\textbf{ARM Processors for Embedded Applications}

- Roadmap for ARM Processors
- ARM Architecture Basics
- ARM Families
- AMBA Architecture

\section*{Current ARM Core Families}

- ARM7:
  - Hard cores and Soft cores
  - Cache with MPU or MMU
  - Real-time debug (RTD) and real-time trace (RTT)
- ARM9/ARM9E-S: (E: DSP-Enhanced; S: Synthesizable)
  - Hard cores and Soft cores
  - Cache with MPU or MMU
  - DSP
  - RTD and RTT
- ARM10E:
  - Vector floating point (VFP) coprocessor
  - RTD and RTT
  - All cores incorporate I- and D-cache
- StrongARM
  - Cache with MMU
- XScale
Evolution of the ARM Architecture

Development Benefit for ARM

- Core designed for embedded applications:
  - High-performance, low power consumption
  - Small die size, tight code density
  - Software debug capabilities
- Compatible instruction set architecture
- Standard bus architecture (AMBA) for design reuse
- ARM and third-party tool chains and ASIC tools
- Multiple RTOS solutions
- Multiple silicon sources
Example for using ARM Processor

ARM Product Roadmap

- 6 families: 100 MIPS to 1000 MIPS
ARM Product Roadmap (Cont.)

• Roadmap for Open Platform Applications

- Dhrystone MIPS
- ARM 1020E architecture
- DSP extensions

- ARM920T
- Harvard cache
- Cache sizes options

ARM1020E
ARM922E
ARM10xxEJ
ARM926EJ
ARM922T
ARM720T
- ETM7 support

ARM Product Roadmap (Cont.)

• Roadmap for Embedded Applications

- Dhrystone MIPS
- ARM 1020E architecture
- DSP extensions

ARM740T
- 8 KB unified cache
- Simple memory protection

ARM940T
ARM946E
ARM966E
ARM1020E
ARM1022E
ARM720T
 ARM920T
- Floating point for ARM 10 family

VFP9
- Floating point for ARM 9E family

ARM968E
- DSP extensions
- Fast interrupt response
- Level-1 RAM
- Variable level-1 RAM cache sizes
– **ARM Processors for Embedded Applications**
  - Roadmap for ARM Processors
  - ARM Architecture Basics
  - ARM Families
  - AMBA Architecture

**ARM Architecture Basics**

- ARM is a 32-bit load/store and RISC architecture
- 37 total registers for 7 processor modes:
  - 20 visible 32-bit registers in privileged modes (17 in user mode)
    - r0-r13 = general purpose registers (r13 = stack pointer (SP))
    - r14 = link register (LR)
    - r15 = program counter (PC)
    - SPSR = saved program status register (only accessible in privileged modes)
    - CPSR = current program status register
      - The first four bits of the Current Program Status Register (CPSR) is set automatically during every arithmetic, logic, or shifting operation
Register Organization

- **User mode**: the only non-privileged mode
  - It has the least number of total registers visible
  - It has no SPSR and limited access to the CPSR
- **FIQ and IRQ modes**: Two interrupt modes of the CPU
  - FIQ mode has an additional five banked registers to provide more flexibility and higher performance when handling critical interrupts
- **Supervisor mode**: The default mode of the processor on start up or reset
- **Undefined mode**: Traps unknown or illegal instructions when they are passed through the pipeline
- **Abort mode**: Traps illegal memory accesses as a result of fetching instructions or accessing data
- **System mode**: Uses the user mode bank of registers
  - Provide an additional privileged mode when dealing with nested interrupts
- **Thumb state**:
  - Register banks are split into low and high register domains
  - The majority of instructions in Thumb state have a 3-bit register specifier
    - These instructions can only access the low registers in Thumb (R0 through R7)
    - The high registers (R8 through R15) have more restricted use

Note: System mode uses the User mode register set
**TDMI**

- **Thumb:**
  - 16-bit subset of the 32-bit ARM instruction set
- **Debug:**
  - Additional core signals for debug use
  - On-chip debugging facilities
- **Multiplier:**
  - Supports for an enhanced multiplier for 64-bit results
- **EmbeddedICE™ Logic:**
  - Logic/registers to control debug facilities
  - On-chip break point and watchpoint supports

