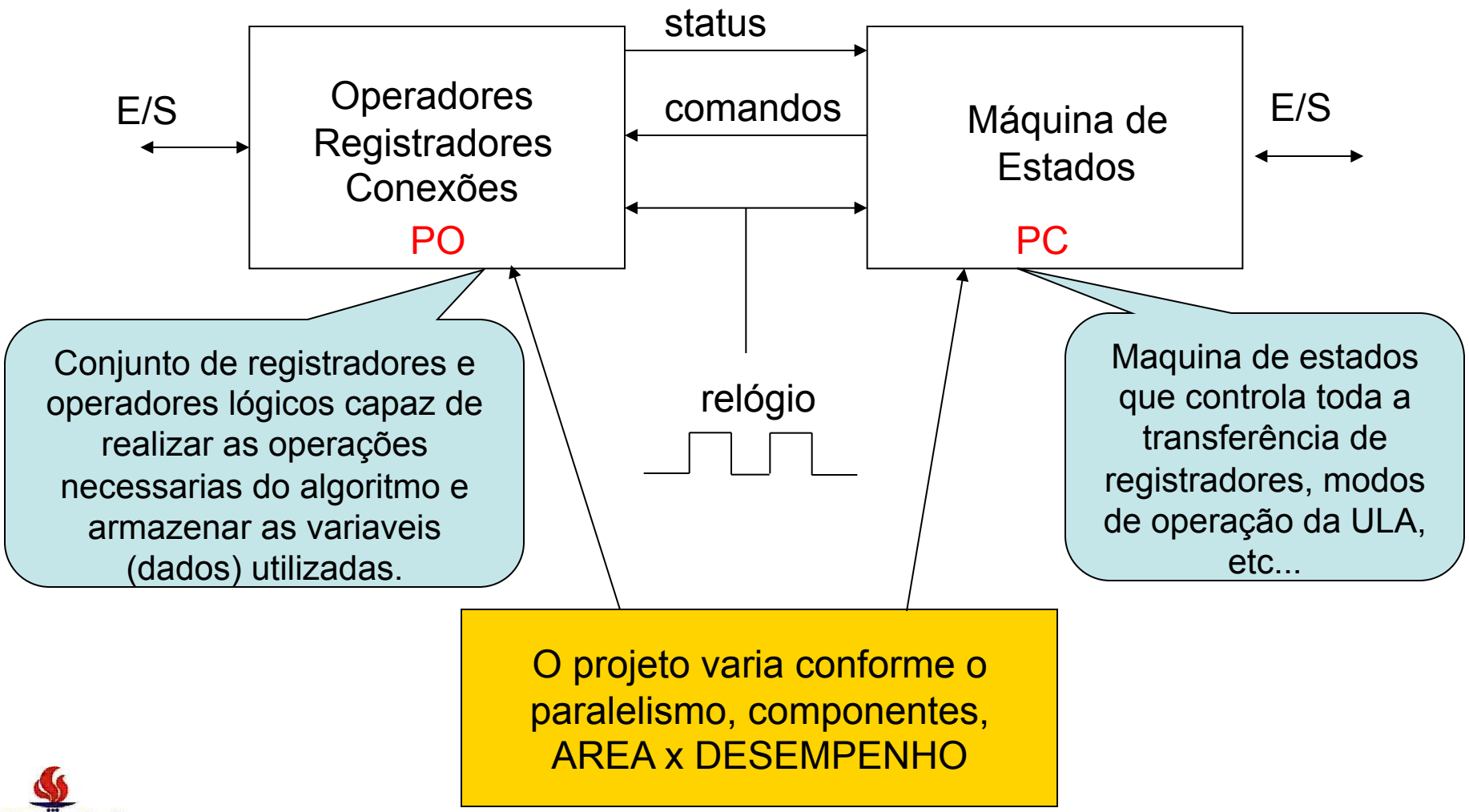
Introdução a Sistemas Digitais

- Projeto Parte de Controle e Parte Operativa
- Descrição em linguagem de hardware RTL
- Implementação SERIAL x PARALELA

Parte Operativa – Parte de Controle

Descrição a nivel de transferencia entre registradores (RTL)



Exemplo 1

Projete um bloco de controle (represente na forma de diagrama de estados) que realize a seguinte operação no datapath a seguir:

$S = A.X^2 + B.X + C$

.

```
inicio: X <= novo valor
        Start =1;
        Wait until done=1
       go to inicio;
```

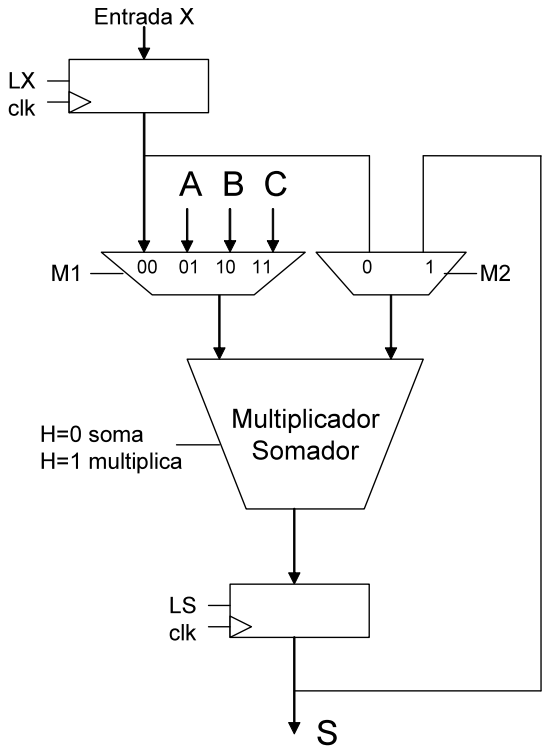
Implementação PARALELA : cada operação tem o seu operador, ou seja, precisamos de 3 multiplicadores e 2 somadores.

Implementação SERIAL : um único operador para somar e multiplicar, ou seja, uma operação por ciclo de relógio, precisa ter registrador para armazenar os estados intermediarios

Exemplo 1

VERSÃO SERIAL

$S = A.X^2 + B.X + C$



A parte operativa tem carater acumulativo, ou seja, multiplica ou soma e acumula no registrador da saida do operador lógico.

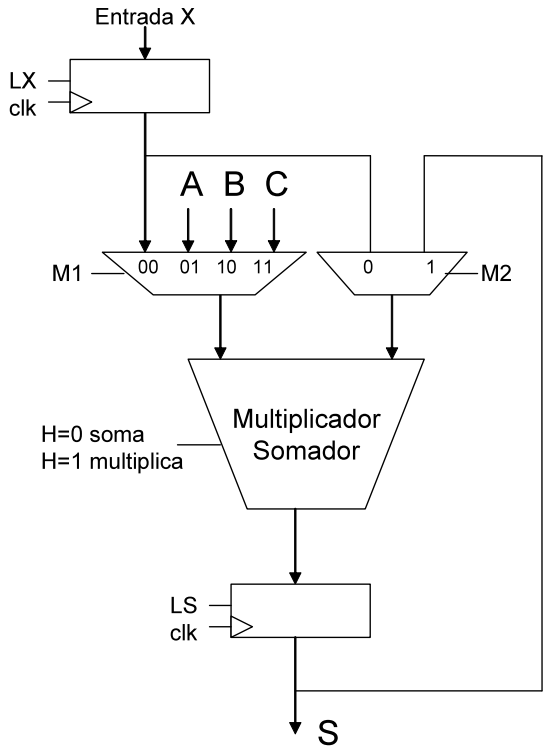
Modificamos a equação

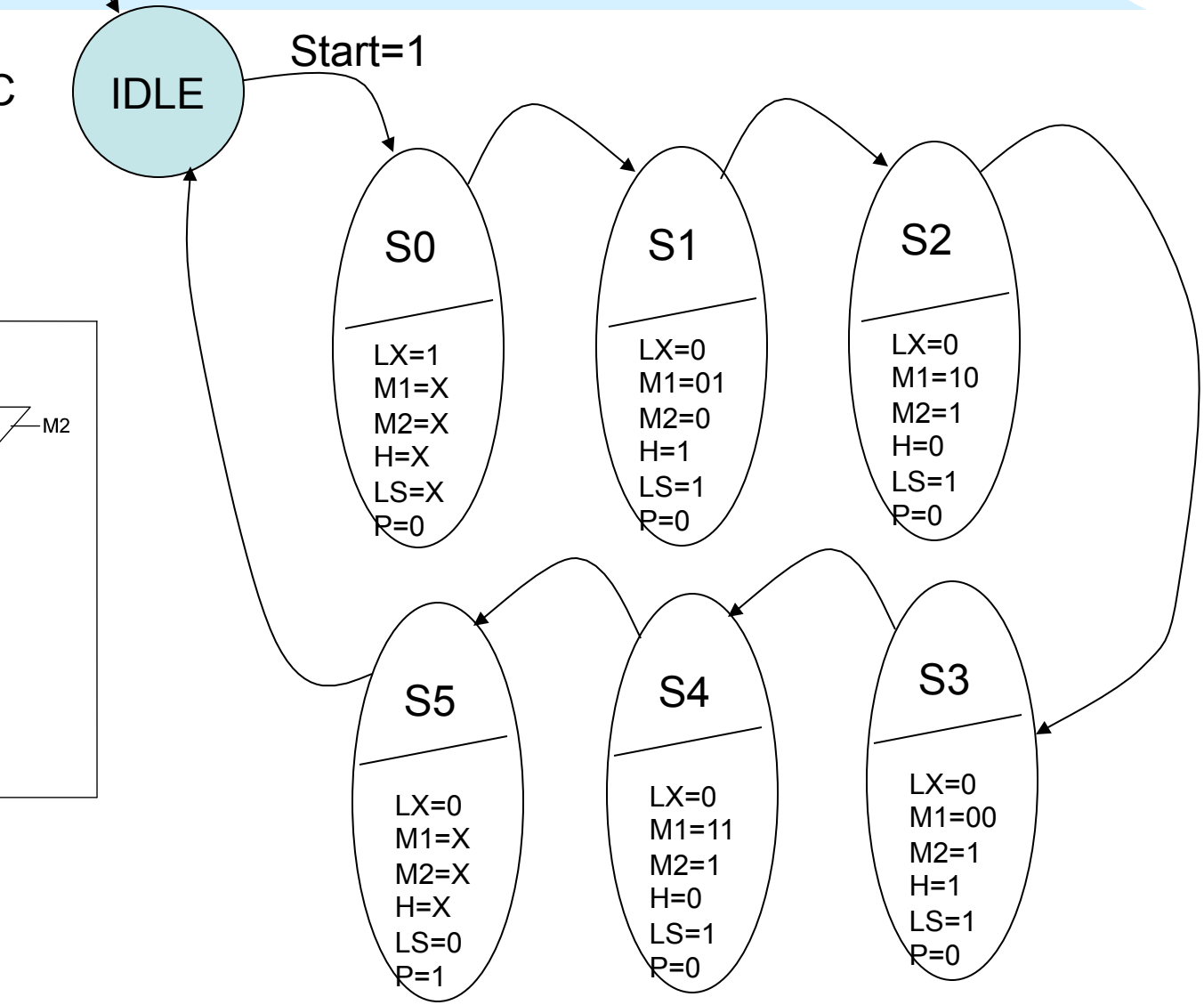
$S = X (A.X + B) + C$

Assim, para calcular S temos A que multiplica por X que soma com B que multiplica por X e soma com C.

