FPGA Capstone

Paper Presentation

Partial dynamic reconfiguration framework for FPGA: A survey with concepts, constraints and trends

Fpga-based SoC design for real-time facial point detection using deep convolutional neural networks with dynamic partial reconfiguration

A remote partial-reconfigurable SoC with a RISC-V soft processor targeting low-end FPGAs

CGRHS

- **Definition**: Systems using pre-designed, large-scale programmable blocks like ALUs and multipliers, optimized for specific tasks.
- **Key Characteristics**:
- Pre-fabricated and rigid architecture.
- Organized in linear, crossbar, or mesh arrays, with mesh arrays offering better routing and parallelism.
- Suitable for fixed-function hardware accelerators.
- **Disadvantages**:
- **Low Flexibility**: Fixed architecture limits adaptability to diverse or evolving applications.
- **High Area Utilization**: Larger macros consume significant chip area compared to fine-grain systems.
- **Static Configuration**: Architecture is fixed at fabrication, making runtime modifications impossible.
- **Low Code Density**: Inefficient for applications requiring detailed customization or granularity.
- **Limited Reusability**: Poor adaptability for reusing resources across multiple applications.
- **Obsolete for Modern Needs**: Could not meet demands for high logic density, real-time adaptability, or dynamic functionality, leading to a shift toward fine-grain systems.
- **Applications**: Retained for niche, high-performance applications like signal processing, multimedia, and wireless communications.

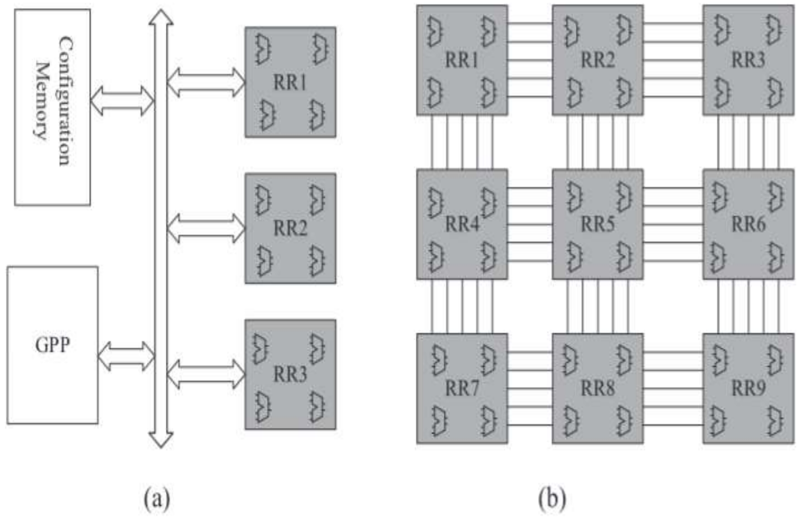


Fig. 1. (a) Standard CGRA (b) Distributed CGRA.

FGRHS

•**Definition**: Systems offering reconfiguration at the 1-bit or logic-element level using Configurable Logic Blocks (CLBs), allowing precise hardware customization.
•**Key Features**:
•**High Flexibility**: Adaptable to diverse and evolving applications.
•**Precise Optimization**: Tailors functionality at the finest level of granularity.
•**Reusability**: Supports resource sharing across multiple tasks or workloads.
•**On-Chip Integration**: Combines reconfigurable logic, embedded processors, and memory for compact, high-performance designs.
•**Why It's the Industry Standard**:
•Offers unmatched flexibility and real-time adaptability for dynamic workloads.
•Meets modern demands for high-performance, logic-dense applications in IoT, AI, and automotive systems.
•Enables partial reconfiguration, reducing downtime and improving efficiency.
•Supported by advanced tools like Xilinx PlanAhead for easier implementation.
•Outperforms Coarse-Grain systems in flexibility, efficiency, and adaptability.
•**Applications**: Adaptive signal processing, Real-time AI accelerators, Multimedia and image filtering, Fault-tolerant and evolvable hardware systems.

