Lógica e Sistemas Digitais

Dados e Controlo microprogramado

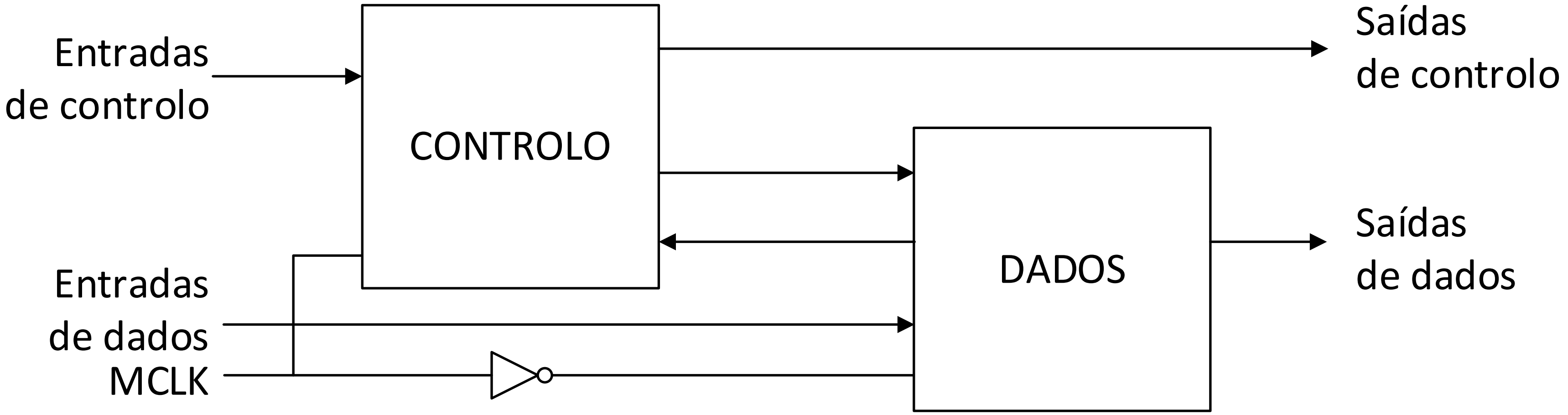
João Pedro Patriarca (jpatri@cc.isel.ipl.pt)

Slides inspirados nos slides do prof. Mário Véstias

ISEL

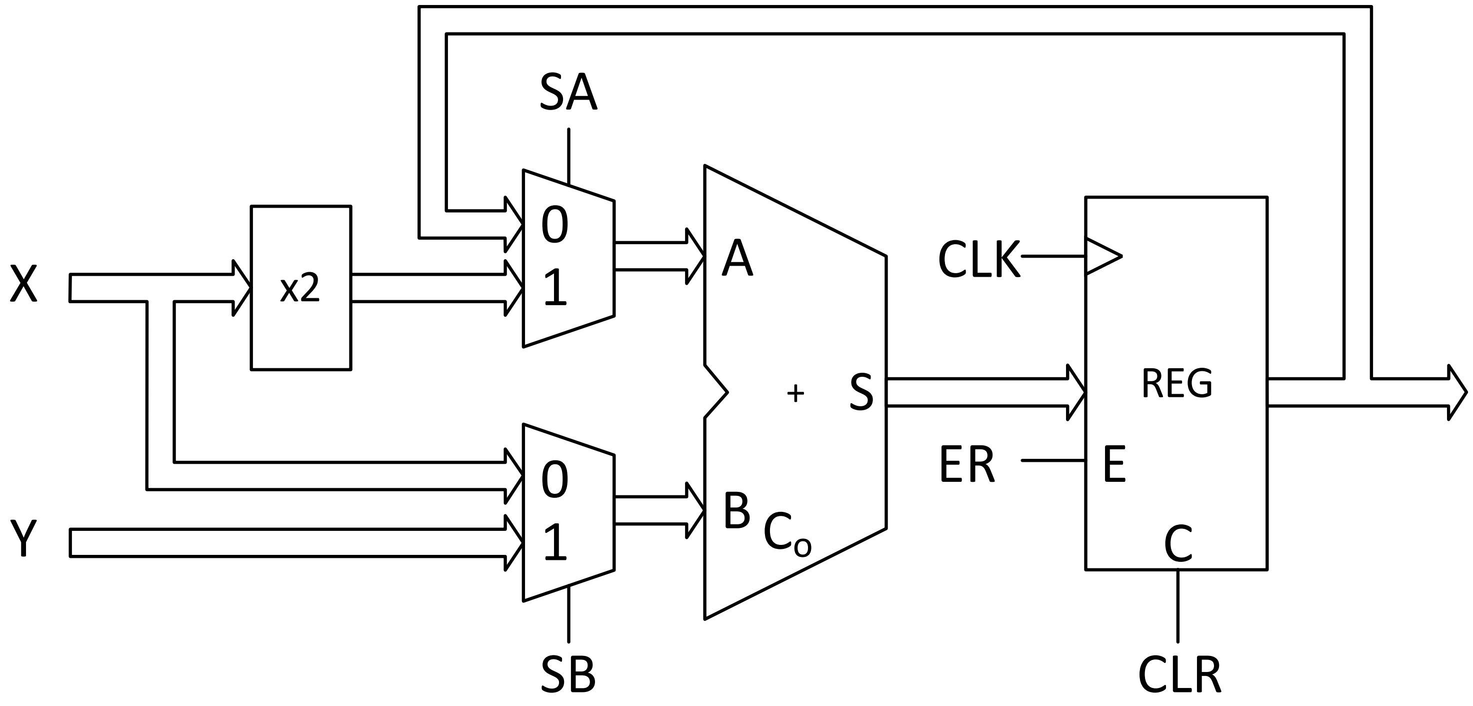
Circuito de dados e controlo

- Dois circuitos principais
  - Circuito de controlo ou módulo de controlo
  - Circuito de dados ou módulos funcionais



