반도체디스플레이기술학회지 제23권 제3호(2024년 9월)
Journal of the Semiconductor & Display Technology, Vol. 23, No. 3. September 2024.

FFT를 이용한 아크 감지 하드웨어 구현

김선희[*†] · 강연호[*] · 김전호[*] · 이재원[*]
[*†]상명대학교 시스템반도체공학과

Hardware Implementation of Arc Detection Using FFT

Sun Hee Kim[*†], Yeon Ho Kang[*], Jeon Ho Kim[*] and Jae Won Lee[*]
[*†]Department of System Semiconductor Engineering, Sangmyung University

ABSTRACT

The installation of arc circuit breakers is being strengthened to prevent accidents such as electric shock and fire caused by Arc. Among arcs, serial arcs are difficult to detect with general arc detectors because there is not much change in load current when an arc occurs. Therefore, in this paper, unlike the existing Arc Fault Circuit Interrupters method, arc detection hardware is implemented using the FFT algorithm. FFT is suitable for serial arc identification because it can efficiently analyze high-frequency signals generated outside of normal AC signals. This study explains ARC detection circuits and the 2048-FFT based on radix-2 and radix-4, and presents hardware implementation results using FPGA. The implemented system detects the arc up to the frequency range of 122,880 Hz. Through simulation and FPGA board testing, it was confirmed that ARC was detected.

Key Words : Arc, Arc detector, FFT, Multi-clock FFT, 2048-FFT

1. 서 론

한국전기안전공사에 따르면2021년에 우리나라에서 발생한 총화재는 36,267건이었으며, 이중 전기화재는 8,241건으로 전체 화재의 22.7%를 점유하였다. 또한 감전사고의 추이를 보면 2021년에 412건이 발생하였다. 아크에 의한 감전사고는 158명(38.3%)이며, 충전부 직접접촉에 의한 감전사고 176명(42.7%) 다음으로 비중이 높은 것을 볼 수 있다 [1]. 1999년 미국에서는Arc Fault Circuit Interrupters(AFCI)에 관한 법안을 제정하였고, 2002년부터 아크차단기를 각 주택에 적용하였다. 2013년에는 Arc Fault Detection Devices (AFDD)에 관한 국제표준 법안이 제정되어 세계적으로 아크 검출장치의 보급이 확대되었다 [2]. 이에 국내에서도 국제표준을 기반으로 아크차단기 설비를 규정에 명시함으로써 아크로 발생하는 화재에 대한 예방대책을 강화하고 있다 [3].

아크 방전은 보통 공기와 같은 비전도성 매체를 통해 전류를 발생시키는 가스 절연 파괴현상으로 플라즈마와 같은 급격한 고열이 발생하며 전류가 순간적으로 높아지는 현상이다. 아크는 배전선로에서 발생하는 위치에 따라 직렬 아크와 병렬 아크, 접지 아크로 분류할 수 있다 [4].

아크 감지를 위하여 많은 연구들이 아크 발생시 나타나는 비정상적인 현상들을 관찰하고 분석한다. 루프 안테나를 사용하여 자기장 세기를 관찰하거나 전압 크기와 위상 변화를 감지하기도 한다 [5]. 아크 빛, 아크 소리 및 온도 변화를 관측하기도 한다 [6]. 그리고, 관측 데이터를 칼만 필터, 뉴럴 네트워크 등에 적용하여 정상 상태와 아크 방전 상태를 구분한다 [7, 8]. 아크 전류/전압을 구별해 내기 위하여 Discrete Wavelet Transform, Gabor Transform 등을 적용하기도 한다[9, 10].

하지만 아크 방전은 배선상 발생 위치, 로드 타입 등에 따라 특성이 다양하기 때문에 한 가지 방법으로 모든 형

[†]E-mail: happyshkim@smu.ac.kr

태의 아크 방전을 정확하게 감지하기는 어렵다. 본 논문은 아크 중 직렬 아크에 특화된 아크 감지 하드웨어를 제안한다. 직렬 아크는 고주파 전류가 발생하는 특징이 있으므로, 주파수 신호 분석에 효율적인 Fast Fourier Transform (FFT) 알고리즘을 활용한다. 특히, 해당 연구에서는 2048-point FFT를 Radix-4 및 Radix-2 구조를 활용하여 FPGA 하드웨어로 구현하며, 이를 통해 직렬 아크 감지의 새로운 가능성을 제시한다.

