A Novel Superscalar Architecture for Fast DCT Implementation

Zhang Yong[1] and Min Zhang[2]

[1]DSP Lab, S2, School of Electronic and Electrical Engineering,
Nanyang Technological University, Singapore, 639798
eyzhang@ntu.edu.sg

[2]Reseach Institute of Planning & Design
Huaihe River Water Resources Commission of MWR,
41, Fengyang West Rd., Bengbu Anhui, P. R. China, 233001
zmxw@bb.ah.cninfo.net

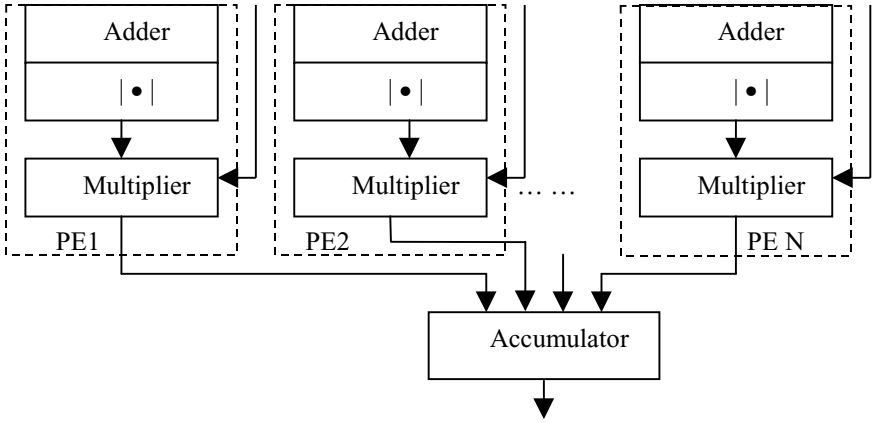
Abstract. This paper presents a new superscalar architecture for fast discrete cosine transform (DCT). Comparing with the general SIMD architecture, it speeds up the DCT computation by a factor of two at the cost of small additional hardware overheads.

1. Introduction

The Discrete Cosine Transform (DCT) is currently the most popular and effective transform coding scheme to reduce the redundancy of the signals in visual communication and multimedia applications. The broader use of DCT in the wider application areas underlines the requirement for a more efficient and systematic approach to DCT implementation.

The SIMD (Single Instruction Multiple Data) processor has been widely employed for DCT implementation in video encoders because of two reasons. Firstly, more flexibility can be achieved by processing the task under software control. Moreover, since all of the low level tasks (e.g., DCT, motion estimation and quantization) in the whole video compression system have similar computing characteristics, a common processor architecture can be shared for these computing requirement so as to simplify the design problem and save the hardware cost. Fig. 1 illustrates a typical SIMD architecture used in both video signal processors (VSP) [1] and generalized multimedia processors [2]. It properly exploits temporal parallelism and spatial parallelism to process DCT as well as other low level computation-intensive tasks.

However, comparing with the functional specific DCT implementations, the SIMD model cannot achieve best performance for Fast DCT (FDCT) algorithms, consequently lowering the whole system's throughput. In this paper, we will present a superscalar modification of the general SIMD structure shown in Fig. 1. It completes the DCT computation twice as fast as the original SIMD does, at the cost of small hardware overheads.



| • | -- Absolute operator

Fig. 1. A typical SIMD architecture

2. Modified SIMD Architecture for Fast DCT

A typical 8-point DCT transform is computed in the following form:

$$X_i = c_i \sum_{k=0}^{7} x_k \cos(\frac{(2k+1)i\pi}{16}),\ 0 \le i \le 7 \qquad (1)$$

where x_i are the original data, X_i are the transformed values and c_i are weight parameters defined as

$$c_i = \begin{cases} \sqrt{2} & i = 0 \\ 1 & otherwise \end{cases} \qquad (2)$$

In order to accelerate the DCT computation, many fast DCT algorithms have been developed. One of the most efficient methods is performing DCT with the aid of the fast Fourier transformation (FFT) [3]. Fig. 2 illustrates a computing flow chart for such FDCT algorithm, in which d_i denote the cosine coefficients and y_i, z_i are the intermediate results during transform.

