Ain Shams University
Cairo, Egypt
Faculty of Engineering
(ECE dept.)

Integrated Circuits Lab

1999-Y2K

A Low Power 1-D DCT/IDCT Core

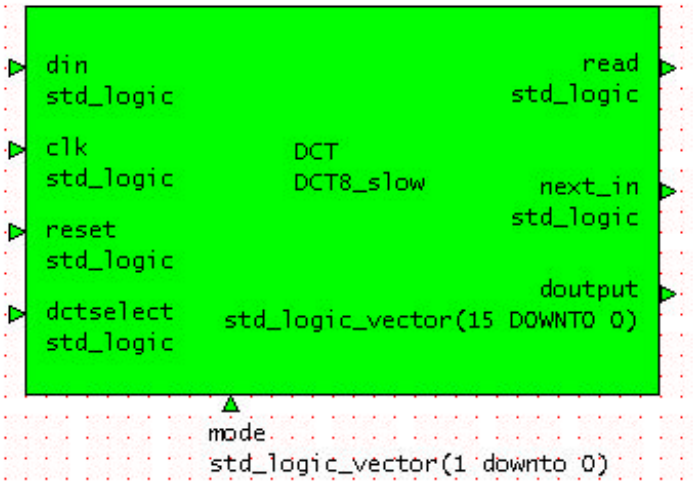
By Sherif T. EID

Abstract

The project describes the design and implementation of a 1-Dimensional eight word DCT/IDCT (*Discrete Cosine Transform / Inverse Discrete Cosine Transform*) processor that can be used in most of video/audio compression CODECs, such as JPEG. The core was designed taking into consideration maximum area optimization. It can process audio frames and JPEG still images (compression and decompression), where high speed is an important issue. It can accept inputs with different resolutions, according to the mode selected the input word width can range from 8 to 12 bits. It consumes 768 clock cycles to process eight 12-bit words, and 512 cycles to process eight 8-bit words. The design was made using Mentor Graphics Tools for design entry and implementation, it was implemented using the AMS 0.8 micron CYB technology, simulated after extracting parasitics (Back Annotation). I'm currently working on implementing my design on Altera (FLEX10K) and Xilinx (XC4000E and Virtex) FPGAs.

Free-DCT-L IP Core v1.0

The Free-DCT-L IP Core is an area-optimized core that performs Forward and Inverse 1D-eight Points Discrete Cosine Transforms. The core is designed to work in JPEG Still image and Audio compression CODECs where high -speed operation is not an issue; the top entity of the core is `dct8_slow`. The Free-DCT-L Core has the following pinout configuration.



din	Serial Data input
clk	Clock
reset	Synchronous Reset
dctselect	Selects between forward and inverse DCT
mode	Selects the resolution mode
read	Triggers to read the next output word
next_in	Triggers to input the next word
doutput	Data output port (parallel output)

Architecture

Calculation of the DCT coefficients can be made by the use of the following matrices

Matrix representation of the Forward Discrete Cosine Transform

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} = \begin{bmatrix} A & A & A & A & A & A & A & A \\ D & E & F & G & -G & -F & -E & -D \\ B & C & -C & -B & -B & -C & C & B \\ E & -G & -D & -F & F & D & G & -E \\ A & -A & -A & A & A & -A & -A & A \\ F & -D & G & E & -E & -G & D & -F \\ C & -B & B & -C & -C & B & -B & C \\ G & -F & E & -D & D & -E & F & -G \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

$$X = M \cdot x$$

Matrix representation of the Inverse Discrete Cosine Transform

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} A & D & B & E & A & F & C & G \\ A & E & C & -G & -A & -D & -B & -F \\ A & F & -C & -D & -A & -G & B & E \\ A & G & -B & -F & A & E & -C & -D \\ A & -G & -B & F & A & -E & -C & D \\ A & -F & -C & D & -A & G & B & -E \\ A & -E & C & G & -A & D & -B & F \\ A & -D & B & -E & A & -F & C & -G \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix}$$

$$x = M^T \cdot X$$

Where

A = Cos (?/4) B = Cos (?/8) C = Sin (?/8) D = Cos (?/16)
E = Cos (3?/16) F = Sin (3?/16) G = Sin (?/16)

