Unit 1 – Embedded Multi-core Systems

Multi-core Processors and Embedded Systems

Embedded Systems

- Embedded computers: Largest class of computers. They span the widest range of applications and performance. These computers are part of everyday products and their presence is often not apparent. They are integral parts of these products and often enclosed within them, hence the term *embedded systems*.
- Many definitions: "low-power, low-performance", "systems outside of personal computers". Common characteristics:
  - ✓ Fixed function: A fixed set of functions not easily expandable (e.g.: MP3 player).
  - ✓ Customized OS: OS can be stripped down including only the key features necessary for the device. This allow for more efficient use of the processor cores and better power utilization. Not all embedded devices possess a customized OS.
  - ✓ Customized form factor: Embedded devices are typically customized for the desired use.
  - ✓ Cross platform development: Typically, the system is customized in terms of OS and form factor, so it may not include the programming facilities necessary for development. The development environment for an embedded system is then different than the embedded system (could be a workstation).

- Fig. 1 depicts a generic block diagram of an embedded system. The microcomputer (a μcontroller: μprocessor with extra features) is the central part of the system. It resembles a PC, but its software programs are often permanently stored to provide only the functions required for the product. This software, critical to the operation of the product, is referred to as *embedded software*. The microcomputer includes some memory; additional memory can be added externally. The human interface of the microcomputer may be limited or nonexistent. Large storage components are not present. Input and output devices that can be analog (e.g.: thermistor, speaker) or digital (e.g.: switch, keypad, stepper motor, LED display).
- Examples of embedded systems:
  - ✓ Automotive: Electronic Control Unit, Airbag system, anti-locking brake system, navigation system.
  - ✓ Home appliances: microwave ovens, washing machines, digital alarm clocks, HVAC system.
  - ✓ Office automation: Copiers, fax, printer, scanners
  - ✓ Industrial Automation: hazard detecting systems, data collection systems, assembly line, industrial robots.
  - ✓ Medical equipment: pacemakers, incubators, glucose monitor, ECG, CT scanner.
  - ✓ Banking and finance: vending machines, cash register, ATM.
  - ✓ Games and toys: handheld games, talking stuffed toys
  - ✓ Communication and Media: mobile phones, routers, satellites, DVD players, digital cameras, flat panel TVs.

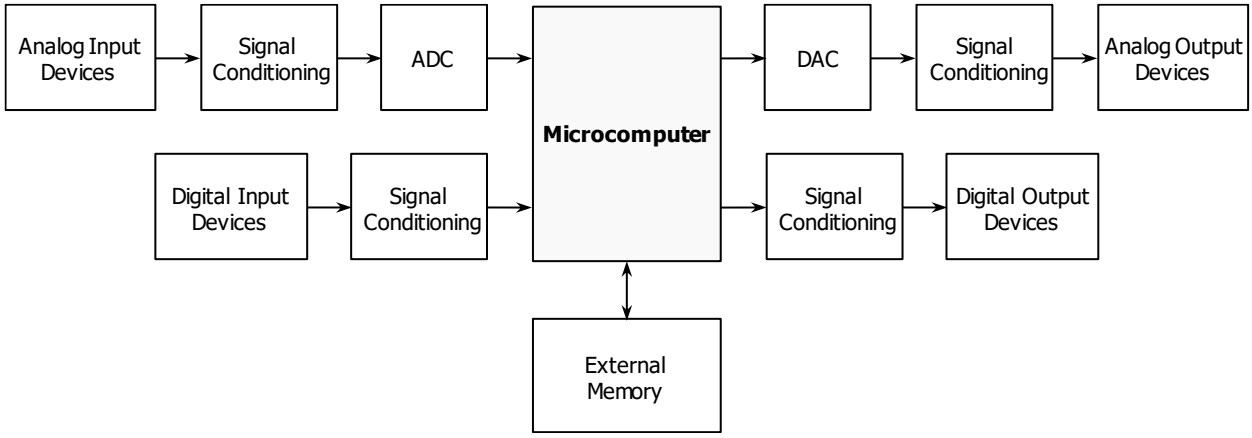

Figure 1. Block Diagram of an Embedded System.

- Employing multi-core processors in embedded systems is motivated by performance: it can result in faster execution time, increased throughput, and lower power usage for embedded applications.
- This need must be balanced by the need to keep power utilization within reasonable limits.
- Many applications in the embedded systems can benefit from the increased performance offered by multi-core processors.
- The advent of multi-core processors is pushing the need to "think parallel" to the embedded market segments.
- To take advantage of the performance benefits available with multi-core processor, you need to:
  - ✓ Multi-task (run multiple processes at the same time) or Partition (assign cores to run specific OSes)
  - ✓ Multi-thread: Design application such that work can be completed by independent workers
- The move to multi-core processor architectures in embedded systems has been spurred by these trends:
  - ✓ Diminishing returns from Instruction-Level Parallelism (ILP).
  - ✓ Clock scaling reaching limits due to power constraints. Due to the power equation ($Power = C \times V^2 \times f$) in CMOS microprocessors, increased frequency is leading to a disproportionate increase in power.

DEFINITIONS

- **Multi-core processor**: Multiple processor cores (core ≡ CPU + extra components) in the same die or chip package and interfaced to the motherboard.
  - ✓ Each core reads an executes program instructions, as if the computer had several processors.
  - ✓ They typically share some resources such as a L2 or L3 cache or memory, and I/O buses.
  - ✓ Multi-core processor architectures vary in terms of number of cores, core types, number and level of caches, how cores are interconnected.

- **Multiprocessor system**: multiple processors residing within one system and connected via the system bus. The processors may be single core or multi-core.

- Fig. 2 depicts a processor with a single core. Besides a CPU, a core may include other components like L1 and L2 caches. A basic system layout has a single-core processor; here processor ≡ core.

- Fig. 3 depicts four different system layouts. In a multi-core processor, the collection of cores constitutes the processor. 2 cores. For multi-processor systems, note that processors do not reside in the same chip package, but they belong to a system.

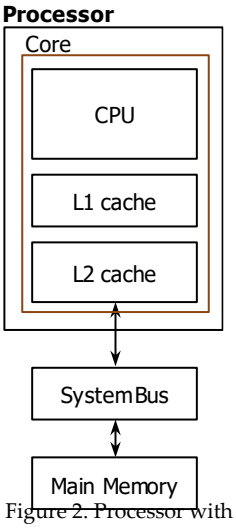

Figure 2. Processor with one core.

