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:** Uses large, pre-designed blocks (ALUs, multipliers) for specific tasks.
- **Rigid Architecture:** Prefabricated with limited adaptability.
- **Array Structures:** Linear, crossbar, or mesh for routing and parallelism.
- **Low Flexibility:** Fixed design restricts evolving applications.
- **High Area Usage:** Large macros consume significant chip space.
- **Static Configuration:** No runtime modifications possible.
- **Low Code Density:** Inefficient for granular customization.
- **Limited Reusability:** Poor adaptability across applications.
- **Obsolete for Modern Needs:** Lacks real-time adaptability and high logic density.
- **Applications:** Used in high-performance signal processing, multimedia, and wireless communications.

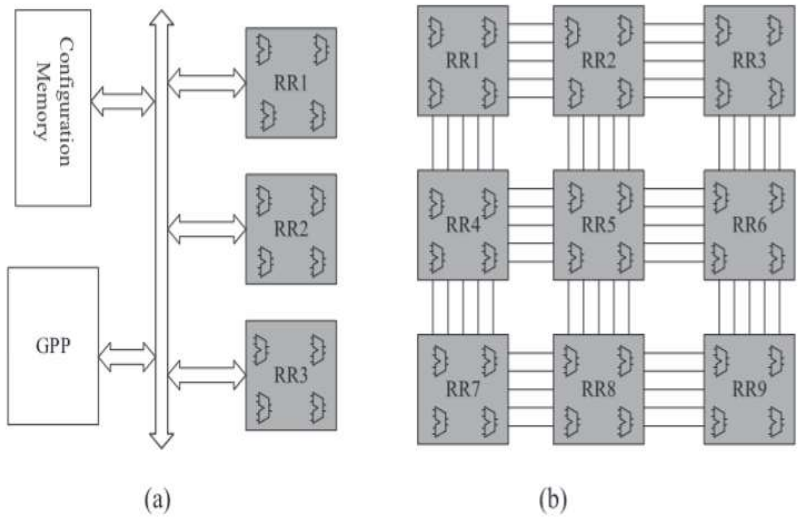


Fig. 1. (a) Standard CGRA (b) Distributed CGRA.

FGRHS

- **Definition:** 1-bit or logic-element reconfiguration via CLBs for precise customization.
- **High Flexibility:** Adapts to diverse, evolving applications.
- **Precise Optimization:** Granular control over functionality.
- **Reusability:** Shares resources across multiple tasks.
- **On-Chip Integration:** Combines logic, processors, and memory.
- **Industry Standard:** Unmatched adaptability for dynamic workloads.
- **Efficiency:** Enables partial reconfiguration, reducing downtime.
- **Tool Support:** Xilinx PlanAhead simplifies implementation.
- **Superiority:** More flexible and efficient than Coarse-Grain systems.
- **Applications:** AI accelerators, signal processing, multimedia, fault-tolerant hardware.

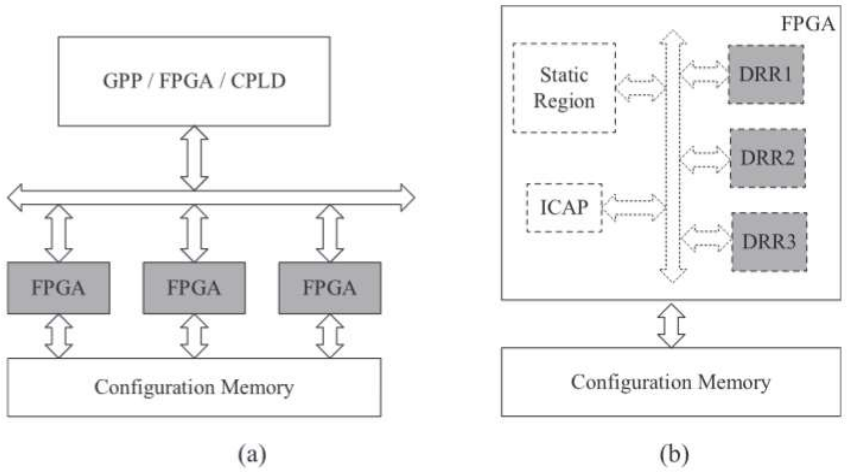


Fig. 2. (a) External FGRA (b) On-chip FGRA.