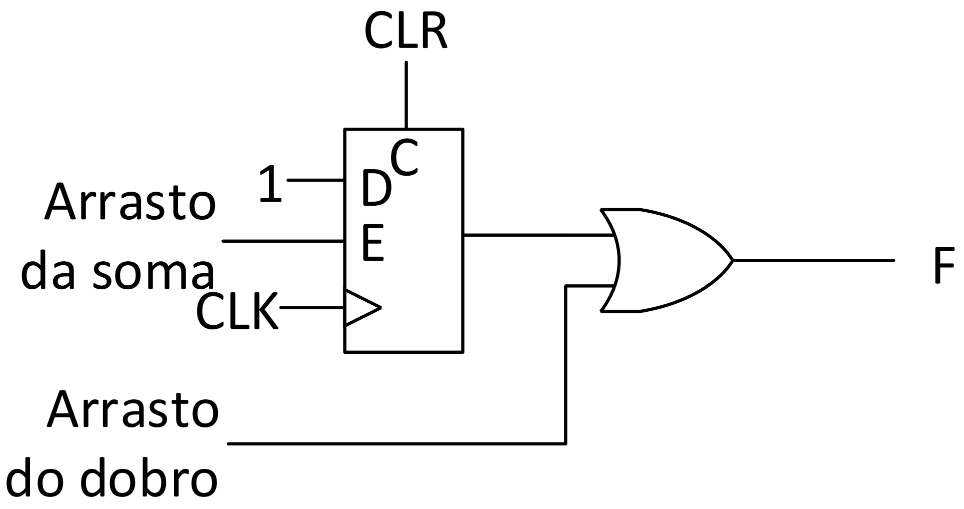
Exemplo – recursos para realizar a operação R = 3X+Y

- Processo para produzir o resultado 3X+Y
  1. Temp1 = 2 * X + X
  2. R = Temp1 + Y

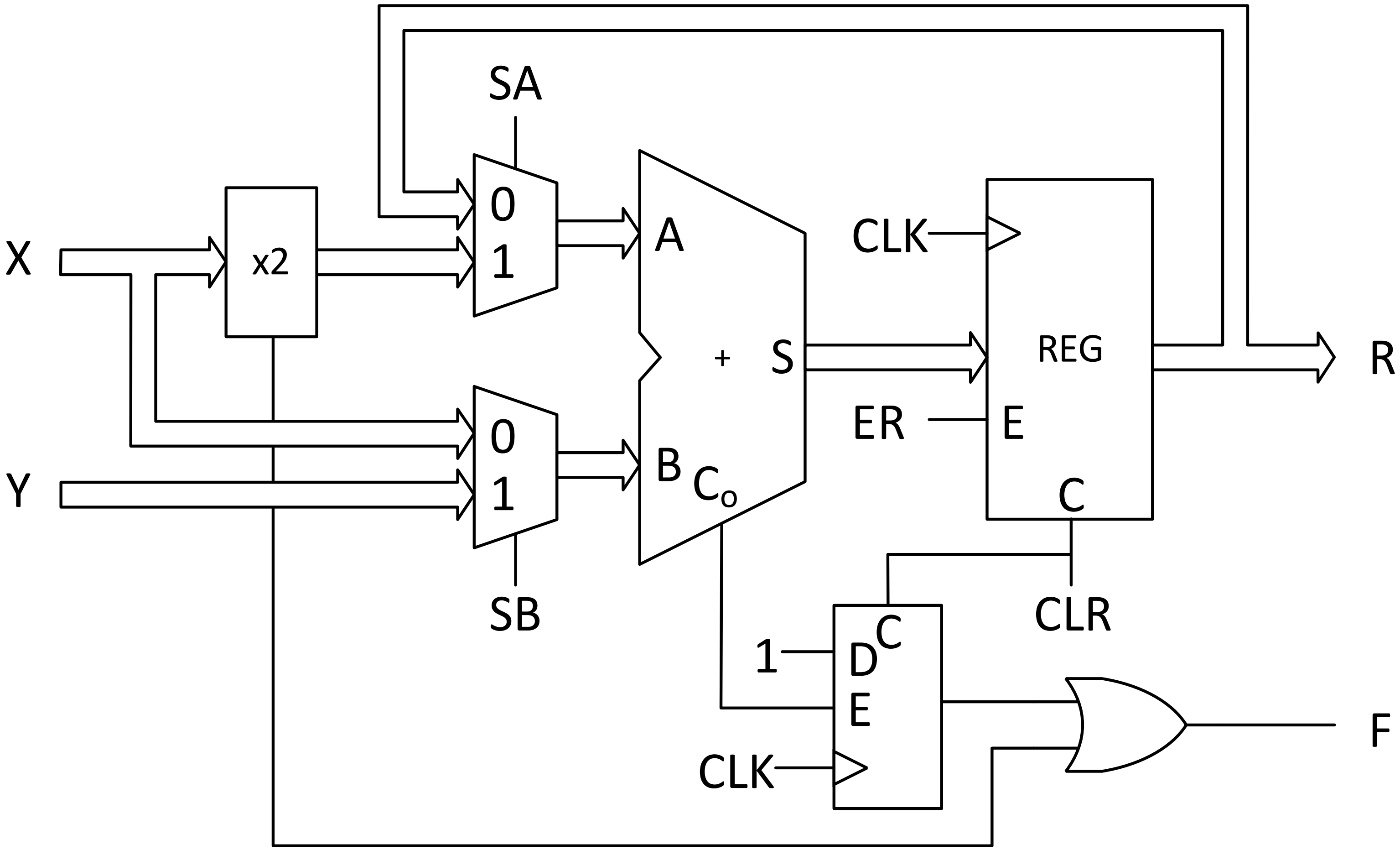


Exemplo – flag de erro

- Produz uma *flag* que fica ativa em caso de resultado com erro
- Processo para produzir a *flag* de erro
    1. Erro se 2 * X produzir arrasto
    2. Erro se 2 * X + X produzir arrasto
    3. Erro se Temp1 + Y produzir arrasto

