논문 2011-48SP-5-16

# 하드웨어 복잡도를 줄인 고속 CA-CFAR 프로세서 설계

## ( Fast CA-CFAR Processor Design with Low Hardware Complexity )

현 유 진*, 오 우 진**, 이 종 훈***

( Eugin Hyun, Woojin Oh, and Jong-Hun Lee )

요 약

본 논문에서는 레이더의 탐지 알고리즘에 적용되는 CA-CFAR 알고리즘을 설계하였다. CFAR 알고리즘의 제곱평균 연산을 위해 근사화 기법을 사용하였으며, 고정 소수점을 이용하여 관련 연산을 처리하였다. 이러한 구조는 하드웨어 복잡도를 줄일 뿐 아니라 계산량을 감소시킬 수 있다. CFAR 연산은 슬라이딩 윈도우 기법을 기반으로 하는데, 이를 고속으로 처리하기 위해 동시 병렬 처리 가능한 다중 윈도우 방식도 제안하였다. 제안된 CA-CFAR 프로세서는 실제 FPGA를 통해 합성되어지고 구현되었다. 또한 FPGA 내에서 제공한 라이버러리를 이용한 제곱평균 연산 방법과 성능 비교를 하였다. 검증 결과 제안된 하드웨어 구조는 399MHz까지 동작가능하며, 전체 계산 시간은 약 70% 향상됨을 확인 할 수 있다.

Abstract

In this paper, we design the CA-CFAR processor using a root-square approximation approach and a fixed-point operation to improve hardware complexity and reduce computational effort. We also propose CA-CFAR processor with multi-window, which is capable of concurrent parallel processing. The proposed architecture is synthesized and implemented into the FPGA and the performance is compared with the conventional processor designed by root-square libarary licensed by FPGA corporation.

**Keywords :** CFAR 프로세서, CFAR 검출기, 레이더 신호처리

## Ⅰ. 서 론

최근 레이더의 기술은 국방, 차량, 의료, 보안, 선박 등 다양하게 적용되고 있다. 레이더의 송신단에서 전송된 전파가 타겟으로부터 반사되어 수신된 신호가 특정 기준치 보다 높은 경우에 이를 타겟으로 탐지하는 것이 이상적인 레이더 센서 시스템의 구조이다. 그러나 실제 환경에서는 타겟 신호가 클러터와 잡음들과 섞여 수신되므로 단순히 기준치를 정해 탐지하는 방법으로는 레이더의 성능을 만족시킬 수 없다. 이러한 문제점을 해결하는 방법이 CFAR(Constant False Alarm Rate) 알고리즘이다[1~4]. 잡음이나 클러터의 분포에 따라 적응 기준치를 적용하는 방법으로 이를 설명하는 블록다이어그램이 그림 1에 나타나 있다.

수신된 데이터는 먼저 제곱평균(Root-square)을 구한 후 레지스터(Register)에 저장된다. 저장된 각 셀들은 슬라이딩 윈도우(Sliding window) 기준으로 참조-셀(Reference Cell)과 테스트-셀(Test Cell)로 구분된다. 테스트-셀은 타겟인지 아닌지를 판단해야 할 셀을 나타내며, 참조-셀은 테스트-셀의 타겟 여부를 판단할 주변 셀들을 나타낸다. 이러한 테스트-셀은 참조셀의 관련 연산을 통해 추출 된 값과 비교하여 타겟인지 아닌지를 결정한다.

레이더 신호처리 모듈에서는 CFAR 알고리즘을

* 정회원, *** 정회원-교신저자, 대구경북과학기술원 (Daegu Gyeongbuk Institute of Science & Technology, DGIST), 로봇시스템연구부 (Robotics Research Division), 디지털레이더연구팀(Advanced Radar Technology Lab.)

