Unidades de Diseño

Fernando Rincón (frincon@inf-cr.uclm.es)

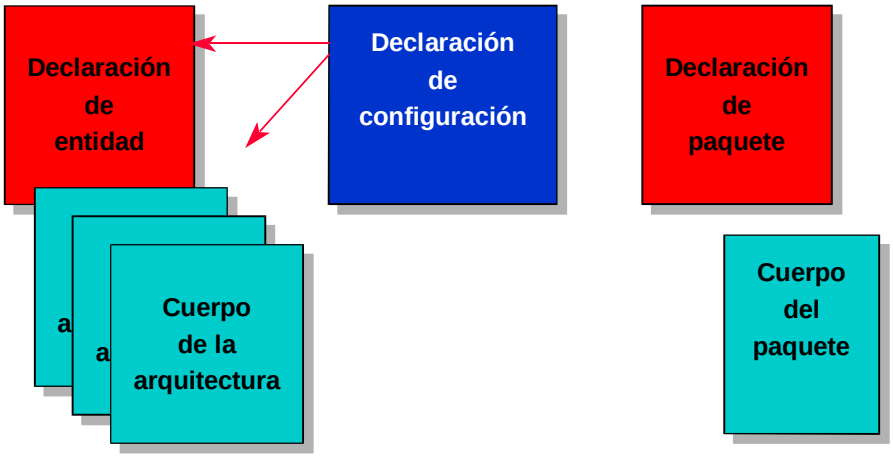
Arquitectura y Redes de Computadores (arco.inf-cr.uclm.es)
Escuela Superior de Informática (www.inf-cr.uclm.es)
Universidad de Castilla-La Mancha (www.uclm.es)

Contenido

- Tipos de unidades
  - Entidad
  - Arquitectura
  - Configuración
  - Paquetes
- Bibliotecas

Tipos

- 5 tipos de unidades



Entidad

- Describe la interfaz del modelo con el exterior
- Oculta la implementación (arquitectura)
- Sintaxis:

```
entity <id> is
  [<genéricos>];
  [<puertos>];

[<declaraciones>];
[begin
<sentencias>];
end [entity]
[<id>];
```

Nombre del dispositivo
parámetros
interfaz con el sistema
declaraciones comunes a las arquitecturas
sentencias comunes a las arquitecturas

Entidad


```
entity MUX21 is
   port( A    : in  bit;
         B    : in  bit;
         Ctrl : in  bit;
         Z    : out bit;
end MUX21;
```

Puertos

Tipos de puertos

Modo	Dirección
In	Entrada
out	Salida
inout	Bidireccional
buffer	Salida

Tipos de datos habituales

integer:	entero
bit:	bit (0/1)
bit_vector:	vector de bits
std_logic:	bit (0/1/Z/X ...)
std_logic_vector:	vector de std_logic

Recomendados

Arquitectura

- Describe la funcionalidad de la entidad que representa
- Pueden existir múltiples arquitecturas para la misma entidad
- Sintaxis:

```
Architecture <id> of <id_entidad> is
  [<declaraciones>];
begin
  <sentencias concurrentes>;
end [architecture] [<id>] ;
```

nombre de la arquitectura

nombre de la entidad

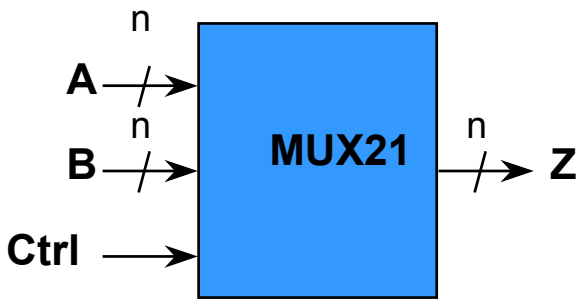
señales, componentes, otras

Sentencias concurrentes:

- asignaciones concurrentes
- referencia a componentes
- bloques
- procesos

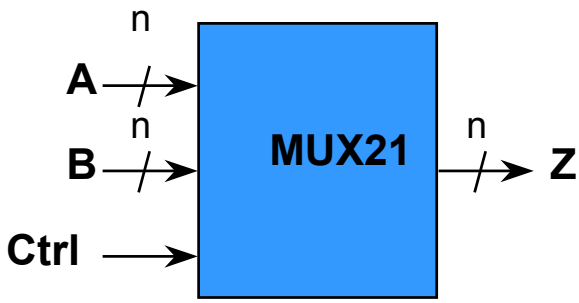
Arquitectura: modelo funcional

```
entity MUX21 is
  generic ( n: natural);
  port( A    : in  bit_vector(n-1 downto 0);
        B    : in  bit_vector(n-1 downto 0);
        Ctrl : in  bit;
        Z    : out bit_vector(n-1 downto 0));
end MUX21;
```

