TÉCNICA DE DEPURACIÓN PARA EL CONTROL DE PROCESADORES RISC-V DE 32 BITS

MARCO EMILIO SARMIENTO BALLESTEROS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICOMECÁNICAS
ESCUELA DE INGENIERÍAS
ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2020

TÉCNICA DE DEPURACIÓN PARA EL CONTROL DE PROCESADORES RISC-V DE 32 BITS

MARCO EMILIO SARMIENTO BALLESTEROS

Trabajo de Grado para optar al título de
Ingeniero Electrónico

Director
Wilmer Daniel Ramirez Vera,
MEng. Electrónica.

Codirector
Elkim Felipe Roa Fuentes,
Philosophy Doctor.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICOMECÁNICAS
ESCUELA DE INGENIERÍAS
ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2020

CONTENIDO

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

RESUMEN

TÍTULO: TÉCNICA DE DEPURACIÓN PARA EL CONTROL DE PROCESADORES RISC-V DE 32 BITS [*]

AUTOR: MARCO EMILIO SARMIENTO BALLESTEROS [**]

PALABRAS CLAVE: System-on-chip, RISC-V, Debug, Testing.

DESCRIPCIÓN:

En este documento se presenta una descripción cualitativa de una técnica de depuración y control en un sistema integrado en chip (SoC) basado en un procesador RISC-V de 32 bits, indicando las ventajas, desventajas y posibles mejoras de las estrategias utilizadas. De igual manera, se presentan funcionalidades adicionales añadidas a la técnica seleccionada que permitan el acceso y la configuración de los registros de control y estado, así como el control de los puntos de quiebre del procesador, con el fin de facilitar el acceso al estado de operación del procesador y tener mayor control de este. Para el testeo de las funcionalidades adicionales, se realizaron simulaciones del chip completo en donde se encuentra integrada la plataforma de depuración, el procesador, buses del sistema y varios periféricos digitales. Además, se presentan resultados de la implementación en FPGA, así como datos obtenidos en su ejecución con el fin de comprobar el acceso a los registros de control y estado, así como el control sobre los puntos de quiebre. Adicionalmente, se presenta un modelo de simulación que permite realizar pruebas sobre el SoC como si se tratase de un sistema real, comunicando una descripción de hardware (en HDL) con una interfaz de software.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: Wilmer Daniel Ramirez Vera, MEng. Electrónica. Codirector: Elkim Felipe Roa Fuentes, Philosophy Doctor.

ABSTRACT

TITLE: DEBUGGING TECHNIQUE TO CONTROL 32-BIT RISC-V PROCESSORS [*]

AUTHOR: MARCO EMILIO SARMIENTO BALLESTEROS [**]

KEYWORDS: System-on-chip, RISC-V, Debug, Testing.

DESCRIPTION:

This document presents a qualitative description of a debugging and control technique in an integrated chip system (SoC) based on a 32-bit RISC-V processor, indicating the advantages, disadvantages and possible improvements of the employed strategies. Likewise, additional functionalities added to the selected technique are presented such as access and configuration of the control and status registers (CSR), besides the control of the processor breakpoints, to allow access to the processor operating state. and have better control of this. In order to test the additional functionalities, simulations of the full SoC are performed. These simulations involve a joint operation of different SoC subsystems such as the debugging platform, the processor, the system bus and the peripheral bus, and several digital peripherals. Moreover, this document presents a new simulation model that integrates hardware and software. It allows testing the debugging platform into the system on chip described using HDL, with programs described un C language emulating a real system operation. Finally, an FPGA implementation is performed and the obtained results are presented. The added functionalities such as control and status registers and the breakpoints control are measured using an integrated logic analyzer of the FPGA. These measurements are reported in this document.

---

[*] Bachelor Thesis

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: Wilmer Daniel Ramirez Vera, MEng. Electrónica. Codirector: Elkim Felipe Roa Fuentes, Philosophy Doctor.

INTRODUCCIÓN

El aumento en la complejidad de los sistemas integrados en chip (SoCs), presenta retos para los diseñadores a la hora de realizar pruebas, provocando un aumento de los tiempos de diseño y desarrollo de chips. Estas pruebas son críticas una vez el dispositivo se encuentra en post-silicio, pues permiten encontrar diferencias entre el comportamiento de los módulos implementados y los resultados obtenidos en simulación, dando información para posibles mejoras. Con el fin de facilitar estas tareas, los SoCs integran plataformas de depuración, que permiten de manera externa controlar y conocer el estado del procesador y del sistema en general, facilitando la búsqueda de errores y posibles mejoras en los diseños.

Las metodologías de depuración post-silicio se pueden dividir en 2 grandes grupos: *run-control* y *trace debugging*. En la primera, la plataforma de depuración tiene control sobre el SoC, permitiendo pausar su funcionamiento, para posteriormente observar el estado interno para encontrar errores y diferencias de diseño. Por otra parte, *trace debugging* está basada en la monitorización de eventos dentro del sistema, sin afectar de manera directa su comportamiento

En la tesis de maestría *"An Embedded Test Platform for System-On-A-Chip Interface"*[1] se presenta una plataforma de depuración post-silicio con comunicación JTAG, como parte de un SoC basado en un procesador RISC-V de 32 bits, fabricado en silicio con tecnología de 180 *nm*. Esta plataforma presenta un enfoque en ambas metodologías de depuración: a través de un monitor de bus permite monitorear las transacciones en el bus del sistema, y por otro lado, a través de un modulo de de-

[1] RAMIREZ, Wilmer. "An Embedded Test Platform for System-On-A-Chip Interface ". Tesis de Maestría (Maestría en Ingeniería, Área Ingeniería Electrónica). Colombia: Universidad Industrial de Santander, jul. de 2019.

puración: controlar y agregar instrucciones en el procesador, además de conocer y modificar el estado de los diferentes periféricos a través del acceso al bus del sistema.

Esta plataforma permite conocer el estado de los periféricos, así como del bus del sistema, sin embargo, no ofrece facilidad para conocer el estado del funcionamiento y operación del procesador del sistema. Si se quisiera conocer el estado del procesador desde el módulo de depuración, se debería agregar instrucciones al procesador que permitan leer los registros de estado. De ser el caso, en escenarios donde el procesador no funcione de manera adecuada y no sea posible cargar instrucciones, no se puede obtener información de su comportamiento. Esta facilidad se logra con el acceso a los registros de control y estado (CSR), los cuales almacenan información de identificación del dispositivo, información de rendimiento, además de permitir el manejo de interrupciones y excepciones y el control de puntos de quiebre.

