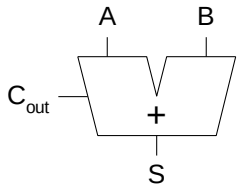
Lógica Combinacional en VHDL (II)

UNIVERSIDAD NEBRIJA

Sumadores de 1-Bit

Half Adder

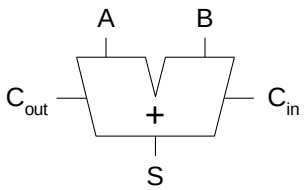


A	B	C_{out}	S
0	0		
0	1		
1	0		
1	1		

S =

C_{out} =

Full Adder



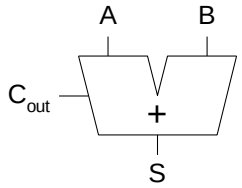
C_{in}	A	B	C_{out}	S
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

S =

C_{out} =

Sumadores de 1-Bit

Half Adder

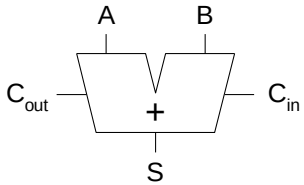


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

S =

C_{out} =

Full Adder



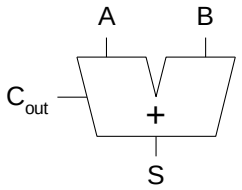
C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S =

C_{out} =

Sumadores de 1-Bit

Half Adder

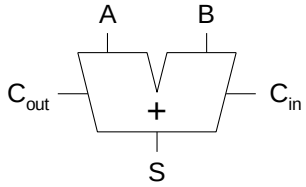


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A \oplus B$

$C_{out} = AB$

Full Adder



C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$S = A \oplus B \oplus C_{in}$

$C_{out} = AB + AC_{in} + BC_{in}$

Implementación en VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity fulladder is
  port(a, b, cin: in  STD_LOGIC;
       s, cout:   out STD_LOGIC);
end fulladder;

architecture synth of fulladder is
  signal p, g: STD_LOGIC;
begin
  p <= a xor b;
  g <= a and b;

  s <= p xor cin;
  cout <= g or (p and cin);
end synth;
```

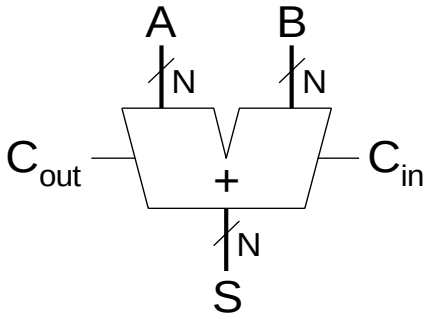
```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity adder is
  generic(N: integer := 8);
  port(a, b: in  STD_LOGIC_VECTOR(N-1 downto 0);
       cin:  in  STD_LOGIC;
       s:    out STD_LOGIC_VECTOR(N-1 downto 0);
       cout: out STD_LOGIC);
end adder;

