Software libre para depuración del LEON-3

Diego J. Brengi, Salvador E. Tropea
Electrónica e Informática
Instituto Nacional de Tecnología Industrial
Buenos Aires, Argentina
Email: brengi@inti.gob.ar, salvador@inti.gob.ar

***Resumen*—El LEON-3 es una descripción en lenguaje VHDL de un procesador de 32 bits tipo SPARC. Este procesador, junto a otros *IP cores* de soporte, pertenece a la biblioteca GRLIB, que permite implementar un SoC en dispositivos FPGA y ASICs. Esta biblioteca VHDL es ofrecida por la empresa Aeroflex-Gaisler bajo licencia GNU GPL. Para realizar la depuración del diseño, mediante comunicación serial asincrónica, la biblioteca incluye el módulo AHBUART "AMBA AHB Serial Debug Interface". El mismo brinda acceso al bus principal del sistema. Para comunicarse con esta unidad, la empresa nos ofrece un software bajo una licencia paga y propietaria. En este trabajo se presenta el desarrollo de una alternativa de software libre, bajo licencia GNU GPL, para comunicarse con el módulo AHBUART de la GRLIB y realizar operaciones básicas.**

I. Introducción

El desarrollo presentado en este trabajo busca como objetivo completar las herramientas de software libre disponibles para abordar un desarrollo utilizando la biblioteca GRLIB y el procesador LEON, ya que no se conoce una alternativa libre para depuración remota del sistema.

Antes de presentar el software desarrollado y sus características, se dará una breve introducción al procesador LEON y la biblioteca GRLIB. Luego se expondrá el desarrollo del software, los resultados obtenidos y los planes a futuro.

I-A. El procesador LEON-3

El procesador LEON-3 respeta la arquitectura SPARC[1] versión 8, que define a un procesador de 32 bits, de tipo RISC y big-endian [1].

La especificación SPARC V8 se publicó en el año 1990, logrando además obtener el estándar IEEE 1754-1994. Es una arquitectura abierta, las especificaciones de diseño están publicadas y son de acceso gratuito. El LEON-3 obtuvo su certificación por parte de SPARC International en mayo de 2005.

El desarrollo de las primeras versiones del LEON fue realizado por el ESTEC "European Space Research and Technology Centre" como una alternativa europea para aplicación en misiones espaciales [2]. En la actualidad su desarrollo lo realiza la empresa Aeroflex-Gaisler [3]. Algunas características del LEON-3 son:

- Implementado en lenguaje VHDL.
- Set de instrucciones compatibles con la especificación SPARC V8.
- Pipeline de 7 etapas.
- Unidades de hardware para multiplicación, división y MAC[2].
- Disponibilidad de FPU[3] IEEE-754 de alta performance (no disponible en la versión gratuita ni bajo licencia GPL).
- Caché para datos e instrucciones.
- Unidad SPARC MMU[4] (SRMMU[5]) con TLB[6] configurable.
- Bus conforme a AMBA-2.0 AHB[7] .
- Soporte avanzado para depuración dentro del chip y *trace buffer* para datos e instrucciones.
- Soporte para multiprocesadores simétricos (SMP).
- Diseño completamente sincrónico con flanco de reloj simple.
- Disponibilidad de versiones a prueba de fallas tipo SEU[8], para aplicaciones espaciales (no disponible en forma gratuita ni en versión GPL).
- Soportado por el compilador GCC y las binutils.

I-B. La biblioteca GRLIB

El procesador LEON-3 se distribuye como parte central de la biblioteca GRLIB, un conjunto de *IP cores* organizados y configurables [4].

Esta librería es altamente configurable, pudiendo seleccionar qué componentes incluir y las características de cada uno.

Los *IP cores* se comunican entre sí a través de los buses AMBA AHB y APB[9] (Ver Fig. 1). El Bus AHB soporta múltiples maestros y transferencias en ráfaga, en cambio el bus APB está diseñado para hardware más simple y de menores requerimientos.

La mayor parte de la GRLIB se distribuye bajo licencia GPL, pero algunos de sus componentes deben comprarse y utilizan una licencia restrictiva.

I-C. IP cores *disponibles*

Entre los *IP cores* GPL más relevantes que posee la GRLIB podemos mencionar:

[1] "Scalable Processor ARChitecture"

[2] Multiply-ACcumulate
[3] Floating Point Unit
[4] Memory Management Unit
[5] SPARC Reference Memory Management Unit
[6] Translation Lookaside Buffer
[7] Advanced High Performance Bus
[8] Single Event Upset
[9] Advanced Peripheral Bus

