FPGA Capestone

Paper Presentation Summary

FPGA Dynamic and Partial Reconfiguration: A Survey of Architectures, Methods, and Applications

KIZHEPPATT VIPIN, Nazarbayev University, Kazakhstan
SUHAIB A. FAHMY, University of Warwick, United Kingdom

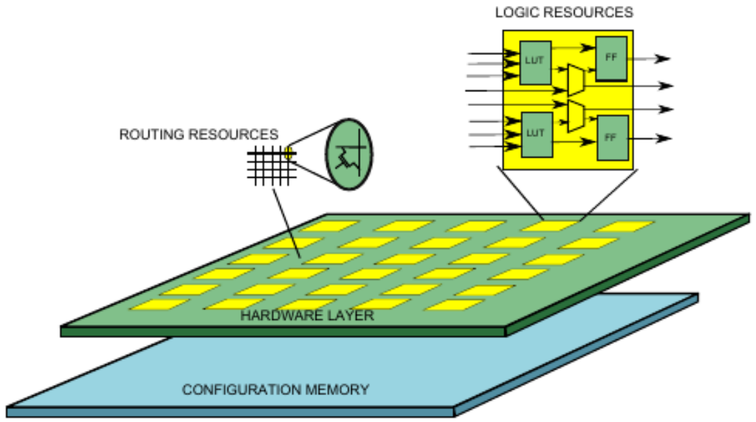
https://doi.org/10.1145/3193827
ACM Computing Surveys: Published: 25 July 2018

Table of Contents

Introduction

Conceptual Layers of a FPGA

- The device can be conceptually split into Hardware layer & Configuration Memory layer
- Hardware layer contains components like LUTs, DSPs, Memory, Transciever, Switches
- Configuration layer contains the config information like routing, initial memory values, set/resets
- Modern FPGAs use SRAMs for configuration layer, therefore are volatile
- Context switching FPGAs perform dynamic reconfiguration but not partial configuration. PR is supported in Xilinx boards through external ports like ICAP



Introduction

Why Partial Configuration

- Higher logic density (functionality of modules can be time multiplexed with lesser hardware)
- Lesser reconfiguration time due to a lesser area for reconfiguration
- Reduced interconnect downtime during reconfiguration
- PR can adapt computation to a changing environment while continuing to process data.
- With increasing logic capacity in modern devices, higher logic density is no longer a major driving factor, but the rest are still valid

Introduction

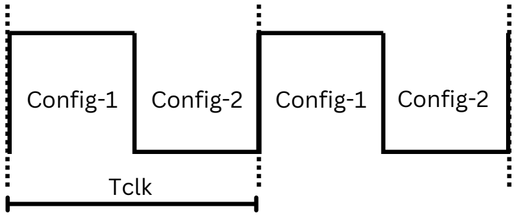
Device features favorable for Partial Reconfiguration

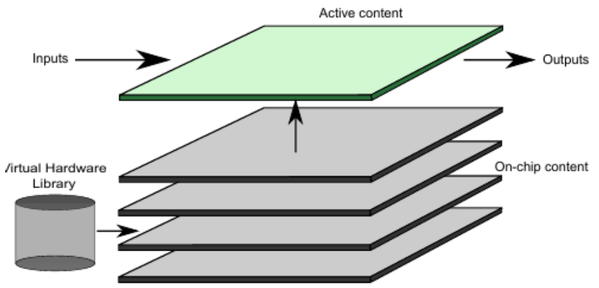
- Support for granularity
- Support for runtime relocation
- Negligible reconfiguration overhead
- Reconfiguration process should be transparent to other applications
- High level tool to automate mapping

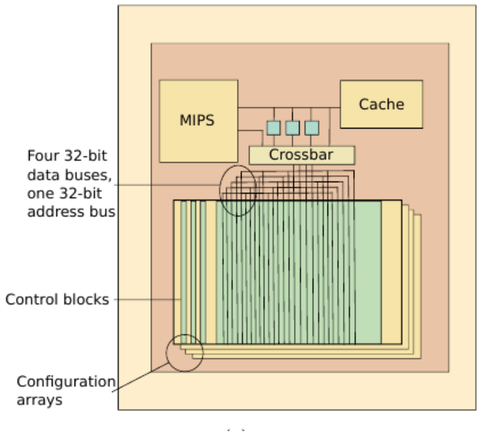
Architecture

Academic and Non-Commercial Architectures

- Initial Design with 2 configuration memory arrays
- DPGA - Dynamically Programmable Gates Array
- Multi-Context FPGAs - FPGAs designed with multiple configuration memory planes, to address faster context switching
- GARP Architecture - Combining a Processor with reconfigurable array





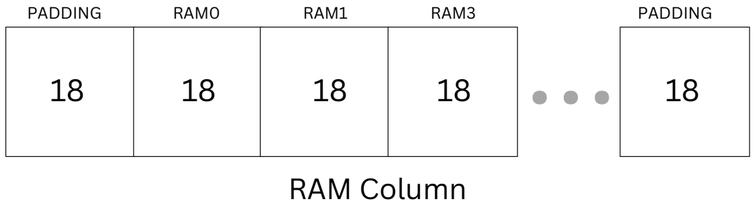


Architecture

Frames

- Smallest unit of configuration that can be read or written
- In the initial design, each frames height is same as the device height and 1-bit wide

CLB Column



RAM Column

- Frames configure a narrow single bit slice of multiple hardware resources
- Since frames configure a narrow slice of resources, they enable efficient partial reconfiguration.
- This makes dynamic updates possible while keeping the rest of the design functional.

Architecture

Configuration Methods:

- Select Map - 8,16,32 bit interface
- Master/Slave - Serial Interface

Ports:

- ICAP - Internal Configuration Access Port
- PCAP - Processor Configuration Access Port
- MCAP - Media Configuration Access Port (through PCIe)

Architecture

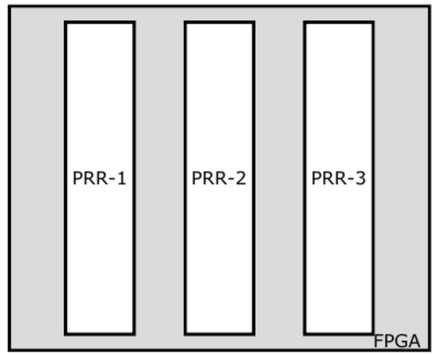
Run-time Relocation Architecture

- Due to the heterogeneous architecture of FPGA runtime relocation is difficult to achieve
- One method to resolve this issue is to separate them into 2 layers:-
  - Homogeneous - LUT, Flip-Flops & associated interconnects
  - Heterogeneous - DSP, Memory, RAM & routing
- A concept similar to rails is utilized, where heterogeneous computing elements are connected to horizontal rails, allowing them to move along these lines.
- However, this increases horizontal delay, which arises due to the relative position of these components being unknown during place and route.

