УДК 621.3.049.77

СПОСОБЫ ИНЖЕКЦИИ СБОЕВ В ПЛИС В РАМКАХ РАЗРАБОТКИ МЕТОДОВ ПОВЫШЕНИЯ СБОЕУСТОЙЧИВОСТИ

Д.Е. Матафонов
ФГУП МОКБ «Марс»
Россия, 127473, Москва, 1-й Щемиловский пер., 16
E-mail: matafonovde@gmail.com

Ключевые слова: ПЛИС, инжекция, сбой, сбоеусотйчивость.

Аннотация: Из-за набирающих популярность производительных малых КА актуальность приобретают вопросы использования коммерческих ПЛИС в подобных миссиях. Главным препятствием является отсутствие радиационной стойкости в подобных решениях, т.о. необходимостью становится разработка методик парирования сбоев. Основным инструментом является имитация сбоев программно-аппаратным способом. В статье рассматриваются существующие способы инжекции сбоев в ПЛИС, а также предлагается новый вариант, обеспечивающий независимость от программной поддержки имитации сбоев производителем ПЛИС.

1. Введение

В связи с увеличением в последние годы количества запусков малых спутников (SmallSats) особую актуальность приобретает вопрос повышения производительности бортовых вычислителей с уменьшением занимаемого объема в космическом аппарате (КА) [1]. Одним из способов достижения требуемых параметров является использование производительных емких коммерческих программируемых логических интегральных схемах (ПЛИС) (Industrial исполнение), которые не являются радиационностойкими. Учитывая меньший по сравнению с большими КА срок службы, лимитированный бюджет и жесткие ограничения в габаритах, такое решение может оказаться приемлемым при обеспечении достаточной сбоеустойчивости.

Для имплементации методов парирования сбоев необходимо на время разработки иметь аппарат инжекции ошибок, с точностью имитирующих эффекты, возникающие при воздействии ионизирующего излучения космоса. Существует два метода имитации сбоев: физическое воздействие на микросхему заряженными частицами в лабораторных условиях и программно-аппаратная имитация [2]. Первый вариант дорог, трудоемок и малодоступен, что делает его подходящим только для финальных исследований уже созданной сбоеустойчивой системы, т.к. наиболее точно имитирует радиационные эффекты в космосе. Второй вариант приемлем и удобен для обычных этапов разработки, т.к. позволяет без дополнительного внешнего оборудования имитировать сбои при разработке методик их парирования.

2. Способы инжекции сбоев

На сегодняшний день существует достаточное количество реализаций способов инжекции ошибок как от компаний-производителей ПЛИС, так и от разработчиков вычислителей.

Фирма Altera поставляет в составе IDE Quartus утилиту In-System Memory Content Editor, которая предоставляет пользователю доступ к внутренней памяти и регистрам ПЛИС, не прерывая ее функционирование [3]. Данный метод прост и удобен, т.к. хорошо описан в прилагающейся документации, а его реализация требует лишь установки на нужных регистрах и памяти флагов «Enable_Runtime_Mod». Для фиксации ошибок используется In-System Sources and Probes Editor (SAP). Пример использования данных утилит приведен в [4], где для автоматизации процесса тестирования на однократные сбои задача была сведена к использованию TCL-скрипта и оценено влияние на программное ядро процессора Leon2. Минусом данного подхода является ограниченное количество семейств ПЛИС, которые поддерживают данный функционал, а также необходимость заранее указывать то, какие элементы должны быть модифицируемыми.

Компания Xilinx предоставляет наиболее широкие возможности для инжекции, детектирования и исправления сбоев памяти своих микросхем. Для обеспечения и тестирования достаточной сбоеустойчивости в составе пакетов разработки предоставляются такие возможности как частичная реконфигурация без выхода из рабочего режима; в семействах UltraScale присутствует возможность использования кодов коррекции конфигурационной памяти, а также проверки CRC; возможность инжекции сбоев как в пользовательскую память, так и в конфигурационную, возможность инжекции ошибки в несколько бит сразу [5]. Несомненным плюсом данного функционала является проработанность, наличие документации, широкие возможности применения. Минусом данного варианта является достаточно высокая стоимость современных ПЛИС, предоставляющих этот функционал, что зачастую может идти вразрез с целью минимальной стоимости системы.

На сегодняшний день существуют также и отдельные решения, созданные разработчиками систем для собственного применения. Например, в статье [6] описывается два варианта инжекции ошибок – для Hard Core и Soft Core процессоров, а также работа системы самоконтроля. В случае с Hard Core процессором (AVR микроконтроллер в составе SoC AT94K) у пользователя имеется возможность частично или полностью побайтно изменять конфигурационную память встроенной ПЛИС, но отсутствует возможность читать ее. Soft Core процессор был реализован на Xilinx Virtex-4 и Virtex-5. Функционал этих ПЛИС предоставляет read/write доступ к конфигурационной памяти и регистрам как через JTAG, так и через ICAP (внутренний порт доступа к конфигурационной памяти), что означает, что пользовательская логика может отслеживать изменения конфигурации. Это, в свою очередь, в комбинации с возможностью частичной реконфигурации предоставляет пользователю возможность реализовать механизм автоматической коррекции ошибок конфигурационной памяти через механизм чтение-модификация-запись. Несомненным плюсом предложенных методик является возможность имитации практических всех типов исправимых сбоев, возможность автоматизации процесса инжекции ошибок и оценки результатов самоконтроля. Минусом является узкая направленность методов под конкретные реализации.

