UNIFIED FORWARD+INVERSE TRANSFORM ARCHITECTURE FOR HEVC

Madhukar Budagavi, Vivienne Sze

Multimedia Lab, Systems and Applications R&D Center, Texas Instruments
madhukar@ti.com, sze@ti.com

ABSTRACT

The upcoming HEVC video coding standard supports many transform sizes ranging from 4-point to 32-point in square and rectangular form. Multiple transform sizes improve coding efficiency, but also increase the implementation complexity. Furthermore both forward and inverse transforms need to be supported in various consumer devices. This paper presents a unified forward+inverse transform architecture for HEVC. The unified architecture makes use of symmetry properties that exist in the HEVC forward and inverse transform matrices to achieve hardware sharing across different transform sizes and also between forward and inverse transforms. It uses 43-45% less area than separate forward and inverse core transform implementations.

Index Terms— Video coding, HEVC, core transform, hardware architecture.

1. INTRODUCTION

The Joint Collaborative Team on Video Coding (JCT-VC) is currently developing the next-generation video coding standard referred to as High Efficiency Video Coding (HEVC). HEVC is expected to provide around 50% reduction in bitrate (at similar visual quality) over the current standard, H.264/AVC, and intended for larger resolutions and higher frame rates. To address these requirements, HEVC utilizes larger block sizes than H.264/AVC. In HEVC, the largest coding unit (LCU) can be up to 64x64 in size and transform sizes of 4x4, 8x8, 16x16, 32x32, 16x4, 4x16, 32x8, 8x32 are supported. Multiple transform sizes improve compression performance, but also increase the implementation complexity.

With the proliferation of products such as camera phones, tablets, video-conferencing, set-top boxes with digital video recording feature, etc., it is necessary to support video capture in addition to video playback on the same device. As a result both forward and inverse transforms need to be implemented in the same device and techniques that can reduce the overall area of the hardware block that implements both forward+inverse transform are desirable.

HEVC supports a core transform [1] and an alternate transform used only for Intra 4x4 blocks [2]. The focus of this paper is on core transform implementation. The core transform is a DCT-like integer transform that can be represented by matrix multiplication. Unlike the H.264/AVC transform [3], the HEVC core transform has decoupled transform and quantization [4].

This paper highlights the symmetry properties of the core transform in HEVC [1] that can be used to reduce overall area of the forward+inverse transform block. Hardware results are presented to demonstrate area savings achieved by using these symmetry properties.

2. SYMMETRY WITHIN AND BETWEEN FORWARD AND INVERSE TRANSFORMS

HEVC *M*x*N* core transforms are implemented as *M*-point (*M*-pt) 1D transforms followed by *N*-pt 1D transforms. Both square and rectangular transforms can share the same 1D transform hardware. Hence 1D transforms are discussed in this paper. The 32-pt HEVC core transform matrix is defined by 31 8-bit constants (ignoring sign bits) [5] – *C*1, *C*2, ..., *C*31 – given by:

```
C1 = 90,  C2 = 90,  C3 = 90,  C4 = 89,  C5 = 88,
C6 = 87,  C7 = 85,  C8 = 83,  C9 = 82, C10 = 80,
C11 = 78, C12 = 75, C13 = 73, C14 = 70, C15 = 67,
C16 = 64, C17 = 61, C18 = 57, C19 = 54, C20 = 50,
C21 = 46, C22 = 43, C23 = 38, C24 = 36, C25 = 31,
C26 = 25, C27 = 22, C28 = 18, C29 = 13, C30 =  9,
C31 =  4.
```

The HEVC core transform has several useful symmetry properties that can be used to reduce implementation cost. These properties also exist for DCT:

1. Even-odd symmetry in transform matrix that can be utilized to reduce implementation complexity [6].
2. The 16-pt, 8-pt, 4-pt transform matrices are subsets of the 32-pt transform matrix. As a result of this symmetry, smaller sized transforms are embedded within larger size transforms and do not need separate implementation.
3. Symmetry between forward and inverse transform.

