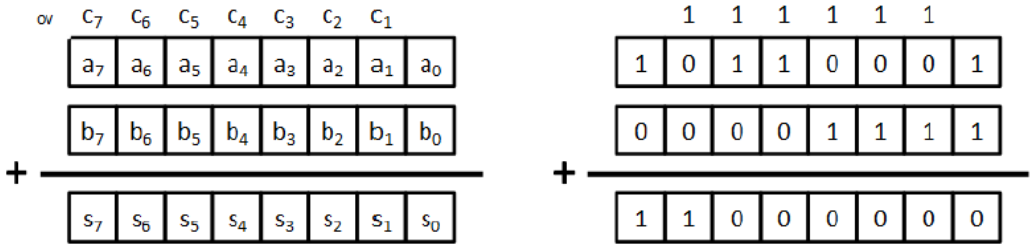
Oitavo Roteiro de Laboratório

Circuitos Aritméticos – Somador

Matricula	Nome

Neste roteiro de laboratório iremos criar uma descrição de hardware em VHDL que implementa os circuitos meio somador e somador completo. A seguir pode-se observar um exemplo do processo de soma binária.



Para criarmos um circuito capaz de somar o primeiro bit (bit menos significativo, LSB, ou bit mais a direita) precisamos considerar apenas os bits na posição zero do primeiro e segundo operadores. Vale lembrar que na soma, não há a possibilidade do considerar um "vem um", ou carry in como o chamaremos a partir deste momento.

Passo 1: A tabela abaixo lista todas as possíveis combinações para a soma do bit 0 dos operandos 1 e 2. As saídas s_0 e c_{OUT} representam respectivamente o resultado da soma e "vai um" para o próximo bit. Levante a expressão em soma de produtos para as saídas s_0 e c_{OUT}.

a_0	b_0	s_0	c_{OUT}	Resultado da soma	"Vai um" (carry out)
0	0	0	0		
0	1	1	0	→	
1	0	1	0	→	
1	1	0	1		→

Passo 2: O resultado de s_0 foi visto em sala de aula diversas vezes. Que porta lógica é equivalente a esta expressão?

s_0 é equivalente a soma de produtos _____ que é equivalente à _____
A saída c_{OUT} representa a porta lógica fundamental _____ e sua soma de produtos é _____.

Passo 3: Com base na expressões algébricas levantadas acima, complete o quadro abaixo com as portas lógicas correspondentes:

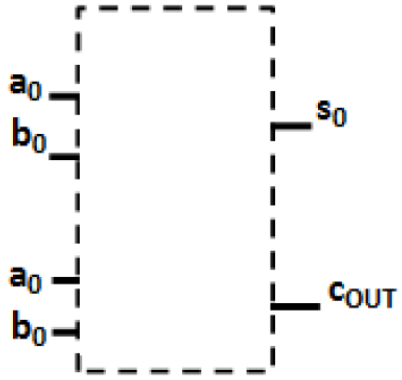
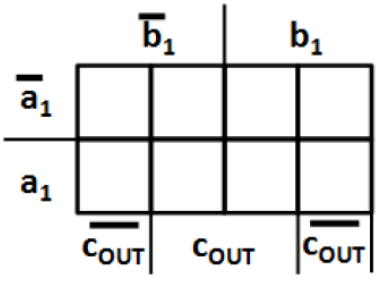
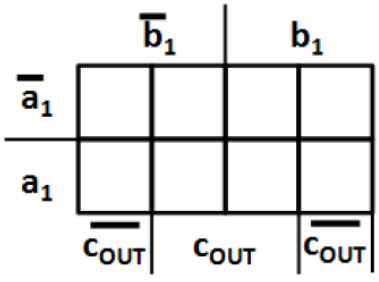


Figura 1 – Meio Somador

Passo 4: Consideremos agora o caso do segundo bit (bit 1), aquele que vem logo após o bit 0.Para este caso, devemos considerar além dos bits de entrada dos operandos 1 e 2 o bit de "vai um" (c_{OUT}) produzido pelo meio somador. Este bit de "vai um" será nomeado c_{IN} no circuito que estamos projetando agora, porque ele é o "vai um" que entra neste circuito. Abaixo é apresentada a tabela verdade do circuito somador completo, que recebe este nome pois ele é capaz de somar dois números e adicionalmente considerar na soma o "vai um" de seu bit predecessor. Levante a expressão em soma de produtos para as saídas s_1 e c_{OUT}.

a_1	b_1	c_{IN}	S_1	c_{OUT}	Resultado da soma	"Vai um" (carry out)
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	→	→
Soma de Produtos						

Passo 5:Simplifique as expressões em soma de produtos das saídas s_1 e c_{OUT} utilizando mapas de Veitch-Karnaugh.