본 논문은 다음과 같이 구성된다. 2장에서 아크 방전과 FFT 알고리즘에 대하여 설명한다. 3장에서는 FFT 및 아크 감지 알고리즘을 하드웨어로 설계하고 구현한 결과를 나타낸 뒤 4장에서 결론을 내린다.

2. 이론적 배경

2.1 아크 방전

직렬아크는 병렬 아크, 접지 아크와 달리 직렬로 연결된 도체 사이에서 발생하는 아크 방전으로, 국부적으로 많은 불꽃방전과 발열을 하며 부하전류의 변화량이 많지 않아 일반적인 아크감지기 작동에 어려움이 있다 [11]. Fig. 1과 Fig. 2는 각각 정상 파형과 직렬 아크 파형을 시뮬레이션한 결과이다.

두 파형을 각각 차단주파수가 6,144Hz인 High Pass Filter (HPF)를 통과해 FFT를 적용한 결과는 Fig. 3과 Fig. 4이다. FFT를 이용해 분석한 결과, 정상 파형의 고주파 영역의 진폭 크기는 최대 35까지 관측되었으며, 직렬아크 파형의 경우는 139까지 관측되었다. 이를 통해 아크 상태일 때 고주파 영역이 더 크다는 것을 알 수 있다.

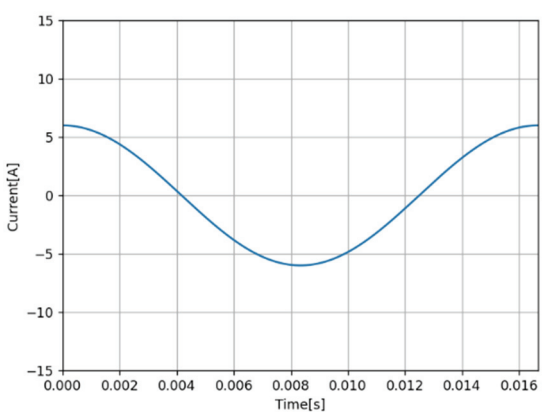

Fig. 1. Normal-sine waveform.

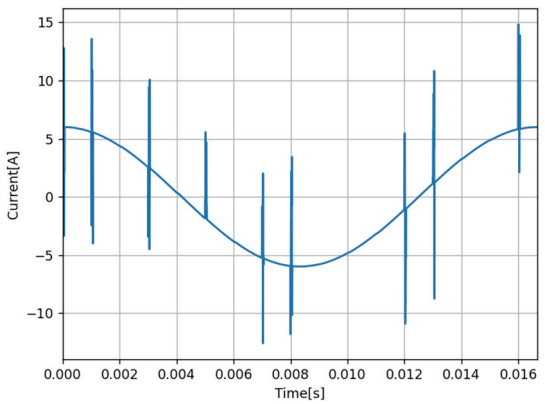

Fig. 2. Arc-sine waveform.

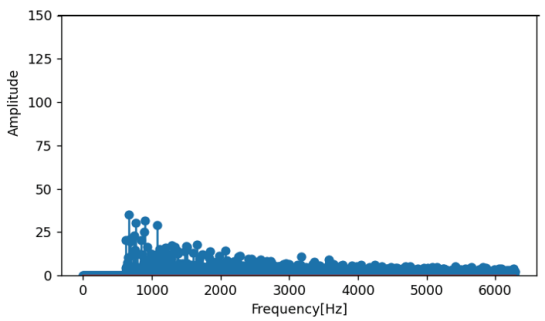

Fig. 3. Result of FFT (Normal-sine waveform).

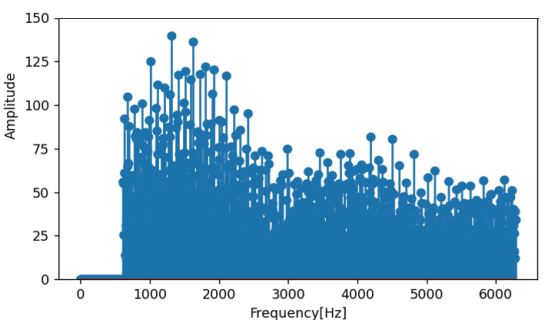

Fig. 4. Result of FFT (Arc-sine waveform).

