# Introducción al uso del procesador LatticeMico32

José Alejandro Logreira Ávila Código: 261722
David Ricardo Martínez Hernández Código: 261931
Edwin Fernando Pineda Vargas Código: 262100

*Palabras clave*— **Bits, Datapath, Makefile, Memoria, MIPS, Pipeline, Síntesis, Variables.**

***Resumen*— Se realizo la instalación de todas las herramientas necesarias en el sistema operativo Linux, distribución Ubuntu, basándonos en la pagina principal de Linux En Caja para la instalación de dichas herramientas. Realizando todas las actividades propuestas en la guía de laboratorio, obteniendo algunos de los resultados de manera eficiente y completa, ya que existieron muchas dificultades al momento de instalar y utilizar las mismas.**

## I. Sistemas operativos basados en el núcleo Linux

### A. Comandos de Linux

- **Man**: Son los manuales de las instrucciones para todos los comandos existentes en Linux. El parámetro es **man** <***comando***>.
- **Sudo**: Es un comando que identifica al usuario como administrador (Usuario con los máximos privilegios), por lo tanto tiene un acceso total al sistema. Sus parámetros son **sudo** <***opción***> <***comando***>.
- **apt-get**: Con éste comando podremos buscar paquetes en nuestros repositorios. Parámetros **apt-get** <***acción***> <***paquete***>.
- **ls**: Muestra en la consola los archivos o carpetas del directorio en donde se encuentra, en diferentes colores dependiendo los permisos que tenga cada archivo o carpeta. Parámetros **ls** <***opción***> <***archivo***>.
- **cd**: Comando utilizado para cambiar de directorio, existen dos identaciones:
  - **cd** <**directorio**>: esta identación permite acceder a un directorio y se puede escribir la dirección completa del mismo.
  - **cd ..** : esta identación permite salir del directorio.
- **mkdir**: Crea un directorio que no existe. Parametros **mkdir** <***opción***> <***directorio***>
- **wget**: Herramienta para descarga no interactiva de archivos desde servidores HTTP,HTTPS y FTP. Es decir, permite descarga de archivos individuales o webs completas. Posee soporte para proxies, soporta conexión Ipv6 y descarga de grandes archivos, por encima de 2GB. Su síntesis es: wget [opciones] ... [URL] ...
- **git**: Es el nombre tanto de un comando, como de un programa de control del sistema diseñado para gestionar proyectos de diversas magnitudes. Posee el histórico completo de las acciones del sistema y lleva registro y control de versiones de aplicaciones.
- **make**: gestiona la ejecución de grupos de programas, y los distintos recursos que cada programa usa. Identifica qué parte de cada programa necesita ser recompilado y ejecuta la recompilación.
- **echo**: Despliega una línea de texto dentro de la línea de comandos indicada.
- **grep**: despliega líneas de texto ubicadas en archivos de texto en direcciones especificadas, que coinciden con un patrón de entrada.

### B. Variables de entorno

Un shell en Linux es un intérprete de comandos por consola. Esto es, las instrucciones no necesitan previa compilación. Existen dos intérpretes de comandos:

- bash
- dash

Una gran cantidad de comandos son compartidos por ambos intérpretes, más sin embargo existen algunas diferencias de tipeo en las instrucciones comunes, así como funciones especializadas sólo presentes en uno de los dos intérpretes. La variable $SHELL contiene la ruta al shell por defecto que usa linux: Bash.

#### 1) Ejemplos de Variables de entorno:

- DESKTOP_SESSION=ubuntu. Especifica el tipo de visualización del lanzador, si es el original de Ubuntu, o si es Gnome, etc.
- HOME=/home/USER. Define la ruta de acceso al directorio de usuario, donde se ubican todos los documentos personales.
- LANG=en_US.UTF-8. Define el idioma del sistema y el tipo de codificación de los caracteres.
- LOGNAME=USER. Define el nombre del usuario.

