Hardware libre: clasificación y desarrollo de hardware reconfigurable en entornos GNU/Linux

Ivan González, Juan González, Francisco Gómez-Arribas

Escuela Politécnica Superior, Universidad Autónoma de Madrid, Spain,

{Ivan.Gonzalez, Juan.Gonzalez, Francisco.Gomez}@ii.uam.es

http://www.ii.uam.es

6 de Septiembre de 2003

Resumen

En este artículo se aborda el *hardware* libre, proponiéndose definiciones en función de su naturaleza. Se estudian los dos tipos, el estático, que tiene existencia física, y el reconfigurable, descrito mediante lenguajes HDL, centrándose en los criterios para considerarlos libres. En la primera parte se revisan y amplían las ideas del *proyecto hardware abierto* de Microbótica, dando una definición de *hardware estático libre* y proponiendo una clasificación según los programas de diseño empleados. En la segunda se aborda el *hardware* reconfigurable, centrándose en las herramientas de desarrollo para GNU/Linux. Finalmente se muestran ejemplos de sistemas diseñados enteramente bajo GNU/Linux, tanto de *hardware* estático como reconfigurable.

Introducción

El ***software* libre**[1] ofrece al usuario **cuatro libertades**: libertad de uso, de estudio y modificación, de distribución, y de redistribución de las mejoras. Existen licencias que las garantizan y que dan una cobertura legal, como por ejemplo la GPL[2].

El ***hardware* abierto o libre**[1] toma estas mismas ideas para aplicarlas en su campo. Es una propuesta casi tan antigua como la del *software* libre[3], sin embargo su empleo no es tan directo. **Compartir diseños *hardware* es más complicado**. No hay una definición exacta (se pueden encontrar referencias a distintos artículos en [4]). Incluso el propio Richard Stallman afirma[5] que las ideas del *software* libre se pueden aplicar a los ficheros necesarios para su diseño y especificación (esquemas, PCB, etc), pero no al circuito físico en sí.

Al no existir una definición clara de *hardware* abierto, cada autor lo interpreta a su manera. Se han creado licencias[6], algunas de las cuales están todavía en desarrollo.

Para abordar el estudio comenzaremos estableciendo una primera clasificación, según su naturaleza. Existen los siguientes tipos:

- ***Hardware* estático**. Conjunto de elementos materiales de los sistemas electrónicos. Tiene una existencia física (se puede "tocar"). Esta propiedad no la tiene el software, por lo que surgen una serie de problemas que se discuten en el apartado 1.1. En la primera parte de este artículo revisaremos y ampliaremos las ideas del **proyecto hardware abierto**[7], comenzado en 1997 por Microbótica S.L. Se propone una definición práctica de *hardware* estático libre y se establece una clasificación de los diseños, en función de las herramientas de desarrollo empleadas, que limitan en mayor o menor medida las libertades ofrecidas por el autor.

[1] En este artículo consideraremos que *hardware abierto* y *hardware libre* son sinónimos, y los utilizaremos indistintamente. Sin embargo, existe ambigüedad en la lengua Inglesa. El término *free hardware* puede significar tanto *hardware libre* como *hardware gratuito*. Esto último es incorrecto.

- **Hardware reconfigurable**. Es el que se describe mediante un lenguaje HDL, (*Hardware Description Language*, lenguaje de descripción hardware) y que permite especificar con todo detalle su estructura y funcionalidad. A partir de este código se generan unos ficheros de configuración (*bitstreams*) para que los dispositivos del tipo FPGA se reconfiguren, funcionando según lo descrito. En la segunda parte se describe con más detalle este *hardware* y cómo lo podemos desarrollar en plataformas GNU/Linux, utilizando el lenguaje VHDL y las FPGAs de la empresa Xilinx[8].

Dada su diferente naturaleza, al hablar de *hardware* libre hay que especificar de qué tipo de hardware se está hablando. Para hacer que el *hardware* reconfigurable sea libre, sólo hay que aplicar una licencia GPL a su código. Sin embargo, no está tan claro qué se entiende por *hardware* libre cuando nos referimos al *hardware* estático.

Parte I
Hardware estático

A lo largo de esta parte, al hablar de *hardware* nos estamos refiriendo al *hardware* estático.

1. Definición

1.1. Los problemas del hardware abierto

No se pueden aplicar directamente las cuatro libertades del software libre al *hardware*, dada su diferente naturaleza. Uno tiene existencia física, el otro no. Aparecen una serie de problemas:

1. **Un diseño físico es único**. Si yo construyo una placa, es única. Para que otra persona la pueda usar, bien le dejo la mía o bien se tiene que construir una igual. La compartición tal cual la conocemos en el mundo del software NO ES POSIBLE.
2. **La compartición tiene asociado un coste**. La persona que quiera utilizar el *hardware* que yo he diseñado, primero lo tiene que fabricar, para lo cual tendrá que comprobar los componentes necesarios, construir el diseño y verificar que se ha hecho correctamente. Todos esto tiene un coste.
3. **Disponibilidad de los componentes**. ¿Están disponibles los chips?. Al intentar fabricar un diseño nos podemos encontrar con el problema de la falta de material. En un país puede no haber problema, pero en otro puede que no se encuentran.