Property 1 has been well studied for implementing 8x8 IDCT in early video coding standards where as Properties 2 and 3 have now become important for practical applications because of the multiple transform sizes in HEVC and the need to support forward+inverse transform in same hardware block respectively. Use of all these three properties for a unified forward+transform implementation will be described in this section using 4-pt and 8-pt forward and inverse transform as examples. Similar symmetry property can be found in 16-pt and 32-pt transforms.

2.1. 4-pt Core transform

Let $M = [M0, M1, M2, M3]^T$ be the input vector and $P = [P0, P1, P2, P3]^T$ denote the output of the forward 4-pt transform. The forward 4-pt transform is defined by following equation: $M = D_4 P$ where D_4 is given by

$$D_4 = \begin{bmatrix} C16 & C16 & C16 & C16 \\ C8 & C24 & -C24 & -C8 \\ C16 & -C16 & -C16 & C16 \\ C24 & -C8 & C8 & -C24 \end{bmatrix} \quad \text{(Eq. 1)}$$

Even-Odd decomposition (also referred to as partial butterfly during HEVC development) of *N*-pt transform of an *N*-pt input consists of following three steps:

1. Add/subtract input to generate *N*-pt intermediate vector. (See Eq. 2).
2. Calculate even part of the output using $N/2 \times N/2$ subset of transform matrix obtained from even rows of transform matrix. (See Eq. 3).
3. Calculate odd part of the output using $N/2 \times N/2$ subset of transform matrix obtained from odd rows of transform matrix. (See Eq. 4).

Even-odd decomposition of forward 4-pt transform is given by (Eq. 2)-(Eq. 4):

Add/sub:

$$[K0, K1, K2, K3]^T = [M0 + M3, M1 + M2, M2 - M1, M3 - M0]^T \quad \text{(Eq. 2)}$$

Even part:

$$\begin{bmatrix} P0 \\ P2 \end{bmatrix} = \begin{bmatrix} C16 & C16 \\ C16 & -C16 \end{bmatrix} \begin{bmatrix} K0 \\ K1 \end{bmatrix} \quad \text{(Eq. 3)}$$

Odd part:

$$\begin{bmatrix} P1 \\ P3 \end{bmatrix} = \begin{bmatrix} -C24 & -C8 \\ C8 & -C24 \end{bmatrix} \begin{bmatrix} K2 \\ K3 \end{bmatrix} \quad \text{(Eq. 4)}$$

Or equivalent odd-part:

$$\begin{bmatrix} P3 \\ P1 \end{bmatrix} = \begin{bmatrix} -C24 & C8 \\ -C8 & C24 \end{bmatrix} \begin{bmatrix} K3 \\ K2 \end{bmatrix} \quad \text{(Eq. 5)}$$

Direct 4-pt transform using (Eq. 1) would require 16 multiplications. Even-odd decomposition on the other hand requires 8 multiplications. Figure 1(a) shows the architecture of forward 4-pt implementation using even-odd decomposition.

Inverse 4-pt transform is defined by following equation:

$$Y = D_4^T X \quad \text{(Eq. 6)}$$

where $X = [X0, X1, X2, X3]^T$ is the input vector and $Y = [Y0, Y1, Y2, Y3]^T$ is the output of 4-pt inverse transform.

Even-odd decomposition of inverse 4-pt transform is given by (Eq. 7) – (Eq. 9):

Even part:

$$\begin{bmatrix} Z0 \\ Z1 \end{bmatrix} = \begin{bmatrix} C16 & C16 \\ C16 & -C16 \end{bmatrix} \begin{bmatrix} X0 \\ X2 \end{bmatrix} \quad \text{(Eq. 7)}$$

Odd part:

$$\begin{bmatrix} Z2 \\ Z3 \end{bmatrix} = \begin{bmatrix} -C24 & C8 \\ -C8 & -C24 \end{bmatrix} \begin{bmatrix} X1 \\ X3 \end{bmatrix} \quad \text{(Eq. 8)}$$

Add/sub:

$$[Y0, Y1, Y2, Y3]^T = [Z0 - Z3, Z1 - Z2, Z1 + Z2, Z0 + Z3]^T \quad \text{(Eq. 9)}$$

Figure 1(b) shows the architecture of inverse 4-pt implementation using even-odd decomposition. Assuming 16-bit inputs, inverse transform uses 16-bit x 8-bit multipliers where as the forward transform uses 17-bit x 8-bit multipliers since the inputs get added first before multiplication in forward transform.

