V.GOKUL KUMAR
Wisig Trainee
Indian Institute of Technology Hyderabad
Email: velicharla@outlook.com

Date:19-03-2023

Hls ASSIGNMENT-2

Repeat the experiments in Assignment 1 with the following three independent changes.
1. Use 32bit inputs and figure out the what the bit width should be for your design using your understanding of digital design and arithmetic operations.
2. Use arbitrary precision data type with bit width of 4 for integer and 24 for fractional part of the inputs and figure out what the bit width of the output should be for your design.
3. Use HLS stream blocking interfaces for the ports in your design.

Record the following observations.
1. Change in resource consumption from Assignment 1 and Assignment 2.1 above. If it changed, what and why it changed? Did the timing report change? Include all code, entire synthesis report, simulation output and C/RTL co-simulation output and report.
2. Change in resource consumption from Assignment 2.1 to Assignment 2.2. If it changed, what and why it changed? Did the timing report change? Include all code, entire synthesis report, simulation output, and C/RTL co-simulation output and report.
3. Repeat Assignment 2.1 and 2.2 with the change mentioned in Assignment 3. Analyse change in timing report from Assignment 1, 2.1, 2.2, and report your understanding of why it changed. Include all code, entire synthesis report, simulation output, and C/RTL co-simulation output and report.

1.

Design:

Multiplier function:

hls_arbitary.cpp | hls_arbitary.h | hls_arbitary_tb.cpp

```cpp
#include "hls_arbitary.h"
void multiplier(dinA_t a, dinB_t b, doutC_t *c)
{
    #pragma HLS INTERFACE ap_none port=a
    #pragma HLS INTERFACE ap_none port=b
    #pragma HLS INTERFACE ap_none port=c
    *c = a * b;
}
```

Header file:

hls_arbitary.cpp | hls_arbitary.h | hls_arbitary_tb.cpp

```
#include <stdio.h>
#include "ap_int.h"
#include <iostream>
using namespace std;
typedef int dinA_t;
typedef int dinB_t;
typedef long long doutC_t;
void multiplier(dinA_t a, dinB_t b, doutC_t *c);

```

Test Bench:

hls_arbitary.cpp | hls_arbitary.h | hls_arbitary_tb.cpp

```
#include "hls_arbitary.h"
int main() {
    doutC_t result;
// Define test inputs
  dinA_t test_inputs[10][2] = {
        {0x00000011, 0x00000001},
        {0x00000011, 0x00000002},
        {0x00000011, 0x00000003},
        {0x00000011, 0x00000004},
        {0x00000011, 0x00000005},
        {0x00000011, 0x00000006},
        {0x00000011, 0x00000007},
        {0x00000011, 0x00000008},
        {0x00000011, 0x00000009},
        {0x00000011, 0x0000000A}
   };
   for (int i = 0; i < 10; i++)
   {
        dinA_t a = test_inputs[i][0];
        dinB_t b = test_inputs[i][1];
        multiplier(a, b, & result);
        cout << a << "*"<< b << "=" << result <<endl;
    // printf("%d * %d = %d\n",a,b,result);
  }
   return 0;
}

```

C simulation printed output:

hls_arbitary.cpp | hls_arbitary.h | hls_arbitary_tb.cpp | multiplier_csim.log

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/hls_arbitary_tb.cpp in debug mode
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/hls_arbitary.cpp in debug mode
   Generating csim.exe
17*1=17
17*2=34
17*3=51
17*4=68
17*5=85
17*6=102
17*7=119
17*8=136
17*9=153
17*10=170
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis Report:

Synthesis Summary Report of 'multiplier'

General Information

Date:	Sun Mar 19 14:15:10 2023	Solution:	solution1 (Vivado IP Flow Target)
Version:	2022.2 (Build 3670227 on Oct 13 2022)	Product family:	zynq
Project:	project_7	Target device:	xc7z020-clg484-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	6.912 ns	2.70 ns

Performance & Resource Estimates

Modules Loops

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
multiplier				-	1	10.000	-	2	-	no	0	3	167	64	0