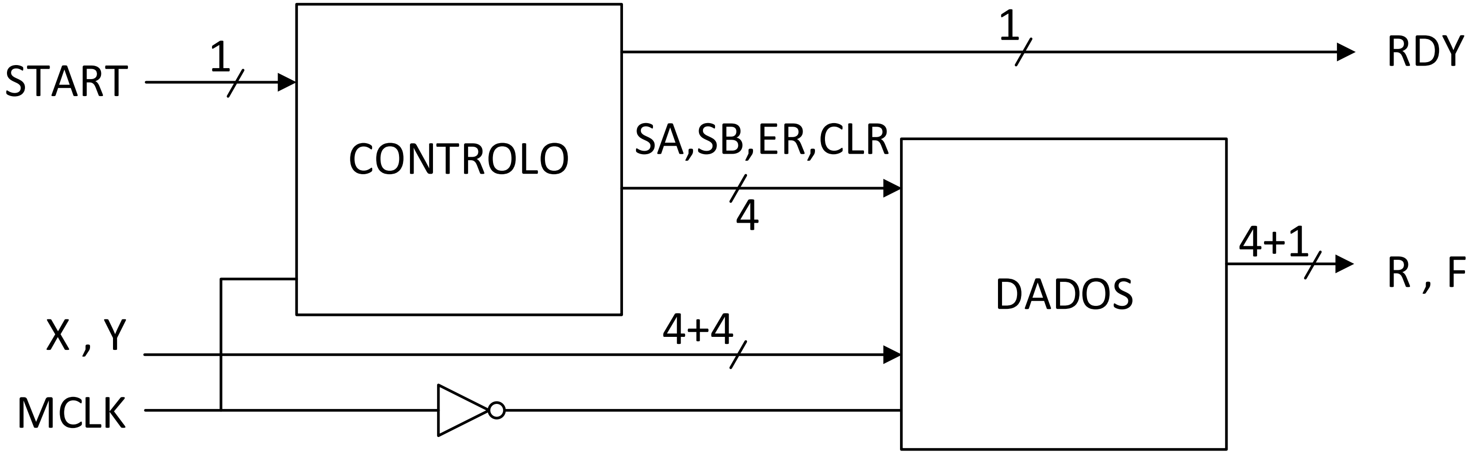


Exemplo – circuito de dados completo

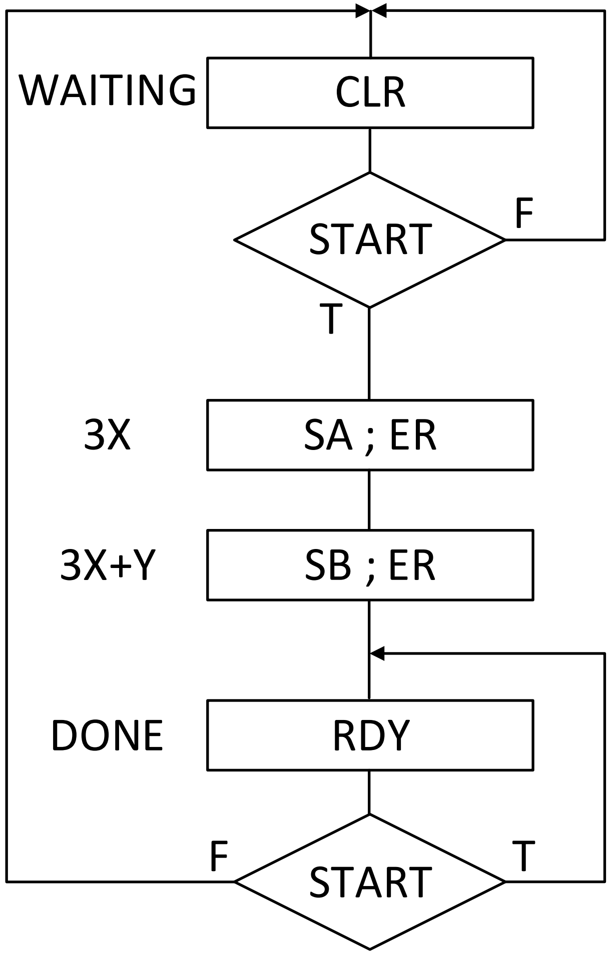


Exemplo – circuito de dados e controlo

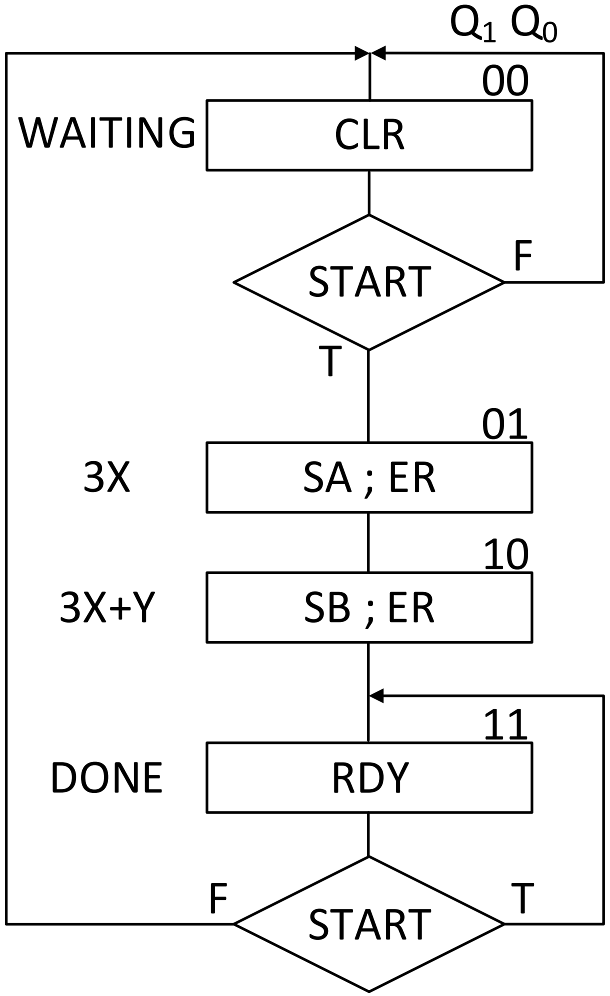
- Entrada START para indicar início de operação
- Mantém último resultado enquanto START ativo
- Saída RDY para indicar resultado produzido
- *Clock* do controlo e dados a funcionar com o mesmo período e em oposição de fase



