A-SSCC 2025 Review

서울대학교 전기정보공학부 박사과정 이재성

Session 8 AI Accelerators and Security Circuits

이번 2025 IEEE A-SSCC의 Session 8은 AI Accelerators and Security Circuits라는 주제로 구성되었으며, 이 세션에서는 AI 연산을 위한 하드웨어 가속기와 보안 회로를 한 자리에서 조망할 수 있도록 총 6편의 논문이 발표되었다. 최근 온디바이스 AI의 적용 범위가 추론을 넘어 생성·적응·학습까지 확장되면서, 엣지 가속기의 설계 목표도 단순 연산 성능(TOPS) 경쟁에서 시스템 효율(메모리 트래픽·지연·전력) 중심으로 빠르게 이동하고 있다. Session 8의 Paper 8.1–8.3은 이러한 흐름을 압축적으로 보여준다. 확산(Diffusion) 기반 생성 모델처럼 반복 단계가 많은 워크로드는 계산량보다 반복 누적되는 데이터 이동과 비선형 연산 오버헤드가 병목이 되기 쉽고, 반대로 현장 적응을 위한 온칩 학습은 업데이트(write) 트래픽이 에너지 비용을 지배한다. 세 논문은 공통적으로 이 병목을 "연산 최적화"가 아닌 데이터플로우 재구성(재사용·스킵·퓨전·파이프라인)으로 풀어낸다.

#8-1 본 논문은 14-nm에서 확산(Diffusion) 기반 텍스트-투-모션(Text-to-Motion) 생성 모델을 엣지에서 실시간 수준으로 구동하기 위한 11.0 TOPS/W급 가속기를 제안한다. 생성형 AI가 이미지/텍스트를 넘어 모션·비디오·멀티모달로 확장되면서, "클라우드에서 만들어 내려받는 콘텐츠"가 아니라 "기기에서 즉시 생성해 상호작용하는 콘텐츠"에 대한 수요가 커지고 있다. 특히 XR/AR, 아바타·가상 캐릭터, 로봇의 동작 생성/보정 같은 응용에서는 지연이 곧 경험 품질을 좌우한다. 이때 diffusion 계열은 단일 pass가 아니라 수십 단계 denoising을 반복하며, 매 단계마다 Transformer 블록이 호출되므로, 단순히 연산량이 큰 수준을 넘어 반복 실행 자체가 시스템 지연과 에너지의 누적 병목으로 직결된다. 따라서 diffusion을 엣지에서 "쓸 만한 속도"로 돌리려면, 단순히 MAC을 빠르게 만드는 가속기만으로는 부족하고, 반복 구조에서 발생하는 중복과 데이터 이동을 "구조적으로" 줄이는 것이 필요하다.
확산 모델의 본질적 병목은 (i) 수십 단계 denoising에 의해 동일한 Transformer 블록이 반복 실행되며 연산/메모리 비용이 누적된다는 점, (ii) 특히 블록 내부에서 FFN이 연산량의 과반(≈60%+)을 차지해 반복 step이 많을수록 FFN 비용이 지배적으로 커진다는 점, (iii) softmax·LayerNorm 등 "비-GEMM" 연산은 FLOP보다 레지스터/버퍼 접근 에너지가 지배적일 수 있어, matmul만 최적화하면 오히려 전체 에너지 효율이 기대만큼 오르지 않는다는 점이다. 이 논문이 설득력 있는 이유는, 이 세 병목을 "각각 따로"가 아니라, diffusion의 반복 구조라는 공통 원인에서 출발해 "재사용·정밀도·비선형 연산 최적화"로

한 묶음으로 풀어냈기 때문이다.

저자들은 "반복 step 사이에는 입력/활성의 시간적 유사성(temporal similarity)이 존재한다"는 관찰을 하드웨어 최적화로 연결하며, 핵심을 세 축으로 정리한다. 첫째, Unstructured activation reuse로 FFN을 무작정 재사용하지 않고, GELU 활성 분포에서 영향이 큰(고활성) 값만 선별 재계산하고 나머지는 재사용하는 방식으로 정확도(모션 품질) 열화를 억제한다. 여기서 중요한 포인트는 "재사용" 자체가 아니라 "재사용의 기준"이다. diffusion step 간에는 어느 정도 유사성이 존재하지만, 생성 모델은 작은 오차가 누적되면 결과가 흔들릴 수 있다. 따라서 이 논문은 FFN 전체를 재사용하는 공격적 접근 대신, 품질에 민감한 부분만 다시 계산하는 선택적 정책을 취한다. 구현 관점에서는 첫 step은 dense 계산으로 기준 출력을 만들고, 이후 step은 마스크 기반으로 필요한 항만 갱신하는 "dense+selective-sparse" 실행 패턴을 구성하여 반복 비용을 구조적으로 낮춘다. 이 방식은 단순 pruning과 달리, 모델 파라미터를 희소화하는 것이 아니라 활성값의 시간적 변화량을 활용해 "이번 step에서 굳이 다시 계산할 필요가 있는가?"를 따지는 점에서, 최근 가속기 연구의 큰 흐름('정적 sparsity'보다 '동적/데이터 의존 sparsity')과 맞닿아 있다. 또한 unstructured(비정형) 희소성은 하드웨어 입장에서는 다루기 까다롭지만, diffusion처럼 반복 횟수가 큰 워크로드에서는 step마다 누적되는 이득이 커서, "까다로워도 할 만한 최적화"가 된다.

