FPGA 상에서 에너지 효율적인 DCT (Discrete Cosine Transform) 모듈 설계 및 구현

장 주 욱[†] · 임 창 현[††] · Ronald Scrofano[†††] · Viktor K. Prasanna[††††]

요 약

불연속 코사인 변환(DCT)은 비디오와 영상 처리의 필수적인 부분이며, JPEG 및 MPEG 표준에서 채택하고 있다. 특히, 모바일 장치에서 비디오를 스트리밍 재생할 때에는 에너지 효율적인 DCT 연산이 매우 중요하다. 본 논문에서는 선형 어레이 PE를 이용한 DCT 연산기 구조를 제안한다. 본 제안은 에너지 효율을 최적화하도록 설계되어 있다. 설계 효율을 보이기 위해 에너지 사용, 면적, 지연 간의 트레이드오프(trade-off)를 분석하고, 그 성능을 Xilinx의 최적화된 IP 코어와 비교하였다.

키워드 : DCT, 에너지 효율, 성능 모델링

Energy-Efficient Discrete Cosine Transform on FPGAs

Ju-wook Jang[†] · Chang-hyeon Lim[††] · Ronald Scrofano[†††] · Viktor K. Prasanna[††††]

ABSTRACT

The 2-D discrete cosine transform (DCT) is an integral part of video and image processing; it is used in both the JPEG and MPEG enciding standards. As streaming video is brought to mobile devices, it becomes important that it is possible to calculate the DCT in an energy-efficient manner. In this paper, we present a new algorithm the DCT with a linear array PEs. This design is optimized for energy efficiency. We analyze the energy, area, and latency tradeoffs available with this design and then compare its energy dissipation, area, and latency to those of Xilinx's optimized IP core.

Key Words : DCT, Energy Efficiency, Performance Modeling

1. 서 론

무선망 업계는 모바일 장치에 비디오 스트리밍 기능을 부가하는 방향으로 진화하고 있다. 이러한 기능 확장 때문에 모바일 장치는 영상처리를 위한 추가적인 전력 소모 부담이 있다. 뿐만 아니라, 배터리 용량이 제한되어 있어 에너지 제한적인 환경이라고도 볼 수 있다. 영상처리를 함에 있어 높은 효율과 낮은 지연시간뿐만 아니라, 에너지 효율까지 고려하는 것이 필요하다.

FPGA는 비디오 스트리밍 과정에서 영상처리에 사용 가능하며, 그 일반적인 구조가 영상처리 분야에 적합하다. 현재 FPGA 공급자들이 곱셈기나 곱셈-덧셈기 등의 하드 IP가 갖는 낮은 지연시간과 높은 성능 등의 장점을 살리기 위해 이들을 포함시키고 있어, FPGA를 영상처리 부분에 사용할 수 있게 되었다[6][7]. 하지만, 앞서 언급한 바와 같이, 영상처리에서 에너지 효율은 매우 중요한 문제가 되었다.

FPGA는 에너지 제한적인 환경에 필요한 계산 기능을 제공할 수 있기 때문에, FPGA를 에너지 효율적으로 설계하는 분야가 최근 새롭게 각광받고 있다. 하지만, 현재에도 여전히 저전력 사양을 가진 수백만 게이트의 상업적 FPGA는 없기 때문에, 본 논문에서 에너지 효율적인 알고리즘과 구조를 설계하고자 한다. 제안하는 방식은 차세대 FPGA가 가진 저전력 사양을 별다른 부가 기능 없이 활용할 것으로 본다.

본 논문에서는, 2차원 DCT를 위한 에너지 효율적인 알고리즘과 구조를 제안한다. 2차원 DCT는 정지영상을 위한 JPEG뿐 아니라 스트리밍 비디오 분야의 표준인 MPEG과 H.263에서도 사용되는 중요한 핵심 기능이다[1]. 여기서는 지연 시간, 에너지 그리고 면적를 예측하고 그 성능을 Xilinx

※ 본 논문은 정보통신부의 출연금으로 수행한 IT SoC 핵심설계인력양성사업의 수행결과임.
† 정 회 원 : 서강대학교 전자공학과 교수
†† 준 회 원 : 서강대학교 전자공학과 박사과정
††† 비 회 원 : Dep. of Electrical Engineering University of Southern California, Ph.D Student
†††† 비 회 원 : Dep. of Electrical Engineering University of Southern California, Professor
논문접수 : 2005년 3월 4일, 심사완료 : 2005년 6월 13일

IP코어의 8×8 DCT블록과 비교하여, 제안된 방식이 더 효율적임을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 정리하고, 3장에서는 2차원 DCT를 위한 새로운 알고리즘 및 구조를 기술한다. 4장에서는 본 설계에서 사용된 최적화 기법을 설명하고, 5장에서는 Domain-specific 모델을 보인다. 6장에서는 DCT 연산에 있어 가능한 에너지, 면적, 그리고 지연 시간의 트레이드-오프(trade-off)를 분석한다. 7장에서는 Xilinx IP 코어와 비교하며, 끝으로 9장에서는 결론 및 추후과제를 기술한다.

2. 관련 연구

본 연구와 같이 FPGA를 위한 저전력 알고리즘 설계와 DCT 연구를 통합하는 관점의 연구는 없었지만, 두 가지 분야의 연구는 꾸준히 진행되어 왔다.

[4]에서는 본 디자인과는 다른 2차원 매시 구조를 위한 시스톨릭 어레이(systolic array) 구조와 알고리즘을 보이고 있다. 본 연구에서는 2차원 매시보다는 선형 어레이를 사용하였다. FPGA 상에서 상호연결은 많은 에너지를 낭비할 수 있기 때문에, 디자인에 필요한 상호연결의 수를 줄여야 한다.

