Cairo University Faculty of Engineering

Dept. of Electronics & Electrical Communications

Fourth Year – Mainstream

VLSI 2 - Digital Design

32-Point FFT Using Cooley-Tukey Project

Team 6

Name	Sec	BN	ID
جمال اسماعيل السيد ابوالعز	2	3	9210319
جينا فريد فرج رياض	2	5	9210324
فتحى مصطفى فتحى عبد الحميد	3	15	9210808
محمد هشام السعيد عبدالعزيز	4	5	9211052
محمد هشام محمد رفعت محمد صالح	4	7	9211057
محمود عبدالحليم احمد عيسى	4	12	9211092

Table of Contents

List of Figures

I. System Design

1. Algorithm Description and Adaptation Flow

The Fast Fourier Transform (FFT) is a fundamental algorithm in digital signal processing, widely used for efficiently computing the Discrete Fourier Transform (DFT). Its primary purpose is to transform a signal from the time or spatial domain into the frequency domain, revealing the constituent frequencies present in the signal. This transformation is crucial for various applications, including telecommunications, audio processing, image analysis, and scientific computing, enabling tasks such as spectrum analysis, filtering, and data compression.

Our specific implementation focuses on a 32-point FFT, utilizing the Decimation-In-Time (DIT) algorithm. This means the algorithm processes input data blocks of 32 samples by recursively breaking down the DFT into smaller DFTs. The structure of a 32-point DIT FFT involves log2(32) = 5 distinct stages of computation. Each stage consists of a series of "butterfly" operations, which are the core computational units of the FFT. Adapting such an algorithm for hardware implementation requires a systematic approach that bridges the gap between the mathematical description and the physical circuit design, differing significantly from traditional hardware design flows that typically start from a block diagram or RTL description. This adaptation involves several key steps, including translating the algorithm into a hardware-friendly block diagram, analyzing fixed-point precision requirements, making architectural decisions regarding parallelism and time-sharing, and designing the necessary control logic and memory structures.

2. Processing Units

The fundamental processing unit (PU) for the FFT is the butterfly unit. This unit performs the core arithmetic operations required at each node of the FFT algorithm. A butterfly unit typically accepts two complex data inputs, say A and B, and a complex twiddle factor, WNk. It computes two complex outputs: one is the sum of A and the product of B and the twiddle factor ($A+B\cdot WNk$), and the other is the difference ($A-B\cdot WNk$).

We used a single stage with time-sharing. Instead of instantiating the many butterfly units required for a fully parallel implementation (which would be 80 butterfly units for a 32-point FFT), we utilize a single stage of PUs. Since each stage of a 32-point radix-2 FFT involves 16 butterfly operations, our design uses 16 physical butterfly units. These 16 PUs are reused over 5 clock cycles to complete the computations for all five stages of the 32-point FFT. This approach significantly reduces the required hardware area compared to direct implementation.

3. Fixed-Point Analysis

For this 32-point FFT implementation, the time-domain **input** samples are assumed to be represented in a fixed-point format denoted as **S(7,0),** totaling 8 bits **(1 sign bit + 7 integer bits).**

We evaluated different fixed-point formats for the complex twiddle factors through simulation, using SQNR to assess performance.

The table below summarizes the SQNR results obtained for various fixed-point representations of the twiddle factors. The format is described by the number of bits used for the integer and fractional parts, excluding the sign bit.

Integer Part	Fraction Part	SQNR (dB)
2	6	44.628
1	6	44.653
0	6	28.900
2	5	37.959
1	5	37.945
1	4	36.405

Analyzing these results, we observe that increasing the number of fractional bits generally improves SQNR. The representation with an **Integer Part of 1 and a Fraction Part of 6** (total 8 bits) provides a good balance, offering high SQNR (44.653 dB) with a reasonable number of bits.

Beyond the twiddle factors, the fixed-point format of the butterfly output also significantly impacts the overall FFT accuracy. We performed simulations to evaluate the SQNR of the butterfly output for various fixed-point formats. The results are summarized in the 2-dimensional table below, and the cells containing the corresponding SQNR values in dB.

Fraction Part Integer Part	0	2	4	6	8	10
11	34.139	-	34.208	-	-	-
12	44.196	-	-	-	-	-
13	44.224	-	44.678	44.671	44.629	44.647
15	-	44.683	44.660	-	-	-
17	44.237	-	-	-	-	-

Based on these simulation results and considering the balance between achievable SQNR and hardware complexity, the **fixed-point format S(12,0) was chosen for the butterfly output**. This format provides an **SQNR of 44.196 dB**, which was deemed acceptable for the overall FFT performance requirements while utilizing a relatively straightforward fixed-point representation with no fractional bits. **Although formats with fractional bits or larger integer parts showed slightly higher SQNRs**, the S(12,0) format offered a good compromise in terms of implementation cost and required precision.

II. FFT TOP Level Architecture.

The hardware which used in this approach is a Butterfly unit contains 16 MAC's, 3 register Banks, two for IOs and one for registering the output for next stage, a MUX selection that chooses which inputs will be loaded this cycle and a Control unit to choose a selection lines of the MUXs.

All registers Operates at the same clock 100 MHZ but for the **Input and output stages** which is assumed that the inputs are ready every 20MHZ, **We add a control signal which enable the IO registers every 5 Cycles.** This will **reduce the Complexity of the design** as we will not need a **clock divider or other input clock** which could be challenging.

We used time sharing concept, so we used only 16 MAC unit to calculate the 5 stages of the FFT.

