의료영상의 JPEG2000 압축을 위한 저전력 DWT 프로세서의 설계 및 구현

論 文
54D-2-10

Design and Implementation of Low-Power DWT Processor for JPEG2000 Compression of Medical Images

張 永 釩† · 李 元 相* · 兪 善 國**
(Young-Beom Jang · Won-Sang Lee · Sun-Kook Yoo)

Abstract - In this paper, low-power design and implementation techniques for DWT(Discrete Wavelet Transform) of the JPEG2000 compression are proposed. In DWT block of the JPEG2000, linear phase 9 tap and 7 tap filters are used. For low-power implementation of those filters, processor technique for DA(Distributed Arithmetic) filter and minimization technique for number of addition in CSD(Canonic Signed Digit) filter are utilized. Proposed filter structure consists of 3 blocks. In the first CSD coefficient block, every possible 4 bit CSD coefficients are calculated and stored. In second processor block, multiplication is done by MUX and addition processor in terms of the binary values of filter coefficient. Finally, in third block, multiplied values are output and stored in flip-flop train. For comparison of the implementation area and power dissipation, proposed and conventional structures are implemented by using Verilog-HDL coding. In simulation, it is shown that 53.1% of the implementation area can be reduced comparison with those of the conventional structure.

Key Words : JPEG2000, DWT, CSD, 9/7 Filter

1. 서 론

정보량이 큰 의료영상 신호는 효율적인 전송이나 저장을 위하여 작은 정보량으로 신호를 압축하고, 복원된 후의 신호가 최대한 작은 변화를 갖도록 하여야 한다. 따라서, 의료영상 데이터를 효율적으로 저장하고 전송하는 표준인 DICOM(Digital Imaging and Communication in Medicine)이 제정되었다.[1] 정지영상의 표준화된 압축방식으로서 DCT(Discrete Cosine Transform)를 기반으로 하는 JPEG (Joint Photograph Experts Group) 방식이 가장 널리 사용되고 있다.[2] 이 JPEG 방식은 의료영상을 압축하는 데에도 효과적이므로 DICOM에 사용되고 있다. 그러나 JPEG의 핵심인 DCT는 영상을 일정한 크기의 블록으로 나누어 압축을 하므로 고압축에서는 블록화 현상(blocking effect)에 의한 화질열화가 발생한다.[3] 이를 개선하기 위하여 최근에는 DWT(Discrete Wavelet Transform)를 사용하는 압축 방식이 연구되었으며[4] 그 결과, 이와 같은 DWT 기반의 JPEG2000과 같은 정지영상 압축 방식이 표준화되었다.[5][6] 따라서 최근에 DICOM 2003에는 JPEG2000이 추가되었다. JPEG2000은 DWT 필터뱅크를 사용하여 신호를 부호화하므로, JPEG에 비해 큰 영상의 경우에 압축효율이 높은 것으로 알려져 있다. JPEG의 DCT에 비하여 복잡도가 큰 JPEG2000의 DWT는 저전력 구현기술이 요구된다. DWT는 기본적으로 트리구조의 필터뱅크로서 분석단에서는 데시메이션 회로로 구성되며, 합성단에서는 인터폴레이션 회로로 구성된다. 따라서 DWT 분석단과 합성단의 구현 면적을 최소화하고, 저전력으로 구현하고, 처리시간을 최소화하기 위해서는 분석 및 합성 필터의 효율적인 설계가 필수적이다. JPEG2000의 DWT 필터뱅크에서는 (9, 7) 필터와 같은 표준화된 필터가 사용되고 있다. 따라서 JPEG2000의 저전력 구현을 위하여 (9, 7)필터의 저전력 구현이 필수적으로 요구된다. 널리 사용되는 FIR 필터의 저전력 구현방법으로는 CSD(Canonic Signed Digit)형 계수, 덧셈기, 쉬프트 연산을 사용하는 CSD 방식이 있다.[7][8] 이 방식은 필터 계수를 CSD형의 2진수로 변환한 후 CSD의 비트 정보에 따라 입력신호를 쉬프트시켜서 출력을 계산하는 방식이다. 이 방식은 곱셈연산을 덧셈연산과 쉬프트 연산을 사용하여 효과적으로 구현할 수 있으나 곱셈 연산이 많으면 비례적으로 덧셈연산이 많아져서 구현 하드웨어가 커지는 단점이 발생한다. 한 개의 덧셈기로 여러 개의 덧셈연산을 수행하기 힘든 구조이기 때문이다. 또 다른 방법으로는 ROM과 덧셈기를 사용하는 DA 방식이 있다.[9][10] 이 방식은 입력신호를 2의 보수형의 이진수로 변환한 후에 비트 정보에 따라 필터계수들을 더하여 출력을 계산하는 방식이다. 이 방식은 필요한 필터계수들의 더한 값들을 ROM에 미리 저장하여 사용하므로 효과적이다. 그러나 탭이 증가하면 ROM의 크기가 매우 커

