Application Acceleration with High-Level Synthesis

LabC　DSP

石思宇 110061422　　邱崇喆 108061173

Overview:

The DSP Library implements a discrete Fourier transform using an FFT algorithm for acceleration on AMD Xilinx FPGAs, and it provides a fully synthesizable PL-based SSR FFT as well as a 2-dimensional FFT version.

This library currently supports two data types, fixed point and floating point complex inputs for synthesis.

Define the parameters in FFT structure:

FFT structure is defined as follows:

```
struct ssr_fft_fix_params:ssr_fft_default_params
{
        static const int N = 1024;
        static const int R = 4;
        static const scaling_mode_enum scaling_mode = SSR_FFT_NO_SCALING;
        static const fft_output_order_enum output_data_order = SSR_FFT_NATURAL;
        static const int twiddle_table_word_length = 18;
        static const int twiddle_table_integer_part_length = 2;
        static const transform_direction_enum transform_direction = FORWARD_TRANSFORM;
        static const butterfly_rnd_mode_enum butterfly_rnd_mode = TRN;
};
```

fig.1 FFT structure

In reference to the information in https://xilinx.github.io/Vitis_Libraries/dsp/2019.2/user_guide/L1.html:

- N is the size of transform
- R is the number of samples to be processed in parallel SSR Factor and radix of FFT algorithm used
- scaling_mode: The scaling mode as enumeration type (SSR FFT has three different scaling modes)
- output_data_order: Which will decide if data will be in natural order or digit reversed transposed order

- twiddle_table_word_length: Defines total number of bits to be used for storing twiddle table factors
- twiddle_table_integer_part_length: The number of integer bits used for storing integer part of twiddles
- transform_direction : Defines the direction of transform, inverse transform (SSR IFFT) or forward transform (SSR FFT)
- butterfly_rnd_mode : Defines the rounding mode used by butterflies in SSR FFT stages

Data types:

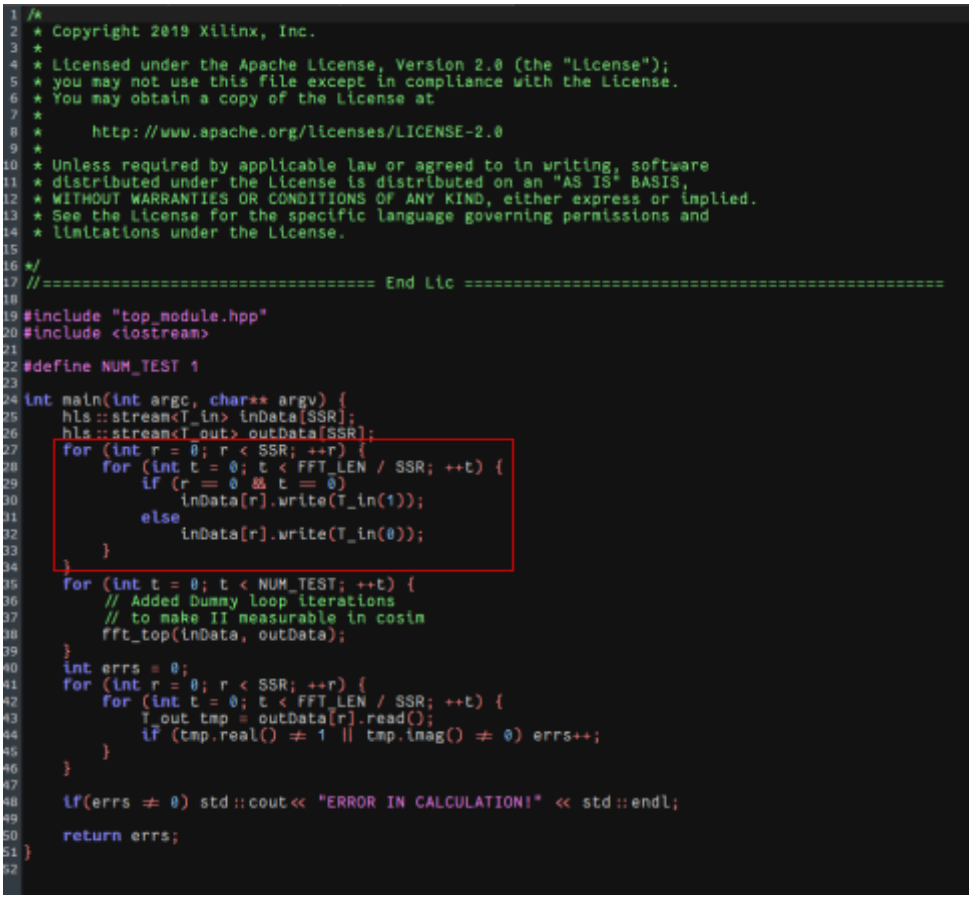
As mentioned earlier, currently the FFT supports fixed point and floating point complex input for synthesis, where fixed point data type should be declared as std::complex<ap_fixed<>> for both synthesis and simulation. While one can use floating point types **std::complex<float>** and **std::complex<double>** for simulation, they are not synthesizable because they might lead to additional resources — more DSP blocks. Instead, we can use **complex_wrapper<float>** for wrapping complex float data and synthesis.

Type	Synthesis	Simulation
std::complex <ap_fixed <>>	YES	YES
std::complex<float>	NO	YES
std::complex<double>	NO	YES
complex_wrapper<double>	NO	YES
complex_wrapper<float>	YES	YES

fig.2 table indicating which data types are synthesizable, extracted from https://xilinx.github.io/Vitis_Libraries/dsp/2019.2/user_guide/L1.html:

1D fixed point FFT with impulse test data:

Test program & test data (The red segment creates impulse data):



```
/*
 * Copyright 2019 Xilinx, Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.

*/
//================================== End Lic =================================================

#include "top_module.hpp"
#include <iostream>

#define NUM_TEST 1

int main(int argc, char** argv) {
    hls::stream<T_in> inData[SSR];
    hls::stream<T_out> outData[SSR];
    for (int r = 0; r < SSR; ++r) {
        for (int t = 0; t < FFT_LEN / SSR; ++t) {
            if (r == 0 && t == 0)
                inData[r].write(T_in(1));
            else
                inData[r].write(T_in(0));
        }
    }
    for (int t = 0; t < NUM_TEST; ++t) {
        // Added Dummy loop iterations
        // to make II measurable in cosim
        fft_top(inData, outData);
    }
    int errs = 0;
    for (int r = 0; r < SSR; ++r) {
        for (int t = 0; t < FFT_LEN / SSR; ++t) {
            T_out tmp = outData[r].read();
            if (tmp.real() != 1 || tmp.imag() != 0) errs++;
        }
    }

    if(errs != 0) std::cout<< "ERROR IN CALCULATION!" << std::endl;

    return errs;
}
```

How to run?

1. source /opt/Xilinx/Vitis/2022.1/settings64.sh
2. export DEVICE=/opt/xilinx/platforms/xilinx_u50_gen3x16_xdma_5_202210_1/xilinx_u50_gen3x16_xdma_5_202210_1.xpfm

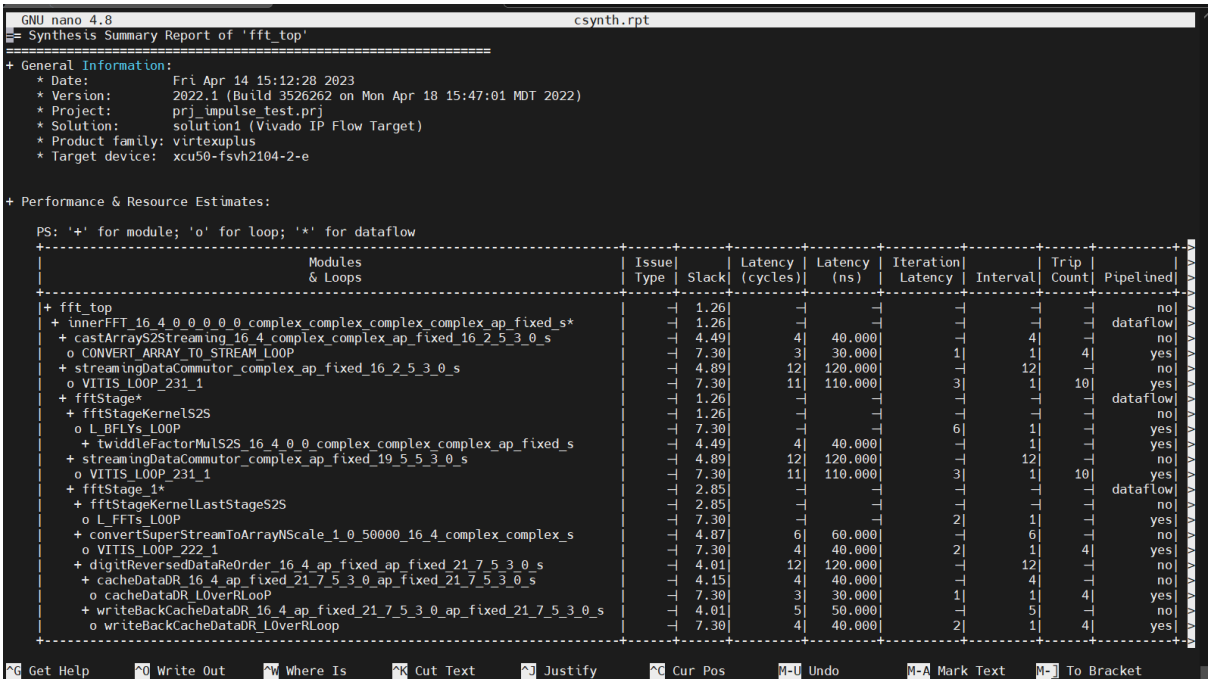
3. modify the Makefile as the following:

```
184
185 # Alias to run, for legacy test script
186 check: run
187
188 CSIM ?= 1
189 CSYNTH ?= 1
190 COSIM ?= 1
191 VIVADO_SYN ?= 0
192 VIVADO_IMPL ?= 0
193 QOR_CHECK ?= 0
194
195 # at least RTL synthesis before check QoR
196 ifeq (1,$(QOR_CHECK))
197 ifeq (0,$(VIVADO_IMPL))
198 override VIVADO_SYN := 1
199 endif
200 endif
201
202 # need synthesis before cosim or vivado
203 ifeq (1,$(VIVADO_IMPL))
204 override CSYNTH := 1
205 endif
206
207 ifeq (1,$(VIVADO_SYN))
208 override CSYNTH := 1
209 endif
210
211 ifeq (1,$(COSIM))
212 override CSYNTH := 1
213 endif
```

4. make run

We'll can see the project have been created named proj_impulse_test.prj, inside which we can find the the reports generated by vitis_hls:

