

UNIT 5

MULTICORE ARCHITECTURES

SOFTWARE AND HARDWARE MULTITHREADING:

MULTITHREADING:

- Modern OS commonly support threads (lightweight processes), and many vendors ship shared memory multiprocessors.
- Multithreading is one method for sharing the work of a single application across multiple processors with shared memory.
- Multithreading computer central processing units have hardware support to efficiently execute multiple threads.
- These are distinguished from multiprocessing systems (such as multi-core systems) in that the threads have to share the resources of a single core: the computing units, the CPU caches and the translation look aside buffer (TLB).
- The multiprocessing systems include multiple complete processing units, multithreading aims to increase utilization of a single core by using thread-level as well as instruction-level parallelism.
- As the two techniques are complementary, they are sometimes combined in systems with multiple multithreading CPUs and in CPUs with multiple multithreading cores.

ADVANTAGES:

- If a thread gets a lot of cache misses, the other thread(s) can continue, taking advantage of the unused computing resources, which thus can lead to faster overall execution, as these resources would have been idle if only a single thread was executed.
- If a thread cannot use all the computing resources of the CPU (because instructions depend on each other's result), running another thread can avoid leaving these idle.
- If several threads work on the same set of data, they can actually share their cache, leading to better cache usage or synchronization on its values.

DISADVANTAGES:

- Multiple threads can interfere with each other when sharing hardware resources such as caches or translation look aside buffers (TLBs).
- Execution times of a single thread are not improved but can be degraded, even when only one thread is executing. This is due to slower frequencies and/or additional pipeline stages that are necessary to accommodate thread-switching hardware.
- Hardware support for multithreading is more visible to software, thus requiring more changes to both application programs and operating systems than multiprocessing.

TYPES OF MULTITHREADING:

1. Software multithreading
2. Hardware multithreading

SOFTWARE MULTITHREADING:

- Multithreading without hardware support for storing state (PC, registers, etc.) for multiple threads simultaneously; often a thread / CPU, reduced utilization
- There are two levels of thread
 1. User level(for user thread)
 2. Kernel level(for kernel thread)

USER LEVEL THREADS:

- User threads are supported above the kernel and are implemented by a thread library at the user level.
- The library provides support for thread creation, scheduling, and management with no support from the kernel.
- Because the kernel is unaware of user-level threads, all thread creation and scheduling are done in user space without the need for kernel intervention.
- User-level threads are generally fast to create and manage User-thread libraries include POSIX Pthreads, Mach C-threads, and Solaris 2 UI-threads.

KERNAL LEVEL THREADS:

- Kernel threads are supported directly by the operating system.
- The kernel performs thread creation, scheduling, and management in kernel space.
- Because thread management is done by the operating system, kernel threads are generally slower to create and manage than are user threads.
- Most operating systems-including Windows NT, Windows 2000, Solaris 2, BeOS, and Tru64 UNIX (formerly Digital UNIX)-support kernel threads.

MULTI-THREADING MODELS:

- There are three models for thread libraries, each with its own trade-offs
 1. Many threads on one LWP (many-to-one)
 2. One thread per LWP (one-to-one)
 3. Many threads on many LWPs (many-to-many)

MANY-TO-ONE:

- The many-to-one model maps many user-level threads to one kernel thread.
- **Advantages:**
 1. Totally portable
 2. More efficient
- **Disadvantages:**
 1. It cannot take advantage of parallelism
 2. The entire process is block if a thread makes a blocking system call
 3. Mainly used in language systems, portable libraries like Solaris 2

ONE-TO-ONE:

- The one-to-one model maps each user thread to a kernel thread.
- **Advantages:**
 1. It allows parallelism Provide more concurrency
- **Disadvantages:**
 1. Each user thread requires corresponding kernel thread limiting the number of total threads Used in Linux Threads and other systems like Windows 2000, Windows NT

MANY-TO-MANY:

- The many-to-many model multiplexes many user-level threads to a smaller or equal number of kernel threads.
- **Advantages:**
 1. Can create as many user thread as necessary
 2. Allows parallelism
- **Disadvantages:**
 1. kernel thread can burden the performance
 2. Used in the Solaris implementation of Pthreads (and several other Unix implementations)

HARDWARE MULTITHREADING:

- Hardware-level support for storing state for multiple threads, permitting fast context switches
- There are two types of hardware multithreading
 1. Block or Coarse-grained multi-threading
 2. Fine-grained or Interleaved multi-threading

COARSE-GRAIN MULTITHREADING:

- The simplest type of multi-threading occurs when one thread runs until it is blocked by an event that normally would create a long latency stall.
- Such a stall might be a cache-miss that has to access off-chip memory, which might take hundreds of CPU cycles for the data to return.
- Instead of waiting for the stall to resolve, a threaded processor would switch execution to another thread that was ready to run.
- Only when the data for the previous thread had arrived, would the previous thread be placed back on the list of ready-to-run threads.
- For example:
 - ✓ Cycle i : instruction j from thread A is issued
 - ✓ Cycle $i+1$: instruction $j+1$ from thread A is issued
 - ✓ Cycle $i+2$: instruction $j+2$ from thread A is issued, load instruction which misses in all caches
 - ✓ Cycle $i+3$: thread scheduler invoked, switches to thread B
 - ✓ Cycle $i+4$: instruction k from thread B is issued
 - ✓ Cycle $i+5$: instruction $k+1$ from thread B is issued
- Conceptually, it is similar to cooperative multi-tasking used in real-time operating systems in which tasks voluntarily give up execution time when they need to wait upon some type of the event.

HARDWARE COSTS:

- The goal of multi-threading hardware support is to allow quick switching between a blocked thread and another thread ready to run.
- To achieve this goal, the hardware cost is to replicate the program visible registers as well as some processor control registers (such as the program counter).
- Switching from one thread to another thread means the hardware switches from using one register set to another.

EXAMPLES:

- Many families of microcontrollers and embedded processors have multiple register banks to allow quick context switching for interrupts.
- Such schemes can be considered a type of block multithreading among the user program thread and the interrupt threads.
- Such additional hardware has these benefits:
- The thread switch can be done in one CPU cycle.
- It appears to each thread that it is executing alone and not sharing any hardware resources with any other threads.
- This minimizes the amount of software changes needed within the application as well as the operating system to support multithreading.
- In order to switch efficiently between active threads, each active thread needs to have its own register set.
- For example, to quickly switch between two threads, the register hardware needs to be instantiated twice.

FINE-GRAINED MULTITHREADING (or) INTERLEAVED MULTI-THREADING:

- Cycle $i+1$: an instruction from thread B is issued
- Cycle $i+2$: an instruction from thread C is issued
- The purpose of this type of multithreading is to remove all data dependency stalls from the execution pipeline.
- Since one thread is relatively independent from other threads, there's less chance of one instruction in one pipe stage needing an output from an older instruction in the pipeline.
- Conceptually, it is similar to pre-emptive multi-tasking used in operating systems.
- One can make the analogy that the time-slice given to each active thread is one CPU cycle.

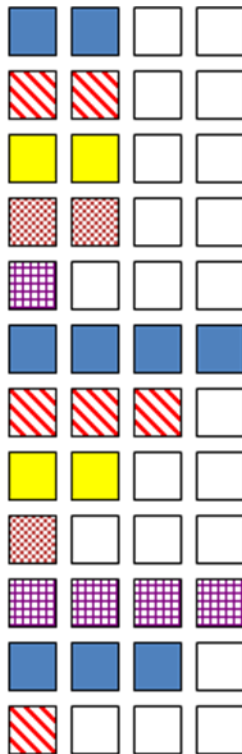
HARDWARE COSTS:

- In addition to the hardware costs discussed in the Block type of multithreading, interleaved multithreading has an additional cost of each pipeline stage tracking the thread ID of the instruction it is processing.

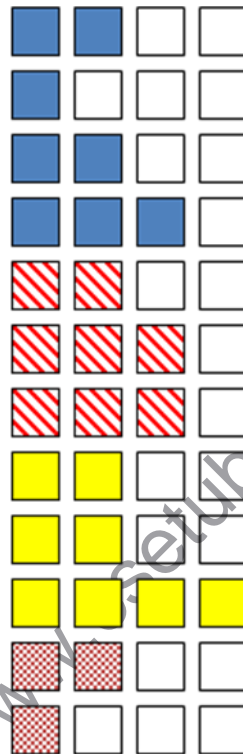
- Also, since there are more threads being executed concurrently in the pipeline, shared resources such as caches and TLBs need to be larger to avoid thrashing between the different threads.


