Embedded Systems Design

Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

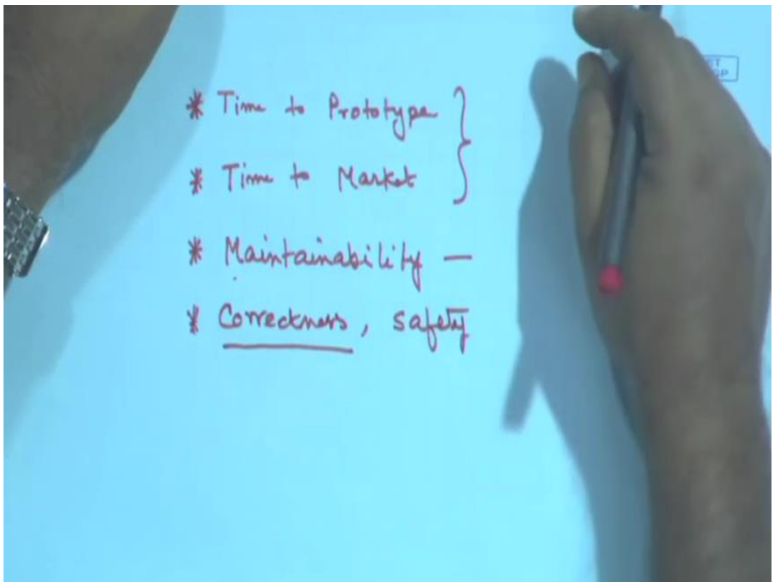
Lecture - 02
Processors

Good morning. In the last class we had discussed about what an embedded system is, and we have talked about some of the different constraints that we need to optimize and which are very important for embedded systems such as the energy cost, right the cost the size of the embedded system that we want to make, the performance, power, power is a very important thing. And I do not remember whether I mentioned or not but I would like to repeat, flexibility is a very important aspect.

Now, when you talk about performance, we distinguish between two parameters mainly: one is the latency and one is the throughput. Latency is for one operation or one transaction how much time it takes that is the speed of the system and depending on the architecture the number of solutions that are coming up can differ if I have got parallel processing and parallel threading. And the design metrics are often competitive among themselves like the power, the performance we can increase the performance for example, by increasing the power, but again we cannot go on increasing the performance power to any arbitrary extent.

Similarly the size; I can put in more and more hardware for an embedded system, but that will increase, the size and there is a size constant and of course, the energy constant is there.

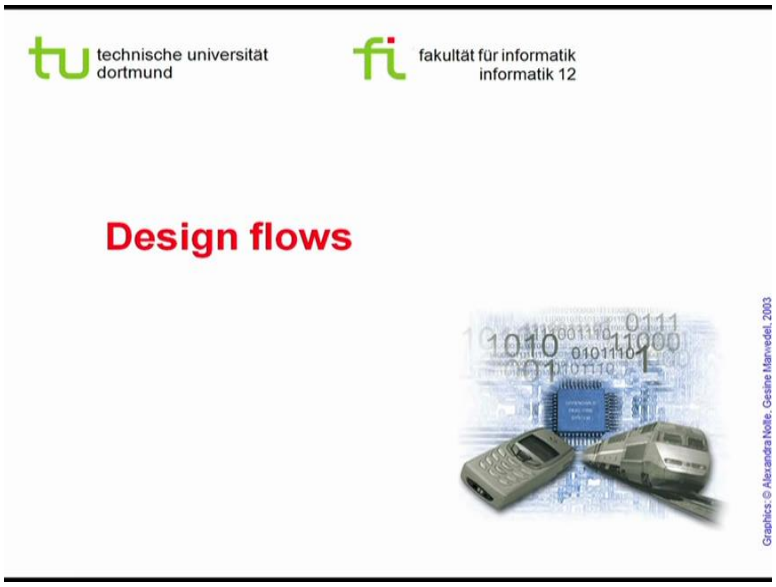
(Refer Slide Time: 02:07)


* Time to Prototype
* Time to Market
* Maintainability —
* Correctness , safety

Now there are two other constraints which I did not mention last time, one is that is the time to prototype, time to prototype; how much time it takes to build a new system? On one hand if I go on delaying I if the time goes on increasing, then it can be adding to the energy cost it may be adding to the energy cost, but the other thing is that there will be other competitors in the market and for other reasons also we need to have a quick prototype. I have got an idea I want to have it implemented very quickly. Will see later, that for that we can take different parts one might be simulation simulating the thing or we can also implement it on programmable devices such as the FPGS we will see those later.

The other is time to market. So, these are somehow related, right that if I make a quick prototype, then I can test, it refine it and then I can reach the market with the proper point of time. The other thing that is important is the maintainability, how easy it is it to maintain? That means, I if there be some problem, how quickly I can recover from that problem right? The other thing is that suppose I want to add a new feature, how quickly can I add that new feature that is also maintaining the system right? I find I came to the market with some objective, and I find my assessment was wrong. So, I need to slightly modify it how quickly I can do that? The other very important thing which is required is a correctness; correctness, safety and others. Now this correctness means many things, it must be correct functionality, functionally it must also be correct; also it must be correct as regards the timing. If I do the correct thing at the wrong time then much delay it I come to the correct solution, but much delayed after the might be some hazard that was supposed to were I was supposed to intercept has occurred right and safety is off course very important.

