dctQidct

8x8 - Discrete Cosine Transform
8x8 - Inverse Discrete Cosine Transform
MPEG4 Quantization (H263 –method)
MPEG4 Inverse Quantization (H263 –method)

IP-Block Documentation
Version 1.1

Tampere University of Technology
Institute of Digital and Computer Systems
Timo Alho (timo.a.alho@tut.fi)
Version history:
1.0 Initial version
1.1 Quantizer parameter is no longer passed through DCT.
1 General description

This IP-block performs 8x8 - Discrete Cosine Transform, MPEG4 Quantization, MPEG4 Inverse Quantization and 8x8 - Inverse Discrete Cosine Transform. Used quantization and inverse quantization methods are explained in detail in appendix A. Block is implemented by using Mentor Graphics HDL Designer 2003.1 and VHDL language. System is optimized for FPGA usage (mainly for Altera’s Stratix –series) and it frequently makes use of embedded memory cells. If you want to use this IP block in your own design check appendix B to see what source files are needed.

Input and output data is in serial order i.e. only one data element (i.e. pixel) is sent/received per clock cycle. Data elements can be sent in row-wise or column-wise order as depicted in Figure 2. Reconstructed (IDCT) output data is in same order as input data, but quantized output data is in transposed order (e.g. if input data is column-wise, quantized output will be row-wise and vice-versa)

1.1 System performance

If system performance is not limited by lack of data, system can calculate DCT or IDCT in around 260 clock cycles when 16-bit internal data width is used. DCT and IDCT are processed in parallel, so when IDCT is being calculated for current 8x8-block, DCT can be calculated for next 8x8-block. System level timing diagram is shown in Figure 3.
Figure 3 - System level timing diagram
2 Interface description

2.1 Interface signals

Figure 4 shows the signals that can be found in the system interface. Data widths for the signals are defined in the VHDL-package files DCT_pkg.vhd, IDCT_pkg.vhd and Quantizer_pkg.vhd. Default values for signals are following:

CONSTANT DCT_inputw_co : integer := 9;  --DCT input width
CONSTANT IDCT_resultw_co : integer := 9;  --IDCT output width
CONSTANT QUANT_resultw_co : integer := 8;  --width of quantized values

clk
clk : IN std_logic;
Clock signal.

rst_n
rst_n : IN std_logic;
Asynchronous active low reset.

data_dct_in
data_dct_in : IN std_logic_vector(DCT_inputw_co-1 DOWNTO 0);
Input bus for 8x8 block data values. Input data values are signed integer values in 2’s complement form. Data must be arranged in serial order i.e. one element (pixel) of 8x8 block is sent at a time.

wr_dct_in
wr_dct_in : IN std_logic;
Write enable signal for input. When \textit{wr\_dct\_in} is high simultaneously with valid data on bus \textit{data\_dct\_in}, new data is sent into \textit{dct\_Qidct}.

**dct\_ready4column\_out**
\begin{verbatim}
dct_ready4column_out : OUT std_logic;
\end{verbatim}
When signal \textit{dct\_ready4column\_out} is high, system is ready for receiving one row/column i.e. 8 data values.

**QP\_in**
\begin{verbatim}
QP\_in : IN std_logic_vector (4 DWNTO 0);
\end{verbatim}
Input bus for quantizer parameter. Detailed information on how to control quantizer can be found on chapter 4.

**intra\_in**
\begin{verbatim}
intra\_in : IN std_logic;
\end{verbatim}
\textit{Intra\_in} should be high when 8x8 block is to be quantized as intra-block. Detailed information on how to control quantizer can be found on chapter 4.

**chroma\_in**
\begin{verbatim}
chroma\_in : IN std_logic;
\end{verbatim}
\textit{Chroma\_in} should be high when 8x8 block is to be quantized as chrominance-block. Detailed information on how to control quantizer can be found on chapter 4.

**loadQP\_in**
\begin{verbatim}
loadQP\_in : IN std_logic;
\end{verbatim}
Control signal for quantizer. When \textit{loadQP\_in} is high, signals \textit{QP\_in}, \textit{intra\_in} and \textit{chroma\_in} are loaded into quantizer. Detailed information on how to control quantizer can be found on chapter 4.

