Escuela de Ingeniería Electrónica
Licenciatura en Ingeniería Electrónica
EL-5811 Verificación Funcional de Circuitos Integrados
II Semestre 2023

TEC | Tecnológico de Costa Rica

Proyecto 3

Verificación funcional de un router bus usando UVM

Jeremy Córdoba, Jesús Rojas y Mario Montero

1. Introducción

En este informe, se presentan los resultados clave derivados de la implementación de un entorno de verificación funcional para un Router Bus usando UVM, utilizando la técnica de aleatorización controlada en capas en System Verilog. Este modelo de verificación contiene elementos esenciales, incluyendo un test, un generador, un scoreboard, un agente y un checker, además de 16 drivers y monitores destinados a gestionar todas las señales del Dispositivo bajo Prueba (DUT). Todos los componentes del ambiente estan estandarizados por la libreria de UVM.

Los escenarios del test se desarrollaron siguiendo un plan de pruebas previamente establecido, lo que permitió abordar exhaustivamente las capacidades del diseño. Además, la creación de una interfaz de programación de aplicaciones (API) resultó fundamental para facilitar la comunicación entre los distintos componentes que conforman este ambiente de verificación funcional.

2. Plan de pruebas

2.1. Diagrama del ambiente de verificación

En la Fig. 1 se presenta el diagrama del testebench realizado, en este se muestran todos los componentes de UVM así como las conexiones entre estos para llevar a cabo la verificación.

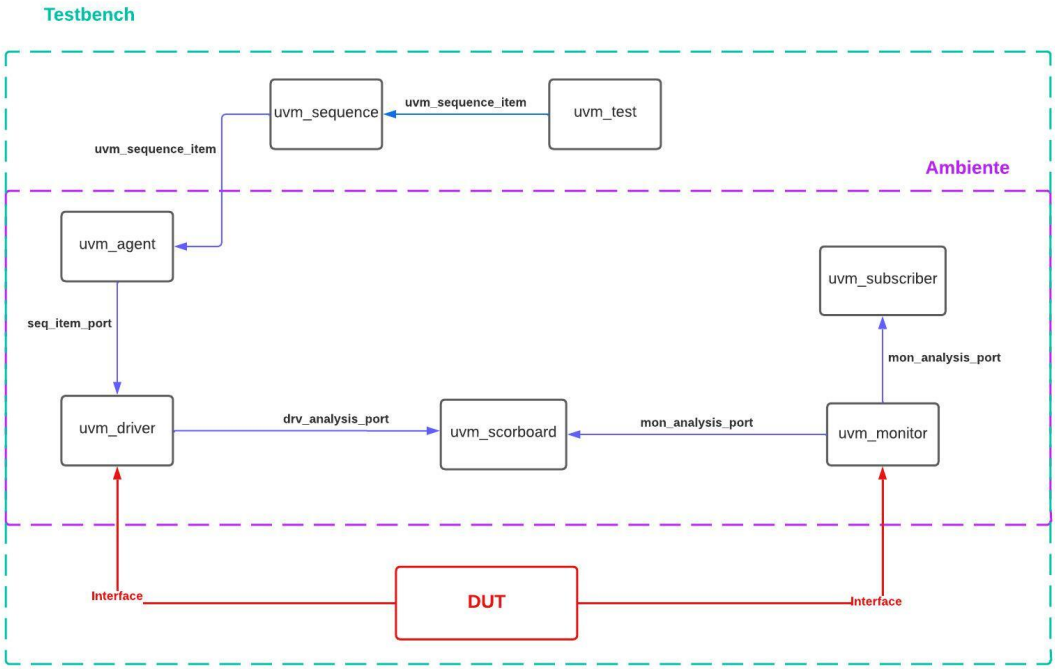

Fig. 1: Estructura del Ambiente de Verificación con UVM

2.2. Paquetes e interfaces de comunicación

1. **secuence_item_test_agent (pckg 1):** Se trata del item que se envia desde el secuenciador a los drivers e incluso a el scoreboard por medio de un puerto de analisis, los atributos de la clase son:

   - Parámetros:
     - ancho del paquete por defecto en 40.
   - Paquete.
   - nxt_jump.
   - mode.
   - row.
   - colum.
   - payload.
   - src.
   - id.
   - retardo_max
   - retardo
   - term_envio
   - term_recibido

- tiempo_envio
- tiempo_recibido
- cola_rutas

2. **sequence_item_scb (pckg 2):** Se trata de el item que maneja el scoreboard para recopilar informacion de todo el Test, los atributos de la clase son:
   - Parámetros:
     - ancho del paquete por defecto en 40.
   - paquete_enviado.
   - paquete_recibio.
   - term_tx.
   - term_rx.
   - tiempo_envio.
   - tiempo_recibido.
   - latencia.
   - completado.

3. Escenarios

En esta sección se definirán los diferentes escenarios de prueba para el DUT, tanto casos de uso común y casos de esquina.