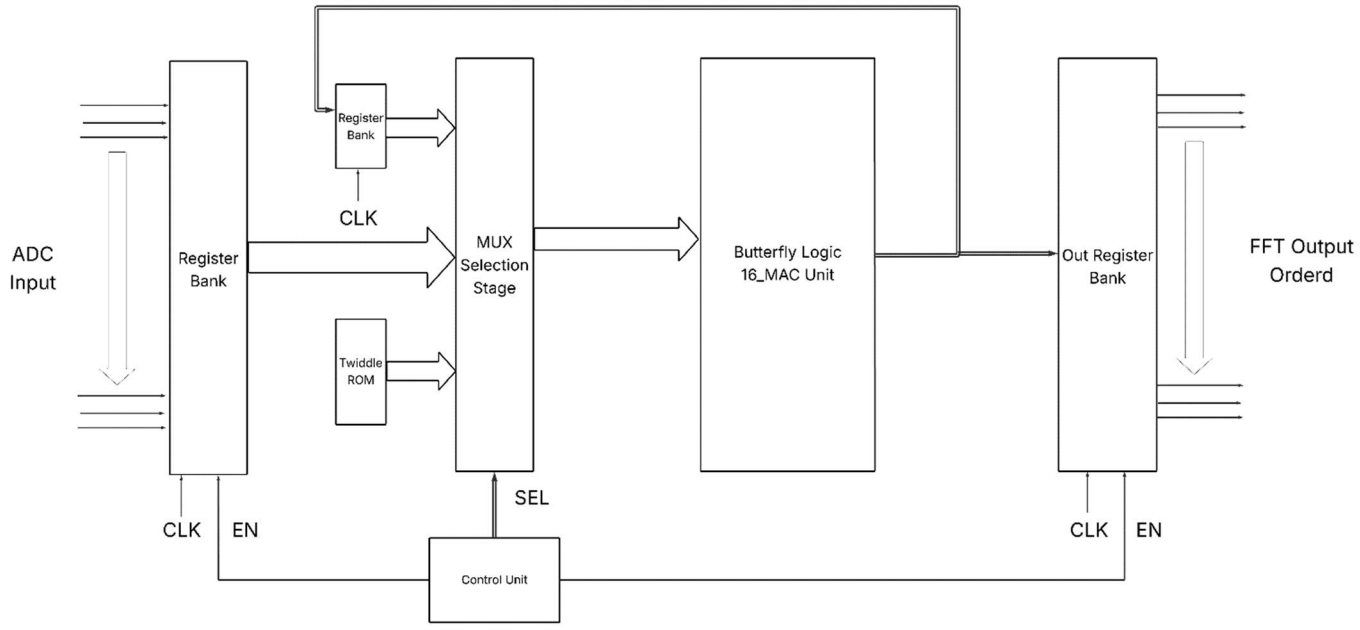


Figure 1: Top Level Architecture

1. Modules Architecture.

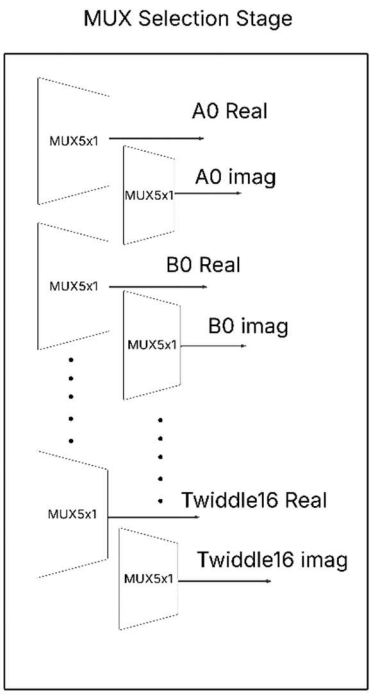


Figure 3: MUX selection Stage

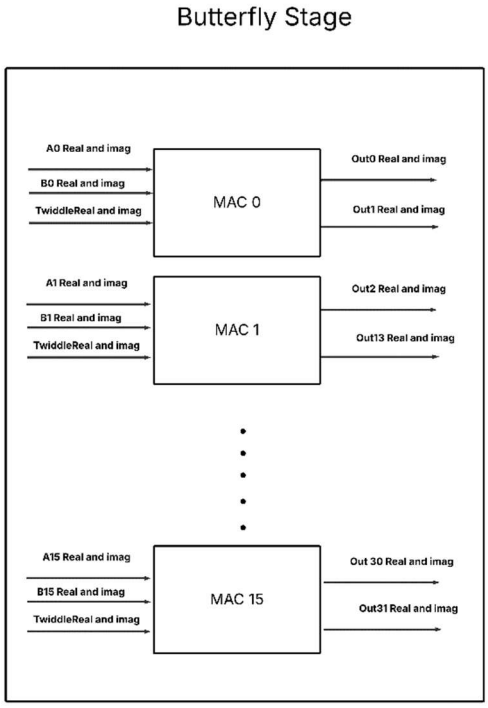


Figure 2: Butterfly Stage

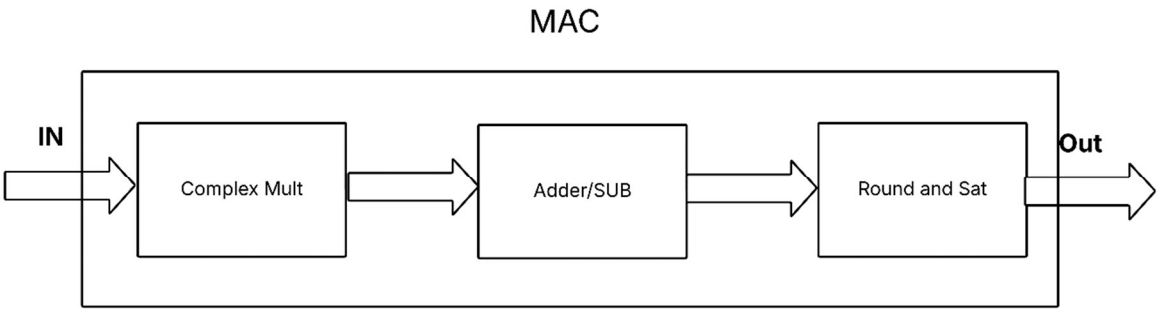


Figure 4: MAC Unit

- MAC Unit Consist of **Complex MULT** then **ADD/SUB** then **round and sat** all of them is combinational logic with its output being register before the MUX stage.
- MUX Stage is reasonable choosing which inputs will be loaded this cycle, we used 16*6 (16 for MAC and 6 for real and imag A, B, Twiddle inputs) **5x1 MUXs**.
- Butterfly stage contains the **16 MAC** unit.
- Control unit is a counter which indicates the selection line of the MUX stage unit.

2. Design Synchronization.

The core of the design operates synchronously with a 100 MHz clock. The input data is available every 20 MHz cycle, and output data is required at the same rate. This means the IO stages need to operate at a **rate five times slower than the main clock**.

Approach 1: Using a **Clock Divider**
A traditional approach to handling different clock rates within a design is to use a clock divider. A clock divider would take the 100 MHz main clock and generate a new clock signal operating at 20 MHz (100 MHz / 5 = 20 MHz). The IO registers would then be clocked by this slower 20 MHz signal
While this approach directly provides a clock signal at the required frequency, **it introduces a second clock domain into the design.**

Approach 2: Using a **Enable Signal**
The proposed alternative approach is to keep all registers, including the IO registers, operating on the same 100 MHz main clock. A control signal, enable, is generated which is active for one cycle every five cycles of the 100 MHz clock. The IO registers will only capture new data or update their outputs when the enable signal is active on the rising edge of the 100 MHz clock.

Advantages of the Enable Approach:
Using an enable signal for the IO stages offers significant advantages over using a clock divider and a separate clock domain:

3. Feature.

Our design is parametrized for fixed point representation for twiddle and output.

```
module FFT_TOP
    #(
        parameter inADC = 8,
        parameter inWordWidth_1 = 13, // butterfly input
        parameter inWordWidth_2 = 8,  // twiddle input
        parameter  I1 = 16,           // butterfly output before round
        parameter  F1 = 6,
        parameter  I2 = 12,           // butterfly output
        parameter  F2 = 0
    )
```

Figure 5: Top design Parameter

```
module round_and_sat #(
    parameter int I1 = 16,  // Input integer bits (not including sign)
    parameter int F1 = 6,   // Input fractional bits
    parameter int I2 = 12,  // Output integer bits (not including sign)
    parameter int F2 = 0    // Output fractional bits
) (
```

Figure 6: Round and Sat parameter

4. MAC Operation.

Our Multiplier is designed to reduce the number of complex **multiplications from 4 multiplications to 3**. This can result in more efficient hardware implementation, potentially saving on area, power consumption.

5. Multiplication Flow.

1. $Z1 \times Z2 = (a+ib)(c+id) = (ac-bd)+i(ad+bc)$
2. ab = in1_real + in1_imag // a+b
3. dc = in2_imag - in2_real // d−c
4. cd = in2_real + in2_imag // c+d
5. k1 = in2_real * ab // c×(a+b)
6. k2 = in1_real * dc // a×(d−c
7. k3 = in1_imag * cd // b×(c+d)

out_real = k1 - k3 // c(a+b)−b(c+d)=ac+bc−bc−bd=ac−bd

out_imag = k1 + k2 // c(a+b)+a(d−c)=ac+bc+ad−ac=bc+ad

6. Fixed Point calculation.

- Input ADC 8 bit extended to 13 bit as we assume that it's integer.
- Twiddle S(1,6) : 1 sign , 1 int , 6 floating
- Complex Multiplication: Twiddle S(1,6) * In S(12,0) + 2Int bit from the addition so the multiplication out is S(15,6).
- ADDER/SUB : After the multiplication we add S(15,6) + (12,0) = S(16,6)
- Round and Sat: We round and sat the MAC out again from S(16,6) to S(12,0) to be ready for next stage.

III. Pre- synthesis Simulation.

We simulate the design by loading random generated **input from the python model and load the expected output** to the Test bench and check the equivalence between the module output and expected output.

We start the test case by loading the input generated from the python model which is **random input** and also the expected output from the model and then we collect the output of the RTL module and then check for any mismatch.
We add an **assert error** so any error in the simulated Results will appear as an error in the end of the simulation.

```
for(int i = 0 ; i < 32 ; i++)begin
  CHECK_real: assert (abs(OUTPUT_sim[i][0] - OUTPUT[i][0]) <= 1 )
    else $error("Assertion CHECK_Real failed! , Expected = %d  , output = %d " ,OUTPUT[i][0] , OUTPUT_sim[i][0] );
  CHECK_imag: assert (abs(OUTPUT_sim[i][1] - OUTPUT[i][1]) <= 1 )
    else $error("Assertion CHECK_imag failed! , Expected = %d  , output = %d " ,OUTPUT[i][1] , OUTPUT_sim[i][1] );
end

#1000 $stop;
```

Figure 7: TB Assert error

```
PS D:\EECE 4Y\First term\Second Term\digital\final\RTL>
vlog *.v *.sv
QuestaSim-64 vlog 2024.1 Compiler 2024.02 Feb  1 2024
Start time: 20:10:45 on May 09,2025
vlog *.v *.sv
-- Compiling module Butterfly
-- Compiling module complex_mult
-- Compiling module control_unit
-- Compiling module MAC
-- Compiling module MUX5x1
-- Compiling module SELECT_STAGE
-- Compiling module Register
-- Compiling module FFT_TOP
-- Compiling module ROM_W_Imag
-- Compiling module W_SELECT_STAGE
-- Compiling module ROM_W_Real
-- Compiling module round_and_sat
-- Compiling module FFT_tb

Top level modules:
       FFT_tb
End time: 20:10:45 on May 09,2025, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
PS D:\EECE 4Y\First term\Second Term\digital\final\RTL> vsim -voptargs=+acc work.FFT_tb -c
Reading pref.tcl
```

Figure 8: Questa Compilation

```
VSIM 1> run -all
# ** Note: $stop    : fft32_tb.sv(264)
#    Time: 1080 ns  Iteration: 0  Instance: /FFT_tb
# Break in Module FFT_tb at fft32_tb.sv line 264
# Stopped at fft32_tb.sv line 264
VSIM 2> q
# End time: 20:11:09 on May 09,2025, Elapsed time: 0:00:18
# Errors: 0, Warnings: 4
PS D:\EECE 4Y\First term\Second Term\digital\final\RTL>
```

Figure 9: Simulation Ends without any Errors

Simulation Ended without any errors.

