



**Universidade Federal de Santa Catarina
Centro Tecnológico – CTC
Departamento de Engenharia Elétrica**



<http://gse.ufsc.br>

“EEL7020 – Sistemas Digitais”

Prof. Eduardo Augusto Bezerra

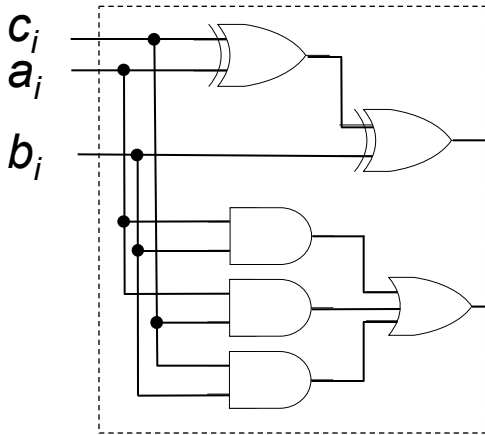
Eduardo.Bezerra@eel.ufsc.br

Florianópolis, agosto de 2012.

Sistemas Digitais

Projeto de somador de 4 bits

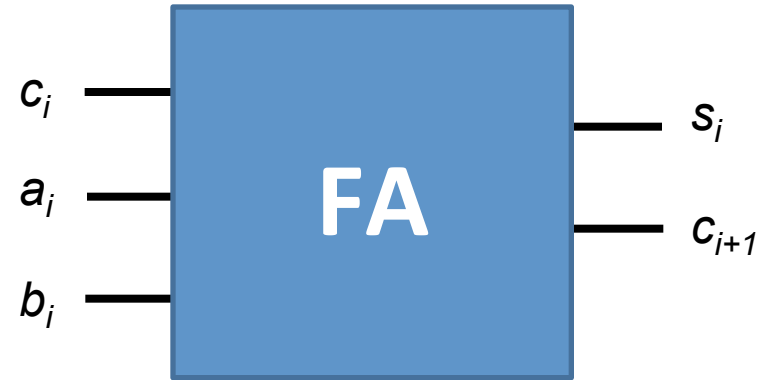
Exemplo: somador de 4 bits



$$s_i = c_i \text{ xor } a_i \text{ xor } b_i$$

$$c_{i+1} = a_i \text{ and } b_i \text{ or } a_i \text{ and } c_i \text{ or } b_i \text{ and } c_i$$

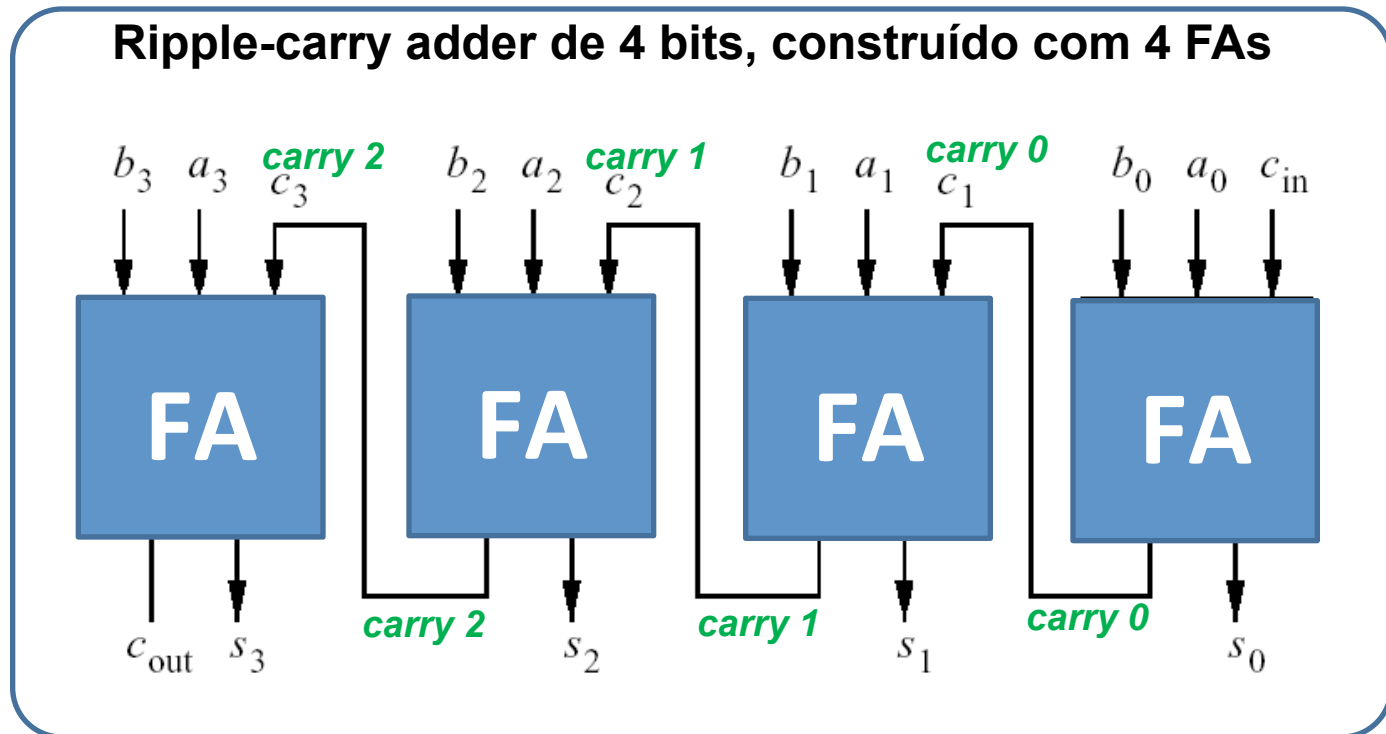
FA (1 bit)



