ARM11 MPCore and its impact on Linux Power Consumption

John Goodacre
Program Manager - Multiprocessing
ARM Ltd

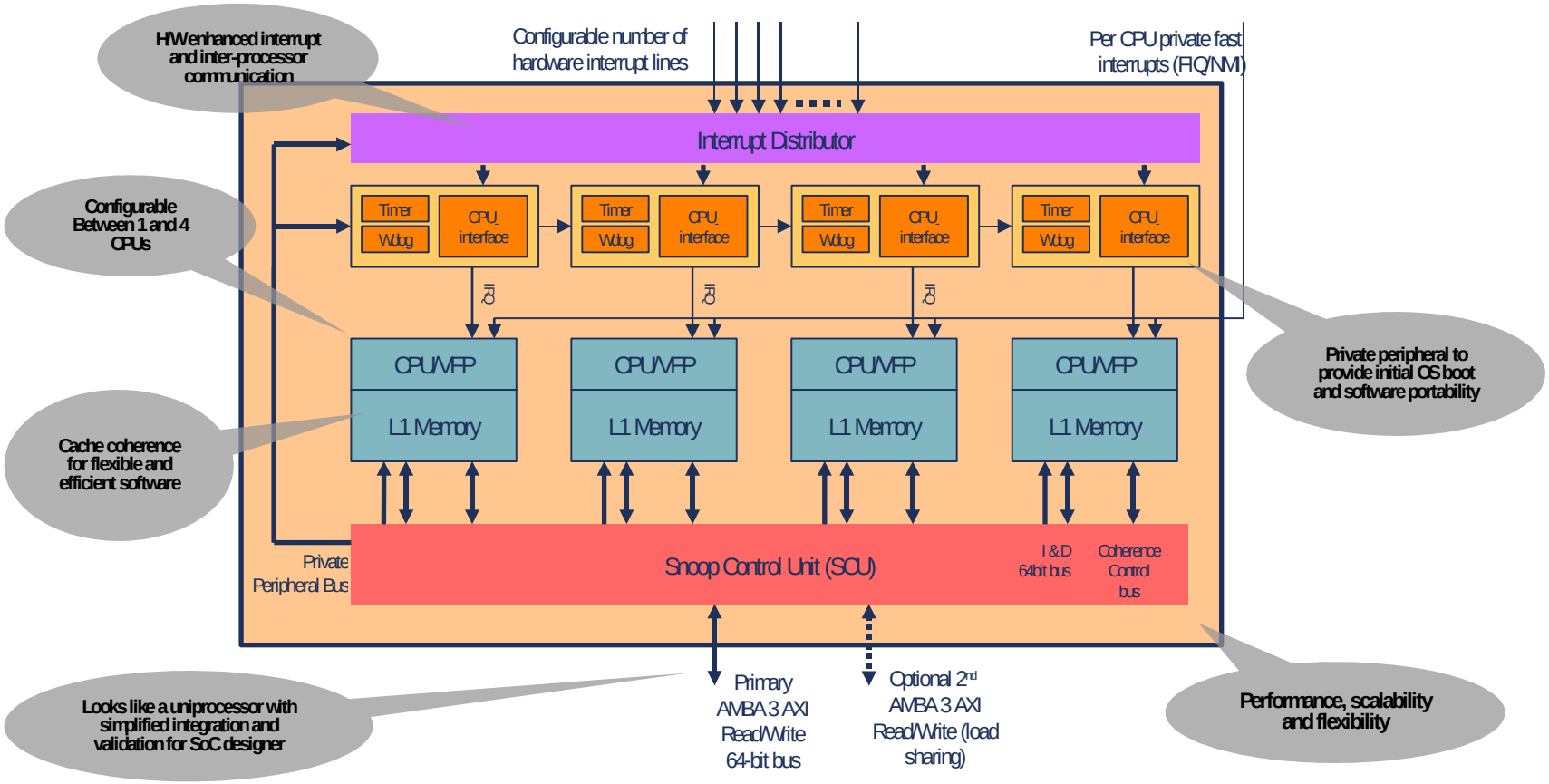
Why did ARM build the MPCore ?

- Embedded designers are always looking in the next generation more performance and/or lower power
  - ARM brought the Cortex-A8 uniprocessor answers this for non-MP software through higher MHz and low-power design methodologies
  - ARM brought the ARM11 MPCore multiprocessor to answer this for MP aware software through duplicating processors and lower MHz by sharing CPUs
- Its now clear that there is an industry-wide adoption of multicore for reasons of providing higher performance and lower power
  - ARM designed MPCore as a multicore processor that wasn't simply multiple uniprocessors sharing a bus
- The longer-term future is very multicore / multiprocessor

MPCore: What's it look like?

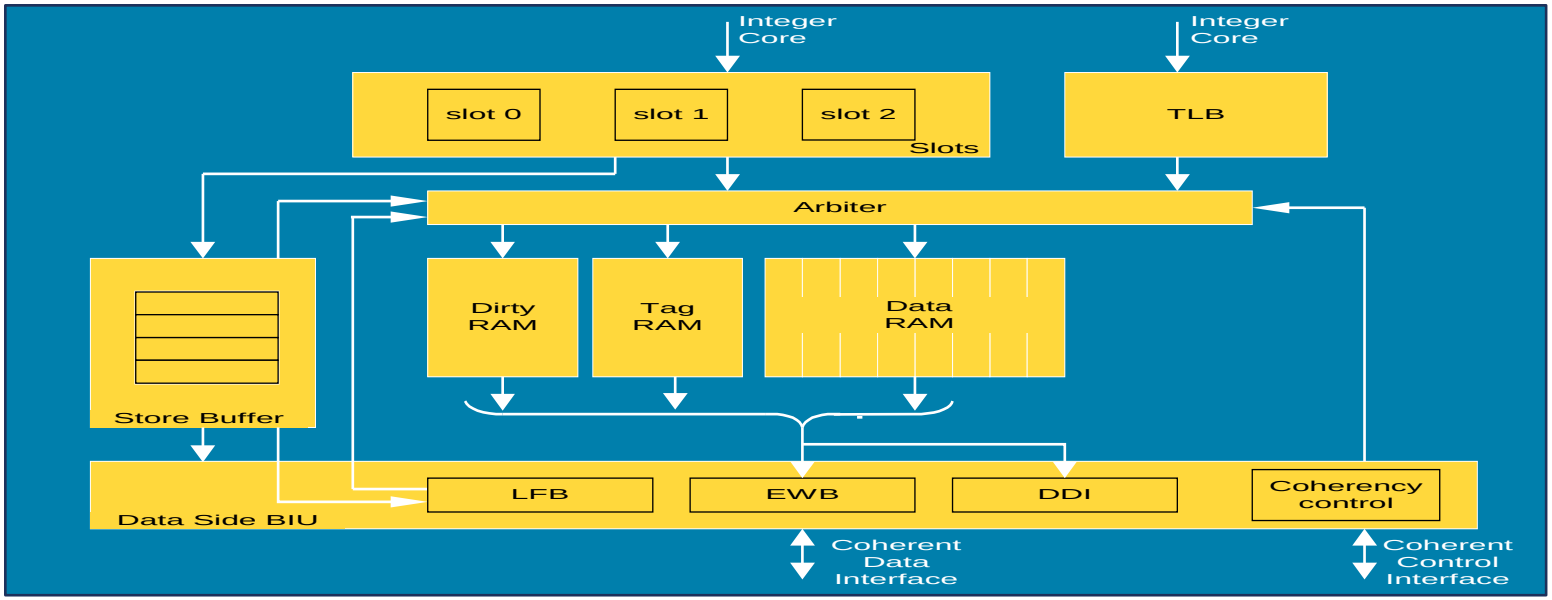
RTL synthesis configurations to define scalability between 1 and 4 CPUs

