HLS ASSIGNMENT-6(KANEKAL KOUSAR)

(Q) Implement a 16bit shift register in HLS. The module should take 3 inputs: 16bit data value, 16bit shift value, and 1 bit left or right shift flag. Shift value is the value by which you need to shift (in the direction denoted by shift flag) the data value and produce the output (also 16bit). Implement it in the most efficient manner possible with what you have learned till now. Do C simulation, synthesis and C/RTL co-simulation like other assignments. The testbench should be a self-checking testbench

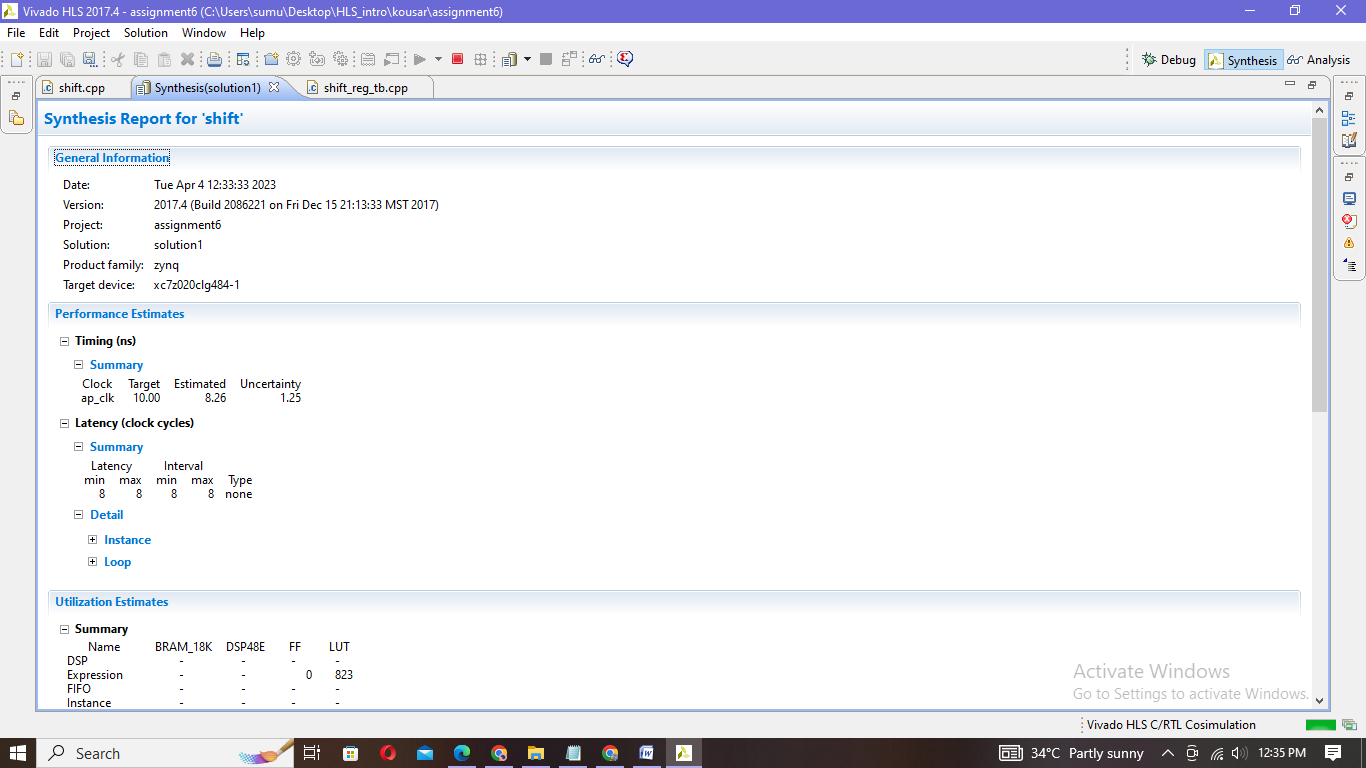
DESIGN CODE:

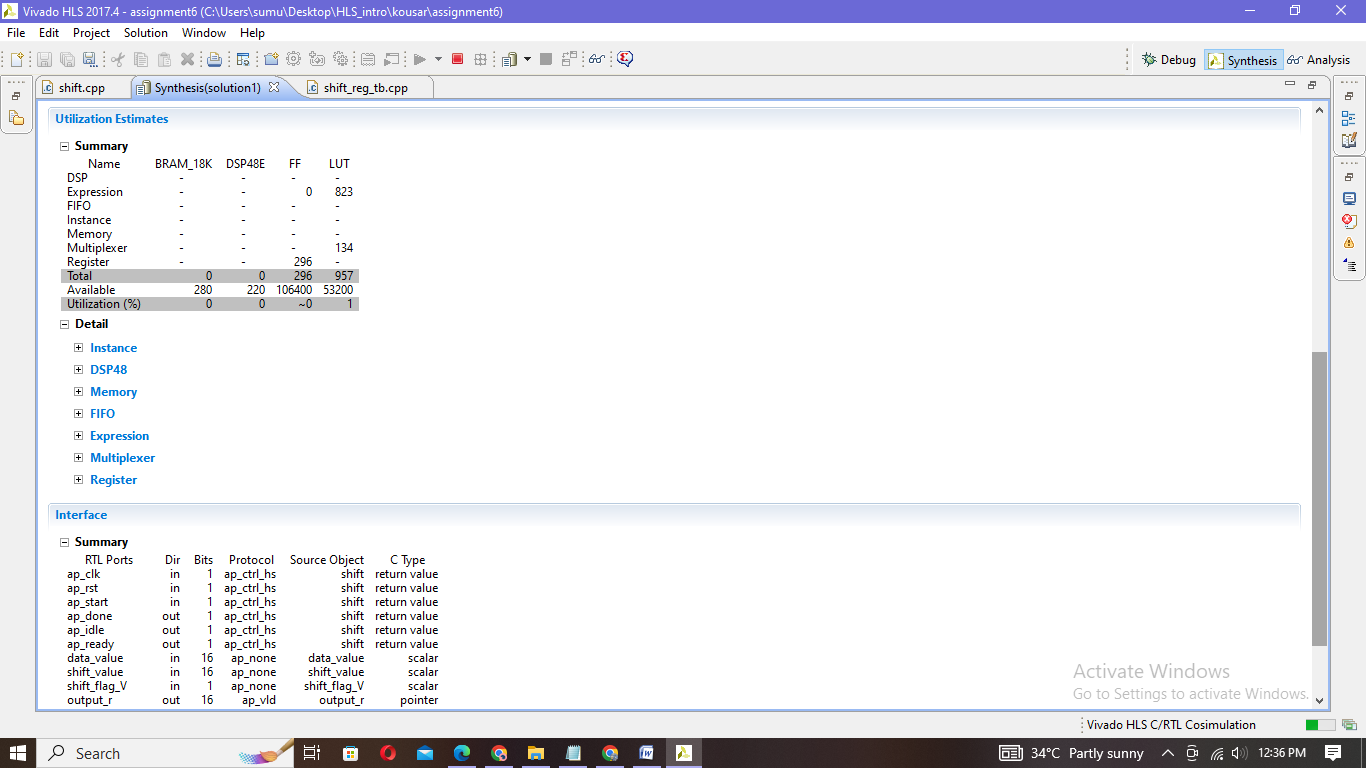
|  |
| --- |
| **#include** <ap\_int.h>  **#include** <hls\_stream.h>  **using** **namespace** hls;  **typedef** ap\_uint<1> int1\_t;  **void** **shift**(**short** data\_value,**short** shift\_value,int1\_t shift\_flag,**short** &output){  int1\_t bin[16];  **#pragma** HLS ARRAY\_RESHAPE variable=bin block factor=16 dim=1  **short** dec=0;  //converting to binary  **for** (**int** i=15;i>=0;i--){  **#pragma** HLS UNROLL  bin[15-i]=((data\_value>>i)& 1);  }  //right shift  **if** (shift\_flag){  **for** (**int** i=15;i>=shift\_value;i--){  **#pragma** HLS LOOP\_TRIPCOUNT  bin[i]=bin[i-shift\_value];  }  **for** (**int** i=0;i<shift\_value;i++){  **#pragma** HLS LOOP\_TRIPCOUNT  bin[i]=0;  }  }  //left shift  **else**{  **for** (**int** i=0;i<=15-shift\_value;i++){  **#pragma** HLS LOOP\_TRIPCOUNT  bin[i]=bin[i+shift\_value];  }  **for** (**int** i=15;i>15-shift\_value;i--){  **#pragma** HLS LOOP\_TRIPCOUNT  bin[i]=0;  }}  //binary to decimal  **for** (**int** i=0;i<16;i++){  **#pragma** HLS UNROLL  **if** (bin[i]==1){  dec+=(1<<(16-i-1));  }}  output=dec;  } |