FA (1 bit)

a_i	b_i	c_i	c_{i+1}	s_i
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

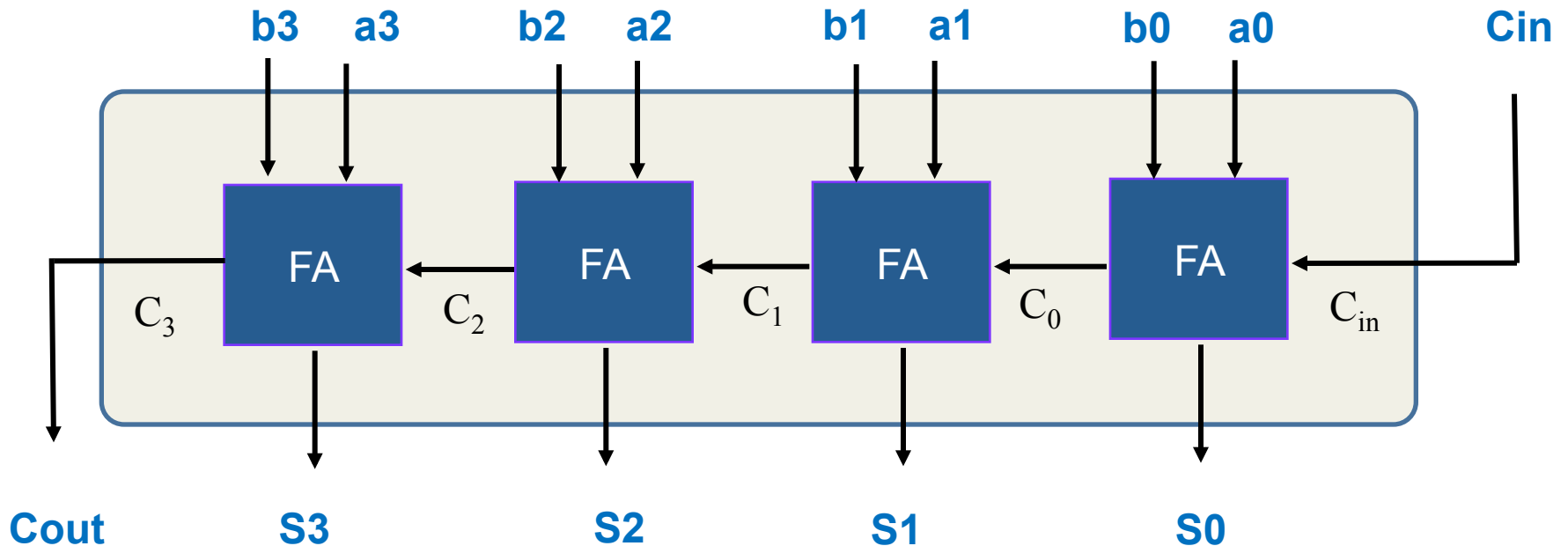
Ripple-carry adder de 4 bits, construído com 4 FAs



Exemplo: somador de 4 bits

$$\begin{array}{r} C_{out} \\ C_3 \quad C_2 \quad C_1 \quad C_0 \quad C_{in} \\ \quad A_3 \quad A_2 \quad A_1 \quad A_0 \\ + \quad B_3 \quad B_2 \quad B_1 \quad B_0 \\ \hline S_3 \quad S_2 \quad S_1 \quad S_0 \end{array}$$

Ripple Carry Addder (RCA)



Exemplo: somador de 4 bits

A seguir são apresentadas três implementações em VHDL para um somador de 4 bits para utilização no kit DE2 da Altera.

