

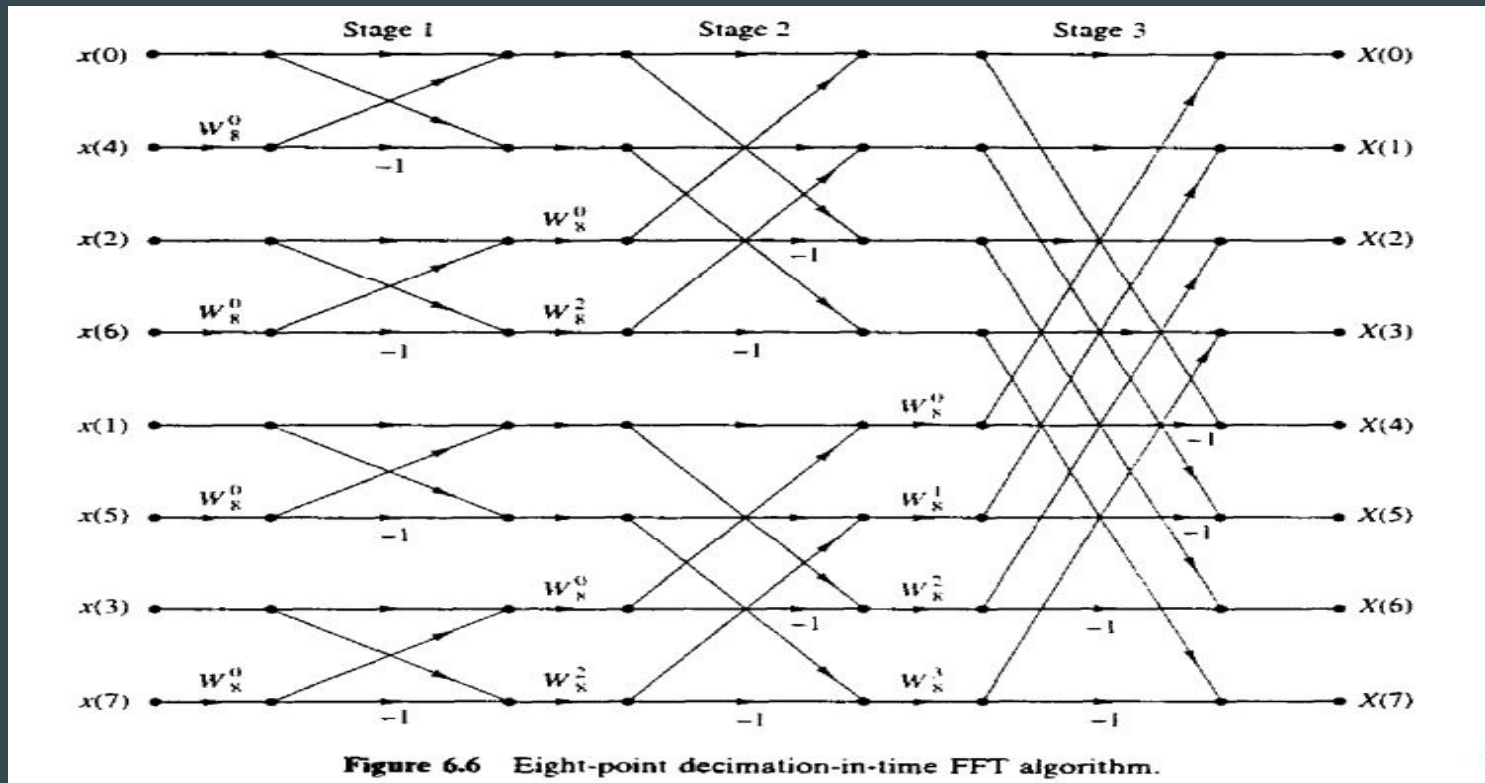
FAST FOURIER TRANSFORM

Fast Fourier Transform, or FFT, is an algorithm for computing the N point DFT with a computational complexity of $O(n \log n)$.

