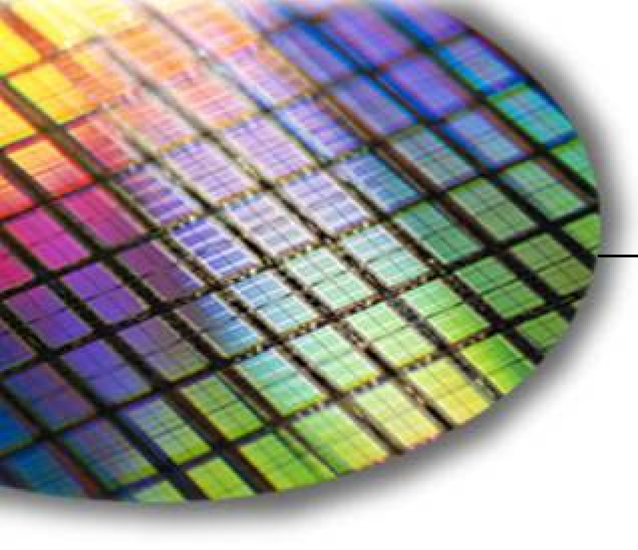
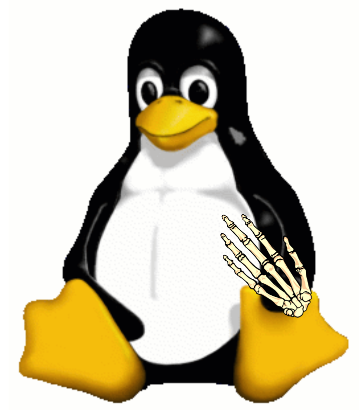


Heterogeneous Multicore

for blackfin implementation



Open Platform Solutions
Steven Miao

ANALOG DEVICES

Outline

- Modern SoC architecture
- Generic AMP/IPC framework overview
- Intercore communication protocol
- Userspace API for inter-processor communication(MCAPI)
- Q & A

The emergence of multicore

- With the ever-increasing demands for devices, multicore solutions are becoming more prevalent.
- Performance
- power management
- Costs

