Fast Discrete Wavelet Transformation Daubechies-Four Architecture

Akhmad Mulyanto, Dani Fitriyanto, Tati R. Mengko, A .Z . R .Langi

Department of Electrical Engineering, Bandung Institute of Technology, Indonesia
email : mul@paume.itb.ac.id

Abstract

A DSP engine for discrete wavelet transformation (DWT) is introduced in this paper. Serial processing DWT architecture provides a better speed and efficiency, makes DWT better choice for image processing than DCT. A dedicated hardware DWT architecture attached to a general purpose microprocessor provided the programmability of data access make possible for dynamic scheduling for various DSP applications. This architecture also has expandability for new processor.

1. Introduction

Image compression based on block data contains 64 pixels for one block. Because of the bandwidth of memory, speed of fastest DCT processing still limited. For example, DCT with 8 parallel multipliers with 64 cycles for multiplication need 64 cycles for reading data from memory to complete DCT one block. By using pipelined DWT consists of 10 constant multipliers, the transformation can be done while data transferred from memory in 64 cycle read + 32 cycle latency. Even tough we use many multipliers but each multiplier has only four constant choices so that we still have efficient multiplier.

Fast DWT, as shown in Figure 1.1, can be achieved using Daubechies DWT 4 implemented in three filter banks. Three filter banks in tree structured develop three stages pipeline. First filter bank consist of two LPF and two HPF, second filter consist of two LPF and two HPF, and third filter consist of a LPF and a HPF. Every filter uses a selected constant multiplier separately make filters can run parallel.

2. Discrete Wavelet Transformation Daubechies-Four Architecture

As shown in figure 1, the DWT processor attached to a microprocessor. The microprocessor acts as master and DWT processor act as slave.

Forward DWT architecture can be seen in figure 3. It needs 3 filter banks to implement forward DWT transformation. Filter bank 1 needs 8 clocks to complete the process, filter bank 2 and filter bank 3 also need 8 clocks to complete the process.

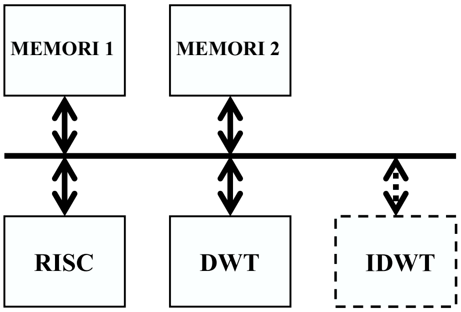

Figure 1 DSP Engine Architecture

Registers is needed to hold temporary values between filter banks. So every process can run simultaneously. Figure 2 shows three consecutive 8-byte data transformed in continues data flow.

This architecture is designed to process every 8 data. It needs particular instruction to run the system. These DWT instructions can be executed every 8 clocks, the remainder of instruction produced can be used to execute other independent resource instructions.

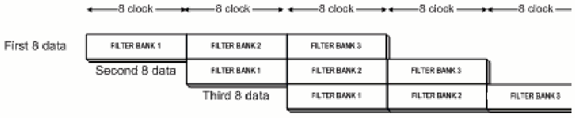

Figure 2 Pipelined Filter

Register buffer are used within each Filter Bank. Filter bank 1 uses 8 register buffer. Filter Bank 2 uses 4 register buffer. Filter Bank 2 uses 8 register buffer.

Forward DWT and Inverse DWT function are implemented in different architecture because the both functions have different precision requirement. By using different architecture, the parallelism of DSP engine will be increased.

