Práctica 3

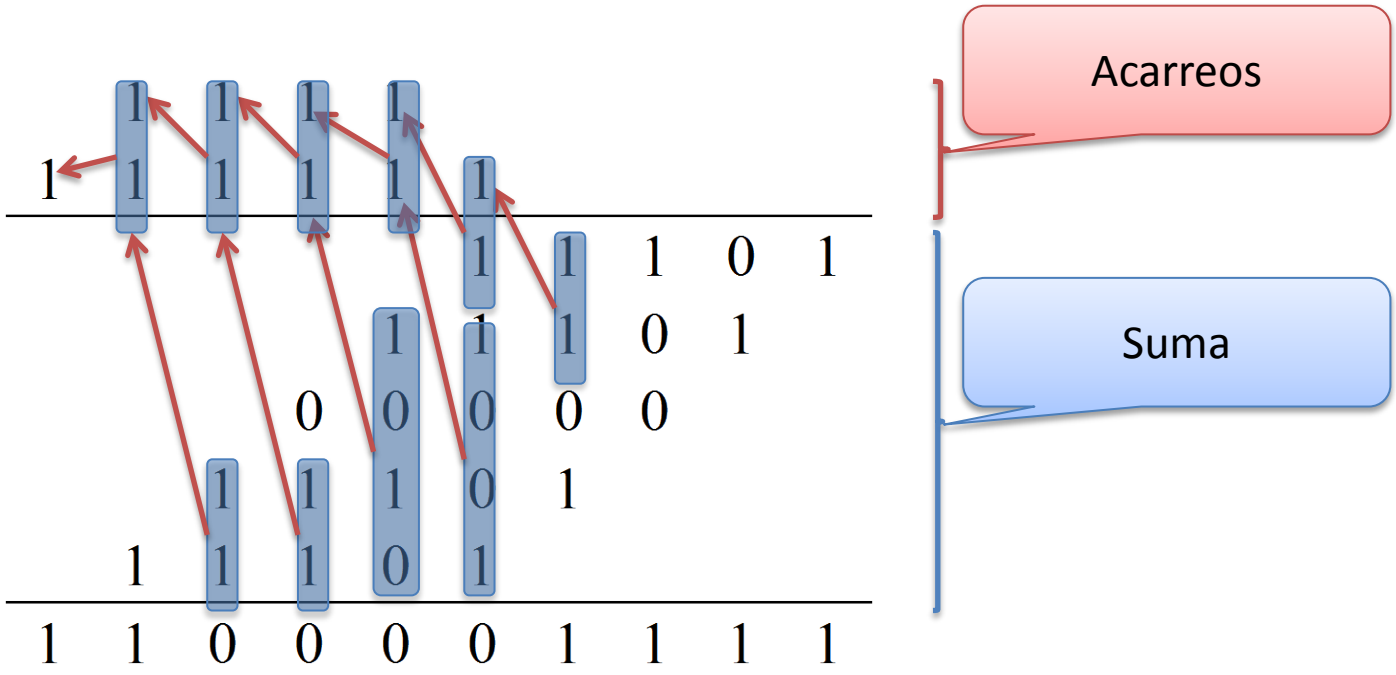
- Multiplicador en array

Multiplicación

					a_4	a_3	a_2	a_1	a_0
					x_4	x_3	x_2	x_1	x_0
					a_4x_0	a_3x_0	a_2x_0	a_1x_0	a_0x_0
				a_4x_1	a_3x_1	a_2x_1	a_1x_1	a_0x_1	
			a_4x_2	a_3x_2	a_2x_2	a_1x_2	a_0x_2		
		a_4x_3	a_3x_3	a_2x_3	a_1x_3	a_0x_3			
	a_4x_4	a_3x_4	a_2x_4	a_1x_4	a_0x_4				
P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