Architecture

Xilinx Virtex PR Architecture

- Virtex
  - Frames extend vertically throughout the height of device and are 4 CLBs wide
  - Uses Slot Machine Architecture
  - All PRR should have similar interface with Static region
  - Tri-State buffer are used as anchor logic(to avoid glitches during reconfiguration & floating nodes/Static Logics in PRR)
  - Limitation - Number of buffers and fixed location
  - Virtex 4 overcomes this issue by using LUTS instead of buffers



Architecture

Concept of Tiles

- To minimize the need for full-height reconfiguration of the FPGA, the die is divided into multiple tiles, allowing for more localized and efficient partial reconfiguration.
- Each tile configurable logic blocks (CLBs), DSPs, memory blocks, and interconnects

- Tile Size of Virtex and 7 Series FPGA:

	CLB Tile	DSP Tile	BRAM Tile
Virtex – 5	20	8	4
Virtex – 6	40	8	8
7 Series	50	10	10

Architecture

Altera

- Altera uses LUT-Based proxy logic for preserving routing between PRR and Static regions
- Two different implementations SCRUB mode and AND/OR method
- SCRUB mode:
  - Re-Writing LABs(Logic Array Blocks) irrespective of previous frame. The static regions are also scrubbed back.
- AND/OR reconfiguration:
  - First Iteration, bits corresponding to PRR in frames are set to 0 using AND
  - In Second pass for each frame new data is ORed with current value of 0 in PRR and static region bits are ORed with.
- Altera uses a PR-IP to transfer data into configuration memory

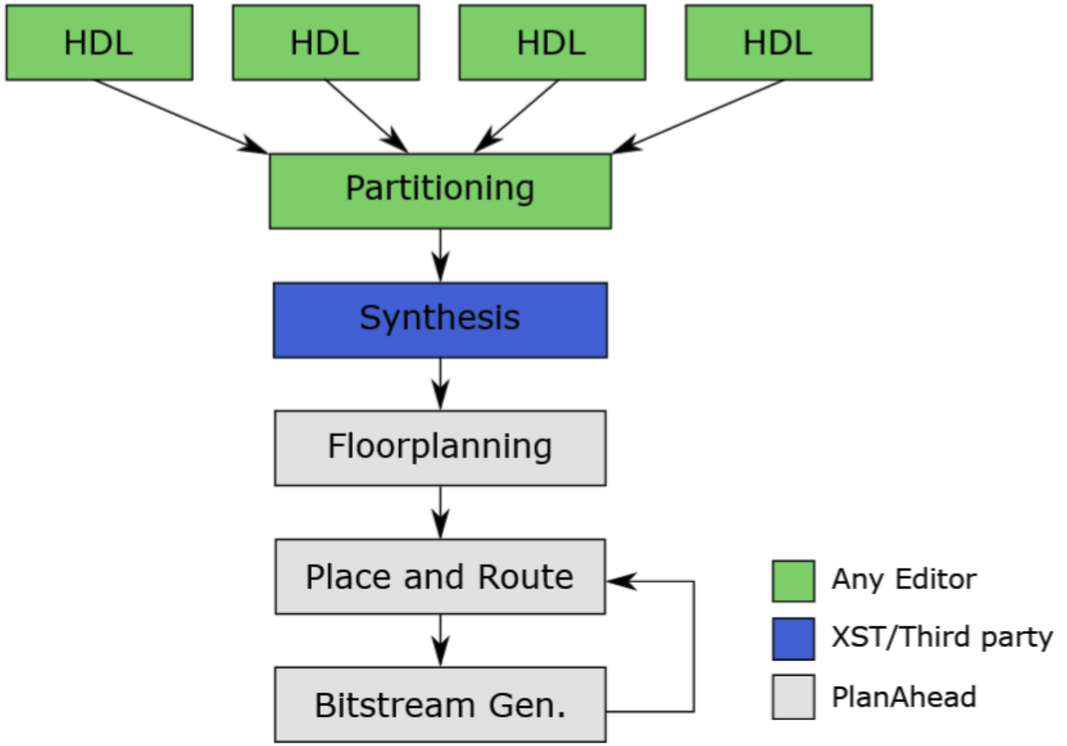
Architecture

Other Vendors

- Lattice Semiconductors: PR was done by setting a bitstream option in the previous configuration seq that would tell the FPGA not to reset all of the config RAM during reconfiguration
- Atmel - architecture was a symmetrical array of identical cells with bus repeaters spaced between every 4 cells. Configuration memory was viewed as a simple memory-mapped address space and user had full read/write access.
- National Semiconductors - They used CLAy logic cell which implemented simple functions like NOR, AND, NAND, FFs etc. each cell had 2 direct connections to each of the 4 nearest neighbors and connections to horizontal and vertical buses
- Tabula - used Spacetime technology similar to MC-FPGAs. The Spacetime compiler automatically mapped , paced and routed a used design into the device.

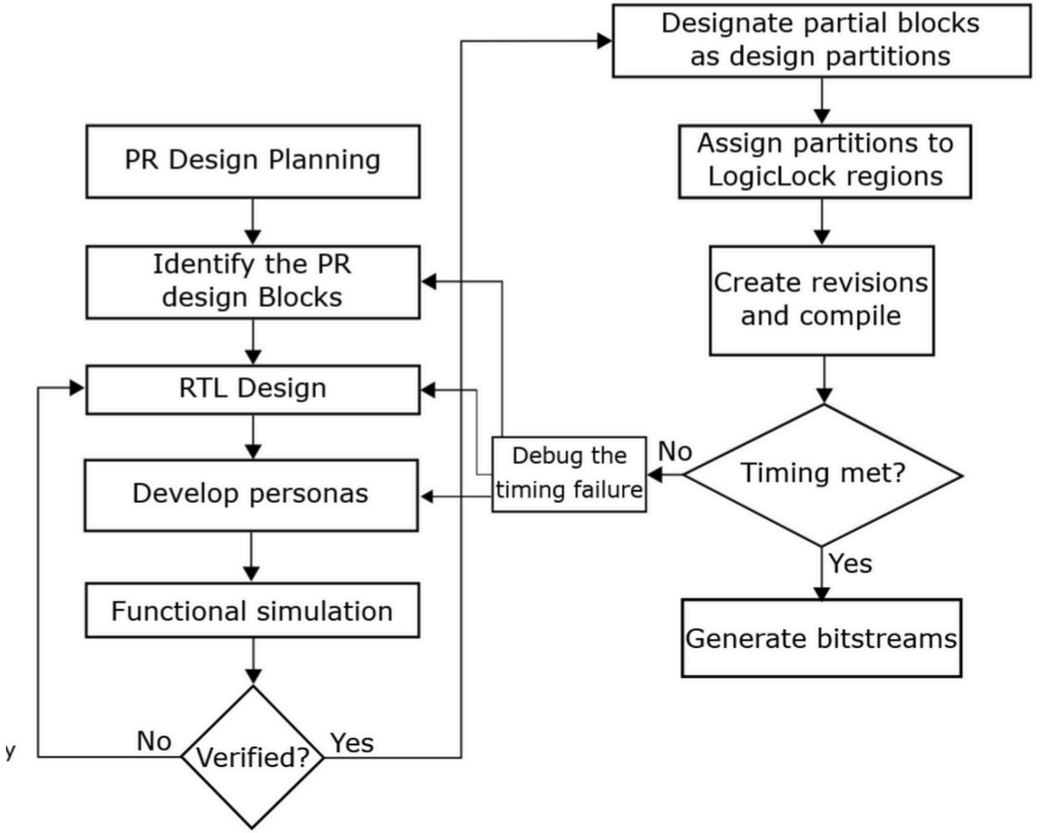
Design, Implementation & Simulation Tool chain Xilinx PR Flows (PlanAhead + Vivado)