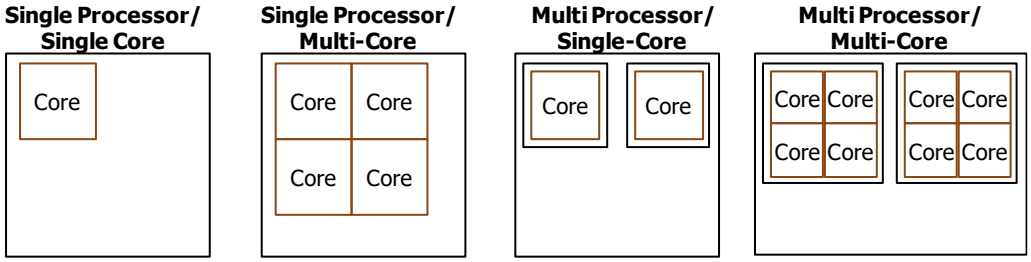

Figure 3. Four system configurations. A processor (single core or multi-core) is contained in a single die or chip package.

ARCHITECTURE

- Multi-core processor: Architectures are varied. For example, Fig. 4 depicts two multi-core architectures that differ in the location of connection between the two processor cores. The architecture in Fig. 4(a) includes two cores, each with L1 and L2 caches, and sharing system bus and connection to memory. The architecture in Fig. 4(b) includes two cores, each with a L1 cache, and sharing the L2 cache, system bus, and connection to memory.
- Multi-core processors can be classified according to the type of cores:
  - ✓ Homogeneous multi-core processor: The cores support the same instruction set architecture, e.g.: Intel® Core™ Duo
  - ✓ Heterogeneous multi-core processor: The cores support different instruction set architectures. It can employ processor cores suited for the specific needs of the application.
  - ✓ Many-core processor: It is a relatively new term. It is comprised of a large number of cores, not necessarily identical, and not necessarily as full featured as the individual cores in a multi-core processor. An example should be a processor that consists of a few powerful processor cores suited to handle general purpose calculations and an array of in-order execution lower power cores that handle specialized processing.

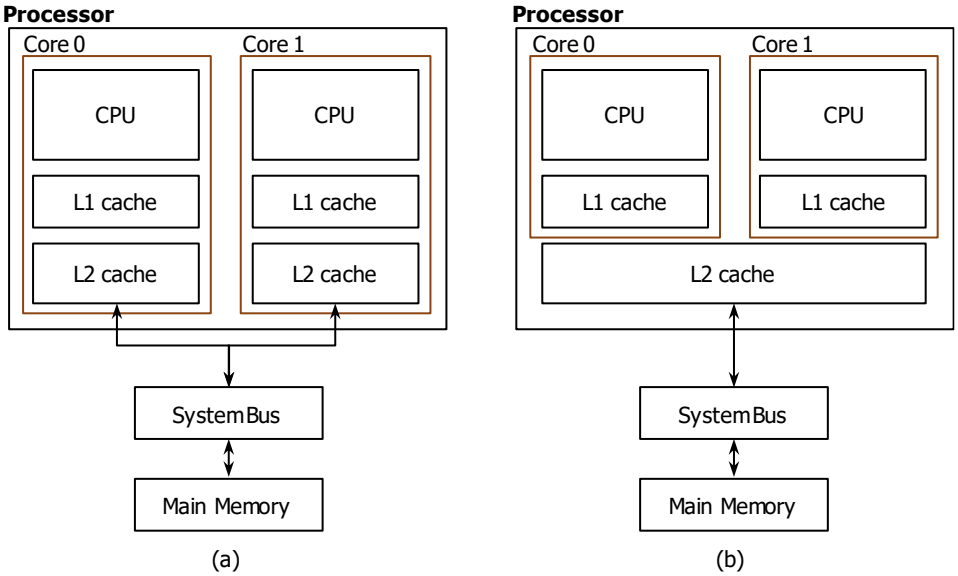

Figure 4. Multi-core processor. (a) Separate L2 cache. (b) Shared L2 cache.

- We can also classify multi-core processors based on how the processor cores appear in relation to the embedded system as a whole. These two types differ in terms of memory access, operating systems, communication between processor cores, as well as load balance and processor affinity.
  - ✓ **Symmetric multi-core processor (SMP)**: The cores have similar views of the system and thus share main memory. As a result, values stored in main memory can be accessed and modified by each of the cores. Typically, one OS controls the cores.
    - Communication between cores is accomplished by sharing memory and relying upon the OS to provide functionality to support synchronization and locking of memory.
    - Load balancing is accomplished by the OS and subject to its scheduling. SMP-enabled OSes support *processor affinity*, which allows the assignment of processes to specific processor cores.
  - ✓ **Asymmetric multi-core processor (AMP)**: The cores view the system differently. Memory regions could be disjoint, implying that two processor cores cannot directly access and modify the same memory location. An embedded system configured as an AMP system may contain multiple OSes that effectively partition the system.
    - Communication between cores is accomplished through message passing facility that serves to marshal data and pass it from one OS to another to use. Or we can provide a memory map that uses specialized hardware to allow two cores to share distinct address spaces between cores.
    - AMP system is statically partitioned so the cores assigned to on OS will not be shared by a different OS in the embedded system.

- Fig. 5 depicts an example of a more detailed system (with other components besides the processor). The processor is the 'brain' of the system taking input from various sources, executing a program written in its instruction set, and then sending the output to the appropriate devices. The ***chipset*** interfaces the different components of the system together (it is the 'nervous system'). The chipset is commonly represented by the memory controller hub (termed *northbridge*), and the I/O controller hub (termed *southbridge*).

