논문 2005-42SD-2-11

벡터양자화를 위한 FNNPDS 인코더의 VLSI 설계

(VLSI design of a FNNPDS encoder for vector quantization)

김 형 철*, 심 정 보*, 조 제 황**

(Hyeung-Cheol Kim, Jeong-Bo Shim, and Je-Hwang Jo)

요 약

벡터양자화에서 고속 인코딩에 사용되는 기존 방법인 PDS(partial distance search)와 FNNS(fast nearest neighbor search)를 결합한 FNNPDS(fast nearest neighbor partial distance search)를 VLSI로 구현하기 위한 설계 방법을 제안하고, 모의실험을 통해 FNNPDS가 다른 방법에 비해 보다 고속화가 이루어짐을 입증한다. 모의실험 방법은 임의의 입력벡터에 대해 최단거리 부호벡터를 찾는 타이밍도를 고찰하고, Lena와 Peppers 영상에 대한 입력벡터당 평균 클럭 사이클을 비교한다. 모의실험 결과에 의하면 FNNPDS의 클럭 사이클 수는 다른 방법들보다 79.2%~11.7% 감소되었다.

Abstract

We propose the design method for the VLSI architecture of FNNPDS combined PDS(partial distance search) and FNNS(fast nearest neighbor search), which are used to fast encoding in vector quantization, and obtain the results that FNNPDS(fast nearest neighbor partial distance search) is faster method than the conventional methods by simulation. In simulations, we investigate timing diagrams described searching time of the nearest codevector for an input vector, and compare the average clock cycles per input vector for Lena and Peppers images. According to the result of simulations, the number of the clock cycle of FNNPDS was reduced to 79.2%~11.7% as compared with the number using the conventional techniques.

Keywords : PDS, FNNS, FNNPDS, fast searching, codebook

I. 서 론

벡터양자화의 부호화는 각각의 입력벡터에 가장 유사한 부호책의 부호벡터를 찾아야 하는데, 기존의 FS(full search)는 부호책의 모든 부호벡터를 탐색하기 때문에, 많은 연산 횟수를 필요로 한다. 이러한 계산양의 문제를 피하기 위해 많은 대안들로 PDS (partial distance search), FNNS (fast nearest neighbor search), ENNS (equal-average nearest neighbor search), ENNSV (equal-average nearest neighbor search with variance) 등이 제시되어 왔으며, 그 결과 계산양의 감소를 가져왔다[1-3].

PDS는 모든 부호벡터를 탐색하되 부호벡터의 전체 성분 중에서 일부만을 계산함으로써 계산양을 줄이는 방법이고, FNNS는 전체 부호벡터 중에서 부호벡터의 일부만을 계산함으로써 계산양을 줄이는 방법이다.

본 논문에서는 부호벡터 탐색시 PDS와 FNNS를 동시에 사용할 경우, 전체 계산양을 효과적으로 줄일 수 있다는 것에 기초하여, PDS와 FNNS를 결합한 FNNPDS를 제안하고, 제안된 방법에 적합한 VLSI 구조를 설계하고자 한다. 벡터양자화의 인코더는 높은 처리 능력이 요구되므로, 단일 프로세서 구조보다는 병렬 프로세서 구조가 더 적합하다[4]. 따라서 FNNPDS의 인코더는 병렬 프로세서의 일종인 시스토릭 어레이 (systolic array) 구조를 사용하여 설계한다.

* 정회원, 동신대학교 전기전자공학과
(Dept. of Electrical & Electronic Eng., Dongshin University)
** 정회원, 동신대학교 정보통신공학부
(Dept. of Information & Communication Eng., Dongshin University)
접수일자: 2004년5월14일, 수정완료일: 2005년1월24일

Ⅱ. 기존의 방법

1. FS

FS는 유클리드 거리에 기반을 둔 기존의 최단거리 이웃조건을 이용하여 임의의 학습벡터에 대해 모든 부호벡터를 전체 탐색하는 방법이다.

k 차원을 갖는 입력벡터 $\boldsymbol{x}$ 의 수는 M 이고, 동일 차원을 갖는 부호벡터 $\boldsymbol{y}$ 의 수가 N 이라 할 때, 각 입력벡터에 가장 가까운 부호벡터를 찾기 위해 유클리드 거리를 사용할 경우, 거리 계산을 위한 수식은 다음과 같이 나타낸다.