Exemplo 1

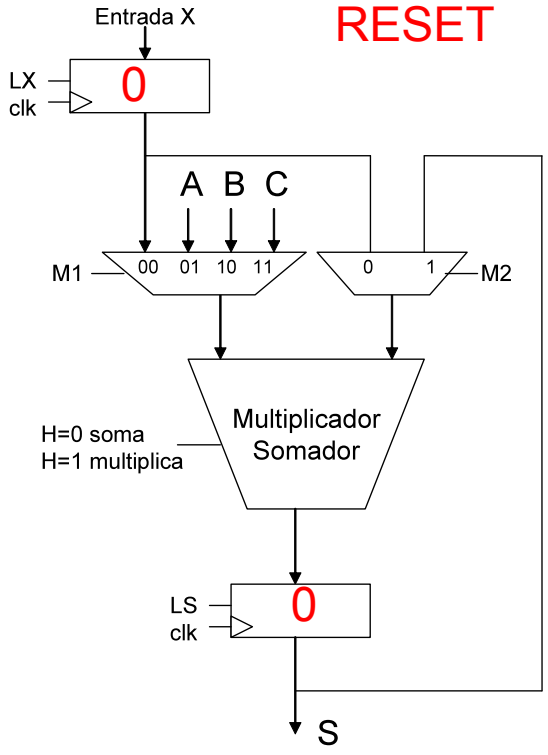
S= X (A.X + B) + C



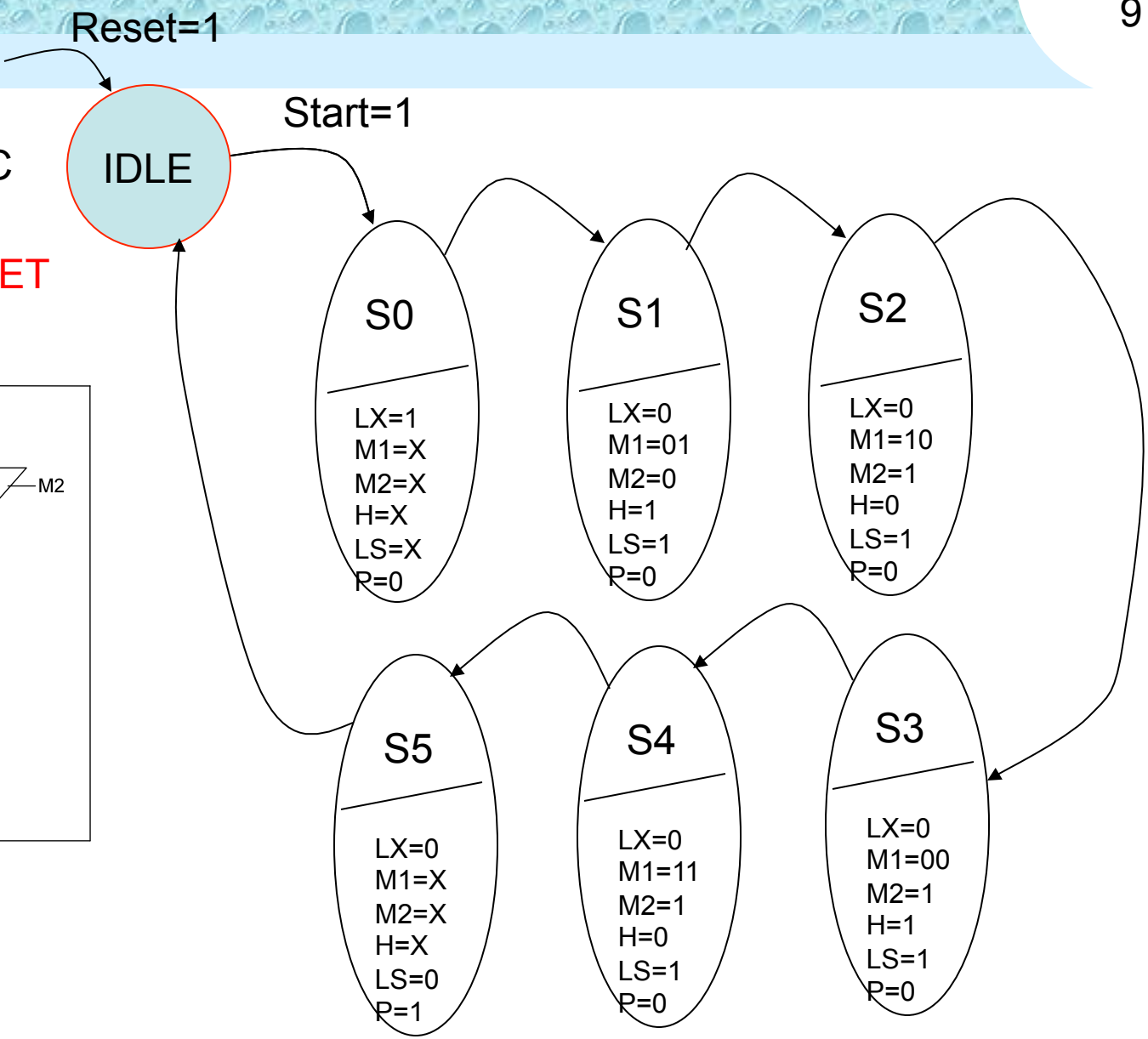


Exemplo 1

S= X (A.X + B) + C

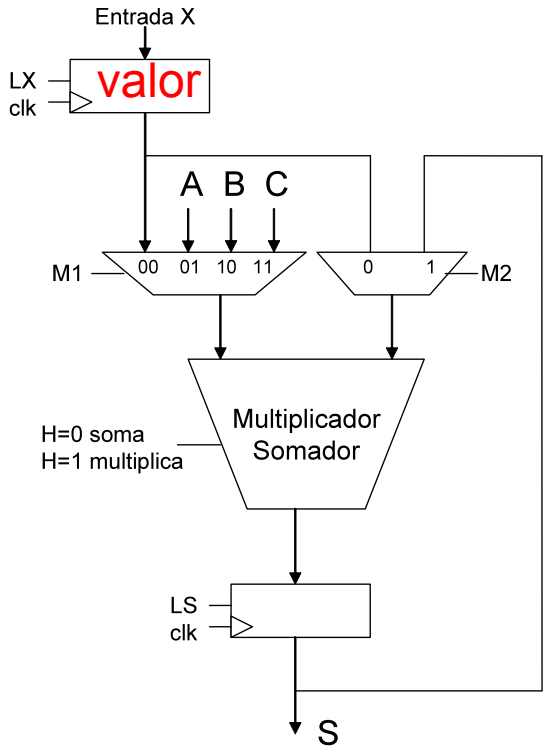


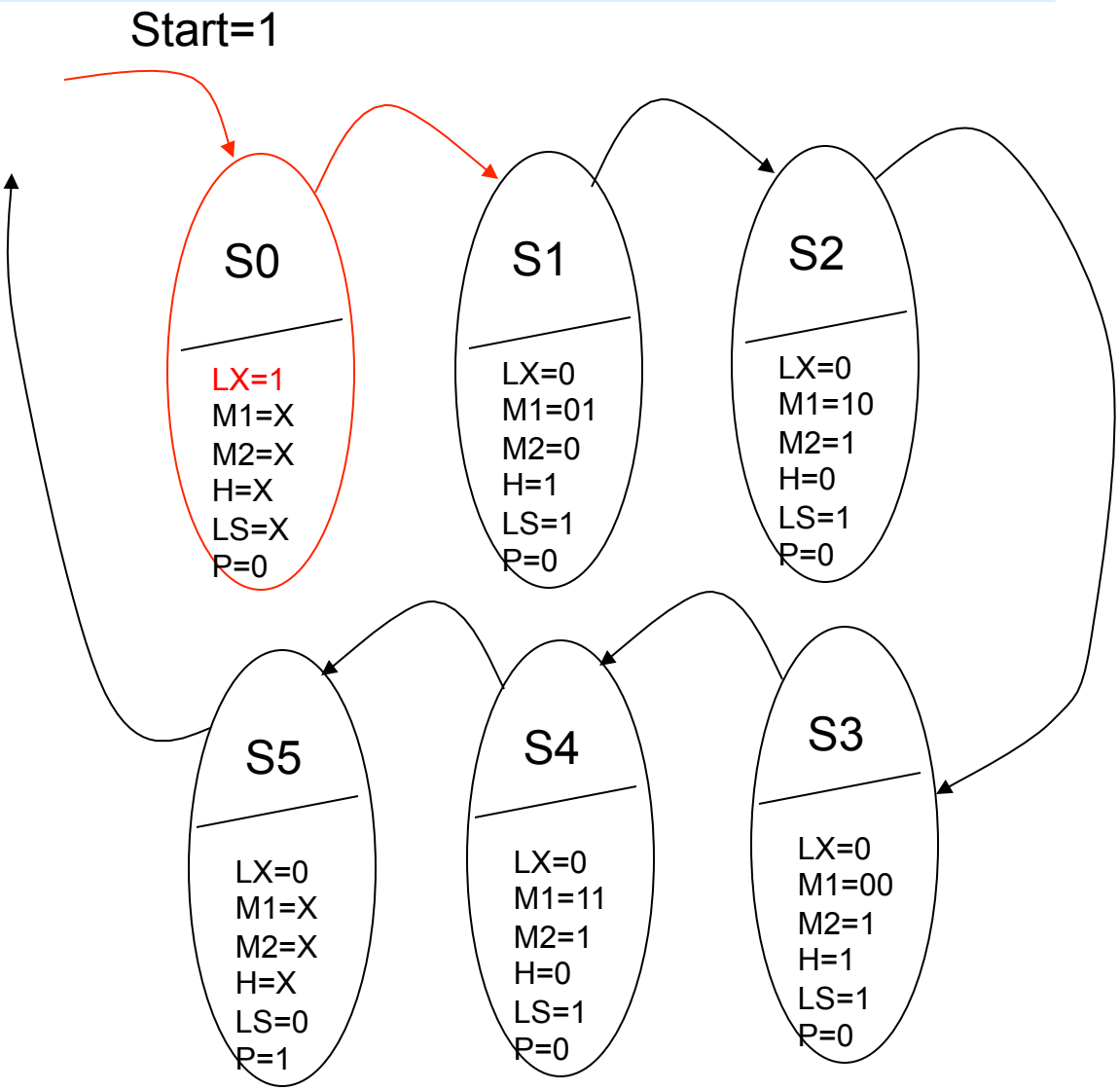
