EA772: CIRCUITOS LÓGICOS

TÓPICO: CIRCUITOS SEQÜENCIAIS

Aspectos de Sistemas Lineares

PEDRO LUIS DIAS PERES

Novembro 2006

Circuitos Seqüenciais — Aspectos de Sistemas Lineares

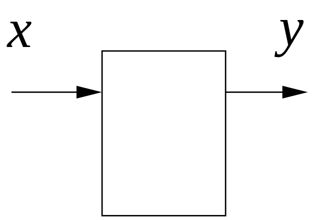
Circuitos lógicos seqüenciais compostos por *flip-flops* D e ou-exclusivos são sistemas lineares.

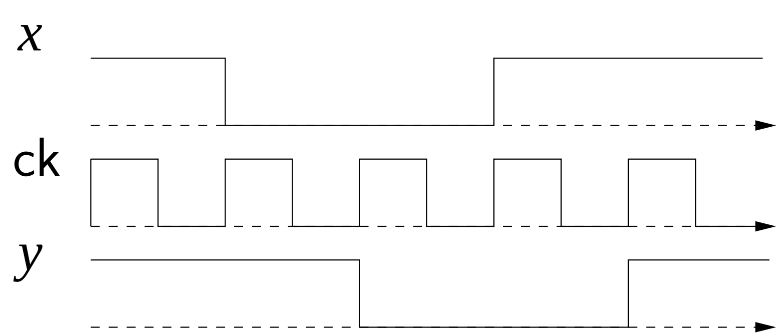
Podem ser descritos por:

- Equações a diferenças;
- Funções de transferência.

Notação

- Letras latinas minúsculas $a, b, \ldots, x, y, z, w$ denotam variáveis *booleanas* $\{0, 1\}$;
- Retângulos representam *flip-flops* do tipo D com *clock* sensível a borda de subida;
- Setas indicam entradas e saídas;





✔ Se os sinais estão sincronizados (isto é, x é a saída de um *flip-flop* que recebe o mesmo sinal de *clock*), a seqüência de zeros e uns do sinal y na saída do *flip-flop* D é igual à seqüência definida pelo sinal x atrasado de um pulso de *clock*.

Notação: Ou-exclusivos

- Ou-exclusivos são (usualmente) representados pelo símbolo $\oplus$;

Tabela verdade do ou-exclusivo
$y = \mathrm{XOR}(x_1, x_2) = x_1 + x_2$

$x + x = 0; \quad x + 0 = x; \quad x + 1 = \bar{x}$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

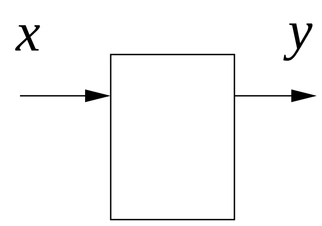
- Ou-exclusivos são associativos, isto é, $x + (y + z) = (x + y) + z$;

- Ou-exclusivos com várias entradas fornecem saída igual a 1 para um número ímpar de entradas iguais a 1 e saída 0 para um número par de entradas iguais a 1;

- A operação ou-exclusivo dos N *bits* que compõem uma palavra $b_0 b_1 b_2 \ldots b_N$ fornece a paridade da palavra. Por exemplo,

$$1 + 0 + 1 + 0 + 1 + 0 + 1 + 1 + 0 = 1 \quad \Rightarrow \quad \text{soma módulo 2}$$

Notação: *flip-flop* tipo D



- Considerando uma seqüência de entrada x_n, tem-se na saída do *flip-flop* a seqüência