Una primera propuesta para definir el *hardware* libre es la siguiente:

> *El hardware libre ofrece las mismas cuatro libertades que el software libre, pero aplicadas a los planos del hardware.*

Si en el software hablamos de fuentes, aquí hablamos de planos. A partir de ellos podemos fabricar el *hardware*. El proceso de construcción tiene asociado un coste, que no existe en el caso del *software*. Sin embargo los planos están disponibles para que cualquiera los pueda usar, modificar y distribuir.

1.2. Tipos de planos en electrónica

Existen tres tipos de planos, o de ficheros, que describen nuestro diseño:

- **Esquemático**: Indica los componentes lógicos y las señales que se conectan entre ellos, pero no nos dice nada de cómo es físicamente la placa.

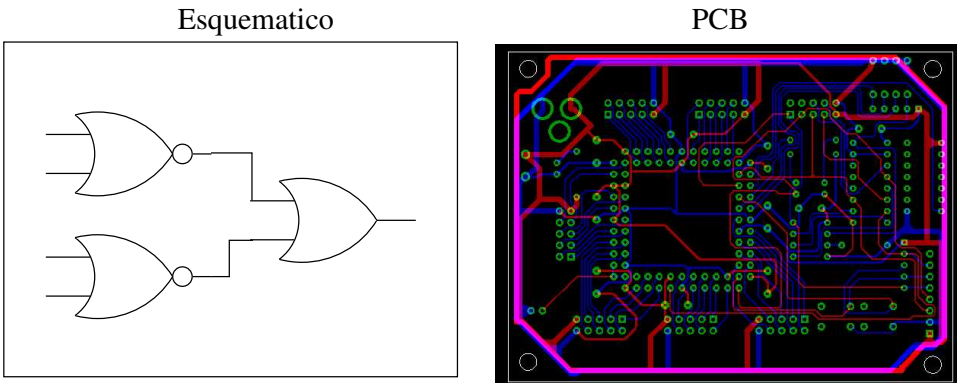


Figura 1: Un trozo de un esquemático y un PCB

- **Circuito Impreso** (PCB, Printed Circuit Board). Indica el lugar físico en el que situar los componentes, sus dimensiones, encapsulados y qué caminos siguen las pistas para unir sus pines. Nos describe con detalle cómo es físicamente la placa y las dimensiones que tiene.
- **Fichero de fabricación** (GERBER). Contiene toda la información necesaria para que se puedan fabricar los PCBs en la industria. Este es un fichero para las máquinas. (Sería algo similar a lo que es el formato *Postcript* para las impresoras).

Para diseñar la placa usamos los esquemáticos y el PCB. El fichero GERBER es el que obtenemos como resultado y nos permite realizar una fabricación industrial de nuestro PCB. Este fichero puede no haberlo generado el diseñador. A veces es el propio fabricante el que lo genera, a partir de la información que hay en el PCB.

1.3. Propuesta de definición

Teniendo en cuenta los tres planos que necesitamos en electrónica, podemos definir el *hardware* abierto de la siguiente manera:

> *Un diseño se considera hardware abierto si ofrece las 4 libertades del software libre en el esquemático, PCB y fichero para la fabricación (este último puede no estar disponible).*

El fichero de fabricación puede no estar disponible si se trata un prototipo, del que no se ha hecho una tirada industrial. O también puede ocurrir que se haya fabricado industrialmente, pero a partir del PCB. En este caso el autor dispone de unos *fotolitos* impresos, pero no de un fichero electrónico.

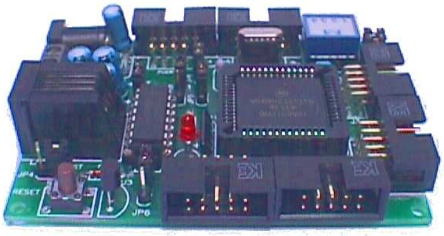
2. Clasificación

2.1. El formato de los planos

Para el fichero de fabricación existe un estándar industrial, denominado GERBER, muy extendido y que está perfectamente documentado. Por ejemplo con el programa libre *gerbv*[9] se pueden visualizar.

Desgraciadamente, **no existe un estándar para el formato de los esquemáticos y el PCB**. Cada aplicación utiliza el suyo propio. Y prácticamente todas las herramientas de diseño electrónico (EDA) son propietarias, con formatos propietarios.

Hay un proyecto en marcha, gEDA[10] (GPL+EDA), que tiene como objetivo desarrollar herramientas EDA bajo licencia GPL. Es un proyecto muy prometedor pero al día de hoy está en desarrollo y todavía no está a la altura de los programas propietarios, usados por la mayoría de diseñadores hardware. No obstante es “usable”.



Tarjeta CT6811 (PPX)

Tarjeta JPS (MML)

Figura 2: Fotos de las placas libres CT6811 y JPS

Actualmente, y desde un punto de vista práctico, **los diseños se hacen usando programas propietarios**, como el Orcad, Tango, Eagle, Protel, etc, **que utilizan formatos propietarios**, un impedimento para su compartición. Sólo podrán ver/modificar ese diseño aquellos usuarios que dispongan de la licencia de ese programa propietario.

Nuestra propuesta, más orientada hacia la práctica que hacia lo "legal", es que **sea el autor el que decida si su diseño es libre o no, con independencia del formato empleado**. Obviamente, si es un formato propietario, las libertades que nos ofrece el autor se ven restringidas.

Y es precisamente esta restricción la que nos va a servir para establecer una clasificación de los diseños libres.

