Resumen

La incorporación de mecanismos de detección de errores es un elemento fundamental en el diseño de sistemas tolerantes a fallos en los que, en muchos casos, la detección de un error (ya sea transitorio o permanente) es el punto de partida que desencadena toda una serie de acciones o activación de elementos que persiguen alguno de estos objetivos: la continuación de las operaciones del sistema a pesar del error, la recuperación del mismo, la parada de sus operaciones llevando al sistema a un estado seguro, etc. Objetivos, en definitiva, que pretenden la mejora de las características de fiabilidad, seguridad y disponibilidad, entre otros, del sistema en cuestión.

Uno de estos elementos de detección de errores es un procesador de guardia; su trabajo consiste en monitorizar al procesador del sistema y comprobar que no se producen errores durante la ejecución del programa.

El principal inconveniente de las propuestas existentes a este respecto y que impiden una mayor difusión de su uso es la pérdida de prestaciones y el aumento de consumo de memoria que sufre el sistema monitorizado. El aumento en el consumo de memoria se debe a la adición al programa original de datos (denominados firmas) que contienen la información necesaria para permitir la detección de errores. Y la pérdida de prestaciones proviene del hecho de que, en general, se trata de que sea el propio procesador del sistema el que realice las operaciones necesarias (o, al menos, que recupere las firmas) para detectar posibles errores durante la ejecución.

En este trabajo se propone una nueva técnica de empotrado de firmas (técnica denominada ISIS — *Interleaved Signature Instruction Stream*) intercaladas dentro del espacio de la memoria del programa. Con esta técnica es un elemento separado del procesador del sistema (un procesador de guardia como tal) el que realiza las operaciones encaminadas a detectar los errores. A pesar de que las firmas se encuentran mezcladas con las instrucciones del programa que está ejecutando, y a diferencia de las propuestas previas, el procesador principal del sistema no se involucra ni en la recuperación de las firmas ni en las operaciones de cálculo correspondientes, lo que reduce la pérdida de prestaciones.

También se propone una novedosa técnica para que el procesador de guardia pueda verificar la integridad estructural del programa que monitoriza que viene a resolver en gran medida el problema de la verificación de que el

procesador del sistema ha realizado un salto válido a una nueva zona del programa cuando existen múltiples posibles destinos válidos de dicho salto. Este problema no tenía una solución adecuada hasta el momento, y aunque la propuesta que aquí se hace no consigue resolver todos los posibles escenarios de salto sí permite incorporar un buen números de ellos al conjunto de saltos verificables.

La propuesta teórica ISIS y sus mecanismos de detección de errores se complementan con la aportación de un sistema completo (procesador, procesador de guardia, memoria caché, etc.) basado en ISIS y que incorpora los mecanismos de detección que aquí se proponen. Se ha denominado HORUS a este sistema, y está desarrollado en lenguaje VHDL sintetizable, de manera que es posible no sólo simular el comportamiento del sistema ante la aparición de un fallo y analizar su evolución a partir de éste sino que también es posible programar un dispositivo lógico programable tipo FPGA para su inclusión en un sistema real.

Para programar el sistema HORUS se ha desarrollado en este trabajo una versión modificada del compilador `gcc` que incluye la generación de las firmas de referencia para el procesador de guardia como parte integral del proceso de creación del programa ejecutable (compilación, ensamblado y montaje) a partir de código fuente escrito en lenguaje C.

Finalmente, otro trabajo desarrollado en esta tesis es el desarrollo de FIASCO (*Fault Injection Aid Software COmponents*), un conjunto de scripts en lenguaje Tcl/Tk que permiten la inyección de un fallo durante la simulación de HORUS con el objetivo de estudiar su comportamiento y su capacidad para detectar los errores subsiguientes. Con FIASCO es posible lanzar cientos o miles de simulaciones en un entorno distribuido para reducir el tiempo necesario para obtener los datos de campañas de inyección a gran escala.

Los resultados demuestran que un sistema que utilice las técnicas que aquí se proponen es capaz de detectar errores durante la ejecución del programa con una mínima pérdida de prestaciones, y que la penalización en el consumo de memoria al usar un procesador de guardia es similar a de las propuestas previas.