Briefly introduction to the algorithm or overall system

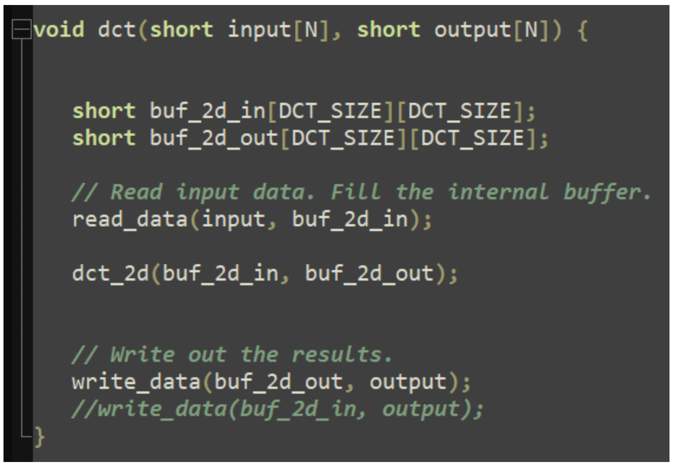
In this tutorial, we are going to learn how to use Vitis HLS to build, analyze, and optimize a hardware kernel. There are a few ways to do it, such as pipelining, using dual-port RAMs, array partitioning, and dataflow optimization.

The algorithm implemented in hardware is discreate cosine transform, DCT. It can process data into frequency domain, similar to Fourier transform. What makes them different is that DCT only uses real numbers. DCT is often used to process signals and images. Assume that we have a matrix with size $N \times N$, and the values are denoted by $f(x, y)$. The formula is shown below.

$$D(i,j) = \frac{1}{2\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)i\pi}{2N} \cos\frac{(2y+1)j\pi}{2N}$$

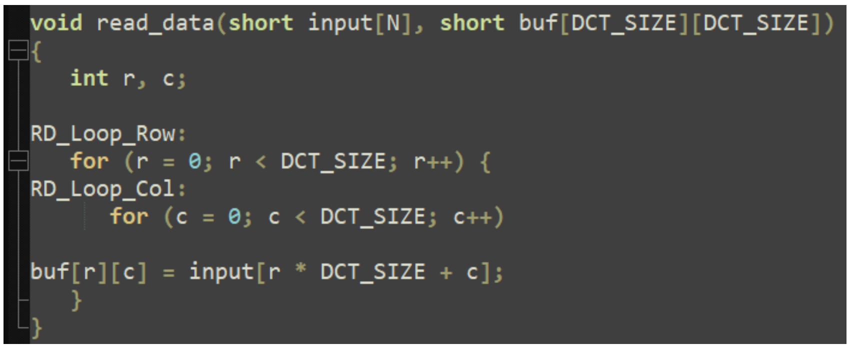
$$C(i) = \begin{cases} \frac{1}{\sqrt{2}}, & if\ i = 0 \\ 1, & otherwise \end{cases}$$

Explain the original code/system/pragmas and how you implement it



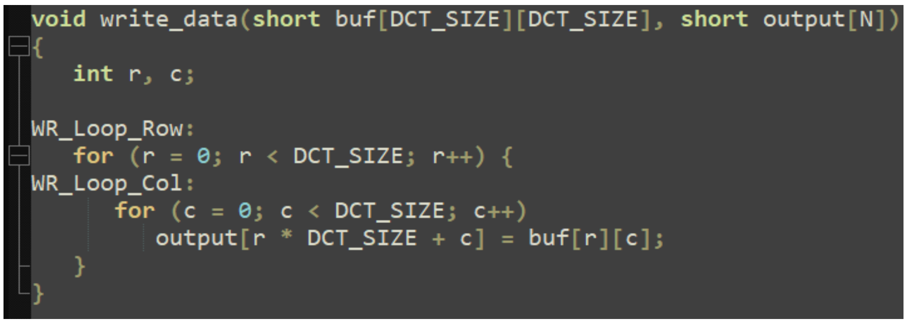
```
void dct(short input[N], short output[N]) {


    short buf_2d_in[DCT_SIZE][DCT_SIZE];
    short buf_2d_out[DCT_SIZE][DCT_SIZE];

    // Read input data. Fill the internal buffer.
    read_data(input, buf_2d_in);

    dct_2d(buf_2d_in, buf_2d_out);


    // Write out the results.
    write_data(buf_2d_out, output);
    //write_data(buf_2d_in, output);
}
```

This is the top function. It is obvious that the kernel will read data, process the data, and then write the data. Note that DCT_SIZE is specified as 8 in the design.



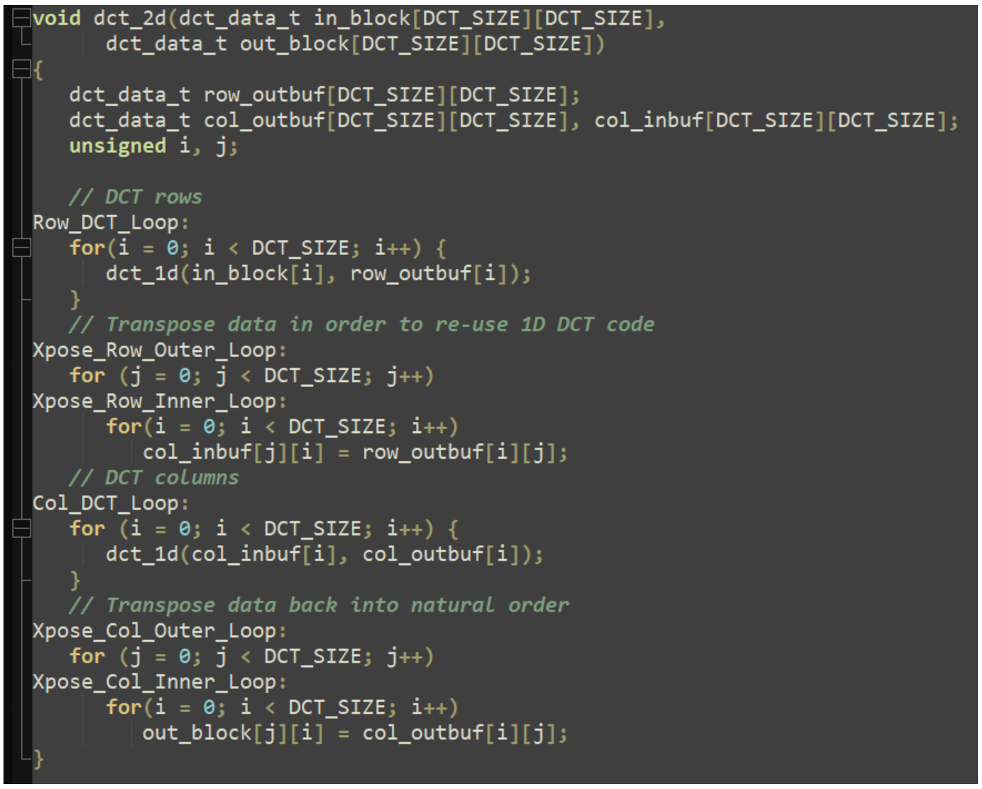
```
void read_data(short input[N], short buf[DCT_SIZE][DCT_SIZE])
{
    int r, c;

RD_Loop_Row:
    for (r = 0; r < DCT_SIZE; r++) {
RD_Loop_Col:
        for (c = 0; c < DCT_SIZE; c++)

buf[r][c] = input[r * DCT_SIZE + c];
    }
}
```

This is the function that reads data. We can see that it reads a 1d array into a 2d array.



```
void write_data(short buf[DCT_SIZE][DCT_SIZE], short output[N])
{
    int r, c;

WR_Loop_Row:
    for (r = 0; r < DCT_SIZE; r++) {
WR_Loop_Col:
        for (c = 0; c < DCT_SIZE; c++)
            output[r * DCT_SIZE + c] = buf[r][c];
    }
}
```

This is the function that writes data. We can see that it writes a 2d array into a 1d array.



```
void dct_2d(dct_data_t in_block[DCT_SIZE][DCT_SIZE],
        dct_data_t out_block[DCT_SIZE][DCT_SIZE])
{
    dct_data_t row_outbuf[DCT_SIZE][DCT_SIZE];
    dct_data_t col_outbuf[DCT_SIZE][DCT_SIZE], col_inbuf[DCT_SIZE][DCT_SIZE];
    unsigned i, j;

    // DCT rows
Row_DCT_Loop:
    for(i = 0; i < DCT_SIZE; i++) {
        dct_1d(in_block[i], row_outbuf[i]);
    }
    // Transpose data in order to re-use 1D DCT code
Xpose_Row_Outer_Loop:
    for (j = 0; j < DCT_SIZE; j++)
Xpose_Row_Inner_Loop:
        for(i = 0; i < DCT_SIZE; i++)
            col_inbuf[j][i] = row_outbuf[i][j];
    // DCT columns
Col_DCT_Loop:
    for (i = 0; i < DCT_SIZE; i++) {
        dct_1d(col_inbuf[i], col_outbuf[i]);
    }
    // Transpose data back into natural order
Xpose_Col_Outer_Loop:
    for (j = 0; j < DCT_SIZE; j++)
Xpose_Col_Inner_Loop:
        for(i = 0; i < DCT_SIZE; i++)
            out_block[j][i] = col_outbuf[i][j];
}
```

This is the core function of DCT algorithm. The function is further divided into 4 parts, processing row, transposing, processing column, and transposing again. In this way, they can reuse the same function. Here, I will show how they divide the formula into row and column steps with $N = 8$.

$$D(i,j) = \frac{1}{2\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)i\pi}{2N} \cos\frac{(2y+1)j\pi}{2N}$$

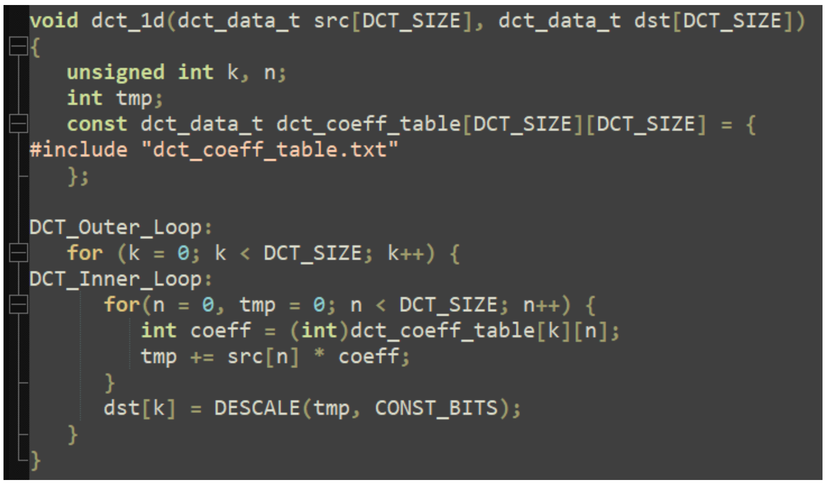
$$= \frac{1}{8} C(i)C(j) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\frac{(2x+1)i\pi}{16} \cos\frac{(2y+1)j\pi}{16}$$

$$= \frac{1}{2\sqrt{2}} C(i) \sum_{x=0}^{7} \left[\frac{1}{2\sqrt{2}} C(j) \sum_{y=0}^{7} f(x,y) \cos\frac{(2y+1)j\pi}{16} \right] \cos\frac{(2x+1)i\pi}{16}$$

$$Temp(x,j) = \frac{1}{2\sqrt{2}} C(j) \sum_{y=0}^{7} f(x,y) \cos\frac{(2y+1)j\pi}{16}$$

