Systematic Approach of Fixed Point 8x8 IDCT and DCT Design and Implementation[1]

Ci-Xun Zhang, Jing Wang, Lu Yu

Institute of Information and Communication Engineering,
Zhejiang University,
Hangzhou, China, 310027

Abstract. MPEG has recently issued a CFP for voluntary fixed point 8x8 IDCT and DCT standards to ease the effort that is needed to implement the IDCT and DCT, and also to help ensure that decoders are implemented in conformance with the MPEG standard. This paper is conclusion and extension of our previous proposal responding to the CFP. A systematic approach of fixed point 8x8 IDCT and DCT design and implementation is proposed that approximate the ideal integer output IDCT and DCT with high fidelity. Performance and complexity issues such as bit width are discussed for different methods using this approach. The methods discussed in this paper can also be easily extended to IDCT and DCT with other size.

Index Terms—DCT, IDCT, MPEG, standard

1. INTRODUCTION

Recently, MPEG has issued a CFP for voluntary fixed point 8x8 IDCT and DCT standards to ease the effort that is needed to implement the IDCT and DCT, and also to help ensure that decoders are implemented in conformance with the MPEG standard [1]. The proposed IDCT should satisfy all the accuracy requirements specified in [2] [3]. This paper is conclusion and extension of our previous proposal responding to this CFP [4]. In this paper, we propose a systematic approach of fixed point 8x8 IDCT and DCT design and implementation that approximate the ideal integer output IDCT and DCT with high fidelity. The paper is organized as follows. In section 2, a detailed description about the design and implementation of the proposed algorithm is presented. Performance and complexity issues such as bit width are also discussed for different methods using this approach. Section 3 concludes the paper.

2. ALGORITHM DESIGN AND IMPLEMENTATION

2.1 Algorithm Design

The basic idea is to apply an exact mathematical equivalent to the following separable process: (For simplicity, we focus our discussion on IDCT and similar design approach can be applied for DCT.)

1. Matrix multiply by the fixed point IDCT matrix $\mathbf{IDCT_{fp}}$ which is produced by multiplying the ideal IDCT matrix by sqrt(8)*2^{SCALE} and rounding the resulting values to the nearest integer as given by (1).

$$\mathbf{IDCT}_{fp} = round(\sqrt{8} \cdot 2^{SCALE} \cdot \mathbf{IDCT}) = \begin{bmatrix} G & A & E & B & G & C & F & D \\ G & B & F & -D & -G & -A & -E & -C \\ G & C & -F & -A & -G & D & E & B \\ G & D & -E & -C & G & B & -F & -A \\ G & -D & -E & C & G & -B & -F & A \\ G & -C & -F & A & -G & -D & E & -B \\ G & -B & F & D & -G & A & -E & C \\ G & -A & E & -B & G & -C & F & -D \end{bmatrix} \quad (1).$$

2. Right shift by ROW_SHIFT bits (with rounding) after 1-D transform and COL_SHIFT bits after 2-D transform (also with rounding). The process can be either row-first ordering or column-first ordering. Without losing generality, we use row-first ordering here, and the process can be represented by (2) and (3) below:

$$\mathbf{IDCT}_{output\_1d} = (\mathbf{IDCT}_{input} \cdot \mathbf{IDCT}_{fp} + (1 << (ROW\_SHIFT - 1))) >> ROW\_SHIFT \quad (2)$$

$$\mathbf{IDCT}_{output\_2d} = (\mathbf{IDCT}_{fp} \cdot \mathbf{IDCT}_{output\_1d} + (1 << (COL\_SHIFT - 1))) >> COL\_SHIFT \quad (3)$$

