LABORATORIO DE CIRCUITOS DIGITALES
(2005-II)
TERCERA CLASE DE VHDL

- ✓ *TIPO DE DATO 'integer'*
- ✓ *DESCRIPCIÓN ESTRUCTURAL*
  - Ejemplos: Comparador y Sumador
  - Agrupación de bloques mediante 'package'
  - Uso de la sentencia 'port map'.
- ✓ *MÓDULO DIGITAL PROGRAMABLE*
  - Introducción

Instructor: Daniel Llamocca

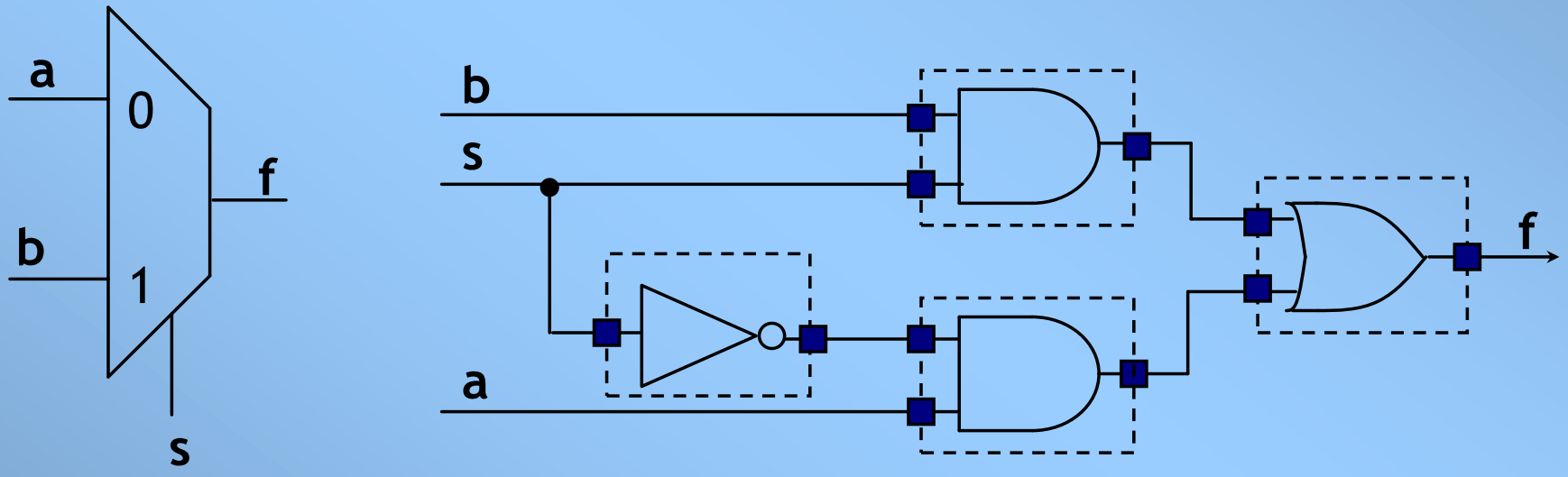
✓ Uso del tipo de dato ‘integer’

- Una señal del tipo INTEGER representa un número binario, con la particularidad de que no se indica la cantidad de bits de la señal, lo que puede ser muy cómodo.
- En el ejemplo se aprecia cómo se define una señal de tipo *integer* y cómo es que se trabaja con este tipo de datos. El ejemplo en mención es un decodificador BCD a 7 segmentos, con la entrada BCD de tipo INTEGER. El sintetizador asigna los bits necesarios a la entrada BCD, que en este caso serían ‘4’.
- El tipo ‘*integer*’ no permite el acceso a los bits por separado, como sí lo permite el ‘*std_logic_vector*’

