HLS LabA Report D10943005 林祐丞

Chapter 6: Design Analysis Run on Vitis HLS

Reference:

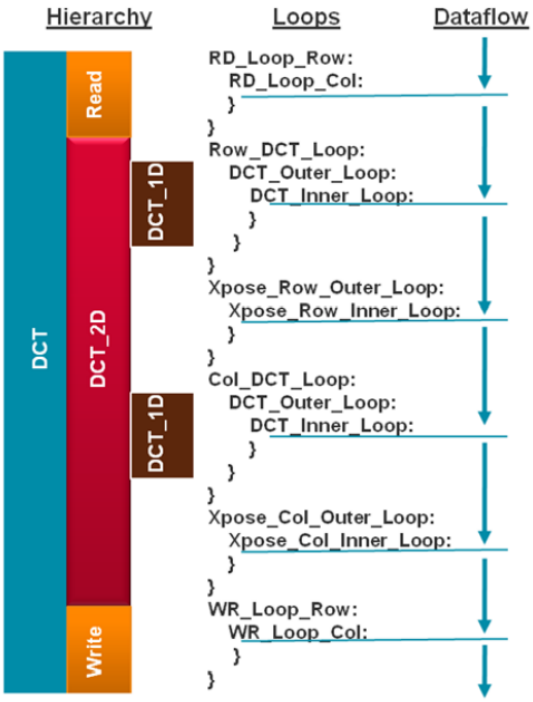
Vivado Design Suite Tutorial: High-Level Synthesis (UG871) - Xilinx

Vitis High-Level Synthesis User Guide (UG1399) - Xilinx

1. Source Code of the DCT Design

Design file:	
--- `dct.cpp`	DCT function file
--- `dct.h`	Header file
Design testbench:	
--- `dct_test.cpp`	Design testbench
Data file:	
--- `dct_coeff_table.txt`	Coefficient data
--- `in.dat`	Input data
--- `out.golden.dat`	Output golden data

The code hierarchy is shown as below:



The top-level function dct has three sub-functions: read_data, dct_2d and write_data. Function dct_2d has a single sub-function dct_1d. Note that in Vitis HLS, all subfunction is inlined during C synthesis, so the hierarchy is flattened.

The read_data and write_data functions read and write DCT data from the buffer. The dct_2d performs row-wise and col-wise dct_1d. Data transposition is performed after dct_1d.

2. Synthesis Log Analysis

The C synthesis of Vitis HLS can be separated into the following steps:

a、 Source Code Analysis and Preprocessing

Analyze the design file dct.cpp.

b、 Compiling Optimization and Transform

Inline functions so the program can be combined into a single function call, e.g., `dct(short*, short*)`.

c、 Checking Pragmas, Standard Transforms, and Checking Synthesizability

d、 Loop, function, and other optimizations*

The loops are pipelined automatically in the function, with loop unrolling performed for better pipelining. In addition, the array dct_coeff_table is partitioned in dimension 2 automatically.

e、 Generating RTL models for each module

i. Scheduling: Pipelining loop and show the resulted `II`.*

ii. Micro-architecture Generation: Exploring resource sharing.