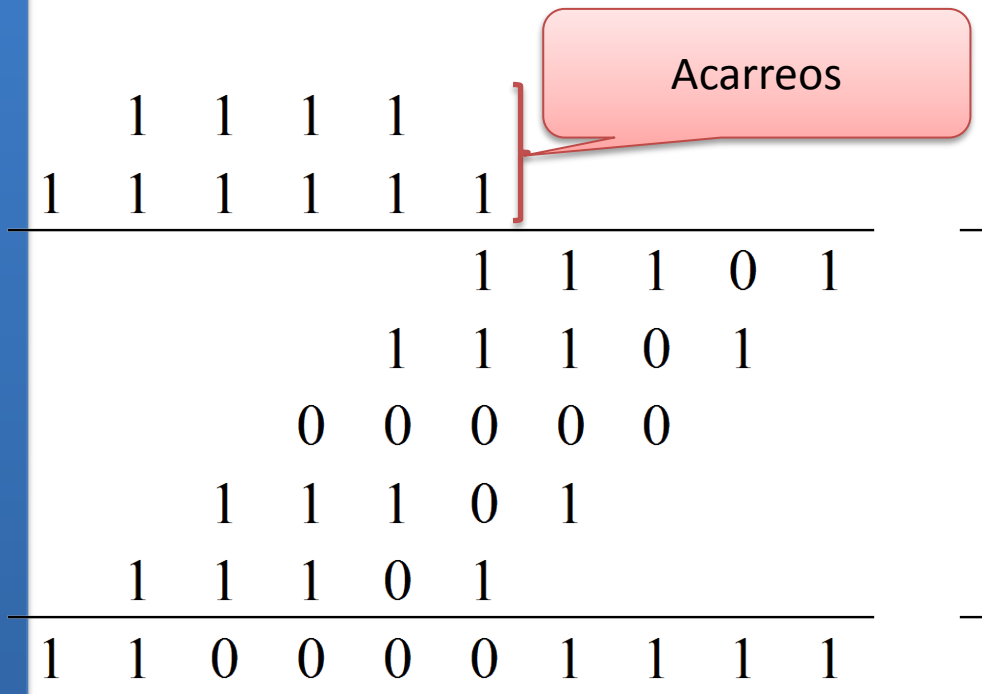
					1	1	1	0	1
					1	1	0	1	1
					1	1	1	0	1
				1	1	1	0	1	
			0	0	0	0	0		
		1	1	1	0	1			
	1	1	1	0	1				
1	1	0	0	0	0	1	1	1	1

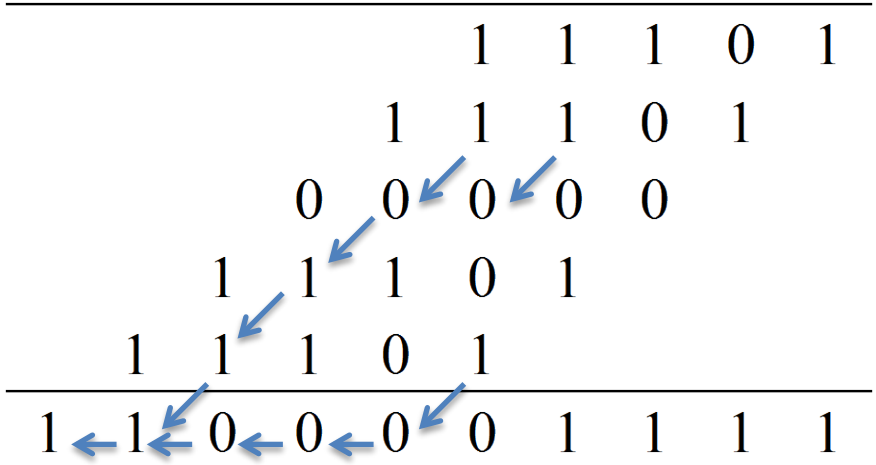
Multiplicación



Puede haber bastante acarreos ¿deberían sumarse a la vez?

Multiplicación





Cada flecha representa un acarreo

¿El resultado sería el mismo?