Partitioning and Reconfiguration Techniques

Partitioning: Enables modular design and efficient resource use.
Static Region: Fixed logic remains active during reconfiguration.
Dynamic Region: Reconfigurable area tailored per application.
Full Reconfiguration: Entire FPGA update, time-intensive but fixed.
Partial Reconfiguration: Updates specific regions, reducing downtime.
Internal Reconfiguration: ICAP enables fast on-chip updates.
External Reconfiguration: SelectMAP/JTAG configures from external sources.
Throughput Bottlenecks: Limited by memory and controller speed.
Granularity Trade-offs: Fine-grain systems add flexibility but slow updates.
Tool Support: Xilinx PlanAhead and GoAhead aid PR implementation.
Future Directions: Secure remote PR, real-time tools, AI/cloud integration.

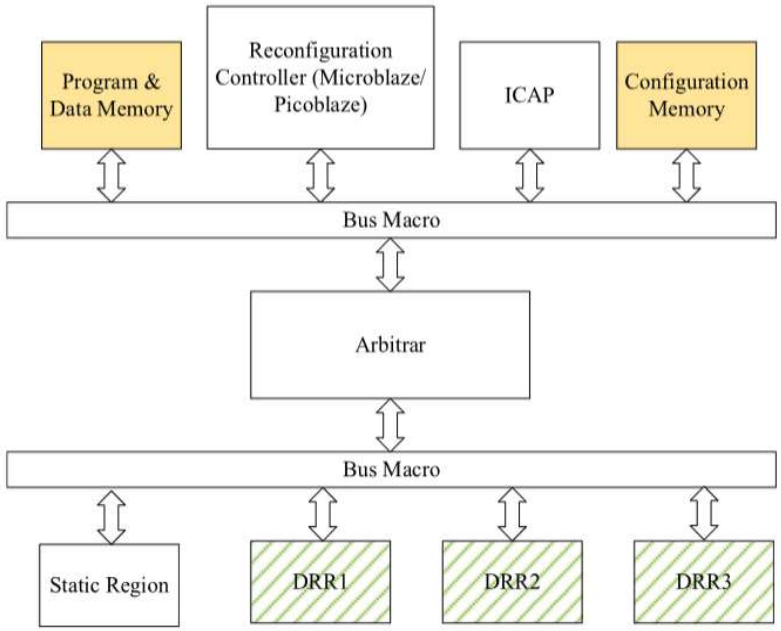


Fig. 5. Two bus architecture of On-chip FGRHS.

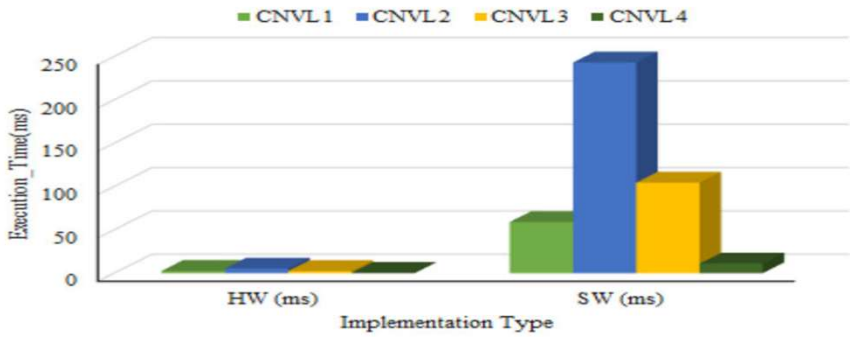
Applications mentioned

- **RRANN:** 500% resource savings but 80% time spent on reconfiguration.
- **Partitioned RRANN:** 50% more neurons, 25% faster reconfiguration.
- **Virtual Hardware Manager:** Optimized scheduling and FIR reconfiguration.
- **Reconfigurable Platform:** PCI-based dynamic circuit switching.
- **DES Encryption:** Faster, efficient, but hard to debug.
- **Evolvable Filter:** Adaptive but struggled with large frames.
- **PR with Compression:** Faster but limited to small regions.