† 교신저자, 正會員 : 詳明大 工大 情報通信工學科 教授 · 工博
E-mail : ybjang@smu.ac.kr
* 正 會 員 : 祥明大 工大 情報通信工學科 碩士課程
** 正 會 員 : 延世大 醫學工學教室 · 移動形 應急醫療 情報 시스템 開發 센터 · 副教授 · 工博
接受日字 : 2004年 11月 3日
最終完了 : 2004年 12月 30日

지므로 4탭 정도로 ROM 크기를 제한하게 되는 단점이 생긴다. 본 논문에서는 CSD 방식의 장점과 DA방식의 장점을 모두 이용하여 JPEG2000용 DWT 필터뱅크의 (9, 7) 필터를 저전력으로 구현하는 새로운 방식을 제안한다.

2. JPEG2000의 2차원 DWT 필터뱅크

정지영상의 압축에 사용되는 2차원 DWT의 구조는 그림 1과 같다. 그림 1은 1 레벨의 DWT 필터뱅크이며 JPEG 2000에서는 5 레벨까지 신호를 분해한다.

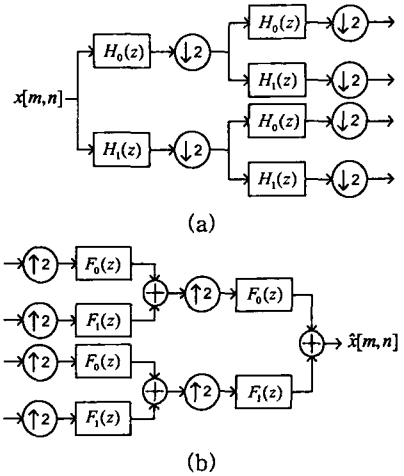


그림. 1 2차원 DWT의 기본 구조, (a)분석단, (b)합성단

Fig. 1 basic two-dimensional DWT structure, (a)Analysis part, (b)Synthesis part.

그림 1(a)의 분석단에서 보듯이, 영상신호는 먼저 수평방향으로 데시메이션 된 후에 수직방향으로 다시 데시메이션 된다. $H_0(z)$는 저역통과 필터이며 JPEG2000에서는 9탭 FIR 필터이므로 9 필터라고 부른다. $H_1(z)$는 고역통과 필터로서 JPEG2000에서는 7탭 FIR 필터이므로 7 필터라고 부른다. 한 레벨의 DWT 변환에 그림 1에서 보듯이 12개의 필터가 필요하므로 5 레벨 JPEG2000의 DWT 변환에는 총 60개의 필터가 필요하다. 따라서 하나의 필터 구조가 총 60개의 필터에 적용되므로 저전력 설계가 매우 중요한 변수가 된다. 본 논문에서는 JPEG2000 저전력 구현을 달성하기 위하여 DWT 변환에 사용되는 9 필터의 저전력 구조를 다음 절에서 제안하도록 하며, 7 필터의 구조는 제안된 9 필터의 구조를 그대로 사용하면 되므로 따로 다루지 않는다.

3. 제안된 저전력 9 필터 구조

3.1 절 필터계수의 CSD형 비트화 블록 설계

JPEG2000용 DWT에서는 그림 1의 2차원 필터뱅크를 사용하여 입력신호를 처리하여야 한다. 그림 1은 1 레벨의 DWT 필터뱅크이며 1 레벨을 처리하는 필요한 9필터와 7필터의 수는 각각 6개이다. JPEG2000에서는 9 필터와 7 필터는 모두 선형위상응답을 갖도록 설계되었다. 이 2개의 필터 중에서 본 논문에서는 9 필터의 저전력 구조를 설계하며, 설계된 구조는 7 필터에도 그대로 적용된다. 필터는 기본적으로 곱셈연산으로 구성되는데, 곱셈연산의 비트 기법은 입력신호의 비트화하는 DA방식과 필터계수를 비트화하는 CSD 방식 등 2 종류가 있다. 본 논문에서는 DA 방식의 장점인 덧셈 프로세서 방식과 CSD 방식의 장점인 덧셈 연산 최소화 방식을 이용하여 구조를 설계한다. 9 필터를 설계하기 위하여, 본 논문에서는 필터계수를 비트화하여 입력신호를 공유하도록 한다. 따라서 그림 2와 같은 Transposed Direct form을 기본 구조로 사용하였다.

