Clase 1:

Estructura de un diseño en VHDL

1.1 Introducción y ejemplo simple

1.2 La sección ENTITY

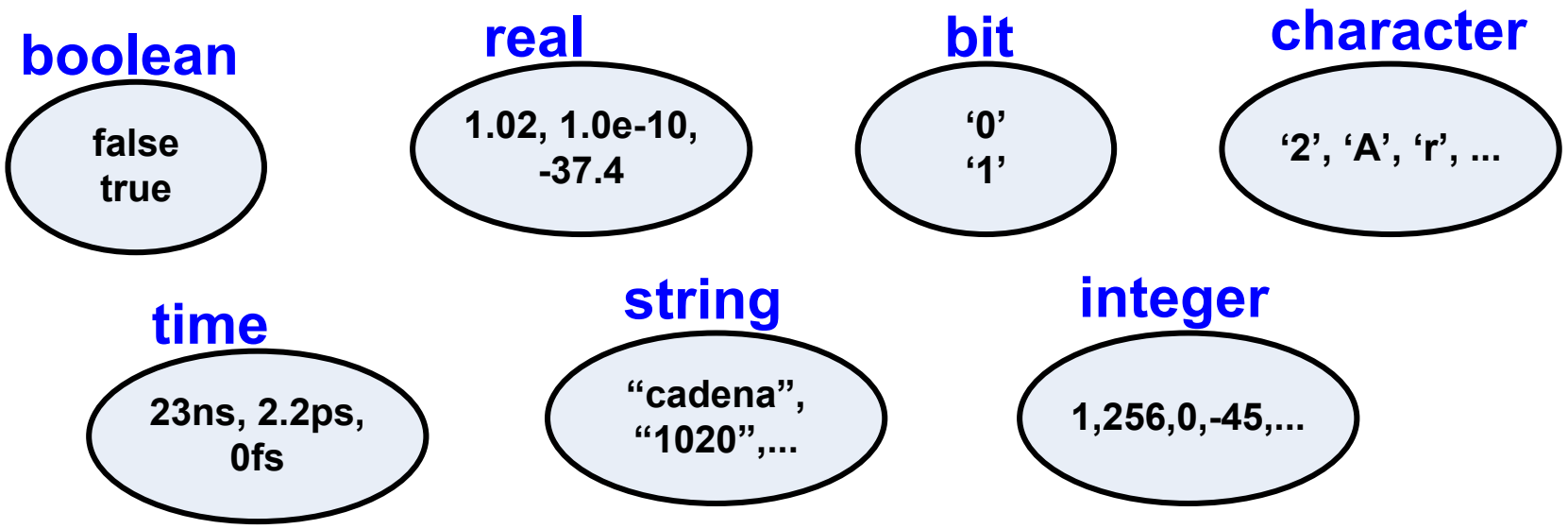
1.3 La sección LIBRARY

1.4 La sección ARCHITECTURE

1.5 La sección CONFIGURATION

Tipos de datos estándar

- En propio lenguaje VHDL define algunos tipos de datos estándar:
  - Todos los tipos de datos pueden adoptar unos valores determinados.
  - El usuario puede definir sus propios tipos.



Tipos de datos estándar. Tipo ***bit***

- Puede tomar sólo valores ‘0’ y ‘1’.
- Para síntesis y simulación necesitamos otros posibles valores:
  - No inicializado
  - Alta impedancia
  - No definido
  - No importa
  - Valores débiles (pull-up).
- Para disponer de estos datos multievaluado necesitamos definir un nuevo *tipo de dato*.
- En la librería IEEE 1164 (STD_LOGIC_1164) se definen datos multievaluados.

IEEE Standard Logic 1164

- Las librerías en VHDL son conjuntos de:
  - Definiciones de tipos de datos.
  - Funciones aritméticas, de conversión y comparaciones.

⑤ Todos nuestros diseños comenzarán por:

```
library IEEE;
  use IEEE.std_logic_1164.all;
```

```
  Opcionalmente
  use IEEE.numeric_std.all;
```

IEEE Standard Logic 1164

- Paquete básico de la librería IEEE
- **USE** IEEE.std_logic_1164.**all**
  - std_logic
  - std_logic_vector(MSB **downto** LSB)
  - integer
  - Lógica booleana (NOT, XOR, etc)

```
.....
type STD_ULOGIC is (
    ` U `,    -- uninitialized
    ` X `,    -- strong 0 or 1 (= unknown)
    ` 0 `,    -- strong 0
    ` 1 `,    -- strong 1
    ` Z `,    -- high impedance
    ` W `,   -- weak 0 or 1 (= unknown)
    ` L `,    -- weak 0
    ` H `,    -- weak 1
    ` - `,    -- don`t care);
.....
```

Dentro de la librería podemos encontrar la definición del tipo std_ulogic (std_logic es un caso particular de std_ulogic).

