V.GOKUL KUMAR
FUTURE WIRELESS COMMUNICATIONS(FWC)
Indian Institute of Technology Hyderabad
Email: velicharla@outlook.com

Date:02-04-2023

Hls ASSIGNMENT-5

Part A:
Implement a DUT that accesses 8 elements from BRAM (the BRAM should be contained within the DUT, you can choose to populate the BRAM in any way you like) and gives out all 8 values as HLS streaming output in a single bundle in a single clock cycle. The starting index will be an input to the DUT and it will be a multiple of 8.

The way to access these 8 elements are as follows:

1. 8 consecutive elements are to be chosen and these elements will make up a bundle.
2. Every 8th element is to be chosen and 8 such elements will make up a bundle.

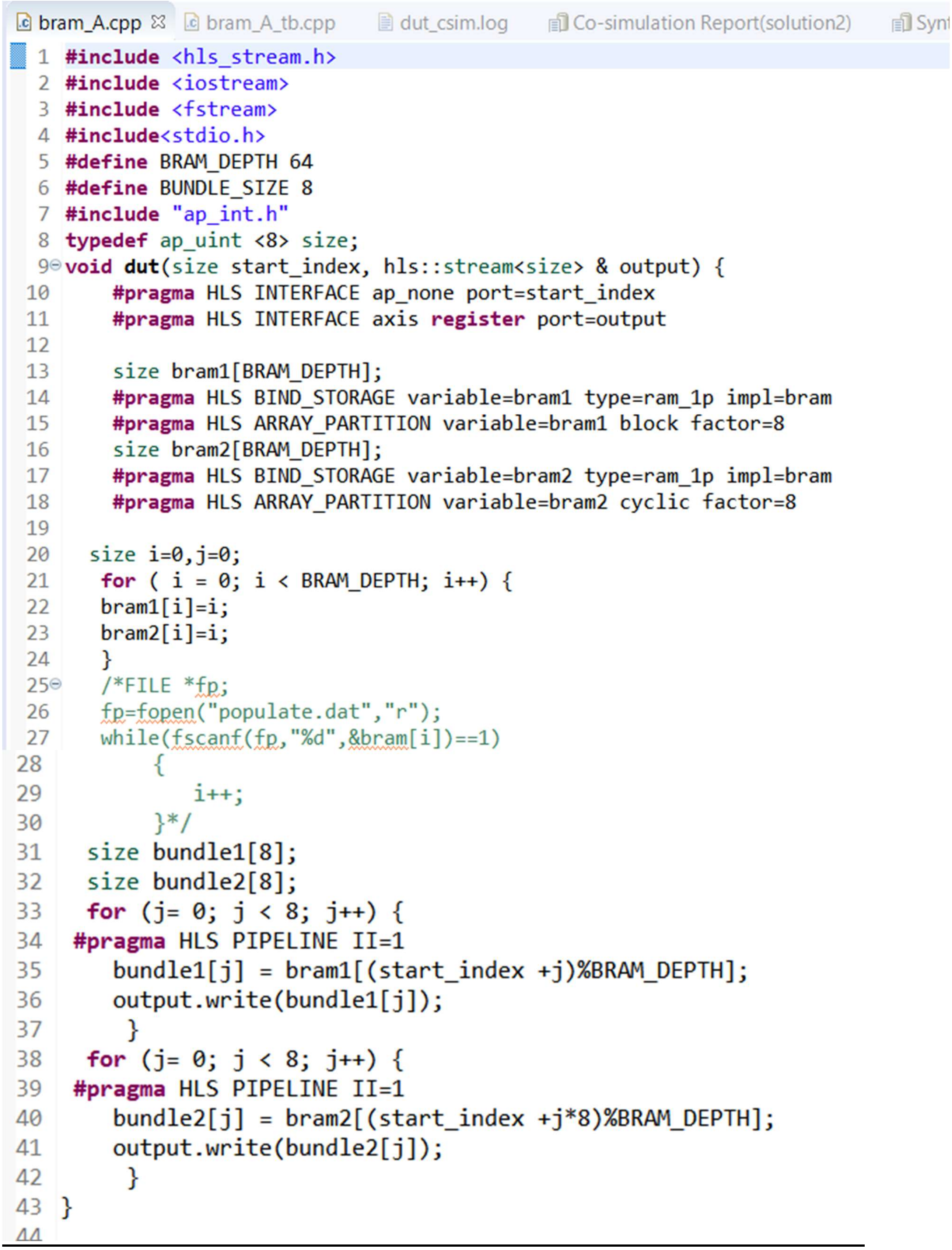
Part B:
Implement a DUT that takes 4 inputs in 4 clock cycles and then saves all the inputs in a BRAM (The BRAM should be contained within the DUT) at a single address. At the same time, it also gives out all the 4 inputs as a single bundle output in the same clock cycle when it receives the final input. The BRAM index will overflow after the BRAM is full and it should overwrite the old values as more inputs keep coming in.