## II. Archivos del LM32 y el proceso para sintetizar

### A. Proceso de Síntesis y Compilación

Los archivos llamados Makefile son ficheros de texto plano que organizan la ejecución de múltiples programas para la creación de archivos objeto, resolviendo dependencias entre los archivos usados y, en nuestro caso, creando un único archivo binario que será el que se cargará directamente a la FPGA.

El archivo Makefile que realiza la compilación del procesador es el encargado de sintetizar todo el procesador y grabar en su memoria el programa que se desee ejecutar. Mediante la ejecuión por consola del comando $ make >> secuencia_compilado_porcesador, se crea un archivo que contiene una copia de todos los comando ejecutados en forma organizada, es decir, que se aprecia el orden real de ejecución de las

instrucciones con las dependencias resueltas. A continuación se lista el orden de ejecución del Makefile:
(Nota: los dos puntos ":" se usan para identificar qué archivos son necesarios en cada paso de ejecución)

- $ make >> secuencia_compilado

# Ikarus verilog simulation

1) system_tb.vvp
2) %.vcd: %.vvp

Esta ejecución da a entender que solo se ejecuta la simulación en verilog del testbench precargado.

- $ make syn >> secuencia_compilado_sintesis

# ISE Sýnthesis

1) system.prj
2) system.ngc:system.prj. Crea las carpetas build y simulation
   Table of contents
   a. Synthesis Options Summary
   b. HDL compilation
   c. Design Hierarchy Analysis
   d. HDL Analysis
   e. HDL Synthesis
   f. Advannced HDL Synthesis
   g. Low Level Synthesis
   h. Partition Report
   i. Final Report
3) system.ngd: system.ngc system.ucf
4) system.ncd: system.ngd
5) system-routed.ncd: system.ncd
6) system.bit: system-routed.ncd

Esta ejecución corre todo el proceso de sintesis de Xilinx para crear el archivo binario, incluyendo el programa que se vaya a cargar en memoria.

- $ make sim >> secuencia_compilado_simulación

# Ikarus verilog simulation

1) system_tb.vvp
2) %.vcd: %.vvp

Esta ejecución muestra que al correr la simulación, se ejecuta la misma secuencia que cuando se ejecuta el comando "make" solo.

- $ make view >> secuencia_compilado_sim_visualización

# Ikarus verilog simulation

- system_tb.vvp
- %.vcd: %.vvp

#Final targets
%.view: %.vcd. Abre el visualizador GTKWave
A continuación se describe el proceso de ejecución del Makefile asociado al ejemplo boot0_serial:

- $ make >> secuencia_compilado_boot0_serial

1) crt0ram.o: crt0ram.s
2) main.o: main.c
3) soc-hw.o: soc-hw.c
4) image: crt0ram.o main.o soc-hw.o
5) image.lst: image
6) image.srec: image image.lst
7) image.ram: image.srec

### *B. Diagrama del procesador*

El procesador posee un datapath para datos de 32 bits de longitud. Su set de instrucciones posee todo el mismo tamaño de 32 bits por instrucción. Es un procesador de tipo RISC (Reduced Instruction Set Computer), basado en la arquitectura MIPS (Microprocessor without interlocked Pipeline Stages).
Este procesador maneja por separado las instrucciones y los datos: recibe por buses independientes las instrucciones de programa y los datos iniciales de programa. Las instrucciones de programa se reciben por un módulo encargado de almacenarlas en una memoria caché habilitada únicamente para instrucciones. Los datos iniciales de programa se reciben por medio de otro módulo que se encarga únicamente de gestionar los datos del programa en ejecución. En este caso, no existe habilitada una memoria caché por defecto para los datos, sino que tienen que ser leídos y escritos en una memoria externa.