2.2. Clasificación según las restricciones impuestas por el programa de diseño

Se pretende que al conocer el tipo de diseño libre, sepamos a qué restricciones estamos sometidos. Para ello a los diseño se le dota de tres componentes, cada una asociada a uno de los tres planos electrónicos: X para el esquemático, Y para el PCB y Z para el fichero de fabricación.

La componente Z sólo puede tomar dos valores: '**L**' si está disponible y '**X**' si no. Si está disponible tiene que estar en **formato Gerber**. Las componentes X e Y pueden tener los siguientes valores:

- **L**. **El Programa con el que se ha diseñado el plano es libre**. Por tanto, no hay ninguna restricción. Por ejemplo si se han usado las herramientas gEDA.
- **P**. **El Programa es propietario** y sólo es se puede ejecutar en **sistemas operativos propietarios**. Por ejemplo si se ha empleado el programa Orcad o Tango, que sólo están disponibles para plataformas Windows (al menos de momento).
- **M**. **El programa es propietario**, pero es **multiplataforma**, siendo alguna de las plataformas un **sistema operativo libre**. Por ejemplo el programa Eagle[11], disponible de forma nativa para Windows y Linux.

Según esta clasificación podemos tener diseños LLL, PPL, PML... Vamos a analizar tres de los tipos más importantes:

- **Diseños LLL**. Son **hardware libre "pura sangre"**. No hay restricciones. Tanto el esquemático como el PCB se han creado con programas libres. Cualquier persona los puede ver/modificar usando el programa libre con el que se han diseñado. Además, el fichero GERBER está disponible. Sólo hay que pagar los costes de fabricación, para disponer del diseño. Los autores no conocen ningún diseño de este tipo, pero eso no quiere decir que no los haya. **Es hacia lo que deben tender los diseñadores de *hardware* libre**.

- **Diseños MML**. Es **lo más práctico para el diseñador**, al día de hoy. Para el esquemático y el PCB se emplea un programa propietario multiplataforma. Cualquiera lo puede fabricar, por estar disponible el fichero GERBER. Para verlo/modificarlo hay que obtener una licencia del programa en cuestión, pero al menos se puede usar un sistema operativo libre. Por ejemplo el programa Eagle, de la casa cadsoft. Además se puede descargar gratuitamente una versión para estudiantes, ilimitada en el tiempo, que restringe el tamaño y la complejidad de los diseños que se pueden realizar. La placa JPS[12, 13] es de este tipo.

- **Diseños PPX**. Es **lo más restrictivo**. Tanto el esquemático como el PCB se han diseñado usando programas propietarios sólo disponibles para sistemas operativos propietarios, como por ejemplo los programas OrCad, Tango, Protel, etc, sólo disponibles para plataformas Windows. Además no está disponible el fichero de fabricación, bien por ser un prototipo o bien porque al fabricante se le ha dado directamente el PCB, y tiene un programa que le permite leerlo y fabricar. Por ejemplo las tarjetas CT6811[14] (ver figura 2) y CT293[15], que son las empleadas para el microbot Tritt[16].

3. Un ejemplo de hardware abierto: La tarjeta JPS

En la Escuela Politécnica de la Universidad Autónoma de Madrid hemos desarrollado la tarjeta JPS[12, 13] (ver figura 2), una entrenadora para las FPGAs de la familia 4000 y Spartan I de Xilinx[8], que se está utilizando en el laboratorio de Estructura y Diseño de Circuitos Digitales. Según la clasificación establecida previamente, es del **tipo MML**. Para su diseño se ha utilizado el programa Eagle[11], en una máquina corriendo Debian/Sarge.

Al tratarse de *hardware* libre, están disponibles todos los planos. Algunas de sus ventajas son:

- **Cualquiera la puede fabricar**. Los alumnos, universidades o empresas que la encuentren interesante la pueden construir.

- **Cualquiera la puede modificar**. Seguramente no se adaptará a las necesidades de docencia de otras universidades o centro de investigación. Tienen la posibilidad de modificar la placa, en vez de empezar a diseñarla desde cero.

- **Cualquier empresa la puede comercializar**, y cobrar por ofrecer los servicios de fabricación y verificación del correcto funcionamiento, así como mantenimiento.

Parte II
Hardware reconfigurable

4. Introducción

El *hardware* reconfigurable es aquél que viene descrito mediante un lenguaje HDL (*Hardware Description Language*). Su naturaleza es completamente diferente a la del *hardware* estático. Se desarrolla de una manera muy similar a como se hace con el software. Ahora nuestros diseños son ficheros de texto, que contienen el "código fuente". Se les puede aplicar directamente una licencia libre, como la GPL.

Los problemas no surgen por la definición de qué es libre o qué debe cumplir para serlo, sino que aparecen con las herramientas de desarrollo necesarias. En esta parte analizaremos los programas que nosotros hemos empleado para su diseño en sistemas GNU/Linux, centrándonos en el lenguaje VHDL y las FPGAs de Xilinx. Veremos que dificultades aparecen y su relación con el software libre.

