Computer Architecture

Prof. Mainak Chaudhuri
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

Lecture - 36
Simultaneous multithreading, multi-cores

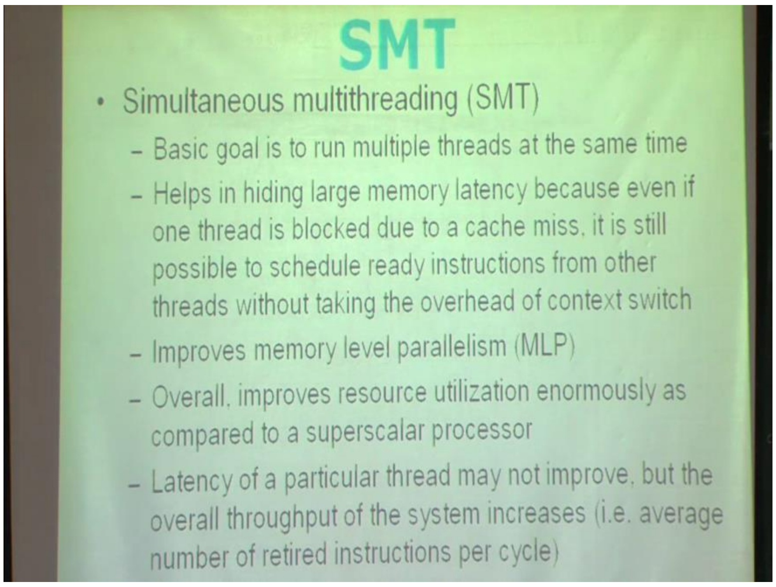
(Refer Slide Time: 00:14)

TLP on Chip:
HT/SMT and CMP

Last class sometime, we discuss something else. So, tip stands for secular parallelism something we have to be take for after that TLP and MLP that is extraction level parallelism in a importable parallelism. So, treadle parallelism essentially talks about program and that is faulty parallelism sets all portion of a program how to manage the threads popularly and HT stand for hyper threading some of you heard about inutility.

We just the high lights for simultaneous duplicity. So, this 1 this particular term was originally called researcher later Intel modified that in little bit hyper city. So, will see the slippen is between what was actually proposed in it was a finally, implicated it make the processor CMP stands for chip multi processor use another 1 doing multi threading on chip. So, you will look at all these architectures.

(Refer Slide Time: 02:54)



SMT

- Simultaneous multithreading (SMT)
  - Basic goal is to run multiple threads at the same time
  - Helps in hiding large memory latency because even if one thread is blocked due to a cache miss, it is still possible to schedule ready instructions from other threads without taking the overhead of context switch
  - Improves memory level parallelism (MLP)
  - Overall, improves resource utilization enormously as compared to a superscalar processor
  - Latency of a particular thread may not improve, but the overall throughput of the system increases (i.e. average number of retired instructions per cycle)

So, let us start with SMT its stands for simultaneous multithreading. So, here the basic goal is to run multiple threads at a same time. So, the in a size of the term same time, because whatever; we you have seen in a system essentially a and multiple threads are usually time multiplex on the machine here the problem about managing multiple threads at the same time who is off course, wait that the hardware support to be and why is it helpful for why should discuss these things, because these steps in hiding memory because running is blocked still possible schedule radiance structure without taking a contacts.

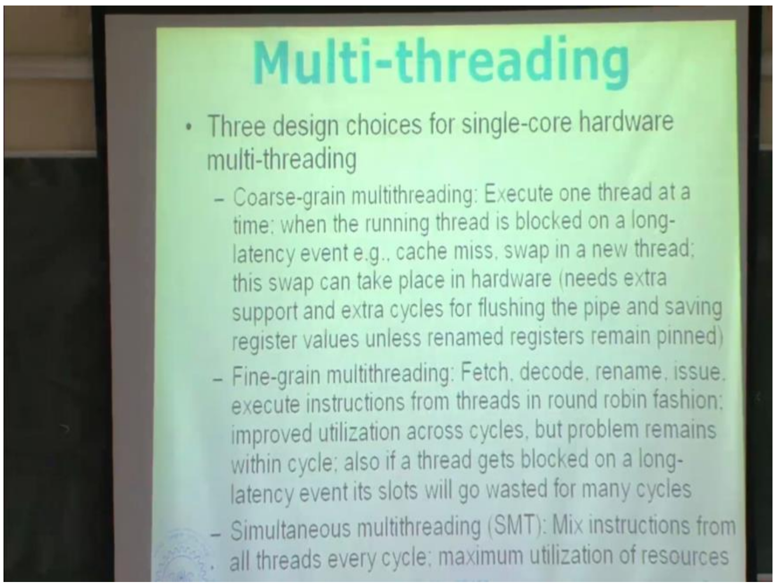
So, 1 following property of the these type of multithreading is that there is nothing always an acting forward pick up the structure and schedule them. So, improves activity memory lessons that because, it essentially allows you to send more memory requires. So, why launch central inputs alone more thread which and a this is a exactly important today's graphics actually explained much largest GPU's for is stands for graphics have if you look at the recent advice recent CPU's coming on to the industry, if you have the more than thousands such steps.

So, the 1000, so will not be that is before 8 steps basic concepts is same off course, scaling thousands threads that is requires certain amount of a knowledge because others are will be will be have main problems like scaling. So, learn get they will livid small number thread is a see what the basic concepts ok.

So, improves research synchronization honestly as compared to super scarer process because super scarer process it is doing a as a single thread and going to the threads at the cash mess it the last cash hierarchy extra option will be top there will be block everything. We just sees what is happening today, here both really knowing is that time to package many to some way. So, it arouses you which have want thread put you happen a 1 cashmeres willing a thread and why not cashmeres.

So, latency of particular stage, but the overall through put of the system will be increases because of the unit is that a unit of time will be that is all there is an average number of retired instruction per cycle.

(Refer Slide Time: 06:40)



Multi-threading

- Three design choices for single-core hardware multi-threading
  - Coarse-grain multithreading: Execute one thread at a time; when the running thread is blocked on a long-latency event e.g., cache miss, swap in a new thread; this swap can take place in hardware (needs extra support and extra cycles for flushing the pipe and saving register values unless renamed registers remain pinned)
  - Fine-grain multithreading: Fetch, decode, rename, issue, execute instructions from threads in round robin fashion; improved utilization across cycles, but problem remains within cycle; also if a thread gets blocked on a long-latency event its slots will go wasted for many cycles
  - Simultaneous multithreading (SMT): Mix instructions from all threads every cycle; maximum utilization of resources

So, there are 3 design choices for single core hardware in multithreading. So, we have talking about just 1 that is what the single core topical and handle what are do unthreading 2 steps 1 is 2 threads is a. So, options 1 is coarse grain multithreading which means execute 1 thread at a time when the running thread is blocked on a long latency event like cash miss swap in a new thread this swap can take place in hardware so; that means, that will needs extra support and extra cycles for flushing the pipe and saving register values unless renamed registers rename remain pinned.

