Pipelining for Performance Demo Script

Introduction

This demonstration script provides high-level instructions on using the PIPELINE directive.

Preparation:

- Required files: Necessary files are located at *C:\training\hls\demos\loop_pipeline*
- Required hardware: None
- Supporting materials: None

Pipelining for Performance

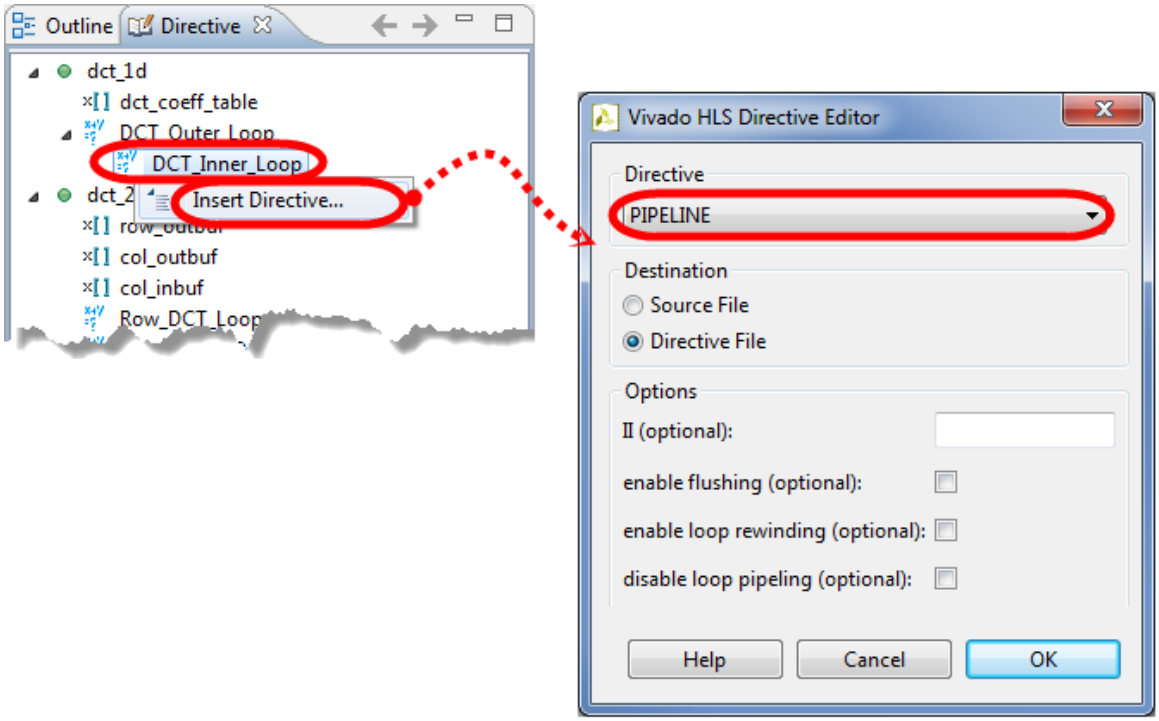
Action with Description	Point of Emphasis and Key Takeaway
• Launch the Vivado® HLS tool. • Open the provided **dct_prj** Vivado HLS tool project located at: *C:\training\hls\demos\loop_pipeline*	You can open existing Vivado HLS tool projects from the Vivado HLS tool Welcome page.

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.