architecture synth of adder is
  signal result: STD_LOGIC_VECTOR(N downto 0);
begin
  result <= ("0" & a) + ("0" & b) + cin;
  s      <= result(N-1 downto 0);
  cout   <= result(N);
end synth;
```

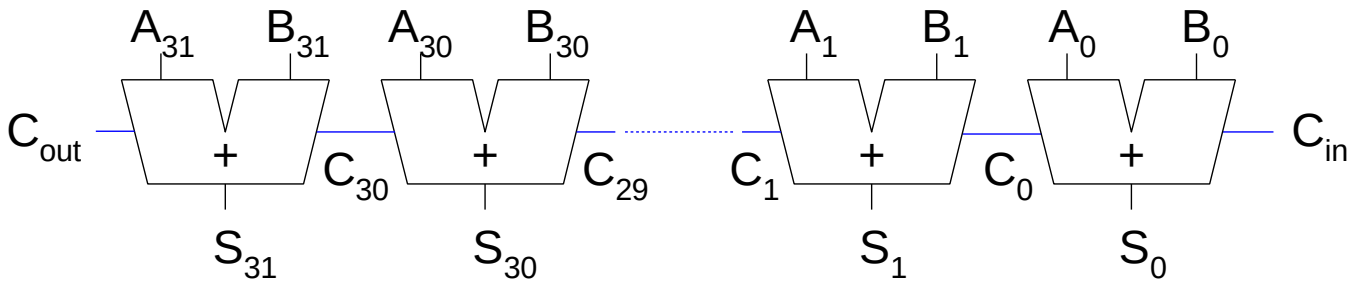
Sumadores multibit (CPAs)

- Tipos de carry propagate adders (CPAs):
  - Ripple-carry (slow)
  - Carry-lookahead (fast)
  - Prefix (faster)
- Carry-lookahead y prefix adders son más rápidos para sumadores más grandes, pero requieren más hardware.



Ripple-Carry Adder

- Cadena de sumadores de 1-bit
- El acarreo se propaga a través de toda la cadena
- Desventaja: lento



Retardo del Ripple-Carry Adder

$$t_{\mathbf{ripple}} = Nt_{FA}$$

donde t_{FA} es el retardo de un full adder

Carry-Lookahead Adder

- Calcula el acarreo (C_{out}) para bloques de *k*-bits usando señales *generar y propagar.*
- **Algunas definiciones:**
  - El elemento/columna *i* produce un acarreo o generando uno Nuevo o propaganda el que le llega a la entrada.
  - Se definen señales Generar (G_i) y Propagar (P_i) para cada columna:
    - La columna *i* generará un acarreo nuevo si A_i y B_i son ambos 1.

$$G_i = A_i B_i$$

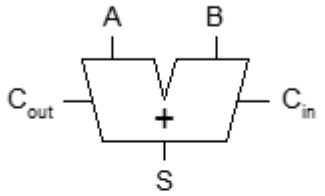
    - La columna *i* propagará un acarreo de la entrada a la salida si A_i o B_i son 1.

$$P_i = A_i + B_i$$

    - El acarreo de la columna *i* (C_i) es:

$$C_i = A_i B_i + (A_i + B_i) C_{i-1} = G_i + P_i C_{i-1}$$

Full Adder



C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Generar

Propagar

UNIVERSIDAD NEBRIJA

Suma Carry-Lookahead

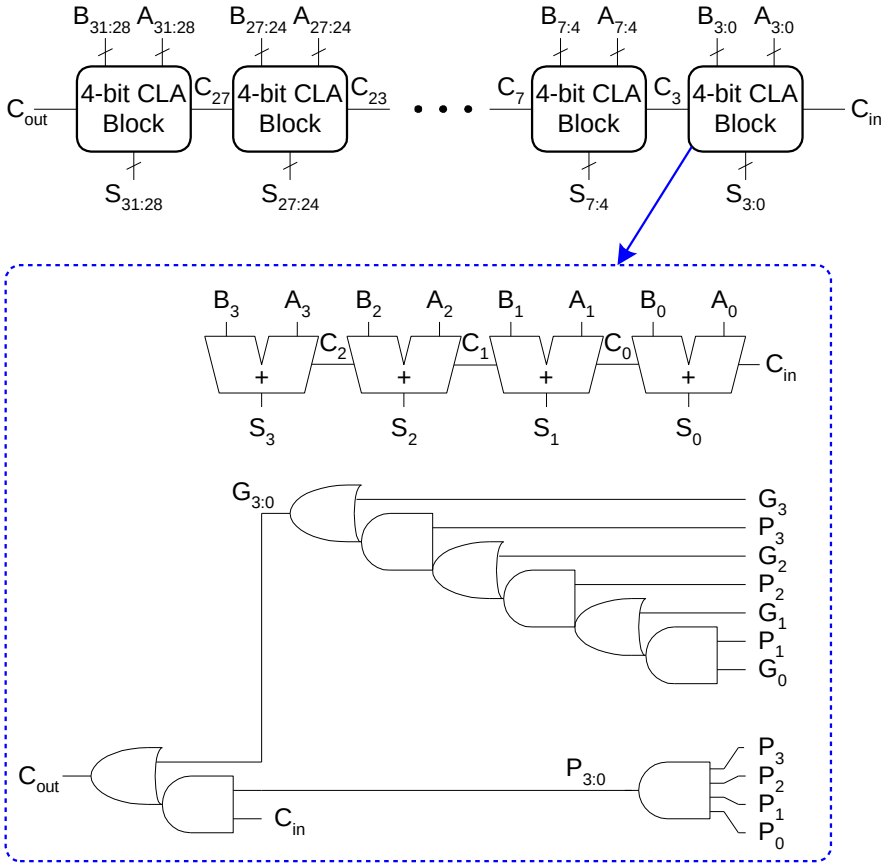
- **Step 1:** Calcula G_i y P_i para todas las columnas
- **Step 2:** Calcula G y P para los bloques de k-bits