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{p=1}^{k} (x_p - y_p)^2 \qquad (1)$$

식 (1)은 하나의 입력벡터와 부호벡터간의 거리계산을 나타낸 것으로 한번의 학습과정에서 필요한 전체 계산양은, kMN 번의 곱셈과 뺄셈, $(k-1)MN$ 번의 덧셈, 그리고 $M(N-1)$ 번의 비교 계산이 요구된다[5]. 본 논문에서 사용되는 512×512 영상에 대해서는 k=16, M=16,384, N=128 이므로 전체 계산양은 각각 33,554,432번의 곱셈과 뺄셈, 31,457,280번의 덧셈, 그리고 2,080,768번의 비교 계산이 요구된다.

2. PDS

PDS는 모든 부호벡터를 탐색하되 부호벡터의 전체 성분 중에서 일부만을 계산함으로써 계산양을 줄이는 방법으로써, 절차는 다음과 같다. 임의의 입력벡터와 부호벡터간의 거리를 정해진 순서대로 계산할 때, 첫 번째 구해진 거리의 값을 $d_{\min}$ 이라 하자. 입력벡터와 두 번째 부호벡터의 거리는 위의 식 (1)과 같이 k 차원까지 전부 계산하지 않고, 차원이 하나씩 증가할 때마다 $d_{\min}$ 과 비교하여 이 값보다 큰 값이 나오면, 즉시 계산을 멈추고, 세 번째 부호벡터의 거리 계산으로 넘어간다. 만약 k 차원까지 구한 거리의 값이 $d_{\min}$ 보다 작으면, 이 값을 새로운 $d_{\min}$ 으로 갱신한다. 위와 같은 과정을 N 번째 부호벡터까지 수행하여 최종적으로 $d_{\min}$ 이 갱신되었을 때의 부호벡터를 입력벡터에 가장 가까운 부호벡터로 결정한다. 이 방법은 FS와 같이 전체 부호벡터를 탐색하지만, 거리 계산은 부호벡터에 따라 부분적으로 하게 되므로 FS보다 계산양을 줄일 수 있고, 이 방법으로 탐색된 부호벡터는 FS에 의해 탐색된 부호벡터와 동일하다[2].

3. FNNS

FNNS는 삼각 부등식을 이용한 대표적인 방법으로 PDS와 달리 부호벡터 간의 거리를 기억해야 하므로 추가적인 메모리가 요구된다. 즉 부호벡터가 N 개인 경우, 각 부호벡터 간의 거리를 나타내는 $N-1$ 개의 거리를 기억해야하는 전처리 과정을 갖는다.

FNNS의 절차는 다음과 같다. 첫 번째 구해진 거리의 값을 $d_{\min}$ 이라 하고, 이 때의 부호벡터를 $\boldsymbol{y}_j$ 라 하자. $\boldsymbol{y}_j$ 와 두 번째 부호벡터간의 거리를 구하여 $2d_{min}$ 보다 크면, 거리를 계산하지 않고 세 번째 부호벡터로 넘어가고, $2d_{\min}$ 보다 작을 경우에만 입력벡터에 대한 거리를 계산한다. 만약 계산된 거리가 $d_{\min}$ 보다 작으면, 이 값을 새로운 $d_{\min}$ 으로 갱신한다. 위와 같은 과정을 반복적으로 수행하여 더 이상 새로운 $d_{\min}$ 이 갱신되지 않으면, 그 때의 부호벡터를 현재 입력벡터의 가장 가까운 부호벡터로 결정한다. 이 방법은 FS와 달리 부호벡터의 일부만을 계산함으로써, 계산양을 줄일 수 있고, 이 방법으로 탐색된 부호벡터는 FS에 의해 탐색된 부호벡터와 동일한 결과를 갖는다[3].