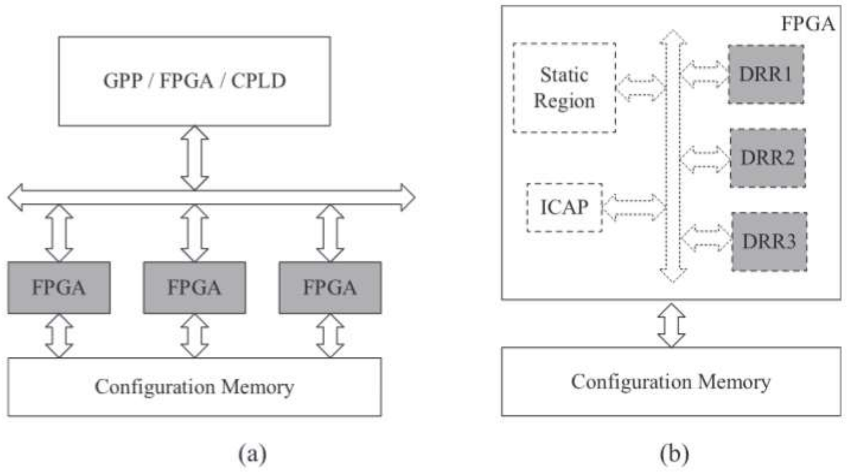


Fig. 2. (a) External FGRA (b) On-chip FGRA.

Partitioning and Reconfiguration Techniques

- **Partitioning**: Ensures efficient use of resources and supports modular design.
  - **Static Region**: Holds fixed logic that remains operational during reconfiguration (e.g., controllers and core functionality).
  - **Dynamic Region (DRR)**: Reconfigurable area synthesized separately for each application.
- **Reconfiguration Techniques**:
  - **Full Reconfiguration**: Reconfigures the entire FPGA, suitable for fixed applications but time-intensive.
  - **Partial Reconfiguration**: Updates only specific regions, reducing downtime and enhancing adaptability.
  - **Internal vs. External Reconfiguration**:
    - Internal: Uses ICAP for high-speed, on-chip updates.
    - External: Relies on SelectMAP or JTAG for configuration from external sources.
- **Challenges and Considerations**:
  - **Throughput Bottlenecks**: Reconfiguration speed is limited by memory and controller performance.
  - **Granularity Trade-offs**: Fine-grain systems offer flexibility but increase complexity and reconfiguration times.
  - **Tool Support**: Tools like Xilinx PlanAhead and GoAhead simplify partitioning and PR implementation.
- **Future Directions**:
  - Enhanced security for remote PR workflows.
  - Optimized tools for LUT-level and real-time reconfiguration.
  - Integration with AI, IoT, and cloud-based platforms for broader adaptability.

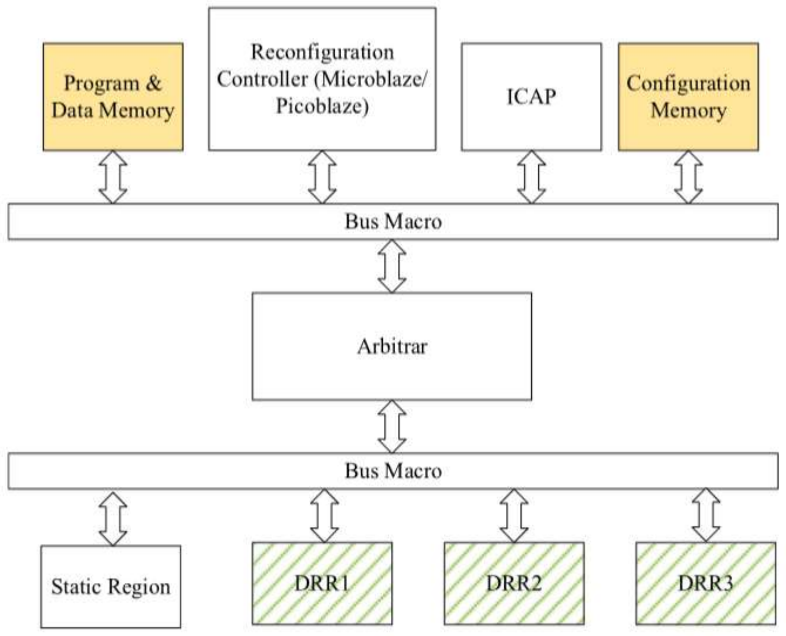


Fig. 5. Two bus architecture of On-chip FGRHS.