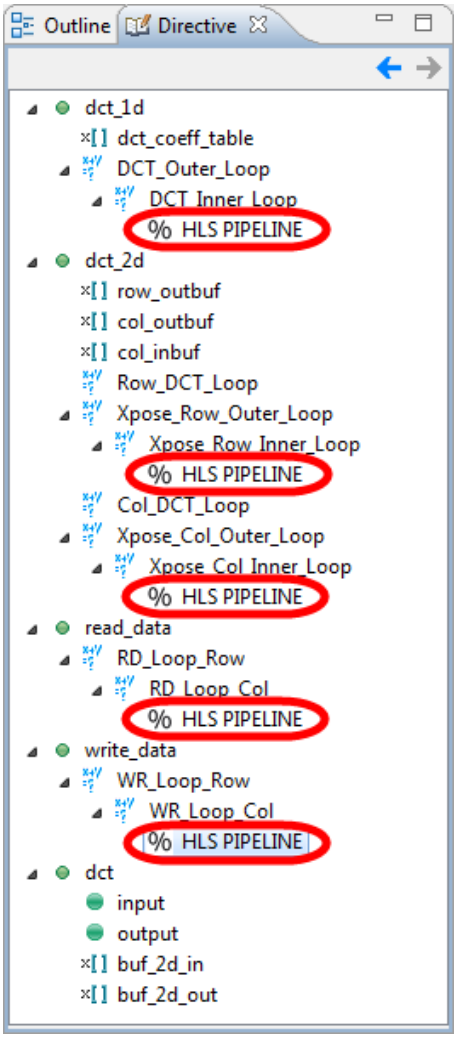
Action with Description	Point of Emphasis and Key Takeaway
• Run C synthesis. • Review the Synthesis report.	Once synthesis completes, the Synthesis report will open in the main viewing area. Notice that the estimated clock period is within the requested clock period. The Synthesis report also contains 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: 6647	
In the Synthesis report, go to the Utilization Estimates section and note the number of DSP48E and block RAMs used to implement the solution. Number of BRAM_18K: 5 Number of DSP48E: 1	
• Double-click **dct_dct_2d_csynth.rpt** to open the Synthesis report available under the *dct_prj* > *solution1* > *syn* > *report* folder in the Explorer pane.	What is the worst-case latency of the *dct_2d* function? Answer: 6100
• Similarly, open the **dct_dct_1d2_csynth.rpt** file under the *dct_prj* > *solution1* > *syn* > *report* folder in the Explorer pane.	What is the worst-case latency of the *dct_1d* function? Answer: 345

Action with Description	Point of Emphasis and Key Takeaway
• Create a new solution named *solution2*. • 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.
• Apply the **PIPELINE** directive on *DCT_Inner_Loop* of the *dct_1d* function (shown below).	Apply the PIPELINE directive on the inner loops. 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_Inner_Loop"`

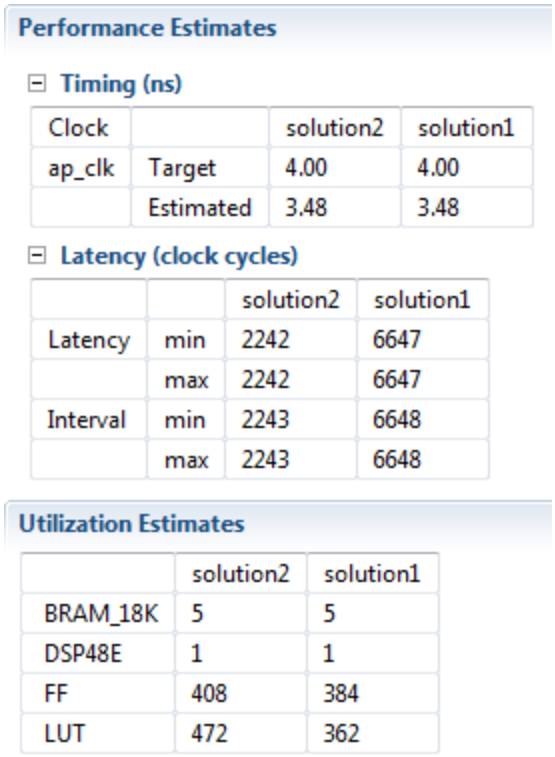
Action with Description	Point of Emphasis and Key Takeaway



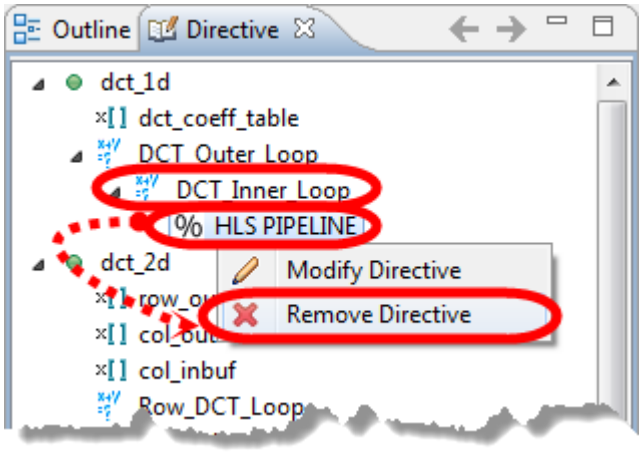
Action with Description	Point of Emphasis and Key Takeaway

