FACULTY OF ENGINEERING
ELECTRONICS & ELECTRICAL
COMMUNICATIONS DEP.

Fast Fourier Transform

Digital Designing and implementation of 32-point FFT block using Cooley-Tukey algorithm with 5-stages pipelining, stages time sharing, Designed from High Level modelling, RTL modelling to logic synthesis.

Table of Contents

List Of Figures

Introduction:

The problem of frequency selective channels is a significant one in wireless communications. An equalizer can be used to invert the channel and cancel the effect of channel selectivity, but it's too complex to implement if the channel is too selective. A better approach to combat selective channels Is orthogonal frequency division multiplexing (OFDM) in which the channel is divided into many small subchannels, converting the selective channel into small flat subchannels. At the transmitter, the subcarriers are modulated in frequency domain, then IFFT is used to transform this into time domain samples. At the receiver, the time domain samples are transformed back to frequency domain using FFT for further processing.

Project description

The required is to design a 32-point FFT in RTL by using The Cooley-Tukey algorithm which is the most used FFT algorithm. To get the simplest design, we use this algorithm and break down the FFT into 2-point FFT.

- ➔ To understand the difference between DFT and FFT and how to implement FFT using The Cooley-Tukey algorithm please go to https://www.youtube.com/watch?v=EsJGuI7e_ZQ

High Level System Modelling

The system can be modelled by the butterfly diagram which is shown below:

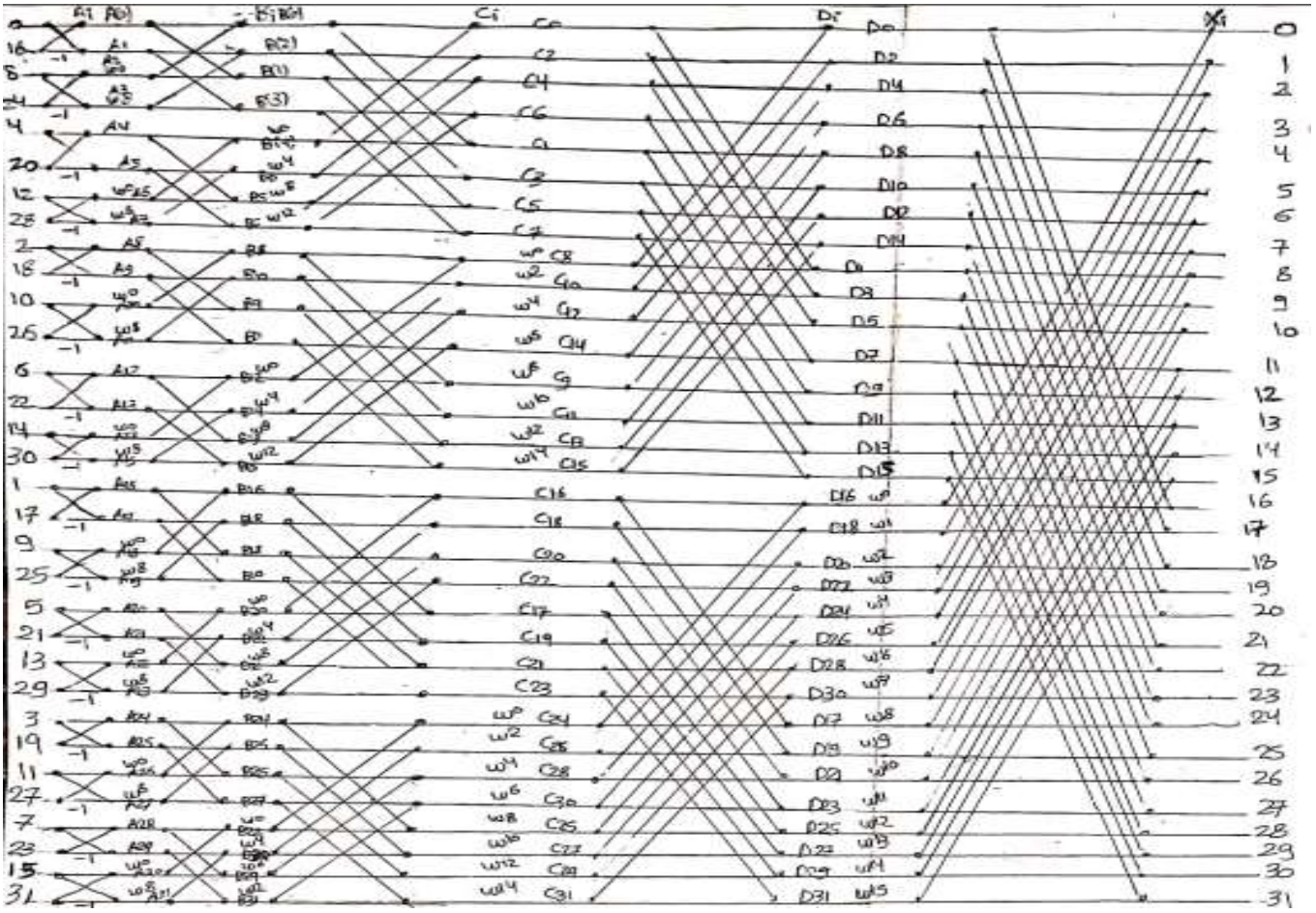

Figure 1: 32-point FFT by Butterfly Diagram

In the above diagram, the system is represented by five stages which takes 32-input in time domain and go through the stages to finally produce 32-output in frequency domain, the approach depends on making two-input butterfly for each two inputs and producing two outputs, that's by the following equations:

$$out1 = in1 + in2 * W_{32}^{k},\ out2 = in1 - in2 * W_{32}^{k}$$

And hence each stage takes 32 inputs which is distributed in certain order and then produces 32 outputs to the next stage governed by the above diagram.