- **PlanAhead Highlights:** Module-based design, static region + PRRs, LUT-based bus macros, manual floorplanning (UCF), rectangular PRRs aligned to clock region boundaries
  - **Technical Details:** Static can use PRR routing resources, LUT bus macros auto-inserted, full & partial bitstreams generated
  - **Limitations:** PRRs exclude clock logic (PLLs, buffers), auto bus macro placement, routing complicates relocation, requires low-level FPGA knowledge
- **Vivado Enhancements:** Tcl-based flow, interconnect tiles replace bus macros, partition pins auto-inserted, improved routing/timing
  - **Technical Details:** ICAP for non-hybrid devices, PCAP for Zynq, no crossing interconnect tiles, DRC checks for PR violations
  - **Limitations:** Static region wires through PRRs require full reimplementation, run-time relocation difficult, potential routing congestion
- **ICAP Controller:** Required for reconfig; simulation model only provides status, not actual reconfiguration

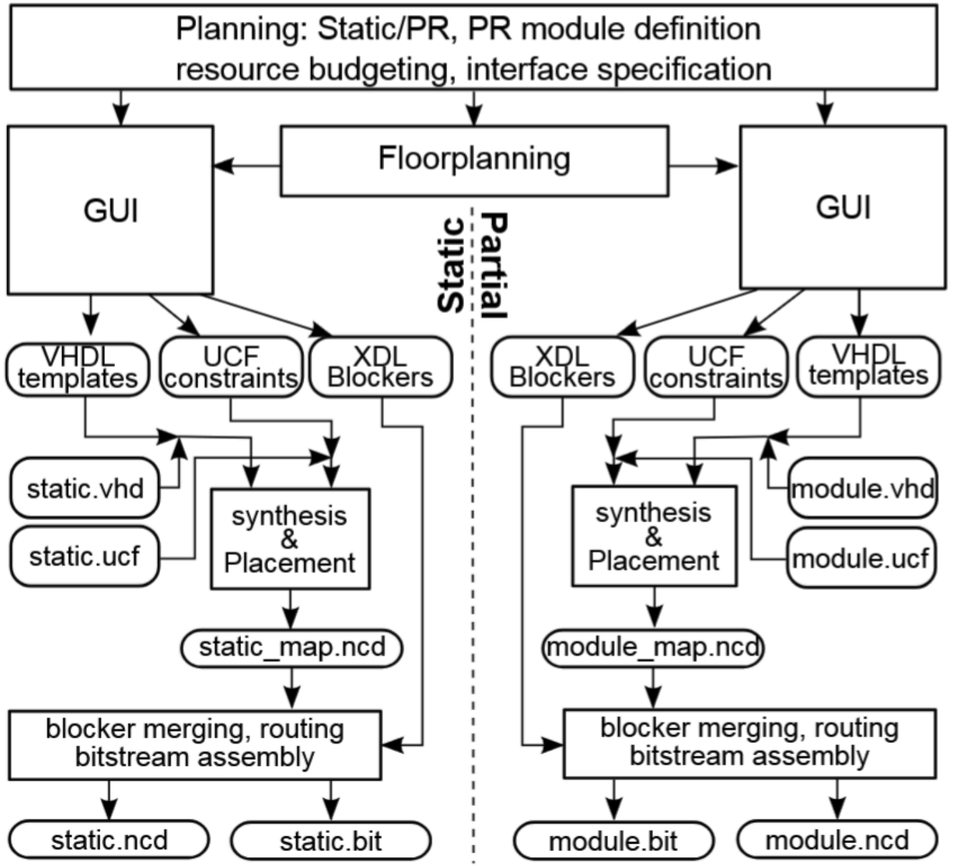


Design, Implementation & Simulation Tool chain
Altera PR Flow & Runtime Control

- **Flow Overview:** LogicLock PRRs, incremental compilation, LUT-based anchor logic auto-inserted, vertical PRRs through FPGA height (if not full-width)
  - **Technical Details:** Configuration frames = programming frames, base revision sets PR boundaries, partial files for other revisions
  - **Limitations:** Full config file only for base revision, RAM init restricted in AND/OR mode, Logic 1 must be written before reconfig
- **Runtime Details:** CRC error detection during reconfig, freeze logic signal to prevent glitches, configuration error if memory not pre-written, unique to Altera architecture