Con este trabajo se busca revisar la plataforma presentada en esta tesis de maestría[1], desarrollada por el grupo de investigación OnChip, con el fin de conocer su funcionamiento y encontrar posibles mejoras. Basado en esta revisión, se desea añadir a la plataforma: nuevas funcionalidades que faciliten el control y la observabilidad del procesador con el acceso desde el módulo de depuración a los CSR y la capacidad de configurar puntos de quiebre (breakpoints) en la operación del procesador. Para el desarrollo de estas funcionalidades, fue necesario realizar un estudio del funcionamiento tanto del sistema de depuración, como del procesador implementado en el SoC, y el set de instrucciones en el cual está basado. Esto permitió representar interfaces de comunicación y de control del depurador hacia el procesador usando lenguajes de descripción de hardware. Con estas adiciones, se busca potenciar el sistema de depuración para ser implementado en futuros proyectos del grupo OnChip.

Este reporte presenta en el capítulo 2 una descripción de las características de la

plataforma seleccionada, junto con sus ventajas y posible mejoras. En el capítulo 3, se presentan y describen las características adicionales que se diseñaron sobre la plataforma presentado, como son el acceso a los registros de control y estado del procesador, y el control sobre los puntos de quiebre. En el capítulo 4 se presentan los resultados de simulación del SoC completo probando las nuevas características. Además se presenta un modelo de simulación que permite realizar pruebas sobre el SoC como si se tratase de un dispositivo real. El capítulo 5, presenta los resultados implementados en FPGA, en donde se realiza una implementación del chip completo integrando la plataforma de depuración. Por último, en los capítulo 7 y 6 se presentan las conclusiones del trabajo realizado así como posible trabajo futuro que abre este proyecto.

1. OBJETIVOS

Objetivo general

- Representar mediante un lenguaje de descripción de hardware, una técnica de depuración para el control de un procesador RISC-V de 32 bits. La descripción se probará usando un lenguaje de descripción de hardware.

Objetivos específicos

- Examinar una técnica de depuración y control de procesadores en sistemas integrados en chip.
- Representar mediante un lenguaje de descripción de hardware, una técnica que permita controlar los puntos de ruptura en la operación de un procesador RISC-V de 32 bits.
- Representar mediante un lenguaje de descripción de hardware, una técnica para acceder y configurar los registros del control y estado de un procesador RISC-V de 32 bits.

2. ESTUDIO DEL SISTEMA DE DEPURACIÓN

La plataforma de depuración fue desarrollada como parte de un trabajo de maestría[1] y fue implementada como parte de un SoC desarrollado en el grupo de investigación OnChip. El SoC fue implementado con un procesador RISC-V de 32 bits y con bus de sistema AHB-lite. La plataforma de depuración permite controlar el procesador, ya que cuenta con la capacidad de detener, continuar y reiniciar su funcionamiento. Adicional a ello, la plataforma cuenta también con la funcionalidad de ser maestro del bus del sistema, permitiendo realizar operaciones de lectura y escritura comunicándose así con todos los periféricos del chip conectados al bus del sistema, y al bus de periféricos. Además, el sistema incluye un monitor de Bus que permite observar el comportamiento del bus para obtener datos de operación y rendimiento. En la Figura 1 se presenta un diagrama simplificado de los diferentes elementos que conforman el SoC, indicando su relación con la plataforma de depuración.
La plataforma está compuesta por 3 módulos principales. A continuación se presentan con sus principales características:

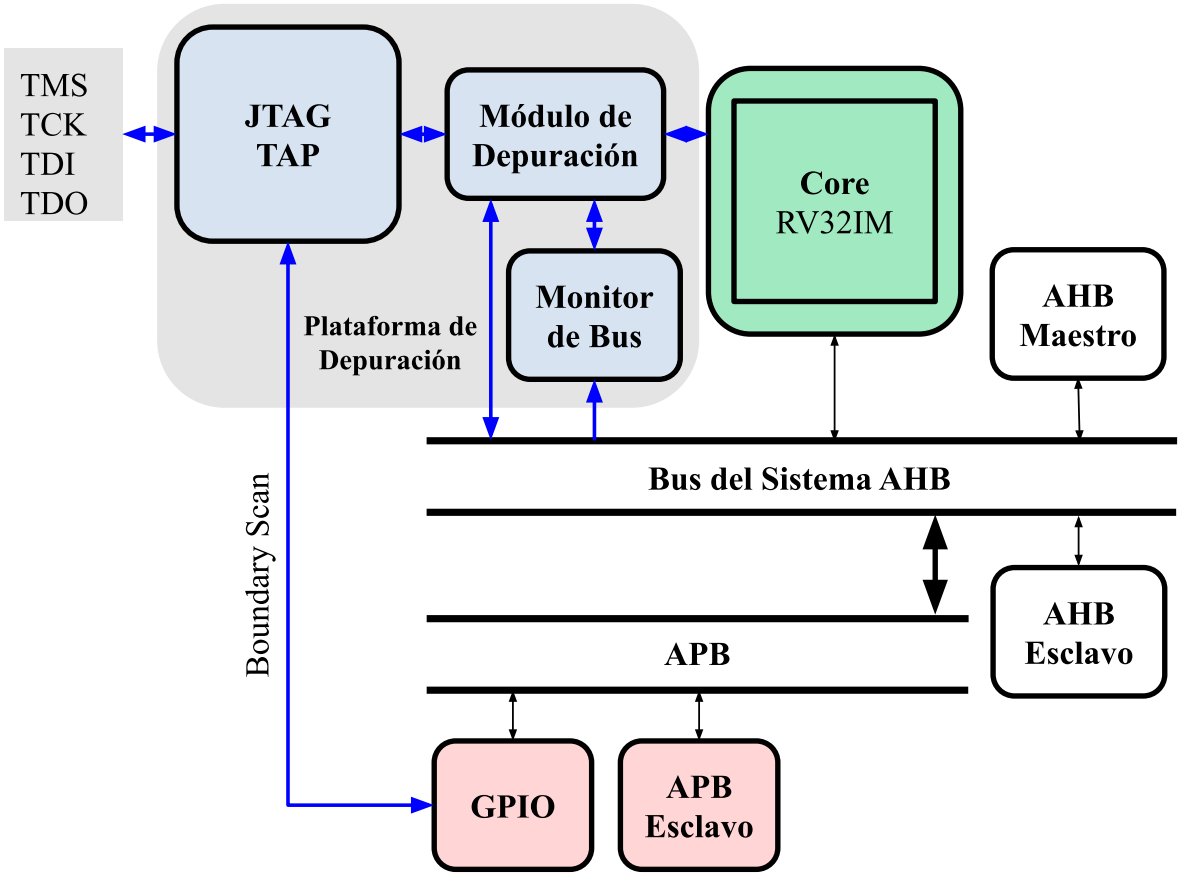
2.1. JTAG-TAP

Este módulo permite la comunicación entre el sistema a depurar (SoC) y un anfitrión de depuración externo, basada en comunicación JTAG de acuerdo al estándar IEEE 1149.1-2013 [2]. Este estándar permite inyectar serialmente datos e instrucciones al interior del SoC. Posteriormente estos datos son paralelizados y decodificados para poder ejecutar determinadas instrucciones basados en los datos inyectados.

