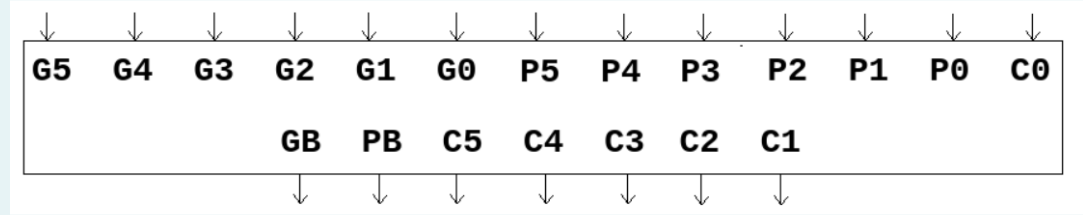


Suponha que queremos construir um somador binário de  $n$  bits, com entradas  $A = A_{n-1} \dots A_0$  e  $B = B_{n-1} \dots B_0$ , além do carry-in  $C_0$ , e saídas  $S = S_{n-1} \dots S_0$ , além do carry-out  $C_n$ . Para tal, poderíamos simplesmente cascatear  $n$  somadores completos de 1 bit. Entretanto, esta abordagem faria com que  $C_n$  tivesse um atraso linearmente proporcional a  $n$ .

O cálculo do vai-um antecipado (carry look-ahead) permite reduzir este atraso. Esta técnica utiliza os bits  $P_i$  e  $G_i$ , que denotam quando a soma no  $i$ -ésimo bit propaga ou gera um carry, e que podem ser definidos, para  $1 \leq i \leq n$ , como:

$P_i = A_i + B_i$  e  $G_i = A_i \cdot B_i$

De posse desses bits, todos os  $C_i$  podem ser calculados com mais 2 camadas de portas lógicas. Circuitos de carry look-ahead fazem exatamente isso, tipicamente com uma diferença: a saída  $C_n$  é trocada por  $P_B$  e  $G_B$ , denotando quando o bloco como um todo propaga ou gera um carry. A figura abaixo mostra um tal circuito com 6 bits, que chamaremos de CLA6:



Note que  $G_B$  e  $P_B$  podem ser calculados em função dos  $G_i$ 's e dos  $P_i$ 's e podem ser usados para calcular  $C_6$ . Determine o atraso das saídas de carry de um somador de 6 bits, que usa CLA6, em relação às entradas  $A$ ,  $B$  e  $C_0$  (fornecidas em  $t = 0$ ).

Carry look-ahead:  $C_1 = G_0 + P_0 \cdot C_0$ ,  $C_2 = G_1 + P_1 \cdot C_1 =$   
 $G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0) =$   
 $G_1 + P_1 G_0 + P_1 P_0 C_0$

$\therefore C_N = G_N + P_N \cdot C_0$

$\{ G_N = G_{N-1} + P_{N-1} \cdot G_{N-2} + P_{N-1} P_{N-2} \cdot G_{N-3} + \dots$   
 $P_N = P_{N-1} \cdot P_{N-2} \cdot P_{N-3} \cdot P_{N-4} \dots$

Para 3: Latência de  $G_n$  e  $P_n$  {  
Latência + 1 {  
Latência + 2 {  
...

The logic diagram shows the internal structure of the CLA6 circuit. It features three main carry outputs labeled C3, C2, and C1. The inputs G0, P0, G1, P1, G2, P2, and G3 are shown at the top. The circuit uses a combination of AND gates (represented by D-shaped symbols) and OR gates (represented by symbols with a curved input) to compute the carry signals. C1 is the output of an OR gate with inputs G0 and P0 AND C0. C2 is the output of an OR gate with inputs G1 and P1 AND C1. C3 is the output of an OR gate with inputs G2 and P2 AND C2.

Para  $A_n, B_n$  e  $C_0$ , latência de  $G_n$  e  $P_n = 1$

Latência do CLA6 ( $1 \leq n \leq 5$ ):  $\text{lat } G_0 P_n + 2 = 3$

$\text{lat}(C_6) = \underbrace{\text{lat CLA6}}_{G_0 \text{ e } P_0} + \text{uma porta p}^{\text{somar}} = 3 + 1 = 4$

Como  $C_6 = C_0$ , os  $C_n$  com  $7 \leq n \leq 11$  tem latência:  $\text{lat}(C_6) + 2 = 6$

Para  $6^x$ , precisaríamos a cada 6 CLA6's, um adicional, ou,  $6^{x-1} + 6^{x-2} + 6^{x-3} \dots 6^0$

Com  $x = 4$ ,  $6^3 + 6^2 + 6^1 + 6^0 = 216 + 36 + 6 + 1 = 259$

Nesta disciplina estamos realizando projetos com a temática do jogo **Jokempô**. No último enunciado, do **PR2**, foi mencionada a formação de uma **startup**, a **JKP**, que tinha como **business plan** a comercialização do jogo. Apareceu um problema que a equipe técnica da **JKP** teria que resolver. Eles teriam que calcular, dada uma magnitude, sem sinal (**unsigned**), representada por 4 bits, ( $d_3, d_2, d_1, d_0$ ), qual seria o resultado da multiplicação desta magnitude por 3 e depois somada de uma unidade, ou em outras palavras, obter a magnitude equivalente a três vezes a anterior, somada de uma unidade. Ocorre que, para realizar um primeiro protótipo experimental a equipe só dispunha de módulos de um somador binário, de duas palavras de 4 bits, com **carry** de entrada e de saída, conforme ilustrado na Figura1.