Operadores en VHDL y std_logic_vector_1164

Operador	Descripción	Tipo de dato de operandos	Tipo de datos de resultados
a ** b	Elevado a	Integer	Integer
a*b	multiplicación		
a/b	división		
a+b	Suma		
a-b	Resta		
a & b	Concatenación	1-D array	1-D array
a = b	igual	cualquiera	boolean
a /= b	distinto		
a < b	Menor que	Integer	boolean
a <= b	Menor o igual		
a > b	Mayor que		
a >= n	Mayor o igual		
not a	negación	Boolean, std_logic, std_logic_vector	El mismo tipo que el operando
a and b	and		
a or b	or		
a xor b	xor		

IEEE Standard Logic Numeric

```
use IEEE.numeric_std.all;
```

- Introduce buses con significado numérico: **SIGNED** y **UNSIGNED.**
- Ejemplo: **"1001"**:
  - STD_LOGIC_VECTOR(3 downto 0): Es simplemente grupo de bits sin significado numerico.
  - SIGNED(3 downto 0): Un número en "complemento a 2" representa un **-7**
  - UNSIGNED(3 downto 0): Un número sin signo, que representa un **9.**

IEEE Standard Logic Numeric: Sobrecarga de operadores

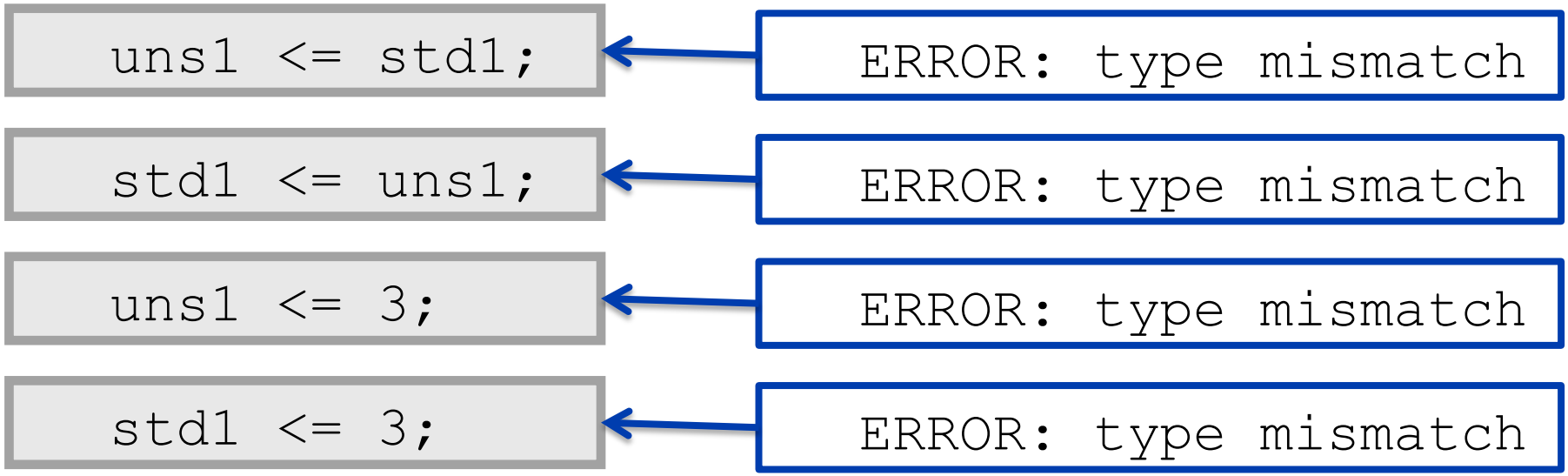
Operador	Descripción	Tipo de dato de operandos	Tipo de dato de resultado
a*b	Operadores aritméticos	Unsigned, natural, signed, integer	Unsigned, signed
a+b			
a-b			
a=b	Comparaciones	Unsigned, natural, signed, integer	boolean
a/=b			
a<b			
a<=b			
a>b			
a>=b			

Conversiones entre std_logic_vector y tipos numéricos

Tipo de dato origen (a)	Tipo de dato destino	Función de conversión
Unsigned, signed	Std_logic_vector	Std_logic_vector(a)
Signed, std_logic_vector	unsigned	Unsigned(a)
Unsigned, std_logic_vector	signed	Signed(a)
Unsigned, signed	integer	To_integer(a)
natural	unsigned	To_unsigned(a,size)
integer	signed	To_signed(a,size)

Ejemplo de asignaciones y operaciones aritméticas

```
signal std1: std_logic_vector(3 downto 0);
Signal uns1: unsigned(3 downto 0);
```



Ejemplo de asignaciones y operaciones aritméticas

```
signal std1: std_logic_vector(3 downto 0);
Signal uns1: unsigned(3 downto 0);
```

```
uns1 <= unsigned(std1);
```

OK

```
std1 <= std_logic_vector(uns1);
```

OK

```
uns1 <= to_unsigned(3,4);
```

OK

```
std1 <= std_logic_vector(to_unsigned(3,4));
```

OK

Ejemplo de asignaciones y operaciones aritméticas

```
signal std1, std2, std3: std_logic_vector(3 downto 0);
Signal uns1, uns2, uns3: unsigned(3 downto 0);
```

```
uns1 <= uns2+uns3;
```

OK. El mismo tipo

```
uns1 <= uns2+3;
```

OK: Sobrecarga de operadores

```
std1 <= std2+std3;
```

```
std1 <= std2+1;
```

ERROR: Para sumar hace falta un significado numérico

OK: Usamos funciones de conversión

```
std1 <= std_logic_vector(unsiged(std2)+unsigned(std3));
```

```
std1 <= std_logic_vector(unsiged(std2)+3);
```