- With the design addressing the key scalability and bottlenecks of traditional MP design
- Interrupt distributor for high throughput and low latency inter-processor communications
- Snoop control unit for high performance and power efficient cache coherency

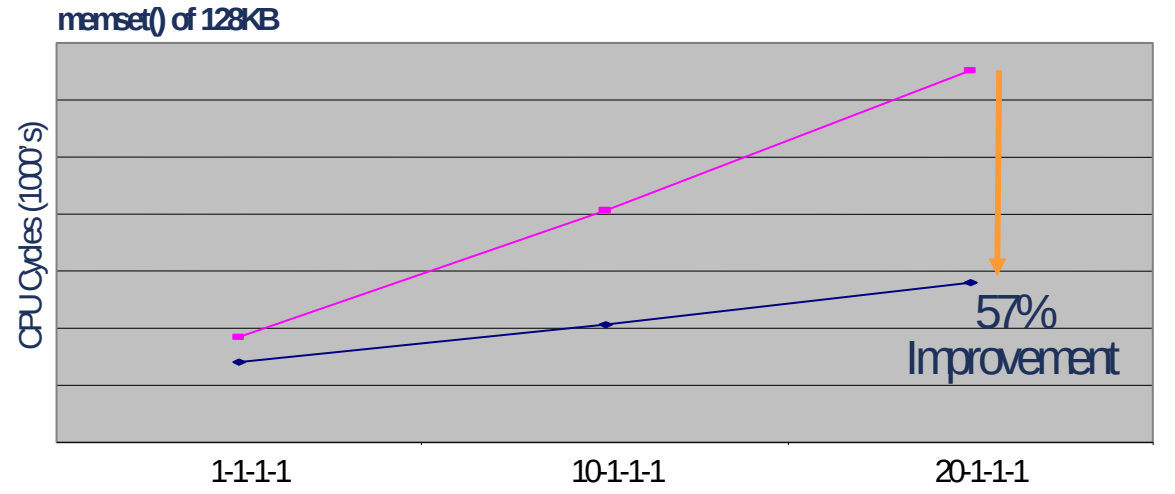


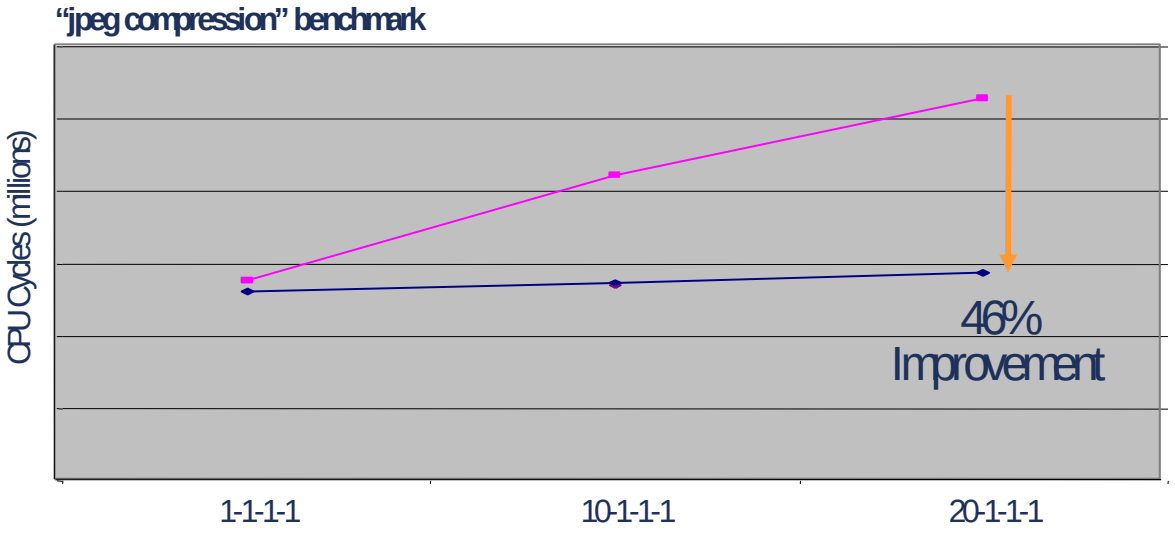
Enterprise capable memory system

- Merging Store Buffer with forwarding for improved bus utilization
  - Saving up to 70% of the CPU cycles wasted due to memory latency
- Physically indexed, physically tagged data cache using 'cork-screw' memory and buffers
  - Allowing single cycle allocation/eviction of cache lines
  - Reducing the software cost from flushing and de-aliasing of data cache
- Scalable to multiple processor designs
  - With full data cache coherence and cache-2-cache transfer capabilities
- Allocates cache line on both read-miss and read-write
  - Reducing the bus write load by up to 50%
  - Automatic adjustment to back-off from write-allocate when necessary



Effect of MPCore's enhanced L1





Memory throughput improvement due to MPCore's new L1 memory system

- Providing better memory bursting
- Providing higher performance from higher latency memory
- Reducing power consumption by less memory activity by around 14%

