HLS ASSIGNMENT5(kanekal kousar)

PART A:

(Q) 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.

1) 8 consecutive elements are to be chosen and these elements will make up a bundle.

Header file

|  |
| --- |
| **#include**<hls\_stream.h>  **using** **namespace** hls;  **#include**<ap\_int.h>  **typedef** ap\_uint<8> bit\_width;  **struct** bundle{  bit\_width data[8];  }; |

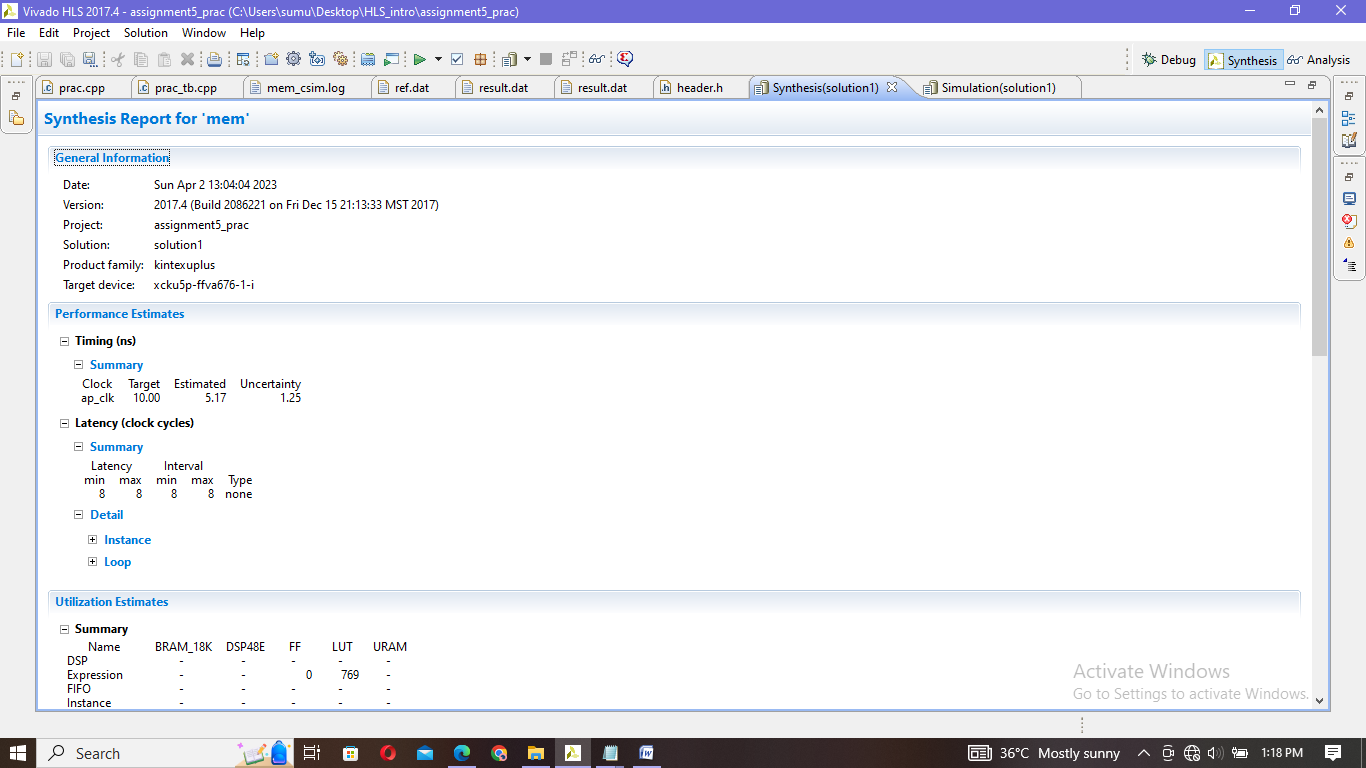
Design code:

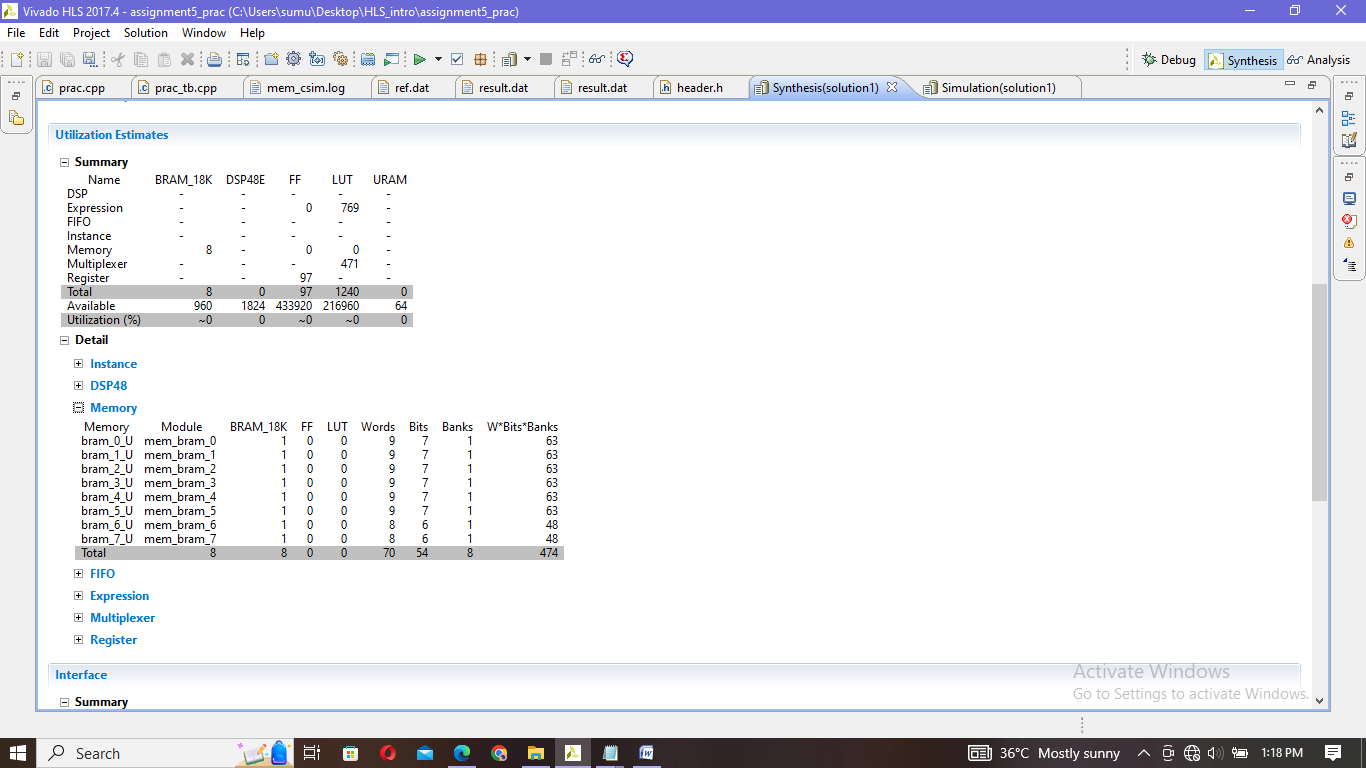
|  |
| --- |
| **#include** "header.h"  **void** **mem**(**int** index,stream<bundle> &output){  bit\_width bram[80]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,60,61,62,63,64,65,66,67,68,69};  **#pragma** HLS RESOURCE variable=bram core=RAM\_1P\_BRAM  **#pragma** HLS ARRAY\_PARTITION variable=bram cyclic factor=8 dim=1  //accessing 8 consecutive elements from bram  bundle out={bram[index],bram[index+1],bram[index+2],bram[index+3],bram[index+4],bram[index+5],bram[index+6],bram[index+7]};  output.write(out);  } |

