Seminar
Fault Classification & Fault Tolerance Metrics for NoC

By
Tassawer Amin
Advisor
Michael Imhof
Table of Contents

1. Table of Figures .................................................................................................................. 2
2. Introduction ...................................................................................................................... 3
3. Reliability Issues ............................................................................................................... 4
   3.1. Variability .................................................................................................................... 4
      3.1.1. Random Dopant Fluctuations .................................................................................. 5
      3.1.2. Sub Wavelength Lithography .................................................................................. 5
      3.1.3. Heat Flux ................................................................................................................ 6
   3.2. Transistor Performance Degradation (Aging) ............................................................... 8
   3.3. Single event upsets (soft errors/transient faults) ......................................................... 9
   3.4. Intermittent Faults ....................................................................................................... 12
4. Network on Chip (NoC) .................................................................................................... 15
5. Fault Tolerance .................................................................................................................. 17
   5.1. Fault Tolerance Metrics .............................................................................................. 18
   5.2. Fault Tolerance Metrics for NoC ................................................................................. 19
      5.2.1. Avoidance ............................................................................................................. 21
      5.2.2. Detection ............................................................................................................... 21
      5.2.3. Containment .......................................................................................................... 22
      5.2.4. Isolation .................................................................................................................. 23
      5.2.5. Recovery ............................................................................................................... 24
6. Summary ............................................................................................................................. 24
7. References .......................................................................................................................... 25
Table of Figures

Figure 1: Increase in no. of transistors On a Chip
Figure 2: Decrease in the transistor physical gate length [11] .................................................. 3
Figure 3: Decrease in the no. of the dopant atoms over the years [10] ........................................ 5
Figure 4: 0.35μm Generation from when wavelength lithography started to use for transistor patterning [12] ........................................................................................................ 6
Figure 5: Heat Flux Challenges [1] .................................................................................................. 7
Figure 6: Heat Flux across a microprocessor die [10] .................................................................... 7
Figure 7: [5] Bath tub curve describing a product life time over time ........................................... 8
Figure 8: [10] Estimated degradation in transistor performance over time .................................... 9
Figure 9: Ionization taking place and glitch in current pulse caused by the high energy ion [3] .......................................................................................................................... 10
Figure 10: Soft event upsets and Soft Error transient ................................................................. 11
Figure 11: Soft error failure in chips in different generations [10] .............................................. 12
Figure 12: Suspending the core execution affected by intermittent fault [7] ............................... 13
Figure 13: Use of spare cores [7] .................................................................................................. 14
Figure 14: OS reconfiguration technique [7] ................................................................................ 14
Figure 15: Overcommitted technique [7] ....................................................................................... 15
Figure 16: Crossroad for large SoC and multi processor chips [2] ................................................ 16
Figure 17: NoC architecture [9] .................................................................................................... 16
Figure 18: NoC Communication Layers [8] .................................................................................. 17
Figure 19: Fault Detection or repair at different Hierarchal levels ............................................. 20
Figure 20: Avoidance Mechanisms ............................................................................................... 21
Figure 21: Different Error detection techniques .......................................................................... 22
Figure 22: Fault containment region ........................................................................................... 23
Fault Classification & Fault Tolerance Metrics for NoC

Introduction

Technology scaling has tremendously improved the performance of VLSI systems and now we are able to integrate millions of transistors on a single chip which is really remarkable and the way in which more and more work is being done on scaling it is estimated that by the year 2014, it would be possible to integrate 2-5 billion transistors with 10-13 nm gate on a chip area of 500-700 mm$^2$ using high volume integration process.

![Figure 1: Increase in no. of transistors On a Chip](image1.png)  ![Figure 2: Decrease in the transistor physical gate length](image2.png)

But on the other hand, more transistors and more computing power means there would be more and more internal capacitance and switching activities which leads to more power dissipation and leakage current. Which are considered as the biggest challenge to future scaling hence reliability of these systems is emerging as the biggest concern which urges the need to better understand the failure mechanisms which are affecting the reliability of these systems.