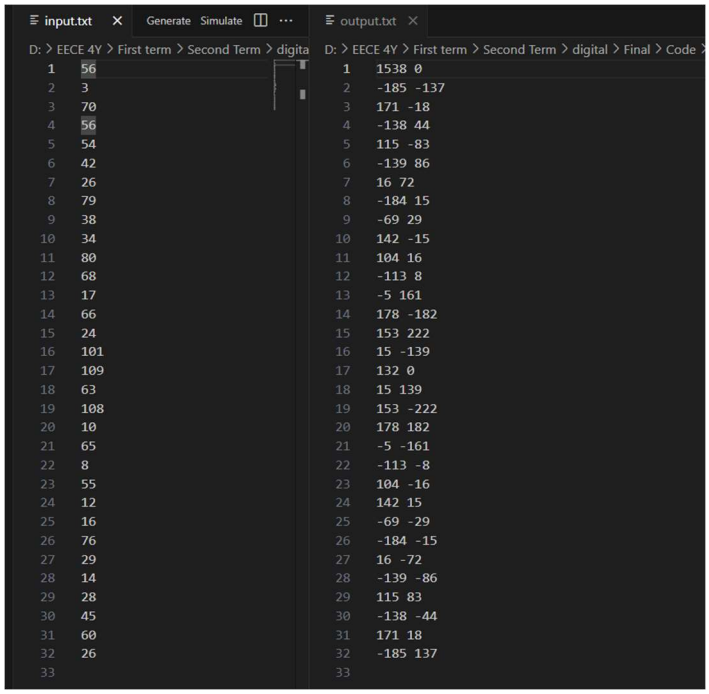


Figure 10: Python model input and expected output

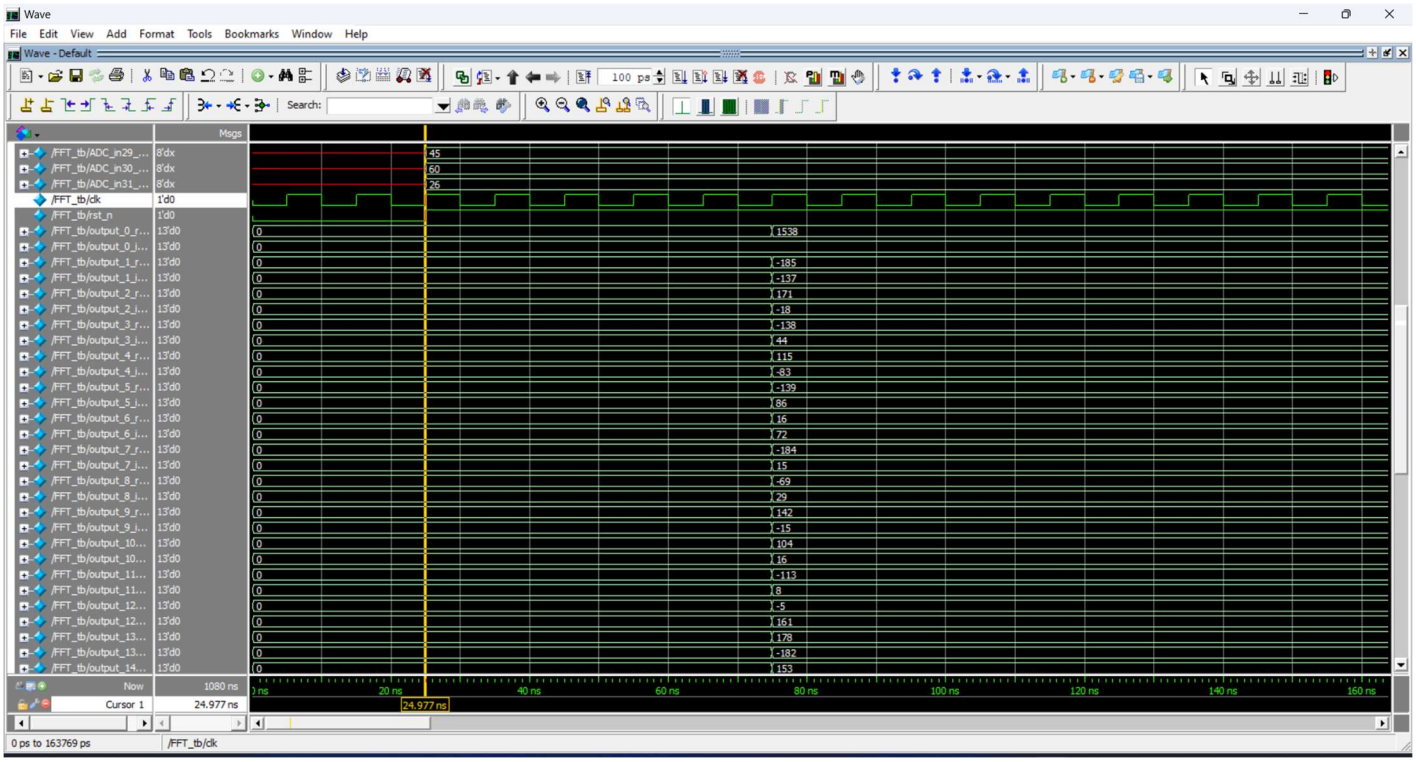


Figure 11: Simulation Wave form

We can see that the output is generated after 5 cycles from the 100 MHZ clock as expected.

1. Common test case.

For more verification we apply a pulse input and check if the output is rectangle as expected.

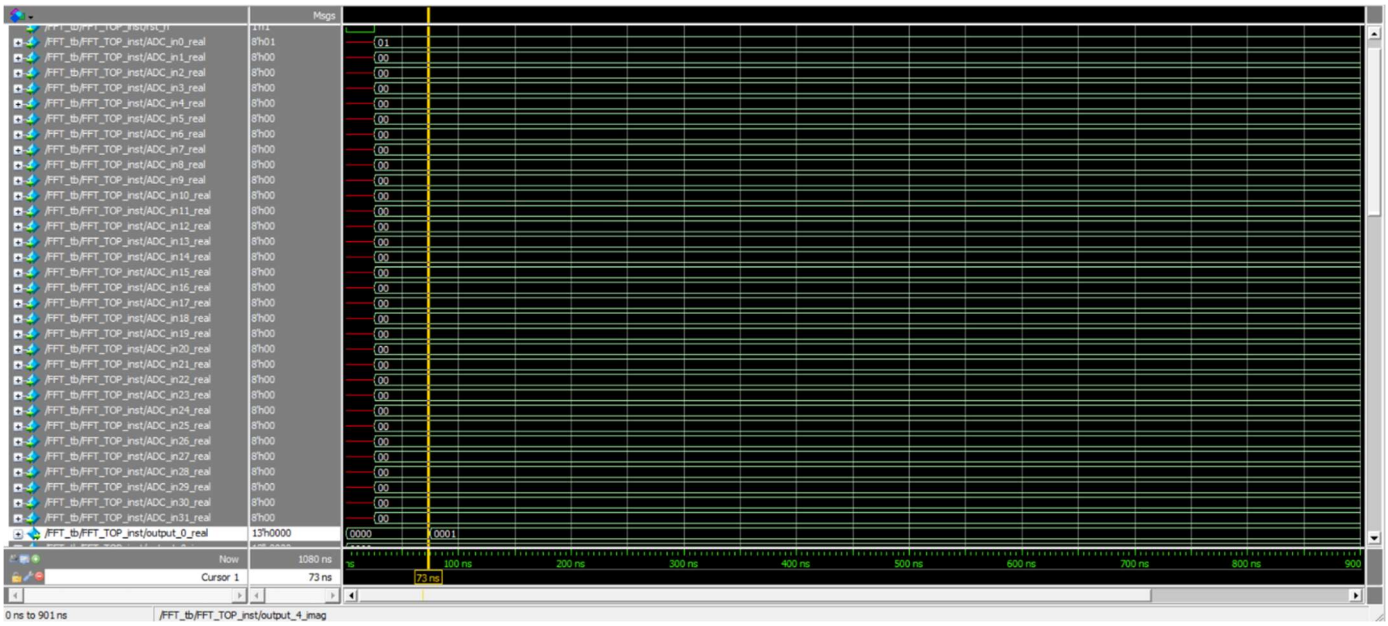


Figure 12: pulse input

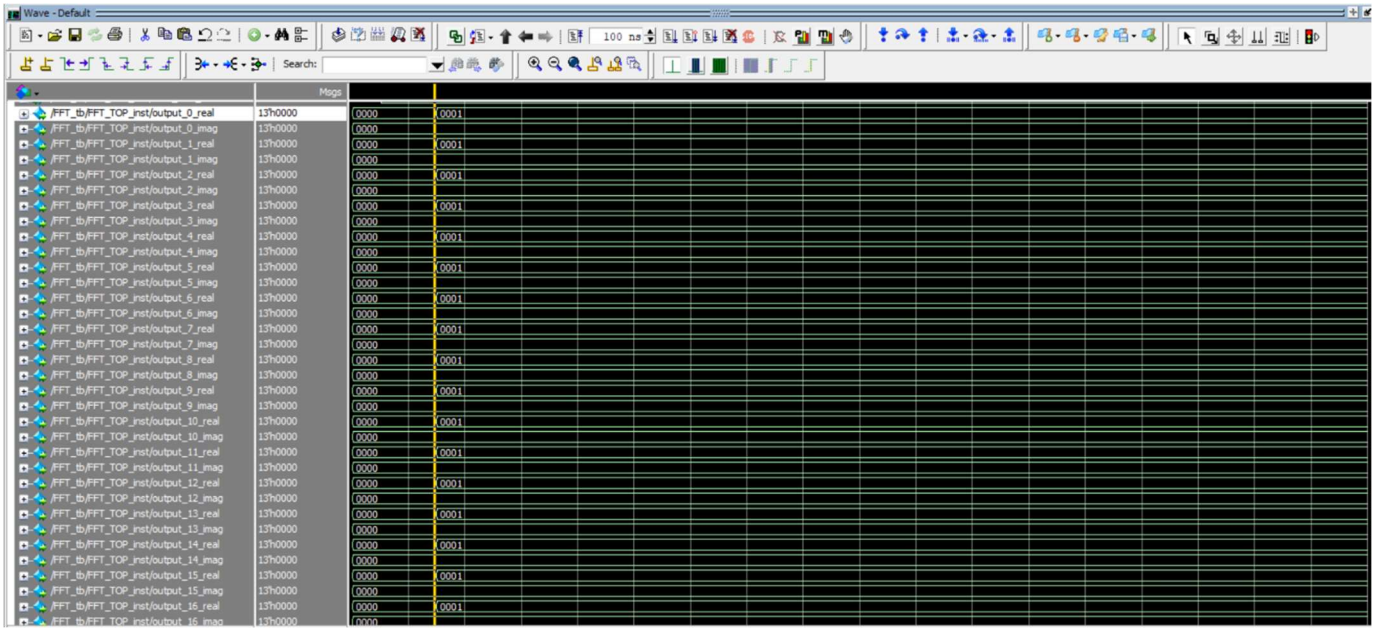


Figure 13: Rect Out

We can see that the output is rectangle as expected from the FFT equation.

