Capítulo 5

Aplicando tolerancia a fallos

...

...

Resumen: En este capitulo se explica cómo se ha aplicado la tolerancia a fallos en el microprocesador.

5.1. Introducción

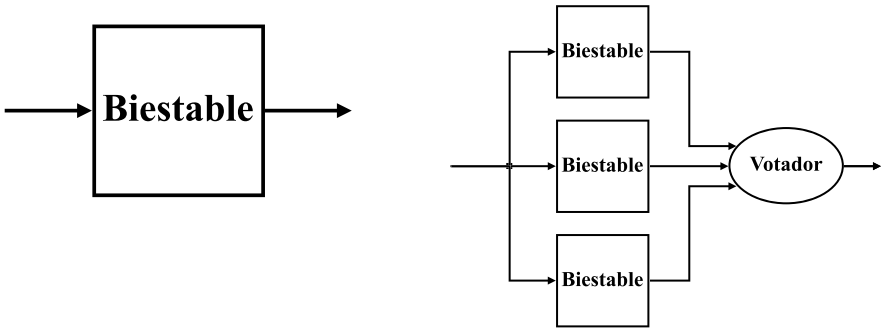
Los fallos más comunes en los sistemas electrónicos son los conocidos como fallos transitorios, estos no dañan el sistema de forma permanente pero provocan un cambio de valor en los elementos de memoria (SEU) o interferencias en las conexiones internas (SET), que si no se estabilizan a tiempo pueden llegar a propagarse hasta una celda de memoria y almacenarse provocando el mismo efecto que un SEU. Con esta información se ha implementado un método para paliar y reducir los efectos tanto de los SEU como de los SET.

5.2. Redundancia modular

La técnica utilizada para paliar los fallos transitorios ha sido la redundancia modular (NMR) con un valor de $N = 3$, conocida como redundancia modular triple (TMR). Consiste en triplicar cada componente de memoria y conectar «votadores» a sus salidas. Los «votadores» permiten enmascarar una cantidad de fallos igual a $\frac{N}{2}$. En nuestro caso significa que el sistema podrá tolerar un fallo en cada conjunto de módulos triplicados.

Para proporcionar al sistema tolerancia frente a los SEU al sistema, se han sustituido todos los biestables por un conjunto compuesto por tres biestables

más un votador. Si el componente inicial era el representado en la figura 5.1a, aplicando lo anterior obtendremos el conjunto representado en la figura 5.1b.



(a) Biestable original. (b) Conjunto de biestables con votador.

Figura 5.1: Sustitución de biestable.

5.3. El Votador

El votador es un circuito combinacional, y su único objetivo es filtrar los valores de entrada, para dar un valor de salida igual al de la mayoría de sus entradas. Esta propiedad hace del votador el componente principal utilizado para otorgar la capacidad de tolerar los fallos transitorios de tipo SEU al sistema.

Como se vé en la tabla 5.1, el valor de salida de un votador es igual al valor que más se repite en sus entradas. Con este método puede enmascararse un fallo en cualquiera de los módulos que alimentan sus entradas.

Entradas			Salida
A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tabla 5.1: Tabla de verdad del votador

La implementación del votador puede realizarse de diferentes formas siempre que cumpla con las restricciones de la tabla 5.1. El votador de este proyecto se ha diseñado siguiendo la fórmula lógica 5.1, utilizando 4 puertas lógicas: 3 puertas «AND» de dos entradas y 1 puerta «OR» de tres entradas. Como se puede observar en la figura 5.2, se introduce únicamente un retardo de 2 puertas lógicas, haciendo que las puertas lógicas «AND» funcionen en paralelo.

