정보학연구 제5권 제1호 2002년 3월(pp.133-142)
The Journal of Information Technology. Vol.5. No.1

알고리즘 분해방법을 이용한 Linear Window Operator의 구현

정재길*

요 약

본 논문에서는 linear window operator의 효율적인 구현 방안을 제안하였다. 블록 상태 공간함수를 이용하여 computational primitive를 찾아내고 이를 이용하여 프로세서의 datapath의 구현에 사용함으로서 효율적인 linear window operator의 구현을 가능하게 하였다. 또한 linear window operator의 실시간 처리를 위하여 데이터 분할 기법을 사용한 다중 프로세서 구조를 제안하였다. 블록의 크기에 따른 성능 분석을 통하여 제시된 프로세서 및 다중 프로세서의 성능이 우수함을 보였다.

Ⅰ. 소개

Linear window operator는 여러 종류의 image 품질 향상, image 분석, 및 기타 다른 2차원 디지털 데이터와 관련된 알고리즘에 자주 사용되는 중요한 도구이다. 특히 최근의 고성능 비디오 전송 등 초고속 멀티미디어 통신 등에는 encoder 이전의 전처리, decoder 이후의 후처리에는 초고속의 window operator가 필요하다. 이들 대부분의 응용분야에서는 실시간 처리를 요구하는데 이를 만족시키기 위해서는 빠른 계산과 많은 데이터의 저장 및 전달이 필요하다. 또한, 향상된 성능 (performance), 정교함 (sophistication), 실시간 (real-time) 신호처리에 대한 요구는 계속 확대되고 있다.

종래의 von Neumann 컴퓨터 구조에 근거한 단일 프로세서 시스템에서 이들 알고리즘을 구현하기에는 프로세서, 메모리, 입출력 장치 사이에 필요로 되는 많은 데이터 전달로 인하여, 많은 제약과 비효율성이 따른다[1]. 따라서, 입출력 속도 또는 프로세서의 계산속도에 의해 그 시스템의 성능이 제한된다.

최근의 VLSI(Very Large Scale Integrated Circuit) 기술의 발전은 디지털 신호처리 시스템의 구현을 위한 방법에 큰 변화를 가져오고 있다. VLSI 기술이 낮은 가격에 거의 무한의 하드웨어 (hardware)를 제공함으로써, 여러 개의 프로세서 (또는 functional unit)를 한 개의 VLSI 소자에 구현할 수 있게 되었을 뿐만 아니라, 논리회로와 메모리회로를 한 개의 소자에 함께 수용할 수 있게 되고, 집적도가 높아지면서 전체 시스템을 한 개의 VLSI 소자에 내장하는 것이 가능하게 되어가고 있다. 이러한 발전은 디지털 신호처리 시스템의 구현에 있어서 알고리즘에 내재된 병렬성을 이용하여 그 성능을 크게 향상시킬 수 있는 가능성을 제시하여 주고 있다.

이러한 VLSI 기술을 최대로 활용하기 위해서는 두 분야에서 연구가 이루어져야한다. 하나는

* 배재대학교 정보통신공학부

효율적인 다중 프로세서 시스템의 구조에 관한 연구이고, 다른 하나는 다중 프로세서 시스템내의 프로세서들을 효율적으로 활용하기 위한 병렬처리 알고리즘에 관한 연구이다. 단일 프로세서 시스템에서 우수한 성능을 보이는 알고리즘이 다중 프로세서 시스템에서 그다지 높은 성능을 보여주지 못하는 경우는 아주 흔하다. 이는 단일 프로세서 시스템에서와는 달리 다중 프로세서 시스템에서는 프로세서간의 데이터 전달 등의 단일 프로세서 시스템에서는 필요 없는 동작이 필요하기 때문이다. 따라서, 이미 널리 알려진 알고리즘의 경우도 다중 프로세서 시스템을 효율적으로 이용하여 그 성능을 향상시키기 위해서는 기존 알고리즘의 분해방법을 연구하여 여러 프로세서간에 일이 균등하게 배당되고 또한 프로세서간의 데이터 전달을 최소화하여야 한다.