X Forward DCT coefficients matrix

x Input values matrix

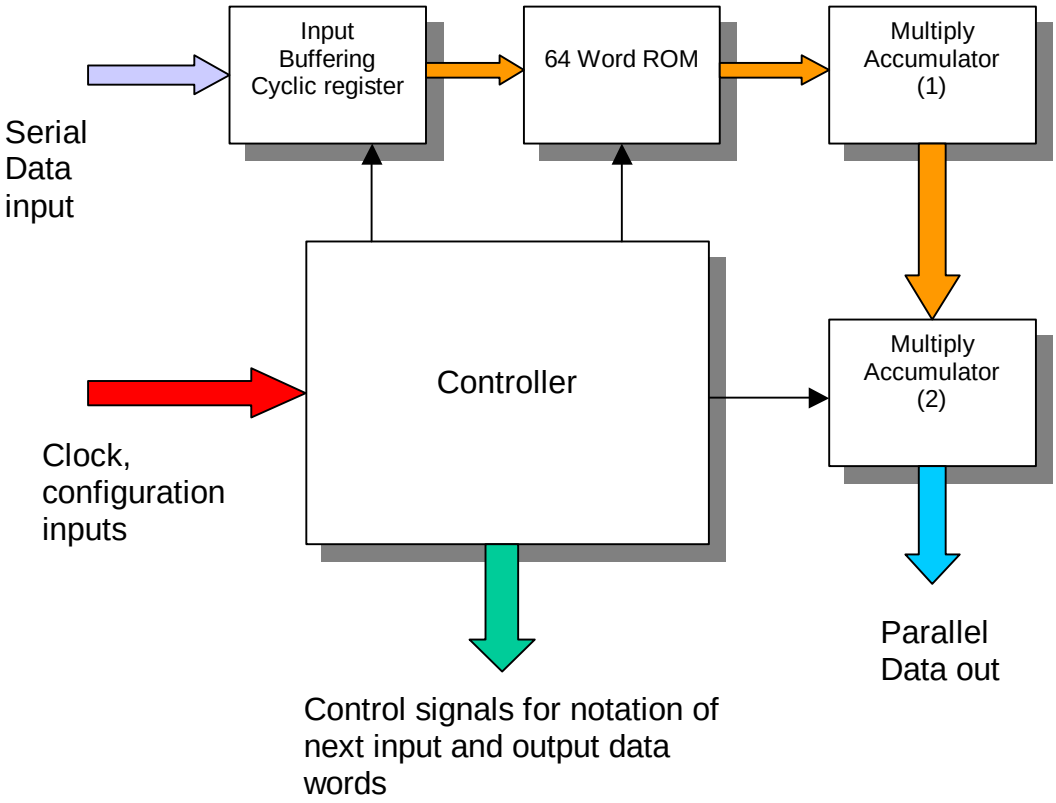
The values A, B, C, D, E, F, G of the matrix M were extracted from the general formula that describes the 1-D DCT

$$G(y) = \sum_{v=0}^{7} F(v)\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

Where

G(y) is the calculated DCT coefficient
F(v) is the input data value

Detailed explanation of the Core architecture will be available in upcoming documents.



Block Diagram of the Free-DCT-L core

How to use the Core

I) Operating modes

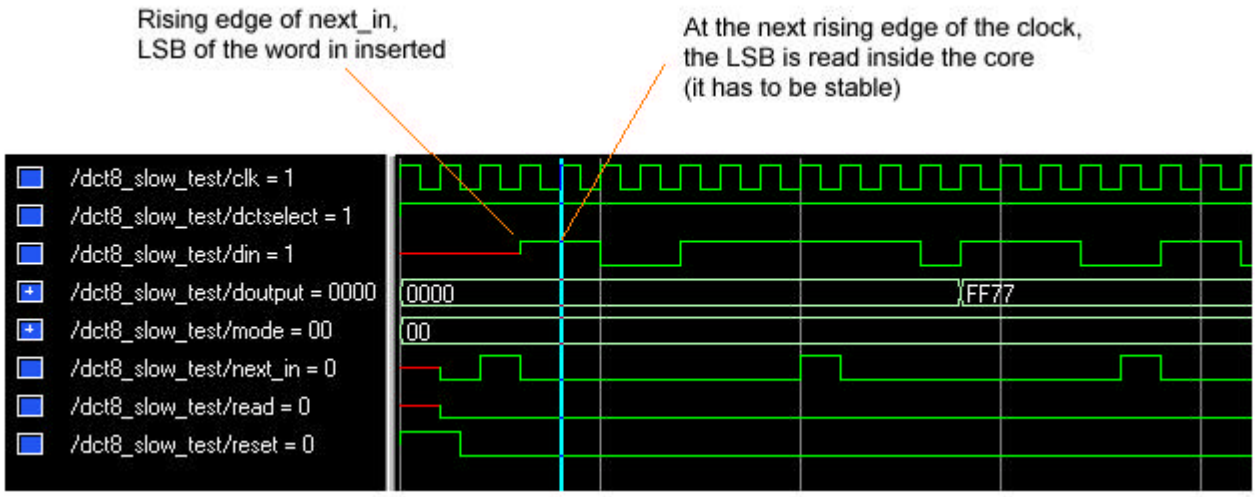
Configuration inputs *(**dctselect** and **mode(1 downto 0))*** control the modes of operation according to the following table