➔ The butterfly diagram is modelled by **MATLAB** and gives the following results:

x	$FFT(x)$ *by MATLAB*	$FFT(x)$ *by butterfly*	*Error*
0.014455 - 0.126667i	0.441757 + 0.213603i	0.4417577 + 0.2136033i	0.0000000 + 0.000000i
0.125529 + 0.039827i	-0.15530 - 1.009028i	-0.155302 - 1.0090282i	1.6653e-16 + 0.000000i
0.028245 - 0.019142i	-0.31133 - 0.520417i	-0.3113348 - 0.520417i	-1.1102e-16 + 0.000000i
0.047868 - 0.217864i	0.953586 - 1.754237i	0.9535863 - 1.754237i	-2.2204e-16 + 0.000000i
0.155870 - 0.113421i	-1.022721 + 0.3753358i	-1.0227216 + 0.375335i	0.0000000 + 1.665e-16i
0.011343 + 0.177033i	0.5881526 +0.57917327i	0.5881526 + 0.5791732i	-1.11e-16 - 1.1102e-16i
-0.056715 + 0.1985140i	0.1543201 + 0.3982074i	0.1543201 + 0.3982074i	1.11e-16 - 5.5511e-17i
0.1070853 + 0.0131092i	0.7600733 - 0.300173i	0.7600733 - 0.300173i	1.1102e-16 - 5.551e-17i
0.0321626 - 0.0538364i	0.5939489 + 0.5159849i	0.5939489 + 0.5159849i	1.1102e-16 + 0.000000i
-0.160227- 0.0034463i	1.4053853 - 1.2194050i	1.4053853 - 1.219405i	-2.220e-16 + 0.0000000i
0.0298709- 0.0606668i	-1.0770367 + 0.530246i	-1.077036 + 0.530246i	-2.220e-16 + 0.0000000i
0.2482999 - 0.0982133i	-0.793085 - 0.2970659i	-0.793085- 0.2970659i	1.1102e-16 + 5.551e-17i
0.1317472 + 0.0081936i	-0.379705 - 0.1509708i	-0.379705 - 0.1509708i	-1.110e-16 + 8.3266e-17i
-0.056980 + 0.0239011i	-0.261978 - 0.1403939i	-0.261978 - 0.1403939i	5.551e-17 + 1.1102e-16i
0.0211006+ 0.1230195i	-0.198495- 0.1400129i	-0.198495 - 0.1400129i	-1.11e-16 + 2.77555e-17i
0.1816322 + 0.2414530i	-0.15682 - 0.142742i	-0.156820 - 0.142742i	8.326e-17 + 2.7755e-17i
-0.091909 + 0.0746960i	-0.126239 - 0.1475279i	-0.126239 - 0.1475279i	0.0000000 + 0.0000000i
-0.223389 - 0.040971i	-0.102469 - 0.1551353i	-0.102469 - 0.1551353i	5.551e-17 - 2.77555e-16i
0.136218 - 0.062547i	-0.087235 - 0.1751370i	-0.087235 - 0.1751370i	-2.77e-17 - 8.32667e-17i
0.059643 - 0.1315135i	-0.133160 + 0.061184i	-0.133160 + 0.0611847i	-5.55e-17 + 5.55111e-17i
-0.240909 - 0.0688317i	0.019842 - 0.3689402i	0.0198420 - 0.3689402i	-2.77e-17 - 5.55111e-17i
-0.042361 + 0.0937527i	-0.86698+ 0.34960764i	-0.866986 + 0.3496076i	0.000000 + 1.66533e-16i
0.139309+ 0.164322i	-0.175472- 0.9509377i	-0.175472 - 0.9509377i	0.000000+ 0.00000000i
0.062154+ 0.0643163i	0.6231356 + 0.416981i	0.6231356 + 0.416981i	1.11022e-16 - 1.665e-16i
0.1211082 - 0.0129757i	-0.405597 - 1.987476i	-0.405597 - 1.987476i	0.00000000 + 0.000000i
0.0894122 + 0.0498223i	-0.063684 + 0.1339475i	-0.063684 + 0.133947i	5.551e-17 - 2.77555e-17i
-0.123143+ 0.070999i	0.0249657 + 1.105588i	0.0249657 + 1.105588i	1.665e-16 + 0.0000000i
-0.044001 - 0.0213394i	-0.135247 - 0.341489i	-0.135245 - 0.3414890i	0.0000000 + 5.5511e-17i
0.003442 - 0.0585115i	1.485249 + 0.599726i	1.485249 + 0.5997266i	2.2204e-16 - 1.1102e-16i
-0.22719 + 0.000250i	-0.01908 + 0.561894i	-0.019082 + 0.5618942i	4.1633e-17 + 0.0000000i
-0.143094 - 0.0301067i	-0.175511 - 0.128807i	-0.175511 - 0.1288075i	0.0000000 + 2.7755e-17i
0.1051824 - 0.0095514i	0.0593235 + 0.0350709i	0.0593235 + 0.0350709i	1.249e-16 + 1.3877e-16i

➔ From above table, there's approximately no error between FFT resulted from Butterfly and Simulated, working with Floating Point Modelling.