(Refer Slide Time: 05:16)


technische universität dortmund

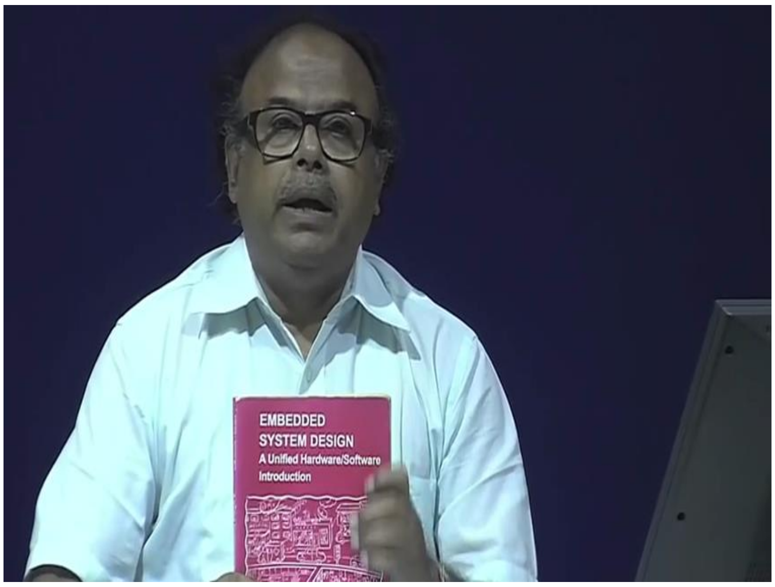
fakultät für informatik informatik 12

Design flows

Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

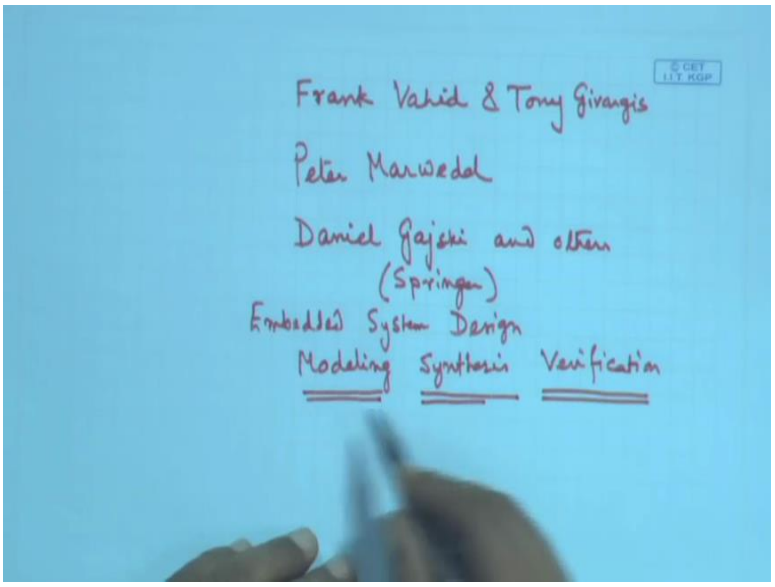
Besides this now we will now move on to the basic design flows that are involved in an embedded system design, this slide as you can see has been taken from the technical university of Dortmund.

(Refer Slide Time: 05:40)



So, by the way the books that will be referring to are one is this book by from Frank Vahid.

(Refer Slide Time: 05:50)



Frank Vahid & Tony Givargis

Peter Marwedel

Daniel Gajski and others
(Springer)

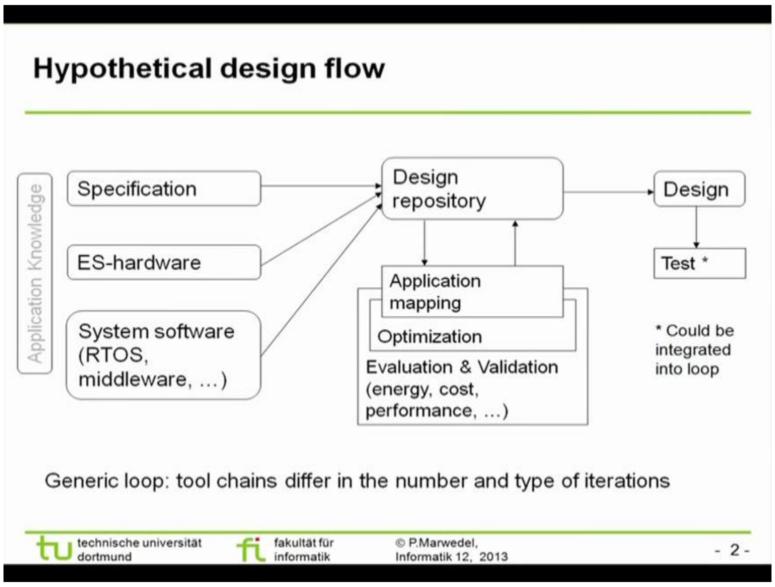
Embedded System Design
Modeling Synthesis Verification

Frank Vahid I will write it down here and Tony Givargis this is embedded system as you can see, the name of the book is embedded system design, a unified hardware software introduction alright. This is a book by John Wiley and it is a low cost book. The other book is by Professor Peter Marwedal from the technical university of Dortmund or university of Dortmund is this book embedded system design, alright the author is Peter Marwedal Vahid and also peter Marwedal at different at during the course of the lecture.