- **Step 3:** C_{in} se propaga a través de cada bloque de generar/propagar para k-bits.

Carry-Lookahead Adder

- **Ejemplo:** bloques de 4-bits ($G_{3:0}$ and $P_{3:0}$) :
  - $G_{3:0} = G_3 + P_3 (G_2 + P_2 (G_1 + P_1 G_0)$
  - $P_{3:0} = P_3 P_2 P_1 P_0$

- **Generalizando,**
  - $G_{i:j} = G_i + P_i (G_{i-1} + P_{i-1} (G_{i-2} + P_{i-2} G_j)$
  - $P_{i:j} = P_i P_{i-1} P_{i-2} P_j$
  - $C_i = G_{i:j} + P_{i:j} C_{i-1}$

32-bit CLA con bloques de 4-bits



Retardo del Carry-Lookahead Adder

- Para un CLA de *N*-bit con bloques de *k*-bits:
- $$t_{CLA} = t_{pg} + t_{pg\_block} + (N/k - 1)t_{AND\_OR} + kt_{FA}$$
  - t_{pg} : retardo para generar todos los P_i, G_i (solo dependen de A_i y B_i)
  - t_{pg_block} : retardo para generar todos los $P_{i:j}$, $G_{i:j}$
  - t_{AND_OR} : retardo desde C_{in} a C_{out} de la puerta AND/OR final en el bloque CLA de *k*-bits
- Un sumador carry-lookahead de *N*-bits es, en general, mucho más rápido que un ripple-carry adder para $N > 16$.

Prefix Adder

- Calcula el acarreo de entrada para cada columna (C_{i-1}) y luego la suma:

$$\boldsymbol{S_i = (A_i \oplus B_i) \oplus C_i}$$

- Calcula G y P para bloques de 1-, 2-, 4-, 8-bits, etc. hasta que todos los G_i se conocen
- $\log_2 N$ etapas

Prefix Adder

- El acarreo de entrada o se genera en la columna anterior o se propaga desde las previas.
- Definimos una columna auxiliar -1 para introducir C_{in}:

  $$\boldsymbol{G_{-1} = C_{in}, P_{-1} = 0}$$

- El acarreo de entrada de la columna i = acarreo de salida de la columna:

  $$\boldsymbol{C_{i-1} = G_{i-1:-1}}$$

- $G_{i-1:-1}$: señal para las columnas i-1 a -1
- Ecuación de la suma:
  - $\boldsymbol{S_i = (A_i \oplus B_i) \oplus G_{i-1:-1}}$