Algumas observações:

- A “Solução I” é bastante semelhante às implementações em VHDL estrutural desenvolvidas nas aulas anteriores (início do curso).
- Na “Solução I”, notar a utilização do *signal carry* na *architecture*.
- Na “Solução II” foi criado um **componente** que implementa um somador completo (*Full-Adder* ou FA) na entity FA, e esse componente é utilizado na *architecture* RCA_stru da *entity* RCA.
- Na “Solução II” foi utilizado um XOR para verificação de overflow. Isso é necessário, uma vez que está sendo assumido o uso do circuito para cálculos com números com sinal (**complemento de 2**).
- Na “Solução III”, foi utilizado o operador **+** para geração do somador de 4 bits.
- Lembrando que para utilizar o operador **+** é necessário incluir o **use IEEE.std_logic_unsigned.all;**

Solução I

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY Somador4 IS
  PORT (
    SW : IN STD_LOGIC_VECTOR(17 DOWNT0 0);
    LEDG : OUT STD_LOGIC_VECTOR(7 DOWNT0 0)
  );
```

```
END Somador4;
```

```
ARCHITECTURE stru OF Somador4 IS
```

```
  signal carry, a, b: std_logic_vector(3 downto 0);
```

```
  BEGIN
```

```
    a <= SW(4 downto 1); b <= SW(8 downto 5);
```

```
    LEDG(0) <= ((a(0) xor b(0)) xor SW(0));
```

```
    carry(0) <= (a(0) and b(0)) or (a(0) and SW(0)) or (b(0) and SW(0));
```

```
    LEDG(1) <= ((SW(2) xor SW(6)) xor carry(0));
```

```
    carry(1) <= (a(1) and b(1)) or (a(1) and carry(0)) or (b(1) and carry(0));
```

```
    LEDG(2) <= ((SW(3) xor SW(7)) xor carry(1));
```

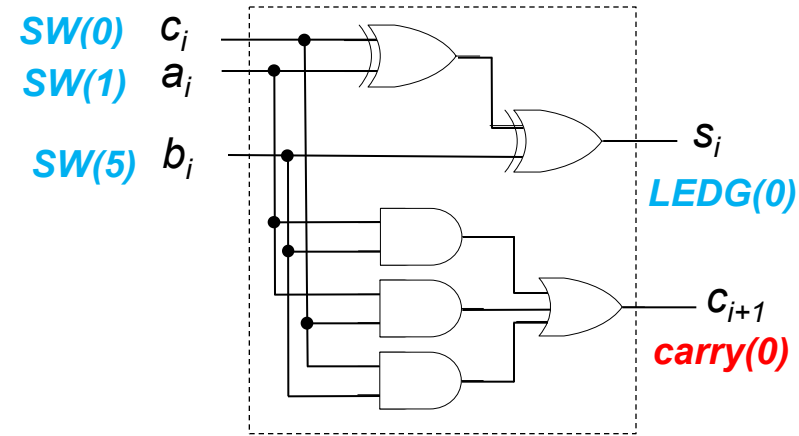
```
    carry(2) <= (a(2) and b(2)) or (a(2) and carry(1)) or (b(2) and carry(1));
```

```
    LEDG(3) <= ((SW(4) xor SW(8)) xor carry(2));
```

```
    carry(3) <= (a(3) and b(3)) or (a(3) and carry(2)) or (b(3) and carry(2));
```

```
    LEDG(4) <= carry(3);
```

```
  END stru;
```



```
-- SW(0) = Cin
```

```
-- SW(1) = A0
```

```
-- SW(2) = A1
```

```
-- SW(3) = A2
```

```
-- SW(4) = A3
```

```
-- SW(5) = B0
```

```
-- SW(6) = B1
```

```
-- SW(7) = B2
```

```
-- SW(8) = B3
```

```
-- LEDG(0) = S0
```

```
-- LEDG(1) = S1
```

```
-- LEDG(2) = S2
```

```
-- LEDG(3) = S3
```

```
-- LEDG(4) = Cout
```

```
library ieee;
use ieee.std_logic_1164.all;
-- somador completo (FA)
entity FA is
  port (a, b, c: in std_logic;
        soma, carry: out std_logic);
end FA;
architecture FA_beh of FA is
begin
  soma <= (a xor b) xor c;
  carry <= b when ((a xor b) = '0')
            else c;
end FA_beh;
```

