
Apéndice

Lenguajes de descripción de hardware

Circuitos Electrónicos Digitales
E.T.S.I. Informática
Universidad de Sevilla

Jorge Juan <jjchico@dte.us.es> 2010-2018

Esta obra esta sujeta a la Licencia Reconocimiento-CompartirIgual 4.0 Internacional de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/4.0/> o envíe una carta Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Contenidos

- Lenguajes de descripción de hardware
- Tipos de descripciones
- Estructura de una descripción Verilog
- Verilog: sintaxis y estructuras principales
- Banco de pruebas y simulación
- Síntesis desde LDH en FPGA
- Herramientas de diseño básicas

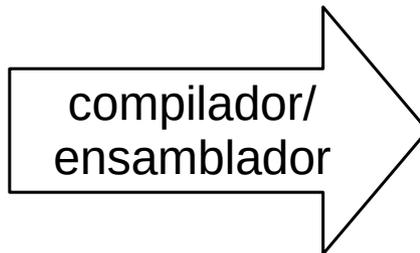
Bibliografía

- Lecturas recomendadas:
 - Libro de LaMeres
 - 5.1, 5.2, 5.3: introducción a los lenguajes de descripción de hardware y al flujo de diseño digital moderno.
 - Curso Verilog basado en ejemplos [<https://gitlab.com/jjchico/curso-verilog.v>]
 - Unidad 1: introducción
 - Unidad 2: bancos de prueba.
 - Unidad 3: circuitos combinacionales.
- Referencia (consultar en caso de dudas):
 - LaMeres 5.4 a 5.7: lenguaje Verilog, descripciones funcionales, estructurales y primitivas lógicas.
 - Verilog HDL Quick Reference Guide (Verilog-2001 standard)

¿como se diseñan los circuitos integrados?

código fuente Pitfall

```
SEG      Code
ORG      $f000, 0
START:
  sei
  cld
  ldx     #$00
Reset:
  lda     #$00
.loopClear:
  sta     $00,x
```

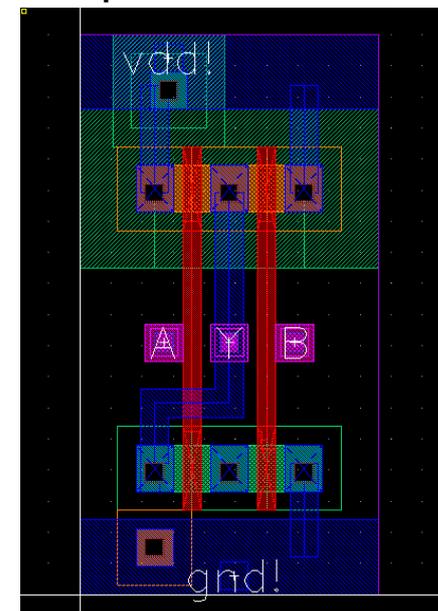


programa Pitfall

```
01111000
10100010
00000000
10101001
00000000
10010101
00000000
```



puerta NAND



¿Qué son los lenguajes de descripción de hardware (LDH)?

- Lenguajes formales para especificar, simular, implementar y verificar circuitos electrónicos.
- Aunque no son lenguajes de programación, su sintaxis es parecida.
- características notables:
 - La mayoría de las expresiones se “ejecutan” concurrentemente.
 - Cada expresión o “instrucción” corresponde a la operación de un bloque de circuito.

```
// AND operation
x = a & b;

// OR operation
y = a | b;

// Combinational function  $z = xy' + x'y$ 
z = x & ~y | ~x & y;
```

¿Por qué son útiles los LDH?

- Simulación
 - A partir de la descripción del circuito en un LDH es posible simular su comportamiento con los valores de entrada que se deseen (vectores de test) mediante herramientas informáticas (simuladores) para comprobar su correcto funcionamiento antes de construir el circuito real.
- Síntesis automática
 - A partir de la descripción del circuito en un LDH es posible diseñarlo de forma automática empleando herramientas informáticas.
 - Equivalente a la compilación del software.
 - La herramienta de síntesis puede realizar optimizaciones.
 - ¡Cuidado! El diseñador debe conocer lo que las herramientas pueden y no pueden hacer.

