High-Level Synthesis Implementation of Transform-Exempted SATD Architectures for Low-Power Video Coding

Tero Partanen, Ari Lemmetti, Panu Sjövall, Jarno Vanne
Ultra Video Group, Tampere University, Finland
{tero.partanen, ari.lemmetti, panu.sjovall, jarno.vanne}@tuni.fi

Abstract— **This paper presents the first known high-level synthesis (HLS) implementation for the Sum of Absolute Transformed Differences (SATD) calculation. The proposed hardware architecture is designed for two SATD algorithms: a widespread Fast Walsh-Hadamard Transform (FWHT-SATD) and a recently introduced Transform Exempted scheme (TE-SATD). This 2-stage architecture is made up of two 1-D Walsh-Hadamard Transform (WHT) stages and a transpose buffer (TB) between them. The chosen HLS approach cuts down design time over contemporary design methods and thereby made it feasible to implement a set of dedicated FWHT-SATD and TE-SATD architectures for 4×4, 8×8, and 16×16 pixel blocks. All these six architectures were synthesized for 28 nm and 45 nm standard cell technologies, and their area and energy consumptions were analysed. TE-based implementations provide 6.0-8.3% total cell area savings and 6.9-12.7% better energy-efficiency than traditional FWHT approaches. Our proposal is the first to introduce TE-SATD architectures for up to 16×16 blocks and each of these tailored architectures was shown to provide better trade-off between silicon area and performance than their reference implementations.**

Keywords—video coding, Sum of Absolute Transformed Differences (SATD), Hadamard transform, High-Level Synthesis (HLS), low-power hardware design

I. Introduction

The proliferation of high-quality and immersive media applications has led to the explosive growth of global video traffic. Therefore, each new video coding standard is aiming to introduce an improved coding efficiency that is substantially beyond that of the prevailing technology. The current mainstream *High Efficiency Video Coding* (*HEVC*) standard [1], [2] is able to halve the bit rate over its predecessor *Advanced Video Coding* (*AVC*) [3], [4] for the same subjective visual quality and the recently ratified *Versatile Video Coding* (*VVC*) [5] standard follows the same trend. However, improvements in coding efficiency tend to result in a large computational cost, which inevitably increases the power and energy dissipation of video codecs. In addition, today's video applications are more often performed on low-power mobile devices. Therefore, developing dedicated hardware architectures is vital to mitigate the increased energy consumption in video coding.

In block-based video coding, a plurality of blocks of $M \times N$ pixels is compared in order to find the most similar candidate block to the current block being encoded. Basically, the search process evaluates several available candidate blocks and the one that minimizes a dissimilarity metric will be chosen. *Sum of Absolute Differences* (*SAD*) [6] and *Sum of Absolute Transformed Differences* (*SATD*) [6] are the most common fast metrics to estimate similarity (or dissimilarity) between two different pixel blocks. In HEVC video coding, SAD is the most commonly used criterion in *integer motion estimation* (*IME*), whereas SATD is a preferred metric in *fractional motion estimation* (*FME*), intra search, and mode decision. On average, these coding tools together account for up to 40% of the whole HEVC encoder complexity.

SAD is simply the sum of absolute values applied to a prediction residual, whereas SATD first transforms the residual to different domain and then calculates the sum of absolute values. Usually, the *Walsh-Hadamard Transform* (*WHT*) [7] is used in SATD as the transforming function because calculation of WHT involves only additions and subtractions and allows thereby some optimizations. SATD tends to yield more accurate block estimates than SAD, because it better correlates with the *Discrete Cosine Transform* (*DCT*), which is a commonly used transform function in the well-known hybrid video coding scheme [8], [9]. However, the transform stage in SATD causes huge computational overhead. Therefore, it is vital to optimize SATD calculation method itself to make efficient hardware implementations possible. If SATD is used in FME or some other encoder stage, dedicated accelerators are recommended especially in mobile devices to meet strict constraints in execution time and power consumption [10].