The third book that you look at is by Daniel Gajski and others by Springer and the name of the book is embedded system design modeling synthesis and verification. Now the title of this book actually throws some light on what we really intend to cover in this course, we will start with some synthesis and will come back to synthesis, but since our objective of this course is to have a learn automated or semi automated approaches to embedded system design, will very much concentrate on the modeling part as well and how computer software can be developed to achieve those objectives and if time permits we will also look at some verification.

So, these are the 3 books, but you can procure them or you can you will get enough material on the internet also and I will also at times site submitted materials.

(Refer Slide Time: 08:56)


Hypothetical design flow

Application Knowledge
Specification
ES-hardware
System software (RTOS, middleware, …)
Design repository
Application mapping
Optimization
Evaluation & Validation (energy, cost, performance, …)
Design
Test *
* Could be integrated into loop

Generic loop: tool chains differ in the number and type of iterations

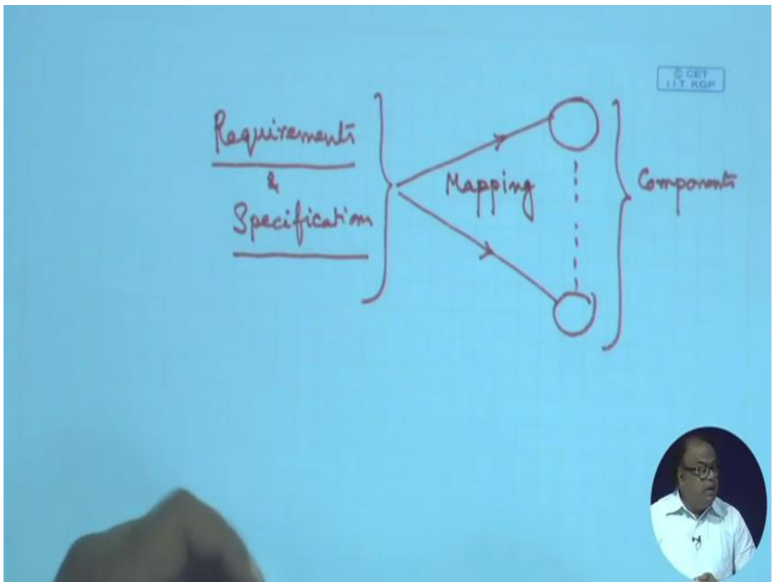
technische universität dortmund fakultät für informatik © P.Marwedel, Informatik 12, 2013 - 2 -

Now, coming to the basic design flow of embedded system design, you can see here this site is application knowledge. We must have the application knowledge for whatever purpose we design an embedded system, what is the domain? If I want to design a motor controller, I need to know how does a motor work; if I want to control temperature, I must know the physics of the temperature process. That is the application. So, accordingly we must build up the specification.

So, one part is a specification; now there are some embedded system hardware, now it can be the designers decision to decide on the hardware that I will be building it on a microprocessor or a specific microprocessor or I will be building it in a FPGA or on whatever. Sometimes it is not the designer's prerogative; it is the customer's prerogative. The customer can also specify that I want this particular embedded system this particular processor alright. So, that will serve as another input.

Then the system software where the real time operating system and all these things will play a role right? Now all these come to the design repository and the design process will involve mapping these applications, what does it mean by mapping these applications?

(Refer Slide Time: 10:37)

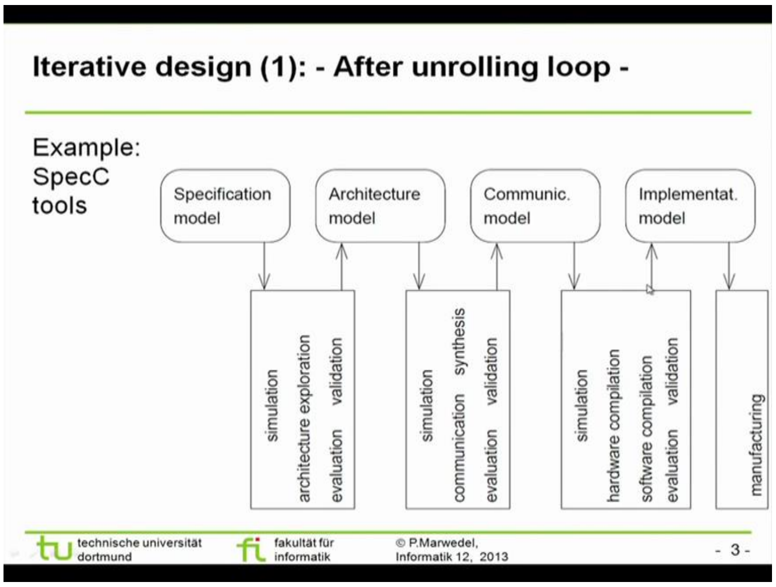


Now I have got some requirements; in the last class I mentioned that requirements and specifications will see how specifications can be stated. Now in the last class I stated I said that the entire ball game of embedded system design is in the lies in the tradeoff between what should be done in hardware and what should be done in software the hardware software tradeoff, which is also a key step in embedded system design, that is known as hardware software partitioning.