** 정회원, 금오공과대학교(Kumoh National Institute of Technology), 전자공학부(Dept. of Electronic Engineering)

※ 본 연구는 교육과학기술부에서 지원하는 대구경북과학기술원 기관고유사업에 의해 수행되었습니다.(11-RS-02)

접수일자: 2011년4월7일, 수정완료일: 2011년8월17일

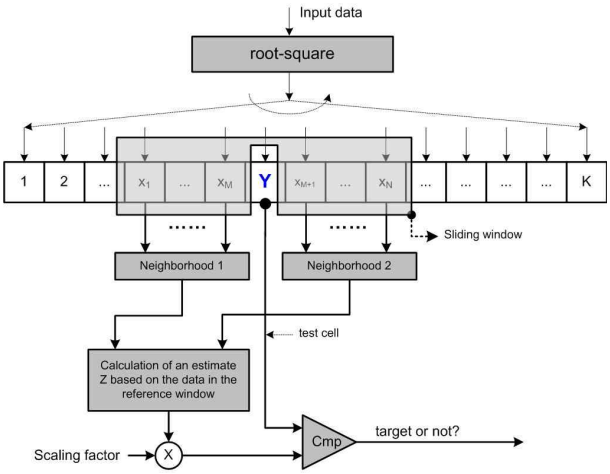


그림 1. 일반적인 CFAR 알고리즘 구조
Fig. 1. General CFAR algorithm.

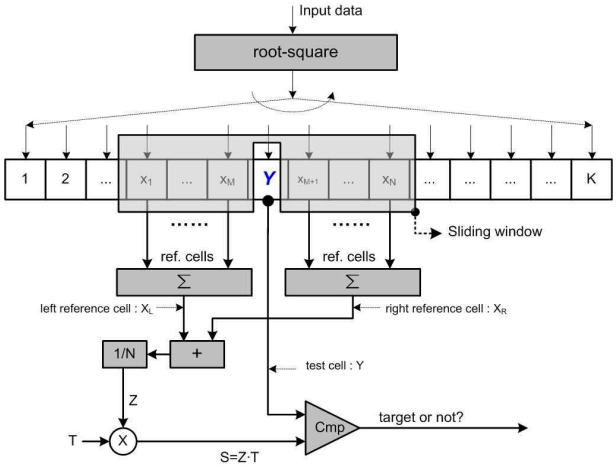


그림 2. 일반적인 CA-CFAR 알고리즘 구조
Fig. 2. Typical CA-CFAR algorithm.

DSP(Digital Signal Processor)나 FPGA(Field Programmable Gate Array)로 효과적으로 구현하기 위해서는 복잡성이 낮은 고정-소수점(Fixed-Point) 방식으로 설계 되어야 한다. 또한 슬라이딩 윈도우 방식으로 모든 셀에 연산이 적용되므로 계산량을 줄이고 고속으로 연산이 가능 하도록 설계되어야 한다.

본 논문에서는 간단한 구조를 가지며 고속 연산이 가능한 CA-CFAR(Cell Average CFAR) 알고리즘의 구조를 제안한다. CA-CFAR는 비교적 계산량이 작고 구조가 간단하여 보편적으로 사용되는 CFAR 알고리즘이다.

본 논문에서는 또한 다중 윈도우 방식을 이용하여 병렬 구조를 가지게 함으로써 연산속도를 향상시킬 수 있는 방법도 제안한다.

제안된 방법은 HDL로 코딩된 후 FPGA 합성 프로그램을 이용하여 성능을 검증하였다. 또한 FPGA 내에서 제공하는 라이버러리(Library)를 이용한 제곱평균 연산과 성능 결과를 비교 나타내었다.

먼저 Ⅱ장에서는 일반적인 CA-CFAR에 대해 설명하고, Ⅲ장에서는 구현된 CA-CFAR 프로세서를 소개한다. Ⅳ장에서는 시뮬레이션 결과를 보여주고, Ⅴ장을 통해 결론짓고자 한다.