The section I of this paper classifies all the faults related to the reliability of the transistors and section II illustrates the transient and intermittent faults & finally in the last section we will discuss about the fault tolerance metrics for NoC.
Reliability Issues

As technology scales further we will face new challenges such as variability, single event upsets (soft errors), and device (transistor performance) degradation, these effects manifesting as inherent unreliability of the components, posing design and test challenges [10].

Although work is being done to reduce the effect of all the above defined factors like error correcting codes are being used in memories to effectively detect & correct the soft errors & also careful frequency binding deals with the transistor performance degradation but the biggest concern is that the way technology is scaling the effect of these factors will also increase and there is a strong need to devise technology to cater these effects in an effective manner so that the reliability of devices can be guaranteed.

Variability

Variability in transistor behavior is subject to increase in future, not only at the manufacturing level but also at runtime as well. When the components of your circuit start showing variable behavior due to static or dynamic causes of variation then it has a very significant impact on the overall behavior of you circuit.

Variations can be either physical or environmental, physical variations are caused during the fabrication process of chip due to incorrect dimensions of chips or difference in effective channel length etc. and environmental variations are caused due the unpredictability of operating conditions like change in chip’s temperature or power supply voltage.

Variations can be further categorized as either inter die or intra die, inter die variations have the same value across the whole die where as compare to intra die variations which varies from die to die and inter die variations are caused due to the shift in the mean value of parameter distribution form its normal value which are produced due to variation in the thickness of different metal layers. A slight shift in the mean value is always expected and also addressed at the system design level. But if that shift is significantly bigger so that it drags the mean value of your circuit out of its safety critical regions then it can variability in your system behavior.

There are mainly three sources which cause variation in a transistor behavior

Random Dopant Fluctuations

Sub Wavelength lithography

Heat Flux
They can be classified either as static or dynamic, those which occur during fabrication process are called static and others are dynamic as they are time and context variant.

**Random Dopant Fluctuations**

Transistor channel is doped with atoms in order to control their sub threshold voltage but the way in which the transistor area is shrinking in every new generation the number of dopant atoms present in the channel of a transistor are remarkably decreasing as well.

![Figure 3: Decrease in the no. of the dopant atoms over the years](image)

As the figure shows there were thousands of dopant atoms present in 1 micron technology but the way technology is scaling we can notice a significant decrease of dopant atoms in transistor channels as well.

For example, when the transistor channel was being doped, number of atoms present on one side of the transistor becomes more as compare to the other sides and if you design a system which also contains some of those transistors then their behavior would be different than other transistors sitting beside them, which you were not expecting when you were designing that system as you were not aware of the fact that there could be some transistors which can exhibit variation in their behavior, so variability in the behavior of those transistors due the difference of doped atoms would affect the overall reliability of the system.

In another case, when impurity atoms get deleted form the transistor’s channel then it would also cause variability in their behavior.

**Sub Wavelength Lithography**

Sub wavelength lithography is another major concern for transistor reliability, which is used for pattering transistors during the fabrication process; sub wavelength lithography is being
used since the 0.35μm generation, right now 193 nm wavelength is use for 65 nm transistors and will continue to be used until some extreme ultra-violet technology (13 nm) becomes available [10].

Sub wavelength lithography causes line edge roughness, corner rounding, line end shortening and local context-dependent line width variations on the transistor surface due to which those transistors tend to show variation in their behavior adding threat to your system reliability.

**Heat Flux**

Heat flux is defined as heat transferred per unit time and per unit area [14].

\[
q = \frac{Q'}{A} \text{ (W/cm}^2\text{)}
\]

When \(Q'\) is uniform over the area \(A\).

When billions of transistors are placed on such a small die size it creates a very big thermal challenge in the chip, just like toady’s desktop computer are dissipating more heat per area than throat of a rocket nozzle or a space shuttle during reentry into earth’s atmosphere.
Heat flux is a dynamic source of variation in transistor as it does not happen at the fabrication level rather it depends upon the functionality of each circuit block as well as the activity and compute load at any given time, so higher the activity higher the heat flux produced by that component would be, higher heat flux increases the temperature of the system which can create hot spots on its surface which causes temperature variation and higher sub threshold leakage. In a given system different portion of that system can have different heat flux values as shown in figure 6, since heat flux merely depends upon the activity of each component if the activities being performed by that component is significantly higher than the others then the value of it’s heat flux would be much higher as compare to other components of that system, for example heat flux value of a cache would be lesser as compare to an execution unit with in a system.

