

Tema 2.1. Primeros pasos VHDL

1. Introducción al lenguaje VHDL
2. Estilos de codificación
3. Unidades de diseño en VHDL
 - a. Entity
 - b. Architecture
 - c. Libraries and Packages
 - d. Tipos de objetos: señales

2.1.1. VHDL: Introducción

- VHDL es un lenguaje de descripción de hardware (HDL)
- Otros lenguajes HDL son: Verilog, ABEL, SystemC, etc.
- VHDL permite modelar circuitos digitales y ser programados en FPGAs
- Desarrollado en 1987 por el departamento de defensa de USA
- Adoptado como estándar por el IEEE: revisiones 1987, 1993, 2000, 2002 y 2008.
- Para saber mas....

<http://en.wikipedia.org/wiki/VHDL>

2.1.1. VHDL: Introducción

Consideraciones en líneas generales

1. Es un lenguaje normalizado (no es un lenguaje propietario), por lo que es compatible con la práctica totalidad de herramientas de diseño disponibles en el mercado.
2. NO es sensible a las mayúsculas (NO *case-sensitive*).
3. No es sensible a espacios.
4. Es importante incluir comentarios para conseguir mejor comprensión del código (“- -”).
5. Los paréntesis ayudan a una mayor legibilidad del código
6. Es un lenguaje fuertemente “*tipado*”.

2.1.1. VHDL: Introducción

Consideraciones en líneas generales

7. Toda sentencia termina con “;” (*semicolon*)
8. Los identificadores deben ser autoexplicativos
9. Existen palabras reservadas de VHDL
10. Si se trata de un diseño complejo, divídelo en bloques más simples.
11. Incluir encabezado con información del módulo

2.1.2. VHDL: Estilo de codificación

1. Comentar el código.

- Para comentar líneas en VHDL usar “ - -” al principio de cada línea.
- Incluir los comentarios antes del código a comentar.
- Si el comentario es corto, es posible incluirlo al final de la sentencia.

Código VHDL	Posibles comentarios	¿Es un buen comentario?
<code>Cuenta<=Numero_Cestas</code>	Número de cestas = 5	Aporta información sobre el valor inicial de Cuenta
	Cuenta es igual al número de cestas.	No aporta nada
	<code>Cuenta<=Numero_Cestas</code>	Sin comentarios al comentario

2.1.2. VHDL: Estilo de codificación

!!!BUENAS PRÁCTICAS DE CODIFICACIÓN!!!

2. Indentar y usar espacios para alinear grupo de código

No indentado	Indentado
<pre>If cuenta="100" then Cuenta='1'; Else Cuenta='0'; End if;</pre>	<pre>If cuenta="100" then Cuenta='1'; Else Cuenta='0'; End if;</pre>

2.1.2. VHDL: Estilo de codificación

3. Usar nombres descriptivos breves para los **identificadores**

	No es descriptivo	Sí es descriptivo
Señal Reloj de 40MHZ	c40	Clock_40MHz
	r40	Clk_40MHz

4. Convenir un formato para los **identificadores** (inglés)

Palabras clave en minúscula	and, end, if, etc
Entradas, salidas y señales: <ul style="list-style-type: none">- Palabras en minúscula- Solo una letra o iniciales en mayúsculas- Varias palabras:<ul style="list-style-type: none">* Separa palabras con “_”* Iniciales en mayúsculas	Data A, B, LED <u>switch_on_LED</u> SwitchOnLED switchOnLed (Camel)
Nombre de entidad y arquitectura en minúsculas	Entity mux2_nbits architecture behavioral

2.1.2. VHDL: Estilo de codificación

```
1  -- library declaration
2  library IEEE;
3  use IEEE.std_logic_1164.all; -- basic IEEE library
4  use IEEE.numeric_std.all;    -- IEEE library for the unsigned type and
5                                -- various arithmetic operators
6
7  -- WARNING: in general try NOT to use the following libraries
8  --           because they are not IEEE standard libraries
9  -- use IEEE.std_logic_arith.all;
10 -- use IEEE.std_logic_unsigned.all;
11 -- use IEEE.std_logic_signed
12
13 -- entity
14 entity my_ent is
15     port ( A,B,C : in  std_logic;
16           F      : out std_logic);
17 end my_ent;
18 -- architecture
19 architecture my_arch of my_ent is
20     signal v1,v2 : std_logic_vector (3 downto 0);
21     signal u1    : unsigned (3 downto 0);
22     signal i1    : integer;
23 begin
24     u1 <= "1101";
25     i1 <= 13;
26     v1 <= std_logic_vector(u1); -- = "1101"
27     v2 <= std_logic_vector(to_unsigned(i1, v2'length)); -- = "1101"
28
29 -- "4" could be used instead of "v2'length", but the "length"
30 -- attribute makes life easier if you want to change the size of v2
31
32     F <= NOT (A AND B AND C);
33 end my_arch;
```