## Ⅱ. 일반적인 CA-CFAR 구조

CFAR 알고리즘은 참조셀을 이용한 평균-클러터-전력을 구하는 방식에 따라 다양한 알고리즘으로 나누어진다. 그 중 CA-CFAR 알고리즘은 평균을 이용하여 클러터-전력을 구하는 방법으로 그림 2에 일반적인 CA-CFAR 프로세서의 구조를 나타내었다.

레지스터에 저장된 수신 데이터의 제곱평균 값은 슬라이딩 윈도우에 의해 오른쪽 참조셀, $X_L$과 왼쪽 참조셀, $X_R$로 나누어진다. 참조셀의 데이터의 덧셈과 나눗셈을 통해 평균-클러터-전력(Mean Clutter Power), $Z$가 생성하게 된다. 그리고 스켈링-펙터(Scaling Factor), $T$와의 곱셈 연산을 통해 기준치인 $S = Z \bullet T$를 생성하게 되고, 테스트-셀, $Y$와 비교를 통해 타겟인지의 여부를 결정한다. 이러한 연산은 윈도우 슬라이딩을 통해 모든 셀에 적용된다.

CA-CFAR 알고리즘을 효과적으로 구현하기 위해서는 제곱평균 연산, 덧셈기, 평균연산($1/N$), 스켈링-펙터(Scaling Factor) 곱셈 연산 등이 낮은 복잡성을 가지며 고속 연산이 가능하도록 구현되어야 한다.

일반적으로 연산 알고리즘을 구현하는 방법으로는 룩-업-테이블(Look-up-table)을 이용하는 방법과 테일러-시리즈(Taylor Series)와 같은 근사화 기법을 이용한 방법이 많이 활용된다. 룩-업-테이블을 이용한 방법은 비교적 간단하지만, 높은 해상도로 구현하기 위해서는 많은 메모리가 필요로 하는 단점이 있다.

따라서 본 논문에서는 근사화 기법을 이용하여 제곱평균 연산을 구현하고, 덧셈기, 곱셈기, 그리고 시프트(Shift) 연산을 이용하여 CFAR 프로세서를 구현하였다.

## Ⅲ. 제안된 CA-CFAR 구조

그림 3은 제안된 CA-CFAR 구조를 나타내었고 주요 특징은 아래와 같다.

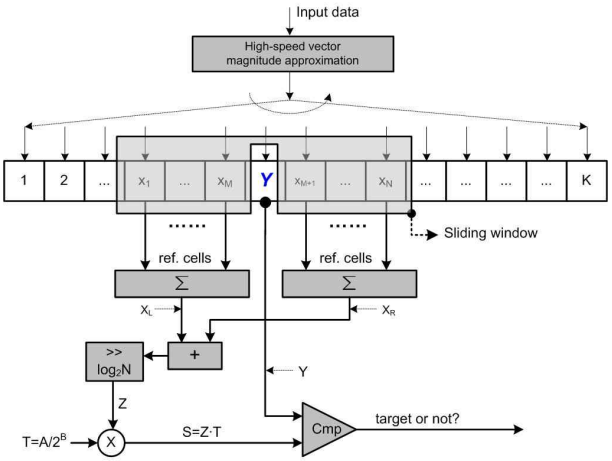


그림 3. 제안된 CA-CFAR 알고리즘 구조
Fig. 3. Proposed CA-CFAR algorithm.

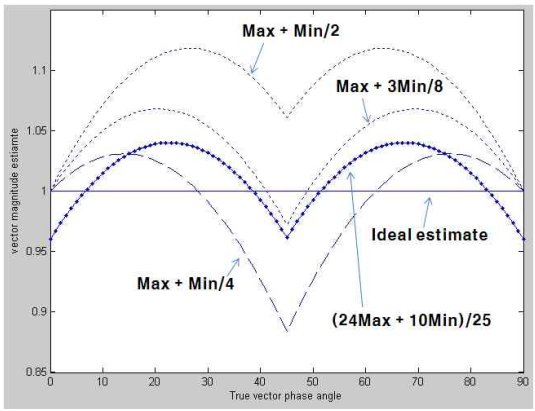


그림 4. 제곱평균 근사화연산의 모의 결과
Fig. 4. Estimation results for root-square approximation.

