HEVC Inverse Transform Architecture Utilizing Coefficient Sparsity

Maleen Abeydeera*, Ajith Pasqual†
Department of Electronic and Telecommunication Engineering,
University of Moratuwa, Sri Lanka
Email: *maleen89@gmail.com, †pasqual@ent.mrt.ac.lk

Abstract—The inverse transform function of the HEVC Decoder has grown greatly in complexity with the addition of larger transform sizes and recent works have focused on efficient architectures that can achieve the required throughput. In this work we make the observation that a majority of the coefficients in a typical transform operation is zero, and therefore has no impact on the final outcome. We propose an architecture that can efficiently operate on such sparse matrices and introduce a scheduling strategy which completes a 2D IDCT with bare minimum iterations, with the added advantage of being able to integrate seamlessly with the entropy decoder without a coefficient reordering buffer. Experiments show that although the performance of this approach is scalable with the bit rate, a 120 MHz operating frequency is sufficient to handle QHD @ 48 Mbps, which is less than one third of the frequency requirement of prior work.

I. INTRODUCTION

Many of the recent video compression standardization efforts have adopted the traditional pipeline which consists of prediction, transform and entropy coding. In this pipeline, the transform stage makes use of the fact that the difference between the actual pixel contents and the predicted contents generally show a predictable pattern. The transformation of these residual pixels to frequency domain results in non-significant high-frequency components. A subsequent quantization step reduces these to zero and the low frequency components are significantly reduced in magnitude, reducing the number of bits required.

The 50% compression efficiency that the latest HEVC (High Efficiency Video Coding) standard brings over the incumbent H.264 comes at the cost of greater hardware complexity. This added complexity spans all stages and the transform is no exception. The H.264 specifies 4x4 and 8x8 Discrete Cosine Transforms (DCT), where as the HEVC adds 16x16 and 32x32. While the larger block sizes significantly increases the memory requirement, the computational requirement is also amplified by the fact that the transform matrices in HEVC use 8 bit coefficients in comparison to 5 bits in H.264. HEVC also mandates the use of a Discrete Sine Transform (DST) for the luma component of 4x4 intra blocks.

During decoding, the transformed coefficients are converted back to the spatial domain via an inverse transform. According the HEVC specification [1], the 2-D IDCT (Inverse DCT) can be expressed as:

$$X = C^T Y C$$  \hspace{1cm} (1)

Where $Y$ is the transformed coefficient matrix, $X$, the residual pixel matrix and $C$, the IDCT matrix.

The 2-D IDCT can be decomposed into two 1-D IDCTs where a column transform is followed by a row transform. In prior work, a single IDCT instance that is time multiplexed between the two, is commonly employed for resource efficiency. The design of this IDCT instance, however has varied considerably between implementations.

The architectures by Park [2], Shen [3] and Tikekar[4] maintain a uniform throughput for all transform sizes, whereas Zhu [5] and Yao [6] trades a larger area for better throughput with a design that can perform a single 1-D IDCT in one cycle. The architecture of Chiang [7] can be reconfigured to use a large hardware block to perform multiple smaller IDCTs. Ziyou [8] observes that most of the prior works achieve greater than required throughputs at a large resource usage and proposes a middle ground.

In this paper, we make two key observations that have been rarely considered in the work so far.

Sparse Matrix: The purpose of a transform from the spatial domain to the frequency domain is to represent the pixel information in as few basis vectors as possible. Hence the transformed coefficients would mostly congregate at the upper left section of the matrix, with the others being zero. During decoding, these zero coefficients do not contribute to the final residual pixel values, and hence can be safely dropped.

In [9], a zero column skipping strategy is adopted to make use of some of this sparsity. However this is limited to the skipping of column transform for columns which consist of all zero coefficients. Our approach, however, is able to skip each and every zero coefficient (both in column and row transform), because we only operate on a single coefficient per cycle. This is in contrast to 4 coefficients per cycle as in [9] which leads it to not being able to take advantage of isolated zero coefficients unless they occur contiguously.

Sequential Entropy Decoding: The transformed coefficients have to undergo a process of entropy decoding prior to their value being available. During entropy encoding, the 2D transformed coefficients are ordered, binarized and each bin is sent through an arithmetic encoder. The arithmetic decoding is highly sequential and cannot be parallelized across coefficients [10]. Therefore the coefficients are available to the inverse transform module sequentially, and in an order that is highly unpredictable.

In all previous work, the column transform stage consumes
coefficients column wise. But this is in direct contrast to the order that coefficients come out of the entropy decoder, which for the most case is diagonal starting from the bottom right corner (horizontal and vertical scans are used in specialized scenarios. However in general a decoder has to support all three - see Fig. 1). Apart from requiring an intermediate 32x32 16-bit buffer (a double buffer if both modules are to be active simultaneously), this also increases the processing latency.