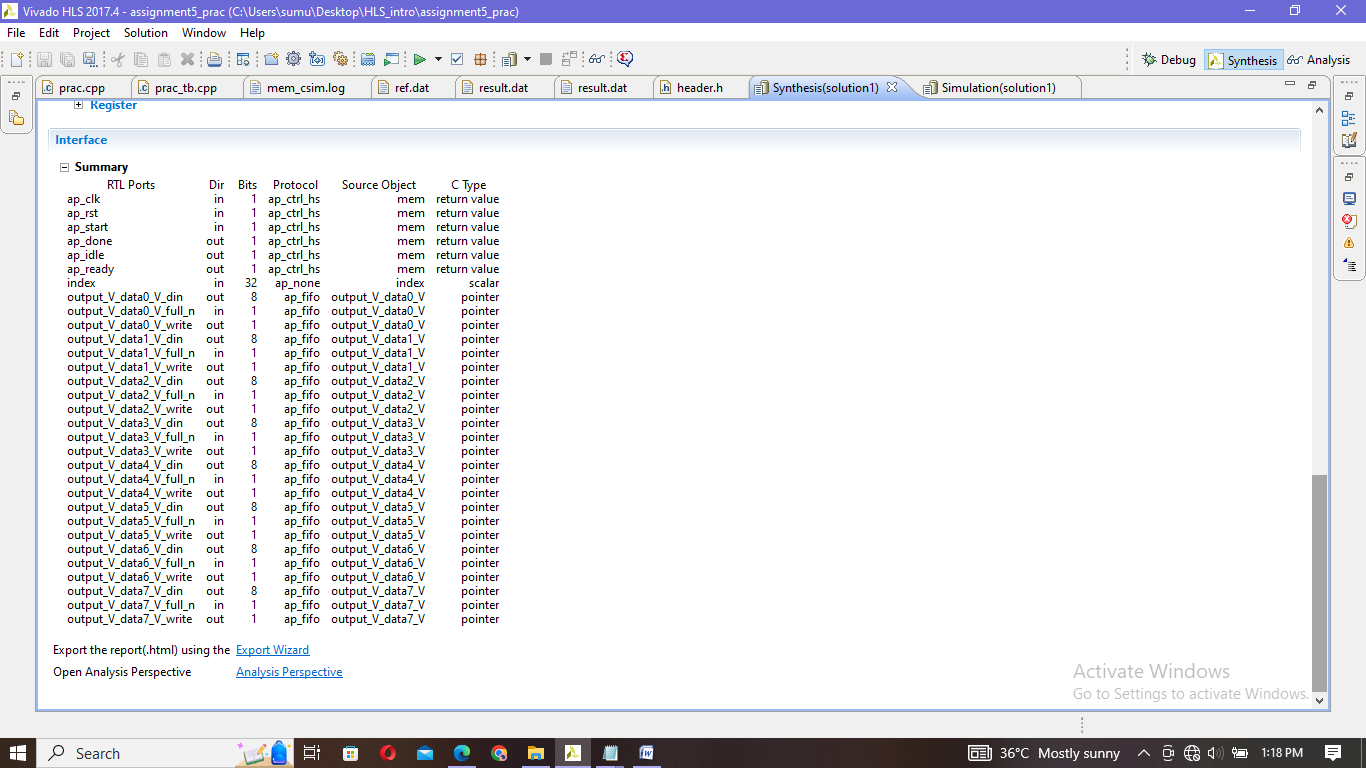
Test bench:

|  |
| --- |
| **#include** "header.h"  **using** **namespace** std;  **#include** <fstream>  **void** **mem**(**int** index,stream<bundle> &output);  **int** **main**(){  stream<bundle> outdata;  **int** index,a,b,c,d,e,f,g,h;  ifstream in;  fstream out\_d;  in.open("ref.dat");  out\_d.open("result.dat");  **int** fail=0;  **while** (in>>index>>a>>b>>c>>d>>e>>f>>g>>h){  mem(index,outdata);  bundle out=outdata.read();  **for** (**int** i=0;i<8;i++){  out\_d<<out.data[i]<<"\t";  }  **if** (out.data[0]==a & out.data[1]==b & out.data[2]==c & out.data[3]==d &out.data[4]==e & out.data[5]==f & out.data[6]==g & out.data[7]==h ){  out\_d<<"passed"<<**endl**;  }  **else**{  out\_d<<"fails"<<**endl**;  fail++;  }  }  in.close();  out\_d.close();  **if** (fail==0){  cout<<"all test cases passed"<<**endl**;  }  **else**{  cout<<fail<<"\t test case failed"<<**endl**;  }} |