It can be seen that there are many features in the FDCT algorithm that impede the SIMD architecture working effectively:

- There are many data transmissions among PEs, therefore a complex interconnection network is required for the FDCT implementation.
- The operation components in different datapaths (x_i to X_i) are various, this irregularity results in that the computation units in PEs cannot be fully utilized.
- Eight PEs are required in order to perform eight-point DCT concurrently. Since the datawidth of one datapath in DCT is normally 16 bits, the total bus width reaches

128bits, which is difficult for physical implementation in terms of the current semiconductor technology.

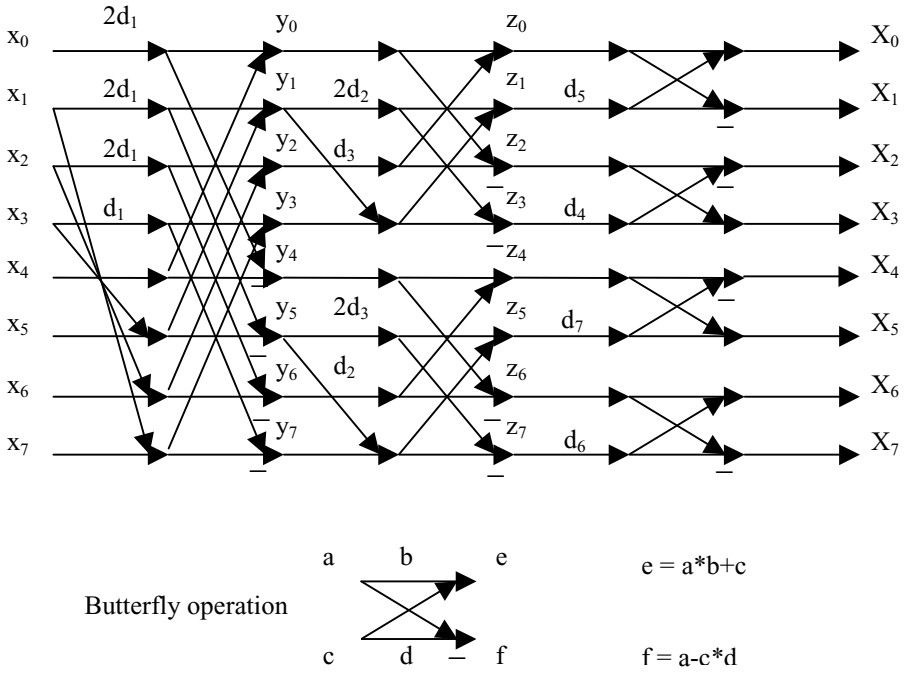


Fig. 2. Fast DCT flow chart

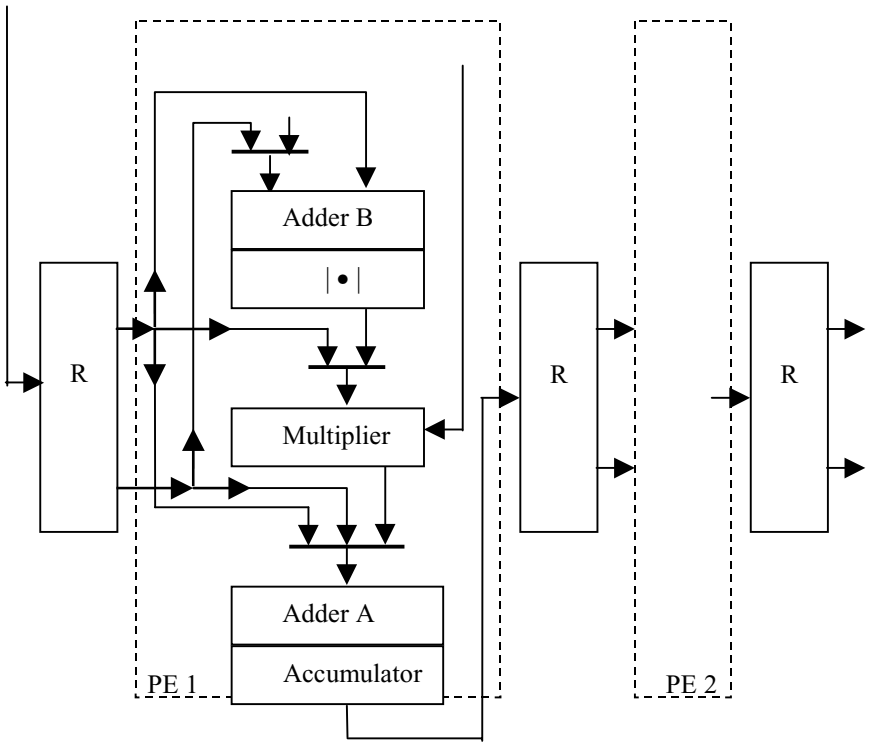
For these reasons, the FDCT algorithm is seldom employed in previous SIMD processors despite of its merit on computational efficiency. In order to solve this problem, some modifications have to be made for the SIMD architecture shown in Fig. 1.

