CAD PARA DISEÑO A NIVEL RT DE UNIDADES DE CONTROL

C. Baena[1], M.P. Parra[1], L.M. López y M. Valencia[1]
1 Departamento de Tecnología Electrónica
Universidad de Sevilla, 41012 Sevilla
Tfno: 95-4552785, Fax: 95-4552764, e-mail: baena@cnm.us.es
1 también, Dpto. Diseño Analógico (IMSE-CNM)

RESUMEN.- En este trabajo presentamos parte de una herramienta para CAD de sistemas digitales a nivel RT. La disponibilidad de una herramienta de este tipo es de gran interés docente. Con ella se automatiza el paso desde la descripción mediante un lenguaje de descripción de hardware (HDL) muy simple hasta la obtención de un diseño del circuito de control a nivel de biestables, garantizándose que el programa de descripción está libre de errores léxicos, sintácticos y semánticos.

1.- INTRODUCCIÓN

El diseño de sistemas digitales a nivel RT posee dos puntos de extremado interés, uno de carácter profesional y otro docente. En primer lugar, desde la perspectiva del diseño digital en general, porque en él se realiza la transición entre los niveles más abstractos (comportamiento como sistema y su arquitectura) y el nivel más concreto (de conmutación y su realización con biestables y puertas lógicas). La enorme complejidad que alcanzan los sistemas digitales hace que, para ser eficaz, el diseño deba realizarse según distintos niveles de abstracción [1, 2]. El diseñador procede entonces a recorrerlos según una metodología, comúnmente desde arriba hacia abajo (Top-Down) [3, 4]. Cada paso desde un nivel a otro se facilita con el uso de las herramientas CAD adecuadas. De aquí que, la realización de un programa que contribuya a automatizar las tareas de diseño a nivel RT es, pues, un objetivo "per se". Además, los procesos de diseño en formas de implementación de sistemas digitales (desde los integrados "full-custom" hasta los realizados con FPGA's) utilizan todos el nivel RT.

En segundo lugar, desde la perspectiva docente, el nivel RT tiende un adecuado puente entre las materias de circuitos digitales y el hardware de los ordenadores. Además, especialmente para los titulados en Informática, la descripción de sistemas digitales a nivel RT realiza una conexión muy adecuada entre los circuitos y los lenguajes de programación.

En esta comunicación se pretende presentar una herramienta que automatiza en parte el diseño de sistemas digitales a nivel RT manteniendo un enfoque eminentemente docente y práctico centrado en la asignatura que impartimos. Con ello continuamos la línea de trabajo iniciada en cursos académicos anteriores [5], con la que se pretende complementar los aspectos teórico-prácticos de la asignatura con el uso de herramientas de software. Este artículo se ha organizado como sigue. Tras esta Introducción, en el apartado 2 se especifica el lenguaje HDL sobre el que se describe el sistema digital. La herramienta CAD desarrollada se presenta en el apartado 3, tratándose sucesivamente cómo entrar en ella, el chequeo que realiza sobre la descripción HDL, el circuito de control que se obtiene y las líneas de ampliación previstas. Por último se exponen las principales conclusiones.

2.- ESPECIFICACIONES. DESCRIPCIÓN DEL LENGUAJE HDL.

La herramienta a desarrollar debe ser útil a nuestros alumnos. De aquí que sus especificaciones proceden de la propia programación de nuestra asignatura. En particular, el diseño de sistemas digitales a nivel RT es presentado como una tarea para realizar según una metodología de actuación. El sistema es dividido en dos unidades, de procesado y de control, cuyo tratamiento pretende ser suficientemente riguroso y formal.
Para la descripción formal del hardware, la representación concreta que utilizamos es un lenguaje tipo HDL (Hardware Description Language). Un lenguaje HDL es parecido a un lenguaje de programación de alto nivel y es, por tanto, muy útil en la realización de sistemas con ayuda del ordenador. En este trabajo utilizamos un lenguaje HDL muy simple, basado en la notación de transferencias entre registros y orientado principalmente a su utilización docente.
El formato general de una instrucción en el lenguaje HDL es el siguiente:

N	$f_0(x)$	T_0/z_0	N_0
	$f_1(x)$	T_1/z_1	N_1
		.....	
	$f_{n-1}(x)$	T_{n-1}/z_{n-1}	N_{n-1}