It is possible to improve the energy efficiency of SATD by exploiting its mathematical properties. Traditionally, WHT is implemented using the *Fast Walsh-Hadamard Transform* (*FWHT*) [11] to reduce computational complexity. The FWHT can be implemented efficiently on hardware by dividing the 2-D transform into two 1-D transform stages and calculating both stages successively with butterfly-style adder/subtractor structure. Several FWHT-based SATD hardware architectures can be found in the literature [12]-[14]. The authors in [15], [16] investigated the usage of linear and transpose buffers between 1-D transform stages with square block sizes from 4×4 to 32×32. *Transpose Buffer* (*TB*) was reported to be more energy-efficient, although it results in greater chip area. The authors in [17], [18] utilized adder/subtractor compressors for butterfly operations and achieved about 10-14% reduction in energy consumption.

A more recent method to further reduce the number of arithmetic operations in Hadamard-based SATD is called a *Transform Exempted* (*TE*) method [19], [20]. In TE-SATD, the second 1-D stage exploits specific properties of absolute values to reduce arithmetic operations. The works in [21], [22] compared different 4×4 TE-SATD and butterfly-based FWHT-SATD architectures, of which TE-SATD architectures turned out to be more area and energy efficient. In [23], the authors introduced a combined 4×4 and 8×8 SATD hardware architecture, which deployed also TE-SATD. In that architecture, the transformed 4×4 block was reused for 8×8 SATD calculation. If both block sizes were required in coding, this approach was shown to save operations and buffers resulting in reduced energy consumption over separate 4×4 and 8×8 SATD architectures. However, in a case where both block sizes are not needed, this approach results in hardware overhead.

Our work proposes separate FWHT-SATD and TE-SATD hardware architectures for block sizes of 4×4, 8×8, and 16×16. They were all implemented using Algorithmic C (C++) and Catapult *high-level synthesis* (*HLS*) tool [24]. To the best of our knowledge, this is the first work that implements SATD units with the HLS design method [25] that is an emerging approach to raise the abstraction level over that of traditional *Hardware Description Languages* (*HDL*) such as *VHDL* and *Verilog*. By using HLS, designers can get 4-6 times increase in productivity over manual *register-transfer level* (*RTL*) flows [26].

The rest of the paper is organized as follows: Section 2 describes the adopted FWHT-SATD and TE-SATD algorithms and Section 3 details the proposed 2-stage architectures for them. Section 4 compares power dissipation and cell areas of these architectures using 28 nm technology. In addition, the respective area and throughput results are reported with 45 nm technology for comparison with related works. Section 5 concludes the paper.

II. SATD Calculation

The SATD is defined as

$$SATD(D_{M\times N}) = c \times \sum_{i=1}^{M}\sum_{j=1}^{N}\left|td_{i,j}\right|, \quad (1)$$

where $c \in \mathbb{R}_+$ is a scaling constant, $D_{M\times N}$ is the pixel matrix of a residual block (difference between candidate and current block being coded), and the $td_{i,j}$ denotes the (i, j)th element of the 2-D transformed residual block

$$T(D_{M\times N}) = T_{M\times N} \times D_{M\times N} \times T^T_{M\times N}. \quad (2)$$

$T_{M\times N}$ is the transform matrix of an integer linear transform, usually WHT. It is derived from a generalized class of the Fourier transform and it uses a Hadamard matrix of size $2^n \times 2^n, n \in \mathbb{R}_+$ [27] as a transform matrix. 1-D *Hadamard transform* (*HT*) is defined as

$$HT(D_{2^n\times 1}) = H_{2^n\times 2^n} \times D_{2^n\times 1}, \quad (3)$$

and 2-D HT is correspondingly defined as

$$HT(D_{2^n\times 2^n}) = H_{2^n\times 2^n} \times D_{2^n\times 2^n} \times H^T_{2^n\times 2^n}, \quad (4)$$

where $H_{2^n\times 2^n}$ is

$$H_{2^n\times 2^n} = \begin{cases} \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, if\ n = 1 \\ \frac{1}{2^{\frac{n}{2}}}\begin{bmatrix}H_{2^{n-1}\times 2^{n-1}} & H_{2^{n-1}\times 2^{n-1}}\\ H_{2^{n-1}\times 2^{n-1}} & -H_{2^{n-1}\times 2^{n-1}}\end{bmatrix}, if\ n > 1. \end{cases} \quad (5)$$

