Optimizing Arrays for Performance Demo Script

Introduction

This demonstration script provides high-level instructions on how to remove bottlenecks caused by arrays in a C design.

Preparation:

- Required files: Necessary files are located at *C:\training\optimize_array_performance\demo*
- Required hardware: None
- Supporting materials: None

Optimizing Arrays for Performance

Action with Description	Point of Emphasis and Key Takeaway
• Launch the Vivado® HLS tool. • Open the provided **dct_prj** Vivado HLS tool project located at: *C:\training\optimize_array_performance\demo\dct_prj*	You can open existing Vivado HLS tool projects from the Vivado HLS tool Welcome page.

© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
• Access and review the source files (*dct.c* and *dct.h*) from the Explorer pane.	This C design uses a discrete cosine transformation (DCT). The function implements a 2D DCT algorithm by first processing each row of the input array via a 1D DCT, then processing the columns of the resulting array through the same 1D DCT. It calls the *read_data*, *dct_2d*, and *write_data* functions. The *read_data* function is defined at line 54 and consists of two loops: **RD_Loop_Row** and **RD_Loop_Col**. The *write_data* function is defined at line 66 and consists of two loops to perform writing the result. The *dct_2d* function, defined at line 23, calls the *dct_1d* function and performs transpose. Finally, the *dct_1d* function, defined at line 4, uses *dct_coeff_table* and performs the required function by implementing a basic iterative form of the 1D Type II DCT algorithm.

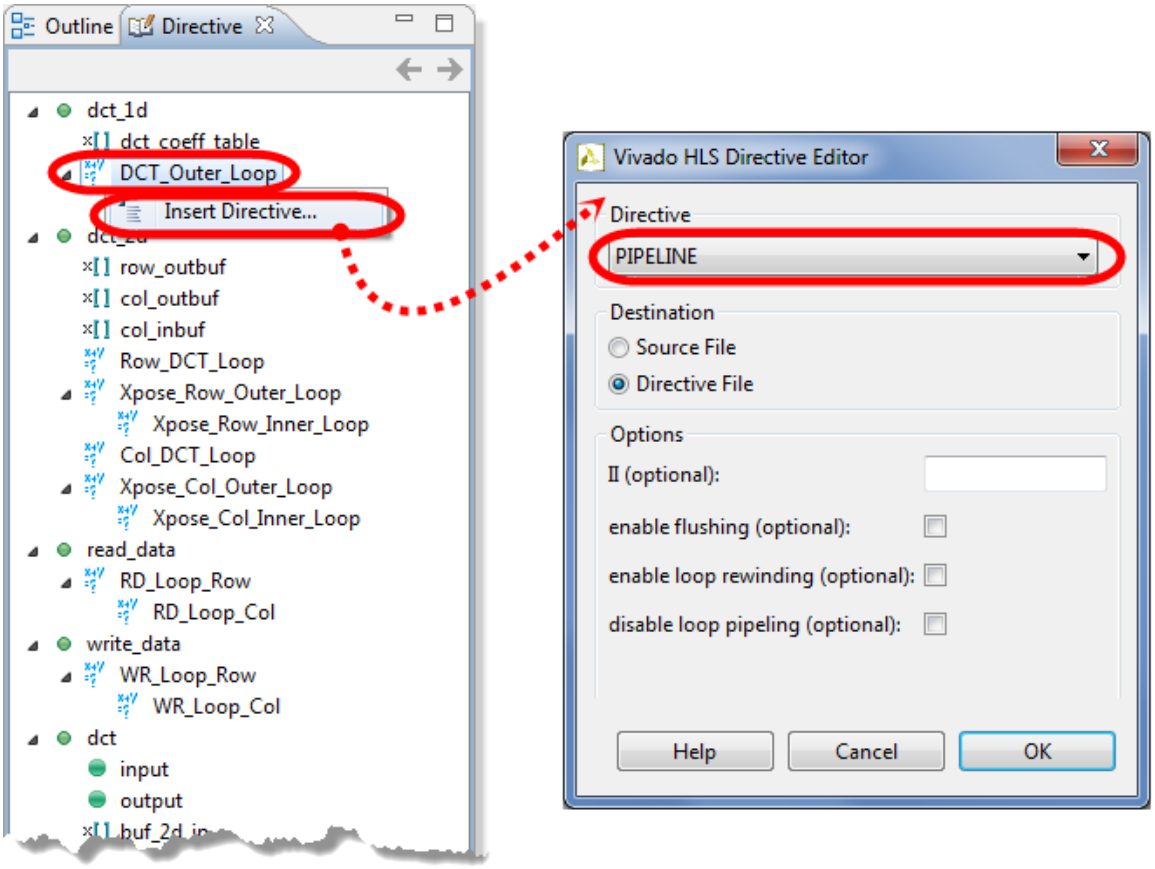
© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
• Run C synthesis. • Review the Synthesis report.	Once synthesis completes, the Synthesis report will open in the main viewing area. Notice that the estimated clock period is within the requested clock period. The Synthesis report also contains latency and throughput information of the design. The results correspond to the default solution (without any directives). You can further reduce the numbers down by specifying your requirements via directives. The Synthesis log is available in the Console pane.
What is the worst-case latency of the design? Answer: 2935	
• Create a new solution named *solution2* (select **Project** > **New Solution**). • Accept the default settings and click **Finish**.	Creating a new solution allows for different optimizations to be compared easily. There is no need to copy directives from the previous solution because the previous solution does not have any directives. Even if the directives are copied (the default setting), there would be no impact to the demo since there are no directives in the initial solution.
As part of the optimization process, as seen in the "Pipeline for Performance" demo, you will begin by pipelining the loops in the design.	