Without enhancement

With enhancements

Dynamic and leakage power

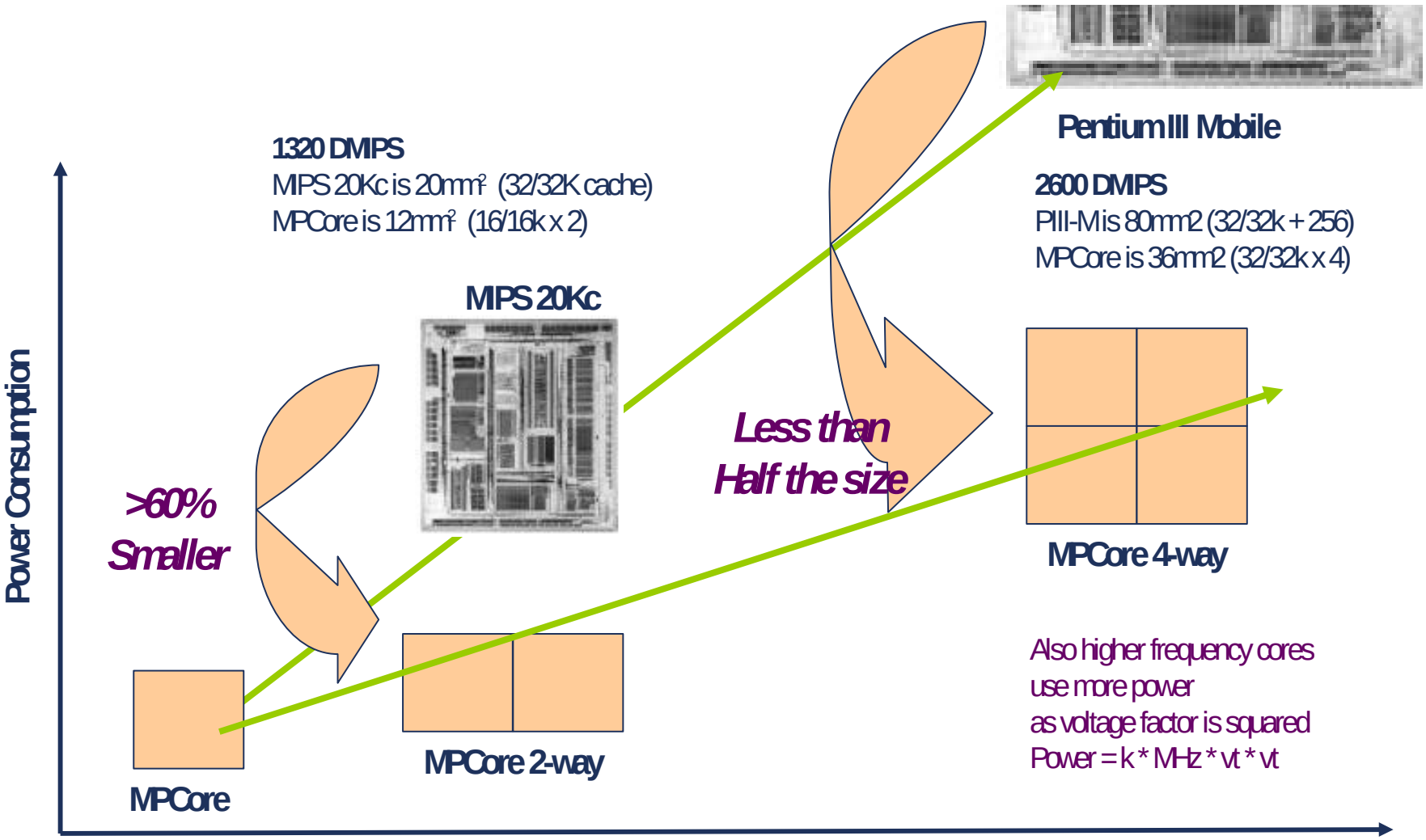
Dynamic energy is consumed when clocking logic and is related to the logic complexity to accomplish a given task

- Long core pipelines with advanced logic functions take more energy to compute a comparable simple operation on a simple RISC core
- There is a non-linear relationship between the amount of logic required to achieve a high-frequency, high throughput design

Leakage power is consumed whenever logic or RAM has power applied

- Getting worse as fabrication geometries reduce below 130nm
- Leakage worsen when you attempt high MHz in a given process
- Broadly speaking, total leakage goes up as die area goes up

The cost of more performance



Comparisons from public information. All processors using 130nm process

Multicore Processing

Higher performance per mm^2 than a uniprocessor design using the same implementation process

- Offering higher performance at lower cost

Lower mW per "DMIP" than a uniprocessor implemented in the same technology of equivalent performance

- Offering longer battery life / lower cooling without sacrificing performance
- Supporting partial shutdown of process to further extend the power controls of a typical uniprocessor with standby, voltage and frequency scaling techniques

Same die size as a multithreaded processor of the same performance

- Removing any reason to using a high design risk multithreaded uniprocessor
- With the advantage of predictable performance and design scalability
- …and without the need to continue to push the MHz for higher performance

Adaptive Shutdown to Standby

Maintains coherence while in standby

- Allowing immediate entry without any preceding cache house keeping
- Allowing for 2 cycle exit, and back into active service
- Does not materially effect the latency of the system

Dynamic energy is saved for entire CPU whenever no task is schedulable on CPU

- Consequence is a direct relationship between consumed dynamic energy and computation accomplished
- ARM ISA offers WFI (wait for interrupt) instruction to hint to enter standby

```
/*
 * The idle thread.  We try to conserve power, while trying to keep
 * overall latency low.  The architecture specific idle is passed
 * a value to indicate the level of "idleness" of the system.
 */
void cpu_idle(void)
{
        local_fiq_enable();

        /* endless idle loop with no priority at all */
        while (1) {
                void (*idle)(void) = pm_idle;

#ifdef CONFIG_HOTPLUG_CPU
                if (cpu_is_offline(smp_processor_id())) {
                        leds_event(led_idle_start);
                        cpu_die();
                }
#endif

                if (!idle)
                        idle = default_idle;
                leds_event(led_idle_start);
                while (!need_resched())
                        idle();
                leds_event(led_idle_end);
                preempt_enable_no_resched();
                schedule();
                preempt_disable();
        }
}
```

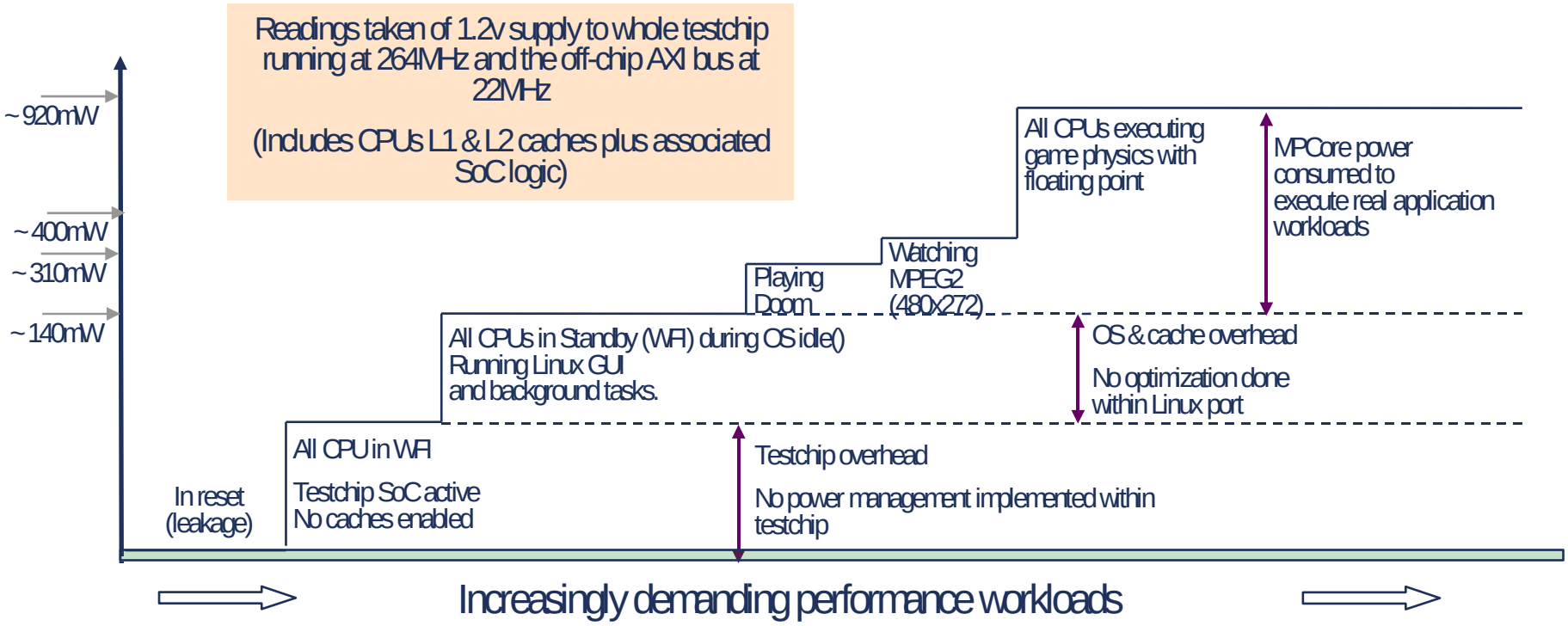
See ./arch/arm/kernel/process.c

Measured Low Power Consumption

Using the MPCore's built-in Adaptive-Shutdown to Standby

Offering a 50% reduction in average power consumption

For further power savings, MPCore supports Adaptive Shutdown to both Dormant and Reset, ***and*** Dynamic Voltage and Frequency Scaling (IEM), to lower the power consumption by over 85%



