ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 13-02-05

Journal of IKEEE.Vol.17,No.2,119~128,June 2013
http://dx.doi.org/10.7471/ikeee.2013.17.2.119

저면적 HEVC 코어 변환기 아키텍처 설계

Design of Low-Area HEVC Core Transform Architecture

한 승 목*, 남 우 진*, 이 성 수*★

Seung-Mok Han*, Woo-Jin Nam*, Seongsoo Lee*★

Abstract

This paper proposes and implements an core transform architecture, which is one of the major processes in HEVC video compression standard. The proposed core transform architecture is implemented with only adders and shifters instead of area-consuming multipliers. Shifters in the proposed core transform architecture are implemented in wires and multiplexers, which significantly reduces chip area. Also, it can process from 4×4 to 16×16 blocks with common hardware by reusing processing elements. Designed core transform architecture in 0.13um technology can process a 16×16 block with 2-D transform in 130 cycles, and its gate count is 101,015 gates.

요 약

본 논문에서는 차세대 동영상 압축 표준인 HEVC의 핵심 프로세스 중 하나인 코어 변환기를 설계하고 이를 합성한 후 검증하였다. 제안하는 코어 변환기는 면적을 많이 차지하는 곱셈기 대신에 덧셈기와 쉬프터만을 사용하였으며, 쉬프터도 실제로는 와이어 연결과 멀티플렉서만을 사용하여 면적을 크게 줄였다. 또한 하나의 하드웨어로 4×4에서 16×16 블록까지 모두 처리할 수 있도록 설계하였으며, 이를 위해서 연산처리기를 재사용하는 아키텍쳐를 제안하였다. 0.13um 공정으로 설계된 코어 변환기는 16×16 블록을 2-D 변환 처리하는데 160 사이클이 소요되며 게이트 수는 101,015 게이트이다.

Key words : HEVC, core transform, low area, shift-and-add, PE reuse

Ⅰ. 서론

최근 들어 HDTV의 해상도를 넘어서는 초고화질, 초고해상도 영상이 속속 등장하면서, 기존의 H.264/AVC (advanced video coding)[1]-[3]와 같은 동영상 압축 표준으로는 이러한 영상을 수용할 수 없는 문제점에 봉착하였다. 특히 4k급, 8k급 UHD (ultra high definition) 영상과 같은 초고화질 초고해상도 영상의 경우 기존의 동영상 압축 표준보다 훨씬 더 높은 압축률을 필요로 하며, 동영상 압축 및 복원에 필요한 연산량도 훨씬 더 높아지게 된다.

이에 따라 ITU-T 산하의 VCEG (video coding experts group)와 ISO/IEC 산하의 MPEG (moving picture experts group)은 JCT-VC (joint collaborative team on video coding)이라는 공동 작업반을 구성하여 2013년 2월 HEVC (high efficiency video coding)[4]라는 새로운 동영상 압축 표준을 개발하였다.

* School of Electronic Engineering, Soongsil University, sslee@ssu.ac.kr, 010-9182-3835

★ Corresponding author

※ Acknowledgment

"This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0025041)"