RESET



Exemplo 1

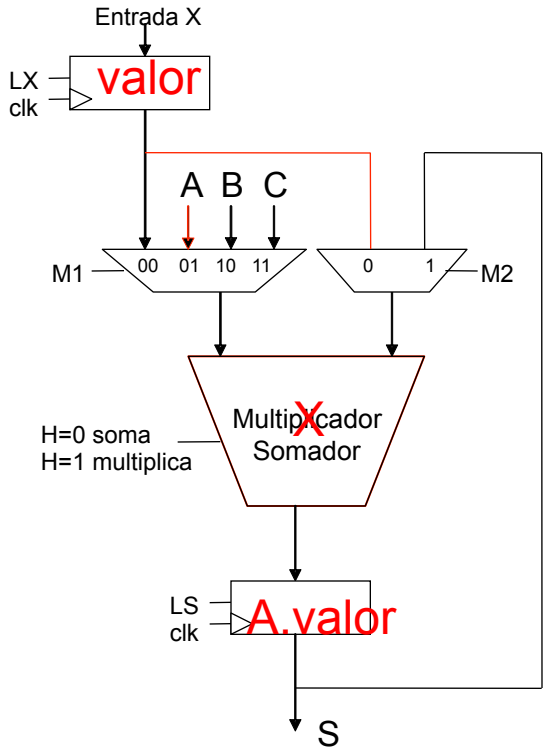
S= X (A.X + B) + C

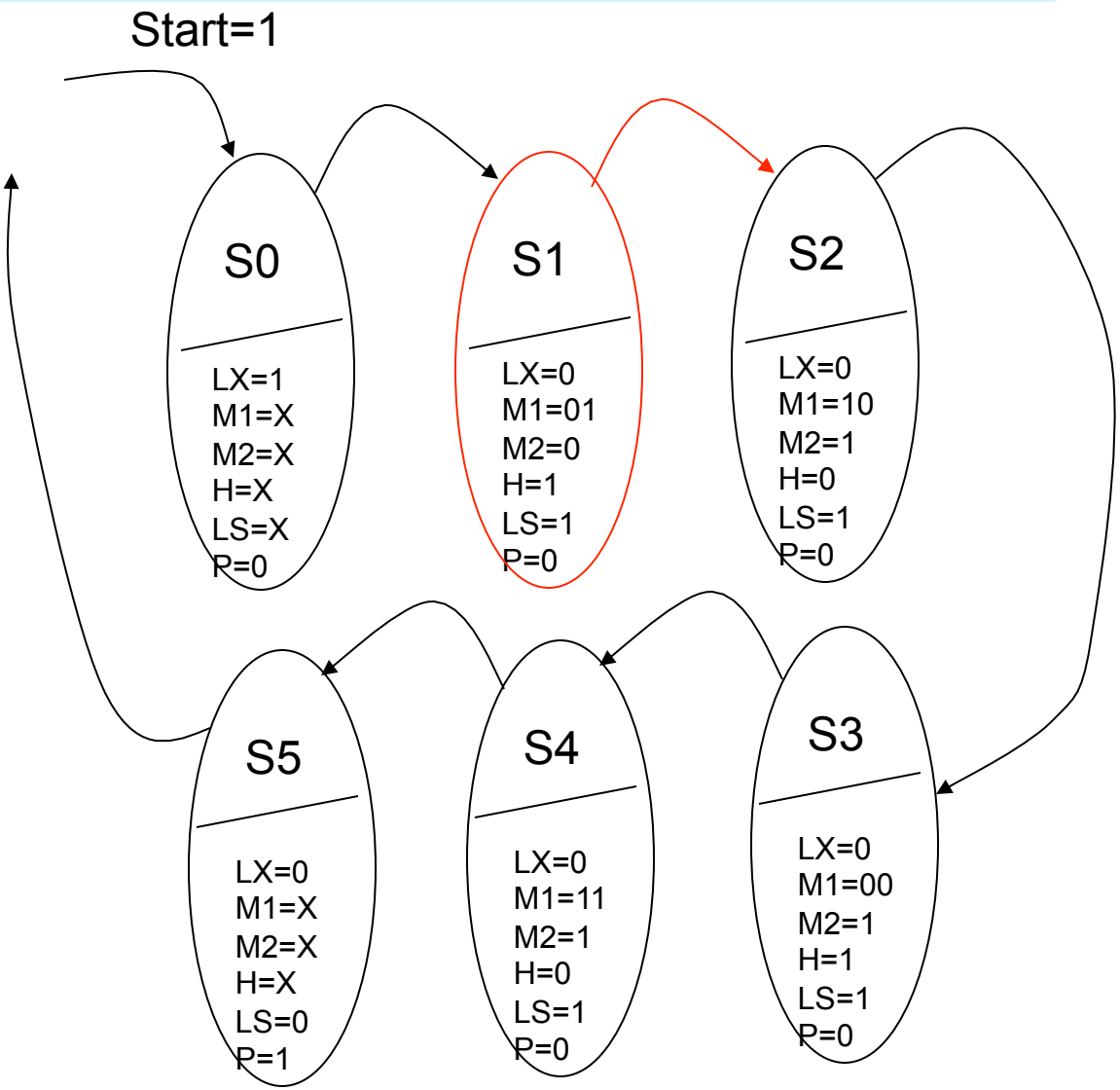




Exemplo 1

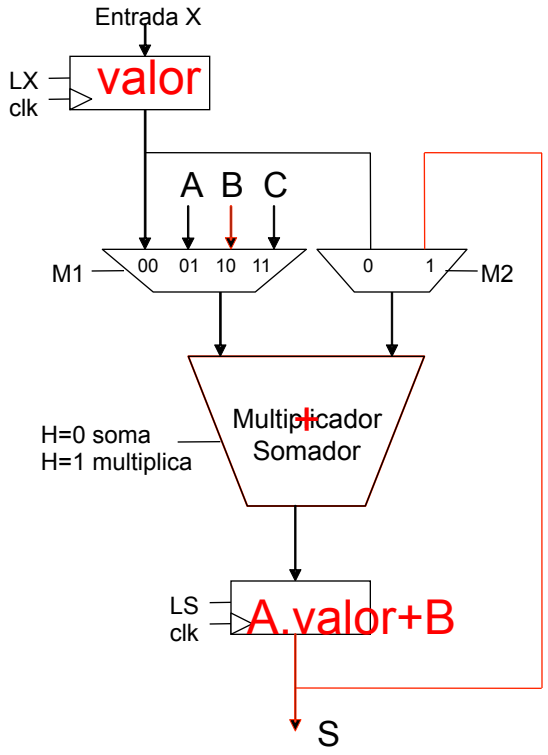
S= X (A.X + B) + C

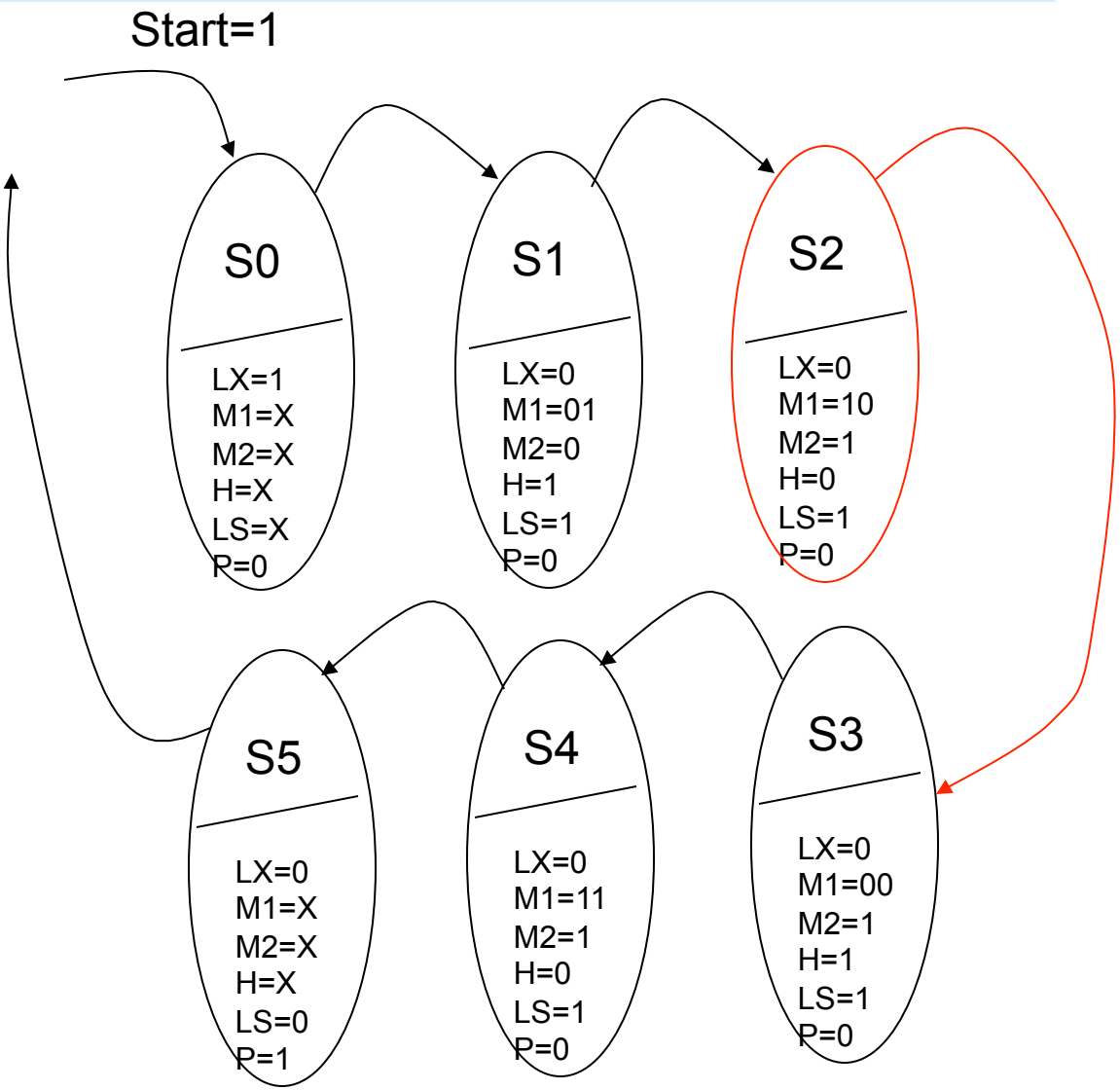