![Figure 6: Heat Flux across a microprocessor die](image)
Transistor Performance Degradation (Aging)

Transistor performance degradation is another major concern for system reliability, traditionally Burn-in test were used to detect and eliminate the components with are quite likely to get early failure rate so that by eliminating these components you can ensure that the rest of the components are likely to work properly for their entire service life time and will not get affected by the aging.

Life time of a product is normally described using a method called “Bathtub curve”, which depicts three periods of a products life, the first period is called infant mortality period in which a product shows decreasing failure rates second phase shows the normal life period in which the product there are very low or constant failure rates and finally the wear out period in which a product is likely to get wear out with the passage of time because of ever increasing error rates.

![Bathtub Curve](image)

**Figure 7: [5] Bath tub curve describing a product life time over time**

But the biggest problem Burn in tests is to quantify and optimize the burn in test periods also it doesn’t guarantee the reliability of an individual component rather it improves the reliability of a whole population of products.

Degradation in transistor performance is really a very big concern right now as even if you produce them with the best of our efforts we just can’t predict any thing about the transistor degradation period as it merely depends upon the stress upon it during it’s operative life, if it would be used rather more extensively it will start to degrade earlier that’s why if you might have used same transistors in different sections of your system, if due to the nature of that system one section of the system being used much more frequently for some reasons while other parts remain idle for most of the time, then degradation in that part of the system would be much earlier and it would start to show variability in its behavior earlier as compare to the other parts.
So, if a system is already suffering with the transistor performance degradation and if it also gets hit by the soft errors as well then combined effects of both of those variability factors can add misery to its life time.

Right now major factor of transistor performance degradation is hot carrier induced degradation which is several affecting the long term reliability of the system. When transistor is operating in saturation region an oxide interface trap is produced near the drain which gradually affects the performance of the transistor and degradation in the transistor performance eventually affects the long term reliability of the whole circuit.

Light Doped Source Drain (LDD) is use to reduce the effect of hot carrier based degradation in which we reduce the strength of the gate electric field at the drain end. Reliability simulation tests and techniques [13] are also used to resolve the hot carrier induced degradation.

The way the transistors are being scaled down in every technology generation this problem is going to become worst as if we can assume that right now if a transistor degrades in 8-10 years then in the coming technology generations when they would more scaled down it is very much likely that they will start to degrade much faster and if their degradation period reaches a minimal value of 2-3 years it would be a really big threat for the future technology generations.

Figure 8: [10] Estimated degradation in transistor performance over time

**Single event upsets (soft errors/transient faults)**

Soft errors are also a very big threat to system reliability as they are also emerging as another major concern for the variability in transistor behavior, they were assumed to be caused by the alpha particles which are emitted from the uranium impurities during the fabrication, and late on it was also found that they are caused by low energy neutrons interaction with the isotopic boron-10 ($^{10}\text{B}$) in IC materials which are used to form insulator layers in IC manufacturing but now by making adjustments at fabrication process they does not exists anymore,
The major cause of soft errors is considered to be the ions which are produced by the high energy neutron reactions with silicon nuclei; the variability caused by these ions depends upon their linear energy transfer which depends upon the mass and energy of the particle and the material in which it is traveling.

![Figure 9: Ionization taking place and glitch in current pulse caused by the high energy ion][3]

The above figure shows how ionization takes and when it comes closer to the depletion region electric field rapidly collects carriers which in result produces a glitch in current pulse.

Soft errors are really a big concern in advanced computer chips right now as the failure rate produced by them is exceedingly bigger than that of the combined effect of all other variability factors, as those normally generate failure rate between 5-150 FITs (Failure in time) but on contrary failure rate produced by these soft errors is almost more than 5000 FITs per chip [3], that’s why these soft errors are considered to be the biggest threat to system reliability.

