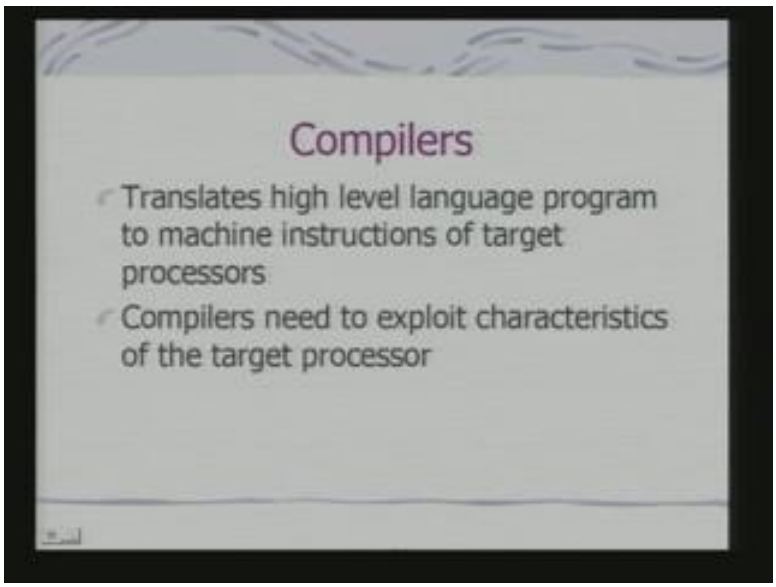
Embedded Systems
Dr. Santanu Chaudhury
Department of Electrical Engineering
Indian Institution of Technology, Delhi

Lecture - 34
Compilers for Embedded Systems

Today, we shall look at the compilers, which are used for generations of the machine code targeted for embedded systems and the processors which have been used in embedded systems. In fact, the compiler technology in a way is part of the design tools that we used for designing embedded systems. Because, if we have to exploit the characteristics of the processor you have to generate optimize code.

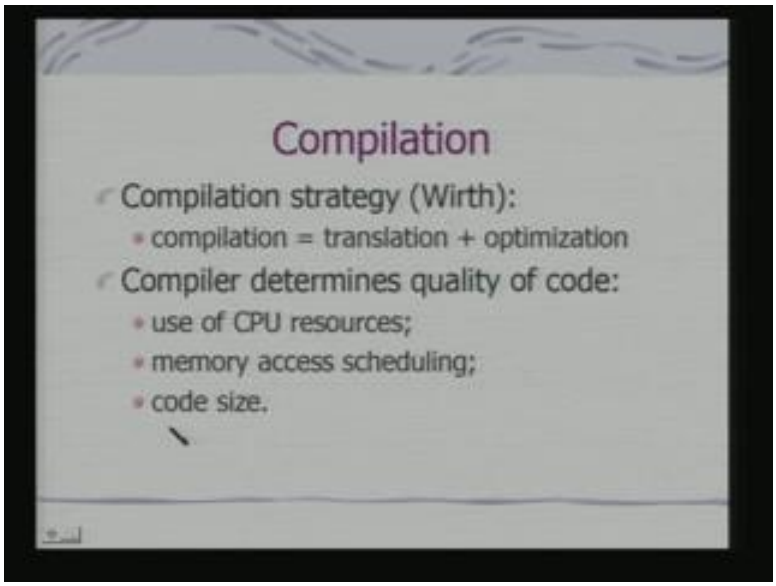
And, compilers provide you the tool to generate optimized code, exploiting features of the target architecture. We shall look at some of the issues, that compilers are concerned with in the context of embedded systems and processors, which are used for embedded systems. So, what are really compilers just recapitulate. Compilers translates high level language program to machine instructions of target processors.

(Refer Slide Time: 01:59)



And compilers; obviously, would need to exploit characteristics of the target processor, if it has to generate optimized code.

(Refer Slide Time: 02: 21)



Compilation, in fact in a way can be considered as combination of two phases. One is your translation, the other part is optimization. In fact, we are actually more concerned with optimization. Because, the techniques which go into the translation phase in a way or generic and not processor specific there may be language specific. But, they are not really processors specific. But, when we look at optimization and the optimization part also includes code generation and using it includes optimized code generation.

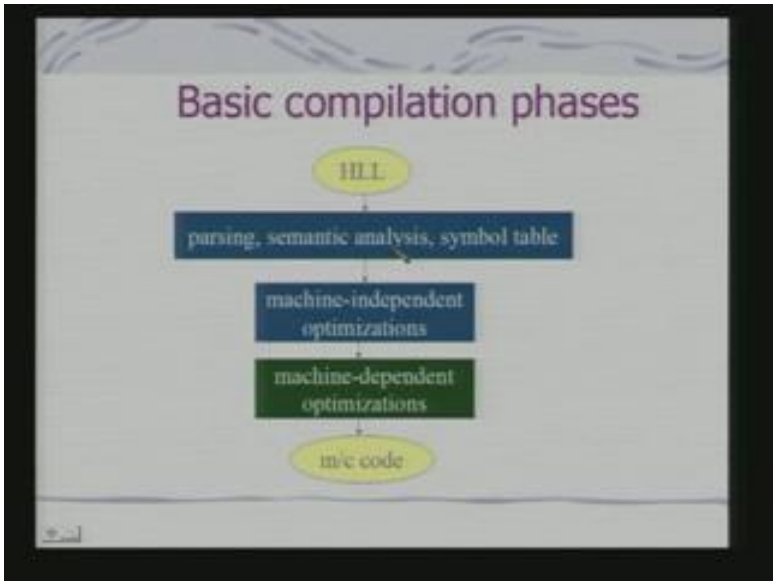
Then, this part is strictly processor specific. In fact, compiler determines the quality of code optimal usage of CPU resources memory access scheduling as well as that of code size. You can understand that all these parameters are determinant for various functional features of the system. Because, if I can use CPU resources optimally I am not accessing memory. Because, one of the most important CPU resources or registers if i am using registers optimally, I can minimize on memory access.

The moment I minimize on memory access my execution time improves at the same time energy consumption decreases. If my CPU consists of multiple functional units, if I can use this multiple functional units optimally then; obviously, the time taken would reduce. See if I consider a very large instruction set computer which has got multiple functional units. Then, the compiler would figure out the appropriate sequence of instructions, so that these functional units can be optimally used.

It should not be that one of the functional units is being used and others are not being exploited. The similar issues related to the memory access scheduling. So, if I can access memory in an optimal fashion both my time and energy is optimized. This is related to the code size. If you remember the example we have considered that if we are using arm processor and for a particular computation if the 16 bit instruction is good enough.

Then, the compiler can detect that and generate may be automatically 16 bit instructions and you have seen that if it is generating 16 bit instructions what has it been the memory required will be less. And; obviously, I can have the code put into a smaller size memory in the embedded system. So therefore, compiler becomes a key resource for designing efficient software for embedded systems.

(Refer Slide Time: 05:41)



Basic compilation phases, this is again brief recapitulation our input is the high level language program. Then, we have these phases, the first phase we have termed as parsing, which will also include lexical analysis that should be followed by the semantic analysis, which is language specific. And in the process, we shall also generate the symbol table that is definition of variables constants functions as well as their type information. Because, that type information is also used during semantic analysis.

Then, we can do machine independent optimizations which are not specific to the target architecture. In fact, in a way this part of the compiler can be built independent of the target architecture. This part is machine dependent optimization; that means, it will

exploit the target architecture it features. Primarily, what does it mean; it will exploit the instruction sets the nature of the instruction sets.

And accordingly, it will optimize the generated code. Optimizing generated code would actually mean selection of appropriate instructions. So, that it may be that your time required for a computation would be less. Because, same computation can be done through combination of different instructions and this combinations may required different number of cycles to execute. So, when I am optimizing I may like to choose a set of instructions which minimizes the number of cycles. And this would lead to time optimization.

The other kind of optimizations could be in terms of memory accesses and register usages depending on the number of registers available. As well as, when you have multiple functional units, the scheduling of instructions also becomes important. It is not necessary that you need to execute instructions in the order in which they are generated from the high level code.

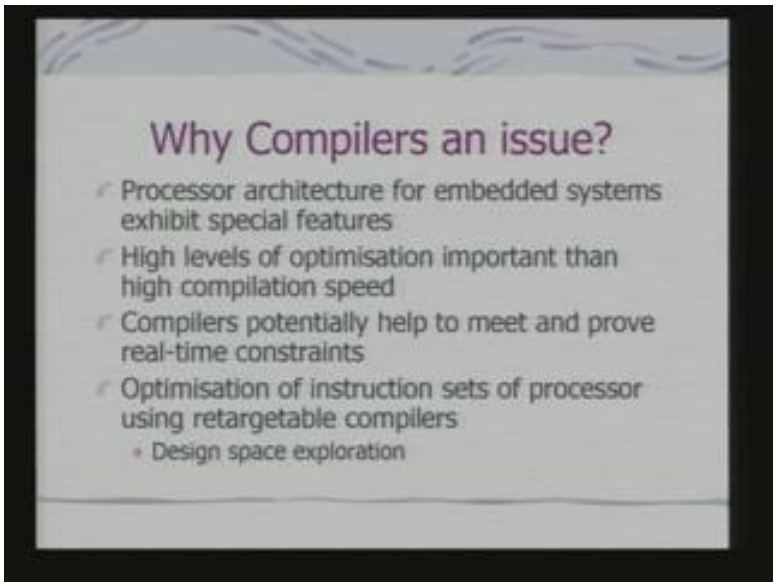
If there are no dependencies, in fact we have already seen such dependencies and how that such dependencies can be model using data flow graph if such dependencies can be detected and if we can re adding instructions. So, that these dependencies are removed then we can have also a better utilization of CPU resources like registers as well as functional units. So, these are the different kinds of machine dependent optimizations and you produce the machine code.

Now, the question could be that if I am writing program in assembly language. Knowing the instruction set of the processors, you can possibly you yourself generate an optimized code. But; obviously, you can realize then what you are sacrificing is the portability of the code. If the architecture is changed from one version to another version then again you need to recode the program using the features of the modified architecture.

But, if you are doing it in a high level language then your entire set of applications can remain unchanged. If you can really design and implement an efficient compiler say for example. If I am switching form arm seven to arm nine architecturally arm 7 and arm 9 are different. If I already have an application coded in C and if I can use an appropriate compiler to translate your application in an optimize fashion for target arm 9 processor.

Then, the course that goes into recoding of the application disappears. Because, compilers in a way can be used for many other applications as well the basic principle of bringing down the cost of the design by reusing tools and modules is equally valid in this context as well.

