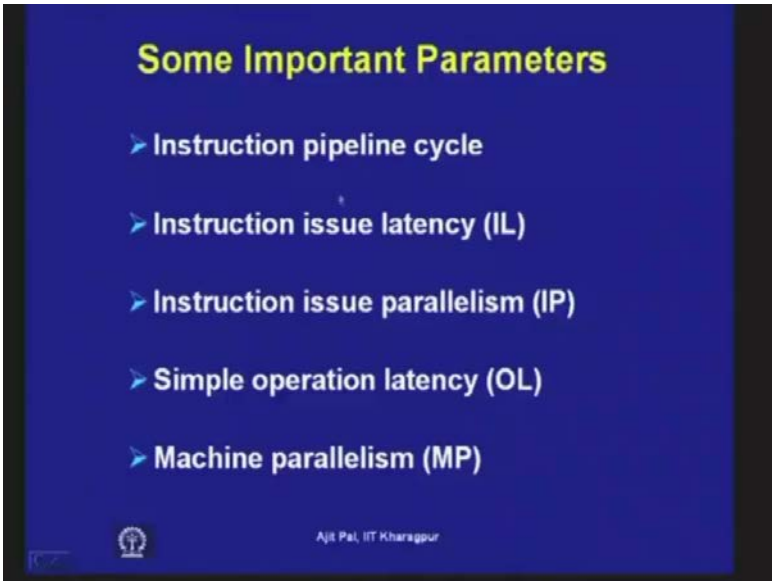
High Performance Computer Architecture
Prof. Ajit Pal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 12
In Quest of Higher ILP (Contd.)

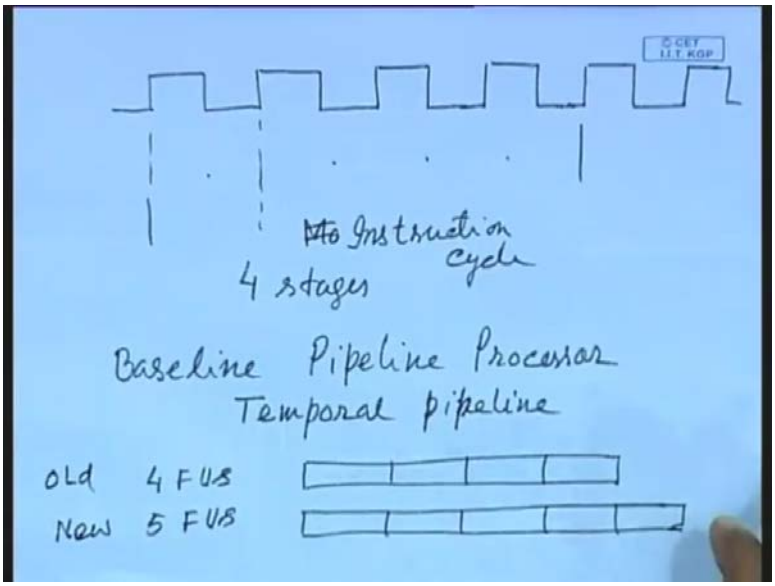
Hello and welcome to today's lecture on in quest of higher ILP. In the last lecture, we have started our discussion on this topic. And we have seen how we can achieve higher instruction level parallelism, and we can have higher I mean machine organization to facilitate that. So, in this lecture, we shall continue on that at topic.

(Refer Slide Time: 01:23)



And before I proceed here are some important parameters I should tell you number one is Instruction pipeline cycle.

(Refer Slide Time: 01:42)



Instruction pipeline cycle essentially tells you the cycle time that means we have seen that the machine is control by clock and whenever you do pipelining then the clock frequency increases and cycle time reduces. So, this is the this is the instruction pipeline cycle and on the other hand it may takes let us assume a machine cycle may take 4 such clock cycle. So, in other words if you trick were 4 stages of the pipeline then 1, 2, 3, 4. So, this will be your machine cycle or the instruction cycle, but pipeline cycle time clock cycle time will be this one.

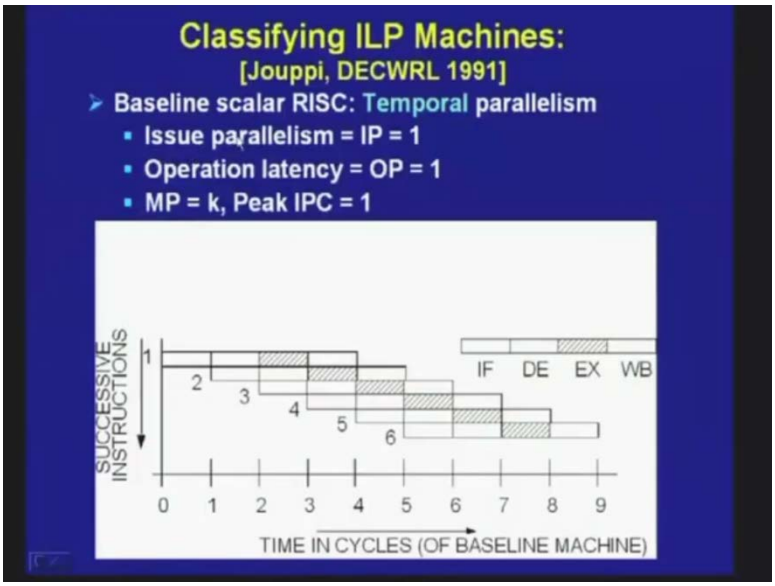
So, this parameter you should remember second is Instruction issue latency. Instruction issue latency is you are issuing instruction one after the other. So, the times require issue and adjust in instruction after a particular previous instruction that tells you the instruction issue latency. That means you cannot issue all the instruction simultaneously so there will be a difference in time or latency in issuing a instruction one after the other.

So, that is defined as the instruction issue latency or a in a abbreviated form I L. Then Instruction issue parallelism I P stands for you know the number of instructions that can be issued simultaneously, concurrently that is actually known as the Instruction issue parallelism. And then the fourth parameter is Simple operation latency that is the processor will be performing operations instruction execution. And the operation latency is the essentially the time required to perform the operation of the particular stage that is the operation latency O L.

And then machine parallelism will tell you the number of instructions in fleche simultaneously at a particular instant of time. For example, if you have got k stage line pipeline then you will find that at when the pipeline is full there will be 4. If there is the case number stage is k. Then k instruction will be in fleche simultaneously as instruction got executed. So, this is the machine parallelism for simple pipeline processor it is equal to k. So, you can start with you know whenever we go for discussing higher and higher I P.

We will see, we will require a reference initially when you discuss the pipeline processor. Our reference was non pipeline processor and with respect to non line non pipeline processor we have discussed various parameters likes speed up, throughput and so on. Now, since we shall be concern with increasing the Instruction level parallelism greater than 1. Our reference point will be the baseline pipeline processor which I have discussed in my previous lectures processor or machine whatever you called it. So, this baseline pipeline processor obviously uses temporal parallelism we have already seen and by doing that it achieves Instruction level parallelism.

(Refer Slide Time: 06:03)

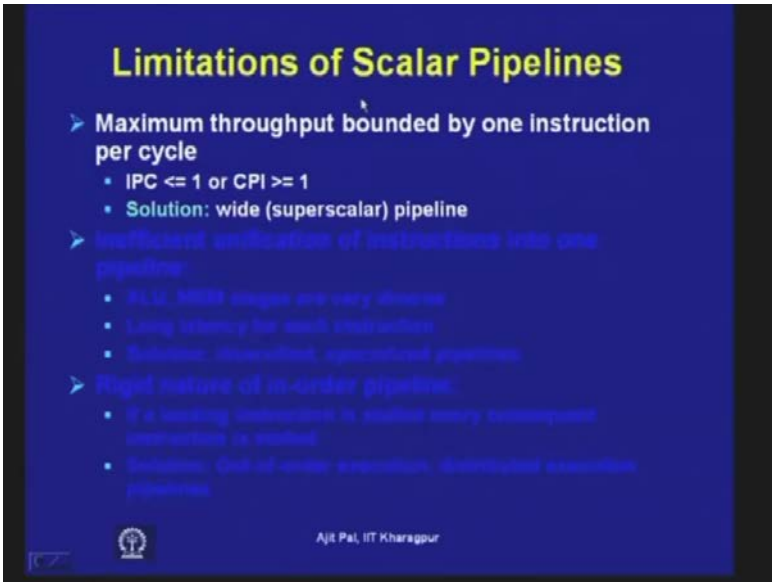


And we have seen that for the baseline pipeline processor which we also called scalar pipeline. Because it issues only 1 instruction at a time that is why since issue parallelism I P is equal to 1. For the baseline scalar R I S C processor another thing we added that is R I S C our discussion will be centered on only this processor that is why this is

invention for the baseline scalar pipeline R I S C processor. The issue parallelism I P is equal to 1 and output latency also 1. Output latency is also 1, because you can see here it generating output at interval of 1 clock cycle. So, the output operation latency that is equal to 1 and the machine parallelism as I have already told which is equal to k, k means as you can see here at any particular instant of time. Since it has got 4 stages this particular pipeline processor maximum of 4 instructions will be simultaneously in fletch during execution.