3. Clip the 2-D transform output to the pre-defined IDCT output range. For SAMPLE_BITS-bit video sample data, this can be represented by (4):

$$\mathbf{IDCT}_{output} = Clip3(-2^{SAMPLE\_BITS}, 2^{SAMPLE\_BITS} - 1, \mathbf{IDCT}_{output\_2d}) \quad (4)$$

where:

$$Clip3(x, y, z) = \begin{cases} x; & z < x \\ y; & z > y \\ z; & otherwise \end{cases} \quad (5)$$

Let INPUT_BITS denote the bit width of the IDCT input data and OUTPUT_BITS denote the bit width of the IDCT output data. INPUT_BITS and OUTPUT_BITS are given by (6) and (7) respectively [1]:

[1] This research is sponsored by NSFC under contract No. 60333020 and 90207005.

$$INPUT\_BITS = SAMPLE\_BITS + 4, \quad (6)$$

$$OUTPUT\_BITS = SAMPLE\_BITS + 1. \quad (7)$$

The relationship of INPUT_BITS and OUTPUT_BITS is given by (8):

$$\begin{aligned} &INPUT\_BITS + 2 \bullet SCALE - ROW\_SHIFT - COL\_SHIFT \\ &= OUTPUT\_BITS \end{aligned}. \quad (8)$$

Combining (6), (7), (8) we can get:

$$\begin{aligned} &2 \bullet SCALE \\ &= ROW\_SHIFT + COL\_SHIFT - 3 \end{aligned}, \quad (9)$$

and thus:

$$SCALE = n \ \ or \ \ SCALE = n + 1/2 \quad n \in \mathbb{Z}. \quad (10)$$

In this paper, we only consider the methods with positive SCALEs. Similar methods with practically negative SCALEs (but not rounded to nearest integers) are discussed in [7].

Note here that conceptually different SCALEs can be used for column and row transform. However, the above derivation is based on using same SCALE for both column and row transform. Such an arrangement is of significant importance for DCT/IDCT chip designs, since only one single butterfly structure implementation is needed.

Bit width is a major issue in the design of the IDCT especially in hardware implementation, and serves as an important complexity consideration when choosing a specific IDCT design. Assuming the IDCT input is the theoretical DCT output (without quantization/dequantization or possible error, etc), a good estimation of the bit-widths is as follows:

The bit width of theoretical 1-D IDCT output value is:

$$\begin{aligned} &OUTPUT\_BITS_{1D} \\ &= \lceil SCALE - ROW\_SHIFT + SAMPLE\_BITS + 5 \rceil \end{aligned}. \quad (11)$$

The bit width of theoretical 2-D IDCT output value is:

$$OUTPUT\_BITS_{2D} = SAMPLE\_BITS + 3. \quad (12)$$

The bit width of theoretical maximum bit width of intermediate value during the transform process is:

$$\begin{aligned} &MAX\_INTER\_BITS \\ &= COL\_SHIFT + SAMPLE\_BITS + 3 \end{aligned}. \quad (13)$$

From (11), (12), (13) we can see that the bit widths are mainly determined by the parameters SCALE, ROW_SHIFT, and COL_SHIFT. (there are actually only two free variables among three.) The method described above is denoted as (SCALE, ROW_SHIFT, COL_SHIFT) in this paper.

2.2 Implementation Schemes

Among all possible methods that meet all the accuracy requirements, the method (13,11,18) is proposed because it is accurate, fast and cost effective. It is similar to the method adopted by the Independent JPEG Group in its popular JPEG implementation but better and more elegant [4] and has lower implementation complexity than that in [5]. Not like many multiplier-less methods [13]-[19], it can be implemented in many different ways that are mathematically equal in output value. In the following, four different implementation schemes are studied and compared.

Scheme 1: The proposed method can be implemented using the butterfly structure in [6] with 12 multiplications and 32 additions. However, from (10), we can see that SCALE can be non-integer, so we use 14 multiplications in Figure 1 to also take these cases into consideration. The two multipliers in the upper two paths of the butterfly structure can be replaced by two shifters when SCALE is an integer. The advantage of this structure is that all the multiplications in each of the two separable stages of the transform can be implemented in parallel with each other. Note that the apparent rounding offset used before the right shift in every 1-D transform can be implemented by just adding a constant to the DC term near the beginning of the process.