(Refer Slide Time: 10:16)



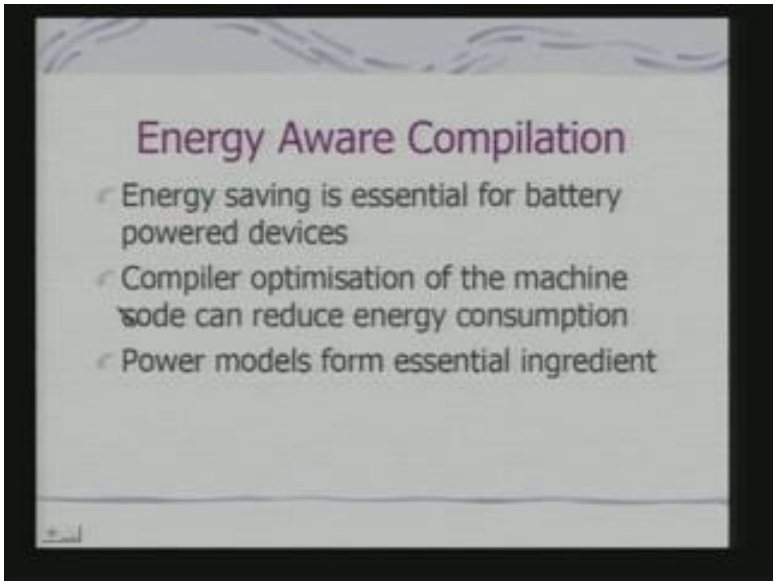
So, why compilers are an issue, in fact this point we have already touched upon. We are again the processor architecture for embedded systems exhibit special features. And, a high level of optimization is more important than high compilation speed, because your code is expected to be compiled once and executed 100 of times. So, you are for more complex compiler and more time in compilation.

Compilers potentially help to meet and prove real time constraints. Because, if we can generate optimize code then the possibility of meeting real time constraints becomes higher. And in fact, you can actually prove from the number of cycles required for a computation and depending on the clock speed of the processor whether this computation would meet certain timing constraints or not.

And, optimization of instruction sets of processor using retargetable compilers. This is another way of design space exploration, when we are dealing with parameterized architectures. In fact, this is one point we shall discussed. If you remember in the last class, we had talked about soft core based platforms. So; that means, your architecture is parameterizable. If your architecture is parameterizable you need to now search for an

appropriate instruction set as well to exploit the modified architecture. There also compiler becomes a useful tool.

(Refer Slide Time: 11:56)



First, we shall look at one important issue for optimization which is typical to embedded systems that is energy aware compilation. Because, energy is always resource particularly for battery operated embedded systems and you have seen then you are getting a number of appliances which are battery operated and at the same time providing you with sophisticated functionalities. See if I have to realize this sophisticated functionalities with minimum consumption of battery power then compiler needs to play a critical role.

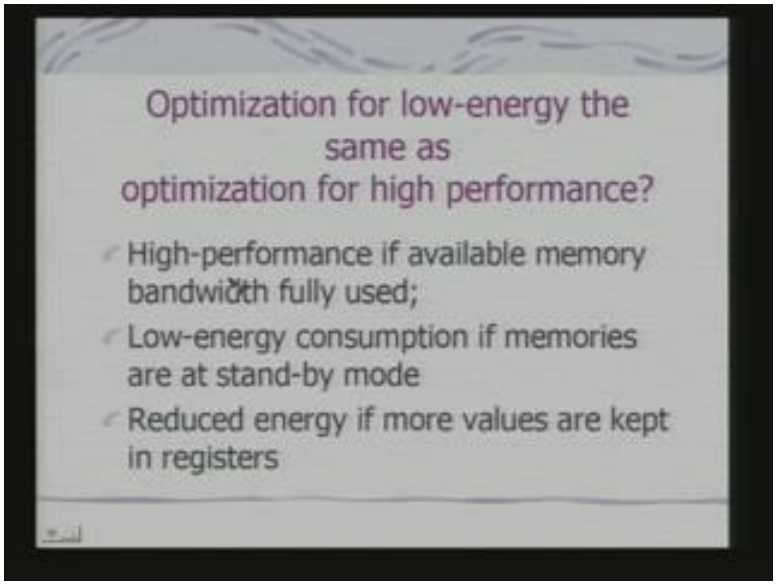
So, compiler can do an optimization of the machine code that reduces energy consumption. And, this optimization is not that obvious. In fact, I had already talked about I can choose a set of instructions for doing a computation which consumes least number of cycles that gives me efficiency in terms of time. But, it may not give you efficiency in terms of energy.

So, the first kind of optimization in a way is universally valid. I would like my code to run faster even on a general purpose computer. But, in the context of embedded systems is not that I would like my code to run faster consuming less power, so the optimization problem next to look at power as well. So energy our compilation is an important feature

for embedded appliances. And in fact, these interest and concern has grown of late in recent years, because of availability of the number of battery power devices.

Now, for such compilation the power models form essential ingredient, what is the power model? Power model actually represents a kind of model which encodes the energy consumption pattern of instructions of the processor.

(Refer Slide Time: 14: 20)

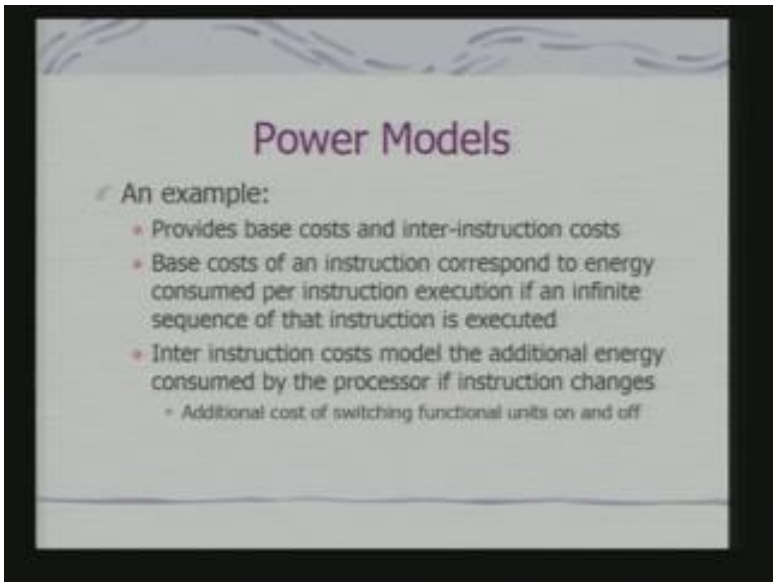


So, let us take an issue example later on we shall look at them. So here, we are looking at the tradeoff between optimization for low energy and optimization for high performance. High performance is available if memory bandwidth is fully used. And, low energy consumption, if memories are primarily outstanding by mode. And reduced energy, if more values are kept in registers you will find that if I look at this possibility.

Then, if we are keeping the variables in registers, what happens, the instructions involving registers may consume less energy. If it is consuming less energy, then it is becoming power efficient. And, in a way is will consume less energy because there are no activities on the bus. So, there will be power efficiency at the same time you are avoiding the memory access machine cycles. So, that would speed up the execution as well. So, if I can come up with these kinds of strategies that become the best for embedded systems.

Now, whether you can come up to these kinds of strategies would depend on whether you have got an appropriate power model to characterized execution sequence for a processor. So, power model actually provides you with the pattern in which energy will be consumed while execution of instructions.

(Refer Slide Time: 15:51)



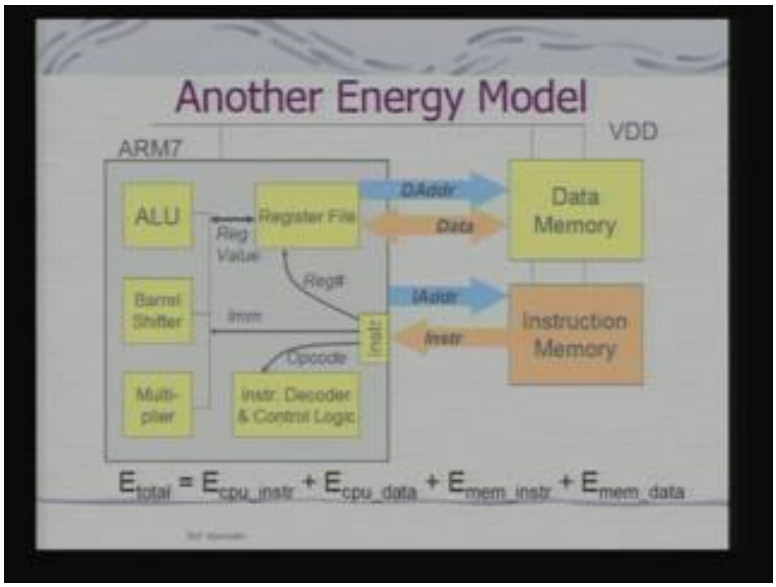
So, let us take an example of the power model. It would provide, what are called base costs and inter instruction costs, what is the base cost? Base costs of an instruction correspond to energy consumed per instruction execution. If an infinite sequence of that instruction is executed that is if each instruction. If we considered an instruction of the processor and if I want to execute that instructions in definitely then they will be certain power consumption.

So, that is the base power consumption for execution of an instruction then there will be inter instruction costs, what is inter instruction costs? The inter instruction costs model the additional energy consumed by the processor if instruction changes. Because, your program will not typically consists of a single instruction. There will be change from one instruction to the other instruction and there will be a cost involving to it.

So, additional cost can come because of switching functional units on and off. Because, there may be you have switching on your floating point functional units. So, that over ride initiation of the floating point functional unit will have an override and that should be model by inter instruction costs. So, typically a power model of a processor would

consist of a base cost for instructions and inter instruction cost. Obviously, this model does not consider the memory access costs and does not attribute that overall energy model.

(Refer Slide Time: 17: 42)



So, we can look at another model. This model is typically keeping arm 7 in mind arm seven is an architecture we had all studied. Now, we are looking at memory this is data memory and instruction memory. Please note that arm seven really does not distinguish between data memory and instruction memory. Because, it is one arm architecture for the purpose of energy modeling, when the energy model is proposed conceptually this has been considered differently.

