WES 237C - Hardware for Embedded Systems, Kastner

Project 4, Project: Fast Fourier Transform (FFT) and OFDM Receiver

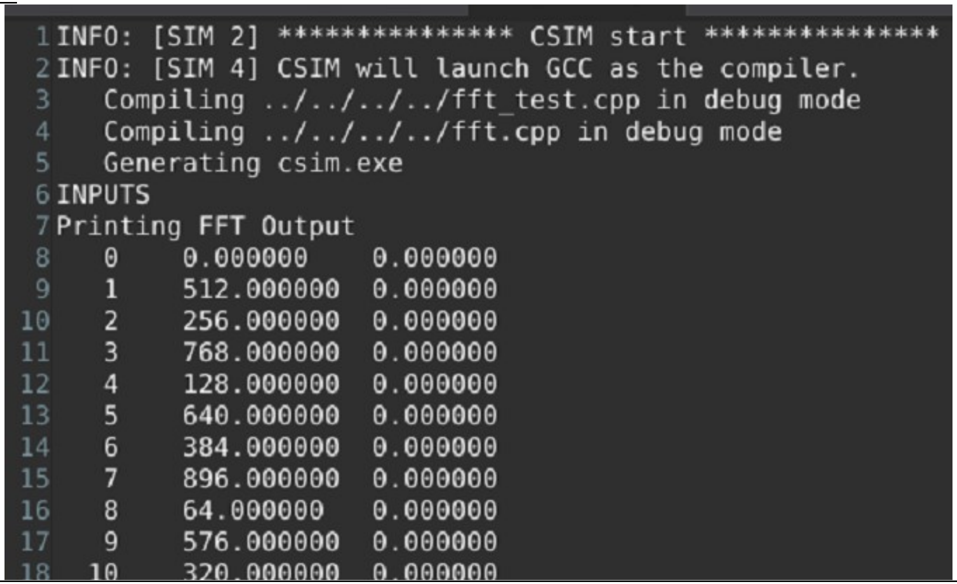
Ricardo Lizárraga

rilizarraga@ucsd.edu **PID: A69028483**

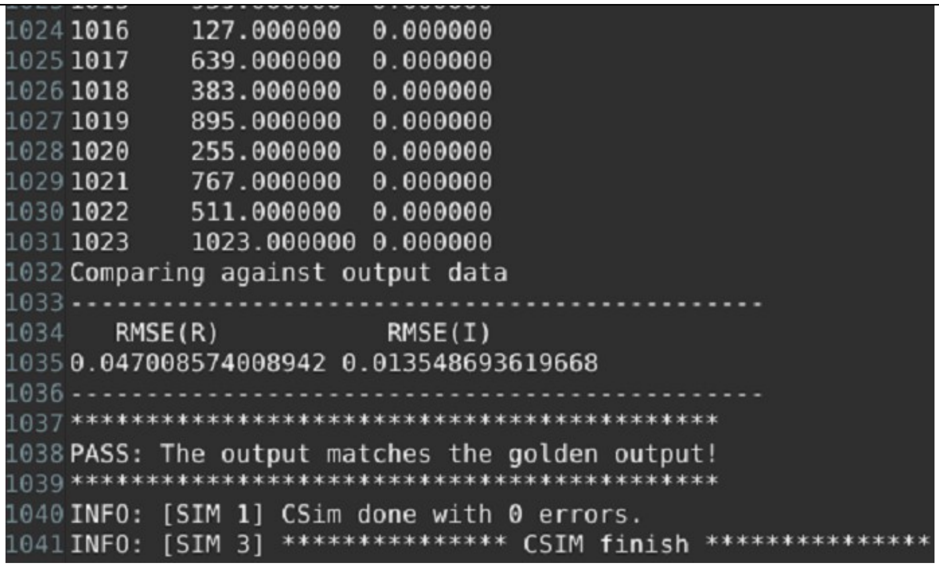
Submit to:
Ryan Charles Kastner
(TA) Zhenghua Ma
(TA) Abarajithan Gnaneswaran

fft1024_Baseline

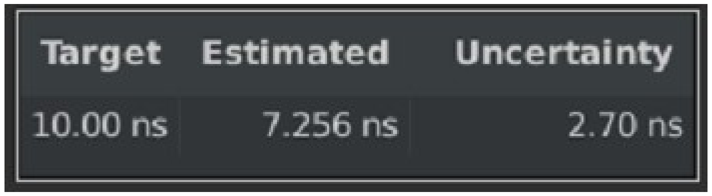
TEST BENCH : PASS



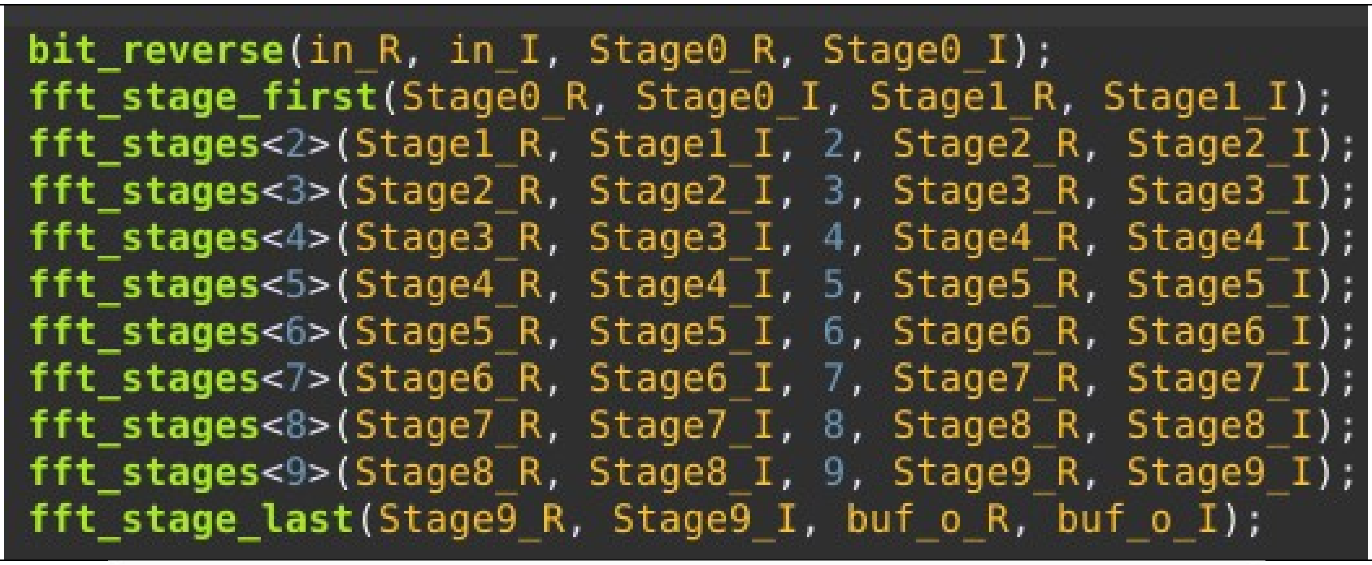
```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../fft_test.cpp in debug mode
   Compiling ../../../../fft.cpp in debug mode
   Generating csim.exe
INPUTS
Printing FFT Output
   0     0.000000    0.000000
   1     512.000000  0.000000
   2     256.000000  0.000000
   3     768.000000  0.000000
   4     128.000000  0.000000
   5     640.000000  0.000000
   6     384.000000  0.000000
   7     896.000000  0.000000
   8     64.000000   0.000000
   9     576.000000  0.000000
  10     320.000000  0.000000
```



```
1016     127.000000  0.000000
1017     639.000000  0.000000
1018     383.000000  0.000000
1019     895.000000  0.000000
1020     255.000000  0.000000
1021     767.000000  0.000000
1022     511.000000  0.000000
1023     1023.000000 0.000000
Comparing against output data
---------------------------------------------
   RMSE(R)           RMSE(I)
0.047008574008942 0.013548693619668
---------------------------------------------
*******************************************
PASS: The output matches the golden output!
*******************************************
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```



Target	Estimated	Uncertainty
10.00 ns	7.256 ns	2.70 ns



```
bit_reverse(in_R, in_I, Stage0_R, Stage0_I);
fft_stage_first(Stage0_R, Stage0_I, Stage1_R, Stage1_I);
fft_stages<2>(Stage1_R, Stage1_I, 2, Stage2_R, Stage2_I);
fft_stages<3>(Stage2_R, Stage2_I, 3, Stage3_R, Stage3_I);
fft_stages<4>(Stage3_R, Stage3_I, 4, Stage4_R, Stage4_I);
fft_stages<5>(Stage4_R, Stage4_I, 5, Stage5_R, Stage5_I);
fft_stages<6>(Stage5_R, Stage5_I, 6, Stage6_R, Stage6_I);
fft_stages<7>(Stage6_R, Stage6_I, 7, Stage7_R, Stage7_I);
fft_stages<8>(Stage7_R, Stage7_I, 8, Stage8_R, Stage8_I);
fft_stages<9>(Stage8_R, Stage8_I, 9, Stage9_R, Stage9_I);
fft_stage_last(Stage9_R, Stage9_I, buf_o_R, buf_o_I);
```

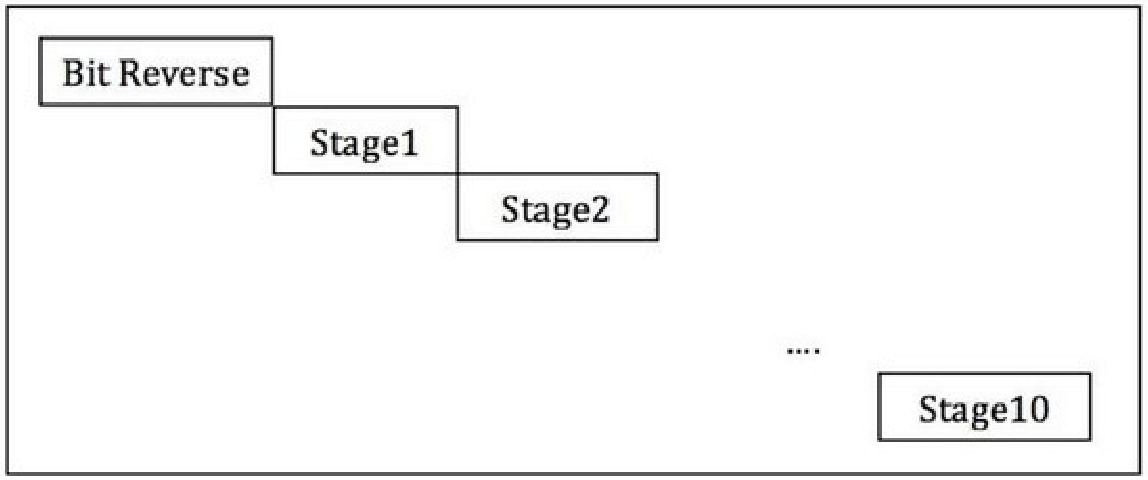
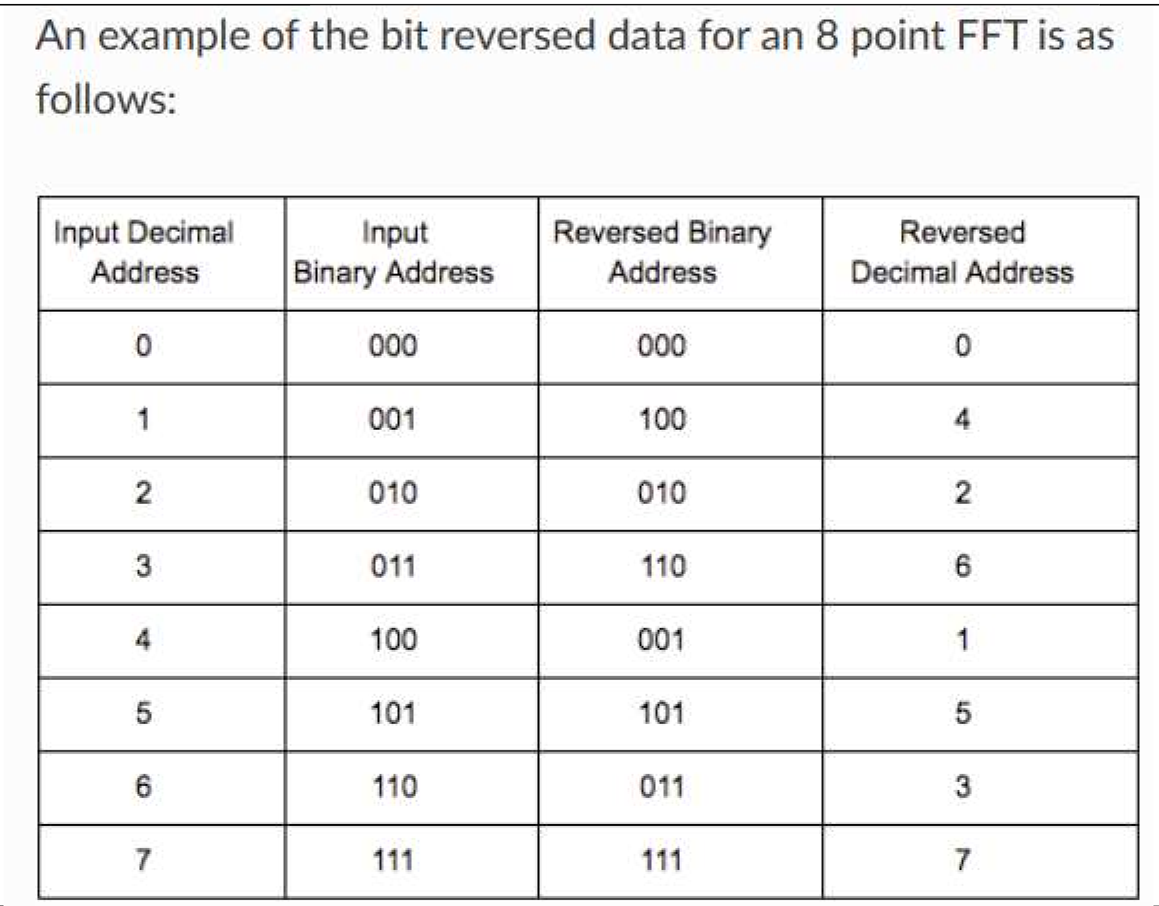
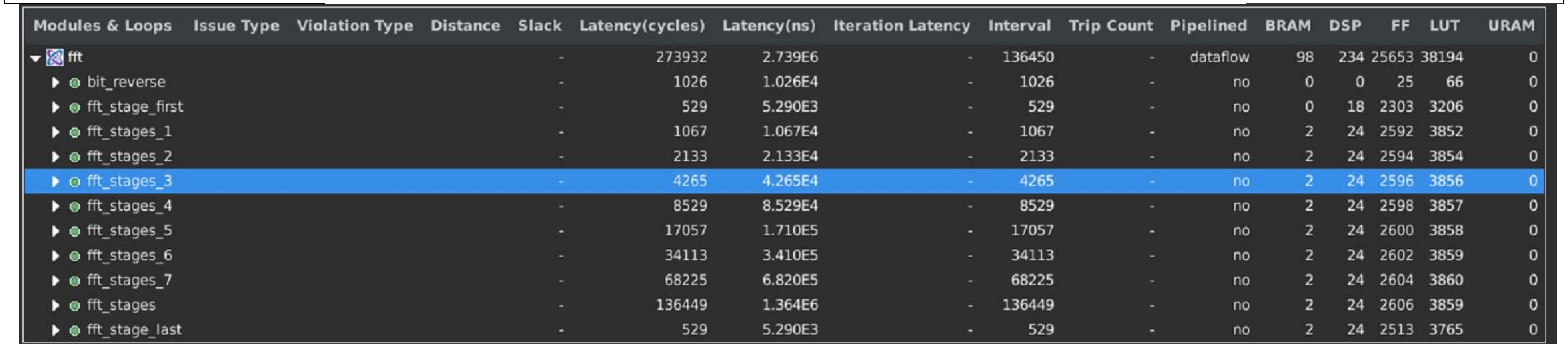


Figure 1: A staged implementation of a 1024 FFT. Bit reversal is followed by 10 stages of butterfly computations. This architecture is capable of pipeline both within the stages and across the stages.



An example of the bit reversed data for an 8 point FFT is as follows:

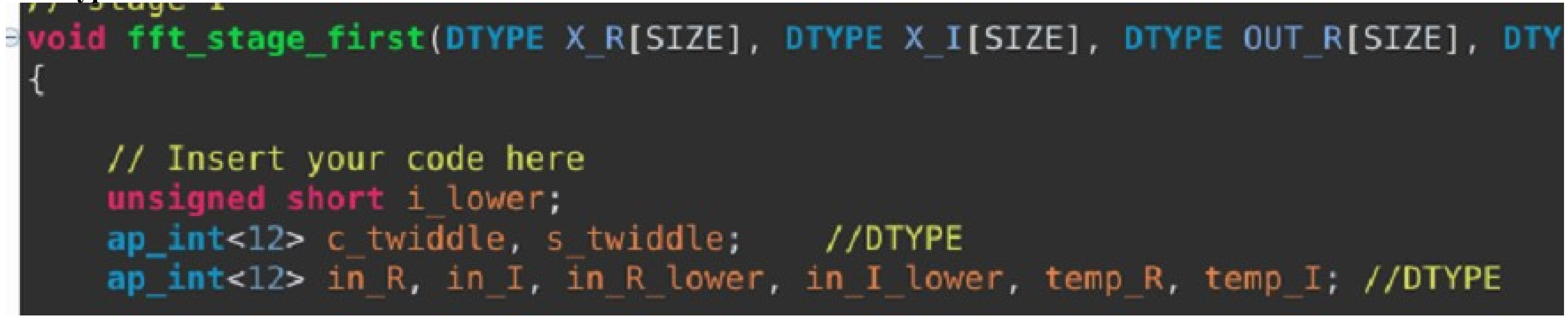
Input Decimal Address	Input Binary Address	Reversed Binary Address	Reversed Decimal Address
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7



Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
fft				-	273932	2.739E6	-	136450	-	dataflow	98	234	25653	38194	0
bit_reverse				-	1026	1.026E4	-	1026	-	no	0	0	25	66	0
fft_stage_first				-	529	5.290E3	-	529	-	no	0	18	2303	3206	0
fft_stages_1				-	1067	1.067E4	-	1067	-	no	2	24	2592	3852	0
fft_stages_2				-	2133	2.133E4	-	2133	-	no	2	24	2594	3854	0
fft_stages_3				-	4265	4.265E4	-	4265	-	no	2	24	2596	3856	0
fft_stages_4				-	8529	8.529E4	-	8529	-	no	2	24	2598	3857	0
fft_stages_5				-	17057	1.710E5	-	17057	-	no	2	24	2600	3858	0
fft_stages_6				-	34113	3.410E5	-	34113	-	no	2	24	2602	3859	0
fft_stages_7				-	68225	6.820E5	-	68225	-	no	2	24	2604	3860	0
fft_stages				-	136449	1.364E6	-	136449	-	no	2	24	2606	3859	0
fft_stage_last				-	529	5.290E3	-	529	-	no	2	24	2513	3765	0

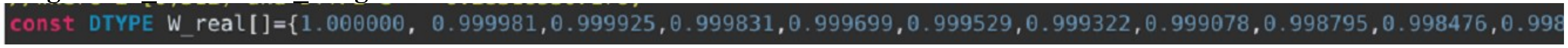
fft1024_Best Optimizations

- **Data type**

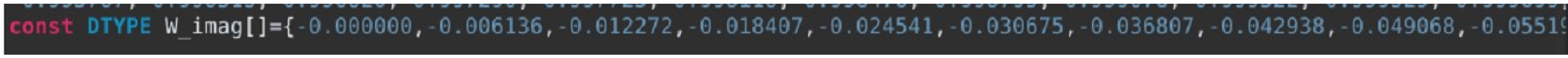