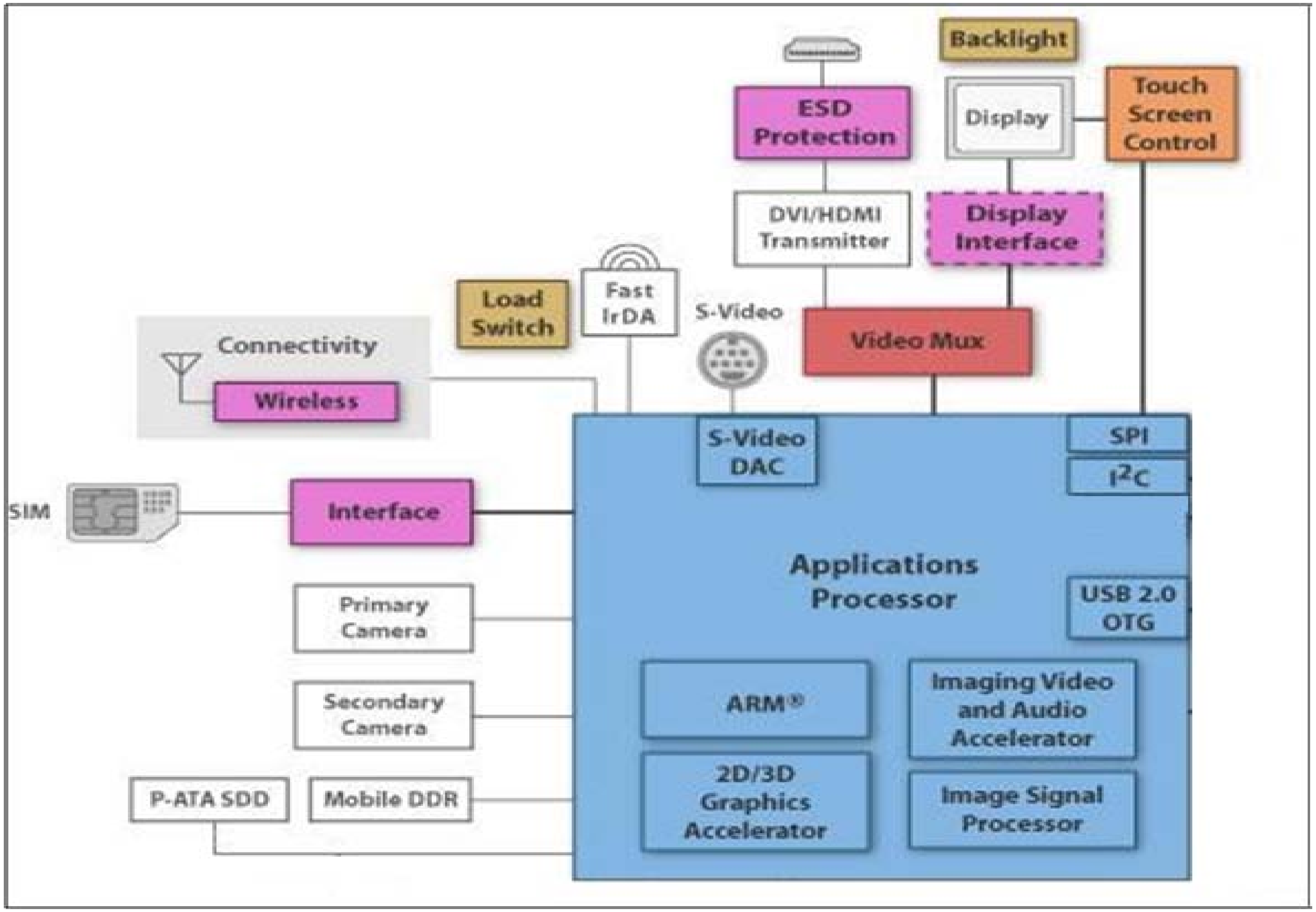
ANALOG DEVICES

Ingredients for multicore development

- Soc architecture
  - Homogenous cores
  - Heterogeneous cores
- Operating system, multicore processing
  - Smp
  - Amp
  - Hybrid
- Application
  - A collection of threads or tasks
- Inter-processor communications or message passing mechanism

