HLS-Assignment 8

May 26, 2023

Sampath Govardhan
FWC22071

VITIS-HLS

1 Problem Statement

```
1. Write an HLS module, along with testbench, for Cyclic Prefix
removal from an incoming set of data inputs.

2. After HLS testing using HLS testbench and synthesis of the
module in HLS, export the module to Vivado application.

3. The input to the CP removal module should be given from another
HLS module which will act as a data generator. Use an array to
store the input values from the attached data set file in the
mail and perform synthesis in HLS to get the data generator module.

4. Interface both the modules with Xilinx FFT IP and perform FFT
on the output of Cyclic Prefix Removal in the Vivado application.

5. Generate a wrapper for the block diagram in Vivado and then write
a test bench in Verilog which will provide clock and reset to these
modules and store the FFT output in a file.

6. Run simulation. Compare the output obtained from Verilog by writing
a code in MATLAB for the same functionality and check for correctness.
```

2 Header File

```
#ifndef _HEADER_H_
#define _HEADER_H_

#include <iostream>
#include <complex>
#include <fstream>
#include <hls_stream.h>
#include <vector>
using namespace std;
#define N 8800
#define P 608    //(320+288)

typedef struct {
    float real;
    float imag;
} ComplexT;

void gen(hls::stream<ComplexT> &gst);
void cyclicPrefixRemoval(hls::stream<ComplexT> &inpstream, hls::stream<ComplexT>

#endif
```

3 Data Generator Code

```
#include "header.h"

void gen(hls::stream<ComplexT> &gst) {
#pragma HLS INTERFACE mode=axis register_mode=both port=gst register
```

```
value z[N]={{0.36492,-0.728851},{-0.752842,0.31251},{0.875913,0.27137},{-0.36993

          for (int i = 0; i < N; i++) {
#pragma HLS PIPELINE II=1
        gst.write(z[i]);

    }
}
```

4 Cyclic Prefix Removal Code

```
#include "header.h"

void cyclicPrefixRemoval(hls::stream<ComplexT> &inpstream,
 hls::stream<ComplexT> &oupstream, hls::stream<bool> &z) {
#pragma HLS INTERFACE mode=axis register_mode=off port=z
#pragma HLS INTERFACE mode=axis register_mode=both port=oupstream register
#pragma HLS INTERFACE mode=axis register_mode=both port=inpstream register

ComplexT invar;
ComplexT x[N];
bool last=false;
for (int i=0;i<N;i++){
#pragma HLS PIPELINE II=1
    invar = inpstream.read();
    x[i]=invar;
}
    for(int i=0;i<N-P;i++){
#pragma HLS PIPELINE II=1
        if(i<4096){
        oupstream.write(x[i+320]);
        }
        else{
        oupstream.write(x[i+608]);
        if (i == (N - P - 1)) {
                            last = true;
                        }
        else{last=false;}
        }

    }
z.write(last);
}
```

5 Test Bench Code

```
#include "header.h"

int main() {

    //PART-1:  Taking input values from given .txt files and
        //          storing resultant complex value in an array
                // and also in another file.

    ComplexT x[N];
    float c, d;
    float a,b;
    ifstream in1("puschTxAfterChannelReal.txt");
    ifstream in2("puschTxAfterChannelImag.txt");
    //ofstream ival("ival.dat");
    ofstream inp("input.dat");
    int l=0;
    for (int i = 0; i < N; i++) {
        l=l+1;
        in1 >> c;
        in2 >> d;
        a=c;b=d;
        x[i].real = a;
        x[i].imag=b;
        inp<<"{"<<x[i].real<<","<<x[i].imag<<"}"<<","<<endl;

       /* ival <<"ComplexT"<< x[i-1]<<",";
       if(l%5==0){
           ival<<endl;
       }*/
    }

    in1.close();
    in2.close();
    inp.close();

    //PART-2:  The output of CPR is strored in another
            // array and also in another file.

    ofstream oup("output.dat");
    ComplexT y[N];
    hls::stream<ComplexT> gst,oput;
    int t=0;
     hls::stream<bool> bf;
```