```
void fft_stage_first(DTYPE X_R[SIZE], DTYPE X_I[SIZE], DTYPE OUT_R[SIZE], DTY
{

    // Insert your code here
    unsigned short i_lower;
    ap_int<12> c_twiddle, s_twiddle;     //DTYPE
    ap_int<12> in_R, in_I, in_R_lower, in_I_lower, temp_R, temp_I; //DTYPE
```

- **Using the W_real and W_imag tables:**

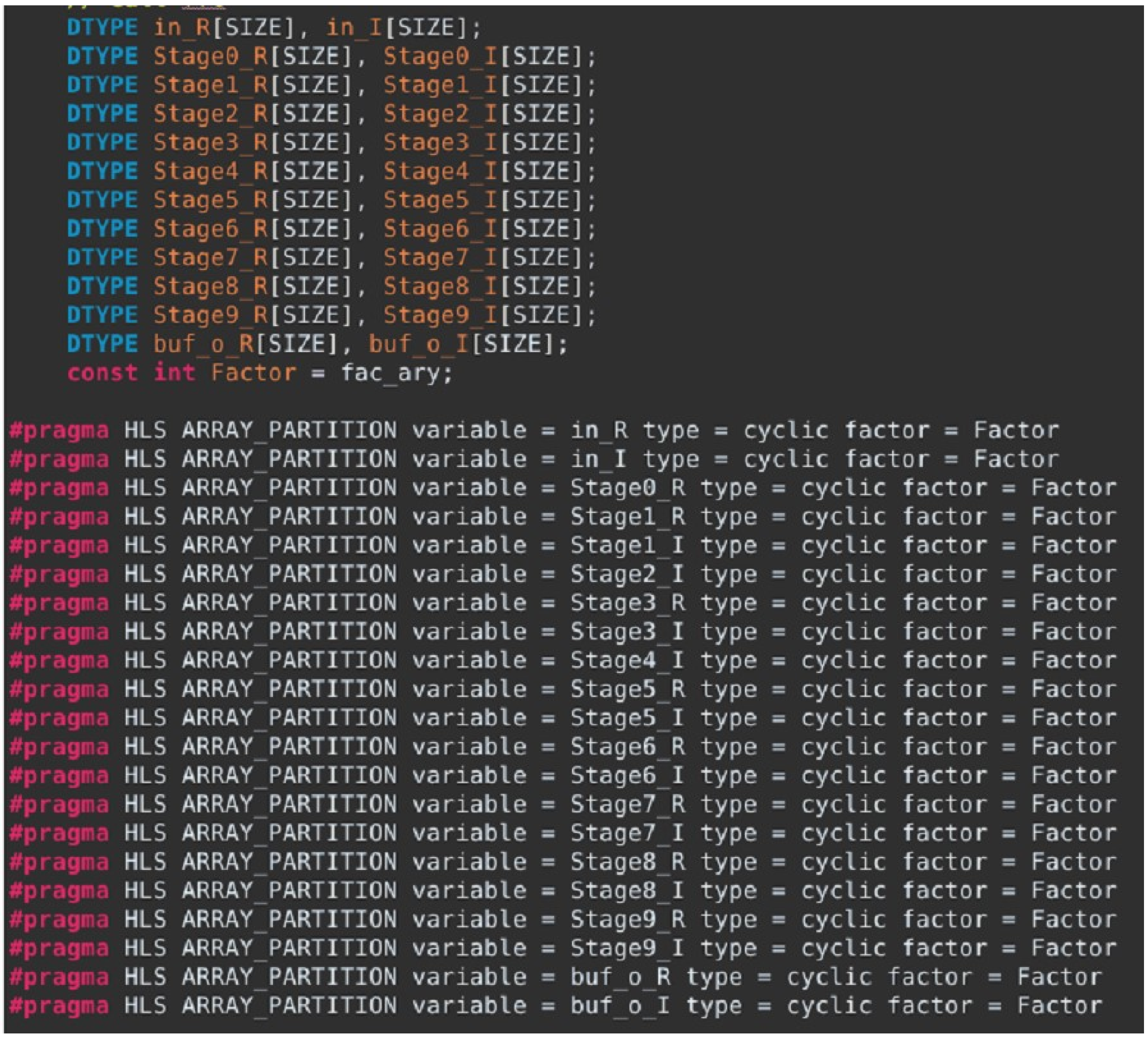


```
const DTYPE W_real[]={1.000000, 0.999981,0.999925,0.999831,0.999699,0.999529,0.999322,0.999078,0.998795,0.998476,0.998
```

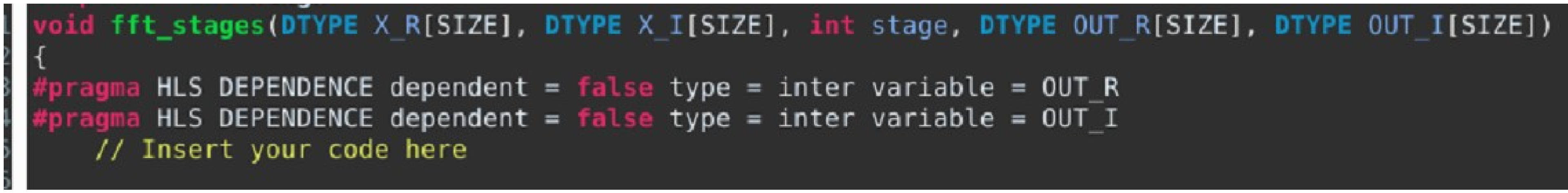


```
const DTYPE W_imag[]={-0.000000,-0.006136,-0.012272,-0.018407,-0.024541,-0.030675,-0.036807,-0.042938,-0.049068,-0.05519
```

- **Memory partitioning:** #pragma HLS ARRAY_PARTITION is used to partition an array into multiple smaller arrays with more number of ports for read and write operations. This results in improved throughput of the design.

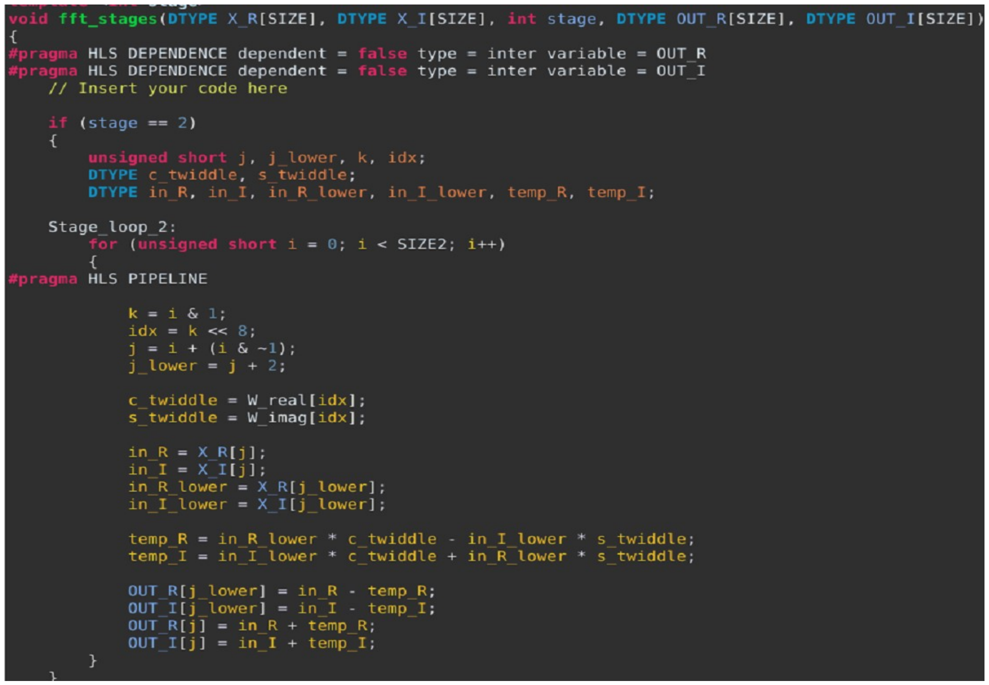


```
    DTYPE in_R[SIZE], in_I[SIZE];
    DTYPE Stage0_R[SIZE], Stage0_I[SIZE];
    DTYPE Stage1_R[SIZE], Stage1_I[SIZE];
    DTYPE Stage2_R[SIZE], Stage2_I[SIZE];
    DTYPE Stage3_R[SIZE], Stage3_I[SIZE];
    DTYPE Stage4_R[SIZE], Stage4_I[SIZE];
    DTYPE Stage5_R[SIZE], Stage5_I[SIZE];
    DTYPE Stage6_R[SIZE], Stage6_I[SIZE];
    DTYPE Stage7_R[SIZE], Stage7_I[SIZE];
    DTYPE Stage8_R[SIZE], Stage8_I[SIZE];
    DTYPE Stage9_R[SIZE], Stage9_I[SIZE];
    DTYPE buf_o_R[SIZE], buf_o_I[SIZE];
    const int Factor = fac_ary;

#pragma HLS ARRAY_PARTITION variable = in_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = in_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage0_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage1_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage1_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage2_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage3_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage3_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage4_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage5_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage5_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage6_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage6_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage7_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage7_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage8_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage8_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage9_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = Stage9_I type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = buf_o_R type = cyclic factor = Factor
#pragma HLS ARRAY_PARTITION variable = buf_o_I type = cyclic factor = Factor
```

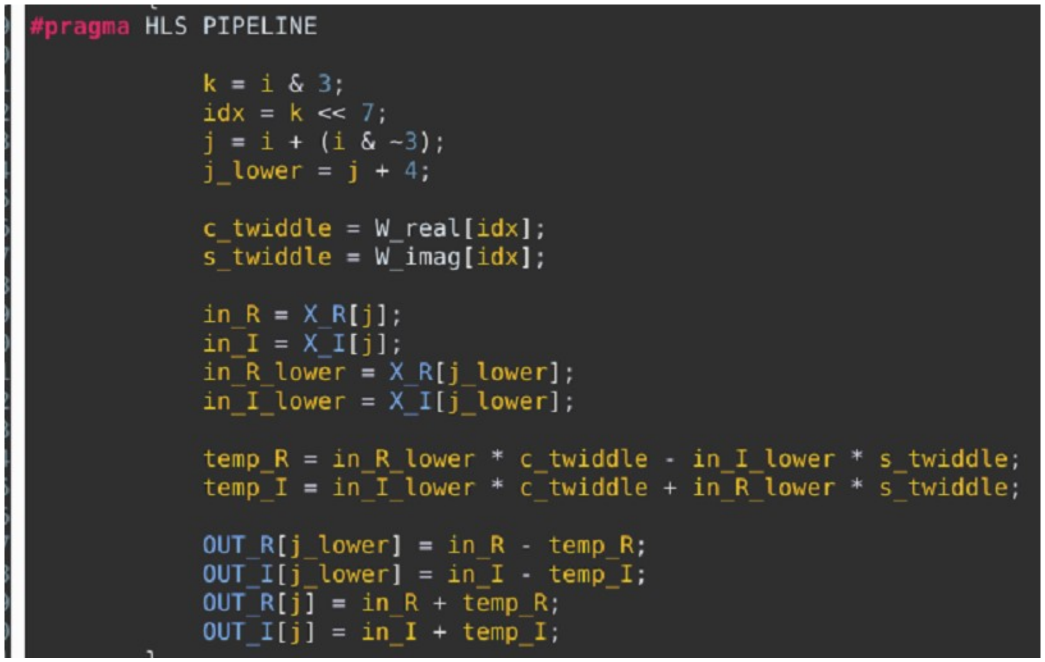


