Circuitos Combinacionais Aritméticos

Pedroni – Capítulo 12

Prof. Odilson Tadeu Valle

Instituto Federal de Santa Catarina – IFSC
Campus São José
odilson@ifsc.edu.br

Conteúdo programático

Conteúdo programático

Adição Binária

O processo de adição binária:

1. Inicia pelo LSB.
2. O processo de soma acarreta em um *carry in* (bit de transporte - vai um).
3. Esse carry deverá ser considerado nos próximos bits somados (*carry out*).

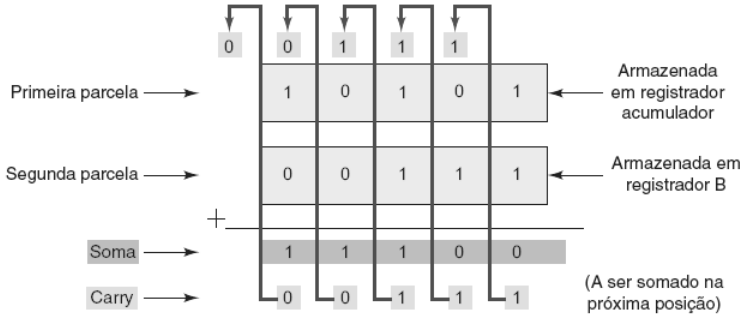
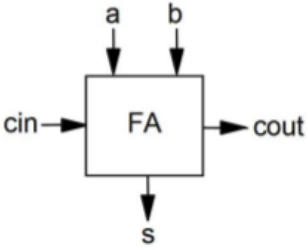


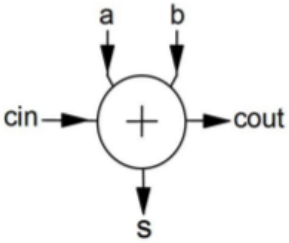
FIGURA 6.5 Processo típico de uma adição binária.

Somadores de um bit

- Há dois tipos de somadores de um bit:
  1. Somador completo de 1 bit (*full adder*, FA)
  2. Meio somador de 1 bit (*half adder*, HA)
- **Somador Completo**



(a)



(b)

(c)

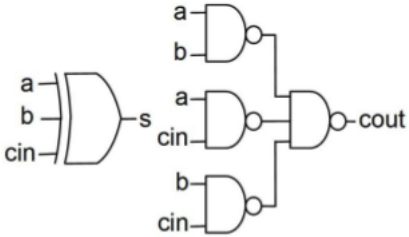
cin	a b	s	cout
0	0 0	0	0
0	0 1	1	0
0	1 0	1	0
0	1 1	0	1
1	0 0	1	0
1	0 1	0	1
1	1 0	0	1
1	1 1	1	1

Somador completo de um bit

$$s = a \oplus b \oplus cin$$

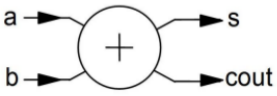
$$cout = a \cdot b + a \cdot cin + b \cdot cin$$

- *cin* (*carry-in*): bit de vem-um
- *cout* (*carry-out*): bit de vai-um
- Os circuitos lógicos para as saídas **s** e **cout** são apresentados na figura (d).



Meio somador de um bit

(a)



(b)

(c)

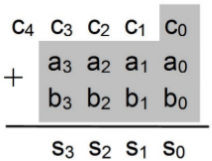
a b	s	cout
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1

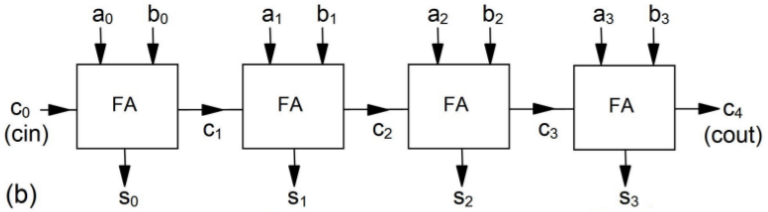
- Observe que a única diferença entre um FA e o HA é que neste não existe o bit de vem-um.
- Qual é a utilidade do Meio somador?
- Qual o circuito lógico que implementam as saídas e e cout?

