

Introducción a HDL

Segundo Cuatrimestre 2022

Organización del Computador I
DC - UBA

Introducción

Sobre la clase de hoy

Hoy vamos a ver una introducción muy breve de los lenguajes de descripción de hardware. La clase se va a dividir de la siguiente manera:

- **Introducción**
- Especificación y síntesis
- Lenguajes de especificación
- Sintaxis de Verilog
- Un pequeño ejemplo

Sobre la clase de hoy

Hoy vamos a ver una introducción muy breve de los lenguajes de descripción de hardware. La clase se va a dividir de la siguiente manera:

- **Introducción**
- **Especificación y síntesis**
- Lenguajes de especificación
- Sintaxis de Verilog
- Un pequeño ejemplo

Sobre la clase de hoy

Hoy vamos a ver una introducción muy breve de los lenguajes de descripción de hardware. La clase se va a dividir de la siguiente manera:

- **Introducción**
- **Especificación y síntesis**
- **Lenguajes de especificación**
- Sintaxis de Verilog
- Un pequeño ejemplo

Sobre la clase de hoy

Hoy vamos a ver una introducción muy breve de los lenguajes de descripción de hardware. La clase se va a dividir de la siguiente manera:

- **Introducción**
- **Especificación y síntesis**
- **Lenguajes de especificación**
- **Sintaxis de Verilog**
- Un pequeño ejemplo

Sobre la clase de hoy

Hoy vamos a ver una introducción muy breve de los lenguajes de descripción de hardware. La clase se va a dividir de la siguiente manera:

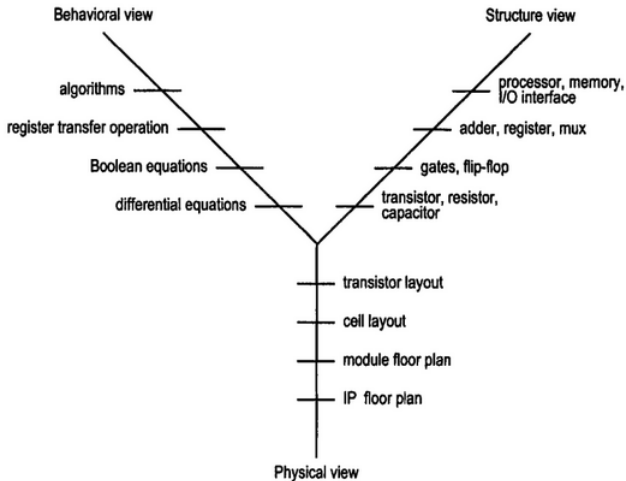
- **Introducción**
- **Especificación y síntesis**
- **Lenguajes de especificación**
- **Sintaxis de Verilog**
- **Un pequeño ejemplo**

Introducción

Tres perspectivas para construir hardware

Veremos que podemos tomar al menos tres perspectivas complementarias a la hora de construir hardware:

Tres perspectivas para construir hardware



Describiendo hardware

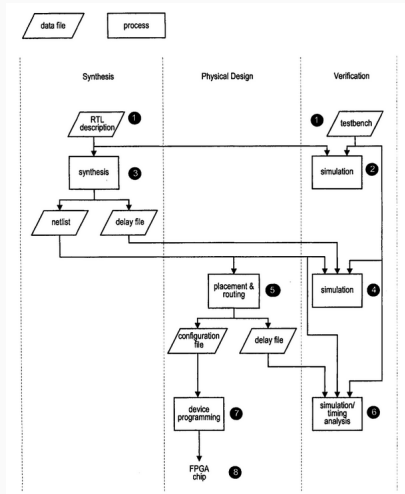
A su vez el hardware puede describirse con distintos niveles de abstracción.

	Typical blocks	Signal representation	Time representation	Behavioral description	Physical description
Transistor	transistor, resistor	voltage	continuous function	differential equation	transistor layout
Gate	and, or, xor, flip-flop	logic 0 or 1	propagation delay	Boolean equation	cell layout
RT	adder, mux, register	integer, system state	clock tick	extended FSM	RT-level floor plan
Processor	processor, memory	abstract data type	event sequence	algorithm in C	IP-level floor plan

Flujo de desarrollo

¿Cómo se ve el flujo de desarrollo o prototipado de una solución basada en hardware?

