ARMv7A Architecture Overview

David A Rusling, ARM Fellow

May 2010

Overview

- ARM
  - Some history
  - How we do architecture
- ARMv7A features
  - Thumb 2
  - TrustZone (?)
  - Neon
  - SMP
  - Atomic memory operations

ARM and Architecture

Definition of Architecture

- The Architecture is the **invariant contract** between the Hardware and the Software
  - Confers rights and responsibilities to both the Hardware and the Software
- The architecture distinguishes between:
  - Architected behaviors:
    - Must be obeyed
    - May be just the **limits** of behavior rather than specific behaviors
  - Implementation specific behaviors
    - Allows degrees of variability to fit the implementation requirements
      - Defines the space in which to innovate for performance (or power or area)
    - Certain areas are declared implementation specific. E.g.:
      - Power-down
      - Timing behaviors of the memory map
- Code obeying the architected behaviors is portable across implementations
  - Reliance on implementation specific behaviors gives no such guarantee

How do we “do” architecture

- Team of full time staff manage the architecture
- Formal internal review body – ARB
  - Responsible for sign-off of changes to the architecture
  - Represent all interested parties – software, tools, etc
- Development of new architecture
  - R&D – scope, shape, viability, initial specification
  - APD – ‘bake’ ready for the implementation teams
  - Implementation – go build and deliver it (and ‘it’ includes tools, OS ports, middleware etc)
- Internal and External consultative bodies
  - AAG (internal) and TAB (external)
  - Ad-hoc communications

Evolution of the ARM Architecture

- Original ARM architecture:
  - 32-bit RISC architecture
  - 16 Registers - 1 being the Program counter – generally accessible
  - Conditional execution on all instructions
  - Load/Store Multiple operations - Good for Code Density
  - Shifts available on data processing and address generation
  - Original architecture had 26-bit address space
    - Augmented by a 32-bit address space early in the evolution
- Thumb instruction set was the next big step
  - ARMv4T architecture (ARM7TDMI)
  - Introduced a 16-bit instruction set alongside the 32-bit instruction set
  - Different execution states for different instruction sets
  - Switching ISA as part of a branch or exception
  - Not a full instruction set – ARM still essential

Evolution of the Architecture (2)

- ARMv5TEJ (ARM926EJ-S) introduced:
  - Better interworking between ARM and Thumb
    - Bottom bit of the address used to determine the ISA
  - DSP-focussed additional instructions
  - Jazelle-DBX for Java byte code interpretation in hardware

- ARMv6 (ARM1136JF-S) introduced:
  - Media processing – SIMD within the integer datapath
  - Enhanced exception handling
  - Overhaul of the memory system architecture

