Na JKP^3, um circuito sequencial síncrono, chamado de **mef**, foi projetado de maneira estrutural, utilizando como componente flip-flops do tipo D:

```
--------------------------------
entity mef is

  port(
    x, reset, clock: in  bit;
    z:               out bit
  );
end mef;

architecture estrutural of mef i
component flipflopd IS
  port(
    D, reset, clock, EN: in  bit
    Q:                   out bit
  );
end component;
signal q1, q0, d1, d0: bit;

begin

    FF0: flipflopd port map(d0,
    FF1: flipflopd port map(d1,
    d0 <= x xor (not q0);
    d1 <= x xor q1;
    z <= q1 or q0;

end estrutural;

--------------------------------
```

Os flip-flops do tipo D são do lote utilizado no projeto Jokempô++, com a descrição em VHDL abaixo:

```
--------------------------------
entity flipflopd IS
  port(
    D, reset, clock, EN: in  bit
    Q:                   out bit
  );
end flipflopd;

architecture behavior of flipflo
begin
  process (reset, clock)
  begin
    if reset='0' then
      Q <= '0';
    elsif clock'EVENT and clock=
      Q <= D;
    end if;
  end process ;
end behavior;

------
```

Agora a JKP^3 já dispõe de outras maneiras de implementar circuitos sequenciais, sem necessariamente usar este flip-flop. Então, sua tarefa é descrever de modo comportamental o circuito sequencial síncrono **mef**, numa arquitetura que leve a exatamente o mesmo comportamento da arquitetura estrutural acima. Para isso, complete os trechos de código abaixo, considerando a seguinte designação de estados: A para q1q0=00, B para q1q0=01, C para q1q0=11 e D para q1q0=10 .

Na arquitetura de flipflop, temos um RESET* ativo em '0', e escrevemos o valor de D em Q quando clock varia de '0' p/ '1'.
Já da arquitetura de **mef**, temos EN (enable) sempre ativo, ou seja, habilitado sempre, como pode ser visto pelo '1' da declaração implícita. Por fim:

{ d0 <= x ⊕ (not q0)
  d1 <= x ⊕ q1
  z <= q1 + q0

(* assíncrono)

Q1Q0 = estado	x=0 d1 : d0	x=1 d1 : d0	z
00 = A	0 : 1 = B	1 : 0 = D	0+0=0
01 = B	0 : 0 = A	1 : 1 = C	0+1=1
11 = C	1 : 0 = D	0 : 1 = B	1+1=1
10 = D	1 : 1 = C	0 : 0 = A	1+0=1
	atual	vai para	

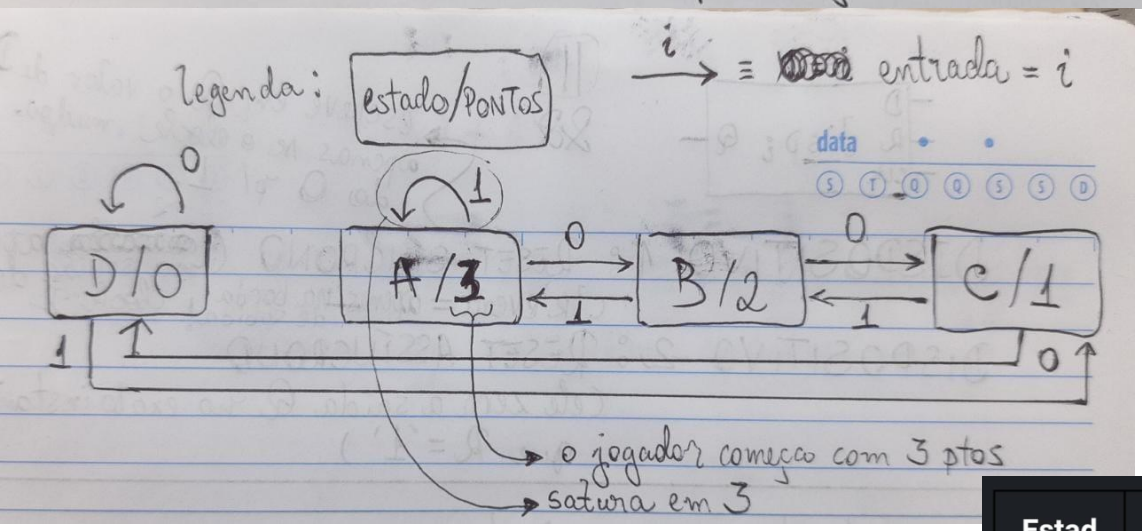
Se RESET = '0', temos Q1Q0="00", ou, estado A.

