ELD BONUS PRROJECT HANDOUT

AIM:

Create hardware IP for FFT using HLS tool.

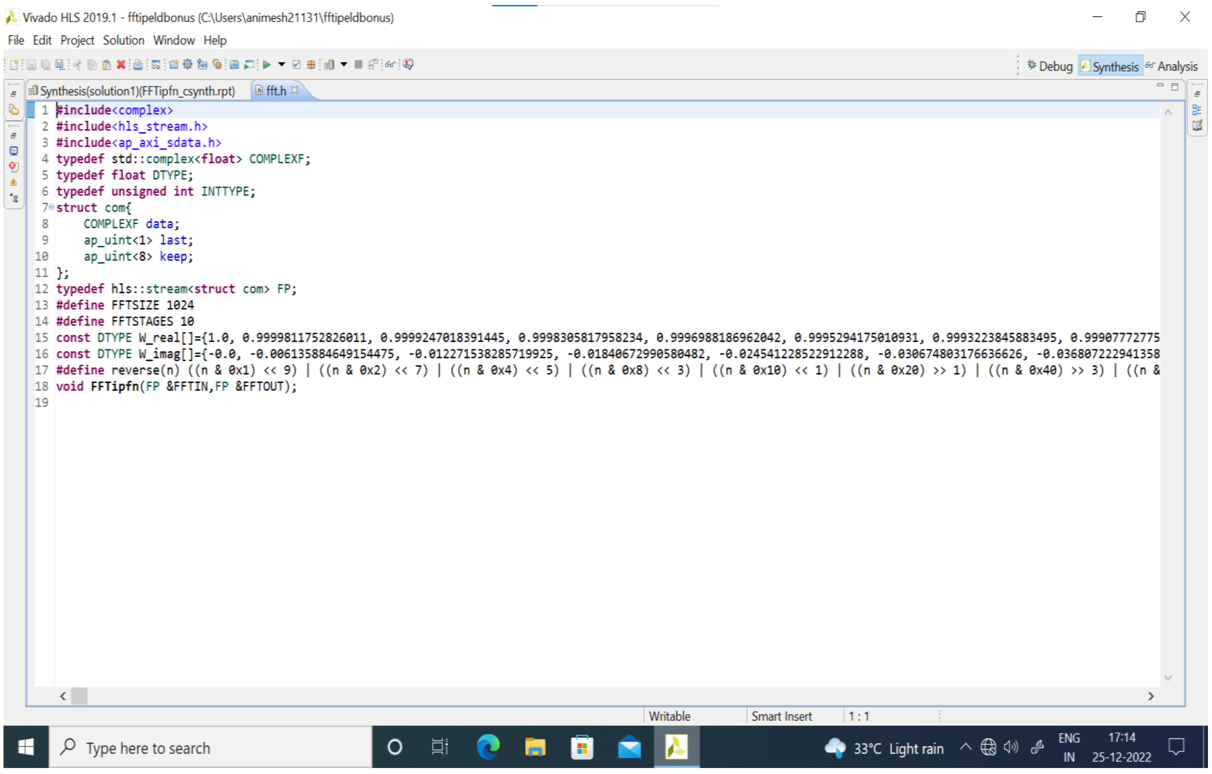
Introduction to project:

We have already implemented 1024 FFT IP on PS as well as on PL in lab12 (on zedboard) using the built-in FFT IP by vivado but now we shall do it using our own custom build FFT IP using HLS Tool. Vivado's High Level Synthesis (HLS) Tool is used to convert C/C++ code into Verilog. Tool can also be used to make IP(s). (**HERE WE WILL BE USING THE FFTSw.c FILE CODE USED IN LAB12 OF ELD**)

Steps:

PART-1 (CREATION OF FFT IP)

1. Open Vivado HLS 2019.1 and create a new project in it.
2. Cleck on next -> next -> next. Now select Board as Zedboard (diligent one) and ensure that period=10ns
3. Add two files (one is header and other code file) in Source and one in Testbench (mine are fft.h, fft.cpp and ffttb.cpp respectively)
4. In header write this code:

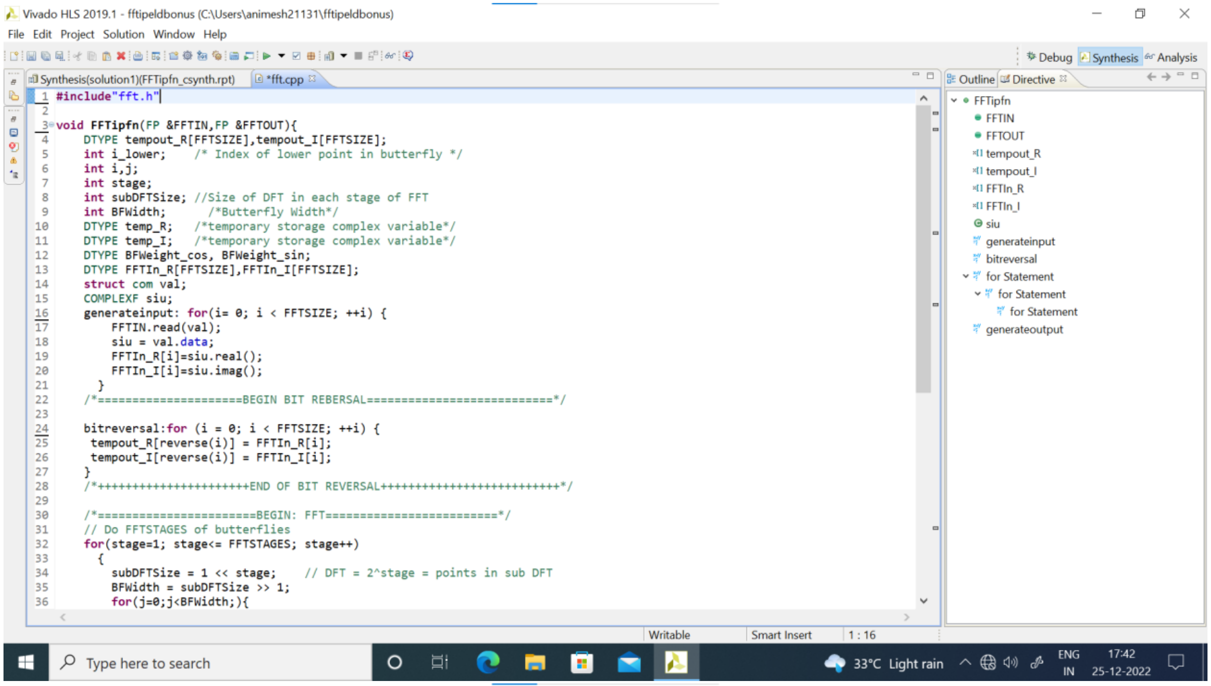


Vivado HLS 2019.1 - fftipeldbonus (C:\Users\animesh21131\fftipeldbonus)
File Edit Project Solution Window Help
Debug Synthesis Analysis
Synthesis(solution1)(FFTipfn_csynth.rpt)
fft.h
Writable
Smart Insert
1 : 1
Type here to search
33°C Light rain
ENG IN
17:14
25-12-2022

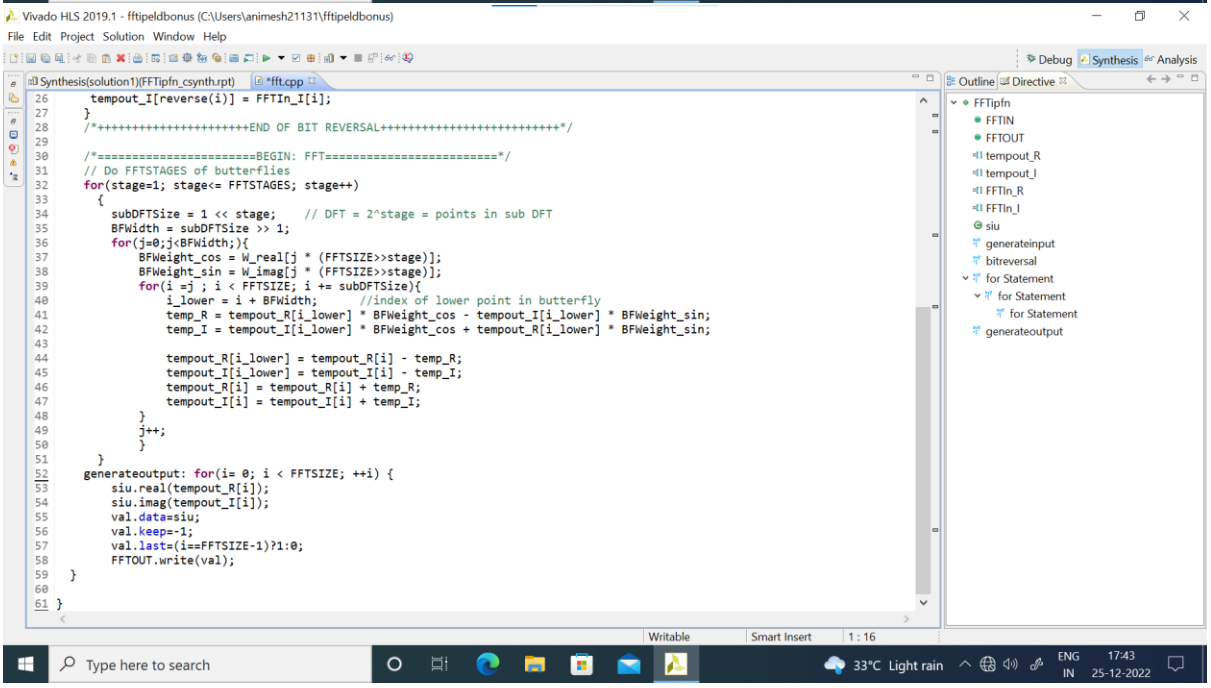
```
#include<complex>
#include<hls_stream.h>
#include<ap_axi_sdata.h>
typedef std::complex<float> COMPLEXF;
typedef float DTYPE;
typedef unsigned int INTTYPE;
struct com{
    COMPLEXF data;
    ap_uint<1> last;
    ap_uint<8> keep;
};
typedef hls::stream<struct com> FP;
#define FFTSIZE 1024
#define FFTSTAGES 10
const DTYPE W_real[]={1.0, 0.9999811752826011, 0.9999247018391445, 0.9998305817958234, 0.9996988186962042, 0.9995294175010931, 0.9993223845883495, 0.99907772775
const DTYPE W_imag[]={-0.0, -0.006135884649154475, -0.012271538285719925, -0.01840672990580482, -0.024541228522912288, -0.030674803176636626, -0.036807222941358
#define reverse(n) ((n & 0x1) << 9) | ((n & 0x2) << 7) | ((n & 0x4) << 5) | ((n & 0x8) << 3) | ((n & 0x10) << 1) | ((n & 0x20) >> 1) | ((n & 0x40) >> 3) | ((n &
void FFTipfn(FP &FFTIN,FP &FFTOUT);
```