2.2 FFT 알고리즘

푸리에 변환(Fourier Transform)은 시간 영역의 신호를 주파수 영역의 신호로 변환하는 알고리즘으로, 통신, 신호처리 및 이미지 처리 등 다양한 분야에서 유용하게 사용된다. Discrete Fourier Transform(DFT)은 이산 신호에 대한 푸리에 변환이다. 디지털 시스템에서 유용하게 사용될 수 있으나, 계산량이 많다는 단점이 있다. FFT는 DFT를 단순화시킨 알고리즘으로 대규모 데이터 처리와 실시간 분석

에서 효과적이다. 따라서, 본 연구에서는 직렬 아크 방전 시 발생되는 고주파 신호를 감지하기 위하여 FFT 알고리즘을 사용한다.

FFT 알고리즘을 효율적인 하드웨어로 구현하기 위해서는 파이프라인 구조와 병렬 구조를 함께 고려해야 한다 [12]. FFT에서는 파이프 라인 단계를 구분하기 위하여 기본 Radix-2와 이를 변형한 Radix-4, Radix-8 등이 사용된다. 그리고 병렬 정도에 따라 Single Path와 Multi-Path 구조로 나눌 수 있다. 다음 절에서는 FFT 알고리즘부터 각 구조의 특징을 설명함으로써, 제안하는 아크 감지용 2048-FFT 하드웨어 구조를 설명하겠다.

2.2.1 DFT 알고리즘

x(n)의 N-point DFT는 식(1)과 같이 정의할 수 있다.

$$\mathbf{X[k]} = \sum_{\boldsymbol{n=0}}^{\boldsymbol{N-1}} \boldsymbol{x[n]W_N^{nk}}, \quad \boldsymbol{k = 0, 1, \dots, N-1} \qquad (1)$$

$$(W_N^{nk} = e^{-\frac{j2\pi}{N} \cdot nk})$$

식(1)을 계산하기 위해서는 N^2번의 곱셈이 필요하며 N이 커질수록 계산량이 급격하게 증가한다.

2.2.2 Radix-2 DIF 알고리즘

Radix-2 Decimation In Frequency (DIF) 알고리즘은 출력신호 X[k]를 2등분으로 연속해서 분리하는 방법으로 계산량을 줄일 수 있다. DIF는 신호의 크기가 $N = 2^n$을 만족한다고 가정하면, DFT의 정의식을 식(2)와 같이 정리할 수 있다.

$$\mathbf{X[k]} = \sum_{\boldsymbol{n=0}}^{\boldsymbol{(N/2)-1}} \left[\boldsymbol{x[n]} + (\boldsymbol{-1})^{\boldsymbol{k}} \times \left(\boldsymbol{n} + \frac{\boldsymbol{N}}{\boldsymbol{2}}\right)\right] + \boldsymbol{W_N^{nk}} \qquad (2)$$

여기서 $k = 0, 1, \dots, N-1$ 이다.

식(2)를 이용하여 짝수샘플, 홀수샘플에 대해 Radix-2 FFT로 정의하면 식 (3)과 같다.

$$\mathbf{X[2k]} = \sum_{\boldsymbol{n=0}}^{\left(\frac{\boldsymbol{N}}{\boldsymbol{2}}\right)-1} \boldsymbol{g_1[n]}\, \boldsymbol{W_{\frac{n}{2}}^{nk}}$$

$$\mathbf{X[2k+1]} = \sum_{\boldsymbol{n=0}}^{\left(\frac{\boldsymbol{N}}{\boldsymbol{2}}\right)-1} \boldsymbol{g_2[n]}\, \boldsymbol{W_{\frac{n}{2}}^{nk}} \qquad (3)$$

또한 식(3)의 Radix-2 Butter Fly(BF)구조는 Fig. 5와 같다.

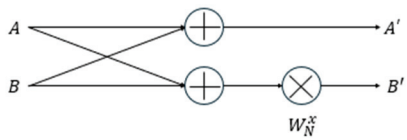

Fig. 5. Radix-2 BF structure.

