

Um componente de um sistema sendo desenvolvido na *JKP3* deve operar com um clock de 1MHz. Entretanto, o clock desse sistema é de 160MHz. Para gerar o clock do componente (1MHz) a partir do clock do sistema (160MHz) -- ou seja, dividir a frequência --, você dispõe de contadores de 4 bits, descritos abaixo:

```
-----
library IEEE;
use IEEE.numeric_bit.all;

entity counter is
  port(
    clk, load, en:  in bit;
    P:              in bit_vector(3 downto 0);
    Q:              out bit_vector(3 downto 0);
    rco:            out bit
  );
end counter;

architecture behavioral of counter is
  signal count: unsigned(3 downto 0);
begin
  process(clk)
  begin
    if clk'event and (clk='1') and (en='1') then
      if load='1' then
        count <= unsigned(P);
      else
        count <= count+1;
      end if;
    end if;
  end process;
  rco <= '1' when (count=15) else '0';
  Q <= bit_vector (count);
end behavioral;
-----
```

Utilizando dois destes contadores, é possível construir um divisor que, recebendo na entrada um clock de 160MHz, gera na saída um clock de 1MHz (não necessariamente uma onda quadrada). Complete a descrição da arquitetura do divisor abaixo para obter esse comportamento:

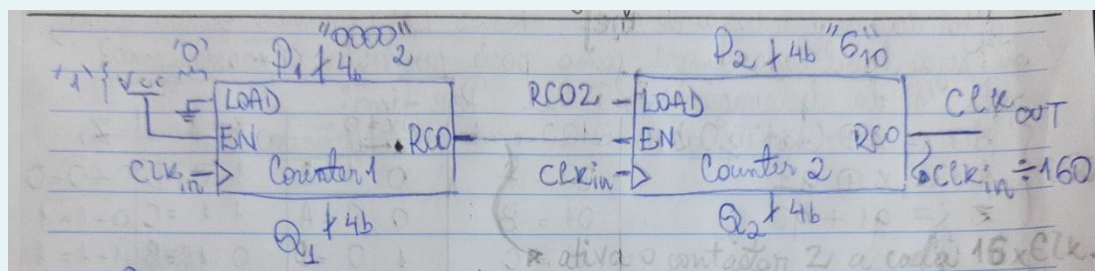
```
-----
entity divisor is
  port(
    clk_in:  in bit;
    clk_out: out bit
  );
end divisor;

architecture estrutural of divisor is
  component counter is
    port(
      clk, load, en:  in bit;
      P:              in bit_vector(3 downto 0);
      Q:              out bit_vector(3 downto 0);
      rco:            out bit
    );
  end component;

  signal P1,P2,Q1,Q2: bit_vector(3 downto 0);
  signal load1, load2, rco1, rco2, en1, en2: bit;
begin
  counter1: counter port map(clk_in, load1, en1, P1, Q1, rco1);
  counter2: counter port map(clk_in, load2, en2, P2, Q2, rco2);
  en1 <= '1';
  load1 <= '0';
  P1 <= "0000";

  en2 <= rco1 & "✓" ; --complete
  load2 <= rco2 & "✓" ; -- complete
  P2 <= "0110" & "✓" ; --complete
  clk_out <= rco2 & "✓" ; --complete

end estrutural;
```



Como temos 2 contadores, cascatear nos permite multiplicar o limite de contagem. Como o RCO só ativa a cada 16 bordas de subida do clock, precisaremos de uma contadora de módulo 10 ($10 \times 16 = 160$).

