Abstract—Within the encoding process of H.264 videos, the function of transform and quantization is to transform and quantize the residual macroblocks resulted from motion prediction for transmission. This report introduces an improved algorithm and presents several more efficient design based on the new algorithm and corresponding architecture transform.

Keywords—H.264, encoding, integer transform, quantization, time-multiplexing, folding

I. INTRODUCTION

An H.264 encoder mainly consists of the motion prediction block, the transform and quantization block, and entropy encoder [1] (shown in Figure.1-1). Among these building blocks, transform and quantization requires huge amount of computations to transform and quantize the residual macroblocks after motion prediction for transmitting.

Figure.1-1. Typical structure of H.264 encoder

For the baseline profile of H.264, each 16 × 16 macroblock needs 16 samples of 4 × 4 “luma” blocks and 8 samples of 4 × 4 “chroma” blocks. These samples need to perform a 4 × 4 forward transform. Additionally, a sample of 4 × 4 “luma” DC coefficients and a sample of 2 × 2 “chroma” DC coefficients require Hadamard transform and a 2 × 2 forward transform, respectively. Observing the three kinds of transform possibly required, without loss of generality, we choose to implement the 4 × 4 forward transform for “luma” and “chroma” blocks.

The conclusions and design rules will apply directly to the rest two transforms because only a few coefficients are modified.

The throughput requirement for transform and quantization in 1080p and Digital Cinema format is shown in Form 1-1. Our goal is to explore efficient designs under these specifications.

Table 1-1. The throughput requirement for 1080p and digital cinema

<table>
<thead>
<tr>
<th>Video Format</th>
<th>1080p</th>
<th>Digital Cinema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1920 × 1080 @60Hz</td>
<td>4096 × 2048 @30Hz</td>
</tr>
<tr>
<td>Throughput</td>
<td>188 Mpixel/s</td>
<td>377 Mpixel/s</td>
</tr>
</tbody>
</table>

Different from previous standards such as H.263, H.264 introduces integer approximation [7] of transform and quantization. This leads to great reduction of computation complexity.

In Section II we introduce the basic integer transform algorithm and reference design for transform and quantization. In Section III, an improved algorithm and the corresponding design using time-multiplexing technique are presented. In Section IV, a folding architecture is implemented for better area efficiency. In Section V, the architectural techniques used in this report are summarized to compare the resulted speed, power and area.

II. BASIC ALGORITHM & REFERENCE DESIGN

The transform process in H.264 is realized by matrix arithmetic with 4 × 4 transform coefficients [1][2]. The forward transform for each 4 × 4 macro-block is shown in Eq. (2-1), after factorizing the original DCT transform so that only additions and shifting operations are needed. Here the coefficients in $E_i$ is used for scaling the result of matrix multiplication to ensure the orthogonal DCT characteristics.

$$Z = C_iX C_i^T \otimes E_i,$$  (2-1)
where \( C_t = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \), \( E_t = \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix} \),

and \( a = \frac{1}{2}, \ b = \frac{2}{\sqrt{5}}, \ d = \frac{1}{2}. \)

The reference design architecture is directly derived from Eq. (2-1). Despite scaling coefficients, the basic algorithm is carried out through two stages:

\[ M = C_t X, \]
\[ Y = M C_t^T. \]

More specifically, assuming the element in \( X \) is denoted as \( X_{ij} \), then

\[ M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix}, \ u = 0, 1, 2, 3. \]

Because of the symmetry of the above partial production, it can thus be realized by 4-input butterfly diagram, as shown in Fig. 2-1.

![Figure 2-1. Basic Algorithm: 4-input, 4-output Butterfly](image)

Similarly, the second stage of forward transform is expressed as following,

\[ Y = \begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & 2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 2M_{00} + X_{0u} + X_{2u} + X_{3u} \\ 2M_{10} + X_{0u} - X_{2u} - 2X_{3u} \\ X_{0u} - X_{1u} - X_{2u} - 2X_{3u} \\ X_{0u} - 2X_{1u} + 2X_{2u} - X_{3u} \end{bmatrix}, \ u = 0, 1, 2, 3. \]

The output of forward transform is processed by a forward quantizer. The forward quantizer absorbs the scaling coefficients in \( E_t \) and it is based on Eq. (2-2)

\[ Z_{ij} = round(Y_{ij} \cdot PF / Q_{step}) \quad (2-2) \]

where \( PF \) is a post-scaling factor. Equivalently, we apply a multiplication factor (MF)

\[ Z_{ij} = round(Y_{ij} \cdot M/F / 2^{qbits}) \]

where \( MF/2^{qbits} = PF/Q_{step} \) and \( qbits = 15 + \lfloor QP/6 \rfloor \) to avoid division operation. Therefore,

\[ |Z_{ij}| = \lfloor |Y_{ij}| \cdot MF \big| \gg qbits \]
\[ \text{sign}(Z_{ij}) = \text{sign}(Y_{ij}) \]

In this report, we choose the quantization parameter \( QP = 4, Q_{step} = 1 \), assuming that the channel can support a quite high transmission rate. In this case, the values of MF are listed in the following Tab. 2-1.

