I Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica, TAEE'94, Madrid julio 1994

UN LABORATORIO DE ESTRUCTURA DE COMPUTADORES BASADO EN VHDL

José J. Ruz y Alvaro Ruiz-Andino

Dept. Informática y Automática
F. Físicas, Ciudad Universitaria
28040 Madrid

Telf: 394-43-75
Fax: 394-46-87
E-mail: jjruz@dia.ucm.es

RESUMEN

En la presente comunicación describimos el contenido y la metodología de un curso de análisis y diseño de estructura de computadores utilizando el lenguaje de descripción hardware VHDL. Con este curso - impartido en la asignatura Laboratorio de Estructura de Computadores de la Escuela Superior de Informática de la Universidad Complutense - hemos pretendido incorporar a las clases prácticas de laboratorio aquellos contenidos teóricos de las asignaturas de hardware que debido a su complejidad y al coste de los equipos físicos tradicionalmente quedaban fuera de las mismas. Desde VHDL (VHSIC Hardware Description Language) - un lenguaje impulsado por el Departamento de Defensa (DoD) de los Estados Unidos dentro del programa VHSIC (Very High Speed Integrated Circuits) y estandarizado por IEEE Computer Society - los alumnos puede estudiar los dispositivos hardware a diferentes niveles de abstracción, acortando considerablemente las fases diseño y teniendo un mejor conocimiento de los mismos cuando abordan la fase de implementación física. El curso está concebido para estudiantes de Informática con conocimientos importantes en ingeniería de software, y está organizado para poner dichos conocimientos al servicio del diseño y la modelado hardware.

1. INTRODUCCIÓN

Los laboratorios de prácticas son una pieza fundamental en los estudios de informática, ya que estimulan la dimensión creativa de los alumnos y generan confianza en el uso de las máquinas. Pero no todas las asignaturas se llevan al terreno práctico con la misma facilidad. Mientras que un lenguaje de programación de propósito general es suficiente para llevar a los laboratorios casi todas las materias relacionadas con el software, no ocurre lo mismo en el caso del hardware. La complejidad creciente de los sistemas y el costo de los equipos ha hecho que muchos contenidos de las asignaturas de hardware hayan quedado tradicionalmente fuera de las clases de laboratorio. Esta deficiencia se ha ido corrigiendo en los últimos años con la ayuda de simuladores específicos de determinados aspectos de la arquitectura de los computadores, del diseño lógico, etc., pero el carácter específico de estos simuladores limita su utilidad. Se necesita cambiar de entorno cada vez que se cambia de materia, con la consiguiente inversión de tiempo para la familiarización con las diferentes herramientas. Con VHDL [1-6] los alumnos pueden estudiar un mismo sistema a diferentes niveles de abstracción sin salir del lenguaje, facilitando una estrategia analítica de diseño (topdown), optimizando el tiempo de aprendizaje de los entornos de utilización y disponiendo de un mejor conocimiento del sistema si tienen que abordar la fase de implementación física.

El curso que presentamos en la presente comunicación está concebido para estudiantes de Informática con conocimientos de ingeniería de software, y está organizado para poner dichos conocimientos al servicio del diseño y la modelado hardware. Otros planteamientos sobre el tema introducen VHDL desde la óptica del diseñador tradicional de hardware, es decir, comenzando con las construcciones VHDL que permiten la descripción estructural y de transferencia de registros de un sistema. Este enfoque resalta la correspondencia existente entre dichas construcciones y los dispositivos físicos que modelan, creando una primera visión del lenguaje que, aunque sencilla, no facilita la comprensión posterior de su modelo temporal, imprescindible para entender y utilizar adecuadamente todas las posibilidades del lenguaje. Dichas construcciones son en realidad formas sintácticamente diferenciadas de la única sentencia concurrente disponible en el lenguaje: el proceso. Nuestro planteamiento es justo el inverso [7], partimos del estudio exhaustivo de las características puramente secuenciales de VHDL, conocidas ya en buena parte por los estudiantes a los que va dirigido el curso. Posteriormente se introduce el modelo temporal que hace posible la ejecución concurrente de procesos, y después las variaciones sintácticas de dichos procesos, es decir las sentencias concurrentes. Esta organización de los contenidos básicos del curso transmite una mayor unidad del lenguaje y proporciona los fundamentos de su semántica operacional, imprescindible para entender el significado de las abundantes posibilidades expresivas de VHDL.