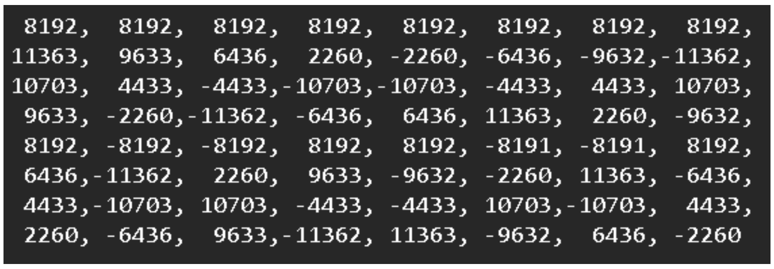
$$D(i,j) = \frac{1}{2\sqrt{2}} C(i) \sum_{x=0}^{7} Temp(x,j) \cos\frac{(2x+1)i\pi}{16}$$

We can see that the formula has the similar operation for sum over x and sum over y. They declare another function to do the operation row by row, and transpose is required.



```
void dct_1d(dct_data_t src[DCT_SIZE], dct_data_t dst[DCT_SIZE])
{
   unsigned int k, n;
   int tmp;
   const dct_data_t dct_coeff_table[DCT_SIZE][DCT_SIZE] = {
#include "dct_coeff_table.txt"
   };

DCT_Outer_Loop:
   for (k = 0; k < DCT_SIZE; k++) {
DCT_Inner_Loop:
      for(n = 0, tmp = 0; n < DCT_SIZE; n++) {
         int coeff = (int)dct_coeff_table[k][n];
         tmp += src[n] * coeff;
      }
      dst[k] = DESCALE(tmp, CONST_BITS);
   }
}
```

In this function, they input a row, and calculate the accumulated sum (inner loop) for each element (outer loop). Here, a 8×8 coefficient table, dct_coeff_table.txt, is used.

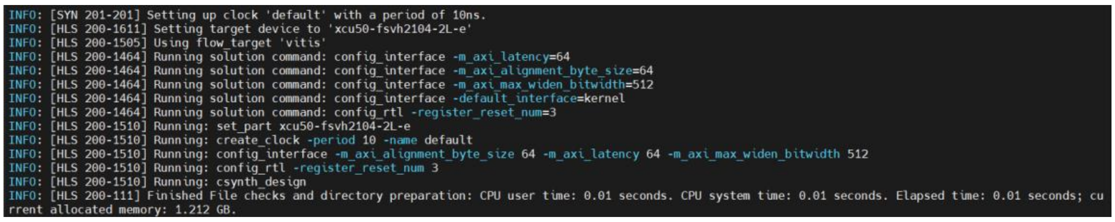


```
 8192,  8192,  8192,  8192,  8192,  8192,  8192,  8192,
11363,  9633,  6436,  2260, -2260, -6436, -9632,-11362,
10703,  4433, -4433,-10703,-10703, -4433,  4433, 10703,
 9633, -2260,-11362, -6436,  6436, 11363,  2260, -9632,
 8192, -8192, -8192,  8192,  8192, -8191, -8191,  8192,
 6436,-11362,  2260,  9633, -9632, -2260, 11363, -6436,
 4433,-10703, 10703, -4433, -4433, 10703,-10703,  4433,
 2260, -6436,  9633,-11362, 11363, -9632,  6436, -2260
```

In general, people convert floating numbers into integers before computing in hardware. Here, they implement the conversion by multiplying the floating numbers by 2^{15}. Therefore, the coefficients can be calculated in this way.

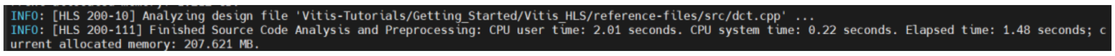
$$Coef(j,y) = \frac{1}{2\sqrt{2}} C(j) \cos\frac{(2y+1)j\pi}{16} \times 2^{15}$$

Examine the synthesis log to list what steps the tool takes during synthesis.



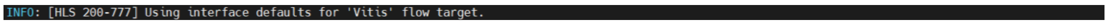
```
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-1611] Setting target device to 'xcu50-fsvh2104-2L-e'
INFO: [HLS 200-1505] Using flow_target 'vitis'
INFO: [HLS 200-1464] Running solution command: config_interface -m_axi_latency=64
INFO: [HLS 200-1464] Running solution command: config_interface -m_axi_alignment_byte_size=64
INFO: [HLS 200-1464] Running solution command: config_interface -m_axi_max_widen_bitwidth=512
INFO: [HLS 200-1464] Running solution command: config_interface -default_interface=kernel
INFO: [HLS 200-1464] Running solution command: config_rtl -register_reset_num=3
INFO: [HLS 200-1510] Running: set_part xcu50-fsvh2104-2L-e
INFO: [HLS 200-1510] Running: create_clock -period 10 -name default
INFO: [HLS 200-1510] Running: config_interface -m_axi_alignment_byte_size 64 -m_axi_latency 64 -m_axi_max_widen_bitwidth 512
INFO: [HLS 200-1510] Running: config_rtl -register_reset_num 3
INFO: [HLS 200-1510] Running: csynth_design
INFO: [HLS 200-111] Finished File checks and directory preparation: CPU user time: 0.01 seconds. CPU system time: 0.01 seconds. Elapsed time: 0.01 seconds; cu
rrent allocated memory: 1.212 GB.
```

Project and solution initialization loads source and constraints files, and configures the active solution for synthesis.



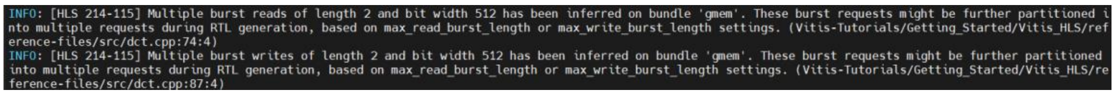
```
INFO: [HLS 200-10] Analyzing design file 'Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp' ...
INFO: [HLS 200-111] Finished Source Code Analysis and Preprocessing: CPU user time: 2.01 seconds. CPU system time: 0.22 seconds. Elapsed time: 1.48 seconds; c
urrent allocated memory: 207.621 MB.
```

Start compilation reads source files into memory.



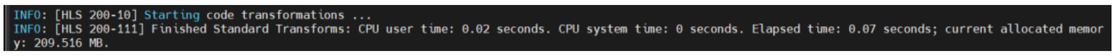
```
INFO: [HLS 200-777] Using interface defaults for 'Vitis' flow target.
```

Interface detection and setup reviews and generates port and block interfaces for the function.



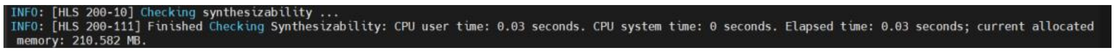
```
INFO: [HLS 214-115] Multiple burst reads of length 2 and bit width 512 has been inferred on bundle 'gmem'. These burst requests might be further partitioned i
nto multiple requests during RTL generation, based on max_read_burst_length or max_write_burst_length settings. (Vitis-Tutorials/Getting_Started/Vitis_HLS/ref
erence-files/src/dct.cpp:74:4)
INFO: [HLS 214-115] Multiple burst writes of length 2 and bit width 512 has been inferred on bundle 'gmem'. These burst requests might be further partitioned
into multiple requests during RTL generation, based on max_read_burst_length or max_write_burst_length settings. (Vitis-Tutorials/Getting_Started/Vitis_HLS/re
ference-files/src/dct.cpp:87:4)
```

Burst read and write analysis for ports/interfaces.



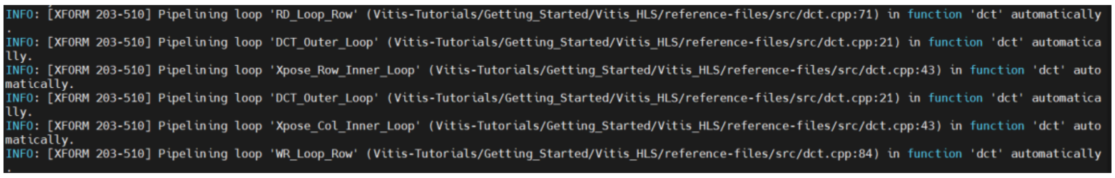
```
INFO: [HLS 200-10] Starting code transformations ...
INFO: [HLS 200-111] Finished Standard Transforms: CPU user time: 0.02 seconds. CPU system time: 0 seconds. Elapsed time: 0.07 seconds; current allocated memor
y: 209.516 MB.
```

Compiler transforms code to operations.



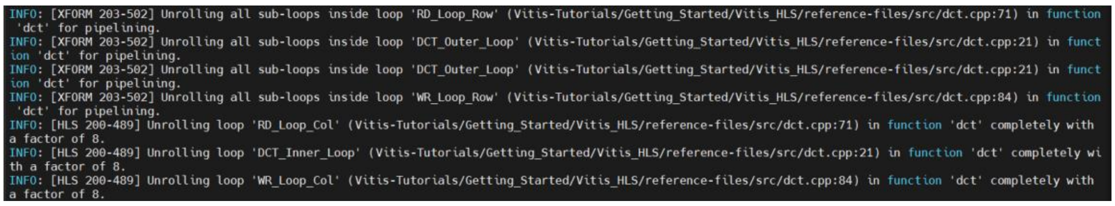
```
INFO: [HLS 200-10] Checking synthesizability ...
INFO: [HLS 200-111] Finished Checking Synthesizability: CPU user time: 0.03 seconds. CPU system time: 0 seconds. Elapsed time: 0.03 seconds; current allocated
 memory: 210.582 MB.
```

Performs Synthesizeability checks.



