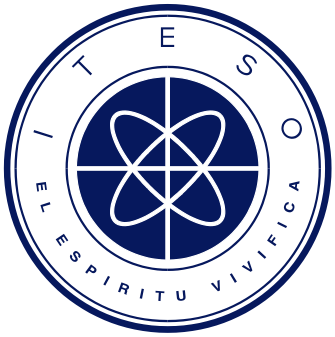
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

MAESTRÍA EN DISEÑO ELECTRÓNICO



REPORTE DE FORMACIÓN COMPLEMENTARIA EN ÁREA DE CONCENTRACIÓN EN DISEÑO DE SISTEMAS DIGITALES

Trabajo recepcional que para obtener el grado de

MAESTRA EN DISEÑO ELECTRÓNICO

Presenta: Roxana Suarez Lara

Asesor: Dr. Mariano Aguirre Hernandez
Asesor: Mtro. German Fabila Garcia
Asesor: Mtro. Cuauhtemoc Aguilera Galicia

San Pedro Tlaquepaque, Jalisco. Septiembre de 2018.

MAESTRO EN INGENIERÍA (2018) ITESO

Maestría en Diseño Electrónico Tlaquepaque, Jal., México

ÁREA DE CONCENTRACIÓN: **“Diseño de Sistemas Digitales”**

AUTOR: Roxana Suarez Lara

Ingeniera en Electronica en Computacion (Centro de Enseñanza Tecnica Industrial, Guadalajara, México)

REVISORES: Dr. Mariano Aguirre Hernandez

Mto. German Fabila Garcia

Mto. Cuauhtemoc Aguilera Galicia

NÚMERO DE PÁGINAS: 181

Contenido

Glosario

ASM..... Algorithmic State Machine

BITS..... Built-In Self-Test

CAD..... Computer Aided Design

CDB..... Command Data Bus

E/S..... Entrada/Salida

FPGA..... Filed Programmable Gate Array

FSM..... Finite State Machine

GPIO..... General Purpose Input Output

IR..... Intruction Register

MAS..... Micro Architecture Spec

MIPS..... Microprocessor without Interlocked Pipeline Stages

PC..... Program Counter

Pipeline..... Consiste en ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, donde la entrada de cada una es la salida de la anterior.

ROB..... Re Order Buffer

SOC..... System On Chip

TAP..... Test Access Port

UART..... Universal Asynchronous Receiver-Transmitter

ATPG..... Automatic Test Pattern Generación

HDL……Hardaware Desihn Lenguage

1. Introducción

Las competencias adquiridas en la Maestría en Diseño Electrónico del ITESO, son herramientas de alto valor agregado para los Ingenieros que trabajan o desean trabajar en la industria electrónica de alta tecnología en nuestro país. El avance acelerado de la Tecnología en este siglo y su complejidad, exige una capacitación constante de Ingenieros preparados con las bases necesarias para el desarrollo de esta tarea, la cual demanda un panorama de conocimientos que abarcan desde la fase del diseño pre-silicio hasta la fase post-silicio. La Maestría en Diseño Electrónico del ITESO ofrece este conocimiento amplio a los Ingenieros que quieren innovar en el campo del diseño electrónico.

Este documento registra los proyectos desarrollados en tres asignaturas del área de concentración de sistemas digitales, los cuales me proporcionaron conocimientos en dicha área para obtener el grado de Maestra en Diseño Electrónico por la modalidad de "Formación Complementaria y Proyectos de Impacto".

2. Área de concentración elegida

Mi elección por los sistemas digitales está basada en que es el área donde encontré más oportunidad de investigación y diseño. Esta Maestría me permitió comprender a bajo nivel la arquitectura de los procesadores y circuitos integrados que hoy en día rigen la tecnología en sus diferentes presentaciones (procesadores para computadoras, tabletas, celulares, etc.). Esta fue mi motivación personal, apoyada en el ideal de un Ingeniero.

Otra razón es la demanda de las empresas en alta tecnología que actualmente se encuentran en Guadalajara, empresas donde se busca que el Ingeniero mexicano sobresalga, que deje de ser solo un ejecutor de pruebas y comience a desarrollarse como un Ingeniero que proponga y dirija proyectos de alto impacto.

Durante el estudio de esta Maestría encontré la vasta gama de conocimientos que ofrecía, no solo en lenguajes de descripción de "hardware", sino también en base a circuitos básicos y avanzados de sistemas digitales, conceptos y metodologías de pruebas, verificación y validación. Precisamente esta amplia posibilidad de aprendizajes me llevó a elegir el área de Sistemas Digitales para el desarrollo de mi formación técnica, profesional y académica. Con la finalidad

de obtener el conocimiento y la experiencia elegí las siguientes materias del área de concentración:

- Diseño de Sistemas Digitales
- Diseño de Microprocesadores
- Prueba de circuitos integrados

A continuación se describen los proyectos realizados en cada una de estas materias conforme a las competencias requeridas para obtener el grado de Maestro en Diseño Electrónico. Los proyectos son:

- Diseño de un procesador MIPS sin pipeline e integración de un módulo UART
- Diseño de un procesador MIPS superescalar con pipeline de 5 etapas
- Diseño de un sistema de prueba de circuitos integrado

3. Curso diseño de sistemas digitales

Esta materia provee conocimientos de lenguajes de descripción de hardware, específicamente Verilog y el uso de la herramienta CAD llamada Modelsim® para la simulación de los circuitos diseñados. En dicho curso se estudian los circuitos combinacionales, secuenciales y diseño de máquinas de estado (FSM y ASM). Estos conocimientos se aplican en la implementación de un procesador MIPS con microarquitectura uni/multi ciclo.

Proyecto: Microprocesador MIPS e integración de un módulo de UART

Este curso está orientado al diseño metodológico de sistemas digitales, aprendizaje de HDLs y el uso del FPGA como una plataforma de implementación. Todas estas técnicas y el conocimiento de herramientas CAD, se aplicaron en un proyecto final que consistió en la diseño e implementación de un MIPS en una FPGA, de un módulo de UART y su conexión al MIPS para comunicarse desde el puerto serial de una computadora. Este proyecto fue desarrollado por David Dávalos y Roxana Suárez. Mi aportación fue el desarrollo del módulo de UART y su conexión al MIPS que implicaba agregar dos GPIO al MIPS para los puertos TX/RX. Esto fueron las partes en las que me enfoque pero tambien desarrolle el mismo MIPS como entrega previa al proyecto final.

3.1 Resumen del proyecto realizado

3.1.1 Microprocesador MIPS e integración de un módulo de UART

Para este proyecto se utilizó la arquitectura básica MIPS que consiste en 5 etapas básicas:

1. FETCH
2. DECODE
3. READ
4. EXECUTE
5. WRITE BACK

Las instrucciones soportadas son: tipo registro (and, or, add, sub, slt), referencia a memoria (load word, store), salto incondicional y condicional.

3.1.1.1 Diseño de la ruta de datos

La ruta de datos general consiste en las siguientes etapas:

- Leer una instrucción de la memoria.
- La instrucción se decodifica para determinar qué acción es necesaria y el flujo que sigue si se dirige hacia la ALU, la unidad SHIFT o a la unidad de saltos.
- La ejecución de una instrucción puede requerir leer datos de la memoria.
- La ejecución de una instrucción puede requerir llevar a cabo alguna operación aritmética o lógica con los datos (ALU).
- Escribir datos en la memoria o en un puerto de e/s (en este caso el UART).

3.1.1.2 Diseño de la ruta de control

La ruta de control es una máquina conformada por 9 estados y sus respectivas señales de control. (Ver detalles de los estados y señales en el Apéndice A). La ruta inicia en el registro PC que contiene la dirección de la siguiente instrucción a ejecutar. La señal de control debe indicar en donde va iniciar el PC de acuerdo con la instrucción anterior, puede ser el caso de un salto o una bifurcación, entonces el PC cambia a lo indicado en dicha instrucción o si va seguir con la

secuencia del programa en ejecución. Inicia la parte de ejecución donde el control es el encargado de mandar los operandos a la ALU. El control debe especificar en cada estado los valores de las señales para la transferencia entre registros. El cambio de estado ocurre en cada flanco de subida del reloj. La secuencia de transferencias se controla mediante el recorrido a través de los estados. El diagrama con la solución completa puede encontrarse en el Apéndice A.

3.1.1.3 Aportaciones y conclusiones durante el proyecto

El modelo Verilog del MIPS y las simulaciones elaboradas del modelo pueden encontrarse en el apéndice A. El proyecto me permitió aplicar el aprendizaje sobre diseño de sistemas digitales, conocer los lenguajes de descripción de hardware y principalmente las metodologías para el desarrollo de estas tareas de la mejor manera en un entorno laboral. Estas tareas consistieron en la comprensión del documento de especificaciones para la elaboración de un diagrama de flujo y de control a nivel de lo general a lo particular. Comprender las máquinas de estados presentadas por el diseñador para de este punto partir con el desarrollo del RTL. El proceso fue muy parecido al llevado en la industria donde el Maestro actuaba como el diseñador y nosotros los desarrolladores.

Este curso utiliza el estudio de arquitecturas básicas de computadoras como vehículo para la clara comprensión de las metodologías de diseño digital. Dichas bases las aplico actualmente en mi trabajo. Conocer la lógica de los sistemas modernos me ha ayudado para la depuración de errores que se reflejan en el usuario final; como puede ser corrupción de video, bloqueo del sistema, errores de comunicación entre los periféricos, etcétera. Muchas veces estas fallas resultan ser un error en la codificación del RTL.

Este proyecto en definitiva permite obtener conocimientos claves para la depuración de errores en HW como desarrollo de RTL.

En el Apéndice A se encuentran todos los diagramas de flujo, el RTL y simulación de los resultados.

4. Curso diseño de microprocesadores

Proyecto: Microprocesador MIPS de arquitectura Superescalar

Se implementó un microprocesador capaz de ejecutar un conjunto de instrucciones MIPS en una arquitectura superscalar con ejecución fuera de orden. La implementación de esta arquitectura permitió estudiar diferentes técnicas de las arquitecturas modernas como son: pipeline superescalar, predicción de saltos, renombramiento de registros, etc. El desarrollo de este MIPS se dividió en dos áreas funcionales para su desarrollo, conocidas como el "front-end" que corresponde al flujo de instrucciones y el "back-end" que corresponde al flujo de datos. El flujo de instrucciones corresponde a las etapas conocidas como: "fetch", "Decode" y "Dispatch". Por lo que yo desarrolle las etapas conocidas como: colas de ejecución, "issue" y "write-back". Revisar el Apéndice B para mayor especificación técnica.

4.1. Resumen del proyecto realizado

4.1.1 Desarrollo

Se analizaron diferentes técnicas utilizadas actualmente para el mejor desempeño de los microprocesadores. Durante el curso se analizaron dos enfoques principales de las arquitecturas modernas:

- Estrategias para la ejecución simultanea de instrucciones.
- Mejoramiento del tiempo de ejecución.

Con base en estas técnicas y las especificaciones dadas por el profesor se llevó a cabo la codificación del modelo del procesador a nivel RTL sintetizable. Una de las principales características del procesador eran el pipeline y ejecución fuera de orden. Este proyecto se elaboró en equipó con Rafael del Rey. Mi aportación fue el desarrollo de "back end" que consisitio en las colas de ejecución, determinar tipo de ejecución por cada cola. Diferenciar los tipos de instrucción: aritméticas, lógicas, de decisión, saltos o de acceso a memoria. Esto me permitió decidir el flujo que deberían de seguir cada uno de los elementos en las instrucciones.

En esta materia se desarrollaron otras técnicas del diseño como la investigación enfocada en el estudio de arquitecturas avanzadas, se estudiaron comparaciones con el MIPS básico. En estas comparaciones se tomaba un conjunto de instrucciones y de acuerdo a sus características se verificaba que estado tomaban en cada uno de los MIPS. Por ejemplo en el MIPS superescalar

con pipeline de 5 etapas, cuando una instrucción termina la etapa de “fetch” inicia esta misma etapa la siguiente instrucción y así sucesivamente. Mientras en el MIPS convencional siempre iniciaba la siguiente instrucción hasta que terminaba la actual. Ejercicios como este se hicieron constantemente en clase, y en lo personal, antes de correr una programa de prueba solía hacer este análisis. Para saber qué resultados podría esperar de la simulación.

4.1.2 Resultados

El MIPS generado se verificó con un algoritmo conocido como “ordenamiento burbuja”. La ejecución fuera de orden y todas las técnicas vistas en clase y aplicadas al proyecto se pudieron ver en simulación con la herramienta Modelsim. Uno de los principales resultados fue como efectivamente se ejecutan varias instrucciones en un solo ciclo. Aquí es donde llevamos de la teoría a la práctica las técnicas antes mencionadas.

Se lograron tener comparaciones significativas, entra arquitecturas básicas y avanzadas. Principalmente en desempeño y velocidad de ejecución. Como se puede observar en el Apéndice B. La implementación de RTL fue más compleja que la utilizada para el MIPS sin pipeline (Apéndice A). Pero el desempeño del MIPS con pipeline fue mucho más eficiente. También la depuración de errores fue mucho más compleja ya que al implementar el RTL se dividió en varias partes y en la integración se encontraron varios errores. Los cuales fueron resueltos hasta lograr una integración exitosa de todos los módulos.

4.1.3 Aportaciones

Este curso fue el que mayor impacto tuvo en mi formación tanto académica y profesional, ya que aportó cuestiones básicas y esenciales para mi trabajo actual. El proyecto invita a no simplemente recibir instrucciones de un proyecto e implementarlo si no también a investigar e ir más allá de las especificaciones entregadas. Este proyecto me ayudo a conocer diferentes metodologías de la arquitectura de computadoras. Todas estas utilizadas actualmente en la industria. Esto nos muestra que no todo está escrito o desarrollado, que una vez que se tiene las bases sobre arquitectura de computadoras, el Ingeniero puede aportar nuevas tecnologías para la optimización de las mismas.

Oportunidad de innovación es la que entrega este proyecto a los Ingenieros que toman este curso.

5. Curso prueba de circuitos integrados

En él se estudian métodos tanto teóricos como prácticos del proceso de diseño de pruebas de manufactura para la detección del mayor número de fallas posibles en los circuitos integrados. Estos temas consisten en determinar las causas que originan los defectos de fabricación.

Conocer y analizar condiciones físicas, estudiar cómo modelar defectos de manufactura y conocer los recursos y las técnicas aplicadas en el desarrollo de pruebas de manufactura. Estas acciones permiten en la industria ver los requerimientos de cobertura de fallas para asegurar la calidad del producto.

Proyecto: Desarrollo de un sistema para pruebas de circuitos integrados basado en un sumador, el sistema incluye simulador de fallas, módulo de TAP y módulo de BIST.[1]

5.1 Resumen del proyecto realizado

En el desarrollo del proyecto se buscó simular el proceso real de prueba de circuitos integrados utilizados en la industria. Las etapas principales fueron: Construcción de un circuito integrado, en él se basó en un sumador completo con retroalimentación, desarrollo de un simulador de fallas, implementación de un módulo TAP y también se incluyó en el proyecto un módulo de bits. Se agregó una retroalimentación entre la salida y la entrada del sumador para tener como resultado mientras estuviera corriendo la serie de Fibonacci. Se eligió un circuito sencillo ya que la finalidad de este curso no era el diseño de circuitos, sino comprender todo el proceso que conllevan las prueba de los circuitos integrados, esto permitió el aprendizaje de varios procesos utilizados en la manufactura de circuitos integrados, ya que no se trataba solamente de teoría de pruebas. Este curso adicionalmente ayudó a comprender la diferencia entre verificación y prueba. El aprendizaje de esta materia es el desarrollo de las pruebas de manufactura aún antes de que el circuito esté fabricado. Para poder incluir en el circuito tanto su funcionalidad, como todo lo necesario para que pueda ser probado.

[1] Programa de Estudios 2010 de la Maestría en Diseño Electrónico del ITESO → Prueba de Circuitos Integrados

5.1.1. Sumador con retroalimentación

El circuito base lo conforman un circuito sumador de 16 – bits donde sus principales elementos fueron dos circuitos multiplexores, dos flip-flops y una compuerta not, como se muestra en la FIGURA 1, a su vez algunos de estos circuitos están formados por diferente lógica combinacional generada a partir de compuertas básicas (and, or, not).

Se le llamo circuito base porque durante el curso se utilizaron las técnicas de pruebas de circuitos integrados estudiadas. El enfoque de esta materia no era el desarrollo del circuito si no el implementar el HW para su prueba dentro del mismo circuito integrado.

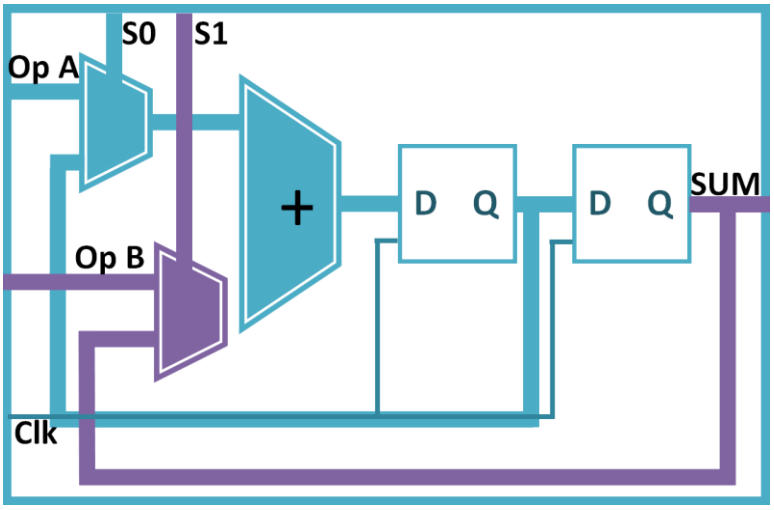


Figura 1. Circuito Base

5.1.2. Simulación de fallas

Como ya se mencionó al inicio de esta sección uno de los objetivos era la construcción de un simulador de fallas. Esta fue la primera técnica de prueba estudiada durante el curso. El propósito de este simulador de fallas, es encontrar defectos de manufactura más que verificación funcional. Existen diferentes defectos de manufactura. En este proyecto se enfocó a cortos circuitos, óxido en las compuertas e interconexiones abiertas. La detección de estos defectos se basa en los niveles lógicos de las compuertas.

A modo de poder simular estos defectos se utiliza la técnica conocida como atrapado a 0 o atrapado a 1 (S@0/S@1). El hecho de simular fallas es para seleccionar las pruebas con mejor cobertura. En este trabajo se buscó simular con la herramienta Modelsim, a modo de aprendizaje, el modelo utilizado en la industria como ATPG. Este modelo genera entradas random y mide el

número de fallas detectadas de las simuladas y va repitiendo ciclos hasta encontrar la mayoría, y de esta manera dejar solo un porcentaje menor para la aplicación de pruebas dirigidas. Un ejemplo sería el de un inversor, el cual debe entregar el valor contrario al que le entra. Una prueba para éste serían las únicas 2 combinaciones posibles que son 0 y 1, si aun cuando mi entrada es 1, me sigue dando como salida un 1 (se implementó un S@1 a la salida para simular la falla). Entonces existe un defecto a la salida.

Las pruebas dirigidas entran cuando el patrón aleatorio no fue capaz de detectar todas las fallas aplicadas al circuito. Una prueba dirigida puede ser en el caso de que no haya sido detectado un S@0 a la salida de una compuerta and es porque el generador de pruebas nunca género que las entradas de la compuerta fueran ambas 1. En este caso se debe hacer una prueba dirigida para evitar que la falla no llegue a ser detectada. Por mencionar un ejemplo sencillo. A continuación se encuentra la lógica y seguimiento que se llevó a cabo para la realizar el simulador de fallas.

5.1.2.1 Cono de control y observación

Para la parte de Cono de control y observación se eligió la señal:

testbench.Adder.Full_Adder.Add0.Cin

La cual es el primer Cin del sumador completo de 1 bit, es decir la lógica que hace la suma de A[0] y B[0].

out_xor0	Cin	Suma[0]
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 1

Como se observa en la tabla 1, se analiza la tabla de verdad de esta etapa del circuito para poder obtener el patrón que detectaría cuando existiera un S@0 o S@1. Como patrón se mantuvieron siempre apagados los selectores del circuito con los que se selecciona si se desea tomar un operando externo o se toma el resultado anterior como operando. Después se observa que la señal afecte directamente a las salidas. Por un lado el resultado se vería afectado por la salida de la xor que entrega el primer bit del resultado completo (Suma[0]) y también se vería afectado el

acarreo de salida ya que el Cout[0] se va reflejando bit por bit hasta llegar a Cout[15] dando esto como resultado el Cout final. También para facilitar el análisis se observa que cuando el resultado de la suma es afectado se refleja la falla.

Por lo que se analizaron los estímulos de la xor que dan como resultado Suma[0]. Existen 4 vectores posibles en una xor

Stuck @ 1		
out_xor0	Cin	Suma[0]
0	0	0
1	0	1

Tabla 2

Stuck @ 0		
out_xor0	Cin	Suma[0]
0	1	1
1	1	0

Tabla 3

En ambos casos se observa que una de las entradas es la salida de una xor y para ello necesitamos buscar un patrón de entradas para esa xor que entregue la salida que se desea probar con la falla. Las entradas a estas xor son los bits A[0] y B[0] por lo que se sugirió el siguiente patrón para probar las fallas:

Stuck @ 1:
0^0 with A[0]=0, B[0]=0, Cin=0
0^0 with A[0]=1, B[0]=1, Cin=0
1^0 with A[0]=0, B[0]=1, Cin=0
1^0 with A[0]=1, B[0]=0, Cin=0

Stuck @ 0:
0^1 with A[0]=0, B[0]=0, Cin=1
0^1 with A[0]=1, B[0]=1, Cin=1
1^1 with A[0]=0, B[0]=1, Cin=1
1^1 with A[0]=1, B[0]=0, Cin=1

5.1.2.2 Cono de control

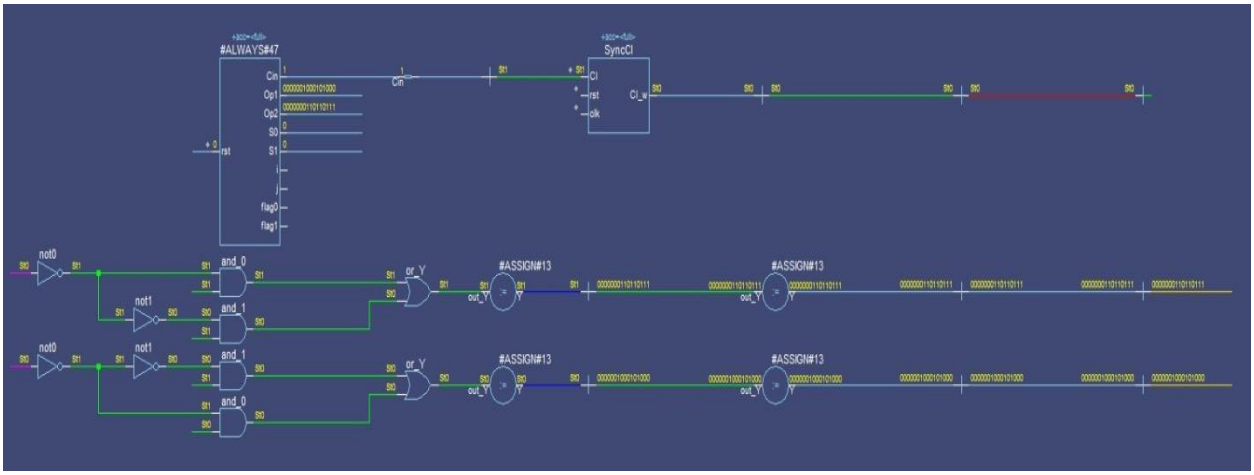


Figura 2. Netlist generado

La línea marcada de rojo es la señal a la que se le está inyectando la falla, donde el control depende básicamente de las entradas del circuito.

5.1.2.3 Cono de observación

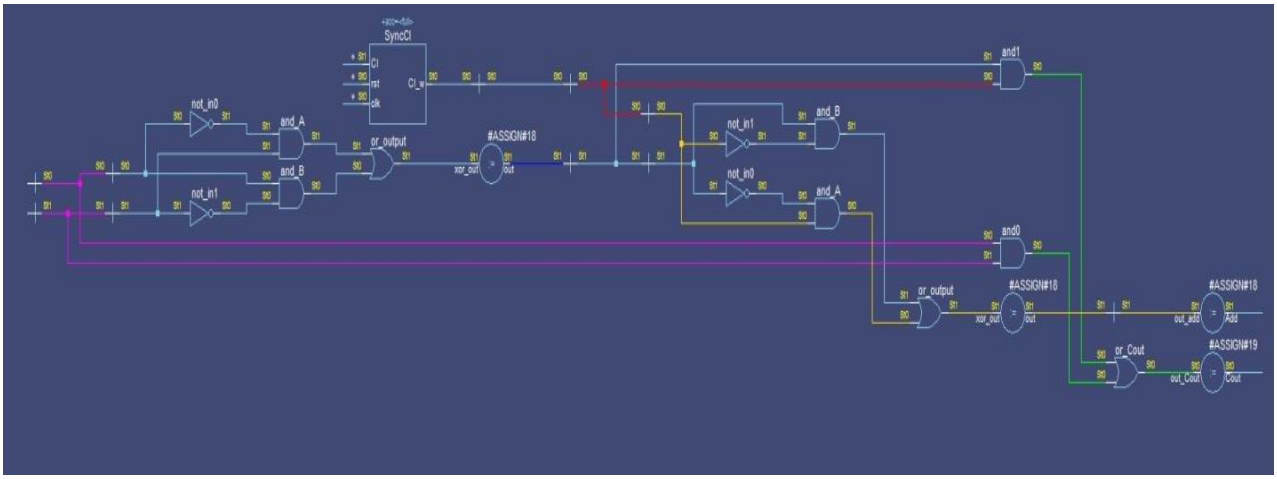


Figura 3. Propagación de la falla

Rojo: Señal a la que se le inyecta la falla(testbench.Adder.Full_Adder.Add0.Cin)

Rosa: Entradas A[0] y B[0], base para lograr el patrón

Verde: Propagación de la falla hacia Suma[0] y Cout[0] que es el Cin al sumador del siguiente.

A partir de este análisis se genera el sistema de prueba de circuitos integrados. Simulaciones y detalles de cada una de las etapas del proyecto se pueden encontrar en el apéndice C.

5.1.3. Aportaciones y resultados

Lo interesante es cómo a partir de un circuito base de arquitectura sencilla se puede comprender todo lo que implica la infraestructura de prueba de circuitos integrados. Mi aportación consistió en el diseño completo del simulador de fallas con base en la teoría vista en clase, así como cada una de las etapas del sistema de prueba de circuitos integrados. Adicional a lo que se pidió desarrollé diferentes scripts con la finalidad de tener un sistema que analizara la cobertura de fallas. Este sistema determina si debe correr nuevamente un patrón aleatorio o se ha llegado a más de un 90% de cobertura y es necesaria el desarrollo de pruebas dirigidas. El sistema trabaja en base al análisis de archivos con resultados generados por el simulador de fallas. La elaboración de scripts para el análisis de resultados en las simulaciones es una práctica común en la industria, ya que siempre es necesario tener la información correcta en menor tiempo. El uso de lenguajes de scripting permite hacer esto posible. En el Apéndice C se pueden observar todos los diagramas de flujo, el RTL y resultados del sistema completo de prueba de circuitos integrados.

6. Referencias

Apéndice A "Diseño de Sistemas Digitales"

- Diseño Digital, 3ra Edición M. Morris Mano
- Arquitectura de Sistemas de Computación, 3ra Edición M. Morris Mano
- Presentación de "Utah University" http://www.eng.utah.edu/~cs6710/slides/mipsx2.pdf

Apéndice B "Diseño de Microprocesadores"

- Arquitectura de Computadoras 5a Edición, Petrice Hall, William Stallings
- "Advance Microprocessors and Interfacing" 1ª Edition, Bradi Ram
- Paralelismo en monoprocesadores y Procesadores Superescalares http://www.exa.unicen.edu.ar/catedras/arqui2/arqui2/filminas/Paralelismo%20en%20monoprocesadores%20-%20Superescalares.pdf
- MIPS Pipeline, Hakim Weatherspoon, Computer Science Cornell University http://www.cs.cornell.edu/courses/cs3410/2012sp/lecture/09-pipelined-cpu-i-g.pdf

Apéndice C "Prueba de Circuitos Integrados"

- "Physical Design Essentials" específicamente el capítulo de "Testing". Online book
- Integrated Circuit Test Engineering, 1a Edicion, Ian A. Grout
- Técnicas para Emulación de Fallos Transitorios, López-Ongil C, García-Valderas M, Portela-García M, Entrena-Arrontes L, Grupo de Microelectrónica. Departamento de Tecnología Electrónica. Escuela Politécnica Superior, Universidad Carlos III de Madrid http://jcraconf.org/JCRA2009/docs/proceedings/2004/pdf/b3_2.pdf

Apéndice A

RTL de microprocesador MIPS con módulo de comunicación tipo UART

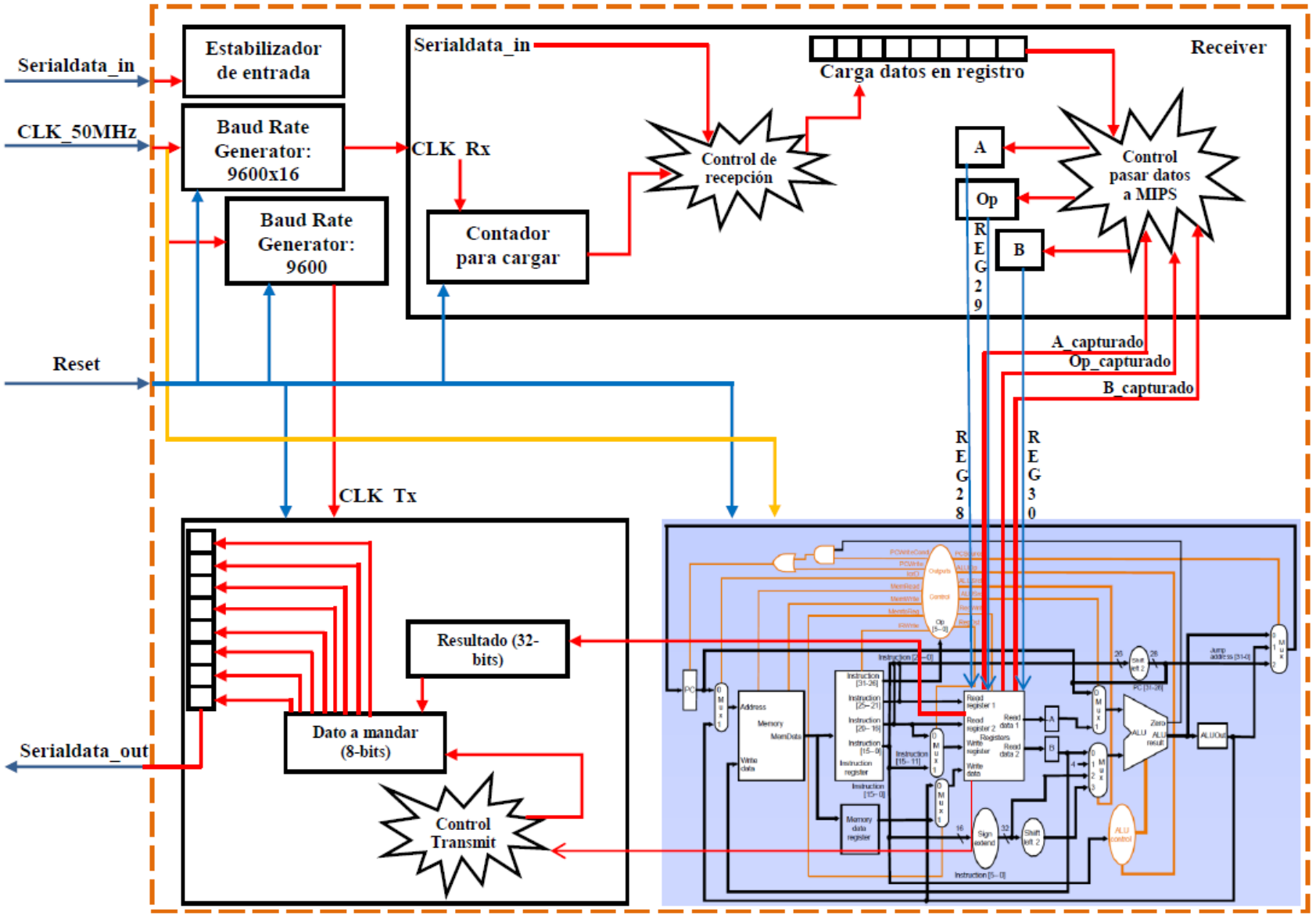
En este apéndice se encuentra el Netlist de Verilog de todo el procesador MIPS y el netlist del módulo de UART.

Contenido

1. Diagrama general a bloques de todo el sistema
2. Máquina de estados del módulo de UART
3. Netlist del módulo de UART
4. Diagrama general de la arquitectura del procesador y las señales agregadas para incluir el módulo de UART
5. Diagrama general de la máquina de estados para el control del MIPS
6. Simulación de resultados

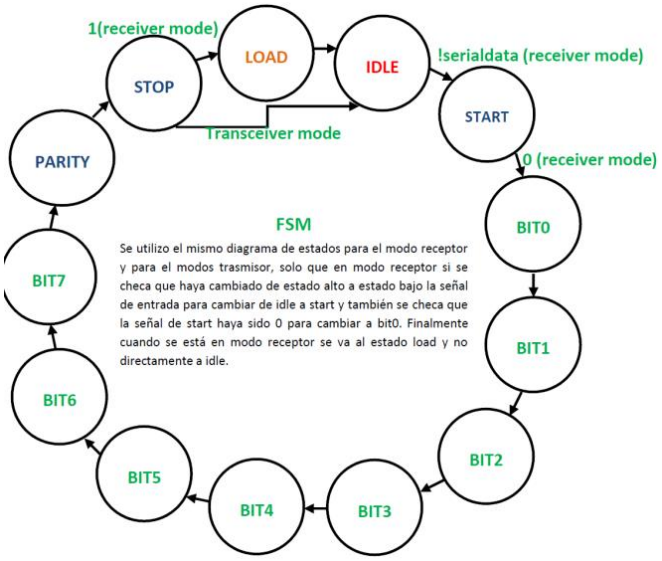
Modulos de Verilog:

Modulo de Verilog	Descripcion
MIPS_Final	Netlist donde se realiza todas las conexiones del sistema completo. Hace la unión entre los módulos MIPS_Top y UART
MIPS_Top	Netlist del MIPS completo. Interconexión de todos los sub-módulos y etapas que conforman el MIPS
MemoryData	Modulo que contiene los registros de la memoria de datos
Register	Modulo que contiene los registros del MIPS
pb_pulse	Módulo para generar un retraso en el uso de “push buttons”
SignExtend	Módulo para el manejo de operaciones con signo extendido
ALUOut	Flipflop para la salida de la ALU
ShiftLeft2	Modulo de operaciones de shift
PC	Program Counter
ALUControl	Módulo para determinar la operación que ejecutará la ALU
InstructionReg	Modulo de los registros de instrucciones
ControlUnit	Ruta de control
MemoryDataReg	Registros de la memoria de datos
RegAux	Flipflop auxiliar
ALU	Modulo encargado de la ejecucion de las operaciones aritmeticas y logicas


Serialdata_in
CLK_50MHz
Reset
Serialdata_out
Estabilizador de entrada
Baud Rate Generator: 9600x16
Baud Rate Generator: 9600
Serialdata_in
Receiver
Carga datos en registro
CLK_Rx
Contador para cargar
Control de recepción
A
Op
B
Control pasar datos a MIPS
REG29
A_capturado
Op_capturado
B_capturado
REG28
REG30
CLK_Tx
Resultado (32-bits)
Dato a mandar (8-bits)
Control Transmit

MIPS_Final.v

```
`timescale 1ns/100ps
module MIPS_Final(clk, rst, serialdata_in, serialdata_out);
input  clk;
input  rst;
input  serialdata_in;
output serialdata_out;

wire wserialdata_in;
wire wA_done;
wire wOpCode_done;
wire wB_done;
wire wsignalflag;
wire [31:0] wResultado;
wire wTransmit;
wire [31:0] wA_UART;
wire [31:0] wOpCode_UART;
wire [31:0] wB_UART;
assign wA_UART[31:8] = 0;
assign wOpCode_UART[31:8] = 0;
assign wB_UART[31:8] = 0;

MIPS_Top MIPS1(.clk(clk), //
               .rst(rst), //
               .inA_UART(wA_UART),
               .inOpCode_UART(wOpCode_UART),
               .inB_UART(wB_UART),
               .outTransmit(wTransmit),
               .outA_done(wA_done),
               .outOpCode_done(wOpCode_done),
               .outB_done(wB_done),
               .outResultado(wResultado),
               .signalflag(wsignalflag));
pb_pulse data(.clk(clk),
                                  .pb_in(serialdata_in),
                                  .pulse_out(wserialdata_in));
UART UART1(.clk_50MHz(clk),
                        .rst(rst),
               .serialdata_in(wserialdata_in), //
               .A_done(wA_done),
               .OpCode_done(wOpCode_done),
               .B_done(wB_done),
               .resultado(wResultado),
               .transmit(wTransmit),
               .serialdata_out(serialdata_out), //
               .A_UART(wA_UART[7:0]),
               .OpCode_UART(wOpCode_UART[7:0]),
               .B_UART(wB_UART[7:0]),
               .signalflag(wsignalflag));
endmodule
```



Maquina de estados del UART

El desarrollo del módulo UART se llevó a cabo desde el diseño de su máquina de estados. Se implementó un divisor de reloj para comunicar el mips y el UART. El mips utiliza un reloj de 50 MHz y el UART trabaja con un reloj a 9600 baudios. También se utilizaron 2 registros de E/S para comunicar el MIPS con el UART.

