Plan de verificación: PicoRV32-IMC

Autor:

José Mario Navarro Bejarano

Fecha:

Noviembre 2024

Revisiones

Número de Revisión	Fecha de revisión	Descripción
1	26 de agosto del 2024	Se crea la primera versión del plan de verificación, incluyendo las funciones para verificar deseadas y la estructura planeada.
2	16 de noviembre del 2024	Se realizan pequeños cambios en diferentes secciones del documento para reflejar los cambios realizados. Se agregan los puntos de cobertura funcional en las secciones correspondientes.
3	28 de noviembre del 2024	Se agregan las aserciones implementadas y la sección de resultados.

1. Contenido

2. Estrategia de verificación:

La estrategia de verificación se enfocará en generar de forma aleatoria todas las instrucciones y sus debidos parámetros, pero bajo restricciones para generar en su mayoría los valores de los campos con base en la especificación del ISA (como generar de forma aleatoria en cuál registro almacenar un valor, no generar constantemente registros que no existen). Como segunda fuente de instrucciones se utilizarán instrucciones en hexadecimal que se podrán leer desde un documento, para así generar una cantidad mayor de pruebas y permitir comprobar las instrucciones J y B de forma más eficiente desde un código en alto nivel.

Al realizar una verificación self-checking se crea un modelo de referencia que permite predecir los cambios en los valores de los registros internos, al procesar las instrucciones en el momento en que se envían al DUT y luego comprobando las predicciones con los registros internos de este. También se predicen los saltos y se comprueba si se realizan de forma correcta, al igual que comprobar que los valores que se cargan desde memoria son los correctos.

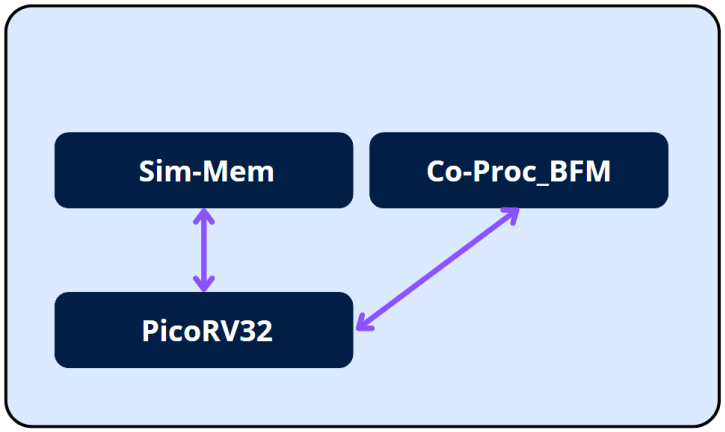
Como enfoque de prueba se utilizará grey box. Esta metodología permite realizar pruebas analizando las señales externas que posee el DUT, pero también permite realizar análisis sobre la estructura interna, por ejemplo, acceder a la información de cada registro. Esto se requiere para comprobar, entre otras cosas, los resultados de las instrucciones tipo R.

3. Niveles de verificación

a. Jerarquía de verificación

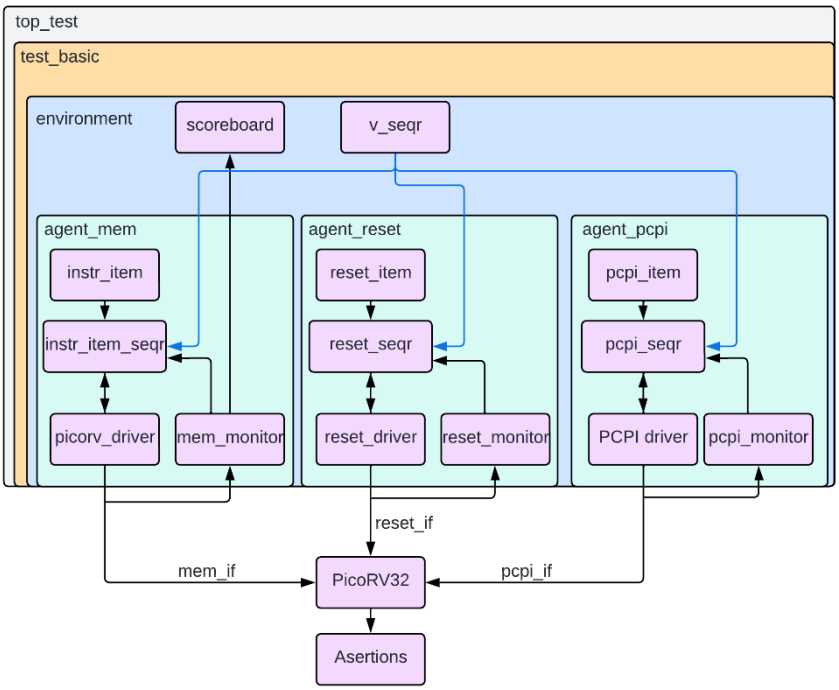
En este proyecto se busca realizar la verificación del PicoRV32-IM, por lo que se desea realizar una verificación a nivel de sistema completo, y no de las partes que componen a este. El DUT es un procesador que se encarga de ejecutar las instrucciones de RV32-IMC, sin embargo, las instrucciones C no serán verificadas. Este posee una serie de puertos que realizan la comunicación con la memoria, tanto para obtener instrucciones como para cargar y descargar valores de esta. Con base en esto la información será suministrada mediante una memoria simulada. Además de esto, es necesario implementar un pequeño BFM que se encargue de trabajar con las instrucciones que el DUT envía por la interfaz PCPI.