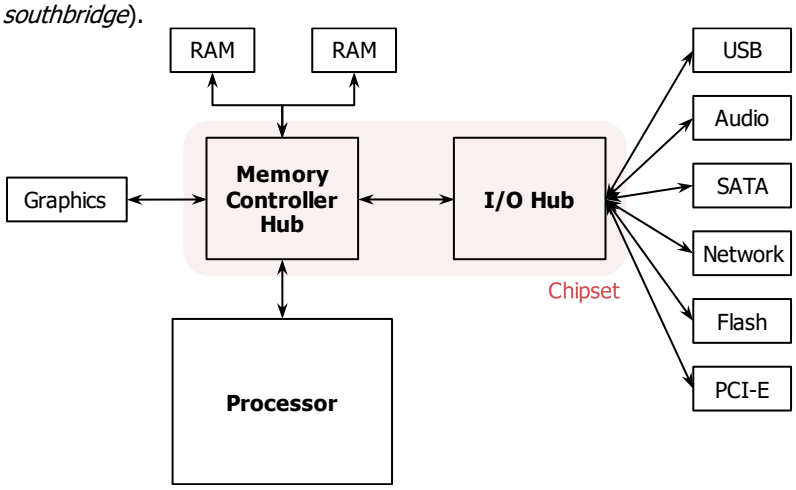

Figure 5. System including a processor, chipset, memory, and I/O interfaces. The memory controller hub provides high speed connections to banks of memory. The I/O hub controls access to relatively slower devices.

MULTI-CORE PROCESSORS IN EMBEDDED SYSTEMS

- Use of multi-core processor in the embedded market segment offers unique challenges (different from those faced in the desktop and server market):
  - ✓ Not all embedded OSes and their software tools fully support multi-core processors.
  - ✓ Many legacy embedded applications do not port easily to modern multi-core processors. They were developed with older 8-bit/16-bit microprocessors and coded with byte ordering assumptions.
  - ✓ Embedded designs with heterogeneous cores increase the programming and communication complexity.

BENEFITS

- We can apply more computer resources to a particular problem. This is advantageous in terms of improved turnaround time or increasing throughput. For example, let's consider processing a transaction in a point-of-sales (POS) system:
  - ✓ The time it takes to process a transaction may be improved by taking advantage of multi-core processors: while one core is updating the terminal display, another core could be tasked with processing the serial input.
  - ✓ Increase in throughput can occur in the servers that handle back-end processing of the transactions arriving the various POS terminal. By taking advantage of multi-core processors, any one server could handle a greater number of transactions in the desired response time.
- Scaling improvements derived from multi-core processors. *Scalability* is the degree to which an application benefits from additional processor cores; more processor cores should result in a corresponding increase in performance. The goal is to employ parallelism to increase the amount of scaling possible in the application.

- Embedded market segments: Wireless Telecommunications Infrastructure, Industrial Control, Enterprise Infrastructure Security, In-vehicle Infotainment, Interactive Clients (check-in kiosks, grocery checkout), Digital Security Surveillance, Storage, Medical (databases, imaging systems).

PERFORMANCE
- This has different meanings depending upon the particular application (e.g.: response time, execution time, tasks completed per unit of time, power utilization).
- Embedded software developer impact on performance: decreasing execution time (latency of the computation task) or improving throughput (measure of how much work is accomplished per time unit).
- The underlying processor is a critical contributor to performance.

EMBEDDED INTEL® ARCHITECTURE PROCESSORS
- *Intel 4004* (1971): World's first processor. Embedded device. 4-bit processor. Addressing capability: 4096 bytes of memory (12-bit address bus). Target application: business calculator.
- *Intel 186* (1982): 16-bit processor like the Intel 8086. It integrates a DMA controller and an interrupt controller. In previous processors, integration (addition of functionality onto an embedded processor) provided in separate ICs. Addressing capability: 1 MB of memory (20-bit address bus). Found in many satellite control systems.
- *Intel 386* (1985): It introduced 32-bit processing (can compute on data 32 bits at a time) and protected memory. 32-bit address bus. The ISA is called **IA-32** (Intel Architecture, 32-bit). Current embedded versions addressing capability: 16 MB and 4 GB. Applications: satellite and robotic control systems. The protected memory model allows the OS to provide:
  - ✓ Memory protection: It restricts the memory that a program can access to the region assigned to it by the OS
  - ✓ Virtual memory: It provides an application with a large address space that appears as contiguous (fragmentation of physical memory is hidden). The OS maps memory addresses used by the program (virtual addresses) into physical addresses in the computer memory. If the physical memory is less than the virtual address space, a hard drive simulates the larger address range.
  - ✓ Task switching (multi-tasking): It can execute multiple processes concurrently.
- *Intel 486* (1989): Similar to Intel 386, but it integrates an FPU and cache. To increase performance, a 5-stage ***instruction pipeline*** is used. Embedded applications: avionic flight systems, point-of-sales (POS) devices, GPS.
- *Intel Pentium* (1993): It incorporated ***superscalar*** execution (duplicating execution resources so that multiple instructions can execute at the same time) to increase performance. First embedded Intel Architecture processor to feature performance monitor counters. This is useful to collect clock cycles of an event, instructions ***retired*** (accepted instructions when branch direction is resolved), cache misses. Embedded applications: POS systems, kiosks, networking.
- *Intel Pentium III* (1999): It introduces larger caches, out-of-order (OOO) execution, Streaming SIMD extensions. Embedded applications: data acquisition, kiosk, gaming, security.
- *Intel Pentium 4* (2000): It employs *hyper-threading* (simultaneous multi-threading: **STM**) technology. With SMT, a processor can appear as multiple processors to the OS. SMT optimizes resource usage by finding work in other threads and processes. Hyper-threading enables instructions associated with two processes to share the execution resources of one processor at the same time. Applications: industrial automation, digital security surveillance (DSS).
- *Intel Pentium M* (2002): Upgrade to Pentium 4 with a goal of decreasing power consumption. Applications: industrial automation, enterprise communications.
- *Dual-Core Intel Xeon® Processors LV and ULV & Dual-Core Intel Xeon® Processor 5100 Series* (2006): 2MB shared cache. Dual processor and dual-core design. Embedded applications: storage area networks, virtual private networks.
- *Intel Core™ 2 Duo Processors for Embedded Computing* (2006): 4MB shared cache, dual-core design. It included the first mainstream **Intel 64** ISA: 64-bit processor (64-bit data/address). Applications: gaming platforms, DSS, medical imaging.
- *Quad-Core Intel Xeon® Processor 5300 Series* (2007): Four processor cores in a package. 8 MB of L2 cache. It is for high performance embedded application: intrusion detection/prevention, services over IP, video on demand.
- *Intel Atom* (2008): Low power processor suitable for embedded systems. Simpler instruction pipeline and reduced power consumption. Superscalar execution, hyperthreading, 64-bit ISA support. No out-of-order execution.