Firstly, the PEs are connected each other to form a PE chain by fetching the result of one PE to its neighboring PE. With this manner, the whole computation flow is mapped onto the SIMD architecture. It can be deduced from Fig. 2 that the 3-PE Chain can fulfill the butterfly computations in 8-point FDCT.

Secondly, a register file ***R*** is added to each PE for storing the intermediate computation results. The complex interconnection of the PEs is consequently avoided by addressing the register properly.

In addition, an additional adder is introduced to the PE for assisting the butterfly computations.

The new architecture is shown in Fig. 3.



| • | -- Absolute operator

Fig. 3. Modified SIMD architecture

3. Superscalar Execution of FDCT

In a typical superscalar processor [5], the incoming instruction stream is fetched and decoded several instructions at a time, the instructions are then initiated for execution in parallel based primarily on the availability of operand data, rather than their original program sequence. Borrowing this important feature of the superscalar processor, we schedule the operations in the FDCT algorithm onto the architecture of Fig. 3 in terms of whether the operand data is ready. The timing chart of one 8-point FDCT execution is illustrated in Fig. 4, where:

the labels on the ***t*** axis are the cycle numbers of the execution;

'R' denotes reading the operand from register files;

'W' denotes writing the result to register files;

'*' denotes the multiplication;

'A+', 'A-', 'B+', 'B-' denote the addition and the subtraction of adder A and B respectively.

4. Comparison and Conclusion

As for the SIMD architecture shown in Fig. 1, when N=4, in order to compute one element in 8-point DCT, two cycles are required (see equation (1)). By contrast, as shown in Fig. 4, the new architecture completes each in one cycle at the pipeline output when the pipeline is full. Notice that in most DCT applications, the data to be transformed are input continuously, thus the preload overhead of the pipeline can be ignored. Therefore, the throughput of the modified superscalar architecture is twice faster than that of the original one.

The modified superscalar architecture and the original SIMD as well as their control circuits have been implemented by the logic synthesis procedure, which are followed by an estimation for the related gate numbers [4]. The results show that the hardware resources of the former are only 14% more than that of the latter.

In conclusion, the proposed architecture gains speedup of two over the previous SIMD one at the cost of small hardware overheads.