---

[2] "IEEE Standard for Test Access Port and Boundary-Scan Architecture". En: *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)* (2013), págs. 1-444. DOI: 10.1109/IEEESTD.2013.6515989.

Figura 1. Diagrama de bloques de la plataforma de depuración.



Además cuenta con los registros de estado con información del comportamiento del módulo de depuración. De acuerdo al estándar, incluye registro para *boundary scan*, permitiendo el control de las entradas y salidas del SoC.

Para la comunicación entre el JTAG-TAP y el módulo de depuración, cuenta con la instrucción *DM_Access*. Esta instrucción permite escribir en un registro la operación a realizar (lectura o escritura), la dirección y el dato. Una mejora de esta plataforma puede encontrarse en la metodología para este acceso. El sistema actual al momento de realizar una operación de lectura, debe ingresar tanto la operación, la dirección y el valor, a pesar de que este ultimo es irrelevante en una operación de lectura, y el

resultado de lectura solo se puede obtener luego de terminar esta operación. Esto se podría optimizar aplicando un modelo similar al recomendado para el acceso a los registro en *The Nexus 5001 Forum*[3], donde almacenar primero la operación y la dirección, en caso de ser una operación de escritura, se pasan los datos a escribir; en caso de ser una operación de lectura, al ya haber pasado los datos de dirección ya se puede obtener el valor del registro.

Por otro lado, al hacer uso de comunicación JTAG, se debe al uso de 4 puertos para las conexiones, lo cual sería problemático en sistemas donde se tenga una cantidad reducida de pines. En estos casos para futuras implementaciones, se podría optar por la versión reducida del JTAG [4], la cual solo requiere de 2 conectores, y presenta las mismas capacidades de *boundary scan*.

2.2. MÓDULO DE DEPURACIÓN

El módulo de depuración (DM) es el encargado de las operaciones de depuración: control del procesador, ser maestro del bus del sistema y controlar el monitor de bus. El control del procesador permite detener, liberar y reiniciar el procesador. Del mismo modo, permite escribir instrucciones en una región de memoria reservada para depuración conocida como *program buffer*.

El DM presenta una interfaz maestra al bus AHB-lite del sistema, permitiendo al módulo de depuración leer y escribir a cualquiera de los periféricos esclavos conectados al bus principal del sistema. El control del monitor de bus permite acceder a las diferentes configuraciones para los filtros del monitor, así como a sus resultados

---

[3] "The Nexus 5001 Forum™ Standard for a Global Embedded Processor Debug Interface". En: *The Nexus 5001 Forum™* (2012), págs. 138-139.

[4] "IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture". En: *IEEE Std 1149.7-2009 - IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture* (2010). DOI: 10.1109/IEEESTD.2013.6515989.

para su posterior análisis.

Todas las operaciones de depuración se controlan a través de un conjunto de registros, a los que se accede desde el JTAP-TAP por un *register router*. Esta característica facilita el agregar nuevas funciones al depurador, de modo que permite agregar registros para almacenar las configuraciones de nuevas operaciones durante el diseño, sin la necesidad de modificar el JTAG-TAP.

2.3. MONITOR DE BUS

El monitor de bus permite realizar observar el comportamiento del bus y obtener valores de rendimiento en el bus. Para ello, este módulo se divide en 2 etapas: filtrado y análisis. Durante la etapa de filtrado, de acuerdo a una configuración especificada, dada por el módulo de depuración, captura las transacciones del bus, basado en parámetros como el ID del maestro, la dirección del esclavo, el tipo de operación que se realiza (escritura o lectura) y los datos enviados. En la etapa de análisis se toman las transacciones que superaron la etapa de filtrado y se registra su cantidad así como la latencia en términos de ciclos de reloj.

3. INTERFAZ DE COMUNICACIÓN A REGISTROS DE CONTROL Y ESTADO

En el capítulo anterior se presentó la plataforma de depuración, con una descripción de su funcionamiento, mostrando sus capacidades y las ventajas que ofrece. En este capítulo se presentan las nuevas funciones agregadas al sistema de depuración que permite el acceso a los CSR, así como las estrategias utilizadas para comunicar el modulo de depuración con el procesador del SoC.

Los CSR dentro de un procesador basado en arquitectura RISC-V, permiten el manejo de excepciones en los diferentes niveles de privilegios en los que puede ejecutarse el dispositivo [5]. Además, estos registros almacenan datos de identificación y rendimiento del procesador. El procesador utilizado en este trabajo usa el nivel de privilegio conocido como modo máquina. En este modo, se tiene acceso completo a la memoria, las entradas y salidas del dispositivo y funcionalidades de bajo nivel.

Dentro de la arquitectura implementada en el procesador del SoC, los CSR se encuentran agrupados dentro de un módulo conocido como *CSRFile*. Este módulo permite el acceso a estos registros mediante el *datapath* del procesador. Además define qué registros se pueden acceder de acuerdo al modo en el cual se ejecute el procesador y tiene conexión con las diferentes puntos del procesador para conocer su estado. El acceso a los CSR definidos por RISC-V, realizados desde el procesador son instrucciones que modifican el registro mientras se lee el valor almacenado en el registro.

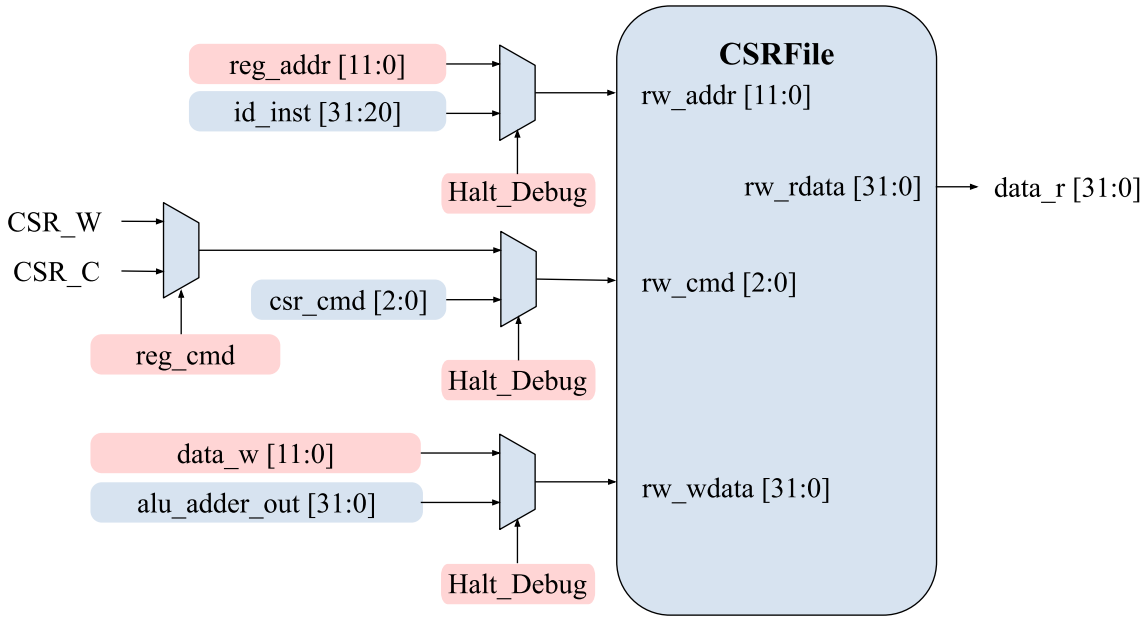
Con el fin de acceder a estos registros se diseñó la interfaz de la Figura 2, en azul las señales provenientes del *datapath* del procesador, en rojo las que vienen del módulo de depuración. Para evitar afectar el funcionamiento normal del procesador,