synthesis:



```
  GNU nano 4.8                                                csynth.rpt
== Synthesis Summary Report of 'fft_top'
================================================================
+ General Information:
    * Date:           Fri Apr 14 15:12:28 2023
    * Version:        2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)
    * Project:        prj_impulse_test.prj
    * Solution:       solution1 (Vivado IP Flow Target)
    * Product family: virtexuplus
    * Target device:  xcu50-fsvh2104-2-e


+ Performance & Resource Estimates:

    PS: '+' for module; 'o' for loop; '*' for dataflow
    +-----------------------------------------------------------------------------+------+------+---------+---------+---------+---------+------+----------+
    |                                Modules                                      | Issue|      | Latency | Latency | Iteration|         | Trip |          |
    |                                & Loops                                      | Type | Slack| (cycles)|   (ns)  |  Latency | Interval| Count| Pipelined|
    +-----------------------------------------------------------------------------+------+------+---------+---------+---------+---------+------+----------+
    |+ fft_top                                                                    |     -|  1.26|        -|        -|        -|        -|     -|        no|
    | + innerFFT_16_4_0_0_0_0_0_complex_complex_complex_complex_ap_fixed_s*       |     -|  1.26|        -|        -|        -|        -|     -|  dataflow|
    |  + castArrayS2Streaming_16_4_complex_complex_ap_fixed_16_2_5_3_0_s          |     -|  4.49|        4|  40.000|        -|        4|     -|        no|
    |   o CONVERT_ARRAY_TO_STREAM_LOOP                                            |     -|  7.30|        3|  30.000|        1|        1|     4|       yes|
    |  + streamingDataCommutor_complex_ap_fixed_16_2_5_3_0_s                      |     -|  4.89|       12|  120.000|        -|       12|     -|        no|
    |   o VITIS_LOOP_231_1                                                        |     -|  7.30|       11|  110.000|        3|        1|    10|       yes|
    |  + fftStage*                                                                |     -|  1.26|        -|        -|        -|        -|     -|  dataflow|
    |   + fftStageKernelS2S                                                       |     -|  1.26|        -|        -|        -|        -|     -|        no|
    |    o L_BFLYs_LOOP                                                           |     -|  7.30|        -|        -|        6|        1|     -|       yes|
    |     + twiddleFactorMulS2S_16_4_0_0_complex_complex_complex_ap_fixed_s       |     -|  4.49|        4|  40.000|        -|        1|     -|       yes|
    |   + streamingDataCommutor_complex_ap_fixed_19_5_5_3_0_s                     |     -|  4.89|       12|  120.000|        -|       12|     -|        no|
    |    o VITIS_LOOP_231_1                                                       |     -|  7.30|       11|  110.000|        3|        1|    10|       yes|
    |   + fftStage_1*                                                             |     -|  2.85|        -|        -|        -|        -|     -|  dataflow|
    |    + fftStageKernelLastStageS2S                                             |     -|  2.85|        -|        -|        -|        -|     -|        no|
    |     o L_FFTs_LOOP                                                           |     -|  7.30|        -|        -|        2|        1|     -|       yes|
    |    + convertSuperStreamToArrayNScale_1_0_50000_16_4_complex_complex_s       |     -|  4.87|        6|  60.000|        -|        6|     -|        no|
    |     o VITIS_LOOP_222_1                                                      |     -|  7.30|        4|  40.000|        2|        1|     4|       yes|
    |    + digitReversedDataReOrder_16_4_ap_fixed_ap_fixed_21_7_5_3_0_s           |     -|  4.01|       12|  120.000|        -|       12|     -|        no|
    |     + cacheDataDR_16_4_ap_fixed_21_7_5_3_0_ap_fixed_21_7_5_3_0_s            |     -|  4.15|        4|  40.000|        -|        4|     -|        no|
    |      o cacheDataDR_LOverRLooP                                               |     -|  7.30|        3|  30.000|        1|        1|     4|       yes|
    |     + writeBackCacheDataDR_16_4_ap_fixed_21_7_5_3_0_ap_fixed_21_7_5_3_0_s   |     -|  4.01|        5|  50.000|        -|        5|     -|        no|
    |      o writeBackCacheDataDR_LOverRLoop                                      |     -|  7.30|        4|  40.000|        2|        1|     4|       yes|
    +-----------------------------------------------------------------------------+------+------+---------+---------+---------+---------+------+----------+

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  M-U Undo  M-A Mark Text  M-] To Bracket
```

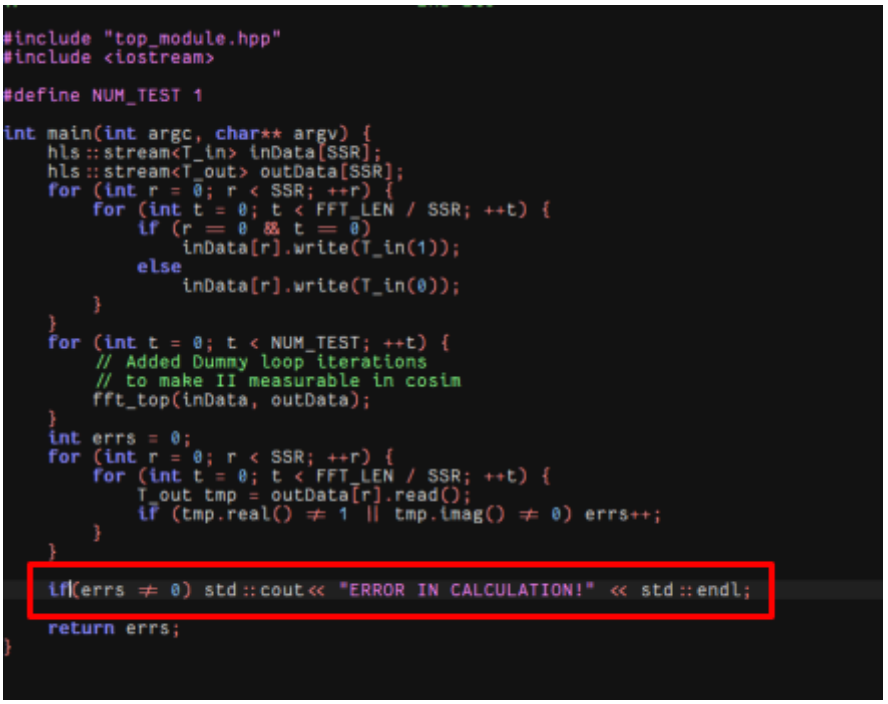
```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+------+---------+--------+-----+
|         Name        | BRAM_18K|  DSP |    FF   |   LUT  | URAM|
+---------------------+---------+------+---------+--------+-----+
|DSP                  |        -|     -|        -|       -|    -|
|Expression           |        -|     -|        0|       4|    -|
|FIFO                 |        -|     -|        -|       -|    -|
|Instance             |        0|    12|     6897|    8820|    0|
|Memory               |        -|     -|        -|       -|    -|
|Multiplexer          |        -|     -|        -|      86|    -|
|Register             |        -|     -|        5|       -|    -|
+---------------------+---------+------+---------+--------+-----+
|Total                |        0|    12|     6902|    8910|    0|
+---------------------+---------+------+---------+--------+-----+
|Available SLR        |     1344|  2976|   871680|  435840|  320|
+---------------------+---------+------+---------+--------+-----+
|Utilization SLR (%)  |        0|    ~0|       ~0|       2|    0|
+---------------------+---------+------+---------+--------+-----+
|Available            |     2688|  5952|  1743360|  871680|  640|
+---------------------+---------+------+---------+--------+-----+
|Utilization (%)      |        0|    ~0|       ~0|       1|    0|
+---------------------+---------+------+---------+--------+-----+
```

fig.3 csynth.rpt & fig.4 fft_top_csynth.rpt

c-sim:

fig.5 c-simulation result

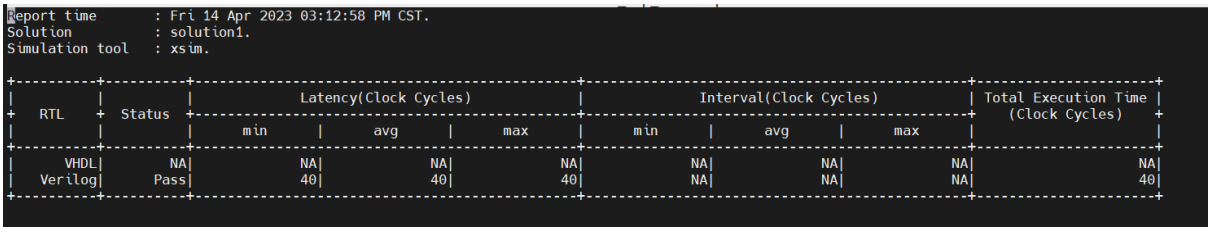
Note: There was nothing printed on the log file, which means the result was correct, because in main.cpp we added a “cout” to ensure the output of an impulse input is one:



```cpp
#include "top_module.hpp"
#include <iostream>

#define NUM_TEST 1

int main(int argc, char** argv) {
    hls::stream<T_in> inData[SSR];
    hls::stream<T_out> outData[SSR];
    for (int r = 0; r < SSR; ++r) {
        for (int t = 0; t < FFT_LEN / SSR; ++t) {
            if (r == 0 && t == 0)
                inData[r].write(T_in(1));
            else
                inData[r].write(T_in(0));
        }
    }
    for (int t = 0; t < NUM_TEST; ++t) {
        // Added Dummy loop iterations
        // to make II measurable in cosim
        fft_top(inData, outData);
    }
    int errs = 0;
    for (int r = 0; r < SSR; ++r) {
        for (int t = 0; t < FFT_LEN / SSR; ++t) {
            T_out tmp = outData[r].read();
            if (tmp.real() != 1 || tmp.imag() != 0) errs++;
        }
    }
    if(errs != 0) std::cout<< "ERROR IN CALCULATION!" << std::endl;
    return errs;
}
```

fig.6 testbench (main.cpp) modification

co-sim:



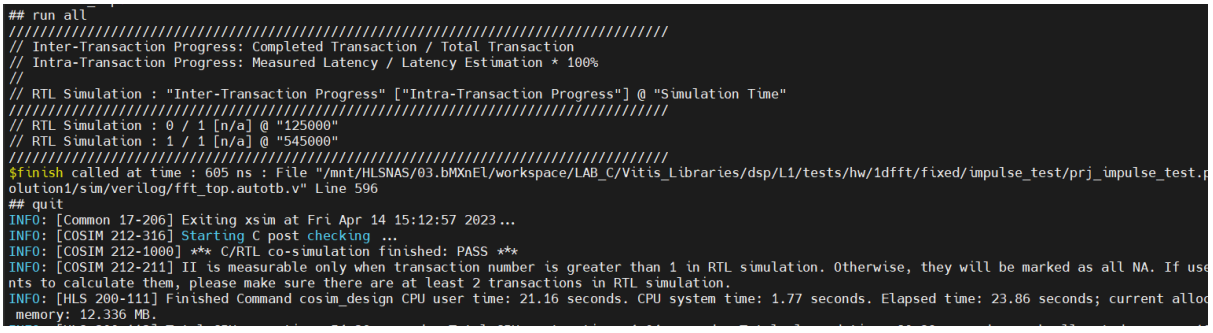


fig.7 co-sim result

Multiple instances:
We can create multiple instance of SSR FFT by changing the second template integer parameter:

```
xf::dsp::fft::fft<fftParams,1><ssr_fft_fix_params>(...);
xf::dsp::fft::fft<fftParams,2><ssr_fft_fix_params>(...);
```

The following showcase a simple code for creating one more instance to run FFT of constant one (or the inverse FFT of the previous result), by modifying the main.cpp and top_module :

```
#define NUM_TEST 1

int main(int argc, char** argv) {
    hls::stream<T_in> inData[SSR];
    hls::stream<T_out> outData[SSR];
    hls::stream<T_in> ConstOne[SSR];
    hls::stream<T_out> FFT_ConstOne[SSR];

    for (int r = 0; r < SSR; ++r) {
        for (int t = 0; t < FFT_LEN / SSR; ++t) {
            ConstOne[r].write(T_in(1));
            if (r == 0 && t == 0)
                inData[r].write(T_in(1));
            else
                inData[r].write(T_in(0));
        }
    }


    for (int t = 0; t < NUM_TEST; ++t) {
        // Added Dummy loop iterations
        // to make II measurable in cosim
        fft_top(inData, outData, ConstOne, FFT_ConstOne);
    }
    int errs = 0;
    for (int r = 0; r < SSR; ++r) {
        for (int t = 0; t < FFT_LEN / SSR; ++t) {
            T_out tmp = outData[r].read();
            if (tmp.real() != 1 || tmp.imag() != 0) errs++;
        }
    }

    if(errs != 0) std::cout<< "ERROR IN CALCULATION!" << std::endl;

    bool err_flag = false;
    for (int r = 0; r < SSR; ++r) {
        for (int t = 0; t < FFT_LEN / SSR; ++t) {
            T_out tmp = FFT_ConstOne[r].read();
            int tmp_real=tmp.real();
            if (r == 0 && t == 0){
              if(tmp_real == 0){
                err_flag = true;
              }
            }
            else if(tmp_real != 0){
              err_flag = true;
            }
        }
    }

    if(err_flag) std::cout<< "ERROR IN CALCULATION !!" << std::endl;

    return errs;
}
```

fig.8 modified version of tb for testing multiple instance

Define IID_NEW for new instance:

```
// Define FFT Size and Super Sample Rate
#define FFT_LEN 16
#define SSR 4

#define IN_WL 16
#define IN_IL 2
#define TW_WL 16
#define TW_IL 2
#define IID 0
#define IID_NEW 1
```

```
#include "top_module.hpp"
#include "data_path.hpp"
void fft_top(hls::stream<T_in> p_inData[SSR], hls::stream<T_out> p_outData[SSR], hls::stream<T_in> ConstOne[SSR], hls::stream<T_out> FFT_ConstOne[SSR]) {
    xf::dsp::fft::fft<fftParams, IID>(p_inData, p_outData);
    xf::dsp::fft::fft<fftParams, IID_NEW>(ConstOne, FFT_ConstOne);
}
```

fig.9 modification of top function — add an instance

we can see every resource in utilization estimates of the top function were nearly doubled, because we instantiated another one.