Generación del Baud Rate a 9600:

- 50MHz/9600 = 5208.3333 (Ciclo completo del reloj del UART)
- 5208.3333/2 = 2604.1666 (Cada cuando debe ocurrir un flanco de subida)

```
always@(posedge clk)
begin
        if(count_baudrate == 2604)begin
        BaudRate_9600 <=
~BaudRate_9600;
        count_baudrate <= 0;
end
        else count_baudrate <=
count_baudrate + 1;
end
```

Implementación de divisor de reloj para uart a 9600 baudios

UART.v

```
module UART(clk_50MHz, rst, serialdata_in, A_done, OpCode_done, B_done, resultado, transmit, serialdata_out,
A_UART, OpCode_UART, B_UART, signalflag);

input clk_50MHz, rst, serialdata_in, transmit, A_done, OpCode_done, B_done;
input [31:0]resultado;
output reg serialdata_out;
output reg signalflag;
output reg [7:0] A_UART;
output reg [7:0] OpCode_UART;
output reg [7:0] B_UART;
```

```
reg [3:0] count_load; //señal que se activa a la mitad del bit
reg [3:0] cur_state; //estado actual para el Receiver
reg [3:0] nxt_state; //estado siguiente para el Receiver
reg [3:0] cur_state_Tx; //estado actual para el Transceiver
reg [3:0] nxt_state_Tx; //estado siguiente para el Transceiver
reg [7:0] count_baudrate; //contador para generar el BaudRate de 9600x16
reg [11:0] count_baudrate_Tx; //contador para generar el BaudRate de 9600
reg [7:0] serialdata_regin; //registro que va guardando los bits de datos
reg [7:0] serialdata_regout; //registro donde se guarda el dato recibido modificado para enviarlo
reg [7:0] serialdata_regin_traduction;
reg [7:0]result;
reg [7:0]resultado1;
reg [7:0]resultado2;
reg [7:0]saveresult;
reg [1:0] datatype;
reg [2:0] DataToTransmit;
reg BaudRate_9600; //señal de reloj (Baud Rate tick) de Transceiver
reg BaudRate_9600x16; //señal de reloj (Baud Rate tick) de Receiver
reg load_data;
reg stop_bit;
reg start_bit;
reg bit_parity;
reg transceiver;
reg bit_parity_Rx; //guarda la paridad recibida
reg bit_parity_out; //registro que guarda la paridad a enviar en modo Transceiver
reg parity_comparator; //se activa una vez que se recibe la paridad para compararla con la calculada

parameter[3:0] idle = 4'b0000, //esperando transferencia
               start = 4'b0001, //start bit
               bit0 = 4'b0010, /*Inicia recepcion de datos*/
               bit1 = 4'b0011,
               bit2 = 4'b0100,
               bit3 = 4'b0101,
               bit4 = 4'b0110,
               bit5 = 4'b0111,
               bit6 = 4'b1000,
               bit7 = 4'b1001, /*Terminan recepcion de datos*/
               parity = 4'b1010, //bit de paridad
               stop = 4'b1011; //stop bit

parameter[1:0] A = 2'b00,
               OpCode = 2'b01,
               B = 2'b10;

parameter[2:0] equal = 3'b000,
               neg = 3'b001,
               Op1 = 3'b010,
               Op2 = 3'b011,
               Enter = 3'b100;

always@(posedge clk_50MHz or posedge rst)
begin
  if(rst)
  begin
    count_baudrate <= 0;
    BaudRate_9600x16 <= 0;
  end
  else
  begin
```

```
    if(count_baudrate == 163)/*50MHz/9600x16 = 325.52 <-- period completo.
                                Para obtener la señal de reloj correcta
                                                    325.52/2 = 162.76 --> 163*/
    begin
      BaudRate_9600x16 <=  ~BaudRate_9600x16;/*Señal de reloj para Receiver
                                                                  16 veces mas rapido que 9600
                                                                  para cargar a la mitad del
                                                                  dato recibido*/
      count_baudrate <= 0;
    end
    else count_baudrate <= count_baudrate + 1;
  end
end

always@(posedge clk_50MHz or posedge rst)
begin
  if(rst)
  begin
    count_baudrate_Tx <= 0;
    BaudRate_9600 <= 0;
  end
  else
  begin
    if(count_baudrate_Tx == 2604)/*50MHz/9600 = 5208.33 <-- Periodo completo.
                            Para obtener la señal de reloj correcta
                                                    5208.33/2 = 2604.16 --> 2604*/
    begin
      BaudRate_9600 <=  ~BaudRate_9600;//Señal de reloj para Transceiver
      count_baudrate_Tx <= 0;
    end
    else count_baudrate_Tx <= count_baudrate_Tx + 1;
  end
end

always @(posedge BaudRate_9600x16 or posedge rst)
begin
  if(rst)
  begin
    stop_bit <= 0;
    bit_parity <= 0;
    bit_parity_Rx <= 0;
    count_load <= 0;
    cur_state <= idle;
    serialdata_regin_traduction <= 0;
  end
  else
  begin
    cur_state <= nxt_state;
    count_load <= count_load + 1; /*obtiene las 16 cuentas que se pueden
                                                    realizar para controlar la carga de datos
                                                    (a mitad del bit trasmitido) y el cambio de estado (cuando se
                                                    termina la trasmision del bit)*/

    /*Acciones a tomar cuando esta recibiendo*/
    case(cur_state)
      idle:
```

```
      begin
        serialdata_regin <= 0; //registro para guardar los datos de entrada
        parity_comparator <= 0; //Se activa para comparar el bit de paridad
      end

      start: if(count_load == 15) start_bit <= serialdata_in;

      /*Guarda los bits de datos recibidos*/
      bit0: if(count_load == 15) serialdata_regin[0] <= serialdata_in;
      bit1: if(count_load == 15) serialdata_regin[1] <= serialdata_in;
      bit2: if(count_load == 15) serialdata_regin[2] <= serialdata_in;
      bit3: if(count_load == 15) serialdata_regin[3] <= serialdata_in;
      bit4: if(count_load == 15) serialdata_regin[4] <= serialdata_in;
      bit5: if(count_load == 15) serialdata_regin[5] <= serialdata_in;
      bit6: if(count_load == 15) serialdata_regin[6] <= serialdata_in;
      bit7: if(count_load == 15) serialdata_regin[7] <= serialdata_in;
      parity:
      begin
        if(count_load == 15)
        begin
          bit_parity_Rx <= serialdata_in; //guarda el bit de paridad
          bit_parity <= (serialdata_regin[0]^serialdata_regin[1]^serialdata_regin[2]^serialdata_regin[3]
      ^serialdata_regin[4]^serialdata_regin[5]^serialdata_regin[6]^serialdata_regin[7]); //calcula la paridad con
los bits de datos recibidos
          parity_comparator <= 1;
        end
        if(parity_comparator)
        begin
          if(bit_parity_Rx == bit_parity)
            begin
              case(serialdata_regin)
                43: serialdata_regin_traduction <= 43;
                45: serialdata_regin_traduction <= 45;
                48: serialdata_regin_traduction <= 0;
                49: serialdata_regin_traduction <= 1;
                50: serialdata_regin_traduction <= 2;
                51: serialdata_regin_traduction <= 3;
                52: serialdata_regin_traduction <= 4;
                53: serialdata_regin_traduction <= 5;
                54: serialdata_regin_traduction <= 6;
                55: serialdata_regin_traduction <= 7;
                56: serialdata_regin_traduction <= 8;
                57: serialdata_regin_traduction <= 9;
                default: serialdata_regin_traduction <= 255;
              endcase
            end
          end
        end
        stop: if(count_load == 15) stop_bit <= serialdata_in;
      endcase
  end
end

always @(posedge BaudRate_9600 or posedge rst)
begin
  if(rst)
  begin
```

```
  signalflag <= 0;
  serialdata_out <= 1;
  datatype <= A;
  load_data <= 0;
  cur_state_Tx <= idle;
  resultado1 <= 8'b00110001;
  saveresult <= 0;
  result <= 0;
  transceiver <= 0;
  serialdata_regout <= 0;
  resultado2 <= 0;
  A_UART <= 0;
  OpCode_UART <= 0;
  B_UART <= 0;
  DataToTransmit <= equal;
end
else
begin
  if(serialdata_regout == 13) signalflag <= 1; else signalflag <= 0;
  if(cur_state == stop) load_data <= 1;
    if(load_data)
    begin
      case(datatype)
        A:
        begin
          A_UART <= serialdata_regin_traduction;
          datatype <= OpCode;
          load_data <= 0;
        end

        OpCode:
        begin
          OpCode_UART <= serialdata_regin_traduction;
          datatype <= B;
          load_data <= 0;
        end
        B:
        begin
          B_UART <= serialdata_regin_traduction;
          datatype <= A;
          DataToTransmit <= equal;
          load_data <= 0;
        end

        default:
        begin
          datatype <= A;
          DataToTransmit <= equal;
          load_data <= 0;
        end
      endcase
    end

  if(B_UART != 0)
  begin
    transceiver <= 1;
```

```
    end

    if(A_done) A_UART <= 0;
    if(OpCode_done) OpCode_UART <= 0;
    if(B_done) B_UART <= 0;

    if(transmit)
    begin
      transceiver <= 1;
      if(resultado[7]) DataToTransmit <= neg;
      else DataToTransmit <= Op1;
    end

    if(resultado[7]) result <= ~(resultado - 1);

    if(resultado > 9)
    begin
      case (resultado)
        10: resultado2 <= 48;
        11: resultado2 <= 49;
        12: resultado2 <= 50;
        13: resultado2 <= 51;
        14: resultado2 <= 52;
        15: resultado2 <= 53;
        16: resultado2 <= 54;
        17: resultado2 <= 55;
        18: resultado2 <= 56;
        default: resultado2 <= 255;
      endcase
    end

    else resultado2 <= 0;
    cur_state_Tx <= nxt_state_Tx;
    bit_parity_out <= (serialdata_regout[0]^serialdata_regout[1]^serialdata_regout[2]^serialdata_regout[3]
                      ^serialdata_regout[4]^serialdata_regout[5]^serialdata_regout[6]^serialdata_regout[7]); //obtiene
el bit de paridad a enviar
    /*Acciones a tomar cuando actua como Transceiver*/
    case(cur_state_Tx)
      idle:
    begin
      serialdata_out <= 1;
      case (DataToTransmit)
        equal: serialdata_regout <= 8'b00111101;
        neg: serialdata_regout <= 8'b00101101;

        Op1:
        begin
        if(resultado[7])
          begin
          case (result)
            0: begin serialdata_regout <= 48; saveresult <= 48; end
            1: begin serialdata_regout <= 49; saveresult <= 49; end
            2: begin serialdata_regout <= 50; saveresult <= 50; end
            3: begin serialdata_regout <= 51; saveresult <= 51; end
            4: begin serialdata_regout <= 52; saveresult <= 52; end
            5: begin serialdata_regout <= 53; saveresult <= 53; end
            6: begin serialdata_regout <= 54; saveresult <= 54; end
            7: begin serialdata_regout <= 55; saveresult <= 55; end
            8: begin serialdata_regout <= 56; saveresult <= 56; end
```

```
          9: begin serialdata_regout <= 57; saveresult <= 57; end
          default: begin serialdata_regout <= 255; saveresult <= 255; end
        endcase
        end
      else if(resultado > 9) serialdata_regout <= resultado1;
        else
        begin
          case (resultado)
            0: begin serialdata_regout <= 48; saveresult <= 48; end
            1: begin serialdata_regout <= 49; saveresult <= 49; end
            2: begin serialdata_regout <= 50; saveresult <= 50; end
            3: begin serialdata_regout <= 51; saveresult <= 51; end
            4: begin serialdata_regout <= 52; saveresult <= 52; end
            5: begin serialdata_regout <= 53; saveresult <= 53; end
            6: begin serialdata_regout <= 54; saveresult <= 54; end
            7: begin serialdata_regout <= 55; saveresult <= 55; end
            8: begin serialdata_regout <= 56; saveresult <= 56; end
            9: begin serialdata_regout <= 57; saveresult <= 57; end
            default: begin serialdata_regout <= 255; saveresult <= 255; end
          endcase
        end
      end
      Op2: serialdata_regout <= resultado2;
      Enter: serialdata_regout <= 8'b00001101;
      default: serialdata_regout <= 0;
    endcase
  end
    start: begin transceiver <= 0; serialdata_out <= 0; end//start bit
    bit0: serialdata_out <= serialdata_regout[0];/*inicia envio de datos*/
    bit1: serialdata_out <= serialdata_regout[1];
    bit2: serialdata_out <= serialdata_regout[2];
    bit3: serialdata_out <= serialdata_regout[3];
    bit4: serialdata_out <= serialdata_regout[4];
    bit5: serialdata_out <= serialdata_regout[5];
    bit6: serialdata_out <= serialdata_regout[6];
    bit7: serialdata_out <= serialdata_regout[7];/*Termina envio de datos*/
    parity: serialdata_out <= bit_parity_out;

    stop:
    begin
      serialdata_out <= 1; //envia stop bit
      if(resultado[7] && result > 9)
        begin
          transceiver <= 1;
          if (serialdata_regout == 45) DataToTransmit <= Op1;
          if (serialdata_regout == resultado1) DataToTransmit <= Op2;
          if (serialdata_regout == resultado2) DataToTransmit <= Enter;
        end
        else if(resultado[7])
        begin
          transceiver <= 1;
          if (serialdata_regout == 45) DataToTransmit <= Op1;
          if (serialdata_regout == saveresult) DataToTransmit <= Enter;
        end
        else if(resultado > 9)
        begin
          transceiver <= 1;
```

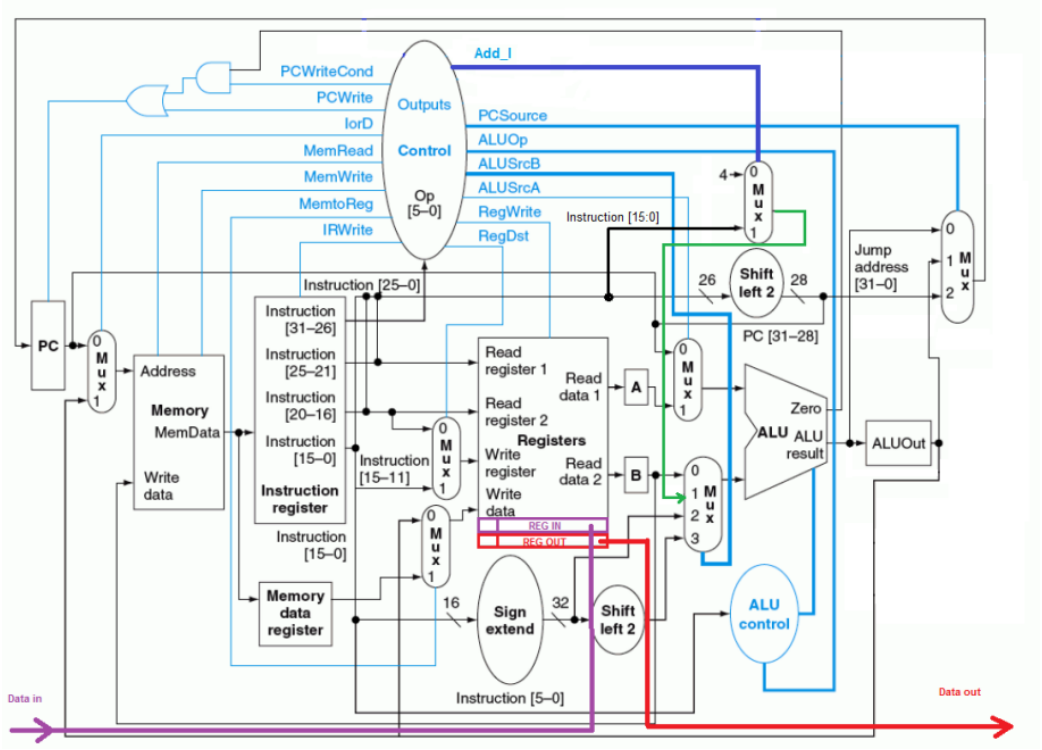
```
            if (serialdata_regout == resultado1) DataToTransmit <= Op2;
            if (serialdata_regout == resultado2) DataToTransmit <= Enter;
          end
          else
          begin
            transceiver <= 1;
            if(serialdata_regout == saveresult) DataToTransmit <= Enter;
          end
          if(DataToTransmit == Enter)
          begin
            transceiver <= 0;
          end
        end
    endcase
  end
end
always@(*)
begin
  nxt_state = cur_state; //Receiver states
  nxt_state_Tx = cur_state_Tx; //Transceivers states
  /*Logica Combinacional del siguiente estado en modo Receiver*/
  case(cur_state)
    idle: if(!serialdata_in) nxt_state = start; //si detecta que la entrada cambio de alto a bajo
    start: if(count_load == 0 && start_bit == 0) nxt_state = bit0;
    bit0: if(count_load == 0) nxt_state = bit1;
    bit1: if(count_load == 0) nxt_state = bit2;
    bit2: if(count_load == 0) nxt_state = bit3;
    bit3: if(count_load == 0) nxt_state = bit4;
    bit4: if(count_load == 0) nxt_state = bit5;
    bit5: if(count_load == 0) nxt_state = bit6;
    bit6: if(count_load == 0) nxt_state = bit7;
    bit7: if(count_load == 0) nxt_state = parity;
    parity: if(count_load == 0) nxt_state = stop;
    stop: if(count_load == 0 && stop_bit == 1) nxt_state = idle;
    //load: if(count_load == 0) nxt_state = idle;
  endcase
  /*Logica Combinacional del siguiente estado en modo Transceiver*/
  case(cur_state_Tx)
    idle:
    begin
      if(transceiver) nxt_state_Tx = start;
    end
    start: nxt_state_Tx = bit0;
    bit0: nxt_state_Tx = bit1;
    bit1: nxt_state_Tx = bit2;
    bit2: nxt_state_Tx = bit3;
    bit3: nxt_state_Tx = bit4;
    bit4: nxt_state_Tx = bit5;
    bit5: nxt_state_Tx = bit6;
    bit6: nxt_state_Tx = bit7;
    bit7: nxt_state_Tx = parity;
    parity: nxt_state_Tx = stop;
    stop: nxt_state_Tx = idle;
  endcase
end
endmodule
```

MIPS_Top.v

```
`timescale 1ns/100ps
module MIPS_Top(clk, rst, inA_UART, inOpCode_UART, inB_UART, outTransmit, outA_done, outOpCode_done,
outB_done, outResultado, signalflag);
  input clk;
  input rst;
  ////// UART Interface //////
  input signalflag;
  input  [31:0] inA_UART;      //Done
  input  [31:0] inOpCode_UART;  //Done
  input  [31:0] inB_UART;      //Done
  output [31:0] outResultado;   //Done
  output        outTransmit;
  output        outA_done;
  output        outOpCode_done;
  output        outB_done;
  ////////////////////////////

  wire wsignalflag;
  // PC wires
  wire [31:0] wPC_in;
  wire        wPCenable_in;
  wire [31:0] wPC_out;
  wire        wAnd_out;
  wire [31:0] wPCMUX_out;

  //ControlUnit wires
  wire        wPCWriteCond_out;
  wire        wPCWrite_out;
  wire        wIorD_out;
  wire  [1:0]  wPCSource_out;
  wire        wMemRead_out;
  wire        wMemWrite_out;
  wire        wIRWrite_out;
  wire        wRegDst_out;
  wire        wMemtoReg_out;
  wire        wALUSrcA_out;
  wire  [1:0]  wALUSrcB_out;
  wire  [1:0]  wALUOp_out;

  //ALU wires
  wire [31:0] wALUOut_out;
  wire        wZero_out;
  wire [3:0]   wALUcntrl_out;
  wire [31:0] wALUResult_out;

  //Memory Data wires
  wire [31:0] wMemData_out;
  wire [31:0] wMemDataReg_out;
  wire [31:0] wMDRMUX_out;

  //Instruction Register wires
  wire [5:0]   wInst_31_26_out;
  wire [4:0]   wInst_25_21_out;
  wire [4:0]   wInst_20_16_out;
  wire [15:0]  wInst_15_0_out;
```

```
  wire [25:0] wInst_25_0_out;
  wire [4:0]   wInst_15_11_out;
  wire [5:0]   wInst_5_0_out;
  wire [4:0]   wIRMUX_out;

  //Registers wires
  wire [31:0] wRegA_out;
  wire [31:0] wRegB_out;
  wire [31:0] wRegAMUX_out;
  wire [31:0] wRegBMUX_out;
  wire        wRegWrite_out;
  wire [31:0] wRegData1_out;
  wire [31:0] wRegData2_out;
  //Extra Modules wires
  wire [31:0] wSE32_out;
  wire [31:0] wSL2_1_out;
  wire [27:0] wSL2_2_out; //42

  /****** PC ******/
  assign wAnd_out = (wZero_out & wPCWriteCond_out);
  assign wPCenable_in = (wAnd_out | wPCWrite_out);
  assign wsignalflag = signalflag;

PC PC1(.clk(clk),.rst(rst),.pc_enable(wPCenable_in),.pc_in(wPC_in),.pc_out(wPC_out));

assign wPCMUX_out = wIorD_out ? wALUOut_out : ({2'b00,wPC_out[31:2]});

/****** Memory Data ******/
MemoryData MD1(.clk(clk),.rst(rst),.rd_mem(wMemRead_out),.wr_mem(wMemWrite_out),
                        .address(wPCMUX_out),.in_data(wRegB_out),.out_data(wMemData_out));

/****** Instruction Register ******/
InstructionReg IR1(.clk(clk),.rst(rst),.wr_enable(wIRWrite_out),.in_data(wMemData_out),
                   .out_inst_31_26(wInst_31_26_out),.out_inst_25_21(wInst_25_21_out),
                   .out_inst_20_16(wInst_20_16_out),.out_inst_15_0(wInst_15_0_out),
                   .out_inst_25_0(wInst_25_0_out),.out_inst_15_11(wInst_15_11_out),
                   .out_inst_5_0(wInst_5_0_out));

assign wIRMUX_out = wRegDst_out ? wInst_15_11_out : wInst_20_16_out;

/****** Memory Data Register ******/
MemoryDataReg MDR1(.clk(clk),.data_in(wMemData_out),.data_out(wMemDataReg_out));

assign wMDRMUX_out = wMemtoReg_out ? wMemDataReg_out : wALUOut_out;

/****** Registers ******/
Registers Reg1(.clk(clk),.rst(rst),.wr_enable(wRegWrite_out),.in_rd_reg1(wInst_25_21_out),
               .in_rd_reg2(wInst_20_16_out),.in_wr_reg(wIRMUX_out),.in_wr_data(wMDRMUX_out),
                 .out_rd_data1(wRegData1_out),.out_rd_data2(wRegData2_out),
                 .inA_UART(inA_UART),
                 .inOpCode_UART(inOpCode_UART),
                 .inB_UART(inB_UART),
                 .outResultado(outResultado),
                 .outTransmit(outTransmit),
                 .outA_done(outA_done),
                 .outOpCode_done(outOpCode_done),
```

```
                .outB_done(outB_done),
                .signalflag(wsignalflag));

//Register A
RegAux RegA(.clk(clk),.in_data(wRegData1_out),.out_data(wRegA_out));

assign wRegAMUX_out = wALUSrcA_out ? wRegA_out : wPC_out;

//Register B
RegAux RegB(.clk(clk),.in_data(wRegData2_out),.out_data(wRegB_out));

assign wRegBMUX_out = (wALUSrcB_out == 2'd0) ? wRegB_out :
                      (wALUSrcB_out == 2'd1) ? 32'd4 :
                      (wALUSrcB_out == 2'd2) ? wSE32_out : wSL2_1_out;

/****** Sign Extend ******/
SignExtend SE(.in_data(wInst_15_0_out),.out_data(wSE32_out));


/****** Shift Left 2 ******/
ShiftLeft2 #(6'd32,6'd32) SL2_32_32(.in_data(wSE32_out),.out_data(wSL2_1_out));

//Shift Left 2 (26 -> 28)
ShiftLeft2 #(6'd26,6'd28) SL2_26_28(.in_data(wInst_25_0_out),.out_data(wSL2_2_out));

assign wPC_in = (wPCSource_out == 2'd0) ? wALUResult_out :
                (wPCSource_out == 2'd1) ? wALUOut_out : {wPC_out[31:28], wSL2_2_out};

/****** ALU Control ******/
ALUControl ALUC1(.in_Op(wALUOp_out),.in_function(wInst_5_0_out),.out_control(wALUcntrl_out));

/****** ALU ******/
ALU ALU1(.in_data_A(wRegAMUX_out),.in_data_B(wRegBMUX_out),.in_control(wALUcntrl_out),
        .out_result(wALUResult_out),.out_zero(wZero_out));

/****** ALU Out ******/
ALUOut AO(.clk(clk),.in_data(wALUResult_out),.out_data(wALUOut_out));
/****** Control Unit ******/
ControlUnit CU1(.clk(clk),.rst(rst),.Op(wInst_31_26_out),.MemtoReg(wMemtoReg_out),
                .RegWrite(wRegWrite_out),.PCWriteCond(wPCWriteCond_out),.IorD(wIorD_out),
                .IRWrite(wIRWrite_out),.PCWrite(wPCWrite_out),.ALUOp(wALUOp_out),
                .RegDst(wRegDst_out),.MemWrite(wMemWrite_out),.ALUSrcA(wALUSrcA_out),
                .MemRead(wMemRead_out),.PCSource(wPCSource_out),.ALUSrcB(wALUSrcB_out));

endmodule
```



Arquitectura General de un procesador MIPS y la integración de UART

MemoryData.v

```
`timescale 1ns/100ps
module MemoryData(clk, rst, address, in_data, rd_mem, wr_mem, out_data);
input  clk;
input  rst;
input  rd_mem; //MemRead
input  wr_mem; //MemWrite
input  [31:0] address;
input  [31:0] in_data;
output [31:0] out_data;


reg [31:0] mem_array [0:63]; //Memoria
//reg [31:0] out_data;

always@(posedge clk, posedge rst) begin
    if(rst) begin
    //— Instrucciones ——— Usadas para Proyecto Final
      mem_array[0] <= {6'd35, 5'd0, 5'd0, 16'd32};
      mem_array[1] <= {6'd35, 5'd0, 5'd28, 16'd32};
      mem_array[2] <= {6'd35, 5'd0, 5'd29, 16'd32};
      mem_array[3] <= {6'd35, 5'd0, 5'd30, 16'd32};
      mem_array[4] <= {6'd35, 5'd0, 5'd31, 16'd32};
      mem_array[5] <= {6'd35, 5'd0, 5'd27, 16'd32};
      mem_array[6] <= {6'd35, 5'd0, 5'd1, 16'd36};
      mem_array[7] <= {6'd35, 5'd0, 5'd2, 16'd37};
      //————————————
      mem_array[8] <= {6'd4, 5'd0, 5'd28, 16'd65535};
```

```
        mem_array[9] <= {6'd43, 5'd0, 5'd28, 16'd33};
        mem_array[10] <= {6'd35, 5'd0, 5'd28, 16'd32};
        mem_array[11] <= {6'd35, 5'd0, 5'd3, 16'd33};
        //-----------------
        mem_array[12] <= {6'd4, 5'd0, 5'd29, 16'd65535};
        mem_array[13] <= {6'd43, 5'd0, 5'd29, 16'd34};
        mem_array[14] <= {6'd35, 5'd0, 5'd29, 16'd32};
        mem_array[15] <= {6'd35, 5'd0, 5'd4, 16'd34};
        //-----------------
        mem_array[16] <= {6'd4, 5'd0, 5'd30, 16'd65535};
        mem_array[17] <= {6'd43, 5'd0, 5'd30, 16'd35};
        mem_array[18] <= {6'd35, 5'd0, 5'd30, 16'd32};
        mem_array[19] <= {6'd35, 5'd0, 5'd5, 16'd35};
        mem_array[20] <= {6'd35, 5'd0, 5'd27, 16'd38};
        //-----------------
        mem_array[21] <= {6'd4, 5'd1, 5'd4, 16'd2};
        mem_array[22] <= {6'd4, 5'd2, 5'd4, 16'd4};
        mem_array[23] <= {6'd2, 26'd0};
        mem_array[24] <= {6'd0, 5'd3, 5'd5, 5'd31, 5'd0, 6'd32};
        mem_array[25] <= {6'd35, 5'd0, 5'd27, 16'd39};
        mem_array[26] <= {6'd2, 26'd0};
        mem_array[27] <= {6'd0, 5'd3, 5'd5, 5'd31, 5'd0, 6'd34};
        mem_array[28] <= {6'd35, 5'd0, 5'd27, 16'd39};
        mem_array[29] <= {6'd2, 26'd0};

      //------ Datos ---------
        mem_array[32] <= 32'd0;
        mem_array[36] <= 32'd43; //SUMA (00000000000000000000000000101011)
        mem_array[37] <= 32'd45; //SUMA (00000000000000000000000000101101)
        mem_array[38] <= 32'd7;//Señales de Control para UART
        mem_array[39] <= 32'd23; //Señales de Control para UART
    end else begin
        if(wr_mem)
          mem_array[address] <= in_data;
      end
    end

    assign out_data = mem_array[address];

endmodule
```

Registers.v

```
`timescale 1ns/100ps
module Registers(clk, rst, wr_enable, in_rd_reg1, in_rd_reg2, in_wr_reg, in_wr_data,
                 inA_UART, inOpCode_UART, inB_UART, outResultado,
                 outTransmit, outA_done, outOpCode_done, outB_done,
                 out_rd_data1, out_rd_data2, signalflag);
/*
    output       outTransmit;
    output       outA_done;
    output       outOpCode_done;
    output       outB_done;
*/
    input        clk;
    input        rst;
    input        wr_enable; //RegWrite
    input        signalflag;
```

```
input   [4:0] in_rd_reg1;
input   [4:0] in_rd_reg2;
input   [4:0] in_wr_reg;
input   [31:0] in_wr_data;

// MIPS Interface with UART //
input   [31:0] inA_UART;
input   [31:0] inOpCode_UART;
input   [31:0] inB_UART;
output reg [31:0] outResultado;

output        outTransmit;
output        outA_done;
output        outOpCode_done;
output        outB_done;
//////////////////////////////

output[31:0] out_rd_data1;
output[31:0] out_rd_data2;

reg     [31:0] reg_array [0:31]; //Registers
reg flag;

always@(posedge clk, posedge rst) begin
  //reg_array[5'd28] <= inA_UART;
  //reg_array[5'd29] <= inOpCode_UART;
  //reg_array[5'd30] <= inB_UART;
  //reg_array[5'd31][32'd3] <= 1'b1;
  if(rst) begin
    flag <= 1;
    reg_array[0]  <= 32'd0;
    reg_array[1]  <= 32'd0;
    reg_array[2]  <= 32'd0;
    reg_array[3]  <= 32'd0;
    reg_array[4]  <= 32'd0;
    reg_array[5]  <= 32'd0;
    reg_array[6]  <= 32'd0;
    reg_array[7]  <= 32'd0;
    reg_array[8]  <= 32'd0;
    reg_array[9]  <= 32'd0;
    reg_array[10] <= 32'd0;
    reg_array[11] <= 32'd0;
    reg_array[12] <= 32'd0;
    reg_array[13] <= 32'd0;
    reg_array[14] <= 32'd0;
    reg_array[15] <= 32'd0;
    reg_array[16] <= 32'd0;
    reg_array[17] <= 32'd0;
    reg_array[18] <= 32'd0;
    reg_array[19] <= 32'd0;
    reg_array[20] <= 32'd0;
    reg_array[21] <= 32'd0;
    reg_array[22] <= 32'd0;
    reg_array[23] <= 32'd0;
    reg_array[24] <= 32'd0;
    reg_array[25] <= 32'd0;
    reg_array[26] <= 32'd0;
```

```
        reg_array[27] <= 32'd0; //Reserved registers for UART Control signal
        reg_array[28] <= 32'd0; //Reserved registers for Input to the MIPS
        reg_array[29] <= 32'd0; //Reserved registers for Input to the MIPS
        reg_array[30] <= 32'd0; //Reserved registers for Input to the MIPS
        reg_array[31] <= 32'd0; //Reserved registers for Output from the MIPS
      end else begin
                        reg_array[28][7:0] <= inA_UART;
                        reg_array[29][7:0] <= inOpCode_UART;
                        reg_array[30][7:0] <= inB_UART;
                        reg_array[28][31:8] <= 0;
                        reg_array[29][31:8] <= 0;
                        reg_array[30][31:8] <= 0;
        if(wr_enable && in_wr_reg != 5'd28 && in_wr_reg != 5'd29 && in_wr_reg != 5'd30) begin
        //if(wr_enable) begin
          reg_array[in_wr_reg] <= in_wr_data;
        end
        if(outTransmit && flag) outResultado <= reg_array[5'd31]; else outResultado <= outResultado;
        if(outTransmit) flag <= 0;
        if(signalflag) flag <= 1;
      end
    end

    assign out_rd_data1 = reg_array[in_rd_reg1];
    assign out_rd_data2 = reg_array[in_rd_reg2];

    // MIPS Output Interface with UART //
    //assign outResultado = (outTransmit && flag) ? reg_array[5'd31] : outResultado;
    assign outA_done      = reg_array[5'd27][32'd0];
    assign outOpCode_done = reg_array[5'd27][32'd1];
    assign outB_done      = reg_array[5'd27][32'd2];
    assign outTransmit    = reg_array[5'd27][32'd4];

endmodule
```

pb_pulse.v

```
module pb_pulse(clk, pb_in, pulse_out);
input clk; // clock signal
input pb_in; // push-button
output pulse_out; // pulse output
// intenal registers
reg delay1;
reg delay2;
reg delay3;
// synchronizing input signal
always@(posedge clk) begin
delay1 <= pb_in;
delay2 <= delay1;
delay3 <= delay2;
end
assign pulse_out = delay3;
endmodule
```

SignExtend.v

```
`timescale 1ns/100ps
module SignExtend(in_data,out_data);
  input  [15:0] in_data;
  output [31:0] out_data;

  reg    [31:0] reg_out_data;

  always@(*) begin
    if(in_data[15] == 1'b1)begin
      reg_out_data = {16'hFFFF,in_data[15:0]};
    end else begin
      reg_out_data = {16'h0000,in_data[15:0]};
    end
  end

  assign out_data = reg_out_data;
endmodule
```

ALUOut.v

```
`timescale 1ns/100ps
module ALUOut(clk, in_data, out_data);
  input  clk;
  input  [31:0] in_data;
  output [31:0] out_data;

  reg    [31:0] reg_out_data;

  always@(posedge clk) begin
  //always@(negedge clk) begin
    reg_out_data <= in_data;
  end

  assign out_data = reg_out_data;
endmodule
```

ShiftLeft2.v

```
`timescale 1ns/100ps
module ShiftLeft2(in_data, out_data);
  parameter LENGTH_in  = 6'b100000; //Tamaño: 6'd32 default (6'b100000)
  parameter LENGTH_out = 6'b100000; //Tamaño: 6'd32 default (6'b100000)

  input   [LENGTH_in - 1:0] in_data;
  output [LENGTH_out - 1:0] out_data;
  reg    [LENGTH_out - 1:0] reg_data;

  always@(*) begin
    reg_data = in_data << 2;
  end

  assign out_data = reg_data;
endmodule
```

PC.v

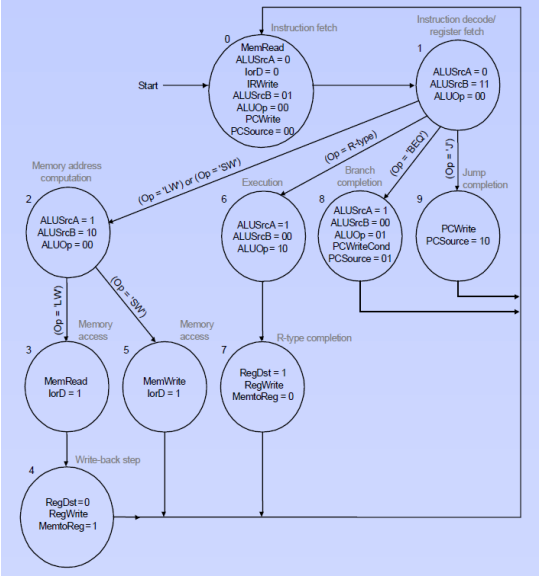
```
`timescale 1ns/100ps
module PC(clk, rst, pc_enable, pc_in, pc_out);
  parameter WIDTH = 6'b100000; //32 bits

  input              clk;
  input              rst;
  input              pc_enable;
  input  [WIDTH - 1:0] pc_in;
  output [WIDTH - 1:0] pc_out;

  reg    [WIDTH - 1:0] pc_out_reg;

  always@(posedge clk) begin
    if(rst) begin
      pc_out_reg <= 0;
    end else begin
      if(pc_enable) begin
        pc_out_reg <= pc_in;
      end
    end
  end

  assign pc_out = pc_out_reg;
endmodule
```

ALUControl.v

```
`timescale 1ns/100ps
module ALUControl(in_function, in_Op, out_control);
 input      [5:0] in_function;
 input      [1:0] in_Op;
 output reg [3:0] out_control;

 always@(*) begin
   case(in_Op)
     2'b00: out_control = 4'b0010; //Suma
     2'b01: out_control = 4'b0110; //Resta
     2'b10:
       begin
         case(in_function)
           6'b100000: out_control = 4'b0010; //Suma
           6'b100010: out_control = 4'b0110; //Resta
           6'b100100: out_control = 4'b0000; //AND
           6'b100101: out_control = 4'b0001; //XOR
           6'b101010: out_control = 4'b0111; //Set on less than
           6'b000011: out_control = 4'b0011; //Shift Left
           6'b111111: out_control = 4'b1111; //Shift Right
                                             default: out_control = 4'bzzzz;   //Error: Function not recognized
         endcase
       end
                         default: out_control = 4'bzzzz; //Error: Operation not recognized
   endcase
 end
endmodule
```

InstructionReg.v

```
`timescale 1ns/100ps
module InstructionReg(clk, rst, wr_enable, in_data, out_inst_31_26, out_inst_25_21, out_inst_20_16,
out_inst_15_0, out_inst_25_0, out_inst_15_11, out_inst_5_0);
  input         clk;
  input         rst;
  input         wr_enable; //IRWrite
  input  [31:0] in_data;
  output  [5:0] out_inst_31_26;
  output  [4:0] out_inst_25_21;
  output  [4:0] out_inst_20_16;
  output [15:0] out_inst_15_0;
  output [25:0] out_inst_25_0;
  output  [4:0] out_inst_15_11;
  output  [5:0] out_inst_5_0;
  reg     [5:0] reg_inst_31_26;
  reg     [4:0] reg_inst_25_21;
  reg     [4:0] reg_inst_20_16;
  reg    [15:0] reg_inst_15_0;
  reg    [25:0] reg_inst_25_0;
  reg     [4:0] reg_inst_15_11;
  reg     [5:0] reg_inst_5_0;
  always@(posedge clk, posedge rst) begin
    if(rst) begin
      reg_inst_31_26 <= 0;
      reg_inst_25_21 <= 0;
      reg_inst_20_16 <= 0;
      reg_inst_15_0  <= 0;
      reg_inst_25_0  <= 0;
      reg_inst_15_11 <= 0;
      reg_inst_5_0   <= 0;
    end else begin

      if(wr_enable) begin
        reg_inst_31_26 <= in_data[31:26];
        reg_inst_25_21 <= in_data[25:21];
        reg_inst_20_16 <= in_data[20:16];
        reg_inst_15_0  <= in_data[15:0];
        reg_inst_25_0  <= in_data[25:0];
        reg_inst_15_11 <= in_data[15:11];
        reg_inst_5_0   <= in_data[5:0];
      end
    end
  end

  assign out_inst_31_26 = reg_inst_31_26;
  assign out_inst_25_21 = reg_inst_25_21;
  assign out_inst_20_16 = reg_inst_20_16;
  assign out_inst_15_0  = reg_inst_15_0;
  assign out_inst_25_0  = reg_inst_25_0;
  assign out_inst_15_11 = reg_inst_15_11;
  assign out_inst_5_0   = reg_inst_5_0;

endmodule
```



Señales de control para la FSM de un MIPS

ControlUnit.v

```
`timescale 1ns/100ps
module ControlUnit(clk, rst, Op, RegDst, RegWrite, ALUSrcA, MemRead, MemWrite, MemtoReg, IorD, IRWrite,
PCWrite, PCWriteCond, ALUOp, ALUSrcB, PCSource);
  input       clk;
  input       rst;
  input  [5:0] Op;
  output      RegDst;
  output      RegWrite;
  output      ALUSrcA;
  output      MemRead;
  output      MemWrite;
  output      MemtoReg;
  output      IorD;
  output      IRWrite;
  output      PCWrite;
  output      PCWriteCond;
  output [1:0] ALUOp;
  output [1:0] ALUSrcB;
  output [1:0] PCSource;

  reg      RegDst;
  reg      RegWrite;
  reg      ALUSrcA;
  reg      MemRead;
  reg      MemWrite;
  reg      MemtoReg;
  reg      IorD;
  reg      IRWrite;
  reg      PCWrite;
  reg      PCWriteCond;
  reg [1:0] ALUOp;
  reg [1:0] ALUSrcB;
  reg [1:0] PCSource;
```

```
// FSM states encoding
parameter [3:0] sFETCH  = 4'b0000,
                sDECODE = 4'b0001,
                sMAC    = 4'b0010,
                sMAR    = 4'b0011,
                sMRCS   = 4'b0100,
                sMAW    = 4'b0101,
                sEXEC   = 4'b0110,
                sRTC    = 4'b0111,
                sBEQ    = 4'b1000,
                sJUMP   = 4'b1001,
                sADDI   = 4'b1010,
                sRTC2   = 4'b1011;

// FSM states registers
reg [3:0] cur_state;
reg [3:0] nxt_state;

// FSM registered process
always@(posedge clk, posedge rst) begin
  if (rst) begin
    cur_state <= sFETCH;
  end else begin
    cur_state <= nxt_state;
  end
end

always@(*) begin
  RegDst     = 1'b0;  //MUX
                    MemtoReg    = 1'b0;  //MUX
                    ALUSrcA     = 1'b0;  //MUX
                    IorD        = 1'b0;  //MUX
                    ALUSrcB     = 2'b00; //MUX
                    PCSource    = 2'b00; //MUX

                    MemRead     = 1'b0;
                    IRWrite     = 1'b0;
                    ALUOp       = 2'b00;
                    PCWrite     = 1'b0;
                    MemWrite    = 1'b0;
                    RegWrite    = 1'b0;
                    PCWriteCond = 1'b0;
                    case(cur_state)
                              sFETCH:
                              begin
                                        RegDst      = 1'b0;  //MUX
                                        MemtoReg    = 1'b0;  //MUX
                                        ALUSrcA     = 1'b0;  //MUX
                                        IorD        = 1'b0;  //MUX
                                        ALUSrcB     = 2'b01; //MUX
                                        PCSource    = 2'b00; //MUX

                                        MemRead     = 1'b1;
                                        IRWrite     = 1'b1;
                                        ALUOp       = 2'b00;
                                        PCWrite     = 1'b1;
```

```
        MemWrite    = 1'b0;
        RegWrite    = 1'b0;
        PCWriteCond = 1'b0;

        nxt_state   = sDECODE;
end

sDECODE:
begin
        RegDst      = 1'b0;  //MUX
        ALUSrcA     = 1'b0;  //MUX
        MemtoReg    = 1'b0;  //MUX
        IorD        = 1'b0;  //MUX
        ALUSrcB     = 2'b11; //MUX
        PCSource    = 2'b00; //MUX

        RegWrite    = 1'b0;
        MemRead     = 1'b0;
        MemWrite    = 1'b0;
        IRWrite     = 1'b0;
        PCWrite     = 1'b0;
        PCWriteCond = 1'b0;
        ALUOp       = 2'b00;

        case(Op)
                6'd35  : nxt_state = sMAC;  //Load
                6'd43  : nxt_state = sMAC;  //Store
                6'd0   : nxt_state = sEXEC; //R-type
                6'd4   : nxt_state = sBEQ;  //Branch
                6'd2   : nxt_state = sJUMP; //Jump
                6'd8   : nxt_state = sADDI; //Addi
                6'd6   : nxt_state = sADDI; //Shift
                default: nxt_state = sFETCH;
        endcase
end

sMAC:
begin
        RegDst      = 1'b0;  //MUX
        ALUSrcA     = 1'b1;  //MUX
        MemtoReg    = 1'b0;  //MUX
        IorD        = 1'b0;  //MUX
        ALUSrcB     = 2'b10; //MUX
        PCSource    = 2'b00; //MUX

        RegWrite    = 1'b0;
        MemRead     = 1'b0;
        MemWrite    = 1'b0;
        IRWrite     = 1'b0;
        PCWrite     = 1'b0;
        PCWriteCond = 1'b0;
        ALUOp       = 2'b00;
        if(Op == 6'd35)
                nxt_state = sMAR;
        else if(Op == 6'd43)
                nxt_state = sMAW;
        else
```

```
                    nxt_state = sFETCH;
end
sMAR:
begin
        RegDst     = 1'b0; //MUX
        ALUSrcA    = 1'b1; //MUX
        MemtoReg   = 1'b0; //MUX
        IorD       = 1'b1; //MUX
        ALUSrcB    = 2'b10; //MUX
        PCSource   = 2'b00; //MUX
        RegWrite   = 1'b0;
        MemRead    = 1'b1;
        MemWrite   = 1'b0;
        IRWrite    = 1'b0;
        PCWrite    = 1'b0;
        PCWriteCond = 1'b0;
        ALUOp      = 2'b00;
        nxt_state = sMRCS;
end
sMRCS:
begin
        RegDst     = 1'b0; //MUX
        MemtoReg   = 1'b1; //MUX
        ALUSrcA    = 1'b1; //MUX
        IorD       = 1'b1; //MUX
        ALUSrcB    = 2'b10; //MUX
        PCSource   = 2'b00; //MUX

        RegWrite   = 1'b1;
        MemRead    = 1'b0;
        MemWrite   = 1'b0;
        IRWrite    = 1'b0;
        PCWrite    = 1'b0;
        PCWriteCond = 1'b0;
        ALUOp      = 2'b00;

        nxt_state = sFETCH;
end
sMAW:
begin
        RegDst     = 1'b0; //MUX
        MemtoReg   = 1'b0; //MUX
        ALUSrcA    = 1'b1; //MUX
        IorD       = 1'b1; //MUX
        ALUSrcB    = 2'b10; //MUX
        PCSource   = 2'b00; //MUX

        RegWrite   = 1'b0;
        MemRead    = 1'b0;
        MemWrite   = 1'b1;
        IRWrite    = 1'b0;
        PCWrite    = 1'b0;
        PCWriteCond = 1'b0;
        ALUOp      = 2'b00;

        nxt_state = sFETCH;
end
```

```
sEXEC:
begin
        RegDst     = 1'b0;  //MUX
        MemtoReg   = 1'b0;  //MUX
        ALUSrcA    = 1'b1;  //MUX
        IorD       = 1'b0;  //MUX
        ALUSrcB    = 2'b00; //MUX
        PCSource   = 2'b00; //MUX

        RegWrite   = 1'b0;
        MemRead    = 1'b0;
        MemWrite   = 1'b0;
        IRWrite    = 1'b0;
        PCWrite    = 1'b0;
        PCWriteCond = 1'b0;
        ALUOp      = 2'b10;
        nxt_state = sRTC;
end
sRTC:
begin
        RegDst     = 1'b1;  //MUX
        MemtoReg   = 1'b0;  //MUX
        ALUSrcA    = 1'b1;  //MUX
        IorD       = 1'b0;  //MUX
        if(Op == 6'd6)
          ALUSrcB    = 2'b10; //MUX
        else
          ALUSrcB    = 2'b00; //MUX
        PCSource   = 2'b00; //MUX

        RegWrite   = 1'b1;
        MemRead    = 1'b0;
        MemWrite   = 1'b0;
        IRWrite    = 1'b0;
        PCWrite    = 1'b0;
        PCWriteCond = 1'b0;
        //ALUOp      = 2'b00;
        ALUOp      = 2'b10;

        nxt_state = sFETCH;
end
sBEQ:
begin
        RegDst     = 1'b0;  //MUX
        MemtoReg   = 1'b0;  //MUX
        ALUSrcA    = 1'b1;  //MUX
        IorD       = 1'b0;  //MUX
        ALUSrcB    = 2'b00; //MUX
        PCSource   = 2'b01; //MUX
        RegWrite   = 1'b0;
        MemRead    = 1'b0;
        MemWrite   = 1'b0;
        IRWrite    = 1'b0;
        PCWrite    = 1'b0;
        PCWriteCond = 1'b1;
        ALUOp      = 2'b01;
```

```
            nxt_state   = sFETCH;
end
sJUMP:
begin
            RegDst      = 1'b0;  //MUX
            MemtoReg    = 1'b0;  //MUX
            ALUSrcA     = 1'b0;  //MUX
            IorD        = 1'b0;  //MUX
            ALUSrcB     = 2'b11; //MUX
            PCSource    = 2'b10; //MUX
            RegWrite    = 1'b0;
            MemRead     = 1'b0;
            MemWrite    = 1'b0;
            IRWrite     = 1'b0;
            PCWrite     = 1'b1;
            PCWriteCond = 1'b0;
            ALUOp       = 2'b00;
            nxt_state = sFETCH;
end
sADDI:
begin
            RegDst      = 1'b0;  //MUX
            MemtoReg    = 1'b0;  //MUX
            ALUSrcA     = 1'b1;  //MUX
            IorD        = 1'b0;  //MUX
            ALUSrcB     = 2'b10; //MUX
            PCSource    = 2'b00; //MUX


            RegWrite    = 1'b0;
            MemRead     = 1'b0;
            MemWrite    = 1'b0;
            IRWrite     = 1'b0;
            PCWrite     = 1'b0;
            PCWriteCond = 1'b0;

            case(Op)
                        6'd8: begin
                          ALUOp     = 2'b00;
                          nxt_state = sRTC2;
                        end

                        6'd6: begin
                          ALUOp     = 2'b10;
                          nxt_state = sRTC;
                        end
                        default: nxt_state = sFETCH;
            endcase
end
sRTC2:
begin
            RegDst      = 1'b0;  //MUX
            MemtoReg    = 1'b0;  //MUX
            ALUSrcA     = 1'b1;  //MUX
            IorD        = 1'b0;  //MUX
            ALUSrcB     = 2'b10; //MUX
            PCSource    = 2'b00; //MUX
```

```
                        RegWrite    = 1'b1;
                        MemRead     = 1'b0;
                        MemWrite    = 1'b0;
                        IRWrite     = 1'b0;
                        PCWrite     = 1'b0;
                        PCWriteCond = 1'b0;
                        ALUOp       = 2'b00;
                        nxt_state = sFETCH;
                    end
                    default:
                    begin
                        RegDst      = 1'b0;  //MUX
                        MemtoReg    = 1'b0;  //MUX
                        ALUSrcA     = 1'b0;  //MUX
                        IorD        = 1'b0;  //MUX
                        ALUSrcB     = 2'b01; //MUX
                        PCSource    = 2'b00; //MUX
                        MemRead     = 1'b1;
                        IRWrite     = 1'b1;
                        ALUOp       = 2'b00;
                        PCWrite     = 1'b1;
                        MemWrite    = 1'b0;
                        RegWrite    = 1'b0;
                        PCWriteCond = 1'b0;
                        nxt_state = sFETCH;
                    end
                endcase
  end
endmodule
```