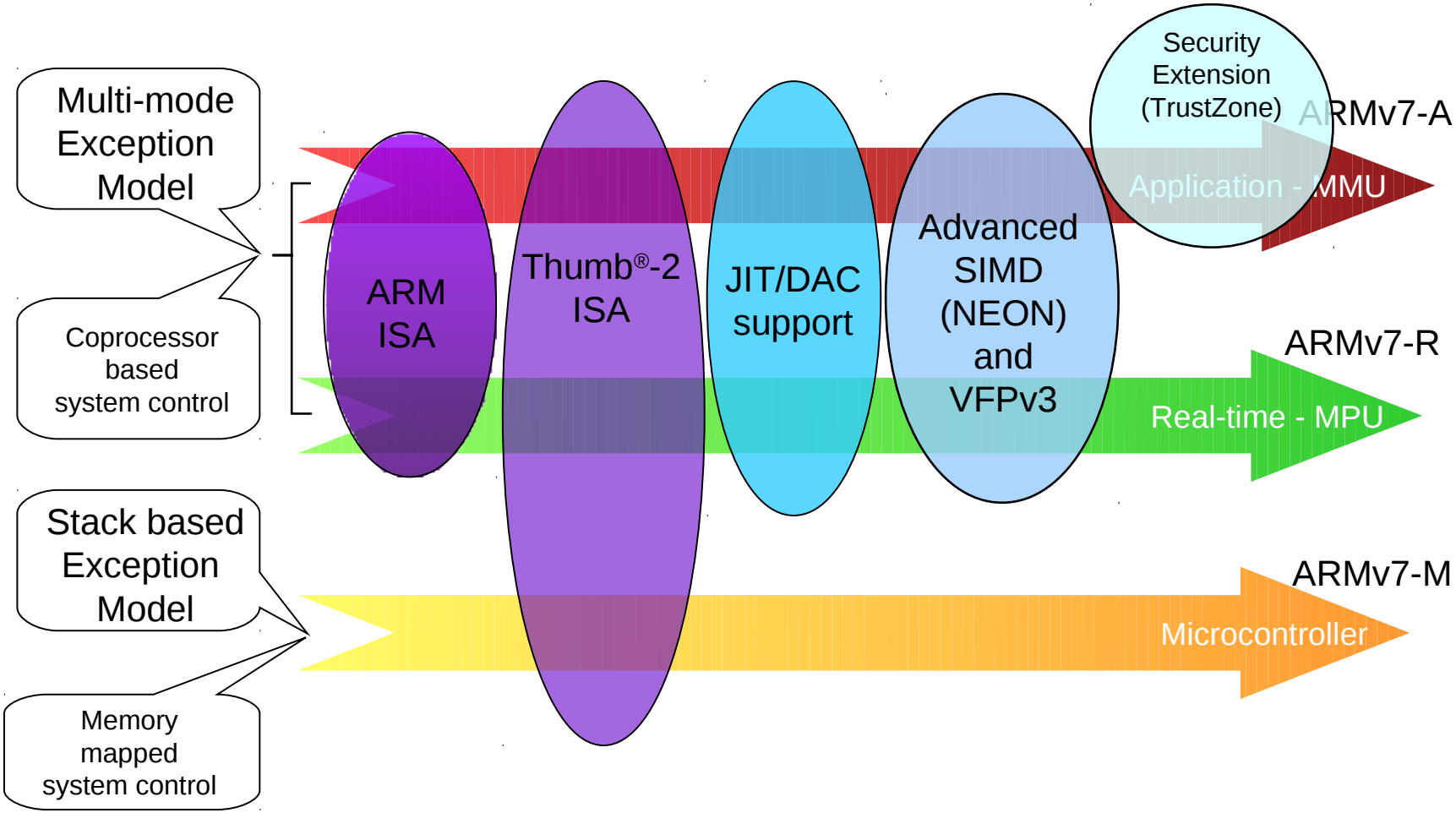
- ARMv7 rolls in a number of substantive changes:
  - Thumb-2*
  - TrustZone*
  - Jazelle-RCT
  - Neon
  - ARMv7 is split into 3 profiles

  * - Introduced initially as extensions to ARMv6

ARM Profiles

- The Application “A” profile
  - Memory management support (MMU)
  - Highest performance at low power
    - Influenced by multi-tasking OS system requirements
- The Real-time “R” profile
  - Protected memory (MPU)
  - Low latency and predictability ‘real-time’ needs
  - Evolutionary path for traditional embedded business
- The Microcontroller “M” profile
  - Lowest gate count entry point
  - Deterministic behaviour a key priority
  - Deeply embedded – strong synergies with the “R” profile