Анализируя вышеприведенные примеры можно выделить следующую общую черту – большинство методик подстраивается под конкретную реализацию, отсутствует возможность создать общую методику отработки инжекции сбоев вне зависимости от базиса.

Для создания более универсального решения, которое бы не зависело от конкретной реализации и поддержки производителем, учитывая необходимость имитации в нормальных условиях, предлагается создать библиотеку, с помощью которой возможно было бы внедрять ошибки в логику проекта ПЛИС. Для этого в проекте в ключевых точках можно создать набор моделей за счет небольшого увеличения занимаемой логической емкости, которые будут вносить требуемые изменения без необходимости редактирования конфигурационной памяти. Использование такого метода отвязывает пользователя от конкретных утилит, позволяющих реализовывать инжекцию сбоев, что открывает возможность реализации одного проекта в разных базисах без необходимости изменения логики или программного обеспечения тестирования.

Для реализации «залипания» можно реализовать логику, представленную на рис. 1.

а) б)

Рис. 1. Модели, имитирующие сбои «залипания». а) – Модель «залипания» в лог. «0», б) – модель «залипания» в лог. «1». Сигнал генерации ошибки fault_in подается лог. «1».

Для реализации сбоя переключения предлагается использовать модель, представленную на рис. 2.

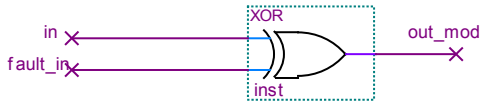

Рис. 2. Модель сбоя переключения. Длительность сбоя переключения определяется пользователем путем удержания Fault_in в лог. «1».

На рис. 3 представлен способ включения возможности инверсии вывода триггера.

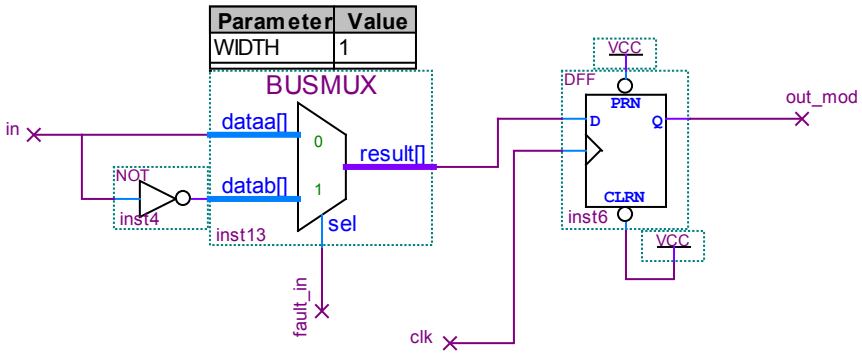

Рис. 3. Модель сбоя инверсии выхода триггера. Длительность сбоя определяется пользователем путем удержания fault_in в лог. «1».

Входы fault_in объединяются в один или несколько адресуемых регистров, управление которыми обеспечивает имитацию однократных сбоев. Пользователь может использовать микроконтроллер, процессор либо дополнительную логику в ПЛИС, чтобы задавать программу инжекции ошибок, которая в нужном порядке будет имитировать сбои в критических точках.

Для облегчения внедрения данных конструкций в существующий проект предлагается использовать package-библиотеку, в которой эти модели используются как функции. Содержимое package-библиотеки представлено ниже:

```
library ieee;
use IEEE.STD_LOGIC_1164.all;

package fault_injection is

function stuck_at_0 (data_for_0: std_logic; fault_in_for_0: std_logic)
return std_logic;

function stuck_at_1 (data_for_1: std_logic; fault_in_for_1: std_logic)
return std_logic;

function          transient_fault          (data_for_tr:          std_logic;
fault_in_for_tr:std_logic)
return std_logic;

function     bit_inv     (data_for_inv:     std_logic;     fault_in_for_inv:
std_logic:='0')
return std_logic;

end package;

package body fault_injection is

function stuck_at_0 (data_for_0: std_logic; fault_in_for_0: std_logic)
return std_logic is
      begin
      return (data_for_0 and (not fault_in_for_0));
      end;

function stuck_at_1 (data_for_1: std_logic; fault_in_for_1: std_logic)
return std_logic is
      begin
      return (data_for_1 or fault_in_for_1);
      end;

function          transient_fault          (data_for_tr:          std_logic;
fault_in_for_tr:std_logic)
return std_logic is
      begin
      return (data_for_tr xor fault_in_for_tr);
      end;

function bit_inv (data_for_inv: std_logic; fault_in_for_inv: std_logic)
return std_logic is
      variable result: std_logic;
      begin
            if fault_in_for_inv = '1' then result := not data_for_inv;
            else result := data_for_inv; end if;
      return result;
      end;

end package body;
```

В файле проекта для возможности инжекции сбоя защелкивания в лог. «1», например, в бит 3 счетчика достаточно вставить строку «counter(3) <= stuck_at_1(counter(3), fault_in(0));», где вызываемые аргументы – данные для модификации и сигнал управления инжекцией.