```
void fft_stages(DTYPE X_R[SIZE], DTYPE X_I[SIZE], int stage, DTYPE OUT_R[SIZE], DTYPE OUT_I[SIZE])
{
#pragma HLS DEPENDENCE dependent = false type = inter variable = OUT_R
#pragma HLS DEPENDENCE dependent = false type = inter variable = OUT_I
    // Insert your code here
```

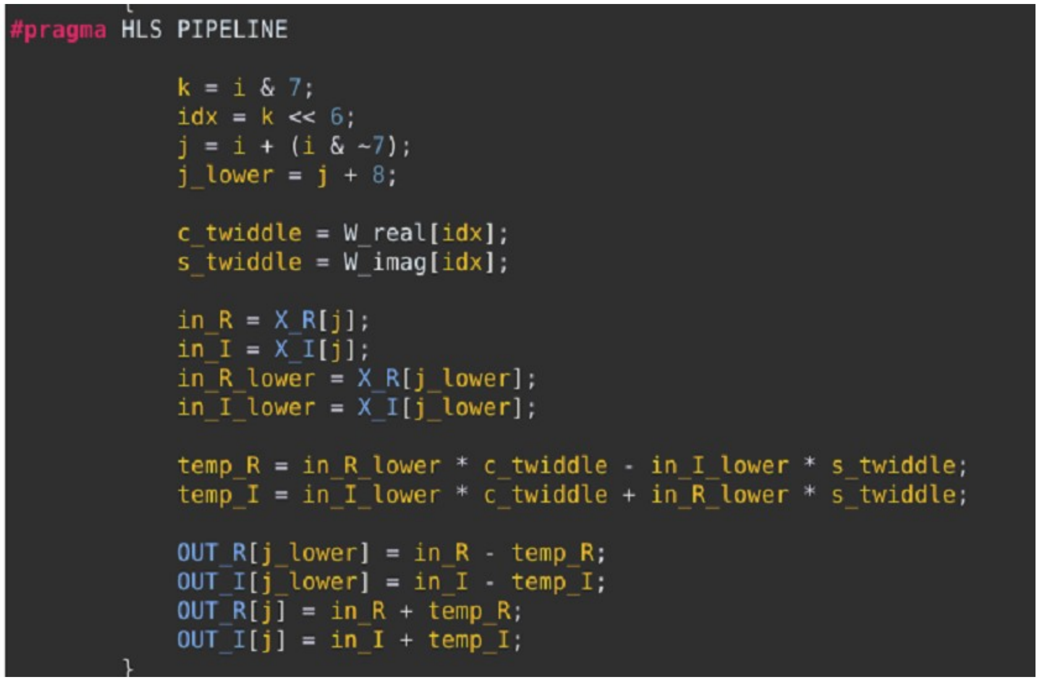
- **Pipelining**
  `#pragma HLS pipeline`



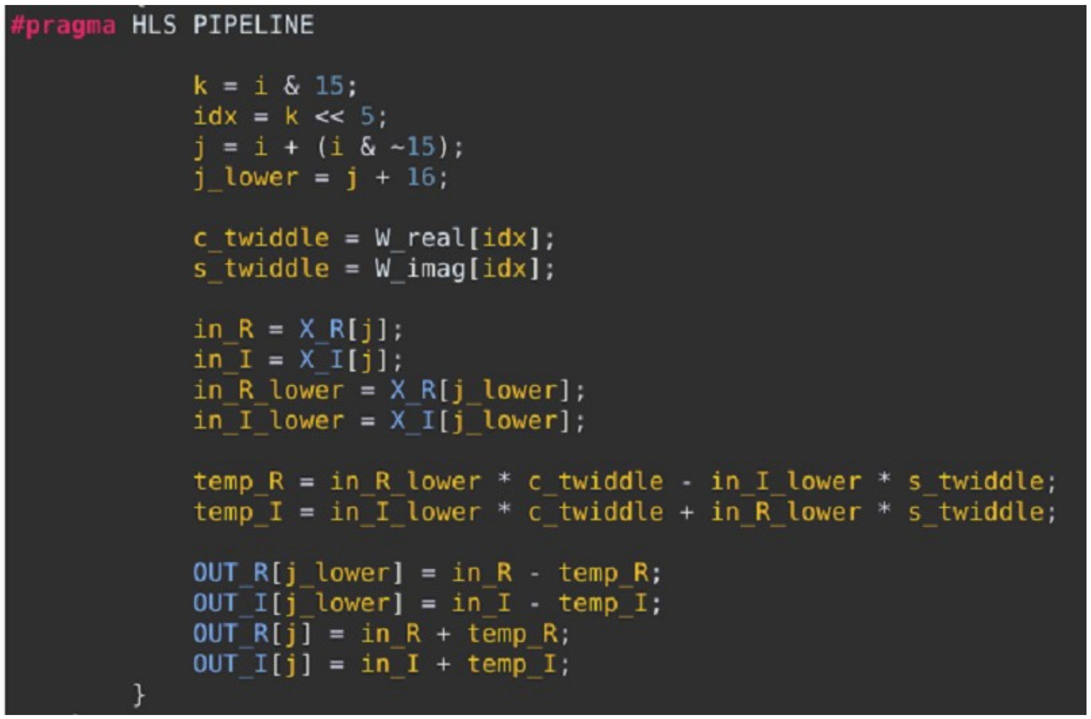
```
void fft_stages(DTYPE X_R[SIZE], DTYPE X_I[SIZE], int stage, DTYPE OUT_R[SIZE], DTYPE OUT_I[SIZE])
{
#pragma HLS DEPENDENCE dependent = false type = inter variable = OUT_R
#pragma HLS DEPENDENCE dependent = false type = inter variable = OUT_I
    // Insert your code here

    if (stage == 2)
    {
        unsigned short j, j_lower, k, idx;
        DTYPE c_twiddle, s_twiddle;
        DTYPE in_R, in_I, in_R_lower, in_I_lower, temp_R, temp_I;

    Stage_loop_2:
        for (unsigned short i = 0; i < SIZE2; i++)
        {
#pragma HLS PIPELINE

            k = i & 1;
            idx = k << 8;
            j = i + (i & ~1);
            j_lower = j + 2;

            c_twiddle = W_real[idx];
            s_twiddle = W_imag[idx];

            in_R = X_R[j];
            in_I = X_I[j];
            in_R_lower = X_R[j_lower];
            in_I_lower = X_I[j_lower];

            temp_R = in_R_lower * c_twiddle - in_I_lower * s_twiddle;
            temp_I = in_I_lower * c_twiddle + in_R_lower * s_twiddle;

            OUT_R[j_lower] = in_R - temp_R;
            OUT_I[j_lower] = in_I - temp_I;
            OUT_R[j] = in_R + temp_R;
            OUT_I[j] = in_I + temp_I;
        }
    }
```

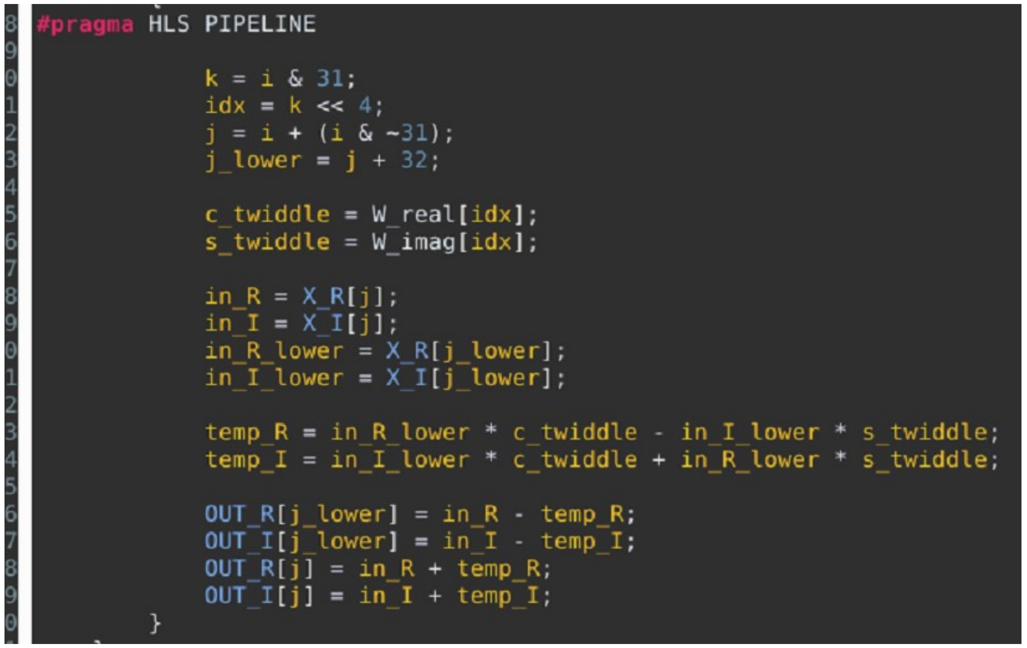


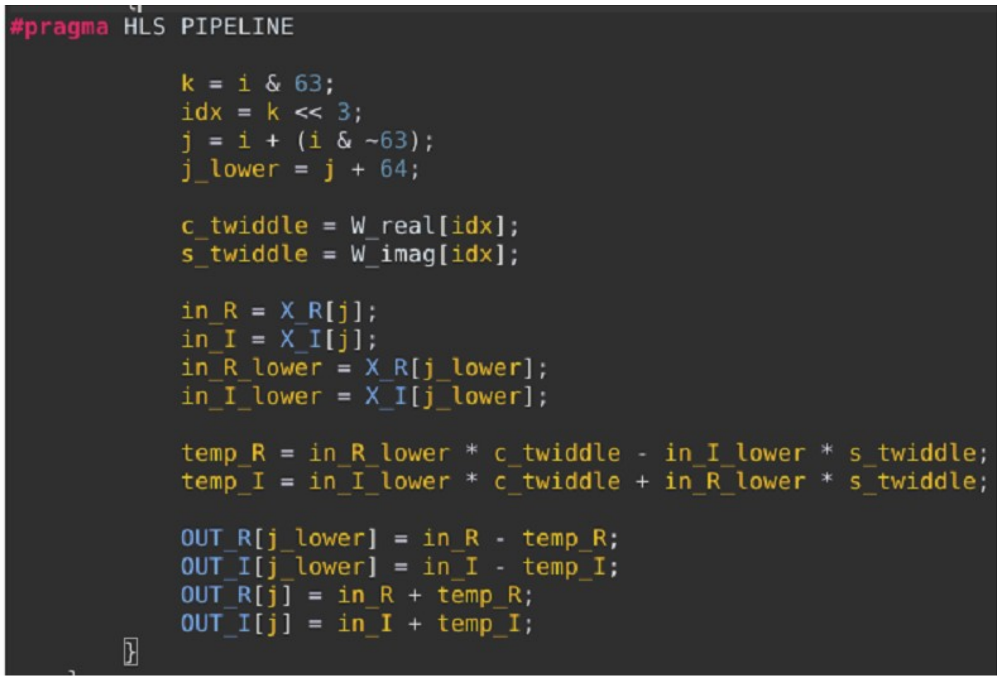
```
#pragma HLS PIPELINE

            k = i & 3;
            idx = k << 7;
            j = i + (i & ~3);
            j_lower = j + 4;

            c_twiddle = W_real[idx];
            s_twiddle = W_imag[idx];

            in_R = X_R[j];
            in_I = X_I[j];
            in_R_lower = X_R[j_lower];
            in_I_lower = X_I[j_lower];

            temp_R = in_R_lower * c_twiddle - in_I_lower * s_twiddle;
            temp_I = in_I_lower * c_twiddle + in_R_lower * s_twiddle;

            OUT_R[j_lower] = in_R - temp_R;
            OUT_I[j_lower] = in_I - temp_I;
            OUT_R[j] = in_R + temp_R;
            OUT_I[j] = in_I + temp_I;
```



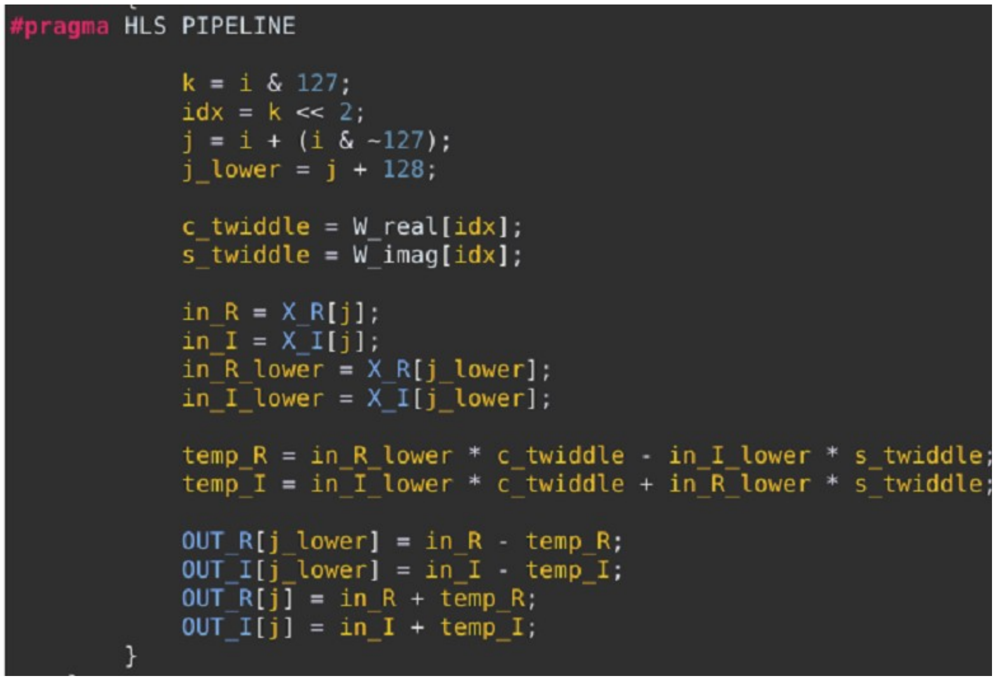
```
#pragma HLS PIPELINE

            k = i & 7;
            idx = k << 6;
            j = i + (i & ~7);
            j_lower = j + 8;

            c_twiddle = W_real[idx];
            s_twiddle = W_imag[idx];

            in_R = X_R[j];
            in_I = X_I[j];
            in_R_lower = X_R[j_lower];
            in_I_lower = X_I[j_lower];

            temp_R = in_R_lower * c_twiddle - in_I_lower * s_twiddle;
            temp_I = in_I_lower * c_twiddle + in_R_lower * s_twiddle;