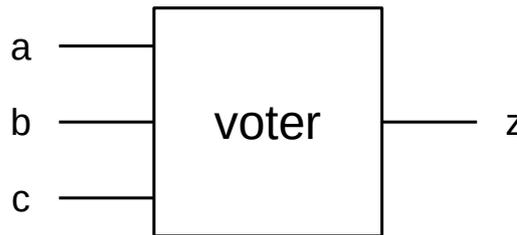
VHDL vs Verilog

- VHDL
 - Sintaxis más compleja, similar a ADA.
 - Sintaxis más estricta: reduce la posibilidad de errores.
 - Mejor soporte para diseños grandes y complejos.
- Verilog
 - Sintaxis más simple, similar a C.
 - Más fácil de aprender
 - Revisiones del lenguaje
 - 1995
 - 2001*
 - 2005
 - System Verilog

Tanto VHDL como Verilog están bien soportados por los fabricantes de hardware y pueden usarse indistintamente e incluso combinarlos en el mismo proyecto. La elección de uno u otro depende a menudo del gusto personal. Nos centraremos en Verilog.

Módulos

- Un módulo Verilog es equivalente a una clase en un lenguaje de programación.
- Describe un tipo de circuito.
- Pueden declararse distintas instancias de un mismo módulo.
- Ejemplo: un votador



a b c	z
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

Expresión lógica:

$$z = ab + ac + bc$$

Expresión en Verilog:

$$z = a \& b \mid a \& c \mid b \& c;$$

```
module voter(  
    output z,  
    input a,  
    input b,  
    input c  
);  
  
    assign z = a&b | a&c | b&c;  
  
endmodule
```

Descripción de un módulo Verilog

- Directivas del preprocesador
- Declaración de la interfaz
 - Nombre del módulo
 - Lista de puertos (conexiones) de entrada y salida
- Declaración de señales y componentes internos
 - Nombre y tipo de señales internas
- Descripción del diseño
 - Pueden mezclarse tipos de descripción
- Cualquier número de módulos puede describirse en un único archivo
- Comentarios
 - `//, /* ... */`

```
`timescale 1ns / 1ps

// Module: cvoter
// Conditional voting circuit
// z = ab + bc + ac if x=1

module cvoter(
    input wire x,
    input wire a,
    input wire b,
    input wire c,
    output reg z
);

    wire v;

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

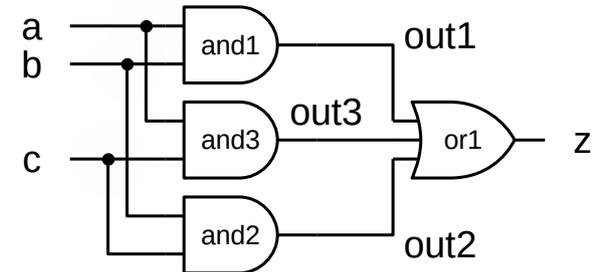
    always @(*)
        if (x == 1)
            z = v;
        else
            z = 0;

endmodule // cvoter
```

Tipos de descripciones

- Funcional (asignación continua)
 - Describe la función combinacional del circuito mediante una expresión lógica.
- Estructural
 - Describe los componentes internos del circuito y como se interconectan.
 - Verilog reconoce las puertas lógicas como módulos básicos.
- Procedimental (bloques *always*)
 - Describe un patrón de test (testbench) o el comportamiento de un circuito mediante un algoritmo.

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



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

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

Sintaxis de Verilog

Verilog HDL Quick Reference Guide
by Stuart Sutherland

http://sutherland-hdl.com/pdfs/verilog_2001_ref_guide.pdf

Verilog: puertos y señales

- Tipos de señales básicos
 - wire: se les puede asignar un valor en una descripción funcional, pero no en una descripción procedimental.
 - reg: se les puede asignar un valor en una descripción procedimental, pero no en una descripción funcional.
- Para cada puerto de entrada o salida se crea automáticamente una señal interna con el mismo nombre. Su tipo (wire, reg) puede especificarse.
- El tipo de los puertos puede especificarse cuando estos se declaran o en el cuerpo del módulo. Si no se especifica se considera de tipo “wire”.

```
module voter(  
    input wire a,  
    input wire b,  
    input wire c,  
    output reg z  
);  
  
always @(a, b, c)  
    if (a == 1)  
        if (b == 1 || c == 1)  
            z = 1;  
        else  
            z = 0;  
    else  
        if (b == 1 && c == 1)  
            z = 1;  
        else  
            z = 0;  
  
endmodule // voter
```