Embedded Trends
- Increased integration and functionality.
- Increase in the number of cores.
- Continued focus on power utilization

PROGRAMMING MODELS FOR MULTI-CORE PROCESSOR DEVELOPMENT
- Cilk Plus: Integrated with C/C++ compilers.
- POSIX Threads (pthreads): It allows a program to control multiple different threads (flows of work) that overlap in time.
- Intel® Threading Building Blocks: Based on a tasking model (tasks are defined rather than explicit threads)
- OpenMP: set of compiler directive, library routines, and environment variables used to specify shared-memory concurrency in C and C++.
- OpenCL: It provides a standard solution for offloading computation to GPUs, CPUs, and accelerators Large range of devices addressed (CPUs, GPUs, embedded processors, FPGAs). Performance portability is an issue.

IA-32 AND INTEL 64 ISA

Fundamental Data Types (Intel nomenclature)

- Byte: 8 bits. Word: 2 bytes. Doubleword: 4 bytes. Quadword: 8 bytes. Double quadword: 16 bytes.
- Byte Order: little-endian, i.e., the low byte (or least significant byte) occupies the lowest address in memory and that address is also the address of the operand.
  - ✓ Note that a specific address refers to one byte, i.e., each byte in memory has its own address.

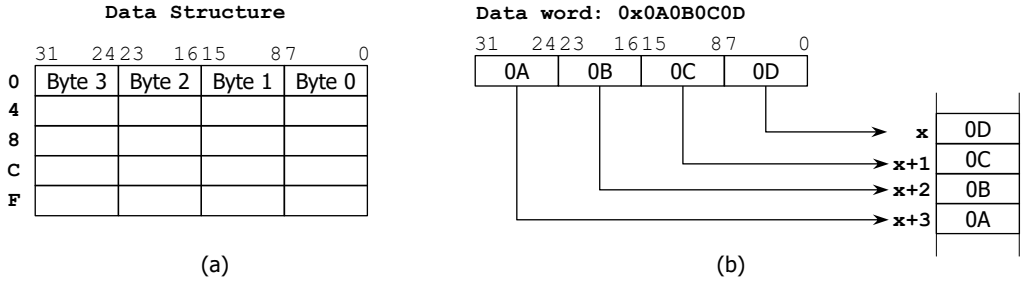

Figure 6. Little endianness. The bytes of a word are numbered starting from the least significant byte.
(a) Illustration of Byte Order. (b) Example with a data word stored in memory: word at address **x** contains `0x0A0B0C0D`

- Numeric Data types:
  - ✓ Integers (unsigned, signed): 8, 16, 32, 64 bits.
- Floating point data types:
  - ✓ Single precision. 32 bits: sign (1), exponent (8), significand (23). C/C++: `float`.
  - ✓ Double precision. 64 bits: sign (1), exponent (11), significand (52). C/C++: `double`.
  - ✓ Double-extended precision. 80 bits: sign (1), exponent (15), significand (64). No hidden 1. C/C++: `long double`.

BASIC EXECUTION ENVIRONMENT

- The execution environment includes memory (the address space), general-purpose data registers, segment registers, the flag register, and the instruction point register.
  - ✓ Any program or task running on a processor is given this set of resources for executing instructions and for storing code, data, and state information.
- The basic execution environment is used jointly by the application programs and the OS running on the processor

Modes of Operation

- The operating mode determines which instruction and architectural features are accessible.
- IA-32: three basic operating modes.
  - ✓ Protected mode: Native state. It features *virtual-8086* mode (ability to directly execute "real-address mode" 8086 software in a protected, multi-tasking environment).
  - ✓ Real-address mode: Implements the programming environment of the Intel 8086 processor with extensions.
  - ✓ System management mode (SMM): Provides an OS a mechanism for implementing platform-specific functions (e.g.: power management, system security). The processor enters SMM when an external SMM interrupt pin (SMI#) is activated or an SMI is received from advanced programmable interrupt controller.
    - In SMM, the processor switches to a separate address space while saving the basic context of the currently running program or task. SMM-specific code may then be executed. Upon returning from SMM, the processor is placed back into its state prior to the SMI.
- Intel-64: It adds IA-32e mode, which has two sub-modes:
  - ✓ Compatibility mode: It allows most legacy 16-bit and 32-bit applications to run without re-compilation under a 64-bit OS.
  - ✓ 64-bit mode: It enables a 64-bit OS to run applications written to access 64-bit address space.
    - 64-bit mode extends the number of general-purpose registers and SIMD extension registers from 8 to 16. General-purpose registers are widened to 64 bits.

IA-32 execution environment

- Address space: Any task or program can access a space of up to 2^{32} bytes (4 GB).
- Basic program execution registers: 8 general-purpose, 6 segment registers, EFLAGS, EIP. These comprise a basic execution environment in which to execute a set of general-purpose instructions. These instructions perform basic integer arithmetic on byte, word, and doubleword integers, handle program flow control, operate on bit and byte strings, and address memory.
- x87 FPU registers: 8 x87 FPU data registers, x87 FPU instruction pointer register, x87 FPU operand (data) pointer register, control register, status register, tag register, opcode register. They provide an execution environment for operating on single, double, and double extended floating-point values; word, doubleword, quadword integers, and BCD values.
  - ✓ x87: floating-point related subset of the x86 architecture instruction set. IA-32, Intel 64: versions of x86 instruction set.
- MMX registers: 8 registers that support execution of SIMD operations on 64-bit packed byte, word, and doubleword integers.
- XMM registers: The 8 XMM data registers and MXCSR register support execution of SIMD operations on 128-bit packed single and double floating-point values and on 128-bit packed byte, word, doubleword, and quadword integers.

- Stack: To support subroutine call and the passing of parameters between subroutines.
- Other resources: I/O ports, control registers, memory management registers (GDTR, IDTR, TR, LDTR), debug registers, memory type range registers, machine specific registers, machine check registers, performance monitoring counters.

