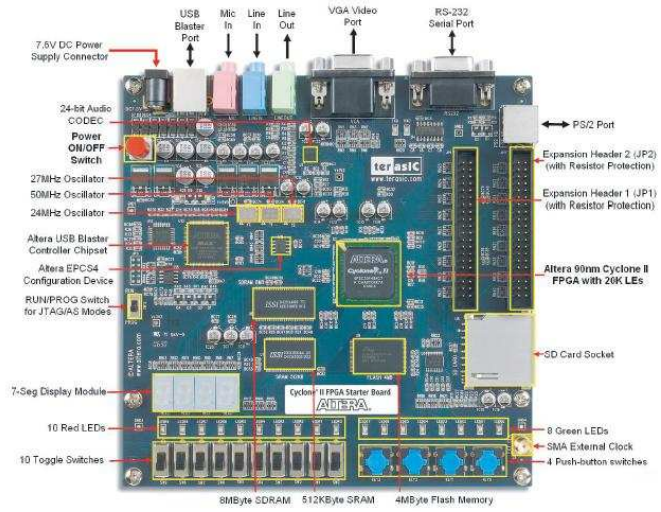
Projeto: Uma Calculadora com Componentes em VHDL

Figure 1–1. Starter Development Board



Descrição Funcional

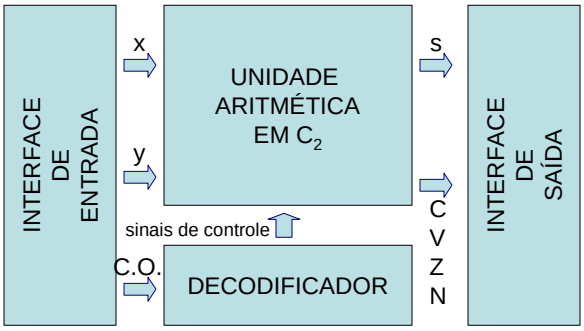
Operandos: x, y (8 *bits*)
Resultado: s (8 *bits*), C,V,Z,N (*bits* de condição)

Código de Operação		Definição	Sinais de Controle
ADD	adição	s=x+y	
SUB	subtração	s=x-y	
CMY	complemento	s=y’	
CSX	troca de sinal	s=-x	

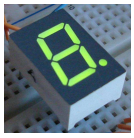
Descrição Estrutural

8 chaves

3 botoeiras

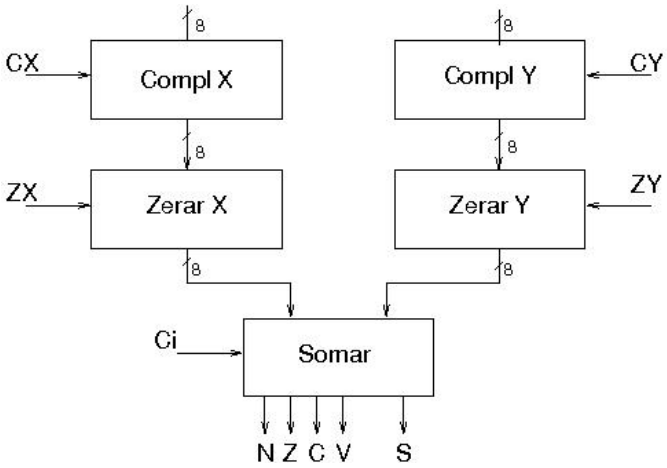


4 *leds*



2 *displays*

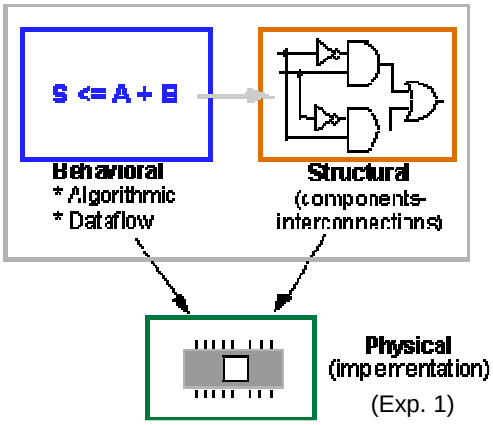
Unidade Aritmética



Sistema Digital: Níveis de Abstração

(Exps. 2 e 3)