MULTITHREADED CATEGORIES:

Fine - Grained



Coarse - grained



 Thread 1  Thread 2  Thread 3  Thread 4  Thread 5  Thread 6

SMT AND CMP ARCHITECTURES:

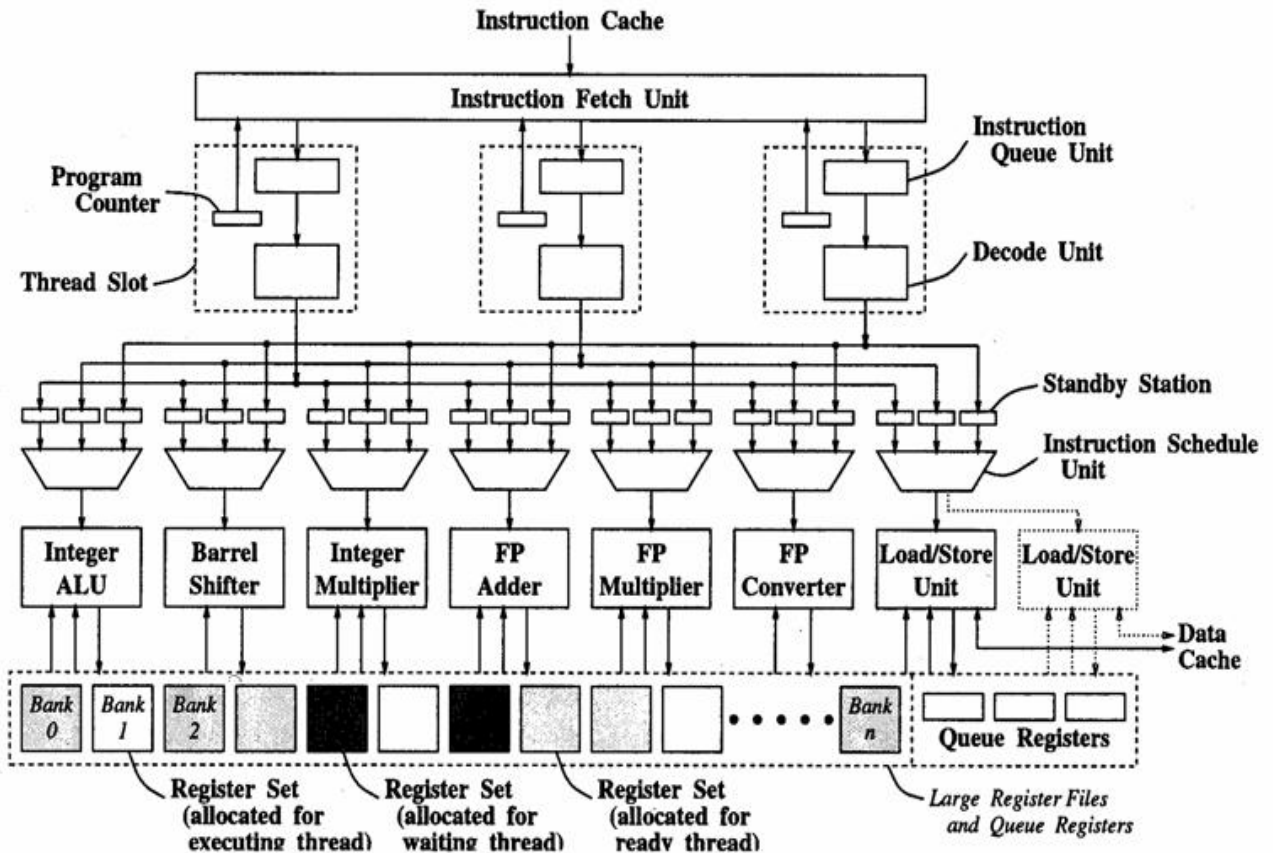
- The two alternative micro architectures that exploit multiple threads of control:
 1. Simultaneous Multithreading (SMT)
 2. Chip Multiprocessors (CMP)
- Simultaneous multithreading (SMT), is a technique that permits multiple independent threads to issue instructions to a superscalar's functional units in a single cycle.
- This promotes much higher utilization of the processor's execution resources and provides latency tolerance in case a thread stalls due to cache misses or data dependencies.
- When multiple threads are not available, however, the SMT simple looks like a conventional wide-issue processor.
- CMPs use relatively simple single-thread processor cores to exploit only moderate amounts of parallelism within any one thread, while executing multiple threads in parallel across multiple processor cores.
- If an application cannot be effectively decomposed into threads, MPs will be underutilized.

SMT ARCHITECTURES:

- Simultaneous multithreading (SMT) allows multiple threads to compete for and share available processor resources every cycle.
- When executing parallel applications, SMT has the ability to use TLP and ILP interchangeably.
- When a program has only a single thread (i.e. it lacks TLP) all of the SMT processor's resources can be dedicated to that thread.
- When more TLP exists, this parallelism can compensate for a lack of per thread ILP.
- Thus SMT most effectively utilizes the available resources to achieve the goals of greater throughput and significant program speedups.
- To keep the processor's execution unit busy SMT processors feature advance branch prediction, register renaming, out-of-order instruction issue, and non blocking caches which result in the processor having rename buffers, issue queues, and register files.
- SMT architectures result in three major hardware design problems:
 1. Their area increases quadratically with the core's complexity. Also the complexity increases the design time.
 2. They can require longer cycle times. Long, high capacitance I/O wires span the large buffers, queues, and register files. Extensive use of multiplexers and crossbars to interconnect these units adds more capacitance. Delays associated with these wires will probably dominate the delay along the CPU's critical path. Complicated logic requires more clock cycles to keep clock period short. Making the pipeline deeper increases the branch mispredict penalty.

- The CPU cores are complicated and composed of many closely interconnected components. As a result, design and verification costs will increase since they must be designed and verified as single, large units.

BASIC SMT ARCHITECTURES:



- As shown in figure the processor consists of several instruction queue unit and decode unit pairs called thread slots.
- The instruction fetch unit and the functional units are shared between the different thread slots.
- The instruction fetch unit has a buffer of size such that if the instruction fetching takes C cycles and there are S threads then $S \times C$ instructions are fetched at one time in C cycles.
- This is done for each thread so that effectively we get $S \times C$ instructions for each thread in $S \times C$ cycles resulting in there being an instruction present in the buffer for each cycle for every thread.

- The decode unit then decodes the instructions and each instruction is checked for dependencies using a score boarding technique.
- The instruction is issued only if there is no dependency, otherwise the issuing is interlocked.
- The issued instructions are dynamically scheduled by the instruction schedule units.
- The use of the functional units is arbitrated accordingly by the schedule unit and if the functional unit is not being used by some other instruction of higher priority according to the arbitration logic, then the instruction is immediately sent to the functional unit for execution.
- The arbitration logic for this architecture basically implements a rotation priority mechanism.
- However if the instruction cannot be sent to the instruction unit immediately it is stored in the standby station of depth one till it can be executed.
- The register file is common to all the threads.
- However it is divided into banks each of which is used as a full register set private to a thread.
- This ensures that each thread accesses only the registers bound to it and not any other.
- The queue registers are used for the communication between the threads at the register transfer level.
- The status of each thread including its registers, etc is collectively called a context Frame.
- The processor always has more context frames than it has threads.
- Thus whenever required, the context is rapidly switched by changing the logical binding between a thread and the context frame without reference to the memory.
- This results in very little overhead while changing contexts.
- The context frame also contains an access requirement buffer to handle loads/stores.
- The outstanding memory accesses are also saved as a part of the context frame when context switching is carried out and this enables us to overcome the effect of memory access latencies by switching contexts.
- Thus this architecture allows us to use instructions from different threads in the same clock cycle thereby reducing the effects of insufficient ILP in a particular thread as well as to switch between contexts.
- This models assume 10 functional units and a capability of issue of 8 instructions per cycle.
- These include
 - ✓ **Full Simultaneous Issue:** Here all the eight threads active can compete for each issue slot. Thus in one case it may even happen that all the eight instructions being simultaneously executed from the same thread.