El datapath está segmentado en diferentes etapas, y articulado por una compleja red de registros, señales de control y pequeñas unidades lógico-secuenciales, llamada Pipeline, cuyas secciones son las etapas que componen el datapath. La sigla MIPS hace referencia a que el flujo de datos a través del datapath no se entrecruza o entrelaza, es decir, en cada etapa del pipeline se ejecuta una parte de cada instrucción, y ninguna etapa de ninguna instrucción se debe entrecruzar con ninguna otra etapa. Esto es, la ejecución de las instrucciones es ordenada y secuencial, y siempre se debe ejecutar todo lo concerniente a una instrucción para poder continuar con las demás ejecuciones; el caso contrario es que si una instrucción sufre un error de ejecución, una detención o un error de lectura o escritura, la ejecución de todas las demás etapas que vienen detrás de tal instrucción se detiene hasta que se resuelva lo concerniente a la instrucción problemática.
Son seis las etapas del pipeline que integran todo el datapath: Address, Fetch, Decode, Execute, Memory, Writeback. Cada etapa está delimitada por un banco de registros que almacena todos los resultados de la etapa anterior, más todo tipo de información de control que se requiera para las ejecuciones futuras. Los datos y las señales de control van avanzando en bloque, de un banco de registros a otro, siendo los datos modificados, operados, o lo que dicte cada instrucción. Al final del datapath los datos que surgen como resultado de las instrucciones son realimentados a un banco de registros, donde son almacenados para ser reutilizados en instrucciones futuras, o pasados a la memoria de datos externa.
La unidad de control de todo el Datapath la constituye todo el entramado de señales de control de registros, de unidades funcionales, y de banderas (flags) que indican el estado de la ejecución en cada etapa. El control es de naturaleza lógico-secuencial, y posee la capacidad de manipular el flujo de información en todas las etapas al mismo tiempo, para evitar el anteriormente mencionado solapamiento o entrecruzado de ejecuciones entre una etapa de una instrucción y la siguiente (Fig 1).

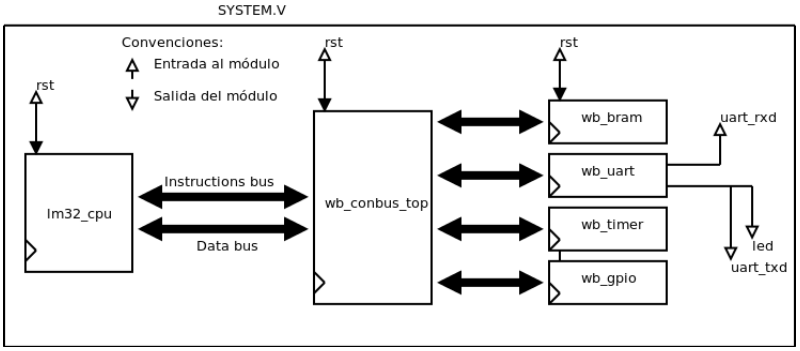


Fig. 1: Diagrama Simplificado para el *System.v*.

## III. Conclusiones

- Las herramientas de instalación proporcionadas al principio del curso fueron de gran ayuda, lámentablemente las mismas desaparecieron durante el desarrollo de esta práctica. De tal modo que dificultó enormemente su desarrollo y por ende la obtención de los resultados esperados.
- El LM32 ofrece una gran versatilidad para el proceso de implemetación, ya que una de sus grandes ventajas es la programación en *C*. El problema radica en el proceso de compilación y lograr obtener el aschivo **.S** y **.bit** para quemar la FPGA.
- .
- .