En la Tabla 1 se muestra un listado de los casos generales de pruebas para el dispositivo bajo prueba (DUT).

En la Tabla 2 se muestra las pruebas para los casos de esquina para el dispositivo bajo prueba (DUT)

Tabla 1: Casos generales para el DUT

Prueba	Objetivo	Recursos
Transacción específica desde el terminal 0 al terminal específico id_row = 5 y id_colum = 4 con modo 1, retardo de 5 ciclos de reloj y payload = 8'haa	Constatar que un paquete enviado por un terminal específico, llegue correctamente a un terminal específico, cumpliendo con el modo de enrutamiento asignado.	El ambiente debe ser capaz de poder generar el paquete específico (que incluye target row, target column, modo y payload) y también el terminal desde el cual se realizará la transacción con su respectivo tiempo de envío.
Secuencia de transacciones aleatorias con numero de transacciones igual a 100 y retardo maximo de 40 con paquete aleatorio (target row aleatorio entre 0 y 5, target column entre 0 y 5, modo entre 0 y 1 y payload aleatorio)	Verificar que cada una de las transacciones generadas de manera aleatorio lleguen al destino cumpliendo con el modo y el retardo asignado aleatoriamente.	El ambiente debe ser capaz de generar el número de transacciones que se indique y aleatorizar el paquete y el terminal de envio para cada una de las transacciones de la secuencia.

Tabla 2: Casos de esquina para el DUT

Enviar desde cada uno de los terminales existentes a todos los demás, con modo especifico igual a 0, retardo de envío aleatorio entre 0 y 40 y payload aleatorio.	Verificar que todos los terminales en el sistema pueden realizar transacciones a todos los demás terminales, menos a sí mismos. Cumpliendo con el modo enrutamiento de 0, es decir primero filas, y el retardo asignado aleatoriamente.	El ambiente debe ser capaz generar paquetes de transacción con modo especifico, retardo de envío aleatorio y payload aleatorio y no debe permitir envíos a sí mismo.
Enviar desde cada uno de los terminales existentes a todos los demás, con modo especifico igual a 1, retardo de envío aleatorio entre 0 y 40 y payload aleatorio.	Verificar que todos los terminales en el sistema pueden realizar transacciones a todos los demás terminales, menos a sí mismos. Cumpliendo con el modo enrutamiento de 1, es decir primero columnas, y el retardo asignado aleatoriamente.	El ambiente debe ser capaz generar paquetes de transacción con modo especifico, retardo de envío aleatorio y payload aleatorio y no debe permitir envíos a sí
Enviar desde uno de los terminales existentes a un terminal que no este dentro del sistema. Terminal de envio $=6$, row $=2$ y colum $=6$, con modo especifico igual a 0, retardo de envío igual a 1 y payload 8'h99.	Verificar que un terminal especifico pueda enviar a un terminal que este fuera del sistema, dando seguimiento a la ruta que toma el paquete.	El ambiente debe ser capaz generar envíos a dispositivos que no existen en el sistema y debe ser capaz de detectar el envío e indicar que no corresponde a una transacción válida.

4. Indicaciones para correr la prueba

Se dispone de un shell script llamado *cmd.sh* para realizar la corrida de la prueba de aleatorización controlada en capas. Para correr el script se puede utilizar el siguiente comando en el directorio *Proyecto_3_Veri/src* en el que se encuentren los archivos fuente de la prueba:

```
#!/bin/bash
source cmd.sh
```

El archivo *cmd.sh* contienen los siguientes comando en donde se aplica la aleatorización de la profundidad de la FIFO del DUT utilizando el *package.sv*

```
source /mnt/vol_NFS_rh003/estudiantes/archivos_config/synopsys_tools.sh;

# Para variar la profundidad de las FIFOS se debe ingresa dicha cantidad desde la consola de linux

# Sin embargo, La prueba del DUT se corre siempre con la misma semilla (default)

read -p "Ingrese la profundidad de las FIFOs del DUT y del driver (entero entre 0 y 30): " profundidad

printf "package params_pkg;\n" > parameters_pkg.sv
printf "  parameter PROFUNDIDAD = %d;\n" $profundidad >> parameters_pkg.sv
printf "endpackage" >> parameters_pkg.sv

vcs -Mupdate tb_dut.sv -o salida -full64 -sverilog  -kdb -lca -debug_acc+all -debug_region+cell+encrypt
 -l log_test +lint=TFIPC-L -cm line+tgl+cond+fsm+branch+assert -ntb_opts uvm-1.2 -timescale=1ns/1ns;

./salida -cm line+tgl+cond+fsm+branch+assert;
~
~
```

Fig. 2: Contenido del archivo cmd.sh para correr la prueba

Dicho script le pedirá que ingrese en la línea de comandos la profundidad de las FIFOs del DUT y de los drivers. Para efectos de esta prueba, fueron utilizadas profundidades de entre 2 y 30, por lo que si se excede de esos límites, cabe la posibilidad de que no tenga suficiente tiempo para ejecutar todas las instrucciones que fueron escritas en el test.

