HLS Assignment 3

(KANEKAL KOUSAR[FWC2022063])

Q)Repeat the experiment in assignment 2.3 by configuring the module as pipelining

3.1)

Header file

```
#ifndef MUL32
#define MUL32

#include <iostream>
#include "hls_stream.h"

using namespace std;
using namespace hls;

typedef long long out;
struct inputs{
       int A;
       int B;
};

#endif
```

C++ code:

```
#include "mul32.h"

void mul32(stream<inputs> &din,stream<out> &dout){
#pragma HLS PIPELINE
       inputs data=din.read();
       dout.write(data.A * data.B);
}
```

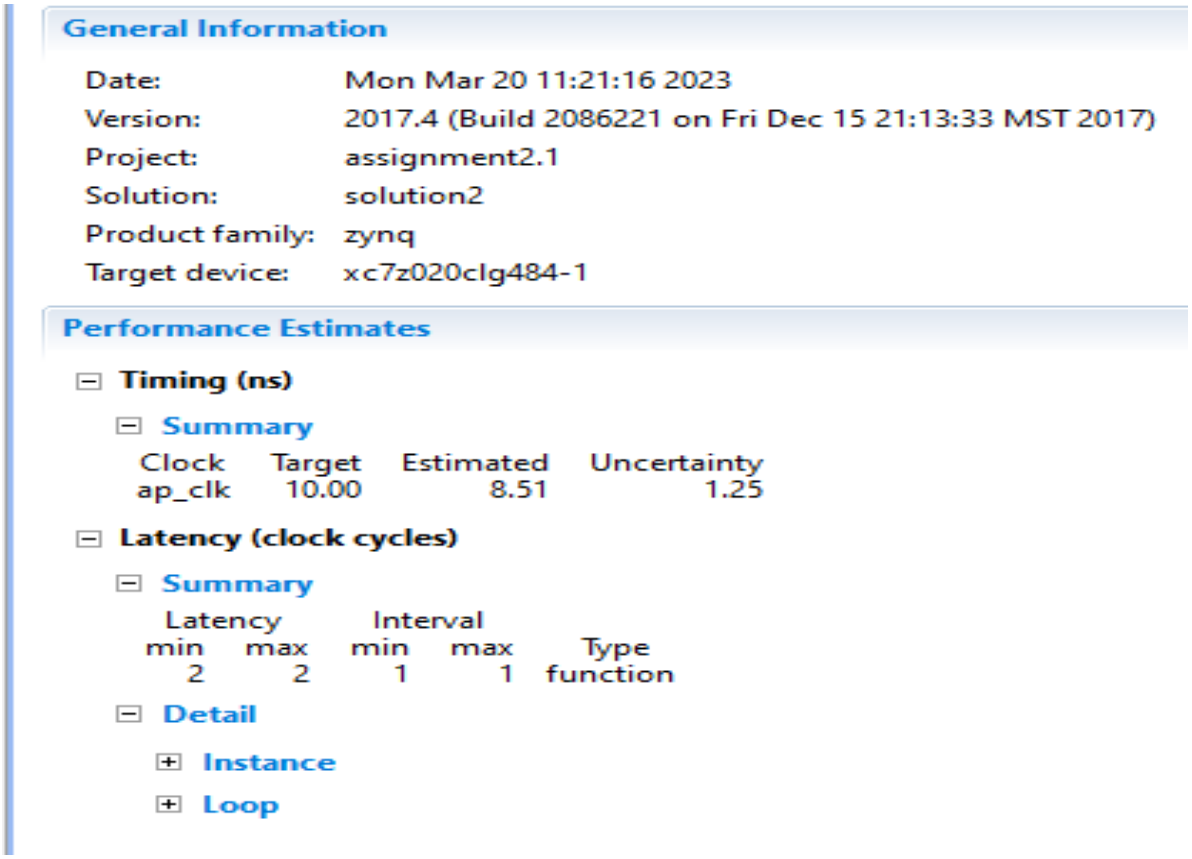
Test bench

```
#include "mul32.h"
void mul32(stream<inputs> &din,stream<long long> &dout);
int main(){
       stream<inputs> indata;
       stream<long long> outdata;
       inputs in;
       long long out;
       int i;
       for (i=0;i<10;i++){
              in.A=i+2;
              in.B=i;
              indata.write(in);
              mul32(indata,outdata);
              outdata>>out;
              cout<<in.A<<"X"<<in.B<<"="<< out <<endl;
       }
       return 0;}
```

Simulation report

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../mul32_tb.cpp in debug mode
   Compiling ../../../mul32.cpp in debug mode
   Generating csim.exe
2X0=0
3X1=3
4X2=8
5X3=15
6X4=24
7X5=35
8X6=48
9X7=63
10X8=80
11X9=99
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis report:



Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	3	0	52
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	27
Register	-	-	99	-
Total	0	3	99	79
Available	280	220	106400	53200
Utilization (%)	0	1	~0	~0

