Programación de Prácticas en Electrónica Digital

Francisco J. Álvarez García, José A. Moreno Zamora, José V. Valverde Sánchez, José M. García Barrero
Dept. Ingeniería Eléctrica, Electrónica y Automática
Universidad de Extremadura, Badajoz, 06006, España
Email: falvarezm@alumnos.unex.es, josan@unex.es, valsan@unex.es, gbarrero@unex.es

***Resumen*—En esta comunicación se presenta la programación de las clases prácticas de laboratorio de la asignatura Electrónica Digital del quinto semestre de la titulación de Grado en Ingeniería Electrónica y Automática (Rama industrial) de la Universidad de Extremadura. Con ella se pretende proporcionar un refuerzo a los conocimientos teóricos de la asignatura y que además proporcionen un diseño de un microprocesador, de carácter didáctico, que corresponde a la ultima parte de la misma.**

I. Introducción

La asignatura de Electrónica Digital se encuadra dentro de la titulación Grado en Ingeniería Electrónica y Automática (Rama industrial), en el quinto semestre. Los alumnos acceden tras haber cursado Componentes y Sistemas Electrónicos, en donde se incluye: introducción a la Electrónica Digital (funciones, simplificaciones, dispositivos lógicos).

Los descriptores de la asignatura son: "Estudio de los sistemas lógicos, circuitos combinacionales, secuenciales, aritmética binaria, introducción a los sistemas de microprocesador.

La carga docente de la asignatura es de 6 créditos: Repartidos en 3 de Grupo Grande, correspondientes a 30 horas de clase y otros 3 de seminario y laboratorio, de las cuales 22,5 corresponden a clases prácticas.

Las prácticas de laboratorio son de una duración de 1.5 horas, con lo que nos da un total de 15 sesiones de laboratorio.

En las clases prácticas se ha pretendido, por un lado cubrir los conceptos teóricos, impartidos de manera clásica, mediante puertas, multiplexores, biestables, contadores, ... etc. y por otro hacer una introducción a los lenguajes de descripción hardware utilizando para ello Verilog HDL[1].

II. Elección de la herramienta software

Las premisas de partida para la elección de los programas a utilizar debían cumplir una serie de requisitos, a saber:

- Facilidad de acceso a la herramienta, por parte de los alumnos, con posibilidad de instalación en sus dispositivos personales.
- No requerir excesivos recursos.
- Ser de fácil instalación y utilización.
- Poder instalarla en diferentes sistemas operativos.

En una primera aproximación se pensó en una herramienta con posibilidad de entorno gráfico, ya que al menos en un principio, permite un uso más amigable e intuitivo.

Existen diferentes herramientas de simulación ligadas a fabricantes que, o bien son de libre distribución o disponen de licencias temporales o para estudiantes. Se descartó en principio esta opción ya que en general necesitaban importantes recursos informáticos. Estas herramientas contemplan la posibilidad de síntesis y "place and route" no necesario en esta asignatura, enfocada a la introducción a los sistemas digitales.

Ya se había utilizado para las prácticas, en años anteriores, el entorno de simulación TkGate 1.8 [2], que permite la simulación de circuitos digitales mediante entrada de componentes estándar en forma gráfica. En la versión 2.0, y posteriores, se puede utilizar una descripción Verilog y además, gracias a su distribución freeware, hizo que se considerara como la mejor opción. No obstante, el inconveniente fundamental es la utilización bajo Windows, sobre todo en las últimas versiones, que da bastantes problemas de estabilidad.

Todo esto nos llevó a la utilización del programa Tkgate 2.0 sobre una máquina virtual. De tal forma que los estudiantes pudieran instalar el programa independientemente del hardware y software que dispusiesen.

