NESSY: Una plataforma de inyección de errores para una FPGA Virtex-5

Felipe Serrano, Víctor Alaminos, Juan Antonio Clemente, Hortensia Mecha[1] y Shih-Fu Liu[2]

Resumen— **La computación reconfigurable es una tecnología prometedora capaz de proporcionar un compromiso interesante entre flexibilidad y prestaciones en el mismo dispositivo. Esta característica es especialmente interesante en campos tales como la aviación, las misiones espaciales, o las aplicaciones nucleares, con requerimientos en tiempo real. Sin embargo, en estos entornos el dispositivo que procesa la información normalmente está expuesto a altas dosis de radiación, ya sean provenientes directamente desde el espacio exterior, o de materiales radioactivos, la cual puede causar errores espontáneos en el funcionamiento del sistema. Esto hace imprescindible desarrollar técnicas que evalúan estos errores. En este artículo proponemos una plataforma de inyección de errores que evalúa el impacto de un bitflip espontáneo en la memoria de configuración de una FPGA. Hemos implementado nuestro sistema en una FPGA Xilinx^{TM}Virtex 5 y la hemos probado usando dos aplicaciones reales representativas. Nuestro sistema presenta varias ventajas respecto a otros sistemas existentes en la literatura. En primer lugar, el coste de implementación de nuestro sistema es considerablemente menor que otros sistemas. En segundo lugar, nuestra plataforma no es intrusiva. En tercer lugar, mejoramos el tiempo que se necesita para inyectar un sólo bitflip en al menos dos órdenes de magnitud. Finalmente, hasta donde tenemos conocimiento, esta es la primera plataforma de inyección de errores que ha sido implementada en una FPGA Virtex-5.**

Palabras clave— **Inyección de errores, Virtex-5, FPGA.**

I. Introducción

LOS dispositivos reconfigurables, y en especial, las FPGAs basadas en memoria SRAM (SRAM-FPGAs) han ganado popularidad para su uso en campos como la investigación nuclear, la aviación y las misiones espaciales, por mencionar algunos. Los principales motivos son su alta densidad de recursos configurables, así como la reconfiguración parcial.

Sin embargo, un problema frecuente en estos campos, especialmente en el sector espacio, es la alta dosis de radiación a la cual estos dispositivos están expuestos [15], [5]. Estas radiaciones pueden causar errores conocidos como Single Event Upsets (SEUs), los cuales pueden alterar el funcionamiento de los componentes electrónicos empotrados en los sistemas embarcados [5]. En el caso de las FPGAs, estos errores afectan a la memoria de configuración, lo cual causa un cambio en la funcionalidad del sistema. Por tanto, para garantizar el correcto funcionamiento de un sistema autónomo bajo tales condiciones, encontrar una manera de solucionar este problema es de vital importancia. Una técnica muy utilizada es la Redundancia Triple Modular (o TMR, por sus siglas en inglés *Triple Modular Redundancy* [14], [7]). Esta técnica consiste en replicar el circuito 3 veces y procesar las 3 salidas resultantes *votándolas* para generar una sola salida (el proceso de votación consiste en seleccionar la salida que más veces se repita). De este modo, si alguna de las instancias del sistema falla, las otras dos pueden corregir y enmascarar el error producido. Sin embargo, su problema es que consume muchos recursos hardware, ya que implica multiplicar el coste del hardware por 3, más el coste del circuito que implementa el votador.

Por este motivo, puede resultar interesante aplicar redundancia en sólo la parte más sensible del circuito. Sin embargo, no es trivial identificar qué zona(s) del sistema son más sensibles a errores. Es más, normalmente la única manera de obtener esta información es realizar un estudio intensivo de radiación en la memoria de configuración del dispositivo.

En este artículo proponemos una plataforma de inyección de errores, a la que hemos llamado NESSY, que es capaz de evaluar el impacto de un bitflip (un cambio espontáneo de 0 a 1 o de 1 a 0) en la memoria de configuración de un sistema reconfigurable. Para ello, esta herramienta realiza un chequeo exhaustivo del efecto de un bitflip en *todos y cada uno* de los bits de la parte de la memoria de configuración que implementa la región del circuito bajo prueba. Hemos implementado esta plataforma en una FPGA Xilinx^{TM}Virtex-5 usando las herramientas de desarrollo Embedded Development Kit (EDK) y Plan Ahead 12.1. Creemos que la plataforma propuesta es interesante porque:

- Tiene bajo coste en recursos, porque sólo se necesita una FPGA y un PC para simular la inyección de un SEU, a diferencia de otros sistemas propuestos en la literatura [2], [3], [4], [1].
- No es intrusivo; es decir, el usuario puede decidir en qué región de la FPGA colocar y rutar el circuito bajo prueba. Por tanto, esta herramienta sólo inyecta bitflips en esta región específica para evaluar el impacto de los bitflips sólo en el circuito que se va a testear. En consecuencia, los resultados experimentales serán realistas.
- Es mucho más eficiente que otros sistemas presentados en la literatura [10], [1]. De hecho, inyectar un bitflip con NESSY es al menos dos órdenes de magnitud más rápido que con cualquiera de estos otros sistemas.
- Hasta donde tenemos conocimiento, esta es la primera plataforma de inyección de errores que ha sido implementada en una Xilinx^{TM}Virtex-

[1]Dpto. de Arquitectura de Computadores y Automática, Univ. Complutense de Madrid, e-mail: horten@fis.ucm.es.
[2]Dpto. de Ingeniería Informática, Univ. Antonio de Nebrija, e-mail: sliu@nebrija.es.

5. En este caso, hemos implementado nuestro sistema en una placa de prototipado XUPV505-LX110T. Sin embargo, nuestro sistema es portable a cualquier dispositivo Virtex-5.

II. Trabajo Relacionado

Las FPGAs basadas en memoria SRAM son especialmente sensibles a errores temporales, especialmente aquellos producidos por el impacto de una partícula cósmica que afecte al estado de una celda de la memoria de configuración [5]. Durante los últimos años, se han desarrollado técnicas que permiten detectar y corregir estos errores [8]. La mayoría de estas técnicas están basadas en TMR [14], [7], con el consecuente incremento del área física. Para reducir este coste, algunos investigadores han presentado técnicas alternativas, como en [11], donde tras un profundo conocimiento del circuito, se replican sólo partes del mismo; o en [6] donde utilizan reconfiguración parcial y un procesador para determinar cuándo es necesario reconfigurar. Al mismo tiempo, las últimas generaciones de FPGAs han incorporado módulos empotrados que facilitan la detección y recuperación de SEUs en la memoria de configuración, como por ejemplo el puerto ICAP, y el módulo para la corrección de errores de la memoria de configuración (*Frame Error Correcting Code core*) disponible en las FPGAs de Xilinx^{TM}Virtex-4 [12] y Virtex-5 [13]. Como estos módulos son indispensables para las aplicaciones tolerantes a fallos, su propia protección también es un tema de investigación actual [9].

Por otra parte, para verificar el correcto comportamiento de un circuito bajo la presencia de un SEU, surge la necesidad de diseñar plataformas que permitan emular estos fallos. Son las llamadas plataformas de inyección de errores, que pueden ser de tres tipos: basadas en simulación, basadas en software y basadas en hardware [10], siendo estas últimas las que más ventajas proporcionan ya que funcionan a frecuencias más altas y permiten trabajar sobre la implementación final del circuito.

La mayor parte de las plataformas hardware de inyección de errores existentes se han desarrollado sobre las FPGAs Xilinx^{TM}Virtex-II. Una de las más importantes es FLIPPER [3], que utiliza dos FPGAs, una para controlar el experimento y la otra para implementar el circuito que se va a testear. La primera FPGA se encarga de inyectar bitflips en la memoria de la segunda FPGA mediante un sistema de registros muy complejo. Esta plataforma ha sido validada de forma experimental en [2], comparando las probabilidades de fallo obtenidas por la plataforma con las obtenidas inyectando partículas y produciendo SEUs con un acelerador de protones. Sin embargo, presenta algunos problemas: en primer lugar su alto coste, al necesitar 2 FPGAs y un diseño ad-hoc para las placas; en segundo lugar resulta muy intrusiva, puesto que evalúa el funcionamiento del sistema inyectando bitflips en todo el área de la FPGA, como si el diseño ocupase completamente la misma; y por último, debido al diseño ad-hoc para las placas no es fácilmente portable a otras tecnologías.