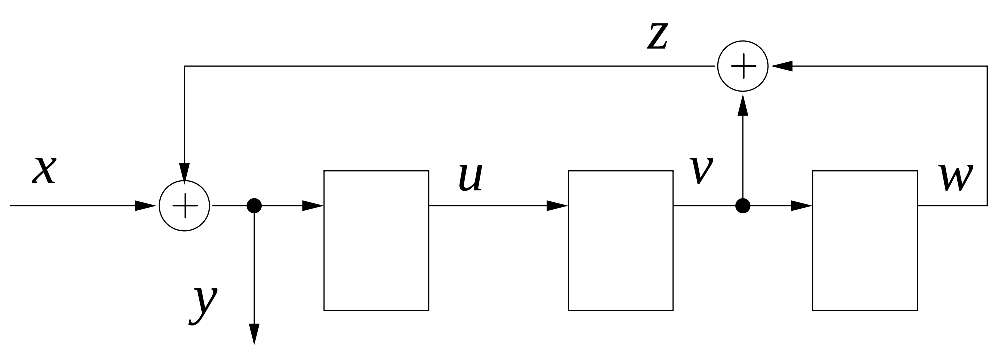
$$y_n = x_{n-1}$$

- O operador D (*delay*) representa um atraso de um pulso de *clock*. Assim,

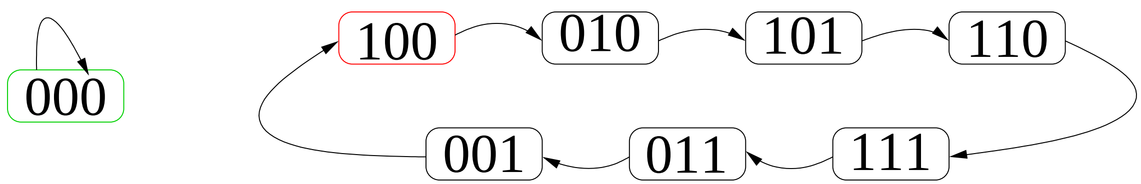
$$y_n = x_{n-1} \quad \Leftrightarrow \quad Y = DX \qquad , \qquad y_n = x_{n-k} \quad \Leftrightarrow \quad Y = D^k X$$

Exemplo 1

- Considere o circuito abaixo com $x_n = 0$, $\forall n$ e a condição inicial $(u_0 v_0 w_0) = (100)$



- A partir da condição inicial (*estado inicial*), pode-se determinar a seqüência de estados u_n, v_n, w_n e também a seqüência de saída y_n (sinal periódico)



$$y_n = 01110010111001\ldots$$

Exemplo 1 (cont.)

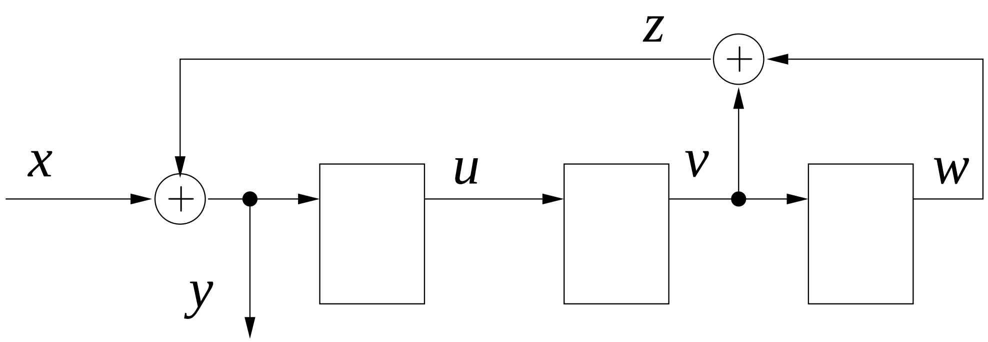
- A saída y, para cada instante n, é dada pela operação

$$y = x + z = x + (v + w) = x + v + w$$

- Assim, a seqüência de estados e a saída podem ser colocadas em uma tabela:

n	x	u	v	w	y
0	0	1	0	0	0
1	0	0	1	0	1
2	0	1	0	1	1
3	0	1	1	0	1
4	0	1	1	1	0
5	0	0	1	1	0
6	0	0	0	1	1
7	0	1	0	0	0
$\vdots$			$\vdots$		

Exemplo 1 (cont.)



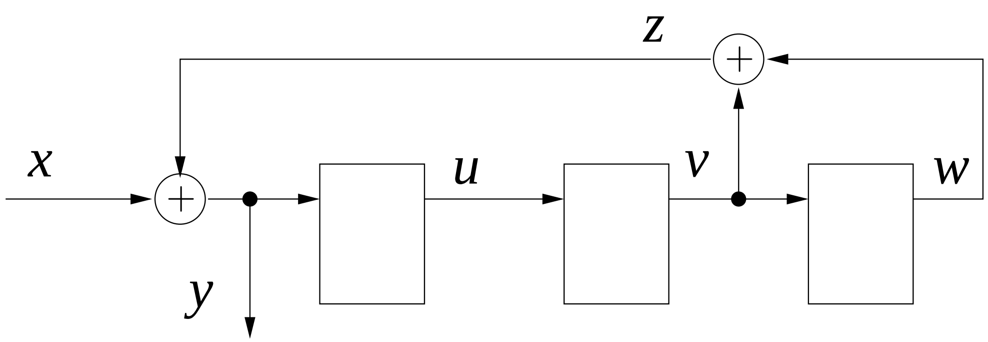
- Do circuito obtêm-se as relações

$$u_n = y_{n-1} \quad , \quad v_n = u_{n-1} \quad , \quad w_n = v_{n-1} \quad , \quad z_n = v_n + w_n$$

- O comportamento dinâmico do circuito é descrito pela equação a diferenças

$$y_n = x_n + y_{n-2} + y_{n-3} \quad , \quad (u_0 v_0 w_0) = (y_{-1} y_{-2} y_{-3}) \;\; \text{(condições iniciais)}$$

Exemplo 1 (cont.)