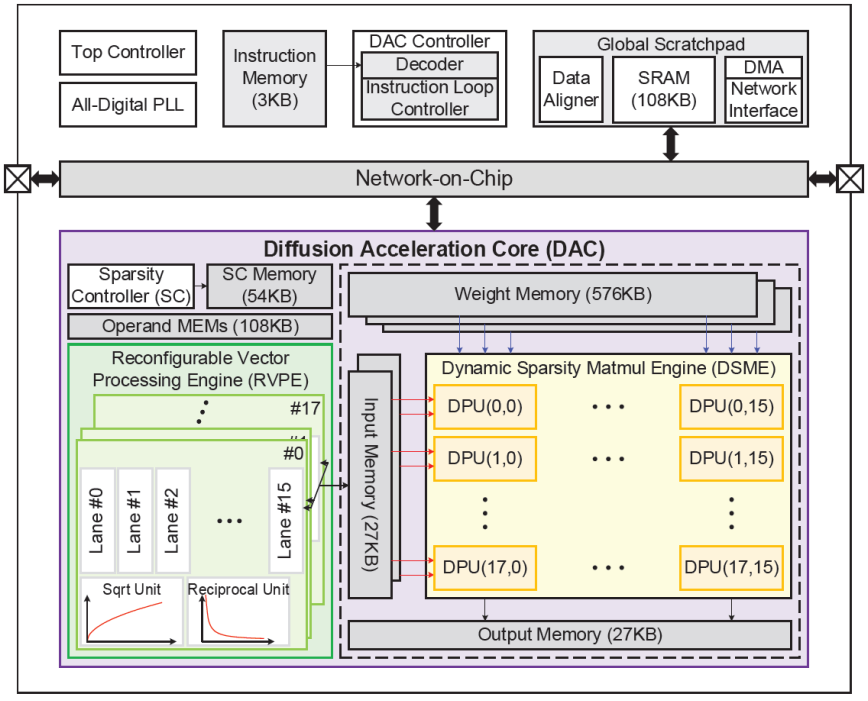
둘째, DSME(동적 희소 matmul 엔진)와 RVPE(재구성 벡터 엔진)를 분리한 DAC(Diffusion Acceleration Core) 아키텍처를 통해, matmul(특히 selective-sparse update)과 LayerNorm/activation/softmax 등 비선형·정규화 연산을 각각 최적 데이터패스로 처리한다. 이 분리는 단순한 모듈 나눔이 아니라, diffusion 가속에서 "진짜 병목이 무엇인지"를 반영한 구조적 선택이다. matmul은 처리량이 중요하지만, softmax·정규화·활성함수는 처리량보다 데이터 재배치/스케일링/정밀도 관리가 성능과 에너지에 더 큰 영향을 준다. 따라서 RVPE는 민감 연산에서 고정밀 누산/가변 정밀 모드를 제공해 중간 requantization으로 인한 품질 손상을 막는 "정확도 방어선" 역할을 한다. 이 설계는 최근 엣지 생성형 가속에서 점점 중요해지는 "혼합정밀(mixed-precision) 관리" 문제를 정면으로 다룬다. 즉, 전부를 높은 정밀도로 유지하면 전력/면적이 커지고, 전부를 낮은 정밀도로 내리면 품질이 무너지는 상황에서, 어떤 연산을 예외로 보호하고(예: LayerNorm), 어떤 연산은 공격적으로 최적화할지(예: FFN 선택적 재계산)를 하드웨어 수준에서 분업한다.

셋째, LUT 기반 softmax에서 연산 자체보다 중간값 저장/읽기(레지스터 R/W)가 더 큰 에너지 비중을 차지한다는 점을 겨냥해, max-subtraction 이후 exp 항에서 작은 값들을 threshold로 제거함으로써 "저장할 가치가 낮은 항"을 계산·저장 단계에서 동시에 줄인다. softmax는 많은 가속기에서 "대충 LUT로 때우면 된다"고 여겨지기 쉬운데, 실제로는 LUT 접근·중간값 저장·정규화 과정에서 데이터 이동이 커져 에너지가 과하게 나갈 수 있다. MoDiff의 threshold softmax는 이 지점에서 "정확도에 덜 기여하는 tail"을 잘라내어, 연산

량뿐 아니라 저장·이동량까지 동시 감소시키려는 전략이다. 이 접근은 diffusion 가속이 matmul 중심에서 벗어나 "비-GEMM을 포함한 end-to-end 효율"로 경쟁 축이 바뀌고 있음을 상징적으로 보여준다.

결과적으로 이 설계는 단순 TOPS/W 향상뿐 아니라, 확산 모델에서 중요한 end-to-end latency(반복 step 누적 지연)와 생성 품질 지표(FID)를 함께 만족시키도록 구성되어 있다. 특히 "실시간"이라는 목표는 단순 peak throughput이 아니라 반복 step 수 × step당 지연의 누적값이 관건이므로, step마다 FFN을 얼마나 덜 돌릴 수 있는지가 핵심이 된다. MoDiff는 이 포인트를 정확히 짚고 FFN 선택적 재계산으로 반복 비용을 줄이며, RVPE와 softmax 근사로 비-GEMM 오버헤드가 발목을 잡지 않게 만든다.

측정 결과, 0.63–0.94 V 및 50–600 MHz에서 동작하며, 600 MHz 기준 6.71 TOPS 수준의 처리량과 최대 11.0 TOPS/W 효율을 보고한다. HumanML3D에서 MLD/MDM 기반 텍스트-투-모션 생성 평가 시 엣지 GPU 대비 최대 17.5× 속도 향상을 달성하면서도 FID 열화는 매우 제한적이라고 제시해, "엣지 생성형 가속"이 실사용 응용(AR/VR·모션 합성 등)으로 넘어가기 위한 정량 목표를 제시한다. 다만 이 논문이 더 강해지려면, diffusion 계열이라도 텍스트-이미지/비디오 등 다른 생성 과제에서 temporal similarity 기반 재사용이 얼마나 잘 성립하는지, 또 모델 구조가 달라질 때(예: attention 비중 증가/감소, 다른 normalization) RVPE/DSME 분할이 얼마나 일반적으로 통하는지에 대한 추가 논의가 뒤따라야 한다. 그럼에도, "diffusion은 엣지에서 어렵다"는 통념에 대해, 반복 구조의 중복을 하드웨어-알고리즘 공동설계로 해체하여 실시간 가능성을 정면으로 보여준 점에서, Session 8의 방향성을 가장 강하게 드러낸 논문 중 하나로 볼 수 있다.



[그림 1] 제안한 MoDiff의 전체 칩 구조

#8-2 본 논문은 22-nm에서 온칩 증분학습(on-chip incremental learning)을 지원하는 비동기(asynchronous) SNN 기반 프로세서를 제안하며, 멀티센서 인식과 적응 학습을 "엣지에서 가능한 비용"으로 만들기 위한 설계 방향을 제시한다. 최근 엣지 AI의 현실적 문제는 "모델이 커서 느리다"만이 아니다. 실제 제품/현장에서는 조명 변화, 배경 변화, 센서 노화/드리프트, 사용자 개인차, 장착 위치 변화 등으로 데이터 분포가 계속 변한다. 이때 모델을 서버에서 주기적으로 재학습해 배포하는 방식은 지연·비용·프라이버시 문제를 동반한다. 결국 엣지 장치는 일정 수준의 적응 능력을 갖춰야 하며, 그 적응은 (i) 완전한 대규모 학습이 아니라, (ii) 새 환경/새 사용자에 맞춘 국소적 업데이트(증분학습/온라인 학습)의 형태로 나타나는 경우가 많다. ANP-R은 바로 이 요구를 SNN 기반으로 풀어내려는 시도이며, "온칩 학습이 가능한 뉴로모픽 프로세서"를 실제 칩 지표(pJ/SOP)로 밀어붙였다는 점에서 의미가 있다.