Of course, they may be indifference stages execution, but you know the maximum number that can be present in the processor time when their in executing during execution is restricted to k, where k is the number of stages in the pipeline processor and we have already seen that peak I P C. I P C is instruction per cycle it is just opposite of C P I cycles per instruction and peak I P C is equal to 1. So, this we shall be use as our reference so there was a paper publish by Jouppi he classified various I L P machine. And this is the first classification where he classified the baseline scalar processor and these are various parameters for that processor.

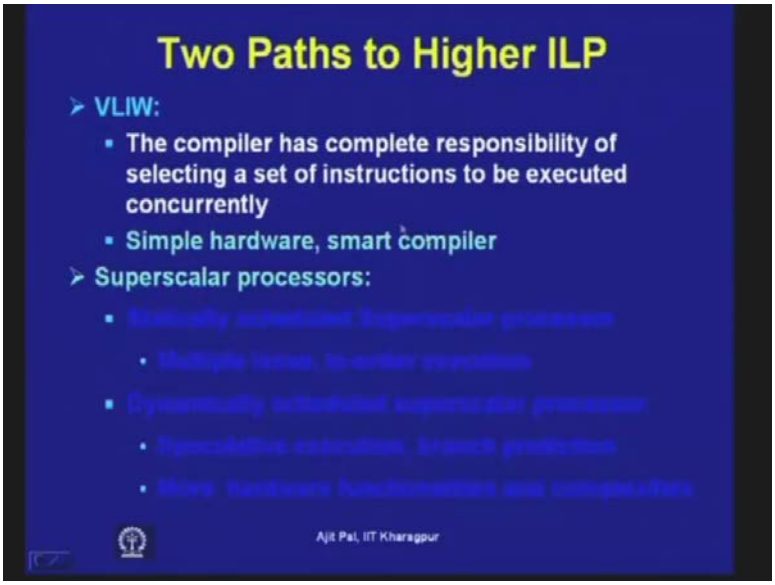
(Refer Slide Time: 08:16)



Now, we have discussed that Scalar pipelines have got many restrictions and first one maximum throughput is bounded by one instruction per cycle. So, I P C is less than 1 or C P I is greater than 1 so what is the solution to this? How we can achieve better performance that means how we can achieve I P C greater than 1 or C P I less than 1?

Obvious solution is making the pipeline wider so; make the pipeline wider which is also known as superscalar. So, by doing that you can increase your I mean you can I PC can be instruction for cycle can be greater than 1 and your C P I can be less than 1.

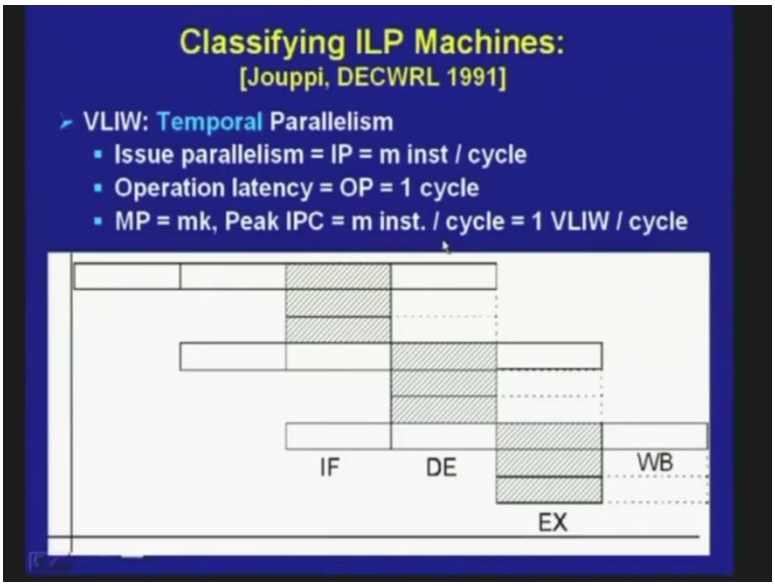
(Refer Slide Time: 09:17)



And we have seen that there are 2 parts in this direction, I have already mentioned it in the in my last lecture; one is your V I LW, the V L IW very large instruction word. And in a last lecture we have discussed in detail how it really works? And we have seen that the compiler has competed responsibility of selecting a set of instruction to be executed concurrently. So, the compiler identifies instruction which can be executed in parallel then several instructions are concurrently issued. Because the behind this of course, will require multiple functional units presenting in the processor.

And for this V L I W processor we have seen that the hardware requirement is simple, because it does not require complex issue hardware. However, it requires a comp smart compiler or it will require what is known as optimizing compiler. That means the compiler has to analyze the instructions after fetching from the memory it has to analyzed they have to kept in a buffer. Then they will be analyzing to identify which instructions can be issued in parallel. And that then they will then it will be compiler will bundle several instruction together and we have already discussed this in detail in the last lecture. And another alternative is Superscalar processors.

(Refer Slide Time: 10:56)



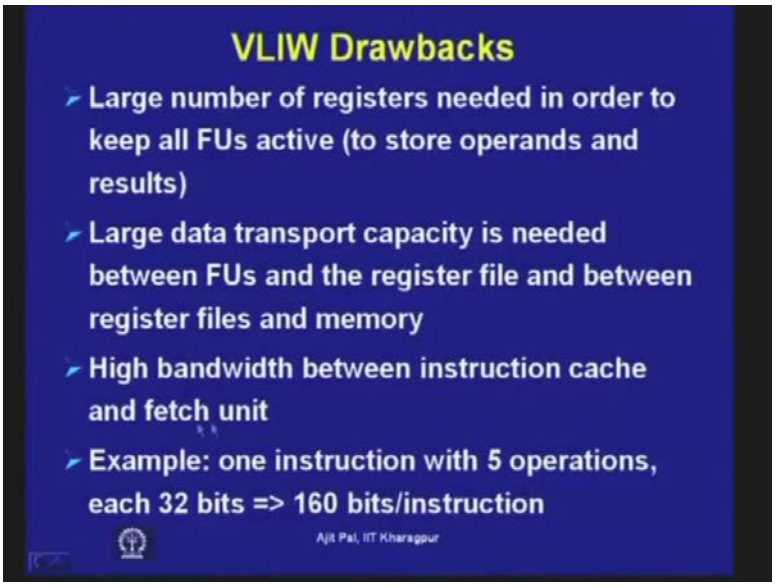
That we shall discuss and we have already mentioned about V L S I. So, based on the classification done by Jouppi, this V L I W has got issue parallelism that is m instruction per cycle, it can issue m instruction per cycle. Obviously, it will depend on the machine parallel the parallelism available in the program. Although maximum possible is m as you can see in this you cannot really fill up all the fields. And as a result the actual parallelism that is possible will be less than m.

So, here also the operation latency c is equal to 1 cycle, because you can see here the after one cycle here the outputs are generated after 1 cycle here also. The issue parallelism is m instruction per cycle of course, m is the maximum value and operation latency is equal to 1 and machine parallelism m is equal m into k. m into k because m is the number of instructions that can be issued in parallel and you can see if all the fills are filled up.

Then you can see that simultaneously you can have m into k means you can see here the pipeline depth is 4. So, 4 into 3 in this particular case you can have 1,2,3,4 into 4 16 there is the possible of 16 instructions getting executed in parallel that means say instructions may in flight. So, that is why the machine parallelism is equal to m into k and a peak I P C instruction level parallelism that is possible is m instruction per cycle or 1 V L I W is instruction per cycle. So, 1 V L I W instruction will comprise m instructions

here m is the number of fields present in the machine. And we have seen that this V L I W processor have got.

(Refer Slide Time: 13:15)

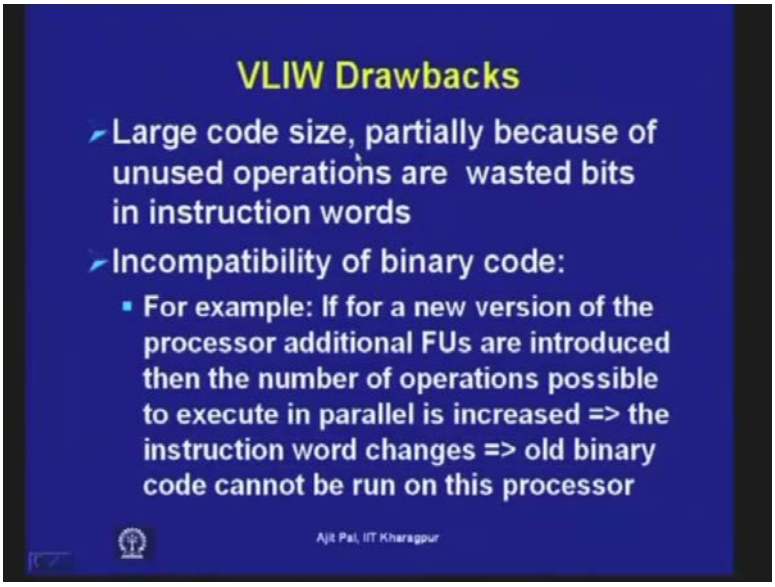


Some drawbacks. Number 1 is it has got large number of registers needed in order to keep the functional units active and these functional units are needed to store operands to store results. And whenever you fill or I mean say pick a 4 instructions to be executed in parallel obviously it will require large number of registers present in the processor. So, this is 1 drawback unless the processor has got large number of registers it is not feasible. Then large data transport capacity is needed between function units and the register file and between register files and memory. So, you can see your instruction is quite long that has to be fetched from memory then those that particular instructions will be loaded in the instruction register. And then instruction register will give I mean will put will apply the input to the execution unit which has to be fetch the operands from the register.