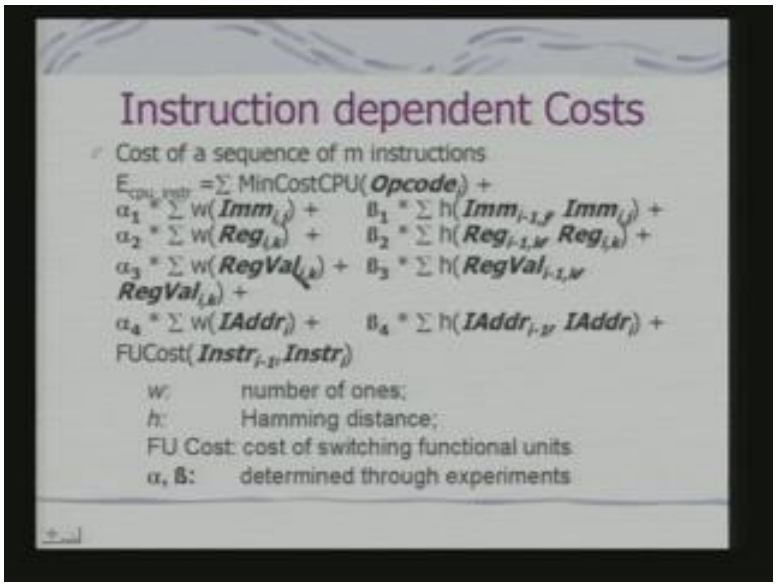
Because of the reason that, you will be fetching instruction for the instruction part of the memory and will be fetching data from the data part of the memory. And, there may be characteristics of the data characteristics of the instructions which may clear role in the energy computation. And, that is the reason why they have been model separately. In fact, the energy model that we are talking about it has been built primarily through experimentation. It may involve simulation; it will can actually involved implementation with the true hardware.

Now, the different components which have been identified in this model if we look into it one is CPU instruction that is the energy consumed by the CPU for execution of instruction energy consumed by the CPU for processing the data. Then, energy

consumed for memory that is instructions fetching from memory. And, the energy consumed by the data from the memory. So, these are the individual components.

Now next, we shall look at how these individual components can be characterized. Because, we have this individual components properly computed and create a power model compiler cannot use this information. Because, it will use this information with respect to a block of code which is the sequence of instructions and that sequence of instructions would involved, what a memory access for instructions, then memory access for data some computation by the ALU. So, all these components have to be taken into account to find out, what is the energy consumption for a block of code.

(Refer Slide Time: 20:15)



So, let us look at this model and how it can be done. So, this is one such model it is not necessarily that this is the universal model and that the only model. So, what it does is that it identifies various components for computing the costs for execution of an instruction by the CPU.

So, one part of it is that minimum cost CPU for the opcode for the op code i because, I am looking at an op code i and looking at a cost of sequence of m instructions. So, the summation over m, what is the cost of execution, of a single instruction that opcode and then the other parameters which come into the picture is depending on that this if you look into it this is a immediate operands. If is a registers then registers address operands,

then the question of is because a register value, if we are looking at the register value then if there is an address as part of the instruction.

This is the address and this is basically the instruction to instruction difference cost, because you are switching from one instruction to another instruction. So in fact, if you look into the whole model this is generalization if the previous model because you are bringing in these terms. You have brought in these terms the previous model talked about the base cost and only the inter instruction cost.

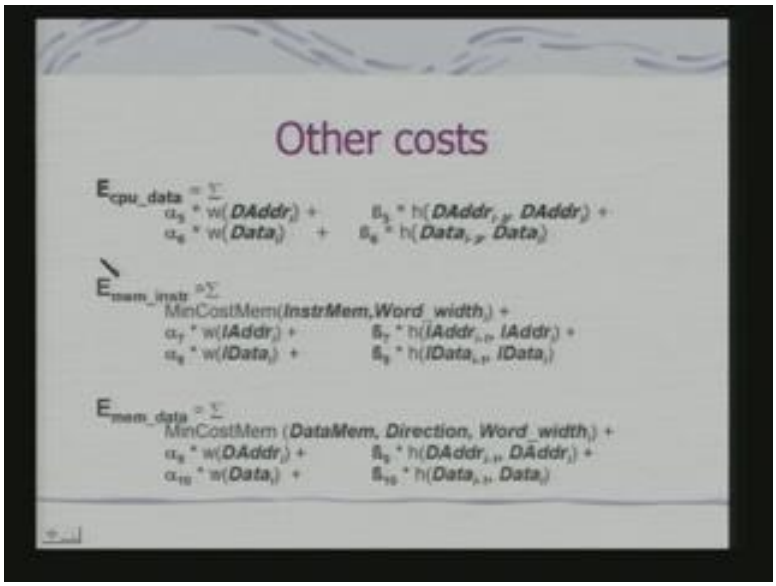
Now, you have got other terms been brought in now, what are the significance of other terms, if you look into it these counts the number of one's. Because, but it does not matter where the one is located, but it counts the number of one's corresponding to these operands which will be part of the instructions.

And, this tells you that hamming cost that is hamming distance hamming distance is the number of differences in the corresponding bit positions of these operands. If they are if they are there that is i minus one and i, so basically from the previous one and the current one. So, what is the basic idea that comes up from here, we say that depending on the number of one, because if you look into it, these are the ones which would actually float an either data and address bus.

These are the ones which have to be activated for execution of instruction they have to be put on the data or address bus. And, these are the transitions which are expected to take this. So, this combination is the energy budget for actual memory fetch or instruction fetch. Because, you can understand everything put setting a site that if I want to have any transactions on the bus, there has to be energy consumption. There has to be current injected.

In fact, these terms actually is modeling the activity which would take place takes place on the bus for execution of an instruction. Because, execution of an instruction cannot take place without reference to operands, operands can be immediate operands, operands can be register operands. There can be addresses for load store operations. So, these terms should coming in the energy model. So, this alpha and beta are nothing but experimental constants experimental constants.

(Refer Slide Time: 24: 07)

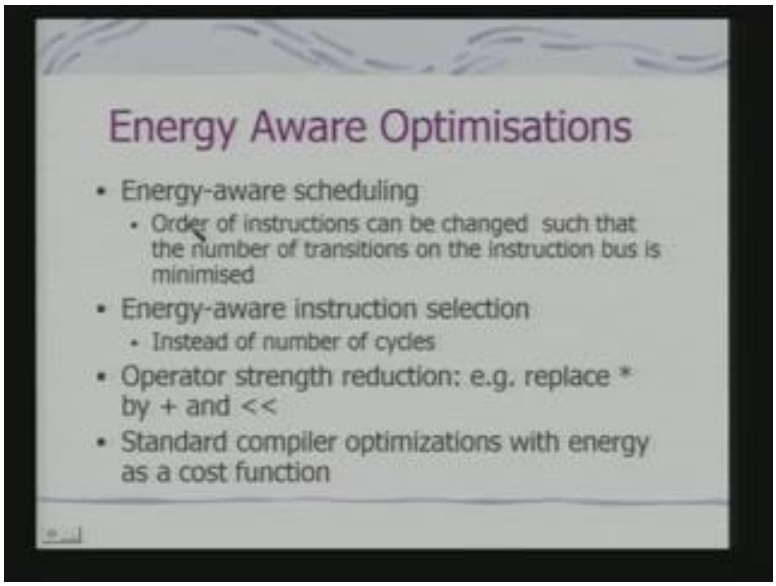


So, there are other terms similarly other terms to coming. So, here it is of the address. And data address, this is corresponding to your number of this hamming distance between the consecutive addresses, this is for the memory part. So, it is the minimum cost memory for instruction memory and word width word width will be; obviously, a function of this. Then, memory data is if we are transferring data memory direction and what width. So, again these parameters to coming in fact, you can understand all these parameters are also transaction based parameters.

So, everything put together I get therefore, the energy costs I can compute this energy cost for execution of a sequence of instructions. Because, the sequence of instructions involve what, execution of an opcode, accessing memory for data, accessing memory for instructions computing the results and then putting back the result onto the memory. All these aspects have been covered in this memory model.

See, if I provide such a memory such a power model to the compiler then for each block of code synthesized. Compiler can compute the corresponding energy budget. See, if it is computed that energy budget, what it can do; see it can do, what is called energy aware scheduling.

(Refer Slide Time: 25: 37)



So, order of instructions takes for example. The order of instructions can be changed such that the number of transitions on the instruction bus is minimized. Your transition in the energy on the bus is minimized then your energy costs terms decreased. If the energy cost term decrease then; that means, the energy which is consumed for a block of code would be less. So, that becomes the energy aware optimization.

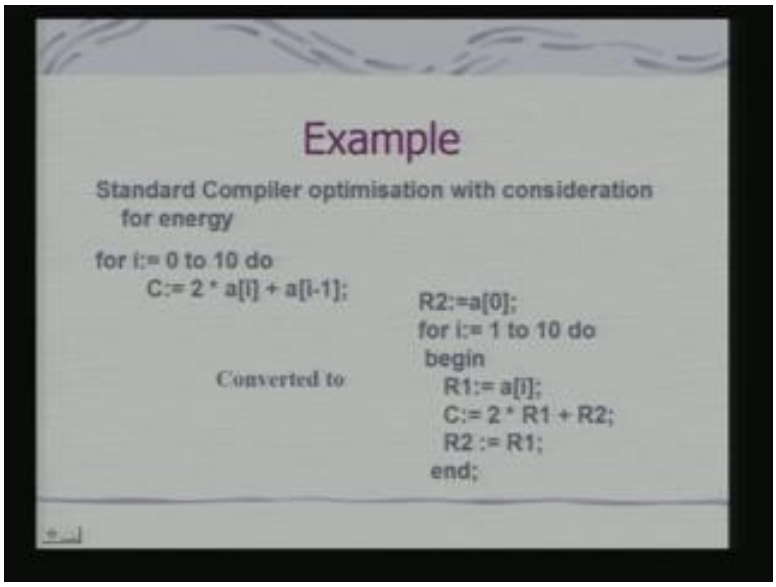
Now these can be done, the order of instructions can be changed, provided you do not have dependencies. So, it might require application of standard compiler optimization techniques, we have seen one such technique of variable renaming. So, I can do a variable renaming remove the dependencies. Then reorder the instructions, to find out a combination of instructions which minimizes the energy consumption. So, this becomes energy aware scheduling of instructions by the compiler.

The other thing could be the energy aware instruction selection, instead of looking at just of the number of cycles. It would look at the combination again we should minimize the energy consumption. In fact, if you see this becomes also part of design space exploration, what are you exploring? Different possible ordering of the instructions and in a way these exploration can be done with reference to each hardware unit that you are using in a embedded system.

The other kind of things which are components operator strength reduction replace star by plus and shift operations. Obviously, here the logic operations for the multiplications

would involve more complex logic circuits. And if it is, if we can avoid that by using simply ALU and parallel shifter an arm, I shall be getting much better energy consumption profile. And, there can be standard compiler optimizations also are used with energy as a cost function.

