System-on-a-Chip (SoC) & ARM Architecture

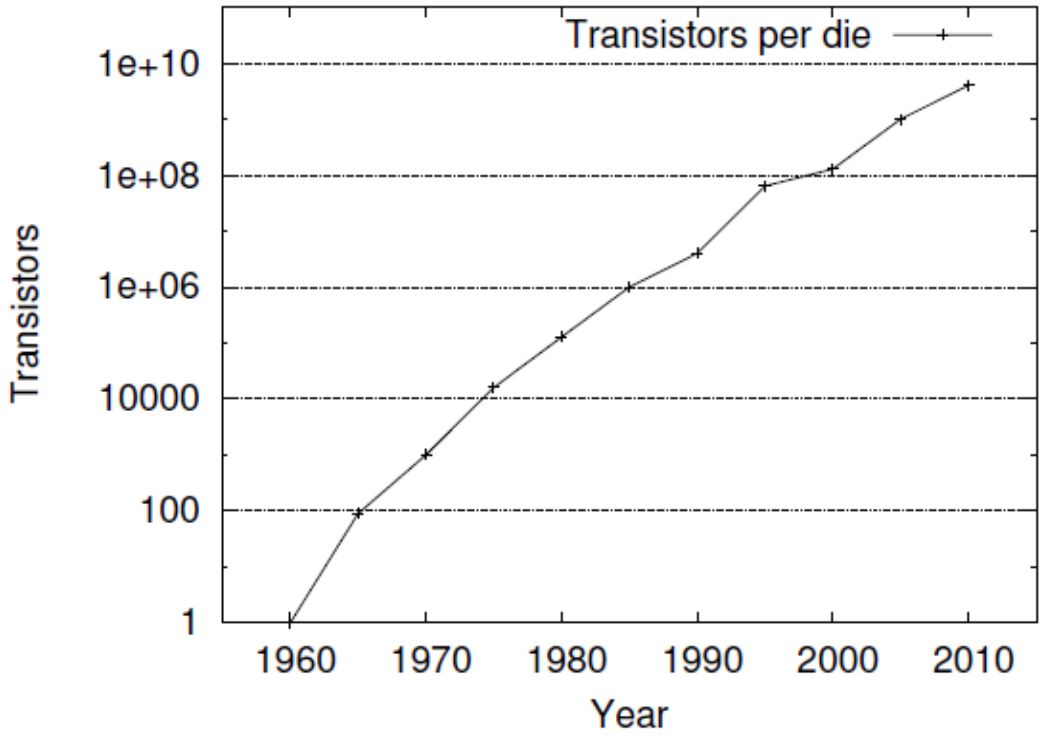
EE2222 Computer Interfacing and Microprocessors

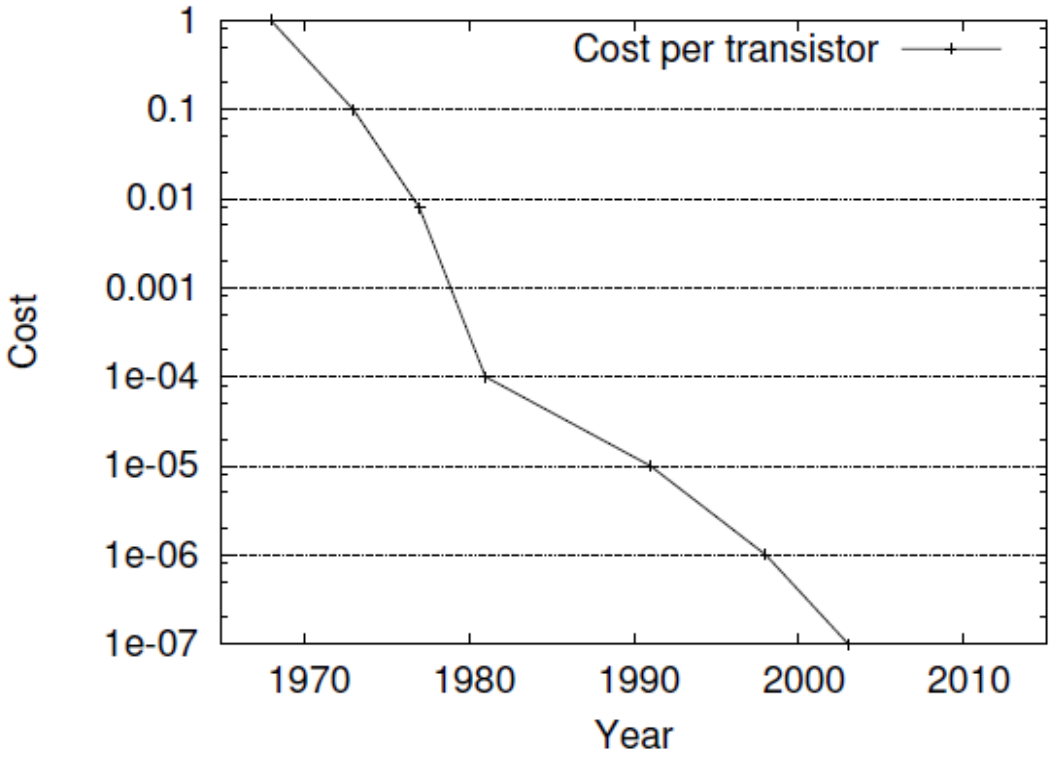
Partially based on
System-on-Chip Design by Hao Zheng