III. Elección de las prácticas

Para elaborar el conjunto de prácticas a realizar por los alumnos se consideró en primer lugar la carga docente y las materias que componen la asignatura, es decir, la Electrónica Digital básica más la introducción a los microprocesadores. De esta manera la realización de un microprocesador muy sencillo divido en módulos, de tal forma que se pudieran ir diseñando al mismo tiempo que se impartían los conocimientos teóricos, parecía una buena idea. Se optó por una versión del procesador gnome que aparecía en las versiones de XILINX Foundation[3]. Se estructuró por un lado para poder ir diseñándolo a medida que se impartían los diferentes temas y por otro para que mantuviera una cierta coherencia con los bloques funcionales de un procesador. De esta manera se hizo tal y como se puede observar en la Figura 1.

Como consecuencia de todo esto las prácticas a realizar son las siguientes:

- Práctica 1. Introducción a TkGate 2.0 (3h).
- Practica 2. Funciones lógicas (1.5h).
- Práctica 3. Circuitos combinacionales básicos (1.5h).
- Práctica 4. Aritmética binaria (1.5h).
- Práctica 5. Biestables (1.5h).
- Práctica 6. Circuitos secuenciales básicos (1.5h).
- Práctica 7. Máquinas de estado (1.5h).
- Práctica 8. Test del conjunto (3h).
- Práctica 9. Diseño de un Sistema digital (7.5h).

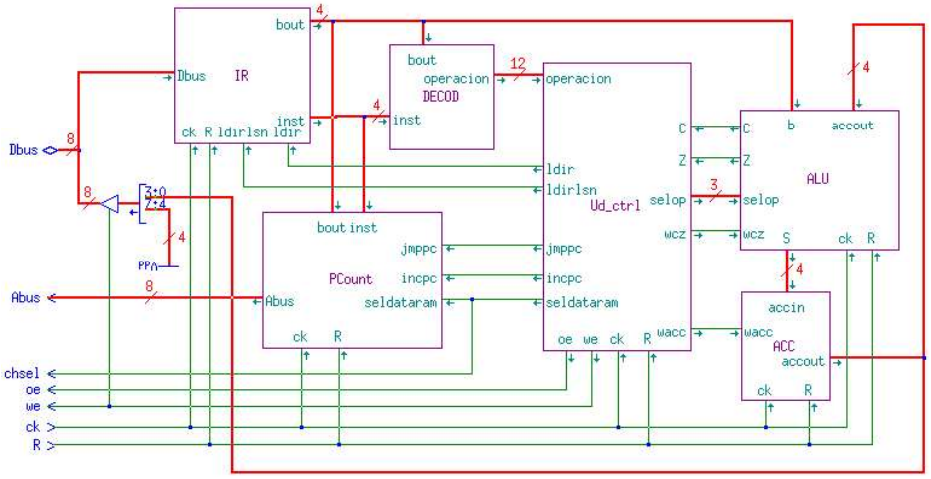


Figura 1. Diagrama general de bloques

En cada práctica el alumno debe realizar el diseño y test del módulo. En las primeras, la tutorización por parte del profesor es alta, disminuyendo a medida que se van realizando prácticas y el alumno maneja con más soltura la herramienta encontrándose más familiarizado con el lenguaje Verilog.

La práctica 1 es un tanto especial, ya que una vez realizada, se volverá a utilizar en la número 8 y, en principio, no contempla la posibilidad de simulación, ya que corresponde tan solo al dibujo de los diferentes bloques (vacios) del Sistema. En ella se hace la primera toma de contacto con el programa, su instalación y uso. Como tutorial se construye el esquema de la Figura 1. Con la creación de este diagrama se ven prácticamente todas las opciones de entrada gráfica del programa, así como, de el uso y organización de ficheros y librerías.

En la práctica 2 se diseña básicamente el circuito combinacional de salida del modulo Ud_ctrl, que a su vez contiene dos módulos internos, como se puede observar en la Figura 2.