<table>
<thead>
<tr>
<th>Positions</th>
<th>(0,0)(2,0)</th>
<th>(2,2)(0,2)</th>
<th>(1,1)(3,3)</th>
<th>Other positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MF</td>
<td>8192</td>
<td>3355</td>
<td>5243</td>
<td></td>
</tr>
</tbody>
</table>

The construction of the reference design applying the above basic algorithm is straightforward, as shown in TransformQuantization_ref.md. The input data wordlength is 9-bit. The wordlength of transform output and quantization output are both 15-bit (no fraction).

Based on the throughput requirement, since the reference design can process 16 pixels in each cycle, the operation clock frequency is 25MHz. This reference design consumes area of 40,157um². Additionally, we find that the highest clock rate for this design can reach 500MHz, which means that our technology is much faster than the specification. As a result, we can use the lowest \( V_{DD} \) supported by the technology for lower power and the main task for the following architecture transforms is to reduce area consumption.

The dynamic power can be estimated with the following equation.

\[ P_d = fCV_{DD}^2 \]

Where \( f \) is the operation frequency and \( C \) is proportional to the area of the design. (The leakage power is also proportional to the area.) Here \( V_{DD} \) is chose to be the lowest possible value and it is same for different implementations. Therefore, the power consumption of the following designs is only related to the operation frequency and area.

III. IMPROVED ALGORITHM & INTERLEAVING STRUCTURE

A. Time-Multiplexing Butterfly Transform Architecture

The symmetric characteristic in the basic 4-input butterfly can be further observed. For example in Fig. 2-1, intermediate data \( M_{00} \) and \( M_{10} \) is calculated in the first clock cycle, and further transmitted to the second stage butterfly operation. In the second clock cycle, \( M_{20} \) and \( M_{30} \) are computed by adding shifters and negaters. In [10]’s proposed directed 2-D transform
algorithm, this interleaving approach is formulated into the following expression,

\[
\begin{align*}
[M_{0u}] &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot [X_{0u} + X_{3u}], \\
[M_{2u}] &= \begin{bmatrix} 2 & 1 \\ 1 & -2 \end{bmatrix} \cdot [X_{0u} - X_{3u}],
\end{align*}
\] (3-2)

\[
\begin{align*}
[M_{3u}] &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot [X_{1u} + X_{2u}], \\
[M_{4u}] &= \begin{bmatrix} 2 & 1 \\ 1 & -2 \end{bmatrix} \cdot [X_{1u} - X_{2u}],
\end{align*}
\] (3-3)

\[u = 0, 1, 2, 3.\]

The interleaved algorithm is illustrated in Fig. 3-1. A control signal is applied to decide whether to implement Eq. (3-2), or Eq. (3-3). Theoretically compared to the reference design, the number of adders in original 4 × 4 block is reduced by half. The time-multiplexing approach increases the latency since every 4 intermediate outputs are transmitted within 2 clock cycles. Therefore, at the second stage, the 4 × 4 blocks needed to deal with intermediate data are also reduced by half.

The transform block is followed with 8 quantization blocks. Each block consists of one 2-1 MUX for MF, one 15-bit input general multiplier and one 15-bit shifter. (Details in TransformQuantization2.mdl).

B. Time-Multiplexing Implementation in Simulink and Synthesis

In Simulink implementation of time-multiplexing algorithm, the control logic is implemented simply by a 1-bit counter.

In Simulink model TransformQuantization2.mdl, the first-stage-multiplexing implementation is presented. This time-multiplexing architecture can process 8 pixels in each cycle so that the required clock rate is 50MHz. The synthesized area for this design is 30,912\(um^2\), 23% lower than the reference design. The dynamic power of is 54% more due to the doubled clock rate.

The same concept of time-multiplexing can also be applied to the second stage of the forward transform, as illustrated in TransformQuantization3.mdl. Since every four intermediate data should be hold for 2 clock cycle at the second stage, the sampling rate at the second stage should also be twice that of the first stage. The control logic is modified accordingly. The MUX select signal at the second stage is twice that of the first stage. The total area is reduced because only 4 multipliers are needed to perform quantization. This design has normalized dynamic power of 2.676 and the area consumption is 33% lower compared to the reference design.

