2차원 DWT/IDWT의 블록 데이터 플로우 구조 설계

정갑천*, 김경덕**, 박성모**
*전남대학교 전자공학과, **전남대학교 컴퓨터공학과
gcjung@ciscom.chonnam.ac.kr

Design of a Block Data Flow Architecture for 2-D DWT/IDWT

Gab-Cheon Jung*, Kyoung-Duk Kim**, Seong-Mo Park**
*Dept. of Electronics Eng., Chonnam National Univ,
**Dept. of Computer Eng., Chonnam National Univ.
gcjung@ciscom.chonnam.ac.kr

Abstract

This paper describes the design of a block data flow architecture(BDFA) which implements 2-D discrete wavelet transform(DWT)/inverse discrete wavelet transform(IDWT) for real time image processing applications. The BDFA uses 2-D product separable filters for DWT/IDWT. It consists of an input module, a processor array, and an output module. It use both data partitioning and algorithm partitioning to achieve high efficiency and high throughput. The 2-D DWT/IDWT algorithm for 256 ×256 lenna image has been simulated using IDL(Interactive Data Language). The 2-D array structured BDFA for the 2-D filter has been modeled and simulated using VHDL.

Ⅰ. 서 론

웨이브렛(wavelet) 변환은 영상처리(패턴인식, 에지 추출), 영상 압축, 신호분석 등 널리 사용되어 있으며, 특히 DWT(Discrete Wavelet Transform)의 실시간 구현은 많은 이미지 처리 응용들에 요구되어진다. DWT는 분해(decomposition)를 위한 필터링 및 decimation, 합성(synthesis)을 위한 필터링 및 interpolation 등의 복잡한 연산을 필요로 하기 때문에, 실시간 처리를 위해 일반적인 DSP 칩을 사용하는 것은 적절하지 않다.

이와 같은 실시간 처리를 위한 연산량 요구를 만족하기 위해 여러 형태의 VLSI구조 등이 제안되었다[1,2,3]. 그중 시스톨릭 어레이 구조는, 알고리즘 분할을 통하여 파이프라인 형태로 처리하는 효율적인 구조로서, 단순하고, 모듈화 되었으며, VLSI로 구현하기에 적합한 구조이다. 그러나 시스톨릭 어레이의 효율성은 전적으로 고속의 동기화된 PE(Processing Element)들 사이의 상호교류에서 통신 채널들의 사용에 달려있고, 또한 시스톨릭 어레이 시스템은 전역의 동기화된 클럭을 사용함으로써 클럭 스큐 문제와 peak 전력 소모의 문제가 발생한다.

블록 데이터 플로우 구조(BDFA)는 메쉬(mesh)구조의 장점인 속도 향상과 파이프라인 구조의 실시간율로 데이터를 처리하기 위해 제안된 구조로서, 프로세서들을 위한 직접적인 통신채널과 프로세서들 사이의 낮은 데이터 전달율, 노드 프로세서의 선택적 flexibility 등의 실시간 처리에 적합한 특성을 지니고 있다.

본 논문에서는 실시간 이미지 처리를 위한 2차원 DWT/IDWT를 구현할 수 있는 2차원 어레이 구조의 블록 데이터 플로우 구조를 설계하였다. 설계된 구조는 데이터 분할과 알고리즘 분할을 이용하며, 프로세서들과 decomposition, reconstruction stage들 사이의 제어와 통신의 복잡성을 감소시킨다.