Conteúdo programático

Somador carry-ripple

- O somador multibit mais simples é o somador carry-ripple.
- Na figura observa-se uma implementação para N = 4 bits.
- Para N bits, basta interligar em série N FAs.
- Como nesse circuito o carry tem que se propagar (ripple) serialmente por todos os estágios, esta é a arquitetura mais lenta para somadores multibit.
- Por outro lado, é o somador mais econômico em termos de área de silício e consumo energético.





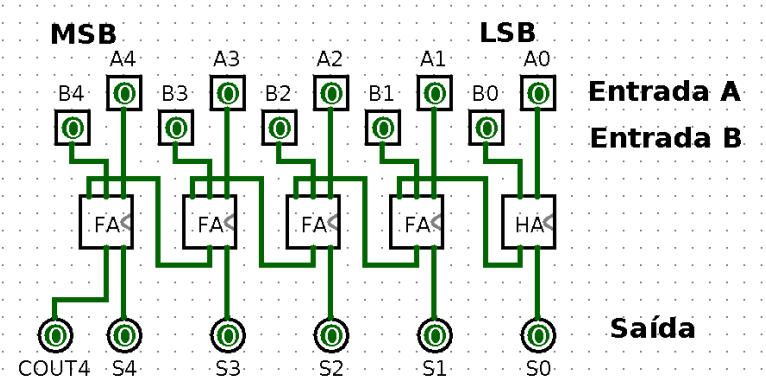
- Executar a soma entre a=1100 e b=1011 no circuito acima.

Exercício

Sabendo que o somador HA (*Half Adder*) tem um *hardware* mais simples que o FA (*Full Adder*), apresentados abaixo, monte o circuito mais simples (econômico) que consiga executar corretamente a soma $\mathbf{11001 + 11101}$. No circuito montado, defina (nomeie) todos os pinos de entrada e saída e discrimine, em cada um deles, o respectivo valor binário (bit) que represente as entradas e o resultado da soma solicitada.

Exercício

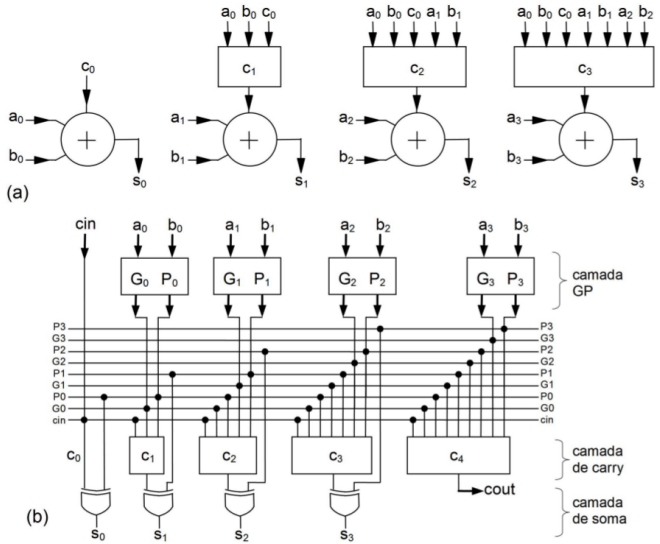
Sabendo que o somador HA (*Half Adder*) tem um *hardware* mais simples que o FA (*Full Adder*), apresentados abaixo, monte o circuito mais simples (econômico) que consiga executar corretamente a soma **11001 + 11101**. No circuito montado, defina (nomeie) todos os pinos de entrada e saída e discrimine, em cada um deles, o respectivo valor binário (bit) que represente as entradas e o resultado da soma solicitada.