IV. FPGA Flow.

Used FPGA: Zynq UltraScale+ ZCU106 Evaluation Platform
Port Number: (xczu7ev-ffvc1156-2-e)

1. Synthesis schematic:

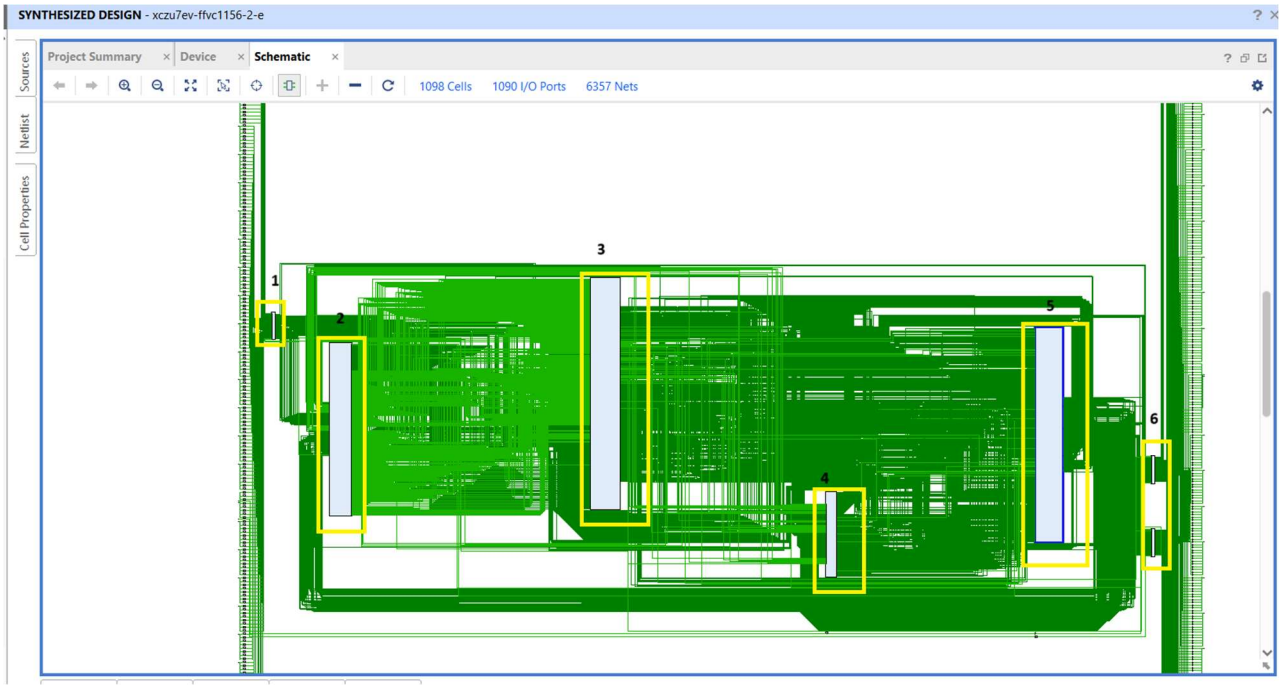


Figure 14: Top View of the synthesized design

We can see clearly the 6 modules we talk about in the architecture:

1. Input register
2. MAC out register
3. MUX selection
4. Control unit
5. Butterfly Unit
6. Output registers

2. Post Synthesis simulation:

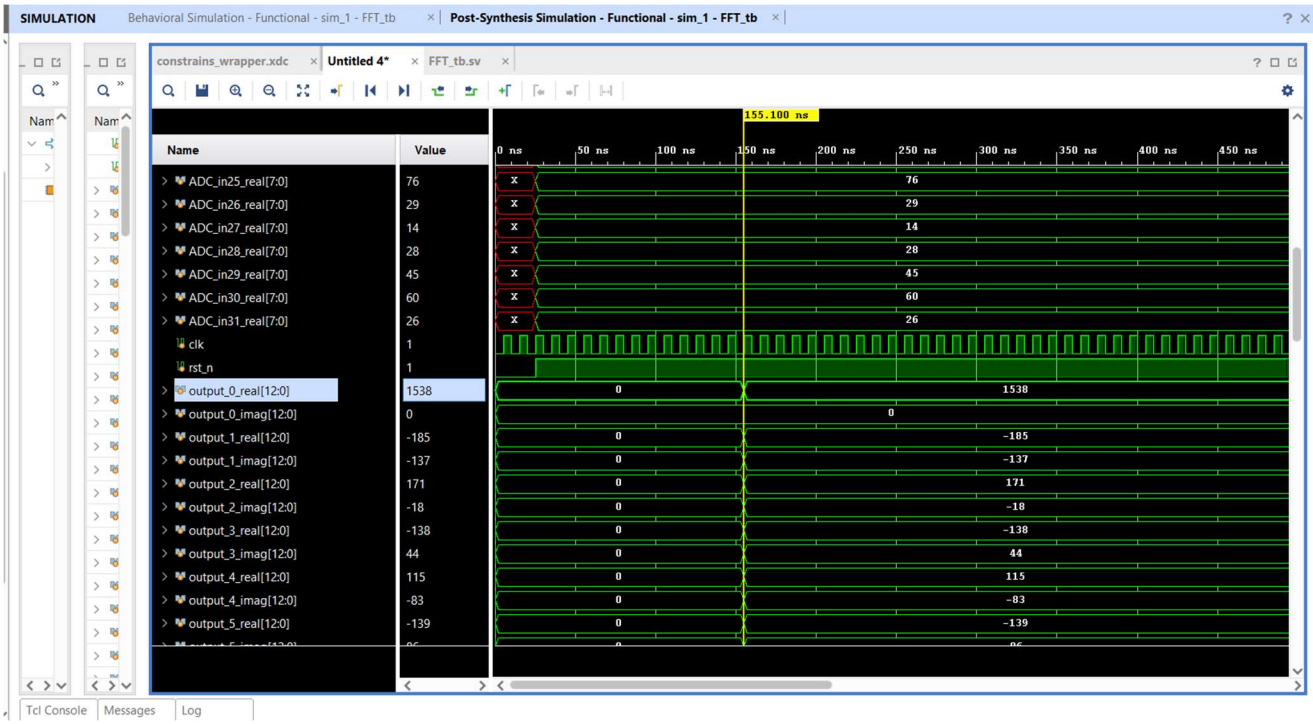


Figure 15: Post synthesis simulation

We can see that the output form Post synthesis simulation is exact as the output pre synthesis simulation.

3. Reports.

```
Hierarchical RTL Component report
Module Register
Detailed RTL Component Info :
+---Registers :
                 8 Bit    Registers := 32
Module control_unit
Detailed RTL Component Info :
+---Adders :
      2 Input      3 Bit       Adders := 1
+---Registers :
                 3 Bit    Registers := 2
+---Muxes :
      2 Input      3 Bit        Muxes := 1
Module Register__parameterized0
Detailed RTL Component Info :
+---Registers :
                13 Bit    Registers := 32
Module complex_mult
Detailed RTL Component Info :
+---Adders :
      3 Input     22 Bit       Adders := 1
      2 Input     22 Bit       Adders := 1
Module round_and_sat
Detailed RTL Component Info :
+---Adders :
      2 Input     24 Bit       Adders := 1
+---Muxes :
      2 Input     13 Bit        Muxes := 2
Module MAC
Detailed RTL Component Info :
+---Adders :
      2 Input     23 Bit       Adders := 2
      3 Input     23 Bit       Adders := 2
---------------------------------------------------------------------------------
Finished RTL Hierarchical Component Statistics
```

Figure 16: RTL Component

```
---------------------------------------------------------------------------------
Finished Writing Synthesis Report : Time (s): cpu = 00:00:39 ; elapsed = 00:00:45 . Memory (MB): peak = 2252.008 ; gain = 887.227
---------------------------------------------------------------------------------
Synthesis finished with 0 errors, 0 critical warnings and 10 warnings.
Synthesis Optimization Runtime : Time (s): cpu = 00:00:31 ; elapsed = 00:00:39 . Memory (MB): peak = 2252.008 ; gain = 734.590
Synthesis Optimization Complete : Time (s): cpu = 00:00:39 ; elapsed = 00:00:45 . Memory (MB): peak = 2252.008 ; gain = 887.227
```

Figure 17: zero error in synthesis

4. Area.

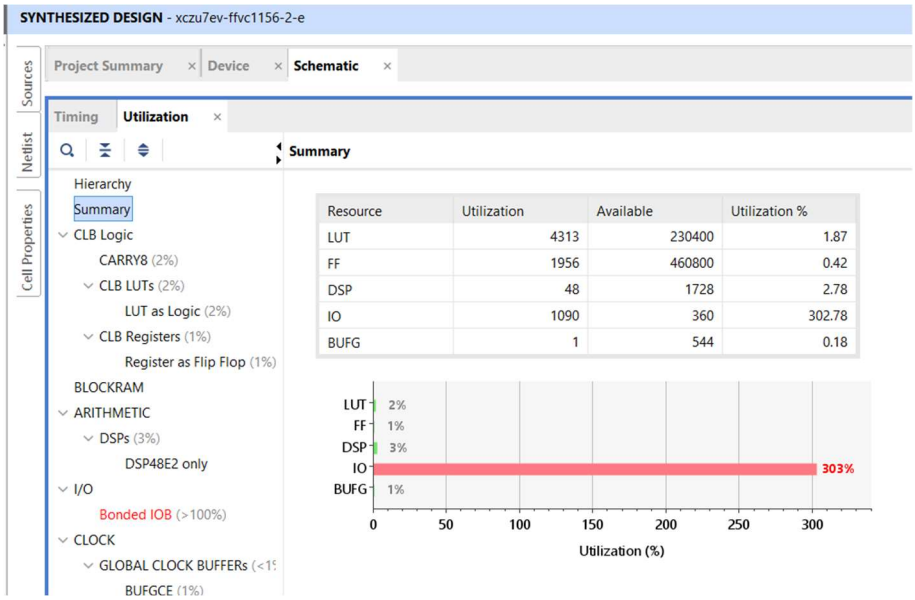


Figure 18: Utilization

We can see that the utilization doesn't exceed 3% in DSP, 2% in LUTs and 1% in FF, But the IOs is expected to fail due to the large number of them, we can use some serial protocols like AXI stream as interface to reduce them.

