Práctica 3

Sumadores

Objetivos

- Diseñar tres tipos sumadores de operandos sin signo, con acarreo de entrada y salida.
  - Sumador basado en el paquete *numeric_std*
  - Sumador con propagación de acarreos (*carry ripple*).
  - Sumador con anticipación de acarreos (*carry lookahead*).
- Comparar sus características de implementación.

Definición entidad

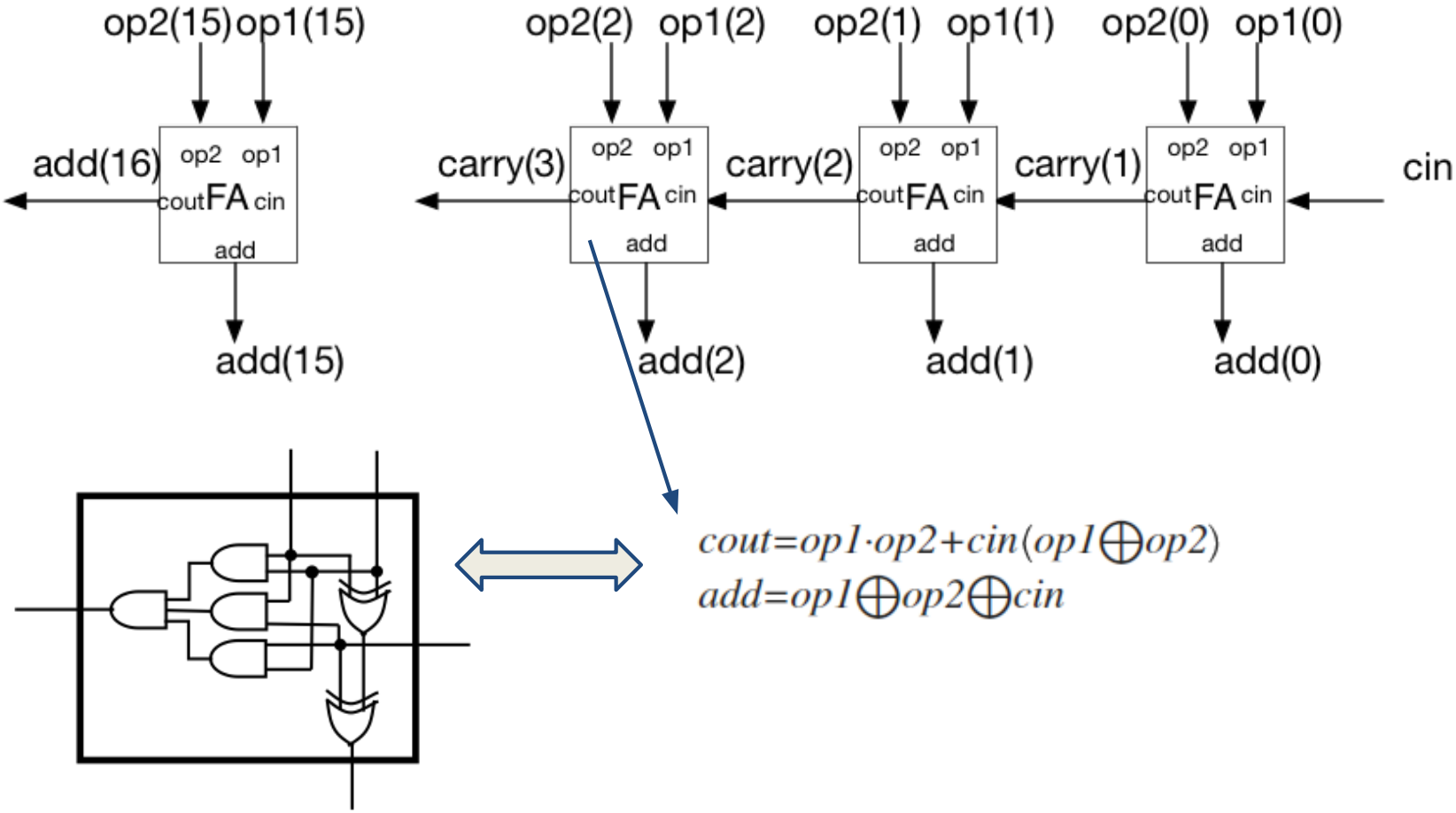
```
entity adder is
generic (
      g_width : natural := 32);
port (
 cin    : in  std_logic;
 op1  : in  std_logic_vector(g_width-1 downto 0);
 op2  : in  std_logic_vector(g_width-1 downto 0);
 add  : out std_logic_vector(g_width downto 0));
end adder;
```