Exemplo – sistema de controlo

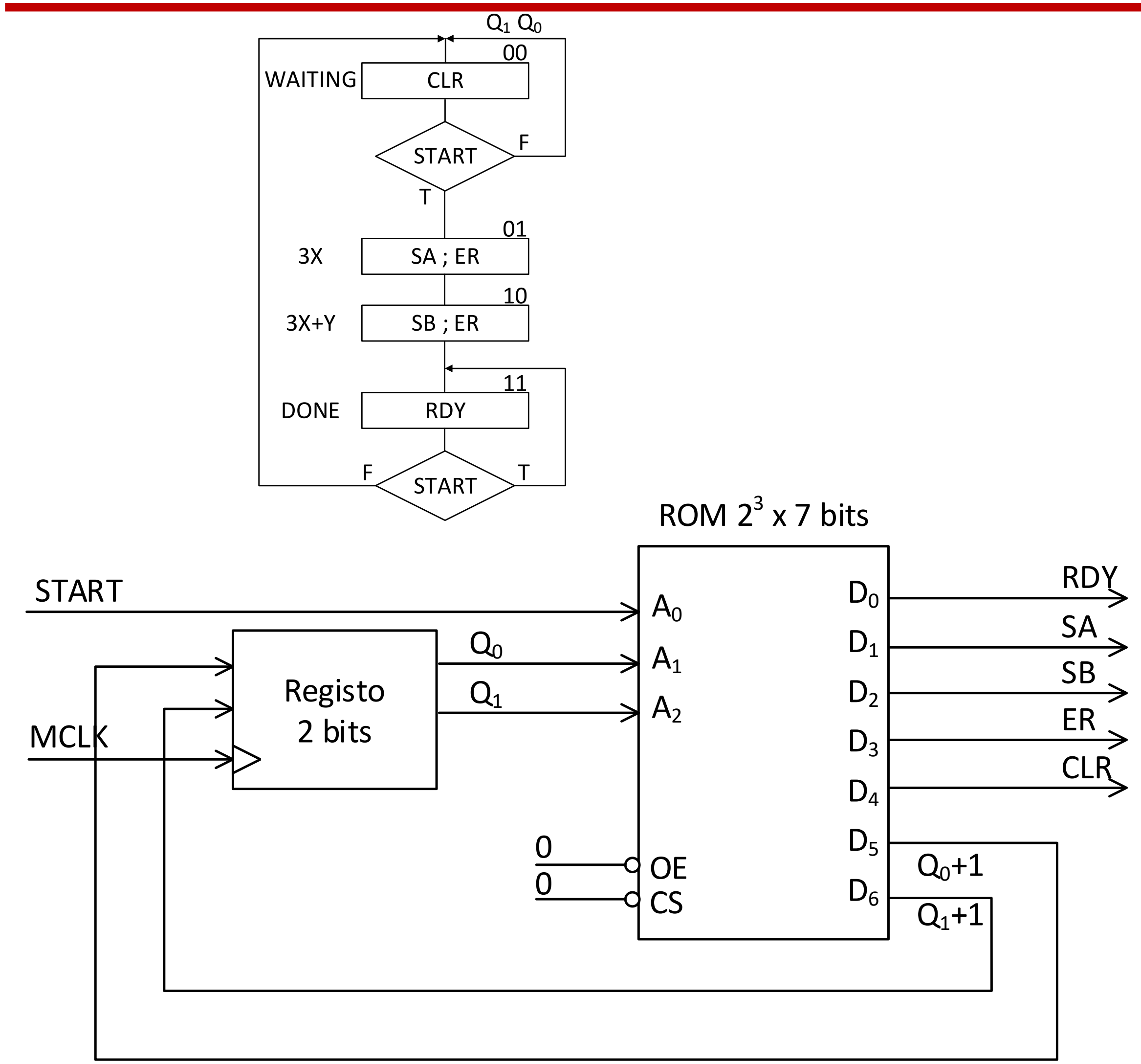


Exemplo – sistema de controlo – tabela de transição de estados e saídas



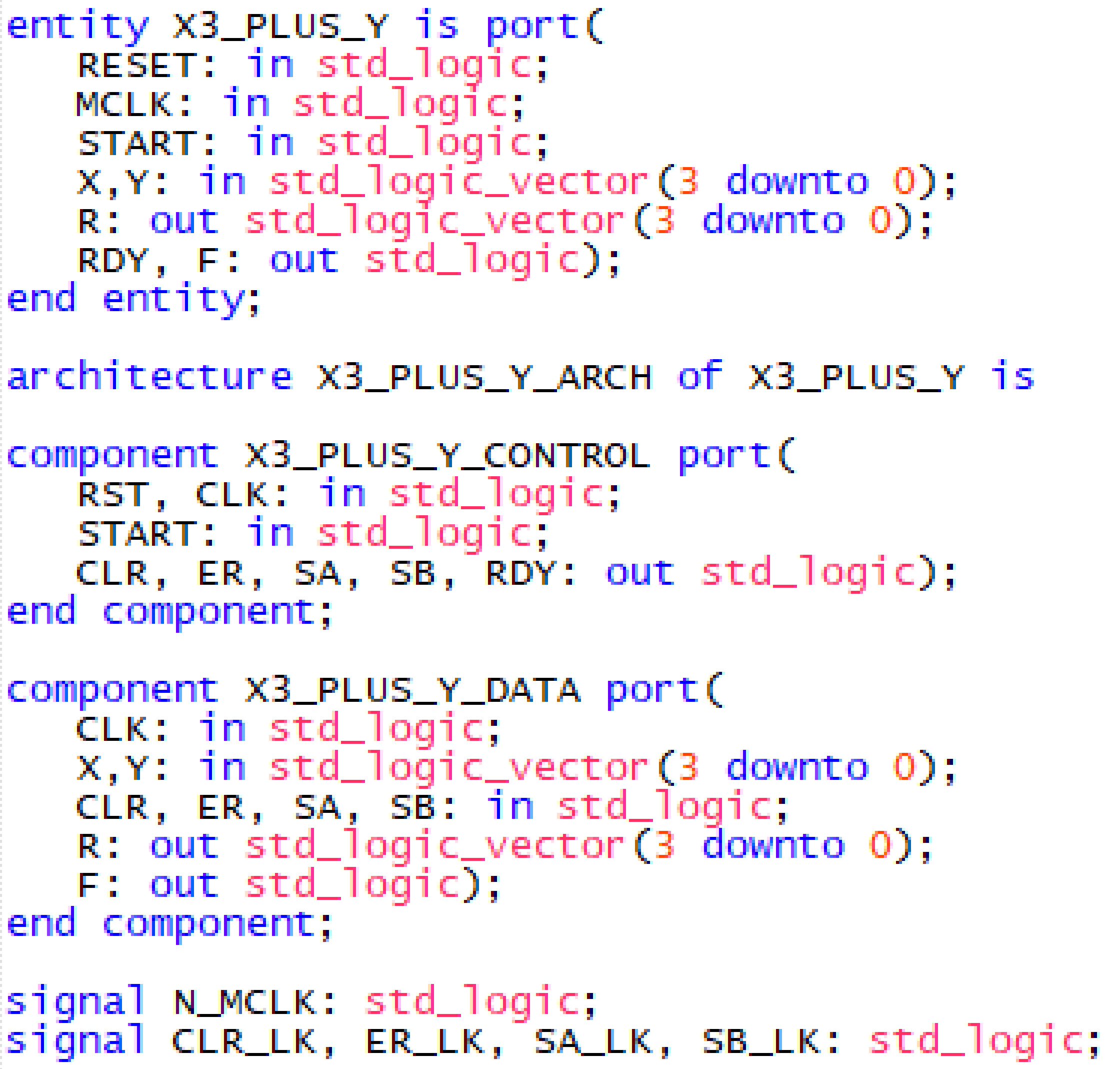
EP		Ent.	ES		Saídas				
Q_1	Q_0	START	Q_1+1	Q_0+1	CLR	ER	SB	SA	RDY
0	0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	0	0	0
0	1	-	1	0	0	1	0	1	0
1	0	-	1	1	0	1	1	0	0
1	1	0	0	0	0	0	0	0	1
1	1	1	1	1	0	0	0	0	1