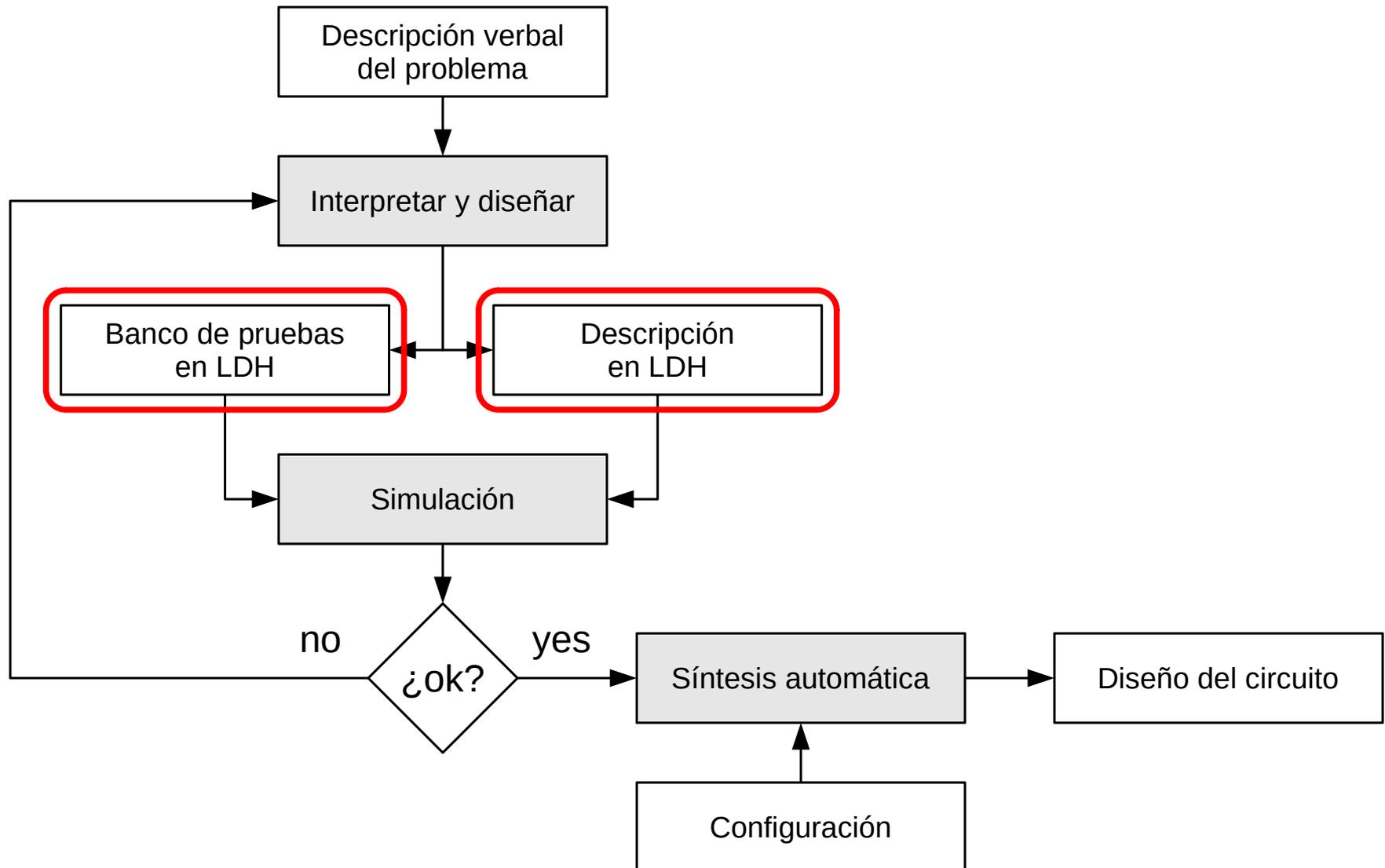
Verilog: procedimientos

- Describen el comportamiento de un circuito mediante estructuras de control: comparaciones, toma de decisión, bucles, etc.
- También describen bancos de pruebas.
- Similar al software, pero representa algo distinto.
- Tipos de procedimientos principales:
 - **initial**
 - Se ejecutan una sola vez al principio de la simulación.
 - Útil principalmente en los bancos de pruebas de simulación.
 - **always**
 - Pueden ejecutarse más de una vez.
 - Lista de sensibilidad: especifica cuando se ejecuta.