So, we have to fetch operands from large number of registers because we have fit them to several function units simultaneously. So, between function units and register file and between register files and memory so this is the larger transport capacity or bandwidth is required. And third is high bandwidth between instruction cache and fetch cache. So, you know you have your program stored in the as you know nowadays all processor used cache memory to store the program. So, instruction programs are stored in the instruction cache and data is the stored in the data cache.

Later on we shall discuss in detail about the cache memory and obviously you will require high bandwidth between the instruction cache and fetch cache. And why it is required because 1 instruction with 5 operations in a particular instruction and each of 32 bits are required 160 bits per instructions. So, these examples shows why you require high bandwidth between the instruction cache and cache and the fetch unit.

(Refer Slide Time: 15:46)



So, there are other drawbacks large code size partially because of unused operations are wasted bits on instruction words. So, you have seen although you have used a large instruction word, but a large number of fields are unused. So, primarily because of this you know you have got large code size partially because of unused operations and wasted bits in instruction words. So, although it is there, but a significant portion is wasted. Another very important aspect is incapability of binary code. You know whenever a next generation processor is introduce, it is expected that it will be backward compatible with the next processor.

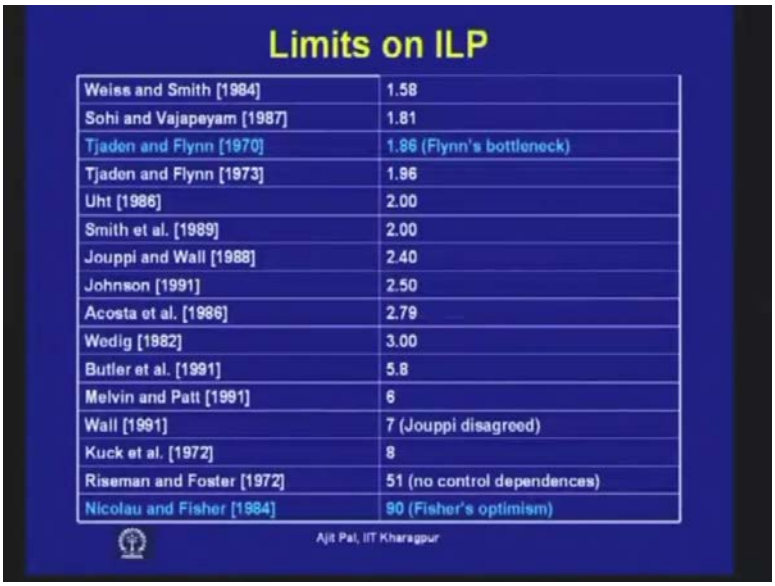
New generation of processor that means we have seen that you can execute 80386 programs in a 803486 machine or 8046 program can be executed in 20 m, but in this particular case there is a problem. Problem is arising because for example, if for a new version of the processor additional function units are introduced. Then the number of operations possible to execute in parallelism is increased. For example, earlier you are

having say 4 function units and obviously your processor that I L W that V I L W instructions are having 4 fields this ways your instruction.

Now a new generation this is a old machine and suppose you have introduced a new machine has been introduced where you have got 5 function units. So, the instruction will be will change so it will be like this and as a consequence the program for this machine cannot be executed in this new machine. So, backward compatibilities lost so this is refer to us incompatibility your binary code.

So, because of limitations V I L W processors although I have discussed in detail for the sake of completeness and commercial processors have been have been implemented commercially processor. If V I L W processor have been implemented for example, transmit as ISO processor because of you know low power and but you know they are not very commercially successful. So, commercial successful processors are superscalar.

(Refer Slide Time: 18:35)



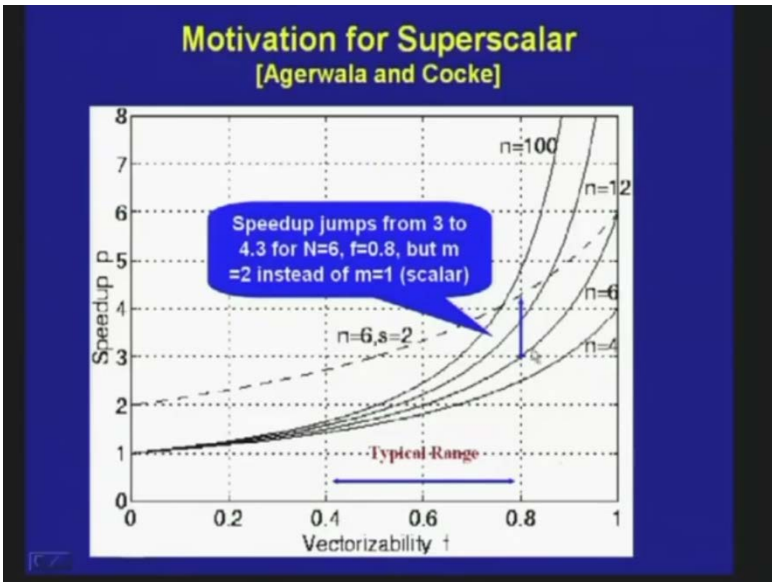
But before I discuss superscalar and various variations of superscalar, let me this touch upon one important aspect that is your limits on I L P. You know all our processors that we shall be discussing are we will be heavily lie on the instruction level parallelism that is available. Now, if the instruction level parallelism that is present is small. Then obviously there is no gain for gain in having by implementing a processor with large with having say large instruction field or processor, which can issue large instructions simultaneously. So, lots of simulation studies have been carried out by many researchers

to identify the limits on instruction level parallelism. And there are 2 extreme observations; number 1 which was by Flynn is that is known as Flynn is bottleneck.

And here it says that instruction level parallelism available in the basic block is always less than 2, it is 1.8, 1.87, and 1.96 so it is less than 2. And that means if you do not do specialized operation like lupa rolling, software pipelining we have discussed. Then if you simply restrict to the basic block as it is present in a program then the instruction level parallelism that is available is only 2. However, contradictory results were published by other researchers for example, fisher along with his colleagues published a paper in back in 1984 where it say that the instruction level parallelism available is 90. So, there is a big gap between 2 and 90 and what he did is actually he identified some programs where he can essentially numerical processing is involved and there he found that the instruction level parallelism available is 90 and in fact later on this was refer to us Fisher is optimism.

So, you can see there is a pessimistic view about the instruction level parallelism available that means it is 2 of 2. On the other hand there is a optimistic view which says that instruction level parallelism is much larger. Subsequent researchers have confirmed that this 90 is really 2 big, but defiantly more than 2 instruction level parallelism is possible. So, 3 5.8 6 7 so these are the different type different in instruction level parallelism which are available. And obviously people will be interested in exploiting the instruction level parallelism that is available in programs. And for that if necessary special techniques like software lupa rolling, software pipelining those techniques are incorporated to increase the instruction level parallelism, so with this background.

(Refer Slide Time: 22:11)

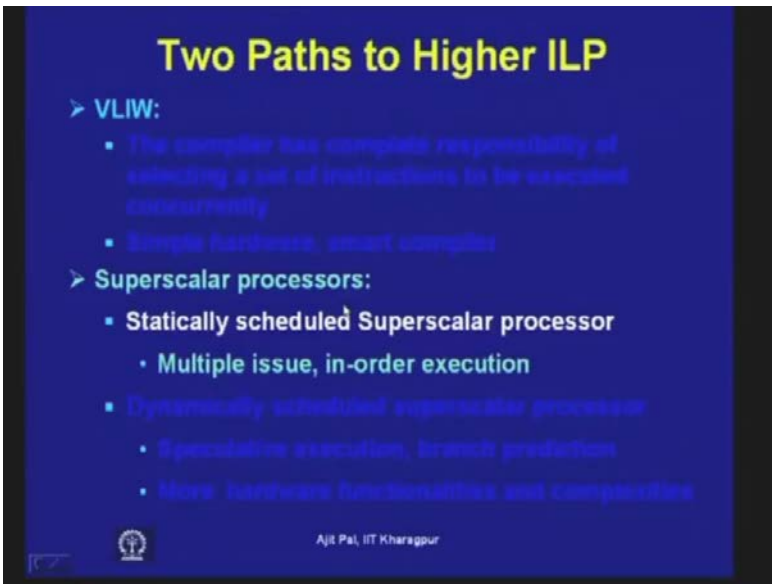


Let us discuss the motivation for Superscalar processor why superscalar processor is from he is proposed or required? You can see here the vectorizability of a program that means a particular program which can be vectorized. Vectorizability means parallelism so you can see vectorizability is varies from its may be 0.8 to 0.42 0.8 and this is a typical range. Now, suppose you have got 2 processors 1 processor is where n is equal to 1 I mean n is the number of stages in this particular case it has been n is the number of stages. Now n is x number of stages and you have got a simple pipeline processors obviously speedup of will vary depending on the vectorizability or parallelism available in the program. So, you can see depending on the parallelism available the speedup of will varies if n is equal to 4. If you have got 4 stages maximum value is 4 maximum value is 4 and this will drops certainly as the vectorizability parameter decreases.

