HLS ASSIGNMENT4[Kanekal kousar(fwc2022063)]

Q) Design a basic integer ALU unit in HLS that takes 3 inputs: Two 8bit operands and one appropriately sized operator. The permitted operations are ADD, SUB, MUL, DIV, AND, OR and XOR. Use AXI-S ports for all I/O ports. The design should be pipelined

Run C simulation first and verify your design with multiple inputs from a file (you need to create this file as well) containing sets of above 3 inputs and 1 reference output to compare your design output in the testbench. The test bench should be a self-checking testbench. It should run the simulation, pass on inputs, compare the outputs with the reference outputs and write the outputs to another file, along with the Pass or Fail status of that particular test as obtained from the comparison. The testbench should also tell (after the simulation output in the console) at the end of the simulation if all the read input sets passed the test or not.

Header file:

```
#ifndef _HEADER_H_
#define _HEADER_H_

#include <hls_stream.h>
using namespace hls;
#include <ap_int.h>

typedef ap_int<8> int8_i;
typedef ap_int<16> int16_o;
typedef ap_uint<3> uint3_op;

struct inputs {
	int8_i operandone;
	int8_i operandtwo;
	uint3_op operation;
};

#endif
```

Function header file:

```
#ifndef _CALCI_H_
#define _CALCI_H_

#include "header.h"
int16_o add(inputs tmp);
int16_o sub(inputs tmp);
int16_o mul(inputs tmp);
int16_o div(inputs tmp);
int16_o And(inputs tmp);
int16_o Or(inputs tmp);
int16_o Xor(inputs tmp);

int16_o add(inputs tmp)
{
return tmp.operandone + tmp.operandtwo;
```

```
}

int16_o sub(inputs tmp)
{
return tmp.operandone - tmp.operandtwo;
}

int16_o mul(inputs tmp)
{
return tmp.operandone * tmp.operandtwo;
}

int16_o div(inputs tmp)
{
return tmp.operandone / tmp.operandtwo;
}

int16_o And(inputs tmp)
{
return tmp.operandone & tmp.operandtwo;
}

int16_o Or(inputs tmp)
{
return tmp.operandone | tmp.operandtwo;
}

int16_o Xor(inputs tmp)
{
return tmp.operandone ^tmp.operandtwo;
}
#endif
```

C++ code:

```
#include "header.h"
#include "calci.h"

void alu_8bit(stream<inputs> &indata,stream<int16_o> &outdata){
#pragma HLS PIPELINE
#pragma HLS INTERFACE axis  port=outdata
#pragma HLS INTERFACE axis  port=indata
	inputs data=indata.read();
	switch (data.operation)
	{
	case 0:
		outdata.write(add(data));
		break;
	case 1:
			outdata.write(sub(data));
			break;
	case 2:
			outdata.write(mul(data));
			break;
	case 3:
```

```
                        outdata.write(div(data));
                        break;
        case 4:
                        outdata.write(And(data));
                        break;
        case 5:
                        outdata.write(Or(data));
                        break;
        case 6:
                        outdata.write(Xor(data));
                        break;
        }
}
```

Test bench:

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

void alu_8bit(stream<inputs> &indata,stream<int16_o> &outdata);
int main(){
        stream<inputs> indata;
        stream<int16_o> outdata;
        inputs data={0,0,0};
        int op1,op2,op,res;
        char arr[]={'+','-','*','/','&','|','^'};
        int case_fail=0;
        //reading input data from in.dat file and writing output to out.dat file
        ifstream in;
        fstream out;
        in.open("in.dat");
        out.open("out.dat");
        while (in>>op>>op1>>op2>>res){
                        data={op1,op2,op};
                        indata.write(data);
                        alu_8bit(indata,outdata);
                        int16_o O=outdata.read();
                        if (O==res){
        out<<data.operandone<<arr[data.operation]<<data.operandtwo<<"="<<O<<"\t\tpass"<<endl;
                                }
                        else{
                                case_fail++;

        out<<data.operandone<<arr[data.operation]<<data.operandtwo<<"="<<O<<"\t\tfail"<<endl;
                                }}
        if (case_fail==0){
                cout<<"---all test case passed---"<<endl;
        }
        else{
                cout<<"---"<<case_fail<<" test case failed---"<<endl;
        }
        in.close();
        out.close();}
```

In.dat file