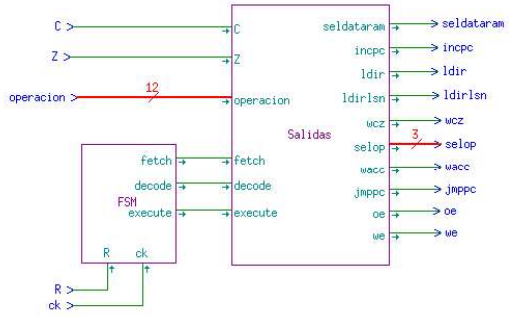


Figura 2. Subesquema de la unidad de control (Ud_ctrl).

La práctica número 3, se corresponde con la realización del módulo DECOD, es decir, con el decodificador de instrucciones.

La práctica 4, corresponde a la realización de la parte combinacional de la ALU.

En la práctica 5 se diseñan los módulos IR, registro de instrucciones, ACC, acumulador, así como la parte correspondiente a los flags de acarreo y cero de la ALU.

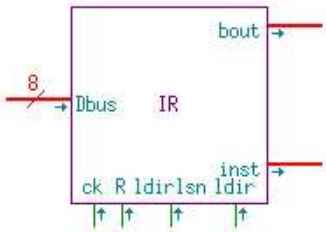
En la práctica 6 se describe el contador de programa.

En la 7 se realiza la parte secuencial del módulo Ud_ctrl, y prueba del módulo completo.

En la práctica 8 se montan todas las descripciones de todos los módulos realizados entre las prácticas 2 y 7 y se monta junto con el modulo de test, como se muestra en la Figura 3. A continuación se muestra como ejemplo la realización de parte de la práctica 5, correspondiente al Registro de instrucciones.

En primer lugar se presenta el módulo y su funcionamiento. El alumno puede optar por realizar un nuevo módulo, la interfaz gráfica, o reutilizar el correspondiente de la práctica 1, esto lo hará dentro de un nuevo archivo con una extensión *.v*, por ejemplo, RI.v

PRACTICA 5. REGISTRO DE INSTRUCCIONES
El módulo obedecerá al siguiente interface:



El funcionamiento será:
Si se produce un reset (R), ambas salidas, bout e inst, toman el valor 0. Si ldir vale 1 entonces un flanco activo de reloj pone DBUS[3:0] en bout y DBUS[7:4] en inst. Si se activa ldirlsn, al llegar el pulso de reloj se copia DBUS[3:0] en bout pero inst permanece invariable.
Téngase en cuenta que ldir y ldirlsn son disjuntas.

El alumno deberá realizar la descripción Verilog del módulo quedando algo parecido a lo siguiente:

```
module IR(Dbus,ck,ldir,ldirlsn,R, bout, inst);
input   [7:0] Dbus;
input   ck ldir,ldirlsn,R;
output  [3:0] bout,inst;

/*bout e inst forman los 8 bits de la
    instruccion completa cuando se lea una
    instruccion deberán modificarse ambos
    registros, cuando se acceda a un dato solo
     bout*/

reg [3:0] bout, inst;
always @(posedge ck or posedge R)
  if (R)
  begin
      bout<=0;
      inst<=0;
  end
  else if (ldir)
    {inst,bout}<= Dbus;
  else if (ldirlsn)
    bout<=Dbus[3:0];
endmodule
```

Una de las ventajas que presenta TkGate es la posibilidad de utilizar, para la simulación, elementos tales como interruptores, diodos, relojes... que permiten al alumno la posibilidad de probar el diseño de forma inmediata y sin preocuparse de tener que realizar el test bench. En la Figura 3 se puede observar la simulación de uno de los módulos siguiendo esta técnica.