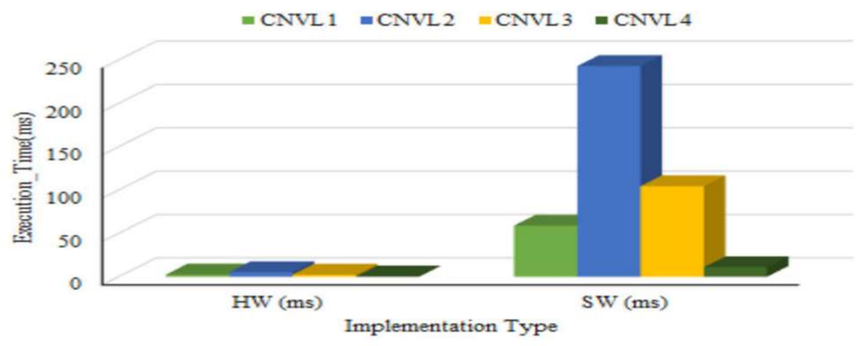
Applications mentioned

•**RRANN (Eldredge & Hutchings)**: Run-time reconfigurable neural network on X12 board with 12 XC3000 FPGAs; dynamically shared resources across three backpropagation stages, achieving 500% savings. However, 80% of execution time was spent on reconfiguration, highlighting the need for optimization.
•**Partitioned RRANN**: Improved RRANN efficiency through physical partitioning of static and dynamic regions, increasing neuron occupancy by 50% and reducing reconfiguration time by 25%, making resource usage more effective.
•**Virtual Hardware Manager (Burns et al.)**: Developed for XC6200 FPGA to manage runtime tasks, including netlist generation, circuit relocation, and dynamic interrupt handling, improving scheduling efficiency and optimizing systolic FIR structure reconfiguration.
•**Reconfigurable Development Platform**: XC6200-based system with an XC4013 FPGA as the controller; demonstrated dynamic circuit switching and design simulation using PCI-based reconfiguration for efficient external setups.
•**Dynamic DES Encryption (JBits)**: Implemented dynamic reconfiguration of encryption keys on Virtex FPGAs to enhance performance in speed, power efficiency, and area usage; faced limitations in debugging and required expert users.
•**Evolvable Image Filter**: Developed on Xilinx Zynq SoC with DDR memory for reconfiguration; used evolutionary algorithms for adaptable image filtering but struggled with the larger frame size on Zynq compared to Virtex platforms.
•**PR with Compression**: Applied bitstream compression algorithms on Xilinx FPGAs to reduce configuration size and improve reconfiguration speed, though the approach was effective only for small dynamic regions.

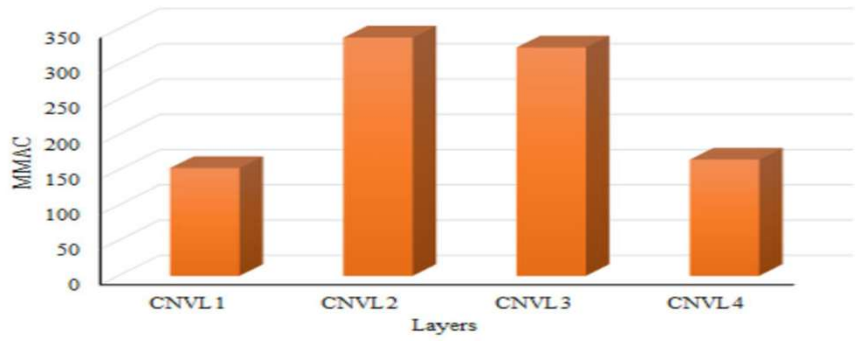
FPGA-based SoC design for real-time facial point detection using deep convolutional neural networks with dynamic partial reconfiguration

