***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);  } |

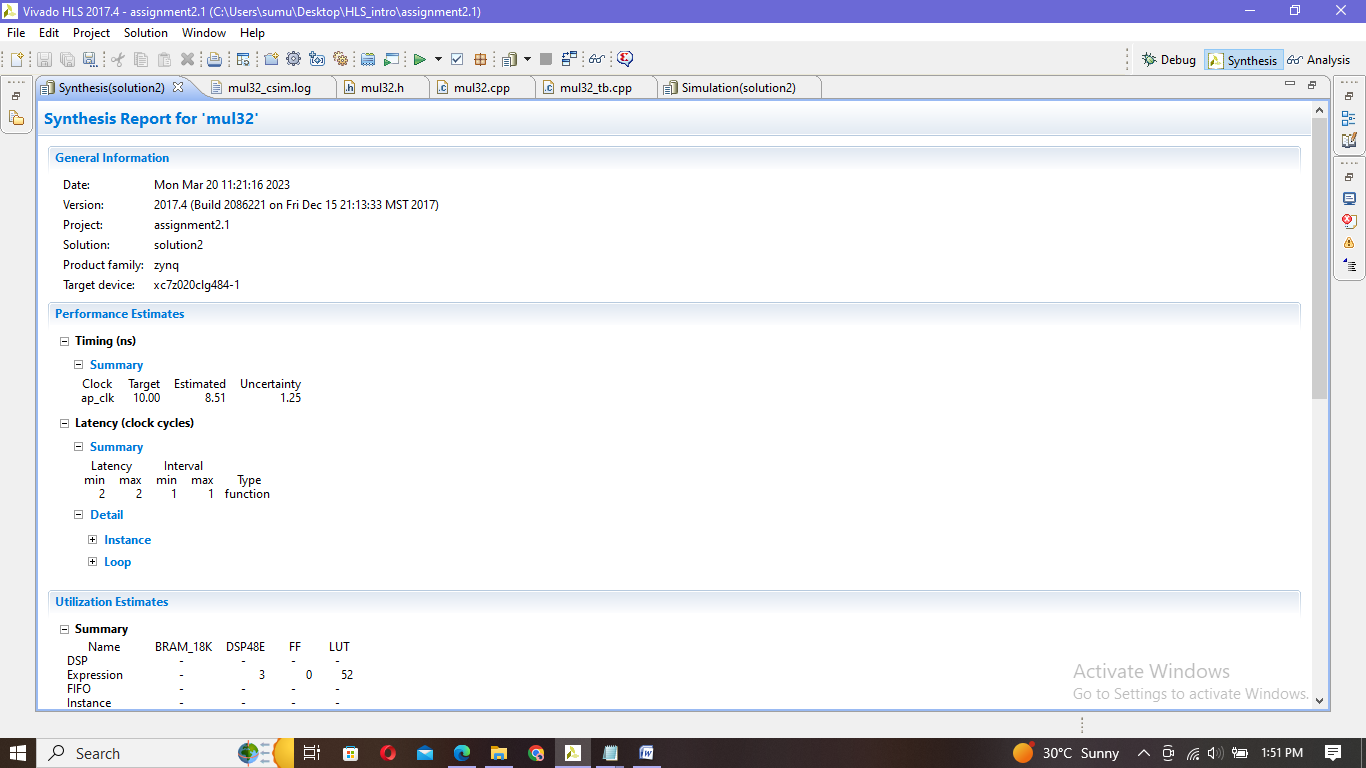
Test bench

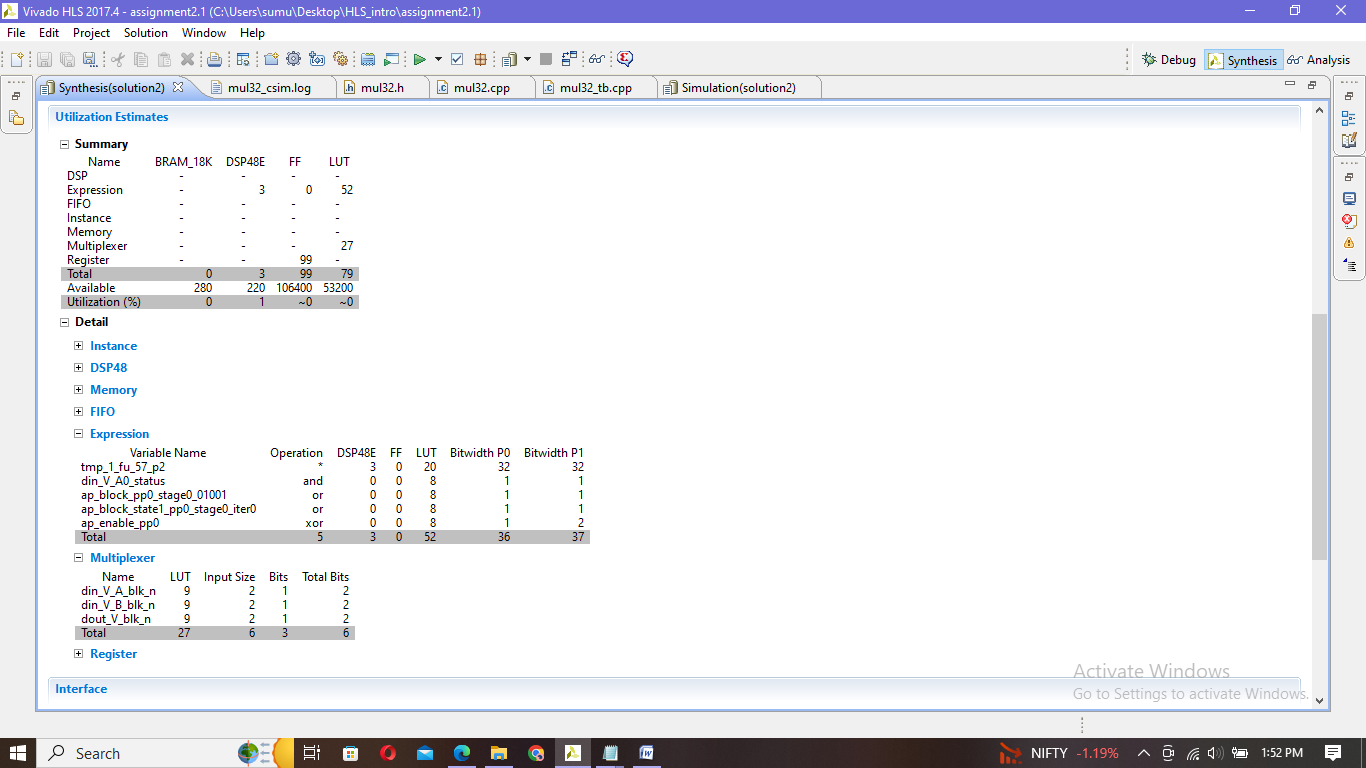
|  |
| --- |
| **#include** "mul32.h"  **void** **mul32**(stream<inputs> &din,stream<**long** **long**> &dout);  **int** **main**(){  stream<inputs> indata;  stream<**long** **long**> outdata;  inputs in;  **long** **long** out;  **int** i;  **for** (i=0;i<10;i++){  in.A=i+2;  in.B=i;  indata.write(in);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