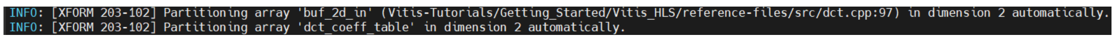
```
INFO: [XFORM 203-510] Pipelining loop 'RD_Loop_Row' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:71) in function 'dct' automatically
.
INFO: [XFORM 203-510] Pipelining loop 'DCT_Outer_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:21) in function 'dct' automatica
lly.
INFO: [XFORM 203-510] Pipelining loop 'Xpose_Row_Inner_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:43) in function 'dct' auto
matically.
INFO: [XFORM 203-510] Pipelining loop 'DCT_Outer_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:21) in function 'dct' automatica
lly.
INFO: [XFORM 203-510] Pipelining loop 'Xpose_Col_Inner_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:43) in function 'dct' auto
matically.
INFO: [XFORM 203-510] Pipelining loop 'WR_Loop_Row' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:84) in function 'dct' automatically
.
```

Automatic pipelining of loops at tripcount threshold.



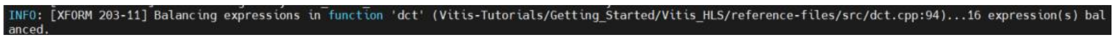
```
INFO: [XFORM 203-502] Unrolling all sub-loops inside loop 'RD_Loop_Row' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:71) in function
 'dct' for pipelining.
INFO: [XFORM 203-502] Unrolling all sub-loops inside loop 'DCT_Outer_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:21) in funct
ion 'dct' for pipelining.
INFO: [XFORM 203-502] Unrolling all sub-loops inside loop 'DCT_Outer_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:21) in funct
ion 'dct' for pipelining.
INFO: [XFORM 203-502] Unrolling all sub-loops inside loop 'WR_Loop_Row' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:84) in function
 'dct' for pipelining.
INFO: [HLS 200-489] Unrolling loop 'RD_Loop_Col' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:71) in function 'dct' completely with
a factor of 8.
INFO: [HLS 200-489] Unrolling loop 'DCT_Inner_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:21) in function 'dct' completely wi
th a factor of 8.
INFO: [HLS 200-489] Unrolling loop 'WR_Loop_Col' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:84) in function 'dct' completely with
a factor of 8.
```

Unrolling loops, both automatic and user-directed.



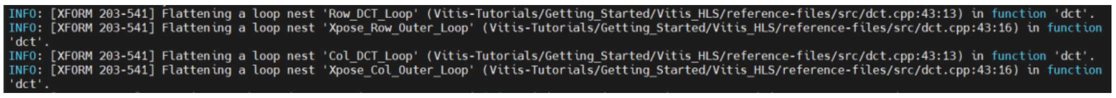
```
INFO: [XFORM 203-102] Partitioning array 'buf_2d_in' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:97) in dimension 2 automatically.
INFO: [XFORM 203-102] Partitioning array 'dct_coeff_table' in dimension 2 automatically.
```

Array partitioning.



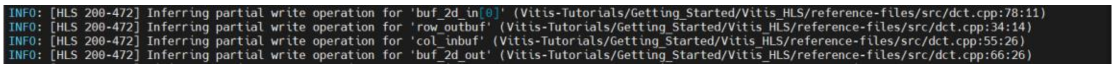
```
INFO: [XFORM 203-11] Balancing expressions in function 'dct' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:94)...16 expression(s) bal
anced.
```

Balance expressions using associative and commutative properties.



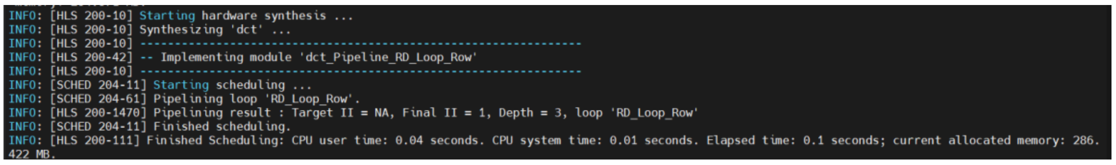
```
INFO: [XFORM 203-541] Flattening a loop nest 'Row_DCT_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:43:13) in function 'dct'.
INFO: [XFORM 203-541] Flattening a loop nest 'Xpose_Row_Outer_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:43:16) in function
'dct'.
INFO: [XFORM 203-541] Flattening a loop nest 'Col_DCT_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:43:13) in function 'dct'.
INFO: [XFORM 203-541] Flattening a loop nest 'Xpose_Col_Outer_Loop' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:43:16) in function
'dct'.
```

Loop flattening to reduce loop hierarchy.



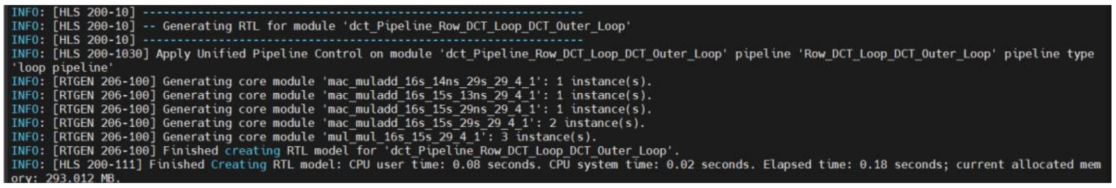
```
INFO: [HLS 200-472] Inferring partial write operation for 'buf_2d_in[0]' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:78:11)
INFO: [HLS 200-472] Inferring partial write operation for 'row_outbuf' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:34:14)
INFO: [HLS 200-472] Inferring partial write operation for 'col_inbuf' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:55:26)
INFO: [HLS 200-472] Inferring partial write operation for 'buf_2d_out' (Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp:66:26)
```

Partial write detection (writing part of a memory word)



```
INFO: [HLS 200-10] Starting hardware synthesis ...
INFO: [HLS 200-10] Synthesizing 'dct' ...
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-42] -- Implementing module 'dct_Pipeline_RD_Loop_Row'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining loop 'RD_Loop_Row'.
INFO: [HLS 200-1470] Pipelining result : Target II = NA, Final II = 1, Depth = 3, loop 'RD_Loop_Row'
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111] Finished Scheduling: CPU user time: 0.04 seconds. CPU system time: 0.01 seconds. Elapsed time: 0.1 seconds; current allocated memory: 286.
422 MB.
```

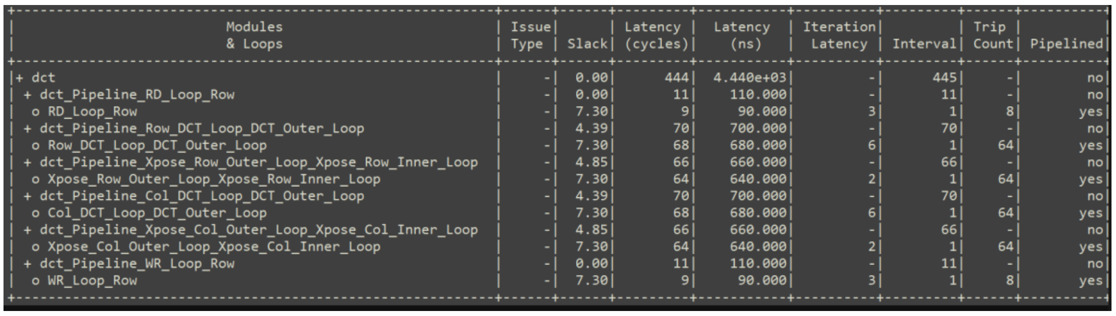
Finish architecture synthesis, start scheduling.



```
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-10] -- Generating RTL for module 'dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-1030] Apply Unified Pipeline Control on module 'dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop' pipeline 'Row_DCT_Loop_DCT_Outer_Loop' pipeline type
'loop pipeline'
INFO: [RTGEN 206-100] Generating core module 'mac_muladd_16s_14ns_29s_29_4_1': 1 instance(s).
INFO: [RTGEN 206-100] Generating core module 'mac_muladd_16s_15s_13ns_29_4_1': 1 instance(s).
INFO: [RTGEN 206-100] Generating core module 'mac_muladd_16s_15s_29ns_29_4_1': 1 instance(s).
INFO: [RTGEN 206-100] Generating core module 'mac_muladd_16s_15s_29s_29_4_1': 2 instance(s).
INFO: [RTGEN 206-100] Generating core module 'mul_mul_16s_15s_29_4_1': 3 instance(s).
INFO: [RTGEN 206-100] Finished creating RTL model for 'dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop'.
INFO: [HLS 200-111] Finished Creating RTL model: CPU user time: 0.08 seconds. CPU system time: 0.02 seconds. Elapsed time: 0.18 seconds; current allocated mem
ory: 293.012 MB.
```

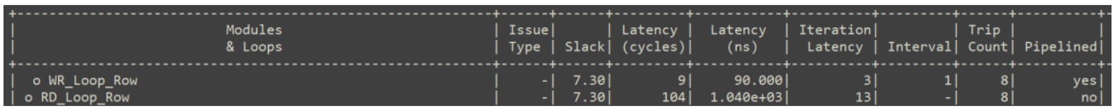
End scheduling, generate RTL code.