MemoryDataReg.v

```
`timescale 1ns/100ps
module MemoryDataReg(clk, data_in, data_out);
  input         clk;
  input  [31:0] data_in;
  output [31:0] data_out;

  reg    [31:0] MemDataReg_out_reg;

  always@(posedge clk) begin
     MemDataReg_out_reg <= data_in;
  end

  assign data_out = MemDataReg_out_reg;
endmodule
```

RegAux.v

```
`timescale 1ns/100ps
module RegAux(clk, in_data, out_data);
  input  clk;
  input  [31:0] in_data;
  output [31:0] out_data;
  reg    [31:0] reg_out_data;
  //always@(posedge clk) begin
```

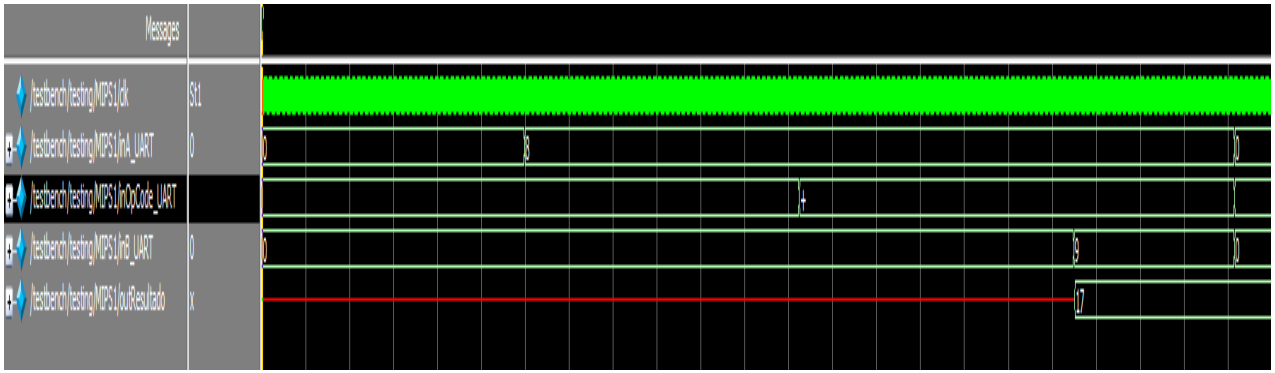
```
  always@(posedge clk) begin
     reg_out_data <= in_data;
  end

  assign out_data = reg_out_data;
endmodule
```

ALU.v

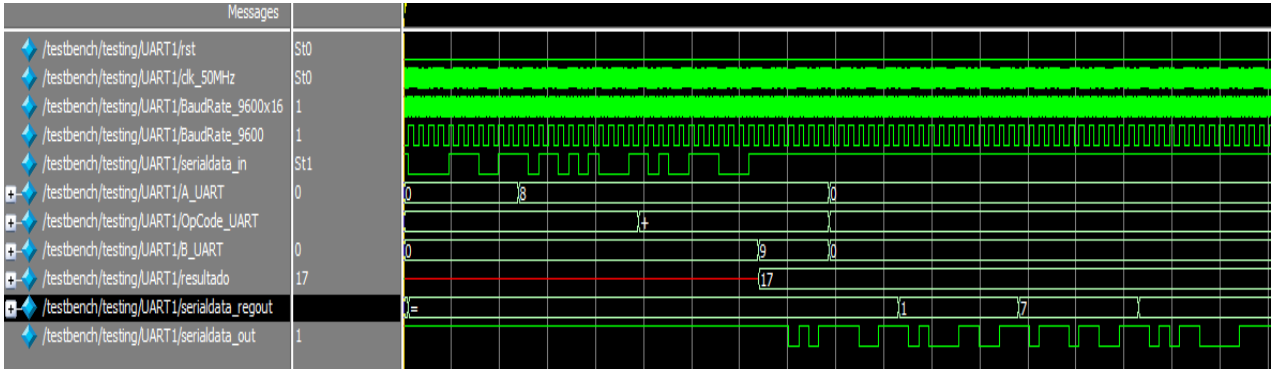
```
`timescale 1ns/100ps
module ALU(in_data_A, in_data_B, in_control, out_zero, out_result);
  input  [31:0] in_data_A;  //RS
  input  [31:0] in_data_B;  //RT
  input   [3:0] in_control;
  output        out_zero;
  output [31:0] out_result; //RD

  reg           out_zero;
  reg    [31:0] out_result;

  parameter [3:0] pADD  = 4'b0010, //Suma
                  pSUB  = 4'b0110, //Resta
                  pAND  = 4'b0000, //AND
                  pXOR  = 4'b0001, //XOR
                  pSHL  = 4'b0011, //Shift Left
                  pSHR  = 4'b1111; //Shift Right

  always@(*) begin
    case(in_control)
      pADD:  out_result = in_data_A +  in_data_B;
      pSUB:  out_result = in_data_A -  in_data_B;
      pSHL:  out_result = in_data_A << {27'd0,in_data_B[10:6]};
      pSHR:  out_result = in_data_A >> in_data_B[10:6];
      pAND:  out_result = in_data_A &  in_data_B;
      pXOR:  out_result = in_data_A ^  in_data_B;
                          default: out_result = 32'dz;    // Error: Control Line not recognized
    endcase

    if(out_result == 31'd0)
      out_zero = 31'd1;
    else
      out_zero = 31'd0;
  end

endmodule
```

Simulaciones

Formas de Onda: Tomando como ejemplo 7 + 9
Forma de Onda del resultado en el mips
Recibe 7 + 8 y el mips entrega como resultado un 17

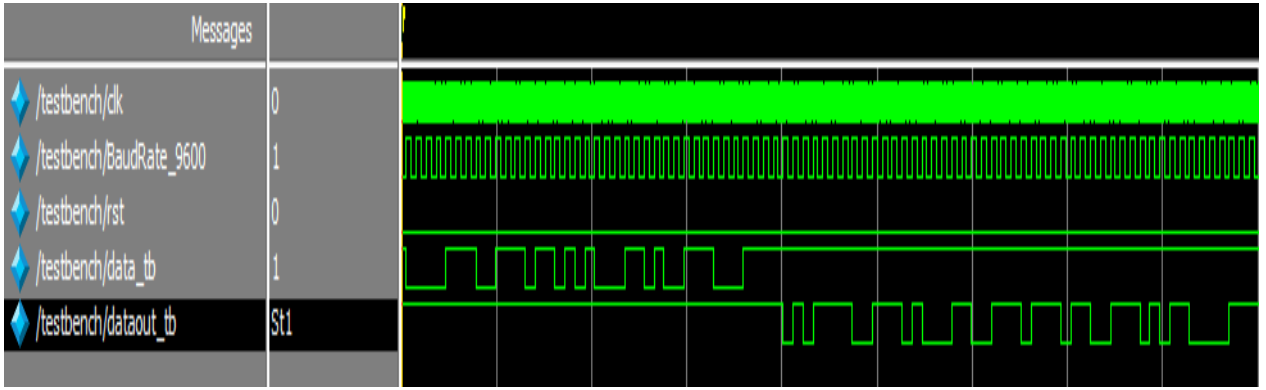


Forma de Onda del resultado y comportamiento en el UART



Conversión de binario a ASCII

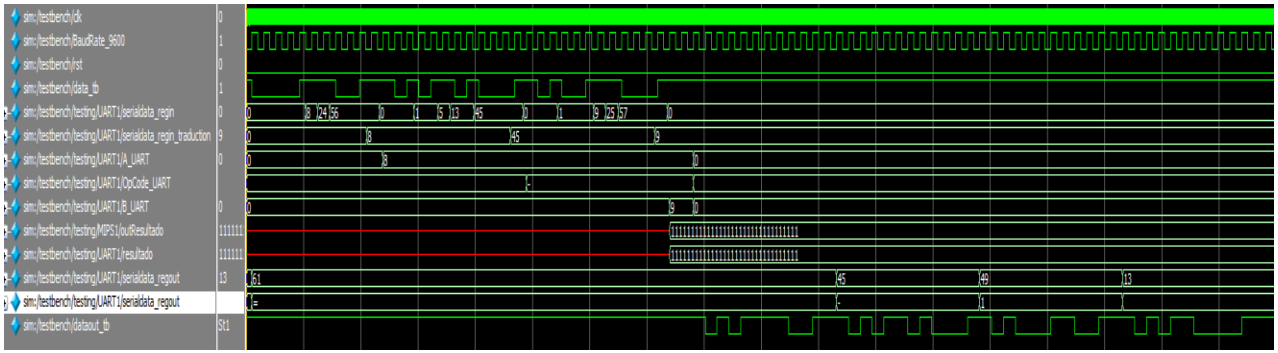
La señal que está marcada con negro se simulo (simuló) poniendo el radix en ASCII, para comprobar que fue correcta la traducción de binario a ASCII. En serialdata_out ya está enviando el resultado en ASCII.



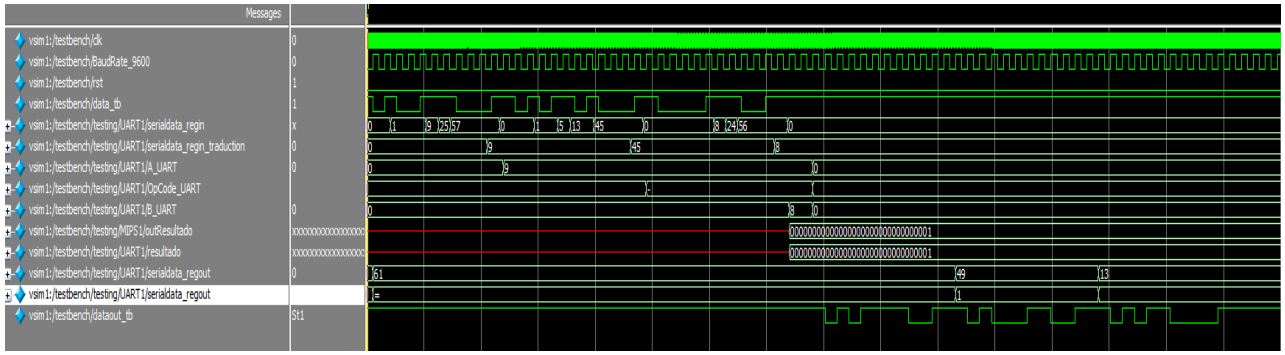
Forma de Onda del resultado y comportamiento en la etapa final

Ejemplos de resta y cuando el resultado es negativo

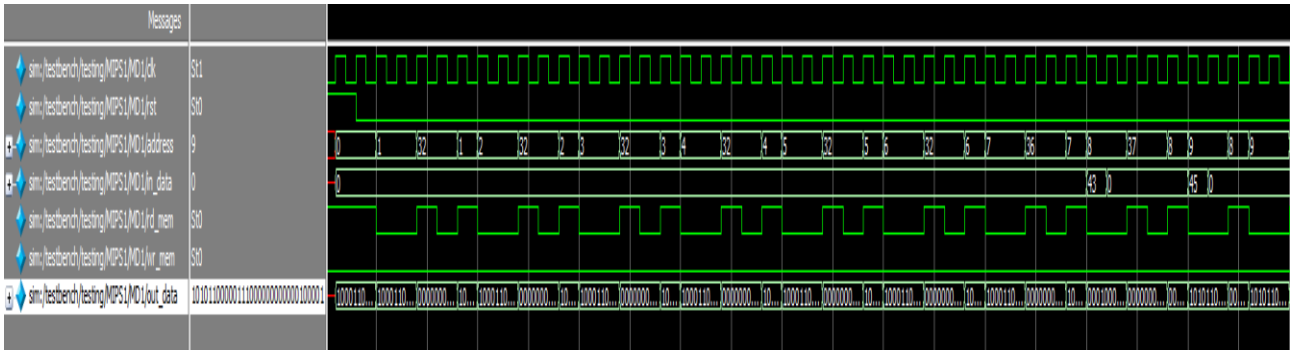
Se reciben 56, 45 y 57 es decir se recibe un 8 un signo "-" y 9, esto se pasa al MIPS, solo los números a sumar o restar son convertidos en ASCII a binario antes de pasar al MIPS, en el módulo de MIPS ser reconoce si es "+" o "-", para este caso el MIPS entrega un resultado de -1 en complemento a 2. Si el UART reconoce que es un resultado en complemento a 2 antes de trasmitir ese dato, obtiene el valor del resultado sin signo, entonces envía un signo "-" y el resultado. El resultado es enviado en código ASCII como se muestra en las últimas 3 señales.



Resta Negativa



En este caso es una resta de 9 – 8 entonces genera un 1 positivo



MemoryData Simulation

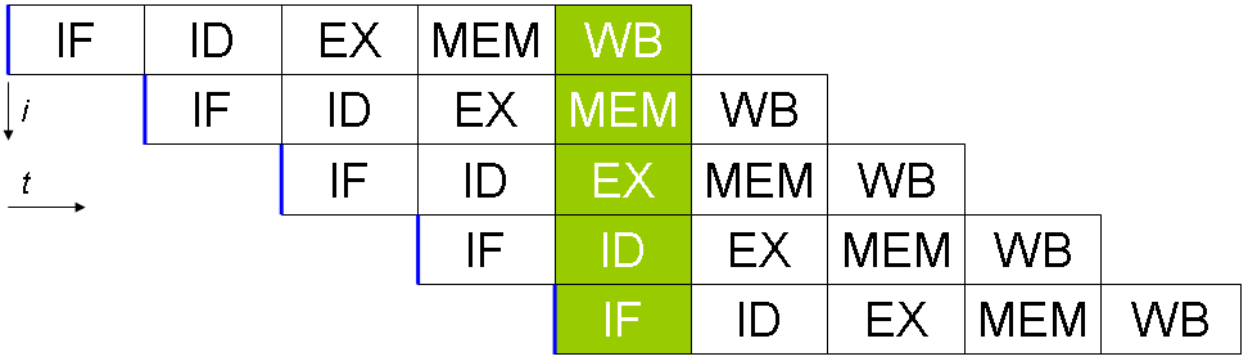
Apéndice B

RTL de Microprocesador Superescalar

En este apéndice se podrán encontrar el "Verilog Netlist" del back-end (flujo de datos) de un MIPS superescalar que formo parte de mis aportaciones. También se incluye un diagrama del modelo de Pipeline que se utilizó para el diseño.

Contenido

Pipeline utilizado:



Modelo de Pipeline utilizado

IFQ Verilog Netlist

```
module IFQ(arst, clk, d_in, wr_en, rd_en, jmp_branch_add, jmp_branch_valid, ic_pc_in, ic_rd_en, disp_pc_out,
d_out, empty_disp);

//Parameters
parameter w_size = 32;                    //Defines word width
```

```
parameter c_line_size = 128;                    //Defines chache line width
//Reset and clk inputs
input                   arst;               //Asynchronous reset
input                   clk;                //System clock
//Interfaces with Inst Cache
input[c_line_size-1:0]d_in; //Data into IFQ (4 instructions, 32bit per instr)
input               wr_en;                                      //Write enable into IFQ
output [w_size-1:0]         ic_pc_in;                                   //Program counter to Ints Cache
wire                                            full;
//IFQ full flag
output                                      ic_rd_en;                                                   //Inst
Cache read enable

//Interfaces with Dispatcher
input                                           rd_en;
//Read enable into IFQ
input[w_size-1:0]                   jmp_branch_add;     //Jump or branch address
input                                           jmp_branch_valid;                                       //Jump
or branch valid
output reg[w_size-1:0]  disp_pc_out;        //Program counter to Dispatcher
output reg[w_size-1:0]  d_out;              //Instruction to Dispatcher
output reg empty_disp;/*Empty signal to dispatcher, takes into account branch
                                        vs fifo rd==wr ptrs*/

//FIFO structure
reg[c_line_size-1:0]    fifo_ifq[3:0];                  //IFQ FIFO registers

//Internal signals
reg[w_size-1:0]             d_out_fifo;                         //Intruction output from FIFO
reg[w_size-1:0]             d_out_byp;                          //Intruction bypass output
reg[4:0]                    wr_ptr;                             //FIFO write pointer
reg[4:0]                    rd_ptr;                             //FIFO read pointer
reg[1:0]                    rd_ptr_byp;             //FIFO read pointer for bypass logic
reg[w_size-1:0]             ic_pc_in_reg;
wire                                bypass_en;
//Enable signal to bypass Instruction to Dispatcher when FIFO is empty
wire                        empty;                          //IFQ is empty when high

wire[w_size-1:0] d_out_fifo_a;
wire[w_size-1:0] d_out_fifo_b;
wire[w_size-1:0] d_out_fifo_c;
wire[w_size-1:0] d_out_fifo_d;

//Combinational assigments----------------------------

        assign empty = (rd_ptr[4:2]==wr_ptr[4:2]);
                                            //Empty flag
        assign full  = (rd_ptr[3:2]==wr_ptr[3:2])&&(rd_ptr[4]!=wr_ptr[4]);      //Full flag
        assign bypass_en = (empty && wr_en);                                                        //Bypass
enable when FIFO is empty and Dipatcher requests Instruction

//Sequential assigments-------------------------------
//Empty flag to Dispatch
always @(posedge clk or posedge arst)
begin
  if(arst)
     empty_disp <= 1;
```

```
  else if(jmp_branch_valid)
              empty_disp <= 1;
  else if(wr_en)
            empty_disp <= 0;
end
//Instruction Cache read enable control process
assign ic_rd_en = (!full || jmp_branch_valid) ? 1 : 0;
/*
always@(posedge clk or posedge arst)
begin
            if(arst)
                        ic_rd_en = 1'b1;
            else if(!full)                                          //Read Cache when IFQ is not full
                        ic_rd_en = 1'b1;
            else if(jmp_branch_valid)
                        ic_rd_en = 1'b1;
            else                                                    //Stop reading Cache when IFQ is full
                        ic_rd_en = 1'b0;
end*/
//FIFO write process
always@(posedge clk or posedge arst) begin
            if(arst) begin
                        //ic_pc_in_reg <= 32'h00000000;
                                    //Clear PC
wr_ptr <= 5'b00000;                                                 //Set write pointer
fifo_ifq[2'b00] <= 128'h00000000000000000000000000000000; //Clear FIFO
fifo_ifq[2'b01] <= 128'h00000000000000000000000000000000;
fifo_ifq[2'b10] <= 128'h00000000000000000000000000000000;
fifo_ifq[2'b11] <= 128'h00000000000000000000000000000000;
            end
            else if(jmp_branch_valid)
            begin
                        wr_ptr <= {5'b00000};
            //Bits 1:0 from wr_ptr are ignored for fifo write but needed
                        //ic_pc_in_reg <= {jmp_branch_add[31:4],4'h0};
            //PC takes Cache Line where jump/branch instruction is located                  //for empty flag
check/assgiment
fifo_ifq[2'b00] <= 128'h00000000000000000000000000000000;       //Flush FIFO
fifo_ifq[2'b01] <= 128'h00000000000000000000000000000000;
fifo_ifq[2'b10] <= 128'h00000000000000000000000000000000;
fifo_ifq[2'b11] <= 128'h00000000000000000000000000000000;
            end
            else if(wr_en && !full) begin
            fifo_ifq[wr_ptr[3:2]] <= d_in;                          //Cache Line writen to FIFO
            wr_ptr <= wr_ptr + 3'h4;//Write pointer assigned to next FIFO line
                        //ic_pc_in_reg <= ic_pc_in + 5'h10;
//PC incremented 16 to get next Cache Line
            end
            //else if(wr_en && full)
                        //ic_pc_in_reg <= ic_pc_in_reg - 5'h10;
End

//FIFO branch address bypass mux
assign ic_pc_in = jmp_branch_valid ? {jmp_branch_add[31:4],4'h0} : ic_pc_in_reg;

//FIFO read process
always@(posedge clk or posedge arst)
```

```
begin
	if(arst) begin
disp_pc_out <= 32'h00000000;			//Reset PC+4 to Dispatch
rd_ptr <= 5'b00000;//Set read pointer to second instruction due to bypass
	end
	else if(jmp_branch_valid)
	begin
		disp_pc_out <= jmp_branch_add;	//PC takes jump/branch address
		rd_ptr <= {3'b000, (jmp_branch_add[3:2])};		//Read
pointer set to instruction pointed by branch addres + 1
	end
	else if(rd_en || bypass_en)
	begin
		rd_ptr <= rd_ptr + 1'h1;//Read pointer sets to next Instrcution
		disp_pc_out <= disp_pc_out + 3'h4;		//PC out
sets to next instruction to dispatcher
	end
end
//FIFO ic pc process
always@(posedge clk or posedge arst)
begin
	if(arst)
	begin
	ic_pc_in_reg <= 32'h00000000;
		//Set read pointer to second instruction due to bypass
	end
	else if(jmp_branch_valid)
	begin
	ic_pc_in_reg <= {jmp_branch_add[31:4],4'h0} + 5'h10;
	end
	else if(ic_rd_en)
	begin
	ic_pc_in_reg <= ic_pc_in_reg + 5'h10;
		//PC out sets to next instruction to dispatcher
	end
	else if(wr_en && full)
		ic_pc_in_reg <= ic_pc_in_reg - 5'h10;
end

always@(posedge clk or posedge arst)
begin
	if(arst)
	begin
		rd_ptr_byp <= 2'b00;
					//Set bypass mux to first instruction
	end
	else if(jmp_branch_valid)
	begin
	   rd_ptr_byp <= {jmp_branch_add[3:2]};
	//Bypass mux set to instruction pointed by Branch address
	end
	else if(wr_en)
		rd_ptr_byp <= 2'b00;
end

//FIFO instruction bypass mux
always@(*)
```

```
begin
	case(rd_ptr_byp[1:0])
		2'b00: d_out_byp = d_in[31:0];
		2'b01: d_out_byp = d_in[63:32];
		2'b10: d_out_byp = d_in[95:64];
		2'b11: d_out_byp = d_in[127:96];
		default d_out_byp = d_in[31:0];
	endcase
end

//FIFO instruction output mux
assign d_out_fifo_a = fifo_ifq[rd_ptr[3:2]][31:0];
assign d_out_fifo_b = fifo_ifq[rd_ptr[3:2]][63:32];
assign d_out_fifo_c = fifo_ifq[rd_ptr[3:2]][95:64];
assign d_out_fifo_d = fifo_ifq[rd_ptr[3:2]][127:96];

always@(*) begin
	case(rd_ptr[1:0])
		2'b00: d_out_fifo = d_out_fifo_a;
		2'b01: d_out_fifo = d_out_fifo_b;
		2'b10: d_out_fifo = d_out_fifo_c;
		2'b11: d_out_fifo = d_out_fifo_d;
		default d_out_fifo = d_out_fifo_a;
	endcase
end
//IFQ Instruction output Mux
always@(posedge clk or posedge arst)
begin
	if(arst) d_out <= 32'h00000000;
	else if(jmp_branch_valid)//Clear output when branch is to be executed
		d_out <= 32'h00000000;
	else if(bypass_en)/*If FIFO is empty, bypass Instruction to Dispatcher
				at same clk when writing to FIFO*/
		d_out <= d_out_byp;
	else if(rd_en)                                          //Read instruction from FIFO
		d_out <= d_out_fifo;
end
endmodule
```

Instruction Cache Verilog Netlist

```
module I_CACHE(clk,rst,PC_in,Dout,Rd_en,Dout_valid);

input clk, rst, Rd_en;

parameter address_depth=7;
parameter word_depth =32;
parameter bank_rows  =2**address_depth;

input  [word_depth-1:0]PC_in;
output reg [(word_depth*4)-1:0]Dout;
output reg Dout_valid;

reg [address_depth:0] presetCount;
//Register Matrix
```

```
reg [word_depth-1:0] mem [0:bank_rows-1];
wire [31:0] addrShifted={2'b00, PC_in[31:4],2'b00};

//assign Dout={mem[addrShifted+3],mem[addrShifted+2],mem[addrShifted+1],mem[addrShifted]};

always @(posedge clk or posedge rst)
begin
        if (rst) begin
        for (presetCount=0; presetCount<bank_rows; presetCount=presetCount+1)
                begin
                        case (presetCount)
                        //Without ident intentionally
                        //Values are supposed to be code.
                        //mul test
                        /*
                        0: mem[presetCount]<=32'h00000020;
                        1: mem[presetCount]<=32'h00625018;
                        2: mem[presetCount]<=32'h00835818;
                        3: mem[presetCount]<=32'h00A46018;
                        4: mem[presetCount]<=32'h00C56018;
                        5: mem[presetCount]<=32'h00A21820;
                        6: mem[presetCount]<=32'h01227018;
                        7: mem[presetCount]<=32'h01037818;
                        8: mem[presetCount]<=32'h03FEE820;
                        9: mem[presetCount]<=32'h00000020;
                        10: mem[presetCount]<=32'h00631818;
                        11: mem[presetCount]<=32'h00631818;
                        12: mem[presetCount]<=32'h00631818;
                        13: mem[presetCount]<=32'h00631818;
                        14: mem[presetCount]<=32'h00000020;
                        */
                        default:  mem[presetCount]<=32'h00000000;
                        endcase
                end
        end

end

always@(posedge clk or posedge rst)
begin
  if(rst)
  begin
     Dout <= 0; //128-bit '0'
     Dout_valid <= 1'b0;
  end
  else if(Rd_en)
  begin
Dout<={mem[addrShifted+3],mem[addrShifted+2],mem[addrShifted+1],mem[addrShifted]};
     Dout_valid <= 1'b1;
  end
  else
  begin
                  Dout <= Dout;
     Dout_valid <= 1'b0;
  end
```

```
end
endmodule
```

