On the Automatic Generation of SoC-based Embedded Systems

Fauze Valério Polpeta
Antônio Augusto Fröhlich

Laboratory for Software/Hardware Integration
Federal University of Santa Catarina

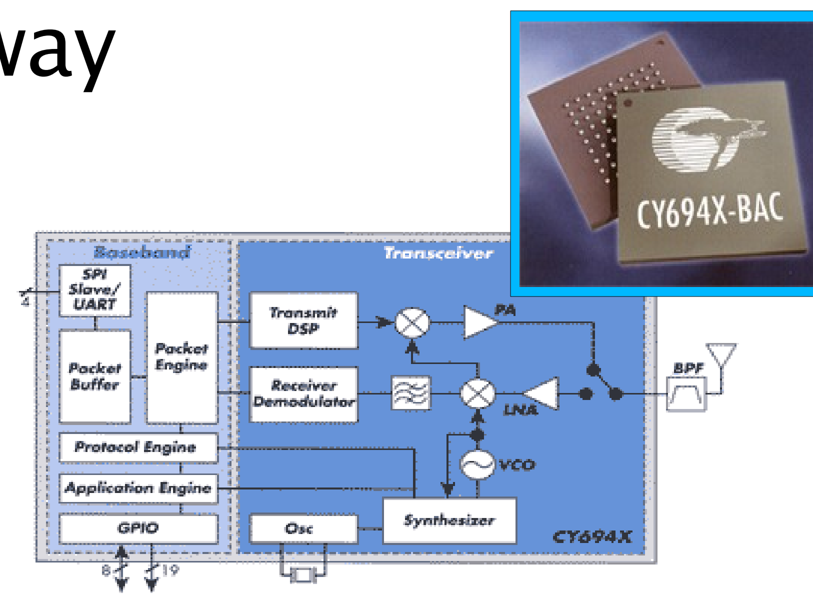
guto@lisha.ufsc.br
http://www.lisha.ufsc.br/

September 21, 2005

System-on-Chip

- Advances on programmable logic devices are enabling developers to integrate complex hardware designs in a single silicon pastille
- Soft-core processors can sustain such designs in a flexible way

Embedded systems implemented as SoCs



Operating System

- The more complex the SoC, the greater the probability it will need some sort of run-time support system
  - Operating system
    - Abstract underlying hardware
    - Sustain a programming model for applications
  - Applications
    - High-level programming languages
    - Reusable software artifacts
- Ordinary operating systems cannot go with the dynamism of SoCs

  "the current specification techniques for sw-hw interfacing are so far from the ideal plug-and-play" [Neville 2003]

SoC x OS: the Traditional View

- Most currently available co-design tools and methodologies pay little attention to run-time support systems
  - Platform-based design

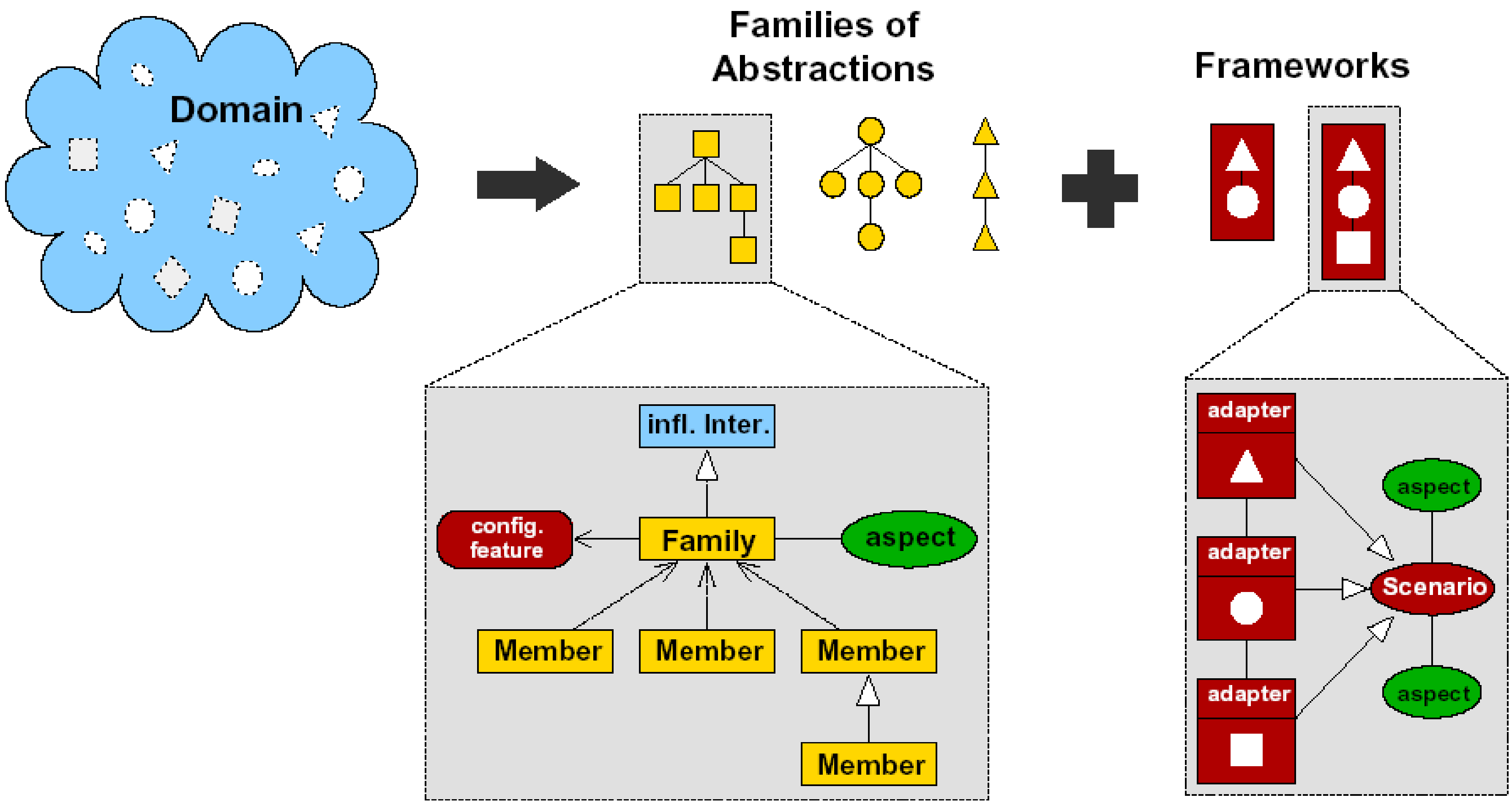
    "In essence, a platform is a frozen architecture. Once the architecture is frozen, you may standardize the interfaces and give the engineers some choice of building blocks"