            OUT_R[j_lower] = in_R - temp_R;
            OUT_I[j_lower] = in_I - temp_I;
            OUT_R[j] = in_R + temp_R;
            OUT_I[j] = in_I + temp_I;
        }
```


```
#pragma HLS PIPELINE

        k = i & 15;
        idx = k << 5;
        j = i + (i & ~15);
        j_lower = j + 16;

        c_twiddle = W_real[idx];
        s_twiddle = W_imag[idx];

        in_R = X_R[j];
        in_I = X_I[j];
        in_R_lower = X_R[j_lower];
        in_I_lower = X_I[j_lower];

        temp_R = in_R_lower * c_twiddle - in_I_lower * s_twiddle;
        temp_I = in_I_lower * c_twiddle + in_R_lower * s_twiddle;

        OUT_R[j_lower] = in_R - temp_R;
        OUT_I[j_lower] = in_I - temp_I;
        OUT_R[j] = in_R + temp_R;
        OUT_I[j] = in_I + temp_I;
    }
```

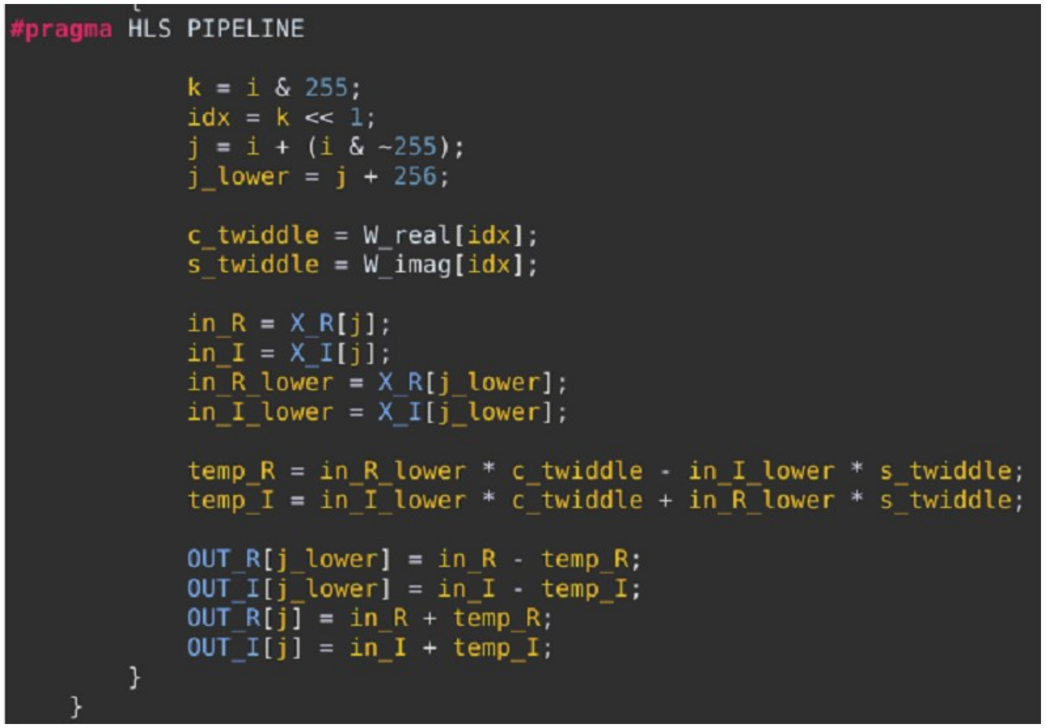

```
#pragma HLS PIPELINE

        k = i & 31;
        idx = k << 4;
        j = i + (i & ~31);
        j_lower = j + 32;

        c_twiddle = W_real[idx];
        s_twiddle = W_imag[idx];

        in_R = X_R[j];
        in_I = X_I[j];
        in_R_lower = X_R[j_lower];
        in_I_lower = X_I[j_lower];

        temp_R = in_R_lower * c_twiddle - in_I_lower * s_twiddle;
        temp_I = in_I_lower * c_twiddle + in_R_lower * s_twiddle;

        OUT_R[j_lower] = in_R - temp_R;
        OUT_I[j_lower] = in_I - temp_I;
        OUT_R[j] = in_R + temp_R;
        OUT_I[j] = in_I + temp_I;
    }
```



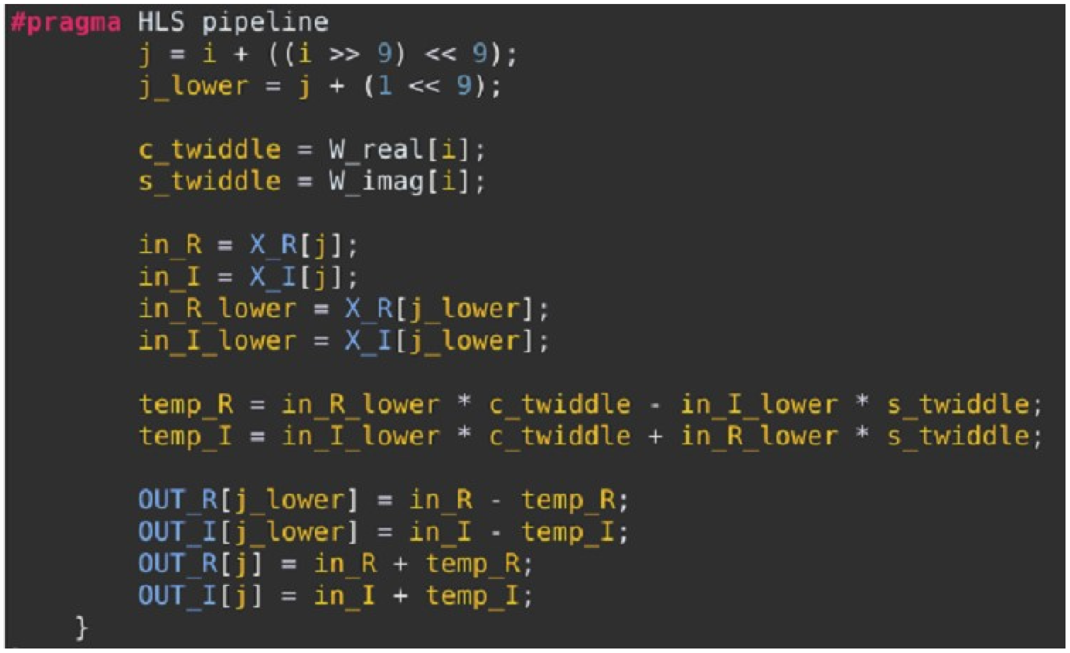
```
#pragma HLS PIPELINE

            k = i & 63;
            idx = k << 3;
            j = i + (i & ~63);
            j_lower = j + 64;

            c_twiddle = W_real[idx];
            s_twiddle = W_imag[idx];

            in_R = X_R[j];
            in_I = X_I[j];
            in_R_lower = X_R[j_lower];
            in_I_lower = X_I[j_lower];

            temp_R = in_R_lower * c_twiddle - in_I_lower * s_twiddle;
            temp_I = in_I_lower * c_twiddle + in_R_lower * s_twiddle;

            OUT_R[j_lower] = in_R - temp_R;
            OUT_I[j_lower] = in_I - temp_I;
            OUT_R[j] = in_R + temp_R;
            OUT_I[j] = in_I + temp_I;
        }
```



```
#pragma HLS PIPELINE

            k = i & 127;
            idx = k << 2;
            j = i + (i & ~127);
            j_lower = j + 128;

            c_twiddle = W_real[idx];
            s_twiddle = W_imag[idx];

            in_R = X_R[j];
            in_I = X_I[j];
            in_R_lower = X_R[j_lower];
            in_I_lower = X_I[j_lower];

            temp_R = in_R_lower * c_twiddle - in_I_lower * s_twiddle;
            temp_I = in_I_lower * c_twiddle + in_R_lower * s_twiddle;

            OUT_R[j_lower] = in_R - temp_R;
            OUT_I[j_lower] = in_I - temp_I;
            OUT_R[j] = in_R + temp_R;
            OUT_I[j] = in_I + temp_I;
        }
```



```
#pragma HLS PIPELINE

            k = i & 255;
            idx = k << 1;
            j = i + (i & ~255);
            j_lower = j + 256;

            c_twiddle = W_real[idx];
            s_twiddle = W_imag[idx];

            in_R = X_R[j];
            in_I = X_I[j];
            in_R_lower = X_R[j_lower];
            in_I_lower = X_I[j_lower];

            temp_R = in_R_lower * c_twiddle - in_I_lower * s_twiddle;
            temp_I = in_I_lower * c_twiddle + in_R_lower * s_twiddle;

            OUT_R[j_lower] = in_R - temp_R;
            OUT_I[j_lower] = in_I - temp_I;
            OUT_R[j] = in_R + temp_R;
            OUT_I[j] = in_I + temp_I;
        }
    }
```

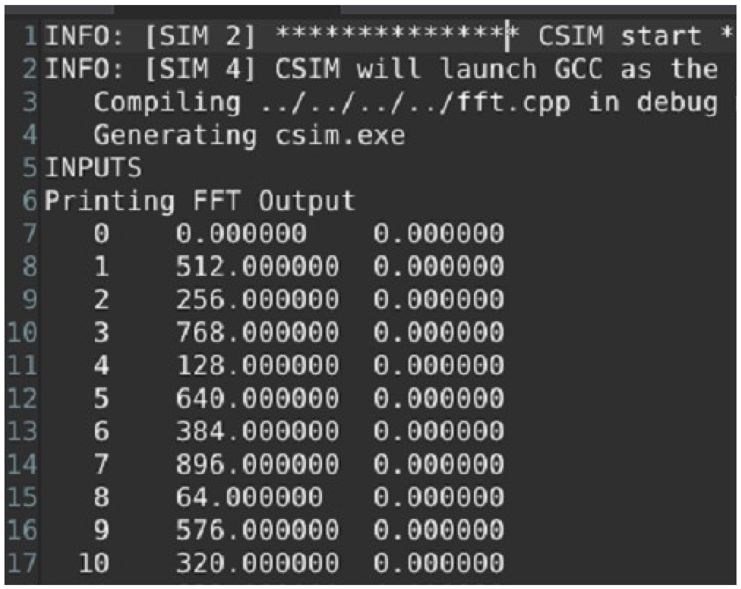


```
#pragma HLS pipeline
        j = i + ((i >> 9) << 9);
        j_lower = j + (1 << 9);

        c_twiddle = W_real[i];
        s_twiddle = W_imag[i];

        in_R = X_R[j];
        in_I = X_I[j];
        in_R_lower = X_R[j_lower];
        in_I_lower = X_I[j_lower];

        temp_R = in_R_lower * c_twiddle - in_I_lower * s_twiddle;
        temp_I = in_I_lower * c_twiddle + in_R_lower * s_twiddle;

        OUT_R[j_lower] = in_R - temp_R;
        OUT_I[j_lower] = in_I - temp_I;
        OUT_R[j] = in_R + temp_R;
        OUT_I[j] = in_I + temp_I;
    }
```

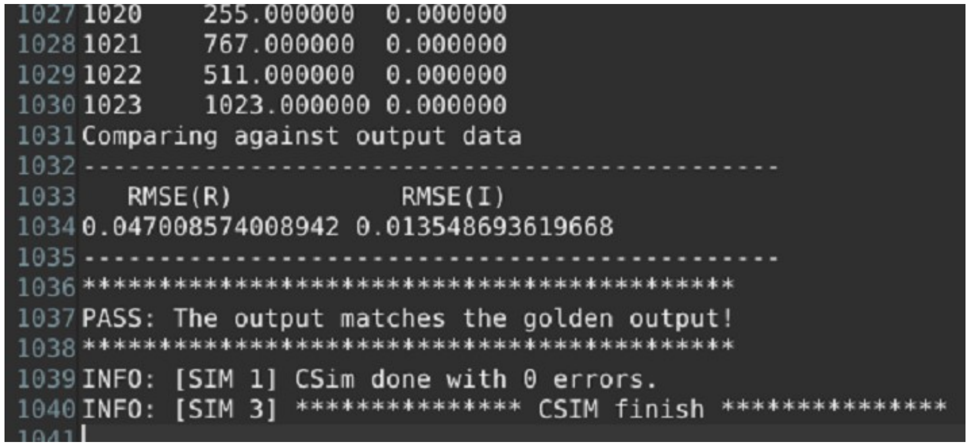
- **Pragma HLS inline:** Removes a function as a separate entity in the hierarchy. After inlining, the function is dissolved into the calling function and no longer appears as a separate level of hierarchy in the RTL. In some cases, inlining a function allows operations within the function to be shared and optimized more effectively with the calling function. However, an inlined function cannot be shared or reused, so if the parent function calls the inlined function multiple times, this can increase the area required for implementing the RTL.

```
#pragma HLS inline region
It was removed, because it was causing II Violation
```

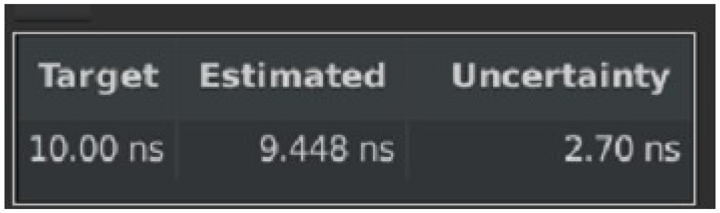
TEST BENCH : PASS



```
INFO: [SIM 2] *************** CSIM start *
INFO: [SIM 4] CSIM will launch GCC as the
   Compiling ../../../../fft.cpp in debug
   Generating csim.exe
INPUTS
Printing FFT Output
   0     0.000000    0.000000
   1     512.000000  0.000000
   2     256.000000  0.000000
   3     768.000000  0.000000
   4     128.000000  0.000000
   5     640.000000  0.000000
   6     384.000000  0.000000
   7     896.000000  0.000000
   8     64.000000   0.000000
   9     576.000000  0.000000
  10     320.000000  0.000000
```

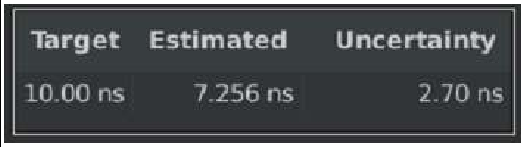


```
1020    255.000000  0.000000
1021    767.000000  0.000000
1022    511.000000  0.000000
1023    1023.000000 0.000000
Comparing against output data
-------------------------------------------
   RMSE(R)            RMSE(I)
0.047008574008942 0.013548693619668
-------------------------------------------
*******************************************
PASS: The output matches the golden output!
*******************************************
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