Dispatch Verilog Netlist

```
module DispatchUnit(clk, rst, du_ren, du_jmp_br_addr, du_jmp_br_val, ifetch_empty_flag, ifetch_instruction,
ifetch_pc_4, du_opcode, du_ld_or_st, du_imm_ld_st, du_shamt, du_rs_val, du_rs_tag, du_rs_data, du_rt_val,
du_rt_tag, du_rt_data, du_rd_tag, du_en_integer0, du_en_integer1, du_en_mult, du_en_ld_st,
issueque_full_integer0, issueque_full_integer1, issueque_full_mult, issueque_full_ld_st,
//ROB outputs
RS_reg, /*rob_rs_ren,*/ RS_token, Rs_Data_spec, ROB_RS_Data_valid, Rt_reg, /*rob_rt_ren,*/ Rt_token,
Rt_Data_spec, ROB_Rt_Data_valid,
//ROB inputs
Dispatch_rd_reg, ROB_pc, ROB_dec_inst_no_rdes, ROB_dec_ld_st_sel, ROB_branch, ROB_dispatch_enable,
//Retire Bus
Retire_tag, Retire_rd_reg, Retire_data, Retire_pc, Retire_branch, Retire_branch_taken, Retire_store_ready,
Retire_valid, Retire_flush);
//Parameters
parameter w_size = 32;                    //Defines word width
parameter rf_bits2addr = 5;    //Number of bits to address the Register File

//Reset and clk inputs
input                rst;              //Asynchronous reset
input                clk;              //System clock

//Interfaces with IFQ
output reg                    du_ren;                                                    //Read
enable flag to IFQ
output[(w_size-1):0]          du_jmp_br_addr;                 //Jump or branch address to IFQ
output                                          du_jmp_br_val;  //Jump or branch valid flag to IFQ
input                                       ifetch_empty_flag;
          //When asserted high indicates IFQ is empty
input[(w_size-1):0]       ifetch_instruction;   //Instruction from IFQ
input[(w_size-1):0]       ifetch_pc_4;           //Program counter from IFQ +4 (ifq_PC + 4)

//Backdoor: Added for Instrumentation in Register File
//input[(rf_bits2addr-1):0]    RegFileAddrBackDoor;
//output[(w_size-1):0]           RegFileDataBackDoor;

//Interfaces with the Queues
//Instruction Fields
output[3:0] du_opcode;
output du_ld_or_st;
output[31:0] du_imm_ld_st; //
output[4:0]du_shamt;

output du_rs_val;
output[4:0] du_rs_tag;
output[(w_size-1):0] du_rs_data;
output du_rt_val;
output[4:0] du_rt_tag;
output[(w_size-1):0] du_rt_data;
output[4:0] du_rd_tag;    //Dest_reg tag(Rd tag for R-type,Rt tag for I-type)

output reg du_en_integer0;
output reg du_en_integer1;
```

```
output du_en_mult;
output du_en_ld_st;

input issueque_full_integer0;
input issueque_full_integer1;
input issueque_full_mult;
input issueque_full_ld_st;

//ROB
//Interfaces with Dispatch Read
output[4:0]                 RS_reg;
//output                        rob_rs_ren;
input[5:0]                  RS_token;
input[31:0]                 Rs_Data_spec;
input       ROB_RS_Data_valid;
output[4:0]                 Rt_reg;
//output                        rob_rt_ren;
input[5:0]                  Rt_token;
input[31:0]                 Rt_Data_spec;
input       ROB_Rt_Data_valid;

//Interfaces with Dispatch Write
output[4:0]                         Dispatch_rd_reg;            //
output[31:0]                ROB_pc;                 //
output                  ROB_dec_inst_no_rdes;       //Intruction without destination register: JMP(does not reach
ROB), BNE, BNQ, SW.
                        //1: instruction does not modify registers
output      ROB_dec_ld_st_sel; //Load Store Selector -> "Operation Code"
                        //0: Load, 1:Store
output                  ROB_branch;                 //
output                  ROB_dispatch_enable;                //

//Interfaces with Retire
input[4:0]    Retire_tag;
input[4:0]    Retire_rd_reg;
input[31:0]   Retire_data;
input[31:0]   Retire_pc;
input       Retire_branch;
input       Retire_branch_taken;
input       Retire_store_ready;     //Store Ready
input       Retire_valid;
input               Retire_flush;

wire[(w_size-1):0] disp_rf_rs_data;
wire[(w_size-1):0] disp_rf_rt_data;
wire[4:0] disp_rd_addr;
wire[4:0] disp_rs_addr;
wire[4:0] disp_rt_addr;
wire[4:0] disp_rdes_addr; //Any type of destination address: rd or rt
wire[25:0] jmp_branch_add;
//wire bj_stall_branch;
wire[15:0] disp_immediate;
wire[5:0] disp_opcode;   //
wire[5:0] disp_funct;
wire[(w_size-1):0] disp_rt_data;
//
wire[4:0] tf_d_out;
```

```
//wire[(w_size-1):0] rst_rf_one_hot_wr_en;
wire disp_ins_type_r_i_sel;
wire[1:0] dec_int_mult_ld_st;
wire dec_zero_sign_ext_sel;
wire dec_inst_no_rdes;
wire tf_empty;
wire tf_full;
wire signed[31:0] Branch_address;

//Operand selector (Rd for R-type, Immediate for I-type
wire disp_log_art_imm_sel;                  //0: logic imm operation (imm = zero extended)
                                            //1: arithmetic imm operation (imm = sign extended)

//Sign Extension Logic
wire[31:0] disp_ZeroExtImm;
wire[31:0] disp_SignExtImm;
//Branch Flag
wire dec_branch;      //Flag to indicate Instruction on Dispatch is a branch

//Instruction fields assigment
assign disp_opcode = ifetch_instruction[31:26];
assign du_shamt = ifetch_instruction[10:6];
assign disp_funct = ifetch_instruction[5:0];
assign disp_immediate = ifetch_instruction[15:0];
assign jmp_branch_add = ifetch_instruction[25:0];
assign disp_rd_addr = ifetch_instruction[15:11];
assign disp_rs_addr = ifetch_instruction[25:21];
assign disp_rt_addr = ifetch_instruction[20:16];

//ROB
assign RS_reg = disp_rs_addr;
assign rob_rs_ren = 1'b1;
assign Rt_reg = disp_rt_addr;
assign rob_rt_ren = 1'b1;

//ROB Interfaces with Dispatch Write
assign Dispatch_rd_reg = disp_rdes_addr;              //
assign ROB_pc = Branch_address;  //ROB_PC is assigned to the branch address: Branch Prediction assumed Not-
Taken
assign ROB_dec_inst_no_rdes = dec_inst_no_rdes;
assign ROB_dec_ld_st_sel = du_ld_or_st;
assign ROB_branch = dec_branch;               //

//
wire disp_tag_valid;
assign disp_tag_valid = (Retire_valid | Retire_branch | Retire_store_ready);

//Register Destination selector
assign disp_rdes_addr = disp_ins_type_r_i_sel ? disp_rt_addr : disp_rd_addr;
assign du_rd_tag = tf_d_out;                                        //assign destination register tag
assign disp_ZeroExtImm = {{16{1'b0}}, disp_immediate};
assign disp_SignExtImm = {{16{disp_immediate[15]}}, disp_immediate};
assign du_rt_data = (disp_ins_type_r_i_sel && !du_ld_or_st && !dec_branch) ? (disp_log_art_imm_sel ?
disp_SignExtImm : disp_ZeroExtImm): disp_rt_data;

assign du_imm_ld_st = dec_zero_sign_ext_sel ? disp_SignExtImm : disp_ZeroExtImm;
```

```
assign ROB_dispatch_enable = ((du_en_mult || du_en_ld_st || du_en_integer0 || du_en_integer1));  //All instruction that require TAG (JMPs don't)

`define SLL_ALU_Ctrl  4'h9
`define SRL_ALU_Ctrl  4'hA

//Dispatch Valid Operand Logic
wire disp_rs_tag_valid;
wire disp_rt_tag_valid;

assign disp_rs_tag_valid = RS_token[5];
assign disp_rt_tag_valid = Rt_token[5];

assign du_rs_tag = RS_token[4:0];
assign du_rt_tag = Rt_token[4:0];

//Logic for rs data
assign du_rs_data = (disp_rs_tag_valid) ? ( (ROB_RS_Data_valid) ? Rs_Data_spec : 0) : disp_rf_rs_data;
assign du_rs_val = ((!disp_rs_tag_valid || ROB_RS_Data_valid) || (du_opcode == `SLL_ALU_Ctrl) || (du_opcode == `SRL_ALU_Ctrl) || ROB_RS_Data_valid);

//Logic for rt data
assign du_rt_val = ((!disp_rt_tag_valid || ROB_Rt_Data_valid) || (disp_ins_type_r_i_sel && !du_ld_or_st) || ROB_Rt_Data_valid);
assign disp_rt_data = (disp_rt_tag_valid) ? ( (ROB_Rt_Data_valid) ? Rt_Data_spec : 0) : disp_rf_rt_data;

//Logic to enable Queues
`define INTEGER_CODE 2'b00
`define MULT_CODE 2'b01
`define LD_ST_CODE 2'b10
`define NOT_USED_CODE 2'b11

//Enable to Mult Queue
assign du_en_mult = (!tf_empty && !issueque_full_mult && (dec_int_mult_ld_st == `MULT_CODE) && !Retire_branch_taken && !ifetch_empty_flag) ? 1'b1 : 1'b0;

//Enable to Load/Store Queue
assign du_en_ld_st = (!tf_empty && !issueque_full_ld_st && (dec_int_mult_ld_st == `LD_ST_CODE) && !Retire_branch_taken && !ifetch_empty_flag) ? 1'b1 : 1'b0;

//Enable to Integer Queues
reg Int_Queue_Sel_reg;

always@(negedge clk or posedge rst)
begin
        if(rst)
                Int_Queue_Sel_reg <= 1;
        //Selector: Int Queue 1 Turn when reset
        else if(du_en_integer0 || du_en_integer1)
   Int_Queue_Sel_reg <= ~Int_Queue_Sel_reg;
  else
   Int_Queue_Sel_reg <= Int_Queue_Sel_reg;
end

always @(*)
begin: INT_QUEUE_ENABLES
        du_en_integer0 = 1'b0;
```

```
  du_en_integer1 = 1'b0;

  if(!tf_empty && (dec_int_mult_ld_st == `INTEGER_CODE) && !ifetch_empty_flag)
  begin
    if(!Int_Queue_Sel_reg)          //Selector: Int Queue 1 Turn
    begin
      if(!issueque_full_integer0 && !Retire_branch_taken)          //Int Queue 1 not full
        du_en_integer0 = 1'b1;
      else if(!issueque_full_integer1 && !Retire_branch_taken)     //Int Queue 1 full but Int Queue 2 not full
        du_en_integer1 = 1'b1;
    end
    else                            //Selector: Int Queue 2 Turn
    begin
      if(!issueque_full_integer1 && !Retire_branch_taken)          //Int Queue 2 not full
        du_en_integer1 = 1'b1;
      else if(!issueque_full_integer0 && !Retire_branch_taken)     //Int Queue 2 full but Int Queue 1 not full
        du_en_integer0 = 1'b1;
    end
  end
end

//IFQ Read Enable Logic
always @(*)
begin: READ_EN_LOGIC
            if(ifetch_empty_flag)
    du_ren = 0;
  else if((issueque_full_mult && (dec_int_mult_ld_st == `MULT_CODE))  || (issueque_full_ld_st &&
(dec_int_mult_ld_st == `LD_ST_CODE)) ||
  (issueque_full_integer0 && issueque_full_integer1 && (dec_int_mult_ld_st == `INTEGER_CODE))) //Queues to
dispatch are full
    du_ren = 0;          //Conditions not to assert rd_en: When issue queue of current instruction in dispatcher is
full.
  else
    du_ren = 1;        //A queue for the instruction is available (not full) or Branch has been resolved
end


//Branch & Jmp Control Logic
wire[31:0] Branch_Imm_extended_sh;
wire[31:0] Jmp_addr;
wire disp_jmp_sel; //1: jmp 0: other

assign Branch_Imm_extended_sh = {{14{jmp_branch_add[15]}}, jmp_branch_add[15:0], 2'b00};
assign Branch_address = ifetch_pc_4 + Branch_Imm_extended_sh;                                    //PC address to
be send to ROB

assign Jmp_addr = {ifetch_pc_4[31:26], jmp_branch_add};//assign Jmp_addr = {ifetch_pc_4[31:28],
jmp_branch_add, 2'b00};

`define JMP_Opcode  6'b00_0010
`define BEQ_Opcode  6'b00_0100
`define BNE_Opcode  6'b00_0101
assign disp_jmp_sel = (disp_opcode == `JMP_Opcode) ? 1'b1 : 1'b0;
assign dec_branch = ((disp_opcode == `BEQ_Opcode) || (disp_opcode == `BNE_Opcode))  ? 1'b1 : 1'b0;

assign du_jmp_br_addr = Retire_branch_taken ? Retire_pc : Jmp_addr;
assign du_jmp_br_val = (disp_jmp_sel ||  Retire_branch_taken) ? 1'b1 : 1'b0;
```

```
Decoder Decoder(
.dec_opcode(disp_opcode),
.dec_funct(disp_funct),
.dec_ins_type_r_i_sel(disp_ins_type_r_i_sel),
.dec_int_mult_ld_st(dec_int_mult_ld_st),
.dec_ld_st_sel(du_ld_or_st),
.dec_zero_sign_ext_sel(dec_zero_sign_ext_sel),
.dec_inst_no_rdes(dec_inst_no_rdes),
.du_opcode(du_opcode),
.disp_log_art_imm_sel(disp_log_art_imm_sel)
);

TAG_FIFO TAG_FIFO(
.rst(rst),
                    //Input:  asynchronous reset
.clk(clk),
                                //Input:  System clock
.tf_d_out(tf_d_out),
.tf_rd_en(ROB_dispatch_enable),              //Read tag to write into RST: asserted
.tf_full(tf_full),                  //Missing from Dispatch Control Logic
.tf_empty(tf_empty),                      //Missing from Dispatch Control Logic
.tf_d_in(Retire_tag),
.tf_wr_en(disp_tag_valid),
.tf_Retire_flush(Retire_flush)
);

RegisterFile RegisterFile(
.rst(rst),                          //Input:  asynchronous reset
.clk(clk),                          //Input:  System clock
.rf_wr_en(Retire_valid),
.rf_Retire_rd_reg(Retire_rd_reg),
.rf_retire_wr_data(Retire_data),
.rf_rs_addr(disp_rs_addr),
.dispatch_rs_data(disp_rf_rs_data),     //rs Operand Register Read to Queues
.rf_rt_addr(disp_rt_addr),
.dispatch_rt_data(disp_rf_rt_data)      //rt Operand Register Read to Queues
//.RegFileAddrBackDoor(RegFileAddrBackDoor),
//.RegFileDataBackDoor(RegFileDataBackDoor)
);
endmodule

module TAG_FIFO(rst, clk, tf_d_out, tf_rd_en, tf_full, tf_empty, tf_d_in, tf_wr_en, tf_Retire_flush);

//Parameters
parameter tag_size = 5;                         //Defines tag line size
parameter tag_fifo_size = 32;                     //Defines tag fifo deep

//Reset and clk inputs
input                   rst;                //Asynchronous reset
input                   clk;                 //System clock

//Interfaces with CDB
input[tag_size-1:0]         tf_d_in;                                    //TAG FIFO Data In from CDB
input               tf_wr_en;                                           //TAG FIFO Write enable from CDB
input                                           tf_Retire_flush;                                    //TAG FIFO Flush due to
missprediction
```

```
//Interfaces with RST
output[tag_size-1:0]        tf_d_out;           //TAG FIFO Data Out to RST
input           tf_rd_en;                   //TAG FIFO Read enable from RST
output              tf_full;                    //TAG FIFO Full Flag to RST
output              tf_empty;                       //TAG FIFO Empty Flag to RST
reg[tag_size-1:0]       tag_fifo[tag_fifo_size-1:0];   //TAG FIFO
integer i;                                          //To count from 0 to tag_fifo_size (32)
//reg[4:0] i;                                   //To count from 0 to tag_fifo_size (32)
reg[5:0]tf_wr_ptr;                                      //TAG FIFO Write Pointer
reg[5:0]tf_rd_ptr;                                          //TAG FIFO Read Pointer

//Combinational assigments----------------------------------
assign tf_empty = (tf_rd_ptr==tf_wr_ptr);
                                                    //TAG FIFO Empty flag: No TAG available to be used.
assign tf_full  = (tf_rd_ptr[4:0]==tf_wr_ptr[4:0])&&(tf_rd_ptr[5]!=tf_wr_ptr[5]);   //TAG FIFO Full flag, all TAGs
available to be used

//Sequential assigments-------------------------------------

//FIFO write process
always@(posedge clk or posedge rst)
begin
        if(rst)
        begin
                for(i=0; i<tag_fifo_size; i=i+1)
                begin
                        tag_fifo[i] <= i;
                end
                tf_wr_ptr <= 6'b100000;             //TAG FIFO Initially Full
        end
        else if(tf_Retire_flush)
        begin
                for(i=0; i<tag_fifo_size; i=i+1)
                begin
                        tag_fifo[i] <= i;
                end
                tf_wr_ptr <= 6'b100000;             //TAG FIFO Initially Full
        end
        else if(tf_wr_en)
        begin
                tag_fifo[tf_wr_ptr[4:0]] <= tf_d_in;
                tf_wr_ptr <= tf_wr_ptr + 1;
        end
end

//FIFO read process
always@(posedge clk or posedge rst)
begin
        if(rst)
        begin
                tf_rd_ptr <= 6'b000000;
        end
        else if(tf_Retire_flush)
        begin
                tf_rd_ptr <= 6'b000000;
        end
```

```
        else if(tf_rd_en)    //Requested by the "Dispatch Unit" only when a instruction with Rdes can be
dispatched.
        begin
                tf_rd_ptr <= tf_rd_ptr + 1;
        end
end

assign tf_d_out = tag_fifo[tf_rd_ptr[4:0]];
endmodule

module RegisterFile(rst, clk, rf_wr_en, rf_retire_wr_data, rf_Retire_rd_reg, rf_rs_addr, dispatch_rs_data, rf_rt_addr,
dispatch_rt_data);
            //RegFileAddrBackDoor, RegFileDataBackDoor);

//Parameters
parameter rf_line_size = 32;    //Defines the Register File line width
parameter rf_bits2addr = 5;     //Number of bits to address the Register File
parameter rf_size = 32;         //Defines the Register File deep: 32-word
//parameter rf_size  = (1<<rf_bits2addr); //32-word

//Reset and clk inputs
input                   rst;                //Asynchronous reset
input                   clk;                //System clock
//Write Port: Interface with the RST & CDB
input[4:0]  rf_Retire_rd_reg;
input       rf_wr_en;                       //Write Enable from RST
input[(rf_line_size-1):0]    rf_retire_wr_data;     //Write Data from CDB
//Read Port 1: Interface with
input [(rf_bits2addr-1):0]   rf_rs_addr;         //Read Register Address 1
output[(rf_line_size-1):0]   dispatch_rs_data;  //Read Data from Mux to Queue
//Read Port 2: Interface with
input [(rf_bits2addr-1):0]   rf_rt_addr;         //Read Register Address 2
output[(rf_line_size-1):0]   dispatch_rt_data;  //Read Data from Mux to Queue

//Backdoor: Added for Instrumentation
//input[(rf_bits2addr-1):0]    RegFileAddrBackDoor;
//output[(rf_line_size-1):0]            RegFileDataBackDoor;

reg [(rf_line_size-1):0] RegFileRAM [(rf_size-1):0]; //32-word x 32-bit

integer i;                      //To count from 0 to 32
//reg [rf_bits2addr:0] i;       //To count from 0 to 32
integer N;          //Index to sweep the 32 conditions of the 32-bit One-Hot Decoder

wire[(rf_line_size-1):0] rf_rs_data;       //Read rs Data from Register File
wire[(rf_line_size-1):0] rf_rt_data;       //Read rt Data from Register File

//Write Process
always@(posedge clk or posedge rst)
begin
  if(rst)
  begin
    for(i=0; i<rf_size; i=i+1)
      RegFileRAM[i] <= i;  //Initialize with i for Bubble sort, 0 for other
  end
  else if(rf_wr_en)
      RegFileRAM[rf_Retire_rd_reg] <= rf_retire_wr_data;
```

```
  else
     RegFileRAM[rf_Retire_rd_reg] <= RegFileRAM[rf_Retire_rd_reg];
end

//Read Process: Reading done combinationally
//Real MIPS implementation considers Register 0 hardwired to 0
assign dispatch_rs_data = (rf_rs_addr) ? RegFileRAM[rf_rs_addr] : 0;
assign dispatch_rt_data = (rf_rt_addr) ? RegFileRAM[rf_rt_addr] : 0;

//Backdoor: Added for Instrumentation
//assign RegFileDataBackDoor = RegFileRAM[RegFileAddrBackDoor];
endmodule

module Decoder(dec_opcode, dec_funct, dec_ins_type_r_i_sel, dec_int_mult_ld_st,  dec_ld_st_sel,
dec_zero_sign_ext_sel, dec_inst_no_rdes, du_opcode, disp_log_art_imm_sel);
//Instruction Type decoding: Register or Immediate

input[5:0] dec_opcode;                          //Instruction Opcode
input[5:0] dec_funct;                          //Instruction Function field
output dec_ins_type_r_i_sel;                     //Selector to Mux that selects rd or rt for renaming at RST.
                                    //0: Register to Register, 1:Immediate
output reg[1:0] dec_int_mult_ld_st;               //Queue Selector.
                                    //00: Integer Queue, 01:Mult Queue
                                    //10: Load Store Queue 11: No Issue
output reg dec_ld_st_sel;          //Load Store Selector -> "Operation Code"
                                    //0: Load, 1:Store
output reg dec_inst_no_rdes;//Without destination register: JMP, BNE, BNQ, SW
                        //meaning: instruction does not modify registers


output dec_zero_sign_ext_sel;                    //0: ZeroExt for Logical Int Imm, 1: SignExt for Arithmetic Int Imm
output reg[3:0] du_opcode;
output reg disp_log_art_imm_sel;                  //0: logic imm operation (imm = zero extended)
                                    //1: arithmetic imm operation (imm = sign extended)

`define ADD_FUNC  6'h20
`define ADDU_FUNC  6'h21
`define AND_FUNC  6'h24
`define NOR_FUNC  6'h27
`define OR_FUNC  6'h25
`define SLT_FUNC  6'h2A
`define SLTU_FUNC  6'h2B
`define SLL_FUNC  6'h00
`define SRL_FUNC  6'h02
`define SUB_FUNC  6'h22
`define ADDI_Opcode  6'h08
`define ADDIU_Opcode  6'h09
`define ANDI_Opcode  6'h0C
`define BEQ_Opcode  6'h04
`define BNE_Opcode  6'h05
`define JMP_Opcode  6'h02
`define ORI_Opcode  6'h0D
`define SLTI_Opcode  6'h0A
`define R_Format_Opcode  6'h00
`define LW_Opcode  6'h23
`define SW_Opcode  6'h2B
`define Mult_Funct  6'h18
```

```
`define ADD_ALU_Ctrl  4'h0
`define ADDU_ALU_Ctrl  4'h1
`define AND_ALU_Ctrl  4'h2
`define BEQ_ALU_Ctrl  4'h3
`define BNE_ALU_Ctrl  4'h4
`define NOR_ALU_Ctrl  4'h5
`define OR_ALU_Ctrl  4'h6
`define SLT_ALU_Ctrl  4'h7
`define SLTU_ALU_Ctrl  4'h8
`define SLL_ALU_Ctrl  4'h9
`define SRL_ALU_Ctrl  4'hA
`define SUB_ALU_Ctrl  4'hB
`define NOP_ALU_Ctrl  4'hF

assign dec_ins_type_r_i_sel = ((dec_opcode == `R_Format_Opcode))  ? 1'b0 : 1'b1;
assign dec_zero_sign_ext_sel = ((dec_opcode == `ANDI_Opcode) || (dec_opcode == `ORI_Opcode))  ? 1'b0 :
1'b1;

always @(*)
begin: DECODER_COMB
   dec_inst_no_rdes = 1'b0;
   dec_ld_st_sel = 1'b0;
  case(dec_opcode)
                        `R_Format_Opcode:
    begin
      if(dec_funct == `Mult_Funct)
        dec_int_mult_ld_st = `MULT_CODE;
      else
        dec_int_mult_ld_st = `INTEGER_CODE;
    end

    `LW_Opcode:
    begin
      dec_int_mult_ld_st = `LD_ST_CODE;
      dec_ld_st_sel = 1'b0;
    end

    `SW_Opcode:
    begin
      dec_int_mult_ld_st = `LD_ST_CODE;
      dec_ld_st_sel = 1'b1;
      dec_inst_no_rdes = 1'b1;  //Instruction does not modify registers
    end

    `BEQ_Opcode,`BNE_Opcode:
    begin
      dec_inst_no_rdes = 1'b1; //Instruction does not modify registers
      dec_int_mult_ld_st = 2'b00;
    end

    `JMP_Opcode:
    begin
      dec_inst_no_rdes = 1'b1;  //Instruction does not modify registers
      dec_int_mult_ld_st = 2'b11;
    end

    //Integer Immediates
```

```
                        default: dec_int_mult_ld_st = 2'b00;
  endcase
end
always @(*)
begin: DECODER_ALU_CTRL
  disp_log_art_imm_sel = 1;
  case(dec_opcode)
                        `R_Format_Opcode:
    begin
      case(dec_funct)
        `ADD_FUNC:  du_opcode = `ADD_ALU_Ctrl;
        `ADDU_FUNC: du_opcode = `ADDU_ALU_Ctrl;
        `AND_FUNC:  du_opcode = `AND_ALU_Ctrl;
        `NOR_FUNC:  du_opcode = `NOR_ALU_Ctrl;
        `OR_FUNC:   du_opcode = `OR_ALU_Ctrl;
        `SLT_FUNC:  du_opcode = `SLT_ALU_Ctrl;
        `SLTU_FUNC: du_opcode = `SLTU_ALU_Ctrl;
        `SLL_FUNC:  du_opcode = `SLL_ALU_Ctrl;
        `SRL_FUNC:  du_opcode = `SRL_ALU_Ctrl;
        `SUB_FUNC:  du_opcode = `SUB_ALU_Ctrl;
        default: du_opcode = `NOP_ALU_Ctrl;
      endcase
    end
    `ADDI_Opcode:   du_opcode = `ADD_ALU_Ctrl;
    `ADDIU_Opcode:  du_opcode = `ADDU_ALU_Ctrl;
    `ANDI_Opcode:
      begin
        du_opcode = `AND_ALU_Ctrl;
        disp_log_art_imm_sel = 0;
      end
    `BEQ_Opcode:    du_opcode = `BEQ_ALU_Ctrl;
    `BNE_Opcode:    du_opcode = `BNE_ALU_Ctrl;
    `ORI_Opcode:
      begin
        du_opcode = `OR_ALU_Ctrl;
        disp_log_art_imm_sel = 0;
      end
    `SLTI_Opcode:   du_opcode = `SLT_ALU_Ctrl;
                        default: du_opcode = `NOP_ALU_Ctrl;
  endcase
end
endmodule
```

Colas de ejecucion - Verilog Netlist

```
module IntegerQueue(rst, clk, dispatch_rs_data_val, dispatch_rs_tag, dispatch_rs_data,
dispatch_rt_data_val, dispatch_rt_tag, dispatch_rt_data, dispatch_opcode, dispatch_shfamt,
dispatch_rd_tag, dispatch_enable, issueque_full, CDB_valid, CDB_tag, CDB_data, retire_flush,
issueque_ready, issueque_rs_data, issueque_rt_data, issueque_rd_tag, issueque_opcode,
issueque_shfamt,issueblk_issue);

//Parameters
parameter w_size = 32;                  //Defines word width
parameter queue_size = 4;
```

```
//Reset and clk inputs
input                rst;                //Asynchronous reset
input                clk;                //System clock
 //Interfaces with Dispatch unit
input dispatch_rs_data_val;
input[4:0] dispatch_rs_tag;
input signed[31:0] dispatch_rs_data;
input dispatch_rt_data_val;
input[4:0] dispatch_rt_tag;
input[31:0] dispatch_rt_data;
input[3:0] dispatch_opcode;
input[4:0] dispatch_shfamt;
input[4:0] dispatch_rd_tag;
input dispatch_enable;    //Write Enable to Queue
output issueque_full;

//Interfaces with CDB
input CDB_valid;
input[4:0] CDB_tag;
input signed[31:0] CDB_data;

//Interfaces with ROB
input retire_flush;

//Interfaces with Functional Unit (ALU)
output reg[31:0] issueque_rs_data;
output reg[31:0] issueque_rt_data;
output reg[3:0] issueque_opcode;
output reg[4:0] issueque_shfamt;
output reg[4:0] issueque_rd_tag;

//Interfaces with Issue Unit
output reg issueque_ready;
input issueblk_issue;

//Integer queue FIFO structure
reg[3:0]          alu_ctrl[queue_size-1:0];
reg[4:0]          rdes_tag[queue_size-1:0];
reg[4:0]    rs_tag[queue_size-1:0];
reg[31:0]   rs_data[queue_size-1:0];
reg             rs_data_valid[queue_size-1:0];
reg[4:0]    rt_tag[queue_size-1:0];
reg[31:0]   rt_data[queue_size-1:0];
reg             rt_data_valid[queue_size-1:0];
reg[4:0]          shamt[queue_size-1:0];
reg             ocuppy[queue_size-1:0];       //Queue occupy

integer N;                                          //To count from 0 to queue_size (3)

reg[2:0]   int_write_ptr; //Write to FIFO wihout considering shift
```

```
reg[3:0]   int_read_ptr;                    //One-hot encoded red pointer
reg                queue_shift; //Shift flag, 1: A instruction was issued,
                                            //queues need to shift 0: No shift required
wire[2:0]  int_fifo_write_ptr; //Write pointer to FIFO considering
                                              //shift operations
//Combinational assigments------------------------------assign issueque_full  =  (ocuppy[0] &&
ocuppy[1] && ocuppy[2] && ocuppy[3] && !issueblk_issue);

//Ready and rd_ptr (one-hot encoded) generation
always @(rt_data_valid[0], rt_data_valid[1], rt_data_valid[2], rt_data_valid[3], rs_data_valid[0],
rs_data_valid[1], rs_data_valid[2], rs_data_valid[3])
begin
        if(rs_data_valid[0] && rt_data_valid[0])
        begin
                issueque_ready = 1;
                int_read_ptr = 4'b0001;
        end
        else if(rs_data_valid[1] && rt_data_valid[1])
        begin
                issueque_ready = 1;
                int_read_ptr = 4'b0010;
        end
        else if(rs_data_valid[2] && rt_data_valid[2])
        begin
                issueque_ready = 1;
                int_read_ptr = 4'b0100;
        end
        else if(rs_data_valid[3] && rt_data_valid[3])
        begin
                issueque_ready = 1;
                int_read_ptr = 4'b1000;
        end
        else
        begin
                issueque_ready = 0;
                int_read_ptr = 0;
        end
end
//Update Queue's Fields indication
reg[3:0]                rs_match;
reg[3:0]                rt_match;
always @(CDB_tag, CDB_valid, rs_tag[0], rs_tag[1], rs_tag[2], rs_tag[3], ocuppy[0], ocuppy[1],
ocuppy[2], ocuppy[3],
                rs_data_valid[0], rs_data_valid[1], rs_data_valid[2], rs_data_valid[3])
begin: RS_MATCH         //2-to-4 One-Hot Binary Decoder
        for(N=0; N < queue_size /*4*/; N=N+1)
        begin
   if((CDB_tag  == rs_tag[N])  && CDB_valid && ocuppy[N] && !rs_data_valid[N])
     rs_match[N] = 1;
   else
     rs_match[N] = 0;
```

```
  end
end
always @(CDB_tag, CDB_valid, rt_tag[0], rt_tag[1], rt_tag[2], rt_tag[3], ocuppy[0], ocuppy[1],
ocuppy[2], ocuppy[3],
                rt_data_valid[0], rt_data_valid[1], rt_data_valid[2], rt_data_valid[3])
begin: RT_MATCH         //2-to-4 One-Hot Binary Decoder
            for(N=0; N < queue_size /*4*/; N=N+1)
            begin
    if((CDB_tag  == rt_tag[N])  && CDB_valid && ocuppy[N] && !rt_data_valid[N])
      rt_match[N] = 1;
    else
      rt_match[N] = 0;
  end
end

//FIFO write process - Write pointer generation
always@(posedge clk or posedge rst)
begin
            if(rst)
                        int_write_ptr <= 0;                                                                 //Clear
write pointer
            else if(retire_flush)
                        int_write_ptr <= 0;                                                                 //Clear
write pointer
            else if(dispatch_enable && queue_shift)
                        int_write_ptr <= int_write_ptr; //If Inst was issued and dispatched, write pointer
remains as dispatched inst will be stored on (wr_ptr - 1)
            else if(dispatch_enable && (int_write_ptr == 4))
                        int_write_ptr <= int_write_ptr; //Write pointer reaches Top, so it does not increment
any more
            else if(dispatch_enable)
                                    int_write_ptr <= int_write_ptr + 1; //Dispactched instruction without Issued
Instruction, increment write pointer
            else if(queue_shift)
                        int_write_ptr <= int_write_ptr - 1; //Issued instruction without Dispactched Instruction,
decrement write pointer due to shift
end

assign int_fifo_write_ptr = queue_shift ? (int_write_ptr - 1) : int_write_ptr;

//FIFO write process - Queue Shift Signal generation

reg[3:0] queue_update_shift; //Flag enable queue updates due to shifts
wire[3:0] queue_update_shift_cdb; //Flag to indicate Rs or Rt were updated from CDB during shifts

always@(*) begin
            if(issueblk_issue) begin
                        queue_update_shift = 0;
                        queue_shift = 1;
                        case(int_read_ptr)
                                    4'b0001:
```

```
                        begin
                                queue_update_shift[0] = 1;
                                queue_update_shift[1] = 1;
                                queue_update_shift[2] = 1;
                        end
                        4'b0010:
                        begin
                                queue_update_shift[1] = 1;
                                queue_update_shift[2] = 1;
                        end
                        4'b0100:
                        begin
                                queue_update_shift[2] = 1;
                        end
                        4'b1000:
                        begin
                                queue_update_shift[3] = 1;
                        end
                        default queue_update_shift = 0;
                endcase
        end
        else
        begin
                queue_update_shift = 0;
                queue_shift = 0;
        end
end
//FIFO write process - FIFO update
assign queue_update_shift_cdb = {queue_update_shift[2:0], 1'b0};
always@(posedge clk or posedge rst) begin
        if(rst) begin
                for(N=0; N < queue_size /*4*/; N=N+1)
        //Clear FIFO
                begin
                        alu_ctrl[N] <= 0;
                        rdes_tag[N] <= 0;
                        rs_tag[N] <= 0;
                        rs_data[N] <= 0;
                        rs_data_valid[N] <= 0;
                        rt_tag[N] <= 0;
                        rt_data[N] <= 0;
                        rt_data_valid[N] <= 0;
                        shamt[N] <= 0;
                        ocuppy[N] <= 0;
                end
        end
        else if(retire_flush)
        begin
                for(N=0; N < queue_size /*4*/; N=N+1)
        //Clear FIFO
                begin
```

```
                alu_ctrl[N] <= 0;
                rdes_tag[N] <= 0;
                rs_tag[N] <= 0;
                rs_data[N] <= 0;
                rs_data_valid[N] <= 0;
                rt_tag[N] <= 0;
                rt_data[N] <= 0;
                rt_data_valid[N] <= 0;
                shamt[N] <= 0;
                ocuppy[N] <= 0;
        end
end
else
begin
                        //Add instruction to queue
        if(dispatch_enable)
        begin
                alu_ctrl[int_fifo_write_ptr[1:0]] <= dispatch_opcode;
                rdes_tag[int_fifo_write_ptr[1:0]] <= dispatch_rd_tag;
                rs_tag[int_fifo_write_ptr[1:0]] <= dispatch_rs_tag;
                rs_data[int_fifo_write_ptr[1:0]] <= dispatch_rs_data;
rs_data_valid[int_fifo_write_ptr[1:0]] <= dispatch_rs_data_val;
                rt_tag[int_fifo_write_ptr[1:0]] <= dispatch_rt_tag;
                rt_data[int_fifo_write_ptr[1:0]] <= dispatch_rt_data;
rt_data_valid[int_fifo_write_ptr[1:0]] <= dispatch_rt_data_val;
                shamt[int_fifo_write_ptr[1:0]] <= dispatch_shfamt;
                ocuppy[int_fifo_write_ptr[1:0]] <= 1;
        end
        if(queue_shift)
        begin
                if(queue_update_shift == 4'b1000)
                begin
                        rt_data_valid[3] <= 0;
                        rs_data_valid[3] <= 0;
                        ocuppy[3] <= 0;
                end
                else
                begin
                        for(N=0; N < (queue_size-1) /*3*/; N=N+1)
        //Update queues as per shift
                        begin
                        if((queue_update_shift[N]) && (ocuppy[N+1]))
                                begin
                                        alu_ctrl[N] <= alu_ctrl[N+1];
                                        rdes_tag[N] <= rdes_tag[N+1];
                                        rs_tag[N] <= rs_tag[N+1];
                                        if(rs_match[N+1])
                                        begin
                                                rs_data[N] <= CDB_data;
                                                rs_data_valid[N] <= 1;
                                        end
```

```
                                        else
                                        begin
                                                rs_data[N] <= rs_data[N+1];
                                    rs_data_valid[N] <= rs_data_valid[N+1];
                                        end

                                        rt_tag[N] <= rt_tag[N+1];

                                        if(rt_match[N+1])
                                        begin
                                                rt_data[N] <= CDB_data;
                                                rt_data_valid[N] <= 1;
                                        end
                                        else
                                        begin
                                                rt_data[N] <= rt_data[N+1];
                                    rt_data_valid[N] <= rt_data_valid[N+1];
                                        end
                                        shamt[N] <= shamt[N+1];
                                        ocuppy[N] <= ocuppy[N+1];
                                        if(N == 2)
                                        begin
                                                rt_data_valid[N+1] <= 0;
                                                rs_data_valid[N+1] <= 0;
                                                ocuppy[N+1] <= 0;
                                        end
                                    end

                else if((queue_update_shift[N]) && !dispatch_enable) begin
                                        rt_data_valid[N] <= 0;
                                        rs_data_valid[N] <= 0;
                                        ocuppy[N] <= 0;
                                    end
                            end
                        end
                end
                for(N=0; N < queue_size /*4*/; N=N+1) begin
                  if(rs_match[N] && !(queue_update_shift_cdb[N])) begin
                        rs_data[N] <= CDB_data;
                        rs_data_valid[N] <= 1;
                  end
                  if(rt_match[N] && !(queue_update_shift_cdb[N])) begin
                        rt_data[N] <= CDB_data;
                        rt_data_valid[N] <= 1;
                  end
                end
        end
end
//FIFO read process
```

```
always@(rs_data[0], rs_data[1], rs_data[2], rs_data[3], rt_data[0], rt_data[1], rt_data[2], rt_data[3],
alu_ctrl[0], alu_ctrl[1], alu_ctrl[2], alu_ctrl[3], shamt[0], shamt[1], shamt[2], shamt[3], rdes_tag[0],
rdes_tag[1], rdes_tag[2], rdes_tag[3],int_read_ptr)
begin
        //Interfaces with Functional Unit (ALU)
        issueque_rs_data = 0;
        issueque_rt_data = 0;
        issueque_opcode = 0;
        issueque_shfamt = 0;
        //Interfaces with Issue Unit
        issueque_rd_tag = 0;
        for(N=0; N < queue_size /*4*/; N=N+1) begin
          if(int_read_ptr[N]) begin
                //Interfaces with Functional Unit (ALU)
                issueque_rs_data = rs_data[N];
                issueque_rt_data = rt_data[N];
                issueque_opcode = alu_ctrl[N];
                issueque_shfamt = shamt[N];
                //Interfaces with Issue Unit
                issueque_rd_tag = rdes_tag[N];
          end
        end
end
endmodule
```

Issue Unit Verilog Netlist

```
module IssueUnit(clk, rst, Ready_int0, Issue_int0, int0_result, int0_tag_out, int0_branch, int0_branch_taken,
Ready_int1, Issue_int1, int1_result, int1_tag_out, int1_branch, int1_branch_taken, Ready_mult, Issue_mult,
mult_result, mult_tag_out, Ready_ld_st, Issue_ld_st, ld_data,ld_tag_out, CDB_data, CDB_tag, CDB_valid,
CDB_branch, CDB_branch_taken, issue_retire_flush);

//Parameters
parameter w_size = 32;                  //Defines word width

//Reset and clk inputs
input           rst;            //Asynchronous reset
input           clk;            //System clock
//Interfaces with Integer Queue and ALU 1
input Ready_int0;
output reg Issue_int0;
input[w_size-1:0] int0_result;
input[4:0] int0_tag_out;
input int0_branch;
input int0_branch_taken;
//Interfaces with Integer Queue and ALU 2
input Ready_int1;
output reg Issue_int1;
input[w_size-1:0] int1_result;
input[4:0] int1_tag_out;
input int1_branch;
input int1_branch_taken;
```

```
//Interfaces with Mult Queue and Multiplier
input Ready_mult;
output reg Issue_mult;
input[w_size-1:0] mult_result;
input[4:0] mult_tag_out;

//Interfaces with Data Cache
input Ready_ld_st;
output reg Issue_ld_st;
input[w_size-1:0] ld_data;
input[4:0] ld_tag_out;

//Interfaces with CDB
output reg[31:0] CDB_data;
output reg[4:0] CDB_tag;
output reg CDB_valid;
output reg CDB_branch;
output reg CDB_branch_taken;
//Interfaces with retire Bus
input issue_retire_flush;
integer N;                                          //To count from 0 to LRU table size (3)
//LRU Issue priority Logic-----------------------------
//Issue candidate logic

wire[3:0] IU_ready_bus;
reg[3:0] IU_Issue_candidate_bus;

reg[3:0] LRU_Table[3:0];          //LRU table, LRU_Table[0] is least used
                                                    //and LRU_Table[3] is the most RU

assign IU_ready_bus = {Ready_int1, Ready_int0, Ready_mult, Ready_ld_st};
always@(LRU_Table[0],LRU_Table[1],LRU_Table[2],LRU_Table[3],IU_ready_bus[3],IU_ready_bus[2],IU_ready_bus
[1],IU_ready_bus[0])
begin
        if(LRU_Table[0] & IU_ready_bus)
                IU_Issue_candidate_bus = LRU_Table[0] & IU_ready_bus[3:0];
        else if(LRU_Table[1] & IU_ready_bus)
                IU_Issue_candidate_bus = LRU_Table[1] & IU_ready_bus[3:0];
        else if(LRU_Table[2] & IU_ready_bus)
                IU_Issue_candidate_bus = LRU_Table[2] & IU_ready_bus[3:0];
        else if(LRU_Table[3] & IU_ready_bus)
                IU_Issue_candidate_bus = LRU_Table[3] & IU_ready_bus[3:0];
        else
                IU_Issue_candidate_bus = 4'b0000;
end
//LRU Issue priority table
always@(posedge clk or posedge rst)
begin
        if(rst)
        begin
                for(N=0; N < 4; N=N+1)                                          //Clear FIFO
                begin
                        LRU_Table[N] <= 1'b1 << N;
                end
        end
        else if(Issue_int0 || Issue_int1 || Issue_mult || Issue_ld_st)
```

```
	begin
		if(LRU_Table[0] & IU_ready_bus)
		begin
			LRU_Table[0] <= LRU_Table[1];
			LRU_Table[1] <= LRU_Table[2];
			LRU_Table[2] <= LRU_Table[3];
			LRU_Table[3] <= LRU_Table[0];
		end
		else if(LRU_Table[1] & IU_ready_bus)
		begin
			LRU_Table[0] <= LRU_Table[0];
			LRU_Table[1] <= LRU_Table[2];
			LRU_Table[2] <= LRU_Table[3];
			LRU_Table[3] <= LRU_Table[1];
		end
		else if(LRU_Table[2] & IU_ready_bus)
		begin
			LRU_Table[0] <= LRU_Table[0];
			LRU_Table[1] <= LRU_Table[1];
			LRU_Table[2] <= LRU_Table[3];
			LRU_Table[3] <= LRU_Table[2];
		end
		else
		begin
			LRU_Table[0] <= LRU_Table[0];
			LRU_Table[1] <= LRU_Table[1];
			LRU_Table[2] <= LRU_Table[2];
			LRU_Table[3] <= LRU_Table[3];
		end
	end
	else
	begin
		LRU_Table[0] <= LRU_Table[0];
		LRU_Table[1] <= LRU_Table[1];
		LRU_Table[2] <= LRU_Table[2];
		LRU_Table[3] <= LRU_Table[3];
	end
end

//Instruction Issue trigger----------------------------
//Mult status register

reg[2:0] IU_mult_status;
wire IU_mult_done;

always@(posedge clk or posedge rst)
begin
	if(rst)
		IU_mult_status <= 0;
	else
		IU_mult_status <= {Issue_mult, IU_mult_status[2:1]};
end

assign IU_mult_done = IU_mult_status[0];

//Instruction issue
//IU_Issue_candidate_bus[0] > Memory Unit
```

```
//IU_Issue_candidate_bus[1] > Mult Unit
//IU_Issue_candidate_bus[2] > Integer Unit 1
//IU_Issue_candidate_bus[3] > Integer Unit 2

wire[3:0] IU_CDB_data_out_sel;
assign IU_CDB_data_out_sel = {Issue_int1, Issue_int0, IU_mult_done, Issue_ld_st};

always@(*)
begin
	Issue_ld_st = 0;
	Issue_mult = 0;
	Issue_int0 = 0;
	Issue_int1 = 0;

	case(IU_Issue_candidate_bus)
		4'b0001:
		begin
			if(!IU_mult_done)
				Issue_ld_st = 1;
		end
		4'b0010:
		begin
			Issue_mult = 1;
		end
		4'b0100:
		begin
			if(!IU_mult_done)
				Issue_int0 = 1;
		end
		4'b1000:
		begin
			if(!IU_mult_done)
				Issue_int1 = 1;
		end
		default:
		begin
			Issue_ld_st = 0;
			Issue_mult = 0;
			Issue_int0 = 0;
			Issue_int1 = 0;
		end
	endcase
end
//CDB data output mux-----------------------------
reg[w_size-1:0] IU_cdb_data_mux;
reg[4:0] IU_cdb_tag_mux;
reg IU_cdb_tag_valid_mux;
reg IU_cdb_branch_mux;
reg IU_cdb_branch_taken_mux;
always@(*)
begin
	IU_cdb_branch_mux = 0;
	IU_cdb_branch_taken_mux = 0;
	IU_cdb_tag_valid_mux = 0;
	IU_cdb_tag_mux = 0;
	IU_cdb_data_mux = 0;
	case(IU_CDB_data_out_sel)
```

```
                4'b0001:
                begin
                        IU_cdb_data_mux = ld_data;
                        IU_cdb_tag_mux = ld_tag_out;
                        IU_cdb_tag_valid_mux = 1;
                end
                4'b0010:
                begin
                        IU_cdb_data_mux = mult_result;
                        IU_cdb_tag_mux = mult_tag_out;
                        IU_cdb_tag_valid_mux = 1;
                end
                4'b0100:
                begin
                        IU_cdb_data_mux = int0_result;
                        IU_cdb_tag_mux = int0_tag_out;
                        IU_cdb_tag_valid_mux = 1;
                        IU_cdb_branch_mux = int0_branch;
                        IU_cdb_branch_taken_mux = int0_branch_taken;
                end
                4'b1000:
                begin
                        IU_cdb_data_mux = int1_result;
                        IU_cdb_tag_mux = int1_tag_out;
                        IU_cdb_tag_valid_mux = 1;
                        IU_cdb_branch_mux = int1_branch;
                        IU_cdb_branch_taken_mux = int1_branch_taken;
                end
                default:
                begin
                        IU_cdb_branch_mux = 0;
                        IU_cdb_branch_taken_mux = 0;
                        IU_cdb_tag_valid_mux = 0;
                        IU_cdb_tag_mux = 0;
                        IU_cdb_data_mux = 0;
                end
        endcase
end

//CDB data output reg-----------------------------
always@(posedge clk or posedge rst)
begin
        if(rst)
        begin
                CDB_data <= 0;
                CDB_tag <= 0;
                CDB_valid <= 0;
                CDB_branch <= 0;
                CDB_branch_taken <= 0;
        end
        else  if(issue_retire_flush)
        begin
                CDB_data <= 0;
                CDB_tag <= 0;
                CDB_valid <= 0;
                CDB_branch <= 0;
                CDB_branch_taken <= 0;
```

```
        end
        else
        begin
                CDB_data <= IU_cdb_data_mux;
                CDB_tag <= IU_cdb_tag_mux;
                CDB_valid <= IU_cdb_tag_valid_mux;
                CDB_branch <= IU_cdb_branch_mux;
                CDB_branch_taken <= IU_cdb_branch_taken_mux;
        end
end
endmodule
```