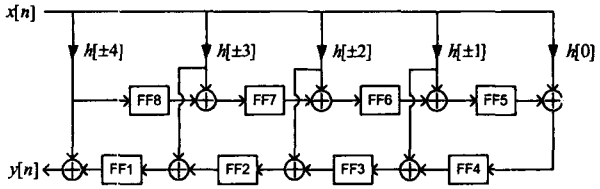


그림 2 9필터의 기본 Transposed Direct form 구조

Fig. 2 Basic 9-filter Transposed Direct form structure

본 논문에서는 그림 2의 기본 구조를 3개의 블록으로 나누어, 각각의 블록에 대한 저전력 구조를 설계하였다. 1단계로 필터계수를 비트화하는 블록은 다음과 같다. 필터 계수의 비트화에는 2의 보수형과 CSD형을 사용할 수 있다. 이 중에서 CSD형이 1의 수가 상대적으로 적게 사용되므로 CSD형을 사용하였다. 9 필터의 계수를 16비트의 CSD형으로 변환하면 표 1과 같다.

표 1에서 -1을 N으로 표기하였다. 곱셈 연산은 입력신호 x[n]을 표 1의 1의 자리만큼 우로 쉬프트시켜서 더해야하는데, 표 1의 2중실선으로 표시된 것과 같이 4 비트씩 연산하는 구조를 제안한다. CSD형의 가능한 4비트의 종류를 모두 계산해 놓은 뒤에 순차적으로 더해서 최종 출력신호를 계산하는 방법을 제안한다. 즉, 표 1에서 보이는 2중 실선의 4비트 연산을 연속적으로 수행하는 저전력 프로세서를 설계하기 위해서는 모든 CSD형의 4비트 종류를 하드웨어로 설계

표 1 9필터의 CSD형 필터 계수

Table 1 CSD filter coefficients of 9filter

	필터계수의 값	2^0	2^-1	2^-2	2^-3	2^-4	2^-5	2^-6	2^-7	2^-8	2^-9	2^-10	2^-11	2^-12	2^-13	2^-14	2^-15
h[±4]	0.02674875741080976	0	0	0	0	0	1	0	0	N	0	0	N	0	N	0	0
h[±3]	-0.01686411844287495	0	0	0	0	0	0	N	0	0	0	N	0	N	0	0	N
h[±2]	-0.07822326652898785	0	0	0	0	N	0	N	0	0	0	0	0	0	N	0	1
h[±1]	0.2668641184427823	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0
h[0]	0.6029490182363579	0	1	0	1	0	N	0	1	0	1	0	N	0	N	0	1

하여야 한다. CSD형의 4비트는 표 2와 같이 모두 21가지가 가능하다.

표 2 4비트 CSD형의 종류
Table 2 Kinds of 4bit CSD

CSD형(양수)					CSD형(음수)				
2^0	2^-1	2^-2	2^-3	값	2^0	2^-1	2^-2	2^-3	값
1	0	1	0	1.25	N	0	N	0	-1.25
1	0	0	1	1.125	N	0	0	N	-1.125
1	0	0	0	1.0	N	0	0	0	-1.0
1	0	0	N	0.875	N	0	0	1	-0.875
1	0	N	0	0.75	N	0	1	0	-0.75
0	1	0	1	0.625	0	N	0	N	-0.625
0	1	0	0	0.5	0	N	0	0	-0.5
0	1	0	N	0.375	0	N	0	1	-0.375
0	0	1	0	0.25	0	0	N	0	-0.25
0	0	0	1	0.125	0	0	0	N	-0.125
0	0	0	0	0.0					

표 2에서 보듯이 4비트의 CSD형 표현은 1.25부터 -1.25까지 21가지 종류가 가능하다. 그러나 음수의 표현은 양수와 부호만 제외하면 같은 값이어서 하드웨어 구현에서는 양수의 값을 그대로 사용하면 되므로 따로 설계할 필요가 없다. 따라서 0.125부터 1.25까지 총 10개의 종류가 가능하므로 입력신호 x[n]의 곱셈연산에 필요한 10개의 CSD형 계수의 곱셈연산은 다음 식과 같이 나타낼 수 있다.

$$\begin{aligned}
1.25x &= x + x\gg 2 && : (1번)\\
1.125x &= x + x\gg 3 && : (2번)\\
1.0x &= x && : (1번)\\
0.875x &= x - x\gg 3 && : (0번)\\
0.75x &= x - x\gg 2 && : (0번)\\
0.625x &= (1.25x)\gg 1 && : (1번)\\
0.5x &= x\gg 1 && : (2번)\\
0.375x &= x\gg 1 - x\gg 3 && : (4번)\\
0.25x &= x\gg 2 && : (5번)\\
0.125x &= x\gg 3 && : (0번)
\end{aligned} \tag{1}$$

식(1)의 가능한 모든 연산 중에서, 표 1의 9 필터에서 실제로 사용되는 수를 식 (1)의 오른쪽 괄호에 표기하였다. 식 (1)의 괄호에서 보듯이 10개중에는 사용되지 않는 종류가 0.875x, 0.75x, 0.125x의 3개가 있음을 알 수 있다. 제안된 식 (1)의 CSD 4비트 곱셈연산의 구조는 그림 3과 같다.