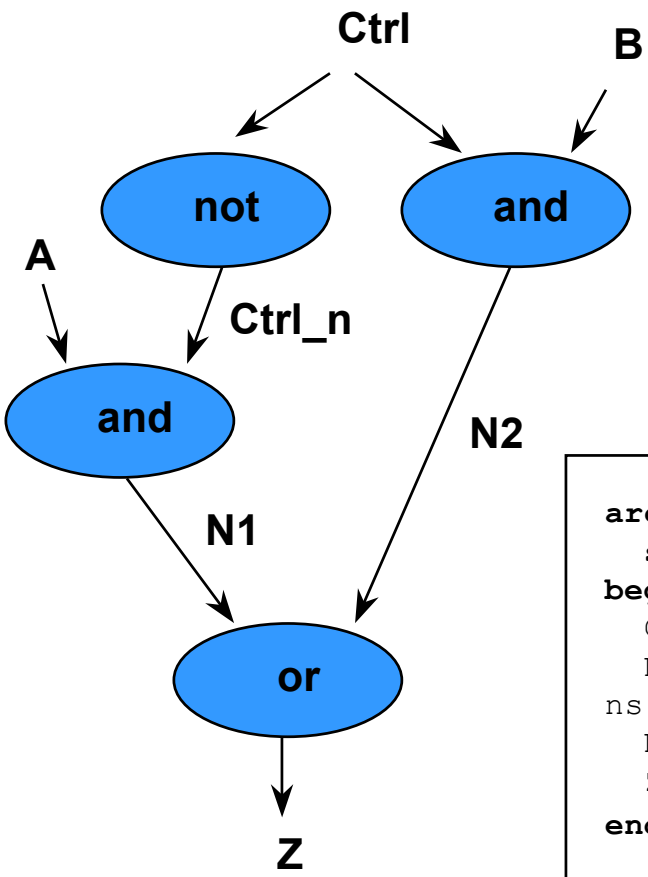



```
architecture Funcional of MUX21 is
begin
  process(A, B, Ctrl)
  begin
    if Ctrl = '0' then
      Z <= A;
    else
      Z <= B;
    end if;
  end process;
end Funcional;
```

Arquitectura: modelo flujo de datos

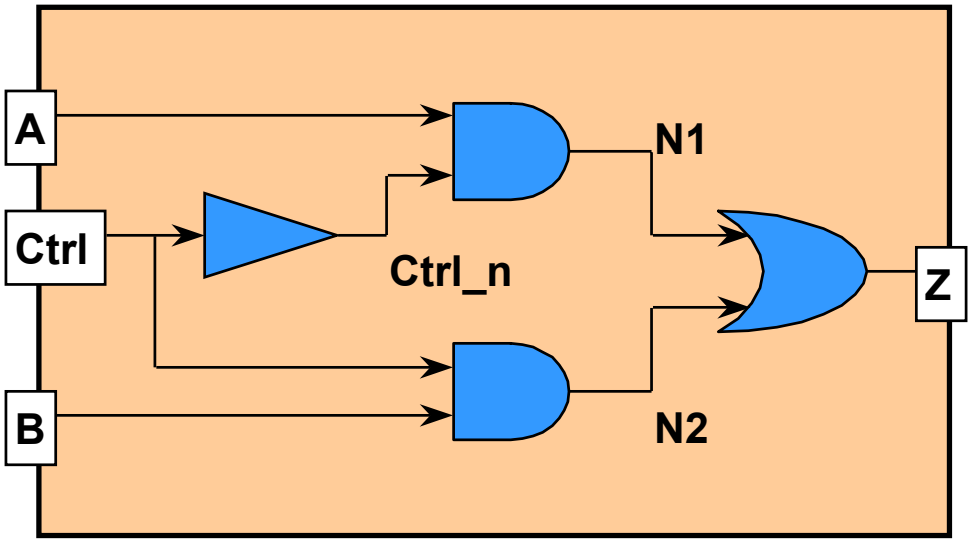



```
entity MUX21 is
port( A, B, Ctrl : in bit;
      Z: out bit);
end MUX21;
```




```
architecture FlujoDatos of MUX21 is
  signal Ctrl_n, N1, N2 : bit;
begin
  Ctrl_n <= not Ctrl after 1 ns;
  N1     <= Ctrl_n and a  after 2
ns;
  N2     <= Ctrl and b after 2 ns;
  Z      <= (N1 or N2) after 2 ns;
end FlujoDatos;
```