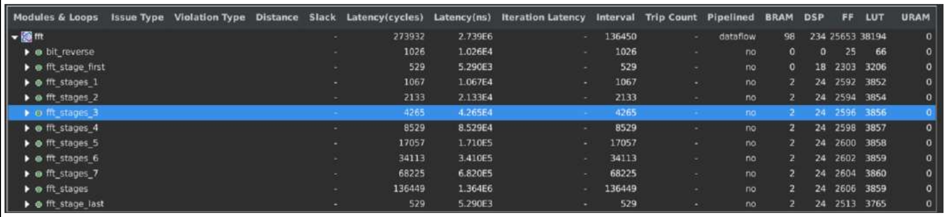


Target	Estimated	Uncertainty
10.00 ns	9.448 ns	2.70 ns

fft1024_Baseline

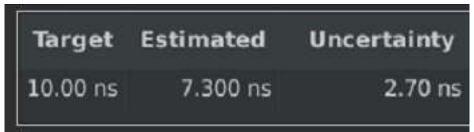


Target	Estimated	Uncertainty
10.00 ns	7.256 ns	2.70 ns

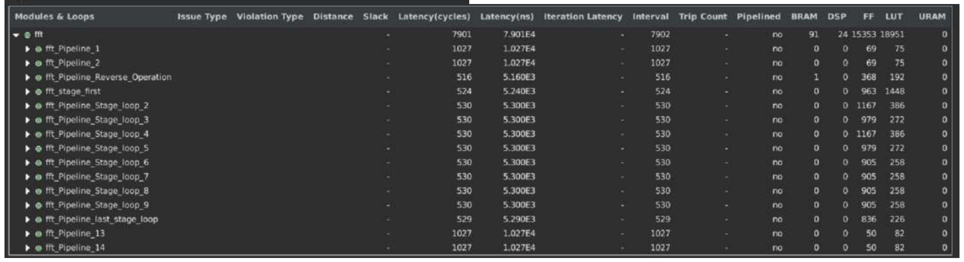


Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
fft				-	273932	2.739E6	-	136450	-	dataflow	98	234	25653	38194	0
bit_reverse				-	1026	1.026E4	-	1026	-	no	0	0	25	66	0
fft_stage_first				-	529	5.290E3	-	529	-	no	0	18	2303	3206	0
fft_stages_1				-	1067	1.067E4	-	1067	-	no	2	24	2592	3852	0
fft_stages_2				-	2133	2.133E4	-	2133	-	no	2	24	2594	3854	0
fft_stages_3				-	4265	4.265E4	-	4265	-	no	2	24	2596	3856	0
fft_stages_4				-	8529	8.529E4	-	8529	-	no	2	24	2598	3857	0
fft_stages_5				-	17057	1.710E5	-	17057	-	no	2	24	2600	3858	0
fft_stages_6				-	34113	3.410E5	-	34113	-	no	2	24	2602	3859	0
fft_stages_7				-	68225	6.820E5	-	68225	-	no	2	24	2604	3860	0
fft_stages				-	136449	1.364E6	-	136449	-	no	2	24	2606	3859	0
fft_stage_last				-	529	5.290E3	-	529	-	no	2	24	2513	3765	0

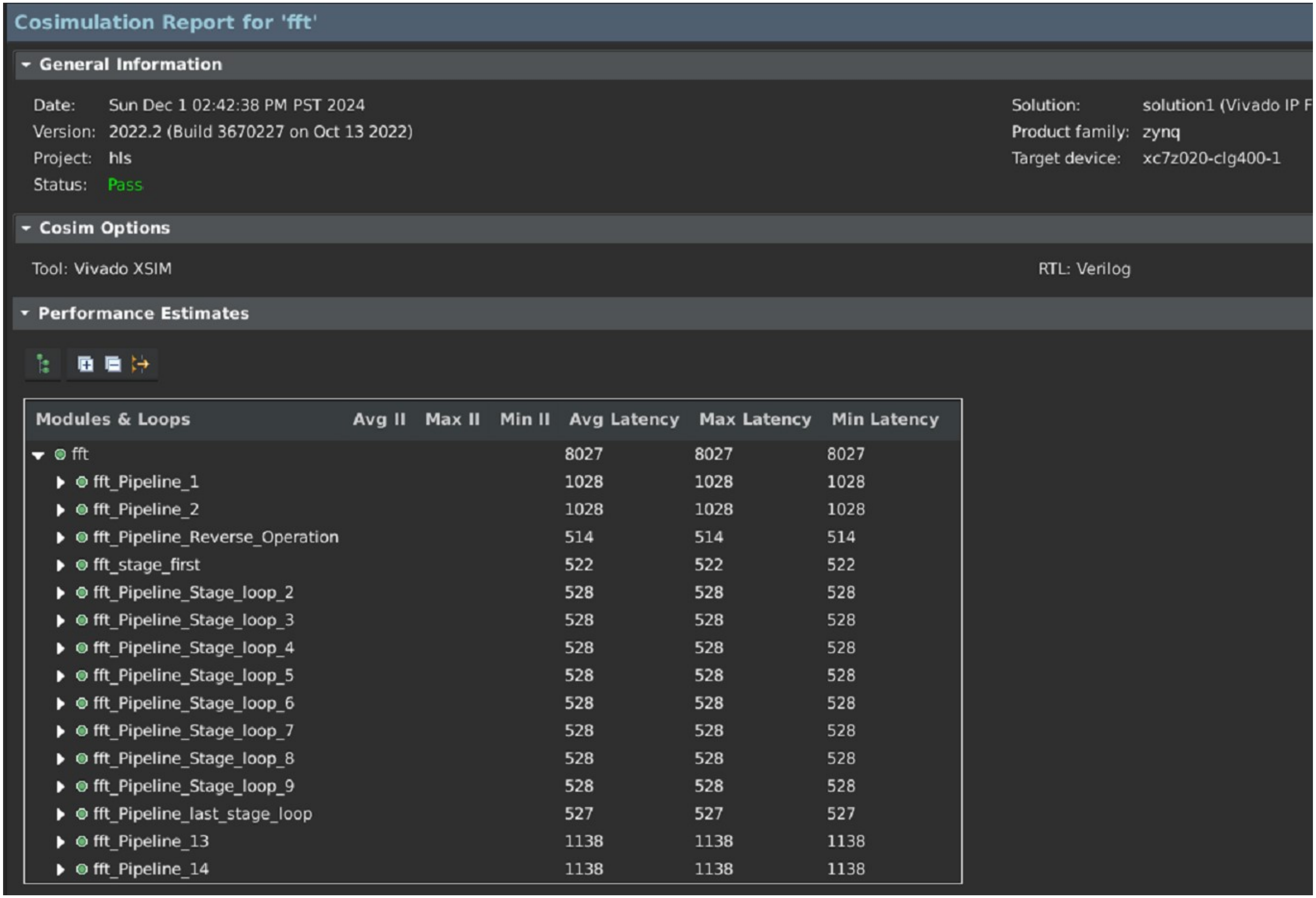
fft1024_Best



Target	Estimated	Uncertainty
10.00 ns	7.300 ns	2.70 ns



Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
fft				-	7901	7.901E4	-	7902	-	no	91	24	15353	18951	0
fft_Pipeline_1				-	1027	1.027E4	-	1027	-	no	0	0	69	75	0
fft_Pipeline_2				-	1027	1.027E4	-	1027	-	no	0	0	69	75	0
fft_Pipeline_Reverse_Operation				-	516	5.160E3	-	516	-	no	1	0	368	192	0
fft_stage_first				-	524	5.240E3	-	524	-	no	0	0	963	1448	0
fft_Pipeline_Stage_loop_2				-	530	5.300E3	-	530	-	no	0	0	1167	386	0
fft_Pipeline_Stage_loop_3				-	530	5.300E3	-	530	-	no	0	0	979	272	0
fft_Pipeline_Stage_loop_4				-	530	5.300E3	-	530	-	no	0	0	1167	386	0
fft_Pipeline_Stage_loop_5				-	530	5.300E3	-	530	-	no	0	0	979	272	0
fft_Pipeline_Stage_loop_6				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_Stage_loop_7				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_Stage_loop_8				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_Stage_loop_9				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_last_stage_loop				-	529	5.290E3	-	529	-	no	0	0	836	226	0
fft_Pipeline_13				-	1027	1.027E4	-	1027	-	no	0	0	50	82	0
fft_Pipeline_14				-	1027	1.027E4	-	1027	-	no	0	0	50	82	0


Cosimulation Report for 'fft'

General Information

Date: Sun Dec 1 02:42:38 PM PST 2024
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: hls
Status: Pass

Solution: solution1 (Vivado IP F
Product family: zynq
Target device: xc7z020-clg400-1

Cosim Options

Tool: Vivado XSIM
RTL: Verilog

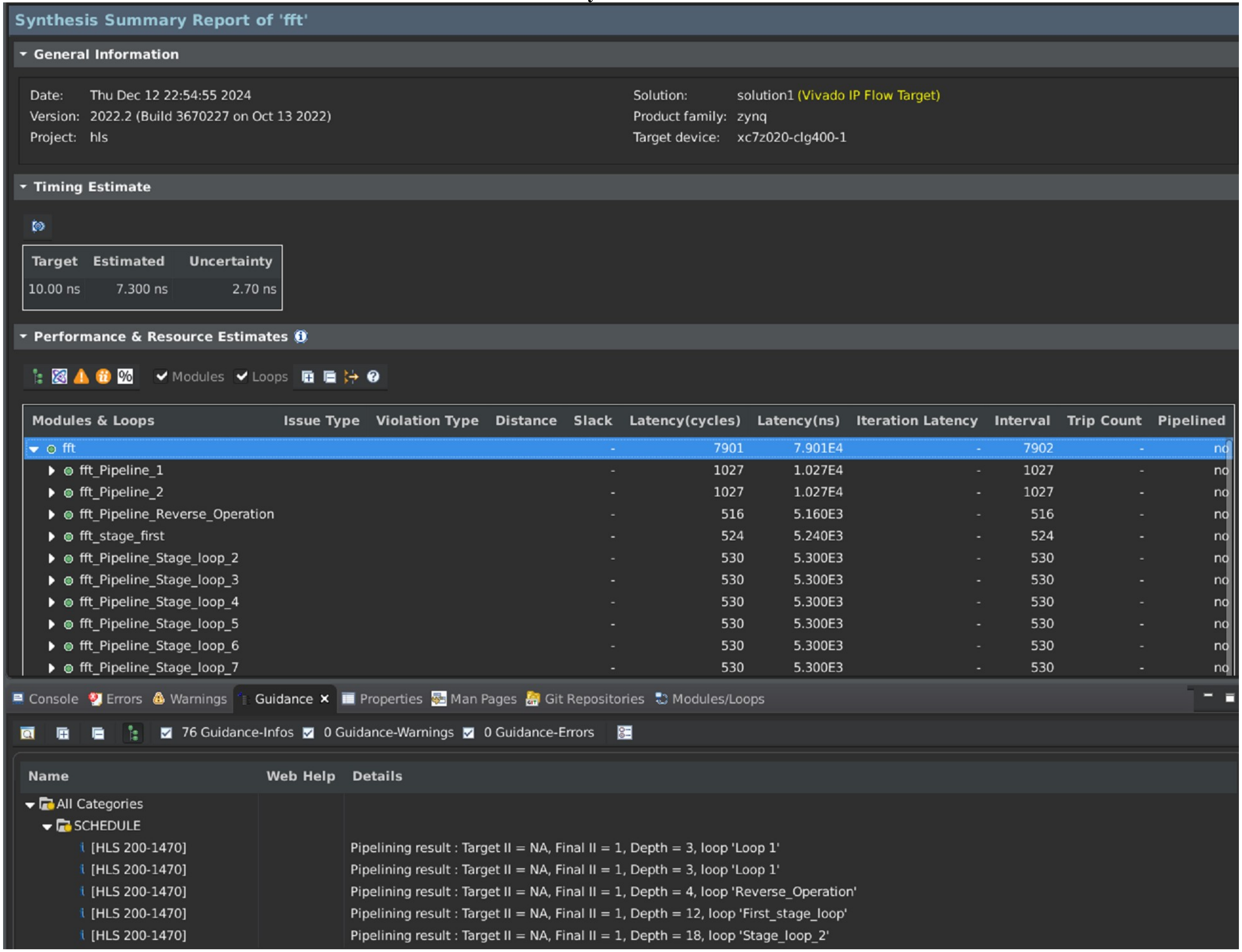
Performance Estimates

Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
fft				8027	8027	8027
fft_Pipeline_1				1028	1028	1028
fft_Pipeline_2				1028	1028	1028
fft_Pipeline_Reverse_Operation				514	514	514
fft_stage_first				522	522	522
fft_Pipeline_Stage_loop_2				528	528	528
fft_Pipeline_Stage_loop_3				528	528	528
fft_Pipeline_Stage_loop_4				528	528	528
fft_Pipeline_Stage_loop_5				528	528	528
fft_Pipeline_Stage_loop_6				528	528	528
fft_Pipeline_Stage_loop_7				528	528	528
fft_Pipeline_Stage_loop_8				528	528	528
fft_Pipeline_Stage_loop_9				528	528	528
fft_Pipeline_last_stage_loop				527	527	527
fft_Pipeline_13				1138	1138	1138
fft_Pipeline_14				1138	1138	1138

INTERFACE DESIGN

```
void fft(DTYPE X_R[SIZE], DTYPE X_I[SIZE], DTYPE OUT_R[SIZE], DTYPE OUT_I[SIZE])
{

#pragma HLS INTERFACE s_axilite port=return bundle=fft

#pragma HLS INTERFACE m_axi depth=1024 port=OUT_R offset=slave bundle=output1
#pragma HLS INTERFACE m_axi depth=1024 port=OUT_I offset=slave bundle=output2

#pragma HLS INTERFACE m_axi depth=1024 port=X_R offset=slave bundle=input1
#pragma HLS INTERFACE m_axi depth=1024 port=X_I offset=slave bundle=input2

#pragma HLS INTERFACE s_axilite port=X_R bundle=fft
#pragma HLS INTERFACE s_axilite port=X_I bundle=fft
#pragma HLS INTERFACE s_axilite port=OUT_R bundle=fft
#pragma HLS INTERFACE s_axilite port=OUT_I bundle=fft
…
```

Synthesis



Synthesis Summary Report of 'fft'

General Information

Date:	Thu Dec 12 22:54:55 2024	Solution:	solution1 (Vivado IP Flow Target)
Version:	2022.2 (Build 3670227 on Oct 13 2022)	Product family:	zynq
Project:	hls	Target device:	xc7z020-clg400-1