    [Smith 2004]
  - Traditional EDA tools are quite restrictive as regards the development of run-time support systems
  - Run-time support is usually regarded as part of application's duties

SoC: the AOSD View

- AOSD was born as an component-based system software development methodology
  - Domain engineering methodology
- Soft IPs are much like software components
- Therefore AOSD should be able to guide the component-based design of SoCs as well
  - Hardware IPs build up the machine
  - Software IPs build up the run-time support system
  - Application requirements drive the process
  - Developers can thus concentrate on what really matters: applications

Application Oriented System Design

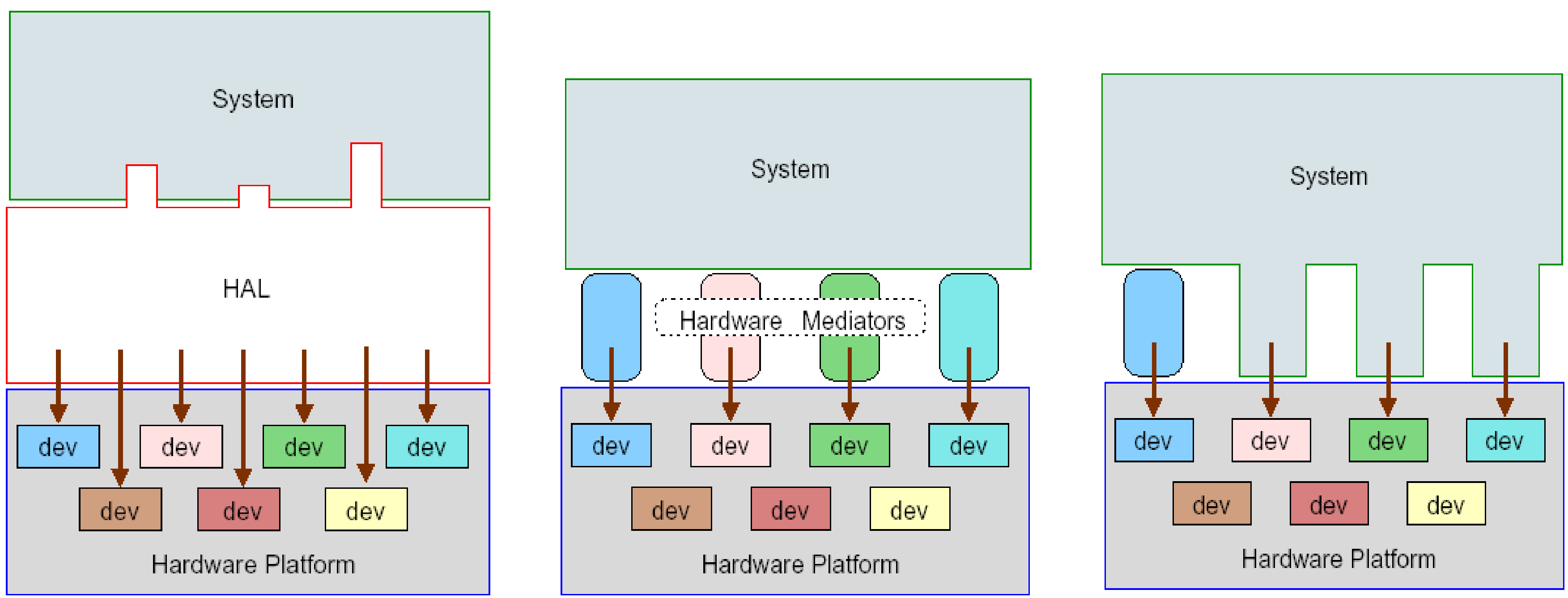


The EPOS System

- Embedded Parallel Operating System
  - A collection of software components designed according to AOSD principles
  - A meta-programmed framework
  - A set of tools to assist the selection, configuration and adaptation of those software components
- Portability
  - EPOS abstractions (user-visible software components) interact with hardware components through mediators
  - Hardware mediators sustain an interface contract between system abstractions and the machine

AOSD Hardware Mediators

- Mediators are mostly meta-programmed
  - No unnecessary code like in ordinary HALs
  - As soon as the interface contract is met, mediators "dissolve" themselves inside abstractions



