

Embedded System Design with ARM
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 02
Design Considerations of Embedded Systems

In our previous lecture, we talked about what an embedded system really is and what it is not. Now, in this lecture we shall be talking about some of the design considerations in embedded system design. Well, when we design an embedded systems, what are the things we should look at, and what are the design consideration and other tradeoffs that you should be aware of. These are the things that we shall be talking as part of this lecture.

(Refer Slide Time: 00:52)



So, we shall broadly be talking about the design challenges as I said, and we shall also be trying to understand some of the more important design tradeoffs. Design trade off means there can be some conflicting parameters you can try to improve one of the parameter, but it might degrade some other parameter. You cannot improve everything all together this is called trade off. And there are some cost metrics which are quite important non-recurring unit cost matrix. Well, we shall also be talking about these.

(Refer Slide Time: 01:31)

Design Challenges

- Primary design goal:
 - An implementation that realizes the desired functionality.
- The main design challenge is ...
 - To simultaneously optimize several design metrics.
 - Often mutually conflicting.
- What is a design metric?
 - Some feature of an implementation that can be measured and evaluated.

The Venn diagram shows three overlapping circles: 'Speed' (top), 'Quality' (bottom-left), and 'Cost' (bottom-right). The intersections are labeled A, B, and C. Checkmarks are present in the 'Speed' and 'Quality' circles, and a checkmark is in the 'Cost' circle.

Let us look at some issues relating to design challenges first. Now, when you are designing an embedded system, your primary design goal will be of course to design a system which realizes the desired functionality. Suppose, you are trying to build an embedded system for a washing machine, so your embedded system should be able to function properly in that environment. It should be able to control the washing machine in a proper way ok, so that as a user you would be happy with the performance.

So, desired functionality is the keyword, some implementation that realizes the design functionality. But in order to have this implementation, you may have some design challenges. Well, in the sense, there may be several design metrics we shall be talking about, and you may have to optimize several of them. Like well, user may say I want something which is low cost, small in size, very powerful and also very rugged. But you see if you try to reduce the cost, naturally your performance will also go down, your ruggedness can also be compromised. So, you cannot have best of all worlds; this is something we refer to as tradeoff. These are mutually conflicting in most cases.

Now, this design metric whatever we say it is some feature of the implementation. We shall be talking about some of them which can be quantified that means you can measure them in some way. Like, for example, I say cost well cost is easy to measure I can say this much rupees or this much dollars that is a method of quantifying the cost. Similarly,

this measurement and evaluation you can compare two alternatives, you can say which one is better than the other.

You see now here I have given a small example where these three circles refer to three parameters or design matrix speed, cost and quality. Well, in most designs, we want to improve on all these three, but as I had said these three are often mutually conflicting. Well, if you try to reduce the cost you will be sacrificing on the speed and quality and so on ok, alright.

(Refer Slide Time: 04:26)

The slide is titled "Common Design Metrics" and lists the following metrics:

- **Non Recurring Engineering (NRE) Cost:** One-time initial cost of designing a system.
- **Unit Cost:** The cost of manufacturing each copy of the system, without counting the NRE cost.
- **Size:** The actual physical space occupied by the system.
- **Performance:** This is measured in terms of the time taken or throughput.
- **Power:** The amount of (battery) power consumed by the system.
- **Flexibility:** The ability to change the functionality of the system.

The slide also features a video feed of a presenter in the bottom right corner and a logo for "swayam" in the bottom left corner.

Now, let us talk about some of the common design metrics that we talk about with respect to an embedded system design ok. We are concentrating ourselves to the design of embedded systems only. First important parameter is called non recurring engineering cost. Well, non recurring engineering cost means this is some kind of initial cost.

Like for example, let us take an example. Suppose you have an idea design idea. Now, you are going to manufacture those systems. But before you start the manufacturing process, you will have to install some machines in your factory which will help you in the manufacturing. And that initial installation will incur some initial cost; this is what is referred to as non recurring expenditure or engineering cost. This will be required only once at the beginning. Now, once we have that infrastructure ready, so you can manufacture your units the systems as many you want you can manufacture 10, 20, 100, 1000 as many you want ok. So, non recurring cost is the one time initial cost.