Manuscript received May 6, 2013; revised May 28, 2013 ; accepted May 28. 2013

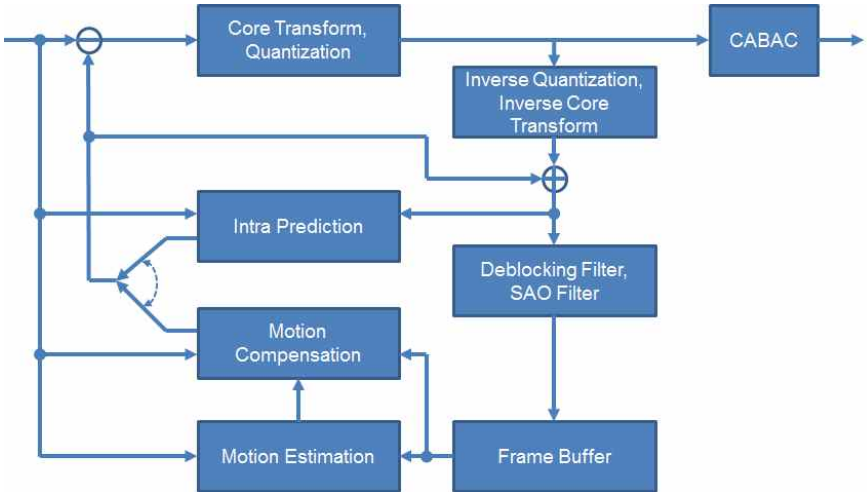

Fig. 1. Block diagram of an HEVC encoder
그림 1. HEVC 부호화기의 블록도

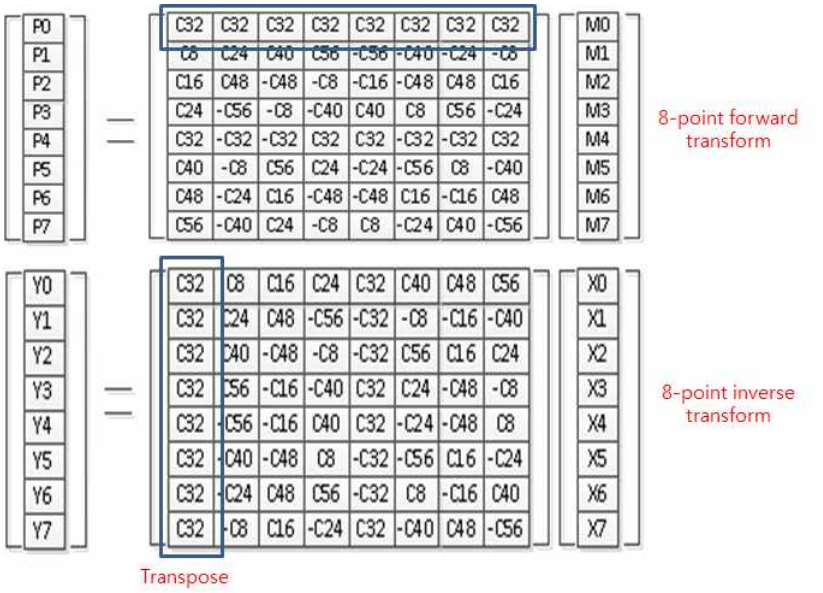

Fig. 2. Relationship between forward and inverse transforms in HEVC core transform
그림 2. HEVC 코어 변환에서 정변환 및 역변환 사이의 관계

HEVC는 기존의 H.264/AVC가 사용하는 압축 기법을 대부분 수용하면서도 압축률을 크게 높이고, 크게 높아진 영상 해상도에 효과적으로 대응하며, 하드웨어 구현에 편리하도록 비트열의 병렬 처리에 적합하도록 개발되었다.

HEVC는 기존의 H.264/AVC 동영상 압축 표준에 비해 1.5배 정도 높은 압축 효율을 보이지만, 압축 효율을 높이기 위해 사용하는 복잡한 압축 방식 때문에 H.264/AVC에 비해 2~4배 가량 높은 연산량이 필요하다. 이 때문에 HEVC를 SoC 칩으로 구현할 때에는 면적을 줄이고 동작 속도를 높이기 위해 아키텍처 개발과 회로 설계에 세심한 주의를 기울여야 할 필요가 있다.

본 논문에서는 HEVC의 핵심 프로세스 중의 하나

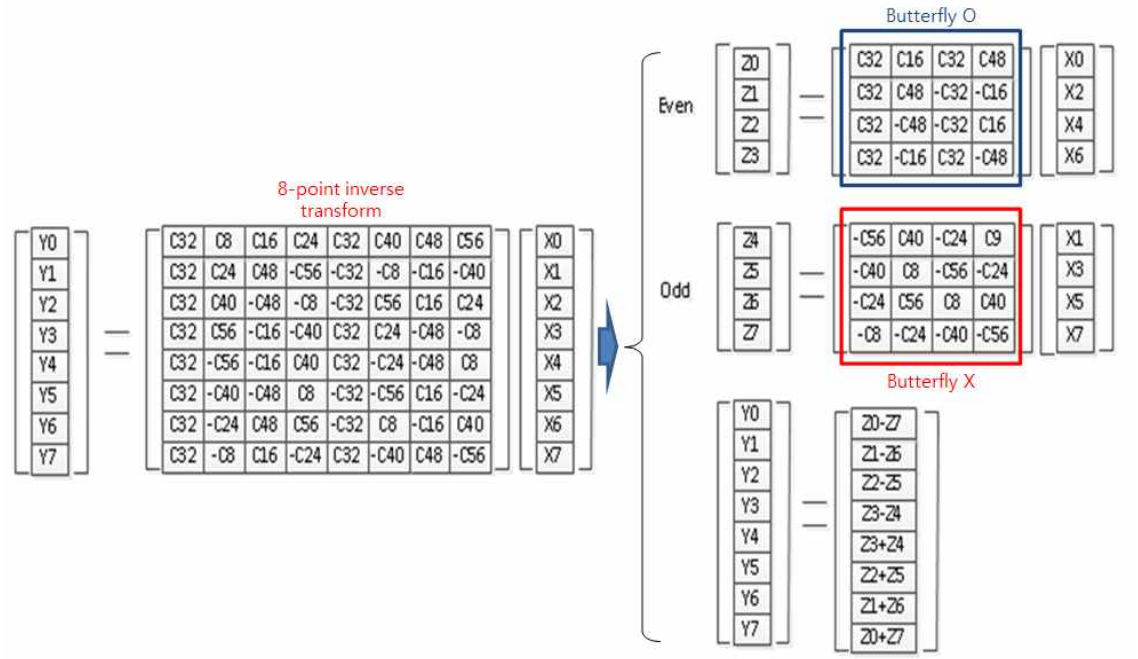

Fig. 3. Butterfly structures of even and odd parts in HEVC core transform
그림 3. HEVC 코어 변환에서 짝수 위치와 홀수 위치의 버터플라이 구조

