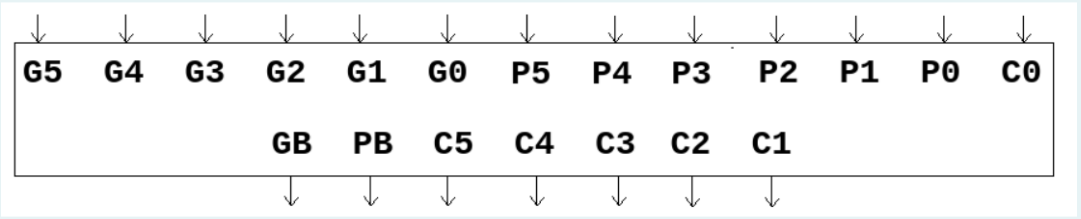
Suponha que queremos construir um somador binário de n bits, com entradas $A = A_{n-1} \dots A_0$ e $B = B_{n-1} \dots B_0$, além do carry-in C_0, e saídas $S = S_{n-1} \dots S_0$, além do carry-out C_n. Para tal, poderíamos simplesmente cascatear n somadores completos de 1 bit. Entretanto, esta abordagem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

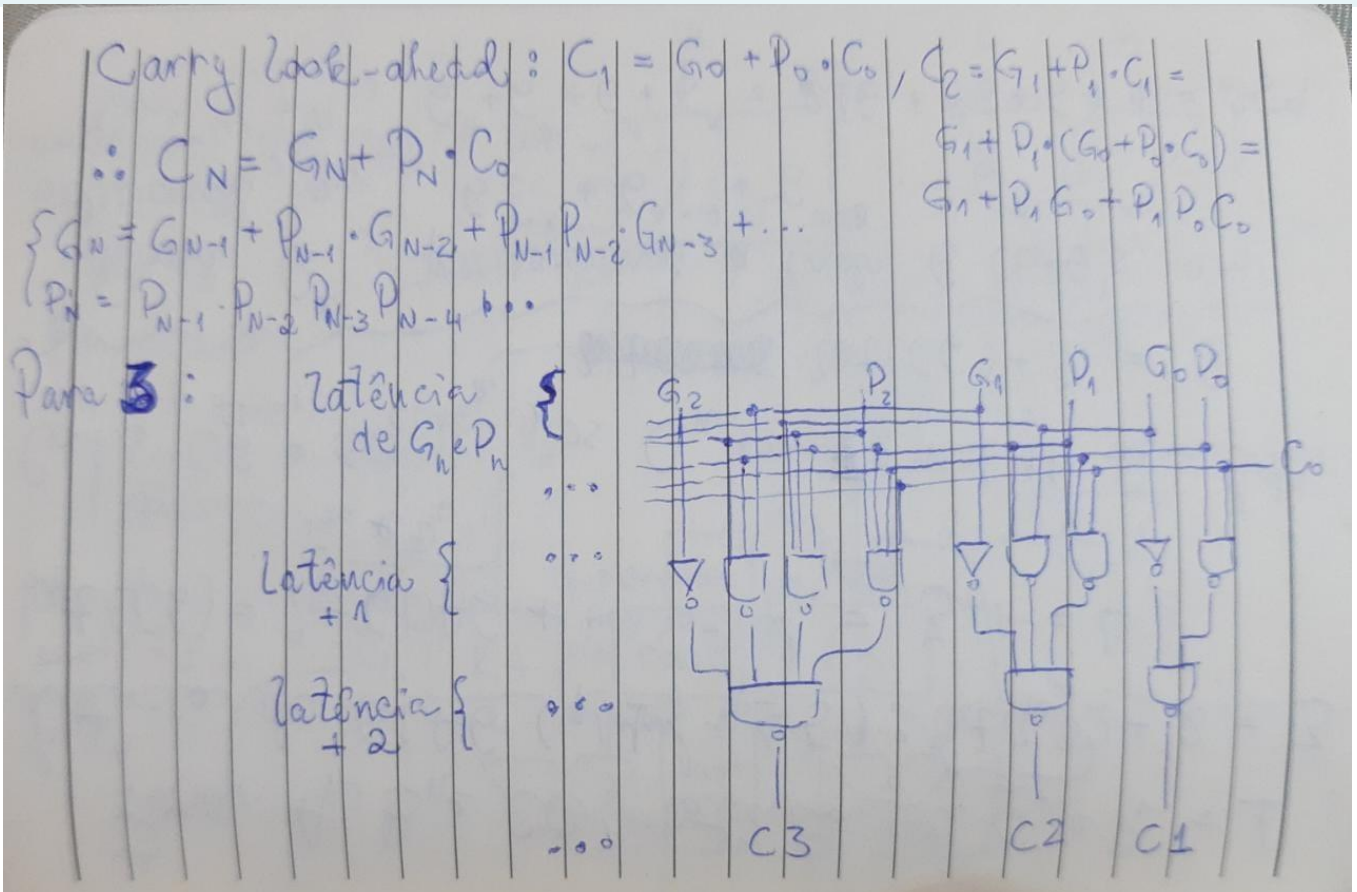
O cálculo do vai-um antecipado (carry look-ahead) permite reduzir este atraso. Esta técnica utiliza os bits P_i e G_i, que denotam quando a soma no i-ésimo bit propaga ou gera um carry, e que podem ser definidos, para $1 \leq i \leq n$, como:

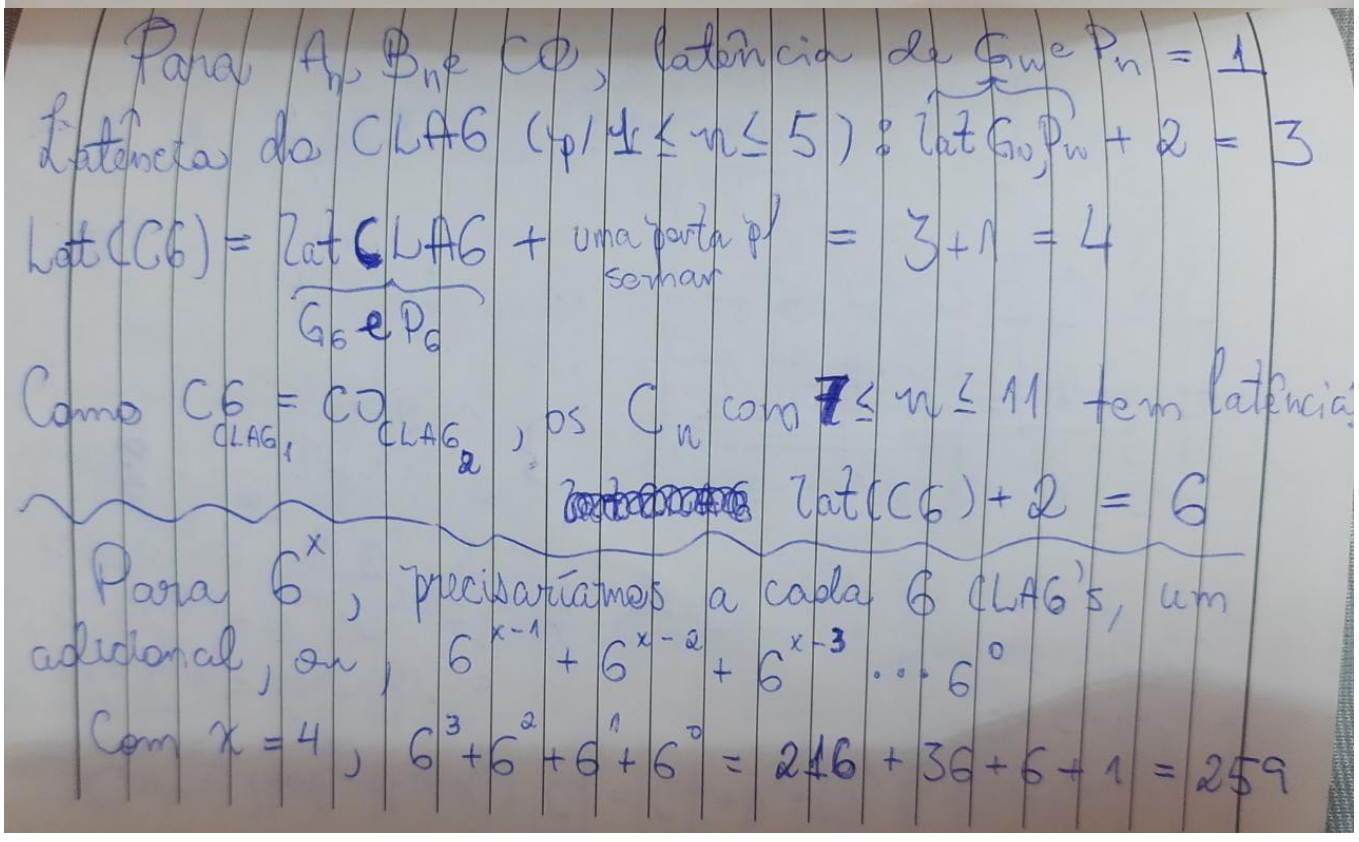
$P_i = A_i + B_i$ e $G_i = A_i \cdot B_i$

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:

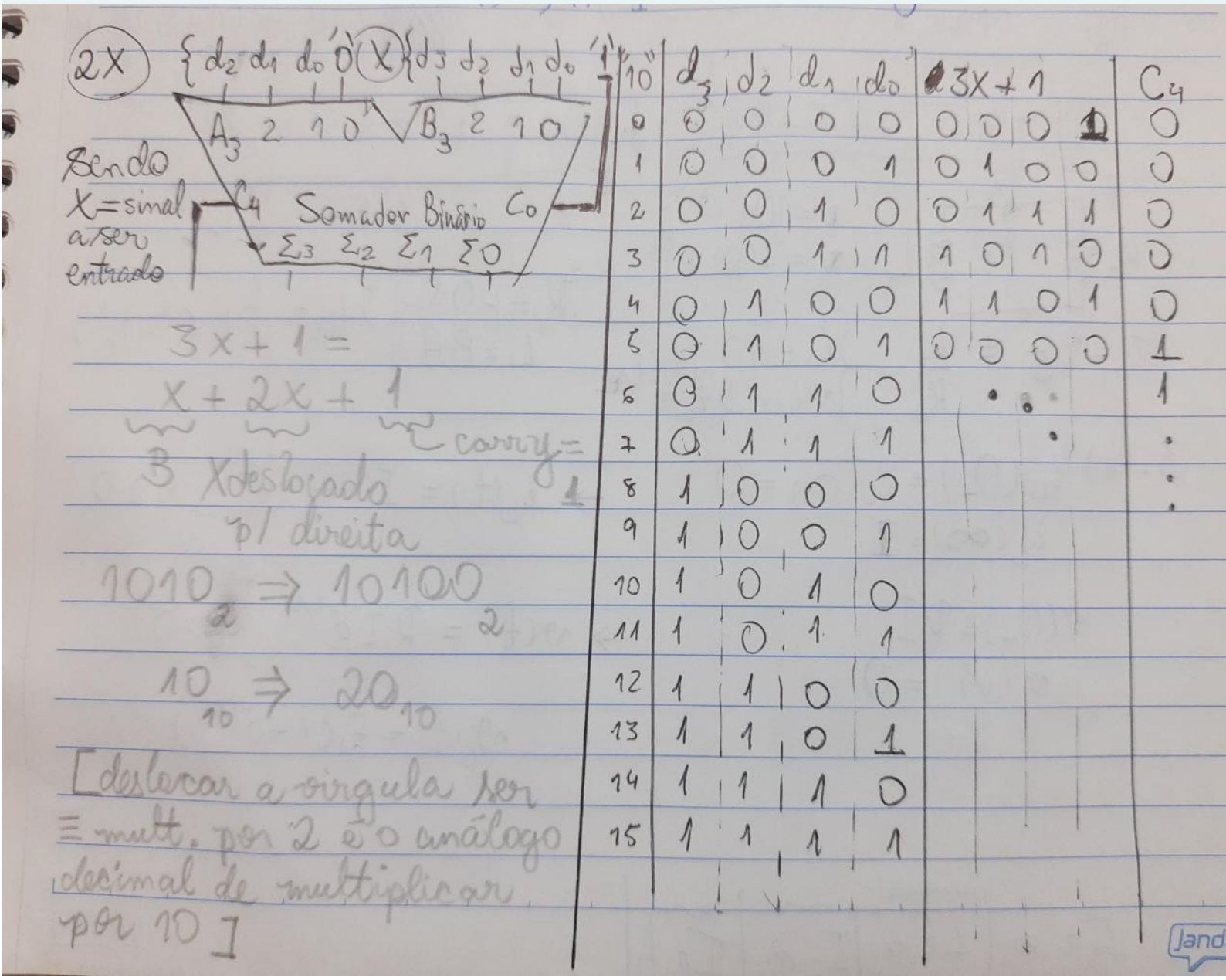


Note que G_B e P_B podem ser calculados em função dos G_i's e dos P_i's e podem ser usados para calcular C_6. Determine o atraso das saídas de carry de um somador de 6 bits, que usa CLA6, em relação às entradas A, B e C_0 (fornecidas em $t = 0$).