Intel 64 execution environment (differences with IA-32)

- Address space: We can access up to 2^{64} bytes.
- Basic program execution registers: 16 64-bit registers. The EIP, EFLAGS (referred to as RFLAGS) are 64 bits.
- XMM registers: 16 registers
- Stack: Stack pointer size is 64 bits.
- Control Registers. Debug Registers: They expand to 64 bits.
- Descriptor table registers: The GDTR (global description table register) and IDTR (interrupt descriptor table register) expand to 10 bytes so they can hold a full 64-bit base address. The LDTR (local descriptor table register) and TR (task register) also expand to hold a full 64-bit base address.

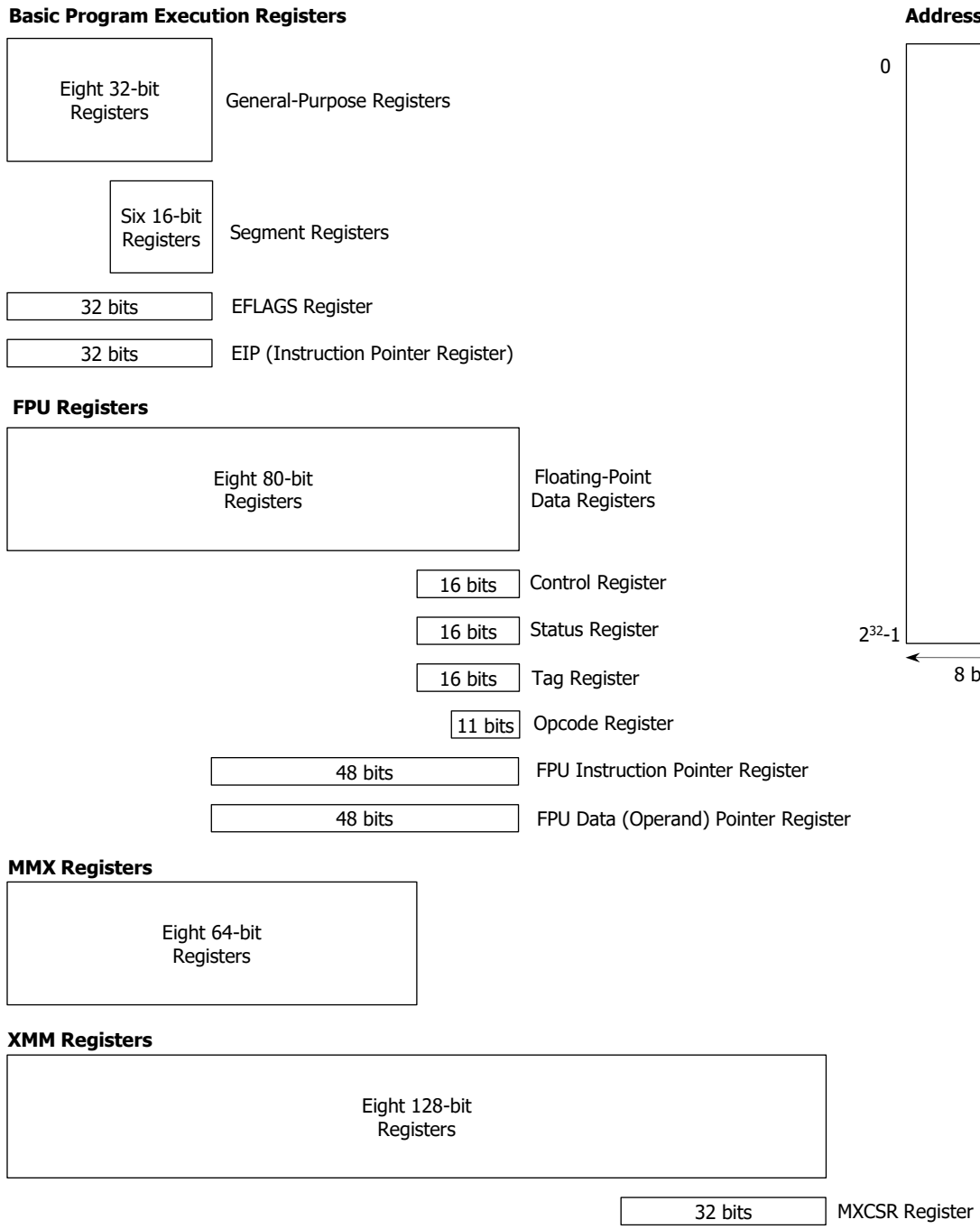

Figure 7. IA-32 Basic Execution Environment

INTEL® ATOM™ PROCESSOR

- Popular brand that is used in netbooks, nettops, embedded applications ranging from health care to advanced robotics, and mobile internet devices.
- Three primary goals: minimize power consumption with respect to previous Intel processors, sufficient performance for a full internet experience, full x86 compatibility.
- No out-of-order processing: too much logic to shuffle instruction, not enough benefit for the power required. No aggressive speculation.
- Support for IA-32 instruction set. Not all Intel Atom processors implement the Intel 64 ISA (Instruction Set Architecture).
- Translation lookaside buffer (TLB): it caches translations of virtual to physical memory references: 2-level TLB with support for page sizes of 4 KB and 4MB.
- Chipsets supporting the Intel Atom have a front side bus (**FSB**) speed from 400 MHz to 667 MHz.
- Intel® Hyper-Threading Technology: When enabled, you see twice the number of CPUs as physical cores.
- Superscalar Pipeline. It can execute and retire two instructions in the same clock cycle. Integer Pipeline: 16 stages. Floating Point pipeline: 19 stages.
- L1, L2 caches.
  - ✓ L1 cache: 32KB instruction cache, 24KB data cache
  - ✓ L2 cache (both instructions and data): 512 KB

PROCESSOR SERIES

- Groupings of Intel Atom® processors that target different categories of use of the processor. They have slightly different combinations of features to optimize for an intended use. The Intel Atom® processor is intended for use across a broad range of application, some of them embedded and some not.
- Table I lists the processor series (the discontinued S series for servers is not listed). Most processors are dual-core, a few (discontinued) were single-core, and there are multicores (4, 8, etc.).