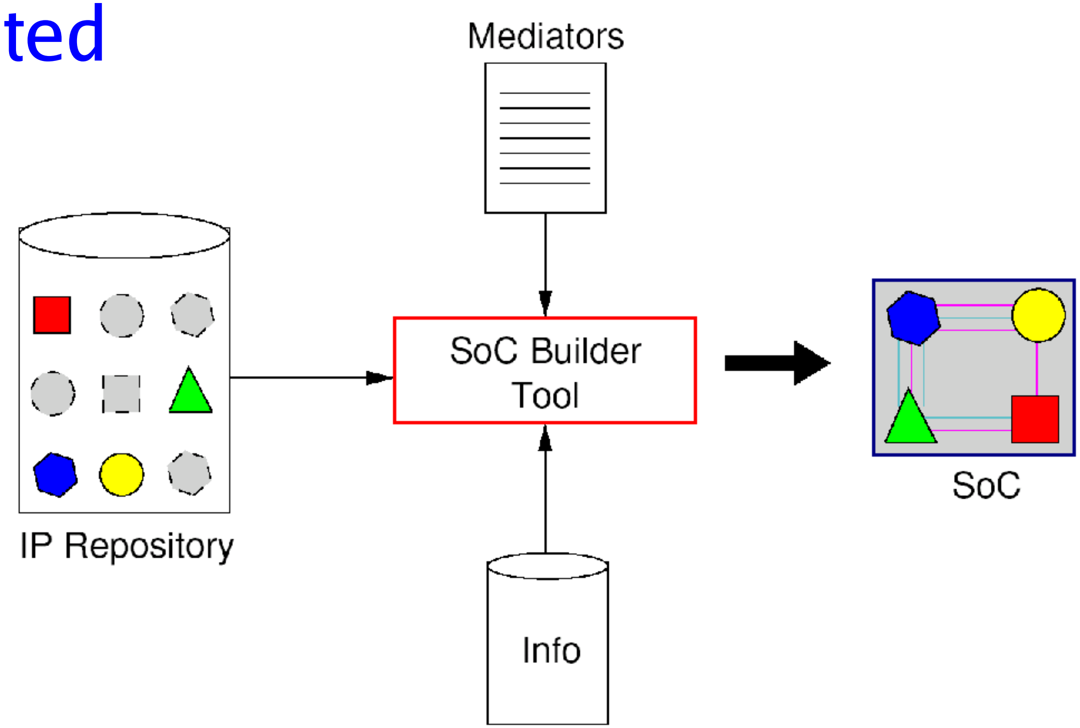
Federal University of Santa Catarina
Laboratory for Software/Hardware Integration

AOSD Hardware Mediators and Software/Hardware Co-Design

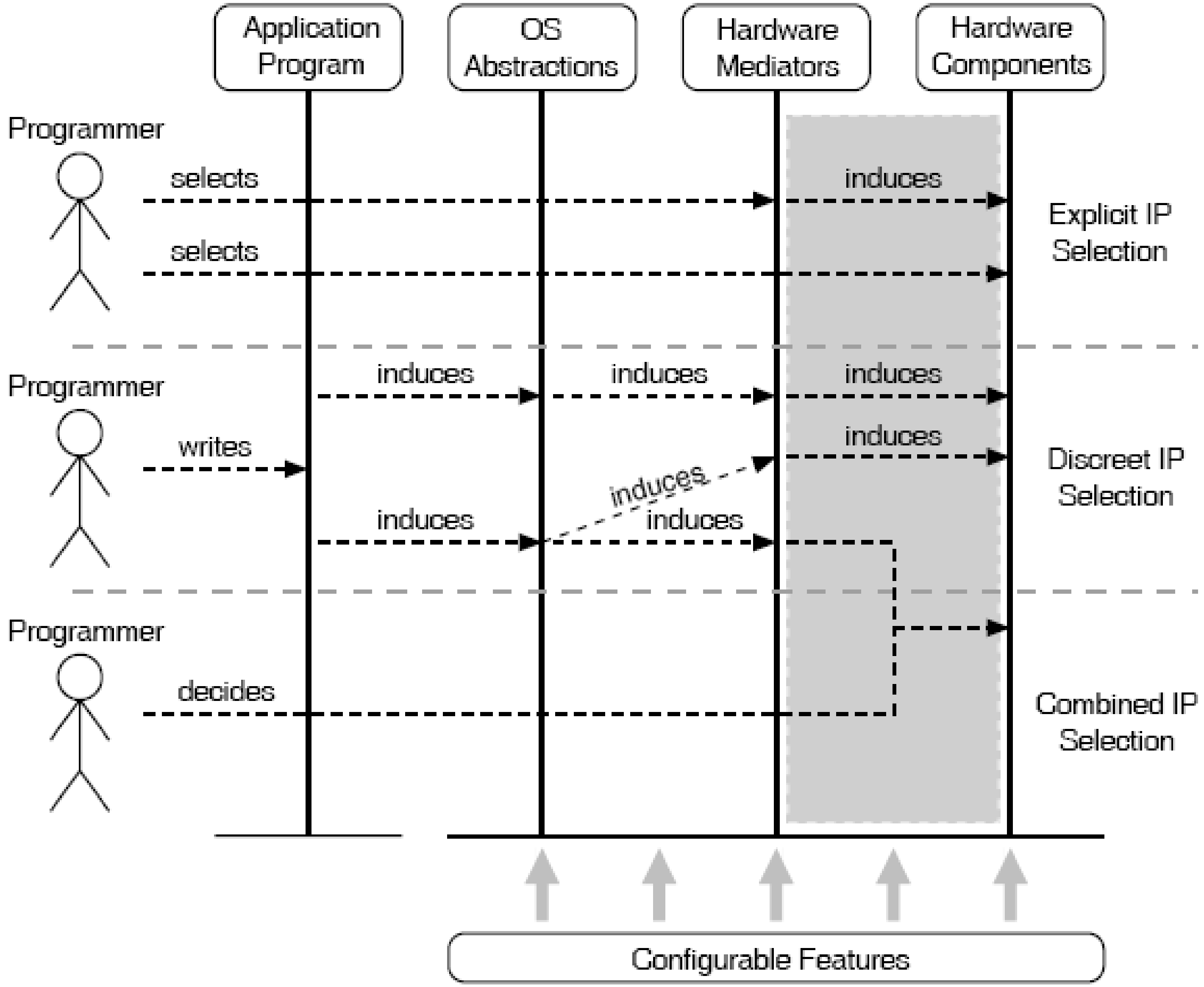
- The system/hardware interface resulting from the instantiation of hardware mediators can also be seen as a concrete specification of application requirements as regards the supporting hardware
  - Each selected mediator designates a hardware component
  - Parameters and the invocation scenario of each mediator can be used to infer hardware features