Performance Pragma

Modules & Loops	Target TI(cycles)	TI(cycles)	TI met
multiplier	-	-	-

HW Interfaces

Other Ports

Interface	Mode	Bitwidth
a	ap_none	32
b	ap_none	32
c	ap_none	64

TOP LEVEL CONTROL

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst	reset	ap_rst
ap_ctrl	ap_ctrl_hs	ap_done ap_idle ap_ready ap_start

SW I/O Information

Top Function Arguments

Argument	Direction	Datatype
a	in	int
b	in	int
c	out	long long int*

SW-to-HW Mapping

Argument	HW Interface	HW Type
a	a	port
b	b	port
c	c	port

Pragma Report

Valid Pragma Syntax

Bind Op Report

No filter settings

Name	DSP	Pragma	Variable	Op	Impl	Latency
multiplier	3					
mul_32s_32s_32_2_1_U1	3		mul_ln7	mul	auto	1

No user config_op information

Bind Storage Report

No bind storage, config_storage information

Cosimulation printed output:

```
////////////////////////////////////////////////////////////////////////////////
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @ "Simulation Time"
////////////////////////////////////////////////////////////////////////////////
// RTL Simulation : 0 / 10 [0.00%] @ "125000"
// RTL Simulation : 1 / 10 [100.00%] @ "155000"
// RTL Simulation : 2 / 10 [100.00%] @ "175000"
// RTL Simulation : 3 / 10 [100.00%] @ "195000"
// RTL Simulation : 4 / 10 [100.00%] @ "215000"
// RTL Simulation : 5 / 10 [100.00%] @ "235000"
// RTL Simulation : 6 / 10 [100.00%] @ "255000"
// RTL Simulation : 7 / 10 [100.00%] @ "275000"
// RTL Simulation : 8 / 10 [100.00%] @ "295000"
// RTL Simulation : 9 / 10 [100.00%] @ "315000"
// RTL Simulation : 10 / 10 [100.00%] @ "335000"
////////////////////////////////////////////////////////////////////////////////
$finish called at time : 395 ns : File "C:/Users/velic/AppData/Roaming/Xilinx/Vitis/project_7/solution1/sim/verilog/multiplier.autotb.v" Line 333
## quit
INFO: [Common 17-206] Exiting xsim at Sun Mar 19 14:17:42 2023...
INFO: [COSIM 212-316] Starting C post checking ...
17*1=17
17*2=34
17*3=51
17*4=68
17*5=85
17*6=102
17*7=119
17*8=136
17*9=153
17*10=170
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
INFO: [HLS 200-111] Finished Command cosim_design CPU user time: 1 seconds. CPU system time: 1 seconds. Elapsed time: 25.416 seconds; current allocated memory: 10.555 MB
INFO: [HLS 200-112] Total CPU user time: 3 seconds. Total CPU system time: 2 seconds. Total elapsed time: 37.868 seconds; peak allocated memory: 106.996 MB.
Finished C/RTL cosimulation.
```

Cosimulation Report:

Cosimulation Report for 'multiplier'

General Information

Date: Sun Mar 19 14:17:43 IST 2023
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: project_7
Status: Pass

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg484-1

Cosim Options

Tool: Vivado XSIM

RTL: Verilog

Performance Estimates

Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
multiplier	2	2	2	1	1	1

- As the bit width increases, more hardware resources are required to process the larger data. This can lead to an increase in resource consumption, such as a larger number of flip-flops, DSP, LUT's
- With larger data sizes, there may be an increase in data movement between different hardware modules, such as between memory and processing units. This increase in data movement can result in longer latencies and slower timing

2.
Design:

Multiplier function:

hls_fixedpoint.cpp | hls_fixedpoint.h | hls_fixedpoint_tb.cpp

```
#include "hls_fixedpoint.h"
void multiplier(dinA_t a, dinB_t b, doutC_t *c)
{
    #pragma HLS INTERFACE ap_none port=a
    #pragma HLS INTERFACE ap_none port=b
    #pragma HLS INTERFACE ap_none port=c
    *c = a * b;
}
```

Header file:

hls_fixedpoint.cpp | hls_fixedpoint_tb.cpp | multiplier_csim.log

```
#include <stdio.h>
#include "ap_fixed.h"
#include <iostream>
using namespace std;
typedef ap_fixed<28,4> dinA_t;
typedef ap_fixed<28,4> dinB_t;
typedef ap_fixed<56,8> doutC_t;
void multiplier(dinA_t a, dinB_t b, doutC_t *c);
```