Flujo de desarrollo

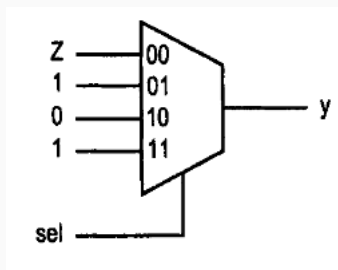


Especificación y síntesis

Primera propuesta

```
with sel select  
    y <= 'Z' when "00",  
        '1' when "01"|"11",  
        '0' when others;
```

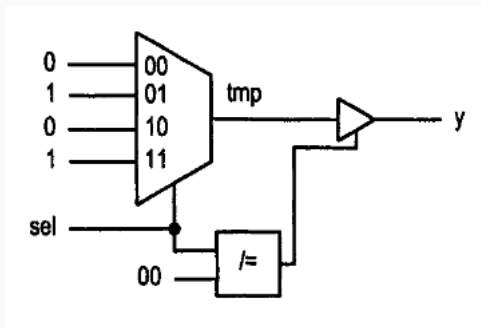
Síntesis



Segunda propuesta

```
with sel select
    tmp <= '1' when "01"|"11",
           '0' when others;
y <= tmp when sel/="00" else
    'Z';
```

Síntesis



HDL y síntesis

Los lenguajes de especificación de hardware (**HDL**) dan una descripción a nivel de transferencia de registros de un circuito (**RTL**).

HDL y síntesis

Los lenguajes de especificación de hardware (**HDL**) dan una descripción a nivel de transferencia de registros de un circuito (**RTL**).

No es, por lo tanto una descripción directa de la **implementación en términos de la lógica discreta a usar**.

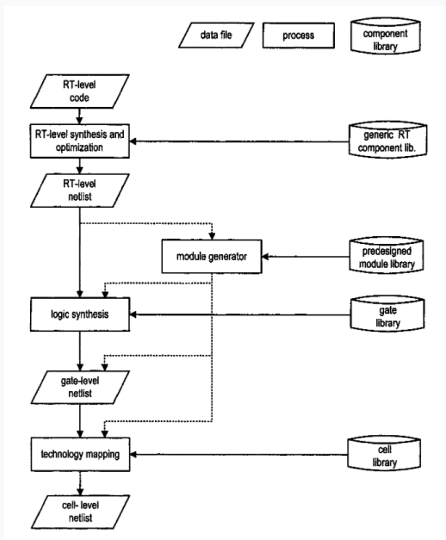
HDL y síntesis

Los lenguajes de especificación de hardware (**HDL**) dan una descripción a nivel de transferencia de registros de un circuito (**RTL**).

No es, por lo tanto una descripción directa de la **implementación en términos de la lógica discreta a usar**.

La implementación del circuito va a realizarse a través de un proceso conocido como **síntesis**.

Flujo de síntesis



Librerías de implementación

El proceso de síntesis va a *realizar* el diseño dado en la descripción de **HDL** sobre una librería de celdas específicas de la tecnología sobre la cuál vamos a construirlo.

Librerías de implementación

El proceso de síntesis va a *realizar* el diseño dado en la descripción de **HDL** sobre una librería de celdas específicas de la tecnología sobre la cuál vamos a construirlo.

Básicamente es una proyección desde los bloques funcionales, dados en a nivel de transferencia de registros a las celdas de la tecnología.

Realizabilidad

El problema de realizar un diseño es computacionalmente complejo. Esto lleva a que las soluciones dadas por la síntesis:

Realizabilidad

El problema de realizar un diseño es computacionalmente complejo. Esto lleva a que las soluciones dadas por la síntesis:

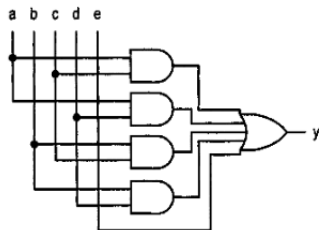
- No sean óptimas.

Realizabilidad

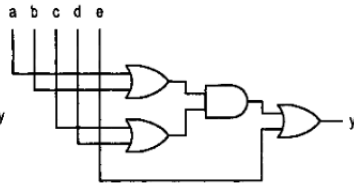
El problema de realizar un diseño es computacionalmente complejo. Esto lleva a que las soluciones dadas por la síntesis:

- No sean óptimas.
- O no sean posibles aún cuando en la simulación podamos ejecutar nuestros diseños.