Design Assumptions

1) All inputs come in parallel and in order shown in the butterfly diagram.
2) Inputs could be real numbers or even complex.
3) Inputs are coming from 8-bit ADC that each input is 8-bit real and 8-bit imaginary.
4) Inputs are normalized to have MSB as sign bit and second one is integer even 0 or 1, and the rest are fixed point bits.
5) All stages produce values less than 2 and more than -2, unless Overflow occurs.
6) Assume that inputs come after asynchronous reset and before start = 1

Design Procedure

The Design requires to produce 50 Msps, which can be divided into five stages, each stage can produce its output in 10 Msps, and as all stages use same hardware and go faster than the algorithm requirement, hence the ***Time Sharing*** can be used here between all stages.

Each stage has 16 MAC takes 32 inputs and 32 outputs, then the usage of 16 parallel MAC units provides that Design is using the ***Parallelism.***

The timing adjustment between the stages timing and the input/output timing will require ***Pipelining,*** as the first stage takes the inputs and provides its output to the next one and so on until we have the final output after the five stages has finished.

Basic Processing Unit (Multiply and Accumulative)

The system is depending on the idea of that each stage has 32 inputs and 32 outputs distributed on 16 nodes, these nodes are Complex multiply and accumulative which the basic block of the design, which is has the following block diagram:

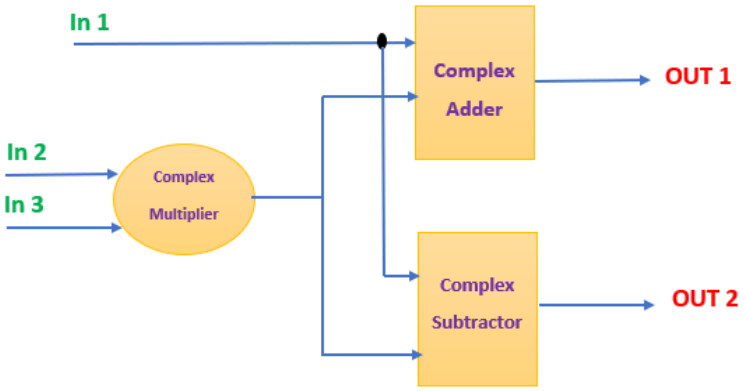

Figure 2: Multiply and Accumulate Block

That is governed by the following equations:

$$out1 = in1 + in2 * in3, \quad out2 = in1 - in2 * in3$$

$$: in1 \ and \ in2 \ are \ complex \ numbers \ and \ in3 = W_{32}^{k}$$

And then the complex addition, subtraction and Multiplication can be implemented by the following equations:

$$Real_{add} = Real1 + Real2. \quad Img_{add} = Img1 + Img2.$$

$$Real_{sub} = Real1 - Real2. \quad Img_{sub} = Img1 - Img2.$$

Which can be expressed by the following block design:

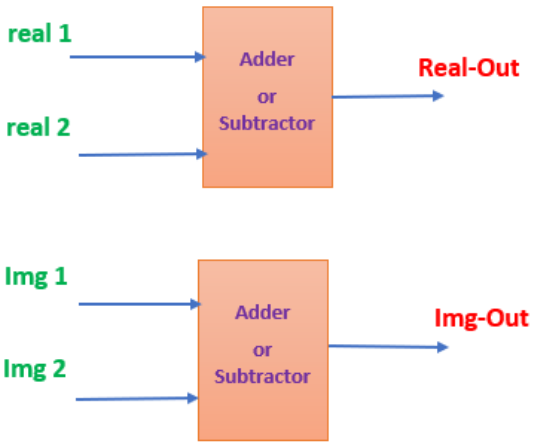

Figure 3: Complex Adder and Subtractor

And for Complex Multiplier is represented by:

$$Real_{mul} = Real1 * Real2 - Img1 * Img2.$$

$$Img_{mul} = Real1 * Img2 + Real2 * Img1.$$

Which is represented by the following block design:

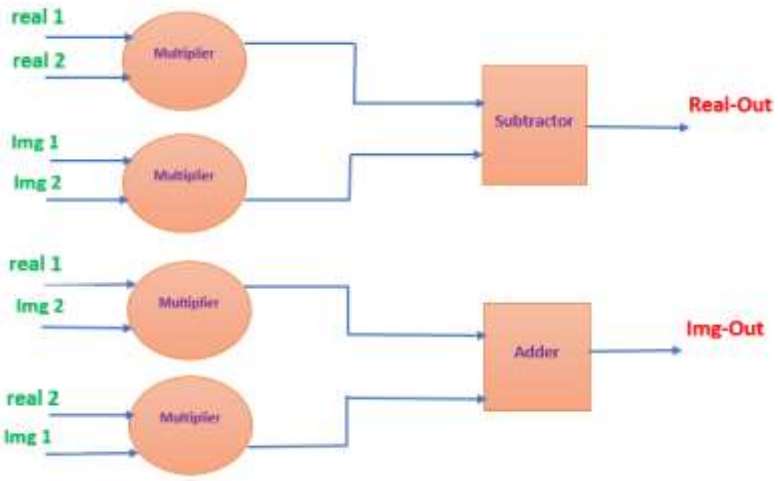

Figure 4: Complex Multiplier

➔ MAC feedback with Overflow flag in case overflow occurs in any adder, subtractor or multiplier.