```
-- SW(0) = Cin
-- SW(1) = A0
-- SW(2) = A1
-- SW(3) = A2
-- SW(4) = A3
-- SW(5) = B0
-- SW(6) = B1
-- SW(7) = B2
-- SW(8) = B3
-- LEDG(0) = S0
-- LEDG(1) = S1
-- LEDG(2) = S2
-- LEDG(3) = S3
-- LEDG(6) = Cout
-- LEDG(7) = Overf
```

```
library ieee;
use ieee.std_logic_1164.all;
entity RCA is
  port (SW : IN STD_LOGIC_VECTOR(17 downto 0);
        LEDG : OUT STD_LOGIC_VECTOR(7 downto 0)
        );
end RCA;

architecture RCA_stru of RCA is
  signal carry: std_logic_vector (3 downto 0);
  component FA
    port (a, b, c: in std_logic;
          soma, carry: out std_logic);
  end component;
begin
  FA0:
    FA port map (sw(1), sw(5), sw(0), LEDG(0), carry(0));
  FA1:
    FA port map (sw(2), sw(6), carry(0), LEDG(1), carry(1));
  FA2:
    FA port map (sw(3), sw(7), carry(1), LEDG(2), carry(2));
  FA3:
    FA port map (sw(4), sw(8), carry(2), LEDG(3), carry(3));
  LEDG(6) <= carry(3); -- carry out
  LEDG(7) <= carry(2) xor carry(3); -- overflow
  -- para tratar números com sinal (complemento de 2)
end RCA_stru;
```

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

```

```

entity RCA is
  port (SW : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
        LEDG : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
end RCA;

```

```

architecture RCA_stru of RCA is
begin

```

```

  LEDG(3 downto 0) <= SW(4 downto 1) + SW(8 downto 5) + ("000" & SW(0));
end RCA_stru;

```

```
-- SW(0) = Cin
```

```
-- SW(1) = A0
```

```
-- SW(2) = A1
```

```
-- SW(3) = A2
```

```
-- SW(4) = A3
```

```
-- SW(5) = B0
```

```
-- SW(6) = B1
```

```
-- SW(7) = B2
```

```
-- SW(8) = B3
```

```
-- LEDG(0) = S0
```

```
-- LEDG(1) = S1
```

```
-- LEDG(2) = S2
```

```
-- LEDG(3) = S3
```


Resultado da síntese (*compile* no Quartus II)