(Pls note: here W_real and W_imag and reverse(n) are same as in the attatched fftSw.c file)

5. Now the source main code file would have code as in my fft.cpp:



```
#include"fft.h"

void FFTipfn(FP &FFTIN,FP &FFTOUT){
    DTYPE tempout_R[FFTSIZE],tempout_I[FFTSIZE];
    int i_lower;    /* Index of lower point in butterfly */
    int i,j;
    int stage;
    int subDFTSize; //Size of DFT in each stage of FFT
    int BFWidth;      /*Butterfly Width*/
    DTYPE temp_R;   /*temporary storage complex variable*/
    DTYPE temp_I;   /*temporary storage complex variable*/
    DTYPE BFWeight_cos, BFWeight_sin;
    DTYPE FFTIn_R[FFTSIZE],FFTIn_I[FFTSIZE];
    struct com val;
    COMPLEXF siu;
    generateinput: for(i= 0; i < FFTSIZE; ++i) {
        FFTIN.read(val);
        siu = val.data;
        FFTIn_R[i]=siu.real();
        FFTIn_I[i]=siu.imag();
      }
    /*=====================BEGIN BIT REBERSAL===========================*/

    bitreversal:for (i = 0; i < FFTSIZE; ++i) {
     tempout_R[reverse(i)] = FFTIn_R[i];
     tempout_I[reverse(i)] = FFTIn_I[i];
    }
    /*++++++++++++++++++++++END OF BIT REVERSAL+++++++++++++++++++++++++*/

    /*======================BEGIN: FFT=========================*/
    // Do FFTSTAGES of butterflies
    for(stage=1; stage<= FFTSTAGES; stage++)
      {
        subDFTSize = 1 << stage;    // DFT = 2^stage = points in sub DFT
        BFWidth = subDFTSize >> 1;
        for(j=0;j<BFWidth;){
```



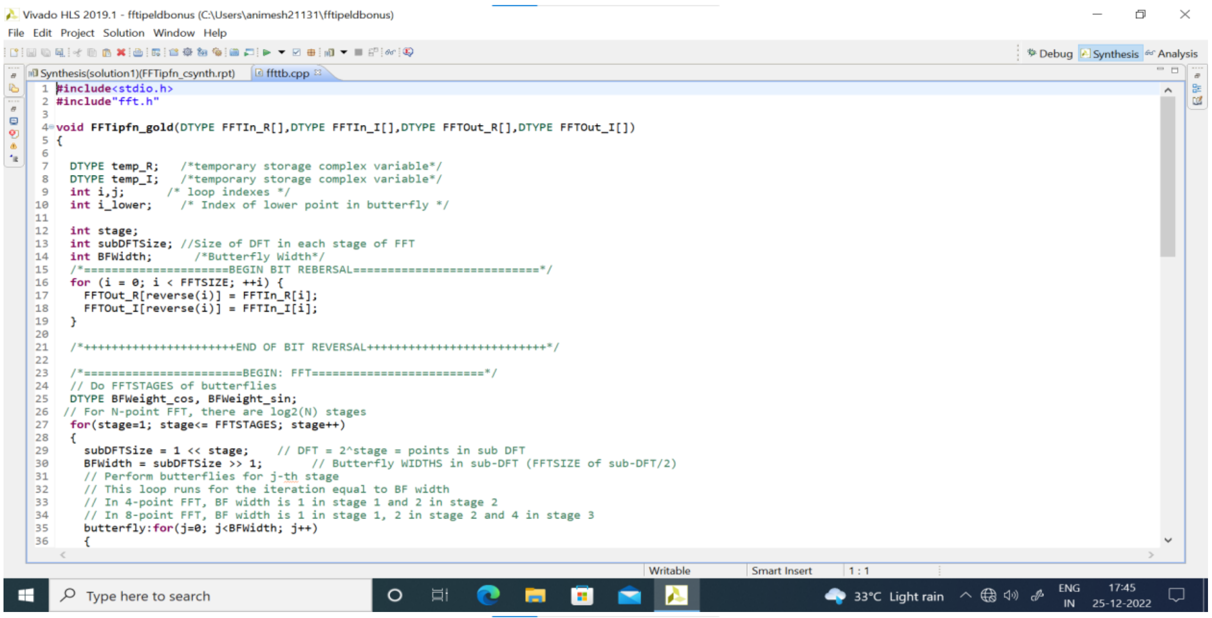
```
     tempout_I[reverse(i)] = FFTIn_I[i];
    }
    /*++++++++++++++++++++++END OF BIT REVERSAL+++++++++++++++++++++++++*/

    /*======================BEGIN: FFT=========================*/
    // Do FFTSTAGES of butterflies
    for(stage=1; stage<= FFTSTAGES; stage++)
      {
        subDFTSize = 1 << stage;    // DFT = 2^stage = points in sub DFT
        BFWidth = subDFTSize >> 1;
        for(j=0;j<BFWidth;){
            BFWeight_cos = W_real[j * (FFTSIZE>>stage)];
            BFWeight_sin = W_imag[j * (FFTSIZE>>stage)];
            for(i =j ; i < FFTSIZE; i += subDFTSize){
                i_lower = i + BFWidth;      //index of lower point in butterfly
                temp_R = tempout_R[i_lower] * BFWeight_cos - tempout_I[i_lower] * BFWeight_sin;
                temp_I = tempout_I[i_lower] * BFWeight_cos + tempout_R[i_lower] * BFWeight_sin;

                tempout_R[i_lower] = tempout_R[i] - temp_R;
                tempout_I[i_lower] = tempout_I[i] - temp_I;
                tempout_R[i] = tempout_R[i] + temp_R;
                tempout_I[i] = tempout_I[i] + temp_I;
            }
            j++;
            }
      }
    generateoutput: for(i= 0; i < FFTSIZE; ++i) {
        siu.real(tempout_R[i]);
        siu.imag(tempout_I[i]);
        val.data=siu;
        val.keep=-1;
        val.last=(i==FFTSIZE-1)?1:0;
        FFTOUT.write(val);
  }

}
```