For a unified forward+transform implementation, additional symmetry between forward and inverse transform matrices can be exploited to further reduce area. Comparing (Eq. 3) and (Eq. 7), it can be observed that the even matrix of forward and inverse transform is identical. Comparing (Eq. 5) and (Eq. 8), it can be observed that the odd matrix of forward and inverse transform is identical. So a unified 4-pt hardware that implements both forward and inverse transform can share hardware block that implements even matrix (Eq. 3 and Eq. 7) and odd matrix (Eq. 5 and Eq. 8). Figure 2 shows the architecture of a unified forward+inverse 4-pt transform with even and odd matrix sharing (labeled as Even4 and Odd4 respectively in the figure). The add-sub network on left is used for forward transform (see also Eq. 2) where as the add-sub network on right is used for inverse transform (see also Eq. 9). The add-sub networks are labeled as AddSub4 in Figure 2. A control signal (inv_fwd_flag) selects whether the circuit behaves as a forward or an inverse transform. The unified architecture uses 17-bit x 8-bit multipliers.

2.2. 8-pt Core transform

Forward 8-pt transform is defined by following equation: $P = D_8 M$, where $M = [M0, \cdots, M7]^T$ is input and $P = [P0, \cdots, P7]^T$ is output, and D_8 is given by:

$$D_8 = \begin{bmatrix} C16 & C16 & C16 & C16 & C16 & C16 & C16 & C16 \\ C4 & C12 & C20 & C28 & -C28 & -C20 & -C12 & -C4 \\ C8 & C24 & -C24 & -C8 & -C8 & -C24 & C24 & C8 \\ C12 & -C28 & -C4 & -C20 & C20 & C4 & C28 & -C12 \\ C16 & -C16 & -C16 & C16 & C16 & -C16 & -C16 & C16 \\ C20 & -C4 & C28 & C12 & -C12 & -C28 & C4 & -C20 \\ C24 & -C8 & C8 & -C24 & -C24 & C8 & -C8 & C24 \\ C28 & -C20 & C12 & -C4 & C4 & -C12 & C20 & -C28 \end{bmatrix} \quad \text{(Eq.10)}$$

Even-odd decomposition is given by (Eq. 11) – (Eq. 13)

Add/sub:

$$[K0, K1, K2, K3]^T = [M0 + M7, M1 + M6, M2 + M5, M3 + M4]^T$$
$$[K4, K5, K6, K7]^T = [M4 - M3, M5 - M2, M6 - M1, M7 - M0]^T \quad \text{(Eq. 11)}$$

Even part:

$$\begin{bmatrix} P0 \\ P2 \\ P4 \\ P6 \end{bmatrix} = \begin{bmatrix} C16 & C16 & C16 & C16 \\ C8 & C24 & -C24 & -C8 \\ C16 & -C16 & -C16 & C16 \\ C24 & -C8 & C8 & -C24 \end{bmatrix} \begin{bmatrix} K0 \\ K1 \\ K2 \\ K3 \end{bmatrix} \quad \text{(Eq. 12)}$$

Odd part:

$$\begin{bmatrix} P1 \\ P3 \\ P5 \\ P7 \end{bmatrix} = \begin{bmatrix} -C28 & -C20 & -C12 & -C4 \\ C20 & C4 & C28 & -C12 \\ -C12 & -C28 & C4 & -C20 \\ C4 & -C12 & C20 & -C28 \end{bmatrix} \begin{bmatrix} K4 \\ K5 \\ K6 \\ K7 \end{bmatrix} \quad \text{(Eq. 13)}$$