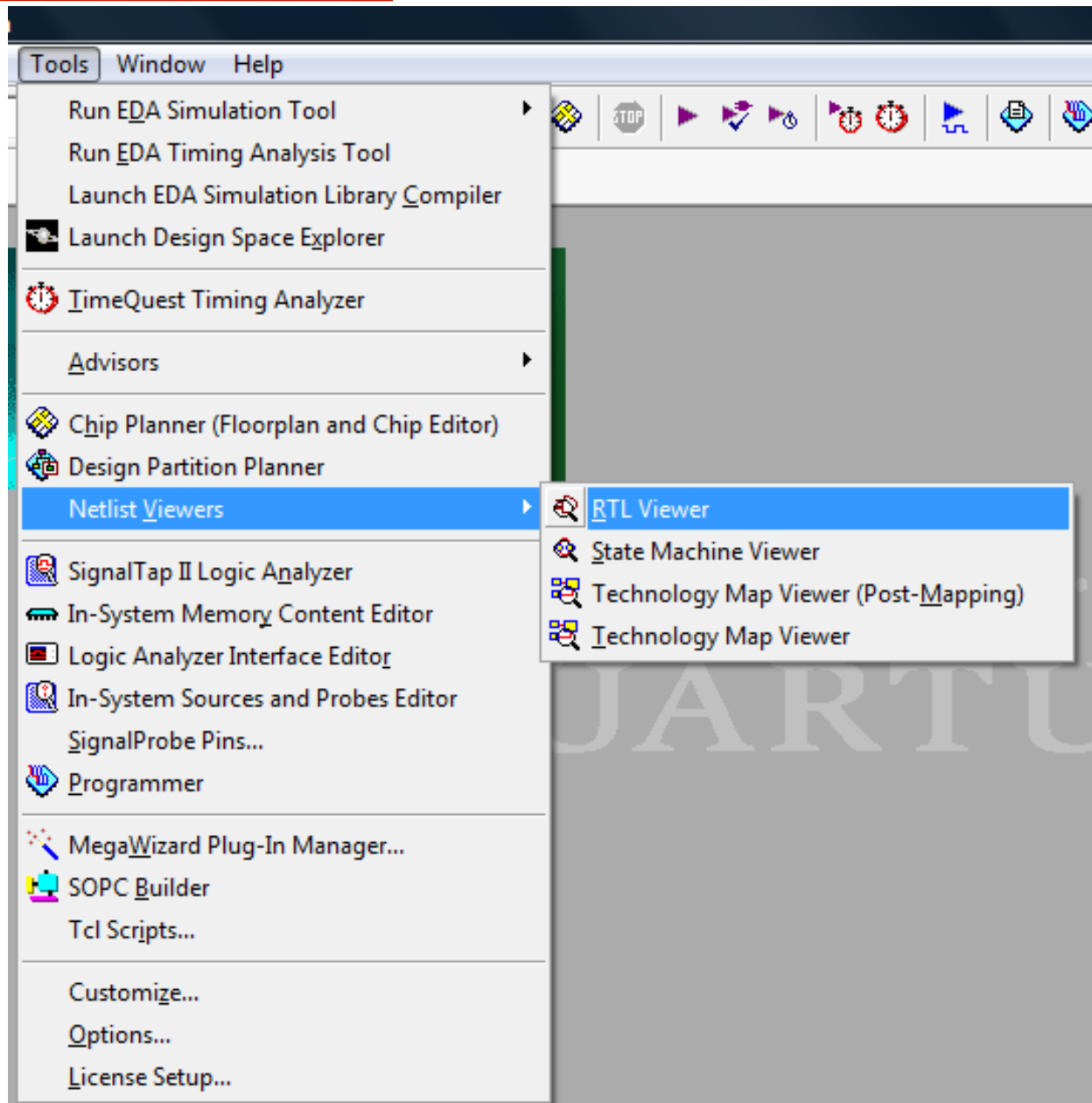
Solução II

Flow Status	Successful - Sat Oct 1
Quartus II Version	9.1 Build 350 03/24/2
Revision Name	rca
Top-level Entity Name	RCA
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	10 / 33,216 (< 1 %)
Total combinational functions	10 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	44 / 475 (9 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Solução III

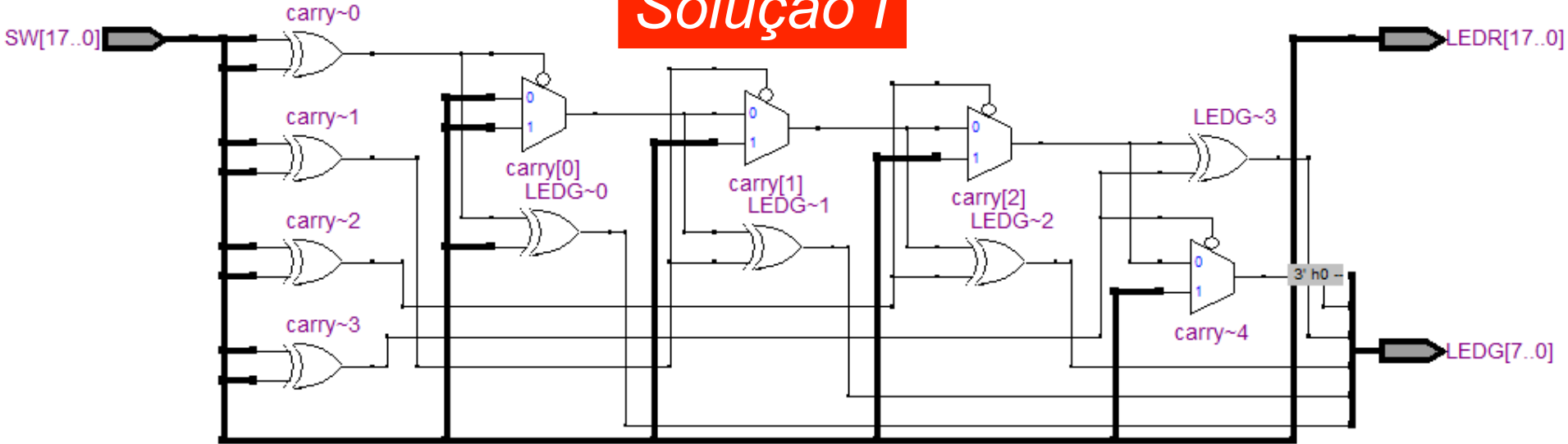
Flow Status	Successful - Sat Oct
Quartus II Version	9.1 Build 350 03/24/
Revision Name	rca
Top-level Entity Name	RCA
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	5 / 33,216 (< 1 %)
Total combinational functions	5 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	26 / 475 (5 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Utilizando o *RTL Viewer* do Quartus II