Technology scaling has worsened the effect of these errors, as earlier when the size of transistor was relatively bigger, the change produced by those ions was not that big so that it can cause any significant variation in transistor behavior but now when the size of transistor has remarkably reduced, variability in transistor behavior caused by those ions is really a very big concern. As in air planes in which so many application consisting of these transistors, are running and they are most likely to effect because of these soft errors as when they would be in sky, soft error rate is likely to increase to very large extent due to the increase in the number of ions and increase in the energy of radiation particles as compare to when these application were running at the ground level, so they will cause variability in the system behavior which is really big threat as the application hit by those soft errors would not be any more reliable. To reduce the effect of soft errors in more critical
applications we normally run two to three copies of the same program and their result are compared consistently in order to prevent them from effects of soft errors.

Memory elements are much more sensitive to soft errors such as DRAM (dynamic random access memory), when a radiation event occurs close to the sensitive node it produces soft errors in DRAM, although these soft errors are easy to find in memories using parity checking in which an additional bit is added to store the parity of data word. Whenever the data is retrieved the checker compares the parity of the data with its own parity bit and if the parity bit does not match, the error is data is detected but this method is not that much effective as it can tell you about the presence of error in your memory but you can’t correct them by means of this method rather error detecting and error correcting codes are normally used to check for soft errors in memories.

Flip-flops and latches are also affected by the soft errors, as they are designed with more transistors and they easily compensate to any suspicious charge collected during radiation events [3], it is really difficult to find and correct soft errors in flip flops.

The energy contained inside the radiation particle which is responsible for that soft error to occur is also of primary importance because if energy contained by that particle is bigger than critical charge of that device that device is most likely to hit by a soft error so another way to protect you devices from the soft errors is to increase the value of their critical charge.

$$\text{Energy of radiation particle} > \text{Critical charge} \rightarrow \text{Soft error}$$

Figure 10: Soft event upsets and Soft Error transient

Suppose you are using latches to store some data which is coming out of some random so event upsets (SEU) are caused when register/latch is hit by a radiation particle and causes a change in the already state of already stored data in your device then this event this is called
SEU because effects of this event will remain, until a new value is written in that register/latch again.

Single event Transient (SET) will occur if your random logic gets hit a soft error and starts producing wrong values.

The figure shows beyond the 90 nm technology generations the soft error rates in latches is significantly increasing in every new technology generations causing more and more problems.

Soft error rates are increasing with every technology node and 8 percent increase in soft error rate per logic state bit per generation is expected [10] as shown in the figure which is really high as the number of logic state bits doubles in every new technology generation so the soft error rates would really cause very big challenge for the system reliability and as it is assumed that it can even reach the value of even more than 100% failure rate in the 16nm technology generation which is of a very big concern for future technology generation.

![Figure 11: Soft error failure in chips in different generations [10]](image)

**Intermittent Faults**

Intermittent faults are another type of temporary faults which tends to occur quite irregularly for a period of time and the main causes are either process variation or in process wear-out along side voltage and temperature fluctuations.

When a system is hit by an intermittent fault it is quite possible that until the fault is active system can produce a false result and produces a correct result when fault in inactive so occurrence of these intermittent faults severely effects the reliability of the system as outputs produced by those systems would not be reliable any more as presence of these faults increases the probability that the produced output can be wrong thus can’t be trusted anymore until the error is removed. Intermittent faults are detected by error correcting
codes and can be corrected using either error control or by restart. Intermittent faults are amongst the biggest cause of system failure and Occur 10-30 times as often as permanent faults.

When ever a system is hit a fault it can be either permanent or temporary, so it is always useful to spend some time to detect the nature of the fault for example, when an error occurs and you declare it as permanent without spending any time to find the nature of the fault then the only option would be to remove the faulty component and replace it with the error free one which is not that much economical in most the cases, so rather if you tend to spend some time in order to find the nature of the fault and it happens to be an intermittent one then it would be much more economical to correct an intermittent fault by either by applying adjusting environmental variations or by restart.