También se accederá a informacion interna de DUT como los valores almacenados en cada registro para permitir observar si algunas instrucciones se están ejecutando correctamente.



4. Ambiente de verificación:

a. Capas y Metodología



Para realizar la verificación funcional del DUT se utilizará una metodología por capas descrita en la anterior imagen. La conexión al DUT se realizará mediante tres interfaces. Una se conectará al agente de memoria, la siguiente contendrá la señal de reset y la tercera contendrá las señales correspondientes a la interfaz PCPI. En la interfaz conectada a la memoria se incluirán las señales que contienen la información sobre el valor almacenado en los registros internos (se hará una pequeña modificación para poder hacer visibles estas

señales desde el exterior). A partir de esta interfaz el monitor de memoria se encargará de observar los valores en cada señal, y los enviará al scoreboard para las correspondientes comparaciones.
Para la generación de los estímulos, se cuenta con dos métodos:

- Se generará de forma aleatoria cada tipo de instrucción. Luego de generar cuál instrucción se va a colocar, se generará de forma aleatoria los valores para los campos de cada instrucción (como números de registro o valores inmediatos).
- Se obtendrán listas de instrucciones desde archivos .txt que serán compilados a partir de archivos de prueba escritos en C, esto permitirá generar secuencias de prueba mejor estructuras y que se simularán el uso real.

Estas instrucciones generadas serán colocadas en la memoria por el monitor (debido a la estructura, la memoria se ubica en el monitor) para que el DUT los pueda acceder cuando lo requiera.

5. Alcance del plan de verificación

El DUT posee el conjunto de instrucciones básicas de RV32I y posee dos extensiones M y C, que agregan un conjunto extra de instrucciones. La mayoría de las instrucciones contenidas en estos conjuntos deben ser verificadas, y debe asegurarse de que el núcleo las ejecuta correctamente, además de que manipula correctamente los registros y las señales externas. También se debe comprobar que el núcleo trata correctamente con instrucciones fuera de su alcance, sea el caso de instrucciones que deben dirigirse a la interfaz PCPI para ser tratadas por otro procesador o que se levante la señal “trap” indicando que se ha generado una excepción o error.
Se excluyen de este proyecto la extensión C, las instrucciones FENCE y SYSTEM del set básico I y la versión simplificada E, debido a que se requeriría más tiempo del disponible para realizar su verificación
Dentro del picorv32.v tambien existen otros módulos que no serán tomados en cuenta para este proyecto, el único que será verificado es el que se denomina “picorv32” y todos los módulos que son instanciados dentro de este, como el módulo para multiplicar y dividir.

6. Requerimientos de cobertura y métricas.

a. Descripción del tipo de cobertura estructural y funcional a utilizar.

Para comprobar que la verificación se está realizando correctamente se espera realizar un análisis de cobertura estructural en donde se pueda comprobar que toda la parte del diseño bajo verificación está siendo estimulado. Se espera recolectar, principalmente, cobertura de transiciones, funcional y cobertura de linea.
Para la cobertura funcional se espera crear una serie de puntos de cobertura que permitan medir si todas las instrucciones están siendo generadas y que además se está accediendo

a todos los registros como registros fuente y como registros destino, además de verificar que las instrucciones que cargan valores inmediatos utilizan valores máximos y mínimos. También se hará uso de aserciones para comprobar que las señales se comportan de forma correcta con base en la documentación del proyecto en GitHub. Con especial atención a las señales que se comunican con la memoria.

b. Metas del proyecto.