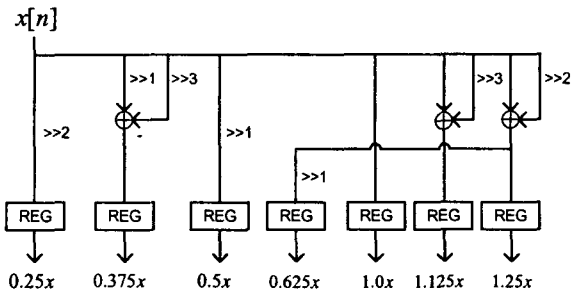


그림 3 제안된 CSD 4비트 곱셈 연산 구조
Fig. 3 Proposed CSD 4bit multiplier Structure

3.2 절 덧셈기 프로세서 블록 설계

그림 2에서 보듯이, 최종 출력 y[n]을 얻기 위해서는 x[n]의 입력신호는 각각 5개의 필터계수와 곱해진 y_4부터 y_0까지의 5개의 중간 결과 값이 필요하다. 이는 식 (1)과 그림 3을 이용하여 다음과 같이 나타낼 수 있다.

$$\begin{aligned}
y_4 &= -(0.5x)\gg 12 - (1.125x)\gg 8 + (0.5x)\gg 4\\
y_3 &= -(1.125x)\gg 12 - (0.25x)\gg 8 - (0.25x)\gg 4\\
y_2 &= -(0.375x)\gg 12 - (1.25x)\gg 4\\
y_1 &= (1.0x)\gg 12 + (0.25x)\gg 8 + (0.25x)\gg 4 + (0.25x)\\
y_0 &= -(0.375x)\gg 12 + (0.375x)\gg 8 - (0.375x)\gg 4 + (0.625x)
\end{aligned} \tag{2}$$

식 (2)의 y_4부터 y_0의 값들은 모두 그림 3에서 만들어진 7개의 값들의 쉬프트 연산으로 이루어지도록 설계하였다. 따라서 식 (2)의 연산은 MUX 회로를 사용하여 값들을 선택하도록 한 후에 덧셈과 쉬프트 연산으로 구현할 수 있다. y_4의 값을 구하는 데에는 3 clock이 필요하며, y_4부터 y_0의 모든 값들을 구하는 데에는 총 16 clock이 필요하게 된다. 계산이 끝난 y_i, $i=4,3,2,1,0$ 의 값들은 각각스위치를 통하여 다음 블록으로 보내지도록 제어한다. 이와 같은 중간 값을 계산하는 구조를 그림 4와 같이 설계하였다.

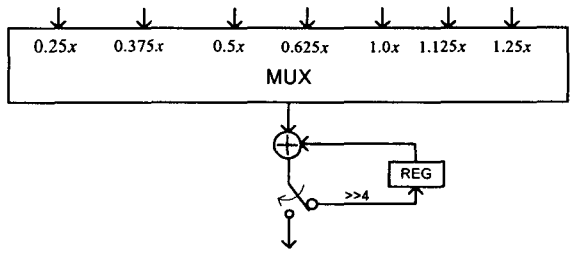


그림 4 제안된 필터계수의 곱셈연산 구조
Fig. 4 Proposed filter coefficients multiplier Structure

그림 4에서 MUX의 select는 0.5->1.125->0.5->1.125->0.25->0.25->0.375->1.25->1.0->0.25->0.25->0.25->0.375

->0.375->0.375->0.625의 순서로 이루어지도록 설계한다. 또한 스위치는 y_i, $i=4,3,2,1,0$ 의 값들이 계산이 완료되면 아래 방향으로 보내도록 제어한다.

3.3 절 출력용 플립플롭 블록 설계

그림 4에서 계산된 중간 값들을 사용해서 출력용 플립플롭(flip-flop, FF)들을 갱신하여야한다. 그림 2의 8개 플립플롭은 8개의 덧셈기를 사용하고 있다. 그러나 y_4부터 y_0의 값들이 동시에 얻어지는 것이 아니므로 8개의 덧셈기를 동시에 사용할 필요는 없다. 따라서 1개의 덧셈기와 8개의 플립플롭을 사용하여 그림 5와 같은 구조를 설계하였다.

