Scalable Processing through Software Threading

John Goodacre
Program Manager - Multiprocessing
ARM Ltd

MPSoc'06
6th International Forum on
Application-Specific Multi-Processor SoC

14-18th August 2006

Scaling Performance

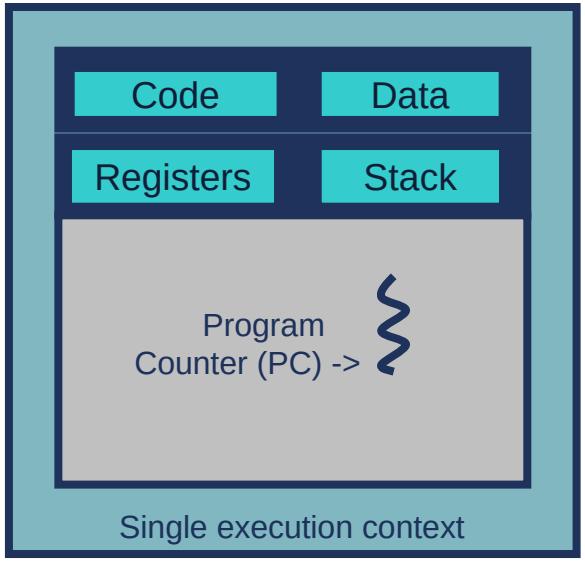
- Many different approaches used in embedded
  - Higher MHz along with more complex processor designs
    - Superscalar, OoO, SMT, finegrain threading, course threading etc
  - Heterogeneous multiprocessors with individual specialized processors doing specialized asymmetric tasks
  - Homogeneous multiprocessors with generalized processors sharing general task workloads (typically symmetrically)
  - Combination and various mixes of the above

- Key challenge: It's really a software problem
  - There are various examples of failed multiprocessing designs due to disparity between the 'ideal' hardware design and what the software programmer could actually use

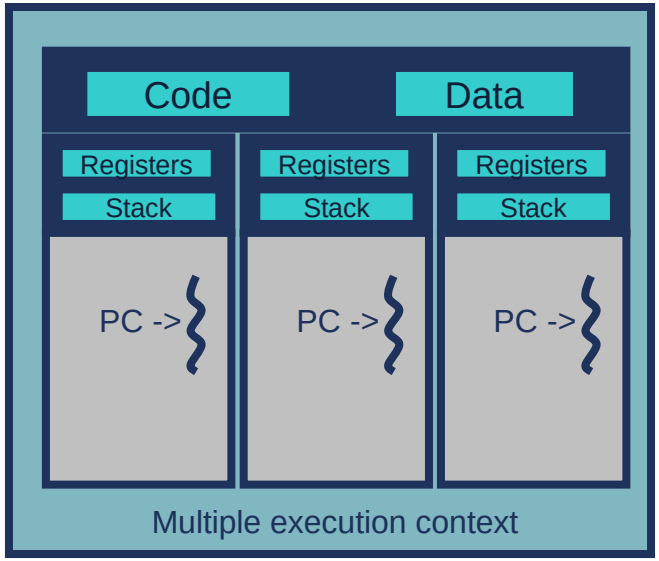
- This mini-keynote looks at the "threaded-software" paradigm

Recap: Meaning of “Threaded-software”

- Threading is an approach used by the software programmer to represent concurrent activities
  - The OS/RTOS can sometimes map independent applications/process to a thread
  - Sometimes a runtime library exposes an API to the programmer
  - …and sometimes the hardware requires the programmer to explicitly partition and assign the thread to specific execution context



Single Threaded Process



Multiple Threaded Process

How do “threads” offer scalability?