For a fixed point IDCT matrix derived by (1), the parameters in the butterfly structure in Figure 1 can be obtained by (14) and assured to be solvable and unique. This is mainly due to the fact that there is at most one multiplication in every path inside the butterfly structure. Theoretical analysis shows that for 8-bit video sample data, only 16bit signed by signed multiplications is needed to implement the method (13,11,18).

$$\begin{aligned} e_0 &= -E - F \\ e_1 &= F \\ e_2 &= E - F \\ d_0 &= -A + B + C - D \\ d_1 &= A + B - C + D \\ d_2 &= A + B + C - D \\ d_3 &= A + B - C - D \\ d_4 &= -B + D \\ d_5 &= -A - B \\ d_6 &= -B - C \\ d_7 &= -B + C \\ d_8 &= B \end{aligned} \quad (14)$$

Scheme 2: The butterfly structure of scheme 1 is also used here. However, in scheme 2, we replace the multipliers in the butterfly structure with VLSI-friendly coefficients using shifters and adders. There are many methods of decomposing constant integer multipliers into representation of shifters and adders. One common method is by using Canonic Signed Digit (CSD) representation which requires 33% fewer nonzero digits than binary [8] [9]. Optimal methods based on the graph representation of the multipliers are described in [10] [11] but parallel implementation may be harmed.

The total number of adders of different methods with SCALE from 10.5 (We note that SCALE as small as 10.5 may suffice to achieve all precision requirements. It is shown in [5] that the smallest SCALE is 11 because non-integer SCALE is not considered there.) up to 16 are also calculated and given in Table 1. Some examples can be seen in [20]. It is expected that when SCALE is not an integer, comparatively more shifters and adders are needed. The methods with SCALE equal to 13 seem to be most cost effective ones (Detailed accuracy and bit widths of methods with different SCALE values can be seen in [20].) and even need fewer adders than methods with SCALE equal to 12 in the optimal sense.

Table 1. Number of adders with SCALE values from 10.5 up to 16

	CSD	Optimal
SCALE=10.5	82	73
SCALE=11	75	70
SCALE=11.5	88	78
SCALE=12	78	73
SCALE=12.5	95	83
SCALE=13	81	72
SCALE=13.5	98	84
SCALE=14	87	77
SCALE=14.5	100	85
SCALE=15	94	81
SCALE=15.5	109	90
SCALE=16	99	82

Scheme 3: On PCs with MMX/SSE/SSE2/SSE3 or other platforms with SIMD instruction, implementations using matrix multiplication can be even faster than butterfly-structure implementations though there are more operations. In scheme 3, we use the "chain matrix multiplication" scheme presented in [12], where it is originally used to do efficient 4x4 matrix multiplication. Here we adapt it to 8x8 matrix multiplication as follows. The row transform is first calculated (Figure 2), and then the column transform (Figure 3). One major problem when using SIMD instruction is that it is costly to load column vector of a matrix into an SIMD register. This scheme effectively avoids this problem by pre-arranging the input data and introducing a structure which assures that the intermediate results stored in the resulting register are just in the right order for the next step of matrix multiplication, thus no shifting or abundant loading operations are needed.