---

[5] PATTERSON, David y WATERMAN, Andrew. *The RISC-V READER: An Open Arquitecture Atlas*. Strawberry Canyon LLC, 2018.

solo es posible acceder a los CSR cuando el procesador se encuentra detenido, indicado por la señal *Halt_Debug*. Del módulo de depuración se recibe *reg_addr* con la dirección del registro a leer o escribir, *reg_cmd* que permite seleccionar el tipo de operación a realizar, *data_w* con el valor a escribir en el registro especificado y se envia al modulo de depuración *data_r* con el valor obtenido de la lectura de los registros.

Figura 2. Interfaz para el acceso a los registros de control y estado.

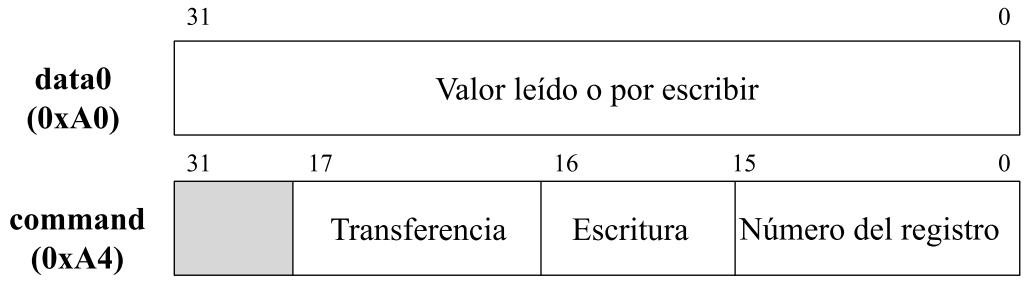


Para leer uno de los CSR se configura la entrada de datos como una mascara para bits, indicado por la señal CSR_C y se configura *data_w* con valor de cero para evitar modificar el registro. Para leer un registro desde el módulo depurador, con la señal CSR_W se selecciona la operación de escritura y se escribe el valor que se encuentre en *data_w*.

Para realizar estas operaciones se añadieron 2 registros dentro del depurador que se presentan en la Figura 3. El registro *data0* almacena el dato a escribir o el último dato leído de los CSR. El registro *command* en el encargado de controlar las opera-

ciones hacia los registros de control y estado. De este modo, cuando transferencia está en alto, si escritura está en alto, escribirá el valor de *data0* en el registro indicado en numero de registro. Si transferencia está en alto y escritura en bajo, leerá el registro indicado por número de registro y almacenara su valor en *data0*. Una vez se termina la lectura o la escritura del registro, el bit de transferencia se configura a cero para permitir nuevas operaciones hacia los registros.

Figura 3. Registros en el módulo de depuración para el manejo de los CSR.



3.1. CONTROL DE PUNTOS DE QUIEBRE

Los puntos de quiebre en los procesadores RISC-V se determinan modificado ciertos conjunto de CSR, conocidos como *Trigger Registers*. Estos registros son opcionales, por lo que su presencia y las capacidades que soportan pueden no ser universales. Para el caso del manejo de los puntos de quiebre se requieren al menos 2 registros: *tdata1*, donde se almacena la configuración de disparo del punto de quiebre y *tdata2* donde se guarda el valor a comparar. Las configuraciones soportadas por estos registros y sus direcciones son definidas por RISC-V[6].

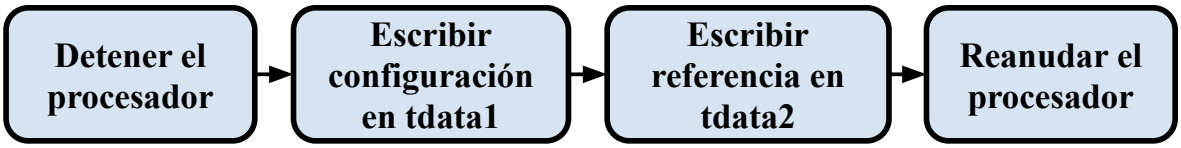
El tipo de disparo de los puntos de quiebre para el procesador que se encuentra en el SoC para este trabajo, puede configurarse de acuerdo a la dirección de la instruc-

---

[6] NEWSOME, Tim y WACHS, Megan. *RISC-V External Debug Support 0.13.2*. `https://riscv.org/specifications/debug-specification`. [En línea]. 2019.

ción, permitiendo ser configurada con una dirección especifica, o valores mayores o menores a la dirección especificada. Al momento de activarse un punto de quiebre, el procesador genera una excepción de *breakpoint*. Con el fin de detener el procesador, la plataforma de depuración captura esta excepción y detiene el procesador. Para añadir un punto de quiebre, se siguió el procedimiento de la Figura 4. En primer lugar se debe asegurar que el procesador esté detenido para poder acceder a los CSR. Luego almacena en *tdata1* la configuración para el disparador del punto de quiebre. A continuación el valor de referencia contra el cual se va a comparar se almacena en *tdata2*. Luego de configurado, se puede reanudar el funcionamiento del procesador.

Figura 4. Procedimiento para añadir un punto de quiebre en el procesador.



4. RESULTADOS DE SIMULACIÓN

Este capítulo presenta los resultados de simulación que permiten comprobar el funcionamiento de las características añadidas al sistema de depuración. Los diseños del SoC y la plataforma de depuración son descritos usando Chisel 3.0 [7], al igual que los nuevos módulos y características presentados anteriormente y añadidos a la plataforma. Estos diseños fueron sintetizados para generar archivos de Verilog, los cuales se utilizaron para realizar las simulaciones. Estas simulaciones fueron ejecutadas en la plataforma de SimVision[8].