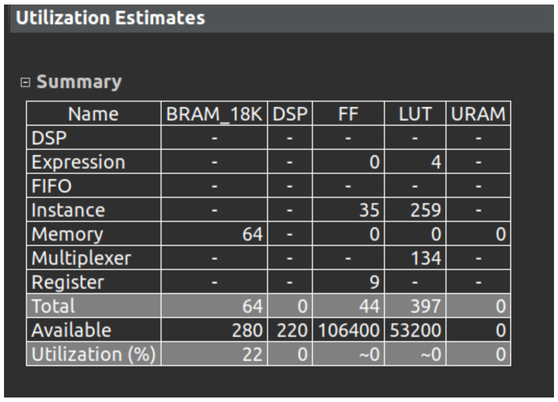
```
hls::stream<int> z;
ComplexT output;

gen(gst);
cyclicPrefixRemoval(gst, oput, bf);
    for (int i=0;i<N-P;i++){
    if (!oput.empty()) {
      output = oput.read();
      y[i].real=output.real;
      y[i].imag=output.imag;
      oup<<"{"<<y[i].real<<","<<y[i].imag<<"}"<<","<<endl;
    }
    /*else {
     cout<<"Skipping Cyclic Prefix BITS"<<endl;
   }*/
}
oup.close();

//PART-3: The input and output values are compared
//        using given precision and
//        then the result is stored in another file

ofstream out("out.dat");
int q=0;
bool f=0;
for (int i = 0; i < N-P; i++) {
    q=q+1;
    out << "Output[" << i << "]: " <<
    "{"<<y[i].real<<","<<y[i].imag<<"}"<<"\t";
    if (q<=4096){
            out<<"FIRST SYMBOL"<<"\t";

    if ((y[i].real-x[i+320].real)/y[i].real < 10e-3 && (y[i].imag
    x[i+320].imag)/y[i].imag <10e-3){
            out<<"Pass"<<endl;
    }
    else{
            f=1;
            //cout<<f<<endl;
            out<<"Fail"<<endl;
    }
}
else{
    out<<"SECOND SYMBOL"<<"\t";
    if ((y[i].real-x[i+P].real)/y[i].real < 10e-3 && (y[i].imag
    x[i+P].imag)y[i].imag < 10e-3){
```

```
                        out<<"Pass"<<endl;
                }
                else{
                        f=1;
                        //cout<<f<<endl;
                out<<"Fail"<<endl;
                }
        }
        }
        out.close();
        if (f==1){cout<<"!!FAIL!! OUTPUT IS NOT TOLERABLE BASED ON GIVEN
        PRECISION"<<endl;}
        else {cout<<"!PASS! OUTPUT IS TOLERABLE BASED ON GIVEN
        PRECISION"<<endl;}
        return 0;

}
```

6 C simulation Output

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
make: 'csim.exe' is up to date.
!PASS! OUTPUT IS TOLERABLE BASED ON GIVEN PRECISION
WARNING [HLS SIM]: hls::stream 'hls::stream<bool, 0>0' contains
leftover data, which may result in RTL simulation hanging.
INFO [HLS SIM]: The maximum depth reached by any hls::stream()
instance in the design is 8800
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