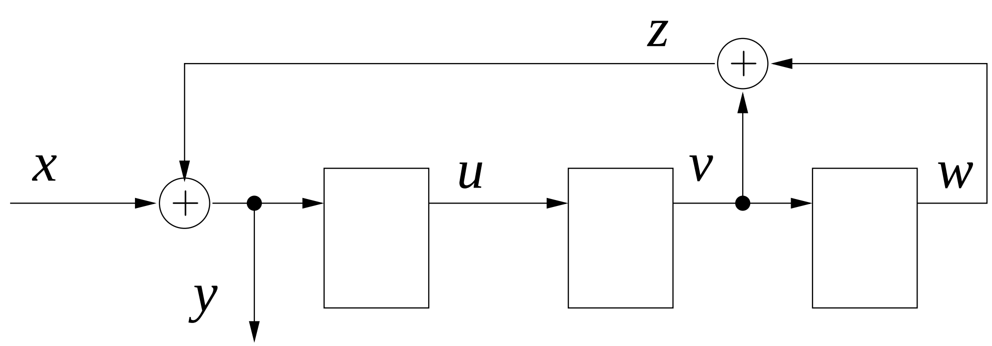
- Descrição em termos das variáveis de estado u_n, v_n e w_n

$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} x_n$$

- Equação da saída y_n

$$y_n = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} x_n$$

Exemplo 1 (cont.)



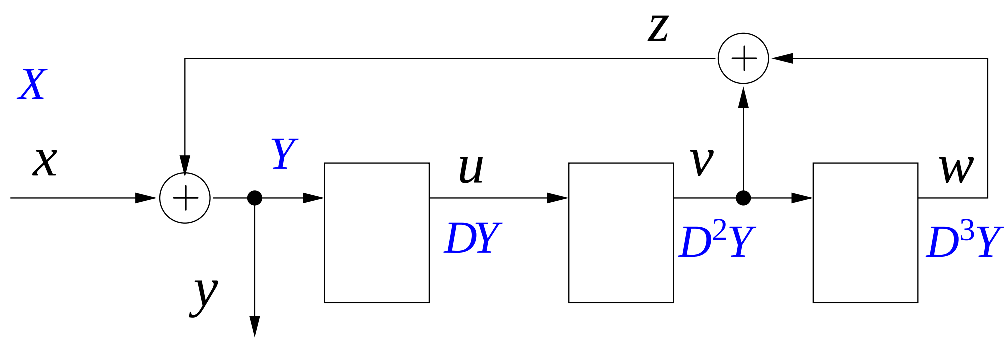
- Manipulando a equação a diferenças, tem-se $y_n + y_{n-2} + y_{n-3} = x_n$

- Utilizando o operador D, pode-se descrever a relação entre a entrada X e a saída Y pela razão entre dois polinômios em D (*função de transferência*)

$$(1 + D^2 + D^3)Y = X \quad \Rightarrow \quad \frac{Y}{X} = \frac{1}{1 + D^2 + D^3}$$

- Os polinômios em D possuem coeficientes 0 ou 1 e o símbolo "$+$" denota a operação ou-exclusivo.

Exemplo 1 (cont.)



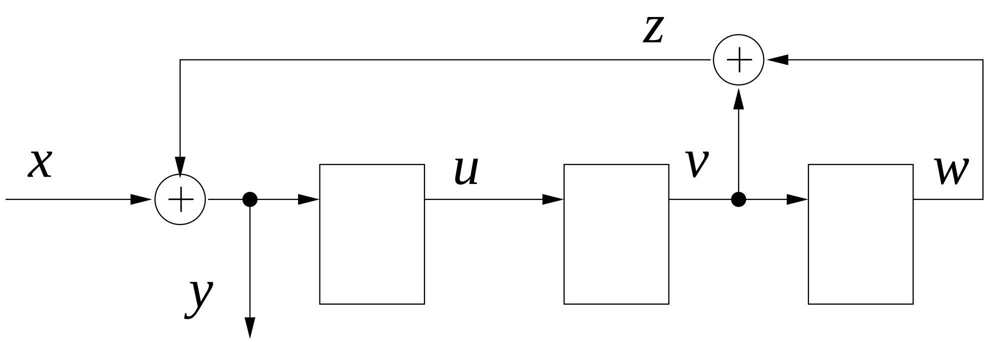
- Manipulando a equação a diferenças, tem-se $y_n + y_{n-2} + y_{n-3} = x_n$

