International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

Designing Reliable Embedded Systems

Mr.B.Ravi [1],Ms.AVP Sarvari [2],
Assistant Professor [1,2],
Department of ECE
Mail Id : bravisrkit@gmail.com, Mail id : a.v.p.sarvari@gmail.com,

Abstract:

In this research, we combine hardware and software fault resistance techniques to provide a new addition to the idea fault-tolerant embedded systems design. Obtaining highest fault tolerance at little cost requires an effective intermediary between hardware crystallisation and system reimplementation in software. This study presents a strategy for developing such a fault-tolerant system, with maximal hardware crystallisation and many software-level execution attempts at minimal cost. All the requirements for a flawless, fault-tolerant system must be met, thus it's important that this one has the highest dependability possible. Time redundancy refers to the practise of re-executing an operation in the event of a failure, such as transitory software errors.

Key words

Fault tolerance, embedded system, fault types, and related terms are discussed.

Introduction

It's possible for a system to fail if it doesn't provide as promised. This breakdown is due to faulty components in the system. Fault-tolerant systems are those that continue functioning as expected despite the presence of mistakes. As it is a fault tolerant system, the system never fails [3]. Applications in the medical, aviation, finance, and engineering fields might benefit greatly from a fault-tolerant system. Redundancy is the key to a fault-tolerant system. Despite malfunctions, a system with redundant parts or components may continue functioning for a longer period of time.

If a system has a backup component, it may continue functioning normally even if a critical component is destroyed. Distinguishing Features What constitutes acceptable performance when a defect is present was determined by your system-level need. Regarding the fault tolerant system, "dependability" is the primary strategy. The following categories will serve to define this. Several categories apply to this situation:

Fault-tolerance is the opposite of fault-tolerance, so what's the difference? System Fault intolerance is a technique for avoiding problems and might be called fault avoidance. This strategy of problem avoidance or fault intolerance may improve the dependability of a system. Before a system's performance at the hardware or software level, the reliability concept eliminates any potential sources of error that may be present in the system [9]. Before discussing the Embedded System Architecture [1], we need to introduce various strategies and procedures with relation to the failures. When the expected behaviour of an element differs from the actual behaviour of that element, we say that element is defective. For theses, we provide two distinct types of poor execution. This author demonstrated in 1982 that irregular performance may occur because of malfunctioning wiring.

In the case of Byzantine Failures, the circuitry in question may be complicit with other bad circuitry in the system or exhibit unpredictable and problematic behaviour on its own. [19]. Schneider had earlier in 1984 presented his ideas in Fail-stop Failures: As soon as the circuitry fails, the element or portion of the system transitions to a state that enables another component of the system to recognise the fault existing in the system, therefore halting the operation before the entire system fails. [21]. When a circuit fails, a Byzantine failure isn't the best fix. This may not always be the case, but it is a dangerous solution for the circuit run. In addition, Byzantine fail-stop failures are used in many different kinds of applications. In 1982, Sitework and Swartz

International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

claimed that a circuit with several components was the most dependable kind of circuit. Due to the unique properties of its individual parts, the system may achieve peak performance in the Maximum interval of the circuit when run. The term "fault tolerant system" describes this setup. Mean time between failures is a common way to define fault tolerance (MTBF). That symbol represents the failure probability over discrete intervals, among other analytic procedures. [22].

Model of Defects

Fault Categories A transient fault is a fault that occurs just once and then disappears after a few seconds. In other words, the flaw faded away as a result of external factors. When a problem occurs, it may disappear for a while and then reappear at a later time without taking on a consistent pattern. This is the worst form of error since it is so hard to track down. The presence of weak connections in the circuitry may cause this issue to manifest. If a permanent issue develops, the only way to restore the system to peak performance is to replace the affected parts. If this sort of issue arises, the only way to restore normal operation to the hampered system is to replace or repair the offending component. This means the problem will remain until the faulty part is either replaced or uninstalled. One such source of error isErrors in the physical hardware are also considered hardware faults. Components at the hardware level include the processor's communication connection, a switch, and so on.

