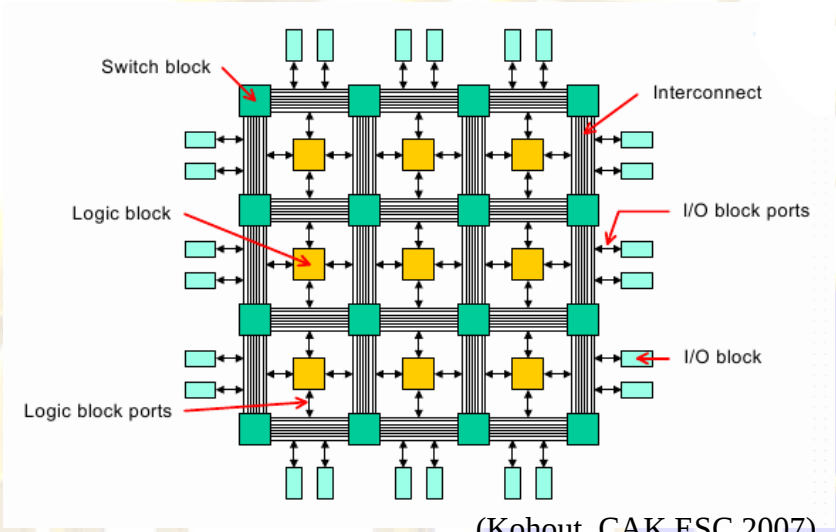
Operating System Support for Difference-Based Partial Hardware Reconfiguration

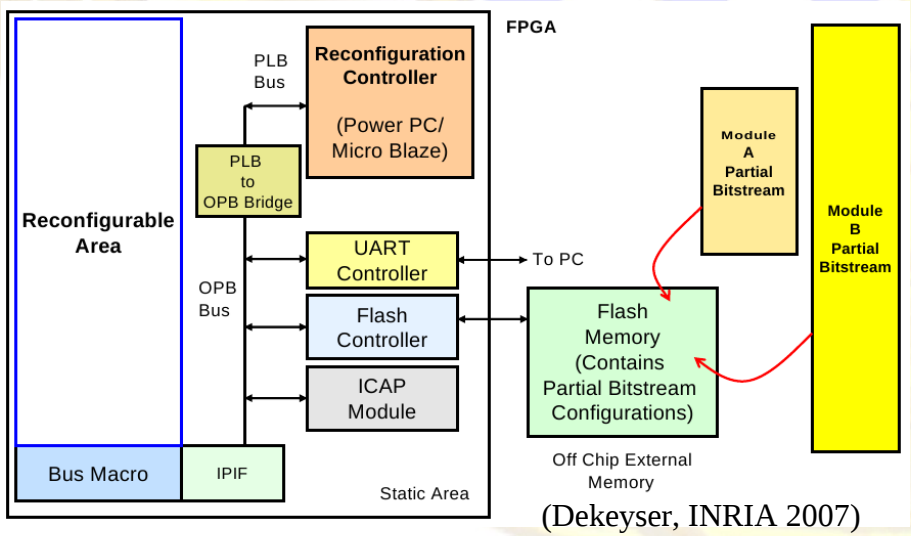
Tiago Reis and Antônio Augusto Fröhlich

Federal University of Santa Catarina
Software/Hardware Integration Lab

Motivation



(Kohout, CAK ESC 2007)



(Dekeyser, INRIA 2007)