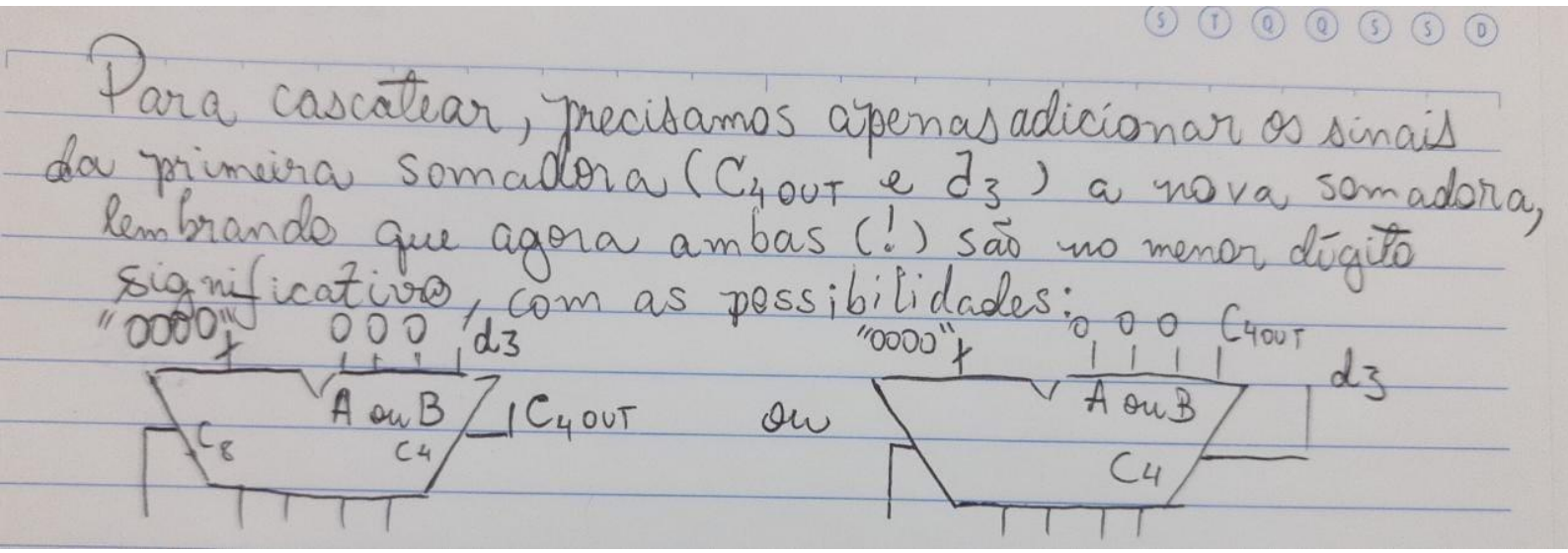


Nesta disciplina estamos realizando projetos com a temática do jogo **Jokempô**. No último enunciado, do **PR2**, foi mencionada a formação de uma *startup*, a **JKP**, que tinha como *business plan* a comercialização do jogo. Apareceu um problema que a equipe técnica da **JKP** teria que resolver. Eles 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.



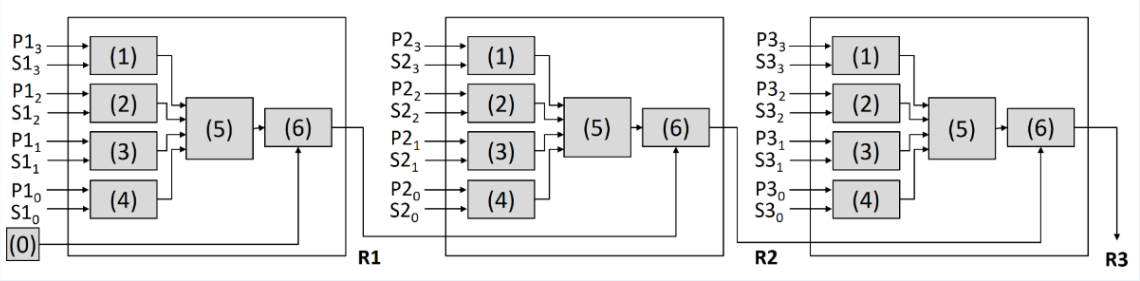
10	d_3	d_2	d_1	d_0	3X+1	C_4
0	0	0	0	0	0001	0
1	0	0	0	1	0100	0
2	0	0	1	0	0111	0
3	0	0	1	1	1010	0
4	0	1	0	0	1101	0
5	0	1	0	1	0000	1
6	0	1	1	0	...	1
7	0	1	1	1		...
8	1	0	0	0		
9	1	0	0	1		
10	1	0	1	0		
11	1	0	1	1		
12	1	1	0	0		
13	1	1	0	1		
14	1	1	1	0		
15	1	1	1	1		

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_4**). Eles pensaram em uma solução e após discutirem chegaram à conclusão que precisariam cascatear mais um módulo somador de 4 bits (Figura2).



Certo jogo de azar consiste no sorteio de três números de 4 bits cada (S1, S2 e S3), que são então comparados com três palpites do jogador (P1, P2 e P3). 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 Bi (onde i = 1, 2, 3), cada um deles considerando o palpites Pi (uma palavra de 4 bits $Pi_3Pi_2Pi_1Pi_0$), o sorteio Si (outra palavra de 4 bits $Si_3Si_2Si_1Si_0$), 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: 1 ✔

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

(1): XOR ✔ (2): XOR ✔

(3): XOR ✔ (4): XOR ✔

(5): OR (6): AND ✔