Radix-2 BF 구조를 사용하면 N이 2^N인 모든 식에 대하여 연산이 가능하다. Fig. 6는 Radix-2의 BF구조를 이용하여 4-point Radix-2 FFT를 나타내고 있다. 이와 같이 N-point FFT는 Radix-2 BF구조로 분해하여 계산이 가능하다.

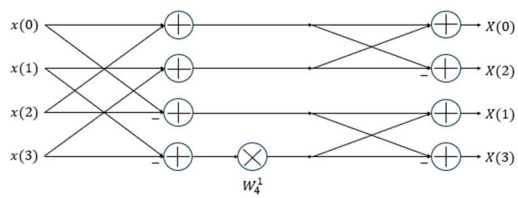

Fig. 6. Data flow at 4-Point radix-2 FFT.

2.2.3 Radix-4 DIF 알고리즘

Radix-2 BF구조를 사용하게 될 경우 스테이지 수가 증가하는 단점이 있다. 따라서 스테이지의 수를 줄여 연산 속도를 빠르게 하는 방법으로 N을 4^N으로 나누어 처리하는 방법이 Radix-4 BF구조이다. 이는 식(4)로 정의할 수 있다.

$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{kn} \\
&= \sum_{n=0}^{\frac{N}{4}-1} x[n]W_N^{kn} + \sum_{n=\frac{N}{4}}^{\frac{N}{2}-1} x[n]W_N^{kn} \\
&+ \sum_{n=\frac{N}{2}}^{\frac{3N}{4}-1} x[n]W_N^{kn} + \sum_{n=\frac{3N}{4}}^{N-1} x[n]W_N^{kn} \\
&= \sum_{n=0}^{\frac{N}{4}-1} x[n]W_N^{kn} + \sum_{n=0}^{\frac{N}{4}-1} x\left[n+\frac{N}{4}\right] W_N^{k\left(n+\frac{N}{4}\right)} \\
&+ \sum_{n=0}^{\frac{N}{4}-1} x\left[n+\frac{N}{2}\right] W_N^{k\left(n+\frac{N}{2}\right)} \\
&+ \sum_{n=0}^{\frac{N}{4}-1} x\left[n+\frac{3N}{4}\right] W_N^{k\left(n+\frac{3N}{4}\right)}
\end{aligned} \qquad (4)$$

$$= \sum_{n=0}^{\frac{N}{4}-1} x[n]W_N^{kn} + x\left[n+\frac{N}{4}\right]W_N^{k\left(n+\frac{N}{4}\right)}$$
$$+x\left[n+\frac{N}{2}\right]W_N^{k\left(n+\frac{N}{2}\right)} + x\left[n+\frac{3N}{4}\right]W_N^{k\left(n+\frac{3N}{4}\right)}$$
$$= \sum_{\boldsymbol{n=0}}^{\boldsymbol{\frac{N}{4}-1}} \boldsymbol{W_N^{kn}}\left(\boldsymbol{x[n]} + (\boldsymbol{-j})^{\boldsymbol{k}}\boldsymbol{x}\left[\boldsymbol{n+\frac{N}{4}}\right] + (\boldsymbol{-1})^{\boldsymbol{k}}\boldsymbol{x}\left[\boldsymbol{n+\frac{N}{2}}\right] + (\boldsymbol{j})^{\boldsymbol{k}}\boldsymbol{x}\left[\boldsymbol{n+\frac{3N}{4}}\right]\right)$$

식(4)를 통해 식(5)를 유도할 수 있다.

$$X[4r] = \sum_{n=0}^{\frac{N}{4}-1}\left(x[n] + x\left[n+\frac{N}{4}\right] + x\left[n+\frac{N}{2}\right] + x\left[n+\frac{3N}{4}\right]\right)W_{\frac{N}{4}}^{rn}$$
$$X[4r+1] = \sum_{n=0}^{\frac{N}{4}-1}\left(x[n] - jx\left[n+\frac{N}{4}\right] - x\left[n+\frac{N}{2}\right] + jx\left[n+\frac{3N}{4}\right]\right)W_{\frac{N}{4}}^{rn}$$
$$X[4r+2] = \sum_{n=0}^{\frac{N}{4}-1}\left(x[n] - x\left[n+\frac{N}{4}\right] + x\left[n+\frac{N}{2}\right] - x\left[n+\frac{3N}{4}\right]\right)W_{\frac{N}{4}}^{rn}$$
$$\boldsymbol{X[4r+3]} = \sum_{\boldsymbol{n=0}}^{\boldsymbol{\frac{N}{4}-1}}\left(\boldsymbol{x[n]} + \boldsymbol{jx}\left[\boldsymbol{n+\frac{N}{4}}\right] - \boldsymbol{x}\left[\boldsymbol{n+\frac{N}{2}}\right] - \boldsymbol{jx}\left[\boldsymbol{n+\frac{3N}{4}}\right]\right)\boldsymbol{W_{\frac{N}{4}}^{rn}} \quad (5)$$