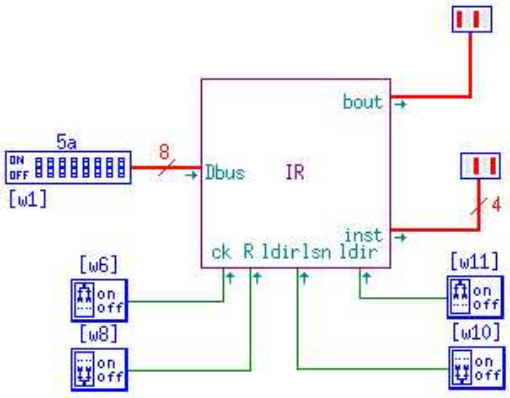


Figura 3. Simulación mediante elementos gráficos.

Evidentemente este método es muy limitado y sólo es de aplicación para diseños simples. No obstante Tkgate permite la utilización de testbenches, pero estos se utilizarán en la simulación del conjunto completo. Incluso permite la presentación de forma temporal de las señales mediante una ventana de simulación como la mostrada en la Figura 4.

Una vez que se ha realizado la práctica se revisa por parte del profesor. El alumno traslada la descripción hardware del

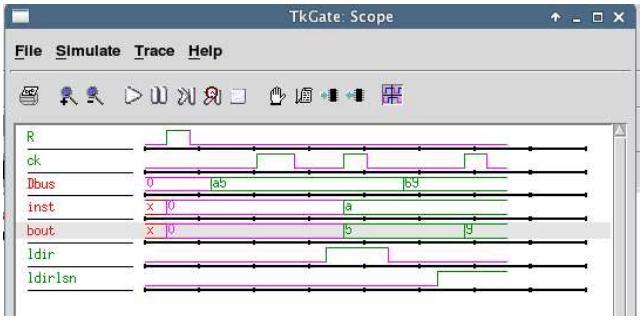


Figura 4. Simulación temporal del módulo IR.

módulo al lugar correspondiente de la primera práctica, el diseño completo del procesador.

Una vez que se han realizado y comprobado todos los módulos se procede a la simulación del procesador en su conjunto, para ello se procede con el diseño de la Figura 5:

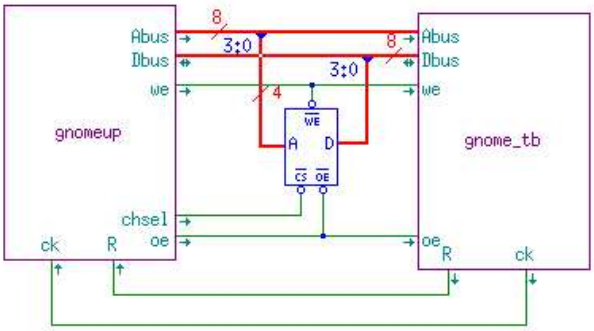


Figura 5. Testbench del procesador.

Como se puede observar además del modulo gnome_tb, correspondiente al tester, se ha incluido un módulo de memoria, suministrado por la herramienta en modo gráfico. Este módulo, que contiene una copia de los registros internos del procesador, permite una fácil visualización de los mismos.

Básicamente el test bench consiste en una memoria ROM que va suministrando las instrucciones a ejecutar por el procesador, junto con las señales de reset y reloj. Estos test de prueba se le suministran a los alumnos ya desarrollados y su labor consiste en comprobar el funcionamiento del procesador. A modo de ejemplo se muestra un pequeño programa que realiza la suma de los datos 0x0A y 0x06 que da como resultado 0x10, es decir 0 y acarreo 1. También realiza la suma del resultado (0) con el dato 0, comprobándose cómo el acarreo producido se suma en el paso siguiente dando como resultado 1.