따라서, 발전된 VLSI기술을 이용하기 위한 알고리즘 연구는 기본적으로 일의 분배에 관한 것이다. 즉, 프로세서들 사이에 일을 적절히 배당함으로써 프로세서간의 데이터 전달량 및 속도를 최소화하고, 모든 프로세서에 같은 양의 일을 배당하여 시스템의 성능을 최대화하는 것이다. 이에는 크게 두 가지 방법, 즉 데이터 분할 (data partitioning) 방법과 알고리즘 분할 (algorithm partitioning) 방법이 사용된다[2]. 데이터 분할 방법이란 데이터를 여러 조각으로 나누어 각각의 프로세서에 균등하게 배당하는 것이고, 알고리즘 분할이란 알고리즘을 여러 조각으로 나누어 각각의 프로세서에 적절히 배당하는 것이다.

본 논문에서는 효율적인 실시간 2차원 디지털 신호처리를 위한 알고리즘 분해방법을 이용하여 실시간 window operator를 구현하였다.

II. 알고리즘(Algorithm)

Linear window operator의 병렬성을 추출하기 위하여 상태공간함수를 도구로 사용하였는데, 이 방법은 주어진 알고리즘에 대해서 계산량의 증가 없이 데이터 전달량을 최소화시킬 수 있는 가능성을 가지고 있을 뿐만 아니라 주어진 알고리즘에 대하여 여러 가지 계산구조를 구할 수 있다[3]. 이를 이용하면 간략화 된 계산 및 데이터 전달 구조, 파라미터 변화에 대한 감소된 sensitivity, 향상된 finite word length arithmetic operations[4]을 얻을 수 있다. 일반적으로 계산량, 계산의 규칙성(regularity), 계산의 정확도 사이에는 trade off가 존재하므로, 본 논문에서는 시스템의 성능(throughput)에 가장 영향을 미치는 계산의 규칙성에 중점을 두었다. 이는 VLSI의 설계시간 단축의 중요한 요소이다. 또한 프로세서간의 데이터 전달의 규칙화 및 지역화(localization)에 중점을 두었는데 이는 시스템의 확장성에 중요한 요소가 된다.

Linear window operator에 대한 일반적인 difference equation은 다음과 같다[5].

$$g(m,n)=\sum_{i=0}^{M}\sum_{j=0}^{N}h(i,j)f(m-i,n-j) \quad (1)$$

이 식에서 $h(i,j)$는 window operator의 특성을 결정하는 상수이고, $f(m,n)$은 각각의 독립변수에 대하여 일정한 간격으로 샘플 된 입력 데이터이고, $g(m,n)$은 각각의 입력 샘플에 해당하는 값을 갖는 출력이다. 시스템상수는 적절한 임의의 값을 가질 수 있기 때문에, 위의 식은 모든 linear window operator를 나타내는 식이 된다. 식 (1)의 Z-변환을 취하면 입력 데이터

의 Z-변환 $F(z_1,z_2)$ 와 출력 데이터의 Z-변환 $G(z_1,z_2)$의 관계를 다음과 같이 나타낼 수 있다.

$$G(z_1,z_2)=\sum_{i=0}^{M}\sum_{j=0}^{N}h(i,j)F(z_1,z_2)z_1^{-i}z_2^{-j} \tag{2}$$

얻을 수 있다.