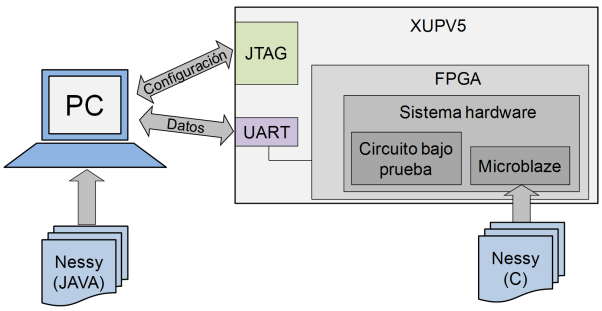

Fig. 1. Arquitectura de Nessy

Otra plataforma hardware de inyección de errores para Xilinx^{TM}Virtex-II es FTUNSHADES [4], que también utiliza 2 FPGAs y un PC. En esta plataforma es necesario implementar 2 copias del diseño que se va a testear en la FPGA, una de ellas es el circuito sin errores y la otra es el diseño donde se inyectan los errores. Esto da lugar a un alto intrusismo en el diseño original, y que el diseño que se testea pueda diferir enormemente del diseño original en cuanto a la colocación y rutado de sus módulos. Además es muy costosa en consumo de recursos.

Una plataforma de bajo coste no intrusiva para Xilinx^{TM}Virtex-II es [10]. Esta plataforma utiliza una lista de errores, un conjunto de estímulos e información detallada sobre la arquitectura "Boundary scan" para acceder al circuito a testear. Las comunicaciones se realizan a través del protocolo JTAG, por lo que son muy lentas, necesitando 0.62 segundos para inyectar un fallo.

En cuanto a tecnologías más modernas, recientemente se ha desarrollado una plataforma hardware para Xilinx^{TM}Virtex-4 [1], que también utiliza un PC y dos placas de desarrollo Xilinx^{TM}ML410, una para controlar el experimento y la otra como circuito a testear. El controlador inyecta los fallos utilizando reconfiguración parcial dinámica a través del puerto ICAP de la DUT. En algunos casos, la inyección produce un fallo en el propio interfaz ICAP del sistema, con lo cual es necesario reconfigurar toda la FPGA a través del interfaz externo de configuración. Estas características hacen que sea una plataforma cara (2 placas) y muy lenta, pues necesita 8 días para un test completo de inyecciones (13 millones de bitflips aproximadamente).

III. NESSY

Nuestra plataforma de inyección de errores, NESSY, consiste en un sistema software que se ejecuta en un PC y un sistema hardware basado en microprocesador implementado sobre la FPGA donde también se implementa el *circuito bajo test*. La figura 1 muestra la arquitectura de NESSY. El procesador del sistema hardware (implementado como un Microblaze) ejecuta otro programa que se sincroniza con el PC. Hemos escrito ambos programas usando JAVA y C, respectivamente.

El PC se conecta con la placa a través de dos canales de comunicación, etiquetados en la figura

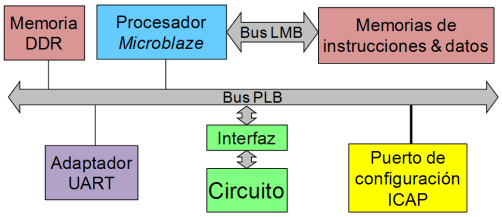

Fig. 2. Arquitectura del sistema hardware

como canales de *Configuración* y *Datos*, respectivamente. Este último conecta el PC y la UART de la placa XUPV5 con el fin de realizar las comunicaciones serie usando el protocolo RS232. Usando este canal de comunicación, los programas que se ejecutan en el PC y en el Microblaze se sincronizan para realizar la inyección de bitflips en el circuito bajo test. Hemos limitado la comunicación a través del puerto serie en la medida de lo posible, ya que el protocolo RS232 es muy lento. En consecuencia, este canal se usa sólo para enviar el bitstream del circuito bajo test del PC al sistema hardware (sólo una vez), y sincronizar los programas que se ejecutan en el PC y en el Microblaze durante la inyección de bitflips.

La plataforma sólo realiza la inyección de bitflips en el circuito bajo test. Para saber cuándo un bitflip ha provocado un error en el circuito, usamos un *testbench*. El usuario proporciona este testbench y consiste en un conjunto de valores representativos de las entradas del circuito bajo test.

Por cada inyección de un bitflip, comparamos los resultados obtenidos al ejecutar el testbench en el circuito modificado con los resultados correctos obtenidos cuando se ejecuta el mismo testbench en el circuito original sin errores. En adelante, nos referiremos a esos "resultados correctos" como *golden*. NESSY es capaz de generar un *golden* a partir de un circuito sin bitflips inyectados, o también puede directamente cargar uno de una ejecución previa del circuito. En las siguientes subsecciones se darán más detalles sobre los sistemas hardware y software de NESSY así como de la plataforma realiza la inyección de bitflips.