Banco de pruebas y simulación

- Un banco de pruebas (testbench) es un módulo que contiene:
 - Un circuito (instancia de otro módulo) que va a ser simulado: Unit Under Test (UUT)
 - Sentencias Verilog que describen las señales de entrada de la UUT para comprobar el correcto funcionamiento de la misma.
 - Directivas del simulador Verilog para controlar opciones de simulación: resolución temporal, final de la simulación, generación de resultados, etc.
- Características específicas de módulos de bancos de pruebas
 - Un módulo de banco de pruebas no está pensado para ser implementado, sólo para ser simulado.
 - Incluye sentencias Verilog que sólo son útiles para simulación. Ej: “initial”.
 - Un módulo de banco de pruebas no tiene entradas ni salidas (externas).

Proceso de diseño usando herramientas CAD (Computer-Aided Design)



Implementación de circuitos integrados

• **Application-Specific Integrated Circuit (ASIC):** Todas las fases del proceso de creación del circuito se realizan en la fabrica del chip.

• El diseñador del circuito establece que componentes habrá en el chip, donde se sitúan y como se interconectan.

• **Full Custom:** El diseñador del circuito se encarga diseño de los componentes a nivel geométrico.

• **Standard Cell:** El diseñador del circuito usa bibliotecas de componentes prediseñados.

• **Gate Array:** El diseñador del circuito sólo establece que componentes del chip se utilizan y como se interconectan.

• **Circuito Configurable:** El interconexionado y/o la configuración de los componentes del circuito se realizan fuera de la fábrica del chip.

• El circuito sólo puede configurarse una vez.

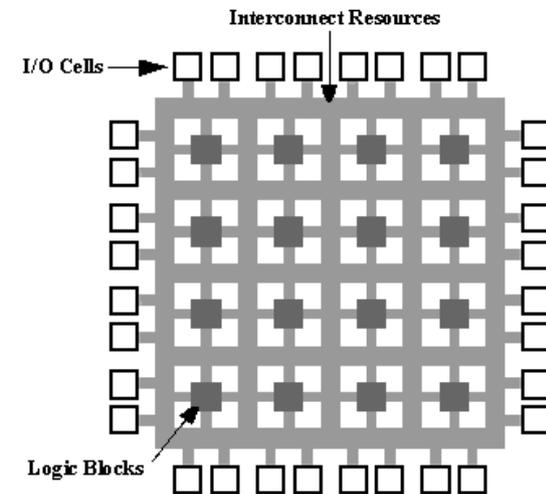
• **Programmable Array Logic (PAL)**
• **Programmable ROM (PROM)**

• El circuito es reconfigurable.