7 HLS Resource Consumption

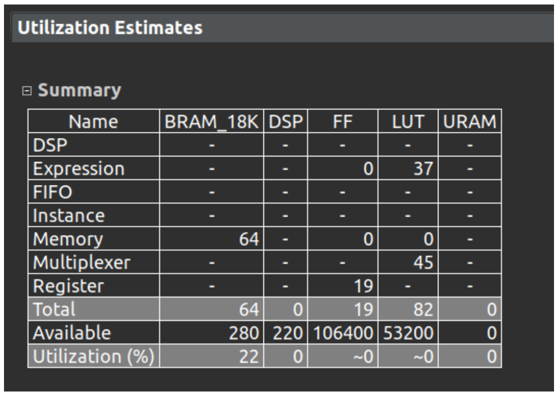


Utilization Estimates

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	4	-
FIFO	-	-	-	-	-
Instance	-	-	35	259	-
Memory	64	-	0	0	0
Multiplexer	-	-	-	134	-
Register	-	-	9	-	-
Total	64	0	44	397	0
Available	280	220	106400	53200	0
Utilization (%)	22	0	~0	~0	0

(a) CP Removal



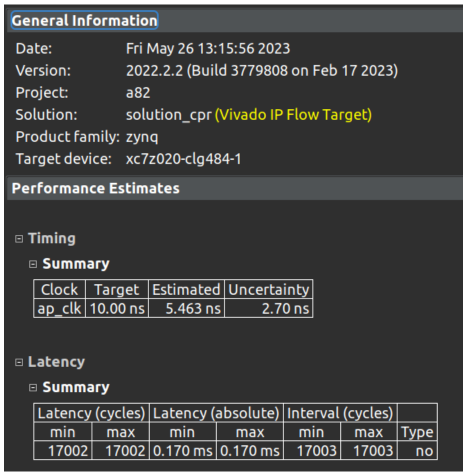
Utilization Estimates

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	37	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	64	-	0	0	-
Multiplexer	-	-	-	45	-
Register	-	-	19	-	-
Total	64	0	19	82	0
Available	280	220	106400	53200	0
Utilization (%)	22	0	~0	~0	0

(b) Generator

8 HLS Timing Report



General Information

Date: Fri May 26 13:15:56 2023
Version: 2022.2.2 (Build 3779808 on Feb 17 2023)
Project: a82
Solution: solution_cpr (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg484-1

Performance Estimates

⊟ Timing

⊟ Summary

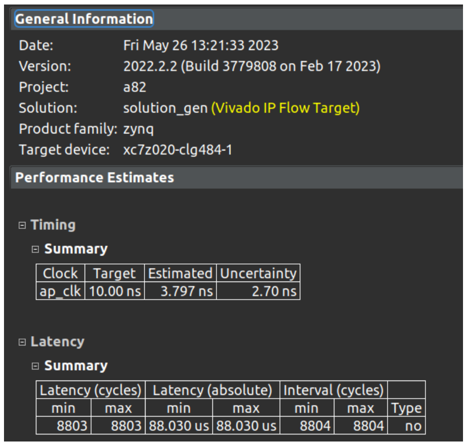
Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	5.463 ns	2.70 ns

⊟ Latency

⊟ Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
17002	17002	0.170 ms	0.170 ms	17003	17003	no

(a) CP Removal



General Information

Date: Fri May 26 13:21:33 2023
Version: 2022.2.2 (Build 3779808 on Feb 17 2023)
Project: a82
Solution: solution_gen (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg484-1

Performance Estimates

⊟ Timing

⊟ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	3.797 ns	2.70 ns

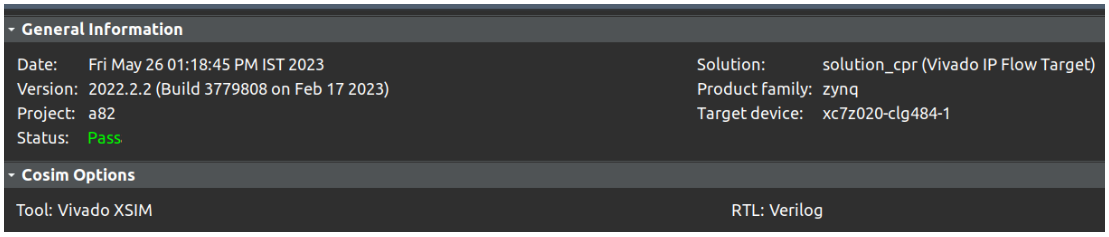
⊟ Latency

⊟ Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
8803	8803	88.030 us	88.030 us	8804	8804	no

