Sr. No.	Processors → Features ↓	8086	80386DX	80386SX	80486DX	Pentium	Pentium Pro	Pentium MMX	Pentium II	Pentium II based Celeron	Pentium III	Pentium II or III based XEON	Pentium 4
1	Processor Size	16 bits	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits
2	Speed	5 to 10 MHz	16 to 33 MHz	16 to 33 MHz	25 to 50 MHz	60 to 66 MHz (later upto 200 MHz)	60 MHz (later upto 200 MHz)	166 MHz (later upto 300 MHz)	233 to 300 MHz (later upto 450 MHz)	233 to 300 MHz (later upto 500 MHz)	450 to 500 MHz (later upto 1.4 GHz)	500 MHz (later upto 1 GHz)	400 MHz (later upto 2.26 GHz)
3	MIPS	0.33 to 0.75	5 -11.4	2.5 to 2.9	20 to 41	100 to 112	-	-	-	-	-	-	6500 to 10000
4	Address bus size	20	32	24	32	32	32	32	32	32	32	32	32
5	Data bus size	16	32	16	32	64	64	64	64	64	64	64	64
6	No. of transistors	29000	275000	275000	1.2	3.1 million	5.5 million	4.2 million	7.5 million	7.5 million	9.5 to 28	9.5 to 28	77 million
7	Addressable memory	1 MB	4 GB	16 MB	4 GB	4 GB	4 GB	4 GB	4 GB	4 GB	4 GB	4 GB	4 GB
8	Virtual memory	-	64 TB	32 GB	64 TB	64 TB	64 TB	64 TB	64 TB	64 TB	64 TB	64 TB	64 TB
9	L1 Cache	-	-	-	8KB Unified	16KB Spilt	16KB Spilt	32KB Spilt	32KB Spilt	32KB Spilt	32KB Spilt	32KB Spilt	12 KμCpcodes + 8KB data
10	L2 Cache	-	-	-	-	-	256KB to 1MB Unified	-	512 KB	-	256 KB ATC (or 512 KB)	256 KB ATC (or 1MB or 2MB)	1 MB ATC

Sr. No.	Processors → / Features ↓	8086	80386DX	80386SX	80486DX	Pentium	Pentium Pro	Pentium MMX	Pentium II	Pentium II based Celeron	Pentium III	Pentium II or III based XEON	Pentium 4
11	L3 Cache	-	-	-	-	-	-	-	-	--	-	-	2 MB
12	On chip FPU	-	-	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
13	Superscalar	-	-	-	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
14	MMX instruction set	-	-	-	-	-	-	SSE1	SSE1	SSE1	SSE2	SSE2	SSE2
15	Hyper threading support	-	No	No	No	No	No	No	No	No	No	No	Yes
16	No. of cores	1	1	1	1	1	1	1	1	1	1	1	1
17	Fabrication process	3 µm	1 µm	1 µm	1 µm	0.6 µm	0.35 µm	0.35 µm	0.35 µm	0.35 µm	0.18 µm	0.25 µm	0.13 µm
18	Generation	P1	P3	P3	P4	P5	P6	P6	P6	P6	P6	P6	NetBurst
19	SMP (multiprocessor) support	No	No	No	No	No	No	No	No	No	No	No	Yes
20	Integer pipeline stages	2	3	3	3	5	14	5	14	14	14	14	20
21	No. of integer pipelines	1	1	1	1	2	2	2	2	2	2	2	4
22	Floating point pipeline stages	-	-	-	-	8	-	8	8	8	11	11	20
23	No. of floating point pipeline stages	-	-	-	1	1	1	1	1	1	1	1	2
24	Overclocking feature	No	No	No	No	No	No	No	No	No	No	No	Yes

