A Universal Approach to Developing Fast Algorithm for Simplified Order-16 ICT

Jie Dong, King Ngi Ngan, Chi Keung Fong and Wai Kuen Cham

Department of Electronic Engineering
The Chinese University of Hong Kong

ISCAS2007, May 27-30, New Orleans, USA
Outline

- Introduction
- Simplified order-16 ICT
- Proposed approach
- Complexity analysis
- Conclusion and future work
Introduction

- **Integer Cosine Transform (ICT)**
  - **Pro:** Integer arithmetic implementation
    - Avoid mismatch between encoder and decoder
    - Good energy compaction capability if well-designed
    - Fast algorithms can be developed in the similar way for DCT
  - **Con:** Orthogonality depends on the elements of transform matrix, when the ICT is larger than order-4.

- **ICT for video coding**
  - Order-4 and Order-8 ICTs in H.264
  - Order-8 ICT in Audio Video Standard (AVS)
  - Order-16 ICT: efficient tool especially for HD video coding

- **Simplified Order-16 ICT**
  - **Pro:** Simpler while preserving the advantages of order-16 ICT
  - **Con:** Cannot develop fast algorithm in the similar way for DCT/ICT
**Simplified Order-16 ICT**

**General transform matrix**

- **• Contain at most 15 different integers**
  - o o o o o o o o o | o o ...
  - a b c d e f g h -h -g ...
  - i j k l -l -k -j -i -i -j ...
  - e f g h -a -b -c -d d c ...
  - m n -n -m -m -n n m m n ...
  - c d -a -b -g -h e f -f -e ...
  - j -l -i -k k i l -j -j l ...
  - h g -f -e d c -b -a a b ...
  - o -o -o o o -o -o o o -o ...
  - g -h e f c -d -a b -b a ...
  - k -i l j -j -l i -k -k i ...
  - b -a -d c -f e h -g g -h ...
  - n -m m -n -n m -m n n -m ...
  - d -c b -a -h g -f e -e f ...
  - l -k j -i i -j k -l -l k ...
  - f -e h -g b -a d -c c -d ...

**ISCAS2007**
Flow Diagram

- Even part ($T_{8e}$)
  - $o$ $o$ $o$ $o$ $o$ $o$ $o$ $o$
  - $i$ $j$ $k$ $l$ $l$ $k$ $j$ $i$
  - $m$ $n$ $n$ $m$ $m$ $n$ $n$ $m$
  - $j$ $-l$ $i$ $-k$ $k$ $i$ $l$ $j$
  - $o$ $-o$ $o$ $o$ $o$ $o$ $o$ $o$
  - $k$ $-i$ $l$ $j$ $j$ $-l$ $i$ $-k$
  - $n$ $-m$ $m$ $n$ $n$ $m$ $m$ $n$
  - $l$ $-k$ $j$ $i$ $i$ $j$ $k$ $l$

- Odd part ($T_{8o}$)
  - $a$ $b$ $c$ $d$ $e$ $f$ $g$ $h$
  - $e$ $f$ $g$ $h$ $-a$ $-b$ $-c$ $-d$
  - $c$ $d$ $-a$ $-b$ $-g$ $-h$ $e$ $f$
  - $h$ $g$ $-f$ $-e$ $d$ $c$ $-b$ $-a$
  - $g$ $-h$ $-e$ $f$ $c$ $-d$ $-a$ $b$
  - $b$ $-a$ $-d$ $c$ $-f$ $e$ $h$ $-g$
  - $d$ $-c$ $b$ $-a$ $-h$ $g$ $-f$ $e$
  - $f$ $-e$ $h$ $-g$ $b$ $-a$ $d$ $-c$
Steps of the Proposed Approach

- Separate the 2-D transform into 2 1-D transforms
- Using 8 butterflies (left block) to exploit the symmetries w.r.t the dash line in the general transform matrix
- Fast algorithm for the even part (upper-right block) that is exactly the general transform matrix of an order-8 ICT.
  - Borrow order-8 ICTs and their fast algorithms, e.g., order-8 ICTs in H.264 or AVS
  - Otherwise, the fast algorithm will be developed in the same way for the odd part.
- Fast algorithm for the odd part (bottom-right block)
An Order-8 ICT and its Fast Algorithm

- Order-8 ICT in H.264 is borrowed as the even part (upper-right block) of the simplified order-16 ICT