Scheme 4: In scheme 4, we use the well known hybrid structure based on (16) and (17). By using this scheme, we expect to have both advantages of matrix multiplication and butterfly structure.

$$\begin{bmatrix} X0 \\ X1 \\ X2 \\ X3 \end{bmatrix} = \begin{bmatrix} G & E & G & F \\ G & F & -G & -E \\ G & -F & -G & E \\ G & -E & G & -F \end{bmatrix} \begin{bmatrix} Y0 \\ Y2 \\ Y4 \\ Y6 \end{bmatrix} + \begin{bmatrix} A & B & C & D \\ B & -D & -A & -C \\ C & -A & D & B \\ D & -C & B & -A \end{bmatrix} \begin{bmatrix} Y1 \\ Y3 \\ Y5 \\ Y7 \end{bmatrix}, \quad (16)$$

$$\begin{bmatrix} X7 \\ X6 \\ X5 \\ X4 \end{bmatrix} = \begin{bmatrix} G & E & G & F \\ G & F & -G & -E \\ G & -F & -G & E \\ G & -E & G & -F \end{bmatrix} \begin{bmatrix} Y0 \\ Y2 \\ Y4 \\ Y6 \end{bmatrix} - \begin{bmatrix} A & B & C & D \\ B & -D & -A & -C \\ C & -A & D & B \\ D & -C & B & -A \end{bmatrix} \begin{bmatrix} Y1 \\ Y3 \\ Y5 \\ Y7 \end{bmatrix}. \quad (17)$$

2.3 Experimental Results and Analysis

Average run time of each implementation scheme is obtained by running 100000 different 8x8 DCT coefficient matrices on PC platform and the results are given in Table 2. Because the butterfly structure in scheme 1 and scheme 2 is not as regular as that used in scheme 3 and scheme 4, when implemented using SIMD instruction, a larger number of pack/unpack/shuffle operations are needed and the speed is rather decreased than increased. So in our tests, we only use the compiler optimization options for C code as a relative fair comparison. It should be noted that although additions and shifts can be implemented faster than multiplications, scheme 1 and scheme 2 just have similar speed performance. This is because on average 4 shifts and additions are needed to replace one multiplication in scheme 2, and it may even reduce the speed when implemented on PCs. It is also shown that scheme 3 and scheme 4 run much faster than the other two schemes and scheme 4 is the fastest among all tested. Scheme 4 needs about half the time of scheme 3 mainly because the operation count in scheme 4 is about half of that in scheme 3. In fact, it is shown in [4] that scheme 4 is also much faster than some multiplier-less IDCT implementation schemes which takes much less operations to implement. Experimental results on other platforms such as DSP, ASIC and corresponding analyses are also presented in [4].

Table 2. Experimental results on PC platform

Implementation Scheme Description	Average Run Time
Reference floating-point IDCT (matrix multiplication. implemented with C)	1023ms
Scheme 1 (implemented with C)	94ms
Scheme 2 (implemented with C)	96ms
Scheme 3 (implemented with SSE2)	40ms
Scheme 4 (implemented with SSE2)	20ms

2.4 Advantages

It is well known that there are many fast IDCT/DCT implementations incorporating the scaling factors into the dequantization step [13]-[16]. However, the proposed methods have the following advantages:

1. The fixed point IDCT/DCT matrix is derived from theoretical IDCT/DCT directly, and results in conceptually simple and straightforward design.
2. The proposed algorithms affect only the basic IDCT/DCT functional blocks and does not entangle with quantization/de-quantization processes. Thus dequantization matrix is not needed and potential loss in performance is avoided [4].
3. There is no right shift in every 1-D transform so that different specific implementations yield exactly the same output result. This gives flexibility of different implementation on different platforms which can not be offered by multiplier-less methods [13]-[19].

2.5 Further Improvements and Observations

The specific method (13,11,18) is one of the most cost effective among all the possible methods that can meet the accuracy requirements. However, if the strict condition (16bit output of 1-D IDCT, 16bit signed by signed multiplication in scheme 1, etc) is loosed, then there are many more possible methods that can give better precision than (13,11,18). The detailed results are given in [20].