So, what you manses that a running a thread. So, takes a cache miss currently we look at the register maps the sudden register maps. So, there will watching on the register suddenly having a maps put. So, you have what you wanted do is that you want to quote this said to see for sometime to a cash miss and will new said and runs. So, the things will be well off course, here to change the program because that is what new threading that is it which new said what going animate the mesh up that the registers use for the old threads a set properly. So, let us come back to the compact enrich.

So, this can tools 1 is that can copy the registers into memory then the thread you could copy the memory copied back to register. So, that other option is that you already, have certain register renamed regarding to register set idea to current physical values you said a well these is the register’s will rename allocated they cannot be used by any other threads that is as to be obvious all saving mechanics you do not have it was the more that these are external quarter.

So, not be cash paid; they remain as a method and that is weight exactly what today’s will not get loads in register that which is while finally, if use had case of and this is the reason and need is because support 1000 yes you have to say the map yes you have to look at that unless you have to support multiple maps. So, will you have see actually these kind of the registers could actually have to multiple. So, abide a saving a of a anywhere you have to check right.

So, here, so why it is called coarse grain multithreading, because what will doing is that we are really not mixing threads instruction very fine way. let a 1 a time only things that upholding system who well doing a threads in which, but a most wasted way that was you are saving system what happens; here are a actually of the threads right. So, fine way threads which is, but you are still ready goes to what abide system you will you are a making what we blocked ok.

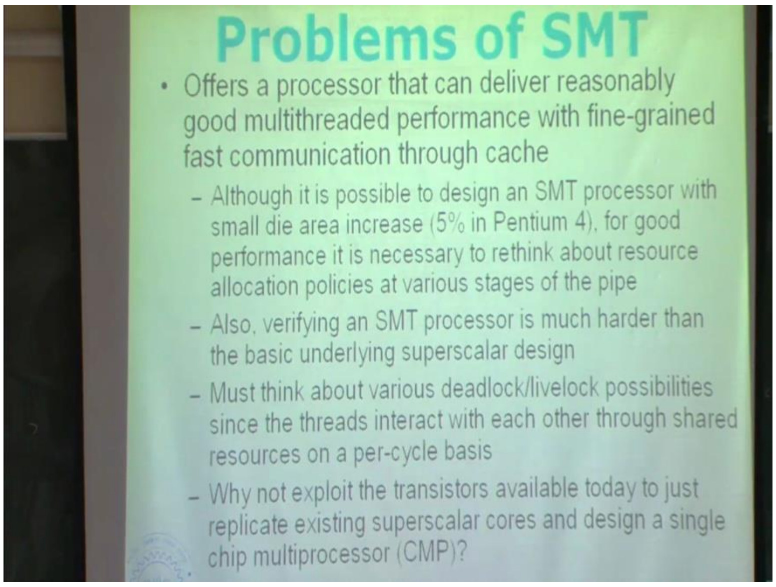
The next is we can proving that is fine grain multithreading that is fetch decode rename issue execute instructions from threads in round robin fashions. So, what it means it that this cycle I have will fetch decode rename issue execute instruction from threads 1 next cycle fetch decode rename issue instructions from thread 2 and. So, would cycle through the threads. So, notice that here I am not I just have single fight here is I am just I am just switching the program congress to make sure what instruction make in.

So, may be walls instruction may be this side to a cycle thread to instruction, so every cycle program so here whets we got the problem. So, its improved utilization across cycle not doubt about this, so into extracts. So, here what may happen is that because of several research are there you will may not be able to explain the entire researcher are there here will solve the problems it across threads sorry across cycle, but the problem remains within a cycle.

So, within a single cycle what may happening in the certain cycle the current is will be get blocked the final in these cycle take on only 2 instruction to be issued the that time is 2 instruction the suppose the divided into 4 issues clubs; that means, 50 percent issues are also if a threads gets blocked on a long latency event its slot will go wasted for many cycles it was what because it is a statistics. We are say that 2 threads what cycle to be given thread 1 said to be thread 2. So, thread wise thread 1 is a cash miss all the cycles thread 2 is a.

So, third solution is simultaneous multithreading here go to do is mix instructions from all threads every cycle. So, what you do is that you first of thread 1 you find that I can factual instruction this cycle within a third instruction of the. So, I do not know what you know, but suppose I am actually instruct for 2 threads finally, have to takes 1 instruction for 2 threads and 1 more concept is every and every stage I will be learn as a my resources as much as possible by missing instructions from all threads. So, I get a maximum utilization of resources and notice that all these case I have no comments alright.

(Refer Slide Time: 13:09)



Problems of SMT

- Offers a processor that can deliver reasonably good multithreaded performance with fine-grained fast communication through cache
  - Although it is possible to design an SMT processor with small die area increase (5% in Pentium 4), for good performance it is necessary to rethink about resource allocation policies at various stages of the pipe
  - Also, verifying an SMT processor is much harder than the basic underlying superscalar design
  - Must think about various deadlock/livelock possibilities since the threads interact with each other through shared resources on a per-cycle basis
  - Why not exploit the transistors available today to just replicate existing superscalar cores and design a single chip multiprocessor (CMP)?

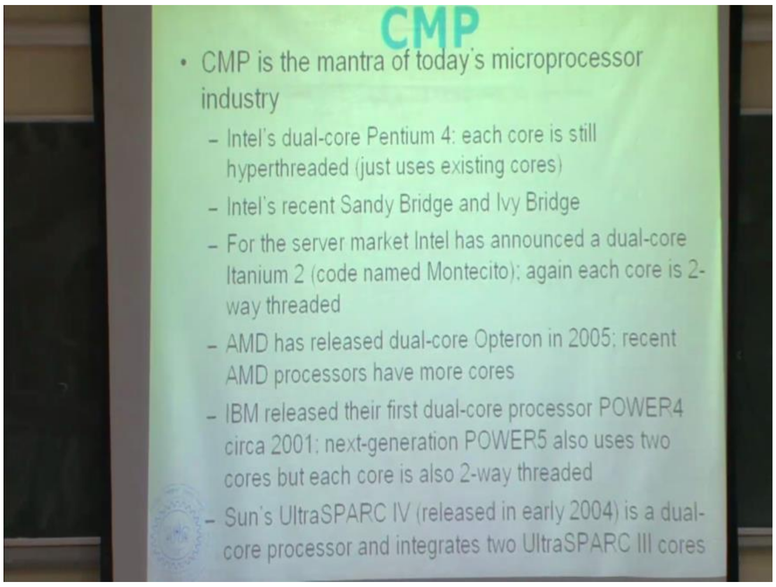
So, what the problems, so it offers a processor that can deliver reasonably good multithreaded performance, with fine grained fast communication through cash it is now what to have to in the I run thread subsequently other thread can see a taken of these thread to be a fashion. So, communication become extremely past although it is possible to design an SMT processor, with small die area increase for example, we talk about podium 4 ht which had only 5 percent extra client to compel to 4 for good performance it become necessary to rethink about resource allocation policies at various stages of the pipe.