라우팅 스위치의 개선을 통한 전력 소모를 줄이기 위해 고속, 저전력, 대기상태를 두고 고속 동작은 기존의 방식을 따르고, 저전력 소모를 위해 속도를 줄이거나 대기상태로 전환하는 방법도 가능하다. 이를 위해 기존 배선 스위치를 개선해 기존 FPGA 내부연결과 연동이 가능하게 한다[8].

[1]과 [5]에서는 각각 FPGA 상에서 DCT의 구현에 관하여 설명하고 있다. 각 기법은 본 연구에서 사용된 기법과는 다르다. [1]은 다항 변환(polynomial transform)을 사용한 DCT 연산을 설명하고 있으며, 연산을 위한 기법과 FPGA 상의 데이터 경로를 기술하고 있다. 또한 이 연구에서는 배경연구로 분산 산술 기법(distributed arithmetic technique)을 설명하고 있는데, 본 논문에서는 결과에서 비교할 Xilinx IP 코어로 이것을 채택하였다. 이 기법은 FIR 필터를 이용하여 행 단위의 1차원 DCT를 수행하고, 그 결과를 이용하여 다시 열 단위의 1차원 DCT를 수행하며, 그 사이에 메모리 교차(transpose)가 필요하다. 본 연구는 [1]과 [13]에서 보인 기법과는 아주 다르며, 비트 레벨의 입력 데이터 회로나, 메모리 교차가 필요하지 않다. 뿐만 아니라, 다항 변환 DCT 연산도 따르지 않는다. 대신, $n \times n$ 행렬의 2차원 DCT 계산을 위한 n개의 PE를 사용한 모듈러(modular) 디자인 기법을 채택한다((그림 1) 참조). 이 기법은 두 $n \times n$ 행렬의 곱셈을 수행하는 것과 유사한 과정을 갖는다.

[2]와 [3]은 FPGA를 위한 에너지 효율적인 디자인과 성능 모델 기법을 기술하고 있다. [2]는 FPGA 상에서 에너지 낭비 요소를 분석하고 에너지 효율을 높이기 위한 알고리즘 및 구조에 사용될 수 있는 기법을 설명하고 있으며, 2차원 DCT를 위한 알고리즘 및 구조에 사용한다(4장 참조). [3]은 FPGA 상으로 매핑할 때 성능 예측을 위한 도메인-스페시픽(domain-specific) 모델링 기법을 설명하고 있다. 본 연구에서는 이 기법을 2차원 DCT 디자인의 성능 예측에 사용한다(5장 참조).

3. 2차원 DCT를 위한 에너지 효율적인 알고리즘 및 구조

n개의 요소를 가진 벡터 B의 1차원 DCT는 아래와 같다.

$$Y_k = a_k \sum_{i=0}^{n-1} b_i cos\left(\frac{2\pi}{4n}(2i+1)k\right), k = 0, ..., n-1 \quad (식\ 1)$$

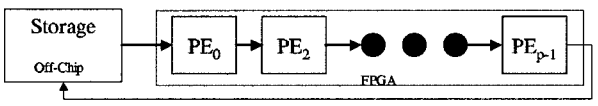
이 때, b_i는 B의 n개의 요소들을 가리키며, $k=0$에서는 $a_k = \frac{1}{\sqrt{n}}$, 그 밖에는 $\sqrt{\frac{2}{n}}$이다.

X는 $n \times n$ 행렬인 입력이다. 전통적으로, 2차원 DCT는 X의 행들에 대하여 1차원 DCT를 수행하고 나서 그 결과들의 열에 대하여 1차원 DCT를 수행한 결과, 즉 행-열 기법에 의해 계산한다. 이 기법은 (식 3)에서 계수 행렬로 정의된 C를 이용한 결과인 Z를 DCT의 결과로 하는 것과 같음을 알 수 있다. C에서는 8개의 서로 다른 계수의 면적만이 중요하다. 즉, 부호를 무시한다면, 오직 8개의 서로 다른 값들만이 계수 행렬에 저장된다는 의미이다. 어떤 행에서 첫 번째 열의 엔트리와 마지막 열의 엔트리의 면적는 같다. 두 번째 열과 마지막에서 두 번째 열의 면적도 같다. 이러한 방식으로 중앙에 위치한 열로 진행한다. (식 2)과 (식 3)에 이런 성질을 표시하였다.

$$Z = CXC^T \quad (식\ 2)$$

$$C = a_k \begin{bmatrix} C_0 & C_0 & C_0 & C_0 & C_0 & C_0 & C_0 & C_0 \\ C_1 & C_3 & C_5 & C_7 & -C_7 & -C_5 & -C_3 & -C_1 \\ C_2 & C_6 & -C_6 & -C_2 & -C_2 & -C_6 & C_6 & C_2 \\ C_3 & -C_7 & -C_1 & -C_5 & C_5 & C_1 & C_7 & -C_3 \\ C_4 & -C_4 & -C_4 & C_4 & C_4 & -C_4 & -C_4 & C_4 \\ C_5 & -C_1 & C_7 & C_3 & -C_3 & -C_7 & C_1 & -C_5 \\ C_6 & -C_2 & C_2 & -C_2 & -C_2 & C_2 & -C_2 & C_6 \\ C_7 & -C_5 & C_3 & -C_1 & C_1 & -C_3 & C_5 & -C_7 \end{bmatrix} \quad (식\ 3)$$

본 디자인에서, Z를 찾기 위해 PE의 선형 어레이를 사용한다. 입력 데이터는 off-chip 메모리 또는 on-chip 메모리에 저장될 수 있다. 본 논문에서는 입력 또는 출력을 저장할 메모리는 고려하지 않으며, 중간 결과를 저장하고 읽는 비용만을 고려한다. (그림 1)에서는 off-chip 선형 어레이의 다이어그램을 보이고 있다. PE는 (그림 2)에 표시하고 있다. DCT는 두 단계로 계산된다, 첫 번째 단계는 $D = XC^T$이고 두 번째 단계는 $Z = CD$이다.



