

ANEXO A

GUÍA DE USUARIO XISCKER

1. INTRODUCCIÓN

Esta guía de usuario hace parte del desarrollo de una “Plataforma para la Emulación y Reconfiguración de Arquitecturas RISC y CISC” en FPGA (Field-Programmable Gate Arrays) como herramienta académica clave en el área de arquitectura y organización de computadores, llamada X-ISCKER por sus siglas en inglés, “Reduced/Complex Instruction Set Computing Key Educational Resource”; elaborada como tesis de pregrado en la Universidad Pontificia Bolivariana Seccional (UPB) Bucaramanga por Alfredo Gualdrón y Jose Pablo Pinilla.

La plataforma X-ISCKER nace de la necesidad de una herramienta de apoyo en la enseñanza de arquitectura y organización de computadores, que ayude a la comprensión de la materia y al mismo tiempo sea implementada en el desarrollo de aplicaciones tangibles en campos como la robótica, el control digital, instrumentación, comunicaciones, entre otros.

La plataforma X-ISCKER está compuesta por la descripción de hardware en lenguaje Verilog de dos microprocesadores, una interfáz de comunicación UART y un software IDE (Integrated Development Environment) que cumplen tres funciones principales: Programación, Emulación y Reconfiguración, con el fin de facilitar el desarrollo de proyectos basados en la tecnología de procesadores embebidos en FPGA, conformando una metodología de diseño que integra el desarrollo de software, la aplicación de una herramienta de depuración y el diseño avanzado de hardware.

2. OBJETIVOS

- Utilizar la plataforma X-ISCKER IDE con el fin de observar el comportamiento de un procesador de arquitectura CISC y uno RISC.
- Utilizar la plataforma como un sistema de procesamiento independiente para aplicaciones en el campo de la electrónica digital.
- Reconfigurar la plataforma para implementar instrucciones y funciones nuevas a cualquiera de las dos arquitecturas.

3. RECURSOS DISPONIBLES

Plataforma para la Emulación y Reconfiguración de Arquitecturas RISC y CISC: Libro principal del proyecto. Contiene la descripción de los objetivos, el soporte teórico de cada arquitectura y la metodología de diseño.

Hojas de Datos: Contiene información detallada de la estructura y funcionamiento de los procesadores RISCKER y CISCKER.

Wiki: Sitio web diseñado para el soporte y continuo desarrollo de la herramienta. Contiene toda la información relacionada con el proyecto.

4. ARQUITECTURA DE LOS MICROPROCESADORES

El siguientes es un cuadro comparativo de los dos microprocesadores disponibles para implementar en la plataforma.

Tabla 1. Comparación de las arquitecturas RISCKER y CISCKER

Característica	RISCKER	CISCKER
# de Instrucciones	30	244
# de Registros	64	4
ALU	16 bits	16bits
Arquitectura de memoria	Harvard	Von Neumann
Tamaño de las instrucciones	Fijo: 16-bits	Variable: 8 a 24-bits
Multiplicación y División	Unidad de hardware dedicada	Iteraciones
Operaciones "Shift"	Logica, Aritmetica y Rotación de 1 a 16 bits por instrucción.	Logica, Aritmetica y Rotación de 1 bit por instrucción.
Modos de Direccionamiento	Inmediato,Directo, Indirecto,Desplazado	INH,REL,IMM,DIR,EXT,IX,I X+
Tamaño máximo de memoria	65536x32bits 65536x16bits	65536x16bits
# de condiciones de salto	2	14
Temporizadores	2 de 16 bits	2 de 16 bits
Interrupciones	2 por Temporizador 2 Externas	2 por Temporizador 2 Externas
Unidad de Control	HCU	MCU
Arquitectura similar	MIPS	HC08 y HC11 Freescale

En las hojas de datos se presentan el set de instrucciones, diagramas de bloques y diagramas esquemáticos del diseño digital de los módulos de cada uno de los microprocesadores, contienen toda la información, de arquitectura y organización, necesaria para entender el funcionamiento general del hardware de la plataforma descrito en modulos Verilog.

5. XILINX SPARTAN 3A & ALTERA DE0-NANO

El desarrollo de esta plataforma se llevo a cabo utilizando dos kits con FPGA: el primero es una tarjeta "Spartan 3A Starter Kit" que posee varios periféricos integrados y pocos pines GPIO (General Purpose Input/Ouput), ideal para utilizar

dispositivos como: Display LCD, encoder, ADC, DAC, entre otros. La segunda tarjeta es el kit de desarrollo DE0-Nano, la cual ofrece mayor flexibilidad gracias a la gran cantidad de GPIO disponibles.