La meta principal de este proyecto es realizar una verificación de las funcionalidades básicas del DUT, es decir, se espera ejecutar todas las instrucciones seleccionadas y hacer uso de todos los registros del sistema.
Se espera alcanzar una cobertura funcional del 100% y una cobertura estructural lo más alta que sea posible dentro de los alcances de este proyecto (existen dentro del RTL funciones extra como compatibilidad con AXI y WB que no son incluidas en este plan).

c. Aserciones

Para comprobar que las señales se comportan de la forma en que se indica en la documentación del diseño, se implementan algunas aserciones para ser comprobadas durante todo el tiempo de simulación.

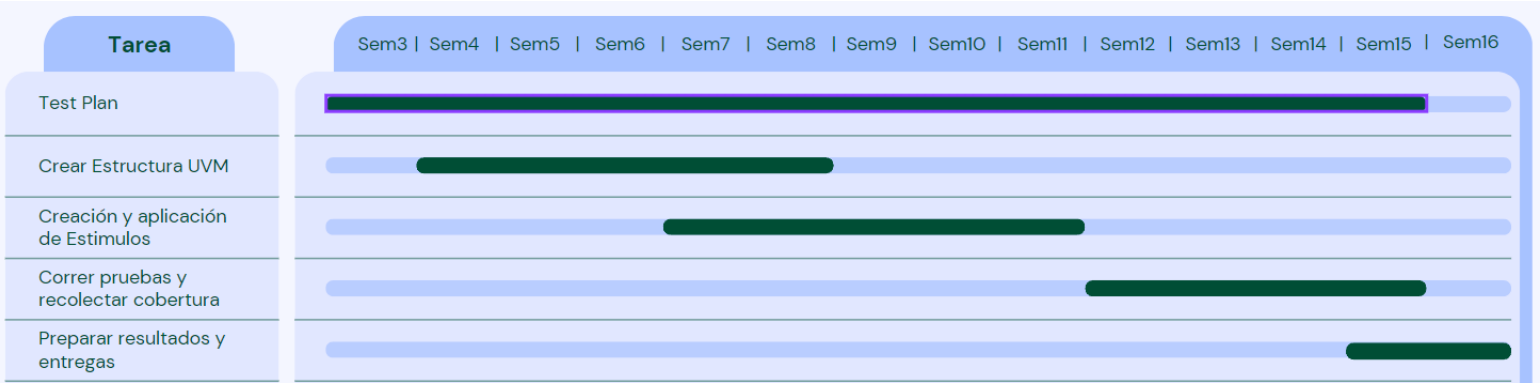
Nombre de la Aserción	Descripción
asrt_mem_instr_stable asrt_mem_addr_stable asrt_mem_wdata_stable asrt_mem_wstrb_stable asrt_pcpi_insn_stable asrt_pcpi_valid_stable asrt_pcpi_rs1_stable asrt_pcpi_rs2_stable	Comprueba que las señales de salida del DUT se mantengan estables siempre que mem_valid sea 1
asrt_mem_valid_use	Comprueba que siempre que la señal mem_valid se active, se debe activar la señal mem_instr indicando una lectura de instrucción o sino, la señal mem_wstrb debe contener uno de los 8 posibles valores, indicando, una lectura o una escritura de datos a memoria: 0000, 1111, 1100, 0011, 1000, 0100, 0010, 0001
asrt_mem_la_addr_use asrt_mem_la_wdata_use asrt_mem_la_wstrb_use asrt_mem_la_read_use	Comprueban el correcto funcionamiento de la interfaz look-ahead, es decir, que la interfaz coloca un ciclo antes los mismos valores que serán puestos en la interfaz de memoria.
asrt_mem_la_write_active asrt_mem_la_read_active	Comprueba que las señales mem_la_write y mem_la_read sí se activan un ciclo de reloj antes de activar mem_valid

7. Herramientas requeridas.

Este proyecto, al ser una verificación funcional reducida, únicamente requiere hacer uso de la herramienta Dsim proporcionado por Metrics, la cual es de uso gratuito (en su version desktop).

8. Calendarización.

Para la ejecución de este proyecto se ha definido un cronograma con tareas básicas que deberán ser completadas preferiblemente en los plazos establecidos. Aunque por la naturaleza de un proyecto de verificación, algunas tareas pueden extenderse más allá de su plazo establecido, sin implicar esto atrasos en el proceso.