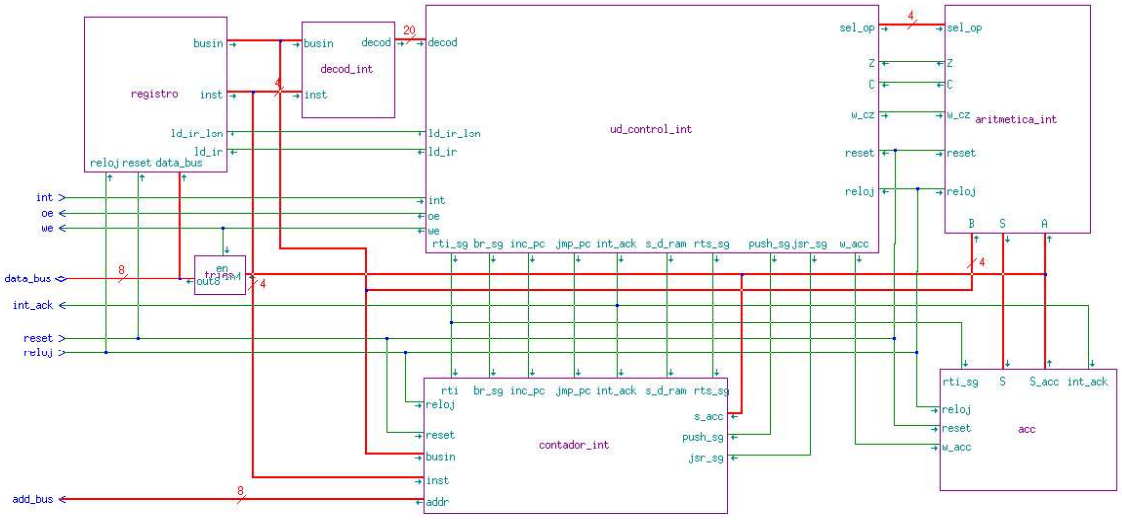


Figura 6. Estructura del gnome, versión mejorada.

```
/* test addrd (suma registros) */
0:  mem_rom={load_d,4'HA};   // carga A en acc
1:  mem_rom={store_rd,4'H2}; // guarda en R2
2:  mem_rom={load_d,4'H5};   // carga 5 en acc
3:  mem_rom={add_rd,4'H2};   // suma con R2
4:  mem_rom={store_rd,4'H5}; // guarda enR5
5:  mem_rom=set_c;           // pone C a 1
6:  mem_rom={load_d,4'H3};   // carga 3 en acc
7:  mem_rom={add_rd,4'H2};   // suma con R2
8:  mem_rom={store_rd,4'H5}; // guarda en R5
```

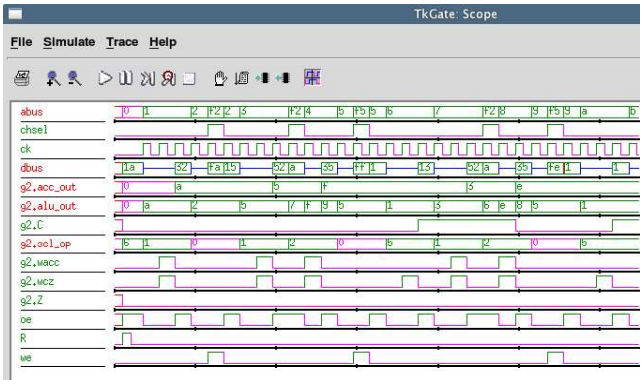


Figura 7. Simulación temporal de suma de registro

Llegado a este punto se han completado la primera parte de las clases prácticas. La segunda consiste en la realización de un trabajo de forma individual y personalizada para cada alumno. Consiste en un sencillo diseño utilizando para ello el procesador que se ha diseñado, o bien, una version "mejorada" que se le suministra al alumno. Las diferencias con el procesador diseñado por él son importantes, aunque la estructura y las instrucciones difieren poco, manteniéndose una compatibilidad ascendente, a modo de resumen las diferencias más importantes son:

- Aumento del número de instrucciones de 12 a 21.
- Instrucciones de salto a subrutina y salto indirecto.
- Instrucciones de rotación.
- Interrupciones.
- Optimización en el número de ciclos de reloj para la ejecución de cada instrucción.