Ⅲ. 제안된 방법

1. 제안된 방법

본 논문에서 제안된 FNNPDS는 PDS와 FNNS를 동시에 사용하여 계산양을 줄이는 방법으로써, 절차는 다음과 같다. 첫 번째 구해진 거리의 값을 d_{min} 이라 하고, 이 때의 부호벡터를 $\boldsymbol{y}_j$ 라 하자. $\boldsymbol{y}_j$ 와 두 번째 부호벡터간의 거리를 구하여 $2d_{\min}$ 보다 크면, 거리를 계산하지 않고 세 번째 부호벡터로 넘어가고, $2d_{\min}$ 보다 작을 경우에만 입력벡터에 대한 거리를 계산한다. 이때 k 차원까지 전부 계산하지 않고, 차원이 하나씩 증가할 때마다 $d_{\min}$ 과 비교하여 이 값보다 큰 값이 나오면, 즉시 계산을 멈추고 세 번째 부호벡터의 거리 계산으로 넘어간다. 만약 k 차원까지 구한 거리의 값이 $d_{\min}$ 보다 작으면, 이 값을 새로운 $d_{\min}$ 으로 갱신한다. 위와 같은 과정을 반복적으로 수행하여, 더 이상 새로운 $d_{\min}$ 이 갱신되지 않으면, 그 때의 부호벡터를 현재 입력벡터의 가장 가까운 부호벡터로 결정한다.

2. 제안된 방법의 VLSI 구현

제안된 FNNPDS의 인코더는 병렬 프로세서의 일종인 시스토릭 어레이 구조를 사용하여 설계한다. 시스토릭 어레이는 동일한 기능을 가진 PE(processing element)들이 네트워크를 구성하여 전체적인 동기 신호에 맞추어 하나의 연산을 수행하는 것으로써, FNNPDS에서의 PE는 입력벡터와 부호벡터의 한 개 성분의 거리를 계산하고, k 개의 PE를 연결하여 입력벡터와 부호벡터간의 거리를 계산하도록 설계된다. PE에 식 (1)을 적용할 경우, 곱셈 연산이 포함되어 있어 하드웨어가 복잡해진다. 곱셈 대신 절대값을 사용할 경우, 약간의 오차가 발생되지만, 하드웨어가 간소화되기 때문에 본 논문에서는 식 (1) 대신에 아래의 식을 사용한다.

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{p=1}^{k} |x_p - y_p| \qquad (2)$$

그림 1은 식 (2)에 기초하여 $d_{p-1} + |x_p - y_p|$ 를 계산하도록 설계된 PE의 세부도이다. PE는 입력벡터와 부호벡터의 한개 성분에 대한 거리를 계산하여 누적시킨다. 여기서 d_p 와 d_{p-1} 은 각각 p 번째와 $(p-1)$ 번째 PE의 출력이고, x_p 와 y_p 는 각각 입력벡터와 부호벡터의 p 번째 성분이다. d_p 는 p 번째 성분까지 계산된 부분거리를 의미한다. 8비트 감산기는 입력벡터와 부호벡터의 한 개 성분에 대한 차를 구한다. 만약 감산결과가 음수이면 2의 보수로 변환시켜 16비트 가산기에 입력한다. 16비트 가산기는 전단의 PE의 연산 결과인 d_{p-1} 과 8비트 감산기의 결과를 더하여 d_p 로 출력하고, 다음 단의 PE로 전달한다. 이러한 방법으로 그림 2에서와 같이 k 개의 PE를 연결하여 입력벡터와 부호벡터간의 거리를 계산할 수 있다. 각각의 PE는 2 사이클의 실행시간을 가지며, k 개의 차원에 대해서 입력벡터와 부호벡터간의 거리계산은 $(k+2)$ 클럭 사이클의 실행시간을 필요로 한다[6][7].

그림 2와 그림 3은 각각 FNNPDS의 구조와 순서도를 나타낸 것이다. 그림 2에서 카운터와 레지스터, k:1 MUX는 각각 그림 3에서 ADDR과 REG, MUX[CNT]에 해당된다.

그림 2의 ROM에는 부호벡터 간의 거리를 저장하고, 레지스터는 ROM의 상위 어드레스를 가리키며, 카운터는 하위 어드레스를 가리킨다. 그림 3에서는 이를 ROM[REG&ADDR]로 표시하며, 해당되는 어드레스에는 부호벡터간의 거리 $d(\boldsymbol{y}_j, \boldsymbol{y}_c)$ 를 저장한다. 그림 2의 ROM1~ROM k 에는 각각 부호벡터의 p 번째 성분을 저장하며, 카운터의 값을 어드레스로 사용한다. 그림 3에서는 ROM1[ADDR]~ROM k [ADDR]로 표시한다.

그림 2의 COMP1은 내부에 2개의 레지스터를 가진 비교기로써, 레지스터에 각각 $d_{\min}$ 과 INDEX를 저장한 후, MUX의 출력인 d_p 를 $d_{\min}$ 과 비교하여 조건에 따라 BREAK 신호를 출력하거나 $d_{\min}$ 과 INDEX를 갱신한다. COMP2는 2개의 입력을 비교하여 ENAB 신호를 출력하는 비교기이다.