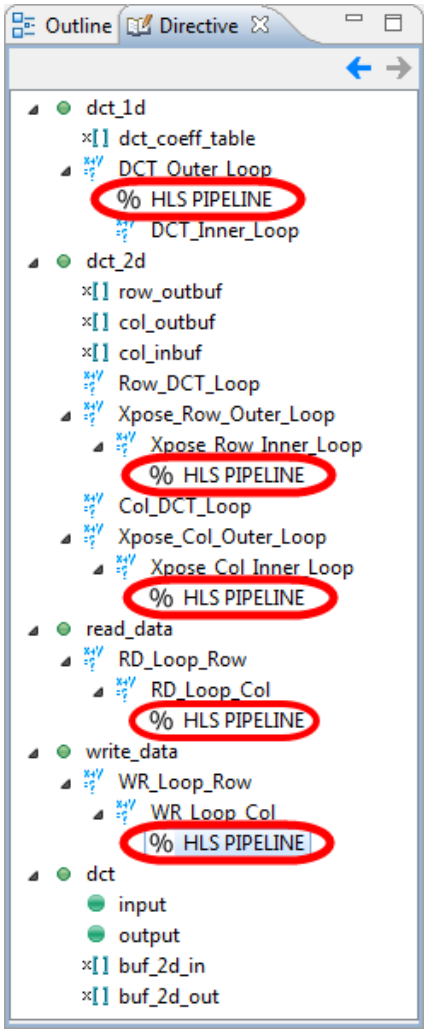
© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
• Apply the **PIPELINE** directive on *DCT_Outer_Loop* of the *dct_1d* function (shown below).	Apply the PIPELINE directive on the outer loop. Moving the PIPELINE directive from the inner loop to the outer loop of *dct_1d* will lead to more parallelism of the multiply and add operations. That is, eight (8) multiply and add operations are performed concurrently, thus minimizing the number of cycles required to compute each value in the output array. Leave the II field blank since the design tries to target II as 1; i.e., it will try to optimize the loop to accept a new input for every cycle. You will find the directive written to the *directive.tcl* file under the *dct_prj* > *solution2* > *constraints* folder. **`set_directive_pipeline "dct_1d/DCT_Outer_Loop"`**

© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
	

© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
• Similarly, apply the **PIPELINE** directive to the following loops: • *Xpose_Row_Inner_Loop* of the *dct_2d* function • *Xpose_Col_Inner_Loop* of the *dct_2d* function • *RD_Loop_Col* of the *read_data* function • *WR_Loop_Col* of the *write_data* function	The Directive tab should look like the figure below after you finish applying the PIPELINE directive. 
• Run C synthesis.	Once synthesis completes, the Synthesis report will open in the main viewing area.
• Compare the results of two solutions (*solution1* and *solution2*).	This allows you to compare the different optimizations of the project. You should see the comparison report as shown below.

© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
Performance Estimates **Timing (ns)** Clock solution2 solution1 ap_clk Target 10.00 10.00 Estimated 7.769 3.770 **Latency (clock cycles)** solution2 solution1 Latency min 843 2935 max 843 2935 Interval min 843 2935 max 843 2935 **Utilization Estimates** solution2 solution1 BRAM_18K 5 5 DSP48E 8 1 FF 491 246 LUT 1364 964 URAM 0 0	
• What is the worst-case latency of the design?	Answer: 843
• Go to the Utilization Estimates section and note the number of DSP48E and block RAMs used to implement *solution2*.	Answer: Number of BRAM_18K: 5 Number of DSP48E: 8

© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
• Select the **Console** tab and review the synthesis information.	From the Synthesis log, note that the design was not able to achieve the requested II on *DCT_Outer_Loop* because of the limited memory ports on the *src* element. The *src* is input to the *dct_1d* function and *dct_1d* is called twice in the *dct_2d* function (line 33 and line 44). At line 33, *in_block* is accessed via the *src* element in *dct_1d*. At line 44, *col_inbuf* is accessed via the *src* element in *dct_1d*. Therefore, you will need to partition both the *col_inbuf* and *in_block* arrays to achieve a throughput of 1.

© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway

```
Console   Errors   Warnings
Vivado HLS Console
INFO: [HLS 200-10] ----------------------------------------------------------
INFO: [HLS 200-10] -- Implementing module 'dct_1d2'
INFO: [HLS 200-10] ----------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining loop 'DCT_Outer_Loop'.
WARNING: [SCHED 204-69] Unable to schedule 'load' operation ('src_load_5', dct.c:17) on array 'src' due to limited memory ports.
INFO: [SCHED 204-61] Pipelining result: Target II: 1, Final II: 4, Depth: 8.
WARNING: [SCHED 204-21] Estimated clock period (5.79ns) exceeds the target (target clock period: 6ns, clock uncertainty: 0.75ns, effe
WARNING: [SCHED 204-21] The critical path consists of the following:
        'mul' operation ('tmp_7_7', dct.c:17) (2.84 ns)
        'add' operation ('tmp7', dct.c:19) (2.95 ns)
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111]  Elapsed time: 5.262 seconds; current allocated memory: 93.807 MB.
INFO: [BIND 205-100] Starting micro-architecture generation ...
INFO: [BIND 205-101] Performing variable lifetime analysis.
INFO: [BIND 205-101] Exploring resource sharing.
INFO: [BIND 205-101] Binding ...
INFO: [BIND 205-100] Finished micro-architecture generation.
INFO: [HLS 200-111]  Elapsed time: 0.066 seconds; current allocated memory: 94.185 MB.
INFO: [HLS 200-10] ----------------------------------------------------------
INFO: [HLS 200-10] -- Implementing module 'dct_2d'
INFO: [HLS 200-10] ----------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining loop 'Xpose_Row_Outer_Loop_Xpose_Row_Inner_Loop'.
INFO: [SCHED 204-61] Pipelining result: Target II: 1, Final II: 1, Depth: 4.
INFO: [SCHED 204-61] Pipelining loop 'Xpose_Col_Outer_Loop_Xpose_Col_Inner_Loop'.
INFO: [SCHED 204-61] Pipelining result: Target II: 1, Final II: 1, Depth: 4.
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111]  Elapsed time: 0.097 seconds; current allocated memory: 94.489 MB.
INFO: [BIND 205-100] Starting micro-architecture generation ...
INFO: [BIND 205-101] Performing variable lifetime analysis.
INFO: [BIND 205-101] Exploring resource sharing.
INFO: [BIND 205-101] Binding ...
INFO: [BIND 205-100] Finished micro-architecture generation.
INFO: [HLS 200-111]  Elapsed time: 0.087 seconds; current allocated memory: 94.807 MB.
INFO: [HLS 200-10] ----------------------------------------------------------
INFO: [HLS 200-10] -- Implementing module 'dct'
INFO: [HLS 200-10] ----------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining loop 'RD_Loop_Row_RD_Loop_Col'.
INFO: [SCHED 204-61] Pipelining result: Target II: 1, Final II: 1, Depth: 4.
INFO: [SCHED 204-61] Pipelining loop 'WR_Loop_Row_WR_Loop_Col'.
INFO: [SCHED 204-61] Pipelining result: Target II: 1, Final II: 1, Depth: 4.
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111]  Elapsed time: 0.113 seconds; current allocated memory: 95.060 MB.
```