Similarly for say for n is equal to 6 that is the n is the number of stages the maximum possible speedup is 6 as we know. And this also drops rapidly as the as the vector the vectorizability or that it may be called as instruction level parallelism that parallelism that is present that decreases. So, but if we go for say superscalar processor by superscalar processor mean the processor can issue more than 1 instruction simultaneously. So, this is a curve correspond this is the corresponding curve so in this curve as you can see speedup even is minimum is speedup is 2 that is possible. And maximum speedup can be 12e with the number of stages in pipelining equals to 6.

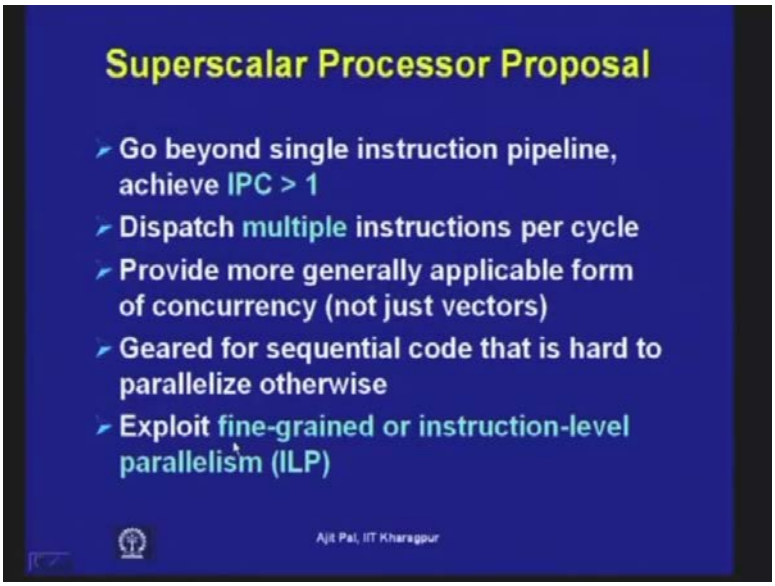
So, for the same number of stages if we increase the number of issue from 1 to 2 as you can see here the this is a this point corresponding to number I L P or vectorizability equal to 0.8 we get 3 for convectional pipeline without any parallelism I mean more than 1 instruction issue. Now you can see here as the as you increase the if you increase the number to issue 2 the speedup jumps from 3 to 4.3 for n equal to 6 n is the number stages and f is equal to 0.8 that is parallelism. But m is equal to 2 m is equal to 2 that means number of instruction that is been issue is equal to 2 instead of m is equal to 1, m equal to 1 correspond to that scalar pipeline processor. So, this simple observation tells you that it is essential to go for superscalar processor to achieve higher speed up and higher speed up means higher performance.

(Refer Slide Time: 26:00)



So, superscalar processors you can have statically scheduled superscalar processor that is 1 version and later on we shall discuss about the dynamically scheduled instruction processor. So, in case of statically scheduled superscalar processor you can have multiple issues in order execution. And later on we shall discuss about dynamically scheduled processor where you will see out order execution is possible.

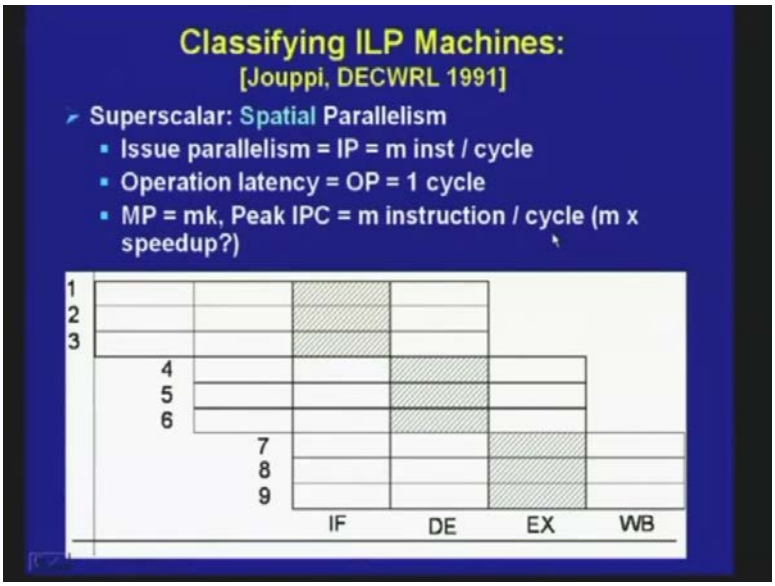
(Refer Slide Time: 26:28)



So, this is the superscalar processor proposal where we will try to go beyond simple instruction pipeline to achieve instruction per cycle greater than 1. Dispatch multiple instructions per cycle, that means you will issue the processor whenever it is executed I mean executing a program it will issue more than 1 instructions and it may be 2 it may 3 it may be 4. So, it will dispatch multiple instructions per cycle and provide more generally applicable form of concurrency. You see vector processing, vectorization or vector processing is a kind of parallelism where you know whenever you are executing a loop different iterations are independent.

So, since different iterations are independent in a vector processor all different iterations can be executed simultaneously. That is the one kind of parallelism, but that kind of parallelism is present only when the program involves vector processing. But conventional programs may not be made on always content vectorizability or processing of vectors. Since such a case how to provide you know concurrency that is what is been done in superscalar processor. So, it is geared for sequential code that is hard to parallelize otherwise. And obviously it will exploit fine grained on instruction level parallelism that is our Superscalar processor and how is done is shown with the help this diagram.

(Refer Slide Time: 28:10)



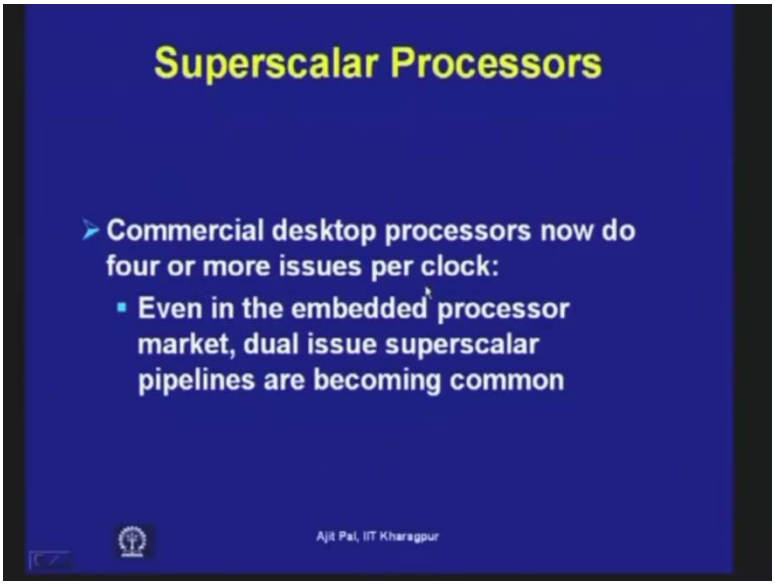
Here you have got we are utilizing special parallelism you know in case of conventional pipelining we have seen we try to use temporal parallelism. So, instructions were issued one of the other in time at the interval of one clocks cycle. But here what is been done several instructions simultaneously issued so issue parallelism is m instructions per cycle. So, this is the parallelism that is present in a superscalar processor. So, 3 instructions are issued here in the next cycle another 3 instructions are issued in the next cycle, another 3 instructions are issued and operation latency. Of course, it remains 1 cycle as you can see the first result is coming out after k cycles, k is the depth of the pipeline and then after 1 cycle you get another result so operation latency is 1 cycle in this case also.

And machine parallelism is equal to m into k m into k as I have already told you can have the total number of instruction in flight simultaneously is m into k. And the peak I P C peak number of instruction per cycle that you can achieve is m instruction per cycle. So, that speedup you know this is a ideal case m instruction per cycle, but obviously the speedup factor will be less than 1 because there will be some stoles. So, here when there is no stole when there is no dependency means instruction can be issued in parallel depending on the parallelism available in the processor.

