ENTORNO DE SIMULACIÓN ELECTRÓNICA DIGITAL PERSONALIZADO E INTUITIVO MEDIANTE PROTOTIPO VIRTUAL

J. A. Cebrián[1], F. J. de Andrés Rodríguez-Trelles[2] y J. L. Villafañe Landeira[3]

[1]Universidad de Valladolid. jesceb@eis.uva.es
[2]Universidad de Valladolid. fratre@eis.uva.es
[3]Universidad de Valladolid. jose2lorenzo@yahoo.com

RESUMEN

Se presenta la descripción de una herramienta para diseño electrónico digital con lenguajes de descripción de hardware. Esta herramienta hace uso de lo que vamos a llamar un prototipo virtual para realizar una evaluación gráfica intuitiva de los resultados de simulación. Se comentan herramientas existentes que utilizan el lenguaje Tcl/Tk para desarrollar esa evaluación intuitiva. Se propone como alternativa, para diseñar el prototipo virtual, utilizar herramientas de desarrollo de lenguajes compilados como Qt Designer para C++.

1. INTRODUCCIÓN

Es habitual en otras ramas de la ingeniería, como en la ingeniería mecánica, utilizar simulaciones gráficas de apariencia real que permitan evaluar de una manera intuitiva el buen funcionamiento del diseño. ¿Por qué no utilizar estas mismas técnicas en diseños electrónicos? Es cierto que muchos de los diseños electrónicos no se prestan a esta evaluación intuitiva, sin embargo sí que existen algunos y desde el punto de vista didáctico sería posible escoger precisamente estos. Algunos ejemplos son un control electrónico de un ascensor, de un cruce de semáforos, una calculadora, o el encendido y apagado de un motor.

El procedimiento habitual de evaluación del funcionamiento de un diseño electrónico es mediante el uso de cronogramas de los resultados de una simulación. Lo que se pretende es utilizar esos mismos resultados de simulación pero visulizándolos de una manera más intuitiva: un modelo gráfico en la pantalla de manera que tenga la apariencia de la interfase del producto que se está desarrollando. A este modelo gráfico le vamos a llamar "prototipo virtual". La simulación del prototipo virtual debe seguir en su comportamiento los resultados de la simulación.

Se encuentran algunas propuestas en herramientas comerciales como ModelSim [4] y BlueHDL [1]. Estas herramientas utilizan un lenguaje interpretado Tcl/Tk [5] para diseñar su interfase de usuario gráfica y dejan abierta la posibilidad para que el usuario pueda personalizar al menos algunos elementos de la interfase para realizar simulaciones con apariencia real. Esta personalización se hace programando en ese lenguaje Tcl/Tk, que cuenta con recursos de pantalla apropiados para ello. La gran ventaja es la posibilidad de una simulación interactiva, debido a que proporcionan una interfase con las herramientas de simulación utilizadas.

También se hace referencia al uso de Tcl/Tk orientado a herramientas de diseño electrónico en uno de los cursos de formación de Doulos [2]. Nombra varias herramientas comerciales que lo utilizan para su entorno gráfico de usuario: ModelSim, Leonardo Spectrum, Synopsys Design & FPGA Compiler, Synplify Pro and Altera Quartus. No

comenta expresamente la posibilidad de desarrollar un entorno personalizado para simulación intuitiva como en las dos referencias anteriores pero sí habla, por ejemplo, de la creación de interfases de usuario personalizadas.

En este artículo se propone como alternativa el uso de un lenguaje de alto nivel compilado, como C++, para superar las limitaciones propias de Tcl/Tk como el hecho de ser un lenguaje más lento por ser interpretado.

Nos vamos a centrar en diseños digitales basado en lenguajes de descripción de hardware (hardware description language, HDL), en particular VHDL (VHSIC HDL). Sin embargo las ideas presentadas serán válidas para otros lenguajes, como Verilog, cuando estos y las herramientas que los utilicen tengan características similares a las que tienen los simuladores de VHDL. También es perfectamente válido si el diseño digital se hace mediante captura de esquemas.

En el resto de este artículo se supone que el objetivo es que el alumno aprenda a diseñar sistemas electrónicos digitales, aunque también se puede proponer que puedan aprender la metodología completa de diseño utilizando herramientas de depuración realistas utilizando un prototipo virtual. En el apartado 2 se comentan los pasos necesarios para realizar una aplicación genérica. En el apartado 3 se describe una aplicación concreta, se trata de un entorno integrado que pretende facilitar el aprendizaje del diseño electrónico digital basado en lenguajes de descripción de hardware. Finalmente en el apartado 4 se comentan las conclusiones de este trabajo.