Problematic software:

flaws in the programme itself are included. Design diversity is an approach that utilises both hardware and software fault tolerance together. This method is used to build computers with several layers of redundancy across multiple communication channels. Each channel is meant to meet certain requirements, and this method may identify a broken channel if it doesn't. The primary goal is to provide a system that can recover from errors both in software and hardware.

Redundancy:

Time duplication:

A time redundancy system is one in which the code is designed to run repeatedly until the malfunctioning circuit can no longer be fixed.

Parallel hardware architecture:

The redundancy of a system may be increased with the addition of additional hardware circuitry, as shown in hardware redundancy.

Multiple copies of the programme:

Software redundancy relies on having many copies of the same programme available so that if one fails, the other copies may continue to run the system normally.

The problem of redundant information: When a mistake is recognised in the system's bits, it may be remedied with the help of the identical data codes provided by information redundancy. These come in a wide variety of forms, such as parity coding, checksum coding, and cyclic coding.

Recovering the Machine

The most important part of a fault-tolerant architecture is a computer system that can recover by itself after a malfunction. It can self-repair in the event of random system failures. Idle (static):

International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

Fault recovery may be masked using passive recovery. For this purpose, the system doesn't have to make any further efforts.

Movement or change:

Active recovery's central idea is the comparison of findings for defect identification. By doing so, it eliminates the problematic parts of the system immediately.

Hybrid:

The system combines active and passive methods until full detection is achieved. Method for building a system fault-tolerant is more costly but more dependable.In order to achieve redundancy in hardware,adding additional hardware components or elements to a system is the essence of hardware redundancy.

Troubleshooting, repair, and concealment:

In a parallel session, many pieces of hardware are used to do the same work, with the results being compared at session's conclusion so that errors may be found and eliminated.

Detection:

If one portion of a system is flawed while the others are functioning normally, the system as a whole will provide conflicting results.

Using the real fluctuation in results, this method is ideal for identifying system faults.

Adjustment and concealment:

The majority result may be utilised to correct for errors in the system if just a small percentage of the system is flawed. Repair or replacement of broken parts: The system's modest fault tolerance capabilities are implemented mostly via correction and masking. With this, the fault tolerance level is determined by the ability to either restore the real system or replace the damaged components. Errors in the requirements or the blueprints:

Hardware and software levels are also vulnerable to these kinds of blunders.

Failures caused by faulty parts:

Manufacturing flaws are to blame for the faulty hardware being used.

A number of environmental factors, including extremes in temperature, radiation, vibration, and humidity, may contribute to hardware failures in an operating environment.

EMBEDDED SYSTEM ARCHITECTURE

International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

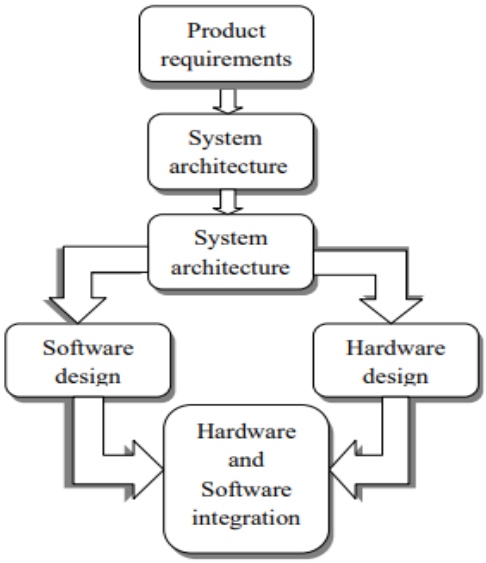

Figure1 Building Embedded Systems

The design process for VLSI embedded systems may be blended to create fault-tolerant designs.

