HLS ASSIGNMENT-7(KANEKAL KOUSAR)

(Q) Model the FIR Filter design in the most efficient manner possible for the hardware (e.g. small clock period, lower initiation interval, less resource consumption, etc.). After designing, you should compare your output against the reference output generated by the C code for the same set of input vectors, using a self checking testbench that allows for a 5% difference in output values generated by the C code and the HLS code. Use at least two different input vectors. Also, the HLS design should use appropriate fixed point format instead of floating point format wherever applicable. The C code from the website can be integrated as part of your HLS testbench if you name both the design modules differently, for e.g., firFloat for C module and firFixed for HLS module. This will enable you to pass the input vectors to both the modules and compare the outputs, in a single testbench.

HEADER FILE

```
#ifndef FIR_H_
#define FIR_H_

#include <ap_fixed.h>
#include <ap_int.h>

typedef ap_fixed<24,16> int16_dt;
typedef ap_fixed<48,32> int32_dt;


#include<hls_stream.h>
using namespace hls;
// maximum number of inputs that can be handled in one function call
#define MAX_INPUT_LEN    7
// maximum length of filter than can be handled
#define MAX_FLT_LEN      7
// buffer to hold all of the input samples
#define BUFFER_LEN       (MAX_FLT_LEN - 1 + MAX_INPUT_LEN)


//for fixed points
struct input{
	int16_dt in[MAX_INPUT_LEN];
};

struct coeff{
	int16_dt cof[MAX_FLT_LEN];
};

struct output{
	int16_dt out[MAX_INPUT_LEN];
};

//for float points
struct inputf{
       double in[MAX_INPUT_LEN];
};

struct coefff{
       double cof[MAX_FLT_LEN];
};

struct outputf{
       double out[MAX_INPUT_LEN];
};

void intToFloat( stream<input> &inputs, stream<inputf> &outputs, int
length );
void floatToInt( stream<outputf> &inputs, stream<output> &outputs,
int length );

#endif
```

c-DESIGN CODE

```
#include "header.h"
#include<ap_int.h>

void fir_float( stream<coefff> &coeffs, stream<inputf> &inputs,
stream<outputf> &outputs,int length, int filterLength )
{
    double acc;     // accumulator for MACs
    inputf data=inputs.read();
    coefff c=coeffs.read();
    outputf o;

    double insamp[ BUFFER_LEN ];
    for (int i=0;i<BUFFER_LEN;i++){
      insamp[i]=0;
    }

    // put the new samples at the high end of the buffer
    for (int i=0;i<length;i++){
      insamp[filterLength-1+i]=data.in[i];
    }


    // apply the filter to each input sample
    for ( int n = 0; n <filterLength-1+length; n++ ) {
        // calculate output n
      int co=0;
      int inp=filterLength-1+n;
        acc = 0;
        for (int k = 0; k < filterLength; k++ ) {
            acc += (c.cof[co]) * (insamp[inp]);
            co++;
            inp--;
        }
        o.out[n]= acc;
        }

    outputs<<o;
}



void intToFloat(  stream<input> &inputs, stream<inputf> &outputs,
int length )
{
 input i2=inputs.read();
 inputf o1;
 for ( int i = 0; i < length; i++ ) {
 o1.in[i] = (double)i2.in[i];
 }
 outputs<<o1;
}

void floatToInt(stream<outputf> &inputs, stream<output> &outputs,
int length )
{
 outputf i1=inputs.read();
 output o1;
 for (int  i = 0; i < length; i++ ) {
 // add rounding constant
 i1.out[i] += 0.5;
 // bound the values to 16 bits
 if ( i1.out[i] > 32767.0 ) {
 i1.out[i] = 32767.0;
 } else if ( i1.out[i] < -32768.0 ) {
 i1.out[i] = -32768.0;
 }
 // convert
 o1.out[i] = (int)i1.out[i];
 }
 outputs<<o1;
}
```