La plataforma ha sido desarrollada para soportar ambos kits simplemente seleccionando el dispositivo con el cual se está trabajando, esto corresponde inicialmente a los dispositivos Xilinx Spartan 3A XC3S700A y Altera Cyclone IV EPC22F17C6, de manera que es posible que la plataforma no funcione en dispositivos de los mismos fabricantes pero de otras familias. Para lograr compatibilidad con otros dispositivos se recomienda primero, revisar las necesidades de hardware del sistema como capacidad del FPGA, memoria y bloques DSP y GPIO. Segundo, se debe modificar el software XISCKER Programmer o utilizar la herramienta de programación proporcionada por el fabricante del dispositivo.

6. ARQUITECTURA DE LA PLATAFORMA

La plataforma consta de los siguientes dispositivos y software:

- Spartan 3A Starter Kit o DE0-Nano Development board.
- Dispositivo de comunicación UART y Drivers.
- Módulos Verilog de los microprocesadores RISCKER y CISCKER.
- Módulos Verilog RISCKER y CISCKER con interfáz de comunicación UART-Observer.
- Software de Altera Quartus II 11.0 o superior ó Software de Xilinx ISE 13.0 o superior.
- Software X-ISCKER IDE (Integrated Development Environment)
- PC Windows con conexión USB y .NET Frameworks 4.0 o superior.

6.1. XISCKER IDE

Para instalar el software X-ISCKER IDE se debe descargar la última versión disponible del sitio: semilleroadt.upbbga.edu.co/XISCKER, ejecutar la aplicación y seguir los pasos del software de instalación.

Durante la instalación se crea la carpeta C:/XISCKER en la cual se almacena la dirección del último espacio de trabajo utilizado. Al ejecutar el programa por primera vez aparecerá un dialogo para crear la carpeta que contendrá toda la información de la última sesión de trabajo: como son la arquitectura seleccionada, el código assembler, el archivo de inicialización de memoria y la configuración de programación. A esta carpeta se le llama *Workspace*.

Al crear un nuevo workspace se genera automáticamente una subcarpeta llamada "sys" con los siguiente archivos:

- AlteraCableItems.item: Contiene un listado de dispositivos de conexión para la programación del FPGA.
- AlteraFamilyItems.item: Contiene un listado con las posibles familias de FPGA Altera para utilizar en el XISCKER Programmer.
- LastAsmSession.bak: Contiene las direcciones del último archivo fuente en lenguaje ensamblador, el código de máquina, tipo de arquitectura y la dirección del espacio de trabajo.
- LastPrgmSession.bak: Contiene la informacion de cada uno de los campos elegidos en la última sesión del XISCKER programmer.
- XilinxFamilyItem.item: Contiene un listado con las posibles familias de FPGA Xilinx para utilizar en el XISCKER Programmer.