ROB Verilog Netlist

```
module ROB(clk, rst,
        //Dispatch Write
                Dispatch_Rd_tag, Dispatch_Rd_reg, Dispatch_pc, ROB_dec_inst_no_rdes, ROB_dec_ld_st_sel,
                dispatch_branch, dispatch_enable,
         //Dispatch Read
                RS_reg, /*disp_rs_ren,*/ RS_token, Rs_Data_spec, ROB_RS_Data_valid,
                Rt_reg, /*disp_rt_ren,*/ Rt_token, Rt_Data_spec, ROB_Rt_Data_valid,
         //Issue
         CDB_tag, CDB_data, CDB_valid, /*cdb_branch,*/ CDB_branch_taken,
         //Retire Bus
         Retire_rd_tag, Retire_rd_reg, Retire_data, Retire_pc,
                Retire_branch, Retire_branch_taken, Retire_store_ready,
                Retire_valid, Retire_flush);

//Reset and clk inputs
input                   rst;  //Asynchronous reset
input                   clk;   //System clock

//Interfaces with Dispatch Write
input[4:0]                      Dispatch_Rd_tag;                        //
input[4:0]                      Dispatch_Rd_reg;                        //
input[31:0]                     Dispatch_pc;                    //
input                   ROB_dec_inst_no_rdes;       //Intruction without destination register: JMP(does not reach
ROB), BNE, BNQ, SW.
                        //1: instruction does not modify registers
input                   ROB_dec_ld_st_sel;//Load Store Selector -> "Operation Code"
                        //0: Load, 1:Store
input                   dispatch_branch;                //
input                   dispatch_enable;                 //

//Interfaces with Dispatch Read
input[4:0]                      RS_reg;
//input                 disp_rs_ren; // reads are combinationals
output[5:0]                     RS_token;
output[31:0]                    Rs_Data_spec;
output          ROB_RS_Data_valid;
input[4:0]                      Rt_reg;
//input                 disp_rt_ren; // reads are combinationals
output[5:0]                     Rt_token;
output[31:0]                    Rt_Data_spec;
output          ROB_Rt_Data_valid;
```

```
//Interfaces with CDB
input[4:0]                  CDB_tag;
input[31:0]                 CDB_data;
input                   CDB_valid;
//input                 cdb_branch;
input                   CDB_branch_taken;



//Outputs from ROB
output[4:0]     Retire_rd_tag;
output[4:0]     Retire_rd_reg;
output[31:0]    Retire_data;
output[31:0]    Retire_pc;
output          Retire_branch;
output          Retire_branch_taken;
output          Retire_store_ready;       //Store Ready
output          Retire_valid;
output                      Retire_flush;

wire rft_Retire_branch;

//reg file temp outputs
wire            retire_out_inst_no_rdes;
wire            rob_retire_valid;

wire            retire_ld_st_sel;
assign Retire_store_ready = retire_ld_st_sel & rob_retire_valid;

wire            retire_spec_data_valid;
wire            retire_out_valid;

assign rob_retire_valid = ( retire_spec_data_valid & retire_out_valid );

wire            Retire_rd_tag_valid;
assign Retire_rd_tag_valid = (~retire_out_inst_no_rdes & rob_retire_valid);

assign Retire_flush = (Retire_branch && Retire_branch_taken); //Engine Predictor: Not Taken

wire            rst_wr_en;
assign rst_wr_en = dispatch_enable & ~ROB_dec_inst_no_rdes; //"1" only when instruction with rdes is
dispatched.

assign Retire_valid = (rob_retire_valid & ~retire_out_inst_no_rdes );

assign Retire_branch = rft_Retire_branch && retire_spec_data_valid;
//Register Status Table (RST) instance
RST RST(
.rst(rst),                                                          //Input:  asynchronous reset
.clk(clk),                                                                  //Input:  System clock
.rst_rd_token({1'b1,Dispatch_Rd_tag}),
.rst_rdes_add(Dispatch_Rd_reg),
.rst_rd_wr_en(rst_wr_en),               //All dispatched with "rdes" instructions will have a TAG
.rst_rs_add(RS_reg),
.rst_rs_token(RS_token),
.rst_rt_token(Rt_token),
```

```
.rst_rt_add(Rt_reg),
.rst_Retire_rd_tag(Retire_rd_tag),
.rst_Retire_rd_tag_valid(Retire_rd_tag_valid),
.rst_retire_flush(Retire_flush)
);

// Order Queue
ORDER_QUEUE ORDER_QUEUE(
.rst(rst),
.clk(clk),
.oq_tag_out(Retire_rd_tag),
.oq_tag_in(Dispatch_Rd_tag),
.oq_disp_en(dispatch_enable),    //All dispatched instructions will be recorded into Order Queue
.oq_retire_valid(rob_retire_valid),
.oq_flush(Retire_flush)
);

RegFileTemp RegFileTemp (
.rst(rst),
.clk(clk),
 //Inputs from Disptacher: New Entry
.rft_dispatch_enable(dispatch_enable),
.rft_Dispatch_Rd_tag(Dispatch_Rd_tag),
.rft_rdes_add(Dispatch_Rd_reg),
.rft_pc(Dispatch_pc),
//Inputs Instruction Type from Dispatcher
.rft_inst_no_rdes(ROB_dec_inst_no_rdes),
.rft_ld_st_sel(ROB_dec_ld_st_sel),
.rft_branch(dispatch_branch),
//Inputs from CDB: Update Entry
.rft_CDB_tag(CDB_tag),
.rft_cdb_spec_data(CDB_data),
.rft_CDB_tag_valid(CDB_valid),
//.rft_cdb_branch(cdb_branch),
.rft_CDB_branch_taken(CDB_branch_taken),
//Input Consult
.rft_rt_tag(Rt_token[4:0]),
.rft_rs_tag(RS_token[4:0]),
//Output Consult
.rft_rt_spec_data(Rt_Data_spec),
.rft_rt_spec_data_valid(ROB_Rt_Data_valid),
.rft_rs_spec_data(Rs_Data_spec),
.rft_rs_spec_data_valid(ROB_RS_Data_valid),
//Inputs: Retire
.rft_oq_tag(Retire_rd_tag),
.rft_retire_valid(rob_retire_valid),
.rft_flush(Retire_flush),
//Outputs: Retire
.rft_out_rdes_add(Retire_rd_reg),
.rft_out_pc(Retire_pc),
.rft_out_inst_no_rdes(retire_out_inst_no_rdes),
.rft_out_ld_st_sel(retire_ld_st_sel),
.rft_out_branch(rft_Retire_branch),
.rft_out_branch_taken(Retire_branch_taken),
.rft_out_data(Retire_data),
.rft_out_spec_data_valid(retire_spec_data_valid),
.rft_out_valid(retire_out_valid)
```

```
);
endmodule

module RST(rst, clk, rst_rd_token, rst_rdes_add, rst_rd_wr_en, rst_rs_token, rst_rs_add, rst_rt_token, rst_rt_add,
rst_Retire_rd_tag, rst_Retire_rd_tag_valid, rst_retire_flush);

//Parameters
parameter token_size = 6;                  //Token Size: Valid + 5-bit Tag
parameter rst_size = 32;                                    //Defines RST deep

//Reset and clk inputs
input                rst;              //Asynchronous reset
input                clk;              //System clock
//Write Port 0
input[5:0]             rst_rd_token;             //Destination Register Token
input[4:0]             rst_rdes_add;          //Destination Register Address
input          rst_rd_wr_en; //Write Enable to rename Destination Register

//Read Port 0:
output[5:0]                                                 rst_rs_token;
input[4:0]                              rst_rs_add;                         //

//Read Port 1:
output[5:0]                                                 rst_rt_token;
input[4:0]                                                  rst_rt_add;                         //

//CDB Interface
input[4:0]                                                  rst_Retire_rd_tag;                  //
input                                                       rst_Retire_rd_tag_valid;    //
input                                                       rst_retire_flush;

reg[5:0]      RST_table[rst_size-1:0];                      //RST table structure

integer i;                                                                                                  //To
count from 0 to rst_size (32)
//reg[5:0] i;
          //To count from 0 to rst_size (32)

//One-Hot Decoder
integer N;                                                                          //Index to sweep the 32
conditions of the 32-bit One-Hot Decoder
//reg[5:0] N;
//Index to sweep the 32 conditions of the 32-bit One-Hot Decoder
//Internal "Write Port 1"
wire[5:0] rst_clear_token; //Clears the tag published (released)
                                        //by the CDB (6'b0 data)
reg[31:0] rst_one_hot_wr_en; //Write Enable one-hot encoded indicating the
                                        //TAG published (released) by the CDB
assign rst_clear_token = 6'b00_0000;

//RST write process
always@(posedge clk or posedge rst)
begin
          if(rst)
          begin
                    for(i=0; i<rst_size; i=i+1)
                    begin
```

```
                        RST_table[i] <= 0;
                //Clear RST when reset
                end
        end
        else if(rst_retire_flush)
        begin
                for(i=0; i<rst_size; i=i+1)
                begin
                        RST_table[i] <= 0;
                //Clear RST when reset
                end
        end
        else
        begin
                if(rst_rd_wr_en)
                begin
                        RST_table[rst_rdes_add] <= rst_rd_token;
                end
                //32-bit One-Hot Decoder implemented as 32-to-5 Priority Decoder
                for(N=0; N<32; N=N+1)
                begin
                        if(rst_one_hot_wr_en[N])
                                RST_table[N] <= rst_clear_token;
                end
        end
end
wire[5:0] rst_retire_token;
assign  rst_retire_token = {rst_Retire_rd_tag_valid, rst_Retire_rd_tag};
//5-to-32 One-Hot Binary Decoder
reg[31:0] rst_rd_32_wr_en;
always @(*)
begin: WRITE_EN_32_RD_DECODED        //5-to-32 Binary Decoder
        for(N=0; N<32; N=N+1)
        begin
                if((rst_rdes_add == N) && rst_rd_wr_en)
                        rst_rd_32_wr_en[N] = 1'b1;
                else
                        rst_rd_32_wr_en[N] = 1'b0;
        end
end
//Write Enable to Register File and to clear RST
always @(rst_retire_token, rst_rd_32_wr_en, rst_Retire_rd_tag_valid,
        RST_table[0], RST_table[1], RST_table[2], RST_table[3], RST_table[4], RST_table[5], RST_table[6], RST_table[7],
RST_table[8], RST_table[9],
        RST_table[10], RST_table[11], RST_table[12], RST_table[13], RST_table[14], RST_table[15], RST_table[16],
RST_table[17], RST_table[18], RST_table[19],
        RST_table[20], RST_table[21], RST_table[22], RST_table[23], RST_table[24], RST_table[25], RST_table[26],
RST_table[27], RST_table[28], RST_table[29],
        RST_table[30], RST_table[31])
begin: WRITE_EN_RD
        for(N=0; N<32; N=N+1)
        begin
                if((rst_retire_token == RST_table[N]) && rst_Retire_rd_tag_valid)
   begin
     if(!rst_rd_32_wr_en[N])
       rst_one_hot_wr_en[N] = 1'b1;   //To Clear RST
     else
```

```
      rst_one_hot_wr_en[N] = 1'b0;
    end
                    else
    begin
      rst_one_hot_wr_en[N] = 1'b0;
    end
  end
end
//RST read process
//assign rst_rs_tag = RST_table[rst_rs_add][4:0];
//assign rst_rs_tag_valid = RST_table[rst_rs_add][5];
assign rst_rs_token = RST_table[rst_rs_add];
//assign rst_rt_tag = RST_table[rst_rt_add][4:0];
//assign rst_rt_tag_valid = RST_table[rst_rt_add][5];
assign rst_rt_token = RST_table[rst_rt_add];
endmodule
module ORDER_QUEUE(rst, clk, oq_tag_out, oq_tag_in, oq_disp_en, oq_retire_valid, oq_flush);
//Parameters
parameter tag_size = 5;                    //Defines tag line size
parameter oq_fifo_size = 32;                 //Defines order queue fifo deep
//Reset and clk inputs
input                    rst;               //Asynchronous reset
input                    clk;                //System clock
//Interfaces with dispatcher
input[tag_size-1:0]             oq_tag_in;                //TAG FIFO Data In from Dispatcher
input                oq_disp_en;     //Ctrl: Write enable from Dispatcher
//Interfaces with Reg File Temp
output[tag_size-1:0]         oq_tag_out;       //TAG FIFO Data Out to RegFileTemp
input                oq_retire_valid;       //Read enable from RegFileTemp
//Interface with ctrl
input                oq_flush;                //To flush data when branch mispredicted
reg[tag_size-1:0]       oq_fifo[oq_fifo_size-1:0];   //ORDER QUEUE FIFO
integer i;
            //To count from 0 to oq_fifo_size (32)
reg[4:0]oq_wr_ptr;
//Order Queue Write Pointer
reg[4:0]oq_rd_ptr;
//Order Queue Read Pointer
//Sequential assigments----------------------------------
//FIFO write process
always@(posedge clk or posedge rst)
begin
            if(rst)
            begin
                        for(i=0; i<oq_fifo_size; i=i+1)
                        begin
                                    oq_fifo[i] <= 0;       //In reset filled with 0s;
                        end
                        oq_wr_ptr <= 5'b00000;                //Order Queue FIFO Initially Empty
            end
            else if(oq_flush)
            begin
                        for(i=0; i<oq_fifo_size; i=i+1)
                        begin
                                    oq_fifo[i] <= 0;       //Flush: filled with 0s;
                        end
                        oq_wr_ptr <= 5'b00000;                //Order Queue FIFO -> Empty
```

```
		end
		else if(oq_disp_en) //Requested by the "Dispatch Unit" only when a
                                    //instruction with Rdes can be dispatched.
		begin
			oq_fifo[oq_wr_ptr] <= oq_tag_in;
			oq_wr_ptr <= oq_wr_ptr + 1;	//Write Pointer increased
		end
end
//FIFO read process
always@(posedge clk or posedge rst)
begin
		if(rst)
		begin
			oq_rd_ptr <= 5'b00000;
		end
		else if(oq_flush)
		begin
			oq_rd_ptr <= 5'b00000;
		end
		else if(oq_retire_valid)
		begin
			oq_rd_ptr <= oq_rd_ptr + 1;
		end
end
assign oq_tag_out = oq_fifo[oq_rd_ptr];
endmodule
module RegFileTemp (rst, clk,
                    //Inputs from Disptacher: New Entry
                    rft_dispatch_enable, rft_Dispatch_Rd_tag, rft_rdes_add, rft_pc,
                    //Inputs Instruction Type from Dispatcher
                    rft_inst_no_rdes, rft_ld_st_sel, rft_branch,
                                    // to generate: st_ready and valid
                   //Inputs from CDB: Update Entry
                   rft_CDB_tag, rft_cdb_spec_data, rft_CDB_tag_valid, /*rft_cdb_branch,*/ rft_CDB_branch_taken,
                   //Input Consult
                   rft_rt_tag, rft_rs_tag,
                   //Output Consult
                   rft_rt_spec_data, rft_rt_spec_data_valid, rft_rs_spec_data, rft_rs_spec_data_valid,
                   //Inputs: Retire
                   rft_oq_tag, rft_retire_valid, rft_flush,
                   //Outputs: Retire
                   rft_out_rdes_add, rft_out_pc,
                   rft_out_inst_no_rdes, rft_out_ld_st_sel, rft_out_branch, rft_out_branch_taken,
                   rft_out_data, rft_out_spec_data_valid, rft_out_valid);

//Reset and clk inputs
input                   rst;                //Asynchronous reset
input                   clk;                 //System clock
//Inputs from Disptacher: New Entry
input rft_dispatch_enable;
input[4:0] rft_Dispatch_Rd_tag;
input[4:0] rft_rdes_add;
input[31:0]  rft_pc;
//Inputs Instruction Type from Dispatcher: New Entry
input rft_inst_no_rdes;
input rft_ld_st_sel;
input rft_branch;
```

```
//Inputs from CDB: Update Entry
input[4:0] rft_CDB_tag;
input[31:0] rft_cdb_spec_data;
input rft_CDB_tag_valid;
//input rft_cdb_branch;
input rft_CDB_branch_taken;
//Input Consult
input[4:0] rft_rt_tag;
input[4:0] rft_rs_tag;
//Output Consult
output[31:0] rft_rt_spec_data;
output rft_rt_spec_data_valid;
output[31:0] rft_rs_spec_data;
output rft_rs_spec_data_valid;
//Inputs: Retire
input[4:0] rft_oq_tag;
input rft_retire_valid;
input rft_flush;
//Outputs: Retire
output[4:0]   rft_out_rdes_add;
output[31:0]  rft_out_pc;
output        rft_out_inst_no_rdes;
output        rft_out_ld_st_sel;
output        rft_out_branch;
output        rft_out_branch_taken;
output[31:0]  rft_out_data;
output        rft_out_spec_data_valid;
output        rft_out_valid;
//ROB FIFO structure
reg[4:0]                rft_fifo_rdes_add[31:0];
reg[31:0]                       rft_fifo_pc[31:0];
reg                         rft_fifo_inst_no_rdes[31:0];
reg                         rft_fifo_ld_st_sel[31:0];
reg                         rft_fifo_branch[31:0];
reg                         rft_fifo_branch_taken[31:0];
reg[31:0]                       rft_fifo_data[31:0];
reg                         rft_fifo_spec_data_valid[31:0];
reg                         rft_fifo_valid[31:0];
integer i;                                              //To count from 0 to oq_fifo_size (32)
parameter buffer_size = 32;          //Defines buffer size
//RegFileTemp FIFO write Process
always@(posedge clk or posedge rst)
begin
        if(rst)
        begin
                for(i=0; i<buffer_size; i=i+1)
                begin
    rft_fifo_rdes_add[i]<= 0;
    rft_fifo_pc[i]<= 0;
    rft_fifo_inst_no_rdes[i]<= 0;
    rft_fifo_ld_st_sel[i]<= 0;
    rft_fifo_branch[i]<= 0;
    rft_fifo_branch_taken[i]<= 0;
    rft_fifo_data[i]<= 0;
    rft_fifo_spec_data_valid[i]<= 0;
    rft_fifo_valid[i]<= 0;
                end
```

```
            end
            else if(rft_flush)
  begin
                        for(i=0; i<buffer_size; i=i+1)
                        begin
     rft_fifo_rdes_add[i]<= 0;
     rft_fifo_pc[i]<= 0;
     rft_fifo_inst_no_rdes[i]<= 0;
     rft_fifo_ld_st_sel[i]<= 0;
     rft_fifo_branch[i]<= 0;
     rft_fifo_branch_taken[i]<= 0;
     rft_fifo_data[i]<= 0;
     rft_fifo_spec_data_valid[i]<= 0;
     rft_fifo_valid[i]<= 0;
                        end
  end
  else
  begin
    if(rft_dispatch_enable) //New Entry Write Process: Data written
                                                //from Dispatch
    begin
      rft_fifo_rdes_add[rft_Dispatch_Rd_tag]<= rft_rdes_add;
      rft_fifo_pc[rft_Dispatch_Rd_tag]<= rft_pc;
      rft_fifo_inst_no_rdes[rft_Dispatch_Rd_tag]<= rft_inst_no_rdes;
      rft_fifo_ld_st_sel[rft_Dispatch_Rd_tag]<= rft_ld_st_sel;
      rft_fifo_branch[rft_Dispatch_Rd_tag]<= rft_branch;
      rft_fifo_valid[rft_Dispatch_Rd_tag]<= 1;
    end
    if(rft_CDB_tag_valid) //Update Write Process: Data written from CDB
    begin
      rft_fifo_branch_taken[rft_CDB_tag]<= rft_CDB_branch_taken;
      rft_fifo_data[rft_CDB_tag]<= rft_cdb_spec_data;
      rft_fifo_spec_data_valid[rft_CDB_tag]<= 1;
    end
    if(rft_retire_valid) //Retire Write Process
    begin
      rft_fifo_rdes_add[rft_oq_tag]<= 0;
      rft_fifo_pc[rft_oq_tag]<= 0;
      rft_fifo_inst_no_rdes[rft_oq_tag]<= 0;
      rft_fifo_ld_st_sel[rft_oq_tag]<= 0;
      rft_fifo_branch[rft_oq_tag]<= 0;
      rft_fifo_branch_taken[rft_oq_tag]<= 0;
      rft_fifo_data[rft_oq_tag]<= 0;
      rft_fifo_spec_data_valid[rft_oq_tag]<= 0;
      rft_fifo_valid[rft_oq_tag]<= 0;
    end
  end
end
//RegFileTemp FIFO Ready process
//Retire Read Process
assign rft_out_rdes_add       = rft_fifo_rdes_add[rft_oq_tag];
assign rft_out_pc             = rft_fifo_pc[rft_oq_tag];
assign rft_out_inst_no_rdes   = rft_fifo_inst_no_rdes[rft_oq_tag];
assign rft_out_ld_st_sel      = rft_fifo_ld_st_sel[rft_oq_tag];
assign rft_out_branch         = rft_fifo_branch[rft_oq_tag];
assign rft_out_branch_taken   = rft_fifo_branch_taken[rft_oq_tag];
assign rft_out_data           = rft_fifo_data[rft_oq_tag];
```

```
assign rft_out_spec_data_valid = rft_fifo_spec_data_valid[rft_oq_tag];
assign rft_out_valid          = rft_fifo_valid[rft_oq_tag];
//rt Consult Read Process
//rs Consult Read Process
//Muxes to bypass Register Data
wire sel_rs;
wire sel_rt;
assign sel_rs = ((rft_CDB_tag==rft_rs_tag) && rft_CDB_tag_valid);
assign sel_rt = ((rft_CDB_tag==rft_rt_tag) && rft_CDB_tag_valid);
assign rft_rt_spec_data = (sel_rt) ?  rft_cdb_spec_data : rft_fifo_data[rft_rt_tag];
assign rft_rs_spec_data = (sel_rs) ?  rft_cdb_spec_data : rft_fifo_data[rft_rs_tag];
assign rft_rt_spec_data_valid = (sel_rt) ?  rft_CDB_tag_valid : rft_fifo_spec_data_valid[rft_rt_tag];
assign rft_rs_spec_data_valid = (sel_rs) ?  rft_CDB_tag_valid : rft_fifo_spec_data_valid[rft_rs_tag];
endmodule
```

ALU en una arquitectura superscalar

```
module ExecUnit(Op1, Op2, Opcode, Shamt, int_rdes_tag,Result, tag_out, ALU_branch, ALU_branch_taken);
//Parameters
parameter w_size = 32;    //Defines word width
//Interfaces with Integer Queue
input[31:0] Op1;
input[31:0] Op2;
input[3:0] Opcode;
input[4:0] Shamt;
input[4:0] int_rdes_tag;
//Interfaces with Issue Unit
output reg[w_size-1:0] Result;
output[4:0] tag_out;
output reg ALU_branch;
output reg ALU_branch_taken;
//ALU operations define
`define ADD  4'h0
`define ADDU  4'h1
`define AND  4'h2
`define BEQ  4'h3
`define BNE  4'h4
`define NOR  4'h5
`define OR  4'h6
`define SLT  4'h7
`define SLTU  4'h8
`define SLL  4'h9
`define SRL  4'hA
`define SUB  4'hB
//Destination Register bypass logic
assign tag_out =  int_rdes_tag;
//ALU operations
wire[w_size-1:0] ALU_SUB;
wire ALU_zero_flag;
assign ALU_SUB = Op1 - Op2;
assign ALU_zero_flag = (Result == 0);
//ALU operation Logic
always@(*)
begin
        ALU_branch = 0;
```

```
        ALU_branch_taken = 0;

        case(Opcode)
                `ADD: Result = Op1 + Op2;
                `ADDU: Result = Op1 + Op2;
                `AND: Result = Op1 & Op2;
                `BEQ:
                begin
                        ALU_branch = 1;
                        ALU_branch_taken = ALU_zero_flag;
                        Result = ALU_SUB;
                end
                `BNE:
                begin
                        ALU_branch = 1;
                        ALU_branch_taken = !ALU_zero_flag;
                        Result = ALU_SUB;
                end
                `NOR: Result = ~(Op1 | Op2);
                `OR: Result = Op1 | Op2;
                `SLT: Result = ALU_SUB[w_size-1];
                `SLTU: Result = (Op1 < Op2) ? 1 : 0;
                `SLL: Result = (Op2 << Shamt);
                `SRL: Result = (Op2 >> Shamt);
                `SUB: Result = ALU_SUB;
                default:
                begin
                        Result = 0;
                        ALU_branch = 0;
                        ALU_branch_taken = 0;
                end
        endcase
end
endmodule
```

Data Cache

```
module DCache(clk, rst, issueque_ready, issueblk_issue, issueque_address, issueque_data, issueque_opcode,
issueque_rd_tag, Ready_ld_st, Issue_ld_st, ld_data, ld_tag_out, retire_sw_ready, wcache_flush);
//DataCacheAddrBackDoor, DataCacheDoutBackDoor);
//Parameters
parameter w_size = 32;                          //Defines word width
parameter cache_size = 32;
//Reset and clk inputs
input                   rst;                    //Asynchronous reset
input                   clk;                    //System clock
//Interfaces with LD SW Issue Queue
input issueque_ready;
output issueblk_issue;
input[31:0] issueque_address;
input[w_size-1:0] issueque_data;
input issueque_opcode;                  //Load Store Selector -> "Operation Code"
//0: Load, 1:Store
input[4:0] issueque_rd_tag;
//Interfaces with Issue Unit
output Ready_ld_st;
```

```
input Issue_ld_st;
output[w_size-1:0] ld_data;
output[4:0] ld_tag_out;

//Interfaces with the retire BUS
input retire_sw_ready;
input wcache_flush;
//Backdoor: Added for Instrumentation
//input[4:0] DataCacheAddrBackDoor;
//output[w_size-1:0] DataCacheDoutBackDoor;
//Write Cache Structure
reg[w_size-1:0]       wc_data[3:0];   //Data
reg[w_size-1:0]       wc_add[3:0];   //Address
reg[w_size-1:0]       wc_valid[3:0];   //Address
reg[1:0]                              wc_wrt_ptr;
reg[1:0]                              wc_rd_ptr;

//Memory Structure
reg[w_size-1:0]       DCACHE[cache_size-1:0];   //Data Cache

wire[w_size-1:0] addr_shifted;
wire[w_size-1:0] Cache_data_out;
reg write_cache_hit;
reg[1:0] wc_hit_add;

integer N;

assign addr_shifted = {2'b00, issueque_address[31:2]}; //Data Cache is Byte-addressable so 2 LSB are discarded.

//Destination Register bypass logic
assign ld_tag_out =  issueque_rd_tag;

//Write Cache write process
always@(posedge clk or posedge rst)
begin
        if(rst)
        begin
                wc_wrt_ptr <= 0;
                for(N=0; N < 4; N=N+1)    //Clear Write Cache
                begin
                        wc_data[N] <= 0;
                        wc_add[N] <= 0;
                        wc_valid[N]  <= 0;
                end
        end
        else if(wcache_flush)
        begin
                wc_wrt_ptr <= 0;
                for(N=0; N < 4; N=N+1)    //Clear Write Cache
                begin
                        wc_data[N] <= 0;
                        wc_add[N] <= 0;
                        wc_valid[N]  <= 0;
                end
        end
        else
        begin
```

```
                if(issueblk_issue && issueque_opcode) //Write Enable asserted when issueque_opcode=1:Store
                begin
                                wc_data[wc_wrt_ptr] <= issueque_data;
                                wc_add[wc_wrt_ptr] <= addr_shifted;
                                wc_valid[wc_wrt_ptr]  <= 1;
                                wc_wrt_ptr <= wc_wrt_ptr + 1;
                end
                if(retire_sw_ready)
                                begin
                                wc_data[wc_rd_ptr] <= 0;
                                wc_add[wc_rd_ptr] <= 0;
                                wc_valid[wc_rd_ptr]  <= 0;
                end
        end
end

//RAM Write Process
always@(posedge clk or posedge rst)
begin
 if(rst) //Write Enable asserted when issueque_opcode=1:Store
        wc_rd_ptr <= 0;
else if(wcache_flush)
        wc_rd_ptr <= 0;
else if(retire_sw_ready)
        wc_rd_ptr <= wc_rd_ptr + 1;
end

always@(posedge clk)
begin
 if(retire_sw_ready) //Write Enable asserted when issueque_opcode=1:Store
                DCACHE[wc_add[wc_rd_ptr]] <= wc_data[wc_rd_ptr];
end

//RAM Read Ports
//Reading done combinationally
assign Cache_data_out = (issueque_opcode) ? issueque_data : DCACHE[addr_shifted]; //Data Cache is Byte-
addressable so 2 LSB are discarded.
//BackDoor: Added for Instrumentation
//assign DataCacheDoutBackDoor = DCACHE[DataCacheAddrBackDoor];
//Issue Ready Logic
assign Ready_ld_st = issueque_ready; //& !issueque_opcode;   //Issue request asserted only when Load
Operation.
assign issueblk_issue = Issue_ld_st; //| (issueque_ready & issueque_opcode);  //When Load: Done asserted
when Issue Unit reserves CDB.
//Write Cache Hit logic
always
@(wc_add[0],wc_add[1],wc_add[2],wc_add[3],addr_shifted,wc_valid[0],wc_valid[1],wc_valid[2],wc_valid[3])
begin
        for(N=0; N < 4; N=N+1)     //Clear Write Cache
        begin
                if((wc_add[N] == addr_shifted)&& (wc_valid[N]))
                begin
                        write_cache_hit = 1;
                        wc_hit_add = N;
                end
                else
                begin
```

```
                        write_cache_hit = 0;
                        wc_hit_add = 0;
                end
        end
end
//Memory Data out mux
assign ld_data = (write_cache_hit) ? wc_data[wc_hit_add] : Cache_data_out;
endmodule
```

Instruction cache

```
module Inst_Cache(arst, clk, pc_in, rd_en, d_out, d_out_valid);

module I_CACHE(clk,rst,PC_in,Dout,Rd_en,Dout_valid);
input clk, rst, Rd_en;

parameter address_depth=7;
parameter word_depth =32;
parameter bank_rows  =2**address_depth;

input  [word_depth-1:0]PC_in;
output reg [(word_depth*4)-1:0]Dout;
output Dout_valid;

reg [address_depth:0] presetCount;

//Register Matrix
reg [word_depth-1:0] mem [0:bank_rows-1];
wire [31:0] addrShifted={2'b00, PC_in[31:4],2'b00};

//assign Dout={mem[addrShifted+3],mem[addrShifted+2],mem[addrShifted+1],mem[addrShifted]};
always @(posedge clk or posedge rst)
begin
        if (rst) begin
                for (presetCount=0; presetCount<bank_rows; presetCount=presetCount+1)
                begin
                        case (presetCount)
                        //Without ident intentionally
                        //Values are supposed to be code.
//mul test
/*0: mem[presetCount]<=32'h00000020;
1: mem[presetCount]<=32'h00625018;
2: mem[presetCount]<=32'h00835818;
3: mem[presetCount]<=32'h00A46018;
4: mem[presetCount]<=32'h00C56018;
5: mem[presetCount]<=32'h00A21820;
6: mem[presetCount]<=32'h01227018;
7: mem[presetCount]<=32'h01037818;
8: mem[presetCount]<=32'h03FEE820;
9: mem[presetCount]<=32'h00000020;
10: mem[presetCount]<=32'h00631818;
11: mem[presetCount]<=32'h00631818;
12: mem[presetCount]<=32'h00631818;
13: mem[presetCount]<=32'h00631818;
14: mem[presetCount]<=32'h00000020;*/
```

```
default:  mem[presetCount]<=32'h00000000;
                                    endcase
                        end
            end
end
always@(posedge clk or posedge rst)
begin
  if(rst)
  begin
     d_out <= 0; //128-bit '0'
     Dout_valid <= 1'b0;
  end
  else if(rd_en)
  begin              Dout<={mem[addrShifted+3],mem[addrShifted+2],mem[addrShifted+1],mem[addrShifted]};
     Dout_valid <= 1'b1;
  end
  else
  begin
                         Dout <= Dout;
     Dout_valid <= 1'b0;
  end
end
endmodule
```

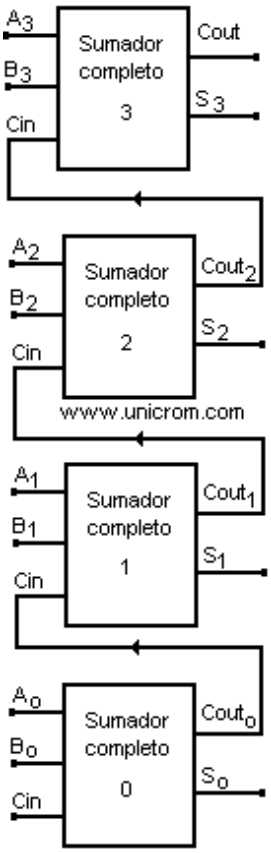
Apéndice C

Diseño de un simulador de fallas en circuitos integrados

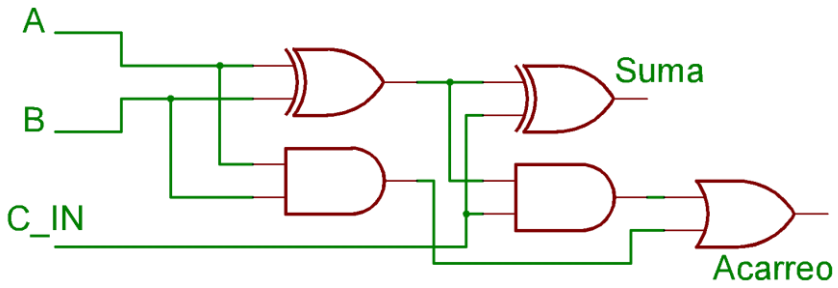
Contenido

Análisis de Circuitos básicos para iniciar el diseño

Sumador de 16 – bits

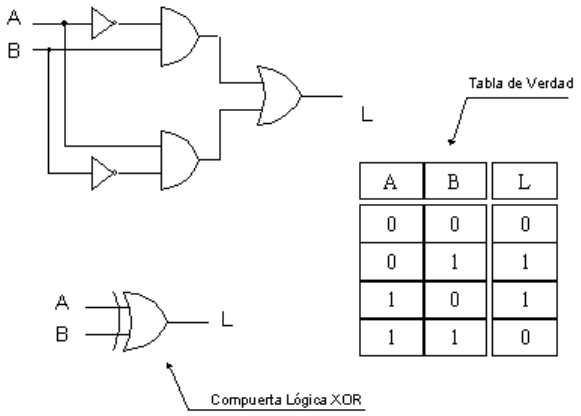


Esta imagen representa la forma general de un sumador de n bits, y es la misma que se utilizó para el diseño del sumador de 16 – bits.



En este diagrama se muestra la lógica combinacional utilizada para el diseño del circuito sumador completo de 1 – bit

Diseño de XOR:



Una vez diseñado el sumador completo de 16 – bits pareciera que el circuito está resuelto, pero el mayor reto está en la sincronización de la retroalimentación; en las figuras de arriba se encuentra el diseño del multiplexor utilizado (se diseñó 1 de 1 bits y a partir de este se generó un multiplexor de 16 – bits). También expone como se tuvieron que agregar 2 flip-flops en cascada para poder mostrar los resultados consecutivamente en diferentes tiempos y se pueda generar correctamente la serie de Fibonacci.

Multiplexor de 16 – bits

Para lograr generar la serie Fibonacci se necesita que el valor de uno de los operadores sea el resultado de la operación anterior y para el otro operador el valor sea el resultado de 2 operaciones anteriores o lo que es lo mismo el estado anterior del otro operador. Se utilizaron 2 flip-flops en cascada donde la salida del primer fli-flop se conecta al operador A y la salida del segundo flip-flop se conecta al operador B así se logra que al operador 1 le entre el resultado anterior y al(el) (o)Operador B tome el estado anterior de A.

Sumador - Verilog Netlist

xor_gate

```
module xor_gate(input in0, input in1, output out);
  wire in0_inv, in1_inv;
  wire and_in0_inv, and_in1_inv, xor_out;
  not not_in0(in0_inv,in0);
  not not_in1(in1_inv,in1);
  and and_A(and_in0_inv, in0_inv, in1);
  and and_B(and_in1_inv, in0, in1_inv);
  or or_output(xor_out, and_in0_inv, and_in1_inv);
  assign out = xor_out;
endmodule
```

dff

```
primitive dff(q, clk, d);
  output q;
  reg q;
  input clk, d;
  table
    // clock data q q+
    (01) 0 : ? : 0 ;
    (01) 1 : ? : 1 ;
    (0?) 1 : 1 : 1 ;
    (0?) 0 : 0 : 0 ;
    // ignore negative edge of clock
    (?0) ? : ? : - ;
    // ignore data changes on steady clock
    ? (??) : ? : - ;
  endtable
endprimitive
```

Mux2to1_16bits

```
module Mux2to1_16bits(input S,
                      input [15:0]in0,
                      input [15:0]in1,
                      output [15:0]Y);
  wire [15:0]out_Y;
  generate
  genvar i;
    for(i=0; i<16; i=i+1)
```

```
    Mux2to1 Mux2to1(.S(S),.in0(in0[i]), .in1(in1[i]), .Y(out_Y[i]));
  endgenerate
  assign Y = out_Y;
endmodule
```

Mux2to1

```
module Mux2to1(input S, input in0, input in1, output Y);
  wire S0, S1, and0, and1, out_Y, test;
  not not0(S0,S);
  not not1(S1,S0);
  and and_0(and0, in0, S0);
  and and_1(and1, in1, S1);
  or or_Y(out_Y, and0, and1);
  assign Y = out_Y;
endmodule
```

Full_Adder_1bit

```
module Full_Adder_1bit(input A, input B, input Cin, output Add, output Cout);
  wire out_xor0;
  wire out_and0;
  wire out_and1;
  wire out_add;
  wire out_Cout;
  xor_gate xor0(.in0(A),.in1(B),.out(out_xor0));
  xor_gate xor1(.in0(out_xor0),.in1(Cin),.out(out_add));
  and and0(out_and0, A, B);
  and and1(out_and1,out_xor0, Cin);
  or or_Cout(out_Cout, out_and0, out_and1);
  assign Add = out_add;
  assign Cout = out_Cout;
endmodule
```

Full_Adder_16bit

```
module Full_Adder_16bits(input [15:0]A,
                         input [15:0]B,
                         input Cin,
                         output [15:0]Add,
                         output Cout);
  wire [14:0]Cout_wire, [15:0]Suma, out_Cout;
  Full_Adder_1bit Add0(.A(A[0]), .B(B[0]), .Cin(Cin), .Add(Suma[0]), .Cout(Cout_wire[0]));
  generate
    genvar i;
    for(i=0; i<14; i=i+1)
    begin
      Full_Adder_1bit Add1(.A(A[i+1]), .B(B[i+1]), .Cin(Cout_wire[i]), .Add(Suma[i+1]), .Cout(Cout_wire[i+1]));
    end
  endgenerate
  Full_Adder_1bit Add15(.A(A[15]), .B(B[15]), .Cin(Cout_wire[14]), .Add(Suma[15]), .Cout(out_Cout));
  assign Cout = out_Cout;
  assign Add = Suma;
endmodule
```

Adder – Circuto requerido con retroalimentación

```
module Adder(input clk,
             input SELA,
             input SELB,
             input CI,
             input [15:0]OpA,
             input [15:0]OpB,
             output CO,
             output [15:0]SUM);
  wire Cout;
  wire [15:0]Op1;
  wire [15:0]Op2;
  wire [15:0]Result;
  wire [15:0]ResultSyncA;
  wire [15:0]ResultSyncB;

  Mux2to1_16bits MuxOpA(.S(SELA), .in0(OpA), .in1(ResultSyncA), .Y(Op1));
  Mux2to1_16bits MuxOpB(.S(SELB), .in0(OpB), .in1(ResultSyncB), .Y(Op2));

  Full_Adder_16bits Full_Adder(.A(Op1), .B(Op2), .Cin(CI), .Add(Result), .Cout(Cout));

  generate
    genvar i;
    for(i=0; i<16; i=i+1)
    begin
      dff SyncA(ResultSyncA[i], clk, Result[i]);
      dff SyncB(ResultSyncB[i], clk, ResultSyncA[i]);
    end
  endgenerate
  assign CO = Cout;
  assign SUM = Result;
endmodule
```

Testbech Code

```
module testbench();
  reg [15:0]Op1;
  reg [15:0]Op2;
  reg Cin;
  reg S0;
  reg S1;
  reg clk;
  wire Cout;
  wire [15:0]Result;
  initial
  begin
       clk = 1;
       forever #5 clk = ~clk;
  end
  initial
  begin
    Cin = 0;
    Op1 = 0;
    Op2 = 0;
    S0 = 0;
    S1 = 0;
```

```
    #0
    //5 operaciones con operandos externos
    repeat (5) begin
      #10
      Op1 = {$random} %15000;
      Op2 = {$random} %15000;
      S0 = 0;
      S1 = 0;
    end
    //5 operaciones donde el operando A es la retroalimentacion de la suma anterior
    repeat (5) begin
      #10
      Op1 = {$random} %1500;
      Op2 = {$random} %1500;
      S0 = 1;
      S1 = 0;
    end
  //5 operaciones donde el operando B es la retroalimentacion de la suma anterior
    repeat (5) begin
      #10
      Op1 = {$random} %150;
      Op2 = {$random} %150;
      S0 = 0;
      S1 = 1;
    end
//5 operaciones donde los 2 operandos son retroalimentados A por el resultado anterior y B por el estado anterior de B
    repeat (5) begin
      #10
      Op1 = {$random} %1500;
      Op2 = {$random} %1500;
      S0 = 1;
      S1 = 1;
    end
    //5 operaciones con operandos externos
    repeat (5) begin
      #10
      Op1 = {$random} %150;
      Op2 = {$random} %150;
      S0 = 0;
      S1 = 0;
    end
    //Inicia la serie de Fibonacci
      #10
      Op1 = 0;
      Op2 = 0;
      #10
      S0 = 1;
      Op2 = 1;
      #10
      S1 = 1;
  end
Adder Adder(.clk(clk), .SELA(S0), .SELB(S1), .CI(Cin), .OpA(Op1), .OpB(Op2), .CO(Cout), .SUM(Result));
endmodule
```

Sumador - Transcript – 500ns simulation

```
VSIM 24> run
# ********************************************************************************
# | Selector 0 | Selector 1 | Operando A | Operando B     |       SUMA     |
# ********************************************************************************
#  |      0       |      0       |          0       |          0       |          0       |
#  |      0       |      0       |       4748       |       8097       |      12845       |
#  |      0       |      0       |      13057       |       2987       |      16044       |
#  |      0       |      0       |       3957       |       8957       |      12914       |
#  |      0       |      0       |       7325       |       9082       |      16407       |
#  |      0       |      0       |       8361       |      11029       |      19390       |
#  |      1       |      0       |        318       |       1097       |      20487       |
#  |      1       |      0       |        129       |        112       |      20599       |
#  |      1       |      0       |        289       |        426       |      21025       |
#  |      1       |      0       |        697       |       1122       |      22147       |
#  |      1       |      0       |        885       |        255       |      22402       |
#  |      0       |      1       |         44       |         39       |      22191       |
#  |      0       |      1       |         36       |          0       |      22438       |
#  |      0       |      1       |        118       |        103       |      22309       |
#  |      0       |      1       |        134       |        143       |      22572       |
#  |      0       |      1       |         79       |        139       |      22388       |
#  |      1       |      1       |        523       |        790       |      44960       |
#  |      1       |      1       |        428       |       1036       |       1812       |
#  |      1       |      1       |       1014       |       1221       |      46772       |
#  |      1       |      1       |        522       |       1431       |      48584       |
#  |      1       |      1       |         61       |       1383       |      29820       |
#  |      0       |      0       |        105       |         85       |        190       |
#  |      0       |      0       |        138       |        132       |        270       |
#  |      0       |      0       |         81       |         29       |        110       |
#  |      0       |      0       |         47       |         20       |         67       |
#  |      0       |      0       |         16       |         58       |         74       |
#  |      0       |      0       |          0       |          0       |          0       |
#  |      1       |      0       |          0       |          1       |          1       |
#  |      1       |      1       |          0       |          1       |          1       |
#  |      1       |      1       |          0       |          1       |          2       |
#  |      1       |      1       |          0       |          1       |          3       |
#  |      1       |      1       |          0       |          1       |          5       |
#  |      1       |      1       |          0       |          1       |          8       |
#  |      1       |      1       |          0       |          1       |         13       |
#  |      1       |      1       |          0       |          1       |         21       |
#  |      1       |      1       |          0       |          1       |         34       |
#  |      1       |      1       |          0       |          1       |         55       |
#  |      1       |      1       |          0       |          1       |         89       |
#  |      1       |      1       |          0       |          1       |        144       |
#  |      1       |      1       |          0       |          1       |        233       |
#  |      1       |      1       |          0       |          1       |        377       |
#  |      1       |      1       |          0       |          1       |        610       |
#  |      1       |      1       |          0       |          1       |        987       |
#  |      1       |      1       |          0       |          1       |       1597       |
#  |      1       |      1       |          0       |          1       |       2584       |
#  |      1       |      1       |          0       |          1       |       4181       |
#  |      1       |      1       |          0       |          1       |       6765       |
#  |      1       |      1       |          0       |          1       |      10946       |
#  |      1       |      1       |          0       |          1       |      17711       |
#  |      1       |      1       |          0       |          1       |      28657       |
VSIM 25>
```

Los vectores que están en naranja son operaciones que se realizan interactuando con el encendido y apagado de los selectores para posteriormente iniciar con la seria de Fobinacci.

Resultados y Aportaciones del simulador de fallas

```
#                              ...Control/Observation Cone...
#
# ...Signal to which the fault is injected: /testbench/Adder/Full_Adder/Add0/Cin...
# TEST SIN FALLAS
# Stuck@0: Test vector -> 0^1 with A[0]=0, B[0]=0, Cin=1
# S0=0 S1=0 748+97=846 Cin=1 Cout=0
# S0=0 S1=0 956+325=1282 Cin=1 Cout=0
# S0=0 S1=0 318+597=916 Cin=1 Cout=0
# S0=0 S1=0 926+197=1124 Cin=1 Cout=0
# S0=0 S1=0 194+39=234 Cin=1 Cout=0
# Stuck@0: Test vector -> 0^1 with A[0]=1, B[0]=1, Cin=1
# S0=0 S1=0 253+884=1138 Cin=1 Cout=0
# S0=0 S1=0 523+290=814 Cin=1 Cout=0
# S0=0 S1=0 221+22=244 Cin=1 Cout=0
# S0=0 S1=0 355+485=841 Cin=1 Cout=0
# S0=0 S1=0 879+147=1027 Cin=1 Cout=0
# Stuck@0: Test vector -> 1^1 with A[0]=0, B[0]=1, Cin=1
# S0=0 S1=0 507+530=1038 Cin=1 Cout=0
# S0=0 S1=0 49+427=477 Cin=1 Cout=0
# S0=0 S1=0 365+194=560 Cin=1 Cout=0
# S0=0 S1=0 975+891=1867 Cin=1 Cout=0
# S0=0 S1=0 153+361=515 Cin=1 Cout=0
# Stuck@0: Test vector -> 1^1 with A[0]=1, B[0]=0, Cin=1
# S0=0 S1=0 854+720=1575 Cin=1 Cout=0
# S0=0 S1=0 418+897=1316 Cin=1 Cout=0
# S0=0 S1=0 544+70=615 Cin=1 Cout=0
# S0=0 S1=0 230+925=1156 Cin=1 Cout=0
# S0=0 S1=0 552+439=992 Cin=1 Cout=0
# ********************************************************************************
```

```
#                                  ...The system has been restarted...
# Injecting Stuck@0 in all vectors of the serie
# S0=0 S1=0 748+97=845 Cin=1 Cout=0
# 1 > Fault Detected:   846 first simulation, 845 current Simulation
# S0=0 S1=0 956+325=1281 Cin=1 Cout=0
# 2 > Fault Detected:  1282 first simulation, 1281 current Simulation
# S0=0 S1=0 318+597=915 Cin=1 Cout=0
# 3 > Fault Detected:   916 first simulation, 915 current Simulation
# S0=0 S1=0 926+197=1123 Cin=1 Cout=0
# 4 > Fault Detected:  1124 first simulation, 1123 current Simulation
# S0=0 S1=0 194+39=233 Cin=1 Cout=0
# 5 > Fault Detected:   234 first simulation, 233 current Simulation
# S0=0 S1=0 253+884=1137 Cin=1 Cout=0
# 6 > Fault Detected:  1138 first simulation, 1137 current Simulation
# S0=0 S1=0 523+290=813 Cin=1 Cout=0
# 7 > Fault Detected:   814 first simulation, 813 current Simulation
# S0=0 S1=0 221+22=243 Cin=1 Cout=0
# 8 > Fault Detected:   244 first simulation, 243 current Simulation
# S0=0 S1=0 355+485=840 Cin=1 Cout=0
# 9 > Fault Detected:   841 first simulation, 840 current Simulation
# S0=0 S1=0 879+147=1026 Cin=1 Cout=0
# 10 > Fault Detected:  1027 first simulation, 1026 current Simulation
# S0=0 S1=0 507+530=1037 Cin=1 Cout=0
# 11 > Fault Detected:  1038 first simulation, 1037 current Simulation
# S0=0 S1=0 49+427=476 Cin=1 Cout=0
# 12 > Fault Detected:   477 first simulation, 476 current Simulation
# S0=0 S1=0 365+194=559 Cin=1 Cout=0
# 13 > Fault Detected:   560 first simulation, 559 current Simulation
# S0=0 S1=0 975+891=1866 Cin=1 Cout=0
# 14 > Fault Detected:  1867 first simulation, 1866 current Simulation
# S0=0 S1=0 153+361=514 Cin=1 Cout=0
# 15 > Fault Detected:   515 first simulation, 514 current Simulation
# S0=0 S1=0 854+720=1574 Cin=1 Cout=0
# 16 > Fault Detected:  1575 first simulation, 1574 current Simulation
```

S0=0 S1=0 418+897=1315 Cin=1 Cout=0

17 > Fault Detected: 1316 first simulation, 1315 current Simulation

S0=0 S1=0 544+70=614 Cin=1 Cout=0

18 > Fault Detected: 615 first simulation, 614 current Simulation

S0=0 S1=0 230+925=1155 Cin=1 Cout=0

19 > Fault Detected: 1156 first simulation, 1155 current Simulation

S0=0 S1=0 552+439=991 Cin=1 Cout=0

20 > Fault Detected: 992 first simulation, 991 current Simulation

Se observa como para todos los vectores se detectó la falla ya que todas las pruebas fueron dirigidas para detectar la falla. En este caso cuando se inserta la falla, se detecta que el resultado es 1 bit menos ya que se hace un Stuck @ 0 para la señal

Transcript: Se generaron 5 vectores de cada patrón mencionado anteriormente

...Control/Observation Cone...