Один из самых распространенных случаев использования ПЛИС в системах – связующая логика между процессором и периферией, включая различные нестандартные интерфейсы. В этом случае легко добиться автоматизации тестов на одиночные сбои:

достаточно сделать регистр fault_in адресуемым и доступным центральному процессору, в этом случае прикладная программа сможет с нужной частотой и длительностью имитировать сбои. Размерность регистра определяется количеством точек, куда требуется вносить изменения.

Используя предложенные конструкции, синтезируемые в логике проекта, можно обеспечить независимость от конкретной реализации. Ярким примером, где данная техника может быть удобна, являются широко распространенные отечественные ПЛИС серии 5576 (например, 5576XC6T), которые основаны на серии Altera FLEX10KE, которые не поддерживают внутрисхемное редактирование конфигурационной и пользовательской памяти. Включение такой дополнительной логики в проект в ключевых узлах поможет отработать реакцию системы на возможные сбои под воздействием радиации.

Для реализации возможности инжекции сбоев потребуется небольшое количество дополнительной логической емкости ПЛИС – порядка 10%, что позволяет сохранять временные параметры в большинстве проектов. Использование этой методики на реальном проекте ПЛИС в составе вычислителя (25 точек внедрения, 4 типа сбоев) показало снижение рабочей частоты проекта с 39,4 МГц до 37,8 МГц, увеличение занимаемой логической емкости – 8%.

Достоинством данной реализации можно считать универсальность, простоту и возможность легкого масштабирования на требуемое количество тестируемых узлов, возможность использования в ПЛИС, не предоставляющих write-доступа к конфигурационной памяти, простая автоматизация. Недостатками метода являются недостаточная гибкость, которая выражается в необходимости изменения проекта ПЛИС для добавления тестируемых точек; покрытие тестами только пользовательской логики, невозможность тестов сбоев конфигурации при отсутствии внутреннего порта доступа к конфигурационной памяти; отсутствие готовых утилит.

3.Заключение

Обозревая существующие технологии в области инжекции сбоев, можно отметить широкое распространение фирменных методов и программных решений. Наибольшее количество реализованных проектов относится ко времени, когда производители не включали все возможности отработки однократных сбоев в свои среды разработки. Подавляющее большинство современных ПЛИС имеют возможность модификации конфигурационной памяти как с помощью отладочных средств, так и с помощью пользовательской логики. Однако реализация в каждом случае зависит от конкретной фирмы и\или микросхемы, а на сегодняшний день по-прежнему встречаются случаи, когда инжекция ошибок стандартными средствами вообще недоступна. Предложенный минимальный набор моделей однократных сбоев может быть реализован в любой ПЛИС, прост в реализации, легок в автоматизации. Данный метод применим при необходимости реализации проекта в разных базисах, либо при отсутствии возможности использовать стандартные методы инжекции сбоев.

Список литературы

1. George A.D., Wilson C.M. Onboard Processing With Hybrid and Reconfigurable Computing on Small Satellites // Proceedings of the IEEE. 2018. Vol. 106, No. 3. 2018. P. 458-459.

2. Velazco R., Foucard G., Peronnard P. Combining Results of Accelerated Radiation Tests and Fault Injections to Predict the Error Rate of an Application Implemented in SRAMbased FPGAs // IEEE Transaction on Nuclear Science. 2010. Vol. 57, No. 6. P. 3500-3505.
3. In-System Modification of Memory and Constants // URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/qts/qts_qii53012.pdf (Дата обращения: 11.12.18)
4. Mohammadi A., Ebrahimi M., Ejlali A., Miremadi S.G.. SCFIT: A FPGA-based fault injection technique for SEU fault model // Design, Automation & Test in Europe Conference & Exhibition, Germany, 2012. DOI: 10.1109/DATE.2012.6176538.
5. UltraScale Architecture Soft Error Mitigation Controller v2.0. // URL: https://www.xilinx.com/support/documentation/ip_documentation/sem_ultra/v2_0/pg187-ultrascale-sem.pdf (Дата обращения 11.12.18)
6. Dutton B., Ali M., Stroud C. Embedded Processor Based Fault Injection and SEU Emulation for FPGAs // Proceedings of the 2009 International Conference on Embedded Systems & Applications. Nevada, USA, 2009.