6. No we will insert Directives in the code. Now on the The Directive panel on right side click the FFTipfn of code then select option of insert directive and then select INTERFACE directive and in mode select ap_ctrl_none.

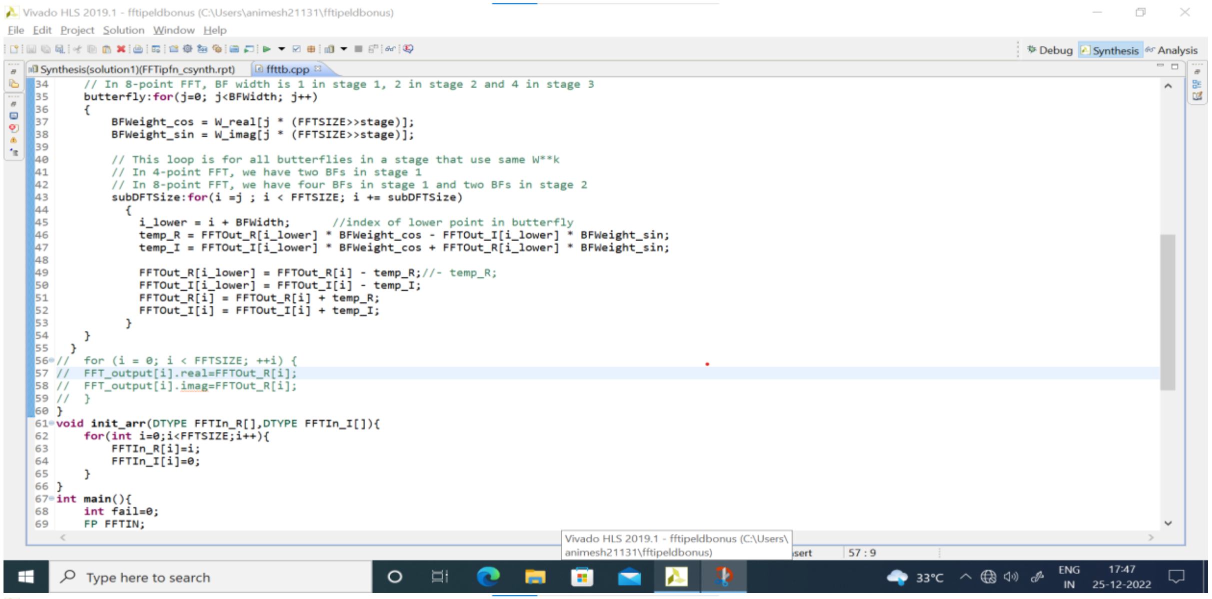
7. Simalarly for other elements follow the table:

Element	directive	mode
FFTIN	Interface	axis
FFTOUT	Interface	axis
generateinput	pipeline	-
bitreversal	pipeline	-
generateoutput	pipeline	-

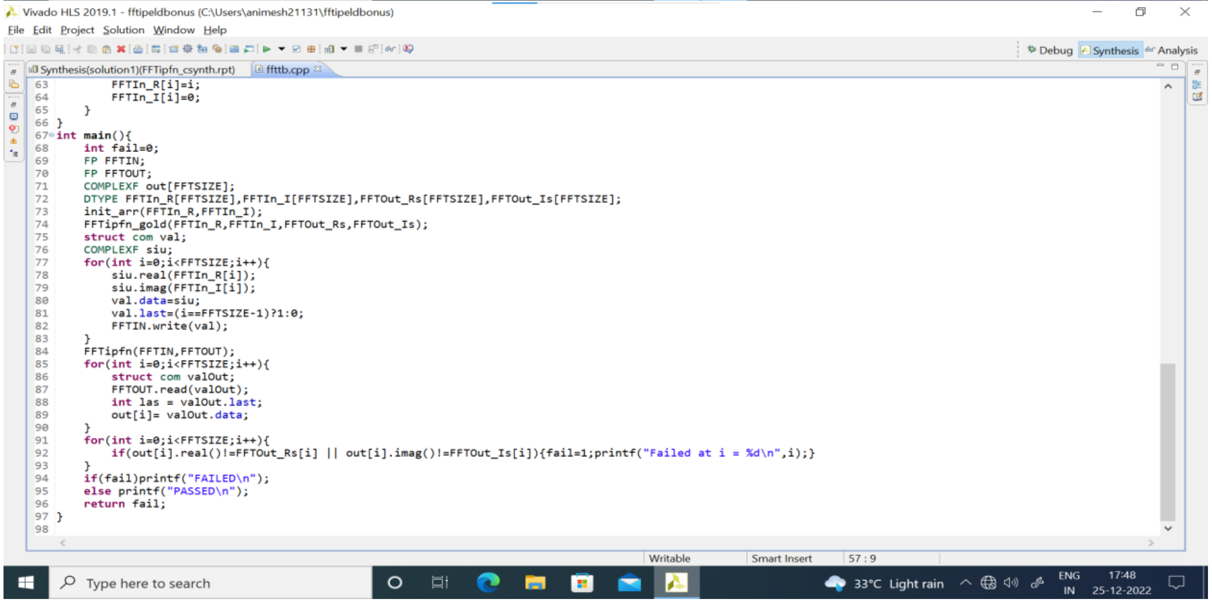
8. Now Testbench file would be:


Vivado HLS 2019.1 - fftipeldbonus (C:\Users\animesh21131\fftipeldbonus)
File Edit Project Solution Window Help
Debug Synthesis Analysis
Synthesis(solution1)(FFTipfn_csynth.rpt)
ffttb.cpp
Writable
Smart Insert
1 : 1
Type here to search
33°C Light rain
ENG IN
17:45
25-12-2022

```
#include<stdio.h>
#include"fft.h"

void FFTipfn_gold(DTYPE FFTIn_R[],DTYPE FFTIn_I[],DTYPE FFTOut_R[],DTYPE FFTOut_I[])
{

  DTYPE temp_R;   /*temporary storage complex variable*/
  DTYPE temp_I;   /*temporary storage complex variable*/
  int i,j;      /* loop indexes */
  int i_lower;    /* Index of lower point in butterfly */

  int stage;
  int subDFTSize; //Size of DFT in each stage of FFT
  int BFWidth;      /*Butterfly Width*/
  /*=====================BEGIN BIT REBERSAL===========================*/
  for (i = 0; i < FFTSIZE; ++i) {
    FFTOut_R[reverse(i)] = FFTIn_R[i];
    FFTOut_I[reverse(i)] = FFTIn_I[i];
  }

  /*++++++++++++++++++++++END OF BIT REVERSAL++++++++++++++++++++++++++*/

  /*======================BEGIN: FFT========================*/
  // Do FFTSTAGES of butterflies
  DTYPE BFWeight_cos, BFWeight_sin;
 // For N-point FFT, there are log2(N) stages
  for(stage=1; stage<= FFTSTAGES; stage++)
  {
    subDFTSize = 1 << stage;    // DFT = 2^stage = points in sub DFT
    BFWidth = subDFTSize >> 1;       // Butterfly WIDTHS in sub-DFT (FFTSIZE of sub-DFT/2)
    // Perform butterflies for j-th stage
    // This loop runs for the iteration equal to BF width
    // In 4-point FFT, BF width is 1 in stage 1 and 2 in stage 2
    // In 8-point FFT, BF width is 1 in stage 1, 2 in stage 2 and 4 in stage 3
    butterfly:for(j=0; j<BFWidth; j++)
    {
```

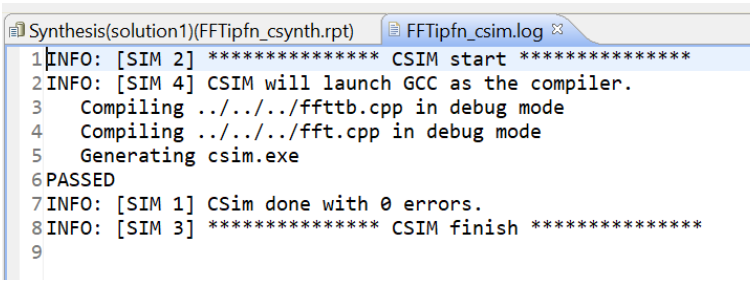

Vivado HLS 2019.1 - fftipeldbonus (C:\Users\animesh21131\fftipeldbonus)
File Edit Project Solution Window Help
Debug Synthesis Analysis
Synthesis(solution1)(FFTipfn_csynth.rpt)
ffttb.cpp
Vivado HLS 2019.1 - fftipeldbonus (C:\Users\animesh21131\fftipeldbonus)
57 : 9
Type here to search
33°C
ENG IN
17:47
25-12-2022

```
    // In 8-point FFT, BF width is 1 in stage 1, 2 in stage 2 and 4 in stage 3
    butterfly:for(j=0; j<BFWidth; j++)
    {
        BFWeight_cos = W_real[j * (FFTSIZE>>stage)];
        BFWeight_sin = W_imag[j * (FFTSIZE>>stage)];

        // This loop is for all butterflies in a stage that use same W**k
        // In 4-point FFT, we have two BFs in stage 1
        // In 8-point FFT, we have four BFs in stage 1 and two BFs in stage 2
        subDFTSize:for(i =j ; i < FFTSIZE; i += subDFTSize)
          {
            i_lower = i + BFWidth;      //index of lower point in butterfly
            temp_R = FFTOut_R[i_lower] * BFWeight_cos - FFTOut_I[i_lower] * BFWeight_sin;
            temp_I = FFTOut_I[i_lower] * BFWeight_cos + FFTOut_R[i_lower] * BFWeight_sin;

            FFTOut_R[i_lower] = FFTOut_R[i] - temp_R;//- temp_R;
            FFTOut_I[i_lower] = FFTOut_I[i] - temp_I;
            FFTOut_R[i] = FFTOut_R[i] + temp_R;
            FFTOut_I[i] = FFTOut_I[i] + temp_I;
          }
    }
  }
//  for (i = 0; i < FFTSIZE; ++i) {
//  FFT_output[i].real=FFTOut_R[i];
//  FFT_output[i].imag=FFTOut_R[i];
//  }
}
void init_arr(DTYPE FFTIn_R[],DTYPE FFTIn_I[]){
    for(int i=0;i<FFTSIZE;i++){
        FFTIn_R[i]=i;
        FFTIn_I[i]=0;
    }
}
int main(){
    int fail=0;
    FP FFTIN;
```