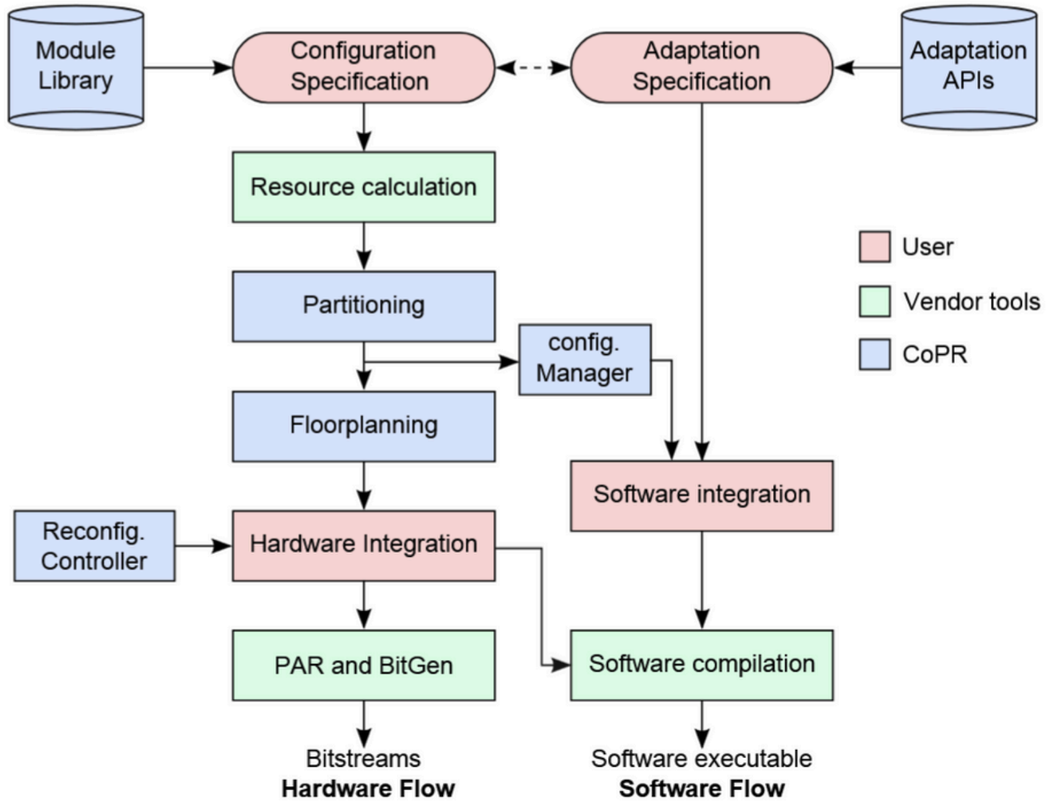


Design, Implementation & Simulation Tool chain
Academic Tools – GoAhead



- **GoAhead:** GUI floorplanner, blocker macro insertion for routing isolation, uses vendor tools for final clock tree generation, automatic floorplanning via connectivity analysis
  - **Technical Details:** Direct manipulation via XDL, macro definitions for PR interfaces, clock control via specialized macros
  - **Drawbacks:** XDL deprecated, complex UI, not compatible with Vivado designs

Design, Implementation & Simulation Tool chain
Academic Tools – CoPR



- **CoPR:** High-level XML + C config, automatic partitioning by resource needs, kernel tessellation for floorplanning, Zynq focus with PCAP reconfig manager
  - **Technical Details:** Auto-generated reconfig manager, abstracted API, C-based adaptation, ARM processor integration
  - **Drawbacks:** Only supports ISE, limited low-level control, Zynq-specific

Design, Implementation & Simulation Tool chain
Additional Academic Tools

- **OSSS+R:** SystemC-based, object-oriented modeling, polymorphic objects for PR, simulates reconfig cost, context switch modeling, VHDL via Fossy synthesis
  - **Drawbacks**: Manual wrappers, no automated floorplanning, limited modern FPGA support, dependent on vendor tools
- **UML-based Modeling:** Standard UML profiles for hardware, RTL interface, embedded constraints, design space exploration
  - **Drawbacks**: Manual floorplanning still required, complex toolchain integration
- **Caronte:** Task-graph driven, runtime task scheduling, embedded processor for dynamic task loading, fixed pre-defined regions
  - **Drawbacks**: No partitioning capabilities, lacks dynamic floorplanning, basic infrastructure

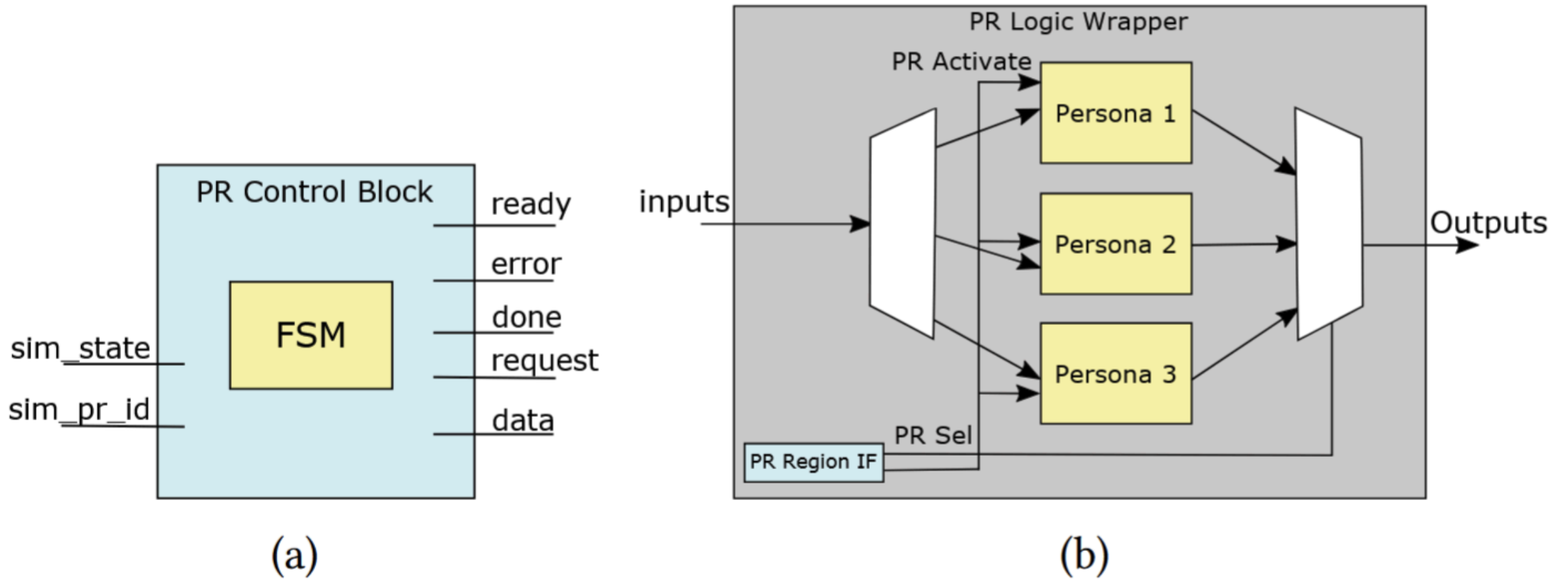
Design, Implementation & Simulation Tool chain
Additional Academic Tools

- **GePaRD:** High-level synthesis enhancement, system model simulation, template abstraction, temporal modularization, physically-aware architecture description
  - **Drawbacks:** Limited practical implementation details, early-stage tool
- **Fahmy et al.:** Two-plane separation (data: hardware, control: software), IP library-based composition, software-managed reconfig
  - **Drawbacks:** Single PRR only, low bandwidth software-hardware interface
- **Navas et al.:** IP block integration with reconfig infrastructure, predefined interfaces (similar to OpenCL), unified software and reconfig interface
  - **Drawbacks:** Resource-based module hosting model, lacks dynamic adaptation