And once you have this NRE cost covered, you talk about unit cost. So, after we have that infrastructure, what is the cost of manufacturing for every copy of the system that you define as the unit cost ok. Now, unit cost will also consider the cost of the raw materials, labor cost and so on and so forth right. Size is an important parameter I talked earlier. Most of the embedded systems need to conform to a very small form factor, so that it can fit nicely inside the environment for which it is meant ok.

Performance well again this depends on the application. For some application you are may not be that much aware about the performance, you need very little computational capability, but there are some applications where performance is important. You need to ensure that performance is assured. Power consumption, this is important. Most of the systems we see today, they run on battery. They need to consume very low power. Well, our mobile phone is a very classic example of an embedded system in that respect ok.

Flexibility, flexibility means flexibility says the ability to change the functionality of the system. The system was initially designed for certain application. So, is it possible to modify it, so that you can also use it for some other application. Well if that flexibility is there, then possibly for the second development your total cost would be much less, designed cost ok. So, flexibility is also an issue that needs to be considered.

(Refer Slide Time: 07:57)

- ✓ **Maintainability:** How easy or difficult it is to modify the design of the system?
- ✓ **Time-to-prototype:** How much time is required to build a working version of the system (i.e. a prototype)?
- ✓ **Time-to-market:** How much time is required to develop a system such that it can be released to the market commercially?
- ✓ **Safety:** Are there any adverse effects on the operating environment?

• Can be many more ...

Maintainability, maintainability says that well I have already purchased an embedded system as part of a larger system. Let us say I have purchased an air conditioning

machine. There is an there is an embedded system inside. Tomorrow the company says that well we have come up with a better AC machine which has a Bluetooth interface, which can interface with your mobile phones and computers and laptops. So, you can control the AC machine even from your mobile phone, even from your laptops by installing some apps.

So, is it possible for the older system to adapt to this new technology? Can you modify this design so that this added functionality can be incorporated. This is what we mean by maintainability. But of course, maintainability is possible only to a limited extent. Just for the example I sited for having Bluetooth you need to have a Bluetooth interface right. So, if it is not there in the first place, you cannot have Bluetooth ok.

Then time to prototype. How much time is required to build the first working version of the system that is called a prototype. How much time is required to have the first version ready so that you can test. And after the prototyping is done, time-to-market. How much time it is required to build a finished product which can be sold in the market? Well, prototyping and finished products are two entirely different things. Finished products have to have durability, finishing, quality everything in place right. And of course, safety is an important issue for many applications. You will have to be sure whether there are any adverse effects on the environment or not because of use of that system ok. And there can be many more such things I have only listed a few of them right ok.

(Refer Slide Time: 10:15)

Design Tradeoff

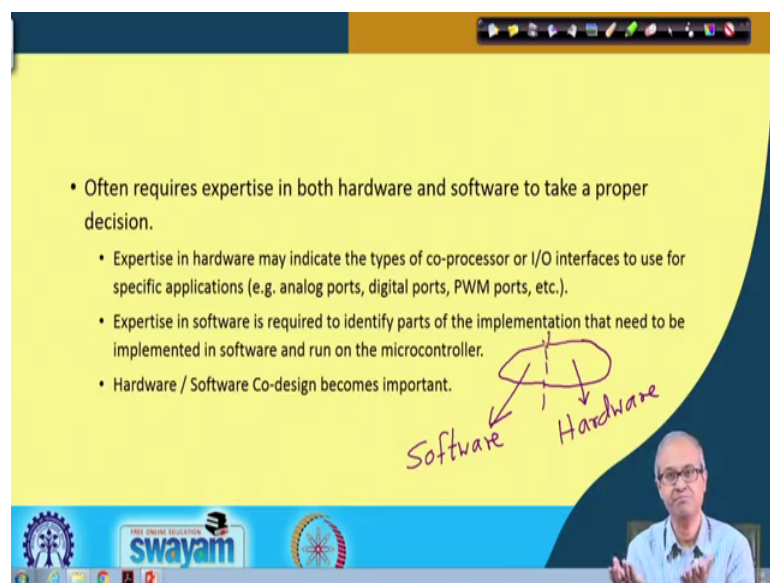
- Many of the design metrics can be mutually conflicting.
 - Improving one may degrade the other (e.g. power, performance, size and NRE cost, etc.).