(Refer Slide Time: 27:59)



Now, let us take an example of this. Let us consider this code, if you look at this code this is simply a array access and I am doing a computation involving array access and taking the value in C. They can be any kind of computation this computation example that we have chosen is in a way random.

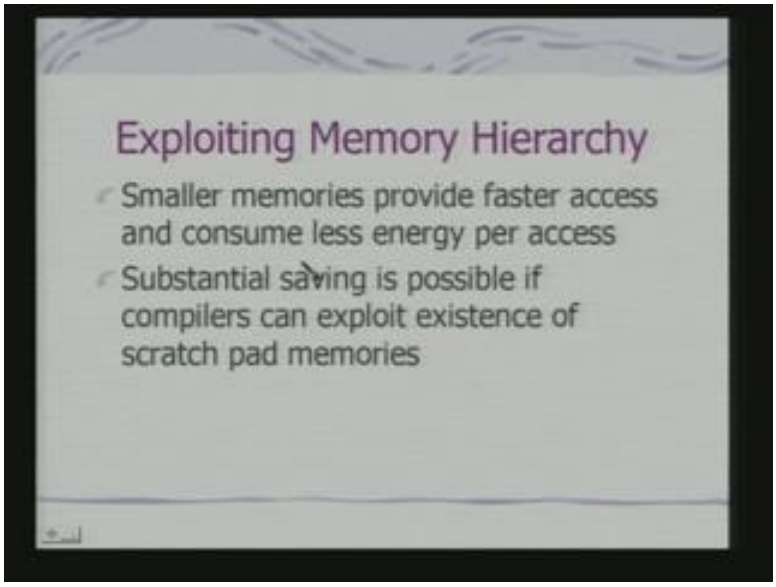
Now, if we can do an appropriate register allocation then we can reduce energy consumption substantially. Because, both these array accesses or memory intensive any array access would means activity on the bus. And, you have seen the activity on the bus contributes to the energy cost. So, activity on the bus would be there if I am accessing this arrays, question is how do, we minimize these accesses.

Now, if we are minimizing the access I need to allocate registers. Let us consider that we are having registers, R 1 R 2 just like arm kind of its scenario where I have effort to registers by R i we get consider that scenario. Now here, I am showing you one possibility where a 0 is assign to R 2 and what we are doing? I am assigning a i because this is 0 to 10 a i to R 1. So, R 2 is actually containing a i minus 1.

And then, I go through this look so this computation is same as that of this computation. But here for each computation, what I have got, I have got two array accesses. The way the code has been written. But here, I have reduced it to a single array access and I am doing the computation here involving simply this registers R 1 and R 2. In fact, a very standard compiler optimization would possibly do this as well. Because, I am saving on time also,, but at the same time, I am saving on energy.

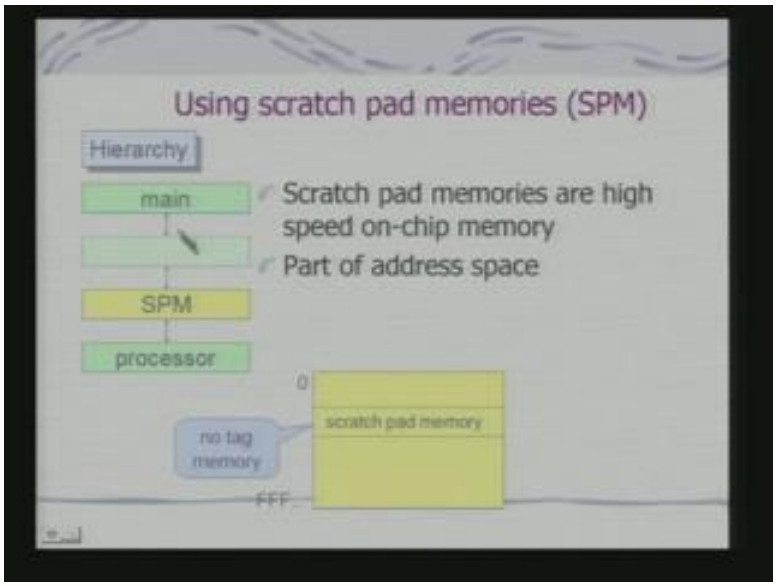
So, energy aware compilation becomes very important for battery power devices. Now, these kinds of instruction choices can say energy to a certain extent. But, the major it has been found that the major saving on the energy can be really done, if we reconsider the memory architecture. Because, you have seen from the all this models memory access becomes a critical component. In fact, some smaller memory provides faster access and consumes less energy per access.

(Refer Slide Time: 30:51)



And, substantial saving is possible if compilers can exploit existence of what are called scratch pad memories. In fact, we have discussed scratch pad memories when we looked at the memory organization.

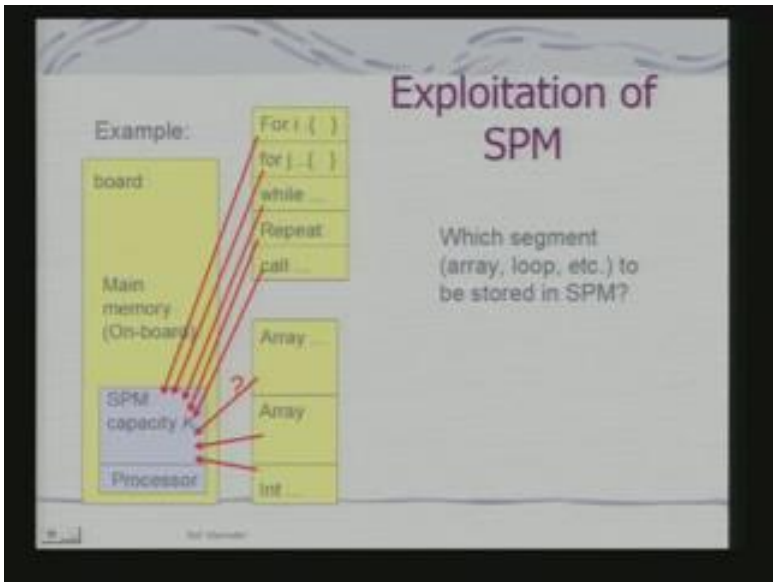
(Refer Slide Time: 31:11)



So, what are really scratch pad memories, scratch pad memories is high speed on chip memory, what is the difference with cache memory? Cache really are not part of an address space its mapped from the main memory to cache, but scratch pad memory is part of the address space and it is on chip and fast.

See if you look into it, what we are looking at is a processor and have got SPM. This can be cache memory and this is the main memory which gets mapped onto cache. And this SPM is part of the address space and that is why I am referring this as no tag memory, because if it is a cache then it has to be tag.

(Refer Slide Time: 31: 59)

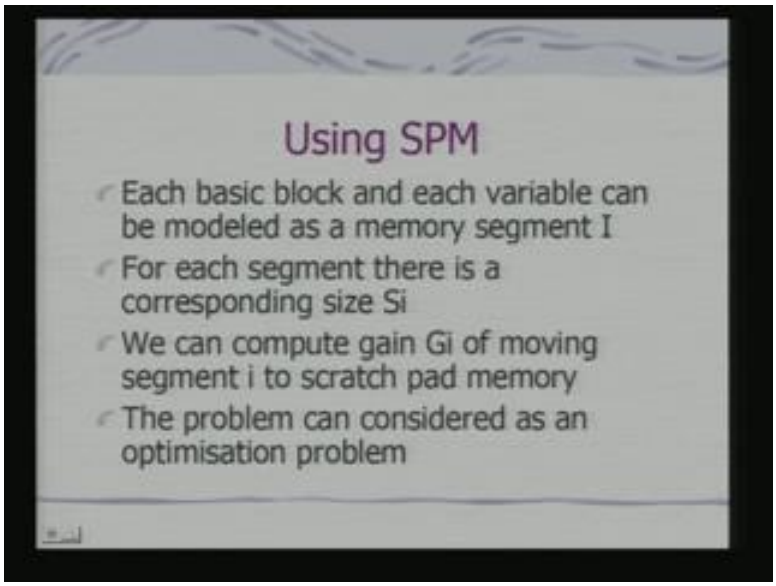


Now, how do you exploit SPM, because if I have SPM I need to exploit it. I can exploit it if compiler knows how to exploit. So, the compiler has to be now made conscious about existence of scratch pad memory. Now, how can compilers, exploits scratch pad memory your program would consist of various blocks there can be for loop while loop repeat call data structures there can be arrays.

The question is compiler has to take a decision about putting which of this blocks code blocks as well as data blocks or either code block or data block into the SPM which has got of fixed capacity. So, that your execution speed is maximized; that means, the time taken for execution of the code is minimized. So, I cannot put blindly the things onto SPM it is not like a cache please keep in mind the problem is different.

In case of a cache, when you make an access then that memory gets mapped on to cache. And, you try to organize the data and instruction in such a way that cache hit is maximized. In fact, we have discussed this point earlier. Now what are we looking at, we are looking at SPM if I looking at SPM. One possibility is a static allocation; that means, the compiler decides at the compile time itself to allocate this blocks onto SPM. So, there will be on the SPM always. In a cache it is not a static allocation cache is typically a dynamic allocation depending on demand.

(Refer Slide Time: 33:49)

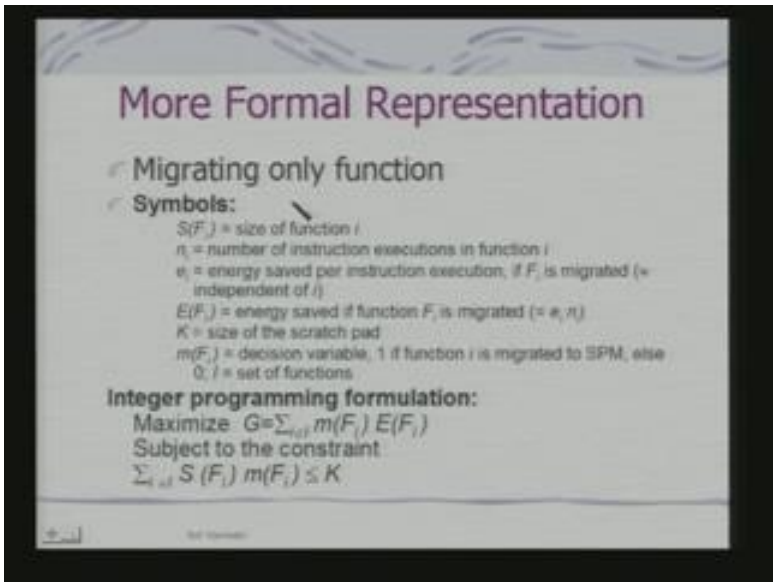


So, how this problem can be solved? So, we can identify each basic block and each variable can be modeled as a memory segment I. So, basically compiler is dealing with a set of memory segments. And for each segment, there is a corresponding size Si. And, we can compute gain Gi of moving segment i to scratch pad memory. Because, what is the gain? Gain would come from if that block I is in the main memory, what is the time taken and what will be the gain? If I move that from main memory, to a scratch pad memory.