Vivado HLS 2019.1 - fftipeldbonus (C:\Users\animesh21131\fftipeldbonus)
File Edit Project Solution Window Help
Debug Synthesis Analysis
Synthesis(solution1)(FFTipfn_csynth.rpt)
ffttb.cpp
Writable
Smart Insert
57 : 9
Type here to search
33°C Light rain
ENG IN
17:48
25-12-2022

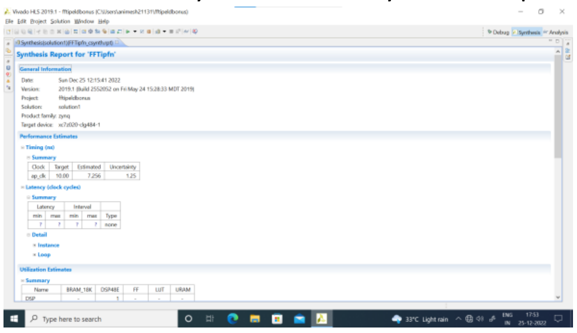
```
        FFTIn_R[i]=i;
        FFTIn_I[i]=0;
    }
}
int main(){
    int fail=0;
    FP FFTIN;
    FP FFTOUT;
    COMPLEXF out[FFTSIZE];
    DTYPE FFTIn_R[FFTSIZE],FFTIn_I[FFTSIZE],FFTOut_Rs[FFTSIZE],FFTOut_Is[FFTSIZE];
    init_arr(FFTIn_R,FFTIn_I);
    FFTipfn_gold(FFTIn_R,FFTIn_I,FFTOut_Rs,FFTOut_Is);
    struct com val;
    COMPLEXF siu;
    for(int i=0;i<FFTSIZE;i++){
        siu.real(FFTIn_R[i]);
        siu.imag(FFTIn_I[i]);
        val.data=siu;
        val.last=(i==FFTSIZE-1)?1:0;
        FFTIN.write(val);
    }
    FFTipfn(FFTIN,FFTOUT);
    for(int i=0;i<FFTSIZE;i++){
        struct com valOut;
        FFTOUT.read(valOut);
        int las = valOut.last;
        out[i]= valOut.data;
    }
    for(int i=0;i<FFTSIZE;i++){
        if(out[i].real()!=FFTOut_Rs[i] || out[i].imag()!=FFTOut_Is[i]){fail=1;printf("Failed at i = %d\n",i);}
    }
    if(fail)printf("FAILED\n");
    else printf("PASSED\n");
    return fail;
}
```