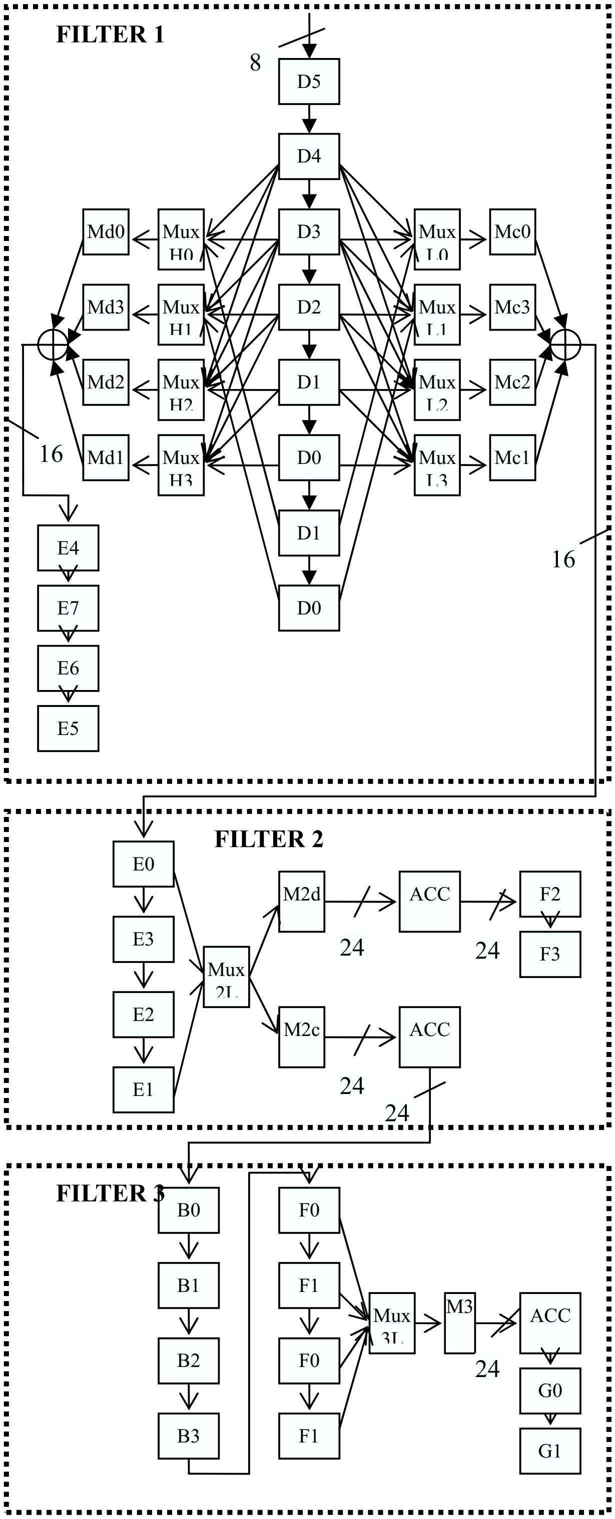

Figure 3 Forward DWT Architecture

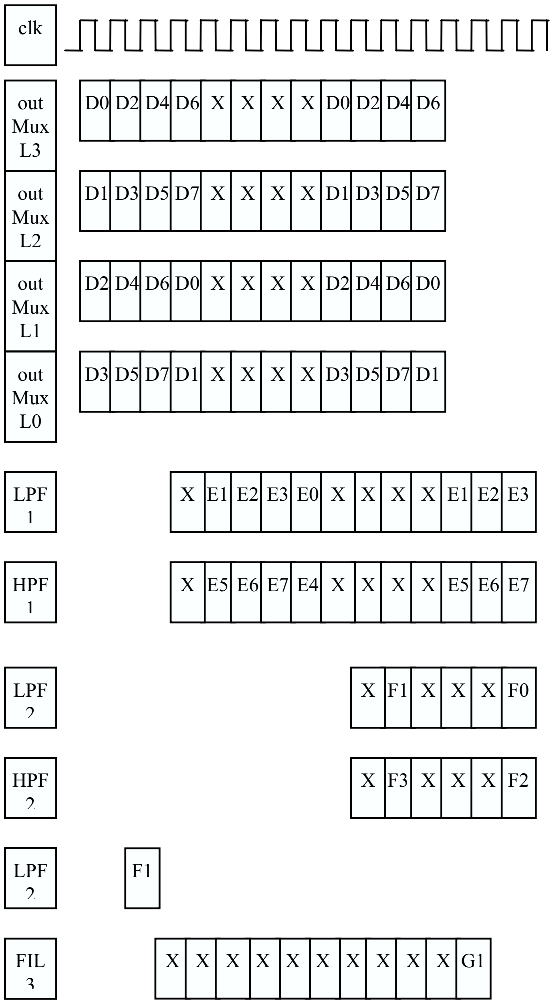
clk
out Mux L3: D0 D2 D4 D6 X X X X D0 D2 D4 D6
out Mux L2: D1 D3 D5 D7 X X X X D1 D3 D5 D7
out Mux L1: D2 D4 D6 D0 X X X X D2 D4 D6 D0
out Mux L0: D3 D5 D7 D1 X X X X D3 D5 D7 D1
LPF 1: X E1 E2 E3 E0 X X X X E1 E2 E3
HPF 1: X E5 E6 E7 E4 X X X X E5 E6 E7
LPF 2: X F1 X X X F0
HPF 2: X F3 X X X F2
LPF 2: F1
FIL 3: X X X X X X X X X X G1