Techniques for Error-Tolerant Architecture in Embedded Systems

A. The software design of the system includes both problem diagnostics and fault tolerance.

B."The Software Architecture," Re-execution is a key part of the development cycle. If a problem is uncovered in a system, the procedure will be redone from the beginning.

C. The re-execution procedure brings back all the original inputs to fix the problem.

D.The following are two methods we use for fault tolerance in software, which we've categorised as method.

E. Active replication with checkpointing for rollback recovery.

A. Recovering from a Previous State with Checkpoints Time spent re-executing the system in search of an error is lowered significantly by using the rollback recovery method. Finding the best and last error-free state in the execution and reverting to that point is the foundation of this method, which aims to prevent the system as a whole from failing. The system's last known good state, or check point, must be stored in static memory for redundancy in case of failure; this allows the system to be restored to its previous state, before the breakdown occurred.

B. Dynamic and Static Replication The primary issue with recovery techniques is that they can only be used to evaluate a small subset of redundant computing nodes. Other approaches that may deal with duplicate scope include rollback recovery, re-execution, active and passive replication, and, at its core, the provision of separate, parallel sessions of the system in which to perform the necessary fault recovery.

C. Openness Only by using a dynamic approach can transient defects be corrected and a system breakdown avoided. The amount of temporary failures allowed in the system is directly correlated to its ability to function. All of the circuit's processing requirements must be accounted for whether making corrections, running tests, or making adjustments. Therefore, it becomes very difficult to do debugging, verification, and testing. Transparency is the key to solving this problem.

International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

A. Requirements for openness in the product market

B. The Framework C. The Software Design D. The Hardware Structure

E. Combining the capabilities of hardware and software.

Dependability Organization of Ideas.

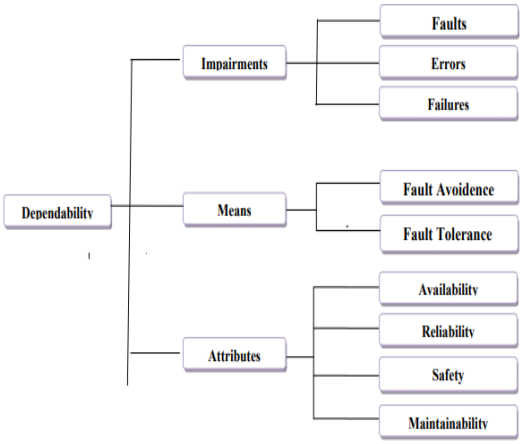

Figure2: Dependability flow chart

Conclusions

The single stuck-at fault model is the clearest example of the benefits of tools and experience working together. Many additional types of faults, such as multiple stuck, stuck-open, and bridging, may be tested for using a stuck-at fault. Delay faults, stuck-short faults, and technology dependant errors all call for their own unique battery of diagnostics tests. Analog and memory circuits, which are particularly susceptible to failure, may be analysed using a number of different kinds of test models. When built for particular test cases, a system may be fault tolerant. It is necessary to update the system's methods of testing and diagnosing faults to keep up with the rapid development of the circuits. Because of the system's hardware and software, there are a wide range of possible circuit tests. In order to create a fault-tolerant circuit, engineers must combine hardware and software, a technique known as embedded system architecture. One new way to manage fault-tolerant systems is to use stuck-at-fault models. Research into fault tolerance is already widespread across many different industries, including control system design, transportation, electronic commerce, space exploration, communications, and many more.

REFERENCE:

[1] F. B. Schneider, "Implementing Fault-Tolerant services using the state machine approach: A tutorial", ACM Computing Surveys, vol 22, No. 4, December 1990

[2] F. B. Schneider, "The State Machine Approach: A tutorial", pp 18-41, Springer, 1990.