The biggest problem is that, in very large and complex systems fault detection and finding its location is really difficult and time consuming also as there are so many factors due to which an intermittent fault occurs and each factor has its own timescale for which it affects the system so it becomes extremely difficult to find to cause of an intermittent faults in those sort of large and complex systems.

So most important thing is to find the component which is constantly being hit by intermittent faults by applying different tests to your system and checking their output against the expected values which system should generate in each case.

We can take the example of a multi core system in which on of the cores gets affected by intermittent fault then suspending the use of the core which is affected by the intermittent fault is the most commonly used method in order to prevent the system from further effects caused by that fault as shown in figure 12.

![Figure 12: Suspending the core execution affected by intermittent fault](image)

But use of this technique is not that much effective as it can lead to dead locks and the threads depended upon those cores would remain blocked until that core becomes active again and it can have severe effect on the latency of critical applications, as well for faults of short duration suspending the core form the system for longer time is not really a smart idea.
Another technique which is used to reduce the impact the intermittent faults could be the use of spare cores so that whenever a core is affected by an error that core is disconnected and the spare core is connected in its place.

![Diagram of Use of spare cores](image)

Figure 13: Use of spare cores [7]

The problems with this method is the extra overhead on application to manage these cores and also not using cores in case of fault free execution on the other hand this technique has the limitation to handle the no of faults limited to the no of spare cores, so at any given time when your system is hit by concurrent faults which exceed the no of spare cores then there is no other mechanism specified to handle these scenarios.

Operating system (OS) reconfiguration is another method which can be used in which whenever any core is affected by a fault you reconfigure your OS to use only fault free cores.

![Diagram of OS reconfiguration technique](image)

Figure 14: OS reconfiguration technique [7]

But OS can take up to several milliseconds to reconfigure itself which can cause high overhead for short duration failures and in highly critical system in which OS is responsible for scheduling threads and maintaining fairness and achieving low latency rate it can result in degradation of your system performance.

Overcommitted is the technique which used to overcome the drawbacks of all previously discussed mechanisms, in which a hardware/software layer is used which virtualizes cores from the system software and system software is not allowed to directly control the physical cores implemented on the chip and rather it controls the virtual processors (VCPUs) which are exposed to it and inside these VCPUs you can implement the technique to either pause a core or to switch to spare cores in case of fault without notifying the system software. In this method two or more VCPU must share one physical core during a fault and also these VCPUs are frequently switched to avoid generic pause technique, as shown in Fig 15, V2 and V3 are sharing a physical core C3 and if C2 gets error probe it would be suspended and then...
on V2 will use C3 while it is not being used by V3 and they can switch whenever V3 also require C3.

![Diagram of SW Threads, Guest VMs, Virtual Processors, ISA, Physical Cores]

**Figure 15: Overcommitted technique** [7]

The most significant factor is to distinguish between permanent and temporary faults, one of the most commonly used mechanism is named as Alpha (α) Count in which we count the number of faults occurring in the system and if within a given time period the number of errors (alpha state) exceeds the already set threshold value (αT) then that component is considered to be permanently damaged and need to be replaced this method is briefly discussed in [6].

### Network on Chip (NoC)

Number of computing resources on a single chip has remarkably increased and now we have resources like DSP, CPU and IP (Intellectual Property) blocks being used to build system on a chip (SoC), but on the other hand intercommunication between these components is also another major concern, normally in SoC a shared bus interconnection approach is use which requires an arbitrator to serialize bus access requests in order to communicate amongst the components. so whenever a component requires the bus he send his request to the arbitrator and if allowed by arbitrator only then that component can use the bus to communicate with other components but this approach has some scalability issues when the number of components attached to the bus increases their interconnection bandwidth also increases and for simple bus based architectures it becomes increasingly difficult to efficiently handle these sort of ever increasing bandwidth requirements for example in many application in the area of multimedia communication and processing requires multi cores SoC with high communication bandwidth for intercommunication among cores.

