

**Principles of Engineering System Design**  
**Dr. T. Asokan**  
**Department of Engineering Design**  
**Indian Institute of Technology, Madras**

**Lecture – 13**  
**Implementing Fault Tolerance in Physical Architecture**

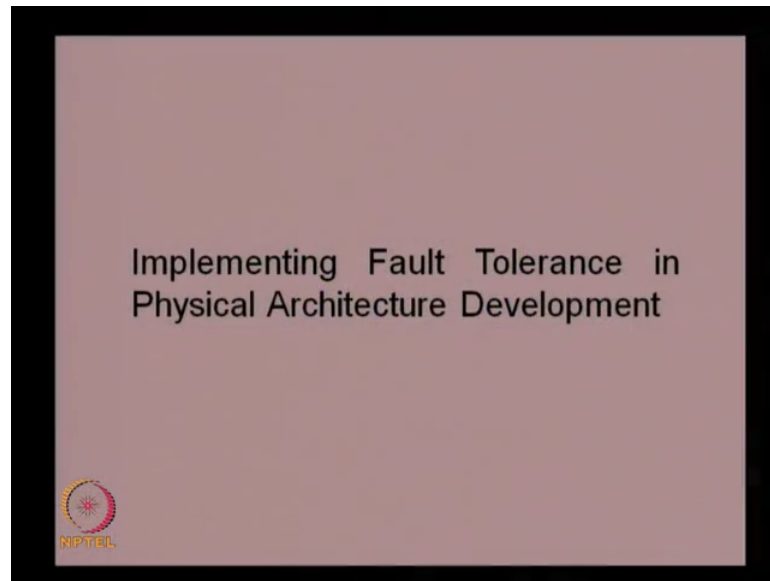
Hello friends, welcome back to another session on system design. In the last lecture we discussed about the physical architecture development for engineering systems. We discussed about how do we convert a functional architecture to a physical architecture basically we develop a generic physical architecture, where the functional elements will be converted to generic elements or there will be the generic elements corresponding to the functions where we will be identified, and a architecture will be developed a hierarchical generic architecture will be developed.

And from there we will go for a alternative options for this generic elements. So, we use a morphological box a method of morphological box to identify the alternatives and using this alternatives, we will develop an instantiated architecture, where the corresponding elements of the function will be convert to physical elements and the details of these physical elements will be specified in terms of the components their mic and their other specifications. So, when we do this we are actually getting a an instantiated physical architecture or probably we will be getting a multiple architectures based on the components we select, and based on an exit criteria we choose the a final architecture for the product or the system.

So, that is what we saw in the previous lecture and as I mentioned in the that lecture we need to incorporate few a components to make sure that the system is having a sufficient fault tolerance, that is we need to identify some errors in the system or whenever a some error develops in the system we should be able to identify these errors as well as we need to identify the source of error, we have to confined the damage to the system and make sure that the system continuously works without any problem. So, in addition to the normal functions identified through the customer requirements, we need to provide these functions and corresponding physical elements also.

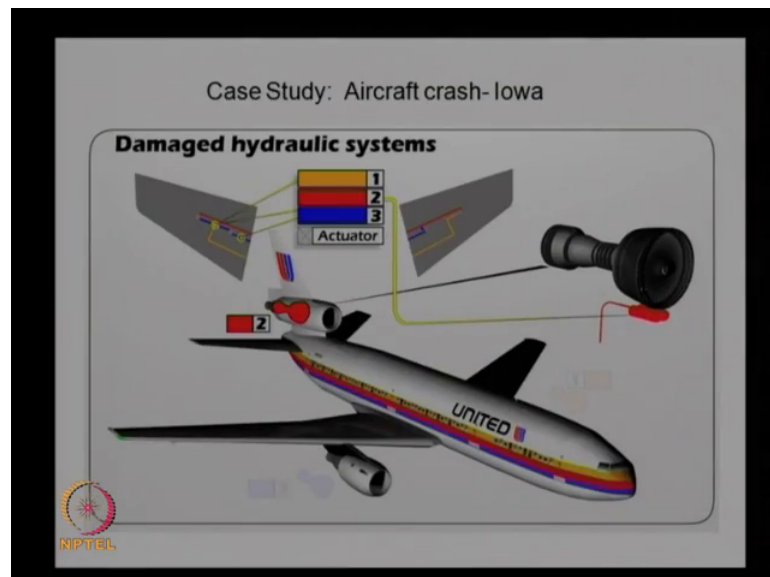
So, in this lecture we will see how do we actually implement these functions through physical elements in terms of fault tolerant elements.

(Refer Slide Time: 02:22)



So, we will look at the implementation of fault tolerance in physical architecture, what are the methods by which we can do this and what are the a procedures and different processes involved in implementing the fault tolerance. So, the importance of fault tolerance we discussed in the a last class and we mention that.

(Refer Slide Time: 02:39)

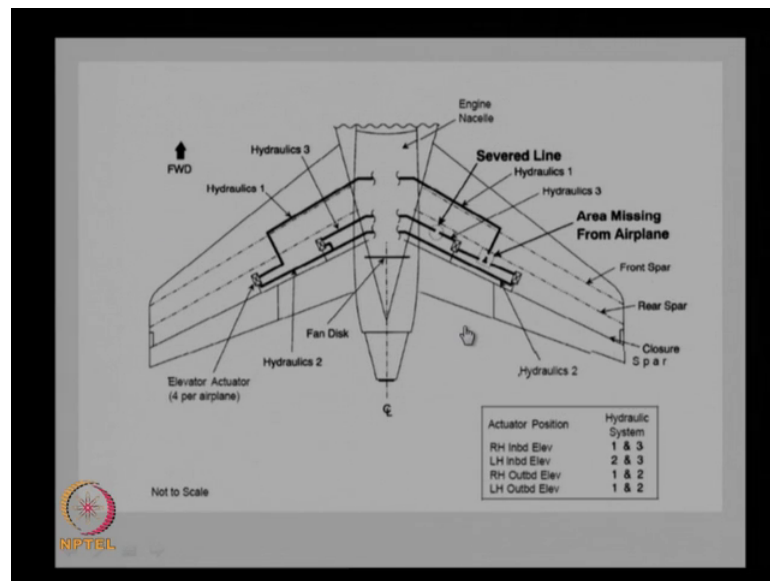


Some of the case studies we showed that the failure of the aircraft united aircraft 232 was basically because of the failure in having a fault tolerance systems or failure in having a

fault tolerance or the position of a single point failure in the system, which actually cause the failure.

So, as we can see in this picture the hydraulic system failed and there was a single point where actually the hydraulic points converge, and it was that this convergence point was the fan disk heat the tyrant and then the damage the hydraulic system to the control planes and the flight lost its control and finally, crashed.


(Refer Slide Time: 03:19)



This is the hydraulics diagram which actually shows that this point the part was missing because this was a point where actually the hydraulic supplied to the control planes on this view are provided, and since it was damaged at this point a single point the system failed and it could not really control the plane control was made impossible because of that failure.

So, we need to make sure that such single point failures are not there in the system, as well as we have enough a redundancy in the system to make sure that even if one system fails the other systems are there to take care of the system functionalities. In the physical architecture development we will look at these aspects and then develop sufficient redundancy in the system in terms of physical elements or in terms of software algorithms, to ensure that we have the redundancy to overcome any kind of emergency scenarios.