A. Sistema hardware

La figura 2 muestra la arquitectura del sistema hardware de NESSY, que consta de: un Microblaze, que ejecuta un programa situado en una memoria RAM interna de la FPGA (conectado al Microblaze usando un *Local Memory Bus* (LMB)); una memoria DDR2; dos adaptadores para la UART y el puerto de configuración ICAP; y el circuito bajo test, que está conectado al sistema a través de una interfaz de 32 bits de ancho (esto permite conectar circuitos con un máximo de 32 bits de entrada/salida). Todos estos elementos están conectados a través de un bus PLB.

El sistema hardware escucha al puerto serie esperando un comando enviado desde el PC. Cuando se lanza un comando, el Microblaze realiza la acción oportuna (éstas seran explicadas en detalle en la siguiente subsección). Hay dos razones fundamentales para usar el procesador Microblaze y para realizar

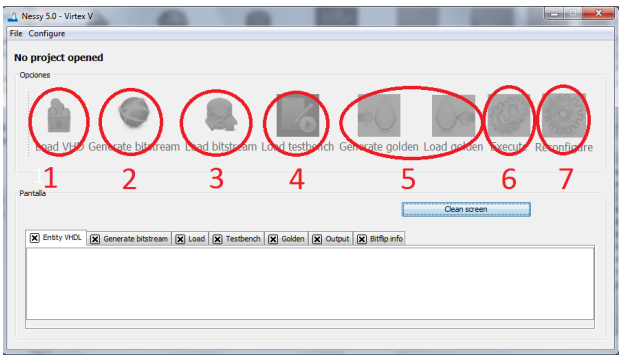

Fig. 3. GUI de NESSY

parte de la carga de trabajo en la FPGA. La primera es usar el puerto de configuración ICAP (en vez del puerto externo JTAG) para realizar las reconfiguraciones parciales del circuito bajo test. De hecho, según nuestras mediciones una reconfiguración parcial hecha a traveés del puerto ICAP es de 4 ordenes de magnitud más rápida que usando los otros dos puertos externos existentes. La segunda razón es evitar el puerto serie en la medida de lo posible. Esto es especialmente interesante para el proceso de inyección de bitflips, ya que al hacerlo directamente en el Microblaze, nos permite almacenar el bitstream parcial del circuito bajo test en la memoria DDR2. Esto reduce drásticamente la cantidad de comunicaciones que tenemos que realizar entre el PC y la placa XUPV5 al sólo necesitar enviar una única vez el bitstream desde el PC al sistema hardware por el puerto serie.

B. Sistema software

El sistema software de NESSY controla el flujo de ejecución en el proceso de inyección de bitflips y también se sincroniza con el programa corriendo en el Microblaze. Este flujo de proceso consiste en una serie de pasos consecutivos, los cuales son:

1. *Load HDL*
2. *Generate bitstream*
3. *Load bitstreams*
4. *Load testbench*
5. *Generate/load golden*
6. *Execute*
7. *Reconfigure*

El usuario controla la ejecución de cada uno de estos pasos a través de una interfaz gráfica (GUI), mostrada en la figura 3. La GUI se ejecuta en el PC y cuando es necesario, envía un comando a la FPGA para que el Microblaze realice parte de los cálculos. Los pasos 1-4 se realizan completamente en el PC, mientras que los pasos 5-7 se ejecutan parte en el PC y parte en el Microblaze.

1) Load HDL: El primer paso es seleccionar archivo *.vhd* top del circuito bajo test. A continuación, NESSY parsea este archivo para obtener el número de entradas, salidas y sus nombres. Todo esto se realiza a través de la GUI, por lo que trabajar con NESSY es muy fácil e intuitivo.

2) Generate Bitstreams: En este paso, el usuario indica la región de la FPGA donde se colocara el circuito bajo test. Esto permite a NESSY saber en qué región de la FPGA debe realizar la inyección de bitflips. Hecho esto, NESY genera dos bitstreams: uno conteniendo todo los elementos del sistema hardware excluyendo el circuito bajo test (en adelante, nos referiremos a este subsitema como el *sistema hardware estático*) y otro conteniendo sólo el circuito bajo test, que se cargará en la FPGA mediante reconfiguración parcial. NESSY crea esos archivos *.bit* usando las herramientas Xilinx^{TM}ISE 12.1, EDK 12.1 y PlanAhead 12.1.