Another major issue is the wiring delay, which is growing exponentially due to increased capacitance caused by narrow channel width and also the bus based architecture which is quite commonly used in SoC, when number of components attached to the bus increase physical capacitance of the attached wire also increases which further increase the wiring delays.

In order to efficiently solve these problems Network on Chip (NoC) is emerging as the most feasible solution which can easily be scaled from dozen of resources to hundred and thousands of resources and uses network like interconnection amongst the components and guarantees incredible performance increase as compare to traditional approaches. NoC
ensures fast, reliable, energy efficient communication between the components present on the chip. The crossroad for SoC scalability and multi core chips leads to NoC.

![Crossroad for large SoC and multi processor chips](image)

**Figure 16: Crossroad for large SoC and multi processor chips [2]**

NoC consists of switches and each switch contains a single resource which can be memory, processor core, FPGA or any other IP block, which fits in to the slot of a switch and each switch is connected to its neighboring switches.

![NoC architecture](image)

**Figure 17: NoC architecture [9]**

Each resource can communicate with other resource by sending and receiving messages by the means of their respective switches “S” which is connected to four neighboring switches through input and output channels, each switch is reasonable for routing and buffering message between resources and each switch can have internal queues to handle overcrowding or blocking of messages and their respective channel consists of two one directional point to point buses between either two switches or a resource and a switch.
Resources having their own unique address are connected to network using switches and they communicate with switches using Resource Network Interface (RNI) which consists of four layers: Physical layer, Data-link layer, Network layer and Transport layer.

In Physical layer we determine how many and how long wires are required to connect each resource to its switch.

In Data-link layer protocol to transmit information between either a resource to switch or between two switches is defined, it takes the responsibility to reliably transmit information among different nodes and also takes care of error correction and error detection.

Network layer defines how to transmit packet over the network from sender to receiver using receiver’s network address, its responsibilities also includes buffering of packets and taking routing decisions while passing through intermediate switches and deciding which path would be the most suitable one to transmit this packet.

At Transport layer hold the responsibility of establishing end to end connections and packing transport level message into network level packets.

![Figure 18: NoC Communication Layers](image)

**Fault Tolerance**

Due to the shrinkage of size in each technology generation different sort of permanent, transient and intermittent faults are affecting the reliability of the system, so providing resilience to these factors is the first and most important part for NoC based system otherwise they will cause degradation in the Quality of Service (QoS (Collection of requirements on the NoC performance in terms of achievable throughput, latency, power dissipation and reliability)) and can eventually result in failure of the whole NoC based system.
Due to low area resources available on the chip and emphasis on low latency communication, traditional fault tolerance algorithms (multipath routing etc.) are infeasible for NoC based system due to their large area and storage overheads.

Error detection and error correction codes are used to protect system from transient errors which occur during communication either at the network level using end to end flow control or at the data link level using switch to switch flow control, we can take the example of simple retransmission schemes in which sender adds error detection codes to the data which is to be transmitted and the receiver checks the data for correctness or and if there is any error found in the data in sends request to the sender to retransmit the data but as these schemes have their own power, performance and reliability tradeoffs so depending upon the requirements of your application you can choose between anyone of them of any hybrid implementation of these approaches.

Also complex error detection and error correction schemes can be used but they have their additional area and energy overhead to your system which can result in effect the latency and throughput of your system.

In case of NoC due the presence of information redundancy different fault tolerance strategies are required to be implemented like stochastic communication, adaptive routing, deflection routing, probabilistic broadcast, random walk etc. are being used to achieve fault tolerance.

But our area of focus would be to define standard metrics for fault tolerance because ultimately we have to analyze each fault tolerance implementation in the context of performance specification so that if any of the fault tolerance implementation does not satisfy the requirements of Our NoC infrastructure in terms of performance specifications they can be eliminated at the very early stages of our system design.

**Fault Tolerance Metrics**

Fault tolerance metrics allows us to measure the system design in terms of its reliability, availability, safety, perform ability, maintainability and dependability against its failure rate, mean time between failure (MTBF) and mean time to repair (MTTR).

