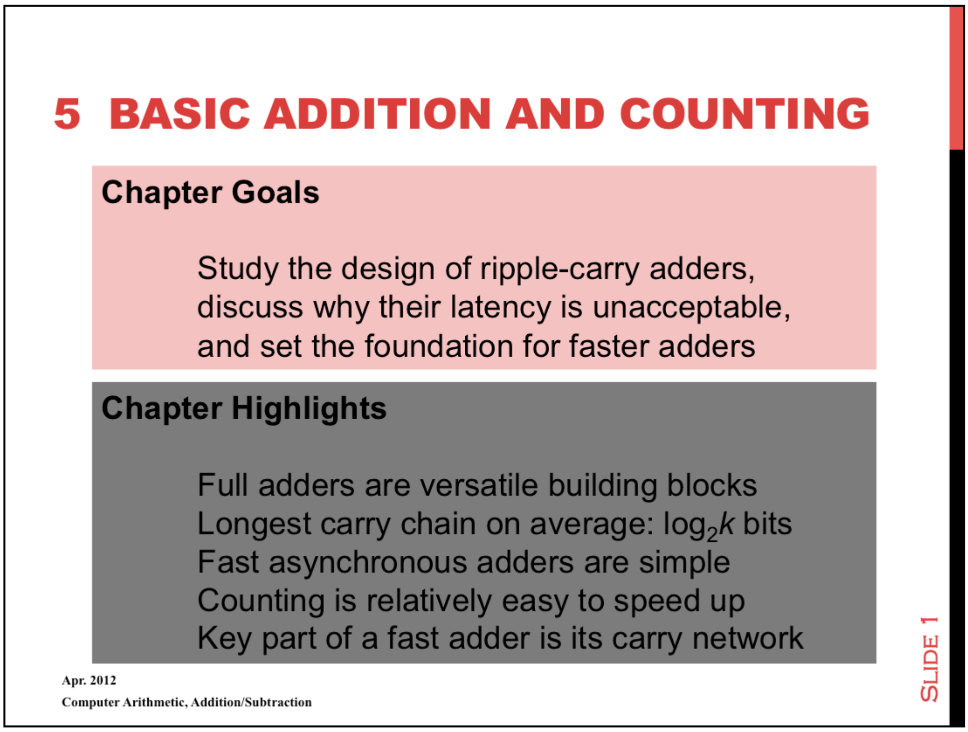


Neste capitulo vamos estudar os princípios básicos de adição e contagem. Veremos como realizar os circuitos para tais princípios.

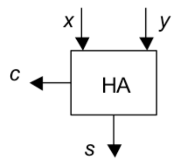
BASIC ADDITION AND COUNTING: TOPICS

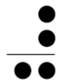
Topics in This Chapter
5.1 Bit-Serial and Ripple-Carry Adders
5.2 Conditions and Exceptions
5.4 Carry Completion Detection
5.5 Addition of a Constant
5.6 Manchester Carry Chains and Adders

Vejamos então os tópicos abordados.

5.1 BIT-SERIAL AND RIPPLE-CARRY ADDERS

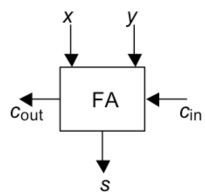
Inputs		Outputs	
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0





Half-adder (HA): Truth table and block diagram

Inputs			Outputs	
x	y	c_{in}	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



Full-adder (FA): Truth table and block diagram

Meio-somadores (HAs) e somadores completos (FAs) são blocos versáteis usados na síntese de somadores e muitos outros circuitos aritméticos.

Um HA recebe dois bits de entrada x e y, produzindo um bit de soma s = “x” xor “y” e um bit de carry c = “x” and “y”.

Já em um FA inserimos o conceito de carry in. Protanto teremos a soma de 3 elementos reduzindo para dois.