REFERENCES

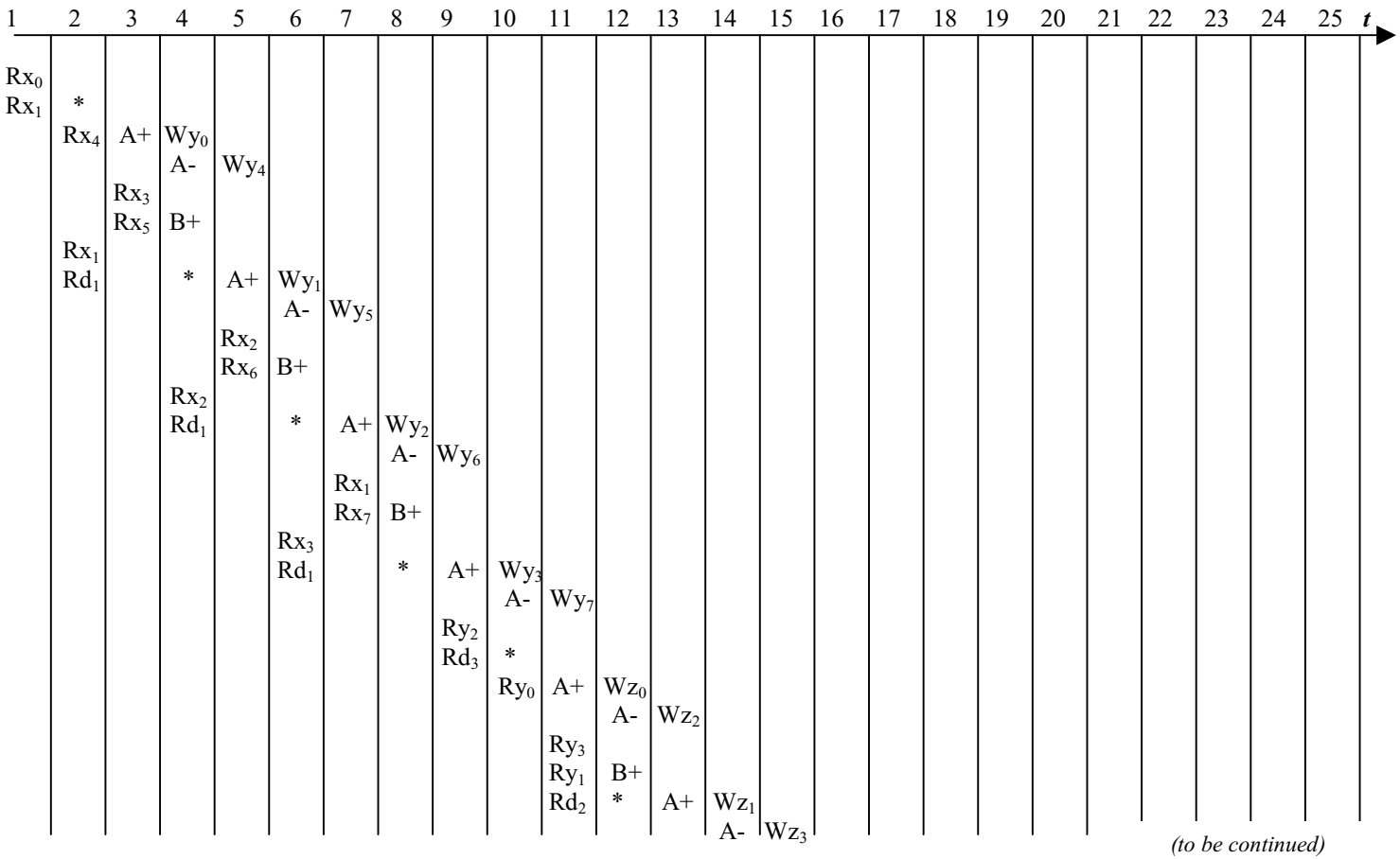
[1] J. Hilgenstock, et al., "A video signal processor for MIMD multiprocessing", Proceedings of Design Automation Conference, 1998, pp. 50 –55.

[2] R. B. Lee, "multimedia extensions for general-purpose processors", IEEE Workshop on Signal Processing Systems, 1997, pp. 9 –23

[3] A. N. Netravali and B. G. Haskell, “Digital Pictures, -Representation, Compression, and Standards”, Plenum Press, 1995.

[4] Zhang Yong, “Research on Video Encoder Design”, Ph. D. Dissertation, Zhejiang University, 1999.