Design, Implementation & Simulation Tool chain

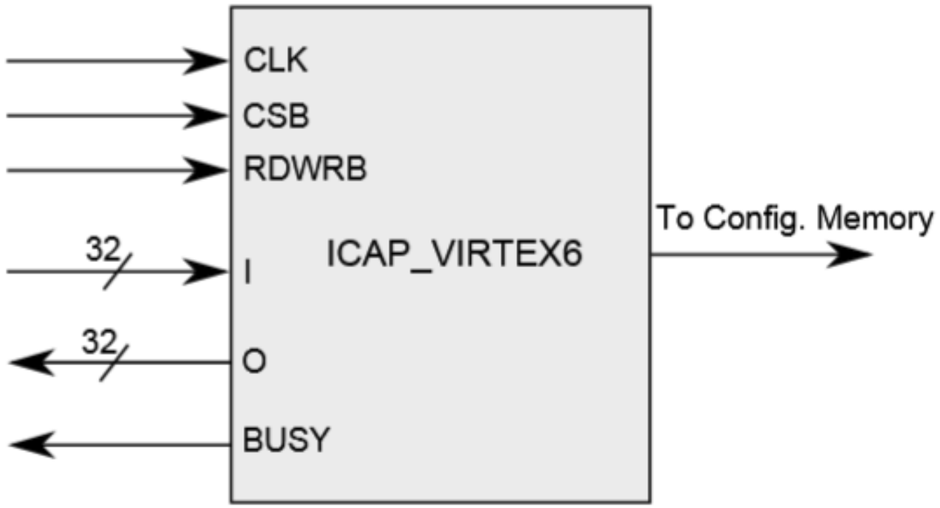
Simulation – Altera Multiplexing Approach

- **Vendor Sim Limits:** No config memory or frame write modeling, no transition state support, manual mux workaround, ICAP sim model status-only
- **Multiplexing (Luk et al.):** Signal-based switching, no isolation logic, no timing accuracy, manually implemented, verifies only before/after states
- **Dynamic Circuit Switching (DCS):** Auto mux insertion post-synthesis, tri-state buffer isolation, scheduler monitors control signals, activates muxes
  - **Limitations:** Config file transfer not simulated, simplified timing, no bit-level accuracy



Design, Implementation & Simulation Tool chain
Simulation – Xilinx ICAP Focus

- **ReSim Library:** Simulates ICAP controller, SIMB format with module IDs, error injection (random/ targeted), configurable timing parameters, SystemVerilog for portability
  - **Technical Details:** Models config memory read/ write, control/status/error signals, portable across simulators, supports functional & preliminary timing verification
  - **Limitations:** Simulation performance overhead, abstracted memory model, limited architecture-specific fidelity
- **ICAP Controller:** Real ICAP provides reconfig control/status, but sim model lacks full configuration behavior



(c)

Design, Implementation & Simulation Tool chain Vendor Feature Comparison

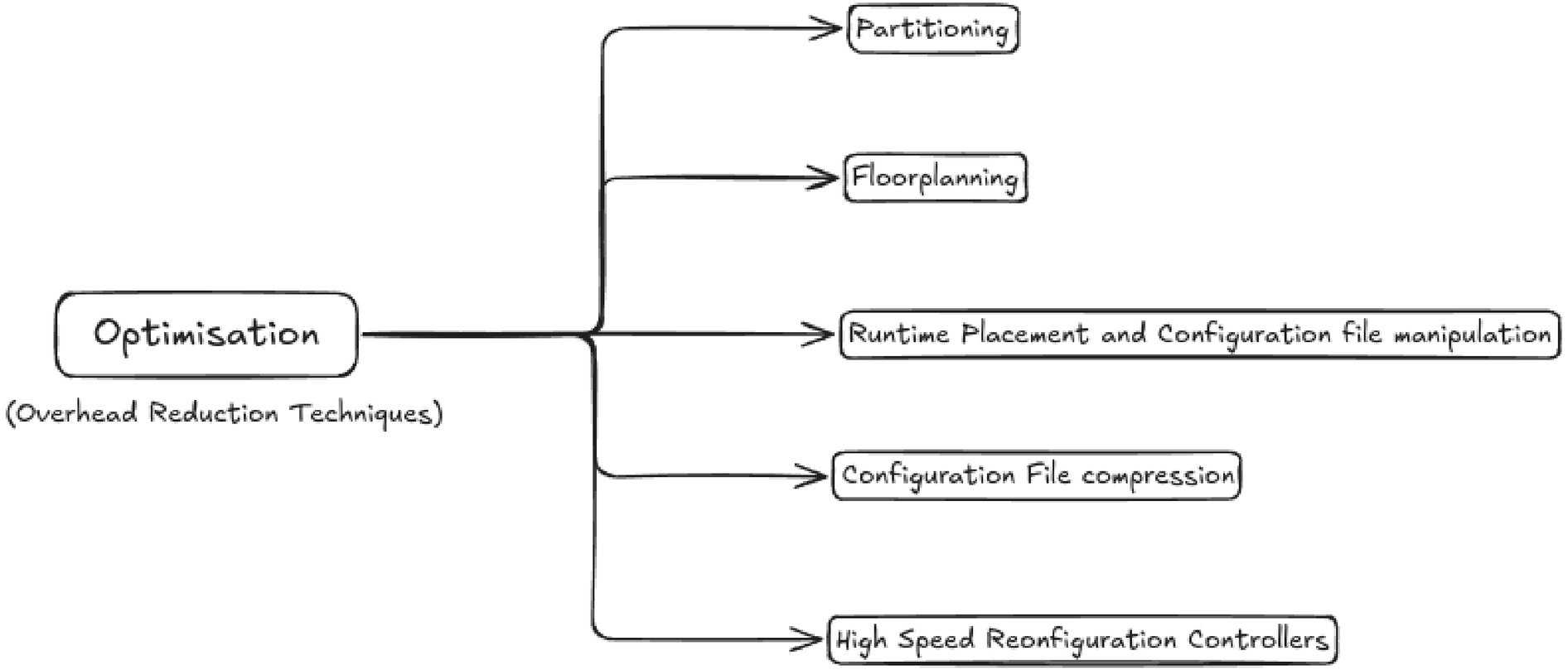
Tool	High-level Spec.	Partitioning	Floorplanning	Physical implementation	Circuit relocation	Run-time Mgmt.
Xilinx PlanAhead	○	○	◐	●	○	○
Xilinx Vivado	○	○	◐	●	◐	○
Altera Quartus	○	○	◐	●	○	○
Xilinx SDAccel	●	○	N/A	●	N/A	●
Altera OpenCL	●	○	N/A	●	N/A	●
OpenPR	○	○	○	◐	●	○
CoPR	◐	●	◐	○	○	◐
GoAhead	○	○	◐	◐	●	○
Caronte	◐	◐	○	○	◐	○
GePaRD	●	○	○	○	○	○
PaRAT	●	●	○	○	○	○
OSSS+R	●	◐	○	○	○	◐

Design, Implementation & Simulation Tool chain Technical Challenges

- **Bitstream Complexity:** Proprietary formats, undocumented frame addressing and CRC mechanisms
- **Routing Conflicts:** Static vs PR region routing, anchor logic differences (bus macros vs interconnect tiles)
- **Clock Distribution Issues:** Constraints across PR boundaries, routing instability
- **Reconfiguration Boundaries:** Signal instability, timing-sensitive behavior
- **Simulation Gaps:** No config memory modeling, limited transition support, abstraction limits bit-level accuracy

Design, Implementation & Simulation Tool chain
Toolchain Gaps & Future Directions