Overview

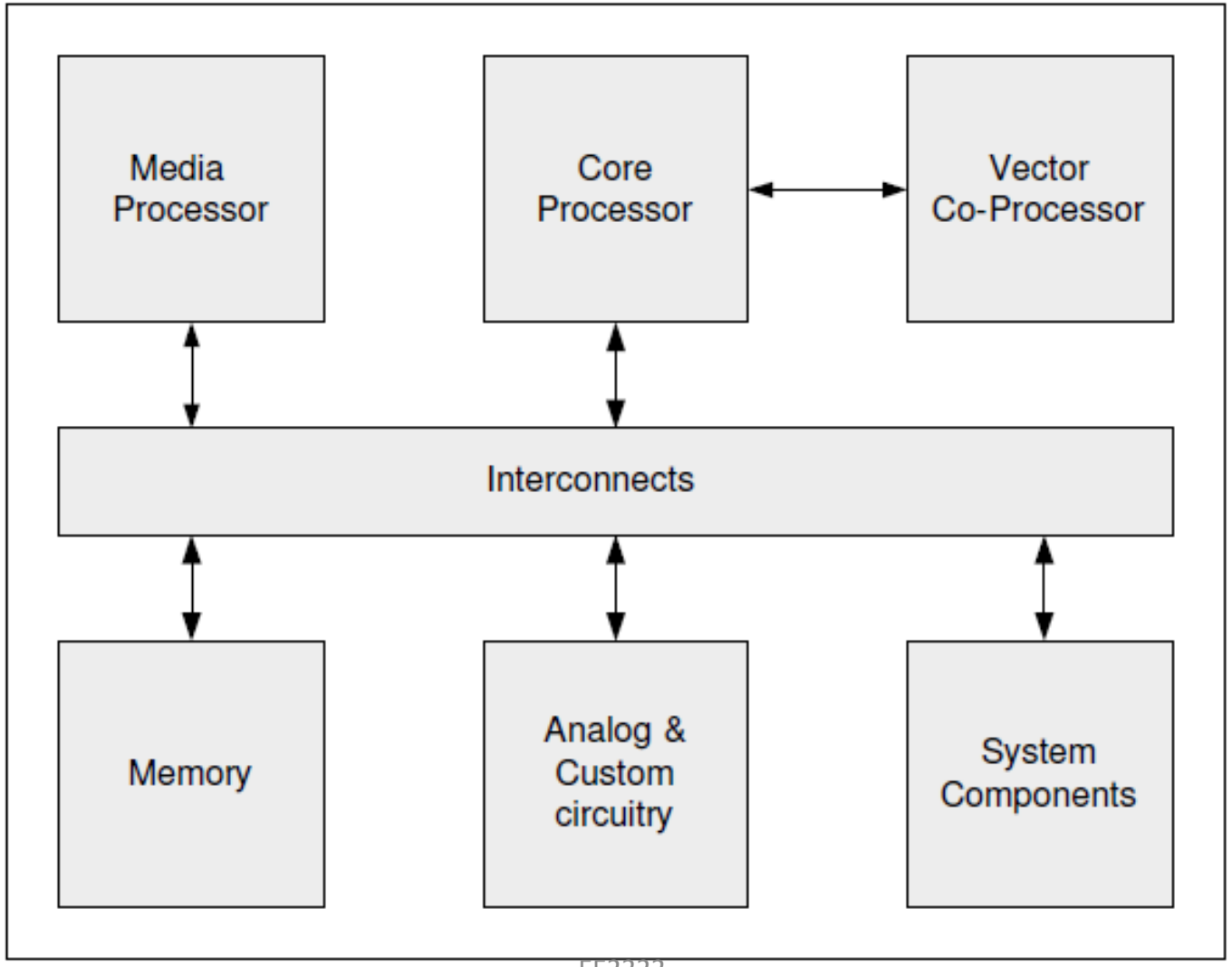
- A system-on-a-chip (SoC):
  - a computing system on a single silicon substrate that integrates both hardware and software.
- Hardware packages all necessary electronics for a particular application.
  - which implemented by SW running on HW.
- Aim for low power and low cost.
  - Also more reliable than multi-component systems.

