Capítulo 7

Análisis de los resultados

...

...

Resumen: En este capitulo se analizan los resultados expuestos en el capítulo anterior.

7.1. Introducción

El presente capítulo se inicia comparando las implementaciones de ambos procesadores, estándar y tolerante a fallos, exponiendo sus principales diferencias. Posteriormente se analizan y comparan los resultados obtenidos al ejecutar las simulaciones sobre las dos implementaciones del microprocesador.

7.2. Comparativa de procesadores

El procesador estándar requiere un total de 1,952 registros para ser utilizados como biestables, un lo que supone aproximadamente el 1,5 del total disponible. De los utilizados 420 se dedican a los registros de control y el resto se utilizan para el banco de registros así como para las memorias de instrucciones y de datos.

En cuanto a la lógica combinacional del microprocesador se utilizan un total de 1,015 LUTs para los elementos como la ALU, el control principal y los sumadores entre otros. Requiere un 1 % de las LUTs disponibles en la FPGA.

Los informes de restricción de tiempo indican que el diseño requiere un periodo mínimo de reloj de 6.435 ns, alcanzando una frecuencia máxima de 155.400 MHz.

El procesador tolerante a fallos se deriva del procesador estándar aplicando la técnica TMR sobre los registros de control entre etapas.

Al incluir una mayor lógica, el procesador tolerante a fallos requiere de 2,798 registros (poco más del 2 %) de los cuales 1,260 se utilizan para los registros de control. Como se ha diseñado el procesador aplicando la técnica TMR, este es el resultado esperado.

La lógica combinacional también se ve afectada ya que se insertan una gran cantidad de votadores, uno por cada 3 biestables de los registros de control. En total se utilizan 1,464 LUTs (2 %), 449 más que en el procesador estándar. Esto incluye los 420 votadores necesarios.

La lógica de los votadores que se inserta a continuación de los registros tiene su incidencia pues ralentiza ligeramente el proceso. El procesador tolerante a fallos requiere un periodo mínimo de reloj de 6.819 ns alcanzando una frecuencia máxima de 146.649 MHz, lo que significa una reducción de la frecuencia máxima de funcionamiento en casi 10 MHz.

7.3. Simulaciones

A continuación se analizan las gráficas generadas por las simulaciones de los apéndices A y B.

7.3.1. Procesador estándar

En esta sección se analizan los resultados obtenidos de la ejecución sobre la implementación estándar del microprocesador.

Prueba de control

En la gráfica C.1 del apéndice C se observa paso a paso la ejecución del programa de control sobre el procesador estándar. A continuación se listan los eventos significativos por orden cronológico.

Durante la ejecución de la prueba de control, resumida en la tabla 7.1, se observa cómo se inicializan, en los ciclos 5 a 9, los valores de las variables de la operación (R1, R2), las variables del control del bucle (R4, R5) y la variable resultado (R3). A continuación se observa cómo se ejecuta el bucle acumulando las sumas en la variable resultado y el decremento en el indicador de iteración. El bucle se ejecuta 5 veces dejando un resultado final de 125 en el registro resultado. Para finalizar el resultado se almacena en memoria (M0) en el ciclo 75.

Ciclo	Señales	Evento
0	-	Inicio de la ejecución
5	R2	Se lee de memoria el valor 5 y se almacena en el registro R2.
6	R1	Se carga el valor 25 en el registro R1.
7	R3	Se inicia el registro R3 a 0 para acumular el resultado.
8, 9	R4, R5	Se inician las variables de control del bucle. R4 almacena el número de iteraciones.
18	R3	Se realiza la primera acumulación.
19	R4	Se decrementa el contador de iteraciones.
25	PC	Se realiza un salto al inicio del bucle.
...	R3, R4, PC	Se realizan las iteraciones del bucle, acumulando el resultado en R3 hasta que R4 valga 0.
65	PC	Se termina de ejecutar el bucle, se sigue avanzando por el código.
75	M0	Se almacena el resultado en la memoria de datos.

Tabla 7.1: Simulación de control sobre el procesador estándar.

Prueba con fallos

Al volver a ejecutar el mismo código de prueba, esta vez insertando fallos (Apéndice C, sección C.2), lo primero que se observa es que el resultado final de M0 no es el correcto, se ha almacenado el valor 75 en vez de el 125 correcto. A continuación analizamos por qué.

El primer fallo ocurre en el bus de memoria de una etapa interna (MEM-bus_reg), y se propaga hasta almacenarse en el registro R2, es cuando se actualiza este registro cuando vemos los efectos del fallo, en vez del 5 esperado se ha almacenado un 4. Esto reduce el número de iteraciones que se realizarán.

Se continúa la ejecución del programa de forma normal hasta que ocurre el segundo fallo, esta vez se activa la señal «MEMWrite» encargada de indicar al módulo de memoria que almacene los datos. Esto se refleja en la señal «M3» que cambia de valor a 1. Este fallo no altera el funcionamiento de este programa pero la escritura en memoria podría alterar datos de otros programas si los hubiera.

El tercer fallo ocurre en el operando B de la ALU en una comparación. En vez de comparar la variable con la constante 0, se ha comparado con la constante 1, finalizando el bucle de ejecución antes de tiempo.