(b) Generator

9 CoSimulation Report



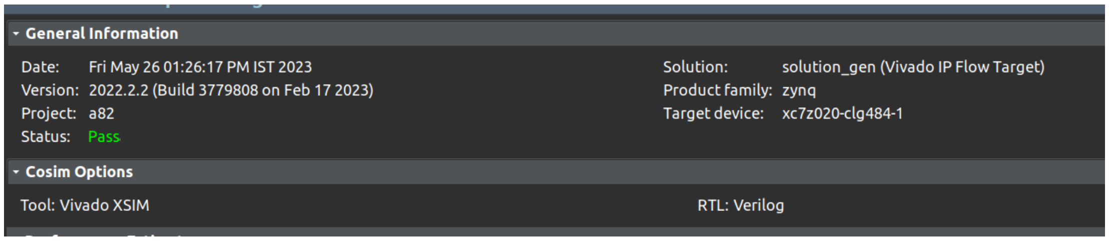
General Information

Date:	Fri May 26 01:18:45 PM IST 2023	Solution:	solution_cpr (Vivado IP Flow Target)
Version:	2022.2.2 (Build 3779808 on Feb 17 2023)	Product family:	zynq
Project:	a82	Target device:	xc7z020-clg484-1
Status:	Pass		

Cosim Options

Tool: Vivado XSIM RTL: Verilog

(a) CP Removal



General Information

Date:	Fri May 26 01:26:17 PM IST 2023	Solution:	solution_gen (Vivado IP Flow Target)
Version:	2022.2.2 (Build 3779808 on Feb 17 2023)	Product family:	zynq
Project:	a82	Target device:	xc7z020-clg484-1
Status:	Pass		