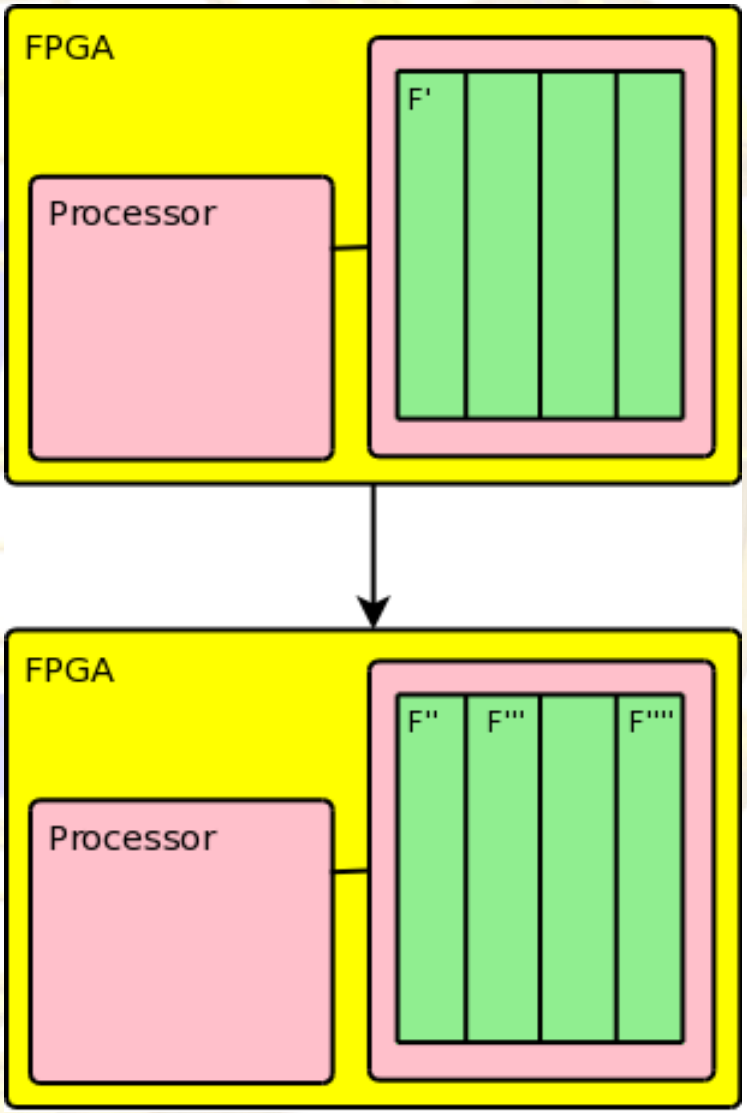
- Dynamic hardware reconfiguration
  - Around for over 10 years
  - Reasonable tool support
- Dynamic software reconfiguration
  - Around for over 25 years
  - Reasonable OS support
- What is missing for real dynamic system reconfiguration?
  - Integrate software and hardware
  - Relief developers from the burden

The HW/SW Reconfiguration Gap

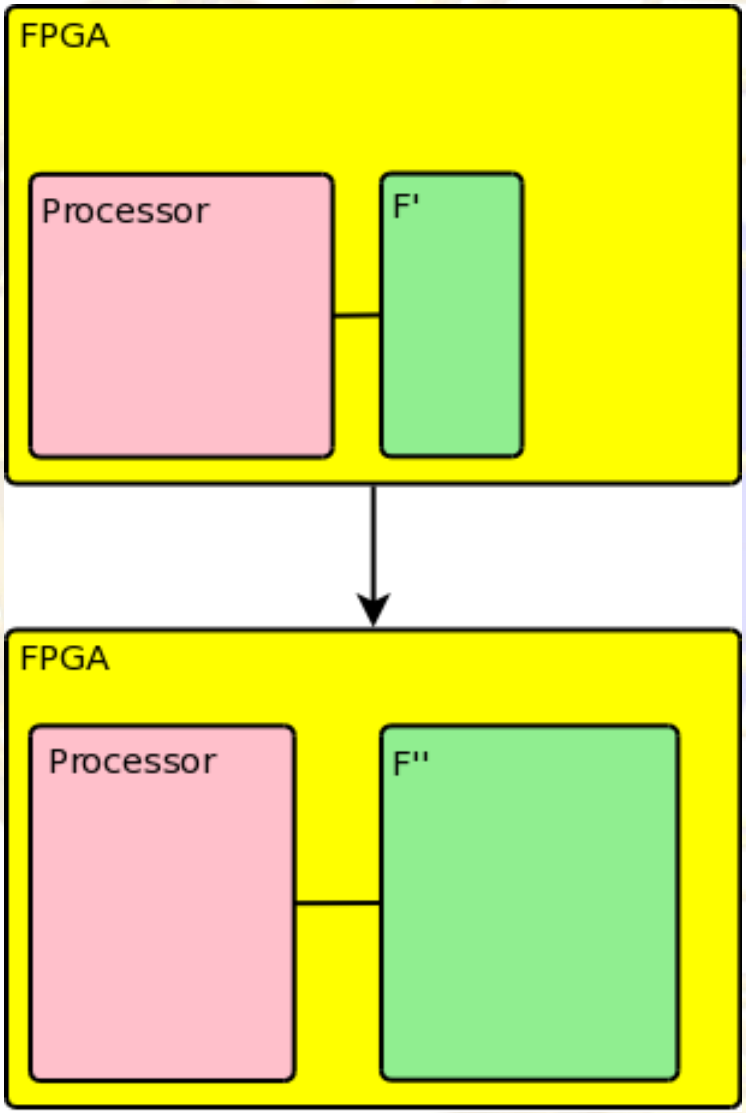
- One domain mostly ignores reconfiguration at the other, relegating it to application developers
  - No consistent handling of reconfiguration side-effects
    - Inconsistencies resolved by hand
  - High-overhead
    - Silicon area for the hardware
    - Processing power for the software
  - Low-portability
    - FPGA-dependent for the hardware
    - Architecture-dependent for the software