먼저 $\sqrt{\alpha^2+\beta^2}$로 표현되는 제곱평균 연산을 고속-벡터-크기-근사화(High-seed vector magnitude approximation) 기법[5] (이하, '제곱평균 근사화연산'이라 명칭 함)을 이용하여 구현하였다.

또한 N개의 참조셀의 평균값을 구하기 위한 $1/N$ 연산은 $\log_2 N$만큼의 오른쪽-쉬프트(Shift Right) 연산 통해 쉽게 구현하였다. 일반적인 CFAR 연산에서 윈도우의 길이는 8, 16, 32 등과 같이 2의 승수로 사용되기 때문에 $\log_2 N$은 정수값으로 표현할 수 있다.

CFAR 연산에서 스켈링-펙터 값은 윈도우의 크기와 오경보율의 상관관계로 구해지며 소수점 값으로 표현된다. 이를 효과적인 고정소수점 연산으로 표현하기 위해서는 $T \approx A/2^B$와 같이 근사화 시킨 후, $A$는 곱셈연산으로 $1/2^B$는 $B$비트 오른쪽-쉬프트 연산으로 쉽게 구현될 수 있다.

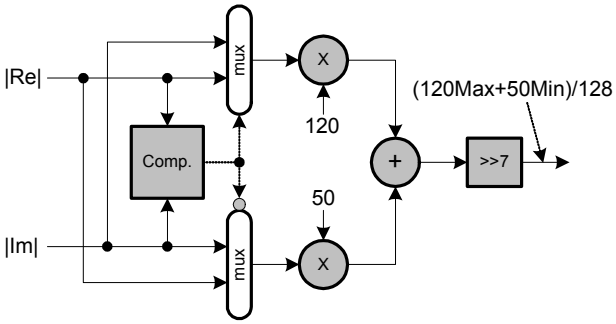


그림 5. 제곱평균 연산 구조 설계
Fig. 5. Hardware implementation for root-square estimation.

그림 4는 구현된 제곱평균 근사화연산의 분석을 나타내고 있다. 가로축은 위상이 0~90°의 범위를 가지며, 세로축은 제곱평균 연산값을 나타낸다. 먼저 "Ideal estimate"는 $\sqrt{\alpha^2+\beta^2}$의 결과로 입력신호의 위상에 상관없이 항상 일정한 값을 가진다. 그러나 근사화 수식 $Max+Min/2$, $Max+3Min/8$, $Max+Min/4$는 위상에 따라 약간씩 오차가 발생한다. 이때 $Max$은 $|\alpha|$ 와 $|\beta|$ 중 큰 값으로 선택되고 $Min$은 나머지 값이다.

본 논문에서는 위상에 따른 오차값을 최소화 하고 이상적인 값을 중심으로 분포될 수 있도록 $(24Max+10Min)/25$의 수식으로 근사화하였다.

제안된 $(24Max+10Min)/25$를 고정소수점을 이용한 하드웨어로 구현하기 위해 $(24Max+10Min)5/128$로 표현할 수 있다. 분모 128은 비트-오른쪽-이동 연산을 통해서, 분자는 정수 곱셈 연산을 통해 구현 가능하며 그 하드웨어 구조가 그림 5에 나타나 있다.

