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 |

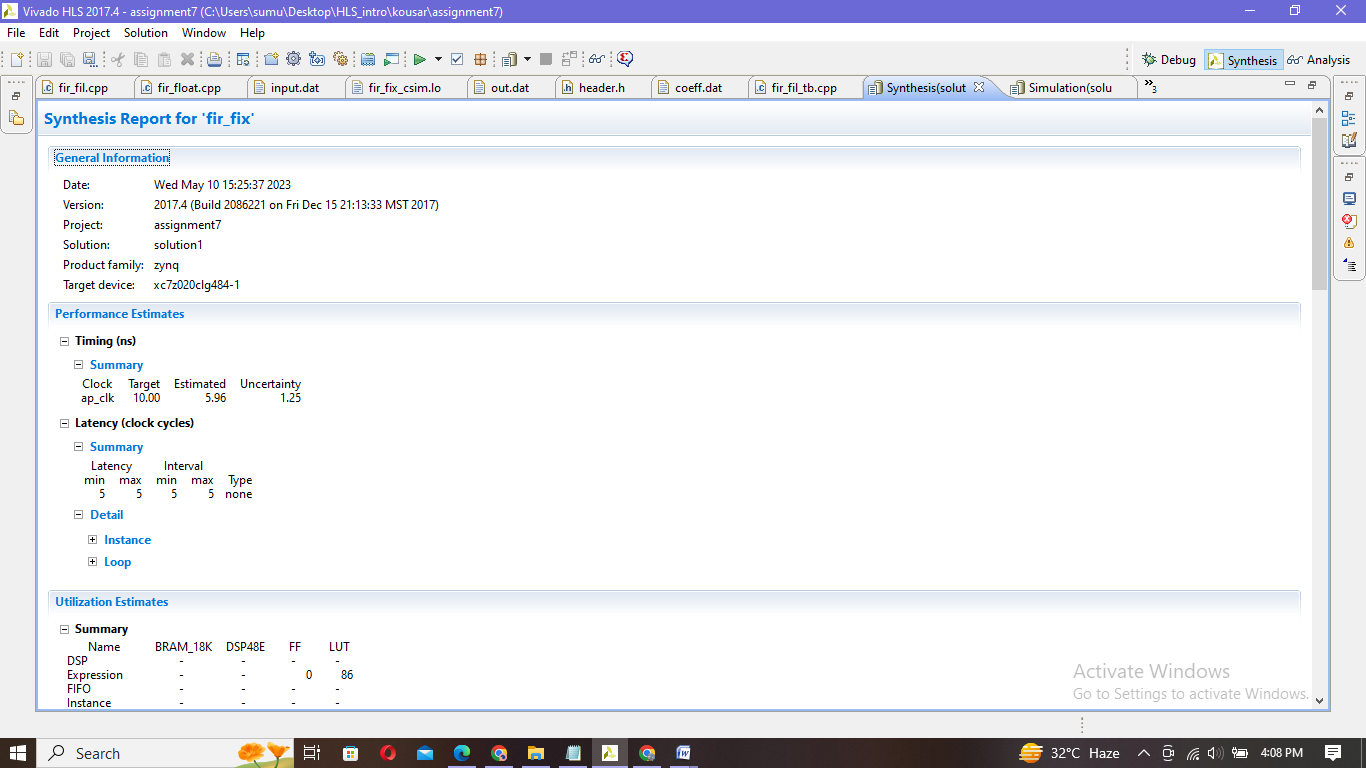
Input data:

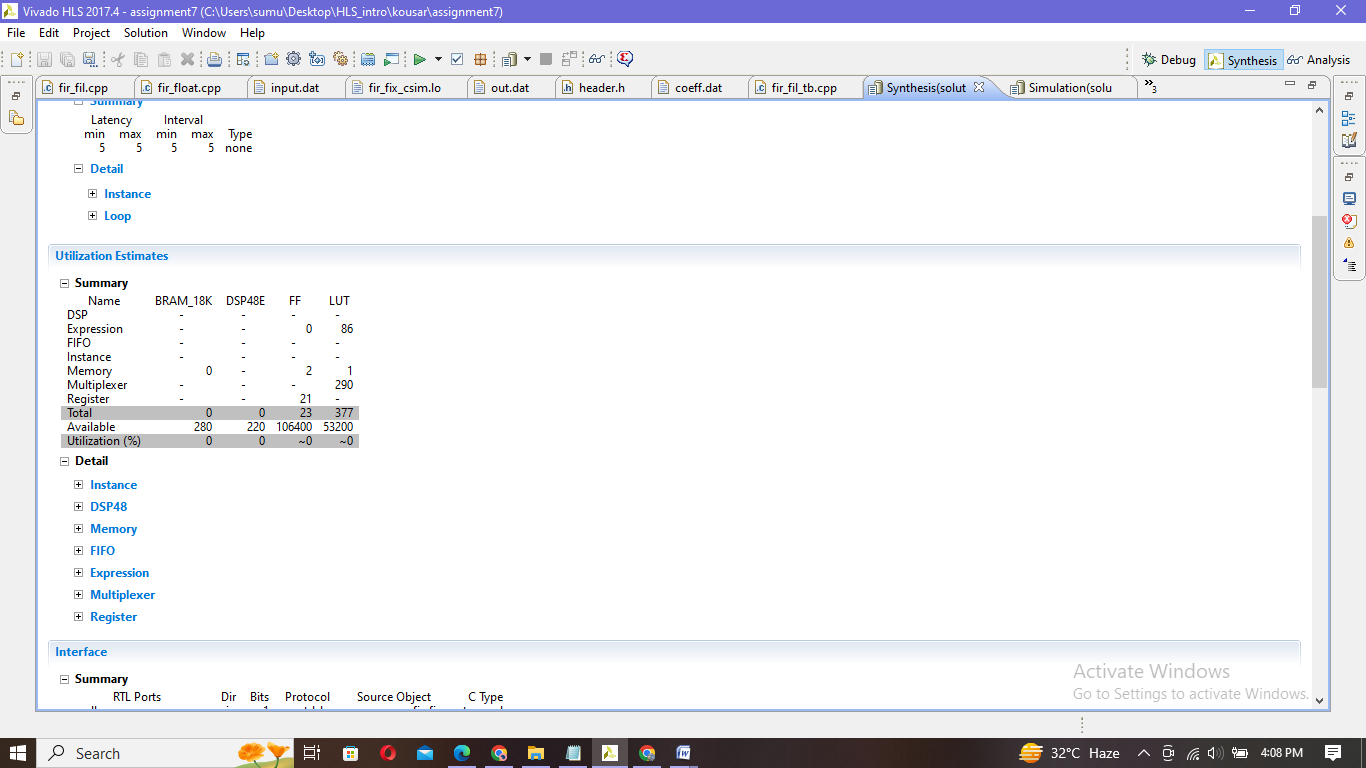
|  |
| --- |
| 6 7 8 9 1  6 7 8 9 1 |

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:





|  |
| --- |
| 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 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

Simulation:

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. |