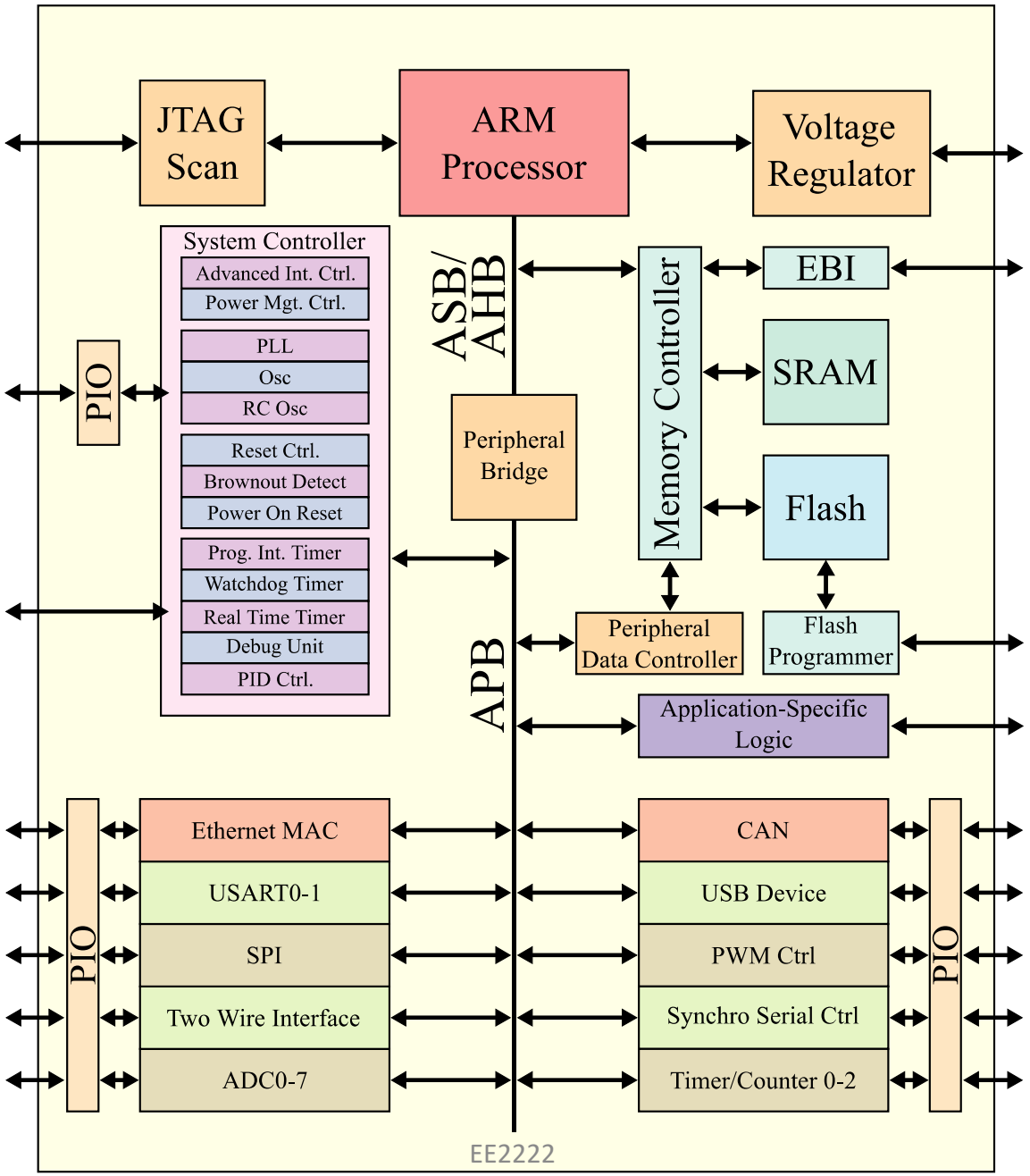
Driven by semiconductor advances





Basic SoC Model




JTAG Scan
ARM Processor
Voltage Regulator
System Controller
Advanced Int. Ctrl.
Power Mgt. Ctrl.
PLL
Osc
RC Osc
Reset Ctrl.
Brownout Detect
Power On Reset
Prog. Int. Timer
Watchdog Timer
Real Time Timer
Debug Unit
PID Ctrl.
PIO
ASB/ AHB
Peripheral Bridge
Memory Controller
EBI
SRAM
Flash
Peripheral Data Controller
Flash Programmer
APB
Application-Specific Logic
Ethernet MAC
USART0-1
SPI
Two Wire Interface
ADC0-7
CAN
USB Device
PWM Ctrl
Synchro Serial Ctrl
Timer/Counter 0-2
PIO
PIO

SoC vs Processors

	System on a chip	*Processors on a chip*
processor	multiple, simple, heterogeneous	few, complex, homogeneous
cache	one level, small	2-3 levels, extensive
memory	embedded, on chip	very large, off chip
functionality	special purpose	general purpose
interconnect	wide, high bandwidth	often through cache
power, cost	both low	both high
operation	largely stand-alone	need other chips

Embedded Systems

- 98% processors sold annually are used in embedded applications.