Multiplicación

a)

				1	1	1	0	1
			1	1	1	0	1	
		0	0	0	0	0		
	1	1	1	0	1			
1	1	1	0	1				
					1	1	1	1

b)

				1	1	1	0	1
			1	1	1	0	1	
		0	0	0	0	0		
	1	1	1	0	1			
1	1	1	0	1				
				0	1	1	1	1

c)

				1	1	1	0	1
			1	1	1	0	1	
		0	0	0	0	0		
	1	1	1	0	1			
1	1	1	0	1				
		0	0	1	1	1	1	

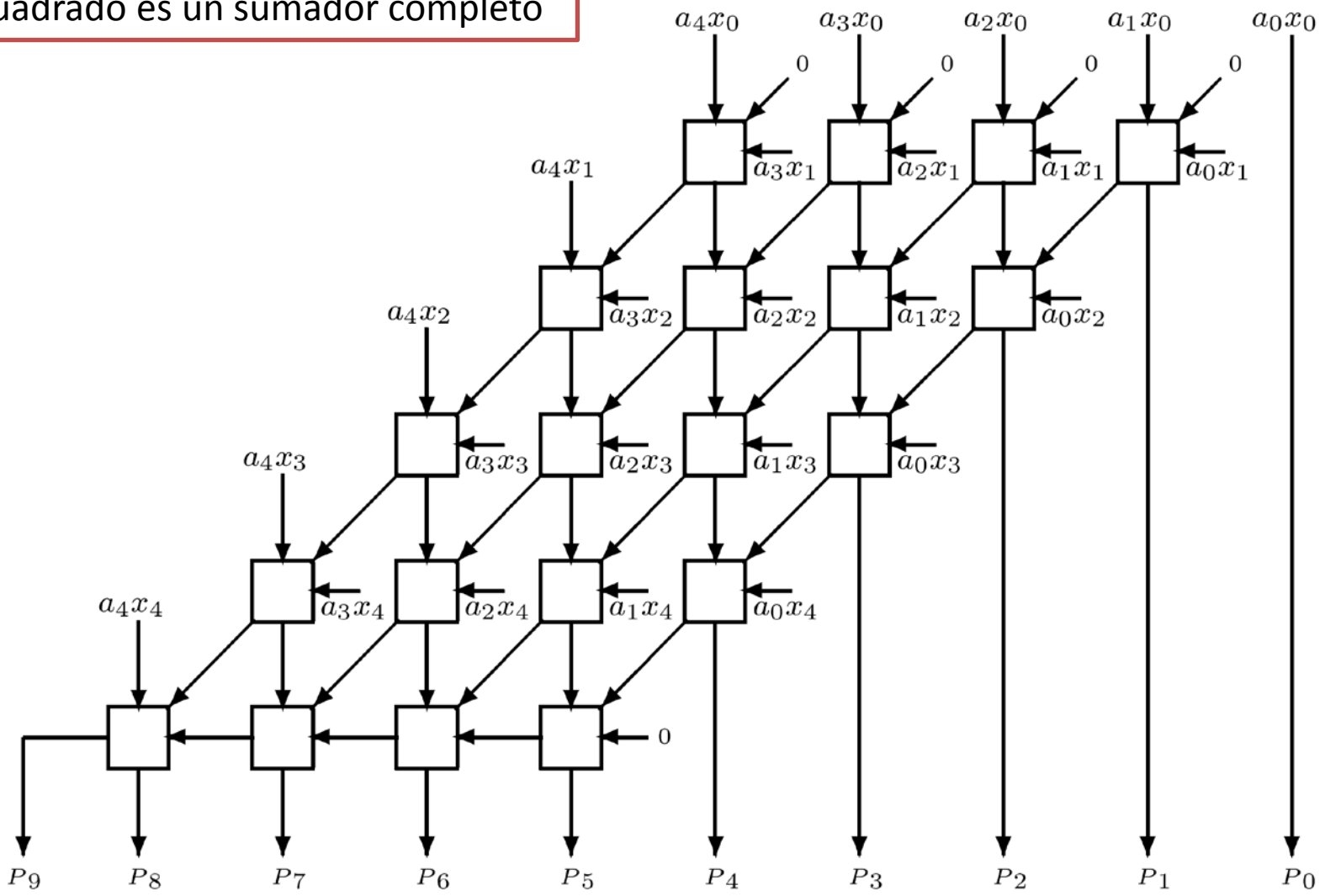
toc

Multiplicación

- ¿Cómo se traduce el esquema a Hardware?
  - Hay que hacer la and de todos los elementos del multiplicando por todos los elementos del multiplicador
  - Se suman todos los elementos de la misma columna mas el acarreo
  - El acarreo se pasa a la siguiente columna una fila más abajo
  - Los acarreos que se pasan al resultado se trasladan a la siguiente columna pero siguen en la fila resultado

