Introducción al uso del procesador LatticeMico32

David Ricardo Martínez Hernández Código: 261931
Juan Sebastian Roncancio Arevalo Código: 261585

Palabras clave— **FPGA, LM32, Make, Verilog.**

***Resumen*— Se realizo la instalación de todas las herramientas necesarias en el sistema operativo Linux, distribución Ubuntu, basandonos 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 todos los resultados de manera eficiente y completa.**

I. Objetivos

- Instalar las herramientas de compilación para el procesador LM32.
- Identificar las características básicas del procesador LM32.
- Realizar una aplicación con el procesador LM32.

II. Introducción

El procesador **LatticeMico32** es un procesador *softcore* lo que quiere decir que su implementación deja de ser física para poder ser implementada por completo con los recursos lógicos de las FPGAs, este procesador esta basado en una arquitectura RISC que es el acrónimo de "**reduced instruction set computer**" estos procesadores tienen las siguientes características:

- Instrucciones de tamaño fijo y presentadas en un reducido numero de formatos.
- Solo las instrucciones de carga y almacenamiento acceden a la memoria de datos.

El objetivo de crear estos procesadores es promover el paralelismo y la segmentación de instrucciones y reducir el acceso a la memoria. Este procesador provee la visibilidad y portabilidad esperada para ser un dispositivo de diseño de hardware libre[1].

III. Materiales

- Cable serial
- Computador con Sistema Operativo Linux Ubuntu
- FPGA

IV. Procedimiento

A. Proceso para sintetizar y compilar

Para sintetizar o simular se deben seguir los siguientes pasos:

1) Ingresar a la consola y ubicarse en el directorio $SIE/SoftCore/lm32/logic/sakc$.
2) Tener un archivo con extensión $.ram$ en el directorio firmware y se realiza el make correspondiente.
3) Escribir en consola make syn (para sintetizar).
4) Escribir en consola make sim (para simular).
5) Escribir en consola make view (para ver los resultados de la simulación en gtkwave).

[1]Tomado de [4], 07 de Marzo de 2012

Este es el código que aparece en el archivo log de la dirección

$SIE/SoftCore/lm32/logic/sakc/firmware/boot0-serial$

```
lm32-elf-gcc -MMD -O2 -Wall -g -s
-fomit-frame-pointer -mbarrel-shift-enabled
-mmultiply-enabled -mdivide-enabled
-msign-extend-enabled -c crt0ram.S
echo "digital2 crt0ram.o"
digital2 crt0ram.o

lm32-elf-gcc -MMD -O2 -Wall -g -s
-fomit-frame-pointer -mbarrel-shift-enabled
-mmultiply-enabled -mdivide-enabled
-msign-extend-enabled -c main.c
echo "digital2 main.o"
digital2 main.o

lm32-elf-gcc -MMD -O2 -Wall -g -s
-fomit-frame-pointer -mbarrel-shift-enabled
-mmultiply-enabled -mdivide-enabled
-msign-extend-enabled -c soc-hw.c
echo "digital2 soc-hw.o"
digital2 soc-hw.o

lm32-elf-ld -nostdlib -nodefaultlibs
-Tlinker.ld  -Map image.map -N -o
image crt0ram.o main.o soc-hw.o
echo "digital2 image"
digital2 image

lm32-elf-objdump -h -S image > image.lst
echo "digital2 image.lst"
digital2 image.lst
lm32-elf-objcopy -j .text -j .rodata -j
.data -O srec image image.srec
echo "digital2 image.srec"
digital2 image.srec
../../tools/srec2vram/srec2vram
   image.srec 0x00000000 0x1000 > image.ram
echo "digital2 image.ram"
digital2 image.ram
make: <<image.ram>> está actualizado.
```

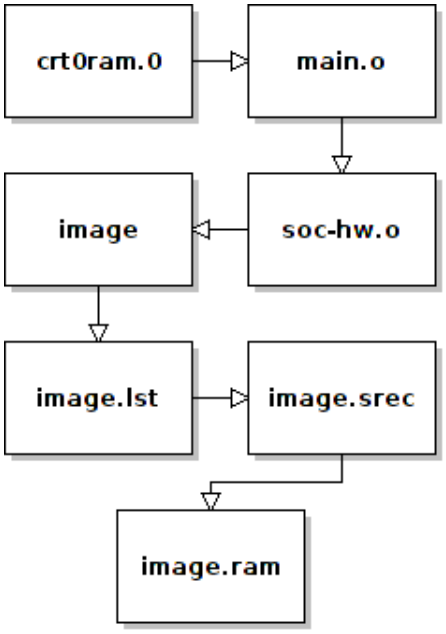

Fig. 1: Diagrama de los pasos que realiza el archivo log

B. RTL y camino de datos

Se realizó el segundo procedimiento descrito en la guía de laboratorio para identificar la arquitectura del procesador LM32:

