A Distributed Object System Approach for Dynamic Reconfiguration

Ronald Hecht, Stephan Kubisch, Harald Michelsen, Elmar Zeeb, Dirk Timmermann

{ronald.hecht, dirk.timmermann}@uni-rostock.de

University of Rostock,
Institute of Applied Microelectronics and Computer Engineering

April 25th, RAW 2006, Rhodes Island, Greece

Outline

Outline

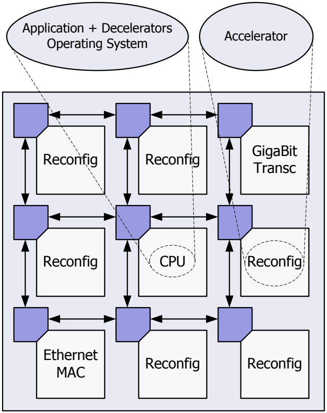
Dynamically Reconfigurable Systems

FPGA

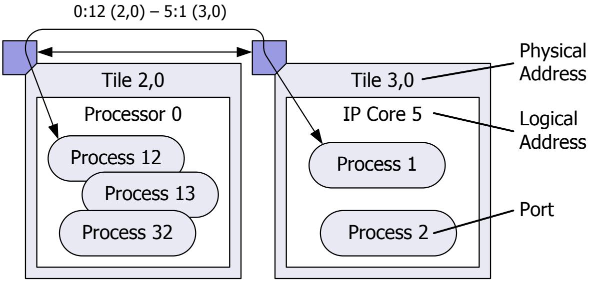
- Network-on-Chip
- Reconfigurable tiles with accelerators
- Embedded processor
- Other fixed cores

Processor

- Operating system
- Applications
- Decelerators



Tiles, IP Cores and Processes



Hierarchy of Reconfigurable Hardware

- Tile is configured with IP core
- IP Core runs processes
- NoC addressing scheme reflects hierarchy

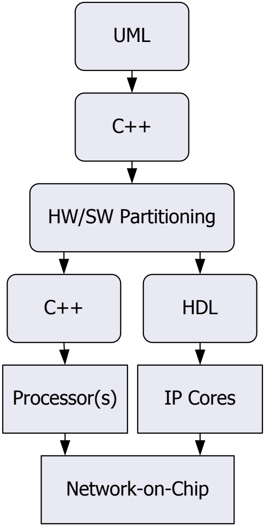
Simplifying the Design Process

Previous Work

- Object-oriented design
- UML entry
- Design Automation

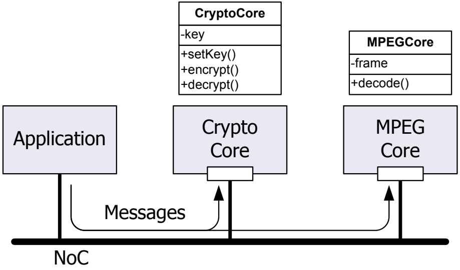
Looking for new methodologies to ...

- Unify software and reconfigurable hardware
- Hide dynamic reconfiguration
- Hide relocation
- Borrow from the software world



Outline

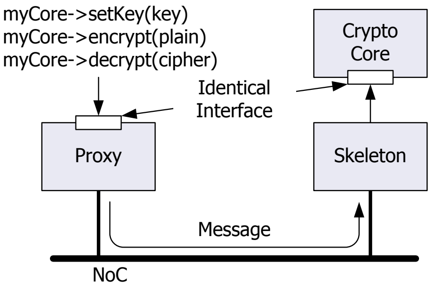
Abstracting the Reconfigurable System



IP Cores ...

- Are objects
- Have interfaces
- Communicate by the use of messages
- Are distributed in the NoC

Distributed IP Cores



Similarities

- Remote method invocation (RMI)
- Client proxy, Server skeleton
- Remote references to access IP cores

C++ Example – Explicit bind

AES Crypto Core

```
// Declare remote reference
AESCoreRef myAESCore;

// Explicit bind, Trigger dynamic reconfiguration
myAESCore = AESCore::getInstance();

// Remote method invocation
myAESCore->setKey(aKey);
myAESCore->encrypt(aMessage);

// Explicit unbind, Unload IP core
myAESCore->releaseInstance();
```

C++ Example – Implicit bind

AES Crypto Core

```
{
  // Declare remote reference
  AESCoreRef myAESCore;

  // Remote method invocation
  // Implicit bind, Trigger dynamic reconfiguration
  myAESCore->setKey(aKey);
  myAESCore->encrypt(aMessage);

  // Implicit unbind, Unload IP core
}
```

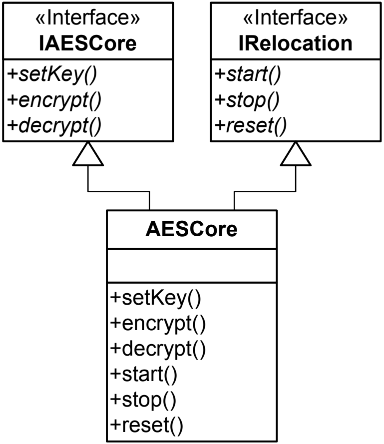
IP Core Relocation

Requirements

- Relocation between hardware and software
- Contexts must be compatible

Object System Approach

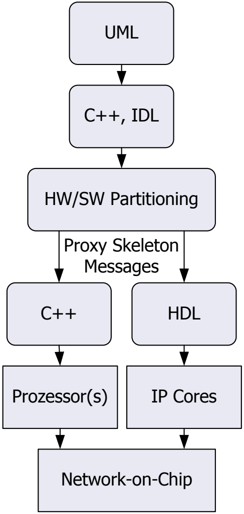
- Relocating objects requires serializing objects
- Extend the IP core interface
- Designer knows best about the context save