Ⅱ. 2차원 DWT/IDWT 알고리즘

2차원 웨이브렛 변환은 행방향으로의 1차원 웨이브렛 변환과 열방향으로의 1차원 웨이브렛 변환으로 생각할 수 있다. 따라서 2차원 웨이브렛 변환은 영상의 행과 열을 1차원 필터들로 순차적으로 연이어 필터링함으로써 계산할 수 있다[5]. 이것은 separable conjugate mirror filter decomposition과 같은 알고리즘으로 2차원 product separable 필터 뱅크의 파라미트 구조로 볼 수 있다. 2차원 product separable 필터는 다음 식(1)과 같이 나타낼 수 있다.

$$y(n_1, n_2) = \sum_{i=0}^{N_1}\sum_{j=0}^{N_2} h(i,j)\,x(n_1-i, n_2-j) \qquad (1)$$
$$= \sum_{i=0}^{N_1} h(i) \sum_{j=0}^{N_2} h(j)\,x(n_1-i, n_2-j)$$

여기서 h(i,j)는 필터의 임펄스 응답을 나타내고, N_1, N_2는 수평 수직 방향의 필터의 차수를 나타낸다. 식(1)의 두 번째 식은 product separable 2차원 필터의 임펄스 응답인 h(i,j)가 1차원 필터의 임펄스 응답인 h(i)와 h(j)의 곱으로 분리되는 것을 보이준다. 또한 2차원 필터링 알고리즘의 행 열 분할(row-column decomposition)을 보여주고 있다. 이와 같이 분할된 알고리즘은 상태 변수 표현(state space representation) 방법을 사용하여 효율적으로 구현할 수 있다[6].

DWT를 위한 analysis 필터들의 경우 식(1)의 2차원 product separable 필터는 수직 상대 변수들(q_{1k}, q_{2k}

본 논문은 한국과학재단 지정 전남대학교 고품질 전기전자 부품 및 시스템 연구센터의 일부에 지원에 의해 연구되었음

$(q_{2,0}, \ldots, q_{2,N_2-1})$과 수직 상태 변수들$(q_{1,1}, q_{1,2}, q_{1,3}, \ldots, q_{1,N_1-1})$을 사용하여 식(2), 식(3)에 나타낸 바와 같이 상태 변수 방정식들의 집합으로 분할할 수 있다.

$$y_0(n_1,n_2) = v(0)x(n_1,n_2) + q_{2,1}(n_1-1,n_2) \quad (2)$$
$$q_{2,i}(n_1,n_2) = v(i)x(n_1,n_2) + q_{2,i+1}(n_1-1,n_2) \ (i=1,2,..N_2-2)$$
$$q_{2,N_2-1}(n_1,n_2) = v(N_2-1)x(n_1,n_2)$$

$$y(n_1,n_2) = h(0)y_0(n_1,n_2) + q_{1,1}(n_1,n_2-1) \quad (3)$$
$$q_{1,i}(n_1,n_2) = h(i)y_0(n_1,n_2) + q_{1,i+1}(n_1,n_2-1) \ (i=1,2,..N_1-2)$$
$$q_{1,N_1-1}(n_1,n_2) = h(N_1-1)y_0(n_1,n_2)$$

식(2)는 입력 $x(n_1,n_2)$와 이전 수직 상태 변수들 $(q_{2,i}(n_1-1,n_2))$를 이용하여 수직 방향으로의 열 연산을 수행한다. 식(3)은 $y_0(n_1,n_2)$와 이전 수평 상태 변수들 $(q_{1,i}(n_1,n_2-1))$을 이용하여 수평 방향으로의 행 연산을 수행한다.

IDWT를 위한 synthesis 필터들의 경우 식(1)의 2차원 product separable 필터는 식(4), 식(5)의 상태 변수 방정식들의 집합으로 분할할 수 있다.

$$y_1(n_1,n_2) = h(0)x(n_1,n_2) + q_{1,1}(n_1,n_2-1) \quad (4)$$
$$q_{1,i}(n_1,n_2) = h(i)y_0(n_1,n_2) + q_{1,i+1}(n_1,n_2-1) \ (i=1,2,..N_1-2)$$
$$q_{1,N_1-1}(n_1,n_2) = h(N_1-1)x(n_1,n_2)$$

$$y(n_1,n_2) = v(0)y_1(n_1,n_2) + q_{2,1}(n_1-1,n_2) \quad (5)$$
$$q_{2,i}(n_1,n_2) = v(i)y_1(n_1,n_2) + q_{2,i+1}(n_1-1,n_2) \ (j=1,2,..N_2-2)$$
$$q_{2,N_2-1}(n_1,n_2) = v(N_2-1)y_1(n_1,n_2)$$

III. 블록 데이터 플로우 구조

위의 상태 방정식들에 기반을 둔 블록 데이터 플로우 구조는 입력 모듈, 프로세서 어레이, 출력 모듈 등의 세 모듈로 구성되어진다.