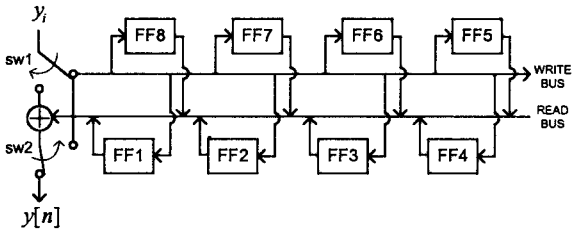


그림 5 제안된 출력용 FF 구조

Fig. 5 Proposed output FF Structure

그림 5에서 보듯이, 제안된 구조는 1개의 덧셈기, 2개의 스위치, 8개의 플립플롭, 그리고 READ와 WRITE BUS로 구성하였다. sw1은 y_4가 들어온 후 1clock 동안 WRITE BUS와 연결되어 FF8을 WRITE하게 되고, 그 다음 clock부터 다시 y_4가 들어올 때까지 덧셈기와 연결되게 된다. sw2는 sw1이 덧셈기와 연결된 후 한 clock 동안 READ된 FF1과 더해져 출력으로 나가게 되고, 다음 y_4가 들어올 때까지 WRITE BUS와 연결된다. y_3이 들어오면 1clock 동안 FF8이 READ되어 y_3과 더해진 후 FF7에 WRITE되고, 다음 clock에는 READ된 FF2와 더해져 FF1을 WRITE하게 된다. 마찬가지로 y_2가 들어오면 2clock 동안 각 1clock씩 READ된 FF7, FF3과 더해져 FF6과 FF2를 WRITE하게 되고, y_1가 들어오면 각 1clock씩 READ된 FF6, FF4와 더해져 FF5와 FF3을 WRITE하게 된다. 마지막으로 y_0가 들어오면 1clock 동안 FF5의 READ된 데이터와 더해져 FF4를 WRITE하게 되어 y_4부터 y_0가 들어오는 1cycle 동안 모든 플립플롭들이 갱신된다. 지금까지 제안한 그림 3, 4, 5의 구조를 연결한 전체 구조는 다음 그림 6과 같다.

