OS CFAR 프로세서에 대한 새로운 시스톨릭 어레이 구조

°송 재 필*, 윤 순 영**, 이 황 수**, 곽 영 길***

*한국통신, **한국과학기술원 전기 및 전자공학과, ***국방과학연구소

A New Systolic Array Architecture for the OS CFAR Processor

°J. P. Song*, S. Y. Youn**, H. S. Lee**, Y. K. Kwag***

*Korea Telecom, **Department of Electrical Engineering, KAIST, ***ADD

Abstract

In this paper, we propose a new systolic architecture for the order statistics(OS) constant false alarm rate(CFAR) processor. In the proposed architecture, each processing element(PE) can compare two reference data cells with one test cell simultaneously in each clock cycle. So the utilization of each PE in this architecture is 100% whereas the utilization of each PE in the systolic architecture previously reported by Ritcey and Hwang is 50% because of one clock delay between two adjacent PE's active in computation. This can speed up the data processing rate by a factor of two. With this architecture, we can obtain the reduced number of communication links between adjacent PE's and reduction of the latency by half in comparison with the one proposed by Ritcey and Hwang.

1. 서론

1968년에 Finn과 Johnson이 cell-averaging(CA) constant false alarm rate(CFAR) 처리기를 제안한 이후로, 최근까지 여러가지 알고리즘이 제안되었다[1]. CA-CFAR 처리기는 균질 클러터 상황에서 가장 좋은 성능을 보이지만, 비균질 환경에서 성능이 매우 저하된다. 비균질 상황에서의 성능개선을 위하여 greatest-of(GO) CFAR 처리기와 smallest-of(SO) CFAR 처리기가 제안되었는데, 이들은 순서통계를 이용한 CFAR 처리기에 비해서 성능이 떨어진다. 순서통계를 이용한 CFAR 처리기 중에서 order statistics(OS) CFAR 처리기와 trimmed-mean(TM) CFAR 처리기가 좋은 성능을 보이는데, TM-CFAR 처리기는 OS-CFAR 처리기에 CFAR 처리기와 trimmed-mean(TM) CFAR 처리기가 좋은 성능을 보이는데, TM-CFAR 처리기는 OS-CFAR 처리기에 비해 복잡한 구조를 가진다. 따라서 본 논문에서는 여러 환경에서 좋은 성능을 보이면서 또한 구조가 간단한 OS-CFAR 처리기에 대한 하드웨어 구조를 연구한다.

OS-CFAR 처리기를 구현하기 위해서는 기준창내의 데이타들 중에서 k번째 작은 것을 선택해야 하는데, 이는 다음과 같은 네가지 방법 중의 하나로 구현이 가능하다. 첫째, 범용컴퓨터에서 sequential 알고리즘으로 구현하는 방법이 있는데[2], 거리분해능이 좋은 레이다 시스템은 높은 표본 주파수(sampling frequency)에서 동작하기 때문에 부적당하다. 둘째는 비교기로 구성된 소팅 네트웍(sorting network)을 이용하는 방법이다[3]. 셋째는 비트 레벨(bit-level)의 이진영역에서 구현하는 방법이다[4]. 넷째는 비교기를 이용하여 시스톨릭(systolic) 구조로 구현하는 방법이다[5]. 이 중에서 시험 대이타를 고려해주고 판정부분을 추가하는데는 네번째 방법이 가장 좋다. 따라서 본 논문에서는 네번째 방법을 이용하여 OS-CFAR 처리기를 구현한다.

시스톨릭 구조는 레이다 시스템에서 이미 MTI 필터나 FFT와 같은 선행검파(pre-detection) 처리과정에서 쓰이고 있다. 그러나 CFAR 처리기에서는 최근에 Ritcey와 Hwang[6]이 높은 표본 주파수에서 동작할 수 있는 시스톨릭 구조의 OS-CFAR 처리기를 제안하였다. 또한 Hwang과 Ritcey[7]는 CA-CFAR 처리기에 대해서도 제안하였는데, 이들은 파이프라인 주기(pipelining period)가 2인 구조이다. 따라서 입력 데이타 사이에 지연이 하나씩 필요하기 때문에 PE의 효율이 50%이다. 이 구조는 하나씩의 지연 PE와 판정 PE 그리고 여러개의 연산 PE로 구성된다.