Cada instrucción se desarrolla en un ciclo de reloj, por lo que representa una micro-operación del sistema. Esta micro-operación ejecuta una o más transferencias entre registros en ese ciclo de reloj.
El primer campo que aparece en una instrucción HDL es N, que corresponde al número de la instrucción en la que estamos. Dentro de cada instrucción el segundo campo corresponde a una serie de funciones combinacionales de las entradas de control del sistema. En cada una de ellas se pregunta si la determinada función lógica $f_i(x)$, donde x toma el papel de señales de controles verdadera o no. Si resulta verdadera, se llevará a cabo la transferencia entre registro T_i que viene a continuación dentro del tercer campo. También existe la alternativa de, en vez de expresar la tranferencia entre registro, listar las señales de control de los dispositivos necesarios para llevar a cabo dicha transferencia . En nuestro lenguaje HDL, esta acción consiste en activar en alto las señales de control z_i indicadas. Por otro lado, en cada instrucción hay que garantizar que el conjunto de funciones $f_i(x)$ sea completo. Es decir, para cada combinación de entradas de control que tenga el sistema sólo una de las $f_i(x)$ que compongan la instrucción se hará verdadera. Así quedan determinadas unívocamente las acciones a tomar. Por otra parte, también se exige que una de las condiciones $f_i(x)$ sea correcta. Esto es, se cumplirá

$$\text{OR}\,(f_0, f_1, \ldots, f_{n-1}) = 1$$
$$\text{AND}\,(f_i, f_j) = 0 \qquad \forall\, i \neq j$$

Por último, el campo final en cada línea es el número de la siguiente instrucción que se ejecutará en el próximo ciclo de reloj.
Descrito ya el formato general de nuestro lenguaje HDL, pasamos a comentar algunas simplificaciones que pueden realizarse sobre dicho formato. En particular, a veces hay campos que pueden eliminarse, como son:
1. Si se cumple que la próxima dirección $N_i = N + 1$, este campo puede desaperecer. Por ej., si

N	$f_0(x)$	T_0/z_0	
	$f_1(x)$	T_1/z_1	N_1

se entendería que la próxima instrucción que se realizará cuando $f_0(x)$ sea verdadera sería N + 1, sin necesidad de indicarlo.
2. Si funciones $f_i(x)$ resultan siempre verdaderas, es decir, determinadas transferencias entre registros son incondicionales, se puede rellenar ese campo por la letra "t" (true), por ej.:

N	t	T_0/z_0	N_0

3. Si para deteminadas condiciones dentro de una instrucción no hay que realizar ninguna transferencia, es decir, no hay que activar ninguna señal de control, el tercer campo de la instrucción puede quedar vacío o rellenarlo con la palabra "NOP" (No Operación). Por ej,

N	$f_0(x)$	-	N_0
	ó		
N	$f_0(x)$	NOP	N_0

3.- EL PROGRAMA DE AYUDA AL DISEÑO

Este trabajo tiene como objetivo acometer el desarrollo de una herramienta CAD sobre PC para diseño de unidades de control cuya operación funcional esté bien establecida. Entre otras tareas, el programa a desarrollar deberá perfilar las herramientas de descripción a nivel RT, incluir cierto grado de corrección y generar diseños a nivel de conmutación. En este apartado presentamos las principales características del programa desarrollado.
La herramienta que proponemos, en último término deberá obtener, para un sistema digital dado, el esquema de su unidad de control a partir de su descripción formal. Para ello se parte de una descripción en lenguaje HDL del comportamiento del sistema digital y, tras comprobar la existencia de posibles errores formales, se llegará al diseño final. En la Figura 1 se muestra un esquema con las etapas que atravesará el programa y que sarán descritas a continuación en más detalle.