Exemplo – sistema de controlo – controlo microprogramado



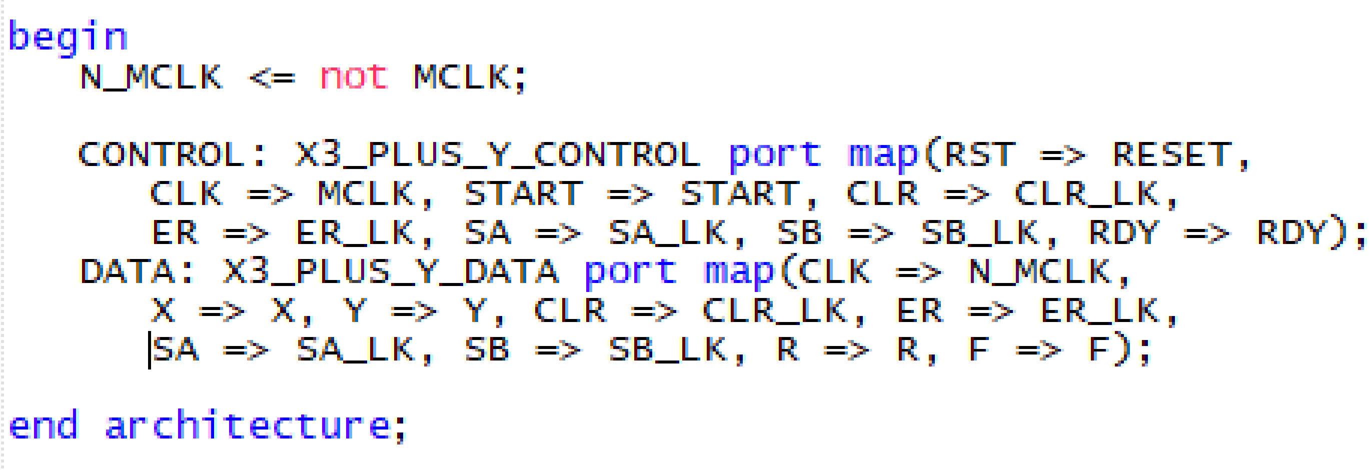
Endereços			Dados						
A_2	A_1	A_0	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Q_1	Q_0	START	$Q_1 + 1$	$Q_0 + 1$	CLR	ER	SB	SA	RDY
0	0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	0	0	0
0	1	0	1	0	0	1	0	1	0
0	1	1	1	0	0	1	0	1	0
1	0	0	1	1	0	1	1	0	0
1	0	1	1	1	0	1	1	0	0
1	1	0	0	0	0	0	0	0	1
1	1	1	1	1	0	0	0	0	1

Exemplo – VHDL da entidade de topo (1 de 2)



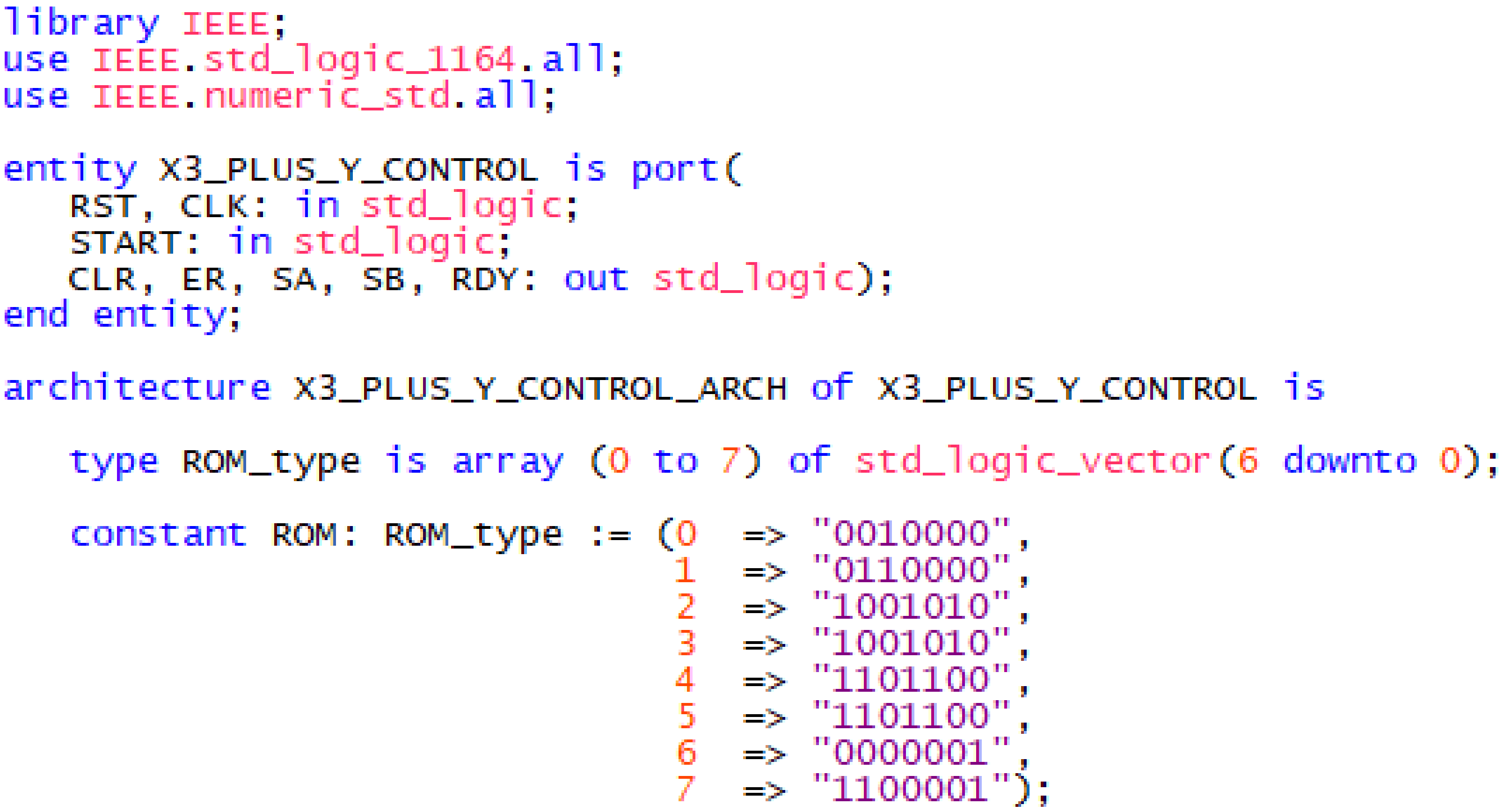
```vhdl
entity X3_PLUS_Y is port(
   RESET: in std_logic;
   MCLK: in std_logic;
   START: in std_logic;
   X,Y: in std_logic_vector(3 downto 0);
   R: out std_logic_vector(3 downto 0);
   RDY, F: out std_logic);
end entity;

architecture X3_PLUS_Y_ARCH of X3_PLUS_Y is

component X3_PLUS_Y_CONTROL port(
   RST, CLK: in std_logic;
   START: in std_logic;
   CLR, ER, SA, SB, RDY: out std_logic);
end component;

component X3_PLUS_Y_DATA port(
   CLK: in std_logic;
   X,Y: in std_logic_vector(3 downto 0);
   CLR, ER, SA, SB: in std_logic;
   R: out std_logic_vector(3 downto 0);
   F: out std_logic);
end component;

signal N_MCLK: std_logic;
signal CLR_LK, ER_LK, SA_LK, SB_LK: std_logic;
```