- ✓ **Single Issue, Dual Issue, Four Issue:** Here each thread can issue one, two and four instructions respectively per cycle making the number of threads respectively as eight, four and two.
 - ✓ **Limited Connection:** Here each context is directly connected to only one instance of each type of a functional unit. Thus there is less flexibility in regards that a particular thread will be able to compete for use of only one instance of a particular functional unit.
- Thus usually the preferred choice for SMT processors is either the 4-issue or the 2-issue model

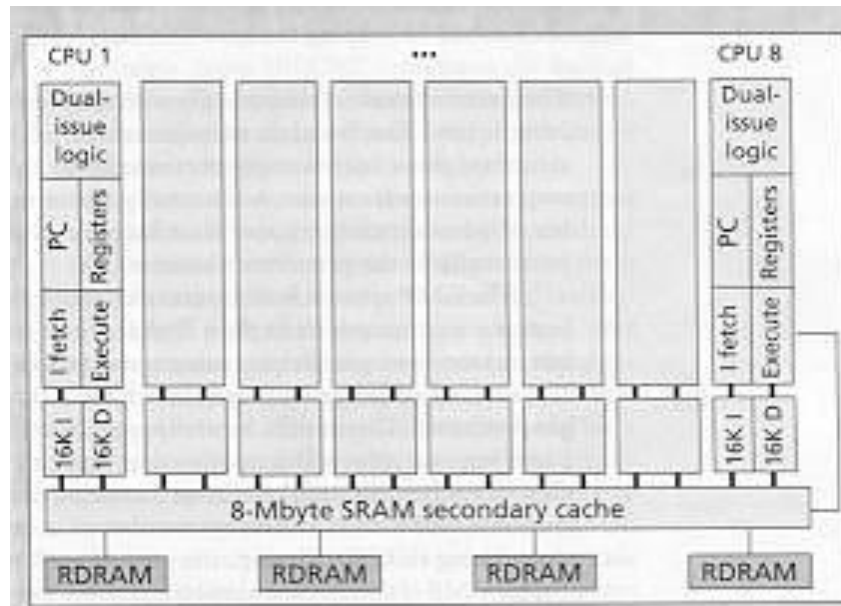
PERFORMANCE OF SMT:

- Single thread performance is likely to go down (caches, branch predictors, registers, etc. are shared) – this effect can be mitigated by trying to prioritize one thread
- While fetching instructions, thread priority can dramatically influence total throughput – a widely accepted heuristic (ICOUNT): fetch such that each thread has an equal share of processor resources
- With eight threads in a processor with many resources, SMT yields throughput improvements of roughly 2-4
- Alpha 21464 and Intel Pentium 4 are examples of SMT

CHIP MULTIPROCESSOR (CMP):

- CMP architecture will be easiest to implement using the billion-transistor
- CMPs use relatively simple single-threaded processor cores to exploit only moderate amount of parallelism within any one thread, while executing multiple threads in parallel across multiple processor cores.
- As CMPs are partitioned into individual processing cores, it limits the effect of interconnect delays, which are becoming much slower than transistor gate delays.
- Thus the design simplicity allows for a faster clock in each of the processing unit as well eases the time-consuming design validation phase.
- Finally, the CMP approach also results in better utilization of the silicon space.
- By avoiding the extra logic devoted to the centralized architecture, a CMP allows the chip to have a higher overall issue bandwidth when compared to a conventional superscalar implemented on the same die area.

BASIC CMP ARCHITECTURE:



- The basic CMP architecture consists of narrow issue multiple cores.
- The architecture shown in figure has eight small 2-issue superscalar processors.
- The cores are completely independent and tightly integrated with their individual pairs of caches. This design is of clustered form leading to simple, high-frequency design for the primary cache system.
- The small cache size and tight connection to these caches allow single cycle Access.
- The rest of the memory system remains essentially unchanged, except that the secondary cache controller adds two extra cycles of secondary cache latency to handle requests from multiple processors.
- To make a shared memory multiprocessor, the data caches could be made write through, or a MESI (modified, exclusive, shared, and invalid) cache coherence protocol could be established between the primary data caches.
- In this way, designers can implement a small-scale multiprocessor with very low interprocessor communication latency.
- To provide enough off-chip memory bandwidth for our high-performance processors, main memory could be designed as multiple banks of Rambus DRAMs (RDRAMs), attached via multiple Rambus channels to each processor.

CMP ARCHITECTURE WITH SPECULATIVE MULTITHREADING:

- From a software perspective, the CMP is an ideal platform to run a multiprogrammed workload or a multithreaded application.

- However, if the CMP is to be fully accepted, it must also be able to give good performance when running sequential applications.
- Since parallelizing compilers, are successful only for a restricted class of applications, typically numerical ones, and the CMP approach would not be able to handle a large class of general-purpose sequential applications.
- To address this problem, speculation may be used.
- In speculative multithreading, the speculative threads in the application needs to be identified either at compile time or completely at runtime with hardware support.
- The different threads are executed in parallel speculatively.
- Added hardware or software support enables detection of and recovery from dependence violations.

PERFORMANCE OF CMP:

- CMP's are now the only way to build high performance microprocessors , for a variety of reasons:
 1. Large uniprocessors are no longer scaling in performance, because it is only possible to extract a limited amount of parallelism from a typical instruction stream.
 2. Cannot simply ratchet up the clock speed on today's processors, or the power dissipation will become prohibitive.
 3. CMT processors support many h/w strands through efficient sharing of on-chip resources such as pipelines, caches and predictors.
 4. CMT processors are a good match for server workloads, which have high levels of TLP and relatively low levels of ILP.

COMPARISION OF SMT AND CMP:

- The performance race between SMT and CMP is not yet decided.
- CMP is easier to implement, but only SMT has the ability to hide latencies.
- A functional partitioning is not exactly reached within a SMT processor due to the centralized instruction issue.
 1. A separation of the thread queues is a possible solution, although it does not remove the central instruction issue.
 2. A combination of simultaneous multithreading with the CMP may be superior.
- Research: combine SMT or CMP organization with the ability to create threads with compiler support of fully dynamically out of a single thread.
 1. Thread-level speculation
 2. Close to multiscalar

INTEL MULTI-CORE ARCHITECTURE:

- Intel continues to focus its near- and long-term efforts on enhancing the overall computing platform to deliver greater value and functionality to personal computer (PC) users.
- About three years ago Intel sharpened its continued focus on platform-level improvements and began providing fundamental technologies and features in a move to bring more benefits to users.
- The Intel vision of a balanced platform is moving beyond gigahertz (GHz) and expanding the company's focus on the fundamental technologies and features for delivering greater value and functionality.
- Intel has realigned its strategy and moved resources away from pure GHz-oriented projects—the result is the company embracing multi-core architecture.
- Intel also continued to invest in its manufacturing capacity during the 2000 economic downturn.
- To ensure that it has the capacity to deliver processors, including multi-core processors, in high volume and at affordable price points.
- Multi-core processor capability is central to the Intel platform centric approach. By enabling enhanced performance, reduced power consumption and more efficient simultaneous processing of multiple tasks, multi-core processors promise to improve the user experience in home and business environments
- Multi-core processor capability is central to the Intel platform centric approach.
- By enabling enhanced performance, reduced power consumption and more efficient simultaneous processing of multiple tasks, multi-core processors promise to improve the user experience in home and business environments.

INTEL MULTI-CORE PROCESSOR ARCHITECTURE:

- Multi-core processor architecture entails silicon design engineers placing two or more Intel® Pentium processor-based “execution cores,” or computational engines, within a single processor.
- This multi-core processor plugs directly into a single processor socket, but the operating system perceives each of its execution cores as a discrete logical processor with all the associated execution resources.
- The idea behind this implementation of the chip's internal architecture is in essence a “divide and conquer” strategy.
- In other words, by divvying up the computational work performed by the single Pentium microprocessor core in traditional microprocessors and spreading it over multiple execution cores, a multi-core processor can perform more work within a given clock cycle.