RTL System Design

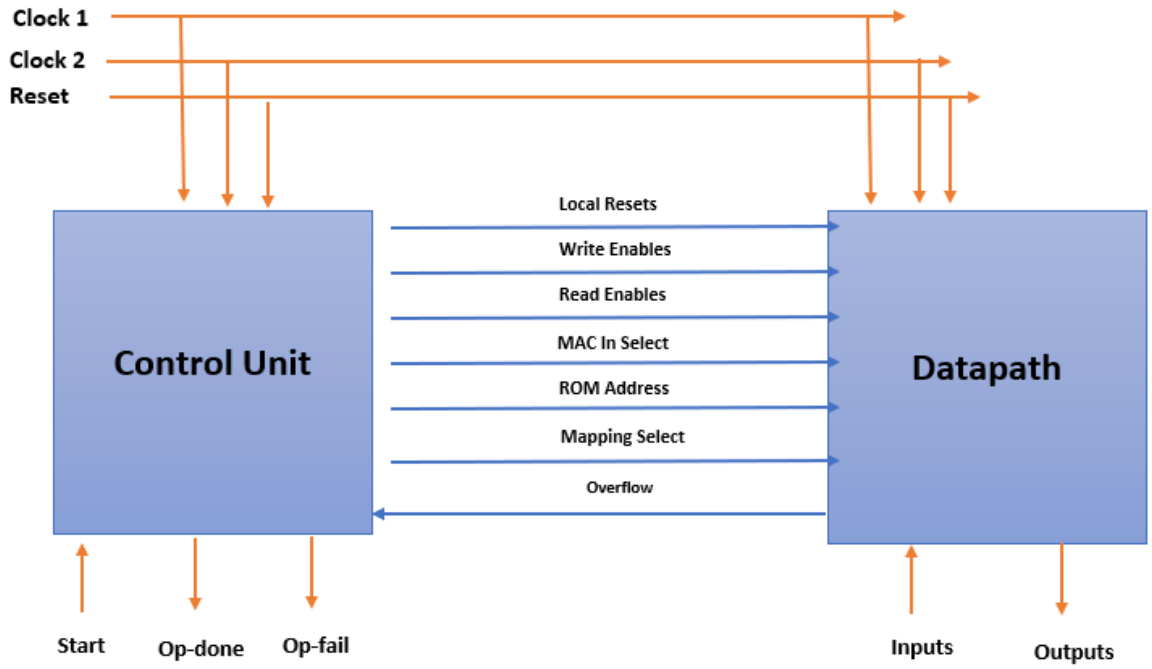

Figure 5: Control Unit and Datapath Communication

Datapath Design

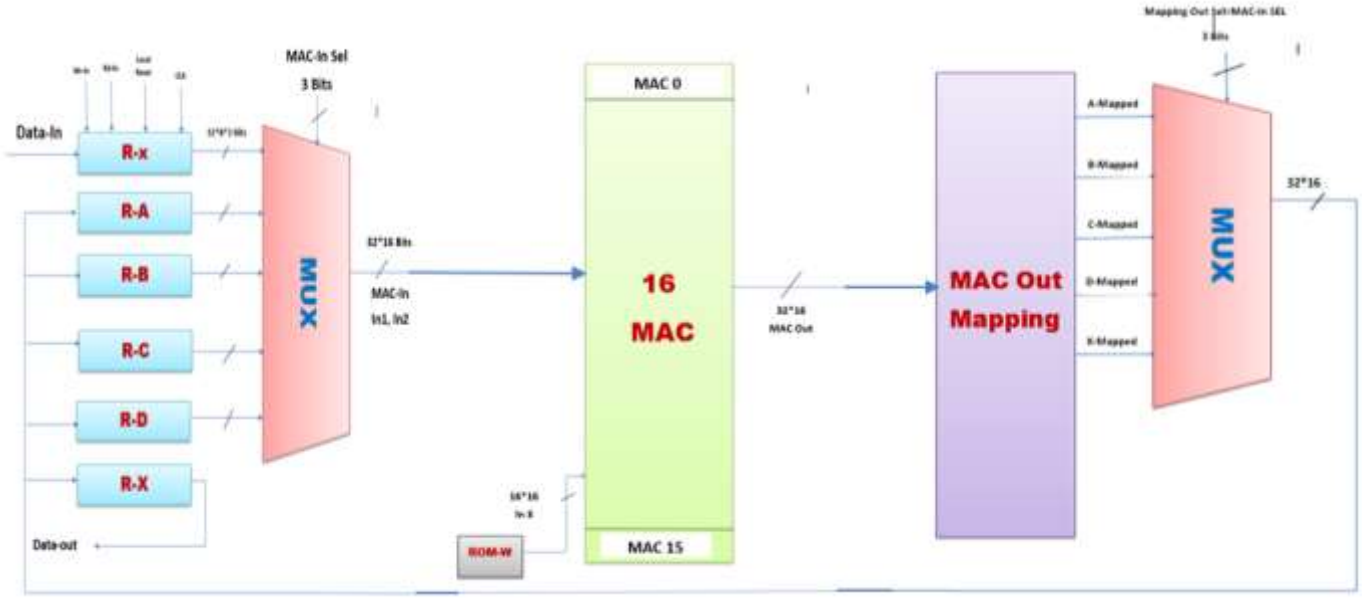

Figure 6: Datapath Block Design

Register File:

It contains six registers (each has 32 inputs each has 16 bit (8 real + 8 imaginary) = 512 bits concatenated), **Input and Output registers** operate with clock = 10 MHz, and the ***Pipelining* Registers** which stores the outputs of intermediate stages operate with clock = 50 MHz, and that in order to satisfy ***Time Sharing*** between stages.