(그림 1) off-chip 메모리에 저장되는 데이터를 처리하는 선형 구조

3.1 1단계

알고리즘의 1단계에서는 X의 엔트리를 수정된 행 우선 순서(modified row-major order)로 정렬하여 입력으로 한다. 예를 들면, 어떤 8×8 DCT에서 X의 두 번째 행과 C^T의 두 번째 열의 입력은 아래 표와 같다. 이 입력을 가지고 구하고자 하는 D_{11}은 아래와 같다.

$$D_{11} = C_1x_{10} + C_3x_{11} + C_5x_{12} + C_7x_{13} + (-C_7)x_{14} + (-C_5)x_{15} + (-C_3)x_{16} + (-C_1)x_{17} \quad \text{(식 4)}$$

〈표 1〉 제안된 입력 행렬 원소의 입력 순서

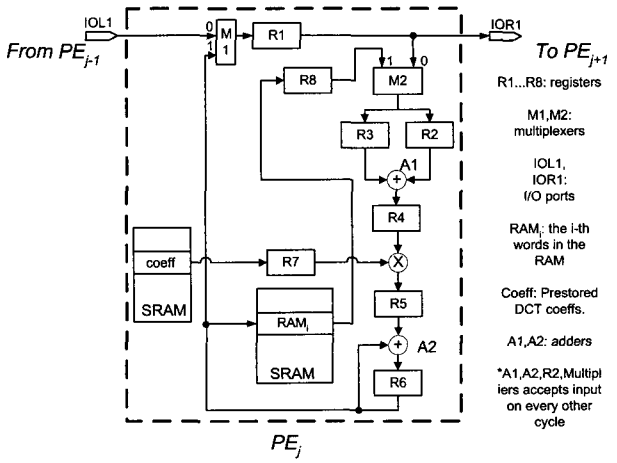
입력 순서	1	2	3	4	5	6	7	8
X의 순서	x_{10}	x_{17}	x_{11}	x_{16}	x_{12}	x_{15}	x_{13}	x_{14}
입력에 곱해지는 계수	C_1	$-C_1$	C_3	$-C_3$	C_5	$-C_5$	C_7	$-C_7$

위 표에서 보듯, 입력 순서는 계수의 절대값이 같은 것이 이어서 올 수 있도록 배치한다. 즉, C_1x_{10}, $(-C_1)x_{17}$을 차례로 계산하여 더하는 대신, 먼저 $x_{10}-x_{17}$을 계산하고 그 값에 C_1을 곱해 $(x_{10}-x_{17})C_1$을 계산하면 한 번의 곱셈을 줄일 수 있다. 나머지 입력에 대해서도 같은 방식으로 수행하면 (식 5)와 같이 4번의 곱셈만으로 D_{11}을 구할 수 있다. 결과적으로 곱셈 횟수를 1/2로 줄일 수 있다.

$$D_{11} = (x_{10}-x_{17})C_1 + (x_{11}-x_{16})C_3 + (x_{12}-x_{15})C_5 + (x_{13}-x_{14})C_7 \quad \text{(식 5)}$$

결과적으로 이 기법은 계수 메모리를 엑세스하는 횟수와 곱셈의 횟수를 1/2로 줄여준다. (그림 2)에서 PE의 선형 어레이는 이러한 방식으로 $D=XC^T$를 수행한다.

각각의 PE_j는 매개 행렬(intermediate matrix) D의 j번째 열을 계산한다. 각 PE의 계수 메모리는 C의 계수들의 면적만을 저장한다. 1단계 동안은 곱셈기 M1과 M2 모두 입력



(그림 2) DCT 연산을 위한 어레이의 PE_j

라인 0을 선택한다. X로부터 입력이 IOL1을 통해 PE로 들어오면, 매 클록 싸이클마다 R1에 저장된다. R1의 데이터는 IOR1을 통해 다음 PE로 전달된다. 부가적으로, R1에 저장된 후에 데이터는 별도로 레지스터 R2와 R3에 기록된다(처음은 R2, 다음은 R3, 그 다음은 R2 등). 매 다른 클록 싸이클마다, R2와 R3의 데이터는 조건에 따라서 음수로 변하였다가 서로 더해져서 R4에 저장된다. R4에 저장된 다음 클록에서는, R4의 데이터를 계수 메모리의 해당 계수와 곱한다 (이 계수는 R4에 데이터가 저장된 것과 같은 클록 싸이클 시점에 R7에 저장된다). 이 곱셈의 결과는 R5에 저장된다. R6은 중간 결과들이 완전히 저장될 때까지 누적하는데 사용한다. 저장이 끝나면 R6의 데이터는 메모리로 옮겨지며 R6의 내용은 지워진다. 이런 방식으로 매개 행렬 D의 모든 열의 계산이 끝나면, 알고리즘은 2단계로 전환한다.

3.2 2단계

2단계에서는 2차원 DCT 연산을 완료하기 위해 $Z=CD$의 곱셈을 계산한다. 이 단계에서는 1단계와 거의 똑같은 과정을 수행한다.