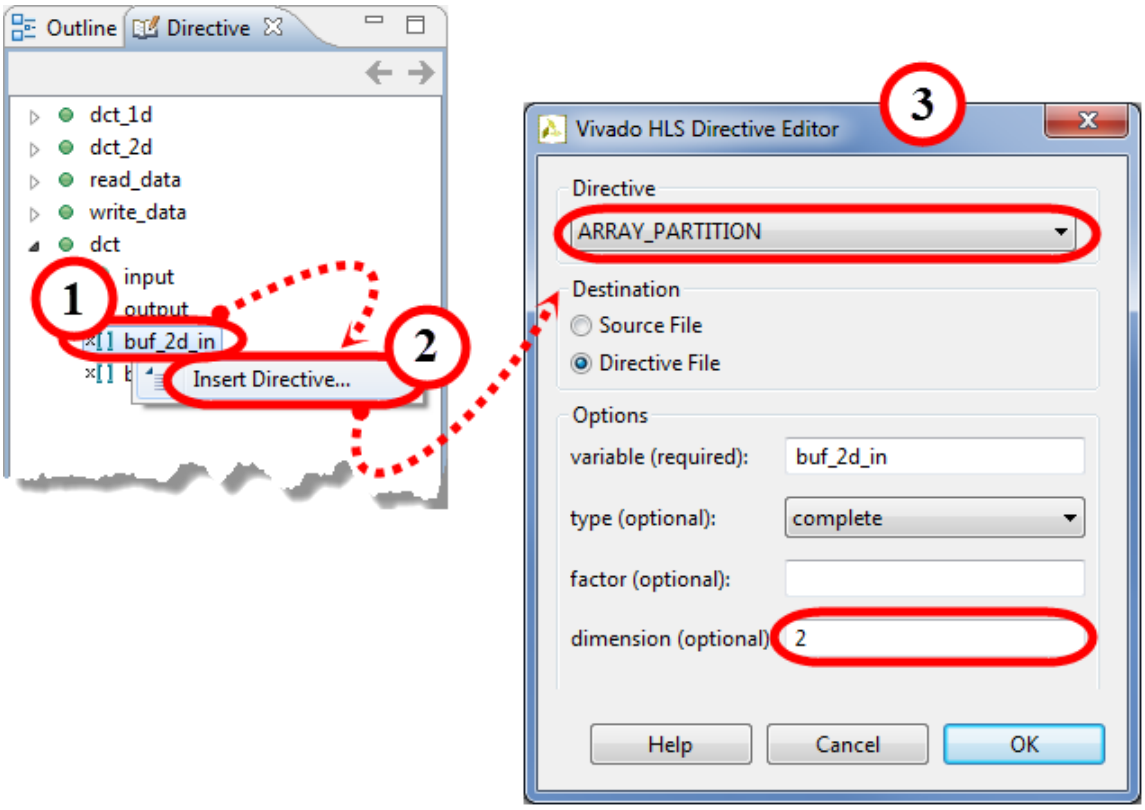
You will now solve the *dct_1d* pipeline II problem by increasing the memory bandwidth available to it.

This will be done by partitioning the arrays from which the *dct_1d* inner loops read data (*in_block* in *Row_DCT_Loop* and *col_inbuf* in *Col_DCT_Loop*).