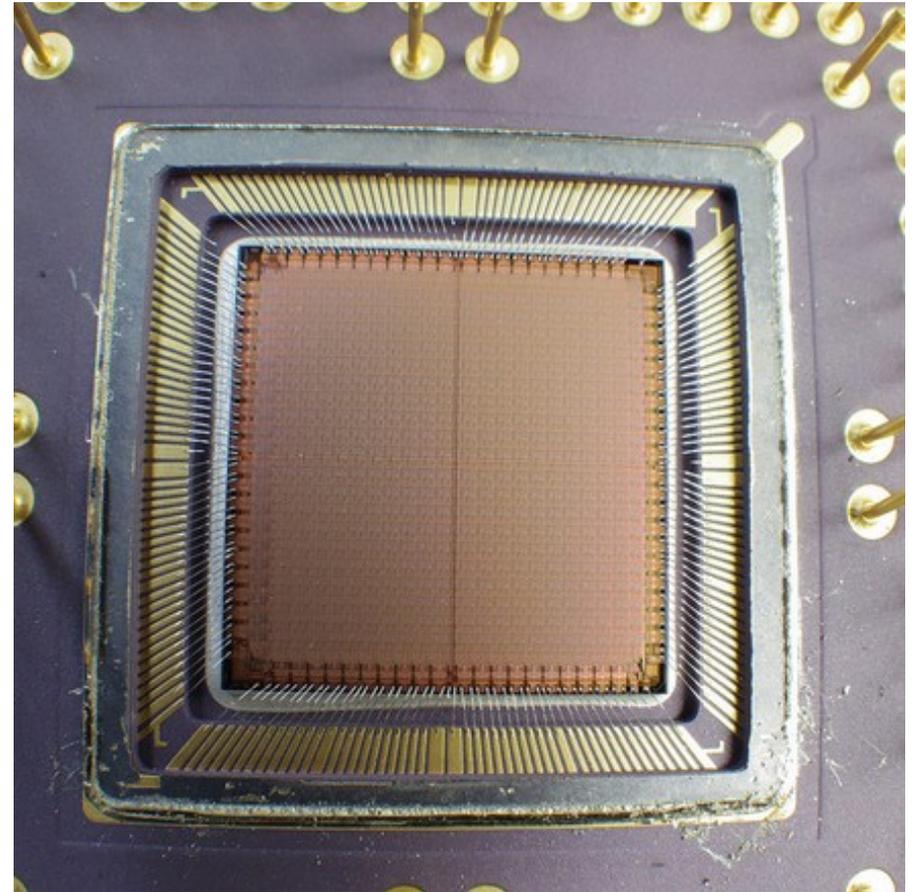
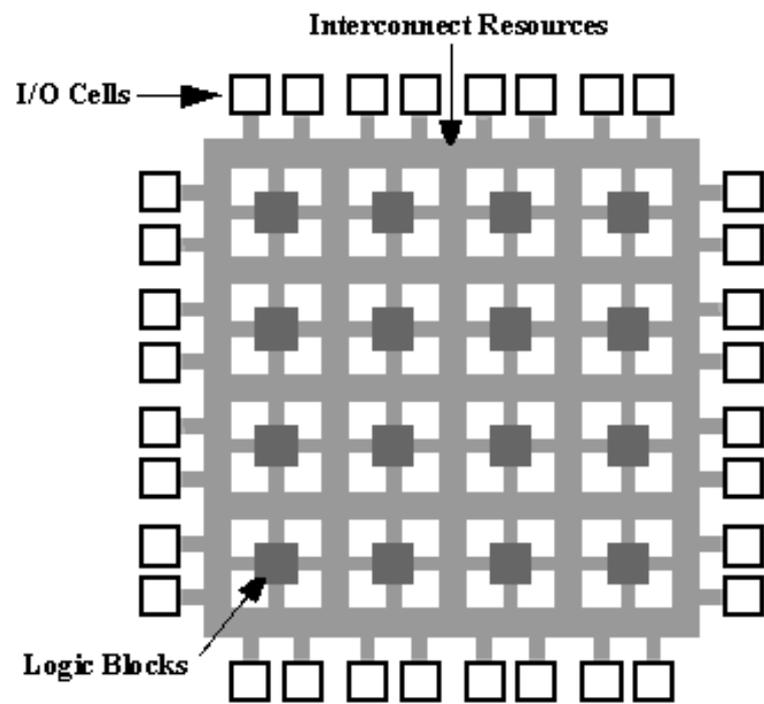
• **Erasable PROM (EPROM)**
• **Generic Array Logic (GAL)**
• **Complex Programmable Logic Device (CPLD)**
• **Field-Programmable Gate Array (FPGA)**

FPGA

- FPGA
 - Field Programmable Gate Array
 - Colección de dispositivos lógicos e interconexiones configurables.
- Configurable Logic Block (CLB)
 - Se configura para hacer una función determinada: AND, OR, XOR, etc.
- Input-Output Block (IOB)
 - Se configuran para actuar como entradas o salidas y conectar a señales internas
- Interconexiones
 - Se configuran para conectar los CLB e IOB a voluntad



FPGA



Síntesis en FPGA

- Síntesis de LDH sobre FPGA
 - El código LDH es analizado y las estructuras que describe se convierten (mapping) en dispositivos lógicos equivalentes.
 - Se seleccionan los CLB adecuados (placement) y se configuran para hacer la función de los dispositivos lógicos necesarios.
 - Se configuran las interconexiones para conectar los dispositivos (routing).
- Restricciones
 - Sólo se puede sintetizar un subconjunto de las estructuras disponibles en un LDH.
 - Cada fabricante de FPGA tiene sus propias restricciones.