ARMv7: profiles & key features

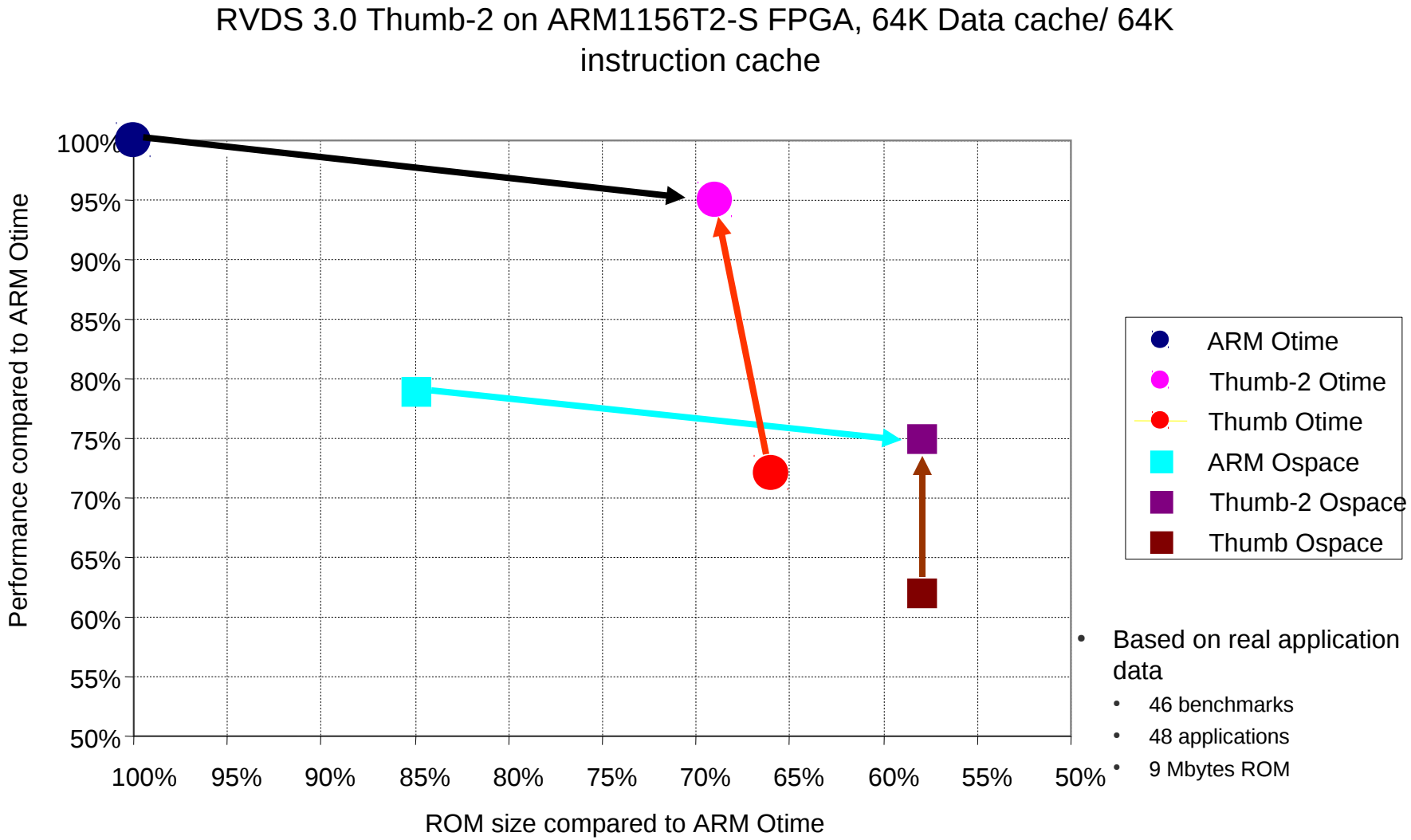


ARMv7A Features

Thumb-2: Smaller and Faster

- New instructions mean that:
  - Thumb-2 can be smaller than Thumb
    - CZB can replace CMP + B[EQ|NE] to reduce code size, i.e. one new 16-bit instruction replaces two 16-bit instructions
    - TBB and IT can replace complex branching structures
  - Thumb-2 can be faster than ARM
    - Single instruction replaces two instructions
      - ORN
      - SUB, ADD, LDRH, STRH: immediate ranges changed
      - LDRD, STRD: register restriction removed
  - Smaller and faster artefacts too:
    - Smaller code results in more efficient use of I-cache

RVDS 3.0 Thumb-2 on ARM1156T2-S



- Based on real application data
  - 46 benchmarks
  - 48 applications
  - 9 Mbytes ROM

ARMv7: Software Compatibility

- Backward software compatibility is key for customers with existing software that runs on earlier ARM cores
- Thumb-2 is a superset of Thumb
  - A single instruction set architecture (ISA)
  - ARM ⇔ Thumb interworking remains the same
- ARMv7A supports ARM and Thumb-2
  - Thumb-2 offers a new freedom on the ARM/Thumb boundary
  - Software can be re-used, retargeted to Thumb-2, or a mixture