Hardware Components in EPOS

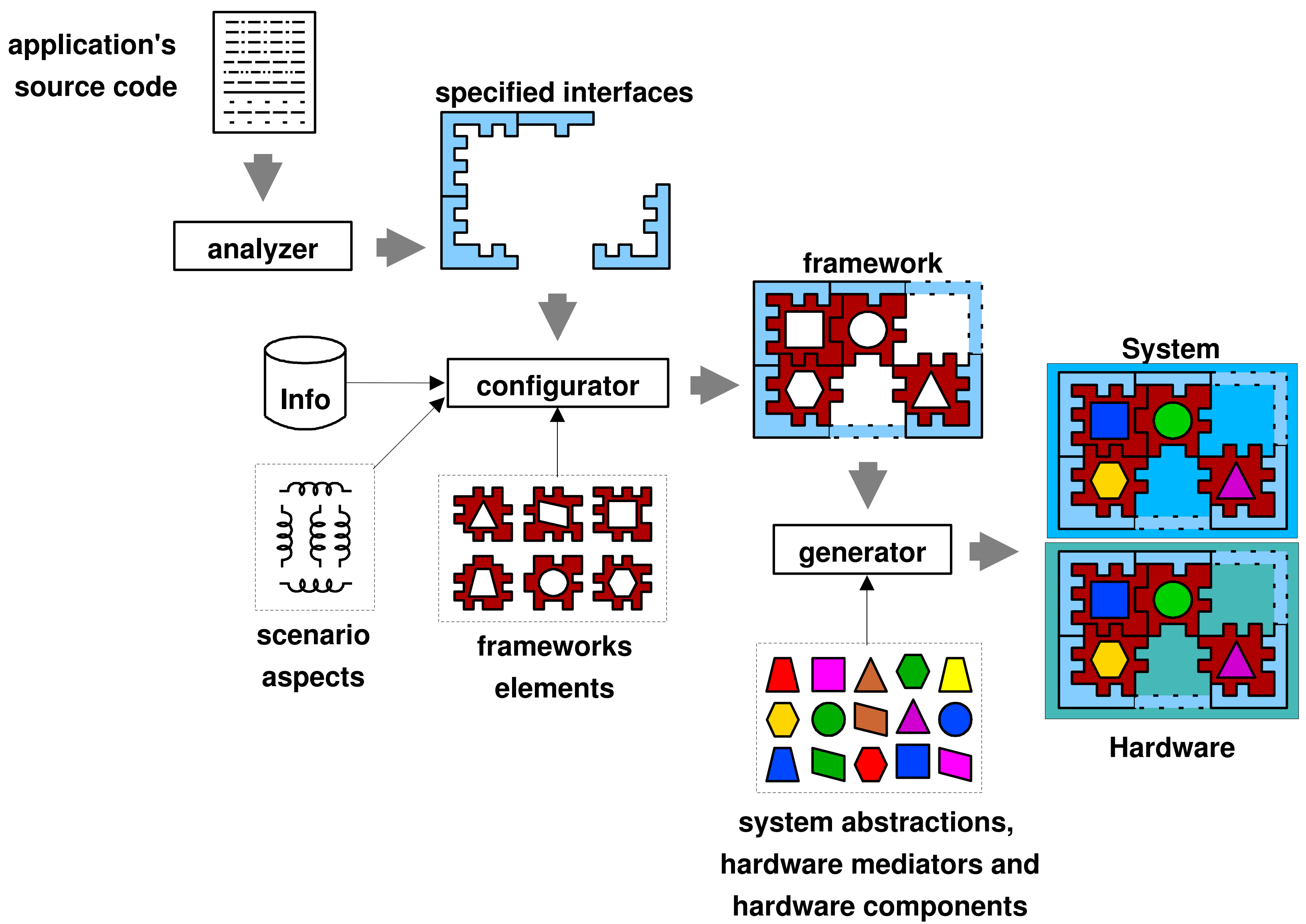
- Hardware mediators can be associated with hardware components (IPs)
  - As soon as a mediator is defined as result of an application requirement, the associated IP is selected



