REVIEW ON VLSI ARCHITECTURE FOR IMAGE COMPRESSION THROUGH DCT STRUCTURE MINIMIZATION

Abhishek kumar verma[1], Ambresh Patel[2]
M.Tech (VLSI) Scholar, RKDF University Bhopal, India[1]
Assistant Professor, Department of Electrical Engineering, RKDF University Bhopal, India[2]

Abstract

Data compression is pivotal in multimedia devices to convey information concisely. Initially, the DCT structure was employed for image compression, offering lower complexity and efficient space utilization. Similarly, 2D DCT achieved reasonable data compression, but its implementation raised concerns due to increased multipliers and adders, leading to higher area requirements and power usage. Addressing these aspects, this study focuses on a VLSI architecture for image compression using a Rom-free DA-based DCT structure. This technique excels in high throughput and real-time implementation. To achieve this, the image matrix is partitioned into odd and even terms, and multiplication operations are eliminated through a shift-and-add approach. Kogge_Stone_Adder techniques are introduced to attain bit-wise image quality, introducing new trade-off levels compared to prior methods. Overall, this architecture yields reduced memory usage, low power consumption, and high throughput. MATLAB serves as the tool for input pixel reception and output image generation. The design is implemented using Verilog HDL, simulated with Model Sim, and synthesized and analyzed for power and area using Quartus II..

Keywords:

Distributed Arithmetic-Discrete Cosine Transform (DA-DCT), Kogge_Stone_Adder (KSA), Inverse Discrete Cosine Transform (IDCT), Very Large Scale Integrated Circuit (VLSI)

1. INTRODUCTION

Navigating the quest for efficient and cost-effective compression within multimedia systems remains a complex puzzle. Vital cogs in these systems, image and video applications, come with a hunger for data processing that demands attention. Moreover, the surge in mobile devices fuels the urgency of ensuring seamless multimedia functionality on the go. Given the bulky nature of multimedia files, carving out space on hard disks is no small feat. The vibrant realm of imaging and video applications rides a wave of rapid expansion, necessitating hefty data loads for image transformations, adding to memory consumption. Enter compression techniques, an attempt to streamline image and video applications. They sculpt file sizes, making them amenable to storage and distribution. This process is a dance of removing repetitive or extraneous data, offering a concise snapshot of the file's core essence. Confronting the magnitude of multimedia data, data compression methods come to the rescue. The journey begins with images morphing into coefficients, then gracefully normalizing. This sets the stage for the grand finale: reconstructed images as the payoff. In the orchestra of compression methods, DCT [1], [2] shines as a stalwart. Amidst them, DA-DCT (Distributed Arithmetic-based DCT) emerges, gaining ground as a twin-flag solution of compact representation and speedy processing. In a world riding high on multimedia needs, the scales of condensed Discrete Cosine Transform) ROM-based multipliers collaborate with adders to accumulate partial products, a technique that trims down area usage. However, ROM-based DCT introduces redundancy, prompting the innovation of a ROM-free DA-based DCT coupled with Parallel Prefix Adder (PPA) methodology. This approach harnesses reconfigurable odd and even DCT architectures within the DA framework. The proposed DCT architecture, governed by level parameters, can fluidly transition between trade-off levels with minimal overhead. When level control signals assume a 'ONE' state, the circuit operates akin to a standard DCT processor, retaining DCT bases intact. Conversely, when these signals are 'zeroes,' certain adders turn into zeroes, and others are powered down, resulting in memory reduction. When 2D-DCT is employed for image compression, a surge in adder and multiplier rates exposes digital errors. This issue is countered by the application of Parallel Prefix Adders (Kogge-Stone Adders) owing to their block compression and speed advantages.

Executing 2D-DCT and its inverse in VLSI design demands precise outcomes. The data processing transpires in a digital realm. Compressed images reside in a lossy compression state, implying that restored images deviate from the originals. Reconstructed image quality is gauged via Peak Signal to Noise Ratio, assessing varied images. This paper introduces an effective DA-based VLSI architecture for DCT, capitalizing on redundancy. Section 2 delves into the design specifics of the DA-based DCT architecture. The third section showcases the proposed Parallel Prefix Adder. Implementation and comparative outcomes concerning adders, area, and power consumption are deliberated in section 4.