- Utilizando o operador D, pode-se descrever a relação entre a entrada X e a saída Y pela razão entre dois polinômios em D (*função de transferência*)

$$(1 + D^2 + D^3)Y = X \quad \Rightarrow \quad \frac{Y}{X} = \frac{1}{1 + D^2 + D^3}$$

- Os polinômios em D possuem coeficientes 0 ou 1 e o símbolo "$+$" denota a operação ou-exclusivo.

Exemplo 1 (comentários)



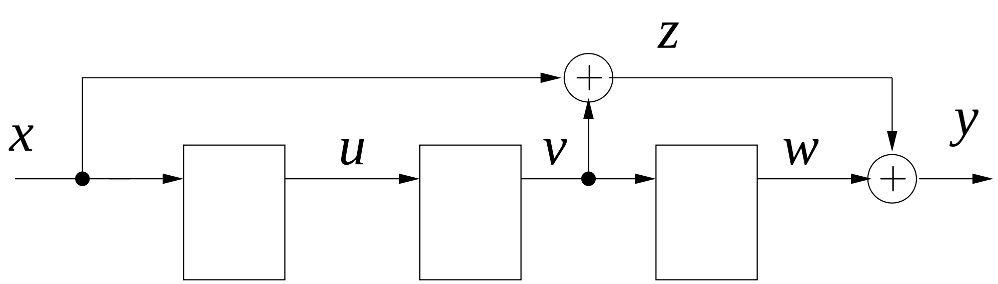
• Note que para $x_n = 0$, $\forall n$ e $(u_0 v_0 w_0) = (000)$, tem-se $y_n = 0$, $\forall n$ e que um circuito formado por N *flip-flops* possui 2^N estados distintos;

• Para $x_n = 0$, $\forall n$ e uma condição inicial qualquer (não nula), o circuito passa por todos os estados possíveis com exceção do 000, sendo por isso chamado de máquina de estado de máxima seqüência ou de gerador pseudo-aleatório. A saída y_n é um sinal periódico de período igual a 7;

✔ Essa propriedade (seqüência de $2^N - 1$ estados distintos) está relacionada ao fato do polinômio $P(D) = 1 + D^2 + D^3$ ser um polinômio *primitivo*.

Exemplo 2

- Considere o circuito abaixo com $x_n = 0$, $\forall n$ e a condição inicial $(u_0 v_0 w_0) = (100)$



- A equação a diferenças que descreve o circuito é dada por

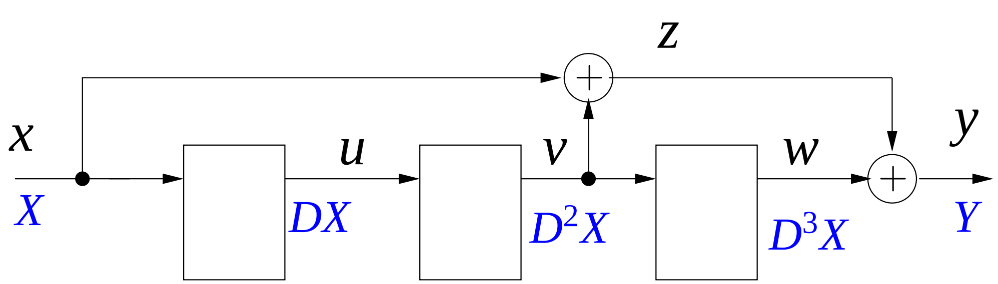
$$y_n = x_n + x_{n-2} + x_{n-3}$$

- A função de transferência é dada por

$$Y = (1 + D^2 + D^3)X \quad \Rightarrow \quad \frac{Y}{X} = 1 + D^2 + D^3$$

Exemplo 2

- Considere o circuito abaixo com $x_n = 0$, $\forall n$ e a condição inicial $(u_0 v_0 w_0) = (100)$