So, we talks about after policies also verifying an s m t processor is much harder than the basic underlying superscalar design because essentially you got doing is a act as soon as bringing 2 threads you have look at the clauses of the which to a threads expects, because it, so and that. So, we must think about various deadlock live lock possibilities since the threads interact with each other through shared resources on a per cycle basis for example, we have to think about.

So, these is the problem that operative system designer actual think about now is the hardware designer is a think about to do this to the what will happen is there the thread may only a resource as other thread will and there will be cycle of resources right see now, so these things. So, these things and. So, the other option that to start thinking about addition to this was that why not exploit the transistors available today to just replicate existing superscalar cores and design a single chip multiprocessor.

So, why not install the transistor just think at coarse to a design a transistors just replicate of his single chip and we have the multiprocessor right. So, we have, so many transistor, so that that make to the concept of chip and that is want you find it ways its essentially the way for many.

(Refer Slide Time: 15:25)



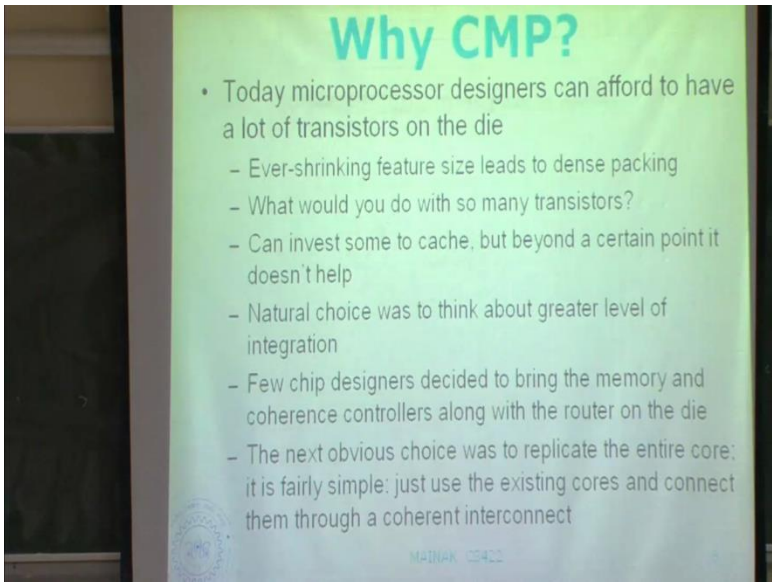
CMP

- CMP is the mantra of today's microprocessor industry
  - Intel's dual-core Pentium 4: each core is still hyperthreaded (just uses existing cores)
  - Intel's recent Sandy Bridge and Ivy Bridge
  - For the server market Intel has announced a dual-core Itanium 2 (code named Montecito); again each core is 2-way threaded
  - AMD has released dual-core Opteron in 2005; recent AMD processors have more cores
  - IBM released their first dual-core processor POWER4 circa 2001; next-generation POWER5 also uses two cores but each core is also 2-way threaded
  - Sun's UltraSPARC IV (released in early 2004) is a dual-core processor and integrates two UltraSPARC III cores

So, right in 90's when efforts people on this came out for the they name which important that is. So, here some list of client that happen in industry know exhaust it we just you examples. So, Intel's dual core Pentium core each core was hyper threaded actually we have a 3 multithreading core and it is a just existing core Intel's recent sandy bridge and ivy bridge they have more cores on chip for the server market Intel has announced what to write in a 2 feature core named Montecito again each core is 2 way threaded AMD has realized dual core option in 2005 in recent AMD processors has more cores IBM released their first dual core processors power and power 4 in 2001.

We talk about power 4 little bit makes power 5 also see this 2 core, but each core is 2 way threaded. So, will also talk about power 5 Sun's ultra spark 4 releasing in early 2004 is a dual core processor integrates 2 ultra spark 3 cores. So, pretty much you know you look at any chief manufacturer they have gone this 1 alright.

(Refer Slide Time: 16:42)



Why CMP?

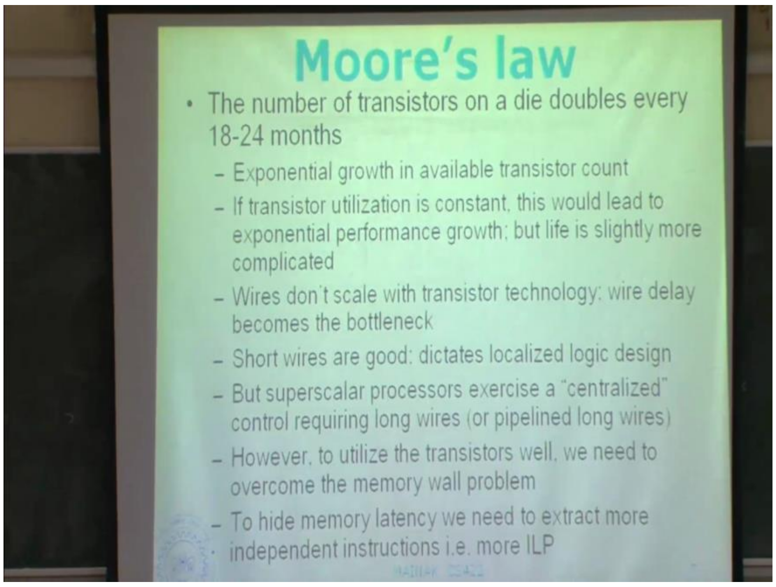
- Today microprocessor designers can afford to have a lot of transistors on the die
  - Ever-shrinking feature size leads to dense packing
  - What would you do with so many transistors?
  - Can invest some to cache, but beyond a certain point it doesn't help
  - Natural choice was to think about greater level of integration
  - Few chip designers decided to bring the memory and coherence controllers along with the router on the die
  - The next obvious choice was to replicate the entire core; it is fairly simple: just use the existing cores and connect them through a coherent interconnect

MAINAK CS422

So, let see what the designs problem here are so first of all we do this. So, 1 thing, I have already entered that that is you have challenges. So, question was how we imply this 1 ok. So, today micro processor designers can afford to have a lot of transistors on their die like for example, if you look at the recent GPU's they have more than in a lot transitions and a ever shrinking feature size leads to dense packing. So, what we would do with, so many transistors. So, 1 obvious solution in that I just use for sometime was said they invested lot of transistors to cash it ok.