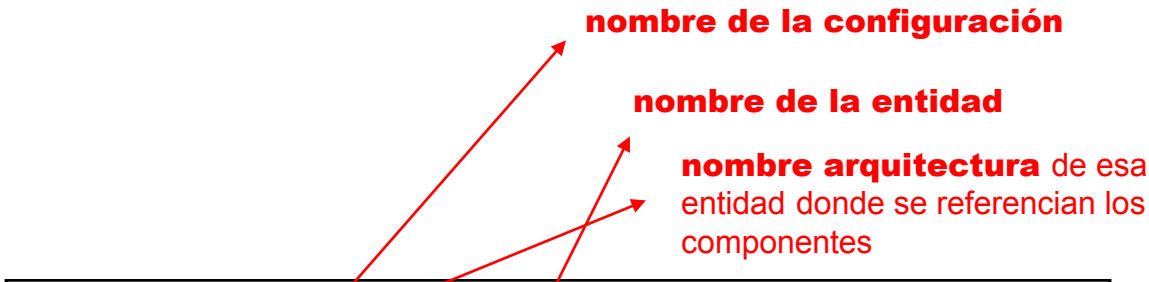
Arquitectura: modelo estructural




```
architecture estructural of MUX21 is
  signal Ctrl_n, N1, N2 : bit;
  component INV
    port( Y : in  bit;
          Z : out bit);
  end component;
  component AND2
    port( X, Y : in  bit;
          Z    : out bit);
  end component;
  component OR2
    port( X, Y : in  bit;
          Z    : out bit);
  end component;
begin
  U0: INV  port map (Ctrl, Ctr_n);
  U1: AND2 port map (Ctrl_n, A, N1);
  U2: AND2 port map (Ctrl, B, N2);
  U3: OR2  port map (N1, N2, Z);
end estructural;
```

Configuración

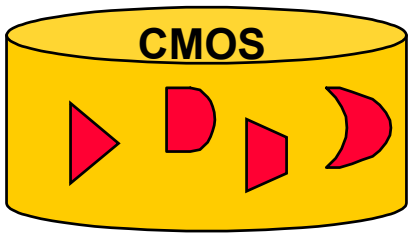
- Determina qué arquitectura se elige para cada entidad *instanciada* (en una arquitectura estructural)
- Sintaxis

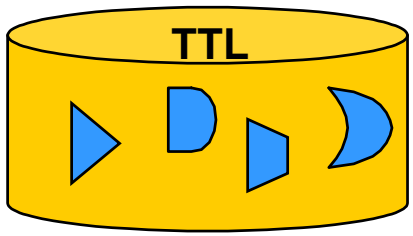



```
configuration <id> of <id_entidad> is
   for <id_arq>
      {for ref_comp {, ...} | others | all: <id_comp>
         use entity <id_entidad>[(<id_arq>); |
         use configuration <id_conf>];
      end for;}
   end for;
end [configuration] [<id>] ;
```