인 코어 변환 (core transform)을 하드웨어로 구현할 때의 면적을 줄이고 동작 속도를 높인 코어 변환기를 설계하고 이를 칩으로 구현하였다. 제안하는 코어 변환기는 면적을 많이 차지하는 곱셈기 대신에 덧셈기와 쉬프터만을 사용하였으며, 쉬프터도 실제로는 와이어 연결과 멀티플렉서만을 사용하여 면적을 크게 줄였다. 또한 HEVC 코어 변환기가 다양한 크기의 블록을 처리해야 하는 점을 감안하여 하나의 하드웨어로 4×4에서 16×16 블록까지 모두 공통으로 처리할 수 있도록 설계하였으며, 이를 위해서 연산처리기 (processing element)를 재사용하는 아키텍처를 제안하였다.

II. HEVC 코어 변환

그림 1은 HEVC 부호화기의 블록도이다. HEVC를 구성하는 주요 프로세스는 코어 변환 (core transform), 양자화 (quantization), 인트라 예측 (intra prediction), 움직임 추정 (motion estimation), 움직임 보상 (motion compensation), 디블록킹 필터 (deblocking filter), SAO 필터 (sample adaptive offset filter), CABAC 부호화 (context adaptive binary arithmetic coding)를 들 수 있다. 이 중에서 코어 변환은 4×4에서 32×32까지 다양한 크기의 블록에 대해 변환을 수행하여야 하나, 실제적으로 많은 경우에 32×32 크기의 블록은 잘 사용되지 않으므로 본 논문에서는 4×4에서 16×16 크기의 블록까지 단일 하드웨어로 처리할 수 있는 코어 변환기를 설계하였다.

HEVC 코어 변환은 H.264/AVC와 마찬가지로 2-D 변환을 위해 X 방향과 Y 방향으로 두 번의 1-D 변환을 수행한다. HEVC 코어 변환은 기존의 H.264/AVC와 크게 다음과 같이 세 가지 점에서 차이가 있다.

먼저 HEVC 코어 변환에서는 정변환 (forward transform)과 역변환 (inverse transform)에 사용되는 행렬이 계수는 동일하고 위치만 전치 (transpose)된다 [5]. 그림 2는 8-point HEVC 코어 변환에서 정변환과 역변환의 계수를 나타낸 것인데, 역변환에 사용되는 계수는 정변환에 사용되는 계수를 행과 열의 위치만 바꿔 전치시킨 것을 알 수 있다. 2-D 변환을 위해서는 어차피 1-D 변환을 X 방향과 Y 방향으로 행과 열을 전치시켜가며 두 번 수행해야 하므로, 결과적으로 HEVC 코어 변환에서는 동일한 하드웨어로 정변환과 역변환을 동시에 수행할 수 있다.

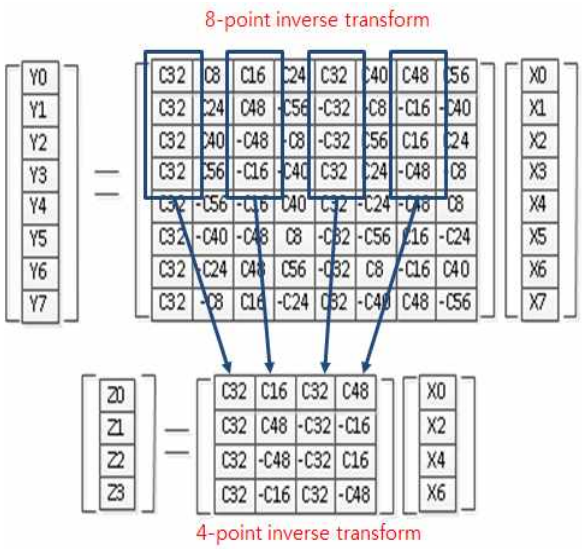

Fig. 4. Relationship between larger size and smaller size blocks in HEVC core transform
그림 4. HEVC 코어 변환에서 상위 크기 블록과 하위 크기 블록 사이의 관계