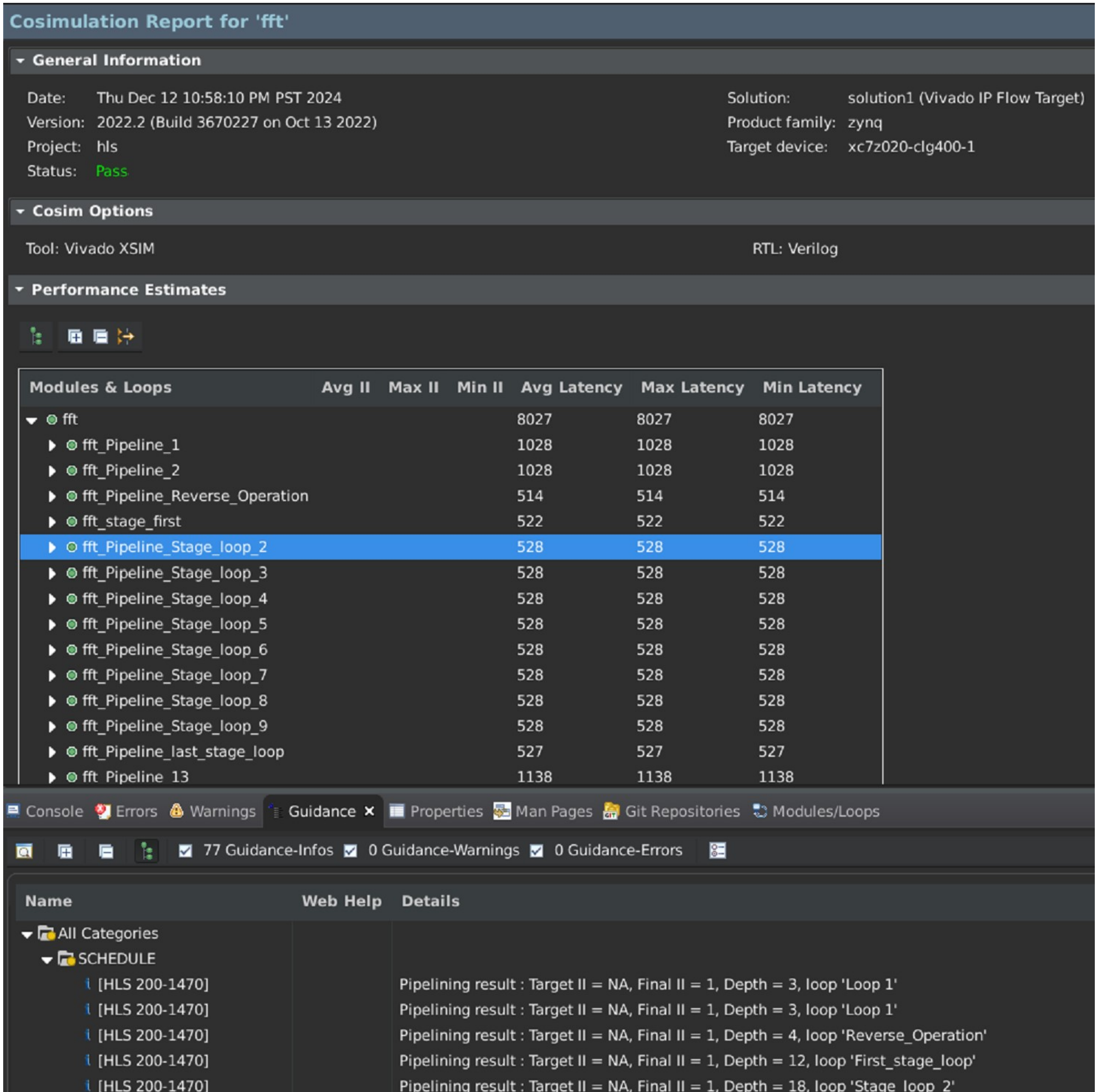
Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.300 ns	2.70 ns

Performance & Resource Estimates

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined
fft				-	7901	7.901E4	-	7902	-	no
fft_Pipeline_1				-	1027	1.027E4	-	1027	-	no
fft_Pipeline_2				-	1027	1.027E4	-	1027	-	no
fft_Pipeline_Reverse_Operation				-	516	5.160E3	-	516	-	no
fft_stage_first				-	524	5.240E3	-	524	-	no
fft_Pipeline_Stage_loop_2				-	530	5.300E3	-	530	-	no
fft_Pipeline_Stage_loop_3				-	530	5.300E3	-	530	-	no
fft_Pipeline_Stage_loop_4				-	530	5.300E3	-	530	-	no
fft_Pipeline_Stage_loop_5				-	530	5.300E3	-	530	-	no
fft_Pipeline_Stage_loop_6				-	530	5.300E3	-	530	-	no
fft_Pipeline_Stage_loop_7				-	530	5.300E3	-	530	-	no

Name	Web Help	Details
All Categories		
SCHEDULE		
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 3, loop 'Loop 1'
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 3, loop 'Loop 1'
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 4, loop 'Reverse_Operation'
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 12, loop 'First_stage_loop'
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 18, loop 'Stage_loop_2'



Cosimulation Report for 'fft'

General Information

Date:	Thu Dec 12 10:58:10 PM PST 2024	Solution:	solution1 (Vivado IP Flow Target)
Version:	2022.2 (Build 3670227 on Oct 13 2022)	Product family:	zynq
Project:	hls	Target device:	xc7z020-clg400-1
Status:	Pass		

Cosim Options

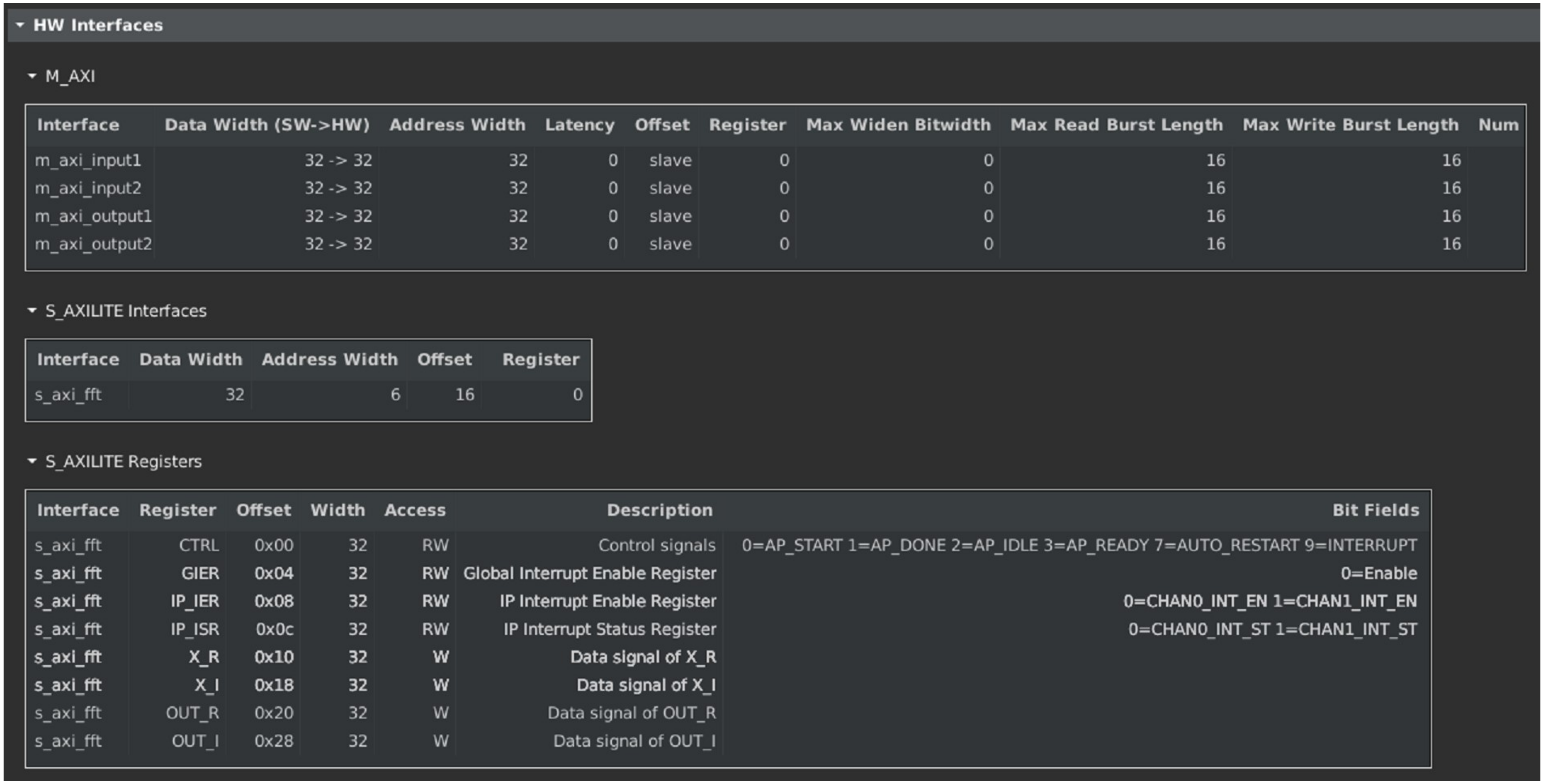
Tool: Vivado XSIM RTL: Verilog

Performance Estimates

Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
fft				8027	8027	8027
fft_Pipeline_1				1028	1028	1028
fft_Pipeline_2				1028	1028	1028
fft_Pipeline_Reverse_Operation				514	514	514
fft_stage_first				522	522	522
fft_Pipeline_Stage_loop_2				528	528	528
fft_Pipeline_Stage_loop_3				528	528	528
fft_Pipeline_Stage_loop_4				528	528	528
fft_Pipeline_Stage_loop_5				528	528	528
fft_Pipeline_Stage_loop_6				528	528	528
fft_Pipeline_Stage_loop_7				528	528	528
fft_Pipeline_Stage_loop_8				528	528	528
fft_Pipeline_Stage_loop_9				528	528	528
fft_Pipeline_last_stage_loop				527	527	527
fft_Pipeline_13				1138	1138	1138

77 Guidance-Infos 0 Guidance-Warnings 0 Guidance-Errors

Name	Web Help	Details
All Categories		
SCHEDULE		
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 3, loop 'Loop 1'
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 3, loop 'Loop 1'
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 4, loop 'Reverse_Operation'
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 12, loop 'First_stage_loop'
[HLS 200-1470]		Pipelining result : Target II = NA, Final II = 1, Depth = 18, loop 'Stage_loop_2'



HW Interfaces

M_AXI

Interface	Data Width (SW->HW)	Address Width	Latency	Offset	Register	Max Widen Bitwidth	Max Read Burst Length	Max Write Burst Length	Num
m_axi_input1	32 -> 32	32	0	slave	0	0	16	16	
m_axi_input2	32 -> 32	32	0	slave	0	0	16	16	
m_axi_output1	32 -> 32	32	0	slave	0	0	16	16	
m_axi_output2	32 -> 32	32	0	slave	0	0	16	16	

S_AXILITE Interfaces

Interface	Data Width	Address Width	Offset	Register
s_axi_fft	32	6	16	0

S_AXILITE Registers

Interface	Register	Offset	Width	Access	Description	Bit Fields
s_axi_fft	CTRL	0x00	32	RW	Control signals	0=AP_START 1=AP_DONE 2=AP_IDLE 3=AP_READY 7=AUTO_RESTART 9=INTERRUPT
s_axi_fft	GIER	0x04	32	RW	Global Interrupt Enable Register	0=Enable
s_axi_fft	IP_IER	0x08	32	RW	IP Interrupt Enable Register	0=CHAN0_INT_EN 1=CHAN1_INT_EN
s_axi_fft	IP_ISR	0x0c	32	RW	IP Interrupt Status Register	0=CHAN0_INT_ST 1=CHAN1_INT_ST
s_axi_fft	X_R	0x10	32	W	Data signal of X_R	
s_axi_fft	X_I	0x18	32	W	Data signal of X_I	
s_axi_fft	OUT_R	0x20	32	W	Data signal of OUT_R	
s_axi_fft	OUT_I	0x28	32	W	Data signal of OUT_I	

VIVADO

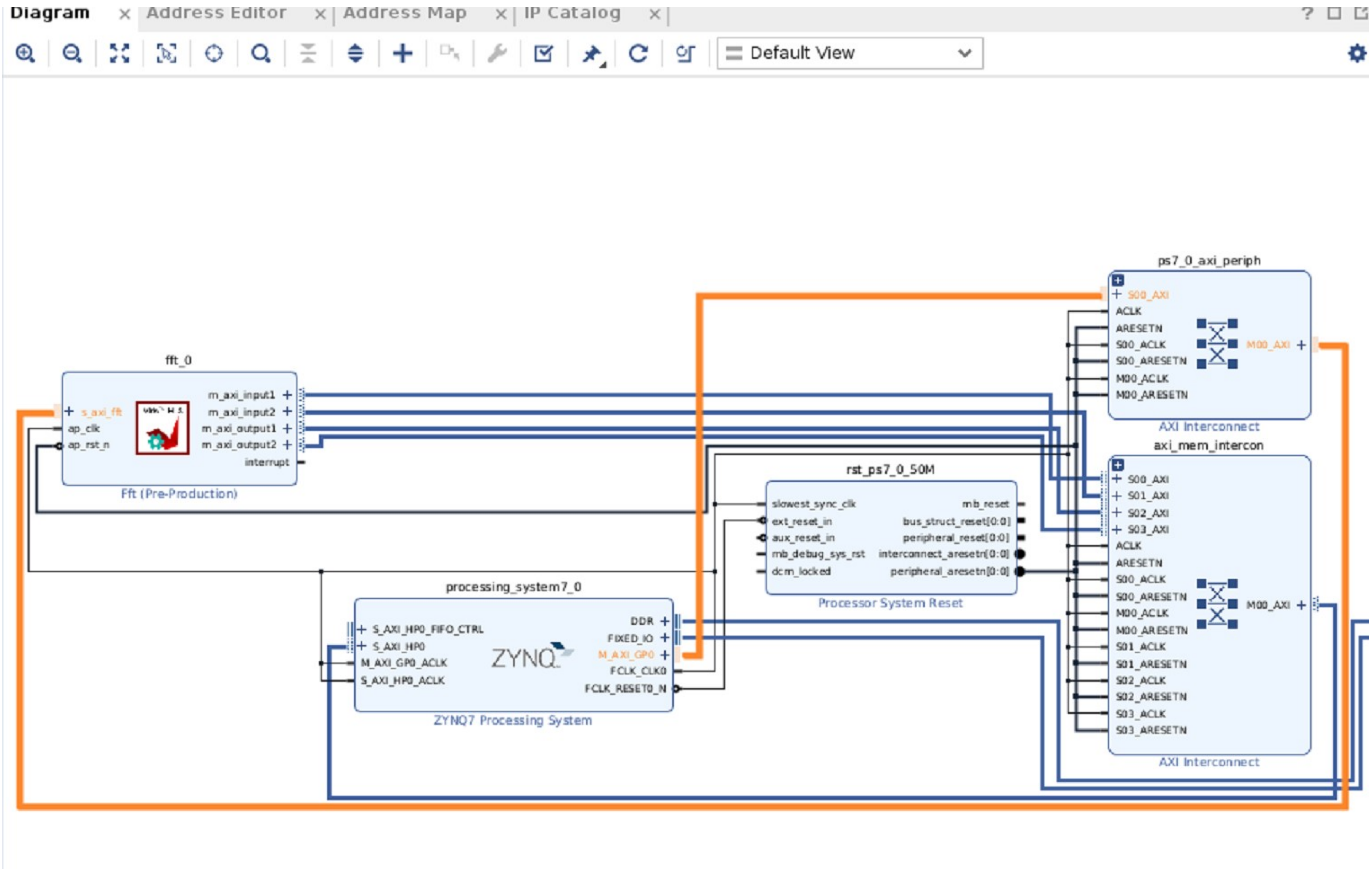
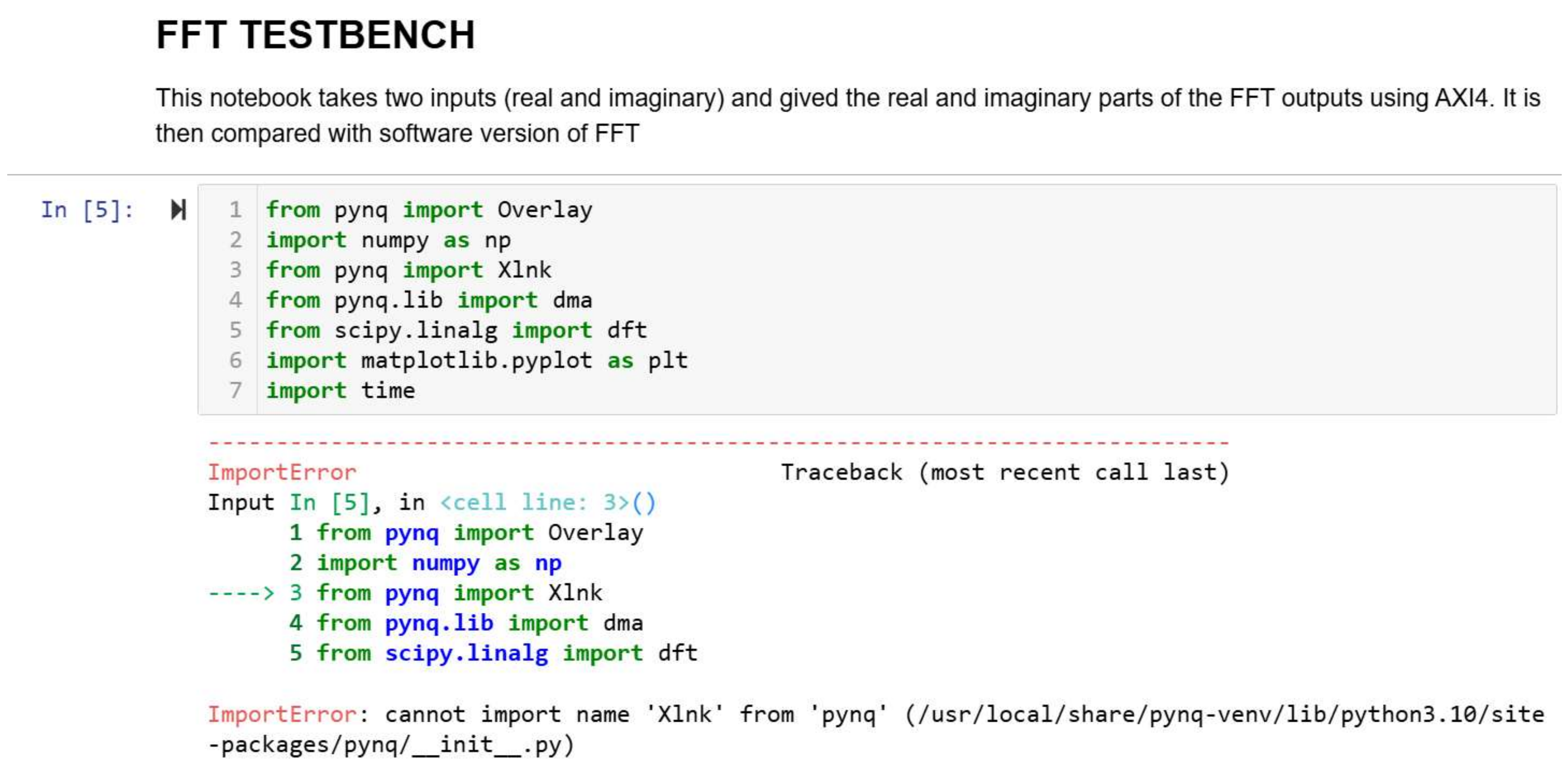