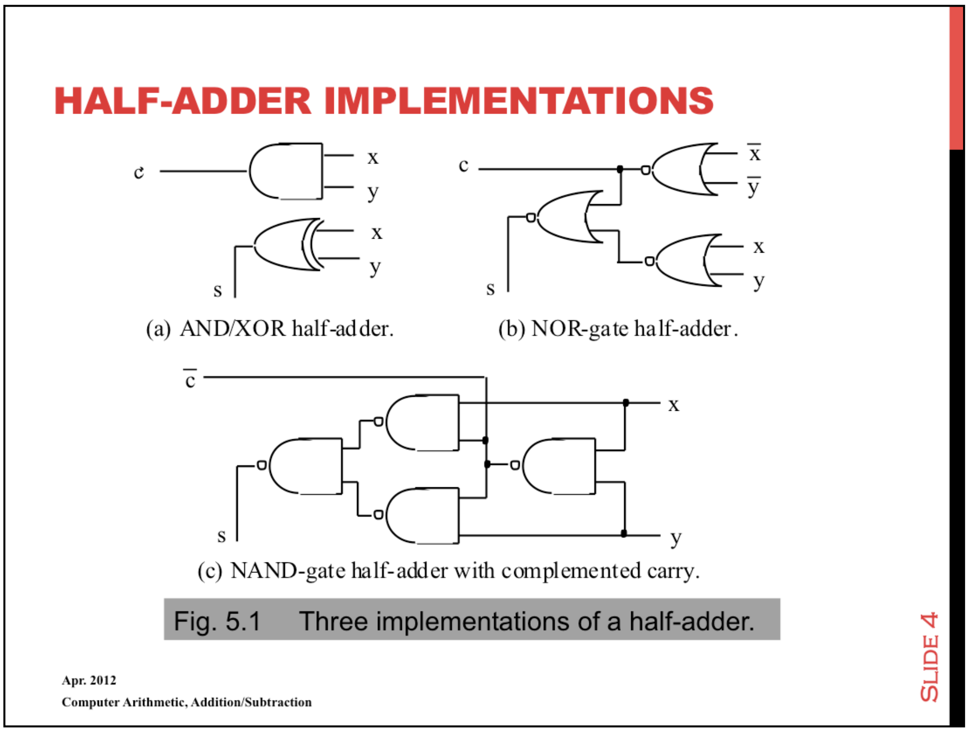
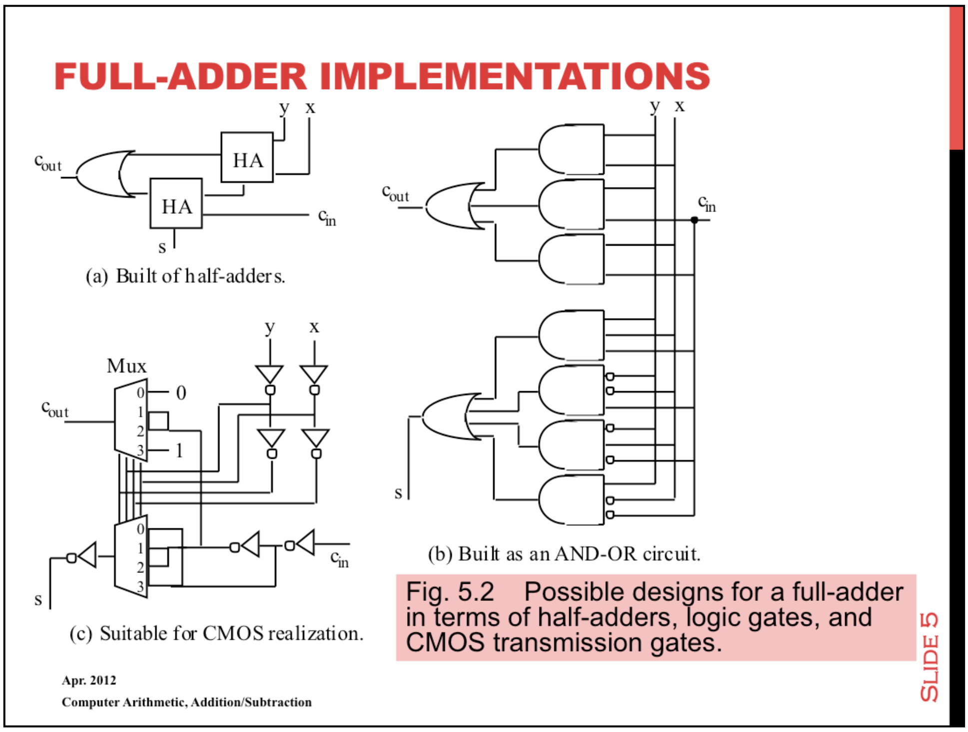


Fig. 5.1 Three implementations of a half-adder.

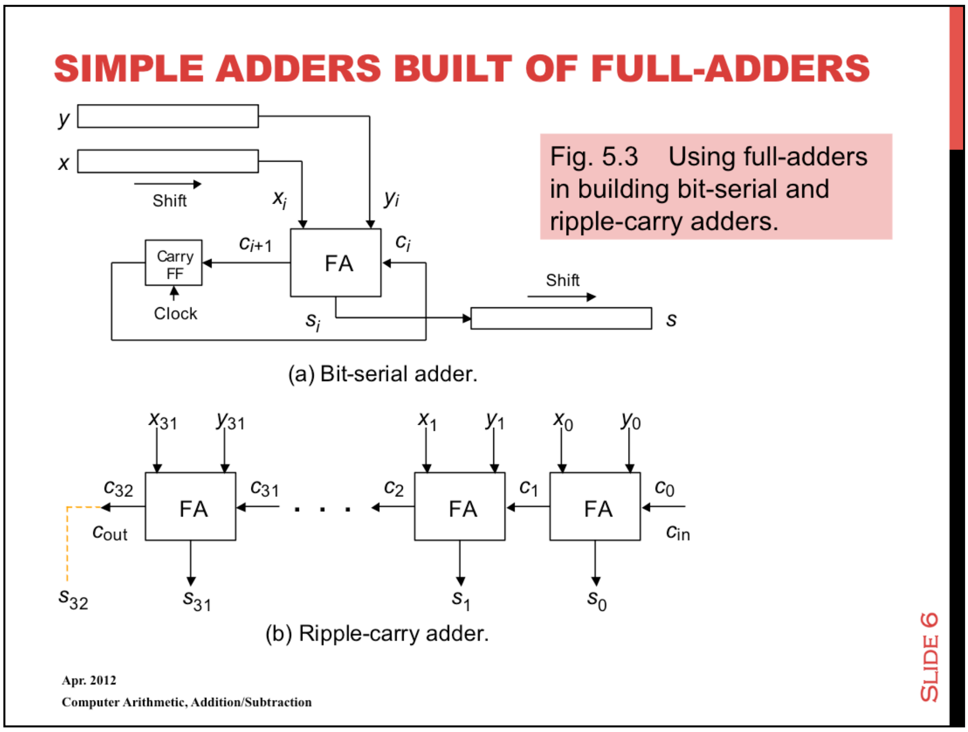
O slide acima descreve três das muitas maneiras para realizações lógicas possíveis de um HA. Um HA pode ser visto como um somador binário que produz a soma de 2 bits de cada uma das suas duas entradas.

