이영우 교수님 전기전자종합설계,

인공지능반도체 설계프로젝트 교과목

수강인원 제외 공유 금지

학사학위 논문

지도교수 이영우

# <STRAIT 구조의 한계를 개선한 병렬 BIST 구조 : BICS-BIST 제안 및 성능 비교>

인하대학교 전기전자공학부

김가현 장준영

2025 년 12 월

2025 년 12 월

지도교수 이 영 우 (인)

인하대학교 전기전자공학

# 목 차

# 그 림 목 차

# 표 목 차

# <STRAIT 구조의 한계를 개선한 병렬 BIST 구조 : BICS-BIST 제안 및 성능 비교>

김가현 장준영

인하대학교 전기전자공학부

(지도교수 : 이영우)

(국문초록)

# 1. 서론

## 1.1 연구 배경

인공지능(AI) 모델의 성능이 지속적으로 향상됨에 따라, 대규모 행렬 곱셈 연산을 고속으로 처리할 수 있는 하드웨어 가속기의 수요가 급증하고 있다. 이러한 요구에 대응하기 위해 개발된 Systolic Array 구조는, 높은 연산 병렬성과 데이터 재사용 효율을 바탕으로 다양한 AI 하드웨어 시스템에서 핵심 연산 블록으로 채택되고 있다. 대표적인 예로 Google TPU(Tensor Processing Unit)는 Systolic Array 기반의 행렬 곱셈 유닛을 활용하여 딥러닝 연산을 효율적으로 수행한다.

하지만 Systolic Array는 수많은 Processing Element(PE)로 구성되어 있어, 하나의 PE에 발생한 결함이 전체 연산 결과에 심각한 영향을 미칠 수 있다. 특히 각 PE가 파이프라인처럼 연산 흐름을 구성하고 있어, 특정 PE의 오류는 다음 연산 단계로 전파되며 최종 출력에 오류를 유발한다. 이에 따라, 하드웨어 가속기의 신뢰성을 보장하기 위해 각 구성 요소의 정상 동작 여부를 시스템 내부에서 스스로 검증할 수 있는 Built-In Self-Test(BIST) 기술의 중요성이 부각되고 있다.

대표적인 구조로는 STRAIT(Scan Test and Repair Architecture for AI Tensor Core)가 있다. STRAIT는 각 PE 내부에 시프트 레지스터(Shift Register)와 캡처 레지스터(Capture Register)를 포함한 Scan Chain을 구성하여, 테스트 벡터를 시리얼로 주입(shift-in)하고, 연산 결과를 캡처한 후, 시리얼로 출력(scan-out)하면서 기대값과 비교하는 구조다. 이러한 방식은 기존 DFT(Design-for-Testability) 기법을 Systolic Array 구조에 효과적으로 적용할 수 있다는 장점이 있지만, Scan Shift 과정에 많은 클럭이 소요되며, 병렬 입력이 불가능하다는 구조적 제약으로 인해 테스트 시간이 배열 크기에 따라 선형적으로 증가하는 단점이 있다.

본 연구에서는 STRAIT 구조의 시간 소모와 구조적 제약을 개선하기 위해, 테스트 데이터를 각 PE에 병렬로 주입하고, 연산 결과를 병렬로 비교하는

BICS-BIST(Broadcast-Input Comparison-Scan BIST) 구조를 제안한다. 제안하는 구조는 전체 테스트 시간을 줄이는 동시에 설계 복잡도와 회로 자원 사용량을 줄이는 것을 목표로 한다. 특히 본 연구에서는 메모리 셀 자체의 결함 검출을 다루는 Memory BIST(MBIST)는 구현 대상에서 제외하고, Systolic Array의 연산 로직에 대한 Logic BIST(LBIST)만을 중점적으로 구현하였다.

## 1.2 연구 목적 및 구성

본 연구의 목적은 기존 STRAIT 구조의 한계를 극복할 수 있는 새로운 BIST 아키텍처인 BICS-BIST를 제안하고, 이를 RTL 수준에서 설계 및 시뮬레이션하여 성능을 정량적으로 비교 분석하는 데 있다. STRAIT 구조는 테스트 데이터를 시리얼 방식으로 전달하는 특성상, Systolic Array의 크기가 증가할수록 테스트에 필요한 클럭 수와 자원이 선형적으로 증가하는 문제를 안고 있다. 특히 모든 PE에 동일한 테스트 벡터를 전달해야 하는 경우에도, 데이터를 순차적으로 이동시켜야 하기 때문에 테스트 시간 최적화에 제약이 존재한다. 이러한 문제를 해결하기 위해 본 연구에서는 병렬 입력 기반의 BIST 구조인 BICS-BIST(Broadcast-Input Comparison-Scan BIST)를 고안하였다. 이 구조는 테스트 데이터를 각 PE에 동시에 병렬로 주입한 후, capture 및 compare 과정을 거치는 방식으로 테스트를 수행하여, 전체 테스트 시간을 줄이고 설계 복잡도를 낮추는 것을 목표로 한다.

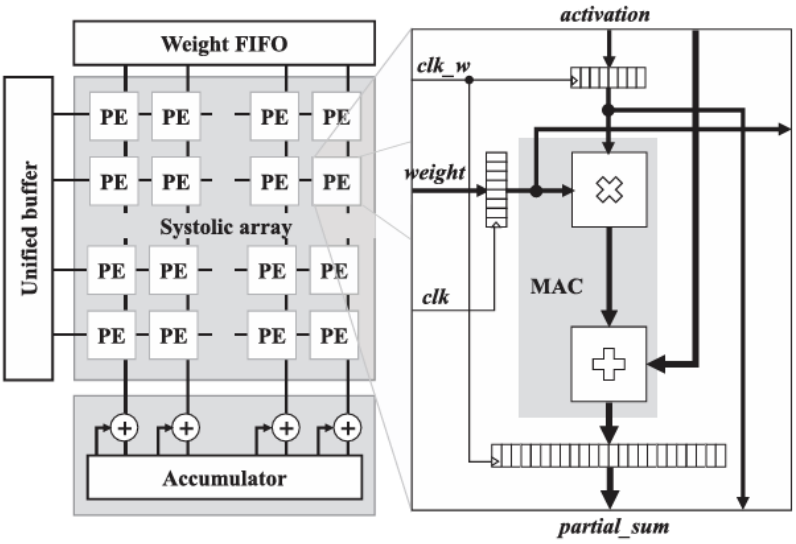
본 보고서는 총 여섯 개의 장으로 구성된다. 제 1장에서는 본 연구의 배경 및 목적을 서술하고, 제 2장에서는 Systolic Array의 기본 동작 원리와 기존 STRAIT 구조의 구성 및 한계점을 기술한다. 이어지는 제 3장에서는 제안하는 BICS-BIST 구조의 설계 개요, 주요 모듈의 기능 및 전체 동작 방식을 상세히 설명한다. 제 4장에서는 STRAIT 구조와 BICS-BIST 구조를 동일한 조건 하에서 RTL로 구현하고 시뮬레이션한 결과를 비교하여, 테스트 시간과 회로 효율 측면에서의 성능 분석을 수행한다. 마지막으로 제 5장에서는 실험 결과에 대한 종합적인 해석 및 트레이드오프 분석, 제안 구조의 장단점, 그리고 향후

연구 방향을 고찰하며 본 연구의 결론을 맺는다.

# 2. 기존 구조 및 관련 연구

## 2.1 Systolic Array 구조

Systolic Array는 규칙적인 격자 형태로 구성된 연산 유닛(Processing Element, 이하 PE)들이 시간의 흐름에 따라 일정한 방향으로 데이터를 흐르게 하면서 연산을 수행하는 구조이다. 이 구조는 1979년 H. T. Kung에 의해 처음 제안되었으며, 이후 고속 행렬 연산, 신호처리, 딥러닝 가속기 등 다양한 분야에서 널리 활용되고 있다. 특히, 데이터 병렬성과 지역성(Locality)을 동시에 만족할 수 있다는 점에서 하드웨어 효율이 뛰어나고, 연산 처리 속도와 에너지 효율 측면에서도 매우 우수한 구조로 평가받고 있다.