3) Load Bitstreams: Este paso carga los bitstreams correspondientes al *sistema hardware estático* y al circuito bajo test en la memoria de configuración de la FPGA usando Xilinx ^{TM}iMPACT 12.1. A partir de ahora, en la FPGA, el Microblaze está esperando comandos enviados por el usuario.

4) Load Testbench: En este paso el usuario selecciona un testbench que se utilizará para verificar el correcto funcionamiento del circuito bajo test para cada inyección de un bitflip.

5) Generate/Load Golden: Este paso proporciona dos maneras de gestionar la generación/actualización del *golden*, etiquetado en la GUI como "Generate Golden" y "Load Golden" respectivamente. La primera opción genera el *golden* ejecutando el testbench seleccionado en el paso 4 en el circuito bajo test. Una vez que se genera el *golden*, el Microblaze lo almacena en la memoria DDR2. La segunda opción carga directamente un *golden* que ha sido previamente generado mediate simulación.

6) Execute: Este paso no es necesario para la inyección de bitflips. El único propósito de este botón es asegurarse de que las salidas del circuito que está actualmente cargado en la FPGA se corresponden a los valores del *golden*. Esto se consigue ejecutando el testbench (seleccionado en el paso 4) una vez. Esto es útil para verificar el correcto funcionamiento del circuito bajo test, e inicialmente fue incluido únicamente para depurar.

7) Reconfigure: Este paso realiza una evaluación exhaustiva del impacto de la inyección de un bitflip en el circuito bajo prueba. Para ello, se lanzan dos procesos: uno en el PC, y el otro en el Microblaze del *sistema hardware estático*. Ambos procesos se ejecutan en paralelo y se sincronizan haciendo uso del puerto serie y del protocolo de comunicación RS232.

Por un lado, el proceso que se ejecuta en el *sistema hardware estático* inyecta los bitflips en la memoria de configuración (donde se encuentran los bits de configuración que implementan el circuito bajo prueba), ejecuta el circuito usando el testbench, compara los resultados con los del *golden* y envía los resultados de las comparaciones al PC usando el puerto serie. Por otro lado, el proceso que se ejecuta en el PC recibe estos resultados y los imprime en la interfaz gráfica de NESSY, y en un fichero de log.

El proceso de inyección de bitflips consiste en 3 pasos, los cuales están representados en la figura 4.

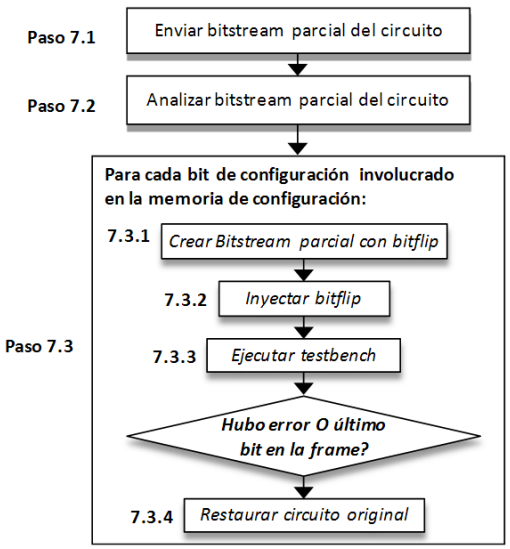

Fig. 4. Organigrama del proceso de inyección de bitflips

En primer lugar (paso 7.1), el PC envía al Microblaze el bitstream correspondiente al circuito a testear (sin la inserción de ningún bitflip), y el Microblaze lo almacena en la memoria DDR2 de la FPGA.

En el segundo paso (paso 7.2), se hace un análisis de este bitstream parcial, para así identificar sus *mapas de configuración*. Un mapa de configuración es un bloque de intrucciones del bitstream que configuran un conjunto de recursos lógicos *adyacentes* (CLBs, Block RAMs...) existentes en la FPGA. De este modo, un bitstream contiene tantos mapas de configuración como bloques de recursos lógicos no adyacentes utilice el circuito. Cada mapa de configuración se compone de una escritura en un registro de dirección existente en la circuitería de configuración de la Virtex-5, el *Frame Address Regiter* (FAR), seguido por una o más escrituras en un registro de datos, el *Frame Data Input* (FDRI). El FAR identifica qué *frame* será reconfigurada, mientras que el FDRI contiene los bits de configuración que serán cargados en ella (una *frame* es el segmento mínimo direccionable en la memoria de configuración de la Virtex-5). De esta manera, existen tantos mapas de configuración como escrituras en el registro FAR. Identificar los mapas de configuración es necesario en el paso siguiente para saber la frame específica que se debe modificar en la inyección de un único bitflip.