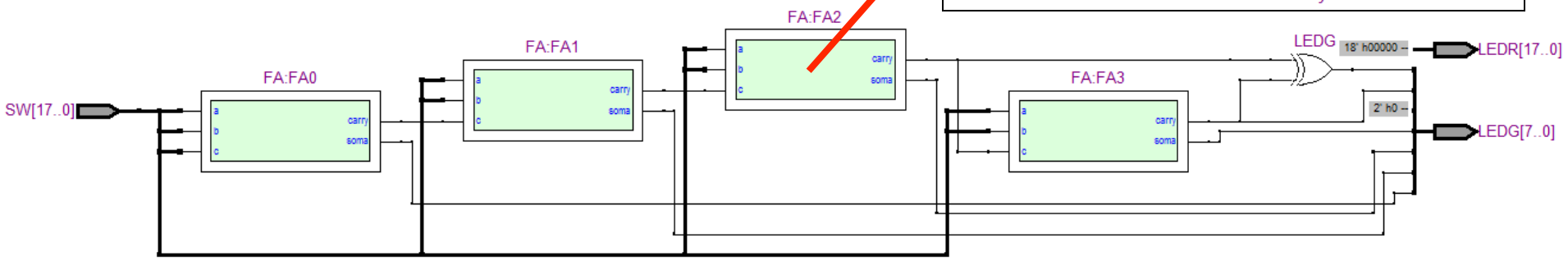
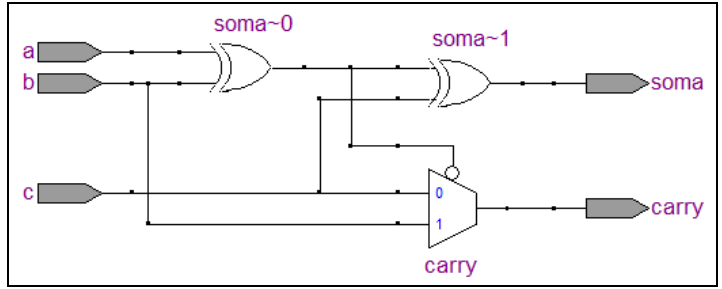
Utilizando o *RTL Viewer* do Quartus II

Solução I



Solução II

Entrando no projeto hierárquico



Flags

- Na Solução II foram realizadas as seguintes atribuições:

$\text{LEDG}(6) \leq \text{carry}(3);$

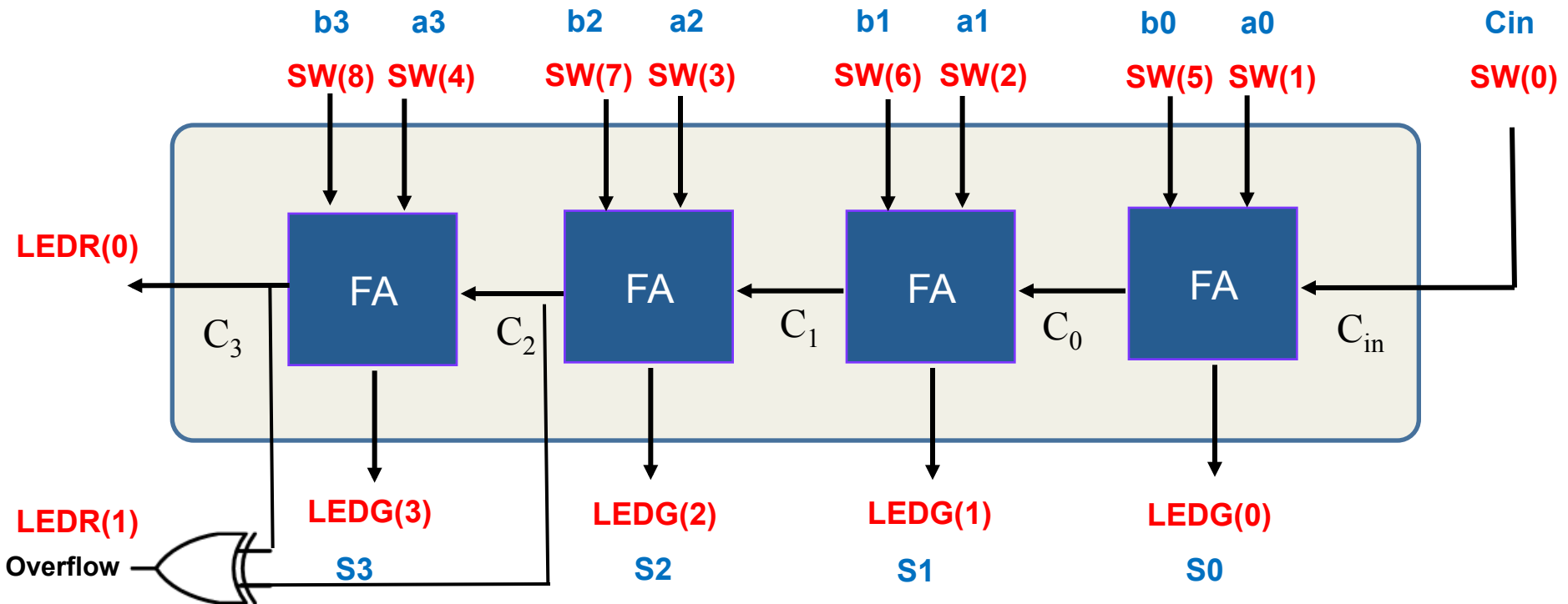
$\text{LEDG}(7) \leq \text{carry}(2) \text{ xor } \text{carry}(3);$