Embedded Systems: Design Challenges

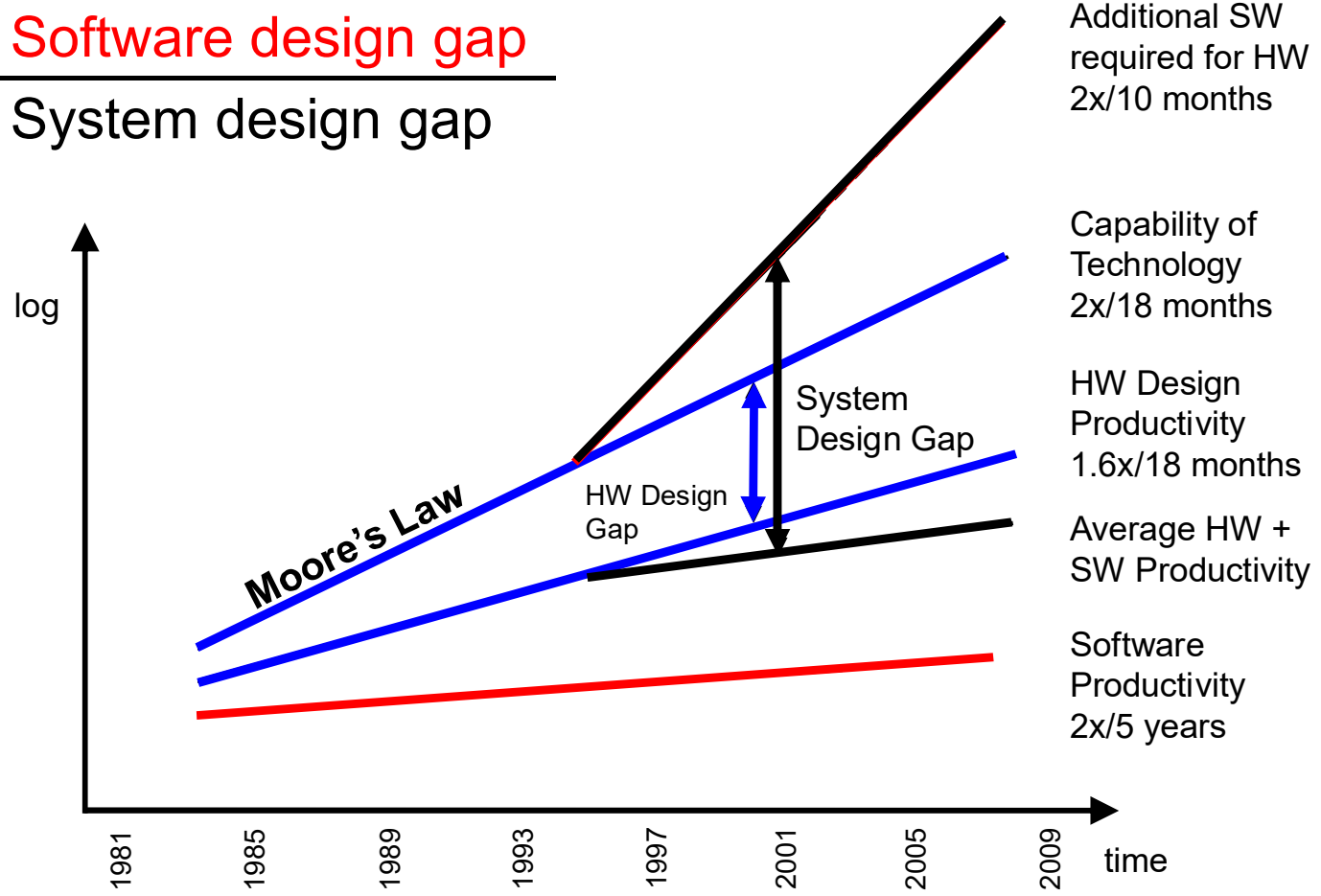
- Power/energy efficient:
  - mobile & battery powered
- Highly reliable:
  - Extreme environment (e.g. temperature)
- Real-time operations:
  - predictable performance
- Highly complex
  - A modern automobile with 55 electronic control units
- Tightly coupled Software & Hardware
- Rapid development at low price