Finalmente, en el paso 7.3 se realiza la inyección de bitflips *para cada bit de configuración en la memoria de configuración que implementa el circuito bajo prueba*. Este proceso se compone de 4 subpasos (también identificados en la figura 4).

En el primero (7.3.1) se inyecta un bitflip en el bit correspondiente de la memoria de configuración. Para ello, NESSY *genera un nuevo bitstream*, el cual aparece en la figura 4 como "Bitstream parcial con bitflip". *Este bitstream sólo contiene una escritura en el registro FAR y una escritura en el registro FDRI*. Estos dos comandos escriben el nuevo valor de los bits de configuración (incluyendo el bitflip) para la frame que se corresponde al bit que ha sido modifi-

cado. La información obtenida durante el análisis del bitstream del circuito en el paso 7.2 se utiliza en este paso para identificar qué frame se debe modificar.

En el siguiente subpaso (7.3.2) se carga el *bitstream parcial con bitflip* en la FPGA usando el puerto de configuración ICAP, para así inyectar el bitflip generado en el subpaso previo. Este bitstream es muy pequeño en comparación con el original que implementa el circuito bajo prueba, debido a que sólo configura una frame.

Una vez que el bitflip ha sido inyectado, en el subpaso 7.3.3 se ejecuta el testbench en el circuito modificado, se comparan los resultados de la ejecución con los valores almacenados en el *golden* y se envía la comparación de resultados al PC usando el puerto serie. Si ambos resultados son diferentes, la inyección de bitflip ha generado un error.

Finalmente, se restaura el circuito original para eliminar el bitflip inyectado en el subpaso 7.3.2. Para ello, se carga otra vez el bitstream completo del circuito bajo test (subpaso 7.3.4). Sin embargo, si el bitflip no ha generado ningún error, se puede restaurar el circuito original al inyectar el siguiente bitflip, ya que el valor escrito en el FDRI para la inyección del bit n restaurará el valor del bit $n-1, n>1$. Esto es cierto *a menos que el bit que tiene que ser restaurado sea el último de una frame*. Por consiguiente, el circuito original completo se restaura sólo cuando el bitflip haya generado un error *o* cuando el bit modificado sea el último de la frame.

IV. Resultados Experimentales

Hemos evaluado la eficiencia de NESSY usando dos aplicaciones reales: un contador de 8 bits y un filtro FFE. Es importante destacar que en ambos experimentos ha sido necesario tomar la siguiente consideración adicional: *Xilinx^{TM}Planahead no puede asegurar que los cables que comunican el sistema hardware estático no crucen la región parcialmente reconfigurable*. En otras palabras, es posible que algunos bits en la memoria de configuración que configura la región reservada para el circuito bajo prueba configuren ciertos cables que comunican el *sistema hardware estático*. Si se inyecta un bitflip en dichos bits, el sistema podría dejar de funcionar. NESSY es capaz de manejar estas situaciones: si el tiempo transcurrido entre la inyección del bitflip y la recepción de los resultados del testbench excede un *timeout*, el sistema completo se reconfigura de nuevo y la inyección de bitflips se reanuda desde el bitflip siguiente al que causó la caída del sistema. Sin embargo, esta opción es muy costosa en tiempo. Por ello, para acelerar el proceso de inyección de bitflips y obtener así resultados realistas, en ambos experimentos hemos seleccionado manualmente una región libre de cables que comuniquen el *sistema hardware estático*, haciendo uso de la herramienta Xilinx^{TM}FPGA Editor.

En ambos experimentos, hemos decidido situar el circuito testeado en la esquina superior derecha de la FPGA, mientras que el *sistema hardware estático* se sitúa en la parte inferior del dispositivo.

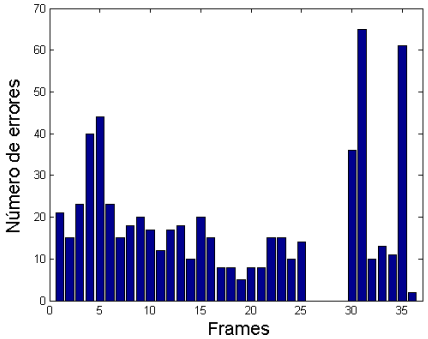

Fig. 5. Número de bitflips por frame que causaron errores en el contador de 8 bits

A. Contador de 8 bits

El contador de 8 bits es una aplicación pequeña que utilizamos en primer lugar para realizar test rápidos. Este circuito sólo necesita 2 slices (o 1 Configurable Logic Block (CLB), que contiene 2 slices). Sin embargo, como se ha comentado anterioremente, la unidad de configuración mínima en una Virtex-5 es una frame, la cual engloba la configuración de la 36ª parte de una columna entera de 20 CLBs. Por tanto, el bitstream correspondiente al circuito parcial realmente ocupa una columna completa de CLBs, que es la región parcialmente reconfigurable mas pequeña que PlanAhead puede generar.

