A comparative study of different StrongARM processors

Presented by

Rajiv Ranjan Singh

Id. No. 2001PHXF419

20-09-2002

Supervisor: Prof. Rahul Banerjee

Two Computer architectures

1. CISC (Complex Instruction Set Computing)

2. RISC (Reduced Instruction Set Computing)

Comparison between RISC & CISC

Sr. No	RISC	CISC
1	Simple instructions taking 1 cycle	Complex instructions taking multiple cycles
2	Only LOADS / STORES reference memory	Any instruction may reference memory
3	Highly pipelined	Not pipelined or less pipelined
4	Instructions executed by the hardware	Instructions interpreted by the microprogram
5	Fixed format instructions	Variable format instructions
6	Few instructions and modes	Many instructions and modes
7	Complexity is in the compiler	Complexity is in the microprogram
8	Multiple register sets	Single register set

History of RISC computers

- First commercial RISC in 1985 by **Acorn**
- First low-cost RISC-powered PC in 1987 by **Acorn**
- **ARM** (joint venture of Acorn+Apple+VLSI) was established in Nov 1990
- **StrongARM** (joint venture of ARM+Digital) established in 1998
- **Intel** bought the license for StrongARM from ARM

ARM Architecture Features

The typical RISC Features:

- A large uniform register file
- A *load/store* architecture, where data-processing operations operate only on register contents, not directly on memory contents
- Simple addressing modes, with all *load/store* addresses being determined from register contents and instruction fields only
- Uniform and fixed-length instructions fields, to simplify instruction decode

ARM Architecture Features (continued)

Additionally, ARM instruction gives:

- Control over both the *ALU* and shifter in every data processing instruction to maximize the use of an ALU and a shifter
- Auto-increment and auto-decrement addressing modes to optimize program loops
- Load and Store multiple instructions to maximize data throughput
- Conditional execution of all instructions to maximize execution throughput

These enhancements to a basic RISC architecture allow **ARM** processor to achieve a good balance of high performance, low code size, low power consumption and low silicon area

ARM instruction set architecture (Version 1)

This version was implemented by **ARM 1** and was never used in a commercial product.

It had only 26-bit address space and is now obsolete.

It contained:

- The basic data processing instructions (not including multiplies)
- Byte,word, and multi-word LOAD / STORE instructions
- Branch instructions,including a branch-and-link instruction designed for subroutine calls
- A software interrupt instruction, for use in making Operating System calls

ARM instruction set architecture (Version 2)

This version extended the **Version 1** architecture by adding:

- Multiply and multiply-accumulate instructions
- Coprocessor support
- Two more banked registers in fast interrupt mode
- Atomic load-and-store instructions called SWP and SWPB (in a slightly variant version called version 2a)

Version 2 and **2a** still only had a 26-bit address space and are now obsolete

ARM instruction set architecture (Version 3)

- Extended the addressing range to 32-bits
- Program Status information which was stored in R15 previously is now been stored in the ***Current Program status Register*** **(CPSR)** and ***Saved Program Status Registers (*SPSRs)** to preserve the **CPSR** contents when an exception occurs.
- The following changes occurred to the instruction set:
  - two instructions **(MRS and MSR)** were added to allow the new **CPSR** and **SPSRs** to be accessed
  - the functionality of instructions previously used to return from exceptions was modified to allow them to continue to be used for that purpose
- Two new processor modes were added to use Data Abort, Prefetch Abort and undefined Instructions exceptions effectively in Operating System codes

ARM instruction set architecture (Version 4)

This version added the following to the architecture **Version 3**:

- Halfword load/store instructions
- Instructions to load and sign-extend bytes and halfwords
- In T variants , an instruction to transfer to Thumb state
- A new privileged processor mode that uses the User mode registers

Version 4 also made it clearer which instructions should cause the undefined Instruction exception to be taken.

ARM instruction set architecture (Version 5)

This version added some new instructions and modified the definitions of some of the instructions of **Version 4** to:

- Improve the efficiency of **ARM/Thumb** ineterworking in T variants
- Allow the same code generation techniques to be used for non-T variants as for T variants

Version 5 also:

- Adds a count leading zeros instruction, which (among other things) allows more efficient integer divide and interrupt prioritization routine
- Adds a software breakpoint instruction
- Adds more instruction options for coprocessors designers
- Tightens the definitions of how flags are set by multiply instructions

Additions to the Architecture feature Version 5 over Version 4

- Tiny pages of 1 Kbytes each
- A new instruction (CLZ) that counts the leading zeros in a data value
- Enhanced ARM-Thumb transfer instructions
- New breakpoint instructions (BKPT)
- A modification of the system control coprocessor, CP15