#

...Signal to which the fault is injected: /testbench/Adder/Full_Adder/Add0/Cin...

TEST SIN FALLAS

Stuck@1: Test vector -> 0^0 with A[0]=0, B[0]=0, Cin=0

S0=0 S1=0 748+97=845 Cin=0 Cout=0

S0=0 S1=0 956+325=1281 Cin=0 Cout=0

S0=0 S1=0 318+597=915 Cin=0 Cout=0

S0=0 S1=0 926+197=1123 Cin=0 Cout=0

S0=0 S1=0 194+39=233 Cin=0 Cout=0

Stuck@1: Test vector -> 0^0 with A[0]=1, B[0]=1, Cin=0

S0=0 S1=0 253+884=1137 Cin=0 Cout=0

S0=0 S1=0 523+290=813 Cin=0 Cout=0

S0=0 S1=0 221+22=243 Cin=0 Cout=0

S0=0 S1=0 355+485=840 Cin=0 Cout=0

S0=0 S1=0 879+147=1026 Cin=0 Cout=0

Stuck@1: Test vector -> 1^0 with A[0]=0, B[0]=1, Cin=0

S0=0 S1=0 507+530=1037 Cin=0 Cout=0

S0=0 S1=0 49+427=476 Cin=0 Cout=0

S0=0 S1=0 365+194=559 Cin=0 Cout=0

S0=0 S1=0 975+891=1866 Cin=0 Cout=0

```
# S0=0 S1=0 153+361=514 Cin=0 Cout=0
# Stuck@1: Test vector -> 1^0 with A[0]=1, B[0]=0, Cin=0
# S0=0 S1=0 854+720=1574 Cin=0 Cout=0
# S0=0 S1=0 418+897=1315 Cin=0 Cout=0
# S0=0 S1=0 544+70=614 Cin=0 Cout=0
# S0=0 S1=0 230+925=1155 Cin=0 Cout=0
# S0=0 S1=0 552+439=991 Cin=0 Cout=0
# ********************************************************************************
#                                ...The system has been restarted...
# Injecting Stuck@1 in all vectors of the serie
# S0=0 S1=0 748+97=846 Cin=0 Cout=0
# 1 > Fault Detected:   845 first simulation, 846 current Simulation
# S0=0 S1=0 956+325=1282 Cin=0 Cout=0
# 2 > Fault Detected:  1281 first simulation, 1282 current Simulation
# S0=0 S1=0 318+597=916 Cin=0 Cout=0
# 3 > Fault Detected:   915 first simulation, 916 current Simulation
# S0=0 S1=0 926+197=1124 Cin=0 Cout=0
# 4 > Fault Detected:  1123 first simulation, 1124 current Simulation
# S0=0 S1=0 194+39=234 Cin=0 Cout=0
# 5 > Fault Detected:   233 first simulation, 234 current Simulation
# S0=0 S1=0 253+884=1138 Cin=0 Cout=0
# 6 > Fault Detected:  1137 first simulation, 1138 current Simulation
# S0=0 S1=0 523+290=814 Cin=0 Cout=0
# 7 > Fault Detected:   813 first simulation, 814 current Simulation
# S0=0 S1=0 221+22=244 Cin=0 Cout=0
# 8 > Fault Detected:   243 first simulation, 244 current Simulation
# S0=0 S1=0 355+485=841 Cin=0 Cout=0
# 9 > Fault Detected:   840 first simulation, 841 current Simulation
# S0=0 S1=0 879+147=1027 Cin=0 Cout=0
# 10 > Fault Detected:  1026 first simulation, 1027 current Simulation
# S0=0 S1=0 507+530=1038 Cin=0 Cout=0
# 11 > Fault Detected:  1037 first simulation, 1038 current Simulation
# S0=0 S1=0 49+427=477 Cin=0 Cout=0
# 12 > Fault Detected:   476 first simulation, 477 current Simulation
```

S0=0 S1=0 365+194=560 Cin=0 Cout=0

13 > Fault Detected: 559 first simulation, 560 current Simulation

S0=0 S1=0 975+891=1867 Cin=0 Cout=0

14 > Fault Detected: 1866 first simulation, 1867 current Simulation

S0=0 S1=0 153+361=515 Cin=0 Cout=0

15 > Fault Detected: 514 first simulation, 515 current Simulation

S0=0 S1=0 854+720=1575 Cin=0 Cout=0

16 > Fault Detected: 1574 first simulation, 1575 current Simulation

S0=0 S1=0 418+897=1316 Cin=0 Cout=0

17 > Fault Detected: 1315 first simulation, 1316 current Simulation

S0=0 S1=0 544+70=615 Cin=0 Cout=0

18 > Fault Detected: 614 first simulation, 615 current Simulation

S0=0 S1=0 230+925=1156 Cin=0 Cout=0

19 > Fault Detected: 1155 first simulation, 1156 current Simulation

S0=0 S1=0 552+439=992 Cin=0 Cout=0

20 > Fault Detected: 991 first simulation, 992 current Simulation

Aquí se observa al igual que en los vectores de Stuck @ 0, que todos los vectores fallan para todos los casos ya que se hicieron pruebas dirigidas. En este caso como la falla fue Stuck @ 1 se observa que resultado es 1 bit más.

Herramientas complementarias del simulador de fallas

Parte del simulador de fallas que se elaboró para este ejercicio son 2 herramientas adicionales además del diseño del TestBench. La primera herramienta obtiene 100 señales random del diseño del circuito dichas señales son:

/testbench/Adder/Full_Adder/genblk1[1]/Add1/xor0/in1_inv
/testbench/Adder/Full_Adder/genblk1[4]/Add1/B
/testbench/Adder/MuxOpB/genblk1[12]/Mux2to1/in1
/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/out
/testbench/Adder/Full_Adder/genblk1[5]/Add1/out_Cout
/testbench/Adder/Full_Adder/genblk1[10]/Add1/Cin
/testbench/Adder/Full_Adder/genblk1[0]/Add1/xor1/in1
/testbench/Adder/Cout
/testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0
/testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/S
/testbench/Adder/MuxOpB/genblk1[13]/Mux2to1/and1
/testbench/Adder/Full_Adder/Add15/xor1/xor_out
/testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/in0
/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1

/testbench/Adder/Full_Adder/genblk1[10]/Add1/Add
/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1
/testbench/Adder/MuxOpA/genblk1[6]/Mux2to1/S
/testbench/Adder/Full_Adder/genblk1[6]/Add1/out_and1
/testbench/Adder/Full_Adder/genblk1[1]/Add1/xor1/and_in0_inv
/testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/S
/testbench/Adder/Full_Adder/genblk1[4]/Add1/xor0/in1
/testbench/Adder/Full_Adder/genblk1[10]/Add1/out_add
/testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/out
/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/and_in0_inv
/testbench/Adder/Full_Adder/Add0/xor0/in1_inv
/testbench/Adder/Full_Adder/Add0/xor0/out
/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/out_Y
/testbench/Adder/Full_Adder/genblk1[10]/Add1/out_and1
/testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv
/testbench/Adder/MuxOpA/genblk1[9]/Mux2to1/S1
/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/out
/testbench/Adder/MuxOpB/genblk1[4]/Mux2to1/Y
/testbench/Adder/MuxOpA/genblk1[4]/Mux2to1/out_Y
/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv
/testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y
/testbench/Adder/Full_Adder/Add0/xor1/in0_inv
/testbench/Adder/Full_Adder/genblk1[0]/Add1/out_xor0
/testbench/Adder/ResultSyncA
/testbench/Adder/Full_Adder/genblk1[4]/Add1/B
/testbench/Adder/Full_Adder/genblk1[13]/Add1/xor0/in1
/testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/in1
/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/and0
/testbench/Adder/MuxOpB/genblk1[10]/Mux2to1/S1
/testbench/Adder/Full_Adder/Cout
/testbench/Adder/Full_Adder/genblk1[1]/Add1/out_and0
/testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/and_in0_inv
/testbench/Adder/Full_Adder/genblk1[7]/Add1/xor1/out
/testbench/Adder/Full_Adder/genblk1[6]/Add1/out_add
/testbench/Adder/MuxOpB/genblk1[1]/Mux2to1/S1
/testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/S1
/testbench/Adder/Full_Adder/genblk1[10]/Add1/out_Cout
/testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0
/testbench/Adder/MuxOpA/genblk1[3]/Mux2to1/S
/testbench/Adder/Full_Adder/genblk1[8]/Add1/xor1/out
/testbench/Adder/MuxOpA/genblk1[1]/Mux2to1/S0
/testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/out
/testbench/Adder/MuxOpA/genblk1[2]/Mux2to1/and1
/testbench/Adder/MuxOpA/genblk1[3]/Mux2to1/S
/testbench/Adder/MuxOpB/Y
/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv
/testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/out_Y
/testbench/Adder/MuxOpB/genblk1[0]/Mux2to1/S
/testbench/Adder/Full_Adder/genblk1[8]/Add1/xor0/in0_inv
/testbench/Adder/Full_Adder/genblk1[11]/Add1/out_xor0
/testbench/Adder/Full_Adder/genblk1[5]/Add1/xor0/in0

/testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/and1
/testbench/Adder/MuxOpB/genblk1[0]/Mux2to1/Y
/testbench/Adder/MuxOpB/genblk1[2]/Mux2to1/out_Y
/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in1_inv
/testbench/Adder/MuxOpB/genblk1[12]/Mux2to1/S
/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1
/testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in1
/testbench/Adder/MuxOpA/genblk1[15]/Mux2to1/Y
/testbench/Adder/Full_Adder/Add0/out_xor0
/testbench/Adder/Full_Adder/genblk1[12]/Add1/xor1/and_in1_inv
/testbench/Adder/MuxOpA/genblk1[15]/Mux2to1/in1
/testbench/Adder/Full_Adder/genblk1[3]/Add1/out_and0
/testbench/Adder/Full_Adder/genblk1[4]/Add1/xor0/in0_inv
/testbench/Adder/MuxOpB/genblk1[4]/Mux2to1/in1
/testbench/Adder/MuxOpA/genblk1[13]/Mux2to1/Y
/testbench/Adder/Full_Adder/genblk1[3]/Add1/xor1/and_in0_inv
/testbench/Adder/Full_Adder/genblk1[5]/Add1/out_xor0
/testbench/Adder/MuxOpB/genblk1[7]/Mux2to1/in1
/testbench/Adder/MuxOpB/genblk1[14]/Mux2to1/and0
/testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y
/testbench/Adder/Full_Adder/genblk1[3]/Add1/xor0/in0_inv
/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in1
/testbench/Adder/Full_Adder/genblk1[3]/Add1/out_Cout
/testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/in1
/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv
/testbench/Adder/Full_Adder/genblk1[7]/Add1/out_and1
/testbench/Adder/Full_Adder/genblk1[0]/Add1/xor0/out
/testbench/Adder/Full_Adder/Add0/out_xor0
/testbench/Adder/Full_Adder/genblk1[13]/Add1/xor1/and_in1_inv
/testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/S
/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/out
/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1
/testbench/Adder/Full_Adder/Add0/xor0/in1
/testbench/Adder/Full_Adder/genblk1[3]/Add1/A
/testbench/Adder/Full_Adder/genblk1[0]/Add1/xor1/and_in0_inv

Proceso de pruebas para conseguir más de un 90% de cobertura

Transcript de la primera prueba

Compilación -> vlog +define+FAULTS=100 +define+VECTORS=15 +define+MAX=15 +define+REGRESSION=0 testbench.v
#TEST SIN FALLAS
...FAULTS SIMULATION...
#
...First Test without faults...
S0=1 S1=1 8+12=1 Cin=1 Cout=0
S0=1 S1=1 2+5=2 Cin=0 Cout=0
S0=0 S1=1 3+2=6 Cin=1 Cout=0
S0=1 S1=1 6+7=12 Cin=0 Cout=0

```
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# ********************************************************************************
#                               ...The system has been restarted...
# Fault Simulation 1 : /testbench/Adder/Full_Adder/genblk1[1]/Add1/xor0/in1_inv with stuck@1
@175 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=8 Cin=0 Cout=0
# 4 > Fault Detected:    12 first simulation, 8 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[1]/Add1/xor0/in1_inv Released @225 ns
#          4/testbench/Adder/Full_Adder/genblk1[1]/Add1/xor0/in1_inv Released @225 ns
#                               ...The system has been restarted...
# Fault Simulation 2 : /testbench/Adder/Full_Adder/genblk1[4]/Add1/B with stuck@0 @225 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=15 Cin=0 Cout=0
# 9 > Fault Detected:    47 first simulation, 15 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[4]/Add1/B Released @325 ns
#          4/testbench/Adder/Full_Adder/genblk1[4]/Add1/B Released @325 ns
#                               ...The system has been restarted...
# Fault Simulation 3 : /testbench/Adder/MuxOpB/genblk1[12]/Mux2to1/in1 with stuck@1 @325 ns
#
# S0=1 S1=1 8+12=8193 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 8193 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[12]/Mux2to1/in1 Released @345 ns
#          4/testbench/Adder/MuxOpB/genblk1[12]/Mux2to1/in1 Released @345 ns
#                               ...The system has been restarted...
```

```
# Fault Simulation 4 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/out with stuck@1 @345
ns
#
# S0=1 S1=1 8+12=4097 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 4097 current Simulation
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/out Released @365 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/out Released @365 ns
#                                ...The system has been restarted...
# Fault Simulation 5 : /testbench/Adder/Full_Adder/genblk1[5]/Add1/out_Cout with stuck@0 @365
ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[5]/Add1/out_Cout Released @525 ns
#          4/testbench/Adder/Full_Adder/genblk1[5]/Add1/out_Cout Released @525 ns
#                                ...The system has been restarted...
# Fault Simulation 6 : /testbench/Adder/Full_Adder/genblk1[10]/Add1/Cin with stuck@1 @525 ns
#
# S0=1 S1=1 8+12=2049 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 2049 current Simulation
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[10]/Add1/Cin Released @545 ns
#          4/testbench/Adder/Full_Adder/genblk1[10]/Add1/Cin Released @545 ns
#                                ...The system has been restarted...
# Fault Simulation 7 : /testbench/Adder/Full_Adder/genblk1[0]/Add1/xor1/in1 with stuck@1 @545 ns
#
# S0=1 S1=1 8+12=3 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 3 current Simulation
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[0]/Add1/xor1/in1 Released @565 ns
#          4/testbench/Adder/Full_Adder/genblk1[0]/Add1/xor1/in1 Released @565 ns
#                                ...The system has been restarted...
#
```

```
# Fault Simulation 8 : /testbench/Adder/Cout with stuck@1 @565 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=1
# 1 > Fault Detected: Cout = 0 in first simulation, Cout = 1 in current Simulation
#*****************************************************************************#
/testbench/Adder/Cout Released @725 ns
#          4/testbench/Adder/Cout Released @725 ns
#                                ...The system has been restarted...
# Fault Simulation 9 : /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 with stuck@0 @725 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ****************************************************************************
# /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 Released @885 ns
#          4/testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 Released @885 ns
#                                ...The system has been restarted...
# Fault Simulation 10 : /testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/S with stuck@1 @885 ns
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=5 Cin=1 Cout=0
# 3 > Fault Detected:     6 first simulation, 5 current Simulation
# ****************************************************************************
# /testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/S Released @925 ns
#          4/testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/S Released @925 ns
#                                ...The system has been restarted...
# Fault Simulation 11 : /testbench/Adder/MuxOpB/genblk1[13]/Mux2to1/and1 with stuck@1 @925
ns
#
# S0=1 S1=1 8+12=8193 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 8193 current Simulation
# ****************************************************************************
# /testbench/Adder/MuxOpB/genblk1[13]/Mux2to1/and1 Released @945 ns
#          4/testbench/Adder/MuxOpB/genblk1[13]/Mux2to1/and1 Released @945 ns
#                                ...The system has been restarted...
# Fault Simulation 12 : /testbench/Adder/Full_Adder/Add15/xor1/xor_out with stuck@0 @945 ns
```

```
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/Add15/xor1/xor_out Released @1105 ns
#          4/testbench/Adder/Full_Adder/Add15/xor1/xor_out Released @1105 ns
#                               ...The system has been restarted...
# Fault Simulation 13 : /testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/in0 with stuck@0 @1105 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/in0 Released @1265 ns
#          4/testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/in0 Released @1265 ns
#                               ...The system has been restarted...
# Fault Simulation 14 : /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 with stuck@0 @1265
ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
```

```
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 Released @1425 ns
#          4/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 Released @1425 ns
#                               ...The system has been restarted...
# Fault Simulation 15 : /testbench/Adder/Full_Adder/genblk1[10]/Add1/Add with stuck@0 @1425 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[10]/Add1/Add Released @1585 ns
#          4/testbench/Adder/Full_Adder/genblk1[10]/Add1/Add Released @1585 ns
#                               ...The system has been restarted...
# Fault Simulation 16 : /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 with stuck@1 @1585
ns
#
# S0=1 S1=1 8+12=1025 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 1025 current Simulation
#
#
 ********************************************************************************
```

```
# /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 Released @1605 ns
#          4/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 Released @1605 ns
#                                ...The system has been restarted...
#
#
# Fault Simulation 17 : /testbench/Adder/MuxOpA/genblk1[6]/Mux2to1/S with stuck@1 @1605 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=5 Cin=1 Cout=0
# 3 > Fault Detected:     6 first simulation, 5 current Simulation
# ************************************************************************
# /testbench/Adder/MuxOpA/genblk1[6]/Mux2to1/S Released @1645 ns
#          4/testbench/Adder/MuxOpA/genblk1[6]/Mux2to1/S Released @1645 ns
#                                ...The system has been restarted...
# Fault Simulation 18 : /testbench/Adder/Full_Adder/genblk1[6]/Add1/out_and1 with stuck@0
@1645 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[6]/Add1/out_and1 Released @1805 ns
#          4/testbench/Adder/Full_Adder/genblk1[6]/Add1/out_and1 Released @1805 ns
#                                ...The system has been restarted...
# Fault Simulation 19 : /testbench/Adder/Full_Adder/genblk1[1]/Add1/xor1/and_in0_inv with
stuck@0 @1805 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=2 Cin=1 Cout=0
# 3 > Fault Detected:     6 first simulation, 2 current Simulation
#
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[1]/Add1/xor1/and_in0_inv Released @1845 ns
#          4/testbench/Adder/Full_Adder/genblk1[1]/Add1/xor1/and_in0_inv Released @1845 ns
```

```
#                                 ...The system has been restarted...
# Fault Simulation 20 : /testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/S with stuck@1 @1845 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=5 Cin=1 Cout=0
# 3 > Fault Detected:     6 first simulation, 5 current Simulation
# ********************************************************************
# /testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/S Released @1885 ns
#          4/testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/S Released @1885 ns
#                                 ...The system has been restarted...
# Fault Simulation 21 : /testbench/Adder/Full_Adder/genblk1[4]/Add1/xor0/in1 with stuck@1 @1885
ns
#
# S0=1 S1=1 8+12=33 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 33 current Simulation
#
# ********************************************************************
# /testbench/Adder/Full_Adder/genblk1[4]/Add1/xor0/in1 Released @1905 ns
#          4/testbench/Adder/Full_Adder/genblk1[4]/Add1/xor0/in1 Released @1905 ns
#                                 ...The system has been restarted...
# Fault Simulation 22 : /testbench/Adder/Full_Adder/genblk1[10]/Add1/out_add with stuck@1
@1905 ns
#
# S0=1 S1=1 8+12=2049 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 2049 current Simulation
# ********************************************************************
# /testbench/Adder/Full_Adder/genblk1[10]/Add1/out_add Released @1925 ns
#          4/testbench/Adder/Full_Adder/genblk1[10]/Add1/out_add Released @1925 ns
#                                 ...The system has been restarted...
#
# Fault Simulation 23 : /testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/out with stuck@1 @1925
ns
#
# S0=1 S1=1 8+12=129 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 129 current Simulation
#
# ********************************************************************
# /testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/out Released @1945 ns
#          4/testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/out Released @1945 ns
#                                 ...The system has been restarted...
# Fault Simulation 24 : /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/and_in0_inv with
stuck@1 @1945 ns
#
# S0=1 S1=1 8+12=1025 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 1025 current Simulation
#
# ********************************************************************
#
# /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/and_in0_inv Released @1965 ns
#          4/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/and_in0_inv Released @1965 ns
```

```
#                                 ...The system has been restarted...
# Fault Simulation 25 : /testbench/Adder/Full_Adder/Add0/xor0/in1_inv with stuck@0 @1965 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=5 Cin=1 Cout=0
# 3 > Fault Detected:     6 first simulation, 5 current Simulation
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/Add0/xor0/in1_inv Released @2005 ns
#          4/testbench/Adder/Full_Adder/Add0/xor0/in1_inv Released @2005 ns
#                                 ...The system has been restarted...
#
# Fault Simulation 26 : /testbench/Adder/Full_Adder/Add0/xor0/out with stuck@1 @2005 ns
#
# S0=1 S1=1 8+12=2 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 2 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/Add0/xor0/out Released @2025 ns
#          4/testbench/Adder/Full_Adder/Add0/xor0/out Released @2025 ns
#                                 ...The system has been restarted...
# Fault Simulation 27 : /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/out_Y with stuck@1 @2025
ns
#
# S0=1 S1=1 8+12=4097 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 4097 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/out_Y Released @2045 ns
#          4/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/out_Y Released @2045 ns
#                                 ...The system has been restarted...
# Fault Simulation 28 : /testbench/Adder/Full_Adder/genblk1[10]/Add1/out_and1 with stuck@1
@2045 ns
#
# S0=1 S1=1 8+12=4097 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 4097 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[10]/Add1/out_and1 Released @2065 ns
#          4/testbench/Adder/Full_Adder/genblk1[10]/Add1/out_and1 Released @2065 ns
#                                 ...The system has been restarted...
# Fault Simulation 29 : /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv with stuck@0
@2065 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
```

```
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv Released @2225 ns
#          4/testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv Released @2225 ns
#                               ...The system has been restarted...
# Fault Simulation 30 : /testbench/Adder/MuxOpA/genblk1[9]/Mux2to1/S1 with stuck@0 @2225 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[9]/Mux2to1/S1 Released @2385 ns
#          4/testbench/Adder/MuxOpA/genblk1[9]/Mux2to1/S1 Released @2385 ns
#                               ...The system has been restarted...
# Fault Simulation 31 : /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/out with stuck@0 @2385
ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
```

```
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ****************************************************************************
# /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/out Released @2545 ns
#          4/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/out Released @2545 ns
#                                ...The system has been restarted...
# Fault Simulation 32 : /testbench/Adder/MuxOpB/genblk1[4]/Mux2to1/Y with stuck@0 @2545 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=31 Cin=1 Cout=0
# 6 > Fault Detected:    47 first simulation, 31 current Simulation
# ****************************************************************************
# /testbench/Adder/MuxOpB/genblk1[4]/Mux2to1/Y Released @2615 ns
#          4/testbench/Adder/MuxOpB/genblk1[4]/Mux2to1/Y Released @2615 ns
#                                ...The system has been restarted...
# Fault Simulation 33 : /testbench/Adder/MuxOpA/genblk1[4]/Mux2to1/out_Y with stuck@0 @2615
ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=31 Cin=1 Cout=0
# 6 > Fault Detected:    47 first simulation, 31 current Simulation
# ****************************************************************************
# /testbench/Adder/MuxOpA/genblk1[4]/Mux2to1/out_Y Released @2685 ns
#          4/testbench/Adder/MuxOpA/genblk1[4]/Mux2to1/out_Y Released @2685 ns
#                                ...The system has been restarted...
# Fault Simulation 34 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv with
stuck@0 @2685 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
```

```
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv Released @2845 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv Released @2845 ns
#                                ...The system has been restarted...
# Fault Simulation 35 : /testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y with stuck@0 @2845
ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y Released @3005 ns
#          4/testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y Released @3005 ns
#                                ...The system has been restarted...
# Fault Simulation 36 : /testbench/Adder/Full_Adder/Add0/xor1/in0_inv with stuck@1 @3005 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=7 Cin=1 Cout=0
# 3 > Fault Detected:     6 first simulation, 7 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/Add0/xor1/in0_inv Released @3045 ns
#          4/testbench/Adder/Full_Adder/Add0/xor1/in0_inv Released @3045 ns
#                                ...The system has been restarted...
# Fault Simulation 37 : /testbench/Adder/Full_Adder/genblk1[0]/Add1/out_xor0 with stuck@1 @3045
ns
#
# S0=1 S1=1 8+12=3 Cin=1 Cout=0
```

```
# 1 > Fault Detected:     1 first simulation, 3 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[0]/Add1/out_xor0 Released @3065 ns
#          4/testbench/Adder/Full_Adder/genblk1[0]/Add1/out_xor0 Released @3065 ns
#                              ...The system has been restarted...
# Fault Simulation 38 : /testbench/Adder/ResultSyncA with stuck@1 @3065 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=1 Cin=0 Cout=0
# 2 > Fault Detected:     2 first simulation, 1 current Simulation
# ********************************************************************************
# /testbench/Adder/ResultSyncA Released @3095 ns
#          4/testbench/Adder/ResultSyncA Released @3095 ns
#                              ...The system has been restarted...
# Fault Simulation 39 : /testbench/Adder/Full_Adder/genblk1[4]/Add1/B with stuck@1 @3095 ns
#
# S0=1 S1=1 8+12=33 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 33 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[4]/Add1/B Released @3115 ns
#          4/testbench/Adder/Full_Adder/genblk1[4]/Add1/B Released @3115 ns
#                              ...The system has been restarted...
# Fault Simulation 40 : /testbench/Adder/Full_Adder/genblk1[13]/Add1/xor0/in1 with stuck@1
@3115 ns
#
# S0=1 S1=1 8+12=16385 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 16385 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[13]/Add1/xor0/in1 Released @3135 ns
#          4/testbench/Adder/Full_Adder/genblk1[13]/Add1/xor0/in1 Released @3135 ns
#                              ...The system has been restarted...
# Fault Simulation 41 : /testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/in1 with stuck@0 @3135 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=15 Cin=0 Cout=0
# 9 > Fault Detected:     47 first simulation, 15 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/in1 Released @3235 ns
#          4/testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/in1 Released @3235 ns
#                              ...The system has been restarted...
# Fault Simulation 42 : /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/and0 with stuck@1 @3235
ns
# S0=1 S1=1 8+12=4097 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 4097 current Simulation
```

```
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/and0 Released @3255 ns
#          4/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/and0 Released @3255 ns
#                                ...The system has been restarted...
# Fault Simulation 43 : /testbench/Adder/MuxOpB/genblk1[10]/Mux2to1/S1 with stuck@1 @3255 ns
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[10]/Mux2to1/S1 Released @3415 ns
#          4/testbench/Adder/MuxOpB/genblk1[10]/Mux2to1/S1 Released @3415 ns
#                                ...The system has been restarted...
# Fault Simulation 44 : /testbench/Adder/Full_Adder/Cout with stuck@0 @3415 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/Cout Released @3575 ns
#          4/testbench/Adder/Full_Adder/Cout Released @3575 ns
#                                ...The system has been restarted...
```

```
# Fault Simulation 45 : /testbench/Adder/Full_Adder/genblk1[1]/Add1/out_and0 with stuck@0 @3575 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=4 Cin=0 Cout=0
# 4 > Fault Detected:    12 first simulation, 4 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[1]/Add1/out_and0 Released @3625 ns
#          4/testbench/Adder/Full_Adder/genblk1[1]/Add1/out_and0 Released @3625 ns
#                                ...The system has been restarted...
# Fault Simulation 46 : /testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/and_in0_inv with stuck@1 @3625 ns
#
# S0=1 S1=1 8+12=129 Cin=1 Cout=0
# 1 > Fault Detected:    1 first simulation, 129 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/and_in0_inv Released @3645 ns
#          4/testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/and_in0_inv Released @3645 ns
#                                ...The system has been restarted...
# Fault Simulation 47 : /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor1/out with stuck@1 @3645 ns
#
# S0=1 S1=1 8+12=257 Cin=1 Cout=0
# 1 > Fault Detected:    1 first simulation, 257 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor1/out Released @3665 ns
#          4/testbench/Adder/Full_Adder/genblk1[7]/Add1/xor1/out Released @3665 ns
#                                ...The system has been restarted...
# Fault Simulation 48 : /testbench/Adder/Full_Adder/genblk1[6]/Add1/out_add with stuck@0 @3665 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
```

```
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[6]/Add1/out_add Released @3825 ns
#          4/testbench/Adder/Full_Adder/genblk1[6]/Add1/out_add Released @3825 ns
#                                ...The system has been restarted...
# Fault Simulation 49 : /testbench/Adder/MuxOpB/genblk1[1]/Mux2to1/S1 with stuck@1 @3825 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[1]/Mux2to1/S1 Released @3985 ns
#          4/testbench/Adder/MuxOpB/genblk1[1]/Mux2to1/S1 Released @3985 ns
#                                ...The system has been restarted...
#
# Fault Simulation 50 : /testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/S1 with stuck@1 @3985 ns
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=55 Cin=0 Cout=0
# 7 > Fault Detected:    23 first simulation, 55 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/S1 Released @4065 ns
#          4/testbench/Adder/MuxOpA/genblk1[5]/Mux2to1/S1 Released @4065 ns
#                                ...The system has been restarted...
# Fault Simulation 51 : /testbench/Adder/Full_Adder/genblk1[10]/Add1/out_Cout with stuck@1
@4065 ns
#
# S0=1 S1=1 8+12=4097 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 4097 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[10]/Add1/out_Cout Released @4085 ns
#          4/testbench/Adder/Full_Adder/genblk1[10]/Add1/out_Cout Released @4085 ns
#                                ...The system has been restarted...
```

```
# Fault Simulation 52 : /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 with stuck@1 @4085
ns
# S0=1 S1=1 8+12=32769 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 32769 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 Released @4105 ns
#          4/testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 Released @4105 ns
#                                ...The system has been restarted...
# Fault Simulation 53 : /testbench/Adder/MuxOpA/genblk1[3]/Mux2to1/S with stuck@0 @4105 ns
# S0=1 S1=1 8+12=9 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 9 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[3]/Mux2to1/S Released @4125 ns
#          4/testbench/Adder/MuxOpA/genblk1[3]/Mux2to1/S Released @4125 ns
#                                ...The system has been restarted...
# Fault Simulation 54 : /testbench/Adder/Full_Adder/genblk1[8]/Add1/xor1/out with stuck@0 @4125
ns
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[8]/Add1/xor1/out Released @4285 ns
#          4/testbench/Adder/Full_Adder/genblk1[8]/Add1/xor1/out Released @4285 ns
#                                ...The system has been restarted...
# Fault Simulation 55 : /testbench/Adder/MuxOpA/genblk1[1]/Mux2to1/S0 with stuck@0 @4285 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=21 Cin=0 Cout=0
# 5 > Fault Detected:     23 first simulation, 21 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[1]/Mux2to1/S0 Released @4345 ns
#          4/testbench/Adder/MuxOpA/genblk1[1]/Mux2to1/S0 Released @4345 ns
#                                ...The system has been restarted...
```

```
# Fault Simulation 56 : /testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/out with stuck@1 @4345 ns
#
# S0=1 S1=1 8+12=129 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 129 current Simulation
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/out Released @4365 ns
#          4/testbench/Adder/Full_Adder/genblk1[6]/Add1/xor0/out Released @4365 ns
#                                ...The system has been restarted...
# Fault Simulation 57 : /testbench/Adder/MuxOpA/genblk1[2]/Mux2to1/and1 with stuck@0 @4365 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=8 Cin=0 Cout=0
# 4 > Fault Detected:     12 first simulation, 8 current Simulation
# ************************************************************************
# /testbench/Adder/MuxOpA/genblk1[2]/Mux2to1/and1 Released @4415 ns
#          4/testbench/Adder/MuxOpA/genblk1[2]/Mux2to1/and1 Released @4415 ns
#                                ...The system has been restarted...
# Fault Simulation 58 : /testbench/Adder/MuxOpA/genblk1[3]/Mux2to1/S with stuck@0 @4415 ns
#
# S0=1 S1=1 8+12=9 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 9 current Simulation
# ************************************************************************
# /testbench/Adder/MuxOpA/genblk1[3]/Mux2to1/S Released @4435 ns
#          4/testbench/Adder/MuxOpA/genblk1[3]/Mux2to1/S Released @4435 ns
#                                ...The system has been restarted...
# Fault Simulation 59 : /testbench/Adder/MuxOpB/Y with stuck@0 @4435 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=1 Cin=0 Cout=0
# 2 > Fault Detected:     2 first simulation, 1 current Simulation
# ************************************************************************
# /testbench/Adder/MuxOpB/Y Released @4465 ns
#          4/testbench/Adder/MuxOpB/Y Released @4465 ns
#                                ...The system has been restarted...
# Fault Simulation 60 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv with stuck@1 @4465 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
```

```
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv Released @4625 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv Released @4625 ns
#                                ...The system has been restarted...
# Fault Simulation 61 : /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/out_Y with stuck@1 @4625
ns
#
# S0=1 S1=1 8+12=32769 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 32769 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/out_Y Released @4645 ns
#          4/testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/out_Y Released @4645 ns
#                                ...The system has been restarted...
# Fault Simulation 62 : /testbench/Adder/MuxOpB/genblk1[0]/Mux2to1/S with stuck@0 @4645 ns
#
# S0=1 S1=1 8+12=13 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 13 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[0]/Mux2to1/S Released @4665 ns
#          4/testbench/Adder/MuxOpB/genblk1[0]/Mux2to1/S Released @4665 ns
#                                ...The system has been restarted...
# Fault Simulation 63 : /testbench/Adder/Full_Adder/genblk1[8]/Add1/xor0/in0_inv with stuck@0
@4665 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[8]/Add1/xor0/in0_inv Released @4825 ns
```

```
#          4/testbench/Adder/Full_Adder/genblk1[8]/Add1/xor0/in0_inv Released @4825 ns
#                                ...The system has been restarted...
# Fault Simulation 64 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/out_xor0 with stuck@1 @4825 ns
#
# S0=1 S1=1 8+12=4097 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 4097 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/out_xor0 Released @4845 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/out_xor0 Released @4845 ns
#                                ...The system has been restarted...
# Fault Simulation 65 : /testbench/Adder/Full_Adder/genblk1[5]/Add1/xor0/in0 with stuck@0 @4845 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[5]/Add1/xor0/in0 Released @5005 ns
#          4/testbench/Adder/Full_Adder/genblk1[5]/Add1/xor0/in0 Released @5005 ns
#                                ...The system has been restarted...
# Fault Simulation 66 : /testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/and1 with stuck@0 @5005 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
```

```
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/and1 Released @5165 ns
#          4/testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/and1 Released @5165 ns
#                                  ...The system has been restarted...
# Fault Simulation 67 : /testbench/Adder/MuxOpB/genblk1[0]/Mux2to1/Y with stuck@0 @5165 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=1 Cin=0 Cout=0
# 2 > Fault Detected:     2 first simulation, 1 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[0]/Mux2to1/Y Released @5195 ns
#          4/testbench/Adder/MuxOpB/genblk1[0]/Mux2to1/Y Released @5195 ns
#                                  ...The system has been restarted...
# Fault Simulation 68 : /testbench/Adder/MuxOpB/genblk1[2]/Mux2to1/out_Y with stuck@1 @5195
ns
#
# S0=1 S1=1 8+12=5 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 5 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[2]/Mux2to1/out_Y Released @5215 ns
#          4/testbench/Adder/MuxOpB/genblk1[2]/Mux2to1/out_Y Released @5215 ns
#                                  ...The system has been restarted...
# Fault Simulation 69 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in1_inv with
stuck@0 @5215 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                  ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in1_inv Released @5375 ns
```

```
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in1_inv Released @5375 ns
#                                ...The system has been restarted...
# Fault Simulation 70 : /testbench/Adder/MuxOpB/genblk1[12]/Mux2to1/S with stuck@0 @5375 ns
#
# S0=1 S1=1 8+12=13 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 13 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[12]/Mux2to1/S Released @5395 ns
#          4/testbench/Adder/MuxOpB/genblk1[12]/Mux2to1/S Released @5395 ns
#                                ...The system has been restarted...
# Fault Simulation 71 : /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 with stuck@1 @5395 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 Released @5555 ns
#          4/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 Released @5555 ns
#                                ...The system has been restarted...
# Fault Simulation 72 : /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in1 with stuck@0 @5555
ns
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
```

```
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ****************************************************************************
# /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in1 Released @5715 ns
#          4/testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in1 Released @5715 ns
#                                ...The system has been restarted...
# Fault Simulation 73 : /testbench/Adder/MuxOpA/genblk1[15]/Mux2to1/Y with stuck@1 @5715 ns
#
# S0=1 S1=1 8+12=32769 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 32769 current Simulation
# ****************************************************************************
# /testbench/Adder/MuxOpA/genblk1[15]/Mux2to1/Y Released @5735 ns
#          4/testbench/Adder/MuxOpA/genblk1[15]/Mux2to1/Y Released @5735 ns
#                                ...The system has been restarted...
# Fault Simulation 74 : /testbench/Adder/Full_Adder/Add0/out_xor0 with stuck@1 @5735 ns
#
# S0=1 S1=1 8+12=2 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 2 current Simulation
# ****************************************************************************
# /testbench/Adder/Full_Adder/Add0/out_xor0 Released @5755 ns
#          4/testbench/Adder/Full_Adder/Add0/out_xor0 Released @5755 ns
#                                ...The system has been restarted...
# Fault Simulation 75 : /testbench/Adder/Full_Adder/genblk1[12]/Add1/xor1/and_in1_inv with
stuck@1 @5755 ns
#
# S0=1 S1=1 8+12=8193 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 8193 current Simulation
# ****************************************************************************
# /testbench/Adder/Full_Adder/genblk1[12]/Add1/xor1/and_in1_inv Released @5775 ns
#          4/testbench/Adder/Full_Adder/genblk1[12]/Add1/xor1/and_in1_inv Released @5775 ns
#                                ...The system has been restarted...
# Fault Simulation 76 : /testbench/Adder/MuxOpA/genblk1[15]/Mux2to1/in1 with stuck@1 @5775 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=1
# S0=1 S1=1 2+5=2 Cin=0 Cout=1
# S0=0 S1=1 3+2=32774 Cin=1 Cout=0
# 3 > Fault Detected:     6 first simulation, 32774 current Simulation
# ****************************************************************************
# /testbench/Adder/MuxOpA/genblk1[15]/Mux2to1/in1 Released @5815 ns
#          4/testbench/Adder/MuxOpA/genblk1[15]/Mux2to1/in1 Released @5815 ns
#                                ...The system has been restarted...
# Fault Simulation 77 : /testbench/Adder/Full_Adder/genblk1[3]/Add1/out_and0 with stuck@1
@5815 ns
#
# S0=1 S1=1 8+12=33 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 33 current Simulation
# ****************************************************************************
# /testbench/Adder/Full_Adder/genblk1[3]/Add1/out_and0 Released @5835 ns
#          4/testbench/Adder/Full_Adder/genblk1[3]/Add1/out_and0 Released @5835 ns
#                                ...The system has been restarted...
```

```
# Fault Simulation 78 : /testbench/Adder/Full_Adder/genblk1[4]/Add1/xor0/in0_inv with stuck@0
@5835 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=15 Cin=0 Cout=0
# 9 > Fault Detected:    47 first simulation, 15 current Simulation
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[4]/Add1/xor0/in0_inv Released @5935 ns
#          4/testbench/Adder/Full_Adder/genblk1[4]/Add1/xor0/in0_inv Released @5935 ns
#                                ...The system has been restarted...
# Fault Simulation 79 : /testbench/Adder/MuxOpB/genblk1[4]/Mux2to1/in1 with stuck@0 @5935 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=15 Cin=1 Cout=0
# 6 > Fault Detected:    47 first simulation, 15 current Simulation
# ************************************************************************
# /testbench/Adder/MuxOpB/genblk1[4]/Mux2to1/in1 Released @6005 ns
#          4/testbench/Adder/MuxOpB/genblk1[4]/Mux2to1/in1 Released @6005 ns
#                                ...The system has been restarted...
# Fault Simulation 80 : /testbench/Adder/MuxOpA/genblk1[13]/Mux2to1/Y with stuck@0 @6005 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ************************************************************************
```

```
# /testbench/Adder/MuxOpA/genblk1[13]/Mux2to1/Y Released @6165 ns
#          4/testbench/Adder/MuxOpA/genblk1[13]/Mux2to1/Y Released @6165 ns
#                                ...The system has been restarted...
# Fault Simulation 81 : /testbench/Adder/Full_Adder/genblk1[3]/Add1/xor1/and_in0_inv with stuck@1 @6165 ns
#
# S0=1 S1=1 8+12=17 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 17 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[3]/Add1/xor1/and_in0_inv Released @6185 ns
#          4/testbench/Adder/Full_Adder/genblk1[3]/Add1/xor1/and_in0_inv Released @6185 ns
#                                ...The system has been restarted...
# Fault Simulation 82 : /testbench/Adder/Full_Adder/genblk1[5]/Add1/out_xor0 with stuck@0 @6185 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
#
# /testbench/Adder/Full_Adder/genblk1[5]/Add1/out_xor0 Released @6345 ns
#          4/testbench/Adder/Full_Adder/genblk1[5]/Add1/out_xor0 Released @6345 ns
#                                ...The system has been restarted...
# Fault Simulation 83 : /testbench/Adder/MuxOpB/genblk1[7]/Mux2to1/in1 with stuck@0 @6345 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
```

```
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[7]/Mux2to1/in1 Released @6505 ns
#          4/testbench/Adder/MuxOpB/genblk1[7]/Mux2to1/in1 Released @6505 ns
#                                  ...The system has been restarted...
#
# Fault Simulation 84 : /testbench/Adder/MuxOpB/genblk1[14]/Mux2to1/and0 with stuck@1 @6505
ns
#
# S0=1 S1=1 8+12=16385 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 16385 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[14]/Mux2to1/and0 Released @6525 ns
#          4/testbench/Adder/MuxOpB/genblk1[14]/Mux2to1/and0 Released @6525 ns
#                                  ...The system has been restarted...
#
#
# Fault Simulation 85 : /testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y with stuck@1 @6525
ns
#
# S0=1 S1=1 8+12=513 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 513 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y Released @6545 ns
#          4/testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y Released @6545 ns
#                                  ...The system has been restarted...
# Fault Simulation 86 : /testbench/Adder/Full_Adder/genblk1[3]/Add1/xor0/in0_inv with stuck@0
@6545 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                  ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
```

```
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[3]/Add1/xor0/in0_inv Released @6705 ns
#          4/testbench/Adder/Full_Adder/genblk1[3]/Add1/xor0/in0_inv Released @6705 ns
#                              ...The system has been restarted...
# Fault Simulation 87 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in1 with stuck@0
@6705 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                              ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in1 Released @6865 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in1 Released @6865 ns
#                              ...The system has been restarted...
# Fault Simulation 88 : /testbench/Adder/Full_Adder/genblk1[3]/Add1/out_Cout with stuck@0
@6865 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=15 Cin=1 Cout=0
# 6 > Fault Detected:    47 first simulation, 15 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[3]/Add1/out_Cout Released @6935 ns
#          4/testbench/Adder/Full_Adder/genblk1[3]/Add1/out_Cout Released @6935 ns
#                              ...The system has been restarted...
# Fault Simulation 89 : /testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/in1 with stuck@1 @6935 ns
#
# S0=1 S1=1 8+12=4097 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 4097 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/in1 Released @6955 ns
```