---

**Thumb 16-bit Instructions**

- Reasons of using Thumb instructions:
  - Reduce memory cost leads to smaller code size and the use of narrower memory
  - Performance loss due to using a narrow memory path
    - E.g. a 16-bit memory path with a 32-bit processor: the processor must take two memory access cycles to fetch an instruction or read and write data
  - Processor need to provide a better code density for embedded processor
- Compressed subset of the 32-bit ARM instruction set:
  - Suitable for lower bus bandwidth from narrow external memory
  - Improves code density
- A Thumb enabled ARM:
  - Executes both 32-bit ARM and 16-bit Thumb instructions
  - Allows runtime inter-working between ARM and Thumb code
  - State change performed via branch with exchange (BX) instruction
    - Application must boot up with ARM and switch to Thumb if required
  - Only the instructions are 16-bit
**Thumb Benefit: An Example**

**Thumb Features**

- The Thumb instruction set was created based upon three different criteria:
  - Instructions that only require 16-bit space to be defined
  - Instructions that were needed by the compiler
  - Instructions that were most commonly used in real world applications
- Thumb programs typically are:
  - ~30% smaller than ARM programs
  - ~30% faster when accessing 16-bit memory
- Thumb reduces 32-bit system to 16-bit cost:
  - Consumes less power
  - Requires less external memory
**Debug Extensions**

- **Goal:** To provide a means to debug a final product without adding extra dependent pins to the design
  - **The Debug extension adds:**
    - Scan chains around the core to monitor what is occurring on the data path of the CPU
    - Signals were added to the core so that processor control can be handed to the debugger when a breakpoint or watchpoint has been reached
    - Enabling user to view characteristics such as register contents, memory regions, and processor status
    - A Test Access Port (TAP) controller to allow access to the scan chains
  - **The EmbeddedICE Logic adds:**
    - A set of registers providing the ability to set hardware breakpoints or watchpoints on code or data
    - Monitors the ARM core signals every cycle to check if a breakpoint or watchpoint has been hit
    - An additional scan chain is used to establish contact between the user and the EmbeddedICE logic

**EmbeddedICE Logic**

- The advantage of on-chip debug can rapidly debug software (especially in ROM)

[Diagram of ARM7TDMI with scan chains and debug logic]
– **ARM Processors for Embedded Applications**
  - Roadmap for ARM Processors
  - ARM Architecture Basics
  - ARM Families
  - AMBA Architecture

---

**The ARM7TDMI Core**

- Architecture version 4T:
  - 3-stage pipeline
  - Unified cache
  - 32-bit ARM plus 16-bit Thumb extension
- Forward compatible code (compatible with ARM9 and ARM10)
- EmbeddedICE on-chip debug
- Hard Macrocell IP
  - Smallest Die Size: 0.53 mm² on 0.18 µm process (around 70,000 transistors)
- Up to 110 MHz on TSMC standard 0.18 µm
- Low power consumption: 0.25 mW/MHz
**ARM720T**

- Cached Macrocell for Platform OS Applications
- ARM7TDMI core:
  - ARM v4T
  - Thumb 16-bit instruction set
- 8 KB cache (data and instruction caches)
- AMBA Advanced System Bus (ASB) interface
  - Can be converted directly from an ASB to the higher performance AHB interface
- Memory Management Unit:
  - Full support for WindowsCE and Symbian OS, Linux, or PalmOS
  - Offers virtual-to-physical address translation
  - 64-entry Translation Lookaside Buffer (TLB)
  - Two-level page tables stored in memory
- Hard Macrocell IP
  - 2.9 mm² on 0.18 mm process
  - 75 MHz on TSMC standard 0.18
- Power: 0.55 mW/Mhz

---

**Strength of ARM7 vs ARM9**

<table>
<thead>
<tr>
<th>ARM7TDMI Family Applications 0 - 70 MIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cellular</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ARM9TDMI Family Applications 60 - 200MIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital STB and NC's</td>
</tr>
</tbody>
</table>
Pipeline Comparison (Cont.)