El curso se inicia con un tutorial de cinco sesiones sobre VHDL en el que partiendo de las características secuenciales del lenguaje se van introduciendo progresivamente aquellos recursos expresivos própios de un lenguaje de descripción hardware. En la segunda parte se abordan dos proyectos de diseño, uno dedicado al desarrollo de un multiplicador binario siguiendo el algoritmo de suma y desplazamiento, y otro, de mayor complejidad, centrado en el diseño completo de un pequeño computador. Con el primero se pretende que los alumnos se inicien en el diseño de máquinas algorítmicas utilizando un ejemplo en el que la sencilla interacción entre ruta de datos y controlador facilita la introducción de técnicas basadas en tipos abstractos de datos para la gestión y depuración simbólica de las relaciones de control. En el segundo proyecto se hace uso de los conocimientos anteriores para llevar al terreno operativo los conceptos sobre diseño y funcionamiento de un computador con arquitectura convencional tipo von Neumann.

2. TUTORIAL VHDL

El tutorial consta de cinco sesiones, las dos primeras dedicadas a VHDL secuencial, las dos siguientes a VHDL concurrente y la última a las unidades de diseño. En la primera sesión se introducen los conceptos básicos del lenguaje, se utilizan sus construcciones secuenciales más importantes y se modelan pequeños sistemas combinacionales y secuenciales. En la segunda, los modelos de comportamiento de varios operadores hardware complejos sirven de pretexto para que los alumnos utilicen exhaustivamente la capacidad expresiva que VHDL tiene para describir procesos. En la tercera se introduce el modelo temporal de VHDL, resaltando la diferencia entre el dominio secuencial, donde opera un funcionamiento puramente algorítmico, y el dominio concurrente, compuesto por un conjunto de procesos ejecutandose asincronamente y comunicandose mediante una red de señales. En la cuarta sesión se aborda el estudio de las sentencias concurrentes como formas particulares de procesos escritos con una sintaxis apropiada para construir los estilos estructural y de transferencia de registros, tradicionales en los lenguajes de descripción hardware. En la quinta y última sesión del tutorial se estudian las unidades de diseño VHDL, las bibliotecas de entidades y las diferentes configuraciones que se pueden establecer para un mismo dispositivo hardware. La tabla 1 resume las prácticas realizadas en cada una de las sesiones del tutorial.

En los dos siguientes subapartados resumiremos el enfoque dado en el curso a las componentes secuencial y concurrente de VHDL.

2.1 VHDL SECUENCIAL

La unidad principal de diseño VHDL es la *entidad*, que puede representar cualquier elemento hardware, desde una puerta lógica a un sistema digital complejo. Una entidad consta de dos partes,

Práctica 1: Conceptos básicos de VHDL

Diseño de sistemas digitales sencillos: descodificadores, codificadores, multiplexores, etc..

Práctica 2: Modelos de Comportamiento con VHDL Secuencial

1. Diseño de una Memoria de tipo Pila
2. Diseño de un Operador Aritmético de Enteros
3. Diseño de un Operador Aritmético de Reales

Práctica 3 : Modelos Concurrentes en VHDL

1. Diseño de dos procesos concurrentes
2. Diseño de un biestable JK síncrono
3. Diseño de dos procesos dialogantes

Práctica 4 : Sentencias Concurrentes

Diseño de un sistema secuencial a tres niveles:
1. Comportamiento
2. Transferencia de registros o flujo de datos
3. Estructural

Práctica 5 : Sentencias de Generación de Procesos

Diseño de redes Iterativas.

Tabla 1. Prácticas realizadas en el tutorial VHDL

la *declaración de entidad* y la *arquitectura de la entidad*. La primera define en forma declarativa la interfaz de la entidad con su entorno, es decir, las señales de entrada/salida, y es equivalente al símbolo gráfico que habitualmente se utiliza para representarla. La *arquitectura* define la relación funcional entre las entradas y las salidas de la declaración de entidad. Dicha relación funcional se puede definir utilizando diferentes estilos del lenguaje, pero básicamente consta de una serie de procesos interconectados mediante señales. Un declaración de entidad puede tener definidas más de una arquitectura, reflejando cada una de ellas aspectos diferentes de la entidad. En la figura 1 hemos representado la declaración de entidad *nombre* junto a tres de sus arquitecturas, *primera*, *segunda* y *tercera*..