1. 입력 모듈

입력 장치와 프로세서 어레이사이의 버퍼역할을 수행하는 입력 모듈은 데이터 스트림을 데이터의 블록으로 변환하는 기능을 수행한다. 입력 모듈은 프로세서 어레이 내부의 각 프로세서에게 해당하는 블록 데이터를 적절하게 할당해준다.

2. 프로세서 어레이

프로세서 어레이내 각 프로세서는 수직 프로세서와 수평 프로세서로 구성된다. 수직 프로세서는 입력모듈로부터 데이터의 행을 받아 식(2)의 열 방향으로의 상태방정식을 계산한다. 수직 프로세서는 필터링 된 출력을 수평 프로세서들에 전달하고, 수직 상태 변수들을 인접하는 다음 프로세서의 수직 프로세서로 전달한다. 각 수평 프로세서는 수직 프로세서로부터 입력을 받아 식(3)의 행방향으로의 1차원 필터링을 수행한다.

IDWT위한 구조에서는 먼저 수평 프로세서가 입력 모듈로부터 데이터의 행을 받아 식(4)의 행방향으로의 상태방정식을 계산하여 필터링된 출력을 수직 프로세서로 전달한다. 수직 프로세서는 수평 프로세서로부터의 데이터를 받아 식(5)의 열방향으로의 1차원 필터링을 수행한다.

프로세서 모듈들을 사용하여 웨이브렛 변환을 위한 2차원 어레이 블록 데이터 플로우 구조를 구성할 수 있는데 대략적인 개념도를 그림 1에 나타내었다. 이것은 m행의 1차원 어레이 모듈이 2차원으로 결합된 형태로, m은 웨이브렛 변환에 요구되는 decomposition stage 수를 나타낸다. 각 행의 출력은 다음 행으로 입력이 되는데, decimation 때문에 다음 행의 프로세서 모듈 수는 절반으로 줄어들고, 현재 행의 절반의 모듈들만 다음 행의 모듈과 연결이 되어 데이터를 전달한다.

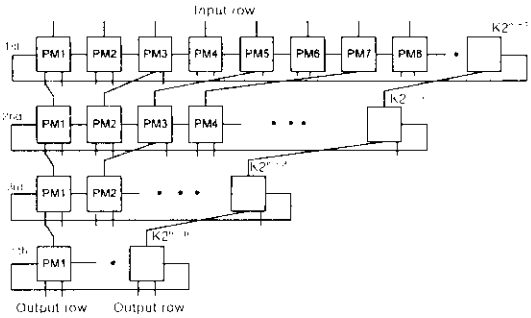

그림 1. 2차원 어레이 블록 데이터 플로우 구조 개념도

3. 출력 모듈

출력 모듈은 각 프로세서로부터 처리된 블록 데이터를 수집하여 동기화된 출력 데이터 스트림으로 변환하여 출력 장치들로 전달한다. 출력 장치는 이를 받아 데이터들을 메모리에 저장하고, 모니터 디스플레이를 위해 재배열한다.

IV. 프로세서 모듈의 모델링

2차원 웨이브렛 변환을 위한 블록 데이터 플로우의 기능 검증을 위해 상위 레벨에서 VHDL로 모델링하였으며, 시뮬레이션을 위해 행방향과 열방향에 대해 6개수를 가지는 cubic spline 웨이브렛을 사용하였다.