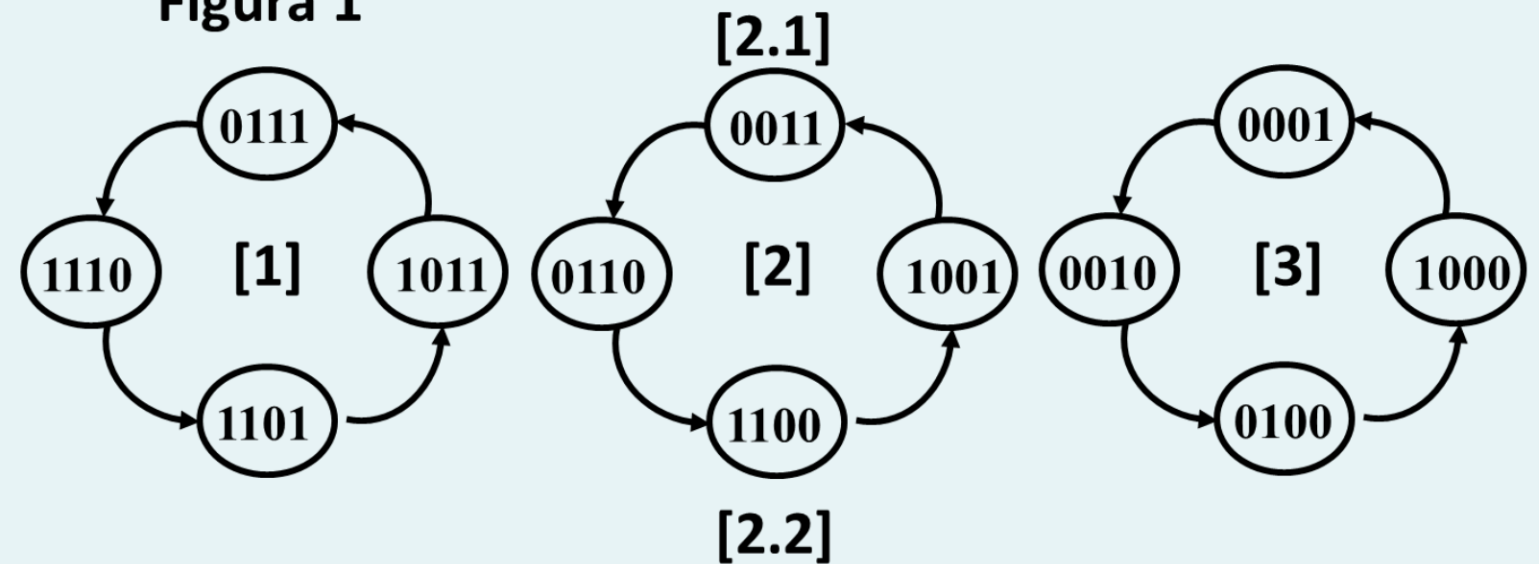
Perceba que associar o clock-out a um dos Q_2 's nos impede de ter um módulo diferente de uma ~~potência de 2~~ potência de 2 (0, 1, 2, 4 e 8). Usamos então o RCO2.

Perceba que o RCO2 ativa apenas se $Q_2 = "1111"_2 = "15"_{10}$ ou seja, módulo 16, precisando de um offset:

$$\text{Módulo } 10 = \text{Módulo } 16 - \text{offset} \Rightarrow \text{offset} = 16 - 10 = 6 = "0110"_{10}$$

Sabe-se que um Registrador Deslocador na configuração *Deslocador Em Anel* é adequado para aplicações de controle por proporcionar facilidade de projeto e a geração de tipos de forma de onda convenientes. Por outro, lado apresenta a deficiência de não ser robusto em alguns casos. Este último problema deve-se ao fato de que existem diferentes seqüências principais de transição de estados, quais sejam, seqüências [1], [2] e [3], de máximo comprimento (Figura 1). Ao adotar uma seqüência principal, se ocorrer um erro devido a um ruído espúrio, por exemplo, o controlador pode sair desta seqüência e não voltar mais a ela. Devem ser buscadas soluções para devolver o controlador para seguir a seqüência de estados para a qual foi projetado.

Figura 1



Considere a seqüência [2] e em particular os Estados [2.1] e [2.2]. Imagine que esta seqüência é gerada com deslocamento à esquerda. Na coluna correspondente assinale as alternativas de quais são os Estados que necessitam de UMA, de DUAS ou de TRÊS bordas de subida do clock, para retornar ao Estado [2.1] ou para retornar ao Estado [2.2], da Figura 1.

UMA borda de subida do clock	DUAS bordas de subida do clock	TRÊS bordas de subida do clock
<input checked="" type="checkbox"/> 1110 ✓ <input type="checkbox"/> 1111 <input type="checkbox"/> 1011 <input type="checkbox"/> 1000 <input type="checkbox"/> 0000 <input type="checkbox"/> 0100 <input type="checkbox"/> Nenhum <input checked="" type="checkbox"/> 0001 ✓ <input type="checkbox"/> 0111	<input type="checkbox"/> 0100 <input type="checkbox"/> Nenhum <input type="checkbox"/> 1011 <input checked="" type="checkbox"/> 0000 ✓ <input checked="" type="checkbox"/> 0111 ✓ <input type="checkbox"/> 1110 <input checked="" type="checkbox"/> 1000 ✓ <input type="checkbox"/> 0001 <input checked="" type="checkbox"/> 1111 ✓	<input type="checkbox"/> 1110 <input type="checkbox"/> 0111 <input type="checkbox"/> 1011 <input type="checkbox"/> 0000 <input type="checkbox"/> 0001 <input type="checkbox"/> 1000 <input type="checkbox"/> 1111 <input checked="" type="checkbox"/> Nenhum ✓ <input type="checkbox"/> 0100
Atingiu 25,00 de 25,00 A resposta correta é: • 0001 • 1110	Atingiu 25,00 de 25,00 A resposta correta é: • 0000 • 1000 • 0111 • 1111	Atingiu 20,00 de 20,00 A resposta correta é: Nenhum