ANALOG DEVICES

Today's SoC architecture

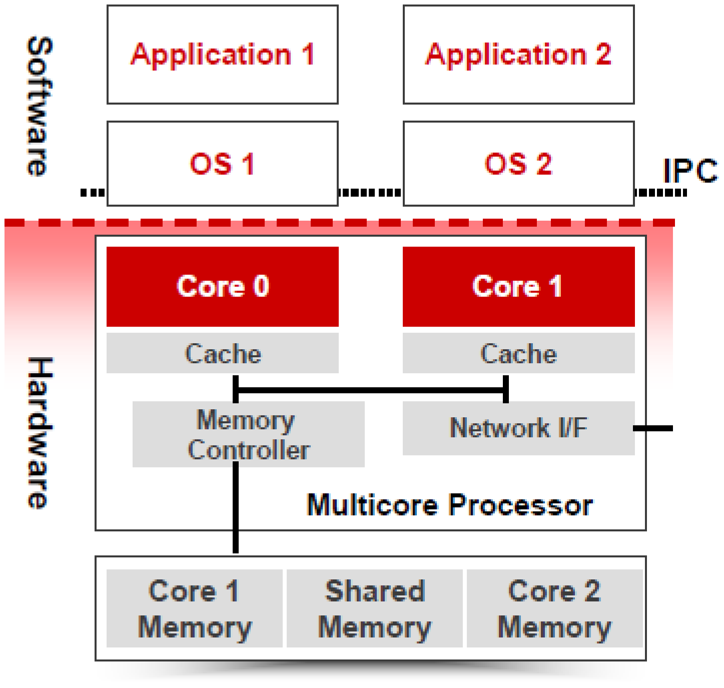


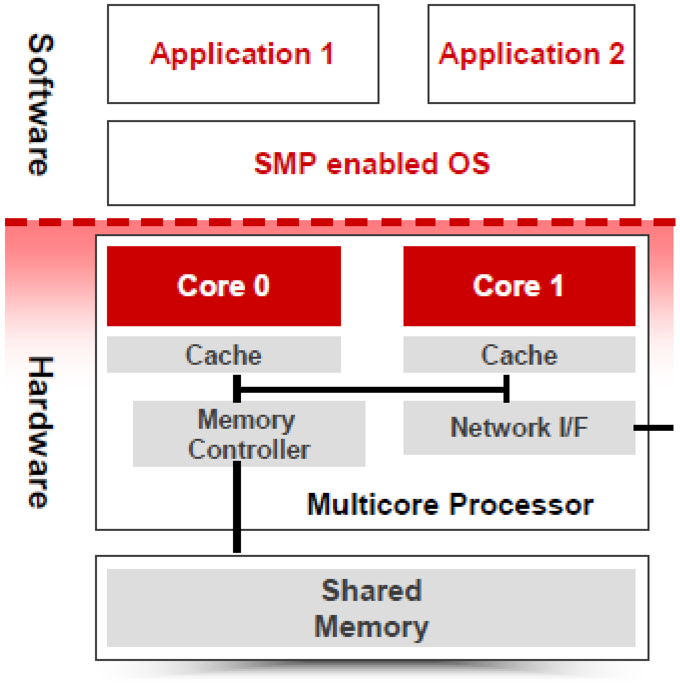
Different Types of multicore

- **Homogeneous multicore - Multicore chips coming to market all have the same instruction sets/data structures.**
- **Heterogeneous multicore – At least one processor in a multicore architecture is different, different instruction sets/data structures.**

Multicore processing

- AMP - A separate OS, or a separate instantiation of the same OS, runs on each CPU.
- SMP - A single instantiation of an OS manages all CPUs simultaneously, and applications can float to any of them.





Multi-core usage

- **Performance Increase system performance by implementing multicore hardware and software parallelism.**
- **Migration Avoid costly porting and software rewrite efforts**
- **Usability Enhance an existing real-time device by adding user-interface features on a general purpose operating system.**
- **Reliability /security Improve system robustness by protecting and isolating operating environments from each other.**
- **Even Safety-Critical Software**

Programming model on heterogeneous multi-core

- **Master and slave module**
- **Typical GPP runs operating system such as Linux, Microsoft Windows Embedded CE…**
- **The DSP as a slave to offload GPP or accelerate Application Specific task like real-time, audio/video codec, signal processing, runs OS as vdk, uc/os, nucleus**

A generic AMP/IPC framework

- **Some existing solutions**
  - **Icc of ADI**
  - **Syslink of TI (syslink, dsplink, dspbridge)**
  - **Qmi of qualcomm(qualcomm msm/modem interface)**

- **Generic framework features for embedded system**
  - **Control (power on, boot, power off)**
  - **Communicate**
  - **Synchronize**
  - **Resource manage**
  - **Minimal foot print**
  - **Source level portability**
  - **Scalable**

Hardware Requirements

- Each core can interrupt the other (required)
- Mailbox register for buffer transfer(optional)
- Shared Memory between cores (required)
  - 16-bit and 32-bit shared memory reads must be indivisible
  - Possible to either disable or flush the data cache on shared memory region
  - DMA to shared memory available from each core
- Same memory addressing on each core (desirable)
  - Address N in shared memory on one core is address N in shared memory on the other
- System-wide memory protection (optional)
  - Each core can protect areas of shared memory