Each register has local reset, write and read enable, which is controlled by Control unit and determined due to the state of operation.

The outputs of Registers are connected to a **Multiplexer,** which selects which register will feed its input to the circuit due to the stage it works on.

ROM of Weights:

From the butterfly diagram, the second input is multiplied to a weight W_{32}^{k} before going through MAC unit, these weights are known and fixed for all stages, so it can be stored on Read-only Memory, which has 5 lines each line has the weights of each stage concatenated (16 weight each is 16-bit = 256 bits) with respect to the order inputs come in. The ROM has Read address which is selected by the controller due to which stage is processed.

MAC Array:

It's 16 MAC concatenated from MAC0 to MAC15, and that because each stage requires 16 MAC unit to process its inputs to outputs ***"Parallelism"*** and that MAC array inputs are fed from the Register File (which feeds with input 1 and input 2) and ROM (which feeds with weights).

And as MAC unit internal Design is shown, it will produce 32 outputs each is 16-bit as an output to the next stage.

MAC Output Mapping Circuit:

As shown in the butterfly diagram, the outputs of each stage are coming in order, but it has to be mapped to another order to be beneficial for the next stage, for example, in 1st stage, inputs are feed to the MAC and results in output in order from 0 to 32, but the next stage needs its input in order that is shown in diagram, for example it needs to take $A_0$and A_2in order to produce B_0, so before storing the output of MAC into a register to be fed to next stage, it has to be mapped to be appropriate for it.

That what happens in Mapping circuit which is nothing but rotating MAC outputs to get them in order which is suitable for next stage.

Mapping Circuits takes the MAC output and Map it by the possible orders of all stages and feed that to Multiplexer which selects which order will be stored according to the feeding stage and next one.

That is fed back to the register file again, to work on the next stage and so on until it has the final output.

Control Unit Design

Control unit is a hardwired Finite State Machine which implement a sequence of operations which is described by the following state table:

Current state	Next state	Control signals
Load x	Load A if start= 1, overflow=0	All
Load A	Load B if overflow=0	All
Load B	Load C if overflow=0	All
Load C	Load D if overflow=0	All
Load D	Load X if overflow=0	All
Load A,B,C,D,X	Load x if overflow=1	All

And that can be represented by the following finite state machine:

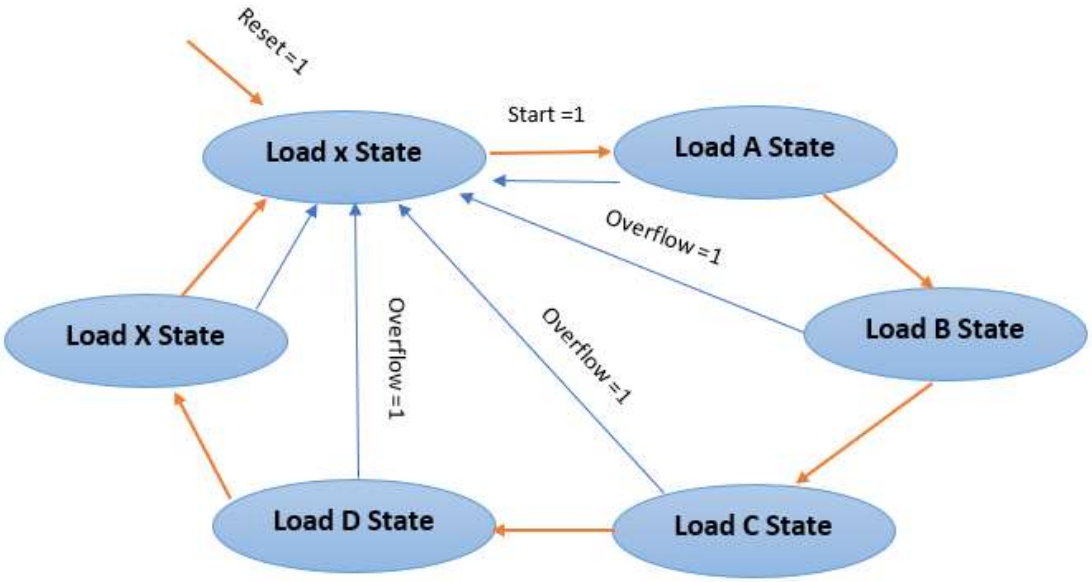

Figure 7: Finite State Machine

Which outs the control signals according to the table as follows:

State	Write Enables	Read Enables	MAC In Select	ROMW Address	Mapping Select
Load x	"100000"	"000000"	"000"	"000"	"000"
Load A	"010000"	"000000"	"000"	"001"	"001"
Load B	"001000"	"010000"	"001"	"010"	"010"
Load C	"000100"	"001000"	"010"	"011"	"011"
Load D	"000010"	"000100"	"011"	"100"	"100"
Load X	"000001"	"000010"	"100"	"101"	"101"

➔ Control outs end of operation flag to inform that its open for another operation.