9. Add FFTipfn as top Function of the Project by going to Synthesis of Project settings under Project tab.
10. Now under solution open Solution settings and edit config_export (in General). Set version value as 1.1.1.
11. Now Run C Simulation and we will get result as PASSED.



```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../ffttb.cpp in debug mode
   Compiling ../../../fft.cpp in debug mode
   Generating csim.exe
PASSED
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

12. Now run the C synthesis and a Synthesis Report will be generated:


Vivado HLS 2019.1 - fftipeldbonus (C:\Users\animesh21131\fftipeldbonus)
File Edit Project Solution Window Help
Debug Synthesis Analysis
Synthesis(solution1)(FFTipfn_csynth.rpt)
Type here to search
33°C Light rain
ENG IN 17:53 25-12-2022

Synthesis Report for 'FFTipfn'

General Information

Date:	Sun Dec 25 12:15:41 2022
Version:	2019.1 (Build 2552052 on Fri May 24 15:28:33 MDT 2019)
Project:	fftipeldbonus
Solution:	solution1
Product family:	zynq
Target device:	xc7z020-clg484-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.256	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
?	?	?	?	none

Detail

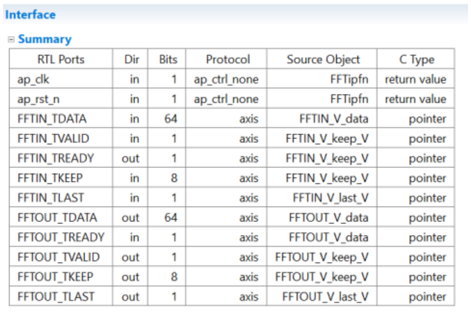
Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	1	-	-	-



Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	FFTipfn	return value
ap_rst_n	in	1	ap_ctrl_none	FFTipfn	return value
FFTIN_TDATA	in	64	axis	FFTIN_V_data	pointer
FFTIN_TVALID	in	1	axis	FFTIN_V_keep_V	pointer
FFTIN_TREADY	out	1	axis	FFTIN_V_keep_V	pointer
FFTIN_TKEEP	in	8	axis	FFTIN_V_keep_V	pointer
FFTIN_TLAST	in	1	axis	FFTIN_V_last_V	pointer
FFTOUT_TDATA	out	64	axis	FFTOUT_V_data	pointer
FFTOUT_TREADY	in	1	axis	FFTOUT_V_data	pointer
FFTOUT_TVALID	out	1	axis	FFTOUT_V_keep_V	pointer
FFTOUT_TKEEP	out	8	axis	FFTOUT_V_keep_V	pointer
FFTOUT_TLAST	out	1	axis	FFTOUT_V_last_V	pointer

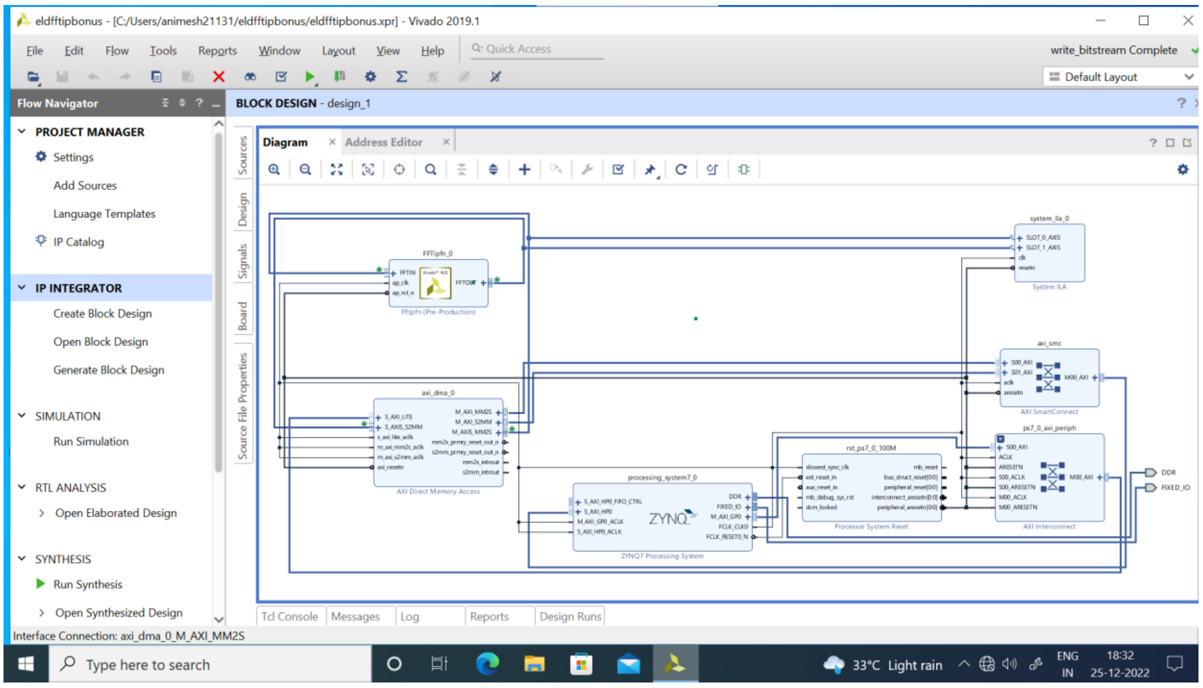
13. Now run C/RTL simulation and then export RTL.

PART-2 (Verification OF FFT IP by integrating it with our lab-12)

(Note- I am using my lab-12 eld project of vivado 2019.1 for this part)

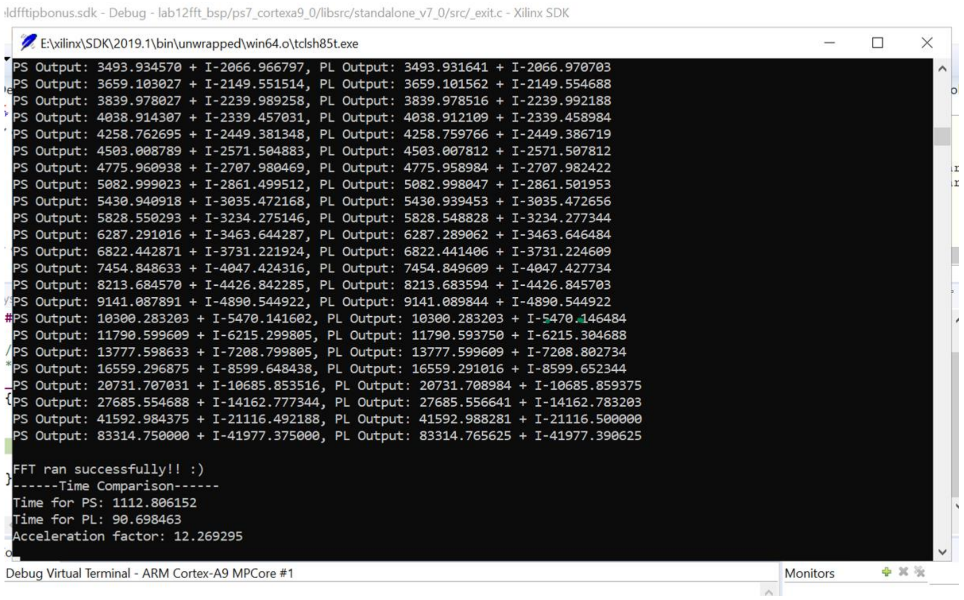
1. Open Vivado 2019.1 and open the existing project lab12.
2. Now go to File->Project->Save as and save it with a new name, say p1.
3. Now close the lab12 project and open p1. (Basically, we have cloned our lab12 project in p1)
4. Now go to Settings under Project manager.
5. Under IP select Repository option. Click on Add (+) and select our HLS project created in previous part. This will add our IP in Vivado's IP catalog.
6. Now Open the Block design of this project. There Delete FFT ip and System ILA.
7. Now click on add (+) and add our FFTIP (FFTipfn) to the block design.
8. Now manually connect the FFTOut of our IP to the S_AXIS_S2MM of AXI DMA IP and FFTIN of our IP to the M_AXIS_MM2S of the AXI DMA IP.
9. Also, Connect the aclk of our IP to the s_axi_lite_aclk of AXI DMA IP and the aresetn pin of our IP to the axi_aresetn pin of AXI DMA IP.
10. Debug the S_AXIS DATA and M_AXIS_DATA of FFT IP by right clicking and then run connection automation.
11. System ILA will be added automatically.
12. The block design is now complete.
13. Validate your design to check for any missing or incorrect connections.

14. Final Design should look like this:



15. Then in the sources tab, right-click on your block design file and select Create HDL Wrapper.
16. Click on OK. This step will generate the corresponding Verilog files of your design.
17. Now generate the bitstream
18. Once the bitstream is generated, “Export Hardware” and while doing so, include the bitstream.
19. Click on Launch SDK and wait for it to open.
20. As we have cloned the project, we will already have lab12’s Application Project there as well (with all lab12’s settings). Same project will work for this project as well. So, lets verify our IP’s working.
21. Add the Hardware target and test its connection, when successfully connected Right click on the lab12’s application project, go to Debug As and then select Debug Configurations.
22. Double click the Xilinx C/C++ Application (System Debugger) to define the debug configuration for the current run. Check reset the entire system and program FPGA. Leave other settings unchanged.
23. Once the system debugger is launched, type jtagterminal in XSCT Console. This will open the terminal for displaying output messages.
24. Click on the resume button.

25. Terminal window will have these outputs:


ldfftipbonus.sdk - Debug - lab12fft_bsp/ps7_cortexa9_0/libsrc/standalone_v7_0/src/_exit.c - Xilinx SDK
E:\xilinx\SDK\2019.1\bin\unwrapped\win64.o\tclsh85t.exe
Debug Virtual Terminal - ARM Cortex-A9 MPCore #1
Monitors