Diagram
Address Editor
Address Map
IP Catalog
Default View
ps7_0_axi_periph
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M00_AXI
AXI Interconnect
fft_0
s_axi_fft
ap_clk
ap_rst_n
m_axi_input1
m_axi_input2
m_axi_output1
m_axi_output2
interrupt
Fft (Pre-Production)
axi_mem_intercon
S00_AXI
S01_AXI
S02_AXI
S03_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
S01_ACLK
S01_ARESETN
S02_ACLK
S02_ARESETN
S03_ACLK
S03_ARESETN
M00_AXI
AXI Interconnect
rst_ps7_0_50M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset
processing_system7_0
S_AXI_HP0_FIFO_CTRL
S_AXI_HP0
M_AXI_GP0_ACLK
S_AXI_HP0_ACLK
ZYNQ
DDR
FIXED_IO
M_AXI_GP0
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System



FFT TESTBENCH

This notebook takes two inputs (real and imaginary) and gived the real and imaginary parts of the FFT outputs using AXI4. It is then compared with software version of FFT

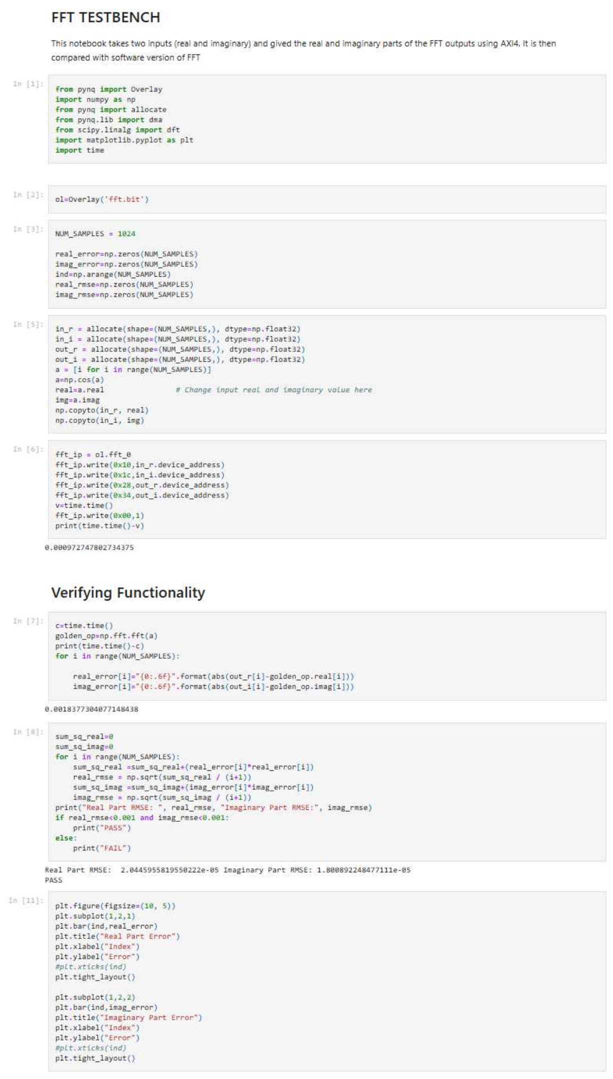
In [5]:

```python
from pynq import Overlay
import numpy as np
from pynq import Xlnk
from pynq.lib import dma
from scipy.linalg import dft
import matplotlib.pyplot as plt
import time
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Input In [5], in <cell line: 3>()
      1 from pynq import Overlay
      2 import numpy as np
----> 3 from pynq import Xlnk
      4 from pynq.lib import dma
      5 from scipy.linalg import dft

ImportError: cannot import name 'Xlnk' from 'pynq' (/usr/local/share/pynq-venv/lib/python3.10/site
-packages/pynq/__init__.py)
```

I had problems with **Xln**, so I had to implement **Allocate** instead

fft1024_Best DEMO



FFT TESTBENCH

This notebook takes two inputs (real and imaginary) and gived the real and imaginary parts of the FFT outputs using AXI4. It is then compared with software version of FFT

```
In [1]:
from pynq import Overlay
import numpy as np
from pynq import allocate
from pynq.lib import dma
from scipy.linalg import dft
import matplotlib.pyplot as plt
import time
```

```
In [2]:
ol=Overlay('fft.bit')
```

```
In [3]:
NUM_SAMPLES = 1024

real_error=np.zeros(NUM_SAMPLES)
imag_error=np.zeros(NUM_SAMPLES)
ind=np.arange(NUM_SAMPLES)
real_rmse=np.zeros(NUM_SAMPLES)
imag_rmse=np.zeros(NUM_SAMPLES)
```

```
In [5]:
in_r = allocate(shape=(NUM_SAMPLES,), dtype=np.float32)
in_i = allocate(shape=(NUM_SAMPLES,), dtype=np.float32)
out_r = allocate(shape=(NUM_SAMPLES,), dtype=np.float32)
out_i = allocate(shape=(NUM_SAMPLES,), dtype=np.float32)
a = [i for i in range(NUM_SAMPLES)]
a=np.cos(a)
real=a.real                # Change input real and imaginary value here
img=a.imag
np.copyto(in_r, real)
np.copyto(in_i, img)
```

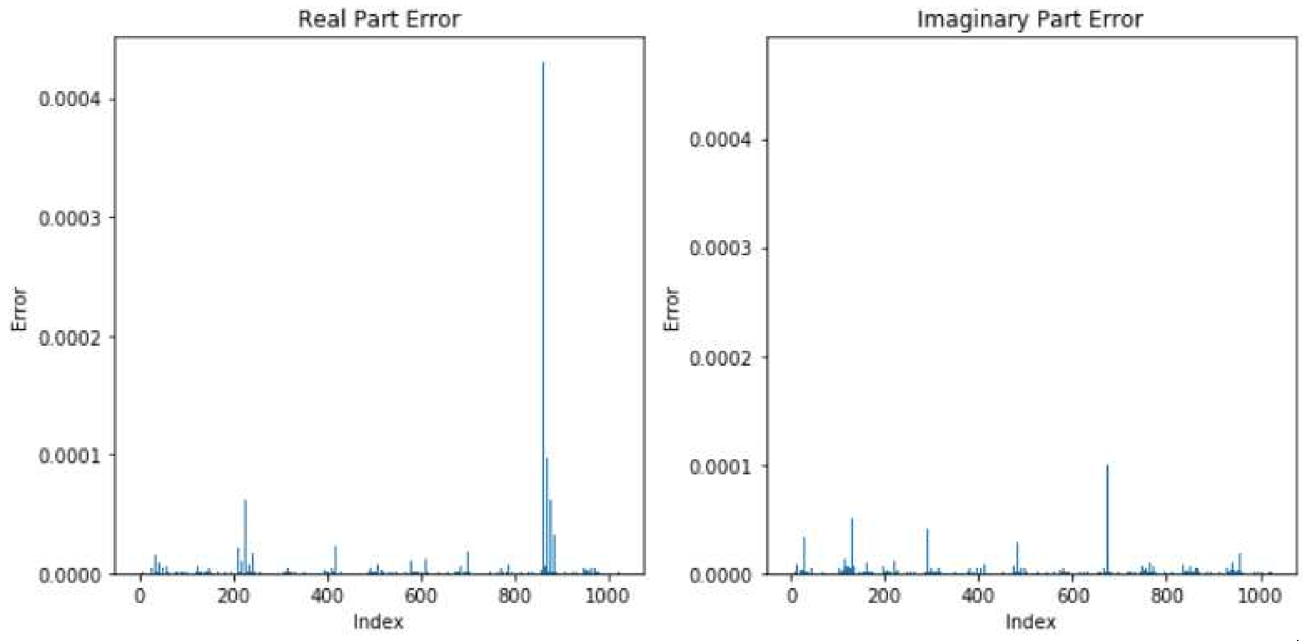
```
In [6]:
fft_ip = ol.fft_0
fft_ip.write(0x10,in_r.device_address)
fft_ip.write(0x1c,in_i.device_address)
fft_ip.write(0x28,out_r.device_address)
fft_ip.write(0x34,out_i.device_address)
v=time.time()
fft_ip.write(0x00,1)
print(time.time()-v)

0.000972747802734375
```

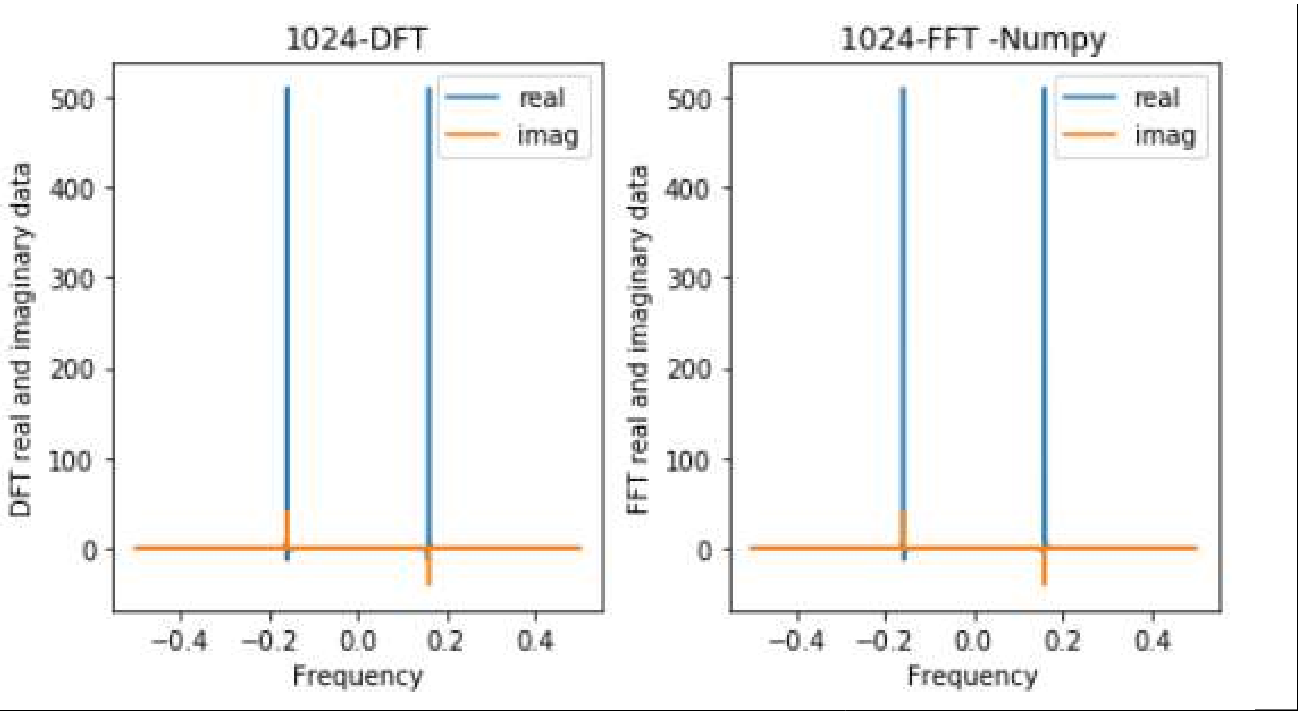
Verifying Functionality

```
In [7]:
c=time.time()
golden_op=np.fft.fft(a)
print(time.time()-c)
for i in range(NUM_SAMPLES):

    real_error[i]="{0:.6f}".format(abs(out_r[i]-golden_op.real[i]))
    imag_error[i]="{0:.6f}".format(abs(out_i[i]-golden_op.imag[i]))

0.0018377304077148438
```