You can assume the data elements to be 8-bit values for both parts.

The purpose of this assignment is to understand the working of the pragmas, ARRAY_PARTITION (block, cyclic, complete) and ARRAY_RESHAPE (horizontal, vertical). Report all the source code files, self-checking testbench files, input and output files, and "ALL" the HLS reports.

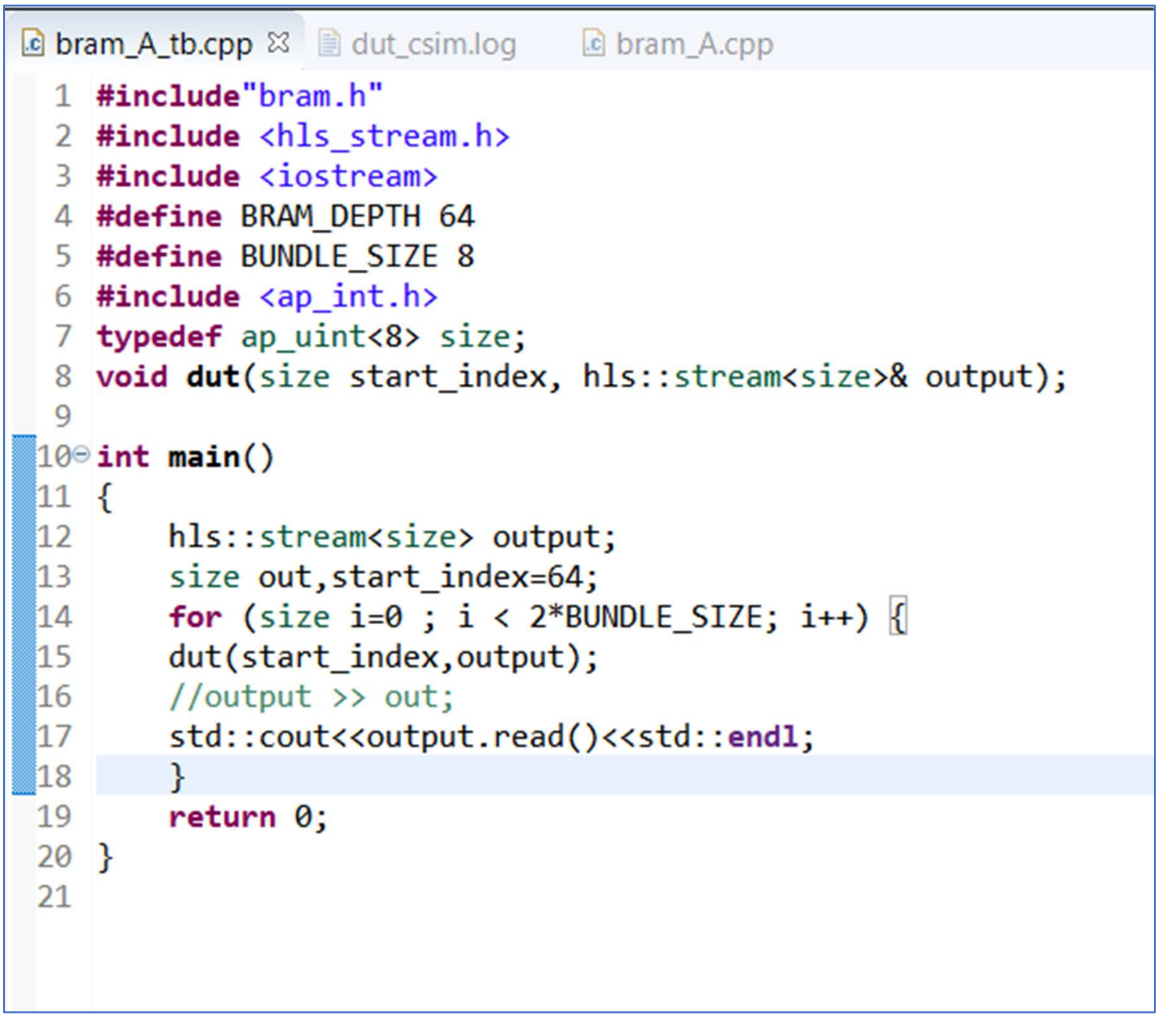
Design:

Main function:



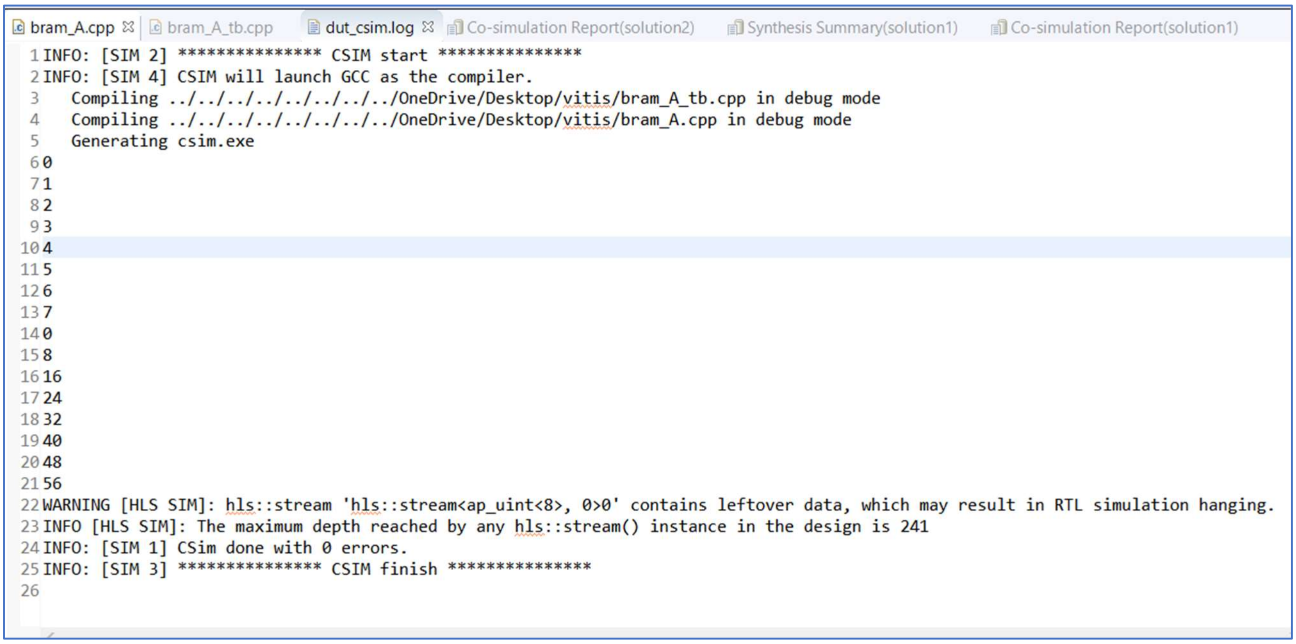
```
#include <hls_stream.h>
#include <iostream>
#include <fstream>
#include<stdio.h>
#define BRAM_DEPTH 64
#define BUNDLE_SIZE 8
#include "ap_int.h"
typedef ap_uint <8> size;
void dut(size start_index, hls::stream<size> & output) {
    #pragma HLS INTERFACE ap_none port=start_index
    #pragma HLS INTERFACE axis register port=output

    size bram1[BRAM_DEPTH];
    #pragma HLS BIND_STORAGE variable=bram1 type=ram_1p impl=bram
    #pragma HLS ARRAY_PARTITION variable=bram1 block factor=8
    size bram2[BRAM_DEPTH];
    #pragma HLS BIND_STORAGE variable=bram2 type=ram_1p impl=bram
    #pragma HLS ARRAY_PARTITION variable=bram2 cyclic factor=8

  size i=0,j=0;
   for ( i = 0; i < BRAM_DEPTH; i++) {
   bram1[i]=i;
   bram2[i]=i;
   }
   /*FILE *fp;
   fp=fopen("populate.dat","r");
   while(fscanf(fp,"%d",&bram[i])==1)
        {
            i++;
        }*/
  size bundle1[8];
  size bundle2[8];
  for (j= 0; j < 8; j++) {
 #pragma HLS PIPELINE II=1
    bundle1[j] = bram1[(start_index +j)%BRAM_DEPTH];
    output.write(bundle1[j]);
     }
  for (j= 0; j < 8; j++) {
 #pragma HLS PIPELINE II=1
    bundle2[j] = bram2[(start_index +j*8)%BRAM_DEPTH];
    output.write(bundle2[j]);
     }
}
```