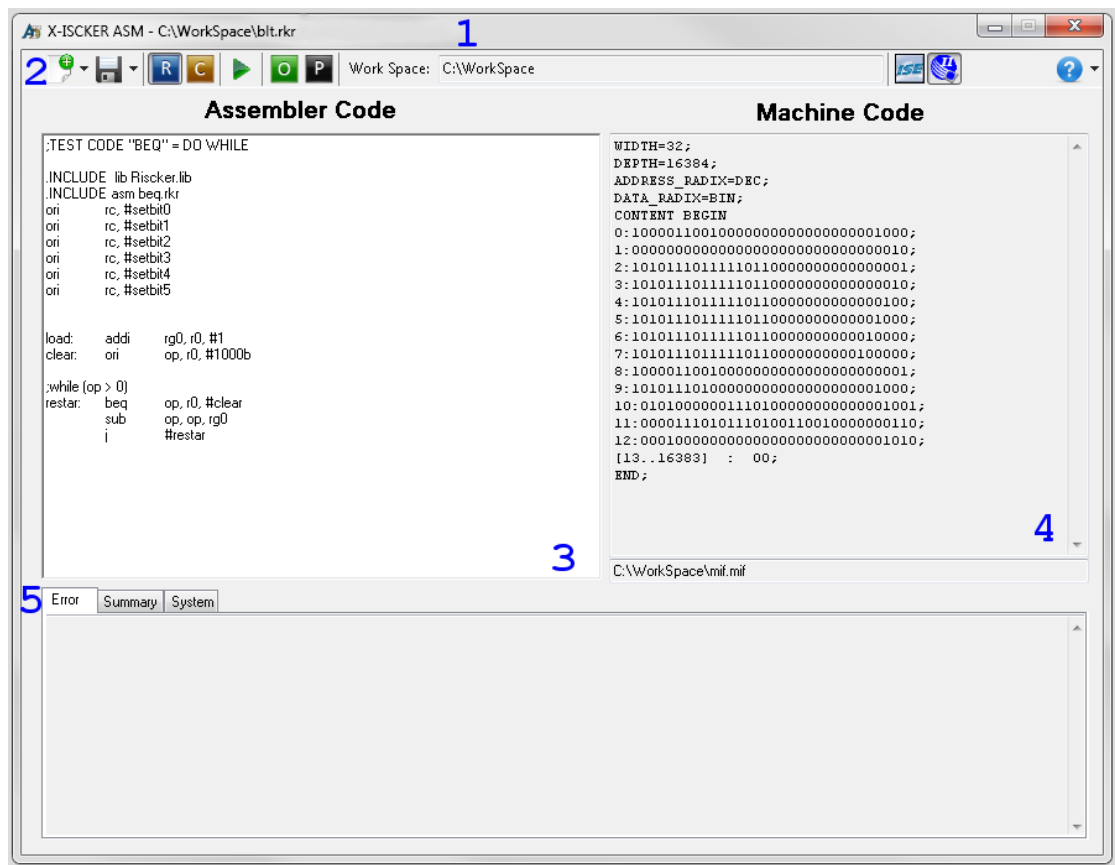
Los demás archivos correspondientes a cada proyecto que se trabaje en el XISCKER IDE son:

- *.rkr: Archivo de texto plano con código en lenguaje RISCKER ASM.
- *.ckr: Archivo de texto plano con código en lenguaje CISCKER ASM.
- *.mif , *.mem o *.dat: Archivos de inicialización de memoria. Contienen datos en binario o en hexadecimal que se escriben directamente a las memorias ROM o RAM del diseño en FPGA.








Cada uno de estos archivos puede tener una ubicación distinta, ya que pertenecen al proyecto actual del usuario y no directamente al software, sin embargo, se recomienda utilizar el espacio de trabajo actual.



A continuación se muestra una imagen con la GUI de la ventana principal del X-ISCKER IDE y una enumeración de cada una de las funciones que ofrece:


Figura 1. GUI de la ventana principal del software X-ISCKER IDE.



1. **Barra de Título:** Muestra la dirección del archivo “*.ckr” o “*.rkr” que se está modificando.
2. **Barra de Herramientas:**

- a.  Abrir: El botón abre un nuevo archivo en blanco para empezar a editar en la caja de código. Ofrece un menú con tres opciones.
- RISCKER ASM: Abre un archivo de extensión .rkr y selecciona la arquitectura para el IDE como RISCKER.
 - CISCKER ASM: Abre un archivo de extensión .ckr y selecciona la arquitectura para el IDE como CISCKER.
 - New blank file: Crea un archivo vacío, es la acción por defecto del botón.
- b.  Guardar: El botón guarda el contenido de la caja de código y de memoria en archivos existentes o consulta la ubicación para crearlos.
- Assembly File: Guarda el archivo contenido en la caja de código con extensión .ckr o .rkr según sea la arquitectura seleccionada.
 - Memory File: Guarda el contenido de la caja de memoria en un archivo de inicialización de memoria *.mif o *.mem.
- c. Selector de Arquitectura  RISCKER  CISCKER: Selección del algoritmo de ensamble para la arquitectura de set de instrucciones RISCKER/CISCKER. Esta selección también modifica el ambiente de desarrollo XISCKER IDE de manera que el código se guarde en un archivo “.rkr” o “.ckr” y para identificar la interfaz con la que se va a lanzar el XISCKER Observer (detallado en el numeral 3.3).
- d.  Ejecutar ensamblador: Inicia la interpretación del programa en lenguaje de ensamblador que se encuentra en la caja de código (Assembler Code), genera el archivo de inicialización de memoria de instrucciones y lo muestra en la caja de memoria (Machine Code).
- e.  Programmer: Lanza una ventana adicional correspondiente a la herramienta X-ISCKER Programmer (detallada en el numeral 3.2).
- f.  Observer: Lanza una ventana adicional correspondiente a la herramienta X-ISCKER Observer (detallada en el numeral 3.3).
- g. Workspace: Muestra la ruta del espacio de trabajo actual.