Exemplo 1

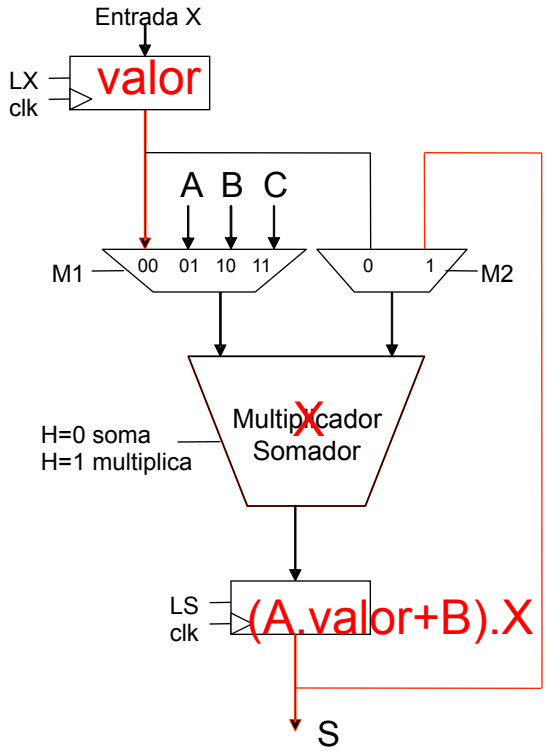
S= X (A.X + B) + C

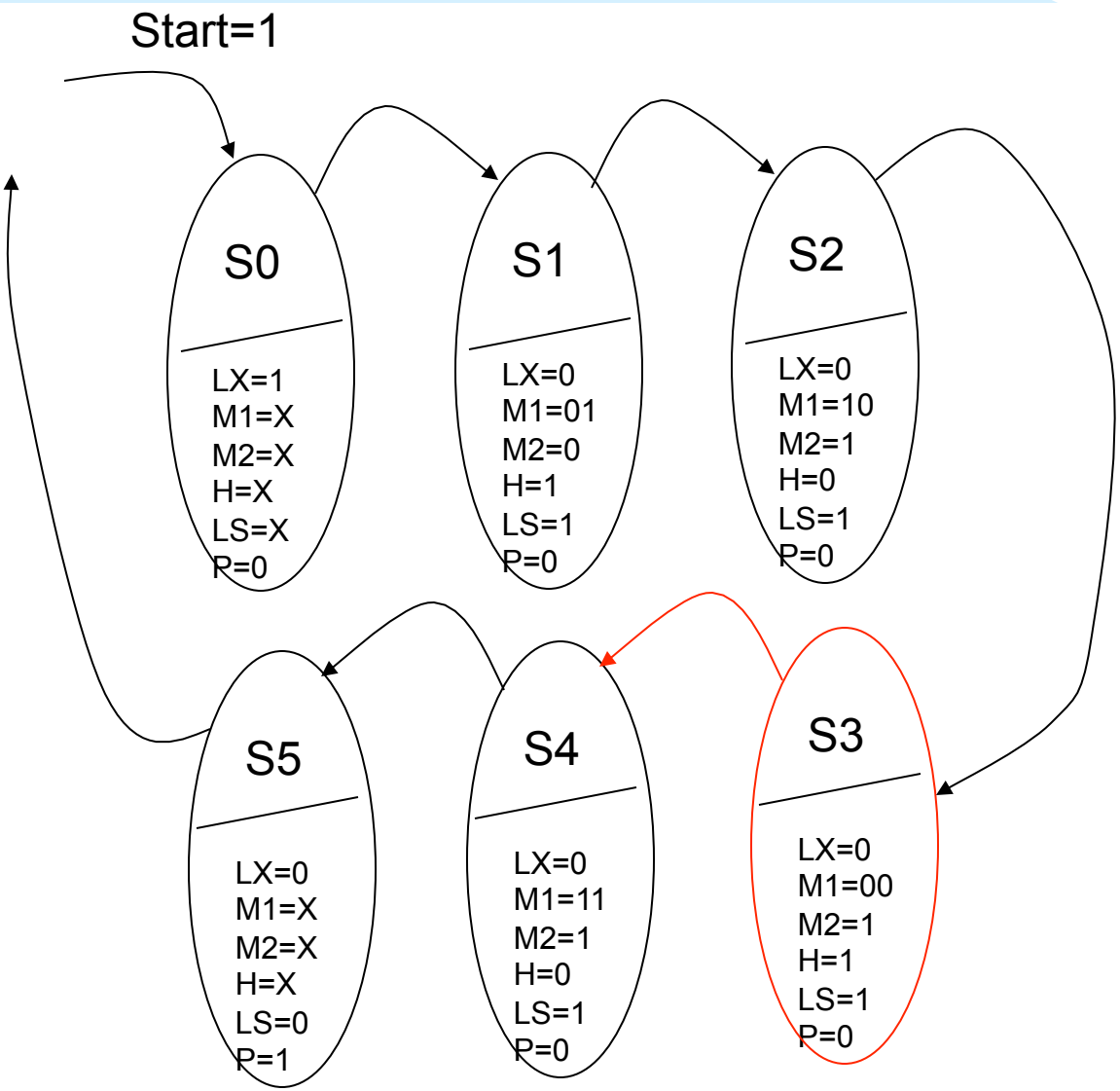




Exemplo 1

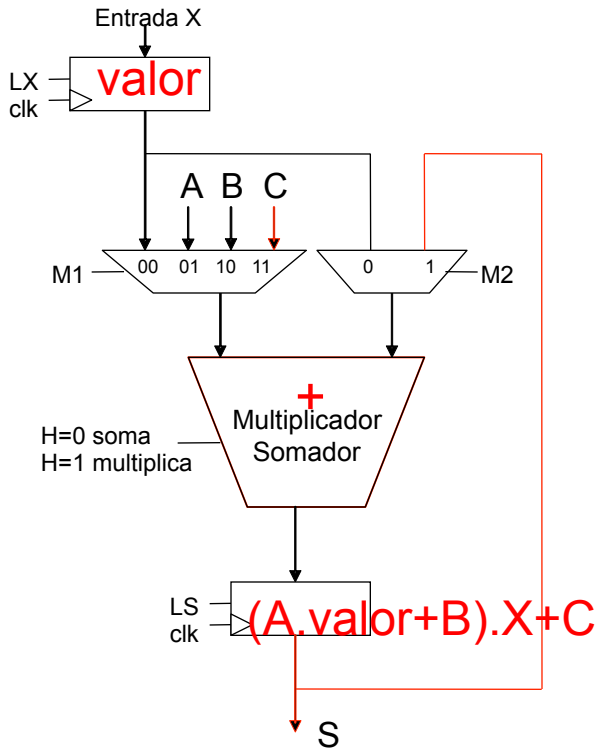
S= X (A.X + B) + C