Exemplo – VHDL da entidade de topo (2 de 2)



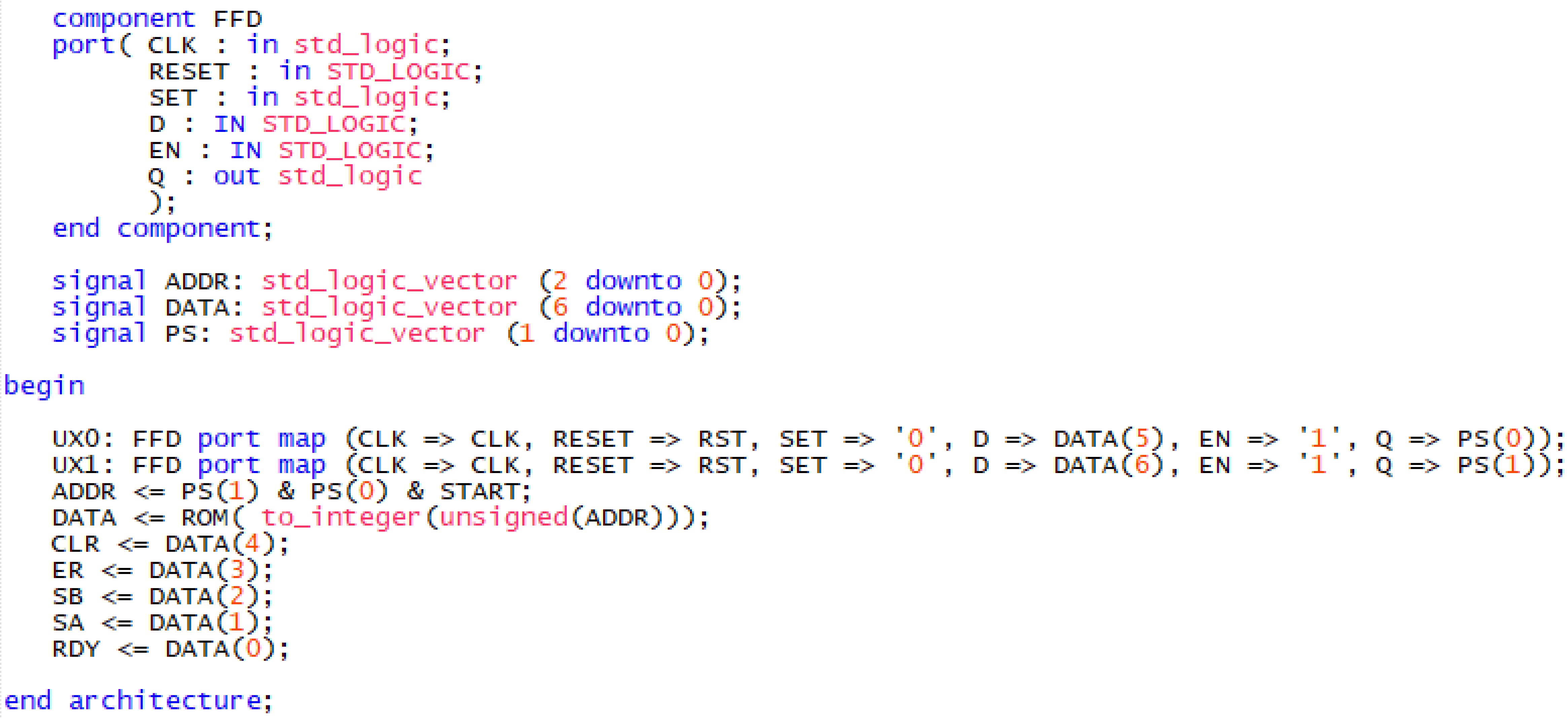
```vhdl
begin
   N_MCLK <= not MCLK;

   CONTROL: X3_PLUS_Y_CONTROL port map(RST => RESET,
      CLK => MCLK, START => START, CLR => CLR_LK,
      ER => ER_LK, SA => SA_LK, SB => SB_LK, RDY => RDY);
   DATA: X3_PLUS_Y_DATA port map(CLK => N_MCLK,
      X => X, Y => Y, CLR => CLR_LK, ER => ER_LK,
      SA => SA_LK, SB => SB_LK, R => R, F => F);

end architecture;
```

Exemplo – VHDL da entidade de controlo (1 de 2)