(Refer Slide Time: 04:17)



- ❑ United 232: 3-engine aircraft crashed on 19/7/1989 while making an emergency landing after losing one of the three engines. 110 people died, 185 survived.
- ❑ Three redundant hydraulic systems, each powered by a unique engine, were available for aircraft stabilisation.
- ❑ The three hydraulic system converged at the location near the tail where the fan disk ripped out, the single point of failure for all the hydraulic systems.

So, this is the case study which I have explained that basically aircraft with the 3 engines and 3 separate hydraulic system, which actually failed because of the a single point failure.

(Refer Slide Time: 04:33)

### Error detection Functions

**Failure:** Deviation in behavior between the system and its requirements

**Error:** A subset of the system state, which may lead to system failure.


**Fault:** a defect in the system that can cause an error.

Fault tolerance is the ability of a system to tolerate faults and continue performing.

- Fault tolerance can be achieved only for those errors that are observed.

Functions associated with fault tolerance are:

- Error detection
- Damage confinement
- Error recovery
- Fault isolation and reporting



So, in order to avoid this one we need to provide the many functionalities basically the fault tolerant system, and we need to have many error detection functions which we discussed in the functional architecture development. Just to recap those terminologies

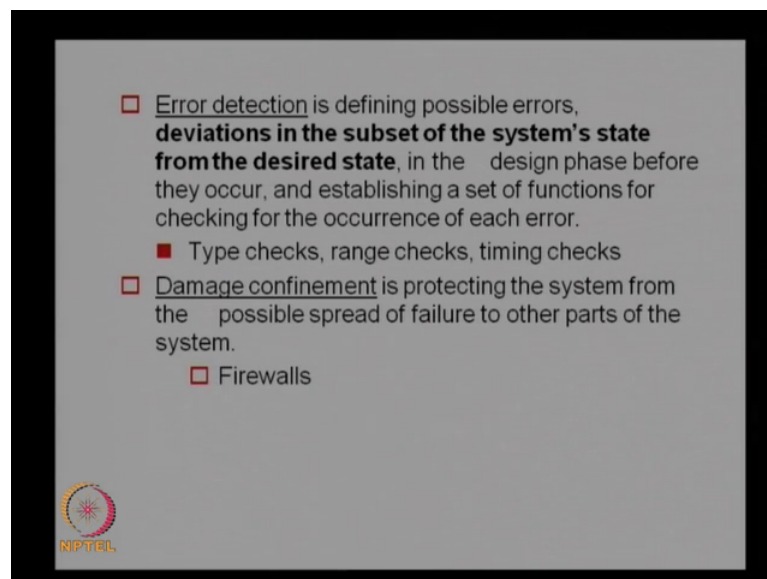
we have this failure which is the deviational behavior between the system and its requirements.

And then we have this error, which is a subset of the system stage which may lead to the system failure and a fault which is a defect in the system that can cause an error. And in the fault tolerance basically we will be looking at the ability of a system, to tolerate faults; and continue performing. So, whenever there is a fault the system should be able to perform continuously, without having the problem or without affecting the performance of the system and that is the fault tolerance in the system.

And this can be achieved over those errors which can be observed we can provide the fault tolerance and which are not observable cannot be tolerated, because the system wont to able to identify those errors and hence cannot have a tolerance for those errors and the functions associated with the fault tolerance are basically the error detection, a damage confinement error recovery and fault isolation. These are the four functions which we need to provide and we need to have physical elements to provide these function that is the detection of error.

Then confinement of the damage is created by the error and then recovering from the error as well as isolation of that error and reporting about that particular error.

(Refer Slide Time: 06:05)

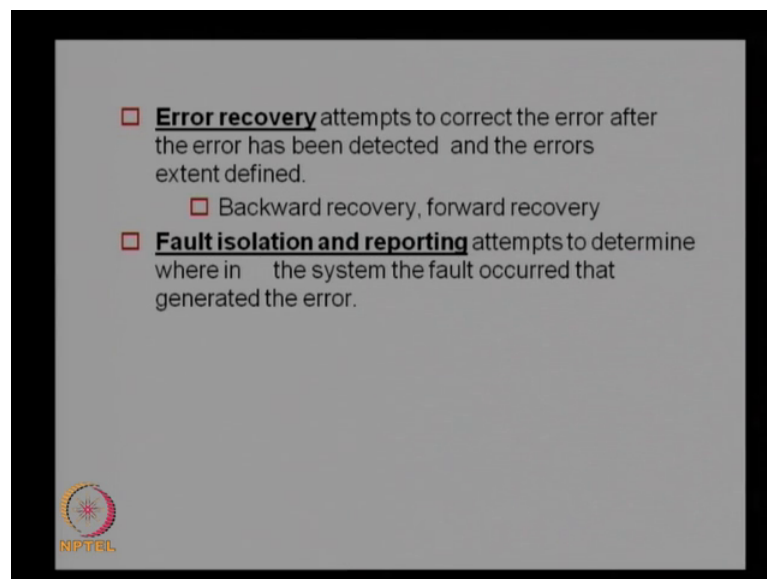


- Error detection is defining possible errors, **deviations in the subset of the system's state from the desired state**, in the design phase before they occur, and establishing a set of functions for checking for the occurrence of each error.
  - Type checks, range checks, timing checks
- Damage confinement is protecting the system from the possible spread of failure to other parts of the system.
  - Firewalls

So, error detection is basically defining the possible errors, which are the deviations in the subset of the system state from the desired state; the designed phase before they occur and establishing a set of functions for checking for the occurrence of each error. So, in error detection we will provide some kind of functions, where it will continuously monitor for the performance and then we keep a track of those performance based on a set value, and then report if there is any deviation then it will be reported as an error.

So, the normal error detection functions are basically the type checks, a range checks and a timing checks. We will be see how this can be implemented at a later stage and damage confinement is protecting the system from the possible spread of failure to other parts of the system. So, as the name suggests damage confinement basically you confine the damage and protect the other system from damage, that is a damage confinement normal way of implementing it is using a firewalls especially in software installation. So, we can see that firewalls should be provided to protect the system from damage from other systems.

(Refer Slide Time: 07:09)

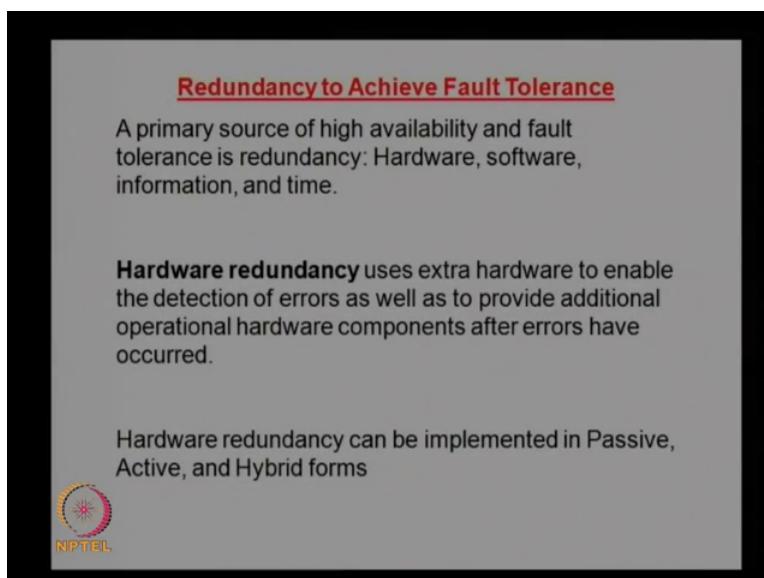


And error recovery attempts to a correct the error, after the error has been detected and the errors extent defined. So, once the error detected and the extend defined and recovery is basically you can get back to the normal modes of operation even if there is an error. So, this can actually be attempted in 2 ways; one is the backward recovery the other one

is the forward recovery. We discussed about these methods in the functional decomposition or the functional development stage.

