MANTECH Publications

Journal of Embedded Systems & Its Applications

Volume 10 Issue 2, July-December 2025

Adaptive Neural Accelerators in Embedded Systems

Sanjeev Rana, Paritosh Baitha, Sruthi Thakur

Deepchand Mehta

Associate Professor Assistant Professor

Department of Electronics and Embedded Systems

Sunrise University of Engineering, India

Sanjeevra2na@rediffmail.com, 12bhaithaparitosh@gmail.com, sruthiyutha@yahoo.com

Abstract

The rapid proliferation of artificial intelligence (AI) and machine learning (ML) applications has transformed embedded systems, necessitating energy-efficient and high-performance computational solutions. Adaptive neural accelerators (ANAs) have emerged as a promising approach to meet these requirements by dynamically optimizing computation, memory access, and power consumption for neural network workloads. This review explores the architecture, design methodologies, and implementation strategies of adaptive neural accelerators in embedded systems. Key design considerations such as resource allocation, adaptive precision, heterogeneous integration, and real-time constraints are discussed. The paper also highlights recent advances, challenges, and future directions in ANA deployment, focusing on balancing performance, energy efficiency, and flexibility for next-generation embedded AI applications.

Keywords: *Adaptive Neural Accelerators, Embedded Systems, Energy Efficiency, Neural Network Hardware, Heterogeneous Computing, AI Hardware, Dynamic Precision, Low-Power AI.*

1. INTRODUCTION

Embedded systems have historically focused on resource-constrained computation with strict requirements on power consumption, thermal budgets, and real-time responsiveness. With the surge of AI applications such as autonomous vehicles, smart sensors, and IoT devices,

conventional microcontrollers and digital signal processors (DSPs) are often inadequate for executing complex neural network algorithms efficiently.

Adaptive Neural Accelerators (ANAs) are specialized hardware modules designed to dynamically adjust computational strategies based on workload requirements. Unlike fixed-function accelerators, ANAs can modify precision, parallelism, memory hierarchy utilization, and other parameters at runtime to achieve optimal trade-offs between performance and energy consumption. This adaptability makes ANAs particularly suitable for embedded systems, where workloads are heterogeneous and power budgets are tight.

This paper reviews the state-of-the-art in ANAs for embedded systems, covering architectural designs, adaptive techniques, applications, and key challenges in deployment. The discussion aims to guide researchers and engineers in developing efficient and flexible embedded AI systems.

2. BACKGROUND AND MOTIVATION

Embedded systems are increasingly integrating artificial intelligence (AI) to enable smart, autonomous, and context-aware applications. From wearable devices that monitor health metrics to autonomous drones navigating complex environments, AI has become a critical component of modern embedded applications. However, embedding AI capabilities into resource-constrained systems introduces a host of challenges that traditional computing architectures struggle to address. Adaptive neural accelerators (ANAs) have emerged as a solution to these challenges, providing specialized, flexible hardware that balances performance, energy efficiency, and adaptability.

2.1 Embedded AI Challenges

The deployment of AI workloads in embedded systems is uniquely challenging due to constraints on energy, computation, memory, and workload diversity. These challenges include:

1. Power and Energy Constraints

Most embedded devices operate on limited battery capacity or energy-harvesting sources. Neural network computations, particularly deep learning models, involve billions of multiply-accumulate operations and frequent memory accesses, leading to high energy consumption. Executing such workloads on general-purpose processors can quickly drain batteries and increase thermal output, making energy-efficient designs critical. Power-efficient

architectures, low-precision computation, and dynamic power management are therefore essential to extend battery life while maintaining acceptable performance levels.

2. Real-Time Performance

Embedded systems often require deterministic performance to meet strict timing constraints. Autonomous vehicles, drones, and industrial robots, for instance, rely on real-time inference for decision-making, object recognition, and motion control. Any latency or delay in processing could lead to catastrophic failures. Therefore, embedded AI systems must provide predictable, low-latency performance, even under variable workloads and environmental conditions. Fixed-function processors are often unable to meet these requirements efficiently, which has motivated the development of specialized neural accelerators.