iii. Binding: Implements the operations using DSP module resources

f、 Creating RTL model for each module

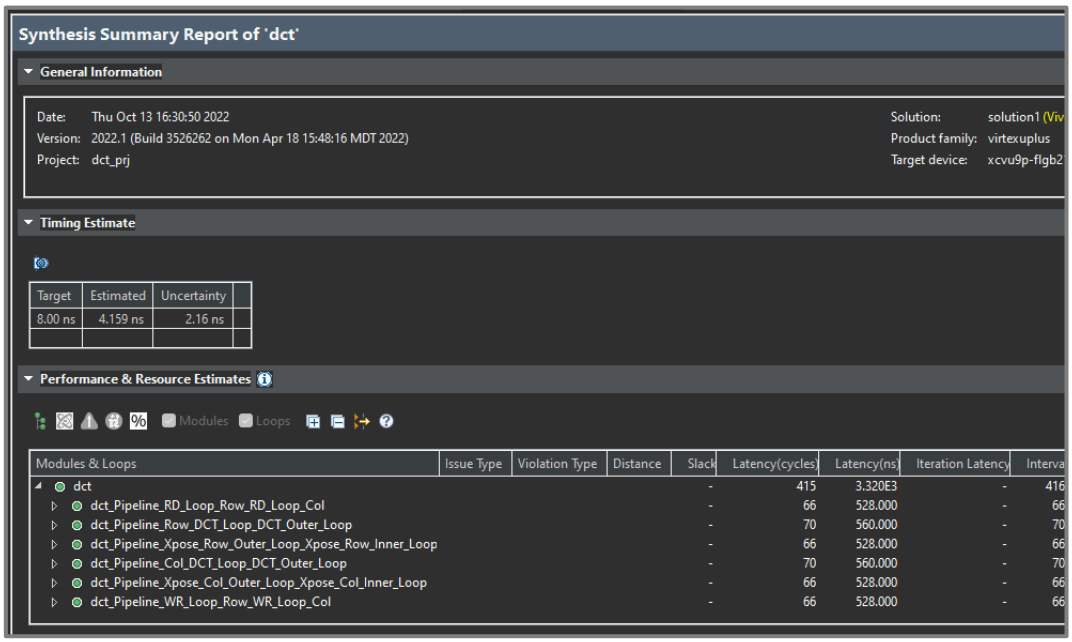
g、 Generating all RTL models

- *In Vitis HLS 2022.1, nested loops are flattened to improve latency. Hence, all nested loops involving two-dimensional arrays in DCT are automatically flattened to a single loop.*
- *The Vitis HLS tool also automatically pipelined loops that have fewer than a specified number of iterations. When pipelining, the tool tries to achieve an `II` of 1. The `II` is the number of clock cycles before the next iteration of the loop is processed. When pipelining the loop with `II` =1, you want the next*

iteration to start at the next clock cycle.

3. Synthesis Report Analysis

The synthesis summary report is generated after C synthesis:



Synthesis Summary Report of 'dct'

General Information

Date: Thu Oct 13 16:30:50 2022
Version: 2022.1 (Build 3526262 on Mon Apr 18 15:48:16 MDT 2022)
Project: dct_prj
Solution: solution1 (Viv
Product family: virtexuplus
Target device: xcvu9p-flgb2

Timing Estimate

Target	Estimated	Uncertainty
8.00 ns	4.159 ns	2.16 ns

Performance & Resource Estimates

Modules Loops

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interva
dct				-	415	3.320E3	-	416
dct_Pipeline_RD_Loop_Row_RD_Loop_Col				-	66	528.000	-	66
dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop				-	70	560.000	-	70
dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop				-	66	528.000	-	66
dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop				-	70	560.000	-	70
dct_Pipeline_Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop				-	66	528.000	-	66
dct_Pipeline_WR_Loop_Row_WR_Loop_Col				-	66	528.000	-	66

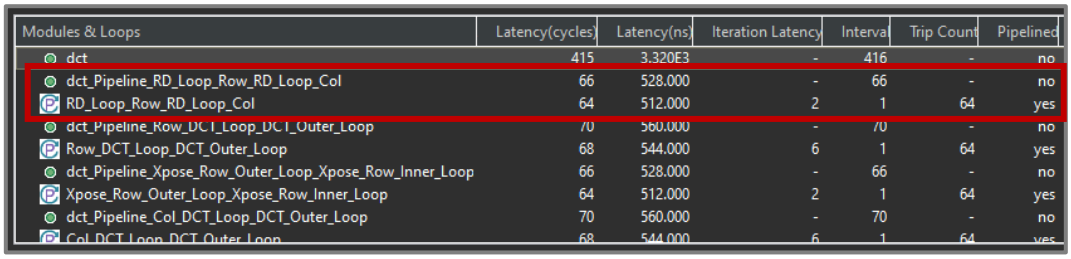
Timing Estimate:

The target clock period is the user-defined clock specification. The timing delay is estimated and displayed in the second column.