$$g(m,n)=c_0f(m,n)+r_{M-1}(m-1,n)+q_{N-1}(m,n-1) \tag{3}$$

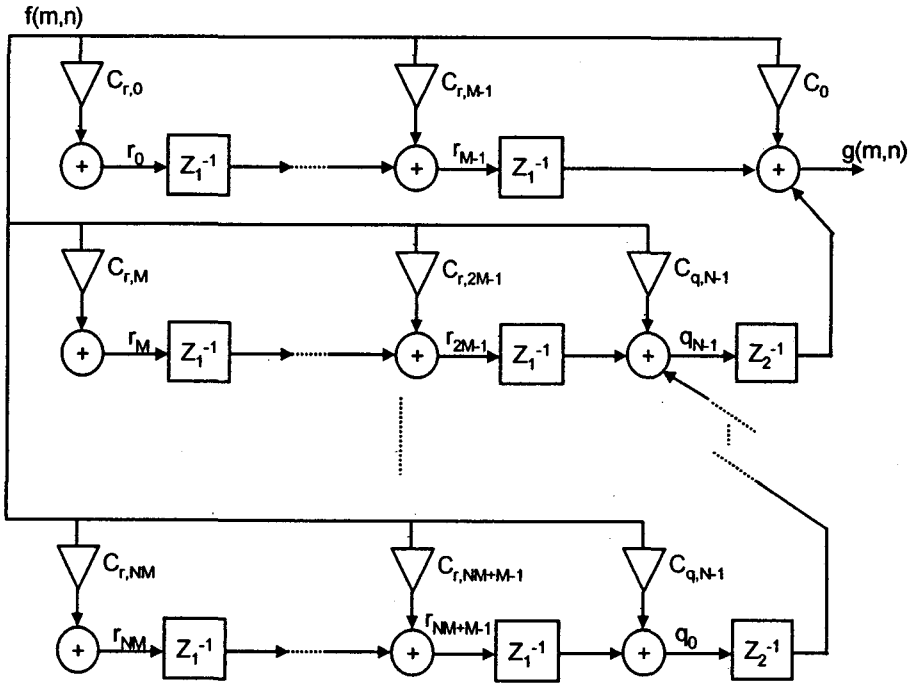

그림 1. Linear window operator의 flow diagram

식(2)로부터 그림 1에 보여진 linear window operator의 flow diagram을 얻을 수 있다. 여기에는 2가지 종류의 지연소자(delay element)가 있는데, z_1^{-1}은 한 화소지연을 z_2^{-1}는 이미지의 한 라인(line)의 지연을 의미한다. 또한 r은 수평상태변수이고 q는 수직상태변수를 나타낸다. 이 diagram으로부터 아래와 같은 상태방정식을

$$r_k(m,n)=c_{r,k}f(m,n)+r_{k-1}(m-1,n) \tag{4}$$

$$q_k(m,n)=c_{q,k}f(m,n)+r_{M(N+1-k)-1}(m-1,n)+q_{k-1}(m,n-1) \tag{5}$$

여기서

$c_0=h(0,0)$,

$c_{r,k}=h(M-i,j)$ for $0\le i\le M-1,\ 0\le j\le N,\ k=i+jM$

$c_{q,k} = h(0, N-j)$ for $1 \le j \le N$, $k = j$,

$r_k(m,n) = 0$ for $k<0$, $m<0$, or $n<0$,

$q_k(m,n) = 0$ for $k<0$, $m<0$, or $n<0$이다

만약 수평방향으로 길이 L의 블록을 사용한다면 식(4)로부터 아래와 같은 식을 얻을 수 있다.

$$r_{k-1}(m-1,n) = c_{r,k-1} f(m-1,n) + r_{k-2}(m-2,n)$$
$$r_{k-2}(m-1,n) = c_{r,k-2} f(m-2,n) + r_{k-3}(m-3,n)$$
$$\cdots\cdots\cdots\cdots$$
$$r_{k-L+1}(m-1,n) = c_{r,k-L+1} f(m-L+1,n) + r_{k-L}(m-L,n)$$

그리고 모든 식이 $r(m-L,n)$으로 표현될 때까지 recursive하게 $r_{k-1}(m-1,n)$부터 $r_{k-L+1}(m-L+1,n)$까지를 식(3), 식(4), 식(5)에 대입하면 다음과 같은 블록상태공간식을 얻을 수 있다.

$$G(m,n) = c_0 F(m,n) + \overline{R}_{M-L}(m-L,n) + Q_{N-1}(m,n-1) \quad (6)$$

$$r_k(m,n) = C_r F(m,n) + r_{k-L}(m-L,n) \quad (7)$$

$$Q_k(m,n) = C_q F(m,n) + \overline{R}_{M(N+1-k)-L}(m-L,n) + Q_{k-1}(m,n-1) \quad (8)$$