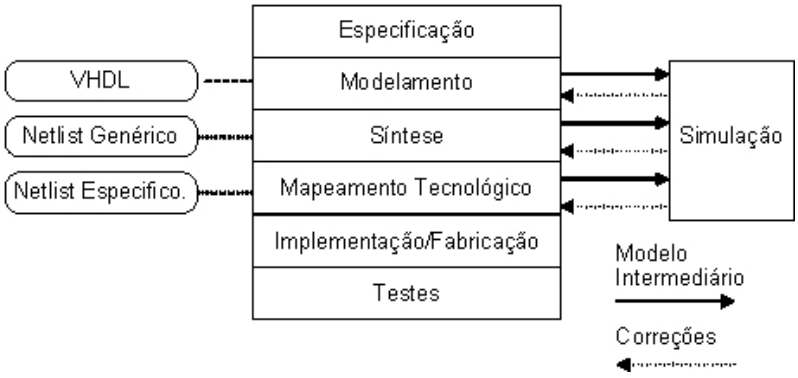
Linguagem de descrição orientada à funcionalidade do circuito → portabilidade, documentação, manutenção



VHDL

- **VHDL** = **V**HSIC (*Very High Speed Integrated Circuits) **H**ardware **D**escription **L**anguage* - 1980
  - Motivação: concisão na documentação em Departamento de Defesa dos Estados Unidos
- Outras linguagens de descrição: Verilog (1983-1984), *AHDL (**A**ltera **H**ardware **D**escription **L**anguage)*

Fluxo de Projeto



http://pt.wikipedia.org/wiki/VHDL

Estrutura Básica de VHDL

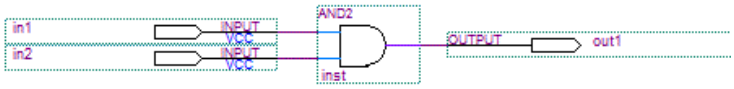
COMPONENTE

INTERFACE
(ENTITY)
Declaração de portas e objetos locais

PROCEDIMENTO
(ARCHITECTURE)
Funcional (Concorrente/Sequencial)
Estrutural

http://www.seas.upenn.edu/~ese201/vhdl/vhdl_primer.html

Modelos



Funcional

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity and2_behavioral is
      port (in1, in2: in std_logic;
            out1: out std_logic);
end entity;

architecture behavioral_2 of AND2 is
begin
      out1 <= in1 and in2;
end behavioral_2;
```

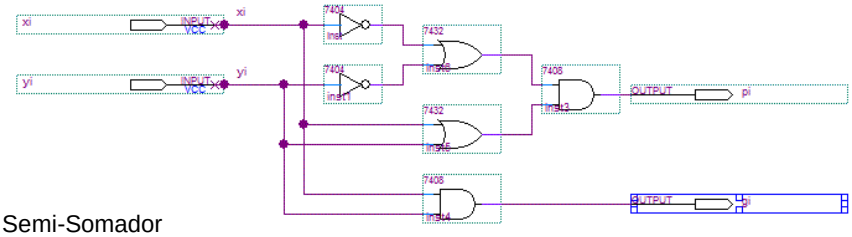
Estrutural

```
entity and2_structural_vhdl is
    port (in1, in2: in std_logic;
          out1: out std_logic);
end and2_structural_vhdl;

architecture structural of and2_structural_vhdl is
   component AND2   -- Altera primitive
      port (in1, in2: in std_logic;
            \out\: out std_logic);
   end component;
begin
   U0: AND2 port map (in1, in2, out1);
end architecture;
```

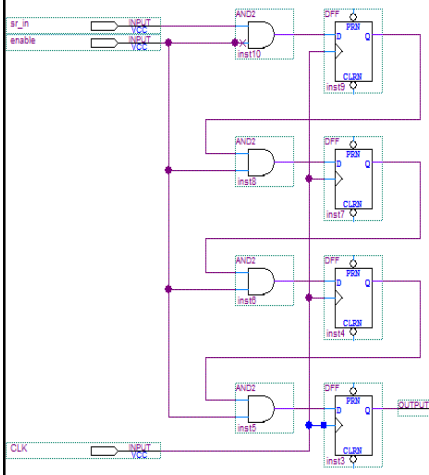
1º. Experimento

1. Analise as duas descrições em VHDL, compile-as e simule-as




```
-- Semi-Somador
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity semi_somador_vhdl is
port      (
          xi,yi          : in std_logic;
          pi,gi          : out std_logic
          );
end semi_somador_vhdl;
```