- *The estimated clock period = the clock period - the uncertainty results*

Performance & Resource Estimates:

Using the first module (dct_Pipeline_RD_Loop_Row_RD_Loop_Col) as an example, we can see why the latency is 528:



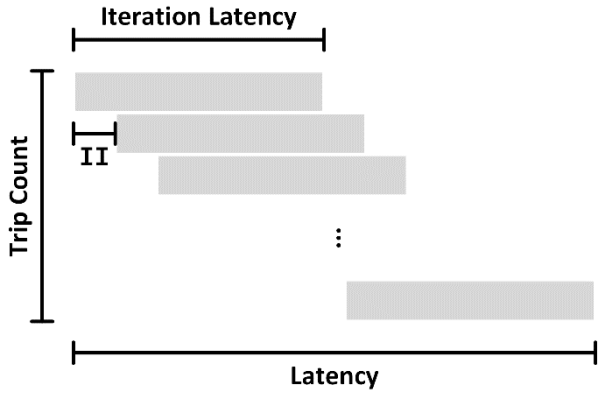
Modules & Loops	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined
dct	415	3.320E3	-	416	-	no
dct_Pipeline_RD_Loop_Row_RD_Loop_Col	66	528.000	-	66	-	no
RD_Loop_Row_RD_Loop_Col	64	512.000	2	1	64	yes
dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop	70	560.000	-	70	-	no
Row_DCT_Loop_DCT_Outer_Loop	68	544.000	6	1	64	yes
dct_Pipeline_Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop	66	528.000	-	66	-	no
Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop	64	512.000	2	1	64	yes
dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop	70	560.000	-	70	-	no
Col_DCT_Loop_DCT_Outer_Loop	68	544.000	6	1	64	yes

a、 The loop RD_Loop_Row_ RD_Loop_Col has a latency of 2 cycles for each iteration of the loop (iteration latency).

b、 A trip count of 64, representing the number of iterations, is obtained by flattening the two nested loops (8 x 8 = 64).

c、 The initiation interval (II), the number of clock cycles before the function can accept new input data, is 1. We can therefore know that the design is fully pipelined.

d、 The latency of RD_Loop_Row_ RD_Loop_Col can be calculated by leveraging the iteration latency, trip count, and interval. An interval of 64 (2 - 1 + 1 x (64 – 1)) is obtained. The illustration of the timing diagram for a pipelined design is shown below:



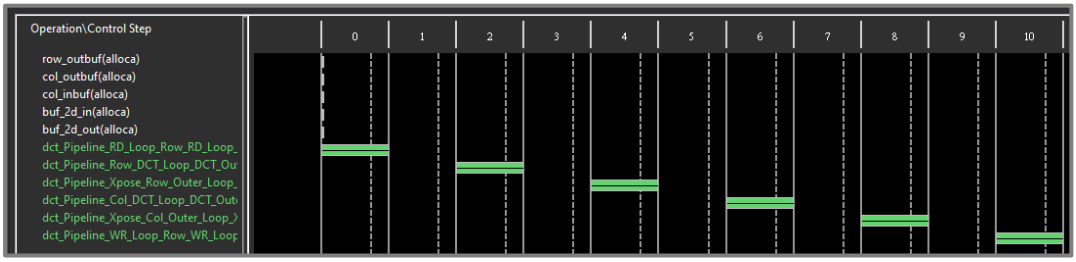
e、 It takes 1 clock to enter loop RD_Loop_Row_ RD_Loop_Col and 1 clock cycle to return. The iteration latency for RD_Loop_Row_ RD_Loop_Col is therefore (1 + 64 + 1) 66 clock cycles.

f、 The total loop latency is 8 x 66 = 528 ns.