Test Bench:



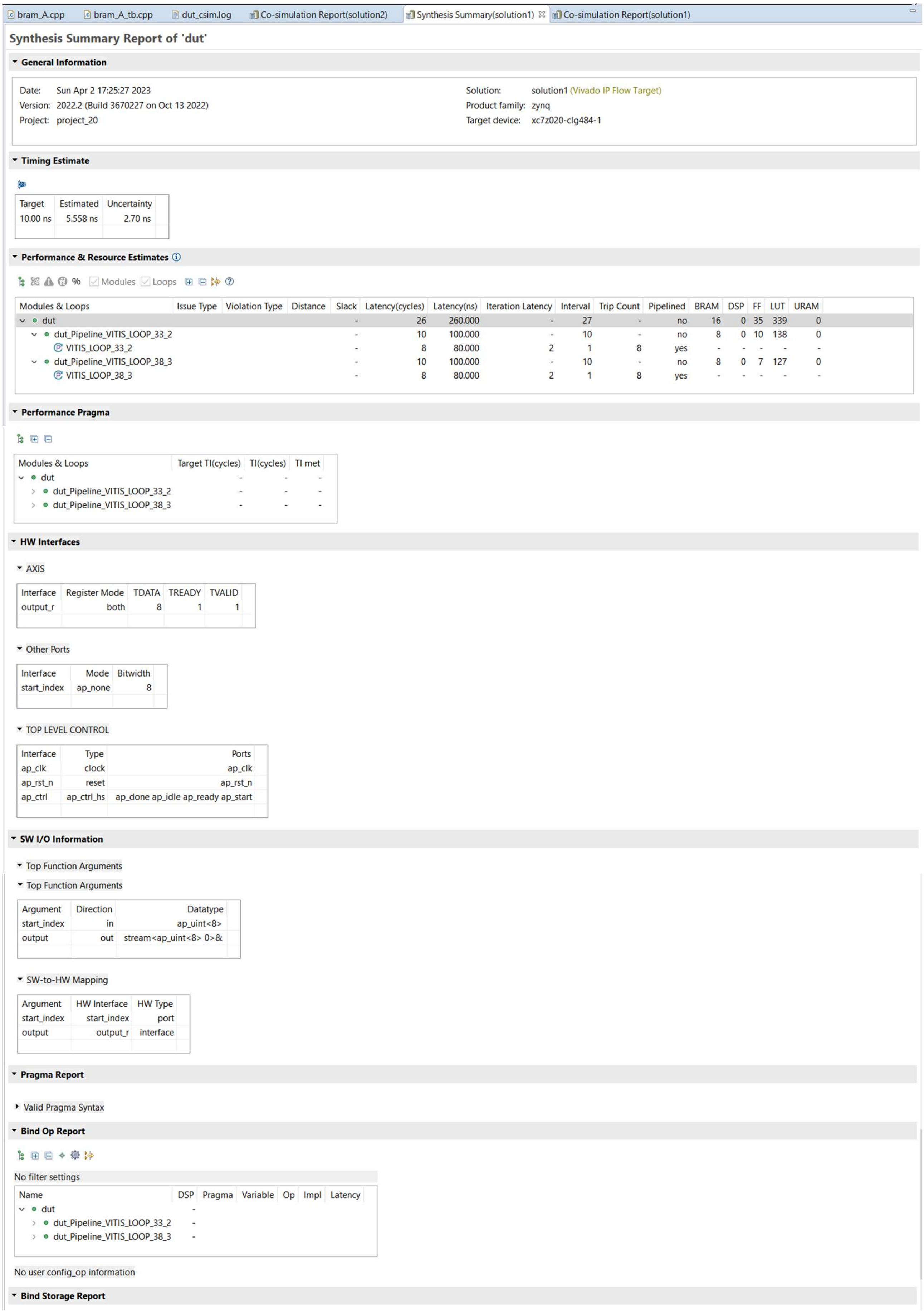
```cpp
#include"bram.h"
#include <hls_stream.h>
#include <iostream>
#define BRAM_DEPTH 64
#define BUNDLE_SIZE 8
#include <ap_int.h>
typedef ap_uint<8> size;
void dut(size start_index, hls::stream<size>& output);

int main()
{
    hls::stream<size> output;
    size out,start_index=64;
    for (size i=0 ; i < 2*BUNDLE_SIZE; i++) {
    dut(start_index,output);
    //output >> out;
    std::cout<<output.read()<<std::endl;
    }
    return 0;
}
```

