Capítulo 2

Titulo por definir

«La verdadera ciencia enseña, sobre todo, a dudar y a ser ignorante.»

Ernest Rutherford

Resumen: En este capítulo se define con detalle lo que es un procesador y su importancia en el mundo hoy en día. También se tratan dos arquitectura más concretas, la arquitectura DLX y la arquitectura ARM.

A continuación se define qué es un fallo y qué tipos de fallos pueden ocurrir en los sistemas. Además se explican algunas técnicas de tolerancia a fallos.

Para terminar se justifica la importancia de la tolerancia en los sistemas y concretamente porque es necesaria la tolerancia en los microprocesadores.

2.1. Tolerancia a Fallos

La tolerancia a fallos se define como la capacidad de un sistema de funcionar correctamente incluso si se produce un fallo o anomalía en el sistema.

En ocasiones se producen fallos que no llegan a propagarse por el sitema y no producen errores en su funcionamiento, esto ocurre cuando los cambios realizados por los fallos se ven enmascarados. Puede deberse a alguna de las siguientes razones:

- **Enmascarado lógico**

  Se evita el error en una puerta lógica, gracias a que el valor del dato no es necesario para estimar la salida. En la figura 2.1 vemos que el valor de la señal invertida es indiferente para calcular el resultado ya que el

resultado de una puerta «or» es «1» siempre que una de sus entradas sea «1».

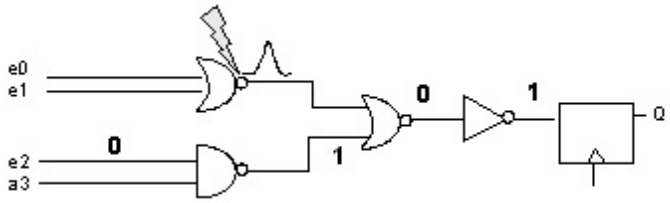

Figura 2.1: Fallo enmascarado por una puerta lógica.

- **Enmascarado eléctrico**

  El fallo producido pierde intensidad en el recorrido lógico y no tiene efecto al llegar al elemento de memoria que lo almacenaría. Figura 2.2.

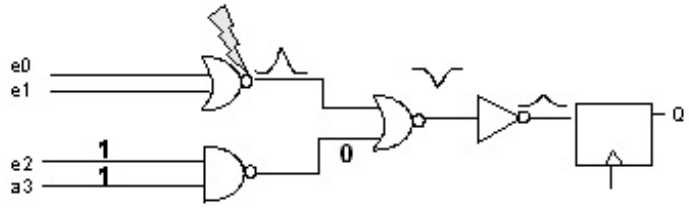

Figura 2.2: Fallo enmascarado eléctricamente.

- **Enmascarado temporal**

  El fallo se propaga con suficiente energía hasta el biestable, sin embargo, ocurre fuera de la de la ventana crítica de tiempo y la señal puede estabilizarse a su valor correcto antes de almacenarse en el biestable Figura 2.3.

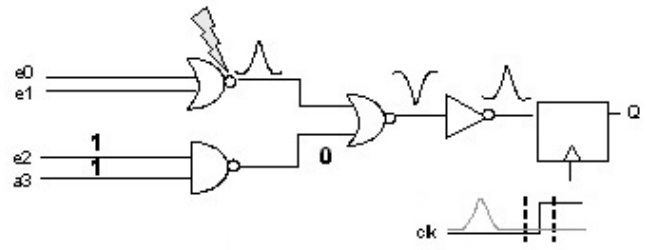

Figura 2.3: Fallo enmascarado por ventana de tiempo.

Hay diferentes grados de tolerancia que dependen de la aplicación del sistema:

- **Tolerancia completa (fail operational)**

  El sistema puede seguir funcionando sin perder funcionalidad ni prestaciones.

- **Degradación aceptable (failsoft)**

  El sistema continua funcionando parcialmente hasta la reparación del fallo.

- **Parada segura (failsafe)**

  El sistema se detiene en un estado seguro hasta que se repare el fallo.