본 논문에서는 OS-CFAR 처리기에 대해, PE의 효율을 100%로 증가시킴으로써 Ritcey와 Hwang이 제안한 구조에 비해 2배의 처리능력을 가지는 시스톨릭 구조를 제시한다. 제안한 구조는 하나씩의 지연 PE와 판정 PE 그리고 여러개의 연산 PE로 구성된다. 기준 데이타의 갯수를 $2n$이라고 할 때, 연산 PE의 갯수는 n이 짝수일 때는 Ritcey와 Hwang이 제안한 구조에 비해 하나가 많은 $(n+1)$, n이 홀수일 때는 $(n+2)$로 비슷한 수준을 유지하는 반면에 인접한 PE 사이에

연결되는 데이타의 링크의 갯수는 2개가 줄어든다. 또한 이 구조는 CA-CFAR 처리기에도 이용할 수 있다. 연산 PE는 OS-CFAR 처리기에서는 하나의 덧셈기와 2개의 비교기로, 그리고 CA-CFAR 처리기에서는 하나의 덧셈기로 구성된다.

2. 기존의 OS-CFAR 처리기 구조

2.1 실시간 처리를 위한 알고리즘

OS-CFAR 처리기에서 시험셀의 데이타를 Y라 하고, 기준셀내의 데이타를 $\{X_1, X_2, \cdots, X_N\}$으로 표시하자. 여기서 $N(=2n)$은 기준셀내의 데이타 갯수를 나타낸다. 이 때, OS-CFAR 처리기는 식(1)에 의해 목표물의 존재 유무를 판정한다. 즉, 시험셀 Y의 값이 (1)식을 만족하면 목표물이 존재한다고 판정한다.

$$Y > T X_{(k)} \tag{1}$$

(1)은 다음과 같이 바꾸어 쓸 수 있다.

$$\frac{Y}{T} > X_{(k)} \tag{2}$$

그런데 윗 식은, 기준창내에 $\frac{Y}{T}$보다 작은 데이타가 k개 이상 있는지를 판정하는 것과 같다. 따라서 Y에 대해 다음과 같은 방법으로 목표물을 판정할 수 있다.

과정1: $\mu_i = 0$

과정2: $j=1, 2, \cdots, N$에 대해서

$\mu_i = \mu_i + 1$ if $x_j < \frac{Y}{T}$

$\mu_i = \mu_i$ *otherwise*

과정3: *Target* if $\mu_i \geq k$

No Target *otherwise*

그림 1은 위의 알고리즘을 위해 변형시킨 OS-CFAR 처리기 블럭도를 나타낸다.

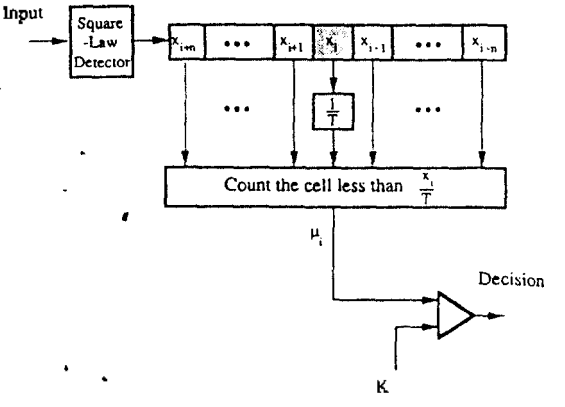

그림 1. 실시간 처리를 위한 OS-CFAR 처리기의 블럭도

2.2 구조 및 기본동작

그림 2는 앞의 알고리즘을 구현한 시스톨릭 구조를 나타낸다. 이 구조는 비교를 하기위한 n개의 연산 PE와 지연을 주는 지연 PE, 마지막으로 판정을 하기위한 판정 PE로 구성된다. 이의 동작은 다음과 같다.