Partial Hardware Reconfiguration

- **Module-based**
  - **Predefined slots**
    - **Requires floor-planning**
    - **FPGA-dependent (bus macros)**
    - **Internal fragmentation**
  - **Strong tool dependency**
  - **Predefined interconnect**
  - **Easy to deploy**
    - **Slot replacement**



Partial Hardware Reconfiguration



- **Difference-based**
  - **No predefined slots**
    - **No need for floor-planning**
    - **FPGA-(possibly)-independent**
    - **No internal fragmentation**
  - **Partial bit-stream tool**
  - **Interconnect-unaware**
  - **Hard to deploy**
    - **Uncontrolled changes**

Partial Software Reconfiguration

- General-purpose OS
  - Large set of requirements
    - Complex run-time
    - High-overhead
  - Reconfiguration support
    - Dynamically Linked Libraries
    - Kernel modules
    - Reflective objects
  - Hardware reconfiguration pairing
    - Interconnect abstraction through communication channels
    - Preserved CPU status
    - OS transparent
- Embedded OS
  - Application-specific requirements
    - Tailored run-time
    - Low-overhead
    - Real-time
    - Energy-aware
  - Reconfiguration support
    - Virtually none
    - Remote software update
  - Hardware reconfiguration pairing
    - Mostly by hand

Closing the HW/SW Reconfiguration Gap

- From the hardware side
  - Partial bit-stream loading support
- From the OS side
  - Software update support
  - Hibernation support (power management)
- In between
  - Well defined HW/SW interface through hardware mediators
- Strategy

1 - Delete mediators of hardware components that will be reconfigured

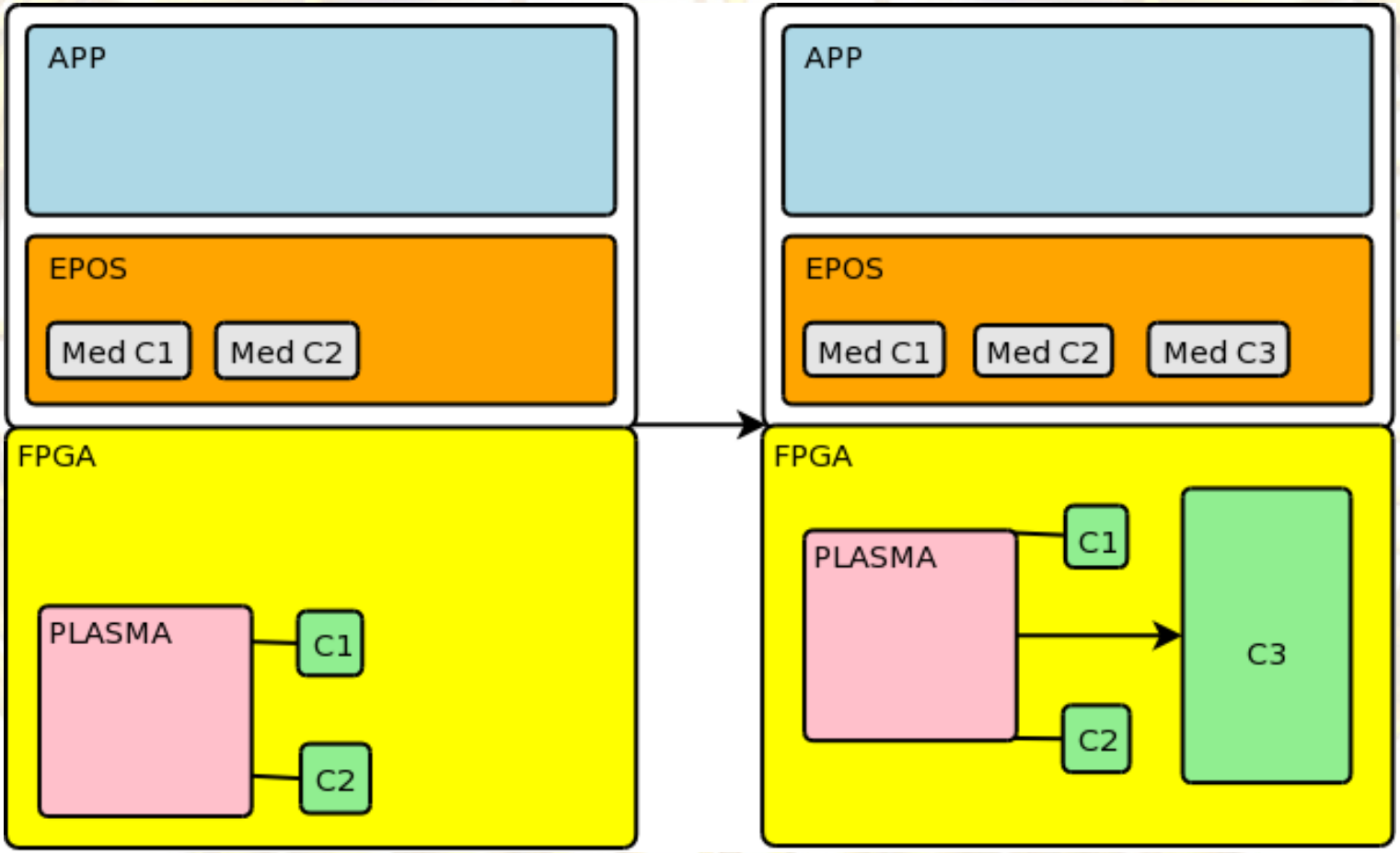
2 - Hibernate the system (quintessential state)