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

entity dec_7_int is
 port(bcd: in integer range 0 to 9;
 -- bcd: 0000 a 1001 ==> ocupa 4 bits
       leds: out std_logic_vector(0 to 6));
end dec_7_int;

architecture bhv of dec_7_int is
begin
  with bcd select --abcdefg
      leds <=   "1111110" when 0,
                "0110000" when 1,
                "1101101" when 2,
                "1111001" when 3,
                "0110011" when 4,
                "1011011" when 5,
                "1011111" when 6,
                "1110000" when 7,
                "1111111" when 8,
                "1111011" when 9;
end bhv;
```

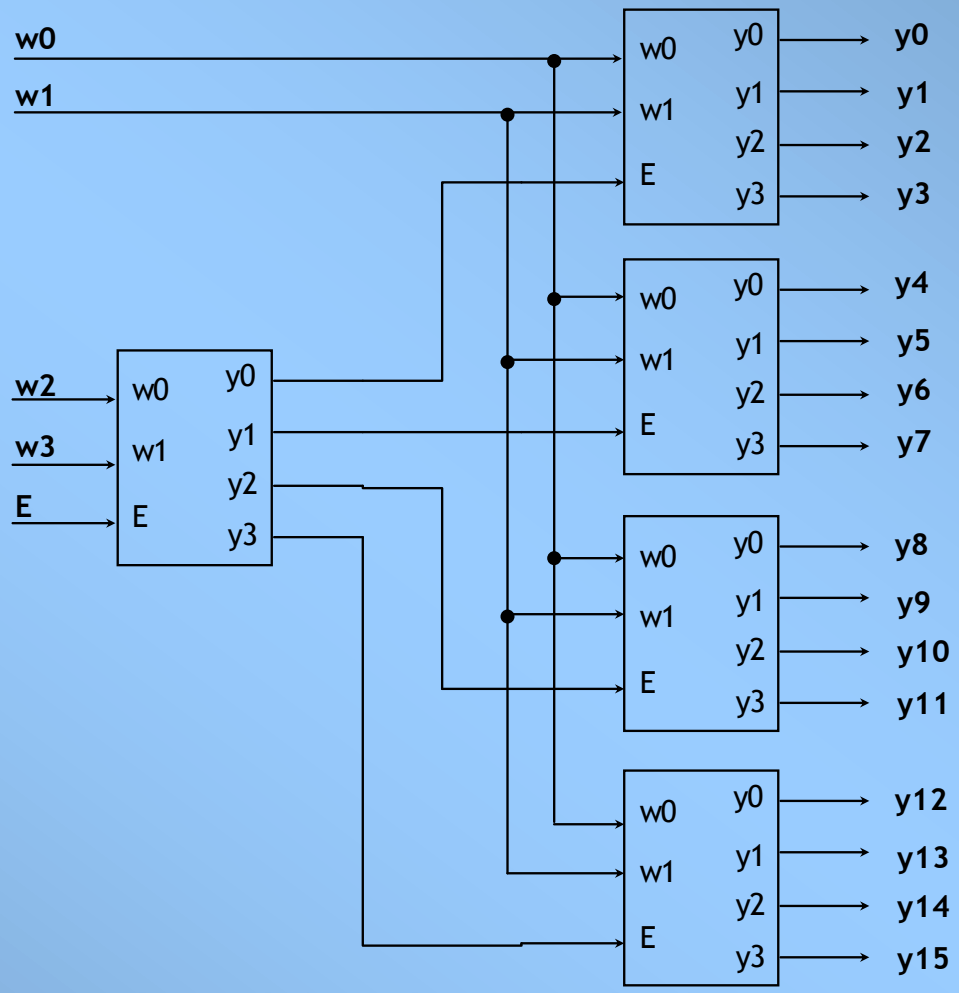
✓ *DESCRIPCIÓN ESTRUCTURAL*

- Es la generalización de la Descripción Concurrente. Los circuitos se describen por medio de la interconexión de sus sub-circuitos. Estos subcircuitos pueden estar descritos en Código Concurrente y/o Código Secuencial.
- **1er Ejemplo**: Multiplexor 2 a 1



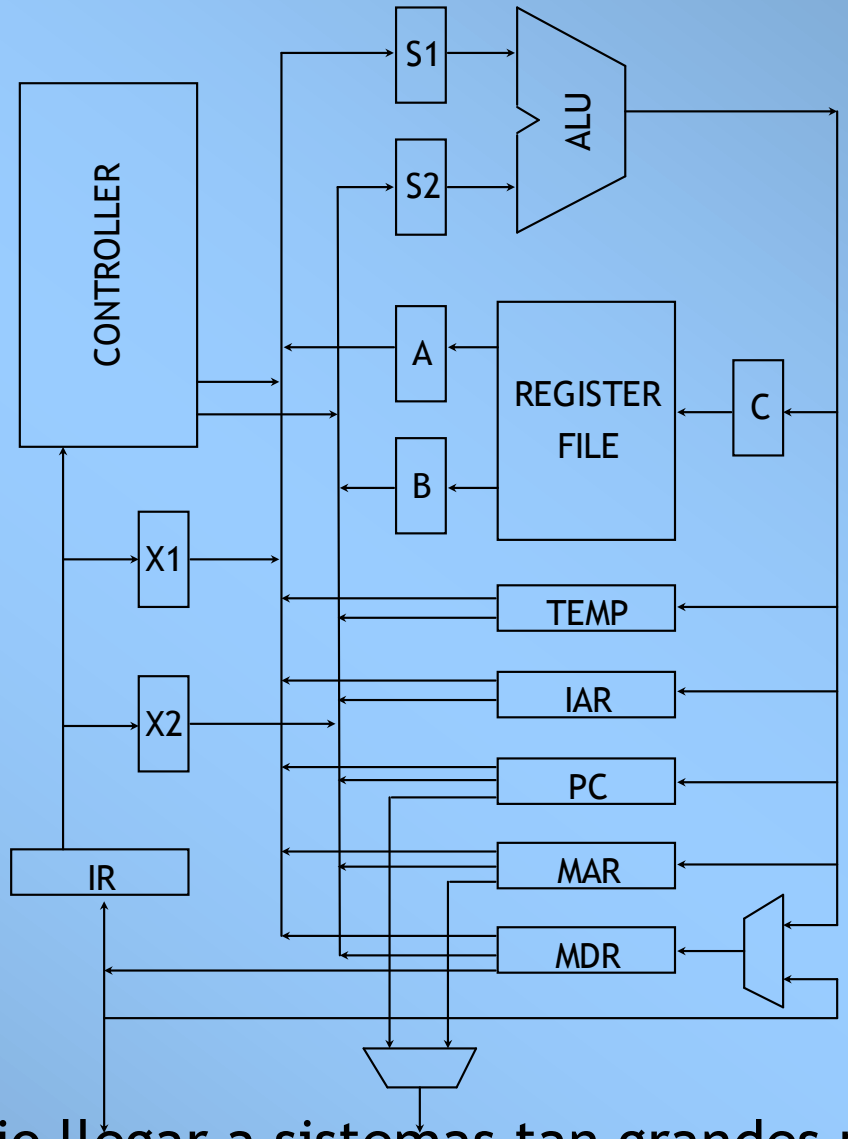
