Take Control of Your Multicore Debugging

Rafael Taubinger – Sr. FAE
Agenda

• Company Overview
• Symmetric vs. asymmetric multicore processing
• Homogenous vs. Heterogeneous multicore processors
• Reasons for using multicore processors
• Programming and Debugging multicore devices
• ETM trace in multicore environments
• Debugging Demo
• Questions?
Providing developers of embedded systems with world-leading software tools

Global professional technical support in 9 languages

Large ecosystem of partners

10 offices worldwide with HQ in Uppsala, Sweden

Uppsala, Munich, Paris, Tokyo, Seoul, Shanghai, San Francisco, Dallas, Boston, Los Angeles

+ Distributor representation in 43 countries

33 years in the industry

32% of revenue invested in R&D

Listed on NASDAQ/Stockholm
Complete offering for ARM

- **Unique independence** with support for all available ARM cores, from all major vendors including TI, Renesas, NXP, ST, Cypress, Toshiba etc.
- **4,500+** supported devices in total
- **8,400+** example projects to enable quick start
- Close cooperation with several SoC vendors

Take full control of your development!

C-STAT:
Powerful static code analysis

C-RUN:
Runtime execution analysis

I-jet and I-jet Trace:
Debugging and trace probes
What is Multicore?
SMP vs. AMP

Symmetric Multicore Processing (SMP):

- Each core runs the same code from the same memory
- Requires homogenous system

Asymmetric Multicore Processing (AMP):

- Each core runs its own code or part of the application
- Cores are independent of each other
- Can be done with both homogenous and heterogeneous multicore processors
Heterogeneous multicore processors

Different cores share a main memory and peripherals

Can be used for application that need both real time performance and signal processing capabilities

Some available devices:
Texas Instruments OMAP Family (ARM + DSP)
NXP i.MX7Solo (Cortex-A7 + Cortex-M4)
NXP LPC43xx (Cortex-M4 + Cortex-M0)
NXP LPC54xxx (Cortex-M4F + Cortex-M0+)
NXP Vybrid VF6x (Cortex-A5 + Cortex-M4)
Homogenous vs. Heterogeneous

Homogenous multicore processors (MCP)

Two or more identical processors (cores) share a main memory, peripherals, interrupt controller etc. and execute the same code (SMP) or different code (AMP)

Each processor has its own local memory, stack and registers

Some available devices:

- Intel Core Duo
- NXP i.MX6Q (4 x Cortex-A9 up to 1.2 GHz)
- XILINX Zynq 7xxx (FPGA + 2 x Cortex-A9 up to 1 GHz)
- ST SPEAr1340 (2 x Cortex-A9 up to 600 MHz)
Why SMP/Homogenous MCP?

High performance requirements

Maximum Clock is limited, e.g. to 1-2 GHz on a ARM Cortex-A9

Higher performance requires more physically independent CPUs or multicore processors

Multi-core offers easier communication, board layout etc. vs. multi-device

```plaintext
CPU-1
N MHz

CPU-2
N MHz

vs.

CPU-1
N MHz

CPU-1
N MHz
```
Why SMP/Homogenous MCP?

Power consumption

Power dissipation increases exponentially with increasing operating frequency due to higher leakage (simplified)

E.g. N * 1 GHz consumes less power than 1 * N GHz

Increased throughput

Instructions per cycle increases with roughly $\sqrt{N}$

Mainly relevant for multi-tasking applications
Why AMP/Heterogeneous MCP?

Applications with different constraints, e.g. throughput vs. interrupt latency

Compromise on Sub-optimal architecture for a given requirement?

Multi-chip design?

Multi-core processor?

CPU-1

vs.

CPU-1

CPU-2
Programming and Debugging in Multicore Systems
Programming homogenous/SMP MCP

Traditional programming is single-processor-oriented and not easily split among cores.
Effective utilization and resource management vs. extra burden for the programmer
An Operating System helps

- Distribution of tasks/threads across the cores
- Load balancing
- Handling of inter-processor communication
- RTOS Example: ThreadX from Express Logic
Programming heterogeneous/AMP MCP

Both cores are independent and run their own application
No extra software design challenges
Only communication between cores through shared memory to be considered
Debugging multicore processors

Chip requirements:
Multicore processor should only have 1 physical debug interface rather than 1 per core (ease of use, space, cost etc.)
Debug Speed should not be impacted by other core, i.e.
   - Cores can run at different clock
   - A core can go to sleep without impacting debugging the other one

JTAG chain is not a good solution
Single Debug Access Port (DAP) in ARM CoreSight
Needs to be supported from debug tools
ARM CoreSight™ On Chip Debug & Trace IP

Source: Arm Ltd.
Debugging multicore processors

Tool requirements:

Visibility of all cores
Start and stop cores simultaneously or individually
Step 1 core while others are running or stopped
Multicore breakpoints, e.g.
  BP on 1 core stops execution on all cores
  BP on core A with condition on core B

Multicore Trace

Very challenging for Heterogeneous MCPs with different Trace capabilities, e.g.
Cortex-A9 has PTM (Program Trace)
Cortex-M3 or Cortex-A5/A8 has ETM
Some have even data trace
IAR Embedded Workbench SMP Support

IAR Embedded Workbench support today:
1 project and debugger instance for all cores
cores can be stopped/run individually or together
IAR Embedded Workbench AMP Support

- Start/stop core0/core1
- Start/stop all cores

Master (Cortex-A)  Slave (Cortex-M4)
I-jet Trace for ARM Cortex-A/R/M

- SuperSpeed USB 3.0 interface (5 Gbps)
- Fully compatible with USB 2.0 (480 Mbps)
- No power supply required, powered entirely by the USB port
- Target power of up to 600mA can be supplied from I-jet Trace with overload protection
- Automatic core recognition
- JTAG, SWD, target power consumption measuring

Trace features
- Trace memory size up to 1 Gbyte
- Up to 16-bit wide trace data collection
- Up to 350MHz ETM trace clock
- 64-bit timestamp with CPU cycle accuracy for timing analysis
- Automatic alignment of parallel trace data skew on individual bits to compensate for PCB layout and signal integrity problems
- Automatic trace data and clock voltage threshold adjustments to get the most reliable capture with noisy or un-terminated target boards
- Support for trace logic levels from 1.2 V to 5V
Full control!

Function profiling
Based on simulator, sampled trace or full trace
Execution time per function
Select time interval

Timeline window shows the application’s profile
Interrupt log, Data log, Event log, Call stack

Stack analysis
calculates maximum stack usage, helps find the optimal stack size, and checks stack integrity at runtime to detect overflow

Code coverage analysis
Which code has been executed?
Demo
Questions?
www.iar.com

Thank you for your attention!