앞서 설명하였듯이 CFAR 연산은 윈도우 슬라이딩 기법을 사용한다. 따라서 테스트 셀을 중심으로 오른쪽과 왼쪽 참조셀이 존재한다. 이때 윈도우 상에 존재하는 참조셀 값은 단순히 한 개의 테스트 셀을 위해 사용되는 것이 아니라, 윈도우가 슬라이딩이 됨에 따라 2번씩 중복으로 사용된다. 예를 들면, 테스트 셀 $Y$가 10번째 데이터이고 윈도우 크기가 8이라면, 왼쪽 참조 셀은 6~9번 째 데이터로, 오른쪽 참조 셀은 11~14번 째 데이터로 구성되어 질 것이다. 이때 오른쪽 참조 셀 11~14번 째 데이터는, 15번째 데이터가 테스트 셀일 때는 왼쪽 참조 셀로 활용된다.

따라서 다중 슬라이딩 윈도우를 이용하여 여러개의 테스트 셀에 동시에 CFAR 연산을 적용한다면, 전체 연산 속도를 향상시킬 뿐 아니라 공통으로 사용되는 참조셀의 연산을 활용할 수도 있다. 그림 6은 2개의 탐지 결

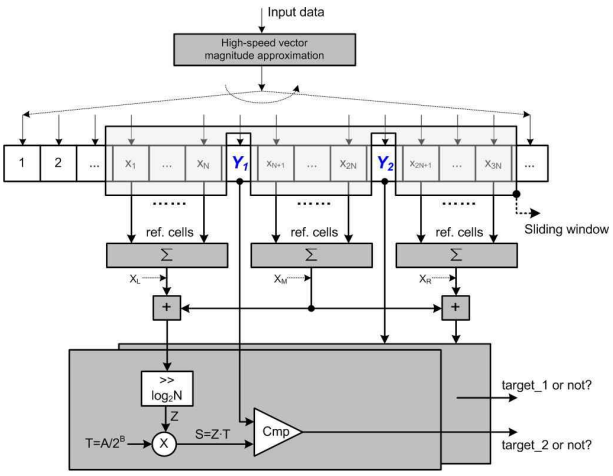


그림 6. 다중 윈도우를 이용한 CA-CFAR 알고리즘 구조

Fig. 6. Proposed CA-CFAR algorithm with multi-window.

과를 동시에 출력 할 수 있는 다중 윈도우 구조의 CA-CFAR 프로세서를 나타낸다.

## Ⅳ. 시뮬레이션 결과

제안된 CA-CFAR 프로세서는 Verilog HDL로 구현되었으며, 합성에 사용되어진 툴(Tool)은 Xilinx ISE v11.0 이다. 성능 검증을 위해 Xilinx Virtex-5 FPGA를 기반으로 합성 및 타이밍 분석이 이루어졌다.

일반적인 CA-CFAR 및 제안된 프로세서는 그림 7과 같이 구현되어 졌다. 입력 신호로는 클럭(CLK), 리셋(RST), 입력 데이터 I 및 Q (Re_Data, Im_Data), 그리고 입력신호 인에이블(Valid_In)로 구성되었고, 출력 신호로는 CFAR 처리 결과(CFAR_result)와 출력신호 인에이블(Valid_Out)로 구성되었다. CFAR 프로세서의 내부 블럭은 크게 3부분으로 구성된다.

첫 번째는 “Root square” 블록으로, 일반적인 CFAR 연산을 구현하기 위해서는 곱셈기를 이용한 제곱연산과 Xilinx에서 제공하는 “Root” 연산 라이버러리를 활용하여 $\sqrt{\alpha^2+\beta^2}$ 를 구현 하였다. 반면에 제안된 방법에서는 “Root square” 블록을 $(24Max+10Min)5/128$ 모델로 구현하였다.

두 번째로 “CFAR Core” 블록에서는 평균기와 스켈링-펙터 곱셈, 그리고 비교기 등을 고정 소수점 방식으로 구현하였다.

