

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.

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.

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
SupportNet: supportnet.mentor.com/

- Go to the next toolkit example

Adding ac_channel Streaming Input Interface

This Toolkit converts “input” into an ac_channel streaming interface and makes “output” an explicit array by defining both array dimensions.

Synthesize the Design

- CD into the DCT_channelized 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 “Synthesize the design” script and click on “Launch Project” in the toolkit window. This sets the technology to Sample 65nm, adds single-port RAM libraries, sets the clock to 300 MHz, and synthesizes the unconstrained design.
- Open the “dct.cpp” file a look at the design interface.
 - The interface has been re-written so that “input” is an ac_channel interface of type “char”.
 - “output” has been explicitly defined as an 8x8 array.
 - A copy loop has been added to copy one row of the input image into a local storage buffer “buf[8]”. The algorithm then operates on the local array “buf”.

```
void dct(ac_channel<char> &input, short output[8][8]) {
    const int coeff[8][8] = {
        {362, 362, 362, 362, 362, 362, 362, 362},
        {502, 425, 284, 99, -99, -284, -425, -502},
        {473, 195, -195, -473, -473, -195, 195, 473},
        {425, -99, -502, -284, 284, 502, 99, -425},
        {362, -362, -362, 362, 362, -362, -362, 362},
        {284, -502, 99, 425, -425, -99, 502, -284},
        {195, -473, 473, -195, -195, 473, -473, 195},
        { 99, -284, 425, -502, 502, -425, 284, -99}
    };

    int mem[8][8];
    int buf[8]; //Local storage for one row of input
    int acc;

    ROW0:for (int i=0; i < 8; ++i){
        COPY_ROW:for(int p=0; p<8; p++)//Copy one row of input into local storage
            buf[p] = input.read();
        COL0:for (int j=0; j < 8; ++j ) {
```

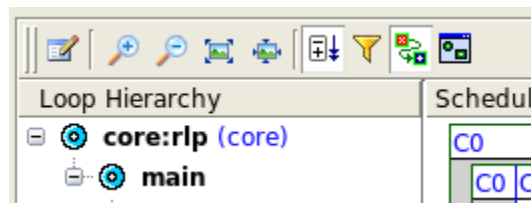
Architectural Constraints View

- Click on “Architecture” in the task bar to open the Architectural Constraints view.
- Expand the interface and arrays folder.
- Note that “input” has been auto-mapped to a “wait” handshaking interface and the “output” and “mem” arrays have been mapped to single-port RAM.

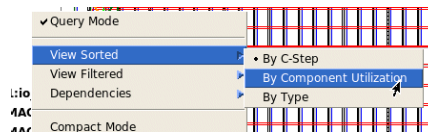
One of the things you may notice while looking at the `dct.cpp` file is that the internal variables are all defined as type `int`. While it is ok to do this for constant arrays and loop variables in a bounded loop, it is not a good idea for internal arrays and some variables. Usually the algorithm should be “quantized” to use bit-accurate data types if Catapult cannot automatically reduce the internal bit-widths. The two places to analyze bit-widths are the Gantt chart and the RTL report file.

Scheduling and Analysis

- Click on “Schedule” in the task bar and expand all the loops in the Gantt chart by clicking on the “down arrow” icon.



- Right-click in the Gantt chart and select “View Sorted > Component Utilization”



- The component utilization view shows the input bit-widths of the schedule operations. For a MUX this is the first number for the listed operation names. For adders and multipliers the first 4 numbers indicate the width and signedness of the “A” and “B” inputs of the operation. Look at the bit-widths of the multiplexors and the adders and note that they are 32-bits. If you see 32-bits or greater it’s usually something that should be investigated.