IP Selection Scenarios



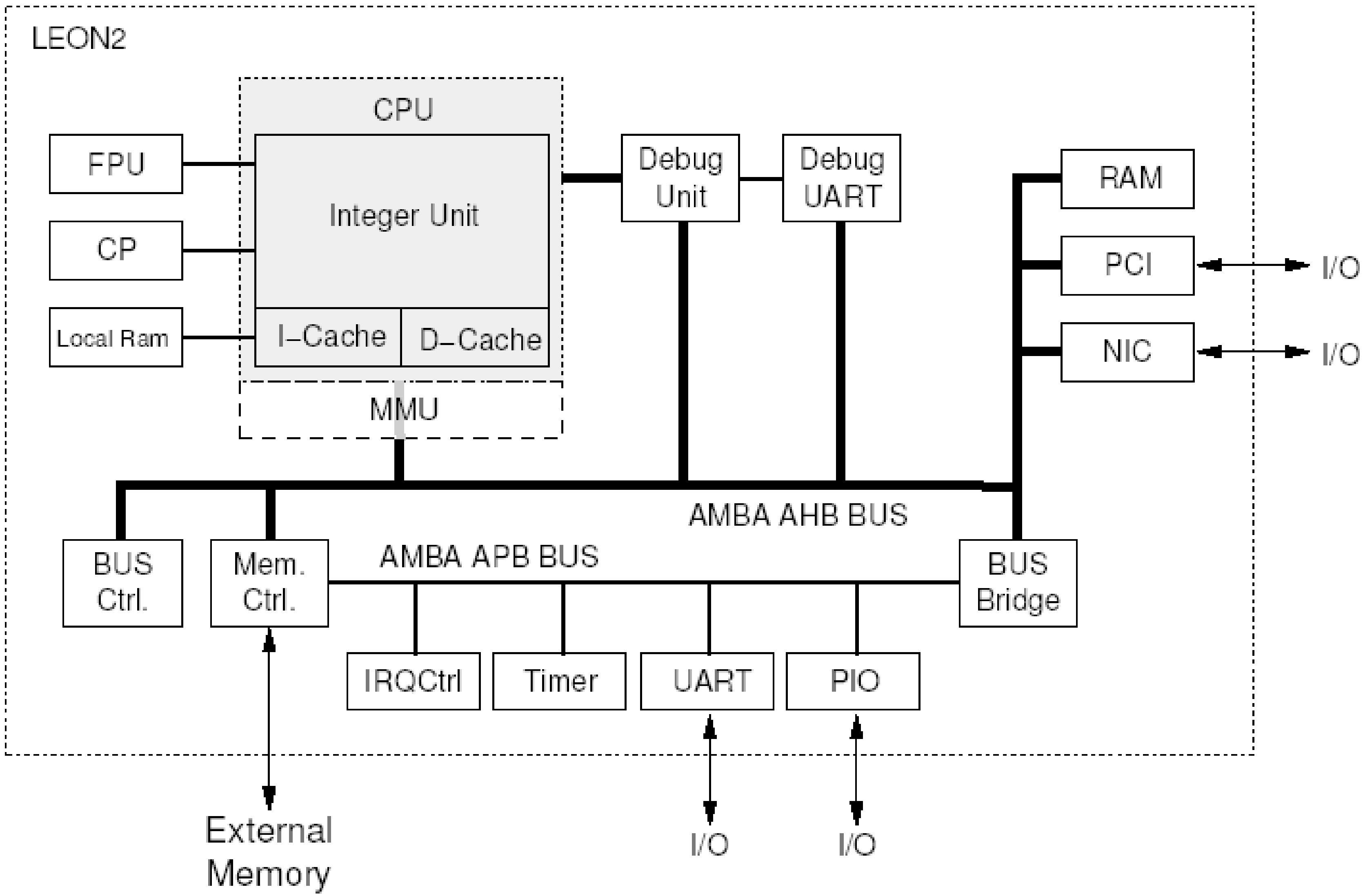
EPOS Instantiation Process



Case Study

- Application
  - Serial relay (bounded buffer)
- SoC
  - Automatically generated by EPOS tools
  - Soft-core decomposed in EPOS components
    - Modular design
    - Standardized BUS
    - Implicit glue-logic
    - GNU GCC (G++)
  - OpenRISC
    - Poor modularization (unnecessary dependencies)
  - Leon2
    - OK

Leon2


LEON2
CPU
FPU
CP
Local Ram
Integer Unit
I–Cache
D–Cache
MMU
Debug Unit
Debug UART
RAM
PCI
NIC
I/O
I/O
AMBA AHB BUS
AMBA APB BUS
BUS Ctrl.
Mem. Ctrl.
BUS Bridge
IRQCtrl
Timer
UART
PIO
External Memory
I/O
I/O

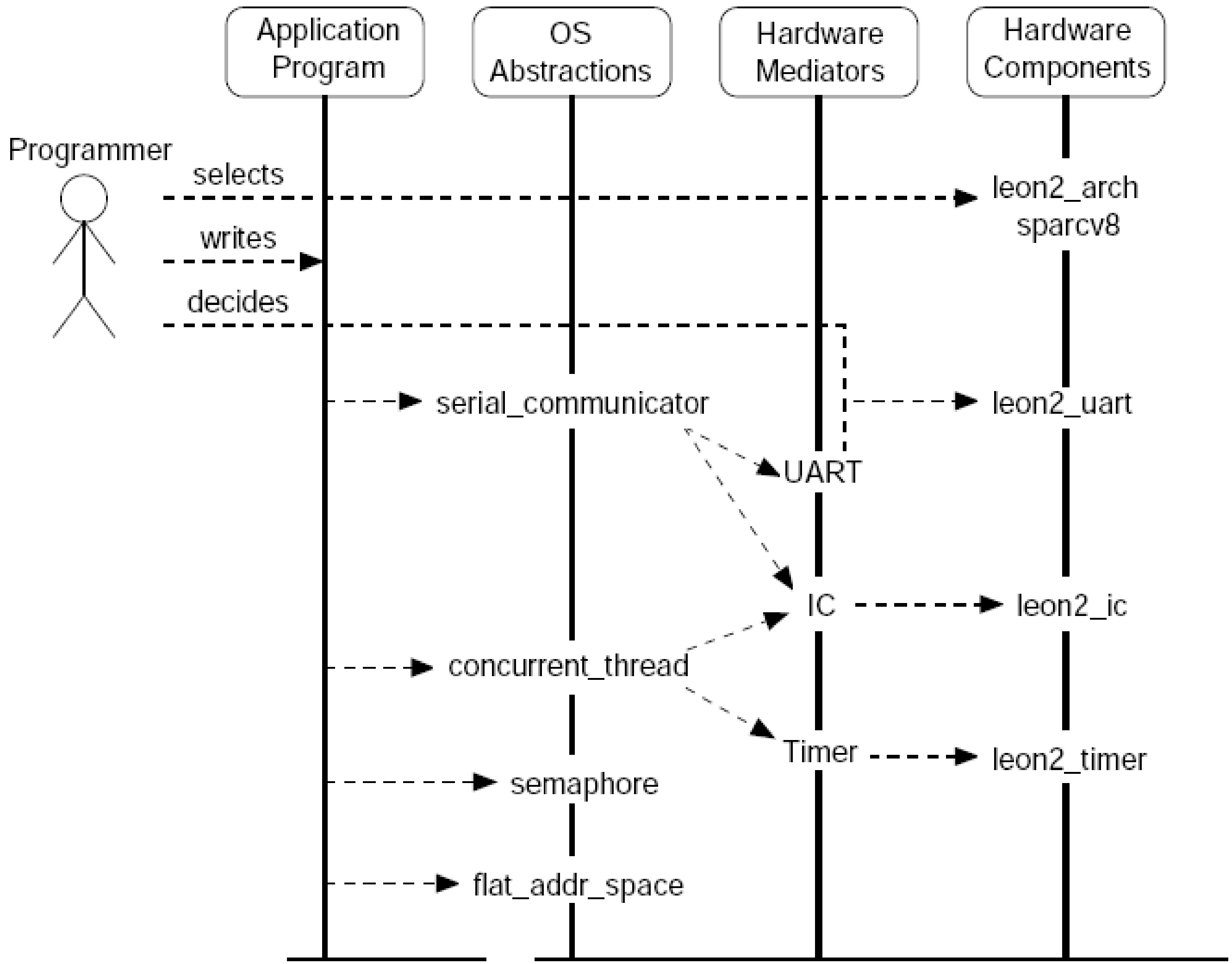
Federal University of Santa Catarina
Laboratory for Software/Hardware Integration