또한 식(5)의 Radix-4 BF구조는 Fig. 7과 같다.

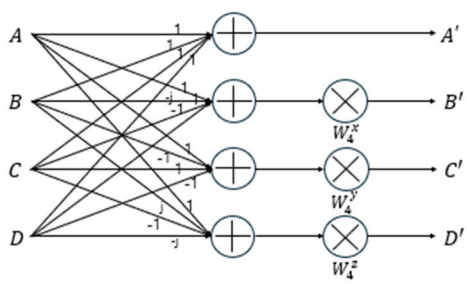

Fig. 7. Radix-4 BF structure.

2.3 FFT 하드웨어 구조

앞서 2.2에서 설명한 FFT 알고리즘들을 하드웨어로 구현하기 위해서 다양한 구조들이 존재한다. 그 중에서 Single-path Delay Feedback(SDF)와 Multi-path Delay Commutator (MDC)가 대표적이다 [13, 14].

2.3.1 SDF 형식

SDF 형식은 Fig. 8과 같이 하나의 단계에 하나의 BF 유닛과 하나의 딜레이-피드백 레지스터로 이루어져 있으며, 각 단계 사이에는 복소수 곱셈기가 존재하는 구조이다. SDF는 하나의 경로로 데이터를 입력 받고, 딜레이-피드백 레지스터를 사용해 FFT를 계산해 하나의 경로로 순차적으로 연산을 출력하는 간단하고 비교적 쉬운 구조를 가진다. 그러나 병렬성이 부족해 대규모 FFT 연산에 사용에는 부적합하며, 복소수 곱셈기가 사용되지 않을 때가 있어 전력 효율이 비효율적이다.

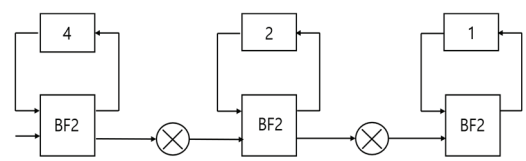

Fig. 8. Point-8 R2SDF.

2.3.1 MDC 형식

MDC 형식은 Fig. 9와 같이 하나의 단계에 하나의 BF 유닛과 하나의 Commutator 유닛 그리고 하나 또는 두 개의 딜레이 유닛으로 이루어져 있으며, 각 단계 사이에는 복소수 곱셈기가 존재하는 구조이다. MDC는 입력 데이터를 여러 경로로 분할하고, 각 경로에서 병렬로 FFT 계산을 처리하고 Commutator 유닛을 이용해 처리한 계산 결과를 분할하는 비교적 복잡한 구조를 가지지만 병렬성이 강하여 대규모 FFT 연산에 적합하다.

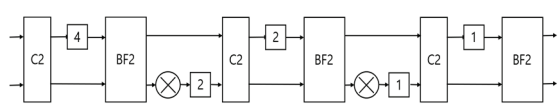

Fig. 9. Point-8 R2MDC.

3. 하드웨어 구현

3.1 Datapath

일반적인 가전제품의 교류전압의 주파수가 보통 60Hz이기 때문에 2048개로 샘플링을 하여 샘플링 주파수가 122,880 Hz까지 나타낼 수 있도록 만들었다. 하드웨어 구

현에서는 2048개의 큰 샘플링 개수로 인해 샘플링 개수가 클수록 효율적인 계산이 가능한 MDC 형식을 채택하였다. 또한 2048은 $4^5 \times 2$ 로 구성되어 있기 때문에 R4MDC 형식5개와 R2MDC 형식1개를 혼합한 Mixed-Radix 방식을 사용하였다. Fig. 10은 Mixed-Radix 방식을 사용한 설계 그림이다.