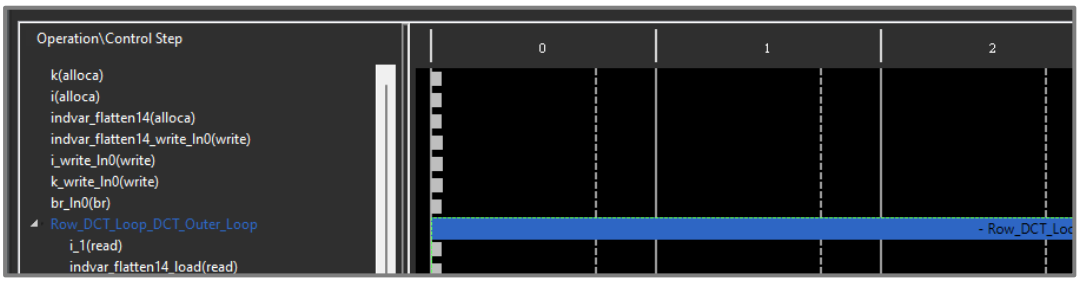
The total latency is the summation of all the loop blocks plus a clock cycle to enter each block.

4. Schedule Viewer

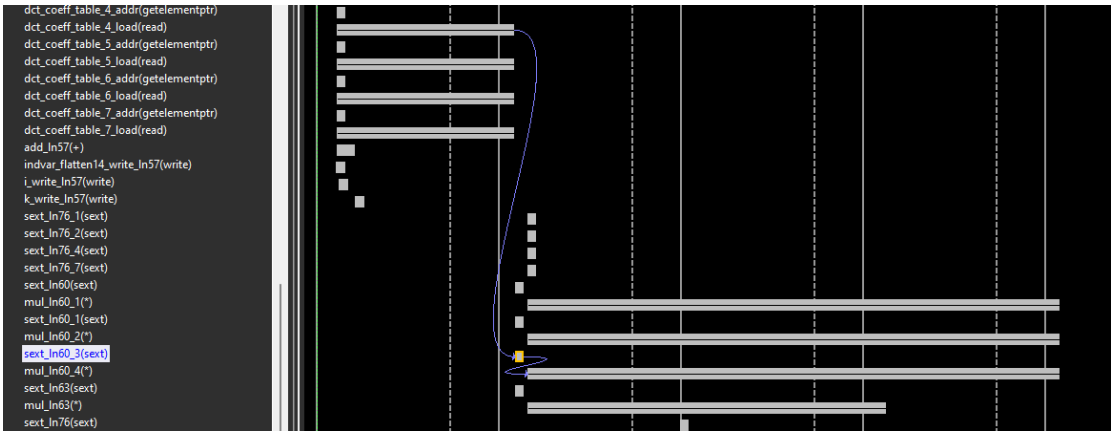
It provides a detailed view of the synthesized RTL, showing each operation and control step of the function and the clock cycle. It helps identify any loop dependencies that are preventing parallelism, timing violations, and data dependencies.



In the schedule viewer of each submodule, the top horizontal axis shows the clock cycles, and the vertical dashed line indicates the clock uncertainty region.



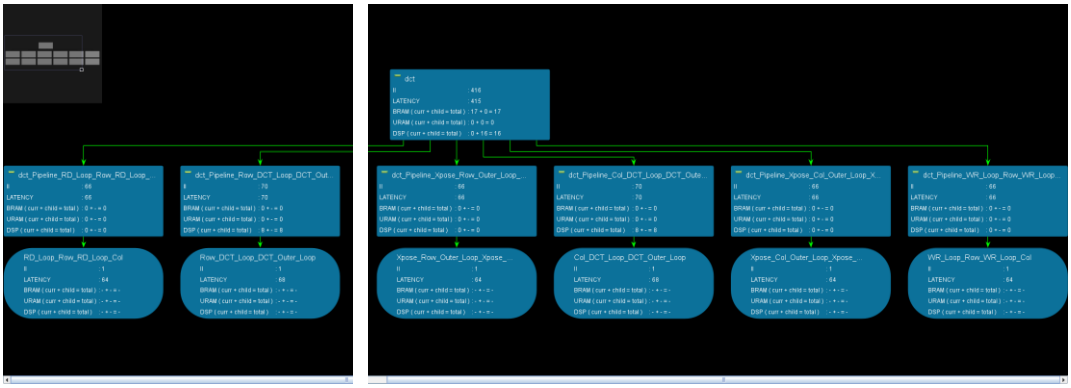
Take the dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop module for example, we can see that the main computation (multiplication of the data and the coefficients) takes roughly three cycles. The data dependency shows that the coefficient needs to be loaded and selected.