Sabe-se que uma maneira simples de se realizar a criptografia de palavras de dados, utilizando-se LFSRs, é aquela mostrada na Figura 2.

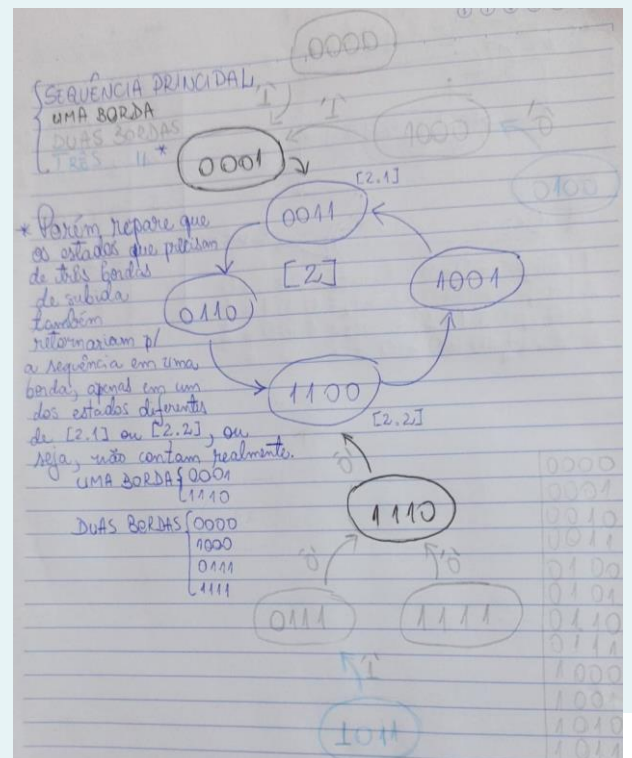
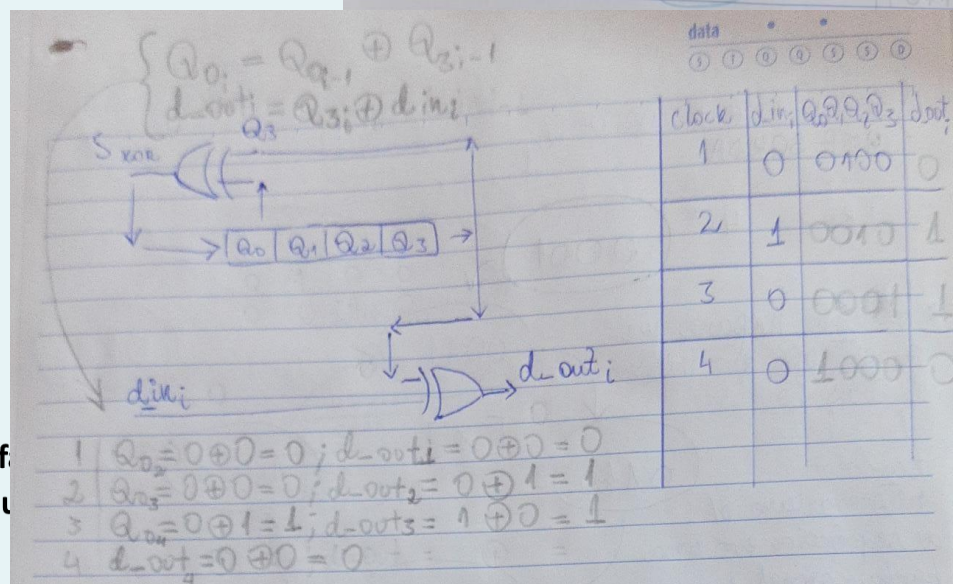
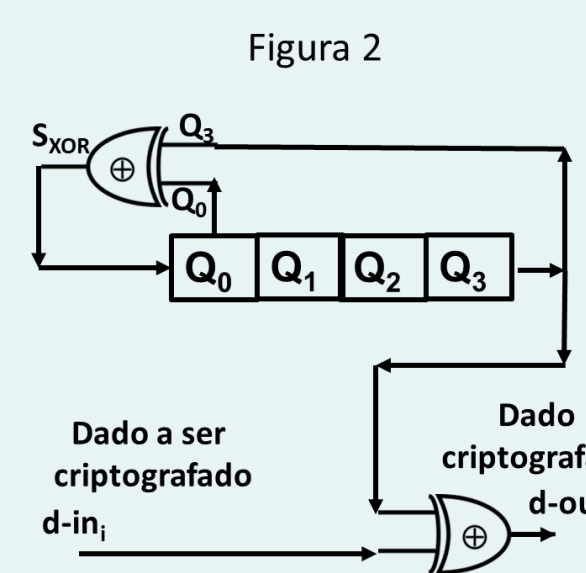
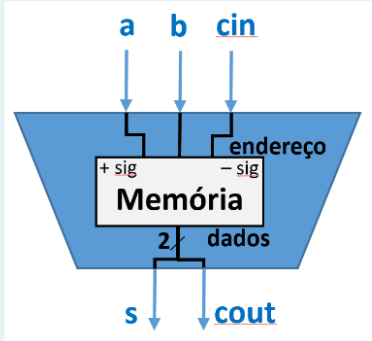