Now, this gain can be estimated using profilers. We are talked about profilers which can give you the estimate of code segments in your program. Choosing profiler, you can figure out what could be the possible gain of moving a segment i to scratch pad memory. After that the problem becomes the optimization problem. Because, what would you like, you would like to optimally exploit, the SPM area.

Optimally exploit the SPM area to maximize the gain. So, it is the pure optimization problem just like a hardware software partitioning problem. In fact, this is the partitioning problem in the sense that you have got the code and you are trying to partition the code between main memory and SPM.

(Refer Slide Time: 35: 24)

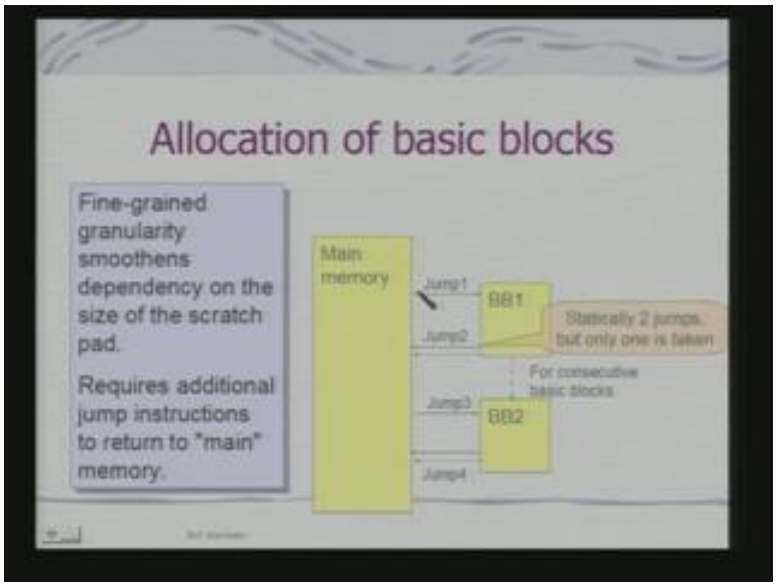


So, a formal representation could be in this form where we are looking at simply migration of functions. We may not look at each variable, but we are looking at say simple migration of functions. So, size is each function could have the fixed size; this is size of the function. n i is the number of instruction execution in the function i, e i is the energy saved per instruction execution if Fi is migrated and it can be really independent of i.

EFi is energy saved if function Fi is migrated. So, it can be computed by multiplying e i and n i. K is the size of the scratch pad memory and m Fi is the decision variable. If 1, if function is migrated to SPM l 0, so I have got effectively of 1 0 optimization problem 1 0 integer programming problem. Because, I am trying to do what, I am trying to maximize this gain. This gain is EFi, EFi has to be multiplied by the decision variable and sum it up. So, i shall be getting that is the total gain.

And I want to get that gain. Subject to the constraint that SFi this is the size of the function i multiplied by the decision variable is less than k which is the size of the scratch pad. So, this becomes an integer programming problem. So, compiler should have this kind of an optimization routine to figure out, what are the functions, which have to be migrated to scratch pad memory if it is required.

(Refer Slide Time: 37:17)



So, effectively we shall allocate the blocks. So in fact, this is the block BB1 and block BB 2 which gets allocated to your scratch pad memory and your main code is in the main memory. So, effectively we have to introduce jump instructions and what we say is that fine grained granularity smoothens dependency on the size of the scratch pad. Required additional jump instructions to return to main memory and in fact, why fine grained granularity smoothen dependency on the size of the scratch pad. Because then, if the granularity small of the unit that we are considering.

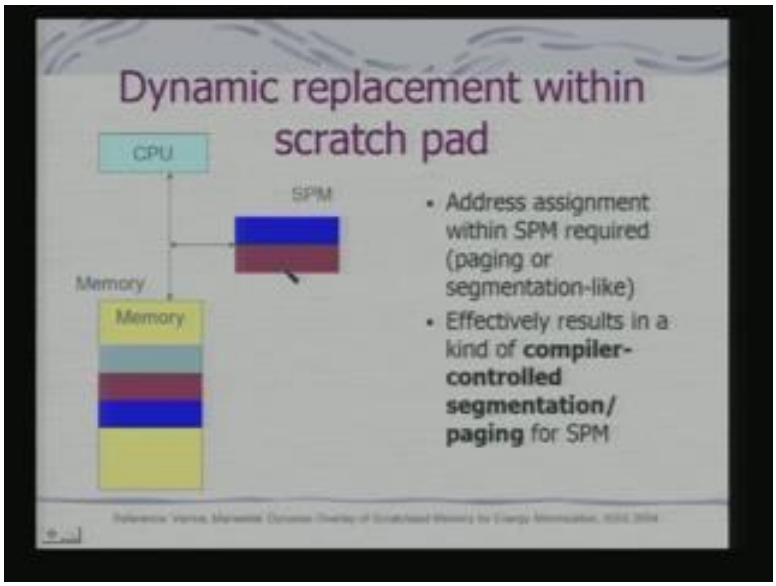
Then, you can migrate them smoothly if there is small increase in the scratch pad then one more granular can moving. So, the efficiency increases in a smooth fashion. And obviously, would requires this additional jump instructions and in fact, there could be an interesting optimization with respect to this also. Because, if there are consecutive blocks mapped then this jump instructions can be actually removed.

Otherwise, the basic frame work is your main code is in the main memory introduce jump instructions migrate part of the code to your SPM. And, again put back put a jump instruction and the end of the code to go back to the main memory. In a way you are violating the principles of structure programming, but you are violating to exploit the memory characteristics.

In fact exploiting this scratch pad memory in such behavior, in fact it is found that use of scratch pad memory and by optimally mapping the blocks to scratch pad memory you

can have substantial energy saving. In fact, many of this cell phone based appliances where they have the GSM coder and GSM coder implementations bench marks have actually shown that migrating. The code onto scratch pad actually enhances battery life substantially.

(Refer Slide Time: 39: 29)



Now, other possibility could be dynamic replacement within scratch pad. So, the basic idea is that address assignment within SPM is required something like paging or segmentation like. The basic idea is this is the code and I can have these are the pages of the code logical pages of the segments of the code. And, these segments are move to SPM depending on the requirements.

And, effectively results in a kind of compiler controlled segmentation or paging for SPM. So here, the ways is not really coming into the picture. But, compiler is coming into the picture, because compiler is mapping the different components of your code which is in the memory to SPM depending on the requirements.

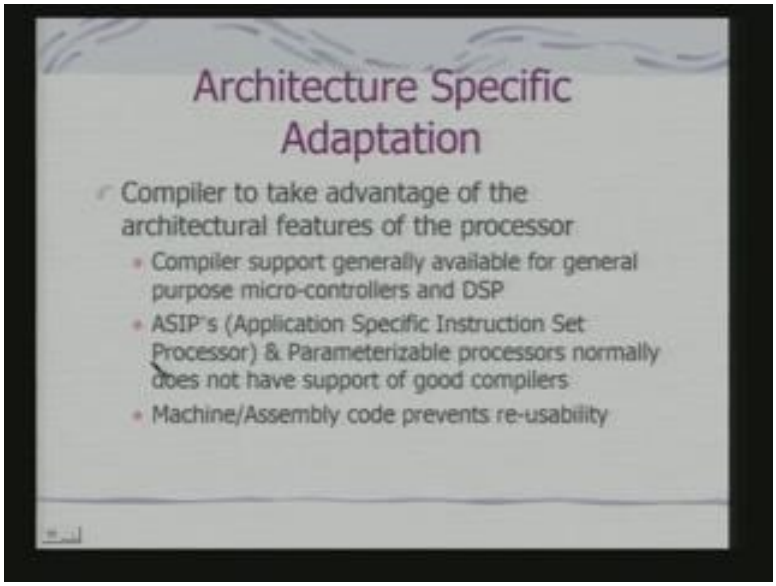
Now, why this should be a desirable strategy, because if you remember the locality of reference is the temporal phenomena; that means, I might be executing a function a number of times over a certain period in the program. At some other point in the program I might be executing some other function. And, it may not be possible format to accommodate both the functions in SPM. Because, SPM also involves cost i cannot have

everything SPM because SPM involves cost. So, I would like therefore, a function which is currently being used every being mapped on to SPM.

And, later on the other part of the function will get mapped onto SPM when that function is being used in my code. In fact, extending this idea would be if you have a multi tasking model multi process model with the OS. Then, how the multiple processors have to accommodate on the SPM, because accommodation of multiple processors and then becomes a problem of OS.

And, under those circumstances you have to see, what is the SPM budget that you are allocating your compiler? If I am statically doing an allocation in a process specific fashion, so I have again changed. So, one has been overridden the refresh I was mapping to temporal locality that it is now then these part which is currently in SPM for execution.

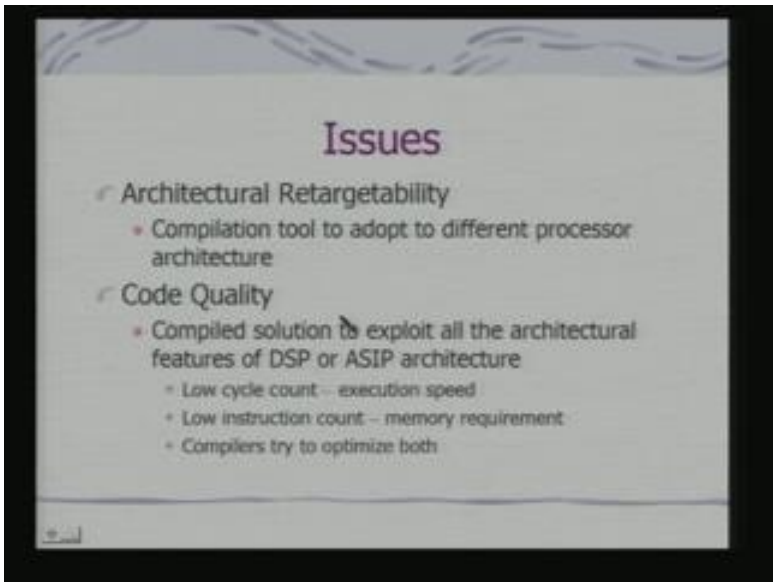
(Refer Slide Time: 42:01)



Next, we shall talk about other architecture specific adaptation. This architecture a specific adaptation refers to processor architecture. Compiler needs to take advantage of the architectural features of the processor. So, it can be for DSP, because DSP instruction set architecture is different. There can be ASIP's, ASIP's are application specific instruction set processor. And parameterizable processors, normally does not have support of good compilers.