Design Complexity Challenges

Hardware design gap

\+ Software design gap

= System design gap



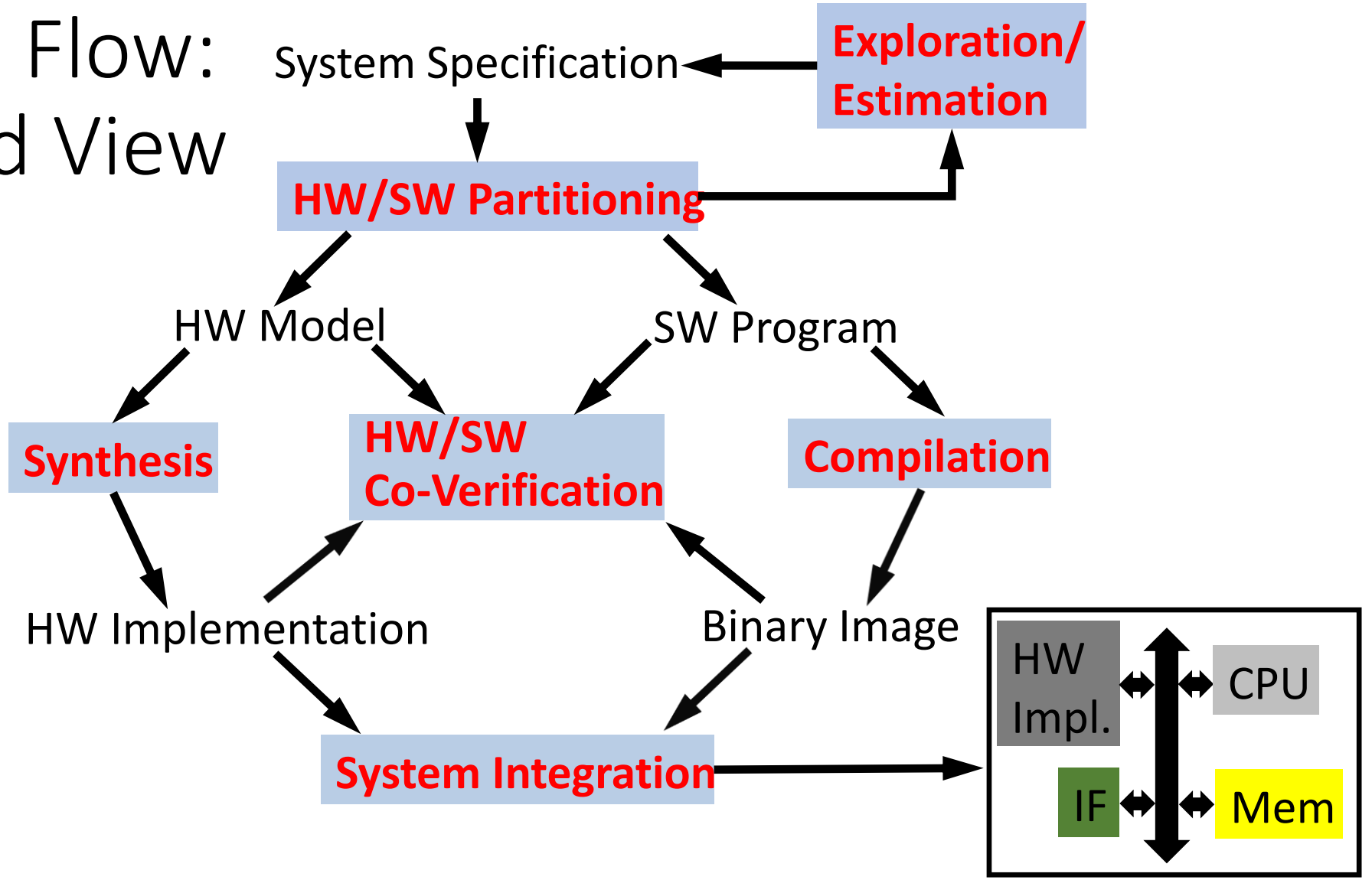
(source: *"Hardware-dependent Software"*, Ecker et al., 2009)

Answer to design complexity challenges: Move to higher levels of abstraction

Levels of Abstraction

- **Circuit**:
  - network of transistors
- **Logic**:
  - network of basic logic gates
  - AND/OR/NOT, latches/FFs, etc.
- **Processor**:
  - network of logic components
  - i.e. ALU, MUX, decoders, registers, etc.
- **System**:
  - network of processors, memories, buses, and other custom processing logic

SoC Design Flow: A Simplified View



System Synthesis

- Processes -> CPUs or custom logic
  - HW/SW partitioning
- Communication -> Buses or NoC
- Flow
  - Profiling & Estimation
  - Component & connection allocation
  - Process and channel binding
  - Process scheduling
  - IF component insertion
  - Model refinement