3 - Load partial bit-stream

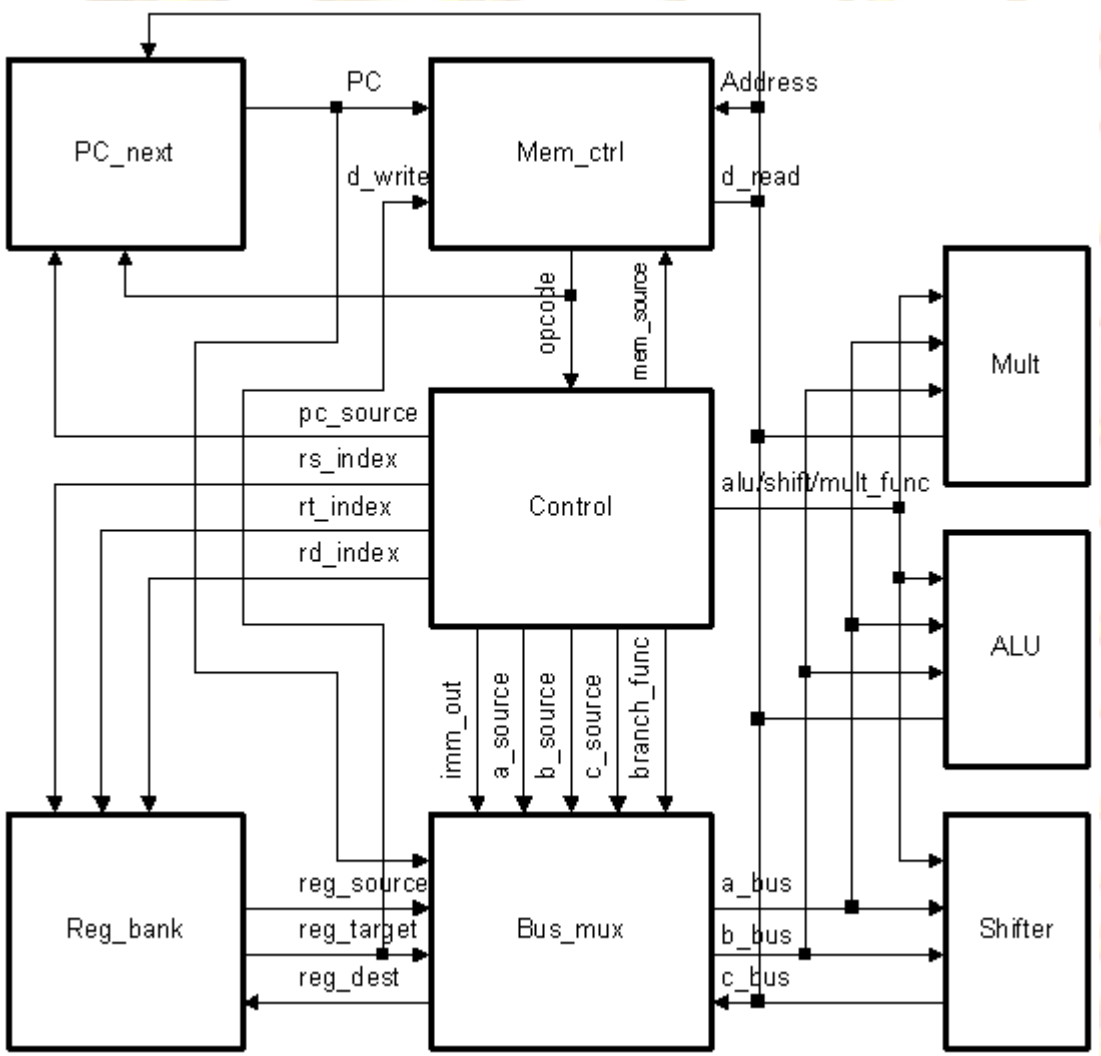
4 - Resume the system

5 - Create mediator for hardware components that have been reconfigured

Proof of Concept

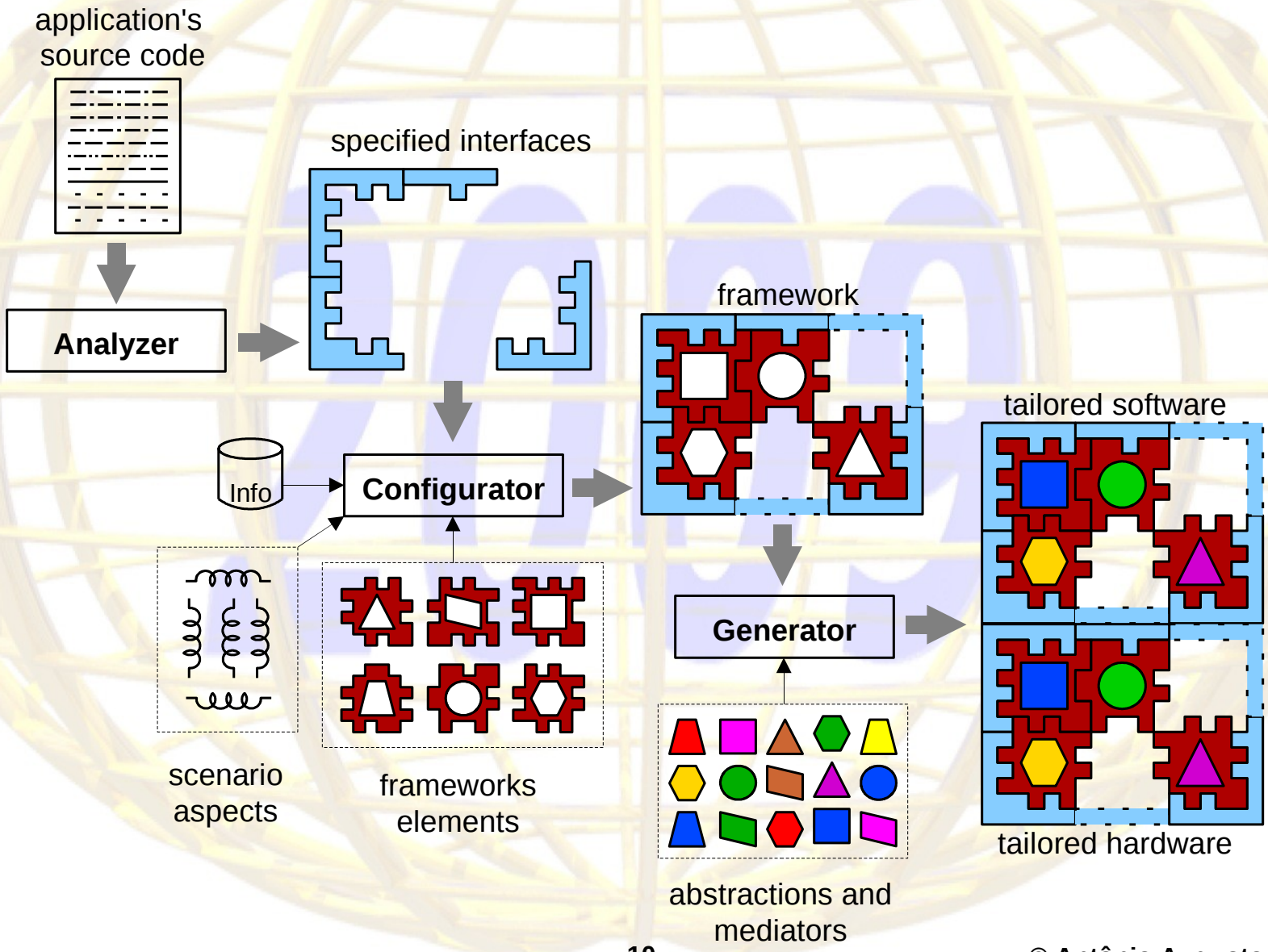


Plasma

