IE0323
Sistemas Digitales I

Clase Magistral
Introducción a Verilog Estructural

Objetivo

- Explicar a las personas estudiantes el proceso de implementación de circuitos digitales utilizando Verilog estructural, una metodología que permite describir sistemas digitales a nivel de compuertas lógicas.

- Entender la sintaxis y semántica de Verilog Estructural para diseñar y simular circuitos digitales.

Índice

1. Introducción a Verilog

2. Verilog Estructural

3. Testbench

4. Visualización de ondas

Introducción a Verilog

- Los **Lenguajes de Descripción de Hardware (HDL)** permiten diseñar, en forma abstracta, sistemas digitales complejos. Estos diseños pueden:
  - **Simularse**: verificar que el diseño es funcionalmente correcto antes de implementarlo
  - **Sintetizar**: convertir la descripción en una interconexión de componentes básicos dentro de un dispositivo programable (como lo es FPGA)
- Creado en 1984 por **Gateway Design Automation**
- Inspirado en C
- Permite distintos niveles de abstracción:
  - **Estructural**: basado en compuertas lógicas básicas
  - **Conductual**: descrito mediantes expresiones lógicas y procedimientos, estudiada más adelante

Verilog Estructural

- Describe los circuitos digitales a través de sus componentes básicos (como compuertas lógicas, memorias, etc) y sus interconexiones
  - Imitando la manera en que se construiría físicamente el hardware.
- Permiten optimizar los circuitos lógicos para maximizar la velocidad y minimizar el área.
- Se puede visualizar cómo se implementará el circuito real en hardware. Sin embargo, este método es más complicado para diseños con comportamientos complejos.

Verilog Estructural

- Se emplean funciones lógicas básicas para representar compuertas
  - **Operadores reservados:** `and`, `or`, `not`, `nand`, `nor`, `xor`, `xnor`, `buf`
  - **Sintaxis:** `or nombre_compuerta (salida, entrada1, entrada2, entrada3);`
  - **Ejemplo:** `or (x, ~a, b, ~c);` → `x = a' + b + c'`

- El inversor y el buffer pueden tener varias salidas, el último elemento de la lista es la entrada.

Verilog Estructural

- Para representar un sistema se requiere definir un **módulo**
  - Los argumentos son los nombres de las **variables** de **entrada**, de **salida** y **bidireccionales** del módulo.
- Luego deben colocarse listas de variables de entrada, de salida y bidireccionales. El módulo debe estar terminado con **endmodule**.
- Cuando se tienen varias compuertas lógicas y las salidas de esas compuertas necesitan ser conectadas entre sí, las señales entre estas compuertas se representan como cables (**wire**)

Ejemplo de estructura de un módulo

```
module ejemplo(A, B, C, D, Z);

  input A, B, C, D;

  output Z;

  wire n1, n2, n3;

  wire notA, notB, notC;

  not(notA, A);  // notA = A'

  not(notB, B);  // notB = B'

  not(notC, C);  // notC = C'


  and(n1, B, notC, D);   // n1 = B · C' · D

  and(n2, notA, notB);   // n2 = A' · B'

  and(n3, A, B, notC);   // n3 = A · B · C'

  or(Z, n1, n2, n3);     // Z = n1 + n2 + n3

endmodule
```

$$Z = B\bar{C}D + \bar{A}\bar{B} + AB\bar{C}$$

Testbench

- Parte esencial del diseño de circuitos digitales
- Permiten probar la funcionalidad de un módulo Verilog antes de sintetizarlo en hardware
- Es un módulo que instancia el Design Under Test (DUT) y le proporciona entradas para verificar su funcionalidad
- Se monitorean las salidas del módulo para asegurarse de que el comportamiento sea el esperado

Visualización de Datos