C simulation printed output:


bram_A.cpp
bram_A_tb.cpp
dut_csim.log
Co-simulation Report(solution2)
Synthesis Summary(solution1)
Co-simulation Report(solution1)

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/bram_A_tb.cpp in debug mode
   Compiling ../../../../../../../../OneDrive/Desktop/vitis/bram_A.cpp in debug mode
   Generating csim.exe
0
1
2
3
4
5
6
7
0
8
16
24
32
40
48
56
WARNING [HLS SIM]: hls::stream 'hls::stream<ap_uint<8>, 0>0' contains leftover data, which may result in RTL simulation hanging.
INFO [HLS SIM]: The maximum depth reached by any hls::stream() instance in the design is 241
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis Report:



bram_A.cpp | bram_A_tb.cpp | dut_csim.log | Co-simulation Report(solution2) | Synthesis Summary(solution1) | Co-simulation Report(solution1)

Synthesis Summary Report of 'dut'

General Information

Date:	Sun Apr 2 17:25:27 2023	Solution:	solution1 (Vivado IP Flow Target)
Version:	2022.2 (Build 3670227 on Oct 13 2022)	Product family:	zynq
Project:	project_20	Target device:	xc7z020-clg484-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	5.558 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
dut				-	26	260.000	-	27	-	no	16	0	35	339	0
dut_Pipeline_VITIS_LOOP_33_2				-	10	100.000	-	10	-	no	8	0	10	138	0
VITIS_LOOP_33_2				-	8	80.000	2	1	8	yes	-	-	-	-	-
dut_Pipeline_VITIS_LOOP_38_3				-	10	100.000	-	10	-	no	8	0	7	127	0
VITIS_LOOP_38_3				-	8	80.000	2	1	8	yes	-	-	-	-	-

Performance Pragma

Modules & Loops	Target TI(cycles)	TI(cycles)	TI met
dut	-	-	-
dut_Pipeline_VITIS_LOOP_33_2	-	-	-
dut_Pipeline_VITIS_LOOP_38_3	-	-	-

HW Interfaces

AXIS

Interface	Register Mode	TDATA	TREADY	TVALID
output_r	both	8	1	1

Other Ports

Interface	Mode	Bitwidth
start_index	ap_none	8

TOP LEVEL CONTROL

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

SW I/O Information

Top Function Arguments

Top Function Arguments

Argument	Direction	Datatype
start_index	in	ap_uint<8>
output	out	stream<ap_uint<8> 0>&

SW-to-HW Mapping

Argument	HW Interface	HW Type
start_index	start_index	port
output	output_r	interface

Pragma Report

Valid Pragma Syntax

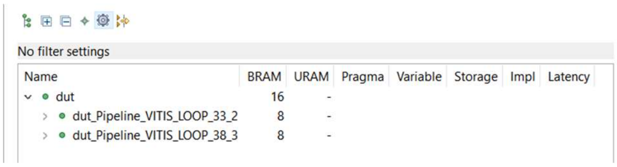
Bind Op Report

No filter settings

Name	DSP	Pragma	Variable	Op	Impl	Latency
dut	-					
dut_Pipeline_VITIS_LOOP_33_2	-					
dut_Pipeline_VITIS_LOOP_38_3	-					

No user config_op information

Bind Storage Report

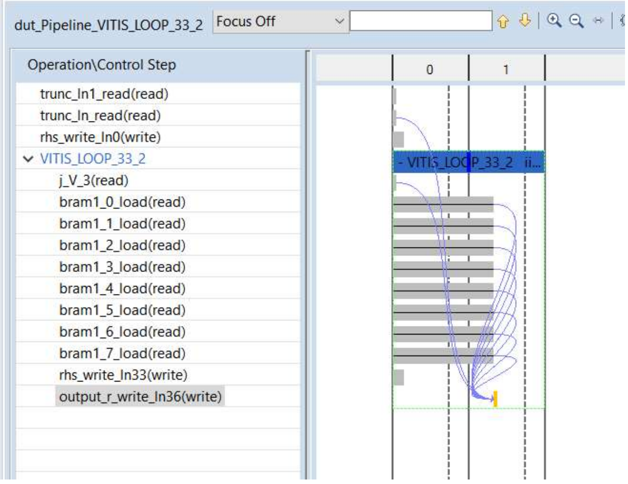


No filter settings

Name	BRAM	URAM	Pragma	Variable	Storage	Impl	Latency
dut	16	-					
dut_Pipeline_VITIS_LOOP_33_2	8	-					
dut_Pipeline_VITIS_LOOP_38_3	8	-					