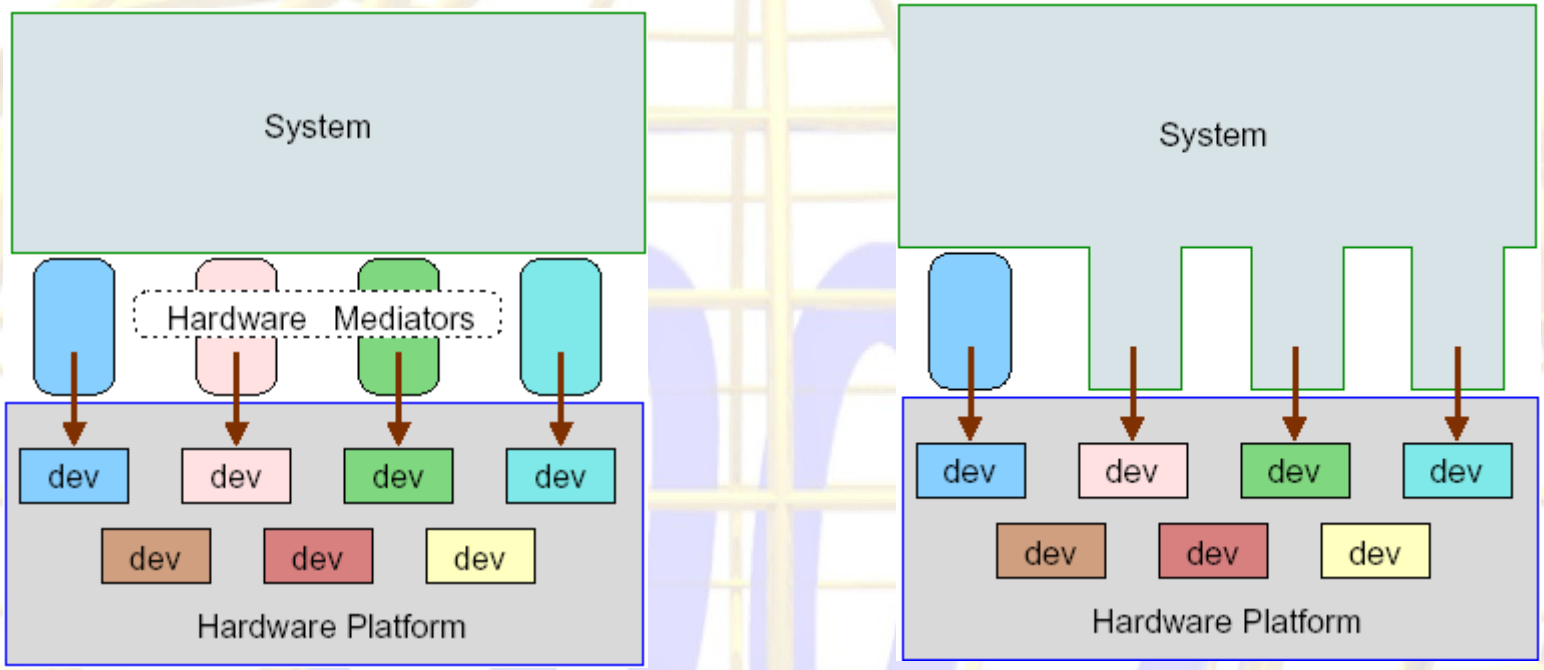


- Small synthesizable 32-bit RISC microprocessor
- MIPS I compliant
- OpenCores

EPOS



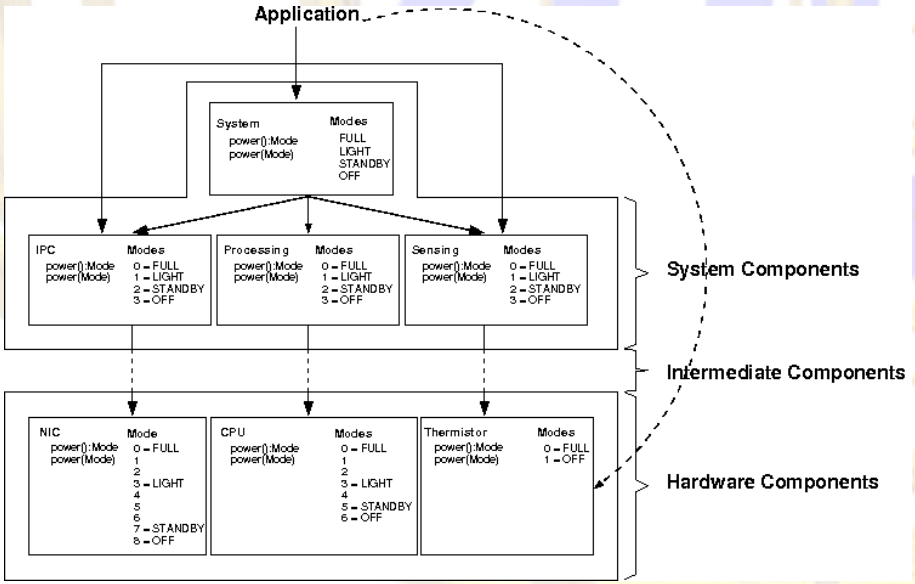
EPOS Hardware Mediators

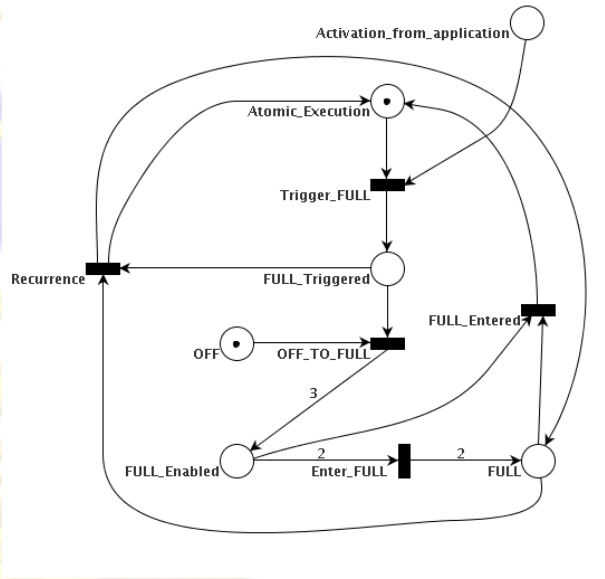