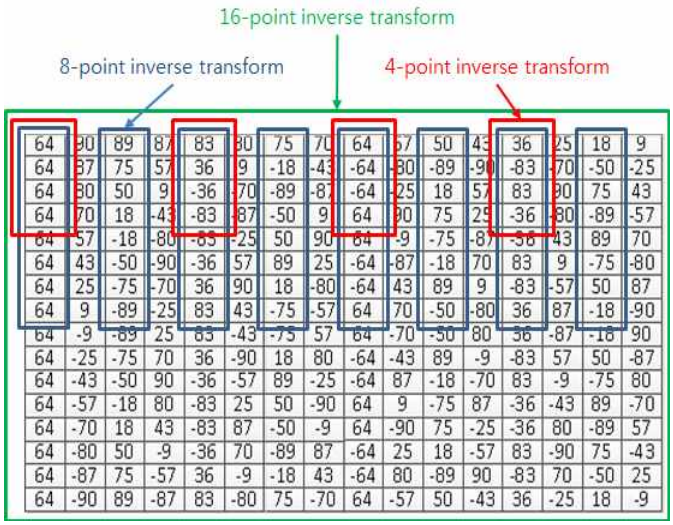

Fig. 5. Coefficients of 16-point HEVC core transform
그림 5. 16-point HEVC 코어 변환의 계수

또한 HEVC 코어 변환에서는 버터플라이 구조 (butterfly structure)가 짝수 위치 부분 (even part)에만 적용되고 홀수 위치 부분 (odd part)에는 적용이 되지 않는다[6]. 그림 3은 8-point HEVC 코어 역변환을 나타낸 것인데, 짝수 위치 부분에서는 부호를 제외한 계수가 상하 대칭이기 때문에 버터플라이 구조를 사용하여 곱셈의 개수를 반으로 줄일 수 있지만 홀수 위치 부분에서는 대각선으로 대칭이기 때문에 곱셈의 개수를 줄일 수 없다.

또한 HEVC 코어 변환에서는 상위 크기 블록에서 짝수 위치 부분의 계수가 하위 크기 블록의 전체 계수와 동일하다[7]. 그림 4는 8-point 및 4-point HEVC 코어 역변환을 나타낸 것인데, 8-point 역변환의 짝수 위치 계수와 4-point 역변환의 전체 계수가

Table 1. Addition-shift operations of 16-point HEVC core transform
표 1. 16-point HEVC 코어 변환의 덧셈-쉬프트 연산

계수	덧셈-쉬프트 연산		덧셈수	뺄셈수	쉬프트수
A*4	= A*(4)	=(A≪2)	0	0	1
A*9	= A*(8+1)	=(A≪3)+(A≪0)	1	0	2
A*13	= A*(8+4+1)	=(A≪3)+(A≪2)+(A≪0)	2	0	3
A*18	= A*(16+2)	=(A≪4)+(A≪1)	1	0	2
A*22	= A*(16+4+2)	=(A≪4)+(A≪2)+(A≪1)	2	0	3
A*25	= A*(16+8+1)	=(A≪4)+(A≪3)+(A≪0)	2	0	3
A*31	= A*(32-1)	=(A≪5)**-(A≪0)**	0	**1**	2
A*36	= A*(32+4)	=(A≪5)+(A≪2)	1	0	2
A*38	= A*(32+4+2)	=(A≪5)+(A≪2)+(A≪1)	2	0	3
A*43	= A*(32+8+2+1)	=(A≪5)+(A≪3)+(A≪1)+(A≪0)	3	0	4
A*46	= A*(32+8+4+2)	=(A≪5)+(A≪3)+(A≪2)+(A≪1)	3	0	4
A*50	= A*(32+16+2)	=(A≪5)+(A≪4)+(A≪1)	2	0	3
A*54	= A*(32+16+4+2)	=(A≪5)+(A≪4)+(A≪2)+(A≪1)	3	0	4
A*57	= A*(32+16+8+1)	=(A≪5)+(A≪4)+(A≪3)+(A≪1)	3	0	4
A*61	= A*(64-2-1)	=(A≪6)**-(A≪1)-(A≪0)**	0	**2**	3
A*64	= A*(64)	=(A≪6)	0	0	1
A*67	= A*(64+2+1)	=(A≪6)+(A≪1)+(A≪0)	2	0	3
A*70	= A*(64+4+2)	=(A≪6)+(A≪2)+(A≪1)	2	0	3
A*73	= A*(64+8+1)	=(A≪6)+(A≪3)+(A≪0)	2	0	3
A*75	= A*(64+8+2+1)	=(A≪6)+(A≪3)+(A≪1)+(A≪0)	3	0	4
A*78	= A*(64+8+4+2)	=(A≪6)+(A≪3)+(A≪2)+(A≪1)	3	0	4
A*80	= A*(64+16)	=(A≪6)+(A≪4)	1	0	2
A*83	= A*(64+16+2+1)	=(A≪6)+(A≪4)+(A≪1)+(A≪0)	3	0	4
A*85	= A*(64+16+4+1)	=(A≪6)+(A≪4)+(A≪2)+(A≪0)	3	0	4
A*87	= A*(64+32-8-1)	=(A≪6)+(A≪5)**-(A≪3)-(A≪0)**	1	**2**	4
A*88	= A*(64+16+8)	=(A≪6)+(A≪4)+(A≪3)	2	0	3
A*89	= A*(64+16+8+1)	=(A≪6)+(A≪4)+(A≪3)+(A≪0)	3	0	4
A*90	= A*(64+16+8+2)	=(A≪6)+(A≪4)+(A≪3)+(A≪1)	3	0	4