- Thus, it is designed to deliver a better overall user experience.
- To enable this improvement, the software running on the platform must be written such that it can spread its workload across multiple execution cores.
- This functionality is called thread-level parallelism or “threading”
- Applications and operating systems (such as Microsoft Windows* XP) that are written to support it are referred to as “threaded” or “multi-threaded.”
- A processor equipped with thread-level parallelism can execute completely separate threads of code.
- This can mean one thread running from an application and a second thread running from an operating system, or parallel threads running from within a single application.
- Multimedia applications are especially conducive to thread-level parallelism because many of their operations can run in parallel.
- As software developers continue to design more threaded applications that capitalize on this architecture, multi-core processors can be expected to provide new and innovative benefits for PC users, at home and at work.
- Multi-core capability can also enhance the user experience in multitasking environments, namely, where a number of foreground applications run concurrently with a number of background applications such as virus protection and security, wireless, management, compression, encryption and synchronization.
- Like other hardware-enhanced threaded capabilities advanced at Intel, multi-core capability reflects a shift to parallel processing—a concept originally conceived in the supercomputing world.
- For example, Hyper-Threading Technology (HT Technology), introduced by Intel in 2002, enables processors to execute tasks in parallel by weaving together multiple “threads” in a single-core processor.
- But whereas HT Technology is limited to a single core’s using existing execution resources more efficiently to better enable threading, multi-core capability provides two or more complete sets of execution resources to increase compute throughput.
- In a technical nutshell, Intel believes multi-core processing could support several key capabilities that will enhance the user experience, including the number of PC tasks a user can do at one time, performing multiple bandwidth-intensive activities and increasing the number of users utilizing the same PC.

A FUNDAMENTAL THEOREM OF MULTI-CORE PROCESSORS:

- Multi-core processors take advantage of a fundamental relationship between power and frequency.
- By incorporating multiple cores, each core is able to run at a lower frequency, dividing among them the power normally given to a single core.
- The result is a big performance increase over a single core processor.

- The following illustration based on used workloads illustrates this key advantage.
- Increasing clock frequency by 20 percent to a single core delivers a 13 percent performance gain, but requires 73 percent greater power. Conversely, decreasing clock frequency by 20 percent reduces power usage by 49 percent, but results in just a 13 percent performance loss.

MULTI-CORE ENERGY-EFFICIENT PERFORMANCE:

- Here the second core on the under clocked is added.
- This result in a dual-core processor that at 20 percent reduced clock frequency effectively delivers 73 percent more performance while using approximately the same power as a single-core processor at maximum frequency.

FUNDAMENTAL RELATIONSHIP BETWEEN FREQUENCY AND POWER:

- This fundamental relationship between power and frequency can be effectively used to multiply the number of cores from two to four, and then eight and more, to deliver continuous increases in performance without increasing power usage.
- To do this though, there is much advancement that must be made that are only achievable by a company like Intel.
- These include:
 - ✓ **Continuous advances in silicon process technology** from 65 nm to 45 nm and to 32 nm to increase transistor density. In addition, Intel is committed to continuing to deliver superior energy-efficient performance transistors.
 - ✓ **Enhancing the performance of each core and optimizing** it for multi-core through the introduction of new advanced micro architectures about every two years.
 - ✓ **Improving the memory subsystem and optimizing data access** in ways that ensure data can be used as fast as possible among all cores. This minimizes latency and improves efficiency and speed.
 - ✓ **Optimizing the interconnect fabric** that connects the cores to improve performance between cores and memory units.
 - ✓ **Optimizing and expanding the instruction set** to enhance the capabilities of Intel® architecture and enable the industry to deliver advanced applications with greater performance and lower power requirements. Some of these instructions can effectively dedicate a core to deliver specific capabilities.
 - ✓ **Continuing to grow Intel's commitment** to developing multi-core software tools and programs by working closely with developers, independent software vendors (ISVs), operating system vendors (OSVs) and academia. Through these efforts,

Intel enables the industry to develop software that runs faster and better on our energy-efficient performance multi-core platforms.

INTEL ACHIEVEMENTS IN DUAL-CORE PROCESSING:

- Intel first implemented multi-core processing through dual-core processors across all key sectors (desktop, workstation, mobile, and mainstream server).
- In accordance with our new cadence for process technology and micro architecture, Intel's second generation of dual core processors, released in the third quarter of 2006, uses the new Intel Core micro architecture.
- These products include Intel® Core™ 2 Duo desktop and mobile processors and Dual-Core Intel® Xeon® 5100 processor 5100 series for dual-processor servers.
- By transitioning the majority of our volume products to Intel Core micro architecture based dual-core processors, Intel took the lead in performance and energy efficiency in most of these product segments.
- According to benchmark tests:
 - ✓ **The Dual-Core Intel Xeon 5100 server processor** delivers up to 135 percent performance improvements² and up to a 40 percent reduction³ in energy consumption over previous Intel server products.
 - ✓ **The Intel Core 2 Duo desktop processor delivers** up to a 40 percent improvement in performance and up to a 40 percent reduction in power as compared to today's high-end Intel® Pentium® D processor 960.4
 - ✓ **The Intel Core 2 Duo mobile processor** delivers greater than 2X CPU performance⁵ and up to a 28 percent power reduction⁶ with new Intel® Centrino® Duo mobile technology laptops based on the Intel Core 2 Duo processor as compared to previous-generation Intel® Centrino® mobile technology based laptops.

INTEL-MULTICORE ARCHITECTURE:

- Intel Turbo Boost Tech.
- Intel Hyper Threading Tech.
- Intel Core Micro architecture.
- Intel Advanced Smart Cache.
- Intel Smart Memory Access.

BENEFITS:

- Multi-core performance.
- Dynamic scalability.
- Design and performance scalability
- Intelligent performance on-demand

- Increased performance on Highly-threaded apps.
- Scalable shared memory.
- Multi-level shared cache.

HETEROGENEOUS MULTI-CORE PROCESSORS:

- A **multi-core processor** is a single computing component with two or more independent actual central processing units (called "cores"), which are the units that read and execute program instructions.
- The instructions are ordinary CPU instructions such as add, move data, and branch, but the multiple cores can run multiple instructions at the same time, increasing overall speed for programs amenable to parallel computing.
- Manufacturers typically integrate the cores onto a single integrated circuit die (known as a chip multiprocessor or CMP), or onto multiple dies in a single chip package.

ARCHITECTURE MODEL:

- In order to satisfy the high-performance and low-power requirements for advanced embedded systems with greater flexibility, it is necessary to develop parallel processing on chips by taking advantage of the advances being made in semiconductor integration.
- Figure illustrates the basic architecture of our heterogeneous multicore.
- Several low-power CPU cores and special purpose processor (SPP) cores, such as a digital signal processor, a media processor, and a dynamically reconfigurable processor, are embedded on a chip.
- In the figure, the number of CPU cores is m . There are two types of SPP cores, SPP a and SPP b on the chip.
- The values n and k represent the respective number of SPP a and SPP b cores.
- Each processor core includes a processing unit (PU), a local memory (LM), and a data transfer unit (DTU) as the main elements.
- The PU executes various kinds of operations.
- For example, in a CPU core, the PU includes arithmetic units, register files, a program counter, control logic, etc., and executes machine instructions.
- With some SPP cores like the dynamic reconfigurable processor, the PU executes a large quantity of data in parallel using its array of arithmetic units.
- The LM is a small-size and low-latency memory and is mainly accessed by the PU in the same core during the PU's execution.
- Some cores may have caches as well as an LM or may only have caches without an LM.
- The LM is necessary to meet the real-time requirements of embedded systems.
- The access time to a cache is non-deterministic because of cache misses.
- On the other hand, the access to an LM is deterministic.