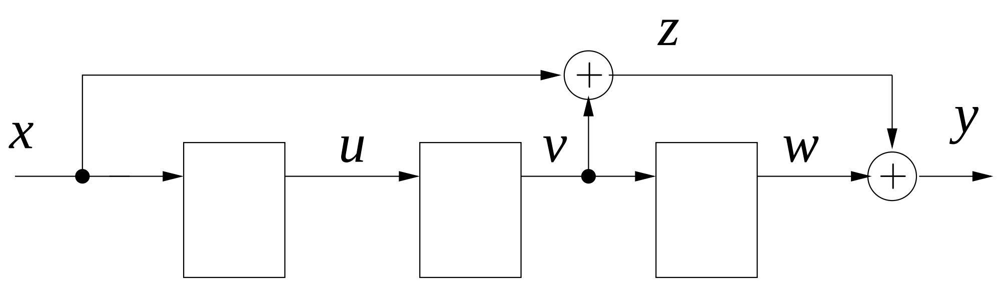
- A equação a diferenças que descreve o circuito é dada por

$$y_n = x_n + x_{n-2} + x_{n-3}$$

- A função de transferência é dada por

$$Y = (1 + D^2 + D^3)X \quad \Rightarrow \quad \frac{Y}{X} = 1 + D^2 + D^3$$

Exemplo 2 (cont.)



- Descrição em termos das variáveis de estado u_n, v_n e w_n

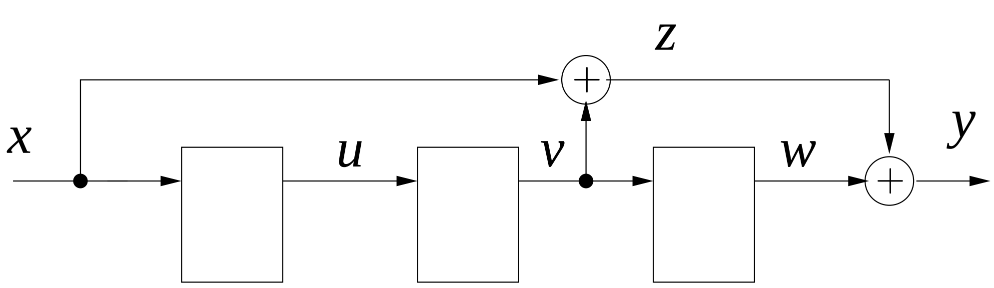
$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} x_n$$

- Equação da saída y_n

$$y_n = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} x_n$$

Exemplo 2 (cont.)

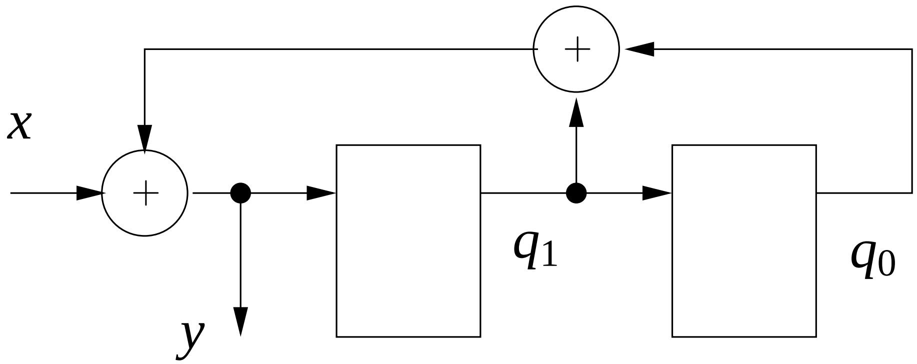
- A seqüência de estados e a saída são dadas por



n	x	u	v	w	y
0	0	1	0	0	0
1	0	0	1	0	1
2	0	0	0	1	1
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
$\vdots$			$\vdots$		

- Note que o efeito da condição inicial desvanece com n

Circuito Embaralhador



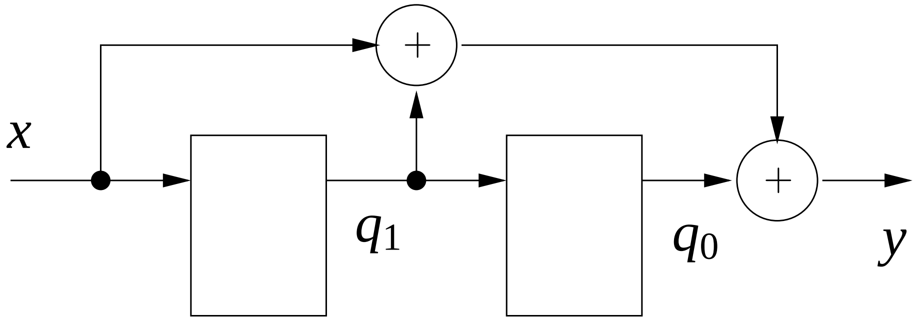
- Função de transferência

$$\frac{Y}{X} = \frac{1}{1+D+D^2}$$

- Considerando $x_n = 0$, $\forall n$ e $q_1 q_0 = 11$, tem-se

$$11 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow \cdots \quad , \quad y_n = 011011011\ldots \quad \text{(período 3)}$$