As expressed in (5), the Hadamard matrices are symmetric, i.e., $H^T = H$. The scaling constant c in (1) is defined for WHT-based SATD as

$$c = \frac{1}{2^n} \quad (6)$$

A naïve method to calculate the transform in (4) would be to use basic matrix multiplications, which have computational complexity of $O(m^3), where\ m = 2^n$. Even though n is usually quite small, the 2-D transform requires two matrix multiplications resulting in a large number of multiplications and additions. In the case of Hadamard matrices, which are composed of only +1 and -1 elements, multiplications can be omitted and only additions and subtractions are needed. But, still the number of operations remains quite high $(m^3 - m^2)$. For example, 448 additions are still needed for one matrix multiplication when $n = 3$.

FWHT is a more efficient method, where the 2-D transform is decomposed into two 1-D steps, namely, rows and columns [7]. In other words, the first 1-D step in (4) would be equal to rows and the second step equal to columns. The associative property of matrix multiplication holds as (HD)H = H(DH), so the order in which the steps are calculated is negligible. Both steps have *n* levels of adders represented by a butterfly structure and each *n* levels has *m* adders to transform one row or column of the residual matrix [7]. Butterfly is constructed so that Hadamard matrices are recursively split into two smaller matrices. The residual matrix is transformed, e.g., by calculating the first step in row-major order and then the second step in column-major order. FWHT has $m^2 \times log_2(m)$ additions for one step. For *n* = 3, it means 192 additions which is 57% less than the ordinary matrix multiplication with entry multiplications omitted. After the transform computation, m^2 absolute operations and $m^2 - 1$ additions are required to complete the SATD. Therefore, the total operation count for FWHT-SATD is $2 \times m^2 \times log_2(m) + m^2 - 1$ additions (447 additions for *n* = 3) and m^2 absolute operations (64 operations for *n* = 3).

In recently introduced TE-SATD method [19], [20], the first 1-D step is computed as in FWHT, but the second 1-D step combines sum, absolute, and compare operations using Property 1 [19]:

$$|d_1 + d_2| + |d_1 - d_2| = 2 \times \max(|d_1|, |d_2|).$$

The left-hand side of Property 1 is substituted by right-hand side to reduce operation counts without impacting the SATD result. In practice, the 2 × multiplication can also be omitted because of the scaling constant (6). The TE-SATD replaces the 1-D Hadamard transform and the sum of absolutes.

1-D SATD can be defined by

$$\begin{aligned} SATD(D_{2^n\times 1}) &= SA\big(T(D_{2^n\times 1})\big) \\ &= SA(H_{2^n\times 2^n} \times D_{2^n\times 1}). \quad (7) \end{aligned}$$

Applying matrix partition and (5) yields

$$\begin{aligned} &SA(H_{2^n\times 2^n} \times D_{2^n\times 1}) \\ &= SA\left(\begin{bmatrix}H_{2^{n-1}\times 2^{n-1}} & H_{2^{n-1}\times 2^{n-1}}\\ H_{2^{n-1}\times 2^{n-1}} & -H_{2^{n-1}\times 2^{n-1}}\end{bmatrix}\begin{bmatrix}D_{1,2,\dots,\frac{2^n}{2}}\\ D_{\frac{2^n}{2}+1,\dots,2^n}\end{bmatrix}\right) \\ &= SA\left(H_{2^{n-1}\times 2^{n-1}}\left(D_{1,2,\dots,\frac{2^n}{2}} + D_{\frac{2^n}{2}+1,\dots,2^n}\right)\right) \\ &+ SA\left(H_{2^{n-1}\times 2^{n-1}}\left(D_{1,2,\dots,\frac{2^n}{2}} - D_{\frac{2^n}{2}+1,\dots,2^n}\right)\right). \quad (8) \end{aligned}$$