ICC protocol of ADI

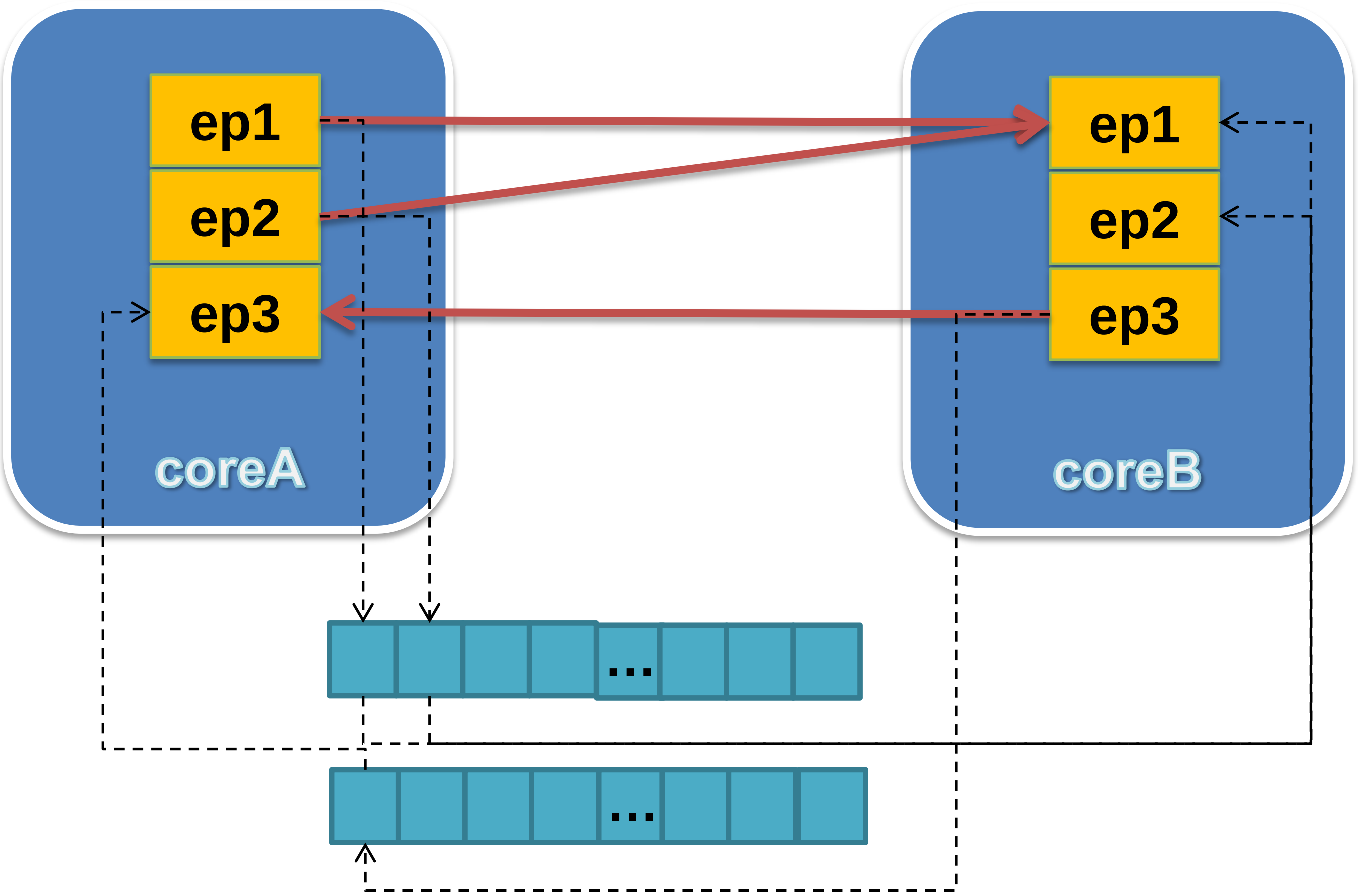
- Shared Memory based Inter-core Communication Protocol
- Message passing
- Dsp control
- Resource management
- Designed for heterogeneous multicore system
- first Implementation on BF561

Message Format

```
typedef struct
 {
        sm_endpoint_t dst_ep, src_ep;
        sm_uint32_t type;
        sm_uint32_t length;
        sm_address_t payload;
} sm_msg_t;
```

All protocol Types

Protocol type	Value	Protocol name
SP_CORE_CONTROL	1	Core control protocol
SP_TASK_MANAGER	2	Task manager protocol
SP_RES_MANAGER	3	Resource manager protocol
SP_PACKET	4	Connectionless packet transfer protocol
SP_SESSION_PACKET	5	Connection packet transfer protocol
SP_SCALAR	6	Connectionless scalar transfer protocol
SP_SESSION_SCALAR	7	Connection scalar transfer protocol