En este experimento, NESSY realiza un test exhaustivo de los 46080 bits de configuración de la columna de CLBs del circuito en 3 minutos y 15 segundos. Cada test completo de bitflip tarda una media de 4.23 ms. Esto comprende generar e inyectar un único bitflip, ejecutar un testbench de 200 entradas en el circuito modificado y restaurar al circuito original si es necesario. Sin embargo, sólo la creación e inyección de un bitflip (subpasos 7.3.1 y 7.3.2 en la figura 4) es mucho más rápida: 0.63 ms. Esto es dos órdenes de magnitud más rápido que otras soluciones aportadas en la literatura [10], [1], que consiguieron 620 y 53 ms *por inyección de un único bitflip*, respectivamente. De los 46080 bitflips inyectados, sólo 617 causan una ejecución erronea del circuito testeado, lo cual es un 1.3% del total.

La figura 5 muestra el número de bitflips que produjeron error en el circuito, por cada frame existente en la columna de CLBs usada (recordemos que una columna de 20 CLBs contiene 36 frames). Las frames 4-5, 30-31 y 35 son más propensas a errores que las demás. Sin embargo, los bitflips no afectan cuando caen en las frames 26-29. Este es un resultado interesante, ya que indica que esta parte del circuito no tiene por qué ser protegida contra radiaciones.

B. Filtro FFE

Un Filtro FFE es un tipo específico de filtro de respuesta finita de impulsos. Cuando se sintetiza e implementa este circuito en la FPGA, EDK infiere alrededor de 200 slices y 32 DSPs empotrados exis-

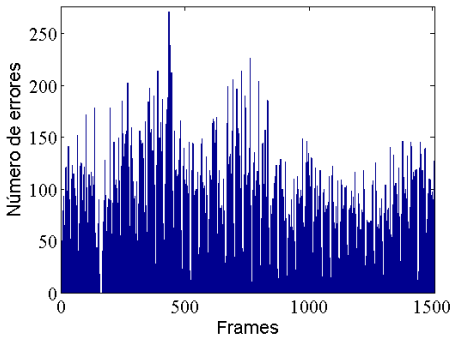

Fig. 6. Número de bitflips por frame que causaron error en el filtro FFE

tentes en la Virtex-5. Sin embargo, esta herramienta permite sintetizar este circuito *sin* usar DSPs. Seleccionando esta opción, los recursos necesarios para su implementación son 1680 slices, lo cual es un 9.72% de los slices existentes en la FPGA. Hemos seleccionado esta opción para evaluar el impacto de la inyección de bitflps en un circuito de tamaño mayor.

En este caso, Nessy realiza una inyección exhaustiva de bitflips en 5 horas, 21 minutos y 9 segundos usando un testbench con 200 entradas diferentes. Esto da como resultado un testeo de bitflip de 9.76 ms de media. El motivo del incremento de este tiempo con respecto al contador de 8 bits (4.23 ms) es debido al retardo adicional introducido al restaurar el bitstream parcial de la FFE, el cual es considerablemente mayor (1680 slices) que el del contador de 8 bits (sólo 2 slices). Sin embargo, el tiempo de inyección de bitflips es exactamente el mismo que en el anterior experimento (0.63 ms). Esto se debe a que este tiempo no depende del tamaño del circuito testeado, ya que la inyección de un bitflip involucra *sólamente* la configuración de una frame de la FPGA.

En este experimento se realizaron un total de 1971200 inyecciones, de las cuales 113771 produjeron error (5.8% del total). La figura 6 muestra cómo se distribuyen los errores entre las 1512 frames usadas para la implementación de este circuito. Tal y como muestra la figura, las frames 157-167 no se vieron afectadas por los bitflips.