4.1. SIMULACIÓN FUNCIONAL USANDO HDL

Para las pruebas del sistema de depuración se utilizó un *test bench* funcional de todo los componentes digitales del circuito, es decir el procesador, la plataforma de depuración, el sistema de buses AHB-lite, memoria RAM y periféricos digitales como GPIOs. Para realizar estas pruebas sobre el SoC, se siguió el esquema de verificación universal (UVM) descrita en SystemVerilog, que permite estimular los puertos de entrada del JTAG, desarrollada como parte del trabajo en [1].
En la Figura 5 se presenta la operación de escritura del CSR *MISA* [9]. Este registro almacena qué conjunto de instrucciones de la arquitectura (ISA), están soportados

---

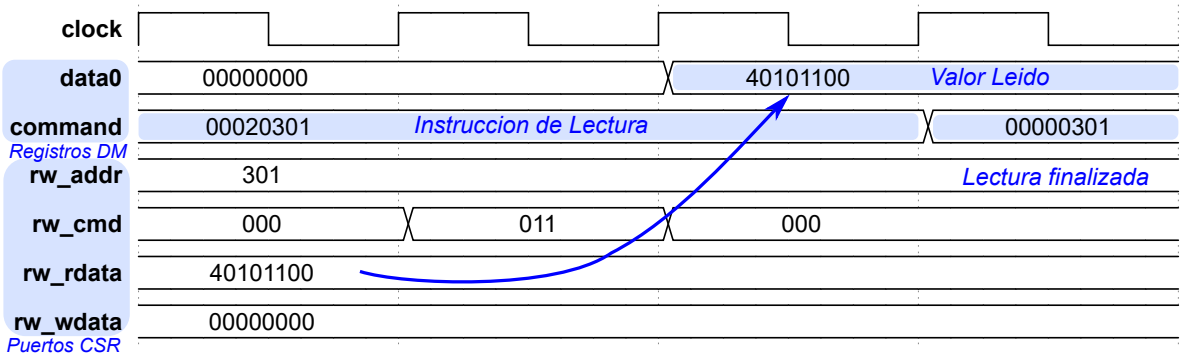
[7] U. OF CALIFORNIA IN BERKELEY. *Chisel - Constructing Hardware in a Scala Embedded Language*. `https://chisel.eecs.berkeley.edu/`. [En línea].

[8] CADENCE DESIGN SYSTEMS. *SimVision Debug A unified graphical debugging environment*. `http://https://www.cadence.com/en_US/home/tools/system-design-and-verification/debug-analysis/simvision-debug.html`. [En línea].

[9] WATERMAN, Andrew y ASANOVIĆ, Krste. *The RISC-V Instruction Set Manual; Volume II: Privileged Architecture.* `https://riscv.org/specifications/privileged-isa/`. [En línea]. 2019.

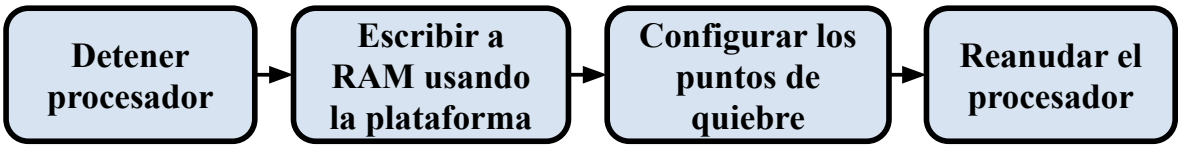
en el procesador.

Figura 5. Operación para la lectura del CSR *MISA*.



Para comprobar la configuración de los puntos de quiebre se diseñó una prueba que modificara las entradas del JTAG-TAP. En la Figura 6, se presenta el esquema para el desarrollo de la prueba. Primero se detiene el procesador, usando las plataforma de depuración. Luego se escribe un programa de prueba en la memoria RAM, aprovechando la capacidad de la plataforma de acceder a los periféricos del sistema a través del bus del sistema, con el fin de hacer avanzar el *program counter* (PC) del procesador. Luego se agregaran las configuraciones de los puntos de quiebre y se reanuda el procesador.

Figura 6. Esquema para pruebas sobre los puntos de quiebre.



Para añadir un punto de quiebre, se siguió el procedimiento presentado en 3.1. Con el procesador detenido, indicado por la señal *Halt_Debug*, se almacena en *tdata1* (0x7A1) la configuración para el disparador del punto de quiebre, que en este caso

corresponde a detener el procesador antes de ejecutar la instrucción que se encuentra en la posición de memoria definida en *tdata2*. Luego de almacenar este dato correctamente, se almacena en *tdata2* (0x7A2) el valor contra el cual se va a comparar. Luego de configurado, se puede reanudar el funcionamiento del procesador, de modo que antes de alcanzar la instrucción en la posición de memoria definida en *tdata2* indicada en el PC, se configura en alto la señal *Halt_Debug*, deteniendo la ejecución del programa.

4.2. VERILATOR

Verilator es una herramienta de software libre que permite convertir Verilog sintetizable en código C++/SystemC, que posteriormente es compilado y ejecutado, de modo que permite hacer simulaciones y generar archivo de trazas *cycle accurate*[10]. Además, soporta instrucciones *Direct Programming Interface* (DPI), de modo que la simulación puede enlazarse con programas externos escritos en C++.

Verilator es una herramienta muy popular tanto en la industria como en la academia. Proyectos como las plataformas Freedom de SiFive [11], así como el proyecto de OpenTitan de LowRISC [12] ofrecen en sus repositorios herramientas para simular sus diseños a través de verilator. Esta popularidad ofrece una amplia cantidad de ejemplos de uso, además de brindar confianza sobre sus resultados.

Para realizar simulaciones usando verilator se deben seguir 2 fases. La primera es convertir el código en verilog sintetizable en modelos en C++ usando la herramienta. Como segundo paso debe crearse un programa en C++ que funcione como *test bench*, donde debe instanciar el modelo. Debido a que en verilator el tiempo no

---

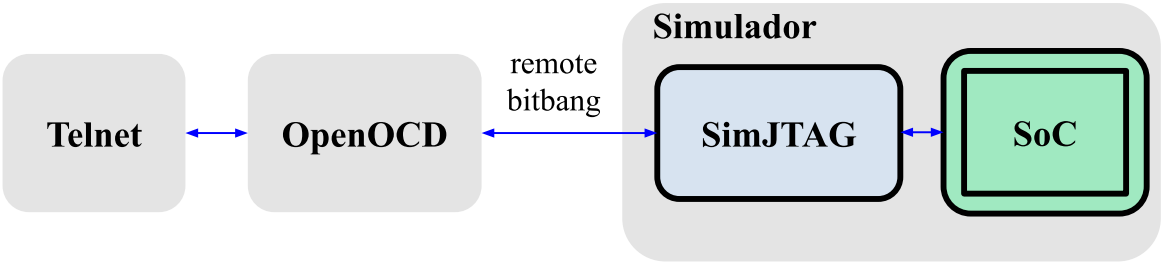
[10] VERIPOOL. *Welcome to Verilator*. `https://www.veripool.org/wiki/verilator`. [En línea].