Ciclo	Señales	Evento
0	-	Inicio de la ejecución.
5	R2	Se lee de memoria el valor 5 y se almacena en el registro R2. Debido al primer fallo se almacena el valor 4.
6	R1	Se carga el valor 25 en el registro R1.
7	R3	Se inicia el registro R3 a 0 para acumular el resultado.
8, 9	R4, R5	Se inician las variables de control del bucle. R4 almacena el número de iteraciones.
17	-	Ocurre el segundo fallo. Se almacena un dato en la dirección 3 de memoria.
18	R3	Se realiza la primera acumulación.
19	R4	Se decrementa el contador de iteraciones.
25	PC	Se realiza un salto al inicio del bucle.
...	R3, R4, PC	Se realizan las iteraciones del bucle, acumulando el resultado en R3 hasta que R4 valga 0.
38	-	Ocurre el tercer fallo. Se termina el bucle antes de tiempo.
42	PC	Termina de ejecutar el bucle a falta de una iteración y se sigue avanzando por el código.
53	M0	Se almacena el resultado incorrecto en la memoria de datos.

Tabla 7.2: Simulación con fallos sobre el procesador estándar.

El resultado final es un programa mal ejecutado, que ha visto reducido las iteraciones que debería haber realizado, almacenando en memoria un dato incorrecto y además ha modificado elementos en memoria que no deberían haberse visto afectados.

7.3.2. Procesador tolerante a fallos

En esta sección se analizan los resultados obtenidos de la ejecución de las mismas pruebas anteriores sobre la implemetnación tolerante a fallos del microprocesador.

Prueba de control

En la gráfica del apéndice C sección C.3 se observa la evolución del programa a lo largo de su ejecución. El proceso sigue los mismos pasos seguidos al ejecutar las pruebas en el procesador estándar obteniendo los mismos resultados (tabla 7.3. Por lo cual se comprueba que la funcionalidad e integridad

Ciclo	Señales	Evento
0	-	Inicio de la ejecución
5	R2	Se lee de memoria el valor 5 y se almacena en el registro R2.
6	R1	Se carga el valor 25 en el registro R1.
7	R3	Se inicia el registro R3 a 0 para acumular el resultado.
8, 9	R4, R5	Se inician las variables de control del bucle. R4 almacena el número de iteraciones.
18	R3	Se realiza la primera acumulación.
19	R4	Se decrementa el contador de iteraciones.
25	PC	Se realiza un salto al inicio del bucle.
...	R3, R4, PC	Se realizan las iteraciones del bucle, acumulando el resultado en R3 hasta que R4 valga 0.
65	PC	Se termina de ejecutar el bucle, se sigue avanzando por el código.
75	M0	Se almacena el resultado en la memoria de datos.

Tabla 7.3: Simulación de control sobre el procesador tolerante a fallos .

del procesador se conservan al introducir la tolerancia a fallos.

Prueba con fallos

Para finalizar las pruebas, se realiza la simulación con fallos sobre el procesador tolerante a fallos. Lo primero que observamos en la simulación (apéndice C sección C.4) es que esta vez sí se ha ejecutado el programa de forma correcta, el resultado almacenado en memoria es el correcto y no se han modificado otras secciones de la memoria. A continuación analizamos la gráfica de la simulación, resumida en la tabla 7.4, para ver las diferencias.

El primer fallo se produce en uno de los registros del bus de memoria. El fallo es enmascarado gracias a que los otros dos registros han mantenido el valor correcto y a que el votador ha realizado su tarea, permitiendo que sea el valor más repetido el que continúe hasta el siguiente componente. El registro R2 almacena el valor 5 correctamente.

En el ciclo 17 se produce el segundo fallo. Este provocaría que se modificara la memoria de datos, sin embargo el sistema de tolerancia realiza su tarea y evita que esto ocurra.

El tercer fallo ocurre en el operando B durante el ciclo 38 de la ALU en una comparación. Este fallo, al haberse tolerado el primero, no habría afectado al programa ya que el valor de R4 es 2 y se compara con el valor 1, y no afecta a la ejecución ya que se realiza una comparación de igualdad. En cualquier caso, el fallo se ha enmascarado debido a la lógica del votador

Ciclo	Señales	Evento
0	-	Inicio de la ejecución
5	R2	Se lee de memoria el valor 5 y se almacena en el registro R2. Se almacena correctamente a pesar del primer fallo.
6	R1	Se carga el valor 25 en el registro R1.
7	R3	Se inicia el registro R3 a 0 para acumular el resultado.
8, 9	R4, R5	Se inician las variables de control del bucle. R4 almacena el número de iteraciones.
17	-	Ocurre el segundo fallo. Se tolera y no se almacena ningún dato en memoria.
18	R3	Se realiza la primera acumulación.
19	R4	Se decrementa el contador de iteraciones.
25	PC	Se realiza un salto al inicio del bucle.
...	R3, R4, PC	Se realizan las iteraciones del bucle, acumulando el resultado en R3 hasta que R4 valga 0.
38	-	Ocurre el tercer fallo. Se enmascara el fallo y no afecta a la ejecución.
65	PC	Se termina de ejecutar el bucle, se sigue avanzando por el código.
75	M0	Se almacena el resultado en la memoria de datos.

Tabla 7.4: Simulación con fallos sobre el procesador tolerante a fallos.

que siempre está activo.

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.