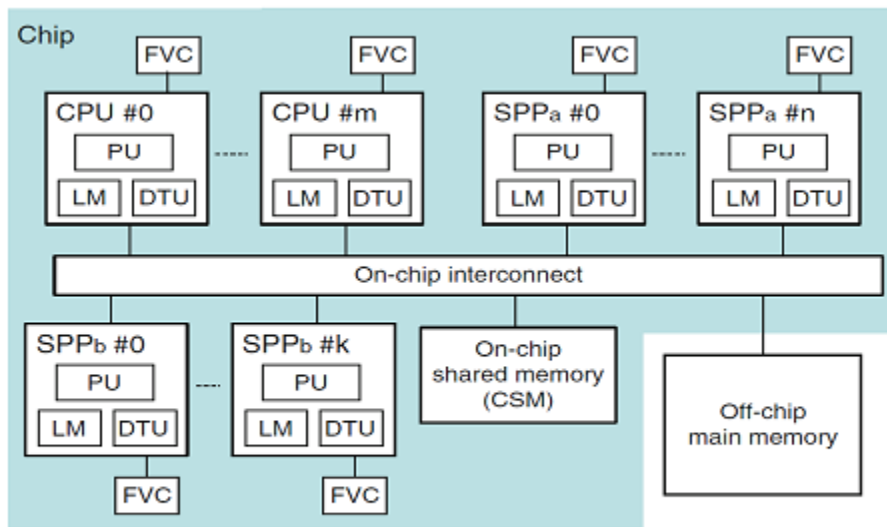


Figure: Heterogeneous Multicore architecture

- By putting a program and data in the LM, we can accurately estimate the execution cycles of a program that has hard real-time requirements.
- A data transfer unit (DTU) is also embedded in the core to achieve parallel execution of internal operation in the core and data transfer operations between cores and memories.
- Each PU in a core processes the data on its LM or its cache, and the DTU simultaneously executes memory-to-memory data transfer between cores.
- The DTU is like a direct memory controller (DMAC) and executes a command that transfers data between several kinds of memories, then checks and waits for the end of the data transfer, etc.
- Some DTUs are capable of command chaining, where multiple commands are executed in order.
- The frequency and voltage controller (FVC) connected to each core controls the frequency, voltage, and power supply of each core independently and reduces the total power consumption of the chip.
- If the frequencies or power supplies of the core's PU, DTU, and LM can be independently controlled, the FVC can vary their frequencies and power supplies individually.
- For example, the FVC can stop the frequency of the PU and run the frequencies of the DTU and LM when the core is executing only data transfers.
- The on-chip shared memory (CSM) is a medium-sized on-chip memory that is commonly used by cores. Each core is connected to the on-chip interconnect, which may be several types of buses or crossbar switches.

- The chip is also connected to the off-chip main memory, which has a large capacity but high latency.
- Figure illustrates a typical model of a heterogeneous multicore architecture.
- A number of variations based on this architecture model are possible.
- Several variations of an LM structure are shown here.
- **Case (a)** is a hierarchical structure where the LM has two levels. LM 1 is a first-level, small-size, low-latency local memory. LM 2 is a second-level, medium-sized, not-so-low-latency local memory.
- For example, the latency from the PU to LM 1 is one processor cycle, and the latency to LM 2 is a few processor cycles.
- **Case (b)** is a Harvard type. The LM is divided into an LM i that stores instructions and an LM d that stores data. The PU has an independent access path to each LM. This structure allows parallel accesses to instructions and data and enhances processing performance.
- **Case (c)** is a combination of (a) and (b).
- The LM i and LM d are first-level local memories for instructions and data, respectively. LM 2 is a second-level local memory that stores both instructions and data. In each case, each LM is mapped on a different address area; that is, the PU accesses each LM with different addresses.

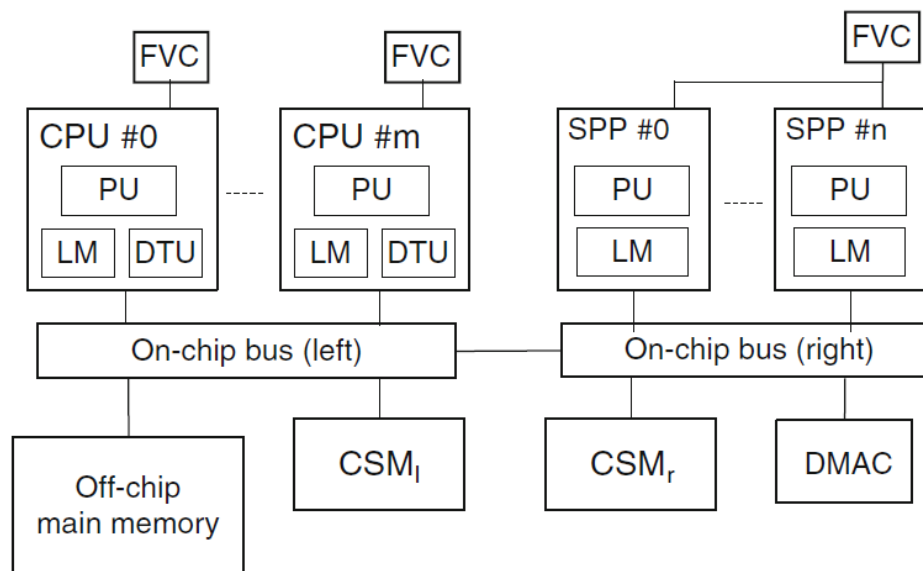


Figure: Example of other heterogeneous multicore configurations

- In Figure, we can see other configurations of a DTU, CSM, FVC, and an on-chip Interconnect.
- First, processor cores are divided into two clusters.
- The CPU cores, the CSM l, and the off-chip main memory are tightly connected in the left cluster.
- The SPP cores, CSM r, and the DMAC are nearly connected in the right cluster.
- Not every SPP core has a DTU inside. Instead, the DMAC that has multiple channels is commonly used for data transfer between an LM and a memory outside an SPP core.
- For example, when data are transferred from an LM to the CSM r, the DMAC reads data in the LM via the right on-chip bus, and the data are written on the CSM r from the DMAC.
- We need two bus transactions for this data transfer.
- On the other hand, if a DTU in a CPU core on the left cluster is used for the same transfer, data are read from an LM by the DTU in the core, and the data are written on the CSM l via the on-chip bus by the DTU.
- Only one transaction on the on-chip bus is necessary in this case, and the data transfer is more efficient compared with the case using the off-core DMAC.
- Although each CPU core in the left cluster has an individual FVC, the SPP cores in the right cluster share an FVC.
- With this simpler FVC configuration, all SPP cores operate at the same voltage and the same frequency, which are controlled simultaneously.

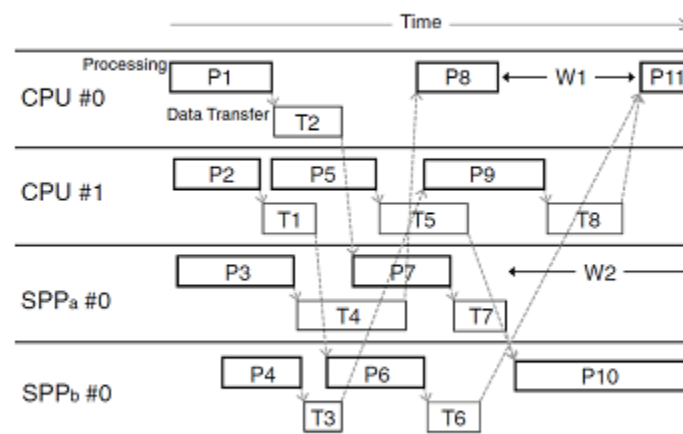


Figure: Parallel Operation

- When a program is executed on a heterogeneous multicore, it is divided into small parts, and each is executed in parallel in the most suitable processor core, as shown in Figure.
- Each core processes data on its LM or cache in a P_i period, and the DTU of a core simultaneously executes a memory–memory data transfer in a T_i period.
- For example, CPU #1 processes data on its LM at a P_2 period, and its DTU transfers processed data from the LM of CPU #1 to the LM of SPP b #0 at the T_1 period.
- After the data transfer, SPP b #0 starts to process data on its LM at a P_6 period.
- CPU #1 also starts a P_5 process that overlaps with the T_1 period.
- In the parallel operation of Figure, there is a time slot like W_1 when the corresponding core CPU #0 does not need to process or transfer data from the core.
- During this time slot, the frequencies of the PU and DTU of CPU #0 can be slowed down or stopped, or their power supplies can be cut off by control of the connected FVC.
- As there are no internal operations of SPP a #0 during the time slot W_2 , the power of SPP a #0 can be cut off during this time slot.
- This FVC control reduces redundant power consumption of cores and can result in lowering the power consumption of a heterogeneous multicore chip.