Somador Carry-Lookahead

- No somador carry-ripple, cada FA depende da informação vinda dos FAs que o antecedem (carry-in) para poder executar seu cálculo.
- Diferentemente, no somador Carry-Lookahead cada célula computa seu próprio bit carry-in, ou seja, cada célula opera de maneira independente.

Somador Carry-Lookahead



Somador Carry-Lookahead

- Três camadas: GP (Sinais G e P), carry e soma
- Não há comunicação entre células de carry.
- Problemas:
  - Grande quantidade de hardware necessária para calcular o carry
  - Alto fan-in ==> reduz a velocidade
- Isso limita seu uso a blocos de cerca de 4 bits.

Sinais Generate e Propagate

- Como o caminho crítico em qualquer somador envolve o cálculo do carry, deve-se buscar arquiteturas que mitiguem esse problema.
- Nesse sentido define-se os sinais:
  - *Generate*(G): Deve ser 1 quando um carry-out deve ser produzido independente do carry-in, isto é, a e b são ambos 1:

$$G = a \cdot b$$

  - *Propagate*(P): Deve ser 1 quando a ou b for 1, caso em que $cout = cin$, isto é, deve-se permitir que cin se propague pelo circuito:

$$P = a \oplus b$$

Sinais Generate e Propagate

- Esses sinais podem ser relacionados com os do FA, resultando em:

$$s = a \oplus b \oplus cin \Rightarrow s = P \oplus cin$$

$$cout = a \cdot b + a \cdot cin + b \cdot cin = a \cdot b + (a + b) \cdot cin = a \cdot b + (a \oplus b) \cdot cin$$

$$\Rightarrow cout = G + P \cdot cin$$

- As equações acima podem ser entendidas como um circuito multibit. As expressões generalizadas resultantes são (i representa o i-ésimo estágio do somador):

$$s_i = P_i \oplus c_i$$

$$c_{i+1} = G_i + P_i \cdot c_i$$

onde:

$$G_i = a_i \cdot b_i$$

$$P_i = a_i \oplus b_i$$

- Observe que G_i e P_i não dependem do carry.

Sinais Generate e Propagate em um somador de 4 bits

$$s_0 = P_0 \oplus c_0$$

$$s_1 = P_1 \oplus c_1$$

$$s_2 = P_2 \oplus c_2$$

$$s_3 = P_3 \oplus c_3$$

$$c_0 = cin$$

$$c_1 = G_0 + P_0 \cdot c_0$$

$$c_2 = G_1 + P_1 \cdot c_1$$

$$c_3 = G_2 + P_2 \cdot c_2$$

$$c_4 = G_3 + P_3 \cdot c_3$$

Sinais Generate e Propagate em um somador de 4 bits

Desenvolvendo as equações para os carry temos:

$$\boxed{c_0 = cin}$$

$$\boxed{c_1 = G_0 + P_0 \cdot c_0}$$

$$c_2 = G_1 + P_1 \cdot c_1 = G_1 + P_1 \cdot (G_0 + P_0 \cdot c_0)$$

$$\boxed{c_2 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot c_0}$$

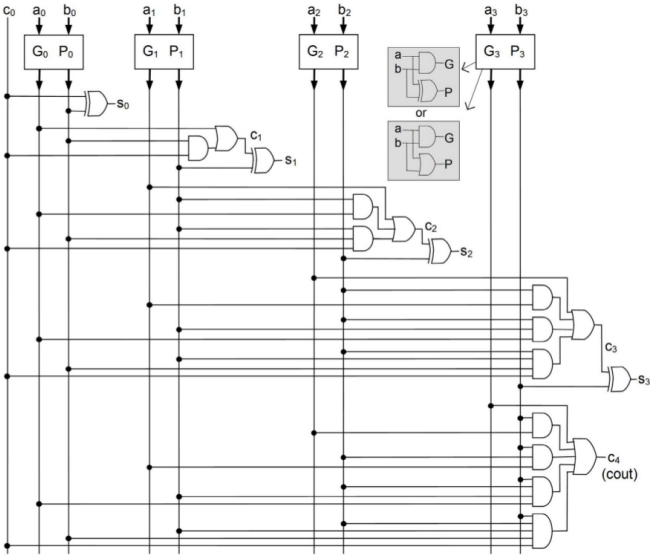
$$c_3 = G_2 + P_2 \cdot c_2 = G_2 + P_2 \cdot (G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot c_0)$$