It is an efficient way of calculating the DFT of $x(n)$



Fast Fourier Transform:



Matlab FFT Code without pipelining:

In this code N point fft is implemented without pipelining i.e it is implemented without any synchronisation with the clock as soon as the input is given we get the output as that of combinational block.

Total elapsed time is 0.0026 sec for without pipelined processor for an 8 point fft.



home > ayush > Desktop > 201530241_assign2_dip > problem_2 > problem_a

Command Window

```
>> fftsp([1 2 3 4 5 6 7 8])
Elapsed time is 0.000153 seconds.
Elapsed time is 0.000040 seconds.
Elapsed time is 0.001653 seconds.
Elapsed time is 0.000035 seconds.
Elapsed time is 0.000027 seconds.
Elapsed time is 0.000236 seconds.
Elapsed time is 0.004178 seconds.

ans =

    36.0000 + 0.0000i
   -4.0000 + 9.6569i
   -4.0000 + 4.0000i
   -4.0000 + 1.6569i
   -4.0000 + 0.0000i
   -4.0000 - 1.6569i
   -4.0000 - 4.0000i
   -4.0000 - 9.6569i

>> 0.000153+0.000040+0.001653+0.000035+0.000027+0.000236+0.004178

ans =

    0.0026

fx >>
```

Editor - /home/ayush/Desktop/201530241_assign2_dip/problem_2/problem_a/fftsp.m

fftsp.m

```
1 function [X] = fftsp(x)
2 tic;
3 N = size(x,2);
4 if N>=4
5     Even = x([1:2:end]);
6     Odd = x([2:2:end]);
7     Evenfft = fftsp(Even);
8     Oddfft = fftsp(Odd);
9     Multiply = exp((-1i*2*pi)/N)*(0:N/2-1)';
10    X = [Evenfft + Multiply.*Oddfft;Evenfft - Multiply.*Oddfft];
11 else
12    X = [1 1;1 -1]*x';
13 end
14 toc;
15 end
```

4 usages of "N" found

fftsp

Ln 3 Col 1

Matlab Code with pipelining:

In this code N point fft is calculated with pipelining i.e implemented each stage in synchronisation with a clock but as clock cannot be generated externally I have used pause function in matlab for clock synchronisation each stage is set to 0.5 sec as the clock rate. It calculates each stage value in some p sec and then pause for $0.5 - p$ sec and then gives its value to the next stage.

By using this latency increases little bit but throughput of the function increases hence the efficiency increases.

The image displays the MATLAB R2016b interface. The top menu bar includes HOME, PLOTS, APPS, SHORTCUTS, EDITOR, PUBLISH, and VIEW. The toolbar contains icons for file operations (New, Open, Save, Compare, Print), navigation (Go To, Find), editing (Insert, Comment, Indent), breakpoints, and running (Run, Run and Advance, Run Section, Run and Time). The Command Window shows the execution of the `ffts` function with an 8-point input vector, displaying elapsed times for each iteration and the resulting complex-valued output vector.

The Editor window shows the `ffts.m` script with the following code:

```
1 function [X] = ffts(x)
2     tic;
3     N = size(x,2);
4     if N>=4
5         Even = x([1:2:end]);
6         Odd = x([2:2:end]);
7         Evenfft = ffts(Even);
8         Oddfft = ffts(Odd);
9         Multiply = exp((-1i*2*pi)/N)*(0:N/2-1)';
10        X = [Evenfft + Multiply.*Oddfft; Evenfft - Multiply.*Oddfft];
11    else
12        temp1 = clock();
13        X = [1 1; 1 -1]*x';
14        temp2 = clock();
15        pause(0.5-temp2(6)+temp1(6));
16    end
17    toc;
18 end
```

The HDL Code Generation panel shows an untitled project with a button to view code generation readiness issues. It lists a MATLAB Function `fftsp.m` and a MATLAB Test Bench `test.m`. A Workflow Advisor button is also present.