Application

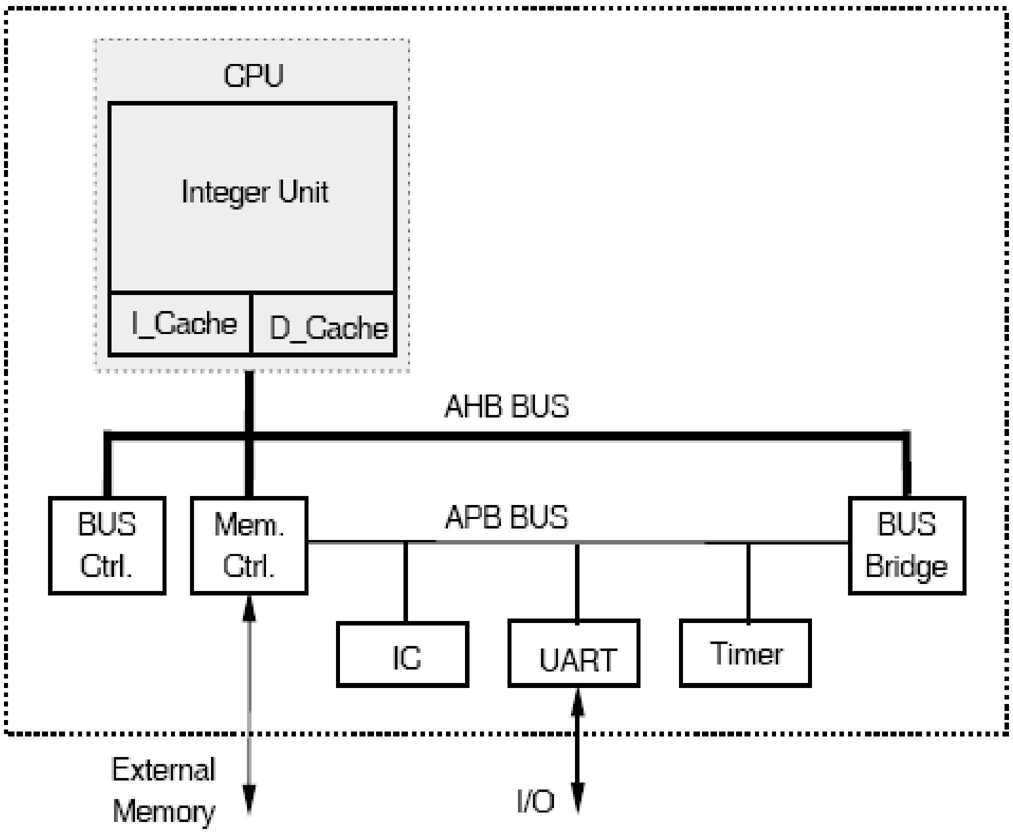
```
char * buf;
Thread * cons;
Semaphore empty(LEN), full(0);
Serial_Communicator comm;

int main() {
  buf = new char[LEN];
  cons = new Thread(&consumer);

  // producer
  int in = 0;
  while(true) {
    empty.p();
    comm->receive(&buf[in], 1);
    // operate on buf[in]
    in = (in + 1) % LEN;
    full.v();
  }
}
```

```
int consumer() {
  int out = 0;
  while(true) {
    full.p();
    // operate on buf[out]
    comm->send(&buf[out], 1);
    out = (out + 1) % LEN;
    empty.v();
  }
}
```