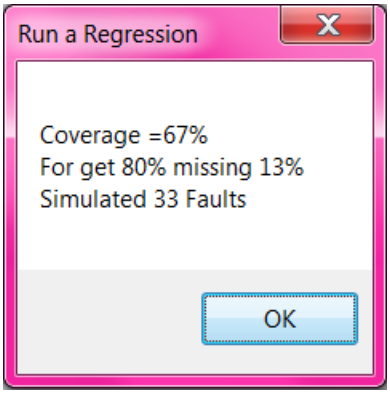
```
#          4/testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/in1 Released @6955 ns
#                                ...The system has been restarted...
# Fault Simulation 90 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv with stuck@1 @6955 ns
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv Released @7115 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv Released @7115 ns
#                                ...The system has been restarted...
# Fault Simulation 91 : /testbench/Adder/Full_Adder/genblk1[7]/Add1/out_and1 with stuck@0 @7115 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[7]/Add1/out_and1 Released @7275 ns
#          4/testbench/Adder/Full_Adder/genblk1[7]/Add1/out_and1 Released @7275 ns
#                                ...The system has been restarted...
```

Fault Simulation 92 : /testbench/Adder/Full_Adder/genblk1[0]/Add1/xor0/out with stuck@0 @7275 ns
#
S0=1 S1=1 8+12=1 Cin=1 Cout=0
S0=1 S1=1 2+5=2 Cin=0 Cout=0
S0=0 S1=1 3+2=6 Cin=1 Cout=0
S0=1 S1=1 6+7=12 Cin=0 Cout=0
S0=0 S1=0 14+9=21 Cin=0 Cout=0
5 > Fault Detected: 23 first simulation, 21 current Simulation
**
/testbench/Adder/Full_Adder/genblk1[0]/Add1/xor0/out Released @7335 ns
4/testbench/Adder/Full_Adder/genblk1[0]/Add1/xor0/out Released @7335 ns
...The system has been restarted...
Fault Simulation 93 : /testbench/Adder/Full_Adder/Add0/out_xor0 with stuck@0 @7335 ns
#
S0=1 S1=1 8+12=1 Cin=1 Cout=0
S0=1 S1=1 2+5=2 Cin=0 Cout=0
S0=0 S1=1 3+2=5 Cin=1 Cout=0
3 > Fault Detected: 6 first simulation, 5 current Simulation
**
/testbench/Adder/Full_Adder/Add0/out_xor0 Released @7375 ns
4/testbench/Adder/Full_Adder/Add0/out_xor0 Released @7375 ns
...The system has been restarted...
#
#
#
Fault Simulation 94 : /testbench/Adder/Full_Adder/genblk1[13]/Add1/xor1/and_in1_inv with stuck@1 @7375 ns
#
S0=1 S1=1 8+12=16385 Cin=1 Cout=0
1 > Fault Detected: 1 first simulation, 16385 current Simulation
**
/testbench/Adder/Full_Adder/genblk1[13]/Add1/xor1/and_in1_inv Released @7395 ns
4/testbench/Adder/Full_Adder/genblk1[13]/Add1/xor1/and_in1_inv Released @7395 ns
...The system has been restarted...
Fault Simulation 95 : /testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/S with stuck@1 @7395 ns
#
S0=1 S1=1 8+12=1 Cin=1 Cout=0
S0=1 S1=1 2+5=2 Cin=0 Cout=0
S0=0 S1=1 3+2=5 Cin=1 Cout=0
3 > Fault Detected: 6 first simulation, 5 current Simulation
**
/testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/S Released @7435 ns
4/testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/S Released @7435 ns
...The system has been restarted...
Fault Simulation 96 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/out with stuck@0 @7435 ns
#
S0=1 S1=1 8+12=1 Cin=1 Cout=0
S0=1 S1=1 2+5=2 Cin=0 Cout=0
S0=0 S1=1 3+2=6 Cin=1 Cout=0

```
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/out Released @7595 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/out Released @7595 ns
#                               ...The system has been restarted...
# Fault Simulation 97 : /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 with stuck@0 @7595 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=2 Cin=0 Cout=0
# S0=0 S1=1 3+2=6 Cin=1 Cout=0
# S0=1 S1=1 6+7=12 Cin=0 Cout=0
# S0=0 S1=0 14+9=23 Cin=0 Cout=0
# S0=1 S1=1 13+14=47 Cin=1 Cout=0
# S0=0 S1=0 13+10=23 Cin=0 Cout=0
# S0=1 S1=1 6+12=47 Cin=1 Cout=0
# S0=0 S1=1 0+10=47 Cin=0 Cout=0
# S0=0 S1=0 14+2=16 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 11+5=17 Cin=1 Cout=0
# S0=0 S1=0 9+7=16 Cin=0 Cout=0
# S0=1 S1=1 4+14=33 Cin=1 Cout=0
# S0=0 S1=1 10+6=43 Cin=0 Cout=0
# S0=0 S1=1 13+1=56 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 Released @7755 ns
#          4/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 Released @7755 ns
#                               ...The system has been restarted...
# Fault Simulation 98 : /testbench/Adder/Full_Adder/Add0/xor0/in1 with stuck@0 @7755 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=1 Cin=0 Cout=0
# 2 > Fault Detected:     2 first simulation, 1 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/Add0/xor0/in1 Released @7785 ns
#          4/testbench/Adder/Full_Adder/Add0/xor0/in1 Released @7785 ns
```

```
#                                  ...The system has been restarted...
# Fault Simulation 99 : /testbench/Adder/Full_Adder/genblk1[3]/Add1/A with stuck@1 @7785 ns
#
# S0=1 S1=1 8+12=17 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 17 current Simulation
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[3]/Add1/A Released @7805 ns
#          4/testbench/Adder/Full_Adder/genblk1[3]/Add1/A Released @7805 ns
#                                  ...The system has been restarted...
# Fault Simulation 100 : /testbench/Adder/Full_Adder/genblk1[0]/Add1/xor1/and_in0_inv with stuck@0 @7805 ns
#
# S0=1 S1=1 8+12=1 Cin=1 Cout=0
# S0=1 S1=1 2+5=0 Cin=0 Cout=0
# 2 > Fault Detected:     2 first simulation, 0 current Simulation
# ...The faults simulation has finished. Sytem has been restored...
# Coverage=67
```

Corriendo la herramienta "UpdateTest.exe" para actualizar el log con las fallas que no han sido detectadas y darnos los resultados finales se obtuvo lo siguiente, (d) Donde se indica que faltan 33 fallas por detectar:



Script para el cálculo de la cobertura

Transcript de la segunda prueba para tratar de cubrir mínimo 13% más:

Compilación -> vlog +define+FAULTS=34 +define+VECTORS=15 +define+MAX=150 +define+REGRESSION=1 testbench.v

```
#                                  ...First Test without faults...
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
```

```
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# ********************************************************************************
#                               ...The system has been restarted...
# Fault Simulation 1 : /testbench/Adder/Full_Adder/genblk1[5]/Add1/out_Cout with stuck@0 @175
ns
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=39 Cin=1 Cout=0
# 6 > Fault Detected:   167 first simulation, 39 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[5]/Add1/out_Cout Released @245 ns
#         4/testbench/Adder/Full_Adder/genblk1[5]/Add1/out_Cout Released @245 ns
#                               ...The system has been restarted...
# Fault Simulation 2 : /testbench/Adder/Cout with stuck@1 @245 ns
# S0=1 S1=1 98+147=1 Cin=1 Cout=1
# S0=1 S1=1 107+125=2 Cin=0 Cout=1
# S0=0 S1=1 18+47=21 Cin=1 Cout=1
# S0=1 S1=1 126+97=42 Cin=0 Cout=1
# S0=0 S1=0 44+39=83 Cin=0 Cout=1
# S0=1 S1=1 103+134=167 Cin=1 Cout=1
# S0=0 S1=0 73+40=113 Cin=0 Cout=1
# S0=1 S1=1 21+72=227 Cin=1 Cout=1
# S0=0 S1=1 105+85=332 Cin=0 Cout=1
# S0=0 S1=0 29+47=76 Cin=0 Cout=1
#                               ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=1
# S0=0 S1=0 99+127=226 Cin=0 Cout=1
# S0=1 S1=1 64+44=453 Cin=1 Cout=1
# S0=0 S1=1 25+141=478 Cin=0 Cout=1
# S0=0 S1=1 103+61=581 Cin=0 Cout=1
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Cout Released @405 ns
#         4/testbench/Adder/Cout Released @405 ns
#                               ...The system has been restarted...
# Fault Simulation 3 : /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 with stuck@0 @405 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
```

```
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 Released @565 ns
#          4/testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 Released @565 ns
#                               ...The system has been restarted...
# Fault Simulation 4 : /testbench/Adder/Full_Adder/Add15/xor1/xor_out with stuck@0 @565 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/Add15/xor1/xor_out Released @725 ns
#          4/testbench/Adder/Full_Adder/Add15/xor1/xor_out Released @725 ns
#                               ...The system has been restarted...
# Fault Simulation 5 : /testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/in0 with stuck@0 @725 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
```

```
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                              ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/in0 Released @885 ns
#          4/testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/in0 Released @885 ns
#                              ...The system has been restarted...
# Fault Simulation 6 : /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 with stuck@0 @885 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                              ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 Released @1045 ns
#          4/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 Released @1045 ns
#                              ...The system has been restarted...
# Fault Simulation 7 : /testbench/Adder/Full_Adder/genblk1[10]/Add1/Add with stuck@0 @1045 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
```

```
#                                ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ****************************************************************************
# /testbench/Adder/Full_Adder/genblk1[10]/Add1/Add Released @1205 ns
#          4/testbench/Adder/Full_Adder/genblk1[10]/Add1/Add Released @1205 ns
#                                ...The system has been restarted...
# Fault Simulation 8 : /testbench/Adder/Full_Adder/genblk1[6]/Add1/out_and1 with stuck@0 @1205
ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=76 Cin=0 Cout=0
# 9 > Fault Detected:   332 first simulation, 76 current Simulation
# ****************************************************************************
# /testbench/Adder/Full_Adder/genblk1[6]/Add1/out_and1 Released @1305 ns
#          4/testbench/Adder/Full_Adder/genblk1[6]/Add1/out_and1 Released @1305 ns
#                                ...The system has been restarted...
# Fault Simulation 9 : /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv with stuck@0
@1305 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=222 Cin=0 Cout=0
# 14 > Fault Detected:   478 first simulation, 222 current Simulation
# ****************************************************************************
# /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv Released @1455 ns
#          4/testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv Released @1455 ns
```

```
#                                 ...The system has been restarted...
# Fault Simulation 10 : /testbench/Adder/MuxOpA/genblk1[9]/Mux2to1/S1 with stuck@0 @1455 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                                 ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[9]/Mux2to1/S1 Released @1615 ns
#          4/testbench/Adder/MuxOpA/genblk1[9]/Mux2to1/S1 Released @1615 ns
#                                 ...The system has been restarted...
# Fault Simulation 11 : /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/out with stuck@0 @1615
ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                                 ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/out Released @1775 ns
#          4/testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/out Released @1775 ns
#                                 ...The system has been restarted...
```

```
# Fault Simulation 12 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv with stuck@0 @1775 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
#
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv Released @1935 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv Released @1935 ns
#                               ...The system has been restarted...
# Fault Simulation 13 : /testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y with stuck@0 @1935 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                               ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
#
# /testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y Released @2095 ns
#          4/testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y Released @2095 ns
```

```
#                                 ...The system has been restarted...
#
#
# Fault Simulation 14 : /testbench/Adder/MuxOpB/genblk1[10]/Mux2to1/S1 with stuck@1 @2095 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                                 ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpB/genblk1[10]/Mux2to1/S1 Released @2255 ns
#          4/testbench/Adder/MuxOpB/genblk1[10]/Mux2to1/S1 Released @2255 ns
#                                 ...The system has been restarted...
# Fault Simulation 15 : /testbench/Adder/Full_Adder/Cout with stuck@0 @2255 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                                 ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/Cout Released @2415 ns
#          4/testbench/Adder/Full_Adder/Cout Released @2415 ns
#                                 ...The system has been restarted...
```

Fault Simulation 16 : /testbench/Adder/Full_Adder/genblk1[6]/Add1/out_add with stuck@0 @2415 ns
#
S0=1 S1=1 98+147=1 Cin=1 Cout=0
S0=1 S1=1 107+125=2 Cin=0 Cout=0
S0=0 S1=1 18+47=21 Cin=1 Cout=0
S0=1 S1=1 126+97=42 Cin=0 Cout=0
S0=0 S1=0 44+39=83 Cin=0 Cout=0
S0=1 S1=1 103+134=39 Cin=1 Cout=0
6 > Fault Detected: 167 first simulation, 39 current Simulation
**
/testbench/Adder/Full_Adder/genblk1[6]/Add1/out_add Released @2485 ns
4/testbench/Adder/Full_Adder/genblk1[6]/Add1/out_add Released @2485 ns
...The system has been restarted...
Fault Simulation 17 : /testbench/Adder/MuxOpB/genblk1[1]/Mux2to1/S1 with stuck@1 @2485 ns
#
S0=1 S1=1 98+147=1 Cin=1 Cout=0
S0=1 S1=1 107+125=2 Cin=0 Cout=0
S0=0 S1=1 18+47=21 Cin=1 Cout=0
S0=1 S1=1 126+97=42 Cin=0 Cout=0
S0=0 S1=0 44+39=83 Cin=0 Cout=0
S0=1 S1=1 103+134=167 Cin=1 Cout=0
S0=0 S1=0 73+40=115 Cin=0 Cout=0
7 > Fault Detected: 113 first simulation, 115 current Simulation
**
/testbench/Adder/MuxOpB/genblk1[1]/Mux2to1/S1 Released @2565 ns
4/testbench/Adder/MuxOpB/genblk1[1]/Mux2to1/S1 Released @2565 ns
...The system has been restarted...
Fault Simulation 18 : /testbench/Adder/Full_Adder/genblk1[8]/Add1/xor1/out with stuck@0 @2565 ns
#
S0=1 S1=1 98+147=1 Cin=1 Cout=0
S0=1 S1=1 107+125=2 Cin=0 Cout=0
S0=0 S1=1 18+47=21 Cin=1 Cout=0
S0=1 S1=1 126+97=42 Cin=0 Cout=0
S0=0 S1=0 44+39=83 Cin=0 Cout=0
S0=1 S1=1 103+134=167 Cin=1 Cout=0
S0=0 S1=0 73+40=113 Cin=0 Cout=0
S0=1 S1=1 21+72=227 Cin=1 Cout=0
S0=0 S1=1 105+85=332 Cin=0 Cout=0
S0=0 S1=0 29+47=76 Cin=0 Cout=0
...Simulated 10 vectors...
S0=0 S1=0 56+80=137 Cin=1 Cout=0
S0=0 S1=0 99+127=226 Cin=0 Cout=0
S0=1 S1=1 64+44=453 Cin=1 Cout=0
S0=0 S1=1 25+141=478 Cin=0 Cout=0
S0=0 S1=1 103+61=69 Cin=0 Cout=0
15 > Fault Detected: 581 first simulation, 69 current Simulation
**
/testbench/Adder/Full_Adder/genblk1[8]/Add1/xor1/out Released @2725 ns
4/testbench/Adder/Full_Adder/genblk1[8]/Add1/xor1/out Released @2725 ns

```
#                             ...The system has been restarted...
# Fault Simulation 19 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv with stuck@0
@2725 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                             ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ****************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv Released @2885 ns
#         4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv Released @2885 ns
#                             ...The system has been restarted...
# Fault Simulation 20 : /testbench/Adder/Full_Adder/genblk1[8]/Add1/xor0/in0_inv with stuck@0
@2885 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                             ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ****************************************************************************
# /testbench/Adder/Full_Adder/genblk1[8]/Add1/xor0/in0_inv Released @3045 ns
#         4/testbench/Adder/Full_Adder/genblk1[8]/Add1/xor0/in0_inv Released @3045 ns
#                             ...The system has been restarted...
```

```
# Fault Simulation 21 : /testbench/Adder/Full_Adder/genblk1[5]/Add1/xor0/in0 with stuck@0 @3045
ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=103 Cin=1 Cout=0
# 6 > Fault Detected:   167 first simulation, 103 current Simulation
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[5]/Add1/xor0/in0 Released @3115 ns
#         4/testbench/Adder/Full_Adder/genblk1[5]/Add1/xor0/in0 Released @3115 ns
#                              ...The system has been restarted...
# Fault Simulation 22 : /testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/and1 with stuck@1 @3115
ns
#
# S0=1 S1=1 98+147=2049 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 2049 current Simulation
# ************************************************************************
# /testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/and1 Released @3135 ns
#         4/testbench/Adder/MuxOpA/genblk1[11]/Mux2to1/and1 Released @3135 ns
#                              ...The system has been restarted...
# Fault Simulation 23 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in1_inv with
stuck@0 @3135 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                              ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in1_inv Released @3295 ns
#         4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in1_inv Released @3295 ns
#                              ...The system has been restarted...
# Fault Simulation 24 : /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 with stuck@1 @3295 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
```

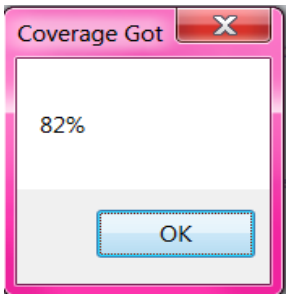
```
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                          ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 Released @3455 ns
#          4/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 Released @3455 ns
#                          ...The system has been restarted...
# Fault Simulation 25 : /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in1 with stuck@1 @3455
ns
#
# S0=1 S1=1 98+147=257 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 257 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in1 Released @3475 ns
#          4/testbench/Adder/Full_Adder/genblk1[7]/Add1/xor0/in1 Released @3475 ns
#                          ...The system has been restarted...
# Fault Simulation 26 : /testbench/Adder/MuxOpA/genblk1[13]/Mux2to1/Y with stuck@1 @3475 ns
#
# S0=1 S1=1 98+147=8193 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 8193 current Simulation
# ********************************************************************************
# /testbench/Adder/MuxOpA/genblk1[13]/Mux2to1/Y Released @3495 ns
#          4/testbench/Adder/MuxOpA/genblk1[13]/Mux2to1/Y Released @3495 ns
#                          ...The system has been restarted...
# Fault Simulation 27 : /testbench/Adder/Full_Adder/genblk1[5]/Add1/out_xor0 with stuck@0 @3495
ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=49 Cin=0 Cout=0
# 7 > Fault Detected:   113 first simulation, 49 current Simulation
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[5]/Add1/out_xor0 Released @3575 ns
```

```
#          4/testbench/Adder/Full_Adder/genblk1[5]/Add1/out_xor0 Released @3575 ns
#                                ...The system has been restarted...
# Fault Simulation 28 : /testbench/Adder/MuxOpB/genblk1[7]/Mux2to1/in1 with stuck@1 @3575 ns
#
# S0=1 S1=1 98+147=257 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 257 current Simulation
# ************************************************************************
# /testbench/Adder/MuxOpB/genblk1[7]/Mux2to1/in1 Released @3595 ns
#          4/testbench/Adder/MuxOpB/genblk1[7]/Mux2to1/in1 Released @3595 ns
#                                ...The system has been restarted...
# Fault Simulation 29 : /testbench/Adder/Full_Adder/genblk1[3]/Add1/xor0/in0_inv with stuck@0
@3595 ns
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                                ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=210 Cin=0 Cout=0
# 12 > Fault Detected:   226 first simulation, 210 current Simulation
#
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[3]/Add1/xor0/in0_inv Released @3725 ns
#          4/testbench/Adder/Full_Adder/genblk1[3]/Add1/xor0/in0_inv Released @3725 ns
#                                ...The system has been restarted...
# Fault Simulation 30 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in1 with stuck@1
@3725 ns
#
# S0=1 S1=1 98+147=4097 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 4097 current Simulation
#
# ************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in1 Released @3745 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in1 Released @3745 ns
#                                ...The system has been restarted...
# Fault Simulation 31 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv with stuck@0
@3745 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
```

```
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                              ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
#
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv Released @3905 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv Released @3905 ns
#                              ...The system has been restarted...
# Fault Simulation 32 : /testbench/Adder/Full_Adder/genblk1[7]/Add1/out_and1 with stuck@1
@3905 ns
#
# S0=1 S1=1 98+147=513 Cin=1 Cout=0
# 1 > Fault Detected:     1 first simulation, 513 current Simulation
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[7]/Add1/out_and1 Released @3925 ns
#          4/testbench/Adder/Full_Adder/genblk1[7]/Add1/out_and1 Released @3925 ns
#                              ...The system has been restarted...
#
# Fault Simulation 33 : /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/out with stuck@0
@3925 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                              ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ********************************************************************************
# /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/out Released @4085 ns
#          4/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/out Released @4085 ns
```

```
#                                  ...The system has been restarted...
# Fault Simulation 34 : /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1 with stuck@1 @4085 ns
#
# S0=1 S1=1 98+147=1 Cin=1 Cout=0
# S0=1 S1=1 107+125=2 Cin=0 Cout=0
# S0=0 S1=1 18+47=21 Cin=1 Cout=0
# S0=1 S1=1 126+97=42 Cin=0 Cout=0
# S0=0 S1=0 44+39=83 Cin=0 Cout=0
# S0=1 S1=1 103+134=167 Cin=1 Cout=0
# S0=0 S1=0 73+40=113 Cin=0 Cout=0
# S0=1 S1=1 21+72=227 Cin=1 Cout=0
# S0=0 S1=1 105+85=332 Cin=0 Cout=0
# S0=0 S1=0 29+47=76 Cin=0 Cout=0
#                                  ...Simulated 10 vectors...
# S0=0 S1=0 56+80=137 Cin=1 Cout=0
# S0=0 S1=0 99+127=226 Cin=0 Cout=0
# S0=1 S1=1 64+44=453 Cin=1 Cout=0
# S0=0 S1=1 25+141=478 Cin=0 Cout=0
# S0=0 S1=1 103+61=581 Cin=0 Cout=0
# FAULT NO DETECTED
#
# ...The faults simulation has finished. Sytem has been restored...
# Coverage=15
```

UpdateTest.exe



Podemos observar que ya se alcanzó la cobertura deseada con pruebas random.

Script para el cálculo de la cobertura

Fallas no detectadas

1. /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0
2. /testbench/Adder/Full_Adder/Add15/xor1/xor_out
3. /testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/in0
4. /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1
5. /testbench/Adder/Full_Adder/genblk1[10]/Add1/Add
6. /testbench/Adder/MuxOpA/genblk1[9]/Mux2to1/S1
7. /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/out
8. /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv
9. /testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y

10. /testbench/Adder/MuxOpB/genblk1[10]/Mux2to1/S1
11. /testbench/Adder/Full_Adder/Cout
12. /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv
13. /testbench/Adder/Full_Adder/genblk1[8]/Add1/xor0/in0_inv
14. /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in1_inv
15. /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1
16. /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv
17. /testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/out
18. /testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1

Logs del simulador de fallas desarrollado

Stucks.log → ..... Archivo donde se guardan los stucks en tiempo de ejecución de las fallas que no son encontradas

NoFind.log → ..... Archivo donde se almacena en tiempo de ejecución las fallas que no son encontradas

100Faults.log → ..... Guarda las 100 señales generadas aleatoriamente por la herramienta "GetSignalsLists.exe"

SignalsList.log → ..... Señales de todo el circuito

NewFaults.log → ..... Va actualizando las fallas que no se han detectado (UpdateTest.exe)

NewStucks.log → ..... Guarda los Stucks de las fallas que no fueron detectadas

Coverage.cmp → ..... Log con todos los test que se realizan hasta encontrar el coverage deseado

Simulation.cmp → ..... Log de la simulación actual

First_Simulation.log → . Guarda los resultados de la primera simulación sin fallas

Pruebas Dirigidas:

Este simulador ayuda a la generación de pruebas dirigidas, por lo que el usuario puede establecer los vectores que desea, y decidir qué falla desea simular de la lista de fallas que no se detectaron, también el usuario debe elegir qué tipo de "Stuck" desea.

```
vlog +define+STUCK_AT=0 +define+FAULTS=18 +define+A=0 +define+B=32768
+define+CIN=0 +define+NUMBER_FAULT=0 +define+MUX_A=0 +define+MUX_B=0
testbench.v
```

```
# S0=0 S1=0 0+32768=32768 Cin=0 Cout=0
# Fault Simulation: /testbench/Adder/MuxOpB/genblk1[15]/Mux2to1/and0 with stuck@0 @45 ns
#
# S0=0 S1=0 0+32768=0 Cin=0 Cout=0
# > Fault Detected: 32768 first simulation, 0 current Simulation
```

**

Elegibidad: *Esta falla afecta directamente al bit más significativo del Operador B, por lo que se necesita que el valor en este bit sea 1 por eso se eligió b'1000000000000000*

vlog +define+STUCK_AT=0 +define+FAULTS=18 +define+A=0 +define+B=32768 +define+CIN=0 +define+NUMBER_FAULT=1 +define+MUX_A=0 +define+MUX_B=0 testbench.v

S0=0 S1=0 0+32768=32768 Cin=0 Cout=0
Fault Simulation: /testbench/Adder/Full_Adder/Add15/xor1/xor_out with stuck@0 @45 ns
#
S0=0 S1=0 0+32768=0 Cin=0 Cout=0
> Fault Detected: 32768 first simulation, 0 current Simulation
**

Elegibidad: *Las entradas a esta xor son los bits más significativos de los operandos para que la XOR nos regrese un 1 y se pueda detectar la falla stuck@0 se eligió para A= b'0000000000000000 y para B= b'1000000000000000*

vlog +define+STUCK_AT=0 +define+FAULTS=18 +define+A=256 +define+B=4 +define+CIN=0 +define+NUMBER_FAULT=2 +define+MUX_A=0 +define+MUX_B=0 testbench.v

S0=0 S1=0 256+4=260 Cin=0 Cout=0
Fault Simulation: /testbench/Adder/MuxOpA/genblk1[8]/Mux2to1/in0 with stuck@0 @45 ns
#
S0=0 S1=0 256+4=4 Cin=0 Cout=0
> Fault Detected: 260 first simulation, 4 current Simulation
**

Elegibidad: *Como aquí se forza el bit 9 del Operando A se escogió para este un vector que de un 1 en este bit por eso se eligió: b'100000000*

vlog +define+STUCK_AT=0 +define+FAULTS=18 +define+A=2047 +define+B=2047 +define+CIN=1 +define+NUMBER_FAULT=3 +define+MUX_A=0 +define+MUX_B=0 testbench.v

S0=0 S1=0 2047+2047=4095 Cin=1 Cout=0
Fault Simulation: /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor1/in1 with stuck@0 @45 ns
#
S0=0 S1=0 2047+2047=3071 Cin=1 Cout=0
> Fault Detected: 4095 first simulation, 3071 current Simulation
**

Elegibidad: *Aquí se forza a 0 el Cin del sumador que nos da la suma del bit 10 se necesita que el Cin se mantenga en 1 hasta este sumador, por lo que se eligieron los vectores de manera que siempre se mantenga esta condición A = 1, B = 1, Cin = 1 y siempre se esté pasando el Cin al siguiente sumador los vectores son A = B = b`11111111111 Cin = 1*

```
vlog +define+STUCK_AT=0 +define+FAULTS=18 +define+A=2000 +define+B=48
+define+CIN=0 +define+NUMBER_FAULT=4 +define+MUX_A=0 +define+MUX_B=0
testbench.v
# S0=0 S1=0 2000+48=2048 Cin=0 Cout=0
# Fault Simulation: /testbench/Adder/Full_Adder/genblk1[10]/Add1/Add with stuck@0 @45 ns
#
# S0=0 S1=0 2000+48=0 Cin=0 Cout=0
# > Fault Detected:  2048 first simulation, 0 current Simulation
# ********************************************************************************
```

Elegibidad: *Se necesita que el resultado en el bit 12 sea 1 para que detecte el Stuck@0 por lo que se utilizaron vectores que al sumarlos dieran b'100000000000*

```
vlog +define+STUCK_AT=0 +define+FAULTS=18 +define+A=15 +define+B=15
+define+CIN=0 +define+NUMBER_FAULT=5 +define+MUX_A=1 +define+MUX_B=0
testbench.v
```

```
# S0=0 S1=1 15+15=15 Cin=0 Cout=0
# Fault Simulation: /testbench/Adder/MuxOpA/genblk1[9]/Mux2to1/S1 with stuck@0 @45 ns
#
# S0=0 S1=1 15+15=60 Cin=0 Cout=0
# > Fault Detected:    15 first simulation, 60 current Simulation
# ********************************************************************************
```

Elegibidad: *Cualquier patrón en los operadores pero el Selector de B debe ser 1*

```
vlog +define+STUCK_AT=0 +define+FAULTS=18 +define+A=1024 +define+B=0
+define+CIN=0 +define+NUMBER_FAULT=6 +define+MUX_A=0 +define+MUX_B=0
testbench.v
```

```
# S0=0 S1=0 1024+0=1024 Cin=0 Cout=0
# Fault Simulation: /testbench/Adder/Full_Adder/genblk1[9]/Add1/xor0/out with stuck@0 @45 ns
# S0=0 S1=0 1024+0=0 Cin=0 Cout=0
# > Fault Detected:  1024 first simulation, 0 current Simulation
# ********************************************************************************
```

Elegibidad: *La salida de la XOR debe ser 1 para que detecte el Stuck@0 por lo que se usó b' 10000000000 y b' o0000000000 para que los bits 11 de cada operador entregaran 1 a la salida de la XOR*

```
vlog +define+STUCK_AT=0 +define+FAULTS=18 +define+A=3 +define+B=512
+define+CIN=0 +define+NUMBER_FAULT=8 +define+MUX_A=0 +define+MUX_B=0
testbench.v
```

```
# S0=0 S1=0 3+512=515 Cin=0 Cout=0
# Fault Simulation: /testbench/Adder/MuxOpB/genblk1[9]/Mux2to1/out_Y with stuck@0 @45 ns
# S0=0 S1=0 3+512=3 Cin=0 Cout=0
# > Fault Detected:   515 first simulation, 3 current Simulation
# ********************************************************************************
```

Elegibidad: *El bit 10 del Operadir B debe ser 1 por lo que se seleccionó para B=b'1000000000*

```
vlog +define+STUCK_AT=1 +define+FAULTS=18 +define+A=24 +define+B=1000
+define+CIN=0 +define+NUMBER_FAULT=9 +define+MUX_A=0 +define+MUX_B=0
testbench.v
```

```
# S0=0 S1=0 24+1000=1024 Cin=0 Cout=0
# Fault Simulation: /testbench/Adder/MuxOpB/genblk1[10]/Mux2to1/S1 with stuck@1 @45 ns
# S0=0 S1=0 24+1000=2048 Cin=0 Cout=0
# > Fault Detected:  1024 first simulation, 2048 current Simulation
# ********************************************************************************
```

Elegibidad: Para que se detecte esta falla se necesita que los selectores se mantengan apagados y el bit 11 del resultado sea 1 (b' 10000000000) por que se eligió 1000+24 para los operandos

82% Random + 9% Dirigido = 91%

Se observó al estar haciendo las pruebas dirigidas que en la detección aleatoria de fallas muchas no se detectaron debido a que eran fallas generadas en la lógica que estimula los bits más significativos y la detección automática solo estaba generando cantidades no más altas de 150.

TAP Module

```
module TAP(input TCLK,          // Test clock
           input TMS,           // Test Mode Select
           input TDI,           // Test Data Input
           input TRST,          // Test Reset
           input DIN,           // Data In from ALU
           output reg TDO,      // Test Data Output
           output reg TM,       // Test Mode (enabled or disabled scan cells)
           output reg DOUT      // Serial data to ALU
           );

 // FSM states encoding
 parameter [3:0] Test_Logic_Reset = 4'b0000,
                 Run_Test_Idle    = 4'b0001,
                 Select_DR        = 4'b0010,
                 Capture_DR       = 4'b0011,
                 Shift_DR         = 4'b0100,
                 Exit1_DR         = 4'b0101,
                 Pause_DR         = 4'b0110,
                 Exit2_DR         = 4'b0111,
                 Update_DR        = 4'b1000,
                 Select_IR        = 4'b1001,
                 Capture_IR       = 4'b1010,
                 Shift_IR         = 4'b1011,
                 Exit1_IR         = 4'b1100,
                 Pause_IR         = 4'b1101,
                 Exit2_IR         = 4'b1110,
                 Update_IR        = 4'b1111;

 // FSM states registers
 reg [3:0] cur_state;
 reg [3:0] nxt_state;
 // Instructions
 parameter [1:0] Input_Scans  = 2'b10,
                 Output_Scans = 2'b01;
 //Data Registers
 reg [34:0]DRIn;
 reg [32:0]DROut;
```

```
//Instruction Register:
//10 - Input Scans
//01 - Output Scans
reg [1:0]IR;
//General registers
reg [1:0]ShiftIR;   //Counter to check the bit shifthed of the instruction
reg [7:0]ShiftDR; //Counter to check the bit shifthed of the data
// FSM registered process
always@(posedge TCLK, posedge TRST) begin
  if (TRST) begin
    cur_state <= Test_Logic_Reset;
    IR <= 0;
    TM <= 0;
    TDO <= 0;
    DOUT <= 0;
    DRIn <= 0;
    DROut <= 0;
    ShiftIR <= 0;
    ShiftDR <= 0;
  end else begin
    cur_state <= nxt_state;
    case(cur_state)
      /***Instruction Regusters Path***/
      Shift_IR:
      begin
        IR[ShiftIR] <= TDI;        //Save the instruction
        TDO <= TDI;                  //Show trough TDO de current instruction
        ShiftIR <= ShiftIR + 1; //Instruction Register Shift
      end
      /***Data Registers Path***/
      Select_DR:
      begin
        if(IR == Input_Scans || IR == Output_Scans)ShiftIR <= 0;
        TM <= 0;
      end
      Shift_DR:
      begin
        if(IR == Input_Scans)
        begin
          TM <= 1;                    //Active input scan cells to capture the test vector introduce trough TDI
          DRIn[ShiftDR] <= TDI;      //Save the vector in a reg just for comparations
          DOUT <= TDI;                //Send the new test vector to the input scan cells
          TDO <= TDI;                 //Show the new test vector trough TDO
          ShiftDR <= ShiftDR + 1;   //Shift control
          if(ShiftDR == 34) ShiftDR <= 0;
        end
        else
        begin
          TDO <= DROut[ShiftDR];        //Sends the test result trough TDO (to compare result in TB)
          ShiftDR <= ShiftDR + 1;         //Shift control
          if(ShiftDR == 32) ShiftDR <= 0;
```

```
        end
      end
      Exit1_DR:
      begin
        TM <= 0;     //Disabled TM to allow test the ALU with the test vector introduce trough TAP
        TDO <= 0;
        ShiftDR = 0;
      end

      Exit1_IR: TDO <= 0;

      Update_DR:
      begin
        TM <= 1;          //Active Test Mode to get the results of the test vector introduce trough TAP
        if(IR == Input_Scans)
        begin
          DROut[ShiftDR] <= DIN;  //start to save the new result
          ShiftDR <= ShiftDR + 1;
        end
      end

      Run_Test_Idle:
      begin
        if(IR == Input_Scans)     //When finish the evaluation of the test vector instroduce trough TAP
        begin
          DROut[ShiftDR] <= DIN;         //Finish to save the result
          ShiftDR <= ShiftDR + 1;
          if(ShiftDR == 32) ShiftDR <= 0;
        end
      end
    endcase
  end
end
//States path
always@(*)
begin
  case(cur_state)
    Test_Logic_Reset:
    begin
      if(TMS) nxt_state = Test_Logic_Reset;
      else nxt_state = Run_Test_Idle;
    end
    Run_Test_Idle:
    begin
      if(TMS) nxt_state = Select_DR;
      else nxt_state = Run_Test_Idle;
    end

Select_DR:
    begin
      if(TMS) nxt_state = Select_IR;
```

```
      else nxt_state = Capture_DR;
    end
    Capture_DR:
    begin
      if(TMS) nxt_state = Exit1_DR;
      else nxt_state = Shift_DR;
    end
    Shift_DR:
    begin
      if(TMS) nxt_state = Exit1_DR;
      else nxt_state = Shift_DR;
    end
    Exit1_DR:
    begin
      if(TMS) nxt_state = Update_DR;
      else nxt_state = Pause_DR;
    end
    Pause_DR:
    begin
      if(TMS) nxt_state = Exit2_DR;
      else nxt_state = Pause_DR;
    end
    Exit2_DR:
    begin
      if(TMS) nxt_state = Update_DR;
      else nxt_state = Shift_DR;
    end
    Update_DR:
    begin
      if(TMS) nxt_state = Select_DR;
      else nxt_state = Run_Test_Idle;
    end
    Select_IR:
    begin
      if(TMS) nxt_state = Test_Logic_Reset;
      else nxt_state = Capture_IR;
    end
    Capture_IR:
    begin
      if(TMS) nxt_state = Exit1_IR;
      else nxt_state = Shift_IR;
    end

  Shift_IR:
    begin
      if(TMS) nxt_state = Exit1_IR;
      else nxt_state = Shift_IR;
    end
    Exit1_IR:
    begin
      if(TMS) nxt_state = Update_IR;
```

```
      else nxt_state = Pause_IR;
    end
    Pause_IR:
    begin
      if(TMS) nxt_state = Exit2_IR;
      else nxt_state = Pause_IR;
    end
    Exit2_IR:
    begin
      if(TMS) nxt_state = Update_IR;
      else nxt_state = Shift_IR;
    end
    Update_IR:
    begin
      if(TMS) nxt_state = Select_DR;
      else nxt_state = Run_Test_Idle;
    end
   endcase
  end
endmodule
```

Scan cell in

```
module scan_cell_in(input TCLK, input TRST, input PDIN, input SDIN, input TM, output OUT);
        wire d, out;
        Mux2to1 MUX(.S(TM), .in0(PDIN), .in1(SDIN), .Y(d));
        dff FF(out, TCLK, TRST, d);
        assign OUT = out;
endmodule
```

Scan cell out

```
module scan_cell_out(input TCLK, input TRST, input PDIN, input SDIN, input TM, output OUT);
        wire d, out;
        Mux2to1 MUX(.S(TM), .in0(PDIN), .in1(SDIN), .Y(d));
        dff_n FF(out, TCLK, TRST, d);
        assign OUT = out;
endmodule
```