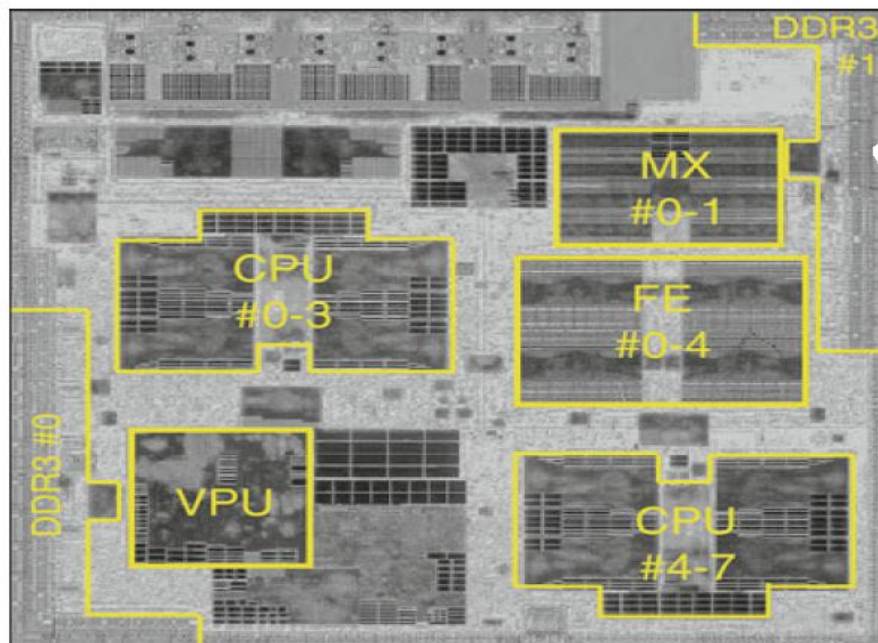


Figure: Heterogeneous Multicore Chip

The above figure is a photograph of the RP-X chip

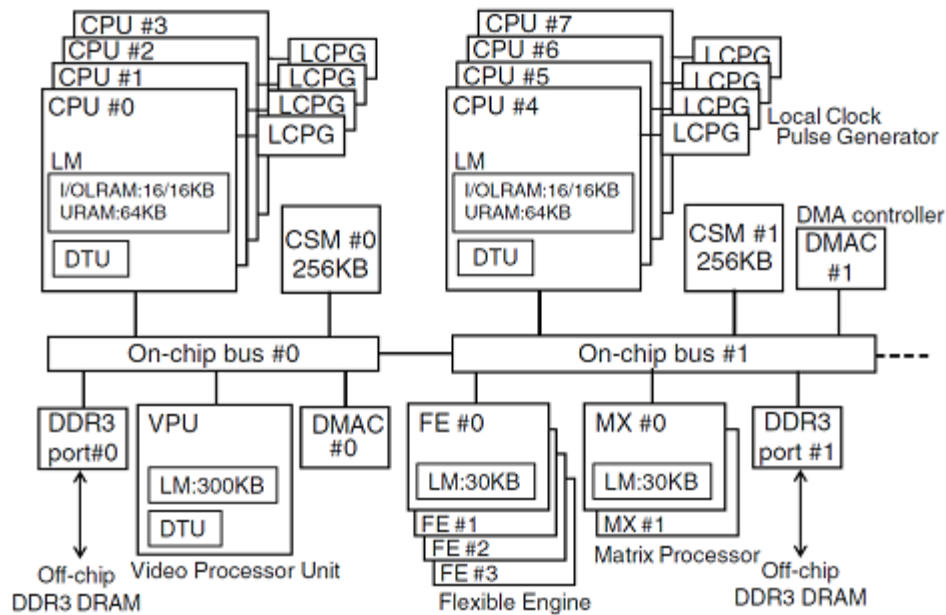


Figure: Block diagram of heterogeneous multicore chip

- The above figure depicts the internal block diagram.
- The chip includes eight CPU cores and seven three-type SPP cores.
- The CPU includes a two-level LM as well as a 32-KB instruction cache and a 32-KB operand cache.
- The LM consists of a 16-KB I/OLRAM for instruction storage, a 16-KB OLRAM for data storage, and a 64-KB URAM for instruction and data storage.
- Each CPU has a local clock pulse generator (LCPG) that corresponds to the FVC and controls the CPU's clock frequency independently.
- The eight CPUs are divided into two clusters.
- Each cluster of four CPUs is connected to independent on-chip buses.
- Additionally, each cluster has a 256-KB CSM and a DDR3 port which is connected to off-chip DDR3 DRAMs.
- Three types of SPPs are embedded on the chip.
- The first SPP is a video processing unit which is specialized for video processing such as MPEG-4 and H.264 codec.
- The VPU has a 300-KB LM and a DTU built-in.
- The second and third SPPs are four flexible engines and two matrix processors and they are included in another cluster.
- The FE is a dynamically reconfigurable processor which is suitable for data-parallel processing such as digital signal processing.
- The FE has an internal 30-KB LM but does not have a DTU.

- The on-chip DMA controller (DMAC) that can be used in common by on-chip units or a DTU of another core is used to transfer data between the LM and other memories.
- The MX has 1,024-way single instruction multiple data (SIMD) architecture that is suitable for highly data-intensive processing such as video recognition.
- The MX has an internal 128-KB LM but does not have its DTU, just as with the FE.
- In the chip photograph, the upper-left island includes four CPUs, and the lower-left island has the VPU with other blocks.
- The left cluster includes these left islands and a DDR3 port depicted at the lower-left side.
- The lower-right island in the photo includes another four CPUs, the center-right island has four FEs, and the upper-right has two MXs.
- The right cluster includes these right islands and a DDR3 port depicted at the upper-right side.
- With these 15 on-chip heterogeneous cores, the chip can execute a wide variety of multimedia and digital-convergence applications at high-speed and low-power consumption.

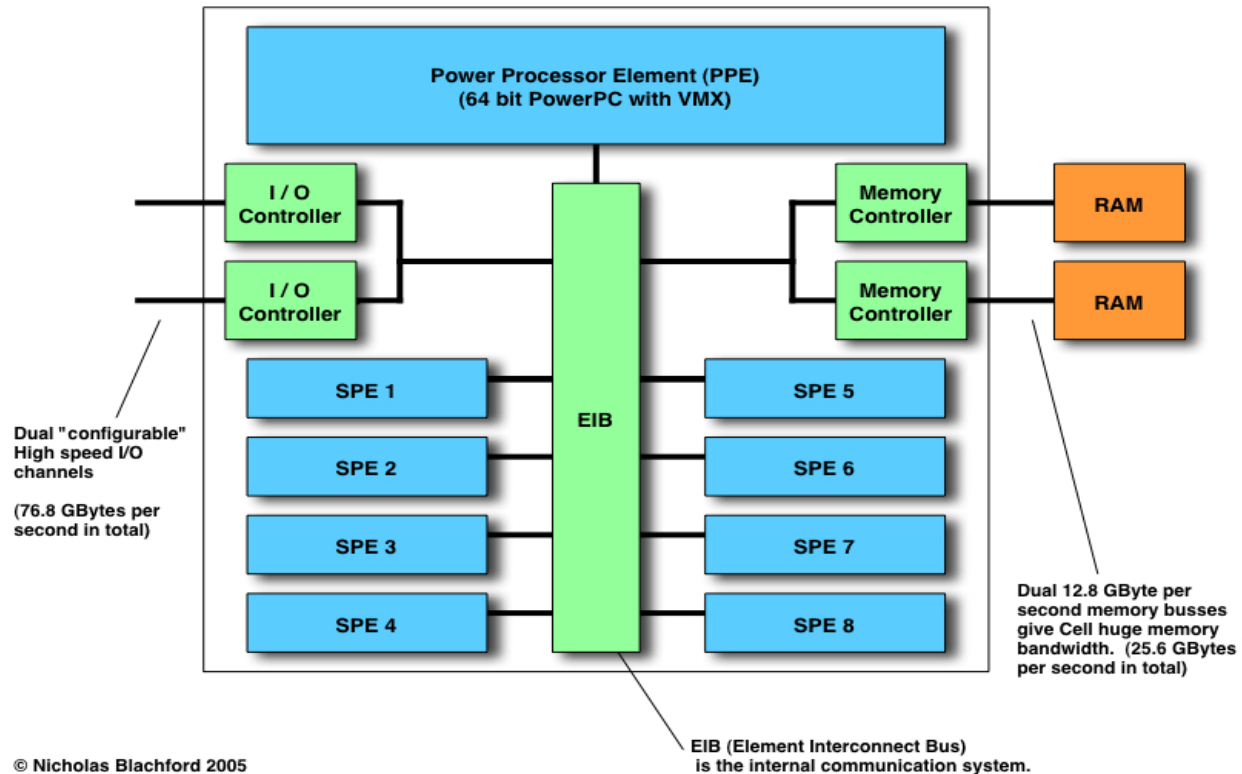
IBM CELL PROCESSOR:

GENERAL OVERVIEW:

- A chip with one PPC hyper-threaded core called PPE and eight specialized cores called SPEs.
- The challenge to be solved by the Cell was to put all those cores together on a single chip.
- This was made possible by the use of a bus with outstanding performance.
- The Cell processor can be split into four components:
 1. External input and Output structures
 2. The main processor called the Power Processing Element (PPE)
 3. Eight fully-functional co-processors called the Synergistic Processing Elements, or SPEs
 4. A specialized high-bandwidth circular data bus connecting the PPE, input/output elements and the SPEs, called the Element Interconnect Bus or EIB

OVERVIEW OF THE ARCHITECTURE OF A CELL CHIP:

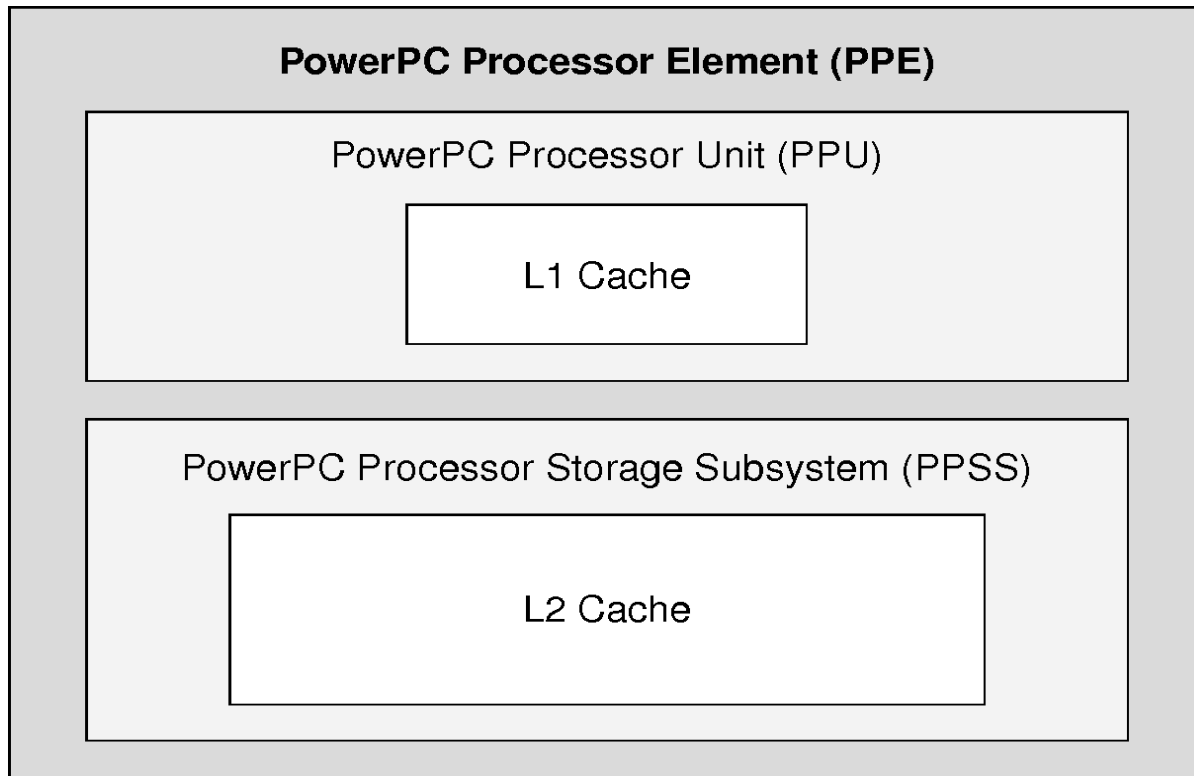
Cell Processor Architecture



POWERPC PROCESSOR ELEMENT : (PPE)

- The PowerPC Processor Element, usually denoted as PPE is a dual-threaded PowerPC processor version 2.02.
- This 64-bit RISC processor also has the Vector/SIMD Multimedia Extension.
- The PPE's role is crucial in the Cell architecture since it is on the one hand running the OS, and on the other hand controlling all other resources, including the SPEs.
- The PPE is made out of two main units:
 - 1: The Power Processor Unit
 - 2: The Power Processor Storage Subsystem (PPSS)

PPE BLOCK DIAGRAM:



PPU:

- It is the processing part of the PPE and is composed of:
 - a. A full set of 64-bit PowerPC registers.
 - b. 32 128-bit vector multimedia registers.
 - c. a 32KB L1 instruction cache.
 - d. a 32KB L1 data cache.
- All the common components of a ppc processors with vector/SIMD extensions (instruction control unit, load and store unit, fixed-Point integer unit, floating-point unit, vector unit, branch unit, virtual memory management unit).
- The PPU is hyper-threaded and supports 2 simultaneous threads.

PPSS:

- This handles all memory requests from the PPE and requests made to the PPE by other processors or I/O devices.
- It is composed of:
 1. A unified 512-KB L2 instruction and data cache.
 2. Various queues
 3. A bus interface unit that handles bus arbitration and pacing on the Element Interconnect Bus

SYNERGISTIC PROCESSOR ELEMENTS: SPE

- Each Cell chip has 8 Synergistic Processor Elements.
- They are 128-bit RISC processor which is specialized for data-rich, compute-intensive SIMD applications.
- This consists of two main units.
 - 1: The Synergistic Processor Unit (SPU)
 - 2: The Memory Flow Controller (MFC)

THE SYNERGISTIC PROCESSOR UNIT (SPU):

- This deals with instruction control and execution.
- It includes various components:
 1. A register file of 128 registers of 128 bits each.
 2. A unified instruction and data 256-kB Local Store (LS).
 3. A channel-and-DMA interface.
 4. As usual, an instruction-control unit, a load and store unit, two fixed-point units, a floating point unit
- The SPU implements a set of SIMD instructions, specific to the Cell.
- Each SPU is independent, and has its own program counter.
- Instructions are fetched in its own Local Store LS
- Data are also loaded and stored in the LS.

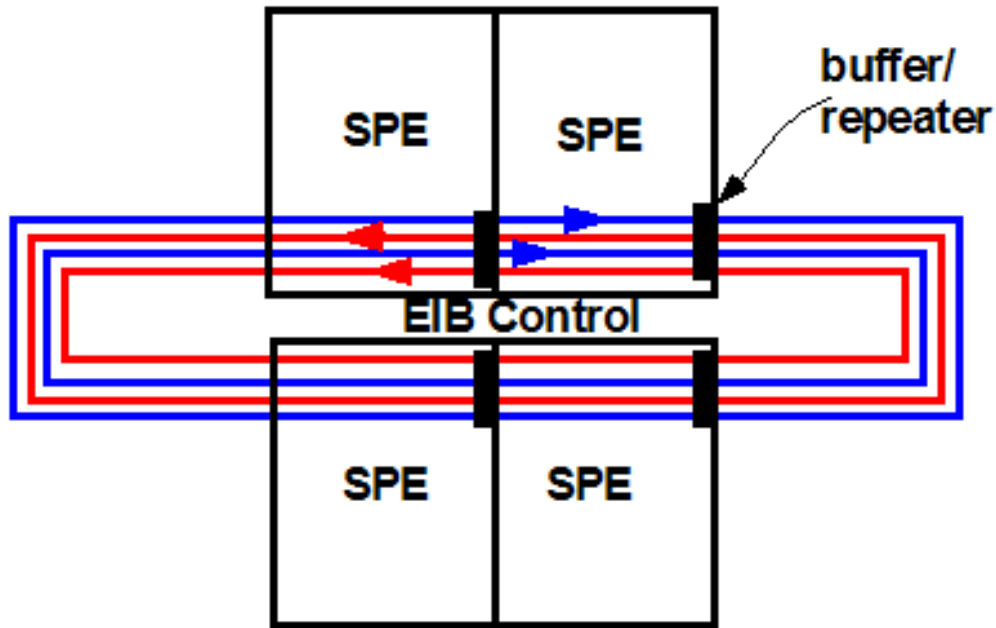
THE MEMORY FLOW CONTROLLER (MFC):

- It is actually the interface between the SPU and the rest of the Cell chip.
- MFC interfaces the SPU with the EIB.
- In addition to a whole set of MMIO registers, this contains a DMA controller.