- Este caso es trivial, ya que la interconexión se realiza por medio de los operadores, pero es un ejemplo de Descripción Estructural.

- **2do ejemplo:** Decodificador 4 a 16



- Si bien este decodificador puede formarse en forma estructural en base a los decodificadores 2 a 4, también es cierto que todo el decodificador 4 a 16 puede ser descrito con una sola sentencia with – select.

- **3er ejemplo:** Procesador DLX

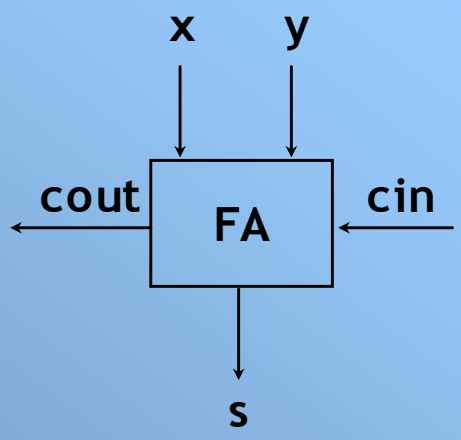


- No es necesario llegar a sistemas tan grandes para darse cuenta de la importancia de la Descripción Estructural.

- **Conclusiones:**
- Muchos sistemas pueden describirse totalmente en forma comportamental, y/o mediante las instrucciones concurrentes with – select ó when – else, todo en un solo bloque compacto. Sin embargo, el abuso de esta técnica tiene muchas desventajas: dificulta la lectura del código, dificulta el proceso de verificación del circuito, dificulta la posterior mejora del código: La optimización sólo podría hacerse a nivel global, no módulo por módulo, ni entre módulo y módulo que es en donde existen las mayores posibilidades de optimización.