Then we get an ideal speedup that is equal to m how m into k. However, we cannot real do that in real life practical programs where there will be dependencies and as a

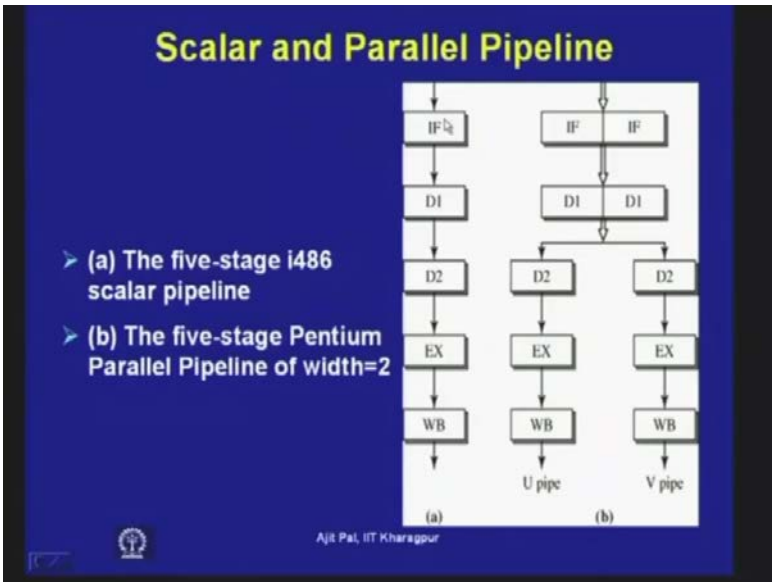
consequence there will be stoles. So, this particular diagram does not show any stole so their speedup factor will be less than 1 so m into point something. So, that will be the value I will get so peak I P C will be in m pack will be less than 1.

(Refer Slide Time: 30:45)



Now, based on this Superscalar processors have been introduced commercially and so commercial desktop processors now do 4 or more issues per clock. And even in the embedded processor market dual issue superscalar pipelines are becoming common. So, we have already seen that minimum instruction level parallelism that is present is 2 even without doing you know lupa rolling and other sophisticated things in the basic global program that instruction level parallelism available is 2. So, super superscalar processor with issue rate of 2 is quite common and then of course, you can go for go beyond to 4 or more.

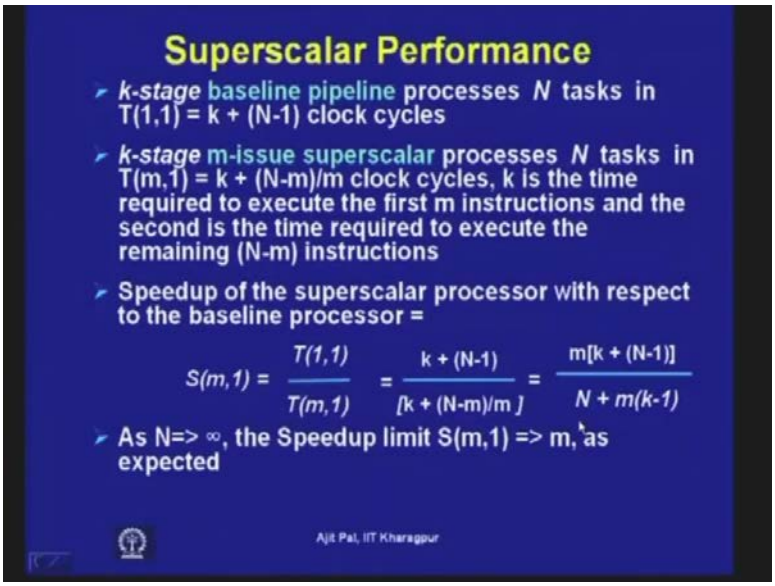
(Refer Slide Time: 31:39)



Here is some example so here 2 processors are shown first one is this correspond to 5 stage i 4 8 6 that is your Intel 4 8 6 processor where no parallelism. I mean Superscalar technique was introduced while it is a simple scalar pipeline with 5 stages instruction fetch. Decode stage 1, decode stage 2 decoding was complex the reason for that was you know these processors has used complex instructions. Since they used complex instructions decoding is little complicated that is why decoder stages was divided into 2 stages decoder stage 1.

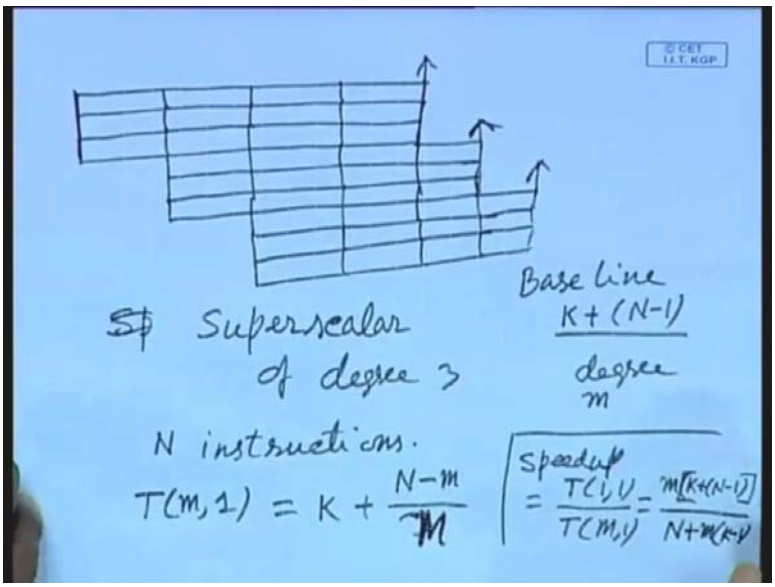
And the order stage 2 then their execution stage and write back or operator row that was the 4 8 6 processor pipeline so it is say scalar pipeline. Now, when Pentium was introduced, Pentium was having a parallel pipeline with a 2. So, it is say superscalar processor with of degree 2 so as you can see here it will fetch 2 instructions decode 2 instructions then issue 2 instructions. Because it has got 2 pipe U pipe and V pipe 2 separate pipelines though which 2 instructions can issued and will get executed in parallel. So, the d 2 you know 2 separate stages execution and write back. So, this the superscalar processor first introduced in Pentium.

(Refer Slide Time: 33:19)



Now, let us focus on the performance that you can achieve whenever you go for superscalar execution. We have seen that k stages baseline pipeline processors will require N tasks and it will require k plus N minus 1 clock cycles. Now, let us see what is the time required to execute the same program ideally obviously we shall go for the ideal one k stage issue m issue superscalar processors. So, here you have got say.

(Refer Slide Time: 33:58)



Let us assume there are 4 stages in pipeline and 3 instructions are issue simultaneously. So, 3 instructions are issue simultaneously then in the next cycle another 3 instructions

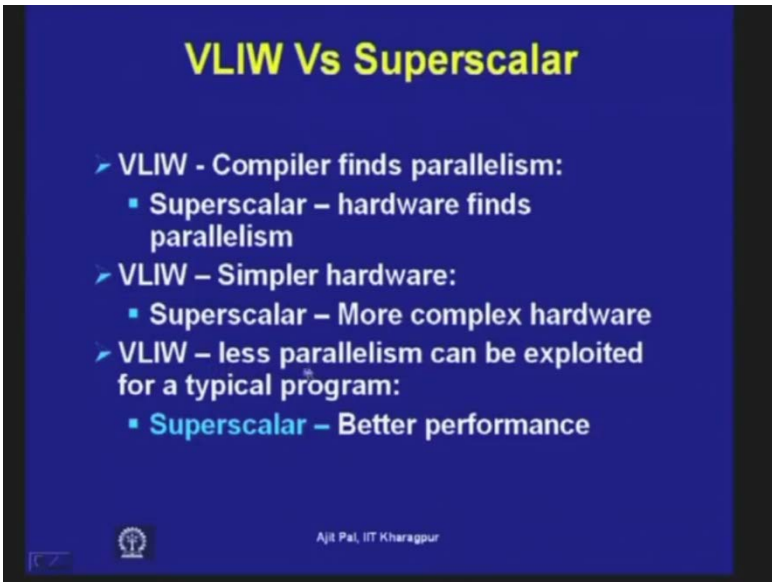
are issued. In a next cycle another 3 instructions are issue in this way continues so this is the superscalar processor of degree 3. So, in this case what is the time required to execute n instruction. So, in for the sake of generating we shall consider it that is has a degree of m, but in this example it is shown degree shown is 3. Now as you can see here the after k clock cycles k is the number of stages it will result for 3 instructions should be available. Then in the subsequent clock cycles as you can see each time you will get result from 3 instructions.

So, here first 3 instructions will require take clock cycles then you are left with N minus m instruction. So, N minus m is instruction will be result should be produced in N minus m by 3 instructions are 3 instructions result of 3 instructions are produce past cycle. So, this by 3 so this is the sorry let me put m in general. So, this is the time required k plus N minus m by m. So, this is the number of you can say T m one for the Superscalar processor. Now, what is the speed up what is the speed up of this processor with respect to our baseline processor? For our baseline processor we have seen the baseline that is your baseline pipeline processor the time required was k plus N minus 1 that was the time required. And on this case in this for the superscalar processor of degree gain time required is k plus N minus m by m.