엣지 환경에서는 성능 유지를 위해 (i) 주기적 재학습이 아니라 기기 내부의 온라인/인크리멘털 업데이트가 필요해진다. 그러나 학습이 포함되는 순간 병목은 MAC 연산량보다 시냅스 업데이트(write) 트래픽과 그 에너지로 이동하며, 특히 SNN에서도 STDP 계열 업데이트가 잦아지면 이벤트 기반 희소성의 이점이 급격히 약화될 수 있다. 즉, "스파이크가 희소하니 저전력"이라는 장점은 추론(inference)에서 더 잘 성립하고, 학습(training/updates) 단계에서는 오히려 업데이트가 빈번해지면서 메모리 write가 지배할 수 있다. ANP-R이 중요한 이유는, 이 문제를 회피하지 않고 "학습 단계에서의 낭비를 줄이는 것"을 설계의 중심에 둔다는 점이다.

ANP-R은 "학습에서 불필요한 업데이트를 줄이고, 업데이트가 필요할 때도 저장 비용을 낮추는" 두 가지 축을 결합한다. 첫째, SSUS(Self-adaptive Synaptic Update Skipping)를 통해 학습 단계에서 의미 없는(변화가 거의 없는) 시냅스 업데이트를 스킵한다. 여기서 핵심은 "학습은 모든 시냅스가 같은 속도로 수렴하지 않는다"는 사실이다. 어떤 가중치는 초기에 빠르게 변하지만, 어떤 가중치는 곧 안정화되어 이후에는 거의 바뀌지 않는다. 그럼에도 전통적인 업데이트 구현은 매번 같은 절차로 update를 시도하기 때문에, 이미 수렴한 부분에 에너지를 계속 태운다. SSUS는 직전 업데이트 결과가 동일하거나 변화가 미미한 구간에서 skip count를 누적해 일정 임계에 도달하면 업데이트를 생략하는 방식으로, 정확도 손실을 최소화하면서도 업데이트 횟수를 큰 폭(논문에서 최대 65% 수준)으로 줄여 학습 에너지를 직접 절감한다. 중요한 건, 이 방식이 단순 "업데이트 간격을 늘리는 스케줄링"이 아니라, 시냅스별/상태별로 "필요성"을 판단하는 적응적 게이팅이라는 점이다. 따라서 동일한 네트워크라도 데이터 분포나 태스크가 바뀌면 스킵 패턴이 달라질 수 있고, 그 자체가 엣지 환경 변화에 대한 유연성을 제공한다.

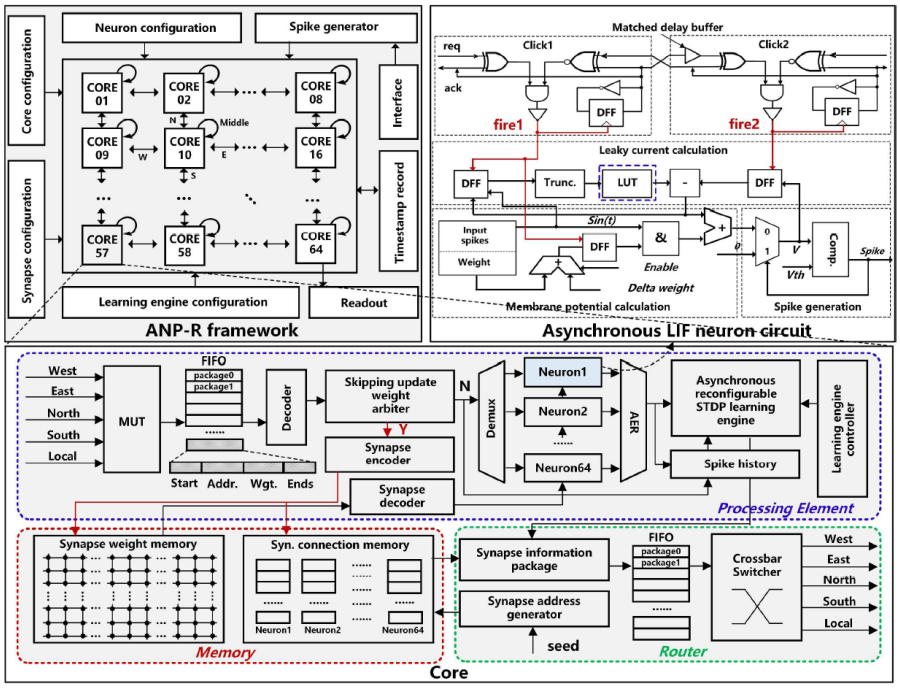
둘째, TWLW(Trained Weights Low-bit Width) coded storage를 도입해 시냅스 저장 포맷 자체를 경량화한다. 학습이 가능한 시스템에서 가중치 표현은 단순 압축이 아니라 "업데이트 통계/정확도 민감도"와 함께 설계되어야 한다. 예컨대 극단적으로 비트를 줄이면 저

장은 싸지지만 학습이 불안정해질 수 있고, 반대로 정밀도를 유지하면 write 에너지가 지배한다. TWLW는 저비트 코딩 기반 저장으로 시냅스 write 에너지(논문에서 62% 수준) 절감과 함께, 성능 열화를 제한된 범위로 관리하는 접근을 제시한다. 이 접근은 "가중치 저장 최적화"를 단순한 메모리 압축으로 보지 않고, 학습 과정의 동역학(어떤 업데이트가 얼마나 자주/얼마나 크게 발생하는가)과 묶어서 보는 흐름에 가깝다. 즉, 엣지 학습에서 중요한 건 '한 번 저장할 때의 비용'뿐 아니라 '학습 동안 몇 번 쓰는지'이므로, write 빈도/패턴을 바꾸는 SSUS와 저장 비용을 낮추는 TWLW를 함께 쓰는 조합이 자연스럽다.