El motivo por el que no se realiza en prácticas esta versión del procesador es que aumenta sensiblemente la complejidad del diseño, y por lo tanto el tiempo de realización del mismo por parte del alumno, con lo que no se conseguiría el fin propuesto.

La Figura 6 muestra el esquema general de bloques de este procesador. Además se suministra un meta-assembler para escribir los programas del procesador y poder, de forma sencilla, insertarlos dentro del Sistema. Esta herramienta, freeware, es el WinTim32.

Todo este material se le facilita al alumno en forma de una máquina virtual Linux, desarrollada por nosotros y denominada "dglab". Esta contiene el entorno de desarrollo integrado Eclipse [4], con un plugin[5] para iVerilog[6] y el visor GTKwave[7]. Alternativamente, puede utilizar este entorno, incluso combinándolo con TkGate, proporcionando unas características más profesionales y que, además, le será muy útil para cursar la asignatura de séptimo semestre denominada Diseño Digital, o realizar el proyecto fin de grado dentro de las materias de la Electrónica Digital.

IV. Realización del proyecto

Un proyecto puede ser el mostrado en la Figura 8. En este caso consiste en la realización de un sistema que envie un dato de 8 bits en formato UART y que lea el dato que se ha enviado, utilizando para ello la interrupción del procesador.

Como se puede observar el sistema contiene:

- 1 Módulo microprocesador (micro_int).
- 1 Módulo de memoria (prog).

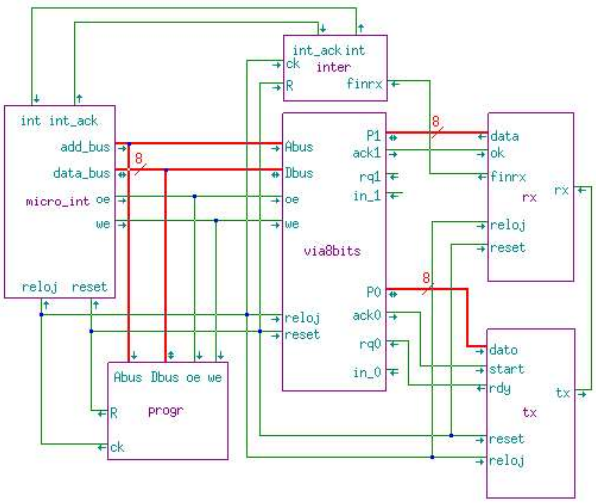


Figura 8. Diagrama de bloques de un proyecto ejemplo.

- 1 Módulo de interfaz entre el procesador y los circuitos auxiliares (via8bits).
- 1 Módulo para generar la interrupción (inter).
- 1 Módulo transmisor (tx).
- 1 Módulo receptor. (rx).

En este caso se suministra al alumno el módulo via8bits, que permite la conexión entre el procesador y la circuitería exterior mediante dos puertos bidireccionales, mas un conjunto de señales de entrada y salida, que pueden programarse de forma sencilla y que facilita el diseño del sistema.

Por otro lado el alumno debe diseñar los módulos correspondientes al receptor y al transmisor serie, así como un pequeño circuito que adapte la señal de fin de recepción (finrx) al formato adecuado para suministrar la interrupción al procesador.

Como se ha comentado el alumno puede optar por realizar el diseño utilizando tan solo la herramienta TkGate, o con Eclipse o con ambas. Veamos la resolución del problema utilizando ambas.

Si se utilizan las dos, para que el diseño sea compatible no se deben tomar de la librería TkGate módulos gráficos, tales como puertas, memorias, sumadores, o dispositivos de entrada salida como LED o SWITCH etc. sino hacer la correspondiente descripción Verilog de los mismos.