Multiplicación

Cada cuadrado es un sumador completo



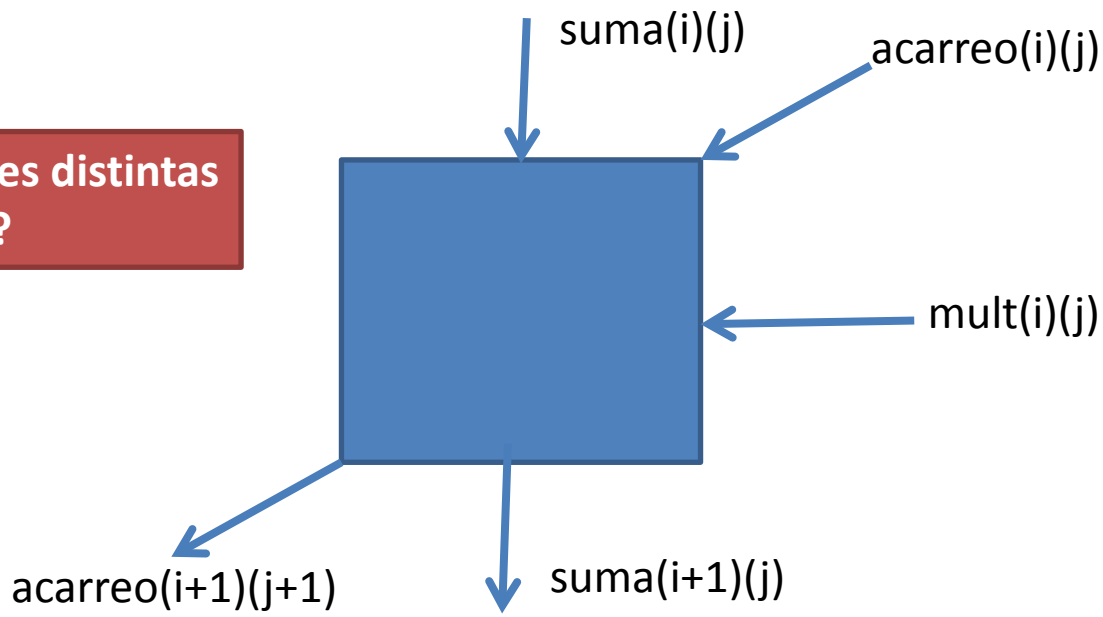
toc

Multiplicación

- Asignar nombres a los cables
  - Como se tiene una matriz (6x10) de bits todas las señales serán del siguiente tipo:

```
type matriz is array (0 to 5) of std_logic_vector(10 downto 0);
```

Sólo hay tres señales distintas ¿seguro?