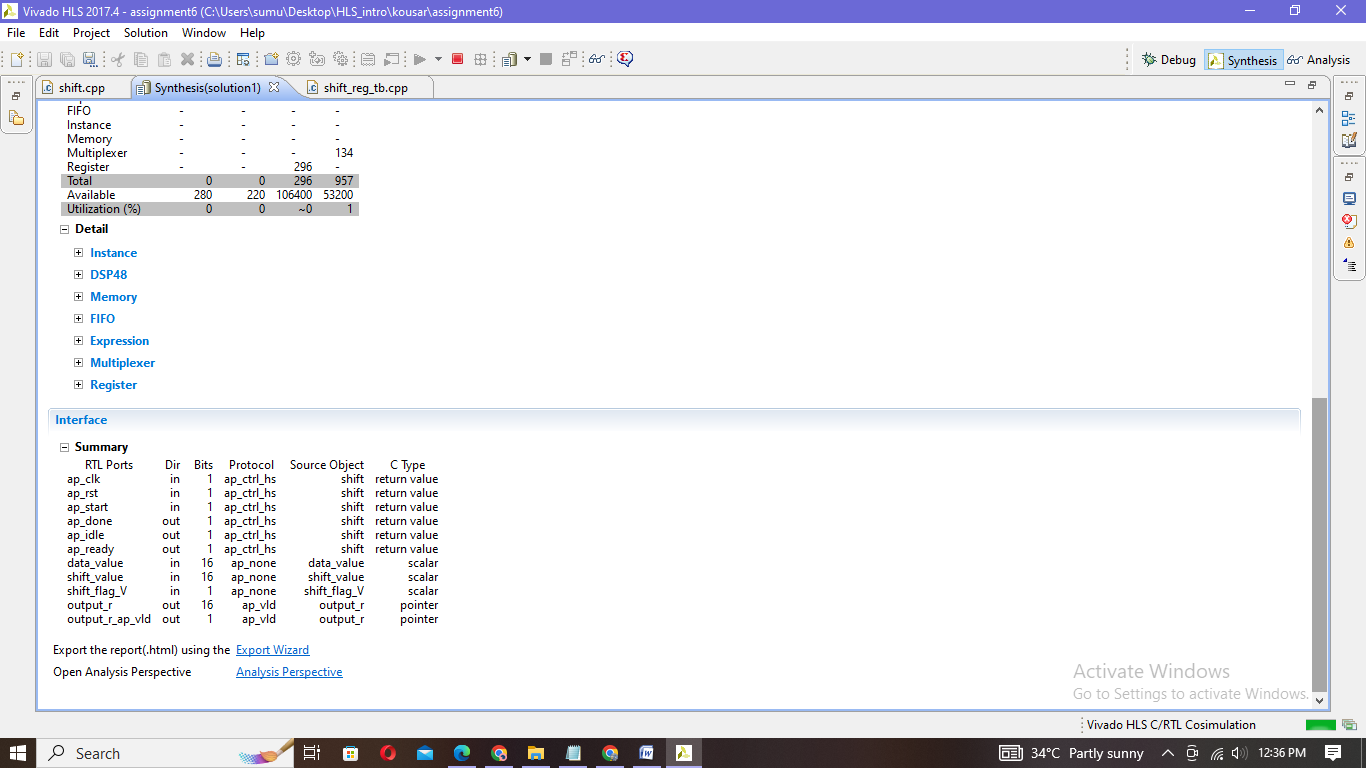
TEST BENCH:

|  |
| --- |
| **#include** <ap\_int.h>  **#include** <hls\_stream.h>  **using** **namespace** hls;  **#include** <iostream>  **using** **namespace** std;  **#include** <fstream>  **typedef** ap\_uint<1> int1\_t;  **void** **shift**(**short** data\_value,**short** shift\_value,int1\_t shift\_flag,**short** &output);  **int** **main**(){  **short** data\_value;  **short** ref,output;  **short** shift\_value;  **short** shift\_flag;  **int** case\_fail=0;  ifstream in;  fstream res;  in.open("input.dat");  res.open("output.dat");  **while** (in>>data\_value>>shift\_value>>shift\_flag>>ref){  shift(data\_value,shift\_value,shift\_flag,output);  res<<data\_value<<"\t"<<shift\_value<<"\t"<<shift\_flag<<"\t"<<output<<"\t";  **if** (ref==output){  res<<"passed"<<**endl**;  }  **else**{  res<<"fail"<<**endl**;  case\_fail++;  }  }  **if** (case\_fail==0){  cout<<"all test cases passed"<<**endl**;  }  **else**{  cout<<case\_fail<<"\t cases failed"<<**endl**;  }  } |

SYNTHESIS REPORT:







SIMULATION REPORT:

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

INPUT.DAT OUTPUT.DAT

|  |
| --- |
| 10 2 0 40  20 3 1 2  30 1 1 15  40 4 0 640  5 1 1 2 |

|  |
| --- |
| 10 2 0 40 passed  20 3 1 2 passed  30 1 1 15 passed  40 4 0 640 passed  5 1 1 2 passed |

Co-simulation:

|  |
| --- |
| INFO: [Common 17-206] Exiting xsim at Tue Apr 4 12:38:23 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. |