여기서

$$F(m,n) = [\, f(m,n) \;\; f(m-1,n) \;\; \cdots \;\; f(m-L+1,n) \,]^T$$
$$G(m,n) = [\, g(m,n) \;\; g(m-1,n) \;\; \cdots \;\; g(m-L+1,n) \,]^T$$
$$\overline{R}_k(m,n) = [\, r_k(m,n) \;\; r_{k+1}(m,n) \;\; \cdots \;\; r_{k+L-1}(m,n) \,]^T$$
$$Q_k(m,n) = [\, q_k(m,n) \;\; q_k(m-1,n) \;\; \cdots \;\; q_k(m-L+1,n) \,]^T$$
$$C_r(m,n) = [\, c_{r,k} \;\; c_{r,k-1} \;\; \cdots \;\; c_{r,k-L+1} \,]$$
$$C_q = \begin{bmatrix} c_{q,k} & C_{r,M(N+1-k)-1} & \cdots & c_{r,M(N+1-k)-L+1} \\ 0 & c_{q,k} & \cdots & c_{r,M(N+1-k)-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{q,k} \end{bmatrix}$$

이다.

위의 식에서 볼 수 있듯이 블록 크기가 L인 window operator의 경우에 가장 복잡한 계산을 필요로 하는 상태변수의 계산을 위해서는 L개의 곱셈과 $L+1$개의 덧셈이 필요하다. 이것을 computational primitive라 정의하고 이러한 computational primitive는 특수목적용 (application specific) 프로세서의 data path설계에 활용된다.

Ⅲ. 프로세서 구조

상태공간모델(State Space Model)은 일반화된 finite state machine[3]으로 나타낼 수 있으므로 곱셈과 덧셈은 finite state machine의 조합논리회로 부분에 구현되고, 상태변수(state variable)는 메모리(delay elements)에 저장되어 진다. 이러한 개념을 이용하여, 한 알고리즘에 대한 상태공간모델은 상응하는 finite state space machine으로 매핑(mapping) 될 수 있다. 상태공간모델과 하드웨어 구조 사이에는 직접적인 상관 관계가 있다. 하드웨어 설계를 단순화하기 위해서, finite state machine의 조합논리회로 부분에서 요구되는 계산을 computational primitive로 매핑 한다. 블록 크기가 L인 window operator에 대한 computational primitive는 식(8)과 같다. 이러한 computational primitive는 L개의 곱셈기와

$L+1$개의 덧셈기의 트리 구조를 갖는 파이프라인을 이용하여 병렬 처리할 수 있다[6-10].

다중프로세서 시스템에서 실시간 window operator를 구현하기 위해서 사용되는 프로세서 설계에는 다음과 같은 중요한 구조적 특성을 고려하였다. 프로세서는 식(7)에서 주어진 computational primitive를 1 사이클에 수행하여야 한다. 이를 위해서는 병렬로 여러 개의 계산을 수행하여야 한다. Window operator의 경우, L개의 곱셈과 $L+1$개의 덧셈, 각각의 functional unit으로의 데이터 공급, 상태변수의 저장, 데이터의 입출력, 주소의 계산 등을 병렬로 처리하여야 한다. 단일 프로세서로는 실시간 처리 시스템의 구현이 불가능하므로 다중프로세서 시스템에서 효율적으로 사용되기 위하여 프로세서간의 통신에 있어서 고속 데이터 전달기능이 있어야 한다. 프로세서는 비동기적인 다중프로세서 시스템을 구현하기 위하여 적절한 handshaking 하드웨어가 필요하다. 여러 가지 다른 목적에 사용하기 위하여, 프로세서를 프로그램어블 하도록 하는 것이 바람직하다. 예를 들면, window의 크기, 시스템 상수, 입력 데이터의 크기 또는 속도를 변경할 수 있어야 한다.

그림2는 블록 크기가 2일 때 window operator를 위한 프로세서의 블록도 이다. 상태변수(state variable)나 시스템 상수를 저장하는 queue의 주소는 자동적으로 발생되고, 데이터는 이들 queue에서 매 사이클마다 해당되는 레지스터로 보내진다. 이러한 특성은 주소계산과 데이터 전달에 따른 overhead가 전혀 없기 때문에 프로세서의 computational cycle time을 크게 줄일 수 있다. 프로세서의 arithmetic block은 2개의 곱셈기와 3개의 덧셈기를 갖고 있다. 입출력 데이터의 저장을 위한 static RAM, 2개의 가변장(variable length) 상수 queue, 상태변수저장을 위한 register file과 여러 개의 레지스터로 구성되어 있다.