x_i를 시험 데이타라 할 때, 처음에 입력데이타 x_i는 μ_i와 함께 x_{i-1}이 전달된 뒤, 한 번의 지연이 있은 다음 가장 오른쪽에 있는 PE로 전달된다. 이 때 v_i($= \frac{x_i}{T}$)가 계산된다. μ_i의 초기값은 0인데 이는 v_i보다 작은 데이타의 갯수를 나타낸다. $\{x_i, v_i, \mu_i\}$는 매 클럭마다 PE_n까지 왼쪽 PE로 전달되면서, $\{x_{i-n}, x_{i-n+1}, \cdots, x_{i-1}\}$의 기준셀내의 데이타를 하나씩 차례로 만난다. 그 때마다 v_i보다 작은 데이타를 만나면 μ_i를 증가시킨다. 다시 클럭이 기해지면 지연 PE를 지나 이번에는 PE_n부터 PE_1까지 오른쪽으로 전달되면서, $\{x_{i+1}, x_{i+2}, \cdots, x_{i+n}\}$의 기준셀내의 나머지

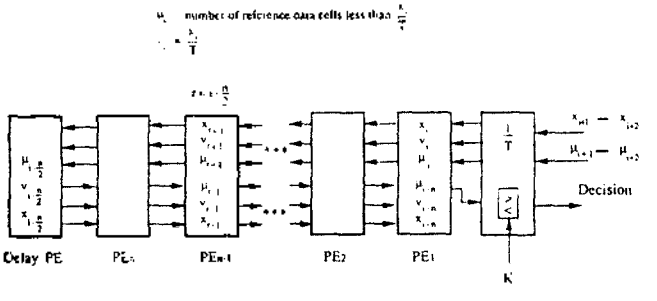

그림 2. OS-CFAR 처리기에 대한 기존의 시스톨릭 구조.

데이타들을 만난다. 물론 v_i보다 작은 데이타를 만나면 μ_i값을 증가시킨다. 위의 과정이 끝나면, x_i를 판정하기 위한 μ_i가 계산된다. 그러면, 마지막에는 μ_i가 가장 오른쪽의 판정 PE에 전달되어, 앞 절에 있는 과정3이 수행됨으로써 x_i의 판정이 완결된다.

위와 같은 x_i에 대한 동작은, $\{x_{i+1}, x_{i+2}, \cdots\}$에 대해서도 연속적으로 진행된다. 즉, x_i가 PE_3에 전달될 때 PE_1에는 x_{i+1}이 전달되고, x_i가 PE_4에 전달될 때 x_{i+1}은 PE_2에 전달되어 μ_i를 계산해 나간다. 그런데 2 클럭이 뒤진 상태에서 진행되기 때문에 2 클럭마다 판정을 수행할 수 있다.

2.3 스냅샷 (Snapshot)

그림 3은 위의 시스톨릭 구조가 어떻게 동작하는지를 보여주는, 기준창의 크기가 4인 경우의 스냅샷들이다. 여기에서 T는 1이다. 시험 데이타 3이 오른쪽으로 진행하면서 {7, 4, 2, 9}의 데이타들을 차례로 만나는 모습과 기준창내의 데이타를 만날 때 μ값이 {1, 2, 2, 3}의 순으로 바뀌는 것을 볼 수 있다.

3에 이어서, {1, 0, ... }의 데이타들에 대해서도 μ값이 계산되고 있음을 확인할 수 있다. 따라서 하나의 PE에서 2개의 시험 데이타에 대한 계산이 이루어지고 있다.