5. Resultados de la prueba

Para la prueba se utilizó el script insertando la profundidad de las FIFOs en 10, con un identificador de broadcast de 0xff (valor por defecto) y un ancho de palabra de 40 bits. Las instrucciones generadas en el test se detallan a continuación:

5.1. Transacción Específica

En esta prueba se establece todo de forma específica en el paquete: el terminal de envío, terminal de destino, payload, modo y retardo, los resultados se muestran a continuación.

```
UVM_INFO @ 0: reporter [RNTST] Running test test...
UVM_INFO Test.sv(36) @ 0: uvm_test_top [TST] Primera Secuencia ---> Transaccion Especifica
UVM_INFO Driver.sv(44) @ 105: uvm_test_top.e0.a0.d[3] [DRV] Transaccion Recibida, pkt = 548000aa, modo = 1, retardo = 4, term_envio = 3, term_destino=11
UVM_INFO Scoreboard.sv(765) @ 170: uvm_test_top.e0.scb [SCB] Paquete: 00548000aa, ID_Recibido: 14, ID_Esperado:14 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 190: uvm_test_top.e0.scb [SCB] Paquete: 02548000aa, ID_Recibido: 24, ID_Esperado:24 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 210: uvm_test_top.e0.scb [SCB] Paquete: 02548000aa, ID_Recibido: 34, ID_Esperado:34 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 230: uvm_test_top.e0.scb [SCB] Paquete: 02548000aa, ID_Recibido: 44, ID_Esperado:44 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 230: uvm_test_top.e0.scb [SCB] El Paquete: 02548000aa ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 245: uvm_test_top.e0.a0.m[11] [MON] T=245 Se recibió un dato: 02548000aa
UVM_INFO Scoreboard.sv(143) @ 245: uvm_test_top.e0.scb [SCB] Paquete esperado = 00548000aa si coincide con el paquete recibido = 02548000aa
```

Fig. 3: Verificación de la transacción específica

Como se puede apreciar en la figura 3 el paquete específico $0x548000aa$ enviado desde la terminal 3 del DUT y pasa por cada router según su modo establecido, en este caso 1, hasta llegar a la terminal de salida 11 de ID 54.

5.2. Llenar FIFOS

En esta prueba se generan n transacciones, donde n es el tamaño de profundidad de las FIFOS con paquetes aleatorios para cada uno de los dispositivos.

```
UVM_INFO Test.sv(52) @ 1675: uvm_test_top [TST] Segunda Secuencia ---> Llenar FIFOs
UVM_INFO Driver.sv(44) @ 1685: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 105af492, modo = 0, retardo = 1, term_envio = 0, term_destino=4
UVM_INFO Scoreboard.sv(765) @ 1720: uvm_test_top.e0.scb [SCB] Paquete: 00105af492, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 1720: uvm_test_top.e0.scb [SCB] El Paquete: 00105af492 ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 1735: uvm_test_top.e0.a0.m[4] [MON] T=1735 Se recibió un dato: 03105af492
UVM_INFO Scoreboard.sv(143) @ 1735: uvm_test_top.e0.scb [SCB] Paquete esperado = 00105af492 si coincide con el paquete recibido = 03105af492
UVM_INFO Driver.sv(44) @ 2235: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 1004ed4a, modo = 0, retardo = 2, term_envio = 0, term_destino=4
UVM_INFO Scoreboard.sv(765) @ 2350: uvm_test_top.e0.scb [SCB] Paquete: 001004ed4a, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 2350: uvm_test_top.e0.scb [SCB] El Paquete: 001004ed4a ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 2365: uvm_test_top.e0.a0.m[4] [MON] T=2365 Se recibió un dato: 031004ed4a
UVM_INFO Scoreboard.sv(143) @ 2365: uvm_test_top.e0.scb [SCB] Paquete esperado = 001004ed4a si coincide con el paquete recibido = 031004ed4a
UVM_INFO Driver.sv(44) @ 2865: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 4229c58, modo = 0, retardo = 1, term_envio = 0, term_destino=3
UVM_INFO Scoreboard.sv(765) @ 2970: uvm_test_top.e0.scb [SCB] Paquete: 0004229c58, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 3050: uvm_test_top.e0.scb [SCB] Paquete: 0104229c58, ID_Recibido: 12, ID_Esperado:12 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 3130: uvm_test_top.e0.scb [SCB] Paquete: 0104229c58, ID_Recibido: 13, ID_Esperado:13 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 3200: uvm_test_top.e0.scb [SCB] Paquete: 0104229c58, ID_Recibido: 14, ID_Esperado:14 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 3200: uvm_test_top.e0.scb [SCB] El Paquete: 0104229c58 ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 3215: uvm_test_top.e0.a0.m[3] [MON] T=3215 Se recibió un dato: 0004229c58
UVM_INFO Scoreboard.sv(143) @ 3215: uvm_test_top.e0.scb [SCB] Paquete esperado = 0004229c58 si coincide con el paquete recibido = 0004229c58
UVM_INFO Driver.sv(44) @ 3485: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 40cf025a, modo = 1, retardo = 3, term_envio = 0, term_destino=7
```