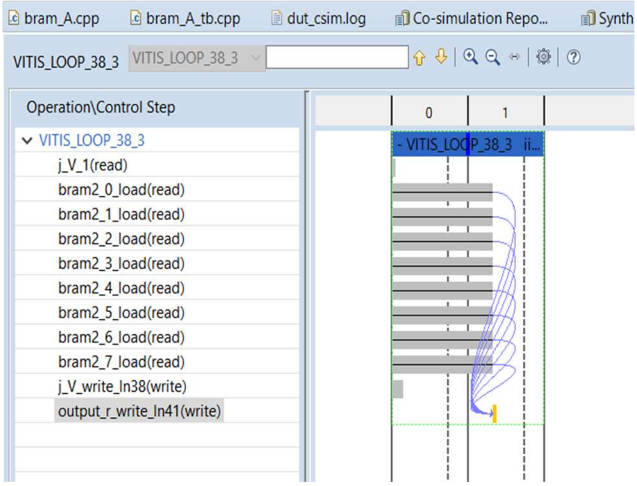
Schedule Viewer comparison:

Consecutive 8 Elements - Block Partitioning,
Every 8th Elements - Cyclic Partitioning



Operation\Control Step

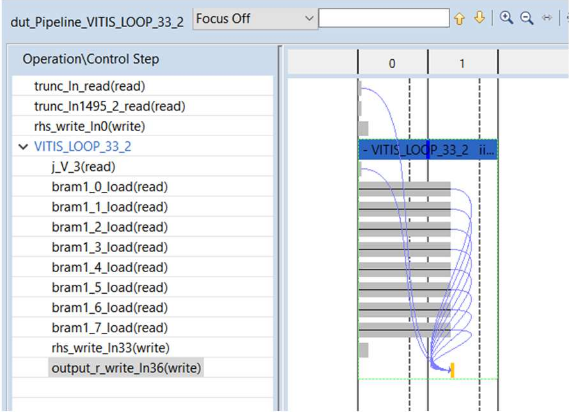
trunc_ln1_read(read)
trunc_ln_read(read)
rhs_write_ln0(write)
VITIS_LOOP_33_2
j_V_3(read)
bram1_0_load(read)
bram1_1_load(read)
bram1_2_load(read)
bram1_3_load(read)
bram1_4_load(read)
bram1_5_load(read)
bram1_6_load(read)
bram1_7_load(read)
rhs_write_ln33(write)
output_r_write_ln36(write)



Operation\Control Step

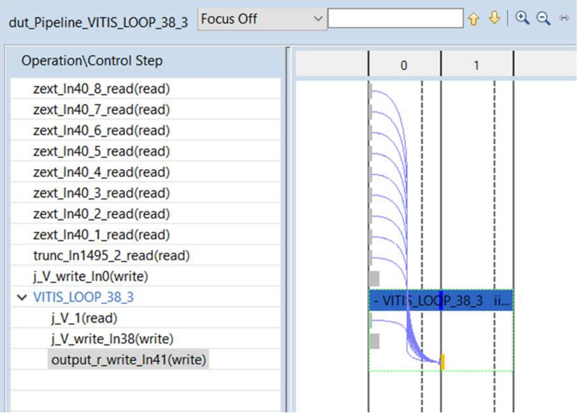
VITIS_LOOP_38_3
j_V_1(read)
bram2_0_load(read)
bram2_1_load(read)
bram2_2_load(read)
bram2_3_load(read)
bram2_4_load(read)
bram2_5_load(read)
bram2_6_load(read)
bram2_7_load(read)
j_V_write_ln38(write)
output_r_write_ln41(write)

Consecutive 8 Elements - Cyclic Partitioning,
Every 8th Elements - Block Partitioning