2. DESARROLLO DE UNA APLICACIÓN

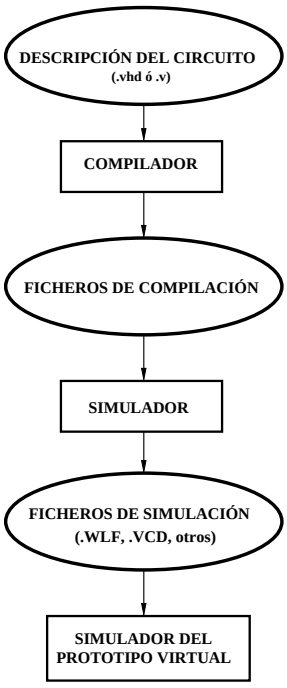

Figura 1: Flujo de datos para simular un diseño digital.

Vamos a partir de un diseño en el HDL seleccionado, que tenga ya, al menos, una corrección sintáctica. Este diseño se compila, y se simula. El simulador proporciona un fichero de resultados que puede utilizar la herramienta de visualización de cronogramas. La idea es utilizar ese mismo fichero de resultados para ver los resultados de una manera más realista, utilizando el simulador del prototipo virtual. Este flujo de datos se esquematiza en la figura 1.

Los pasos más destacados que hay que realizar son los siguientes:

1. Diseñar en el HDL el modelo del producto que se quiere desarrollar.
2. Diseñar el modelo de interfase gráfico: el prototipo virtual.
3. Definir las señales necesarias para conectar el modelo gráfico con la simulación digital.
4. Establecer las conexiones entre el prototipo virtual y los ficheros de simulación.

Desde el punto de vista del aprendizaje de diseño digital, un alumno debería seguir solamente los pasos 1 y 3. El prototipo virtual ya debería estar diseñado y el paso 4 debería hacerlo un programa, diseñado también a partir de la

información suministrada en el paso 3, del fichero donde están los resultados y de la información incorporada en el prototipo virtual.

El diseño del prototipo virtual, que se puede ver facilitado por herramientas de diseño adecuadas, va a depender de la herramienta de programación en alto nivel elegida. Se trata de diseñar un modelo gráfico que tenga una apariencia suficientemente cercana al dispositivo físico que se pretende simular y diseñar y, sobre todo, que permita comprobar fácilmente su funcionalidad en el momento de la simulación. Es necesario incorporarle el programa relativo al apartado 4, un programa que vaya leyendo los resultados de la simulación del fichero. La animación del modelo gráfico estará controlada por la lectura de ese fichero de simulación, de manera que, por ejemplo, puedan apagarse y encenderse iconos o desplazarse dibujándolos en otro punto de la pantalla. Otras posibilidades, dentro de la animación gráfica o como complemento, son mostrar mensajes en pantalla con el valor de ciertos resultados, mostrar datos estadísticos y representar funciones gráficas de los resultados directamente o aplicando alguna ecuación para obtener un valor de interés. En la figura 2 se esquematiza la obtención del programa de simulación del prototipo virtual. Aquí se supone que las librerías precompiladas incluyen las herramientas necesarias para la lectura del fichero de resultados de simulación.