<그림 1> 일반적인 Systolic Array-based Ai 가속기 구조 [2]

Systolic Array는 일반적으로 2차원 형태의 PE 배열로 구성되며, 각 PE는 인접한 PE로부터 데이터를 전달받아 연산을 수행한 후, 그 결과를 다시 다른 방향의 PE로 전파하는 방식으로 동작한다. 가장 대표적인 사용 예는 행렬 곱셈(Matrix Multiplication)이다. 예를 들어, 두 행렬 A (MxK)와 B (KxN)를 곱하여 결과 행렬 C (MxN)를 계산할 때, A의 요소는 왼쪽에서 오른쪽으로, B의 요소는 위쪽에서 아래쪽으로 각각 이동하며, 각 PE는 자신이 담당하는 결과 행

렬 C의 원소 하나를 계산하는 역할을 수행한다. 이때 PE는 입력된 A의 요소와 B의 요소를 곱하고, 이전 단계에서 계산된 partial sum을 누적하여 최종 결과를 생성한다.

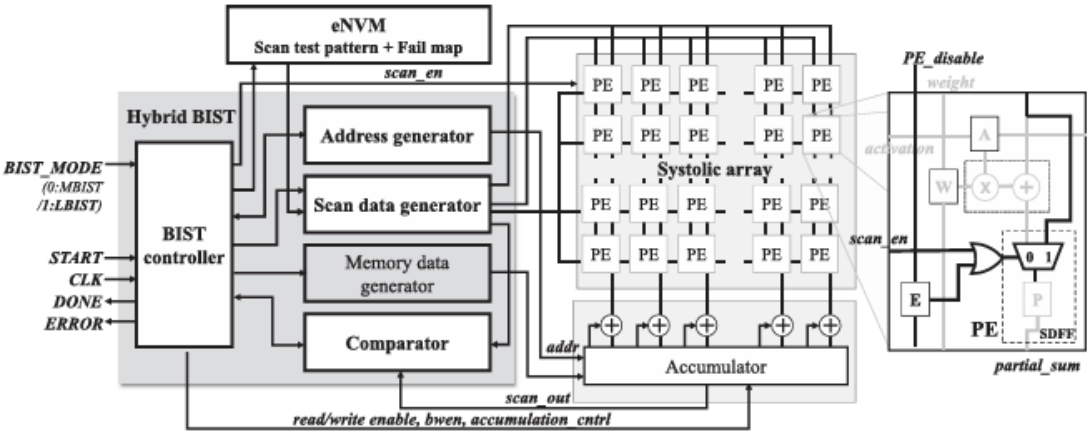
하나의 PE는 주로 세 가지 기능을 수행한다. 첫째, 입력된 두 데이터(Activation과 Weight)에 대한 곱셈 연산(Multiply), 둘째, 이전 단계의 partial sum을 더하는 덧셈 연산(Accumulate), 셋째, 결과를 다음 PE로 전달하는 전파(Propagation) 기능이다. 이와 같은 연산 방식은 MAC(Multiply-Accumulate) 구조로 요약되며, CNN과 같은 딥러닝 모델에서 핵심 연산으로 활용된다. Systolic Array는 클럭 사이클마다 각 PE가 입력 데이터를 받아 MAC 연산을 동시에 수행하기 때문에 높은 병렬성을 구현할 수 있으며, 인접한 PE 간의 데이터 전달만으로 동작이 가능하므로 외부 메모리 접근을 최소화할 수 있어 에너지 효율 또한 높다.

Systolic Array의 동작은 입력 데이터를 시계열적으로 전달한다는 점에서 큰 장점을 갖는다. Activation은 일반적으로 왼쪽에서 오른쪽으로, Weight는 위쪽에서 아래쪽으로 한 클럭마다 한 칸씩 이동하며, 각 PE는 이 데이터를 받아 MAC 연산을 수행한다. 이를 통해 전체 구조는 일정한 클럭 사이클마다 각 위치에서 동시에 연산을 수행하게 되며, 구조적으로 매우 높은 병렬성을 가지게 된다. 또한, 이러한 시간 흐름 기반의 연산 방식은 각 PE가 필요한 데이터를 인접한 PE로부터 받아오기 때문에, 외부 메모리와의 데이터 교환을 최소화하고, 전체 에너지 효율을 향상시킬 수 있다.

이러한 특성 때문에 Systolic Array는 하드웨어 딥러닝 가속기에서 가장 핵심적인 구조로 채택되고 있으며, 대표적인 예로는 Google의 TPU(Tensor Processing Unit), NVIDIA의 Tensor Core 등이 있다. 특히, TPU의 경우 256x256 크기의 대규모 Systolic Array를 내장하여 CNN, LSTM, Transformer와 같은 다양한 딥러닝 모델의 연산을 고속으로 처리할 수 있도록 설계되어 있다. 이러한 구조적 특성은 AI 연산뿐만 아니라, 향후 뉴로모픽 컴퓨팅, 엣지 컴퓨팅 등 다양한 분야로의 확장이 가능하다는 점에서도 중요한 의미를 갖는다.

하지만 높은 병렬성과 연산 밀도를 가진 Systolic Array 구조는, 동시에 많은 수의 PE를 포함하기 때문에, 하드웨어 결함(Fault)에 매우 민감하다는 단점도 가지고 있다. 하나의 PE에 발생한 오류가 전체 행렬 연산 결과에 영향을 미칠 수 있으며, 이러한 결함을 빠르게 탐지하고 복구할 수 있는 테스트 구조가 반드시 필요하다. 이에 따라 최근에는 Systolic Array 구조 내부에 BIST(Built-In Self-Test)를 통합하여, 외부 테스트 장비 없이도 자체적으로 결함을 진단할 수 있는 구조가 적극적으로 연구되고 있다. 본 논문에서 다루는 STRAIT 및 BICS-BIST 구조는 이러한 BIST 기법을 각각 Serial, Parallel 방식으로 적용한 예시이며, 다음 절에서는 기존 STRAIT 구조의 동작 방식과 한계점에 대해 보다 자세히 다룬다.

## 2.2 STRAIT 기반 BIST 구조




<그림 2> STRAIT의 BIST 구조도 [2]

Systolic Array는 수많은 PE로 구성되어 있고, 이들 각각은 MAC 연산을 수행하는 핵심 단위이기 때문에, 하나의 PE라도 오작동할 경우 전체 행렬 연산 결과에 심각한 영향을 미칠 수 있다. 이에 따라 Systolic Array의 신뢰성을 확보하기 위해 결함 탐지를 위한 내장 자가 테스트 기법인 BIST(Built-In Self-Test)의 적용이 필수적이다. 최근 발표된 STRAIT(Scan Test and Repair

Architecture for AI Tensor Core)는 이러한 요구에 대응하기 위해 제안된 대표적인 BIST 아키텍처 중 하나이다. STRAIT는 각 PE 내부에 scan chain을 삽입하여 테스트 데이터를 주입하고, 내부 연산 결과를 캡처한 뒤, 이를 기대값과 비교하는 방식으로 결함을 탐지한다.

STRAIT 구조에서 가장 핵심적인 특징은 테스트 벡터를 각 PE에 shift 방식으로 전달한다는 점이다. 이 구조에서는 test mode로 진입한 후, scan enable 신호(scan_en)를 활성화하여, 테스트 데이터를 scan-in 방식으로 한 비트씩 순차적으로 이동시키며 각 PE의 입력에 도달하게 한다. 예를 들어, 각 PE에 세 개의 입력 벡터(S1, S2, S3)를 주입하는 경우, scan chain을 통해 이 데이터를 순차적으로 시프트하면서 각 PE의 내부 레지스터에 데이터를 채워 넣게 된다. 이 과정은 Scan Shift Process라고 하며, Systolic Array 내 모든 PE에 동일한 테스트 벡터를 공급할 수 있는 장점을 제공한다.

Scan Shift 과정이 완료되면, scan_en 신호를 비활성화하고 capture 신호를 통해 PE 내부의 연산 결과를 레지스터에 저장하는 Scan Capture Process가 수행된다. 각 PE는 입력된 A(W) 데이터를 기반으로 곱셈 연산을 수행하고, 그 결과를 내부 레지스터에 저장한다. 이후, 이 레지스터의 값은 Main Comparator로 전달되어, 예상 결과(expected value)와 비교된다. 만약 실제 결과와 기대값이 일치하지 않으면 에러 플래그가 상승하며, 최종적으로 모든 PE의 에러 상태를 집계하여 총 에러 개수를 산출하게 된다. 이 과정이 Compare 및 Error Count Process이며, 테스트 종료 시점에는 done 신호가 high로 전환되어 전체 테스트가 완료되었음을 나타낸다.