$$Z = (A * B) + (B * C) + (A * C) \tag{5.1}$$

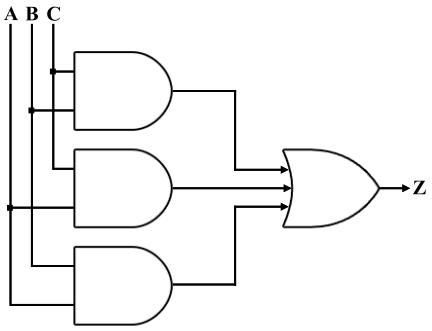


Figura 5.2: Diseño de votador con puertas lógicas

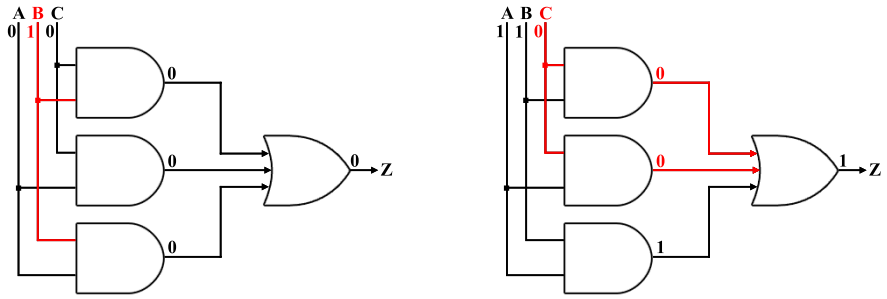
Para exponer cómo funciona el votador se muestran las figuras 5.3a y 5.3b. En la figura de la izquierda observamos cómo la entrada B es diferente al resto, eso quiere decir que el módulo origen de esa señal ha sufrido un fallo. Se observa cómo tal fallo queda enmascarado por las puertas «AND» por su funcionalidad $(0 * 1 = 1 * 0 = 0)$. En el caso de la figura derecha, el fallo ocurre en la entrada C, en este caso queda enmascarado por la puerta «OR»$(0 + 0 + 1 = 1)$.

5.3.1. Implementación

La implementación de este módulo se divide en dos secciones:

- **Registros triplicados**

  Al aplicar la la redundancia tipo TMR a los registros, estos deben triplicarse. Para ello se define una constante «N_Tolerancia» para establecer el número de replicas que se desea implementar, en nuestro caso este número es 3. Después se declara un conjunto de tres registros de un tamaño variable determinado por una variable. Esto facilita utilizar el componente para datos tanto de tamaño 32 como de tamaño 1. A continuación se declara un proceso por el cual se actualizan los tres registros al mismo tiempo y con los mismos datos.



(a) Fallo en entrada B (0 ->1) (b) Fallo en entrada C (1 ->0)

Figura 5.3: Ejemplos de fallos en entradas.

```
constant N_Tolerancia : integer := 3;
type tipo_conjunto_registros is array (0 to N_Tolerancia
    -1) of STD_LOGIC_VECTOR (tamaño-1 downto 0);
signal registros : tipo_conjunto_registros;

[...]

p_regs: process (clk, rst)
begin
   if rst='0' then
    for i in 0 to N_Tolerancia-1 loop
            registros(i) <= (others=>'0');
       end loop;
   elsif rising_edge(clk) then
       for i in 0 to N_Tolerancia-1 loop
            registros(i) <= dato_entrada;
       end loop;
   end if;
end process;
```

- **Votador**

  Aplicando la fórmula 5.1 a los registros triplicados, el votador obtiene el valor con un mayor número de repeticiones.

```
dato_salida <= (registros(0) and registros(1)) or
               (registros(0) and registros(2)) or
               (registros(1) and registros(2));
```

5.4. Aplicación

La segmentación del procesador requiere una cantidad importante de biestables para almacenar todos los datos que deben transmitirse de una

etapa a la siguiente. Debido al enorme aumento del número de lugares críticos donde puede ocurrir un SEU, se ha visto necesario aplicar la técnica TMR especialmente en esta parte del microprocesador.

Esta técnica se ha aplicado a todos los registros de control para evitar que un fallo no controlado produzca un error en medio de la ejecución de una instrucción en cualquiera de sus etapas. Por ejemplo: cambiando la dirección de un salto, cambiando el dato que debe almacenarse en memoria o habilitando la escritura en registro cuando no debería almacenarse el resultado de la instrucción.

Sin incluir las memorias de datos e instrucciones que normalmente son externas, el componente «registros de control» es el mayor elemento de memoria dentro del microprocesador. Un fallo en este componente altera de forma imprevisible la ejecución de las instrucciones que estén en ejecución dentro del procesador en ese momento, evitar este problema es el objetivo de este trabajo y por ello nos centraremos en modificar este componente para ofrecer un mayor grado de confiabilidad. En su caso, también se podría aplicar este método a otros componentes como el banco de registros y el registro «contador de programa».