4.1. Lenguajes HDL

El *hardware* se puede describir utilizando un lenguaje HDL, como por ejemplo VHDL, Verilog, Handel C, etc. De esta forma, los diseños se convierten en ficheros de texto ASCII ("código fuente"), que describen

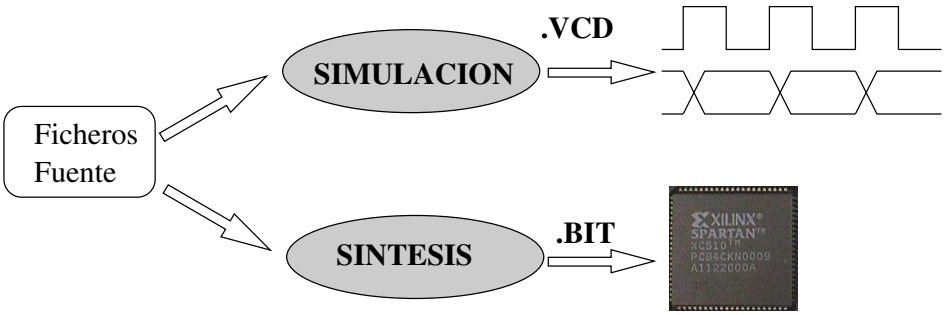


Figura 3: Ciclo de diseño de hardware reconfigurable: simulación y síntesis

tanto la estructura del diseño como el comportamiento de las partes integrantes. Se pueden crear librerías de componentes que luego se usan en diseños más complejos.

Inicialmente estos lenguajes se empleaban para describir de una forma no ambigua el *hardware* y poder **realizar simulaciones**.

4.2. FPGAs

Las FPGAs son dispositivos que nos permiten implementar circuitos digitales. Están compuestos por bloques iguales, configurables, llamados CLBs, que se unen dinámicamente según cómo se especifique en la memoria de configuración. De esta forma, cambiando el contenido de la esta memoria, se establecen unas uniones diferentes entre los CLBs, obteniéndose un dispositivo u otro.

El fichero que contiene la configuración se denomina *bitstream*. La características fundamental de las FPGAs es que son **Dispositivos universales**. Se pueden "convertir" en cualquier diseño digital, según el *bitstream* que se cargue en su memoria de configuración.

4.3. Hardware reconfigurable

Tomando por un lado los lenguajes HDL y por otro las FPGAs, obtenemos el **hardware reconfigurable**. A partir de los ficheros fuente, se realiza una simulación, lo que nos permite comprobar si nuestro diseño cumple las especificaciones. Por otro lado, podemos obtener el *bitstream*, que al descargarlo en una FPGA conseguimos que nuestro diseño hardware se "materialice". Esta fase se denomina genéricamente síntesis, aunque internamente consta de dos fases: síntesis e implementación. En la figura 3 se ha esquematizado.

4.4. Hardware reconfigurable libre

El **hardware reconfigurable se puede compartir exactamente igual que el software**. Las características que tiene son:

- Se pueden ofrecer las 4 mismas libertades del software libre a los ficheros en HDL. Por ejemplo distribuyéndolos bajo licencia GPL.
- Aparecen **comunidades *hardware*** que comparten información, como OpenCores[17] y OpenCollector[18]
- Se pueden crear **repositorios hardware**, lo que permite que muchas personas puedan participar en el desarrollo.
- Pueden aparecer **distribuciones** que recopilen todo el hardware libre existente.

De igual forma que para ejecutar el software necesitamos un máquina que tenga un procesador, para probar nuestro *hardware* necesitamos una plataforma con una FPGA en la que "descargar" los diseños (un

hardware estático). La tarjeta JPS es una de ellas, que además es libre (En el sentido que se le ha dado al hardware estático).

5. Simulación en GNU/Linux

Nos centraremos en el lenguaje VHDL y las FPGAs de Xilinx. Las herramientas que hemos empleado para el diseño de hardware reconfigurable son el GHDL[19], para realizar la simulación, y el GTKWAVE[20] para visualizar los resultados. Ambas aplicación se distribuyen bajo licencia GPL.

5.1. GHDL y GTKWAVE

El GHDL es un analizador/simulador de VHDL basado en el compilador GCC. Es un proyecto muy prometedor y que aunque está todavía en desarrollo, permite realizar simulaciones complejas, como por ejemplos los procesadores DLX[21] y leon1[22].

GHDL genera un fichero ejecutable a partir de los archivos en VHDL. Al ejecutarlo se realiza la simulación. Podemos obtener los resultados en un fichero VCD, lo que nos permite visualizar las señales usando el GTKWAVE.

5.2. Un ejemplo

A continuación se muestra un ejemplo de un contador de 8 bits:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity cont8 is
  port (clk   : in std_logic; -- Reloj
        clear : in std_logic;
        q     : out std_logic_vector (7 downto 0)); --Salida
end cont8;

architecture beh of cont8 is
signal cuenta : std_logic_vector (7 downto 0);