High-Level Synthesis

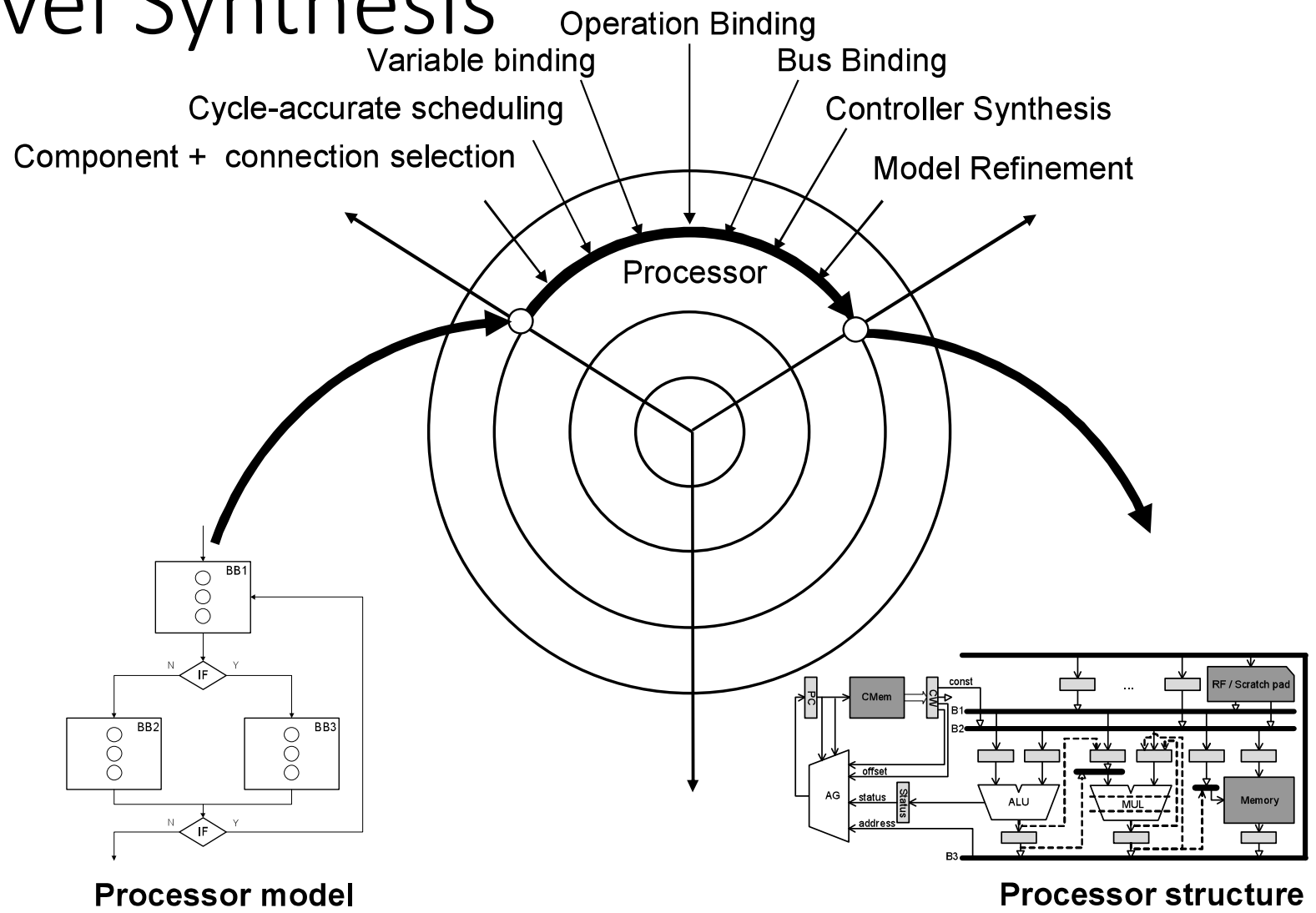


FIGURE 1.6 Processor synthesis

Hardware/Software Co-Design

- Definition:
  - *HW/SW co-design is the design of cooperating HW components and SW components in a single design effort.*

- Alternative definition:
  - *HW/SW co-design means meeting system level objectives by exploiting the synergy of HW and SW through concurrent design.*

Platform Methodology

- Reuse of previous defined platforms
  - With well-defined structures and standard components.
- Add more components necessary for an application.
  - These components are then synthesized.
- System implementation is generated by combining the layouts of existing and custom components.
- Advantages: faster development, lower cost,

ARM Architecture

Why ARM?

- Leading provider of 32-bit embedded RISC microprocessors
  - 75% of market
- Common architecture
- High performance
- Low power consumption
- Low system cost
- Solutions for
  - Embedded real-time systems for mass storage, automotive, industrial and networking applications
  - Secure applications – smartcards and SIMs
- Open platforms running complex operating systems

History of ARM

- ARM (Acorn RISC Machine) started as a new, powerful, CPU design for the replacement of the 8-bit 6502 at Acorn Computers in 1985
- First models had only a 26-bit program counter, limiting the memory space to 64 MB (a lot at that time)
- 1990 spin-off: ARM renamed Advanced RISC Machines
- ARM now focuses on Embedded CPU cores
  - IP licensing: Almost every silicon manufacturer sells some microcontroller with an ARM core. Some even compete with their own designs.
  - Processing power with low current consumption
  - Ideal for portable devices

ARM processors vs. ARM architectures

- ARM architecture
  - Describes the details of instruction set, programmer's model, exception model, and memory map
  - Documented in the Architecture Reference Manual
- ARM processor
  - Developed using one of the ARM architectures
  - More implementation details, such as timing information
  - Documented in processor's Technical Reference Manual