Analyze the timing/performance/utilization



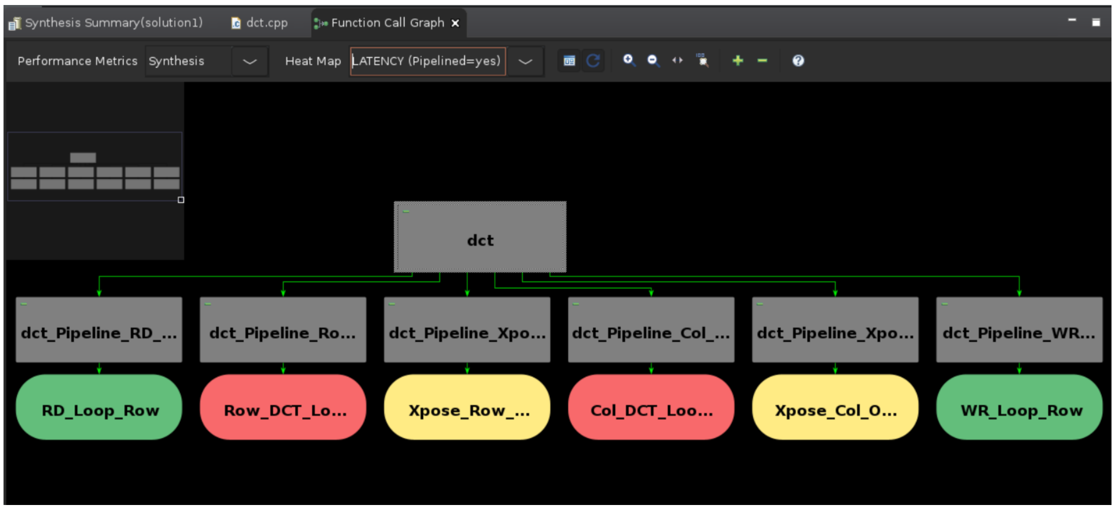
```
+-------------------------------------------------------------+------+-----+---------+-----------+----------+---------+------+----------+
|                          Modules                            | Issue|     | Latency |  Latency  | Iteration|         | Trip |          |
|                          & Loops                            | Type | Slack| (cycles)|    (ns)   |  Latency | Interval| Count| Pipelined|
+-------------------------------------------------------------+------+-----+---------+-----------+----------+---------+------+----------+
|+ dct                                                        |     -|  0.00|      444|  4.440e+03|         -|      445|     -|        no|
| + dct_Pipeline_RD_Loop_Row                                  |     -|  0.00|       11|    110.000|         -|       11|     -|        no|
|  o RD_Loop_Row                                              |     -|  7.30|        9|     90.000|         3|        1|     8|       yes|
| + dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop                  |     -|  4.39|       70|    700.000|         -|       70|     -|        no|
|  o Row_DCT_Loop_DCT_Outer_Loop                              |     -|  7.30|       68|    680.000|         6|        1|    64|       yes|
| + dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop    |     -|  4.85|       66|    660.000|         -|       66|     -|        no|
|  o Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop                |     -|  7.30|       64|    640.000|         2|        1|    64|       yes|
| + dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop                  |     -|  4.39|       70|    700.000|         -|       70|     -|        no|
|  o Col_DCT_Loop_DCT_Outer_Loop                              |     -|  7.30|       68|    680.000|         6|        1|    64|       yes|
| + dct_Pipeline_Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop    |     -|  4.85|       66|    660.000|         -|       66|     -|        no|
|  o Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop                |     -|  7.30|       64|    640.000|         2|        1|    64|       yes|
| + dct_Pipeline_WR_Loop_Row                                  |     -|  0.00|       11|    110.000|         -|       11|     -|        no|
|  o WR_Loop_Row                                              |     -|  7.30|        9|     90.000|         3|        1|     8|       yes|
+-------------------------------------------------------------+------+-----+---------+-----------+----------+---------+------+----------+
```

This is the latency/loop table resulting from the original setting in the tutorial. To see how it calculate the latency of loops, I disable the pipeline of RD_Loop_Row. The result is shown below.

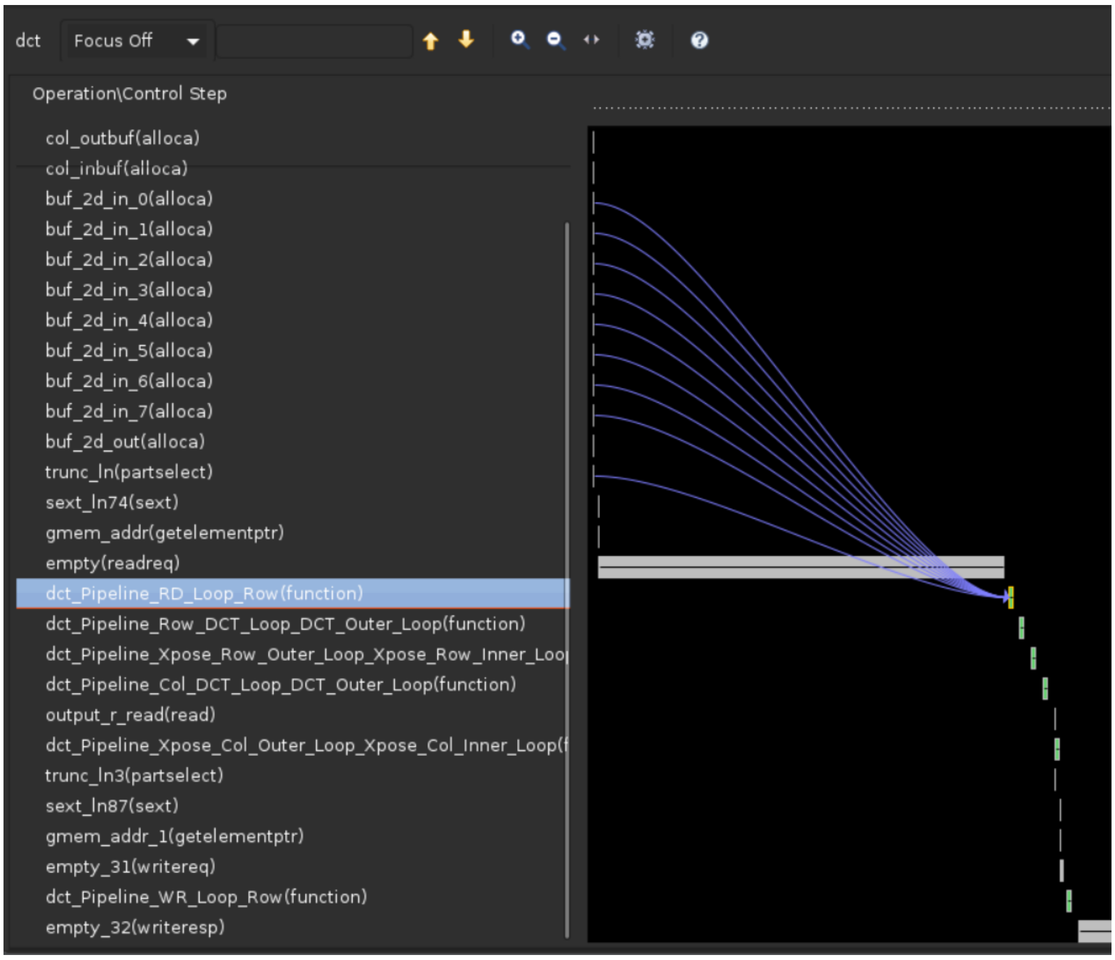


```
+-------------------------------------------------------------+------+-----+---------+-----------+----------+---------+------+----------+
|                          Modules                            | Issue|     | Latency |  Latency  | Iteration|         | Trip |          |
|                          & Loops                            | Type | Slack| (cycles)|    (ns)   |  Latency | Interval| Count| Pipelined|
+-------------------------------------------------------------+------+-----+---------+-----------+----------+---------+------+----------+
|  o WR_Loop_Row                                              |     -|  7.30|        9|     90.000|         3|        1|     8|       yes|
| o RD_Loop_Row                                               |     -|  7.30|      104|  1.040e+03|        13|        -|     8|        no|
```

We can see that if a loop is pipelined, the latency is small. The value is approximately $Iteration\ Latency + Interval \times (Trip\ Count - 1) = 10$. If a loop is not pipelined, the latency is large. The value is $Iteration\ Latency \times Trip\ Count = 104$.
From the latency table, we can also see that the critical blocks to optimize are the dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop and dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop, since their latency are largest. This can also be seen on Function Call Graph Viewer by setting heat map.

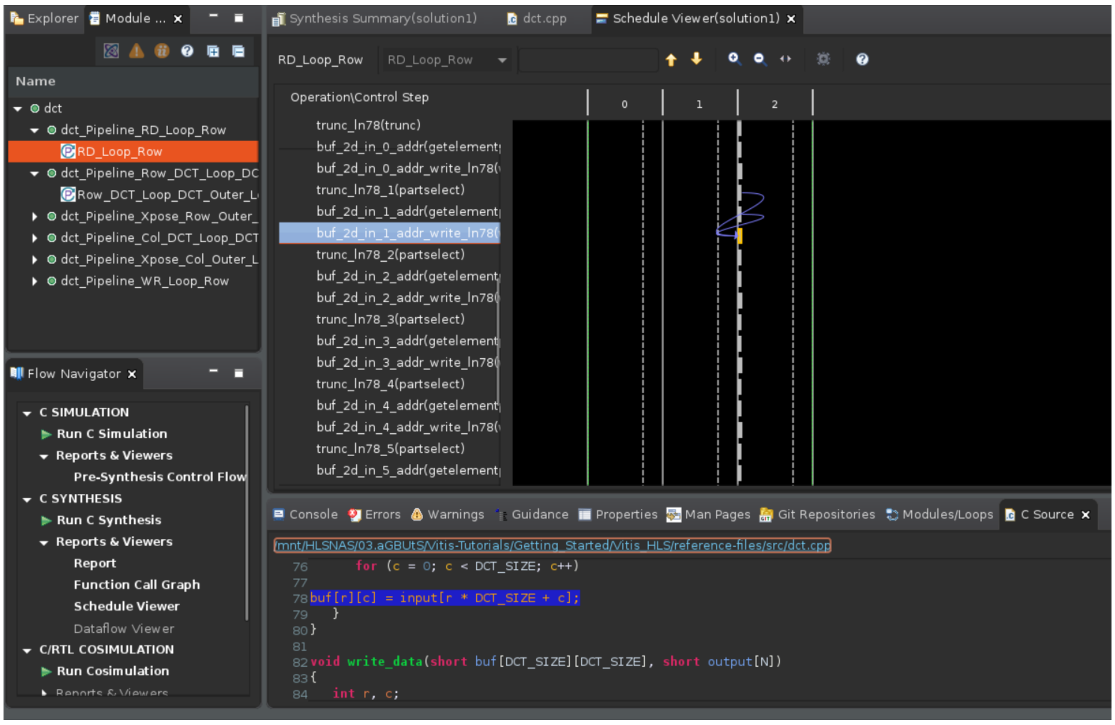


Schedule viewer



```
col_outbuf(alloca)
col_inbuf(alloca)
buf_2d_in_0(alloca)
buf_2d_in_1(alloca)
buf_2d_in_2(alloca)
buf_2d_in_3(alloca)
buf_2d_in_4(alloca)
buf_2d_in_5(alloca)
buf_2d_in_6(alloca)
buf_2d_in_7(alloca)
buf_2d_out(alloca)
trunc_ln(partselect)
sext_ln74(sext)
gmem_addr(getelementptr)
empty(readreq)
dct_Pipeline_RD_Loop_Row(function)
dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop(function)
dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loo
dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop(function)
output_r_read(read)
dct_Pipeline_Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop(
trunc_ln3(partselect)
sext_ln87(sext)
gmem_addr_1(getelementptr)
empty_31(writereq)
dct_Pipeline_WR_Loop_Row(function)
empty_32(writeresp)
```

From the viewer of the top module, we can find those modules introduced above, and they are arranged in order. We can also see some blue lines representing the data dependencies.