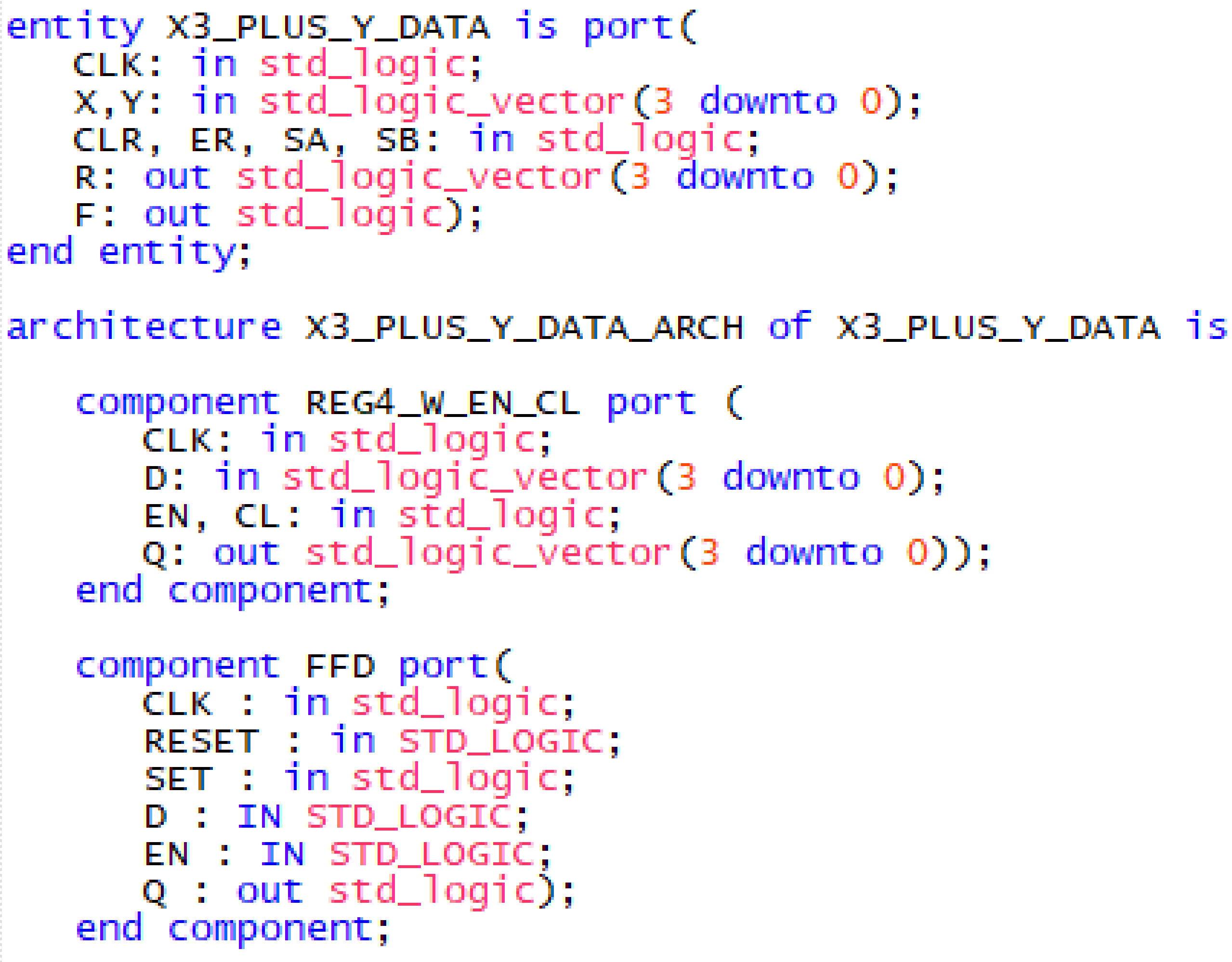
```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity X3_PLUS_Y_CONTROL is port(
   RST, CLK: in std_logic;
   START: in std_logic;
   CLR, ER, SA, SB, RDY: out std_logic);
end entity;

architecture X3_PLUS_Y_CONTROL_ARCH of X3_PLUS_Y_CONTROL is

   type ROM_type is array (0 to 7) of std_logic_vector(6 downto 0);

   constant ROM: ROM_type := (0  => "0010000",
                              1  => "0110000",
                              2  => "1001010",
                              3  => "1001010",
                              4  => "1101100",
                              5  => "1101100",
                              6  => "0000001",
                              7  => "1100001");
```

Exemplo – VHDL da entidade de controlo (2 de 2)



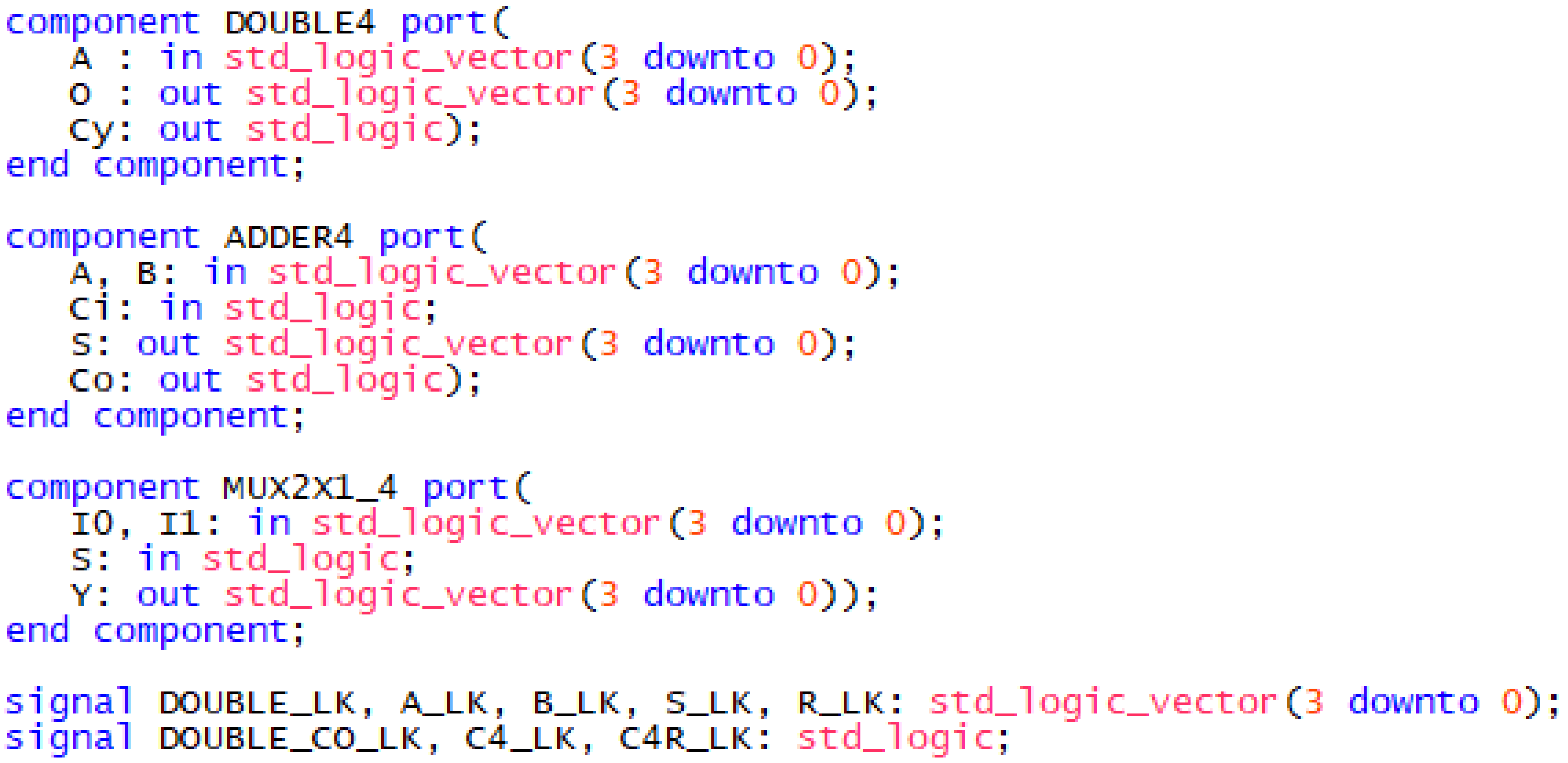
```
   component FFD
   port( CLK : in std_logic;
         RESET : in STD_LOGIC;
         SET : in std_logic;
         D : IN STD_LOGIC;
         EN : IN STD_LOGIC;
         Q : out std_logic
         );
   end component;

   signal ADDR: std_logic_vector (2 downto 0);
   signal DATA: std_logic_vector (6 downto 0);
   signal PS: std_logic_vector (1 downto 0);

begin

   UX0: FFD port map (CLK => CLK, RESET => RST, SET => '0', D => DATA(5), EN => '1', Q => PS(0));
   UX1: FFD port map (CLK => CLK, RESET => RST, SET => '0', D => DATA(6), EN => '1', Q => PS(1));
   ADDR <= PS(1) & PS(0) & START;
   DATA <= ROM( to_integer(unsigned(ADDR)));
   CLR <= DATA(4);
   ER <= DATA(3);
   SB <= DATA(2);
   SA <= DATA(1);
   RDY <= DATA(0);

end architecture;
```

