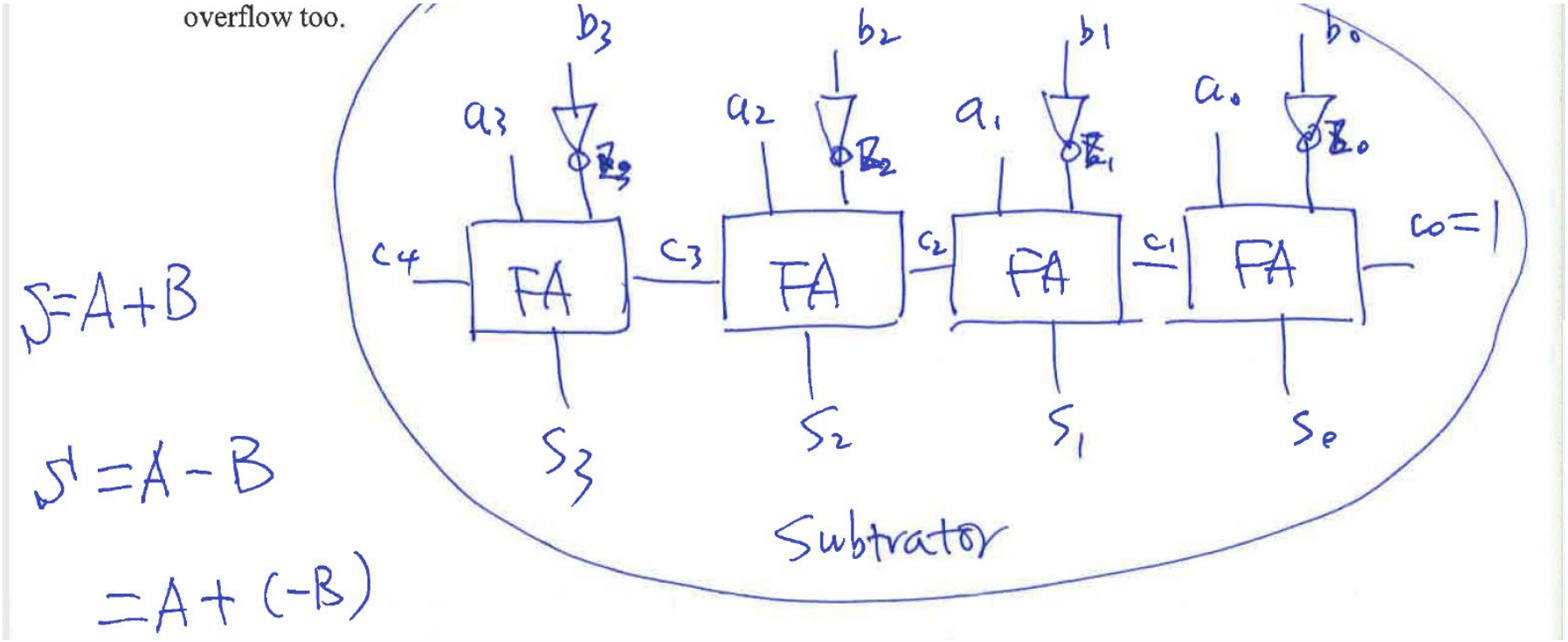
Lab 06: Design A

- Structural VHDL implementation of a 4-bit subtractor (ripple-carry structure)
  - Component: Full adder (one bits), which can be written in data-flow style
  - Instantiate the full adder four times and wire them (4 bits)
  - Make a 4-bit subtractor, using Full adders and NOT gates
  - Add an overflow flag (using XOR gate)
- You need to perform the following
  - Create a project, create VHDL codes, and compile
  - Create test vectors and perform functional simulation

4-bit subtractor

- Use four Full Adders and four NOT gates



Full adder in VHDL

- As in Lab 05

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY fulladd IS
        PORT ( Cin, x, y : IN STD_LOGIC ;
                        s, Cout : OUT STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
        s <= x XOR y XOR Cin ;
        Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;
END LogicFunc ;
```

Ripple-carry adder in VHDL

- Structural style
- Declaration of Component
- Instantiation statement
- Named or positional association

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY adder4 IS
        PORT ( Cin : IN STD LOGIC ;
                        x3, x2, x1, x0        : IN STD_LOGIC ;
                        y3, y2, y1, y0        : IN STD_LOGIC ;
                        s3, s2, s1, s0        : OUT STD_OGIC ;
                        Cout                  : OUT STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
        SIGNAL c1, c2, c3 : STD_LOGIC ;
        COMPONENT fulladd
                PORT ( Cin, x, y        : IN STD_LOGIC ;
                                s, Cout       : OUT STD_LOGIC ) ;
        END COMPONENT ;
BEGIN
        stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 ) ;
        stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
        stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
        stage3: fulladd PORT MAP (
                Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
END Structure ;
```

Ripple-carry adder in VHDL (cont'd)