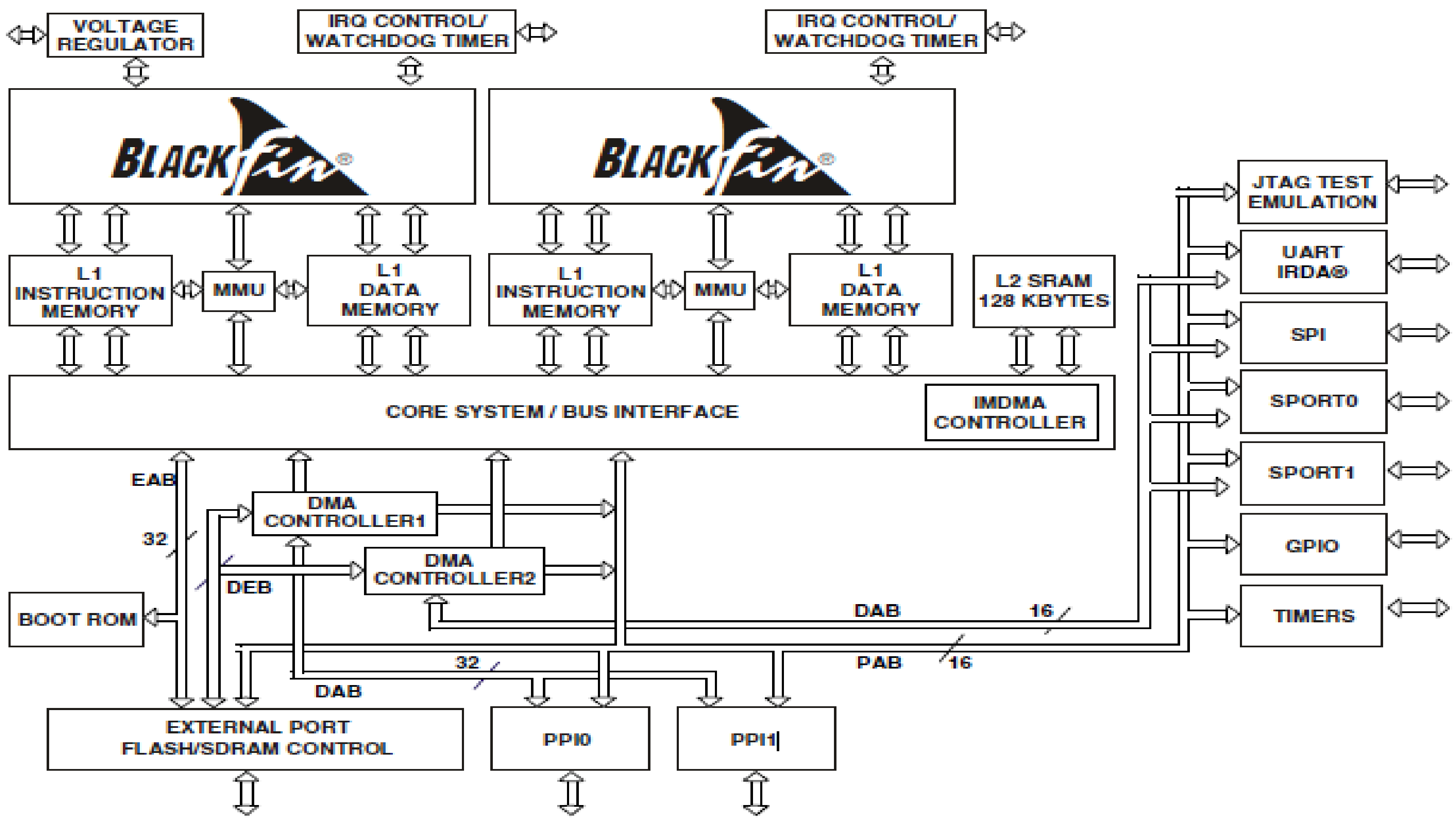

ep1
ep2
ep3
coreA
ep1
ep2
ep3
coreB
...
...