begin
  output: process(clk,clear)
  begin
    if (clear='0') then
      q<="00000000";
      cuenta<="00000000";
    elsif (clk'event and clk='1') then
      cuenta<=(cuenta+1);
      q<=cuenta;
    end if;
  end process;
end beh;
```

Este contador se encuentra en el fichero *cont8.vhdl*. Para simularlo es necesario crear un banco de pruebas (testbench) que introduzca la señal de reloj y coloque un valor adecuado en la entrada de *reset*. Lo llamamos *tb_cont8.vhdl*. El proceso para realizar la simulación completa es el siguiente:

1. **Realizar el análisis**. A partir de los ficheros tb_cont8.vhdl y cont8.vhdl se obtendrán los ficheros objeto .o:

   ```
   $ ghdl -a -ieee=synopsys *.vhdl
   ```

2. **Elaboración**. A partir de los ficheros objeto obtenemos el ejecutable (tb_cont8):

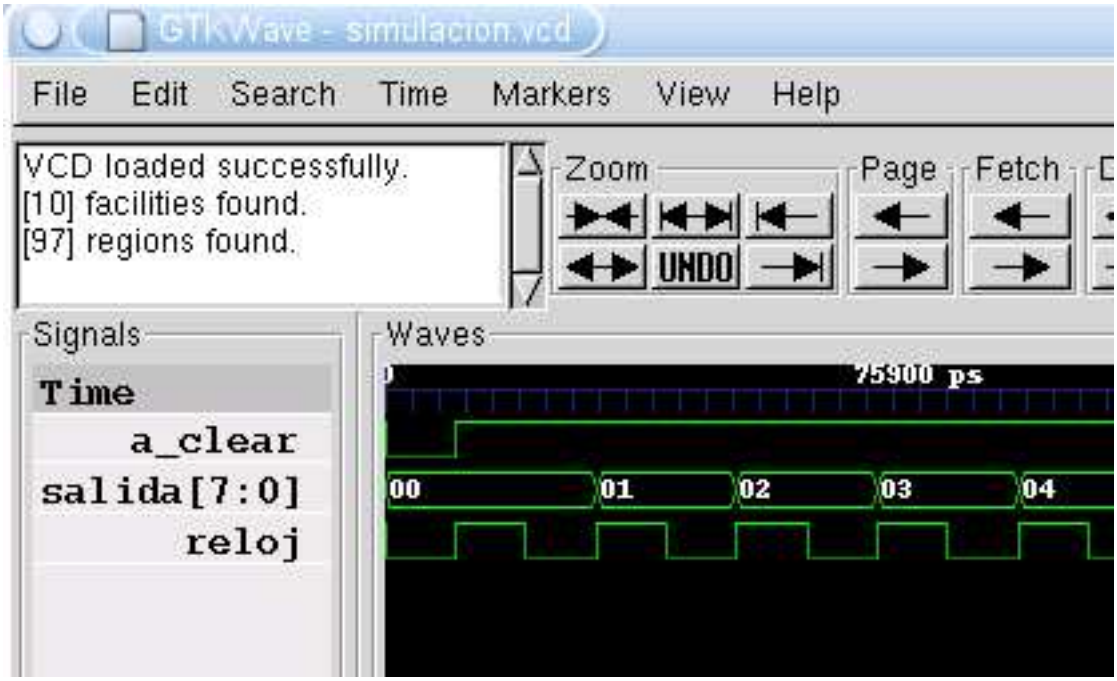


Figura 4: Resultados de la simulación con GTKWAVE

```
$ ghdl -e --ieee=synopsys tb_cont8
```

3. **Simulación**. Ejecutamos el programa y obtenemos el fichero de simulación (simulación.vcd). Especificamos el tiempo que queremos simular.

```
$ ./tb_cont8 --vcd=simulacion.vcd --stop-time=200ns
```

4. **Visualización**. Con el programa gtkwave vemos los resultados de la simulación