이 단계에서 각 PE는, 외부로부터 입력을 받지 않고, 계산 과정에서 데이터 램에 저장된 중간 데이터를 이용한다. 그리하여, M2는 입력 라인 1을 선택한다. R6에 누적된 데이터가 데이터 램에는 저장되지 않고 R1에 기록된다. 리셋 싸이클 동안에는 마지막 PE를 통하여 어레이 내의 모든 결과 엔트리가 출력되는 경로를 설정하기 위해, M1은 입력 0을 선택한다. 데이터는 행 순서의 역방향으로 어레이를 빠져나온다. 예를 들면, 8×8 DCT에서, 선형 어레이의 첫 번째 출력은 z_{07}이고, 그 다음에 z_{06}, 그리고 마지막으로 z_{70}이 출력된다. 이 과정은 Z의 모든 n^2개의 엔트리가 나올 때까지 계속된다. 이 때, PE는 1단계로 돌아가서 다른 DCT를 계산할 수 있다.

4. 에너지 성능의 최적화

본 디자인의 에너지 성능을 최적화하기 위해, 아키텍처 선택(architecture selection) 기법을 사용한다. FPGA에서는 긴 상호 연결은 전력소모가 크며, 아키텍처에 따라 서로 다른 에너지 성능과, 지연시간과, 성능 등을 보인다. 본 DCT 설계에서는, 처리 요소(processing element, PE)를 선형 어레이로 사용하는 구조를 선택하였다. 선형 어레이의 PE는 오직 이웃한 PE하고만 통신을 하므로 긴 와이어의 사용을 줄일 수 있어 상호 연결에 필요한 면적을 줄일 뿐 아니라 에너지 효율이 높다.

본 논문에서 사용한 또 다른 기법은 적절한 바인딩 선택(choosing the appropriate bindings)이다. 여기서는, 계수 및 데이터 저장을 위한 분산 메모리와, 연산기로 임베디드 곱셈기를 선택했다.

마지막으로, 알고리즘 수준에서 전체 에너지 소비의 상당 부분을 소비하는 핫-콤포넌트(hot component)의 엑세스 횟

수를 줄이는 기법이다. 본 연구에서는 곱셈기, 메모리, 그리고 레지스터가 핫-콤포넌트들이며, 이들 각 콤포넌트의 엑세스 횟수를 줄였다. 예를 들면, 곱셈 양이 50% 감소했으며, 계수 메모리 읽기 엑세스 횟수는 1/3만큼 줄어들었다.

5. 도메인-스페시픽 에너지 모델

본 설계를 제작하기 위해, 도메인-스페시픽(Domain-specific) 모델링을 채택했다(자세한 기술은 [3] 참조). 이 모델링은 성능 모델에 대한 하이브리드 (top-down과 bottom-up 방식의 조합) 접근방식이며, 기준에 가장 부합하는 설계를 결정하기 위해 관련 알고리즘과 아키텍처를 빠르게 평가할 수 있다. 이 모델에서 아키텍쳐는 재배치 가능 모듈(Relocatable modules, RModule)과 상호 연결로 나뉜다.

본 설계에서는, 곱셈기, 덧셈기, 멀티플렉서(multiplexor), 램, 그리고 레지스터가 RModule이다. 본 연구에서는 PE의 선형 어레이를 사용하기 때문에, 가장 가까운 RModule끼리 연결을 갖는다는 것을 가정하며, 그 에너지 소비는 RModule에서 소비되는 양에 비해 매우 적을 것이라고 가정한다.

아래에서 설명하는 모델에서는 $n \times n$ DCT가 $p \le nPEs$를 이용하여 계산한다. n은 p로 나누어진다고 가정한다. 각 PE는 매개 및 최종 행렬의 n/p 개의 열을 계산한다.

5.1 지연 시간 예측

지연에 대한 방정식은 각 알고리즘과 PE의 단계를 분석함으로서 세울 수 있다. 첫 번째 단계에서 첫 번째 데이터를 R6에 저장하기 위해서는 6 싸이클이 필요하다. 매 중간과정 결과마다 $n/2$번 R6에 기록한다. 기록 후에 R6는 한 싸이클만큼 쉬고 그 다음 싸이클에 기록 한다. 그러므로 R6에 첫 번째 기록을 마치고 데이터 SRAM에 기록하기 위해서는 6+ $n/2$(2)-1=n+5 싸이클이 필요하다. 각 PE는 그것이 계산하는 중간과정 행렬 n/p열의 각각의 n 엔트리를 계산한다. 그러나 첫 번째 엔트리를 연산할 때를 제외한 모든 경우에는 레지스터의 파이프라인이 채워져 있으므로, 엔트리를 연산하고 RAM에 기록하는데 $(n/2)(2)$ 싸이클만이 소요된다. 그러므로 단계 1에서 PE로 첫 번째 중간과정 열을 계산하기 위한 하나의 엔트리는 n+5 싸이클을 소요한다. 그에 반해 그 열의 다른 n-1개의 엔트리들은 $(n/2)(2)$ 싸이클만 계산을 수행한다. 그러므로 단계 1의 첫 번째 중간과정 열을 계산하는 지연시간은,

$$L_{p1c1} = (n+5) + (2 \times (n/2) \times (n-1)) \qquad \text{(식 6)}$$

이 되고, 간단히 하면,

$$L_{p1c1} = n^2 + 5 \qquad \text{(식 7)}$$

가 된다.

p <n 일때 연산할 수 있는 알고리즘은 두 가지가 있다. 첫 번째는 단계 1의 모든 중간과정 결과를 계산하고 그것을 저장하는 방식이다. 이 방식은 non-alternating mode라고 언급할 것이며 이 방식의 지연은 Lp1nonalt라고 표시할 것이다. 나머지 다른 방식은 각각의 PE가 단계 1의 하나의 중간과정 열을 계산한 다음 단계 2의 결과 열을 계산하고 그런 다음 또 다른 단계 1의 중간과정 열을 계산하고 단계 2의 결과 열을 계산기를 반복하는 방식이다. 이 방식은 alternating mode라고 언급할 것이며 이 방식의 지연은 Lp1alt라고 표시할 것이다. non-alternating mode 에서는 선형 어레이의 파이프라인이 가득 찬 상태로 유지됨으로써 단계 1의 지연은

$$L_{p2alt} = \frac{n}{p}(n^2) + 5 \qquad \text{(식 8)}$$

이 된다.