Pentium 4 – Processor of the New Millennium

rithesh.kini@thadomal.org

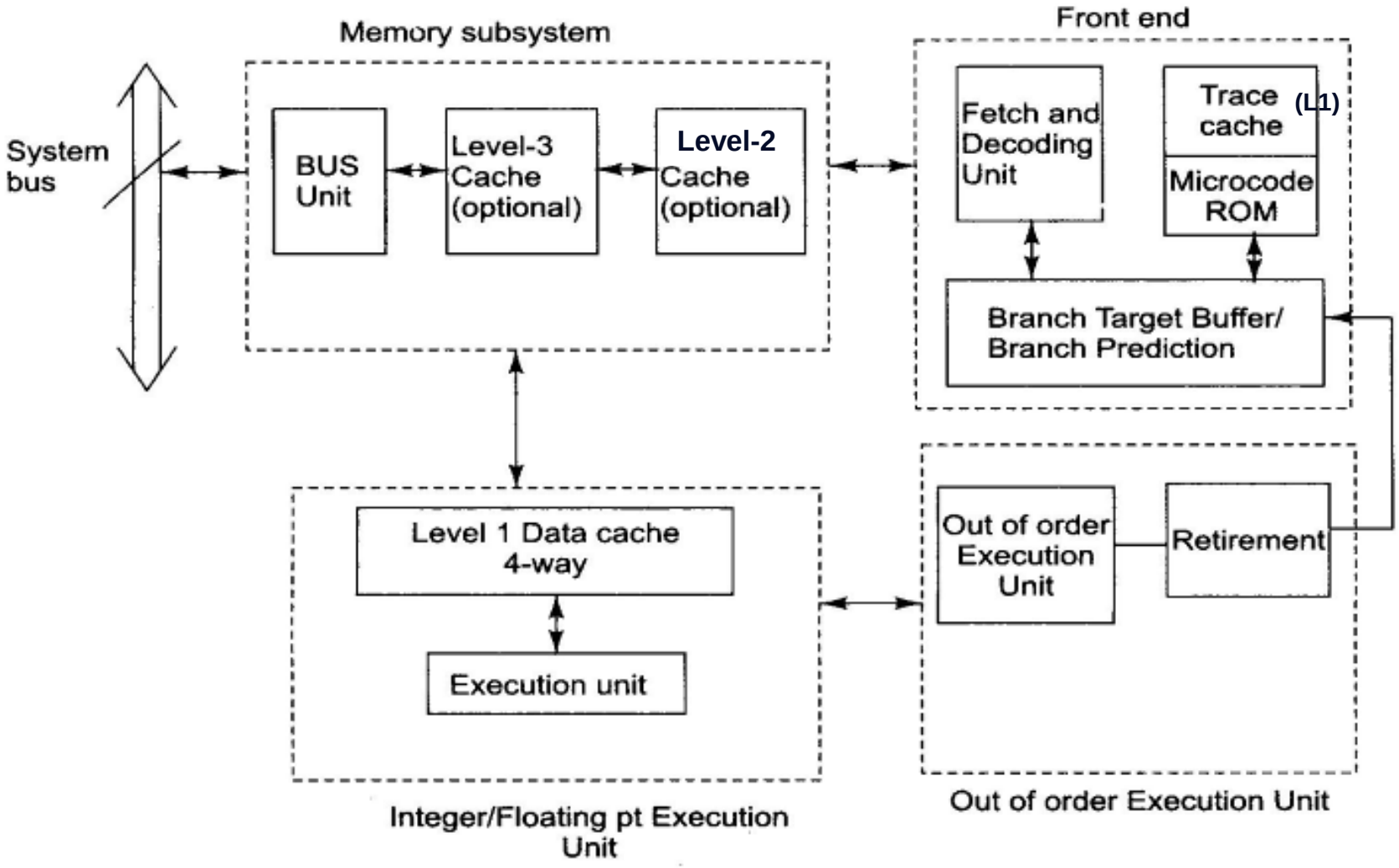
Salient Features of Pentium – 4

Pentium 4 microprocessor arrived in the scene in June, 2000. After Pentium PRO processor, designed using P6 micro-architecture, which was released in 1995, Pentium-4 is the next x86 processor from Intel. This new processor with Pentium-4 Net Burst architecture utilizes all the features of earlier P6 architecture of Pentium 3 and includes many more. Some of the features of Pentium 4 are as follows:

(i) It is based on NetBurst microarchitecture
(ii) It has 42 million transistors, fabricated using 0.18 micron CMOS process.
(iii) Its die size is 217 sq. mm, and power consumption is 50W
(iv) Clock speed varies from 1.4 GHz to 1.7 GHz. At 1.5 GHz the microprocessor delivers 535 SPECint2000 and 558 SPECfp 2000 of performance.
(v) It has hyper-pipelined technology—Its pipeline depth extends to 20 stages.
(vi) In addition to the L1 8 KB data cache, it also includes an Execution Trace Cache that stores up to 12 K decoded micro-ops in the order of program execution.
(vii) The on-die 256KB L2-cache is non-blocking, 8-way set associative. It employs 256-bit interface that delivers data transfer rates of 48 GB/s at 1.5 GHz.
(viii) Pentium-4 NetBurst microarchitecture introduces Internet Streaming SIMD Extensions 2 (SSF2) instructions. This extends the SIMD capabilities that MMX technology and SSF techonology delivered by adding 144 new instructions. These instructions includes 128-bit SIMD integer arithmetic and 128-bit SIMD double-precision floating-point operations.
(ix) It supports 400 MHz system bus, which provides up to 3.2 GB/s of bandwidth. The bus is fed by dual PC800 Rambus channel. This compares to the 1.06 GB/s delivered on the Pentium-III processor's 133-MHz system bus.
(x) Two Arithmetic Logic Units (ALUs) on the Pentium 4 processor are clocked at twice the core processor frequency. This allows basic integer instructions such as Add, Subtract, Logical AND, Logical OR, etc. to execute in a half clock cycle.
(xi) Advanced dynamic execution.

Single Instruction and Multiple Data Stream.

Net burst Micro Architecture for Pentium – 4



Block Diagram of Pentium 4 Microarchitecture

rithesh.kini@thadomal.org

Net burst Micro Architecture for Pentium – 4

- Front End Module
  - The Front End Module of Pentium 4 processor contains
    - IA-32 Instruction decoder,
    - Trace Cache
    - Microcode ROM
    - Front end branch Predictor

rithesh.kini@thadomal.org

Net burst Micro Architecture for Pentium – 4

- **IA 32 Instruction decoder**
  - The instructions supported by Pentium 4 are variable length and are supported by many different addressing modes.
  - The role of instruction decoder is to decode these instructions concurrently and translate them in to micro operations known as **µops**.
  - A single instruction decoder decodes one instruction per clock cycle.
  - Some instructions are translated into single µops while others are translated into multiple numbers of µops.
  - In case of a complex instruction, when the instruction needs to be translated into more than four µops , the decoder usually does not decodes such instructions.
  - Rather it transfers the task to a **Microcode ROM**.

rithesh.kini@thadomal.org

Net burst Micro Architecture for Pentium – 4

- **Trace Cache (TC)**
  - The basic function of front end module is to fetch the instructions to be executed, decode them and feed decoded instruction to the next module, which is the **out of order** execution module.
  - The instructions are first decoded into basic micro operations known as µops, and the stream of decoded instructions are fed to a level - 1 (L1) instruction cache.
  - This special instruction cache is known as Trace Cache, which is a special feature of Pentium micro architecture.
  - It is special because it does not store the instructions but decoded stream of instructions, i.e., micro operations or µops, thus enhancing the execution speed considerably.

rithesh.kini@thadomal.org

Net burst Micro Architecture for Pentium – 4

- **Microcode ROM**
  - When some complex instructions like interrupt handling or string manipulation etc. appear, Trace cache transfers the control to a micro code ROM, which stores the µops corresponding to these complex instructions.
  - When control is passed to the microcode ROM, the corresponding µops are issued.
  - After the µops are issued by the microcode ROM, the control goes to the trace cache once again.
  - The µops delivered by the trace cache and the microcode ROM are buffered in a queue in an orderly fashion.
  - The resultant flow of µops is next fed to the execution engine.

rithesh.kini@thadomal.org

Net burst Micro Architecture for Pentium – 4

- **Front end branch Predictor in Pentium 4**
  - The other important unit in the Front end is the branch Prediction logic unit.
  - This unit predicts the locations from where the next instruction bytes are fetched.
  - The predictions are made based on past history of the program execution.
  - The earlier generation processor follows simple branching strategy.
  - When the processor comes across a branch instruction, it evaluates the branch condition.
  - The condition evaluation may involve a complex calculation, which may consume time and the processor has to wait till the condition is computed and thereafter it decides whether to take the branch or not.