The following conclusions can be drawn from the experimental results:

1. The bit widths enough for pseudo-random test in [1] coincides with corresponding theoretical values for 8-bit video sample data (see (11), (12), (13)).
2. Perfect match to ideal integer output IDCT for all "near-DC" tests will be obtained when the bit width of the 1-D IDCT output is getting larger (16 bit or more).
3. All methods with non-integer SCALE have "near-DC" test results equal to 1. We believe the main reason is the two inaccurate approximated multipliers in the upper two paths of the butterfly structure in these cases. Further, this approximation error will propagate to all output terms during the final stage in the IDCT butterfly structure shown in Figure 1.
4. The methods with SCALE equal to 13 or 14 seem to be better than others. For example, (13,9,20) and (14,10,21) can be chosen for very high fidelity applications. They are comparable to the methods in [17]-[19] considering the operation counts using additions and shifts in the optimal sense.
5. Different accuracy versus complexity trade-offs can be easily achieved with same pre-defined SCALE value by adjusting ROW_SHIFT and corresponding COL_SHIFT according to (9). No change of the multipliers or its adder-shifter-representation is needed which is required by many multiplier-less IDCT schemes [15]-[19] [13].

3. CONCLUSION

In this paper, a systematic approach of fixed point 8x8 IDCT and DCT design and implementation is proposed that approximate the ideal integer output IDCT and DCT with high fidelity. Performance and complexity issues such as bit width are discussed for different methods using this approach. The methods discussed in this paper can also be easily extended to IDCT and DCT with other size.

ACKNOWLEDGEMENTS

The authors would like to thank Oscar Gustafsson for his help.

REFERENCES

1. G. Sullivan, A. Luthra, "CFP on fixed point 8x8 IDCT and DCT standard," MPEG output document N7335, July. 2005.
2. ISO/IEC 11172-6, "Specification of accuracy requirements for implementation of integer-output 8x8 inverse DCT," FCD, March. 2005.
3. ISO/IEC, "Study of ISO/IEC 11172-6 FCD," MPEG output document N7546, Oct. 2005.
4. C.-X. Zhang, J. Wang, X.-H. Chen, Q. Hu, X. Li, L. Yu, "Fixed-Point 8x8 IDCT, further result," MPEG input contribution M12617, Oct. 2005.
5. G. Sullivan, J. Lou, "Response to N7335 CfP on Fixed-Point 8x8 IDCT and DCT Standard," MPEG input contribution M12665, Oct. 2005.
6. C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11

multiplications," Proc. IEEE Intl. Conf on Acoust., Speech, and Signal Proc. (ICASSP), vol. 2, pp. 988-991, Feb. 1989.
7. C. -X. Zhang and L. Yu, "Low complexity and High Fidelity Fixed-Point Multiplier-less DCT/IDCT Implementation Scheme," MPEG input document M12936, Jan. 2006.
8. A. Avizienis, "Signed-digit number representation for fast parallel arithmetic," IRE Trans. Electronic Comp., vol. 10, pp. 389-400, Sept. 1961.
9. H. L. Garner, "Number systems and arithmetic," Advances in Computers, vol. 6. pp. 131-194, 1965.
10. A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," IEE Proc. Circuits Devices Syst., vol. 141, no. 6, pp. 407-413, Oct, 1994.
11. O. Gustafsson, A. G. Dempster and L. Wanhammer, "Extended results for minimum-adder constant integer multipliers," Proc. IEEE International Symposium on Circuits and Systems. (ISCAS), vol. 1, pp. 26-29, May. 2002.
12. X. Zhou, E. Q. Li, and Y.-K. Chen, "Implementation of H.264 Decoder on General-Purpose Processors with Media Instructions," in SPIE Conf. on Image and Video Communications and Processing, pp. 224-235, Jan. 2003.
13. P. Topiwala, et al, "Analysis and Summary of IDCT Proposals to MPEG 74th Meeting in Nice," MPEG output document N7565, Oct. 2005.
14. Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," Trans. IEICE, vol. E-71, no. 11, pp. 1095-1097, Nov. 1998.
15. Y. Reznik, H. Garudadri, H. Chung, P. Sagetong, N. Srinivasamurthy, "Fixed Point Multiplication-Free 8x8 DCT/IDCT Approximation," MPEG input document M12607, Oct. 2005.
16. A. T. Hinds and J. L. Mitchell, "Fixed-Point Discrete Cosine Transform Proposal," MPEG input document M12689, Oct. 2005.
17. T. D. Tran, L. Liu, P. N. Topiwala, "Fast Fixed-Point Multiplier-less DCT/IDCT Approximation Based on the Lifting Scheme," MPEG input document M12508, Oct. 2005.
18. T. D. Tran, L. Liu, P. N. Topiwala, "Fast Fixed-Point Multiplier-less DCT/IDCT Approximation Based on Loeffler Structure," MPEG input document M12509, Oct. 2005.
19. T. D. Tran, L. Liu, P. N. Topiwala, "Fast Fixed-Point Multiplier-less DCT/IDCT Approximation Based on Loeffler and ANN Structure," MPEG input document M12510, Oct. 2005.
20. C. -X. Zhang, J. Wang, L. Yu, "Extended Results for Fixed-Point 8x8 DCT/IDCT Design and Implementation", MPEG input contribution M12935, Jan. 2006.