But, beyond the point what happens is that you will going to get the return for what you investing natural choice also think about few chip designers decided to bring the memory and co hence controllers along with the router on the die. So, 1 example was compeller example. So, what you do is that you could just if you need any. The next obvious choice was to replicate the entire cores just use the existing cores and correct them for alright.

(Refer Slide Time: 18:20)



Moore's law

- The number of transistors on a die doubles every 18-24 months
  - Exponential growth in available transistor count
  - If transistor utilization is constant, this would lead to exponential performance growth; but life is slightly more complicated
  - Wires don't scale with transistor technology: wire delay becomes the bottleneck
  - Short wires are good: dictates localized logic design
  - But superscalar processors exercise a "centralized" control requiring long wires (or pipelined long wires)
  - However, to utilize the transistors well, we need to overcome the memory wall problem
  - To hide memory latency we need to extract more independent instructions i.e. more ILP

So, the for all these things moors law how many of you heard of this. So, what he says in that the number of transistors on a die doubles every 18 to 24 months alright. So, that is; that means, an exponential growth in available transistor count if transistor utilization is constant this would lead to exponential performance growth, but life is slightly more complicated. So, some of you might have seen these as the modes of that you know performance doubles every months that is not.

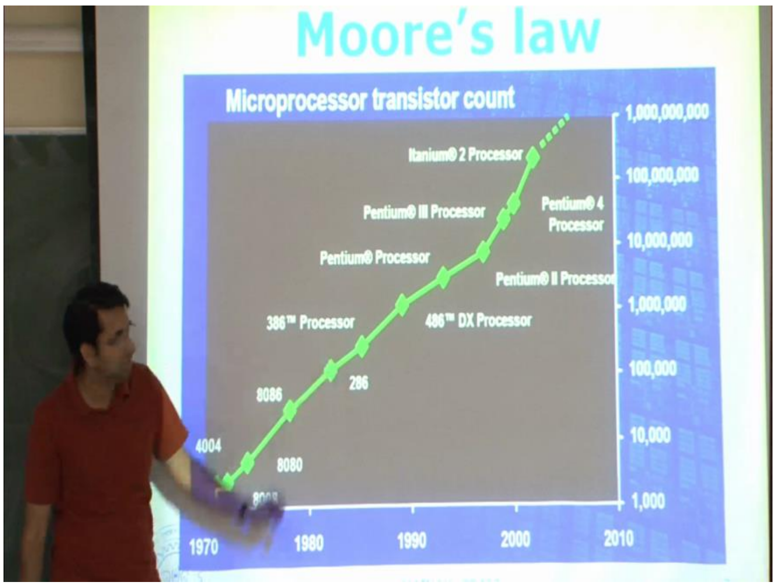
The problem is that where transistors technology. So, where delay becomes the constant our transistors get passed with the communication becomes more costly. So, short way as a good which dictates the localized logic design you do not want long way as you distributed logic and all these things alright, but the superscalar processor exercise a centralized control requiring long wires because pretty much if you look at the issue here that controls everything. So, the will control the central part it decides what instructions we issue and what instructions we like to rectify.

So, a power in a pipe line because by pass and all those requirements; however, to utilize the transistors well we need to overcome the memory wall problem. So, this we talked about the problem that your processors most of kind all very important to hide memory latency.

We need to extract more independent instructions that more instructions that is only because it saying that what you can do is that you have a study for a instructions come you want to hide during this entire time instructions and executing the essentially why extracting more independent instructions right. So, what that requires mostly you heard that because if you do not have molding flights ok.

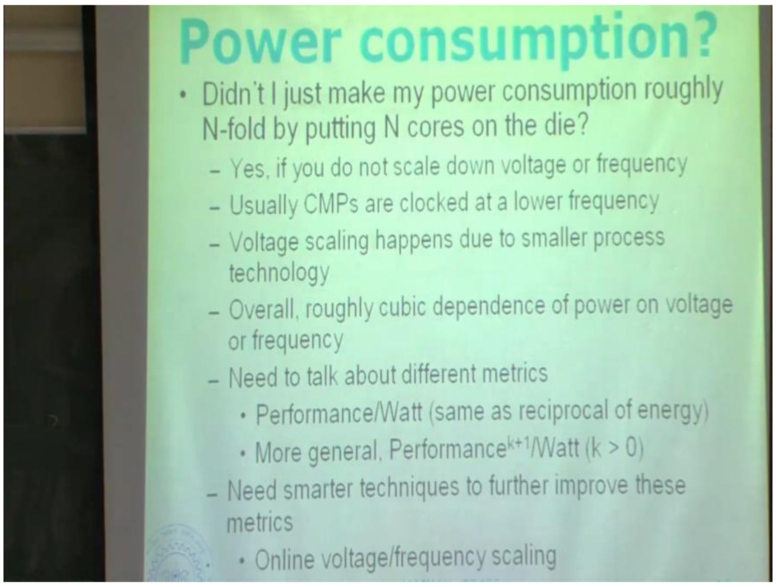
But, for that we need bigger POB which, in turn requires a bigger register file also we need to have bigger issue queues to be able to find more parallelism none of the structures scale well main problem is wiring. So, they require long wiring. So, best solution to utilize these transistors effectively with a low cost must not require long wires and must be able to leverage existing technology. So, CMP satisfies these goals exactly which use the existing processors and invest transistors to have more of these on chip instead of trying to scale the existing processor for more. So, multi core was actually a natural choice go with this direction ok.

(Refer Slide Time: 21:40)



So, here we have the chart that shows this slightly actually where off course thirty today this 1 this 1 only shown up to I think these are these are chart of intervals processors. So, what the point is that y access the large scale x axis is linear scale this line is. So, that actually substantial it is not totally linear, but more or less.

(Refer Slide Time: 22:10)



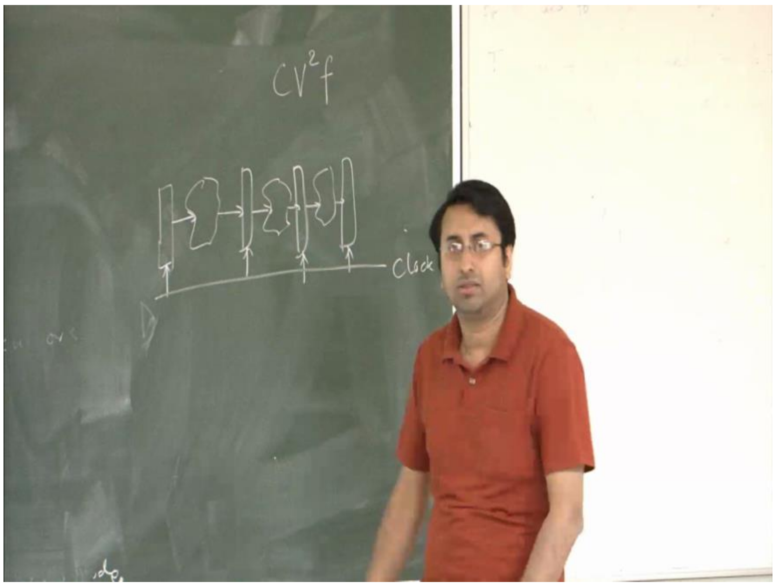
Power consumption?