ICC on BF561

- L2 memory shared between coreA and coreB
- supplemental interrupt 0 to notifiy the other core

- ADSP-BF561

ADSP-BF561 Block Diagram



User Interface

- **Linux User Interface, device node /dev/icc**

```
ioctl(fd, CMD_DSP_START, NULL);
ioctl(fd, CMD_DSP_STOP, NULL);
```

- **DSP User Interface**

DSP initialization

Application initialization

Entry(icc_task_init)

Entry(icc_task_exit)

DSP task on core B

- **Link coreB application**

```
$(LD) -static $(LDFLAGS) -o $@ -T coreb_task.lds --just-symbol $(ICC_CORE)

coreb_task.lds:
 .text_l1        :
  {
    /* basiccrt.o(.text) */
    *(.l1.text)
  } >MEM_L1_CODE =0
 .text           :
  {
    /* *(EXCLUDE_FILE (*basiccrt.o ) .text) */
    *(.text .stub .text.* .gnu.linkonce.t.*)
    KEEP (*(.text.*personality*))
    /* .gnu.warning sections are handled specially by elf32.em.  */
    *(.gnu.warning)
  } >MEM_ICC =0
```

Run DSP task on core B

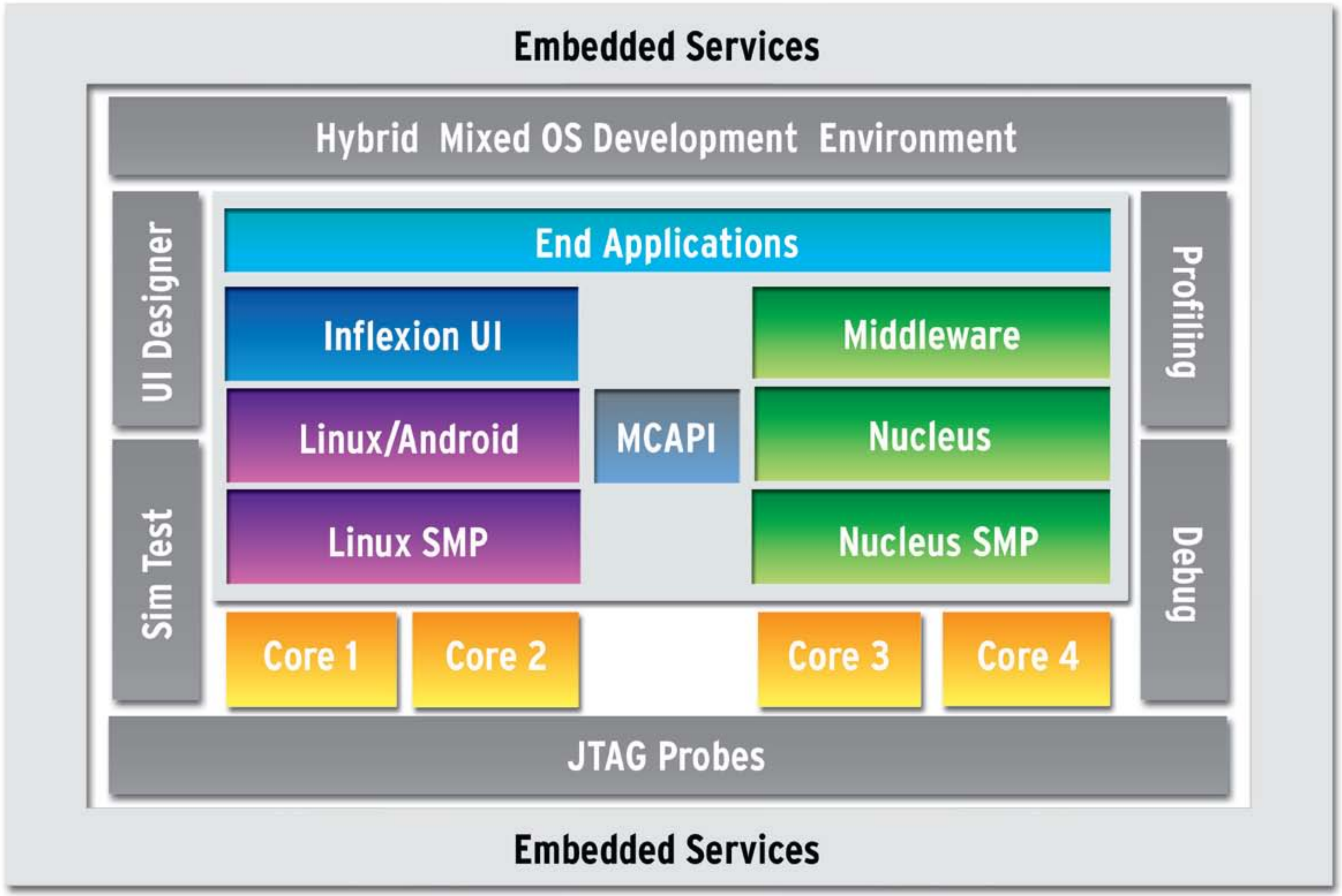
- Dynamic load by icc_loader

- Parsing the executable image file format(ELF, vdsp ELF)
- Find the task entries
- Loading and controlling the DSP bare metal application via core control protocol.
- Core B will start running task from task init entry