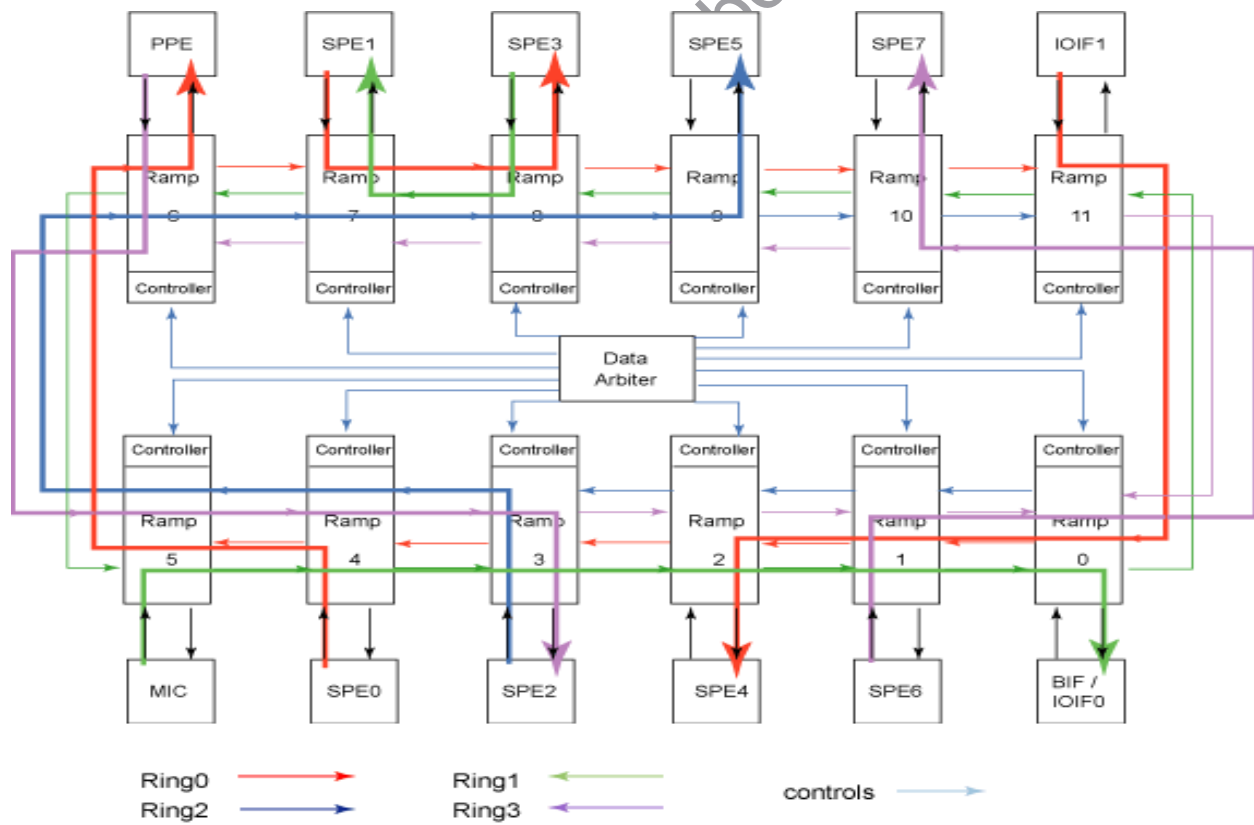
BUS DESIGN AND COMMUNICATION AMONG THE CELL:

1: The Element Interconnect Bus:

- This bus makes it possible to link all parts of the chip.
- The EIB itself is made out of a 4-ring structure (two clockwise, and two counterclockwise) that is used to transfer data, and a tree structure used to carry commands.
- It is actually controlled by what is called the Data Arbiter.
- This structure allows 8 simultaneous transactions on the bus.



CONCURRENT TRANSACTIONS IN THE BUS:



2: Input/output Interfaces

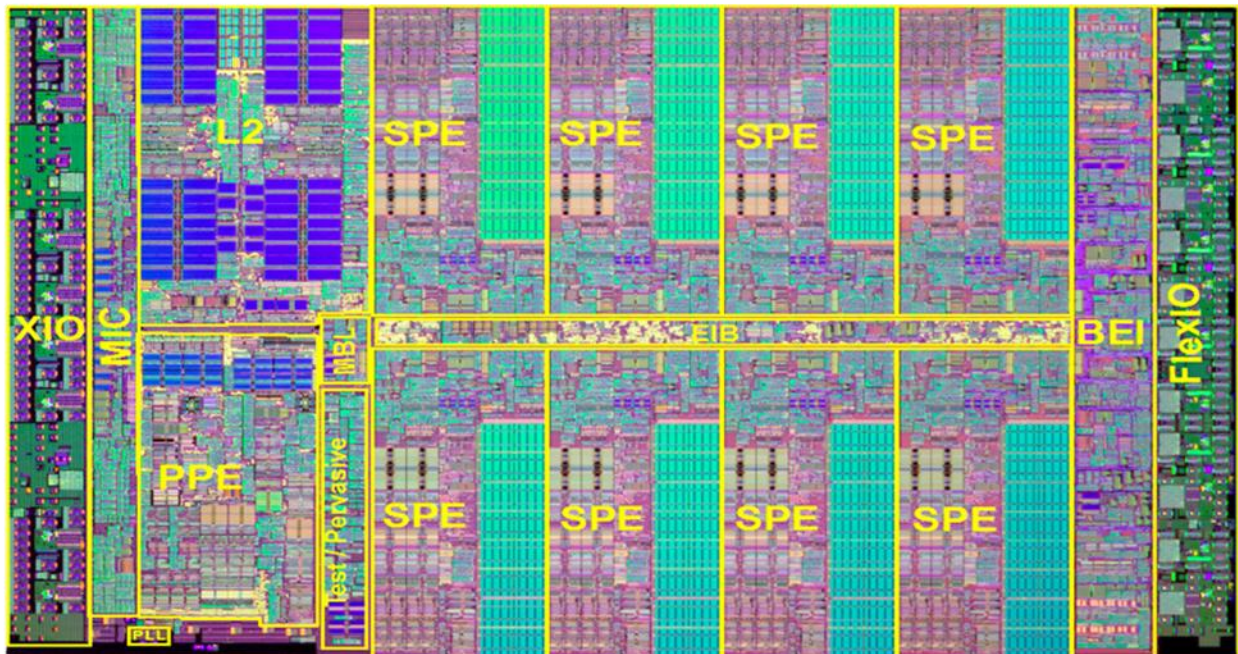
➤ The Memory Interface Controller (MIC).

- It provides an interface between the EIB and the main storage.
- It currently supports two Rambus Extreme Data Rate (XDR) I/O (XIO) memory channels.

➤ The Cell Broadband Engine Interface (BEI).

- This is the interface between the Cell and I/O devices, such as GPUs and various bridges.
- It supports two Rambus FlexIO external I/O channels.
- One of these channels only supports non-coherent transfers. The other supports either coherent or no coherent.

CELL:



- IBM/Toshiba/Sony joint project - 4-5 years, 400 designers
 - ✓ 234 million transistors, 4+ Ghz
 - ✓ 256 Gflops (billions of floating pointer operations per second)

KEY ATTRIBUTES:

- ✓ Cell is Multi-Core
- ✓ Cell is a Flexible Architecture
- ✓ Cell is a Broadband Architecture
- ✓ Cell is a Real-Time Architecture
- ✓ Cell is a Security Enabled Architecture

www.csetube.in