아키텍처적으로는 64개의 비동기 코어와 라우터 기반 연결로 구성된 coarse-grained reconfigurable 구조를 취해, 다양한 SNN 토폴로지/학습 연산 흐름을 코어 단위로 재구성할 수 있도록 한다. 여기서 "비동기(asynchronous)"는 단순히 클록을 없앴다는 의미를 넘어, 이벤트가 없을 때는 자연스럽게 멈추고(event-driven idling), 필요한 곳만 동작할 수 있다는 점에서 always-on 센서 지능과 궁합이 좋다. 또한 coarse-grained reconfigurable이라는 선택은, 뉴로모픽이 본질적으로 다양한 연결/학습 규칙/태스크를 요구한다는 점을 반영한다. 코어당 LIF/IF 뉴런과 시냅스 블록을 갖추고, 전체적으로 4096 neurons 및 0.262M synapses 규모를 구성하여 "단일 태스크 데모" 수준을 넘어 멀티태스크/멀티센서 실험을 수행한다. 여기서 실무적으로 중요한 질문은 "확장성"이다. 코어 수가 늘어나면 라우팅/통신이 병목이 되기 쉽고, 비동기 네트워크에서 혼잡/지연 편차가 학습 안정성에 영향을 줄 수도 있다. ANP-R은 라우터 기반 연결로 확장성을 주장하지만, 향후 더 큰 규모(혹은 더 복잡한 멀티센서 융합)로 갈 때 어떤 병목이 먼저 나타날지(통신? 메모리? update?)는 후속 연구의 핵심 질문이 될 것이다.

측정 결과는 효율을 pJ/SOP 관점에서 제시하며, 0.6 V에서 0.88 pJ/SOP의 피크 효율을 보고한다. 또한 태스크별로 학습 step당 에너지와 SOP 효율을 제시해, "학습 포함 엣지 프로세서"에서 중요한 지표가 단순 처리량이 아니라 energy per step / update cost임을 강조한다. 이 점은 독자에게 매우 중요한 메시지를 준다. 엣지 학습의 평가는 종종 '정확도만' 또는 '추론 에너지'만으로 논의되지만, 실제로는 학습이 포함되면 "학습 1 step에 얼마를 쓰는가", "업데이트를 얼마나 줄였는가"가 시스템 배터리 수명과 직결된다. ANP-R은 update skipping과 coded storage로 그 비용을 직접 제어할 수 있음을 보여준다.

정리하면 ANP-R은 비동기·이벤트 기반 계산이라는 SNN의 강점을 "학습"까지 확장하면서, 업데이트 스킵과 저장 경량화로 온칩 적응의 비용을 통제해, 향후 엣지 AI가 요구할 "항상-동작(always-on) + 환경 적응"의 실질적 구현 경로를 제시한다. 특히 이 논문이 남기는 가치 있는 질문은 다음과 같다. (i) SSUS의 스킵 임계/정책은 태스크별로 어떻게 설정되는가(고정? 적응?), (ii) TWLW의 코딩 방식은 장기 학습 안정성에 어떤 영향을 주는가, (iii) 비동기 라우팅 네트워크에서 통신 지연의 분산이 학습/추론 성능에 어떤 영향을 주는가. 이 질문들은 곧 "온칩 학습 뉴로모픽"이 실사용으로 갈 때 반드시 풀어야 할 과제이며, ANP-R은 그 출발점을 꽤 설득력 있게 제시한 사례로 볼 수 있다.



[그림 2] 제안한 ANP-R의 시스템 다이어그램(좌상), 이너코어 프레임워크(하단), 그리고 이너코어에 적용된 제안 비동기 LIF 뉴런 회로(우상)

#8-3 본 논문은 40-nm 에서 엣지 환경에서의 DNN/SNN 학습과 추론을 모두 지원하는 Learning-in-Memory 프로세서를 제안하며, 학습 워크로드의 본질적 병목을 "연산"이 아닌 외부 메모리 접근(EMA)과 단계 분리(Forward/Backward 분리 실행)로 규정한다. 이 관점은 매우 중요한데, 실제로 학습은 inference 와 달리 (i) forward 활성 저장이 필요하고, (ii) backward/gradient 계산을 위해 중간 텐서를 다시 읽어야 하며, (iii) 업데이트를 위해 weight 및 optimizer state 까지 반복적으로 접근한다. 결과적으로 학습에서는 "연산기 처리량"이 충분해도 DRAM 왕복이 성능을 묶고, DRAM 이 에너지를 지배하는 상황이 흔하다. 특히 엣지에서는 DRAM 대역폭이 제한적이고, DRAM 왕복은 전력/발열/배터리 수명에 치명적이므로, 학습을 엣지로 가져오려면 결국 "외부 메모리 접근을 줄여야 한다"는 결론에 도달한다. Lemem 은 이 결론을 전제로, PIM 매크로만 제시하는 수준을 넘어 학습 루프 전체를 시스템적으로 재배선하려는 시도라는 점에서 주목할 만하다.

학습은 forward 뿐 아니라 error propagation, gradient generation, weight update 까지 포함하기 때문에 중간 텐서와 업데이트 트래픽이 폭증하고, 결과적으로 DRAM 왕복이 성능과 에너지의 상한을 만든다. Lemem 은 이를 해결하기 위해 (i) D-PIM(dual-mode multi-precision ping-pong PIM macro)로 데이터가 메모리 근처에서 곧바로 연산되도록

만들고, (ii) PFBC(pipelined forward-backward processing core)로 학습의 전·후방 흐름을 같은 코어 내부에서 파이프라인으로 엮어 유휴 시간을 줄이며, (iii) 레이어 퓨전(Layer-fused) 라우팅 구조(LFR/I2FR)로 레이어 경계에서의 외부 메모리 왕복을 구조적으로 차단한다. 이 세 가지는 각각 "연산 위치(메모리 근처)", "시간 구조(파이프라인)", "공간/연결 구조(퓨전 라우팅)"를 담당하며, 합쳐서 EMA 를 줄이는 방향으로 정렬된다.

먼저 D-PIM 은 '학습/추론 모두'를 염두에 둔 dual-mode, multi-precision, ping-pong 구조를 통해 데이터 이동을 줄이려 한다. 엣지 학습에서 단순히 PIM 이 유리하다는 주장만으로는 부족하고, 실제로는 (1) 정밀도 선택이 학습 안정성에 미치는 영향, (2) 버퍼링/스케줄링(핑퐁)에서 생기는 idle time, (3) 매크로 주변 로직의 오버헤드가 성능을 갉아먹지 않는지가 관건이다. Lemem 은 D-PIM 을 "학습 친화적으로" 구성하고, 나머지 파이프라인과 결합해 매크로를 최대한 놀리지 않는 방향으로 설계를 전개한다.