A primeira abrodagem, mesmo com apenas duas portas, não indica que seja a mais otimizada. Dependera da tecnologia. Por exemplo as portas NANDs são muito menos custosas que a AND e XOR.



Da mesma forma são apresentados os esquemas para o FA. a) usando HAs e OR-2, b) usando AND-OR aproach, b) e finalmente usando MUX na abordagem.

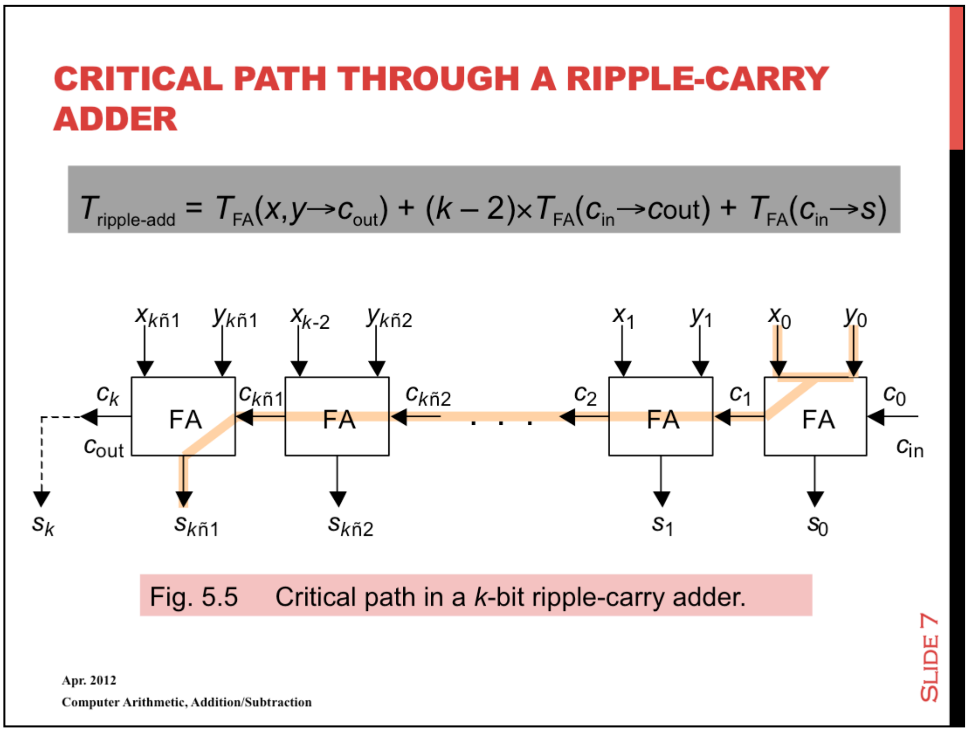
Os custos e atrasos são muito diferentes nas três estruturas escolhidas. Desta forma vemos que podemos trocar área por velocidade. A escolha de alguma delas dependera dos requisitos do projetista.



Somadores completos (FA) e meio-somadores (HA) podem ser usados para realizar uma variedade de funções aritméticas.

Por exemplo, um somador bit-serial pode ser construído a partir de um FA e um carry flip-flop, conforme mostrado na Fig. 5.3(a). Os operandos são fornecidos ao FA, 1 bit por ciclo de clock, começando com o bit menos significativo, a partir de um par de registradores de deslocamento, e a soma é deslocada para um registrador de resultado. A adição de números de k bits pode, portanto, ser concluída em k ciclos de clock.

Um somador binário “ripple-carry” de k-bit requer k FAs, com o carry-out do i-ésimo FA conectado à entrada de carry-in do (i + 1)-ésimo FA. O somador de k bits resultante produz uma saída de soma de k bits e um carry-out. Alternativamente, Cout pode ser visto como o bit mais significativo de uma soma de (k + 1) bits. A Figura 5.3b mostra um somador ripple-carry para operandos de 4 bits, produzindo uma soma de 4 ou 5 bits.

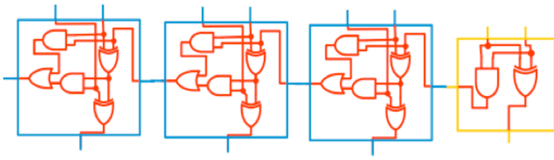


- A latência de um somador “ripple carry” de k bits pode ser derivada considerando o pior caso de caminho de propagação de sinal. Conforme mostrado na Fig. 5.5, o caminho crítico geralmente começa na entrada x0 ou y0, prossegue pela cadeia de propagação de transporte até o FA mais à esquerda e termina na saída sk − 1.

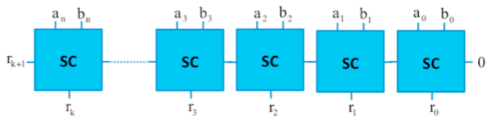
- Vemos que a latência cresce linearmente com k, tornando o projeto ripple-carry indesejável para valores de k muito grandes ou para unidades aritméticas de alto desempenho.

PROBLEMAS

Problema 5.1 Qual o atraso do caminho crítico do circuito abaixo, considerando que cada porta lógica resulta em um atraso de 3 ns?