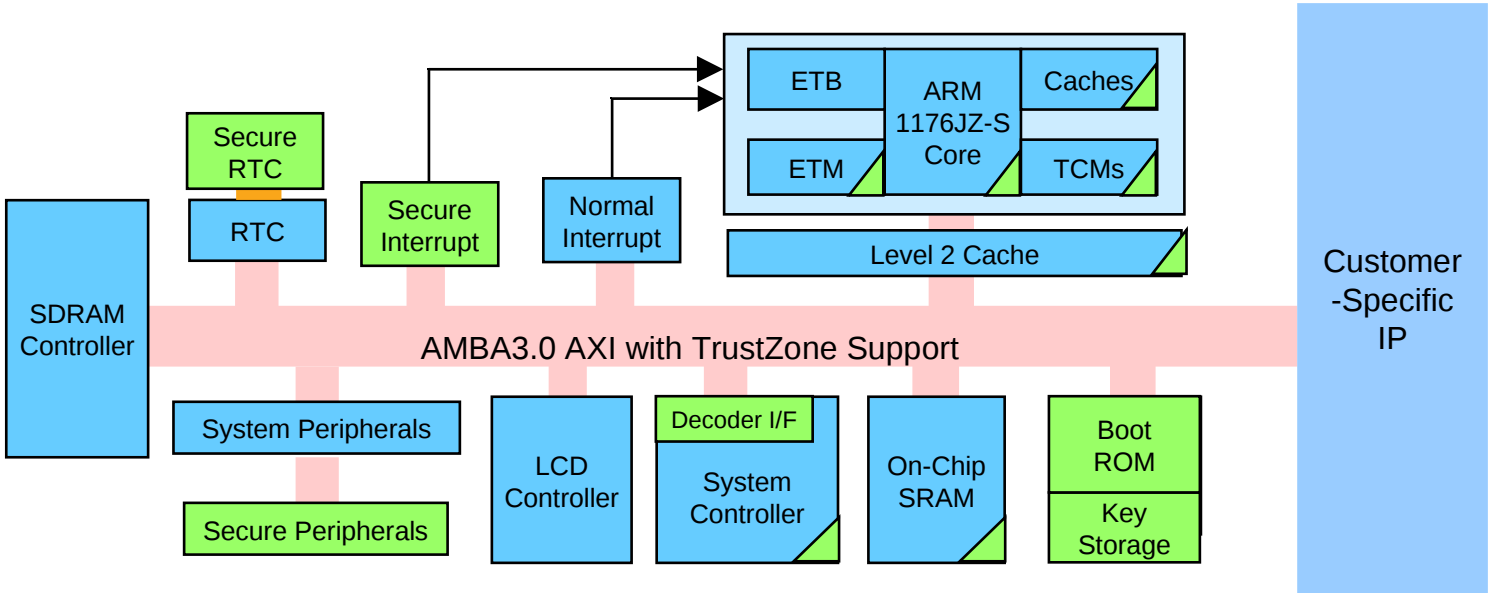
ARMv7-A key features

- Uses the *traditional* ARM programmers'/exception model
  - Supports Monitor mode/the Security Extension
  - Hivecs configuration option for exception entry
- Virtual Memory System Architecture
  - Supports shared and local memory
  - Normal, Device and Strongly ordered memory types
  - Configurable cache policy with hierarchical cache operations (no broadcast operations for MP – coherency maintained with software)
  - Pagetables in memory – entries cached in TLBs
- Enhanced instruction set support over ARMv6
  - Updates to barrier and NOP hint instruction support
  - ThumbEE state execution
  - Advanced SIMD (NEON) option
  - VFPv3 option (VFPv2 in earlier architecture variants)
- Debug model refinement

ARMv7-A Security Extension

- Physical protection
  - Physical address space divided between Secure and Non-secure *'worlds'*
  - Memory system support for caches and translation buffers
    - Tagged resources
    - Secure and Non-secure virtual memory managed separately
- *Monitor* mode supports the Secure ⇔ Non-Secure transition
  - Boot time configuration
  - Transparent for legacy code
  - Dynamic reconfiguration of Secure/Non-secure resource allocation supported by the architecture
  - Interrupt model support

Security: TrustZone System



Tagged Non-Secure

Tagged Secure

Secure-aware

- Normal peripherals can be accessed by all bus masters
- Secure peripherals can only be accessed by secure-aware masters (i.e. are “Trusted”)
- Secure-aware slaves have areas that can only be accessed by trusted masters

ARMv7-AR: VFPv3 – Floating Point support

- Builds on VFPv2
  - Double precision register count increased from 16 to 32
  - Fixed ⇔ Float conversion instructions
    - Signed and unsigned conversions
    - Integer value: 16- or 32-bit
    - FP value: single or double
  - Floating point constant loads
- User traps now an architecture option (VFPv3U)
  - A change of emphasis from VFPv2
  - Fits in with Advanced SIMD exception-free execution

  Trap free operation provides best performance
  & best suited to the majority of ARM target markets