- La Descripción estructural permite un diseño Jerárquico: Se puede ver todo el circuito con las partes que lo componen, identificar puntos críticos, proponer mejoras entre los bloques, mejorar cada bloque individual, etc.
- Es conveniente siempre tener bloques básicos a partir de los cuales se puedan formar circuitos más complejos. Esto permite que un bloque o subsistema pueda reusarse en otro circuito.

✓ *DESCRIPCIÓN ESTRUCTURAL:*

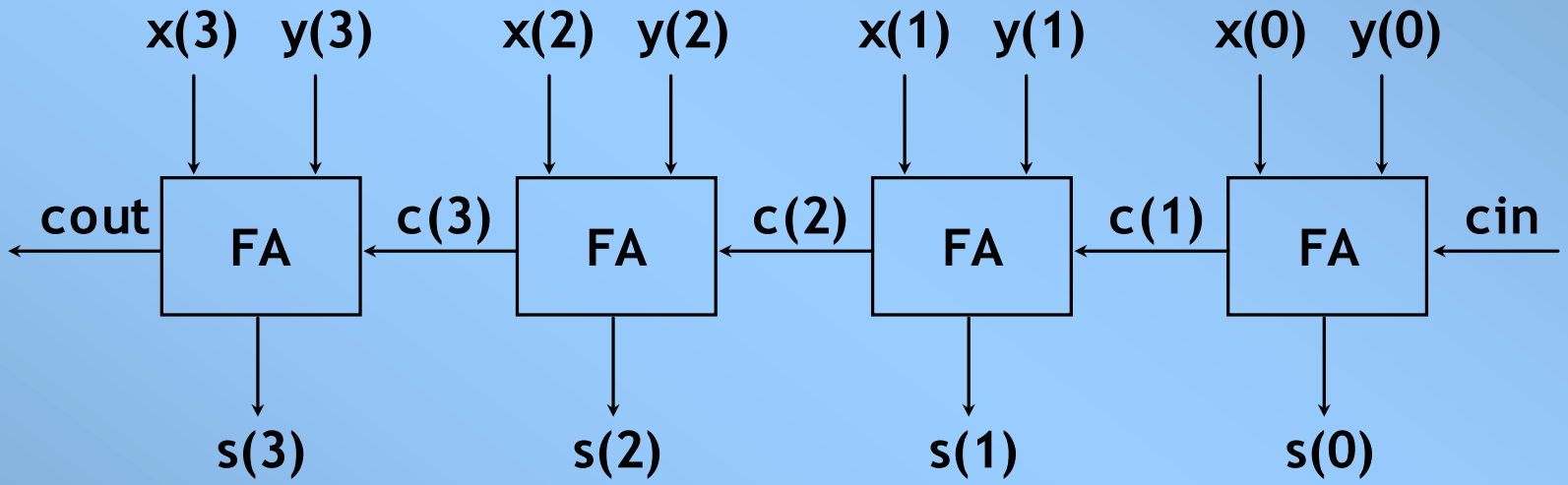
- **SUMADOR COMPLETO:**
- Se construirá un sumador de 4 bits en base a 4 bloques sumadores de 1 bit cada uno.
- Descripción VHDL para el sumador completo de 1 bit:




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

entity fulladd is
 port( cin, x, y : in std_logic;
       s, cout   : out std_logic);
end fulladd;

architecture LogicFunc of fulladd is
begin
 s <= x xor y xor cin;
 cout <= (x and y) or (x and cin) or (y and cin);
end LogicFunc;
```