Equation (8) shows how SATD can be calculated using the lower order SATDs. However, calculating SATD for two different block sizes at once can introduce some computational and hardware overhead. Therefore, this work implements separate SATD units for each block size.

In [19], 1-D TE-SATD for $D_{4\times1} = [d_1, d_2, d_3, d_4]^T$ was derived as

$$TE\text{-}SATD(D_{4\times1}) = 2 \times (max(|d_1 + d_3|, |d_2 + d_4|) + max(|d_1 - d_3|, |d_2 - d_4|)) \quad (9)$$

In this work, recursive property shown in (8) was also used with 8×8 and 16×16 blocks. For 8×8 block size, TE-SATD is computed for $D_{8\times1} = [d_1, \dots, d_8]^T$ as

$$\begin{aligned} TE\text{-}SATD(D_{8\times1}) &= SA(H_{8\times8} \times D_{8\times1}) \\ &= SA\left(\begin{bmatrix} H_{4\times4} & H_{4\times4} \\ H_{4\times4} & -H_{4\times4} \end{bmatrix}\begin{bmatrix} D_{1,2,3,4} \\ D_{5,6,7,8} \end{bmatrix}\right) \\ &= SA\left(H_{4\times4}(D_{1,2,3,4} + D_{5,6,7,8})\right) \\ &\quad + SA\left(H_{4\times4}(D_{1,2,3,4} - D_{5,6,7,8})\right), \quad (10) \end{aligned}$$

which is composed of two 4×1 SATD calculations and by applying (9)

$$\begin{aligned} TE\text{-}SATD(D_{8\times1}) = 2 \times & \\ [max(|(d_1 + d_5) + (d_3 + d_7)|, |(d_2 + d_6) + (d_4 + d_8)|) & \\ + max(|(d_1 + d_5) - (d_3 + d_7)|, |(d_2 + d_6) - (d_4 + d_8)|) & \\ + max(|(d_1 - d_5) + (d_3 - d_7)|, |(d_2 - d_6) + (d_4 - d_8)|) & \\ + max(|(d_1 - d_5) - (d_3 - d_7)|, |(d_2 - d_6) - (d_4 - d_8)|)] & \\ (11) & \end{aligned}$$

Finally, the 8×8 TE-SATD is computed as

$$SATD(D_{8\times8}) = SA(H_{8\times8} \times D_{8\times8} \times H_{8\times8}^T), \quad (12)$$

in two 1-D steps by first calculating $R_{8\times8} = D_{8\times8} \times H_{8\times8}^T$ as in FWHT and then applying (11) for each column of $R_{8\times8}$.

Similarly, (8) and (11) were applied to derive the TE-SATD for 16×16 block size, but the respective equations are omitted in this paper for brevity.

Table I summarizes the operation counts for complete FWHT-SATD and TE-SATD calculations. For example, in the case of 8×8 TE-SATD the final 1-D step has 255 operations (159 additions, 64 absolutes, and 32 compares). In FWHT, the last 1-D step takes 319 operations. Thus, TE-SATD saves 64 operations (13%).

III. Proposed Hardware SATD Implementation

Since 2-D WHT is orthogonal, it is inherently separable and can be decomposed into two successive 1-D WHT stages. However, the second WHT stage uses the output of the first stage, so a buffer is needed between them if the transforms are not fully parallel. Several architecture explorations between parallel and buffer-based schemes can be found in the literature [15], [16], [21], [22]. A two-stage architecture with a TB [12] between the stages has proven to be a good trade-off between energy consumption and area, so it was selected as basis for this work.