- A princípio, essas **sinalizações** (*flags*) não são necessárias, uma vez que o resultado da soma já está sendo indicado em $\text{LEDG}(3)$, $\text{LEDG}(2)$, $\text{LEDG}(1)$, $\text{LEDG}(0)$.
- Porém, flags são bastante úteis para sinalizar situações, tais como (números com sinal):
 - Resultado está incorreto devido a *overflow* (**$\text{LEDG}(7)$**)
 - Somador gerou um carry out (**$\text{LEDG}(6)$**)
 - Resultado da soma foi um valor negativo
 - Resultado da soma foi ZERO

***Tarefa a ser realizada:
Projeto de somador de 8 bits com flags***

Projeto de somador paralelo RCA

- O diagrama a seguir apresenta um RCA de 4 bits
- Utiliza um XOR entre o C_2 e C_3 para indicar ocorrência de *overflow*, assumindo que o circuito trabalho com números com sinal
- Sinaliza em LEDR(0) a ocorrência de carry out



Tarefa

A partir da **Solução II** apresentada anteriormente, implementar um **somador de 8 bits** com as seguintes entradas e saídas:

Entrada Cin e A

Entrada B

Saída Soma

Saída Flags

SW(0) = Cin

SW(1) = A0

SW(2) = A1

SW(3) = A2

SW(4) = A3

SW(5) = A4

SW(6) = A5

SW(7) = A6

SW(8) = A7

SW(9) = B0

SW(10) = B1

SW(11) = B2

SW(12) = B3

SW(13) = B4

SW(14) = B5

SW(15) = B6

SW(16) = B7

LEDG(0) = S0

LEDG(1) = S1

LEDG(2) = S2

LEDG(3) = S3

LEDG(4) = S4

LEDG(5) = S5

LEDG(6) = S6

LEDG(7) = S7

LEDR(0) = carry out

LEDR(1) = overflow

LEDR(2) = negativo

LEDR(3) = zero

C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1		←	“vai-um” (<i>carry</i>)
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	←	1° operando
+	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	←	2° operando
	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0	←	soma

Tarefa (continuação)

- Os LEDs verdes (LEDG) deverão apresentar o resultado da soma dos operandos A e B (fornecidos com SW).
- O LED vermelho 0 deverá acender, sempre que o resultado de uma soma gerar um *carry out*, caso contrário deverá permanecer apagado.
- O LED vermelho 1 deverá acender, sempre que o resultado de uma soma resultar em *overflow*, caso contrário deverá permanecer apagado.
- O LED vermelho 2 deverá acender, sempre que o resultado de uma soma resultar em um valor negativo (bit 7 = '1'), caso contrário deverá permanecer apagado.
- O LED vermelho 3 deverá acender, sempre que o resultado de uma soma resultar em zero, caso contrário deverá permanecer apagado.