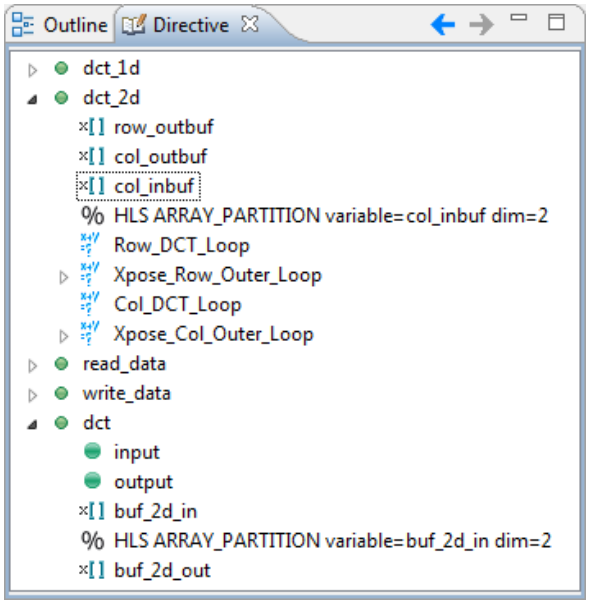
• Create a new solution named *solution3*. • Accept the default settings and click **Finish**.	In this solution, you will apply the ARRAY_PARTITION directive to *buf_2d_in* of the *dct* function and *col_inbuf* of the *dct2d* function.

© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
• Apply the **ARRAY_PARTITION** directive to *buf_2d_in* of the *dct* function as shown in the figure below.	Partitioning large arrays into multiple smaller arrays or into individual registers can help improve access to data and remove block RAM bottlenecks.



© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
• Similarly, apply the **ARRAY_PARTITION** directive *col_inbuf* of the *dct_2d* function.	The Directive tab should look like the figure below after you finish applying the ARRAY_PARTITION directive. 
• Run C synthesis.	Once synthesis completes, the synthesis report will open in the main viewing area.
Examine the Synthesis log. Has the PIPELINE II directive been met? Yes, the pipeline directive met II=1. The memory bandwidth increased via the array partitioning and thus the design met the requested II value.	
• Compare the results of the two solutions (*solution2* and *solution3*).	You should see the comparison report as shown below.

© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway

Performance Estimates

Timing (ns)

Clock		solution3	solution2
ap_clk	Target	10.00	10.00
	Estimated	7.517	7.769

Latency (clock cycles)

		solution3	solution2
Latency	min	477	843
	max	477	843
Interval	min	477	843
	max	477	843