- Didn't I just make my power consumption roughly N-fold by putting N cores on the die?
  - Yes, if you do not scale down voltage or frequency
  - Usually CMPs are clocked at a lower frequency
  - Voltage scaling happens due to smaller process technology
  - Overall, roughly cubic dependence of power on voltage or frequency
  - Need to talk about different metrics
    - Performance/Watt (same as reciprocal of energy)
    - More general, Performance^{k+1}/Watt ($k > 0$)
  - Need smarter techniques to further improve these metrics
    - Online voltage/frequency scaling

So, now ensuring that you are done this you put multiple task on a single chip, but then did not I just make my power consumption roughly n fold by putting n cores on the die answer is yes if you do not scale down voltage or frequency. So, usually chip multi processors are clocked at lower frequency you must have noticed this that has number of cores increases normally the processor is not get the slower to the reason is actually, this voltage scaling happens due to smaller process technology because physics tells you that if I make a transistors that smaller I have to reduce the supply otherwise transistor will break down.

So, you have to scaled on voltage you cannot that is you start reducing the. So, which means if you keep on the higher speed there will be fundamental problem in designing pipelines. So, is that, so have a pipe lines where is chocks.

(Refer Slide Time: 26:01)



So, have a pipeline these are my pipeline an here I found. So, this is the input this is output like etcetera. So, that I am doing is that I have scale down the voltage of these transistors that implement a logic or else that locking the transistors is there any problem. So, I this is the clock supply is gone down I have not drop down the top figures and I say that if you bring down the supply is it will be go down you transistor will naturally operate at a slower rate what will happen.

Sorry

Exactly stay is do not be copied. So, even report is computation complicit side that those of this latch and I will get a wrong value from the latch alright. So, you have bring down frequency for other reasons, but not the less to keep your part assumptions you also have to lower the frequency. So, what happens is that you know that energies in this square route for your basic physics alright and power consumption is energy part any time. So, what you get is the supportiveness frequency that is the more or less linearly related we were some.

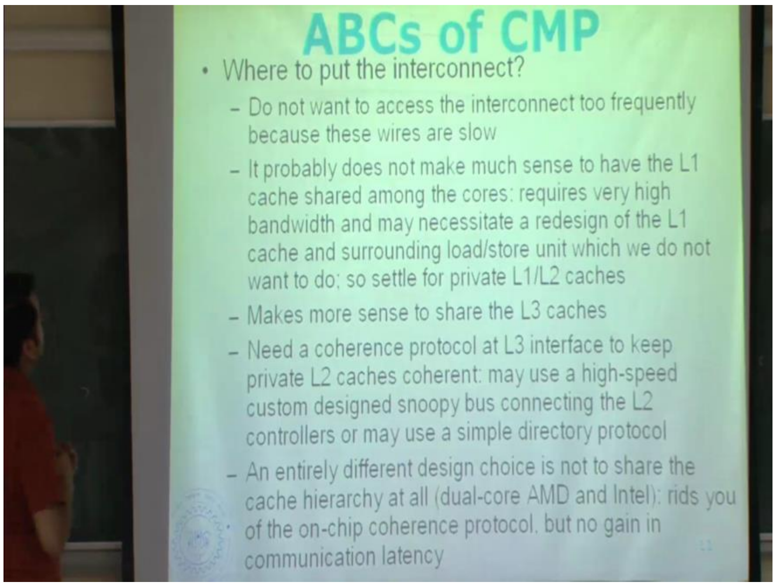
So, what we get is of a dependence of power on voltage frequent that is what you have to get. So, which is good because as your transistors saying you have to bring down the supply voltage automatically you get a cubic saving ok. So, that actually helps here which means that if you have cores you do not have to get a 1 way clock because you are gaining something here actually at same block that can we put point here is that when you designing these multi core processor you cannot just. So, what we do is that we change a electric.

We talk about performance per watt just 1 thing that occurring to which is actually same alright why because your performance is all about all over time right because we said that if the time goes down performances increases right. So, performance all other time, so these is essentially time for watt. So, these were essentially tells you that if we have less energy it is good alright. So, is somewhat, because all we need to do is to. So, about general metric is performance.

So, energy we can now talk about energy times time that is often following for example, if you set k equal to 1 that what exact energy multiplied by execution time ok. If you set k equal to 2 energy execution time square. So, that will put different affairs on energy and we needs smarter techniques to further improve these metrics for example, today all processors implement online voltage frequency scaling which dynamically monitor your processor performance and decides whether; you need to rise he true voltage or I can actually stay it down and see performance and when we stay on any of the environment that used by on dynamic voltage frequency scaling.

If you interested, I can use the weapons when the raw idea is there you dynamically monitor your processors performance try to establish an correlation performance not the potential hold time and actually apply it and then. So, in all the processors we actually discuss 1 implementation on this would not much on that value.

(Refer Slide Time: 28:47)

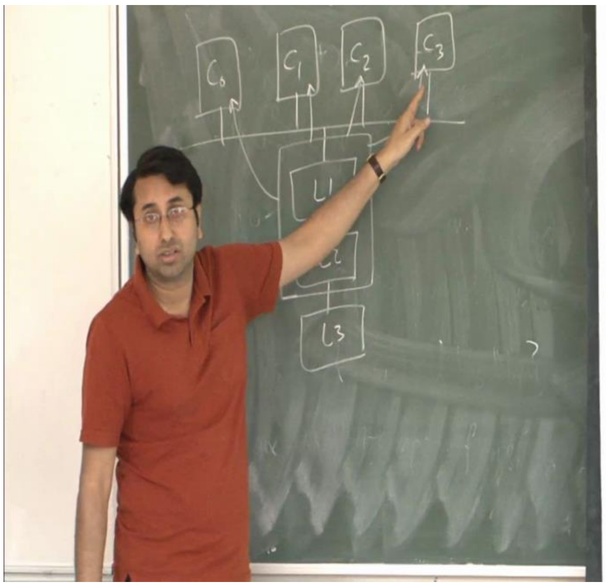


ABCs of CMP

- Where to put the interconnect?
  - Do not want to access the interconnect too frequently because these wires are slow
  - It probably does not make much sense to have the L1 cache shared among the cores: requires very high bandwidth and may necessitate a redesign of the L1 cache and surrounding load/store unit which we do not want to do; so settle for private L1/L2 caches
  - Makes more sense to share the L3 caches
  - Need a coherence protocol at L3 interface to keep private L2 caches coherent: may use a high-speed custom designed snoopy bus connecting the L2 controllers or may use a simple directory protocol
  - An entirely different design choice is not to share the cache hierarchy at all (dual-core AMD and Intel): rids you of the on-chip coherence protocol, but no gain in communication latency

So, some of the basics about chit fund processor the following try to put in that way correction between the processors right. So, we do not want to access interconnect too frequently because, these wires are slow that was the fundamental reason why you want a chit from the long wire desires ok. So, it probably does not make much sense to have the L1 cache shared among the cores. So, what is saying is that.