Solución en dos niveles vs multi-niveles



(a) Two-level implementation






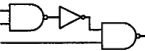

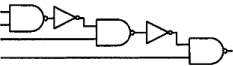

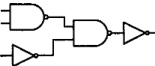

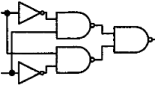


(b) Multilevel implementation

Ejemplo de una librería

Veamos cómo se ve una librería de celdas.

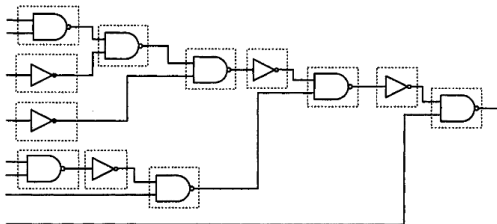
Ejemplo de una librería

cell name (cost)	symbol	nand-not representation
not (2)		
nand2 (3)		
nand3 (4)		
nand4 (5)		
aoi (4)		
xor (4)		

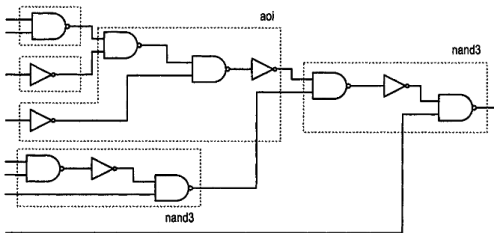
Proyecciones a la librería

Ahora veamos dos posibles proyecciones de un mismo diseño sobre las celdas de esa librería.

Proyecciones a la librería



(a) Initial mapping



(b) Better mapping

Implementando sobre FPGAs

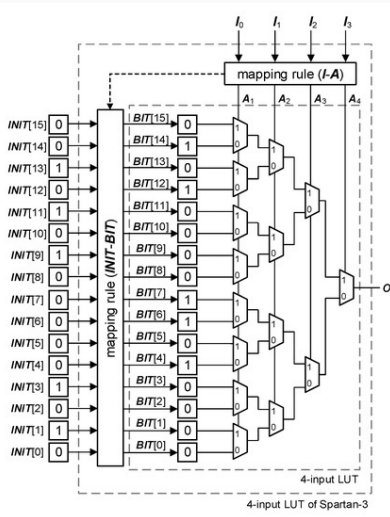
Las placas **FPGA** permiten (re)configurar lógica discreta en un soporte de hardware de forma que realice nuestros diseños dados en HDL.

Implementando sobre FPGAs

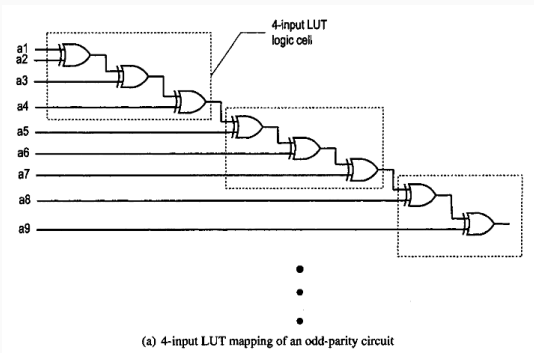
Las placas **FPGA** permiten (re)configurar lógica discreta en un soporte de hardware de forma que realice nuestros diseños dados en HDL.

El flujo de prototipado es el mismo que presentamos hasta ahora, y las celdas utilizadas en su librería suelen consistir de **LUTs** (look up tables).

LUT de 4 entradas



Proyectando a LUTs



Sobre la síntesis

El proceso de síntesis es computacionalmente complejo y depende entre otras cosas de:

Sobre la síntesis

El proceso de síntesis es computacionalmente complejo y depende entre otras cosas de:

- La complejidad del diseño inicial.

Sobre la síntesis

El proceso de síntesis es computacionalmente complejo y depende entre otras cosas de:

- La complejidad del diseño inicial.
- La librería de celdas sobre la cual queremos realizar el diseño.

Sobre la síntesis

El proceso de síntesis es computacionalmente complejo y depende entre otras cosas de:

- La complejidad del diseño inicial.
- La librería de celdas sobre la cual queremos realizar el diseño.
- La madurez y eficiencia de los algoritmos involucrados en el flujo de síntesis.

Sobre la síntesis

Debido a esto resulta vital poder escribir especificaciones que faciliten la síntesis para conseguir una solución **primero realizable, pero también eficiente.**

Lenguajes de especificación

Lenguajes de uso industrial

Actualmente existen dos lenguajes dominantes en el campo de la especificación de hardware, éstos son:

- **VHDL**

Lenguajes de uso industrial

Actualmente existen dos lenguajes dominantes en el campo de la especificación de hardware, éstos son:

- **VHDL**
- y **Verilog**.