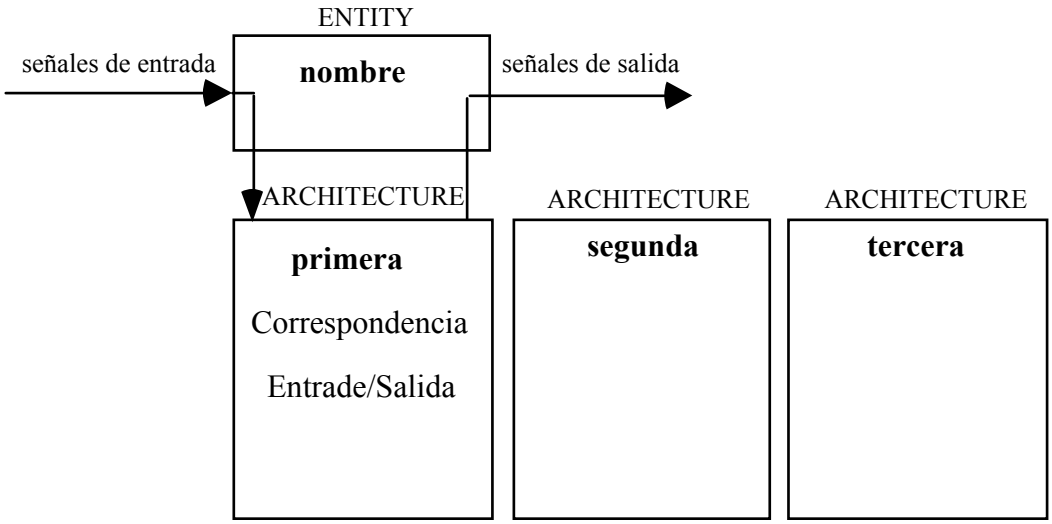

Figura 1. Declaración de entidad junto a tres de sus arquitecturas

Una declaración de entidad VHDL declara el *nombre* de la entidad, las *señales de entrada y salida*, seguidas por los respectivos *modos* (IN para las entradas, OUT para las salidas) y sus *tipos*. En la figura 2 aparecen las estructuras sintácticas de la declaración de entidad y la arquitectura.

```
ENTITY  nombre  IS
        PORT ( señales_de_entrada  :  IN  tipo;  señales_de_salida  :  OUT  tipo);
END  nombre;
```

```
ARCHITECTURE  primera  OF  nombre  IS
BEGIN
        PROCESS
                Declaraciones
        BEGIN
                Cuerpo del Proceso
                (Algorítmo Secuencial)
        END PROCESS;
END primera;
```

Figura 2. Estructura sintáctica de la declaración de entidad y arquitectura.

El algoritmo secuencial del cuerpo de un proceso se construye con sentencias típicas de los lenguajes imperativos. Básicamente existen tres grupos: *sentencias de asignación de variable*, *sentencias de control de flujo de ejecución*, y *sentencias de interacción con el exterior*. La semántica de los dos primeros grupos es igual que la de los lenguajes software. Su sintaxis es la siguiente:

1) Sentencia de asignación de variable

```
        variable  :=  expresión;
```

2) Sentencias de control del flujo de ejecución

a) Condicional

```
        IF condición THEN  sentencias_1  ELSE  sentencias_2;
```

b) Alternativa

```
        CASE  expresión  IS
                WHEN  valor_1  =>  sentencias_1;
                WHEN  valor_2  =>  sentencias_2;
                ---------------------------------------------------------
                WHEN  valor_n  =>  sentencias_n;
        END CASE;
```

c) Bucle iterativo

```
        FOR índice  IN rango  LOOP  sentencias  END  LOOP;
```

Las sentencias de interacción con el exterior son específicas del lenguaje VHDL y responden a necesidades particulares del modelado de los dispositivos hardware. La sintaxis simplificada es la siguiente:

3) Sentencias de interacción con el exterior

a) Asignación de señal

```
señal <= expresión AFTER retardo;
```

b) Espera de eventos

```
WAIT ON lista_de_señales;
```

Una *señal* es un objeto de datos VHDL que tiene asociado un *driver*, es decir, una lista de pares (*valor*, *tiempo*) ordenados por la componente *tiempo*. El driver de una señal representa la evolución temporal futura de la misma, es decir, los valores que irá tomando en los diferentes instantes de tiempo. El valor de una señal en un instante *t* de simulación es el valor del par de su driver cuya componente de tiempo coincide con el de simulación. En la figura 3 aparecen el driver de la señal *s1* y la representación gráfica de su evolución temporal.