Figura 2



Considere uma palavra de 4 bits, <d-in₀ : d-in₃>, fornecida sequencialmente, bit a bit, em 4 bordas de subida do clock, sucessivas. Assuma ademais que, na primeira borda de subida do clock, um estado inicial pré-definido é carregado no LSFR da Figura 1 (ver Tabela 1). Pede-se que seja completada a Tabela 1, com os valores dos estados seguintes da LFSR e os respectivos valores dos dados criptografados <d-out>, correspondentes.

Deseja-se construir um circuito digital combinatório cuja interface é aquela mostrada abaixo: trata-se de um circuito somador completo de 1 bit, cujas entradas são **a**, **b** e **cin** (todos de 1 bit), dando como saída o resultado da soma, **s** (1 bit) e o vai-um **c** (1 bit). Para a construção desse circuito, dispõe-se exclusivamente de um dispositivo de memória com 3 bits de endereço e 2 bits de dados (conforme mostrado no interior da figura).

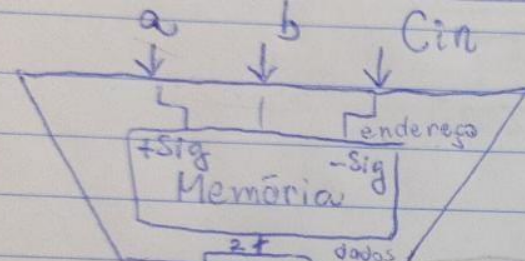


Nesse cenário, é possível construir o circuito pedido usando apenas esse dispositivo de memória? Em caso positivo, preencha a tabela abaixo com o conteúdo que deve ser carregado nessa memória (não é preciso justificar). Em caso negativo, use a caixa de texto para justificar a razão para essa impossibilidade (não é preciso preencher a tabela).

endereço	dados
000	<div>0 ✓</div> <div>0 ✓</div>
001	<div>1 ✓</div> <div>0 ✓</div>
010	<div>1 ✓</div> <div>0 ✓</div>
011	<div>0 ✓</div> <div>1 ✓</div>
100	<div>1 ✓</div> <div>0 ✓</div>
101	<div>0 ✓</div> <div>1 ✓</div>
110	<div>0 ✓</div> <div>1 ✓</div>
111	<div>1 ✓</div> <div>1 ✓</div>

Justificativa: ✓

ENDEREÇO			DADOS	
0	000	0	00	
1	001	2	10	
2	010	2	10	
3	011	1	01	
4	100	2	10	
5	101	1	01	
6	110	1	01	
7	111	3	11	
$a \quad b \quad cin$			$s \quad cout$	



dados: $(d_1, d_0) = (s, cout) =$
 $(a \oplus b \oplus cin, a \cdot cin + b \cdot cin + a \cdot b)$

Mas é possível? SIM

Para cada endereço (a, b, cin) de entrada, podemos definir uma via de dados no par ordenado $(s, cout)$.

ENDEREÇO	DADOS
----------	-------