Fig. 1 depicts a conceptual block diagram of our TB-based SATD hardware architecture. It supports data blocks of size $2^n \times 2^n$, where n = 2, 3, 4. All these architectures are pipelined and they output a new SATD result in every 2^n clock cycles. Altogether, six different hardware implementations were realized, i.e., separate FWHT-SATD and TE-SATD architectures for the block sizes of 4×4, 8×8, and 16×16.

Table I. Number of Operations in SATD Calculation

Operation	4 x 4 SATD FWHT	4 x 4 SATD TE	8 x 8 SATD FWHT	8 x 8 SATD TE	16 x 16 SATD FWHT	16 x 16 SATD TE
Add	79	55	447	351	2303	1903
Abs	16	16	64	64	256	256
Cmp		8		32		128
Total	95	79	511	447	2559	2287
Saving		-17 %		-13 %		-11 %

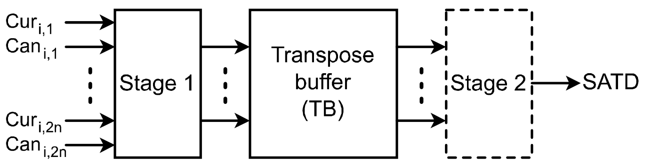

Fig. 1. A conceptual block diagram of the proposed SATD architecture.

The Stage 1 takes as input the *candidate* (*Can*) and *current* (*Cur*) data blocks row-by-row. It calculates first the residuals and then the 1-D WHT using a butterfly structure. The Stage 1 has input registers, but all data calculations are combinational.

The TB is a transposing shift register array [12] of size $2^n \times 2^n$. It changes shift direction every 2^nth cycle, i.e., at the beginning of a new block. Thus, it allows fully pipelined operation on a quite simple hardware structure.

The Stage 1 and TB are identical to the FWHT-SATD or TE-SATD algorithms, but the Stage 2 has to be tailored to either of them. It calculates the second 1-D Hadamard transform and accumulates the final sum of absolute values. In the FWHT-SATD architecture, the second 1-D Hadamard transform is computed using the butterfly structure as in the Stage 1 and the resulting absolute values are accumulated as in (1). In the proposed TE-SATD architecture, the Stage 2 is made up of addition and absolute operations, but also compare operations (right-hand side of Property 1), which are basically implemented as multiplexers. Finally, the SATD result, stored in separate registers, is the output of the architectures.

Having the design implemented with Catapult HLS tool, several advantages were attained over a traditional RTL flow. As the RTL code is generated from the HLS code, it was straightforward to generate all six SATD architectures with minimal changes to the HLS code. In addition, because HLS code is technology independent, it is possible to generate optimized RTL for both ASIC and FPGA synthesis by only changing the target from Catapult HLS. In this work, only the ASIC results are presented. As an example, for RTL optimizations, Catapult HLS calculates and modifies the throughput of the design based on the desired clock frequency. Depending on timing and the used technology, the number of combinatorial operations that Catapult allocates per clock cycle varies. The state machine of the architecture can be pipelined by the tool for the best throughput.

TABLE II. SYNTHESIS RESULTS FOR 28 NM PROCESS TECHNOLOGY, 2.5 NS CLOCK PERIOD, AND 0.95 V SUPPLY VOLTAGE

Architecture	Size	Cycles/ SATD	Cell Area (μm^2)	Area Saving	Cell Internal Power	Net Switching Power	Total Dynamic Power	Total Power (μW)	Energy/SATD (pJ/SATD)	Energy Saving
FWHT-SATD (baseline)	4×4	4	1315		655	235	889	890	8.9	
TE-SATD (proposed)			1206	-8.3%	598	179	777	777	7.8	-12.7%
FWHT-SATD (baseline)	8×8	8	4465		2386	768	3154	3154	63.1	
TE-SATD (proposed)			4198	-6.0%	2247	632	2879	2879	57.6	-8.7%
FWHT-SATD (baseline)	16×16	16	17969		9494	2320	11815	11818	472.7	
TE-SATD (proposed)			16747	-6.8%	8799	2196	10994	10997	439.9	-6.9%