Synthesis report:







Simulation:

|  |
| --- |
| 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 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

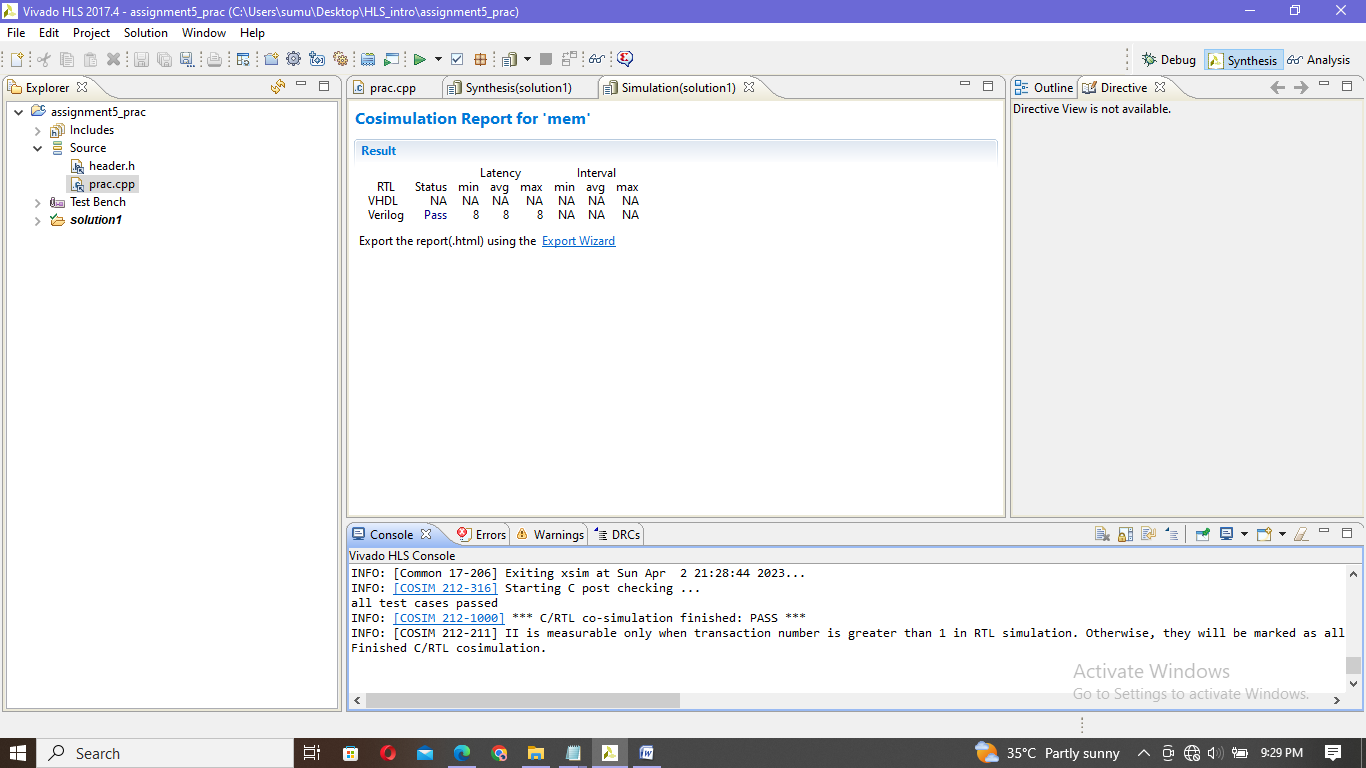
Ref file Result file

|  |
| --- |
| 8 8 9 10 11 12 13 14 15  16 16 17 18 19 20 21 22 |

|  |
| --- |
| 8 9 10 11 12 13 14 15 passed  16 17 18 19 20 21 22 23 passed |

Cosimulatio:

|  |
| --- |
| INFO: [Common 17-206] Exiting xsim at Sun Apr 2 15:15:01 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. |



I have not used Complete array\_partition because,after sysnthesis its not partitioning the blocksRAMs into smaller blocks RAMs

2) Every 8th element is to be chosen and 8 such elements will make up a bundle

Header file:

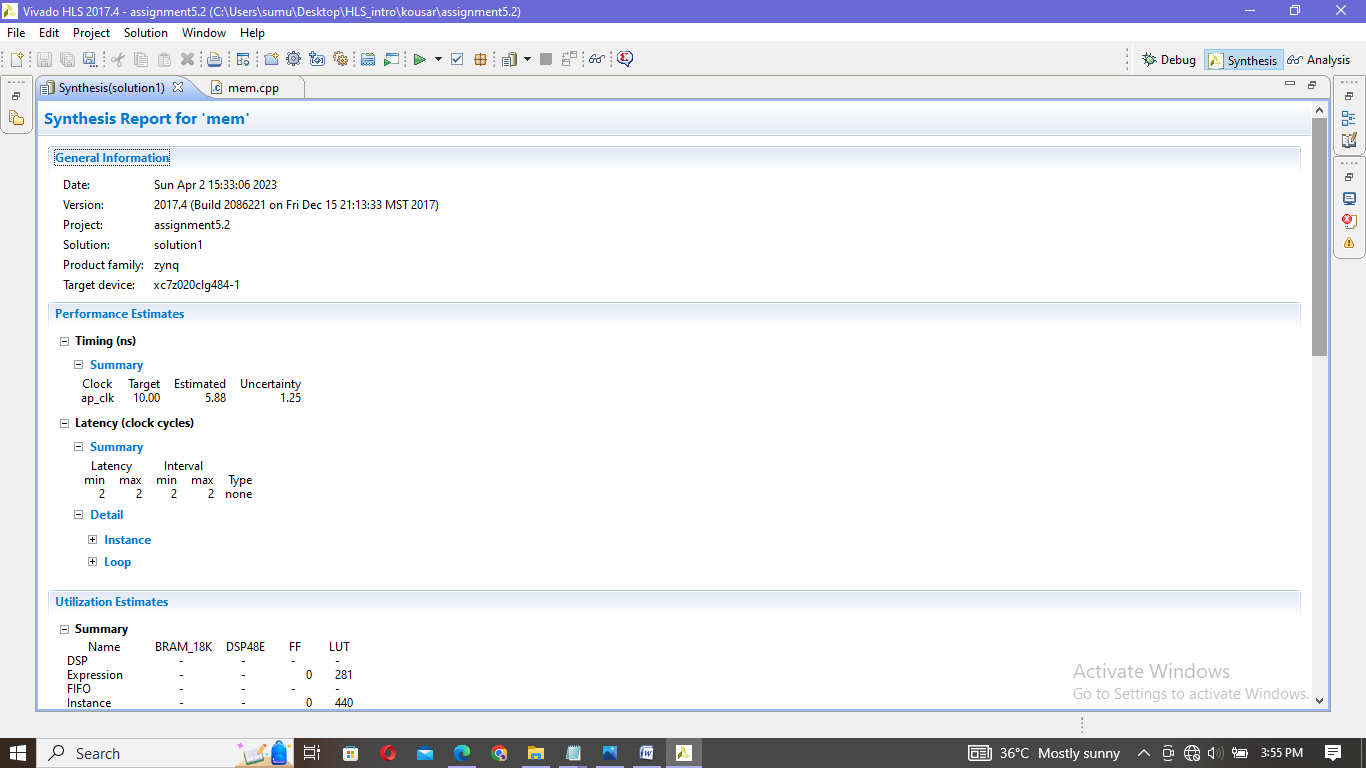
|  |
| --- |
| **#include**<hls\_stream.h>  **using** **namespace** hls;  **#include**<ap\_int.h>  **typedef** ap\_uint<8> bit\_width;  **struct** bundle{  bit\_width data[8];  }; |

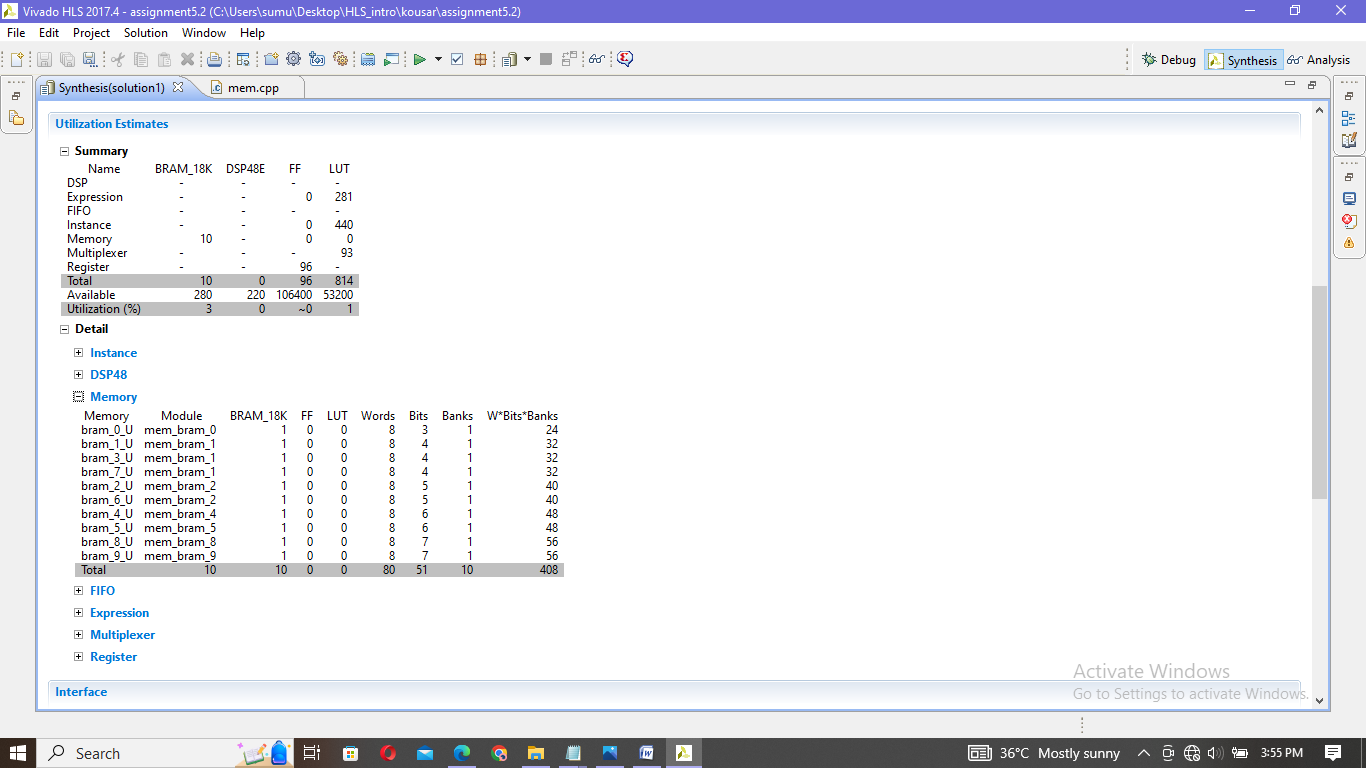
Design code:

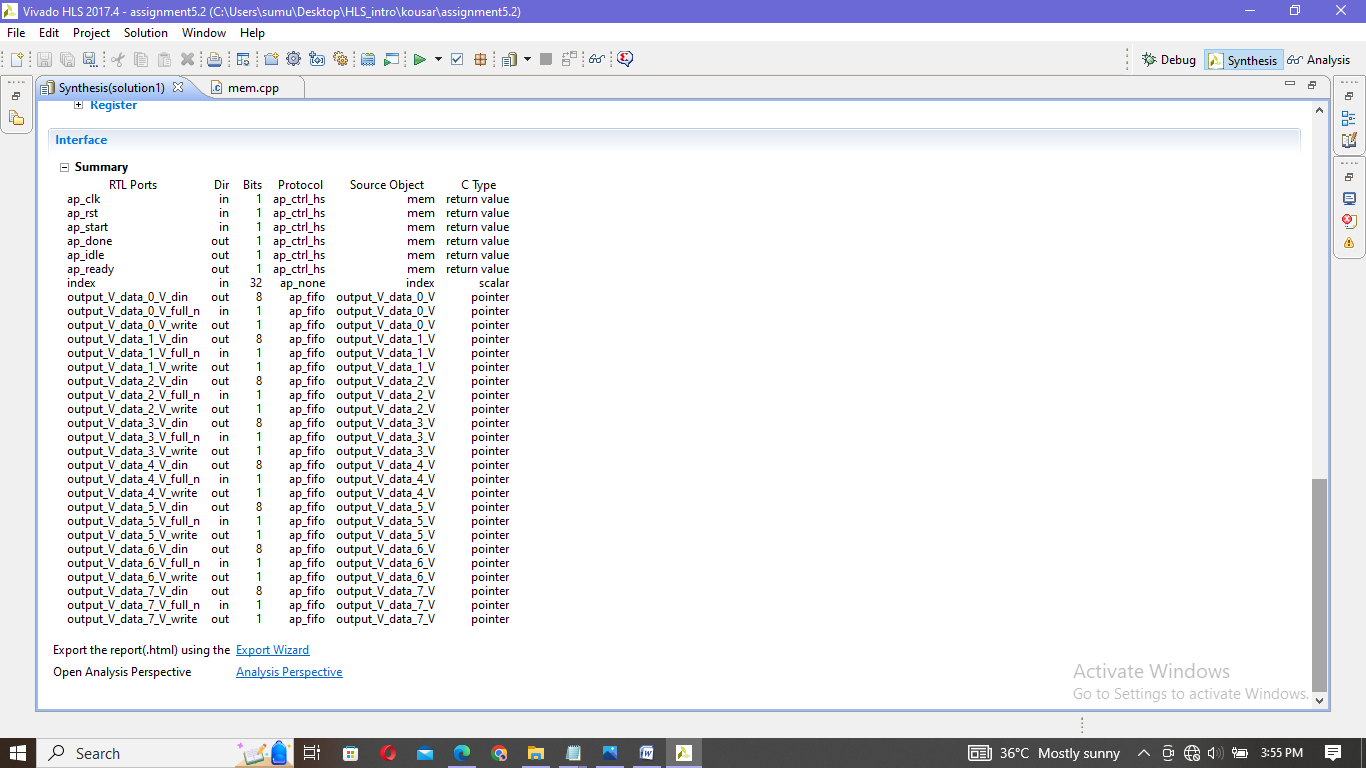
|  |
| --- |
| **#include** "hea.h"  **void** **mem**(**int** index,stream<bundle> &output){  bit\_width bram[80]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,60,61,62,63,64,65,66,67,68,69};  **#pragma** HLS RESOURCE variable=bram core=RAM\_1P\_BRAM  **#pragma** HLS ARRAY\_PARTITION variable=bram block factor=10 dim=1  //accessing 8 consecutive elements from bram  bundle out={bram[index],bram[index+8],bram[index+16],bram[index+24],bram[index+32],bram[index+40],bram[index+48],bram[index+56]};  output.write(out);  } |

Testbench code:

|  |
| --- |
| **#include** "hea.h"  **using** **namespace** std;  **#include** <fstream>  **void** **mem**(**int** index,stream<bundle> &output);  **int** **main**(){  stream<bundle> outdata;  **int** index,a,b,c,d,e,f,g,h;  ifstream in;  fstream out\_d;  in.open("in.dat");  out\_d.open("out.dat");  **int** fail=0;  **while** (in>>index>>a>>b>>c>>d>>e>>f>>g>>h){  mem(index,outdata);  bundle out=outdata.read();  **for** (**int** i=0;i<8;i++){  out\_d<<out.data[i]<<"\t";  }  **if** (out.data[0]==a & out.data[1]==b & out.data[2]==c & out.data[3]==d &out.data[4]==e & out.data[5]==f & out.data[6]==g & out.data[7]==h ){  out\_d<<"passed"<<**endl**;  }  **else**{  out\_d<<"fails"<<**endl**;  fail++;  }  }  in.close();  out\_d.close();  **if** (fail==0){  cout<<"all test cases passed"<<**endl**;  }  **else**{  cout<<fail<<"\t test case failed"<<**endl**;  }  } |

Synthesis report:





Simulation:

|  |
| --- |
| INFO: [SIM 2] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CSIM start \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  INFO: [SIM 4] CSIM will launch GCC as the compiler.  Compiling ../../../mem\_tb.cpp in debug mode  Compiling ../../../mem.cpp in debug mode  Generating csim.exe  all test cases passed  INFO: [SIM 1] CSim done with 0 errors.  INFO: [SIM 3] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CSIM finish \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