h.  Xilinx  Altera: Al elegir uno de los dos fabricantes se modifican los parámetros del XISCKER Programmer y el formato en que se crea el archivo de memoria.

i.  Ayuda: Este botón lanza una ventana con información acerca del XISCKER IDE.

- User Guide: Esta guía de usuario.
- Wiki: Enlace de la wiki XISCKER, contiene los archivos y avances más recientes del proyecto.
- About: Lanza la ventana con información acerca de la versión y desarrollo del software XISCKER IDE. Es la opción por defecto del botón.

3. **Caja de Código:** Caja de texto para editar el código en lenguaje ensamblador de la arquitectura seleccionada.

4. **Caja de memoria:** Caja de texto que muestra el archivo de inicialización de memoria generado por el proceso de ensamblado. En la parte inferior se muestra la ruta del archivo que se está mostrando.

5. **Pestañas de Procesos:** Pestañas que muestran información de los procesos de ensamblado y configuración del dispositivo.

- a. Error: Muestra las líneas de código con errores generados durante el proceso de ensamblado o la programación del dispositivo. El número de la línea se especifica en la pestaña Summary.
- b. Summary: Muestra un resumen de las diferentes etapas del proceso de ensamblado. Asigna un número a cada línea de código para facilitar su depuración.
- c. System: Muestra información sobre los comandos para generar el archivo de configuración del dispositivo, los errores y las advertencias que estos generan.

6.2. XISCKER Programmer

Es la herramienta de XISCKER IDE que permite programar el FPGA con el microprocesador de la arquitectura seleccionada. Es necesario tener instalado el IDE de la tecnología del dispositivo que se va a programar (Quartus-Altera, ISE-Xilinx). Esta herramienta tiene dos modos de funcionamiento:

Update Memory: Actualiza únicamente los bloques relacionados a la memoria de instrucciones del procesador y programa el FPGA. Es la opción más rápida.

Compile Project: Compila el proyecto seleccionado (Synthesis, Assembly, Place and Route) y programa el FPGA. Sólo es necesaria cuando el proyecto se compila por primera vez después de una modificación en el hardware. Es la opción más lenta.

A continuación se muestra una imagen de la ventana de diálogo del X-ISCKER Programmer y una enumeración de cada una de las funciones que ofrece:

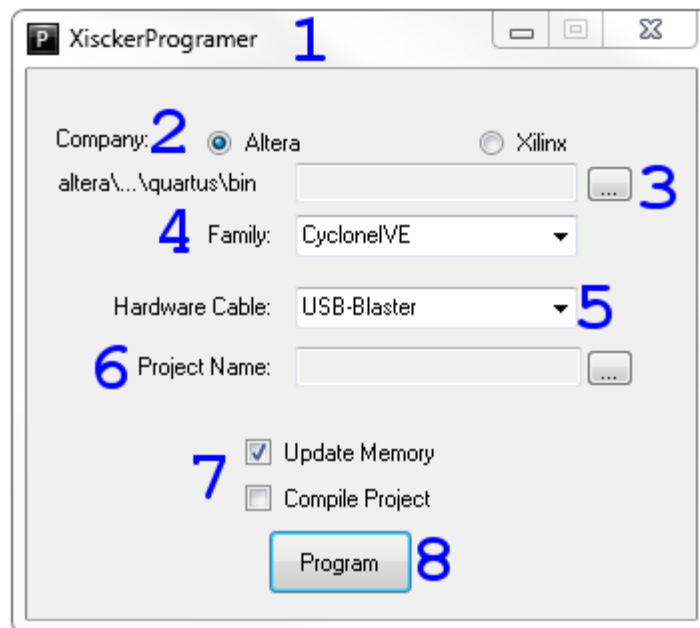


Figura 2. GUI de la herramienta de software X-ISCKER Programmer

1. **Barra de título:** Muestra el nombre del programa.
2. **Company:** Selección de la marca del dispositivo que desea configurar.
3. **Folder:** Selección de la carpeta “bin”, en la cual se encuentran los ejecutables necesarios para configurar el dispositivo (Carpeta de instalación de Quartus o Xilinx ISE, según la tarjeta de desarrollo).
4. **Family:** Selección de la familia o nombre del dispositivo.
5. **Hardware cable:** Selección del driver de configuración del dispositivo. Para Xilinx esta selección es automática.