Fig. 4: Prueba llenar fifos

5.3. Envío de hacia una terminal que no existe

Con esta prueba se verificó envíos a terminales que no existen en el sistema. En la Fig.5 se observa que en la prueba se esta enviando un paquete al dispositivo 22 (fuera del rango del sistema) el paquete llega a la terminal 13 por el router 24, pero el SCB lanza un error indicando que esa terminal no existe.

```
UVM_INFO Test.sv(61) @ 1675: uvm_test_top [TST] Tercera Secuencia ---> Transaccion a Terminal que no existe
UVM_INFO Driver.sv(44) @ 1685: uvm_test_top.e0.a0.d[6] [DRV] Transaccion Recibida, pkt = 26000099, modo = 0, retardo = 1, term_envio = 6, term_destino=22
UVM_INFO Scoreboard.sv(765) @ 1780: uvm_test_top.e0.scb [SCB] Paquete: 0026000099, ID_Recibido: 31, ID_Esperado:31 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 1860: uvm_test_top.e0.scb [SCB] Paquete: 0126000099, ID_Recibido: 32, ID_Esperado:32 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 1940: uvm_test_top.e0.scb [SCB] Paquete: 0126000099, ID_Recibido: 33, ID_Esperado:33 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 2010: uvm_test_top.e0.scb [SCB] Paquete: 0126000099, ID_Recibido: 34, ID_Esperado:34 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 2060: uvm_test_top.e0.scb [SCB] Paquete: 0026000099, ID_Recibido: 24, ID_Esperado:24 -> Ruta Correcta
UVM_INFO Monitor.sv(37) @ 2075: uvm_test_top.e0.a0.m[13] [MON] T=2075 Se recibió un dato: 0126000099
UVM_ERROR Scoreboard.sv(172) @ 2075: uvm_test_top.e0.scb [SCB] El id 22 no existe en el sistema
```

Fig. 5: Prueba envío a terminal desconocido

5.4. Envío de todas las terminales a todas las terminales con modo específico 0

Con esta prueba se verificó que todas las terminales fueran capaces de enviar paquetes a todas las demás terminales del sistema, el ambiente fue capaz de generar paquetes aleatorios para cada transacción que se generaba en cada terminal. En la Fig. 6 se muestra un fragmento del resultado en consola de la simulación hecha con esta prueba, en esta se observa como la terminal 0 envía paquetes aleatorios con el modo especificado en 0, para este caso, a todas las demás terminales, de igual manera la terminal y así con todas las demás.

```
UVM_INFO Test.sv(75) @ 1675: uvm_test_top [TST] Cuarta Secuencia ---> Todas a Todas Modo 0
UVM_INFO Driver.sv(44) @ 1685: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 1268bda, modo = 0, retardo = 0, term_envio = 0, term_destino=0
UVM_INFO Scoreboard.sv(765) @ 1710: uvm_test_top.e0.scb [SCB] Paquete: 0001268bda, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 1710: uvm_test_top.e0.scb [SCB] El Paquete: 0001268bda ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 1725: uvm_test_top.e0.a0.m[0] [MON] T=1725 Se recibió un dato: 0001268bda
UVM_INFO Scoreboard.sv(143) @ 1725: uvm_test_top.e0.scb [SCB] Paquete esperado = 0001268bda si coincide con el paquete recibido = 0001268bda
UVM_INFO Driver.sv(44) @ 2225: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 236fb8c, modo = 0, retardo = 5, term_envio = 0, term_destino=1
UVM_INFO Scoreboard.sv(765) @ 2370: uvm_test_top.e0.scb [SCB] Paquete: 000236fb8c, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 2450: uvm_test_top.e0.scb [SCB] Paquete: 010236fb8c, ID_Recibido: 12, ID_Esperado:12 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 2450: uvm_test_top.e0.scb [SCB] El Paquete: 010236fb8c ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 2465: uvm_test_top.e0.a0.m[1] [MON] T=2465 Se recibió un dato: 000236fb8c
UVM_INFO Scoreboard.sv(143) @ 2465: uvm_test_top.e0.scb [SCB] Paquete esperado = 000236fb8c si coincide con el paquete recibido = 000236fb8c
UVM_INFO Driver.sv(44) @ 2885: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 31131ce, modo = 0, retardo = 4, term_envio = 0, term_destino=2
UVM_INFO Scoreboard.sv(765) @ 3020: uvm_test_top.e0.scb [SCB] Paquete: 00031131ce, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 3110: uvm_test_top.e0.scb [SCB] Paquete: 01031131ce, ID_Recibido: 12, ID_Esperado:12 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 3190: uvm_test_top.e0.scb [SCB] Paquete: 01031131ce, ID_Recibido: 13, ID_Esperado:13 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 3190: uvm_test_top.e0.scb [SCB] El Paquete: 01031131ce ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 3205: uvm_test_top.e0.a0.m[2] [MON] T=3205 Se recibió un dato: 00031131ce
UVM_INFO Scoreboard.sv(143) @ 3205: uvm_test_top.e0.scb [SCB] Paquete esperado = 00031131ce si coincide con el paquete recibido = 00031131ce
UVM_INFO Driver.sv(44) @ 3535: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 41167e5, modo = 0, retardo = 1, term_envio = 0, term_destino=3
UVM_INFO Scoreboard.sv(765) @ 3640: uvm_test_top.e0.scb [SCB] Paquete: 00041167e5, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 3730: uvm_test_top.e0.scb [SCB] Paquete: 01041167e5, ID_Recibido: 12, ID_Esperado:12 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 3820: uvm_test_top.e0.scb [SCB] Paquete: 01041167e5, ID_Recibido: 13, ID_Esperado:13 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 3890: uvm_test_top.e0.scb [SCB] Paquete: 01041167e5, ID_Recibido: 14, ID_Esperado:14 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 3890: uvm_test_top.e0.scb [SCB] El Paquete: 01041167e5 ha llegado a su destino por la ruta correcta
```