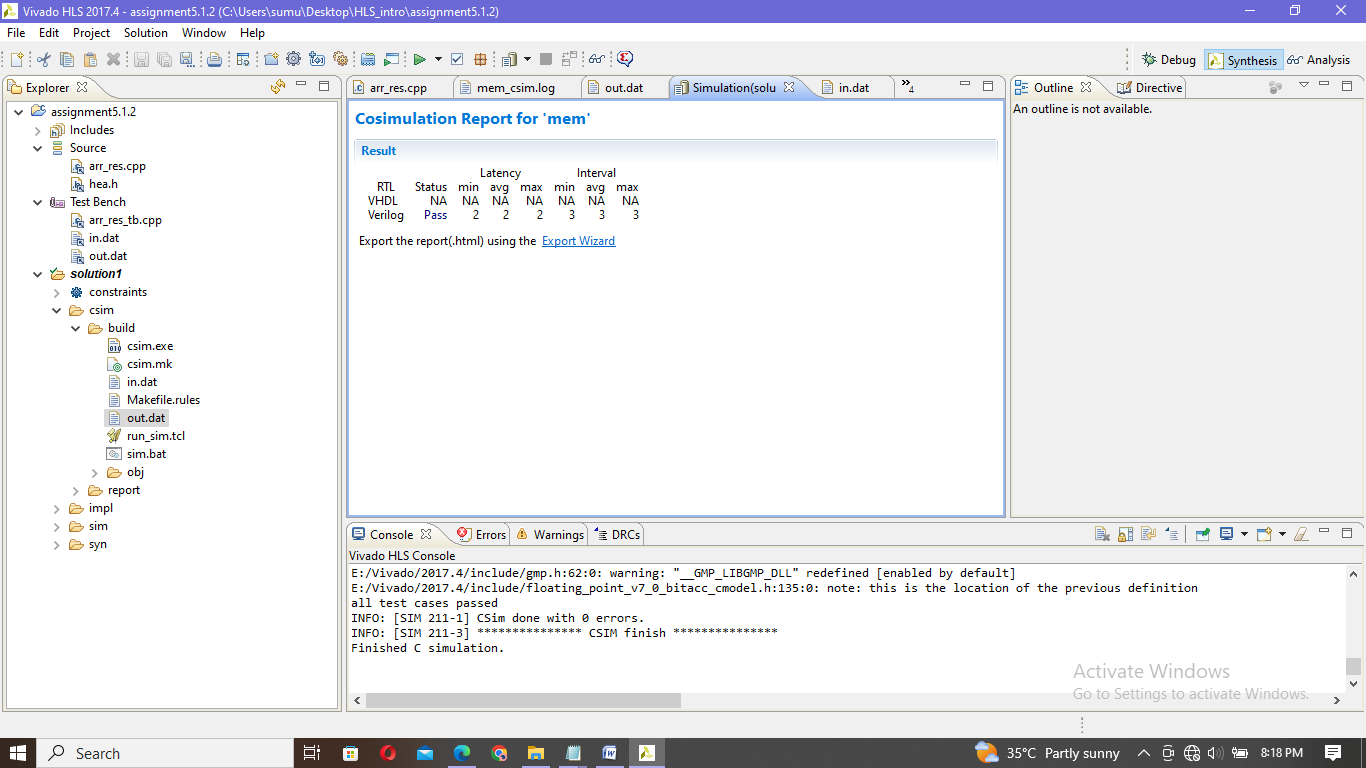
|  |
| --- |
| 8 8 16 24 32 40 48 56 64  16 16 24 32 40 48 56 64 62 |

|  |
| --- |
| 8 16 24 32 40 48 56 64 passed  16 24 32 40 48 56 64 62 passed |

Ref file result file

Cosimulation:

|  |
| --- |
| INFO: [Common 17-206] Exiting xsim at Sun Apr 2 20:08:41 2023...  INFO: [COSIM 212-316] Starting C post checking ...  all test cases passed  INFO: [COSIM 212-1000] \*\*\* C/RTL co-simulation finished: PASS \*\*\*  Finished C/RTL cosimulation. |



PART-B)

(Q)implement a DUT that takes 4 inputs in 4 clock cycles saves all the inputs in BRAM 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 recieves the final input.the BRAM index will overflow after the bram is full and it should overwrite the old values as more inputs coming in.

Header file:

|  |
| --- |
| **#include**<hls\_stream.h>  **#include** <ap\_int.h>  **#include**<iostream>  **#include** <fstream>  **using** **namespace** std;  **using** **namespace** hls;  **typedef** ap\_uint<8> WIDTH;  **typedef** ap\_uint<3> addr\_t;  **const** **int** DEPTH=8;  **const** **int** ELEMENTS=4;  **struct** bundle{  WIDTH data[ELEMENTS];  }; |