Inverse 8-pt transform is defined by following equation where $X = [X0, \cdots, X7]^T$ is input and $Y = [Y0, \cdots, Y7]^T$ is output:

$$Y = D_8^T X \quad \text{(Eq. 14)}$$

Even-odd decomposition is given by (Eq. 15) – (Eq. 17):
Even part:

$$\begin{bmatrix} Z0 \\ Z1 \\ Z2 \\ Z3 \end{bmatrix} = \begin{bmatrix} C16 & C8 & C16 & C24 \\ C16 & C24 & -C16 & -C8 \\ C16 & -C24 & -C16 & C8 \\ C16 & -C8 & C16 & -C24 \end{bmatrix} \begin{bmatrix} X0 \\ X2 \\ X4 \\ X6 \end{bmatrix} \quad \text{(Eq. 15)}$$

Odd part:

$$\begin{bmatrix} Z4 \\ Z5 \\ Z6 \\ Z7 \end{bmatrix} = \begin{bmatrix} -C28 & C20 & -C12 & C4 \\ -C20 & C4 & -C28 & -C12 \\ -C12 & C28 & C4 & C20 \\ -C4 & -C12 & -C20 & -C28 \end{bmatrix} \begin{bmatrix} X1 \\ X3 \\ X5 \\ X7 \end{bmatrix} \quad \text{(Eq. 16)}$$

Add/sub:

$$[Y0, Y1, Y2, Y3]^T = [Z0 - Z7, Z1 - Z6, Z2 - Z5, Z3 - Z4]^T$$
$$[Y4, Y5, Y6, Y7]^T = [Z3 + Z4, Z2 + Z5, Z1 + Z6, Z0 + Z7]^T \quad \text{(Eq. 17)}$$

Comparing (Eq. 12) to (Eq. 1) and (Eq. 15) to (Eq. 6), it can be observed that the even matrix of 8-pt forward and inverse transform are identical to 4-pt forward and inverse transform matrix. Hence the unified architecture described in Section 2.1 can be used to implement the even part of both 8-pt forward and inverse transform.

Comparing (Eq. 13) and (Eq. 16), it can be seen that the odd matrix of forward and inverse 8-pt transform have identical multiplicands which differ in sign for half the elements shown by the circled elements in (Eq. 16).

Figure 3 shows the architecture of unified forward+inverse transform for 8-pt transform. The architecture is similar to that used for 4-pt transform. The even and odd matrix multiplications are shared between forward and inverse transform. Add-sub network on left is used for forward transform whereas add-sub network on right is used for inverse transform. Constant "s" in the odd-matrix is equal to 1 for inverse transform and equal to -1 for forward transform. The sum-of-product terms involving coefficients with and without "s" are calculated separately and the sign is changed in the final sum of product terms. The 16-pt and 32-pt unified transform architectures are similar to Figure 3.

Note that the final results of both forward and inverse transform are rounded before being stored. This is not explicitly shown in the figures but are included in hardware results in Section 3. The rounding circuit is also shared between forward and inverse transform.

3. HARDWARE RESULTS

The unified forward+inverse transform was implemented in RTL for a throughput of one 32-pt 1D transform per cycle. A 32x32 2D transform requires 64 cycles. Separate forward and inverse transforms were also implemented. The implementations were synthesized in 45-nm CMOS. Table 1 lists the area estimates (in kgates) at 250 MHz for separate and unified implementations. The unified implementation requires around 44% less area than separate implementation. Hardware area savings at other frequencies are provided in [7] and are in range of 43-45%.

Table 1: HEVC 32-pt 1D transform area estimate

Freq	Fwd	Inv	Separate Fwd+Inv	Unified Fwd+Inv	%Area savings
250	148	130	278	156	44%

Table 2 provides the area breakdown of different components of 32-pt transform. *N*-pt transform implementation consists of three main components: *N*/2-pt transform, Odd matrix multiplication circuit and add-sub network. Hence smaller size transforms do not need separate implementation leading to area savings.