So, I am not go into the details of backward recovery and forward recovery, but these are the 2 methods by which we can implement the error recovery. And the last one is a fault isolation and reporting which attempts to determine wherein the system the fault occurred that generated the error. So, it look for the fault in the system, and then see from where that error occurred and accordingly you can isolate that particular fault. So, that is the fault isolation and reporting.

(Refer Slide Time: 08:09)



Now, how do we actually implement this fault tolerance? So, that is the in physical architecture development that is a important question how do we actually implement the fault tolerance using the physical elements. The

primary source of high availability and fault tolerance is redundancy. So, in any system the primary source is redundancy. If you can provide a redundant components in the system that actually ensures to a great extent the fault tolerance of the system, and this can actually be done using hardware, software, information and time. So, you can have hardware redundancy, you can have information redundancy, you can have software redundancy and you can have time redundancy also on the system.

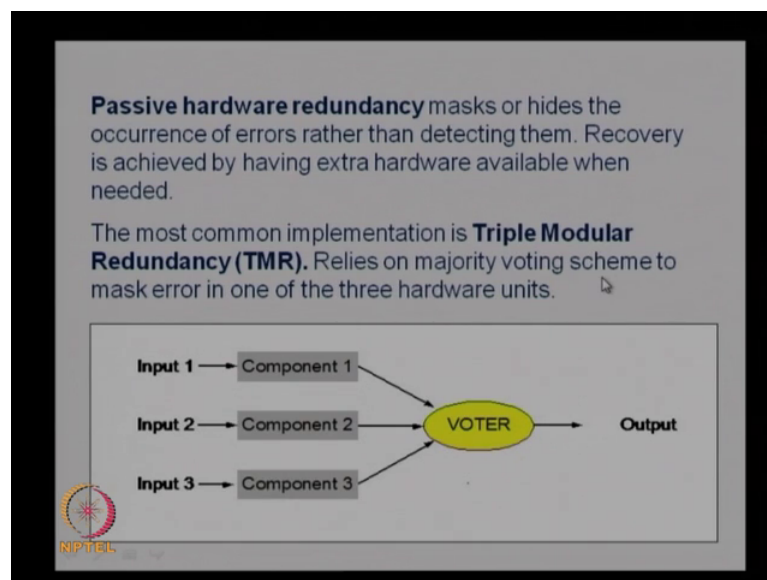
So, hardware redundancy basically uses extra hardware to enable the detection of errors as well as to provide additional operation hardware components after errors have occurred. So, this hardware redundancy as a name suggest will have multiple components of the same function. So, you will be having a 2 or 3 components, which actually provide the same function and whenever there is an error occurring in one of this

a hardware, the that error can be detected and their other hardware can actually take over the a functioning of that the particular a failed hardware.

So, this is the basic hardware redundancy like in the case of that aircraft, we have a 3 hydraulic systems while only one hydraulic system is sufficient to provide the control plane actuation, we provide 3 hydraulic system and 3 will be a all the 3 will be powered separately. So, again 3 separate power sources for this hydraulic system. So, basically we provide a redundancy in the hardware hydraulic system. But then again we have to ensure that these 3 hardware alone is not sufficient, but we need to have additional hardware also because the single point failure needs to be eliminated.

So, but hardware redundancy is one of the a primary method for fault tolerance in the system. And this can be implemented in passive active and hybrid forms. So, you can have a passive hardware redundancy or we can have an active hardware redundancy or we can have a combination of passive and a active which is known as the hybrid a form of hardware redundancy. We will look at the how do we actually implement these redundancy, that is hardware redundancies in passive active and hybrid forms.

(Refer Slide Time: 10:26)



The passive hardware redundancy, it actually masks or hides the occurrence of errors rather than detecting them. Recovery is achieved by having extra hardware available when needed.



So, here in the passive hardware redundancy it will not really detect, the error it will basically mask the error. If whenever there is an error occurring the system will automatically mask that error or hide that error and function as if nothing has happened or as if there is no error in the system. So, what actually happens the if there are multiple hardwares or the redundant hardware, if the one of the hardware fails then it the automatically the other hardware will take over without informing the system that there is an error or without knowing that there is an error in the system. So, that is the passive hardware redundancy, where it will try to hide or mask the error rather than detecting them.

A recovery is achieved by having extra hardware available when needed. So, since there are additional hardware available. So, recovery is you see, it automatically recover from the error because the other hardware will take over the function and provide you the normal functioning of the system. So, that is the passive hardware redundancy. The most common implementation of a passive hardware redundancy is known as a triple modular redundancy. This relies on majority voting scheme to mask error in one of the 3 hardware units. So, this is the very basic implementation triple modular redundancy where the error will be masked or hidden by the system and a recovery will be achieved through additional hardware which is available. The triple modular redundancy basically the diagram for the typical modular redundancy is shown here. As you can see here there are a 3 components component one, component 2 and component 3 and all are identical and the there will be getting the a same input from the a one of the previous system or the subsystem.

And that will be processing the input and getting an output over here. So, all the 3 will be providing identical output in the normal situation and the all the output will go to a voter. And voter will take the voting out from this 3 and then decide whether all the 3 are equal or 2 are equal and or they are a totally different. So, based on this voter a scheme an output will be generated. So, we can see here the 1 2 and 3 will be identical and if one is different from 2 and 3 then the output from 2 will be taken or if the a one and 2 are identical then the output from 1 or 2 will be given as an output. So, 3 will be disconnected.

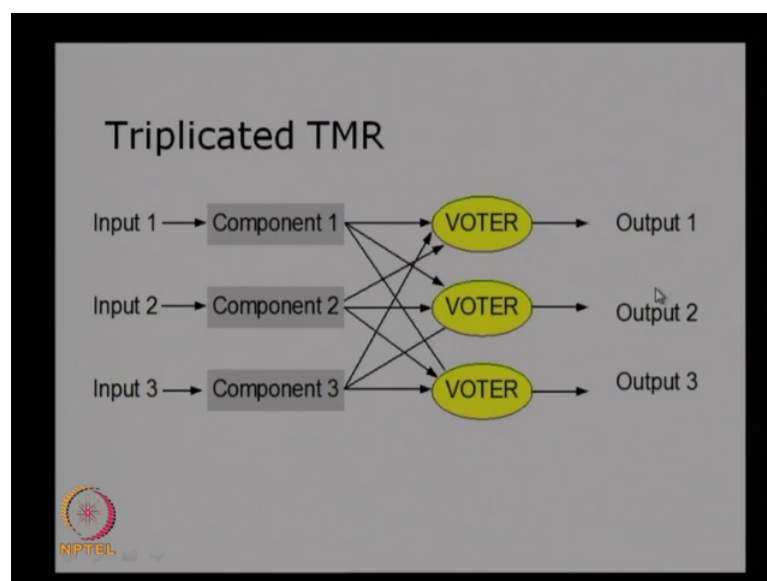
So, in any case if there is an error one error, then this one error can be easily masked by the this system that is why it is known as triple modular redundancy. So, we can have

modular redundancy in this case by providing 3 redundant hardware and providing the output from this hardware and the voter will decide whether there is an error or not. And if there is an error or one of the component output is not equal to the other 2 and that will be an error and without even declaring that is an error, it will go for a next 2 identify the other outputs and then given output to the system.