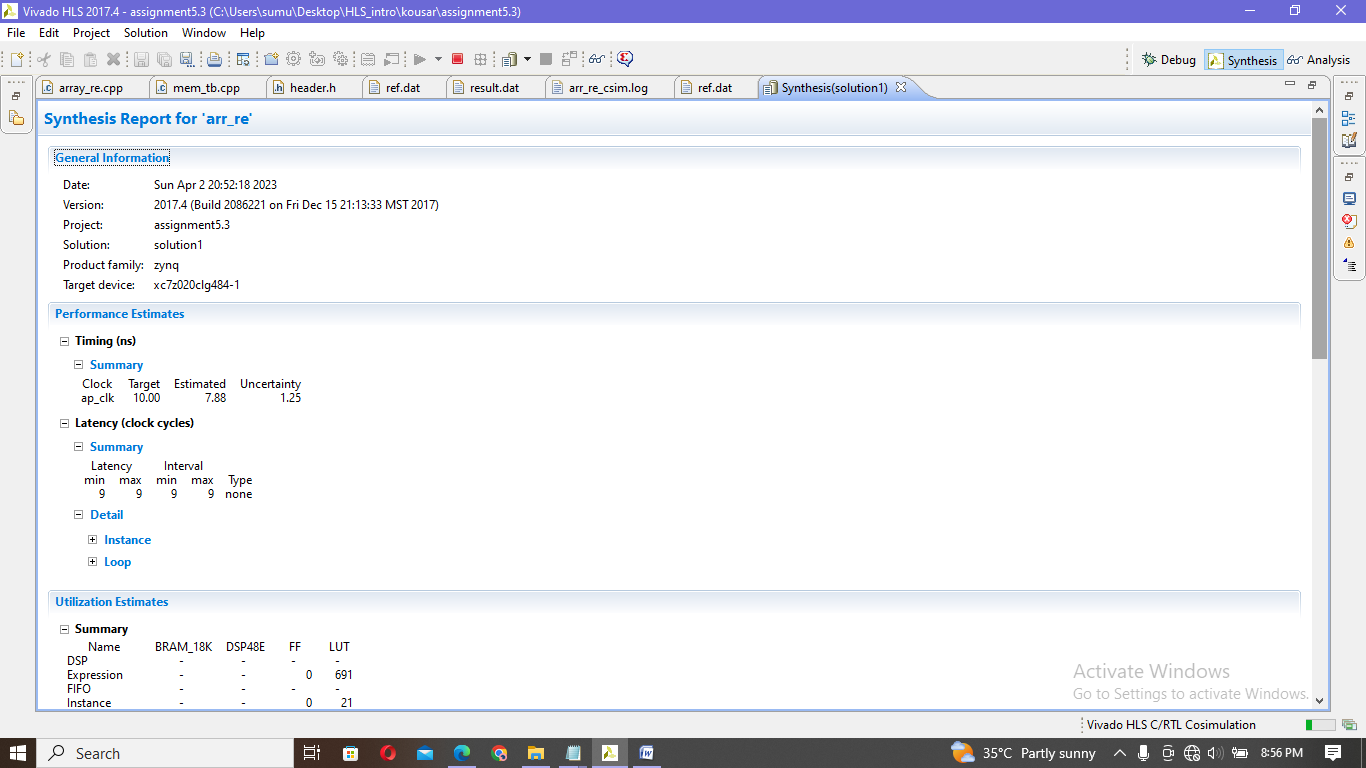
Design code:

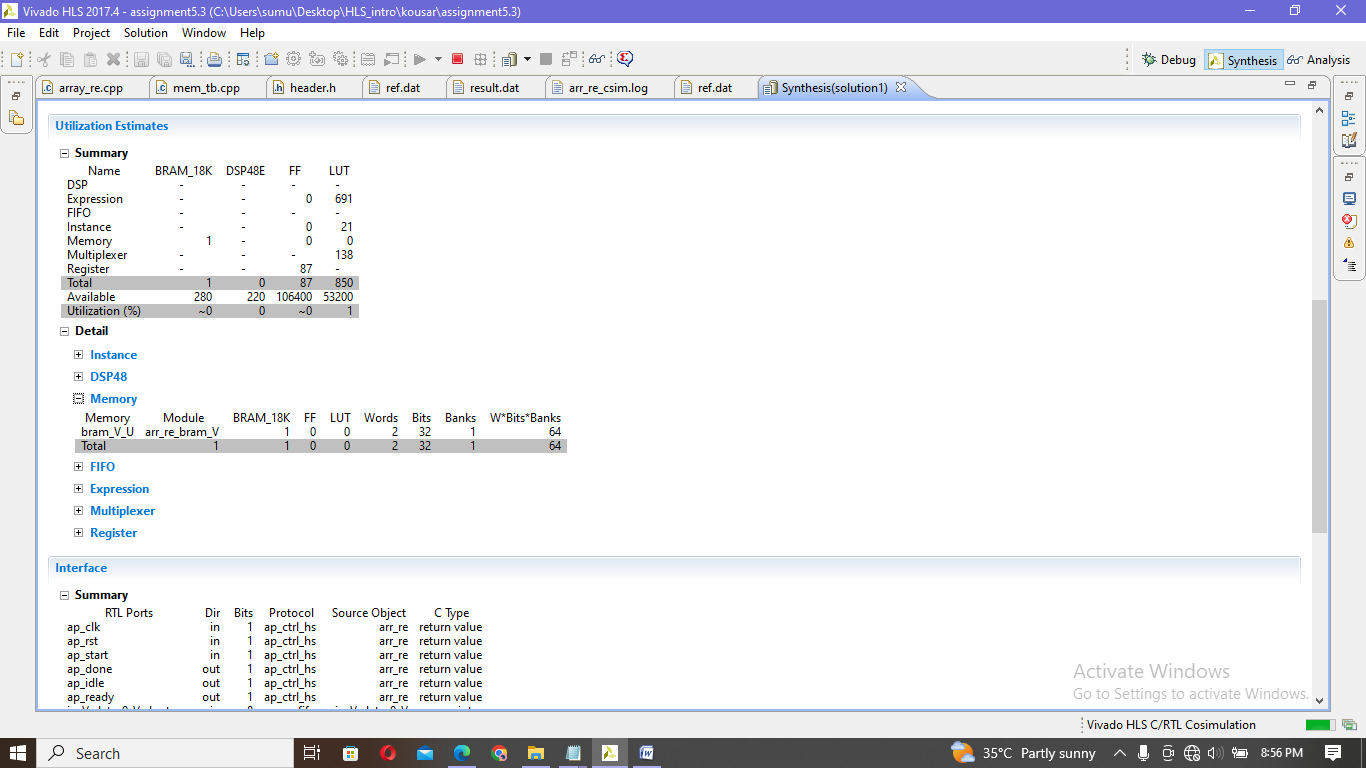
|  |
| --- |
| **#include** "header.h"  **void** **arr\_re**(stream<bundle> &in, stream<bundle> &out){  bundle input=in.read();  bundle output={0,0,0,0};  WIDTH bram[DEPTH];  **#pragma** HLS RESOURCE variable=bram core=RAM\_1P\_BRAM  **#pragma** HLS ARRAY\_RESHAPE variable=bram cyclic factor=4 dim=1  addr\_t count=0,add=0;  **int** ind=0;  //storing elements in bram  **for** (**int** i=0;i<ELEMENTS+(ELEMENTS/4)-1;i++){  **if** (count<DEPTH){  bram[add]=input.data[ind];  count++;  add++;  ind++;  //accessing elements from bram  **if** (count==4){  output.data[0]=bram[0];  output.data[1]=bram[1];  output.data[2]=bram[2];  output.data[3]=bram[3];  }  }  **else**{  //if number of elements >DEPTH ,storing from starting index  count=0;  add=0;  }  }  out<<output;  } |