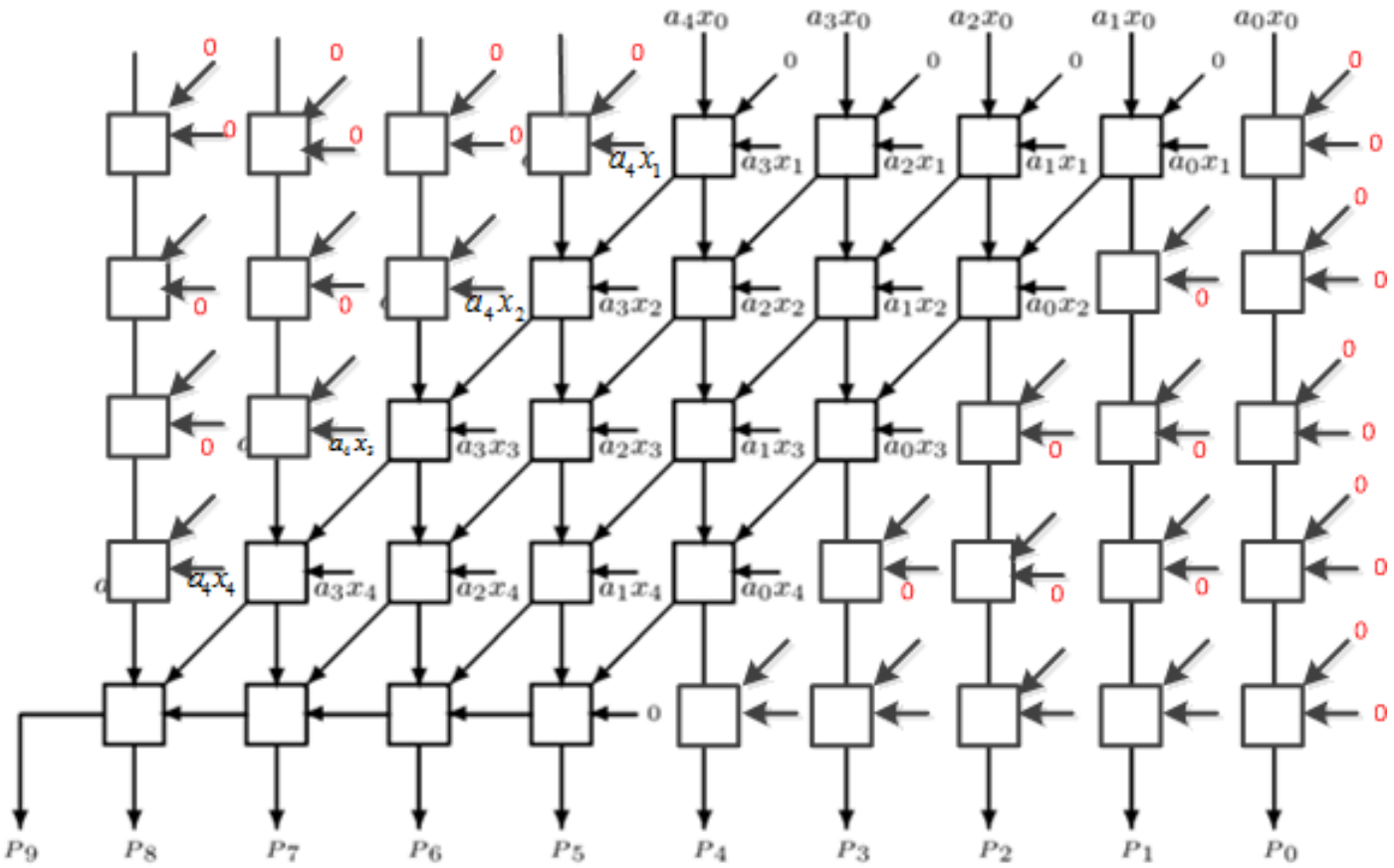


Multiplicación

```
entity celda is
port (suma_i, acarreo_i, mult: in std_logic;
            acarreo_o, suma_o: out std_logic);
end celda;
architecture circuito of celda is
begin
suma_o <= (suma_i xor acarreo_i) xor mult;
acarreo_o <= (suma_i and mult)or(acarreo_i and suma_i)or(acarreo_i and mult);
end circuito;
```

Multiplicación nxn

- Transformamos un paralelepípedo en un cuadrado



Práctica 3a

- Reconstruir en VHDL el cuadrado anterior
  - No hace falta generalizarlo, simplemente traducir el dibujo a código VHDL
  - Práctica 2a funcionando sobre FPGA: 2 puntos
    - Hay 12 switches en la FPGA (placa inferior y superior)
    - Pero sólo hay 8 LEDS, utilizar dos segmentos del display 7 segmentos para representar los bits más significativos de la multiplicación
      - Ej:

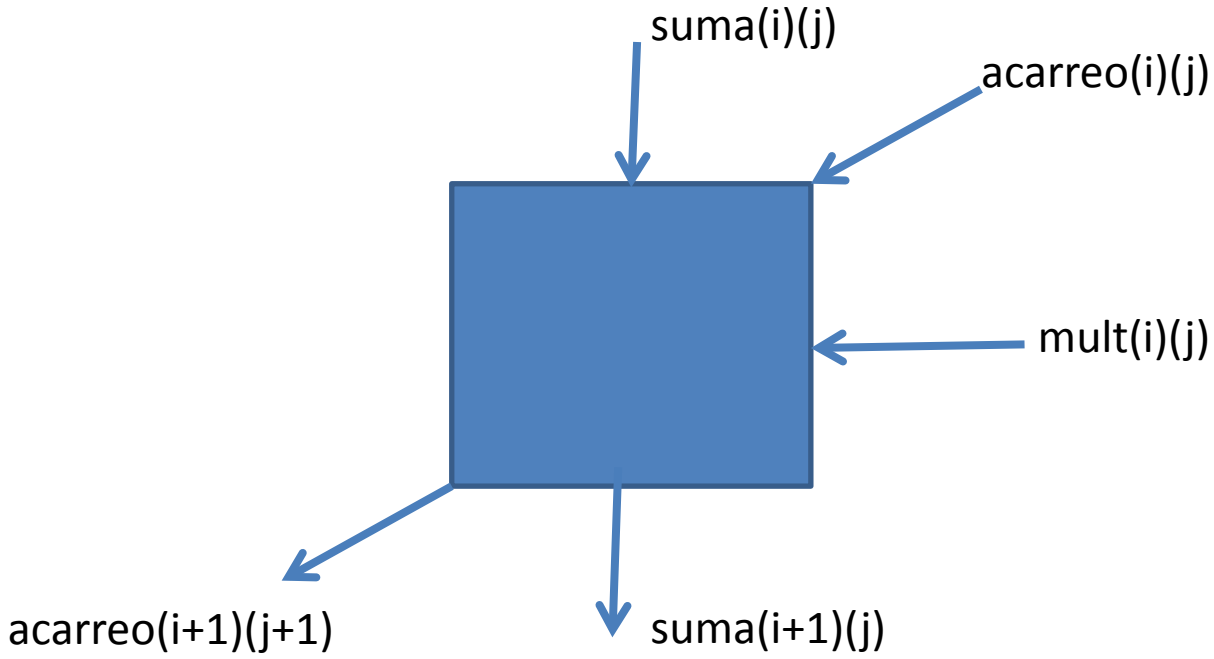
```
NET        P<8> LOC=H14;
NET        P<9> LOC=M4;
```

Cómo se generaliza

```
Filas: for i in 0 to N-1 generate
  Columnas: for j in 0 to 2xN-2 generate
```

- ¿Todas las columnas son iguales?
- ¿Todas las filas son iguales?
  - La interconexión de la última fila es diferente => La última fila no se hace en los dos for encadenados
    - for i tiene que ir hasta N - 2
    - La última fila se hará en un for generate aparte

Cómo se generaliza



Cómo se generaliza

- Condiciones de contorno
  - ¿Cuánto vale la suma (como entrada) para la fila 0?
  - ¿Cuánto vale acarreo (como entrada) para la fila 0 y para la columna 0?

Cómo se generaliza

0	0	0	0	a_4x_0	a_3x_0	a_2x_0	a_1x_0	a_0x_0
X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X

suma

acarreo

0	0	0	0	0	0	0	0	0
X	X	X	X	X	X	X	X	0
X	X	X	X	X	X	X	X	0
X	X	X	X	X	X	X	X	0
X	X	X	X	X	X	X	X	0

X = indeterminado, da igual su valor
ya lo calculará el circuito

Cómo se generaliza

- Ejemplo para condiciones de contorno de la señal suma:

```
process(a,x)
begin
        for j in 0 to 2*N-1 loop
                suma(0)(j)<='0';
                if j<(N-1) then
                        suma(0)(j)<=a(j) and x(0);
                end if;
        end loop;
end process;
```

Cómo se generaliza

- Valores iniciales

mult →

0	0	0	a_4x_1	a_3x_1	a_2x_1	a_1x_1	a_0x_1	0
0	0	a_4x_2	a_3x_2	a_2x_2	a_1x_2	a_0x_2	0	0
0	a_4x_3	a_3x_3	a_2x_3	a_1x_3	a_0x_3	0	0	0
a_4x_4	a_3x_4	a_2x_4	a_1x_4	a_0x_4	0	0	0	0

¿Dónde está el producto (AND) de x_0 con todas las a?

