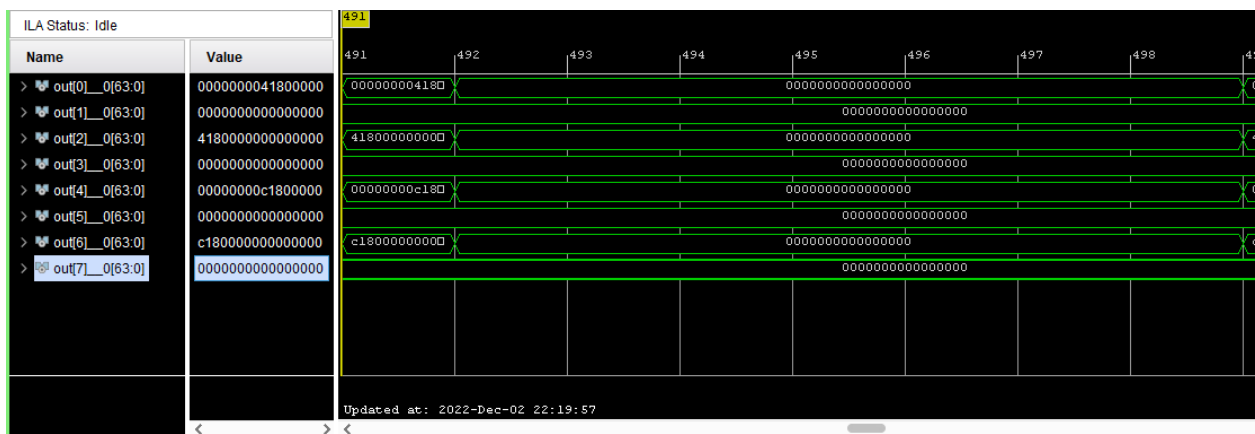
## Output observed in matlab

16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

## Output of Matlab in a tabular form

16	0	0	0	0	0	0	0
0j	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Output observed on ILA



Output of ILA in a tabular form

16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
16j	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-16j	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

**Test case 2 :** The Magnitudes of outputs obtained in matlab and FPGA implementation are same same but phase is different.



## Input 3

```
inp = [  
    1,1,1,1,1,1,1,1;  
    2,2,2,2,2,2,2,2;  
    3,3,3,3,3,3,3,3;  
    4,4,4,4,4,4,4,4;  
    5,5,5,5,5,5,5,5;  
    6,6,6,6,6,6,6,6;  
    7,7,7,7,7,7,7,7;  
    8,8,8,8,8,8,8,8;  
];
```

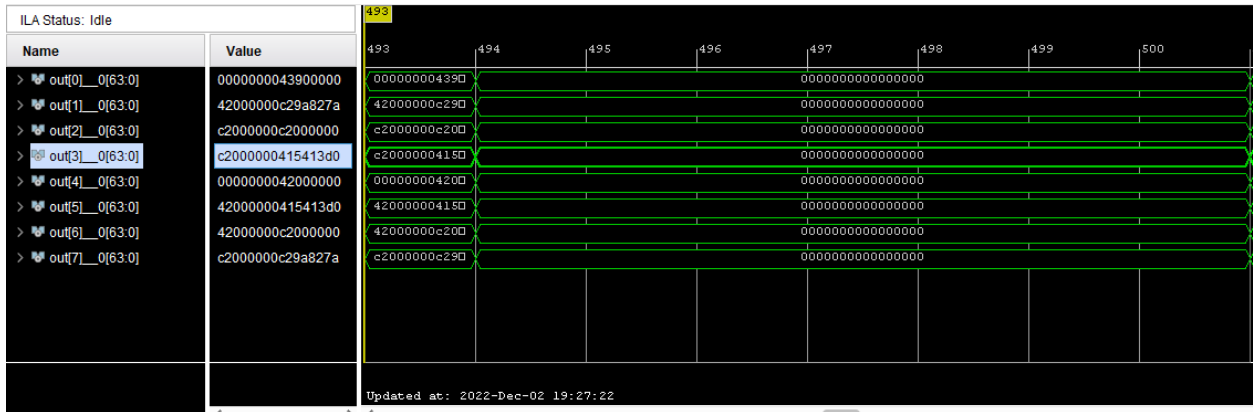
## Output observed in matlab

2.8800e+02 + 0.0000e+00i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
-32.0000 + 77.2548i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
-32.0000 + 32.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
-32.0000 + 13.2548i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
-32.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
-32.0000 - 13.2548i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
-32.0000 - 32.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
-32.0000 - 77.2548i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i

## Output of Matlab in a tabular form

288	0	0	0	0	0	0	0
-32+77.25j	0	0	0	0	0	0	0
-32+32j	0	0	0	0	0	0	0
-32+13.25j	0	0	0	0	0	0	0
-32	0	0	0	0	0	0	0
-32-13.25j	0	0	0	0	0	0	0
-32-32j	0	0	0	0	0	0	0
-32-77.25j	0	0	0	0	0	0	0

Output observed on ILA



Output of ILA in a tabular form

288	0	0	0	0	0	0	0
-77.25+32j	0	0	0	0	0	0	0
-32 - 32j	0	0	0	0	0	0	0
-32j+13.25	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0
32j+13.25	0	0	0	0	0	0	0
32j-32	0	0	0	0	0	0	0
-32j-77.25	0	0	0	0	0	0	0

**Test case 3 :** The Magnitudes of outputs obtained in matlab and FPGA implementation are same same but phase is different.

**Note :**

- The first 32 bits are real part and the next 32 are imaginary part. The FFT IP uses single precision data type.

**Future work**

- We gave input via a top wrapper module manually. We could have used Block Rams to give inputs.
- Finding the reason for the phase difference found in above cases while using fft inbuilt ip, and correcting it.
- We could replace the fft inbuilt ip with own created fft module in a better way so that Vivado implementation is better than fft implementation.
- Finding the reasons for the wrong invalid Post-synthesis, and Post implementation simulations.

**Resources**

- <https://youtu.be/v743U7gvLq0> link for How computers compute 2-D FFT. from 1-D fft.
- [https://youtu.be/Plo5fME\\_oyc](https://youtu.be/Plo5fME_oyc) FFT IP implementation video!!
- [https://www.youtube.com/watch?v=HKeaBs\\_3V04&t=192s](https://www.youtube.com/watch?v=HKeaBs_3V04&t=192s) FFT IP
- <https://www.youtube.com/watch?v=5QV6n360Nlo&t=477s> 1-D FFT