Hls-design code:

```
#include "header.h"
#include<ap_int.h>

void fir_fix( stream<coeff> &coeffs, stream<input> &inputs,
stream<output> &outputs,ap_int<4> length, ap_int<4> filterLength )
{
    int32_dt acc;     //accumulator
    input data=inputs.read();
//stores input data
    coeff c=coeffs.read();
//stores filter coefficients
    output o1;

    //initialize the buffer to zero
    int16_dt insamp[ BUFFER_LEN ];
    for (int i=0;i<BUFFER_LEN;i++){
             insamp[i]=0;
        }
#pragma HLS ARRAY_RESHAPE variable=insamp block factor=10 dim=1

  //   put the new samples at the high end of the buffer
       for (int i=0;i<length;i++){
#pragma HLS PIPELINE
#pragma HLS LOOP_TRIPCOUNT
       insamp[filterLength-1+i]=data.in[i];
    }


    // apply the filter to each input sample
    for ( int n = 0; n <filterLength-1+length; n++ ){
#pragma HLS UNROLL
#pragma HLS LOOP_TRIPCOUNT
        // calculate output n
       ap_int<8> co=0;
       ap_int<16> inp=filterLength-1+n;
        acc = 0;
        for (int k = 0; k < filterLength; k++ )
#pragma HLS PIPELINE
 {
#pragma HLS LOOP_TRIPCOUNT
            acc += (c.cof[co]) * (insamp[inp]);
            co++;
            inp--;
        }
        o1.out[n]= acc;
        }

    outputs<<o1;

}
```

TEST BENCH CODE

```
#include "header.h"
#include<iostream>
using namespace std;

#include <fstream>

void fir_fix(stream<coeff> &coeffs, stream<input> &inputs, stream<output>
&outputs,ap_int<4> length, ap_int<4> filterLength );
void fir_float( stream<coefff> &coeffs, stream<inputf> &inputs, stream<outputf>
&outputs,int length, int filterLength );




int main(){

       int length=5;
       int filterLength=4;

       input i2;
       input i;
       coefff c2;
       coeff c;


       ifstream inputFile("input.dat");//float
       ifstream inputFile1("input.dat");//fix
       ifstream inputFile2("coeff.dat");
       ifstream inputFile3("coeff.dat");
       ofstream outputfile("out.dat");

       stream<inputf> inputs2; //fir float
       stream<input> inputs;  //fir fixed
       stream<input> input_itf;



       stream<coefff> coeffs2; //fir float
       stream<coeff> coeffs;    //fir fixed

       stream<output> outputs_fti;//fir float
       stream<outputf> outputs2;//fir float
       stream<output> outputs;   //fir fixed


       for (int p=1;p<3;p++){
              for (int j = 0; j < length; j++) {
                         inputFile >> i2.in[j];
                     }input_itf<<i2;
             for (int j = 0; j < length; j++) {
                          inputFile1 >> i.in[j];
                     }inputs<<i;
              for (int j = 0; j <filterLength; j++) {
                         inputFile2 >> c2.cof[j];
                      }coeffs2<<c2;

              for (int j = 0; j < filterLength; j++) {
                         inputFile3 >>c.cof[j];
                           }coeffs<<c;

              //floating point
              intToFloat( input_itf, inputs2, length );
              fir_float(coeffs2,inputs2,outputs2,length,filterLength);
              floatToInt(outputs2,outputs_fti,length);

              fir_fix(coeffs,inputs,outputs,length,filterLength);
              output o1=outputs.read();
              output o2=outputs_fti.read();
                    double sum1=0;
                 int32_dt sum2=0;
                for (int j=0;j<length;j++){
                     sum1+=(int)o2.out[j];
                     sum2+=o1.out[j];
              }
                if (abs(sum1- double(sum2))/double(sum2) > 0.05){
                           cout << "TEST CASE " <<p<<" DID NOT PASSED AS
DIFFERENCE IS MORE THAN 5% " << endl;

                     }
                        else{
                           cout << "TEST CASE "<<p<<" PASSED " << endl;


                        }
                     outputfile<<"TEST CASE :"<<p<<" "<<"(HLS ~= C)"<<endl;
                      outputfile<<"     "<<endl;
                      for (int j = 0; j < length; j++) {
                         outputfile << o1.out[j]<<"  ~=  "<<o2.out[j]<< endl;
                       }
                      outputfile<<" "<<endl;

       }
        inputFile.close();
        inputFile1.close();
        inputFile2.close();
        inputFile3.close();
        outputfile.close();

}
```