Cosim Options

Tool: Vivado XSIM RTL: Verilog

(b) Generator

10 Block Design

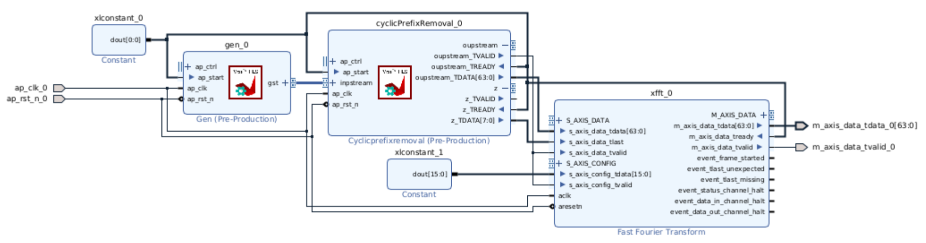


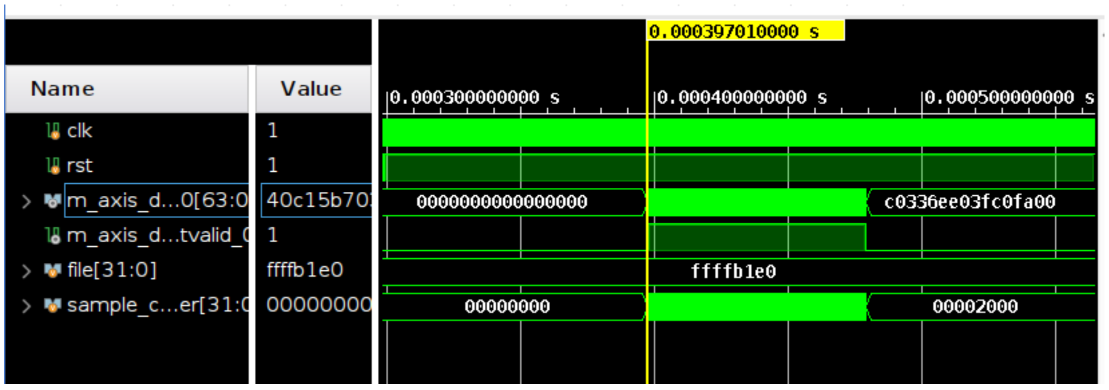
Figure 4: Cosimulation Report

11 Verilog Testbench

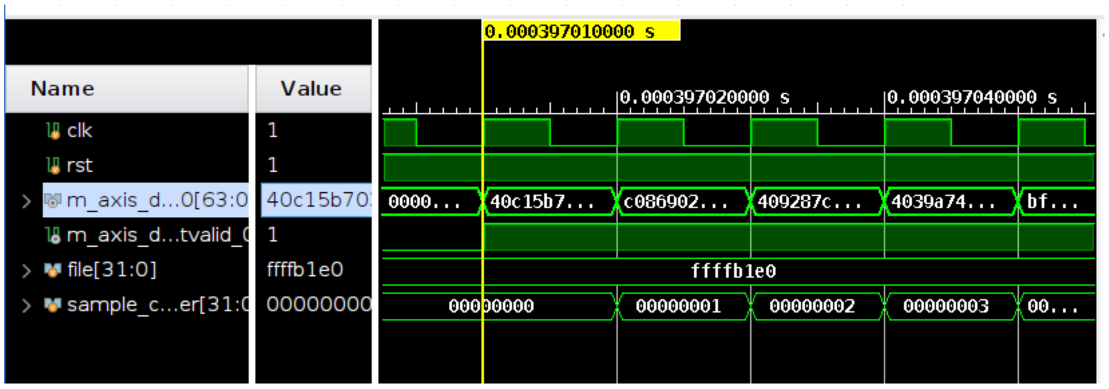
```
// Engineer:
//
// Create Date: 05/25/2023 11:59:07 AM
// Design Name:
// Module Name: wav_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
```

```
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////////////////////


module wav_tb(

    );

  reg clk;
  reg rst;
  wire [63:0]m_axis_data_tdata_0;
  wire m_axis_data_tvalid_0;
  integer file;
  reg [31:0] sample_counter;

  design_1_wrapper i
       (.ap_clk_0(clk),
        .ap_rst_n_0(rst),
        .m_axis_data_tdata_0(m_axis_data_tdata_0),
        .m_axis_data_tvalid_0(m_axis_data_tvalid_0));



        always #5 clk=~clk;
        initial begin

        rst=0;clk=1;
        #100 rst=1;
        #89130000 $finish;
        end

        initial begin

        file=$fopen("fft_output_vivadoip.txt","w");
        sample_counter = 0;

    while (sample_counter < 8192) begin
      #10;  // Assuming a sampling rate of 10 units

        if (m_axis_data_tvalid_0) begin
         $fwrite(file, "%h\n", m_axis_data_tdata_0);
        sample_counter = sample_counter + 1;
      end
    end
    $fclose(file);
  end
endmodule
```

12 Output Waveform



(a) output of FFT



(b) Zoomed format of above figure

13 Matlab Code

```
clc;
close all;
x = [complex(-0.601084,-0.059909) complex(0.571592,0.530777) complex(0.096652,-0
X = fft(x);

disp('FFT Output:');
disp(X);

fileID = fopen('fft_output_matlab.txt', 'w');
fprintf(fileID, 'Real \t Imaginary \n');
for k = 1:length(X)
    fprintf(fileID, '%d \t %d \n', real(X(k)), imag(X(k)));
end
fclose(fileID);
```

14 Conclusion

The Output of FFTIP is matching with Output of Matlab with Precision using this floating Point Converter Online :

`https://www.h-schmidt.net/FloatConverter/IEEE754.html`

GITHUB : `https://github.com/dk-425/Training.git`