STRAIT 구조는 설계가 비교적 단순하며, 기존 DFT 설계 흐름과 유사한 방식으로 구성되어 있어 실제 구현이 용이하다는 장점을 가진다. 특히 각 PE가 동일한 scan 구조를 가지기 때문에, 전체 구조가 규칙적이고 모듈화되어 있다는 점에서 복잡한 AI 가속기 시스템에 적용하기에 적합하다. 또한, 동일한 테스트 벡터를 모든 PE에 반복 적용할 수 있으므로, 테스트 벡터 설계 측면에서도 효율적이다. 이러한 특징으로 인해 STRAIT는 Systolic Array 기반 AI 가속기의 테스트 신뢰성을 확보하는 데 있어 유망한 구조로 평가받고 있다.

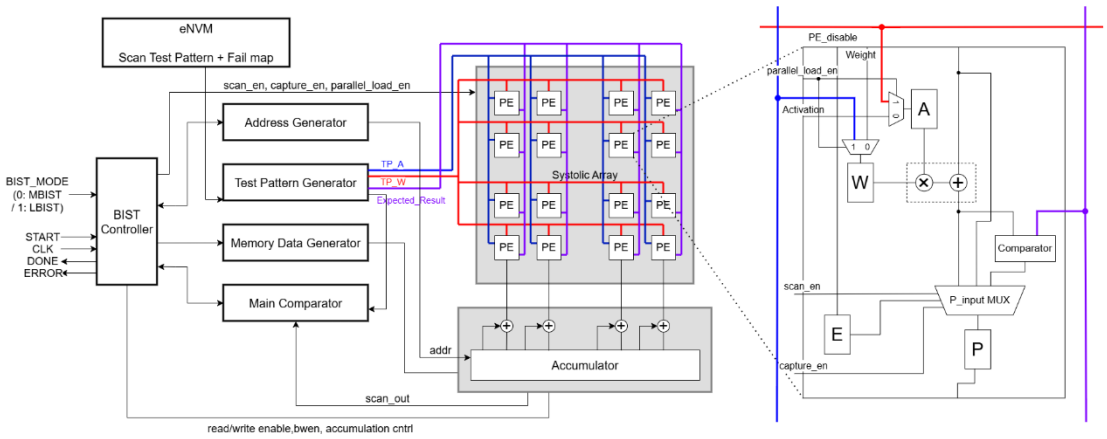
### 2.2.1 STRAIT 구조의 한계

하지만 STRAIT 구조는 테스트 데이터의 전달 방식이 serial 방식에 의존하고 있다는 근본적인 한계를 내포하고 있다. 테스트 데이터가 scan chain을 따라 순차적으로 이동해야 하기 때문에, 전체 테스트 시간은 scan chain의 길이에 따라 선형적으로 증가한다. 예를 들어, 4x4 구조의 Systolic Array에서는 PE당 3개의 테스트 벡터를 주입해야 한다면, 총 3×16=48 클럭 이상의 shift 과정이 필요하게 된다. 이로 인해 전체 테스트 시간이 길어지고, 테스트 중 회로가 long-hold 상태로 유지되어야 하므로 전력 소비 측면에서도 비효율적일 수 있다. 또한, shift-in 시 데이터가 각 PE에 동시에 주입되지 않기 때문에, 테스트 중간 단계에서의 예측 불가능한 출력이 발생할 수 있으며, 이는 테스트 정확도에 영향을 줄 가능성도 존재한다.

이러한 문제를 해결하고자 본 연구에서는 STRAIT의 scan shift 구조를 제거하고, 테스트 데이터를 각 PE에 동시에 병렬 주입하는 BICS-BIST 구조를 제안한다. BICS-BIST 구조는 scan chain이 아닌 broadcast 방식으로 데이터를 주입하여 테스트 시간을 획기적으로 줄이는 것을 목표로 하며, 다음 장에서 그 구조 및 동작 방식에 대해 자세히 설명한다.

# 3. 기존 구조 및 관련 연구

## 3.1 설계 개요




<그림 3> BICS-BIST 구조도

본 장에서는 기존 STRAIT 구조의 한계를 개선하고자 본 연구에서 제안하는 새로운 병렬 BIST 구조인 BICS-BIST(Broadcast-Input Comparison-Scan Built-In Self-Test)의 설계 개요를 서술한다. 제안하는 구조는 Systolic Array의 테스트 효율을 높이기 위해 scan chain 기반의 순차적 테스트 데이터 주입 방식을 제거하고, 그 대신 테스트 데이터를 병렬로 각 PE에 동시에 전달하는 방식을 채택하였다. 이로써 전체 테스트 시간을 획기적으로 줄이고, 복잡한 scan shift 제어 로직을 단순화하여 회로 자원 사용량을 절감할 수 있다.

기존 STRAIT 구조에서는 테스트 데이터를 scan-in 방식으로 전달하기 위해 각 PE 내부에 shift register를 포함한 scan chain을 삽입해야 했다. 이 구조는 데이터가 한 클럭씩 이동하며 순차적으로 각 PE에 도달하므로, 전체 Systolic Array 크기에 비례하여 테스트 시간이 선형적으로 증가하는 문제가 있었다. 반면, BICS-BIST 구조는 모든 테스트 데이터를 한 클럭에 모든 PE에 동시에 주입하는 broadcast 방식을 사용하므로, 테스트 시간 측면에서 scan shift 구조에 비해 우수한 성능을 보일 수 있다.

BICS-BIST 구조의 핵심은 총 네 단계의 동작으로 요약된다. 첫 번째는 초기화 단계(Initialization)로, 클럭 및 리셋 신호를 통해 전체 시스템을 안정된

초기 상태로 설정하는 과정이다. 두 번째는 테스트 벡터 주입 단계(Test Pattern Injection)로, Memory Data Generator를 통해 생성된 activation 및 weight 테스트 벡터가 각각 scan_data_A와 scan_data_W 경로를 통해 모든 PE에 병렬로 입력된다. 이 과정에서는 shift register를 거치지 않고, 각 PE의 내부 입력 레지스터에 바로 데이터가 저장되므로, 주입 시간이 고정된 클럭 수만 소요되며 구조적으로도 간단하다. 세 번째는 연산 및 캡처 단계(Capture)로, 주입된 테스트 벡터를 바탕으로 각 PE가 MAC 연산을 수행하고, 그 결과를 내부 레지스터에 저장하는 과정이다. 마지막으로 네 번째는 비교 및 에러 검출 단계(Compare & Error Count)로, 각 PE의 출력 결과와 기대값(expected value)을 비교하여 에러 유무를 판단하고, 최종적으로 총 에러 개수를 계산한다.

제안하는 BICS-BIST 구조는 전체 제어 흐름을 하나의 Finite State Machine(FSM)으로 제어하며, 각 단계에서 필요한 enable 신호들을 순차적으로 생성한다. FSM은 총 다섯 개의 상태로 구성되어 있으며, 각 상태는 Initialization, Load, Capture, Compare, Done에 해당한다. 특히, 테스트 패턴 주입이 한 클럭에 완료되기 때문에 FSM의 전체 상태 수를 최소화할 수 있고, 상태 간 전이 조건도 간결하게 구성할 수 있다는 장점이 있다. 이를 통해 제어 로직의 복잡성을 줄이는 동시에, 테스트 처리 속도를 높일 수 있다.

BICS-BIST 구조는 이론적으로 scan-in 기반 구조에 비해 훨씬 빠른 테스트를 제공할 수 있지만, 실제 구현에서는 제어 타이밍, 파이프라인 동기화, 비교기 구조 등 다양한 요소가 전체 성능에 영향을 미친다. 따라서 본 구조는 Verilog로 구현한 후 Vivado 환경에서 STRAIT 구조와 동일한 조건으로 시뮬레이션을 진행하여 비교 평가하였으며, 테스트 시간, FSM 상태 수, 회로 자원 사용량 등을 중심으로 그 성능을 분석하였다. 이러한 비교 분석을 통해 본 구조의 효용성과 한계점을 명확히 파악하고, 향후 개선 방향을 도출하는 것이 본 장 이후의 주요 목표이다.