3. Memory Limitations

High-capacity memory modules are both expensive and power-hungry, limiting their use in embedded devices. Neural networks, particularly deep convolutional or recurrent models, require substantial memory bandwidth to store weights, activations, and intermediate results. Limited memory resources can result in frequent off-chip data transfers, significantly increasing latency and energy consumption. Efficient memory hierarchies, on-chip caching, and data reuse strategies are therefore vital to enable AI inference in embedded systems.

4. Workload Variability

AI workloads are inherently heterogeneous. For example, convolutional layers involve dense matrix multiplications, fully connected layers involve large-scale dot products, and activation layers require nonlinear operations. Additionally, workload characteristics vary depending on the input data, model size, and application domain. Embedded systems must adapt to these dynamic requirements, balancing computation, memory, and power resources in real-time. Fixed-function accelerators cannot respond effectively to these variations, leading to suboptimal performance or wasted energy.

In summary, embedded AI demands high computational throughput, low power consumption, and the ability to adapt to variable workloads—all within constrained physical and thermal budgets. These requirements have driven the search for novel hardware architectures capable of delivering AI capabilities efficiently.

2.2 Emergence of Neural Accelerators

To address the unique demands of embedded AI, neural accelerators have emerged as specialized hardware designed for executing neural network operations efficiently. Unlike general-purpose processors, neural accelerators focus on accelerating matrix operations, convolutions, and other computationally intensive layers commonly found in AI models.

Traditional Neural Accelerators

Traditional accelerators are often fixed-function and optimized for specific neural network types (e.g., convolutional neural networks) and fixed precision (e.g., 16-bit or 8-bit arithmetic). While these designs improve performance and reduce energy consumption for their target workloads, they lack flexibility. Any deviation in model type, layer complexity, or precision requirement can result in underutilization of hardware resources or excessive power consumption.

Adaptive Neural Accelerators

Adaptive neural accelerators (ANAs) overcome these limitations by introducing runtime configurability. Key adaptive features include:

- **Dynamic Precision Adjustment:** ANAs can switch between floating-point, fixed-point, or reduced-bit precision operations depending on the layer or input data. This flexibility reduces energy consumption for less sensitive computations while maintaining accuracy for critical layers.
- **Reconfigurable Data Paths and Memory Hierarchies:** ANAs allow dynamic rearrangement of compute units, memory access patterns, and data routing to match the specific workload, improving utilization and reducing bottlenecks.
- **Workload-Aware Parallelism Optimization:** ANAs can allocate or scale computational resources based on real-time workload intensity. For example, more cores or parallel units can be activated during high-demand operations, while idle units are power-gated during light workloads.

These adaptive features make ANAs particularly suited for embedded systems, where workloads and operating conditions vary frequently. By intelligently managing computation, memory, and energy resources, ANAs enable efficient, flexible, and real-time AI processing in resource-constrained environments.

In essence, the combination of embedded AI challenges and the limitations of fixed-function hardware has motivated the emergence of adaptive neural accelerators as a key enabler for next-generation embedded intelligence.

3. ARCHITECTURE OF ADAPTIVE NEURAL ACCELERATORS

The architecture of an Adaptive Neural Accelerator (ANA) is designed to meet the competing requirements of high computational throughput, low energy consumption, and runtime adaptability. Unlike traditional fixed-function accelerators, ANAs are built with flexibility in mind, allowing dynamic adjustments in computation precision, memory access, and parallelism based on workload requirements. The overall architecture typically follows a modular design, where each component can be independently optimized and configured.

3.1 Core Components

An ANA generally comprises four primary components: **compute units**, **memory subsystems**, **control logic**, and an **interconnect network**. Each of these components plays a critical role in enabling adaptive and efficient execution of neural network workloads.