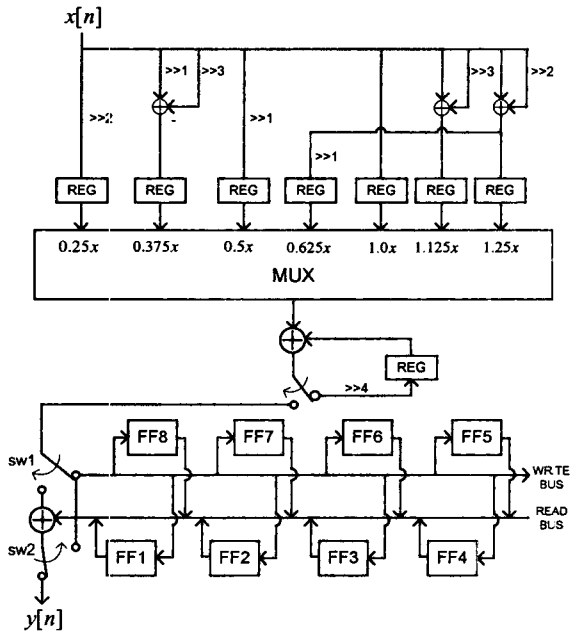


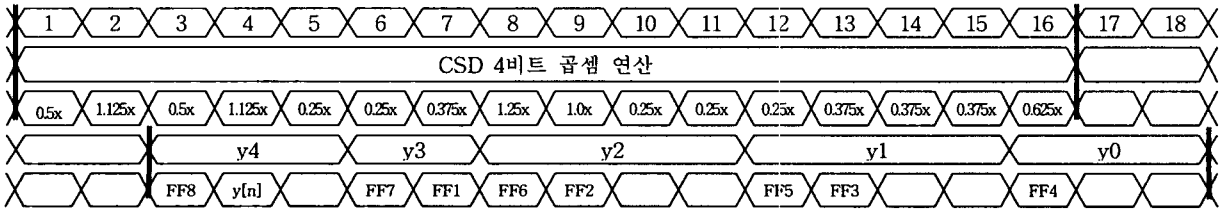
그림 6 제안된 저전력 9 필터 구조

Fig. 6 Low-Power of proposed 9 filter Structure

그림 6은 JPEG2000용 9 필터의 제안된 구조이다. 그림 6에서 보듯이 입력신호 x[n]은 쉬프트와 덧셈기를 사용하여 0.25x[n]부터 1.25x[n]까지 7가지 종류가 만들어진 후에 레지스터에 각각 저장된다. 실질적인 연산은 표1의 $h[\pm 4]$의 LSB 4비트부터 시작된다. 따라서 한 개의 16비트 계수의 곱셈을 수행하는데 4 clock이 필요하며 5개의 필터계수 곱셈을 모두 수행하려면 20 clock이 필요하지만 0은 계산하지 않아도 되기 때문에 16 clock이면 된다. 16개의 각각의 clock에 요구되는 모든 종류의 가능한 값들은 식 1에서 알 수 있듯이 7개의 레지스터에 미리 계산되어 있으므로 MUX를 통하여 선택하기만 하면 된다. 이렇게 완성된 $h[\pm n]$, $n=4,3,2,1,0$ 곱셈이 완료된 중간 값들은 출력 y[n]과 FF들의 값을 갱신하는데 사용된다. 제안된 구조는 미리 계산된 7가지의 곱셈 값들이 매 clock마다 1개씩 선택하여 사용하는 덧셈 프로세서 방식이므로 구현 하드웨어 크

그림 7 제안된 9 필터 구조의 타이밍

Fig. 7 Timing of proposed 9 filter Structure



기를 현저히 감소시킬 수 있다. 즉, 제안된 9 필터 구조는 5개의 덧셈기와 8개의 FF으로 구성되는 저전력 프로세서 구조이다. 이와 같이 16 clock이 지나면 x[n]에 대한 모든 필터 연산이 완료되므로 16 clock에 대한 연산의 타이밍은 그림 7과 같다.

그림 7의 1행은 clock의 순서이고, 2행은 그림 3의 CSD 4비트 곱셈연산 블록으로서 16 clock마다 연산을 수행한다. 그림 7의 3행은 그림 4의 덧셈 프로세서의 연산으로서 매 clock마다 덧셈연산을 수행한다. 그림 7의 4행은 y_4부터 y_0의 곱셈 결과 값들을 연산하는 타이밍이다. 마지막으로 그림 7의 5행은 출력 y[n]과 8개의 플립플롭을 갱신하는 시간 순서이다.

4. 구현 및 성능비교

4.1 절 덧셈의 수 성능 비교

본 논문이 제안한 구조와 기존의 CSD형 필터 구조의 성능을 비교해보기로 한다. 이 절에서는 먼저 각 구조의 덧셈기의 수를 비교하였다. 기존의 구조는 [8]에서 제안한 공통패턴 공유 방식을 사용하여 구현하였다. 먼저 CSD형 계수에 공통패턴을 2중 실선으로 표기하면 표 3과 같다.

표 3 CSD형 계수를 사용한 기존 9필터 구조의 덧셈수
Table 3 Sum of additional CSD coefficients used 9filer

	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	덧셈수
h0	0	1	0	1	0	N	0	1	0	1	0	N	0	N	0	1	3
h1	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	2
h2	0	0	0	0	N	0	N	0	0	0	0	0	0	N	0	1	1
h3	0	0	0	0	0	0	N	0	0	0	N	0	N	0	0	N	2
h4	0	0	0	0	0	1	0	0	N	0	0	N	0	N	0	0	2

표 3에서 공통패턴을 만드는데 2개의 덧셈기가 필요하며 각 필터계수를 곱하는데 필요한 덧셈의 수를 표 3의 맨 오른쪽 열에 나타내었다. 이와 같은 CSD형 계수와 공통패턴 공유방식을 사용한 기존의 필터구조는 그림 8과 같다.

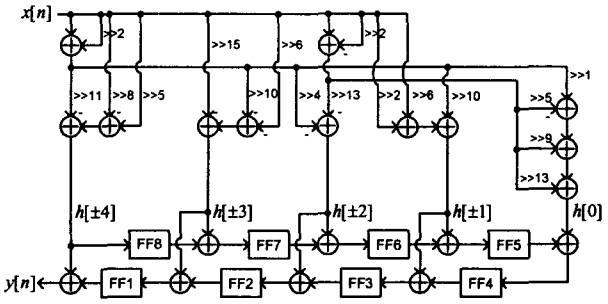


그림 8 기존의 CSD형 공통패턴을 사용한 9필터 구조
Fig. 8 CSS CSD used 9filer structure

그림 8에서 보듯이 20개의 덧셈 연산이 사용된다. 이와 같이 만들어진 기존 9필터 구조와 3절에서 제안된 그림 6 구조의 덧셈 연산의 수를 비교하면 표 4와 같다.