1. 2차원 DWT의 블록 데이터 플로우 구조 모델링

2차원 DWT를 위한 블록 데이터 플로우 구조는 각 decomposition stage(1,2,3)들에 따라 모델링 되었으며, 그림 2는 2 decomposition stage에 대한 VHDL 모델을 나타낸다. 그림에서의 입력 모듈 유닛은 하나의 demux와 프로세서 모듈로 전달될 데이터의 블록을 저장할 8개의 FIFO 버퍼를 포함하며, 입력 모듈 내 FIFO 버퍼들과 프로세서 모듈사이의 데이터 전달은 비동기적인 통신 프로토콜에 의해 수행되므로 프로세서들은 비동기적으로 동작이 가능하다. 이는 다중 프로세서 시스템에서 전역 동기화에 따른 클럭 스큐 문제를 해결할 수 있다.

프로세서 어레이 유니트(PA_unit)은 decomposition stage들의 수에 따라 8, 12, 14개의 프로세서 모듈들을 포함한다. 실시간 처리를 위해 어레이 내 프로세서 수는 영상 크기, 입력 데이터 율, 클럭 속도, 계수의 수에 따라 결정되어지며, 프로세서 수가 증가함에 따라 선형적인 속도향상을 얻을 수 있다.

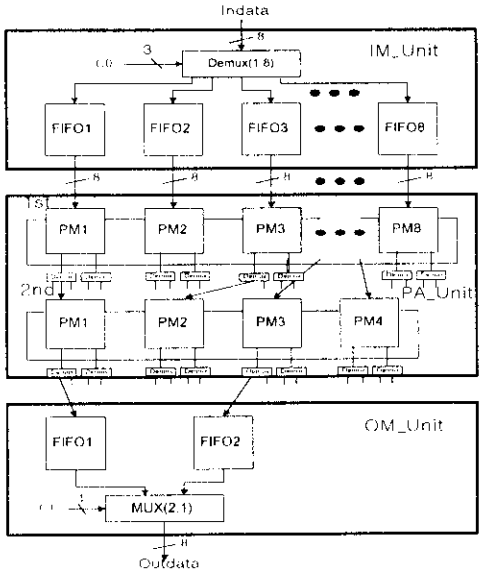

그림 2. 2차원 DWT를 위한 블록 데이터 플로우 구조의 VHDL 모델

그림 2의 출력 모듈 유니트(OM_unit)는 하나의 mux와 PA_unit로부터 데이터 스트림을 받은 2개의 FIFO들을 포함한다. 이 모듈 유니트는 데이터의 블록들을 병렬화된 출력 스트림으로 변환한다.

2. 2차원 IDWT의 블록 데이터 플로우 구조 모델링

2차원 IDWT에 대한 블록데이터 플로우 구조는 reconstruction stage들(1,2,3)에 따라 모델링되었으며, 그림 3은 2 reconstruction stages에 대한 VHDL 모델을 나타낸다. 각 구성 블록 및 모듈의 기능은 DWT에 대한 블록 데이터 플로우 구조와 같으며, 프로세서 어레이 유니트(PA_Unit1, PA_Unit2)내 덧셈기들이 추가적으로 요구된다.

3. 프로세서 모듈 모델링

프로세서 어레이내 프로세서 모듈의 블록도는 그림 4에 나타내었으며 식(2), (3), (4), (5)에서의 곱셈과 덧셈 연산들을 수행한다. 수직 프로세서내 FIFO 버퍼는 인접하는 프로세서 모듈에서 계산되어 전달된 수직 상태 변수를 받는데 사용된다. 수평 프로세서내 레지스터 버퍼(r_buff)는 다음 픽셀의 처리를 위해 수직 상태 변수들을 저장하는데 사용된다. 수평 프로세서 내 decimation 로직은 DWT를 위해 열 방향으로의 down sampling을 수행하며, 수직 프로세서내 interpolation 로직은 IDWT를 위해 열방향으로의 up sampling을 수행한다.