GPU & HLS Approach

•F1 network was implemented on a GPU which achieved 89.01%, precision of 91.63%, and recall of 90.25%
•The power consumption in GPU was very high.
•Hardware acceleration using HLS significantly improved execution times for CNVL layers, with the second CNVL layer being accelerated by 45.01 times and convolution 1 by 41.8 times compared to software.
•Implemented on Pynq-Z1 MPSoC FPGA. Where 2 HDMI ports are configured for Video input and output through the PL.
•Processing system of the SOC receives the image through the HDMI port via a DMA buffer.
•The PS does preprocessing where the Faces are detected, designated with a bounding box, and cropped from the picture in the first phase.
•This method is very promising but the resource utilization is very high and also power consumption is high



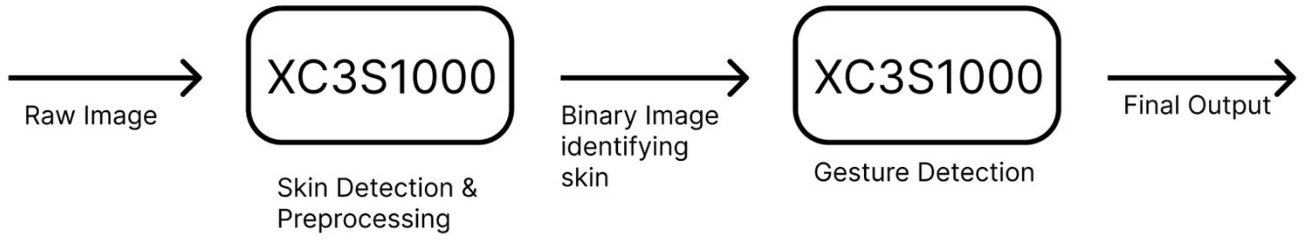
(c) Execution time of the CNVL IPs in HW and in SW