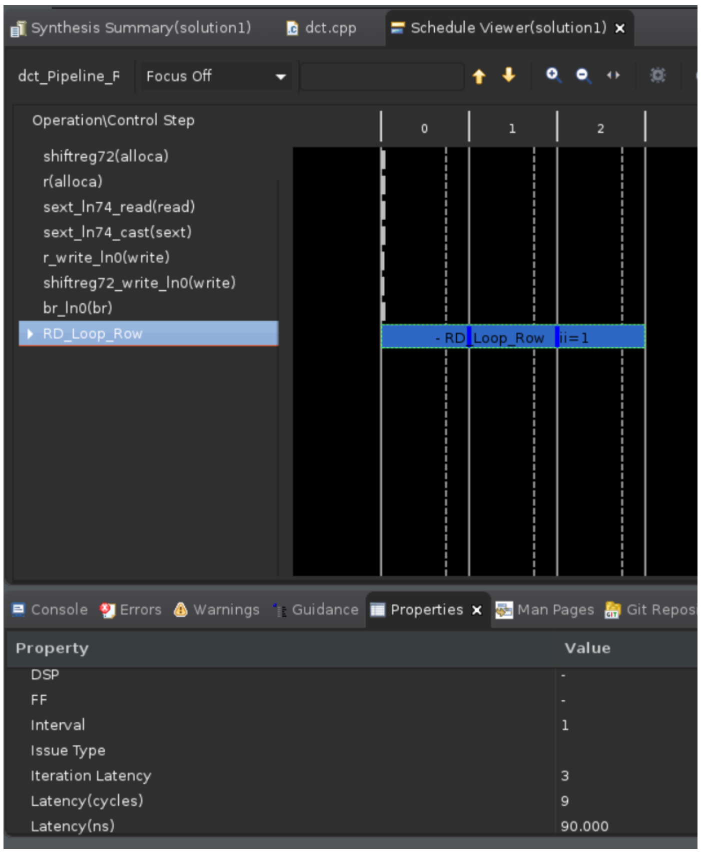


```
trunc_ln78(trunc)
buf_2d_in_0_addr(getelement
buf_2d_in_0_addr_write_ln78(
trunc_ln78_1(partselect)
buf_2d_in_1_addr(getelement
buf_2d_in_1_addr_write_ln78(
trunc_ln78_2(partselect)
buf_2d_in_2_addr(getelement
buf_2d_in_2_addr_write_ln78(
trunc_ln78_3(partselect)
buf_2d_in_3_addr(getelement
buf_2d_in_3_addr_write_ln78(
trunc_ln78_4(partselect)
buf_2d_in_4_addr(getelement
buf_2d_in_4_addr_write_ln78(
trunc_ln78_5(partselect)
buf_2d_in_5_addr(getelement
```

/mnt/HLSNAS/03.aGBUtS/Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp

```
      for (c = 0; c < DCT_SIZE; c++)

buf[r][c] = input[r * DCT_SIZE + c];
   }
}

void write_data(short buf[DCT_SIZE][DCT_SIZE], short output[N])
{
   int r, c;
```

From the module hierarchy, we can select the modules and loops to view their schedule. Besides, we can also right click on the operation to see where it was called.



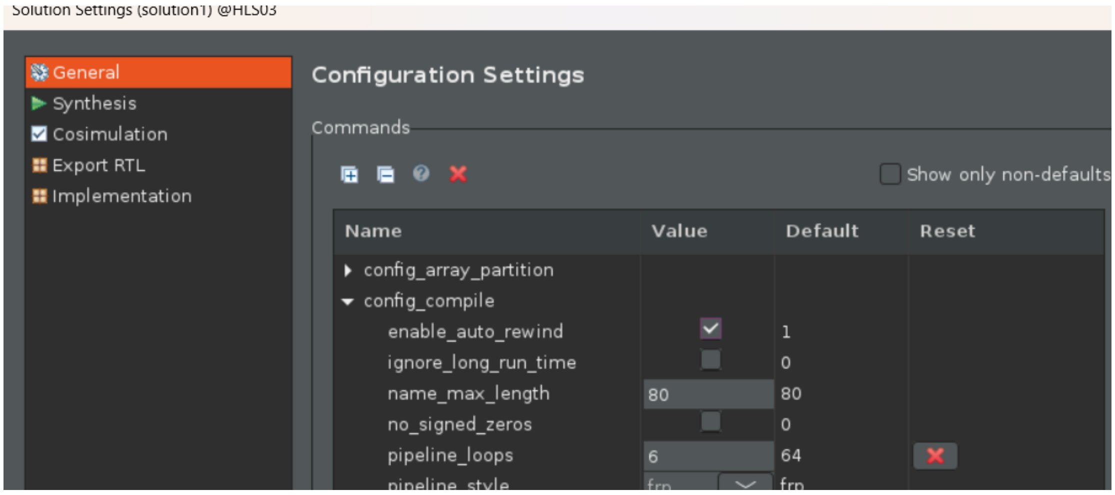
```
shiftreg72(alloca)
r(alloca)
sext_ln74_read(read)
sext_ln74_cast(sext)
r_write_ln0(write)
shiftreg72_write_ln0(write)
br_ln0(br)
RD_Loop_Row
```

Property	Value
DSP	-
FF	-
Interval	1
Issue Type	
Iteration Latency	3
Latency(cycles)	9
Latency(ns)	90.000

From the properties, we can also see the latency of selected operation.

If possible, share how you optimize the design and the trade-off you make

1. Configure the pipeline loops threshold



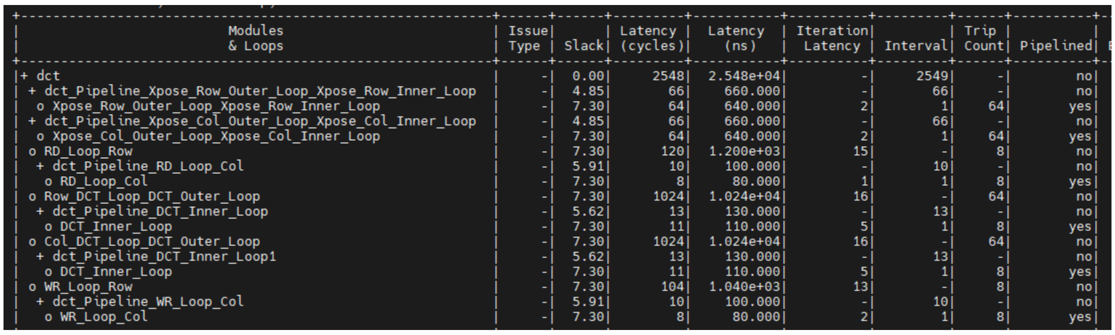
Solution Settings (solution1) @HLS03

Configuration Settings

Commands

Name	Value	Default	Reset
config_array_partition			
config_compile			
enable_auto_rewind	✓	1	
ignore_long_run_time		0	
name_max_length	80	80	
no_signed_zeros		0	
pipeline_loops	6	64	✗
pipeline_style	frp	frp	

In this method, we are asked to modify pipeline_loops as 6. It enables loops with an iteration count below 6 to be pipelined automatically.

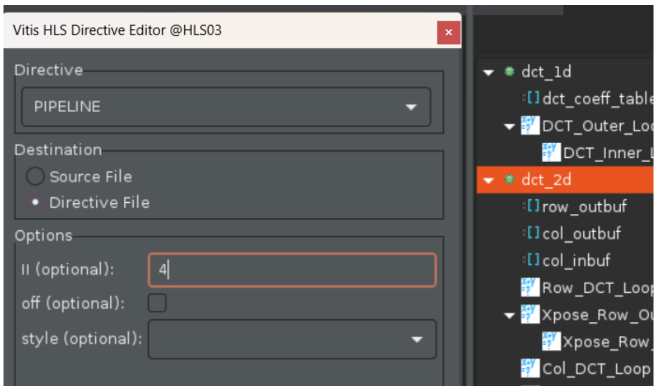


```
+------------------------------------------------------------+------+------+---------+-----------+----------+---------+------+----------+-
|                          Modules                           | Issue|      | Latency |  Latency  | Iteration|         | Trip |          |
|                          & Loops                           | Type | Slack| (cycles)|    (ns)   |  Latency | Interval| Count| Pipelined|
+------------------------------------------------------------+------+------+---------+-----------+----------+---------+------+----------+-
|+ dct                                                       |     -|  0.00|     2548|  2.548e+04|         -|     2549|     -|        no|
| + dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop   |     -|  4.85|       66|    660.000|         -|       66|     -|        no|
|  o Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop               |     -|  7.30|       64|    640.000|         2|        1|    64|       yes|
| + dct_Pipeline_Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop   |     -|  4.85|       66|    660.000|         -|       66|     -|        no|
|  o Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop               |     -|  7.30|       64|    640.000|         2|        1|    64|       yes|
| o RD_Loop_Row                                              |     -|  7.30|      120|  1.200e+03|        15|        -|     8|        no|
|  + dct_Pipeline_RD_Loop_Col                                |     -|  5.91|       10|    100.000|         -|       10|     -|        no|
|   o RD_Loop_Col                                            |     -|  7.30|        8|     80.000|         1|        1|     8|       yes|
| o Row_DCT_Loop_DCT_Outer_Loop                              |     -|  7.30|     1024|  1.024e+04|        16|        -|    64|        no|
|  + dct_Pipeline_DCT_Inner_Loop                             |     -|  5.62|       13|    130.000|         -|       13|     -|        no|
|   o DCT_Inner_Loop                                         |     -|  7.30|       11|    110.000|         5|        1|     8|       yes|
| o Col_DCT_Loop_DCT_Outer_Loop                              |     -|  7.30|     1024|  1.024e+04|        16|        -|    64|        no|
|  + dct_Pipeline_DCT_Inner_Loop1                            |     -|  5.62|       13|    130.000|         -|       13|     -|        no|
|   o DCT_Inner_Loop                                         |     -|  7.30|       11|    110.000|         5|        1|     8|       yes|
| o WR_Loop_Row                                              |     -|  7.30|      104|  1.040e+03|        13|        -|     8|        no|
|  + dct_Pipeline_WR_Loop_Col                                |     -|  5.91|       10|    100.000|         -|       10|     -|        no|
|   o WR_Loop_Col                                            |     -|  7.30|        8|     80.000|         2|        1|     8|       yes|
```

After synthesis, we can see that the latency increases a lot, and that many loops are not pipelined, resulting to large latency. By the way, increasing the value does not help in this case, since there is no loop with iteration counter larger than the default value, 64.

2. Configure the pipeline initiation interval

In the tutorial, we are asked to modify the II of dct_2d as 4.