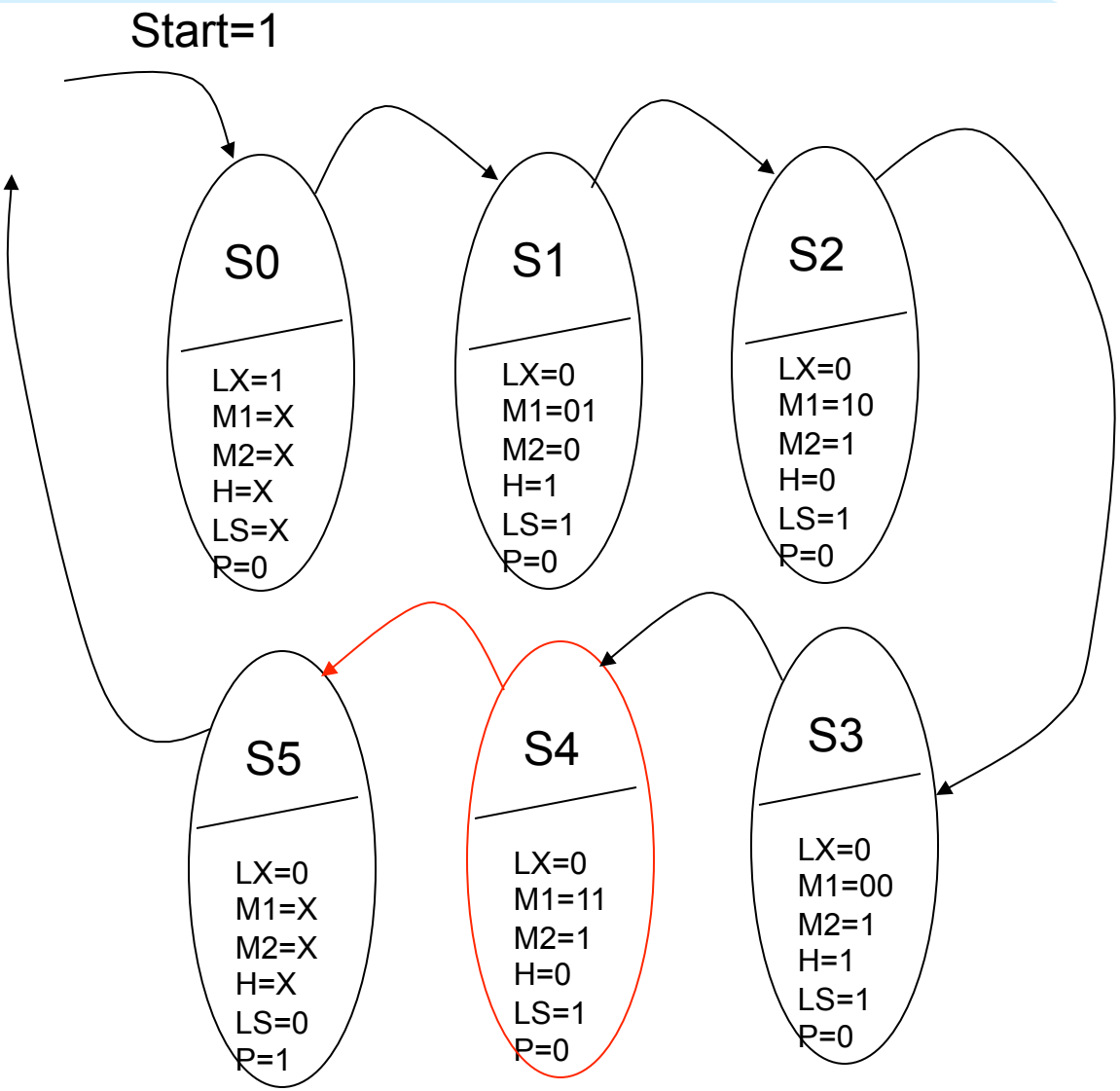




Exemplo 1

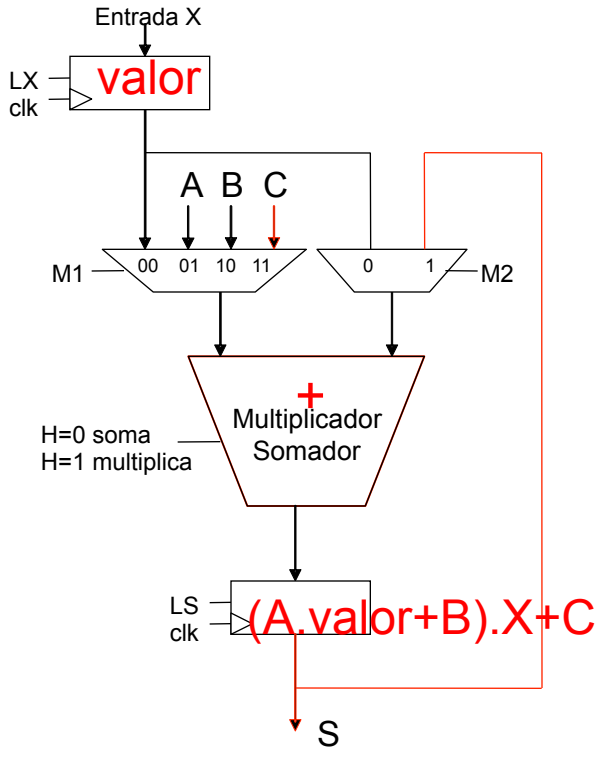
S= X (A.X + B) + C



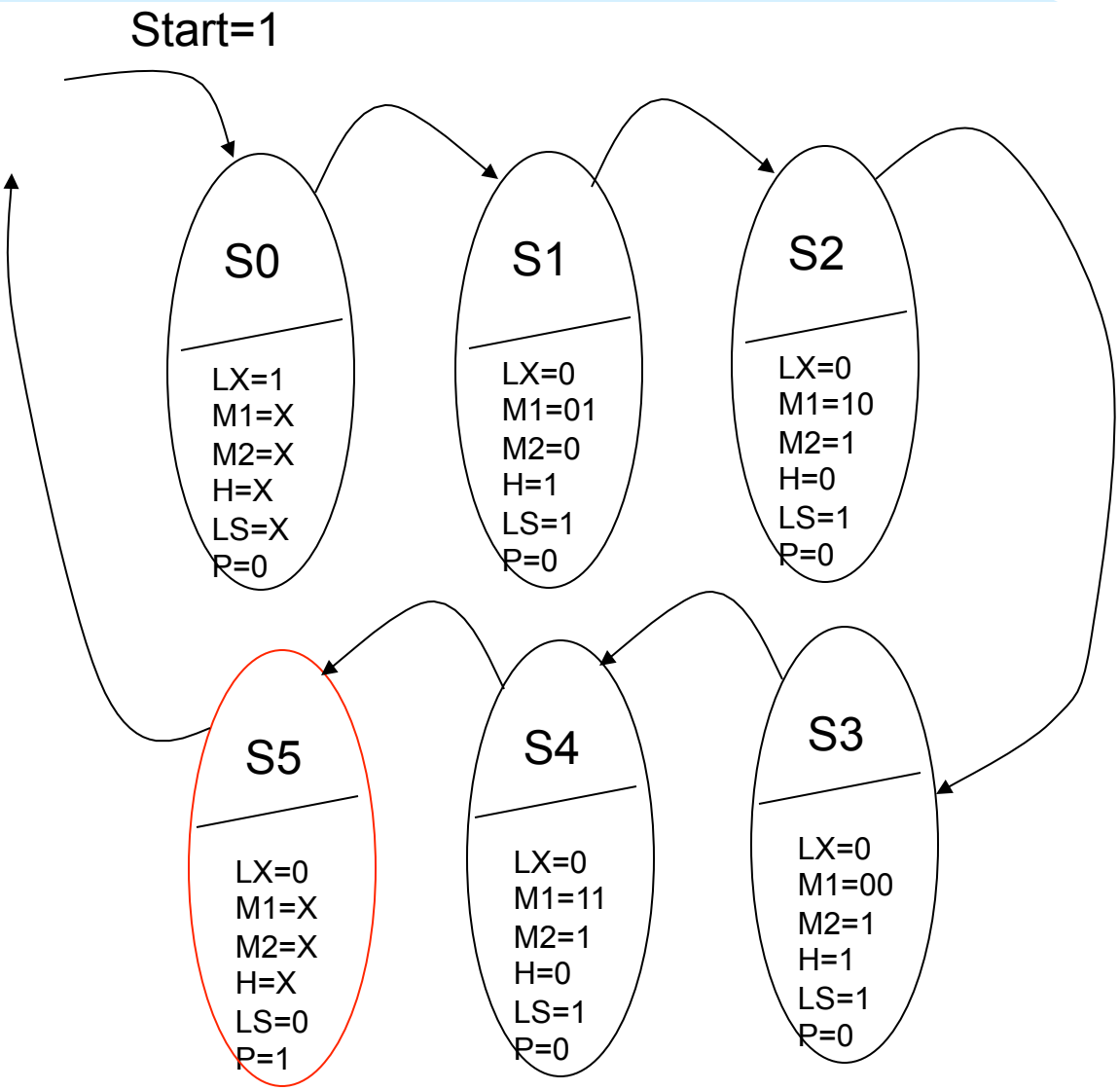


Exemplo 1

S= X (A.X + B) + C



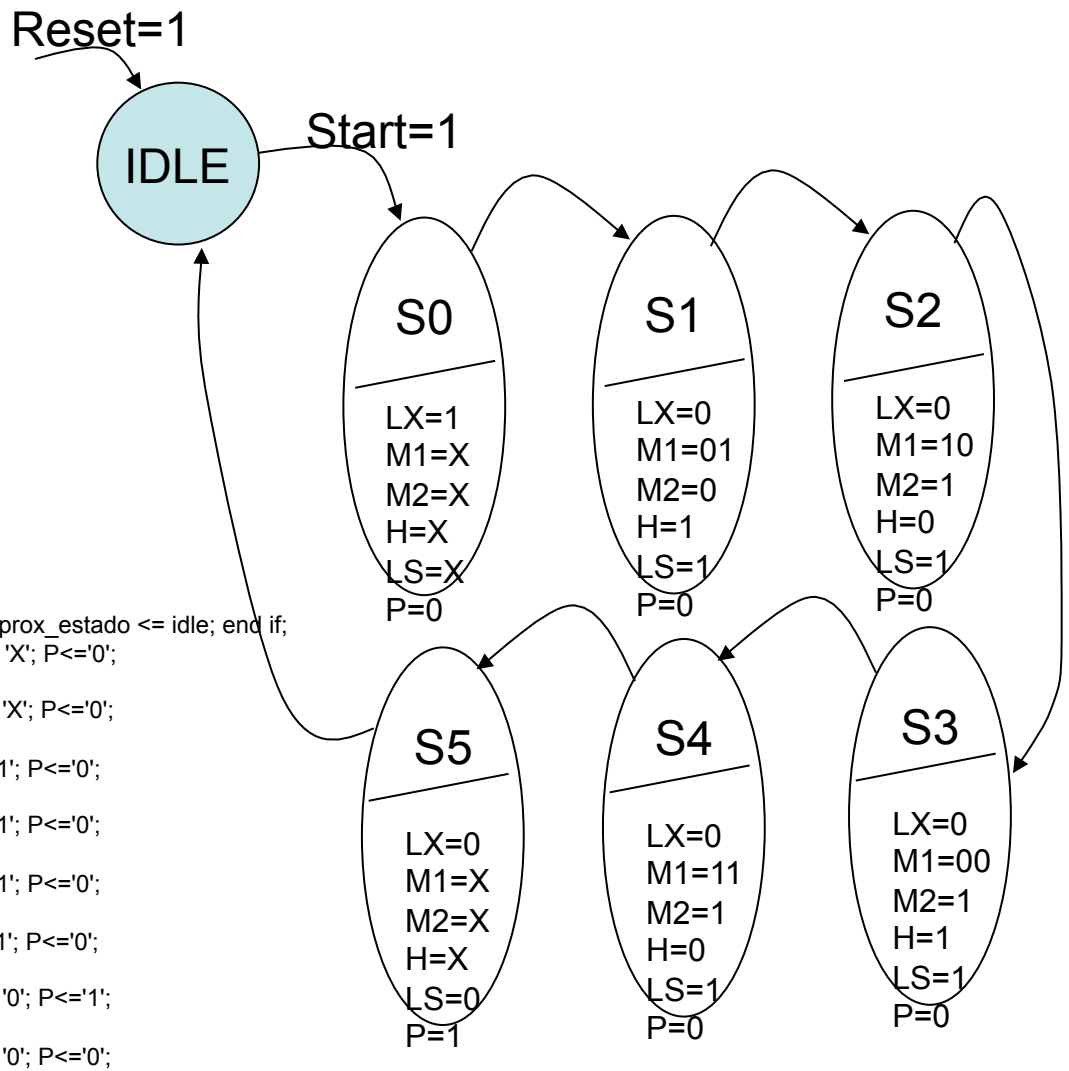
P=1 (FIM)