Test Bench:

hls_fixedpoint.cpp | multiplier_csim.log | Synthesis Summary(solution1) | C

```cpp
#include"hls_fixedpoint.h"
int main()
{
    doutC_t result;
    dinA_t test_inputs[10][2] = {
            {4.125,3.264589},
            {4.56215,2.4568912},
            {3.69845632,2.456891},
            {3.69845632,2.456891},
            {1.25698456,3.569874},
            {1.4567896512,5.623366},
            {5.6684258,6.54566321},
            {6.2255620,5.636115},
            {4.6985230,2.5698523},
            {1.563995,3.2656343}
      };
     for (int i = 0; i < 10; i++)
      {
         dinA_t a=test_inputs[i][0];
         dinB_t b=test_inputs[i][1];
          multiplier(a,b,&result);
        std::cout<< a << "*"<< b << "="<< result<<std::endl;
        //printf("%d * %d= %d\n",a,b,result);
      }
     return 0;
}
```

C simulation printed output:

hls_fixedpoint.cpp | multiplier_csim.log | Synthesis Summary(solution1) | Co-simulation Report(solutio... | hls_fi

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/hls_fixedpoint_tb.cpp in debug mode
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/hls_fixedpoint.cpp in debug mode
   Generating csim.exe
4.125*3.26459=13.4664
4.56215*2.45689=11.2087
3.69846*2.45689=9.0867
3.69846*2.45689=9.0867
1.25698*3.56987=4.48728
1.45679*5.62337=8.19206
5.66843*6.54566=37.1036
6.22556*5.63611=35.088
4.69852*2.56985=12.0745
1.56399*3.26563=5.10744
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis Report:

Synthesis Summary Report of 'multiplier'

General Information

Date: Sun Mar 19 14:37:34 2023
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: project_8

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg484-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.080 ns	2.70 ns

Performance & Resource Estimates

Modules Loops

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
multiplier				-	2	20.000	-	3	-	no	0	3	144	132	0

Performance Pragma

Modules & Loops	Target TI(cycles)	TI(cycles)	TI met
multiplier	-	-	-

HW Interfaces

Other Ports

Interface	Mode	Bitwidth
a	ap_none	28
b	ap_none	28
c	ap_none	56

TOP LEVEL CONTROL

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst	reset	ap_rst
ap_ctrl	ap_ctrl_hs	ap_done ap_idle ap_ready ap_start

SW I/O Information

Top Function Arguments

Argument	Direction	Datatype
a	in	ap_fixed<28 4 AP_TRN AP_WRAP 0>
b	in	ap_fixed<28 4 AP_TRN AP_WRAP 0>
c	out	ap_fixed<56 8 AP_TRN AP_WRAP 0>*

SW-to-HW Mapping

Argument	HW Interface	HW Type
a	a	port
b	b	port
c	c	port

Pragma Report

Valid Pragma Syntax

Bind Op Report

No filter settings

Name	DSP	Pragma	Variable	Op	Impl	Latency
multiplier	3					
mul_28s_28s_56_3_1_U1	3		r_V	mul	auto	2