Failure rate is the expected number of failures a system can suffer during a specific time period which is extremely important aspect of the system analysis and mean time to failure is the estimated time till which the system will keep on functioning correctly before the first error occurs and MTBF is the average time to next failure.

\[
MTBF = \frac{Total\ Operating\ time}{Number\ of\ failures\ encountered}
\]
MTTR is the average time required to repair the system which is calculated experimentally by injecting a set of errors to the system one at a time and checking the time required repairing the system in each case and then we can take the average of those repair times to get the MTTR.

\[
MTTR = \frac{\text{Time Spent for repair}}{\text{Number of repairs}}
\]

Hence the system availability can be used to calculate the impact of these failures on the system.

\[
\text{Availability} = \frac{MTBF}{MTBF + MTTR} \cdot 100\%
\]

But in case of NoC these metrics are not that much effective because some of their properties like MTBF in case of high failure rate does misrepresents the properties of a NoC system to rapid recovery in case of a failure making the impact of failures to be absolutely minimal so that application does not get affected by those failures at all.

Secondly In NoC to guarantee tight QoS in case of a failure we have to met performance constraints like latency and throughput for all possible instance as in these stranded metrics we use average values by using which we can’t guarantee the performance constraints for all instances.

**Fault Tolerance Metrics for NoC**

As we have seen the traditional fault tolerance metrics does not satisfy different NoC architectures in very effective and comprehensive manner so we need to define new set for metrics for NoC which should be easy to implement and readily available so that we can easily and comprehensively analyze the effectiveness of each fault tolerance method in the context of specific NoC implementation so these metrics will allows us to get insight knowledge of actual performance of fault tolerance implementations so that when we have strict QoS requirements during our design process we can easily select which of the fault tolerance mechanism would the most suitable one in the context of our specific NoC implementation.

These metrics allows use to use different combinations of fault tolerance mechanisms at different layer (Physical, Data link etc.) of NoC implementation giving us so many options to combine different fault tolerance mechanisms to different NoC layers which gives us more understanding on how a specific fault tolerance implementation will have a significant impact on the performance and operation of NoC subsystem.
Since communication in a NoC based is done in hierarchal fashion starting from physical layer up to transport layer so at each level faults can be categorized by their respective type, their source, how frequently they occur and their significant impact on the system, so the most effective method is to detect and recover the fault at the lowest level (physical level) if we are not able to do so then the fault prorogates to the next level and then you have to deal with it at the next level using appropriate fault tolerance techniques with respect to cost and area effectiveness.

So in general, higher in the hierarchy you have to deal with the fault the more time it takes to detect and to recover from that fault.

For a comprehensive approach to fault tolerant design the key element of NoC metrics are

- Avoidance
- Detection
- Containment
- Isolation
- Recovery

Figure 19: Fault Detection or repair at different Hierarchal levels
Avoidance

The goal of the avoidance is to reduce the chances for an error to occur by either employing information redundancy in which you add extra information which is beyond what is essentially required to perform a function just like using error correcting and error detecting codes in NoC in which you add either parity information or error correcting codes with your data which is to be transmitted so that if the receiver finds any error he can either send a retransmission request or can himself correct the data using error correcting code or by employing hardware redundancy or avoidance can also be achieved by adding some extra hardware in order to detect or to tolerate the faults or by using time redundancy in which you use additional time to perform a function in side you system so that if any fault occurs it can easily detected or tolerated.

Redundancy can have a very big impact on your system performance and power consumption as the time required by these mechanisms to perform their operation would be added to the total latency of data being transported across the NoC so taking into account the NoC infrastructure in which the either at switch-switch or end to end avoidance implementation is provided the time overhead of switch-switch would be much bigger than end –end avoidance mechanism. So with respect to the QoS requirements of your design you can choose which of the available fault tolerance technique would be most suitable for your system.