$$\boxed{c_3 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot c_0}$$

$$c_4 = G_3 + P_3 \cdot c_3 = G_3 + P_3 \cdot (G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot c_0)$$

$$\boxed{c_4 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot c_0}$$

Somador Carry-Lookahead de 4 bits

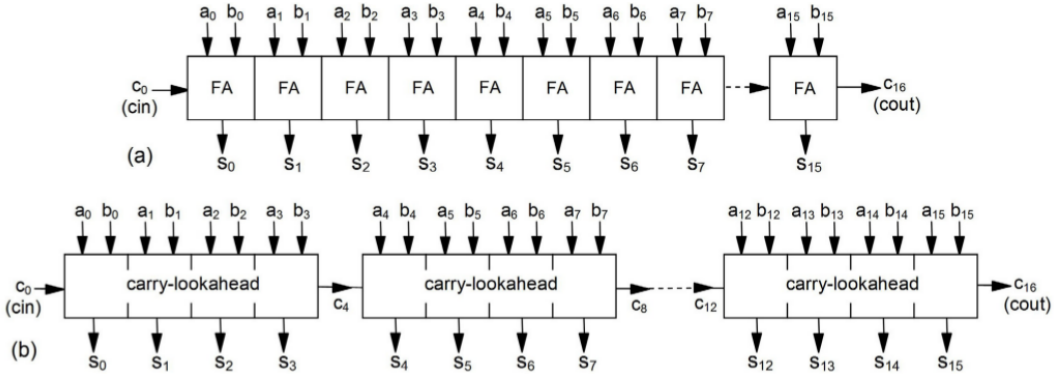


Somador Lookahead Grandes

- Como visto, somadores Lookahead com mais de 4 bits não obedecem completamente a filosofia dos somadores carry-Lookahead.
- De maneira resumida, há duas maneiras de construir somadores Lookahead grandes:
  1. Ligar diversos somadores Carry-Lookahead de 4 bits em série.
  2. Utilizar um grande número de unidades Generate/Propagate, interligados em forma de árvore.

Conteúdo programático

Somadores Grandes



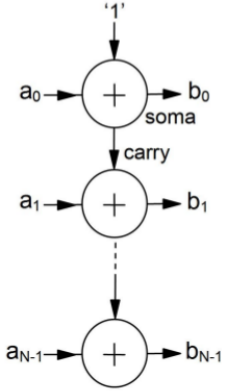
a) Somador inteiramente carry-ripple: i) células iguais, ii) requer menos área de silício, iii) baixo consumo energético e iv) lento.

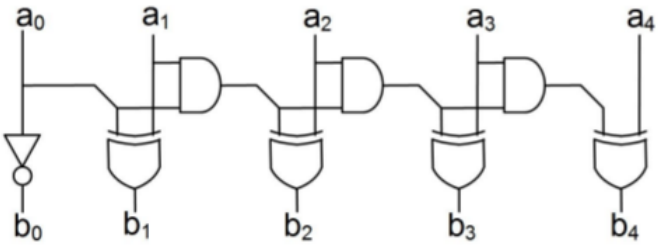
b) Associação de somadores Carry-Lookahead de 4 bits: Carry-Lookahead internamente, carry-ripple externamente.