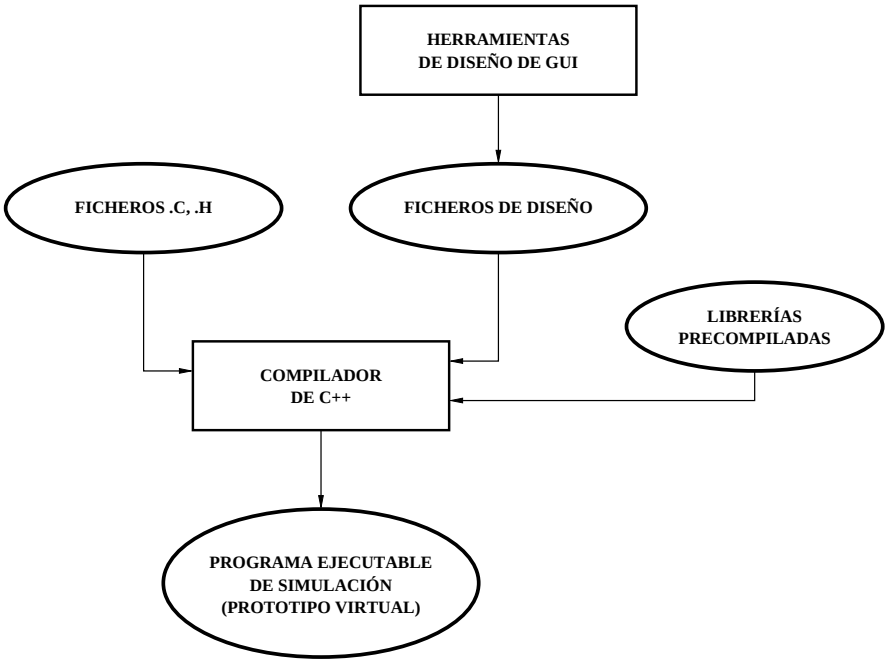

Figure 2: Diseño general del prototipo virtual.

Dependiendo del HDL seleccionado y de la herramienta que lo soporte, podemos encontrarnos distintas alternativas para generar el fichero de resultados de simulación:

- El propio HDL incluye instrucciones para generar ficheros.
- El formato estándar VCD (value Change Dump-IEEE Std 1364) desde la herramienta.
- El formato propio de fichero utilizado por la herramienta.

La primera alternativa es la más genérica y más portable. El simulador del HDL debe poder generar ficheros, con los valores de las señales de interés o sus transiciones. En particular, si los ficheros generados son de tipo texto, esta opción será independiente tanto de la herramienta de simulación como de la plataforma donde corra el programa. Por un lado no todas las herramientas permiten generar ficheros. Además tiene el problema de que a la hora de diseñar el modelo hay que incluir instrucciones extra para escribir en los ficheros de

resultados. Si además se quiere cambiar el conjunto de señales y/o variables de interés habrá que modificar el diseño perdiendo por tanto en flexibilidad. Esto es además, desde el punto de vista didáctico, un contratiempo si nos queremos centrar en la enseñanza del diseño electrónico.

Hay muchas herramientas que pueden proporcionar ficheros de resultados en formato VCD. Al ser estándar es una ventaja. Las herramientas permiten seleccionar señales y variables cuyas transiciones, junto con el instante de tiempo en que se producen se guardarán en el fichero. No es necesario modificar los ficheros para seguir la traza de una nueva señal. Tampoco es necesario a la hora de modelar, tener en cuenta qué señales y variables se registrarán en el fichero. Se tiene por lo tanto una mayor flexibilidad.

Un problema con los ficheros VCD es su excesivo tamaño y lentitud en su procesamiento posterior. Estos ficheros son de formato texto y no todas las herramientas los comprimen y los manejan así comprimidos (por ejemplo ModelSim no lo hace). Si el número de señales que se necesitan guardar y/o el tiempo de simulación es muy grande se obtienen ficheros excesivamente grandes. Además para el caso de utilizar VHDL se puede encontrar el problema de que no todos los tipos de datos se pueden guardar en un fichero VCD directamente, sin convertirlos previamente, por ejemplo, todos los tipos de datos de números enteros o reales.

Hay herramientas que tienen un formato de fichero de resultado de simulación propio, en general la herramienta está optimizada para ese formato. En el caso de ModelSim se tiene el fichero de formato WLF (Wave Log Format). Este fichero además de estar comprimido permite guardar todos los tipos de datos que se utilizan en VHDL. Un problema es que el formato de estos ficheros no está documentado, sin embargo ModelSim proporciona una librería de funciones para acceder a los ficheros en sólo lectura [3]. La aplicación realizada en lenguaje de alto nivel puede a través de esta librería acceder a los valores de las distintas señales y variables que se grabaron durante la simulación.

3. EL ENTORNO DE DESARROLLO INTEGRADO QMWSIM

Como aplicación de lo expuesto en el apartado anterior se ha realizado en el Departamento de Tecnología Electrónica de la Universidad de Valladolid un Entorno de Desarrollo Integrado (IDE) para diseño electrónico digital descrito en los lenguajes VHDL y Verilog, poniendo especial énfasis en el tratamiento y visualización gráfica de los resultados de simulación [9].