Adaptive Shutdown to Reset (/Dormant)

Power save scheme to remove voltage applied to core logic and RAM, and thereby save all associated dynamic and leakage current

- Leakage becoming significant part (30-50%) of consumed energy

MPCore allows individual CPU within the SMP cluster to isolate themselves from the coherence domain

- Also used by designs requiring processor isolation to run AMP software
- Requires all dirty cache data to be evicted before coherence isolation
- MPCore defines external signalling to tell SoC to also remove power when software next enters standby
- Wake up is via another CPU telling the SoC, and the awoken CPU reloading any processor state and rejoining the coherence domain

HOTPLUG Integration

See from Kernel v2.6.15 ./arch/arm/mach-realview/hotplug.c

Device /sys/devices/system/cpu/cpu[0-3]/online

- Write "0" to unplug the CPU from the SMP cluster and power it down
- Write "1" to bring the CPU back on line
- Read to find current state
- Illegal to unplug CPU0

Unplugging CPU isolated it from been available to scheduler

Precise implication is architecture dependent, for ARM

- Ensure no hardware interrupt set to be distributed to CPU
- Removes the CPU from the coherence domain
- Interacts with the SoC power controller to request power isolation from both CPU logic and RAMs associated with the CPU

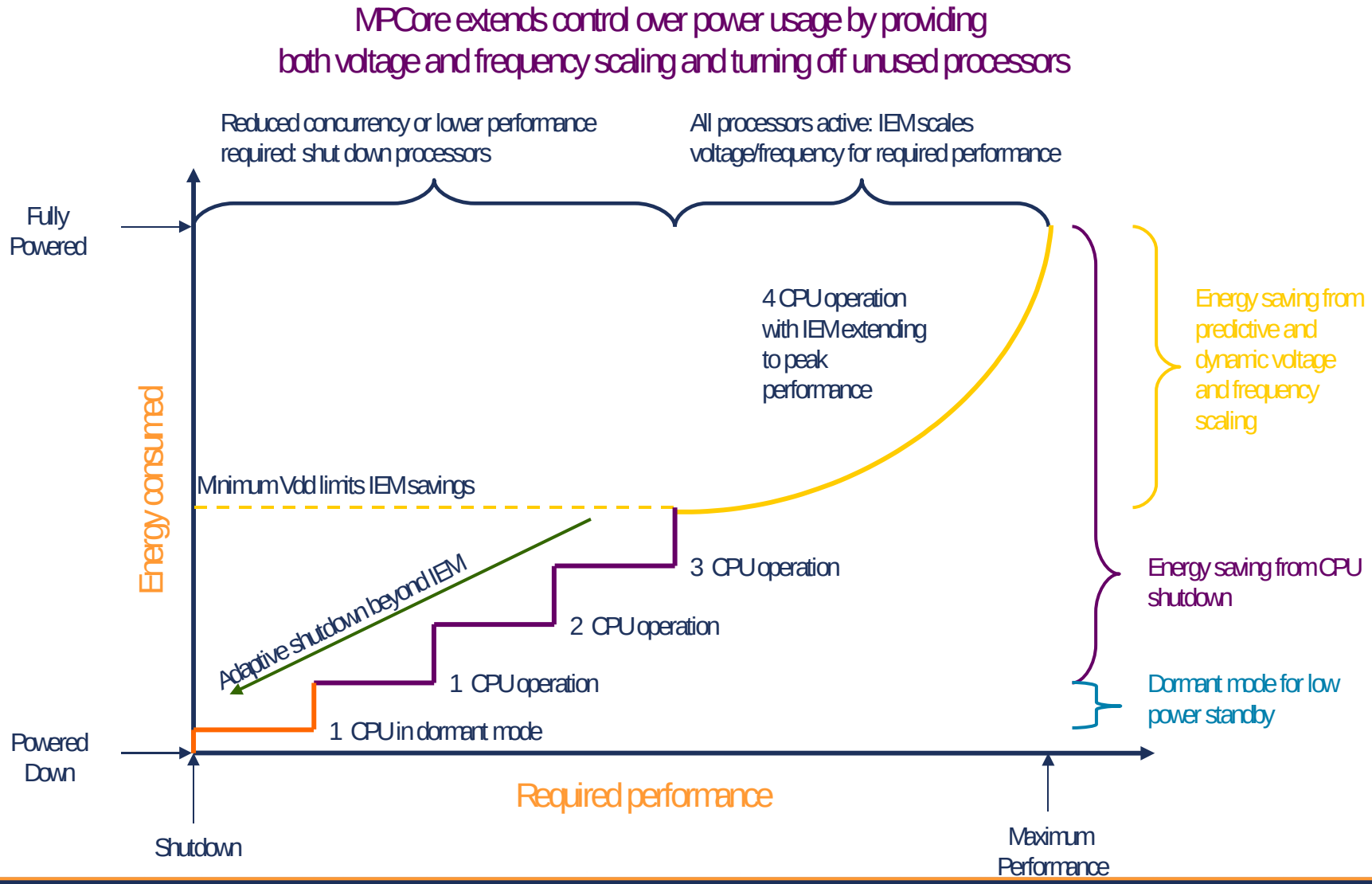
Summary of Per CPU Power Control

	CPU Logic	RAM Arrays	*Wake-up Mechanism*
Running	Powered up Everything clocked	Powered	*N/A*
WFI (also WFE)	Powered up Only wake-up logic clocked	Powered	*Wake up on interrupts (external I/O or Timer / WatchDog)* *L1 memory system only wakes up temporally to process SCU coherency requests*
Dormant	Power off	Retention state voltage only	*Via external wake-up event from power controller*
Powered Off	Power off	Power off	*Via external wake up event from power controller*

Intelligent Energy Management

- Assuming the MPCore implementation included options to
  - Dynamically adjust the (whole multiprocessor's) voltage and frequency
  - Each CPU was isolated so that it could be individually powered down
    - SoC power controller was integrated in the specified manner to implement required power control requests
- Then the expected MPCore power scheme would be
  - If concurrency exists, then run maximum number of CPUs at the lowest MHz and voltage appropriate to accomplish the given work load
    - Map processes to CPU in a manner than best balances utilization
  - If concurrency temporally is less than number of CPU, move to standby
  - If concurrency drops for 'significant' period, then move CPUs into reset
  - If only one CPU is currently powered
    - Go into standby as necessary,
    - If no work for longer periods, move CPU into dormant