FNNPDS의 동작 순서를 살펴보면 다음과 같다. 먼저 레지스터와 카운터의 값을 0으로 초기화한다. 입력벡터가 입력되면 입력벡터와 첫 번째 부호벡터간의 거리인 $d(\boldsymbol{x}, \boldsymbol{y}_0)$ 를 계산하고, COMP1은 이때 계산된 거리를 $d_{\min}$ 으로, INDEX는 현재 카운터의 값인 0으로 갱신한다. 첫 번째 부호벡터에 대한 거리를 계산한 후, 두 번째 부호벡터에 대한 거리를 계산하기 위해 카운터는

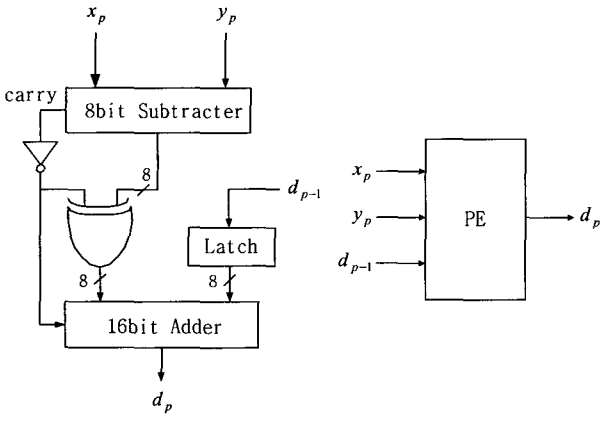

그림 1. PE의 구조
Fig. 1. Architecture of PE.

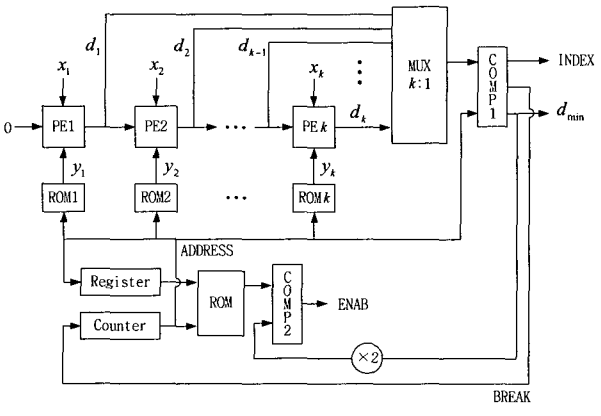

그림 2. FNNPDS의 구조
Fig. 2. Architecture of FNNPDS.