O estado A vai para A com X=1 e atual=D ; X=0 e atual=B
" " " " B com X=1 e " =C ; X=0 e " =A
" " " " C com " e " =B ; " e " =D
" " " " D com " e " =A ; " e " =C

Deseja-se construir um circuito digital sequencial síncrono para controlar a pontuação de um jogo de simulação de instrumentos: o jogador começa com uma pontuação de 3; toda vez que ele erra uma nota (entrada = '0'), ele perde 1 ponto, até o limite inferior de 0; toda vez que ele acerta uma nota (entrada = '1'), sua pontuação é incrementada de 1, até o limite superior de 3; se o jogador chegar a 0 pontos, o circuito emite '1' como saída, indicando para ativar um circuito externo que emite o som de vaias. O exemplo abaixo ilustra o comportamento desejado (onde "R" representa "reset"):

Entrada (1 bit):	R000001001101010011111101100000110011000
Saída (1 bit):	000111011000000010000000000011100010011 1
Pontos:	321000100121212101233332332100012101 2100

Como D são as vaias, sua saída é a única = '1'.
Se adotarmos A como inicial, ele precisa transitar entre os demais, A → B → C quando x=0 (necessariamente, vide tabela fornecida)
Com x=1, fazemos o oposto, "aumentando os pontos", porém saturando no último (pontuação máx. = 3)



Estado Atual	Próximo Estado		Saída
	0	1	
A	B	A ✓	0 ✓
B	C ✓	A ✓	0 ✓
C	D ✓	B ✓	0 ✓
D	D ✓	C ✓	1 ✓

Problema:

É fornecida uma cadeia de bits **x** de entrada. Deve-se obter uma saída **z** que forneça **dois bits zero** em sequência, e posteriormente, reproduza a cadeia **x**, tal como foi recebida.

Exemplo:

x = 1101010011000111...

z = **00**1101010011000111...

Tem-se que realizar a síntese da FSM que resolve este problema, a partir de sua Tabela de Transição de Estados/Saída (Figura1). Na Figura aparece a determinação do código binário que deve ser adotado para representar cada um dos Estados.

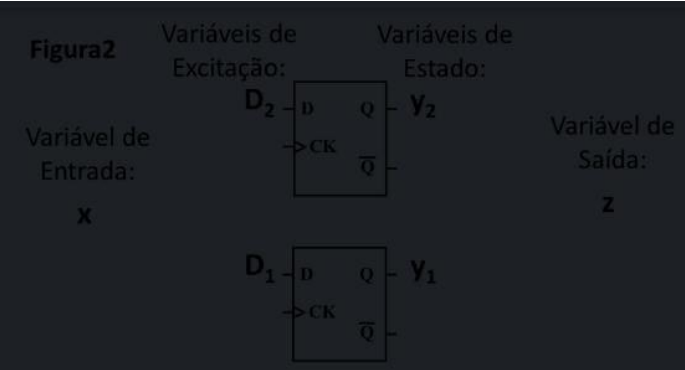
Figura1

Estado	Código Binário $y_2\ y_1$
A	00
B	01
C	10
D	11

	ParaEntrada		
Estado	x=0	x=1	
S^t	S^{t+1}	S^{t+1}	Saída
A	A	B	0
B	C	D	0
C	A	B	1
D	C	D	1

O circuito sequencial da FSM deverá utilizar *Flip-flops* tipo **D**, sensíveis à borda de subida do *clock* (Figura 2). Esta etapa do processo de síntese consistirá na obtenção das expressões de chaveamento de D_1, D_2 e **z**.

Figura2



Pede-se que sejam assinaladas, na Tabela que segue, as expressões de chaveamento correspondentes às variáveis de excitação D_1 e D_2 e também a da variável de saída **z**.

D_1	D_2	z
○ y_2	○ y_2	● y_2 ✓
○ y_1	● y_1 ✓	○ y_1
○ $y_2\ XOR\ y_1$	○ $y_2\ XOR\ y_1$	○ $y_2\ XOR\ y_1$
○ $x\ XOR\ y_2$	○ $x\ XOR\ y_2$	○ $x\ XOR\ y_2$
○ $x\ XOR\ y_1$	○ $x\ XOR\ y_1$	○ $x\ XOR\ y_1$
● x ✓	○ x	○ x

Convertendo na tabela:

(★ Lembrar que as duas últimas linhas trocam de lugar (código de Gray))

	ENTRADA X=0	ENTRADA X=1	
S^t	S^{t+1}	S^{t+1}	Saída
A = 00	A = 00	B = 01	0
B = 01	C = 10	D = 11	0
C = 10 ★	A = 00	B = 01	1
D = 11	C = 10	D = 11	1
$Y_2\ Y_1$	$D_2\ D_1$	$D_2\ D_1$	} dos flipflops

Por Karnaugh em D_2, D_1 e depois na Saída, temos:

$D_2 = Y_1$

$D_1 = X$

Saída $= Y_2$