한편, alternating mode의 경우는 단계 1을 시작할 때마다 파이프라인은 비어있게 되어 단계 1의 종합적인 지연은

$$L_{p1alt} = \frac{n}{p}(n^2 + 5) \qquad \text{(식 9)}$$

이 된다.

단계 2는 입력이 변환된 행렬이 아니라 중간과정의 결과라는 점을 제외하고는 단계 1과 같은 과정이다. 따라서 단계 1에서의 모든 중간과정 결과가 계산되었을 때, 단계 2의 지연은 (식 8)과 같으며, 그리고 단계 1과 단계 2가 alternating mode일 때 단계 2의 종합적 지연은 (식 9)과 같다.

어떠한 방식이던지 단계 1의 계산은 단계 2의 계산과 겹치지 않는다. 따라서 DCT의 계산에서 한 PE의 지연은 Lp1과 Lp2의 합과 같아진다. 선형 어레이에서 마지막 PE는 첫 번째 PE 보다 p-1 싸이클 뒤에 오므로 총 지연에 p-1 만큼이 더해진다. 이러한 딜레이를 지나서 첫 번째 PE가 어레이의 끝에서 출력되기까지 또 다른 p-1 싸이클의 지연이 존재한다. 그래서 두 번째 p-1 딜레이가 총 지연에 더해져야한다. non-alternating mode의 경우에는 이러한 딜레이가 오직 한번만 발생하며 alternating mode에서는 이러한 지연이 매 단계 1-단계 2 마다 발생한다. 그러므로 총 지연은

$$L_{noalt} = \left[\frac{n}{p}n^2 + 5\right] + \left[\frac{n}{p}n^2 + 5\right] + 2(p-1) \qquad \text{(식 10)}$$

$$L_{alt} = 2 \times \frac{n}{p}(n^2 + 5) + 2 \times \frac{n}{p}(p-1) \qquad \text{(식 11)}$$

간단히 하면

$$L_{noalt} = 2\left(\frac{n^3}{p}\right) + 2p + 8 \qquad \text{(식 12)}$$

$$L_{alt} = \frac{n}{p}(2n^2 + p + 8) \quad \text{(식 13)}$$

이 된다.

5.2 에너지 예측

본 설계에서 에너지 소비를 예측하기 위해서, 각 PE 콤포넌트의 동작을 결정하는 알고리즘을 분석한다. 이 분석을 통해 얻는 각 콤포넌트가 활성화 시간과 각 콤포넌트의 전력 소모량을 곱하면 에너지 소모량 예측이 가능하다. 콤포넌트 각각의 유형 별 전력 소모는 저수준 시뮬레이션을 통해 얻을 수 있다. SRAM, 곱셈기, 그리고 덧셈기는 입력에 레지스터를 사용하지 않으며 출력에 레지스터를 사용한다. (그림 3)에서는 에너지 예측에서 제외한 레지스터들을 점선으로 표시하였다.

계산 결과는 각 콤포넌트가 alternating 기법인가 또는 non-alternating 기법인가에 관계없이 같은 값을 보이지만, 메모리의 전력 소모에서 차이를 보인다. Non-alternating 기법의 경우에 alternating 기법에 비해서 각각의 PE가 많은 양의 메모리를 필요로 하므로, 큰 메모리로 인한 더 많은 전력 소모를 하게 된다.

5.2.1 1단계

본 알고리즘의 1단계에서는 매 2 싸이클마다 덧셈기, 곱셈기, 레지스터 R5, 그리고 계수 RAM을 엑세스한다. 모두 n^2의 입력이 각 PE에서 나오는 매개 열에 대해 동작한다. 그래서 이 콤포넌트들 각각은 매개 열마다 $n^2/2$번 활성화된다. 생성되는 전체 매개 열의 개수는, PE의 숫자와는 무관하게, 전체 매개 행렬이 n개의 열을 갖기 때문에 n이 된다. 그래서 덧셈기, 곱셈기, 레지스터 R5, 그리고 계수 RAM은 $n^2/2$싸이클의 모든 1단계에 있는 모든 PE에 대하여 각각 활성화된다.

한 PE가 매개열을 계산할 때마다, 원래 행렬에서 n^2개의 모든 엔트리를 읽어야 한다. 그러므로, 소스 행렬을 PE_0으로 n/p번 입력해야 한다. 따라서, 각 PE에서 M1과 M2은 $(n/p)n^2$ 싸이클 동안 활성화되는 것이다. p개 이상의 PE일 때는 각 곱셈기는 n^3싸이클 동안 활동한다.

매개 행렬의 모든 엔트리는 반드시 데이터 SRAM에 기록된다. SRAM에 기록되는 총 횟수는 PE의 개수와 무관하게 n^2이다.

이 정보들을 하나의 에너지 관계식으로 정리하면 아래와 같다.

$$E_1 = n^3\left[\frac{1}{2}(2P_{add} + P_{mult} + P_{reg} + P_{cRAM1p}) + 2P_{mux} + \frac{1}{p}(P_I)\right] + n^2 P_{dRAM1p} \quad \text{(식 14)}$$

P_{add}, P_{mult}, P_{mux}, P_{reg}, P_{cRAM1p}, P_{dRAM1p}, 그리고 P_I는 각각 덧셈기/뺄셈기, 곱셈기, 멀티플렉서, 레지스터, 데이터 저장용 1포트 RAM, 그리고 입력 8bit 데이터의 전력 상수들이다.