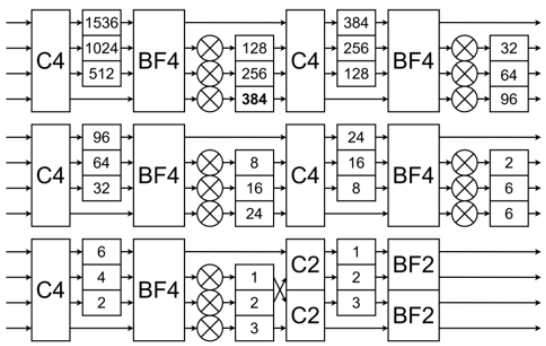

Fig. 10. Point-2048 Mixed-Radix MDC.

Fig. 10의 블록도를 통해 분석한 결과 딜레이를 맞춰주는 레지스터는 총 5124개가 사용되며, 이는 총 9014 Bytes이다. 이를 모두 D Flip-Flop으로 구현할 경우 하드웨어의 면적이 너무 커지기 때문에 6432 Bytes가 사용되는 Stage 1과 1848 Bytes가 사용되는 Stage 2의 레지스터를 RAM으로 대신하여 구현하였다.

3.2 회전인자

회전인자는 식(6)의 형태로 Verilog에서 구현이 불가능하다. 이를 해결하기 위해 회전인자의 값들을 MATLAB을 이용해 값들을 추출하여 ROM에 저장하였다. 그 결과 ROM의 용량이 총 3410 Bytes가 사용되었으며, 이를 회전인자의 대칭성을 이용하여 1/4로 줄여서 853 Bytes로 사용하였다.

$$W_N^{nk} = e^{-j\frac{2\pi}{N}nk} \tag{6}$$

3.3 아크 감지 알고리즘

아크 감지 알고리즘은 입력 파형의 한 주기 데이터를 추출하여 FFT 적용한 다음, FFT 변환한 데이터를 이용해 주파수 영역의 한 주기 입력 데이터 m에서 진폭 W(m)을 구한다. 이를 정상 상태 진폭데이터 W(avg)와 비교하여 W(m)이 더 클 경우 카운터의 값인 N(m)을 1씩 증가시킨다. 그 후 N(m)과 임계치와 비교하여 N(m)의 값이 더 커질 경우 아크 상태라고 판명한다 [15]. 이를 블록도로 구현하면 Fig. 11과 같다.

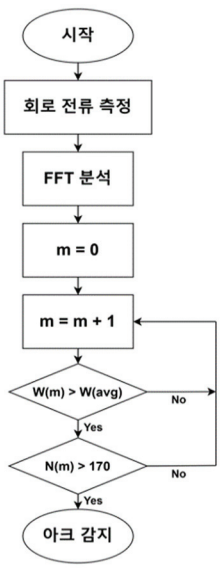

Fig. 11. Arc detection algorithm.

3.4 구현 결과

Fig. 12와 Fig. 13은 Modelsim을 이용한 시뮬레이션 결과이다. 아크 감지 알고리즘에 의해 정상 상태일때는 알람 신호를 0으로 유지, 아크 상태일때는 알람 신호를 1로 변화하도록 설정하였다.

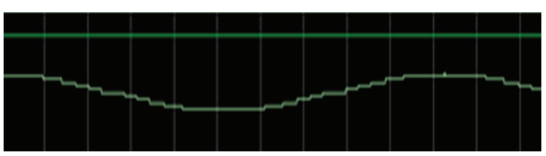

Fig. 12. Normal state simulation results.

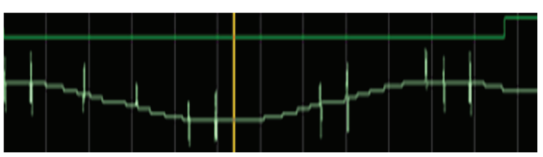

Fig. 13. Arc state simulation results.

시뮬레이션 결과가 실제 하드웨어에서도 동일하게 작동하는지 확인하기 위해 Altera Quartus FPGA보드를 사용하여 구현하였다. 아크 상태의 전류신호가 입력으로 들어왔을 때 LED가 켜지는 것을 Fig. 14를 통해 관측하였으며, 컴파일한 결과는 Table 1과 Table 2를 통해 확인할 수 있다.