9. RV32I Base Integer Instruction Set:

a. Instrucciones tipo R.

Función para verificar:	Instrucciones tipo R
Sección de la especificación:	2
Descripción:	El conjunto básico de instrucciones posee instrucciones de tipo R, que realizan operaciones entre registros. Estas son: ADD, SUB, SLL, SLT, SLTU, XOR, SRL, SRA, OR, AND.
Prioridad:	Crítica
Autor (Dueño de la función):	Mario Navarro
Requerimientos del banco de pruebas:	Generar instrucciones con valores aleatorios pero restringidos para cada parte de la instrucción, exceptuando el OPCODE.
Pruebas:	Realizar múltiples operaciones que incluyan el mayor rango posible de valores almacenados en todos los registros.
Criterio de chequeo:	El valor resultante de cada operación debe quedar almacenado correctamente en el registro de destino.
Cobertura funcional:	-Todos los registros son accedidos como registros fuente y como registros destino. -Todas las operaciones son ejecutadas (incluyendo operandos con valores extremos).
Puntos de cobertura asociados:	addXrd - addXrs1 - addXrs2 subXrd - subXrs1 - subXrs2 sllXrd - sllXrs1 - sllXrs2 sltXrd - sltXrs1 - sltXrs2 sltuXrd - sltuXrs1 - sltuXrs2 xorXrd - xorXrs1 - xorXrs2 srlXrd - srlXrs1 - srlXrs2 sraXrd - sraXrs1 - sraXrs2 orXrd - orXrs1 - orXrs2 andXrd - andXrs1 - andXrs2

b. Instrucciones tipo I.

Función para verificar:	Instrucciones tipo I
Sección de la especificación:	2
Descripción:	El conjunto básico de instrucciones posee instrucciones de tipo I, que realizan operaciones entre registros y valores inmediatos. También acceden a memoria para cargar datos a los registros. Estas son: LB, LH, LW, LBU, LHU, ADDI, SLTI, SLTIU, XORI, ORI, ANDI, SLLI, SRLI, SRAI.
Prioridad:	Crítica.
Autor (Dueño de la función):	Mario Navarro
Requerimientos del banco de pruebas:	Generar instrucciones con valores aleatorios pero restringidos para cada parte de la instrucción.
Pruebas:	Realizar múltiples operaciones sobre todos los registros.
Criterio de chequeo:	Todas las operaciones de carga deben realizar la operación correctamente con base en los valores almacenados en el registro rs1, y almacenar el resultado en el registro rd.
Cobertura funcional:	-Todos los registros son accedidos como registros fuente y como registros destino. -Todas las operaciones son ejecutadas
Observaciones:	Existen otras instrucciones que utilizan valores inmediatos, pero en esta sección únicamente se toman en cuenta las instrucciones que usan el formato I según se detalla en la página 104 de la especificación v2.2.
Puntos de cobertura asociados:	lbXrd - lbXrs1 lhXrd - lhXrs1 lwXrd - lwXrs1 lbuXrd - lbuXrs1 lhuXrd - lhuXrs1 addiXrd - addiXrs1 sltiXrd - sltiXrs1 sltiuXrd - sltiuXrs1 xoriXrd - xoriXrs1

	oriXrd - oriXrs1 andiXrd - andiXrs1 slliXrd - slliXrs1 - slliXrs2 srliXrd – srliXrs1 - srliXrs2 sraiXrd - sraiXrs1- sraiXrs2 addiXval_imm sltiXval_imm sltiuXval_imm xoriXval_imm oriXval_imm andiXval_imm lbXval_imm lhXval_imm lwXval_imm lbuXval_imm lhuXval_imm

c. Instrucciones tipo S.

Función para verificar:	Instrucciones tipo S
Sección de la especificación:	2
Descripción:	El conjunto básico de instrucciones posee instrucciones de tipo S, que acceden a memoria para almacenar valores. Estas son: SB, SH, SW.
Prioridad:	Crítica
Autor (Dueño de la función):	Mario Navarro
Requerimientos del banco de pruebas:	Generar instrucciones con valores aleatorios pero restringidos para cada parte de la instrucción.
Pruebas:	Realizar múltiples operaciones que utilicen todos los registros como rs1 y rs2. Se debe utilizar un amplio rango de números como valores inmediatos.
Criterio de chequeo:	Cuando se genere una instrucción de store, se debe comprobar que se accede al o los bytes correctos según el tipo de instrucción. Además, se debe comprobar que se genera correctamente la dirección de almacenamiento.
Cobertura funcional:	-Todos los registros son accedidos como registros rs1 y rs2. -Todas las operaciones son ejecutadas (incluyendo valores inmediatos extremos).
Puntos de cobertura asociados	sbXrs1 - sbXrs2 shXrs1 - shXrs2 swXrs1 - swXrs2 sbXimm1 - sbXimm2 shXimm1 - shXimm2 swXimm1 - swXimm2

d. Instrucciones tipo U.