Circuito Desembaralhador



- Função de transferência

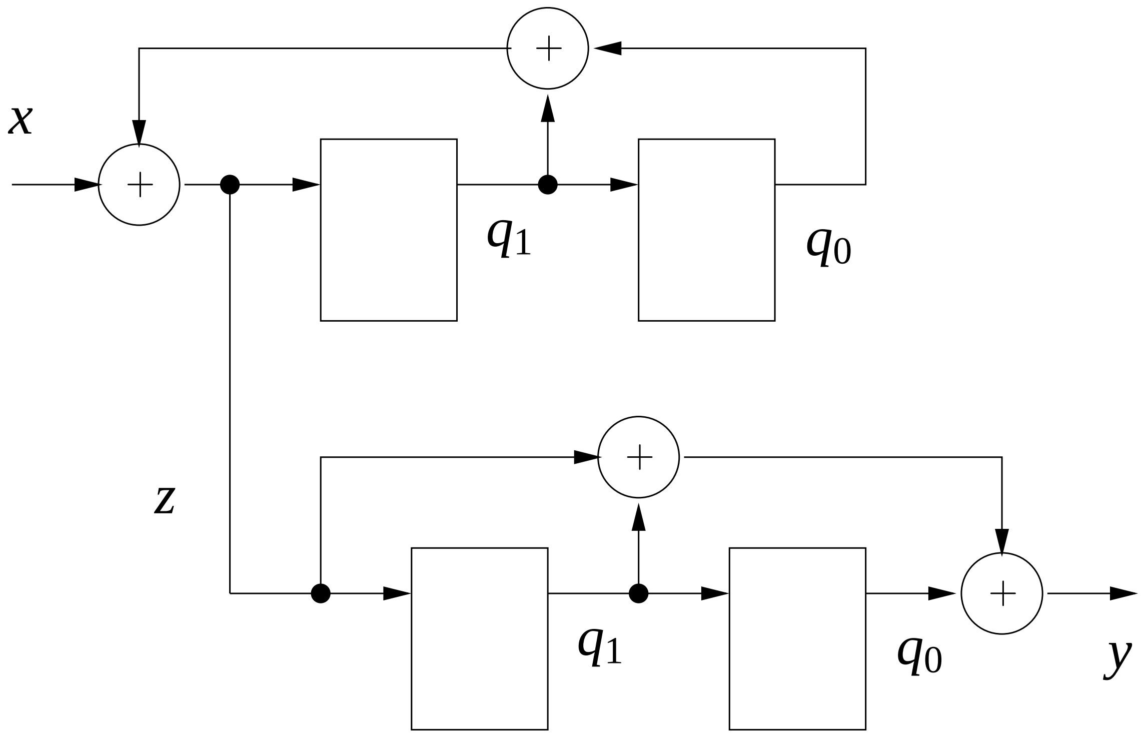
$$\frac{Y}{X} = 1 + D + D^2$$

- Considerando $x_n = 0$, $\forall n$ e $q_1 q_0 = 11$, tem-se

$$11 \to 01 \to 00 \to 00 \to \cdots \quad , \quad y_n = 0100000\ldots$$

Circuito Embaralhador-Desembaralhador

- Conectando a saída do circuito embaralhador na entrada do circuito desembaralhador (e chamando de z o ponto de conexão) tem-se



Circuito Embaralhador-Desembaralhador (cont.)

- A função de transferência do circuito é dada por

$$\frac{Z}{X} = \frac{1}{1+D+D^2} \quad , \quad \frac{Y}{Z} = 1+D+D^2 \quad \Rightarrow \quad \frac{Y}{X} = 1$$

- Considerando $x_n = 0$, $\forall n$, $q_1 q_0 = 11$ no embaralhador e $q_1 q_0 = 00$ no desembaralhador, tem-se

n	x	q_1	q_0	z	q_1	q_0	y	
0	0	1	1	0	0	0	0	trans.
1	0	0	1	1	0	0	1	trans.
2	0	1	0	1	1	0	0	
3	0	1	1	0	1	1	0	
4	0	0	1	1	0	1	0	
5	0	1	0	1	1	0	0	
6	0	1	1	0	1	1	0	

Circuito Embaralhador-Desembaralhador (cont.)