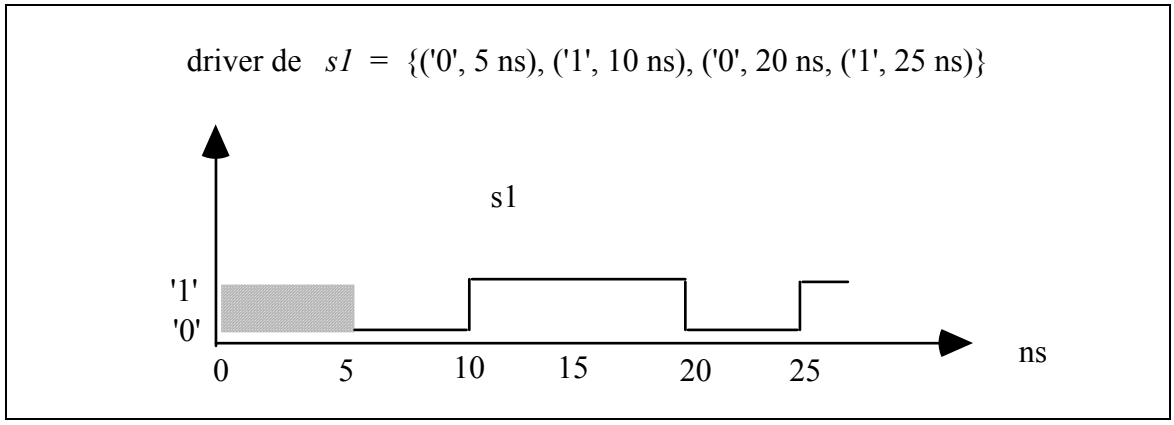

Figura 3. Driver y representación gráfica de una señal.

La *sentencia de asignación de señal* crea en el driver de la señal asignada un nuevo par (*valor, tiempo*). El *valor* es el resultado de evaluar la *expresión* de la sentencia, y el *tiempo* es igual a la suma del tiempo actual de simulación (en el que se ejecuta la sentencia) más el *retardo* declarado a la derecha de la palabra reservada AFTER. Digamos que una sentencia de asignación de señal planifica un valor futuro para dicha señal, valor que se convertirá en el actual cuando transcurra un intervalo de tiempo igual al retardo. Este es el mecanismo que utiliza VHDL para simular los tiempos de retardo de los dispositivos hardware.

La *sentencia de espera de eventos* permite la sincronización entre diferentes procesos. En efecto, las sentencias del cuerpo de un proceso se ejecutan secuencial y cíclicamente de la primera a la última: para suspender un proceso hay que ejecutar la sentencia WAIT, reanudandose la ejecución cuando se produzcan determinados eventos en las señales del proceso o se cumplan determinadas condiciones. Aunque una arquitectura puede constar de varios procesos conectados a través de señales, en las dos primeras sesiones se consideran solo arquitecturas con un único proceso para centrar el estudio en las construcciones secuenciales.

2.2 VHDL CONCURRENTE

En VHDL la concurrencia se construye en torno a un modelo temporal discreto que esencialmente consta de un conjunto de procesos ejecutándose asíncronamente y comunicándose mediante una red

de señales. En el interior de un proceso las acciones se suceden en el orden secuencial marcado por las construcciones de control típicas de un lenguaje imperativo, pero en un tiempo nulo. El tiempo se expresa en forma de retardo asociado a los valores que se asignan a las señales utilizando la *sentencia de asignación de señal.* Cada valor se convertirá en el valor actual de la señal cuando transcurra un tiempo de simulación igual al retardo. En VHDL tenemos, pues, un dominio secuencial en el que opera un funcionamiento puramente algorítmico, y un dominio concurrente en el que evolucionan los procesos a ritmo de eventos discretos. La figura 4 muestra los ámbitos de estos dos dominios dentro de la arquitectura de una entidad.