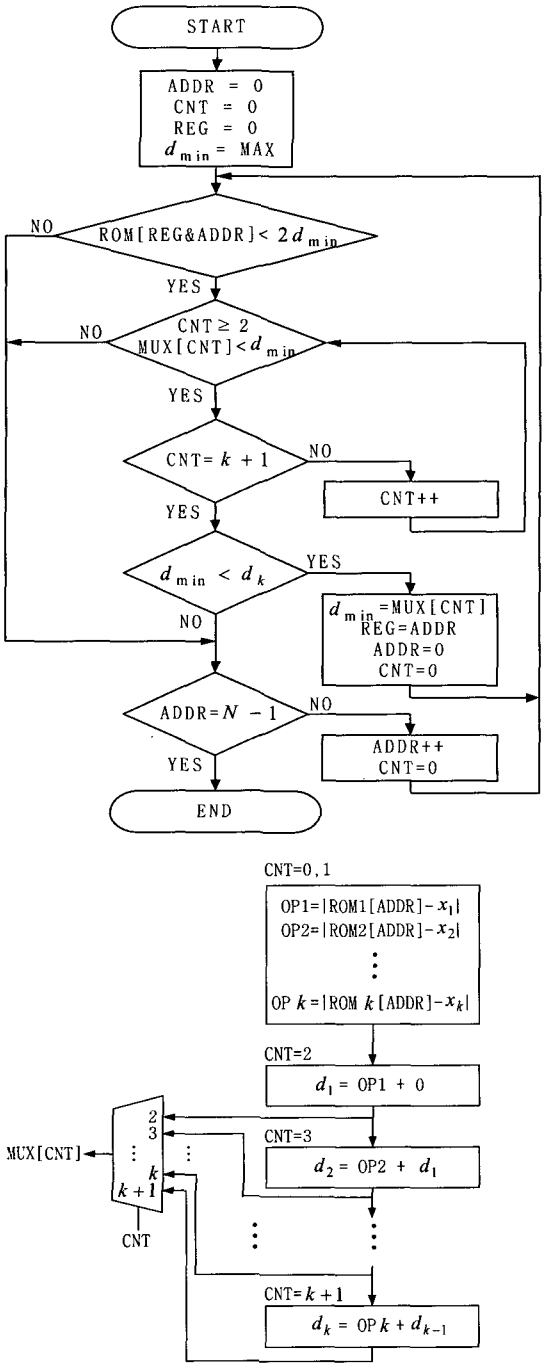

그림 3. FNNPDS의 순서도
Fig. 3. Flowchart of FNNPDS.

1 증가하고, ROM은 $d(\boldsymbol{y}_0, \boldsymbol{y}_1)$ 을 COMP2에 출력한다. COMP2는 $d(\boldsymbol{y}_0, \boldsymbol{y}_1)$ 과 $2d_{\min}$ 을 비교하여 거리를 계산할지를 결정하는 ENAB 신호를 출력한다. 만약 $d(\boldsymbol{y}_0, \boldsymbol{y}_1) > 2d_{\min}$ 일 경우, 거리계산에서 제외하고 다음 부호벡터로 넘어가기 위해 카운터는 1 증가한다. $d(\boldsymbol{y}_0, \boldsymbol{y}_1) < 2d_{\min}$ 일 경우, PE는 입력벡터와 현재 부호벡터에 대한 거리계산을 시작한다. 첫 번째 부호벡터에 대한 거리계산을 제외한 나머지 부호벡터에 대한 거리계산은 k 차원까지 전부 계산하지 않는다. 즉, 차원이 하나씩 증가할 때마다 k:1 MUX를 통해 d_p 를 $d_{\min}$ 과 비교하여, $p<k$ 일 때 $d_p > d_{\min}$ 이면, BREAK신호를 출력하여 거리계산을 즉시 중단하고 다음 부호벡터로 넘어가며, $p=k$ 일 때, $d_k < d_{\min}$ 이면, 카운터의 값을 레지스터에 저장하고 0으로 초기화한다. 이때 COMP1은 d_k 를 $d_{\min}$ 으로 갱신하고, 카운터의 값을 INDEX로 갱신한다. 이러한 방법으로 카운터가 0에서부터 $N-1$ 이 될 때까지 d_{min} 이 한번이라도 갱신되지 않으면, FNNPDS 인코더는 부호벡터 탐색을 종료하고, COMP1의 레지스터에 최종적으로 갱신된 $d_{\min}$ 과 INDEX를 출력한다.

Ⅳ. 모의실험 및 결과

1. 모의실험 방법

모의실험을 위해 256 그레이 레벨을 갖는 512×512 크기의 Lena와 Peppers 영상을 4×4로 분할하여 학습벡터로 사용하고, K-평균 알고리즘을 사용하여 크기 128의 부호책을 생성하며, 초기 부호책은 이진미소분리 방법을 사용한다. 인코더의 모의실험은 Lena와 Peppers 영상을 입력한 후 수행된 결과를 출력하고, C 언어로 작성된 프로그램으로 부호책의 성능을 검증하며, 인코더의 모의실험 결과와 비교하여 구현된 VLSI의 동작을 검증한다. 평가를 위해 PSNR(peak signal to noise ratio)을 사용한다.

$$PSNR = 20\log_{10}\left(\frac{255}{\sqrt{\frac{1}{512^2}\sum_{i=1}^{512}\sum_{j=1}^{512}(f_{ij}-g_{ij})^2}}\right) \quad (3)$$

여기서 f_{ij} 는 원 영상의 화소값이고, g_{ij} 는 복원된 영상의 화소값이다.

FNNPDS의 인코더는 VHDL을 사용하여 설계하고, Altera Maxplus 툴을 사용하여 설계에 대한 동작을 검증한다.

2. 모의실험 결과

C 언어로 작성된 프로그램 결과와 인코더의 모의실험 결과에서 PSNR은 Lena=30.44와 Peppers=30.10으로 FS, PDS, FNNS, FNNPDS 모두 동일한 성능을 보였다.