<table>
<thead>
<tr>
<th>8 8 8 8 8 8 8 8</th>
<th>10 6 3 -6 -10 -12</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 4 -4 -8 -8 -4 4 8</td>
<td>10 -3 -12 -6 6 12 3 -10</td>
</tr>
<tr>
<td>8 -8 -8 8 8 -8 -8 8</td>
<td>6 -12 3 10 -10 -3 12 -6</td>
</tr>
<tr>
<td>6 -12 3 10 -10 -3 12 -6</td>
<td>4 -8 8 -4 -4 8 -8 4</td>
</tr>
<tr>
<td>3 -6 10 -12 12 -10 6 -3</td>
<td>3 -6 10 -12 12 -10 6 -3</td>
</tr>
</tbody>
</table>

Fast Algorithm for order-8 ICT

ISCAS2007
Fast Algorithm for the Odd Part

- $T_{8o}$: a type of dyadic transform having different structures with the odd part of DCT/ICT

- Decompose $T_{8o}$ to the multiplication of three 8x8 matrices instead of butterfly operations
  
  $T_{8o} = M_2 \times M_3 \times M_4$

  $M_2$, $M_3$, $M_4$: Stage 2~4 in the flow diagram

- Considerations for $M_2$, $M_3$, $M_4$
  - Contain integers only
  - Small magnitude to avoid multiplications
  - Be sparse
Fast Algorithm for the Odd Part (Cont.d)

- Constraints for $M_2$, $M_3$, $M_4$: each has the same properties of $T_{80}$
  - Orthogonality
  - Basis vectors has the same length (length of vector $a$: $axa^T$)

- Conditions (necessary, not sufficient) of existence under the constraints ($n_{80}$, $n_2$, $n_3$, and $n_4$ represent the length of each matrix)
  \[
  |\det(T_{80})| = |\det(M_2)| \times |\det(M_3)| \times |\det(M_4)| \tag{1}
  \]
  \[
  \Rightarrow n_{80}^4 = n_2^4 \times n_3^4 \times n_4^4 \tag{2}
  \]
  \[
  \Rightarrow n_{80} = n_2 \times n_3 \times n_4 \tag{3}
  \]

$n_{80}$ is the product of at least 3 prime numbers

- (3) means $n_2$, $n_3$, and $n_4$ are much smaller than $n_{80}$, which indicates the elements in the three matrices have very small magnitudes and may also contain many zeros.
Fast Algorithm for the Odd Part (Cont.d)

- Search $M_2$, $M_3$, $M_4$, start from $M_4$
  \[ T_{8o}x(M_4)^{-1} = M_2xM_3xM_4x(M_4)^{-1} \Rightarrow T_{8o}x(M_4)^T/n_4 = M_2xM_3 \tag{4} \]

  - Notice, in (4)
    - $T_{8o}x(M_4)^T/n_4$ contains only integers
    - $(M_4)^T$ can be regarded as a set of column vectors

- Search $M_4$
  - Establish a set of column vectors $\{b_i\}$, satisfying
    - Length of $b_i$ is $n_4$, i.e., $b_i^xb_i^T = n_4$
    - $T_{8o}x b_i / n_4$ contains only integers
    - Pick out eight orthogonal column vectors from $\{b_i\}$ to form $(M_4)^T$ and thus get $M_4$. 

- Search $M_3$ and $M_2$ (similar to searching $M_4$)
  \[ [T_{8o}x(M_4)^T/n_4]x(M_3)^{-1} = M_2xM_3x(M_3)^{-1} \Rightarrow [T_{8o}x(M_4)^T/n_4]x(M_3)^T/n_3 = M_2 \tag{5} \]
An example

- **Even part** (T_{8e})
  
  \[
  \begin{array}{cccccccc}
  8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\
  10 & 9 & 6 & 2 & -2 & -6 & -9 & -10 \\
  10 & 4 & -4 & -10 & -10 & -4 & 4 & 10 \\
  9 & -2 & -10 & -6 & 6 & 10 & 2 & -9 \\
  8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\
  6 & -10 & 2 & 9 & -9 & -2 & 10 & -6 \\
  4 & -10 & 10 & -4 & -4 & 10 & -10 & 4 \\
  2 & -6 & 9 & -10 & 10 & -9 & 6 & -2 \\
  \end{array}
  \]