<table>
<tr><td>dctselect</td><td colspan="2">When logic ‘1’ the core performs Forward DCT, when logic ‘0’ the core perfo rms Inverse DCT</td></tr>
<tr><td rowspan="5">Mode(1 downto 0)</td><td colspan="2">This input controls the resolution of the input words to process according to the following values</td></tr>
<tr><td>Logic “00”</td><td>8 bits</td></tr>
<tr><td>Logic “01”</td><td>9 bits</td></tr>
<tr><td>Logic “10”</td><td>10 bits</td></tr>
<tr><td>Logic “11”</td><td>12 bits</td></tr>
</table>

II) Timing

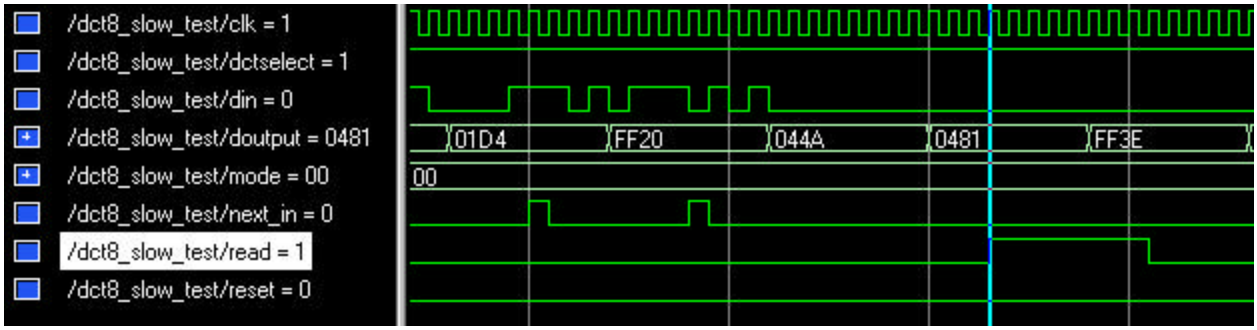
a) Inputting data

The core has a single serial data input, it accepts a **two's complement** representation of the input words of the chosen bit -width (according to ***mode(1 downto 0)***). It outputs a signal **next_in**, after one clock cycle of the rising edge of this signal the **LEAST SIGNIFICANT BIT** of the next input word is inserted taking into consideration that it will be evaluated in the core on the next rising edge of the clock. After that, the data input rate continues to be one bit per clock cycle until reaching the most significant bit of the word.



b) Reading output

The **read** signal can trigger an external register to connected to the **doutput** port to latch the value of the calculated DCT of IDCT word. The output is a 16 -bit word in two's complement representation.



When calculating 8-points, the core consumes 64*(chosen resolution) clock cycles to evaluate the 8 DCT or IDCT coefficients, the two extremes are 512 cycles (8 -bit) and 768 cycles (12-bit).