ARMv7 Advanced SIMD: NEON Technology

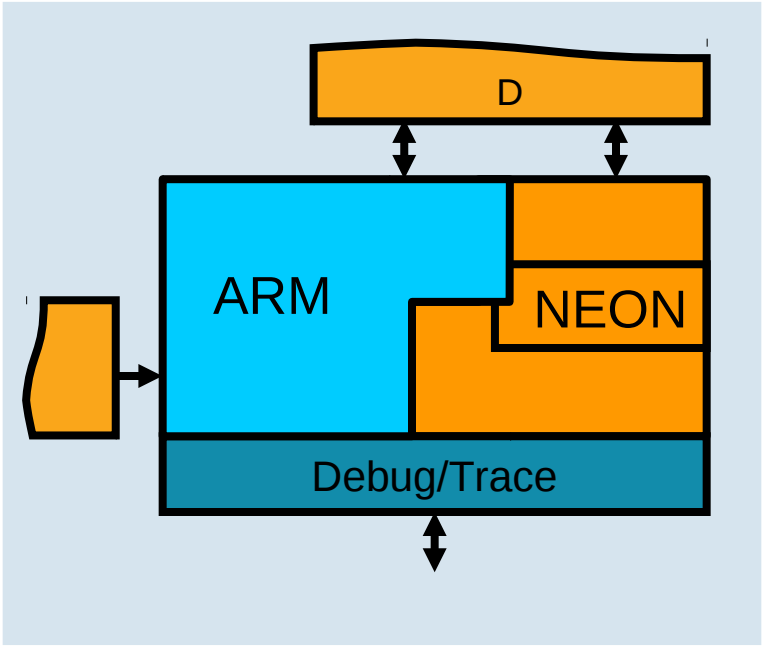
- Designed to accelerate multimedia and DSP applications
  - Tightly integrated, separate execution hardware

- ARMv6 SIMD instructions an expansion of the ARM and Thumb-2 instruction sets
  - NEON = complete architecture with its own register file and pipeline

- Packed SIMD processing
- OpenMAX: industry initiative for media APIs and associated software support