So, speedup is equal to T 1, 1 by T m, 1 so that means you have to divide this by this and you will get is equal to m into k plus N minus 1 by this factor N minus N plus m into k minus 1. So, this speedup they will get in case of your Superscalar processor. So, this m 1 this is shown here. Now, whenever you are executing large number of instructions then N is infinity for N is equal to infinity. This is speedup limit s m , 1 that becomes is equal to m we have seen in case of your normal processor mean pipeline processor we are considering pipeline processor the speedup was k with respect to non pipeline.

Now, in this particular case speedup is m with respect to the baseline pipeline processor. So, if we consider this speedup with respect to the original non pipeline processor the speedup will be equal to maximum speedup with respect to the non pipeline processor will be m into k or machine parallelism is m into m into k, as expected with respect to the baseline pipeline processor the speedup is m.

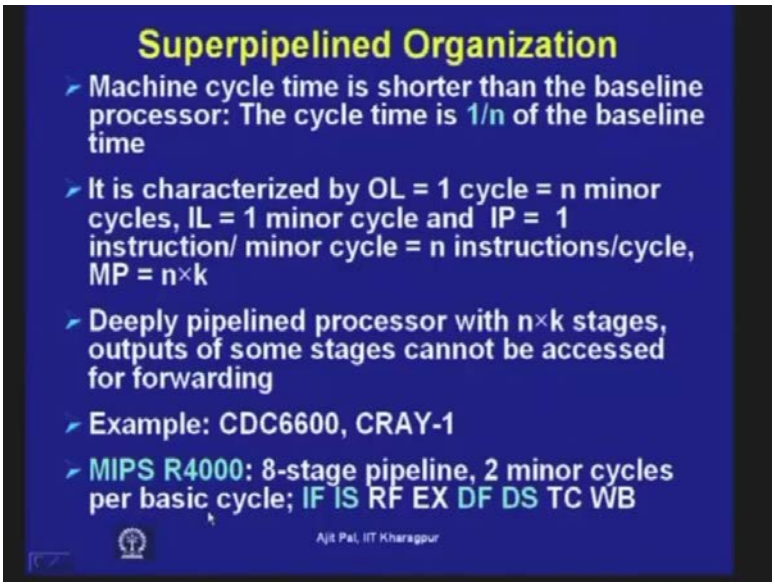
(Refer Slide Time: 39:14)



Now, here is some comparison between the V L I W and Superscalar processor you have seen that V L I W in case of V L I W compiler finds the parallelism in superscalar hardware finds the parallelism. So, V L I W simple hardware Superscalar will require more complex hardware and but one important point that you should notice here is that V L I W has less parallelism. Because it can exploit less parallelism because you know that parallelism is exploits extracted with the help of the program or compiler.

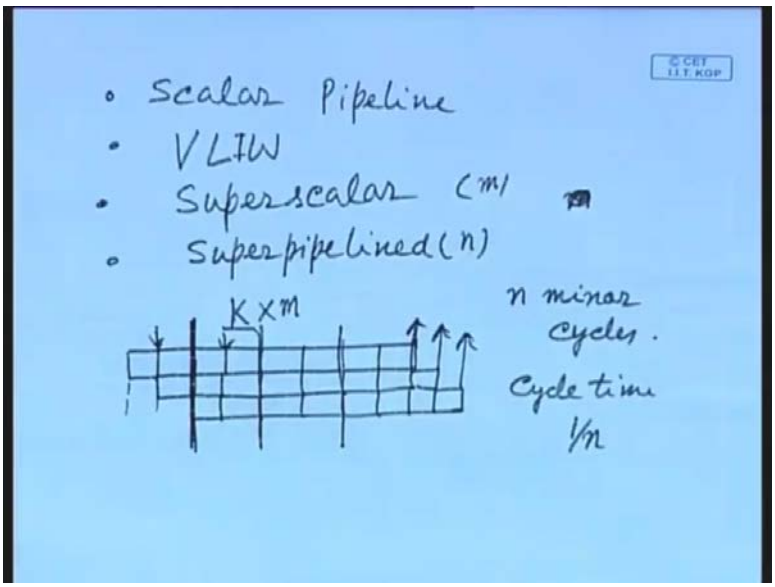
And compiler cannot identify all the parallelism that is present in the program at compile time. On the other hand since it is done by hardware by the Superscalar processor at run time it can identify more parallelism instruction level parallelism. As a result superscalar gives you better performance so ideally you know we have seen for the both cases v is m, but in case of V L I W lesser number of fields of the V L I W instruction will be fill up. On the other hand in case of Super superscalar processor more number of instruction will be execute in parallel because the parallelism is extracted with the help of hardware. So, superscalar will give you better performance.

(Refer Slide Time: 40:57)



Now, let us look at the Superscalar organization machine cycle time is shorter than the baseline processor. The cycle time is another variation that is super pipelined we have discussed about simple pipeline or scalar pipeline.

(Refer Slide Time: 41:19)



We have discussed about V L I W, we have discussed about superscalar. Now we are considering another very I mean another extension that is known as super pipelined what is the basic idea behind the Super pipelined processor? It has been observed that we have seen that the different stages of a pipeline processor do not take unknown uniform.

Because you know we are trying to combine different types of instructions which are not a same. So, as a consequence there was a possibility to for the divide a particular stage of pipeline. For example, suppose originally a processor was having see 4 stages now each stage this is the basic number of stages. Now this is for the divided each stages for the divided into 2 sub stages. So, you can say you have a major cycle as define by the pipeline now you have got minor cycle. So, in each cycle some minor cycles are introduced what is the advantage that is that you get?

Advantage is now as if the number of stages increases so you have introduced n minor cycles. And as you introduced n minor cycles the clock frequency is the cycle time is now becomes one by n of the baseline processor. So, in this particular case I have shown n is equal to 2 so the clock frequency doubles or the cycle time is one by n cycle time is reduce. So, this is one by 2 of this now what benefit do we get out of it? Benefit is now another instruction is issued after a minor cycle. So, instead of waiting for the major cycle instructions are issued after a minor cycle.

So, as a consequence although the initial delay is same as the pipeline processor that means it will again take K clocks K clock cycles or K into n you know clock cycles in terms of this Super pipelined processor. But subsequently you will get result at the interval of 1 by n of the cycle time of the basic pipeline processor. So, throughput will increase and it has been found that your m I mean whenever you have got Superscalar processor degree m. And super pipelined of degree n when m and n are equal you will get morally same performance, but what is the difference between the 2? Difference between the 2 is in this particular case you can see the number of I mean clock frequency is increasing. Cycle time is getting reduced and the processor has to run at a faster clock with a faster clock. So, here it has to run at a faster clock and however the number of you knows that number of the number of instructions that will be exe if you consider throughput it will although it is same, but the clock frequency is increasing.

So, it is characterized by output latency of 1 cycle that is n minus cycles and I L is equal to 1 minus cycle and I P is equal to 1 instruction per minus cycle or you can say n instruction per cycle. So, here machine parallelism is again is equal to n into k and in case of superscalar processor we have seen it is equal to m into k. When m and m and n minor same the parallelism that available for both the machine is identical. So, it is not different but you are achieving the same performance using a different approaches or all

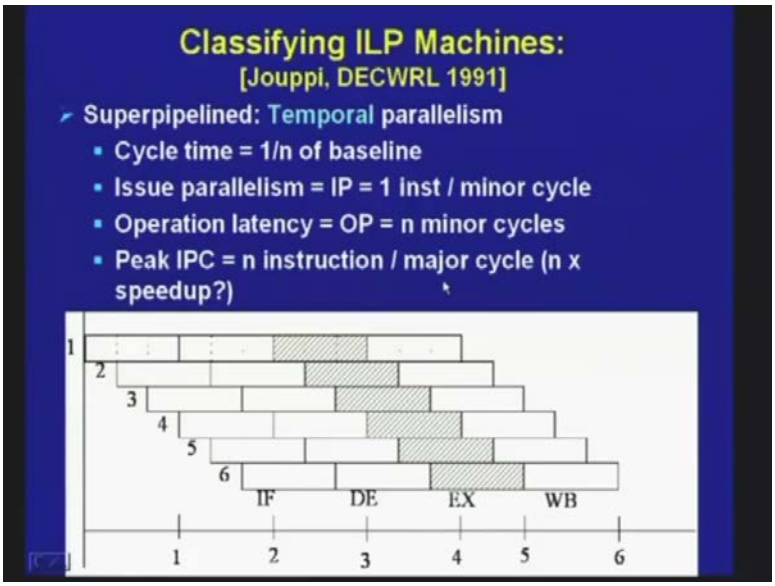
together different approaches. And it may be consider as a deeply pipeline processor with n into k stages.

So, if you want to sense simple terms you may say that again it is nothing but the pipeline processor only thing that the number of stages has been increased. Earlier it was having k stage, now it is now it is having k into m stages. But although this statement is correct, but in fact is there is some difference, what is the difference? Difference is we have seen whenever we go for forwarding into whenever we go for forwarding results from the pipe the buffers are applied to the functional units. That means you are from this form output of this output from here is fed is taken output form here is taken and output form I mean form this intermediates stages are taken.