Scan Chain implementation

```
module Adder(input clk, input rst, input SELA, input SELB, input CI, input [15:0]OpA, input [15:0]OpB,
                input DIN, input TM, output DOUT, output CO, output [15:0]SUM);

  wire SELA_w;
  wire SELB_w;
  wire CI_w;
  wire [15:0]OpA_w;
  wire [15:0]OpB_w;
  wire CO_w;
```

```
  wire Cout;
  wire [15:0]Op1;
  wire [15:0]Op2;
  wire [15:0]Result;
  wire [15:0]ResultSyncA;

  scan_cell_in SyncSELA(.TCLK(clk), .TRST(rst), .PDIN(SELA), .SDIN(DIN), .TM(TM), .OUT(SELA_w));
  scan_cell_in SyncSELB(.TCLK(clk), .TRST(rst), .PDIN(SELB), .SDIN(SELA_w), .TM(TM), .OUT(SELB_w));
  scan_cell_in SyncCI(.TCLK(clk), .TRST(rst), .PDIN(CI), .SDIN(SELB_w), .TM(TM), .OUT(CI_w));
  scan_cell_in SyncOpA0(.TCLK(clk), .TRST(rst), .PDIN(OpA[15]), .SDIN(CI_w), .TM(TM),
.OUT(OpA_w[15]));

  generate
    genvar i;
    for(i=0; i<15; i=i+1)
    begin
      scan_cell_in SyncOpA(.TCLK(clk), .TRST(rst), .PDIN(OpA[i]), .SDIN(OpA_w[i+1]), .TM(TM),
.OUT(OpA_w[i]));
    end
  endgenerate
  scan_cell_in SyncOpB0(.TCLK(clk), .TRST(rst), .PDIN(OpB[15]), .SDIN(OpA_w[0]), .TM(TM),
.OUT(OpB_w[15]));
  generate
    genvar j;
    for(j=0; j<15; j=j+1)
    begin
      scan_cell_in SyncOpB(.TCLK(clk), .TRST(rst), .PDIN(OpB[j]), .SDIN(OpB_w[j+1]), .TM(TM),
.OUT(OpB_w[j]));
    end
  endgenerate
  scan_cell_out Feedback_out0(.TCLK(clk), .TRST(rst), .PDIN(Result[0]), .SDIN(ResultSyncA[1]), .TM(TM),
.OUT(ResultSyncA[0]));
  generate
    genvar k;
    for(k=1; k<15; k=k+1)
    begin
      scan_cell_out Feedback_out(.TCLK(clk), .TRST(rst), .PDIN(Result[k]), .SDIN(ResultSyncA[k+1]),
.TM(TM), .OUT(ResultSyncA[k]));
    end
  endgenerate
  scan_cell_out Feedback_out15(.TCLK(clk), .TRST(rst), .PDIN(Result[15]), .SDIN(CO_w), .TM(TM),
.OUT(ResultSyncA[15]));
  scan_cell_out SyncCout(.TCLK(clk), .TRST(rst), .PDIN(Cout), .SDIN(CO_w), .TM(TM), .OUT(CO_w));
  Mux2to1_16bits MuxOpA(.S(SELA_w), .in0(OpA_w), .in1(ResultSyncA), .Y(Op1));
  Mux2to1_16bits MuxOpB(.S(SELB_w), .in0(OpB_w), .in1(ResultSyncA), .Y(Op2));
```

```
  Full_Adder_16bits Full_Adder(.A(Op1), .B(Op2), .Cin(CI_w), .Add(Result), .Cout(Cout));
  assign DOUT = ResultSyncA[0];
  assign CO = CO_w;
  assign SUM = ResultSyncA;
endmodule
```

Test Bench Module

```
module testbench_TAP();
  //Interface with DUT
  reg [15:0]Op1;   //Input Adder
  reg [15:0]Op2;   //Input Adder
  reg Cin;          //Input Adder
  reg S0;          //Input Adder
  reg S1;          //Input Adder
  reg TMS;
  reg TDI;
  wire Cout;              //Ouput Adder
  wire [15:0]Result;      //Ouput Adder
  wire dout;
  wire din;
  wire TDO;
  reg clk, rst, flag0, flag1;
  reg [34:0]test;
  reg [32:0]compare;
  reg [7:0]cycle;
  //Reset Generator
  initial
  begin
    rst = 1;
    #5 rst = 0;
  end
  //Clock Generator
  initial
  begin
    clk = 0;
    forever #5 clk = ~clk;
  end
  initial
  begin
    cycle = 0;
    Cin = 0 + {$random} % (2 - 0);
    Op1 = {$random} %16384;
    Op2 = {$random} %16384;
    S0 = 0;
```

```
    S1 = 0 + {$random} % (2 - 0);
    test = {$random} %262144;
  end
  initial begin
  $display("Simulation without faults");
  $display("*************************************************************************");
  $display("| Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |");
  $display("*************************************************************************");
//NOTE: Since are not a lot faults to simulate and is necessary a lot of clock cycles to simulate the
          //test sequence, is launched a simulation for each fault
  //$display("Simulation with Stuck@0 and Stuck@1");
  //$signal_force("/testbench_TAP/Adder/Full_Adder/genblk1[5]/Add1/out_Cout","2#1"); //Fault
Injection (S@1)
  //$signal_force("/testbench_TAP/Adder/Full_Adder/genblk1[6]/Add1/out_and1","2#0"); //Fault
Injection (S@0)
  //$signal_force("/testbench_TAP/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv","2#1"); //Fault
Injection (S@1)
  //$signal_force("/testbench_TAP/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv","2#0");
//Fault Injection (S@0)
  //$signal_force("/testbench_TAP/Adder/Full_Adder/Cout","2#1"); //Fault Injection (S@1)
end
  always@(posedge clk, posedge rst) begin
    if (rst) begin
      TDI <= 0;
      TMS <= 0; //Test_Logic_Reset
      //test <= {1'b0, 1'b0, 1'b0, 16'd15000, 16'd3000};
      //       S0   S1    Cin      Op1       Op2
      compare <= 0;
    end
    else begin
      case(cycle)
        0: begin
          TMS <= 1; cycle <= cycle + 1;
  $display("Normal Test - Vector introduce since PI");
  $display("|    %d      |    %d      | %d     |   %d    |  %d  |%d |   %d  |", S0, S1, Op1, Op2, Cin, Result,
Cout);
          end  //Run_Test_Idle
        1: begin
          TMS <= 1; cycle <= cycle + 1;
  $display("|    %d      |    %d      | %d     |   %d    |  %d  |%d |   %d  |", S0, S1, Op1, Op2, Cin, Result,
Cout);
          end  //Select_DR
        2: begin TMS <= 0; cycle <= cycle + 1; end  //Select_IR
        3: begin TMS <= 0; cycle <= cycle + 1; end  //Capture_IR
        4: begin TMS <= 0; cycle <= cycle + 1; TDI <= 0; end  //Shift_IR
```

```
        5: begin TMS <= 1; cycle <= cycle + 1; TDI <= 1; end  //Shift_IR
        6: begin TMS <= 1; cycle <= cycle + 1; end  //Exit_IR
        7: begin TMS <= 0; cycle <= cycle + 1; end  //Update_IR
        //Finish to capture IR (Input Scans)
        8: begin TMS <= 1; cycle <= cycle + 1; end  //Run_Test_Idle
        9: begin TMS <= 0; cycle <= cycle + 1; end  //Select_DR
        10: begin TMS <= 0; cycle <= cycle + 1; end  //Capture_DR
        11: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[0]; end  //Shift_DR_0
        12: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[1]; end  //Shift_DR_1
        13: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[2]; end  //Shift_DR_2
        14: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[3]; end  //Shift_DR_3
        15: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[4]; end  //Shift_DR_4
        16: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[5]; end  //Shift_DR_5
        17: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[6]; end  //Shift_DR_6
        18: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[7]; end  //Shift_DR_7
        19: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[8]; end  //Shift_DR_8
        20: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[9]; end  //Shift_DR_9
        21: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[10]; end  //Shift_DR_10

        22: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[11]; end  //Shift_DR_11
        23: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[12]; end  //Shift_DR_12
        24: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[13]; end  //Shift_DR_13
        25: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[14]; end  //Shift_DR_14
        26: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[15]; end  //Shift_DR_15
        27: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[16]; end  //Shift_DR_16
        28: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[17]; end  //Shift_DR_17
        29: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[18]; end  //Shift_DR_18
        30: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[19]; end  //Shift_DR_19
        31: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[20]; end  //Shift_DR_20
        32: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[21]; end  //Shift_DR_21
        33: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[22]; end  //Shift_DR_22
        34: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[23]; end  //Shift_DR_23
        35: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[24]; end  //Shift_DR_24
        36: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[25]; end  //Shift_DR_25
        37: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[26]; end  //Shift_DR_26
        38: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[27]; end  //Shift_DR_27
        39: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[28]; end  //Shift_DR_28
        40: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[29]; end  //Shift_DR_29
        41: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[30]; end  //Shift_DR_30
        42: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[31]; end  //Shift_DR_31
        43: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[32]; end  //Shift_DR_32
        44: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[33]; end  //Shift_DR_33
        45: begin
            TMS <= 1; cycle <= cycle + 1; TDI <= test[34];
$display("*************************************************************************");
```

```
$display("\t\t\tTAP CONTROLLER");
$display("Test vector trough TAP: %b", test);
$display("|    %d     |    %d     | %d    |   %d   |  %d  |", test[34], test[33], test[31:16], test[15:0],
test[32]);
          end  //Shift_DR_34
       46: begin TMS <= 1; cycle <= cycle + 1; end  //Exit_DR
       47: begin TMS <= 0; cycle <= cycle + 1; end  //Update_DR
       //Save test vector bye TAP
       48: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       49: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       50: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       51: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       52: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register

       53: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       54: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       55: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       56: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       57: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       58: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       59: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       60: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       61: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       62: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       63: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       64: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       65: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       66: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       67: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       68: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       69: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       70: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       71: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       72: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       73: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       74: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       75: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       76: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       77: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       78: begin TMS <= 0; cycle <= cycle + 1; end  //Save result in output data register
       79: begin TMS <= 1; cycle <= cycle + 1; end  //Save result in output data register
       //Finish to get the test and start to pass result truogh TDO
       80: begin TMS <= 1; cycle <= cycle + 1; end  //Select_DR
       81: begin TMS <= 0; cycle <= cycle + 1; end  //Select_IR
       82: begin TMS <= 0; cycle <= cycle + 1; end  //Capture_IR
```

```
83: begin TMS <= 0; cycle <= cycle + 1; TDI <= 1; end  //Shift_IR
84: begin TMS <= 1; cycle <= cycle + 1; TDI <= 0; end  //Shift_IR
85: begin TMS <= 1; cycle <= cycle + 1; end  //Exit_IR
86: begin TMS <= 1; cycle <= cycle + 1; end  //Update_IR
87: begin TMS <= 0; cycle <= cycle + 1; end
//Finish to capture IR
88: begin TMS <= 0; cycle <= cycle + 1; end
89: begin TMS <= 0; cycle <= cycle + 1; end  //Select_DR
90: begin TMS <= 0; cycle <= cycle + 1; end  //Capture_DR
91: begin TMS <= 0; cycle <= cycle + 1; end  //Shift_DR_0
92: begin TMS <= 0; cycle <= cycle + 1; compare[0] <= TDO; end  //Shift_DR_1
93: begin TMS <= 0; cycle <= cycle + 1; compare[1] <= TDO; end  //Shift_DR_2
94: begin TMS <= 0; cycle <= cycle + 1; compare[2] <= TDO; end  //Shift_DR_3
95: begin TMS <= 0; cycle <= cycle + 1; compare[3] <= TDO; end  //Shift_DR_4
96: begin TMS <= 0; cycle <= cycle + 1; compare[4] <= TDO; end  //Shift_DR_5
97: begin TMS <= 0; cycle <= cycle + 1; compare[5] <= TDO; end  //Shift_DR_6
98: begin TMS <= 0; cycle <= cycle + 1; compare[6] <= TDO; end  //Shift_DR_7
99: begin TMS <= 0; cycle <= cycle + 1; compare[7] <= TDO; end  //Shift_DR_8
100: begin TMS <= 0; cycle <= cycle + 1; compare[8] <= TDO; end  //Shift_DR_9
101: begin TMS <= 0; cycle <= cycle + 1; compare[9] <= TDO; end  //Shift_DR_10
102: begin TMS <= 0; cycle <= cycle + 1; compare[10] <= TDO; end  //Shift_DR_11
103: begin TMS <= 0; cycle <= cycle + 1; compare[11] <= TDO; end  //Shift_DR_12
104: begin TMS <= 0; cycle <= cycle + 1; compare[12] <= TDO; end  //Shift_DR_13
105: begin TMS <= 0; cycle <= cycle + 1; compare[13] <= TDO; end  //Shift_DR_14
106: begin TMS <= 0; cycle <= cycle + 1; compare[14] <= TDO; end  //Shift_DR_15
107: begin TMS <= 0; cycle <= cycle + 1; compare[15] <= TDO; end  //Shift_DR_16
108: begin TMS <= 0; cycle <= cycle + 1; compare[16] <= TDO; end  //Shift_DR_17
109: begin TMS <= 0; cycle <= cycle + 1; compare[17] <= TDO; end  //Shift_DR_18
110: begin TMS <= 0; cycle <= cycle + 1; compare[18] <= TDO; end  //Shift_DR_19
111: begin TMS <= 0; cycle <= cycle + 1; compare[19] <= TDO; end  //Shift_DR_20
112: begin TMS <= 0; cycle <= cycle + 1; compare[20] <= TDO; end  //Shift_DR_21
113: begin TMS <= 0; cycle <= cycle + 1; compare[21] <= TDO; end  //Shift_DR_22
114: begin TMS <= 0; cycle <= cycle + 1; compare[22] <= TDO; end  //Shift_DR_23
115: begin TMS <= 0; cycle <= cycle + 1; compare[23] <= TDO; end  //Shift_DR_24
116: begin TMS <= 0; cycle <= cycle + 1; compare[24] <= TDO; end  //Shift_DR_25
117: begin TMS <= 0; cycle <= cycle + 1; compare[25] <= TDO; end  //Shift_DR_26
118: begin TMS <= 0; cycle <= cycle + 1; compare[26] <= TDO; end  //Shift_DR_27
119: begin TMS <= 0; cycle <= cycle + 1; compare[27] <= TDO; end  //Shift_DR_28
120: begin TMS <= 0; cycle <= cycle + 1; compare[28] <= TDO; end  //Shift_DR_29
121: begin TMS <= 0; cycle <= cycle + 1; compare[29] <= TDO; end  //Shift_DR_30

122: begin TMS <= 0; cycle <= cycle + 1; compare[30] <= TDO; end  //Shift_DR_31
123: begin TMS <= 1; cycle <= cycle + 1; compare[31] <= TDO; end  //Shift_DR_32
124: begin
    TMS <= 1; cycle <= cycle + 1; compare[32] <= TDO;
```

```
        $display("Result trough TDO: %b - %d", compare, compare);
        end  //Send to Exit_DR
      125: begin TMS <= 0; cycle <= cycle + 1; end  //Send to Update_DR
      //Finish TEST ready for compare
    endcase
  end
end
//Instance of DUT
 TAP TAP(.TCLK(clk), .TRST(rst), .TMS(TMS), .TDI(TDI), .TDO(TDO), .TM(TM), .DIN(din), .DOUT(dout));
 Adder Adder(.clk(clk), .rst(rst), .SELA(S0), .SELB(S1), .CI(Cin), .OpA(Op1), .OpB(Op2), .CO(Cout),
.SUM(Result), .TM(TM), .DIN(dout), .DOUT(din));  endmodule
```

Resultados

5 FAULTS SIMULATION (S@0 & S@1) – COVERAGE

Fault list (all signals S@0 & S@1):

1. "/testbench_TAP/Adder/Full_Adder/genblk1[5]/Add1/out_Cout" S@1
2. "/testbench_TAP/Adder/Full_Adder/genblk1[6]/Add1/out_and1" S@0
3. "/testbench_TAP/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv" S@1
4. "/testbench_TAP/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv" S@0
5. "/testbench_TAP/Adder/Full_Adder/Cout" S@1

Test vector without fault

```
# ********************************************************************************
# | Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
# ********************************************************************************
# Simulation with Stuck@0 and Stuck@1
# Normal Test - Vector introduce since PI
# |    0     |    1     |   7809    |   5641    | 0 |   0 |  0 |
# |    0     |    1     |   7809    |   5641    | 0 | 7809 |  0 |
# ********************************************************************************
#                                TAP CONTROLLER
# Test vector trough TAP: 00000000110101111001011111011000011101
# |    0     |    0     |   1721    |   31501    | 0 |
# Result trough TDO: 00000000000000000100000011100011 -     33222
```

S@1: "/testbench_TAP/Adder/Full_Adder/genblk1[5]/Add1/out_Cout"

```
# ********************************************************************************
# | Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
# ********************************************************************************
# Simulation with Stuck@0 and Stuck@1
# Normal Test - Vector introduce since PI
# |    0     |    1     |   7809    |   5641    | 0 |   0 |  0 |
# |    0     |    1     |   7809    |   5641    | 0 | 7937 |  0 |
# ********************************************************************************
#                                TAP CONTROLLER
```

Test vector trough TAP: 00000000110101110010111101100001101
| 0 | 0 | 1721 | 31501 | 0 |
Result trough TDO: 00000000000000000100000100100110 - 33350

Test vector without fault

**
| Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
**
Simulation with Stuck@0 and Stuck@1
Normal Test - Vector introduce since PI
| 0 | 1 | 7809 | 5641 | 0 | 0 | 0 |
| 0 | 1 | 7809 | 5641 | 0 | 7809 | 0 |
**
TAP CONTROLLER
Test vector trough TAP: 00000000000001110010111101100001101
| 0 | 0 | 57 | 31501 | 0 |
Result trough TDO: 00000000000000000011110111010000110 - 31558

S@0: "/testbench_TAP/Adder/Full_Adder/genblk1[6]/Add1/out_and1"

**
| Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
**
Simulation with Stuck@0 and Stuck@1
Normal Test - Vector introduce since PI
| 0 | 1 | 7809 | 5641 | 0 | 0 | 0 |
| 0 | 1 | 7809 | 5641 | 0 | 7809 | 0 |
**
TAP CONTROLLER
Test vector trough TAP: 00000000000001110010111101100001101
| 0 | 0 | 57 | 31501 | 0 |
Result trough TDO: 00000000000000000011110111010000110 - 31558

Test vector without fault

**
| Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
**
Simulation with Stuck@0 and Stuck@1
Normal Test - Vector introduce since PI
| 0 | 1 | 7809 | 5641 | 0 | 0 | 0 |
| 0 | 1 | 7809 | 5641 | 0 | 7809 | 0 |
**
TAP CONTROLLER
Test vector trough TAP: 00000000111000010000000101110111000
| 0 | 0 | 1800 | 3000 | 0 |
Result trough TDO: 00000000000000000000010011011000000 - 4800

S@1: "/testbench_TAP/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv"

**
| Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
**
Simulation with Stuck@0 and Stuck@1

```
# Normal Test - Vector introduce since PI
# |    0    |    1    |  7809   |   5641   | 0 |   0 |  0  |
# |    0    |    1    |  7809   |   5641   | 0 | 7809 |  0  |
# ********************************************************************************
#                                TAP CONTROLLER
# Test vector trough TAP: 000000001110000100000001011101110000
# |    0    |    0    |  1800   |   3000   | 0 |
# Result trough TDO: 000000000000000000001001111000000 -      5056
```

Vector de prueba con falla

```
# ********************************************************************************
# | Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
# ********************************************************************************
# Simulation with Stuck@0 and Stuck@1
# Normal Test - Vector introduce since PI
# |    0    |    1    |  7809   |   5641   | 0 |   0 |  0  |
# |    0    |    1    |  7809   |   5641   | 0 | 7809 |  0  |
# ********************************************************************************
#                                TAP CONTROLLER
# Test vector trough TAP: 000000001110010101100001110111000010
# |    0    |    0    |  1835   |   3810   | 0 |
# Result trough TDO: 000000000000000000001011000001101 -      5645
S@0: "/testbench_TAP/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv"

# ********************************************************************************
# | Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
# ********************************************************************************
# Simulation with Stuck@0 and Stuck@1
# Normal Test - Vector introduce since PI
# |    0    |    1    |  7809   |   5641   | 0 |   0 |  0  |
# |    0    |    1    |  7809   |   5641   | 0 | 7809 |  0  |
# ********************************************************************************
#                                TAP CONTROLLER
# Test vector trough TAP: 000000001110010101100001110111000010
# |    0    |    0    |  1835   |   3810   | 0 |
# Result trough TDO: 000000000000000000001011000001101 -      5645
```

Vector de prueba sin falla

```
# ********************************************************************************
# | Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
# ********************************************************************************
# Simulation with Stuck@0 and Stuck@1
# Normal Test - Vector introduce since PI
# |    0    |    1    |  7809   |   5641   | 0 |   0 |  0  |
# |    0    |    1    |  7809   |   5641   | 0 | 7809 |  0  |
# ********************************************************************************
#                                TAP CONTROLLER
# Test vector trough TAP: 000001110101001100000001011101110000
```

| 0 | 0 | 15000 | 3000 | 0 |
Result trough TDO: 00000000000000000001000110010100000 - 18000

S@1: "/testbench_TAP/Adder/Full_Adder/Cout"
**
| Selector 0 | Selector 1 | Operando A | Operando B | Cin | SUMA | Cout |
**
Simulation with Stuck@0 and Stuck@1
Normal Test - Vector introduce since PI
| 0 | 1 | 7809 | 5641 | 0 | 0 | 0 |
| 0 | 1 | 7809 | 5641 | 0 | 7809 | 1 |
**
TAP CONTROLLER
Test vector trough TAP: 00000111010100110000000101110111000
| 0 | 0 | 15000 | 3000 | 0 |
Result trough TDO: 01111111111111111101000110010100000 – 4294919760
COVERAGE: 60% - 3 faults detected of 5

BIST MODULE

En este proyecto se incluyó todo en un solo módulo que es el que se presenta a continuación, donde se puede observar el polinomio de grado 51 para el generador y el polinomio de grado 17 para el compresor y las modificaciones que se le tuvo que hacer alrededor de las scan cells, también incluye la instancia de la ALU.

Verilog netlist

```
module Adder(input clk,
             input rst,
             input SELA,
             input SELB,
             input CI,
             input [15:0]OpA,
             input [15:0]OpB,
             input [15:0]Feedback,
             input DIN,
             input TM,
             input gen,
             output DOUT,
             output CO,
             output [15:0]SUM);

  wire SELA_w;
  wire SELB_w;
  wire CI_w;
  wire [15:0]OpA_w;
  wire [15:0]OpB_w;
  wire [15:0]Feedback_w;
  wire CO_w;
```

```
  wire Cout;
  wire [15:0]Op1;
  wire [15:0]Op2;
  wire [15:0]Result;
  wire [15:0]ResultSyncA;
  wire lfsr_in;
  wire lfsr_out;
  wire scan_input;
  wire out_51_16;
  wire out_15;
  wire [15:0]signature;
 //polynomio –> x^51+ x^16 + x^15 + x + 1
  xor_gate xor_51_16(.in0(Feedback_w[0]),.in1(OpA_w[3]),.out(out_51_16));
  xor_gate xor_15(.in0(out_51_16),.in1(OpA_w[4]),.out(out_15));
  xor_gate xor_lfsr_in(.in0(out_15),.in1(SELA_w),.out(lfsr_in));
  //Mux to select between seed and input generator
  Mux2to1 Mux_Seed_Gen(.S(gen), .in0(DIN), .in1(lfsr_in), .Y(scan_input));
  scan_cell_in SyncSELA(.TCLK(clk), .TRST(rst), .PDIN(SELA), .SDIN(scan_input), .TM(TM), .OUT(SELA_w));
  scan_cell_in SyncSELB(.TCLK(clk), .TRST(rst), .PDIN(SELB), .SDIN(SELA_w), .TM(TM), .OUT(SELB_w));
  scan_cell_in SyncCI(.TCLK(clk), .TRST(rst), .PDIN(CI), .SDIN(SELB_w), .TM(TM), .OUT(CI_w));
  scan_cell_in SyncOpA0(.TCLK(clk), .TRST(rst), .PDIN(OpA[15]), .SDIN(CI_w), .TM(TM),
.OUT(OpA_w[15]));
  generate
    genvar i;
    for(i=0; i<15; i=i+1)
    begin
      scan_cell_in SyncOpA(.TCLK(clk), .TRST(rst), .PDIN(OpA[i]), .SDIN(OpA_w[i+1]), .TM(TM),
.OUT(OpA_w[i]));
    end
  endgenerate
   scan_cell_in SyncOpB0(.TCLK(clk), .TRST(rst), .PDIN(OpB[15]), .SDIN(OpA_w[0]), .TM(TM),
.OUT(OpB_w[15]));

  generate
    genvar j;
    for(j=0; j<15; j=j+1)
    begin
      scan_cell_in SyncOpB(.TCLK(clk), .TRST(rst), .PDIN(OpB[j]), .SDIN(OpB_w[j+1]), .TM(TM),
.OUT(OpB_w[j]));
    end
  endgenerate

  scan_cell_in SyncFeedback0(.TCLK(clk), .TRST(rst), .PDIN(Feedback[15]), .SDIN(OpB_w[0]), .TM(TM),
.OUT(Feedback_w[15]));
  generate
    genvar k;
    for(k=0; k<15; k=k+1)
    begin
      scan_cell_in SyncFeedback(.TCLK(clk), .TRST(rst), .PDIN(Feedback[k]), .SDIN(Feedback_w[k+1]),
.TM(TM), .OUT(Feedback_w[k]));
    end
```

```
  endgenerate
  //Start Output Scans
  scan_cell_out out0(.TCLK(clk), .TRST(rst), .PDIN(Result[0]), .SDIN(signature[0]), .TM(TM),
.OUT(ResultSyncA[0]));
  xor_gate signature0(.in0(Result[0]),.in1(ResultSyncA[1]),.out(signature[0]));

  generate
    genvar s;
    for(s=1; s<15; s=s+1)
    begin
      scan_cell_out Feedback_out(.TCLK(clk), .TRST(rst), .PDIN(Result[s]), .SDIN(signature[s]), .TM(TM),
.OUT(ResultSyncA[s]));
      xor_gate signature(.in0(Result[s]),.in1(ResultSyncA[s+1]),.out(signature[s]));
    end
  endgenerate
  scan_cell_out out15(.TCLK(clk), .TRST(rst), .PDIN(Result[15]), .SDIN(signature[15]), .TM(TM),
.OUT(ResultSyncA[15]));
  xor_gate signature16(.in0(Result[15]),.in1(CO_w),.out(signature[15]));
  scan_cell_out SyncCout(.TCLK(clk), .TRST(rst), .PDIN(Cout), .SDIN(lfsr_out), .TM(TM), .OUT(CO_w));
  //polynomio –> x^17 + x^3 + 1
  xor_gate xor_lfsr_out(.in0(ResultSyncA[14]),.in1(ResultSyncA[0]),.out(lfsr_out));

  Mux2to1_16bits MuxOpA(.S(SELA_w), .in0(OpA_w), .in1(Feedback_w), .Y(Op1));
  Mux2to1_16bits MuxOpB(.S(SELB_w), .in0(OpB_w), .in1(Feedback_w), .Y(Op2));
  Full_Adder_16bits Full_Adder(.A(Op1), .B(Op2), .Cin(CI_w), .Add(Result), .Cout(Cout));

  assign DOUT = ResultSyncA[0];
  assign CO = CO_w;
  assign SUM = ResultSyncA;
endmodule
```

TEST BENCH MODULE

```
module testbench_TAP();

  //Interface with DUT
  reg [15:0]Op1;       //Input Adder
  reg [15:0]Op2;       //Input Adder
  reg [15:0]Feedback;
  reg Cin;             //Input Adder
  reg S0;              //Input Adder
  reg S1;              //Input Adder
  reg TMS;
  reg TDI;
  reg gen;
  reg clk, rst, flag0, flag1;
  reg [50:0]test;
  reg [16:0]compare;
  reg [7:0]cycle;

  wire Cout;           //Ouput Adder
  wire [15:0]Result;   //Ouput Adder
```

```
  wire dout;
  wire din;
  wire TDO;
  wire TM;

  //Reset Generator
  initial
  begin
    rst = 1;
    #5 rst = 0;
  end
  //Clock Generator
  initial
  begin
    clk = 0;
    forever #5 clk = ~clk;
  end
  initial
  begin
    cycle = 0;
    Cin = 0 + {$random} % (2 - 0);
    Op1 = {$random} %16384;
    Op2 = {$random} %16384;
    S0 = 0;
    S1 = 0 + {$random} % (2 - 0);
    Feedback = 0;
    gen = 0;    `
    test[25:0] = {$random} %33554432;
    test[50:26] = {$random} %33554432;
  end
    initial begin
  $display("Simulation without faults");
  $display("*************************************************************************");
  $display("| Selector 0 | Selector 1 | Operando A | Operando B | Feedback | Cin | SUMA | Cout |");
  $display("*************************************************************************");
//NOTE: Since are not a lot faults to simulate and is necessary a lot of clock cycles to simulate the test
sequence is launched a simulation for each fault
//$display("Simulation with Stuck@0 and Stuck@1");
//$signal_force("/testbench_TAP/Adder/Full_Adder/genblk1[5]/Add1/out_Cout","2#1"); //Fault
Injection (S@1)
//$signal_force("/testbench_TAP/Adder/Full_Adder/genblk1[6]/Add1/out_and1","2#0"); //Fault
Injection (S@0)
//$signal_force("/testbench_TAP/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv","2#1"); //Fault
Injection (S@1)
//$signal_force("/testbench_TAP/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv","2#0");
//Fault Injection (S@0)
//$signal_force("/testbench_TAP/Adder/Full_Adder/Cout","2#1"); //Fault Injection (S@1)
//$signal_force("/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv","2#0"); //Fault
Injection (S@0)
//$signal_force("/testbench/Adder/Full_Adder/genblk1[7]/Add1/out_and1","2#1"); //Fault Injection
(S@1)
```

```
//$signal_force("/testbench/Adder/Full_Adder/genblk1[11]/Add1/xor0/out","2#0"); //Fault Injection
(S@0)
//$signal_force("/testbench_TAP/Adder/Full_Adder/genblk1[5]/Add1/out_Cout","2#1"); //Fault
Injection (S@1)
//$signal_force("/testbench/Adder/MuxOpA/genblk1[12]/Mux2to1/S1","2#0"); //Fault Injection (S@0)
  end
  always@(posedge clk, posedge rst) begin
    if (rst) begin
      TDI <= 0;
      TMS <= 0; //Test_Logic_Reset
      //test <= {1'b0, 1'b0, 1'b0, 16'd15000, 16'd3000, 16'b0};
      //       S0   S1    Cin     Op1       Op2   Feedback
      compare <= 0;
    end
    else begin
      case(cycle)
        0: begin
           TMS <= 1; cycle <= cycle + 1;
           $display("Normal Test - Vector introduce since PI");
           $display("|    %d     |    %d     | %d    |   %d    | %d  |%d |   %d  |   %d  |", S0, S1, Op1, Op2,
Feedback, Cin, Result, Cout);
         end  //Run_Test_Idle
        1: begin
           TMS <= 1; cycle <= cycle + 1;
           $display("|    %d     |    %d     | %d    |   %d    | %d  |%d |   %d  |   %d  |", S0, S1, Op1, Op2,
Feedback, Cin, Result, Cout);
         end  //Select_DR
        2: begin TMS <= 0; cycle <= cycle + 1; end  //Select_IR
        3: begin TMS <= 0; cycle <= cycle + 1; end  //Capture_IR
        4: begin TMS <= 0; cycle <= cycle + 1; TDI <= 0; end  //Shift_IR
        5: begin TMS <= 1; cycle <= cycle + 1; TDI <= 1; end  //Shift_IR
        6: begin TMS <= 1; cycle <= cycle + 1; end  //Exit_IR
        7: begin TMS <= 0; cycle <= cycle + 1; end  //Update_IR
        //Finish to capture IR
        8: begin TMS <= 1; cycle <= cycle + 1; end  //Run_Test_Idle
        9: begin TMS <= 0; cycle <= cycle + 1; end  //Select_DR
        10: begin TMS <= 0; cycle <= cycle + 1; end  //Capture_DR
        11: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[0]; end  //Shift_DR_0
        12: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[1]; end  //Shift_DR_1
        13: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[2]; end  //Shift_DR_2
        14: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[3]; end  //Shift_DR_3
        15: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[4]; end  //Shift_DR_4
        16: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[5]; end  //Shift_DR_5
        17: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[6]; end  //Shift_DR_6
        18: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[7]; end  //Shift_DR_7
        19: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[8]; end  //Shift_DR_8
        20: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[9]; end  //Shift_DR_9
        21: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[10]; end  //Shift_DR_10
        22: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[11]; end  //Shift_DR_11
        23: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[12]; end  //Shift_DR_12
        24: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[13]; end  //Shift_DR_13
```

```
        25: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[14]; end  //Shift_DR_14
        26: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[15]; end  //Shift_DR_15
        27: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[16]; end  //Shift_DR_16
        28: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[17]; end  //Shift_DR_17
        29: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[18]; end  //Shift_DR_18
        30: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[19]; end  //Shift_DR_19
        31: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[20]; end  //Shift_DR_20
        32: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[21]; end  //Shift_DR_21
        33: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[22]; end  //Shift_DR_22
        34: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[23]; end  //Shift_DR_23
        35: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[24]; end  //Shift_DR_24
        36: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[25]; end  //Shift_DR_25
        37: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[26]; end  //Shift_DR_26
        38: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[27]; end  //Shift_DR_27
        39: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[28]; end  //Shift_DR_28
        40: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[29]; end  //Shift_DR_29
        41: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[30]; end  //Shift_DR_30
        42: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[31]; end  //Shift_DR_31
        43: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[32]; end  //Shift_DR_32
        44: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[33]; end  //Shift_DR_33
        45: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[34]; end  //Shift_DR_34
        46: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[35]; end  //Shift_DR_35
        47: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[36]; end  //Shift_DR_36
        48: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[37]; end  //Shift_DR_37
        49: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[38]; end  //Shift_DR_38
        50: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[39]; end  //Shift_DR_39
        51: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[40]; end  //Shift_DR_40
        52: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[41]; end  //Shift_DR_41
        53: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[42]; end  //Shift_DR_42
        54: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[43]; end  //Shift_DR_43
        55: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[44]; end  //Shift_DR_44
        56: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[45]; end  //Shift_DR_45
        57: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[46]; end  //Shift_DR_46
        58: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[47]; end  //Shift_DR_47
        59: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[48]; end  //Shift_DR_48
        60: begin TMS <= 0; cycle <= cycle + 1; TDI <= test[49]; end  //Shift_DR_49
        61: begin
            TMS <= 1; cycle <= cycle + 1; TDI <= test[50];

$display("*************************************************************************");
            $display("\t\t\tTAP CONTROLLER");
            $display("Seed trough TAP: %b @%0d ns", test, $time);
            end  //Shift_DR_50
        62: begin TMS <= 1; cycle <= cycle + 1; end  //Exit_DR
        63: begin TMS <= 0; cycle <= cycle + 1;
                gen <= 1; //Finish to insert seed and enables LFSR Generator & Update_DR
            end
        64: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
        65: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
        66: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
        67: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
```

```
      68: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      69: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      70: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      71: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      72: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      73: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      74: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      75: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      76: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      77: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      78: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      79: begin TMS <= 0; cycle <= cycle + 1; end  //Save signature in output data register
      80: begin TMS <= 1; cycle <= cycle + 1; end  //Save signature in output data register
      //Finish to get the test and start to pass signature truogh TDO
      81: begin TMS <= 1; cycle <= cycle + 1; end  //Select_DR
      82: begin TMS <= 0; cycle <= cycle + 1; end  //Select_IR
      83: begin TMS <= 0; cycle <= cycle + 1; end  //Capture_IR
      84: begin TMS <= 0; cycle <= cycle + 1; TDI <= 1; end  //Shift_IR
      85: begin TMS <= 1; cycle <= cycle + 1; TDI <= 0; end  //Shift_IR
      86: begin TMS <= 1; cycle <= cycle + 1; end  //Exit_IR
      87: begin TMS <= 1; cycle <= cycle + 1; end  //Update_IR
      88: begin TMS <= 0; cycle <= cycle + 1; end
      //Finish to capture IR
      89: begin TMS <= 0; cycle <= cycle + 1; end
      90: begin TMS <= 0; cycle <= cycle + 1; end  //Select_DR
      91: begin TMS <= 0; cycle <= cycle + 1; end  //Capture_DR
      92: begin TMS <= 0; cycle <= cycle + 1; compare[0] <= TDO; end  //Shift_DR_0
      93: begin TMS <= 0; cycle <= cycle + 1; compare[1] <= TDO; end  //Shift_DR_1
      94: begin TMS <= 0; cycle <= cycle + 1; compare[2] <= TDO; end  //Shift_DR_2
      95: begin TMS <= 0; cycle <= cycle + 1; compare[3] <= TDO; end  //Shift_DR_3
      96: begin TMS <= 0; cycle <= cycle + 1; compare[4] <= TDO; end  //Shift_DR_4
      97: begin TMS <= 0; cycle <= cycle + 1; compare[5] <= TDO; end  //Shift_DR_5
      98: begin TMS <= 0; cycle <= cycle + 1; compare[6] <= TDO; end  //Shift_DR_6
      99: begin TMS <= 0; cycle <= cycle + 1; compare[7] <= TDO; end  //Shift_DR_7
      100: begin TMS <= 0; cycle <= cycle + 1; compare[8] <= TDO; end  //Shift_DR_8
      101: begin TMS <= 0; cycle <= cycle + 1; compare[9] <= TDO; end  //Shift_DR_9
      102: begin TMS <= 0; cycle <= cycle + 1; compare[10] <= TDO; end  //Shift_DR_10
      103: begin TMS <= 0; cycle <= cycle + 1; compare[11] <= TDO; end  //Shift_DR_11
      104: begin TMS <= 0; cycle <= cycle + 1; compare[12] <= TDO; end  //Shift_DR_12
      105: begin TMS <= 0; cycle <= cycle + 1; compare[13] <= TDO; end  //Shift_DR_13
      106: begin TMS <= 0; cycle <= cycle + 1; compare[14] <= TDO; end  //Shift_DR_14
      107: begin TMS <= 0; cycle <= cycle + 1; compare[15] <= TDO; end  //Shift_DR_15
      108: begin TMS <= 1; cycle <= cycle + 1; compare[16] <= TDO;
              $display("Signature trough TDO: %b - %d @%0d ns", compare, compare, $time);
          end  //Shift_DR_16
      109: begin TMS <= 1; cycle <= cycle + 1; end
      110: begin TMS <= 0; cycle <= cycle + 1; end
      //Finish ready for compare
    endcase
  end
end
```

```
  //Instance of DUT
  TAP TAP(.TCLK(clk), .TRST(rst), .TMS(TMS), .TDI(TDI), .TDO(TDO), .TM(TM), .DIN(din), .DOUT(dout));
  Adder Adder(.clk(clk), .rst(rst), .SELA(S0), .SELB(S1), .CI(Cin), .OpA(Op1), .OpB(Op2),
.Feedback(Feedback), .CO(Cout), .SUM(Result), .gen(gen), .TM(TM), .DIN(dout), .DOUT(din));
endmodule
```

Valores Forzados

1. "/testbench_TAP/Adder/Full_Adder/genblk1[5]/Add1/out_Cout" (S@1)
2. "/testbench_TAP/Adder/Full_Adder/genblk1[6]/Add1/out_and1" (S@0)
3. "/testbench_TAP/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv" (S@1)
4. "/testbench_TAP/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv" (S@0)
5. "/testbench_TAP/Adder/Full_Adder/Cout" (S@1)
6. "/testbench_TAP /Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv" (S@0)
7. "/testbench_TAP /Adder/Full_Adder/genblk1[7]/Add1/out_and1" (S@1)
8. "/testbench_TAP /Adder/Full_Adder/genblk1[11]/Add1/xor0/out" (S@0)
9. "/testbench_TAP/Adder/Full_Adder/genblk1[5]/Add1/out_Cout" (S@1)
10. "/testbench_TAP /Adder/MuxOpA/genblk1[12]/Mux2to1/S1" (S@0)

Comparación con la firma correcta: *La firma utilizada es la que está marcada en rojo*

Good Signature

```
# Simulation without faults
# ****************************************************************************
#                              TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011111011000011101 @615 ns
# Signature trough TDO: 01111011011100010 -  63202 @1085 ns
```

S@1: "/testbench_TAP/Adder/Full_Adder/genblk1[5]/Add1/out_Cout"

```
# Simulation with Stuck@0 & Stuck@1
# ****************************************************************************
#                              TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011111011000011101 @615 ns
# Signature trough TDO: 00100001101100011 -  17251 @1085 ns
```

S@0: "/testbench_TAP/Adder/Full_Adder/genblk1[6]/Add1/out_and1"

```
# Simulation with Stuck@0 & Stuck@1
# ****************************************************************************
#                              TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011111011000011101 @615 ns
# Signature trough TDO: 01111011011100010 -  63202 @1085 ns
```

S@1: "/testbench_TAP/Adder/Full_Adder/genblk1[7]/Add1/xor0/in0_inv"

```
# Simulation with Stuck@0 & Stuck@1
# ****************************************************************************
#                              TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011111011000011101 @615 ns
# Signature trough TDO: 01111101011100011 -  64227 @1085 ns
```

S@0: "/testbench_TAP/Adder/Full_Adder/genblk1[11]/Add1/xor0/and_in0_inv"

```
# Simulation with Stuck@0 & Stuck@1
```

```
# ********************************************************************************
#                                TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011110110000 1101 @615 ns
# Signature trough TDO: 011110110111000 10 -  63202 @1085 ns

S@1: "/testbench_TAP/Adder/Full_Adder/Cout"
# Simulation with Stuck@0 & Stuck@1
# ********************************************************************************
#                                TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011110110000 1101 @615 ns
# Signature trough TDO: 011110110111000 10 -  63202 @1085 ns
S@0: "/testbench_TAP/Adder/Full_Adder/genblk1[11]/Add1/xor1/in0_inv"
# Simulation with Stuck@0 & Stuck@1
# ********************************************************************************
#                                TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011110110000 1101 @615 ns
# Signature trough TDO: 010100110111000 10 -  42722 @1085 ns

S@1: "/testbench_TAP /Adder/Full_Adder/genblk1[7]/Add1/out_and1"
# Simulation with Stuck@0 & Stuck@1
# ********************************************************************************
#                                TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011110110000 1101 @615 ns
# Signature trough TDO: 001001000111000 11 -  18659 @1085 ns

S@0: "/testbench_TAP /Adder/Full_Adder/genblk1[11]/Add1/xor0/out "
# Simulation with Stuck@0 & Stuck@1
# ********************************************************************************
#                                TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011110110000 1101 @615 ns
# Signature trough TDO: 011010110111000 10 -  55010 @1085 ns

S@1: "/testbench_TAP/Adder/Full_Adder/genblk1[5]/Add1/out_Cout"
# Simulation with Stuck@0 & Stuck@1
# ********************************************************************************
#                                TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011110110000 1101 @615 ns
# Signature trough TDO: 001000011011000 11 -  17251 @1085 ns

S@0: "/testbench_TAP /Adder/MuxOpA/genblk1[12]/Mux2to1/S1"
# Simulation with Stuck@0 & Stuck@1
# ********************************************************************************
#                                TAP CONTROLLER
# Seed trough TAP: 01101111110011001100011010010111001011110110000 1101 @615 ns
# Signature trough TDO: 011110110111000 11 -  63203 @1085 ns
```