Now, this is the very big problem when you are talking about a platform based design. Because if we are parameterizing of processor, how can you have a compiler, which is efficient and have put in all this optimization, so that you can translate your high level language code to the target architecture. So, design space exploration actually can involve designing of the processor along with the compiler with selection of instruction set. So, that finally, you get optimized implementations.

(Refer Slide Time: 43:09)

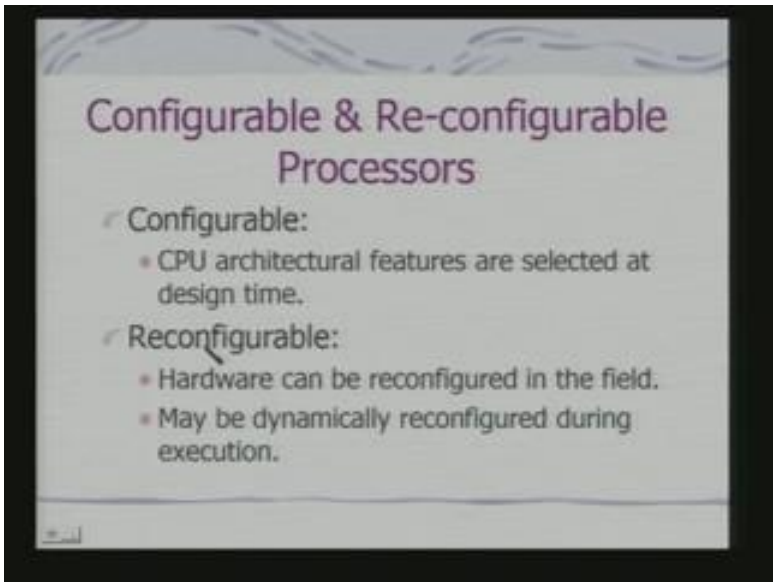


So here, the issues are architectural retargetability. Compilation tool to adopt to different processor architecture. So, we have talked about so far compilers tuned for a particular architecture.

Now, we are talking about whether I can have compiler retargeted for various architectures. In fact, that is more feasible if I have a compiler targeted for a family, but now I parameterized on various features of that family. So, in that case can I have a compiler generator for a parameterized processor. Because, the moment I am parameterizing processor, I have also the flexibility to choose it instruction set.

And accordingly, the compiler should get automatically modified to generate the code because using that you can explore the design space. The code quality is; obviously, compiled solution to exploit all the architectural features of DSP or ASIP architecture would be low cycle count low instruction count and you would like to optimize this along with the energy as well.

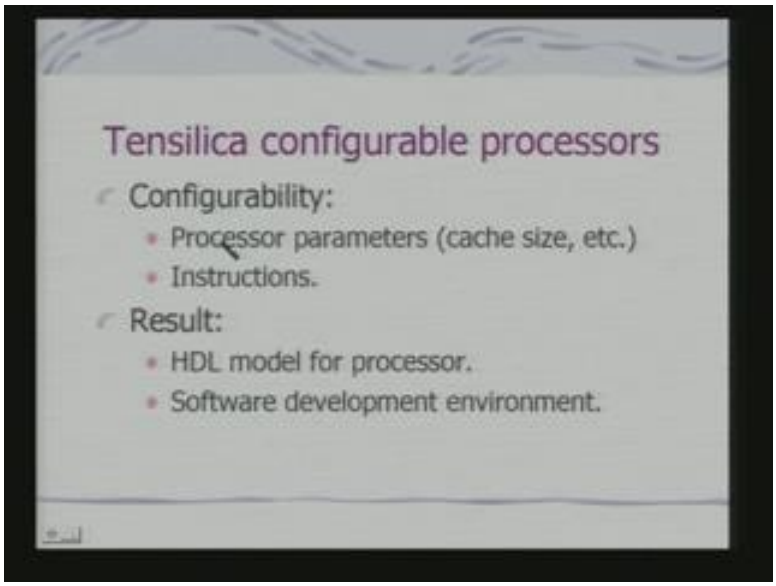
(Refer Slide Time: 44:14)



Let us take an example of these configurable and reconfigurable processors. Because, this exactly the point we have discussed earlier also. If I have the configurable processor the CPU architectural features are selected at design time. Reconfigurable means hardware can be reconfigured in the field and may be dynamically reconfigured during execution.

Now, in that case what kind of instruction do you generate, what kind of instructions the compiler should generate to the optimal with respect to the architecture which can undergo modifications. This is actually an open and series problem. We shall obviously, not look at this problem in its complete complexity we shall just briefly touch upon its different aspects.

(Refer Slide Time: 45:01)

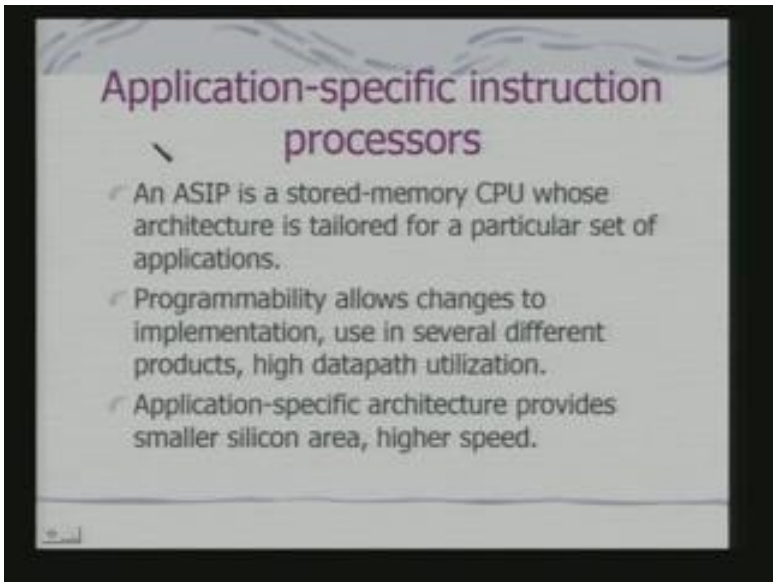


We are looking at an example say Tensilica configurable processors. So, you can actually configure the processor parameters see again an example of the soft core kind of a scenario. So effectively, what you get, you can configure processor parameters as you configure parameters, we can also design an instruction for the processor.

Now, you get a result is HDL model for processor. Hardware description language model which can be in VHDL and you need actually a software development environment. If you do not have a software development environment, you really do not know, what to do with configure processors.

If you see we have studied the design of embedded systems design of embedded systems is what not just designing the hardware. If you are coming with a task we need to mapped software to hardware. Hardware block if required task should be mapped to hardware blocks develop the software for it. So, I need a complete environment evaluate the processor and do a design space exploration to select the platform instance.

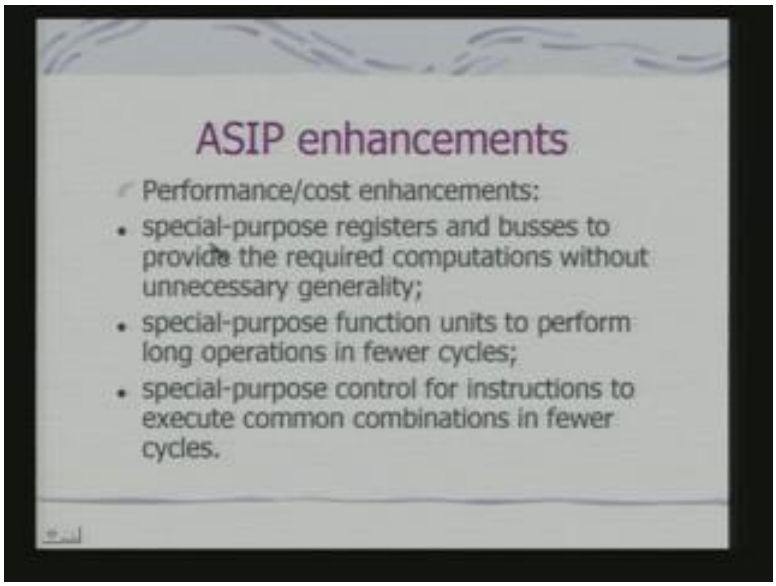
(Refer Slide Time: 46:06)



Related to this is a concept of application specific instruction processors. ASIP is a stored memory CPU whose architecture is tailored for a particular set of applications. In fact, the current trend is go for ASIP rather than ASIC. Because then, ASIP gives you the kind of a flexibility. So, that it can be used for a number of applications exploiting the similar characteristics instead of just designing an ASIC targeted for one specific function.

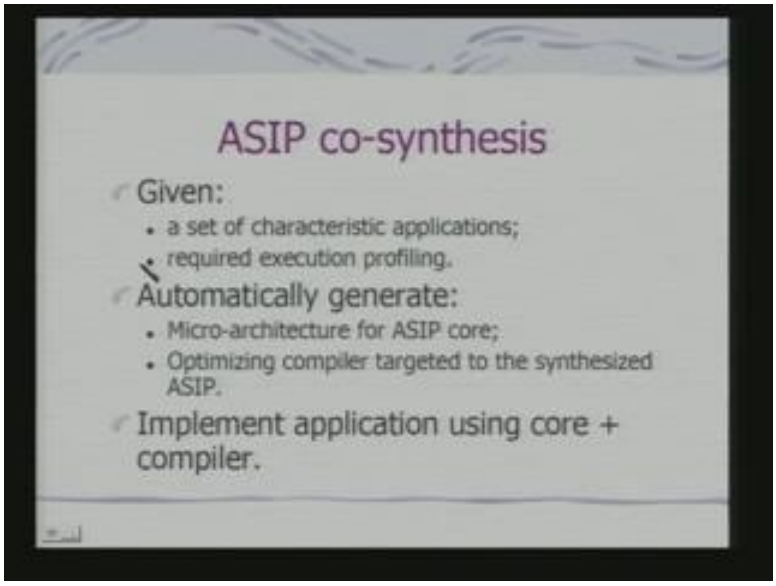
So, the programmability allows changes to implementation use in several different products and high data path utilization. In fact, this is the basic reason for going for ASIPs. And, application specific architecture provides smaller silicon area higher speed.

(Refer Slide Time: 46:51)



So, enhancements of performance and cost enhancements, special purpose registers and busses to provide the required computations without unnecessary generality. Special purpose function units to perform long operations in fewer cycles and special purpose control for instructions to execute common combinations in fewer cycles. Now, all these are features of ASIPs now and these ASIP can be actually designed in an application specific fashion.