So, it actually mask that error in one of the components and then provide continue to provide the output, that is known as the triple modular redundancy. A problem here is that you can only mask one error; if there are errors in the 2 of these then it will not be able to mask and then that will be a failure here. Because if all the 3 are different and then in that case if there are 2 errors in these 2 components any 2 of this components then that error cannot be masked and it will be a failure here that is one problem with that triple modular redundancy and the other one is the about the voter which is a single voter. If there is a failure in this voter then again it will fail then this becomes a single point failure.

So, a triple modular redundancy is a basic a building block for hardware redundancy, but as such it alone cannot have a great value in hardware redundancy, because this it can mask only one error and the a voter a error it will not be able to be tolerated because it becomes a single point failure. So, in order to overcome this another scheme is proposed which is known as the triplicated the TMR.

(Refer Slide Time: 14:35)



So, as the name suggest a triplicated TMR is basically a 3 voters. So, we have a tripled modular redundancy as a building block and then we triplicate it.

So, we have this there are 3 voters here. So, the single point failure is eliminated here and then we will have a output voter all the voters are here. So, we take this voter and a output from here as an output 1 output 2 and output 3. So, as we can see this output from component one is given to the 3 voters, similarly component 2 is given to the 3 voters, and component 3 also given the output of component 3 also given to 3 voters and this input to the voters will be compared here and then an output will be generated; if the output from 2 and 3 are a same and 1 is not correct then the output from 2 will be given to here and this output will be coming from here.

So, like this all these 3 voters will be providing an output. So, even if one of this voter is not performing well, these 2 output will be the same. So, you can actually get these 2 outputs same and this output will be different and again this will be connected to a TMR and similar to the previous one and then finally, there will be a single voter and that will give an output. So, the first level of single point redundancy, a single point failure is eliminated by providing 3 voters and again this will be connected to a TMR and that will give you a again another output.

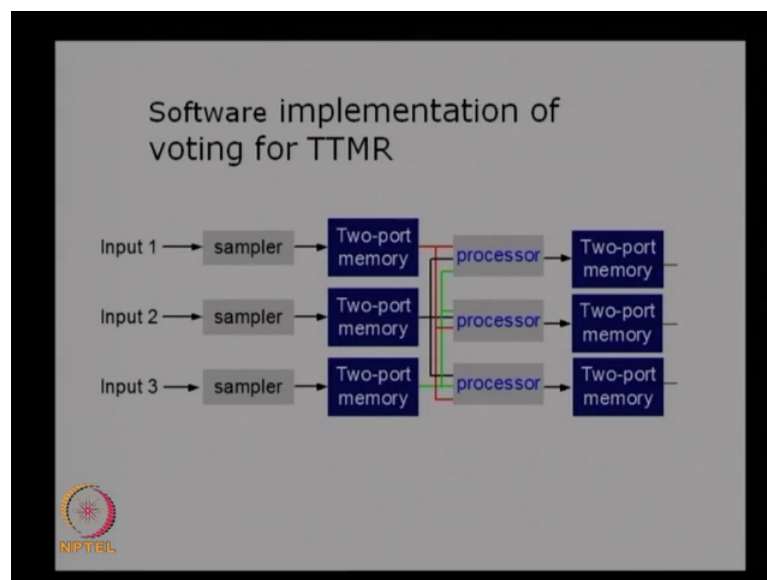
So, like that we can eliminate the a possibilities of error in voter by having a triplicated TMR. So, that is one way of improving the a performance of a triplicated modular redundancy yeah. And then problem with the TMR as I mentioned is that it can mask only a single error. So, here it can be masked only a single error. So, if you want to mask more than or hide more than a one error we need to increase the number of components. So, when we say triple modular redundancy we are talking about 3 components and one voter. So, we can have n modular redundancy instead of triple we can have n modular redundancy, in that case if you have a 5 modular redundancy then we will be having a 5 components here.

So, it can mask 2 errors similarly if you have a 7 we can actually mask a 3 errors and so on. So, if you want to have to mask more errors we to have more number of components. So, we can actually go for triple or a 5 or s7 or a n modular redundancy. So, instead of TMR we can go for NMR to hide more number of errors, that is the way how the passive hardware redundancy is implemented in the system to a hide the errors and recover from

the errors. So, this is the passive hardware redundancy and this voter is a one of the a important a element in the TMR, and here again we have the issues of a synchronization and computational time.

So, if the synchronization of the voters are not voter is not properly implemented, all the computational times are different then you will be having there is a possibility of error. So, these need to be taken into account when we are having when we implement the voters. So, voter's voter can be implemented either through hardware or through software and implementing through hardware is bit costly. So, most of the times voter will be implemented through software and the software implementation of a voter is shown here.

(Refer Slide Time: 17:57)



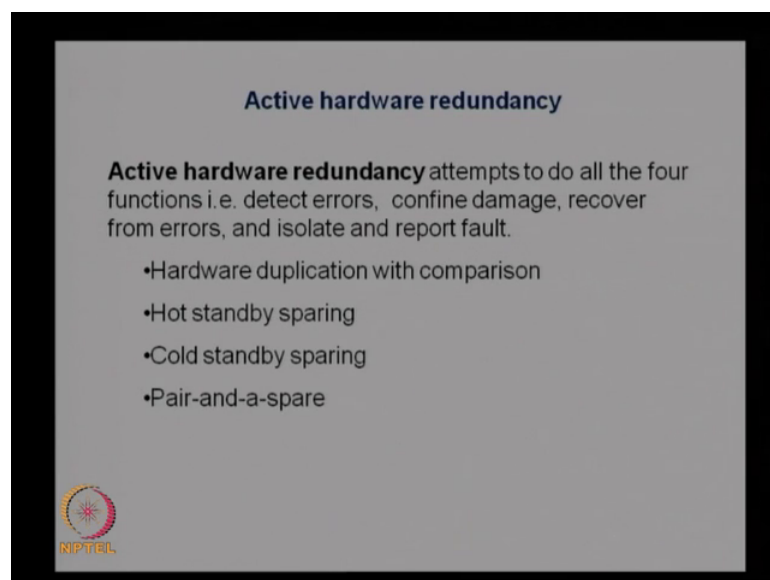
So, we will take the input from a different sources and then it will pass through a sampler and that will be given to a 2 port memory and then to a processor.

This output from this will be given to the processors the 3 processors will be there, and these 3 processors will be a processing this data all the memory will be connected to these processors. So, each processor will be getting 3 inputs, and then it will check this values it will compare the values and then it will send an output to a next to 2 port memory and this is the way how the voting is implemented in triplicated TMR.

So, this is the very important in implementing the redundancy, because voter is one of the most crucial critical item in passive hardware redundancy. But most of the time it will be implemented using software because it is one way it is easy to implement and the other one is that cost of implementation is also reduced here, but of course, there is a possibility of failure because of the computation time and synchronization, that has to be taken care while we implement a voters in a triplicated TMR as well as other modular redundancies.

So, that was about the a passive hardware redundancy the next one is the active hardware redundancy.