So, from here I can do the entire thing in software and that will make the system very flexible, quick, but will make it very slow also right I can make it entirely hardware, it will take more time to develop, it will be fast, but will be costly unless flexible. So, depending on my requirements I will have to decide on which part should be done map to which component. The components can be hardware component or software component, these are the different components right this components can be either hardware component or software component, I am not making a distinction between them. So, this which part should be done by what this is known as the application mapping.

Now, once I do the mapping then will have to do the optimization and these things often go hand in hand. I can try with different mappings it is basically turning out to be an optimization problem, where I can have I have got different options and I have got a set of constraints which I have to satisfy. So, it is a constant satisfaction problem as well as an optimization problem, where I have to optimize some functional objectives which may be the cost, the performance, whatever and then I have to evaluate and validate the system, which will be the energy. I will validate how much energy it is, what is the cost, what is the performance all through this after doing this I am getting the design and after that design I am doing the testing and this entire thing can be integrated into a look.

(Refer Slide Time: 13:14)



Iterative design (1): - After unrolling loop -

Example: SpecC tools

technische universität dortmund | fakultät für informatik | © P.Marwedel, Informatik 12, 2013 | - 3 -

For example if I unroll that look, it is another way of showing that earlier thing earlier thing could be in this way, now if I look at it in a different way what is happening? I am taking the specification.

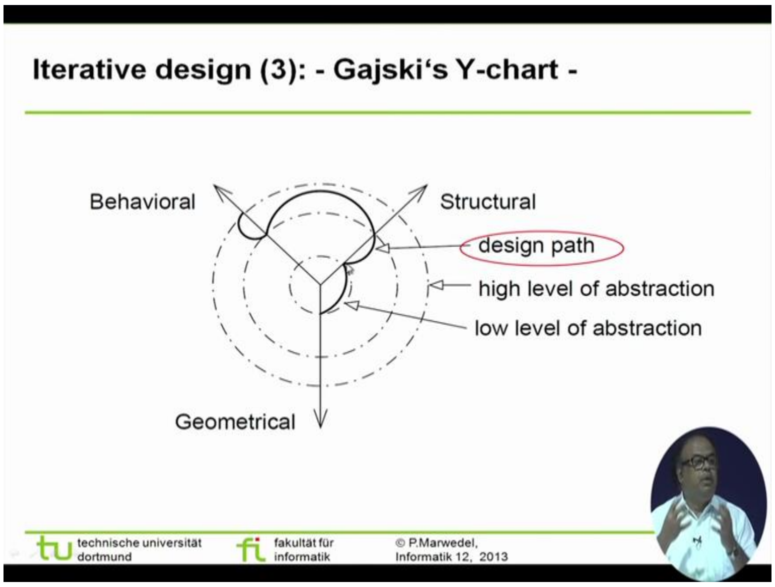
Now, here you see the specification can itself be simulated, whether the specification is meaningful, whether the specification is what I really want? Now very important thing is specification, because if I ask you to specify what you want, how do you specify? You can specify that well I need a temperature controller that will control the temperature of a oil bath between this and this now I can specify in English. Now when I specify in English or in natural language, there is a scope of lot of ambiguities. Many things may not be told, many things maybe missed out, many redundant things might be told. Now I have to have some way of evaluating whether the specification is itself complete or not whether the specification is complete or the whether the specification is what is expressing what I really intended to do, whether it is reflecting the design intent, so that I do.

So I see that, I can show some languages by which we can check the specification and then we perform architecture exploration, evaluation, validation then we will come to an architecture model. Again that architecture can be simulated, then if you remember the block diagram that we had shown last time, that we will have a processor, we will some IOs, we have to establish the communication between them.

Therefore, we have to design the communication system, we will have the communication model, and then after that after we get the schematic of the processes, hardware or software components and the communications, there can be different ways of communication after they do that then I can actually synthesize, I can synthesize the hardware, I can synthesize software. Synthesis of the software is known as the software compilation and hardware is known as hardware compilation or hardware synthesis and then we evaluate validate the whole thing again. You see every time we are doing in a loop, we are doing something validating, doing something validating and ultimately we can implement and then go for manufacturing.

So, this is the typical design flow and we will in this course we will try to address different parts of this these blocks.