2. DESIGN OF DA-BASED DCT STRUCTURE

The Discrete Cosine Transform (DCT) stands as a pivotal compression technique, thanks to its near-optimal performance and unmatched energy compaction efficiency among various transforms. The transformation algorithm's exposition can be found in reference [6].

2.1 DCT ARCHITECTURE

Utilizing the DCT architecture followed by the DWT yields several advantages, including elevated throughput, reduced complexity, and the elimination of complex number manipulation. When performing the computation for a 2D DCT, a substantial count of multipliers and adders is essential to enforce intricate compression organization. This intricate procedure, being notably time-consuming, can be entirely circumvented with

2.2 DA-BASED DCT

Distributed Arithmetic (DA) stands out as an effective technique for inner product computations. It leverages lookup tables and replaces accumulators with multipliers for inner product calculations within DCT. The DA-based DCT architecture is well-regarded in VLSI implementation circles due to its capacity to decrease ROM size, leading to area savings [8]. The architecture utilizes even-odd frequency decomposition of the DCT, augmenting its efficiency through memory reduction. The construction of 1D 8-point DCT involves a DA-Butterfly-Matrix featuring even and odd processing elements, alongside the integration of Parallel Prefix Adder (Kogge-Stone Adder), illustrated in Fig.1.

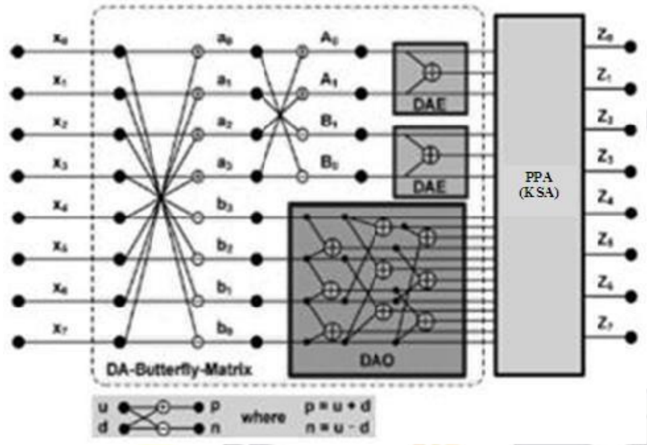

Fig.1. Architecture of 1D 8-point DA-DCT

The 1D-DCT employs the DA-based architecture [6] and proposed Kogge_Stone_Adder to achieve a high-speed, small area and low power design. The 1D 8-point DCT can be expressed as follows in Eq.(1).

$$z_n = \frac{1}{2} k_n \sum_{m=0}^{7} x_m \times \cos\left(\frac{(2m+1)np}{16}\right) \qquad (1)$$

where, x_m denotes the input data;

z_n denotes the transform output;

$k_n = \frac{1}{\sqrt{2}}$ for $n = 0$;

$0 \le n \le 7$; $k_n = 1$ for other n values.

Disregarding the scaling factor of ½, the 1D 8-point DCT in Eq. (1) can be bifurcated into distinct even and odd components, as outlined in [7]. Operating within the DA-based DCT framework involves the segmentation of input pixel data into even and odd parts, all while representing the cosine basis in a Canonical Sign Digit (CSD) format [7]. The mechanics of image compression come into play as the input image is partitioned into 8 × 8 blocks and subjected to multiplication by the DCT matrix. Following the multiplication, an addition process ensues.

Consider a scenario where the input pixel value, say 120, is multiplied by the DCT fraction value of 0.707, yielding 84.84. In this calculation, the complexity is heightened, and delays increase due to the carry component. To navigate this intricacy and delay, this paper introduces the utilization of Canonical Sign Digit within the DA-based DCT structure, as elaborated in [6]. This strategy involves disregarding unnecessary LSBs, achieved by representing input pixels with larger numbers, such as 210 as illustrated below.$2^{10} \times 0.707 = 723.968$

By multiplying pixel value with larger number we are getting,

$$724 \times 120 = 86880$$

For 86880 the binary values are 10101001101100000, and by neglecting the 10 LSB original values are held as indicated below.