Coefficients:

```
 2 3 4 1
0.4 0.3 2 1
```

Input data:

```
 6 7 8 9 1
  6 7 8 9 1
```

Output file:

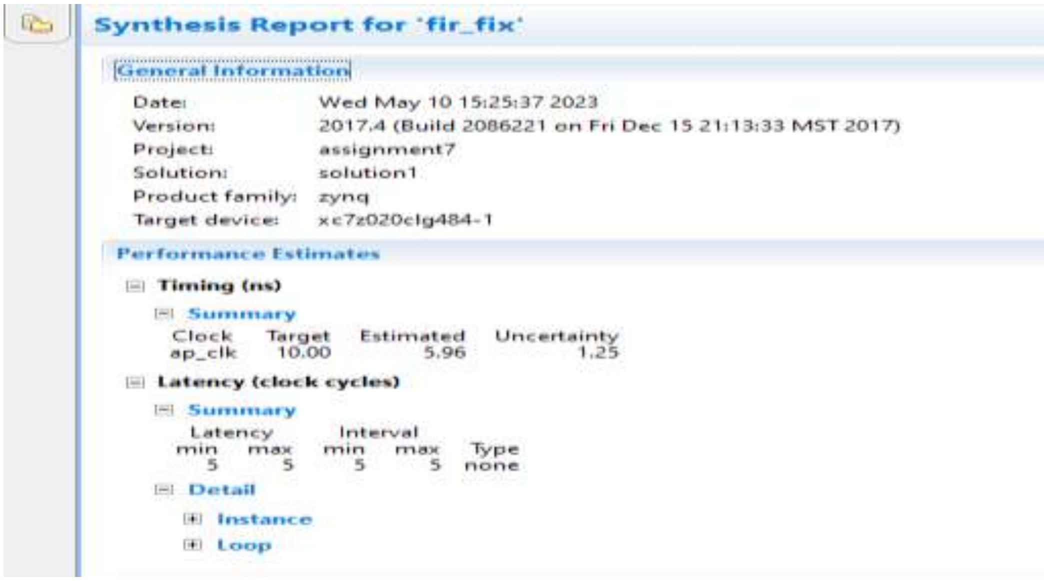
```
TEST CASE :1 (HLS ~= C)

2.39063  ~=  2
4.57031  ~=  5
17.2656  ~=  17
25.9609  ~=  26
26.0703  ~=  26

TEST CASE :2 (HLS ~= C)

12  ~=  12
32  ~=  32
61  ~=  61
76  ~=  76
68  ~=  68
```

Synthesis report:



Synthesis Report for 'fir_fix'

General Information

Date:	Wed May 10 15:25:37 2023
Version:	2017.4 (Build 2086221 on Fri Dec 15 21:13:33 MST 2017)
Project:	assignment7
Solution:	solution1
Product family:	zynq
Target device:	xc7z020clg484-1

Performance Estimates

⊟ Timing (ns)

⊟ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	5.96	1.25

⊟ Latency (clock cycles)