6. Project Name:

- Para implementación con dispositivos Altera: Selección de la ubicación del proyecto (Archivo de extensión “.qpf”).
- Para implementación con dispositivos Xilinx: Selección de la ubicación del módulo principal (Archivo Verilog o VHDL) del proyecto.

7. Update Memory/Compile Project: Selección de los procesos que se desean llevar a cabo.

8. Program: Botón para iniciar la secuencia de programación.

6.3. XISCKER Observer

El XISCKER Observer es una interfaz que permite la visualización periódica de las señales más relevantes del microprocesador programado en el FPGA mediante etiquetas sobrepuestas a un diagrama de la arquitectura seleccionada, como lo muestran las figuras 3 y 4.

El registro de las señales adquiridas puede almacenarse en un archivo *.csv para una visualización y análisis posterior. A continuación se explican los distintos componentes presentes en la GUI del Observer RISCKER y CISCKER.

1. Barra de título: Muestra el nombre del programa y la arquitectura actual.

2. Barra de Herramientas: Para ambas configuraciones es la misma. En esta barra se hace la configuración del puerto VCP que desea utilizar para comunicarse con el microprocesador seleccionado y las opciones de conexión y registro de sesión. La barra de herramientas contiene los siguientes items:




- a.  Conectar  Desconectar: Botón de conexión y desconexión del puerto COM. El icono cambia indicando el estado del puerto. Al iniciar la comunicación se deshabilitan las opciones de configuración del puerto.
- b.  Definir archivo de registro: Botón para definir la ubicación del archivo de registro de sesión. Al momento de seleccionar la ubicación, el programa creará e inicializará una fila de un documento *.csv.

Figura 3. Imagen de la GUI del Observer RISCKER.