WAVEFORM:

EPWave Waveform Viewer - Mozilla Firefox

Navigation bar: S18TS18 | Intifada | Install M | (736) Stu | Edit cod | EDA EPWa x | Central | f (1) Faceb | Assignm | FFT algo | Untitled | f (1) Life | > + -

Address bar: <https://www.edaplayground.com/launchEpwave> Search

EPWave *beta* Load Save Examples About Apps ayush.malpani@students.iit.ac

EDA Playground <https://www.edaplayground.com/> From: 0s To: 1s

Get Signals Radix Q 100% < > - + x

0
ai[5:0]
ar[5:0]
bi[5:0]
br[5:0]
c1[5:0]
c2[5:0]
c3[5:0]
c4[5:0]
d1[5:0]
d2[5:0]
d3[5:0]
d4[5:0]
wi[5:0]
wr[5:0]
xi[5:0]
xr[5:0]
yi[5:0]
yr[5:0]
ai[5:0]
ar[5:0]
bi[5:0]
br[5:0]
c1[5:0]
c2[5:0]
c3[5:0]
c4[5:0]

FFT Using Verilog Without pipelining:

In this 8 point fft is implemented using verilog and without pipelining i.e all blocks are considered as asynchronous combinational blocks as soon one block gets its output it sends its output to the next block.

I have used online EDA - tool for synthesis and simulation of code.

FFT Using Verilog With pipelining:

In this 8 point fft is implemented using verilog and with pipelining i.e all blocks are considered as synchronous blocks which gives its value to the next stage on rising edge of the clock.

However latency of pipelined processor increases slightly throughput increases.

WITHOUT PIPELINING SYNTHESIS REPORT:

Edit code - EDA Playground - Mozilla Firefox

https://www.edaplayground.com/#

EDA playground

Run Save*

Brought to you by DOULOS

Languages & Libraries

Testbench + Design

SystemVerilog/Verilog

UVM / OVM

None

Other Libraries

None

OVL 2.8.1

SVUnit 2.11

Enable TL-Verilog

Enable Easier UVM

Tools & Simulators

Yosys 0.3.0

use ABC with cell library

Show diagram after run

Download files after run

Examples

Community

Collaborate

Forum

Follow @edaplayground

testbench.sv

```
SV/Verilog Testbench
7 wire[5:0] br[0:7];
8 wire[5:0] bi[0:7];
9 real wr[0:6];
10 real wi[0:6];
11 wire[5:0] xr[0:7];
12 wire[5:0] xi[0:7];
13 twobutterfly
14 t1(sr[0],si[0],sr[4],si[4],wr[0],wi[0],ar[0],ai[0],ar[1],ai[1]);
15 twobutterfly
16 t2(sr[2],si[2],sr[6],si[6],wr[0],wi[0],ar[2],ai[2],ar[3],ai[3]);
```

design.sv

```
SV/Verilog Design
10 endmodule
11
12
13 module twobutterfly(ar,ai,br,bi,wr,wi,xr,xi,yr,yi);
14 input[5:0] ar,ai,br,bi,wr,wi;
15 output[5:0] xr,yr,xi,yi;
16 wire[5:0] d1,d2,d3,d4;
17 wire[5:0] c1,c2,c3,c4;
18 reg[5:0] xr,yr,xi,yi;
19 multiply m1(bi,wi,c1);
20 multiply m2(br,wr,c2);
```