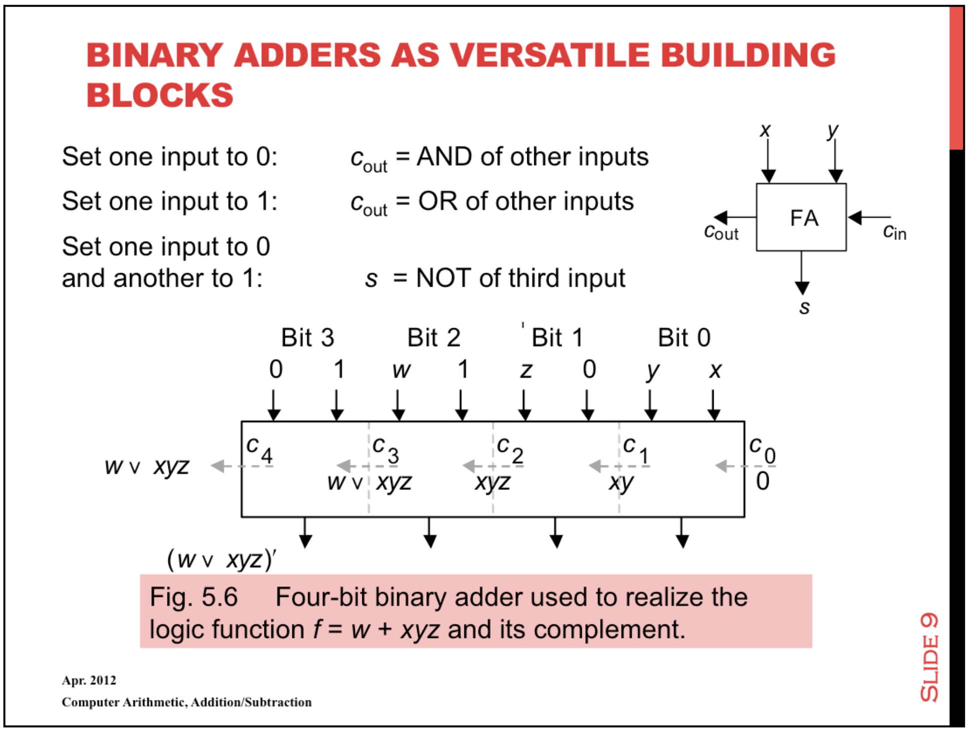


Problema 5.2 Qual o atraso do caminho crítico do circuito abaixo, considerando que cada bloco resulta em um atraso de 5 ns?



SLIDE 8

Gabarito no Moodle



FA, Ha e somadores multibits, são blocos de construção poderosos que também podem ser usados na realização de funções não aritméticas, se necessário.

Por exemplo, um somador binário de 4 bits com cin, duas entradas, de operando de 4 bits, cout e uma saída de soma de 4 bits pode ser usado para sintetizar a função lógica de quatro variáveis f = w + xyz e seu complemento, conforme representado na Fig. 5.6.

PROBLEMAS

Problema 5.5. Usando unicamente um somador de 4 bits implemente:

a)Um somador de 3 bits, com *carry-in* e *carry-out*;

b)Dois somadores independentes de 1 bit (*Full-Adder*);

c)Um somador de um bit (*Full-Adder*) e um somador de dois bits operando de forma independente.

d)Um gerador de imparidade de 4 bits (4-bit XOR).

e)Dois geradores independentes de imparidade de 3 bits.

f) Uma porta AND de 5 entradas.

g)Uma porta OR de 5 entradas.

h)Um circuito que implemente a função lógica de 4 variáveis $wx+yz$.

i) Um circuito que implemente a função lógica de 4 variáveis $wxy+wxz+wyz+xyz$.

j) Um multiplicador $f=15y$, onde entrada y é de dois bits e f de 6 bits.

k)Um circuito que compute $x+4y+8z$, onde x, y, e z são números de 3 bits sem sinal.

l) Um contador paralelo de 5 entradas, produzindo uma saída de 3 bits.

SLIDE 10

Gabarito no Moodle

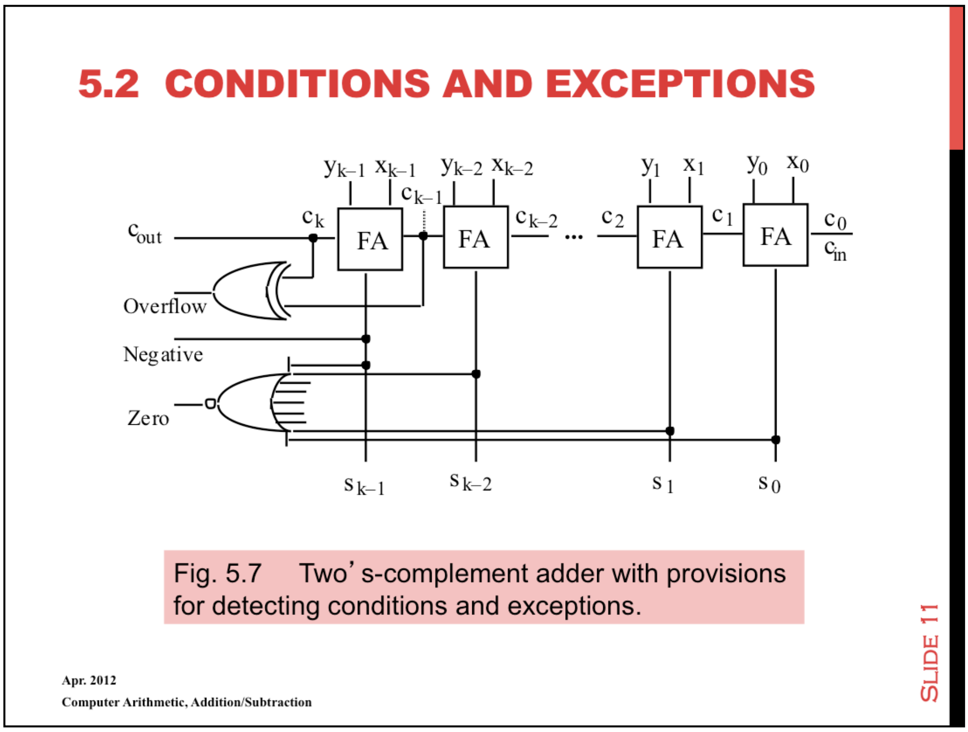


Fig. 5.7 Two's-complement adder with provisions for detecting conditions and exceptions.