표 4 제안 구조의 덧셈수 비교
Table 4 Compared additions of proposed structure

	제안구조	CSD 공통패턴 사용 기존 구조
블록별 덧셈수	CSD 계수용(3개) 프로세서용(1개) 출력 FF용(1개)	CSD 계수용(12개) 출력 FF용(8개)
총 덧셈수	5개	20개

표 4에서 보듯이 제안구조의 덧셈수는 총 5개로서 기존 구조의 20개와 비교하여 75%가 감소됨을 볼 수 있다. 따라서 제안 구조는 기존의 9 필터 구조와 비교하여 구현면적을 현저히 감소시킬 수 있음을 알 수 있다.

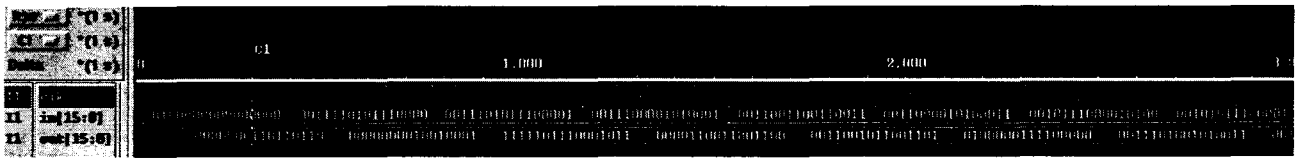


그림 9 제안 구조의 Simulation 결과
Fig. 9 Simulation Result of proposed structure

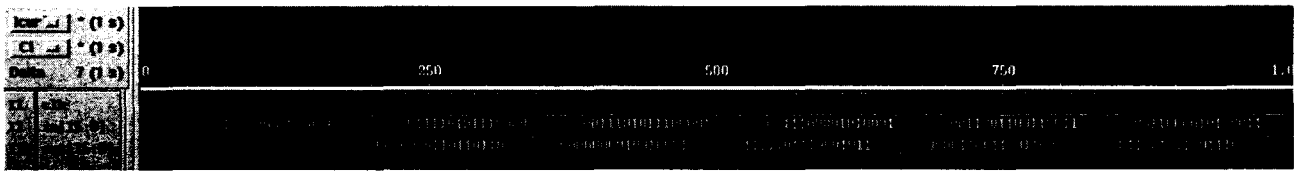


그림 10 기존 구조의 Simulation 결과
Fig. 10 Simulation Result of 9filter structure

4.2 절 구현면적과 전력소모 성능 비교

이 절에서는 Verilog-HDL 코딩 구현을 통하여 제안구조와 기존 구조의 구현면적과 총 dynamic 전력소모를 비교하였다. 제안된 그림 6의 9 필터 구조와 비교하기 위한 기존 구조로는 이전 절에서 설계한 그림 8의 구조를 사용하였다. 먼저 동작 검증을 위하여 각각의 구조를 Synopsys VCS를 사용하여 검증한 결과 그림 9와 10과 같이 같은 결과가 출력됨을 확인하였다. 즉, 그림 9와 10의 16비트 output을 관찰하면 같은 값이 출력되고 있음을 관찰할 수 있다.

Verilog-HDL 코딩을 합성한 결과 9 필터에 대한 각 구조의 구현면적은 표 5와 같다. 즉, 표 5에서 보듯이 곱셈기를 사용한 기존 구조와 비교하여 53.1%의 구현 면적을 감소시킬 수 있었다.

표 5 제안 구조의 구현면적과 전력소모 비교(9 필터)
Table 5 Relative Power Consumption and area of proposed Structure

	제안 구조	곱셈기 사용 기존 구조
곱셈기	0	5
덧셈기	5	8
플립플롭	8	8
구현면적	2048.34 (46.9%)	4366.02

제안 구조와 기존 구조의 합성된 논리회로도는 각각 그림 11과 12와 같다.