Función para verificar:	Instrucciones tipo U
Sección de la especificación:	2
Descripción:	El conjunto básico de instrucciones posee instrucciones de tipo U, que utilizan un espacio de 20 bits para valores inmediatos que son muy grandes. Estas instrucciones son: LUI, AUIPC.
Prioridad:	Crítica
Autor (Dueño de la función):	Mario Navarro
Requerimientos del banco de pruebas:	Generar instrucciones con valores aleatorios pero restringidos para cada parte de la instrucción.
Pruebas:	Realizar múltiples operaciones que utilicen todos los registros como registro rd.
Criterio de chequeo:	Las instrucciones deben almacenar correctamente el valor cargado o calculado en el registro correspondiente.
Cobertura funcional:	-Todos los registros son accedidos como rd. -Se ejecutan todas las instrucciones.
Puntos de cobertura asociados:	luiXrd auipcXrd luiXimm auipcXimm

e. Instrucciones tipo B.

Función para verificar:	Instrucciones tipo B
Sección de la especificación:	2
Descripción:	El conjunto básico de instrucciones posee instrucciones de tipo B. Estas realizan saltos dentro del codigo con base en la comparación de dos registros rs1 y rs2. Estas instrucciones son: BEQ, BNE, BLT, BGE, BLTU, BGEU.
Prioridad:	Crítica
Autor (Dueño de la función):	Mario Navarro
Requerimientos del banco de pruebas:	Generar instrucciones con valores aleatorios pero restringidos para cada parte de la instrucción.
Pruebas:	Realizar múltiples operaciones con todos los registros como registros fuente.
Criterio de chequeo:	Las instrucciones deben realizar correctamente las operaciones de comparación, y con base en el resultado modificar el registro pc o mantenerlo sin modificaciones (solo cambiar por PC+4).
Cobertura funcional:	-Se ejecutan todas las instrucciones. -Todos los registros son accedidos como rs1 y rs2.
Puntos de cobertura asociados	beqXrs1 - beqXrs2 bneXrs1 - bneXrs2 bltXrs1 - bltXrs2 bgeXrs1 - bgeXrs2 bltuXrs1 - bltuXrs2 bgeuXrs1 - bgeuXrs2 beqXimm1 beqXimm2 bneXimm1 bneXimm2 bltXimm1 bltXimm2 bgeXimm1 bgeXimm2 bltuXimm1 bltuXimm2 bgeuXimm1 bgeuXimm2

f. Instrucciones tipo J.

Función para verificar:	Instrucciones tipo J
Sección de la especificación:	2
Descripción:	El conjunto básico de instrucciones posee instrucciones de tipo J. Estas modifican el flujo con base en un valor inmediato o con base en un registro y un valor inmediato. Estas instrucciones son: JAL, JALR.
Prioridad:	Crítica
Autor (Dueño de la función):	Mario Navarro
Requerimientos del banco de pruebas:	Generar instrucciones con valores aleatorios pero restringidos para cada parte de la instrucción.
Pruebas:	Realizar múltiples saltos dentro del codigo.
Criterio de chequeo:	Las instrucciones deben calcular correctamente la dirección del salto y modificar el registro PC. Además, deben almacenar correctamente la dirección de PC+4.
Cobertura funcional:	-Se ejecutan todas las instrucciones. -Se guarda la info en todos los registros de destino. -Se usan todos los registros como registros fuente rs1.
Puntos de cobertura asociados:	jalXrd jalrXrd - jalrXrs1 jalrXval_imm jalXimm