Quando um somador de k bits é usado em uma unidade logica aritmética (ALU), é comum fornecer a soma de k bits em conjunto com informações sobre os seguintes resultados:

Overflow: Quando a soma excede a representação maxima.

Negative: Indica se temos um numero positivo ou negativo (caso usemos Complemento a 2)

Zero: Se a saida é zero.

Na adição em complemento de 2, cout não tem significância. No entanto, uma vez que um único somador é frequentemente usado para adicionar números sem sinal e complementos de 2, cout também é uma saída útil. A Figura 5.7 mostra uma implementação de um somador "ripple-carry" sem sinal ou um somador de complemento de 2 com saídas auxiliares para condições e exceções. Por causa do grande número de entradas na porta NOR que testa para o valor 0, ela deve ser implementada como uma árvore OR seguida por um inversor.

PROBLEMAS

Problema 5.6. Considere os seguintes números, representados com 4 bits em complemento para dois:

A = 0011 ; B = 1001

Indique, para a operação A + B:

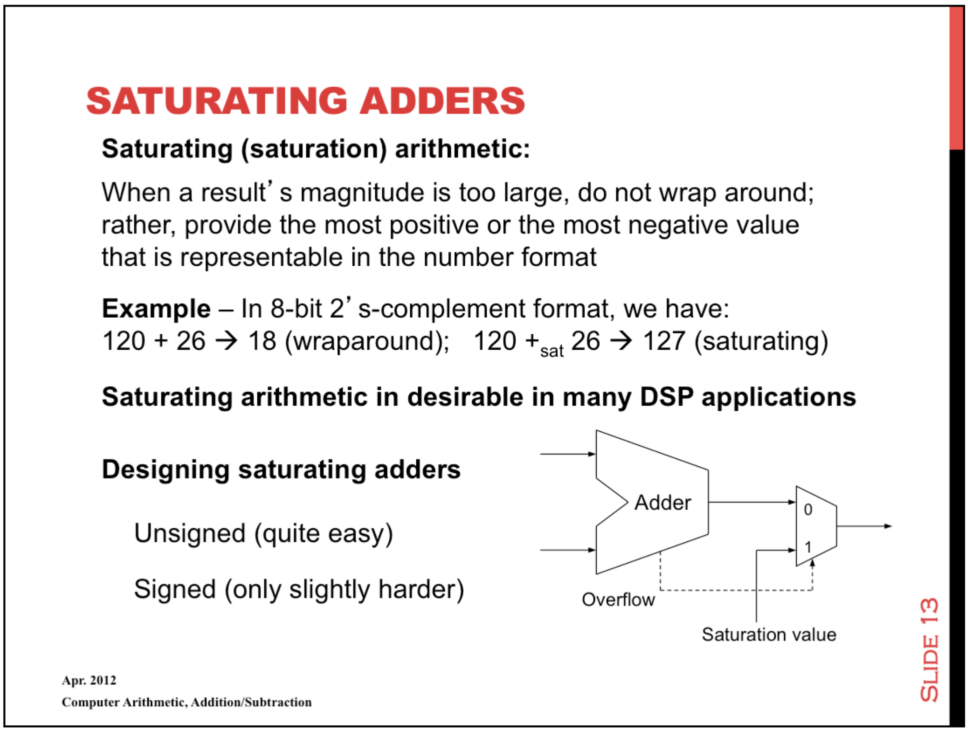
a)o vector de soma (S) resultante;

b)o vector constituído pelos vários bits de transporte (C) gerados ao longo da operação;

o valor das *flags* zero (Z), negativo (N) e *overflow* (V) à saída da unidade aritmética.

SLIDE 12

Gabarito no Moodle



Quando a soma dos operandos de entrada sem sinal é muito grande para representação em k bits, uma exceção de "estouro" (overflow) é indicada pelo sinal cout na Fig. 5.5 e um valor "empacotado" (wrapped), que é 2k menor do que a soma correta, aparece como a saída. Um valor empacotado semelhante pode aparecer para adição com sinal, no caso de estouro.

Um somador que leva em conta a saturação (overflow) pode ser obtido a partir de qualquer projeto de somador usando um multiplexador na saída, com sua entrada de controle ligada ao sinal de estouro do somador.

PROBLEMAS

Problema 5.7. Usando a ideia de somadores com saturação:

a) Implemente a operação $A+B$ (4 bits) quando B é par e $2(A+B)$ quando B é impar.

b) Adicione um valor de saturação de 15_{10} quando exista *overflow*.