Accurate	Inaccurate part
1010100	1101100000
MSB	LSB (Neglected)

84

By this proposed method the decimal values are completely carried away and complexity reduces largely by left shifting as described the cosine basis in [6].

3. PARALLEL PREFIX ADDER

A comparative analysis of various types of adders is presented in [7]. The complexity challenge can be effectively mitigated through the utilization of the Parallel Prefix Adder (PPA). This adder stands out for its rapid computation of carries, organized in a tree structure for each bit [9]. Diverse variants of PPA adders are accessible, with Brent-Kung and Kogge-Stone emerging as particularly notable. Within this context, the Kogge-Stone Adder (KSA) takes the spotlight in the DA-based DCT structure, selected for its remarkable speed.

3.1 KOGEE_STONE_ADDER

The operation of an adder entails combining two supplementary signals, frequently integrated into other arithmetic form carry look-ahead adder. It generates carry signals in log $_2n$ In the KSA, carriers are swiftly computed through parallel computation, albeit at the expense of increased area (n*log2n - n+1). Within this adder, the generate and propagate stages play a pivotal role. These signals are determined by logic equations for each bit i of the adder, with generate (Gi) being represented as

$$G_i = a_i \cap b_i \qquad (2)$$

where, G_i indicates whether a carry is generated from that bit or not, and also for each bit i of adder propagate P_i as presented in Eq.(3).

$$P_i = a_i \oplus b_i \qquad (3)$$

Here, Pi signifies whether a carry is Propagated from that bit or not. The G and P blocks constituting the state are visually represented in Fig.2. The functioning of a 4-bit Kogge-Stone Adder is outlined in Fig.3, where the carry look-ahead state serves as input, as elaborated in reference [4]. **(G_{left}, P_{left})**

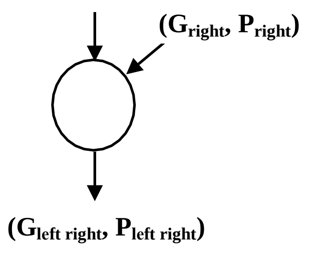
(G_{right}, P_{right})
($G_{left\ right}$, $P_{left\ right}$)

Fig.2. KSA carry operator

Carry look ahead network differentiates KSA from other adders and is the main force behind its high performance. This step involves computation of carries corresponding to each bit. It uses group propagate and generate as intermediate signals which are given by the logic Eqs.(4) and (5) as mentioned below:

$$P_i : j = P_i : k{+}1 \text{ and } P_k : j \qquad (4)$$

$$G_i : j = G_i : k{+}1 \text{ or } (P_i : k{+}1 \text{ and } G_k : j) \qquad (5)$$

Post processing is the final step it involves computation of sum bits. Sum bits are computed by the logic given below in Eq.(6).

$$S_i = P_i \oplus C_{i-1} \qquad (6)$$

To strike a balance between computational complexity and image quality, this approach strives for minimal degradation in image quality while simultaneously reducing the overall count of adders needed for the addition function. This technique involves computing adders at different levels within the DCT process, thereby optimizing the trade-off. When it comes to image compression, post the execution of 2D DCT, a quantization table is applied to eliminate AC coefficients. Consequently, the complexity remains consistent across various compression rates. This is where the parameterizable level comes into play, enabling image compression with adjustable quantization tables. Through

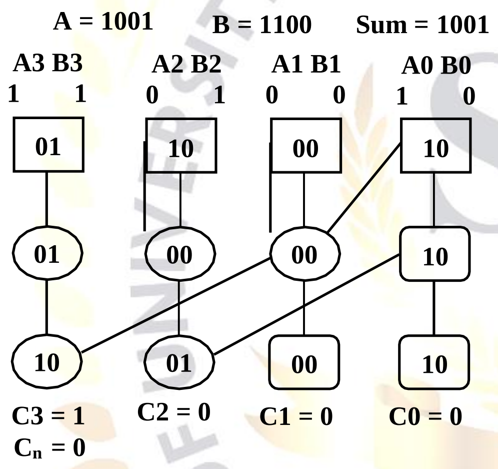

Fig.3. 4 Bit Kogge_Stone_Adder

Table.1. Comparison outputs of adders

ADDER TYPE	HARDWARE COMPLEXITY	SPEED (MHz)
Ripple Carry Adder	98	164.58 MHz
Carry Save Adder	97	205.21 MHz
Carry Look Ahead Adder	99	256.61 MHz
Kogge_Stone_Adder	83	317.97 MHz