Log Share

ABC: Nodes = 212. Total 5-feasible cuts = 5863. Per node = 27.7. Time = 0.01 sec

ABC: Delay : Delay = 0.00 Flow = 91.6 Area = 804.0 0.0 % Time = 0.00 sec

ABC: AreaFlow : Delay = 23.00 Flow = 91.6 Area = 804.0 0.0 % Time = 0.00 sec

ABC: Area : Delay = 23.00 Flow = 0.0 Area = 767.0 4.6 % Time = 0.00 sec

ABC: Area : Delay = 23.00 Flow = 0.0 Area = 704.0 12.4 % Time = 0.00 sec

ABC: Output n35 : Delay = (23.00, 23.00) POS

ABC: Output n79 : Delay = (20.00, 20.00) POS

ABC: Output n70 : Delay = (15.00, 15.00) POS

ABC: Output n56 : Delay = (9.00, 9.00) POS

ABC: Output n37 : Delay = (4.00, 4.00) POS

ABC: Total runtime = 0.01 sec

12.1.2. Re-integrating ABC results.

ABC RESULTS: NAND cells: 61

ABC RESULTS: NOR cells: 91

ABC RESULTS: NOT cells: 32

ABC RESULTS: internal signals: 63

ABC RESULTS: input signals: 12

ABC RESULTS: output signals: 6

12.1.3. Removing temp directory `/tmp/yosys-abc-7CbH4o':

WITH PIPELINING SYNTHESIS REPORT:

Edit code - EDA Playground - Mozilla Firefox

jee main | S18TS18 | Intifada | Install M | (736) Stu | EDA Edit c | Central | Assignm | FFT algo | Untitled | f (1) Faceb | Getting | +

https://www.edaplayground.com/#

EDA playground

Run Save*

Log Share

testbench.sv design.sv b

Brought to you by DOULOS

▼ Languages & Libraries

Testbench + Design

SystemVerilog/Verilog

UVM / OVM

None

Other Libraries

None

OVL 2.8.1

SVUnit 2.11

☐ Enable TL-Verilog

☐ Enable Easier UVM

▼ Tools & Simulators

Yosys 0.3.0

use ABC with cell library

☐ Show diagram after run

☐ Download files after run

► Examples

▼ Community

Collaborate

Forum

Follow @edaplayground

ABC: + map -v

ABC: Converting "demo.genlib" into supergate library "demo.super".

ABC: Performing mapping with choices.

ABC: Maximum level: Original = 16. Reduced due to choices = 16.

ABC: Choice stats: Choice nodes = 51. Total choices = 104.

ABC: Nodes = 395. Total 5-feasible cuts = 7685. Per node = 19.5. Time = 0.01 sec

ABC: Delay : Delay = 0.00 Flow = 422.0 Area = 1484.0 0.0 % Time = 0.00 sec

ABC: AreaFlow : Delay = 17.00 Flow = 422.0 Area = 1484.0 0.0 % Time = 0.00 sec

ABC: Area : Delay = 17.00 Flow = 0.0 Area = 1422.0 4.2 % Time = 0.00 sec

ABC: Area : Delay = 17.00 Flow = 0.0 Area = 1342.0 9.6 % Time = 0.00 sec

ABC: Output n71 : Delay = (17.00, 17.00) POS

ABC: Output n101 : Delay = (17.00, 17.00) NEG

ABC: Output n178 : Delay = (17.00, 17.00) NEG

ABC: Output n204 : Delay = (17.00, 17.00) NEG

ABC: Output n66 : Delay = (15.00, 15.00) POS

ABC: Total runtime = 0.02 sec

12.1.2. Re-integrating ABC results.

ABC RESULTS: BUF cells: 2

ABC RESULTS: NAND cells: 122

ABC RESULTS: NOR cells: 170

ABC RESULTS: NOT cells: 58

ABC RESULTS: internal signals: 145

ABC RESULTS: input signals: 36

ABC RESULTS: output signals: 24

12.1.3. Removing temp directory `/tmp/yosys-abc-AiaRkw`:

[...]

COMPARISON BETWEEN PIPELINED AND WITHOUT:

	PIPELINED	WITHOUT PIPELINE
Area	1400	890
Throughput	100 instructions per sec(0.01 sec for one ins)	300 instructions per second (0.01/3 sec for one ins)
Latency	Around 0.01 sec	Slightly greater than 0.01 sec
Cost	More for pipelined as number of gates and area is more	Less as number of gates is less and area reqd. Is less
Gates reqd	120 nand cells as d-flip flops are required more number of gates	70 nand cells