ARM architecture features

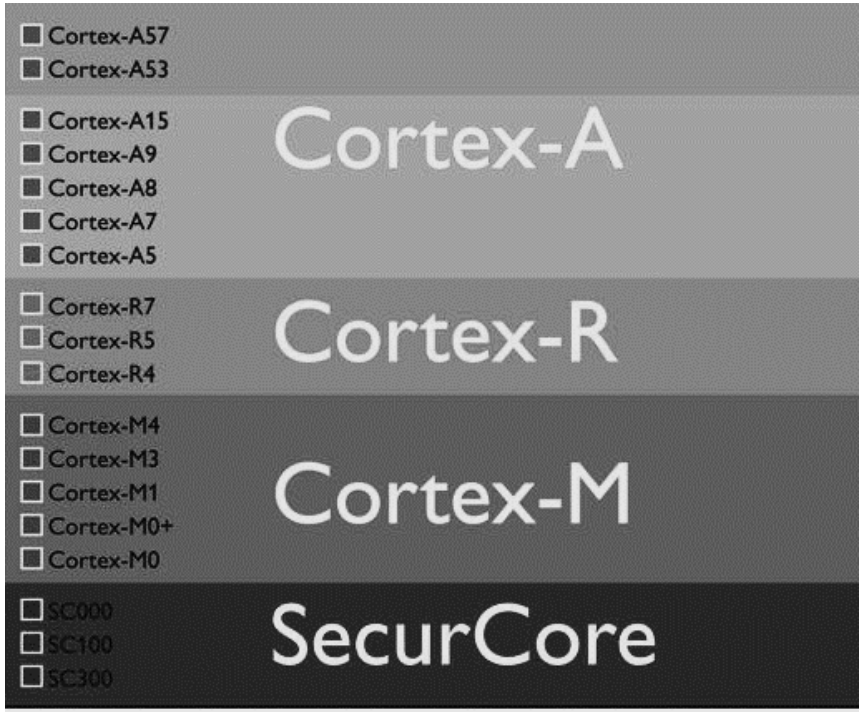
- The typical RISC features:
  - A large uniform register file
  - A load/store architecture, where data-processing operations operate only on register contents, not directly on memory contents
  - Simple addressing modes, with all load/store addresses being determined from register contents and instruction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

ARM architecture features

- 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 processor lines

- ARM architectures and processor families can be profiled into four groups:
- The Cortex-M profile
  - Processors of the M profile are optimized for cost sensitive and microcontroller applications, like automotive body electronics, smart sensors.
- The Cortex-A profile
  - It aims at high-end applications running open and complex OSs, like smartphones, tablets, netbooks, eBook readers.
- The Cortex-R profile
  - It marks processors for real time applications, like mass storage or printer controllers.
- The SecureCore profile
  - The ARM SecurCore™ processor family provides processors with security features for applications like smartcards, pay TV, eGovernement.