- **Toolchain Gaps:**
  - No unified HLS-to-PR implementation framework
  - Vendor-specific intermediate formats (.ncd, .xdl)
  - ISE-Vivado incompatibilities hinder tool continuity
  - Academic tools reliant on reverse engineering
- **Future Directions:**
  - HLS + PR integration (e.g., PaRAT's PRML)
  - Flexible, standardized PR interfaces beyond OpenCL
  - Accurate, portable simulation (e.g., ReSim with memory + error modeling)
  - PR-aware placement/routing algorithms with automation
  - Cloud-based PR design and floorplanning tools
  - Modular PR frameworks for embedded/autonomous systems


Optimisation
(Overhead Reduction Techniques)
Partitioning
Floorplanning
Runtime Placement and Configuration file manipulation
Configuration File compression
High Speed Reonfiguration Controllers

Overhead Reduction Techniques

Partitioning

- Mutually exclusive modules are often assigned to the same PRR to optimize resource usage.
- Increasing the number of modules within a PRR raises the area and reconfiguration time.
- A more efficient approach involves creating a dependency graph of tasks and scheduling them onto a limited number of PRRs.
- Designers choose the number and size of PRRs
- Assigning of hardware onto these PRRs are carried out via simulated annealing
- Early research assumed homogeneous FPGA resources, but post-2010 studies addressed the heterogeneous nature of modern FPGAs

Overhead Reduction Techniques

Floorplanning

- Vendor tools lack automatic floorplanning support.
- Manual floorplanning is time-consuming and often results in suboptimal designs.
- FPGA floorplanning is typically treated as a fixed-outline problem.

Research Approaches:

- Significant research has focused on developing floorplanning techniques for partial reconfiguration in heterogeneous FPGAs.
- Efforts have also been made to automate the detailed aspects of the floorplanning process.

Overhead Reduction Techniques

Runtime Placement and Configuration File Manipulation

- To reduce toolchain runtime, runtime relocation through placement, routing, and bitstream manipulation was introduced.
- However, limited processing power in embedded systems and the heterogeneous nature of modern FPGAs create significant bottlenecks.
- A key challenge is the lack of portability in online PnR tools, requiring modifications for different devices, even within the same FPGA family.

Research Approaches:

- Tools like PARBIT, REPLICA, and ReplicaPro provide solutions for bitstream manipulation and relocation on modern FPGAs.
- They also enhance placeability to support more efficient partial reconfiguration and relocation

Overhead Reduction Techniques

Configuration File Compression

- Configuration bitstreams often contain redundancies, both within and across different bitstreams.
- Compression can leverage this trait

Research:

- Improve reconfiguration throughput using vendor-supported compressed bitstream configuration without decompression.
- For example, Xilinx ICAP's Multiple Frame Write Register (MFWR) configures repeating frames to different memory locations.
- A reference PRR is stored uncompressed, while subsequent PRRs store only differences and their locations relative to the initial PRR.

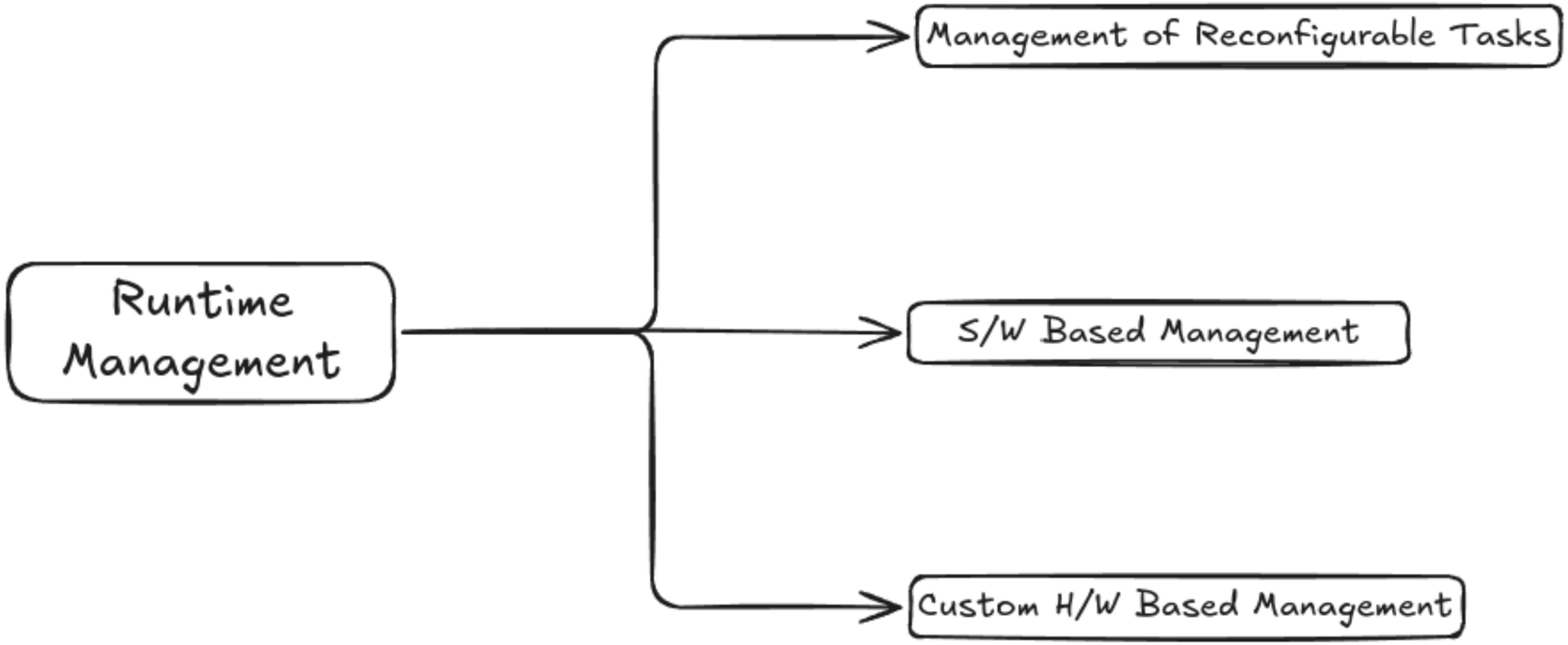
Overhead Reduction Techniques

High-Speed Reconfiguration Controllers

- The reconfiguration controller manages bitstream delivery to the ICAP macro.
- Vendor-provided controllers offer limited throughput (~10 MBps) compared to ICAP's potential (400 MBps).