Fig. 6: Verificación de la transacción de envío de todas a todas modo 0

5.5. Envío de todas las terminales a todas las terminales con modo específico 1

Con esta prueba se verificó que todas las terminales fueran capaces de enviar paquetes a todas las demás terminales del sistema, el ambiente fue capaz de generar paquetes aleatorios para cada transacción que se generaba en cada terminal. En la Fig. 7 se muestra un fragmento del resultado en consola de la simulación hecha con esta prueba, en esta se observa como la terminal 0 envía paquetes aleatorios con el modo especificado en 1, para este caso, a todas las demás terminales.

```
UVM_INFO Test.sv(84) @ 171035: uvm_test_top [TST] Quinta Secuencia ---> Todas a Todas Modo 1
UVM_INFO Driver.sv(44) @ 171045: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 1a7b5eb, modo = 1, retardo = 10, term_envio = 0, term_destino=0
UVM_INFO Scoreboard.sv(765) @ 171200: uvm_test_top.e0.scb [SCB] Paquete: 0001a7b5eb, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 171200: uvm_test_top.e0.scb [SCB] El Paquete: 0001a7b5eb ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 171215: uvm_test_top.e0.a0.m[0] [MON] T=171215 Se recibió un dato: 0001a7b5eb
UVM_INFO Scoreboard.sv(143) @ 171215: uvm_test_top.e0.scb [SCB] Paquete esperado = 0001a7b5eb si coincide con el paquete recibido = 0001a7b5eb
UVM_INFO Driver.sv(44) @ 171715: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 2bfe053, modo = 1, retardo = 6, term_envio = 0, term_destino=1
UVM_INFO Scoreboard.sv(765) @ 171870: uvm_test_top.e0.scb [SCB] Paquete: 0002bfe053, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 171900: uvm_test_top.e0.scb [SCB] Paquete: 0102bfe053, ID_Recibido: 12, ID_Esperado:12 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 171900: uvm_test_top.e0.scb [SCB] El Paquete: 0102bfe053 ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 171915: uvm_test_top.e0.a0.m[1] [MON] T=171915 Se recibió un dato: 0002bfe053
UVM_INFO Scoreboard.sv(143) @ 171915: uvm_test_top.e0.scb [SCB] Paquete esperado = 0002bfe053 si coincide con el paquete recibido = 0002bfe053
UVM_INFO Driver.sv(44) @ 172385: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 3e6a7de, modo = 1, retardo = 2, term_envio = 0, term_destino=2
UVM_INFO Scoreboard.sv(765) @ 172500: uvm_test_top.e0.scb [SCB] Paquete: 0003e6a7de, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 172590: uvm_test_top.e0.scb [SCB] Paquete: 0103e6a7de, ID_Recibido: 12, ID_Esperado:12 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 172620: uvm_test_top.e0.scb [SCB] Paquete: 0103e6a7de, ID_Recibido: 13, ID_Esperado:13 -> Ruta Correcta
UVM_INFO Scoreboard.sv(767) @ 172620: uvm_test_top.e0.scb [SCB] El Paquete: 0103e6a7de ha llegado a su destino por la ruta correcta
UVM_INFO Monitor.sv(37) @ 172635: uvm_test_top.e0.a0.m[2] [MON] T=172635 Se recibió un dato: 0003e6a7de
UVM_INFO Scoreboard.sv(143) @ 172635: uvm_test_top.e0.scb [SCB] Paquete esperado = 0003e6a7de si coincide con el paquete recibido = 0003e6a7de
UVM_INFO Driver.sv(44) @ 173015: uvm_test_top.e0.a0.d[0] [DRV] Transaccion Recibida, pkt = 4e9c5e4, modo = 1, retardo = 10, term_envio = 0, term_destino=3
UVM_INFO Scoreboard.sv(765) @ 173210: uvm_test_top.e0.scb [SCB] Paquete: 0004e9c5e4, ID_Recibido: 11, ID_Esperado:11 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 173300: uvm_test_top.e0.scb [SCB] Paquete: 0104e9c5e4, ID_Recibido: 12, ID_Esperado:12 -> Ruta Correcta
UVM_INFO Scoreboard.sv(765) @ 173390: uvm_test_top.e0.scb [SCB] Paquete: 0104e9c5e4, ID_Recibido: 13, ID_Esperado:13 -> Ruta Correcta
```