```
$ gtkwave simulacion.vcd
```

Todos los pasos anteriores se realizan más fácilmente creando un fichero *Makefile*, pero se ha dejado así por claridad. En la figura 4 se muestra una parte de los resultados obtenidos con GTKWAVE.

6. Síntesis/Implementación

La fase de síntesis/implementación toma el diseño en vhdl y genera el *bitstream*. Analizaremos las dos fases por separado y su relación con GNU/Linux.

6.1. Síntesis

En el proceso de síntesis se infieren los componentes *hardware* básicos a partir del diseño en vhdl y se genera un fichero *netlist*, que especifica qué componentes básicos se utilizan y cómo están conectados.

A partir de los ficheros .vhdl se obtiene el netlist, en formato EDIF, que está estandarizado. Los autores no conocen herramientas de síntesis libres pero podría haberlas, puesto que **la información necesaria está disponible**. Si no existen ahora se podrán programar en un futuro.

6.2. Implementación

La implementación toma el fichero *netlist* y genera el *bitstream*, realizando muchos pasos intermedios que no se detallan en este artículo. **Esta parte es muy dependiente del dispositivo FPGA** utilizado y es necesario conocer todos sus detalles internos para poder programarlo. Y aquí es donde surgen los problemas. **La tecnología de los dispositivos FPGA es considerada secreto industrial**. No es posible conocer toda la información de configuración. Por tanto **no es posible el diseño de herramientas libres**. Este punto es muy importante. No existen herramientas libres y además, potencialmente, no pueden existir, al menos con el panorama actual.

6.3. Trabajando en GNU/Linux

Para realizar la fase de síntesis/implementación es necesario emplear **herramientas propietarias**, algunas de las cuales están disponibles para Linux. Tienen una licencia con coste alto, aunque es posible disponer de versiones gratuitas limitadas en el tiempo. Inicialmente estaban disponibles sólo para estaciones de trabajo bajo Unix. Actualmente se han portado a arquitecturas PC con Windows. Poco a poco, están empezando a interesarse por el sistema operativo Linux.

En el caso particular de trabajar con FPGAs de la familia Xilinx, es necesario emplear su herramienta integrada de diseño, **ISE**[23], disponible para plataformas Windows y Solaris. Esta herramienta permite cubrir todas las fases del diseño: **edición** en VHDL/Verilog, **compilación**, **simulación** mediante la herramienta externa ModelSim, **síntesis** (XST), **implementación** y **generación** del bitstream. Además, posibilita la integración con otros sintetizadores como Leonardo, Synplicity, FPGA Express, etc.

La síntesis e implementación sólo se puede realizar con esta herramienta. **Se puede utilizar en Linux a través de Wine**[24]. En el manual de instalación se muestran los pasos a seguir para su instalación en Red Hat 7.2[25, 26, 27], aunque estas mismas instrucciones permiten instalar la herramienta sobre otras distribuciones, como Debian.

Esta solución es la que se ha probado y con la que se ha conseguido cerrar el ciclo de diseño desde máquinas Linux, aunque es necesario disponer de una licencia.

7. Plataformas hardware

Para poner en funcionamiento el *hardware* reconfigurable sintetizado es necesaria una plataforma. En el mercado están disponibles distintas placas de desarrollo para FPGA, con interfaces muy diferentes: puerto serie/paralelo, PCI, etc. El interface serie/paralelo permite a priori un sencillo manejo de la placa incluso desde Linux, aunque es posible no disponer de la aplicación para el mismo (y por el momento nada de software libre). **Una excepción es la placa JPS, que nació con la filosofía abierta, y dispone de un software de descarga (download) para Linux.**

Las placas PCI presentan el problema de la disponibilidad del driver adecuado. La placa RC1000PP de Celoxica dispone de un driver (incluyendo las fuentes) para el kernel 2.4x y librerías de manejo de la placa, aunque no se da soporte. Esta placa resulta muy interesante para aquellos desarrollos donde es necesario la convivencia del diseño con un PC a alta velocidad. Otro aspecto importante, son aquellas plataformas con Ethernet las cuales permiten su manejo desde la web. Este es el caso de la plataforma Labomat3[28].

8. Cerrando el ciclo de diseño en Linux

El problema hasta ahora es que no existen herramientas que nos permitan cerrar el ciclo de diseño de *hardware* reconfigurable bajo máquinas GNU/Linux. No ya que no haya herramientas libres, sino que ni siquiera las hay propietarias. Mostraremos la solución que hemos empleado. Recapitulando:

- **Existen herramientas libres de edición**, para editar los ficheros fuentes. Cualquier editor ASCII nos valdría. En concreto, EMACS, tiene un modo de resaltado de sintaxis para los ficheros en VHDL.
- **Existen herramientas de simulación libres**. Las que hemos empleado con éxito son el GHDL y el GTKWAVE