(Refer Slide Time: 16:28)

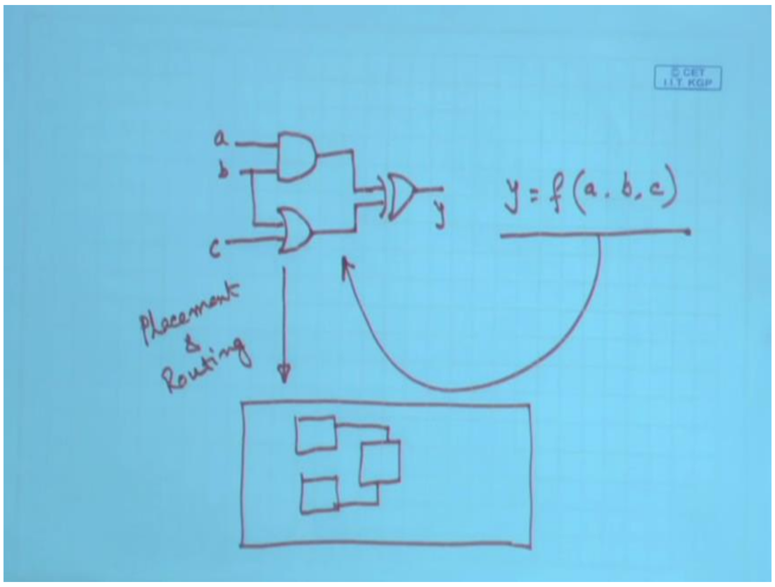

Iterative design (3): - Gajski‘s Y-chart -

Behavioral
Structural
design path
high level of abstraction
low level of abstraction
Geometrical
technische universität dortmund
fakultät für informatik
© P.Marwedel, Informatik 12, 2013

Now, what is synthesis essentially this is the famous diagram known as Gajskis Y-chart this is proposed by Daniel Gajski. If we look at in 3 dimensions, one is the behavior; what behavior I want my system to exhibit an adder. Adder got has got a behavior and what is the behavior it will take some inputs, now behavior may be taking 2 inputs adding them along with the carry and produce the output or the behavior could be it is just single bit adders, taking two bits and adding them, not caring about the carry whatever the behavior is?

Now, after that behavior when you synthesize, when we actually build an adder, what we come up with? We come up with a structure of gates all of you have done; digital logic or computer organization you know that. So, when I synthesize this adder full adder for example, or a multiplier, the specification has been given and that I synthesize and I come to a structure for example, adder will consists of some and gates, some exclusive or gates etc, but then that I get as a schematic of an adder. Immediately I cannot just take that a piece of paper with my drawings of gate and give to the manufacturer, for that I have to come to the geometry of the whole thing; that means, if I have got some components some functionality right.

(Refer Slide Time: 18:19)

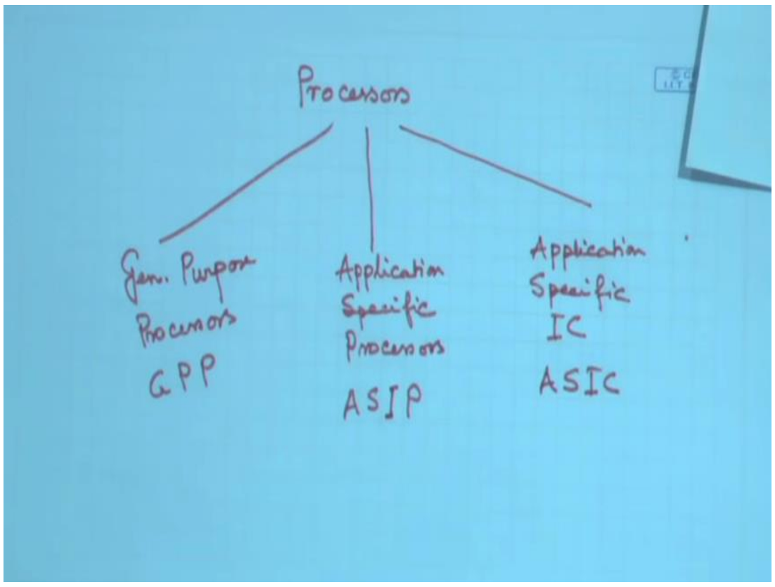


Some Boolean function I have implemented, some Boolean function was given I have implemented them. Suppose here some function I implement here, alright it is a b c, some logic function y of which is the logic function of a b c.

Now, this was my behavior, now from there I got the structure. Now that I need to now map this to a particular layout and placement that where the adder should, where the and gate should be, where the or gate should be, where the exclusive or gate should be and how the wire should be connected everything should be laid out that is the geometry, it is too simple a case. So, you do not find any complexity in this mapping, but if there are many components and the space you want to minimize, this would itself be a very important struck task, which is in the balance of VLSI design, that is placement and routing task. Where I place the cell, so place the objects and carry out the routing. So, these are the three components of any design and that is iterated in the way that we have just now discussed.