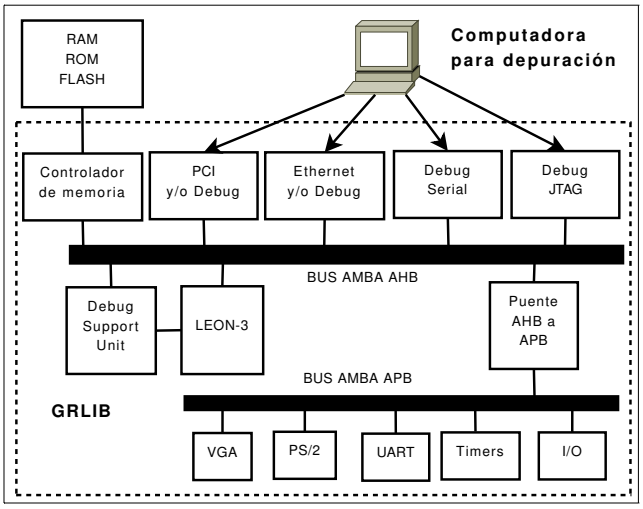

Figura 1. Esquema general de la GRLIB y sus módulos de depuración.

- Procesador LEON-3 y su unidad de depuración.
- Unidades de depuración por JTAG, Ethernet y puerto serie.
- Ethernet MAC 10/100.
- PCI Master y Target, con DMA.
- Teclado PS/2, UART, discos ATA, CAN (OpenCores), temporizadores, FIFOs y E/S de propósitos generales.
- Trace buffers y Analizador lógico.
- Controladores de memorias RAM, ROM, SDRAM PC-133, DDR2 y DDR266.
- Controlador VGA y SVGA.
- Módulo I^2C maestro y esclavo. Controlador SPI.

Están soportados los sintetizadores y las FPGAs más utilizadas en el mercado, además de disponer de diseños de referencia para varias tarjetas de desarrollo.

I-D. Linux embebido

Existe una versión de Snapgear Linux mantenida por Aeroflex-Gaisler para el LEON-3. Esta versión puede utilizar un kernel Linux de la serie 2.6 si se incorpora la unidad MMU y un kernel μCLinux 2.0 para sistemas sin MMU [**?**].

Esta distribución se ofrece como código fuente en conjunto con las herramientas de cross compilación y con una interfaz de configuración de opciones similar a la del kernel Linux. También se brindan imágenes en formato ELF para algunas de las tarjetas FPGA capaces de ejecutar estos sistemas.

I-E. Depuración

La GRLIB dispone de varias opciones para realizar la depuración y prueba del sistema a través de módulos periféricos conectados al bus AHB. Esto permiten tomar el control del bus y los componentes conectados al mismo. Los módulos de depuración son:

- AHBJTAG, *JTAG Debug Link* con interfaz AHB.
- AHBUART, *AMBA AHB Serial Debug Interface*.
- GRETH, *Ethernet Media Access Controller*.
- PCITARGET, *Simple 32-bit PCI Target* con interfaz AHB.
- USBDCL, *USB Debug Communication Link*.

Salvo el USB, estos componentes se brindan gratuitamente dentro de la GRLIB bajo licencia GPL.

Para comunicarse con estos módulos desde una computadora, es necesario utilizar un software llamado Grmon brindado por la empresa Aeroflex-Gaisler bajo una licencia paga y propietaria.

Utilizando alguno de los módulos de hardware, este software permite realizar operaciones sobre nuestro diseño. Por ejemplo:

- Listado de *IP cores* registrados en el sistema. Esto se logra mediante un mecanismo de *Plug&Play* que utiliza cada *core* en el bus AMBA.
- Consulta y modificación de los registros de cada *IP core* presente en el bus.
- Acceso a las memorias del sistema. Esto permite, por ejemplo, la grabación de memorias flash conectadas a la FPGA para cargar el *firmware* del procesador.
- Control y depuración del procesador a través de la DSU[10].

II. Desarrollo

Luego de investigar alternativas, se eligió la biblioteca GRLIB como base para futuros trabajos por poseer una licencia de libre uso y modificación, estar realizada en VHDL, por la capacidad de correr sistemas GNU/Linux o μCLinux y sus casos publicados de uso y aplicación [5] [6].

Pasando las etapas de configuración, síntesis y transferencia a la FPGA, se hizo evidente la necesidad de una herramienta de depuración para verificar el funcionamiento del hardware, realizar pruebas y para cargar la imagen del sistema operativo Linux en la memoria flash. La opción natural para esto es el software Grmon [**?**]. Por tratarse de una herramienta de importancia para el uso de la GRLIB y debido a que no se conocen alternativas de software libre, se decidió desarrollar un reemplazo, ya que dentro del laboratorio se considera un objetivo estratégico la utilización de herramientas de software libre y estándares abiertos.

Como primera etapa se abordó la depuración por línea serial tipo RS-232, usando el módulo AHBUART, con las funciones de detección de *IP cores* en ambos buses, lectura y escritura de registros y grabación de datos en la memoria flash.

II-A. Hardware de pruebas utilizado

Existen varias tarjetas FPGA capaces de alojar sistemas realizados con la GRLIB sin necesidad de adaptaciones. Para este trabajo se utilizó la tarjeta GR-XC3S de Pender Electronic Design que posee un dispositivo Spartan 3 1500 (Ver Fig. 2). Para esta tarjeta existen imágenes precompiladas de Snapgear Linux para transferir a la memoria Flash.