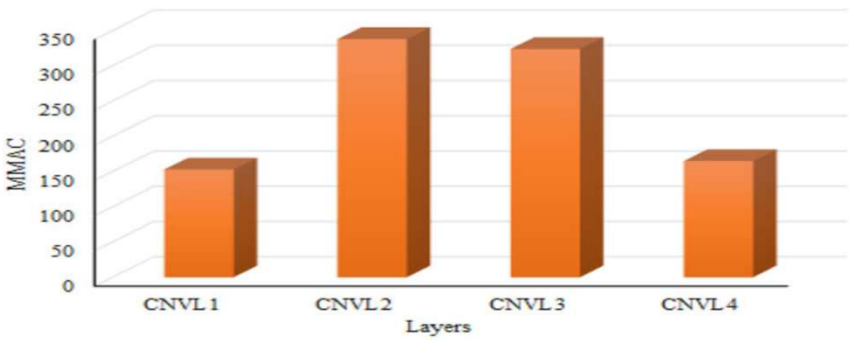
FPGA-based SoC design for real-time facial point detection using deep convolutional neural networks with dynamic partial reconfiguration

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



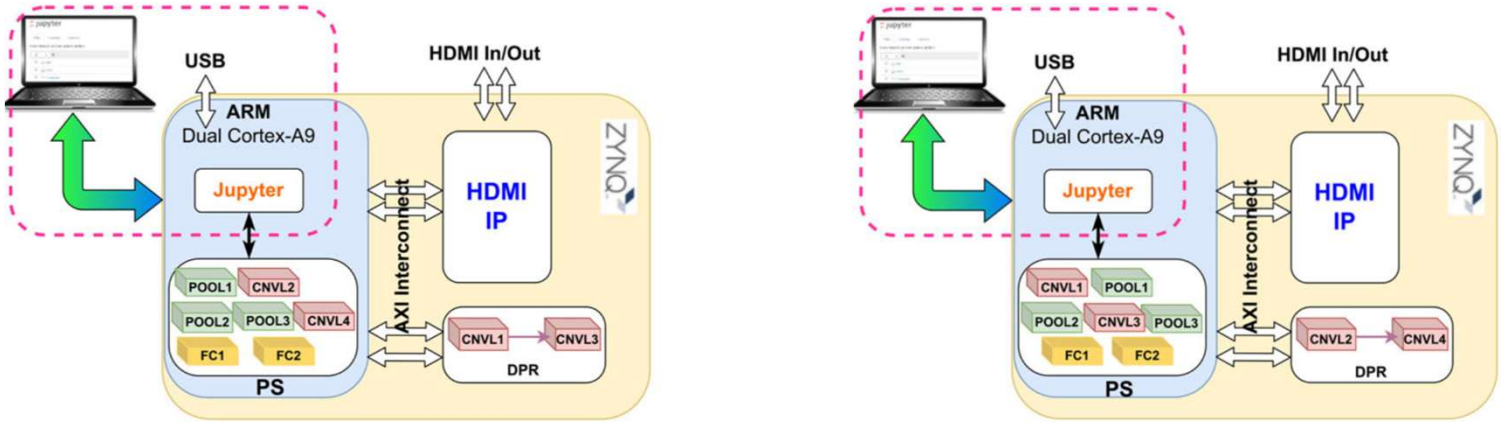
(d) Mega Multiply-Accumulate Operation per second

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.