Research Approaches:

- **ICAP-FSL Connection**: Improved throughput by linking ICAP to the Fast Simplex Link (FSL). However, the MicroBlaze processor remains burdened with bitstream fetching.
- **DMA Utilization**: Achieved significant throughput improvement by using Direct Memory Access (DMA) for bitstream transfer.
- **ICAP Overclocking**: Enhanced performance through device-specific overclocking of the ICAP primitive.
- **PCIe Direct Streaming**: Bypassed onboard memory by directly streaming bitstreams via PCIe, further increasing throughput


Runtime Management
Management of Reconfigurable Tasks
S/W Based Management
Custom H/W Based Management

Runtime Management of PR

Management of Reconfigurable Tasks

- Early research assumed FPGAs were composed of homogeneous units, allowing seamless creation of hardware tasks.
- This approach was demonstrated using the Xilinx XC6200 FPGA.
- However, it is unsuitable for modern FPGAs due to their fine-grained heterogeneous architecture
- Relocation is an important factor in implementing scheduled tasks
- One solution involves maintaining a database of partial bitstreams and all possible PRRs (Partially Reconfigurable Regions).
- Another approach standardizes PRRs in terms of dimensions and resources, enabling any bitstream to be configured to any PRR. An online scheduler can then manipulate bitstream addresses using algorithms like Best Fit or First Fit.

Runtime Management of PR

Software-Based Management

- Similar to OpenCL frameworks, this approach abstracts reconfiguration from the programmer, allowing configuration switching through API calls. However, the programmer must specify which bitstreams to load.

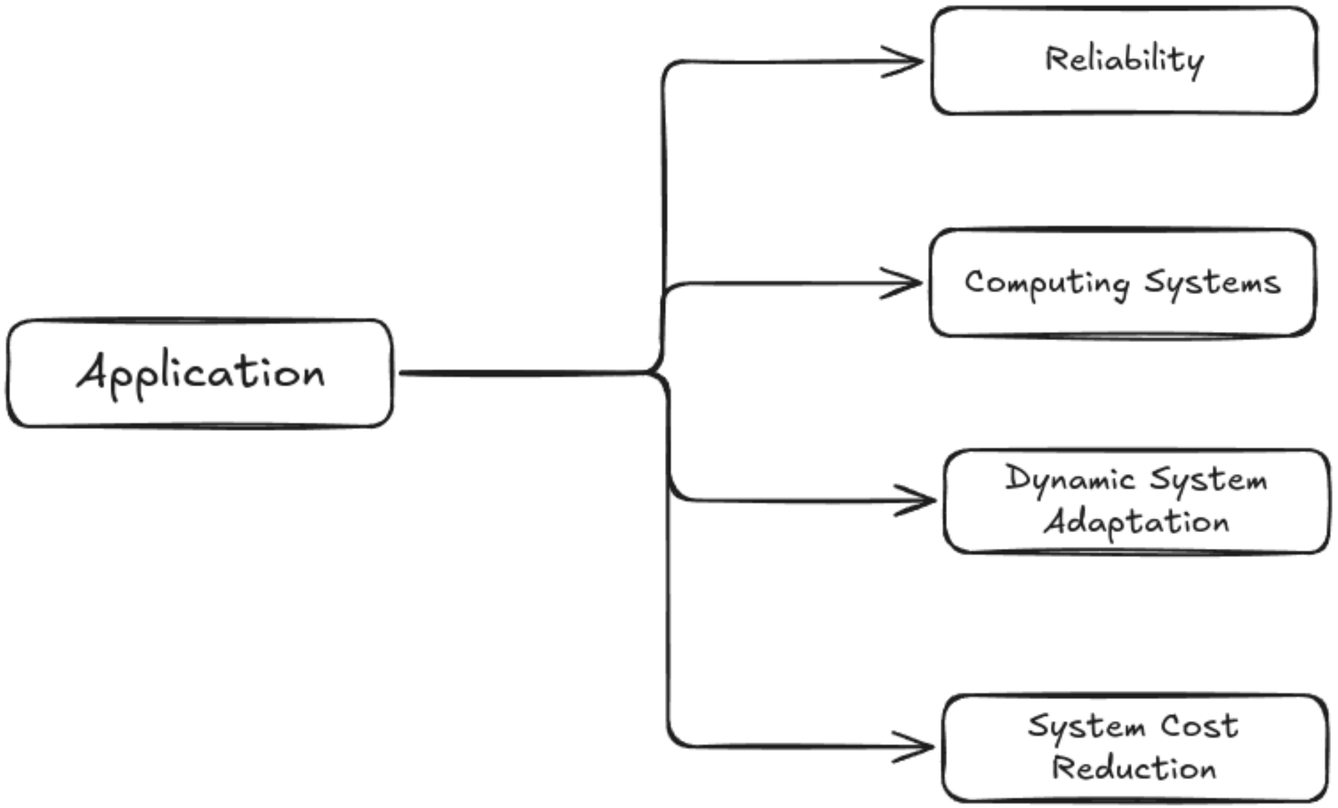
Research Developments:

- Runtime Reconfiguration with GNU/Linux: System calls were introduced to configure hardware modules on demand, offering a reasonable level of abstraction. However, this adds significant overhead due to multiple software layers.
- ReconOS: A unified operating system supporting partial reconfiguration, providing a standardized interface for custom accelerators. It operates using hardware threads (HARTs) and offers synchronization and communication utilities.

Runtime Management of PR

Custom Hardware-Based Management

- This approach offers high reconfiguration throughput using controllers with direct memory access (DMA).
- However, it sacrifices flexibility, as implementing complex algorithms at the hardware level is challenging.
- A balanced solution involves combining high-level software for abstraction with low-level hardware management blocks for efficiency.


Application
Reliability
Computing Systems
Dynamic System Adaptation
System Cost Reduction

Application

Reliability

Space

- SEU - Single Event Upsets are changes of state caused by ion or EM interference affecting a sensitive node in microelectronics
- This is a major problem in SRAM based FPGAs

- A way to mitigate this was partitioning the design and isolating possible fault sites and partially reconfiguring them from memory in case of detection of fault
- Fault was detected by using a reference model and comparing the output

- Another method was blind scrubbing, in which the partitions are reconfigured periodically (blind scrubbing)

- Alternatively detect the error by analysing the output and then apply reconfiguration

- In advanced methods, combining TMR (Triple module redundancy) and PR

Automotive electronics

- Redundant ECU

- Network security of automotive systems. Network controller is not loaded onto the node until h/w and s/w configuration checksums are not verified

Computing Systems

DISC (Dynamic Instruction Set Computer)

- The h/w modules for each instruction are paged in dynamically on demand

High Performance Reconfig Computing

- High performance reconfigurable computer, where the application is too large to fit in a single FPGA, use PR to run different computations at runtime

Accelerated Cloud Computing

- Applying PR in accelerating cloud compute
- FPGA acceleration in bing search

- Possible integration of dynamically accessed custom accelerators
- Virtualisation of single FPGA for multiple vFPGAs