TABLE I. INTEL ATOM® PROCESSOR SERIES. STATUS (DISCONTINUED, LAUNCHED) IS AS OF 08/2020

Series	Target Applications	Key Characteristics	Comments
Z Series	Mobile phones, mobile internet devices (MIDs)	Slower bus speeds than N or D series. No Intel 64 ISA support	Discontinued
N Series	Netbooks, embedded systems	Pairing of Intel Atom with a traditional notebook class chipset. 64-bit ISA support	Only N2600 is available.
D Series	Low-power, low-end desktop systems	Target platforms with higher power utilization than N series	Discontinued
E Series	Embedded: Mobile, portable, and small-scale devices Internet of Things devices.	It integrates processor, graphics, memory controller, I/O interfaces in a SoC	E3845, E3827, E3826, E3825, E3815, E3805 available
C Series	Light-scale-out workloads to the network edge: Network routers, storage, dynamic web serving.	Server SoC: It delivers extra value-add throughput in data transmission, security, and compression acceleration	C3XXX, C2XXX available
P Series	High-density network edge and security solutions	High throughput, low latency on a SoC. Meet reqs. of 5G base transceiver stations. Cores: 8, 12, 16, 24	P5962B, P5942B, P5931B, P5921B available (Q1'2020)
X Series (x3,x5,x7)	Mobile devices with enhanced gaming, HD, built-in security: high-performance smartphones, next-generation tablets	4 cores, except for x5-E3930 (2 cores) x3: extended battery life x5: high performance on the go x7: fast communication and processing, 3D capture and Context Aware	Large variety of components available in x3, x5, x7 versions.

PLATFORM ARCHITECTURES

- This is comprised of the Intel Atom processor and the supporting chips that enable creation of an embedded system.
- Components that typically comprise the key building blocks of a system:
  - ✓ Processor: Provides control and computation facilities for the platform.
  - ✓ Memory Controller (*northbridge*): Provides the interface between the processor and the memory subsystem.
  - ✓ I/O Controller (*southbridge*): Provides the interface between the processor and the I/O devices on the platform.
  - ✓ Graphics Controller: Provides the interface between the processor and the display, as well as hardware support for accelerated 2D/3D output.
  - ✓ Custom Accelerator: Provide hardware support for accelerated processing of application-specific algorithms.
- There are many different arrangements for how the functionality of the components can be provided in a platform. In embedded systems, it is becoming standard to integrate the functionality onto one chip, i.e., System-on-a-Chip (SoC).
  - ✓ Platforms based on Intel Atom usually require a second chip for I/O functionality, so the systems are considered two-chip solutions. This is provided by the Intel® NM10 Express Chipset. The chipset support eight USB 2.0 ports, two 3-Gbps SATA ports, two 32-bit PCIe slots, four PCIe lanes, HD Audio, LAN, WLAN, etc.

- For example, Fig. 8 depicts the System Block diagram for the Intel Atom™ D2000/N2000 series. Unlike the system of Fig. 5, the chipset only implements the I/O functionality, while the memory controller and graphics are integrated in the processor in a single die:
  - ✓ Integrated memory controller: It supports DDR3 protocols with a 64-bit channel accessing 2 DIMMs.
  - ✓ Integrated graphics processing unit (GPU): It includes a graphics engine, video decode, and a display controller. Display ports: eDP/DP (Display Port), HDMI/DVI, LVDS (single channel), VGA.
  - ✓ To interface to the Intel® NM10 Express Chipset, the system includes an integrated DMI (Direct Media Interface).

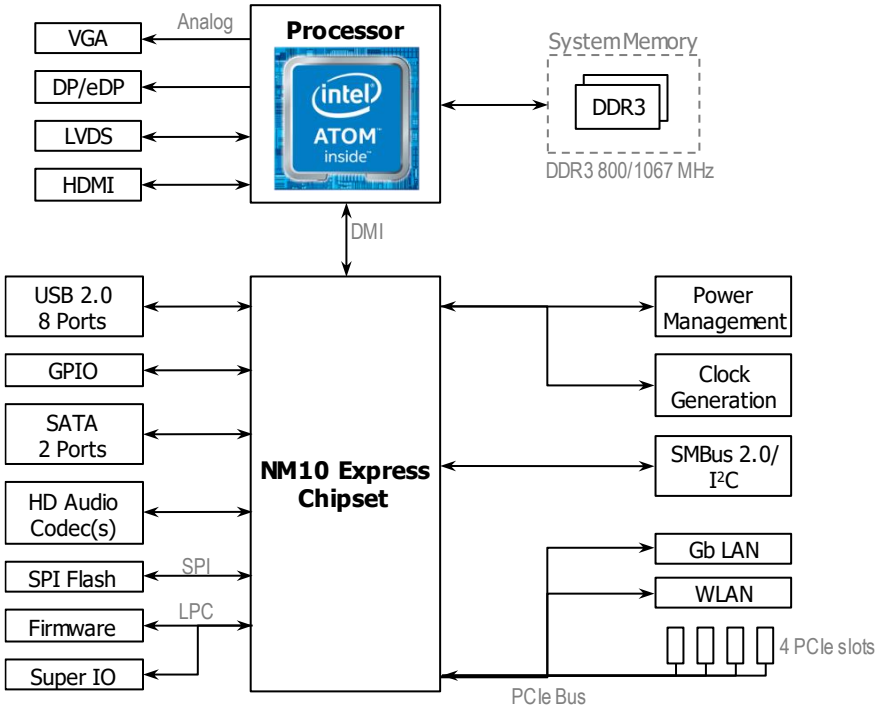

Figure 8. Intel Atom™ D2000/N2000 Series System Block Diagram. Source: *Intel Atom™ Processor D2000 and N2000 Series Datasheet*.