LEDR(0) = carry out
LEDR(1) = overflow
LEDR(2) = negativo
LEDR(3) = zero

Tarefa (continuação)

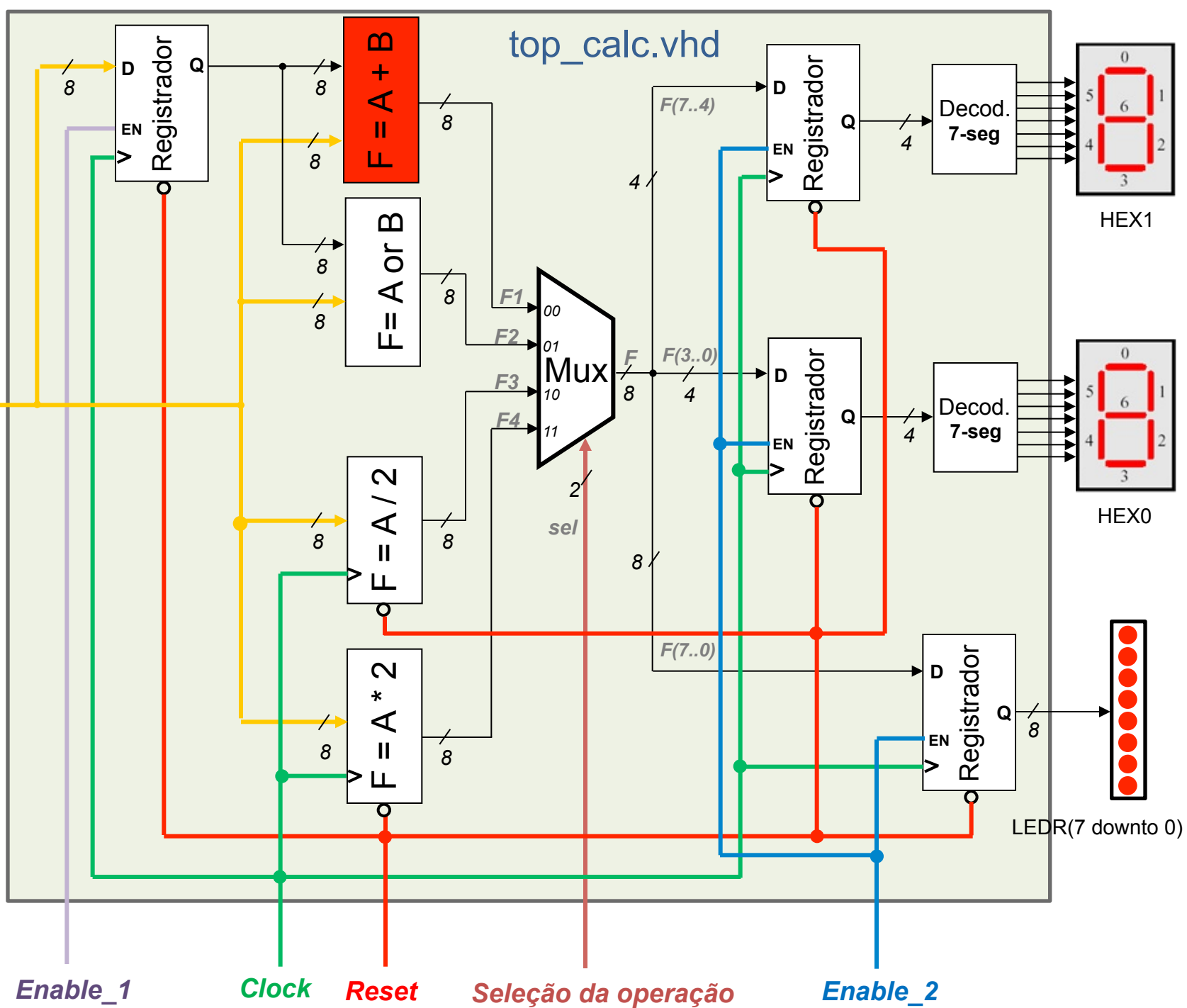
- Realizar a **simulação temporal** (e não a **funcional** das aulas anteriores), de forma a verificar os valores intermediários da soma, até o circuito combinacional do somador fornecer o resultado correto na saída.

Utilização do novo somador na mini-calculadora

Somador paralelo RCA na mini-calculadora

- Após o novo somador estar funcionando na simulação temporal, alterar o projeto da mini-calculadora desenvolvida durante o semestre, de forma a utilizar esse novo circuito.
- O componente C1 utilizado desde as primeiras aulas para realizar a soma (com o operador de '+'), deve ser removido, e o novo somador RCA deve ser incluído no seu lugar.
- Realizar todas as alterações necessárias na interface (entity) do novo somador, de forma a incluí-lo na mini-calculadora sem precisar alterar os demais componentes, conforme o diagrama de blocos a seguir.
- Como os **LEDs vermelhos** já estão em uso pelo circuito da mini-calculadora, nesse caso, os **LEDs verdes** poderiam ser utilizados para apresentar os flags do novo somador.

Operandos
SW(7 downto 0)



TOPO

Operandos
SW(7 downto 0)