Exemplo – VHDL da entidade de dados (1 de 3)



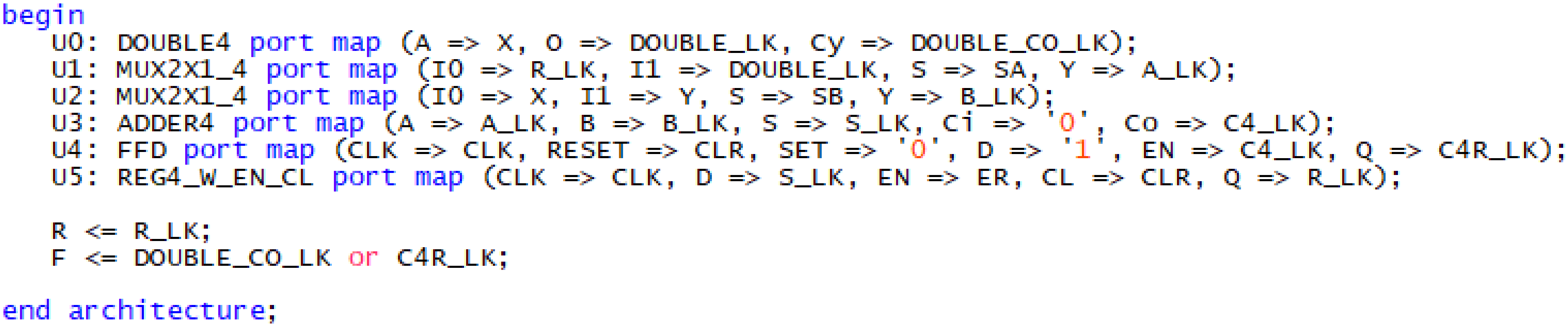
```
entity X3_PLUS_Y_DATA is port(
   CLK: in std_logic;
   X,Y: in std_logic_vector(3 downto 0);
   CLR, ER, SA, SB: in std_logic;
   R: out std_logic_vector(3 downto 0);
   F: out std_logic);
end entity;

architecture X3_PLUS_Y_DATA_ARCH of X3_PLUS_Y_DATA is

   component REG4_W_EN_CL port (
      CLK: in std_logic;
      D: in std_logic_vector(3 downto 0);
      EN, CL: in std_logic;
      Q: out std_logic_vector(3 downto 0));
   end component;

   component FFD port(
      CLK : in std_logic;
      RESET : in STD_LOGIC;
      SET : in std_logic;
      D : IN STD_LOGIC;
      EN : IN STD_LOGIC;
      Q : out std_logic);
   end component;
```

Exemplo – VHDL da entidade de dados (2 de 3)



```vhdl
component DOUBLE4 port(
   A : in std_logic_vector(3 downto 0);
   O : out std_logic_vector(3 downto 0);
   Cy: out std_logic);
end component;

component ADDER4 port(
   A, B: in std_logic_vector(3 downto 0);
   Ci: in std_logic;
   S: out std_logic_vector(3 downto 0);
   Co: out std_logic);
end component;

component MUX2X1_4 port(
   I0, I1: in std_logic_vector(3 downto 0);
   S: in std_logic;
   Y: out std_logic_vector(3 downto 0));
end component;

signal DOUBLE_LK, A_LK, B_LK, S_LK, R_LK: std_logic_vector(3 downto 0);
signal DOUBLE_CO_LK, C4_LK, C4R_LK: std_logic;
```

Exemplo – VHDL da entidade de dados (3 de 3)



```vhdl
begin
   U0: DOUBLE4 port map (A => X, O => DOUBLE_LK, Cy => DOUBLE_CO_LK);
   U1: MUX2X1_4 port map (I0 => R_LK, I1 => DOUBLE_LK, S => SA, Y => A_LK);
   U2: MUX2X1_4 port map (I0 => X, I1 => Y, S => SB, Y => B_LK);
   U3: ADDER4 port map (A => A_LK, B => B_LK, S => S_LK, Ci => '0', Co => C4_LK);
   U4: FFD port map (CLK => CLK, RESET => CLR, SET => '0', D => '1', EN => C4_LK, Q => C4R_LK);
   U5: REG4_W_EN_CL port map (CLK => CLK, D => S_LK, EN => ER, CL => CLR, Q => R_LK);

   R <= R_LK;
   F <= DOUBLE_CO_LK or C4R_LK;

end architecture;
```