No user config_op information

Bind Storage Report

No bind storage, config_storage information

Cosimulation printed output:

```
////////////////////////////////////////////////////////////////////////////////
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @ "Simulation Time"
////////////////////////////////////////////////////////////////////////////////
// RTL Simulation : 0 / 10 [0.00%] @ "125000"
// RTL Simulation : 1 / 10 [100.00%] @ "165000"
// RTL Simulation : 2 / 10 [100.00%] @ "195000"
// RTL Simulation : 3 / 10 [100.00%] @ "225000"
// RTL Simulation : 4 / 10 [100.00%] @ "255000"
// RTL Simulation : 5 / 10 [100.00%] @ "285000"
// RTL Simulation : 6 / 10 [100.00%] @ "315000"
// RTL Simulation : 7 / 10 [100.00%] @ "345000"
// RTL Simulation : 8 / 10 [100.00%] @ "375000"
// RTL Simulation : 9 / 10 [100.00%] @ "405000"
// RTL Simulation : 10 / 10 [100.00%] @ "435000"
////////////////////////////////////////////////////////////////////////////////
$finish called at time : 495 ns : File "C:/Users/velic/AppData/Roaming/Xilinx/Vitis/project_8/solution1/sim/verilog/multiplier.autotb.v" Line 333
## quit
INFO: [Common 17-206] Exiting xsim at Sun Mar 19 14:38:37 2023...
INFO: [COSIM 212-316] Starting C post checking ...
4.125*3.26459=13.4664
4.56215*2.45689=11.2087
3.69846*2.45689=9.0867
1.25698*3.56987=4.48728
1.45679*5.62337=8.19206
5.66843*6.54566=37.1036
6.22556*5.63611=35.088
4.69852*2.56985=12.0745
1.56399*3.26563=5.10744
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
INFO: [HLS 200-111] Finished Command cosim_design CPU user time: 1 seconds. CPU system time: 1 seconds. Elapsed time: 40.636 seconds; current allocated memory: 9.508 MB
INFO: [HLS 200-112] Total CPU user time: 3 seconds. Total CPU system time: 2 seconds. Total elapsed time: 53.214 seconds; peak allocated memory: 106.727 MB.
Finished C/RTL cosimulation.
```

Cosimulation Report:

Cosimulation Report for 'multiplier'

General Information

Date: Sun Mar 19 14:38:38 IST 2023
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: project_8
Status: Pass

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg484-1

Cosim Options

Tool: Vivado XSIM

RTL: Verilog

Performance Estimates

Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
multiplier	3	3	3	2	2	2

- fixed-point arithmetic can be more hardware-efficient than arbitrary arithmetic. This is because fixed-point arithmetic uses a fixed number of bits to represent both the integer and fractional parts of a number

- In terms of timing requirements, fixed-point arithmetic can offer faster computation times than arbitrary precision arithmetic.

3.

Design:

Multiplier function:

Synthesis Summary(solution1) | hls_stream1.cpp | hls_stream1_tb.cpp | hls_stream1.h | multiply_cs

```cpp
#include"hls_stream1.h"
void multiply(hls::stream<dinA_t> &A, hls::stream<dinB_t>& B,hls::stream<doutC_t> &C)
{
    dinA_t a;
    dinB_t b;
    doutC_t result;
    a=A.read();
    b=B.read();
    result=a*b;
    C.write(result);
}
```

Header file:

hls_stream1.cpp | hls_stream1_tb.cpp | hls_stream1.h | multiply_csim.log | Co-simulation Repo

```cpp
#include<stdio.h>
#include<hls_stream.h>
#include "ap_int.h"
typedef int dinA_t;
typedef int dinB_t;
typedef long long doutC_t;
void multiply(hls::stream<dinA_t>&A,hls::stream<dinB_t>&B,hls::stream<doutC_t>&C);

```

Test Bench:

hls_stream1.cpp | hls_stream1_tb.cpp | hls_stream1.h | multiply_csim.log

```cpp
#include"hls_stream1.h"
int main()
{
    hls::stream<dinA_t> a;
    hls::stream<dinB_t> b;
    hls::stream<doutC_t> c;
    dinA_t test_inputs[10][2] = {
            {0x00000011,0x00000001},
            {0x00000011,0x00000002},
            {0x00000011,0x00000003},
            {0x00000011,0x00000004},
            {0x00000011,0x00000005},
            {0x00000011,0x00000006},
            {0x00000011,0x00000007},
            {0x00000011,0x00000008},
            {0x00000011,0x00000009},
            {0x00000011,0x0000000A},
      };
      for (int i = 0; i < 10; i++)
      {
          a.write(test_inputs[i][0]);
          b.write(test_inputs[i][1]);
          multiply(a,b,c);
          //out >> result;
        std::cout<<  test_inputs[i][0] << "*"<< test_inputs[i][1] << "="<<c.read()<<std::endl;
      }
}
```

C simulation printed output:

hls_stream1.cpp | hls_stream1_tb.cpp | hls_stream1.h | multiply_csim.log | Synthesis Summary(solution1)