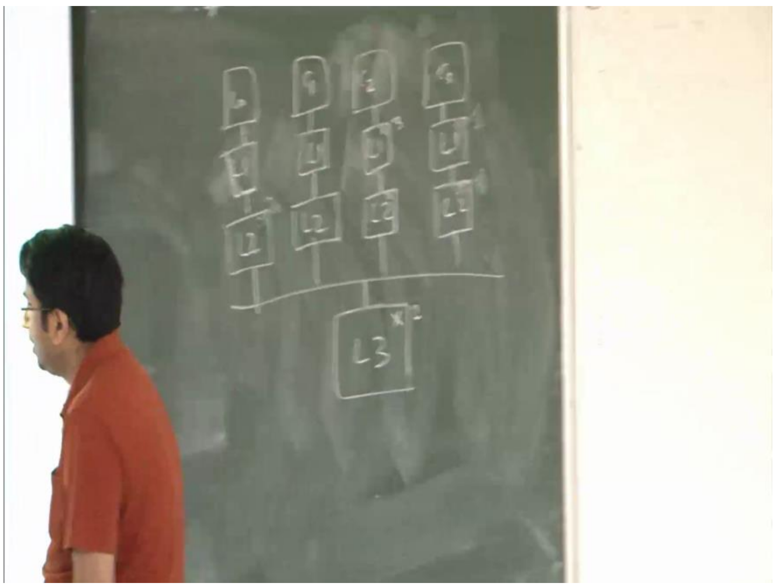
(Refer Slide Time: 29:31)



So, I have these cores here let us say we have 4 cores and it content the pipe line for executing results alright. So, first option talks about putting the interconnected connect the cores and then put the cache we have L1 L2 L3 etcetera alright. So, essentially what I am saying is that I have shared L1 which is a good idea the answer is no, because every cores instruction from every core will have to and that clearly depicts this whole process alright. So, it requires very high bandwidth and necessitate a redesign of the L1 cache and surrounding load store unit which we do not want to do because, what you require is on L1 cache. So, that you can brand access to all of them concurrently ok.

So, what people do is that they settle for private L1 L2 cache. So, what for means is that I need a private hierarchy here that can help for most of the load to operation alright? So, what usually today we do is that we put L1 into here. So, they distribute this alright. So, each core gets private L1 L2 cache and you share the L3 cache. So, what will happen is that there will be a variable x. So, which is deciding in all the L2 caches here alright? So, in 1 of the core is right to value of that is you tell the others. So, that the value has changed alright.

(Refer Slide Time: 31:51)



So, 1 may use high speed custom designs will be connecting the L2 controllers or may use the. So, let me explain this alright. So, image that you have a variable in a program which is shared by all the processors. So, we are suppose currently x as deciding here and here and the values 2 x for the variable x alright. So, now, a program says that C2 has the right to the variable axis alright. So, we will access every 1 is the cache access L2 we will access l three. So, x will be here if we have inclusive cache alright. So, L3 to get x and in cache as here and here hierarchy and generates a new value for x which is 3.

So, here it is 2 this is 3 is also 2 fine. So, now, what will happen is that. So, this guy may want to right x alright copies x here generates a new value which is 5 and I can keep on doing like this eventually I can show you that all processors will have the variable x in. There is the variable x everybody sees a new value of x and your program is to make sure that while this might happen. So, forget about this 1 while this right happens you could tell these 2 guys that is a new value been generated for x you should be aware of it alright.

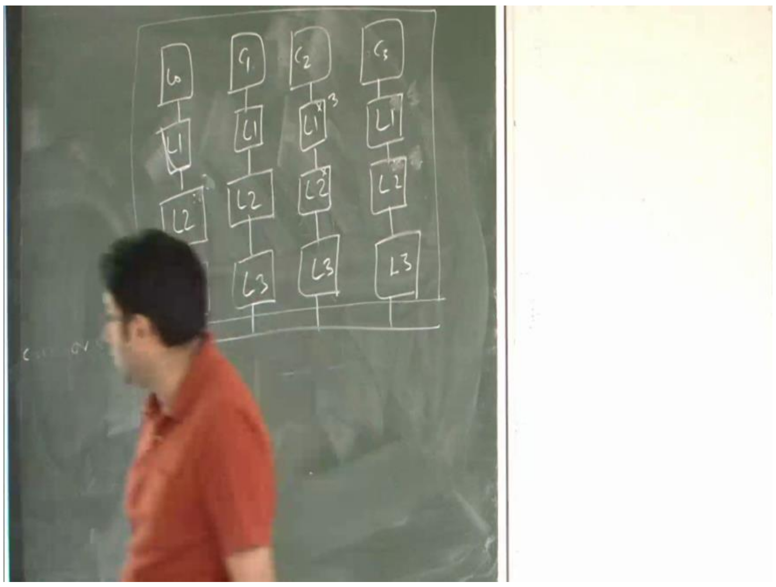
There is 2 doing that 1 is called the smoothly protocol what it does is that whenever x tries to access from the share L3 cache it will launch the address of the variable on the process every else which loops the process pickup the address and make a comparison of the values with all the address here. So, essentially not all the addresses we pick up the address index into L2 cache and see if that particular cache block is and if it is present and if they launched command was actually a right command I will be box from here alright.

So, in this case these 2 guys will actually match cache. So, that there will be always modified value of x alright. So, that guarantees this point and this is now wrong subsequently c not will access x if you goes an access x from here assuming that right back cache and this value only when C2 the block alright make sure that whenever somebody tries access on it gets directed to the on current owner of the particular variable. So, in this case what will happen is that C not tries to access x will not everybody this guy will get a match and actually will generate a new value on the process which we picked up while.

So, that is what which make sure that the value, now here the problem is that if you have the large number of processors will have to whenever a right is generated is to tell the everybody that somebody right in to this particular variable. So, that requires the broadcast, but here what we really want to do is we just want to tell these 2 guys because others do not care because others do not have to. So, what you could do is that you could met in a directory system which design will go and look up figure of that go only c not and C3 all the processors and you send 2 point to point messages to these 2 guys saying that alright that is called the directory protocol.

So, I am not going to much more detail about this protocols, but this is the basic thing that requires and entirely this is different desired choice is not to share the cache hierarchy at all. So, what you could do is that you could pull L3 up also here. So, nothing stops you from designing, where each processor also has the L3 and then; so there is nothing shared alright.