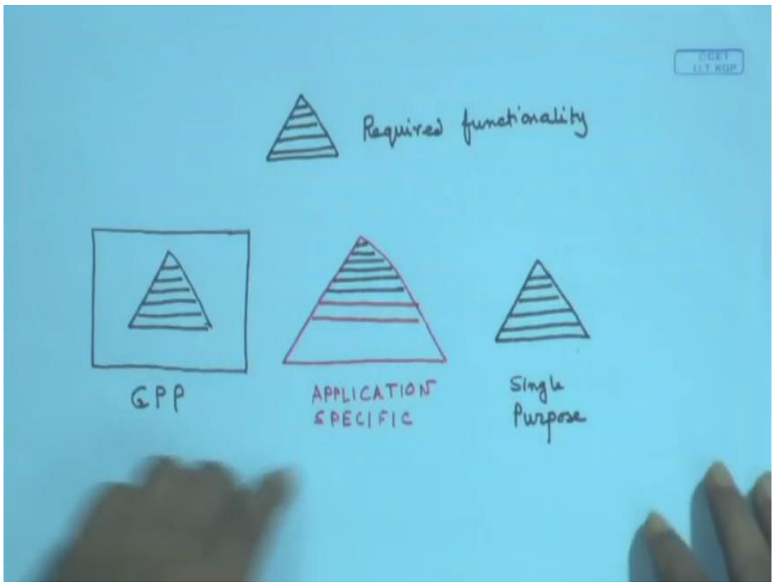
So, now we will come to our discussions on processors. Now let us come to our discussion on processors to start with. Processors are integrated and very important component of any embedded system by definition.

(Refer Slide Time: 20:51)



Now, this processors can be general purpose processor GPPs, or they can be application specific processors or ASIP application specific information processors, right that is why this I comes in or it can be absolutely an I C, which can be an application specific I C which is ASIC alright. It can come in either of these 3 flavors.

(Refer Slide Time: 22:07)

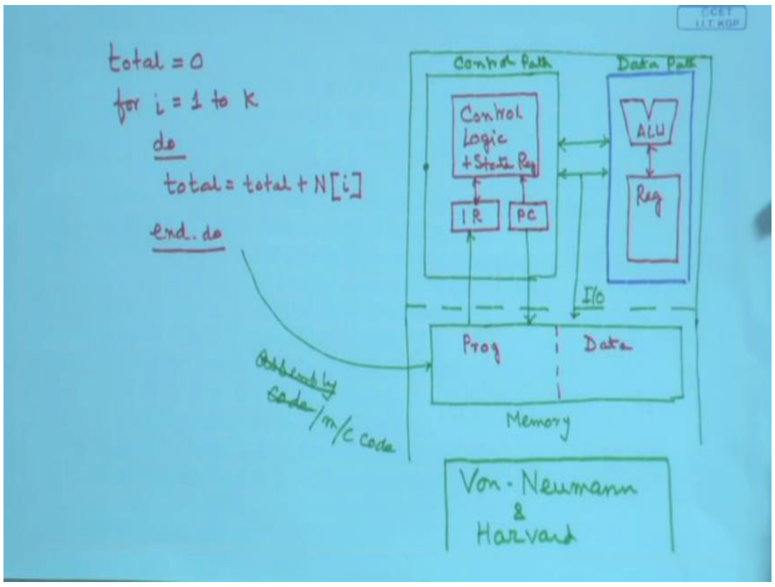


Now, what is the difference between these 3? Let us look at this diagram we have got a functionality say, I have to achieve this functionality alright. Now I have got a general purpose processor, which may be computer in your lab or maybe a microprocessor, which can do several things. Now this particular functionality can be mapped into this general purpose processor, can be mapped to this general purpose processors along with many other things. Now as the diagram shows in this way, we are not utilizing this processor to the fullest extent, because this person has been designed for general purpose, it can implement this functionality, triangular functionality or maybe a pentagonal functionality or an octagonal functionality, everything can be done by this.

Here it is an application specific processor, this processor has been designed it is just an analogue analogous application that I am talking of. As if this processor has been designed only for triangular functionalities say. So, it can accommodate this functionality, it can also accommodate some other triangular functionality is possible, but it cannot accommodate a rectangular functionality right. So, this is less general than this at right this is less general than this and this is on the other hand single purpose processor. It is purely designed for the required functionality only for this.

Now, how do they really differ, how are they differing, what is the difference between an application specific and a general purpose; will talk about this later, will come to this in a moment. But right now I think it is clear what is meant by a general purpose processor, and what is an application specific processor. So, let us now have a look at a particular function that I we want to achieve.

(Refer Slide Time: 25:00)



```
total = 0
for i = 1 to K
    do
        total = total + N[i]
    end.do
```

Now, let us look at this simple functionality, all of you can understand this quote total 0, for i 1 to K in a loop I do total, assign total class some from an array iteratively, and I want to implement this. What I have at my disposal is a general purpose processor.