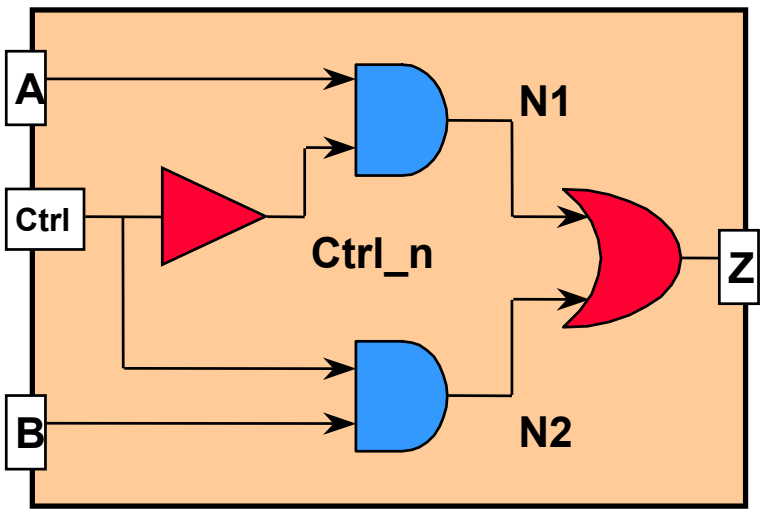
componente referenciado

Configuración






```
library CMOS, TTL;
use CMOS.all, TTL.all;
```




```
configuration mux21_cfg of MUX21 is
  for estructural
    for  U0 : INV
       use entity CMOS.INV(Funcional);
    for all : AND2
       use entity TTL.AND2(Funcional);
    for  U3 : OR2
       use entity CMOS.OR2(Funcional);
  end for;
end mux21_cfg;
```

Paquete

- Contiene declaraciones y funciones utilizadas por varios modelos
- Facilita la reutilización del código
- Dos partes:
  - Declaración del paquete: oculta la implementación
  - Cuerpo del paquete

Paquete

- Declaración de paquete:

```
package <identificador>
  [<declaraciones>];
end [package] [<identificador>]
```

nombre del paquete

declaración de constantes, funciones y procedimientos

- Cuerpo del paquete

```
package body <identificador>
  [<cuerpo de declaraciones>];
end [package body] [<identificador>]
```

nombre del paquete

asignación de constantes e implementación de funciones y procedimientos

Paquete

```
package Retardos is
  constant RetNOT  : time;
  constant RetAND2 : time;
  constant RetOR2  : time;
end Retardos;
```

```
Package body Retardos is
  constant RetNOT  : time := 1 ns;
  constant RetAND2 : time := 2 ns;
  constant RetOR2  : time := 2 ns;
end Retardos;
```

Entidad

Nombre del paquete

Declaración de constantes

Asignación de constantes

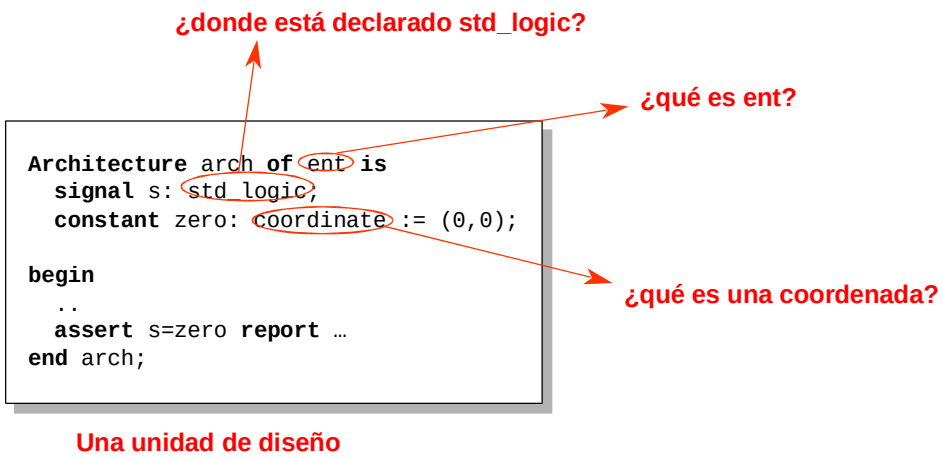
Arquitectura

```
architecture FlujoDatos of MUX21 is
  signal Ctrl_n, N1, N2 : bit;
begin
  Ctrl_n <= not Ctrl after Bibl.Retardos.RetNOT;
  N1     <= Ctrl_n and a  after Bibl.Retardos.RetAND2;
  N2     <= Ctrl and b after Bibl.Retardos.RetAND2;
  Z      <= (N1 or N2) after Bibl.Retardos.RetOR2;
end FlujoDatos;
```