✓ *DESCRIPCIÓN ESTRUCTURAL:*

- **SUMADOR COMPLETO DE 4 BITS: *adder.vhd***
- Se construye un sumador de 4 bits en base a 4 bloques sumadores de 1 bit cada uno:



✓ *DESCRIPCIÓN ESTRUCTURAL:*

- **Agrupación de bloques mediante 'package':**

- La sintaxis del VHDL estipula que en el código de más alta jerarquía se deben declarar todos los componentes a usarse, de la siguiente forma:

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

package digitales_package is
 component a
  port ( <entradas> : in std_logic;
         <salidas>  : out std_logic);
 end component;

 component b
  port ( <entradas> : in std_logic;
         <salidas>  : out std_logic);
 end component;

 component c
  port ( <entradas> : in std_logic;
         <salidas>  : out std_logic);
 end component;
...
end digitales_package;

library ieee;
use ieee.std_logic_1164.all;
use work.digitales_package.all;
```

OJO →

✓ *DESCRIPCIÓN ESTRUCTURAL:*

- **SUMADOR COMPLETO DE 4 BITS: *adder.vhd***
- En este caso, sólo debe declararse el bloque 'fulladd.vhd' descrito anteriormente, pues con 4 de estos se construirá un sumador de 4 bits. El código quedaría así:

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

package digitales_package is
 component fulladd
  port ( cin, x, y: in std_logic;
         s, cout: out std_logic);
 end component;
end digitales_package;

library ieee;
use ieee.std_logic_1164.all;
use work.digitales_package.all; -- Se indica que está haciendo
                                -- uso del package digitales

-- continúa en la página siguiente...
```