[10]Debug Support Unit

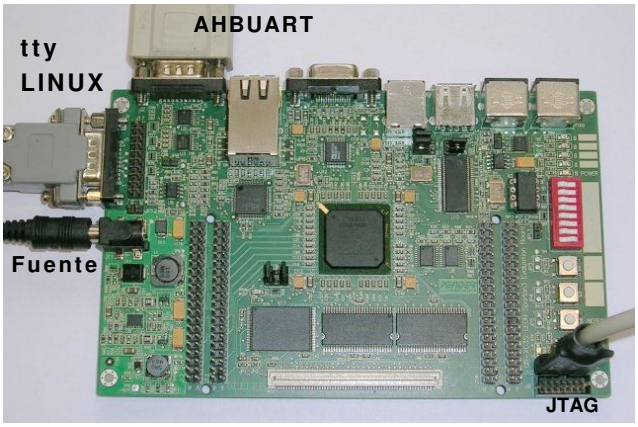

Figura 2. Tarjeta GR-XC3S utilizada en el desarrollo.

II-B. Plataforma de desarrollo

Para el desarrollo del software en PC se utilizó sistema operativo Debian GNU/Linux "Etch" y lenguajes C y C++ con el compilador de GNU. Para la edición se utilizó el editor Setedit y GDB para depuración. Para automatizar el proceso de compilación se utilizó GNU Make.

II-C. Bibliotecas

El programa utiliza la biblioteca "readline" para realizar el manejo del ingreso de comandos y la biblioteca "libelf" para realizar la interpretación de secciones de archivos en formato ELF[11]. Estos archivos se utilizan para las imágenes de un sistema operativo GNU/Linux listo para cargar en la memoria flash. Además se utiliza la biblioteca "cppfio" (de uso interno) para tareas de uso corriente como parseo de los argumentos de llamada, manejo de errores, etc.

II-D. Protocolo de comunicación

El protocolo de comunicación entre la PC y el módulo AHBUART es sencillo y se encuentra detallado en la documentación de la GRLIB [7]. Está basado en bytes, dadas las limitaciones del puerto serial de las computadoras. La PC actúa de maestro y puede originar comandos de lectura y de escritura, donde el primer byte enviado indica el tipo de comando y la cantidad de datos de 32 bits a enviar o recibir. Los siguientes 4 bytes se utilizan para la dirección de comienzo, que luego se autoincrementa si es necesario. Finalmente, para el caso de las escrituras, se envían grupos de 4 bytes con las palabras a escribir. En la Fig. 3 se ilustra este protocolo.

El módulo AHBUART posee la capacidad de detectar el baudrate empleado. Para hacer esto, los primeros dos bytes que debe recibir luego de un reset deben tener un valor de 0x55.

Para analizar la comunicación serie se realizó un programa que recibe un flujo serie de datos capturados y muestra los comandos, direcciones y datos en forma fácil de interpretar. Se expone un ejemplo de uso:

[11]Executable and Linking Format

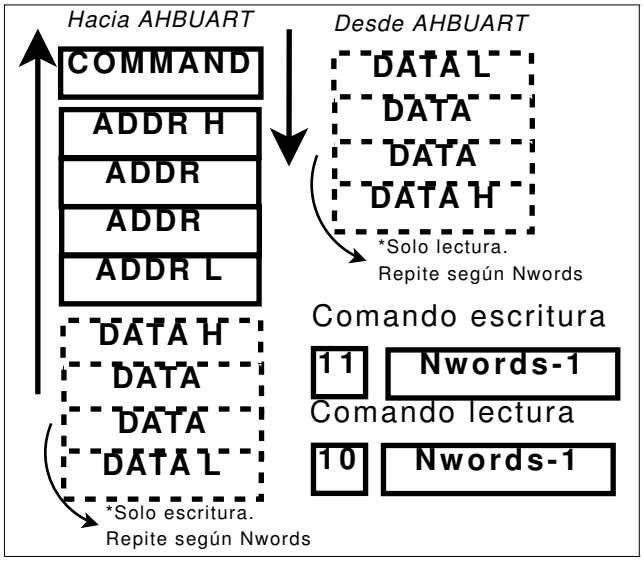

Figura 3. Protocolo de comunicación entre PC y AHBUART.

```
bash$ interceptty /dev/ttyS1 -p /dev/ptyp0 | ./flemonspy \
      >commlog.txt&
bash$ ./flemon -D /dev/ttyp0
 [ ... Uso del programa y finalización ]
bash$ cat commlog.txt
Flemonspy. Basic analizer
SYNC: 55
SYNC: 55
001 Rd01(80) Addr[80000014]
    Resp: 82206020
002 Rd08(87) Addr[FFFFF000]
    Resp: 01003000 00 00 00 00 00 00 00
003 Rd08(87) Addr[FFFFF020]
    Resp: 01007000 00 00 00 00 00 00 00
[...]
031 Wr01(C0) Addr[00000000] FFFF50FF
286 Rd01(80) Addr[80000000]
    Resp: 00000888
032 Wr01(C0) Addr[80000000] 00000088
 Final report: Read commands 286 , Write commands 32
 Bytes: transmited 1734, received 2104, Total 3838
```