Operation\Control Step

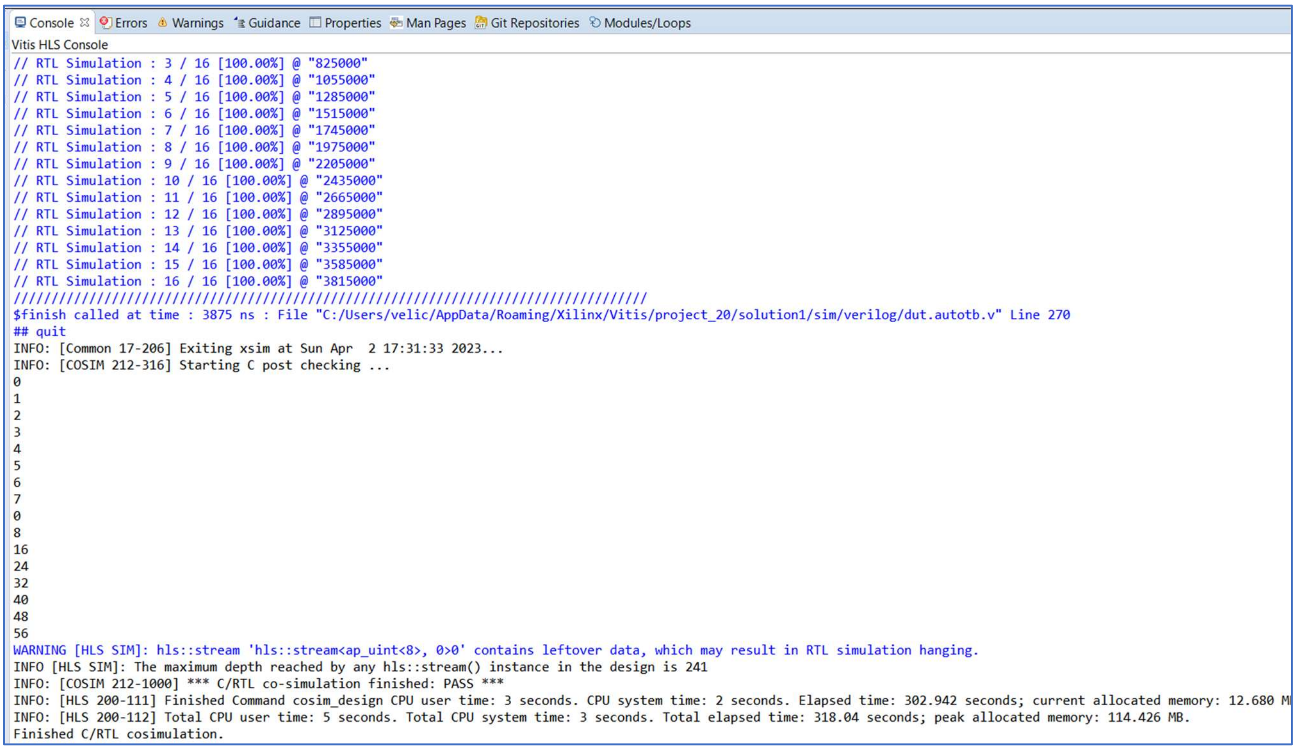
trunc_ln_read(read)
trunc_ln1495_2_read(read)
rhs_write_ln0(write)
VITIS_LOOP_33_2
j_V_3(read)
bram1_0_load(read)
bram1_1_load(read)
bram1_2_load(read)
bram1_3_load(read)
bram1_4_load(read)
bram1_5_load(read)
bram1_6_load(read)
bram1_7_load(read)
rhs_write_ln33(write)
output_r_write_ln36(write)



Operation\Control Step

zext_ln40_8_read(read)
zext_ln40_7_read(read)
zext_ln40_6_read(read)
zext_ln40_5_read(read)
zext_ln40_4_read(read)
zext_ln40_3_read(read)
zext_ln40_2_read(read)
zext_ln40_1_read(read)
trunc_ln1495_2_read(read)
j_V_write_ln0(write)
VITIS_LOOP_38_3
j_V_1(read)
j_V_write_ln38(write)
output_r_write_ln41(write)