Fig. 7: Verificación de la transacción de envío de todas a todas modo 1

6. Ancho de banda y retraso promedio de la transmisión

A partir de los informes que genera el Scoreboard con cada prueba, se llevaron a cabo diversas pruebas con distinta profundidad de FIFOs, sin embargo, manteniendo constantemente el ancho de palabra en 40 bits. Por medio de los archivos .csv se generaron los gráficos de las Figs. 8, 9, 10 y 11.

La Fig. 8 corresponde a la tasa de bits máxima en la transmisión de paquetes de 40 bits con respecto a la profundidad de las FIFOs.

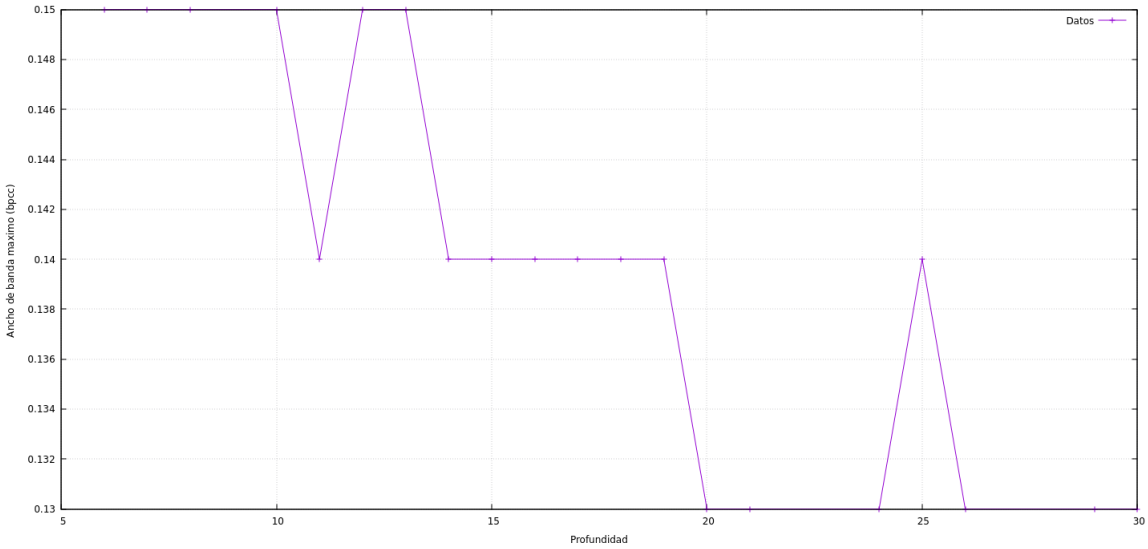

Fig. 8: Ancho de banda máximo

La Fig. 9 representa el ancho de banda minimo del DUT.