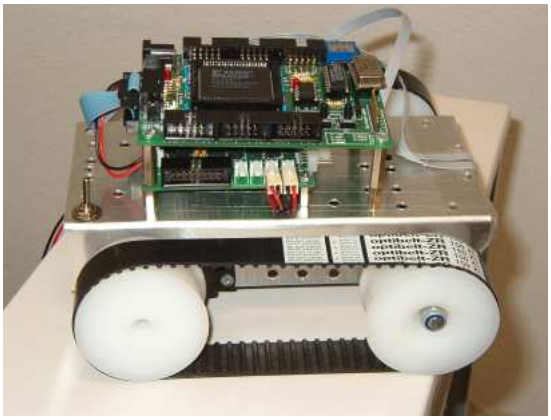


Figura 5: Robot de docencia

- **Existen herramientas de síntesis, pero no son libres**. Al menos las hay para Linux.
- No hay herramientas para la implementación. Pero lo peor es que no puede haberlas libres. Existen varias soluciones:
  - Usar VMWARE, una máquina virtual, sobre la que se instala Windows y las herramientas de diseño.
  - Utilizar WINE. Esta es la opción que hemos empleado con éxito y que permite ejecutar el entorno de diseño ISE 4.1 enteramente bajo Linux y es bastante "usable". Esta herramienta nos permite editar, sintetizar, implementar y configurar el dispositivo.
- En el caso de trabajar con la placa JPS (hardware estático libre, tipo MML), se dispone de un software para Linux que permite descargar los *bitstreams* en ella, utilizando un cable constituido por la tarjeta CT6811.

Parte III
Aplicaciones y Conclusiones

9. Aplicaciones

En este apartado mostraremos ejemplos de aplicaciones relacionadas con el *hardware* libre y su desarrollo bajo GNU/Linux

9.1. Robot de Docencia

En el Laboratorio de Arquitectura e Ingeniería de Computadores, los alumnos trabajan en el estudio y diseño de una CPU de docencia, llamada *PandaBear*, de tipo RISC, de 16 bits y con un total de 8 instrucciones. Ha sido implementada en VHDL, sintetizada en una FPGA de tipo Spartan I (XCS10) y probada en la placa JPS.

Una de las aplicaciones realizadas ha sido el **robot de docencia**[31], mostrado en la figura 5, programado para seguir una línea negra sobre un fondo blanco. Este es un ejemplo de aplicación que tiene un *hardware* estático libre, la tarjeta JPS, y un *hardware* reconfigurable.

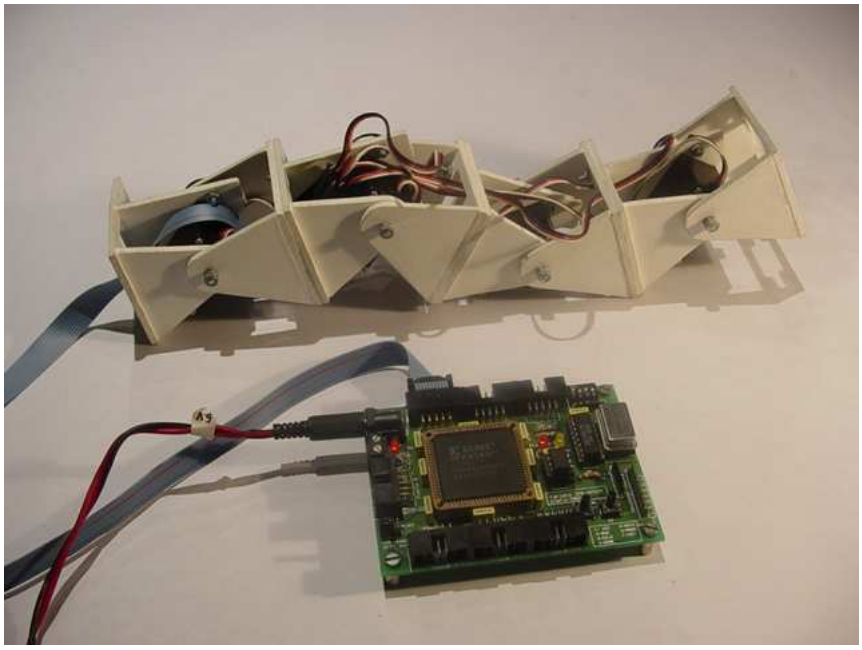


Figura 6: El robot Gusano Cube Reloaded, controlado desde una tarjeta JPS

9.2. Laboratorio Virtual

En el laboratorio[29] se dispone de un conjunto de aplicaciones que permiten al usuario, con solo disponer de un navegador, la utilización de los recursos disponibles en la plataforma reconfigurable Labomat3. Se ha desarrollado una arquitectura cliente-servidor en Linux, que consigue el funcionamiento en remoto del compilador de código fuente, la carga de módulos HW diseñados con las herramientas del fabricante utilizadas en remoto. Se consigue la monitorización del funcionamiento conjunto de la parte HW y SW del sistema. Este laboratorio se ha utilizado para un curso de post-grado en codiseño[30].

Este proyecto **se está actualmente expandiendo para permitir que se pueda realizar el proceso de síntesis/implementación**, permitiendo que cualquier alumno con acceso a un navegador pueda enviar su diseño y obtener el *bitstream*. De esta manera se puede realizar la simulación desde GNU/Linux y luego enviar el diseño al laboratorio para su síntesis. No hace falta tener licencias de la herramienta en cada puesto.

9.3. Cube Reloaded

Otro ejemplo de robot que se controla con la tarjeta JPS es **Cube Reloaded**, un robot ápodo (ver figura 6). Se ha implementado un sistema de control basado en un circuito combinacional y secuencial descrito en VHDL[32] y cargado en la tarjeta JPS. En una memoria ROM se han almacenado las secuencias de movimiento, que se van enviando a los servos para conseguir que avance en línea recta. Este *hardware* reconfigurable está bajo licencia GPL.

9.4. Sistema de cifrado

Aplicación de codiseño para el cifrado de imágenes, usando el algoritmo IDEA. Como plataforma se ha empleado la tarjeta RC1000PP, pinchada en un slot PCI en una máquina GNU/Linux. Se ha desarrollado un coprocesador en la FPGA para el cifrado y una aplicación en GTK que muestra los resultados al usuario.

No sólo es una aplicación para GNU/Linux, sino que todo el proceso de diseño se ha desarrollado en la misma máquina, utilizándose el ISE 5.1 a través de Wine y los drivers para Linux que incluye la tarjeta.

10. CONCLUSIONES

Al hablar de hardware libre hay que hacer primero la distinción entre **hardware estático** y **reconfigurable**. Para el ***hardware* estático** se ha propuesto una definición y se ha establecido una clasificación de los diseños en función de las restricciones impuestas por las aplicaciones de diseño. **Es el autor el que decide si su diseño es o no abierto, y no la aplicación empleada**.

En el caso del ***hardware* reconfigurable**, se ha conseguido cerrar el ciclo completo de diseño en una máquina GNU/Linux, realizándose la compilación, simulación, síntesis y descarga en una FPGA. Para la compilación y simulación hemos empleado el **GHDL** junto con el **GTKWAVE**, ambos programas libres y para la síntesis el entorno **ISE** de Xilinx, ejecutado a través de **Wine**.

Se podrían realizar sintetizadores libres que generen un *netlist* en formato EDIF, pero actualmente **no sería posible disponer de un entorno completamente libre puesto que los fabricantes no publican la información, considerada como secreto industrial**. El primer paso para lograrlo sería la existencia de una "Open FPGA".