3.1.1 Compute Units

Compute units form the computational heart of an ANA. These units are specialized for neural network operations such as:

- **Matrix Multiplication:** The most common operation in fully connected layers and attention mechanisms.
- **Convolution Operations:** Key for convolutional neural networks (CNNs), optimized using systolic arrays or parallel multiply-accumulate (MAC) units.
- **Activation Functions:** Non-linear operations like ReLU, Sigmoid, or Tanh applied to intermediate outputs.
- **Pooling and Normalization:** Operations such as max-pooling or batch normalization that require intermediate computations.

Design Features of Compute Units in ANAs:

- **Parallelism:** Compute units are highly parallelized to process multiple neurons or channels simultaneously, improving throughput.

- **Adaptive Precision Support:** Each unit can dynamically switch between different data precisions (e.g., 4-bit, 8-bit, 16-bit, or floating-point), balancing energy efficiency and accuracy.
- **Resource Sharing:** Units can be allocated to high-demand layers during runtime or powered down during low workload phases to save energy.

3.1.2 Memory Subsystems

Memory is a critical component in ANAs because neural network computations are often memory-bound rather than compute-bound. Efficient memory design directly impacts energy consumption and latency. ANAs typically feature hierarchical memory subsystems, including:

- **Registers and Local Buffers:** Ultra-fast, low-capacity memory located within compute units to hold frequently accessed data like neuron activations and weights.
- **On-Chip SRAM:** Medium-capacity memory that stores larger weight matrices and intermediate results. SRAM offers faster access than off-chip memory and reduces energy-intensive data transfers.
- **Off-Chip DRAM:** Large-capacity memory for storing full neural network models and input datasets. Accessing DRAM is energy-intensive, so ANAs employ strategies like caching and data reuse to minimize DRAM access.

Adaptive Memory Features:

- **Dynamic Data Placement:** Data is moved between memory layers depending on access patterns, reducing latency and energy consumption.
- **Prefetching and Reuse:** Frequently used weights or activations are prefetched into on-chip memory for repeated use, optimizing throughput.
- **Compression and Quantization:** Memory footprint is reduced by storing weights and activations in low-bit formats when full precision is unnecessary.

3.1.3 Control Logic

The control logic is the “brain” of an ANA, responsible for monitoring workload characteristics and dynamically adjusting the operation of compute and memory resources.

Functions of Control Logic:

- **Precision Management:** Determines the optimal bit-width for computations in each neural network layer to balance energy efficiency and accuracy.
- **Resource Scheduling:** Allocates compute units and memory bandwidth based on layer complexity and workload demand.
- **Power and Thermal Management:** Controls clock gating, voltage scaling, and power islands to reduce energy usage while preventing overheating.
- **Runtime Adaptation:** Adjusts operational parameters based on dynamic changes in input data, model layers, or execution environment.

The control logic ensures that ANAs maintain high efficiency and adaptability across diverse workloads, making them suitable for heterogeneous embedded AI applications.

3.1.4 Interconnect Network

The interconnect network provides flexible data routing between compute units, memory subsystems, and peripheral components. Its design is critical for maintaining high throughput and avoiding communication bottlenecks.

Key Characteristics:

- **Reconfigurable Topology:** Supports dynamic re-routing of data based on the active compute units and memory access patterns.
- **High Bandwidth:** Provides sufficient bandwidth to feed compute units with activations and weights in real-time.
- **Low Latency:** Minimizes data transfer delays to maintain real-time processing performance.
- **Scalable Design:** Can accommodate additional compute or memory units as models grow or workloads increase.

Common interconnect architectures include **mesh networks**, **ring buses**, and **network-on-chip (NoC)** designs, each optimized for latency, throughput, and energy efficiency.