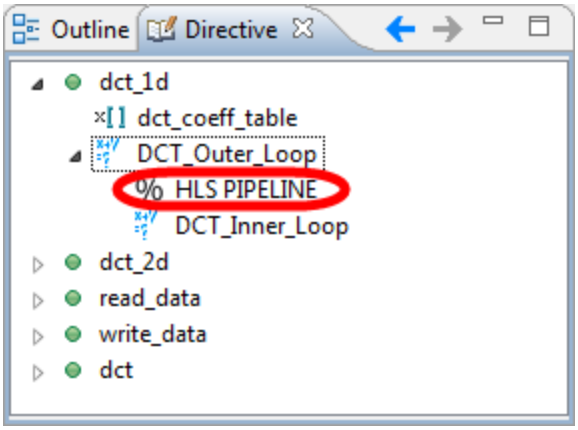
• Similarly, apply the **PIPELINE** directive to the following loops:
  • *Xpose_Row_Inner_Loop* of the *dct_2d* function
  • *Xpose_Col_Inner_Loop* of the *dct_2d* function
  • *RD_Loop_Col* of the *read_data* function
  • *WR_Loop_Col* of the *write_data* function
	The Directive tab should look like 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 the 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.

<table>
<tr><th>Action with Description</th><th>Point of Emphasis and Key Takeaway</th></tr>
<tr><td colspan="2">


</td></tr>
<tr><td>• What is the worst-case latency of the design?</td><td>Answer: 2242</td></tr>
<tr><td>• Go to the Utilization Estimates section and note the number of DSP48E and block RAMs used to implement solution2.</td><td>Answer:
Number of BRAM_18K: 5
Number of DSP48E: 1</td></tr>
<tr><td>• Double-click dct_dct_2d_csynth.rpt to open the Synthesis report available under the dct_prj > solution2 > syn > report folder in the Explorer pane.</td><td>What is the worst-case latency of the dct_2d function?
Answer: 2105</td></tr>
<tr><td>• Similarly, open the dct_dct_1d2_csynth.rpt file under the dct_prj > solution2 > syn > report folder in the Project Explorer pane.</td><td>What is the worst-case latency of the dct_1d function?
Answer: 121</td></tr>
</table>

Action with Description	Point of Emphasis and Key Takeaway
Note that the design latency and throughput (Interval) are improved with the application of the PIPELINE directive.	
• Create a new solution named *solution3*. • Accept the default settings and click **Finish**.	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.
• Delete the **PIPELINE** directive from *DCT_Inner_Loop* of the *dct_1d* function.	Delete the inner loop pipeline and apply the PIPELINE directive from the outer loop.
	

Action with Description	Point of Emphasis and Key Takeaway
• Apply the **PIPELINE** directive on *DCT_Outer_Loop* of the *dct_1d* function.	
• Run C synthesis.	Once synthesis completes, the Synthesis report will open in the main viewing area.
• Compare the results of the two solutions (*solution2* and *solution3*).	You should see the comparison report as shown below.

Performance Estimates

Timing (ns)

Clock		solution3	solution2
ap_clk	Target	4.00	4.00
	Estimated	3.48	3.48

Latency (clock cycles)

		solution3	solution2
Latency	min	946	2242
	max	946	2242
Interval	min	947	2243
	max	947	2243

Utilization Estimates

	solution3	solution2
BRAM_18K	5	5
DSP48E	8	1
FF	849	408
LUT	556	472

Action with Description	Point of Emphasis and Key Takeaway
Notice that throughput and latency are improved. With the PIPELINE directive applied on the outer loop, the inner loop automatically unrolls, which results in more resource utilization compared to *solution2*.	

Summary

The PIPELINE directive inserts pipeline registers in the generated RTL to improve the latency of the design. When it was applied on the outer loop, it automatically unrolled the inner loops and inserted the required pipeline registers in the design. By applying this directive properly, you can increase the throughput of the design.

In this demo, you learned how to apply the PIPELINE directive and observed its impact on a design.

References:

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