동일하므로 8-point 코어 변환기의 짝수 위치 부분을 4-point 코어 변환기로 활용할 수 있다. 마찬가지로 16-point 코어 변환기의 짝수 위치 부분도 8-point 코어 변환기로 활용할 수 있다. 따라서 이 점을 잘 활용하면 16×16 코어 변환기 하나로 16×16뿐만 아니라 8×8 및 4×4 코어 변환도 동시에 수행할 수 있다.

III. HEVC 코어 변환기 설계

1. 아키텍처 설계

위에서 살펴보았듯이, HEVC 코어 변환은 (1) 정변환과 역변환의 계수가 동일하며, (2) 짝수 위치 부분에서는 버터플라이 구조가 적용되지만 홀수 위치 부분에서는 적용되지 않으며, (3) 상위 크기 블록의 짝

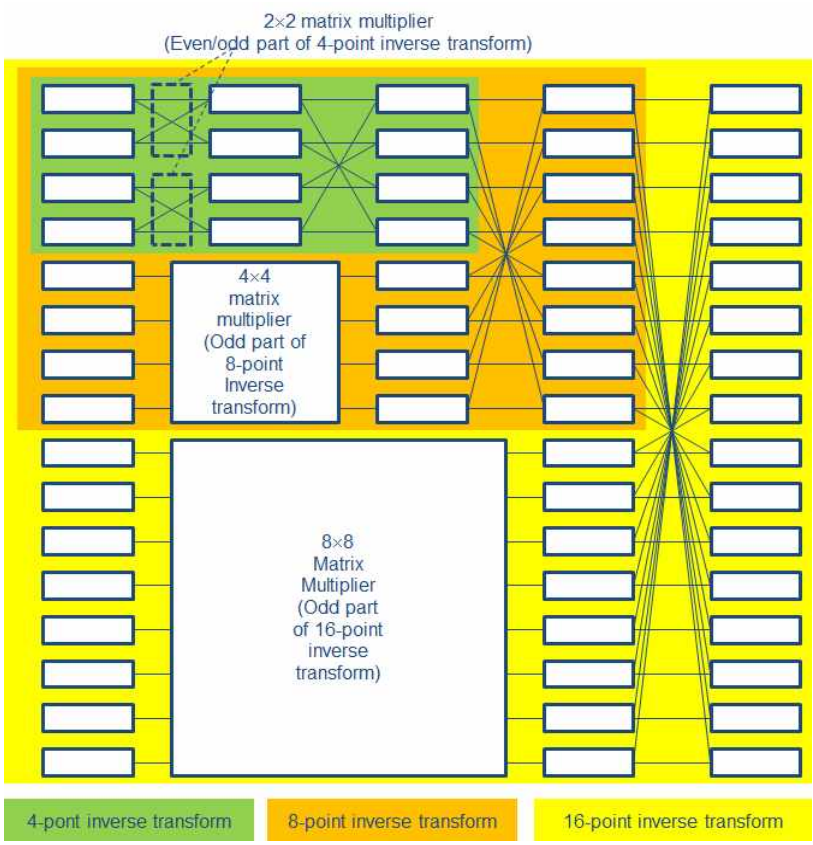

Fig. 6. Conventional HEVC core transform architecture
그림 6. 기존의 HEVC 코어 변환기 구조

수 위치 부분과 하위 크기 블록 전체의 계수가 동일하다. 따라서 16×16 크기 블록을 처리할 수 있는 16-point HEVC 코어 역변환기 하나로 4×4, 8×8, 16×16 크기의 블록의 HEVC 코어 정변환 및 역변환을 모두 처리할 수 있다.

이러한 성질에 따라 그림 5의 계수를 사용하는 HEVC 코어 변환기를 설계하면 그림 6과 같이 구현할 수 있으며[8], 단일 하드웨어로 4×4, 8×8, 16×16 크기 블록을 모두 처리할 수 있다. 그러나 HEVC에서 16×16 크기 블록 1개는 4×4 크기 블록 16개에 해당하지만 그림 6의 구조는 4-point, 8-point, 16-point 역변환을 처리하는데 걸리는 시간이 동일하기 때문에 4×4 크기 블록 16개를 처리하는데 걸리는 시간은 16×16 크기 블록 1개를 처리하는데 걸리는 시간의 4배가 된다.