rithesh.kini@thadomal.org

Net burst Micro Architecture for Pentium – 4

- **Branch Prediction**
  - The modern day fast processors cannot wait till the branch condition is evaluated to decide whether to take a branch since this will unnecessarily slow down the speed of execution.
  - These processors take the strategy of speculating whether the branch condition will be satisfied.
  - Pentium, for example, makes a guess about the branching using strategy called **speculative execution**.
  - This strategy involves making a guess at which direction the branch is going to be taken and then branching at the new branch target even before the branching condition is actually evaluated.
  - Many strategies have been suggested for speculative prediction and the guess is made used one of these branch prediction strategies.

rithesh.kini@thadomal.org

Net burst Micro Architecture for Pentium – 4

- If, however the processor incorrectly predicts a branch it may lead to severe problem.
- In case of incorrect prediction, these instructions are fetched from the wrong branch predictions and may be executed incorrectly for a wrong speculation.
- In such a case, the pipeline has to be flushed of the erroneous speculative instructions and results.
- After flushing out the wrong instructions, the instructions from the correct branch address are fetched, and executed.
- Flushing the pipeline of instructions and results is expensive and produces a delay of several cycles.
- The delay depends on the level of pipelining in the processor.
- Also there is a delay associated with loading the new instruction stream.
- The resulting delay invariably degrades the system performance significantly, if such erroneous predictions take place often.