En primer lugar hay que crear un Proyecto en Eclipse del tipo Verilog/VHDL e importar el archivo creado con TkGate, este archivo contiene una gran cantidad de información relativa a la parte gráfica, no presenta ningún inconveniente, ya que se presenta en forma de comentarios y por lo tanto Verilog lo ignorará. A continuación se muestra un trozo del código de estas características.

```
//: /hdlBegin Salidas
//: interface  /sz:(40, 40) /bd:[ ] /pd: 0 /pi
    : 0 /pe: 0 /pp: 1
//: enddecls
```

Una vez importado tan solo queda la construcción del proyecto. Una vez hecho esto, debe aparecer en el árbol del proyecto algo parecido a lo mostrado en la Figura 9.

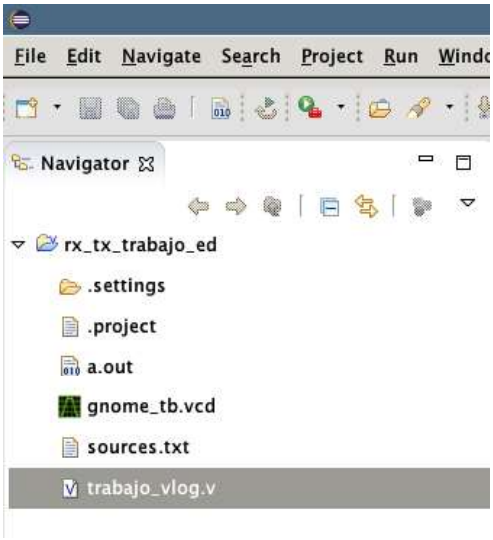


Figura 9. Árbol del proyecto en Eclipse.

Cabe destacar el archivo gnome_tb.vcd, que sirve de Fuente para el visor GTKWave. Este archivo muestra la evolución de las diferentes señales que hayan sido seleccionadas en el test_bench.

En la Figura 10, se ven el dato que se envía, su transmisión y el dato recibido.

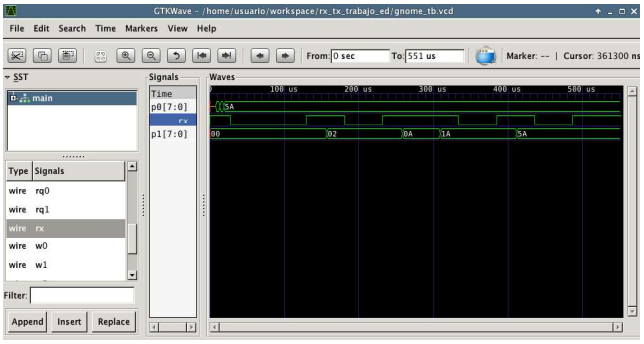


Figura 10. Simulación con Icarus Verilog y GTKWave.

También se puede, evidentemente, ver señales con más detalle. En la Figura 11 se muestran el Bus de Datos, el Bus de Direcciones, las señales de write enable (we) y ouput enable (oe) junto con los valores del acumulador.

En Figura 12 se muestra cómo el sistema se encuentra "ocioso" a la espera de que se produzca la interrupción y cómo una vez producida comienza la ejecución de la subrutina de interrupción.

Una vez finalizado el trabajo, se mostrará al profesor el diseño realizado para su evaluación. Presentará , además del trabajo, las simulaciones y los archivos verilog correspondientes a las prácticas de laboratorio.

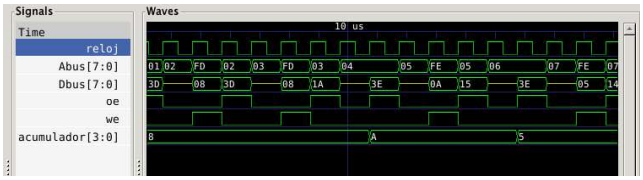


Figura 11. Detalle de simulación.