Conteúdo programático

Incrementador

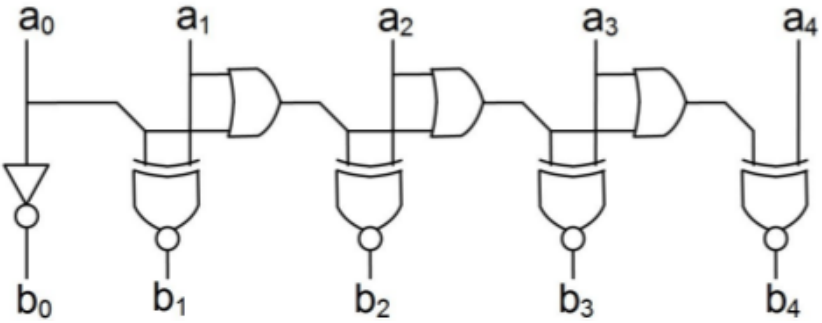
- Para incrementar um número, isto é, somar 1 a ele, pode-se utilizar o circuito mostrado na figura.
- É composto por uma série de somadores parciais (HA) de 1 bit ou de um circuito específico.





- Incrementar o número 6

Decrementador



Decrementar o número 6

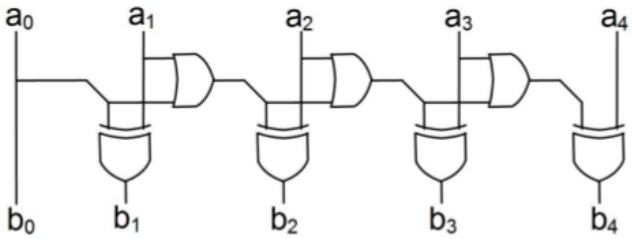
Complementador de Dois

A operação Complemento de dois ($b = a' + 1$) pode ser implementada de duas formas:

- Através da inversão da entrada + um circuito incrementador.
- Ou através de um decrementador com saída invertida.
  - Como $a' = (2^N - 1)_2 - a$ e $(2^N - 1)_2 = 11\ldots11$, pode-se reescrever a expressão para:

$$b = a' + 1 = (2^N - 1)_2 - a + 1 = (2^N - 1)_2 - (a - 1) = 2^N - 1 - a_{decrem} = a'_{decrem}$$

  - Consequentemente, basta inverter as saídas do decrementador para obter o complementador de dois.



- Complementar o número 5

Conteúdo programático

Somadores/subtratores com sinal

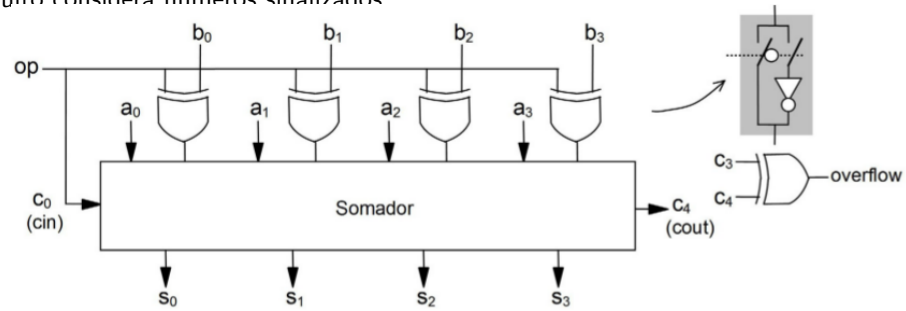
- Em um sistema aritmético com sinal, qualquer valor positivo é representado do mesmo modo que um valor sem sinal, enquanto qualquer número negativo é representado em forma de complemento de dois.
- Para **somar** dois valores, sejam positivos ou negativos, podemos utilizar qualquer somador já estudado, desde que os números negativos estejam representados em complemento de dois, por exemplo: 7 + (-5).
- Por outro lado, para realizar uma operação de **subtração**, por exemplo $a - b$, devemos primeiro obter o complemento dois de b, independente deste ser positivo ou negativo, e então utilizar um somador para proceder a soma. Exemplo, a = 5 (0101) e b = -2 (1110):

$$a - b = (5) - (-2) = (5) + (\textit{complemento de dois de} - 2)$$
$$= 0101 + 0010 = 0111 = 7$$

- Um subtrator pode ser obtido pela combinação de um circuito complementador de dois com um somador qualquer.