그림 4는 임의의 입력벡터에 대해 최단거리의 부호

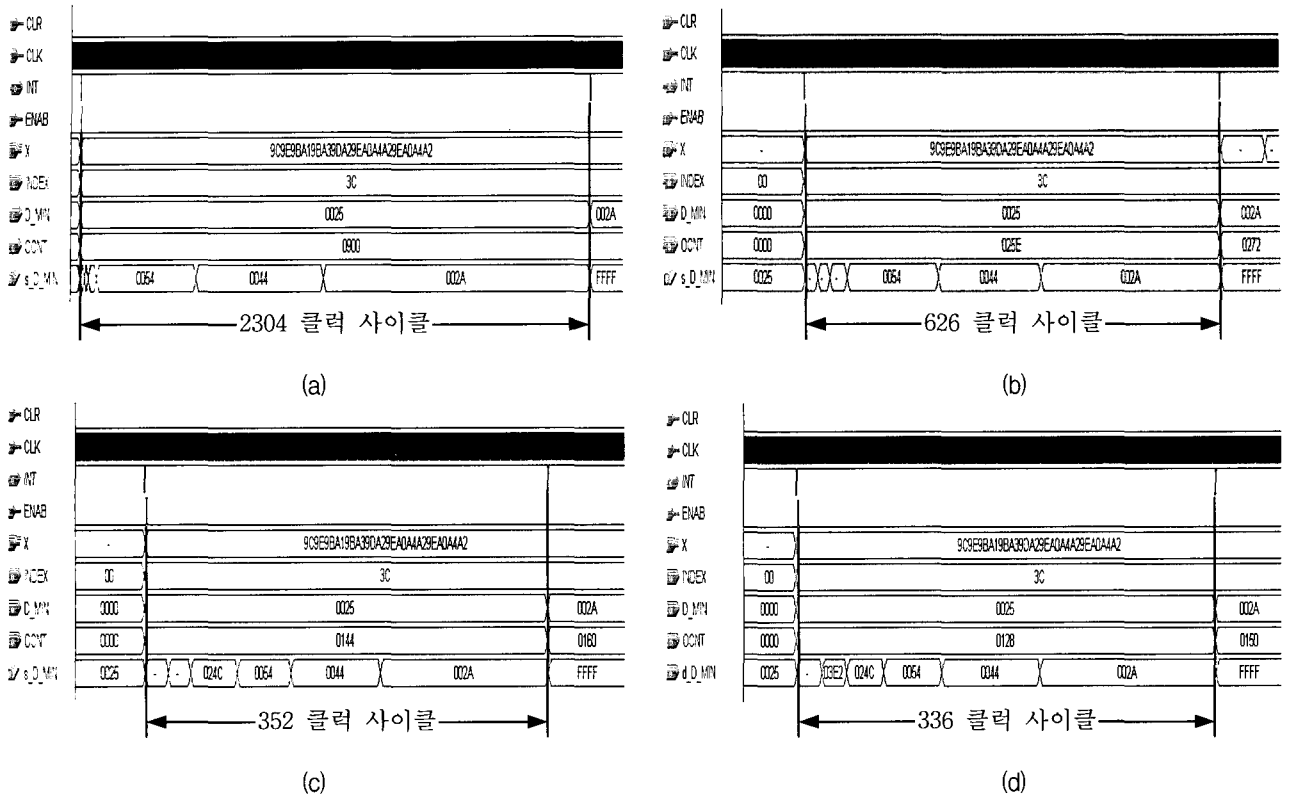

그림 4. 모의실험 타이밍도; (a) FS, (b) PDS, (c) FNNS, (d) FNNPDS
Fig. 4. Timing diagrams of simulation; (a) FS, (b) PDS, (c) FNNS, (d) FNNPDS.

표 1. $N=128$, $k=16$일때의 512×512 영상에 대한 입력 벡터당 평균 클럭 사이클 비교
Table 1. The comparison of average clock cycles per input vector for an 512×512 image at $N=128$, $k=16$.