5.2.2 2단계

2단계에서는 덧셈기, 곱셈기, 레지스터 R5, 계수 SRAM, 그리고 멀티플렉서 M2의 동작이 1단계에서와 같다. 데이터 SRAM에 기록하지는 않지만 대신 매 싸이클마다 한 요소가 읽혀진다. PE의 숫자와 무관하게, n^3번의 데이터 SRAM 읽기를 수행해야 한다. 각 결과 엔트리는 n번 읽기를 해야 하며, 총 n^2개의 엔트리가 있다. 각 PE에 있는 데이터 SRAM의 면적와 그로 인한 전력 소모는 PE의 개수와 alternating 기법을 사용하는지 여부에 달려 있지만, 전체적으로 읽기 횟수는 같다.

이 단계에서 PE_j의 멀티플렉서 M1은, 어레이로부터의 출력처럼 PE_{j-1}로부터 출력이 PE_j로 전달될 때와 PE_j의 R6의 데이터가 PE_{j+1}로 기록될 때 활성화된다. 그리하여 멀티플렉서 M1은 자체 출력과, 그 전에 모든 PE의 모든 출력을 선택할 때 PE_j에서 활성화된다. 따라서 PE_j 내의 M1은 계산된 출력의 각 열에 대해서 $n(j+1)$ 싸이클만큼 활성화된다. 각 PE는 출력의 n/p개의 열을 계산하고, 멀티플렉서 M1이 활성화되는 2단계 전체에 걸쳐서 모든 PE의 총합은 $n\left(\frac{n}{p}\right)\left(\frac{p(P+1)}{2}\right)$싸이클이다.

부가적으로, 전체 n^2 개의 요소들이 어레이의 출력이라는 사실을 고려하여 2단계에서 총 에너지 소모는 다음과 같이 예측한다.

$$E_2 = n^3\left[\frac{1}{2}(2P_{add} + P_{mult} + P_{reg} + P_{cRAM1p}) + P_{mux} + P_{dRam1p}\right] + n^2 P_O + \left(\frac{n^2(p+1)}{2}\right)P_{mux} \quad \text{(식 15)}$$

P_O는 8비트 데이터 출력에 대한 전력이고, 다른 상수들은 앞서 기술한 바와 같다.

5.2.3 총합

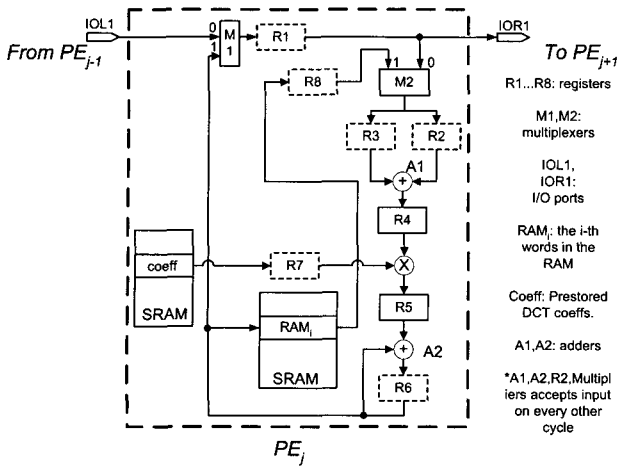
E_1와 E_2를 합치면 에너지 소모량이 된다. 전체 에너지 소모량을 구하려면, 비활성 에너지 소비량을 반드시 더해야 한다. 그러면 전체 에너지 소모 총합은 다음과 같다.

$$E = E_1 + E_2 + LP_Q \quad \text{(식 16)}$$

L은 (식 12) 또는 (식 13)에서 구하며, P_Q는 장치의 비활성 전력 소모량이다.

5.3 면적 예측

(그림 3)의 점선 표시로 된 콤포넌트들의 면적는 다른 콤



(그림 3) DCT 연산을 위한 어레이의 PE_j. 점선으로 표시된 레지스터는 다른 콤포넌트의 일부

포넌트들의 면적으로 둘러싸여진다. 그러면, PE의 면적는 하나의 레지스터, 적절한 면적의 2 포트 RAM들 두 개의 2-to-1 멀티플렉서, 두 개의 덧셈기, 그리고 한 개의 곱셈기로 구성된다. p개의 PE의 예측된 면적는 다음과 같다.

$$A = p(A_{reg} + A_{cRAMp} + A_{dRAMp} + 2A_{mux} + 2A_{add} + A_{mult}) \quad \text{(식 17)}$$

이 때 A_{reg}, A_{dRAMp}, A_{cRAMp}, A_{mux}, A_{add}, 그리고 A_{mult}는 각각 레지스터, 데이터 저장을 위한 1포트 RAM, 계수 저장을 위한 1포트 RAM, 멀티플렉서, 덧셈/뺄셈기, 그리고 곱셈기의 면적을 가리킨다.

6. 트레이드오프

이번 장에서 이 디자인에서 에너지, 면적, 지연시간간의 트레이드오프(tradeoff)를 분석하고자 한다. 비교를 위해 가장 일반적인 면적의 DCT인 8×8 DCT를 사용한다. 이것은 JPEG 과 MPEG에서 많이 사용된다. 8×8 DCT 일 때 선형 어레이에서 가능한 PE의 개수는 1, 2, 4, 8이다.

