Introducción a los lenguajes de descripción de hardware

Noviembre 2009
Jorge Juan Chico <jjchico@dte.us.es>
Departamento de Tecnología Electrónica
Universidad de Sevilla

Lección 1. Contenidos

- ¿Qué es un LDH?
- ¿Para qué sirve?
- Estructura de una descripción LDH
- Tipos de descripción
  - Asignación continua (combinacional)
  - Procedimientos
  - Estructural: módulos y primitivas
- Herramientas

¿Qué es un lenguaje de descripción de hardware (LDH)?

- Es un lenguaje formal que sirve para describir la operación de circuitos electrónicos.
- Es similar a un lenguaje de programación pero con notables diferencias:
  - Las directivas representan operaciones concurrentes
  - Existen diferentes formas de describir una función

```
// Operación AND
x = a & b;

// Operación OR
y = a | b;

// Función combinacional z = xy' + x'y
z = x & !y | !x & y;
```

¿Para qué sirven los LDH?

- Simulación
  - Permite simular la operación del circuito descrito mediante LDH
  - Permite asegurar la correcta operación del circuito antes de su fabricación (implementación)
- Síntesis automática
  - Hay programas de síntesis automática que permiten obtener una implementación de un circuito a partir de su descripción en LDH
  - Equivalente a la “compilación” en el software

Ejemplos de LDH

- VHDL
  - Sintaxis parecida al lenguaje ADA
  - Sintaxis más compleja
  - Gran número de tipos de datos
- Verilog
  - Sintaxis parecida al lenguaje C
  - Sintaxis más simple
  - Número de tipos de datos reducido

Estructura de una descripción LDH

- Declaración del módulo
  - Nombre del módulo
  - Definición de entradas y salidas
- Declaración de señales
  - Nombres y tipos de señales internas
- Descripción del diseño
  - Estructuras de procesamiento
  - Directivas del simulador
  - Directivas del pre-procesador (definiciones, etc.)

```
`timescale 1ns / 1ps

// Módulo: mayoria
// Descripción: función mayoría
//   f = ab + bc + ac

module mayoria(
    input a,
    input b,
    input c,
    output f
    );

    assign f = a&b | b&c | a&c;

endmodule    // mayoria
```

Tipos de descripciones

- Funcional (asignación continua)
  - Representa funciones (circuitos) combinacionales
  - Se asigna el valor definido constantemente
- Procedimental
  - Los valores asignados se describen mediante un procedimiento
  - El procedimiento se ejecuta ante el cambio de ciertas señales (lista de sensibilidad)
  - Pueden representar funciones (circuitos) combinacionales o secuenciales
  - Pueden emplearse estructuras de control
    - if-else, case, for, while, ...
  - En general esta descripción es la más fácil de escribir
- Estructural
  - Describe la interconexión de módulos y primitivas lógica
  - Equivalente al esquema del circuito

Tipos de descripciones

```
`timescale 1ns / 1ps

// Módulo: mayoria
// Descripción: función mayoría
//   f = ab + bc + ac

module mayoria(input a, input b, input c, output f);
```

```
assign f= a&b | b&c | a&c;
```

```
always @(a or b or c)
    if(a == 1)
        if(b == 1 || c == 1)
            f = 1;
        else
            f = 0;
    else
        if(b == 1 && c == 1)
            f = 1;
        else
            f = 0;
```

```
wire out1, out2, out3;

and and1 (out1, a, b);
and and2 (out2, b, c);
and and3 (out3, a, c);
or or1 (f, out1, out2,
out3);
```

```
endmodule    // mayoria
```

Resumen

- Los lenguajes de descripción de hardware (LDH) describen (modelan) el comportamiento de los circuitos electrónicos digitales
- Permiten definir módulos con diferente nivel de detalle
- Facilitan la simulación de los diseños antes de su realización
- Permiten la síntesis automática de circuitos a partir de sus descripciones en LDH

Herramientas

- Editor de texto
  - Escritura de código Verilog
- Compilador
  - Análisis del código y generación de datos para simulación
- Simulador
  - Simulación del circuito en base a las directivas de un banco de pruebas (test bench)
- Herramientas de síntesis
  - Implementación del circuito en una tecnología o dispositivo dado
  - Depende del fabricante de la tecnología
  - Es frecuente la implementación en dispositivos FPGA

Icarus Verilog

- http://www.icarus.com/eda/verilog/
- Icarus Verilog es un compilador y simulador de Verilog simple, rápido y compacto con una cobertura amplia del estándar.
- Junto con un editor de textos y el visor de ondas “gtkwave” tenemos un entorno de desarrollo en Verilog fácil de instalar y de usar.
- Icarus y gtkwave son software libre y se encuentran disponibles para una variedad de sistemas operativos.
- Los principales fabricantes de FPGA (Xilinx y Altera) disponen de potentes entornos de desarrollo que incluyen soporte para Verilog.
- Estos entornos pueden usarse para la implementación final del código previamente simulado con Icarus.

Icarus Verilog. Instalación en GNU/Linux

- Icarus y gtkwave están disponibles como paquete en la mayoría de las distribuciones de GNU/Linux.
- Por ejemplo, en Debian/Ubuntu:
  - Opción 1: Sistema->Administración->Gestor de paquetes Synaptic
    - Instalar los paquetes “verilog” y “gtkwave”
  - Opción 2: Ejecutar en un terminal:
    - $ sudo apt-get install verilog gtkwave
- Cualquier editor de texto plano sirve para editar código Verilog.
  - gedit (el editor de textos de GNOME) es fácil de usar y resalta la sintaxis del código verilog (archivos .v)

Icarus Verilog. Instalación en MS-Windows(TM)

- http://www.bleyer.org/icarus/
- En la dirección anterior se incluyen instaladores para Windows con Icarus y gtkwave.
- El autor de este documento ha tenido problemas con la instalación en carpetas con espacios en su nombre. Se recomienda la instalación en una ruta como “C:\programas\verilog”.
- Cualquier editor de texto plano sirve para editar código Verilog.

Bibliografía

- Verilog Tutorial
  - http://www.asic-world.com/verilog/veritut.html
- Online Verilog-1995 Quick Reference Guide
  - http://www.sutherland-hdl.com/online_verilog_ref_guide/vlog_ref_top.html
  - Imprescindible tenerlo a mano durante el diseño para consultar los detalles del lenguaje.