- **Odd part** (T_{8o})
  
  \[
  \begin{array}{cccccccc}
  11 & 11 & 11 & 9 & 8 & 6 & 4 & 1 \\
  8 & 6 & 4 & 1 & -11 & -11 & -11 & -9 \\
  11 & 9 & -11 & -11 & -4 & -1 & 8 & 6 \\
  1 & 4 & -6 & -8 & 9 & 11 & -11 & -11 \\
  4 & -1 & -8 & 6 & 11 & -9 & -11 & 11 \\
  11 & -11 & -9 & 11 & -6 & 8 & 1 & -4 \\
  9 & -11 & 11 & -11 & -1 & 4 & -6 & 8 \\
  6 & -8 & 1 & -4 & 11 & -11 & 9 & -11 \\
  \end{array}
  \]

- \{a, b, c, d, ..., n, o\} =
  
  \{11, 11, 11, 9, 8, 6, 4, 1, 10, 9, 6, 2, 10, 4, 8\}

- **Even part** (T_{8e})

  order-8 ICT in AVS

- **Odd part** (T_{8o})

  \[|T_{8o}| = 9.9049 \times 10^{10}\]

  \[n_{8o} = 561 = 3 \times 11 \times 17\]

  \[n_2 = 17, \ n_3 = 3, \ n_4 = 11\]

  Search \ M_2, \ M_3, \ M_4 \ using \ the \ method \ in \ the \ previous \ page
An Example (Cont.d)

\[ T_{80} = M_2 \times M_3 \times M_4 = \]

\[
\begin{bmatrix}
-2 & 0 & 1 & -1 & -1 & 3 & -1 & 0 \\
3 & -1 & 1 & 1 & 0 & 2 & 0 & 1 \\
-1 & -3 & 1 & 0 & 1 & 0 & 2 & -1 \\
0 & 1 & 0 & 1 & 3 & 1 & -1 & -2 \\
1 & -1 & -3 & -2 & 0 & 1 & 0 & -1 \\
1 & 1 & 1 & 0 & -2 & 0 & 1 & -3 \\
0 & -2 & 0 & 1 & -1 & -1 & -3 & -1 \\
-1 & 0 & -2 & 3 & -1 & 1 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & -1 & -1 & -2 & 0 \\
-1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \\
-1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
0 & -2 & 0 & -1 & 0 & 2 & -1 & 1 \\
0 & -1 & 0 & 2 & 1 & -1 & 0 & 2 \\
0 & -2 & 1 & 1 & 0 & 0 & 1 & -2 \\
-2 & 0 & -1 & 0 & 2 & 0 & -1 & -1 \\
-1 & 0 & 2 & 0 & -1 & -1 & -2 & 0 \\
-2 & 0 & 1 & -1 & 0 & 0 & 2 & 1 \\
-1 & 1 & 0 & 2 & -1 & 2 & 0 & 0 \\
-1 & -1 & -2 & 0 & -2 & -1 & 0 & 0 \\
\end{bmatrix}

ISCAS2007
12
An Example (Cont.d)
## Complexity Analysis

### Operation comparison with matrix multiplication

<table>
<thead>
<tr>
<th>Operation</th>
<th>Fast simplified ICT</th>
<th>Matrix multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>150</td>
<td>240</td>
</tr>
<tr>
<td>Multiplication</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>Shifting</td>
<td>32</td>
<td>0</td>
</tr>
</tbody>
</table>

### Execution time comparison with matrix multiplication

Transform 10,000 data blocks using 3.2GHz CPU and the data in the blocks are uniformly distributed in [-256,255]

<table>
<thead>
<tr>
<th></th>
<th>Fast algorithm</th>
<th>Matrix multiplication</th>
<th>Saving time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCT</td>
<td>0.035 s</td>
<td>0.550 s</td>
<td>93.6%</td>
</tr>
<tr>
<td>Simplified ICT</td>
<td>0.025 s</td>
<td>0.293 s</td>
<td>91.4%</td>
</tr>
<tr>
<td>$T_{80}$</td>
<td>0.004 s</td>
<td>0.036 s</td>
<td>88.9%</td>
</tr>
</tbody>
</table>
Conclusion and Future Work

- In this paper, a universal approach to developing fast algorithms for simplified order-16 ICT is proposed.
  - A general transform matrix for simplified order-16 ICT
  - Decomposed matrix multiplication to addition and shifting operations by a universal method
  - Save 90% of the computational time compared with matrix multiplication

- Future work
  - When decomposing the odd part
    - Relax the constraints of each $M_i$ matrices and explore whether number of operations can be reduced
    - Instead of exhaustive search, use new algorithm to search for a set of orthogonal vectors among a pool of vectors
Thank you!

Q&A