Table 2: Area breakdown of different components in 32-pt transform

					% Area
32-pt	16-pt	8-pt	4-pt	Even4	0.2
				Odd4	1.0
				AddSub 4	0.8
			Odd8		5.7
			AddSub8		1.8
		Odd16			16.9
		AddSub16			3.2
	Odd32				54.0
	AddSub32				6.1
Mux, Demux, Rounding					10.3
Total					100

4. SUMMARY

HEVC uses multiple transform sizes to improve coding efficiency. This however increases complexity. In this paper, it was shown that symmetry properties of HEVC core transform enable hardware sharing across different transform sizes and also between forward and inverse transforms. Table 3 summarizes the hardware sharing between forward and inverse transform that enabled an area reduction of over 40%.

Table 3: Hardware sharing between forward and inverse transform in HEVC

	Shared	Not-shared
4-pt	Even4 matrix mult (Eq. 3 & 7) Odd4 Matrix mult (Eq. 5 & 8)	AddSub4 networks (Eq. 2 & 9)
8-pt	4-pt matrix mult (Eq. 12 & 15) Odd8 Matrix mult (Eq. 13 & 16)	AddSub8 networks (Eq. 11 & 17)
16-pt 32-pt	Even and odd (Odd16, Odd32) matrix mult	AddSub16,32 network

5. REFERENCES

[1] A. Fuldseth, G. Bjøntegaard, M. Budagavi, V. Sze, "CE10: Core transform design for HEVC," JCTVC-G495, Nov. 2011.

[2] A. Saxena, F. Fernandes, "Mode-dependent DCT/DST without 4*4 full matrix multiplication for intra prediction," JCTVC-E125, Mar. 2011.

[3] H. S. Malvar et. al., "Low-Complexity Transform and Quantization in H.264/AVC," IEEE Trans. CSVT, July 2003.

[4] M. Sadafale, M. Budagavi, "Low-complexity configurable transform architecture for HEVC," JCTVC-C226, Oct. 2010.

[5] M. Tikekar, et. al., "Core Transform Property for Practical Throughput Hardware Design," JCTVC-G265, Nov. 2011.

[6] C.-Yu Hung, P. Landman, "Compact inverse discrete cosine transform circuit for MPEG video decoding", IEEE SIPS, pp.364-373, Nov. 1997.

[7] M. Budagavi, V. Sze, M. Sadafale,"Hardware analysis of transform and quantization," JCTVC-G132, Nov. 2011.

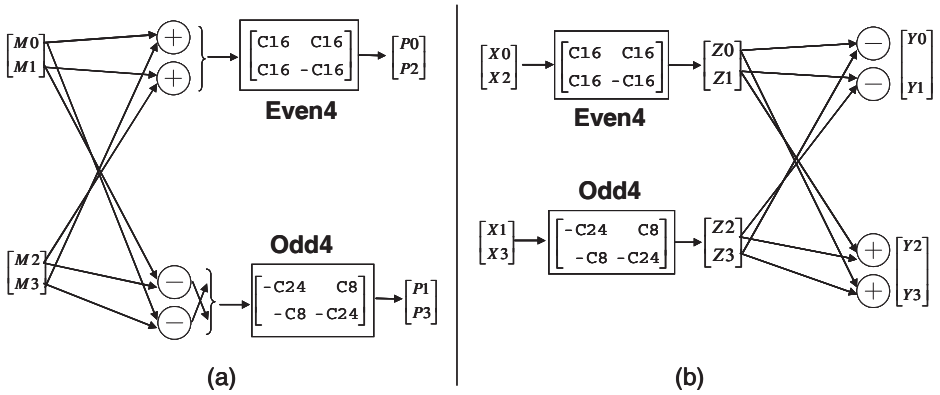

Figure 1: (a) Forward transform architecture for 4-pt transform. (b) Inverse transform architecture for 4-pt transform.