(Refer Slide Time: 19:13)



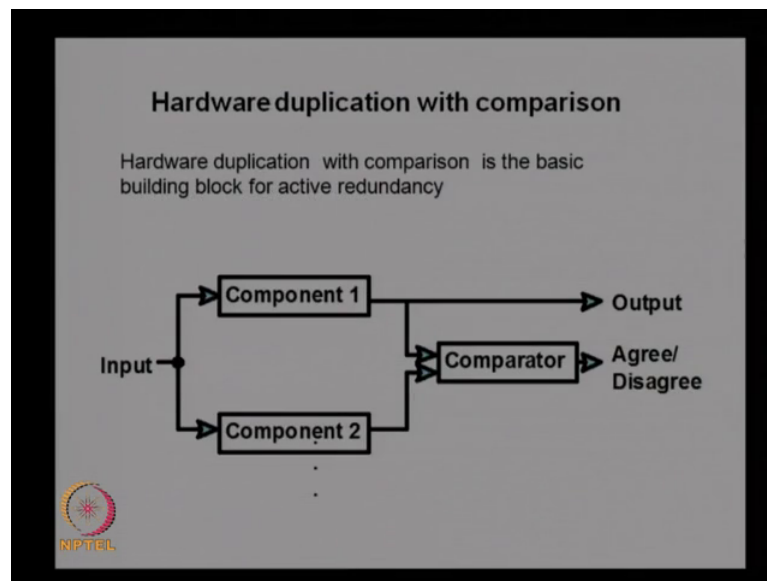
So, compared to a passive hardware redundancy, in active hardware redundancy we will try to identify the location of the a error, you have to do all the functions of a fault tolerance. In passive hardware redundancy we do not really declare an error or we do not know whether there is an a error happened, because it will simply hide the error. But in the case of a active hardware redundancy, instead of hiding the error we will try to get the source of the error and then carryout all the other operations of a damage confinement and reporting and other activities.

So, that is the basic difference between a passive hardware redundancy and active hardware redundancy. In active hardware redundancy we will try to identify the source of error and declare the error and report the error and then mask the error and then

carryout all the other operations like reporting and damage confinement and recovery. Those things will be carried out once we declare the error and identify the source of error; that is active hardware redundancy and it will do all the four functions i e detect errors, confine damage recover from errors and isolate and report fault.

So, all these four functions need to be carried out in the active hardware redundancy. We can actually do it by a different methods, hardware duplication with comparison is one of the basic blocks for active hardware redundancy. So, to implement the active hardware redundancy, hardware duplication with comparison is a must. So, that becomes one of the basic building blocks for active hardware redundancy. And the methods are basically hot standby sparing that is one method and the other one is cold standby sparing and the another one is pair and a spare methods. So, these are the different methods of implementing the active hardware redundancy. Let us look at the methods by which we do this the hardware duplication with comparison.

(Refer Slide Time: 21:11)



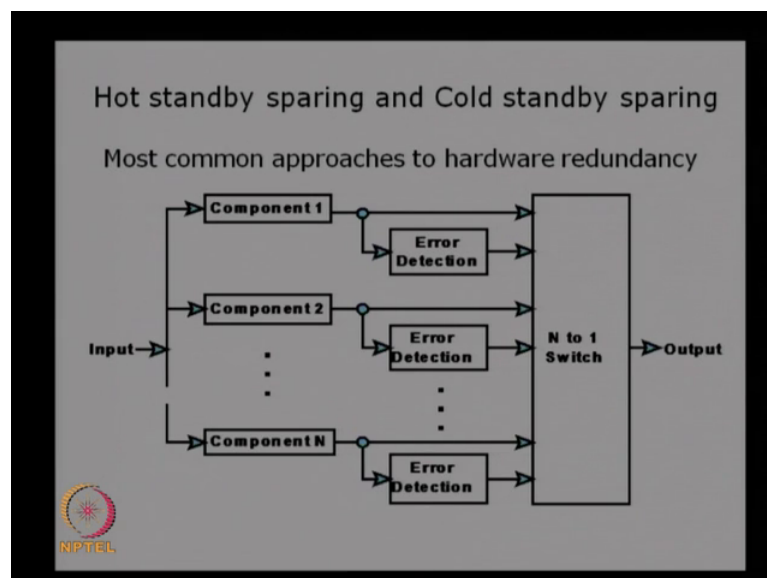
So, you can see here and this is a basic building block for active redundancy as I mentioned. So, here you can see these basic building blocks are needed for any implementation of hardware redundancy. So, what we will do here is to have n redundant components. So, we will be having component 1 2 3 etcetera depending on the requirement, we will be having many number of components and then the output from single inputs and the output will be compared in a comparator and the comparator has

been set with the predefined error value. So, if the output from this one is the comparator is and the component one compared here and component 2 is also compared.

And if they do not agree with the predefined value, then an error will be declared then it will be declared that component one is not in line with the expected output therefore, there is an error in this component. So, that is the way how we declare the error and then output will be if there is a output going from here, this one component one then it will be declared and send a error therefore, this output will be discarded and then another output will be taken from component 2. So, like this if we have n number of components we can have if there is an error in one that will be declared and the output will not taken from the instead 2 will be taken, similarly if there is an error in 2 the other one will be taken.

So, like this the duplication with comparison is there. So, we have a duplicate a components and the comparison of the output also therefore, the comparison actually helps to declare the error or to identify where the error has occurred. And that way the error detection is made possible in the hardware duplication with comparison. So, that is the building block for any type of active hardware redundancy. So, here you can see that this is the way how the duplication with comparison works, and then if you want to implement the.

(Refer Slide Time: 22:57)



Hot standby sparing and cold standby sparing these are the 2 important methods in hardware redundancy and they are most commonly implemented and the again the building block is from the hardware duplication and comparison.

So, we will have that the basic blocks in here and then implement the hot standby sparing and cold standby sparing. So, let us see how do we actually implement it. So, this is the way how the hot standby sparing is implemented. As you can see here this is the basic building block where we have the component one and then error detection that is the duplication with the error detection. So, we have component 1 2 etcetera up to n and each one is having an error detection function also.

So, all these functions the components are same; that means, you have duplicate hardware or the redundant hardware's which actually provide similar outputs from the using a similar inputs. So, all these components will be giving the same output to the system and using the same input. And then there is an error detection function over here, the role of this a component error detection component is basically to look at the output from this component and compare it with a predefined value or a set value.

So, this detection this algorithm or will be written or this will be having a predefined value like the pressure or the temperature expected temperature or expected pressure or expected time of processing, and they will check whether this is the output from this component one is coming as the same or not. And if it is not same then there will be an error declared. So, it will be declaring a error here. Similarly component 2 also will be having an error detection function. So, whenever there is a variation from the desired output it will declare an error. So, same way we will be having many n components and then error detection functions and this will be used for detecting an error in the system. So, all this output from these components and the error detection will be send to a n to one switch.

So, always one output will be going from here. So, this switch will actually look at these outputs from all the components and error detection functions. So, if component one the output and the error detection function, if the error detection declares there is no error and then the output from this will be given as an final output from the switch. If this declares an error and then this output will be discarded immediately it will switched to the second switch or the second output will be taken. So, this switch will basically switch



between the output from components. Based on the error detection it will decide from which output to be chosen from here and send as an output.

So, that is the n to one switch. So, here whenever there is an error detected in one of the components, that component will be declared as a faulty system or a faulty component and the data will be stored in n to one switch and then the later all the other actions like a damage confinement and all other functions of fault tolerance will be carried out at the after once the error is declared. So, here the error detection is there and this error detection data will be sent to the next level for other functions in the fault tolerance. So, that is how the standby sparing works? Then what is the difference between hot standby and cold standby in hot standby all these components will be always active.