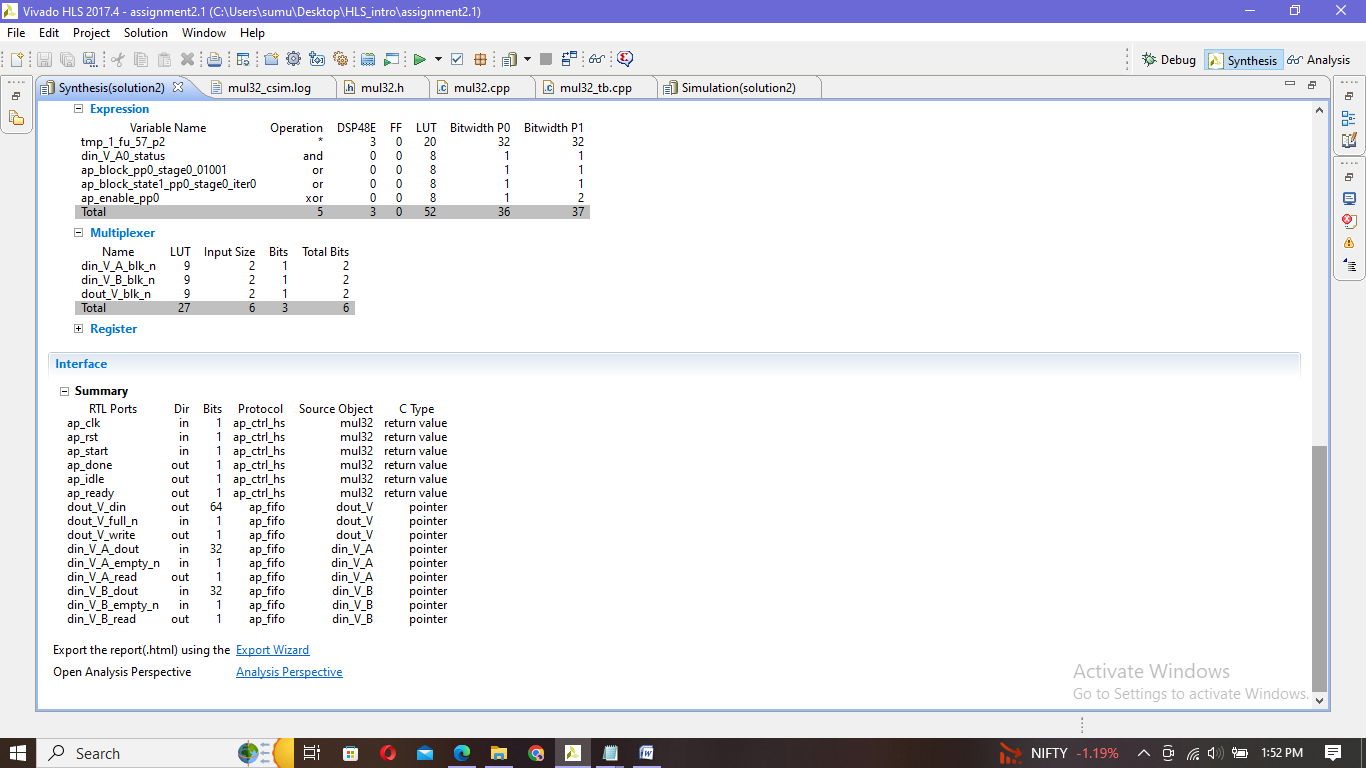
Simulation report

|  |
| --- |
| 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:

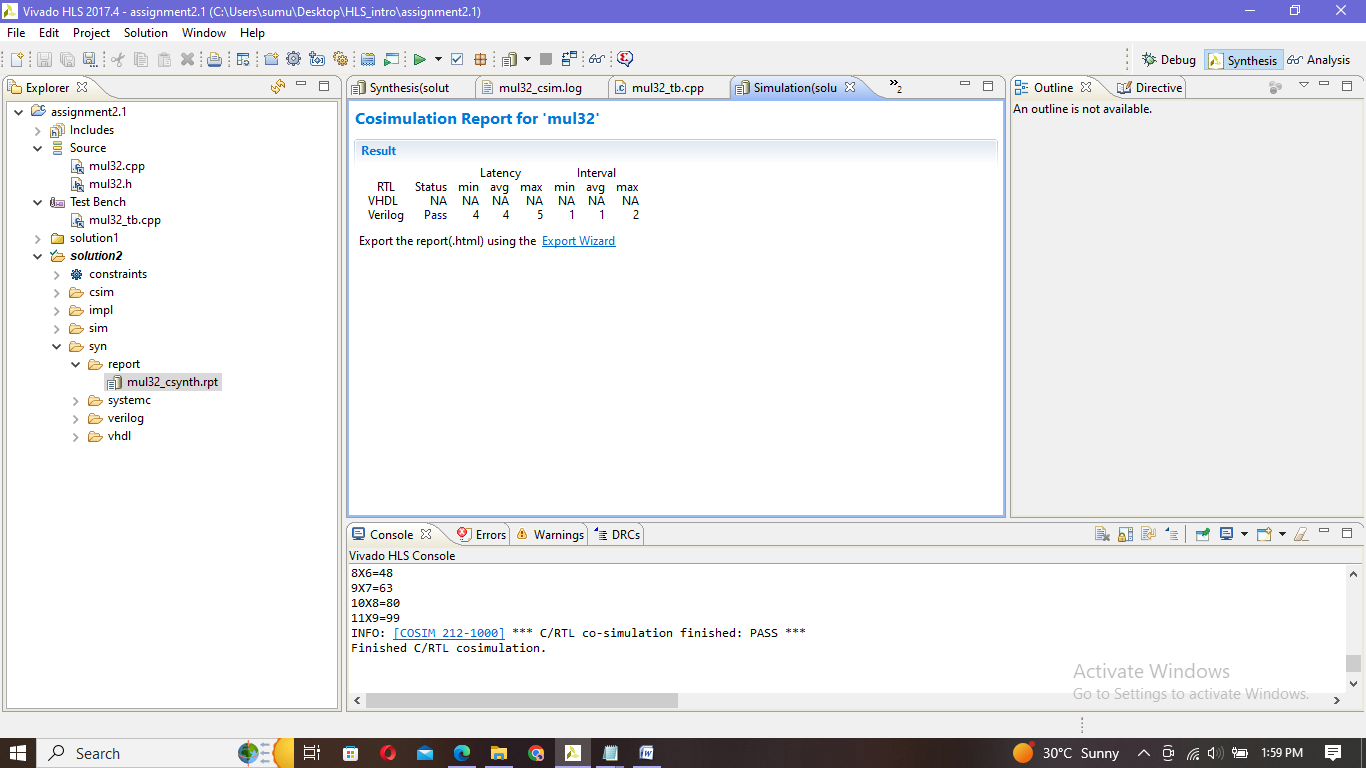






Co-simulation

|  |
| --- |
| 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. |



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);  } |

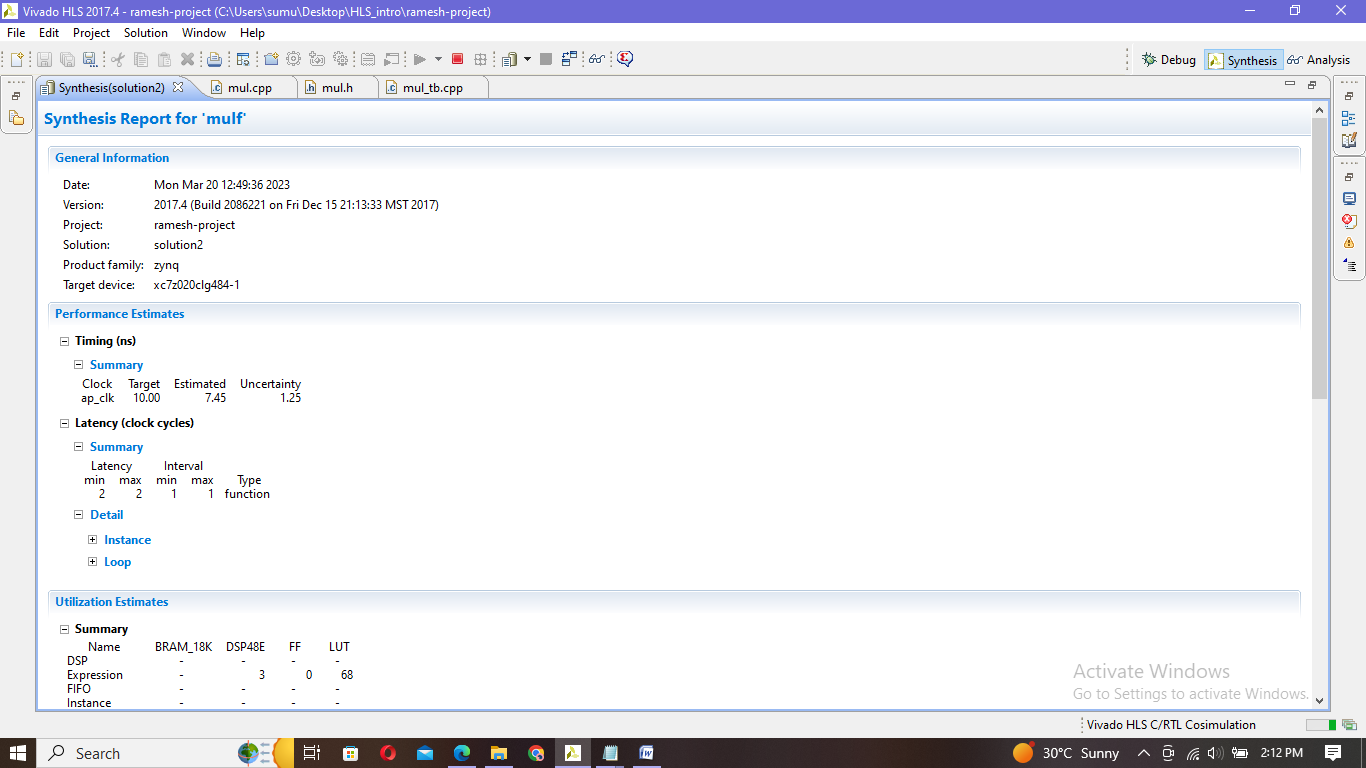
Test bench

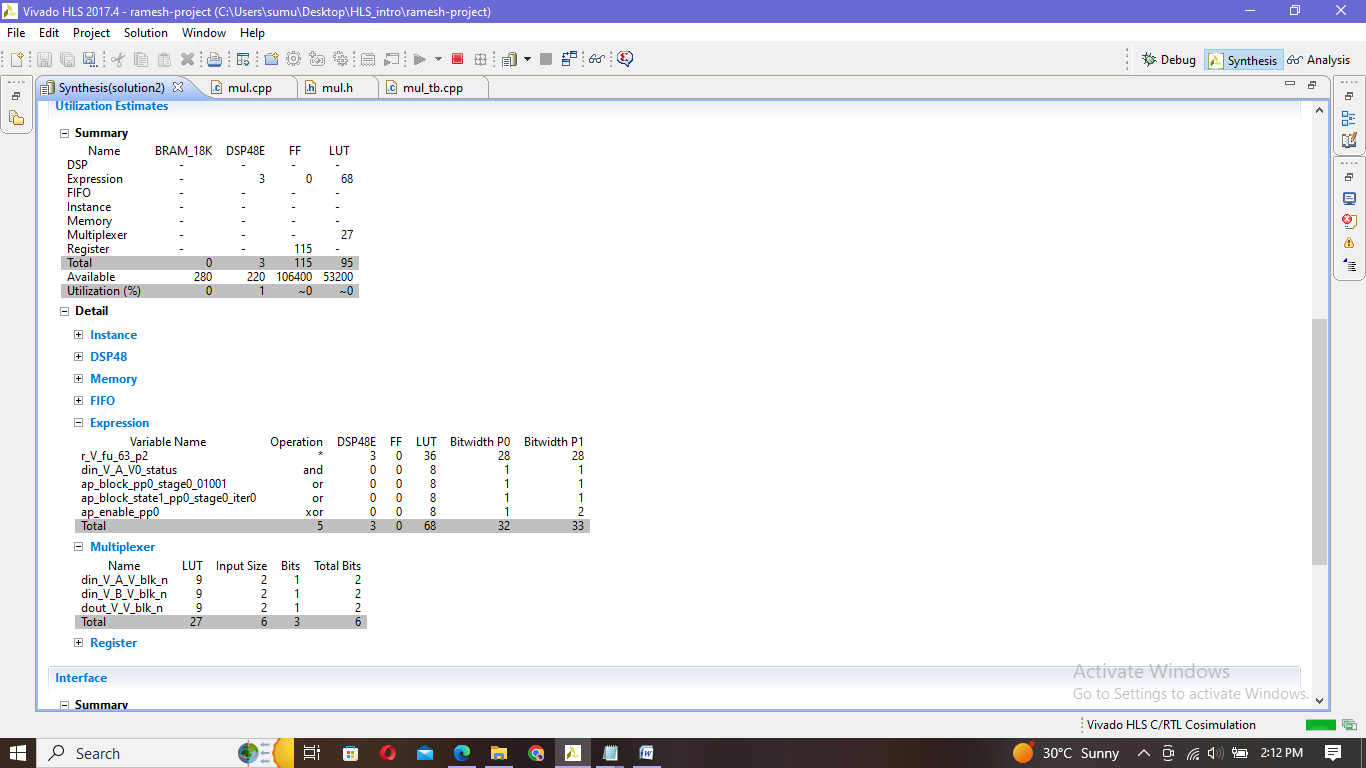
|  |
| --- |
| **#include** "mul.h"  **void** **mulf**(stream<inputs> &din,stream<fix56\_8> &dout);  **int** **main**(){  stream<inputs> indata;  stream<fix56\_8> outdata;  **int** i;  inputs in={0,0};  fix56\_8 out;  **for** (i=0;i<10;i++){  in.A=i+0.6;  in.B=i+0.5;  indata.write(in);  mulf(indata,outdata);  outdata>>out;  cout <<in.A<<"X"<<in.B<<"="<<out<< **endl**;}} |