VHDL: Exemplo 1

RTL – VERSAO SERIAL

```
entity pc_funcao1 is
    Port ( reset : in  STD_LOGIC;
           start : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           LX : out  STD_LOGIC;
           LS : out  STD_LOGIC;
           M1 : out  STD_LOGIC_VECTOR(1 downto 0);
           M2 : out  STD_LOGIC;
           P : out  STD_LOGIC;
           H : out  STD_LOGIC);
end pc_funcao1;


architecture Behavioral of pc_funcao1 is
type tstate is (idle, S0, S1, S2, S3, S4, S5);
signal estado, prox_estado : tstate;
begin

process(reset, clk)
begin
if (reset='1') then
   estado <= idle;
elsif (clk'event and clk='1') then
   estado <= prox_estado;
end if;
end process;
process(estado, start)
begin
CASE estado IS
WHEN idle => if start='1' then prox_estado <= S0; else prox_estado <= idle; end if;
          LX<= 'X'; M1<="XX"; M2 <= 'X'; H <= 'X'; LS <= 'X'; P<='0';
WHEN S0 => prox_estado <= S1;
          LX<= '1'; M1<="XX"; M2 <= 'X'; H <= 'X'; LS <= 'X'; P<='0';
WHEN S1 => prox_estado <= S2;
          LX<= '0'; M1<="01"; M2 <= '0'; H <= '1'; LS <= '1'; P<='0';
WHEN S2 => prox_estado <= S3;
          LX<= '0'; M1<="10"; M2 <= '1'; H <= '0'; LS <= '1'; P<='0';
WHEN S3 => prox_estado <= S4;
          LX<= '0'; M1<="00"; M2 <= '1'; H <= '1'; LS <= '1'; P<='0';
WHEN S4 => prox_estado <= S5;
          LX<= '0'; M1<="11"; M2 <= '1'; H <= '0'; LS <= '1'; P<='0';
WHEN S5 => prox_estado <= idle;
          LX<= '0'; M1<="XX"; M2 <= 'X'; H <= 'X'; LS <= '0'; P<='1';
WHEN others =>  prox_estado <= idle;
          LX<= '0'; M1<="XX"; M2 <= 'X'; H <= 'X'; LS <= '0'; P<='0';
END CASE;
end process;
```



VHDL

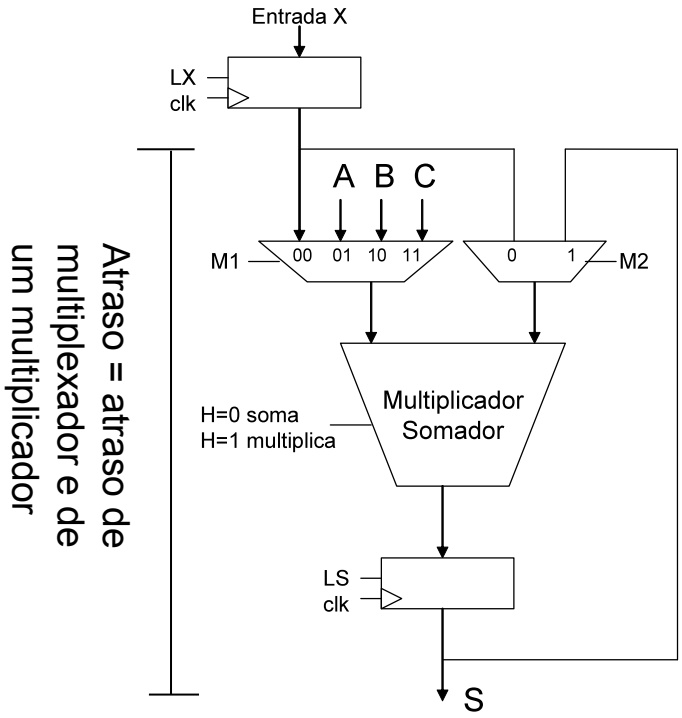
RTL

```
entity PO_funcao1 is
    Port ( reset : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           LX : in  STD_LOGIC;
           M1 : in  STD_LOGIC_VECTOR(1 downto 0);
           M2 : in  STD_LOGIC;
           LS : in  STD_LOGIC;
           H : in  STD_LOGIC;
           dado : in  STD_LOGIC_VECTOR (7 downto 0);
           A : in  STD_LOGIC_VECTOR (7 downto 0);
           B : in  STD_LOGIC_VECTOR (7 downto 0);
           C : in  STD_LOGIC_VECTOR (7 downto 0);
           Saida_funcao : out  STD_LOGIC_VECTOR (15 downto 0));
end PO_funcao1;
```

```
architecture Behavioral of PO_funcao1 is
signal dado_16, A_16, B_16, C_16, ula, mux1, mux2, regx, regs : std_logic_vector(15 downto 0);
begin

process(clk, reset)
begin
if reset='1' then
   regx <= "0000000000000000";
elsif (clk'event and clk='1') then
   if LX ='1' then  regx <= dado_16;
else regx <= regx;
end if; end if;
end process;

process(clk, reset)
begin
if reset='1' then
   regs <= "0000000000000000";
elsif (clk'event and clk='1') then
   if LS ='1' then    regs <= ula;
else regs <= regs;
end if; end if;
end process;

process(mux1, mux2, H)
begin
if H='0' then    ula <= mux1 + mux2;
else     ula <= mux1 * mux2;
end if;
end process;
```

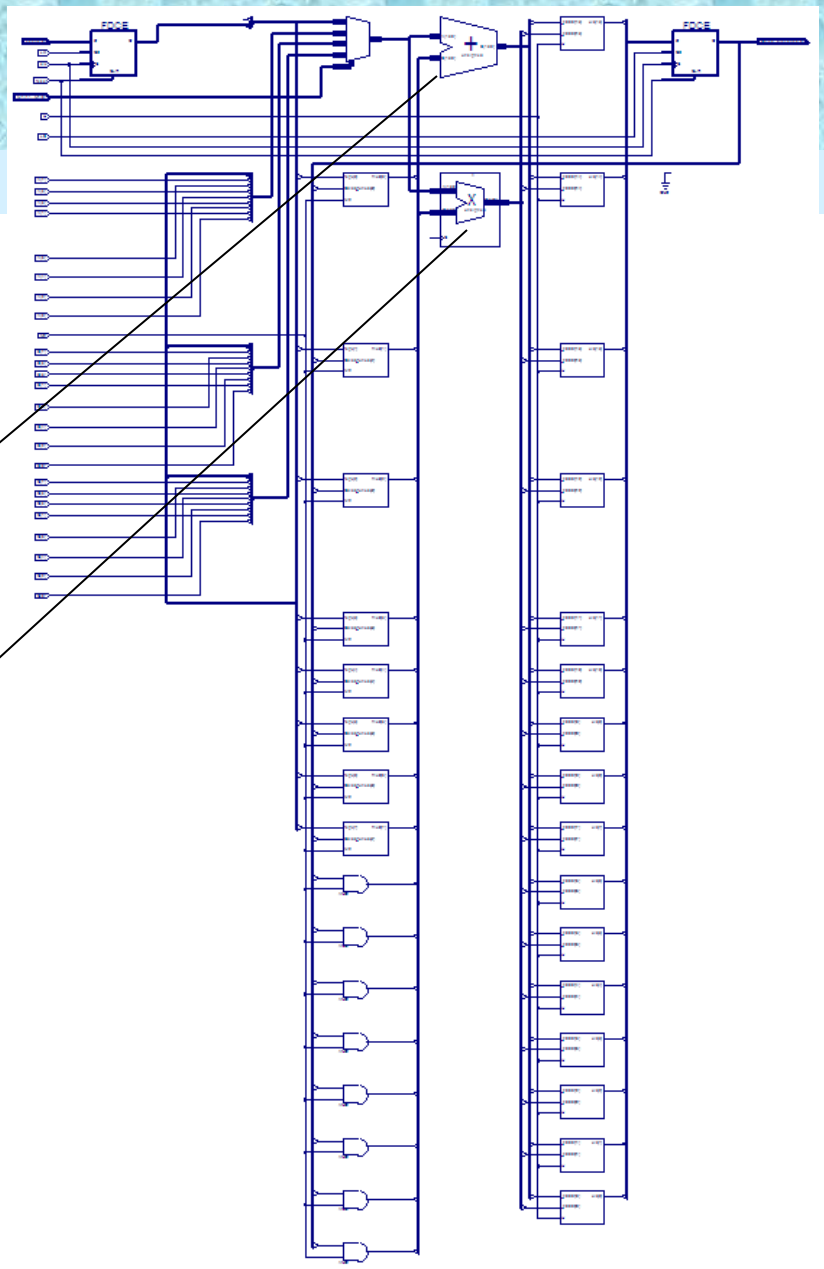
```
process(A_16, B_16, C_16, regx, M1)
begin
CASE M1 IS
WHEN "00" => mux1 <= regx;
WHEN "01" => mux1 <= A_16;
WHEN "10" => mux1 <= B_16;
WHEN others => mux1 <= C_16;
END CASE;
end process;

process(regx, regs, M2)
begin
if M2 = '1' then    mux2 <= regs;
else  mux2 <= regx;
end if;
end process;

saida_funcao <= regs;
A_16 <= "00000000"&A;
B_16 <= "00000000"&B;
C_16 <= "00000000"&C;
dado_16 <= "00000000"&dado;
end Behavioral;
```