- Considerando a seqüência $x_n = 101010101\ldots$, $q_1q_0 = 11$ no embaralhador e $q_1q_0 = 00$ no desembaralhador, tem-se

n	x	q_1	q_0	z	q_1	q_0	y
0	1	1	1	1	0	0	1
1	0	1	1	0	1	0	1
2	1	0	1	0	0	1	1
3	0	0	0	0	0	0	0
4	1	0	0	1	0	0	1
5	0	1	0	1	1	0	0
6	1	1	1	1	1	1	1
7	0	1	1	0	1	1	0
8	1	0	1	0	0	1	1
9	0	0	0	0	0	0	0
				$\vdots$			

Circuito Embaralhador-Desembaralhador (cont.)

- Se as condições iniciais do embaralhador e do desembaralhador forem iguais, não há transitório e $y_n = x_n$, $\forall n$.

n	x	q_1	q_0	z	q_1	q_0	y
0	1	1	0	0	1	0	1
1	0	0	1	1	0	1	0
2	1	1	0	0	1	0	1
3	0	0	1	1	0	1	0
4	1	1	0	0	1	0	1
5	0	0	1	1	0	1	0
6	1	1	0	0	1	0	1
				$\vdots$			

✔ Aplicações em criptografia.

Resposta ao Impulso

- Para $x_n = 100000\ldots$ e $q_1q_0 = 00$, tem-se

Embaralhador

x	q_1	q_0	y
1	0	0	1
0	1	0	1
0	1	1	0
0	0	1	1
0	1	0	1
0	1	1	0
$\vdots$			

Circuito do tipo IIR

Desembaralhador

x	q_1	q_0	y
1	0	0	1
0	1	0	1
0	0	1	1
0	0	0	0
0	0	0	0
0	0	0	0
$\vdots$			

Circuito do tipo FIR

✔ As respostas ao impulso apresentam um comportamento similar ao obtido com $x_n = 0$, $\forall n$ e condição inicial não nula

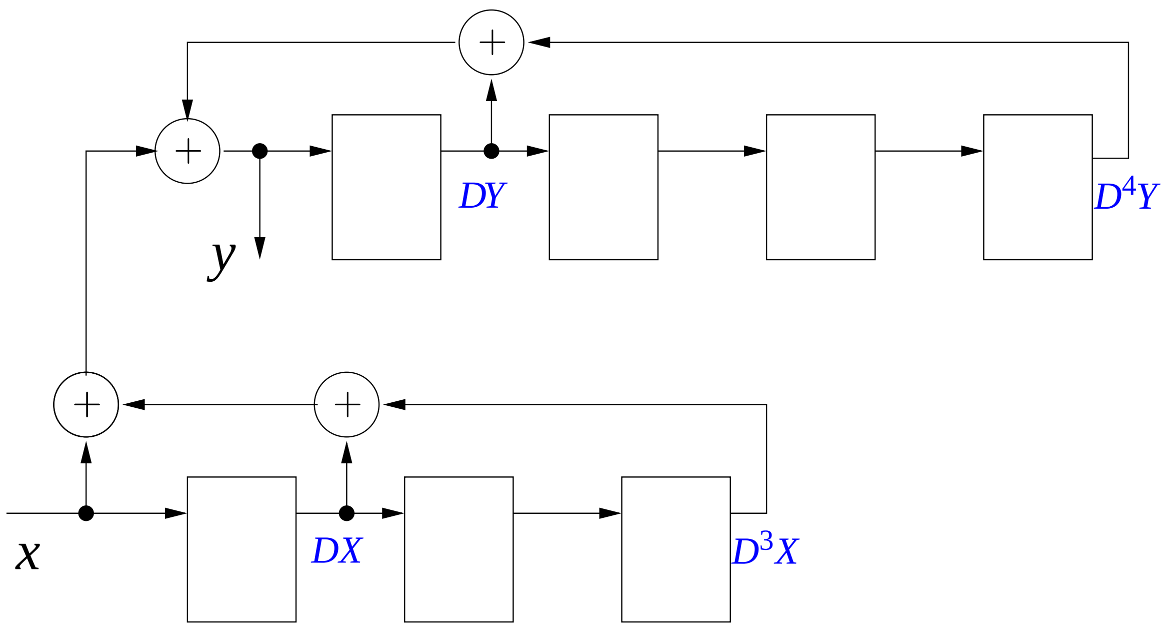
$\rightsquigarrow$ O circuito do Exemplo 1 também é do tipo *IIR — Infinite Impulse Response*, enquanto que o do Exemplo 2 é do tipo *FIR — Finite Impulse Response*;