(단위: 클럭 사이클)

실험 영상 / 탐색 방법	Lena	Peppers	평균 백분율(%)
FS	2304.00	2304.00	100.0
PDS	792.21	750.65	33.4
FNNS	568.50	521.73	23.6
FNNPDS	498.88	463.56	20.8

벡터를 찾는 FS, PDS, FNNS, 그리고 제안된 FNNPDS의 모의실험 타이밍도를 나타낸 것으로, 입력벡터가 입력된 후, 색인이 출력될 때까지 소요된 클럭 사이클 수를 나타내었다. FS, PDS, FNNS, FNNPDS 방법은 각각 2304, 626, 352, 336 클럭 사이클 후 최단거리와 그에 해당하는 색인이 출력되었다. 입력벡터가 입력된 후 색인이 출력되기까지의 클럭 사이클 수는 설계 방법에 따라 다르게 나타났지만, 출력된 최단거리와 색인은 동일하였다.

표 1은 벡터의 차원과 부호책의 크기가 동일한 조건에서 각각의 인코더가 512×512 영상을 압축하는데 소요된 총 클럭 사이클 수를 16,384개의 입력벡터 수로 나눈 평균을 나타낸 것이다. FS에서 소요된 클럭 사이클 수를 100%라 할 때, PDS는 33.4%, FNNS는 23.6%, FNNPDS는 20.8%의 클럭 사이클이 소요되었다. 따라서 제안된 방법이 다른 방법들에 비해 계산양이 감소됨을 알 수 있었다.

V. 결 론

벡터양자화의 부호화에서 입력벡터에 가장 유사한 부호책의 부호벡터를 찾아야 하는데, 기존의 FS는 입력벡터와 부호책의 모든 부호벡터를 탐색하는 것을 요구하므로, 많은 연산 횟수를 필요로 한다. 이러한 계산양의 문제를 피하기 위해 PDS, FNNS, ENNS, ENNSV 등의 대안들이 제시되었다. 본 논문에서는 부호벡터 탐색시 PDS와 FNNS를 동시에 사용할 경우, 전체 계산양을 효과적으로 줄일 수 있다는 것에 기초하여, PDS와 FNNS를 결합한 FNNPDS를 제안하고, 제안된 방법에 적합한 VLSI 구조를 설계하였다. 인코더의 모의실험은 Lena와 Peppers 영상을 입력한 후 수행하였으며, 모의실험 결과 제안된 FNNPDS에서 소요된 클럭 사이클 수는 기존

의 FS, PDS, FNNS보다 각각 79.2%, 37.6%, 11.7% 감소되었다.

참 고 문 헌

[1] A. Gersho and R. M. Gray, *"Vector quantization and signal compression,"* KAP, pp. 309-343, 1992.

[2] C. D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Trans. Commun.*, Vol. COM-33, pp. 1132-1133, October 1985.

[3] M. Orchard, "A fast nearest neighbor search algorithm," *Proc. IEEE ICASSP,* pp. 2297-2300, May 1991.

[4] P. Pirsch, *"VLSI implementations for image communications,"* Elsevier Science Publishers B.V., pp. 283-309, 1993.

[5] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, Vol. COM-28, pp. 84-95, January 1980.

[6] A. Nakada, T. Shibata, M. Konda, T. Morimoto and T. Ohmi, "A fully parallel vector -quantization processor for real-time motion -picture compression," *IEEE Journal of Solid-State Circuits,* Vol. 34, no. 6, pp. 822-829, June 1999.

[7] P. A. Ramamoorthy, B. Potu and T. Tran, "Bit-serial VLSI implementation of vector quantizer for real-time image coding," *IEEE Trans. on Circuits and Systems,* Vol 36, no. 10, pp. 1281-1290, October 1989.

저 자 소 개

김 형 철(정회원)
1997년 동신대학교 전자공학과 졸업(학사)
1999년 동신대학교 전기전자 공학과 졸업(석사)
2004년 동신대학교 전기전자 공학과 졸업(박사)
2004년～현재 (주)디셈 대표이사
<주관심분야 : 영상처리, 패턴인식, 임베디드시스템>

조 제 황(정회원)
1984년 광운대학교 전자공학과 졸업(학사)
1986년 광운대학교 전자공학과 졸업(석사)
1990년 광운대학교 전자공학과 졸업(박사)
1989년～현재 동신대학교 정보통신공학부 교수
<주관심분야 : 적응신호처리, 영상처리, 패턴인식, SoC>

심 정 보(정회원)
2002년 동신대학교 전기전자 공학과 졸업(학사)
2004년 동신대학교 전기전자 공학과 졸업(석사)
2004년～현재 (주)디셈 연구개발부장
<주관심분야 : 영상처리, RTOS, 마이크로프로세서>