In this work, we present an inverse transform architecture that is compatible with sequential and out of order coefficient input, which is detailed in Section II. In section III, a scheduling algorithm is introduced, that feeds the architecture with only non-zero coefficients, with the expectation of completing a 2D IDCT/IDST with the bare minimum iterations. Section IV present the synthesis and simulation results which show that required frequency is scalable with the bit rate but still very low in comparison with prior work.

II. HARDWARE ARCHITECTURE

![Diagram of inverse transform architecture](image)

Figure 1: The coefficient scan orders in HEVC. Conventional inverse transform architectures require a vertical input

The architecture in Figure 1 can be time multiplexed to perform both row and column transforms. Output coefficients from the entropy decoder form the input, while a side channel is used to provide supplementary information (such as coefficient location, the transform size and transform type) to the scheduler. The output is generated one row of pixels at a time.

The functionality of each module is described below.

**Scaling:** During the row transform phase, the incoming quantized transform coefficient is first sent through a scaling process. The HEVC specification defines a Quantization Parameter (QP) which takes on a value in the range [0,51]. The actual scaling factor is to be obtained via a further operation on this QP.

\[ s = c \times 2^{(QP/6)} \times levelScale[QP \mod 6] \gg (log_2(N) - 1) \]

where \( c \) is the input coefficient, \( s \) the scaled coefficient, \( levelScale \) is the 6 element array \([40,45,51,57,64,72]\) and \( N \times N \) is the transform size.

To save on the resources of a dedicated multiplier, 6 constant multipliers is used whose output is selected by \( QP \mod 6 \). (Figure 2(a)).

**MCM:** The output from scaling is sent to a Multiple Constant Multiplier (MCM) block. Both DCT and the DST matrices specified in the HEVC specification uses only 33 distinct coefficients (29 in DCT and 4 in DST). When 33 separate constant multipliers are used this required 1861 Look-Up-Tables (LUTs) when synthesized on a Xilinx Virtex 7 (485T) FPGA. Further optimizations are possible since the different multiplications could re-use the same sub-expressions. Spiral tool [11] was used to generate an adder tree that produces identical results with only 567 LUTs.

**Permute and Negate:** This block consists of 32 parallel instances of Fig. 2(b), where each instance consists of one 33-1 Mux (the permute part) and another 2-1 mux to select between the negated and the un-negated values. The selection for the muxes is done by the scheduler and depends on factors such as transform size, coefficient column (or row in case of row transform) and the type of transform (DST or DCT).
Parallel Adder/ Row Accumulate: Each of the 32 parallel adders take one input from the P&N block. The other input can come from different sources. If the current coefficient is the first in its column (or row), then no addition needs to be done. If its not the first in column in a column transform, the output from the memory is used and for a row transform the input is taken from the accumulator register (Fig. 2(d)).

Intermediate Memory: For a column transform, the output from the adders are written into the Intermediate Memory which is organized as 32 banks each with 32 entries of 29-bits (Fig. 2(c)). The write location for all banks are the same and is equivalent to the column (x coordinate) of the input coefficient. Note that unlike in prior work where 16-bit storage is sufficient, this design requires the intermediate coefficients be kept to full 29-bit precision due to its support for arbitrary input ordering of transformed coefficients.

During a row transform, the required coefficient is read from the intermediate memory (a mux being used to select the row) and produced to the MCM.

Saturate and Clip: For the HEVC main profile, the pixels are accurate to 8-bit precision, and hence the residuals should be accurate to 9-bits. This blocks makes sure the pixel value falls within the range (-256,255) and presents to the next pipeline stage (prediction) one row at a time.

III. SCHEDULING STRATEGY

This section describes a scheduling algorithm that is compatible with the above architecture. It is important to make the following observation regarding coefficient distribution at the end of each 1-D IDCT.

1. During the column transform, the effect of a non zero coefficient in the incoming matrix is limited to the column in which the coefficient is present. Therefore an empty column in the input matrix always yields an empty column in the intermediate matrix. However one non-zero coefficient is enough to trigger a fully filled column.

2. During the row transform stage, the number of iterations required per row is limited to the number of columns where non-zero coefficients are present in the intermediate matrix. This is equivalent to the number of columns with at least one non-zero coefficient in the original input matrix and will always be constant for each row.

These observations lead to a simple strategy for ensuring maximum utilization.

Strategy: During the column transform stage, the scheduler will keep track of which columns contain non-zero coefficients and only these will be fed to the row transform stage. 

Example: Consider a \( 4 \times 4 \) DCT transformed coefficient matrix with three non-zero coefficients (Fig 4(a)) . Assuming a diagonal coefficient scanning method, the entropy decoder would produce the coefficients in the order marked. (b)-(d) represents column transform and (e)-(h) represents the row transform. Since only 2 columns have non-zero coefficients in the input matrix, each row transform would consume 2 iterations.