Programmer's Model

- Single instruction stream
- Single view of memory
- Single debug and trace

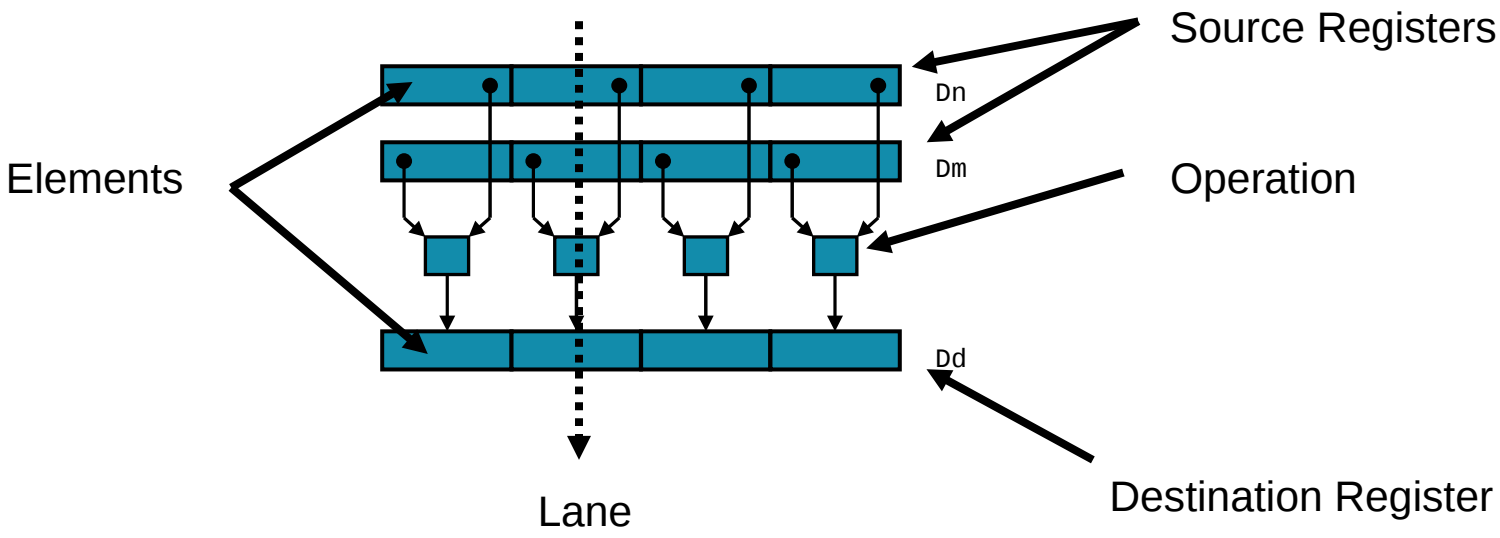


- **ARM** handles control plane
  - Loops, branches and function calls
  - Address calculation
  - Hardware optimised for tight control

- **NEON** handles data plane
  - Integer, Fixed-point and Floating-point processing
  - Hardware optimised for high throughput

Neon SIMD (Single Instruction Multiple Data)

- NEON Instructions are based on “Packed SIMD” processing
  - Registers are considered as vectors of elements of the same data type
  - Instructions perform the same operation in all lanes



- NEON adheres very strictly to this model
  - Avoids use of “ad-hoc” SIMD instructions
  - Enables consistent techniques for mapping algorithms to NEON

ARMv7A: Debug

- CP14 support introduced in ARMv6

or

- Memory mapped support
  - More flexible for multiprocessor systems and system level debug

- Architected Debug Access Port (DAP)
  - Consistent debug programmers model from host for ARMv7
  - ‘Coresight’ or CP14 access mechanisms hidden from the debugger

- Supports
  - Halting debug-mode
  - Monitor debug-mode
  - Trace: instruction and/or data

Multi-processing

- To some extent, ARM has always been in multi-processing
  - Common ARM systems typically have a DSP alongside the ARM core
  - Communications flows are predictable and manageable in software
    - No special hardware for coherency in most ARM systems
- ARM introduced the ARM11MPCore in 2004
    - Between 1 and 4 processors in a coherent on-chip cluster
      - Optimised “Snoop Control Unit” for coherency traffic
    - Silicon available from ARM partners today
- ARM A9 SMP by design, as all future A class cores will be

Atomic Memory Operations [1]

- Before ARMv6, SWP instruction was used
  - Gave us uninterruptible read and write operations
  - Complex memory hierarchies and long memory latencies mean that SWP causes performance bottlenecks
- ARMv6k (and ARMv7) added load and store exclusives (LDREX, STREX)
  - Principle of a memory monitor (address range varies by implementation)
  - Tags the location in memory with the identity of the agent trying to modify it
  - Exclusive load tags the memory, following exclusive store will fail if the memory has been accessed by another agent

Atomic Memory Operations [2]

- SWP instruction not guaranteed to work in an SMP implementation
- ARMv7 allows for the SWP instruction to be disabled
- Means that SWP use can be trapped and safely emulated in the SMP SWP case
  - Slow versus data corruption / correctness

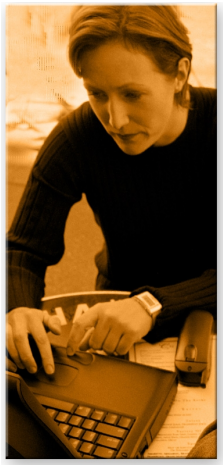
Backup Slides

ARMv7 Execution Environment Support (Jazelle RCT)

- Derived from Thumb-2: Thumb-2EE architectural name
  - Support mandated in ARMv7-A
- Supports JIT/DAC and AOT techniques:
  - JIT = ‘Just in Time’ compiler (real-time)
  - DAC = ‘Dynamic Adaptive Compilation’ (real-time)
  - AOT = ‘Ahead of Time’ compiler (install or on download)
- Provide a new instruction set that allows for
  - JIT compiled code size within 10% of original bytecode
  - Equivalent performance to existing ARM or Thumb-2 instruction sets
  - Simple and low cost to implement in hardware
- Improved memory efficiency
  - ROM, RAM and cache
  - Benefits apply to down-loadable applications in RAM as well as those pre-installed in ROM
    - ROM: Compile for minimal code size performance gains from AOT
    - RAM: Compile for performance at runtime, by in-lining more frequently

ARM technology lies at the heart of advanced digital products

EMBEDDED SOLUTIONS



ENTERPRISE SOLUTIONS

HOME SOLUTIONS

MOBILE SOLUTIONS

EMERGING APPLICATIONS