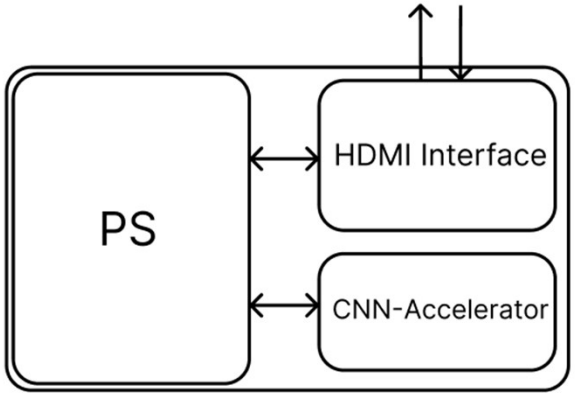
Dynamic Partial Reconfiguration Optimization

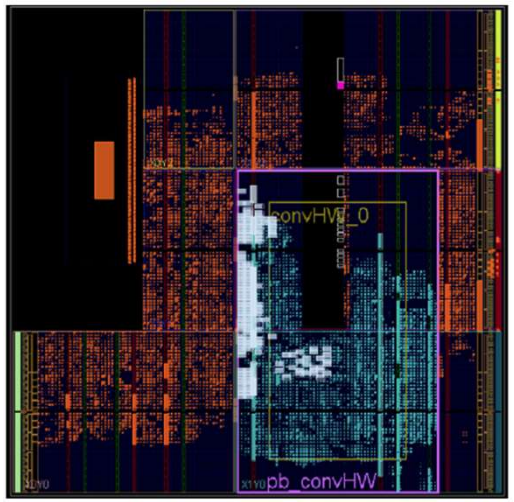
•Based on Reconfig-time and complexity of the layer they are moved into PS & PL.
•Strategy 1 CNVL1 IP & CNVL3 are moved into PL while CNVL2 & CNVL4 are in PS. This acceleration can speed up the software run time by 1.5 times.
•Strategy 2 CNVL2 IP & CNVL4 are moved into PL while CNVL1 & CNV34 are in PS.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





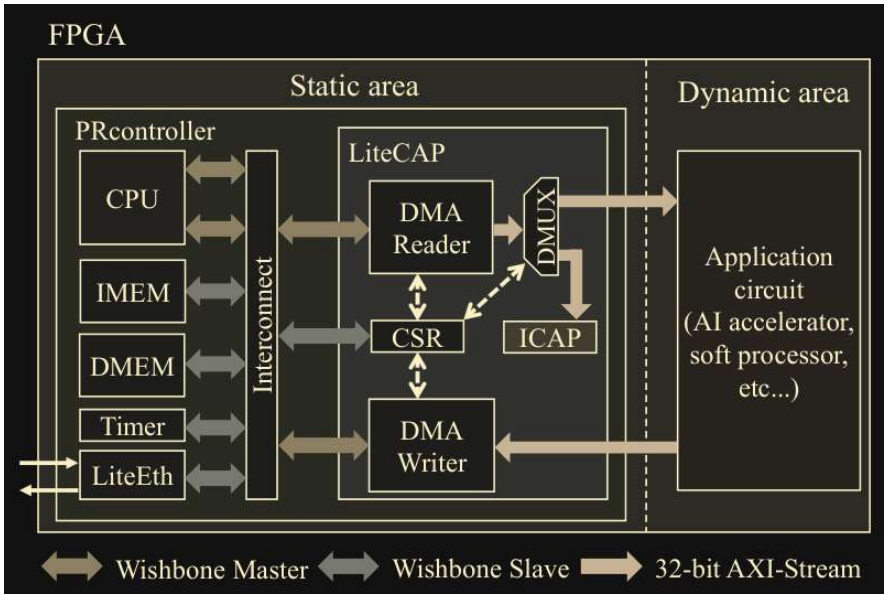
A remote partial-reconfigurable SoC with a RISC-V soft processor targeting low-end FPGAs

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)