본 논문에서는 이러한 문제점을 해결하기 위해서 그림 7과 같은 구조의 HEVC 코어 변환기를 제안한다. 제안하는 HEVC 코어 변환기는 연산처리기를 재사용함으로서 (1) 그림 7과 같이 5 사이클 걸려서 1개의 16-point 역변환기로 동작하거나 (2) 그림 8과 같이 4 사이클 걸려서 2개의 8-point 역변환기로 동작하거나 (3) 그림 9와 같이 2 사이클 걸려서 4개의 4-point 역변환기로 동작이 가능하다. 따라서 (1) 16×16 크기 블록 1개를 1-D로 처리하는 데에는 80 사이클 (2-D의 경우 160 사이클), (2) 8×8 크기 블록 4개를 1-D로 처리하는 데에는 64 사이클 (2-D의 경우 128 사이클), (3) 4×4 크기 블록 16개를 1-D로 처리하는 데에는 32 사이클 (2-D의 경우 64 사이클)이 소요된다.

2. 연산처리기 설계

16-point HEVC 코어 변환에 사용되는 계수는 고정되어 있으므로 곱셈기 없이 덧셈기와 쉬프터만으로

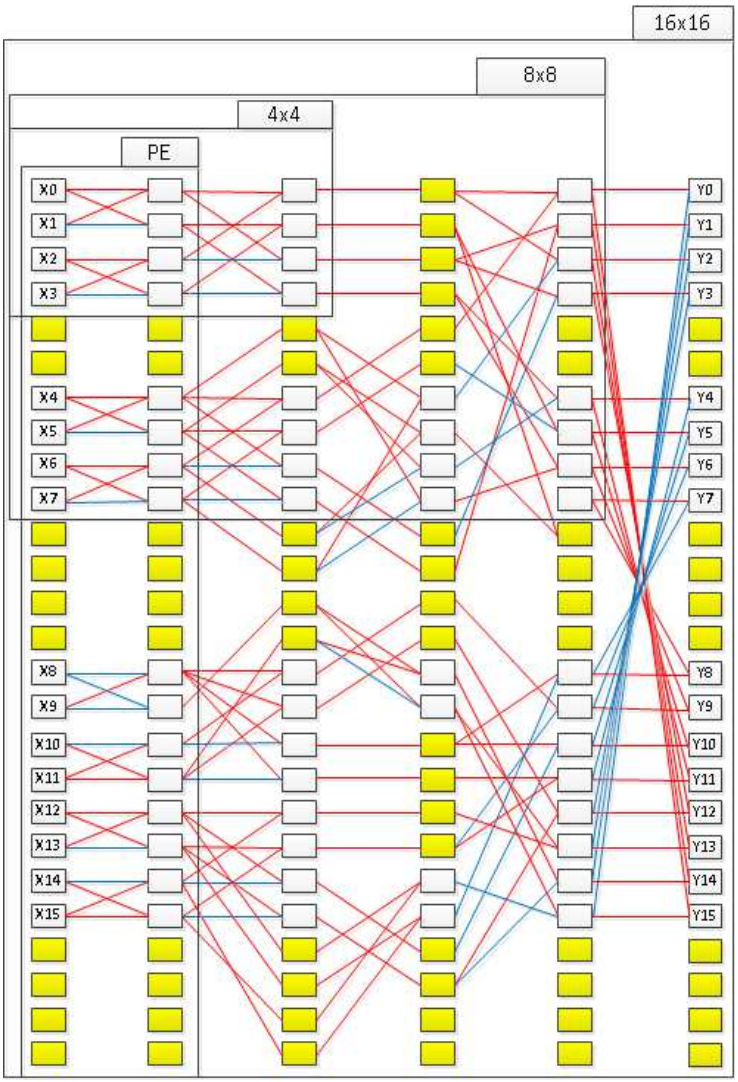

Fig. 7. Proposed architecture of HEVC core transform
그림 7. 제안하는 HEVC 코어 변환기의 아키텍쳐

구현이 가능하다[5][9]. 그림 5의 계수를 분석해보면 부호를 제외하고 4부터 90까지 28가지의 값을 가지며, 표 1과 같이 최대 3번의 덧셈/뺄셈과 최대 4번의 쉬프트 연산만으로 구현이 가능하다. 예를 들어 A*75 = A*(64+8+2+1) = (A≪6)+(A≪3)+(A≪1)+(A≪0)이므로 3번의 덧셈과 4번의 쉬프트 연산이 필요하며, A*87 = A*(64+32-8-1) = (A≪6)+(A≪5)-(A≪3)-(A≪0)이므로 1번의 덧셈, 2번의 뺄셈과 4번의 쉬프트 연산이 필요하다.

