



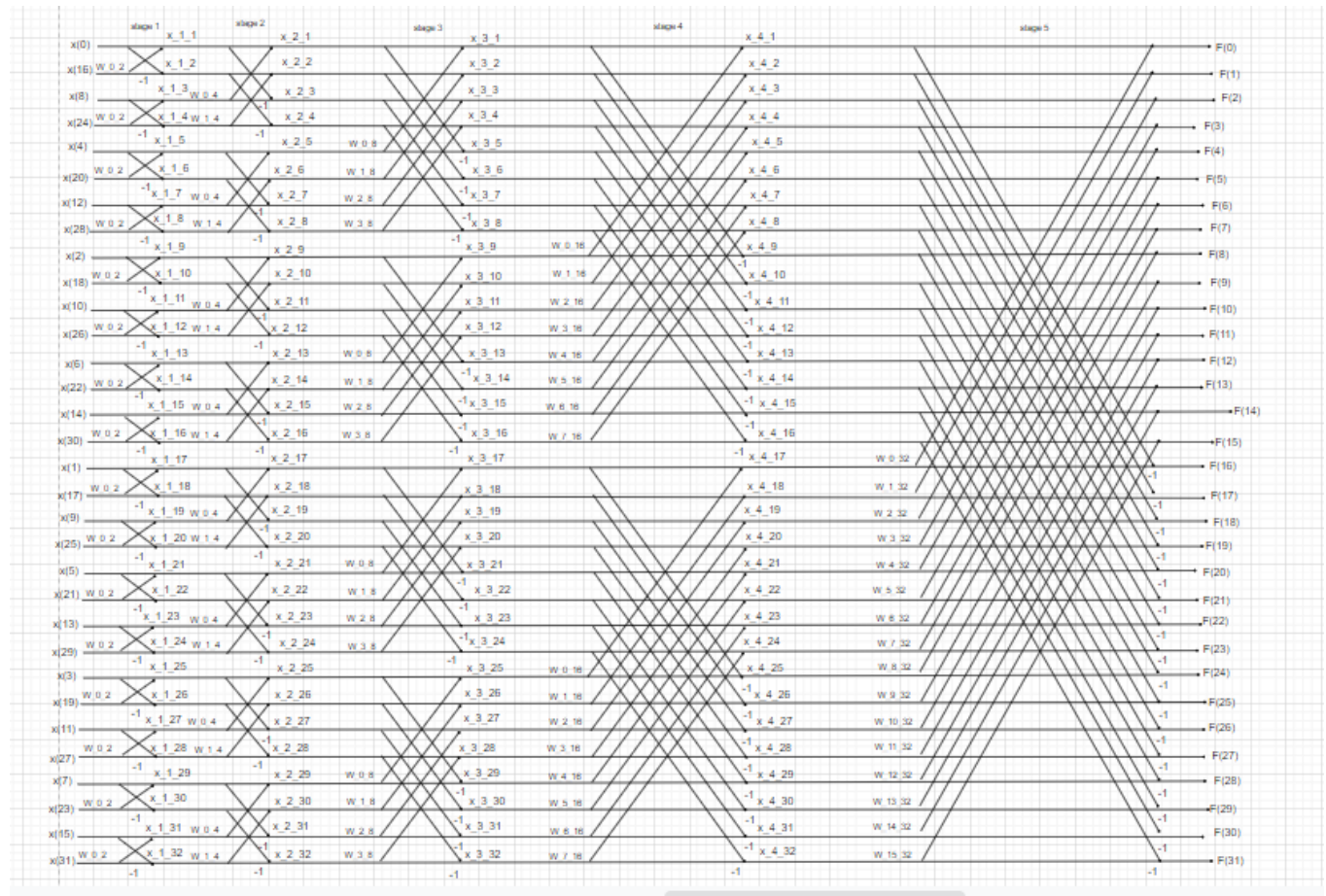
VLSI-2 project  
Design a 32-point FFT  
Submitted to Dr. Kareem Osama

Name	Section	B.N
عمر أحمد سعيد	3	23
محمد أحمد صالح	3	39
مصطفى محمد عبدالحميد	4	26

## Cooley-Tukey Algorithm:

The Cooley-Tukey algorithm is the most used FFT algorithm and here is the schematic of it which depend on several times of 2-point FFT executed by 2-point multiply accumulator (MAC).