2.1.2. VHDL: Estilo de codificación

Listing 2.7: A short list of VHDL reserved words.

access	after	alias	all	attribute	block
body	buffer	bus	constant	exit	file
for	function	generic	group	in	is
label	loop	mod	new	next	null
of	on	open	out	range	rem
return	signal	shared	then	to	type
until	use	variable	wait	while	with

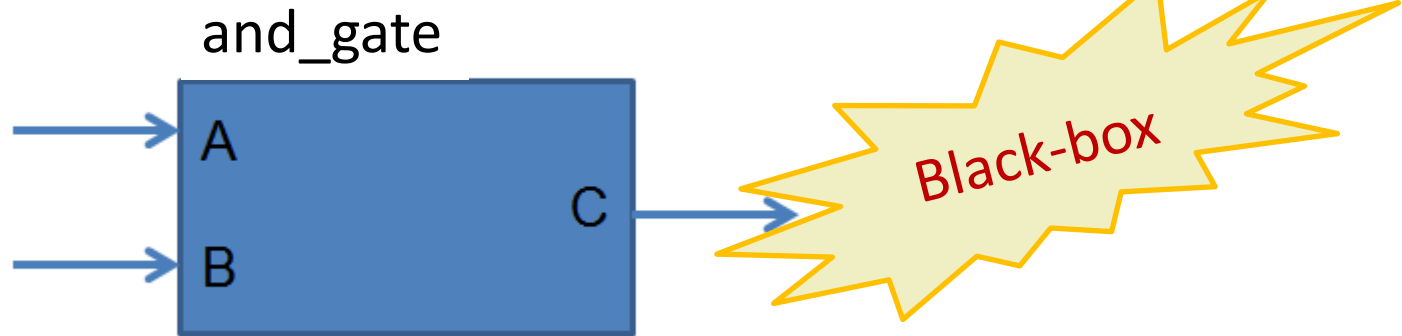
2.1.3. VHDL: Unidades de diseño

ENTITY

- VHDL emplea un enfoque de “caja negra” o “diagrama de bloques”
- Ventajas:
 - El diagrama del circuito es más fácil de entender
 - Cada bloque del diagrama es reutilizable en otro circuito
- Mas tarde se detallan los elementos dentro del bloque
- Un bloque se corresponde con la entidad (*Entity*)
- La descripción del comportamiento/contenido del bloque se corresponde con la arquitectura (*Architecture*).

2.1.3. VHDL: Unidades de diseño

ENTITY



- Describe el interfaz entre el módulo y el exterior (u otro módulo)
- Debe incluir:

- **Nombre** de la *Entity*

- Nombre de los **Port**

- **Modo** del *Port*:

- In

- Out

- Inout

```
entity and_gate is
  Port ( A : in  STD_LOGIC;
         B : in  STD_LOGIC;
         C : out STD_LOGIC);
end and_gate;
```


- **Tipo**: establece el tipo de dato para los “ports”

2.1.3. VHDL: Unidades de diseño

ENTITY

```
entity and_gate is
    Port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          C : out  STD_LOGIC);
end and_gate;
```

```
entity and_gate is
    Port ( A, B : in  STD_LOGIC;
          C    : out  STD_LOGIC);
end and_gate;
```



2.1.3. VHDL: Unidades de diseño

ARCHITECTURE

Describe el **comportamiento** que debe tener el circuito.