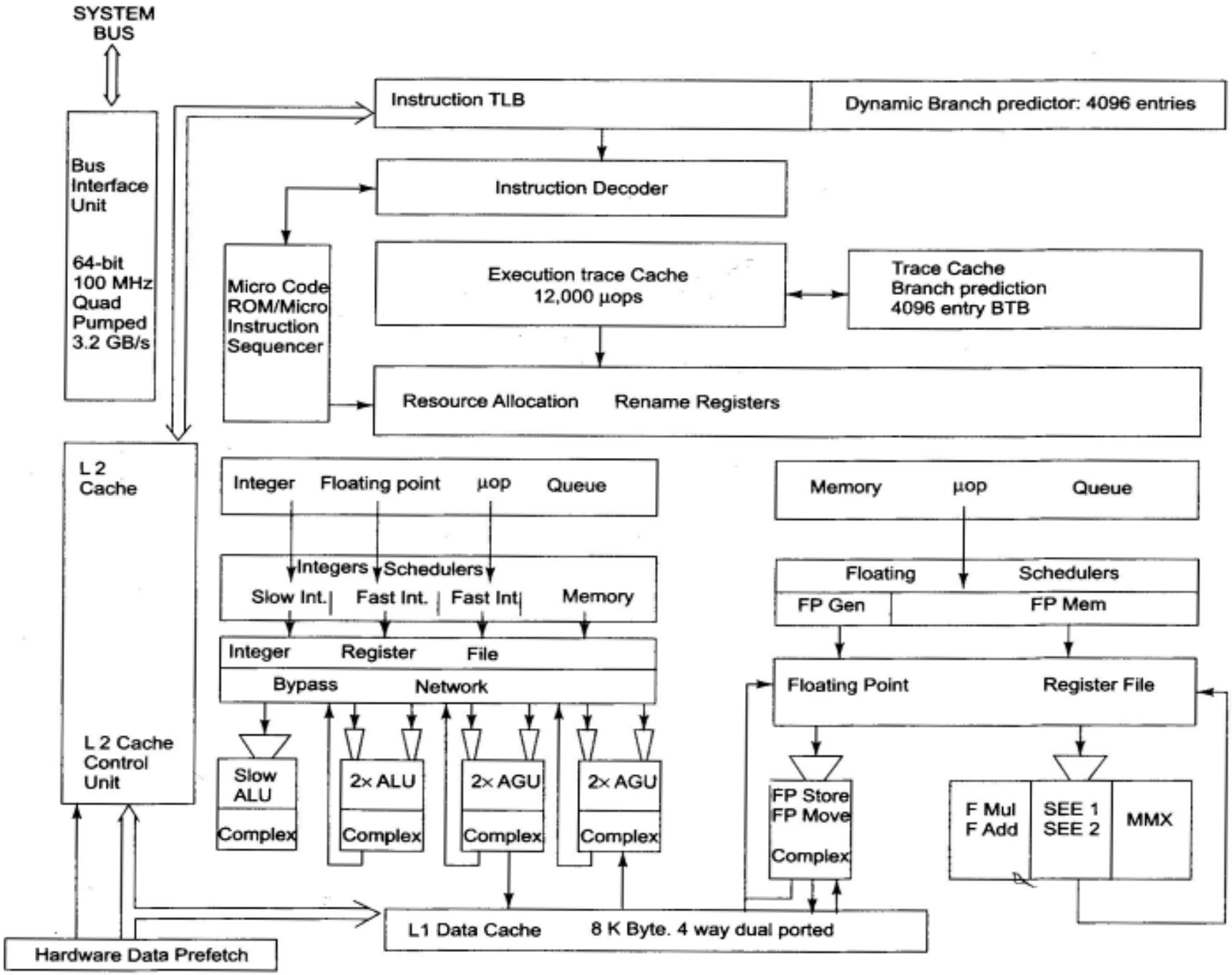
rithesh.kini@thadomal.org

Net burst Micro Architecture for Pentium – 4

- As the length of pipeline in a processor increases, the degradation also increases proportionately.
- In case of a wrong prediction, Pentium – 4 with 20 stages will have to wait for considerable number of cycles, while new instructions are loaded from the cache.
- The P4 has minimum loss of 19 clock cycles due to each erroneous prediction.
- This is the loss incurred when the code resides in the L1 cache.
- The loss will be higher if the correct branch is not found in the L1 cache, since in that case the data has to be fetched from L2 cache.

rithesh.kini@thadomal.org

D



Detail Architecture of Pentium Processor

rithesh.kini@thadomal.org

Instruction Translation Look Aside Buffer (ITLB) And Branch Prediction

- If there is a trace cache miss, then instructions bytes are required to be fetched from the L2 cache.
- These are next decoded into μops to be placed in the Trace cache (TC).
- The instruction Translation look aside Buffer (ITLB) receives the request from the TC to deliver a new instruction, and it translates the next instruction pointer address to a physical address.
- A request is sent to the L2 cache, and instruction bytes are returned.
- These bytes are placed into streaming buffers, which hold the byte until they are decoded.

rithesh.kini@thadomal.org

Instruction Translation Look Aside Buffer (ITLB) And Branch Prediction

- Since there are two logical processors there are two ITLBs.
- Thus each logical processor has its own ITLB and its own instruction pointer to track the progress of instruction fetch for each of them.
- Now suppose both the logical processors request the access of L2 cache, the instruction fetch logic performs arbitration based on which processor request has arrived first.
- Accordingly, it sends requests to the L2 cache and grants the request of the first processor.
- It, however, reserves at least one request slot for each logical processor.
- In this way, both logical processors can access and fetch data from L2 cache without any conflict.

rithesh.kini@thadomal.org

Instruction Translation Look Aside Buffer (ITLB) And Branch Prediction

- Before the instructions are decoded, they are stored in streaming buffers.
- Thus each logical processor has its own set of 64 byte streaming buffers, which store the instruction bytes and subsequently they are dispatched to the instruction decode stage.

rithesh.kini@thadomal.org

Out of order Execution Engine

- Allocation, register renaming, scheduling, execution
- Allocator logic –
  - 126 reorder buffer entries
  - 128 integer, 128 floating point physical registers
  - 48 load and 24 store buffer entries
- register rename
  - rename onto machine's physical registers
  - 8 general purpose registers --→ expanded to use 128 physical registers
  - Register alias table
- Instruction Scheduling
  - 5 schedulers

rithesh.kini@thadomal.org

Rapid Execution Module and Memory Subsystem

RAPID EXECUTION MODULE

Pentium 4 has two ALUs (Arithmetic Logic Unit) and two AGUs (Address Generation Unit), which run at twice the processor speed. This implies that the ALUs in a 1.4 GHz processor works at 2.8 GHz. The doubled speed of these units means twice the number of instructions being executed per clock cycle.

Arithmetic and Logic Unit is responsible for carrying out all integer calculations (add, subtract, multiplication, division) and logical operations. AGUs are primarily used to resolve indirect mode of memory addressing. As can be comprehended, these units are quite important for high-speed processing which includes frequent fetching of instructions and arithmetic calculations.

MEMORY SUBSYSTEM

The memory subsystem involving virtual memory and paging is briefly described below.

Paging and Virtual Memory

With the flat or the segmented memory model, linear address space is mapped into the processors physical address space either directly or through paging when using direct mapping (paging disabled), each linear address has a one-to-one correspondence with a physical address. Linear address bits are sent out on the processor's address lines without translation.