II-E. Relevamiento de IP cores *presentes en el bus AHB*

La GRLIB define un espacio de direcciones llamado *Plug-&Play* de sólo lectura, donde se coloca información de cada *IP core* presente en el bus AHB, utilizando 32 bytes por *core*. De esta área podemos obtener el identificador del módulo y sus espacios de direcciones asignadas. Bajo esta estructura pueden existir hasta 64 maestros (0xFFFF000 - 0xFFFF7FF) y hasta 64 esclavos (0xFFFF800 - 0xFFFFFFF).

Luego de establecer la comunicación con el módulo AHBUART y lograr el acceso al bus AMBA, se puede realizar la lectura de esta información. La misma permite identificar cada módulo y luego realizar una encuesta de los mismos. El programa desarrollado posee un archivo de texto externo donde se colocan las asociaciones entre códigos de identificación y textos descriptivos para los módulos y sus fabricantes.

```
VENDOR=0x01="Gaisler Research"
 0x002="LEON2 Debug Support Unit"
 0x003="LEON3 SPARC V8 Processor"
 0x004="LEON3 Debug Support Unit"
 0x005="10/100 Mbit Ethernet MAC"
 0x006="AHB/APB Bridge with Plug&Play"
```

```
 0x007="AHB Debug UART"
 0x008="8/32bit Sram/Prom Mem Controller"
 [...]
VENDOR=0x04="European Space Agency"
 0x002="LEON2 SPARC V8 Processor"
 0x003="LEON2 Peripheral Bus"
 0x005="LEON2 Interrupt Ctrl"
 0x006="LEON2 Timer Unit"
 0x007="LEON2 UART Unit"
 0x008="LEON2 Configuration register"
 0x009="LEON2 I/O port"
 0x00f="8/16/32bit PROM/IO/SRAM/SDRAM LEON2 Mem Ctrl"
 [...]
```

De particular importancia es la detección del controlador de memoria flash que nos permitirá luego acceder a la misma y la detección del puente AHB/APB para realizar la detección de *IP cores* en el bus de periféricos.

II-F. Relevamiento de IP cores *presentes en el bus APB*

Una vez que se obtiene la dirección del puente AHB/APB, se puede acceder a la sección *Plug&Play* del bus APB, con dirección 0xFF000 relativa de inicio. Esta sección es similar a la del bus AHB pero aquí solo hay *IP cores* esclavos y cada uno se informa en 8 bytes. Esto permite hasta 512 esclavos por bus, pero debido a la penalización de tiempo de comunicación por el puerto serie, sólo se escanean las primeras 16 entradas, ya que ese es el límite por defecto colocado en la GRLIB y el kernel Linux. Esto mismo se hace también para maestros y esclavos en el bus AHB mencionado en la sección anterior.

II-G. Memoria Flash

En las placas utilizadas el acceso a la memoria flash se debe realizar a través del módulo llamado MCTRL, identificado con *Vendor ID*:0x04 “European Space Agency” y *Device ID*: 0x00f, con descripción “8/16/32bit PROM/IO/SRAM/SDRAM LEON2 Mem Ctrl”. Este dispositivo posee registros de configuración en el bus APB y brinda acceso al bus dedicado de memoria externa a través del bus AHB.

Una vez que se obtiene acceso al bus de datos y direcciones de la memoria, es necesario utilizar el protocolo CFI[12], un estándar abierto JEDEC, para obtener información de uso sobre la memoria [8]. Mediante este protocolo se puede averiguar el tamaño y geometría de la memoria, los bancos, timeouts de escritura y borrado, etc. Además se puede realizar el borrado de los bancos y la escritura de datos (Ver Fig. 4). En el modo *Read Array* se realiza la lectura normal de los datos de la memoria flash.

III. Uso del programa desarrollado

III-A. Opciones de inicio

La versión actual de programa permite el pasaje de parámetros por línea de comandos como por ejemplo −D DEVICE o −−device=DEVICE para seleccionar el puerto serie a utilizar. También es posible crear archivos de configuración para evitar repetir las opciones más comunes.