Detail

Instance

DSP48

Memory

FIFO

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
tmp_1_fu_57_p2	*	3	0	20	32	32
din_V_A0_status	and	0	0	8	1	1
ap_block_pp0_stage0_01001	or	0	0	8	1	1
ap_block_state1_pp0_stage0_iter0	or	0	0	8	1	1
ap_enable_pp0	xor	0	0	8	1	2
Total		5	3	0	52	36

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
din_V_A_blk_n	9	2	1	2
din_V_B_blk_n	9	2	1	2
dout_V_blk_n	9	2	1	2
Total	27	6	3	6

Register

Interface

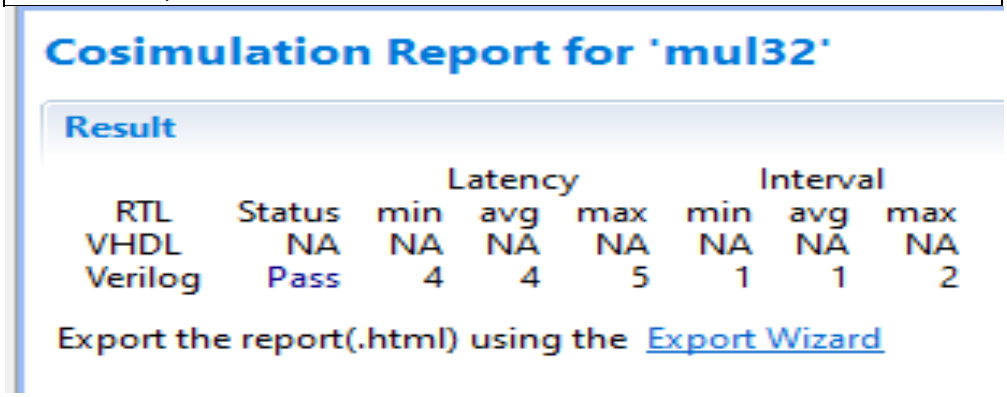
Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	mul32	return value
ap_rst	in	1	ap_ctrl_hs	mul32	return value
ap_start	in	1	ap_ctrl_hs	mul32	return value
ap_done	out	1	ap_ctrl_hs	mul32	return value
ap_idle	out	1	ap_ctrl_hs	mul32	return value
ap_ready	out	1	ap_ctrl_hs	mul32	return value
dout_V_din	out	64	ap_fifo	dout_V	pointer
dout_V_full_n	in	1	ap_fifo	dout_V	pointer
dout_V_write	out	1	ap_fifo	dout_V	pointer
din_V_A_dout	in	32	ap_fifo	din_V_A	pointer
din_V_A_empty_n	in	1	ap_fifo	din_V_A	pointer
din_V_A_read	out	1	ap_fifo	din_V_A	pointer
din_V_B_dout	in	32	ap_fifo	din_V_B	pointer
din_V_B_empty_n	in	1	ap_fifo	din_V_B	pointer
din_V_B_read	out	1	ap_fifo	din_V_B	pointer

Co-simulation

```
INFO: [Common 17-206] Exiting xsim at Mon Mar 20 13:06:30 2023...
INFO: [COSIM 212-316] Starting C post checking ...
2X0=0
3X1=3
4X2=8
5X3=15
6X4=24
```

```
7X5=35
8X6=48
9X7=63
10X8=80
11X9=99
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
Finished C/RTL cosimulation.
```



Cosimulation Report for 'mul32'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	4	4	5	1	1	2

Export the report(.html) using the Export Wizard

3.2)

Header file

```
#ifndef MULFIX
#define MULFIX

#include <iostream>
#include "ap_fixed.h"
#include "hls_stream.h"

using namespace std;
using namespace hls;

typedef ap_ufixed<28,4> fix28_4;
typedef ap_ufixed<56,8> fix56_8;

struct inputs{
	fix28_4 A;
	fix28_4 B;
};

#endif
```

C++ code

```
#include "mul.h"

void mulf(stream<inputs> &din,stream<fix56_8> &dout){
#pragma HLS PIPELINE
	inputs data=din.read();
	dout.write(data.A * data.B);
}
```

Test bench

```
#include "mul.h"
void mulf(stream<inputs> &din,stream<fix56_8> &dout);
int main(){
	stream<inputs> indata;
	stream<fix56_8> outdata;
	int i;

	inputs in={0,0};
	fix56_8 out;
	for (i=0;i<10;i++){
		in.A=i+0.6;
		in.B=i+0.5;
		indata.write(in);
		mulf(indata,outdata);
		outdata>>out;
		cout <<in.A<<"X"<<in.B<<"="<<out<< endl;}}
```

Simulation:

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../mul_tb.cpp in debug mode
   Compiling ../../../mul.cpp in debug mode
   Generating csim.exe
0.6X0.5=0.3
1.6X1.5=2.4
2.6X2.5=6.5
3.6X3.5=12.6
4.6X4.5=20.7
5.6X5.5=30.8
6.6X6.5=42.9
7.6X7.5=57
8.6X8.5=73.1
9.6X9.5=91.2
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis report