PLATFORM EXAMPLE: TERASIC DE2i-150 FPGA DEVELOPMENT KIT

- This embedded platform fuses together the world of high-performance embedded processing (Intel Atom™ N2600 processor) with the flexibility of an Altera Cyclone IV GX FPGA. The Intel Atom™ processor and the FPGA device are linekd together via two high-speed PCIe lanes. Fig. 9 depicts the block diagram. The components are:

- **Microprocessor Side**:
  - ✓ Intel Atom Dual Core N2600 Processor (launched 2011), 1.6 GHz
    - 2 cores, 4 threads, Base frequency: 1.6 GHz, 512KB L2 cache per processor (it amounts to 1MB)
    - Memory: DDR3 (1 Channel) SO-DIMM SDRAM: **2 GB**
    - 1 MB L2 Cache (512KB per core)
    - Intel® Hyper-Threading Technology (4 execution threads)
    - Instruction Set: 64-bit
    - Integrated Graphics: **VGA, HDMI 1.3a connectors**
    - Debug Interface: XDP Header
  - ✓ Chipset: Intel® NM10 Express (extra chip required for I/O functionality)
    - DMI x2 to Processor
    - Intel® High Definition Audio: **Realtek ALC272VA3-GR Audio Codec**
    - SATA 3 Gb/s: **64 GB SSD**
    - USB Hi-speed 2.0: **4 ports**
    - PCI Express Gen 1
    - Ethernet: **Intel® 82583 Gigabit Ethernet, 10/100/1000 Mb/s RJ45.**
    - WLAN: Intel® Centrino® Wireless-N 135: 802.11 b/g/n, Bluetooth 4.0, Wi-Fi Direct.

- **FPGA side:**
  - ✓ Cyclone IV EPCGX160DF31 FPGA: 149,760 LEs, 720 M9K memory blocks, 6,480 Kbits embedded memory.
  - ✓ FPGA Configuration: JTAG and AS, EPCS64, on-board USB blaster.
  - ✓ Memory devices: 128 MB SDRAM, 4 MB SSRAM, 64 MB Flash with 16-bit mode
  - ✓ SD Card socket: SPI and 4-bit SD mode for SD Card Access

- ✓ Connectors: Ethernet 10/100/1000 Mbps ports, High-Speed Mezzanine Card (HSMC), 40-pin expansion port, VGA connector, DB9 serial connector for RS-232 port.
- ✓ Clock: 3 50 MHz oscillator clock inputs, SMA external connectors.
- ✓ Display: VGA, 16x2 LCD Module
- ✓ Switches and indicates: 18 slides switches, 4 push buttons, 27 LEDs, 8 7-segment displays
- ✓ Others: Infrared remote-control receiver module, TV Decoder and TV-in connector.

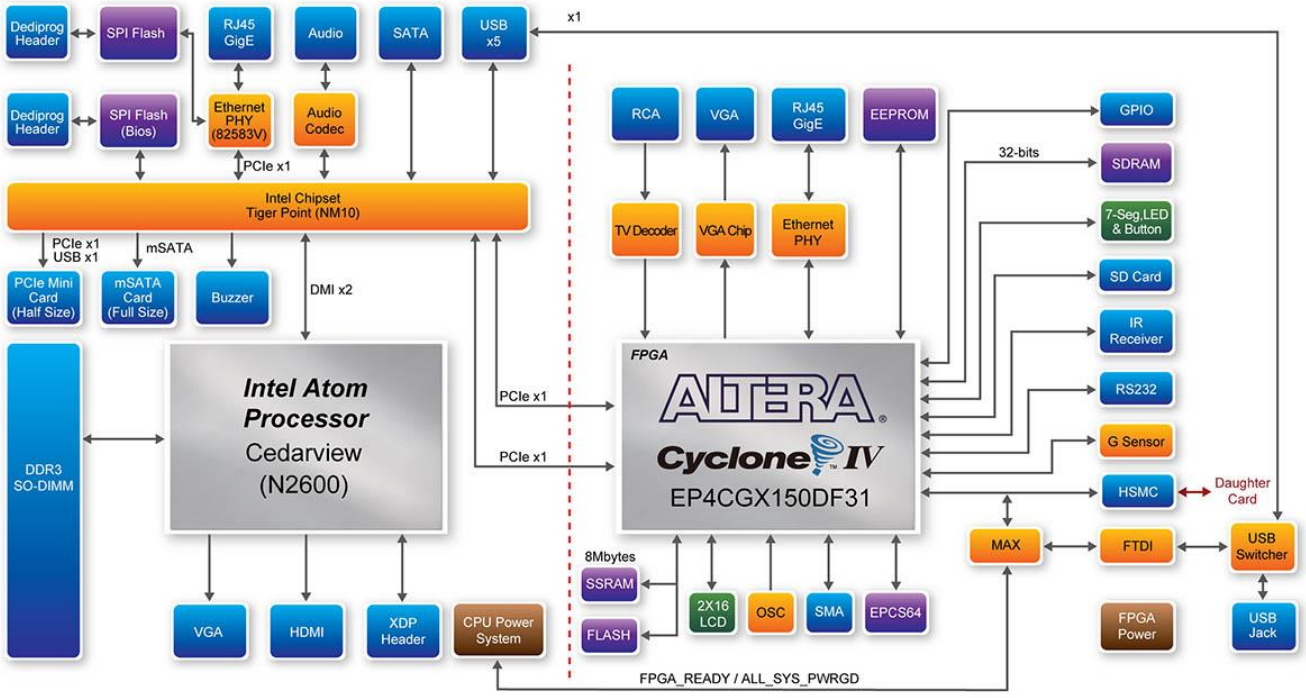

Figure 9. Block Diagram of the DE2i-150 FPGA Development Kit. There is a processor side and a FPGA side.
Source: *Terasic website*

INTEL® ATOM™ PROCESSOR MICROARCHITECTURE

- Set of components that enables it to implement and provide support for the IA-32 and Intel 64 ISA.
- Two-wide superscalar pipeline (it can execute and retire two instructions in the same clock cycle), in-order processing.

Pipeline

- The pipeline is divided into six phases of instruction processing:

Pipeline Phase	Description	Integer: # of Stages	Floating-point: # of stages
Instruction Fetch	Obtain instruction from instruction cache	3	3
Instruction Decode	Breaks instructions into micro-operations, i.e., generates control signals that drive the microarchitecture.	3	3
Instruction Issue	Check and satisfy dependencies, read registers and issue for execution	3	3
Data Access	Generate address if needed and access data cache	3	3
Execute	Execution of the operation	1	7
Write Back	Check for exceptions and commit results	3	1

- Integer Pipeline: 16 stages. Floating-Point pipeline: 19 stages.
  - ✓ Normal pipeline operation: each stage takes one cycle to execute
- Pipelining applies to each stage (we can say that each phase is pipelined). Example: 3 instructions in different stages of the instruction fetch phase (IF1, IF2, IF3) can execute at the same time.