디자인을 분석하기위해 V 장에서 유도된 방정식을 사용한다. 지연 방정식(latency equation)(식 12, 13)을 이용하면 같은 수의 PE일 때 non-alternating mode 의 알고리즘이 더 낮은 지연을 보인다는 것은 쉽게 알아낼 수 있다. 그러나 non-alternating mode는 alternating mode 보다 각 PE의 저장횟수가 많아진다. 그 이유는 non-alternating mode 에서는 PE에 의해 처리된 모든 단계 1 결과가 그 PE에 저장되어야 하지만 alternating mode에서는 언제든지 n개 중간 결과만이 PE에 저장된다. 그러므로 non-alternating mode는 $n(n/p)$개의 엔트리들을 저장할 수 있는 메모리가 필요하다. 반면에 alternating mode에서 PE는 단지 n 엔트리의 메모리만이 필요하다.

〈표 2〉 에너지 예측 방정식을 위한 상수값

Constant	Size	Power(mW)
P_{reg}	8 bits	1.41
$P_{RAMp-16}$	width=8 bits, depth=16	1.85
$P_{RAMp-32}$	width=8 bits, depth=32	5.58
$P_{RAMp-64}$	width=8 bits, depth=64	6.45
P_{mux}	8 bits	1.85
P_{add}	8 bits	1.85
P_{mult}	8 bits	10.56
P_I	8 bits	0.63
P_O	8 bits	12.50
P_Q	N/A	150

〈표 3〉 면적 예측 방정식을 위한 상수값

Constant	Size	Area(slices)
A_{reg}	8 bits	4
$A_{RAMp-16}$	width=8 bits, depth=16	4
$A_{RAMp-32}$	width=8 bits, depth=32	16
$A_{RAMp-64}$	width=8 bits, depth=64	16
A_{mux}	8 bits	4
A_{add}	8 bits	4
A_{mult}	8 bits	16

에너지, 면적, 지연시간의 차이가 어떤 영향을 가져오는지 분석하기위해 low-level stimulation 으로부터 이끌어낸 상수와 도메인-스페시픽 모델을 이용할 것이다. <표 2>는 100MHz의 Xilinx Virtex-II에서 매핑했을 때, 디자인의 콤포넌트들의 손실값을 보여준다. 주의할 점은 다른 몇몇 디자인들은 8 메모리 엔트리만을 요구하지만 Xilinx Virtex-II에서는 16 메모리 엔트리의 최소 면적 메모리를 요구한다. <표 3>은 Virtex-II[2]가 충족될 때 본 디자인의 콤포넌트들의 면적를 표로 나타낸다.

〈표 4〉 에너지, 면적, 지연, 그리고 EAT(Non-alternating mode의 경우, EAT는 8 PE에 대해 normalization하였다.)

No. PEs	Nonalternating Mode				Alternating Mode			
	E (nJ)	A(slices)	T (us)	EAT	E (nJ)	A(slices)	T (us)	EAT
8	370.51	352	1.52	1.00	370.51	352	1.52	1.00
4	548.54	176	2.72	1.32	572.54	176	2.88	1.46
2	947.65	112	5.24	2.81	980.16	88	5.60	2.44
1	1718.68	56	10.34	5.02	1797.18	44	11.04	4.40

<표 4>는 두가지 방식 모두에서 1, 2, 4, 8 PE들로 디자인했을 때 에너지, 면적, 지연시간의 값을 나타내었다. 게다가 이것은 에너지, 면적, 지연시간을 non-alternating mode 경우에서 8 PE 인 경우를 기준으로 두고 각각의 경우 에너지(energy)와 면적(area), 시간(time)을 곱한값(EAT)을 나타낸다. 이때 EAT의 경우, 낮은 값이 좋은 것이다. 이 표는 8 PE 디자인이 가장 빠르고 가장 에너지 효율적이며, 또한 가장 큰 면적에도 불구하고 가장 낮은 EAT를 가진다는 것을 보여준다.

7. 성능 비교

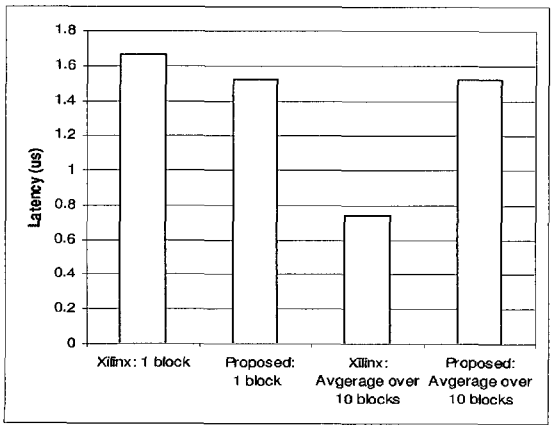
7.1 실험 환경

이제 우리의 디자인과 Xilinx IP 코어 for 2D DCT의 경우에 지연시간, 에너지 손실, 면적측면을 고수준에서 예측하여 비교해 볼 것이다. 각 디자인에서 8-bit precision과 100MHz clock 주파수를 가정한다. 우리의 디자인인 경우 VI 장 표의 값과 방정식을 이용한다. Xilinx IP 코어의 경우 값을 비교하기 위해 그 디자인을 설정하고 시뮬레이션한다. 먼저 Xilinx 코어Generator(특별한 언급이 없으면 모든 tool은 Xilinx ISE 4.2i를 기준으로 한다.)를 이용한다. Xilinx XPower를 통하여 디자인에 의한 전력 손실을 분석한다. 한편, DSP와의 비교를 위하여, TMS320C6415에 대한 결과값은 Texas Instruments의 Code Composer Studio 2.1을 통해 얻는다[9].