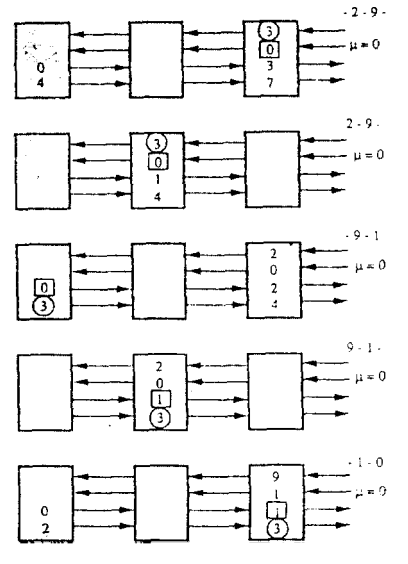

그림 3. 기존의 OS-CFAR 처리기에 대한 스냅샷.

3. 제안된 시스톨릭 구조

3.1 구조 및 기본동작

그림 4는 제안한 시스톨릭 구조의 OS-CFAR 처리기에 대한 블럭도이다. 이 구조는 비교를 하기 위한 $n+1$개의 연산 PE와 하나의 지연 PE, 마지막으로 판정을 수행하기 위한 판정 PE로 구성된다. 지연 PE와 연산 PE에 대한 동작은 그림 5에 명시되어 있다. 연산 PE의 내부구조는 그림 6과 같이 비교기 2개와 증산기 하나로 구성된다. 동작은 다음과 같다.

μ_i의 초기값은 0으로 한다. 시험 데이타를 x_i라 하면 처음에 x_i는 $\{v_i, \mu_i\}$와 함께 오른쪽에 있는 지연 PE로 전달된다. 지연 PE에서는 단지 한 클럭 지연만 시킨다. x_i는 지연 PE를 지나, PE_1에서 $PE_{\frac{n}{2}-1}$까지 전달되는 동안에, { (x_{i-n}, x_{i-n+1}), (x_{i-n+2}, x_{i-n+3}), $\cdots$, (x_{i-4}, x_{i-3}) }의 기준창내의 데이타쌍들을 차례로 만난다. 그 동안에 각 연산 PE에서는, v_i보다 작은 기준셀내의 데이타들을 만나면 μ_i의 값을 증가시킨다. 계속해서, $PE_{\frac{n}{2}}$과 $PE_{\frac{n}{2}+1}$에서 x_{i-2}와 x_{i+1}을 만나고, $PE_{\frac{n}{2}+2}$에서 PE_n까지 전달되는 동안, { (x_{i+2}, x_{i+3}), (x_{i+4}, x_{i+5}), $\cdots$, (x_{i+n-2}, x_{i+n-1}) }의 기준창내의 데이타쌍들을 차례대로 만난다. 마지막으로 PE_{n+1}에서 x_{i-1}과 x_{i+n}을 만난다. 물론, v_i보다 작은 기준창내의 데이타들을 만나면 μ_i의 값을 증가시킨다. 위와 과정이 끝나면, x_i를 판정하기 위해 필요한 μ_i의 값이 계산된다. 그러면 μ_i의 값은 판정 PE에 전달되어, $\mu_i \geq k$ 이면 목표물이 있는 것으로, 아니면 없는 것으로 판정된다.

지금까지 x_i에 대한 판정과정을 설명했는데, x_i에 대한 계산이 진행되는 동안 위의 과정은 $\{x_{i+1}, x_{i+2}, \cdots\}$에 대해서도 연속적으로 진행된다. 그런데 Ritcey가 제안한 구조와는 달리, x_i가 PE_2에 전달되면 바로 PE_1에 x_{i-1}이 전달되고, x_i가 PE_3에 전달될 때 x_{i+1}은 PE_2에 전달되는 등, μ_{i+1}을 한 클럭이 뒤진 상태에서 계산할 수 있다. 따라서 매 클럭마다 연속적으로 판정을 수행할 수 있다.