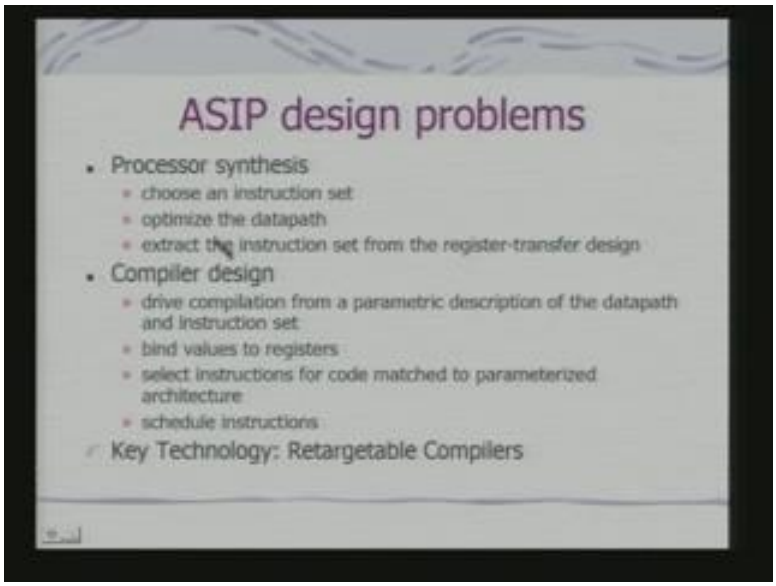
(Refer Slide Time: 47:18)



So, the problem that term out is what is called ASIP co synthesis problem. So, you are synthesizing a processor. So, given a set of characteristics applications and required execution profile. You would like to automatically generate micro architecture for ASIP core and optimizing compiler targeted to the synthesized ASIP.

So, you are not just designing now the architecture, you are also designing the compiler. Because, if you do not have the compiler you cannot really have your task executed on the target processor in an optimized fashion. So your question, this is problem is that of designing the architecture as well as designing the compiler. So, implement application using core and compiler.

(Refer Slide Time: 48:07)

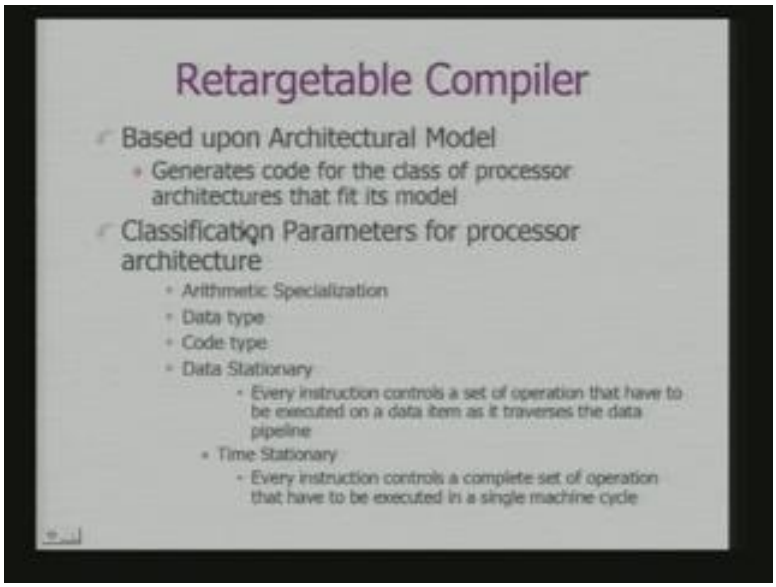


So, processor synthesis chooses an instruction set optimize the data path. Extract the instruction set from the register transfer design. And, compiler design drive compilation from a parametric description of the data path and instruction set. Bind values to registers select instructions for code matched to parameterized architecture and scheduled instructions.

If you remove the parameterized part of the architecture these are the job which are done by the compiler as it is. But, your standard compiler does not work with the specification of the processor. You have already known the processor and then you are built in the algorithms in the compiler to do the optimization. Now here for the compiler, you need an additional input that is the description of the architecture itself.

So, the key technology in this case is retargetable compiler. So, what is the retargetable compiler, it is the compiler which is just not a language specific compiler. But, it is placed upon architectural model.

(Refer Slide Time: 49:15)



It generates code for the class of processor architectures that fit its model. Because, even when you are generating and ASIP. In fact, ASIP if I can generate from any of these soft core model; that means, you have got a parameterized family in many of the cases. So, you have not changing the architecture. But, you are playing with some parameters some features of that architecture keeping basic family and changed.

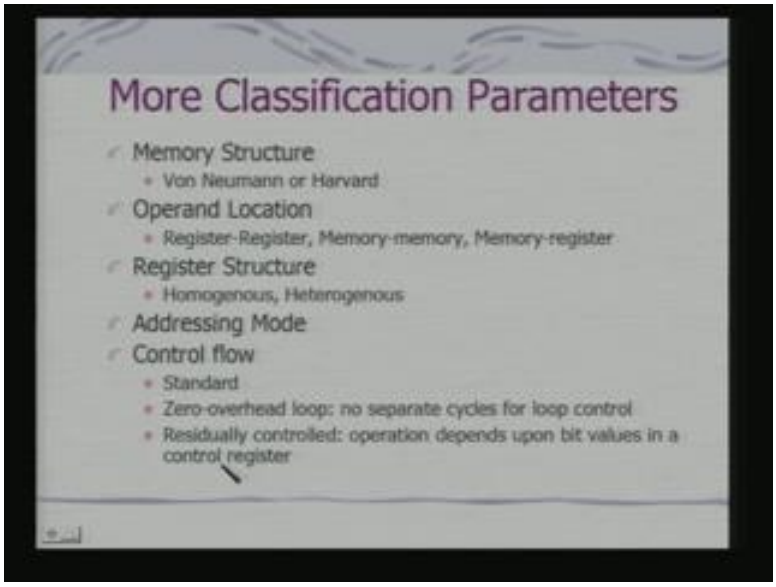
So, there are various ways to classify parameters for processor for architecture. It cans an arithmetic specialization that whether we are using a real or a floating or integer or fixed point. Data type being used. Then the code type, now code type can be two kinds in fact data stationary or time stationary, every instruction controls a set of operations that have to be executed on a data item as it traverses the data pipelines. So, this is called data stationary code type architecture.

Time stationary is every instruction controls a complete set of operation that have to be executed in a single machine cycle. Now, please try to understand the difference between the two. Data stationary is every instruction controls a set of operation that have to be executed on a data item as it traverses the data pipeline. There is the data pipeline, the

data traverses and you are talking about the set operations should be executed on the data path.

Time stationary is controls a complete set of operation that have to be executed in a single machine cycle. If you considered a VLIW kind of a processor, whether a multiple functional units then I can schedule operations on each functional unit and that can work on different data. So, you have got a time stationary model vice versa a data stationary model in terms of architectural features.

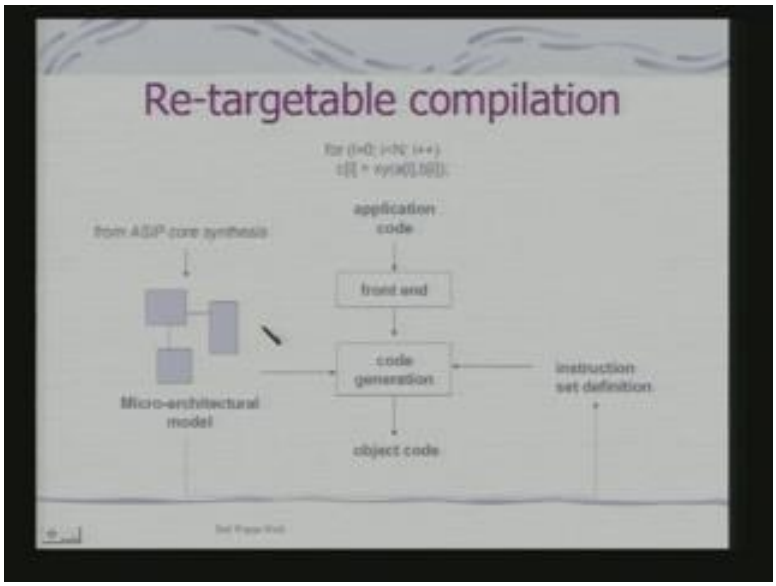
(Refer Time: 51:28)



There are other classification parameters as well say memory structure Von Neumann or Harvard. Operand location whether it is a your register, register operation or memory memory or memory register, register structure if it is a RISC kind of thing it will be homogeneous if it is a CISC then it will be heterogeneous, addressing modes control flow. Because, they can be standard control flow or if you are bringing in some of the DSP features, you may have zero overhead loops as a control flow feature in the architecture itself.

And, so they need not be separate cycle for loop control also. There will be residually controlled in the sense operation depends upon bit values in a control register. Say for example, with arm if you said the bit the instructions or operations are 16 bit.

(Refer Slide Time: 52:17)

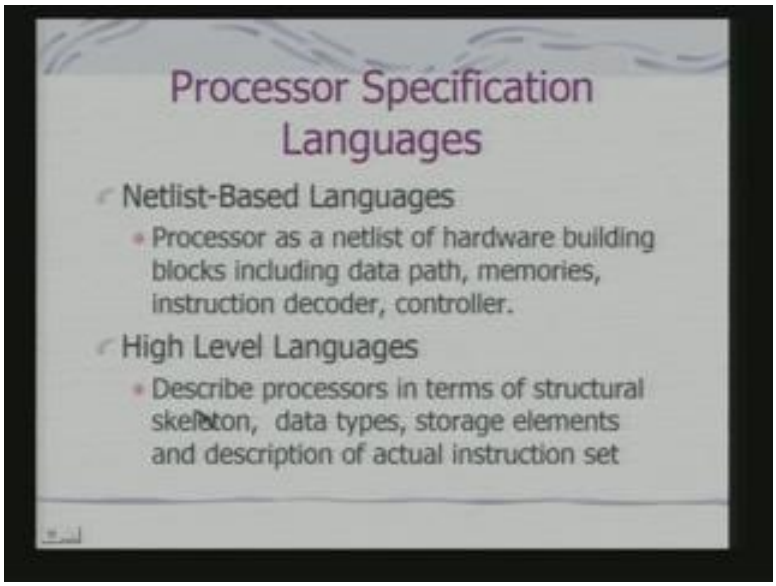


So, then with these features you get this model of retargetable compilation. So, this is say simple C code this is an application code which goes to front end. You code generation makes use micro architectural model that architectural model can be in hardware description language or various other languages which can used for describing high level architecture as well as the instruction set.