The architecture is pipelined and has a throughput of 1 iteration per cycle. However it consumes 3 additional cycles when transitioning from the column transform to the row transform. This is a result of the requirement that the row transform cannot start until the final coefficient in the columns stage has gone through the pipeline and has written the outputs onto memory.

Based on this strategy, a single \( N \times N \) transform with \( c \) non-zero coefficients spanning across \( k \) \( (k \leq c \text{ and } k \leq N) \) columns consumes \( c + Nk \) iterations. \(^1\) \( (c + Nk + 3 \text{ cycles.}) \)

Unlike prior architectures where the throughput is fixed for a given resolution and frame rate, the minimum required frequency of the proposed scheme is highly dependent on the coefficient distribution. Figure 6 shows that the minimum required operating frequency is comfortably within 3\( \times \) the bit rate for Quad HD streams.

A noticeable property of this strategy is that the cost of an additional coefficient is significant when the distribution within the block is sparse. An empty \( 4 \times 4 \) transform takes zero cycles and a single non-zero coefficient increases this to 8. When encoded in a fixed bit rate configuration, sequences with high motion tend to cluster the residual coefficients into these changing regions. This is evident in the sequence ReadySetGo which has very high motion and requires a low operating frequency.

IV. SYNTHESIS RESULTS AND COMPARISON

The architecture has been optimized for FPGAs and upon synthesis on a Xilinx Virtex 7 - 485T, was found to consume

\(^1\)This is of course assuming that the entropy decoder can produce one non-zero coefficient per cycle. Even if it doesn’t, the inverse transform doesn’t consume inputs during the column transform stage and a FIFO placed before the scaler can be used to regulate the flow.
TABLE I: Comparison with prior work. The required frequency of operation is considered for the worst case (extrapolated to QHD 120 fps based on the numbers reported). For [9], the number is based on a calculation from the 4 coefficients per cycle figure adjusted by 27% downwards for the zero column skip effect. For this work 120 MHz is considered as an upper bound based on Fig. 6.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipelined / Multiplexed IDCT</td>
<td>Multiplexed</td>
<td>Multiplexed</td>
<td>Pipelined</td>
<td>Pipelined</td>
<td>Multiplexed</td>
</tr>
<tr>
<td>IDCT internal design</td>
<td>Multi size re-configurable</td>
<td>4 coefficients per cycle</td>
<td>1-D IDCT per cycle</td>
<td>1 coefficient per cycle</td>
<td>1 coefficient per cycle</td>
</tr>
<tr>
<td>Out of Order Coefficient Input</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Resources - ASIC</td>
<td>133.8k gates</td>
<td>125.8k gates + 24.6k SRAM</td>
<td>238.2k gates</td>
<td>80.2k gates + 32.7k SRAM</td>
<td>101k gates + 29.7k SRAM</td>
</tr>
<tr>
<td>Resources - FPGA</td>
<td>N/A</td>
<td>N/A</td>
<td>34.7k ALM</td>
<td>N/A</td>
<td>11.2k LUT</td>
</tr>
<tr>
<td>Required Operating Frequency for QHD 120 fps</td>
<td>1080 MHz</td>
<td>545 MHz</td>
<td>373 MHz</td>
<td>1648 MHz</td>
<td>120 MHz</td>
</tr>
</tbody>
</table>

The design was also synthesized with the Synopsys Design Compiler to facilitate comparison with prior work. Using the 90 nm educational library, the design consumed 101k NAND-2 equivalent gates and 29.7 kbits of SRAM.

A high level overview of how the proposed scheme stacks against prior work is in Table I. For a fair comparison, the required operating frequency for prior work is extrapolated to QHD 120 fps (Since the throughput for prior work is directly proportional to picture dimensions and frame rate, this extrapolation is straightforward). Even though the proposed method only operates on a single pixel/cycle, the fact that it does not process on zero coefficients make it very efficient and can be seen to achieve very high throughput at an acceptable area cost. However, as explained in section I, the other designs require a further buffer of 32.7 kbits be placed at the entropy decoder stage for coefficient reordering which is not accounted for in the above table.

V. CONCLUSION

We have proposed a hardware architecture and a scheduling strategy for the HEVC inverse transform that is very efficient on sparse matrices. This architecture can seamlessly be integrated with an entropy decoder without the need of a reordering buffer. Results show that the required frequency of operation, is significantly low compared to prior work in addition to being highly correlated to the input bit rate. This would have benefits in power saving and applications could be devised that could dynamically vary the frequency to match the input bit rate.

Figure 6: Required frequency of operation for 3840×2160 120 FPS streams encoded at different bitrates. The 4K sequences were obtained from [12].

REFERENCES


3One ALM (Adaptive Logic Module) in the Altera architecture can be used to implement up to 2 6-input LUTs.