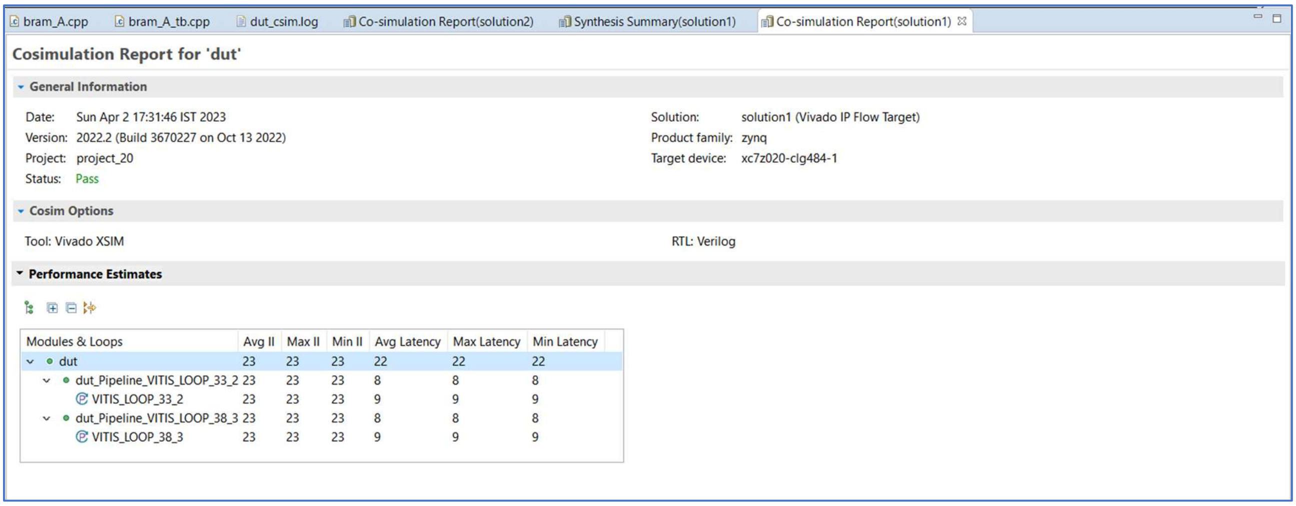
Cosimulation printed output:



Vitis HLS Console

```
// RTL Simulation : 3 / 16 [100.00%] @ "825000"
// RTL Simulation : 4 / 16 [100.00%] @ "1055000"
// RTL Simulation : 5 / 16 [100.00%] @ "1285000"
// RTL Simulation : 6 / 16 [100.00%] @ "1515000"
// RTL Simulation : 7 / 16 [100.00%] @ "1745000"
// RTL Simulation : 8 / 16 [100.00%] @ "1975000"
// RTL Simulation : 9 / 16 [100.00%] @ "2205000"
// RTL Simulation : 10 / 16 [100.00%] @ "2435000"
// RTL Simulation : 11 / 16 [100.00%] @ "2665000"
// RTL Simulation : 12 / 16 [100.00%] @ "2895000"
// RTL Simulation : 13 / 16 [100.00%] @ "3125000"
// RTL Simulation : 14 / 16 [100.00%] @ "3355000"
// RTL Simulation : 15 / 16 [100.00%] @ "3585000"
// RTL Simulation : 16 / 16 [100.00%] @ "3815000"
////////////////////////////////////////////////////////////////////////////////////
$finish called at time : 3875 ns : File "C:/Users/velic/AppData/Roaming/Xilinx/Vitis/project_20/solution1/sim/verilog/dut.autotb.v" Line 270
## quit
INFO: [Common 17-206] Exiting xsim at Sun Apr  2 17:31:33 2023...
INFO: [COSIM 212-316] Starting C post checking ...
0
1
2
3
4
5
6
7
0
8
16
24
32
40
48
56
WARNING [HLS SIM]: hls::stream 'hls::stream<ap_uint<8>, 0>0' contains leftover data, which may result in RTL simulation hanging.
INFO [HLS SIM]: The maximum depth reached by any hls::stream() instance in the design is 241
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
INFO: [HLS 200-111] Finished Command cosim_design CPU user time: 3 seconds. CPU system time: 2 seconds. Elapsed time: 302.942 seconds; current allocated memory: 12.680 M
INFO: [HLS 200-112] Total CPU user time: 5 seconds. Total CPU system time: 3 seconds. Total elapsed time: 318.04 seconds; peak allocated memory: 114.426 MB.
Finished C/RTL cosimulation.
```

Cosimulation Report:


bram_A.cpp
bram_A_tb.cpp
dut_csim.log
Co-simulation Report(solution2)
Synthesis Summary(solution1)
Co-simulation Report(solution1)

Cosimulation Report for 'dut'

General Information

Date: Sun Apr 2 17:31:46 IST 2023
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: project_20
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
dut	23	23	23	22	22	22
dut_Pipeline_VITIS_LOOP_33_2	23	23	23	8	8	8
VITIS_LOOP_33_2	23	23	23	9	9	9
dut_Pipeline_VITIS_LOOP_38_3	23	23	23	8	8	8
VITIS_LOOP_38_3	23	23	23	9	9	9

Github:

https://github.com/velicharlagokulkumar/vitis-hls/tree/main/project_20