The diagram illustrates the tradeoff between four design metrics: Power, Performance, Size, and NRE cost. A central white circle is connected to four surrounding circles. Arrows indicate that improving one metric leads to a degradation in another: Power (top) has an upward arrow; Performance (left) has a leftward arrow; Size (right) has a rightward arrow; and NRE cost (bottom) has a downward arrow. The slide also features a video feed of a presenter in the bottom right corner and logos for 'swayam' and 'THE OPEN UNIVERSITY' at the bottom.

Talking about design tradeoffs, this we have already mentioned earlier that there can be several design requirements which are mutually conflicting. Here in this diagram, I am showing four such matrix, power consumption, performance, non recurring expenditure, and size. Now, these four can be mutually conflicting if you pull one the others will be pulled in the reverse direction, means if you want to improve one maybe the others might get degraded. So, you will have to take a holistic view. This is the task of the designer. You cannot improve everything at once.

Like for example, the very slim laptops that are being built. Well you do not expect that those laptops will become powerful in all respects, no, to make it slim somewhere there is some compromise which has been made. These are examples of design tradeoffs ok. So, this is something which you have to keep in mind.

(Refer Slide Time: 11:32)



- Often requires expertise in both hardware and software to take a proper decision.
 - Expertise in hardware may indicate the types of co-processor or I/O interfaces to use for specific applications (e.g. analog ports, digital ports, PWM ports, etc.).
 - Expertise in software is required to identify parts of the implementation that need to be implemented in software and run on the microcontroller.
 - Hardware / Software Co-design becomes important.

And another important thing is that talking about design tradeoffs. Well, for embedded systems you need a different kind of trained professionals. Well, you think about the conventional computing days or traditional computing environment, where you have a set of engineers, who developed the hardware who developed the processes like Pentium processors is developed by in developed by Intel by a set of highly qualified hardware engineers. There are some other groups who develop software for those computer system that use those chips.

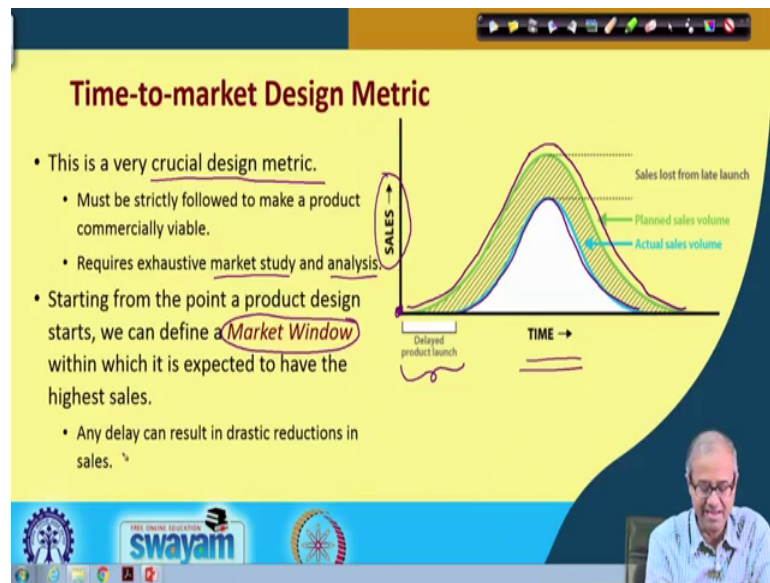
For example, in Microsoft there are some software engineers who developed the operating system, who develop utilities like Microsoft Office and so on. So, there are hardware experts, there are software experts. Their role is more or less independent of each other. The hardware expert need not know software very well and software experts need only know very little about the hardware. But now when you are designing an embedded system, you are talking about a very small system where both hardware and software will be there. And talking about the design trade off, you will have to play with both of them.