5. Timing.

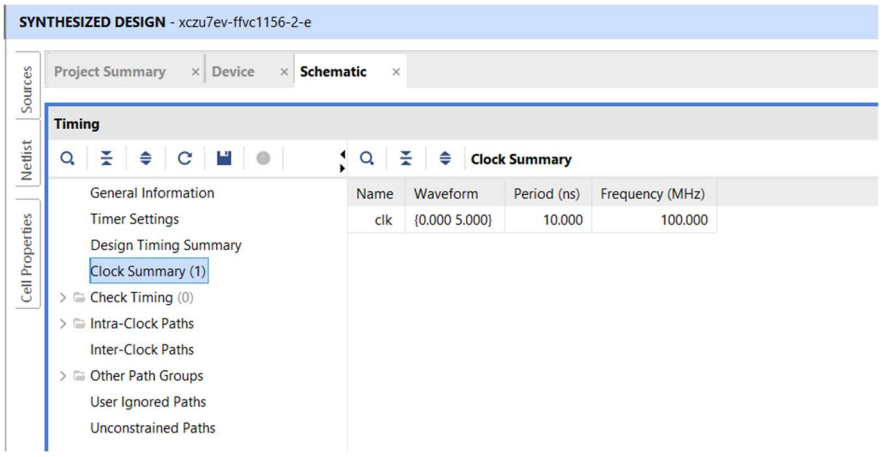


Figure 19: Used Clock

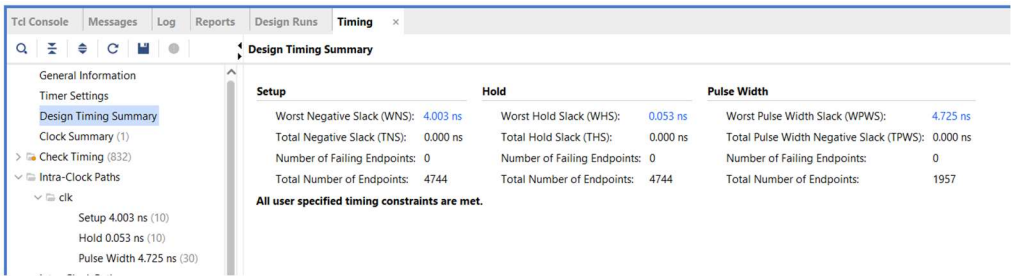
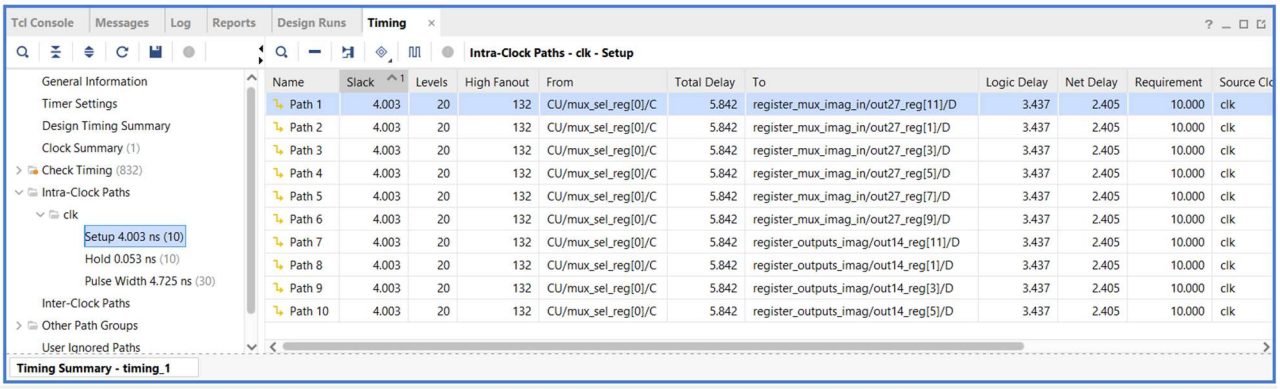


Figure 20: Slack

We can see that we meet setup and hold time.
Setup slack: 4.003 ns and **hold time slack**: 0.053 ns.

6. Setup worst path.

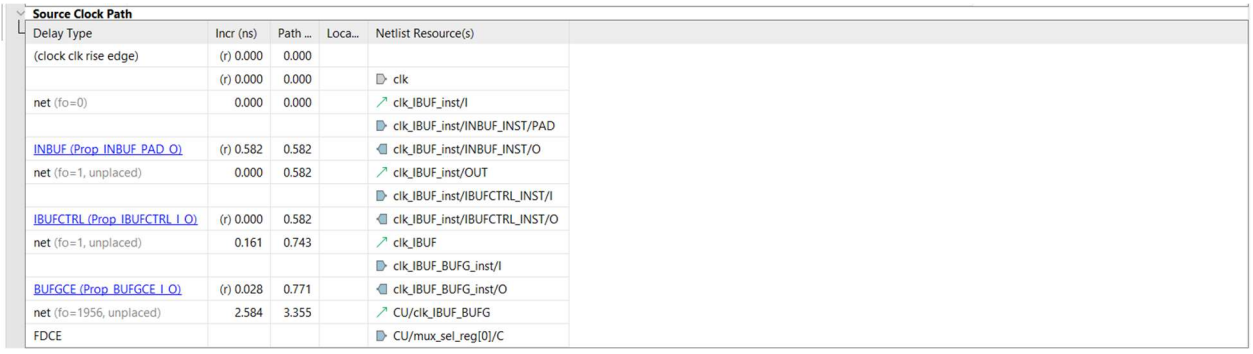


Name	Slack	Levels	High Fanout	From	Total Delay	To	Logic Delay	Net Delay	Requirement	Source Clo
Path 1	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_mux_imag_in/out27_reg[11]/D	3.437	2.405	10.000	clk
Path 2	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_mux_imag_in/out27_reg[1]/D	3.437	2.405	10.000	clk
Path 3	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_mux_imag_in/out27_reg[3]/D	3.437	2.405	10.000	clk
Path 4	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_mux_imag_in/out27_reg[5]/D	3.437	2.405	10.000	clk
Path 5	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_mux_imag_in/out27_reg[7]/D	3.437	2.405	10.000	clk
Path 6	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_mux_imag_in/out27_reg[9]/D	3.437	2.405	10.000	clk
Path 7	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_outputs_imag/out14_reg[11]/D	3.437	2.405	10.000	clk
Path 8	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_outputs_imag/out14_reg[1]/D	3.437	2.405	10.000	clk
Path 9	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_outputs_imag/out14_reg[3]/D	3.437	2.405	10.000	clk
Path 10	4.003	20	132	CU/mux_sel_reg[0]/C	5.842	register_outputs_imag/out14_reg[5]/D	3.437	2.405	10.000	clk

Figure 21: Setup Worst Path

7. Critical path.

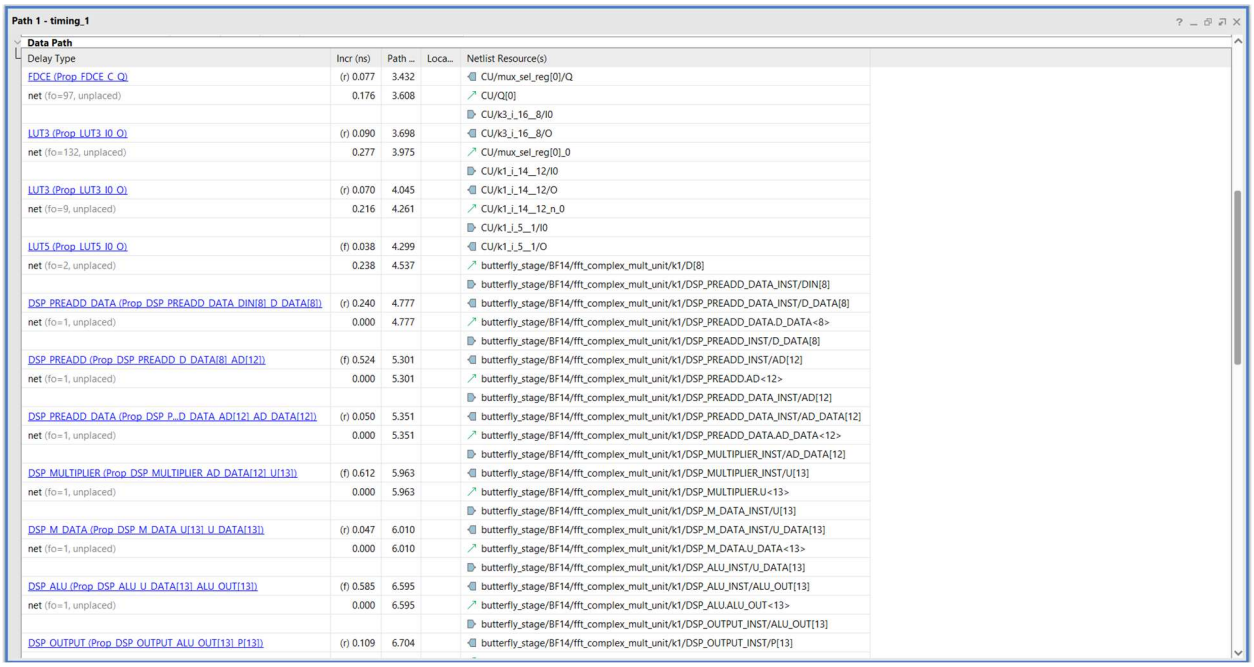
We can see that the **Worst path starts from the Mux to the MAC unit ended in the input of the mux register as expected** as the MAC unit has a **complex mult and adder** which was expected to be the critical path.