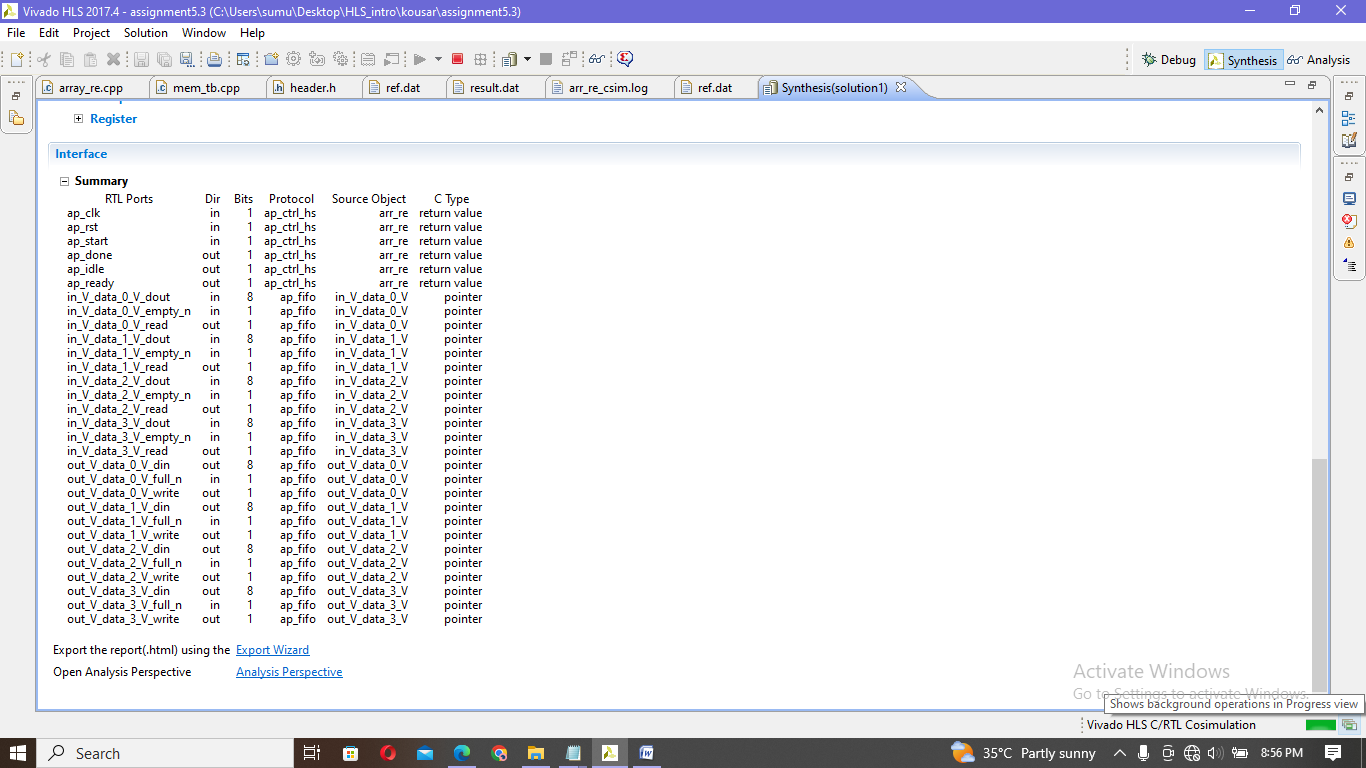
Test bench:

|  |
| --- |
| **#include** "header.h"  **void** **arr\_re**(stream<bundle> &in, stream<bundle> &out);  **int** **main**(){  stream<bundle> indata;  stream<bundle> outdata;  bundle input;  ifstream in;  fstream out\_d;  in.open("ref.dat");  out\_d.open("result.dat");  //access elements form ref.file and store in input  **int** i,index=0,fail=0;  **while** (in>>i && index<ELEMENTS){  input.data[index]=i;  index++;  }  indata<<input;  arr\_re(indata,outdata);  bundle output=outdata.read();  **for** (**int** j=0;j<4;j++){  out\_d<<output.data[j]<<"\t";  }  **if** (input.data[ELEMENTS-4]==output.data[0] & input.data[ELEMENTS-3]==output.data[1] & input.data[ELEMENTS-2]==output.data[2] & input.data[ELEMENTS-1]==output.data[3] ){  out\_d<<"passed"<<**endl**;  }  **else**{  out\_d<<"fail"<<**endl**;  fail++;  }  in.close();  out\_d.close();  **if** (fail==0){  cout<<"all passed"<<**endl**;  }  **else**{  cout<<"failed"<<**endl**;  }  } |

Synthesis report:







Simulation :

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

|  |
| --- |
| 1 2 3 5 1 2 3 5 |

|  |
| --- |
| 1 2 3 5 passed |

In file out file

Cosimulation:

|  |
| --- |
| INFO: [Common 17-206] Exiting xsim at Sun Apr 2 20:58:45 2023...  INFO: [COSIM 212-316] Starting C post checking ...  all 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. |