So, now you need to have the expertise of knowing the hardware, how to use it and also software how to program it in an efficient way. So, both hardware and software expertise are required this is what I wanted to say here. Now, you may need expertise in harder to identify what kind of coprocessors or IO interfaces you will be requiring for a certain applications, because you may have several choices. There can be some analog ports, digital ports, pulse width modulation ports, but which of them will be best for a certain application? So, unless you are an hardware expert, you really cannot take a good decision here ok.

Similarly, you also need to have expertise in software, because you need to decide which parts of the implementation you need to implement in software, and which part is already implemented by some hardware interface or can be implemented by some hardware interface. So, you talk now about something called hardware software co design means you are designing both hardware and software subsystems together. Like it is something like this, you have a large system to be designed.

Now, you will have to decide that well I make a demarcation, well one of that part will be implemented in software, this will be running as a program on a microcontroller and the other part will be implemented by some specialized hardware circuit. So, where this boundary will be this is a matter of design you can say trade off, the designer will decide how to shift this dotted line, so that desired performance and other criteria are satisfied in the best possible way all right.

(Refer Slide Time: 15:29)



Now, there is an important thing here, time to market design metric. Now, let us try to explain this, what this mean. See a company whenever it manufactures some products the ultimate goal will be to generate some profit out of it. So, marketing is the most important issue for any company. The quality of the product becomes secondary. Sometimes many companies they compromise on the quality of product in order to bring the product to the market earlier, so that they can reap greater profit out of it. Time to market is a very important or crucial design metric.

As it said must be strictly followed to make a product commercially viable. If you delay in the launch of a product, you may incur a big loss. Maybe some of your competitors already have hit the market with a similar product, and people will buy naturally from them and not from you ok. And this requires very careful market study and analysis. Now, this we are illustrating with a diagram like this as I showed here. Here the x-axis shows the time and y-axis shows the sales typically.

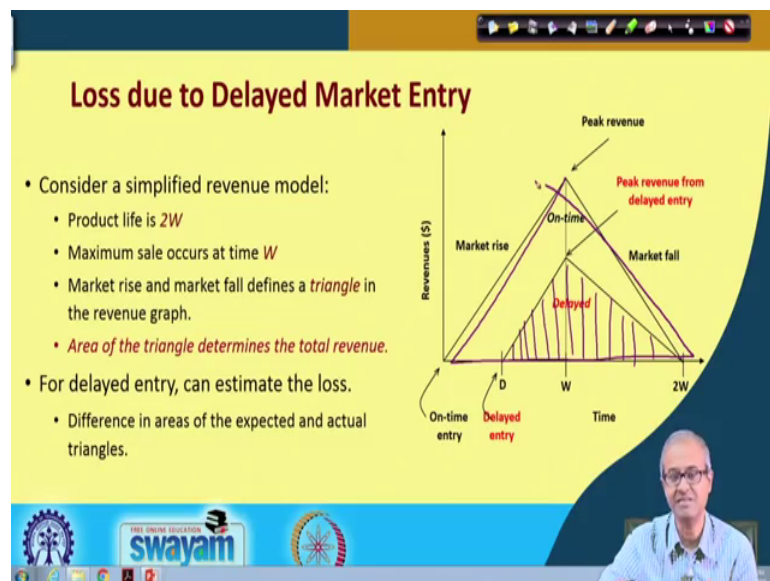
Suppose your origin indicates the point where you are expected or you were expected to bring the product in the market. So, what happens when a product comes to the market, well the users or the customers also need some time to learn about the product. So, the sale increases slowly over time, like you see it increases slowly over time this green line. And there will be a point where this sale will saturate and beyond that point again this

sail will start dropping down. And there will be a point where the product will become obsolete no one is buying it anymore. This is a typical curve, sale versus time.