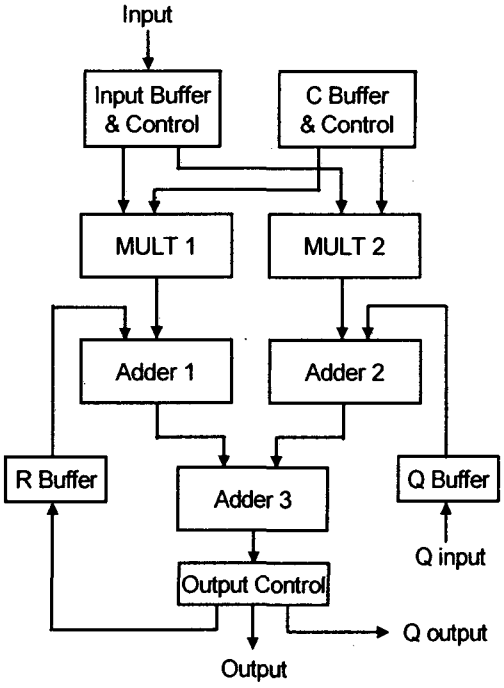

그림 2. 블록크기가 2인 window operator를 위한 프로세서 구조

Ⅳ. 시스템 구현

다중프로세서를 이용한 고속 실시간 처리를 위하여 데이터 분할(data partitioning)기법을 활용하였다. 데이터 분할기법이란 전체 처리하여야할 데이터를 여러 개의 부분(partition)으로 나누어서 여러 개의 프로세서에서 처리할 수 있도록 하는 방법이다. 데이터 분할에서 중요한 것은 분할된 데이터 사이의 데이터 의존성을 최소화할 수 있도록 하는 것이다. 의존성이 많으면 프로세서간에 많은 양의 통신이 필요하게 된다.

본 논문에서는 프로세서간의 통신을 최소화하기 위하여 한 개의 라인을 1개의 데이터 부분으로 정의하였다. 그러면 식(6), 식(8)에서 이전 라인의 수직상태변수 만을 필요로 하므로 이 것 만을 다음 프로세서에 전달하면 된다. 또한 실시간 처리를 위한 프로세서의 수도 요구되는 성능에 따라 자유롭게 정의할 수 있을 뿐만 아니라 일정한 수의 프로세서를 갖는 시스템에서 상태식으로 표시될 수 있는 모든 알고리즘을 구현할 수 있다.

그림 3은 실시간 window operator 시스템의 블록도 이다. 각각의 프로세서는 한 번에 1개의 데이터부분을 처리하도록 설계되어 있다. 이때 한 개의 데이터 부분은 한 라인의 데이터로 구성된다. 예를 들면 512×512크기의 image를 처리하고자 한다면 512개의 데이터가 1개의 데이터 부분이 된다.

Window operator에서 데이터 전달은 일반적으로 규칙적이고 예측할 수 있다. Wavefront array 프로세서는 프로그램어블 프로세서로서 VLSI로 구현하기에 유리한 규칙성(regularity)과 지역성(locality)를 갖고 있을 뿐만 아니라, dataflow array[11-12] 프로세서의 data-driven 특성을 갖고 있기 때문에, 데이터 전달이 규칙적이고 예측 가능한 알고리즘을 이용하는 window operator를 위한 다중 프로세서 구조의 좋은 개념적 모델이다. 본 논문에서 제안된 구조는 wavefront array 구조를 모델로 하여, 해당 알고리즘의 computational primitive를 효율적이고 빠르게 계산할 수 있는 특수 목적용(application-specific) 프로그램어블 프로세서로 구성된, 비동기식 다중프로세서 시스템이다.