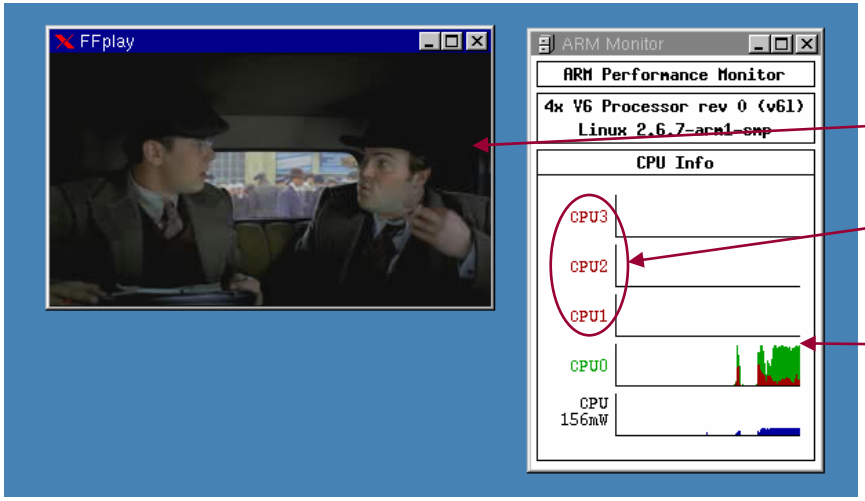
- Multiple threads run concurrently across multiple processors
  - More processor units equates to more performance
  - Can scale beyond what’s realistic of any form of single processor
  - Achieves higher performance in less power and less area
  - Processors may be the same or different architecture / ISA / pipeline etc
    - If the same, OS/RTOS can typically help assign threads to processors
    - If different, the designer/programmer typically needs to work it out!

- Multiple threads sharing the resources of a single processor
  - Demonstrated by various forms of hardware multi-threading
  - Can only scale as far as a well utilized single processor, overall limited by design complexity and power consumption
  - Performance is limited by the theoretical maximum uniprocessor performance
  - Suffers from high power since its still a complex uniprocessor
  - Due to shared resources, programmer needs to work out how threads interact

Potential problems with threading

- Determinism & Predictability
  - Does a thread always do the same thing or will it be influenced by another thread ?
  - Can your software manage determinism through synchronization and explicit control over where and how a thread executes?
  - …or will data dependent access patterns on another thread fundamentally change the predictability of your first thread?
- In many cases can you utilize enough concurrent execution contexts simply with multi-tasking of independent activities
  - But watch out for hardware level contention between threads that will limit realized throughput and require additional software complexity
- The ARM MPCore design allows threads to run independently with unification addressed within the processor macroblock

Measuring multi-core power advantage

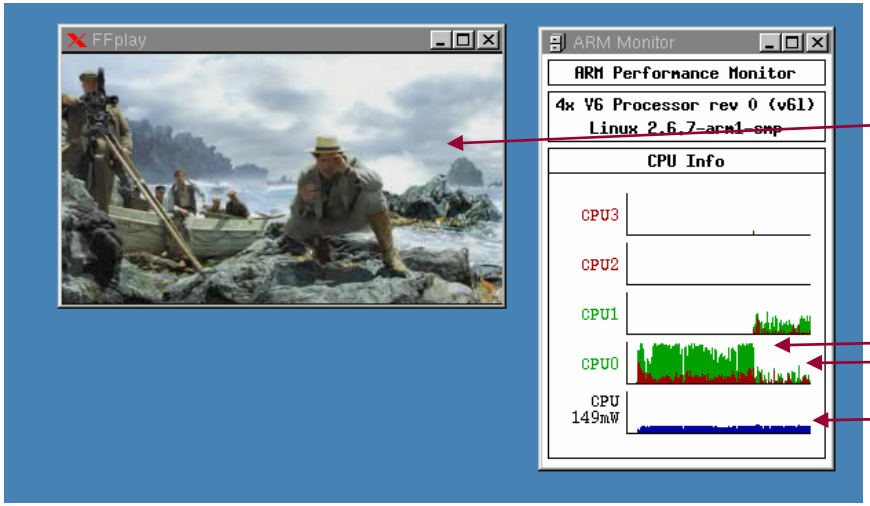


Single-CPU, (ARM MPCore)

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 (using same MHz, same voltage)