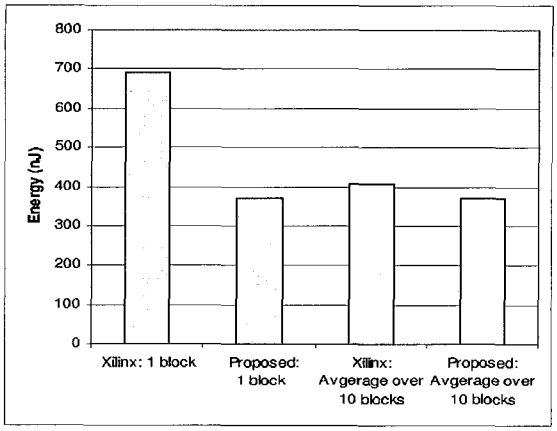
7.2 결과

비교를 위해 8×8 DCT를 다시 사용한다. 본 설계의 경우 8개의 PE가 가장 효율적이어서 Xilinx 설계와 비교를 위해 사용한다.

(그림 4)와 (그림 5)는 본 설계와 Xilinx IP 코어의 지연과 에너지를 보이고 있다. 각각의 그림은 두 가지를 비교하고 있다. 첫 번째는 한 개의 8×8 DCT에 대한 것이다. 이 경우에, 본 설계는 Xilinx에 비해서 1.1배의 속도 향상과, 320nJ 정도의 낮은 에너지 소모를 하고 있다.



(그림 4) 지연시간과 평균값



(그림 5) 에너지와 평균값

두 번째는 10개의 8×8 DCT 블록을 처리할 때 DCT 당 평균 지연 시간과 평균 에너지 소모를 비교하고 있다. Xilinx 설계의 경우 지연 시간과 에너지 소모 모두 급격하게 낮아짐을 알 수 있다. 실제로는, Xilinx의 설계는, 파이프라인 설계를 했기 때문에, 본 설계보다 에너지 소비는 많으나 짧은 지연 시간을 갖는다. 그래서 IP 코어는 64 클록 싸이클마다 한 개의 8×8 DCT를 계산한다. 본 설계는 파이프라인을 사용하지 않기 때문에 한 개의 DCT 처리 경우와 크게 다르지 않다.

(식 16)을 이용하면, 본 설계의 면적는 352개, 그 Xilinx 설계에서는 1001개의 슬라이스를 사용하므로, 본 설계가 면적 면에서 더 우수하다.

DSP와의 비교는 아래 <표 5>에 간단히 보인다.

〈표 5〉 DSP와의 에너지소모 및 지연시간 비교

	E (decrease)	T (decrease)
DSP (TI)	716.4	0.6
Proposed	409.4	0.16

8. 결론 및 추후과제

결론적으로, 본 논문은 FPGA 상에서 2차원 DCT를 위한 알고리즘과 아키텍처를 제안했다. 본 설계는 Xlinx IP 코어에 비해 더 에너지 효율적이다. 본 설계의 우수성은 도메인-스페시픽 모델을 통해 고수준 예측의 결과에서 보이고 있다.

다양한 형태의 추후 과제가 가능하다. 첫째로, 고수준 예측의 검증을 위해 저수준의 시뮬레이션을 끝낼 것이다. 그리고 본 설계의 높은 처리량을 갖는 버전을 개발 중에 있다. 이 설계에서는 DCT i의 2단계 연산이 DCT $i+1$의 1단계 연산을 중복시킨다. 연산은 본 논문에서 드러난, 덧셈기와 곱셈기가 쉬는 싸이클을 이용하여 중복 처리가 가능하다.

참 고 문 헌

[1] C.Dick, "Minimum multiplicative complexity implementation of the 2-D DCT using Xilinx FPGAs", in Proceedings of SPIE's Photonics East, November, 1998.

[2] S. Choi, R.Scrofano, V. K. Prasanna, and J.-W. Jang, "Energy-efficient signal processing using FPGAs", in Proceedings of the Eleventh ACM International Symposium on Field-Programmable Gate Arrays, 2003.

[3] S. Choi, J.-W. Jang, S. Mohanty, and V. K. Prasanna, "Domain-specific modeling for rapid system-wide energy estimation of reconfigurable architectures", in Engineering of Reconfigurable System and Algorithms, 2002.

[4] H. Lim, V. Piuri, and E. E. Swatzlander, Jr., "A serial-parallel architecture for two-dimensional discrete cosine and inverse discrete cosine transforms", IEEE Transactions on Computers, vol.49, no.12, pp.1297-1309, December, 2000.

[5] L. Mintzer, The Role of Distributed Arismetic to Digital Signal Processing in FPGAs, Xilinx, Corp.

[6] Xilinx Inc., http://www.xilinx.com.

[7] Altera Corp., http://www.altera.com.

[8] J. Anderson, F. Najm, "Low- power programmable routing circuitry for FPGAs," IEEE/ACM Conference on Computer Aided Design ICCAD-2004, pp.602-609.

[9] Texas Instruments, http://www.ti.com.

장 주 욱

1983년 서울대학교 전자공학(학사)
1985년 한국 과학기술원(석사)
1993년 미국 University of Southern California 컴퓨터공학과(박사)
1985년~1994년 삼성전자 컴퓨터개발실
1995년~현재 서강대학교 전자공학과 교수
관심분야 : 병렬처리, IT SoC, 인터넷 프로토콜

임 창 현

1998년 서강대학교 전자공학(학사)
2000년 서강대학교 전자공학(석사)
2000년~현재 서강대학교 전자공학과 박사과정
관심분야 : IT SoC, 인터넷 프로토콜

Ronald Scrofano

현 재 미국 University of Southern California 전자공학 박사과정
관심분야 : 슈퍼컴퓨팅, 병렬 분산 시스템, 네트워크 컴퓨팅

Viktor K. Prasanna

인도 Bangalore University 전자공학(학사)
인도 Indian Institute of Science 전자공학(석사)
미국 Pennsylvania State University 컴퓨터학(박사)
관심분야 : 연산처리, 병렬 분산 시스템, 네트워크 컴퓨팅, 임베디드 시스템