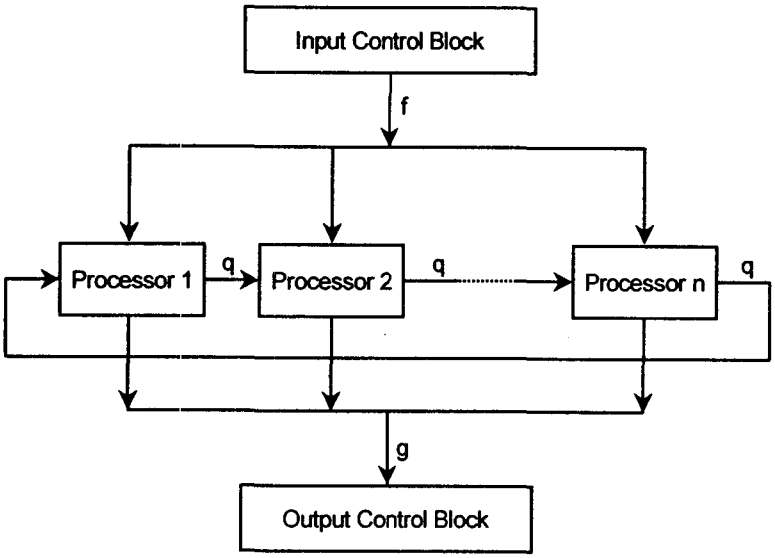

그림 3. 실시간 window operator의 다중프로세서 구조

V. 분석

제안된 알고리즘에 의해서 구해진 computational primitive를 data path로 갖는 프로세서를 이용하여 얻을 수 있는 최대 throughput은 cycle당 L output이다. Window 크기가 $M\times N$인 window operator에서 블록의 크기가 1인 경우, 1개의 출력을 계산하기 위해서는 $M\times N$개의 계산이 필요하다. 만약 각각의 프로세서가 한 개의 computational primitive를 한 사이클에 계산할 수 있다면 $M\times N$개의 프로세서를 이용하면 실시간(input data rate)처리가 가능하다. 블록크기를 증가시키면 실시간 처리를 위한 프로세서의 수를 줄일 수 있다. 예를 들어 블록의 크기가 2가 되면 window 크기가 $M\times N$ 인 경우 $(M+1)N/2$개의 프로세서가 필요하다. 따라서 window의 크기가 커지면 약 반정도의 프로세서만이 필요하게 된다.

분석을 위하여 곱하기나 더하기를 한 사이클에 계산할 수 있는 프로세서를 이용하여 한 개의 데이터를 처리하는데 필요한 사이클의 수를 T_1이라 하자. Window 크기가 $M\times N$인 window operator를 식(1)에 표시된 방식으로 직접 계산하면 $2MN-1$번의 곱하기와 $2MN-2$번의 더하기가 필요하다. 이 경우 T_1의 값은 $4MN-3$이 된다. 다중프로세서 시스템에서 프

〈표 1〉 블록 크기에 따른 성능 비교

블록크기		1	2	4	8
P		3	5	9	17
Window크기=3x3 (T_1=17)	T_P	9	6	4.5	3.75
	S_P	1.89	2.83	3.78	4.53
	E_P	0.63	0.57	0.54	0.27
Window크기=9x9 (T_1=161)	T_P	81	45	27	18
	S_P	1.99	3.58	5.96	8.94
	E_P	0.66	0.72	0.66	0.53
Window크기=32x32 (T_1=2177)	T_P	1089	561	297	165
	S_P	2.00	3.88	7.33	13.19
	E_P	0.87	0.78	0.81	0.78
Window크기=128x128 (T_1=33281)	T_P	16641	8385	4257	2193
	S_P	2.00	3.97	7.82	15.18
	E_P	0.87	0.79	0.87	0.89
Window크기=512x512 (T_1=526337)	T_P	263196	131841	66177	33345
	S_P	2.00	3.99	7.95	15.78
	E_P	0.87	0.80	0.88	0.93

로세서의 수를 P라 하고, computational primitive를 한 사이클에 계산할 수 있는 프로세서에서 한 개의 데이터를 처리하는데 필요한 사이클의 수를 T_P라 하자. 여기서 T_P는 한 개의 데이터를 처리하는데 필요한 상태식 또는 상태변수의 수와 출력의 수를 더한 것 나누기 출력의 수가 됨을 알 수 있다. Window 크기가 $M \times N$이고 블록의 크기가 L인 window operator에 필요한 사이클의 수를 살펴보자. 수평상태변수의 수는 $(M-1)N$, 수직상태변수의 수는 $L(N-1)$, 출력의 수는 L이다. 따라서 T_P의 값은 $(M+L-1)N/L$이 된다. 또한 최대 가능 speedup과 efficiency를 각각 $S_P = T_1/T_P$, $E_P = S_P/P$로 정의 할 수 있다. 여러 윈도우의 크기와 블록의 크기에 대하여 표1에 정리하였다. 이 표에서 window 크기가 큰 경우에 블록 크기가 커질수록 efficiency가 높아짐을 알 수 있고 프로세서의 수가 증가함에 따라 거의 linear speedup을 얻을 수 있음을 알 수 있다.