The Thumb Instruction set (T variants)

Thumb Instruction Set are:

- Introduced with architecture version 4
- Re-encoded subset of ARM instruction set
- Half the size of ARM instructions (16-bits compared with 32), hence greater code density

Limitations:

- Thumb code usually uses more instructions for the same job, so ARM code is usually best for maximizing the performance of time-critical code
- The Thumb instruction set does not include some instructions that are needed for exception handling, so ARM code needs to be used for at least top-level exception handlers (Due to this reason Thumb Instruction is used in conjunction with a suitable ARM instruction set)

Enhanced DSP instructions (E variants)

They include:

- Several new multiply and multiply-accumulate instructions that act on 16-bit data items
- Addition and subtractions instructions that perform *saturated* signed arithmetic. This is a form of integer arithmetic that produces the maximum negative or positive values instead of wrapping around if the calculation overflows the normal integer range.
- Load **(LDRD)**, store **(STRD)** and coprocessor register transfer **(MCRR** and **MRRC)** instructions that act on 2 words of data
- A cache preload instruction **PLD**

These instructions were first defined as a variant of architecture version 5T and their presence is denoted by the variant letter E (i.e. **ARMv5TE**)

Classification of ARM Instruction set

1. Branch instructions
2. Data processing instructions
3. Status register transfer instructions
4. Load and store instructions
5. Coprocessor instructions
6. Exceptions-generating instructions

ARM processor modes (7 modes)

Modes	Abbrv.	Description
User	usr	Normal program execution mode
FIQ	fiq	Supports a high speed data transfer or channel process
IRQ	irq	Used for general purpose interrupt handling
Supervisor	svc	A protected mode for the operating system
Abort	abt	Implements virtual memory and/or memory protection
Undefined	und	Supports software emulation of hardware coprocessors
System	sys	Runs privileged operating system tasks (**ARMv4** and above)

Processor Modes Features

User Mode:

- Most application programmes are executed
- The executing program can not access some protected system resources as well as change mode but can cause an exception to occur
- Allows a suitably written operating system to control the use of system resources

Privileged Mode:

1. **Exception Mode:**

   --- There are five modes (FIQ, IRQ, Supervisor,Abort, Undefined)

   --- These are entered when specific exceptions occur and each have some additional registers to avoid corrupting User mode state when these exception occur.

2. **System Mode:** Used by operating system tasks which need access to system resources but avoids using additional registers associated with exception modes.

ARM Registers

The ARM processors has a total of 37 registers:

- 31 general purpose 32-bit registers including a program counter. At any one time 16 of these registers are visible.The other registers are used to speed up exception processing. The main bank of 16 registers is used by all unprivileged code. Two of the 16 visible registers have special roles:

  Link Registers: This register holds the address of the next instruction after a Branch and Link (BL) instruction, which is the instruction used to call a subroutine. At all other times it acts as a general purpose registers.

  Program Counter: Acts as pointer to the instruction which is two instruction after the instruction being executed.Because all ARM instructions are aligned on a word boundary.

ARM Registers (continued)

- 6 status registers each 32-bit wide but only 12 of the 32-bit are allocated or need to be implemented.

CPSR: This is accessible in all processor modes and contains condition code flags, interrupt disable bits, the current processor mode and the other status and control information.

SPSR: Available for each exception mode (i.e. 5 registers) and is used to preserve the value of the CPSR when the associated exception occurs.

Comparison between ARM and StrongARM processors

S.N.	Features	StrongARM	ARM7	ARM9
1	Architecture	Harvard	Von Neumann	Harvard
2	Pipeline	5 / 7 stage Pipeline	3 stage Pipeline	5 stage Pipeline
3	Instruction set support	ARM+ Thumb+ Enhanced DSP instructions	Both ARM and Thumb	Both ARM and Thumb
4	Architecture Version	ARM architecture Version 4 & 5TE	architecture 4T (with Thumb)	architecture 4T (with Thumb)
5	Booth's multiplier	12 bit	8-bit	8 bit
6	Data abort model	Base restored	Base updated	Base restored

Changes to the SA-1100 / SA-1110 Core from the SA-110

- Data cache reduced from 16 Kbyte to 8 Kbyte
- Interrupt vector adjust capability
- Read buffer (non blocking)
- Minicache for alternate data caching
- Hardware breakpoints
- MMU enhancements
- Process ID mapping