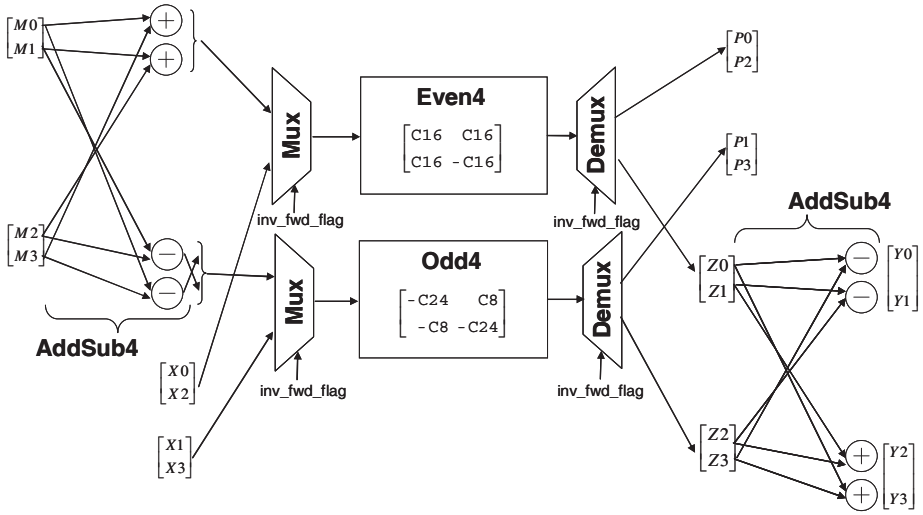

Figure 2: Unified forward+inverse 4-pt transform with single set of input and output ports.

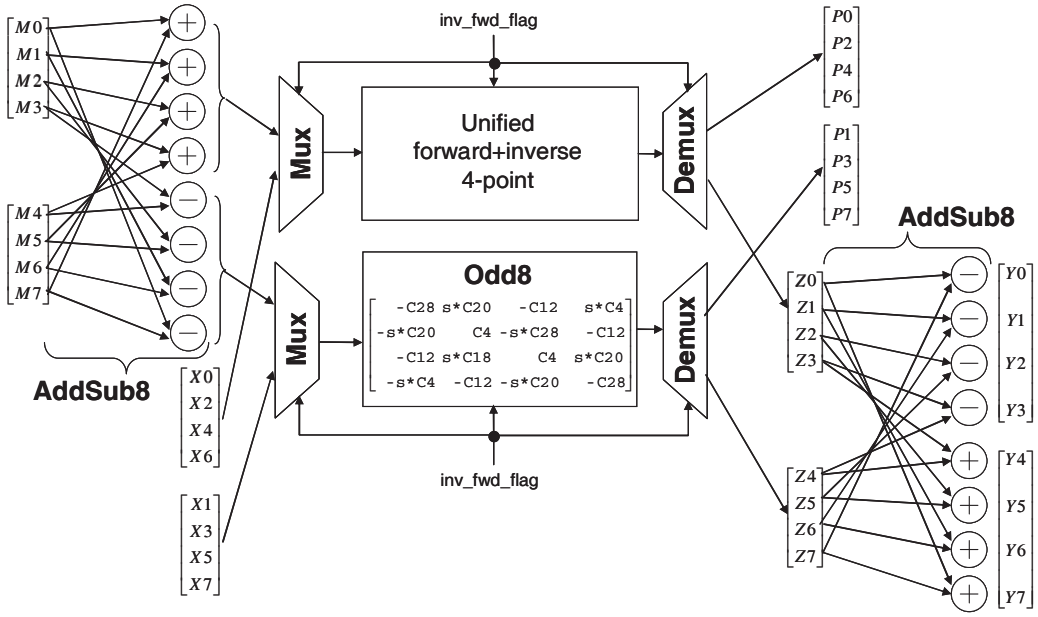

Figure 3: Unified forward+inverse 8-pt transform architecture.