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+------+---------+--------+-----+
|         Name        | BRAM_18K|  DSP |    FF   |   LUT  | URAM|
+---------------------+---------+------+---------+--------+-----+
|DSP                  |        -|     -|        -|       -|    -|
|Expression           |        -|     -|        0|      10|    -|
|FIFO                 |        -|     -|        -|       -|    -|
|Instance             |        0|    28|    13806|   18224|    0|
|Memory               |        -|     -|        -|       -|    -|
|Multiplexer          |        -|     -|        -|     158|    -|
|Register             |        -|     -|        8|       -|    -|
+---------------------+---------+------+---------+--------+-----+
|Total                |        0|    28|    13814|   18392|    0|
+---------------------+---------+------+---------+--------+-----+
|Available SLR        |     1344|  2976|   871680|  435840|  320|
+---------------------+---------+------+---------+--------+-----+
|Utilization SLR (%)  |        0|    ~0|        1|       4|    0|
+---------------------+---------+------+---------+--------+-----+
|Available            |     2688|  5952|  1743360|  871680|  640|
+---------------------+---------+------+---------+--------+-----+
|Utilization (%)      |        0|    ~0|       ~0|       2|    0|
+---------------------+---------+------+---------+--------+-----+
```

fig.9 utilization estimates of creating two instances

and the c-sim and co-sim were also successfully run:

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
make[1]: Entering directory '/mnt/HLSNAS/03.bMXnEl/workspace/LAB_C/Vitis_Libraries/dsp/L1/tests/hw/1dfft/fixed/impulse_test/prj_impulse_test.prj/solution1/csim/build'
   Compiling ../../../../src/main.cpp in debug mode
   Generating csim.exe
make[1]: Leaving directory '/mnt/HLSNAS/03.bMXnEl/workspace/LAB_C/Vitis_Libraries/dsp/L1/tests/hw/1dfft/fixed/impulse_test/prj_impulse_test.prj/solution1/csim/build'
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

```
## run all
////////////////////////////////////////////////////////////////////////////////
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @ "Simulation Time"
////////////////////////////////////////////////////////////////////////////////
// RTL Simulation : 0 / 1 [n/a] @ "125000"
// RTL Simulation : 1 / 1 [n/a] @ "545000"
////////////////////////////////////////////////////////////////////////////////
$finish called at time : 605 ns : File "/mnt/HLSNAS/03.bMXnEl/workspace/LAB_C/Vitis_Libraries/dsp/L1/tests/hw/1dfft/fixed/impulse_test/prj_impulse_test.prj/solution1/sim/verilog/fft_top.autotb.v" Line 1032
## quit
INFO: [Common 17-206] Exiting xsim at Fri Apr 14 19:53:41 2023...
INFO: [COSIM 212-316] Starting C post checking ...
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.
INFO: [HLS 200-111] Finished Command cosim_design CPU user time: 26.26 seconds. CPU system time: 1.93 seconds. Elapsed time: 22.99 seconds; current allocated memory: 10.289 MB.
INFO: [HLS 200-112] Total CPU user time: 69.71 seconds. Total CPU system time: 4.65 seconds. Total elapsed time: 98.78 seconds; peak allocated memory: 1.371 GB.
INFO: [Common 17-206] Exiting vitis_hls at Fri Apr 14 19:53:43 2023...
```

fig.10 successful c-sim & co-sim of creating two instances

========== ==========

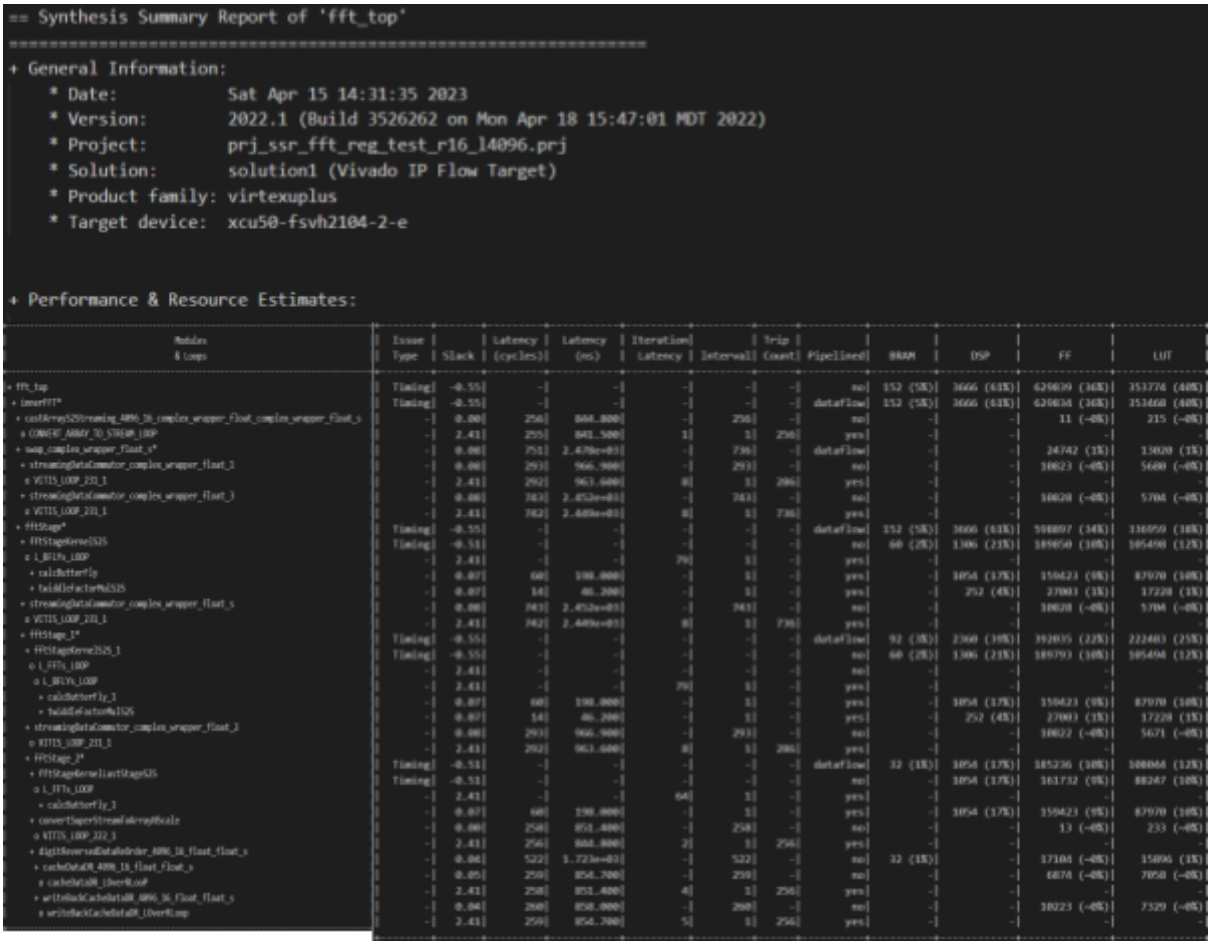
1D float point FFT with test data in /L1/tests/common_float/verif/:

The test data input file used in this experiment is fftStimulusIn_L4096.verif, and the golden output file used for verification is fftGoldenOut_L4096.verif:

```
4096
9.20659596040256644756
12.92746544283520648833
5.93800120776925144384
-2.86931512498761431829
9.91246943611761288651
-2.66690187629595776286
-17.63462151916279907482
-19.95775095737172577515
-31.32222132136553938153
-3.08521101278948517432
-7.79760667271630225628
-5.79936656748928491822
7.56072172358180694118
-7.09100249994199849368
-24.50207072565649468743
-1.76773374266216953821
2.11522636081379555861
-3.00672079096654876196
26.42191696659537214487
-8.13999728384190035513
-0.81171396629295222702
-40.51150184918373753362
0.33088126730945377485
16.97813990267743378126
17.96349704351164788818
-9.87015850993069676633
-12.27577156099332356121
-14.76402930727036988401
1.63618061082191768030
-39.62788907700327456496
```

fig.11 a snippet of test data in fftStimulusIn_L4096.verif

synthesis summary:



```
== Synthesis Summary Report of 'fft_top'
================================================================
+ General Information:
    * Date:           Sat Apr 15 14:31:35 2023
    * Version:        2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)
    * Project:        prj_ssr_fft_reg_test_r16_l4096.prj
    * Solution:       solution1 (Vivado IP Flow Target)
    * Product family: virtexuplus
    * Target device:  xcu50-fsvh2104-2-e


+ Performance & Resource Estimates:
```

Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT
+ fft_top	Timing	-0.55	-	-	-	-	-	no	152 (5%)	3666 (61%)	629839 (36%)	353774 (40%)
+ innerFFT*	Timing	-0.55	-	-	-	-	-	dataflow	152 (5%)	3666 (61%)	629834 (36%)	353668 (40%)
+ castArray52Streaming_ARRA_16_complex_wrapper_float_complex_wrapper_float_s		-	0.00	256 | 864.000	-	256	-	no	-	-	11 (~0%)	215 (~0%)
o CONVERT_ARRAY_TO_STREAM_LOOP		-	2.41	255 | 861.500	1	1	256	yes	-	-	-	-
+ swap_complex_wrapper_float_s*		-	0.00	751 | 2.678e+03	-	736	-	dataflow	-	-	24742 (1%)	13020 (1%)
+ streamingDataCommutor_complex_wrapper_float_1		-	0.00	293 | 966.900	-	293	-	no	-	-	10823 (~0%)	5680 (~0%)
o VITIS_LOOP_233_1		-	2.41	292 | 963.600	8	1	286	yes	-	-	-	-
+ streamingDataCommutor_complex_wrapper_float_2		-	0.00	743 | 2.452e+03	-	743	-	no	-	-	10828 (~0%)	5704 (~0%)
o VITIS_LOOP_233_1		-	2.41	742 | 2.449e+03	8	1	736	yes	-	-	-	-
+ fftStage*	Timing	-0.55	-	-	-	-	-	dataflow	152 (5%)	3666 (61%)	588897 (34%)	336959 (38%)
+ fftStageKernel525	Timing	-0.51	-	-	-	-	-	no	60 (2%)	1306 (21%)	189850 (10%)	105498 (12%)
o L_BFLYs_LOOP		-	2.41	- | -	79	1	-	yes	-	-	-	-
+ calcButterfly		-	0.07	60 | 198.000	-	1	-	yes	-	1056 (17%)	159423 (9%)	87970 (10%)
+ twiddleFactorMul525		-	0.07	14 | 46.200	-	1	-	yes	-	252 (4%)	27083 (1%)	17228 (1%)
+ streamingDataCommutor_complex_wrapper_float_s		-	0.00	743 | 2.452e+03	-	743	-	no	-	-	10828 (~0%)	5704 (~0%)
o VITIS_LOOP_233_1		-	2.41	742 | 2.449e+03	8	1	736	yes	-	-	-	-
+ fftStage_1*	Timing	-0.55	-	-	-	-	-	dataflow	92 (3%)	2360 (39%)	392035 (22%)	222483 (25%)
+ fftStageKernel525_1	Timing	-0.55	-	-	-	-	-	no	60 (2%)	1306 (21%)	189793 (10%)	105498 (12%)
o L_FFTs_LOOP		-	2.41	- | -	-	-	-	no	-	-	-	-
o L_BFLYs_LOOP		-	2.41	- | -	79	1	-	yes	-	-	-	-
+ calcButterfly_1		-	0.07	60 | 198.000	-	1	-	yes	-	1056 (17%)	159423 (9%)	87970 (10%)
+ twiddleFactorMul525		-	0.07	14 | 46.200	-	1	-	yes	-	252 (4%)	27083 (1%)	17228 (1%)
+ streamingDataCommutor_complex_wrapper_float_3		-	0.00	293 | 966.900	-	293	-	no	-	-	10822 (~0%)	5671 (~0%)
o VITIS_LOOP_233_1		-	2.41	292 | 963.600	8	1	286	yes	-	-	-	-
+ fftStage_2*	Timing	-0.51	-	-	-	-	-	dataflow	32 (1%)	1054 (17%)	185236 (10%)	108068 (12%)
+ fftStageKernelLastStage525	Timing	-0.51	-	-	-	-	-	no	-	1054 (17%)	161732 (9%)	88247 (10%)
o L_FFTs_LOOP		-	2.41	- | -	64	1	-	yes	-	-	-	-
+ calcButterfly_2		-	0.07	60 | 198.000	-	1	-	yes	-	1054 (17%)	159423 (9%)	87970 (10%)
+ convertSuperStreamToArrayBScale		-	0.00	258 | 851.400	-	258	-	no	-	-	13 (~0%)	233 (~0%)
o VITIS_LOOP_232_1		-	2.41	256 | 844.800	2	1	256	yes	-	-	-	-
+ digitReverseDataReorder_ARRA_16_float_float_s		-	0.04	522 | 1.723e+03	-	522	-	no	32 (1%)	-	17104 (~0%)	15896 (1%)
+ cacheDataOR_ARRA_16_float_float_s		-	0.05	259 | 854.700	-	259	-	no	-	-	6878 (~0%)	7058 (~0%)
o cacheDataOR_1OverR.coP		-	2.41	258 | 851.400	4	1	256	yes	-	-	-	-
+ writeBackCacheDataOR_ARRA_16_float_float_s		-	0.04	260 | 858.000	-	260	-	no	-	-	10223 (~0%)	7329 (~0%)
o writeBackCacheDataOR_1OverR.cop		-	2.41	259 | 854.700	5	1	256	yes	-	-	-	-

c-sim:

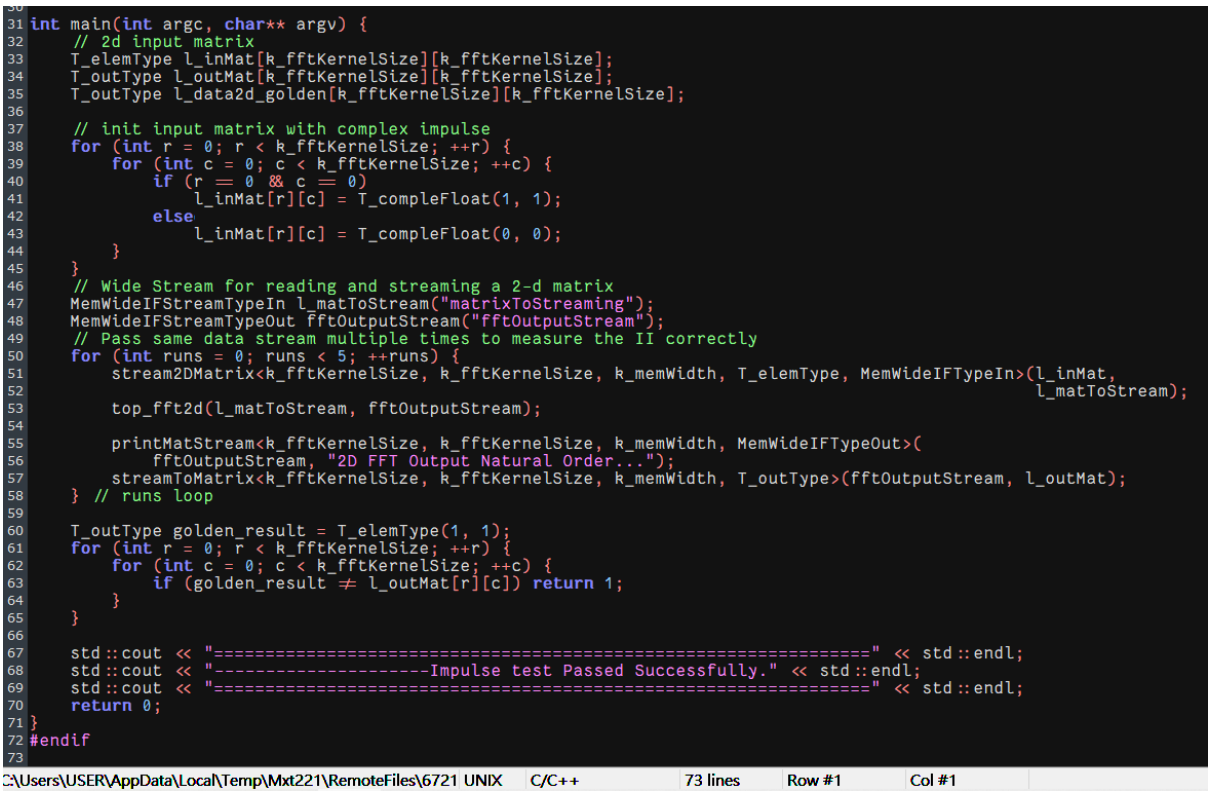
```
247
248 s-p=========================================================================================================s-p
249 OVERL ALL Simulation was SUCCESSFULL Done with L=4096  R=16
250 SNR Double precision Model  : 301.133 dbs
251 SNR Single Precision Model        : 137.771 dbs
252 e-p=========================================================================================================e-p
253
254 INFO: [SIM 1] CSim done with 0 errors.
255 INFO: [SIM 3] *************** CSIM finish ***************
256
```

co-sim:

```
Report time       : Sat 15 Apr 2023 02:37:14 PM CST.
Solution          : solution1.
Simulation tool   : xsim.

+----------+---------+-----------------------------------------+-----------------------------------------+---------------------+
|          |         |            Latency(Clock Cycles)        |            Interval(Clock Cycles)       | Total Execution Time |
|   RTL    |  Status +-----------------------------------------+-----------------------------------------+    (Clock Cycles)   |
|          |         |     min     |     avg     |     max     |     min     |     avg     |     max     |                     |
+----------+---------+-------------+-------------+-------------+-------------+-------------+-------------+---------------------+
|      VHDL|       NA|           NA|           NA|           NA|           NA|           NA|           NA|                   NA|
|   Verilog|     Pass|         2497|         2497|         2497|           NA|           NA|           NA|                 2497|
+----------+---------+-------------+-------------+-------------+-------------+-------------+-------------+---------------------+
```

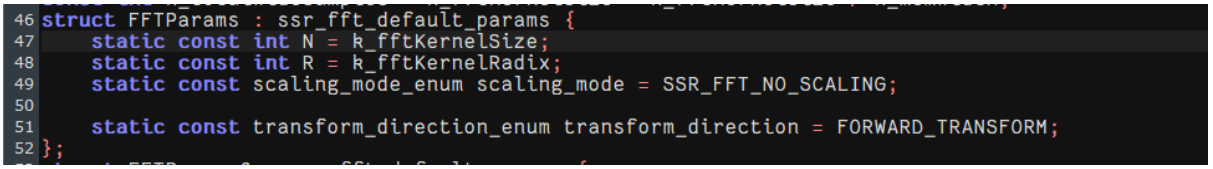
2D fixed point FFT with impulse test data:

test program and test data:



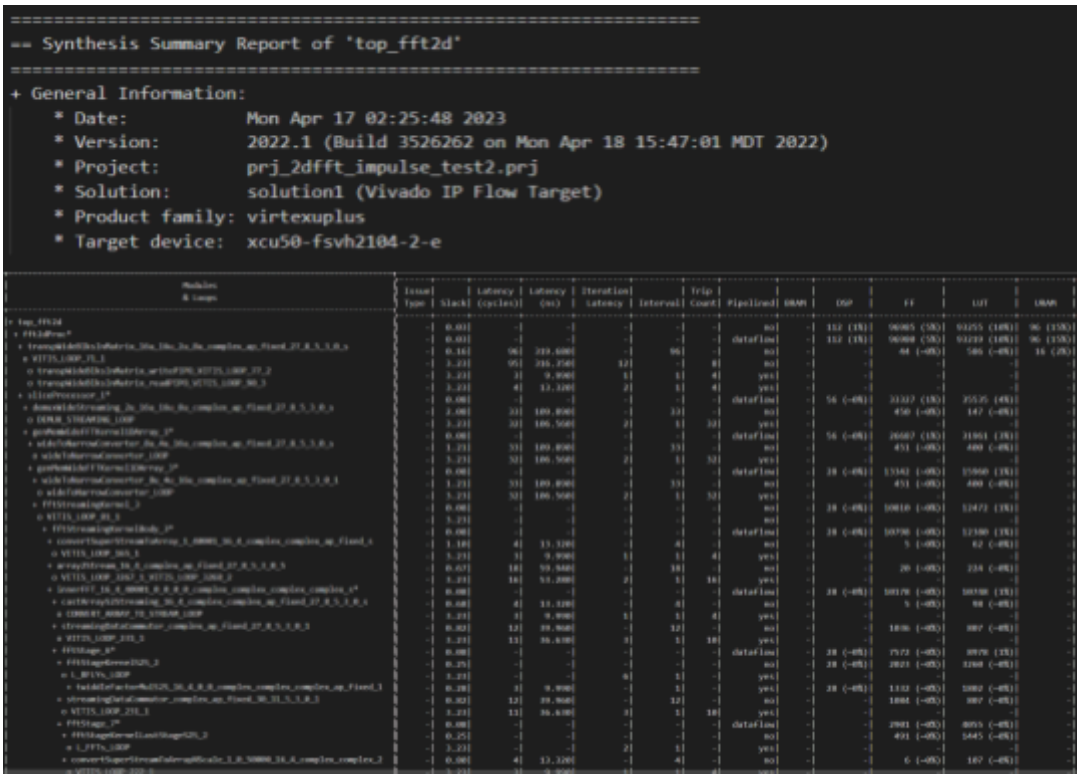
```
int main(int argc, char** argv) {
    // 2d input matrix
    T_elemType l_inMat[k_fftKernelSize][k_fftKernelSize];
    T_outType l_outMat[k_fftKernelSize][k_fftKernelSize];
    T_outType l_data2d_golden[k_fftKernelSize][k_fftKernelSize];

    // init input matrix with complex impulse
    for (int r = 0; r < k_fftKernelSize; ++r) {
        for (int c = 0; c < k_fftKernelSize; ++c) {
            if (r == 0 && c == 0)
                l_inMat[r][c] = T_compleFloat(1, 1);
            else
                l_inMat[r][c] = T_compleFloat(0, 0);
        }
    }
    // Wide Stream for reading and streaming a 2-d matrix
    MemWideIFStreamTypeIn l_matToStream("matrixToStreaming");
    MemWideIFStreamTypeOut fftOutputStream("fftOutputStream");
    // Pass same data stream multiple times to measure the II correctly
    for (int runs = 0; runs < 5; ++runs) {
        stream2DMatrix<k_fftKernelSize, k_fftKernelSize, k_memWidth, T_elemType, MemWideIFTypeIn>(l_inMat,
                                                                                                  l_matToStream);
        top_fft2d(l_matToStream, fftOutputStream);

        printMatStream<k_fftKernelSize, k_fftKernelSize, k_memWidth, MemWideIFTypeOut>(
            fftOutputStream, "2D FFT Output Natural Order...");
        streamToMatrix<k_fftKernelSize, k_fftKernelSize, k_memWidth, T_outType>(fftOutputStream, l_outMat);
    } // runs loop

    T_outType golden_result = T_elemType(1, 1);
    for (int r = 0; r < k_fftKernelSize; ++r) {
        for (int c = 0; c < k_fftKernelSize; ++c) {
            if (golden_result != l_outMat[r][c]) return 1;
        }
    }

    std::cout << "=================================================================" << std::endl;
    std::cout << "---------------------Impulse test Passed Successfully." << std::endl;
    std::cout << "=================================================================" << std::endl;
    return 0;
}
#endif

C:\Users\USER\AppData\Local\Temp\Mxt221\RemoteFiles\6721 UNIX | C/C++ | 73 lines | Row #1 | Col #1
```

Similar FFT structure definition for 2D FFT can be found in top_2d_fft_test.hpp:



```
struct FFTParams : ssr_fft_default_params {
    static const int N = k_fftKernelSize;
    static const int R = k_fftKernelRadix;
    static const scaling_mode_enum scaling_mode = SSR_FFT_NO_SCALING;

    static const transform_direction_enum transform_direction = FORWARD_TRANSFORM;
};
```

synthesis:

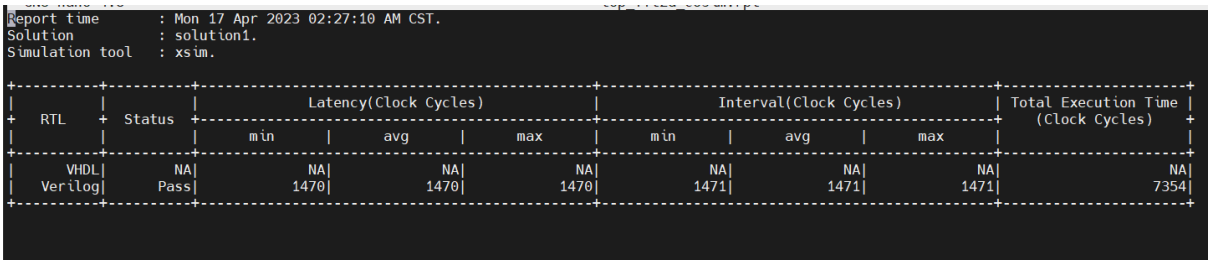


```
================================================================
== Synthesis Summary Report of 'top_fft2d'
================================================================
+ General Information:
    * Date:           Mon Apr 17 02:25:48 2023
    * Version:        2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)
    * Project:        prj_2dfft_impulse_test2.prj
    * Solution:       solution1 (Vivado IP Flow Target)
    * Product family: virtexuplus
    * Target device:  xcu50-fsvh2104-2-e
```

c-sim:

```
242 ================================================================================
243 ============================================================
244 ============================================================
245 ============================================================
246 2D FFT Output Natural Order...
247 Stream Size :32
248 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
249 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
250 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
251 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
252 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
253 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
254 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
255 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
256 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
257 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
258 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
259 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
260 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
261 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
262 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
263 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
264 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
265 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
266 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
267 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
268 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
269 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
270 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
271 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
272 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
273 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
274 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
275 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
276 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
277 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
278 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
279 (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
280 ============================================================
281 ============================================================
282 ============================================================
283 ================================================================
284 ----------------------Impulse test Passed Successfully.
285 ================================================================
286 INFO: [SIM 1] CSim done with 0 errors.
287 INFO: [SIM 3] *************** CSIM finish ***************
288
```

co-sim:



```
Report time       : Mon 17 Apr 2023 02:27:10 AM CST.
Solution          : solution1.
Simulation tool   : xsim.