And suppose there is a delay in the product launch. So, instead of this curve, you are now following this inner curve. There was an initial delay. So, you see the curve will be similar again, but the peak the maximum sale can that you can achieve is becoming much less. The total area under this curve will denote, your total revenue, how much total sale you have been able to done sales versus time. The total area under the curve will indicate your total sales over that period of time. So, this curve area under the curve is defined as the market window. So, any delay can result in a drastic reduction in sales ok.

Now, let us make a simple calculation with some approximation. Well the curve looks like this, but to a first approximation we can assume that these are straight lines, you can assume that this is straight line, this is also a straight line.

(Refer Slide Time: 19:10)



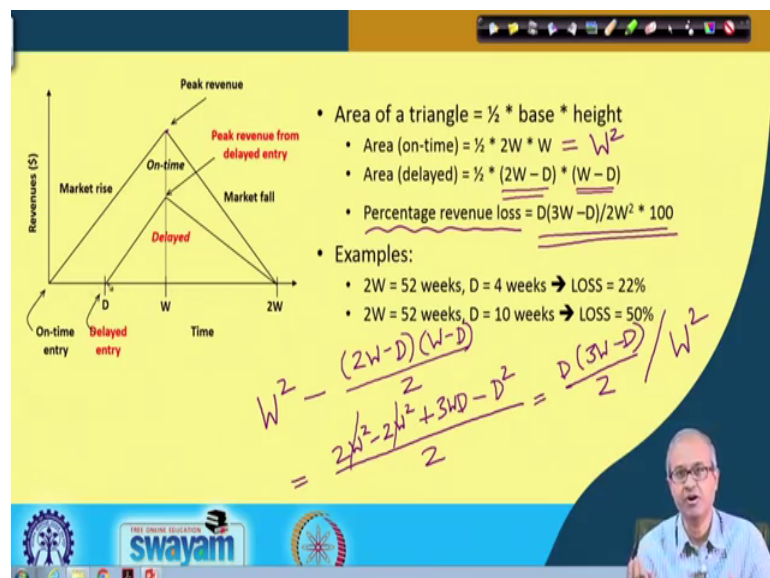
So, with that assumption let us try to estimate that how much loss were expected to incur if there is a delay in entering the market. We considered a simplified revenue model, you can modify it very easily to means whatever is more realistic in your specific case. Here what we assume is that as this graph shows that the product life is $2W$. So, from 0 up to $2W$, this is your total product life. After $2W$, no one will be buying your product. And let us assume that the maximum sell occurs in the middle of it, at point W the cell

becomes maximum that means, this point. Market rise and market fall as I had said we approximate it by straight line that means we define a triangle a triangular structure.

And as I mentioned the total area under this triangle will determine the total revenue that you can generate, the total sales ok. Now, if we have a triangle you know how to measure the area of the triangle it becomes easier ok. Now, for delayed entry as I had said in the previous diagram also, suppose there is a delay in the entry by this much. So, now, your triangle starts from here at this point D, but it will again reach the peak at the same point because depending on market dynamics beyond a certain point interest in that product will starting will start to go down.

So, the fall will start happening at the same point W. So, from then again it will continue up to 2 W beyond which there will be no interest in that product anymore, but your initial delay is actually shifting the left edge or the left vertex of your triangle to the right, this is what is happening right. So, there will be a difference in the area of the expected and actual triangles. Like your actual triangle area will be only this much right, but your expected area was the area of this whole triangle larger triangle. So, it is naturally much less in this case because of the delayed launch ok.

(Refer Slide Time: 21:46)



Let us make a calculation. Well, we show that same plot on the left. Well, we know that the area of a triangle is defined as half into base into height, half base into altitude. So, for the on time case, that means, the larger triangle area will be half into base is 2 W, and

height is W let us assume this is an equilateral triangle, it is rising at 45 degree slope, let us assume. So, height let us assume this is also W the height is also W so, $2W$ into W . So, this becomes how much W square, W square.