IV. FOLDING ARCHITECTURE AND RESOURCE SHARING

A. Folding Architecture for Transform

By observing the implementations in Part III, we find that it is possible to merge the two stage of processing into one by exploiting the folding technique to further reduce the area. The basic idea is eliminating the second processing stage and feedback the 8 output data streams to 8 MUXs. The MUXs pass the feedback data or the external input data with control logic. The input wordlength of this single stage is increased to 12-bit to accommodate the feedback M data and the output wordlength is still 15-bit.

This folding architecture needs 4 clock cycles to process one 4 × 4 block. The input data has the frequency of 25MHz and the MUXs at the input sample the data at the rate of 100MHz. The clock diagram is shown in Fig. 4-1. In the first cycle, MUX control signal feedback = 0, computation mode signal mode = 0. The MUXs pass the 16 input data X_{ij} and the output data are M_{0u} and M_{2u} (u = 0, 1, 2, 3). In the second cycle, feedback = 1, mode = 0 , and M_{0u} and M_{2u} are feedback to the processing stage . X_{yu} and X_{yu} are obtained at the output. In the third cycle, feedback = 0, mode = 1, M_{1u} and M_{3u} are calculated from the same X_{ij} data as Cycle 1. Y_{1u} and Y_{3u} come to the output in Cycle 4, feedback = 1, mode = 1.

B. Quantization without Resource Sharing

The above folding architecture is followed by 8 quantization blocks. Each block consists of one 2-1 MUX for MF, one 15-bit input general multiplier and one 15-bit shifter. The output wordlength is 15-bit (no fraction). The Simulink model of this design is TransformQuantization4.mdl.

According to the synthesis result, this folding transform architecture without resource sharing for quantizer consumes area of 29,245\(um^2\). The normalized dynamic power is 2.912.
This design is not efficient compared to the two implementations in Part III. We need to find a better solution.

C. Quantization with Resource Sharing

With further examining the operation of the quantization stage in IV.B, we find that the multipliers and shifters only have 50% utilization, which means that nearly 50% area in the quantization stage is wasted. Fortunately, the pattern of output streams of the folding transform block provides the possibility for resource sharing. The quantization stage can only use 4 multipliers and shifters if a feedback loop is added at the transform output to replace the \( M \) value with the previous \( Y \) value. Each of the four blocks in the quantization stage receives two transform output data streams and a 2-1 MUX is used to select either one of them based on the control signal. The clock diagram with resource sharing is shown in Fig. 4-2. The Simulink model is `TransformQuantization4_2.mdl`.

![Figure 4-2. Clock Diagram of Resource Sharing](image)

In Cycle 1, \( M_{0u} \) and \( M_{2u} \) are calculated but not valid at the output. In Cycle 2, output loop feedback control signal (also the path select signal) \( f_{b_{out}} = 0 \). The following quantization stage picks the data from Path 1, 3, 5, and 7, and produce four output \( Z_{00}, Z_{02}, Z_{20}, Z_{22} \). In Cycle 3, \( f_{b_{out}} = 1 \) and the output value \( Y \) is preserved. The following quantization block picks the data from Path 2, 4, 6, and 8, and produce four output \( Z_{01}, Z_{03}, Z_{21}, Z_{23} \). In Cycle 4 and 5, the operation is similar as Cycle 2 and 3. Data \( Z_{10}, Z_{12}, Z_{30}, Z_{32} \) and \( Z_{11}, Z_{13}, Z_{31}, Z_{33} \) are obtained respectively.

The total area for transform and quantization with resource sharing is 25.975 \( \mu m^2 \), 11% less than the implementation without resource sharing, and 35% less than the reference design.

V. ARCHITECTURE TRANSFORM SUMMARY

In this report, a total of 5 implementations are presented. The operation frequency, power and area of each design is shown in Table 5-1.

<table>
<thead>
<tr>
<th>Ref</th>
<th>Clock (MHz)</th>
<th>Area (( \mu m^2 ))</th>
<th>Normalized Area / Leakage Power</th>
<th>Normalized Dynamic Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>25</td>
<td>40.157</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>30.912</td>
<td>0.770</td>
<td>1.540</td>
</tr>
<tr>
<td>4-1</td>
<td>100</td>
<td>26.878</td>
<td>0.669</td>
<td>2.676</td>
</tr>
<tr>
<td>4-2</td>
<td>100</td>
<td>29.245</td>
<td>0.728</td>
<td>2.912</td>
</tr>
</tbody>
</table>

With the above comparison, we can find that the designs 2 and 4-2 are more efficient in terms of power and area. Time-multiplexing can significantly reduce area consumption at the price of higher dynamic power. Techniques such as folding and resource sharing can be used for further area saving. However, the control logic overhead occupies some area and higher clock rate results in higher power consumption.

REFERENCES