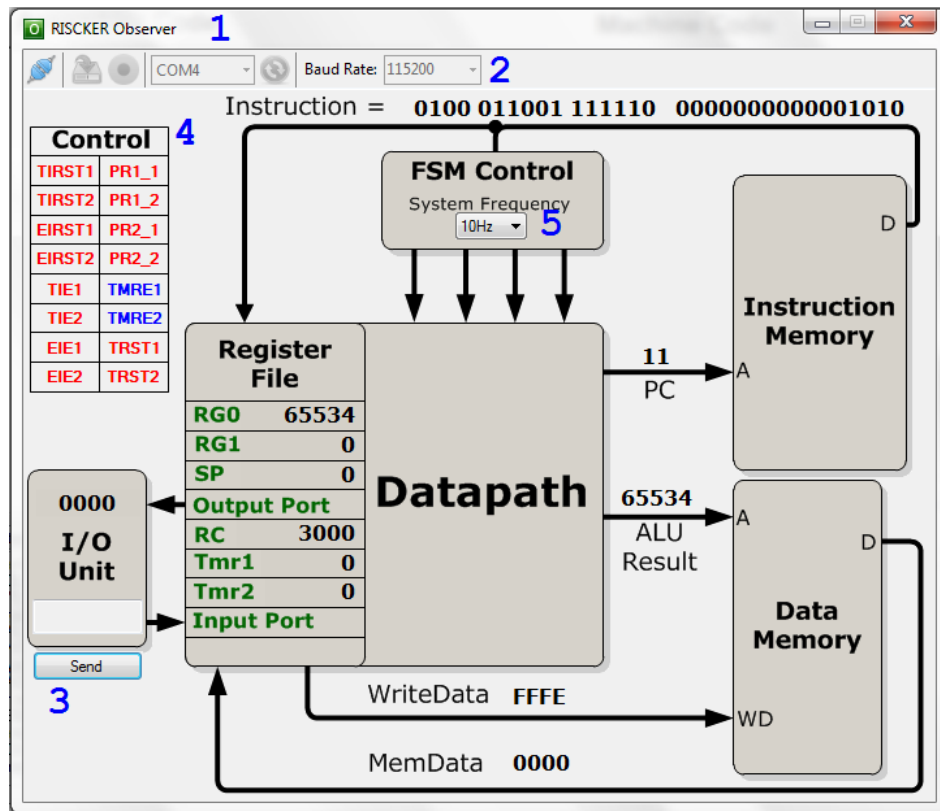
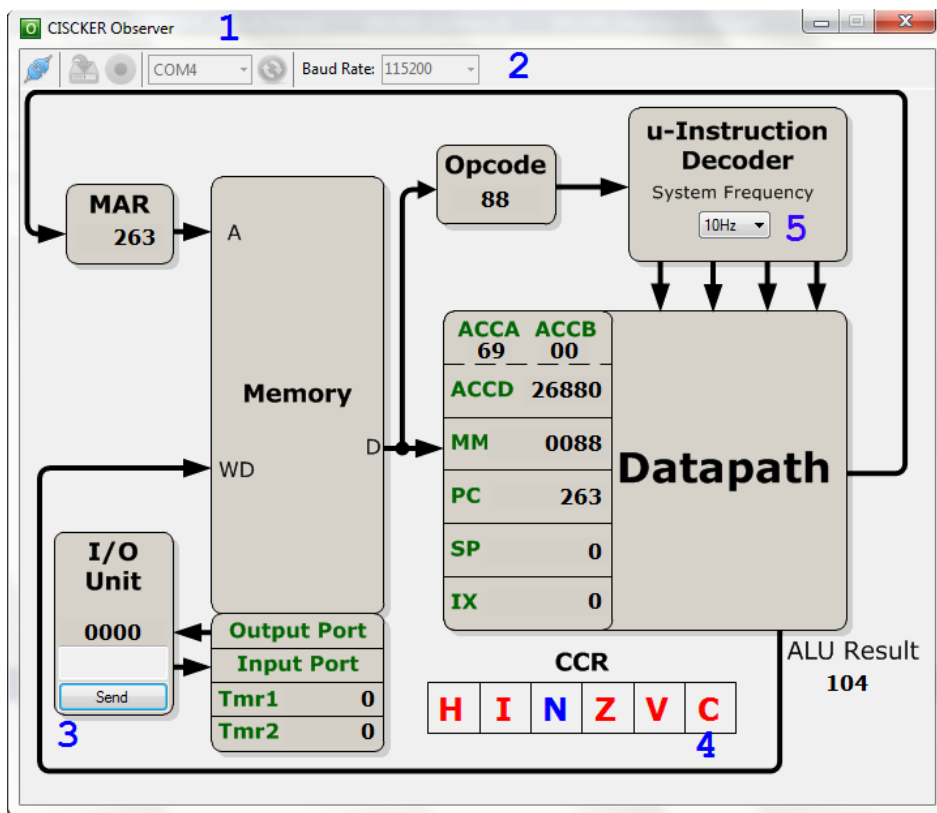





Figura 4. Imagen de la GUI del Observer CISCKER.



- c.  Grabar  Pausa: Botón para activar o desactivar el registro de la sesión que va a ser o está siendo ejecutada. El icono cambia indicando la opción de pausar el registro actual o reanudarlo.
- d. Listado de puertos COM: Muestra una lista de puertos COM disponibles en su sistema para seleccionar como medio de comunicación con la plataforma.
- e.  Actualizar puertos COM: Botón para actualizar el listado de puertos COM en caso de conectar el dispositivo después de la inicialización del programa.
- f. Tasa de Baudios: Muestra una lista de selección de tasa de baudios predeterminados. Permite edición para tasas de baudios personalizadas.

3. **Puerto de Entrada:** XISCKER Observer permite controlar e interactuar con el microprocesador implementado. Este campo envía la palabra de 16 bits escrita en hexadecimal (FFFFh), binario (1010101010101010b), octal (177777o) o decimal (65535) a un registro conectado directamente al puerto de entrada del microprocesador implementado.
4. **CCR (Condition Code Register) / Registro de Control:** Las casillas muestran el registro de códigos de condición (CCR) para el microprocesador CISCKER o el registro de control (RC) en el microprocesador RISCKER. El color rojo corresponde al valor zero y el azul al valor uno. En las hojas de datos de cada procesador encontrará más información sobre cada una de las señales que aparecen en las casillas.
5. **Configuración de frecuencia:** Esta lista muestra distintas frecuencias a las cuales se puede operar el microprocesador implementado y la opción de detenerlo. Las frecuencias que se despliegan en la lista aseguran un rango de observabilidad y de comunicación fiable entre el FPGA y el computador personal.