5. Function Call Graph Viewer

Displays the complete design after C synthesis or C/RTL co-simulation to show the throughput of the design in terms of latency and `II`.

In this design, we can see that all five submodules have a `II` of 1, which means that the design is fully pipelined. The main computing modules (dct_Pipeline_Row_DCT_Loop_DCT_Outer_Loop and dct_Pipeline_Col_DCT_Loop_DCT_Outer_Loop) has 8 DSPs, while others have zero. This makes sense because only these two modules involve real computation.



We can also use the heat map feature (on the top of the graph) to highlight several metrics of interest, including `II`, latency, and stalling time percentage.

6. Comparison with Non-Pipelined Design

In this part, the automatic pipeline mechanism is turned off by adding this pragma:

```
#pragma HLS pipeline off
```

After running C synthesis, we can see that the design is no longer pipelined. In addition, the loop is not unrolled and flattened.

Modules & Loops	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined
dct	5990	4.792E4	-	5991	-	no
RD_Loop_Row	144	1.152E3	18	-	8	no
RD_Loop_Col	16	128.000	2	-	8	no
Row_DCT_Loop	2704	2.163E4	338	-	8	no
DCT_Outer_Loop	336	2.688E3	42	-	8	no
DCT_Inner_Loop	40	320.000	5	-	8	no
Xpose_Row_Outer_Loop	144	1.152E3	18	-	8	no
Xpose_Row_Inner_Loop	16	128.000	2	-	8	no
Col_DCT_Loop	2704	2.163E4	338	-	8	no
DCT_Outer_Loop	336	2.688E3	42	-	8	no
DCT_Inner_Loop	40	320.000	5	-	8	no
Xpose_Col_Outer_Loop	144	1.152E3	18	-	8	no
Xpose_Col_Inner_Loop	16	128.000	2	-	8	no
WR_Loop_Row	144	1.152E3	18	-	8	no
WR_Loop_Col	16	128.000	2	-	8	no

We analyze the latency of RD_Loop_Row for example:

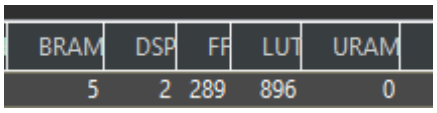
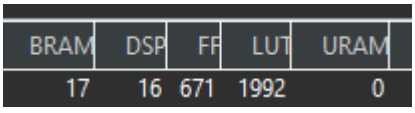
a、 Sub-loop RD_Loop_Col has a latency of 2 cycles for each loop iteration and a trip count of 8: 2 x 8 = 16 clock cycles total latency for the loop.

b、 From RD_Loop_Row, it takes 1 clock to enter loop RD_Loop_Col and 1 clock cycle to return to RD_Loop_Row. The iteration latency for RD_Loop_Row is therefore (1 + 16 + 1) 18 clock cycles.

c、 RD_Loop_Row has a trip count of 8, so the total loop latency is 8 x 18 = 144 clock cycles.

The total latency is the summation of all the loop blocks plus a clock cycle to enter each block. A derived latency of 5990 cycles is **14.43x longer** than the pipelined design (415 cycles).