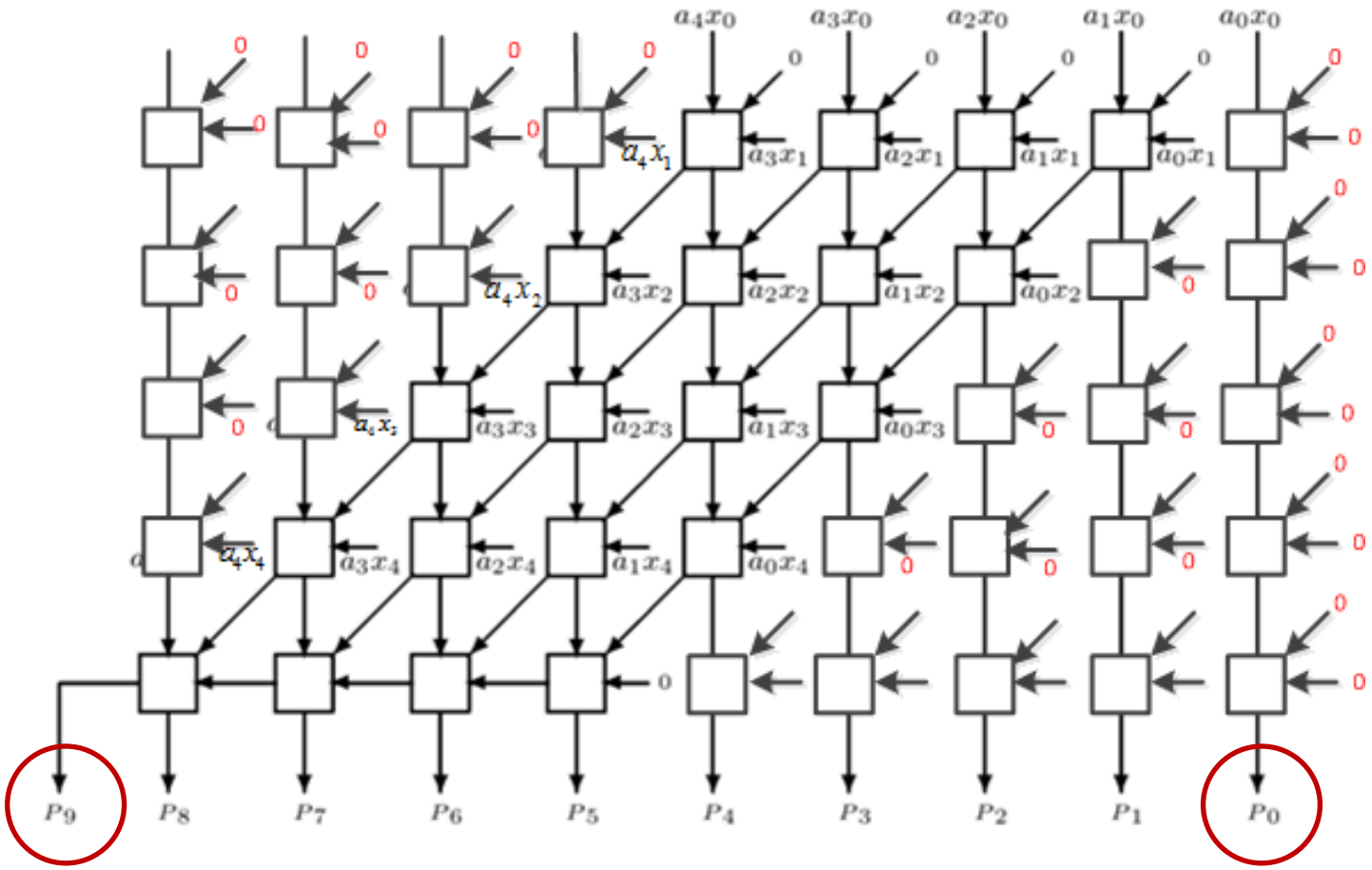
Como se generaliza

- En la fila 0 que columnas están ocupadas
  - De la 1 a la 5
- En la fila 1 que columnas están ocupadas
  - De la 2 a la 6
- En la fila 4 que columnas están ocupadas
  - De la 5 a la 9