Circuito Somador e/ou Subtrator: $a \pm b$

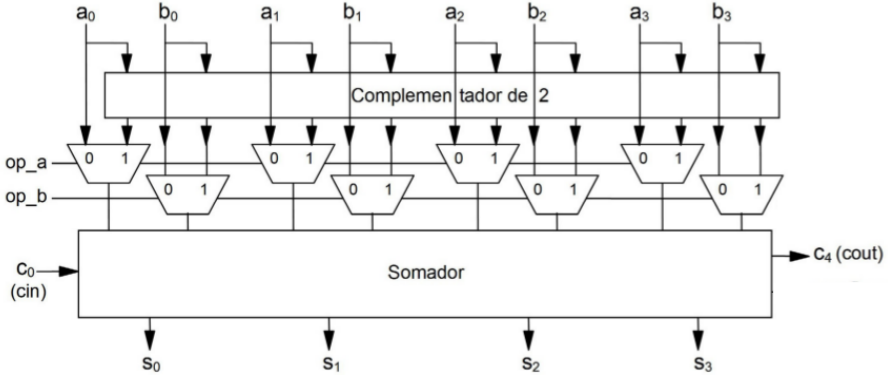
- Na figura são admitidos dois modos de operação: $a \pm b$.
  - Quando o operador (op) é 0, as saídas das portas XOR são iguais as entradas e cin =0, resultando numa adição normal, ou seja, $a + b$.
  - Quando o operador (op) é 1, as portas XOR invertem as entradas e cin =1, resultando em $a + b' + 1$ ($b' + 1$ complemento de dois de b), ou seja, $a - b$.
  - Circuito considera números sinalizados



- Analise o circuito para a=1001, b=0101 e op: 0 e 1.

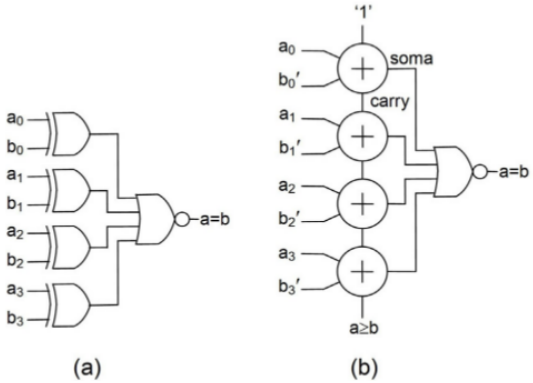
Circuito Somador e/ou Subtrator: $\pm a \pm b$

- Somador/subtrator para números sinalizados completamente genérico: $\pm a \pm b$ com carry-in. *Overflow*: $c_3 \oplus c_4$
  - Possui um complementador de dois, com todas as entradas passando por ele e indo ao somador.
  - Multiplexadores controlados pelos operadores *op_a* e *op_b* são utilizados para definir quais sinais devem entrar no somador: *op* baixo = adição, *op* alto = subtração.



Conteúdo programático

Comparadores

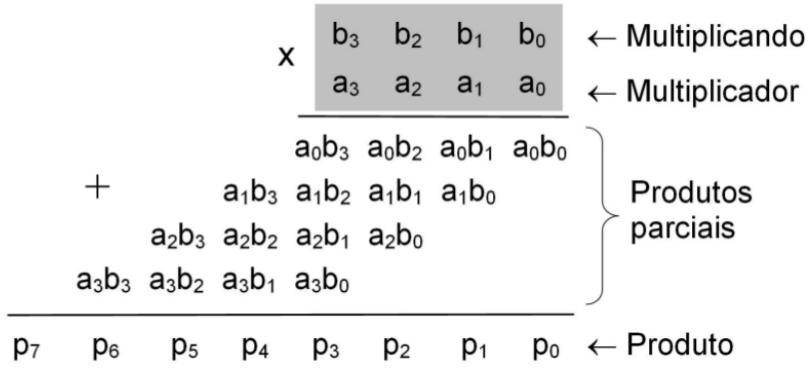


a) Comparador de igualdade, bit a bit
b) Comparador de igualdade e de grandeza relativa sem signal
c) Executar: a=5, b=4 e a=4, b=4.