## 3.2 시스템 구성 및 모듈 설명

본 연구에서 제안한 BICS-BIST 구조는 Systolic Array 기반 AI 가속기의 결함을 효율적으로 탐지하기 위한 병렬 입력 기반의 BIST 시스템으로, 크게 다섯 개의 주요 구성 요소로 이루어진다. 이 구성 요소들은 각각 테스트 패턴의 생성, 주소 제어, 데이터 병렬 주입, 연산 수행, 결과 비교 및 에러 검출의 역할을 분담하며, 시스템 전체는 중앙 FSM 제어 블록에 의해 순차적으로 동작한다. 주요 구성 모듈로는 Test Pattern ROM, Memory Data Generator, Address Generator, Systolic Array(PE Array), 그리고 Main Comparator가 있으며, 각 구성 요소의 기능과 역할은 다음과 같다.

### 3.2.1 Test Pattern Generator

첫 번째 구성 요소인 Test Pattern ROM은 Systolic Array의 BIST에 사용될 테스트 벡터를 미리 저장해 놓은 읽기 전용 메모리로, 일반적으로 Verilog의 case 문 또는 .mem 파일을 통해 구현된다. 본 구조에서는 activation, weight, expected output 데이터를 각각 별도의 ROM으로 구성하였으며, 이는 테스트 모드에 따라 필요한 입력 벡터를 Memory Data Generator에 전달하는 기반이 된다. 테스트 모드는 Single Activation(SA) Test와 Timing Delay(TD) Test로 구성되어 있으며, 사전에 정밀하게 계산된 결정론적 패턴(Deterministic Pattern)을 기반으로 동작한다. 이를 통해 무작위 패턴으로는 검증하기 힘든 연산의 코너 케이스(Corner Case)와 타이밍 마진을 효과적으로 테스트할 수 있도록 설계하였다. 각 모드에 따라 ROM에서 참조하는 address 경로가 달라지며, 적절한 테스트 시나리오에 맞는 데이터를 선택적으로 사용할 수 있다.

### 3.2.2 Address Generator

Address Generator는 Test Pattern ROM에 저장된 데이터를 순차적으로 불러오기 위해 사용되는 카운터 기반의 제어 모듈이다. 각 테스트 모드에서 요구

되는 패턴 수에 따라 ROM의 address를 증가시키며, address_valid 신호를 통해 ROM 출력의 유효성을 동기화한다. 이 모듈은 FSM의 Load 상태에서 활성화되며, bist_mode에 따라 참조할 ROM 종류가 달라지므로 선택적으로 activation, weight, expected 데이터에 접근할 수 있도록 구성되어 있다. 또한, Address Generator는 카운터가 최대값에 도달하면 자동으로 동작을 중지하고, FSM이 다음 단계인 Capture 상태로 전이되도록 제어 신호를 제공한다.

### 3.2.3 Systolic Array Module

Systolic Array 모듈은 본 구조에서 테스트의 핵심 대상이 되는 연산 블록으로, 규칙적인 2차원 배열 형태로 구성된 다수의 Processing Element(PE)로 이루어져 있다. 각 PE는 입력으로 주어진 activation 값과 weight 값을 곱하고, 이전 단계의 partial sum과 누적하는 MAC(Multiply-Accumulate) 연산을 수행한다. 본 구조에서는 각 PE 내부에 partial sum 저장용 P 레지스터를 포함시켜 연산 결과를 유지하며, 이후 Compare 단계에서 이를 내부의 Internal Comparator를 통해 Expected Value와 즉시 비교하여 Fail Flag를 생성한다. 기존 STRAIT 구조와는 달리, 본 구조에서는 scan chain이나 shift register를 사용하지 않고, 각 PE에 병렬로 데이터를 주입하는 방식으로 설계하였다. 이를 통해 전체 테스트 시간을 크게 줄이고, 구조의 간소화를 달성할 수 있었다.

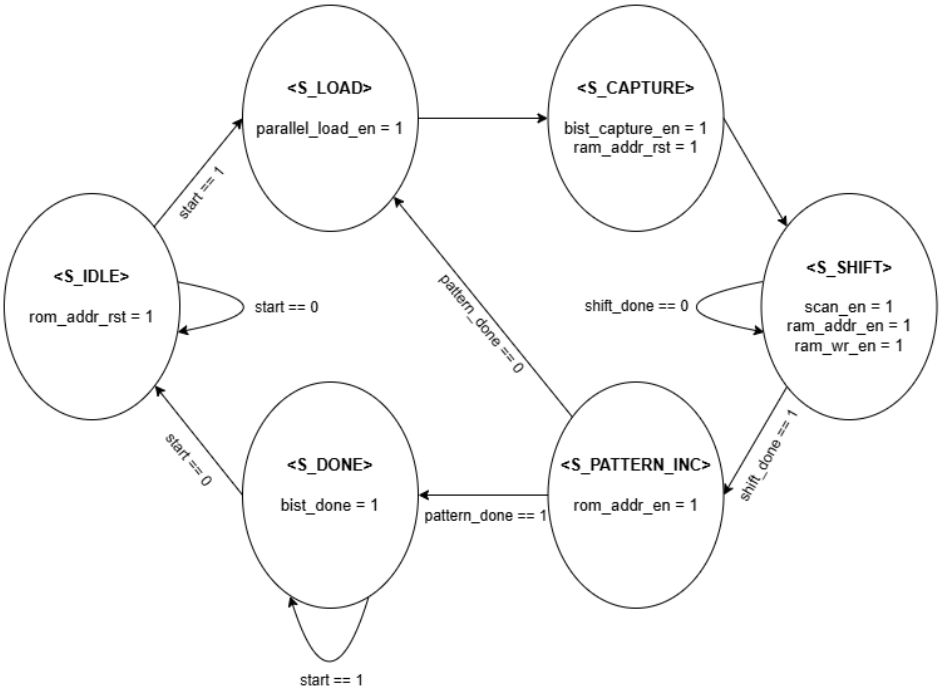
### 3.2.4 Main Comparator

Main Comparator는 Capture 단계에서 각 PE의 연산 결과와 미리 정의된 expected value를 비교하여 결함 여부를 판단하는 모듈이다. 모든 PE는 연산 결과를 동시에 출력하며, 이 결과는 병렬 비교 구조를 통해 expected value와 1:1 대응으로 비교된다. 비교 결과가 일치하지 않는 경우 해당 위치의 에러 플래그가 상승하며, 전체 에러 플래그는 summation logic을 통해 error_count로 집계된다. 최종적으로 FSM의 Done 상태에서 error_count가 외부로 출력

되며, 동시에 done 신호를 통해 테스트 종료 여부를 상위 모듈에 전달한다.

이처럼 제안된 BICS-BIST 구조는 각 구성 요소들이 고도로 병렬화되어 있으며, scan shift를 제거함으로써 테스트 속도와 구조 단순화 측면에서 STRAIT 구조에 비해 유리한 구조적 특성을 가진다. 다음 절에서는 각 모듈의 FSM 구성과 동작 타이밍을 포함한 전체 시스템의 제어 흐름을 상세히 기술한다.

## 3.3 FSM 동작 방식




<그림 4> BICS-BIST Controller의 FSM 다이어그램

본 연구에서 제안한 BICS-BIST 구조는 전체 테스트 흐름을 일관되게 제어하기 위해 Finite State Machine(FSM, 유한 상태 기계) 기반의 제어 방식으로 설계되었다. FSM은 각 동작 단계마다 필요한 제어 신호를 순차적으로 생성하고, 상태 간 전이를 통해 시스템의 동작 흐름을 명확하게 정의하는 역할을 수행한다. 본 구조의 FSM은 총 다섯 개의 상태로 구성되어 있으며, 각각의 상

태는 Initialization, Load, Capture, Compare, Done 상태로 구분된다. 각 상태는 명확한 역할을 가지며, 상위 모듈의 클럭 및 bist_en 신호에 따라 상태가 변화하게 된다. 다음은 각 상태의 정의와 세부 동작 방식에 대한 설명이다.

FSM의 시작 상태는 Initialization 상태이다. 이 상태는 시스템 리셋 직후 진입하며, 전체 시스템을 안정된 초기 상태로 설정하기 위한 단계이다. 이 단계에서는 모든 내부 레지스터 및 상태 변수들이 초기화되며, Address Generator의 카운터도 0으로 리셋된다. Initialization 상태는 외부로부터 bist_en 신호가 high로 입력되기 전까지 유지되며, 해당 신호가 감지되면 다음 상태인 Load 상태로 전이된다.