So, here you can see that component 1 2 3 they are all having the same function and all will be active. All the functions will be active and all the components will be in active modes, and there are in the hot situation or that is why is known as the hot standby. So, all those components are in active modes and whenever there is an error detected immediately the output will be taken from here. So, there is no time delay between the output from here, because all the outputs are available at anytime the outputs are available and whenever there is an error detection in one of this immediately the switch will change to a next mode or the output from the next step component will be automatically chosen and given as an output.

So, there is no time delay always will be getting an output from the system that is known as the hot standby when the system where we cannot have a delay in output or very critical and you cannot have a stoppage of output for a short duration in such situation, we need to go for hot standby. You can actually compare it with system where we have a a UPS and a power supply to a computer. So, most of the times your ups also will be in a ready mode to supply power. So, it is an active mode and or you can consider the battery or source of a laptop and the power supply.

So, you will be giving a power supply and power supply will be always available. So, when power supply is available, it will directly take the power supply from the external source and start functioning. But at the same time batteries also in an active mode because it is its ready to give power at any time you do not need to change the mode to get to the battery mode to get the power supply. So, this can be considered as a simple

form of a hot standby sparing method. So, we have a battery and an external power source both providing the power supply to the computer. And one of the phase if the external power supply phase, you will not even notice that there is a failure because automatically the battery will take over the power supply without any considerable time delay and automatically you will be getting the power supply and the system will start functioning with the as if there is no problem happened.

But of course, it can actually the system can actually identify that there is a power failure, and that data will be sent to the other processors and so, the mode will change your brightness of the display will change and all those the things will take place within the system because already a an error has been declared, and base all the actions for that tolerance of that particular error will be taking place after that. So, that is why the hot standby actually helps to reduce the time delay in taking over or the change of outputs. So, here you can see that you can take component one as a an external power supply and component 2 as the battery source in the laptop then you can see the error detection is basically where there is a power coming from the external source or not.

So, if this detects an error, automatically this output will be taken and been given to the system the processors the always there will be a continuous power supply to the computer even if there is a failure in one of this. So, that is a simple example for a hot standby sparing. But in cold standby the system will be almost the same the only difference is that there this will not be in an active mode. So, they are all capable of doing the same job, they can actually provide the power supply whenever needed, but you may need to activate those these components to provide the power supply.

So, that is why it is known as cold standby they are not in a hot active mode, but they are in a cold mode. So, if you want you can actually take. So, normally this component 1 will be providing an output and 2 and other components will not be providing any outputs, and whenever there is an error detection immediately this will be started and the output will be coming from here and then this will be going as an output. So, there may be a small time delay between these 2 outputs, since they are not in the hot mode they are in the cold mode and they need to be activated.

So, there will be a time delay in getting the output. So, that is the main difference between hot standby and cold standby this can be explained with again a very simple

example where you have a generator to provide power supply, and you have the external power supply coming from the normal supply lines. So, whenever there is a power failure we can actually start the generator and then give the power supply. The only problem here is that the generator will not be in an active mode. So, whenever you need you need to start the generator and then get the power supply though the power supply can come within a short duration. So, that short duration is important because since they are not in an active mode we need to take care of those that delay and ensure that the system can actually tolerate that delay without having major problems in functioning.

So, if that is a situation where that such delays can be accommodated, then we can go for cold standby. So, if you are having a presentation going on in the classroom and suddenly the power fails we can actually change to a generator some other a power source and start the power source and continue with the presentation. So maybe a loss of one or 2 minutes not so, critical in that such situation we can go for a cold standby sparing because the time delay is not a major issue. But in the case of a hospital where there is a surgery going on and continuous power supplies needed they cannot wait for some failures or something if the power goes then system has to be boot up or there are some other issues.

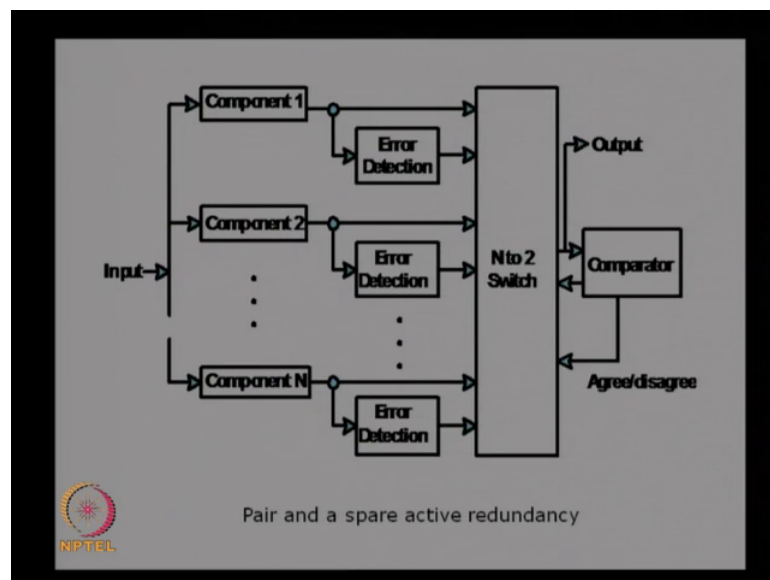
So, in such situations nobody can actually wait for a few minutes to get the power supply, a continuous power supply to be ensured under such situation we need to keep the generator on and the main power supply on whenever there is a failure in one of this the automatically the other power supply should come and take over the activities. So, action should continue the system performance for function it should continue without any time delay. So, such situation we need to go for a hot standby sparing methods. So, these are the 2 common approaches in hardware redundancy; one is the hot standby and cold standby both will be physically both they will appear to be the same there are not much of difference in the way it is implemented, the main difference in their functioning is that in one system that is in a hot standby all the spare components are in the active modes.

But in the cold standby they are not in the active mode, they need to be activated depending on the needs that is the main difference between the hot standby and cold standby sparing method. So, the main components here are basically the duplicate components which actually provide the function and then your error detection function

which actually will declare there is an error in any of these components, and there is an n to one switch. So, these are the main components needed for implementing the standby sparing method whether it is a cold standby or hot standby. So, these are the components needed and then this switch will actually decide which output to go from here.

So, when even there is a error any one of this there will not be any problem. So, if you have 3 components even if the even 2 fails it will actually takeover and the output can be given from this switch. So, unlike the triple modular redundancy where TMR can mask only one error; here actually we can mask any number of errors depending on the number of hardwares it provides. So, that is the a stand by sparing methods.

(Refer Slide Time: 33:43)



Another method is the pair and a spare active redundancy. So, this is again another method of a redundancy a active redundancy, this again can be a hot or cold depending on the way the components are arranged.