But not you cannot take form the minor cycles that means the output from at in the output available that is available from the minor cycles are not accessible are not available. So, that means whenever you go for implementing forwarding which is necessary to overcome data hazards. As you have seen by hardware means there you will feel the difference there you cannot consider it as a pipeline with m into k stages. There you will consider it as if it is a pipeline with k stages so there lays the difference. So, output of some stages cannot access for forwarding, which is available for minor cycle. And some super pipelined processors have been designed one is C D C 6600 that was designed CRAY 1 is also super pipelined and M I P S R4000 is also a super pipelined processor which has got 8 stage pipeline with 2 minor cycles. For example, 2 minor cycles per basic cycles it has got instruction fetch that is the first stage. And instruction fetch second I s stands for instruction fetch second.

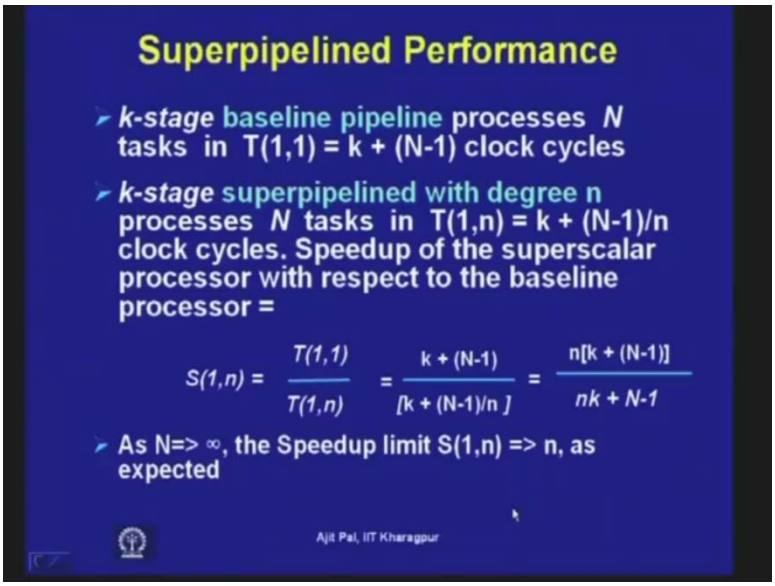
So you can see instruction fetch is now divided into 2 stages. Then the second stage that you know the instruction decode and register read our execution stage again divided into 2 stage register fetch and execution. Then that then data fetch and data is data second that is you are this stage is also divided into 2 stages. Similarly, the write back stage that is also divided the t the stands for tag check you know you are reading form the cache memory, so name has been tag check. So, you can see that it has it has got 4 stages you can consider it has 8 stages pipeline. However here it has got 2 minor cycles n its stages so this an example of the super pipelined organization.

(Refer Slide Time: 50:14)



So, according to the classification of an Jouppi this is the super pipelined processor where the time is 1 by n of the baseline processor issue parallelism is one instruction per minor cycles. So, you can see this you are issuing 1 instruction per minor cycle output latency is m minor cycles. So you can see after m minor you are introducing output generated after m minor cycles. And peak I P C is n instruction per major cycle so n into speedup. So, here you can achieve as I have said machine parallelism you can achieve is n into k.

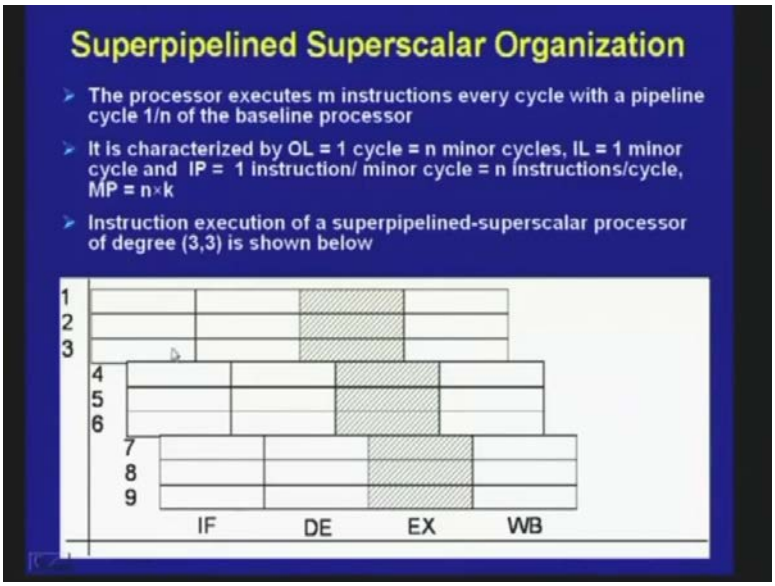
(Refer Slide Time: 50:55)



So, in a similar way you can find out the performance of the Super pipelined processor and time required is you can say T 1 N is equal to k plus n minus 1 per m. So, you can consider it as a pipeline processor and in that way you can find out. And speedup is equal to T 1 1 that is a baseline pipeline processor by T 1 that is super pipelined processor. We get k plus N minus 1 by k plus N minus 1 by N and so this is the speedup that we get N into k plus N minus 1 by N k plus N minus 1.

And whenever N is very large then you get S 1 N is equal to N as expected that means with respect to the baseline pipeline processor we get a speedup of N. However, whenever we considered this speedup pipeline with respect to non pipeline processor we shall get a speedup of N into k. So, here always we are considering with respect to the pipeline processors that is why speedup limit is N, but that is again ideal situation now we can extend the idea.

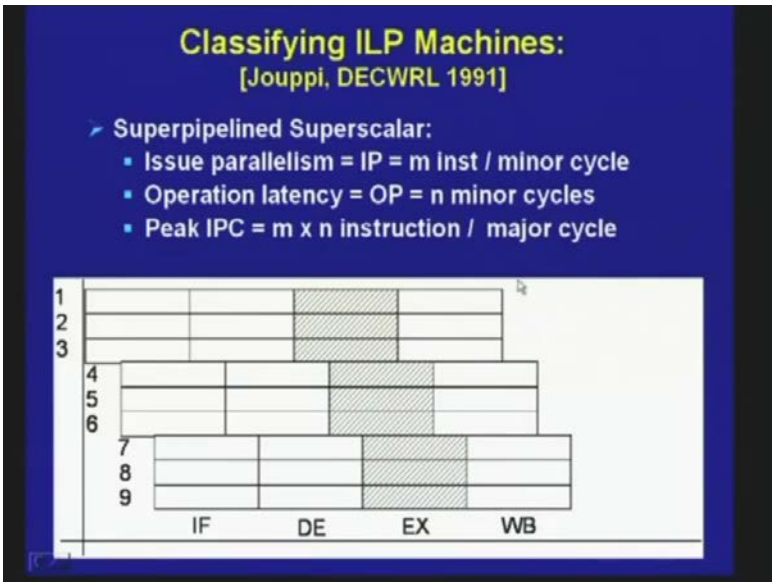
(Refer Slide Time: 52:13)



Further and we can have Super pipelined, Super superscalar organization that means we can combine superscalar along with super pipelined and processors commercial processors are available for that. And here the processor executes m instructions every cycle with a pipeline cycle 1 by n of the baseline processor and it is characterized by O L is equal to 1 cycles is equal to n minor cycle or I L is equal to 1 minor cycle. And I P is equal to 1 instruction per minor cycle or n instruction per cycle and here machine parallelism is equal to n into k no it will be n into m into k I believe.

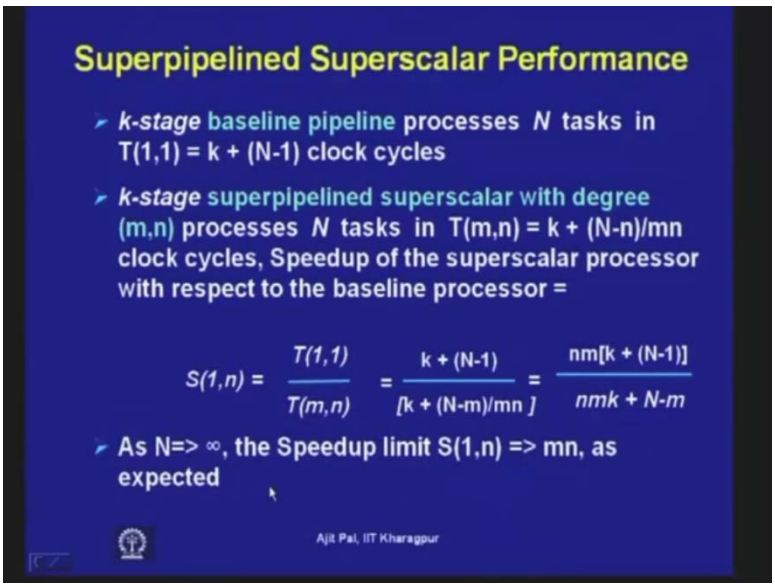
So, here it is wrong machine parallelism will be n into m into k because it is you can see here the number of instruction which are number means instruction in flight during execution will be equal to n into m into k. So, machine parallelism will be m into n into k so this should be modified. So, the execution of instruction for a super pipelined superscalar processor of degree 3, 3 is shown here. So, 3 instructions are issued and after each minor cycle another 3 instructions issue. So, after 3 minor cycles another 3 instructions issued so this is how instructions issue is taking place and execution is taking place this is ideal situation.