```
architecture behavior of semi_somador_vhdl is
begin
          pi <= xi xor yi;
          gi <= xi and yi;
end behavior;
```



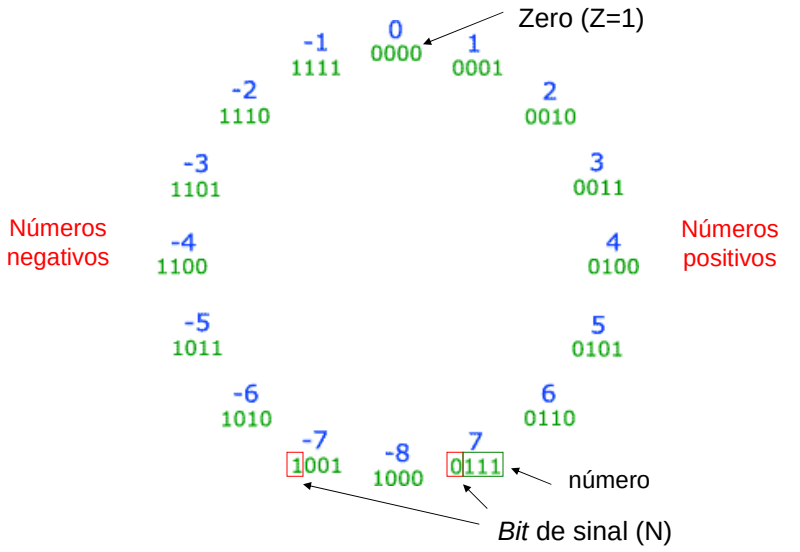

```
-- Registrador de deslocamento
library ieee;
use ieee.std_logic_1164.all;
entity registrador_deslocamento_vhdl is
   generic  ( NUM_STAGES : natural := 4 );
   port       (
              clk, enable : in std_logic;
              sr_in: in std_logic;
              sr_out   : out std_logic
              );
end entity;

architecture behavior of registrador_deslocamento_vhdl is
   type registrador is array ((NUM_STAGES-1) downto 0)
              of std_logic;
   signal sr: registrador;
   begin
      sr_out <= sr(NUM_STAGES-1);
      process (clk)
      begin
         if (rising_edge(clk)) then
              if (enable = '1') then
              sr((NUM_STAGES-1) downto 1) <=
                        sr((NUM_STAGES-2) downto 0);
              sr(0) <= sr_in;
              end if;
         end if;
      end process;
end architecture;
```

Unidade Aritmética (UA)

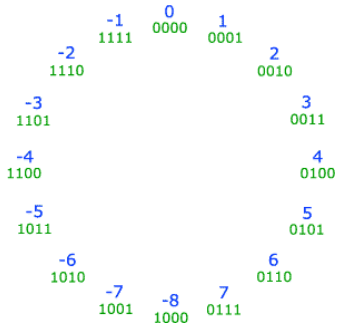
- Circuito combinacional que realiza as operações aritméticas em complemento de 2 (adição, subtração, complemento, troca de sinal)

Representação em C_2



Adição em C_2

1. Somar os dois números de n *bits*, *bit* a *bit*, inclusive o *bit* de sinal (N), e descartar o 'vai-um'.



2. Avaliação dos casos de **estouro** (V=1):

$$V = \overline{x_{n-1}}\,\overline{y_{n-1}}\,z_{n-1} + x_{n-1}\,y_{n-1}\,\overline{z_{n-1}}$$

$$V = C_{n-1} \oplus C_n$$

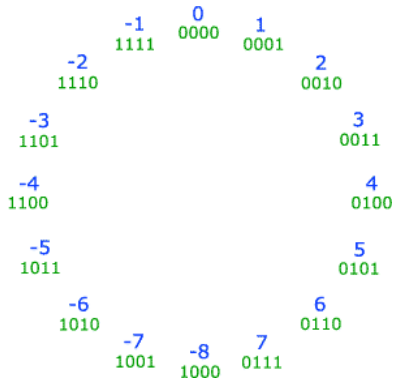
x_{n-1}	y_{n-1}	z_{n-1}	V	C_{n-1}	C_n
0	0	0	0	0	0
1	0	0	0	1	1
0	1	0	0	1	1
1	1	0	1	0	1
0	0	1	1	1	0
1	0	1	0	0	0
0	1	1	0	0	0
1	1	1	0	1	1

Subtração em C_2

1º Passo >	1111 – 1001	Subtração primária
2º Passo >	1001 → 0110 (original) (invertido)	Inversão dos números do subtraendo.
3º Passo >	0110+ 1 0111	Soma-se "1" ao número invertido.
4º Passo >	1111+ 0111 10110	Soma-se com o minuendo.
5º Passo >	10110 → 0110	Retira o número "1" à esqueda.

Complemento e Troca de Sinal

- Complemento: complementar *bit* a *bit.*
- Troca de sinal: complementar *bit* a *bit* e somar “1”.



2º. Experimento

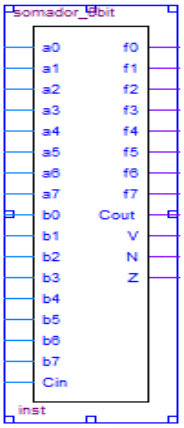
1. Analise a descrição de um somador de 4 *bits* por meio de simulações.