Implementação SERIAL



Parte Operativa

XC2v80 (VirtexII)



Alocação de recursos:

Processamento serial

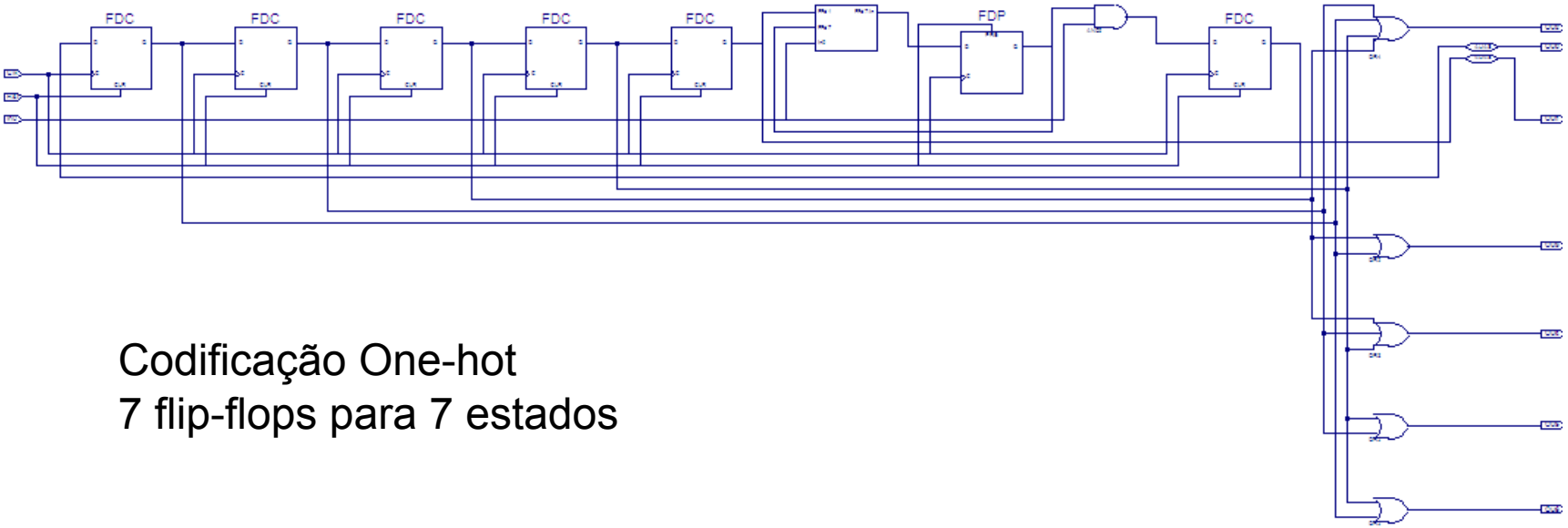
Somador/subtrator

Multiplicador

Parte de Controle

XC2v80 (VirtexII)