- **ARM9 TDMI:**
  - Decode stage: Allows direct decoding of Thumb instructions to remove the need to translate and decode as with the ARM7
  - Operations previously performed in the **execute** stage of ARM7 are spread across four stages in the ARM9 pipeline: **decode**, **execute**, **memory**, and **write**
    - The reorganization and removal of these critical paths resulted in a much higher clock frequency
    - Memory: an additional cycle for data memory access
      - An interruption of the pipeline is required for ARM7 since it only has one single memory port (for both data and instruction)
- **ARM10 TDMI (6 Stages pipeline):**
  - **Issue**: between the **fetch** and **decode** stages to allow more time for instruction decode
    - Helpful when an unpredicted branch is executed
ARM9TDMI Processor Core

- ARM 32-bit and Thumb 16-bit instructions
- Harvard 5-stage pipeline implementation:
  - Enable simultaneous access to instructions and data memories
  - Higher performance from reduced cycle per instruction
- Coprocessor interface for on-chip coprocessors:
  - Allows floating point, DSP, graphics accelerators
- EmbeddedICE debug capability with extensions:
  - Hardware single step
  - Breakpoint on exception
- High code compatibility with ARM7TDMI
- Portable to 0.25, 0.18 µm CMOS and below

---

ARM940T Macrocell

- Processor for real-time embedded applications:
  - ARM9TDMI Core
  - 4 KB instruction and 4 KB data caches
  - Memory protection unit (MPU) for RTOS
    - Allows eight separate regions of memory
    - Each with cache, write buffer enable, and access permission
    - Configured using on-chip registers
      - Eliminates the need for page-mapping tables stored in memory
  - Code compatible from ARM7TDMI
  - Hard Macro IP: 4.2mm² on 0.18
  - Up to 200 MHz on TSMC standard 0.18 µm
  - Power: 0.75 mW/MHz
  - Applications: wireless networking devices, printers, or automotive control devices
ARM9E Architecture

- Hard macrocells always have been the ultimate answer for optimized performance and die size
  - But newer synthesized design flows are pushing the envelope for SoC applications
- ARM released a whole family with high performance configurable CPUs
  - The ARM9E family was built upon the standard set by the ARM9TDMI family
  - But it also provides freedom for defining the cache and SRAM configurations used by the core
- Fully code compatible with ARMv4T architecture cores
- 32-bit load/store RISC architecture
- Efficient 5-stage pipeline
- ARM and Thumb instruction sets

ARM9E Architecture (Cont.)

- ETM9 (Embedded Trace Macrocell) interface
  - Enhance the debug capabilities
- The first family of CPUs designed to use the AHB bus of the AMBA 2.0 specification
- Coprocessor interface
- Synthesizable or soft IP
- Includes DSP extensions for true real-time systems
  - Improvement to introduce additional multiply and saturated math instructions for use by complex DSP algorithms

<table>
<thead>
<tr>
<th>I-Decode</th>
<th>Register Bank 37 x 32-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit Shifter</td>
<td>MAC &amp; SAT</td>
</tr>
<tr>
<td>32-bit ALU</td>
<td></td>
</tr>
</tbody>
</table>
ARM9E Family

ARM10E Architecture Enhancements

- ARM10E implements:
  - Harvard 6-stage pipeline
  - Supports v5TE instruction set
  - Fully compatible with v4T architecture
  - Enhanced EmbeddedICE debug logic
- Advanced microprocessor cores with 390-700 MIPS integer performance
- Performance enhancements include:
  - Branch prediction:
    - Eliminates 70% of branches on typical code sequences
  - VFP10: high performance vector floating-point support
    - Single and double precision IEEE 754 floating-point arithmetic
    - Vector load/stores can run concurrently
  - New energy saving power down modes
    - To achieve low-power operation on high performance processes
  - These features improve code performance by lowering the average number of cycles per instruction of the processor
Family Summary