**data\_quant\_out**
\begin{verbatim}
data_quant_out : OUT std_logic_vector (QUANT\_resultw\_co-1 DWNTO 0);
\end{verbatim}
\textit{Data\_quant\_out} is output bus for quantized data values. Quantized output values are signed integers in 2's complement form. Data values are in transposed order compared to input values. Data is arranged in serial order i.e. one element of 8x8 -block is received at a time.

**wr\_quant\_out**
\begin{verbatim}
wr_quant_out : OUT std_logic
\end{verbatim}
Write enable signal for quantizer output. When \textit{wr\_quant\_out} is high, there is valid data on bus \textit{data\_quant\_out}.

**quant\_ready4column\_in**
\begin{verbatim}
quant_ready4column_in : IN std_logic;
\end{verbatim}
Receiving device should pull signal \textit{quant\_ready4column\_in} high, when receiving device is capable of receiving one quantized column/row. NOTE: Once \textit{quant\_ready4column\_in} is pulled high, it must not be pulled down until first data-word of column/row has been received.
**data_idct_out**

`data_idct_out : OUT std_logic_vector (IDCT_resultw_co-1 DOWNTO 0);`

*Data_idct_out* is output bus for IDCT data values. IDCT output values are signed integers in 2’s complement form. Values are in same order that input data values. Data is arranged in serial order i.e. one element of 8x8 -block is received at a time.

**wr_idct_out**

`wr_idct_out : OUT std_logic;`

Write enable signal for IDCT output. When `wr_idct_out` is high, there is valid data on bus `data_idct_out`.

**idct_ready4column_in**

`idct_ready4column_in : IN std_logic;`

Receiving device should pull signal `idct_ready4column_in` high, when receiving device is capable of receiving one IDCT column/row. NOTE: Once `idct_ready4column_in` is pulled high, it must not be pulled down until first data-word of column/row has been received.
3 Block usage

3.1 Data input

Data can be fed into system in row-wise or column-wise order. Data is sent one element (i.e. pixel) at a time arranged as blocks of one row/column.

The data should be fed into system in following manner:

1. Wait until system is ready to receive new column/row i.e. signal `dct_ready4column_out` is high.
2. Put valid data-word into bus `data_dct_in`, and simultaneously pull `wr_dct_in` high.
3. Repeat step 2 eight times. Data can be sent in consecutive clock cycles.
4. Go to step 1.
3.2 Data output
Data is transmitted one element at a time arranged as blocks of one row/column. Quantized data values are in transposed order compared to input data order (e.g. if input data is column-wise, quantized data will be row-wise). IDCT output data values are in same order as input data.

![Timing diagram for output data](image)

The quantized output data is received from the system in following manner:
1. When next block is ready to receive entire column/row, signal `quant_ready4column_in` should be pulled high.
2. Read valid data from bus `data_quant_out` when `wr_quant_out` is high.
3. Repeat step 2 eight times.
4. Go to step 1.

IDCT output data is received exactly in the same way as quantized data.

3.3 Typical application

To achieve full performance, user should ensure that data is read and written with required rate. Figure 7 presents one way to implement this, if receiving blocks cannot guarantee that they will be able to receive data at any instant.
4 Quantizer control

Quantizing and inverse quantizing is controlled with 4 input signals; \texttt{QP\_in}, \texttt{intra\_in}, \texttt{chroma\_in} and \texttt{loadQP\_in}. \texttt{DctQidct} is pipelined system, so in order to apply quantizer parameter to correct 8x8-block one must pay attention when these signals are controlled.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{QP_in}</td>
<td>Quantizer parameter. UNSIGNED number in range 1 to 31.</td>
</tr>
<tr>
<td>\texttt{intra_in}</td>
<td>If ‘1’ block is quantized as intra-block, otherwise as inter-block</td>
</tr>
<tr>
<td>\texttt{chroma_in}</td>
<td>If ‘1’ block is quantized as chrominance-block, otherwise as luminance-block.</td>
</tr>
<tr>
<td>\texttt{loadQP_in}</td>
<td>Load signal. Loads signals mentioned above into quantizer.</td>
</tr>
</tbody>
</table>

Table 1 - Quantizer control signals

To control quantizer put control data to inputs \texttt{QP\_in}, \texttt{intra\_in} and \texttt{chroma\_in} (see Table 1 for details). After this, data must be written into quantizer by pulling signal \texttt{loadQP\_in} up for at least one clock cycle. Note that \textbf{Quantizer parameter changes take effect only on beginning of 8x8 blocks}. Figure 8 depicts when quantizer parameters can be loaded into \texttt{dctQidct}.