Source Clock Path

Delay Type	Incr (ns)	Path ...	Loca...	Netlist Resource(s)
(clock clk rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000		clk
net (fo=0)	0.000	0.000		clk_IBUF_inst/I
				clk_IBUF_inst/INBUF_INST/PAD
INBUF (Prop INBUF_PAD_O)	(r) 0.582	0.582		clk_IBUF_inst/INBUF_INST/O
net (fo=1, unplaced)	0.000	0.582		clk_IBUF_inst/OUT
				clk_IBUF_inst/IBUFCTRL_INST/I
IBUFCTRL (Prop IBUFCTRL_I_O)	(r) 0.000	0.582		clk_IBUF_inst/IBUFCTRL_INST/O
net (fo=1, unplaced)	0.161	0.743		clk_IBUF
				clk_IBUF_BUFG_inst/I
BUFGCE (Prop BUFGCE_I_O)	(r) 0.028	0.771		clk_IBUF_BUFG_inst/O
net (fo=1956, unplaced)	2.584	3.355		CU/clk_IBUF_BUFG
FDCE				CU/mux_sel_reg[0]/C

Figure 22: Source Clock Path

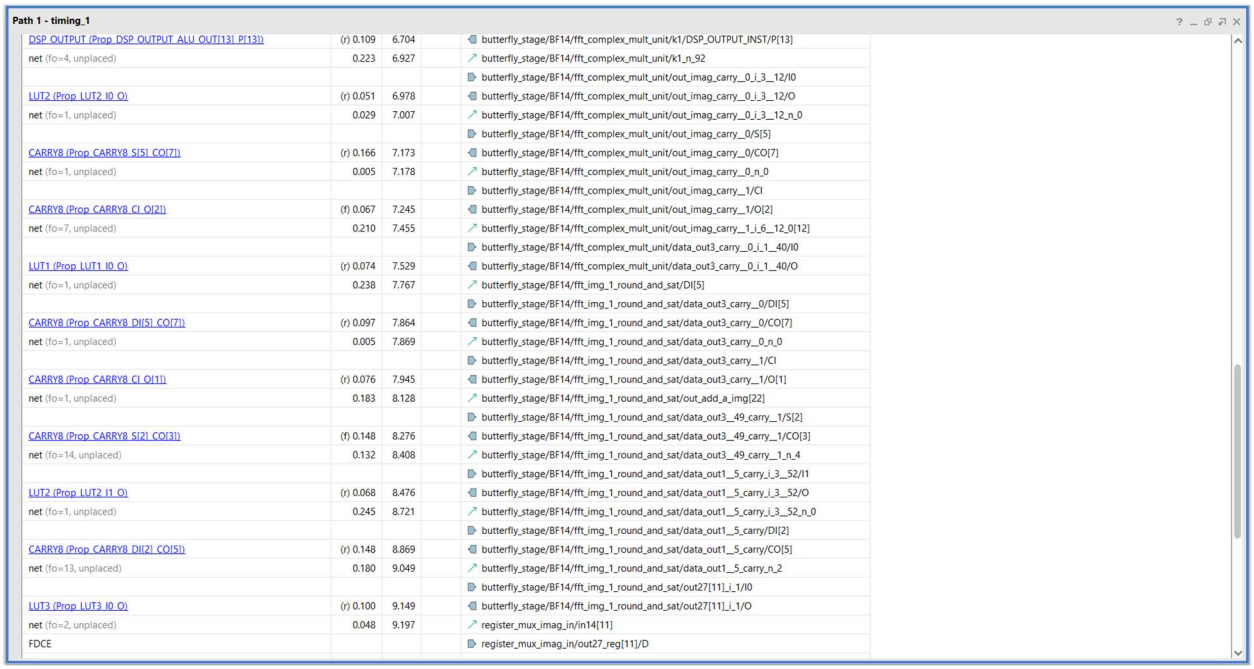


Path 1 - timing_1

Data Path

Delay Type	Incr (ns)	Path ...	Loca...	Netlist Resource(s)
FDCE (Prop_FDCE_C_Q)	(r) 0.077	3.432		CU/mux_sel_reg[0]/Q
net (fo=97, unplaced)	0.176	3.608		CU/Q[0]
				CU/k3_i_16__8/I0
LUT3 (Prop_LUT3_I0_O)	(r) 0.090	3.698		CU/k3_i_16__8/O
net (fo=132, unplaced)	0.277	3.975		CU/mux_sel_reg[0]_0
				CU/k1_i_14__12/I0
LUT3 (Prop_LUT3_I0_O)	(r) 0.070	4.045		CU/k1_i_14__12/O
net (fo=9, unplaced)	0.216	4.261		CU/k1_i_14__12_n_0
				CU/k1_i_5__1/I0
LUT5 (Prop_LUT5_I0_O)	(f) 0.038	4.299		CU/k1_i_5__1/O
net (fo=2, unplaced)	0.238	4.537		butterfly_stage/BF14/fft_complex_mult_unit/k1/D[8]
				butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_PREADD_DATA_INST/DIN[8]
DSP_PREADD_DATA (Prop_DSP_PREADD_DATA_DIN[8]_D_DATA[8])	(r) 0.240	4.777		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_PREADD_DATA_INST/D_DATA[8]
net (fo=1, unplaced)	0.000	4.777		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_PREADD_DATA.D_DATA<8>
				butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_PREADD_INST/D_DATA[8]
DSP_PREADD (Prop_DSP_PREADD_D_DATA[8]_AD[12])	(f) 0.524	5.301		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_PREADD_INST/AD[12]
net (fo=1, unplaced)	0.000	5.301		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_PREADD.AD<12>
				butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_PREADD_DATA_INST/AD[12]
DSP_PREADD_DATA (Prop_DSP_P..D_DATA_AD[12]_AD_DATA[12])	(r) 0.050	5.351		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_PREADD_DATA_INST/AD_DATA[12]
net (fo=1, unplaced)	0.000	5.351		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_PREADD_DATA.AD_DATA<12>
				butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_MULTIPLIER_INST/AD_DATA[12]
DSP_MULTIPLIER (Prop_DSP_MULTIPLIER_AD_DATA[12]_U[13])	(f) 0.612	5.963		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_MULTIPLIER_INST/U[13]
net (fo=1, unplaced)	0.000	5.963		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_MULTIPLIER.U<13>
				butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_M_DATA_INST/U[13]
DSP_M_DATA (Prop_DSP_M_DATA_U[13]_U_DATA[13])	(r) 0.047	6.010		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_M_DATA_INST/U_DATA[13]
net (fo=1, unplaced)	0.000	6.010		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_M_DATA.U_DATA<13>
				butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_ALU_INST/U_DATA[13]
DSP_ALU (Prop_DSP_ALU_U_DATA[13]_ALU_OUT[13])	(f) 0.585	6.595		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_ALU_INST/ALU_OUT[13]
net (fo=1, unplaced)	0.000	6.595		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_ALU.ALU_OUT<13>
				butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_OUTPUT_INST/ALU_OUT[13]
DSP_OUTPUT (Prop_DSP_OUTPUT_ALU_OUT[13]_P[13])	(r) 0.109	6.704		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_OUTPUT_INST/P[13]