Load 상태는 테스트 벡터를 각 PE에 병렬로 전달하는 단계이다. 본 구조에서는 Test Pattern ROM으로부터 읽어온 activation 및 weight 데이터를 Memory Data Generator를 통해 scan_data_A와 scan_data_W로 분배하여, Systolic Array 내부의 모든 PE에 동시에 주입한다. 이 과정은 기존 STRAIT 구조에서 필요한 다수의 클럭을 단 한 클럭만에 수행할 수 있다는 점에서 매우 효율적이다. Load 상태에서는 Address Generator가 현재 테스트에 사용할 ROM 주소를 출력하고, 이를 기반으로 Memory Data Generator가 scan_data를 생성한다. 이때 address_valid 신호를 활용하여 ROM 출력 데이터의 유효성을 보장하며, 필요한 모든 데이터가 PE에 주입된 후에는 FSM이 Capture 상태로 전이된다.

Capture 상태는 PE 내부의 연산을 수행하고 결과를 저장하는 단계이다. 각 PE는 이전 상태에서 주입된 activation과 weight 데이터를 이용하여 MAC 연산을 수행하며, 그 결과를 내부 P 레지스터에 저장한다. 본 구조에서는 capture_en 신호가 high가 될 때 연산을 개시하며, 이후 한 클럭 내에 연산 결과가 내부에 저장된다. Capture 단계는 단일 클럭 사이클로 충분하며, 이후 FSM은 자동으로 Compare 상태로 진입하게 된다.

Compare 상태에서는 각 PE의 연산 결과를 expected value와 비교하는 과정이 이루어진다. Compare 단계에서 활성화되는 main_compare_en 신호는 Main Comparator 모듈을 동작시켜, 각 PE의 결과값과 expected_data를 병렬

로 비교하게 한다. 비교 결과는 PE별 에러 플래그로 표현되며, 모든 에러 플래그는 중앙 모듈에서 집계되어 최종 error_count 값으로 산출된다. 이 단계는 테스트 신뢰성을 평가하는 데 가장 핵심적인 과정이며, 병렬 비교 구조 덕분에 비교 연산 또한 한 클럭 내에 완료될 수 있도록 설계되어 있다.

FSM의 마지막 단계는 Done 상태이다. 이 상태에서는 전체 테스트 과정이 완료되었음을 bist 시스템 외부에 알리기 위해 done 신호를 high로 설정하며, error_count와 관련된 데이터 역시 이 시점에서 외부 출력 포트로 전달된다. FSM은 done 상태에서 bist_en이 다시 low가 될 때까지 대기하며, 이후 bist_en이 비활성화되면 초기 상태인 Initialization 상태로 되돌아가 다음 테스트 준비를 시작하게 된다.

FSM은 전반적으로 단순하면서도 효율적인 구조를 지니며, 각 상태는 명확히 구분되고, 불필요한 대기 상태 없이 최소 클럭 수 내에 전체 테스트 과정을 완료할 수 있도록 구성되어 있다. 이러한 FSM 구조는 병렬 BIST 아키텍처에 적합한 형태로 설계되었으며, 실제 구현 시에도 클럭 수 낭비 없이 빠른 테스트 수행이 가능하다는 점에서 기존 STRAIT 구조에 비해 명확한 장점을 가진다. 다만, 구현 환경이나 모듈 간 타이밍 제약 조건에 따라 각 상태 간 전이 시 클럭 지연이 발생할 수 있으므로, 이에 대한 상세한 검토는 4장에서 제시할 실험 및 분석 결과를 통해 다루도록 한다.

# 4. 기존 구조 및 관련 연구

## 4.1 실험 환경

본 연구에서 제안한 BICS-BIST(Broadcast-Input Comparison-Scan BIST)의 구조의 기능 검증 및 성능 평가는 Xilinx Vivado Design Suite (버전 2023.2)를 활용하여 수행하였다. 하드웨어 설계 언어(HDL)로는 Verilog-2001 표준을 준수하여 RTL(Register Transfer Level) 설계를 진행하였으며, 기능 검증을 위해 Vivado Simulator (XSim)를 이용한 Behavioral Simulation을 수행하였다.

합성(Synthesis) 및 구현(Implementation)을 위한 타겟 디바이스는 Xilinx Artix-7 계열의 FPGA를 선정하였으며, 시스템 클럭 주기는 10ns(100MHz)를 목표로 설정하였다. 특히, STRAIT와 BICS-BIST 두 구조 간의 공정한 하드웨어 자원(Area) 비교를 위해, 입출력 핀(I/O Pin) 개수의 제약을 배제하고 순수한 모듈 자체의 로직 면적만을 측정할 수 있는 Out-of-Context (OOC) 모드를 적용하여 합성을 수행하였다.

전력 소비량(Power Consumption) 분석은 합성된 넷리스트(Netlist)를 기반으로 Vivado의 Power Report 기능을 사용하여 측정하였으며, 동적 전력(Dynamic Power)과 정적 전력(Static Power)을 포함한 총 온칩 전력(Total On-Chip Power)을 비교 지표로 삼았다. 타이밍 분석(Timing Analysis) 또한 동일한 제약 조건(Constraints) 하에서 수행하여, 각 구조의 임계 경로(Critical Path)와 최대 동작 주파수($F_{max}$)를 산출하였다.

## 4.2 실험 데이터 및 검증 시나리오

본 실험의 DUT(Design Under Test)인 Systolic Array는 16x16 크기의 배열로 구성하였으며, 각 PE(Processing Element)는 8-bit 정수형(INT8) 데이터를 처리하도록 설계하였다. 테스트의 목적은 Systolic Array의 연산 로직이 정상적으로 동작하는지 검증하고, 결함 발생 시 이를 BIST 회로가 올바르게 탐지하는

지 확인하는 데 있다. 실험은 크게 기능 검증과 결함 주입 두 가지 시나리오로 진행되었다.

### 4.2.1 기능 검증 (Functional Test)

무작위 난수 생성 방식의 불확실성을 배제하기 위해, Test Pattern ROM에 미리 정의된 결정론적(Deterministic) 패턴을 사용하였다. 검증 패턴은 양수 간 연산, 음수 혼합 연산, 0 포함 연산 등 코너 케이스(Corner Case)를 포함하여 총 16개의 테스트 벡터로 구성되었다. 이를 통해 MAC 연산의 오버플로우(Overflow) 처리 및 부호 확장(Sign Extension) 로직의 정상 동작 여부를 RTL 시뮬레이션 레벨에서 전수 검사하였다.

### 4.2.2 결함 주입 시나리오 (Fault Injection Scenarios)

제안 구조의 결함 검출 커버리지(Fault Coverage)를 검증하기 위해, Verilog의 force 명령어를 사용하여 다음과 같은 세 가지 유형의 결함을 인위적으로 주입하였다.

1. **단일 고착 결함 (Single Stuck-at Fault):** 임의의 PE 하나에서 출력 노드가 VDD(Stuck-at-1) 또는 GND(Stuck-at-0)로 단락되는 상황이다.
2. **다중 버스트 결함 (Multi-Burst Fault):** 특정 행(Row)이나 열(Column)에 속한 다수의 PE가 동시에 오작동하는 상황. 이는 칩 내부의 전원 라인 결함이나 클럭 트리 결함 시 발생할 수 있는 실제적인 불량 유형이다.
3. **지연 결함 (Transition Delay Fault):** 신호 값 자체는 정상이지만, 신호가 안정화되는 시간이 클럭 주기를 초과하여 셋업 타임 위반(Setup Time Violation)을 유발하는 상황. 이를 위해 입력 패턴이 0x00에서 0xFF로 급격히 변하는 전이 구간에서 신호 지연을 강제 주입하였다.

각 결함은 BIST의 Capture 동작 타이밍을 확실하게 포함하기 위해 100 클럭 사이클 동안 지속되도록 설정하였다.