## Referencias

[1] Patterson, David & Hennessy John *"'Computer Organization And Design - The Hardware-Software Interface"'*. Kindle Edition, Fourth Edition, 2006.
[2] http://www.latticesemi.com/products/intellectualproperty/ipcores/mico32/index.cfm
[3] http://www.ohwr.org/documents/68
[4] http://www.linuxencaja.net/wiki/Arquitectura_LM32_JPRR_%28261744%29

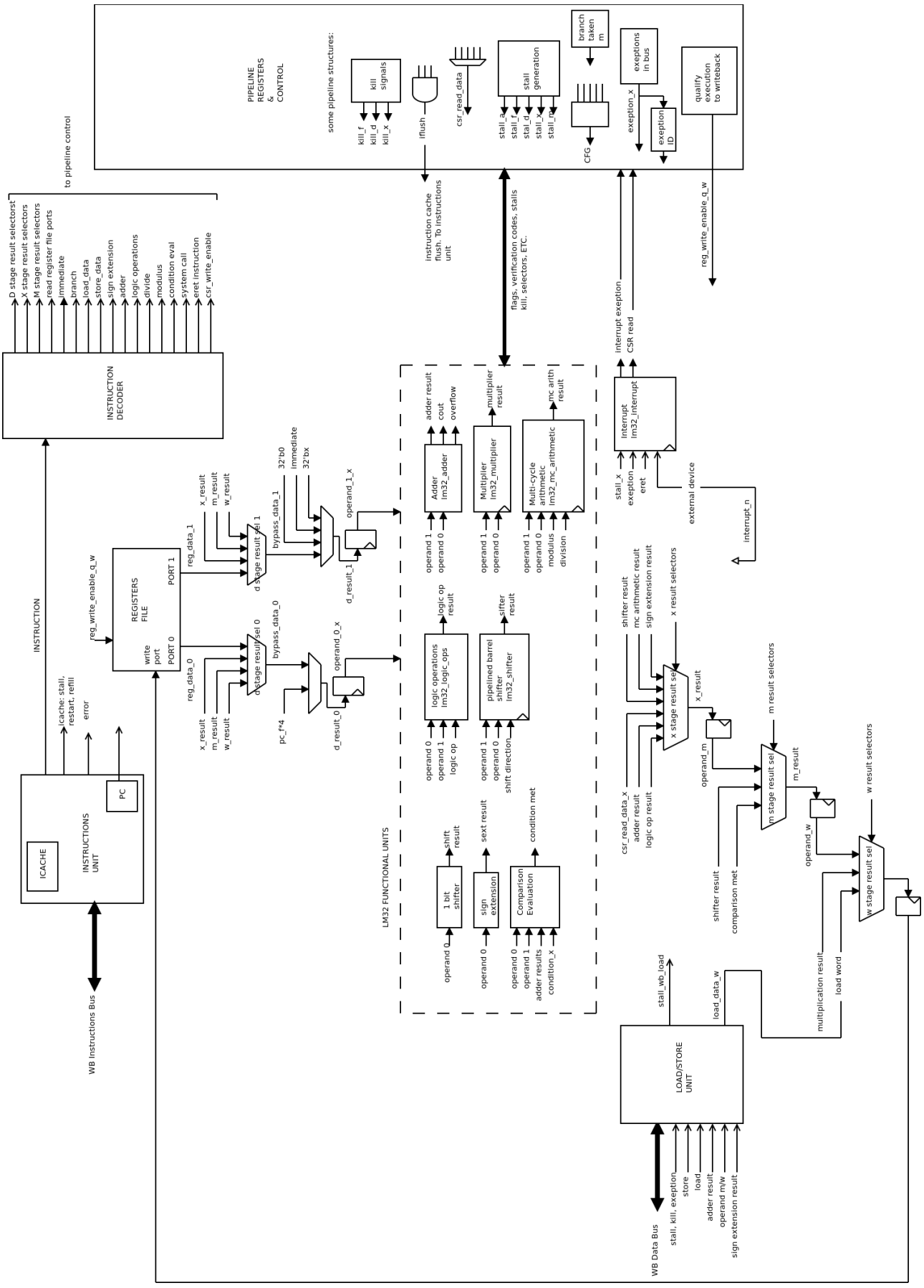


Fig. 2: Diagrama completo para el *datapath*.