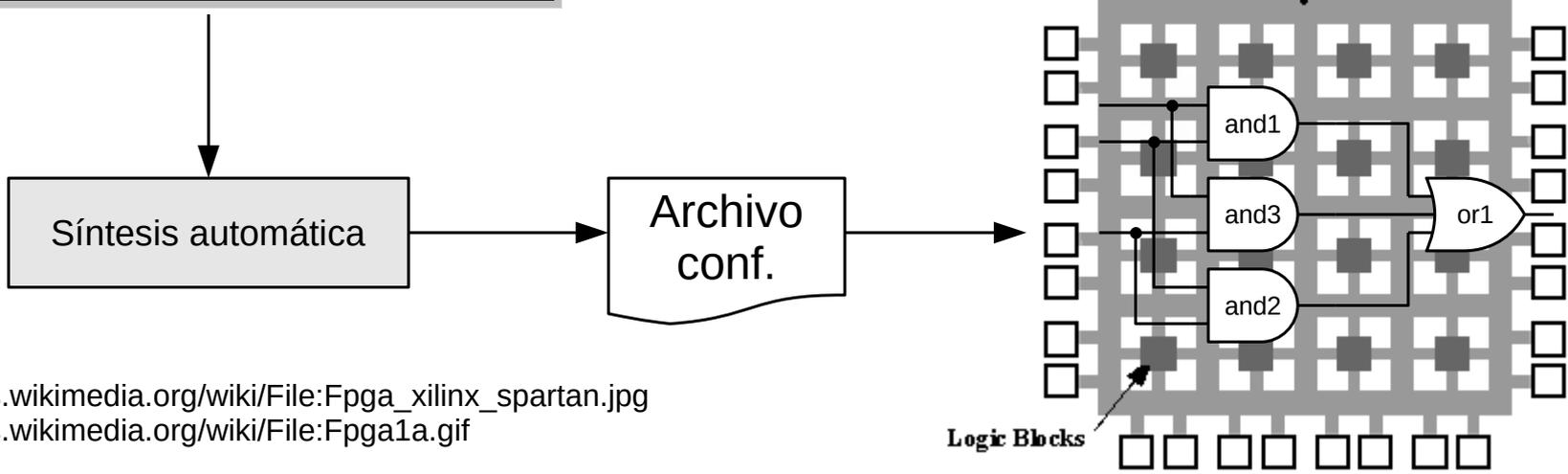
REGLA DE ORO

Si el diseñador no puede imaginar cómo implementar el circuito descrito, la herramienta tampoco puede.

Síntesis en FPGA

```
module voter(  
  input wire a, b, c,  
  output reg z);  
  always @(a, b, c)  
  if (a == 1)  
    if (b == 1 || c == 1)  
      z = 1;  
    else  
      z = 0;  
  else  
    if (b == 1 && c == 1)  
      z = 1;  
    else  
      z = 0;  
endmodule
```

400K
puertas
equivalentes

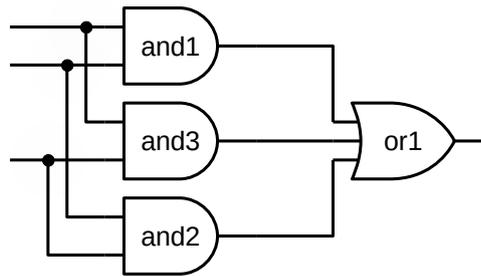


http://commons.wikimedia.org/wiki/File:Fpga_xilinx_spartan.jpg
<http://commons.wikimedia.org/wiki/File:Fpga1a.gif>