So, if the components are in active mode then it is a hot standby if they are in a non-active mode then it is a cold standby method. So, here you can see the first part remains almost the same you have components and error detection here. So, in addition to that, we have the a switch in set of an N to 1 switch we will have an N to 2 switch and then this N to 2 switch will be the output will be connected to a comparator and then it will declare whether there is an error or not. So, what this switch will do? It will take instead

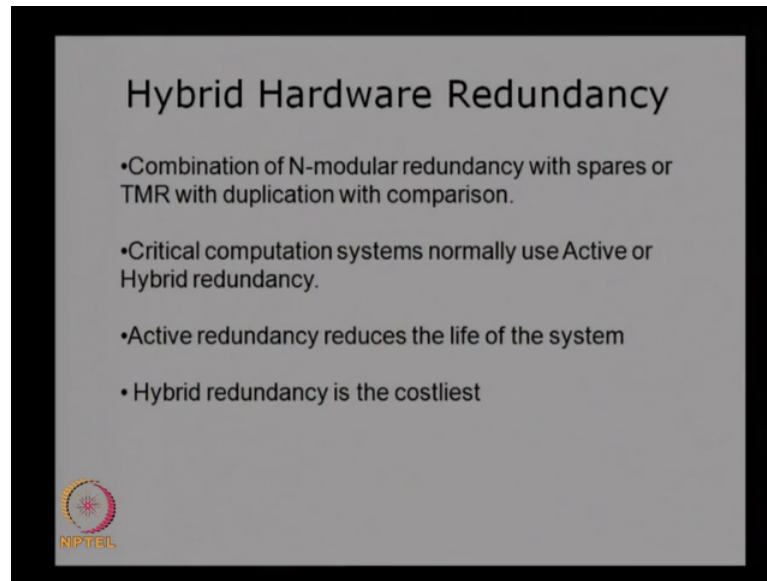
of in the previous case it will actually take one output and then check whether it is an error is there or not and give an output.

But in this case the N to 2 switch will take 2 outputs from here. So, any 2 component output will be taken which are error free. So, if these 2 are error free these 2 outputs will be taken and then it will be sent to a comparator, and then compare the outputs and then declare an error or not. If the comparator agrees that the 2 outputs are the same then that output will be send as an output from this N to 2 switch. So, we are actually including one more stage of comparison of 2 outputs to give a main output from here. So, this is known as a pair and a spare active redundancy.

So, instead of having one output going from here, you take 2 outputs compare the output and then see whether there is an error or not and based on that an output will be given. So, it is a little bit more redundant, because we have a more comparison over here. So, even if these 2 is not declaring any error still we will go one more level of checking whether these 2 are having the same output, then only the output will be given. If these 2 are not same then another 2 outputs will be taken and that will be compared here and if they are agreeing then only that output will go.

So, we are having an additional level of checking to ensure that there is no error between the outputs. So, even if there is an fault in this error detection. So, if this error detection fades still we can actually get an output from here because there is an another comparison at this level. So, the error the failure of this component can actually be masked or hidden using this comparator. So, that is why it is known as a pair and a spare active redundancy and as I mentioned it can be a hot or cold redundancy based on the where it is being implemented. So, a component if they are active, then they are hot redundancy otherwise it is a cold redundancy.

(Refer Slide Time: 36:20)



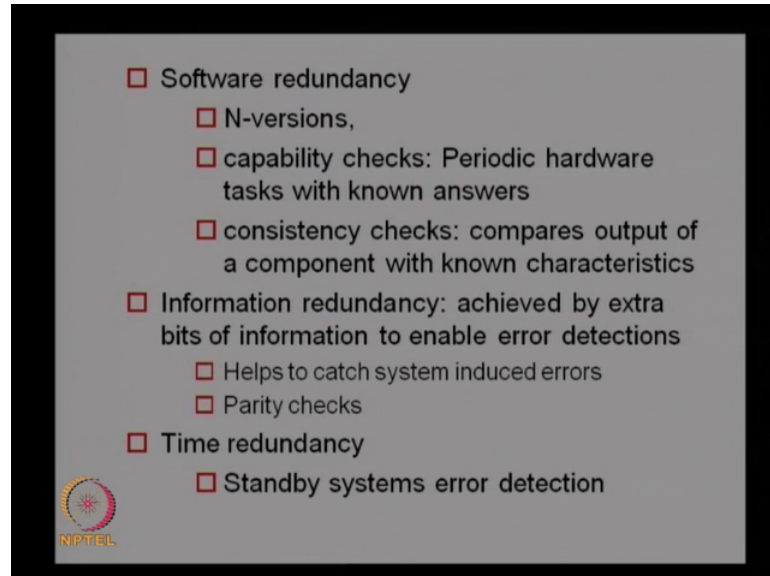
And as I mentioned there can be a hybrid redundancy also, we can have a active redundancy and passive redundancy and we can have a hybrid redundancy, but we will use a combination of N modular redundancy, with spares or TMR with duplication with comparison. So, mentioned about the triple modular redundancy or N modular redundancy. So, this is actually a passive system. So, we can actually combine this n modular redundancy with spares the standby spares we discussed. So, we can actually combine this or we can compare the TMR with duplication with comparison.

So, if we have this kind of a combination, then it is known as a hybrid modular redundancy or hybrid hardware redundancy. Basically depending on the situation this can actually be implemented in some cases somewhere you may need a active redundancy and some cases you may need passive or in some cases we need to go for a combination of these methods. Most of the critical computation systems normally use active or hybrid redundancy; because active redundancies are basically we will tell you the location of error is and where actually it is coming from and then we can actually confined the damages, that is why most of the critical computation systems normally use active or hybrid redundancy.

Active redundancy reduces the life of the system basically when you go for a active especially the hot redundancy, active redundancy in terms of hot standby reduces the life of the system because it actually always in an active modes and hybrid redundancy is the

costliest one to implement. Because you need to have the passive as well as the active redundancy and therefore, it is the costliest to implement practically.

(Refer Slide Time: 37:58)



And apart from the hardware redundancy, we have the software redundancy also. In software redundancy there are different ways of implementing the software redundancy one way is to have a different versions of a software. So, if you have developed different versions by a different set of people so that the errors in one of the version will not be there in the other version. So, you can actually switch over from one version to the other version and ensure the smooth functioning of the software. So, you can have 2 operating systems in your computer. So, if one operating system fails you can go for the other one. Similarly you can have a software also, if you have use a explorer for browsing and explorer is giving an error you can actually switch to some other browser and then continue with your browsing activities.

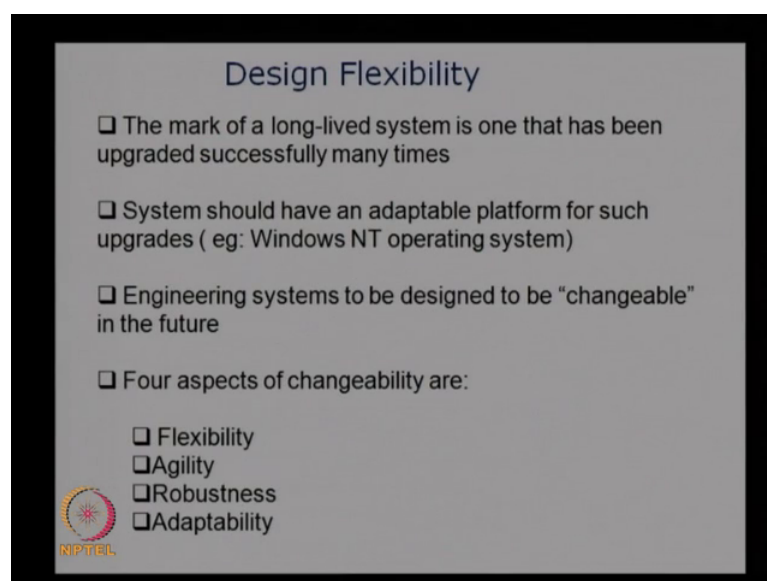
That is one of the ways of having different kinds of software or you can have different versions of the same software to have the software redundancy. And other methods of doing the software redundancy is basically capability checks, in capability checks a periodic hardware tasks with the non-answers will be carried out by the system. So, without specifically instructing the system to carry out such task, continuously doing periodic hardware task; there will be some specified task it will carry out some specified

tasks like running a motor or switching of a light and will check whether it is working or not, that is again a implementation of redundancy the software.