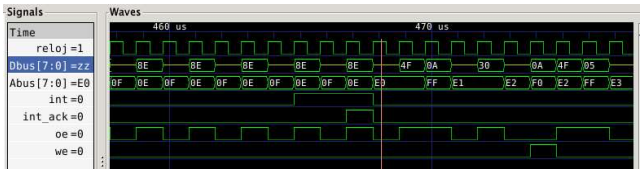


Figura 12. Proceso de salto a la subrutina de interrupción

V. Encuesta a los alumnos

Se ha llevado a cabo una encuesta para conocer las opiniones de los alumnos que han terminado el trabajo respecto a la metodología de las prácticas. Los resultados obtenidos se pueden observar en la Tabla 1.

Cuadro I
Resultados de encuesta a los alumnos.

Preguntas / Respuestas	***Media***	***Desviación estándar***
Las prácticas se adecuan al programa de la asignatura.	9.00	1.41
Se deberían realizar más sesiones teóricas sobre Verilog (a costa de horas de laboratorio).	5.25	3.28
Trabajar con una máquina virtual facilita la compatibilidad de los ordenadores.	6.38	3.11
El esfuerzo que hay que realizar en el trabajo y las prácticas es muy superior al 30 % de la carga de la asignatura.	7.25	2.49
El trabajo aporta conocimientos sobre la asignatura.	9.50	1.07
La labor tutorial del profesor ha sido adecuada.	9.75	0.71

Además los alumnos han reportado otros tipos de sugerencias, entre las que cabe destacar:

- Incluir una práctica en la que se ejemplifique el trabajo de diseño que deberán abordar.
- Establecer los requisitos del trabajo antes de finalizar las prácticas, para disponer de mayor tiempo para la realización del mismo.
- Desarrollar con mayor detalle la introducción teórica a Verilog.

VI. Resultados y problemas encontrados

Si bien la primera parte, realización del procesador, de las prácticas no conlleva para el alumno una carga excesiva de trabajo, la segunda parte sí. La realización del Proyecto, involucra una elevada carga de horas al alumno (a cargo de las horas no presenciales), así como una carga tutorizante importante por parte del profesor. La aplicación de este método, en principio, estaría restringida a grupos en los que el ratio alumno profesor no sea muy elevado.

Se ha detectado por parte de algunos alumnos dificultades a la hora de realizar la descripción hardware de los circuitos, debido al funcionamiento paralelo del hardware y en otros casos como consecuencia de no haber profundizado lo suficiente en los conceptos teóricos o por apoyarse más de lo conveniente en sus compañeros. Por lo tanto, a la hora de enfrentarse a un problema diferente al de los demás, le supone un gran esfuerzo. Consideramos, no obstante, que el alumno, una vez terminado el trabajo, alcanza un conocimiento de la asignatura satisfactorio. Debido a la labor que supone la parte práctica de la asignatura y a los resultados obtenidos, el peso de este bloque es importante y corresponde al 40 % de la evaluación final[9].

Referencias

[1] IEEE Standard for Verilog Hardware Description Language,ïn IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001) , vol., no., pp.0_1-560, 2006 doi: 10.1109/IEEESTD.2006.99495 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1620780&isnumber=33945
[2] *TkGate*, http://www.tkgate.org
[3] Xilinx, *Foundation Series*. http://www.xilinx.com/support/sw_manuals/2_1i/download/qsprint.pdf
[4] Laboratorio de Electrónica Digital desarrollado por los autores del trabajo. http://digital.unex.es/files
[5] Eclipse Foundation, entorno de desarrollo integrado *Eclipse*, https://eclipse.org
[6] Plugin *VEditor* para la integración de lenguajes HDL en Eclipse. http://sourceforge.net/projects/veditor/
[7] *Icarus Verilog*, http://iverilog.icarus.com
[8] Visor de Formas de onda *GTKWave* http://gtkwave.sourceforge.net
[9] Plan de estudios de la asignatura Electrónica Digital de la UEX, https://docs.google.com/spreadsheets/d/15gLOmWo9blJSnyBSUFBsapazAiTZrlflGUOMbf0lnVQ/edit?usp=drive_web