(d) Mega Multiply-Accumulate Operation per second

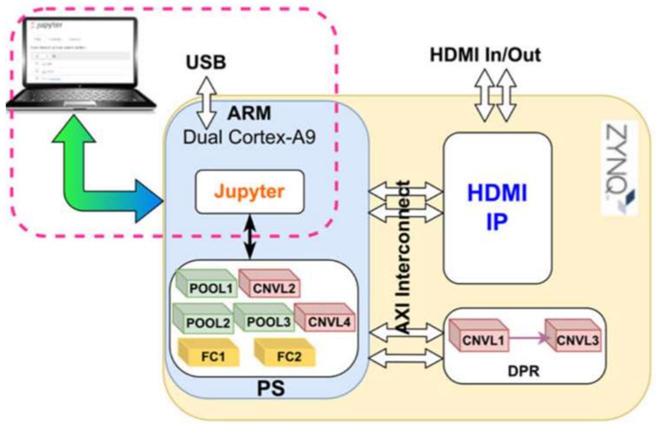
Multi-FPGA cluster

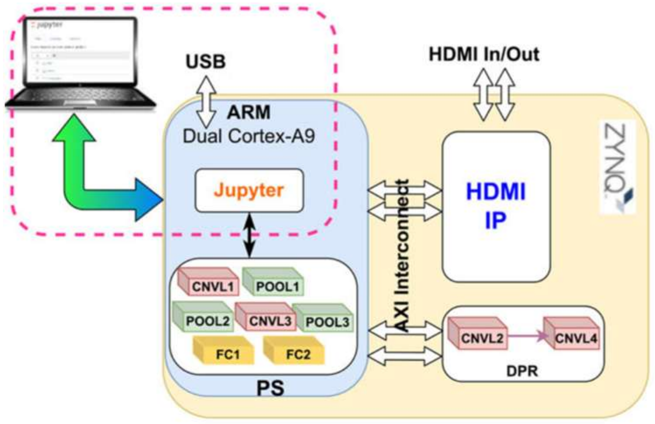
- The PYNQ boards are being used in this cluster as they are economical Zynq based boards.
- Implementation of ResNet-50 on a 4-FPGA cluster showed performance of 75FPS throughput & 292 GOPS performance.
- This performance was much faster than CPUs and much more efficient than GPUs
- **Cascading** low-end FPGAs to split the workload. Hand Gesture Recognition(HGR) was implemented on 2 Spartan Boards.
- System could detect 5 hand gestures with 97.3% accuracy.



Dynamic Partial Reconfiguration Optimization

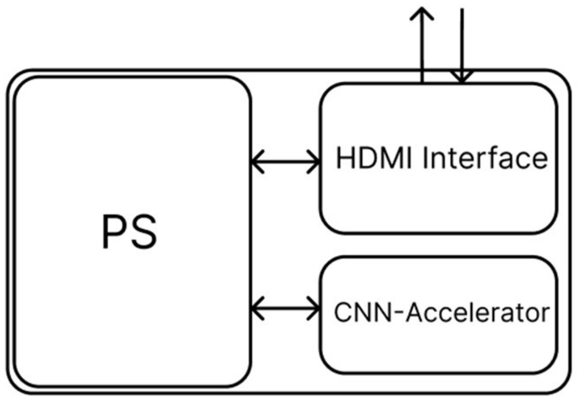
•Based on Reconfig-time and complexity of the layer they are moved into PS & PL.
•Strategy 1 CNVL1 IP & CNVL3 are moved into PL while CNVL2 & CNVL4 are in PS. This acceleration can speed up the software run time by 1.5 times.
•Strategy 2 CNVL2 IP & CNVL4 are moved into PL while CNVL1 & CNV34 are in PS. This acceleration can speed up the software run time by 2.3 times.

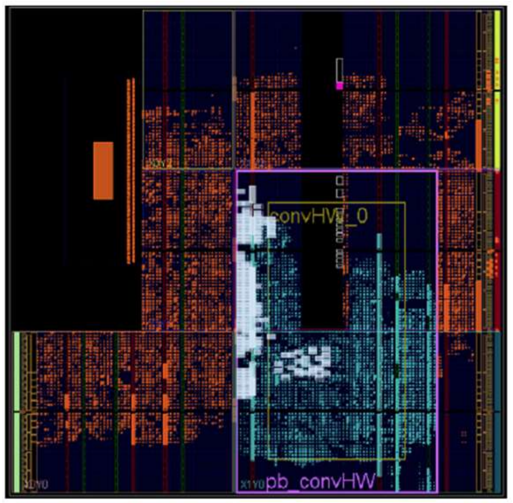




Dynamic Partial Reconfiguration Observations

•In this approach final benchmarks are worse than the software implementation(CPU) which is about 440mS.
•The results showed execution time exceeding 500mS.
•Though the CNVL IP has a mean computation time of 2.6ms. Majority of the time is spent on reconfiguration. whose is mean is 137ms hence resulting is high computation time.