Figure 4 Forward DWT Timing Diagram

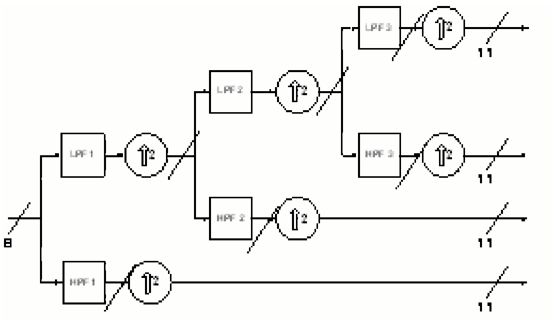

Figure 5 Forward DWT

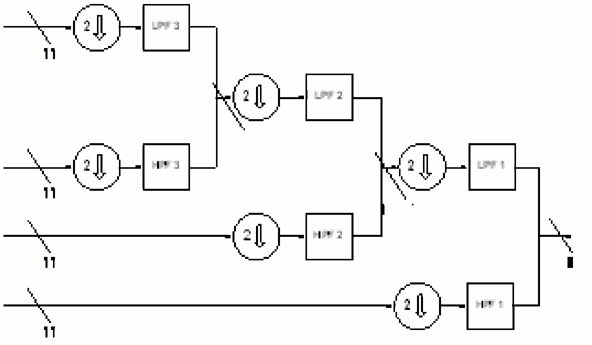

Figure 6 Inverse DWT

3. Experiment Result Using SystemC

System model has developed using SystemC. Fixed-point arithmetic is used during implementation. Figure 5 and Figure 6 shows truncated bit at output of forward DWT, and truncated bit at input/output of inverse DWT. Bit length between 2 filters is wl, iwl is integer. Table 1 and Table 2 shows fixed-point value used during experiment. Every SNR value is obtained from calculation of 10000 data using random input.

As shown in table 1, the optimum multiplier coefficient length is 13 bit (15 bit total) or 14 bit (16 bit total). Table 2 shows that the optimum multiplier-accumulator coefficient length is 21 bit (32 bit total).

Table 1. Experiment Result using different bit length of multiplier coefficient

Mult coeff		Mult Acc		SNR (dB)
wl	iwl	wl	Iwl	
6	2	17	11	32.131644
7	2	17	11	45.261646
8	2	17	11	55.573556
9	2	17	11	71.467154
10	2	17	11	81.191757
11	2	17	11	86.253077
12	2	17	11	87.608806
13	2	17	11	88.330382
14	2	17	11	88.266038
15	2	17	11	88.139136
16	2	17	11	88.078648
17	2	17	11	88.155805
18	2	17	11	88.168921
22	2	17	11	88.191617
23	2	17	11	88.191617
24	2	17	11	88.190421
32	2	17	11	88.198795
15	2	32	11	89.991053
16	2	32	11	90.050076
17	2	32	11	89.963609
22	2	32	11	89.969332
23	2	32	11	89.975059
24	2	32	11	89.962179
32	2	32	11	89.952172

Table 2 Experiment Result using different bit length of multiplier and accumulator

Mult coeff		Mult Acc		SNR (dB)
Wl	iwl	Wl	iwl	
14	2	17	11	88.266038
14	2	18	11	89.093308
14	2	19	11	89.606987
14	2	20	11	89.872659
14	2	21	11	90.020483
14	2	22	11	90.069538
14	2	23	11	90.085461
14	2	24	11	90.079668
14	2	25	11	90.081116
14	2	26	11	90.079668
14	2	27	11	90.079668
14	2	28	11	90.079668
14	2	29	11	90.079668
14	2	30	11	90.079668
14	2	31	11	90.079668
14	2	32	11	90.079668

4. Error Estimation using Kiro-kiro Algorithm

For some of image processing transformations, e.g Discrete Wavelet Transform (DWT), Discrete Cosine Transform (DCT), we do not need precision result. For example in MPEG 4, H.263/H.264, transformation result from DCT or DWT will be limited become 11 bits. The accuracy of multiplier result is limited by maximum error. A DWT using 12 stage multiplier 16 bit x 24 bit produce 90 dB SNR, however, for transmission process using 11 bit length produce approximately 66 dB SNR

Because of large amount data that must be processed within image processing unit, it is needed high speed and efficient multipliers. In order to get optimum bit length used in multiplier, we proposed algorithm to estimate error produced by particular bit length, we called kiro-kiro algorithm.