La tolerancia a fallos hardware se trata principalmente aplicando la redundancia de uno o varios de los siguientes tipos:

2.1.1. Redundancia en la información

La redundancia de datos se basa en mantener varias copias de todos los datos en diferentes ubicaciones junto a códigos de detección y corrección de errores. La replicación de datos consigue que la perdida o daño de una memoria no implique la perdida de los datos que almacena. Los códigos de detección y corrección permiten comprobar los datos en busca de errores y corregir los datos si fuese necesario.

El ejemplo más claro de este tipo de tolerancia es el conocido como *conjunto redundante de discos independientes* o *redundant array of independent disks (RAID)*. Las diferentes clases de RAID proporcionan un acceso a los datos rápido y transparente para el sistema operativo [Torrecillas]. Por ejemplo:

- **RAID 1:** Se basa en la utilización de discos adicionales sobre los que se realiza una copia de los datos que se están modificando.

- **RAID 5:** Reparte la información en bloques con bits de paridad, que se guardan en diferentes discos.

2.1.2. Redundancia en el tiempo

La redundancia en el tiempo es efectiva contra los fallos transitorios. Esta consiste en ejecutar parte de un programa o el programa completo varias veces. Los fallos transitorios, como se ha explicado anteriormente, se producen en zonas aleatorias del chip, siendo poco probable que aparezca el mismo error en el mismo lugar.

Este tipo de redundancia requiere una menor cantidad de hardware y de software, pero requiere ejecutar varias veces el programa, reduciendo el rendimiento del sistema.

Algunas técnicas de redundancia en el tiempo se basan en «puntos de control» o «checkpoints». Estas técnicas requieren almacenar los datos con los que se está trabajando cada cierto tiempo, creando un «punto de control», esto permite que al detectar un error se pueda recurrir al último «checkpoint» en vez de reiniciar el programa completo Kadav et al. (2013).

2.1.3. Redundancia en el hardware

La redundancia hardware se basa en la inserción de módulos extra para la detección y corrección de los fallos. Aunque su objetivo es el de reducir el número de fallos que provocan errores, la inserción de módulos extra implica un aumento en la complejidad del sistema, con ello aumenta la posible aparición de nuevos fallos.

El tolerancia con hardware redundante se clasifica en:

- **Tolerancia estática:** Los componentes redundantes están siempre activos.
- **Tolerancia dinámica:** Los componentes redundantes se activan cuando se detecta un fallo.
- **Tolerancia híbrida:** Combinan tolerancia estática con tolerancia dinámica.

Algunas técnicas se detallan a continuación Kirrmann2005.

Redundancia modular n-ésima

Redundancia dinámica

Re-configuración dinámica

2.1.4. Tolerancia en microprocesadores

...

Bibliografía

Y así, del mucho leer y del poco dormir,
se le secó el celebro de manera que vino
a perder el juicio.

Miguel de Cervantes Saavedra

Brinkgreve, R., Swolfs, W. y Engin, E. *ARM Architecture Reference Manual Thumb-2 Supplement*. 2011. ISBN 9781597180948.

Habinc, S. Functional Triple Modular Redundancy (FTMR). *Design and Assessment Report, Gaisler Research*, páginas 1–56, 2002.

Hennessy, J. L. y Patterson, D. a. *Computer Architecture, Fourth Edition: A Quantitative Approach*. 0. 2006. ISBN 0123704901.

Hu, A. C. y Zain, S. NSEU Mitigation in Avionics Applications. vol. 1073, páginas 1–12, 2010.

Investigation, A. O. ATSB TRANSPORT SAFETY REPORT Aviation Occurrence Investigation AO-2008-070 Final. (October), 2008.

Kadav, A., Renzelmann, M. J. y Swift, M. M. Fine-grained fault tolerance using device checkpoints. *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems - ASPLOS '13*, página 473, 2013. ISSN 15232867.

Sadasivan, S. An introduction to the arm cortex-m3 processor. 2006.

Torrecillas, J. M. RAID - Tolerancia a Fallos. ????