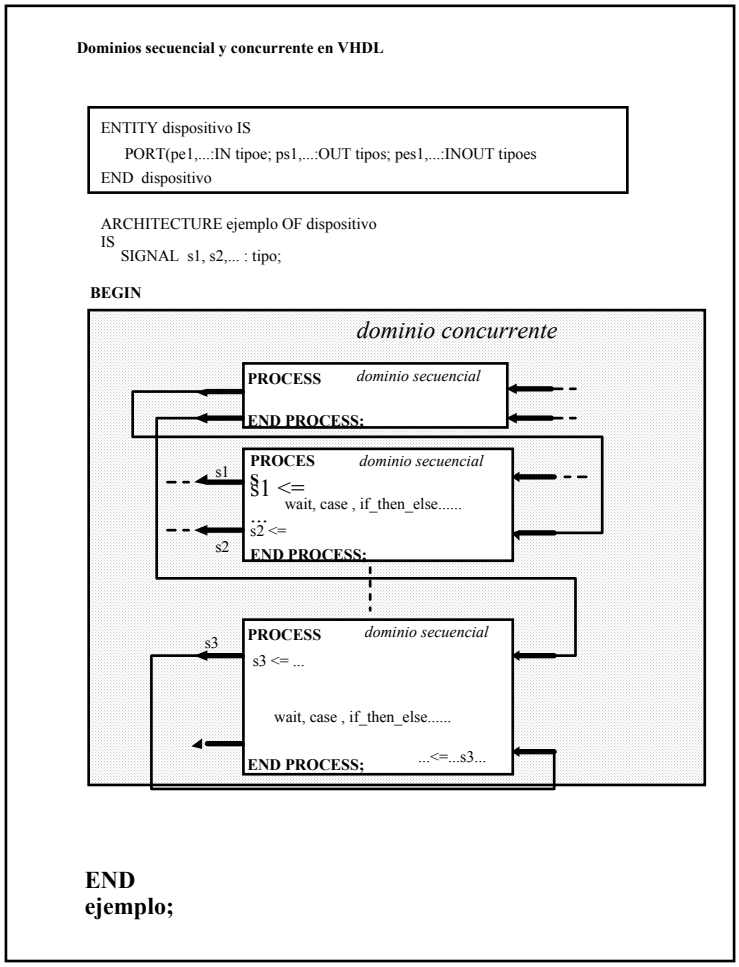

Figura 4. Dominios secuencial y concurrente en VHDL

Para el dominio secuencial VHDL dispone de unos recursos y una disciplina software heredados en su mayor parte del lenguaje Ada. Para el dominio concurrente dispone fundamentalmente de la sentencia de proceso (*process*), que define los límites de un dominio secuencial, y diferentes sentencias concurrentes que son en realidad formas particulares de procesos escritos con una sintaxis diferenciada que permiten construir los estilos estructural y de transferencia de registros (RTL) tradicionales en los lenguajes de descripción hardware.

Cuando en el cuerpo de una arquitectura hay más de un proceso, estos se ejecutan concurrentemente, es decir, sin que exista un orden entre ellos. Los procesos pueden compartir señales (declaradas al principio de la arquitectura) para comunicarse entre si. En unos procesos la señal compartida aparecerá como señal asignada por una *sentencia de asignación de señal*, es decir, a la izquierda del símbolo <=. En este caso diremos que el proceso es un *proceso fuente* de esa señal. En general, una señal solo puede tener un proceso fuente. En otros procesos la señal compartida aparecerá en el lado derecho de una sentencia de asignación de señal, de una asignación de variable o en otras construcciones secuenciales. En este caso diremos que el proceso es un

proceso destino de la señal. Cada proceso representará un componente hardware del sistema modelado, siendo las señales los canales de comunicación (hilos conductores) entre componentes. En cada ciclo de simulación, los procesos se ejecutan con los valores actuales de las señales y generan nuevos valores para el futuro a través de las sentencias de asignación de señal.

3. DISEÑO CON VHDL

El diseño con VHDL se aborda en la segunda parte del curso utilizando dos proyectos, uno sobre máquinas algorítmicas (diseño lógico) y otro sobre estructura de computadores con arquitectura convencional. La tabla 2 resume las correspondientes prácticas.

Práctica 6 : Diseño Lógico (ruta de datos)

1 Diseño de las unidades funcionales de la ruta de datos del multiplicador
2. Diseño estructural de la ruta de datos del multiplicador

Práctica 7 : Diseño Lógico (controlador)

1. Diseño del controlador del multiplicador
2. Conexión controlador-ruta de datos

Práctica 8 : Modelo de Comportamiento de un Computador

1. Modelo de comportamiento de un computador utilizando un único proceso
2. Modelos de comportamiento independientes para la CPU y la Memoria: conexión estructural.