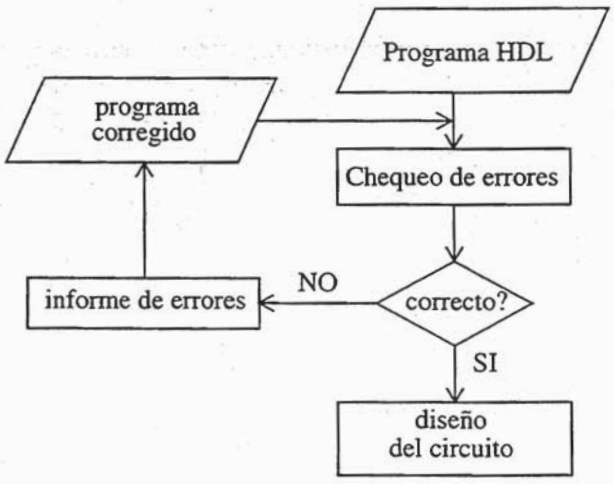

Figura 1.- Diagrama de bloques del programa

3.1.- Entrada al programa

El fichero de entrada al programa está estructurado en tres partes. La primera es la cabecera, en donde se especifica en nombre del programa. La segunda parte ocupa la declaración de

señales que podrán ser variables binarias, palabras o buses y funciones combinacionales, y que se usarán posteriormente. La última parte es el cuerpo del programa en sí, donde se describe el fichero en HDL.
El lenguaje HDL que admitirá nuestro programa será de las características descritas en el apartado anterior. Los operadores lógicos que se pueden utilizar a la hora de describir las funciones combinacionales $f_i(x)$ serán los usuales en la lógica digital (AND, OR, NOT, NAND, NOR y XOR). Asímismo, se permiten las simplificaciones antes comentadas de omisión de determinados campos dentro de una instrucción.

3.2.- Verificación del lenguaje HDL

Una vez capturado el fichero de entrada, el sistema inicia una fase de chequeo de posibles errores en la descripción de entrada generando en su caso un informe de los que hayan sido encontrados.
Los tres tipos de errores que detecta el programa son a nivel léxico, sintáctico y semántico.
El análisis léxico se encarga de un chequeo de lo sustancial del fichero HDL, es decir, actuará como un filtro eliminando todo aquello que no sea de interés para la obtención final de la unidad de datos a desarrollar, como son los comentarios y caracteres que no pertenezcan al lenguaje.
La salida de esta etapa actúa como entrada para el siguiente nivel de verificación: el análisis sintáctico. Éste se encargará de comprobar que el fichero HDL cumple las normas generales de su formato. Errores típicos en este nivel podrían ser por ejemplo una mala expresión de las funciones combinacionales $f_i(x)$ por haber utilizado operadores lógicos distintos a los que fueron declarados al inicio y no serán reconocidos por el sistema. Cuando al verificar el fichero de entrada el programa encuentra un error sintáctico, devolverá un informe del error indicando la posición en que se encuentra y dando una breve descripción del mismo. A modo de ejemplo, en la Figura 2 se muestra la respuesta que daría el programa ante dos instrucciones en las que, por una parte, el campo de funciones combinacionales aparece sin operadores entre las variables x e y, por otra parte, aparece un número de instrucción no válido, etc.:

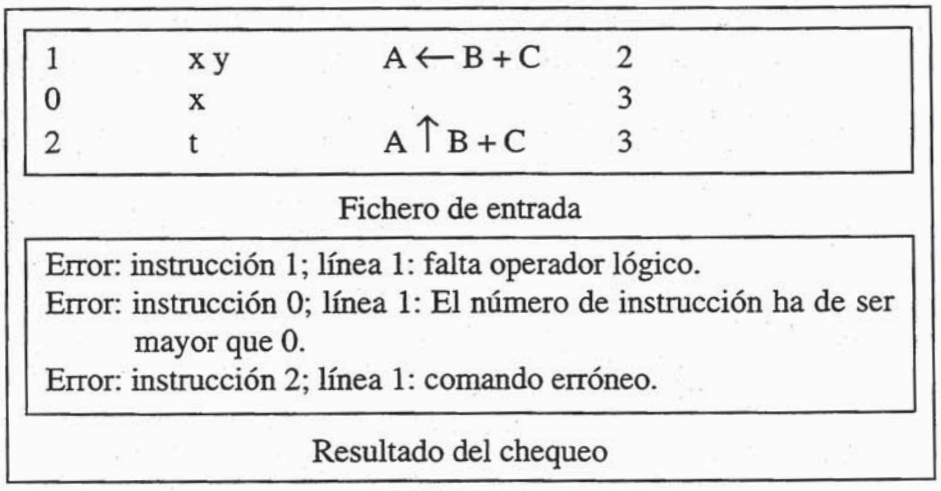

Figura 2.- Ejemplo de errores sintácticos

La última fase del chequeo es el análisis semántico. En esta etapa se comprueba que la información que se describe en el fichero de entrada en HDL sea coherente, es decir, que tenga un significado en conjunto. Buscará errores como, por ejemplo, los que se referencian en la Figura 3. El primero de ellos ocurre si se ha escrito un número de próxima instrucción no exis-