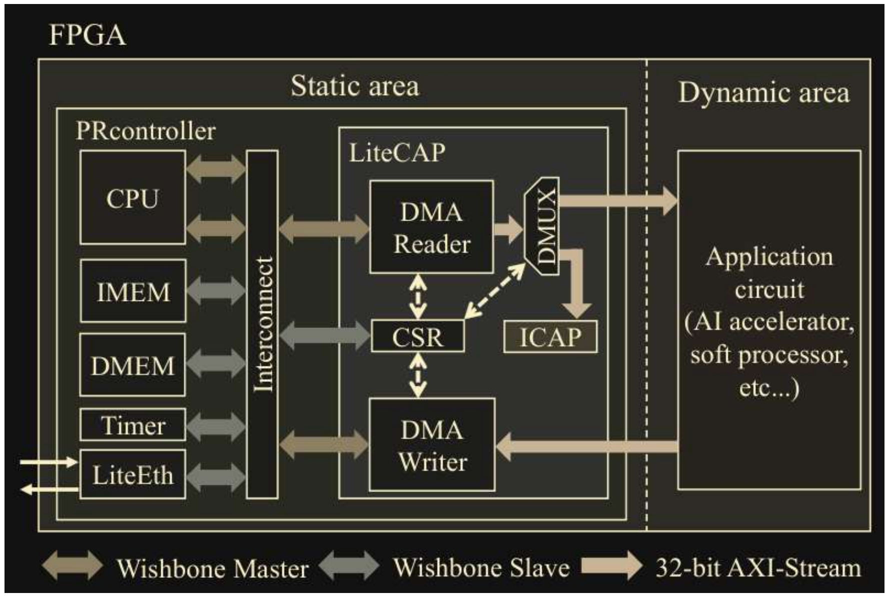
FPGA-based SoC design for real-time facial point detection using deep convolutional neural networks with dynamic partial reconfiguration

Introduction

•**Conventional Approach to Partial Reconfiguration**
•Typically, FPGAs are paired with an embedded processor (e.g., ARM Cortex or RISC-V) to manage partial reconfiguration.
•This setup increases hardware costs, reducing scalability.
•**Alternative FPGA Control Mechanism**
•An alternative approach involves using an external server to control the FPGA via JTAG cables.
•However, this is impractical due to signal degradation and noise over long distances.
•**Proposed Solution**
•The FPGA is partitioned into **static** and **dynamic** regions.
•The **static region** hosts the Partial Reconfiguration (PR) controller.
•The **dynamic region** accommodates reconfigurable application circuits.
•**ReON: Partial Reconfig Protocol:** A network partial configuration framework that was apt for high end FPGAs. (Not suitable for low end FPGAs)
•**Technical Details**
•**Partial Bitstreams:** Partial bitstreams are the bitstreams created by the EDA tool to implement Partial Configuration on an FPGA.
•**LiteX:** Generates a RISC-V soft-core for FPGA implementation.

Proposal

- **FPGA Fabric Partitioning:**
- The FPGA is divided into **static** and **dynamic** sections.
- The **static region** houses the Partial Reconfiguration (PR) controller.
- The **dynamic region** hosts the reconfigurable application circuit.
- **TCP-Based Communication**
- Two dedicated TCP connections facilitate interaction between the host and FPGA:
    - One for transmitting the partial bitstream.
    - Another for sending data to the configured application circuit.