When using IA-32 architecture's paging mechanism (paging enabled) linear address space is divided into pages which are mapped to virtual memory. The pages of virtual memory are then mapped as needed into physical memory when an operating system or execution uses paging. The paging mechanism is transparent to an application program. All that the application sees is linear address space.

In addition, IA-32 architecture's paging mechanism includes extension that support:

- Page Address Extensions (PAE) to address physical address space greater than 4G Bytes.
- Page Size Extension (PSE) to map linear address to physical address in 4 M bytes page.

Cache

The access to DRAM main memory is often very slow. To enhance the speed of data access fast SRAM caches are used to reduce this latency.

rithesh.kini@thadomal.org

Hyper-threading Technology

HYPERTHREADING TECHNOLOGY

Before discussing the Hyperthreading technology, let us look into the concept of threading and multithreading. As we have observed earlier, each process has a "context" that reflects all the information which completely describes the current state of execution of the process. For example, a process may use as its context the contents of the CPU registers, the program counter, the flags, etc. Each process, in turn, contains at least one thread and sometimes more than one thread. In case a process has multiple threads, each of these threads has its own local context. Also the process, as such, has a context, which is shared by all the threads in that process.

The features of the thread are (i) the threads may be bunched together into a process, (ii) the threads may be independent, (iii) the threads are usually simple in structure and are lightweight in the sense that they may enhance the speed of operation of the overall process. In a multiprocessor environment, different processes may run on different processors. Also different threads from the same process can also run on different processors. Thus compared to the use of single thread, multiple threads enhance the performance in a multiprocessor system.

Hyper-threading Technology

Thread Level Parallelism (TLP)

In many applications it is important to have multiple number of processes or threads to be executed in parallel. For example, in multiple object tracking in on line video surveillance, it is advantageous to execute several threads concurrently. These threads, may correspond to the tracking of each individual object. This kind of parallelism, known as thread level parallelism, yields better performance in many on line applications. Also most of the server applications today require multiple threads or processes that can be executed in parallel.

When the time slice assigned to the currently executing process is over, its context is saved to the memory. When the process begins executing again, the context of this process is again restored to the exactly same state that it was in when its execuction was halted. This whole process of (i) saving the context of the currently executing process, when the time slice is over, (ii) flushing the CPU of the same process and (iii) loading the context of the new next process is called a context switch. Now if a process contains M number of threads, then the total time for this context switching will obviously be M times that of a single thread context switching time. Thus context switching consumes a number of CPU cycles. Thus we infer two things here: (i) Multiple number of threads yields better performance, (ii) More number of threads consumes more time in context switching.

From the above discussion it is clear that to enhance the performances, we have to (i) reduce the number of context switches and (ii) provide more CPU execution time to each process. The solution is to execute more than one process at the same time. This again can be done by increasing the number of CPUs. In a system with multiple number of processors, the scheduler in the OS can schedule two processes to two different CPUs simultaneously for execution at the exact same time. Thus with two or more number of CPUs in the system, the process will not have to wait for a long duration to get executed.

Hyper-threading Technology

Strategies of Implementation

There are several strategies to implement hyperthreading.

As has been mentioned earlier, a single processor can execute multiple threads by switching between them. The scheme of context switching may again be of several types.

They are:

(i) *Time-slice multithreading* In this scheme the processor switches between different process threads after a fixed time slice. Since there is only one processor, Time-slice multithreading can result in loss of execution cycles. However, each thread is guaranteed to get attention of the processor when its turn comes. In case there is a cache miss, which is a long latency event, automatically the processor will switch to another thread.

(ii) *On chip multiprocessing* (CMP) This scheme involves the use of two processors on a single die. The two processors each have a full set of execution and architectural resources. The processors may or may not share a large on-chip cache. CMP is largely orthogonal to conventional multi-processor systems, as you can have multiple CMP processors in a multiprocessor configuration. CMP chip is significantly larger than the size of single core-chip and therefore more expensive to manufacture (i) the die size is obviously more and (ii) power consumption is also higher.

(iii) *Hyperthreading* Finally, there is simultaneous multi-threading, where multiple threads can be executed on a single processor without switching. The threads execute simultaneously and make much better use of the resources. This approach makes the most effective use of processor resources. How can it be done ?