+----------+---------+---------------------------------------------+---------------------------------------------+---------------------+
|          |         |            Latency(Clock Cycles)            |            Interval(Clock Cycles)           | Total Execution Time |
+   RTL    +  Status +---------------------------------------------+---------------------------------------------+   (Clock Cycles)   +
|          |         |     min     |     avg     |     max     |     min     |     avg     |     max     |                     |
+----------+---------+---------------------------------------------+---------------------------------------------+---------------------+
|      VHDL|       NA|           NA|           NA|           NA|           NA|           NA|           NA|                   NA|
|   Verilog|     Pass|         1470|         1470|         1470|         1471|         1471|         1471|                 7354|
+----------+---------+---------------------------------------------+---------------------------------------------+---------------------+
```

2D float point FFT with test data in

~//LAB_C/Vitis_Libraries/dsp/L1/tests/common_2dfft/2dFFTVerificationData/:

The test data input file used in this experiment is fft2DStimulusIn_L256.verif, and the golden output file used for verification is fft2DGoldenOut_L256.verif:

```
256
0.13495389793197751227
-0.06406211656053811820
0.17622033855159391358
0.34712859840742421591
-0.64534976269170163388
0.74866804101593042997
1.00000000000000000000
-0.06146610742496064206
0.46772932549568185800
-0.65622790258932495178
-0.08903002169072719851
-0.65833159228754312142
-0.52225560628116507456
-0.44263471171097534818
-0.04976702465150872889
0.53144878195202394089
0.69318843359757431610
0.52544431495266175070
-0.57574563742951245793
-0.21034345833465675768
-0.05028152629100325621
0.40781838856447949482
0.39053009979088842973
0.67057669642186612524
0.38800927215363972822
0.36909247552335028297
-0.30190495517227572408
0.22676939551259081562
-0.05711416075249285684
0.23491045635489121723
-0.14717234329839321005
0.32152924811687993989
0.13495389793197751227
-0.06406211656053811820
0.17622033855159391358
0.34712859840742421591
-0.64534976269170163388
```

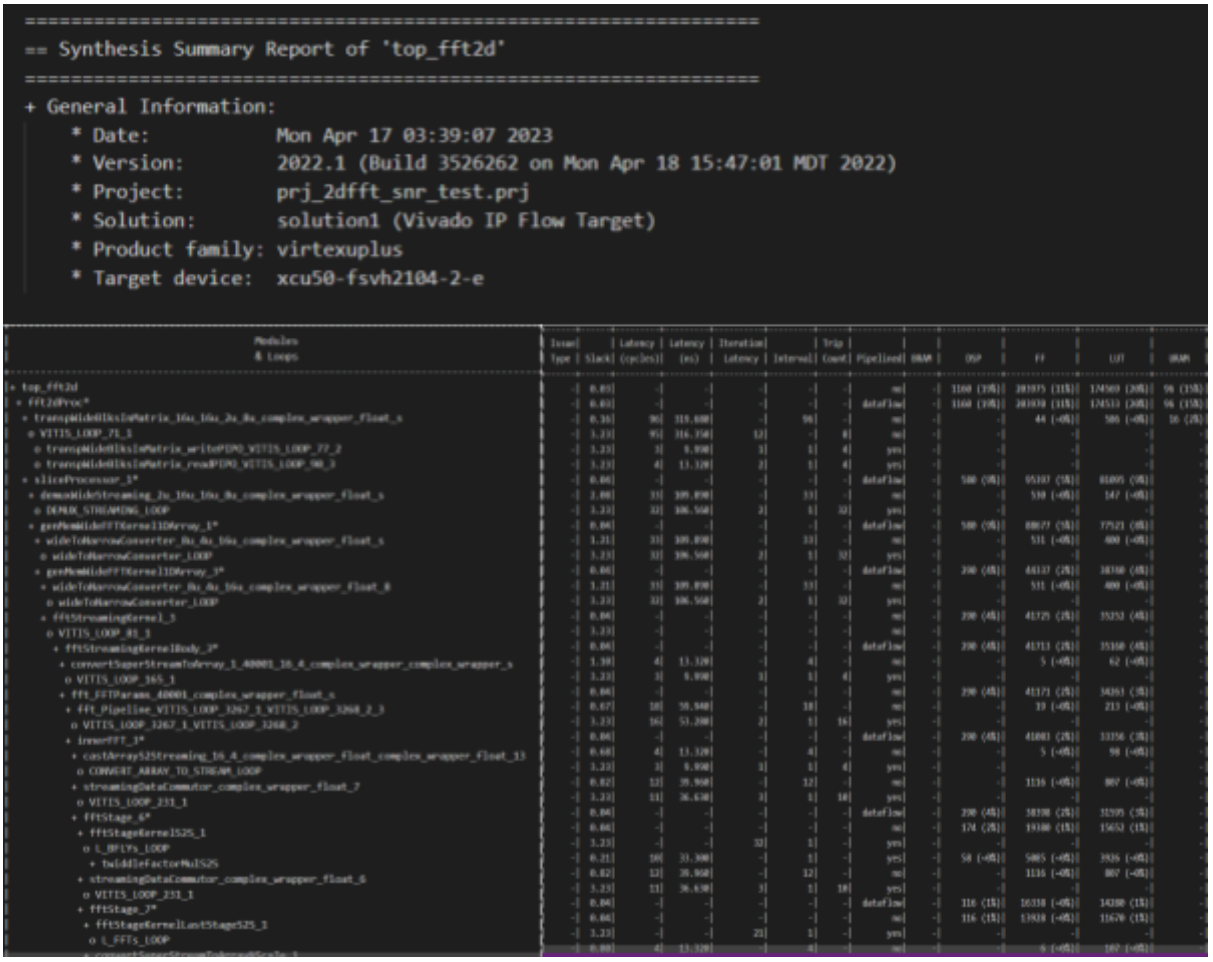
fig.12 a snippet of test data in fft2DStimulusIn_L256.verif

test program (red segment part loads the input file):

```
int main(int argc, char** argv) {
    // 2d input matrix
    T_elemType l_inMat[k_fftKernelSize][k_fftKernelSize];
    T_outType l_outMat[k_fftKernelSize][k_fftKernelSize];
    T_outType l_data2d_golden[k_fftKernelSize][k_fftKernelSize];

    ////////////////////////// File Paths for stimulus files/////////////////
    std::string root_path = "";
    std::string inputStimulusDataVerifFileName;
    std::stringstream strStream;
    ////////////////////////// Log Stream for writing output messages////////
    std::ofstream logStreamBB;
    std::string blockBoxFilename = root_path + "2d_fft_verification_log.log";
    logStreamBB.open(blockBoxFilename.c_str(), std::fstream::out);

    strStream << root_path << "fft2DStimulusIn_L" << k_fftKernelSize * k_fftKernelSize << ".verif";
    inputStimulusDataVerifFileName = strStream.str();
    std::cout << "The Stimulus file and constructed path: " << inputStimulusDataVerifFileName << "\n";

    std::stringstream strStream1;
    std::string goldenOutputDataVerifFileName;
    strStream1 << root_path << "fft2DGoldenOut_L" << k_fftKernelSize * k_fftKernelSize << ".verif";
    goldenOutputDataVerifFileName = strStream1.str();

    // Data Arrays to read stimulus data from file
    T_compleFloat* l_fileData = new T_compleFloat[k_fftKernelSize * k_fftKernelSize];

    // readComplexArrayFromFile<T_innerFloat>(logStreamBB, "din_file", inputStimulusDataVerifFileName, &l_inMat[0][0],
    // k_fftKernelSize*k_fftKernelSize);
    readComplexArrayFromFile<T_innerFloat>(logStreamBB, "din_file", inputStimulusDataVerifFileName, l_fileData,
                                           k_fftKernelSize * k_fftKernelSize);

    // init input matrix with stimuls, this loop will also take care of casting if types doesnt match
    int k = 0;
    for (int r = 0; r < k_fftKernelSize; ++r) {
        for (int c = 0; c < k_fftKernelSize; ++c) {
            l_inMat[r][c] = l_fileData[k];
            k++;
        }
    }
    delete l_fileData;
```

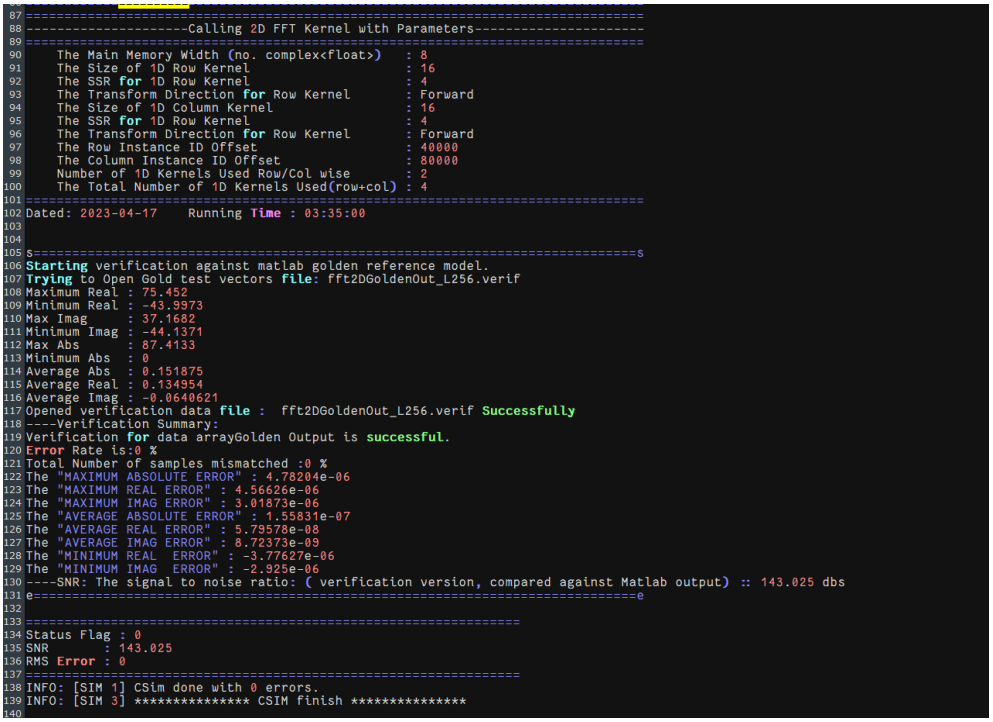
```
    // Wide Stream for reading and streaming a 2-d matrix
    MemWideIFStreamTypeIn l_matToStream("matrixToStreaming");
    MemWideIFStreamTypeOut fftOutputStream("fftOutputStream");
    // Pass same data stream multiple times to measure the II correctly
    for (int runs = 0; runs < 5; ++runs) {
        stream2DMatrix<k_fftKernelSize, k_fftKernelSize, k_memWidth, T_elemType, MemWideIFTypeIn>(l_inMat,
                                                                                                  l_matToStream);
        top_fft2d(l_matToStream, fftOutputStream);