```
library  ieee;
use  ieee.std_logic_1164.all;

entity somador_4bit is
    port (a0, a1, a2, a3, b0, b1, b2, b3: in std_logic;
          Cin : in std_logic;
          f0, f1, f2, f3: out std_logic;
          Cout, V, N, Z: out std_logic);
end somador_4bit;
architecture fouradder_structure of somador_4bit is
    signal c, sum: std_logic_vector (4 downto 0);
    component FULLADDER
          port(a, b, c: in std_logic;
                sum, carry: out std_logic);
    end component;
begin
   FA0: FULLADDER port map (a0, b0, Cin, sum(0), c(1));
   FA1: FULLADDER port map (a1, b1, C(1), sum(1), c(2));
   FA2: FULLADDER port map (a2, b2, C(2), sum(2), c(3));
   FA3: FULLADDER port map (a3, b3, C(3), sum(3), c(4));
    V <= c(3) xor c(4);
    Cout <= c(4);
    f0 <= sum(0);
    f1 <= sum(1);
    f2 <= sum(2);
    f3 <= sum(3);
    Z <= not(sum(0) or sum(1) or sum(2) or sum(3));
   N <= sum(3);
end fouradder_structure;
```

2º. Experimento

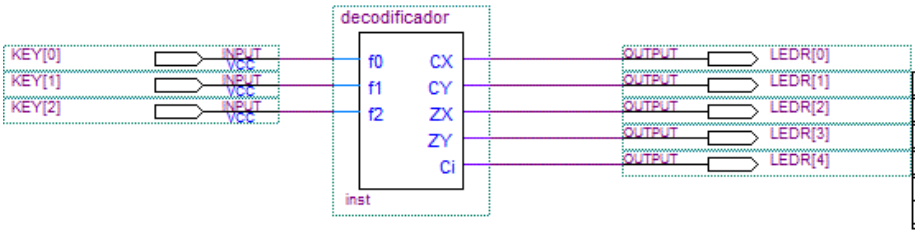
2. Estenda a descirção para 8 *bits.*



3. Altere a descrição do circuito de forma que o componente fique também sensível aos sinais de controle CX, CY, ZX e ZY.
4. Simule o circuito para os casos listados no roteiro.

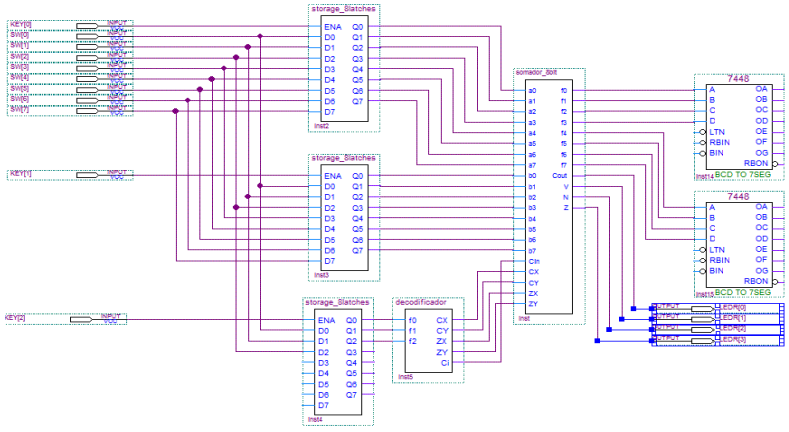
3º. Experimento

1. Sintetize a lógica do decodificador.
2. Descreva o comportamento lógico do decodificador com uso de VHDL.
3. Simule o circuito para todos os códigos de operação.
4. Programe o circuito e teste todos os casos simulados. Monitore visualmente os sinais com uso de *leds*.



4º. Experimento

1. Projete o circuito de interface de entrada com uso de *latches*.
2. Projete o circuito de interface de saída para 2 *displays* e 4 *leds* vermelhos.



3. Simule o circuito para todos os casos do 2º. Experimento.
4. Programe o circuito e teste todos os casos simulados.