V. Conclusiones y trabajo futuro

En este trabajo hemos presentado una nueva plataforma de inyección de errores, NESSY, sobre una FPGA Virtex-5. Esta plataforma presenta numerosas ventajas con respecto a otras que se pueden encontrar en la literatura. En primer lugar, no es intrusiva, ya que el usuario puede decidir exactamente dónde se coloca el circuito. En segundo lugar, resulta menos costosa que otras plataformas, al necesitar un PC y sólo una FPGA. Por último, permite acelerar la inyección de errores, ya que la emulación de un SEU se ha reducido en al menos 2 órdenes de magnitud respecto a otras plataformas existentes.

En el futuro queremos incluir en esta plataforma varias herramientas automáticas que permitan mejorar la fiabilidad de un determinado diseño, teniendo en cuenta los resultados obtenidos por NESSY sobre la vulnerabilidad de las distintas partes del mismo durante la inyección de errores. Además, queremos incluir la posibilidad de emular fallos en los que el impacto de una partícula cósmica afecte a más de una celda de memoria, es decir, multi-bitflips, efecto cada vez más cercano con las nuevas tecnologías.

Agradecimientos

Este trabajo ha sido financiado por los proyectos de investigación TIN2009-09806 y AYA2009-13300-C03-02.

Referencias

[1] Monson, J. S., Wirthlin, M. and Hutchings, B., *Fault Injection Results of Linux Operating on an FPGA Embedded Platform*, Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs (ReConFig), pp. 37-42, 2010.

[2] Alderighi, M. et al., *Experimental Validation of Fault Injection Analyses by the FLIPPER Tool*, IEEE Transactions on Nuclear Science, Vol. 57, No. 4, pp. 2129-2134, 2010.

[3] Alderighi, M. et al., *Evaluation of Single Event Upset Mitigation Schemes for SRAM based FPGAs using the FLIPPER Fault Injection Platform*, IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT), pp. 105-113, 2007.

[4] Sterpone, L., Aguirre, M., Tombs, J. and Guzman-Miranda, H., *On the design of tunable fault tolerant circuits on SRAM-based FPGAs for safety critical applications*, Design, Automation and Test in Europe (DATE), pp. 336-341, 2008.

[5] Anderson, K., *Low-Cost, Radiation-Tolerant, On-Board Processing Solution*, IEEE Aerospace Conference, pp. 1-8, 2005.

[6] Ilias, A., Papadimitriou, K. and Dollas, A., *Combining Duplication, Partial Reconfiguration and Software for On-line Error Diagnosis and Recovery in SRAM-Based FPGAs*, IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 73-76, 2010.

[7] Ichinomiya, Y., Tanoue, S., Amagasaki, M., Iida, M., Kuga, M. and Sueyoshi, T., *Improving the Robustness of a Softcore Processor against SEUs by Using TMR and Partial Reconfiguration*, IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 47-54, 2010.

[8] Parris, Matthew G., Sharma, Carthik A. and Demara, Ronald F., *Progress in autonomous fault recovery of field programmable gate arrays*, ACM Computing Surveys, Vol. 45, No. 4, pp. 31:1-31:30, 2011.

[9] Dutton, B. F. and Stroud, C. E., *Built-In Self-Test of Embedded SEU Detection Cores in Virtex-4 and Virtex-5 FPGAs*, International Conference on Embedded Systems & Applications (ESA), pp. 149-155, 2009.

[10] Battezzati, N., Sterpone, L. and Violante, M., *A new low-cost non intrusive platform for injecting soft errors in SRAM-based FPGAs*, Industrial Electronics, IEEE International Symposium on Industrial Electronics (ISIE), pp. 2282-2287, 2008.

[11] Shih-Fu Liu et al., *Increasing Reliability of FPGA-Based Adaptive Equalizers in the Presence of Single Event Upsets*, IEEE Transactions on Nuclear Science, Vol. 58, No. 3, pp. 1072-1077, 2011.

[12] Jones, L., *Single Event Upset (SEU) Detection and Correction Using Virtex-4 Devices*, Xilinx Application Note, XAPP714 (v 1.5), Xilinx Inc., 2007.

[13] Chapman, K. and Jones, L., *SEU Strategies for Virtex-5 Devices*, Xilinx Application Note, XAPP864 (v1.0.1), Xilinx Inc., 2010.

[14] Stott, E. A., Sedcole, N. P. and Cheung, P. Y. K. *Fault tolerance and reliability in field-programmable gate arrays*, IET Computers & Digital Techniques, Vol. 4, No. 3, pp. 196-210, 2010.

[15] Ziegler, J. F. and Lanford, W. A. *The Effect of Sea-Level Cosmic Rays on Electronic Devices*, Journal of Applied Physics, Vol. 52, No. 6, pp. 4305-4312, 1981.