➔ In case of overflow occurs then it informs fail of operation.

Behavioral Simulation and Testing

Here's the sequence of operations from applying the inputs to getting the outputs showing what's happening in the intermediate stages:

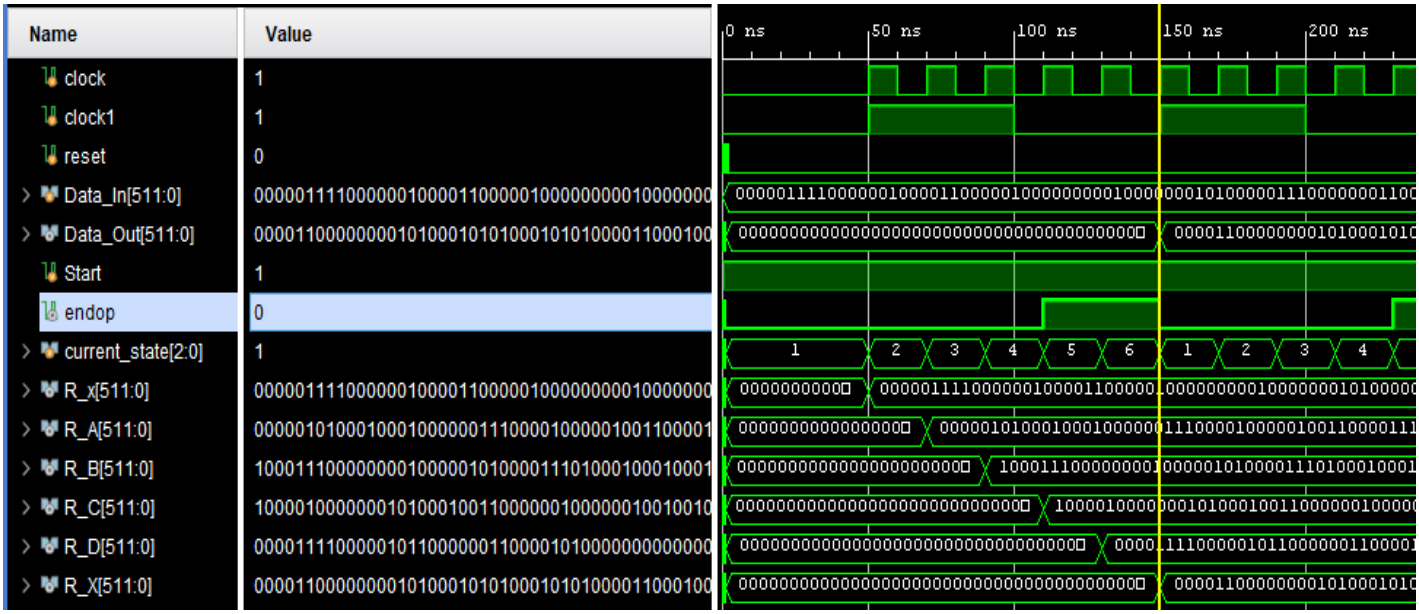

Figure 8: Waveform of the whole system

Here, there are two clocks, as $f_{clk} = 5\ f_{clk1}$, and the input comes synchronized with clock 1, and processed in MAC to get the output stored in the intermediate registers which is synchronized with clock, to finish the five stages and finally get the output at the next edge of clock.

By taking these results to compare with simulated results gotten from MATLAB, the follow is obtained:

x	$FFT(x)$ *by MATLAB*	$FFT(x)$ *by Hardware*	*Error*
0.014455 - 0.126667i	0.441757 + 0.213603i	0.47619 + 0.285714i	-0.03443 - 0.072110i
0.125529 + 0.039827i	-0.15530 - 1.009028i	-0.19047 - 0.968253i	0.0351741 - 0.040774i
0.028245 - 0.019142i	-0.31133 - 0.520417i	-0.30158- 0.6031746i	-0.0097475 + 0.082757i
0.047868 - 0.217864i	0.953586 - 1.754237i	1.000 - 1.73015i	-0.0464136 - 0.0240783i
0.155870 - 0.113421i	-1.022721 + 0.3753358i	-1.14285 + 0.47619i	0.120135- 0.10085458i
0.011343 + 0.177033i	0.5881526 +0.57917327i	0.5396825 + 0.5396825i	0.048470+ 0.039490i
-0.056715 + 0.1985140i	0.1543201 + 0.3982074i	0.238095 + 0.23809523i	-0.08377 + 0.160112i
0.1070853 + 0.0131092i	0.7600733 - 0.300173i	0.6190476 + 0.396825i	0.141025 - 0.696998i
0.0321626 - 0.0538364i	0.5939489 + 0.5159849i	0.57142 + 0.71428571i	0.022520- 0.198300i
-0.160227- 0.0034463i	1.4053853 - 1.2194050i	1.50793 - 1.1904761i	-0.1025511 - 0.02892i
0.0298709- 0.0606668i	-1.0770367 + 0.530246i	-1.0000 + 0.4761904i	-0.077036+ 0.05405582i
0.2482999 - 0.0982133i	-0.793085 - 0.2970659i	-0.7936507 - 0.25396i	0.00056507 - 0.043097i
0.1317472 + 0.0081936i	-0.379705 - 0.1509708i	-0.4603174 + 0.0952380i	0.080612 - 0.2462089i
-0.056980 + 0.0239011i	-0.261978 - 0.1403939i	-0.22222 - 0.1587301i	-0.0397563 + 0.018336i
0.0211006+ 0.1230195i	-0.198495- 0.1400129i	-0.126984- 0.41269841i	-0.071511 + 0.272685i
0.1816322 + 0.2414530i	-0.15682 - 0.142742i	-0.222222 - 0.19047619i	0.0654016 + 0.0477341i
-0.091909 + 0.0746960i	-0.126239 - 0.1475279i	-0.09523 + 0.063492i	-0.0310017 - 0.211020i
-0.223389 - 0.040971i	-0.102469 - 0.1551353i	-0.12698 - 0.2380952i	0.024515 + 0.08295i