## 4.3 실험 결과 및 비교

```
============================================================
 [CASE 1] Normal Operation Benchmark (No Faults)
============================================================
[Time 100000] Test Started...
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_System_Top_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:09 ; elapsed = 00:00:15 . Memory (MB): peak = 1153.805 ; gain = 25.277
run all
------------------------------------------------------------
>> [CASE 1 Result] Speed Comparison
   - BICS Cycles   : 304
   - STRAIT Cycles : 560
>> [CASE 1 Result] BICS Status: PASS (Correct)
```

<그림 5-1> BICS-BIST와 STRAIT 구조의 테스트 소요 시간 벤치마킹 결과

```
============================================================
 [CASE 2] Extensive Stuck-at Fault Injection Scenarios
============================================================

>>> [CASE 2-A] Single Fault at PE[0][0] (Stuck-at-1)
    >> RESULT: DETECTED (Success)

>>> [CASE 2-B] Row Burst Fault at Row 2 (Stuck-at-0)
    >> RESULT: DETECTED (Success)

>>> [CASE 2-C] Column Burst Fault at Col 4 (Stuck-at-1)
    >> RESULT: DETECTED (Success)
```

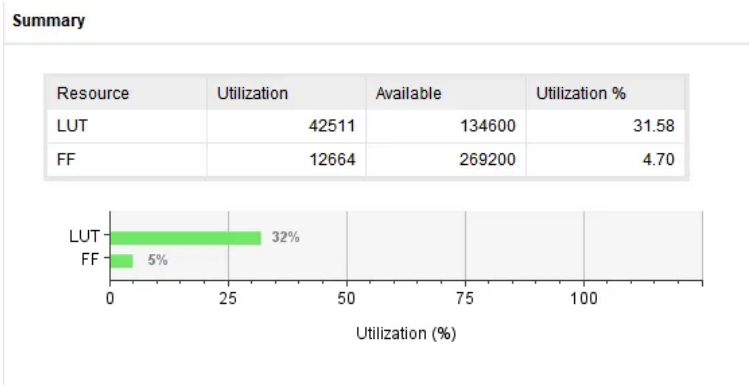
<그림 5-2> 다양한 고착 결함(Stuck-at-Fault) 시나리오 시뮬레이션 결과

```
============================================================
 [CASE 3] Transition Delay (TD) Fault Injection
============================================================
   >>> [MONITOR] ROM Address 6 (TD Pattern) Detected!
   >>> [INJECTION] Force Signal Delay (Hold at 0x00) during Capture Window
   >>> [RELEASE]   Signal Released (Too Late!)
    >> RESULT: DETECTED (Success)
    >> INFO: BICS successfully caught the Slow-to-Rise Transition Fault.

============================================================
                  FINAL SUMMARY
============================================================
 All Tests (Normal, Stuck-at-0/1, Transition Delay) Completed.
 System Robustness Verified.
============================================================

$finish called at time : 18465 ns : File "D:/Vivado_workspace/BICS-BIST/BICS-BIST.srcs/sim_1/new/tb_System_Top.v" Line 270
```

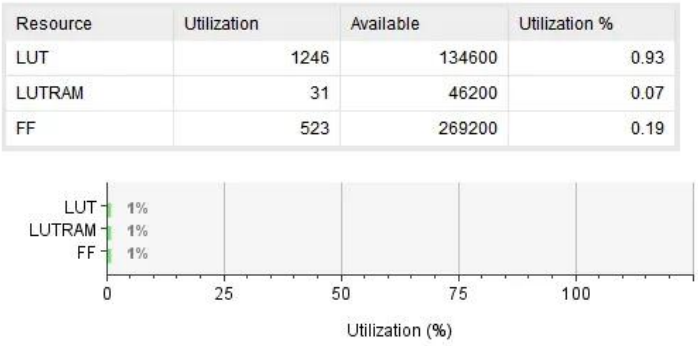
<그림 5-3> 고속 동작 시 지연 결함(Transition Delay Fault) 시뮬레이션 결과

본 절에서는 기존 STRAIT 구조와 제안하는 BICS-BIST 구조의 성능을 정량

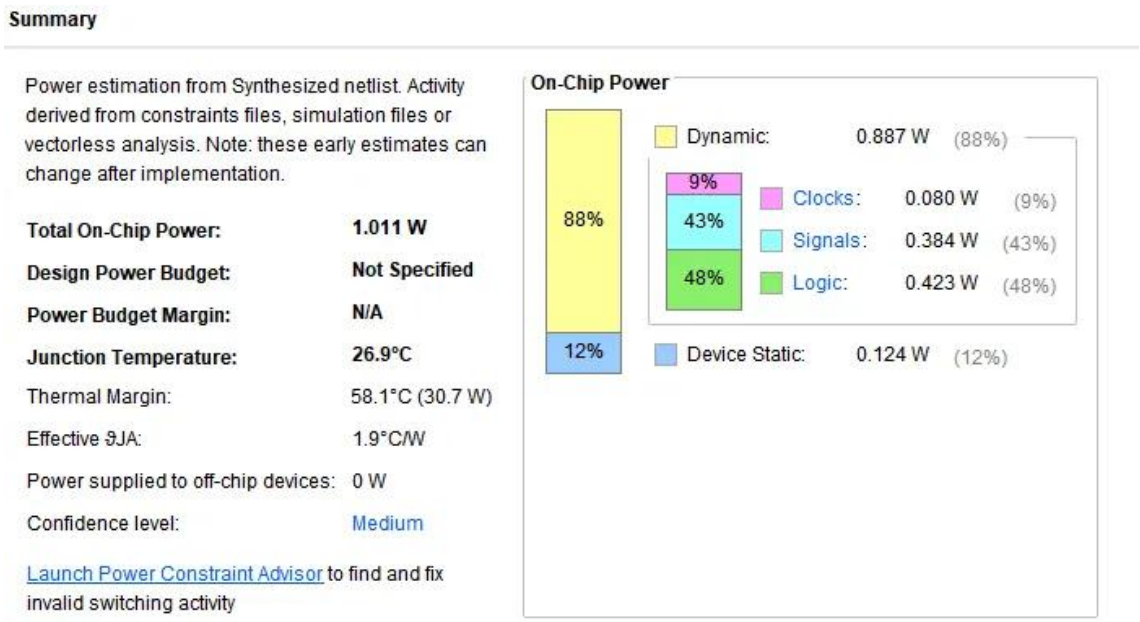
적으로 비교 분석한다. 비교 지표로는 하드웨어 자원 사용량(Hardware Cost), 전력 소모(Power Consumption), 동작 속도($F_{\max}$), 그리고 테스트 효율성(Test Efficiency)을 사용하였다.