SLIDE 14

Gabarito no Moodle

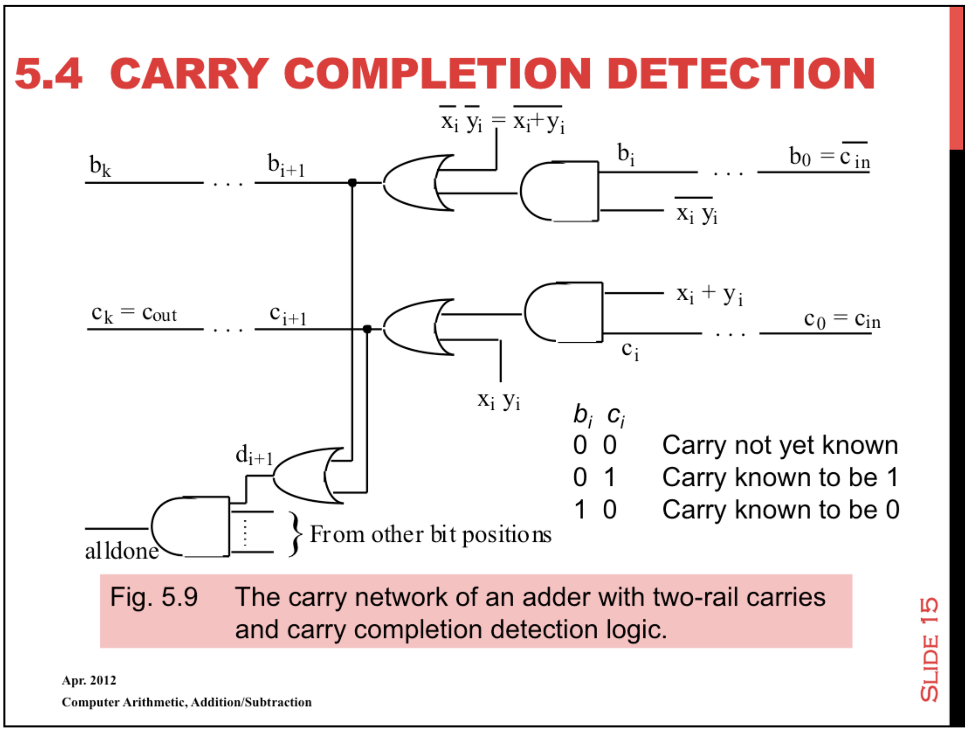


Fig. 5.9 The carry network of an adder with two-rail carries and carry completion detection logic.

Um somador com "carry completion detection" retira vantagem do comprimento médio da cadeia de carry mais longa para adicionar dois números binários de k bits em tempo mais rápido quando comparado ao somador ripple carry comum.

Este somado apresnetado no slide é essencialmente um somador ripple carry em que um carry de 0 também é explicitamente representado e pode se propagar entre os estágios.

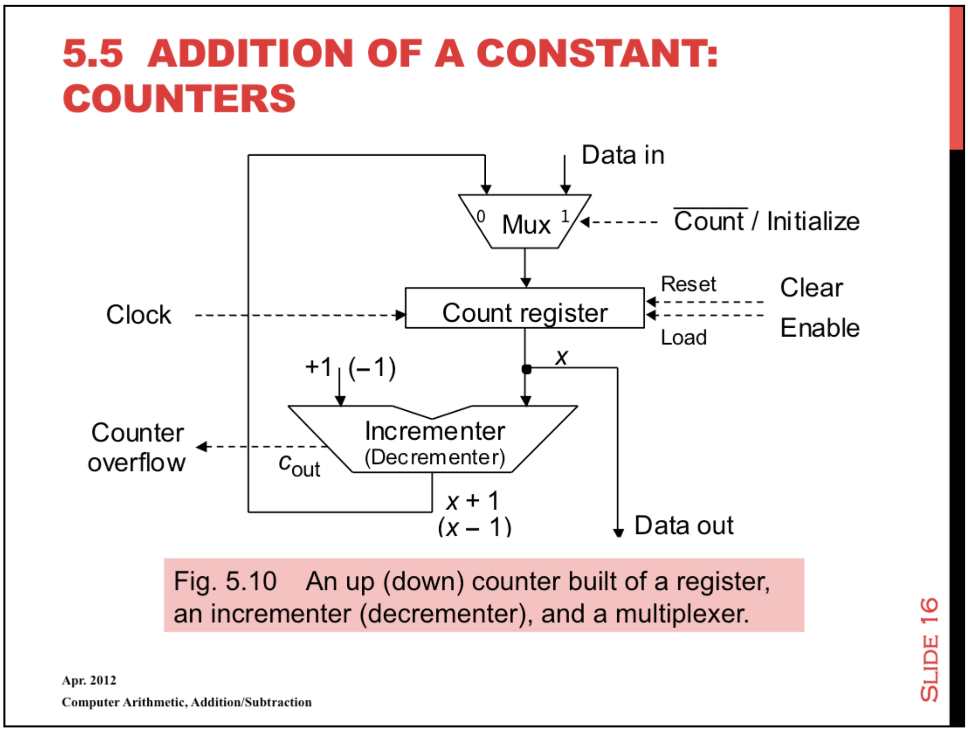
O determinado carry para o estágio i é representado pelo código que indicar (carry ainda não conhecido, carry igual a zero ou igual a um).

Desta forma, como dois bits iguais a 1 nos operandos geram um transporte de 1 que se propaga para a esquerda, dois bits iguais a zero produziriam um transporte de 0. Inicialmente, todos os transportes são (0, 0) ou desconhecidos.

Após a inicialização, uma posição de bit com xi = yi faz a determinação se há carry ou não, e injeta o carry apropriado na cadeia de propagação de carry da Figura 5.9 por meio das portas OR.

