RADemics

Parallel Processing Frameworks on FPGA for High Throughput Neural Network Inference

R. Jegadeesan, A. Devi
JYOTHISHMATHI INSTITUTE OF TECHNOLOGY & SCIENCE, HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Parallel Processing Frameworks on FPGA for High Throughput Neural Network Inference

[1]R. Jegadeesan, Professor & HOD, Department of Computer Science and Engineering, Jyothishmathi Institute of Technology & Science, Nustulapur, Karimnagar, India. ramjaganjagan@gmail.com

[2]A. Devi, Assistant Professor, Department of CSE, Hyderabad Institute of Technology and Management, Hyderabad, Telangana, India. devim.cse@hitam.org

Abstract

The increasing demand for real-time, energy-efficient, and high-throughput inference of deep neural networks has positioned FPGAs as a compelling hardware platform due to their inherent parallelism, reconfigurability, and customizability. This book chapter investigates advanced parallel processing frameworks on FPGAs tailored for neural network acceleration, emphasizing architectural strategies that balance throughput, latency, and resource constraints. A comprehensive analysis of data-level, task-level, pipeline, spatial, and hybrid parallelism is presented, with a focus on their synergistic deployment to meet the unique computational requirements of diverse deep learning models. Particular attention is given to loop pipelining and systolic array-based spatial parallelism for matrix-intensive workloads, along with latency-optimized inter-PE communication schemes. Model-specific parallelism control using meta-compilers and high-level synthesis (HLS) pragmas is explored to demonstrate how automation and model-awareness can drive architectural customization and performance scaling. By integrating hardware-efficient techniques such as LUT-based activation computation, memory-optimized dataflows, and pragma-directed code generation, the chapter outlines a practical path from algorithmic description to deployable FPGA inference engines. The interaction between architectural design choices and neural model characteristics is dissected to uncover optimization opportunities for edge AI, embedded processing, and real-time signal interpretation. Experimental insights and synthesis-driven validations further reinforce the feasibility of proposed frameworks under realistic resource and timing constraints.

Keywords: FPGA, parallel processing, neural network inference, loop pipelining, systolic array, high-level synthesis.

Introduction

The exponential growth in deep learning applications across domains such as autonomous systems, healthcare diagnostics, and real-time video analytics has brought renewed focus on specialized hardware for neural network inference [1]. While general-purpose CPUs and GPUs have been traditionally used for deep learning tasks [2]. Their energy consumption, limited real-time determinism, and relatively high latency pose challenges for edge-based and embedded deployments [3]. In contrast, Field Programmable Gate Arrays (FPGAs) offer a promising alternative by enabling application-specific acceleration with lower power budgets and higher predictability [4]. Their fine-grained parallelism and configurability provide an adaptable substrate

to map computational graphs of neural networks directly onto hardware, allowing for custom dataflows and tightly coupled control over processing pipelines [5].

Neural networks, particularly deep and convolutional architectures, involve large-scale matrix multiplications, nonlinear activation functions, and hierarchical feature extraction, which demand high arithmetic throughput and efficient memory management [6]. FPGAs support spatial and temporal parallelism that can be tailored to the exact workload, making them suitable for deploying inference models with minimal bottlenecks [7]. The advancements in High-Level Synthesis (HLS) tools have lowered the barrier to FPGA programming [8]. By abstracting low-level hardware complexities into high-level representations, enabling algorithmic descriptions to be translated into optimized RTL implementations [9]. This capability allows rapid design-space exploration for various forms of parallelism, including pipelined computation, task-level concurrency, and data-parallel execution across multiple processing elements [10].