Resultado da soma	"Vai um" (carry out)
	
s_1 =	c_{OUT} =

Note que a expressão de c_{OUT} pode ser simplificada. No entanto s_1 apresenta uma configuração no mapa de Veitch-Karnaugh que não admite simplificação uma vez que todos os "1"s encontram-se isolados. De fato, esta é uma exceção conhecida e sabemos que quando encontramos este tipo de padrão de distribuição de "1"s a expressão pode ser simplificada para $\oplus$'s entre todos os elementos do mapa. Abaixo apresentamos a prova via manipulação algébrica.

$$
\begin{aligned}
s_n = \quad & (\bar{a}_n \cdot \bar{b}_n \cdot c_n) + (\bar{a}_n \cdot b_n \cdot \bar{c}_n) + \\
& (a_n \cdot \bar{b}_n \cdot \bar{c}_n) + (a_n \cdot b_n \cdot c_n) \\
s_n = \quad & \bar{a}_n \cdot [(\bar{b}_n \cdot c_n) + (b_n \cdot \bar{c}_n)] + \\
& a_n \cdot (\bar{b}_n \cdot \bar{c}_n) + (b_n \cdot c_n) \\
s_n = \quad & \bar{a}_n \cdot (b_n \oplus c_n) + a_n \cdot (b_n \otimes c_n) \\
s_n = \quad & \bar{a}_n \cdot (b_n \oplus c_n) + a_n \cdot \overline{(b_n \oplus c_n)} \\
s_n = \quad & a_n \oplus b_n \oplus c_n
\end{aligned}
$$

Passo 6: Com base nas duas expressões levantadas, construa utilizando portas lógicas o circuito do somador completo abaixo.

Figura 2 - Somador Completo

Todos os bits adicionais do somador possuirão o mesmo circuito que deve ser replicado para cada bit. Note que a entrada c_{IN} do bit deve ser conectada a saída c_{OUT} do bit predecessor.

Passo 7: A seguir, utilizando o software ModelSim implemente as seis descrições de hardware elencadas a seguir:

1. **meioSomador.vhd** - implementa o sistema digital meio somador;

2. **tb_meioSomador.vhd** - implementa o testbench que deve testar sistematicamente (todas as quatro possibilidades) o sistema **meioSomador**;
3. **somadorCompleto.vhd** - implementa o sistema digital somador completo;
4. **tb_somadorCompleto.vhd** - implementa o testbench que deve testar sistematicamente (todas as 16 possibilidades) o sistema **somadorCompleto**;
5. **somador8bits.vhd** - implementa um sistema digital que executa a soma de números naturais utilizando os sistemas meioSomador e somadorCompleto como blocos fundamentais (1 MS e 7 SCs);
6. **tb_somador8bits.vhd** - testa o sistema digital descrito anteriormente. (alguns casos incluindo casos que geram overflow).

Abaixo é apresentado um circuito exemplo para o somador8bits.

```
1  ----------------------------------------------------
2  --somador.vhd
3  --
4  --Desc: circuito que implementa um somador de 8
5  -- bits utilizando apenas portas lógicas
6  --
7  -- DDA 24/07/2013
8  ----------------------------------------------------
9  library ieee;
10 use ieee.std_logic_1164.all;
11
12 entity somador is
13   port(
14     a0,a1,a2,a3,a4,a5,a6,a7: in  std_logic;
15     b0,b1,b2,b3,b4,b5,b6,b7: in  std_logic;
16     s0,s1,s2,s3,s4,s5,s6,s7: out std_logic
17     );
18 end somador;
19
20 architecture a_somador of somador is
21   signal c0, c1, c2, c3, c4, c5, c6, c7 : std_logic;
22 begin
23   --meio somador, bit 0
24   s0 <= a0 xor b0;
25   c0 <= a0 and b0;
26   -- somador completo, bit 1
27   s1 <= a1 xor b1 xor c0;
28   c1 <= (a1 and c0) or (a1 and b1) or (b1 and c0);
29
30   --coloque o restante do código de seu somador aqui
31
32 end architecture somador;
```

Passo 8: Execute os três testbenchs implementados.

Boa Diversão Pessoal!