덧셈/뺄셈과 쉬프트 연산만으로 곱셈기를 대치하면서 그림 7의 데이터 흐름대로 동작하는 연산처리기의 구조는 그림 10과 같다. 그림 10의 쉬프트기는 0~6비트까지 쉬프트를 수행하지만, 실제로는 자릿수만 변경하여 와이어로 연결하는 방식으로 구현되기 때문에 MUX와 AND 이외에는 별도의 하드웨어가 필요하지 않다. 버터플라이 연산은 덧셈/뺄셈을 선택해서 수행할 수 있는 덧셈기 트리로 구현된다. 그림 10에서 점선으로 표시된 화살표는 MUX와 덧셈기를 제어하는 제어 신호를 의미한다.

연산처리기에서 2개의 입력은 다른 연산처리기의

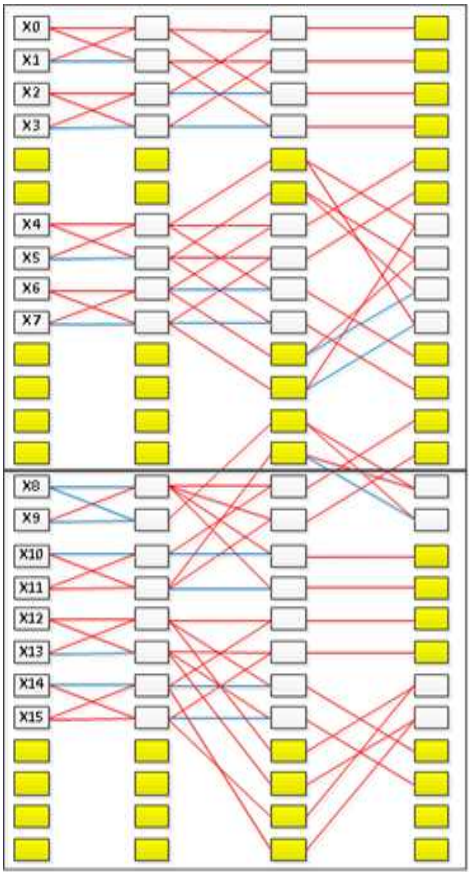

Fig. 8. 8-point inverse transform operation in the proposed architecture
그림 8. 제안하는 아키텍쳐의 8-point 역변환 동작

출력 중에서 선택된다. 그림 7에는 모두 26개의 연산처리기가 포함되어 있기 때문에 이론상으로는 26:1 MUX를 사용해야 하나, 그림 7에서 연산처리기 사이의 연결 관계를 분석해보면 연산처리기의 두 입력은 자신 또는 다른 연산처리기의 출력과 연결되는 경우가 각각 3, 5개를 넘지 않는다. 따라서 그림 10과 같이 3:1과 5:1 MUX를 사용하면 된다.

3. 게이트 수 비교

본 논문에서는 제안하는 아키텍쳐의 동작을 검증하고, 기존 아키텍쳐와 비교하기 위해서 제안하는 아키텍쳐와 기존 아키텍쳐 모두를 0.13um 공정으로 직접 설계하고 합성하였다. 비교 대상이 되는 기존 아키텍쳐는 연산처리기를 재사용하지 않고 동일한 처리 능력을 가지도록 하였고 다른 부분은 제안하는 아키텍쳐와 동일하게 설계하였다.

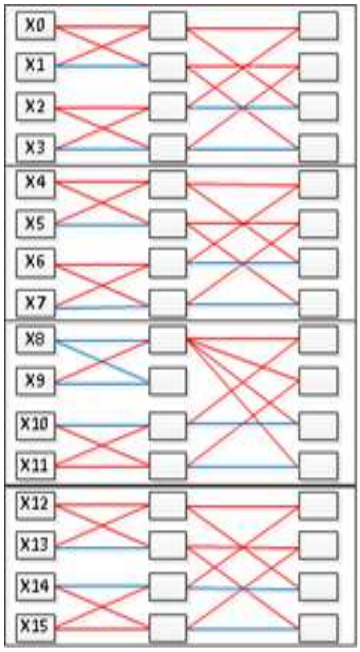

Fig. 9. 4-point inverse transform operation in the proposed architecture
그림 9. 제안하는 아키텍쳐의 4-point 역변환 동작

0.13um 공정에서 합성하고 동작을 검증해본 결과, 제안하는 아키텍쳐와 기존 아키텍쳐의 게이트 수는 각각 101,013게이트와 292,325게이트이다. 따라서 제안하는 아키텍쳐는 게이트 수를 약 1/3로 줄일 수 있음을 확인할 수 있었다.

Ⅳ. 결론

본 논문에서는 HEVC 코어 변환기를 설계하고 이를 합성하고 검증하여 동작을 확인하였다. 제안하는 코어 변환기는 면적을 많이 차지하는 곱셈기 대신에 덧셈기와 쉬프터만을 사용하였으며, 쉬프터도 실제로는 와이어 연결과 멀티플렉서만을 사용하여 면적을 크게 줄였다. 또한 하나의 하드웨어로 4×4에서 16×16 블록까지 모두 처리할 수 있도록 설계하였으며, 이를 위해서 연산처리기를 재사용하는 아키텍쳐를 제안하였다.