표1은 DWT를 위한 프로세서 모듈의 계산 과정을 나타낸다. 표에 나타나바와 같이 이전 수평 상태 변수들 $q_{1,3}(n_1,n_2-1)$과 이전 수직 상태 변수들 $q_{2,4}(n_1,n_2-1)$은 정규적인 패턴으로 현재 상태변수들과 출력의 계산에 사용된다. 수직 프로세서는 수직 프로세서들 사이의 데이터 의존도에 따라 인접한 다음 수직 프로세서 내 FIFO 버퍼로 순차적으로 수직 상태 변수들($q_{2,1}$, $q_{2,2}$, $q_{2,3}$, $q_{2,4}$, $q_{2,5}$)을 전달한다. 데이터 의존도가 없는 수평 프로세서는 수직 프로세서로부터의 결과 y_0가 계산되면 바로 출력과 수평 상태 변수들($q_{1,1}$, $q_{1,2}$, $q_{1,3}$, $q_{1,4}$, $q_{1,5}$)을 계산한다. 수평 상태 변수들은 다음 픽셀들의 사용을 위해 수평 프로세서 내 레지스터 버퍼에 저장된다.

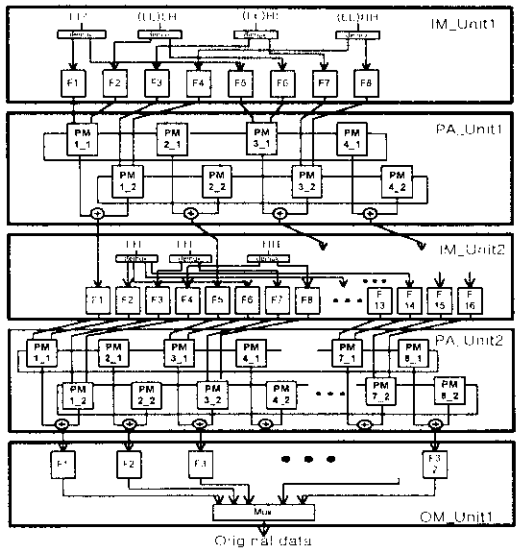

그림 3. 2차원 IDWT를 위한 블록 데이터 플로우 구조의 VHDL 모델

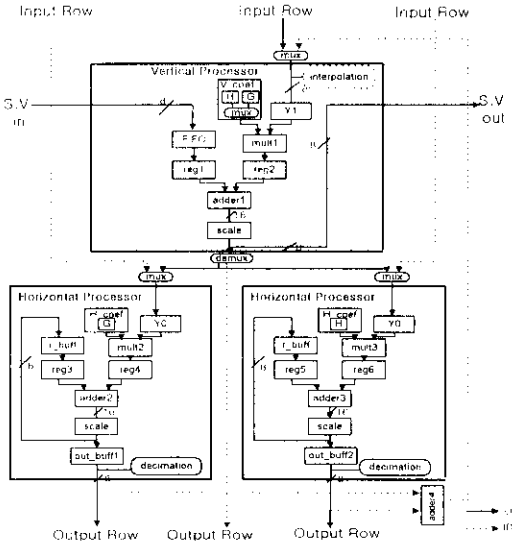

그림 4. 프로세서 모듈의 블럭도

V. 시뮬레이션 및 성능 분석

1. 시뮬레이션

본 논문에서는 DWT/IDWT를 위한 블록 데이터 플로우 구조를 상위 레벨에서 VHDL로 모델링 하였으며,

표 1. 프로세서 모듈의 입력 순서

cycle	veof	input	reg1	S.V.out	hcoef1	reg3	hcof2	reg5	out_buff1	out_buf12
1	G(0)	$x(n_1,n_2)$	$q_{2,1}(n_1-1,n_2)$	$y_0(n_1,n_2)$	0	0	0	0	0	0
2	G(1)	$x(n_1,n_2)$	$q_{2,2}(n_1-1,n_2)$	$q_{2,1}(n_1,n_2)$	G(0)	$q_{1,1}(n_1,n_2-1)$	H(0)	$q_{0,1}'(n_1,n_2-1)$	$y(n_1,n_2)$	$y'(n_1,n_2)$
3	G(2)	$x(n_1,n_2)$	$q_{2,3}(n_1-1,n_2)$	$q_{2,2}(n_1,n_2)$	G(1)	$q_{1,2}(n_1,n_2-1)$	H(1)	$q_{0,2}'(n_1,n_2-1)$	$q_{1,1}(n_1,n_2)$	$q_{1,1}'(n_1,n_2)$
4	G(3)	$x(n_1,n_2)$	$q_{2,4}(n_1-1,n_2)$	$q_{2,3}(n_1,n_2)$	G(2)	$q_{1,3}(n_1,n_2-1)$	H(2)	$q_{0,3}'(n_1,n_2-1)$	$q_{1,2}(n_1,n_2)$	$q_{1,2}'(n_1,n_2)$