A continuación se muestra una tabla con las señales recibidas de cada microprocesador:

Tabla 2. Señales transmitidas por el hardware UART desde el FPGA para cada arquitectura.



RISCKER	CISCKER
Program Counter (PC)	Program Counter (PC)
Instruction (Opcode Operands Function)	Memory Address Register (MAR)
Arithmetic-Logic Unit Register (ALU Result)	Operation Code (Opcode)
Memory Data (MemData)	Memory Data (MM)
Write Memory Data (WriteData)	Arithmetic-Logic Unit Result (ALU Result)
Register General purpose 0 (RG 0)	Accumulator D (ACCD)
Register General purpose 1 (RG 1)	Index Register (IX)
Stack Pointer (SP)	Condition Code Register (CCR)
Output Port	Stack Pointer (SP)
Control Register (RC)	Output Port (I/O Unit)
Timers 1 y 2 (TMR1, TMR2)	Timers 1 y 2 (TMR1, TMR2)

6.4. Programación en XISCKER ASM

Para programar un procesador de la plataforma, una vez escritos los algoritmos en el lenguaje ensamblador apropiado para cada arquitectura, es necesario generar el archivo de inicialización de memoria (*.mif o *.mem) dando click en el botón ejecutar ensamblador (Play), o presionar la tecla F5. Una vez generado el código de máquina, éste debe ser guardado en la carpeta del proyecto en el que se encuentre instanciado el procesador y reemplazar el archivo de la misma extensión. En caso que se quiera modificar la ubicación de este archivo, se debe actualizar ese cambio en el proyecto de hardware *.qpf o *.ise para que el módulo de memoria de programa reconozca la nueva ubicación del código de máquina.

Los recursos correspondientes a esta etapa son el software X-ISCKER IDE y las hojas de datos de cada microprocesador, en las cuales se encuentra la información de la ISA (Instruction Set Architecture) completa desde el punto de vista del programador que le permite conocer todas las instrucciones y módulos internos para utilizar, como puertos, temporizadores e interrupciones.

6.4.1. Procedimiento

1. Ejecute la aplicación X-ISCKER IDE
2. Si es la primera vez que ejecuta el programa, seleccione una carpeta, en la cual se crearán los archivos relacionados con el proyecto que vaya a realizar (Workspace).
3. Seleccione la arquitectura con la que va trabajar
4. Escriba el código en lenguaje ensamblador
5. Haga click en  “Run” (F5), Para generar el archivo de inicialización de memoria.
6. Guarde el archivo de inicialización de memoria en la carpeta que contiene el proyecto del microprocesador (.qpf ó .ise).
7. Haga click en  “Programmer” (F7). En la nueva ventana:
 - a. Seleccione la dirección de la carpeta bin, según sea Altera o Xilinx.
 - b. Seleccione la Familia de dispositivo que desea configurar.
 - c. Seleccione el Cable de la tarjeta correspondiente (Xilinx es Automático).
 - d. Seleccione la dirección del proyecto donde está instanciado el procesador.
 - Para dispositivos Altera: Seleccione el archivo del proyecto (*.qpf).
 - Para dispositivos Xilinx: Seleccione el archivo del módulo principal del proyecto (Verilog o VHDL).
 - e. Haga click en “Program”

Nota: Para dispositivos Altera asegurese de haber generado el archivo con extensión “.cdf” desde la herramienta “Programmer” del software Quartus en la opción “Save” del menú “File”.

- f. En la pestaña “System” aparecerá el resultado de los comandos ejecutados y en la pestaña “Error” las líneas que indican algún error ocurrido en el proceso.

6.5. Emulación en XISCKER ASM





El objetivo de esta etapa es la observación del comportamiento de un microprocesador, es decir, el usuario programa una rutina utilizando los recursos ofrecidos, implementa el microprocesador seleccionado en el FPGA, comunica este sistema con un computador personal y visualiza el estado real de las señales más relevantes del circuito.

A pesar de que esta configuración es totalmente funcional en el sentido de ejecución de algoritmos de cualquier complejidad, el usuario se va a ver limitado en frecuencia de funcionamiento del sistema y en accesibilidad a los puertos I/O de los microprocesadores, ya que estos parámetros están sujetos al software XISCKER Observer. Esto da la oportunidad de comprender internamente la plataforma, observando detalladamente el comportamiento de las señales dispuestas en cada arquitectura, antes de continuar con una utilización más avanzada.