4. EXPERIMENTAL RESULTS

The process commences by converting the image into pixel values using MATLAB, subsequently storing these values within a text file. The MODELSIM ALTERA platform accesses this text file, initiating the calculation of corresponding 2D DCT coefficients. Once computed, these coefficients are inputted into

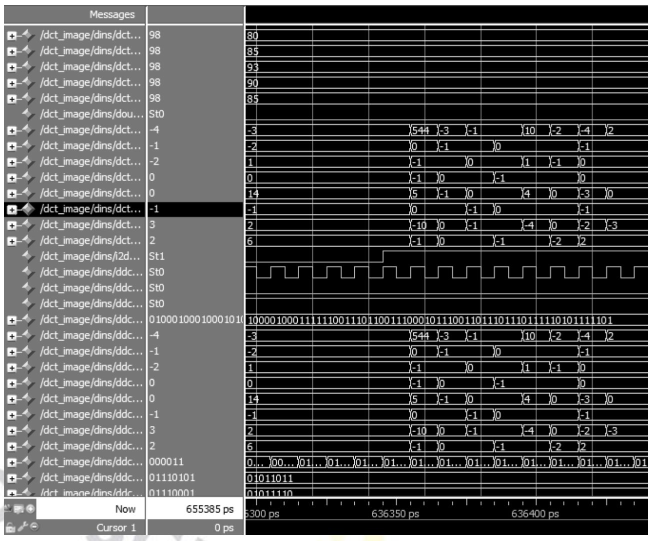

Fig.4. 2D DCT coefficient

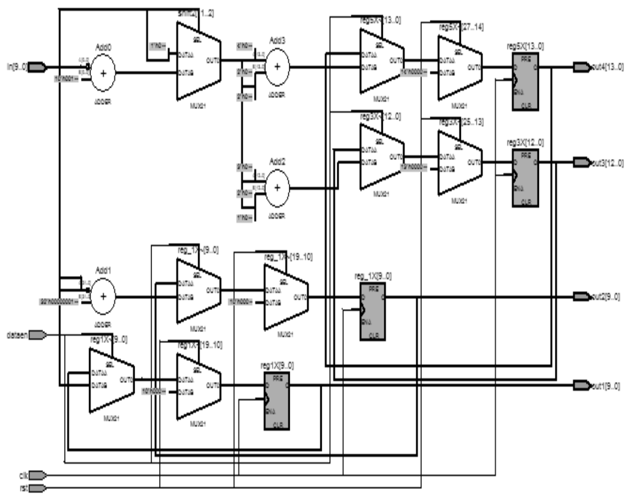

Fig.5. RTL for 2D DCT

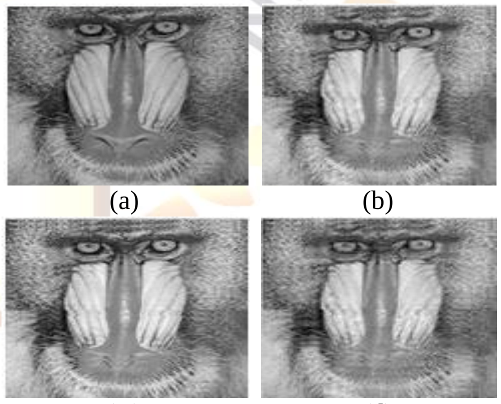

Fig.6(a). Original Image, (b). Image reconstruction by level 0, (c). Image reconstruction by level 1, (d). Image reconstruction by level 2

These data are written to a text file, then image reconstructed from the text file using MATLAB coding. Finally hardware and

QUARTUS II EDA tool is harnessed to gauge speed optimizations. The outcomes of simulations are visually depicted in Fig.4 and Fig.5.