        // printMatStream<k_fftKernelSize, k_fftKernelSize, k_memWidth>(fftOutputStream, "2D FFT Output Natural
        // Order...");
        streamToMatrix<k_fftKernelSize, k_fftKernelSize, k_memWidth, T_elemType, MemWideIFTypeOut>(fftOutputStream,
                                                                                                   l_outMat);
    } // runs loop
    l_fileData = new T_compleFloat[k_fftKernelSize * k_fftKernelSize];
    k = 0;
    for (int r = 0; r < k_fftKernelSize; ++r) {
        for (int c = 0; c < k_fftKernelSize; ++c) {
            l_fileData[k] = l_outMat[r][c];
            k++;
        }
    }
    std::ofstream logStream;
    VerificationResults verif_res_cpp_float_vs_octave;
    verif_res_cpp_float_vs_octave = verifyArrayOutput_with_snr<T_innerFloat>(
        logStream, "Golden Output", goldenOutputDataVerifFileName, l_fileData, 10, 5,
        k_fftKernelSize * k_fftKernelSize); // error given in %

    std::cout << "============================================================" << std::endl;
    std::cout << verif_res_cpp_float_vs_octave << std::endl;
    std::cout << "============================================================" << std::endl;
    return verif_res_cpp_float_vs_octave.m_statusFlag;
}
#endif
```

synthesis summary:



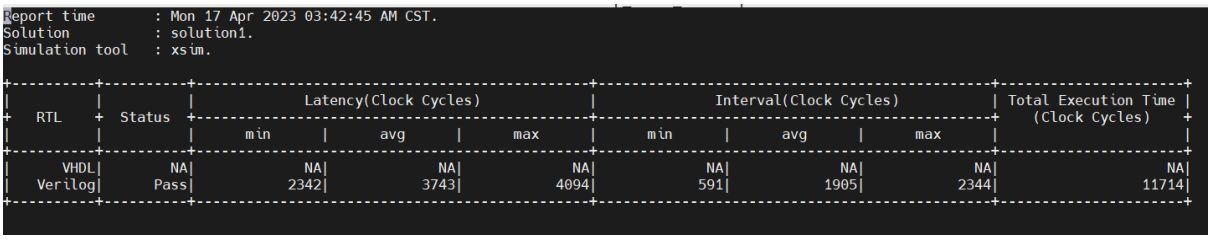
```
================================================================
== Synthesis Summary Report of 'top_fft2d'
================================================================
+ General Information:
    * Date:           Mon Apr 17 03:39:07 2023
    * Version:        2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)
    * Project:        prj_2dfft_snr_test.prj
    * Solution:       solution1 (Vivado IP Flow Target)
    * Product family: virtexuplus
    * Target device:  xcu50-fsvh2104-2-e
```

c-sim:



```
==========
----------Calling 2D FFT Kernel with Parameters----------
==========
    The Main Memory Width (no. complex<float>)   : 8
    The Size of 1D Row Kernel                    : 16
    The SSR for 1D Row Kernel                    : 4
    The Transform Direction for Row Kernel       : Forward
    The Size of 1D Column Kernel                 : 16
    The SSR for 1D Row Kernel                    : 4
    The Transform Direction for Row Kernel       : Forward
    The Row Instance ID Offset                   : 40000
    The Column Instance ID Offset                : 80000
    Number of 1D Kernels Used Row/Col wise       : 2
    The Total Number of 1D Kernels Used(row+col) : 4
==========
Dated: 2023-04-17    Running Time : 03:35:00


s==========s
Starting verification against matlab golden reference model.
Trying to Open Gold test vectors file: fft2DGoldenOut_L256.verif
Maximum Real : 75.452
Minimum Real : -43.9973
Max Imag     : 37.1682
Minimum Imag : -44.1371
Max Abs      : 87.4133
Minimum Abs  : 0
Average Abs  : 0.151875
Average Real : 0.134954
Average Imag : -0.0640621
Opened verification data file :  fft2DGoldenOut_L256.verif Successfully
----Verification Summary:
Verification for data arrayGolden Output is successful.
Error Rate is:0 %
Total Number of samples mismatched :0 %
The "MAXIMUM ABSOLUTE ERROR" : 4.78204e-06
The "MAXIMUM REAL ERROR" : 4.56626e-06
The "MAXIMUM IMAG ERROR" : 3.01873e-06
The "AVERAGE ABSOLUTE ERROR" : 1.55831e-07
The "AVERAGE REAL ERROR" : 5.79578e-08
The "AVERAGE IMAG ERROR" : 8.72373e-09
The "MINIMUM REAL  ERROR" : -3.77627e-06
The "MINIMUM IMAG  ERROR" : -2.925e-06
----SNR: The signal to noise ratio: ( verification version, compared against Matlab output) :: 143.025 dbs
e==========e

==========
Status Flag : 0
SNR         : 143.025
RMS Error : 0
==========
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

co-sim:



```
Report time       : Mon 17 Apr 2023 03:42:45 AM CST.
Solution          : solution1.
Simulation tool   : xsim.

+----------+---------+-----------------------------------------+-----------------------------------------+--------------------+
|          |         |            Latency(Clock Cycles)        |            Interval(Clock Cycles)       | Total Execution Time |
+   RTL    +  Status +-----------------------------------------+-----------------------------------------+   (Clock Cycles)    +
|          |         |     min     |     avg     |     max     |     min     |     avg     |     max     |                    |
+----------+---------+-------------+-------------+-------------+-------------+-------------+-------------+--------------------+
|     VHDL|       NA|           NA|           NA|           NA|           NA|           NA|           NA|                  NA|
|   Verilog|     Pass|         2342|         3743|         4094|          591|         1905|         2344|               11714|
+----------+---------+-------------+-------------+-------------+-------------+-------------+-------------+--------------------+
```

L2 of SSR FFT

As the L3 function for DSP library hasn't been included in the project, our principal goal in the second part is to identify L2 modules, use the kernel's appropriate sources to build XCLBIN file to run 1) software emulation, 2) hardware emulation, and 3) build and test on hardware. The kernel for this experiment will be SSR FFT. For this part, we will only cover the openCL kernel as instructed. Following the instruction instruction from the readme file in the library, the files which will be needed to run the OpenCL kernel application project can be found inside the directory "test" under L2.

Just like L1, SSR FFT's L2 consists of 1D and 2D FFT, each in their turn consists of fixed point and floating point data type. Due to their similarity when implementing and constant error from the platform for floating point, we've decided to work only on 1D's fixed point.

Create and Run the project

1. Open Vitis IDE
2. Select create application project, in Platform windows, choose u50 acceleration card.
3. Import the sources needed for both the kernel and the host, these files can be found in 1) L2/tests/hw/1dfft/fixed/host and kernel, 2) L2/include/hw/vitis_fft/fixed and 3) L1/include/hw/vitis_fft/fixed.
4. To run the emulations and Hardware test, do as follow for each build configuration:
    a. build : 1- kernel, 2- hw_link, 3-host, 4-system.
    b. run configuration with setting: **Xilinx xilinx_u50_gen3x16_xdma_5_202210_1 -xclbin ./binary_container_1.xclbin**, with host trace and openCL trace checked.
    c. Check run_summary.xrt to visualize and analyze the timeline-trace with Vitis Analyzer.

Host's main code snippet

The following figures show some important part of the host, in Fig 2.1, it shows the declaration and initialization of memory locations that will hold input and output data. Fig 2.2 shows the declaration and how the memory mapping between DDR and the

host's virtual memory is done, using OpenCL's buffer(type of memory object used to store data in the global memory space). Fig 2.3 shows the code line for data transfer between the kernel and global memory.

```
ap_uint<512>* inData = aligned_alloc<ap_uint<512> >(FFT_LEN * nffts / SSR);
ap_uint<512>* outData = aligned_alloc<ap_uint<512> >(FFT_LEN * nffts / SSR);
// impulse as input
for (int n = 0; n < nffts; ++n) {
    for (int t = 0; t < FFT_LEN / SSR; ++t) {
        if (t == 0)
            inData[n * FFT_LEN / SSR + t] = 1;
        else
            inData[n * FFT_LEN / SSR + t] = 0;
    }
}
std::cout << "Host buffer has been allocated and set.\n";
```

Fig 2.1.

```
cl::Kernel kernel(program, "fft1DKernel", &err);
logger.logCreateKernel(err);
std::cout << "Kernel has been created.\n";

cl_mem_ext_ptr_t mext_in, mext_out;
mext_in = {XCL_MEM_DDR_BANK0, inData, 0};
mext_out = {XCL_MEM_DDR_BANK0, outData, 0};

cl::Buffer in_buff;
in_buff = cl::Buffer(context, CL_MEM_EXT_PTR_XILINX | CL_MEM_USE_HOST_PTR | CL_MEM_READ_ONLY,
                     (size_t)(sizeof(ap_uint<512>) * (FFT_LEN * nffts / SSR)), &mext_in);
cl::Buffer out_buff;
out_buff = cl::Buffer(context, CL_MEM_EXT_PTR_XILINX | CL_MEM_USE_HOST_PTR | CL_MEM_WRITE_ONLY,
                      (size_t)(sizeof(ap_uint<512>) * (FFT_LEN * nffts / SSR)), &mext_out);
std::cout << "DDR buffers have been mapped/copy-and-mapped\n";
```

fig 2.2.

```
// write data to DDR
std::vector<cl::Memory> ib;
ib.push_back(in_buff);
q.enqueueMigrateMemObjects(ib, 0, nullptr, &write_events[0][0]);

// set args and enqueue kernel
int j = 0;
kernel.setArg(j++, in_buff);
kernel.setArg(j++, out_buff);
kernel.setArg(j++, nffts);
q.enqueueTask(kernel, &write_events[0], &kernel_events[0][0]);

// read data from DDR
std::vector<cl::Memory> ob;
ob.push_back(out_buff);
q.enqueueMigrateMemObjects(ob, CL_MIGRATE_MEM_OBJECT_HOST, &kernel_events[0], &read_events[0][0]);

// wait all to finish
q.flush();
q.finish();
gettimeofday(&end_time, 0);
```

Fig 2.3.

Software emulation

Allows us to test and debug on a host before developing on FPGA. Below, Fig 2.4 shows software emulation's profile summary while 2.5 and 2.6 show the timeline trace(write, kernel execution, read). The kernel takes 59807.200 ms to execute.

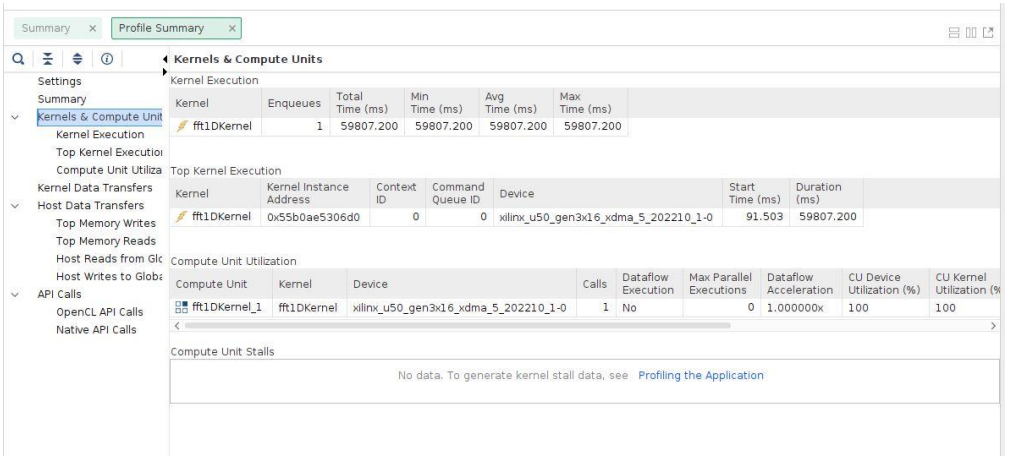

Summary
Profile Summary
Settings
Summary
Kernels & Compute Unit
Kernel Execution
Top Kernel Executio
Compute Unit Utiliza
Kernel Data Transfers
Host Data Transfers
Top Memory Writes
Top Memory Reads
Host Reads from Glo
Host Writes to Globa
API Calls
OpenCL API Calls
Native API Calls
Kernels & Compute Units
Kernel Execution
Kernel | Enqueues | Total Time (ms) | Min Time (ms) | Avg Time (ms) | Max Time (ms)
fft1DKernel | 1 | 59807.200 | 59807.200 | 59807.200 | 59807.200
Top Kernel Execution
Kernel | Kernel Instance Address | Context ID | Command Queue ID | Device | Start Time (ms) | Duration (ms)
fft1DKernel | 0x55b0ae5306d0 | 0 | 0 | xilinx_u50_gen3x16_xdma_5_202210_1-0 | 91.503 | 59807.200
Compute Unit Utilization
Compute Unit | Kernel | Device | Calls | Dataflow Execution | Max Parallel Executions | Dataflow Acceleration | CU Device Utilization (%) | CU Kernel Utilization (%
fft1DKernel_1 | fft1DKernel | xilinx_u50_gen3x16_xdma_5_202210_1-0 | 1 | No | 0 | 1.000000x | 100 | 100
Compute Unit Stalls
No data. To generate kernel stall data, see Profiling the Application

Fig 2.4

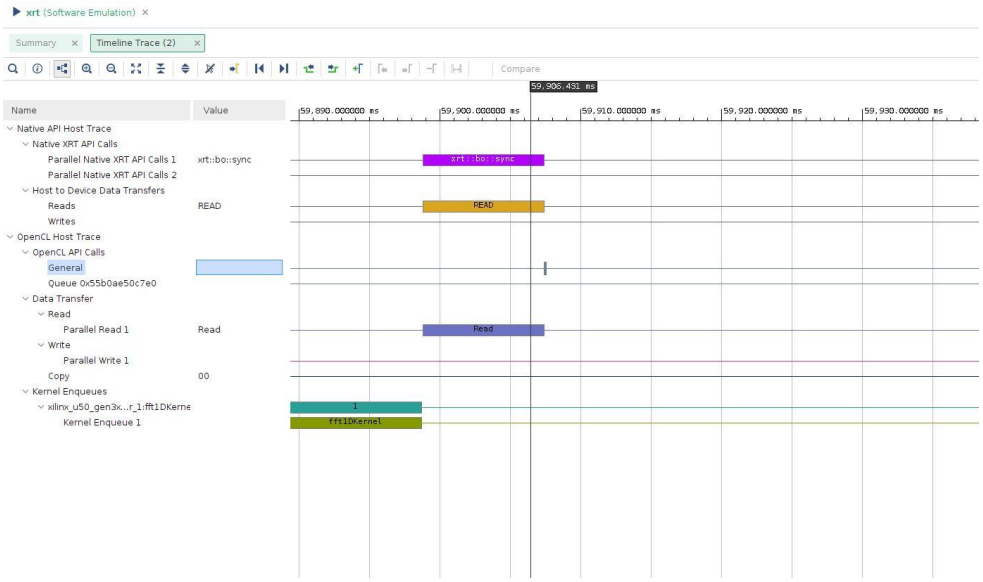

xrt (Software Emulation)
Summary
Timeline Trace (2)
Compare
59,906.431 ms
Name
Value
59,890.000000 ms
59,900.000000 ms
59,910.000000 ms
59,920.000000 ms
59,930.000000 ms
Native API Host Trace
Native XRT API Calls
Parallel Native XRT API Calls 1
xrt::bo::sync
Parallel Native XRT API Calls 2
Host to Device Data Transfers
Reads
READ
Writes
OpenCL Host Trace
OpenCL API Calls
General
Queue 0x55b0ae50c7e0
Data Transfer
Read
Parallel Read 1
Read
Write
Parallel Write 1
Copy
00
Kernel Enqueues
xilinx_u50_gen3x...r_1:fft1DKerne
Kernel Enqueue 1
1
fft1DKernel

Fig 2.5.