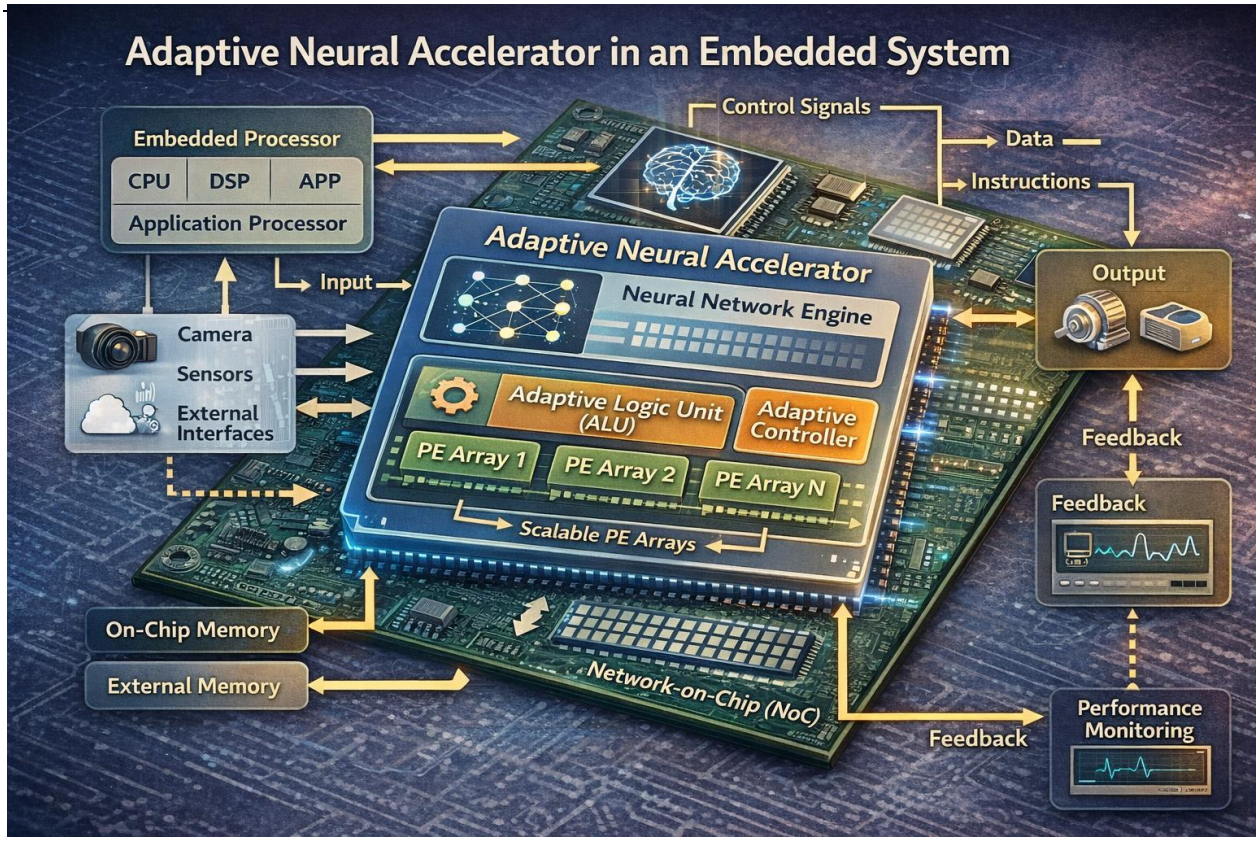

Figure 1 shows a typical architecture of an adaptive neural accelerator in an embedded system.

3.2 Adaptive Precision

Adaptive precision is one of the core features of ANAs, enabling the accelerator to adjust the numerical bit-width of computations dynamically according to workload requirements. Traditional accelerators often use fixed precision—commonly 16-bit or 32-bit operations—which can be overkill for certain neural network layers or datasets, leading to unnecessary energy consumption and reduced throughput. Adaptive precision addresses this by selectively lowering or raising the bit-width of operations while preserving overall model accuracy.

Mechanism of Adaptive Precision

1. **Layer-Level Precision Selection:** Different layers in a neural network exhibit varying sensitivity to numerical precision. For instance:
   - Early convolutional layers may tolerate 8-bit operations without significant accuracy loss.
   - Fully connected layers or output layers may require 16-bit or floating-point precision to maintain accuracy.