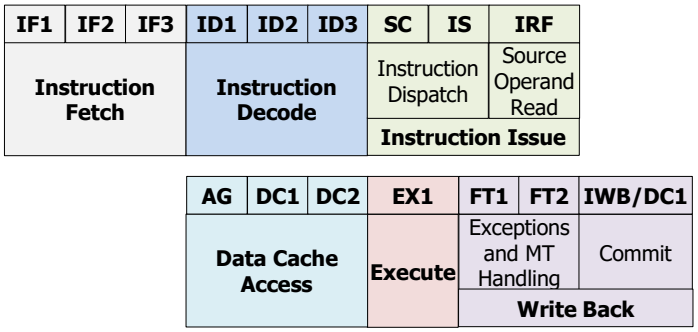

Figure 10. Integer Pipeline: 16 stages

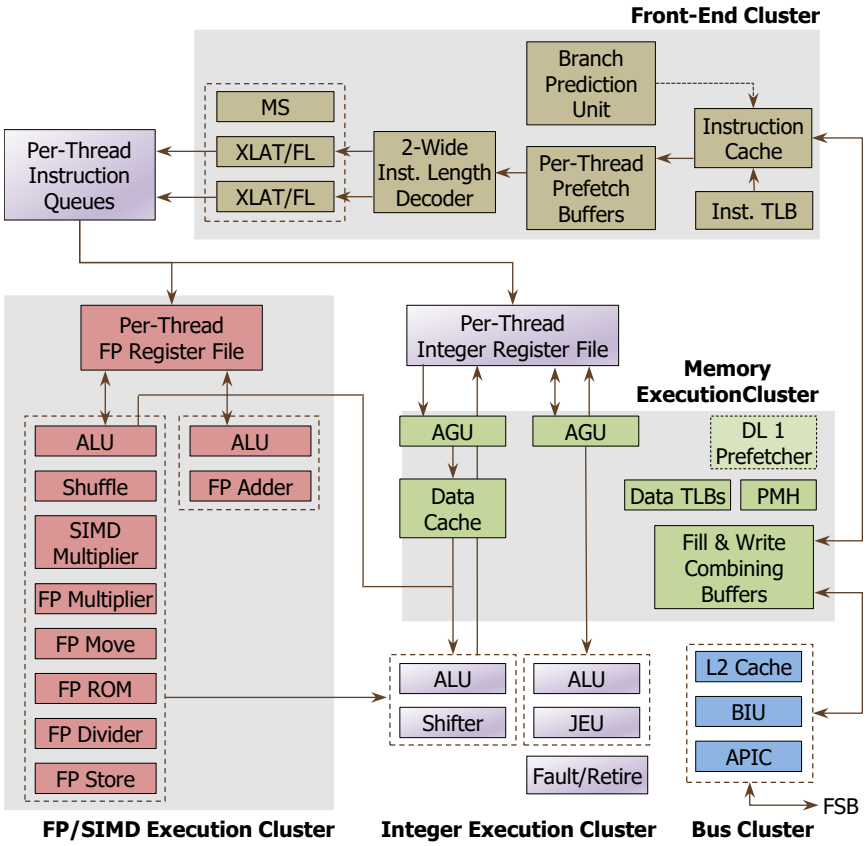

Figure 11. Intel® AtomTM processor - Microarchitecture

Front-End Cluster

- The front-end components are charged with finding and placing the instructions into the execution units.
  - ✓ Branch Prediction Unit: Predicts the target address for branches.
  - ✓ Instruction TLB: Translates virtual to physical addresses.
  - ✓ Instruction Cache: Fast access to recently executed instructions.
  - ✓ Prefetch Buffer: Holds instruction bytes ready for decoding.
  - ✓ Decode. Perform the decode.
  - ✓ Microcode Sequencer (MS): Complex instruction decode.
- The front-end of the microarchitecture performs the instruction fetch and instruction issue phases of the pipeline.
  - ✓ To place an instruction into the pipeline, we first need to obtain the address of the instruction.
  - ✓ Instruction cache: It keeps recently executed instruction closer to the processor. To improve decode speed, it contains predecode bits to demarcate individual instructions.
    - IA-32, Intel 64 ISA: variable-length instructions. Size of the instructions unknown until the instruction has been partially decoded.
  - ✓ Instruction Decoders (2): up to 2 instructions (assumption: boundary of instruction known) can be decoded per cycle.
  - ✓ In each cycle, the front end may transmit two instruction at most to the instruction queue for scheduling.
- Instruction Decode: The IA-32/Intel 64 instruction are decoded into another instruction that drives the microarchitecture. The microcode sequence (MS) decodes the more complex instruction into smaller operations for execution in the pipeline.
  - ✓ In some cases, the decoder can only decode one instruction per cycle (x87 floating point instructions, branch instruction).
- Instruction Queue: holds instructions until they are ready to execute in the Memory execution cluster, the integer execution cluster, or the FP/SIMD execution cluster.

Memory Execution Cluster
- Provides functionality for generating addresses and data. Components:
  - ✓ Address Generation Unit (AGU): Generates data address: base address + scale + offset
  - ✓ Data TLB. Translates virtual addresses to physical addresses
  - ✓ Data Cache. Holds recently accessed data.
  - ✓ Prefetcher. Predicts future access and fetches by analyzing previous accesses
  - ✓ Write-Combining Buffers. Allows grouping of individual write operations before being sent on to the L2 cache; enable more efficient memory bandwidth utilization.
- In an optimal case, data is in the L1 cache. If data is not in L1 cache, an L2 cache request is made.
  - ✓ Data Prefetcher: It analyzes historical patterns and attempts to fetch future data in advance.

Integer Execution Cluster
- 2 ALUs, enabling joint execution of instructions.

FP/SIMD Execution Cluster
- It executes x87, SIMD, and integer multiply instructions.
- 2 ALUs. It supports limited combinations of dual execution.

Bus Cluster
- It is the connection from the processor to the memory subsystem
- It includes a 512 KB, eight-way L2 cache.
- It contains the bus interface unit (BIU) and Advanced programmable interrupt controller (APIC).