10. RV32M:

Función para verificar:	Extensión de instrucciones para la multiplicación y división de enteros.
Sección de la especificación:	6
Descripción:	Las instrucciones para multiplicación y división de enteros se proporcionan mediante la extensión estándar “M”. Estas son: MUL, MULH, MULHSU, MULHU, DIV, DIVU, REM, REMU.
Prioridad:	Alta
Autor (Dueño de la función):	Mario Navarro
Requerimientos del banco de pruebas:	Generar instrucciones con valores aleatorios pero restringidos para cada parte de la instrucción.
Pruebas:	Realizar múltiples operaciones que incluyan el mayor rango posible de valores almacenados en todos los registros.
Criterio de chequeo:	El valor resultante de cada operación debe quedar almacenado correctamente en el registro de destino.
Cobertura funcional:	-Todos los registros son accedidos como registros fuente y como registros destino. -Todas las operaciones son ejecutadas (incluyendo operandos con valores extremos).
Puntos de cobertura asociados	mulXrd - mulXrs1 - mulXrs2 mulhXrd - mulhXrs1 - mulhXrs2 mulhsuXrd - mulhsuXrs1 - mulhsuXrs2 mulhuXrd - mulhuXrs1 - mulhuXrs2 divXrd - divXrs1 - divXrs2 divuXrd - divuXrs1 - divuXrs2 remXrd - remXrs1 - remXrs2 remuXrd - remuXrs1 - remuXrs2

11. Interfaz PCPI:

Función para verificar:	Pico Co-Processor interfaz
Sección de la especificación:	Se documenta en el repositorio del proyecto.
Descripción:	La implementación PicoRV32 incluye una interfaz que permite enviar por esta las instrucciones que no son soportadas por el procesador, para que otro co-procesador pueda decodificarla y ejecutarla.
Prioridad:	Baja
Autor (Dueño de la función):	Mario Navarro
Requerimientos del banco de pruebas:	Generar instrucciones con valores aleatorios que resulten en instrucciones no soportadas, es decir, cualquier instrucción fuera de IMC o instrucciones de este conjunto, pero con datos incorrectos.
Pruebas:	Realizar múltiples operaciones con instrucciones aleatorias.
Criterio de chequeo:	Para cada instrucción errónea el procesador debería enviar la instrucción y los datos de los registros por la interfaz PCPI.
Cobertura funcional:	-Se ejecuta la instrucción fake, que se resuelve en el agente PCPI, haciendo uso de todos los registros como rd, rs1 y rs2.
Observaciones:	Se hará uso de las aserciones para comprobar que el comportamiento de las señales es correcto, según se indica en la documentación.
Puntos de cobertura asociados	fakexrd – fakexrs - fakexrs2

12. Resultados

a. Resultado de cobertura

Luego de realizar la simulación, se obtuvieron los siguientes resultados, según el reporte generado por Dsim Desktop:

Path	Functional	Line/block	Toggle	Assertion	Expression
$unit.funct_coverage	[100.0%]				
top_test		[100.0%]	[71.2%]		
top_test.dut		[68.6%]	[79.4%]		
top_test.dut.genblk1.pcpi_mul		[91.9%]	[99.1%]		
top_test.dut.genblk2.pcpi_div		[100.0%]	[88.6%]		
top_test.dut.genblk3		[100.0%]			
top_test.assrt			[87.2%]	[100.0%]	
top_test.intf_mem_if			[92.0%]		
top_test.intf_reset_if			[37.5%]		
top_test.intf_pcpi_if			[74.4%]		

Los dos puntos más relevantes de estos resultados es que la cobertura funcional alcanzó un valor de 100%, lo que indica que se logró ejecutar todas las características que se esperaba verificar. Se puede ver tambien un 100% en las aserciones, es decir, el comportamiento de las señales fue el mismo que el que se esperaba según la documentación del PicoRV32-IMC

En cuanto a la cobertura de linea y de toggle no fue posible alcanzar el 100% de cobertura, sin embargo, esto se debe a que muchas secciones del código estaban diseñadas para ejecutarse en otras condiciones, como por ejemplo cuando se usaran instrucciones comprimidas, cosa que no se realizó en este proyecto, o cosas mas pequeñas como modificación de algunos parámetros. En el reporte en formato .html se agregó en una nueva columna la justificación del porqué algunas señales o no hicieron toggle o del todo no se ejecutaron.

b. Resultado de errores

Luego de analizar los resultados finales, y sumado al analisis que se realizó mientras se desarrollaba el modelo de referencia y el entorno de verificación, se concluye que no existen bugs o errores en el funcionamiento del diseño, todas las características testeadas se ejecutan correctamente con base en la especificación del diseño y del conjunto de instrucciones de RiscV. Además, se ejecutan correctamente algunas otras características que no se detalla en la documentación del diseño como, por ejemplo, alinear las direcciones cuando se accede a direcciones desalineadas, o el “ignorar” instrucciones que se cargaron mientras se resolvía una instrucción de salto.