Práctica 9 : Modelo Estructural de un Computador

1. Modelo estructural de la ruta de datos (RD)
2. Modelo de comportamiento la unidad de control (UC)
3. Conexión estructural UC-RD-MEM

Tabla 2. Prácticas de diseño con VHDL

3.1 DISEÑO LOGICO

El diseño de un sistema digital complejo se aborda descomponiendolo en una *ruta de datos* (datapath) y un *controlador* . En la ruta de datos se mantiene y transforma el estado del sistema. El controlador gobierna dichas transformaciones, ejecutando el algoritmo que determina las acciones y el orden temporal (paso de control) a realizar en la ruta de datos.

La ruta de datos está constituida por un serie de registros para soportar el estado del sistema, una serie de operadores para su transformación, y una serie de buses de comunicación con una determinada topología. Para guiar las transformaciones existen una serie de puntos de control distribuidos por la ruta de datos sobre los que actúa el controlador del sistema. La figura 5 representa un trozo genérico de una ruta de datos

Para representar una ruta de datos utilizaremos una señal *r1* para mantener el estado de un registro, otra señal *ir1* para el estado del bus de entrada al registro y sentencias de asignación concurrentes condicionales para realizar las transferencias de información y sus transformaciones combinacionales. La figura 6 representa el esquema VHDL de la ruta de datos de la figura 5.

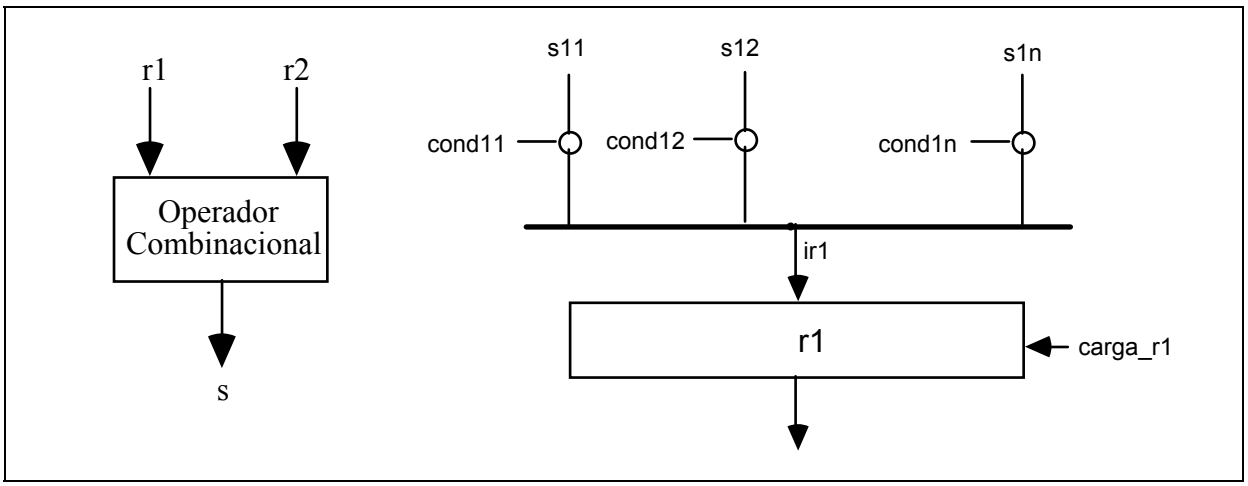

Figura 5. Trozo genérico de ruta de datos

```
 ARCHITECTURE rtl OF ruta_datos IS
.....................................................
 SIGNAL ir1, ir2, ... ,irm; r1, r2, ... ,rm;

 ir1     <=      s11 WHEN cond11 ELSE
                 .....................
                 s1n WHEN cond1n ELSE
                 ideterminado;
.....................................................
 irm     <=      sm1 WHEN condm1 ELSE
         .....................
                 smp WHEN condmp ELSE
                 indeterminado;

 registros : PROCESS
 BEGIN
  WAIT UNTIL flanco_positivo(clk);
  IF     control(carga_r1) = '1'        THEN r1 <= ir1;       END IF;
                                    .....................
  IF     control(carga_rm) = '1'        THEN rm <= irm;       END IF;
 END PROCESS registros;
```

Figura 6. Esquema VHDL de la ruta de datos de la figura 5.