마지막으로 “Counter” 부분에서는 전체 프로세서의

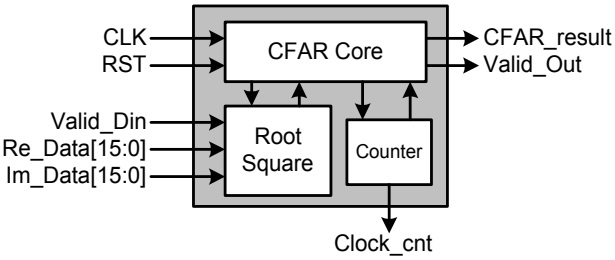


그림 7. 구현된 CA-CFAR 프로세서 상위 블럭

Fig. 7. Top block of designed CA-CFAR processor.

동작 클럭수를 모니터 할 수 있도록 하였다.

성능 비교를 위해 입력 데이터는 1024개로 하였으며, 윈도우의 크기는 8로 고정하였다.

일반적인 CA-CFAR 프로세서, 제안된 CA-CFAR 프로세서, 2중 윈도우를 가지는 CA-CFAR 프로세서의 전체 소모 클럭수를 HDL 합성 및 타이밍 시뮬레이션을 통해 표 1과 같은 결과를 추출하였다.

전체 CA-CFAR 연산 소모 클럭수에서 일반적인 방법이 가장 많은 클럭이 소모되는데, 그 이유는 제곱평균 연산에 많은 연산 시간이 소모되기 때문이다. 즉, 일반적인 방법에서는 한 개의 셀 당 제곱평균 연산에 평균 24 클럭이 소모되는데 반해, 제안된 방법에서는 5 클럭만 소모되므로 약 76%의 성능 향상이 이루어진 셈이다. 또한 2중 윈도우를 이용하는 경우 윈도우 슬라이딩의 수를 반으로 줄일 수 있으므로 평균 연산, 곱셈 연산, 비교 연산에 소모되는 시간이 반으로 줄게 됨을 알 수 있다. 1024 클럭이 소모되는 데이터 입력 시간과 그 외 연산을 모두 합한 경우, 한 개의 셀 당 CFAR 전체 연산에 소모되는 평균 연산 속도 성능은, 단일 및 2중 윈도우를 사용하는 경우 각각 60%와 68% 향상됨을 알 수 있다.

그림 8~10은 각 CA-CFAR 프로세서의 타이밍 시뮬레이션 결과를 나타내었다. 복소수 입력 데이터 Re_Data[15:0]와 Im_Data[15:0]가 인에이블(Eanble) 신호인 Valid_Din과 함께 입력되면, CFAR 출력 값은 인에이블 신호 Valid_Dout과 함께 CFAR_Result 신호로 출력된다. 특히, 그림 10의 경우 다중 윈도우를 사용한 구조로 CFAR 결과값이 두 개의 신호로 출력됨을 확인할 수 있다. 이때 카운터(Countor) 값을 이용하여 최종 연산 시간을 확인할 수 있다.

“Root square” 블록를 Xilinx Virtex-5로 합성하여 소모된 하드웨어 리소스를 표 2에 나타내었다. 소모된 하드웨어 리소스는 레지스터 수(Number of Slice

표 1. 프로세서 간의 소모 클럭수 비교
Table 1. The number of clock comparisons between the typical and the proposed architectures.

| 클럭수 | | 일반적인 방법 | 제안된 방법 | |
|---|---|---|---|---|
| | | | 단일 윈도우 | 다중 윈도우 |
| CFAR 연산 전체 | | 25,571 | 10,215 | 8,174 |
| 기능별 | 데이터 입력 | 1,024 | 1,024 | 1,024 |
| | 제곱평균 연산 | 21,506 | 5,126 | 5,126 |
| | 그 외 연산 | 4,065 | 4,065 | 2,024 |
| 테스터 1개 셀 당 | CFAR 연산 전체 | 21 | 5 | 5 |
| | 제곱평균 연산 | 24 | 8 | 10 |

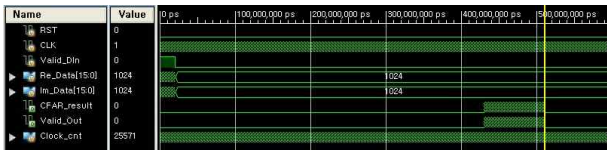


그림 8. 일반적인 CA-CFAR 알고리즘 구현 결과
Fig. 8. Implementation result of the typical CA-CFAR processor.