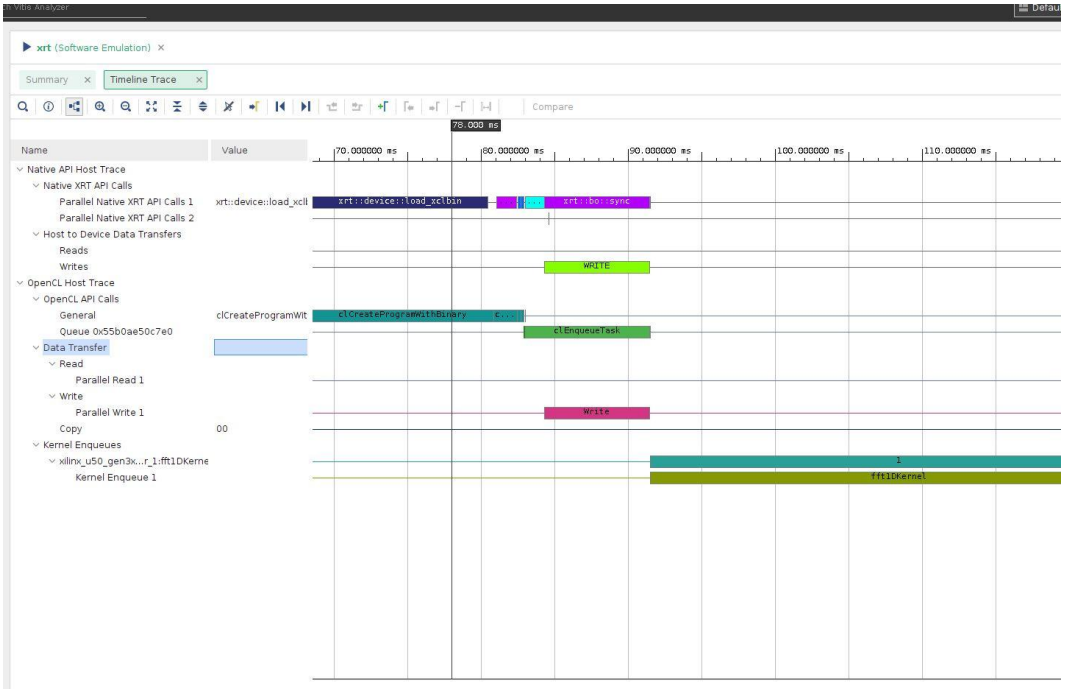


Fig 2.6

Hardware Emulation

Because we were dealing with a large set of data 1024 points FFT, the hardware emulation took longer than the time we had in our hands. Fig 2.7shows the console when executing hardware emulation.

```
Host buffer has been allocated and set.
Found Platform
Platform Name: Xilinx
Info: Context created
Info: Command queue created
Selected Device xilinx_u50_gen3x16_xdma_5_202210_1
INFO: Importing /mnt/HLSNAS/02.tObflU/LabC/DSP_1dfft_system/Emulation-HW/binary_container_1.xclbin
Loading: '/mnt/HLSNAS/02.tObflU/LabC/DSP_1dfft_system/Emulation-HW/binary_container_1.xclbin'
INFO: [HW-EMU 01] Hardware emulation runs simulation underneath. Using a large data set will result in long simulation times. It is reco
configuring dataflow mode with ert polling
scheduler config ert(1), dataflow(1), slots(16), cudma(0), cuisr(0), cdma(0), cus(1)
INFO: [HW-EMU 07-0] Please refer the path "/mnt/HLSNAS/02.tObflU/LabC/DSP_1dfft/Emulation-HW/.run/29829/hw_em/device0/binary_0/behav_wav
Info: Program created
Info: Kernel created
Kernel has been created.
DDR buffers have been mapped/copy-and-mapped
INFO::[ Vitis-EM 22 ] [Time elapsed: 4 minute(s) 52 seconds, Emulation time: 0.889473 ms]
Data transfer between kernel(s) and global memory(s)
fft1DKernel_1:m_axi_gmem0-HBM[0]          RD = 20.000 KB              WR = 0.000 KB

INFO::[ Vitis-EM 22 ] [Time elapsed: 9 minute(s) 52 seconds, Emulation time: 1.64863 ms]
Data transfer between kernel(s) and global memory(s)
fft1DKernel_1:m_axi_gmem0-HBM[0]          RD = 20.000 KB              WR = 0.000 KB

INFO::[ Vitis-EM 22 ] [Time elapsed: 14 minute(s) 52 seconds, Emulation time: 2.46379 ms]
Data transfer between kernel(s) and global memory(s)
fft1DKernel_1:m_axi_gmem0-HBM[0]          RD = 20.000 KB              WR = 0.000 KB

INFO::[ Vitis-EM 22 ] [Time elapsed: 19 minute(s) 53 seconds, Emulation time: 3.75205 ms]
Data transfer between kernel(s) and global memory(s)
fft1DKernel_1:m_axi_gmem0-HBM[0]          RD = 20.000 KB              WR = 0.000 KB

INFO::[ Vitis-EM 22 ] [Time elapsed: 24 minute(s) 53 seconds, Emulation time: 5.09732 ms]
Data transfer between kernel(s) and global memory(s)
fft1DKernel_1:m_axi_gmem0-HBM[0]          RD = 20.000 KB              WR = 0.000 KB
```

Fig 2.7

Running fir_129t_sym, an L2 example:

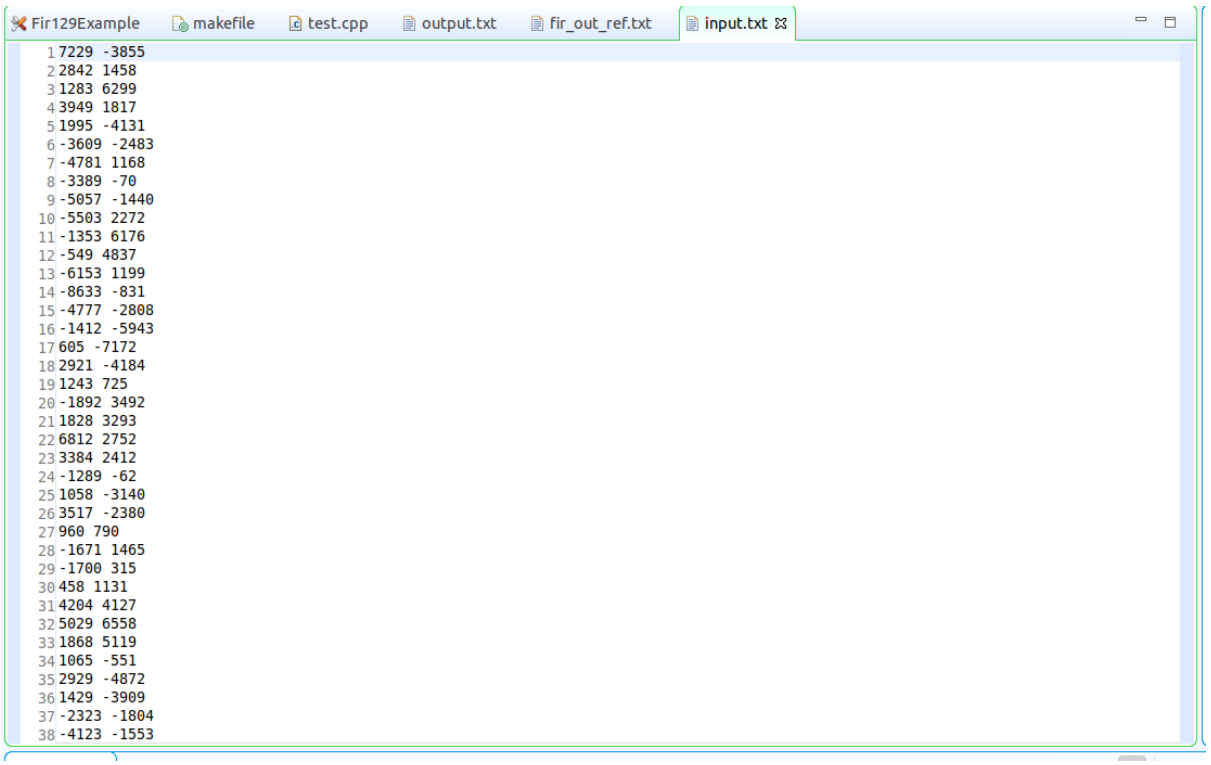
test program & test data:

Fir129Example | makefile | test.cpp | output.txt | fir_out_ref.txt | input.txt

```
/*
 * Copyright 2021 Xilinx, Inc.
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
#include "fir_graph.h"


simulation::platform<1,1> platform("data/input.txt", "data/output.txt");
FirKernel filter ;

connect<> net0(platform.src[0], filter.in);
connect<> net1(filter.out, platform.sink[0]);

int main(void) {
filter.init() ;
filter.run(4) ;
filter.end() ;
return 0 ;
}

```

Fir129Example | makefile | test.cpp | output.txt | fir_out_ref.txt | input.txt | fir_graph.h

```
 * limitations under the License.
 */
#pragma once

#include "system_settings.h"

using namespace adf;

class FirKernel: public graph
{
private:
    // FIR coefficients
    std::vector<int16> m_taps = std::vector<int16>{-1, -3, 3, -1, -3,
        6, -1, -7, 9, -1, -12, 14, 1, -20, 19, 5, -31, 26, 12, -45, 32, 23, -63,
        37, 40, -86, 40, 64, -113, 39, 96, -145, 33, 139, -180, 17, 195, -218, -9,
        266, -258, -53, 357, -299, -118, 472, -339, -215, 620, -376, -360, 822,
        -409, -585, 1118, -437, -973, 1625, -458, -1801, 2810, -470, -5012, 10783,
        25067};

    //FIR Graph class
    xf::dsp::aie::fir::sr_sym::fir_sr_sym_graph<DATA_TYPE, COEFF_TYPE, FIR_LEN, SHIFT,
    ROUND_MODE, WINDOW_SIZE> firGraph;

public:
    port<input> in;
    port<output> out;

    // Constructor - with FIR graph class initialization
    FirKernel():firGraph(m_taps) {
        // Make connections
        // Size of window in Bytes.
        // Margin gets automatically added within the FIR graph class.
        // Margin equals to FIR length rounded up to nearest multiple of 32 Bytes.
        connect<>(in, firGraph.in[0]);
        connect<>(firGraph.out[0], out);
    };
};

```


Fir129Example
makefile
test.cpp
output.txt
fir_out_ref.txt
input.txt
7229 -3855
2842 1458
1283 6299
3949 1817
1995 -4131
-3609 -2483
-4781 1168
-3389 -70
-5057 -1440
-5503 2272
-1353 6176
-549 4837
-6153 1199
-8633 -831
-4777 -2808
-1412 -5943
605 -7172
2921 -4184
1243 725
-1892 3492
1828 3293
6812 2752
3384 2412
-1289 -62
1058 -3140
3517 -2380
960 790
-1671 1465
-1700 315
458 1131
4204 4127
5029 6558
1868 5119
1065 -551
2929 -4872
1429 -3909
-2323 -1804
-4123 -1553

include file:

fir_sr_sym.cpp
fir_sr_asym.cpp
widget_api_cast.cpp

L2:
fir_graph_utils.hpp
fir_sr_sym_graph.hpp
fir_sr_asym_graph.hpp
graph_utils.hpp