Para realizar la emulación del microprocesador RISCKER ó CISCKER se requiere el software X-ISCKER IDE y los archivos de descripción de hardware correspondientes a esta etapa, aquellos que contienen la descripción del microprocesador seleccionado junto a un circuito de comunicación UART para ser utilizado con cualquier interfáz física de interconexión bidireccional con un computador personal (VCP ó RS232).

Esta etapa es la más importante en el reconocimiento de la plataforma, ya que se utilizarán todas las herramientas que dispone el X-ISCKER IDE y requiere de un conocimiento básico previo de la arquitectura y set de instrucciones expuestas en las hojas de datos de los microprocesadores.

6.5.1. Procedimiento

1. Haga click en  “Observer” para abrir la herramienta desde el XISCKER IDE.
2. Asegúrese que el diagrama de hardware corresponda a la arquitectura que está trabajando.
3. Conecte el la tarjeta de desarrollo con FPGA al computador personal en modo serial. Para esto se disponen los siguientes pines:
 - a. Terasic DE0-Nano: Rx=PIN_A3 Tx=PIN_C3
 - b. Spartan 3A Starter Kit: Rx=V14 Tx=V15
4. Configure la conexión serial en la barra de herramientas del XISCKER Observer.
 - a. Puerto COM
 - b. Tasa de Baudios
5. (Opcional) Elija el archivo *.csv en que desea tener el registro de los datos adquiridos ().
6. (Opcional) Active el almacenamiento de datos del puerto serial con el botón de grabar ().
7. Inicie la adquisición haciendo click en el botón de conexión ()

Nota: Si elije almacenar los datos recibidos y el hardware implementado en el FPGA ya contiene una comunicación serial, estos primeros datos se verán reflejados en las primeras líneas del registro *.csv.

8. Complete el procedimiento de programación de cualquier procesador XISCKER.
9. Utilice el menú que se encuentra en la unidad de control del diagrama para cambiar la frecuencia de funcionamiento del procesador o para detenerlo.
10. Utilice el campo en la unidad I/O del diagrama para enviar datos de 16 bits al puerto de entrada del procesador.
11. Una vez terminada la sesión, desconecte la comunicación serial ().
12. Para abrir el archivo *.csv en un visor/editor de hojas de calculo es

importante definir las columnas de los datos binarios como texto, de lo contrario el programa elimina los ceros a la izquierda afectando la lectura inicial.

6.6. Reconfiguración del Hardware XISCKER

El conocimiento de la plataforma en esta etapa requiere una profundización en la descripción en Verilog que está disponible en la Wiki o en el sitio del Semillero ADT del microprocesador seleccionado. Los esquemas que también hacen parte de las hojas de datos y documentación de este proyecto son una descripción detallada de la organización de hardware de los microprocesadores y los módulos anexos, es por esto que se recomienda estudiarlos junto a la descripción de hardware para tener una visión estructurada del diseño y funcionamiento de todo el sistema.

Esta etapa no tiene un objetivo específico pero sí implica reconfigurar la plataforma para implementar nuevas instrucciones o funciones. Por ejemplo: Agregar un número cualquiera de módulos PWM al procesador para hacer un controlador de servomotores de forma paralela ó acelerar una instrucción en hardware que obtenga el valor medio entre dos variables.

7. BIBLIOGRAFÍA

Wiki: Sitio web con los archivos necesarios para trabajar con la plataforma XISCKER, documentación y evolución del proyecto.
semilleroadt.upbbga.edu.co/XISCKER.

Sitio web del Semillero de Tecnologías Avanzadas ADT de la Universidad Pontificia Bolivariana seccional Bucaramanga. <https://semilleroadt.upbbga.edu.co>

FREESCALE SEMICONDUCTOR. CPU08 Central Processor Unit. 2006.

FREESCALE SEMICONDUCTOR. 68HC11 Technical Data. 2001.

HARRIS, David Money; HARRIS, Sarah L. Digital design and computer architecture. San Francisco. Morgan Kaufmann Publishers, Elsevier, 2007.

SHIVA, Sajjan G. Computer Organization, Design, and Architecture. 4a Edición.