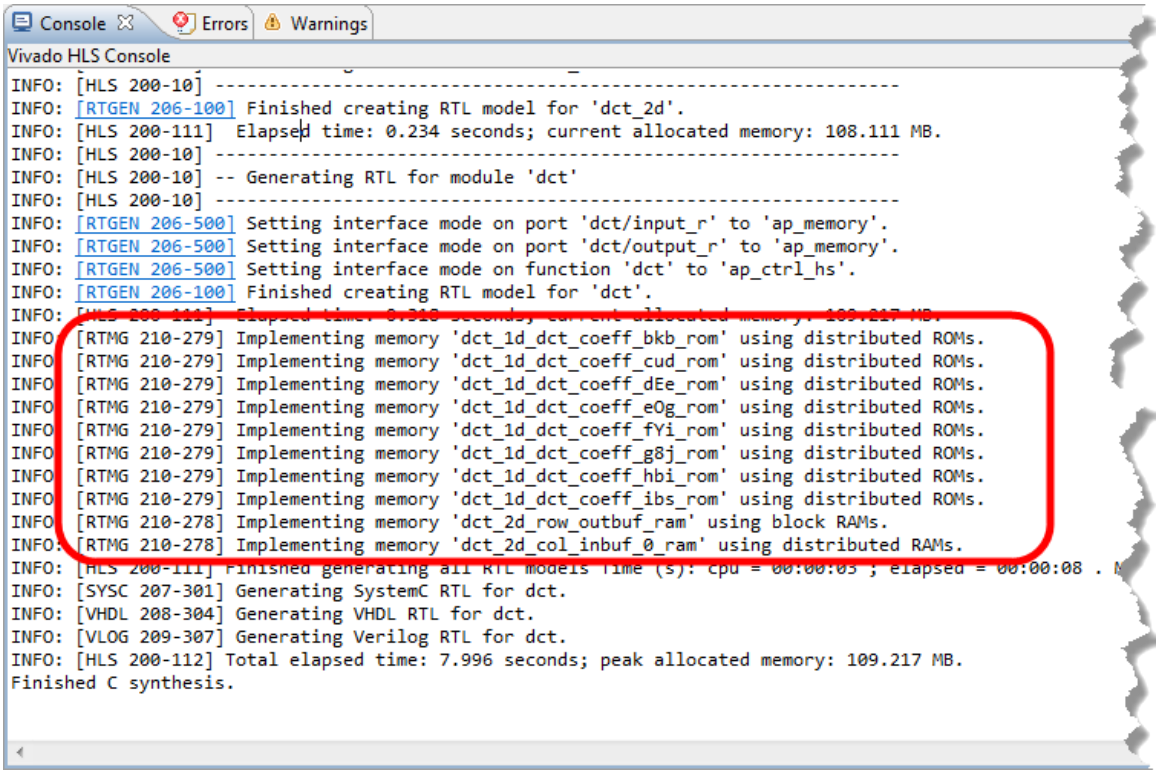
Utilization Estimates

	solution3	solution2
BRAM_18K	3	5
DSP48E	8	8
FF	844	491
LUT	1879	1364
URAM	0	0

Latency was reduced to 477. Block RAM usage decreased to 3 from 5.

Memory utilization will usually be more after array partitioning. But in this case, some of the memory elements were implemented in the distributed RAMs/ROMS (you can observe this in the Synthesis log in the Console tab) and hence the number of block RAMs was actually reduced compared to the previous solution.

© Copyright 2018 Xilinx

Action with Description	Point of Emphasis and Key Takeaway
	

```
Console   Errors   Warnings
Vivado HLS Console
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [RTGEN 206-100] Finished creating RTL model for 'dct_2d'.
INFO: [HLS 200-111]  Elapsed time: 0.234 seconds; current allocated memory: 108.111 MB.
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-10] -- Generating RTL for module 'dct'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [RTGEN 206-500] Setting interface mode on port 'dct/input_r' to 'ap_memory'.
INFO: [RTGEN 206-500] Setting interface mode on port 'dct/output_r' to 'ap_memory'.
INFO: [RTGEN 206-500] Setting interface mode on function 'dct' to 'ap_ctrl_hs'.
INFO: [RTGEN 206-100] Finished creating RTL model for 'dct'.
INFO: [HLS 200-111]  Elapsed time: 0.218 seconds; current allocated memory: 109.217 MB.
INFO: [RTMG 210-279] Implementing memory 'dct_1d_dct_coeff_bkb_rom' using distributed ROMs.
INFO: [RTMG 210-279] Implementing memory 'dct_1d_dct_coeff_cud_rom' using distributed ROMs.
INFO: [RTMG 210-279] Implementing memory 'dct_1d_dct_coeff_dEe_rom' using distributed ROMs.
INFO: [RTMG 210-279] Implementing memory 'dct_1d_dct_coeff_eOg_rom' using distributed ROMs.
INFO: [RTMG 210-279] Implementing memory 'dct_1d_dct_coeff_fYi_rom' using distributed ROMs.
INFO: [RTMG 210-279] Implementing memory 'dct_1d_dct_coeff_g8j_rom' using distributed ROMs.
INFO: [RTMG 210-279] Implementing memory 'dct_1d_dct_coeff_hbi_rom' using distributed ROMs.
INFO: [RTMG 210-279] Implementing memory 'dct_1d_dct_coeff_ibs_rom' using distributed ROMs.
INFO: [RTMG 210-278] Implementing memory 'dct_2d_row_outbuf_ram' using block RAMs.
INFO: [RTMG 210-278] Implementing memory 'dct_2d_col_inbuf_0_ram' using distributed RAMs.
INFO: [HLS 200-111] Finished generating all RTL models Time (s): cpu = 00:00:03 ; elapsed = 00:00:08 . M
INFO: [SYSC 207-301] Generating SystemC RTL for dct.
INFO: [VHDL 208-304] Generating VHDL RTL for dct.
INFO: [VLOG 209-307] Generating Verilog RTL for dct.
INFO: [HLS 200-112] Total elapsed time: 7.996 seconds; peak allocated memory: 109.217 MB.
Finished C synthesis.
```

Summary

Memory elements may become bottlenecks when it comes to meeting throughput and latency requirements. In this demo, you learned how to apply a partition directive on arrays in the design and observed its impact on resources and throughput.

References:

- Supporting materials
  - *Vivado Design Suite Tutorial: High-Level Synthesis* (UG871)
  - *Vivado Design Suite User Guide: High-Level Synthesis* (UG902)

© Copyright 2018 Xilinx