Este IDE gráfico, que se le ha denominado QMWSim (Qt Multi Windows Simulation), además de ser de un uso sencillo, debía cumplir las siguientes funciones:

- Tratamiento sobre ficheros de texto que contienen código VHDL. Es decir, edición, compilación y simulación.
- Creación y ejecución de macros.
- Visualización de resultados de simulación mediante cronogramas, representación gráfica de funciones y especialmente la animación gráfica del prototipo virtual.

La primera de las funcionalidades es la típica de un IDE incluyendo el resaltado sintáctico, equilibrado de paréntesis e indentación automática y posibilidad de múltiples documentos. El acceso a compilación y simulación es a través de llamadas a los programas externos accesibles desde línea de órdenes.

Se ha incorporado un lenguaje de macros interno, para ello se ha empotrado el interprete de Tcl en la aplicación, encapsulando la biblioteca Tcl_api en la clase TclQt. Se ha conseguido la posibilidad de construir un lenguaje interno propio útil para automatizar tareas

repetitivas o tediosas tanto en tareas genéricas de edición como en las más específicas de compilación, simulación o visualización de resultados.

QMWSim se ha desarrollado íntegramente en C++. Esto favorece la reutilización del código en futuros desarrollos o ampliaciones. Para su diseño se ha seguido el criterio de elegir una herramienta libre y portable, siendo la biblioteca de clases Qt de la empresa Troll Tech [6,7] la que cumplía mejor estos criterios. Además la herramienta Qt Designer [8] incluida facilita mucho el desarrollo del prototipo virtual.

Las funciones de la biblioteca wlf_api se han encapsulado, combinándolas con la característica más sobresaliente de Qt, las "signals-slot", para formar la clase que se ha denominado "dumplog". El uso de la clase "dumplog" ha quedado facilitado de tal modo que basta con seguir los siguientes pasos:

1. Indicar el fichero de resultados y las señales sobre las que se trabaja.
2. Registrar la función a ejecutar cuando se produzca una transición en el valor de la señal.
3. Seleccionar el intervalo de tiempo en el que se quiere averiguar el valor de las señales.

Para ilustrar el procedimiento se comenta a continuación un ejemplo sencillo.

3.1. Modelo virtual de un convertidor BCD a siete segmentos

El ejemplo elegido es muy simple, un convertidor BCD con salida a siete segmentos, tal como muestra su simulación en la figura 3. El diseño visual consiste simplemente en 7 imágenes (los segmentos) generadas mediante el procedimiento de copiar y pegar, que se pueden encender o apagar.

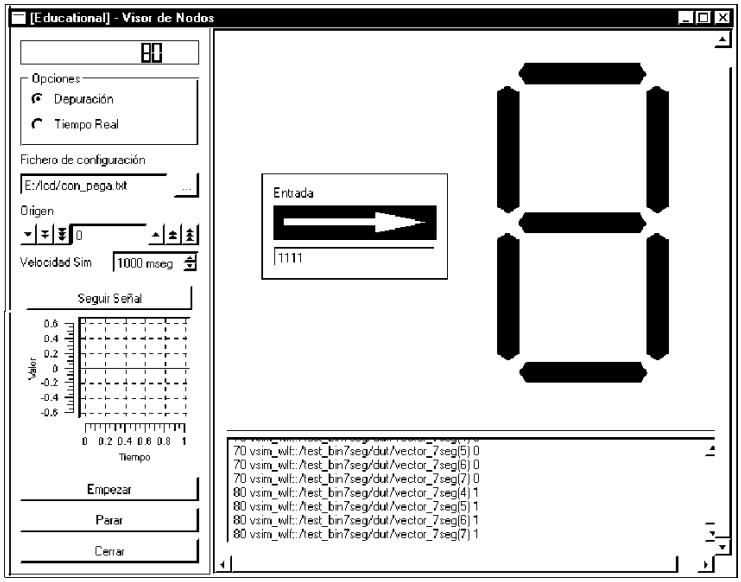

Figure 3: Prototipo virtual de un convertidor BCD a siete segmentos.