- Para una misma entidad pueden describirse distintas **arquitecturas equivalentes**

```
architecture behavioral of and_gate is
-- zona de declaraciones
begin
-- Cuerpo de la arquitectura
    C <= A and B;
end behavioral;
```

2.1.3. VHDL: Unidades de diseño

ARCHITECTURE

Comprende dos zonas:

- ❑ Descripción de **elementos** : señales, constantes, otros módulos, etc
- ❑ Descripción del **comportamiento** que debe tener el circuito, elementos que lo formarán.

```
architecture behavioral of Modulo is
-- zona de declaraciones
begin
-- Cuerpo de la arquitectura
end behavioral;
```

2.1.3. VHDL: Unidades de diseño

ARCHITECTURE

- El estilo de la descripción de la arquitectura influye en el **sintetizador**
 - Existen tres técnicas para modelar una arquitectura:
 - *Data-flow or RTL model*
 - *Behavioral model*
 - *Structural model*
- Son los que usaremos
- Es habitual combinar varias técnicas de modelado

2.1.3. VHDL: Unidades de diseño

ARCHITECTURE

Estructural	Comportamental
Codificación de bajo nivel (esquematicos)	Codificación de mayor nivel
Instanciación de otros bloques: primitivas o módulos	Estructuras de programación: bucle, if, case, etc
Requiere mucho tiempo de diseño	Menor tiempo de diseño
Interesante para modelar el módulo de mayor jerarquía en un diseño constituido por varios bloques	

2.1.3. VHDL: Unidades de diseño

TOP_Module

Entity TOP

Architecture TOP (**Estructural**)

Entity 01

Archit.
(Behavioral)

Module_01

Entity 02

Archit.
(Behavioral)