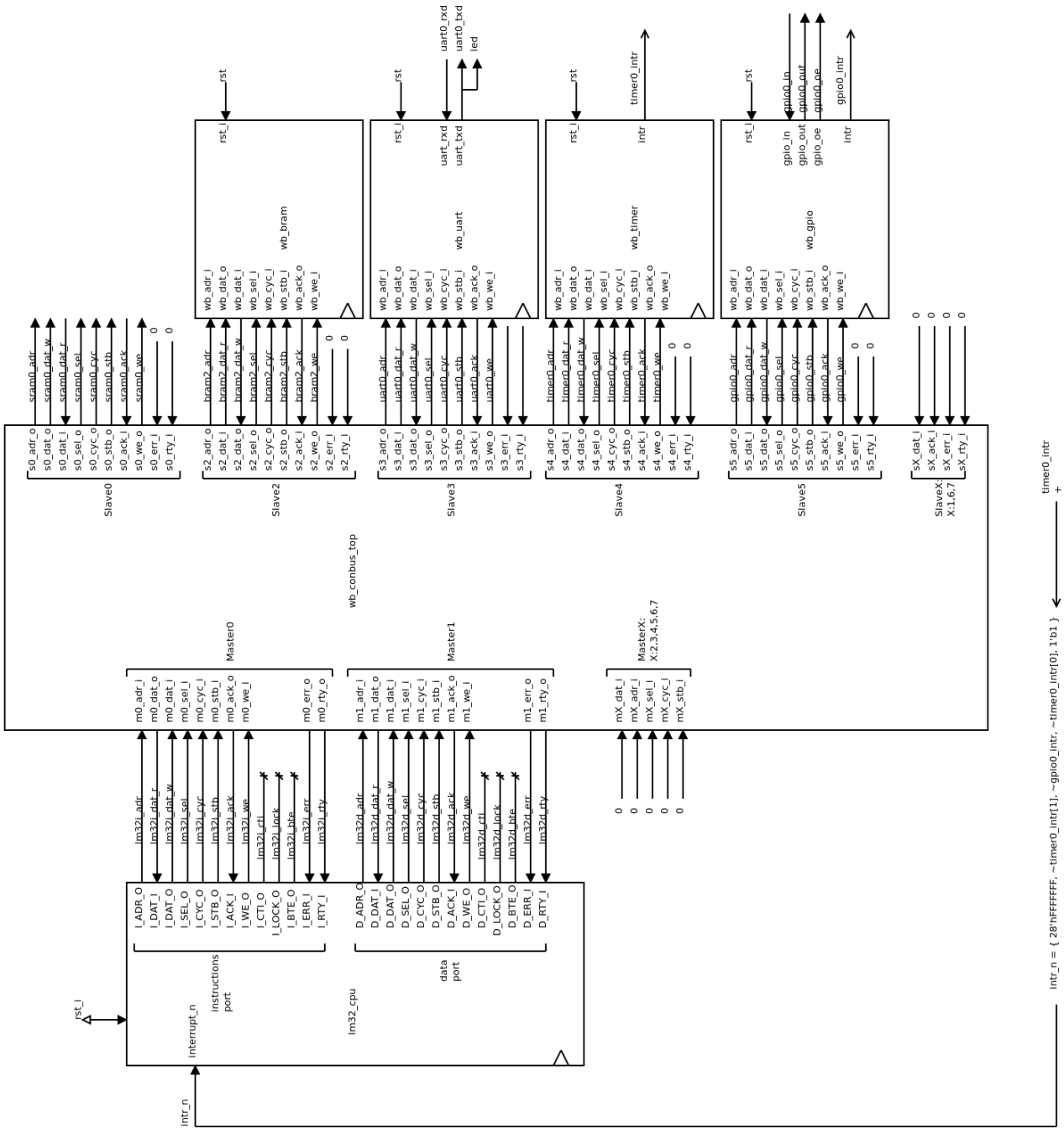


Fig. 3: Diagrama completo para el *System.v*.