Hyper-threading Technology

HYPERTHREADING IN PENTIUM

This new technology which was first introduced on the Intel Xeon processor in early 2002 was subsequently employed in Pentium 4 in November 2002 at clock frequencies of 3.06 GHz and higher.

In Pentium architecture a single physical processor appears as two logical processors. All the physical resources of the system are shared between these two logical processors. This means that the user programs can schedule the processes or threads to the two logical processors as if there are indeed two different physical processors. The instructions from both the logical processors can be executed simultaneously on shared execution resources.

Hyperthreading used the concept of simultaneous multithreading and shows an improvement in the Intel microarchitecture development. At the expense of an added cost of less than only 5 percent in added die area, the performance increases by about 25 percent.

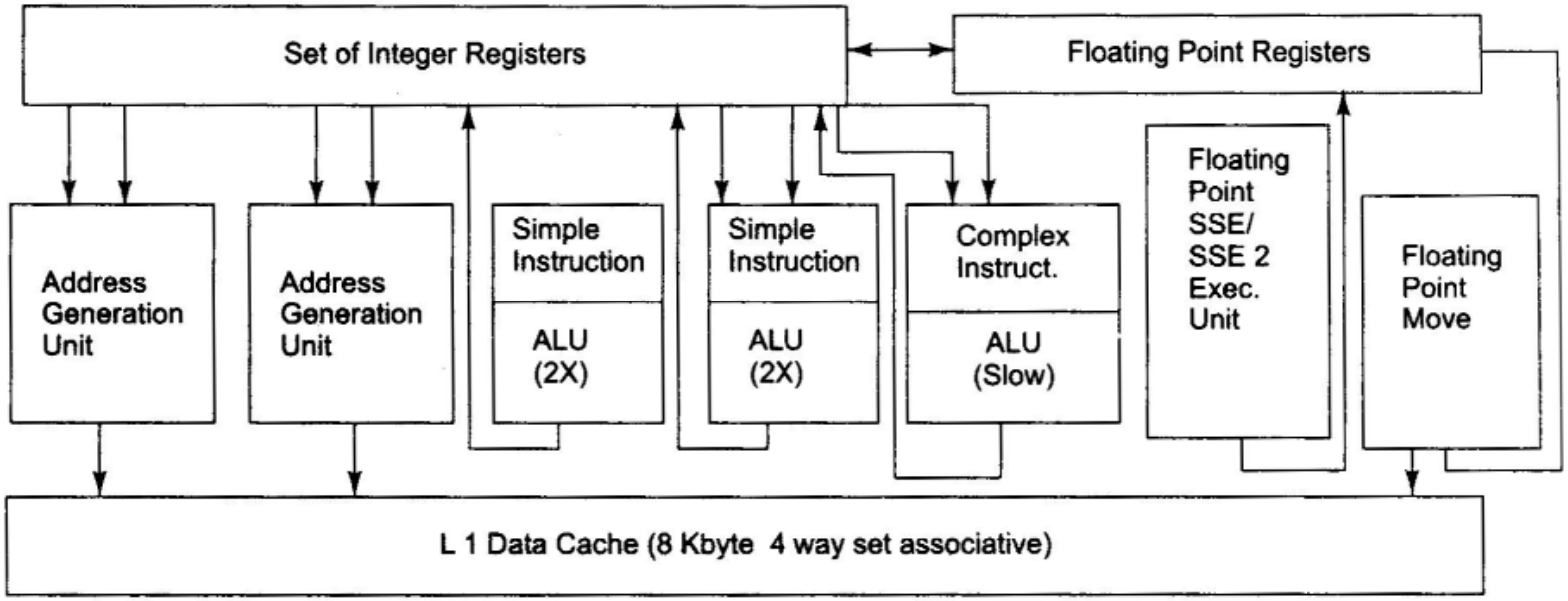
The major elegance of this architecture lies in devising appropriate resource sharing policy for each shared resource. Several resource sharing strategies have been investigated by the developers. Some of these are (a) partitioned resources, (b) threshold sharing, and (c) full sharing. The choice of sharing strategy to be adopted depends on several factors, such as, the traffic pattern, size of the resource, potential deadlock probabilities and other considerations.