MPCore extends beyond simply DVFS

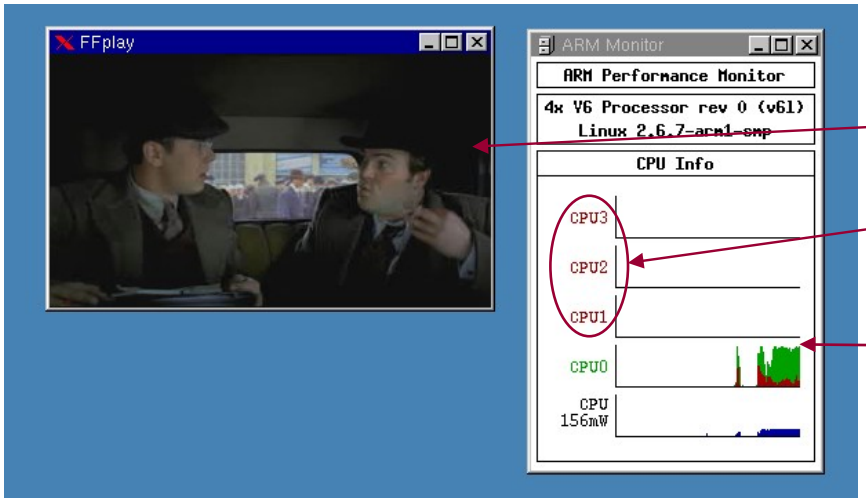


High performance, low power spinlocks

```
static inline void _raw_spin_lock(spinlock_t *lock)
{
    unsigned long tmp;

    asm__ __volatile__(
" 1:     ldrex    %0, [%1]\n"              ; exclusive read lock
"        teq      %0, #0\n"                ; check if free
"        wfene \n"                         ; if not, wait (saves power)
"        strexeq  %0, %2, [%1]\n"          ; attempt to store to the lock
"        teqeq    %0, #0\n"                ; Were we successful ?
"        bne      1b"                      ; no, try again
   :     "=&r" (tmp)
   :     "r" (&lock->lock), "r" (1), "r" (0)
   :     "cc", "memory");

    rmb();          // Read data memory barrier, Stops WO reads << lock write
}                   // This is NOP on MPCore since dependent reads are sync'ed

static inline void _raw_spin_unlock(spinlock_t *lock)
{
    wmb();          // data write memory barrier, Ensure payload write visible
                    // Ensure data ordering, but does not necessarily wait
   _asm__ __volatile__(
"        str      %1, [%0]\n"              ; Release, invalidates any LDREX
"        mcr      p15, 0, %1, c7, c10, 4\n" ; DrainStoreBuffer (flushes to RAM)
"        sev      \n"                      ; Signal to any CPU waiting
   : "r" (&lock->lock), "r" (0)
   : "cc", "memory");
}
```

See ./include/asm-arm/spinlock.h

Demonstration of power save with MP

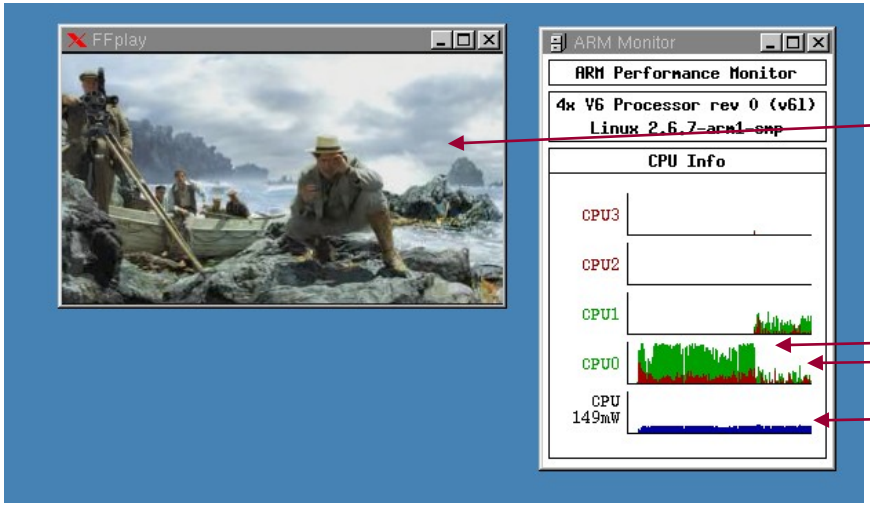


Single-CPU

For a given workload requirement

Unused processor are 'turned off' and isolated from OS (HOTPLUG)

Using a single CPU design point requires in this example 1 CPU @ 260MHz, consuming ~160mW



Dual-CPU (same MHz, same Vt)

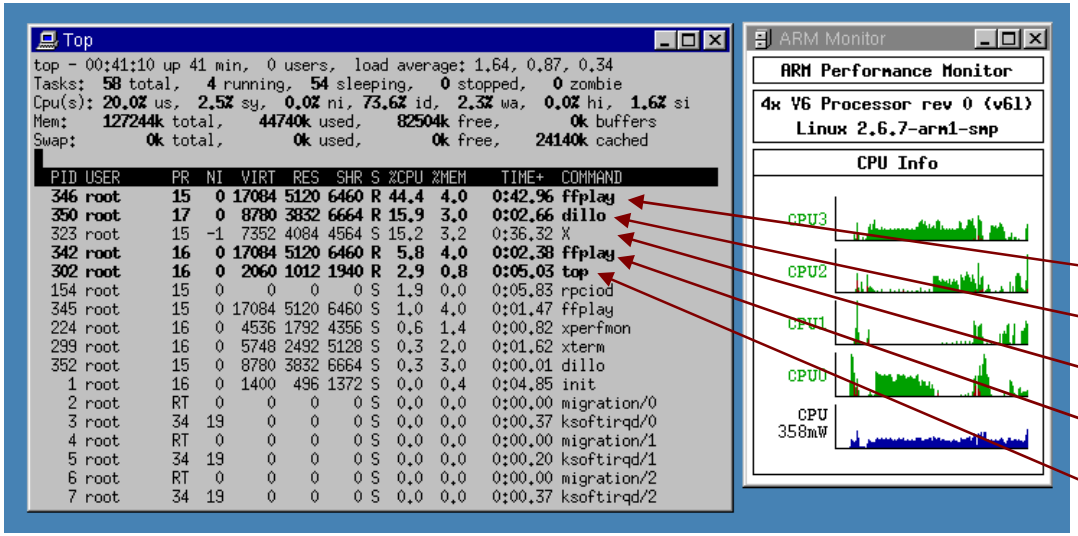
For the same workload level
This is a single threaded application, concurrency is with the operating system

Reduced MHz allow for lower supply voltage which enables more than 50% energy save

Lower power in dual-CPU than single-CPU at same MHz
- Reduction in context switching
- Increase in cache effectiveness

Once you have threaded code, MP offers more performance at lower MHz and without suffering from the cost of memory speed disparity and associated inefficiencies

Realization of concurrency



Inherent within the applications and operating system

- Video Playback
- Browser
- User Interface (X11)
- Audio Playback
- Other user applications