tente, en cuyo caso el programa respondería tal como se muestra en la Figura 3. En el segundo ejemplo se supone que las variables x e y pueden valer 0 simultáneamente. En este caso, el conjunto de condiciones expresadas por las funciones combinacionales $f_i(x)$ no sería completo ya que ante esta instrucción, faltaría por especificar la evolución del sistema que ocurre para el caso $x = 0$, $y = 0$. En este caso, no se cumplirían ninguna de las condiciones $f_i(x)$ de la instrucción y el sistema no tendría determinada la conducta a seguir, lo que produce un error de operación. El chequeo produce la respuesta mostrada en la Figura 3.

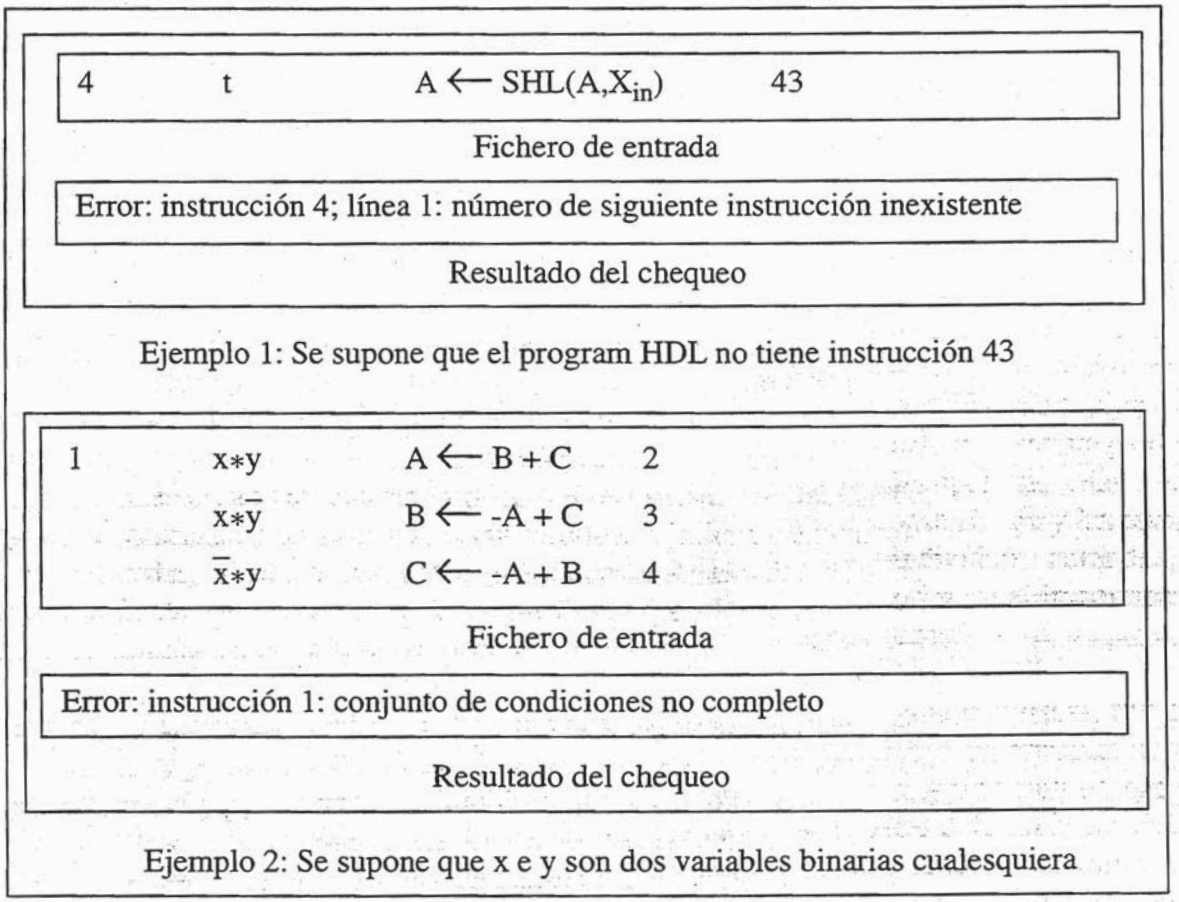

Figura 3.- Ejemplo de errores semánticos

3.3.- Obtención de la unidad de control

Tras la localización sucesiva de errores del código HDL y su corrección se procede (Figura 1) a la obtención del esquema del circuito. En el estado de desarrollo actual hemos debido establecer dos requisitos para poder hacer operativa la herramienta. El primero de ellos es que el diseño del circuito se realizará automáticamente sólo si el programa HDL describe unidades de control. Esto es, el campo de acciones de la instrucción corresponderá a las señales de entrada de control que hay que activar en los componentes de la unidad de procesado de datos. El segundo requisito es que, entre las múltiples opciones [2, 4] de realización para las unidades de control, sólo se ha implementado la basada en 1 biestable por estado (a veces llamada también de asignación "1-activo" o basada en registro de desplazamiento).
La estrategia de diseño elegida ha sido implementada realizando un proceso en dos pasos. En el primero se ejecuta una aproximación formal desde el programa HDL hacia el circuito de control. En el segundo se reduce algún hardware sobrante que introduce la aproximación formal anterior.

Mediante esta estrategia, a cada instrucción en nuestro lenguaje se le asocia un biestable. A la salida de este podrá haber un demultiplexor o conjunto de demultiplexores controlados por las señales que son variables independientes en las condiciones de la instrucción. Las salidas de los demultiplexores estarán asociadas a cada condición y activarán a las señales que se indican como comandos en su línea de condición. Asímismo, cada línea terminará en el biestable que represente a la próxima instrucción.

3.4.- Líneas de ampliación