MCAPI for heterogeneous systems

- A unified communications API to handle this multitude of hardware and software
- Standardized
- Scalable
- Portable

An example of hybrid multicore



Goals of MCAPI

- Generic API for any target and OS
- Fast, lightweight, and scalable
- Multiple channels
- Flexible buffer management
- Portability

key concepts in MCAPI

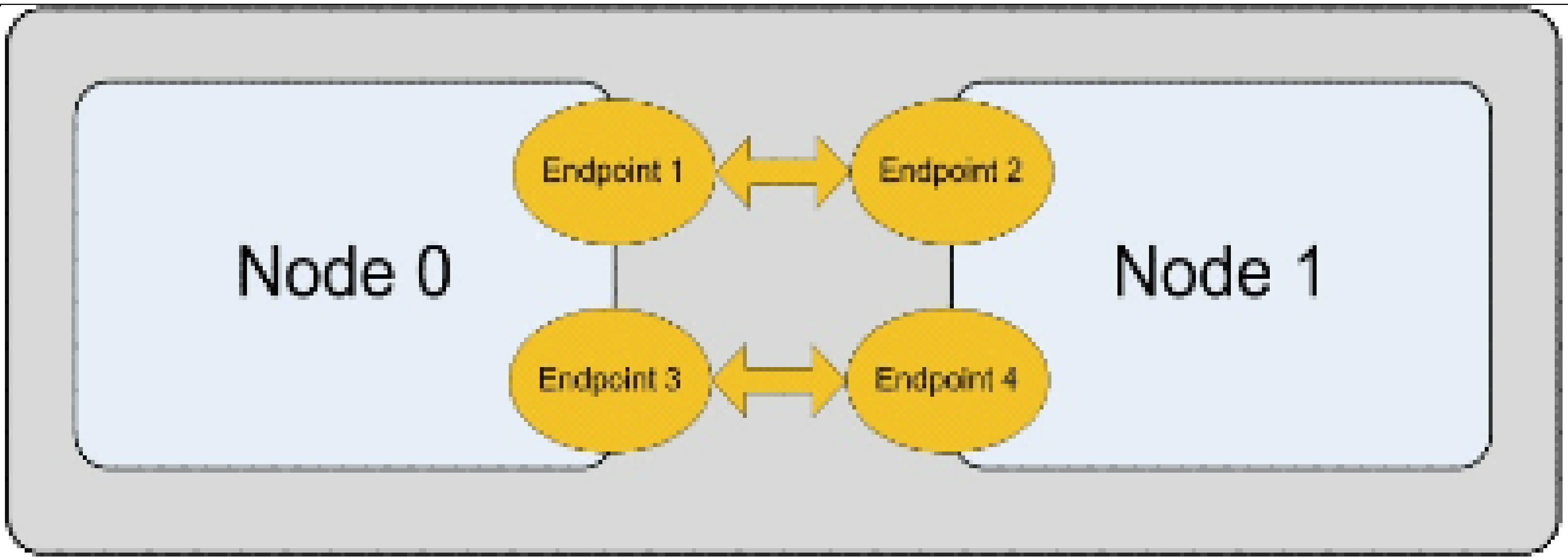
- **Endpoints, the start and end points of the communication**