[5] D. A. Patterson, J. L. Hennessy, “Computer Architecture a Quantitative Approach”, Morgan Kaufmann Publishers Inc., 1996.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 t
Rx0
Rx1
*
Rx4
A+
Wy0
A-
Wy4
Rx3
Rx5
B+
Rx1
Rd1
*
A+
Wy1
A-
Wy5
Rx2
Rx6
B+
Rx2
Rd1
*
A+
Wy2
A-
Wy6
Rx1
Rx7
B+
Rx3
Rd1
*
A+
Wy3
A-
Wy7
Ry2
Rd3
*
Ry0
A+
Wz0
A-
Wz2
Ry3
Ry1
Rd2
B+
*
A+
Wz1
A-
Wz3
(to be continued)

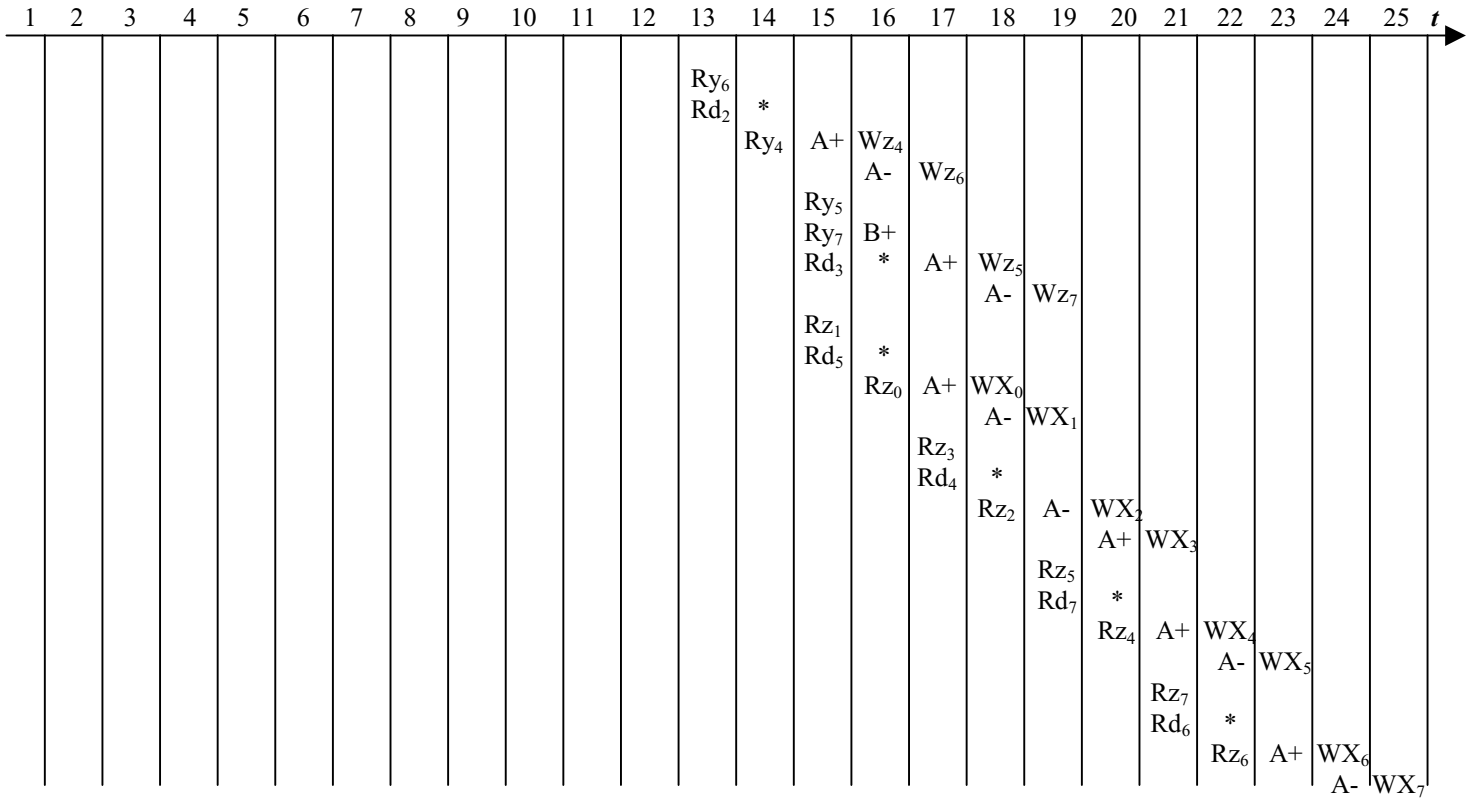


Fig. 4. Timing chart of the superscalar execution for FDCT