Feature Additions to the SA-1100 [SA-1110] Core from the SA-110

- Memory controller supporting ROM, Flash, EDO, DRAM, SRAM
  [SDRAM, SMROM, SRAM like variable latency I/O]
- LCD controller (1-,2-,4-bit gray scale levels / 8-,12-,16- color levels)
- 230-Kbps UART
- Touch-screen, audio, telecom port
- Infrared data (IrDA) serial port (115 Kbps, 4 Mbps)
- Six channel DMA controller
- Integrated two slot PCMCIA controller
- Twenty-eight general-purpose I/O ports
- Real-time clock with interrupt capability

Feature Additions to the SA-1100 / SA-1110 Core from the SA-110

- On-chip oscillators for clock sources
- Interrupt controller
- Power management features
  - ---- Normal (full-on) mode
  - ---- Idle (power-down mode)
  - ---- Sleep (power-down) mode
- Four general purpose interruptible timers
- 12-Mbps USB device controller
- Synchronous serial port (UCB1100, UCB1200, SPI, TI, Wire)
- Serial communications module supporting SDLC

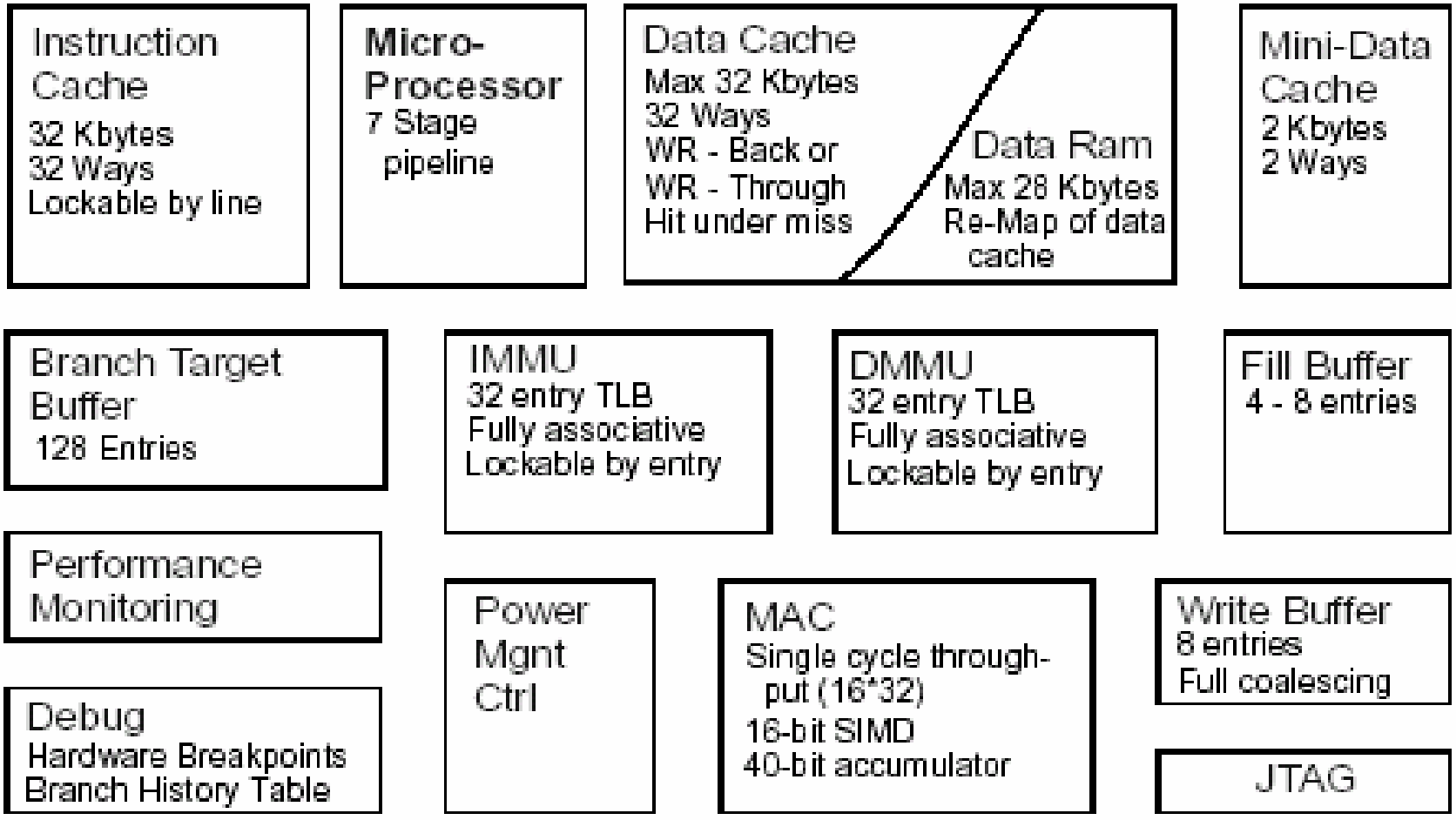
Feature Additions to the SA-1110 core from the SA-1100