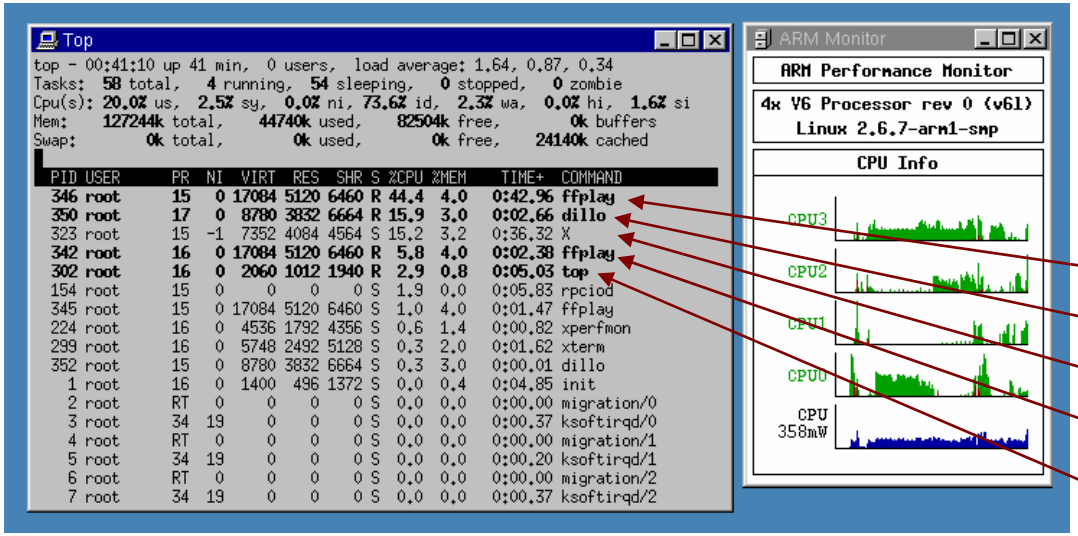
For the same workload level
This is a single threaded application, concurrency is with the operating system.

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

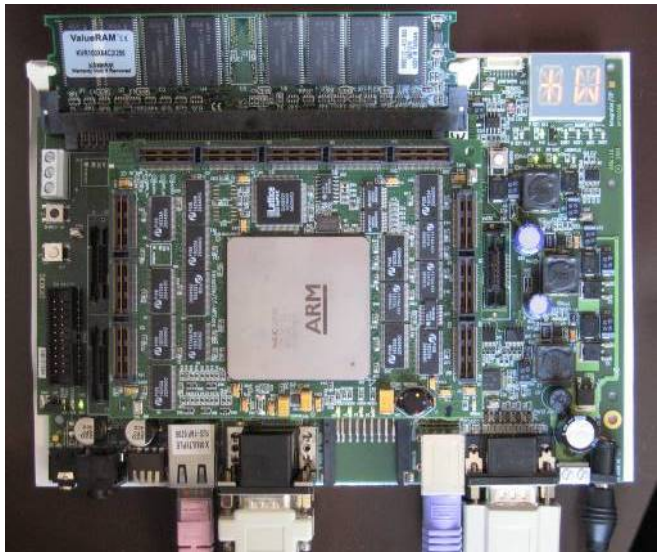
Once you have threaded code, MP offers more performance at lower MHz and without suffering from the cost of memory speed disparity and associated inefficiencies