Síntesis sobre FPGA

Interconnect Resources

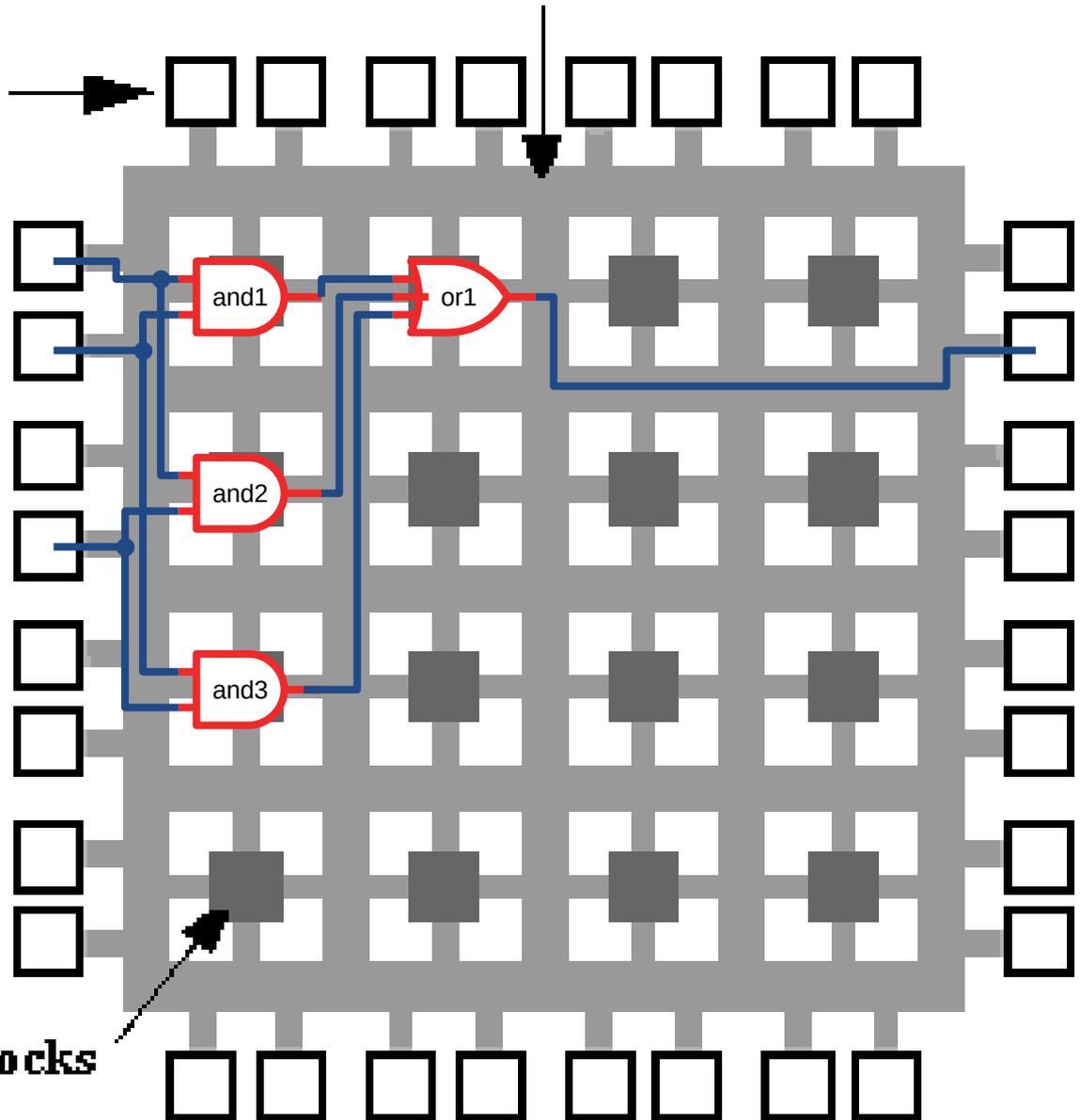
I/O Cells



Colocación
(placement)

Enrutado
(routing)

Logic Blocks



Herramientas de diseño básicas

- Editor de texto
 - Escritura de código Verilog.
- Compilador de Verilog
 - Análisis del código. Detección de errores de sintaxis.
- Simulador
 - Simulación de bancos de pruebas.
- Herramienta de síntesis
 - Implementación del circuito en una determinada tecnología.
 - Depende del suministrador de la tecnología.
 - Ejemplo: FPGA
- Entorno integrado
 - Incluye todo lo anterior.
 - A veces suministrado por el propietario de la tecnología.
 - Existen entornos integrados en la WWW.