## Diagram of the Algorithm:



32-FFT butterfly diagram

## Assumptions:

- We assume each sample from 32 input samples represented in 48 bits 24 bits for real part and 24 bits for imaginary part (inputs have only real part, outputs of MAC and final outputs have both real part and imaginary part).
- 24 bits for each part and divided to 1 bit for sign, 13 bits for integer and 10 bits for fraction.
- MAC operates at 100 MHz
- Final outputs at 20 MHz

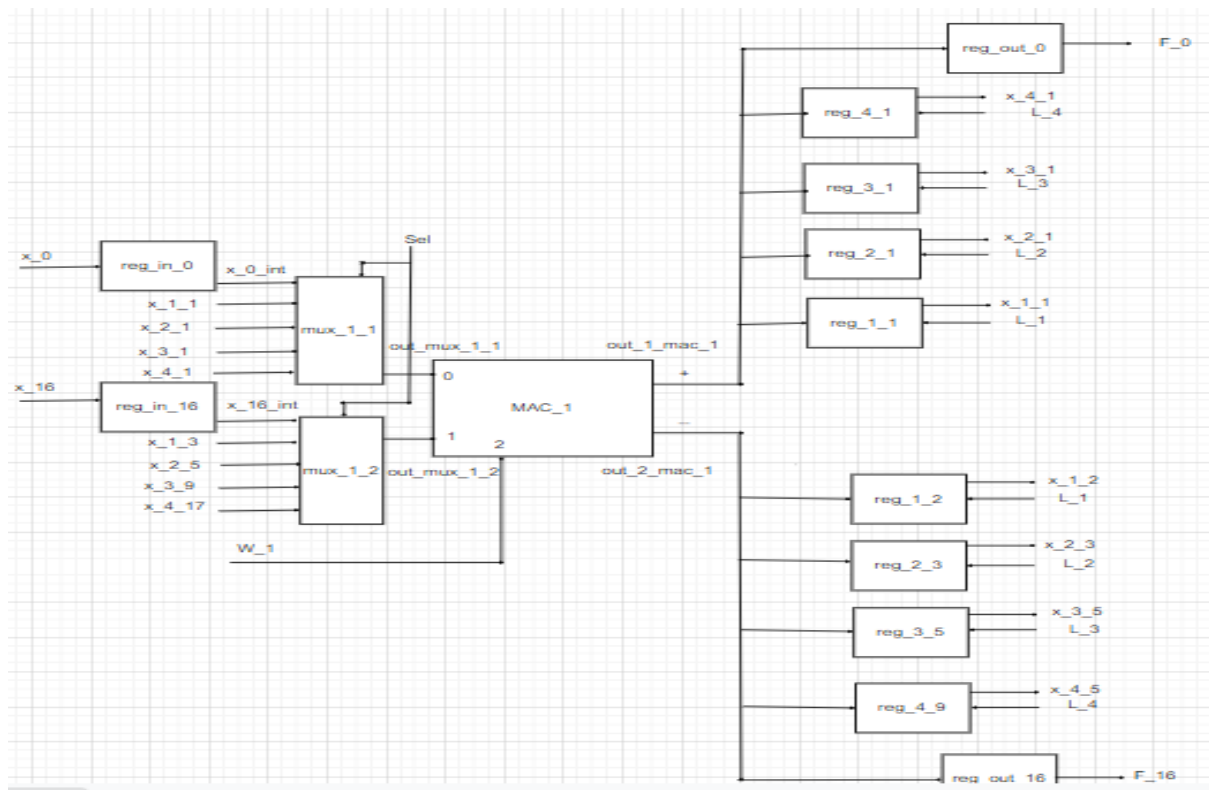
## **Design of the algorithm and MAC:**

### **Steps of the design:**

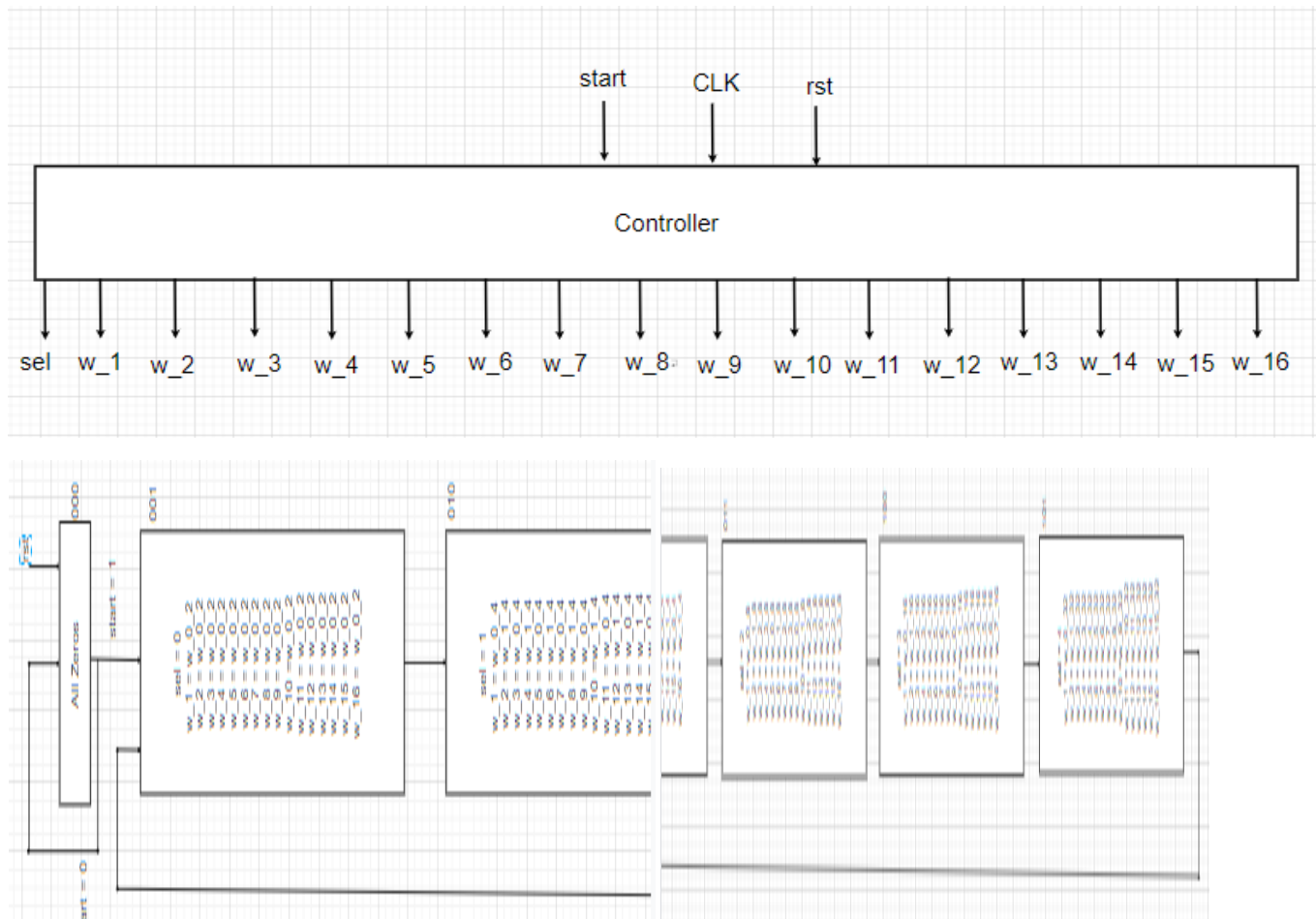
- If we look at the butterfly diagram, we can see it as five columns. each column has 32 inputs and 16 butterfly operations. So, we can execute one full column in one cycle of the fastest clock (100 MHz) by 16 MACs operate in parallel together at the same time.
- Each MAC has 3 inputs, 2 MUX and twiddle factor (which come from the control unit with the selection lines of the mux). each Mux have 5 inputs. The first one of them is the input sample which is used for operation of the first column and the other 4 inputs are feedback from the output register of the other 16 mac or even from the same mac.
- Selection lines of the mux choose second input to pass through the mux to execute the second operation which produce outputs of the second column and so on until the execution of the 5<sup>th</sup> column which will be the desired output.
- Output of each MAC is registered at two registers.
- One operates at 100 MHz which is connected to the suitable mux of other MACs
- The second register operates at 20 MHz which will capture the final sample after the execution of the 5<sup>th</sup> column.

So, we have here latency of one clock cycle thanks to the full utilization of the MAC which is used five times in each cycle.

**Below is schematic of one sample of 16 MAC instances are used in the design:**



**Controller and state diagram:**



**Hint:**

- All design files and modules are made from scratch.
- all graphes, photos and Schematics are atached with design file.
- Schematics are made with drawio website and all its files have the (.drawio) extension.