2. **Dynamic Precision Switching:** During runtime, the control logic monitors activations and layer requirements and instructs compute units to switch precision accordingly. For example, a layer with low variance in activations might be processed using 4-bit arithmetic, while a layer with high variance switches to 16-bit operations.
3. **Mixed-Precision Pipelines:** ANAs can implement multiple precision units in parallel, enabling simultaneous execution of low-precision and high-precision operations within the same network.

Benefits of Adaptive Precision

- **Energy Reduction:** Energy consumption in digital circuits scales approximately linearly with bit-width. For example, reducing precision from 16-bit to 8-bit can cut the energy required for a multiply-accumulate (MAC) operation by nearly 50%.
- **Improved Throughput:** Lower-bit operations require fewer hardware resources per computation, allowing more operations to execute in parallel. This improves overall throughput, particularly for layers with large neuron counts.

- **Accuracy Preservation:** By dynamically allocating higher precision only to sensitive layers or computations, ANAs maintain model accuracy while benefiting from energy savings in less sensitive layers.
- **Reduced Memory Footprint:** Lower-bit precision reduces the size of weights and activations, allowing more data to fit in on-chip memory and reducing expensive off-chip memory accesses.

Example:

Consider a convolutional neural network with 5 layers: the first three layers can run on 8-bit arithmetic, the fourth layer requires 16-bit precision, and the output layer requires 32-bit for accurate classification. An ANA with adaptive precision can adjust computations per layer dynamically, achieving energy savings and throughput improvement without compromising accuracy.

3.3 Dynamic Resource Allocation

Dynamic resource allocation complements adaptive precision by efficiently distributing compute, memory, and energy resources according to workload demands. ANAs operate

under the principle that workload intensity varies both spatially (across layers) and temporally (over

time). Efficient resource allocation ensures high utilization of compute units, optimized memory access, and minimized energy consumption.

Key Strategies

1. Compute Resource Sharing

- Idle or underutilized compute units can be dynamically reallocated to layers with higher computational demand.
- For instance, in a deep neural network, fully connected layers may require more MAC units than convolutional layers at certain stages. By sharing idle units, throughput is maximized without increasing hardware area.
- Dynamic core allocation also supports multi-task workloads, enabling concurrent execution of multiple models or layers.

2. Memory Bandwidth Optimization

- Memory access is often a bottleneck in ANAs. Dynamic memory management prioritizes layers that require frequent reuse of weights and activations.
- Techniques include:
  - **Layer Fusion:** Combining multiple layers to reduce intermediate data transfers.
  - **Prefetching:** Loading frequently used weights into on-chip memory before execution.
  - **Banked Memory Access:** Using multiple memory banks in parallel to reduce latency.

3. Power Management

- Unused or idle compute units and memory blocks are power-gated or clock-gated to reduce leakage power.
- Dynamic Voltage and Frequency Scaling (DVFS) allows the accelerator to reduce power for less demanding layers while maintaining real-time performance for critical layers.
- These techniques are particularly important for battery-powered embedded devices, where even small energy savings can significantly extend operational time.

Interaction Between Adaptive Precision and Resource Allocation

Adaptive precision and dynamic resource allocation work synergistically:

- **Precision affects resource requirements:** Lower-precision operations require fewer resources, freeing compute units for other tasks.

- **Resource allocation informs precision decisions:** Layers with limited memory bandwidth or compute availability may benefit from reduced precision to maintain real-time performance.

4. DESIGN METHODOLOGIES

4.1 Hardware-Software Co-Design

Effective ANA deployment requires collaboration between hardware and software. Key strategies include:

- **Compiler-Aware Mapping:** Neural network graphs are mapped to hardware units based on layer complexity and memory footprint.
- **Runtime Profiling:** Accelerators monitor execution to adjust precision and parallelism dynamically.
- **Layer Fusion and Scheduling:** Multiple layers may be fused to reduce memory access and energy overhead.

4.2 Heterogeneous Integration

ANAs can be integrated with other embedded components:

- **DSPs for Signal Processing:** Complementing ANAs for tasks like sensor preprocessing.
- **General-Purpose Cores:** Handling control and non-neural computation.
- **Memory Hierarchy Optimization:** Combining on-chip SRAM with off-chip DRAM for large models.