Para obtener el prototipo virtual, el único código nuevo que se necesita escribir es el que se comenta a continuación: Cuando se crea un objeto de la clase `sieteSegmentos`, se guardan los segmentos en un vector para acceder a ellos de forma ordenada.

```
void sieteSegmentos::init() {
arrayPixmaps=new QLabel*[7];
arrayPixmaps[0]=a;
...
arrayPixmaps[6]=g;
hideAll();
```

```
}
```

Inicialmente se apagan todos los elementos del LED:

```
void sieteSegmentos::hideAll() {
for (int i=0;i<NUMSEG;i++) arrayPixmaps[i]->hide();
}
```

Para mostrar un número determinado es necesario encender o apagar los segmentos correspondientes de acuerdo con la señal recibida, esto se hace llamando a la función `segmento`:

```
void sieteSegmentos::segmento(result myResultado)
{
QString signal;
for(int i=1;i<=7;i++)
{
  signal.sprintf("vsim_wlf::/test_bin7seg/dut/vector_7seg(%d)",i);
  if(signal.compare(myResult.name)==0) {
  if(myResult.value=="0") arrayPixmaps[i]->show(i-1);
  else arrayPixmaps[i]->hide(i-1);
}
}
```

Donde `result` es una estructura con los campos tiempo, nombre y valor de la señal procedente del fichero `.wlf` de resultados de simulación, estos valores se muestran además en una ventana de texto.

Esta sería la programación necesaria para el funcionamiento del prototipo virtual `sieteSegmentos`. Para conectarlo con el fichero de resultados de simulación, se utilizan las funciones de la clase dumplog. La líneas que es necesario añadir son las siguientes:

```
...
dumplog* myDumplog=new dumplog();
myDumplog->setData(miFichero,"/dut","");
connect(myDumplog,SIGNAL(signalStruct(result)),sieteSegmentos,SLOT(segmento(result)));
myDumplog->dispara();
...
```

Lo que hemos hecho es crear un objeto capaz de leer los resultados de simulación guardados en el fichero miFichero, seleccionando únicamente aquellas señales que contienen la cadena "`/dut`".

La línea siguiente es la más importante:

```
connect(myDumplog,SIGNAL(signalStruct(result)),sieteSegmentos,SLOT(segmento(result)));
```

En ella utilizamos una de las características más importantes de Qt, el mecanismo "signal-slot". Estamos diciendo que cuando el elemento dumplog al leer los resultados de simulación encuentre una transición en alguna de las señales de simulación que contiene "/dut", llame a la función `segmentos(result)` del elemento antes diseñado sieteSegmentos. Y la llamamos con toda la información necesaria para representar su estado.

El código en VHDL que el alumno escribiría tendría de una manera resumida las siguientes especificaciones: La señal de entrada al diseño en VHDL será un número entero entre 0 y 15 y la salida será un vector de 7 bits correspondientes a cada uno de los segmentos que se activarán de acuerdo con el número decimal de la entrada o bien con la letra E de error si el código decimal es mayor que 9.

En realidad el prototipo virtual de 7 segmentos podría servir para evaluar otros diseños digitales, con muy pocas modificaciones.

4. CONCLUSIONES

Se ha diseñado una herramienta que facilita tanto el diseño digital como su aprendizaje. En general el uso del prototipo virtual, y en particular esta herramienta, mejora la motivación del alumno en el diseño digital.

Se pretende en el futuro mejorar la herramienta para conseguir verdadera interactividad. Esto por otro lado estará más sujeto al tipo de HDL y herramienta que se utilice.

Finalmente queremos destacar que esta idea de evaluación intuitiva mediante prototipo virtual se puede extender a otro tipo de diseños ya sean digitales, basados en microprocesador, analógicos o mixtos. Es sin embargo necesario que, de alguna manera, la herramienta original de simulación haga los resultados accesibles a otros programas, en particular al programa de simulación del prototipo virtual.

5. BIBLIOGRAFÍA

[1] Blue Pacific Computing. *BlueHDL User's Manual*. www.bluepc.com, September 2001.

[2] Doulos Ltd. "Essential Tcl/Tk". World Wide Web, www.doulos.com, 2003.

[3] Model Technology. "Model technology technical note. using the WLF API", 1999.

[4] Model Technology. *ModelSim SE Tutorial Version 5.5d*, August 2001.

[5] Tcl Developer Xchange, World Wide Web, www.tcl.tk, 2004.

[6] Trolltech. *Programming with Qt*, www.troltech.com, 2001.

[7] Trolltech. "Qt 3.0 whitepaper", www.troltech.com, 2001.

[8] Trolltech. *Qt Designer Manual*, www.troltech.com, 2001.

[9] J. L. Villafañe, *Entorno gráfico de programación con aplicación a diseños VHDL,* Proyecto Fin de Carrera, Universidad de Valladolid, 2003.