[11] SIFIVE Inc. *Freedom*. `https://github.com/sifive/freedom`. [En línea]. 2019.

[12] LOWRISC Inc. *OpenTitan*. `https://github.com/lowRISC/opentitan`. [En línea]. 2017.

avanza, se debe invocar una función que modifique una señal de reloj y una función que evalúe las señales internas al ocurrir estos cambios en el reloj. Además, verilator permite exportar archivos con la formas de ondas de las señales internas del modelo simulado.

Figura 7. Esquema para simulación con Verilator y OpenOCD.



Aprovechando las características de verilator,las nuevas características agregadas en el módulo de depuración como parte de este trabajo fueron simuladas, junto con todo el SoC. Para permitir la comunicación del sistema con OpenOCD se agrego una interfaz JTAG simulada, la cual permite conectar los puerto de JTAG con un programa externo. En la Figura 7 se presenta un esquema general para la simulación del SoC usando las herramientas antes mencionadas. Open On-Chip Debugger (OpenOCD)[13] es una herramienta usada para la depuración, programación y testeo *boundary-scan* para dispositivos embebidos. OpenOCD funciona como interfaz entre el sistema a depurar y el servidor de depuración, que a su vez, también es gestionado por esta herramienta. Entre los diferentes adaptadores de depuración que soporta esta herramienta, es importante resaltar remote_bitbang. Este adaptador permite conectarnos a un JTAG virtual, permitiendo comunicarnos con el sistema de depuración y el SoC simulado. Esta comunicación se puede lograr usando un

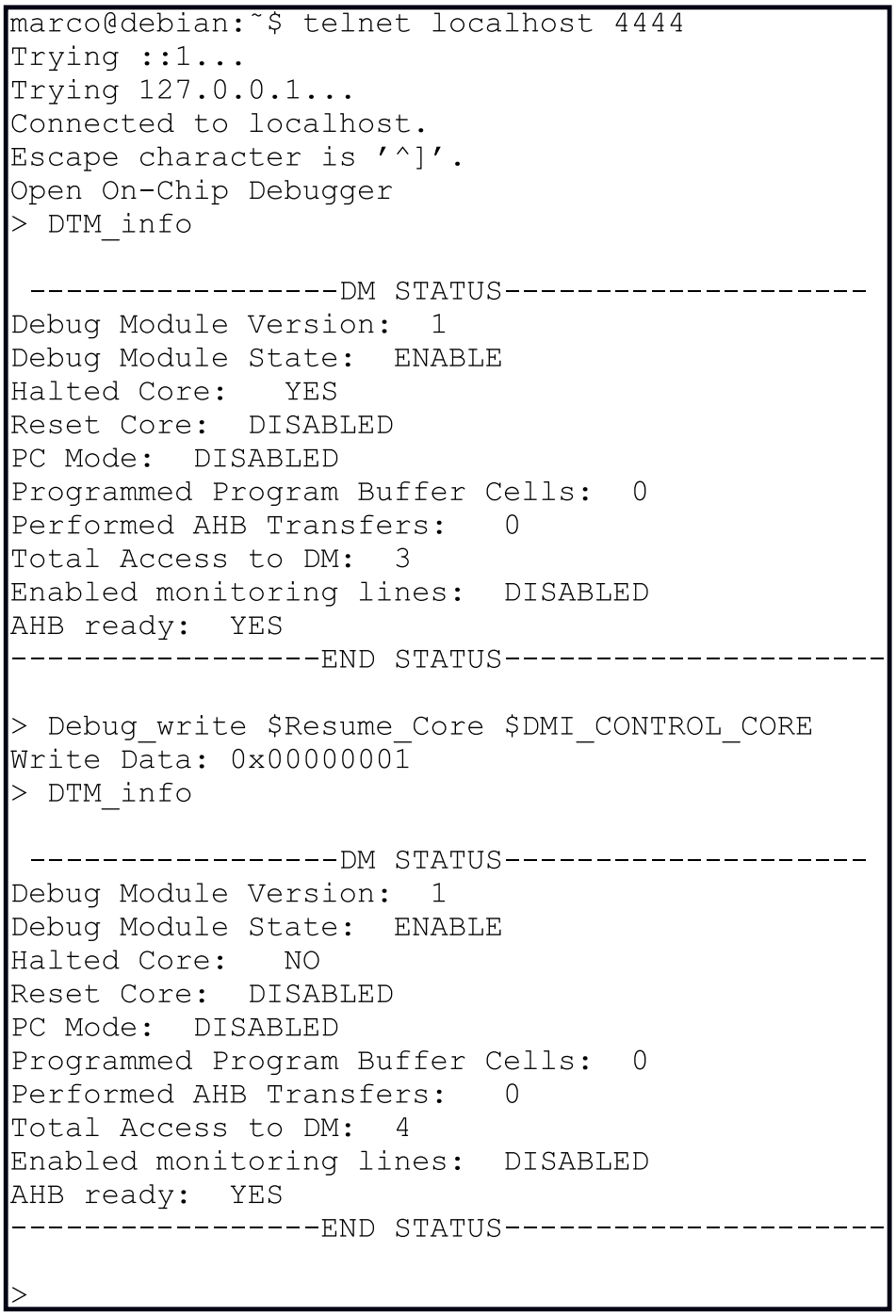
[13] BROWNELL, David. *Open On-Chip Debugger: OpenOCD User's Guide*. `http://openocd.org/doc-release/pdf/openocd.pdf/`. [En línea].

servidor Telnet. Además con el soporte de OpenOCD a secuencias de comandos en Tcl [14], se puede crear funciones que faciliten y automaticen las actividades de depuración.

En la Figura 8, se presenta la terminal del servidor Telnet para comunicarse con el SoC simulado y realizar operaciones de depuración. Con el fin de obtener información sobre el estado del sistema se creó la función DTM_info, la cual accede al registro de estado del módulo de depuración y muestra su información de manera legible. La funcion Debug_write permite escribir en un registro dentro del módulo de depuración.

[14] TCL CORE TEAM. *Welcome to the Tcl Developer Xchange!* `http://tcl.tk/`. [En línea]. 2019.