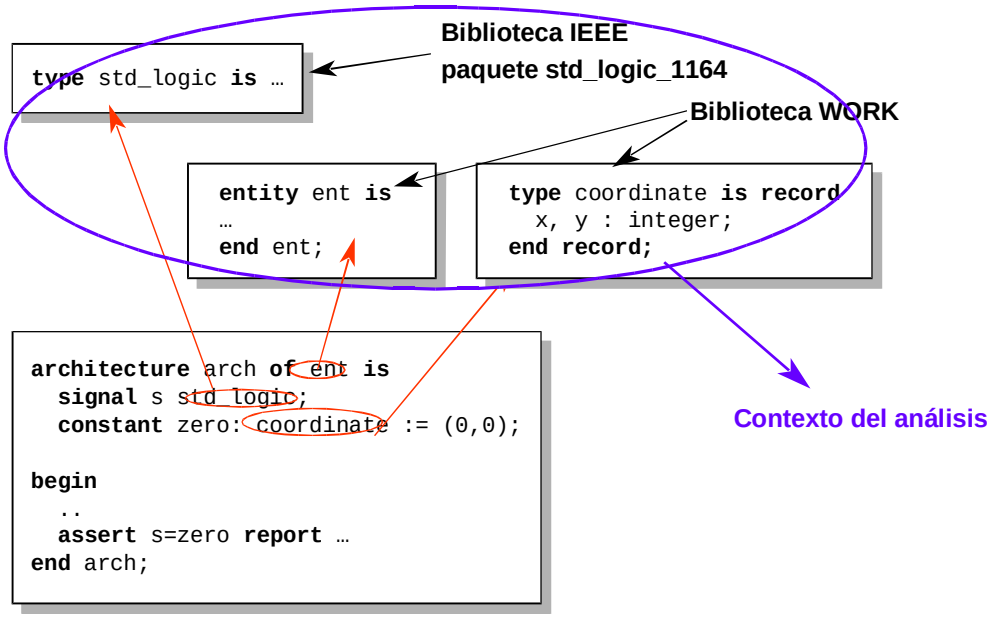
Paquetes STANDARD y TEXTIO

- Los paquetes STANDARD y TEXTIO forman parte del lenguaje
  - sus tipos de datos y funciones pueden utilizarse directamente
- STANDARD
  - tipos de datos predefinidos: boolean, bit, bit_vector, integer, natural, ...
- TEXTIO
  - tipos de datos y funciones para uso de ficheros

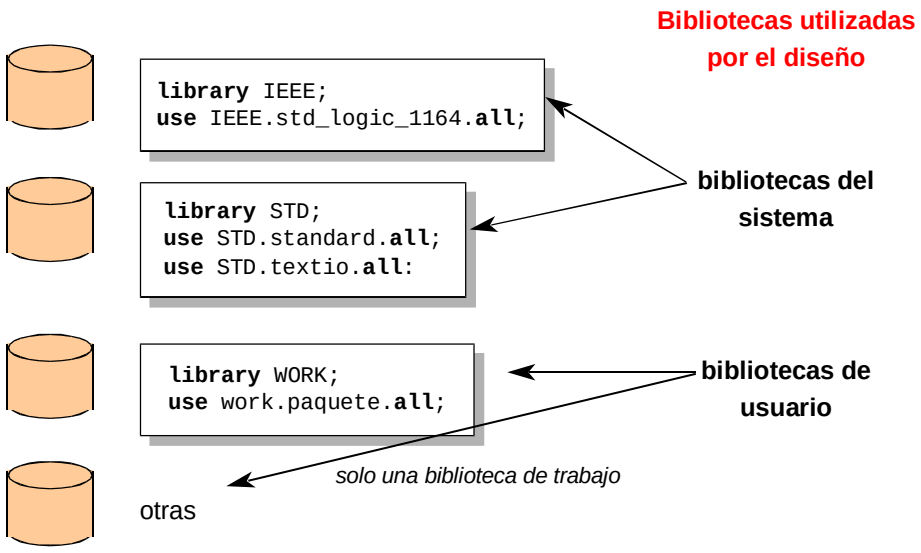
Bibliotecas



Bibliotecas



Bibliotecas



Bibliotecas del Sistema

- Biblioteca STD
  - Paquetes STANDARD y TEXTIO
  - No es necesario incluirla explícitamente
- Biblioteca IEEE
  - Paquete standard 1164 (std_logic_1164)
    - lógica multivaluada: (‘U’,’X’,‘0’,’1’,’Z’,’W’,’L’,’H’,’−’)
    - Subtipos: (‘X’,’0’,’1’), (‘X’,’0’,’1’,’Z’), ...
    - Operaciones lógicas, funciones de conversión de tipos
  - Paquete aritmético (std_logic_arith)
    - Funciones aritméticas sobre tipos std_logic
    - Funciones de conversión enteros <=> std_logic