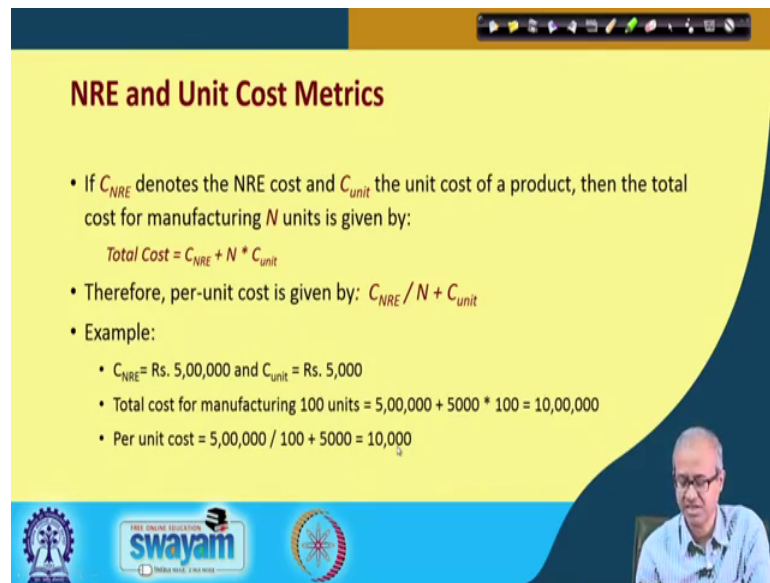
Now, for the delayed case your base becomes $2W$ minus D , this becomes $2W$ minus D . And your height also because of this delay this will become W minus D there will be a delay. So, this calculation can be done this is parallel to it. So, there will be a decrease by this factor. Now, if we want to estimate what is the revenue loss because of this you see initially it was W square right, but in the modified version it becomes $2W$ minus D multiplied by W minus D divided by 2.

So, if you just calculate how much is this it will be twice W square. So, if you make this calculation to be twice W square, then minus $2WD$ and $1WD$, it will become plus $3WD$, and D square, there will be a minus D square. So, these two will be cancelling out. So, what you get is if you take D common, $3W$ minus D by 2, this is your decrease in the total revenue.

But if you want to calculate the percentage revenue loss you will have to divide it by your original total revenue which was W square. So, multiplied by 100, this is your percentage revenue loss, that means, revenue loss divided by your initial total revenue multiplied by 100 right. So, in this way you can make a simple calculation and see that for various values of D how much revenue loss you are incurring.

Just a simple calculation as an example, suppose your window size is 1 year - 52 weeks 2 years sorry 52 weeks is 1 year. And your if your delay is 4 weeks, let us say you are 1 month delayed if you substitute these values here your loss becomes 22 percent. But if you are delayed by 10 weeks, your loss becomes as large as 50 percent ok. So, you see that your loss increases very rapidly, and this delay becomes very costly for the financial viability of a product right. This is a very important consideration.

(Refer Slide Time: 25:28)



NRE and Unit Cost Metrics

- If C_{NRE} denotes the NRE cost and C_{unit} the unit cost of a product, then the total cost for manufacturing N units is given by:
$$Total\ Cost = C_{NRE} + N * C_{unit}$$
- Therefore, per-unit cost is given by: $C_{NRE} / N + C_{unit}$
- Example:
 - $C_{NRE} = Rs. 5,00,000$ and $C_{unit} = Rs. 5,000$
 - Total cost for manufacturing 100 units = $5,00,000 + 5000 * 100 = 10,00,000$
 - Per unit cost = $5,00,000 / 100 + 5000 = 10,000$

Now, talking about the NRE Non Recurring Engineering and unit cost well if C_{NRE} let us say denotes the non recurring engineering cost, and C_{unit} denotes the unit cost. Then for manufacturing N units of the product the total cost will be non recurring cost this is this is a onetime cost plus number of product multiplied by unit cost. Now, if you try to calculate how much cost effectively means we have incurred for every unit, that means, per unit cost, you divide the total cost by the number of you divided by N . So, it becomes C_{NRE} divided by N plus unit this is your per unit cost right.