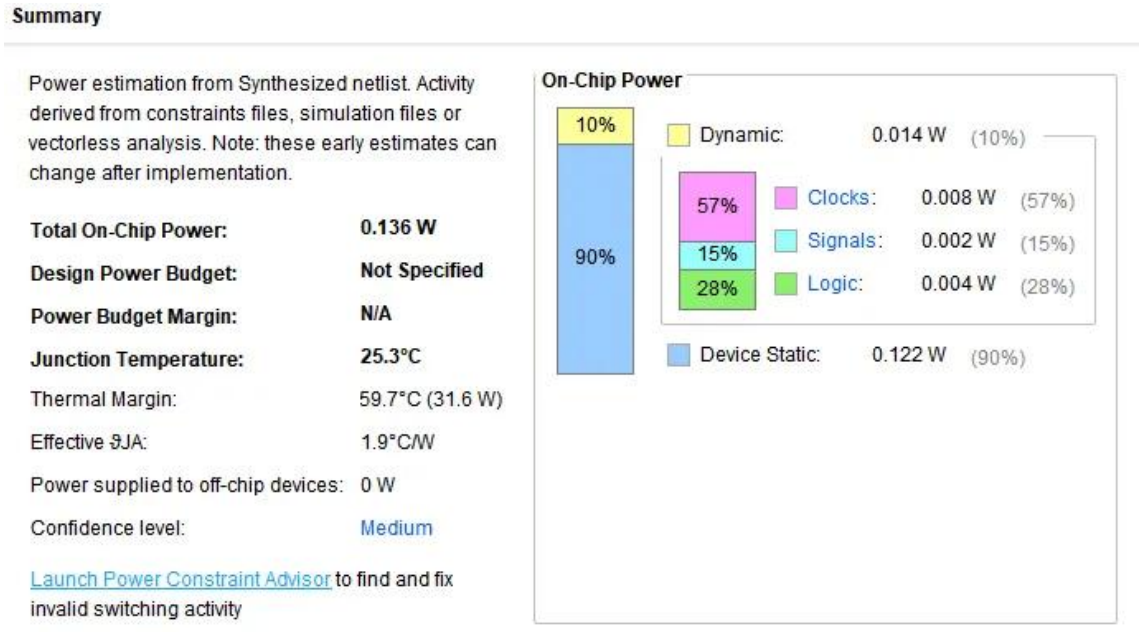

<그림 5-4> BICS-BIST의 Report Utilization Summary



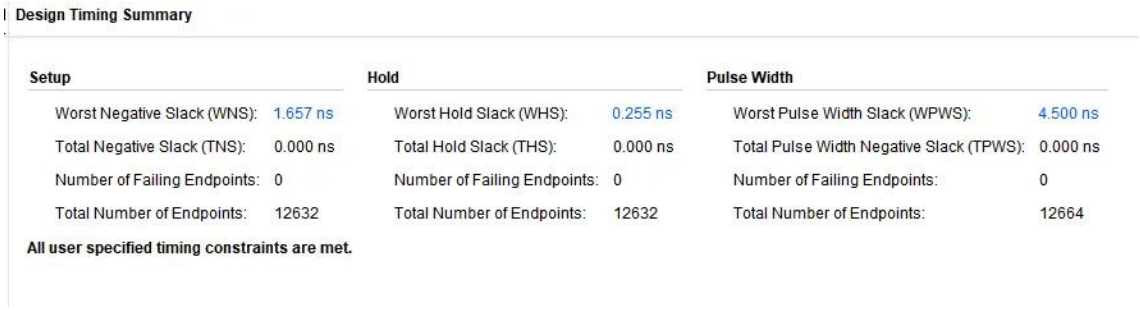

<그림 5-5> STRAIT의 Report Utilization Summary



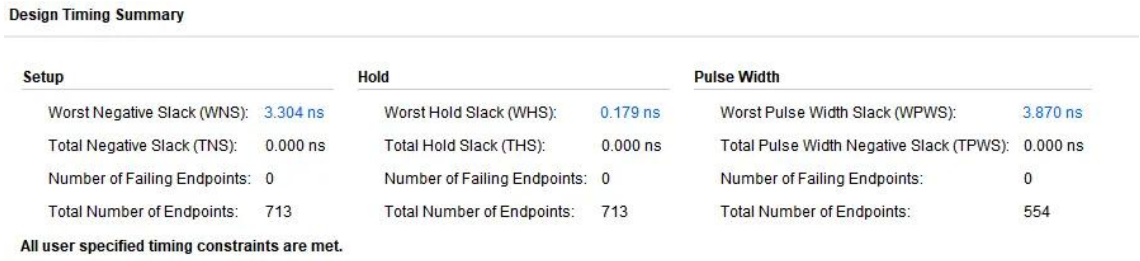

<그림 5-6> BICS-BIST의 Report Power Summary




<그림 5-7> STRAIT의 Report Power Summary




<그림 5-8> BICS-BIST의 Report Timing Summary




<그림 5-9> STRAIT의 Report Timing Summary

### 4.3.1 하드웨어 자원 및 동작 속도

| 구분 (Category) | 성능 지표 (Metric) | STRAIT (Baseline) | BICS − BIST (Proposed) | 변화율 (Change) |
|---|---|---|---|---|
| Hardware Cost | Area (LUTs) | 1,246 | 42,511 | +3,311% |
| | Area (FFs) | 523 | 12,664 | +2,321% |
| | Power (Total) | $0.136W$ | $1.011W$ | +643% |
| Speed | Max Freq($F_{max}$) | $149.34MHz$ | $119.86MHz$ | −19.74% |
| Efficiency | Test Latency | 560 $Cycles$ | 304 $Cycles$ | −45.71% |
| | Execution Time | $3.71\mu s$ | $2.54\mu s$ | −32.3% |

<표 1> STRAIT와 BICS-BIST의 PPA 및 효율성 비교 요약

표 4-1에 나타난 바와 같이, 제안하는 BICS-BIST 구조는 기존 STRAIT 대비 LUT 사용량이 약 34배, 레지스터(FF) 사용량이 약 24배 증가하였다. 이러한 면적 증가는 주로 테스트 패턴을 저장하기 위한 ROM이 최적화되지 않은 FPGA 내부의 분산 로직(Distributed Logic)으로 합성되었기 때문이다. 이에 대한 상세한 원인 분석과 실제 상용 칩 설계 시의 해결책은 5.1절에서 논의한다.

더불어, 제안하는 BICS-BIST 구조는 기존 STRAIT와 달리 각 PE 내부에 32-bit 크기의 내장 비교기(Internal Comparator)를 추가하여 In-Situ Comparison을 수행하도록 설계되었다. 기존 구조는 연산 결과를 외부로 스캔 아웃한 후 중앙 비교기에서 순차적으로 비교하지만, 제안 구조는 분산된 비교기를 통해 연산 직후 즉시 결함 여부를 판단한다. 이로 인해 PE 당 로직 사용량이 증가하여 전체 LUT 및 FF 사용량 증가에 기여하였다. 하지만 이는 병렬성을 극대화하여 테스트 속도를 획기적으로 높이기 위한 필수적인 트레이드오프(Trade-off) 로 분석된다.

또한, 타이밍 분석 결과 BICS-BIST의 최대 동작 주파수($F_{max}$)는 119.86 MHz로, STRAIT(149.34 MHz) 대비 약 19.7% 감소하였다. 최대 동작 주파수 계산식은 다음과 같다.

$$F_{\max} = \frac{1}{Target\ Clock\ Period - WNS}$$

이는 병렬 데이터 주입 및 비교를 위한 추가적인 배선(Routing)과 제어 로직이 데이터 경로에 삽입되면서 임계 경로(Critical Path)가 다소 길어졌기 때문이다. 그러나 이는 목표 동작 주파수인 100MHz를 충분히 상회하므로, 실제 시스템 적용에 있어 타이밍 마진(Timing Margin)은 안정적으로 확보된 것으로 판단된다.

### 4.3.2 테스트 효율성 (Test Efficiency)

가장 유의미한 성과는 테스트 효율성에서 확인되었다. STRAIT 구조는 직렬 스캔(Serial Scan) 방식으로 인해 테스트 완료까지 560 클럭 사이클이 소요되었으나, BICS-BIST는 병렬 주입(Broadcast Input) 방식을 적용하여 304 클럭 사이클만에 테스트를 완료하였다. 이를 실제 물리적인 시간(Execution Time, 100MHz 기준)으로 환산하면, BICS-BIST는 3.04$\mu s$ 만에 테스트를 수행하여 STRAIT(5.60$\mu s$) 대비 약 1.84배(45.7% 시간 단축) 더 빠른 실제 처리 속도를 달성하였다. 이는 배열의 크기가 커질수록 스캔 체인 길이가 늘어나는 STRAIT와 달리, BICS-BIST는 배열 크기에 영향을 받지 않는 병렬 주입 구조

를 가지므로 대규모 시스템일수록 그 효율성이 더욱 극대화될 것임을 시사한다.

### 4.3.3 결함 탐지 신뢰성 검증 (Robustness Verification)

제안하는 BICS-BIST 구조의 신뢰성을 입증하기 위해 앞서 설계한 세 가지 결함 주입 시나리오에 대한 시뮬레이션을 수행하였다. 그림 5-1, 5-2, 5-3은 Vivado 시뮬레이터의 실행 로그 결과이며, 아래 표는 그 결과를 요약한 표다.

| 시나리오 | 결함 유형 | 상태 | 결과/비고 |
|---|---|---|---|
| Case 1 | Normal Operation (Benchmark) | PASS | BICS: 304 Cycles vs STRAIT: 560 Cycles |
| Case 2-A | Single Fault at PE[0][0] (Stuck-at-1) | Detected | Success |
| Case 2-B | Row Burst Fault at Row 2 (Stuck-at-0) | Detected | Success |
| Case 2-C | Column Burst Fault at Col 4 (Stuck-at-1) | Detected | Success |
| Case 3 | Transition Delay Fault (Slow-to-Rise) | Detected | Success (ROM Addr 6) |

<표 2> 결함 시나리오별 탐지 결과 요약

실험 결과, BICS-BIST는 단일 PE의 고장(Case 2-A)뿐만 아니라, 특정 행(Row)이나 열(Column) 전체에 걸쳐 다발적인 오류가 발생하는 상황(Case 2-B, 2-C)에서도 모든 결함을 정확하게 탐지(DETECTED)하고 최종 결과로 Fail 신호를 출력함을 확인하였다.