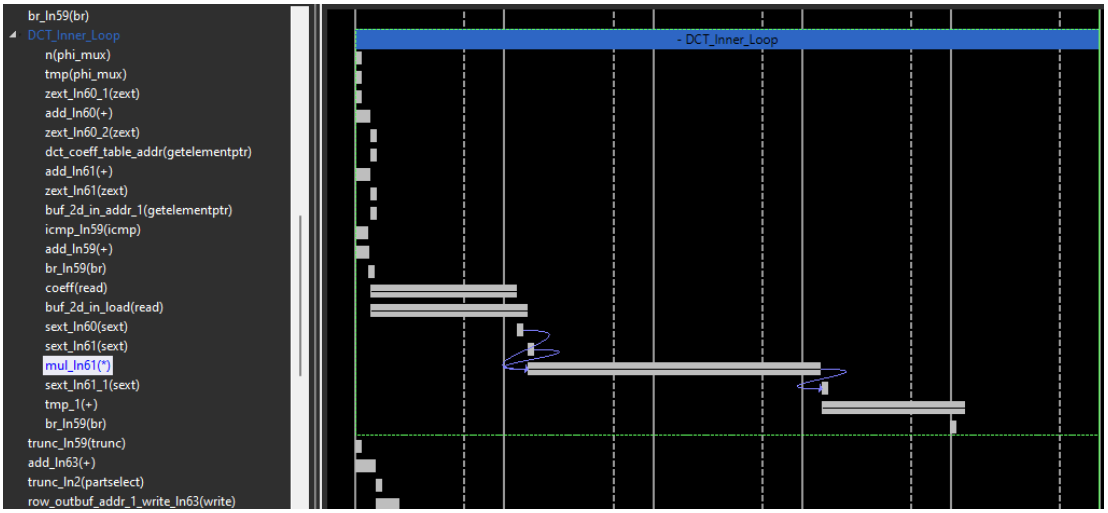
However, the hardware resources are smaller for the non-pipelined solution. Note that the pipelining method increases the parallelism of hardware. The larger parallelism increases memory number/bandwidth (BRAM), DSP modules, and LUT.

Non-Pipelined	Pipelined
	

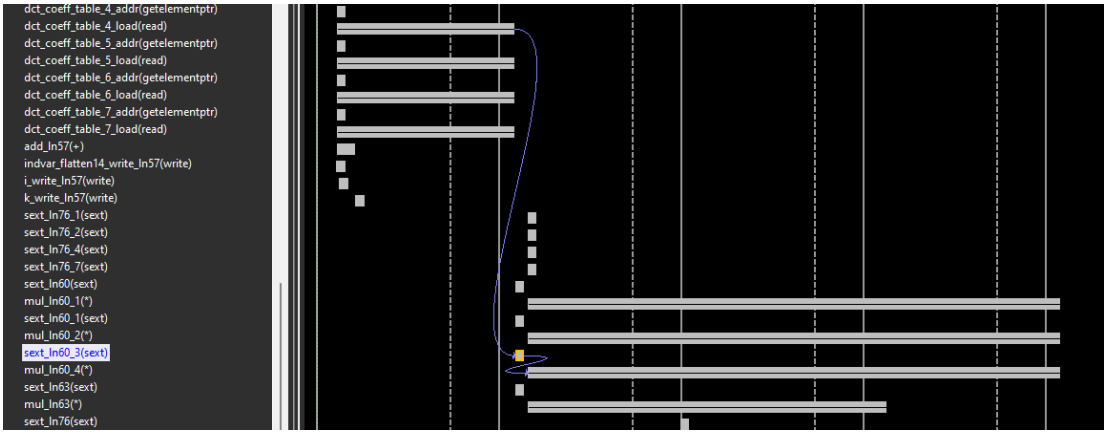
7. Memory Partition

Continued from the previous part. We further analyze memory partition using the schedule viewer. We already know that BRAM increases from 5 to 17 when performing data pipelining. Comparing the DCT_Inner_Loop part in the schedule viewer:

Non-Pipelined:



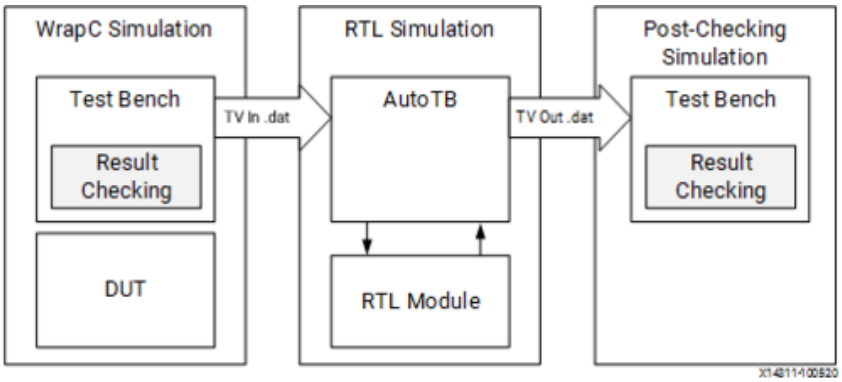
Pipelined:



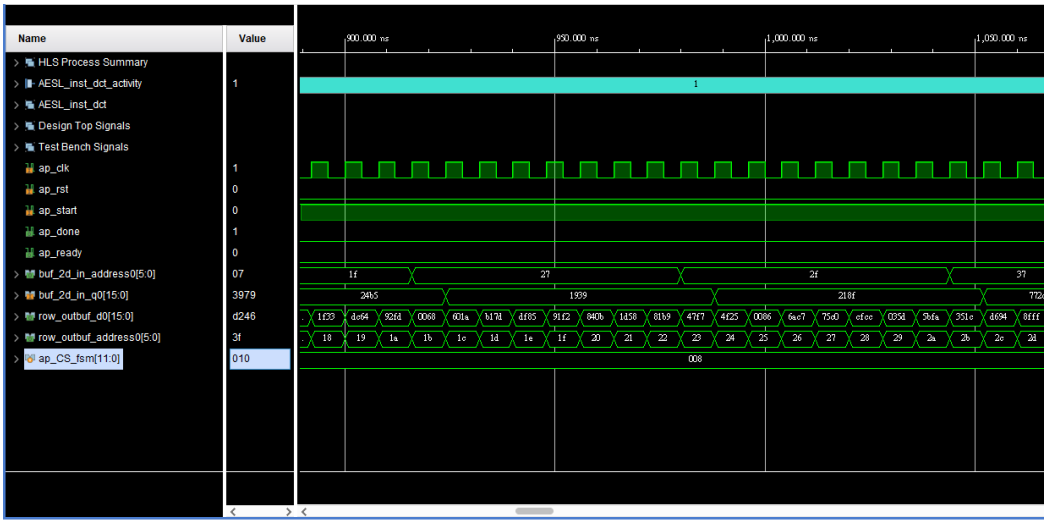
By comparing the two solutions, we can see that **memory partition** helps increase memory bandwidth, so more data can be accessed in each cycle. This facilitates hardware pipelining.

8. C/RTL Co-Simulation

C/RTL co-simulation uses a C test bench, running the main() function, to automatically verify the RTL design running in behavioral simulation. The C/RTL verification process consists of three phases:



By opening the waveform viewer, we can analyze the operations and the dataflow in a cycle-accurate manner:


Name
Value
HLS Process Summary
AESL_inst_dct_activity
1
AESL_inst_dct
Design Top Signals
Test Bench Signals
ap_clk
1
ap_rst
0
ap_start
0
ap_done
1
ap_ready
0
buf_2d_in_address0[5:0]
07
buf_2d_in_q0[15:0]
3979
row_outbuf_d0[15:0]
d246
row_outbuf_address0[5:0]
3f
ap_CS_fsm[11:0]
010
900.000 ns
950.000 ns
1,000.000 ns
1,050.000 ns
1
1f
27
2f
37
24b5
1939
218f
1f33
dc64
926d
0068
601a
b171
df85
91f2
840b
1d58
81b9
47f7
4f25
0086
6ac7
75e0
cfcc
0958
5bfa
351c
d694
8fff
18
19
1a
1b
1c
1d
1e
1f
20
21
22
23
24
25
26
27
28
29
2a
2b
2c
2d
008