Se copia lo mismo de la entidad de fulladd.vhd

✓ *DESCRIPCIÓN ESTRUCTURAL:*

- **SUMADOR COMPLETO DE 4 BITS:**

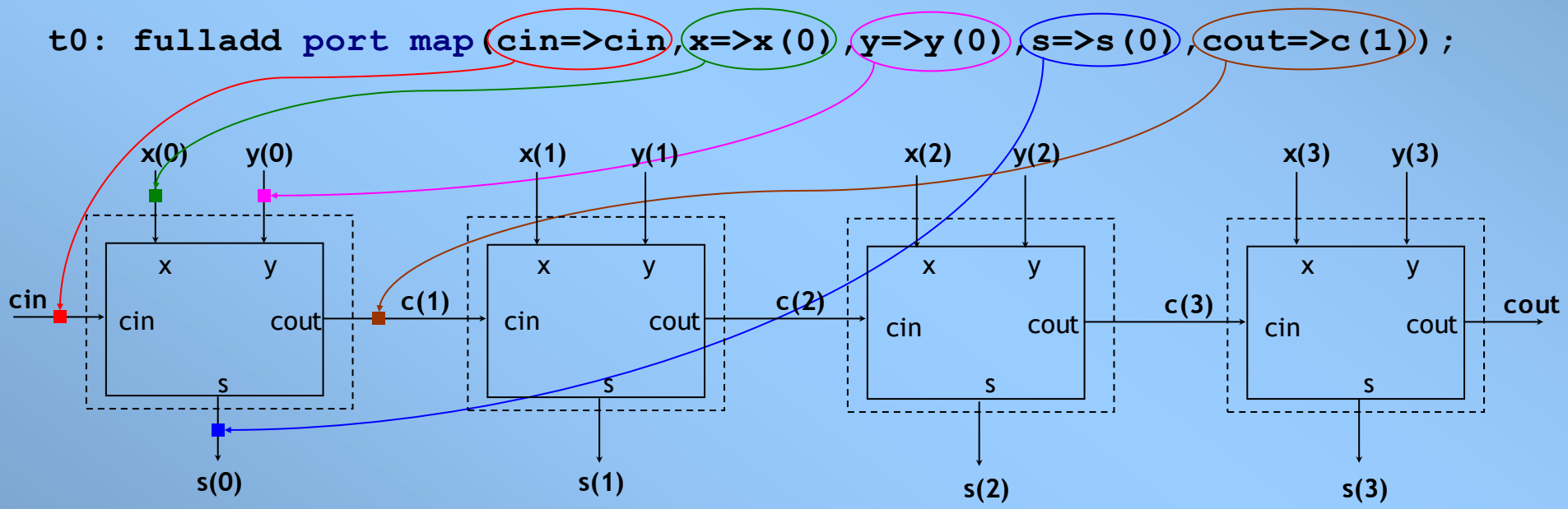
```
-- viene de la página anterior

entity adder is
 port( cin : in std_logic;
       x,y : in std_logic_vector (3 downto 0);
       s   : out std_logic_vector (3 downto 0);
       cout: out std_logic);
end adder;

architecture structure of adder is
 signal c: std_logic_vector(1 to 3);
begin
 t0: fulladd port map (cin=> cin,x=>x(0),y=>y(0),s=>s(0),cout=>c(1));
 t1: fulladd port map (cin=>c(1),x=>x(1),y=>y(1),s=>s(1),cout=>c(2));
 t2: fulladd port map (cin=>c(2),x=>x(2),y=>y(2),s=>s(2),cout=>c(3));
 t3: fulladd port map (cin=>c(3),x=>x(3),y=>y(3),s=>s(3),cout=>cout);
end structure;
```

✓ *DESCRIPCIÓN ESTRUCTURAL:*

- **Uso de la sentencia 'port map'**

  port map ([señal bloque < jerarquía] => [señal bloque de > jerarquía])