Sumador *numeric_std*

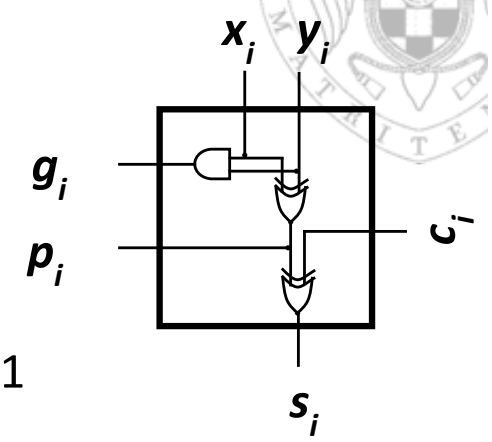
- Usad los operadores y tipos de datos definidos en el paquete *numeric_std*
  - Tipo: unsigned
  - Operador: suma ‘+’

Sumador con propagación de acarreos

- Instanciar 16 veces el componente FA (*full-adder*)



Sumador con anticipo de acarreos



- Se busca reducir el tiempo de cálculo:

$$c_{i+1} = x_i \cdot y_i + (x_i \oplus y_i) \cdot c_i = g_i + p_i c_i$$

  - gi es 1 si una celda genera un arrastre, es decir xi = yi = 1
  - pi es 1 si una celda propaga un arrastre, es decir xi $\oplus$ yi = 1

$$s_i = (x_i \oplus y_i) \oplus c_i = P_i \oplus c_i$$

$$c_{i+1} = x_i \cdot y_i + x_i \cdot c_i + c_i \cdot y_i = x_i \cdot y_i + (x_i \oplus y_i) \cdot c_i = G_i + P_i \cdot c_i$$

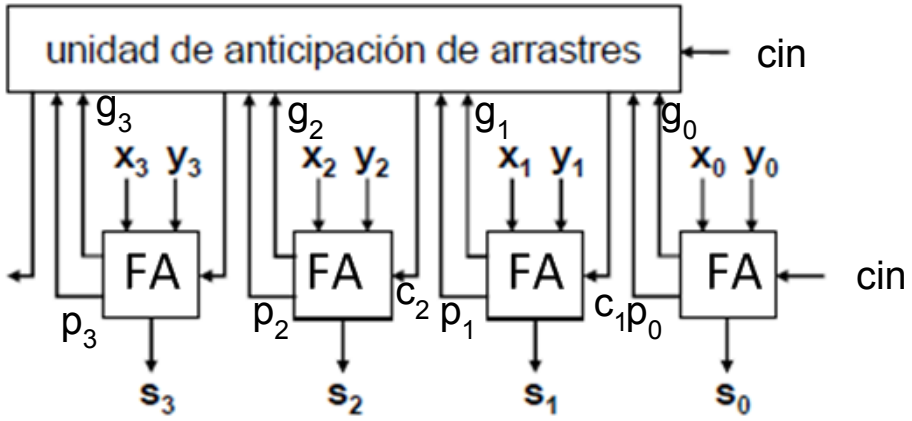
  - 4 niveles de puertas para cualquier bit de la suma
  - Conforme aumenta el número de bits el número de términos producto y el número de factores en ellos crece demasiado: para 32 bits el cálculo de c32 tiene 33 t.p y 33 factores

$$\begin{aligned}
c_1 &= G_0 + P_0 \cdot c_0 \\
c_2 &= G_1 + P_1 \cdot c_1 \\
&= G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot c_0 \\
c_3 &= G_2 + P_2 \cdot c_2 \\
&= G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot c_0 \\
c_4 &= G_3 + P_3 \cdot c_3 \\
&= G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot c_0
\end{aligned}$$

toc

Sumador con anticipo de acarreos

- Usa un circuito combinacional para anticipar el acarreo
  - No hay que esperar a que se propague a través de una red iterativa



  - Definir señales g_i y p_i
  - Usa [generación] señales para calcular [propagación] a partir de c_{i+1} c_i