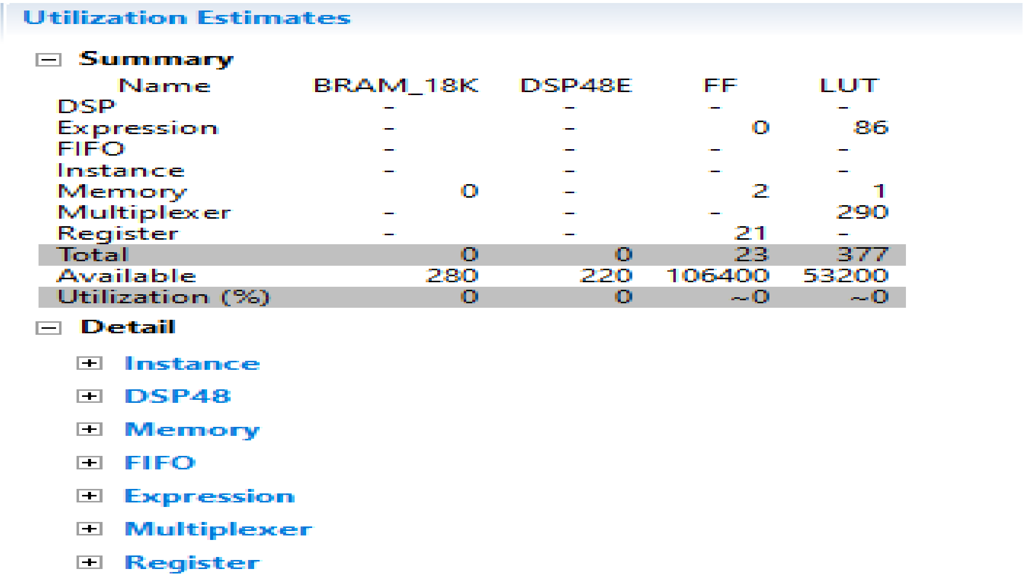
⊟ Summary

Latency		Interval		
min	max	min	max	Type
5	5	5	5	none

⊟ Detail

⊞ Instance

⊞ Loop



Utilization Estimates

⊟ Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	86
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	0	-	2	1
Multiplexer	-	-	-	290
Register	-	-	21	-
Total	0	0	23	377
Available	280	220	106400	53200
Utilization (%)	0	0	~0	~0

⊟ Detail

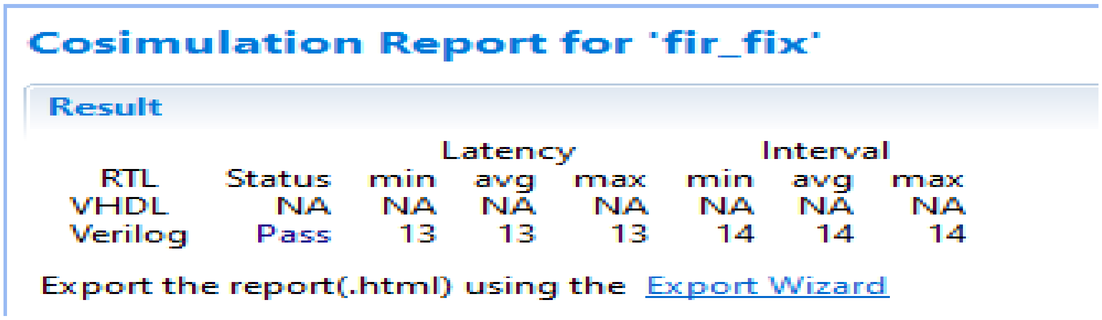
- ⊞ Instance
- ⊞ DSP48
- ⊞ Memory
- ⊞ FIFO
- ⊞ Expression
- ⊞ Multiplexer
- ⊞ Register

Simulation:

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

Co-simulation report:

```
INFO: [Common 17-206] Exiting xsim at Wed May 10 16:04:23 2023...
INFO: [COSIM 212-316] Starting C post checking ...
TEST CASE 1 DID NOT PASSED AS DIFFERENCE IS MORE THAN 5%
TEST CASE 2 DID NOT PASSED AS DIFFERENCE IS MORE THAN 5%
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
Finished C/RTL cosimulation.
```



Cosimulation Report for 'fir_fix'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	13	13	13	14	14	14

Export the report(.html) using the Export Wizard