(Refer Slide Time: 37:42)

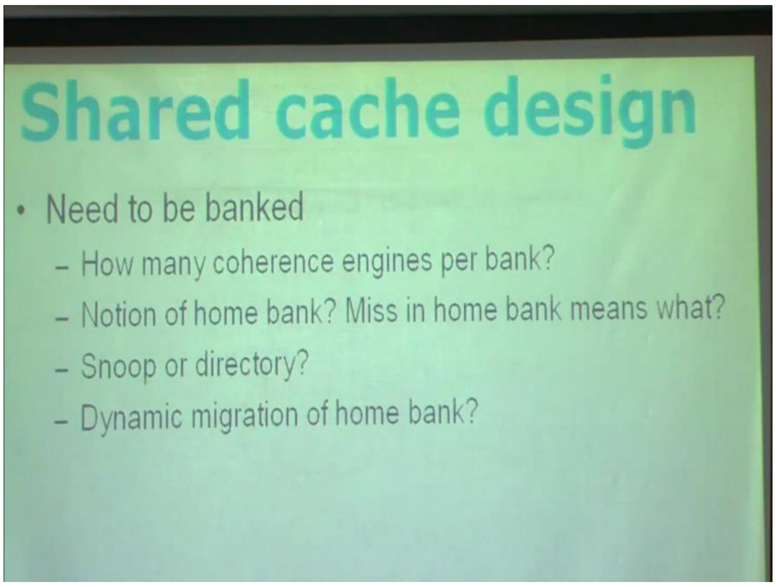


So, hardly dual core AMD and Intel actually you get this it reads you of the, but your gain in the convocation previously, what you could do is that you could share certain variables in L3 cache very first c not to drive something and right; it back to the L3 C3 could consume that to L3 and now you have to all the interconnect go all the way up to get a back. So, you today you will probably not find any such that either a complete private hierarchy the last is always shared ok.

Yes in this also.

It is not exactly because normally this particular interconnect will be outside and this protocol will keep the L3.

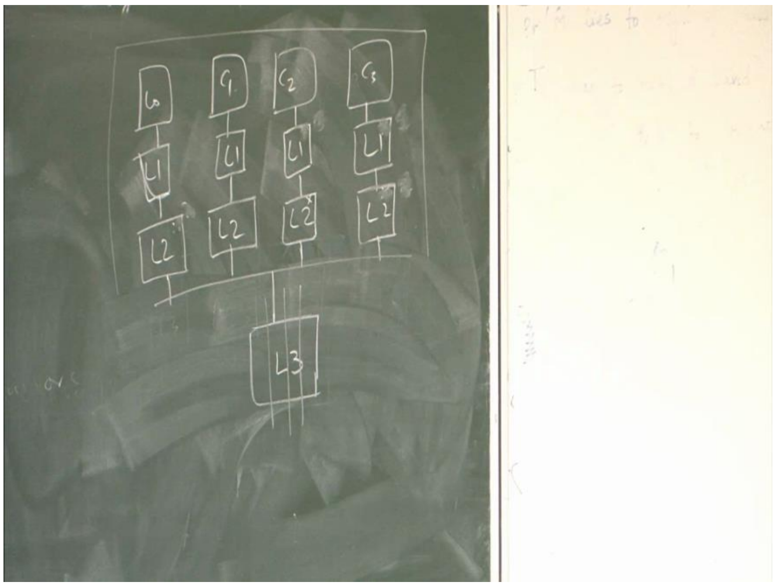
(Refer Slide Time: 39:06)


Shared cache design

- Need to be banked
  - How many coherence engines per bank?
  - Notion of home bank? Miss in home bank means what?
  - Snoop or directory?
  - Dynamic migration of home bank?

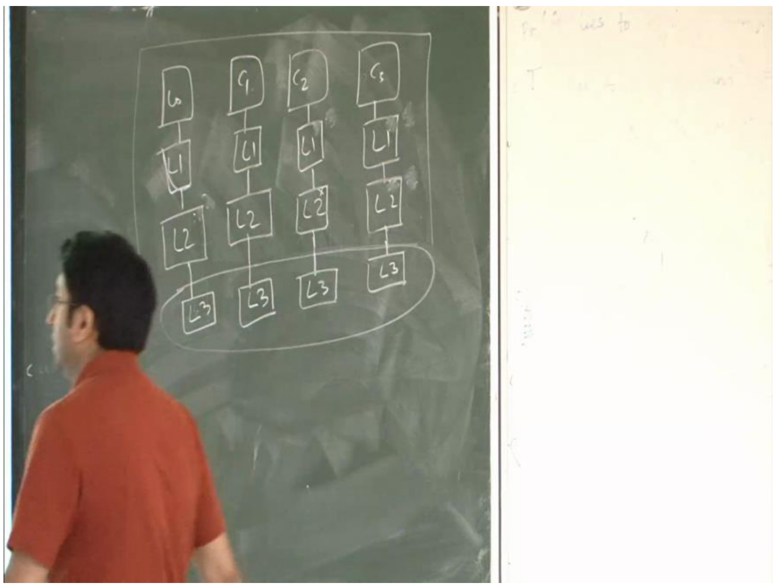
So, if you do have a shared last cache need to be banked we have talked about that already while because, otherwise it increases vacancy. So, then we have to talk about how many coherence engines per bank that is. So, having more engines may actually speed up your coherence processing now the question is that if you have bank architecture.

(Refer Slide Time: 39:35)



Let me have the and let suppose that this is actually a banked L3 caches it may be alright. So, now try to put a address on the box if we have not the globally responded. So, then may be 1 of the L3 banks actually has a. So, the cache of the particular bank that holds the cache it is called home bank cache. So, it is a simple function which takes the address and figures out of home bank. So, it just a partially approved and a miss in the home bank means that the block is. So, this all we talked about directory is there any reason why I might want to change the home bank of cache yes right exactly see if you have an interconnect which is said that.