<table>
<thead>
<tr>
<th>Integer core Architecture Version</th>
<th>ARM7TDI ARMV4T</th>
<th>ARM9TDI ARMV4T</th>
<th>ARM9E-S ARMV5TE</th>
<th>ARM10E ARMV5TE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline/Bus Architecture</td>
<td>3-stage, Von Neumann</td>
<td>5-stage, Harvard</td>
<td>5-stage, Harvard</td>
<td>6-stage, Harvard</td>
</tr>
<tr>
<td>Typical Die Size** mm²</td>
<td>0.54 (0.18 u, 4 LM)</td>
<td>1.1 (0.18 u, 4/5 LM)</td>
<td>1.0 (0.18 u, 4/5 LM)</td>
<td>7.5 (0.18 u, 5 LM)</td>
</tr>
<tr>
<td>Power Consumption (Ave) mW/MHz</td>
<td>0.25</td>
<td>0.26</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Clock Speed** MHz (worst case)</td>
<td>100</td>
<td>220</td>
<td>150</td>
<td>225</td>
</tr>
<tr>
<td>Cycle per instruction</td>
<td>1.9</td>
<td>1.5</td>
<td>1.5</td>
<td>1.2</td>
</tr>
<tr>
<td>Core Derivatives</td>
<td>720T, 740T</td>
<td>920T, 922T, 940T</td>
<td>996E-S, 946E-S, 926EJ-S</td>
<td>1022E, 1020E</td>
</tr>
</tbody>
</table>

** Numbers will vary with partner process.

– **ARM Processors for Embedded Applications**
  - Roadmap for ARM Processors
  - ARM Architecture Basics
  - ARM Families
  - AMBA Architecture
**AMBA: An Open Bus Standard**

- Easy interconnection of macrocells
- Optimizes system power
- Simplifies reuse
- Eases testing
- Proven and open standard
- Reduces time to market

**Advanced High Performance Bus (AHB)**

- The AHB bus is designed with all timing placed on the rising clock edge
  - This timing maximizes the efficiency of the system and aligns with modern synthesis design flows
- Multiple bus masters
  - Shares resources between different bus masters (CPU, DMA Controller, etc)
  - Optimizes system performance
- Pipelined and burst transfers
  - Allows high speed memory and peripheral access
- Split transactions supported
  - Enables high latency slaves to release the system bus during long transactions
- Wide data bus configuration
  - 32/64/128 up to 1024-bits wide
**Advanced Peripheral Bus (APB)**

- Simplicity and power conservation are the driving features behind the design of the APB side of the bridge.
- Simple bus
  - Non-pipelined architecture
  - All peripherals act as slaves on the APB, easing the integration of this bus
    - The bridge is the only APB bus master
  - Simpler interface means low gate count
- Low power
  - Isolated peripherals behind the bridge reduces load on the main system bus
  - Peripheral bus signals active only during low bandwidth peripheral transfers
- Ideal for ancillary or general-purpose peripherals
  - E.g. Timers, interrupt controllers, and I/O ports

---

**AMBA Design Kit**

- Components for EASY (Example AMBA Systems) development:
  - The example bus master provides a starting point for bus master development
  - A file reader bus master generates AMBA transactions to exercise the bus and peripherals
  - An arbiter provides a simple priority algorithm to control master access to the AHB bus
**ARM7TDMI and 922T EASYs**

**AMBA Design Kit Components**

- **AHB**
  - File Reader Bus Master
  - Example Bus Master
  - Multi-Layer Bus Switch
  - AHB TIC (Test Interface Controller)
  - AHB to APB and AHB to AHB Bridges
  - AHB Arbiter
  - Internal Memory
  - Example AHB Slave with Retry
  - Simple External Memory Interface

- **APB**
  - Timers
  - Interrupt Controller
  - Remap/Pause Controller

- Example AMBA Bus Systems
- Macrocell Wrappers
  - ARM720T
  - ARM740T
  - ARM940T
  - ARM920T
  - ARM922T
  - ARM7TDMI
  - ARM7TDMI-S

- Example Software
  - C and ARM Assembler

- Documentation
  - AMBA Specification
  - User Guide and Release Notes