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/hls_stream1_tb.cpp in debug mode
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/hls_stream1.cpp in debug mode
   Generating csim.exe
17*1=17
17*2=34
17*3=51
17*4=68
17*5=85
17*6=102
17*7=119
17*8=136
17*9=153
17*10=170
INFO [HLS SIM]: The maximum depth reached by any hls::stream() instance in the design is 1
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis Report:

Synthesis Summary Report of 'multiply'

General Information

Date:	Sun Mar 19 13:34:02 2023	Solution:	solution1 (Vivado IP Flow Target)
Version:	2022.2 (Build 3670227 on Oct 13 2022)	Product family:	zynq
Project:	project_11	Target device:	xc7z020-clg484-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	6.912 ns	2.70 ns

Performance & Resource Estimates

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
multiply				-	3	30.000	-	4	-	no	0	3	265	104	0

Performance Pragma

Modules & Loops	Target TI(cycles)	TI(cycles)	TI met
multiply	-	-	-

HW Interfaces

AP_FIFO

Interface	Data Width
A	32
B	32
C	64

TOP LEVEL CONTROL

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst	reset	ap_rst
ap_ctrl	ap_ctrl_hs	ap_done ap_idle ap_ready ap_start

SW I/O Information

Top Function Arguments

Argument	Direction	Datatype
A	in	stream<int 0>&
B	in	stream<int 0>&
C	out	stream<long 0>&

SW-to-HW Mapping

Argument	HW Interface	HW Type
A	A	interface
B	B	interface
C	C	interface

Bind Op Report

No filter settings

Name	DSP	Pragma	Variable	Op	Impl	Latency
multiply	3					
mul_32s_32s_32_2_1_U1	3		result	mul	auto	1

No user config_op information

Bind Storage Report

No bind storage, config_storage information

Cosimulation printed output:

```
////////////////////////////////////////////////////////////////////////////////
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @ "Simulation Time"
////////////////////////////////////////////////////////////////////////////////
// RTL Simulation : 0 / 10 [0.00%] @ "125000"
// RTL Simulation : 1 / 10 [100.00%] @ "175000"
// RTL Simulation : 2 / 10 [100.00%] @ "215000"
// RTL Simulation : 3 / 10 [100.00%] @ "255000"
// RTL Simulation : 4 / 10 [100.00%] @ "295000"
// RTL Simulation : 5 / 10 [100.00%] @ "335000"
// RTL Simulation : 6 / 10 [100.00%] @ "375000"
// RTL Simulation : 7 / 10 [100.00%] @ "415000"
// RTL Simulation : 8 / 10 [100.00%] @ "455000"
// RTL Simulation : 9 / 10 [100.00%] @ "495000"
// RTL Simulation : 10 / 10 [100.00%] @ "535000"
////////////////////////////////////////////////////////////////////////////////
$finish called at time : 595 ns : File "C:/Users/velic/AppData/Roaming/Xilinx/Vitis/project_11/solution1/sim/verilog/multiply.autotb.v" Line 324
## quit
INFO: [Common 17-206] Exiting xsim at Sun Mar 19 13:49:47 2023...
INFO: [COSIM 212-316] Starting C post checking ...
17*1=17
17*2=34
17*3=51
17*4=68
17*5=85
17*6=102
17*7=119
17*8=136
17*9=153
17*10=170
INFO [HLS SIM]: The maximum depth reached by any hls::stream() instance in the design is 1
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
INFO: [HLS 200-111] Finished Command cosim_design CPU user time: 2 seconds. CPU system time: 1 seconds. Elapsed time: 36.937 seconds; current allocated memory: 12.664 MB
INFO: [HLS 200-112] Total CPU user time: 4 seconds. Total CPU system time: 2 seconds. Total elapsed time: 49.474 seconds; peak allocated memory: 108.566 MB.
Finished C/RTL cosimulation.
```

Cosimulation Report:

Cosimulation Report for 'multiply'

General Information

Date: Sun Mar 19 13:49:47 IST 2023
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: project_11
Status: Pass

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg484-1

Cosim Options

Tool: Vivado XSIM

RTL: Verilog

Performance Estimates

Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
multiply	4	4	4	3	3	3

4.