[12]Common Flash Interface

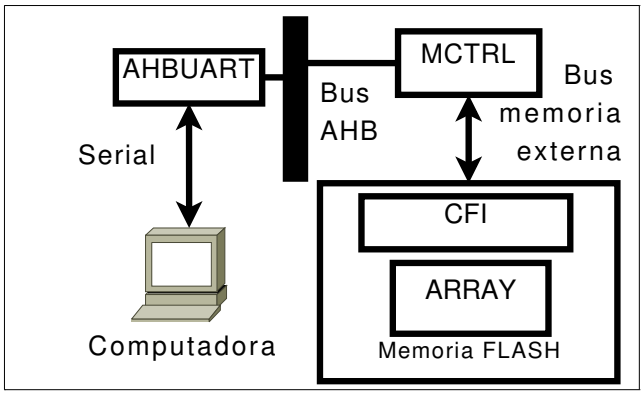

Figura 4. Acceso a la memoria flash.

III-B. Interfaz de usuario

Luego de ejecutar el programa se ingresa a un intérprete de comandos con autocompletado e historial similar al BASH, y funciones específicas para comunicarse con la GRLIB. Con el comando *help* podemos obtener un listado de las funciones disponibles:

```
FLeMon$ help
FLeMon - FPGALibre Leon Monitor
[...]
Commands:
  help          Display this text.
  quit          Quit using FleMon.
  sync          Send SYNC to establish connection.
  scanh         Scan only AHB bus.
  scanp         Scan only APB bus.
  scan          Scan AHB and APB buses.
  ls            List AHB/APB detected cores.
                -l extended. -c collapse.
  fscan         Flash detection.
  flash         Flash report.
  flock         Flash lock [all | offset].
  funlock       Flash unlock (all banks)
  fstat         Flash lock status.
  fcfi          Flash CFI dump.
  ferase        Flash bank erase. [offset] or 'all'.
  fdumpblock    Flash dump block. [offstart] [offend].
  fwrite        Flash write a single word. [offset] [data]
  fcheck        Flash blank check. [offstart] [offend]
  fload         Flash load ELF file. [filename]
  mem           Report memory map.
```

III-C. Detección de IP cores

Para detectar los *IP cores* es necesario primero enviar los bytes de sincronismo para el módulo AHBUART. Esto se realiza con el comando *sync*. Luego con el comando *scan* se toman los datos del área *Plug&Play* del bus AHB. En caso de encontrarse un puente AHB a APB, se realiza posteriormente la encuesta de los módulos presentes en el mismo. Habiendo realizado esto, podemos listar los *IP cores* detectados con el comando *ls*:

```
FLeMon$ sync
Ok!
FLeMon$ scan
===== AHB SCANNING RESULTS ==============================
AHB   N  slotADDR   IDValue    BAR0    BAR1   BAR2  BAR3
Mastr 00 0xFFFFF000 0x01003000
Mastr 01 0xFFFFF020 0x01007000
Mastr 02 0xFFFFF040 0x0101C000
Mastr 03 0xFFFFF060 0x01063000
Mastr 04 0xFFFFF080 0x0101D000
Mastr 05 0xFFFFF0A0 0x01022000
```

```
Slave 64 0xFFFFF800 0x0400F020 0x3E002 0x2E002 0x4003C00
Slave 65 0xFFFFF820 0x01006000 0x8000FFF2
Slave 66 0xFFFFF840 0x01004020 0x9000F002
Slave 67 0xFFFFF860 0x01024006 0xA000FFF3
===== APB SCANNING RESULTS ============================
BUS0  slotADDR   IDValue    BAR
01    0x800FF000 0x0400F020 0x0000FFF1
02    0x800FF008 0x0100C022 0x0010FFF1
[...]

FLeMon$ ls -c
=============== DETECTED SYSTEM CORES (collapsed) ======
01-LEON3 SPARC V8 Processor             Gaisler Research
02-AHB Debug UART                       Gaisler Research
03-JTAG/AHB Debug Interface             Gaisler Research
04-SVGA video frame buffer              Gaisler Research
05-GR Ethernet MAC                      Gaisler Research
06-USB 2.0 Debug Comunication Link      Gaisler Research
07-8/16/32bit PROM/IO/SRAM/SDRAM LEON2 Mem Ctrl
                                  European Space Agency
08-AHB/APB Bridge with Plug&Play        Gaisler Research
09-LEON3 Debug Support Unit             Gaisler Research
10-IDE/ATA Controller                   Gaisler Research
11-Generic APB UART                     Gaisler Research
12-LEON3 Multi proccess Interrupt Ctrl  Gaisler Research
13-Modular Timer Unit                   Gaisler Research
14-APB Keyboard PS/2 interface          Gaisler Research
15-APB Keyboard PS/2 interface          Gaisler Research
16-General purpose I/O port             Gaisler Research
17-AHB Status reg (failing address)     Gaisler Research
```