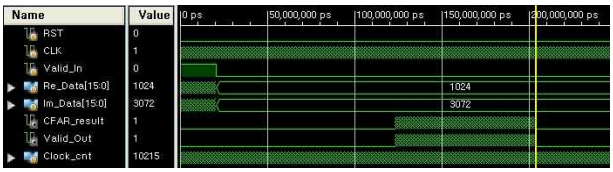


그림 9. 제안된 CA-CFAR 알고리즘 구현 결과
Fig. 9. Implementation result of the proposed CA-CFAR processor.

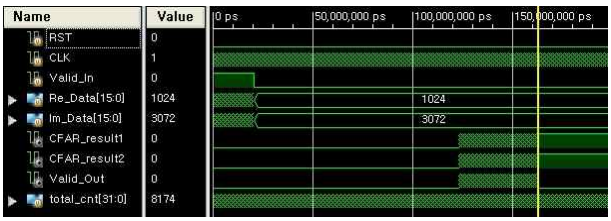


그림 10. 다중 윈도우 구조로 제안된 CA-CFAR 알고리즘 구현 결과
Fig. 10. Implementation result of the proposed CA-CFAR processor with multi-window.

Registers), LUT 수(Number of Slice LUTs), LUT-FF 조합수(Number of fully used LUT-FF pairs), 곱셈연산기 수(umber of DSP48Es)로 나타난다.

LUT는 Xilinx FPGA 내에서 조합회로를 구현하기 위해 룩-업-테이블(Look Up Talbe)을 나타내는 것으로 소모되는 하드웨어 리소tm를 대표하는 블럭이다.

LUT-FF은 순차회로를 구현하기 위해 소모되는 LUT 로직과 플립플롭(Flipflop)의 조합을 나타낸다.

표 2. 제곱평균 연산에 사용된 하드웨어 리소스 비교
Table 2. Hardware resource comparisons between the typical and the proposed root-square block.

| 항목 | 일반적인 방법 | 제안된 방법 |
|---|---|---|
| Number of Slice Registers | 459 | 18 |
| Number of Slice LUTs | 407 | 74 |
| Number of fully used LUT-FF pairs | 208 | 17 |
| Number of DSP48Es | 6 | 2 |

표 3. 프로세서 간의 소모 시간수 비교
Table 3. Time comparisons between the typical and the proposed architectures.

| 구분 | 일반적인 방법 | 제안된 방법 | |
|---|---|---|---|
| | | 단일 윈도우 | 다중 윈도우 |
| 동작 소모 클럭수 | 25,571 | 10,215 | 8,174 |
| 동작 주파수 (MHz) | 341 | 399 | 399 |
| 전체 소모 시간 (us) | 75 | 26 | 20 |

마지막으로 DSP48Es는 Xilinx FPGA 내에 실장된 곱셈연산기를 나타낸다.

CA-CFAR 프로세서의 합성 결과 일반적인 방법에 비해 제안된 방법에서 레지스터의 수는 96%, LUT 수는 82%, LUT-FF 조합은 92%가, 곱셈연산은 67% 만큼 사용된 하드웨어 리소스가 줄어든다. 일반적인 방식의 경우 "$\sqrt{\ }$" 연산을 위해 사용된 Xilinx IP가 하드웨어 리소스를 많이 소모하고, $\alpha^2+\beta^2$을 위해 추가적으로 하드웨어 리소스가 소모되기 때문이다.