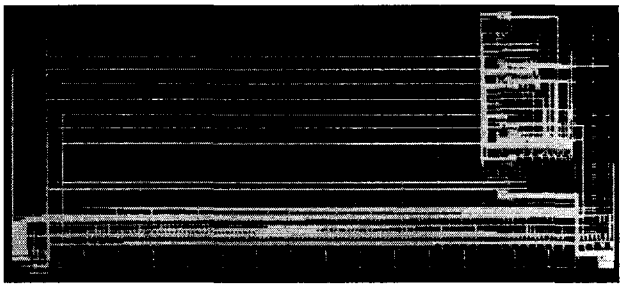


그림 11 제안 구조의 Synthesis 결과
Fig. 11 Synthesis Result of proposed structure

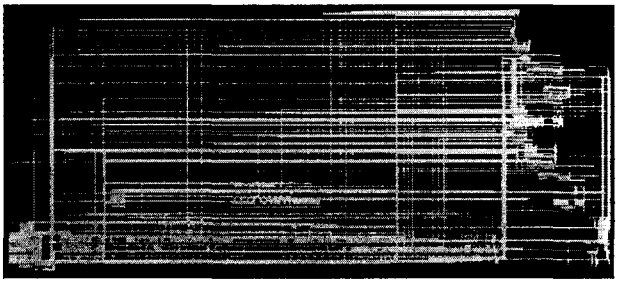


그림 12 기존 구조의 Synthesis 결과
Fig. 12 Synthesis Result of 9filter structure

4. 결 론

본 논문에서는 의료영상의 JPEG2000 압축을 위한 DWT 변환용 (9, 7) 필터의 저전력 설계 및 구현 방식을 제안하였다. 필터 계수를 4비트의 CSD형으로 표현하여 덧셈의 수를 최소화할 수 있었으며, 4비트씩 계산된 중간결과를 한 개의 덧셈 연산 프로세서를 사용하여 구현함으로서 구현면적을 현저히 줄일 수 있었다. 이와 더불어 기존 곱셈기 구조에서 사용되는 출력용 플립플롭 8개와 덧셈기 8개는 1개의 덧셈기와 8개의 플립플롭을 사용하여 효율적으로 구현될 수 있음을 보였다. 제안된 덧셈 프로세서 구조는 기존의 CSD 방식을 사용한 구조와 비교하여 덧셈 연산의 수를 20개에서 5개로 줄일 수 있었다. 구현면적과 전력소모를 알아보기 위하여 Synopsys VCS를 사용하여 Verilog-HDL 코딩 시뮬레이션을 수행한 결과 기존 CSD 구조의 구현면적과 비교하여 53.1%를 줄일 수 있었다. 따라서 제안된 덧셈 프로세서 구조는 의료영상의 JPEG2000용 저전력 DWT 구조로서 널리 사용될 수 있을 것이다.

참 고 문 헌

[1] PS 3.1-2003, DICOM(Digital Imaging and Communication in Medicine) Part 1: Introduction and Overview, National Electrical Manufacturers Association, 2003.

[2] ITU-T Recommendation T.81, "Information Technology - Digital Compression and Coding of Continuous-tone Still Images-requirements and Guidelines", ITU-T, Feb. 1992.

[3] Rafel C Gonzalez, Richard E. Woods, Digital image processing, Prentice Hall, pp. 467-485, 2002.J. C. Ventura, Digital audio gain control for hearing aid, Proc. IEEE ICASSP, pp. 2049-2052, 1989.

[4] C. Sidney Burrus, Ramesh A. Gopinath, Haitao Guo, Introduction to Wavelets and Wavelet Transforms, Prentice Hall, pp. 1-40, 1998.

[5] Information Technology - JPEG2000 Image Coding System, ISO/IEC Final Draft International Standard 15444-1, ITU Recommendation T.800, 2000.

[6] M. Rabbani, R. Joshi "An overview of the JPEG2000 still image compression standard," Signal Processing: Image Communication, 17, pp. 3-48, 2002.

[7] R. W. Reitwiesner, "Binary arithmetic," in Advances in Computers, New York: Academic, vol. 1, pp. 231-308, 1966.

[8] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing, vol. 43, No. 10, pp. 677-688, Oct. 1996.

[9] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review", IEEE ASSP Magazine, pp. 4-19, Jul. 1989.

[10] A. Sinha and M. Mehendale, "Improving area

efficiency of FIR filters implemented using distributed arithmetic," Proc. Eleventh International Conference on VLSI Design, pp. 104-109, 1998.

저 자 소 개

장 영 범 (張 永 釩)

1958년 8월 8일생. 1981년 연세대 전기공학과 졸업. 1990년 Polytechnic Univer- sity(New York) 대학원 공학석사. 1994년 Polytechnic University (New York) 대학원 공학박사. 1983년~1999 삼성전자 시스템LSI사업부 수석연구원, 2002년~현재 상명대학교 정보통신공학과 교수

Tel : 041) 550 - 5353
Fax : 041) 550 - 5355
E-mail : ybjang@smu.ac.kr

이 원 상 (李 元 相)

1978년 10월 4일생. 2004년 상명대학교 컴퓨터 시스템 졸업. 2004년 2월 ~현재 상명대학교 대학원 컴퓨터정보통신공학과 석사과정

Tel : 041) 550 - 5349
Fax : 041) 550 - 5355
E-mail : windstorm5@smu.ac.kr

유 선 국(兪 善 國)

1959년 1월 8일 생. 1981년 연세대 전기공학과 졸. 1983년, 1989년 동대학원 전기공학과(석, 박사). 1990-1995 순천향대 전기공학 전임강사, 조교수. 1998-2000 The University of Iowa Visiting Associate. 1995년 - 현재 연세대학교 의학공학교실 부교수

E-mail : sunkyoo@yumc.yonsei.ac.kr