IV. RESULTS

All the proposed SATD architectures were described in Algorithmic C and converted to RTL using Catapult Ultra Synthesis 10.5b tool. The RTL designs were synthesized with Synopsys Design Compiler [28] and evaluated on a 28 nm ASIC standard cell technology. The process technology was *Fully Depleted Silicon On Insulator* (*FD-SOI*) with 0.95 V supply voltage, typical process corner, and 25°C temperature.

Switching activity interface format (*SAIF*) were used to estimate the power consumption of the synthesized designs. The files were produced using Questa Sim [29] inside the Catapult power estimation flow. The SAIF files contained real SATD inputs that were collected by encoding CityAlley, ReadySetGo, and ShakeNDry 4K test video sequences [30] with Kvazaar open-source HEVC encoder [31]. Architectures were synthesized for 2.5 ns clock period that was estimated to provide adequate SATD throughput for a full search FME algorithm in real-time (60 fps) 4K HEVC encoding [23].

The synthesis results for 28 nm technology, in terms of area, power, and energy, are presented in Table II. The results reported for the FWHT-SATD architectures serve as a baseline to highlight improvements of the proposed TE-SATD architectures. Cell leakage power was negligible thus it is omitted from the results. The proposed TE-SATD architectures resulted in 6.0-8.3% smaller total cell area and 6.9-12.7% lower energy consumption than the FWHT-SATD baseline architectures. Hence, switching from FWHT-SATD to TE-SATD gives relatively significant savings considering that the Stage 1 and TB are identical in these two architectures. Because Catapult generated a flattened RTL designs, hierarchical area or power results are not analysed. However, synthesis results for each FWHT-SATD architecture declare that about 65% of the total area is used for registers and 35% for combinatorial logic. Thus, it can be estimated that approximately 60% of the total area is consumed by the TB. Considering this, TE method brings relatively substantial area and power savings.

It is not straightforward to compare our results with prior works because they used mixed supply voltages and distinct target clock periods. However, area results for 45 nm technology are reported in some works, so the proposed TE-SATD architectures were also synthesized for 45 nm NanGate standard cell technology [32] using 2.5 ns clock period. Table III presents these results for the proposed and related TB-based SATD architectures. In comparison with the existing 4×4 SATD architectures [15], [22], the proposed solution has competitive area, although it only takes one-fifth and one-fourth of their clock periods, respectively. In the case of 8×8 SATD architectures, the proposed work is the smallest. Furthermore, a combined 4×4 & 8×8 SATD architecture in [23] resulted larger area than the aggregated area of the proposed 4×4 and 8×8 SATD architectures. The 16×16 SATD architecture in [15] has 25 times as long clock period as ours, but still the area results are similar. The throughput determined by computed blocks per second (blocks/s) is mainly comprised from target period, thus our proposal outperforms most of the related works with high margins.

TABLE III. COMPARISON OF THE PROPOSED AND EXISTING SOLUTIONS

Work	Size	Method	Library (45nm)	Target Period (ns)	Area (μm^2)	Blocks/s
[15]	4×4	FWHT	TSMC	15.63	2943*	16M
[22]	4×4	FWHT	TSMC	10.40	2600**	16M
[22]	4×4	TE	TSMC	10.40	2500**	16M
Proposed	4×4	TE	NanGate	2.50	2541	100M
[18]	8×8	FWHT	NanGate	3.00	14833	41.6M
[14]	8×8	FWHT	NanGate	1.69	12231	72M
[15]	8×8	FWHT	TSMC	31.25	9468*	4M
[23]	4×4 & 8×8	TE	TSMC	2.50	11935	50M
Proposed	8×8	TE	NanGate	2.50	9337	50M
[15]	16×16	FWHT	TSMC	62.50	33000**	1M
Proposed	16×16	TE	NanGate	2.50	33667	25M
* Precise value from [23]; ** Estimated value from a graph						