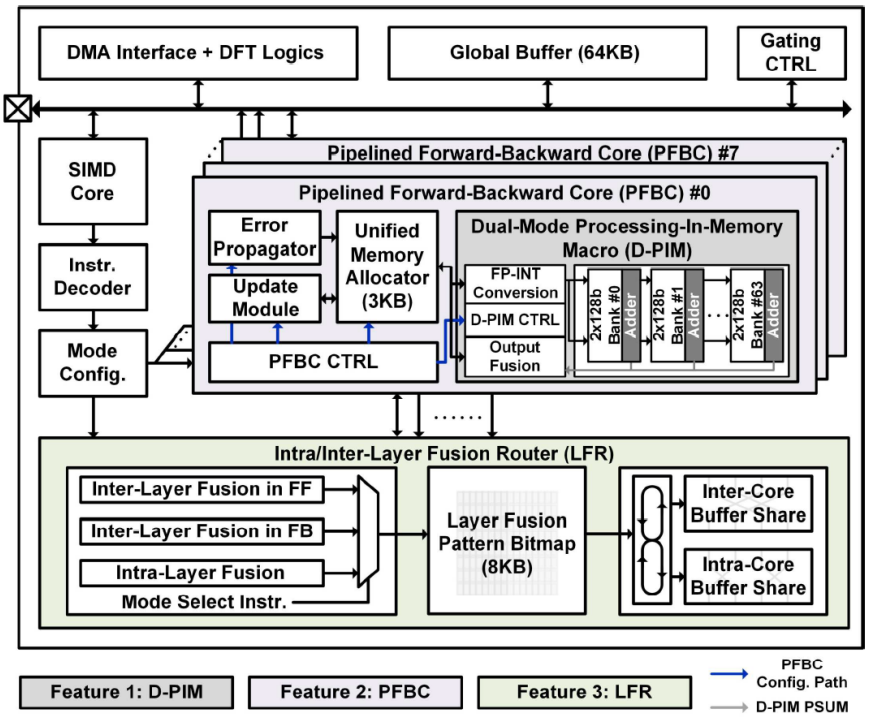
그 다음 PFBC 는 Lemem 의 핵심적인 시스템 아이디어 중 하나다. 학습의 전/후방은 전통적으로 분리되어 실행되며, 레이어 간 경계에서 activation/gradient 를 저장하고 다시 읽는다. PFBC 는 forward, error propagation, gradient generation, weight update 를 단절된 단계로 실행하지 않고, 서로 다른 모듈(D-PIM/EP/WUM 등)을 동시 구동해 "생성된 activation/feature 를 즉시 다음 단계에 소비"하게 함으로써 학습 지연과 EMA 를 동시에 줄인다. 여기서 중요한 건 "파이프라인"이라는 단어가 단순히 스테이지를 나눴다는 의미가 아니라, 학습의 종속 구조를 가능한 범위에서 겹쳐 실행(overlap) 해 유휴 시간을 줄이는 것이다. 엣지에서는 클록을 올려 해결하기 어렵기 때문에, 이런 구조적 overlap 이 처리량을 좌우한다.

마지막으로 레이어 퓨전 라우팅(LFR/I2FR)은 "왜 매번 DRAM 을 가야 하는가?"라는 질문에 대한 하드웨어적 답이다. 많은 학습 가속은 레이어 내부 최적화(예: GEMM 효율)로 끝나지만, 실제로는 레이어 사이에서 feature map 이 이동하며 트래픽이 터진다. Lemem 은 inter-layer FF fusion 과 intra-layer FB fusion 을 함께 고려해 학습 시 필요한 데이터가 온칩에서 순환하도록 만드는 데 목적이 있다. 즉, forward 에서 생성된 데이터가 가능한 한 빠르게 backward/gradient 단계에서 소비되고, 레이어 경계에서 "저장-재로딩"이 반복되는 구조를 끊겠다는 것이다. 이는 최근 학습 가속에서 자주 언급되는 "operator/layer fusion"을 라우팅/캐시 패브릭까지 확장한 형태로 볼 수 있으며, '연산기'가 아니라 '데이터 이동 경로'를 설계의 중심으로 끌어올린다는 점에서 트렌드의 정점에 있다.

이 접근의 효과는 정량적으로도 강하게 제시된다. 논문은 EMA를 최대 4.54× 감소시키고, DDR5 전송 대역폭을 2.0× 증가, 그 결과 에너지를 74.7% 절감하고 GPU 대비 3.48× 속도 향상을 보고한다. 또한 성능/효율 지표로 24.21 TFLOPS 처리량과 179.8 TFLOPS/W 에너지 효율을 제시해 "학습 포함 프로세서"의 경쟁력을 강조한다. 여기서 주목해야 할 것은, Lemem 이 단순히 "PIM 이니까 효율이 좋다"라고 말하는 것이 아니라, EMA 감소/대역폭 증가/에너지 절감/속도 향상이라는 시스템 레벨 연쇄 효과를 근거로 제시한다는 점이다. 엣지에서 학습을 하려면 결국 시스템 전체(메모리·네트워크·스케줄링)가 바뀌어야 하는데, 이 논문은 그 변화를 하나의 통합 설계로 보여준다.

더 나아가 Fashion-MNIST, CIFAR-10, N-MNIST, DVS-Gesture 등 DNN/SNN 을 아우르는 벤치마크 구성을 통해 범용성을 주장하며, 엣지에서 학습이 가능해지기 위한 조건(샘플당 에너지/iteration 지연 등)을 실측 수치로 제시한다. DNN 과 SNN 을 함께 다루는 점은 특히 의미가 있는데, 실제 엣지 시스템은 프레임 기반 인식(DNN)과 이벤트 기반 센싱(SNN)이 공존하는 방향으로 가고 있고, "학습 포함" 요구가 생기면 플랫폼이 분리될수록 통합 비용이 커진다. Lemem 은 이 공존을 하드웨어 구조 차원에서 수용하려는 방향성을 보여준다.

다만 Lemem 이 강한 주장을 하는 만큼, 독자가 자연스럽게 던지게 되는 질문도 있다. (i) layer-fused pipeline 을 어떤 범위까지 자동화할 수 있는가(워크로드가 달라지면 스케줄이 얼마나 바뀌는가), (ii) 다양한 네트워크 구조(복잡한 skip connection, attention-heavy 모델, 멀티모달 결합)에서도 EMA 감소가 같은 정도로 유지되는가, (iii) 학습의 수치 안정성/정확도 보존을 위해 어떤 정밀도/스케일링 정책이 필요한가. 다시 말해, 이 논문은 "엣지 학습이 가능하다"를 강하게 보여주지만, 그 가능성이 실사용에서 보편화되려면 툴체인/컴파일/모델 다양성까지 포함한 확장 논의가 뒤따라야 한다. 그럼에도 불구하고, Lemem 은 "학습은 결국 시스템"이라는 관점을 가장 정면으로 구현한 논문으로, PIM 매크로 단품 성능이 아니라 학습 루프 전체를 관통하는 데이터플로우(파이프라인·퓨전·캐시/라우팅)를 설계의 중심에 두어 엣지 학습의 실현 가능성을 한 단계 끌어올린다.