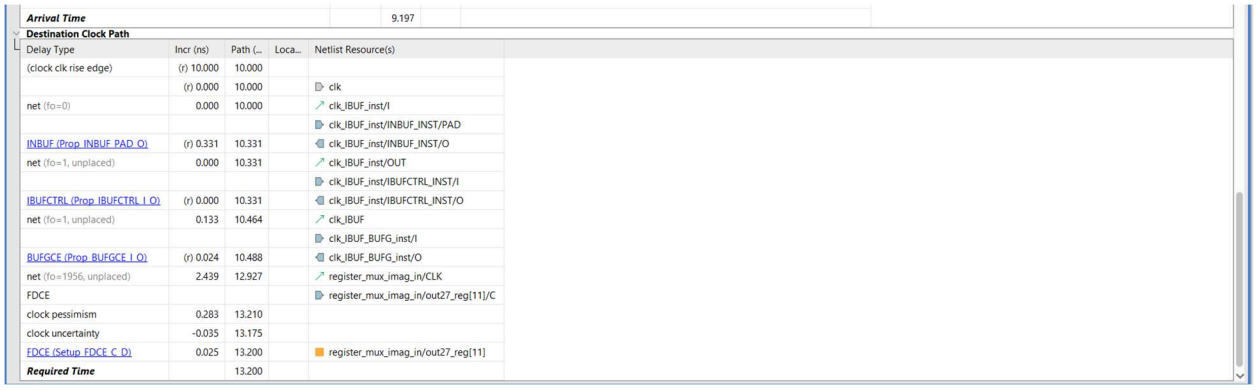
Figure 23: Path timing



Path 1 - timing_1

DSP_OUTPUT (Prop_DSP_OUTPUT_ALU_OUT[13]_P[13])	(r) 0.109	6.704		butterfly_stage/BF14/fft_complex_mult_unit/k1/DSP_OUTPUT_INST/P[13]
net (fo=4, unplaced)	0.223	6.927		butterfly_stage/BF14/fft_complex_mult_unit/k1_n_92
				butterfly_stage/BF14/fft_complex_mult_unit/out_imag_carry__0_i_3__12/I0
LUT2 (Prop_LUT2_I0_O)	(r) 0.051	6.978		butterfly_stage/BF14/fft_complex_mult_unit/out_imag_carry__0_i_3__12/O
net (fo=1, unplaced)	0.029	7.007		butterfly_stage/BF14/fft_complex_mult_unit/out_imag_carry__0_i_3__12_n_0
				butterfly_stage/BF14/fft_complex_mult_unit/out_imag_carry__0/S[5]
CARRY8 (Prop_CARRY8_S[5]_CO[7])	(r) 0.166	7.173		butterfly_stage/BF14/fft_complex_mult_unit/out_imag_carry__0/CO[7]
net (fo=1, unplaced)	0.005	7.178		butterfly_stage/BF14/fft_complex_mult_unit/out_imag_carry__0_n_0
				butterfly_stage/BF14/fft_complex_mult_unit/out_imag_carry__1/CI
CARRY8 (Prop_CARRY8_CI_O[2])	(f) 0.067	7.245		butterfly_stage/BF14/fft_complex_mult_unit/out_imag_carry__1/O[2]
net (fo=7, unplaced)	0.210	7.455		butterfly_stage/BF14/fft_complex_mult_unit/out_imag_carry__1_i_6__12_0[12]
				butterfly_stage/BF14/fft_complex_mult_unit/data_out3_carry__0_i_1__40/I0
LUT1 (Prop_LUT1_I0_O)	(r) 0.074	7.529		butterfly_stage/BF14/fft_complex_mult_unit/data_out3_carry__0_i_1__40/O
net (fo=1, unplaced)	0.238	7.767		butterfly_stage/BF14/fft_img_1_round_and_sat/DI[5]
				butterfly_stage/BF14/fft_img_1_round_and_sat/data_out3_carry__0/DI[5]
CARRY8 (Prop_CARRY8_DI[5]_CO[7])	(r) 0.097	7.864		butterfly_stage/BF14/fft_img_1_round_and_sat/data_out3_carry__0/CO[7]
net (fo=1, unplaced)	0.005	7.869		butterfly_stage/BF14/fft_img_1_round_and_sat/data_out3_carry__0_n_0
				butterfly_stage/BF14/fft_img_1_round_and_sat/data_out3_carry__1/CI
CARRY8 (Prop_CARRY8_CI_O[1])	(r) 0.076	7.945		butterfly_stage/BF14/fft_img_1_round_and_sat/data_out3_carry__1/O[1]
net (fo=1, unplaced)	0.183	8.128		butterfly_stage/BF14/fft_img_1_round_and_sat/out_add_a_img[22]
				butterfly_stage/BF14/fft_img_1_round_and_sat/data_out3__49_carry__1/S[2]
CARRY8 (Prop_CARRY8_S[2]_CO[3])	(f) 0.148	8.276		butterfly_stage/BF14/fft_img_1_round_and_sat/data_out3__49_carry__1/CO[3]
net (fo=14, unplaced)	0.132	8.408		butterfly_stage/BF14/fft_img_1_round_and_sat/data_out3__49_carry__1_n_4
				butterfly_stage/BF14/fft_img_1_round_and_sat/data_out1__5_carry_i_3__52/I1
LUT2 (Prop_LUT2_I1_O)	(r) 0.068	8.476		butterfly_stage/BF14/fft_img_1_round_and_sat/data_out1__5_carry_i_3__52/O
net (fo=1, unplaced)	0.245	8.721		butterfly_stage/BF14/fft_img_1_round_and_sat/data_out1__5_carry_i_3__52_n_0
				butterfly_stage/BF14/fft_img_1_round_and_sat/data_out1__5_carry/DI[2]
CARRY8 (Prop_CARRY8_DI[2]_CO[5])	(r) 0.148	8.869		butterfly_stage/BF14/fft_img_1_round_and_sat/data_out1__5_carry/CO[5]
net (fo=13, unplaced)	0.180	9.049		butterfly_stage/BF14/fft_img_1_round_and_sat/data_out1__5_carry_n_2
				butterfly_stage/BF14/fft_img_1_round_and_sat/out27[11]_i_1/I0
LUT3 (Prop_LUT3_I0_O)	(r) 0.100	9.149		butterfly_stage/BF14/fft_img_1_round_and_sat/out27[11]_i_1/O
net (fo=2, unplaced)	0.048	9.197		register_mux_imag_in/in14[11]
FDCE				register_mux_imag_in/out27_reg[11]/D

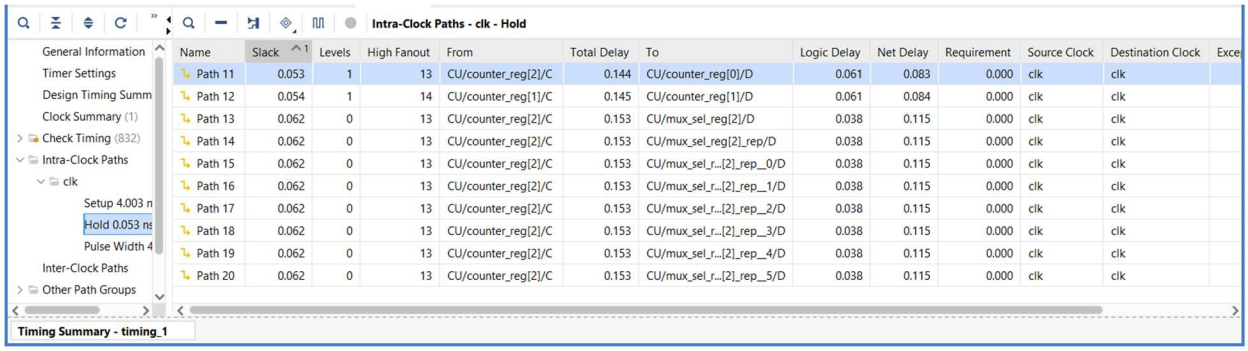
Figure 24: Path Timing



Arrival Time				9.197
Destination Clock Path				
Delay Type	Incr (ns)	Path (...	Loca...	Netlist Resource(s)
(clock clk rise edge)	(r) 10.000	10.000		
	(r) 0.000	10.000		clk
net (fo=0)	0.000	10.000		clk_IBUF_inst/I
				clk_IBUF_inst/INBUF_INST/PAD
INBUF (Prop_INBUF_PAD_O)	(r) 0.331	10.331		clk_IBUF_inst/INBUF_INST/O
net (fo=1, unplaced)	0.000	10.331		clk_IBUF_inst/OUT
				clk_IBUF_inst/IBUFCTRL_INST/I
IBUFCTRL (Prop_IBUFCTRL_I_O)	(r) 0.000	10.331		clk_IBUF_inst/IBUFCTRL_INST/O
net (fo=1, unplaced)	0.133	10.464		clk_IBUF
				clk_IBUF_BUFG_inst/I
BUFGCE (Prop_BUFGCE_I_O)	(r) 0.024	10.488		clk_IBUF_BUFG_inst/O
net (fo=1956, unplaced)	2.439	12.927		register_mux_imag_in/CLK
FDCE				register_mux_imag_in/out27_reg[11]/C
clock pessimism	0.283	13.210		
clock uncertainty	-0.035	13.175		
FDCE (Setup_FDCE_C_D)	0.025	13.200		register_mux_imag_in/out27_reg[11]
Required Time		13.200		

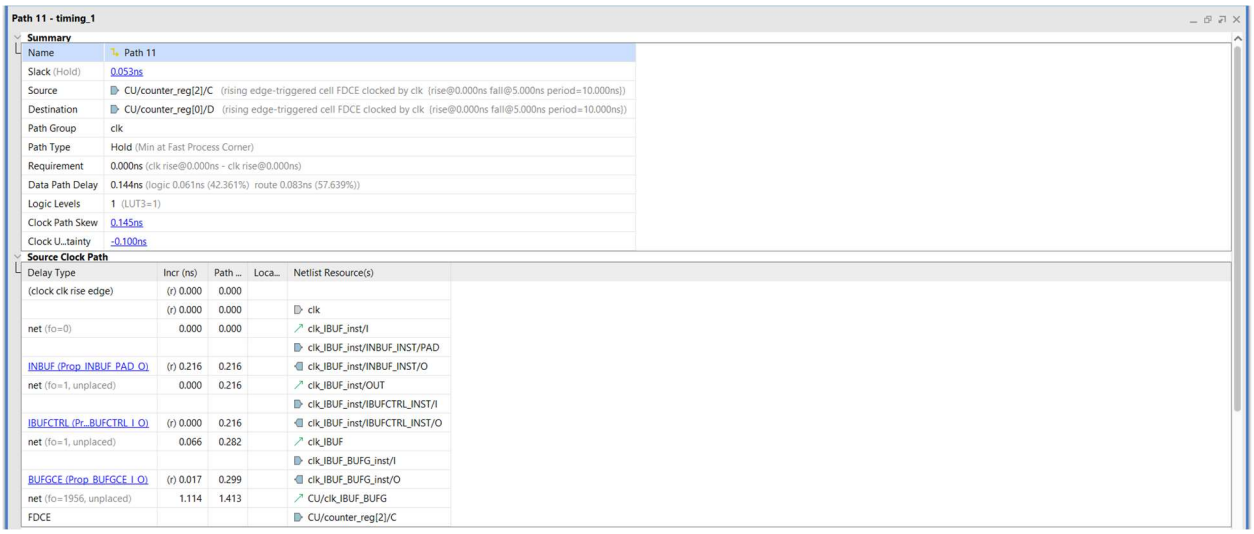
Figure 25: Arrival and Required time

8. Hold Time report.



Name	Slack	Levels	High Fanout	From	Total Delay	To	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 11	0.053	1	13	CU/counter_reg[2]/C	0.144	CU/counter_reg[0]/D	0.061	0.083	0.000	clk	clk
Path 12	0.054	1	14	CU/counter_reg[2]/C	0.145	CU/counter_reg[1]/D	0.061	0.084	0.000	clk	clk
Path 13	0.062	0	13	CU/counter_reg[2]/C	0.153	CU/mux_sel_reg[2]/D	0.038	0.115	0.000	clk	clk
Path 14	0.062	0	13	CU/counter_reg[2]/C	0.153	CU/mux_sel_reg[2]_rep/D	0.038	0.115	0.000	clk	clk
Path 15	0.062	0	13	CU/counter_reg[2]/C	0.153	CU/mux_sel_r...[2]_rep__0/D	0.038	0.115	0.000	clk	clk
Path 16	0.062	0	13	CU/counter_reg[2]/C	0.153	CU/mux_sel_r...[2]_rep__1/D	0.038	0.115	0.000	clk	clk
Path 17	0.062	0	13	CU/counter_reg[2]/C	0.153	CU/mux_sel_r...[2]_rep__2/D	0.038	0.115	0.000	clk	clk
Path 18	0.062	0	13	CU/counter_reg[2]/C	0.153	CU/mux_sel_r...[2]_rep__3/D	0.038	0.115	0.000	clk	clk
Path 19	0.062	0	13	CU/counter_reg[2]/C	0.153	CU/mux_sel_r...[2]_rep__4/D	0.038	0.115	0.000	clk	clk
Path 20	0.062	0	13	CU/counter_reg[2]/C	0.153	CU/mux_sel_r...[2]_rep__5/D	0.038	0.115	0.000	clk	clk