Icarus Verilog + Gtkwave

- Icarus
 - Compilador y simulador Verilog pequeño y simple
- Gtkwave
 - Visor de formas de onda: visualización de resultados de simulación.
- Ed. texto + Icarus + Gtkwave: entorno de desarrollo Verilog básico
 - Interfaz mediante línea de comandos
 - Ligero
 - icarus (1,5MB) + Gtkwave (2,5MB) = 4MB
 - Fácil de usar
 - Software libre

<http://iverilog.icarus.com/>

Icarus Verilog en GNU/Linux

- Icarus y Gtkwave disponibles en la mayoría de distribuciones Linux
- Instalación en Debian/Ubuntu:
 - Paquetes "iverilog" y "gtkwave"
- Editor de textos
 - Vale cualquier editor de texto plano.
 - Ej. Gedit
 - Estándar en Ubuntu
 - Incluye resaltado de sintaxis de Verilog.

```
$ sudo apt-get install iverilog gtkwave  
...
```

Icarus Verilog en MS-Windows(TM)

- Busca instalador iverilog + gtkwave en www.bleyer.org
 - Importante: instalar el software en una ruta sin espacios
 - Recomendado: “c:\iverilog\”
- Abrir un terminal (cmd) y probar instalación:
 - Ejecutar “iverilog” en el terminal.
- Si no se encuentra el comando “iverilog” hay que añadir las rutas a los comandos manualmente. Buscar “agregar rutas a la variable path” para la versión de Windows adecuada. Añadir estas rutas:
 - c:\iverilog\bin
 - c:\iverilog\gtkwave\bin
- Editor de textos
 - Usar un buen editor de textos (no el Bloc de Notas -Notepad-)
 - Ej. Notepad++ (notepad-plus-plus.org), Atom, VCS, etc.

Icarus Verilog + Gtkwave sesión típica (Linux)

```
$ cd 02-1_votador1  
  
$ ls  
votador_tb.v  votador.v  
  
$ iverilog votador.v votador_tb.v  
  
$ vvp a.out  
VCD info: dumpfile votador_tb.vcd opened for output.  
  
$ ls  
a.out  votador_tb.v  votador_tb.vcd  votador.v  
  
$ gtkwave votador_tb.vcd &
```

Entornos integrados de Xilinx

- Xilinx
 - Uno de los principales fabricantes de FPGA.
- Dos entornos disponibles y parecidos
 - ISE: tradicional, modelos de FPGA “antiguos”.
 - Vivado: nuevo entorno. Sólo para modelos FPGA recientes.
- Implementación sólo sobre FPGA's de Xilinx.
- Entorno integrado incluyendo gestor de proyectos, editor de código, simulación, síntesis y mucho más.
- Completo pero complejo (no mucho) para principiantes.
- Pesado de descargar (~5GiB) e instalar (~10GiB).
- Es necesario registrarse.
- Licencias gratuitas para uso académico con limitaciones.
- Versiones para MS-Windows(TM) y GNU/Linux.

Entorno en WWW: EDA Playground

The screenshot displays the EDA Playground web interface. On the left, there is a sidebar with navigation options like 'Languages & Libraries', 'Tools & Simulators', and 'Examples'. The main area contains two code editors: 'testbench.sv' and 'design.sv'. The 'testbench.sv' editor shows Verilog testbench code for a voter circuit, including input initialization, stimulus, and simulation control. The 'design.sv' editor shows the functional Verilog code for the voter circuit. Below the code editors, there is a social sharing section with a share button and a description: 'Voter circuit (3 input). Output is 1 when two or more inputs are 1.' To the right, an EPWave window shows a timing diagram with signals 'a', 'b', 'c', and 'z' over time, with markers at 20ns, 40ns, and 60ns.

- Permite hacer y simular diseños en Verilog (y otros lenguajes).
- Varios simuladores disponibles, incluyendo Icarus Verilog.
- Incluye editor de texto y visor de formas de onda.
- Requiere registro para simular y guardar diseños.

<https://www.edaplayground.com/>

Resumen. LDH

- Descripción del comportamiento de circuitos digitales.
- Posibilidad de descripción desde varios puntos de vista: estructural, funcional, procedimental.
- Permite la simulación del sistema antes de su implementación (fabricación)
- Permite automatizar total o parcialmente el proceso de implementación usando herramientas informáticas.