합성 결과 일반적인 방법에서는 동작 주파수가 341MHz로, 제안된 기법이 적용된 방법에서는 399MHz로 조금 향상됨을 알 수 있다. 이를 기준으로 전체 소모 시간을 계산해보면 표 3과 같다. 전체 소모 시간이 일반적인 방법에 비해 약 70% 향상됨을 확인 할 수 있다. 이때 입력 데이터 수는 1024개이며 윈도우 크기는 8이다.

## V. 결 론

본 논문에서는 일반적인 CA-CFAR 프로세서를 고속으로 동작하며 간단한 구조를 가지는 하드웨어 아키텍처를 설계하였다. 제안된 하드웨어 아키텍처에서는 제

곱평균 연산에 근사화 기법을 도입하여 속도를 향상시켰으며, 평균연산과 소수점 곱셈 연산에 고정-소수점 방식을 도입시켜 구현이 간단하도록 하였다. 또한 다중 윈도우 방식을 이용하여 병렬 구조를 가지게 함으로써 연산속도를 향상시킬 수 있는 방법도 제안하였다.

제안된 하드웨어는 향후 레이더 신호처리 알고리즘을 구현함에 있어서 효과적으로 활용 될 수 있을 것이다.

## 참 고 문 헌

[1] 현유진, 오우진, 이종훈, “FMCW 차량용 레이더의 이동타겟 탐지 알고리즘 제안”, 대한전자공학회 논문제 제47권 SC편 제6호, 27-32쪽, 2010년 11월

[2] 김상동, 현유진, 이종훈, 최준혁, 박정호, 박상현, “차량용 FMCW 레이더의 탐지 성능 분석 및 신호처리부 개발”, 한국해양정보통신학회논문지 제14권 12호, pp. 2628-2635, 2010년 12월

[3] Rohling H., : ‘Radar CFAR Thresholding in Clutter and Multiple Target Situations’, IEEE Transaction on Aerospace and Electronics System, Vol. AES-19, No. 4, pp. 608-620, July, 1983

[4] Mark A. Richards : ‘Fundamentals of Radar Signal Processing’, MCGraw-Hill, pp. 347-383

[5] By Richard, G. Lyons, “Digital Signal Processing Tricks - High-speed vector magnitude approximation”, www.embedded.com, October. 2007

## 저 자 소 개

**현 유 진**(정회원)

1999년 영남대학교 전자공학과 공학사

2001년 영남대학교 전자공학과 공학석사

2005년 영남대학교 전자공학과 공학박사

2005년～현재 대구경북과학기술원(DGIST) 로봇시스템연구부 선임연구원.

2007년～현재 영남대학교 대학원 전자공학과 겸임교수.

<주관심분야 : 레이더 디지털 신호처리, 디지털 신호처리 프로세서 구현, 레이더 체계 공학>

**오 우 진**(정회원)

1989년 한양대학교 전자공학과 공학사

1991년 한국과학기술원 전자공학과 공학석사

1996년 한국과학기술원 전자공학과 공학박사

1996년～1998년, SK 텔레콤 선임 연구원.

1998년～현재 금오공과대학교 전자공학부 교수.

2008년～현재, 대구경북과학기술원(DGIST) 로봇시템연구부 겸임연구원

<주관심분야 : 레이더 신호처리, 통신 신호처리>

**이 종 훈**(정회원)

1996년 성균관대학교 전자공학과 공학사

1998년 성균관대학교 전기전자컴퓨터공학과 공학석사

2002년 성균관대학교 전기전자컴퓨터공학과 공학박사

2002년～2005년 삼성전자 통신연구소 책임연구원

2005년～현재 대구경북과학기술원(DGIST) 로봇시스템연구부 선임연구원(과제책임자)

2007년～현재 영남대학교 대학원 정보통신공학과 겸임교수

<주관심분야 : 레이더 신호처리, FMCW/UWB 레이더, 레이더 체계공학>