[3] David A. Rennels, "Fault -tolerant computing :encyclopedia of computer science" pp 698-702, John Wiley and Sons Ltd. Chichester, UK

[4] Krishnendu Chakrabarty ," VLSI System Testing: A Book," Section 4.4 (pp. 60-70), Spring 2011-2015

[5] A. D. Singh "Interstitial Redundancy: An Area-Efficient Fault-Tolerance Scheme for Larger Area VLSI Processor Arrays", IEEE Trans. Computers, vol. 37, no. 11, pp.1,398 -1,410 1988 .

International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

[6] Anderson, T., and R. Kerr, "Recovery Blocks in Action: A System Supporting High Reliability", Proceedings of the Second International Conference on Software Engineering, 1976, 447-457.

[7] Avizienis, A. and J.P.J. Kelly, "Fault Tolerance by Design Diversity: Concepts and Experiments", IEEE Computer, vol. 17 no. 8, August 1984, 67-80.

[8] Avizienis, A., et al., (Ed.). Dependable Computing and Fault-Tolerant Systems Vol. 1: The Evolution of Fault-Tolerant Computing, Vienna: Springer-Verlag. (Though some what dated, the best historical reference available.)

[9] Wenbing Zhao, "Application-Aware Byzantine Fault Tolerance Dependable", Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference, Year: 2014,pp. 45-50.

[10] A. Avi˘zienis and J.-C. Laprie. Dependable computing: From concepts to design diversity. Proceedings of the IEEE, 74(5), 1986.

[11] A. Azagury, D. Dolev, G. Goft, J. Marberg, and J. Satran. Highly available cluster: A case study. In1994 IEEE 24th International Symposium On Fault-Tolerant Computing, pages 404–413, 1994.

[12] W. G. Cuan. Fault tolerance: Tutorial and implementations. Computing Futures (supplement to IEEE Computer November 1989), 22(Inaugural issue), 1989.

[13] Y. Huang and C. Kintala. Software implemented fault tolerance: Technologies and experience. In 1993 IEEE 23th International Symposium On Fault-Tolerant Computing, pages 2–9, 1993.

[14] Y. Huang and C. Kintala. Software implemented fault tolerance: Technologies and experience. In 1993 IEEE 23th International Symposium On Fault-Tolerant Computing, pages 2–9, 1993.

[15] N. Kandasamy, J. P. Hayes, B. T. Murray, "Transparent Recoveryf rom Intermittent Faults in Time-Triggered Distributed Systems", IEEE Trans. on Computers, 52(2), 113-125, 2003.

[16] S. Punnekkat, A. Burns, R. Davis, "Analysis of Check pointing for Real-Time Systems", Real-Time Systems Journal, 20(1), 83-102, 2001. [17] Ying Zhang and K. Chakrabarty, "A Unified Approach for Fault Tolerance and Dynamic Power Management in Fixed-Priority RealTime Embedded Systems", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 25(1), 111-125, 2006

[18] A. M. Álvarez, Felipe Restrepo-Calle, Luis Alberto Vivas Tejuelo, Sergio Cuenca-Asensi, Fault tolerant embedded systems design by multi-objective optimization, Expert Systems with Applications, Volume 40, Issue 17, 1 December 2013, Pages 6813-6822.

[19] Leslie Lamport and S. Merz,"Specifying and Verifying Fault-tolerant System", Proceedings of the Third International Symposium on Formal Techniques in Real Time and Fault Tolerant Systems.

[20] Viacheslav Izosimo, V., "Analysis and Optimization of Fault-Tolerant Embedded Systems with Hardened Processors", Design, Automation & Test in Europe Conference & Exhibition, 2009.

[21] Schneider , "Implementing Fault-Tolerant Service using the state machine", ACM Computing Surveys, Vol. 22, No. 4, December 1990.

[22] Siewiorek and R.swarz ,"Reliable computer system : Design and Evaluation" MA : Digital Press, c1982.

[23] A. Sung and B. Choi, "An Interaction Testing Technique between Hardware and Software in Embedded Systems", Proceedings of Ninth Asia-Pacific Software Engineering Conference, 2002. 4-6 Dec. 2002 Page(s):457 – 464.