- **Objetivo:** Calcular rápidamente $G_{0:-1}$, $G_{1:-1}$, $G_{2:-1}$, $G_{3:-1}$, $G_{4:-1}$, $G_{5:-1}$, … (llamados ***prefijos***)

Prefix Adder

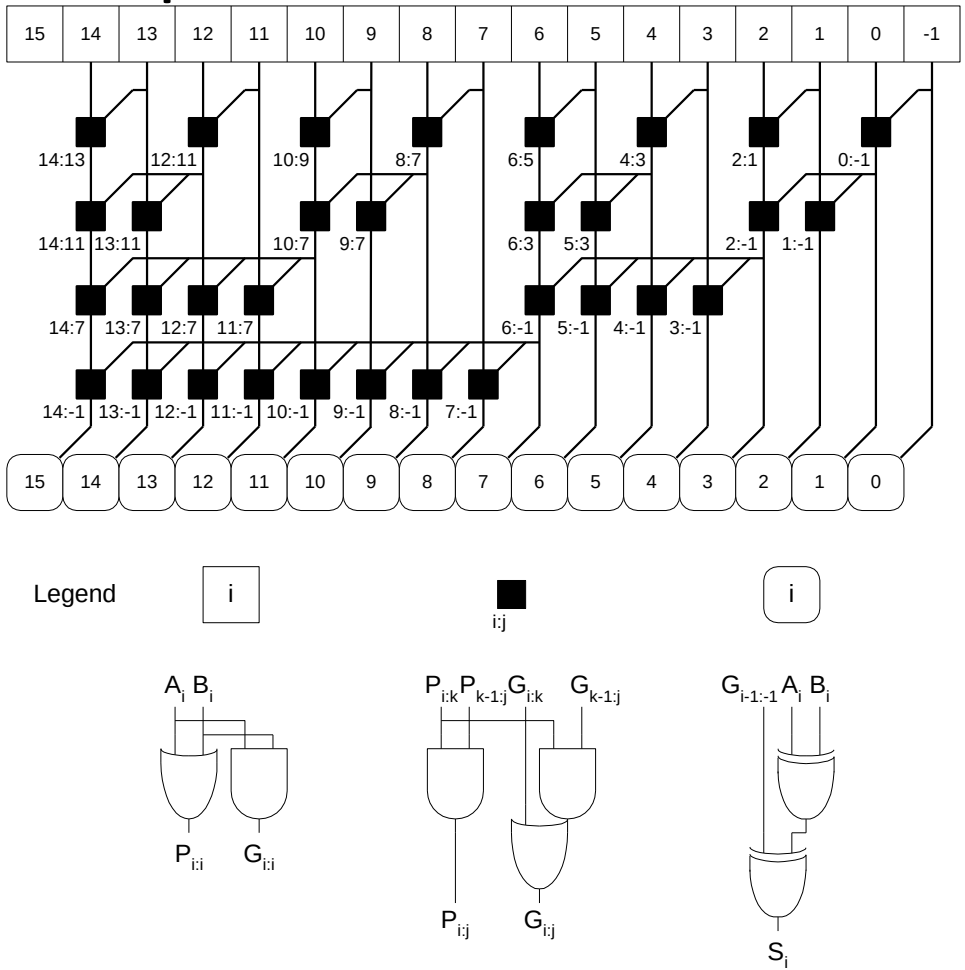
- Generar and propagar señales para un bloque que cubre los bits *i*:*j*:

$$G_{i:j} = G_{i:k} + P_{i:k}\, G_{k-1:j}$$

$$P_{i:j} = P_{i:k} P_{k-1:j}$$

- En palabras:
  - **Generar:** el bloque *i*:*j* generará acarreo si:
    - La parte superior (*i*:*k*) genera un acarreo o
    - La parte superior propaga un acarreo generado por la parte inferior (*k*-1:*j*)
  - **Propagar:** el bloque *i*:*j* propagará un acarreo si tanto la parte superior como la inferior lo propagan.

Esquema del Prefix Adder



Retardo del Prefix Adder

$$t_{PA} = t_{pg} + \log_2 N(t_{pg\_\text{prefix}}) + t_{\text{XOR}}$$

- t_{pg}: retardo en producir P_i G_i (puerta AND o OR)
- t_{pg_prefix}: retardo de las celdas de prefijo negras (AND + OR)

Comparación de retardo de sumadores

- Compara el retardo de: sumadores de 32-bit ripple-carry, carry-lookahead, y prefix adder
- CLA tiene bloques de 4-bits
- Retado de puerta de 2 entradas = 100 ps;
- Retardo de full adder = 300 ps

Comparación de retardo de sumadores

- Compara retardo de: 32-bit ripple-carry, carry-lookahead, y prefix adder
- CLA tiene bloques de 4-bits
- Retado de puerta de 2 entradas = 100 ps;
- Retardo de full adder = 300 ps

$$t_{\text{ripple}} = Nt_{FA} = 32(300 \text{ ps})$$
$$= \mathbf{9.6\ ns}$$

$$t_{CLA} = t_{pg} + t_{pg\_\text{block}} + (N/k - 1)t_{\text{AND\_OR}} + kt_{FA}$$
$$= [100 + 600 + (7)200 + 4(300)] \text{ ps}$$
$$= \mathbf{3.3\ ns}$$

$$t_{PA} = t_{pg} + \log_2 N(t_{pg\_\text{prefix}}) + t_{\text{XOR}}$$
$$= [100 + \log_2 32(200) + 100] \text{ ps}$$
$$= \mathbf{1.2\ ns}$$