And consistency checks are another way of doing it basically it will compare the output of a component with the non-characteristics. sometimes it will check the output from a component and then see whether it is consistently giving the same output or not. So, that is kind of a method is known as consistency checks. And then we can have information redundancy that is achieved by extra bits of information to enable error detection. So, we will have additional bits in a data so that we can actually compare that additional bit with required data and then see whether we are getting the correct information or not. So, that is known as information redundancy.


And this helps to catch the system induced errors and as well as you can do a parity checks on also with this end of information redundancy. And then time redundancy is basically using the extra time available in the processor to carry out some kind of additional checks; if there is a spare time available then the system will run some checks using the spare time and see whether there is any errors in the system and that is known as the time redundancy implementation for a software's or implementation of redundancy. Using the time redundancy is known as standby system error detection is easily possible using the time redundancy.

(Refer Slide Time: 40:32)



**Design Flexibility**

- The mark of a long-lived system is one that has been upgraded successfully many times
- System should have an adaptable platform for such upgrades ( eg: Windows NT operating system)
- Engineering systems to be designed to be "changeable" in the future
- Four aspects of changeability are:
  - Flexibility
  - Agility
  - Robustness
  - Adaptability

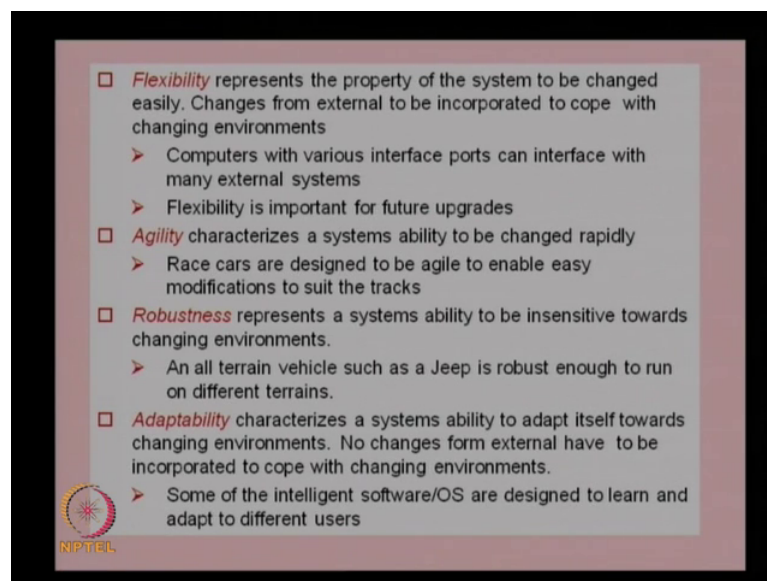
 NIPTEL



And the additional the important activities to be incorporated in the physical architecture development is basically the design flexibility. In design flexibility we need to ensure that long lived system is that has been upgraded successfully. So, we need to ensure that the system has the flexibility to upgrade at in the future and system should have an adaptable platform for such a upgrades. So, you need to ensure that the hardware system has a platform where we can go for future expansion and they should be a changeable in the future also we do not make it. So, rigid it should be always made a changeable or easy to make changes in the future.

So, any hardware, any physical architecture we develop should be made in such a way that it can be changed in the future. So, the changeability in order to ensure the changeability there are four aspects, they are known as the flexibility agility robustness and adaptability; all these four features need to be incorporated in the architecture so that we can have the prepared changes various stages.

(Refer Slide Time: 41:33)



So, flexibility is basically the property of the system to be changed easily, changes from external way incorporated to cope with the changing environment. So, whenever there is a change in the environment you should be able to change the system and that is known as flexibility.

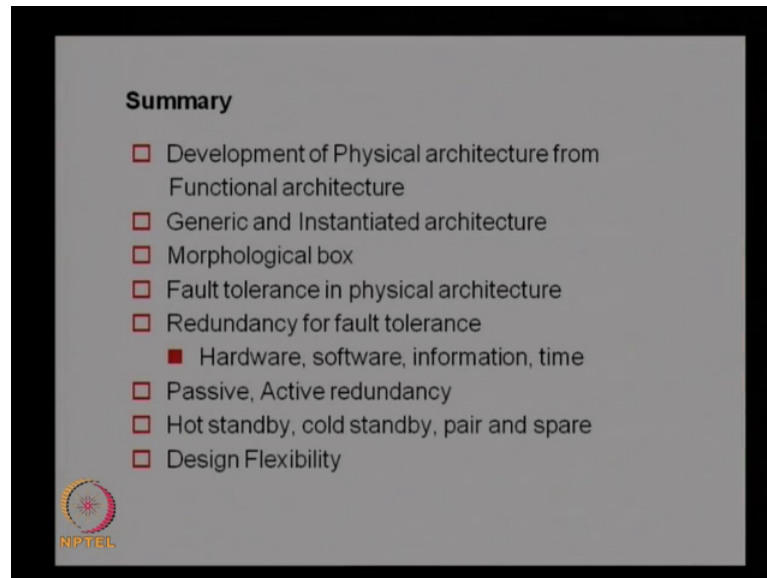
Some examples are computers with various interface ports can interface with many external systems. So, you provide with the a various interfaces so that you can interface

the computer with various external systems. So, we have the flexibility of connecting a printer or a thumb drive or a camera to the computer. So, that kind of a flexibility can be provided in the a physical architecture and this is very important for the future upgrades also. Agility characterizes a systems ability to be changed rapidly. So, if you want to change the system very fast then the agility is one important characteristic. Like race cars are designed to be agile to enable easy modifications to suit the tracks.

Then the next one is the robustness, which represents a systems ability to be insensitive towards changing environments and all terrain vehicle such as a jeep is robust enough to run on different terrain. So, that is how we make it robust. So, various terrains you make a system capable of operating under various situations then it becomes a very robust. And that is last one is the adaptability, which characterizes a systems ability to adapt itself towards changing environments no changes from external have to be incorporated to cope with changing environments.

So, some of the intelligent software operating system are designed to learn and adapt to different users. So, that is the adaptability. So, we adapt that is the way for the changing environment or we learn from the environment and then modify the system or modify the performance accordingly and this is known as adaptability though. All these characteristics are important for any system and when we develop the physical architecture we must ensure that we take care of these factors also in the identification of the architecture as well as identification of the components.

(Refer Slide Time: 43:35)



So, with that we will complete the physical architecture development for the system and here we actually discussed the development of physical architecture from functional architecture and then we discussed about the generic and instantiated physical architecture, we discussed about the morphological boxes, and we discussed about fault tolerance in physical architecture and how this is implemented using redundancy hardware or software and information we discussed about the passive and active redundancy, and cold standby pair and a spare method also.

So, all these methods of implementing the physical architecture is the code for developing a system and we need to ensure that whenever we develop such systems, we look at the physical architecture look at the functional architecture and then from the functional architecture make a logical step towards a physical architecture. Now identify the components implement the additional functions like fault tolerance and make sure that the system has got the characteristics for future expansion also.

And once we implement all these when we are getting the a physical architecture for a system, which will actually form the basis for further development of the system or further implementation of the system or practical development of the system. So, we will look at the other aspects of the system development in the next lecture and till then good bye.