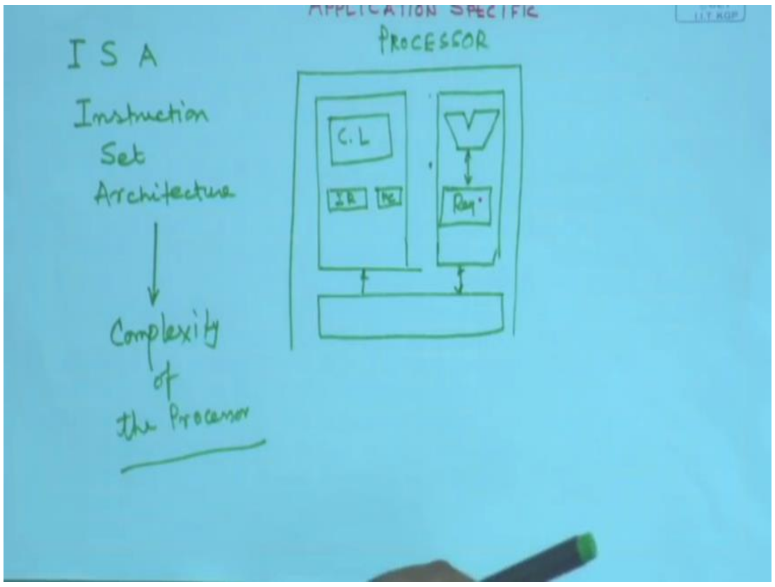
Now, all of you are familiar with the architectural of the computer, but here I want to reiterate that that because I want to introduce two terms, which I am not too sure whether you are familiar with. They are the control path and the data path of a processor. So, this part is green one is the control path or the controller part. So, I can say that this is a control path all the data flows the flows of the information here is a control path and this is the data path, we will see that in detail later. Now what is there in the controller? In the control path we have got a control logic all of you know that, there are some state registers, there is an instruction register, there is a program counter; right? On the other hand we on the data path we have got an ALU and a set of registers. All these things together make the CPU that we are familiar with we have got the memory.

Now, this code which is which I want to implement, you can implement in this general purpose processors, this is a pure general purpose processor right. So, here what I want to do is I load this, I actually should not be ultimately after compilation I can load the source code and do the compilation in this processor itself; ultimately I will get the machine code. So, let me just cut out this assembly code, ultimate the machine code is here and the execution goes on from here.

Now, here I am showing I have shown two partitions of the memory; the program and the data memory. But actually all of you know that in the computer system that we usually use the program and there is no distinction between program and data memory, but there are systems where the program memory and the data memory are separated. Accordingly we have got two types of architectures, do you recall that? One is the Von-Neumann architecture, where there is no distinction between program memory and data memory and the other is the Harvard architecture. So, often in DSP's we will find will encounter Harvard architectures, where the program memory and data memory are separate.

Now, coming back to this, this is a general purpose, now we can do many things right other than this particular code that I want to implement.

(Refer Slide Time: 29:02)


APPLICATION SPECIFIC PROCESSOR
I S A
Instruction Set Architecture
Complexity of the Processor
C.L
IR
PC
Reg

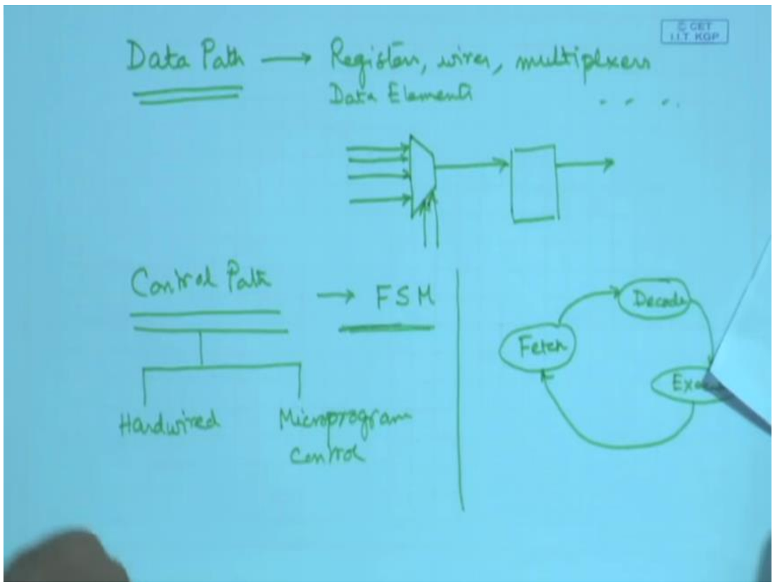
On the other hand; if I want to have an application specific processor, an application specific processor what would I have? I would have for the same code I would rather have this processor will be same, same thing there will be the control path, there will be the data path, in this there will be the ALU and there will be the registers and here I will have the control logic everything is remaining same, instruction registers and program counters right and they are connected to the memory same thing alright.

Now, what is the difference therefore, in this case? It is an I am saying that see this was a general purpose processor, and this one the next one that I am showing is an application specific processor, what is the difference between these two can you tell me what is the difference between these two? Here in this memory I will have the same code, what will be the difference between these two? I am saying it is an application specific; another one is a general purpose. I can I am showing that it has got the general purpose one has got a control path and data path, the application specific also has got the if I just put them side by side you will say more or less is the same thing right. So, where is the difference?

Now, we are coming to fundamental question, how what do you mean by an architecture, what defines the architecture of a processor? You might have heard the term instruction set architecture. Now again coming back to the fundamentals of computer system processor architecture a processor a see a general purpose processor give me a some examples of general purpose processor; like Pentium. Now Pentium has got a particular instruction set, that instruction set from that a instruction set we select the instructions to build an assembly code. The power of a processor is determined by the instructions that it can execute; now if I want to have a general purpose processor the instruction set will be much larger, but if I say that I will be doing only applications, where there will be no complicated functions, but simple addition, subtraction, multiplication and division different varieties of those, then I can have a simpler processor; what do you mean by a simpler processor?