Lenguajes de uso industrial

Actualmente existen dos lenguajes dominantes en el campo de la especificación de hardware, éstos son:

- **VHDL**
- y **Verilog**.

Para el alcance de esta introducción vamos a considerarlos equivalentes.

¿Cómo se ve una especificación de HDL (VHDL)?

```
-- (this is a VHDL comment)

-- import std_logic from the IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- this is the entity
entity name_of_entity is
    port (
        IN1 : in std_logic;
        IN2 : in std_logic;
        OUT1: out std_logic);
end entity name_of_entity;

-- here comes the architecture
architecture name_of_architecture of name_of_entity is

    -- Internal signals and components would be defined here

begin

    OUT1 <= IN1 and IN2;

end architecture name_of_architecture;
```


Sintaxis de Verilog

Verilog

Vamos a revisar la sintaxis de **Verilog**:

https://www.hdlworks.com/hdl_corner/verilog_ref/.

Verilog - Módulos

```
module Mod1(A, B, C);  
    input A, B;  
    output C;  
    assign C = A & B;  
endmodule
```

Verilog - Primitivas

```
and u1 (Q, A, B);  
and #(2.1, 2.8) u2 (Q, A, B);  
and (pull0, strong1) (Q, A, B);
```

Verilog - Instanciación

```
Dff #(4) u1 (.Clk(Clock), .D(D_In), .Q(Q_Out));  
Dff u2 (Clock, D_In, Q_Out);  
Cnt u3 (Clk, , A&&B, Q);  
Nand (weak1, pull0) #(2) u4 (Q, A, B);
```

Verilog - Declaración de puertos

```

input Clk;
output [7:0] Q;
input wire Clk;                                // Verilog-2001
output reg [7:0] Q;                            // Verilog-2001

module Cnt (output reg [7:0] Q,
            input wire Clk, Reset, Enable,
            input wire [7:0] D );              // Verilog-2001 ANSI-style

```

Verilog - Números

```
24  
1'b1  
8'hA8  
-0.5  
5.8E3  
2e-4
```

Verilog - Nets

```

wire [7:0] Data;
triereg (large) C1;
wire Q = A || B;           // continuous assignment
wire [7:0] Array [0:255][0:255][0:255]; // Verilog-2001 Multidimensional array

```


Verilog - Tipos de dato

```
reg [7:0] Data;  
integer Int;  
time Now;  
reg [15:0] Memory [0:1023];  
reg [7:0] A = 8'h3C; // Verilog-2001  
reg [7:0] Array [0:255][0:255][0:255]; // Verilog-2001
```

Verilog - Always

```
always #10 Clk = !Clk;

always @(posedge Clk or negedge Reset)
begin
    if (!Reset)
        Q <= 0;
    else
        Q <= D;
end
```

Verilog - Case

```
case (Addr)
  0 : Q <= 1;
  1 : begin
      Q <= 1;
      R <= 0;
    end
  2, 3 : R <= 1;
  default : $display("Illegal Addr value", Addr);
endcase
```

```
casez (Opcode)
  2'b1?? : Q <= 2'b01;
  2'b000 : Q <= 2'b00;
  2'b10? : Q <= 2'b10;
  default : Q <= 2'bxx;
endcase
```

Ejemplo para probar

Simulación online

En esta página van a poder ver algunos ejemplos siendo simulados online:

<https://8bitworkshop.com/v3.10.0/?platform=verilog>.

Cierre

Sobre la clase de hoy

Hoy vimos:

- **Introducción a HDL**

Sobre la clase de hoy

Hoy vimos:

- **Introducción a HDL**
- **Especificación y síntesis**

Sobre la clase de hoy

Hoy vimos:

- **Introducción a HDL**
- **Especificación y síntesis**
- **Lenguajes de especificación**

Sobre la clase de hoy

Hoy vimos:

- **Introducción a HDL**
- **Especificación y síntesis**
- **Lenguajes de especificación**
- **Sintaxis de Verilog**

Sobre la clase de hoy

Hoy vimos:

- **Introducción a HDL**
- **Especificación y síntesis**
- **Lenguajes de especificación**
- **Sintaxis de Verilog**

Estén a la espera de noticias porque es muy probable que el primer cuatrimestre de 2023 dictemos una materia de diseño de hardware con VHDL sobre FPGAs con Marcos Cervetto y Egardo Marchi.

Preguntas