4.3 Low-Power Techniques

Energy efficiency is critical in embedded ANAs:

- **Voltage and Frequency Scaling (DVFS):** Reduces energy for less demanding tasks.
- **Approximate Computing:** Accepts minor errors in low-sensitivity operations to save power.
- **Clock Gating and Power Islands:** Idle modules are disabled to reduce leakage.

5. Applications in Embedded Systems

Adaptive neural accelerators have found use in multiple domains:

Application Domain	Description	Benefits of ANA
Autonomous Vehicles	Object detection, path planning	Low-latency, energy-efficient AI
Smart Cameras	Image recognition, event detection	Real-time processing, reduced power
Wearable Health Devices	ECG, activity monitoring	Low-power continuous inference
IoT Sensors	Anomaly detection, predictive maintenance	Edge AI reduces network dependency
Robotics	Navigation, object manipulation	Dynamic workload adaptation

6. RECENT ADVANCES

Recent studies have advanced ANA design:

1. **Sparse Neural Networks:** Exploiting sparsity in weights and activations to reduce computation and memory access.
2. **Reconfigurable Interconnects:** Using networks-on-chip (NoC) to dynamically reroute data for different workloads.
3. **Edge AI Optimizations:** Lightweight models combined with adaptive accelerators to reduce latency and energy in real-time embedded applications.
4. **Integration with Neuromorphic Computing:** Incorporating event-driven computation for ultra-low-power scenarios.

7. CHALLENGES AND OPEN ISSUES

Despite progress, several challenges remain:

- **Model-Hardware Co-Optimization:** Ensuring models can fully exploit adaptive features.
- **Overhead of Adaptation:** Control logic and monitoring consume area and power, which may offset gains.

- **Standardization:** Lack of common APIs and design frameworks for ANAs.
- **Thermal Management:** High-performance modes may cause localized heating in embedded devices.

8. FUTURE DIRECTIONS

Future research may focus on:

- **Cross-Layer Adaptation:** Integrating adaptation at algorithm, compiler, and hardware layers.
- **Self-Learning Accelerators:** ANAs that learn optimal configurations from workload patterns.
- **Security-Aware Design:** Protecting neural models against side-channel attacks in embedded scenarios.
- **Integration with 3D Chiplet Architectures:** Improving bandwidth and latency by stacking accelerators with memory and logic.

9. CONCLUSION

Adaptive neural accelerators are a key enabler for bringing advanced AI to embedded systems. Their ability to dynamically adjust computation and memory usage provides substantial benefits in performance, energy efficiency, and flexibility. While challenges remain in overhead management, standardization, and thermal control, recent advances in hardware-software co-design, heterogeneous integration, and adaptive precision indicate a promising future. As AI applications expand across embedded domains, ANAs are expected to play a central role in enabling intelligent, low-power, real-time systems.

REFERENCES

1. Chen, Y., Emer, J., & Sze, V. (2016). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12), 2295–2329.
2. Jouppi, N. P., et al. (2017). In-datacenter performance analysis of a tensor processing unit. *ISCA*, 1–12.
3. Reagen, B., et al. (2019). Minerva: Enabling low-power, highly-accurate deep neural network accelerators. *MICRO*, 1–14.

4. Zhang, C., et al. (2015). Optimizing FPGA-based accelerator design for deep convolutional neural networks. *FPGA*, 161–170.
5. Horowitz, M. (2014). 1.1 computing's energy problem (and what we can do about it). *IEEE ISSCC*, 10–14.
6. Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. *NIPS*, 1135–1143.
7. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4), 743–768.
8. Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017). Pruning filters for efficient convnets. *ICLR*, 1–13.
9. Song, L., et al. (2020). Edge AI: Neural network acceleration for embedded devices. *IEEE Transactions on Computers*, 69(12), 1806–1820.
10. Zhao, R., et al. (2021). Adaptive neural network accelerators: Design and implementation. *ACM TACO*, 18(4), 1–23.