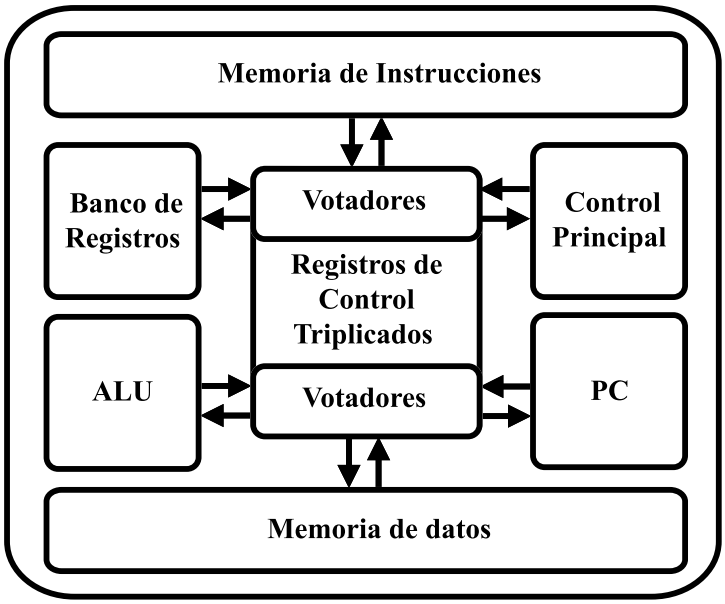


Figura 5.4: Procesador con TMR en los registros de control.

El resultado de aplicar este método sobre los registros de control se puede observar en la figura 5.4. Obtenemos un componente de mayor tamaño, pero que proporciona una mayor confiabilidad en que las instrucciones se ejecutarán correctamente.

El resultado es un procesador que ocupa un mayor porcentaje del área ocupada en la FPGA, pero que a cambio proporciona una mayor fiabilidad.

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

[1] R. Brinkgreve, W. Swolfs, and E. Engin. *ARM Architecture Reference Manual Thumb-2 Supplement.* 2011.

[2] S. Brown and J. Rose. Architecture of FPGAs and CPLDs: A tutorial. *IEEE Design and Test of Computers*, 13(2):42–57, 1996.

[3] C. T. Bustillos. Simulador arm en el ámbito docente. 2012.

[4] I. N. de Estadistica. Penetración de ordenador en hogares. 2014.

[5] S. Flash. Nexys4 FPGA Board Reference Manual Ethernet connector. pages 1–29, 2013.

[6] J. Gaisler. A portable and fault-tolerant microprocessor based on the SPARC V8 architecture. *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 409–415, 2002.

[7] J. C. González Salas. *Filtro adaptativo tolerante a fallos.* PhD thesis, 2014.

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

[9] J. L. Hennessy and D. A. Patterson. *Arquitectura de Computadores: Un enfoque cuantitativo.* Mcgraw Hill Editorial, 1993.

[10] J. L. Hennessy and D. a. Patterson. *Computer Architecture, Fourth Edition: A Quantitative Approach.* Number 0. 2006.

[11] A. C. Hu and S. Zain. NSEU Mitigation in Avionics Applications. 1073:1–12, 2010.

[12] O. Ieee-std. LEON3 7-Stage Integer Pipeline. (March), 2010.

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

[14] Jedec. Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Error in Semiconductor Devices: JESD89A. *JEDEC Sold State Technology Association*, pages 1–85, 2006.

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

[16] H. Kirrmann. Fault Tolerant Computing in Industrial Automation. *Lecture notes ABB Corporate ResearchETH*, 2005.

[17] I. Kuon, R. Tessier, and J. Rose. FPGA Architecture: Survey and Challenges. *Foundations and TrendsÂ® in Electronic Design Automation*, 2(2):135–253, 2007.

[18] A. R. M. Limited. ARM7TDMI-S. (Rev 3), 2000.

[19] a. R. M. Limited. ARM Architecture Reference Manual. pages 1–1138, 2007.

[20] W. K. Melis. *Reconstruction of High-energy Neutrino-induced Particle Showers in KM3NeT .* PhD thesis, 2014.

[21] C. Mobile. Streaming 4K Ultra HD video at home and on the go. pages 0–1.

[22] J. Rose, A. E. Gamal, and A. Sangiovanni-Vincentelli. Architecture of Field-Programmable Gate Arrays.

[23] E. Rotenberg. AR-SMT: a microarchitectural approach to fault tolerance in microprocessors. *Digest of Papers. Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (Cat. No.99CB36352)*, 1999.

[24] D. J. Sorin and S. Ozev. Fault Tolerant Microprocessors for Space Missions. *Memory*, pages 1–4.

[25] U. States. Reduce Cost and Board Space. 374:1–8, 2011.

[26] I. S. Summary, T. C. Field, M. Long, S. D. Transfer, U. Instruction, and I. S. Examples. ARM Instruction Set. pages 1–60.

[27] J. M. Torrecillas. RAID - Tolerancia a Fallos.

[28] C. Weaver and T. Austin. A fault tolerant approach to microprocessor design. *Proceedings of the International Conference on Dependable Systems and Networks*, (July):411–420, 2001.

[29] Xilinx. Xilinx Artix-7 Fpgas: a New Performance Standard for Power-Limited, Cost-Sensitive Markets.