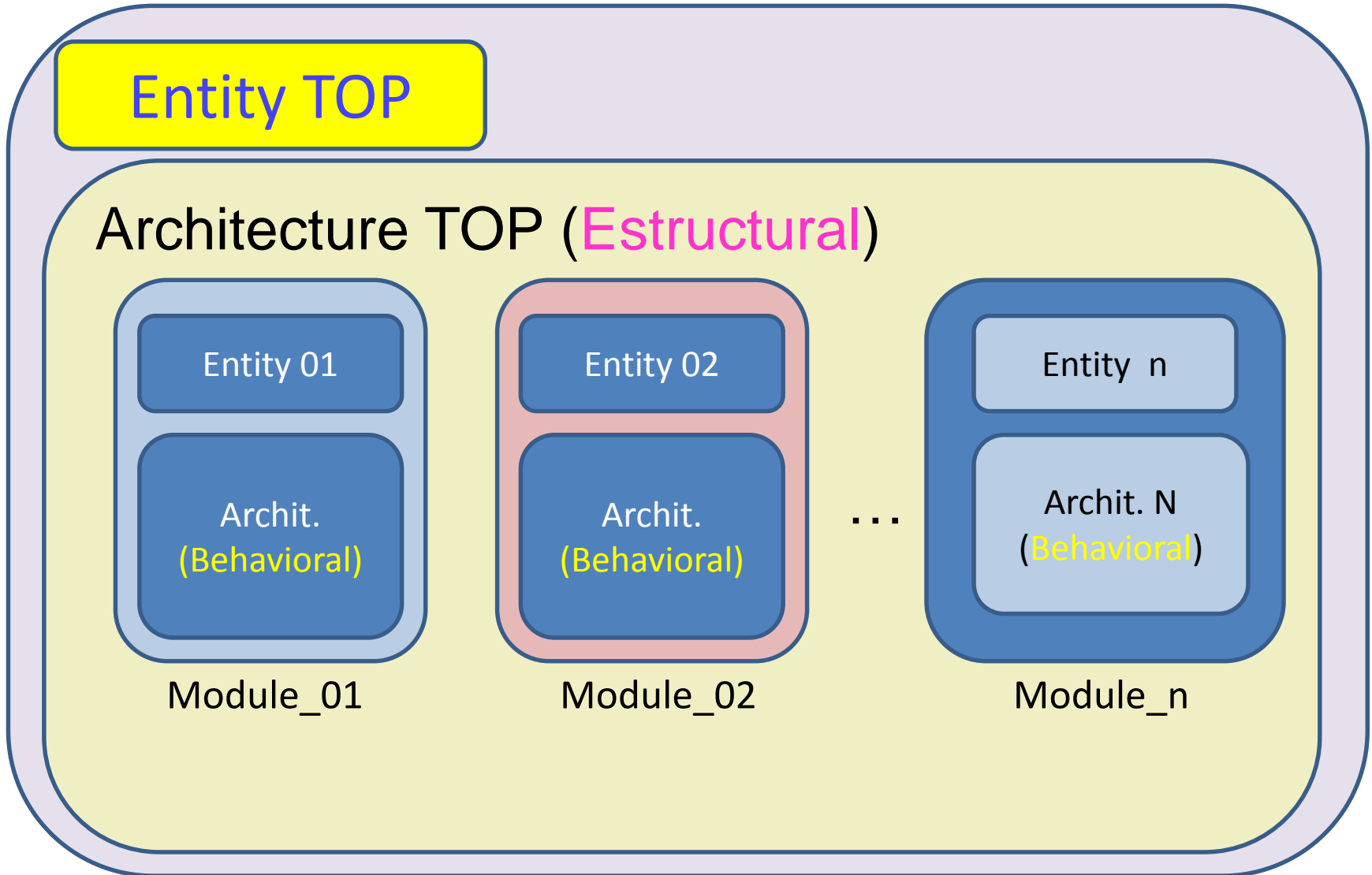
Module_02

...

Entity n

Archit. N
(Behavioral)

Module_n



2.1.3. VHDL: Unidades de diseño

Libraries and Packages

Una biblioteca (*library*) es un conjunto de *packages* que contienen a su vez definiciones de tipos de datos, funciones, componentes, etc.

Las *libraries* y los *packages* pueden ser creados por el usuario, ofrecidas por terceras compañías o establecidas por el IEEE (**librerías estándar**).

```
1 library ieee;  
2 use ieee.std_logic_1164.all;
```

- Línea 1: Invoca a una **library** llamada “**ieee**”
- Línea 2: la palabras **use** y **all** hacen el *package* **std_logic_1164** visible al completo a las unidades de diseño. Este *package* es necesario si se quieren utilizar los tipos de datos **std_logic** y **std_logic_vector**.

2.1.3. VHDL: Unidades de diseño

Libraries and Packages

```
1 -- library declaration
2 library IEEE;
3 use IEEE.std_logic_1164.all; -- basic IEEE library
4 use IEEE.numeric_std.all;    -- IEEE library for the unsigned type and
5                               -- various arithmetic operators
6
7 -- WARNING: in general try NOT to use the following libraries
8 --           because they are not IEEE standard libraries
9 -- use IEEE.std_logic_arith.all;
10 -- use IEEE.std_logic_unsigned.all;
11 -- use IEEE.std_logic_signed
```

Libraries implícitas

- “work” es la “library” donde se compila el diseño → Ver en ISEWebpack
- “std” es una “library” siempre visible → No es necesario invocarlas explícitamente (incluye tipo “bit”).

2.1.3. VHDL: Unidades de diseño

Tipos de Objetos: señales

- ❑ Tipo de objeto que más se usa en VHDL
- ❑ Son el principal método para mover información dentro de la FPGA.
- ❑ Se declaran dentro de una arquitectura, no se “conectan” con el exterior → No se declaran en la entidad
- ❑ Tras la síntesis las señales se convierten en “cables”.
- ❑ VHDL es fuertemente tipado , cada señal debe ser declarada como de un determinado tipo antes de ser usada. (p.e.: `std_logic`)

2.1.3. VHDL: Unidades de diseño

Tipos de Objetos: señales

`signal X,Y: std_logic := '0';`

Palabra clave

Lista de nombres

Tipo de dato

Valor inicial

```
architecture Behavioral of prueba is
    signal X,Y: std_logic;
begin
    --
end Behavioral;
```

2.1.3. VHDL: Unidades de diseño

Tipos de Objetos: señales

$$F3 = \overline{L} \cdot \overline{M} \cdot N + L \cdot M$$



```
-- entity
entity my_ckt_f3 is
port ( L,M,N : in  std_logic;
      F3      : out std_logic);
end my_ckt_f3;
-- architecture
architecture f3_2 of my_ckt_f3 is
begin
  F3<= ( (NOT L) AND (NOT M) AND N) OR (L AND M);
end f3_2;
```

1

```
-- entity
entity my_ckt_f3 is
port ( L,M,N : in  std_logic;
      F3      : out std_logic);
end my_ckt_f3;
-- architecture
architecture f3_1 of my_ckt_f3 is
  signal A1, A2 : std_logic; -- intermediate signals
begin
  A1 <= ((NOT L) AND (NOT M) AND N);
  A2 <= L AND M;
  F3 <= A1 OR A2;
end f3_1;
```