- 1) Ingresar a la consola en el directorio $SIE/SoftCore/lm32/logic/sakc/$
  2) Ingresar en la consola el comando **grep** de la siguiente forma:

grep -IR RTL *

Salida desde consola de la carpeta *sack* en consola el comando anterior:

```
123.txt:RTL Output
     : yes
123.txt:RTL Top Level Output File Name
     : system.ngr
build/system.ngc_xst.xrpt:
<item stringID=
"XST_RTL_TOP_LEVEL_OUTPUT_FILE_NAME"
value="system.ngr"/>
build/system.srp:RTL Output
     : yes
build/system.srp:RTL Top Level Output
 File Name
     : system.ngr
digital1.log:RTL Output
     : yes
digital1.log:RTL Top Level Output File Name
     : system.ngr
digital1.log:RTL Output
     : yes
digital1.log:RTL Top Level Output File Name
     : system.ngr
digital1.txt:RTL Output
     : yes
digital1.txt:RTL Output
     : yes
digital1.txt:RTL Top Level Output File Name
     : system.ngr
digital1.txt:RTL Output
     : yes
digital1.txt:RTL Top Level Output File Name
     : system.ngr
digital2.txt:RTL Output
     : yes
digital2.txt:RTL Top Level Output File Name
     : system.ngr
digi.txt:RTL Output
     : yes
laboratorio1/wb_sram32.syr:RTL Output
     : Yes
laboratorio1/wb_sram32.syr:RTL Top Level
 Output File Name
     : wb_sram32.ngr
laboratorio1/wb_sram32_xst.xrpt:
<item stringID=
"XST_RTL_TOP_LEVEL_OUTPUT_FILE_NAME"
 value="wb_sram32.ngr"/>
laboratorio1/laboratorio1.restore:
"A" "" "" "" "PROP_xstGenerateRTLNetlist" "Yes"
log:RTL Output
     : yes
log:RTL Top Level Output File Name
     : system.ngr
sim/ddr/readme.txt:To run simulations for this
sample configuration, user has to generate
the RTL from the tool for the following GUI
```

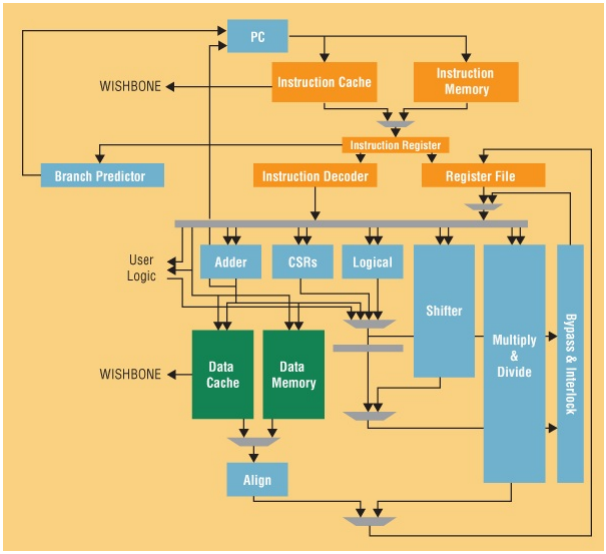

Fig. 2: Diagrama del RTL.(Tomado de [4])

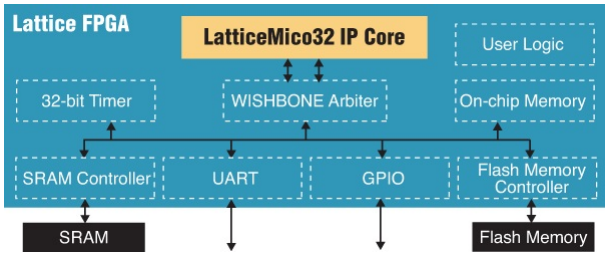

Fig. 3: Diagrama del camino de datos (Datapath).(Tomado de [4])

Las instrucciones del procesador $LM32$ son de 32 bits y se encuentran en cuatro formatos básicos que se listan a continuación:

1-bit	5-bit	5-bit	5-bit	16-bits
0	Opcode	Reg 0	Reg 1	Immediate

TABLA I: Formato Inmediato

1-bit	5-bit	5-bit	5-bit	5-bit	11-bits
1	Opcode	Reg 0	Reg 1	Reg 2	11'b0

TABLA II: Formato Registro Registro

1-bit	5-bit	5-bit	5-bit	16-bits
1	Opcode	CSR 0	Reg	161'b0

TABLA III: Registro de Control

6-bit Register	26-bit Immediate
11100(/11)0	Immediate

TABLA IV: Formato Inmediato

Algunas de las características clave de este procesador de 32-bits incluyen

- RISC architecture
- 32-bit data path
- 32-bit instructions
- 32 general-purpose registers
- Up to 32 external interrupts
- Optional instruction cache
- Optional data cache
- Dual WISHBONE memory interfaces (instruction and data)

Para acelerar el desarrollo de procesos del procesador, se pueden adicionar componentes periféricos. Específicamente, estos componentes están conectados al procesador a través de un WISHBONE interfaz de bus. Por utilizar este interfaz de origen de bus abierto, puede incorporar los componentes de Wishbone propias en los diseños embebidos. Los componentes incluyen:

- Memory controllers
- Asynchronous SRAM
- Double data rate (DDR)
- On-chip
- Input/output (I/O) ports
- 32-bit timer
- Direct memory access (DMA) controller
- General-purpose I/O (GPIO)
- I2C master controller
- Serial peripheral interface (SPI)
- Universal asynchronous receiver transmitter (UART)

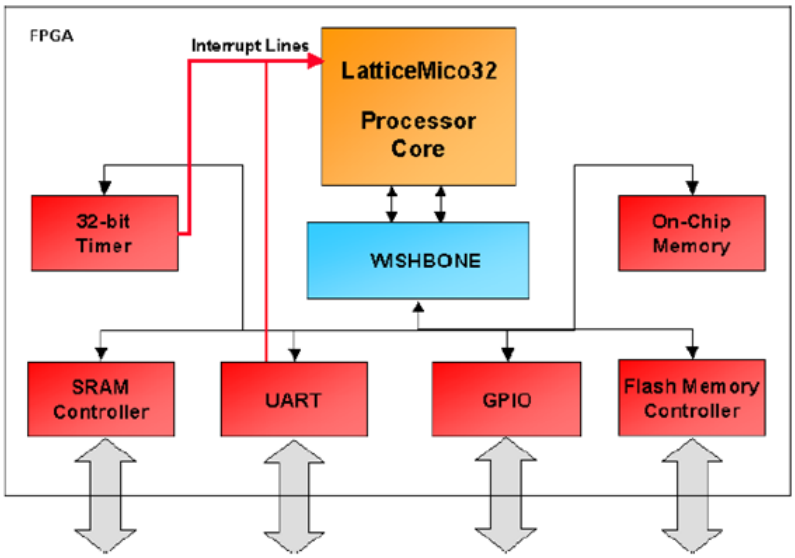

Fig. 4: Sistema completo utilizando el procesador junto con varios componentes

V. Aplicación

Este es el código utilizado para la aplicación de sumar 2 números, es decir el *main.c* y también se muestra la simulación del programa tomando como ejemplo $1+1$ fig. 5

```
#include "soc-hw.h"
/*prototypes */
uint32_t read_uint32()
{
uint32_t val = 0, i;
    for (i = 0; i < 4; i++) {
        val <<= 8;
        val += (uint8_t)uart_getchar();
    }
    return val;
}

int main(int argc, char **argv)
{
// Initialize UART
uart_init();

char a,b,c;
a= uart_getchar();
b= uart_getchar();
c= a+b;
uart_putchar(c);
}
```

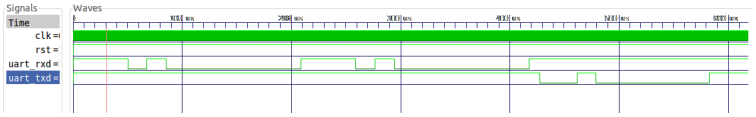

Fig. 5: Simulación

VI. Conclusiones

- Debido a la disponibilidad de los componentes de software libre, estos procesadores son y serán los pilares del desarrollo futuro, ya que por su bajo costo y su fácil implementación son las herramientas perfectas para el desarrollo de muchos dispositivos.
- Debido al escaso conocimiento sobre el sistema operativo basado en Linux nos tomo más tiempo del necesario instalar los componentes en dicho sistema operativo ya que es un poco diferente a lo que estamos acostumbrados a instalar en Windows, por lo tanto tuvimos que aprender a manejar comandos para usar la consola del sistema operativo basado en Linux. Y por esta razón el desarrollo de la practica estuvo un poco sesgada a la disponibilidad de los programas instalados en el sistema operativo.

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