Siempre están ocupadas sólo N columnas

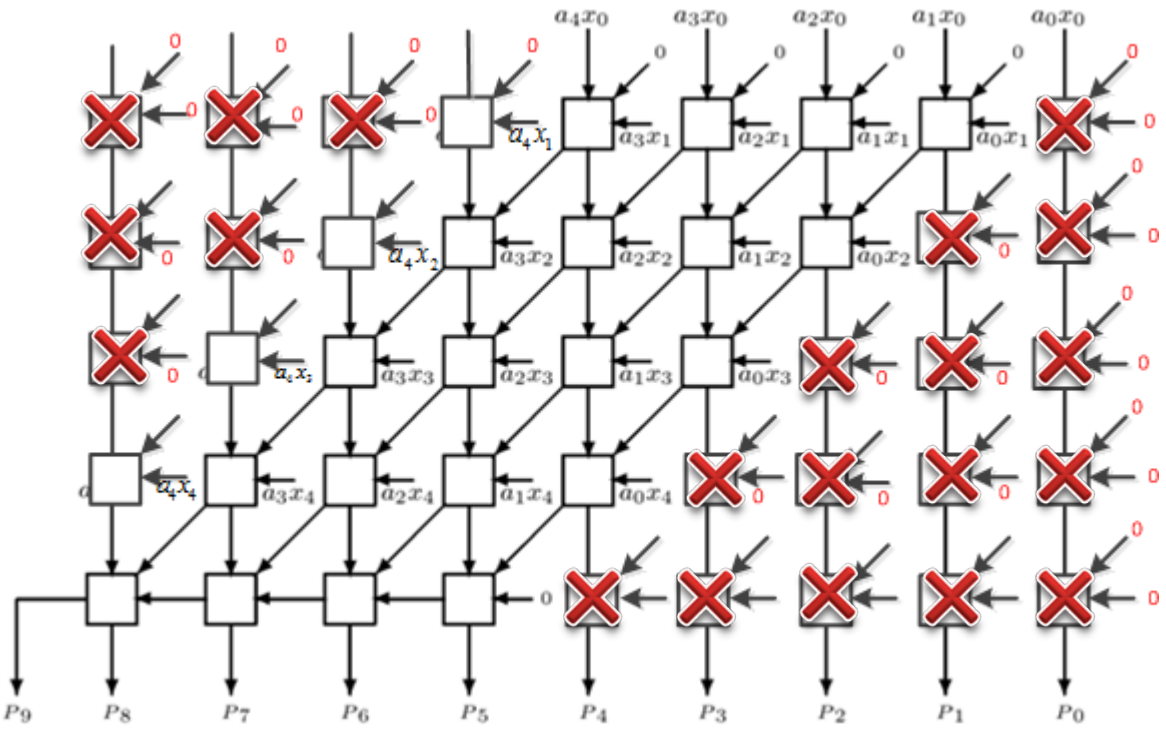
Multiplicación nxn

- Los extremos: p(0) y p(2*N-1) se tienen que asignar a mano



¿Qué pasa con todo el HW que hemos añadido de más?

- Durante la síntesis se elimina lógica “innecesaria”



¿Qué pasa con el HW que hemos añadido de más?

- Revisar los *warnings* para asegurarnos que no ha borrado celdas de más
- Revisar View RTL Schematics para asegurarnos que el circuito final es el esperado

¿Qué pasa con el HW que hemos añadido de más?

Práctica 3b

- Construir en VHDL un multiplicador nxn
  - Contestar preguntas de test 5 puntos
  - Si la práctica b os funciona, podéis enseñar directamente en la FPGA el apartado 2b para N=5, en lugar de enseñar el 2a (+ 2 ptos.)

```
entity multiplicador is
  generic ( N: integer := 5);
  port (a, x: in std_logic_vector(N-1 downto 0);
  p: out std_logic_vector((2*N)-1 downto 0));
end multiplicador;
```

Práctica 3c

- Avanzado:
  - En el laboratorio se pedirá a los alumnos una modificación del código presentado.
  - + 3 puntos