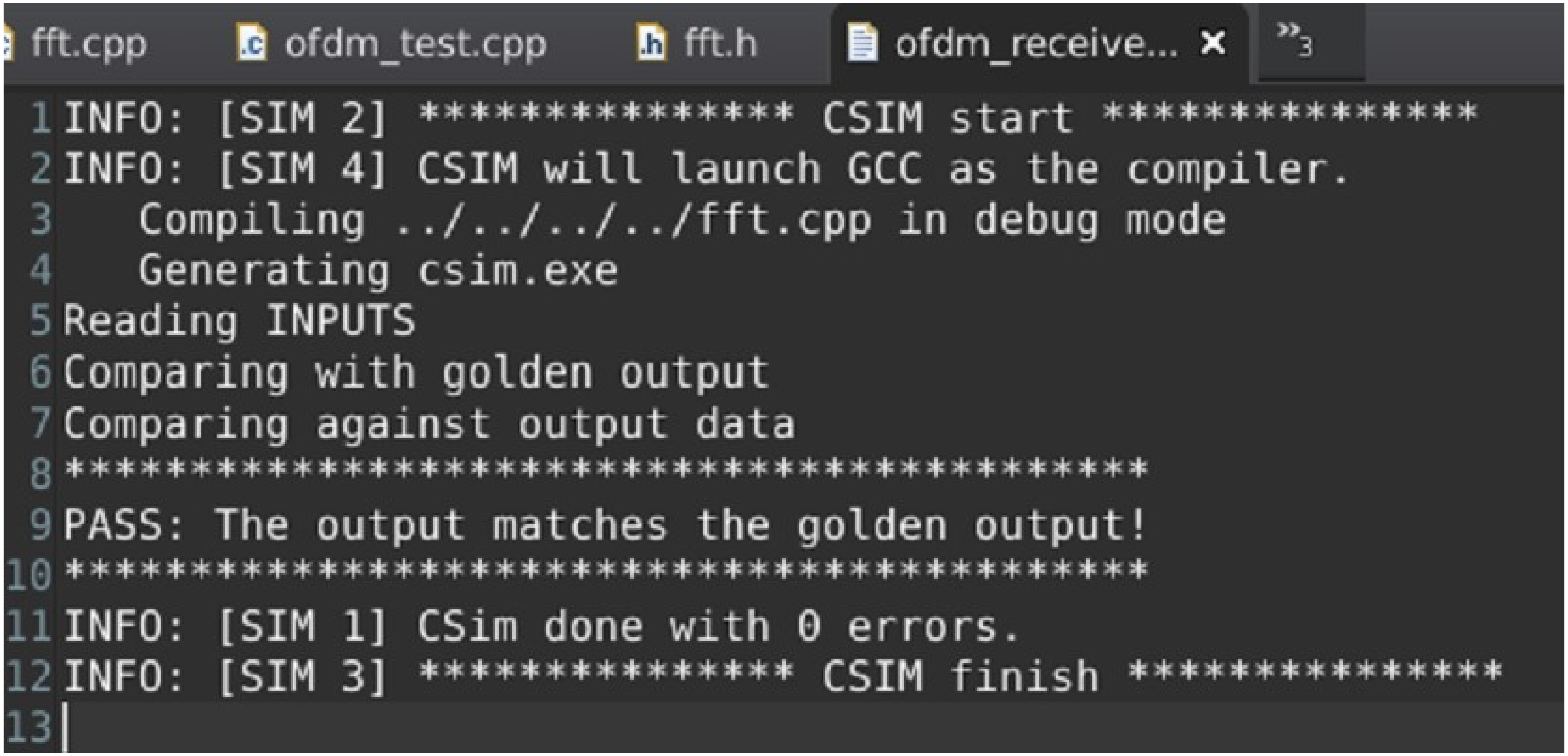
```
In [8]:
sum_sq_real=0
sum_sq_imag=0
for i in range(NUM_SAMPLES):
    sum_sq_real =sum_sq_real+(real_error[i]*real_error[i])
    real_rmse = np.sqrt(sum_sq_real / (i+1))
    sum_sq_imag =sum_sq_imag+(imag_error[i]*imag_error[i])
    imag_rmse = np.sqrt(sum_sq_imag / (i+1))
print("Real Part RMSE: ", real_rmse, "Imaginary Part RMSE:", imag_rmse)
if real_rmse<0.001 and imag_rmse<0.001:
    print("PASS")
else:
    print("FAIL")

Real Part RMSE:  2.0445955819550222e-05 Imaginary Part RMSE: 1.800892248477111e-05
PASS
```

```
In [11]:
plt.figure(figsize=(10, 5))
plt.subplot(1,2,1)
plt.bar(ind,real_error)
plt.title("Real Part Error")
plt.xlabel("Index")
plt.ylabel("Error")
#plt.xticks(ind)
plt.tight_layout()

plt.subplot(1,2,2)
plt.bar(ind,imag_error)
plt.title("Imaginary Part Error")
plt.xlabel("Index")
plt.ylabel("Error")
#plt.xticks(ind)
plt.tight_layout()
```

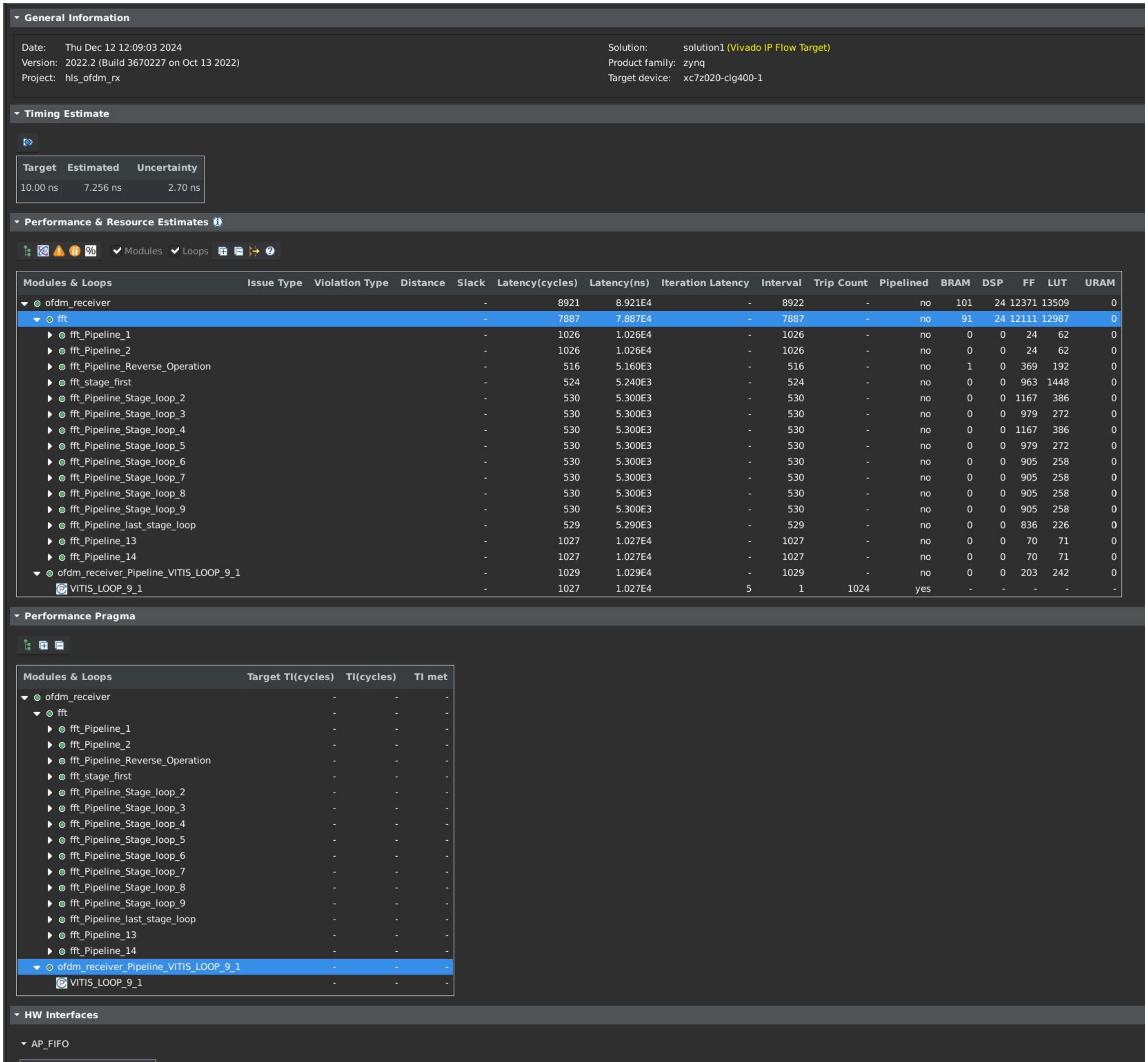




OFDM Receiver



```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../fft.cpp in debug mode
   Generating csim.exe
Reading INPUTS
Comparing with golden output
Comparing against output data
*******************************************
PASS: The output matches the golden output!
*******************************************
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```



General Information

Date: Thu Dec 12 12:09:03 2024
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: hls_ofdm_rx

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.256 ns	2.70 ns

Performance & Resource Estimates

✓ Modules ✓ Loops

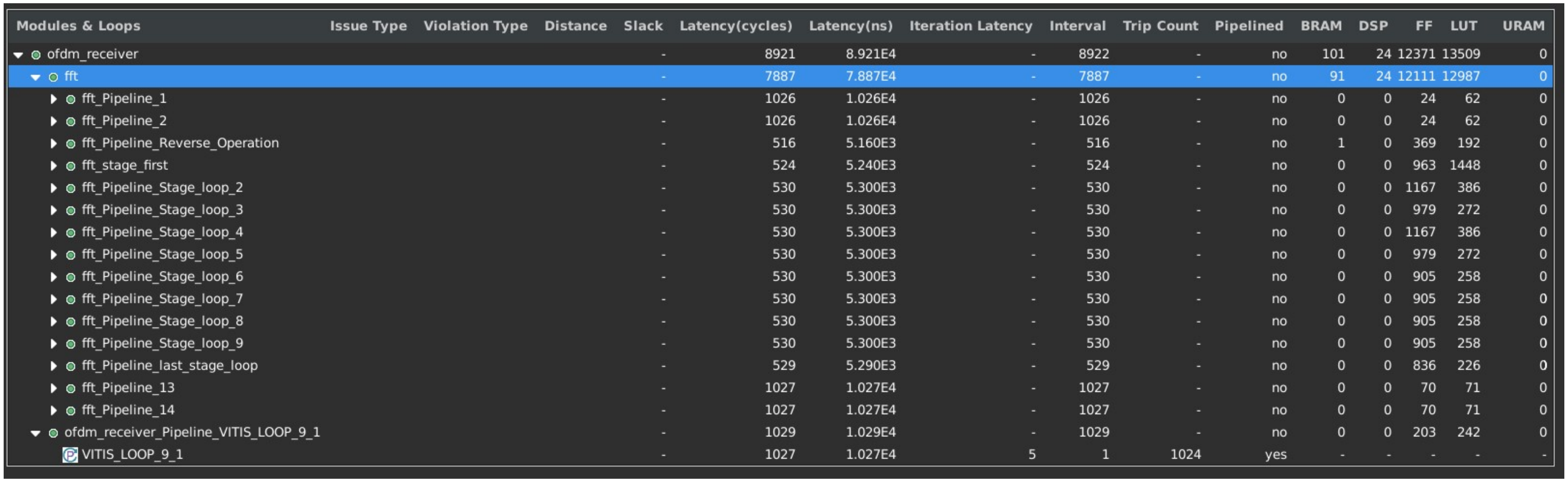
Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
ofdm_receiver				-	8921	8.921E4	-	8922	-	no	101	24	12371	13509	0
fft				-	7887	7.887E4	-	7887	-	no	91	24	12111	12987	0
fft_Pipeline_1				-	1026	1.026E4	-	1026	-	no	0	0	24	62	0
fft_Pipeline_2				-	1026	1.026E4	-	1026	-	no	0	0	24	62	0
fft_Pipeline_Reverse_Operation				-	516	5.160E3	-	516	-	no	1	0	369	192	0
fft_stage_first				-	524	5.240E3	-	524	-	no	0	0	963	1448	0
fft_Pipeline_Stage_loop_2				-	530	5.300E3	-	530	-	no	0	0	1167	386	0
fft_Pipeline_Stage_loop_3				-	530	5.300E3	-	530	-	no	0	0	979	272	0
fft_Pipeline_Stage_loop_4				-	530	5.300E3	-	530	-	no	0	0	1167	386	0
fft_Pipeline_Stage_loop_5				-	530	5.300E3	-	530	-	no	0	0	979	272	0
fft_Pipeline_Stage_loop_6				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_Stage_loop_7				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_Stage_loop_8				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_Stage_loop_9				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_last_stage_loop				-	529	5.290E3	-	529	-	no	0	0	836	226	0
fft_Pipeline_13				-	1027	1.027E4	-	1027	-	no	0	0	70	71	0
fft_Pipeline_14				-	1027	1.027E4	-	1027	-	no	0	0	70	71	0
ofdm_receiver_Pipeline_VITIS_LOOP_9_1				-	1029	1.029E4	-	1029	-	no	0	0	203	242	0
VITIS_LOOP_9_1				-	1027	1.027E4	5	1	1024	yes	-	-	-	-	-

Performance Pragma

Modules & Loops	Target TI(cycles)	TI(cycles)	TI met
ofdm_receiver	-	-	-
fft	-	-	-
fft_Pipeline_1	-	-	-
fft_Pipeline_2	-	-	-
fft_Pipeline_Reverse_Operation	-	-	-
fft_stage_first	-	-	-
fft_Pipeline_Stage_loop_2	-	-	-
fft_Pipeline_Stage_loop_3	-	-	-
fft_Pipeline_Stage_loop_4	-	-	-
fft_Pipeline_Stage_loop_5	-	-	-
fft_Pipeline_Stage_loop_6	-	-	-
fft_Pipeline_Stage_loop_7	-	-	-
fft_Pipeline_Stage_loop_8	-	-	-
fft_Pipeline_Stage_loop_9	-	-	-
fft_Pipeline_last_stage_loop	-	-	-
fft_Pipeline_13	-	-	-
fft_Pipeline_14	-	-	-
ofdm_receiver_Pipeline_VITIS_LOOP_9_1	-	-	-
VITIS_LOOP_9_1	-	-	-

HW Interfaces

AP_FIFO



Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
ofdm_receiver				-	8921	8.921E4	-	8922	-	no	101	24	12371	13509	0
fft				-	7887	7.887E4	-	7887	-	no	91	24	12111	12987	0
fft_Pipeline_1				-	1026	1.026E4	-	1026	-	no	0	0	24	62	0
fft_Pipeline_2				-	1026	1.026E4	-	1026	-	no	0	0	24	62	0
fft_Pipeline_Reverse_Operation				-	516	5.160E3	-	516	-	no	1	0	369	192	0
fft_stage_first				-	524	5.240E3	-	524	-	no	0	0	963	1448	0
fft_Pipeline_Stage_loop_2				-	530	5.300E3	-	530	-	no	0	0	1167	386	0
fft_Pipeline_Stage_loop_3				-	530	5.300E3	-	530	-	no	0	0	979	272	0
fft_Pipeline_Stage_loop_4				-	530	5.300E3	-	530	-	no	0	0	1167	386	0
fft_Pipeline_Stage_loop_5				-	530	5.300E3	-	530	-	no	0	0	979	272	0
fft_Pipeline_Stage_loop_6				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_Stage_loop_7				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_Stage_loop_8				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_Stage_loop_9				-	530	5.300E3	-	530	-	no	0	0	905	258	0
fft_Pipeline_last_stage_loop				-	529	5.290E3	-	529	-	no	0	0	836	226	0
fft_Pipeline_13				-	1027	1.027E4	-	1027	-	no	0	0	70	71	0
fft_Pipeline_14				-	1027	1.027E4	-	1027	-	no	0	0	70	71	0
ofdm_receiver_Pipeline_VITIS_LOOP_9_1				-	1029	1.029E4	-	1029	-	no	0	0	203	242	0
VITIS_LOOP_9_1				-	1027	1.027E4	5	1	1024	yes	-	-	-	-	-

DEMO for OFDM receiver was not required