Component Inference Flow



Tailored Leon2



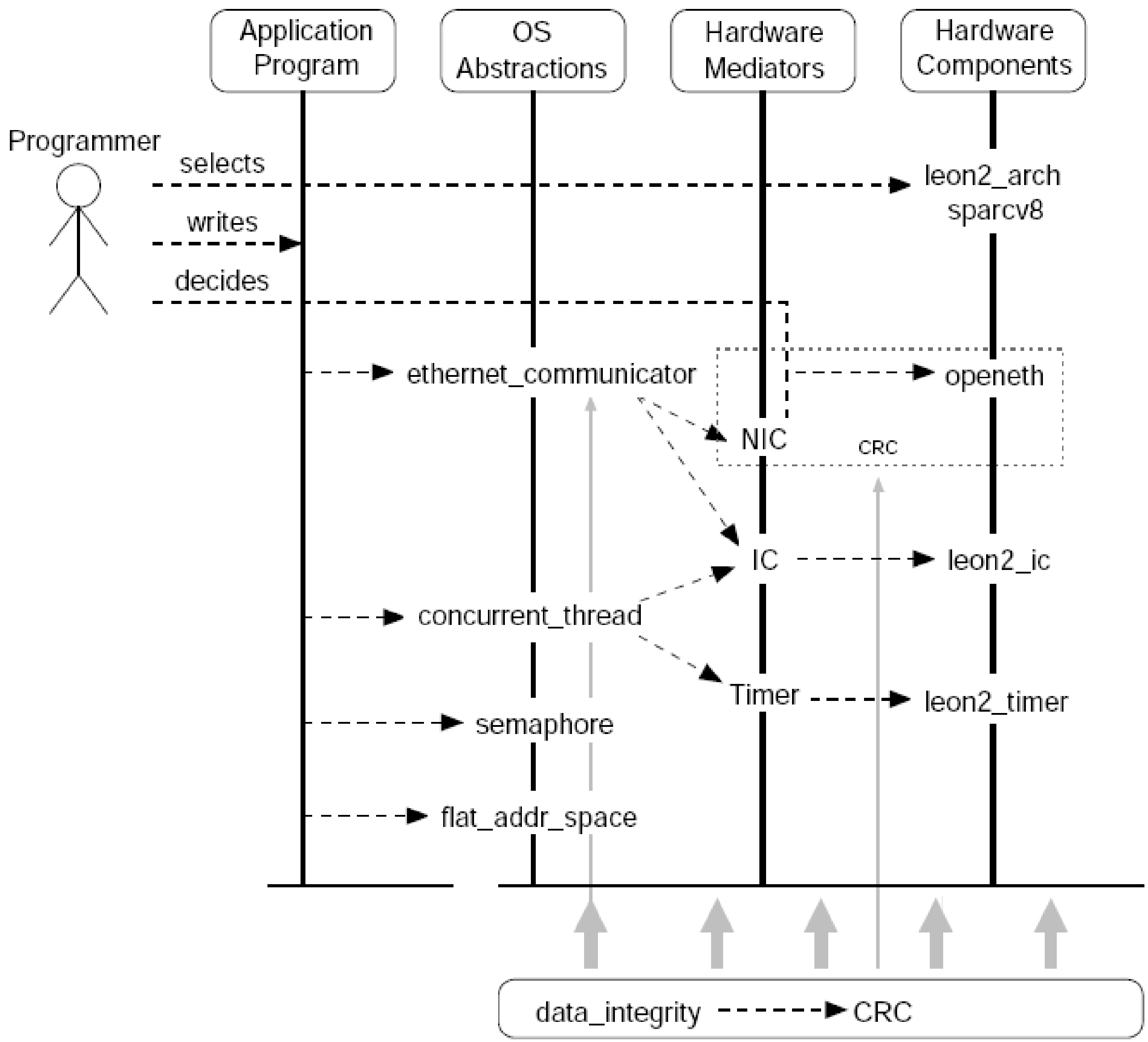
Some EPOS SoC Figures

SoC	Size (LUTs)	Virtex2 Area
Full	14,582	67%
Tailored	6,792	31%

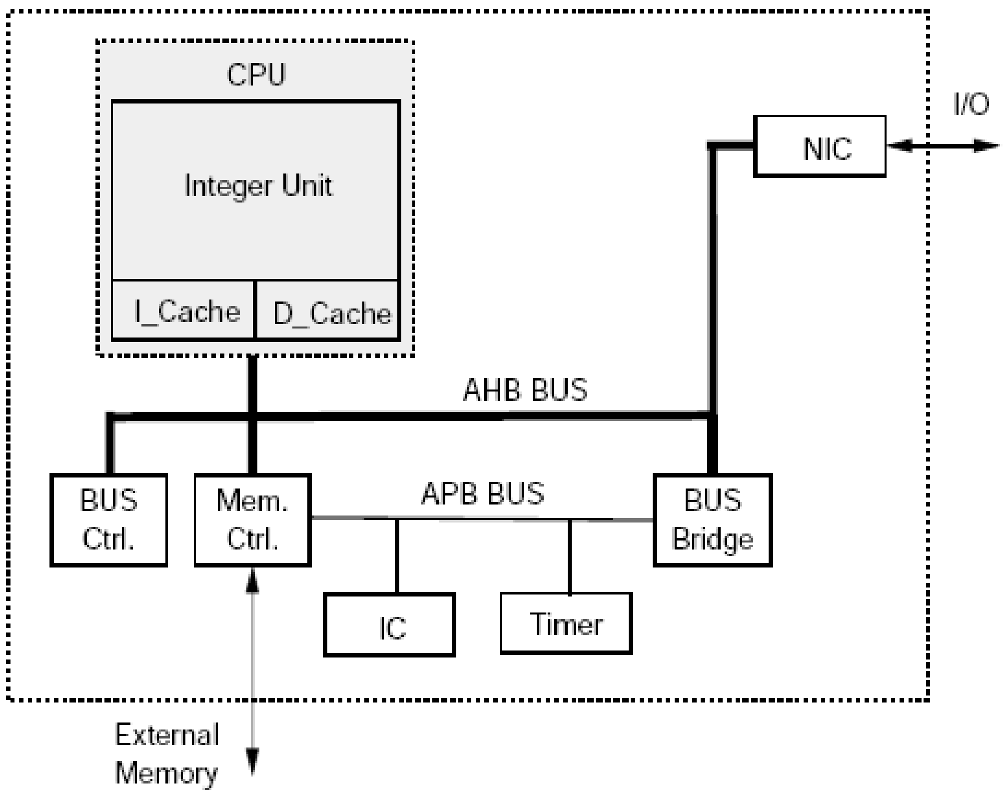
OS	.text(bytes)	.data(bytes)	.bss(bytes)
μCLinux	840,712	44,700	72,649
eCos	17,152	796	34,040
EPOS	8,988	28	8,400