- SDRAM support
- SMROM support (32-bit only) on CS0-3
- Ready input signal for variable latency I/O devices (for example, graphic chips)
- CS4 and CS5 for variable latency I/O devices, ROM or Flash memory
- CS3 support for variable latency I/O devices (instead of SRAM)
- Support for burst (page-mode) read timings from Flash memory
- Support for 16-bit data buses on all memory types (except SMROM)
- Support for SRAM, DRAM and SDRAM in the same system.

Intel® PXA250 and PXA210 application processors

- They are the first highly integrated-system-on-chip (SOC) and includes a high-performance low power Intel® XScale™ core with a variety of system peripherals
- They are also known as Application Specific Standard Products (ASSP) to be used in embedded products such as handheld devices, networking, storage, remote access servers etc.
- Provide industry-leading MIPS/mw performance for Handheld Computing Applications
- PXA250 is a 32-bit version of the application processor
- PXA210 is a 16-bit version of the application processor

Intel® XScale™ Microarchitecture Architecture Features



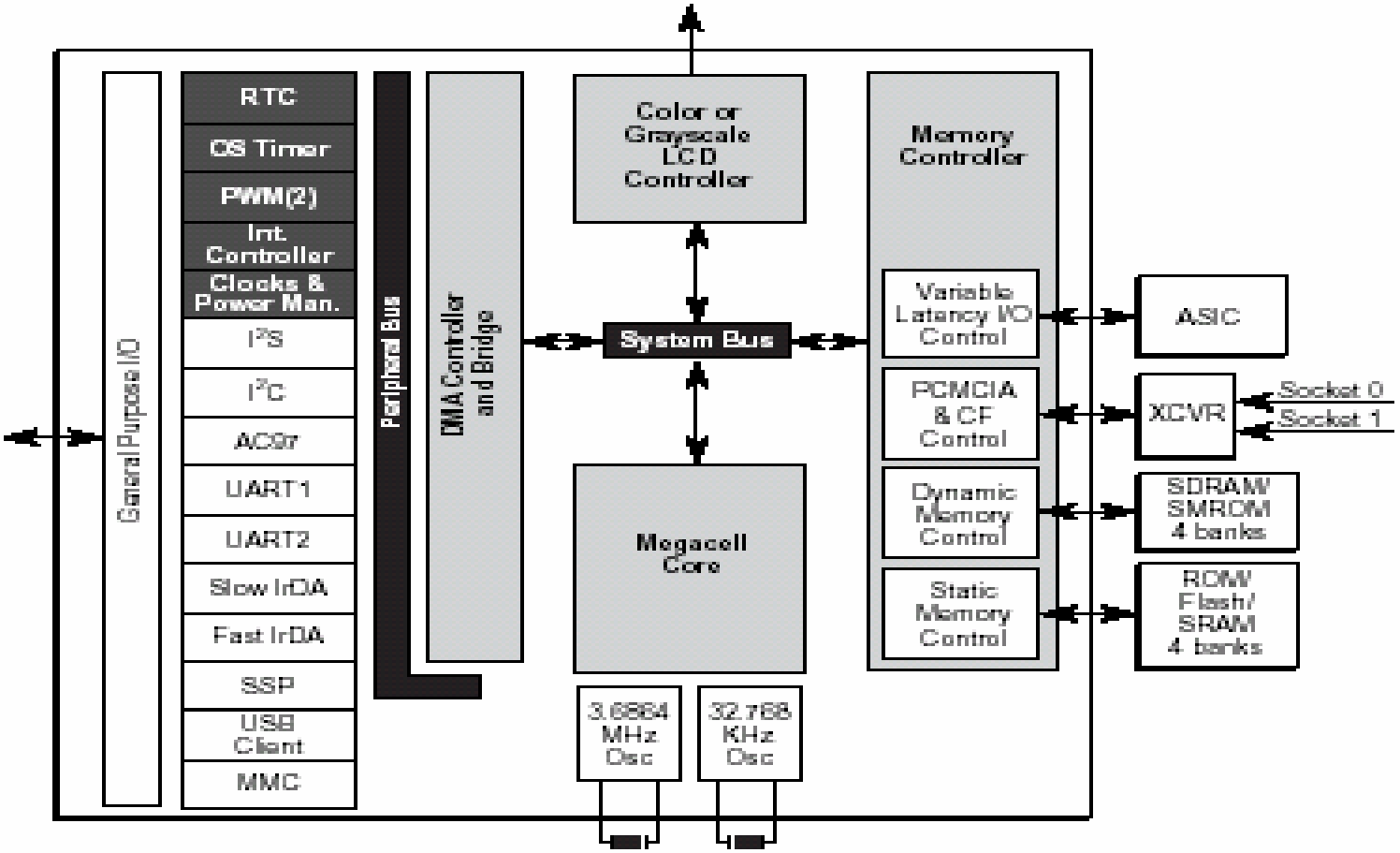
Intel® XScale™ Core Features

- ARM™ Version 5TE ISA compliant
  - -ARM Thumb Instruction Support
  - -ARM DSP Enhanced Instructions Support
- Low Power consumption and high performance
- Intel Media Processing Technology
  - - Enhanced 16-bit multiply
  - - 40-bit accumulator
- 32-KByte Instruction Cache
- 32-KByte Data Cache
- 2-KByte Mini Instruction cache

Intel® XScale™ Core Features (continued)

- 2-KByte Mini Instruction cache
- Instruction and Data Memory Management Units
- Branch Target Buffer
- Debug capability via JTAG support

Intel® PXA250 and PXA210 Processor Architectural Block Diagram



PXA250 AND PXA210 Architectural block brief description

1. **USB Client:** It supports up to 16 endpoints. The USB device controller provides FIFOs with DMA access to or from memory.
2. **DMA controller:** The DMAC provides up to 16 prioritized channels to service transfer requests from internal peripherals and up to two data transfer requests from external companion chip. This is compatible with peripherals that use word, half-word or byte data size.
3. **I^2S Controller:** Provides a serial link to standard I^2S CODECs for digital stereo sound. The controller provides FIFOs that support DMA access to memory.