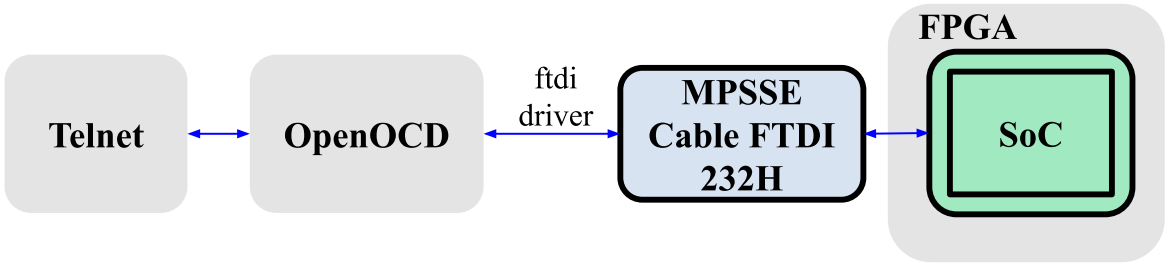
Figura 8. Terminal del Servidor Telnet para realizar actividades de depuracion.



5. RESULTADOS DE IMPLEMENTACIÓN EN FPGA

El SoC con la plataforma de depuración fue sintetizado e implementado en una FPGA Artix-7 35-T[15], con el fin de comprobar el acceso a los CSR, y el control de los puntos de quiebre del procesador. Los resultados obtenidos fueron capturados usando un analizador lógico integrado (ILA), que fue añadido durante la implementación en FPGA. Para controlar de manera externa SoC, se requirió un conector MSSPE FTDI 232H, que permite las comunicación entre USB y JTAG. Las operaciones de depuración se llevan a cabo de la misma manera que con verilator, modificando únicamente la interfaz entre OpenOCD y el SoC, de acuerdo a la Figura 9. En la Figuras 10, 11, 12 y 13 se muestra el control de los puntos de quiebre del procesador, de manera similar a la expuesta en la sección 4.1.

Figura 9. Esquema de conexión para las pruebas con FPGA y OpenOCD.



Los resultados de síntesis para la implementación en FPGA se resumen en la Tabla 1

[15] XILINX. *Artix-7 35T Arty FPGA Evaluation Kit.* `https://www.xilinx.com/products/boards-and-kits/arty.html#documentation`. [En línea].

Tabla 1. Resumen de síntesis de la implementación en FPGA del SoC.

Recurso	Cantidad
LUT totales	10461
LUT logicas	9767
LUT RAM	382
Latch S-R	312
Flip flop	11313

Figura 10. Operación para la configuración de los puntos de quiebre en FPGA.

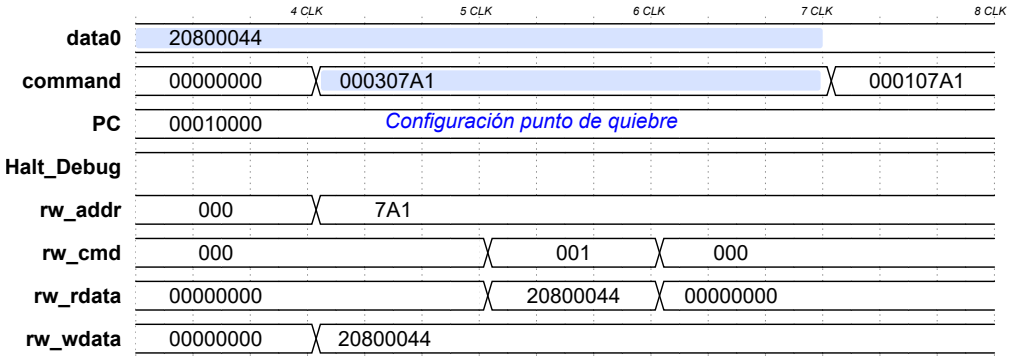


Figura 11. Configuración de la referencia del punto de quiebre en FPGA.

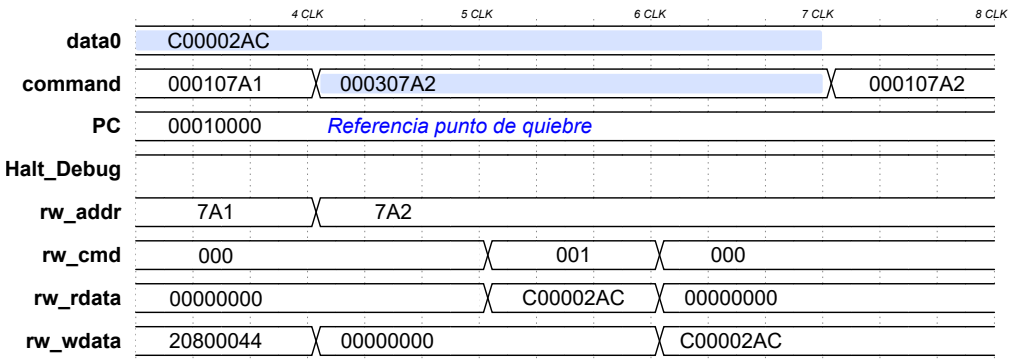


Figura 12. Procesador reanudando su funcionamiento.

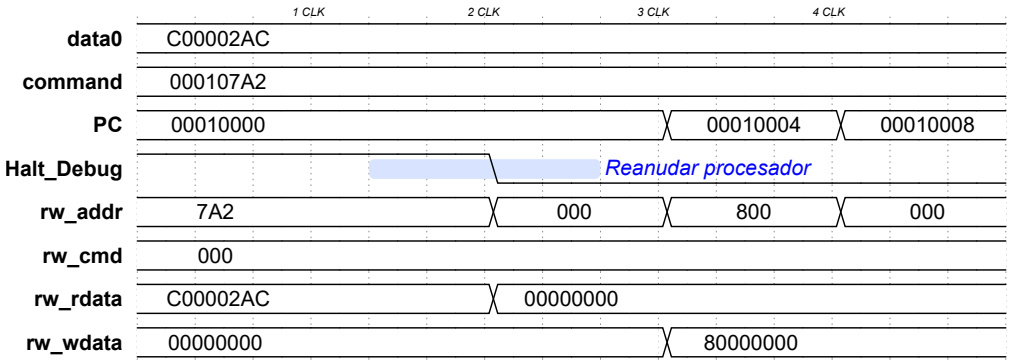
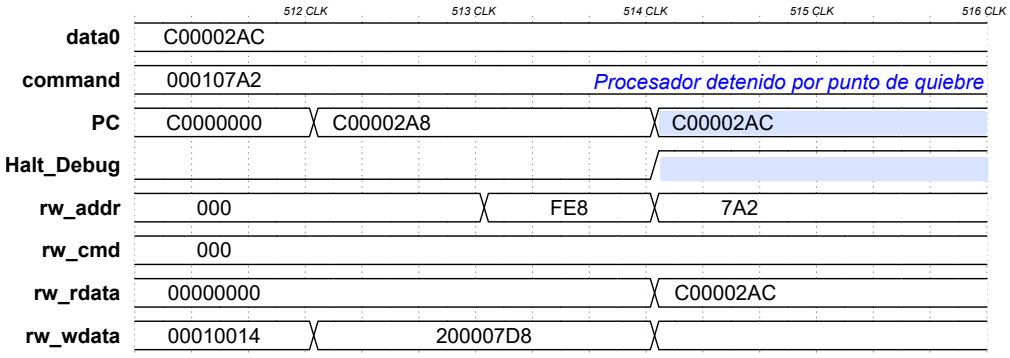


Figura 13. Procesador detenido al alcanzar el punto de quiebre.