3.2 스냅샷

그림 7은 기준창의 크기가 8이고 T가 1인 경우에, 위의 동작을 보여주는 스냅샷들이다. Ritcey가 제안한 구조에서는 하나의 PE에서 2개의 시험 데이타에 대한 μ값을 동시에 계산하는 반면에, 제안한 구조에서는 하나의 시험 데이타에 대한 μ값만을 계산한다. 그 대신에 하나의 PE에서 2개의 기준창내의 데이타를 처리하고 있다. 그림 7에서 시험 데이타 9가, { (7,4), 3, 1, (0,5), (2,6) }의 기준창내의 데이타들을 차례로 만나는 모습과 μ_i값이 { 2, 3, 4, 6, 8 }로 계산되는 것을 볼 수 있다.

그림 7에서, 시험 데이타 9에 대한 2번째 기준창내의 데이타 3이 처리될 때, 시험 데이타 1에 대한 첫 번째 기준창내의 데이타 {3, 4}가 처리되고 있음을 알 수 있다. 계속 진행되어 9에 대한 판정을 행할 때, 1에 대한 마지막 기준창내의 데이타가 {9, 8}이 처리되어, 바로 다음에 1에

μ_i : number of reference data cells less than $\frac{x_i}{T}$

$v_i = \frac{x_i}{T}$

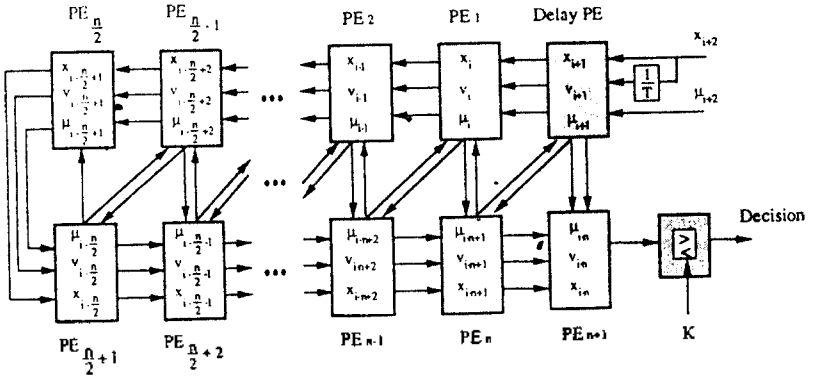

그림 4. OS-CFAR 처리기에 대해 제안된 시스톨릭 구조.

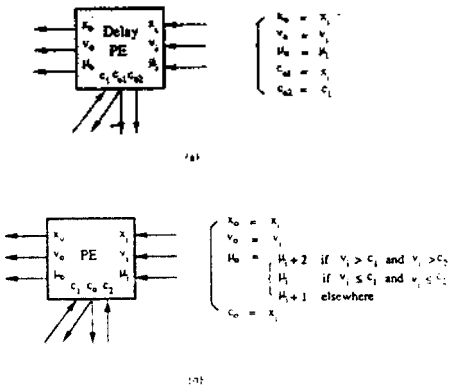

그림 5. OS-CFAR 처리기에 대한 PE의 동작

(a) 지연 PE의 기능. (b) 연산 PE의 기능.

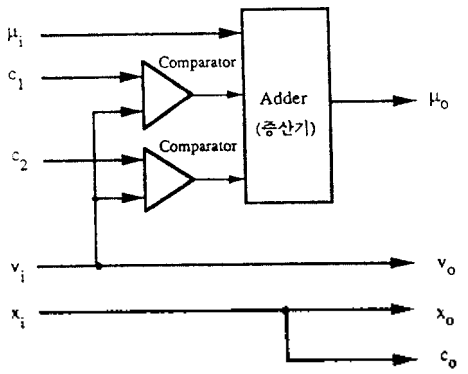

그림 6. OS-CFAR 처리기에 대한 연산 PE의 기능별 블럭도.