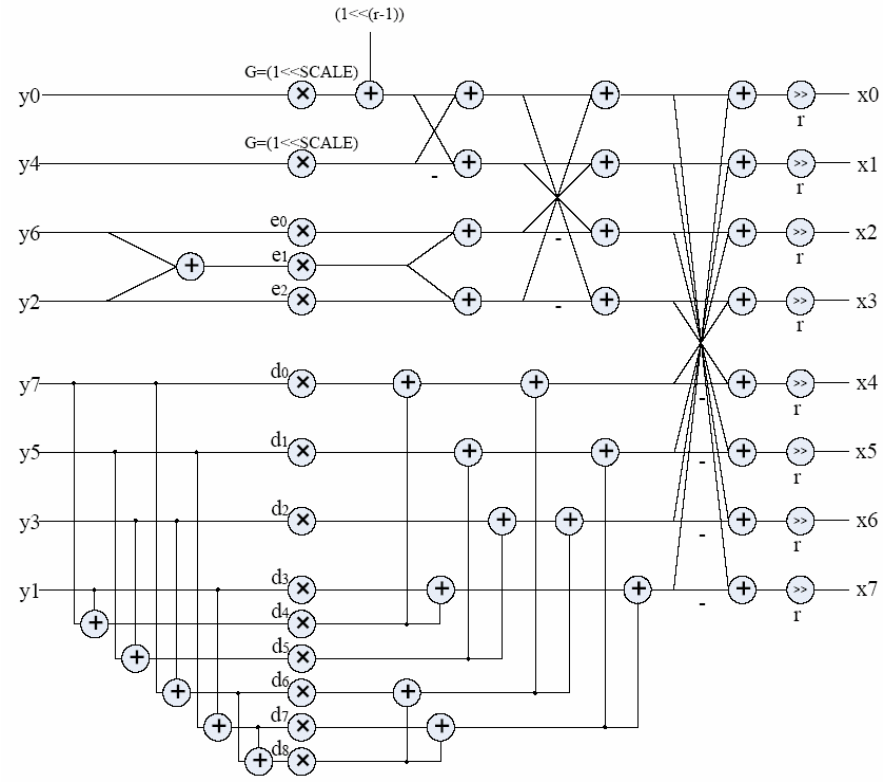

Fig. 1. IDCT Butterfly structure (including rounding)