4. **Multimedia Card (MMC) Controller:** Provides serial interface to standard memory cards and serial data transfer up to 20 Mbps.This also has FIFOs that support DMA access to and from memory.

PXA250 AND PXA210 Architectural block brief description (continued)

5. **I²C Bus Interface Unit:** Provides a general purpose 2-pin serial communication port. The interface uses one pin for data and address and one pin for clocking.
6. **UART:** Provides three UARTs each can be used as a slow infrared (SIR) transmitter / receiver based on the Infrared Data Association serial Infrared (SIR) Physical Layer Link Specification. All have FIFOs with DMA access to and from memory.
   - Full Function UART (FFUART): Baud rate programmable up to 230 Kbps
   - Bluetooth UART (BTUART): Baud rate programmable up to 921 Kbps
   - Standard UART (STUART): Baud rate programmable up to 230 Kbps.
7. **AC97 Controller:** This support the AC97 Revision 2.0 CODECs which operates at sample rate of up to 48 KHz. The controller supports five independent 16-bit channels with FIFOs that support DMA access to memory

PXA250 AND PXA210 Architectural block brief description (continued)

8. **Fast Infrared (FIR) Communication Port:** This is based on the 4-Mbps Infrared Data Association (IrDA) specification which operates at half duplex and has FIFOs with DMA access to memory. This port uses the STUART's transmit and receive pins to directly connect to external IrDA LED transceivers.
9. **Synchronous Serial Protocol Controller (SSPC):** The SSP port provides a full duplex synchronous serial interface that operates at bit rates from 7.2 KHz to 1.84 MHz. It supports National Semiconductor's Microwire, Texas Instrument's Synchronous Serial Protocol, and Motorola's Serial Peripheral Interface. The SSPC has FIFOs with DMA access to memory.
10. **PWM:** It has two independent programmable outputs to drive two GPIOs.
11. **Interrupt Control:** This directs processor interrupts into core's IRQ & FIQ inputs.

Why StrongARMs?

- Low power consumption
- Suitable for PDAs application
- Various serial device support (e.g. USB, I^2C, UART etc.)
- LCD controller support
- Various Memory control units
- Variable latency I/O support to connect ASIC devices

REFERENCES

- www.arm.com
- www.intel.com
- ARM Architecture Reference Manual
- SA-1100 Microprocessor, Technical Reference Manual
- Intel StrongARM SA-1110 Microprocessor, Developer's Manual
- Intel PXA250 and PXA210 Application Processors, Developer's Manual

Thank You !