Estados: Idle, S0, S1, S2, S3, S4, S5



Codificação One-hot
7 flip-flops para 7 estados

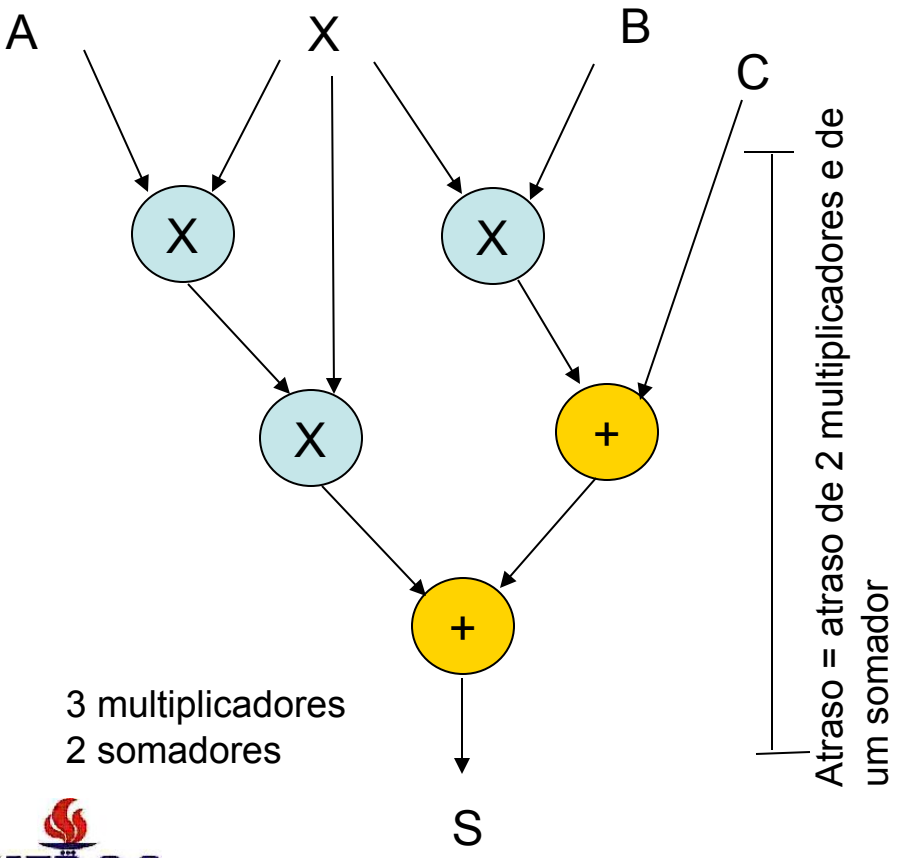
Periodo mínimo de clk = 2ns

Descrição em VHDL: Exemplo 1

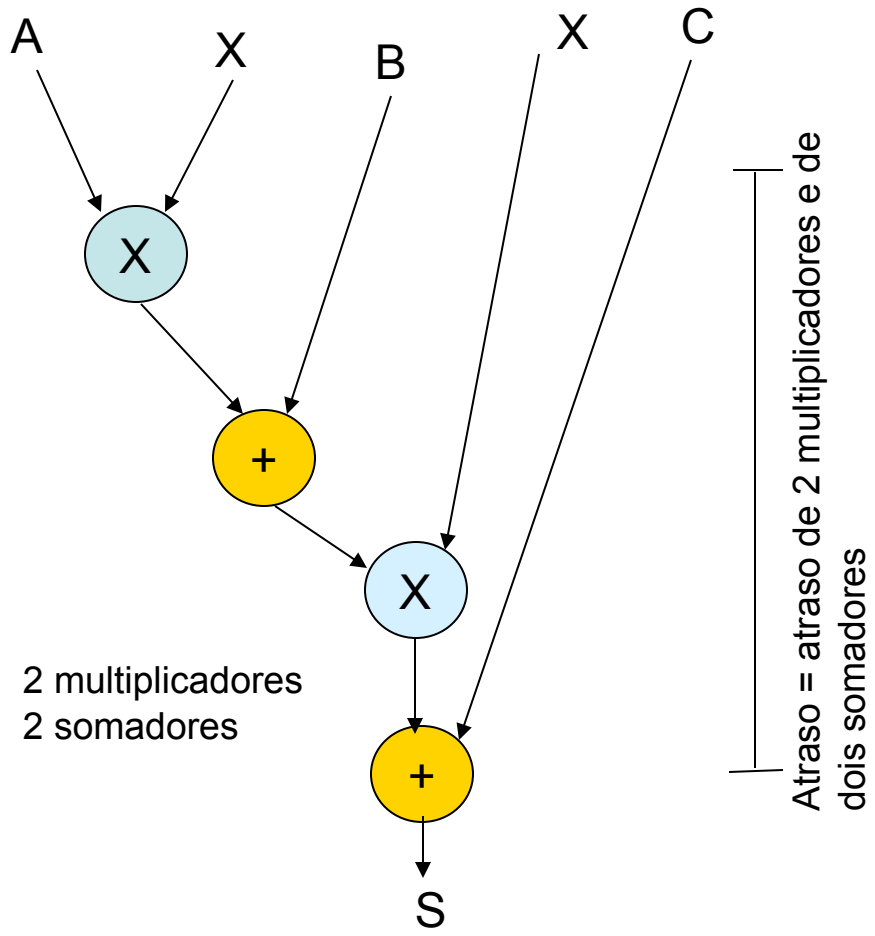
VERSÃO PURAMENTE COMBINACIONAL

- Paralelismo máximo

$S = A.X^2 + B.X + C$

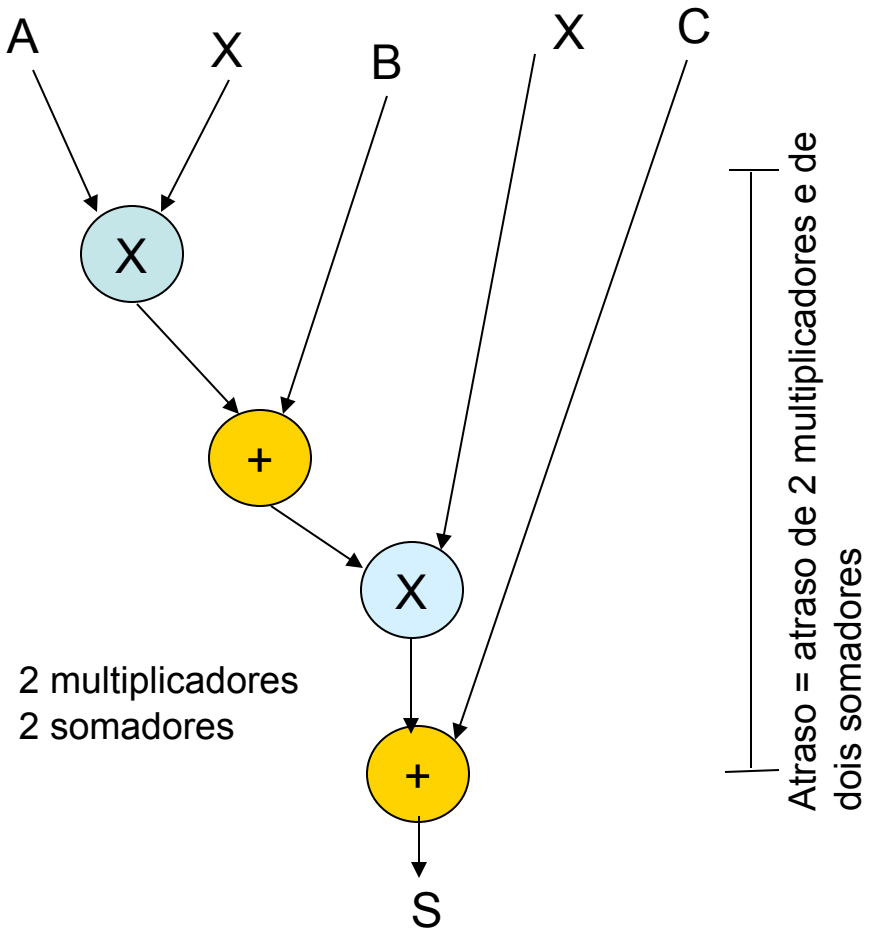


$S = X.(A.X + B) + C$



```
entity funcao1_comb is
    Port ( dado : in STD_LOGIC_VECTOR(7 downto 0);
           clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           start : in STD_LOGIC;
           a : in  STD_LOGIC_VECTOR(7 downto 0);
           b : in  STD_LOGIC_VECTOR(7 downto 0);
           c : in  STD_LOGIC_VECTOR(7 downto 0);
           saida_funcao : out STD_LOGIC_VECTOR(15 downto 0));
end funcao1_comb;

architecture Behavioral of funcao1_comb is

signal dado_16, A_16, B_16, C_16, regx, regs : std_logic_vector(15 downto 0);

begin

saida_funcao <= regs;
A_16 <= "00000000"&A;
B_16 <= "00000000"&B;
C_16 <= "00000000"&C;
dado_16 <= "00000000"&dado;

process(clk, reset)
begin
if reset='1' then
   regx <= "0000000000000000";
elsif (clk'event and clk='1') then
   if start ='1' then
   regx <= dado_16;
 else  regx<=regx;
end if; end if;
end process;

process(clk, reset)
begin
if reset='1' then
   regs <= "0000000000000000";
elsif (clk'event and clk='1') then
   if start='0' then
   regs <= (((A_16 * regx) + B_16)* regx ) + C_16;
else
    regs <= regs;
end if;
end if;
end process;

end Behavioral;
```

$$S = X.(A.X + B) + C$$



Descrição em VHDL: Exemplo 1

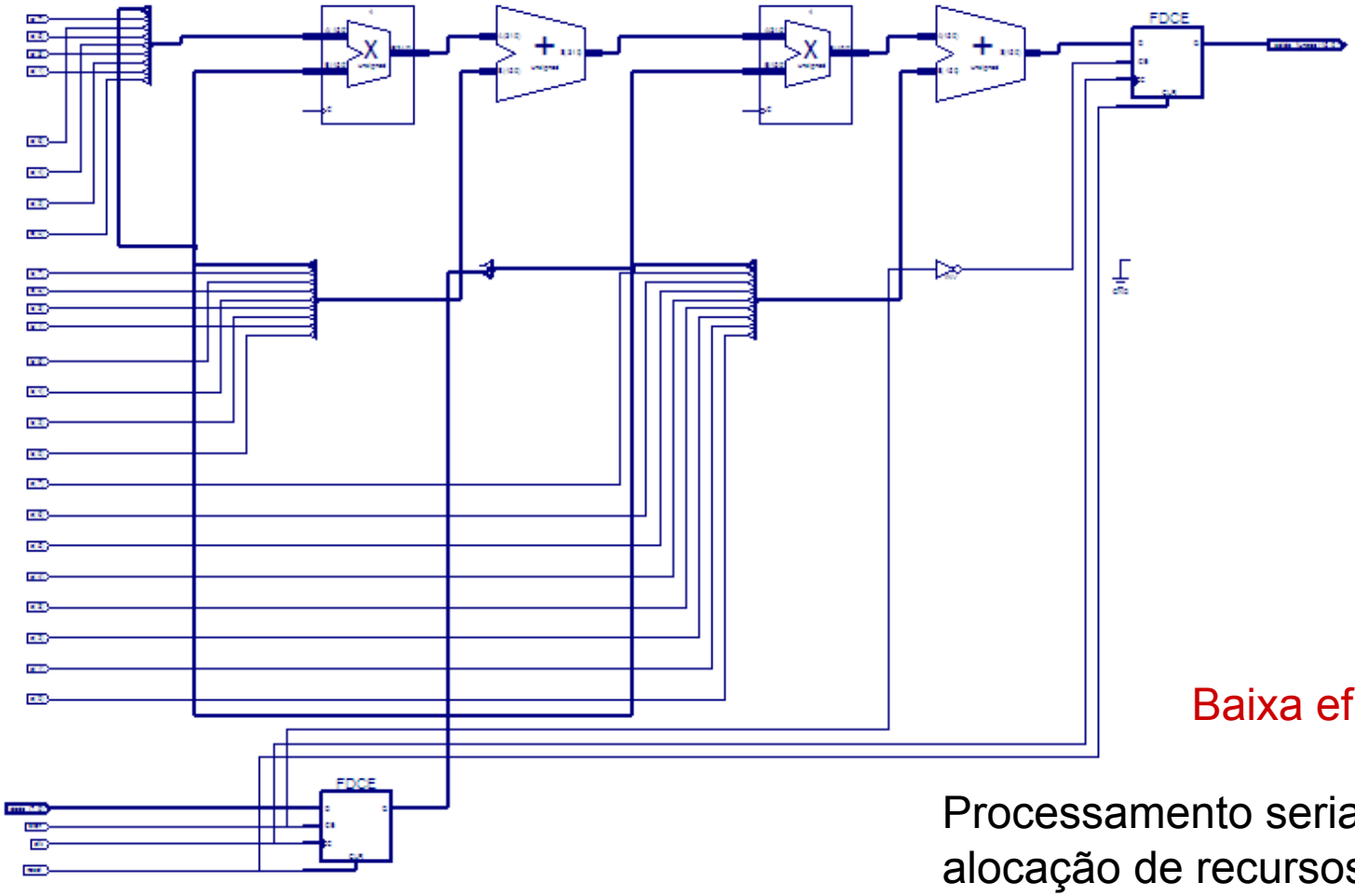
RTL – versão 2

```
entity funcao1_altonivel is
    Port ( reset : in  STD_LOGIC;
           clk : in  STD_LOGIC;
                                                    start : in std_logic;
           dado : in  STD_LOGIC_VECTOR(7 downto 0);
           A, B, C : in  STD_LOGIC_VECTOR(7 downto 0);
           saida_funcao : out  STD_LOGIC_VECTOR(15 downto 0));
end funcao1_altonivel;

architecture Behavioral of funcao1_altonivel is

signal dado_16, A_16, B_16, C_16, regx, regs : std_logic_vector(15 downto 0);
signal cont : std_logic_vector(1 downto 0);
begin


saida_funcao <= regs;
A_16 <= "00000000"&A;
B_16 <= "00000000"&B;
C_16 <= "00000000"&C;
dado_16 <= "00000000"&dado;

process(clk, reset)
begin
if reset='1' then
   regx <= "0000000000000000";
elsif (clk'event and clk='1') then
   if start ='1' then
   regx <= dado_16;
                        else
                        regx<=regx;
                        end if;
end if;
end process;
```

```
process(reset, clk)
begin
if (reset='1' or start='1') then
  cont <= "00";
elsif clk'event and clk='1' then
 if start='0' then
  cont <= cont +1;
  end if;
end if;
end process;

process(clk, reset)
begin
if reset='1' then
   regs <= "0000000000000000";
elsif (clk'event and clk='1') then
   if start='0' then
                        CASE CONT IS
                         WHEN "01" => regs <=  A_16 * regx;
                         WHEN "10" =>  regs <= regs + B_16;
                         WHEN  "11" => regs <= regs * regx;
                         WHEN others => regs <= regs + C_16;
                        END CASE;
else
regs <= regs;
end if;
end if;
end process;

end Behavioral;
```

Descrição em VHDL: Exemplo 1

RTL – versão 2



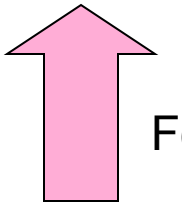
Baixa eficiência

Processamento serial mas alocação de recursos paralelo

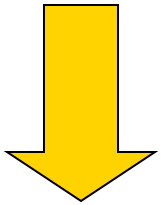
Comparação de área e desempenho

XC2v80 (VirtexII)

Versão RTL (serial)
PC => 7 LUTS e 7 Flip-flops
PO => 78 LUTs e 24 flip-flops e 1 MULT18x18
=> periodo mínimo 6.8ns



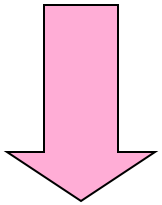
Feq.



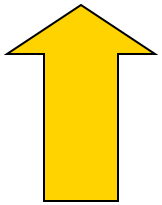
Resposta lenta

6x 6.8ns=40ns

Versão COMB e paralela
=> 30 LUTS, 24 flip-flops e 2 MULT18x18
=> periodo mínimo 12.4 ns



Freq.



Resposta rápida

12.4ns

Devido ao caminho crítico
E ao paralelismo x serial

Exemplo 2

Implementar um hardware para realizar as seguintes operações:

$F(x) = (A.\ x^2 + B)/4 + C$

Máximo desempenho
(comprometimento em área)

```
inicio: X <= novo valor
        Start =1;
        Wait until done=1
       go to inicio;
```

Mínima área
(comprometimento em desempenho)