Con el comando *ls -l* podremos ver un reporte más detallado de los módulos presentes y con el comando *mem* se listan en forma organizada las asignaciones en los buses:

```
FLeMon$ ls -cl
========= DETECTED SYSTEM CORES (long) (collapsed) ========
01-LEON3 SPARC V8 Processor (Id:0x03 vendor:0x01)
  Vendor:Gaisler Research ahb_mastr Slot:0
02-AHB Debug UART (Id:0x07 vendor:0x01)
  Vendor:Gaisler Research  ahb_mastr Slot:1 apb Slot:8
     apb i/o: 80000700-80000800
03-JTAG/AHB Debug Interface (Id:0x1C vendor:0x01)
  Vendor:Gaisler Research  ahb_mastr Slot:2
04-SVGA video frame buffer (Id:0x63 vendor:0x01)
  Vendor:Gaisler Research  ahb_mastr Slot:3 apb Slot:7
     apb i/o: 80000600-80000700
05-GR Ethernet MAC (Id:0x1D vendor:0x01)
  Vendor:Gaisler Research  Irq:12 ahb_mastr
   Slot:4 apb Slot:12
     apb i/o: 80000B00-80000C00
06-USB 2.0 Debug Comunication Link (Id:0x22 vendor:0x01)
  Vendor:Gaisler Research  ahb_mastr Slot:5
07-8/16/32bit PROM/IO/SRAM/SDRAM LEON2 Mem Ctrl
  (Id:0x0F vendor:0x04)
  Vendor:European Space Agency  ahb_slv Slot:64 apb Slot:1
     ahb mem: 00000000-20000000 Prefetchable Cacheable
     ahb mem: 20000000-40000000
     ahb mem: 40000000-80000000 Prefetchable Cacheable
     apb i/o: 80000000-80000100
08-AHB/APB Bridge with Plug&Play (Id:0x06 vendor:0x01)
  Vendor:Gaisler Research  ahb_slv Slot:65
     ahb mem: 80000000-80100000
09-LEON3 Debug Support Unit (Id:0x04 vendor:0x01)
  Vendor:Gaisler Research  ahb_slv Slot:66
     ahb mem: 90000000-A0000000
10-IDE/ATA Controller (Id:0x24 vendor:0x01)
  Vendor:Gaisler Research  Irq:6 ahb_slv Slot:67
     ahb i/o: FFFA0000-FFFA0100
11-Generic APB UART (Id:0x0C vendor:0x01)
  Vendor:Gaisler Research  Irq:2 apb Slot:2
     apb i/o: 80000100-80000200
12-LEON3 Multi proccess Interrupt Ctrl (Id:0x0D vendor:0x01)
  Vendor:Gaisler Research  apb Slot:3
     apb i/o: 80000200-80000300
13-Modular Timer Unit (Id:0x11 vendor:0x01)
  Vendor:Gaisler Research  Irq:8 apb Slot:4
     apb i/o: 80000300-80000400
         Scaler:8 bits Timers:2 Width:32 bits.
         Separate interrupts.  Irq:8 Timerfreeze: No
14-APB Keyboard PS/2 interface (Id:0x60 vendor:0x01)
  Vendor:Gaisler Research  Irq:4 apb Slot:5
     apb i/o: 80000400-80000500
15-APB Keyboard PS/2 interface (Id:0x60 vendor:0x01)
  Vendor:Gaisler Research  Irq:5 apb Slot:6
     apb i/o: 80000500-80000600
16-General purpose I/O port (Id:0x1A vendor:0x01)
  Vendor:Gaisler Research  apb Slot:9
     apb i/o: 80000800-80000900
17-AHB Status reg (failing address) (Id:0x52 vendor:0x1)
  Vendor:Gaisler Research  Irq:7 apb Slot:16
     apb i/o: 80000F00-80001000

FLeMon$ mem
ahb mem:  00000000-20000000  8/16/32bit PROM/IO/SRAM/SDR
ahb mem:  20000000-40000000  8/16/32bit PROM/IO/SRAM/SDR
ahb mem:  40000000-80000000  8/16/32bit PROM/IO/SRAM/SDR
ahb mem: [80000000-80100000] AHB/APB Bridge w Plug&Play
apb i/o:  80000000-80000100  8/16/32bit PROM/IO/SRAM/SDR
apb i/o:  80000100-80000200  Generic APB UART
apb i/o:  80000200-80000300  LEON3 Multi proccess Interr
apb i/o:  80000300-80000400  Modular Timer Unit
apb i/o:  80000400-80000500  APB Keyboard PS/2 interface
apb i/o:  80000500-80000600  APB Keyboard PS/2 interface
apb i/o:  80000600-80000700  SVGA video frame buffer
apb i/o:  80000700-80000800  AHB Debug UART
apb i/o:  80000800-80000900  General purpose I/O port
apb i/o:  80000B00-80000C00  GR Ethernet MAC
apb i/o:  80000F00-80001000  AHB Status reg (failing addr
ahb mem:  800FF000-800FF080  APB BUS Plug&Play info(infe
ahb mem:  90000000-A0000000  LEON3 Debug Support Unit
ahb i/o:  FFFA0000-FFFA0100  IDE/ATA Controller
ahb mem:  FFFFF000-FFFFF800  AHB BUS Plug&Play info
ahb mem:  FFFFF800-00000000  AHB BUS Plug&Play info
```