Design Flow

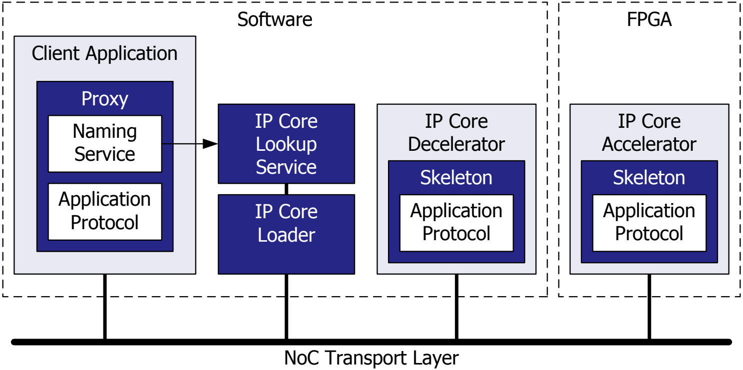
Facilitates Automation

- High-level design entry
- Late partitioning
- Border is defined by the designer
- Automatic generation of
  - Messages
  - Proxy
  - Skeleton
- Make use of IDL compilers



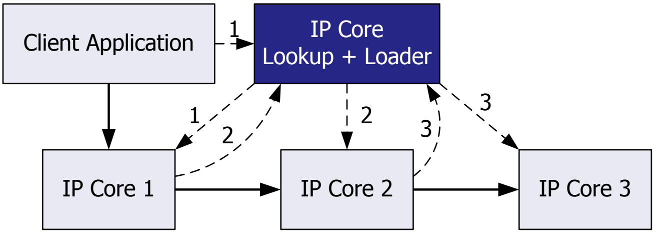
Outline

Distributed Object System



- Middleware simplifies using IP cores
- Remote references allow IP cores to access other IP cores
- IP core look-up service is visible within the network

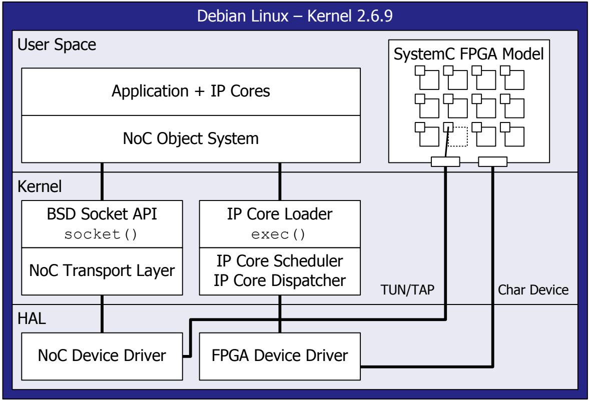
Self-triggered Dynamic Reconfiguration



New Approach

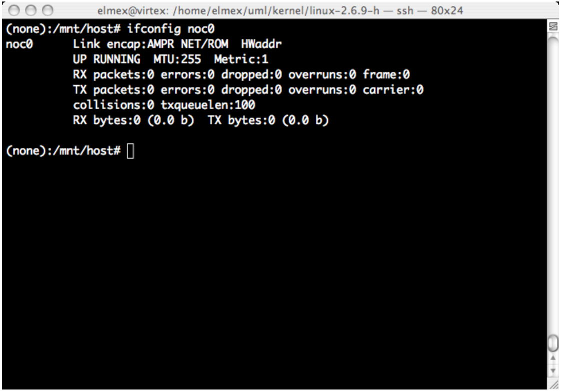
- IP cores are allowed to initiate dynamic reconfiguration
- Software, decelerators, and accelerators have equal rights
- Unifies dynamic reconfiguration for hardware and software

Linux Extensions



- Extension of BSD Socket API and `exec()` system call
- SystemC FPGA model for evaluation

NoC Integration



- NoC interface is a standard network device
- Easy to use. BSD Socket network programming

FPGA Integration

elmex@virtex: /home/elmex/uml/kernel/linux-2.6.9-h — ssh — 80x38

```
/proc/vce/devices:
total 0
dr-xr-xr-x  4 root root 0 Nov 11 11:46 0

/proc/vce/devices/0:
total 0
--w-------  1 root root 0 Nov 11 11:46 dispatcher
dr-xr-xr-x  2 root root 0 Nov 11 11:46 ipcores
-rw-r--r--  1 root root 0 Nov 11 11:46 sched
-r--r--r--  1 root root 0 Nov 11 11:46 state
dr-xr-xr-x  2 root root 0 Nov 11 11:46 tiles

/proc/vce/devices/0/ipcores:
total 0

/proc/vce/devices/0/tiles:
total 0
-r--r--r--  1 root root 0 Nov 11 11:46 0
-r--r--r--  1 root root 0 Nov 11 11:46 1
-r--r--r--  1 root root 0 Nov 11 11:46 10
-r--r--r--  1 root root 0 Nov 11 11:46 11
-r--r--r--  1 root root 0 Nov 11 11:46 12
-r--r--r--  1 root root 0 Nov 11 11:46 13
```

- OS extensions are accessible through /proc file system
- IP cores are loadable in the shell or with exec()

Outline

Summary

Approach

- Reconfigurable system is a distributed system in the small
- Distributed object system
- Applying a well-known and proven technology

Results

- Perfectly blends with an OOP design approach
- Automation with IDL tools possible
- Self-triggered dynamic reconfiguration
- Middleware and OS extensions