Detection

In this phase of fault tolerant design we have to detect the presence of faults in our system which were caused because either there was no avoidance mechanism was implemented by
the system or the correction mechanisms failed to correct them, so early detection of these faults is really necessary so that if the detection mechanism assesses the nature of the fault to be fatal which can’t be corrected then recovery mechanisms are required to be initiated so quicker the fatal error is detected quicker the recovery mechanism can be initiated normally error detection is an integral part of most of the error correcting codes which can easily detect the amount of uncorrected faults present in your system.

In NoC based systems fault detection can performed using either on end-end approach so that whenever the information is received it can be checked for errors on each end but this approach has its own drawbacks because of higher detection latency (amount of time between moment the fault occurs and the moment it is detected) as in this case an error can only be detected only when it reaches the other end as compare to switch-switch approach in which error detection is performed at each switch so we can very quickly find that there is an error present in the data which is transmitted either from the previous switch or at the link which connects two switches.

So as switch-switch error detection approach is much more effective than end–end approach so according to error latency requirements of your system’s design you can choose between anyone of them to provide fault tolerance.

![Figure 21: Different Error detection techniques](image)

**Containment**

Fault containment relates to limiting the impacts of faults to specific boundaries so that problems caused by those faults do not propagate outside that region. So if your system is hit by a fault then we would try to contain the impact of the errors caused by those faults so that they do not propagate affecting the whole system by finding Fault containment regions (FCR) inside your NoC system.
FCR inside the NoC should be independent of each other so that if a fault occurs on one of the containment region it does not affect the other one. The advantage of finding independent FCRs is that if fault is detected on one of the regions we can provide communication between the resources using the alternative FCR so that there is a danger that if we would keep on using faulty region the error can propagate to the whole system and can result in whole system crash/failure.

Isolation

Isolation means to eliminate the use of those processing elements of your system because of which error is being generated; so that when they would not used any more the risk of error caused by those faulty processing elements and their propagation to higher layers would be minimal.

At physical layer in case of permanent fault isolation can be done by disconnecting the faulty NoC components and the propagation of errors caused by those components to application layer can be restricted by dropping those data packets which contain error so that they does not interfere with rest of the data.

For example if you detect an error in your system and want to know the location of component because of which this error is being generated you can accomplish this by sending data though all containment regions and validating the output coming form each containment region if you find that out there is one of the region though which whenever data is transferred it gets faulty so you can find the fault component by applying switch-switch error detection technique on that path to find the fault component in your system and then on you will avoid the use of that until it is recovered.
Recovery

Whenever a system is hit by an error the most important aspect is to provide means to recover from that failure, time performed by the recovery process is the most important factor for evaluating the effectiveness of any fault tolerance scheme.

Area and power consumption along side for the recovery process are them most important factors to be considered while selecting any of the recovery method for your system, if you QoS specification does not bound you to any time restriction then late recovery method resulting in successful recovery can be accommodated and vice versa.

So in NoC error free communication between two resources can be accomplished by providing some hardware correction at lower (physical) layer or at the higher layer it can done be sending retransmission requests is send by the receiver to the sender to send an error free copy of the data, in case if it detects an error in the data.

Summary

Wee have seen that with every new generation the problems caused by soft and transient errors along side random dopant fluctuations, sub wavelength lithography, heat flux and degradation of transistor performance are the biggest threat to system reliability, in order to make your system more reliable fault tolerance must be provided to either avoid their occurrence or to reduce their impact.

Traditional fault tolerance metrics were used to measure the system design in terms of its reliability, availability, safety, perform ability, maintainability and dependability against its failure rate, MTBF (Mean time between failure) and MTTR (Mean time to recover) but they are not suitable for the Network on chips (NoC) architecture because of their limitations so new set of metrics (avoidance, detection, containment, isolation, recovery )are defined to comprehensively analyze the effectiveness of each fault tolerance method in context of specific NoC implementation and by using combination of different fault tolerance methods at different communication layers helps us to categorize them by their respective type, source, frequency of occurrence and their significant impact on the system and by doing so we can easily choose among them keeping in view our QoS (Quality of service) requirements and also by applying combination of different fault tolerance methods at different layers impact caused by those errors can be significantly reduced.
References