Figure 26: Hold time path



Path 11 - timing_1

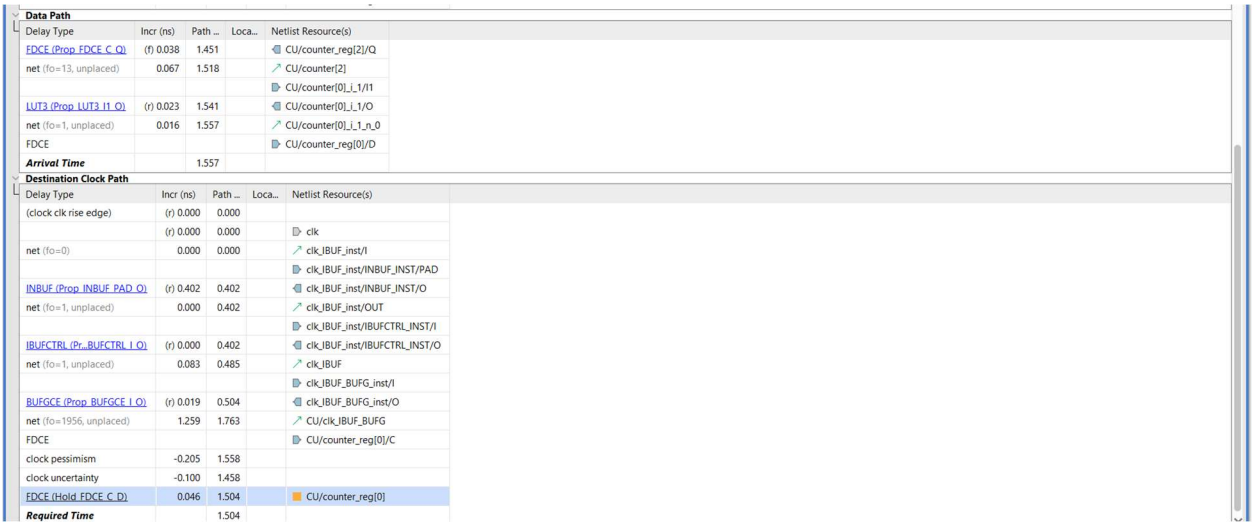
Summary

Name	Path 11
Slack (Hold)	0.053ns
Source	CU/counter_reg[2]/C (rising edge-triggered cell FDCE clocked by clk {rise@0.000ns fall@5.000ns period=10.000ns})
Destination	CU/counter_reg[0]/D (rising edge-triggered cell FDCE clocked by clk {rise@0.000ns fall@5.000ns period=10.000ns})
Path Group	clk
Path Type	Hold (Min at Fast Process Corner)
Requirement	0.000ns (clk rise@0.000ns - clk rise@0.000ns)
Data Path Delay	0.144ns (logic 0.061ns (42.361%) route 0.083ns (57.639%))
Logic Levels	1 (LUT3=1)
Clock Path Skew	0.145ns
Clock U...tainty	-0.100ns

Source Clock Path

Delay Type	Incr (ns)	Path ...	Loca...	Netlist Resource(s)
(clock clk rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000		clk
net (fo=0)	0.000	0.000		clk_IBUF_inst/I
				clk_IBUF_inst/INBUF_INST/PAD
INBUF (Prop_INBUF_PAD_O)	(r) 0.216	0.216		clk_IBUF_inst/INBUF_INST/O
net (fo=1, unplaced)	0.000	0.216		clk_IBUF_inst/OUT
				clk_IBUF_inst/IBUFCTRL_INST/I
IBUFCTRL (Pr...BUFCTRL_I_O)	(r) 0.000	0.216		clk_IBUF_inst/IBUFCTRL_INST/O
net (fo=1, unplaced)	0.066	0.282		clk_IBUF
				clk_IBUF_BUFG_inst/I
BUFGCE (Prop_BUFGCE_I_O)	(r) 0.017	0.299		clk_IBUF_BUFG_inst/O
net (fo=1956, unplaced)	1.114	1.413		CU/clk_IBUF_BUFG
FDCE				CU/counter_reg[2]/C

Figure 27: Hold time path



Data Path

Delay Type	Incr (ns)	Path ...	Loca...	Netlist Resource(s)
FDCE (Prop_FDCE_C_Q)	(f) 0.038	1.451		CU/counter_reg[2]/Q
net (fo=13, unplaced)	0.067	1.518		CU/counter[2]
				CU/counter[0]_i_1/I1
LUT3 (Prop_LUT3_I1_O)	(r) 0.023	1.541		CU/counter[0]_i_1/O
net (fo=1, unplaced)	0.016	1.557		CU/counter[0]_i_1_n_0
FDCE				CU/counter_reg[0]/D
Arrival Time		1.557		

Destination Clock Path

Delay Type	Incr (ns)	Path ...	Loca...	Netlist Resource(s)
(clock clk rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000		clk
net (fo=0)	0.000	0.000		clk_IBUF_inst/I
				clk_IBUF_inst/INBUF_INST/PAD
INBUF (Prop_INBUF_PAD_O)	(r) 0.402	0.402		clk_IBUF_inst/INBUF_INST/O
net (fo=1, unplaced)	0.000	0.402		clk_IBUF_inst/OUT
				clk_IBUF_inst/IBUFCTRL_INST/I
IBUFCTRL (Pr...BUFCTRL_I_O)	(r) 0.000	0.402		clk_IBUF_inst/IBUFCTRL_INST/O
net (fo=1, unplaced)	0.083	0.485		clk_IBUF
				clk_IBUF_BUFG_inst/I
BUFGCE (Prop_BUFGCE_I_O)	(r) 0.019	0.504		clk_IBUF_BUFG_inst/O
net (fo=1956, unplaced)	1.259	1.763		CU/clk_IBUF_BUFG
FDCE				CU/counter_reg[0]/C
clock pessimism	-0.205	1.558		
clock uncertainty	-0.100	1.458		
FDCE (Hold_FDCE_C_D)	0.046	1.504		CU/counter_reg[0]
Required Time		1.504		

Figure 28: Hold time path

9. Power.

We start calculating the saif file for 1080ns of simulation time then we upload the **saif file** to have **accurate calculation for the power**.

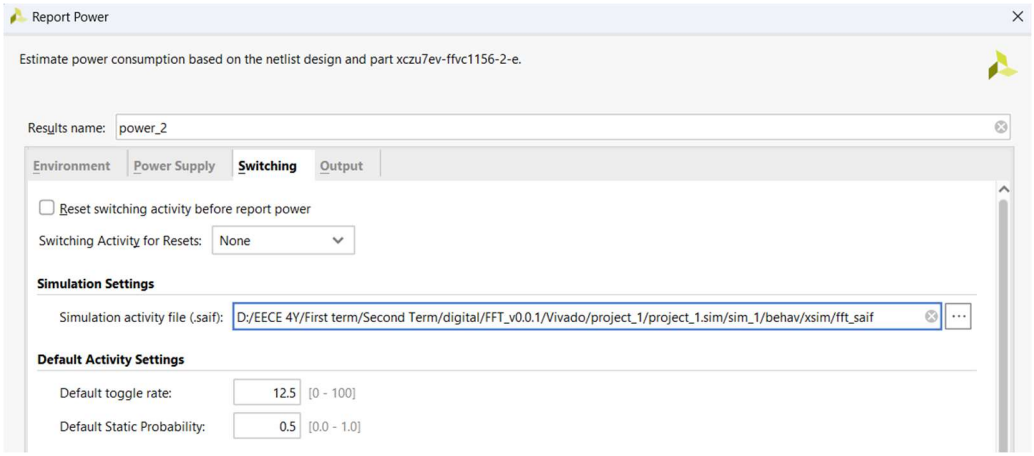


Figure 29: adding saif file

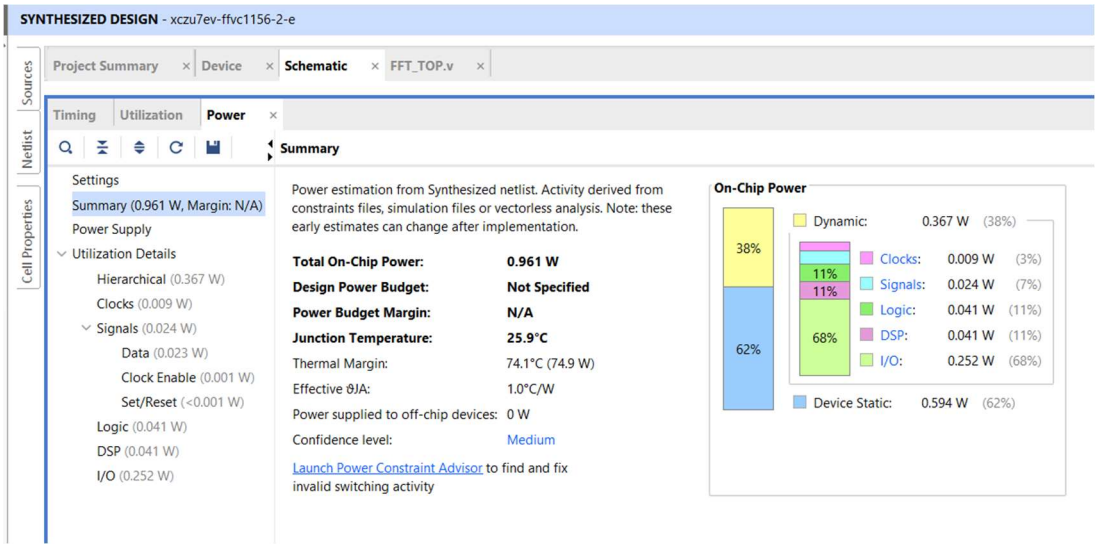


Figure 30: Power Report after adding saif file