- Data objective in vectors

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY adder4 IS
        PORT (      Cin     : IN     STD_LOGIC ;
                    X, Y    : IN     STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                    S       : OUT    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                    Cout    : OUT    STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
        SIGNAL C : STD_LOGIC_VECTOR(1 TO 3) ;
        COMPONENT fulladd
                PORT ( Cin, x, y            : IN STD_LOGIC ;
                                s, Cout     : OUT STD_LOGIC ) ;
        END COMPONENT
BEGIN
        stage0: fulladd PORT MAP ( Cin, X(0), Y(0), S(0), C(1) ) ;
        stage1: fulladd PORT MAP ( C(1), X(1), Y(1), S(1), C(2) ) ;
        stage2: fulladd PORT MAP ( C(2), X(2), Y(2), S(2), C(3) ) ;
        stage3: fulladd PORT MAP ( C(3), X(3), Y(3), S(3), Cout ) ;
END Structure ;
```

Lab 06: Design B

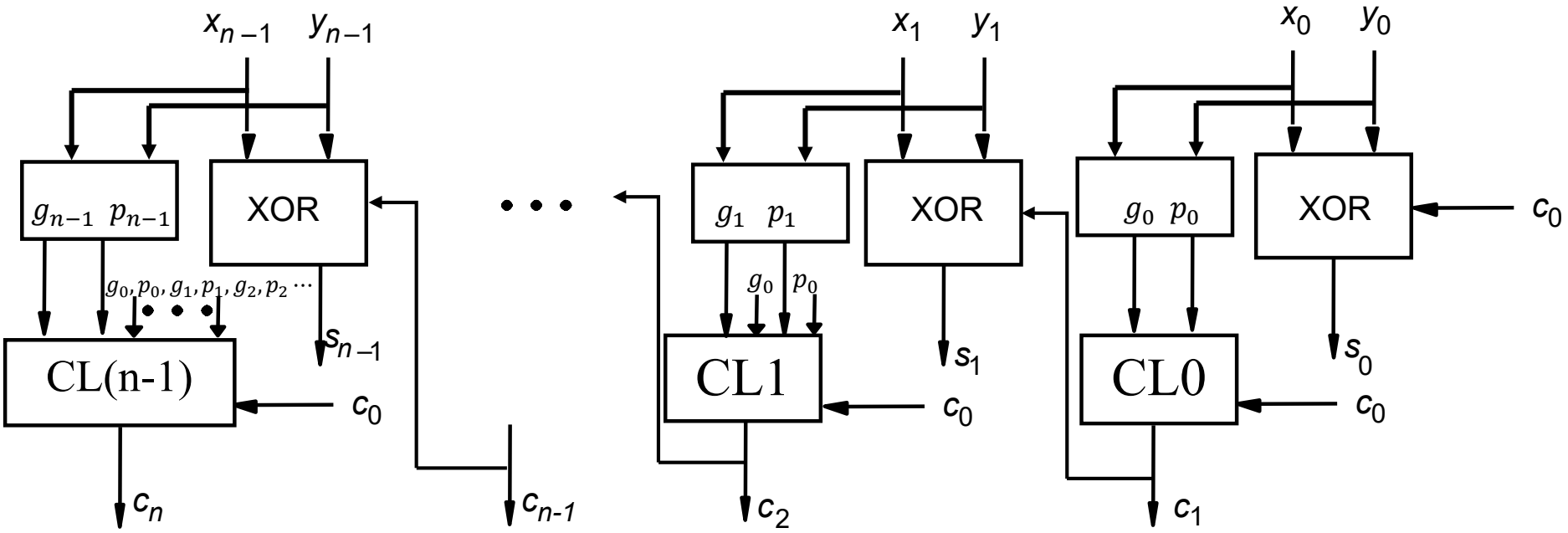
- Data-flow VHDL for the carry-lookahead 4-bit adder
  - Write down expressions for sum bits as well as carry-out bits
- You need to perform the following
  - Create a project, write your VHDL code, and compile
  - Perform functional simulation with test vectors
  - Use 7-segment LED display to show the results (in HEX format)
  - Download the design to DE1 board and test the DE1 board with test vectors

Design of CLA

- Carry-out bit:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + + p_3 p_2 p_1 p_0 c_0$
  - $\cdots$
  - $c_n = g_{n-1} + p_{n-1} g_{n-2} + p_{n-1} p_{n-2} g_{n-3} + \cdots + p_{n-1} p_{n-2} \cdots p_2 p_1 g_0 + p_{n-1} p_{n-2} \cdots p_2 p_1 p_0 c_0$
- Sum bit still: $s_{n-1} = c_{n-1} \oplus x_{n-1} \oplus y_{n-1}$

Structure of CLA

- No full adders
- CL=Carry-lookahead has inputs
  - c_0, g_0, p_0 for c_1 (stage 0)
  - c_0, g_0, p_0, g_1, p_1 for c_2 (stage 1)
  - $c_0, g_0, p_0, g_1, p_1, g_2, p_2$ for c_3 (stage 2)
  - And so on



Sign or MSB position

LSB position

Further explanation

- Generate input/output lists:
  - Input list should contain 1) vectors X, Y as data inputs, both 4-bit wide and 2) a carry-in, c0.
  - Output list should contain 1) sum vector S, 4-bit wide and 2) a carry-out, *Cout*.
- Specify the input/output relationship.
  - Establish multiple internal signals, c1, c2, c3, c4, which are the carry-out signals for each stage of the adder. See page 12, for equations of these internal signals.
  - Write the sum bits, S[0] to S[3], in terms of inputs, X, Y, and internal carry-out. See page 12 for equations
  - Finally *Cout*=c4.