Diseño de un *testbench* UVM para la verificación de un procesador RISC-V

Autor: Francisco Duque El Ayachi

Tutores: Dr. Valentín de Armas Sosa, Dr Félix B. Tobajas Guerrero

Máster Universitario Ingeniería de Telecomunicación, Julio de 2024

ULPGC Universidad de Las Palmas de Gran Canaria

EITE ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

INTRODUCCIÓN

La verificación de productos en sistemas de hardware digital está ganando mucha relevancia y se ha convertido en una de las etapas más largas del proceso de diseño. Por esta razón, es cada vez más importante encontrar estándares y métodos de verificación que faciliten estas tareas. Un ejemplo de ello es la metodología UVM (Universal Verification Methodology) [1].

OBJETIVO

El objetivo principal de este TFM consiste en el diseño y desarrollo de un *testbench* UVM usando como dispositivo a verificar un procesador con arquitectura RISC-V que implementa el juego de instrucciones básico y con la capacidad de realizar una cobertura funcional de cada una de ellas.

METODOLOGÍA

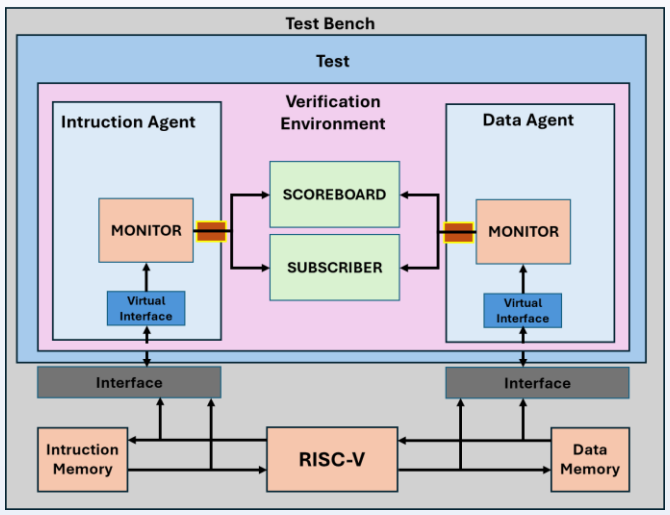

Figura I: Entorno Verificación UVM

El plan de verificación del DUV [2] se llevó a cabo siguiendo el entorno mostrado en la Figura I, Para ello, se extrajeron ambas memorias de instrucción y datos para interactuar con ellas de manera independiente y se configuró el entorno con agentes pasivos conectados a monitores que recogen los datos a través de interfaces virtuales vinculadas a las respectivas memorias, supervisando las transacciones entre memorias y procesador, por lo que el componente UVM *subscriber* será capaz de recibir los datos y generar los resultados de cobertura del test ejecutado, que en este caso, se llevó a cabo directamente desde el módulo TOP cargando en la memoria de instrucciones el programa a ejecutar donde fue necesario utilizar las *toolchains* propias de RISC-V para la generación de los estímulos al DUV mediante programas descritos en ensamblador.

RESULTADOS

Se llevó a cabo la ejecución de diferentes test y su visualización en QuestaSim mediante programas descritos en ensamblador, el ejemplo mostrado en la Figura II consiste en el cálculo del décimo número de la serie Fibonacci donde se comprobó el correcto funcionamiento de las diferentes unidades que conforman el pipeline del procesador RISC-V.

Name	Coverage	Goal	% of Goal	Status	Included
/riscv_pkg/subscriber	49.43%				
TYPE cg_R	40.69%	100	40.69%		
TYPE cg_I	61.32%	100	61.32%		
TYPE cg_S	43.47%	100	43.47%		
TYPE cg_B	52.22%	100	52.22%		

Figura II: Resultados cobertura para test Fibonacci

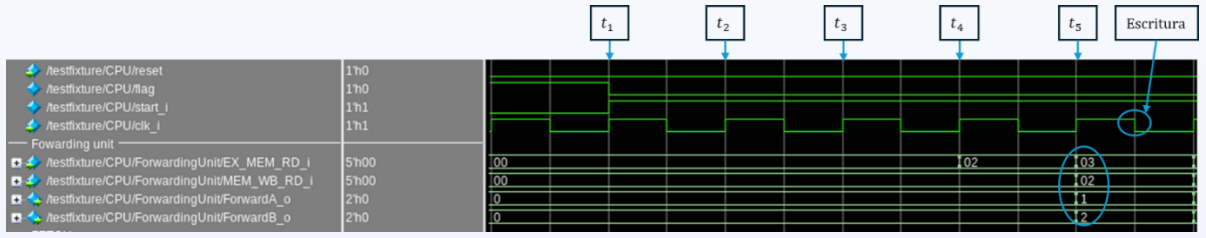

Figura II: Simulación de la ejecución unidad *Forwarding*

Se realizó un plan de cobertura con la estrategia de analizar la cobertura para cada tipo de instrucción soportado por el procesador RISC-V a verificar. Ejecutando diferentes tests comprobando el correcto funcionamiento del entorno desarrollado.

CONCLUSIONES

A la vista de los resultados obtenidos, todos los objetivos propuestos han sido cumplidos con éxito. E l desarrollo de un entorno de verificación UVM junto a la capacidad de reutilización del código de los componentes programados, resulta un método eficaz, donde cada día se impone más la necesidad de buscar estándares y métodos de verificación que simplifiquen y faciliten las funciones de verificación

REFERENCIAS

[1] “Universal Verification Methodology (UVM) 1.2 User’s G uide,” 2015, Accessed: J un. 01, 2022. [O nline]. Available:http://www.apache.org/licenses/.

[2] “GitHub - jasonlin316/RISC-V-CPU: A RISC-V 5-stage pipelined CPU that supports vector instructions. Tape-out with U18 technology.” Accessed: Jun. 20, 2024. [Online].Available: https://github.com/jasonlin316/RISC-V-CPU