Conteúdo programático

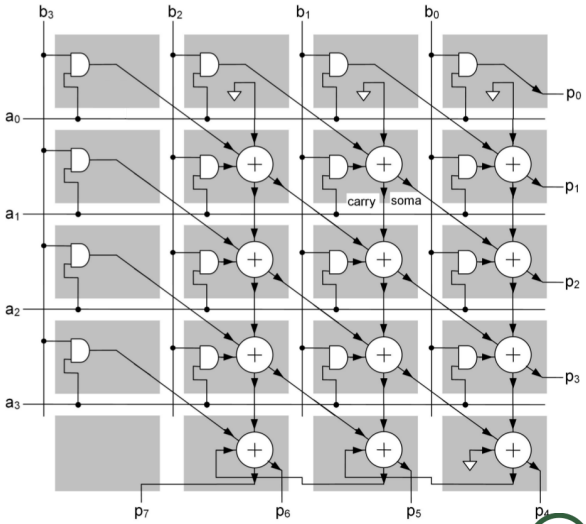
Multiplicadores

- Entradas $a = a_3a_2a_1a_0$ (multiplicador) e $b = b_3b_2b_1b_0$ (multiplicando)
- Saída $p = p_7p_6p_5p_4p_3p_2p_1p_0$ (produto).
- Para entradas de N bits de números não sinalizados, a saída deve ter 2N bits de largura para evitar *overflow*.

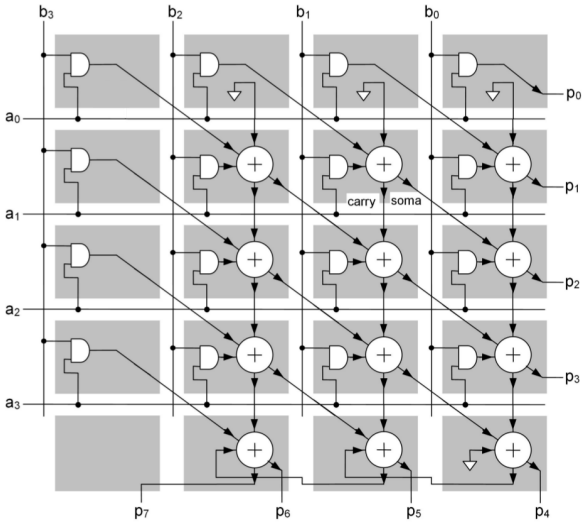


Multiplicador paralelo sem sinal

- Circuito com entradas e saídas paralelas (*array multiplier*)
- O circuito é combinacional
- Emprega um conjunto de portas AND e somadores de 1 bit (FA)
- $p_0 = a_0 b_0, p_1 = a_0 b_1 + a_1 b_0, p_2 = a_0 b_2 + a_1 b_1 + a_2 b_0 + carry(p_1) \ldots .$
- Funciona somente com entradas não negativas, isto é, números sem signal.



Multiplicador paralelo sem sinal



- Executar:5×7

Circuitos aritméticos comerciais

http://wiki.sj.ifsc.edu.br/wiki/index.php/CIL-EngTel_(p%C3%A1gina)#Circuitos_Aritm.C3.A9ticos_Combinacionais

Circuitos aritméticos comerciais

http://wiki.sj.ifsc.edu.br/wiki/index.php/CIL-EngTel_(p%C3%A1gina)#Circuitos_Aritm.C3.A9ticos_Combinacionais

1. Faça o diagrama de ligações para montar um somador de 8 bits a partir do 74X283.