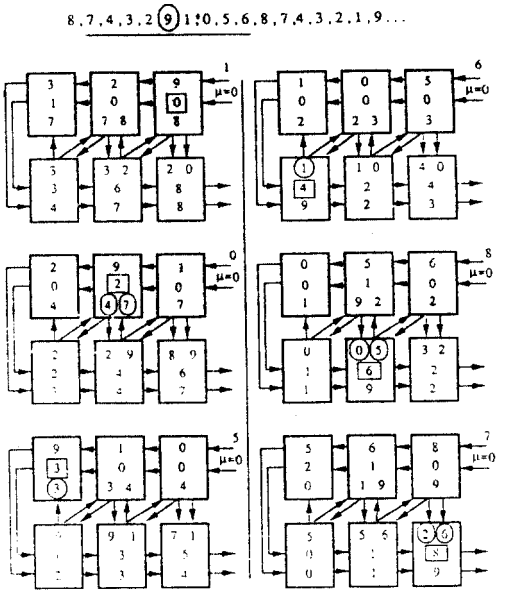

그림 7. 제안된 OS-CFAR 처리기에 대한 스냅샷.

대한 판정을 수행할 수 있다. 이는 뒤에 전달되는 데이타에 대해서도 마찬가지이므로, 매 클럭마다 연속적으로 판정을 수행할 수 있다.

3.3 CA-CFAR 처리기에 대한 구조

그림 8은 제안된 시스톨릭 구조의 CA-CFAR 처리기에 대한 블럭도이다. OS-CFAR 처리기와 비교해 볼때, v_i가 없는 것과 연산 PE의 동작만 다르다. 지연 PE와 연산 PE에 대한 동작은 그림 9에 명시되어 있다.

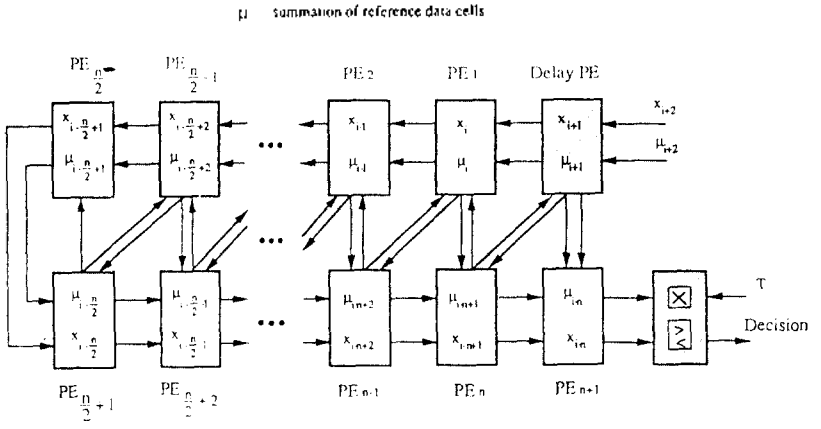

그림 8. CA-CFAR 처리기에 대해 제안된 시스톨릭 구조.

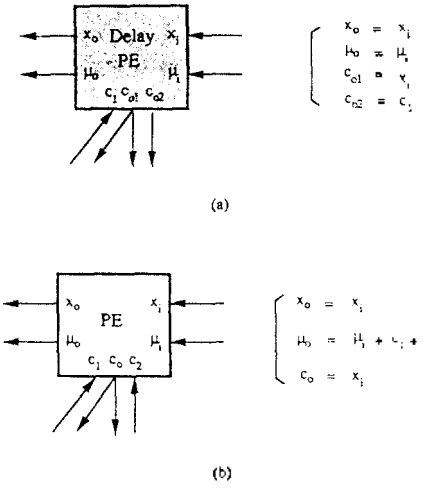

그림 9. CA-CFAR 처리기에 대한 PE의 동작

(a) 지연 PE의 기능, (b) 연산 PE의 기능.