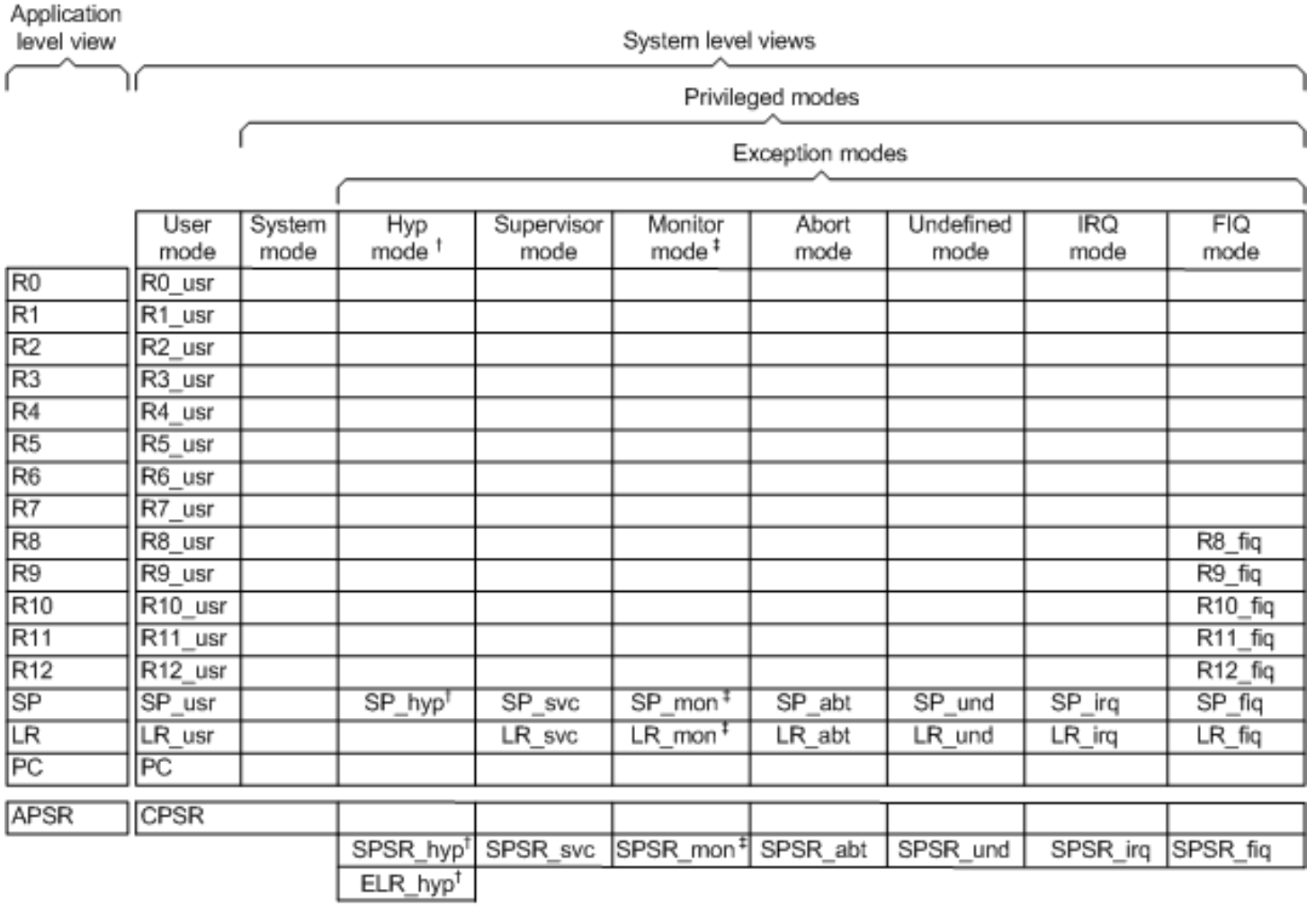


ARM processor modes

- The ARM has seven basic operating modes:
  - User: unprivileged mode under which most tasks run
  - FIQ: entered when a high priority (fast) interrupt is raised
  - IRQ: entered when a low priority (normal) interrupt is raised
  - Supervisor: entered on reset and when a Software Interrupt instruction is executed
  - Abort: used to handle memory access violations
  - Undef: used to handle undefined instructions
  - System: privileged mode using the same registers as user mode

ARM register set

- ARM processors provide general-purpose and special-purpose registers.
- Some additional registers are available in privileged execution modes.



Application level view		System level views							
		Privileged modes							
		Exception modes							
	User mode	System mode	Hyp mode †	Supervisor mode	Monitor mode ‡	Abort mode	Undefined mode	IRQ mode	FIQ mode
R0	R0_usr								
R1	R1_usr								
R2	R2_usr								
R3	R3_usr								
R4	R4_usr								
R5	R5_usr								
R6	R6_usr								
R7	R7_usr								
R8	R8_usr								R8_fiq
R9	R9_usr								R9_fiq
R10	R10_usr								R10_fiq
R11	R11_usr								R11_fiq
R12	R12_usr								R12_fiq
SP	SP_usr		SP_hyp†	SP_svc	SP_mon ‡	SP_abt	SP_und	SP_irq	SP_fiq
LR	LR_usr			LR_svc	LR_mon ‡	LR_abt	LR_und	LR_irq	LR_fiq
PC	PC								
APSR	CPSR								
			SPSR_hyp†	SPSR_svc	SPSR_mon ‡	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq
			ELR_hyp†						

† Hyp mode and the associated banked registers are implemented only as part of the Virtualization Extensions

‡ Monitor mode and the associated banked registers are implemented only as part of the Security Extensions

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

ARM instruction set architecture (Version 6)

- Key ARMv6 Improvements:
  - Memory Management
  - Multiprocessing
  - Multimedia Support
  - Data Handling
  - Exceptions and Interrupts

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)

Advanced Microcontroller Bus Architecture (AMBA)

- AMBA (Advanced Microcontroller Bus Architecture) protocols are an open standard, on-chip interconnect specification for the connection and management of functional blocks in a System-on-Chip (SoC).
- It facilitates right-first-time development of multi-processor designs with large numbers of controllers and peripherals.

ARM vs. x86

- ARM processors require significantly fewer transistors than typical PC processors because of the fact that it has a RISC-based design.
- Fewer transistors minimizes power use, heat and production cost. All qualities that are preferred for battery-powered devices such as laptops, tablets, and smartphones.
- On the other hand x86 processors usually consumes a lot of energy but the are also a lot faster.
- This makes x86 based processors ideal for desktops, gaming and super computer that require speed more than energy efficiency.

ARM vs. x86

- The main difference between ARM and x86 architecture is that ARM is RISC based while x86 is CISC based.
- CISC design is to execute multiple complex (larger) instructions.
- While the RISC design is perfect for small, simple instructions.
- The ARM has a lot more registers than x86.
- The ARM has a thumb mode to increase code density so programs fit in less memory.
- All these features help ARM save power almost everywhere it can.

Resources

- The ARM University Program, ARM Architecture Fundamentals
  - https://www.youtube.com/watch?list=PLqsfB23JsD0FUtaDmaMskIW1wRtFLjmTu&v=7LqPJGnBPMM
- ARM lectures by Dr. Santanu Chaudhury, EE Department, IIT Delhi
  - http://www.youtube.com/watch?v=4VRtujwa_b8&playnext=1&list=PL95AFA4ABA8B28627&feature=results_main
- The ARM Instruction Set Architecture
  - http://users.ece.utexas.edu/~valvano/EE345M/Arm_EE382N_4.pdf