Una posible solución al tema de las licencias de las herramientas propietarias es utilizar un laboratorio virtual, constituido por un servidor en el que se ejecute el sintetizador propietario, de momento a través de Wine, que permita a los usuarios realizar la síntesis (¿Granjas de síntesis?). Se está trabajando en esa dirección.

Referencias

[1] Definición de Software Libre en la página de la *Free Sofware Foundation* (FSF). `http://www.gnu.org/philosophy/free-sw.es.html`. [Última consulta: 03/Sep/2003]

[2] Licencia pública GNU. Traducción al castellano. `http://gugs.sindominio.net/gnu-gpl/gples.html`. [Última consulta: 03/Sep/2003]

[3] Graham Seaman. "Free Hardware Design - Past, Present, Future". Oekonux conference, May 2001. Disponible on-line en `http://www.opencollector.org/Whyfree/freedesign.html`. [Última consulta: 03/Sep/2003].

[4] Diferentes artículos sobre hardware abierto. Página del proyecto OpenCollector. `http://www.opencollector.org/Whyfree/`. [Última consulta: 03/Sep/2003]

[5] Richard Stallman. "Free Hardware". Linux Today. Jun-1999. Disponible en línea en: `http://linuxtoday.com/news_story.php3?ltsn=1999-06-22-005-05-NW-LF`. [Última consulta: 03/Sep/2003]

[6] Resumen de licencias para hardware abierto. Página del proyecto OpenCollector. `http://www.opencollector.org/hardlicense/licenses.html`. [Última consulta: 03/Sep/2003]

[7] Proyecto Hardware Abierto, de Microbótica S.L. `http://www.microbotica.com/ha.htm`. [Última consulta: 03/Sep/2003]

[8] Xilinx. Empresa fabricante de FPGAs. `http://www.xilinx.com/`. [Última consulta: 03/Sep/2003]

[9] Un visualizar libre para ficheros GERBER. `http://gerbv.sourceforge.net/`. [Última consulta: 05/Sep/2003]

[10] Herramientas de diseño electrónico bajo licencia GPL. `http://www.geda.seul.org/`. [Última consulta: 05/Sep/2003]

[11] Programa de diseño electrónico EAGLE, de la casa Cadsoft. `http://www.cadsoft.de/`. [Última consulta: 05/Sep/2003]

[12] J. Gonzalez, P. Haya, S. Lopez-Buerdo, and E. Boemo, "Tarjeta entrenadora para FPGA, basada en un hardware abierto", Seminario Hispabot 2003, Alcalá de Henares, Mayo 2003.

[13] Tarjeta JPS. Esquemas, PCB y planos en línea. http://www.iearobotics.com/personal/juan/doctorado/jps-xpc84/jps-xpc84.html. [Última consulta: 05/Sep/2003]

[14] Tarjeta CT6811. Hardware abierto (PPX). http://www.iearobotics.com/proyectos/ct6811/ct6811.html. [Última consulta: 05/Sep/2003]

[15] Tarjeta CT293. Hardware abierto (PPX). http://www.iearobotics.com/proyectos/ct293/ct293.html. [Última consulta: 05/Sep/2003]

[16] Microbot Tritt, basado en las tarjetas abiertas CT6811 y CT293. http://www.iearobotics.com/proyectos/tritt/tritt.html. [Última consulta: 05/Sep/2003]

[17] Proyecto OpenCores. http://www.opencores.org/. [Última consulta: 05/Sep/2003]

[18] Base de datos con diseños y herramientas EDA libres. http://www.opencollector.org/. [Última consulta: 05/Sep/2003].

[19] GHDL Home page. http://ghdl.free.fr/. [Última consulta: 05/Sep/2003]

[20] GTKWAVE Home Page. http://6371.lcs.mit.edu/currentsemester/cadtools/gtkwave/wave.html. [Última consulta: 05/Sep/2003]

[21] Procesador DLX en VHDL. http://www.cse.lehigh.edu/~caar/lou/dlx.html. [Última consulta: 05/Sep/2003]

[22] Procesador Leon1. http://www.estec.esa.nl/microelectronics/leon/leon.html. [Última consulta: 05/Sep/2003]

[23] Herramienta ISE, de Xilinx. http://www.xilinx.com/ise/ise_promo/ise5_eval.htm. [Última consulta: 05/Sep/2003]

[24] Proyecto Wine. http://www.winehq.com/. [Última consulta: 05/Sep/2003]

[25] Configurar Linux para ISE: http://support.xilinx.com/xlnx/xil_ans_display.jsp?iLanguageID=1&iCountryID=1&getPagePath=12384. [Última consulta: 05/Sep/2003]

[26] Como instalar ISE: http://toolbox.xilinx.com/docsan/xilinx4/data/docs/irn/ch32.html. [Última consulta: 05/Sep/2003]

[27] Configurar variables de entorno de ISE: http://toolbox.xilinx.com/docsan/xilinx5/data/docs/irn/irn0027_7.html. [Última consulta: 05/Sep/2003]

[28] Tarjeta Labomat3. http://lslwww.epfl.ch/labomat/. [Última consulta: 05/Sep/2003]

[29] Proyecto Laboweb. http://www.ii.uam.es/~laboweb/LabWeb/index.php3?seccion=0&pagina=0. [Última consulta: 05/Sep/2003]

[30] Gómez-Arribas FJ, González I, González J y Martínez J, "Laboratorio Web para Prototipado y Verificación de Sistemas Hardware/Software". III Jornadas sobre Computación Reconfigurable y Aplicaciones, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Septiembre 2003.

[31] Robot de Docencia. Club de Robótica-Mecatrónica de la UAM. http://www.ii.uam.es/~mecatron/index.php3?seccion=4&pagina=8. [Última consulta: 06/Sep/2003]

[32] González J., González I. y Boemo E, "Alternativas Hardware para la Locomoción de un Robot Ápodo". III Jornadas sobre Computación Reconfigurable y Aplicaciones, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Septiembre 2003.