6. TRABAJO FUTURO

Para futuras versiones del SoC, se pueden agregar más funcionalidades a los disparadores para los puntos de quiebre del sistema, como el activarlos de acuerdo al conteo de instrucciones o al ocurrir interrupciones o excepciones en el funcionamiento del procesador. Por otro lado, la plataforma OpenOCD ofrece diversas facilidades a la hora de depurar sistemas integrados en chip. En este trabajo su uso es básico, basado en el control de los estados del JTAG-TAP. Agregando una descripción más detallada del depurador y el SoC a OpenOCD, se pueden explotar más funcionalidades de esta herramienta, como podría ser el uso de GNU Debugger [16] como servidor de depuración, ofreciendo facilidad para el control de puntos de quiebre.

Por otro lado, verilator podría ofrecer otro tipo de interfaces, además de un JTAG virtual, para que durante las simulaciones permitiera conocer mas información del sistema, como el caso de GPIOs virtuales u otro tipo de interfaces de comunicación como UART o SPI.

---

[16] Inc FREE SOFTWARE FOUNDATION. *GNU. Gdb: The GNU Project Debugger.* `https://www.gnu.org/software/gdb/`. [En línea]. 2019.

7. CONCLUSIONES

De acuerdo a la revisión de la plataforma de depuración se presentaron las características y ventajas que ofrece la plataforma, entre ellas el acceso a periféricos del SoC, el control del dispositivo y la integración de un monitor de bus que permite encontrar valores de rendimiento del los diferentes periféricos sin afectar el funcionamiento del sistema.

Entre las posible mejoras se encontró que se podría optimizar el acceso desde el JTAG al módulo de depuración, evitando pasos innecesarios a la hora de leer datos.

Se diseñó una nueva capacidad de la plataforma de depuración, permitiendo el acceso a los Registros de control y estado. Fue puesta a prueba usando simulaciones funcionales a nivel de chip, así como la resultados de la implementación en FPGA.

De igual manera, con el acceso a los Registros de control y estado, se permite controlar los puntos de quiebre del procesador de acuerdo a la direcciones de memoria donde se ejecutan las instrucciones. Se presentaron simulaciones funcionales a nivel de chip, así como resultados de la implementación en FPGA, donde se muestra el control de los puntos de quiebre.

Tabla 2. Resumen de las capacidades de la plataforma de depuración.

Característica	En Tesis de maestría [17]	Este trabajo
Control del procesador	Presente	Presente
Maestro del bus del sistema	Presente	Presente
Monitoreo no intrusivo del Bus del sistema	Presente	Presente
Memoria reservada	Presente	Presente
Acceso a los registros de control y estado	No Presente	Presente
Configuración de puntos de ruptura	No presente	Presente

Adicionalmente, se presenta un modelo de simulación usando la herramienta verilator, permitiendo realizar pruebas sobre el SoC simulado, como si se tratase de un sistema real, comunicando una descripción de hardware con una interfaz de software.

En la Tabla 2 se presenta un resumen de las capacidades de la plataforma presentada en tesis de maestría[18], así como las nuevas características añadidas con este trabajo.

[18] RAMIREZ, Wilmer. “An Embedded Test Platform for System-On-A-Chip Interface ”. Tesis de Maestría (Maestría en Ingeniería, Área Ingeniería Electrónica). Colombia: Universidad Industrial de Santander, jul. de 2019.

BIBLIOGRAFÍA

BROWNELL, David. *Open On-Chip Debugger: OpenOCD User's Guide*. http://openocd.org/doc-release/pdf/openocd.pdf/. [En línea] (vid. pág. 25).

FREE SOFTWARE FOUNDATION, Inc. *GNU. Gdb: The GNU Project Debugger.* https://www.gnu.org/software/gdb/. [En línea]. 2019 (vid. pág. 31).

"IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture". En: *IEEE Std 1149.7-2009 - IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture* (2010). DOI: 10.1109/IEEESTD.2013.6515989 (vid. pág. 16).

"IEEE Standard for Test Access Port and Boundary-Scan Architecture". En: *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)* (2013), págs. 1-444. DOI: 10.1109/IEEESTD.2013.6515989 (vid. pág. 14).

Inc, LOWRISC. *OpenTitan*. https://github.com/lowRISC/opentitan. [En línea]. 2017 (vid. pág. 24).

Inc, SIFIVE. *Freedom*. https://github.com/sifive/freedom. [En línea]. 2019 (vid. pág. 24).

NEWSOME, Tim y WACHS, Megan. *RISC-V External Debug Support 0.13.2*. https://riscv.org/specifications/debug-specification. [En línea]. 2019 (vid. pág. 20).

PATTERSON, David y WATERMAN, Andrew. *The RISC-V READER: An Open Arquitecture Atlas*. Strawberry Canyon LLC, 2018 (vid. pág. 18).

RAMIREZ, Wilmer. “An Embedded Test Platform for System-On-A-Chip Interface ”. Tesis de Maestría (Maestría en Ingeniería, Área Ingeniería Electrónica). Colombia: Universidad Industrial de Santander, jul. de 2019 (vid. págs. 10, 11, 14, 22, 32, 33).

SYSTEMS, CADENCE DESIGN. *SimVision Debug A unified graphical debugging environment*. `http://https://www.cadence.com/en_US/home/tools/system-design-and-verification/debug-analysis/simvision-debug.html`. [En línea] (vid. pág. 22).

TEAM, TCL CORE. *Welcome to the Tcl Developer Xchange!* `http://tcl.tk/`. [En línea]. 2019 (vid. pág. 26).

“The Nexus 5001 Forum™ Standard for a Global Embedded Processor Debug Interface”. En: *The Nexus 5001 Forum™* (2012), págs. 138-139 (vid. pág. 16).

U. OF CALIFORNIA IN BERKELEY. *Chisel - Constructing Hardware in a Scala Embedded Language*. `https://chisel.eecs.berkeley.edu/`. [En línea] (vid. pág. 22).

VERIPOOL. *Welcome to Verilator*. `https://www.veripool.org/wiki/verilator`. [En línea] (vid. pág. 24).

WATERMAN, Andrew y ASANOVIĆ, Krste. *The RISC-V Instruction Set Manual; Volume II: Privileged Architecture.* `https://riscv.org/specifications/privileged-isa/`. [En línea]. 2019 (vid. pág. 22).

XILINX. *Artix-7 35T Arty FPGA Evaluation Kit.* `https://www.xilinx.com/products/boards-and-kits/arty.html#documentation`. [En línea] (vid. pág. 28).