## MATLAB results Vs Post Synthesis simulations:

### 1. Time constrained passed with +ve slack:

```
Timing Details:
-----
All values displayed in nanoseconds (ns)

-----
Timing constraint: Default period analysis for Clock 'clk'
Clock period: 9.489ns (frequency: 105.389MHz)
Total number of paths / destination ports: 173669883494718 / 1591
-----
Delay:          9.489ns (Levels of Logic = 50)
Source:         controller_inst/current_state_FSM_FFd4_1 (FF)
Destination:    inst_l0/mac_inst/output_ima_sub_22 (FF)
Source Clock:   clk rising
Destination Clock: clk rising
```

### 2. First test with input samples [2, 2, 2, 0, 0, 0, .....]

MATLAB result:

Vs

design result:



	1
1	6.0000 + 0.0000i
2	5.8093 - 1.1555i
3	5.2620 - 2.1796i
4	4.4283 - 2.9589i
5	3.4142 - 3.4142i
6	2.3458 - 3.5107i
7	1.3512 - 3.2620i
8	0.5424 - 2.7269i

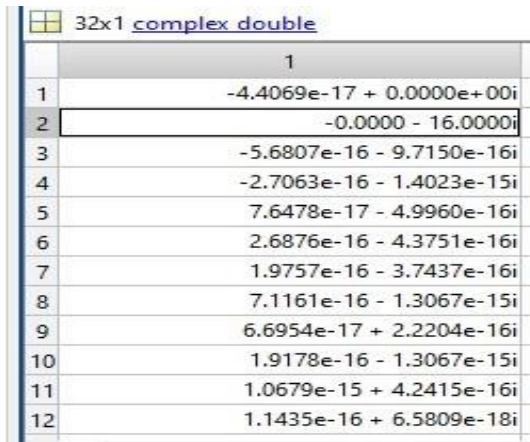
```
-----X[k]-----
X[0] = 6.000000 + j 0.000000
X[1] = 5.000000 - j 1.000000
X[2] = 5.000000 - j 2.000000
X[3] = 4.000000 - j 2.000000
X[4] = 3.000000 - j 3.000000
X[5] = 2.000000 - j 3.000000
X[6] = 1.000000 - j 3.000000
X[7] = 0.000000 - j 2.000000
ISim> |
```

### 3. second test with Sin(wt) samples

MATLAB result:

Vs

design result:

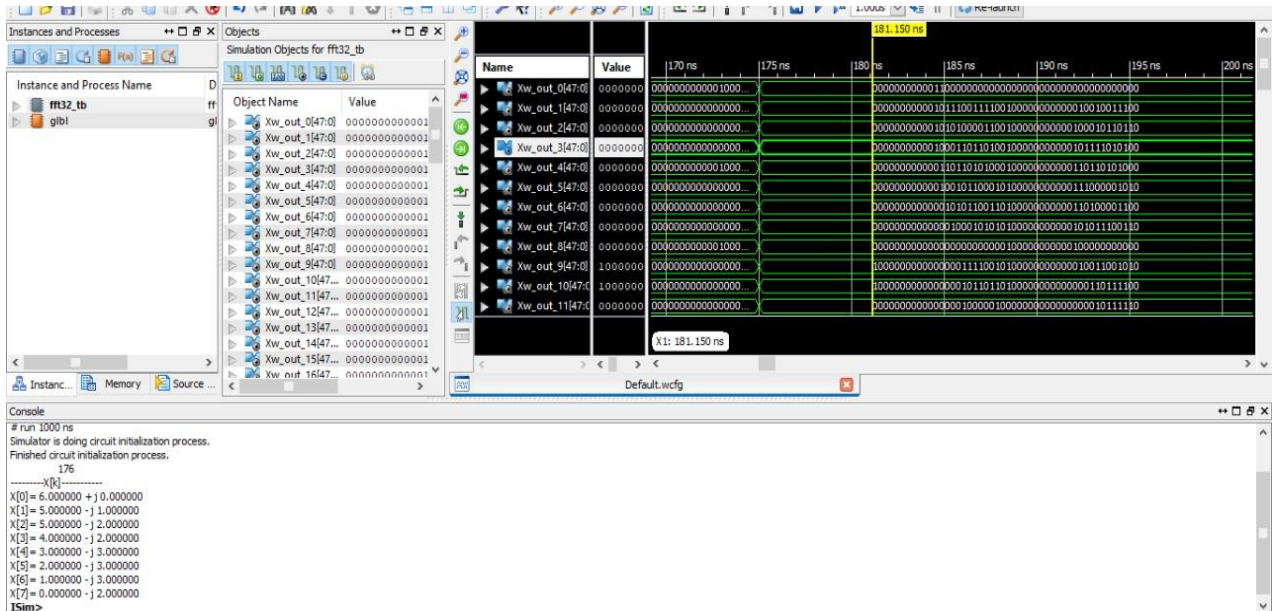


	1
1	-4.4069e-17 + 0.0000e+00i
2	-0.0000 - 16.0000i
3	-5.6807e-16 - 9.7150e-16i
4	-2.7063e-16 - 1.4023e-15i
5	7.6478e-17 - 4.9960e-16i
6	2.6876e-16 - 4.3751e-16i
7	1.9757e-16 - 3.7437e-16i
8	7.1161e-16 - 1.3067e-15i
9	6.6954e-17 + 2.2204e-16i
10	1.9178e-16 - 1.3067e-15i
11	1.0679e-15 + 4.2415e-16i
12	1.1435e-16 + 6.5809e-18i

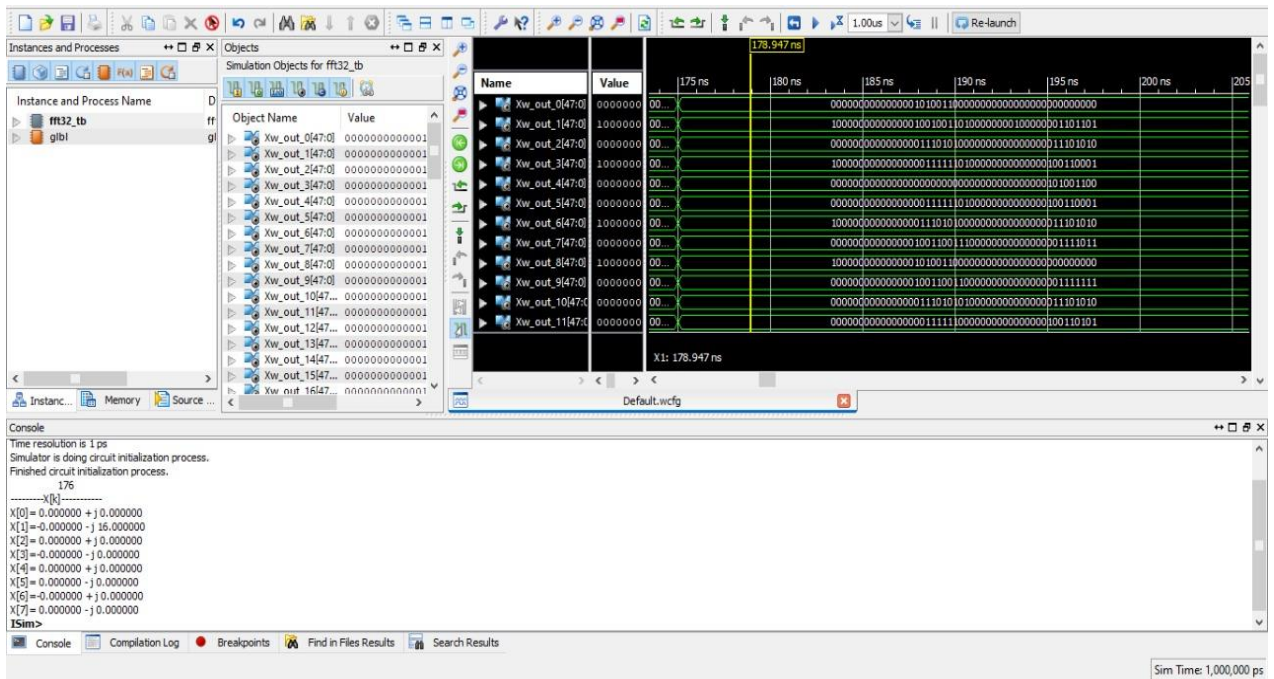
```
-----X[k]-----
X[0] = 0.000000 + j 0.000000
X[1] = -0.000000 - j 16.000000
X[2] = 0.000000 + j 0.000000
X[3] = -0.000000 - j 0.000000
X[4] = 0.000000 + j 0.000000
X[5] = 0.000000 - j 0.000000
X[6] = -0.000000 + j 0.000000
X[7] = 0.000000 - j 0.000000
```



#### 4. post synthesis simulation: first test :



#### Second test:



All codes, figures, schematics and synthesis report are attached with this links:

- [Google Drive](#)
- [GitHub](#)