Synthesis Report for 'mulf'

General Information

Date:	Mon Mar 20 12:49:36 2023
Version:	2017.4 (Build 2086221 on Fri Dec 15 21:13:33 MST 2017)
Project:	ramesh-project
Solution:	solution2
Product family:	zynq
Target device:	xc7z020clg484-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.45	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
2	2	1	1	function

Detail

- Instance
- Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	3	0	68
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	27
Register	-	-	115	-
Total	0	3	115	95
Available	280	220	106400	53200
Utilization (%)	0	1	~0	~0

Detail

- Instance
- DSP48
- Memory
- FIFO
- Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
r_V_fu_63_p2	*	3	0	36	28	28
din_V_A_V0_status	and	0	0	8	1	1
ap_block_pp0_stage0_01001	or	0	0	8	1	1
ap_block_state1_pp0_stage0_iter0	or	0	0	8	1	1
ap_enable_pp0	xor	0	0	8	1	2
Total		5	3	0	68	32	33

- Multiplexer

Name	LUT	Input Size	Bits	Total Bits
din_V_A_V_blk_n	9	2	1	2
din_V_B_V_blk_n	9	2	1	2
dout_V_V_blk_n	9	2	1	2
Total	27	6	3	6

- Register

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	mulf	return value
ap_rst	in	1	ap_ctrl_hs	mulf	return value
ap_start	in	1	ap_ctrl_hs	mulf	return value
ap_done	out	1	ap_ctrl_hs	mulf	return value
ap_idle	out	1	ap_ctrl_hs	mulf	return value
ap_ready	out	1	ap_ctrl_hs	mulf	return value
dout_V_V_din	out	56	ap_fifo	dout_V_V	pointer
dout_V_V_full_n	in	1	ap_fifo	dout_V_V	pointer
dout_V_V_write	out	1	ap_fifo	dout_V_V	pointer
din_V_A_V_dout	in	28	ap_fifo	din_V_A_V	pointer
din_V_A_V_empty_n	in	1	ap_fifo	din_V_A_V	pointer
din_V_A_V_read	out	1	ap_fifo	din_V_A_V	pointer
din_V_B_V_dout	in	28	ap_fifo	din_V_B_V	pointer
din_V_B_V_empty_n	in	1	ap_fifo	din_V_B_V	pointer
din_V_B_V_read	out	1	ap_fifo	din_V_B_V	pointer

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

Co-simulation:

```
INFO: [Common 17-206] Exiting xsim at Mon Mar 20 14:13:10 2023...
INFO: [COSIM 212-316] Starting C post checking ...
0.6X0.5=0.3
1.6X1.5=2.4
2.6X2.5=6.5
3.6X3.5=12.6
4.6X4.5=20.7
5.6X5.5=30.8
6.6X6.5=42.9
7.6X7.5=57
8.6X8.5=73.1
9.6X9.5=91.2
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
Finished C/RTL cosimulation.
```

Cosimulation Report for 'mulf'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	4	4	5	1	1	2

Export the report(.html) using the Export Wizard

OBSERVATIONS:

Solution1:without pipelining function

Solution2:with pipelining function

(3.1)

All Compared Solutions

solution1: xc7z020clg484-1
solution2: xc7z020clg484-1

Performance Estimates

Timing (ns)

Clock		solution1	solution2
ap_clk	Target	10.00	10.00
	Estimated	8.51	8.51

Latency (clock cycles)

		solution1	solution2
Latency	min	2	2
	max	2	2
Interval	min	2	1
	max	2	1

Utilization Estimates

	solution1	solution2
BRAM_18K	0	0
DSP48E	3	3
FF	99	99
LUT	84	79

(3.2)

Performance Estimates

Timing (ns)

Clock		solution1	solution2
ap_clk	Target	10.00	10.00
	Estimated	7.45	7.45

Latency (clock cycles)

		solution1	solution2
Latency	min	2	2
	max	2	2
Interval	min	2	1
	max	2	1

Utilization Estimates

	solution1	solution2
BRAM_18K	0	0
DSP48E	3	3
FF	115	115
LUT	100	95

- Pipelining allows code to run much faster because, ideally, it permits a new instruction to be issued every clock cycle(II) so that multiple instructions can be run simultaneously over the course of several clock cycles - it does not increase the speed with which an individual instruction can be completed, but the throughput is increased
- By applying the pipelining to function,the performance is increased
  - Initiation Interval (II) is decreased[initiation interval (II) –The number of clock cycles before the function can accept new data]or the throughput is increased
    - Form solution1 to solution2 the II is decreased from 2 to 1
  - The number of resources gets reduced
    - Number of LUT'S reduced

Assignment		Resources		
		LUT	DSP48E	FF
2.3.1	Without pipelining	84	3	99
3.1	With pipelining	79	3	99
2.3.2	Without pipelining	100	3	115
3.2	With pipelining	95	3	115