5	G(4)	$x(n_1,n_2)$	$q_{2,5}(n_1-1,n_2)$	$q_{2,4}(n_1,n_2)$	G(3)	$q_{1,4}(n_1,n_2-1)$	H(3)	$q_{0,4}'(n_1,n_2-1)$	$q_{1,3}(n_1,n_2)$	$q_{1,3}'(n_1,n_2)$
6	G(5)	$x(n_1,n_2)$	0	$q_{2,5}(n_1,n_2)$	G(4)	$q_{1,5}(n_1,n_2-1)$	H(4)	$q_{0,5}'(n_1,n_2-1)$	$q_{1,4}(n_1,n_2)$	$q_{1,4}'(n_1,n_2)$
7	H(0)	$x(n_1,n_2)$	$q_{2,1}(n_1-1,n_2)$*	$y_0(n_1,n_2)'$	G(5)	0	H(5)	0	$q_{1,5}(n_1,n_2)$	$q_{1,5}'(n_1,n_2)$
8	H(1)	$x(n_1,n_2)$	$q_{2,2}(n_1-1,n_2)$*	$q_{2,1}(n_1,n_2)'$	G(0)	$q_{1,1}(n_1,n_2-1)$*	H(0)	$q_{0,1}'(n_1,n_2-1)$*	$y(n_1,n_2)$*	$y'(n_1,n_2)$*
9	H(2)	$x(n_1,n_2)$	$q_{2,3}(n_1-1,n_2)$*	$q_{2,2}(n_1,n_2)'$	G(1)	$q_{1,2}(n_1,n_2-1)$*	H(1)	$q_{0,2}'(n_1,n_2-1)$*	$q_{1,1}(n_1,n_2)$*	$q_{1,1}'(n_1,n_2)$*
10	H(3)	$x(n_1,n_2)$	$q_{2,4}(n_1-1,n_2)$*	$q_{2,3}(n_1,n_2)'$	G(2)	$q_{1,3}(n_1,n_2-1)$*	H(2)	$q_{0,3}'(n_1,n_2-1)$*	$q_{1,2}(n_1,n_2)$*	$q_{1,2}'(n_1,n_2)$*
11	H(4)	$x(n_1,n_2)$	$q_{2,5}(n_1-1,n_2)$*	$q_{2,4}(n_1,n_2)'$	G(3)	$q_{1,4}(n_1,n_2-1)$*	H(3)	$q_{0,4}'(n_1,n_2-1)$*	$q_{1,3}(n_1,n_2)$*	$q_{1,3}'(n_1,n_2)$*
12	H(5)	$x(n_1,n_2)$	0	$q_{2,5}(n_1,n_2)'$	G(4)	$q_{1,5}(n_1,n_2-1)$*	H(4)	$q_{0,5}'(n_1,n_2-1)$*	$q_{1,4}(n_1,n_2)$*	$q_{1,4}'(n_1,n_2)$*
13	G(0)	$x(n_1,n_2+1)$	$q_{2,1}(n_1-1,n_2+1)$*	$y_0(n_1,n_2+1')$	G(5)	0	H(5)	0	$q_{1,5}(n_1,n_2)$*	$q_{1,5}'(n_1,n_2)$*