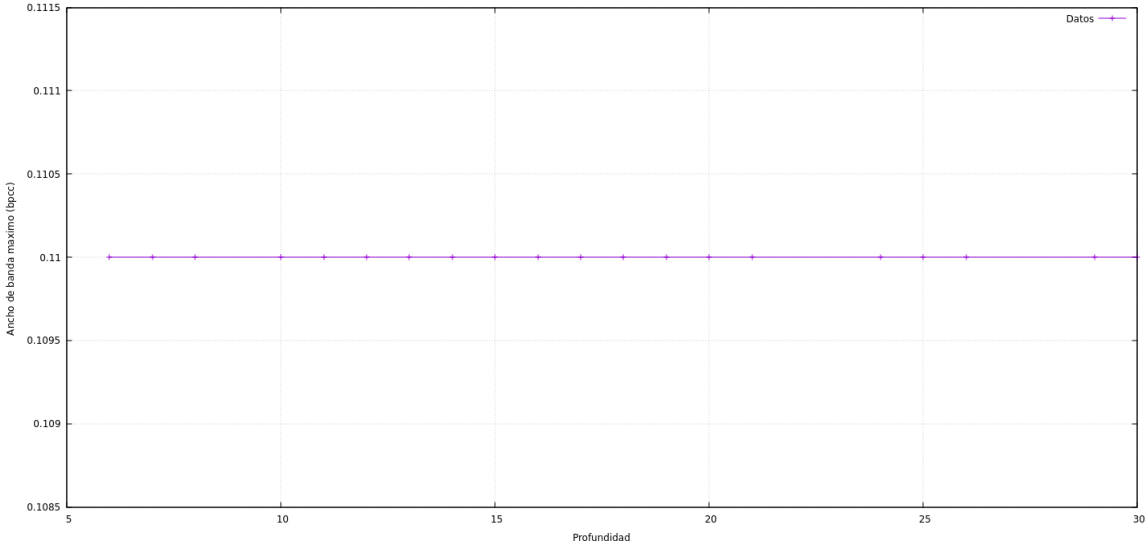

Fig. 9: Ancho de banda mínimo

La Fig. 10 representa el ancho de banda promedio del DUT.

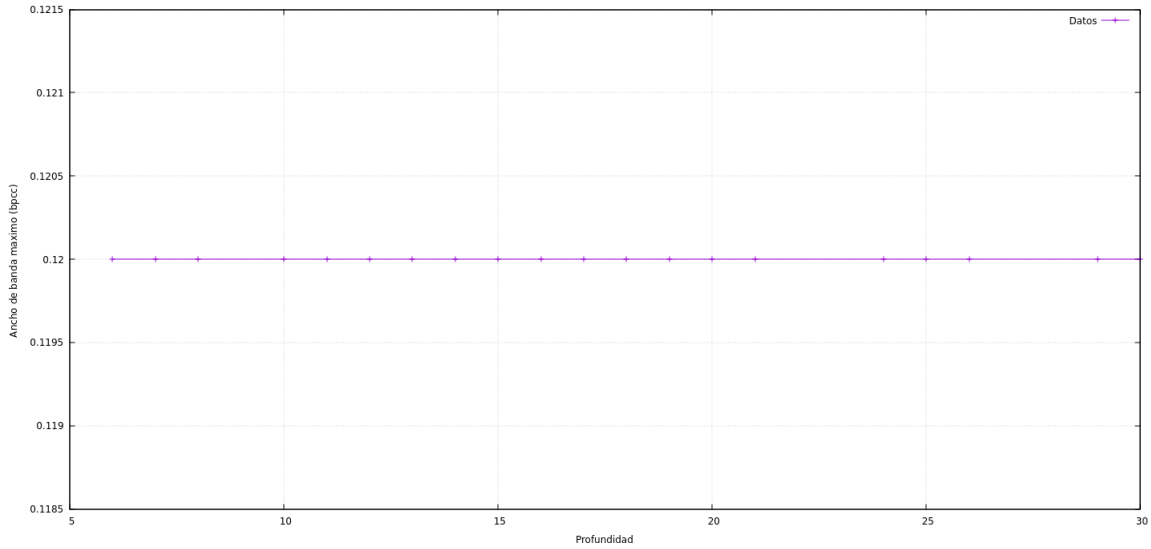

Fig. 10: Ancho de banda promedio

La Fig. 11 representa el retardo promedio del DUT.

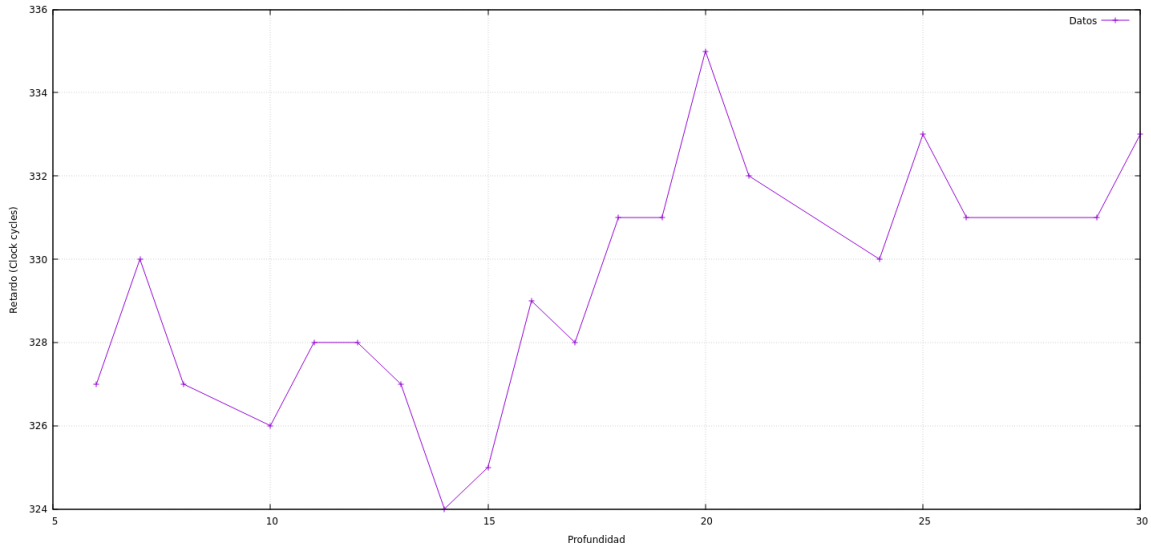

Fig. 11: Retardo promedio

7. Cobertura Funcional

Para la medición de la cobertura de las pruebas se utilizó un *uvm_subscriber* para recibir los datos del monitor mediante un puerto de analiss. En el grupo de cobertura se incluyó un *coverpoint*, el cual se utiliza para medir la cobertura de los *id* de los routers internos del DUT. Dentro de *coverpoint*, se definieron dos bins (contenedores) uno que representan valores específicos de los