![Figure 8 - Quantizer control timing](image)

To control quantizer put control data to inputs \texttt{QP\_in}, \texttt{intra\_in} and \texttt{chroma\_in} (see Table 1 for details). After this, data must be written into quantizer by pulling signal \texttt{loadQP\_in} up for at least one clock cycle. Note that \textbf{Quantizer parameter changes take effect only on beginning of 8x8 blocks}. Figure 8 depicts when quantizer parameters can be loaded into \texttt{dctQidct}.
Appendix A – Quantization and inverse quantization method

NOTE: This quantization method is the same as described in *MPEG-4 Video Verification Model version 18.0*

Quantization

For INTRA: \( \text{LEVEL} = \lfloor\text{COF} \rfloor / (2 \times QP) \)

For INTER: \( \text{LEVEL} = (\lfloor\text{COF} \rfloor - QP/2) / (2 \times QP) \)

Clipping to [-127:127] is performed for all coefficients except intra DC.

**Intra DC – coefficient**

<table>
<thead>
<tr>
<th>Component/Type</th>
<th>dc_scaler for Quantizer (Qp) range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 through 4</td>
</tr>
<tr>
<td>Luminance: Type1</td>
<td>8</td>
</tr>
<tr>
<td>Chrominance: Type2</td>
<td>8</td>
</tr>
</tbody>
</table>

The forward quantization is performed as follows.

\( \text{level} = \text{dc\_coef} / \text{dc\_scaler} \)

The reconstructed DC values are computed as follows.

\( \text{dc\_rec} = \text{dc\_scaler} \times \text{level} \)

**Inverse quantization**

\[\text{COF'} = 0, \quad \text{if LEVEL} = 0\]
\[\text{COF'} = 2 \times QP \times \text{LEVEL} + QP, \quad \text{if LEVEL} \neq 0, \text{QP is odd}\]
\[\text{COF'} = 2 \times QP \times \text{LEVEL} + QP - 1, \quad \text{if LEVEL} \neq 0, \text{QP is even}\]

The sign of COF is then added to obtain COF': \( \text{COF'} = \text{Sign(COF)} \times |\text{COF'}| \)

Clipping to [-2048:2047] is performed before IDCT.

The DC coefficient of an INTRA block is inverse quantized as described above.
Appendix B – Required files

If you want to use this IP block in your own design copy following files to your design directory. Files are here in hierarchical order (top level entity is last).

vhdl/common_da/Column_to_elements.vhd
vhdl/common_da/DPRAM.vhd
vhdl/common_da/Elements_to_column.vhd
vhdl/common_da/FlipFlop.vhd
vhdl/common_da/Mux2to1.vhd
vhdl/common_da/Parallel2Serial.vhd
vhdl/common_da/Serial_adder.vhd
vhdl/common_da/Serial_multiplier.vhd
vhdl/common_da/Serial_multiplier4idct.vhd
vhdl/common_da/Serial_subtractor.vhd

vhdl/dct/DCT_pkg.vhd
vhdl/dct/Rom_dct_sub.vhd
vhdl/dct/Rom_dct_sum.vhd
vhdl/dct/DCT1D_DA.vhd
vhdl/dct/DCT_control.vhd
vhdl/dct/DCT_core.vhd

vhdl/idct/IDCT_pkg.vhd
vhdl/idct/IDCT_post_sum.vhd
vhdl/idct/Rom_idct_even.vhd
vhdl/idct/Rom_idct_odd.vhd
vhdl/idct/IDCT1D_DA.vhd
vhdl/idct/IDCT_control.vhd
vhdl/idct/IDCT_core.vhd

vhdl/quantizer/Quantizer_pkg.vhd
vhdl/quantizer/IQuant.vhd

vhdl/dctQidct/IDCT_fifo.vhd
vhdl/dctQidct/dctQidct_core.vhd