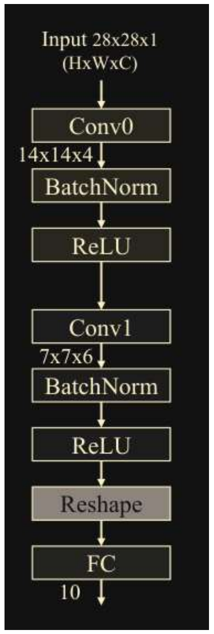
Verification/Testing

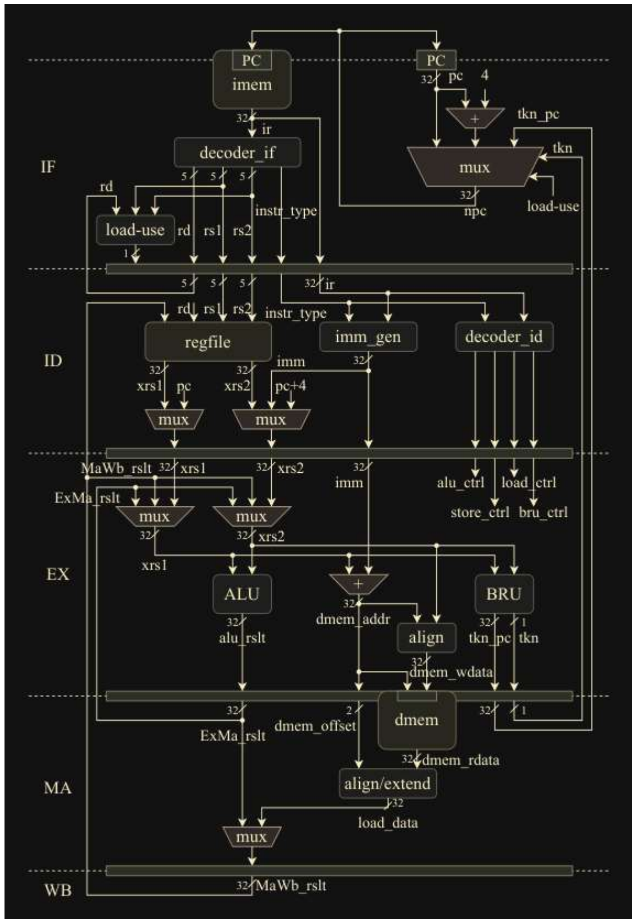
Environment

- **FPGA Board:** Arty A7-35T
- **Compiler:** riscv64-unknown-elf-gcc 12.2.0
- **Frameworks:** LiteX, VexRiscv (RV32I)

Test Cases

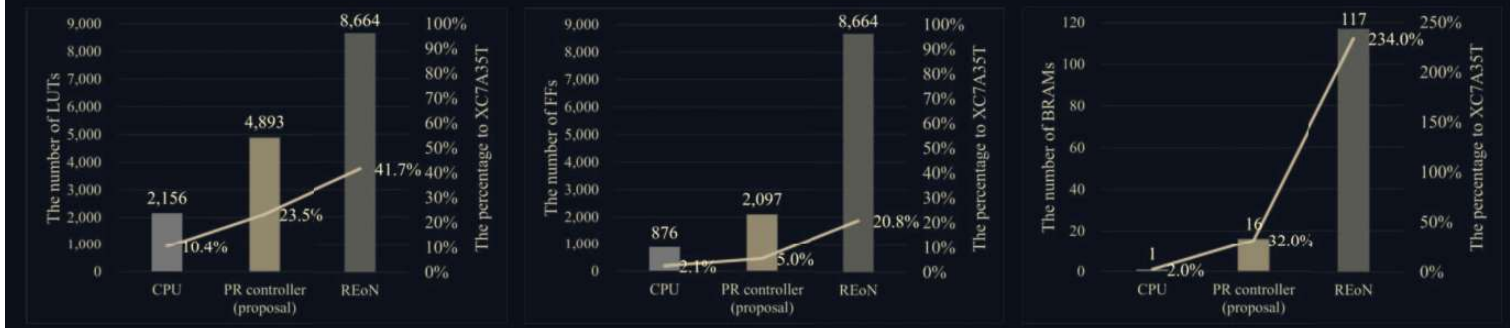
1. **CNN Network (MNIST Classification)**
   1. Implemented using FINN on FPGA.
2. **RISC-V Soft Processor**
   1. In-order execution in Verilog.
   2. "Hello, World!" output via AXI-Stream.





Evaluation Results

- CPU - With a CPU on board the FPGA
- REoN - Network protocol for partial configuration on high end FPGAs
- PR Controller - The static partition of our SoC
- **Conclusion**:
- A streamlined version of existing protocols and modules is implemented to optimize resource utilization on low-end FPGAs (e.g., Arty vs. Zynq).
- The proposed architecture is well-suited for data centers and **Network Function Virtualization (NFV)** applications.
- By eliminating the need for an external CPU or controller, the design enables standalone FPGA operation, enhancing scalability and deployment efficiency.



(a) LUT Utilization. **(b)** FF Utilization. **(c)** BRAM Utilization.

Fig. 7: The amount of hardware used by CPU of the proposed SoC, the whole static area, and REoN.