III-D. Uso de la memoria flash

Para realizar la búsqueda y detección de la memoria flash, es necesario utilizar el comando *fscan*. Si se detecta la memoria, se leerán los parámetros de la misma mediante el protocolo CFI, reportando los resultados:

```
FLeMon$ flash
============= FLASH MEMORY REPORT ====================
MCFG11 = 0x00000888
MCTRL mode: 8 bit  flash on D[31:24]
Prom Waitstates:  read=08   write=08
Prom write enable: YES
CFI Vendor: 0x89 Device: 0x17  INTEL 64 Mbit
Command set: 0x0001 - Intel/Sharp Extended.
Extended query table address: 0x0031
--------CFI System interface info---------
VCC Program/erase voltage (Min): 2.7 Volts
VCC program/erase voltage (Max): 3.6 Volts
VPP pin not present.
Single byte/word write timeout(Typ):64 usec
Single byte/word write timeout(Max):4 times 64=256us
Buffer byte/word write timeout(Typ):128 usec
Buffer byte/word write timeout(Max):8 times 128=1024us
Indiv. Block erase timeout(Typ):1024 msec
Indiv. Block erase timeout(Max):4 times 1024=4096ms
Full chip erase not supported.
--------CFI Geometry info---------
Device size: 8388608 bytes
Device interface:0x02 - x8 & x16 via #BYTE, asynchronous
Maximum number of bytes in buffer write: 32 bytes
Erase block regions: 1
Erase blocks size in region 1: 131072 bytes (512)
Erase blocks in region 1: 64 blocks
Total Size: 8388608 bytes
```

Luego se pueden consultar los bancos, leer o escribir posiciones independientes de memoria, leer porciones de la misma, borrar y luego grabar un archivo ejecutable. En el ejemplo se muestra un borrado de la memoria con el comando *ferase* y luego con el comando *fload* se realiza la grabación de la flash con una imagen de Snapgear Linux:

```
FLeMon$ ferase all
Erased block 1 of 64
Erased block 2 of 64
```

```
[...]
Erased block 64 of 64
Erasing all flash...

FLeMon$ fload grlib_linux_pender/image.flashbz
  Flemon Flash loader
  Opening file [grlib_linux_pender/image.flashbz]
   ELF 32-bit Executable Sparc
2257036bytes Off:0x00010000 addr:0x00000000 .text<-This
0000000bytes Off:0x0023B100 addr:0x4031B100 .data
0033848bytes Off:0x0023B100 addr:0x4031B100 .bss
0000054bytes Off:0x0023B100 addr:0x00000000 .comment
0000000bytes Off:0x0023B136 addr:0x00000000 .note.GNU-st
0000069bytes Off:0x0023B136 addr:0x00000000 .shstrtab
0001680bytes Off:0x0023B2E4 addr:0x00000000 .symtab
0001165bytes Off:0x0023B974 addr:0x00000000 .strtab

32/2257036 (0\%) Time: 0,23      Write/Skip: W
22624/2257036 (1\%) Time: 17,220 Write/Skip: W
[...]
2236640/2257036 (99\%) Time: 17,76 Write/Skip: W
  bytes:2257036 Time:1714,367 seg Kb/seg: 10.53
```