시뮬레이션은 Mentor Graphics사의 quick VHDL을 이용하였다. 먼저 프로세서 모듈의 모델링 및 검증 후 프로세서 어레이가 시뮬레이션 되었다. 그 후 2차원 DWT 및 IDWT를 위한 분할 데이터 플로우 구조가 모델링되고 상위 레벨에서 검증되었다. 실제적인 시스템은 그레이 레벨의 256×256 lenna 영상에 대한 VHDL 테스트벤치를 사용하여 검증되었으며, 알고리즘 레벨에서의 IDL(Interactive Data Language)에서의 DWT/IDWT 입력 결과와 비교되었다.

2. 성능분석

2차원 DWT의 응용인 비디오 처리에서 실시간 처리를 위해 비디오 프로세서는 초당 30 프레임의 속도에서 한 프레임을 33ms내에 처리해야 한다. 각 프로세서 모듈은 각 픽셀에 대해 모든 상태 변수들과 출력들을 계산하는데 124 클럭 사이클이 소요되기 때문에 512×512 영상을 처리하기 위해서는 총 12×512×512 = 3207168 사이클이 소요된다. 표2는 512×512 비디오 이미지를 실시간에 처리하기 위해 필요한 클럭 주파수를 나타내고, 표3은 첫 번째 decomposition stage의 프로세서 모듈의 수가 8개일 때 다른 사이즈의 이미지를 처리하는데 필요한 클럭 주파수를 나타낸다. 표2, 표3에 나타낸바와 같이 프로세서 주파수 또는 프로세서의 수는 응용분야의 입출력율 요구에따라 조절할 수 있다.

표 2. 512×512 이미지에 대해 요구되어지는 프로세서의 클럭 주파수

첫 번째 stage에서의 프로세서 수	4	8	16	32
요구되는 클럭 주파수(MHz)	25	15	8	4

표 3. 첫 번째 stage에 8개 프로세서 모듈의 사용시 요구되는 프로세서의 클럭 주파수

이미지의 크기	256×256	600×800	1024×768
요구되는 클럭 주파수 (MHz)	3	25	40

VI. 결 론

본 논문에서는 2차원 product separable 필터뱅크로서 2차원 DWT/IDWT를 구현할 수 있는 2차원 어레이 구조를 제안하고 분할 데이터 플로우 구조를 VHDL로서 모델링하였다. 설계된 구조는 전역 클럭을 사용하지 않으며 프로세서들 사이의 간단한 제어 로직을 가진다. 분할 데이터 플로우 방식의 사용은 데이터 통신에 대한 요구량을 감소시키며, 전역 동기화에 대한 문제들을 해결한다. 설계된 구조는 프로세싱 소자간의 데이터 전송 속도가 낮아도 되므로 실시간 영상처리를 위한 VLSI 구조에 적합하며, 선형지연 속도 향상과 함께 고성능과 높은 효율성을 얻을수 있다.

참고 문헌

[1] Aware Wavelet Transform Processor Preliminary. Cambridge, MA :Aware, 1992.

[2] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms", IEEE Trans. on VLSI Systems, pp. 191-202, June 1993.

[3] A. Grzeszczak, et al., "VLSI Implementation of Discrete Wavelet Transform", IEEE Trans. on VLSI Systems, Vol. 4, No. 4, pp. 421-433, Dec 1996.

[4] W. E. Alexander, et al. "Parallel Image Processing with the Block Data Parallel Architecture", Proc. of the IEEE, Vol. 84, No. 7, pp. 947-968, July 1996.

[5] S. G. Mallet, "A theory of multiresolution signal decomposition :the wavelet representation", IEEE Trans. on Pattern Recognition and Machine Intelligence, Vol. 11, No. 7, pp. 674-693, July 1989.

[6] S. M. Park, et al., "A novel VLSI architecture for the real-time implementation of 2-D signal processing systems", Proc. of Int. Conf. on Computer Design: VLSI in Computers and Processors, pp. 582-585, 1988.