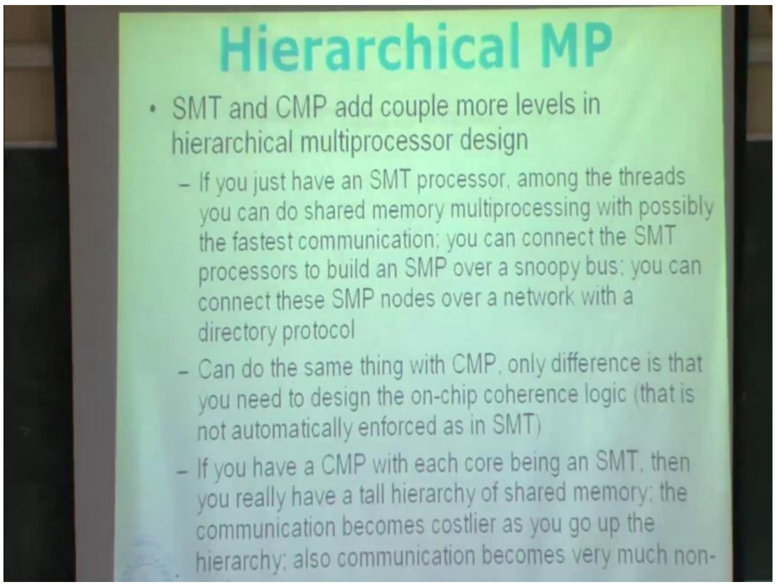
(Refer Slide Time: 41:39)



Let me show you 1 example and I’ll just showing it is a logical diagram. So, of course, the blocks will be distributed on the ring alright. So, will be there, so let us assume that each ring stop has a bank of l three. So, we have a 4 bank L3 and the banks distributed like this you have c not is that it means a particular cache block whose home is here and let suppose that scenery is unable to catch this drop very efficiently.

So, then it make hence actually change the home bank from here to here, but this can only be done dynamically at a monitor in the axes banking. So, there are proposals which actually does this if we do dynamic migration at the if you can figure out that the particular page that actually you can move the entire page you can change all the home of all the cache block in that page from here to here.

(Refer Slide Time: 43:03)

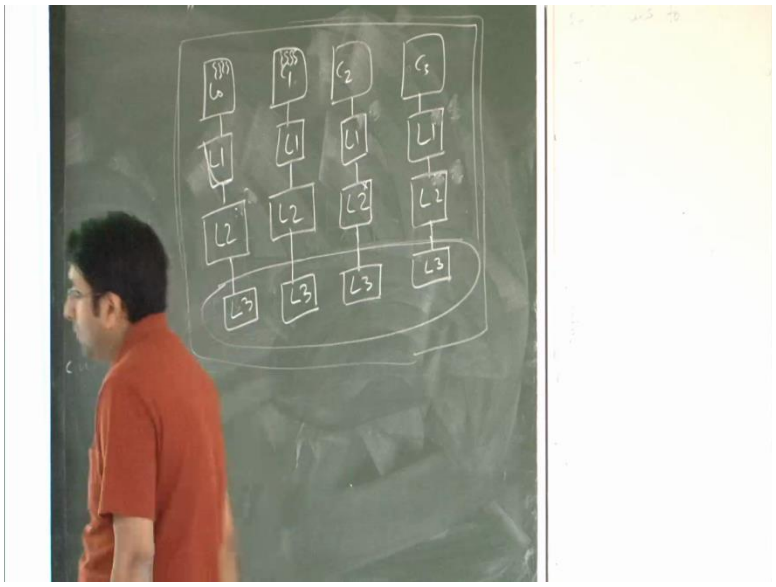


Hierarchical MP

- SMT and CMP add couple more levels in hierarchical multiprocessor design
  - If you just have an SMT processor, among the threads you can do shared memory multiprocessing with possibly the fastest communication; you can connect the SMT processors to build an SMP over a snoopy bus; you can connect these SMP nodes over a network with a directory protocol
  - Can do the same thing with CMP, only difference is that you need to design the on-chip coherence logic (that is not automatically enforced as in SMT)
  - If you have a CMP with each core being an SMT, then you really have a tall hierarchy of shared memory; the communication becomes costlier as you go up the hierarchy; also communication becomes very much non-

Now, the SMT and CMP add couple more levels in hierarchical multiprocessor design. So, what may happen now is that if you take this And say that well each core now has several threads on.

(Refer Slide Time: 43:16)



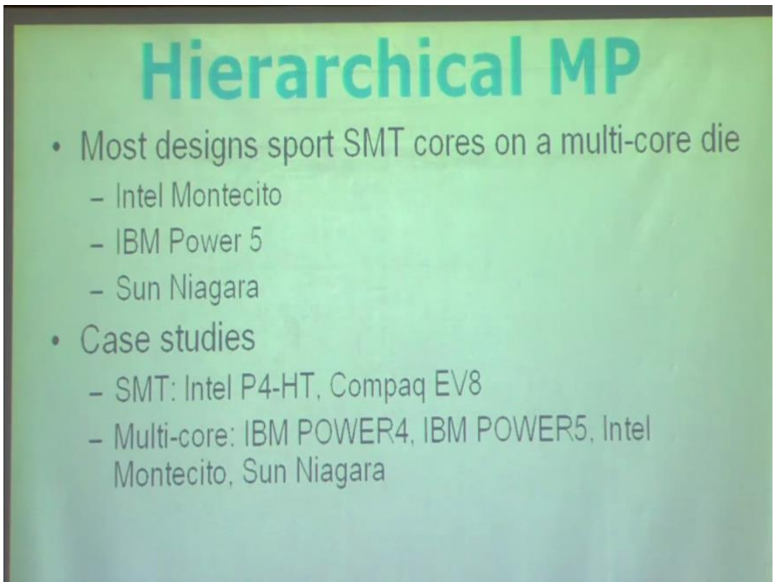
So, that these all are SMT code. So, if just have an SMT processor that is this much or may be this and this alright among the threads you can do shared memory multiprocessing with possibly the fastest probably gets why because, this thread right something to L1 the other thread will consume directly from l. You can connect that

SMT processor to build a s m t over a sooty bus. So, like this for example, I have connected 4 SMT cores over a sooty bus here if is a ring it could be pass out and you connect these SMT boards over a taking this whole thing in correcting few of these over a new draw alright. So, that I get a pretty large multiprocessor.

For example if I. So, 1 single loop gives me sixteen threads we connect just 4 of these I get 64 threads, you could the same thing the CMP only different is that you need to design the on chip core is which is not automatically as in this. So, notice that in SMT I do not need any quarries protocol either this might be itself is shared ok. If you have the same thing with each core being a SMT like this the communication becomes costlier as you go up the hierarchy right as you go up it becomes more and more costly.

Because, it takes time also communication becomes very much right. So, here your compiler can give a very good job in actually, producing very good.

(Refer Slide Time: 45:27)



Hierarchical MP

- Most designs sport SMT cores on a multi-core die
  - Intel Montecito
  - IBM Power 5
  - Sun Niagara
- Case studies
  - SMT: Intel P4-HT, Compaq EV8
  - Multi-core: IBM POWER4, IBM POWER5, Intel Montecito, Sun Niagara

So, most designs sport SMT cores on a multi core die like that will talk about these 5 just that is why I listed these 3 also we talk about 2 SMT case studies that is Intel plenty of 4 hyper study and Compaq EV 8. So, this 1 is actually 2 not 464, we talked about 21264. This particular processor got cancelled because, that is exactly when Compaq got bought by and will discuss these 4 I have IBM power 4 etcetera. So, I gone a stop here today.