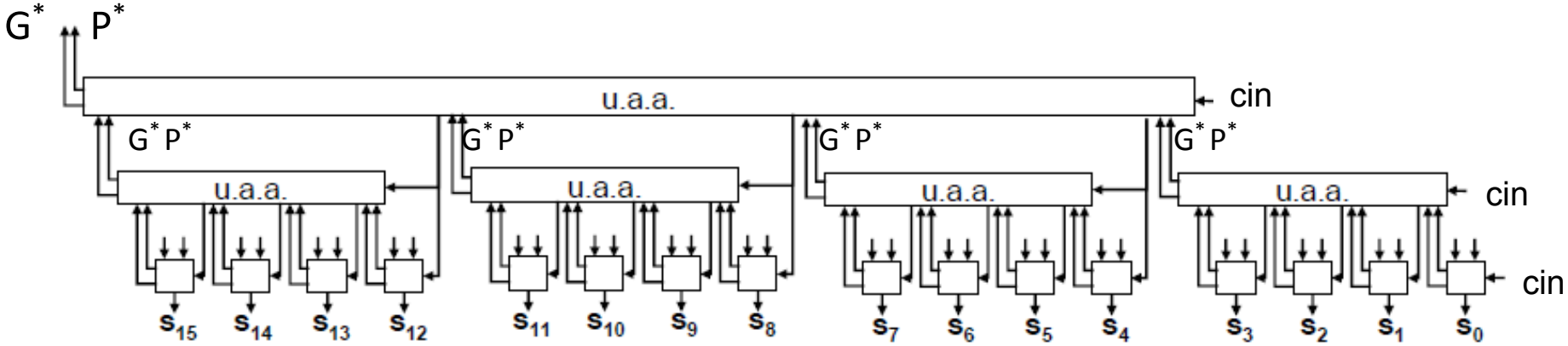
toc

Sumador con anticipo de acarreos

- Problema
  - Al aumentar el número de bits que se anticipan aumenta la complejidad de la lógica.
  - Para calcular c_{32} se requieren 33 términos producto cada uno de ellos con 33
- Solución
  - Anticipación de arrastre multinivel