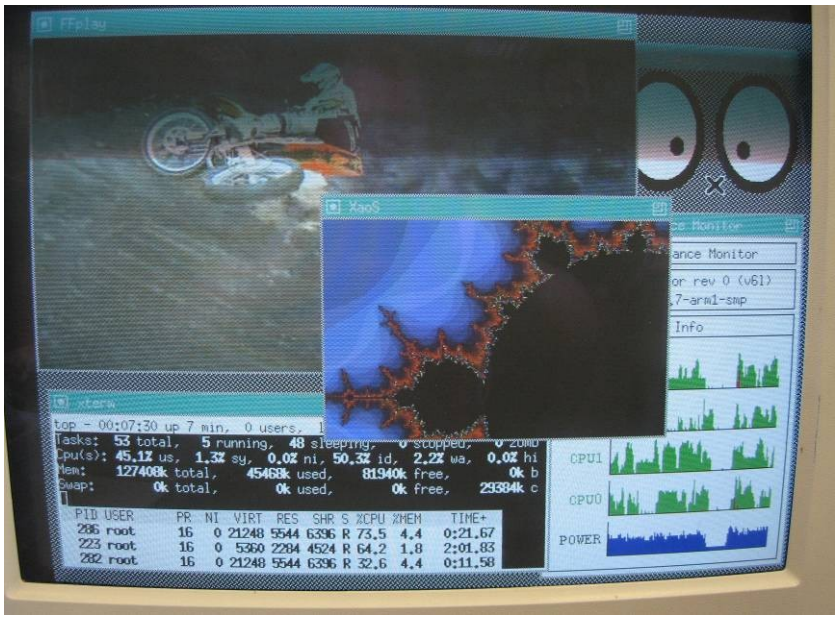
In addition, software developer can thread an application

- Offloading tasks to specialist processors
- Creating a 'pool' of tasks the operating system can share across general SMP processor
- Exposing utility tasks that can be scheduled in the background

Threaded software is already very common even if not widely used in Linux today

- Typical mechanism a OS/RTOS uses to enable developer to express multiple tasks on a (timesliced) single processor
- Software that needs to share a processor itself is complex to write, debug and maintain

ARM11 MPCore – Silicon - right first time



CT11-MPCore Coretile

ARM Integrator/CP Baseboard

Linux 2.6 SMP with standard X11 applications

First test silicon of the 4 way SMP MPCore multiprocessor available on schedule and working to specification

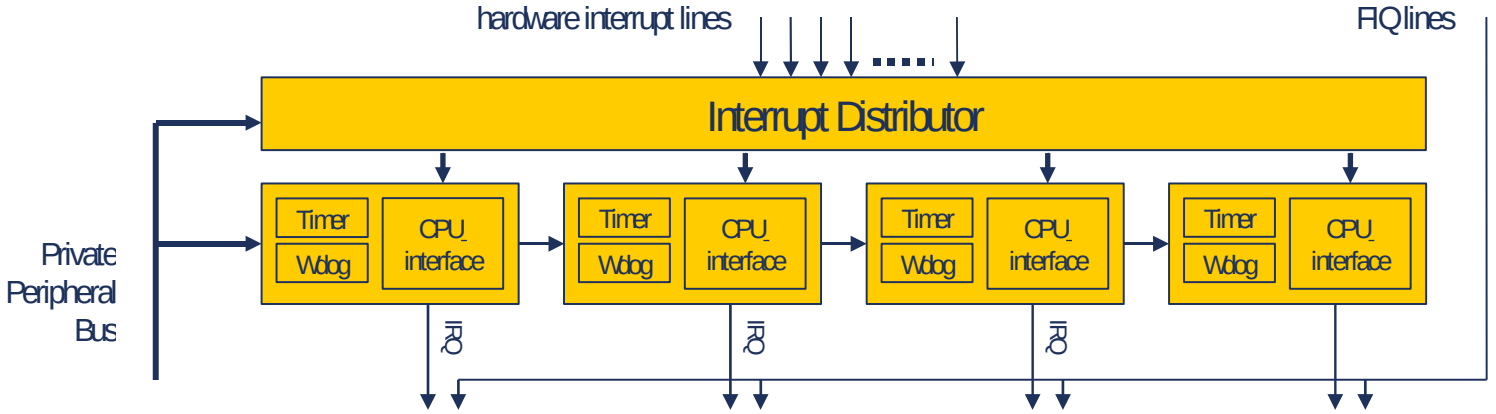
Built for functionality testing but delivering the equivalent of 1.2GHz ARM11 at around 600mW (130nm process)

The highest performance ARM yet!

Demonstrating openly available Linux applications dynamically sharing the CPUs and delivering stunning media performance

Closely coupled communications

- ARM Generic Interrupt Controller, (currently moving to an architectural definition)
  - Software control of priority, routing and masking of interrupts
  - Current Linux implementation maps all IPI through only 1 of the 16 available IPI vectors
- ARM11 MPCore implementation:
  - 16 Levels of hardware prioritization
    - With 'binary point' capabilities to reduce level of pre-emption
  - Configurable between 0 and 255 hardware interrupt inputs
    - 16 software ID per CPU for inter-processor communication
    - Typically combined with shared memory for message passing
- Timer and watchdog interrupts defined per CPU
- Ability to 'Interrupt-broadcast' to, all but self, self, and specific



See ./arch/arm/common/gic.c

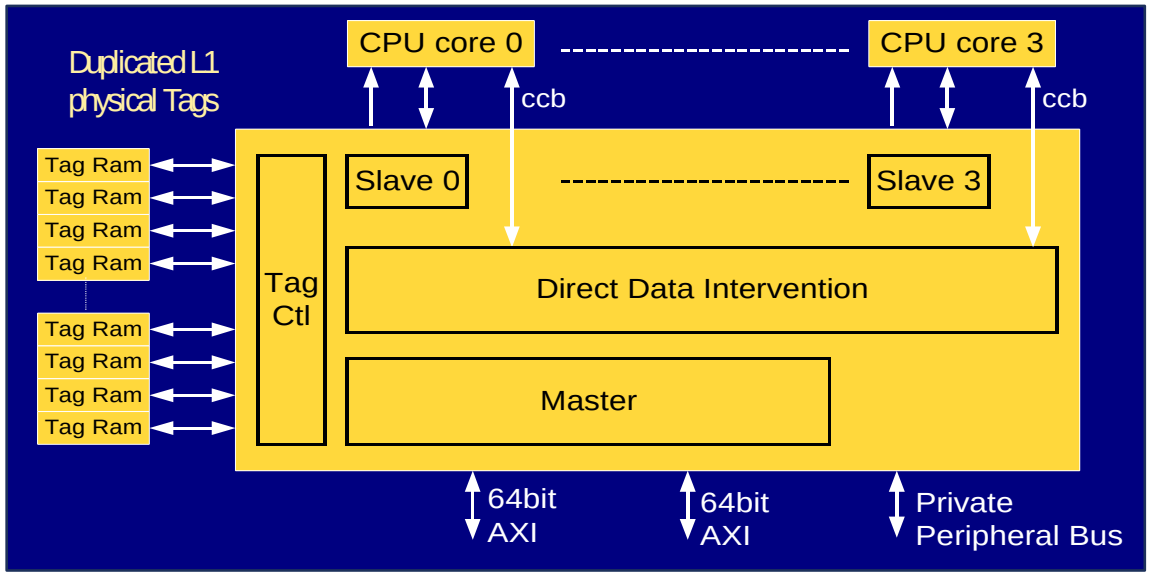
Rapid access to shared data

- The MPCore's SCU was designed to resolves most of the traditional bottlenecks around access to shared data and the scalability limitation introduced by coherence traffic
  - Intelligent monitoring of operations on shared data allows optimized MESI state migration
  - Locally caching global cache state limits snoop interaction between CPU to only CPUs that share data
    - Design limits snoop intrusion to only 4 cycles
  - Direct data intervention permits a local cache miss to resolved in a remote cache
    - Subsequently providing access to shared data 50% faster than the data could be otherwise access from a shared L2 cache
- The historically perceived scalability and performance limitations of SMP are no longer valid
  - Multitasking applications typically scale more than linear to CPU count