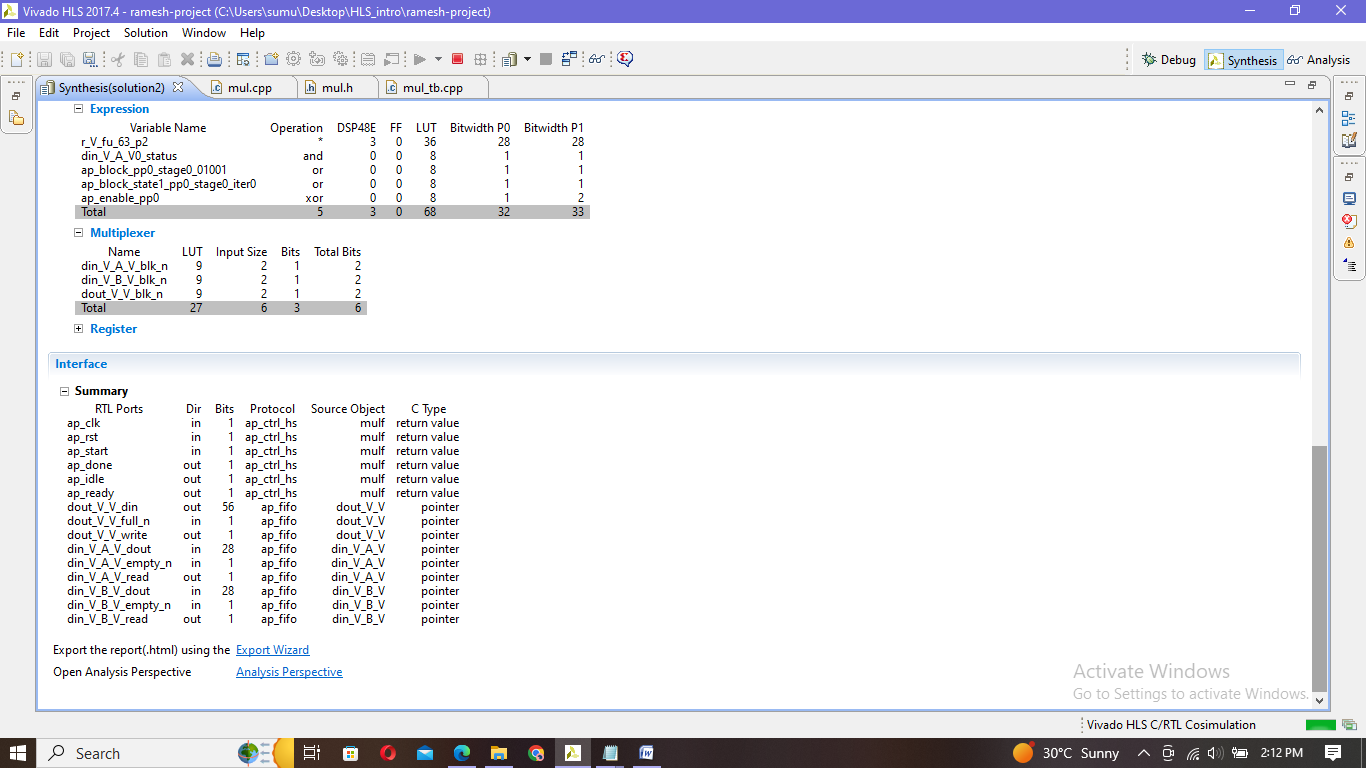
Simulation:

|  |
| --- |
| 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

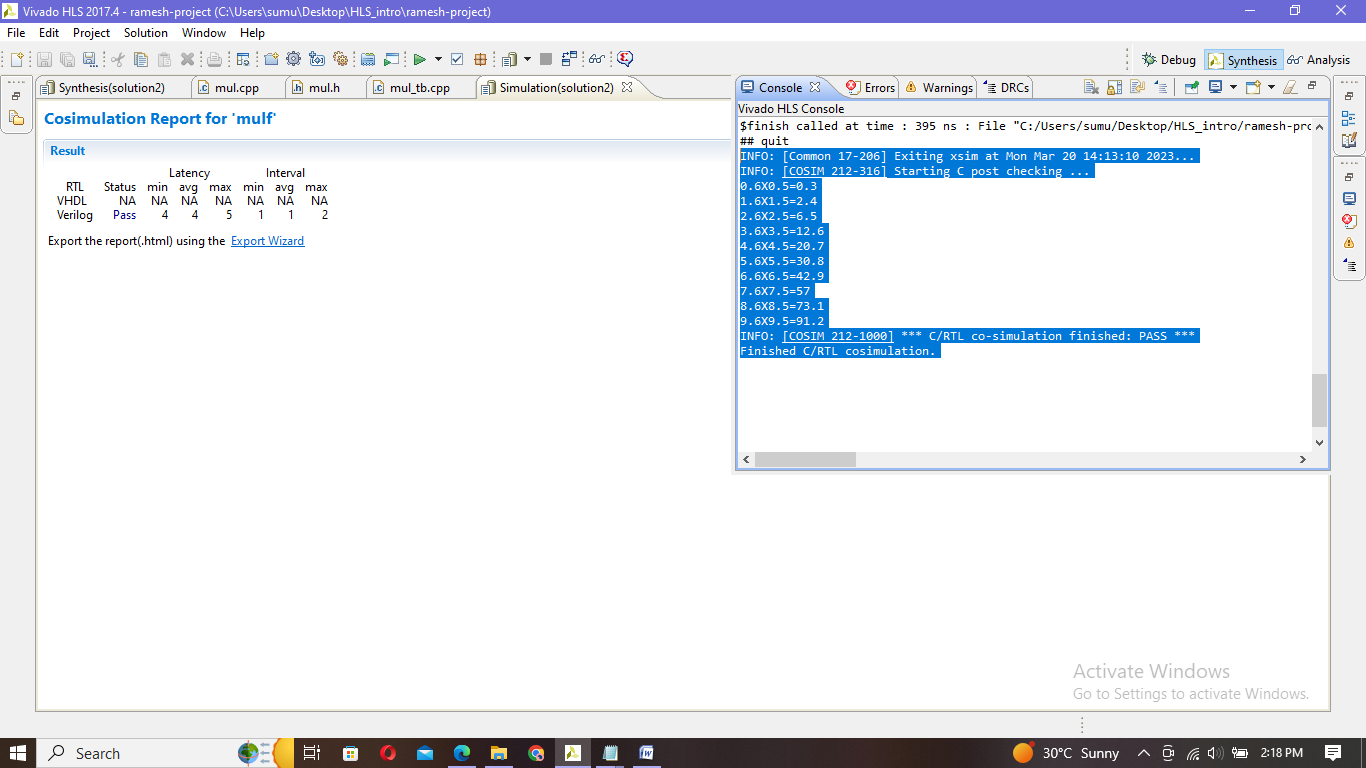






Co-simulation:

|  |
| --- |
| 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. |

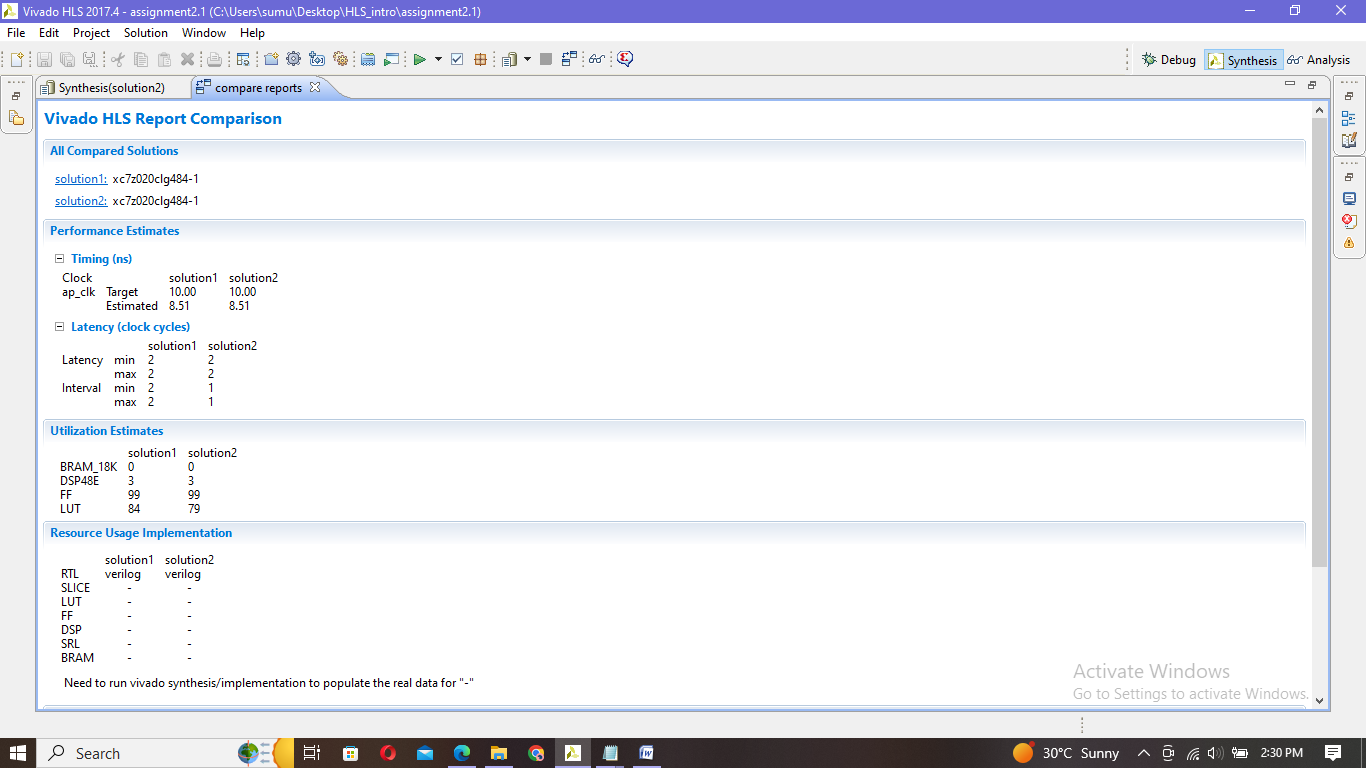


OBSERVATIONS:

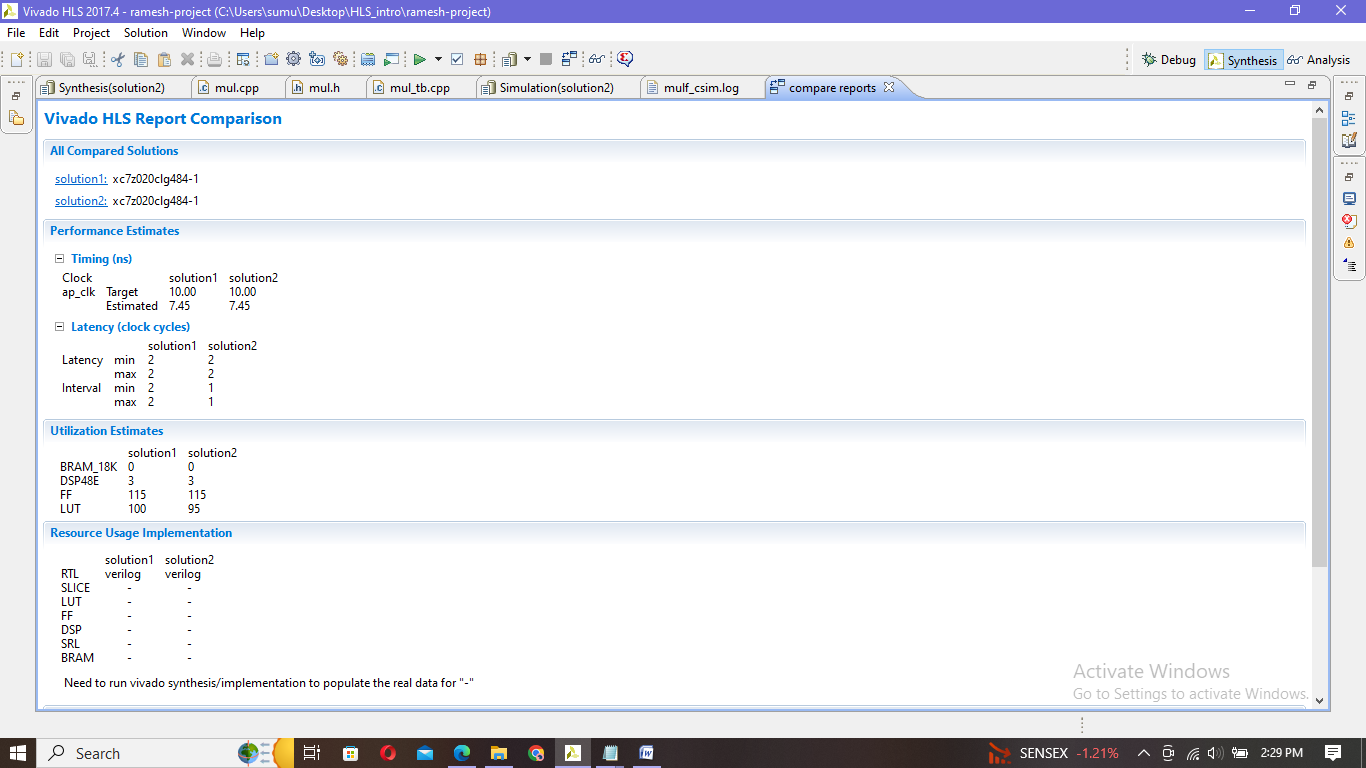
Solution1:without pipelining function

Solution2:with pipelining function

(3.1)



(3.2)



* 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 |