3.2. DISEÑO DE UN COMPUTADOR

En las dos últimas sesiones se aborda el diseño de un computador elemental como el que aparece en la figura 7.

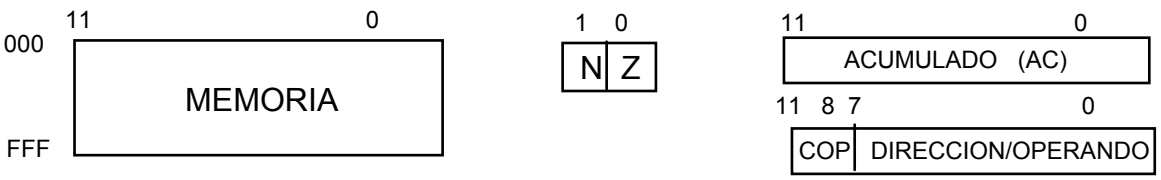

Figura 7. Zonas de datos y formato de instrucción del computador

En primer lugar se modela el comportamiento a nivel del repertorio de instrucciones utilizando un único proceso. Las zonas de datos se representan con variables, las transferencias de información con sentencias de asignación de variables y las transformaciones de información con operadores y funciones. Por ejemplo, la fase de búsqueda vendrá representada por el siguiente segmento VHDL:

```
mar := pc;                          -- mar <-- <pc>
mdr := mem(b_a_n(mar));             -- lectura
ir := mdr;                          -- ir  <-- <mdr>
pc := n_a_b((b_a_n(pc) + 1), 12);   -- pc  <-- <pc> + 1
```

Después se descompone el sistema en dos subsistemas, el procesador y la memoria, y se introduce el comportamiento temporal de los buses que conectan ambas unidades (figura 8).

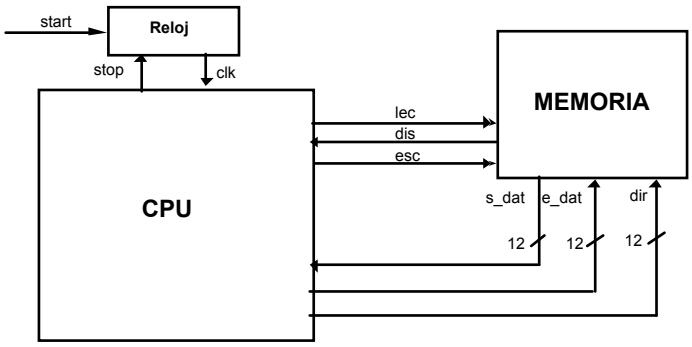

Figura 8. Descomposición CPU-Memoria

Finalmente, se diseña el modelo estructural a nivel de registros y unidades operacionales (figura 9).

