Enabling the Full Power of a Multiprocessor SoC

Jeff Hancock
Senior Product Manager
Embedded Platform Solutions

December 10, 2019
Embedded Industry: Consolidation

Benefits
- Reducing BOM and power consumption
- Increasing performance and capacity of the system

Hardware Consolidation
- SOC
- Subsystems
- Systems

Concerns
- Complexity of the system design, development, debugging, isolation, functional safety

A block diagram of SiFive’s U54-MC Coreplex. (Image source: SiFive)
Trend toward Heterogeneous Multiprocessing in embedded System On Chips (SoCs) ..

There is a definitive need for Asymmetric Multiprocessing Paradigms!

Communications / share workload

Heterogeneous Software

Heterogeneous hardware

Application Cores
Real-Time Cores
DSPs
FPGA Soft cores

Linux
RTOS
Hypervisor
RTOS
Bare metal
RTOS
Bare metal
RTOS
Bare metal
RTOS
Bare metal

Boot
Multicore Hardware Types

- 3 broad possibilities:
  - Homogeneous multicore: all cores identical
  - Heterogeneous multicore: all cores different
  - Hybrid: multiple identical cores, but some different
    - May be high- and low-power options
    - Possibly regular CPUs and specialized cores
Symmetric Multi Processing (SMP)

- Provides overall higher compute bandwidth
- Maximum parallelization possible governed by Amdahl’s law
- OS scheduler balances workload across available cores in a homogeneous, cache-coherent core cluster
- Facilities provided to affine tasks to specific cores/core-groups
- A number of embedded RTOS products and Linux offer SMP capability
- SMP OSs can be deployed on metal (native) or in supervised environments (on a Hypervisor)
Asymmetric Multi Processing (AMP)

- Multi-OS Model
- System designer chooses the right OS environment and processing core for the application workload
- May contain SMP compute domains
- Well suited for consolidation of system functions
  - Real-time/Non Real-Time (HMI)
  - Trusted /Un Trusted
  - Safety certified/ Non certified

- Design considerations
  - Separation requirements
  - System Resource sharing requirements
  - Lifecycle management of cores and associated software contexts
  - Communications between OS contexts

- Further categorized into
  - Unsupervised systems with no separation requirements
  - AMP systems with separation requirements
  - Supervised, systems with separation and/or virtualization requirements
Asymmetric Multi Processing (AMP) using OpenAMP/Multicore Framework

- Multi-OS model
  - Independent OS instances on independent cores
  - Cores and OS instances can be heterogeneous

- Each OS runs natively on the assigned core
  - No separation
  - Co-operative coexistence

- On demand bring-up of compute and software
  - Low power
  - Compute offload/acceleration

- OpenAMP Open Source project
  - remoteproc - Life cycle management
  - rpmsg - Inter-Processor (Inter-OS) Communications

- OpenAMP is a standalone library that enables RTOS and Bare-metal environments

- OpenAMP is compatible with upstream Linux remoteproc and rpmsg components

- Mentor Embedded Multicore Framework
  - Commercial implementation of OpenAMP
Multicore Issues

- Inter-CPU communications
- Inter-CPU safety
- Boot order
- System integrity
Multicore Management

- **Unsupervised**
  - Framework
    - OpenAMP / Multicore Framework

- **Supervised**
  - Hypervisor

- **Hybrid**
Multicore Management

- **Unsupervised**
  - Framework
    - OpenAMP / Multicore Framework

- **Supervised**
  - Hypervisor

- **Hybrid**
Unsupervised

App 1
OS 1
MEMF
CPU

App 2
OS 2
MEMF
CPU

App 3
OS 3
MEMF
CPU

App 4
OS 4
MEMF
CPU

App 5
OS 5
MEMF
µC

App 6
OS 6
MEMF
µC
Unsupervised
Unsupervised

App 1  App 2  App 3  App 4  App 5  App 6

OS 1  OS 2

CPU  CPU  CPU  CPU  µC  µC

MEMF  MEMF
Multicore Management

- Unsupervised
  - Framework
    - OpenAMP/Multicore Framework

- **Supervised**
  - Hypervisor

- Hybrid
Supervised - Hypervisor
Multicore Management

- Unsupervised
  - Framework
    - OpenAMP/Multicore Framework

- Supervised
  - Hypervisor

- Hybrid
Hybrid

App 1
 RTOS
 CPU

App 2
 Linux
 CPU

App 3
 Bare Metal
 CPU

App 4

Hypervisor
 MEMF

MEMF

RISC-V Summit 2019
System Implementation

Mixed time domain

Mixed criticality
System Implementation

Mixed time domain

Mixed criticality
Mixed time domain

Each CPU/OS doing what it is good at

- Hard real-time
- Deterministic – predictable
- Responsive
- Compact code
- Fast execution
- Fast start-up

- Non-real-time
- User interface
- Number crunching
- Data processing
- External storage etc.

- Probably real-time
- Optimum CPU performance
- No overheads

App 1: RTOS
App 2: Linux
App 3: Bare Metal
App 4: MEMF
System Implementation

Mixed time domain

Mixed criticality
Critical systems

- Secure systems
  - Banking infrastructure
  - POS terminals
  - ATMs
  - Medical

- Safety critical systems
  - Automotive
  - Mil/aero
  - Medical

- Certification
Certification

- Expensive
- Time consuming
- Size of code matters
  - Small code -> less time/cost
- Operating system
  - Typically cannot be certified alone – whole system is certified
  - May be “pre-certification” option
  - OS certification track record is key
Mixed criticality
Mixed criticality

- App 1: RTOS
- App 2: Linux
- App 3: Bare Metal
- App 4: Certified RTOS
- App 5: MEMF
- App 6: MEMF

No certification

Certification
Mentor Embedded comprehensive offering for Multiprocessor Software Development

- Mentor Embedded Linux Flex OS
- Mentor Embedded Linux Omni OS
- Nucleus RTOS
- Nucleus SafetyCert RTOS
- Mentor Embedded Hypervisor
- Mentor Embedded Multicore Framework
- Mentor Embedded Multicore Framework Cert

Sourcery CodeBench Development Tools

Develop

Profile

Debug

Deploy

Sourcery Analyzer

RTOS

Linux

Other sources