So, instruction set architecture is actually leading to the complexity of the processor. Now what do I mean by the complexity of the processor? The complexity of the processor is actually coming from this complexity of this control path and data path, when I say see here I have drawn a very schematic simplistic diagram, but actually the data path; what is the data path, what is there inside the ALU and inside these registers or in this control logic, what is there what is there in the data path?

(Refer Slide Time: 34:28)



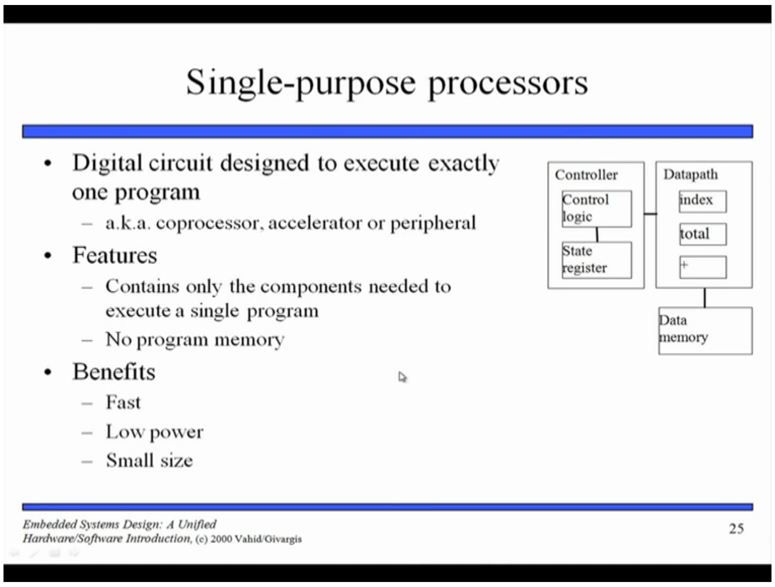
If you just think about that, the data path consists of registers, where’s of course, multiplexers, etcetera. So, our data is going and here there is a multiplexer, I am assuming that you have done the basic organizations. So, there are number of lines coming up, there some control lines here and one line is connected to another device maybe and there are data elements adder or subtracter, multiplier etcetera.

So, there are wires multiplexers and the different data elements. Now as I implement more and more complicated logic functions, this data path will become more and more complicated. On the other hand what is there in the control path? Control path is essentially a control machine which is basically a finite state machine, which is sending the control signals, which is sending the control signals to the different elements of the data path like the multiplexers and all those right. Therefore, as the data path becomes more complicated the control path will also become more complicated right and the control path again just brushing up your knowledge, it can be hardware controlled or micro program controlled different varieties are there and the any execution, any instruction execution goes to a common cycle known as the Fetch, Decode and Execute cycle.

So, an instruction is fetched from the memory through the program counter, and that instruction goes to the instruction register, it is decoded and then it is decided, what is to be done right and according the actions are taken. So, based on this who drives this cycle, the cycle is driven by this controller. Now as my instruction set becomes large I this data path will become complicated, this control path will also become much more complicated. Now if I want to have a application specific processor, for this type of code only then; obviously, the instruction set that is required will be much more limited therefore, the complexity of the data path and the control path will be even more limited right equally limited. So, that will be easier and will be, but it will not be as flexible as a general purpose processor; but in order to synthesize a general purpose processor is much more time consuming and costly right here we do not need to do that. So, we can have.

Now, why do I go for application specific processor, why do I go for a processor, what is the logic for doing this? I am reducing the complexity to some extent what else? The other thing is that as I did reduce this, I am still keeping it programmable it is still programmable I can put it the things in the memory, I can if it if the control path could be micro programmed I can also change it a little bit, I have got different options of flexibility. I am not getting rid of the flexibility completely.

(Refer Slide Time: 39:18)



Single-purpose processors

- Digital circuit designed to execute exactly one program
  - a.k.a. coprocessor, accelerator or peripheral
- Features
  - Contains only the components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size

Controller
Control logic
State register
Datapath
index
total
+
Data memory

On the other hand I come to this the third version that is a single purpose processor like this, single purpose processor if I want to do that, for the same functionality that I was showing you here that is this total this thing. If I want to make a single purpose processor that is it will do this and only this and nothing else, then how it look like? I can simply decide again a processor single purpose processor; where there will be some data memory, no program memory alright and I can have some controller, which will take the index value because what is here look at this thing there is an index value, and based on the index I am adding to the total right and there is a register, alright from where I am taking this.

So, it will be simply very simple data path, fixed for that program only and does data memory where this array N will be stored the array N that I am showing here, that is that will be stored here and that is all. So, what is the advantage it is fast, it with be slope low power and small size. So, in between comes up this is on one extreme and in between I get the application specific processor, and the other extreme we have got the general purpose processor.

So, when we design embedded systems, we have to be very judicious on deciding. Usually what we do you know, we quickly make the algorithm and test it on a general purpose processor, whether my algorithm is correct or not whether my function is correct or not then we start designing. Maybe we will first design it on a application specific processor so that I still have got some flexibility, then I move on to if required on a particular single purpose processor if required. So, we stop here today.