- **Sustain an interface contract between software and hardware components**
- **Mostly meta-programmed**
  - **No unnecessary code like in ordinary HALs**
  - **As soon as the interface contract is met, mediators "dissolve" themselves inside components**

Power Management in EPOS

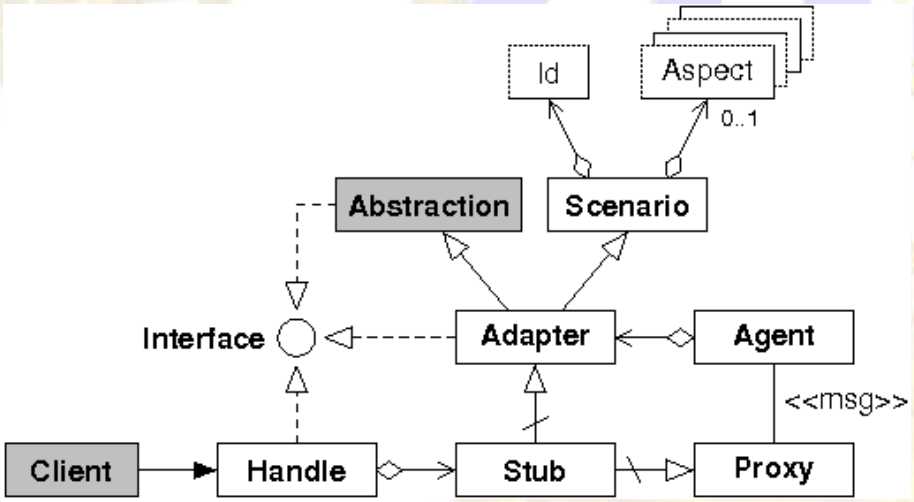
- Application-driven
- Hierarchical
  - At high-level abstraction propagated to mediators
- Semantic modes
  - Extended to include hibernation
- Drives the system into a quintessential state