```
PS Output: 3493.934570 + I-2066.966797, PL Output: 3493.931641 + I-2066.970703
PS Output: 3659.103027 + I-2149.551514, PL Output: 3659.101562 + I-2149.554688
PS Output: 3839.978027 + I-2239.989258, PL Output: 3839.978516 + I-2239.992188
PS Output: 4038.914307 + I-2339.457031, PL Output: 4038.912109 + I-2339.458984
PS Output: 4258.762695 + I-2449.381348, PL Output: 4258.759766 + I-2449.386719
PS Output: 4503.008789 + I-2571.504883, PL Output: 4503.007812 + I-2571.507812
PS Output: 4775.960938 + I-2707.980469, PL Output: 4775.958984 + I-2707.982422
PS Output: 5082.999023 + I-2861.499512, PL Output: 5082.998047 + I-2861.501953
PS Output: 5430.940918 + I-3035.472168, PL Output: 5430.939453 + I-3035.472656
PS Output: 5828.550293 + I-3234.275146, PL Output: 5828.548828 + I-3234.277344
PS Output: 6287.291016 + I-3463.644287, PL Output: 6287.289062 + I-3463.646484
PS Output: 6822.442871 + I-3731.221924, PL Output: 6822.441406 + I-3731.224609
PS Output: 7454.848633 + I-4047.424316, PL Output: 7454.849609 + I-4047.427734
PS Output: 8213.684570 + I-4426.842285, PL Output: 8213.683594 + I-4426.845703
PS Output: 9141.087891 + I-4890.544922, PL Output: 9141.089844 + I-4890.544922
PS Output: 10300.283203 + I-5470.141602, PL Output: 10300.283203 + I-5470.146484
PS Output: 11790.599609 + I-6215.299805, PL Output: 11790.593750 + I-6215.304688
PS Output: 13777.598633 + I-7208.799805, PL Output: 13777.599609 + I-7208.802734
PS Output: 16559.296875 + I-8599.648438, PL Output: 16559.291016 + I-8599.652344
PS Output: 20731.707031 + I-10685.853516, PL Output: 20731.708984 + I-10685.859375
PS Output: 27685.554688 + I-14162.777344, PL Output: 27685.556641 + I-14162.783203
PS Output: 41592.984375 + I-21116.492188, PL Output: 41592.988281 + I-21116.500000
PS Output: 83314.750000 + I-41977.375000, PL Output: 83314.765625 + I-41977.390625

FFT ran successfully!! :)
------Time Comparison------
Time for PS: 1112.806152
Time for PL: 90.698463
Acceleration factor: 12.269295
```

26. From these outputs we saw that our IP is correctly functioning and we successfully made our first AXI stream based IP using Vivado HLS 2019.1

References and Special Thanks:

IIITD ECE573 AELD: Lab_10_Part_4: AXI Stream IP via HLS #zedboard #iiitd #iiitdelhi - YouTube

IIITD ECE573 AELD: Lab_10_Part_5: Microblaze + HLS IP + AXI DMA #zedboard #iiitd #iiitdelhi - YouTube

Complex numbers in C++ | Set 1 - GeeksforGeeks

Lab 11 Handout (google.com)

Lab 12 Handout (google.com)

Special thanks to Syed sir for helping me solve the complex.h inclusion problem in HLS and for fftSw.c (lab12 code)

Also Special thanks to Sumit sir for AELD and ELD amazing courses

---

End of Handout