- Survey of FPGA based accelerators for cloud computing

Self Modifying H/W

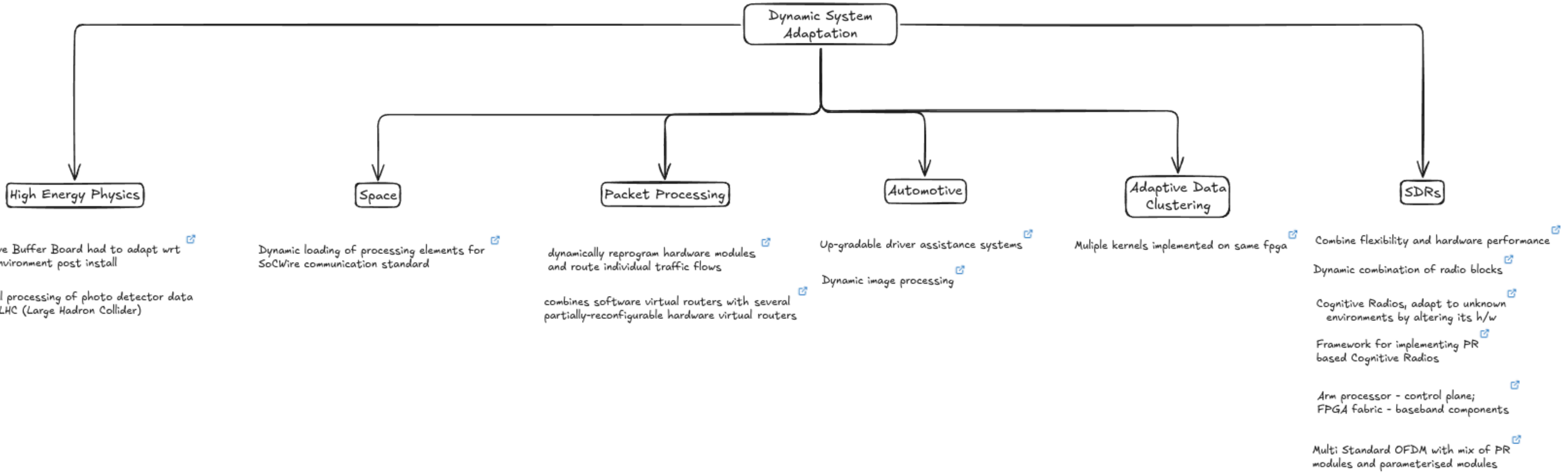
- FPGA creates new bitstreams, therefore can implement self modifying h/w

NN Implementations

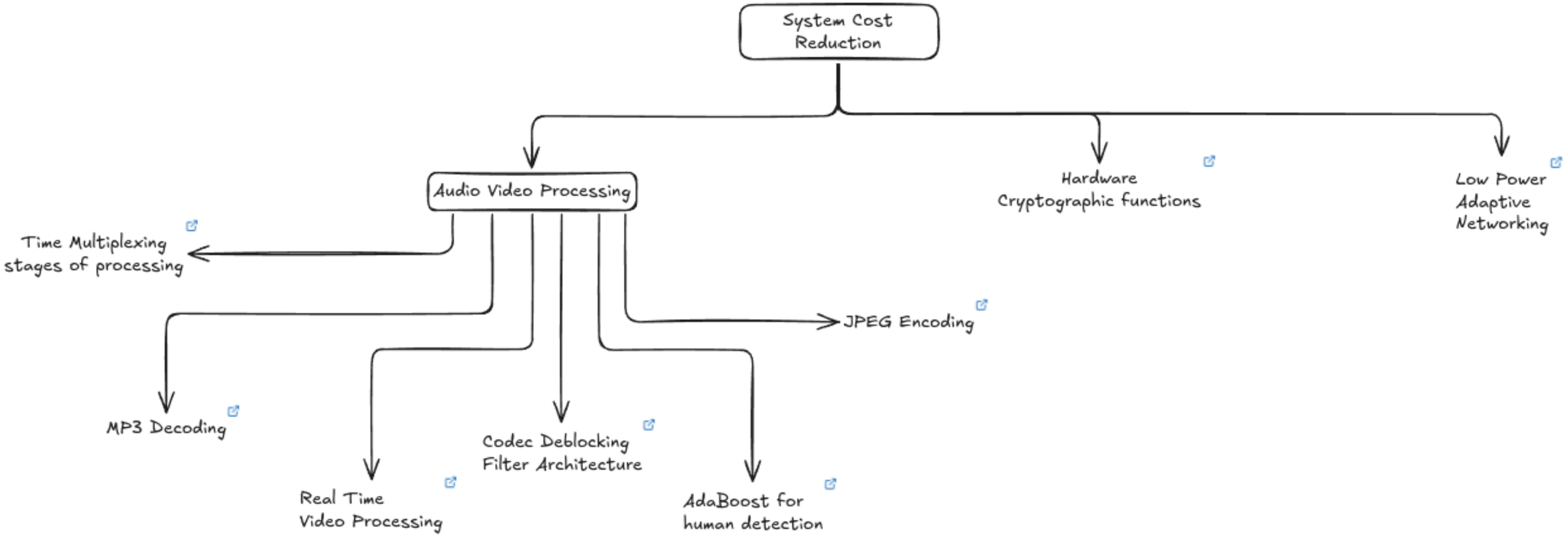
- High performance in Neural network computing, but needs to tailor inference to application specific tasks

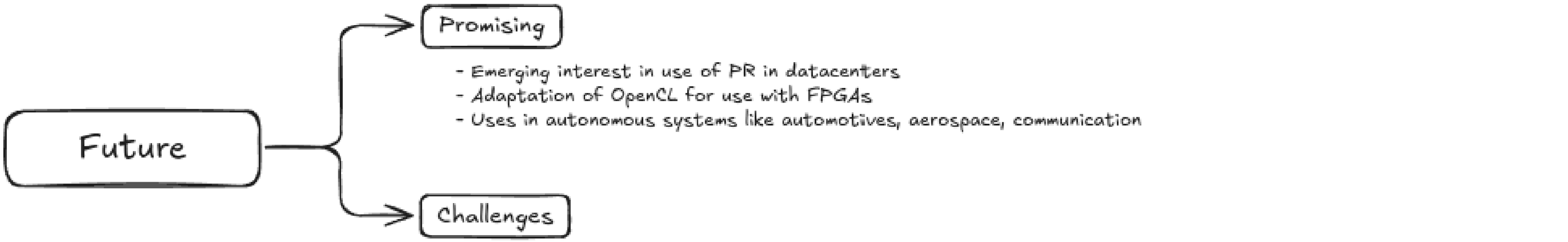
- Dynamically evolved classification module using PR

Application



Application




Future
Promising
- Emerging interest in use of PR in datacenters
- Adaptation of OpenCL for use with FPGAs
- Uses in autonomous systems like automotives, aerospace, communication
Challenges
- Many techniques are architecture (vendor) specific
- Better architectural support for multiple and dynamic usage of accelerators on commercial FPGAs
- Need to compile all the research done to date and choose optimal techniques to abstract away the hardware details
- Exploring autonomous/ self configurable intelligent PR applications
- In frameworks and applications, finding better application-oriented ways of describing adaptive systems that can be automatically mapped to PR implementations

Thank You