routers internos, con esto se logró rastrear cuántas veces se alcanzan estos valores durante la simulación. Tambien se definió un coverpoint para todos los terminales posibles y modos posibles ademas un *cross* para los ids posibles y modos posibles.

En las Figs. 12 y se observan los resultados respecto al *coverage* de los routers del DUT.

```
UVM_INFO Coverage.sv(42) @ 490185: uvm_test_top.e0.cov [COV] Cobertura Funcional total es:  100.00 %
UVM_INFO Coverage.sv(43) @ 490185: uvm_test_top.e0.cov [COV] Cobertura de IDs posibles:  100.00 %
UVM_INFO Coverage.sv(44) @ 490185: uvm_test_top.e0.cov [COV] Cobertura de modos posibles:  100.00 %
UVM_INFO Coverage.sv(45) @ 490185: uvm_test_top.e0.cov [COV] Cobertura de terminales de recepción:  100.00 %
UVM_INFO Coverage.sv(46) @ 490185: uvm_test_top.e0.cov [COV] Cobertura de cruce de IDs con modos posibles:  100.00 %
```

Fig. 12: Cobertura funcional del DUT

8. Cobertura de toggle

Al observar la Fig. 13 se puede apreciar que la cobertura de linea es de un 95 % por lo que se puede decir afirmar que la mayoría de lineas de código se están ejecutando, además, juzgando la cobertura de toggle se afirma que todos los parámetros de entrada y salida del DUT pasaron de 1 a 0 y viceversa.

Name	Score	Line	Toggle	FSM	Condition	Branch	Assert
tb	81.13%	95.11%	90.46%	71.43%	77.76%	92.00%	60.0
_if	100.00%		100.00%				
DUT	81.13%	95.11%	90.46%	71.43%	77.76%	92.00%	60.0
uvm_custom_i...	30.91%	31.82%				30.00%	
bs_gnrtr	29.80%	35.49%	0.00%		42.67%	41.03%	
bs_gnrtr_n_rbtr	28.01%	33.58%	0.00%		39.88%	38.58%	
mem_latch	24.26%	50.00%	0.00%		0.00%	47.06%	
fifo_ltch	20.34%	48.95%	0.00%		0.00%	32.41%	
uvm_custom_i...	11.56%	5.12%				18.00%	
prll_bus_gnrtr_...	8.38%	38.89%	0.00%	0.00%	0.00%	3.03%	
ntrpt_cam_fifo	1.62%	4.53%	0.00%		0.00%	1.95%	
fifo_ltch_no_rst	1.50%	3.00%	0.00%		0.00%	3.00%	
prll_rgstr	0.00%	0.00%	0.00%			0.00%	
prll_bs_gnrtr_n...	0.00%	0.00%	0.00%		0.00%	0.00%	

Fig. 13: Cobertura de toggle y line

9. Link al repositorio

En siguiente link de github se encuentra el respositorio con todos los archivos del proyecto. `https://github.com/mariomont17/Proyecto_3_Veri`

10. Conclusiones

El entorno de verificación elaborado muestra ser capaz de ejercitar las capacidades del DUT de la manera más eficiente posible. Asimismo, el scoreboard/checker creado usando es capaz de calcular

y hacer seguimiento de la ruta que debe seguir cada paquete a partir de su fuente, destino y modo, lo que colaboró en la búsqueda de los errores del diseño.

El DUT es capaz de transmitir un volumen alto de datos, sin embargo, es necesario establecer un retardo mínimo que le permita transmitir cada uno de los datos sin que alguno se pierda en el proceso, ya que internamente no usa máquinas de estados, por lo que cada dato necesita un periodo en el cual pueda ser recibido y transmitido sin interferencia de alguno otro dato.

Con el *coverage* que se muestra en la Fig. 12 se concluye que al menos todos las terminales de recepcion y routers id internos tienen una cobertura funcional del 100 %, también se verifica que el cruce de todos los ids posibles y modos posibles se cubren totalmente.

Por ultimo la cobertura de Toggle y line que se aprecia en la Fig. 13 se obtuvo un porcentaje de 90.46 % para toggle y un 95.11 % para line, con la exclusion de bits innecesario como los del *nxt_jump*, con este se concluye que las pruebas realizada cubren satisfactoriamente la verificación del dispositivo.