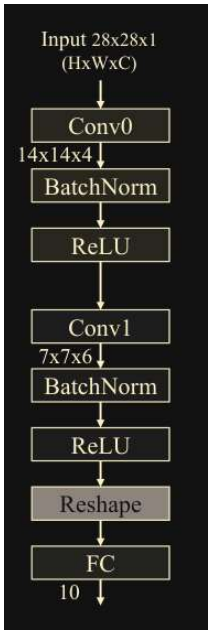
Test Cases

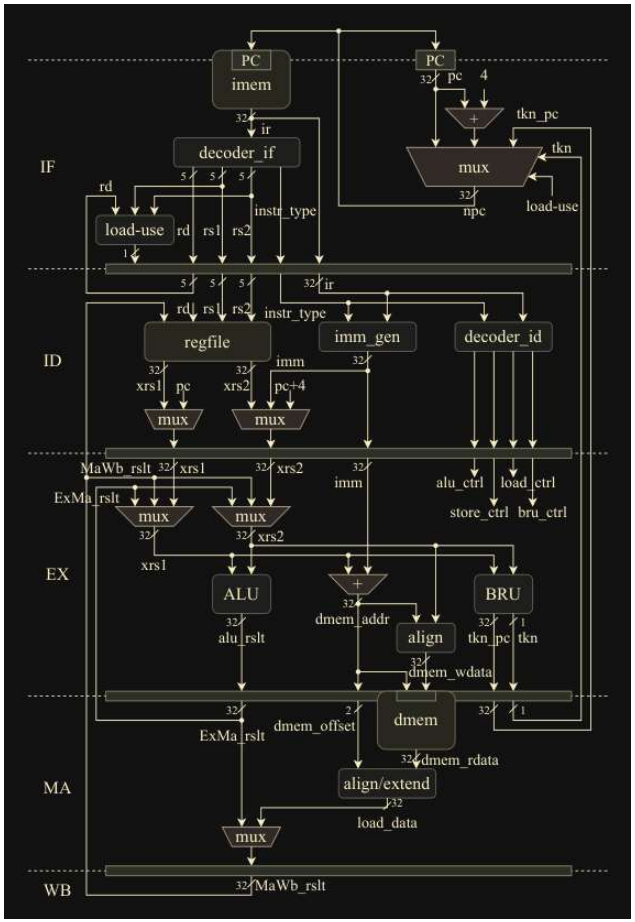
1.CNN Network (MNIST Classification)

1. Implemented using FINN on FPGA.

2.RISC-V Soft Processor

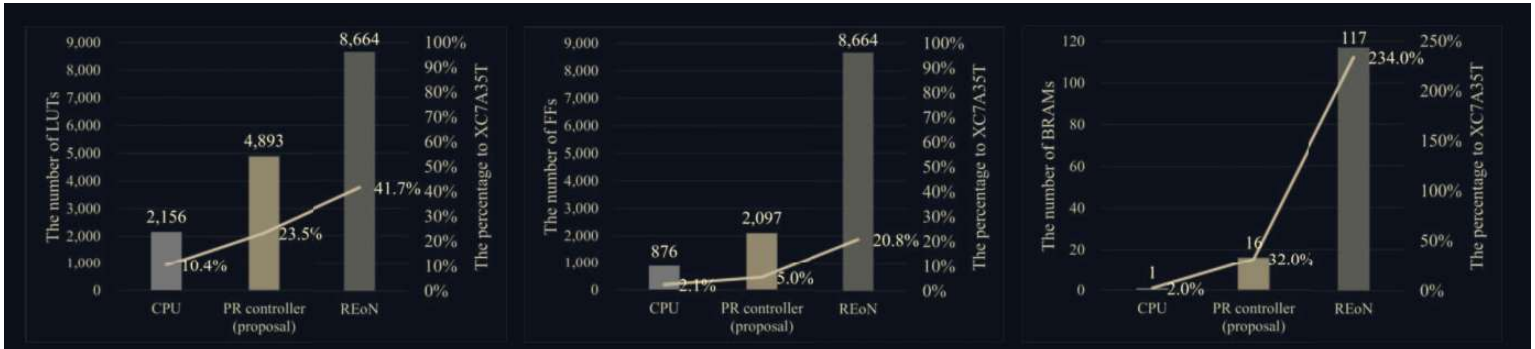
1. In-order execution in Verilog.
2. "Hello, World!" output via AXI-Stream.





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.