**2X** {  $d_3 d_2 d_1 d_0$  } {  $d_3 d_2 d_1 d_0$  }

$A_3 \ 2 \ 1 \ 0 \quad B_3 \ 2 \ 1 \ 0$

$\Sigma_3 \ \Sigma_2 \ \Sigma_1 \ \Sigma_0$

$C_4$  Somador Binário  $C_0$

$X = \text{signal a ser entrada}$

$3X + 1 =$

$X + 2X + 1$

$\underbrace{\hspace{1cm}} \underbrace{\hspace{1cm}} \underbrace{\hspace{1cm}} \text{ carry} =$

3 X deslocado p/ direita

$1010_2 \Rightarrow 10100_2$

$10_{10} \Rightarrow 20_{10}$

[deslocar a virgula ser  $\equiv$  mult. por 2 e o análogo decimal de multiplicar por 10]

$10^0$	$d_3$	$d_2$	$d_1$	$d_0$	$3X + 1$	$C_4$
0	0	0	0	0	0 0 0 1	0
1	0	0	0	1	0 1 0 0	0
2	0	0	1	0	0 1 1 1	0
3	0	0	1	1	1 0 1 0	0
4	0	1	0	0	1 1 0 1	0
5	0	1	0	1	0 0 0 0	1
6	0	1	1	0	.	1
7	0	1	1	1	.	.
8	1	0	0	0	.	.
9	1	0	0	1	.	.
10	1	0	1	0	.	.
11	1	0	1	1	.	.
12	1	1	0	0	.	.
13	1	1	0	1	.	.
14	1	1	1	0	.	.
15	1	1	1	1	.	.

A equipe técnica da **JKP** ficou entusiasmada com a solução, a ponto de esquecer de um possível problema. Esqueceram-se de levar em conta o que ocorreria quando a magnitude calculada ultrapassava o espaço de representação de 5 bits, isto é, ( $d_3, d_2, d_1, d_0$ ), e mais (**C<sub>4</sub>**). Eles pensaram em uma solução e após discutirem chegaram à conclusão que precisariam cascatear mais um módulo somador de 4 bits (Figura2).

Para cascatear, precisamos apenas adicionar os sinais da primeira somadora ( $C_{4out}$  e  $d_3$ ) a nova somadora, lembrando que agora ambas (!) são no menor dígito significativo, com as possibilidades:

"0000" +  $d_3$

$C_8$   $A \text{ ou } B$   $C_4$   $C_{4out}$

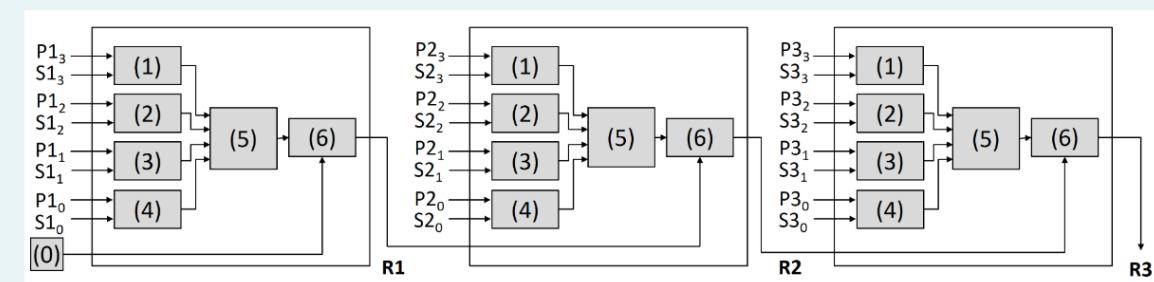
"0000" +  $C_{4out}$

$C_8$   $A \text{ ou } B$   $C_4$   $d_3$



Certo jogo de azar consiste no sorteio de três números de 4 bits cada ( $S_1$ ,  $S_2$  e  $S_3$ ), que são então comparados com três palpites do jogador ( $P_1$ ,  $P_2$  e  $P_3$ ). Deseja-se construir um hardware capaz de auxiliar na verificação do resultado desse jogo, informando quando o jogador errou todos os palpites. Um requisito nesse caso é utilizar um circuito iterativo, de modo que seja possível alterar o número de sorteios conforme necessidade, usando o mesmo circuito base.

O circuito abaixo mostra uma arquitetura de hardware iterativo capaz de atender a esse requisito, já montado com três blocos idênticos  $B_i$  (onde  $i = 1, 2, 3$ ), cada um deles considerando o palpite  $P_i$  (uma palavra de 4 bits  $P_{i3}P_{i2}P_{i1}P_{i0}$ ), o sorteio  $S_i$  (outra palavra de 4 bits  $S_{i3}S_{i2}S_{i1}S_{i0}$ ), e a saída do bloco anterior.



Considerando que o circuito deve operar no modo ativo-alto (i.e., use "1" na saída do circuito completo para indicar que o resultado desejado foi obtido), pede-se:

Qual deve ser o valor da entrada marcada como (0)? Resposta:  ✓

Qual porta lógica (se alguma) deve ser colocada internamente a cada bloco, nas posições marcadas de (1) a (6)?

- (1):  ✓
- (2):  ✓
- (3):  ✓
- (4):  ✓
- (5):  ✓
- (6):  ✓