(Refer Slide Time: 53:51)



So, according to the classification by Iouppi again I P is equal to m instruction per minor cycle. So, o p is equal to n minor cycles and I P C is equal to m into n instruction per major cycle.

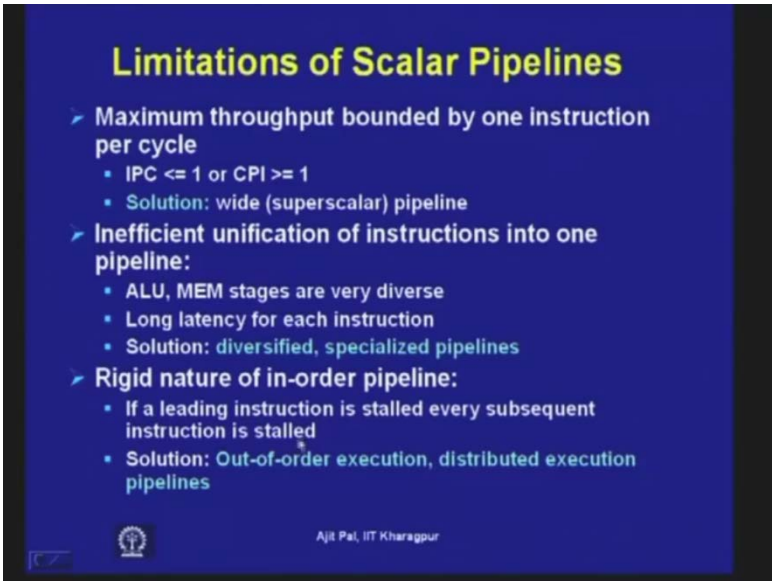
(Refer Slide Time: 54:08)



So, this the performance of super pipelined, superscalar with degree n here T m n time required execute n instruction will equal to k plus N minus m by m n. So, you see you are dividing a factor of the first n instruction first k instruction will I mean first m instruction will take k cycles and after that remaining n minus m will be require N by m

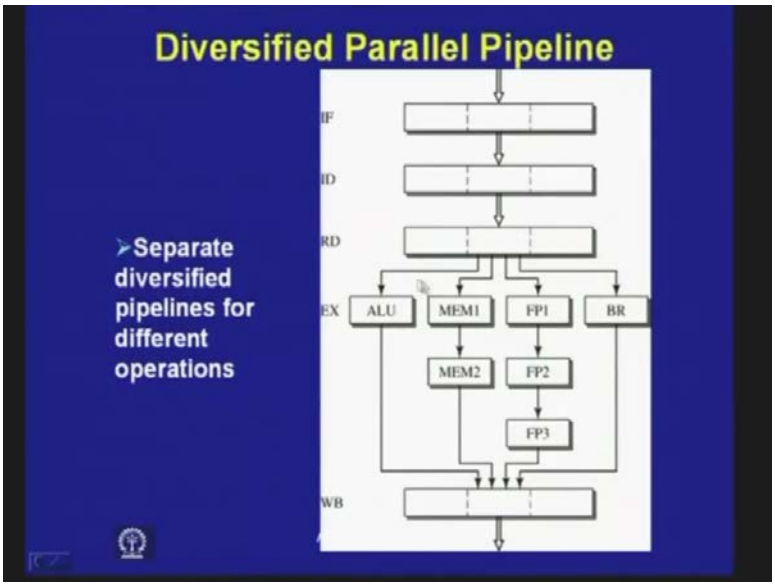
by m n cycles. So, based on that you get a speedup which is equal to which is equal to which will be this will be S 1 n is equal to n m into k plus N minus 1 by n m k plus N minus m. So, speedup limit in this case is m n as expected.

(Refer Slide Time: 55:09)



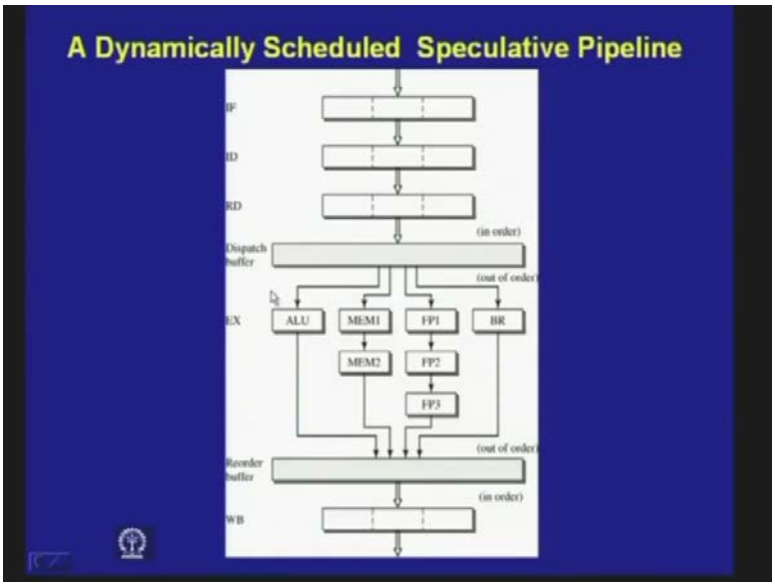
Now, we can we have seen that inefficient unification of instructions leaked to some problem because we try to execute to combine different types of pipelines which are not identical so A L U operation, memory operation. So, instead of that we can do for diversified specialized pipeline as it is shown here.

(Refer Slide Time: 55:36)



A L U through 1 pipeline memory operation through another pipeline flooding point through another pipeline. Instead of trying to combine different types of instructions in a single pipeline we can have separate diversified pipeline for different stages. And along with that we can combine super pipelined, superscalar execution and another limitation that we have seen that is because of rigid nature of in order pipeline. So, this problem can be overcome by having out of order execution and distributed execution pipeline as you can see here.

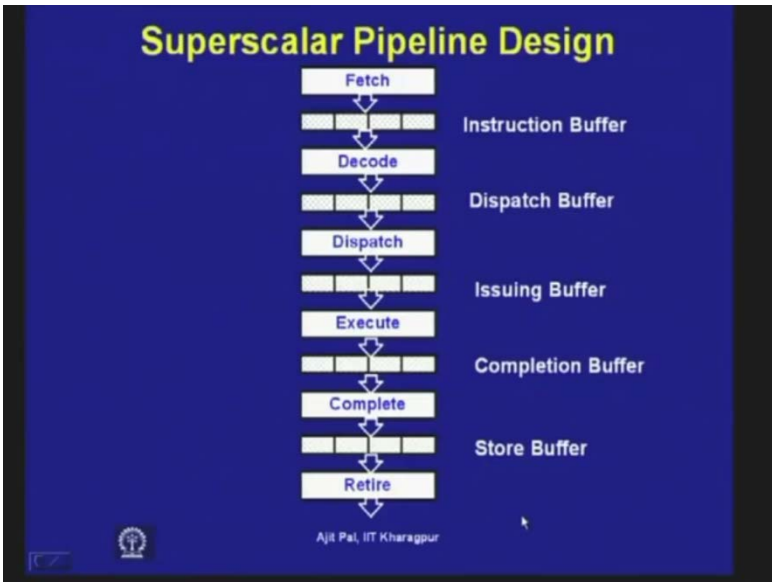
(Refer Slide Time: 56:13)



However this will require dispatch buffer that means you will require some buffer where the instructions will be temporarily stored. So, multiple entry instructions buffer will be require for example, in this case this dispatch buffer through which you will be storing several instructions multiple instructions. And then you with the order in which the instruction cell will be that will be issue from the dispatch buffer may not be same as the program order. That means that this from the dispatch buffer instruction will be issued out of order.

So, whenever the instructions are issued out of order there is a possibility that the instructions they instruction will be executed also out of order. So, out of order execution will take place and results should be stored in another buffer known as reorder buffer. So, the reorder buffers the order in which the results are store in the reorder buffer you know. Ultimately you have to convert them in order the way it is available in your in program order. So, form this reorder buffer the outputs are generated as it is as it comes out from the program order so then the writing in a processor is required.

(Refer Slide Time: 57:36)



So, the processor design superscalar processor design will involve different functional units fetch, decode, dispatch, execute, complete, and retire. Along with that you will require different types of buffers instruction buffer, dispatch buffer, issuing buffer, completion buffer, and store buffer. And these designs will be we shall in my subsequent lectures I shall discuss about the design of superscalar processor involving this type of buffers and functional unit is diversified functional units in my next 2 lectures.

Thank you.