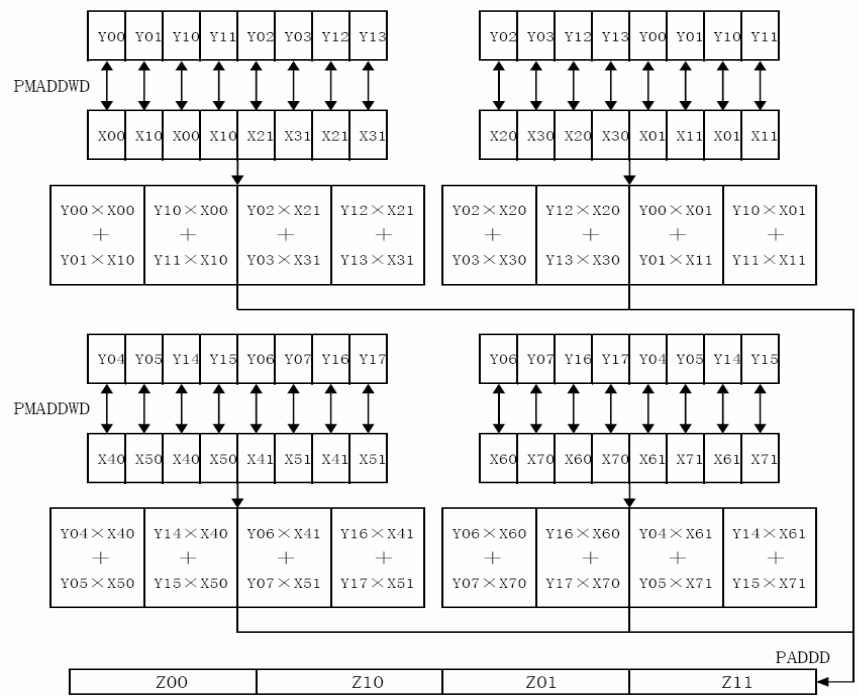

Fig. 2. Efficient 8x8 matrix multiplication – I

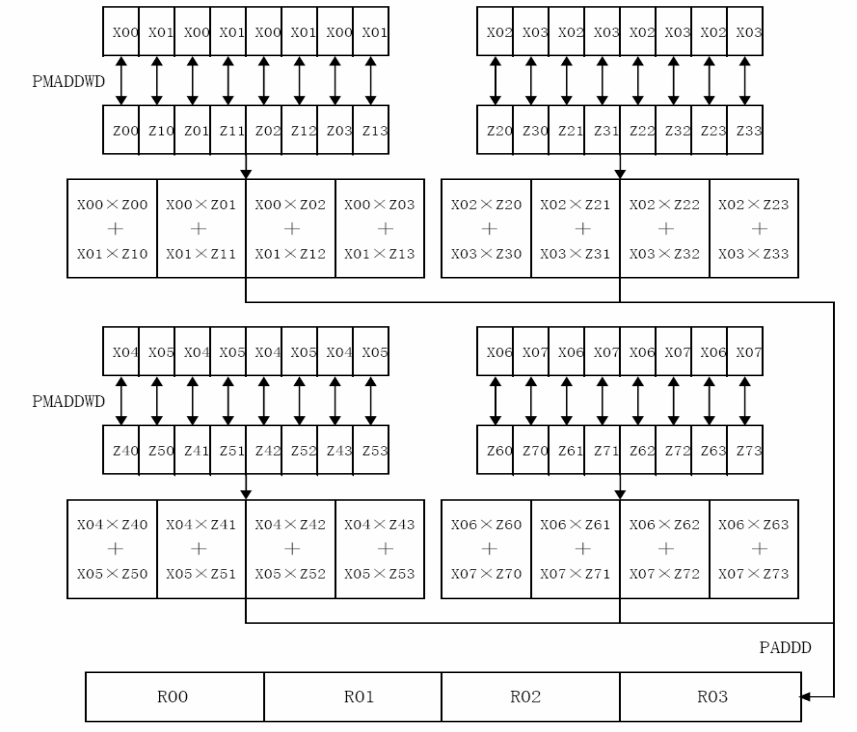

Fig. 3. Efficient 8x8 matrix multiplication – II