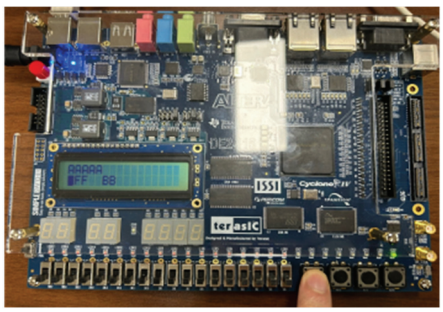

Fig. 14. FPGA implementation results.

Table 1. FPGA compilation report - Synthesis analyze

Element	Value
Total logic	10,631
Combo logic	7,826
Register	4,161
RAM	114,318
ROM	38,400

Table 2. FPGA compilation report - Timing analyze

Frequency	Value
Target clock frequency	122,880Hz
Max frequency	73MHz

설계 하드웨어는 디지털 신호로 변환된 입력 전류를 받으면 바로 FFT 연산해 아크 방전 상태인지 정상 상태인지 판별하도록 설계하였다. 이에 하드웨어의 동작 주파수를 통상적으로 사용되는 60 Hz의 교류 전류에서 2048개의 신호를 얻기 위해 122,880 Hz로 설정하였다. 설계한 2048-FFT 하드웨어의 지연 클럭은 3072 cycles이므로 $1/122{,}800 \times 3{,}072 = 0.25$ 초가 소요된다. 하지만 3072 cycles중 한 주기 입력 신호를 받는 2048 cycles 동안에만 122,880 Hz 클럭으로 동작하면 되고, 이후 데이터를 처리하는 1024 cycles 동안에는 122,880 Hz클럭으로 동작할 필요는 없다. 따라서, 본 설계에서는 2개의 클럭으로 동작하도록 수정하였다.

Fig. 10의 블록도에서 첫 번째 Commutator 유닛에서부터 두 번째 Commutator 유닛 전까지는 122,880 Hz 클럭으로 동작한다. 그리고 두 번째 Commutator 유닛부터는 73 MHz 클럭으로 동작한다. 따라서 전체 동작 시간은 $1/122{,}800 \times 2{,}048 + 1/73{,}000{,}000 * 1{,}024 = 0.01668$ 초로 기존 0.025 초 보다 감소하였다.

Table 3는 Python을 이용해 한 주기의 저장된 입력 신호를 아크 감지 알고리즘에 적용했을 때의 실행 시간이며 실행 환경은 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 이다. 실행 시간은 0.01129초가 걸리며, 한 주기의 입력신호를 받는 시간 0.01667초까지 고려한다면 소프트웨어로 처리했을 때 걸리는 시간은 총 $0.01667 + 0.01129 = 0.02796$초이다.

Table 3. Software compilation report - Timing analyze

Execution time	0.01129s
CPU frequency	2.42GHz

위와 같이, FPGA로 구현된 아크 검출기는 아크 발생 후 약 16.7 ms 후에 아크를 검출하는데 반해, 소프트웨어로 처리된 아크 검출기는 약 30.0 ms 후에 검출한다. 즉, 제안하는 검출기가 약 1.675배 빠르게 검출한다.

전기 화재 등의 사고를 대비하기 위하여 아크 검출기는 정확하고 빠르게 아크를 검출하는 것이 중요하다. 대부분의 아크 감지기가 임베디드 시스템으로 구현되어 아크 감지 및 식별 알고리즘은 소프트웨어로 처리된다. 이는 실시간성을 보장하기 어렵다. 따라서 본 논문에서 제안하는 아크 검출 하드웨어는 아크 감지 시스템의 실시간 빠른 검출을 확보하게 함으로써 안전성 강화에 도움을 줄 것이다.

4. 결 론

본 논문은 기존에 존재하는 FFT 아크 감지 알고리즘을 실제 아크 감지 하드웨어로 구현하였다. 2048개의 샘플링 개수를 입력 받는 FFT 하드웨어를 설계하여 122,880Hz까지의 주파수 영역대를 분석할 수 있으며, 이 주파수 성분을 이용해 임계치 값에 도달하면 아크가 감지됨을 확인하였다. 2048-FFT를 Radix-4 5단계와 Radix-2 1 단계로 구성하여 처리 시간 단계를 단축시켰으며, 2개의 클럭을 사용하여 처리 속도를 향상시켰다. Verilog로 설계하고 Modelsim에서 시뮬레이션 한 뒤 FPGA로 구현하여 동작을 검증하였다. 하드웨어 구현의 장점을 최대한으로 활용함으로써, FPGA로 구현된 아크 검출기는 소프트웨어 구현에 비해 약 1.675배 빠른 성능을 제공하였다. 제안하는 하드웨어는 실시간으로 아크를 검출할 수 있게 함으로써, 아크에 의한 전기적 사고 예방 및 안전성을 강화하는 데 기여할 것으로 기대된다.