OS	Time(ms)
eCos	132.67
EPOS	45.42

Configurable Feature Deployment



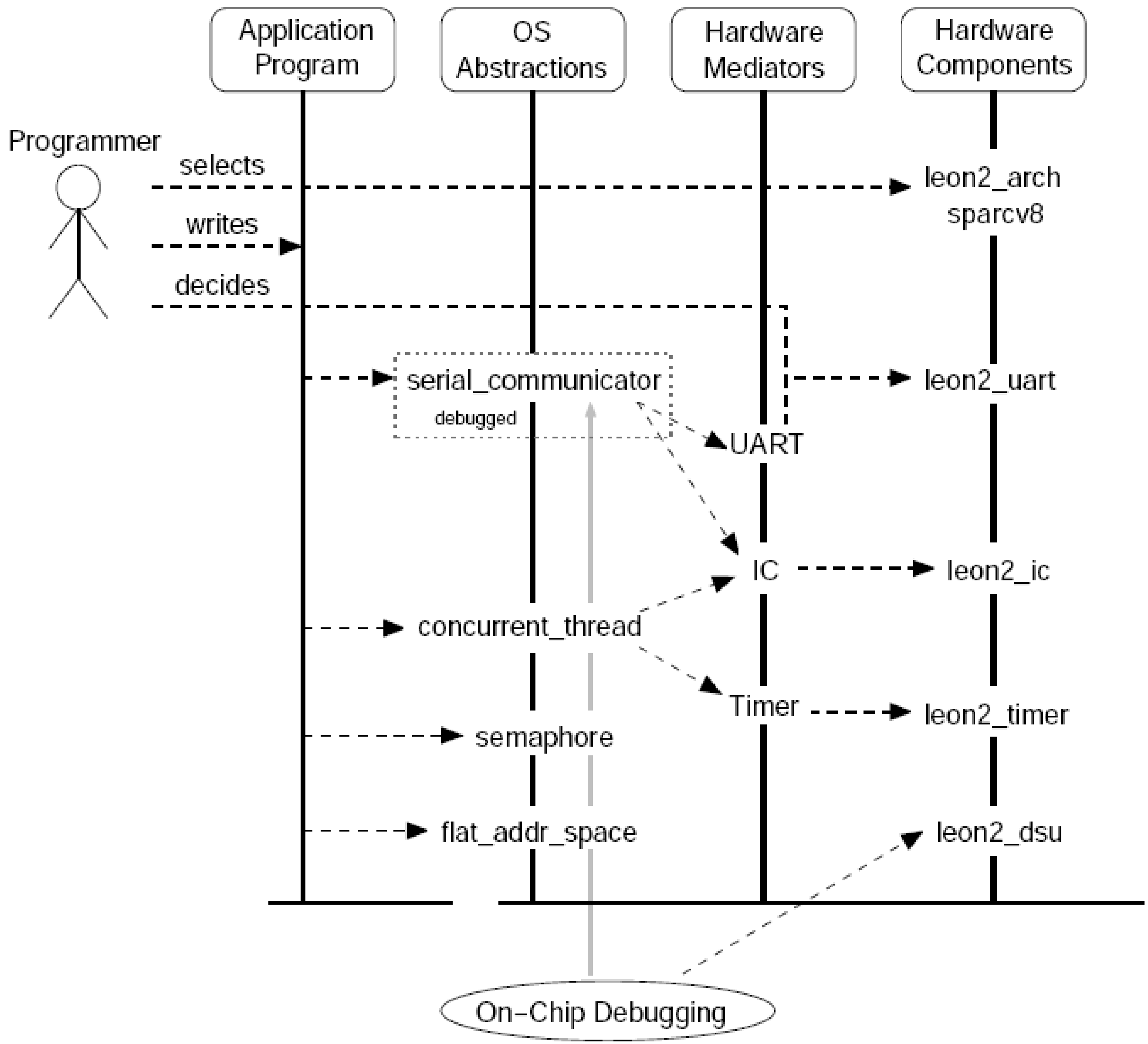
Tailored EPOS SoC



SoC Size: 8,302 LUTs

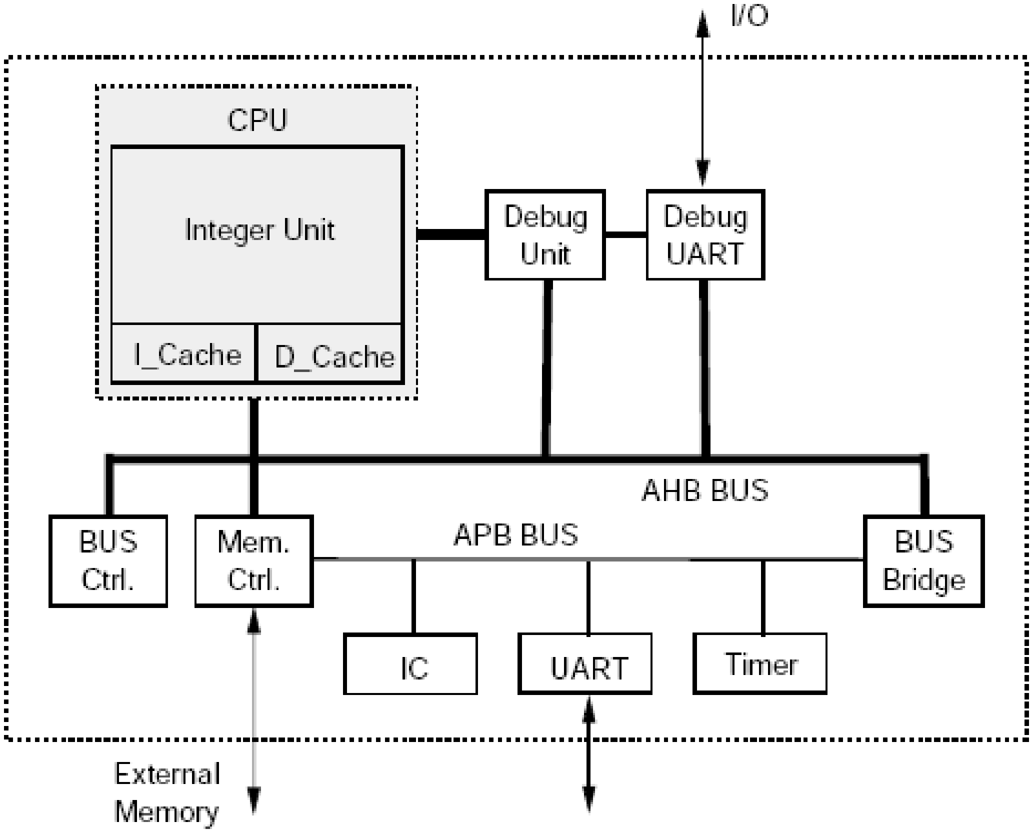
EPOS Size: 19,924 Bytes

Aspect Deployment



Federal University of Santa Catarina
Laboratory for Software/Hardware Integration

Tailored EPOS SoC



SoC Size: 10,428 LUTs

EPOS Size: 17,616 Bytes

Conclusions

- Hardware Soft IPs fit perfectly in EPOS
  - Described, selected and configured just like software components
- EPOS tools are now able to tailor a SoC
  - As long as the IPs have the proper design
  - A set of hardware mediators exists for the target machine
- Application-oriented SoC
  - Tools seem to be OK!
  - What about AOSD?
    - Develop our own soft-core processor (MIPS)
    - Aware of HDL restrictions (aspects, metaprograms, etc)