V. CONCLUSIONS

This paper presented the first known HLS implementation of the FWHT-SATD and TE-SATD architectures for block sizes of 4×4, 8×8, and 16×16 pixels. TE-SATD method has earlier been proven to decrease the needed operations in SATD calculation so it can potentially improve the energy efficiency in video coding. Our results confirm the advantages of the TE-SATD scheme over that of the FWHT-SATD. Synthesis results for 28 nm technology showed that SATD-architectures implementing the TE-method can save 6.0-8.3% on cell area and 6.9-12.7% on consumed energy. In addition, the proposed architectures were synthesized for 45 nm technology and the results denoted that TE-SATD architectures designed with HLS offer better area-performance trade-off than the existing implementations. Hence, replacing traditional HDL design approach with the HLS flow not only increased design productivity and code reusability in this case, but also performance was at least equal to that of the traditional HDL design flow.

ACKNOWLEDGMENT

This work was supported in part by the European ECSEL project ADACORSA (under the grant agreement 876019).

References

[1] *High Efficiency Video Coding document ITU-T Rec. H.265 and ISO/IEC 23008–2 (HEVC)*, ITU-T and ISO/IEC, Nov. 2019.

[2] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1649-1668.

[3] *Advanced Video Coding for Generic Audiovisual Services document*, ITU-T Rec. H.264 and ISO/IEC 14496–10 (AVC) ITU-T and ISO/IEC, Mar. 2009.

[4] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders," *in Proc. Picture Coding Symp.*, San Jose, California, USA, Dec. 2013.

[5] B. Bross, J. Chen, S. Liu, and Y. K. Wang, "Versatile Video Coding (Draft 10)," *document JVET-P2001,* July 2020.

[6] I. E. Richardson, *The H.264 Advanced Video Compression Standard, Second Edition*, John Wiley & Sons Ltd, 2010.

[7] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *IEEE*, vol. 57, no. 1, Jan. 1969, pp. 58-68.

[8] T. Wiegand and H. Schwarz, "Video coding: Part II of fundamentals of source and video coding," in *Video Coding: Part II of Fundamentals of Source and Video Coding,* Nov. 2016.

[9] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Performance and computational complexity assessment of high-efficiency video encoders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1899–1909.

[10] J. Vanne, M. Viitanen, T. D. Hämäläinen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1885–1898.

[11] B. J. Fino and V. R. Algazi, "Unified matrix treatment of the fast walsh-Hadamard transform," *IEEE Trans. Comput.*, vol. 25, no. 11, Nov. 1976, pp. 1142-1146.

[12] T. C. Wang, Y. W. Huang, H. C. Fang, and L. G. Chen, "Parallel 4 ×4 2D transform and inverse transform architecture for MPEG-4 AVC/H.264," *in Proc. Int. Symp. Circuits Syst.*, Bangkok, Thailand, May 2003.

[13] J. S. Dominges Jr, V. N. Possani, D. S. Silveira, L. S. da Rosa Jr, and L. V. Agostini, "High throughput 4×4 and 8×8 SATD similarity criteria architectures for video coding applications," *in Proc. IEEE Southern Conf. Programmable Logic,* Córdoba, Argentina, Apr. 2011.

[14] E. Silveira, C. Diniz, M. B. Fonseca, and E. Costa, "SATD hardware architecture based on 8×8 Hadamard transform for HEVC encoder," *in Proc. IEEE Int. Conf Electron. Circuits Syst.*, Cairo, Egypt, Dec. 2015.