Realization of concurrency

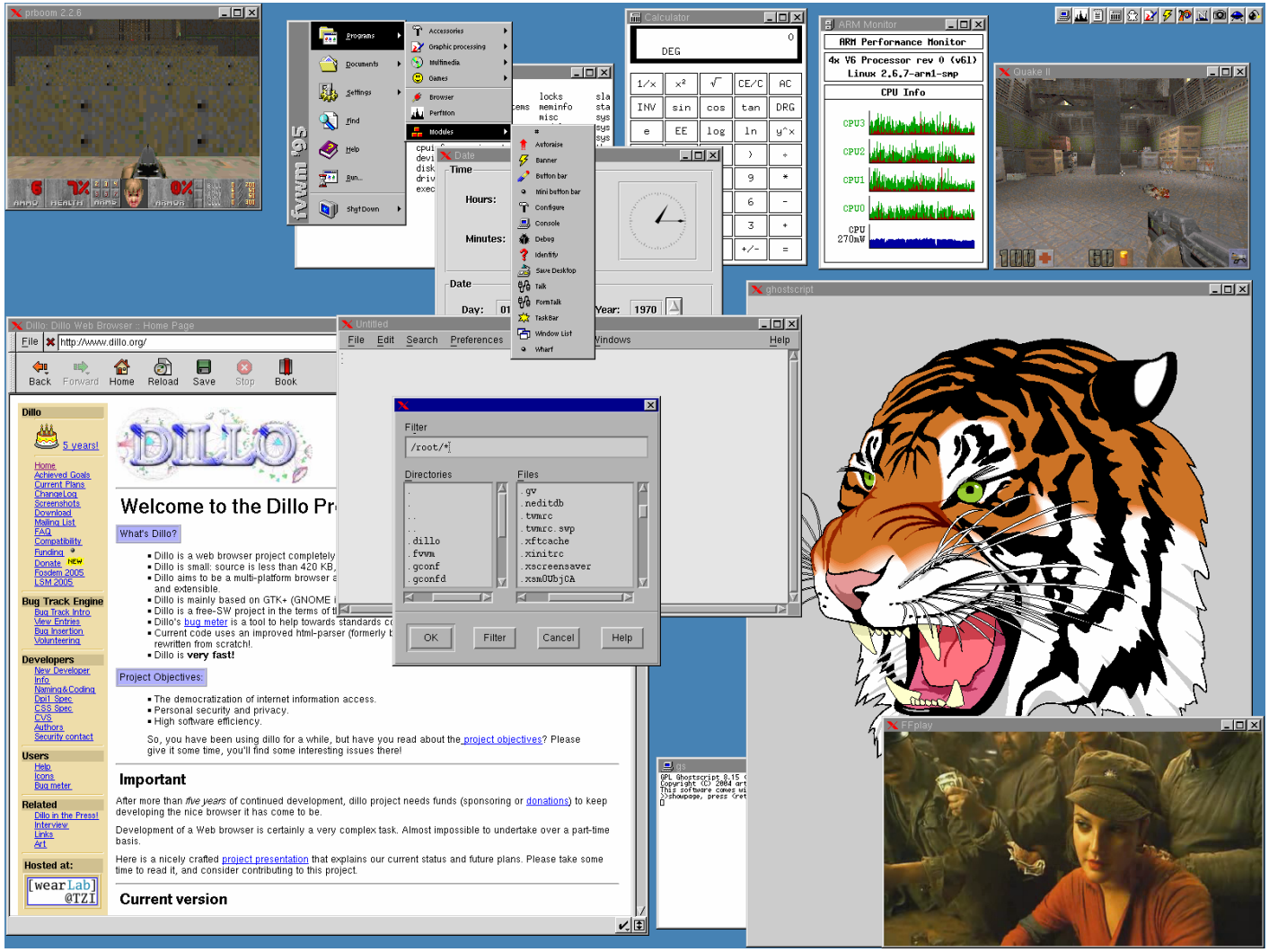


- Inherent within the applications and operating system
  - Video Playback
  - Browser
  - User Interface (X11)
  - Audio Playback
  - Other user applications

- **Evaluation silicon of ARM11 MPCore**
  - Quad-core 32K ID with 1MB L2
  - 264MHz CPU (and 22MHz SoC!)
- **Linux 2.6.15 from kernel.org**
  - X11, Window manager, apps all from Debian without change
- Concurrency inherent in OS and multi-tasking of applications



Threaded software (and multi-tasking)



- 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 - Linux 2.6 X11 Multimedia Desktop

Summary

Threaded software is a fairly natural approach for software programmers to express concurrent execution

There are a lot of different hardware architectures and multi-processor designs claiming they support threaded software

! Designer Beware !

All (software) Threads Are Not Equal

Hardware needs to map as close as possible to the general threaded software paradigm

Specialized cores are often the most power/area efficient solution to a specific task

Independent (multiple) processors provide comparable levels of determinism as single CPU