특히 Case 2-B (Row Burst Fault)의 성공적인 검출은, 각 PE의 1-bit Fail Flag가 시프트 레지스터를 통해 수직 방향으로 전파되는 과정에서 데이터 충돌 없이 정상적으로 동작함을 증명한다. 또한 Case 2-C (Column Burst Fault)

의 검출 성공은, 다수의 에러가 동시에 발생하여 최종 출력단의 Bitwise-OR 로직에 입력되더라도, 에러 신호(Logic High)가 소실되지 않고 확실하게 최종 에러 카운터에 반영됨을 시사한다.

더 나아가 Case 3 (Transition Delay Fault) 실험을 통해, 고속 동작 시 신호가 제시간에 도착하지 못하는 지연 결함(Delay Fault)까지 성공적으로 포착함을 확인하였다. 이는 BICS-BIST 구조가 단순히 '값의 오류(Stuck-at)'뿐만 아니라 '시간의 오류(Timing Violation)'까지 검출할 수 있는 높은 신뢰성을 갖추었음을 실험적으로 입증한 결과이다.

# 5. 고찰 및 결론

## 5.1 실험 결과 해석

본 연구의 실험 결과를 종합해 볼 때, 제안하는 BICS-BIST 구조는 하드웨어 자원(Area)과 동작 주파수(Speed)에서의 손실(Penalty)을 감수하고, 테스트 처리량(Throughput)과 신뢰성(Reliability)을 극대화하는 공학적 트레이드오프(Trade-off) 관계를 명확히 보여준다.

가장 주목할 점은 '속도와 효율성의 역설'이다. 단순 하드웨어 동작 속도($F_{max}$)는 기존 STRAIT 구조가 약 19.7% 더 빠르지만, 실제 전체 테스트를 완료하는 데 걸리는 물리적 시간(Execution Time)은 BICS-BIST가 약 1.48배 더 빠르다. 이는 STRAIT 구조가 데이터를 한 비트씩 순차적으로 밀어 넣는 비효율적인 스캔 시프트(Scan Shift) 동작에 많은 시간을 허비하는 반면, BICS-BIST는 아키텍처 레벨에서의 병렬성을 극대화하여 불필요한 대기 시간을 제거했기 때문이다.

또한, 각 PE 내부에서 수행되는 In-Situ Comparison 기법의 도입은 테스트 데이터의 출력 대역폭(Output Bandwidth) 효율성을 획기적으로 개선하였다. 기존 STRAIT 구조는 결함 검출을 위해 32-bit의 연산 결과를 모두 스캔 아웃(Scan-out) 해야 했으므로, 배열 크기와 데이터 비트 수에 비례하여 스캔 시간이 급증하는 병목 현상이 존재했다. 반면, BICS-BIST는 PE 내부에서 즉시 비교를 수행하고 1-bit의 Fail Flag만을 캡처하여 스캔 아웃한다. 이는 스캔 체인을 통해 이동해야 하는 데이터의 양을 1/32로(32-bit 데이터 기준) 대폭 감소시키는 효과를 가져왔으며, 결과적으로 테스트 레이턴시 단축과 스캔 전력 소모 감소에 크게 기여한 것으로 분석된다.

결과적으로 본 연구는 하드웨어의 단순 클럭 속도보다는 아키텍처의 구조적 효율성(Architectural Efficiency)이 시스템의 전체 테스트 성능에 더 결정적인 영향을 미칠 수 있음을 입증하였다. 한편, 실험 결과에서 면적 및 전력 오버헤드가 크게 측정된 주원인은 FPGA 합성 과정에서 ROM이 최적화되지 않은 LUT 기반 로직으로 구현되었기 때문이다. 실제 상용 칩(ASIC) 설계에서는 이

부분이 고밀도 SRAM이나 외부 메모리 인터페이스로 대체될 것이므로, 제어 로직 자체의 순수 오버헤드는 본 실험 결과보다 현저히 낮을 것으로 판단된다.

본 연구에서 조금 더 심화적인 연구 방향은 다음과 같이 생각해볼 수 있다. 첫째, 메모리 최적화 연구이다. 현재 LUT로 구현된 ROM을 FPGA 내부의 Block RAM (BRAM)이나 외부 DRAM 인터페이스로 대체하여 면적 효율을 개선하는 설계를 적용해볼 수 있다. 둘째, 대규모 배열에 대한 확장성 검증이다. 본 연구의 16x16 배열을 넘어 64x64, 128x128 이상의 대규모 Systolic Array에 적용할 경우 발생할 수 있는 Fan-out 문제와 신호 지연(IR-Drop 등) 문제를 해결하기 위해, 전체 배열을 여러 개의 구역으로 나누어 순차적으로 테스트하는 분할 테스트 기법 도입을 고려할 수 있다.

## 5.2 제안 구조의 장점

**장점:**

1. **테스트 시간 단축:** 병렬 데이터 주입(Broadcast Input) 방식을 통해 테스트 레이턴시(Latency)를 45.7% 획기적으로 단축하여, 시스템 부팅 시간이나 유휴 시간(Idle Time)을 최소화할 수 있다.
2. **스캔 대역폭 최적화 (Bandwidth Optimization):** In-Situ Comparison 방식을 통해 PE 내부에서 32-bit 연산 결과를 1-bit Fail Flag로 압축하여 출력한다. 이는 외부로 전송해야 할 데이터 양을 최소화하여, 제한된 I/O 핀이나 스캔 채널 대역폭 환경에서도 효율적인 테스트를 가능하게 한다.
3. **구조적 직관성 및 호환성:** 복잡한 Scan Chain 제어 없이 주소 생성기와 비교기 기반의 직관적인 구조를 가지며, 1-bit Fail Flag 스캔 방식은 기존 STRAIT의 진단 로직(Diagnosis Logic Chain)과 호환되므로 기존의 결함 위치 추적(Localization) 알고리즘을 그대로 활용할 수 있다는 강점이 있다.

## 5.3 향후 연구 방향

본 연구의 한계점을 보완하고 실용성을 높이기 위해 다음과 같은 후속 연구가 필요하다. 첫째, 메모리 최적화 연구이다. 현재 LUT로 구현된 ROM을 FPGA 내부의 Block RAM (BRAM)이나 외부 DRAM 인터페이스로 대체하여 면적 효율을 개선하는 설계가 요구된다. 둘째, 대규모 배열에 대한 확장성 검증이다. 본 연구의 16x16 배열을 넘어 64x64, 128x128 이상의 대규모 Systolic Array에 적용할 경우 발생할 수 있는 Fan-out 문제와 신호 지연(IR-Drop 등) 문제를 해결하기 위해, 전체 배열을 여러 개의 구역(Zone)으로 나누어 순차적으로 테스트하는 분할 테스트 기법 도입을 고려해야 한다.

## 5.4 맺음말

본 연구에서는 인공지능 가속기의 핵심인 Systolic Array의 신뢰성을 확보하기 위해, 기존 직렬 스캔 방식의 한계를 극복한 병렬 입력 기반 BICS-BIST 아키텍처를 제안하였다. 실험을 통해 제안된 구조가 하드웨어 자원의 증가를 수반함에도 불구하고, 테스트 시간을 획기적으로 단축시키고 고속 동작 시의 지연 결함까지 효과적으로 탐지할 수 있음을 입증하였다. 이는 하드웨어의 물리적 한계를 아키텍처 레벨의 병렬성과 데이터 압축 효율성으로 극복한 사례로서, 향후 고성능·고신뢰성이 요구되는 차세대 AI 반도체 시스템의 테스트 기술 발전에 기여할 수 있을 것으로 기대된다.

# 6. 참고문헌

[1] Xu R, Ma S, Guo Y, Li DS. A Survey of Design and Optimization for Systolic Array-based DNN Accelerators. ACM Comput Surv. 2023;56(1):1-37.

[2] Lee H, Kim J, Park J, Kang S. STRAIT: Self-Test and Self-Recovery for AI Accelerator. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2023;42(9).