[그림 3] 제안한 Lemem 프로세서의 전체 구조

저자정보

이재성 박사과정 대학원생

- 소속 : 서울대학교
- 연구분야 : SNN 및 PIM
- 이메일 : jslee0122@snu.ac.kr
- 홈페이지 : https://sites.google.com/view/ic3-snu

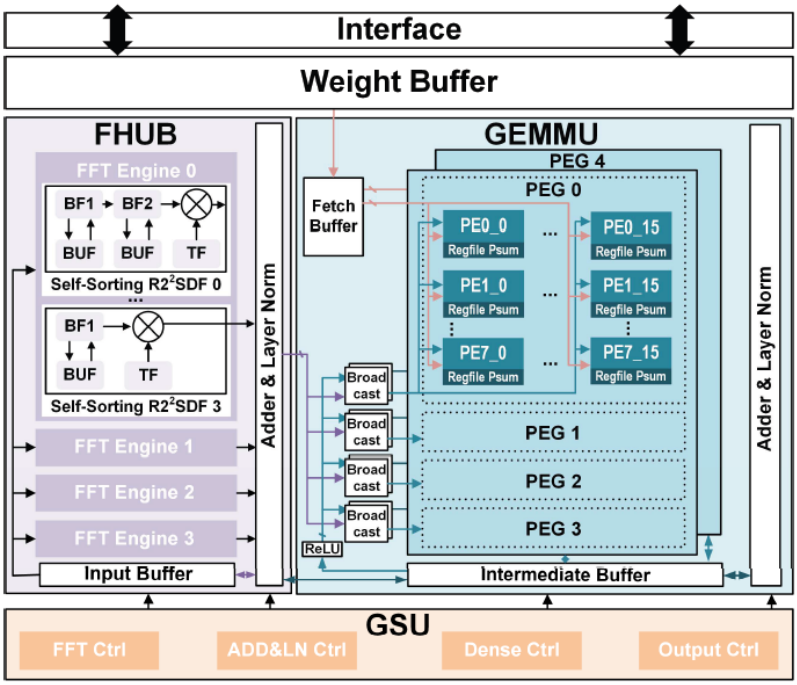
A-SSCC 2025 Review

한국과학기술원 전기및전자공학부 석사과정 박성용

Session 8 AI Accelerators and Security Circuits

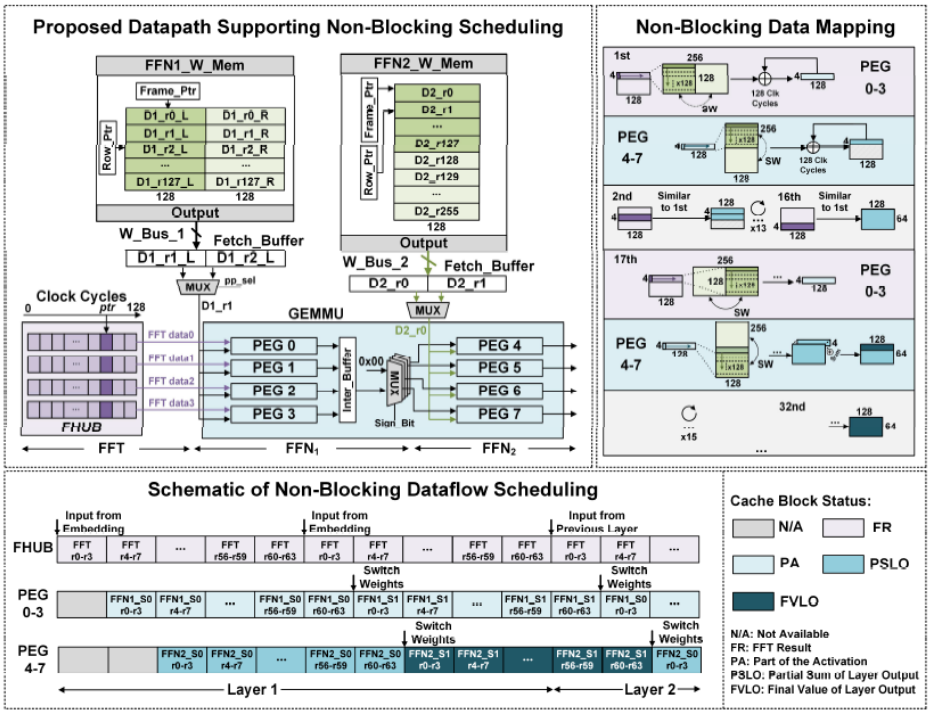
이번 A-SSCC 2025의 Session 8은 AI Accelerators and Security Circuits 라는 주제로 총 6편의 논문이 발표되었다. 이 세션에서는 AI 연산을 가속시키는 architecture와, AI를 활용한 TRNG, PUF와 같은 HW 보안 회로에 중점을 두었다.

#8-4 본 논문은 EdgeFTX 구조를 제안하여, Transformer 모델의 self-attention을 FFT(Fast Fourier Transform) 기반 Token Mixing으로 대체함으로써, 배터리 제약이 극심한 IoT 및 웨어러블 디바이스와 같은 엣지 컴퓨팅 환경에서 최신 Transformer 모델을 구동하고자 한다. 본 연구는 Attention 메커니즘 자체를 경량화하거나 희소성(Sparsity)을 활용하는 데에 집중하는 대신, FFT 기반의 Attention-Free 구조를 채택함으로써, 알고리즘의 시간 복잡도를 $O(N^2)$에서 $O(N \log N)$으로, 공간 복잡도를 $O(N)$으로 낮춰 모델을 경량화하였다. 본 연구는 NAS(Neural Architecture Search)를 활용하여 모델의 정확도와 하드웨어 효율성(FFT 연산 수, SRAM 접근 횟수) 간의 파레토 최적 지점을 탐색하였다. 그 결과, 베이스라인 대비 FFT 연산은 75%, SRAM 접근은 50% 줄이면서도 정확도 저하는 0.76%로 최소화하였다.