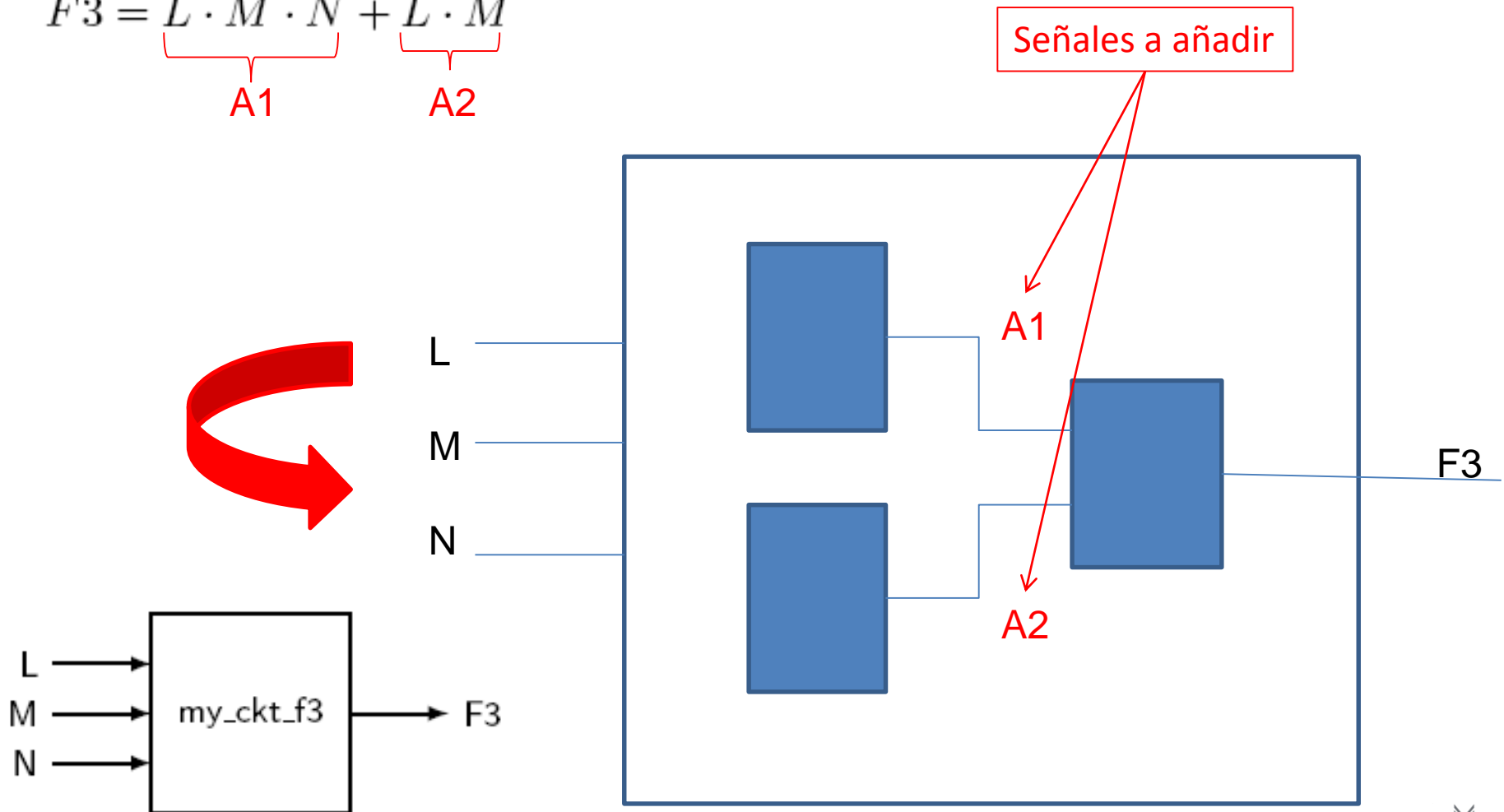
2

Estilo
Dataflow

2.1.3. VHDL: Unidades de diseño

Tipos de Objetos: señales

$$F3 = \underbrace{\overline{L} \cdot \overline{M} \cdot N}_{A1} + \underbrace{L \cdot M}_{A2}$$



2.1. VHDL: Bibliografía

- **Free range VHDL.** Bryan Mealy, Fabrizio Tappero. (Creative Commons). <http://www.freerangefactory.org> (Mayo 2013)
- **VHDL 101.** William Kfig. Editorial Elsevier. 2011