

# **Designing a JPEG DCT in C/C++ using Catapult Synthesis**

June 2017

---

**© 2015-17 Mentor Graphics Corporation  
All rights reserved.**

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**U.S. GOVERNMENT LICENSE RIGHTS:** The software and documentation were developed entirely at private expense and are commercial computer software and commercial computer software documentation within the meaning of the applicable acquisition regulations. Accordingly, pursuant to FAR 48 CFR 12.212 and DFARS 48 CFR 227.7202, use, duplication and disclosure by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in the license agreement provided with the software, except for provisions which are contrary to applicable mandatory federal laws.

**TRADEMARKS:** The trademarks, logos and service marks ("Marks") used herein are the property of Mentor Graphics Corporation or other parties. No one is permitted to use these Marks without the prior written consent of Mentor Graphics or the owner of the Mark, as applicable. The use herein of a third- party Mark is not an attempt to indicate Mentor Graphics as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A current list of Mentor Graphics' trademarks may be viewed at: [www.mentor.com/trademarks](http://www.mentor.com/trademarks).

The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

**End-User License Agreement:** You can print a copy of the End-User License Agreement from: [www.mentor.com/eula](http://www.mentor.com/eula).

Mentor Graphics Corporation  
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.  
Telephone: 503.685.7000  
Toll-Free Telephone: 800.592.2210  
Website: [www.mentor.com](http://www.mentor.com)  
SupportNet: [supportnet.mentor.com/](http://supportnet.mentor.com/)

## Introduction

NOTE: This training material assumes that the reader has taken the Catapult Basic Training. Many of the Basic Training concepts are discussed and used in this document without giving detailed background information.

This class covers the design and optimization of a 2-D Discrete Cosine Transform (DCT), which is one of the functional units in a JPEG pixel-pipe image compression engine. The pixel-pipe engine takes a stream of RGB image data and compresses it into a bit stream.

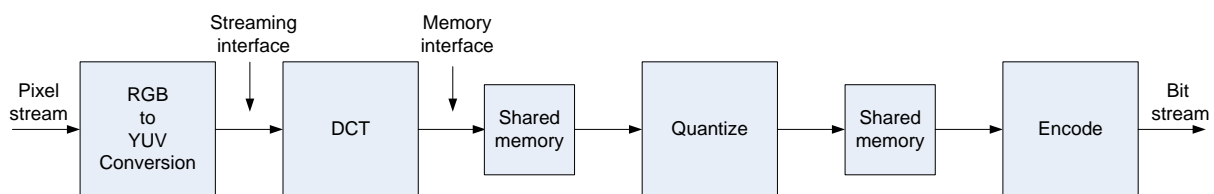


Figure 1 Pixel-pipe Hardware Architecture

The 2-D DCT algorithm transforms an 8x8 array of image pixels into an 8x8 array of spatial frequency values. This is done as a two-step process using two 1-D filters and some storage to store the intermediate calculations. The 1-D filters access of the intermediate array data is in column and row order respectively. Because of this the vertical processing filter cannot start until the horizontal processing filter has completed. This is an important aspect of the algorithm to understand in order to architect a high-performance hardware implementation

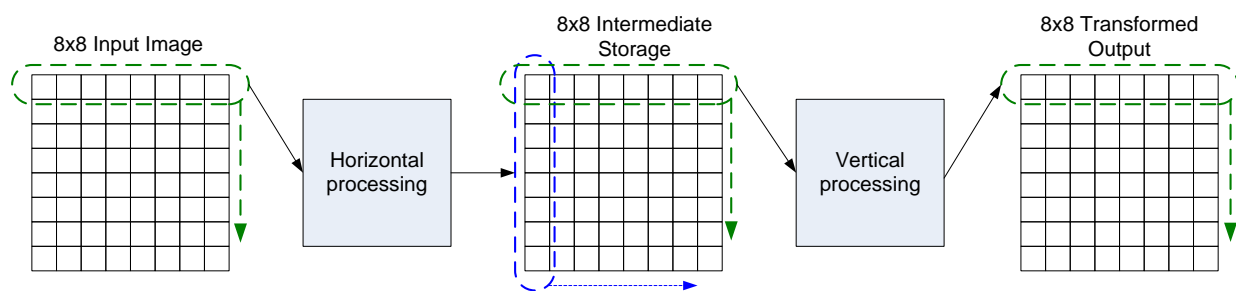


Figure 2 2-D DCT Algorithm

The design goals of this class are:

- Make the original design synthesizable with a streaming input interface
- Analyze the design in Catapult to find bit-widths that can be optimized
- Quantize the design using bit-accurate data types to reduce area

- See the effects of loop pipelining/loop flattening to achieve a design throughput of under 1200 cycles
- Use the Catapult automated verification flow to verify the design throughput and functionality
- Understand pipelined scheduling failures due to memory resource competition
- Make an architectural code change to pass scheduling and achieve a throughput of 128 cycles
- Make an architectural code change to reduce area
- Use Catapult "Hierarchy" to build a high-performance multi-block DCT with a throughput of 64 cycles
- Use Catapult CCOREs to reduce area

## Algorithmic DCT

The starting point for a Catapult C design is often a piece of C++ code that was written by an algorithm or systems engineer. This code is often not suitable for C++ synthesis since it may contain non-synthesizable constructs such as new/malloc or certain types of pointer operations that are not suitable for synthesis. Furthermore, even when synthesizable, the C++ source may not have the architectural details that are sufficient to achieve the desired interfaces, performance, or area goals.

## Compile the Design

- CD into DCT\_Algorithm directory and launch Catapult by typing “catapult” at the command prompt (This step can be skipped if running the toolkit interactively).
- Go to File > Run Script and run the directives1.tcl file. If running interactively select the “Compile the original design” script and click on “Launch Project” in the toolkit window. This will attempt to compile and synthesize the design.
- Look in the Catapult transcript window and note that there is a compilation error indicating that pointers-to-pointers is not supported on the design interface.

#	Message
#	Found top design routine 'dct' specified by directive
#	Synthesizing routine 'dct'
✖	Unsupported synthesis feature 'pointers-to-pointers on the interface'
✖	Compilation aborted
ℹ	Completed transformation 'analyze' on solution 'solution.v1': elapsed ti
#	> end dofile /scratch1/sb8dbg/main/ixl/Mgc_home/pkgs/ccs_toolkits_inhous
✖	go compile: Failed compile

## Error Analysis

- Double-click on the error message to cross-probe to the C++ source
- Look at the design interface and note that the interface uses arrays or pointers for the “input” and “output” arrays. There are two problems with this implementation, one is that arrays of pointers, or pointers to pointers, is not supported. The second problems is that this “algorithmic” implementation does not match the interface requirements of the pixel-pipe architecture, which needs a streaming interface for the DCT input.

```
// -----  
// DCT 2D 8x8 Top  
void dct(char *input[8], short *output[8]) {  
    const int coeff[8][8] = {
```