Quando cada carry assumiu um dos valores (0, 1) ou (1, 0), a propagação estará completa.

Os sinais locais "feitos" são combinados por uma função AND global em "all done", o que indica o fim da propagação.



Quando uma das entradas da operação de adição é um número constante, o projeto de hardware pode ser simplificado ou otimizado em comparação com o de um somador geral de dois operandos.

Considere uma constante “y” a ser adicionada a “x”. O bit menos significativo da soma é x0. Os bits restantes de s podem ser determinados por um somador “ripple-carry” (k - 1) -bit, com cin = x0, com cada uma de suas células sendo um HA (yi = 0) ou um HA modificado (yi = 1).

Desta forma iremos realizar uma contagem um determinado número de vezes igual a essa constante a qual deseja-se somar à variável. O circuito resultante é conhecido como incrementador (decrementador) e é usado no projeto de contadores.

PROBLEMAS

Problema 5.8. Usando incrementadores/decrementadores, registradores, deslocadores e multiplexadores implemente os sequenciadores:

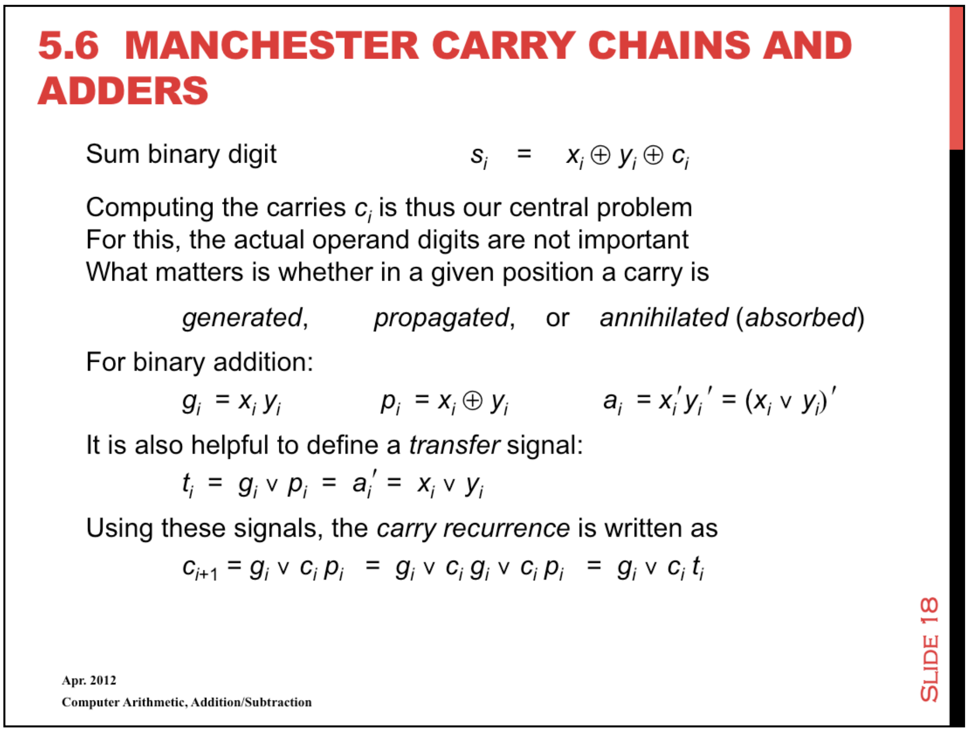
a) 0 → 2 → 6 → 14 → 30 → ...

b)0 → 3 → 7 → 15 → 31 → ...

c)4 → 6 → 10 → 18 → 34 →...

SLIDE 17

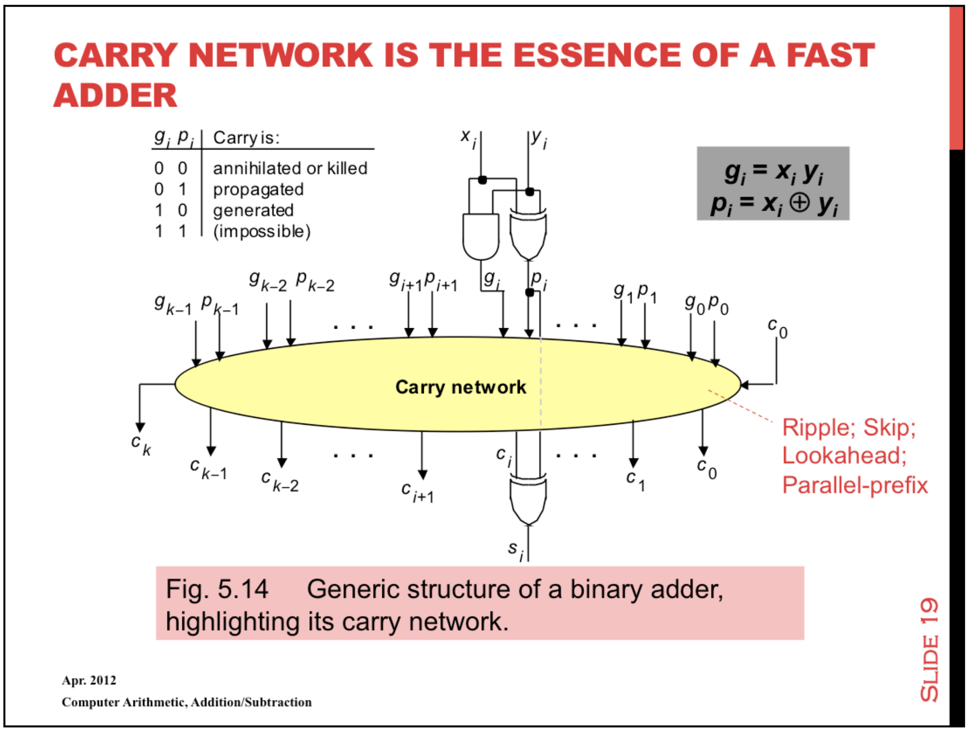
Gabarito no Moodle



Para dois operandos, a chave para uma adição rápida é uma rede de transporte de carry de baixa latência, uma vez que o carry para a posição i é conhecido, o dígito da soma pode ser determinado a partir dos dígitos do operando xi e yi e o carry de entrada ci.