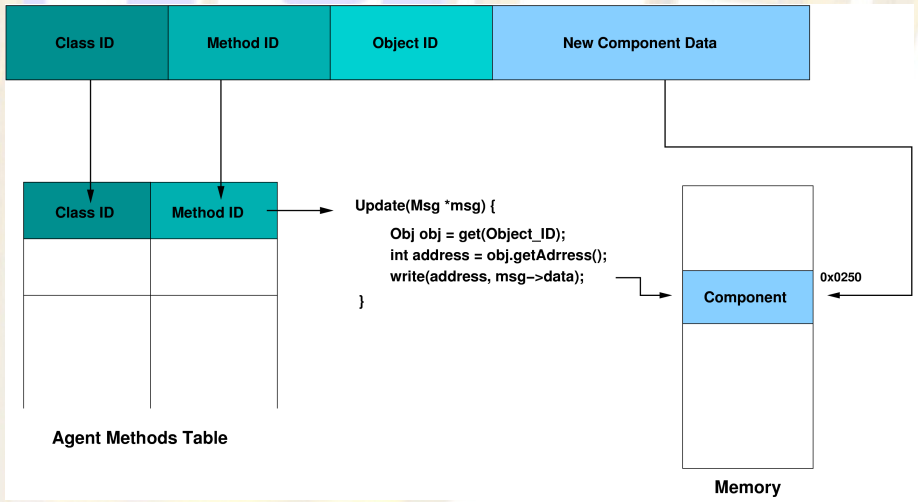




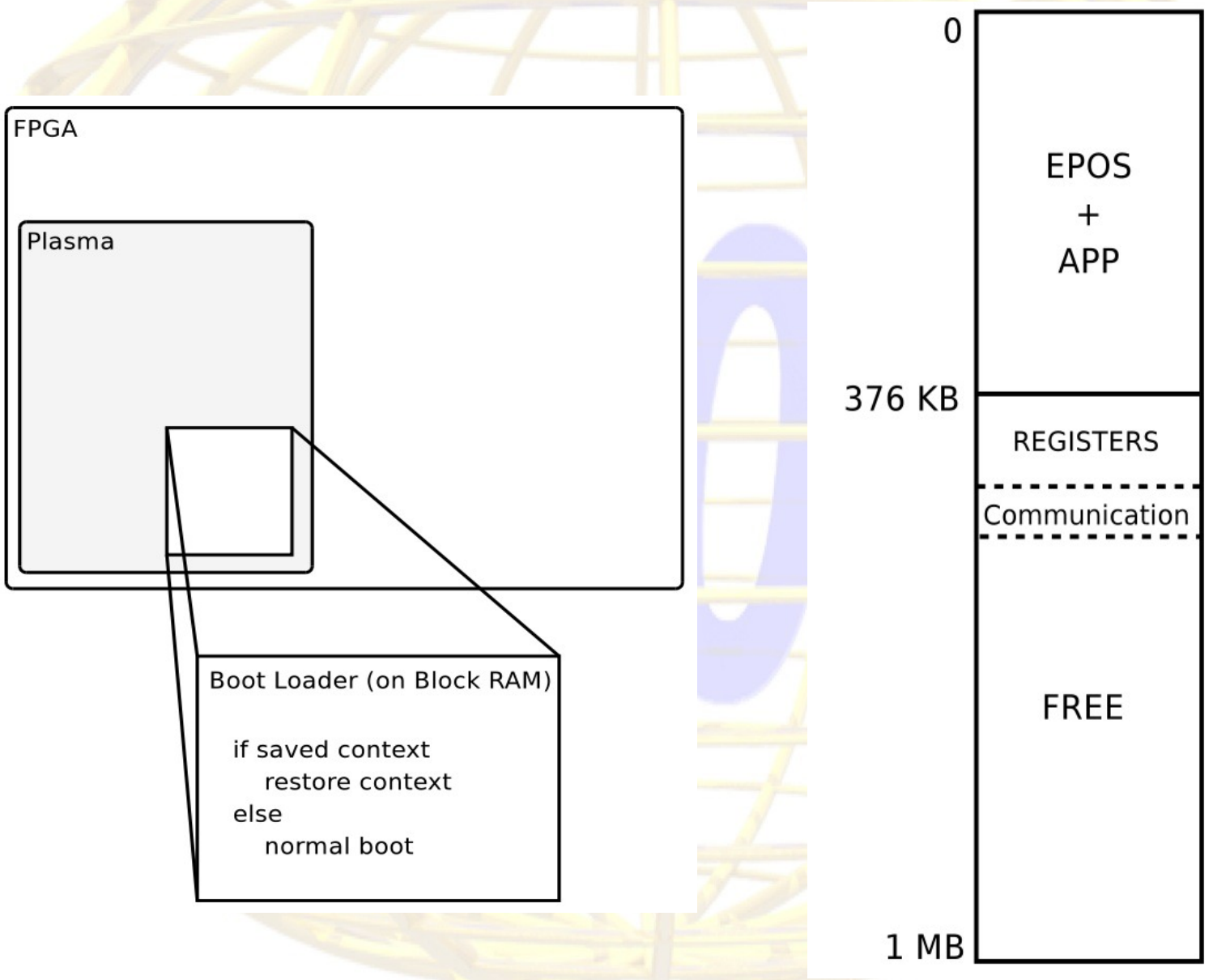
Software Update in EPOS

- **Based on the remote aspect**
  - Decouples client and component implementation
- **Component code mapping**
  - Buildup maps at compilation
- **Dissemination mechanism**
  - Essential for WSN





Partial Reconfiguration in EPOS



Results and Evaluation

- **EPOS and boot loader sizes (in bytes)**

	Original	Reconf.	Increase (%)
EPOS	18,768	19,048	280 (1.49 %)
Boot loader	4,468	4,808	340 (7.6 %)

- **Save and restore execution time (in µs)**

	CPU registers	RAM copying
Save	3.36	32,203.40
Restore	26,955.00	55,304.00
Total	26,958.36	87,507.40

Conclusion

- Hardware and software infrastructure have been designed mostly independently
  - Distinct goals
  - Distinct strategies
- Relegating part of system reconfiguration to the application is not an viable alternative
  - Has been done mostly due to the gap between HW and SW reconfiguration infrastructures
- Modern embedded operating systems can offer a concrete alternative
  - Well defined HW/SW interface through hardware mediators
  - Component update support
  - Power management support with hibernation
- EPOS has demonstrated the approach's viability