L1:
fir_common_traits.hpp
fir_params_default.hpp
fir_sr_asym_traits.hpp
fir_sr_asym_utils.hpp
fir_sr_asym.hpp
fir_sr_sym_traits.hpp
fir_sr_sym_utils.hpp
fir_sr_sym.hpp
fir_utils.hpp
widget_api_cast_traits.hpp
widget_api_cast_utils.hpp
widget_api_cast.hpp
kernel_api_utils.hpp
kernel_broadcast.hpp
kernel_coeff_reload.hpp

RUN AIE EMULATOR and compare the generated output file with reference output:

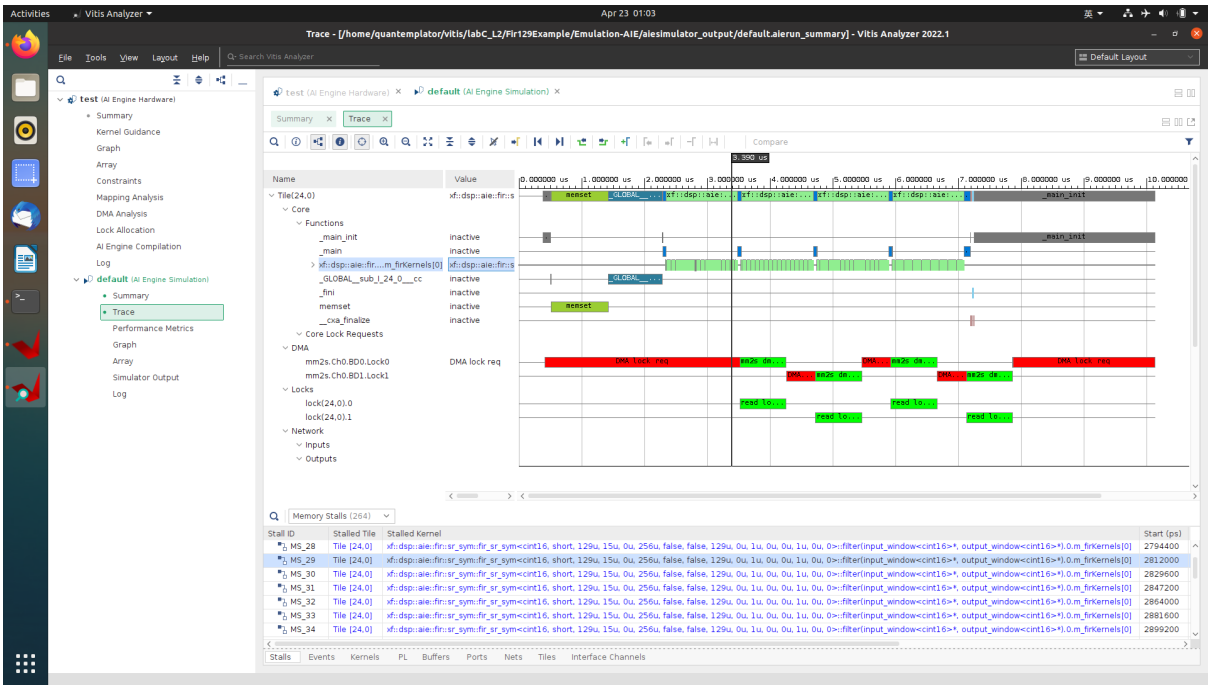
Fir129Example | makefile | test.cpp | output.txt | fir_out_ref.txt

```
-1 0
-1 0
0 -1
-1 -1
-2 0
1 -1
0 -1
-2 1
2 -1
1 -2
-4 1
2 -2
2 -3
-5 3
2 -1
4 -4
-8 5
1 0
6 -6
-10 5
1 0
10 -9
-13 6
0 2
15 -12
-16 8
-2 5
21 -17
-20 9
-7 9
28 -21
-24 9
-13 14
38 -27
-27 9
-21 22
50 -34
-28 8
```

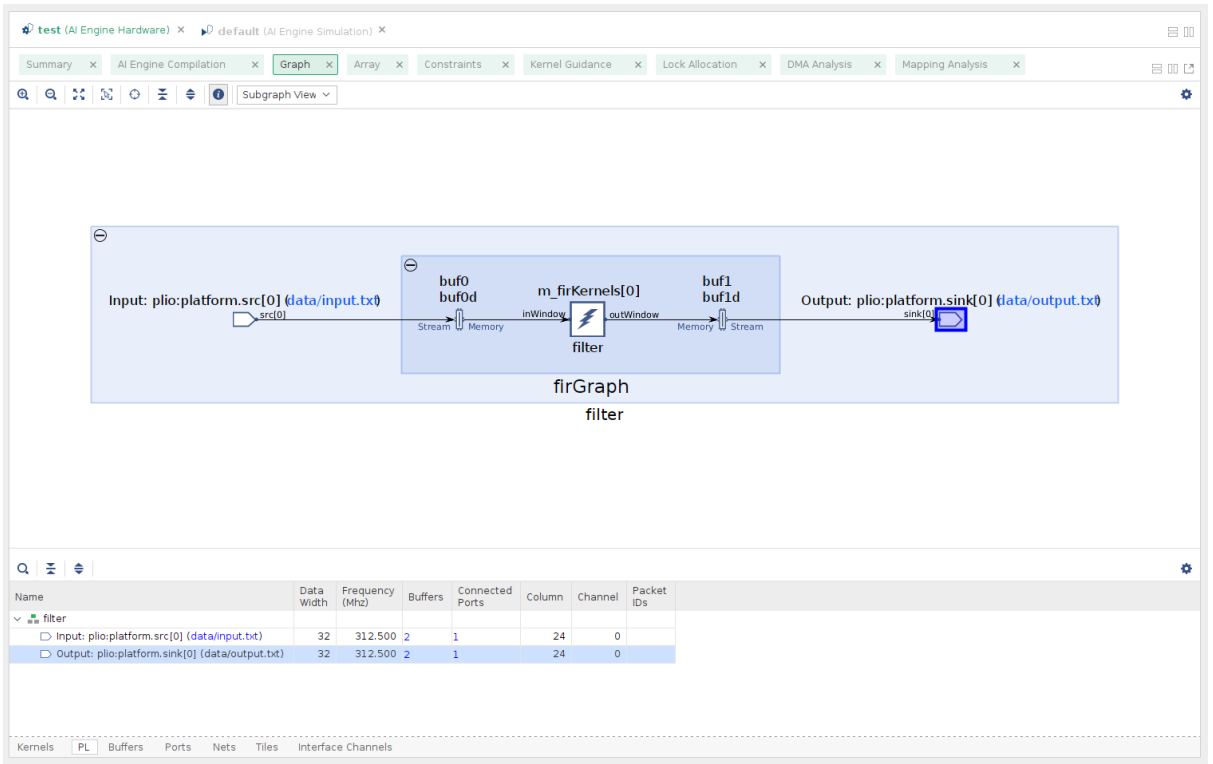
Fir129Example | makefile | test.cpp | output.txt | fir_out_ref.txt

```
T 3536 ns
-1 0
T 3539200 ps
-1 0
T 3542400 ps
0 -1
T 3545600 ps
-1 -1
T 3548800 ps
-2 0
T 3552 ns
1 -1
T 3555200 ps
0 -1
T 3558400 ps
-2 1
T 3561600 ps
2 -1
T 3564800 ps
1 -2
T 3568 ns
-4 1
T 3571200 ps
2 -2
T 3574400 ps
2 -3
T 3577600 ps
-5 3
T 3580800 ps
2 -1
T 3584 ns
4 -4
T 3587200 ps
-8 5
T 3590400 ps
1 0
T 3593600 ps
6 -6
```

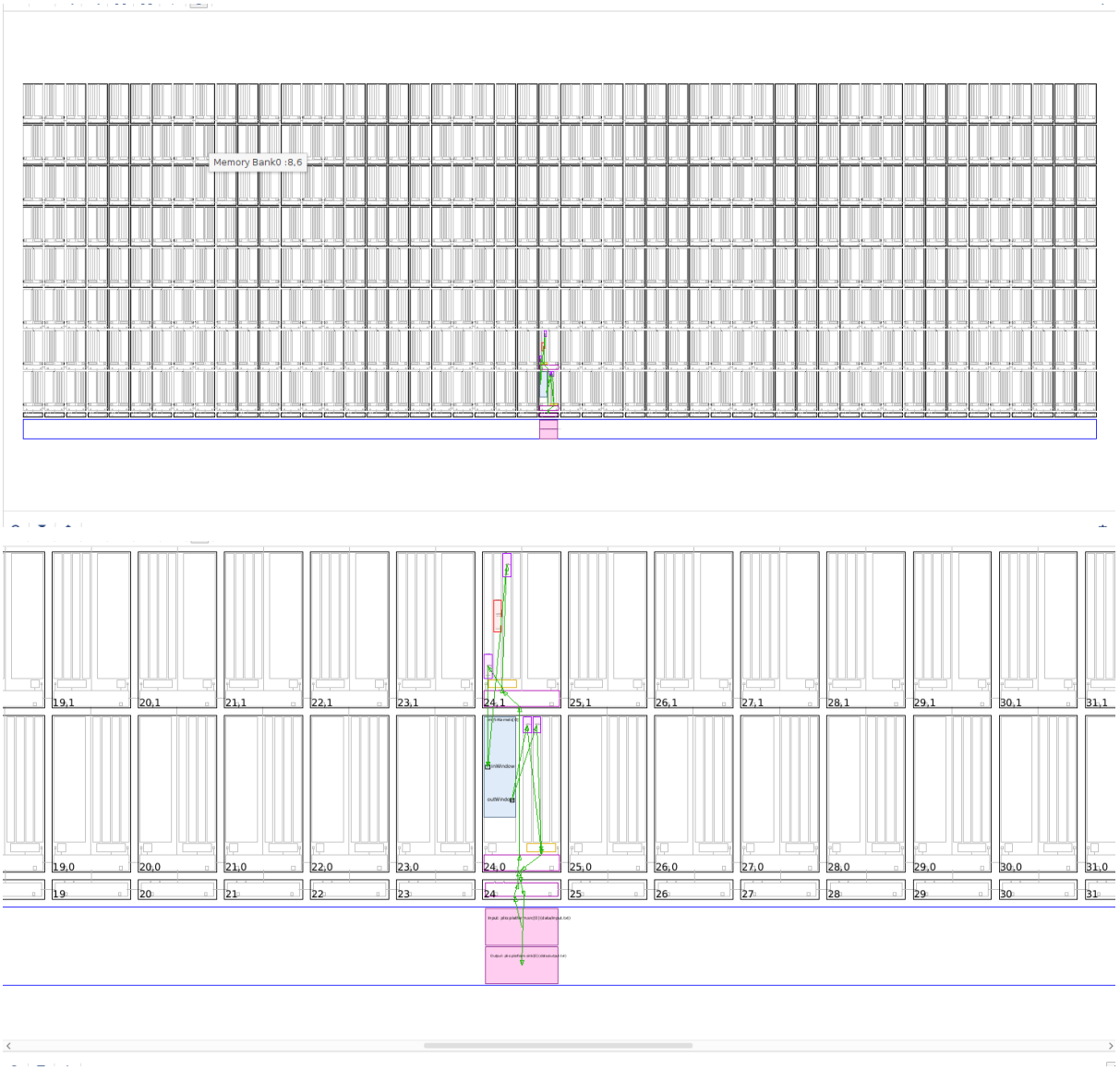
timeline trace:



graph:



aie array:



AIE reference documents:

1. https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2022_2/ug1079-ai-engine-kernel-coding.pdf
2. https://xilinx.github.io/Vitis_Libraries/dsp/2021.1/user_guide/L2/3-using-examples.html

Lesson and observation:

During the lab experiment, we were facing lots of problems, such as understanding the file dependencies, and the functions provided in the program files. To find file dependency we had to investigate the run_hls.tcl and the makefile, and there are some needed modifications in the makefile in order to run vitis hls by "make run" command. These things took us plenty of time, but we definitely ended up learning a lot as expected from the professor.

The other part we took some time was the AIE example, since we had to figure out what is AIE and understand the source codes, also the required header files and their dependencies.

github link:

邱崇喆: https://github.com/ccontemplator/AAHLS_LAB_C
石思宇: https://github.com/freud96/AAHL_LABC_DSP