Assume we want to multiply a(n+1) with b(n+1). Multiplication result is shown as follow

$$a * b = (\alpha + p.\alpha) * (\beta + q.\beta)$$
$$= \alpha\beta + p.\alpha\beta + q.\alpha\beta + p.q.\alpha\beta \text{ ....(1)}$$

where

$$\alpha = a_i >> i \text{ ....(2)}$$
$$\beta = b_j >> j \text{ ....(3)}$$
$$p = w(a_i << i) \text{ ....(4)}$$
$$w = a_{i-1}a_{i-2}..a_1a_0 \text{ ....(5)}$$
$$q = z(b_j << j) \text{ ....(6)}$$
$$z = b_{i-1}b_{i-2}..b_1b_0 \text{ ....(7)}$$

The value of p.q.αβ can be approximated by w'z' value

$$w'z' = (w_{sb(w)_1} >> sb(w)_1) * (z_{sb(z)_1} >> sb(z)_1)$$
$$= '1' >> (sb(w)_1 + sb(z)_1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(8)$$

Assume

$$w' = \eta w = (a_{i-1}\ SHR\ i)$$

and

$$z' = \mu z = (b_{j-1}\ SHR\ j)$$

hence

$$\begin{aligned} w'z' &= (a_{i-1}\ SHL\ i)(b_{j-1}\ SHL\ j) \\ &= (a_{i-1}\ \&\ b_{j-1}) SHL(i+j) \\ &= \eta\mu wz \end{aligned}$$

We can estimate error by following method

$$Error = \frac{(1-\eta\mu)pq}{1+p+q+pq}$$

$$where\ \ 0 \le \eta \le 1\ \ and\ \ 0 \le \mu \le 1$$

By using *kiro-kiro* algorithm, it is known that only 6 non zero bit is needed. By making coefficient length become 12 bits, 6 non zero bits inside all of multiplier constant will be covered.

5. Implementation in Silicon Level

Two different RISC processor is already implemented during this research. We called them CAPRA and GAJAH. CAPRA is a simple load store µP 8 bit, and GAJAH is a ARM7 compatible RISC. These designs can be tested by in circuit testing. Synthesis result using *SYNOPSYS Design Analyzer* and 0.18 µm technology is shown in Table 3.

Table 3. RISC Processor CELL AREA using 0.18 µm technology

RISC	CELL AREA	TEST COVERAGE
CAPRA	19036.99	99.92%
GAJAH	427841.57	Not Tested Yet

The architecture at Figure 3 equipped with scan register for in circuit testing is already synthesis using *SYNOPSYS Design Analyzer* tool and 0.18 µm technology. Table 4 and table 5 show the synthesis result.

Table 4. CELL AREA of DWT Architecture using 0.18 µm technology

DESAIN	CELL AREA
Mc0	3928
Mc1	4943
Mc2	4753
Mc3	3649
Md0	3991
Md1	4776
Md2	4943
Md3	4244
M2c	20048
M2d	19442
M3	27133
Filter 1	60776
Filter 2	58295
Filter 3	59419

All of multipliers in Filter 1 consist of 1 pipeline, however, multipliers in Filter 2 and filter 3 consist of 2 pipelines. All of multipliers use booth multiplier with carry look up and Brent & Kung adder.

Timing analysis result at 20 ns clock period and circuit testing result using *SYNOPSYS TetraMax* are shown in table 5.

Table 5 Timing Analysis and Circuit Testing Result

DESAIN	TIMING ANALISYS (SLACK)	TEST COVERAGE
Filter 1	14.34	99.99%
Filter 2	13.21	99.31%
Filter 3	9.24	100.00%

6. Conclusion

In system-on-chip memory implementation, DWT offers excellent speed and high efficiency. DWT also gives better compression quality for image and video. The optimum performance of one dimension DWT can be reached using 32 x 16 bit multiplier. By implementing constant multiplier, we can make a fast and efficient 32 x 16 bit multiplier. To increase parallelism, Forward DWT and Inverse DWT can be implemented inside different dedicated hardware. Combining microcontroller unit (MCU) with dedicated hardware and memory inside one chip will produce high quality DSP engine. Verification and debugging process are done easily and fast by modeling a system using SystemC platform..

References

[1] I. Daubechies, "Ten Lectures on Wavelets" CBMS-NFS Reg.Conf. Series Appl. Math. SLAM, 1992.

[2] John L Hennessy, David A Patterson, "Computer Architecture A Quantitative Approach", Morgan Kaufmann, 1996