HLS ASSIGNMENT-6(KANEKAL KOUSAR)

(Q) Implement a 16bit shift register in HLS. The module should take 3 inputs: 16bit data value, 16bit shift value, and 1 bit left or right shift flag. Shift value is the value by which you need to shift (in the direction denoted by shift flag) the data value and produce the output (also 16bit). Implement it in the most efficient manner possible with what you have learned till now. Do C simulation, synthesis and C/RTL co-simulation like other assignments. The testbench should be a self-checking testbench

DESIGN CODE:

```
#include <ap_int.h>
#include <hls_stream.h>
using namespace hls;
typedef ap_uint<1> int1_t;
void shift(short data_value,short shift_value,int1_t shift_flag,short &output){
	int1_t bin[16];
#pragma HLS ARRAY_RESHAPE variable=bin block factor=16 dim=1
	short dec=0;
	//converting to binary
	for (int i=15;i>=0;i--){
#pragma HLS UNROLL
		bin[15-i]=((data_value>>i)& 1);
	}
	//right shift
	if (shift_flag){
		for (int i=15;i>=shift_value;i--){
#pragma HLS LOOP_TRIPCOUNT
			bin[i]=bin[i-shift_value];
		}
		for (int i=0;i<shift_value;i++){
#pragma HLS LOOP_TRIPCOUNT
				bin[i]=0;
			}
	}
	//left shift
	else{
		for (int i=0;i<=15-shift_value;i++){
#pragma HLS LOOP_TRIPCOUNT
				bin[i]=bin[i+shift_value];
			}
		for (int i=15;i>15-shift_value;i--){
#pragma HLS LOOP_TRIPCOUNT
				bin[i]=0;
			}}
	//binary to decimal
	for (int i=0;i<16;i++){
#pragma HLS UNROLL
		if (bin[i]==1){
			dec+=(1<<(16-i-1));
		}}
	output=dec;
}
```

TEST BENCH:

```
#include <ap_int.h>
#include <hls_stream.h>
using namespace hls;
#include <iostream>
using namespace std;
#include <fstream>

typedef ap_uint<1> int1_t;

void shift(short data_value,short shift_value,int1_t shift_flag,short &output);

int main(){
	short data_value;
	short ref,output;
	short shift_value;
	short shift_flag;
	int case_fail=0;

	ifstream in;
	fstream res;
	in.open("input.dat");
	res.open("output.dat");
	while (in>>data_value>>shift_value>>shift_flag>>ref){
		shift(data_value,shift_value,shift_flag,output);

	res<<data_value<<"\t"<<shift_value<<"\t"<<shift_flag<<"\t"<<output<<"\t";
		if (ref==output){
			res<<"passed"<<endl;
		}
		else{
			res<<"fail"<<endl;
			case_fail++;
		}
	}
	if (case_fail==0){
		cout<<"all test cases passed"<<endl;
	}
	else{
		cout<<case_fail<<"\t cases failed"<<endl;
	}
}
```

SYNTHESIS REPORT:

General Information

Date:	Tue Apr 4 12:33:33 2023
Version:	2017.4 (Build 2086221 on Fri Dec 15 21:13:33 MST 2017)
Project:	assignment6
Solution:	solution1
Product family:	zynq
Target device:	xc7z020clg484-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.26	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
8	8	8	8	none

Detail

- Instance
- Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	823
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	134
Register	-	-	296	-
Total	0	0	296	957
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	shift	return value
ap_rst	in	1	ap_ctrl_hs	shift	return value
ap_start	in	1	ap_ctrl_hs	shift	return value
ap_done	out	1	ap_ctrl_hs	shift	return value
ap_idle	out	1	ap_ctrl_hs	shift	return value
ap_ready	out	1	ap_ctrl_hs	shift	return value
data_value	in	16	ap_none	data_value	scalar
shift_value	in	16	ap_none	shift_value	scalar
shift_flag_V	in	1	ap_none	shift_flag_V	scalar
output_r	out	16	ap_vld	output_r	pointer
output_r_ap_vld	out	1	ap_vld	output_r	pointer

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

SIMULATION REPORT:

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

INPUT.DAT

```
10    2    0    40
20    3    1    2
30    1    1    15
40    4    0    640
5     1    1    2
```

OUTPUT.DAT

```
10    2    0    40    passed
20    3    1    2     passed
30    1    1    15    passed
40    4    0    640   passed
5     1    1    2     passed
```

Co-simulation:

```
INFO: [Common 17-206] Exiting xsim at Tue Apr  4 12:38:23 2023...
INFO: [COSIM 212-316] Starting C post checking ...
all test cases passed
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
INFO: [COSIM 212-211] II is measurable only when transaction number is greater than 1
in RTL simulation. Otherwise, they will be marked as all NA. If user wants to calculate
them, please make sure there are at least 2 transactions in RTL simulation.
Finished C/RTL cosimulation.
```

Cosimulation Report for 'shift'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	24	24	24	NA	NA	NA

Export the report(.html) using the Export Wizard