Vitis HLS Directive Editor @HLS03

Directive: PIPELINE

Destination: Source File [✓ Directive File]

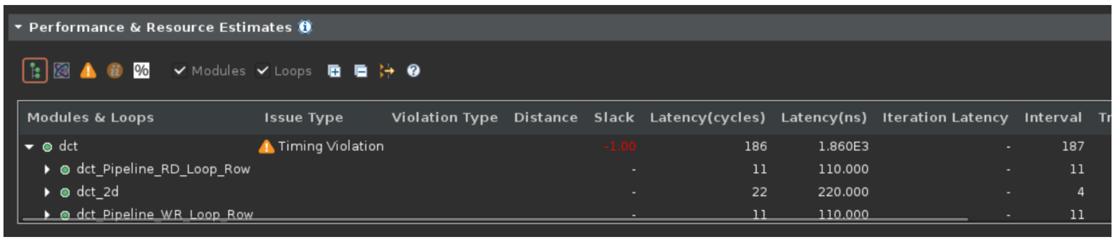
Options

II (optional): 4

off (optional):

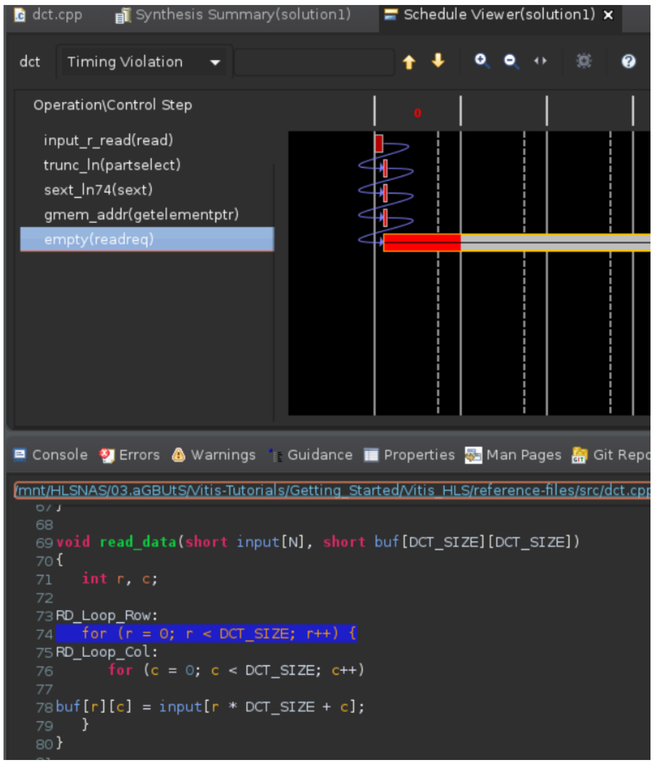
style (optional):

After synthesis, although the overall latency is decreased, there is timing violation.



Performance & Resource Estimates

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Tr
dct	Timing Violation			-1.00	186	1.860E3	-	187	
dct_Pipeline_RD_Loop_Row				-	11	110.000	-	11	
dct_2d				-	22	220.000	-	4	
dct_Pipeline_WR_Loop_Row				-	11	110.000	-	11	

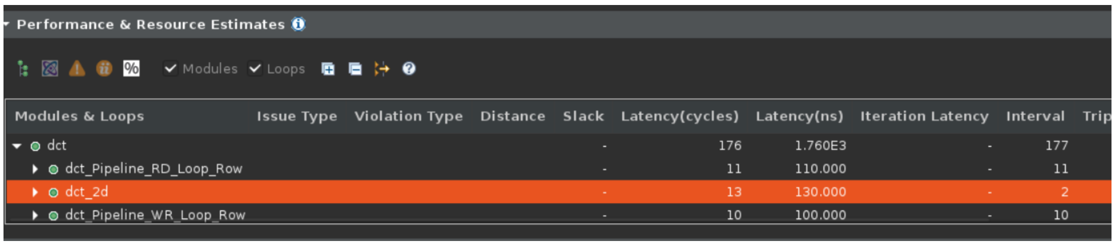


/mnt/HLSNAS/03.aGBUtS/Vitis-Tutorials/Getting_Started/Vitis_HLS/reference-files/src/dct.cpp

```

void read_data(short input[N], short buf[DCT_SIZE][DCT_SIZE])
{
   int r, c;

RD_Loop_Row:
   for (r = 0; r < DCT_SIZE; r++) {
RD_Loop_Col:
      for (c = 0; c < DCT_SIZE; c++)

buf[r][c] = input[r * DCT_SIZE + c];
   }
}
```

We can see that the timing violation occurred during reading data. I tried to figure out how to fix it by adding some directives, but end up failing.

I tried to modify the II as 2, and it works well. The figure below is the result performance. We can see that the latency is greatly decreased.

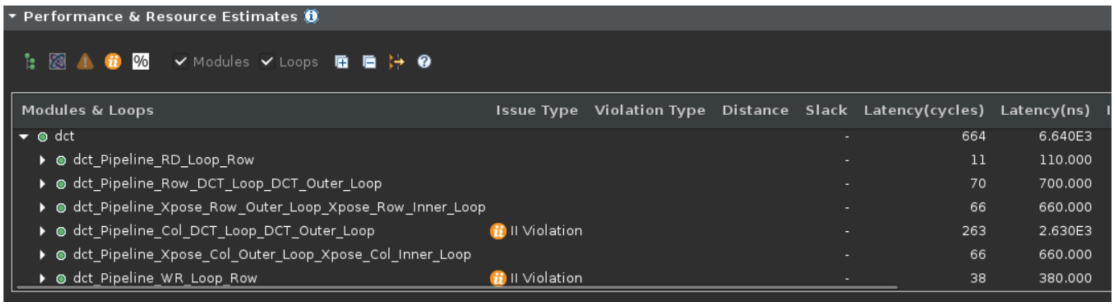


Performance & Resource Estimates

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip
dct				-	176	1.760E3	-	177	
dct_Pipeline_RD_Loop_Row				-	11	110.000	-	11	
dct_2d				-	13	130.000	-	2	
dct_Pipeline_WR_Loop_Row				-	10	100.000	-	10	

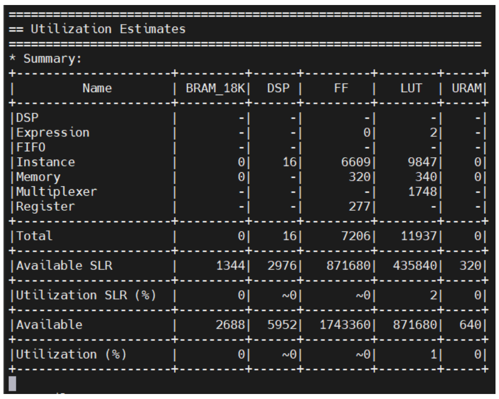
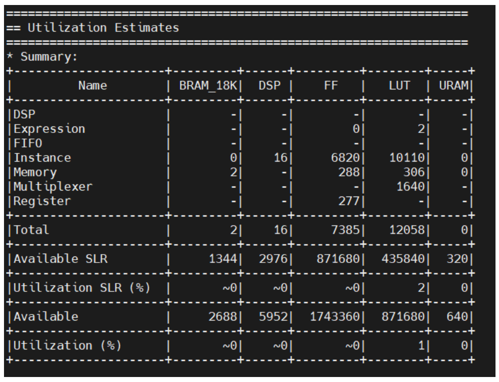
3. Assign dual-port RAMs with BIND_STORAGE

   In the tutorial, we are asked to use BRAMs to store col_inbuf and buf_2d_out. After synthesis, the result is shown in the figure below.



```
Performance & Resource Estimates
Modules  Loops

Modules & Loops                                        Issue Type     Violation Type  Distance  Slack  Latency(cycles)  Latency(ns)
dct                                                                                             -      664              6.640E3
  dct_Pipeline_RD_Loop_Row                                                                      -      11               110.000
  dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop                                                      -      70               700.000
  dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop                                        -      66               660.000
  dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop               II Violation                           -      263              2.630E3
  dct_Pipeline_Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop                                        -      66               660.000
  dct_Pipeline_WR_Loop_Row                               II Violation                           -      38               380.000
```

We can see that the total latency increases a lot. Besides, II violation is issued (II violation does not influence the co-simulation result. In HLS, II violation will be issued if II is not 1.) Take a look at the report, and we can see that it uses more resource as well. Therefore, I think that this method is not for this application.

Original method	This optimization method
	

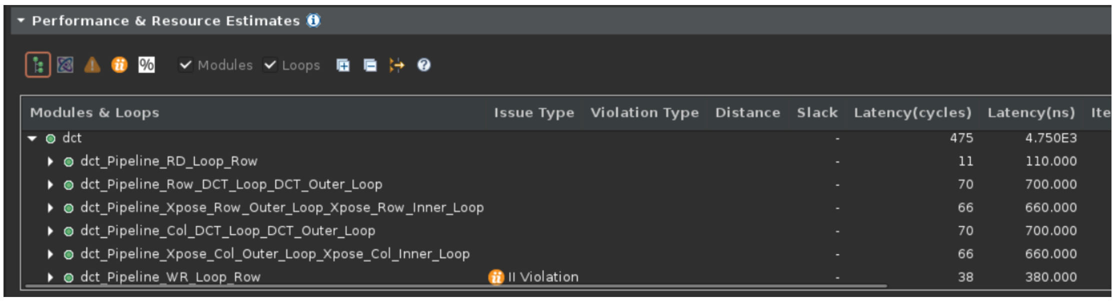
Original method:

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+------+---------+--------+-----+
|         Name        | BRAM_18K|  DSP |    FF   |   LUT  | URAM|
+---------------------+---------+------+---------+--------+-----+
|DSP                  |        -|     -|        -|       -|    -|
|Expression           |        -|     -|        0|       2|    -|
|FIFO                 |        -|     -|        -|       -|    -|
|Instance             |        0|    16|     6609|    9847|    0|
|Memory               |        0|     -|      320|     340|    0|
|Multiplexer          |        -|     -|        -|    1748|    -|
|Register             |        -|     -|      277|       -|    -|
+---------------------+---------+------+---------+--------+-----+
|Total                |        0|    16|     7206|   11937|    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|
+---------------------+---------+------+---------+--------+-----+
```

This optimization method:

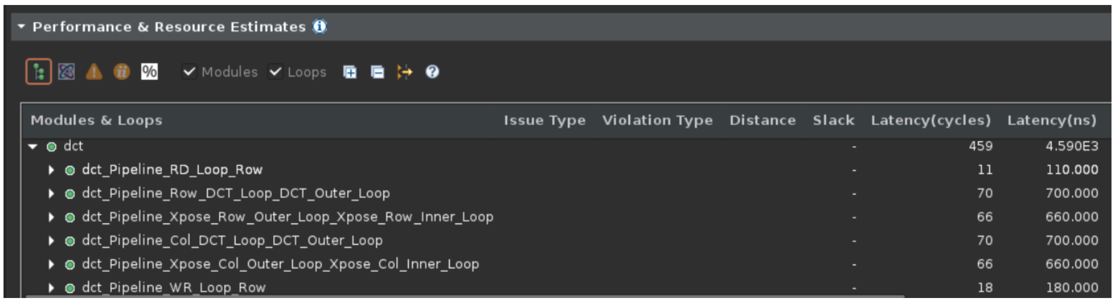
```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+------+---------+--------+-----+
|         Name        | BRAM_18K|  DSP |    FF   |   LUT  | URAM|
+---------------------+---------+------+---------+--------+-----+
|DSP                  |        -|     -|        -|       -|    -|
|Expression           |        -|     -|        0|       2|    -|
|FIFO                 |        -|     -|        -|       -|    -|
|Instance             |        0|    16|     6820|   10110|    0|
|Memory               |        2|     -|      288|     306|    0|
|Multiplexer          |        -|     -|        -|    1640|    -|
|Register             |        -|     -|      277|       -|    -|
+---------------------+---------+------+---------+--------+-----+
|Total                |        2|    16|     7385|   12058|    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|
+---------------------+---------+------+---------+--------+-----+
```

I also tried to remove the II violation. I think that violation happened since the bandwidth of BRAM is not large enough. Therefore, I try to partition the array (the next optimization method). The result is shown below.



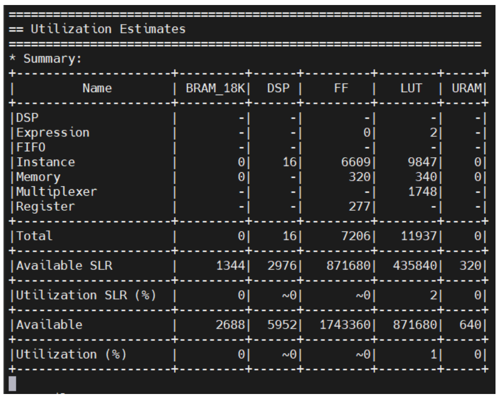
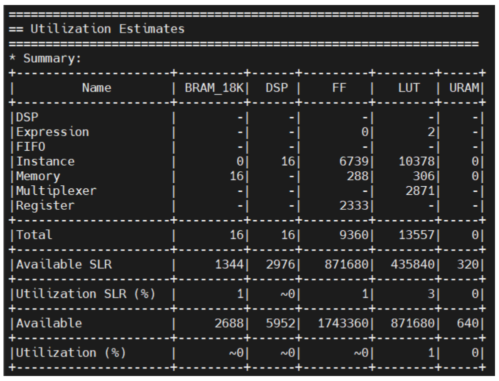
```
Performance & Resource Estimates
Modules  Loops

Modules & Loops                                        Issue Type     Violation Type  Distance  Slack  Latency(cycles)  Latency(ns)
dct                                                                                             -      475              4.750E3
  dct_Pipeline_RD_Loop_Row                                                                      -      11               110.000
  dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop                                                      -      70               700.000
  dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop                                        -      66               660.000
  dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop                                                      -      70               700.000
  dct_Pipeline_Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop                                        -      66               660.000
  dct_Pipeline_WR_Loop_Row                               II Violation                           -      38               380.000
```

We can see that the latency is reduced (664 -> 475), but still larger than that of the original method (444). To remove the II violation of writing output, I enlarge the II of WR_Loop_ROW as 2. The result is shown below.



```
Performance & Resource Estimates
Modules  Loops

Modules & Loops                                          Issue Type  Violation Type  Distance  Slack  Latency(cycles)  Latency(ns)
dct                                                                                             -      459              4.590E3
  dct_Pipeline_RD_Loop_Row                                                                      -      11               110.000
  dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop                                                      -      70               700.000
  dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop                                        -      66               660.000
  dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop                                                      -      70               700.000
  dct_Pipeline_Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop                                        -      66               660.000
  dct_Pipeline_WR_Loop_Row                                                                      -      18               180.000
```

No II violation is issued, and the latency decreases a little (still larger than that of the original method).

Original method	This optimization method
	

Original method:

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-----+--------+--------+-----+
|       Name      | BRAM_18K| DSP |   FF   |   LUT  | URAM|
+-----------------+---------+-----+--------+--------+-----+
|DSP              |        -|    -|       -|       -|    -|
|Expression       |        -|    -|       0|       2|    -|
|FIFO             |        -|    -|       -|       -|    -|
|Instance         |        0|   16|    6609|    9847|    0|
|Memory           |        0|    -|     320|     340|    0|
|Multiplexer      |        -|    -|       -|    1748|    -|
|Register         |        -|    -|     277|       -|    -|
+-----------------+---------+-----+--------+--------+-----+
|Total            |        0|   16|    7206|   11937|    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|
+-----------------+---------+-----+--------+--------+-----+
```

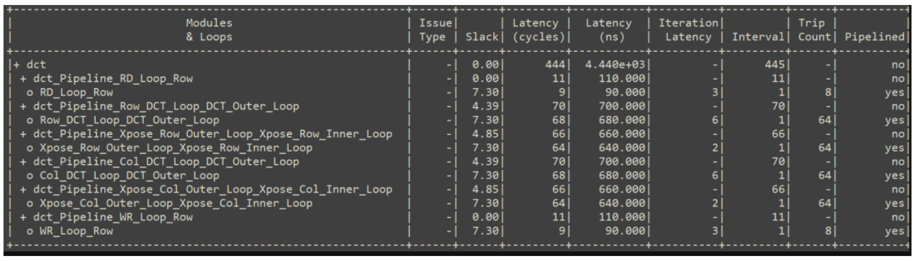
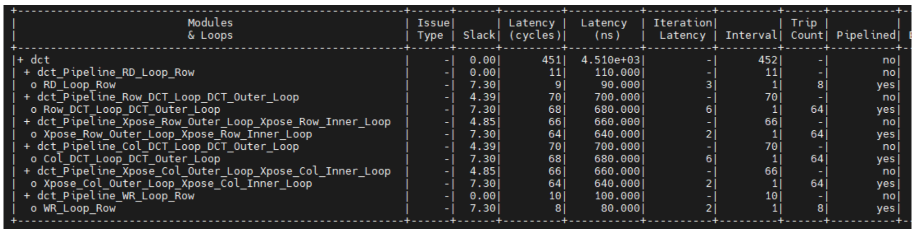
This optimization method:

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-----+--------+--------+-----+
|       Name      | BRAM_18K| DSP |   FF   |   LUT  | URAM|
+-----------------+---------+-----+--------+--------+-----+
|DSP              |        -|    -|       -|       -|    -|
|Expression       |        -|    -|       0|       2|    -|
|FIFO             |        -|    -|       -|       -|    -|
|Instance         |        0|   16|    6739|   10378|    0|
|Memory           |       16|    -|     288|     306|    0|
|Multiplexer      |        -|    -|       -|    2871|    -|
|Register         |        -|    -|    2333|       -|    -|
+-----------------+---------+-----+--------+--------+-----+
|Total            |       16|   16|    9360|   13557|    0|
+-----------------+---------+-----+--------+--------+-----+
|Available SLR    |     1344| 2976|  871680|  435840|  320|
+-----------------+---------+-----+--------+--------+-----+
|Utilization SLR (%)|      1|   ~0|       1|       3|    0|
+-----------------+---------+-----+--------+--------+-----+
|Available        |     2688| 5952| 1743360|  871680|  640|
+-----------------+---------+-----+--------+--------+-----+
|Utilization (%)  |       ~0|   ~0|      ~0|       1|    0|
+-----------------+---------+-----+--------+--------+-----+
```

Again, compare their utilization, and we can see that the original method looks better.

4. Assign an array partition

In the tutorial, we are asked to partition the arrays, col_inbuf and buf_2d_out. The result is shown in the figures below.

Original method	
Array partition	

Original method:

```
+-----------------------------------------------------------+------+------+---------+-----------+----------+---------+------+----------+
|                          Modules                          | Issue|      | Latency |  Latency  | Iteration|         | Trip |          |
|                          & Loops                          | Type | Slack| (cycles)|    (ns)   |  Latency | Interval| Count| Pipelined|
+-----------------------------------------------------------+------+------+---------+-----------+----------+---------+------+----------+
|+ dct                                                      |     -|  0.00|      444|  4.440e+03|         -|      445|     -|        no|
| + dct_Pipeline_RD_Loop_Row                                |     -|  0.00|       11|    110.000|         -|       11|     -|        no|
|  o RD_Loop_Row                                            |     -|  7.30|        9|     90.000|         3|        1|     8|       yes|
| + dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop                |     -|  4.39|       70|    700.000|         -|       70|     -|        no|
|  o Row_DCT_Loop_DCT_Outer_Loop                            |     -|  7.30|       68|    680.000|         6|        1|    64|       yes|
| + dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop  |     -|  4.85|       66|    660.000|         -|       66|     -|        no|
|  o Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop              |     -|  7.30|       64|    640.000|         2|        1|    64|       yes|
| + dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop                |     -|  4.39|       70|    700.000|         -|       70|     -|        no|
|  o Col_DCT_Loop_DCT_Outer_Loop                            |     -|  7.30|       68|    680.000|         6|        1|    64|       yes|
| + dct_Pipeline_Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop  |     -|  4.85|       66|    660.000|         -|       66|     -|        no|
|  o Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop              |     -|  7.30|       64|    640.000|         2|        1|    64|       yes|
| + dct_Pipeline_WR_Loop_Row                                |     -|  0.00|       11|    110.000|         -|       11|     -|        no|
|  o WR_Loop_Row                                            |     -|  7.30|        9|     90.000|         3|        1|     8|       yes|
+-----------------------------------------------------------+------+------+---------+-----------+----------+---------+------+----------+
```

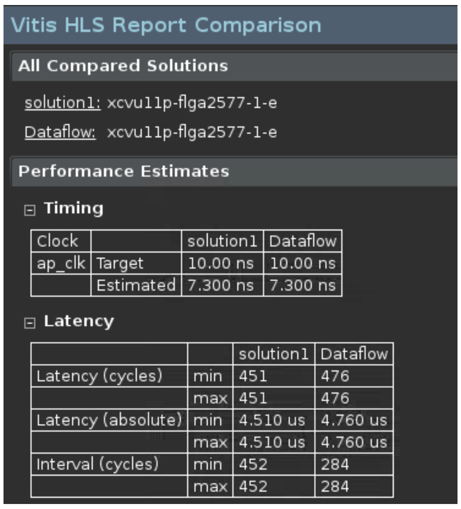
Array partition:

```
+-----------------------------------------------------------+------+------+---------+-----------+----------+---------+------+----------+-
|                          Modules                          | Issue|      | Latency |  Latency  | Iteration|         | Trip |          |
|                          & Loops                          | Type | Slack| (cycles)|    (ns)   |  Latency | Interval| Count| Pipelined| E
+-----------------------------------------------------------+------+------+---------+-----------+----------+---------+------+----------+-
|+ dct                                                      |     -|  0.00|      451|  4.510e+03|         -|      452|     -|        no|
| + dct_Pipeline_RD_Loop_Row                                |     -|  0.00|       11|    110.000|         -|       11|     -|        no|
|  o RD_Loop_Row                                            |     -|  7.30|        9|     90.000|         3|        1|     8|       yes|
| + dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop                |     -|  4.39|       70|    700.000|         -|       70|     -|        no|
|  o Row_DCT_Loop_DCT_Outer_Loop                            |     -|  7.30|       68|    680.000|         6|        1|    64|       yes|
| + dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop  |     -|  4.85|       66|    660.000|         -|       66|     -|        no|
|  o Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop              |     -|  7.30|       64|    640.000|         2|        1|    64|       yes|
| + dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop                |     -|  4.39|       70|    700.000|         -|       70|     -|        no|
|  o Col_DCT_Loop_DCT_Outer_Loop                            |     -|  7.30|       68|    680.000|         6|        1|    64|       yes|
| + dct_Pipeline_Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop  |     -|  4.85|       66|    660.000|         -|       66|     -|        no|
|  o Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop              |     -|  7.30|       64|    640.000|         2|        1|    64|       yes|
| + dct_Pipeline_WR_Loop_Row                                |     -|  0.00|       10|    100.000|         -|       10|     -|        no|
|  o WR_Loop_Row                                            |     -|  7.30|        8|     80.000|         2|        1|     8|       yes|
+-----------------------------------------------------------+------+------+---------+-----------+----------+---------+------+----------+-
```

We can see that the iteration latency of WR_Loop_Row decreases. However, I have no idea why the overall latency increases.

5. Dataflow optimization

In the tutorial, we are asked to add pragma DATAFLOW to the directives. This pragma enables task-level pipelining. The HLS tool will analyze the dataflow between functions and see whether they can overlap or not. The comparison result is shown in the figure below.



Vitis HLS Report Comparison

All Compared Solutions

solution1: xcvu11p-flga2577-1-e

Dataflow: xcvu11p-flga2577-1-e

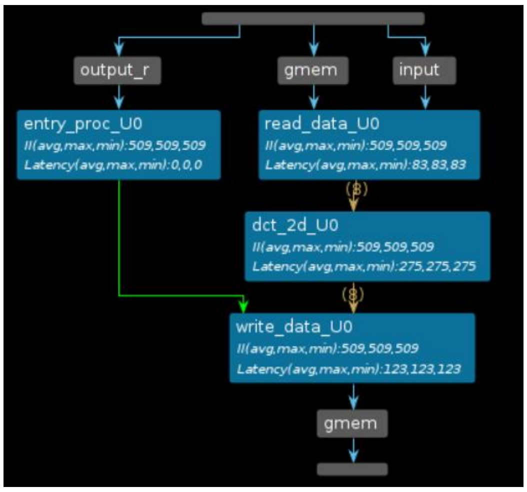
Performance Estimates

⊟ Timing

Clock		solution1	Dataflow
ap_clk	Target	10.00 ns	10.00 ns
	Estimated	7.300 ns	7.300 ns

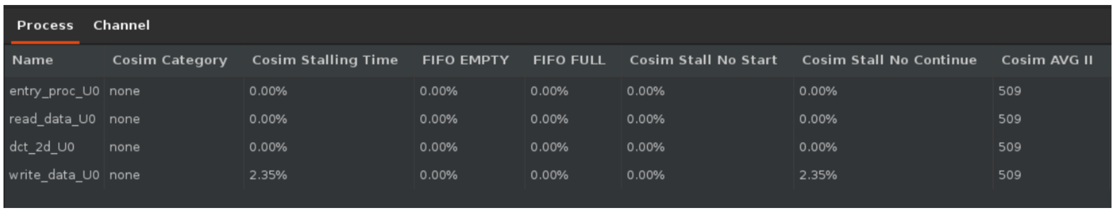
⊟ Latency

		solution1	Dataflow
Latency (cycles)	min	451	476
	max	451	476
Latency (absolute)	min	4.510 us	4.760 us
	max	4.510 us	4.760 us
Interval (cycles)	min	452	284
	max	452	284


output_r
gmem
input
entry_proc_U0
II(avg,max,min):509,509,509
Latency(avg,max,min):0,0,0
read_data_U0
II(avg,max,min):509,509,509
Latency(avg,max,min):83,83,83
dct_2d_U0
II(avg,max,min):509,509,509
Latency(avg,max,min):275,275,275
write_data_U0
II(avg,max,min):509,509,509
Latency(avg,max,min):123,123,123
gmem

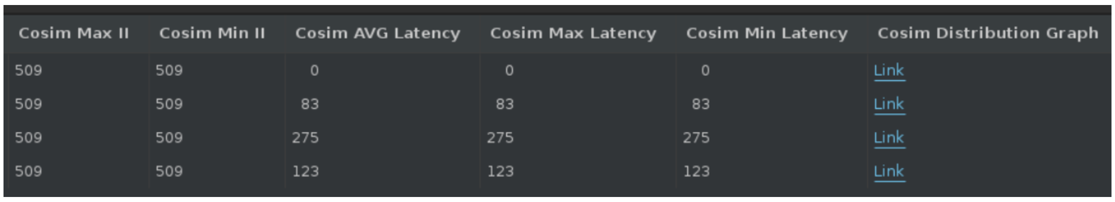
We can see that the interval between functions decreases a lot.

After running co-simulation, we can see the dataflow graph. From the graph, we can see that the three blocks, dct_2d, read_data, write_data are not running in parallel. From the process table below, we can see some information.



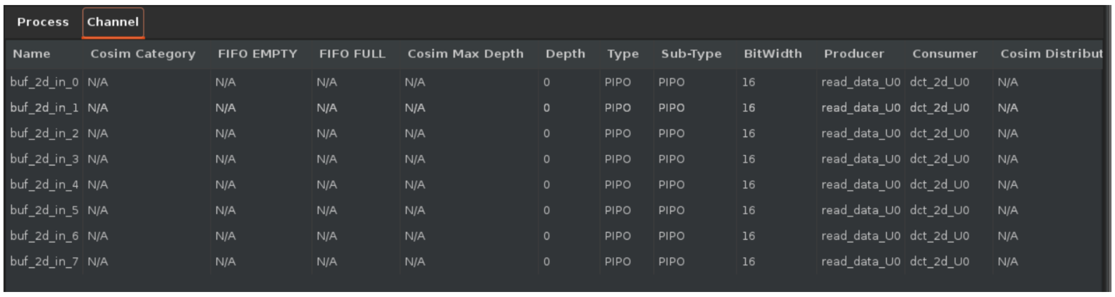
Process Channel

Name	Cosim Category	Cosim Stalling Time	FIFO EMPTY	FIFO FULL	Cosim Stall No Start	Cosim Stall No Continue	Cosim AVG II
entry_proc_U0	none	0.00%	0.00%	0.00%	0.00%	0.00%	509
read_data_U0	none	0.00%	0.00%	0.00%	0.00%	0.00%	509
dct_2d_U0	none	0.00%	0.00%	0.00%	0.00%	0.00%	509
write_data_U0	none	2.35%	0.00%	0.00%	0.00%	2.35%	509



Cosim Max II	Cosim Min II	Cosim AVG Latency	Cosim Max Latency	Cosim Min Latency	Cosim Distribution Graph
509	509	0	0	0	Link
509	509	83	83	83	Link
509	509	275	275	275	Link
509	509	123	123	123	Link

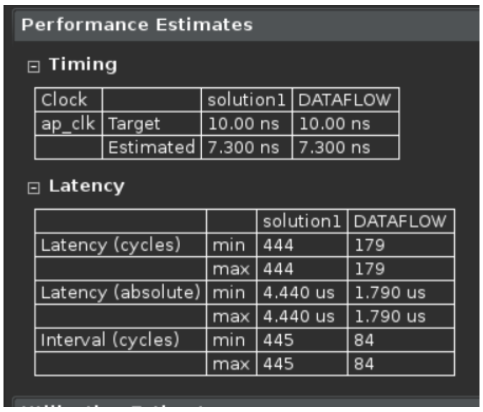
From the channel table below, we can see the channel type and the bit width.



Process Channel

Name	Cosim Category	FIFO EMPTY	FIFO FULL	Cosim Max Depth	Depth	Type	Sub-Type	BitWidth	Producer	Consumer	Cosim Distribut
buf_2d_in_0	N/A	N/A	N/A	N/A	0	PIPO	PIPO	16	read_data_U0	dct_2d_U0	N/A
buf_2d_in_1	N/A	N/A	N/A	N/A	0	PIPO	PIPO	16	read_data_U0	dct_2d_U0	N/A
buf_2d_in_2	N/A	N/A	N/A	N/A	0	PIPO	PIPO	16	read_data_U0	dct_2d_U0	N/A
buf_2d_in_3	N/A	N/A	N/A	N/A	0	PIPO	PIPO	16	read_data_U0	dct_2d_U0	N/A
buf_2d_in_4	N/A	N/A	N/A	N/A	0	PIPO	PIPO	16	read_data_U0	dct_2d_U0	N/A
buf_2d_in_5	N/A	N/A	N/A	N/A	0	PIPO	PIPO	16	read_data_U0	dct_2d_U0	N/A
buf_2d_in_6	N/A	N/A	N/A	N/A	0	PIPO	PIPO	16	read_data_U0	dct_2d_U0	N/A
buf_2d_in_7	N/A	N/A	N/A	N/A	0	PIPO	PIPO	16	read_data_U0	dct_2d_U0	N/A

Can it be further optimized? I tried the pipeline method taught in second method, and removed the array partitioning. The result is shown in the figure below.



Performance Estimates

Timing

Clock		solution1	DATAFLOW
ap_clk	Target	10.00 ns	10.00 ns
	Estimated	7.300 ns	7.300 ns

Latency

		solution1	DATAFLOW
Latency (cycles)	min	444	179
	max	444	179
Latency (absolute)	min	4.440 us	1.790 us
	max	4.440 us	1.790 us
Interval (cycles)	min	445	84
	max	445	84

We can see that the interval is now only 84.

Explain what you observed and learned

In this lab, I learned some tools to help us analysis our design, such as schedule viewer, function call graph, and dataflow viewer. I also learned some optimization methods, such as pipelining, using dual-port RAMs, array partitioning, and dataflow optimization.

Explain what problem you encountered and how you solved it

Timing violation in the second optimization method.
II violation in the third optimization method.
Solutions have been introduced in the above section.

Submit your work to Github

link: https://github.com/JasonYanggg/HLS