Sumador con anticipo de acarreos

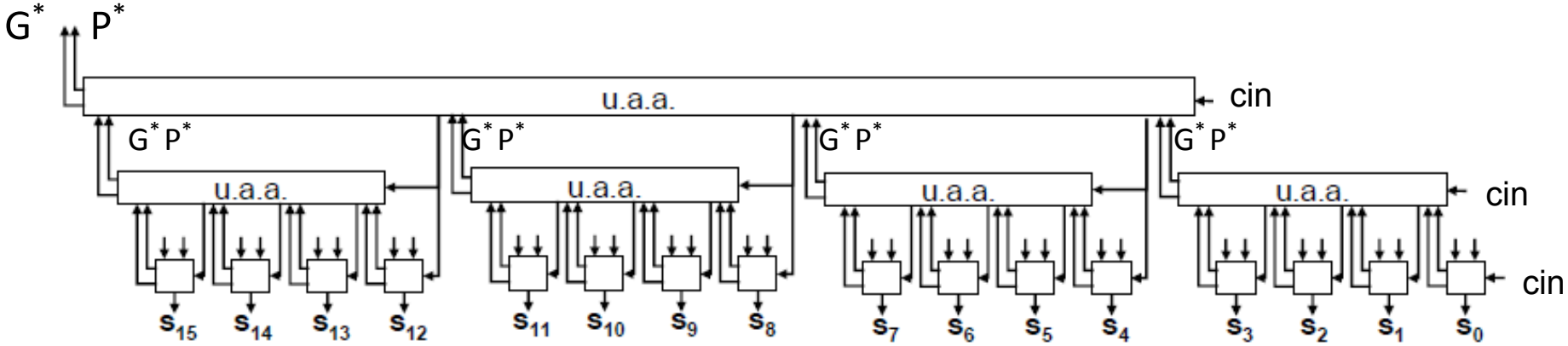
- Anticipación de arrastre multinivel



  - La anticipación se calcula sobre módulos con un mayor número de bits

Sumador con anticipo de acarreos

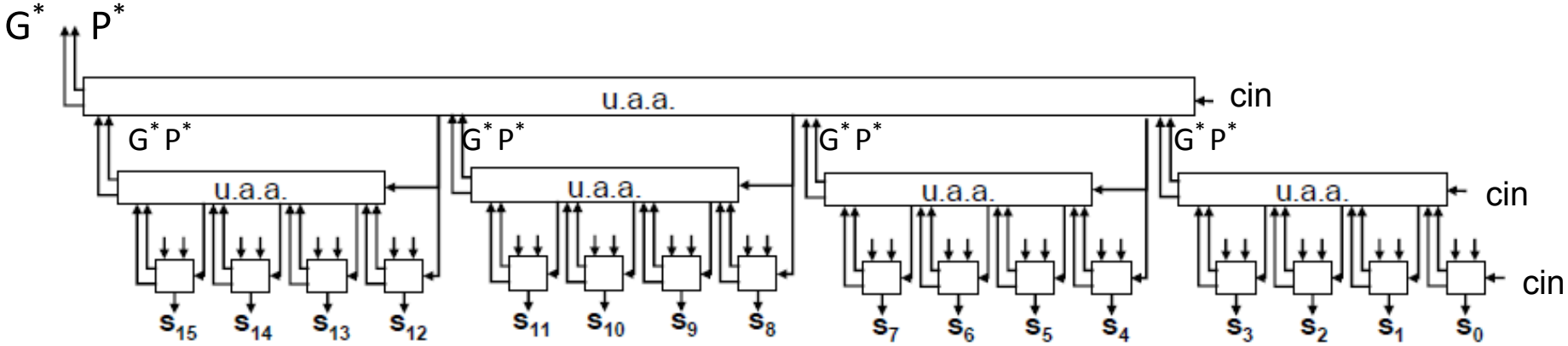
- Anticipación de arrastre multinivel



  - Un módulo genera un arrastre si se genera en alguna de sus celdas internas y se propaga hasta la salida
  - Un módulo propaga un arrastre si el arrastre de entrada es uno y todas las celdas intermedias lo propagan

Sumador con anticipo de acarreos

- Anticipación de arrastre multinivel



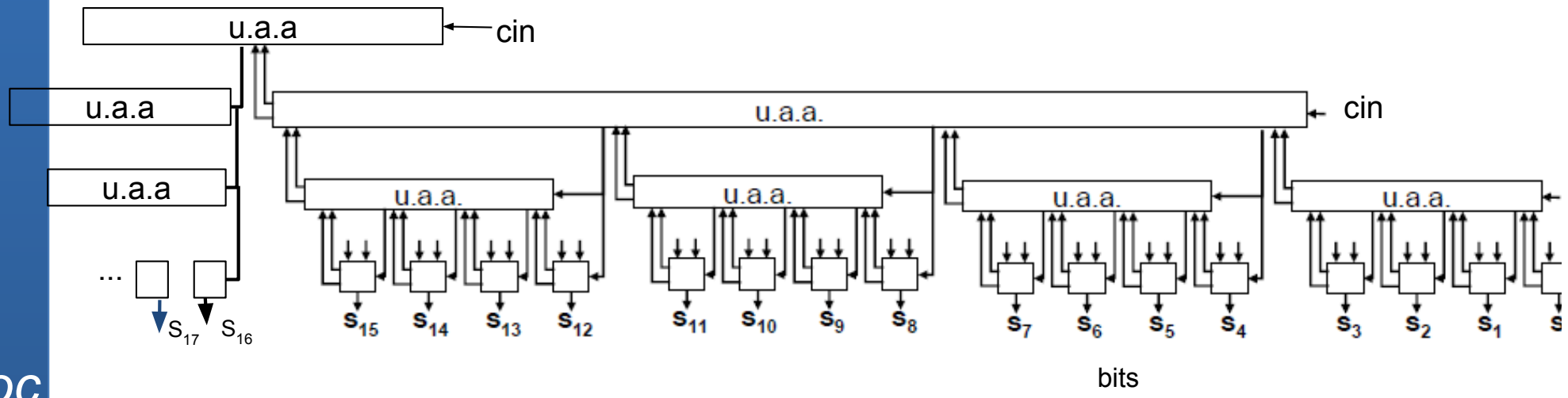
  - Para 4 bits

$$
\begin{aligned}
G^* &= G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 \\
P^* &= P_3 \cdot P_2 \cdot P_1 \cdot P_0 \\
c_{out} &= G^* + P^* \cdot c_{in}
\end{aligned}
$$

toc

Sumador con anticipo de acarreos

- Para 32 bits añadimos un nuevo nivel con u.a.a del que solo usaremos las dos primeras entradas.
- Para 64 bits se utilizarán todas las entradas del nuevo u.a.a.



Apartado básico

- Diseñar en VHDL los tres tipos de sumadores con anchos de 16 y 32 bits.
- Comparar su temporización y los recursos hardware. Presentar los resultados en dos gráficas.

slices y
camino crítico (ns)