[15] I. Seidel, A. Beims Bräscher, J. L. Güntzel, and L. Agostini, "Energy-efficient SATD for beyond HEVC," *in Proc. IEEE Int. Symp. Circuits Syst.*, Montreal, Canada, May 2016.

[16] A. B. Bräscher, I. Seidel, and J. L. Güntzel, "Improving the energy efficiency of a low-area SATD hardware architecture using fine grain PDE," *in Proc. Symp. Integr. Circuits Syst. Des.*, Fortaleza, Brazil, Aug. 2017.

[17] B. Silveira, R. Ferreira, G. Paim, C. Diniz, and E. Costa, "Low power SATD architecture employing multiple sizes Hadamard transforms and adder compressors," *in Proc. IEEE Int. New Circuits Syst. Conf.*, Strasbourg, France, June 2017.

[18] B. Silveira, B. Abreu, G. Paim, M. Greller, R. Ferreira, C. Diniz, E. Costa, and S. Bampi, "Using adder and subtractor compressors to sum of absolute transformed differences architecture for low-power video encoding, " *in Proc. IEEE Int. Conf. Electron., Circuits Syst.*, Batumi, Georgia, Dec. 2017.

[19] C. Zhu and B. Xiong, "Transform-exempted calculation of sum of absolute Hadamard transformed differences," in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 8, Aug. 2009, pp. 1183-1188.

[20] F. D. Jou, "Method for fast SATD estimation," U.S. Patent 0 198 622 A1, Aug. 23, 2007.

[21] L. H. Cancellier, A. B. Bräscher, I. Seidel, and J. L. Güntzel, "Energy-efficient Hadamard-based SATD architectures," *in Proc. Symp. Integr. Circuits Syst. Des.*, Aracaju, Brazil, Sept. 2014.

[22] L. H. Cancellier, I. Seidel, A. B. Brascher, J. L. Guntzel, and L. Agostini, "Exploring optimized Hadamard methods to design energy-efficient SATD architectures," *J. Integr. Circuits Syst.*, vol. 10, no. 2, Aug. 2015, pp. 113-122.

[23] I. Seidel, M. Monteiro, B. Bonotto, L. V. Agostini, and J. L. Güntzel, "Energy-efficient Hadamard-based SATD hardware architectures through calculation reuse," *IEEE Trans. Circuits Syst.*, vol. 66, no. 6, June 2019, pp. 2102-2115.

[24] Catapult High-Level Synthesis: Overview. [Online] Available: https://www.mentor.com/hls-lp/catapult-high-level-synthesis

[25] P. Coussy, D. Gajski, M. Meredith, and A. Takach, "An introduction to high-level synthesis," *IEEE Des. Test. Comput.*, vol. 26, no. 4, July–Aug. 2009, pp. 8-17.

[26] S. Lahti, P. Sjövall, J. Vanne, and T. D. Hämäläinen, "Are we there yet? A study on the state of high-level synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 5, May 2019, pp. 898-911.

[27] S. W. Golomb and L. D. Baumert, "The search for Hadamard matrices," *The American Mathematical Monthly*, vol. 70, no. 1, pp. 12–17, Jan. 1963.

[28] Synopsys. Synopsys Design Compiler, Version Q-2019.12-SP1. [Online] Available: https://www.synopsys.com/

[29] Mentor. Mentor Questa Sim, Version 10.7c 2018.08. [Online] Available: https://www.mentor.com/

[30] A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120fps 4K sequences for video codec analysis and development," *in Proc. ACM Multimedia Syst. Conf.*, Istanbul, Turkey, June 2020.

[31] A. Lemmetti, M. Viitanen, A. Mercat, and J. Vanne, "Kvazaar 2.0: fast and efficient open-source HEVC inter encoder," in *Proc. ACM Multimedia Syst. Conf.*, Istanbul, Turkey, June 2020.

[32] Nangate 45nm. FreePDK45 NanGate Open Cell Library, version PDKv1.3 v2010_12