<domain, node, port>

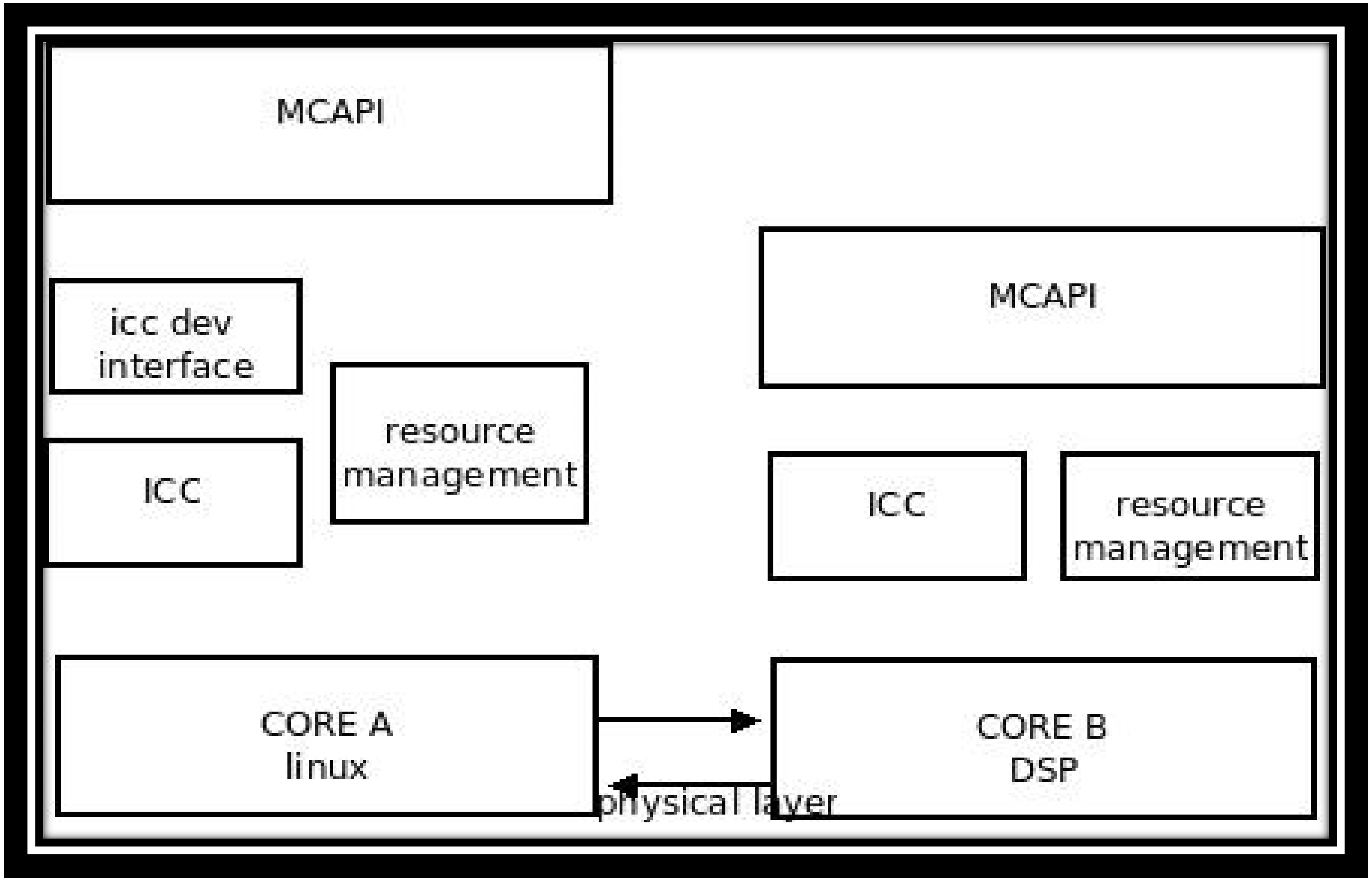
- **Channel**
- **Messages, connectionless, buffer communication**
- **Packet channels, connected, unidirectional , buffer communication**
- **Scalar channels, connected, unidirectional, scalar (8, 16, 32 or 64 bit) communication**

Mcapi topology

- Nodes exchange data via endpoints
- A node is implementation defined, i.e. a core, a OS, a thread
- An endpoint is created on a node to send/receive data
- MCAPI has no knowledge of the underlying transport mechanism, share memory, virtio, ethernet, linkport, etc.



MCAPI implementation on top of ICC



Resources

- https://docs.blackfin.uclinux.org/doku.php?id=protocols:icc#inter_core_communication_introduction

Q&A