[그림 1] 제안된 Transformer 가속기 전체 구조

그림 1은 EdgeFTX의 전체 architecture에 대한 block diagram이다. EdgeFTX는 FFT 연산과 기존 FFN(Feed Forward Network)의 행렬 연산을 효율적으로 처리하기 위해, 4개의 직렬 FFT 엔진(FHUB)과 8개의 병렬 PE 배열(GEMMU)로 구성된 FFT-FFN heterogeneous 아키텍처를 설계하였다. FHUB는 Attention 메커니즘을 대체하는 Token Mixing을 수행하는 전용 모듈이다. 이 모듈은 4개의 직렬 FFT 엔진으로 구성되며, 각 엔진은 128-point FFT를 처리한다. 내부적으로는 Radix-2^2 SDF(Single-path Delay Feedback) 구조를 기반으로 설계되었는데, 가장 큰 특징은 Self-Sorting In-Place 회로를 사용했다는 점이다. 일반적인 FFT 하드웨어는 연산 결과인 주파수 도메인 데이터의 순서가 뒤섞이기 때문에 이를 바로잡기 위한 대용량의 재정렬 버퍼(Reordering Buffer)가 필수적이다. 그러나 FHUB 내의 버터플라이 모듈은 연산과 동시에 데이터를 제자리에 저장(In-place)하며 자동으로 정렬되도록 설계되어, 추가적인 버퍼나 정렬 로직을 제거하여 전력을 7% 절감하고 면적을 11% 줄이는 결과를 보였다. GEMMU는 Transformer의 FFN 연산, 즉 행렬 곱셈을 가속하기 위한 모듈이다. 이는 8개의 병렬 PE array Group(PEG)로 구성되며, 각 PEG는 128개의 PE를 포함한다. 각 PE는 가중치를 로드하고 MAC 연산을 수행하여 FFN의 벡터 스케일링 연산을 병렬로 처리한다.



[그림 2] Non-Blocking Dataflow Scheduling

또한, FFT-FFN 유닛 간의 데이터 의존성으로 인한 파이프라인 Stall를 막기 위해 Non-Blocking Scheduling(NBS) 기법을 사용하였으며, 자세한 dataflow 스케줄링은 그림 2에서 확인할 수 있다. 이는 FFN을 세부 태스크로 나누어, 데이터가 완전히 생성될 때까지 기다리지 않고, FFT 결과가 나오는 즉시 미세 단위(Fine-grained)로 다음 연산에 공급하여 연산 유닛의 가동률을 100%에 가깝게 유지하였다. 이를 통해 layer execution에 걸리는

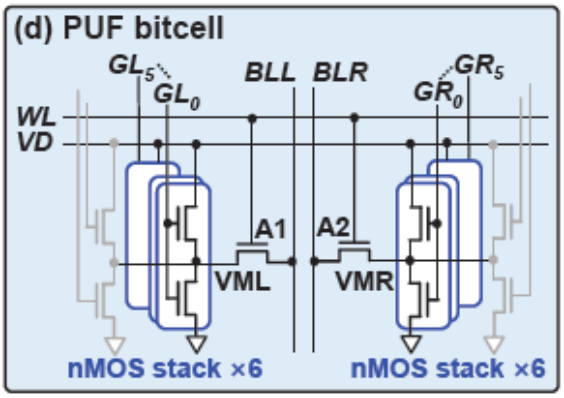
latency를 기존 직렬 스케줄링 대비 1.49배 단축하였다.

22nm CMOS 공정으로 제작된 EdgeFTX 칩은 0.51V, 100MHz 동작 조건에서 3.6mW의 전력을 소모하며, 1.39 TOPS/W의 에너지 효율을 달성하였다. SST-2 벤치마크에서 동일 규모의 BERT 모델 대비 2.3% 높은 정확도를 보였으며, 최신 General-purpose edge 가속기 대비 전력 소모를 최대 17.6배 낮춤으로써, FFT 기반 접근법이 유효함을 보여주었다.

#8-6 본 논문은 ML(Machine Learning)을 활용하여, PUF의 전압 및 온도 변동(VT variation)에 의한 Bit Flipping 오류를 해결하고자 하였다. 먼저 PUF(Physically Unclonable Function)는, 동일한 웨이퍼와 마스크를 사용하더라도 반도체 제조 공정에서 발생하는 Vth mismatch와 같은 Process Variation을 이용한다. 특정 입력을 인가했을 때, 미세한 mismatch를 증폭하여 0 또는 1의 값으로 출력함으로써 Chip에 고유한 지문(fingerprint)을 생성하는 기법이다. PUF는 반도체 지문으로서 고유해야 하지만, 동시에 언제 어디서나 동일한 값을 출력해야 하는 안정성이 필수적이다. 그러나, 동일한 Chip이여도 측정 시마다 온도, 전압 등의 환경 변화에 의해 bit-flipping되어 error가 발생하게 된다.

기존의 Dark-bit masking 기법은 불안정한 셀을 찾아 masking하므로 유효 셀이 줄어드는 문제가 있다. 이러한 문제를 해결하기 위한 Reconfigurable 기법 또한 선택지 제한으로 인해 BER(Bit Error Rate)가 존재하게 된다. 본 연구는 다음과 같은 방법을 통해, -40℃ ~ 120℃에서 zero-BER을 달성하는 reconfigurable PUF 구조를 제안하였다.