ARM MPCore – SCU

Key to fast MP:

- Interface up to 4 multiprocessing CPU with each other and the L2 memory system



- Act as Bus Manager in single CPU case
  - Redundant logic removed via synthesis scripts
- Management of Direct Data Intervention (DDI) traffic
- Management of coherent traffic at CPU core frequency
  - Maintain coherency between coherent L1 data caches,
    - NOTE: not data with instruction, or instruction with instruction
  - Route non coherent data traffic (CPU in AMP mode)
  - Routing of all instruction traffic

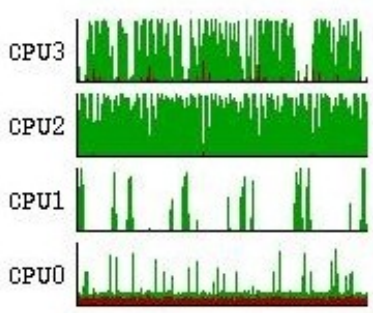
Extracting thread level parallelism

Only required if task needs more performance than a single processor can provide

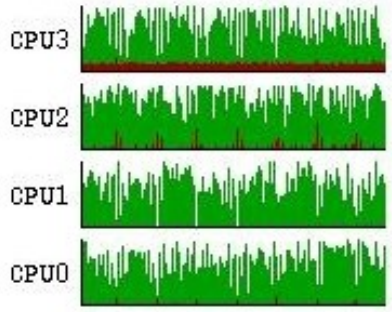
Example: MPEG2 decoder

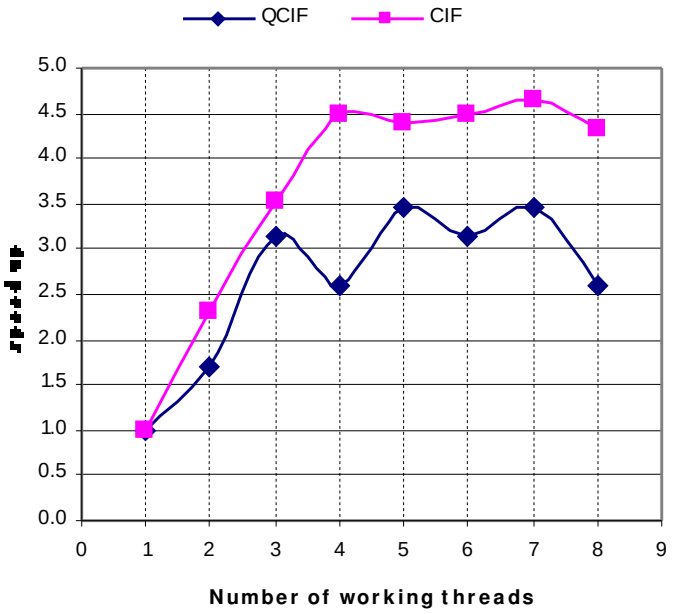
- Sampled from the ARM SMP Evaluation Platform
- Demonstrates utilization of addition processors