3.4 기존 구조와의 비교

기준창내의 데이타 갯수가 $2n$이라고 할때, 본 논문에서 제안한 구조는 n이 홀수일 경우에는 그대로 사용할 수 없고, 짝수일때보다 1개가 많은 $(n+2)$개의 연산 PE가 필요하다. 예를 들면, 6개의 기준창내의 데이타를 처리하려면, 8개의 데이타를 처리할 수 있는 구조에서, 처음과 마지막 데이타는 처리하지 않고 6개만 처리하도록 구조를 약간 수정한다. 따라서 n이 홀수일 경우에는, 기준창내의 데이타가 $2(n+1)$개인 경우와 PE의 갯수가 같고, 하드웨어 복잡도는 비슷하다.

표 1은 n이 짝수인 경우에, 기존의 OS-CFAR 처리기 구조와 본 연구에서 제안한 구조를 비교한 것이다. 표 1에서 보듯이, 제안한 구조는 기존의 구조와 하드웨어 복잡도면에서 비슷하면서도, PE의 효율이 50%에서 100%로 증가되어 처리량이 2배로 늘었음을 알 수 있다.

표 1 OS-CFAR 처리기 구조에 대한 기존 구조와의 비교.

	기존의 구조	제안된 구조
PE 갯수	$n+2$	$n+3$
비교기 갯수(연산 PE)	2	2
곱셈기 갯수(연산 PE)	2 (1개 증가)	1 (2개 증가)
Latency	$2n+3$	$n+3$
PE Efficiency	50%	100%

4. 결 론

본 논문에서는 CFAR 처리기 중에서 성능이 좋은 OS-CFAR 처리기에 대한 시스톨릭 구조를 제안하였다. 이 구조는 CA-CFAR 처리기에도 쓰일 수 있다. 또한, Ritcey와 Hwang이 제안한 구조에 비해, 지연시간(latency time)은 $(2n+3)$에서 $(n+3)$으로 줄어들고, PE의 효율은 50%에서 100%로 증가하였다. 그러나 n이 짝수일 때와 홀수일 때를 따로 고려해주어야 하는 번거로움이 있는데, PE의 갯수는 n이 짝수일 때는 1개가, 홀수일 때는 2개가 많기 때문에 n이 어느정도 크면 비슷한 수준을 유지한다. 실제로 n의 크기는 10 이상이므로 하드웨어 복잡도는 비슷하다.

끝으로 제안된 구조를 살펴볼 때, PE 사이에 연결되는 데이타 수가 줄기는 했지만 아직도 이를 더욱더 줄일 수 있다.

따라서 앞으로 이를 줄이기 위한 연구가 필요하다. 즉, multirate 시스톨릭 구조나 비트 레벨 시스톨릭 구조로 구현하기 위한 연구가 필요하다.

참 고 문 헌

[1] Gandhi, P. P., and Kassam, S. A., "Analysis of CFAR processors in nonhomogeneous background," *IEEE trans. AES*, AES-24, pp.427-445, July 1988.

[2] Ataman, E., Aatre, V. K., and Wong, K. M., "A fast method for real-time median filtering," *IEEE Trans. Acoust., Speech, SignalProcessing*, vol. ASSP-28, pp. 415-421, Aug, 1980.

[3] Oflazer, K., "Design and implementation of a single chip 1-D median filter", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1164-1168, Oct. 1983.

[4] Chen, K., "Bit-serial realization of a class of nonlinear filters based on positive boolean functions," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 785-794, Jun. 1989.

[5] Kung, S. Y., *VLSI Array Processors*, Prentice-Hall, 1988.

[6] Ritcey, J. A and Hwang, J. N., "Detection performance and systolic architectures for OS-CFAR detectors," *IEEE 1990 Int. Radar Conf.*, pp 112-116, May 1990

[7] Hwang, J. N. and Ritcey, J. A., "Systolic architectures for radar CFAR detectors," *Proc. Int. Conf. IEEE ASSP*, pp. 1025-1028, April 1990.