Ⅵ. 결론

다차원 디지털 데이터의 실시간 처리를 요구하는 Application의 증대에 따라, 이를 수행할 수 있는 시스템의 개발 경쟁은 매우 치열하다. 기존의 상용 프로세서가 충분한 성능을 제공하지 못하고, 또한 이를 이용한 다중 프로세서 시스템의 비효율성으로 인하여 개발기간 및 비용이 상대적으로 많이 소요되는 주문형 VLSI에 의존하고 있다. 본 논문에서는 적용될 알고리즘에 내재된 병렬성을 알고리즘분해기법을 이용하여 추출하고, 분해된 알고리즘의 효율적인 병렬처리를 위한 다중프로세서 시스템구조 및 이를 위한 프로세서의 구조를 제안함으로써, 실시간 window operator의 효율성을 크게 증진시킬 수 있었다.

참고문헌

1. A. V. Kulkami and D. W. L. Yen, " Systolic processing and implementation for signal and image processing," *IEEE Trans. Comput.*, vol. C-31, no. 10, pp. 1000-1009, Oct. 1982.
2. J. G. Jeong and J. W. Jang, "An efficient 2-D digital signal processor based upon the block state space representation," 한국통신학회지 제20권 제2호, pp.362-371, 1995.
3. M. Y. Dabbagh and W. E. Alexander, "Multiprocessor implementation of 2-D denominator-separable digital filters for real-time processing," *IEEE Trans. Circuits Syst.*, vol. CAS-37, no.6, pp.872-881, 1989.
4. W. W. Edmonson and W. E. Alexander, " Error analysis of a high throughput state-space model for digital signal processing," in *Proc. of the IEEE pacific RIM conf. on Communications, Computers, and Signal Processing*, June, 1987.
5. D. Dudgeon and R. Mersereau, *Multi-dimensional Digital Signal Processing.*

New Jersey: Prentice-Hall Inc., 1984.

6. J. H. Kim and W. E. Alexander, "A multiprocessor architecture for spatial domain digital filters," *IEEE Trans. Comput.*, vol. C-36, no. 7, pp. 876-884, 1987.
7. J. G. Jeong and W. E. Alexander, "The efficient real-time spatial domain 2-D IIR and FIR digital filter implementation," in *Proc. of IEEE 23rd SSST*, pp. 394-398, Mar. 1991.
8. S. M. Park, et al., "A novel VLSI architecture for the real-time implementation of 2-D dignal processing systems", in *Proc. of IEEE ICCD*, Oct. 1988.
9. J. G. Jeong, "A multiprocessor implementation of the real-time digital filter", *KITE journal of Electronics Engineering*, vol. 4, no. 1A, pp. 95-101, July 1993.
10. J. G. Jeong, H. Y. Xu, and W. E. Alexander, "A block dataflow architectures for digital signal processing", *U.S. Patent Application* No. 071837314, Mar. 1992.
11. J. B. Dennis, "Dataflow supercomputer," *IEEE Computer*, pp. 48-56, November, 1980
12. K. P. Gostelow and R. E. Thomas, "Performance of a simulated data flow computer," *IEEE Trans. Comput.*, vol. C-29, no. 10, pp. 905-919, 1980.

A Linear Window Operator Based Upon the Algorithm Decomposition

Jae-Gil Jeong*

Abstract

This paper presents an efficient implementation of the linear window operator. I derived computational primitives based upon a block state space representation. The computational primitive can be implemented as a data path for a programmable processor, which can be used for the efficient implementation of a linear window operator. A multiprocessor architecture is presented for the realtime processing of a linear window operator. The architecture is designed based upon the data partitioning technique. Performance analysis for the various block size is provided.

* Information Communication Engineering, Pai Chai University