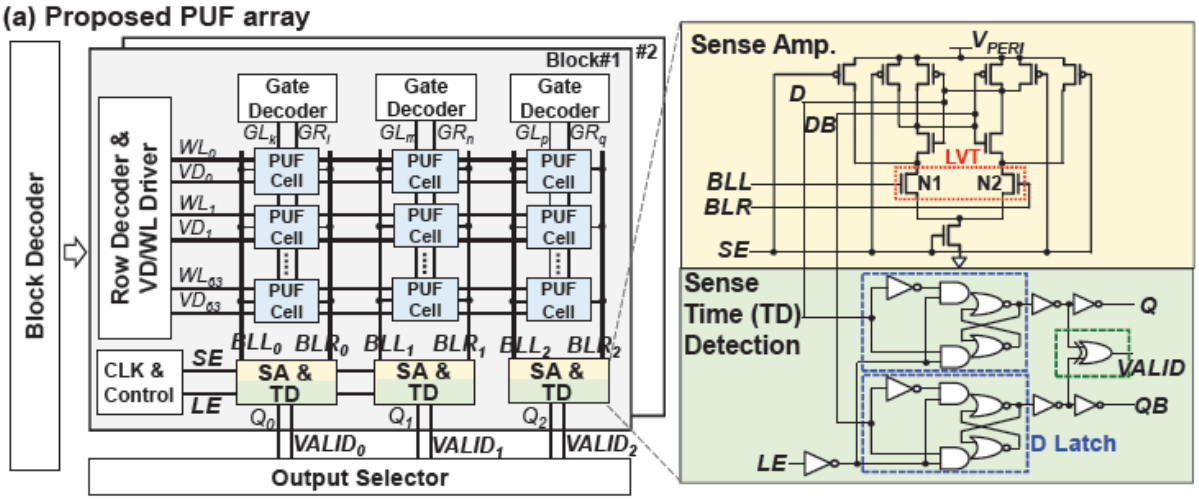
제안된 구조는 PUF 셀 하나당 36가지의 경로(Way) 구성을 지원하도록 설계되었다. 통계적 모델 분석에 따르면, 구성 가능한 경로의 수(N)가 증가할수록 해당 셀 내에서 공정 mismatch가 크고 안정적인 구성을 찾을 확률이 높아진다. N=2 또는 N=4인 기존 연구와 달리 N=36으로 확장함으로써, 불안정한 영역을 벗어난 안정적인 응답을 생성할 수 있는 확률을 높였다. 이를 통해 비트 셀을 버리지 않고도 셀당 4비트의 stable한 출력을 얻을 수 있었다.



[그림 3] PUF 비트 셀의 회로도

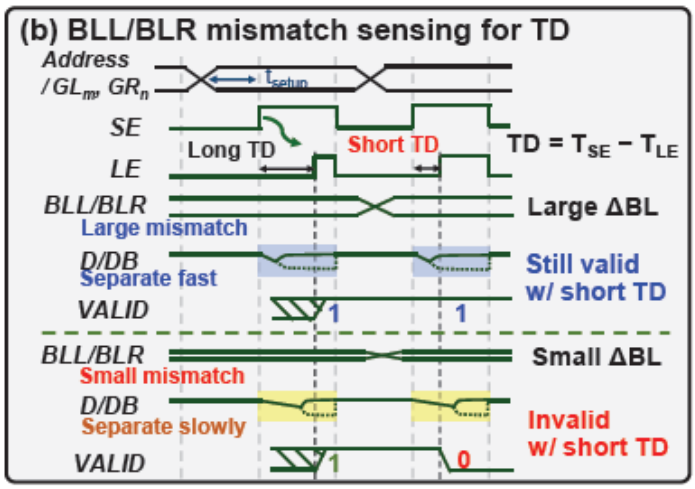
비트 셀의 구조는 그림 3과 같다. 셀은 대칭 구조로 이루어져 있으며, 좌측과 우측에 각각 6개의 직렬 NMOS 스택이 배치되어 있다. 외부에서 인가되는 게이트 선택 신호를

통해 좌우 각각 하나의 NMOS 경로를 활성화할 수 있으며, 이 조합을 통해 단일 셀 내에서 총 36가지의 서로 다른 전류 경로를 형성할 수 있다. 반도체 공정의 mismatch로 인해 선택된 좌우 NMOS 쌍 사이에는 구동 전류 차이가 발생하고, 이는 좌우 Bit line 간의 전압 차이로 나타난다.



[그림 4] PUF array의 블록도와 회로도

제안된 PUF array는 그림 4와 같다. Cell Array에는 Bit line 간의 미세한 전압 차이를 감지하여 증폭시키는 StrongARM latch로 구성된 SA(Sense Amplifier)와, 전압 차이가 증폭되어 0과 1의 안정 상태로 진입하기까지의 지연 시간을 측정하는 TD(Time Detection)가 연결되어 있다. Mismatch가 클수록 Bit line간 전압 차이가 크므로, 0과 1로 더 빠르게 분리되어 TD 값이 작아지게 된다. 따라서 측정된 TD 값을 mismatch를 측정하는 지표로 사용하게 된다. Mismatch에 따른 TD 값의 차이는 그림 5에서 확인할 수 있다. TD sweep을 수행하여 TD 값의 평균과 분산을 측정하여 mismatch의 정도를 나타내고, 동시에 전압 sweep을 수행하여 전압 민감도(VDD sensitivity)를 측정한다.



[그림 5] mismatch sensing을 통해 TD를 구하는 timing diagram

한편, 온도 변화에 따른 안정성을 확보하기 위해서는 일반적으로 높은 비용의 온도 chamber 테스트(Temperature Sweep)가 필요하다. 그러나 본 연구에서는 전압 민감도(VDD Sensitivity)가 온도 민감도(Temperature Sensitivity)와 높은 상관관계를 가짐을 확인하였다. 이를 통해 ML model은 mismatch를 나타내는 TD 값의 분포와 전압 민감도를 입력으로 받아 선형 회귀(linear regression)를 수행하여 -40℃ ~ 120℃ 온도 범위에서 BER을 예측하고, Hungarian 알고리즘을 이용하여 BER이 최소가 되는 4개의 경로 조합을 찾아내게 된다. 이러한 ML-based selection 기법을 사용하여, 해당 온도 범위에서 BER이 0에 가깝게 되도록 설계되었다.

65nm 공정에서 제작된 칩을 대상으로 광범위한 온도 테스트를 진행하여 다음과 같은 결과를 얻었다. 첫째로, 기존 휴리스틱 방식이나 단순 전압 Sweep 방식은 일부 오류를 남긴 반면, ML-based selection은 측정된 모든 온도 범위에서 BER이 6.51E−8 미만으로, 사실상 에러가 없는(zero-BER) 안정성을 보여주었다. 둘째로, 36-Way 구성을 위해 셀 내부 트랜지스터 수가 30개로 늘어났음에도 불구하고 dark-bit masking으로 버려지는 셀이 없어, 유효 비트당 면적은 674 F^2/bit로 기존 고신뢰성 PUF 대비 가장 작은 수준을 달성하였다. 셋째, 생성된 키의 고유성(Uniqueness)을 나타내는 해밍 거리(Hamming Distance)는 이상적인 값인 50%에 근접한 49.1%였으며, NIST의 randomness test를 모두 통과하여 보안 키로서의 적합성을 확인하였다.

저자정보

박성용 석사과정 대학원생

- 소속 : 한국과학기술원 전기및전자공학부
- 연구분야 : Digital Circuit Design, ECC Hardware Design
- 이메일 : sypark@ics.kaist.ac.kr
- 홈페이지 : https://ics.kaist.ac.kr/