Luego de grabar la memoria flash podemos resetear la tarjeta y observar el arranque del kernel mediante el segundo puerto serial utilizado como tty:

```
decompress_kernel(to: 40000000,freemem:40323538,freemem
output_data:40000000,free_mem_ptr:40323538,free_mem_ptr
  - Inputbuffer [ptr: 3168, sz: 223f24]
.....................................................
done [sz:0x31b100], booting the kernel.
Booting Linux...
PROMLIB: Sun Boot Prom Version 0 Revision 0
Linux version 2.6.21.1 (daniel@neptune) (gcc version 3.
ARCH: LEON
             Vendors         Slaves
Ahb masters:
 0( 1:  3| 0):   VENDOR_GAISLER    GAISLER_LEON3
 1( 1:  7| 0):   VENDOR_GAISLER  GAISLER_AHBUART
 2( 1: 1c| 0):   VENDOR_GAISLER  GAISLER_AHBJTAG
 3( 1: 63| 0):   VENDOR_GAISLER     GAISLER_SVGA
 4( 1: 1d| 0):   VENDOR_GAISLER   GAISLER_ETHMAC
 5( 1: 22| 0):   VENDOR_GAISLER Unknown device 22
Ahb slaves:
 0( 4:  f| 0):       VENDOR_ESA        ESA_MCTRL
 +0: 0x0 (raw:0x3e002)
 +1: 0x20000000 (raw:0x2000e002)
 +2: 0x40000000 (raw:0x4003c002)
 1( 1:  6| 0):   VENDOR_GAISLER   GAISLER_APBMST
 +0: 0x80000000 (raw:0x8000fff2)
 2( 1:  4| 0):   VENDOR_GAISLER GAISLER_LEON3DSU
 +0: 0x90000000 (raw:0x9000f002)
 3( 1: 24| 6):   VENDOR_GAISLER  GAISLER_ATACTRL
 +0: 0xfffa0000 (raw:0xa000fff3)
Apb slaves:
 0( 4:  f| 0):       VENDOR_ESA        ESA_MCTRL
 + 0: 0x80000000 (raw:0xfff1)
 1( 1:  c| 2):   VENDOR_GAISLER  GAISLER_APBUART
 + 0: 0x80000100 (raw:0x10fff1)
 2( 1:  d| 0):   VENDOR_GAISLER    GAISLER_IRQMP
 + 0: 0x80000200 (raw:0x20fff1)
 3( 1: 11| 8):   VENDOR_GAISLER  GAISLER_GPTIMER
 + 0: 0x80000300 (raw:0x30fff1)
 4( 1: 60| 4):   VENDOR_GAISLER      GAISLER_KBD
 + 0: 0x80000400 (raw:0x40fff1)
 5( 1: 60| 5):   VENDOR_GAISLER      GAISLER_KBD
 + 0: 0x80000500 (raw:0x50fff1)
 6( 1: 63| 0):   VENDOR_GAISLER     GAISLER_SVGA
 + 0: 0x80000600 (raw:0x60fff1)
 7( 1:  7| 0):   VENDOR_GAISLER  GAISLER_AHBUART
 + 0: 0x80000700 (raw:0x70fff1)
 8( 1: 1a| 0):   VENDOR_GAISLER  GAISLER_PIOPORT
 + 0: 0x80000800 (raw:0x80fff1)
 9( 1: 1d|12):   VENDOR_GAISLER   GAISLER_ETHMAC
 + 0: 0x80000b00 (raw:0xb0fff1)
10( 1: 52| 7):   VENDOR_GAISLER Unknown device 52
 + 0: 0x80000f00 (raw:0xf0fff1)
TYPE: Leon2/3 System-on-a-Chip
[...]
Sash command shell (version 1.1.1)
/> uname -a
Linux sparky 2.6.21.1 #81 Thu Dec 6 10:27:29 CET 2007
/> cat /proc/cpuinfo
cpu             : Gaisler Research - Leon3 SoC
fpu             : reserved
promlib         : Version 0 Revision 0
prom            : 0.0
type            : leon2
ncpus probed    : 1
ncpus active    : 1
CPU0Bogo        : 19.76
CPU0ClkTck      : 40000 kHz
MMU type        : Leon2
contexts        : 256
nocache total   : 524288
nocache used    : 114432
```

III-E. Licencia del programa

El código desarrollado será publicado bajo licencia GNU GPL y su desarrollo futuro se realizará bajo el modelo colaborativo del software libre. Este trabajo forma parte del proyecto FPGALibre [9], lugar desde donde podrá descargarse luego de esta publicación.

IV. Conclusión y trabajos futuros

Se dispone actualmente de un programa en fase inicial de desarrollo pero con la funcionalidad necesaria para listar los *IP cores* presentes en un diseño y grabar la memoria flash con una imagen del sistema operativo. Este software facilita el desarrollo con la GRLIB utilizando solamente herramientas libres.

Como trabajos futuros se plantea realizar pruebas con otras plataformas de hardware, distintas configuraciones, implementar la depuración por puerto Ethernet para bajar los tiempos de transferencia de las imágenes y utilizar las funciones de la unidad DSU para depuración del procesador LEON.

Referencias

[1] SPARC International Inc. The SPARC architecture manual: Version 8. [Online]. Available: http://www.sparc.org/standards/V8.pdf

[2] J. Gaisler. Leon-1 processor, first evaluation results. European Space Research and Technology Centre. [Online]. Available: http://klabs.org/DEI/Processor/sparc/Papers/gaisler.pdf

[3] Aeroflex Gaisler. LEON processor & Grlib IP-core library. [Online]. Available: http://www.gaisler.com/

[4] J. Gaisler, "An open-source VHDL IP library with plug&play configuration," in *IFIP Congress Topical Sessions*, R. Jacquart, Ed. Kluwer, 2004, pp. 711–718.

[5] Nemerix S.A. Nemerix nj1030a, gps baseband processor. [Online]. Available: http://www.nemerix.com/products/NJ1030A-ds13.pdf

[6] D. Mattsson and M. Christensson, "Evaluation of synthesizeable cpu cores," Ph.D. dissertation, Chalmers University of Technology, 2004. [Online]. Available: http://www.gaisler.com-/doc/Evaluation_of_synthesizable_CPU_cores.pdf

[7] *GRLIB IP Core User's Manual*, 1.0.19 ed. Gaisler Research, 2008, pp. 67–70.

[8] JEDEC Solid State Technology Association. Common Flash Interface (CFI). [Online]. Available: http://www.jedec.org/download/search/jesd68-01.pdf

[9] INTI Electrónica e Informática *et al.*, "Proyecto FPGA Libre," http://fpgalibre.sourceforge.net/.