Hemos debido restringir el dasarrollo de la herramienta presentada en este trabajo con el fin de hacerla operativa. Aunque las restricciones son compatibles con la forma actual de impartir la asignatura, lo cual la hace útil en nuestra docencia, es nuestro propósito dotarla con más funcionalidad conforme vayamos avanzando a su último estado de desarrollo. Así, tenemos planteado un conjunto de ampliaciones como son:

- A corto plazo:
  - incluir entrada vía cartas ASM y convertir descripción HDL a/desde ASM.
  - diversificar las formas de realizar los controladores, incluyendo la posibilidad de conexión a herraminetas de síntesis de circuitos secuanciales síncronos y de implantación de técnicas microprogramadas.
- A medio plazo, crecer hacia el tratamiento de unidades de procesado de datos, la verificación funcional y la emigración hacia VHDL.

4.- CONCLUSIONES

En este trabajo presentamos parte de una herramienta para CAD de sistemas digitales a nivel RT. La disponibilidad de una herramienta de este tipo es de gran interés docente, ya que facilita la resolución de problemas, para resolverlos se aplican procedimientos de programación y, en conjunto, se establecen dos puentes muy adecuados, uno entre el hardware y el software del ordenador, y el otro entre los niveles bajos (puertas-biestables) y altos (instrucciones) de los sistemas digitales.

La herramienta recibe un programa HDL que describe una secuencia de procesos sobre datos a nivel RT o de acciones de control. Entonces chequea su correción léxica, sintáctica y semántica ofreciendo, en su caso, información de los errores. Si está correcto y el programa HDL describe control, también genera el diseño de la unidad de control basado en la asignación 1-activo. Además, la herramienta ha sido diseñada para ser fácilmente ampliada.

5.- REFERENCIAS

[1] Weste N. H. E. y Eshraghian K.: "Principles of CMOS VLSI design". 2nd edition. Ed. Addison-Wesley. 1993.

[2] Hayes J. P.: "Diseño Lógico Digital". Ed. Addison-Wesley Iberoamericana. 1996.

[3] Goto S.: " :Design methodologies". Ed. North-Holland. 1986.

[4] Katz R. H.: "Contemporary Logic Design". Ed. The Benjamin/Cummings. 1994.

[5] Parra M.P., Baena M.C., Bellido M.J. y Valencia M.: "Enseñanza integrada: una aplicación a la docencia de circutos secuenciales". *Actas del I Congreso Sobre Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE'94)*, pp. 213-222. 1994.