- **SUMADOR COMPLETO DE 4 BITS: *adder.vhd***



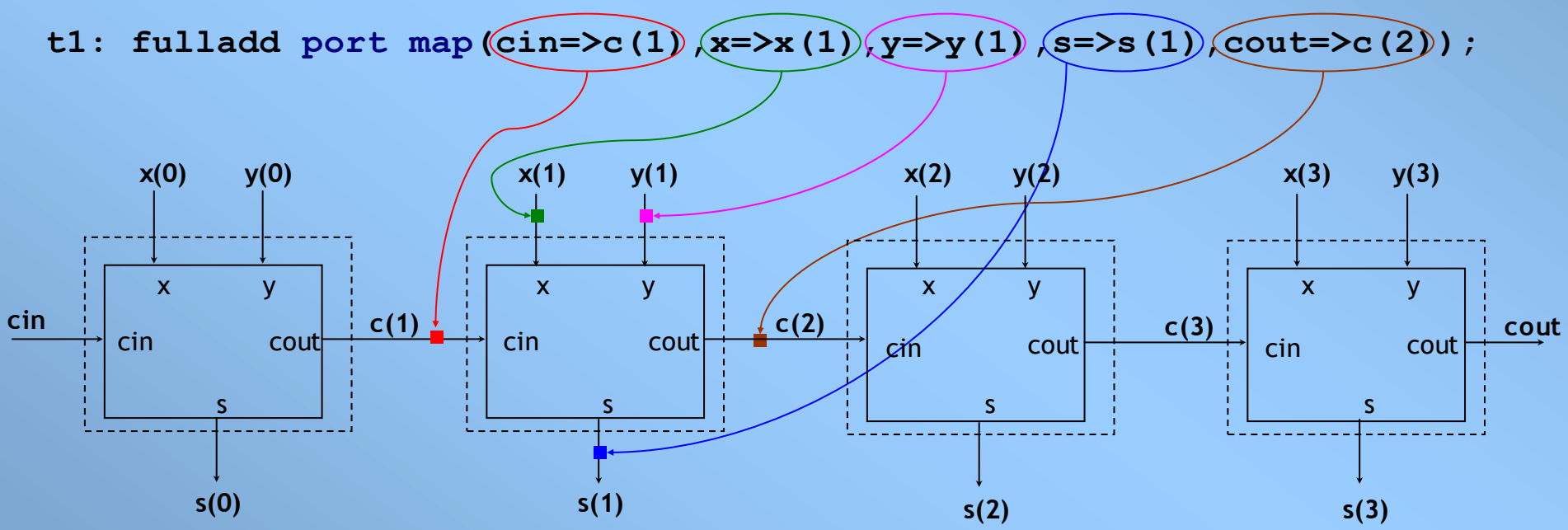
✓ *DESCRIPCIÓN ESTRUCTURAL:*

- **Uso de la sentencia 'port map'**

  port map ([señal bloque < jerarquía] => [señal bloque de > jerarquía])

- **SUMADOR COMPLETO DE 4 BITS: *adder.vhd***

```
t1: fulladd port map(cin=>c(1),x=>x(1),y=>y(1),s=>s(1),cout=>c(2));
```



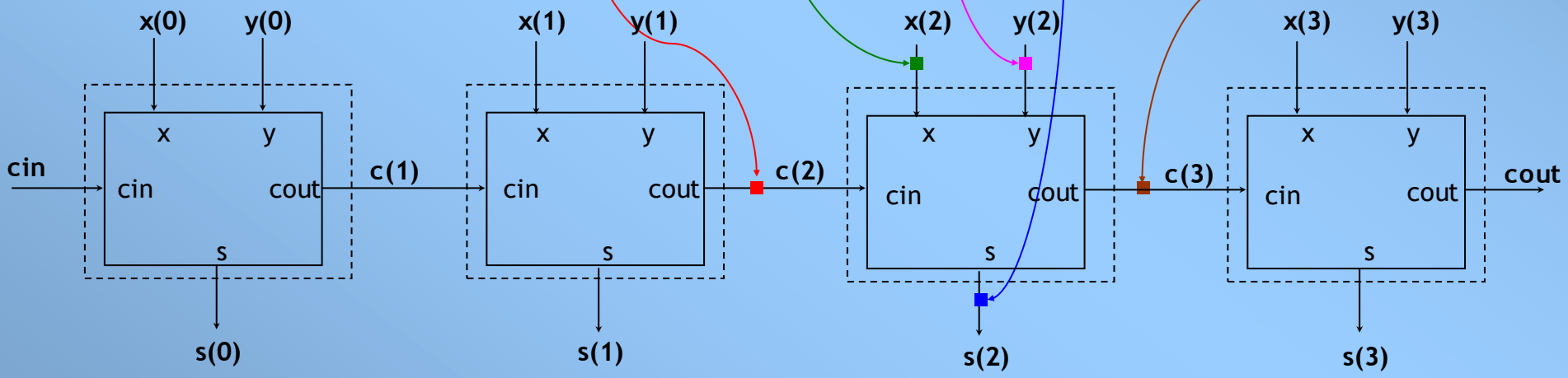
✓ *DESCRIPCIÓN ESTRUCTURAL:*

- **Uso de la sentencia 'port map'**

  port map ([señal bloque < jerarquía] => [señal bloque de > jerarquía])

- **SUMADOR COMPLETO DE 4 BITS: *adder.vhd***

```
t2: fulladd port map(cin=>c(2),x=>x(2),y=>y(2),s=>s(2),cout=>c(3));
```



✓ *DESCRIPCIÓN ESTRUCTURAL:*

- **Uso de la sentencia 'port map'**

  port map ([señal bloque < jerarquía] => [señal bloque de > jerarquía])

- **SUMADOR COMPLETO DE 4 BITS: *adder.vhd***

```
t3: fulladd port map(cin=>c(3),x=>x(3),y=>y(3),s=>s(3),cout=>cout);
```