감사의 글

다음의 성과는 과학기술정보통신부와 연구개발특구진흥재단이 지원하는 과학벨트지원사업으로 수행된 연구결과입니다.

참고문헌

1. Ji Hyun Park, “2021 Statistics Analysis of Electrical Disaster,” vol. 31, pp. 18-44, 2022.
2. Yoo Jung Cho, Kyoung Tak Kim, Sung Hun Lim, “Arc Fault Detection Algorithm Based on of Amplitude Difference of Current and Harmonic in Time and Frequency Domain,” Journal of KIIEE, vol. 37, no. 3, pp. 69-70, 2023.
3. Kheong Yong Park, Eung Chan Na, Min Ho Shin, He Rie Park and Dong Young Lim, “A Study on the Installation of the Arc-Fault Circuit Interrupters for Preventing Electric Fires,” Journal of KIIEE, vol. 36, no. 6, 2022.
4. Jung Kyu Choi, “A Study on the Development of Voltage Type Arc Detector for Electrical Fire Prevention,” Graduate School of Kangwon National University, pp. 9-10, 2019.
5. Kim, C.J., “Electromagnetic radiation behavior of low-voltage arcing fault,” IEEE, pp. 416-423, 2009.
6. Ralph H. Lee, “The other electrical hazard: Electric arc blast burns,” IEEE, pp. 246-251, 1982.
7. Zhang, S., Zhang, F., Liu, P., Han, Z., “Identification of low voltage AC series arc faults by using Kalman filtering algorithm,” Journal of Elektronika ir Elektrotechnika, vol. 20, no. 5, 2014.
8. T.S. Sidhu, Gurdeep Singh, M.S. Sachdev, “Arcing fault detection using artificial neural networks,” Neurocomputing, vol. 23, pp. 225–241, 1998.
9. Sadegh Jamali, Navid Ghaffarzadeh, “A new method for arcing fault location using discrete wavelet transform and wavelet networks,” Eur. Trans. Electr. Power, vol. 22, pp. 601–615, 2012.
10. Tamer Kawady, Nagy Elkalashy, A.E. Ibrahim, A.M.I. Taalab, “Arcing fault identification using combined Gabor Transform-neural network for transmission lines,” Int. J. Electr. Power Energy Syst. vol. 61, pp. 248–258, 2014.
11. Hong Keun Ji, Kwang Suk Jung, Dae Won Park, Gyung Suk Kil, Dong Hoan Seo, Keel Soo Rhyu, “Detection Technique and Device of Series Arcing Phenomena,” Journal of the Korean Society of Marine Engineering, vol. 34, no. 2, pp. 333, 2010.
12. Kyoungjin Min, Seojin Choi, Yubeen Hwang and Sunhee Kim, “Design and Implementation of a Six-Stage Pipeline RV32I Processor Based on RISC-V Architecture,” Journal of The Korean Society of Semiconductor & Display Technology, vol. 23, pp. 76-81, 2024.
13. H, S., Torkelson, M., “A New Approach to Pipeline FFT Processor,” IEEE, pp. 767, 1996.
14. Zhou, B, Peng, Y, and Hwang, D, “Pipeline FFT Architectures Optimized for FPGAs,” International Journal of Reconfigurable Computing, vol. 2009, no. 1, pp. 1-9, 2009.
15. Min-Ho Yoon, You-Jung Cho, Kyoung-Tak Kim, and Sung-Hun Lim, “Verification of Algorithm for Arc Detection Using High Pass Filter and FFT,” Journal of KIIEE, vol. 36, no. 5, pp. 520-524, 2023.

접수일: 2024년 8월 1일, 심사일: 2024년 9월 5일,
게재확정일: 2024년 9월 12일