Do ponto de vista da propagação do carry, os dígitos reais do operando não são importantes. O que importa é se em uma determinada posição um “carry" é gerado, propagado ou aniquilado (absorvido). O slide então demonstra as equações associadas a cada um desses elementos respectivamente. (g, p, a)

Então, para enfrentar o problema de progação do carry, utilizamos a equação descrita no slide como recorrência de carry. Esta recorrência afirma essencialmente que um transporte entrará no estágio i + 1 se for gerado no estágio i ou se entrar no estágio i e for propagado por aquele estágio.



A partir do que foi descrito no slide anterior, podemos tirar vantagem dos sinais de geração e propagação.

Desta forma, como demonstrado na figura 5.14, este somador terá dois conjuntos de portas AND e XOR na parte superior para formar os sinais "gi" e "pi", e terá um conjunto de portas XOR na parte inferior para produzir os bits de soma "Si".

RIPPLE-CARRY ADDER REVISITED

The carry recurrence: $c_{i+1} = g_i \vee p_i c_i$

Latency of k-bit adder is roughly $2k$ gate delays:

1 gate delay for production of p and g signals, plus
$2(k - 1)$ gate delays for carry propagation, plus
1 XOR gate delay for generation of the sum bits

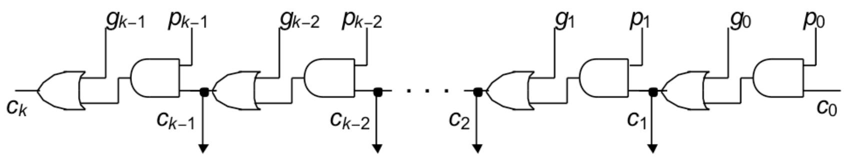


Fig. 5.15 Alternate view of a ripple-carry network in connection with the generic adder structure shown in Fig. 5.14.

Apr. 2012
Computer Arithmetic, Addition/Subtraction

SLIDE 20

Entretanto, no projeto de uma rede de carry, que é representada pelo grande bloco oval na Fig. 5.14 podemos utilizar diferentes soluções. Por exemplo, um somador “ripple carry” pode ser visto como tendo a rede de transporte mostrada na Fig. 5.15.

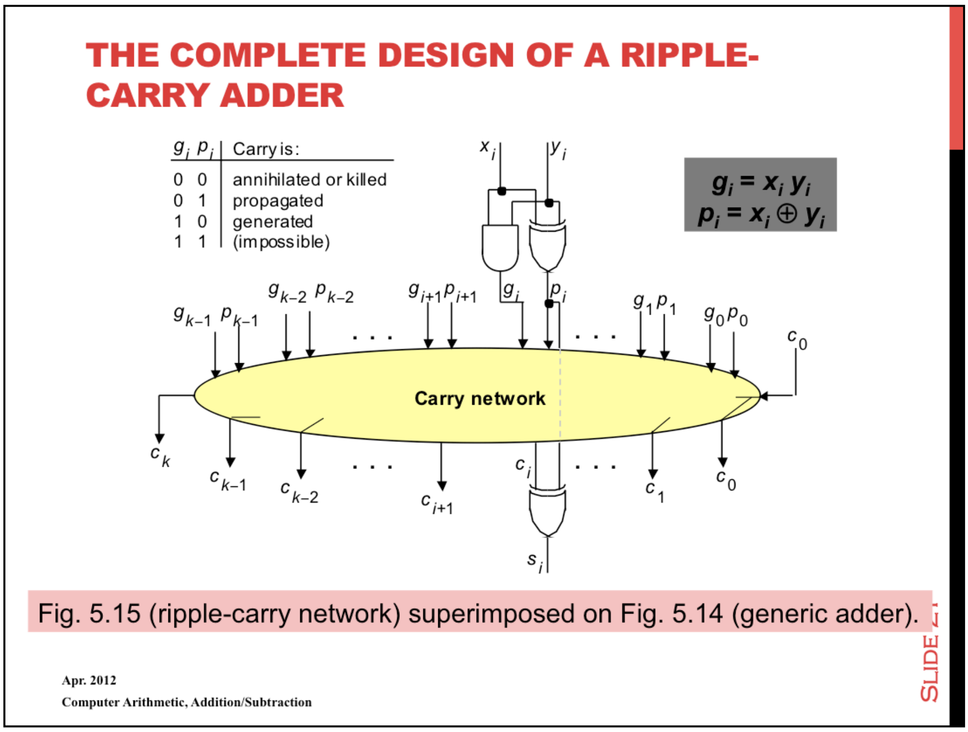


Fig. 5.15 (ripple-carry network) superimposed on Fig. 5.14 (generic adder).

A inserção da rede de carry utilizando “ripple carries” no projeto genérico da Fig. 5.14 produzirá um somador completo.