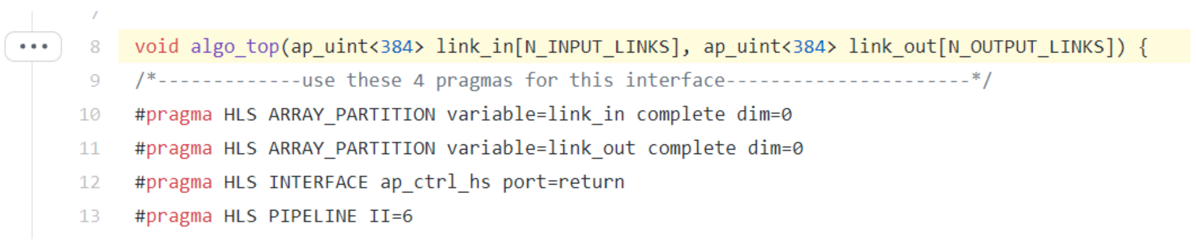
Instructions for using the new IO interface in HLS and generating the bitstream for 16Gbps IO link speed

- To use this new interface, check the following code in github (null algo*) and use the pragmas as specified in the algo_top.cpp file

  https://github.com/piyushkumarhcu/NullAlgo72IO/blob/main/vivado_hls/src/algo_top.cpp



```
void algo_top(ap_uint<384> link_in[N_INPUT_LINKS], ap_uint<384> link_out[N_OUTPUT_LINKS]) {
/*-------------use these 4 pragmas for this interface---------------------*/
#pragma HLS ARRAY_PARTITION variable=link_in complete dim=0
#pragma HLS ARRAY_PARTITION variable=link_out complete dim=0
#pragma HLS INTERFACE ap_ctrl_hs port=return
#pragma HLS PIPELINE II=6
```

- Create your new RCT (16Gbps IO) project based on the pragmas defined in the null algo

Instructions for simulation, synthesis, RTL evaluation, and the BitStream generation (17x2 RCT as a reference#)

- Clone the following github directory.

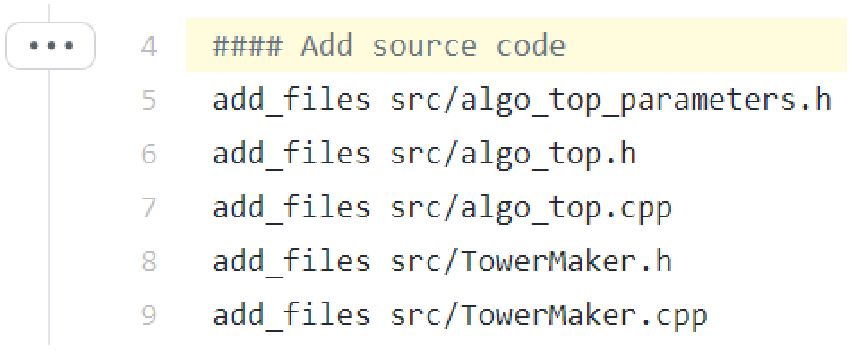
  ```
  git clone git@github.com:SridharaDasu/CMSPhase2RCT.git
  ```

- Check out to the RCT-Gen2-384-Bit branch by using the following command

  ```
  git checkout RCT-Gen2-384-Bit
  ```

- Now go to the "vivado_hls" directory of the project

- Open the "sources.tcl" file and change or add the filename in the "Add source code" section according to the source files available in your project (as shown in the following figure)



```
#### Add source code
add_files src/algo_top_parameters.h
add_files src/algo_top.h
add_files src/algo_top.cpp
add_files src/TowerMaker.h
add_files src/TowerMaker.cpp
```

- Open the “src” directory (in vivado_hls dir) and add all the relevant .cpp and .h files of your project (and remove the existing one).

- Add the following lines in your test bench (algo_top_tb.cpp)

    - Before calling the “algo_top” function

```
// Run the algorithm
ap_uint<384> test_in[N_INPUT_LINKS];
ap_uint<384> test_out[N_OUTPUT_LINKS];

for(size_t iLink=0; iLink<N_INPUT_LINKS; iLink++){
        test_in[iLink] = link_in[iLink].read().data;
}
```

    - After calling the algo_top function

```
for(size_t k=0; k<N_INPUT_LINKS; k++){
        link_out[k].write({test_out[k], 0, 1});
}
```

    - For reference check the following test bench code

https://github.com/piyushkumarhcu/RCT17x2NewInterface/blob/main/vivado_hls/src/algo_top_tb.cpp

- Now apply the test_in and test_out variable in the algo_top function

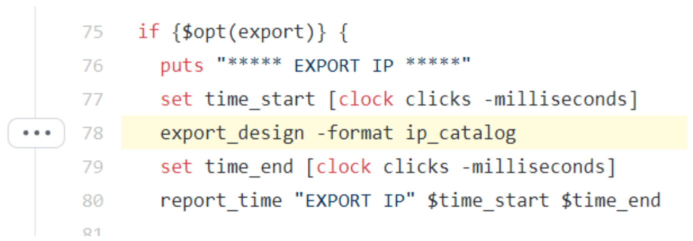
```
algo_top(test_in, test_out);
```

- Now, simulate and synthesize the project by following the instructions given on this github.

https://github.com/SridharaDasu/CMSPhase2RCT/tree/RCT-Gen2-384-Bit

- If the project is synthesized, and to perform the RTL evaluation (Vivado Place and Route), open the “run_hls.tcl” file in “vivado_hls” dir of the project.

- Change line number 78 (as shown in the diagram) as following



```
75  if {$opt(export)} {
76    puts "***** EXPORT IP *****"
77    set time_start [clock clicks -milliseconds]
78    export_design -format ip_catalog
79    set time_end [clock clicks -milliseconds]
80    report_time "EXPORT IP" $time_start $time_end
81
```

```
export_design -flow impl -rtl verilog -format ip_catalog
```

- Run the following command

```
vivado_hls -f run_hls.tcl export=1
```

- If the project passes the given timing constraint then proceed for bitstream generation
- Go to the "rtl" directory of the "hls" and remove both the wrappers and replaced it with the wrappers available in the following github directory.

https://github.com/piyushkumarhcu/RCT17x2NewInterface/tree/main/rtl

- Now run the following command available on this github to generate the bitstream

https://github.com/SridharaDasu/CMSPhase2RCT/tree/RCT-Gen2-384-Bit

```
cd $AP_FW_BASE_DIR/
git clone --recursive ssh://git@gitlab.cern.ch:7999/asvetek/phase2-rct.git -b 384b
cd $AP_FW_BASE_DIR/phase2-rct/
mkdir build
make
```

***: https://github.com/piyushkumarhcu/NullAlgo72IO**

#: https://github.com/piyushkumarhcu/RCT17x2NewInterface