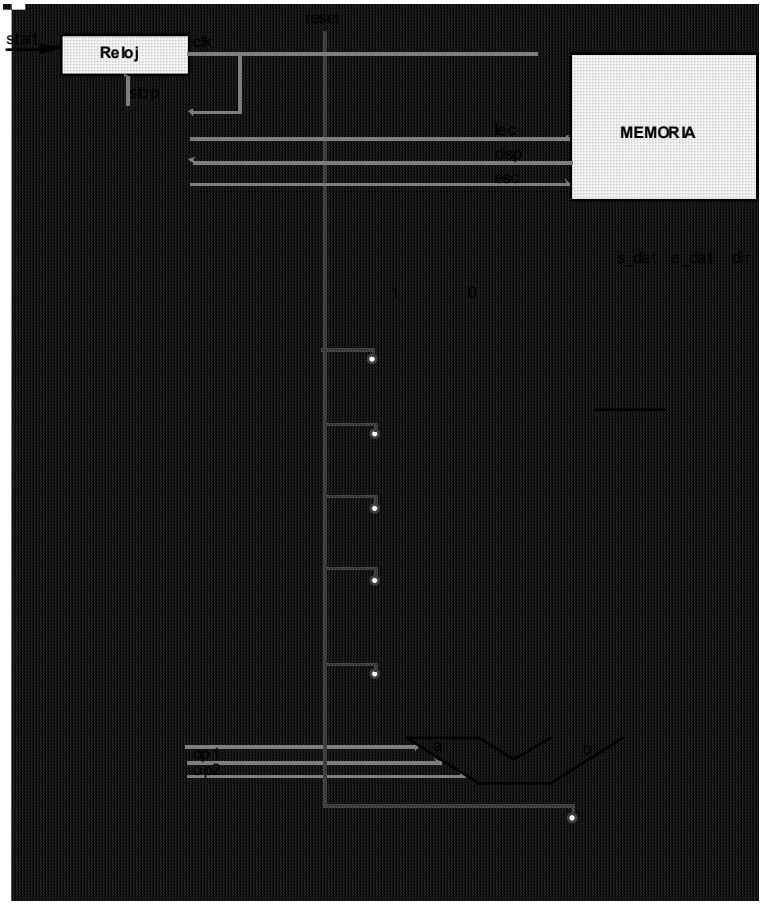

Figura 9. Modelo Estructural del computador

6. Resultados y Conclusiones

Los resultados obtenidos en el primer año de rodaje del curso han sido altamente positivos. El 90% de los alumnos, trabajando en equipos de dos, han finalizado con éxito un diseño original de un pequeño computador. El trabajo lo han desarrollado en tres fases. En la primera han modelado el comportamiento de la arquitectura a nivel de repertorio de instrucciones. En la segunda han descompuesto el sistema en UCP y Memoria, diseñando y modelando el comportamiento individual de estas unidades y los buses de comunicación. Finalmente, en la tercera fase han descompuesto la UCP en Control y Ruta de Datos, y ésta última en registros, unidades funcionales y buses internos de comunicación. Desde este modelo estructural han realizado medidas de rendimiento para diferentes programas de prueba, poniendo de manifiesto el efecto sobre las medidas de algunos cambios practicados en la arquitectura.

Para cada unidad funcional (memoria, registros UALs, etc.) han diseñado modelos de comportamiento, es decir, modelos que no contemplan la estructura interna de la unidad sino su comportamiento externo observable. Posteriormente, cuando han adquirido los conocimientos precisos sobre el funcionamiento interno de estas unidades, han diseñado los correspondientes modelos estructurales definidos en términos de componentes más elementales, hasta llegar al nivel de puertas. Por ejemplo, para la memoria se ha utilizado un protocolo de comunicación asíncrono tipo "handshaking" para que posteriormente pudiesen integrar el comportamiento de un sistema de memoria tipo cache con diferentes políticas de emplazamiento, sustitución de bloques etc.

Como herramienta práctica se ha utilizado el V-Systen de Model Technology, complementado con los siguientes módulos: un presentador gráfico de señales, un compilador de EDIF a VHDL estructural que permite la introducción de esquemas gráficos editados desde el DRAFT de OrCAD/STD III, y un analizador y formateador de VHDL. Todos estos módulos, desarrollados en nuestro departamento, se han integrado en un único entorno de ventanas.

Con este curso hemos pretendido, además, que los alumnos dispusiesen de una herramienta operativa válida para otras asignaturas de la carrera. Por ejemplo, como lenguaje de entrada para las herramientas de síntesis CAD (VHDL Logic Synthesis de Synopsys Inc., Autologic de Mentor Graphics, etc.) utilizadas en las asignaturas sobre diseño de C.I. Para la implementación de modelos no interpretados de análisis del rendimiento de configuraciones de sistemas. O bien para analizar diferentes modelos de arquitecturas de computadores(procesadores segmentados, RISC, vectoriales, superescalares, arquitecturas especificas de la aplicación, multiprocesadores, etc.).

REFERENCIAS

[1] J. Armstrong, "Chip Level Modelling in VHDL, Prentice hall, 1988

[2] J.M.Bergé, A.Fonkoua, S.Maginot, and J.Rouillard, "VHDL Designer's Reference", Kluwer, 1992.

[3] D.R.Coelho, "TheVHDL Handbook", Kluwer Academic Publishers, 1989.

[4] IEEE Std 1076-1987, "IEEE Standard VHDL Language Reference Manual", IEEE Inc., New York, USA, 1987.

[5] R.Lipsett, C.Schaefer, C.Ussery, "VHDL: Hardware Description and Design", Intermetrics, Inc., Kluwer Academic Publishers, 1989.

[6] D.L.Perry, "VHDL", McGrawHill, 1991.

[7] J.J. Ruz, "Laboratorio de Estructura de Computadores", Notas del Curso, UCM, 1993.