Design:

Multiplier function:

hls_stream2_tb.cpp | Synthesis Summary(solution1) | Co-simulation Report(solution1) | hls_stream2.h

```
#include"hls_stream2.h"
void multiply(hls::stream<dinA_t> &A, hls::stream<dinB_t>& B,hls::stream<doutC_t> &C)
{
    dinA_t a;
    dinB_t b;
    doutC_t result;
    a=A.read();
    b=B.read();
    result=a*b;
    C.write(result);
}
```

Header file:

hls_stream2_tb.cpp | Synthesis Summary(solution1) | Co-simulation Report(solution1) | hls_stream2.h

```
#include<stdio.h>
#include<ap_fixed.h>
#include<hls_stream.h>
typedef ap_fixed <28,4> dinA_t;
typedef ap_fixed <28,4> dinB_t;
typedef ap_fixed <56,8> doutC_t;
void multiply(hls::stream<dinA_t>&A,hls::stream<dinB_t>&B,hls::stream<doutC_t>&C);
```

Test Bench:

hls_stream2.h | hls_stream2.cpp | hls_stream2_tb.cpp | Synthesis Summary(solution1) | Co-simulation Report(solution1

```
#include"hls_stream2.h"
int main()
{
    hls::stream<dinA_t> a;
    hls::stream<dinB_t> b;
    hls::stream<doutC_t> c;
    dinA_t test_inputs[10][2] = {
            {4.125,3.264589},
            {4.56215,2.4568912},
            {3.69845632,2.456891},
            {3.69845632,2.456891},
            {1.25698456,3.569874},
            {1.4567896512,5.623366},
            {5.6684258,6.54566321},
            {6.2255620,5.636115},
            {4.6985230,2.5698523},
            {1.563995,3.2656343}
      };
     for (int i = 0; i < 10; i++)
      {
          a.write(test_inputs[i][0]);
          b.write(test_inputs[i][1]);
          multiply(a,b,c);
          //out >> result;
        std::cout<<  test_inputs[i][0] << "*"<< test_inputs[i][1] << "="<<c.read()<<std::endl;
      }
}
```

C simulation printed output:

hls_stream2.h | hls_stream2.cpp | hls_stream2_tb.cpp | Synthesis Summary(solution1) | Co-simulation Report(sc

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/hls_stream2_tb.cpp in debug mode
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/hls_stream2.cpp in debug mode
   Generating csim.exe
4.125*3.26459=13.4664
4.56215*2.45689=11.2087
3.69846*2.45689=9.0867
3.69846*2.45689=9.0867
1.25698*3.56987=4.48728
1.45679*5.62337=8.19206
5.66843*6.54566=37.1036
6.22556*5.63611=35.088
4.69852*2.56985=12.0745
1.56399*3.26563=5.10744
INFO [HLS SIM]: The maximum depth reached by any hls::stream() instance in the design is 1
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis Report:

Synthesis Summary Report of 'multiply'

General Information

Date:	Sun Mar 19 13:12:43 2023	Solution:	solution1 (Vivado IP Flow Target)
Version:	2022.2 (Build 3670227 on Oct 13 2022)	Product family:	artix7
Project:	project_12	Target device:	xc7a200t-fbg676-2

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	6.216 ns	2.70 ns

Performance & Resource Estimates

Modules Loops

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
multiply				-	2	20.000	-	3	-	no	0	3	115	63	0

Performance Pragma

Modules & Loops	Target TI(cycles)	TI(cycles)	TI met
multiply	-	-	-

HW Interfaces

AP_FIFO

Interface	Data Width
A	28
B	28
C	56

TOP LEVEL CONTROL

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst	reset	ap_rst
ap_ctrl	ap_ctrl_hs	ap_done ap_idle ap_ready ap_start

SW I/O Information

Top Function Arguments

Argument	Direction	Datatype
A	in	stream<ap_fixed<28 4 AP_TRN AP_WRAP 0> 0>&
B	in	stream<ap_fixed<28 4 AP_TRN AP_WRAP 0> 0>&
C	out	stream<ap_fixed<56 8 AP_TRN AP_WRAP 0> 0>&