To do this, there is one copy of the architecture state for each logical processor, and the logical processors share a single set of physical execution resources. From a software or architecture perspective, this means operating systems and user programs can schedule processes or threads to

Hyper-threading Technology

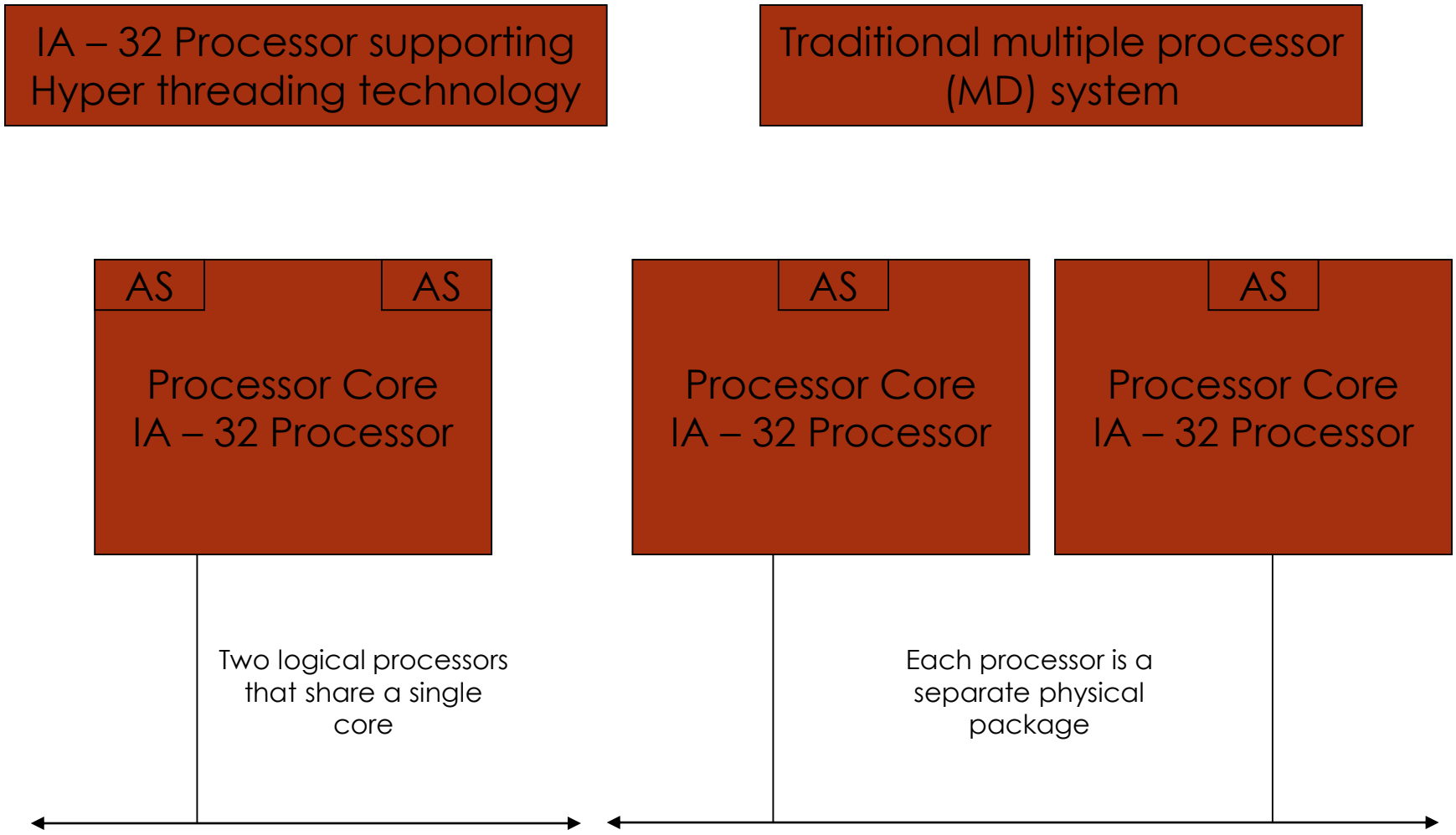
logical processors as they would on conventioal physical processors in a multi-processor system. From a microarchitecture perspective, this means that instructions from logical processors will persist and execute simultaneously on shared execution resouces.

Figure shows a multiprocessor system with two processors which does not incorporate Hyperthreading Technology.



A Multiprocessor System without Hyperthreading Technology

Comparison of hyper threading technology and traditional data processor system



rithesh.kini@thadomal.org