So, an example if your non recurring engineering cost is rupees 5 lakhs let us say and unit cost is 5000, then for manufacturing 100 units the total cost if you calculate using this formula, it becomes rupees 10 lakhs. So, per unit cost will be C_{NRE} divided by N plus C_{unit} C_{NRE} divided by N plus C_{unit} 10000, like this you can calculate right.

(Refer Slide Time: 27:04)

Per-unit cost = $C_{NRE} / N + C_{unit}$

- We can compare technologies by cost:
 - Choice A: $C_{NRE} = \text{Rs. } 20,000$, $C_{unit} = \text{Rs. } 8,000$ ✓
 - Choice B: $C_{NRE} = \text{Rs. } 4,00,000$, $C_{unit} = \text{Rs. } 3,000$ ✓
 - Choice C: $C_{NRE} = \text{Rs. } 10,00,000$, $C_{unit} = \text{Rs. } 8,000$ ✓
- Of course, time-to-market cost must also be considered.

$N = 1000$

THE ONLINE EDUCATION swamyam

Just a very simple calculation I am just illustrating. Suppose you have several choices like to manufacture a product you see that there are several different kind of equipments you can purchase to manufacture them. And the initial cost of the equipments are also quite different. Suppose you have three choices for the first choice the non-recurring costs is 20000, second one 4 lakhs, third one 20 lakhs. And the unit cost for the three cases are coming to this, this and this.

So, what do you can do you can calculate the per unit cost for some particular value of n well I leave it as an exercise for you. Suppose I tell I want to manufacture 1000 units of product, then using this formula you calculate the per unit cost for choice A, choice B, and choice C and C which one of them is better. Not only that you will also have to consider the time to market cost, because may be means one choice is good in terms of cost, but you are taking a long time to start that machine, because installing, training and just using that machine it is requiring much more time that also you also have to need to consider. That not only the cost, but also the time to market ok.

(Refer Slide Time: 28:46)

Performance Design Metric

- Most widely used, but can also be most misleading.
 - Must be careful in the evaluation.
- Some of the measures:
 - Clock frequency (MIPS) -- not very good for comparison
 - Latency (response time)
 - Throughput
- Measure of speedup among design alternatives.

And in addition you have some performance design metrics that also you need to consider. Normally, these are traditionally used for evaluating processor performances CPU processors. So, the same consideration you can also use or try to use for an embedded system, but there are a few things you need to remember. This performance design matrix tell you that which processor is faster than the other in some respect. These are quite widely used, but it is also true that if you do not understand the meaning of these metrics they can be the most misleading.

Most of the computer manufacturers they try to mislead the customers by quoting some figures with respect to benchmarking and speeds which if you dig deeper and try to understand, you will actually feel that well for your particular environment those numbers make no sense. You may require to know something else, which will be best suited for your application ok. So, you must be very careful in the evaluation this is what I wanted to say.

Some of the typical measures which are used again these are not good measures. Clock frequency, well you see someone is saying 1 gigahertz, 2 gigahertz, 3 gigahertz does faster clock frequency means a faster computer not necessarily. There are many other things which determine the actual performance of a computer system ok. There are some measures like million instructions per second – MIPS; this also does not mean anything. What kind of instructions are they simple instructions or complex instructions. Your

MIPS figure will vary drastically among them. So, it does not give any fair measure. Only with respect to some specific application, you should be able to measure that how much time it is taking to run my application that is something which would be most you can say beneficial in your assessment ok.

Latency response time you give an input after how much time you are getting back the output this can be of course, one good measure for many real time applications. Throughput, throughput again may not be that good for an embedded system. Throughput means number of computations that can be carried out per unit time. Normally, we talk about throughput for conventional computer systems, but for an embedded system which is meant for a very specific application, we normally do not talk about throughput.

And if you have several design alternatives, you should have a way to compare their speeds, speeds of operations ok. So, let me repeat these assessments are not easy. The best way is to run the application you want to run on a real computing machine and see how much time it takes. Whatever the manufacturer says you take them with a pinch of salt, do not trust them 100 percent ok, all right.

So, with this we come to the end of this lecture where we talked about some of the design metrics that are relevant in embedded system design and some of the implications with respect to the cost and revenue.

Thank you.