0.13um 공정으로 설계되고 검증된 코어 변환기는 16×16 블록을 2-D 변환 처리하는데 160 사이클이 소요되며 게이트 수는 101,015 게이트이다. 제안하는 아키텍쳐는 기존 아키텍쳐에 비해 동일한 공정, 동일한

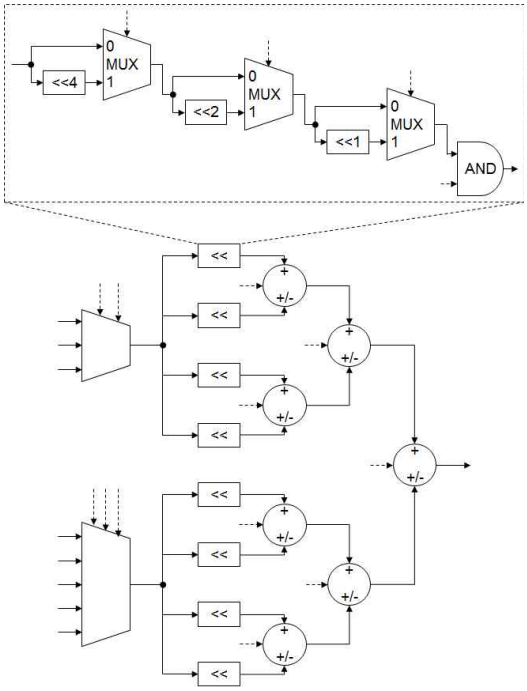

Fig. 10. Proposed processing unit of HEVC core transform

그림 10. 제안하는 HEVC 코어 변환기의 단위 처리기

처리능력에서 비교할 때 게이트 수를 1/3로 감소시켰다.

References

[1] T. Wiegand, G. Sullivan, G. Bjontgaard, and A. Luthra, “Overview of the H.264/AVC Video Coding Standard”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, Jul. 2003.

[2] J. Jung and K. Lee, “Implementation of IQ/IDCT in H.264/AVC Decoder Using GP-GPU”, Journal of IKEEE, vol. 14, no. 2, pp. 76-81, Jul. 2010.

[3] C. Lee, “Design of High Performance Dual Channel Pipelined Interpolators for H.264 Decoder”, Journal of IKEEE, vol. 13, no. 4, pp. 110-115, Dec. 2009.

[4] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[5] A. Fuldseth, G. Bjøntegaard, and M. Budagavi, “CE10: Core Transform Design for HEVC,” JCTVC-G495, Nov. 2011.

[6] J. Park, W. Nam, S. Han, and S. Lee, “High Efficiency Video Coding(HEVC) 16x16 & 32x32 Inverse Transform IP Design for Large-Scale Displays”, Proceedings of International Technical Conference on Circuits/Systems, Computers, and Communications, pp. 153-155, Jun. 2011.

[7] M. Budagavi, V. Sze, and M. Sadafale, “Hardware analysis of transform and quantization,” JCTVC-G132, Nov. 2011.

[8] M. Budagavi and V. Sze, “Unified Forward+Inverse Transform Architecture for HEVC”, Proceedings of IEEE International Conference on Image Processing, pp. 209-212, 2012.

[9] J. Park, W. Nam, S. Han, and S. Lee, “2-D Large Inverse Transform (16x16, 32x32) for HEVC (High Efficiency Video Coding)”, Journal of Semiconductor Technology and Science, vol. 12, no. 2, pp. 203-211, Jun. 2012.

BIOGRAPHY

Seung-Mok Han (Member)

2011 : BS degree in Electronic Engineering, Soongsil University.

2011~Now : MS candidate in Electronic Engineering, Soongsil University

<Main Interest> HEVC, Low-Power SoC Design, Multimedia SoC Design, Battery Management

Woo-Jin Nam (Member)

2011 : BS degree in Electronic Engineering, Soongsil University.
2011~Now : MS candidate in Electronic Engineering, Soongsil University
<Main Interest> HEVC, Low-Power SoC Design, Multimedia SoC Design, Battery Management

Seongsoo Lee (Life Member)

1991 : BS degree in Electronic Engineering, Seoul National University.
1993 : MS degree in Electronic Engineering, Seoul National University.
1998 : PhD degree in Electrical Engineering, Seoul National University.
1998~2000 : Research Associate, University of Tokyo
2000~2002 : Research Professor, Ewha Womans University
2002~Now : Associate Professor in School of Electronic Engineering, Soongsil University
<Main Interest> HEVC, Low-Power SoC Design, Multimedia SoC Design, Battery Management