2 Threads

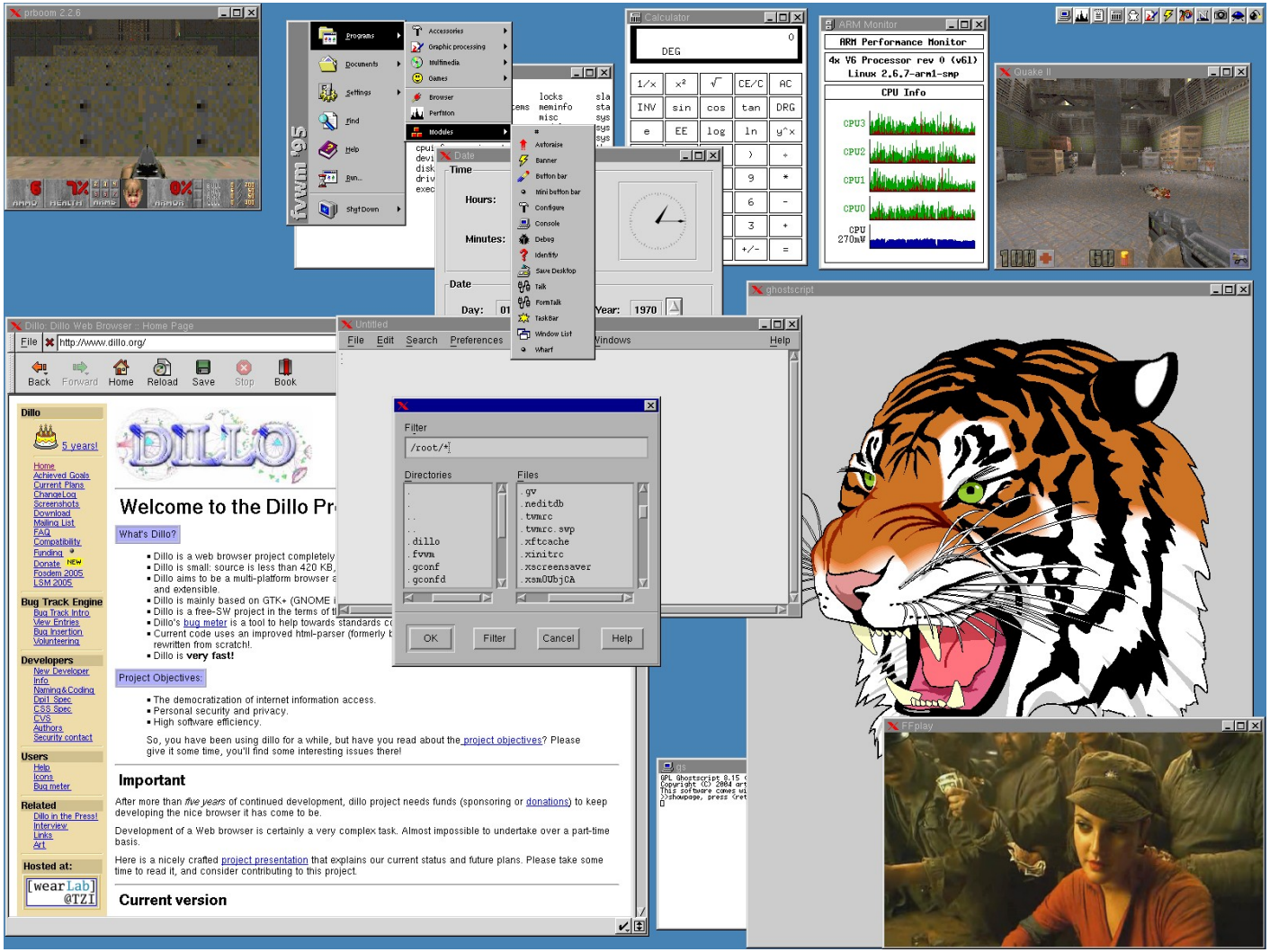


4 Threads





Scalable general purpose processing

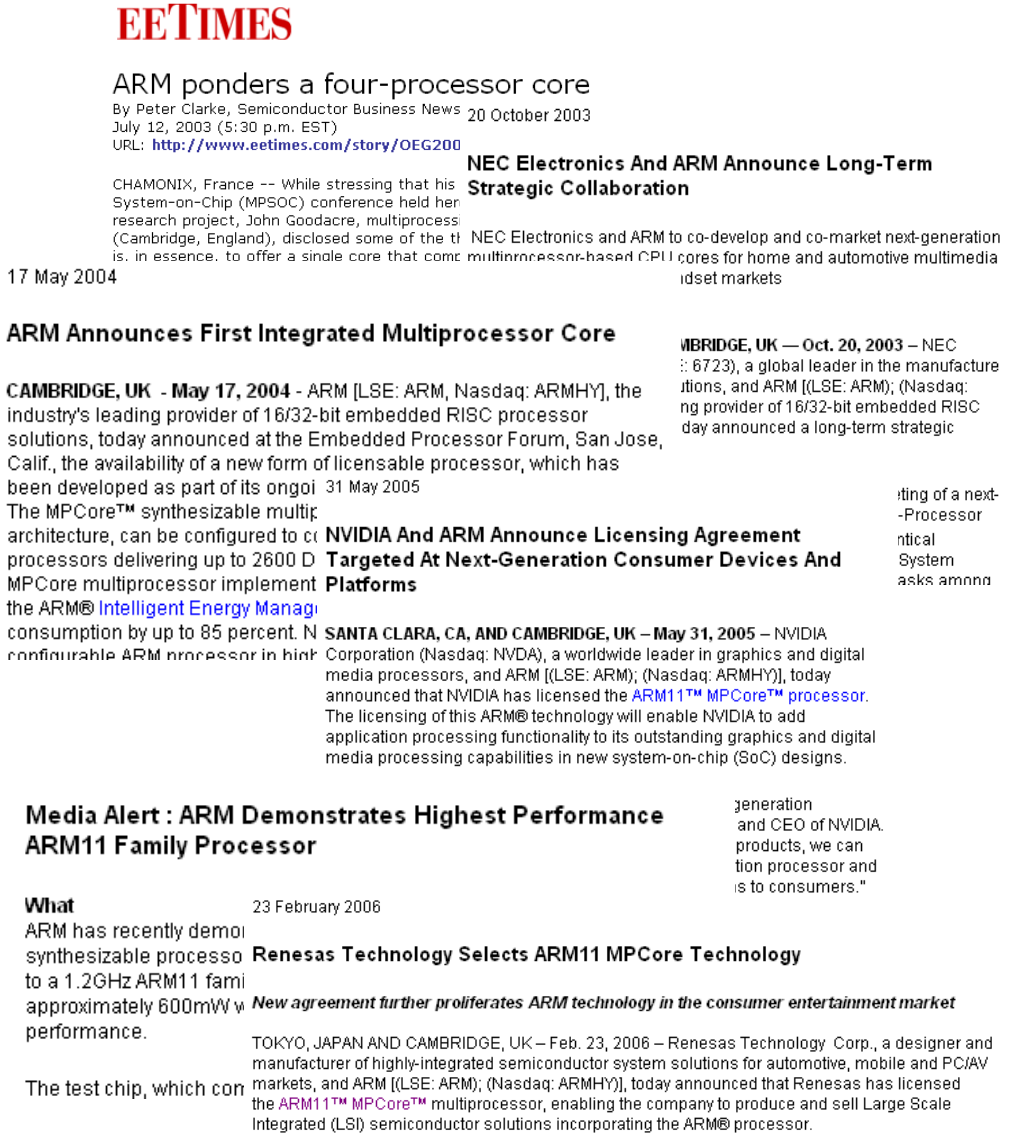


ARM11 MPCore - Linux 2.6 X11 Multimedia Desktop

- No modification of Linux applications
- Noticeably more responsive interface
- Power consumed directly related to CPU activity
- Rich application experience
- Scaleable and low power solution

ARM11 MPCore: Public Adoption

- First public disclosure (July'03)
  - "4 CPU's look interesting"
- NEC in collaboration (Oct'04)
  - Bring SMP capable cores to market
- MPCore announced (May'04)
  - "Desktop performance at handheld power levels"
- NVIDIA selects MPCore (May'05)
  - "To add applications processing"
- Working first silicon (July'05)
  - "Highest performance ARM"
- Renesas select MPCore (Feb'06)
  - "Consumer entertainment



EETIMES

ARM ponders a four-processor core

By Peter Clarke, Semiconductor Business News
July 12, 2003 (5:30 p.m. EST)
URL: http://www.eetimes.com/story/OEG200

CHAMONIX, France -- While stressing that his
System-on-Chip (MPSOC) conference held her
research project, John Goodacre, multiprocess
(Cambridge, England), disclosed some of the t
is, in essence, to offer a single core that comp

20 October 2003

NEC Electronics And ARM Announce Long-Term Strategic Collaboration

NEC Electronics and ARM to co-develop and co-market next-generation multiprocessor-based CPU cores for home and automotive multimedia

17 May 2004

ARM Announces First Integrated Multiprocessor Core

CAMBRIDGE, UK - May 17, 2004 - ARM [LSE: ARM, Nasdaq: ARMHY], the industry's leading provider of 16/32-bit embedded RISC processor solutions, today announced at the Embedded Processor Forum, San Jose, Calif., the availability of a new form of licensable processor, which has been developed as part of its ongoi
The MPCore™ synthesizable multip
architecture, can be configured to c
processors delivering up to 2600 D
MPCore multiprocessor implement
the ARM® Intelligent Energy Manag
consumption by up to 85 percent. N
configurable ARM processor in high

31 May 2005

NVIDIA And ARM Announce Licensing Agreement Targeted At Next-Generation Consumer Devices And Platforms

SANTA CLARA, CA, AND CAMBRIDGE, UK – May 31, 2005 – NVIDIA Corporation (Nasdaq: NVDA), a worldwide leader in graphics and digital media processors, and ARM [(LSE: ARM); (Nasdaq: ARMHY)], today announced that NVIDIA has licensed the ARM11™ MPCore™ processor. The licensing of this ARM® technology will enable NVIDIA to add application processing functionality to its outstanding graphics and digital media processing capabilities in new system-on-chip (SoC) designs.

Media Alert : ARM Demonstrates Highest Performance ARM11 Family Processor

What
ARM has recently demo
synthesizable processo
to a 1.2GHz ARM11 fami
approximately 600mW w
performance.

The test chip, which con

23 February 2006

Renesas Technology Selects ARM11 MPCore Technology

New agreement further proliferates ARM technology in the consumer entertainment market

TOKYO, JAPAN AND CAMBRIDGE, UK – Feb. 23, 2006 – Renesas Technology Corp., a designer and manufacturer of highly-integrated semiconductor system solutions for automotive, mobile and PC/AV markets, and ARM [(LSE: ARM); (Nasdaq: ARMHY)], today announced that Renesas has licensed the ARM11™ MPCore™ multiprocessor, enabling the company to produce and sell Large Scale Integrated (LSI) semiconductor solutions incorporating the ARM® processor.

Take-aways

MPcore is a mature solution rapidly been adopted for the latest high-performance and low power designs

- General availability of testchip development boards
- Kernel, tools and filesystem available

Full GNU tools support

- Current Codesorcery release includes full thread-local-storage support and NPTL for efficient threaded software
- Supporting the high performance ARMv6 instruction set architecture

Full architectural kernel support

- Mainline kernel from 2.6.15 includes all necessary ARM SMP patches for full MPCore support including Adaptive shutdown to standby and reset