The efficacy of the proposed approach featuring DA-based DCT is assessed through performance analysis across various levels, as illustrated in Fig.6. The levels in question are customizable, aligned with user specifications. When levels are augmented, complexity diminishes at the expense of image quality. Conversely, at level 0, the complexity increases while image quality significantly improves. This evaluation is effectively quantified using Peak Signal to Noise Ratio (PSNR), as presented in Table.2.

Table.2. PSNR and F_{max} values for the reconstructed image

LEVELS	**PSNR VALUES**	**FMAX**
Level 0	5.3250	322.68 MHz
Level 1	5.3246	314.76 MHz
Level 2	5.3232	429.55 MHz

Then the comparison of area and power consumption is shown below in Table.3 and Table.4. Various frequency levels are taken place at Table.2.

Table.3. Area analysis for the reconstructed images

Parameter	**Total**	**Level 0**	**Level 1**	**Level 2**
Total logic elements	15408	543	742	175
Logic register	15408	226	243	52
Registers	15408	243	226	52
Total pins	347	171	171	171
Total memory bits	516096	0	0	0
PLL	4	0	0	0

Table.4. Power analysis for the reconstructed images

Powers	**Level 0**	**Level 1**	**Level 2**
Thermal Power Dissipation	329.79mW	329.79mW	80.06mW
Dynamic Power Dissipation	0.00mW	0.00mw	0.00mW
Static Power Dissipation	303.03mW	303.03mW	51.80mW
I/O Thermal Power	26.76mW	26.76mW	28.26mW

5. CONCLUSION

The idea of employing DA-based DCT and IDCT architectures leverages algorithmic strength reduction techniques to curtail device utilization, resulting in lowered power consumption. These architectures have been devised and incorporated within VLSI designs. Notably, DCT calculations are executed using the DA-based approach with commendable precision, yielding satisfactory quality. This is achieved through the implementation of the Kogge_Stone_adder, which eliminates the need for carry propagation. The suggested DA-based DCT architecture excels when compared to a multiplier-based approach, attaining heightened efficiency. Consequently, this

REFERENCES

[1] Yao Wang, J. Ostermann and Ya-Qin Zhang, "*Video Processing and Communications*", First Edition, Prentice-Hall, 2002.

[2] Gilbert Strang, "The Discrete Cosine Transform", *Society for Industrial and Applied Mathematics Review*, Vol. 41, No. 1, pp. 135-147, 1999.

[3] G. K. Wallace, "The JPEG still picture compression standard", *IEEE Transactions on Consumer Electronics*, Vol. 38, No. 1, pp. xviii–xxxiv, 1992.

[4] Dhanya Geethanjali Sasidharan and Aarathy Iyer, "Comparison of Multipliers Based on Modified Booth Algorithm", *International Journal of Engineering Research and Applications*, Vol. 3, No. 1, pp. 1513-1516, 2013.

[5] V. Muralitharan and M. Jagadeeswari, "An Enhanced Carry Elimination Adder for Low Power VLSI Implementation", *International Journal of Engineering Research and Applications*, Vol. 2, No. 2, pp. 1477-1482, 2012.

[6] N. R. Divya and K. Kannadasan, "Logic Complexity Reduction and VLSI Architecture for Image Compression Using Conventional Adders", *International Journal of Electrical and Communication Engineering for Applied Research*, Vol. 2, pp. 31-35, 2014.

[7] N. R. Divya and K. Kannadasan, "Image compression using DA-DCT Logic Through VLSI structure with Power Enhancement Scheme", *International Journal of Graphics and Image Processing*, Vol. 4, No. 1, pp. 32-37, 2014.

[8] A. M. Shams, A. Chidanandan, W. Pan and M. A. Bayoumi, "NEDA: A low-power high-performance DCT architecture", *IEEE Transactions on Signal Processing*, Vol. 54, No. 3, pp. 955-964, 2006.

[9] Young-Ho Seo and Dong-Wook Kim, "A New VLSI Architecture of Parallel Multiplier Accumulator Based on Radix-2 Modified Booth Algorithm", *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 18, No. 2, pp. 201-208, 2010.