And then, you generate the code, because C this instruction set definition also comes from here. So, you make use of the instruction set definition look at the micro architectural model and then look at the object code. Now, if we are actually doing a design space exploration it means that you have the object code execute.

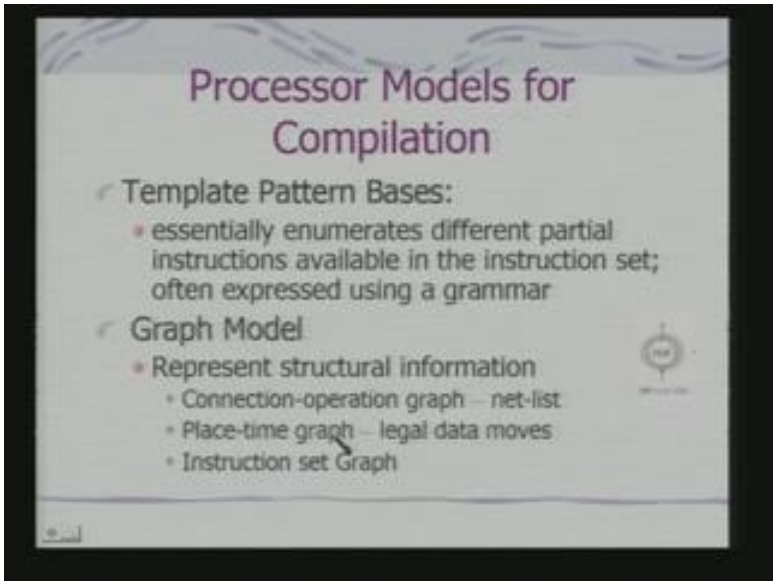
The object code get the performance estimates go back modify the micro architectural model modify the code generation again try it out. So, if you are actually doing this kind of a soft core base development or a platform based development this is one way to do design space exploration.

(Refer Slide Time: 53:24)



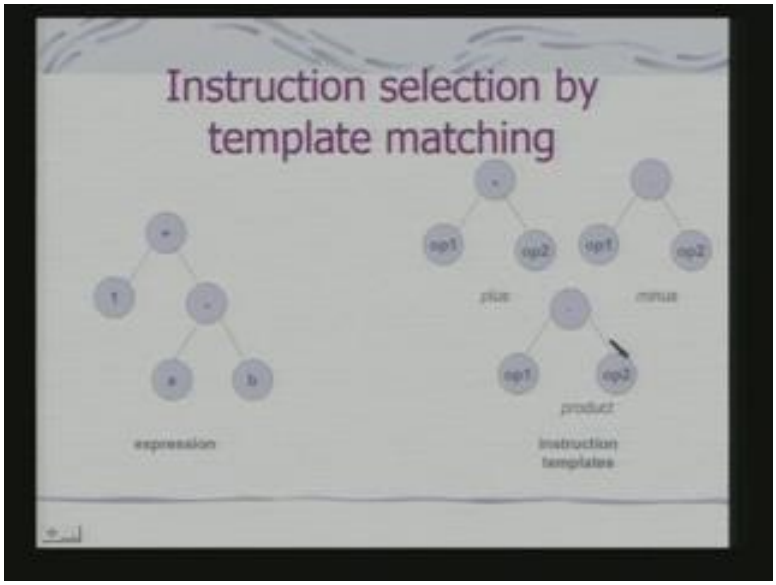
The point I was mapping that compilers are also tools for design space exploration. In fact, it is a design tool for embedded system. So, there are various ways you can specify the processor, there can be typically Netlist Based Languages. So, Netlist Based Language is gives you a structural model there can be high level languages which can be describe processors in terms of structural skeleton data types storage elements and description of actual instruction set.

(Refer Slide Time: 53:59)



Then, there are template pattern bases essentially enumerates. Different partial instructions available in instruction set and often expressed using a grammar. So, by modifying these rules you can generate other instructions. Also, there can be graphical model. So, you can have connection operation graph place time graph which permits you how the legal data moves also instruction set graph.

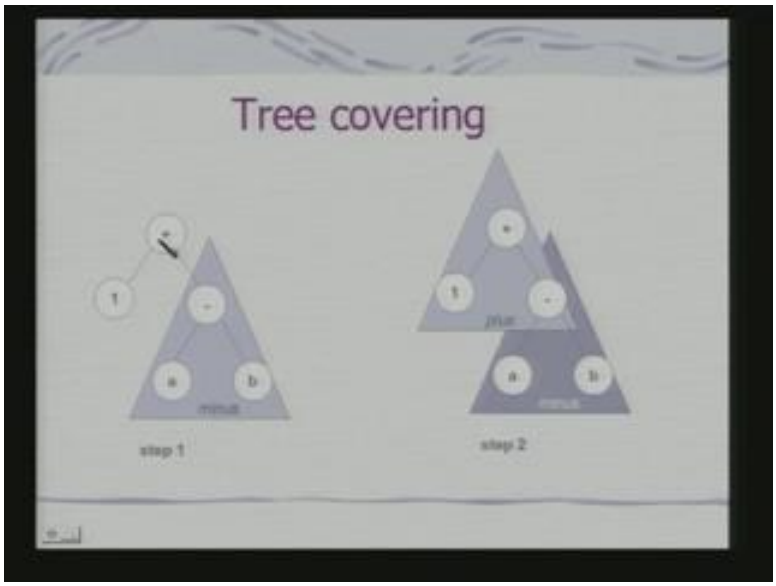
(Refer Slide Time: 54:23)



In fact, we can see that if I have an instruction set graph, what is that mean, if it is a plus operation. This is the simple instruction set graph, this is the minus, this is multiplication. And, this is expression tree which is generated through a code generation and semantic analysis. So, your code generation would therefore mean mapping of this onto this instruction graph and. In fact, you can modify this instruction by changing the graph.

And, there could be various ways of describing these instructions. So, just keep in mind that I am specifying the processor. I can design a processor, I can get a hardware description language or which describes the micro architectural model. From there I can derive the instruction set. I can describe instruction set in various ways and the compiler makes use of the micro architectural model as well as the instruction set to generate optimized code.

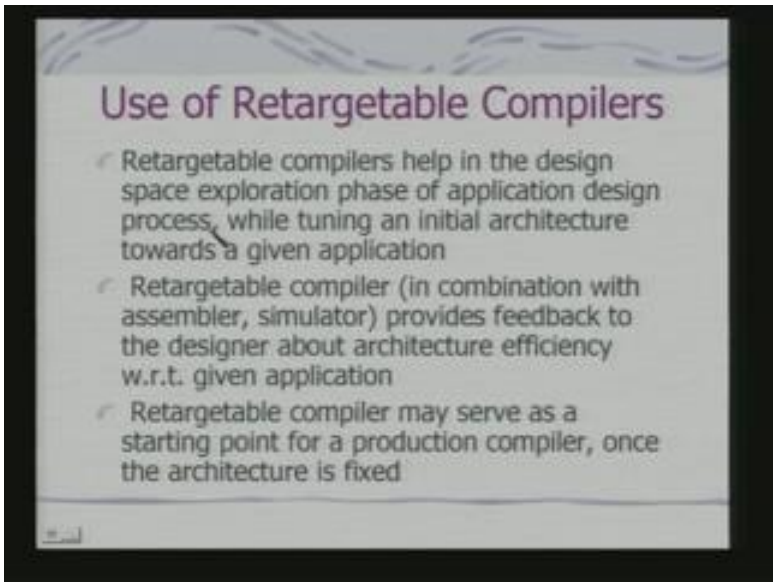
(Refer Slide Time: 55:23)



Now in fact, the tree covering is very simple algorithm is that of generating code in this way. Because this is instruction, so I have covered this with one operator. Then I choose, what is the next operator? I can make use of. So, I get this next part covered so; that means, this could be a register operand which has to be replaced to it. So, that is how I can generate, now instructions given a graphical model of a set of instruction giving in a graphical model.

In fact, this algorithm is used for general instruction selection as well can be used for general. It is actually used for general instruction selection, but in that case these models are provided right at the beginning there are not input to the compilation phase. For the retargetable compiler, the input during compilation phase itself.

(Refer Slide Time: 56:11)

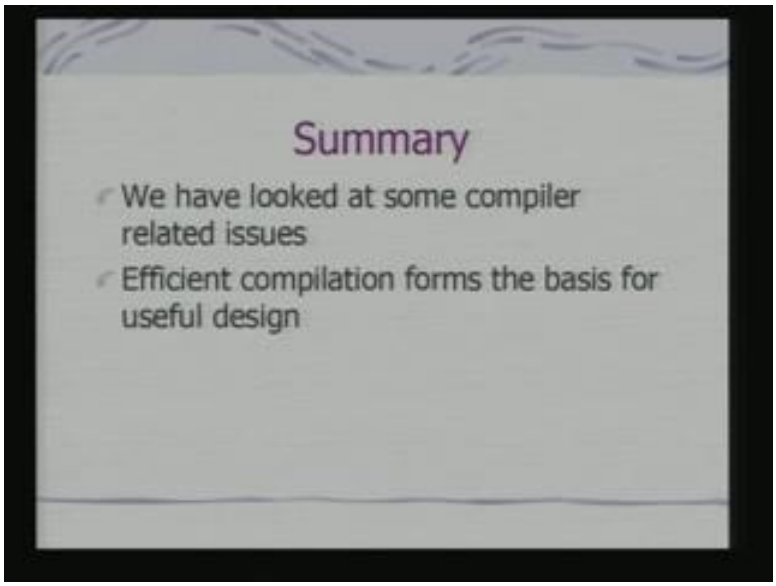


So, we see what are the uses of retargetable compilers, retargetable compiler helps in the design space exploration phase of application design process while tuning an initial architecture towards a given application. Because, I can modify the architecture modify the instruction set generate instructions set absolutely tuned for my application, if I have this kind of parameterizable features.

Retargetable compilers in combination with may be assemblers and simulator provides feedback to the designer about architecture efficiency with respect to a given application. And retargetable compiler may serve as a starting point for a production compiler. Once the architecture is fixed; that means, when you are designing the architecture this is not for a configurable or reconfigurable platforms. This is a design process of a processor itself.

So, in a way retargetable compiler is an interesting design tool for embedded system designers if you are trying to design an embedded appliance on a configurable system or using some kind of a platform which gives you the features to modify the hardware architecture.

(Refer Slide Time: 57:24)



So, today we have looked at some of this compiler related issues and we have understood that efficient compilation forms the basis for useful design of embedded systems.

Any questions?