```
1  2  3 -1
4  3  4  0
2  4  5  20
5  5  6  7
1  6  7  -1
3  9  8  1
4  7  9  1
2  9  10  90
4  10  11  10
3  11  12  0
2  12  13  156
5  13  14  15
6  14  15  1
6  16  17  1
0  1  18  19
3  16  8  2
```

out.dat file

```
2-3=-1        pass
3&4=0         pass
4*5=20        pass
5|6=7         pass
6-7=-1        pass
9/8=1         pass
7&9=1         pass
9*10=90       pass
10&11=10      pass
11/12=0       pass
12*13=156     pass
13|14=15      pass
14^15=1       pass
16^17=1       pass
1+18=19       pass
16/8=2        pass
```

Synthesis report

Synthesis Report for 'alu_8bit'

General Information

Date:	Fri Mar 24 22:13:46 2023
Version:	2017.4 (Build 2086221 on Fri Dec 15 21:13:33 MST 2017)
Project:	assignment4
Solution:	solution1
Product family:	zynq
Target device:	xc7z020clg484-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	4.17	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
14	14	1	1	function

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	0	0	254
FIFO	-	-	-	-
Instance	-	-	239	160
Memory	-	-	-	-
Multiplexer	-	-	-	173
Register	0	-	322	96
Total	0	0	561	683
Available	280	220	106400	53200
Utilization (%)	0	0	~0	1

Detail

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

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	alu_8bit	return value
ap_rst_n	in	1	ap_ctrl_hs	alu_8bit	return value
ap_start	in	1	ap_ctrl_hs	alu_8bit	return value
ap_done	out	1	ap_ctrl_hs	alu_8bit	return value
ap_idle	out	1	ap_ctrl_hs	alu_8bit	return value
ap_ready	out	1	ap_ctrl_hs	alu_8bit	return value
outdata_V_V_TREADY	in	1	axis	outdata_V_V	pointer
outdata_V_V_TDATA	out	16	axis	outdata_V_V	pointer
outdata_V_V_TVALID	out	1	axis	outdata_V_V	pointer
indata_V_operandone_V_TDATA	in	8	axis	indata_V_operandone_V	pointer
indata_V_operandone_V_TVALID	in	1	axis	indata_V_operandone_V	pointer
indata_V_operandone_V_TREADY	out	1	axis	indata_V_operandone_V	pointer
indata_V_operandtwo_V_TDATA	in	8	axis	indata_V_operandtwo_V	pointer
indata_V_operandtwo_V_TVALID	in	1	axis	indata_V_operandtwo_V	pointer
indata_V_operandtwo_V_TREADY	out	1	axis	indata_V_operandtwo_V	pointer
indata_V_operation_V_TDATA	in	8	axis	indata_V_operation_V	pointer
indata_V_operation_V_TVALID	in	1	axis	indata_V_operation_V	pointer
indata_V_operation_V_TREADY	out	1	axis	indata_V_operation_V	pointer

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

Simulation:

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
make: `csim.exe' is up to date.
---all test case passed---
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Co-simulation:

```
INFO: [Common 17-206] Exiting xsim at Fri Mar 24 23:06:33 2023...
INFO: [COSIM 212-316] Starting C post checking ...
---all test case passed---
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
```

```
Finished C/RTL cosimulation
```

Cosimulation Report for 'alu_8bit'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	14	14	14	1	1	1

Export the report(.html) using the Export Wizard