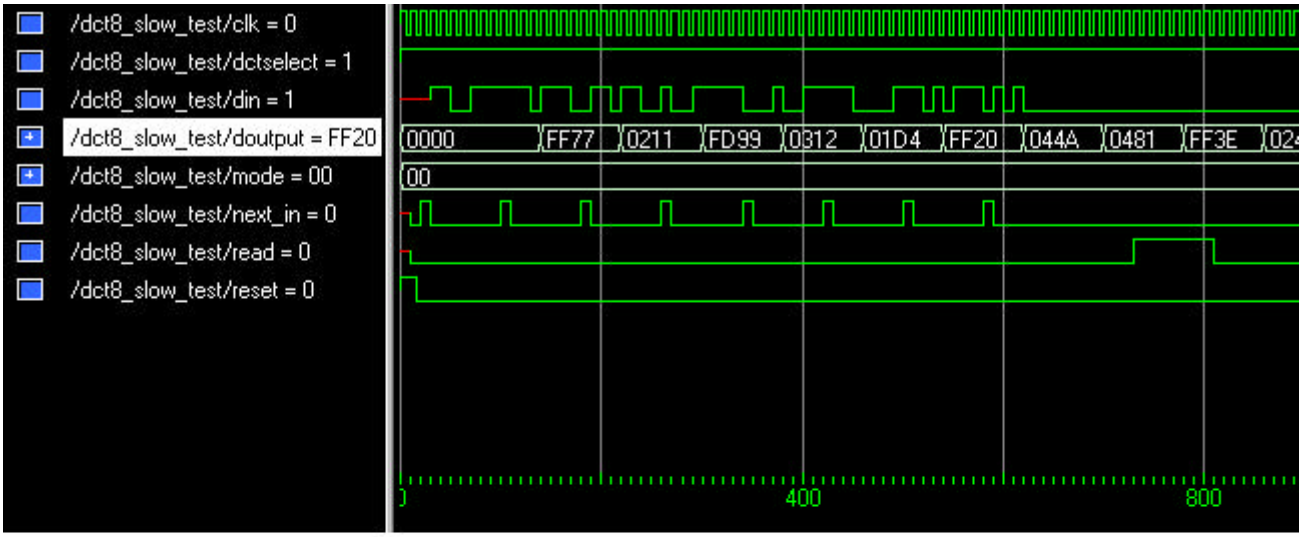
***Note**: To guarantee proper operation the **reset** port must be force to **'1'** for a complete clock cycle at least (The core resets when **reset** is **'1'** on the next clock rising edge).*

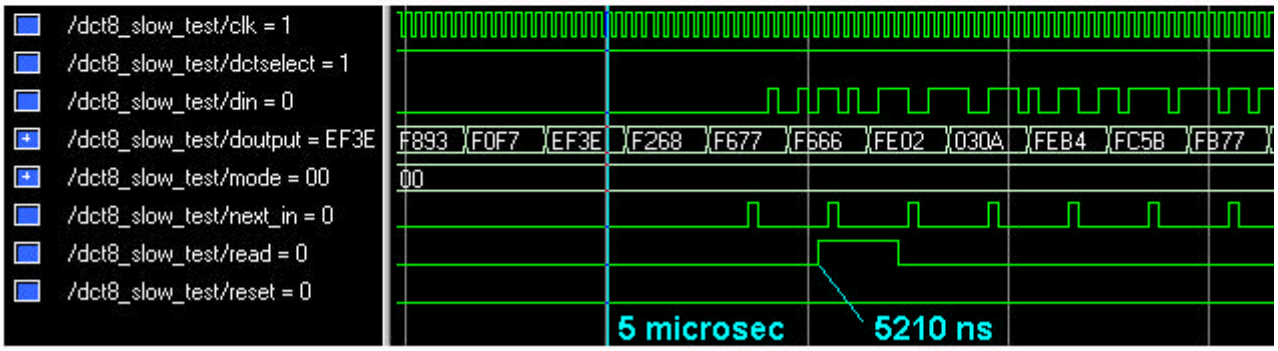
Considerations

?? The 8 words are entered serially during **K** cycles (where **K** = 8**resolution*), the first **next_in** rising edge happens after resetting by one cycle, it rises 8 times during the first **K** cycles. Then it takes the value '0' until a few cycles (*depends on resolution)* before the 8th rising edge of **read**.

*While **next_in** = '0' after the **8th word**, the **din** values are ignored*

Using mode "00" and a 100MHz clock the following results were obtained





Simulation results using a 100MHz Clock

?? *Useful output bits are **doutput(14 downto 0)**, **doutput(15)** can be ignored.*

Design Implementation

This core has been made using **Mentor Graphics** Tools for design entry and implementation. Mentor Graphics Packaged Power was used for VHDL entry, behavioral Simulation and Synthesis (*the new version of Packaged Power has been renamed to* ***FPGA Advantage***). Mentor Graphics IC Station was used for layout generation, and QuickSim was used for timing simulation. The file **chipdct-l.gif** contains a snapshot of the layout.

The design was synthesized and implemented using AMS 0.8 ？CYB technology (AMS HIT Kit 2.40), after extracting parasitics and back-annotaion, timing simulation showed that the implemented chip can operate properly on clock frequencies up to 50 MHz. The area of the chip is about 3.8 mm^2 (The pad **doutput(16)** could be eliminated resulting in less area consumption)

I am currently working on implementing my design using Altera (FLEX10K) and Xilinx (XC4000E) FPGAs, the core uses about 3000 gates.

Please feel free to send me any queries or suggestions at sherif_taher@ieee.org

Sherif Taher Eid