SW I/O Information

Top Function Arguments

Argument	Direction	Datatype
A	in	stream<ap_fixed<28 4 AP_TRN AP_WRAP 0> 0>&
B	in	stream<ap_fixed<28 4 AP_TRN AP_WRAP 0> 0>&
C	out	stream<ap_fixed<56 8 AP_TRN AP_WRAP 0> 0>&

SW-to-HW Mapping

Argument	HW Interface	HW Type
A	A	interface
B	B	interface
C	C	interface

Bind Op Report

No filter settings

Name	DSP	Pragma	Variable	Op	Impl	Latency
multiply	3					
mul_28s_28s_56_1_1_U1	3		r_V_1	mul	auto	0

No user config_op information

Bind Storage Report

No bind storage, config_storage information

Cosimulation printed output:

```
////////////////////////////////////////////////////////////////////////////////
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @ "Simulation Time"
////////////////////////////////////////////////////////////////////////////////
// RTL Simulation : 0 / 10 [0.00%] @ "125000"
// RTL Simulation : 1 / 10 [100.00%] @ "165000"
// RTL Simulation : 2 / 10 [100.00%] @ "195000"
// RTL Simulation : 3 / 10 [100.00%] @ "225000"
// RTL Simulation : 4 / 10 [100.00%] @ "255000"
// RTL Simulation : 5 / 10 [100.00%] @ "285000"
// RTL Simulation : 6 / 10 [100.00%] @ "315000"
// RTL Simulation : 7 / 10 [100.00%] @ "345000"
// RTL Simulation : 8 / 10 [100.00%] @ "375000"
// RTL Simulation : 9 / 10 [100.00%] @ "405000"
// RTL Simulation : 10 / 10 [100.00%] @ "435000"
////////////////////////////////////////////////////////////////////////////////
$finish called at time : 495 ns : File "C:/Users/velic/AppData/Roaming/Xilinx/Vitis/project_12/solution1/sim/verilog/multiply.autotb.v" Line 324
## quit
INFO: [Common 17-206] Exiting xsim at Sun Mar 19 13:14:08 2023...
INFO: [COSIM 212-316] Starting C post checking ...
4.125*3.26459=13.4664
4.56215*2.45689=11.2087
3.69846*2.45689=9.0867
1.25698*3.56987=4.48728
1.45679*5.62337=8.19206
5.66843*6.54566=37.1036
6.22556*5.63611=35.088
4.69852*2.56985=12.0745
1.56399*3.26563=5.10744
INFO [HLS SIM]: The maximum depth reached by any hls::stream() instance in the design is 1
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
INFO: [HLS 200-111] Finished Command cosim_design CPU user time: 1 seconds. CPU system time: 1 seconds. Elapsed time: 64.537 seconds; current allocated memory: 11.793 MB
INFO: [HLS 200-112] Total CPU user time: 3 seconds. Total CPU system time: 2 seconds. Total elapsed time: 77.21 seconds; peak allocated memory: 108.477 MB.
Finished C/RTL cosimulation.
```

Cosimulation Report:

Cosimulation Report for 'multiply'

General Information

Date: Sun Mar 19 13:14:15 IST 2023
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: project_12
Status: Pass

Solution: solution1 (Vivado IP Flow Target)
Product family: artix7
Target device: xc7a200t-fbg676-2

Cosim Options

Tool: Vivado XSIM

RTL: Verilog

Performance Estimates

Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
multiply	3	3	3	2	2	2

- HLS stream interface, particularly when it comes to timing requirements. Specifically, because the stream interface is designed to process data in a pipelined fashion, it may require more complex timing considerations than traditional fixed or arbitrary data types. This can make it more difficult to ensure that the design meets timing requirements, particularly when working with high-speed data streams.

- HLS stream interface can help reduce resource consumption, particularly when dealing with large data sets. This is because the stream interface enables data to be processed in a pipelined fashion, which can help reduce the amount of storage and processing resources required to handle large data sets.

- stream interface can also help to reduce the overall memory footprint of a design, since data can be streamed in and out

Github:

https://github.com/velicharlagokulkumar/vitis-hls/tree/main/project_7
https://github.com/velicharlagokulkumar/vitis-hls/tree/main/project_8
https://github.com/velicharlagokulkumar/vitis-hls/tree/main/project_11
https://github.com/velicharlagokulkumar/vitis-hls/tree/main/project_12