0.136218 - 0.062547i	-0.087235 - 0.1751370i	-0.20634 - 0.3492063i	0.11911 + 0.1740693i
0.059643 - 0.1315135i	-0.133160 + 0.061184i	-0.174603 + 0.095238i	0.04144254 - 0.034053i
-0.240909 - 0.0688317i	0.019842 - 0.3689402i	0.1111111- 0.1587301i	-0.091269 - 0.2102101i
-0.042361 + 0.0937527i	-0.86698+ 0.34960764i	-0.85714 + 0.3174603i	-0.009844 + 0.0321473i
0.139309+ 0.164322i	-0.175472- 0.9509377i	-0.3015873- 1.1111i	0.1261 + 0.16017i
0.062154+ 0.0643163i	0.6231356 + 0.416981i	0.74603 - 0.30158i	-0.1228 + 0.71856i
0.1211082 - 0.0129757i	-0.405597 - 1.987476i	-0.3174603 - 1.9365i	-0.08813 - 0.05096i
0.0894122 + 0.0498223i	-0.063684 + 0.1339475i	-0.063492 + 0.12698i	-0.00019207 + 0.00696i
-0.123143+ 0.070999i	0.0249657 + 1.105588i	-0.142857 + 0.98412i	0.16782 + 0.1214613i
-0.044001 - 0.0213394i	-0.135247 - 0.341489i	-0.095238 - 0.31746i	-0.0400076 - 0.0240287i
0.003442 - 0.0585115i	1.485249 + 0.599726i	1.61904+ 0.730158i	-0.13379 - 0.130432i
-0.22719 + 0.000250i	-0.01908 + 0.561894i	-0.095238 + 0.57142i	0.07615 - 0.00953i
-0.143094 - 0.0301067i	-0.175511 - 0.128807i	-0.19047 - 0.2222222i	0.0149648 + 0.093414i
0.1051824 - 0.0095514i	0.0593235 + 0.0350709i	0.190476 + 0.031746i	-0.13115+ 0.003324i

➔ The Significant error shown above is due to low resolution resulted from only 6-bit fixed point, but still acceptable with respect to the application, so higher number of bits used in fixed point, higher resolution and lower error is obtained.

Synthesis

By making logic synthesis on FPGA platform, Setup and Hold timing report is obtained as follows:

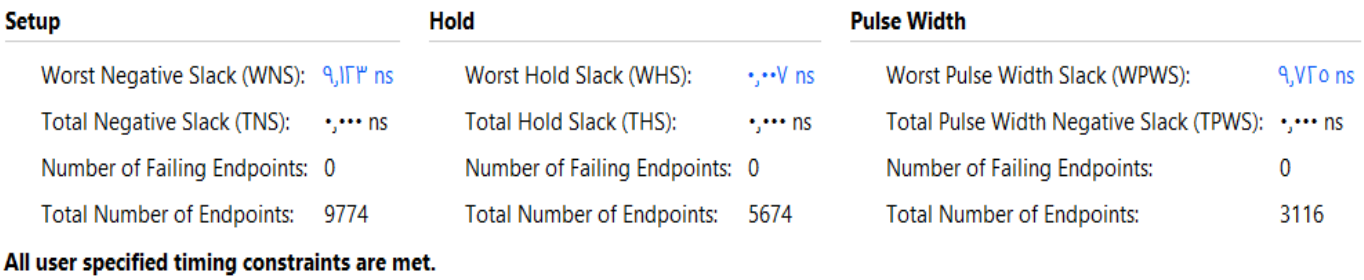

Figure 9: Setup and Hold Timing Constraints

And the critical paths delay is obtained as follows:

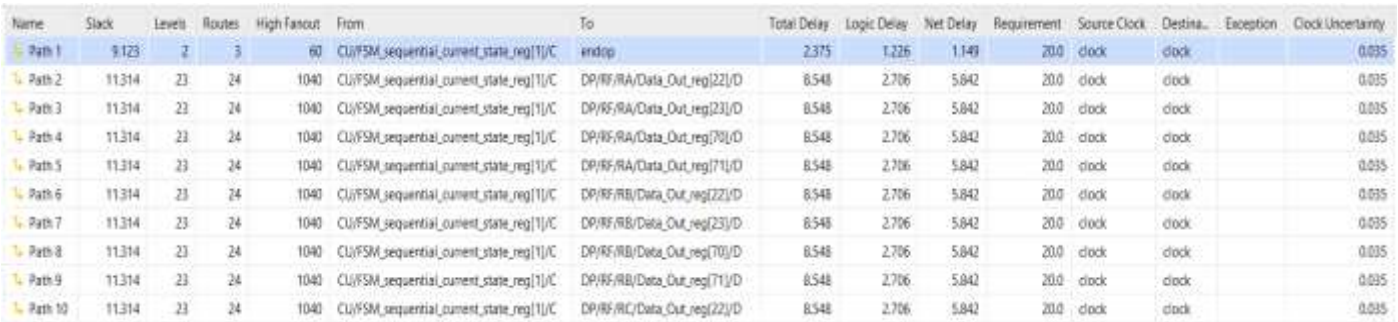

Figure 10: Setup time Slacks

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 11	0.007	1	2	30	CU/FSM_sequential_current_state_reg[0]/C	CU/FSM_sequential_current_state_reg[1]/D	0.196	0.106	0.090	0.0	clock	clock
Path 12	0.007	1	2	30	CU/FSM_sequential_current_state_reg[0]/C	CU/FSM_sequential_current_state_reg[2]/D	0.196	0.106	0.090	0.0	clock	clock
Path 13	0.008	1	2	30	CU/FSM_sequential_current_state_reg[0]/C	CU/FSM_sequential_current_state_reg[0]/D	0.197	0.103	0.094	0.0	clock	clock
Path 14	0.011	1	2	60	CU/FSM_sequential_current_state_reg[2]/C	DP/ROMW/W_out_reg[206]/D	0.200	0.103	0.097	0.0	clock	clock
Path 15	0.011	1	2	60	CU/FSM_sequential_current_state_reg[1]/C	DP/ROMW/W_out_reg[244]/D	0.200	0.103	0.097	0.0	clock	clock
Path 16	0.011	1	2	60	CU/FSM_sequential_current_state_reg[1]/C	DP/ROMW/W_out_reg[244]_rep/D	0.200	0.103	0.097	0.0	clock	clock
Path 17	0.011	1	2	60	CU/FSM_sequential_current_state_reg[1]/C	DP/ROMW/W_out_reg[244]_rep__0/D	0.200	0.103	0.097	0.0	clock	clock
Path 18	0.011	1	2	60	CU/FSM_sequential_current_state_reg[1]/C	DP/ROMW/W_out_reg[244]_rep__1/D	0.200	0.103	0.097	0.0	clock	clock
Path 19	0.011	1	2	60	CU/FSM_sequential_current_state_reg[1]/C	DP/ROMW/W_out_reg[244]_rep__2/D	0.200	0.103	0.097	0.0	clock	clock
Path 20	0.011	1	2	60	CU/FSM_sequential_current_state_reg[1]/C	DP/ROMW/W_out_reg[244]_rep__3/D	0.200	0.103	0.097	0.0	clock	clock

Figure 11: Hold time Slacks

➔ That Timing Reports show that there's no negative slacks for the setup and hold time, and that fulfills the constraints of design

The Utilization report can be obtained as follows:

Site Type	Used	Fixed	Available	Util%
CLB LUTs*	7447	0	230400	3.23
LUT as Logic	7447	0	230400	3.23
LUT as Memory	0	0	101760	0.00
CLB Registers	3114	0	460800	0.68
Register as Flip Flop	3114	0	460800	0.68
Register as Latch	0	0	460800	0.00
CARRY8	421	0	28800	1.46
F7 Muxes	0	0	115200	0.00
F8 Muxes	0	0	57600	0.00
F9 Muxes	0	0	28800	0.00

Figure 12: Utilization Report

Post Synthesis Simulation

Although the code doesn't have any warning or implicit latches, the post synthesis simulation has the following result:

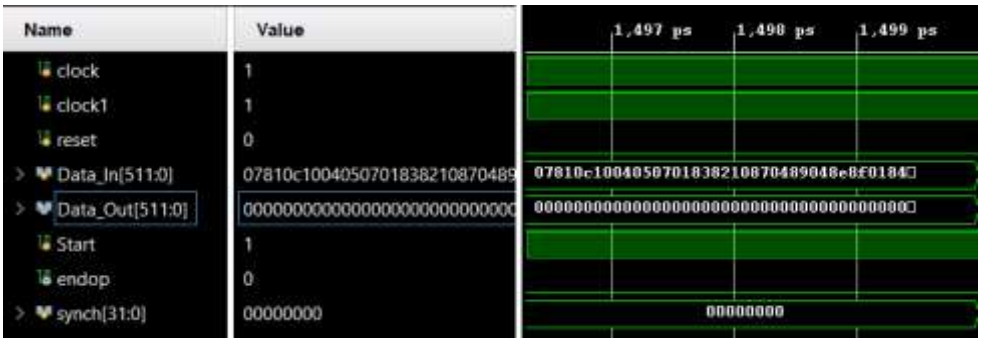

Figure 13: Post Synthesis Simulation result

Which indicates that the registers don't sample data in aright manner due to timing problems which does not match the correct behavioral simulation.