Realizações

- A função de transferência

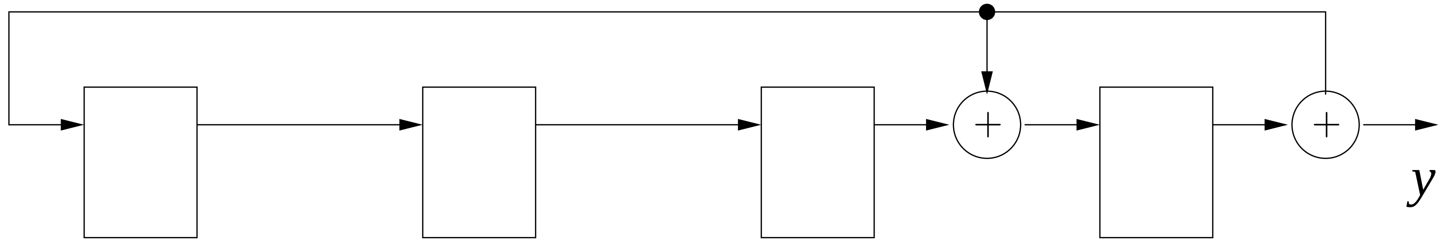
$$\frac{Y}{X} = \frac{1+D+D^3}{1+D+D^4} \quad \Rightarrow \quad Y = DY + D^4Y + X + DX + D^3X$$

pode ser realizada pelo circuito



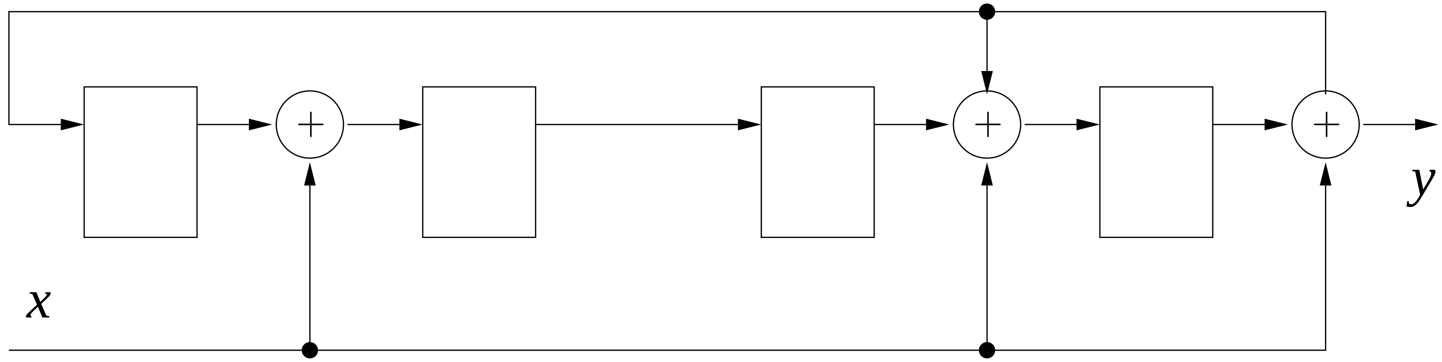
Realizações

- Outra realização da função de transferência $Y = DY + D^4Y + X + DX + D^3X$ pode ser construída, primeiro para as parcelas que dependem de Y



Realizações

- Outra realização da função de transferência $Y = DY + D^4Y + X + DX + D^3X$ pode ser construída, primeiro para as parcelas que dependem de Y



e depois completando com as parcelas que dependem de X

✔ Princípio da superposição

Geradores Pseudo-Aleatórios

- Uma máquina linear com entrada nula é denominada *autônoma*.

- Circuitos autônomos definidos pela função de transferência

$$\frac{1}{P(D)}$$

podem gerar uma seqüência de tamanho máximo se $P(D)$ for um polinômio *primitivo*.

- Esses circuitos são denominados *geradores pseudo-aleatórios*.

Polinômios Primitivos

• Um polinômio é *irredutível* se não puder ser fatorado por polinômios de graus menores.

• Um polinômio irredutível de grau n é *primitivo* se for fator do polinômio $D^m + 1$ ($m = 2^n - 1$) e não for fator de $D^k + 1$ para $k < m$.

• Exemplos de polinômios primitivos

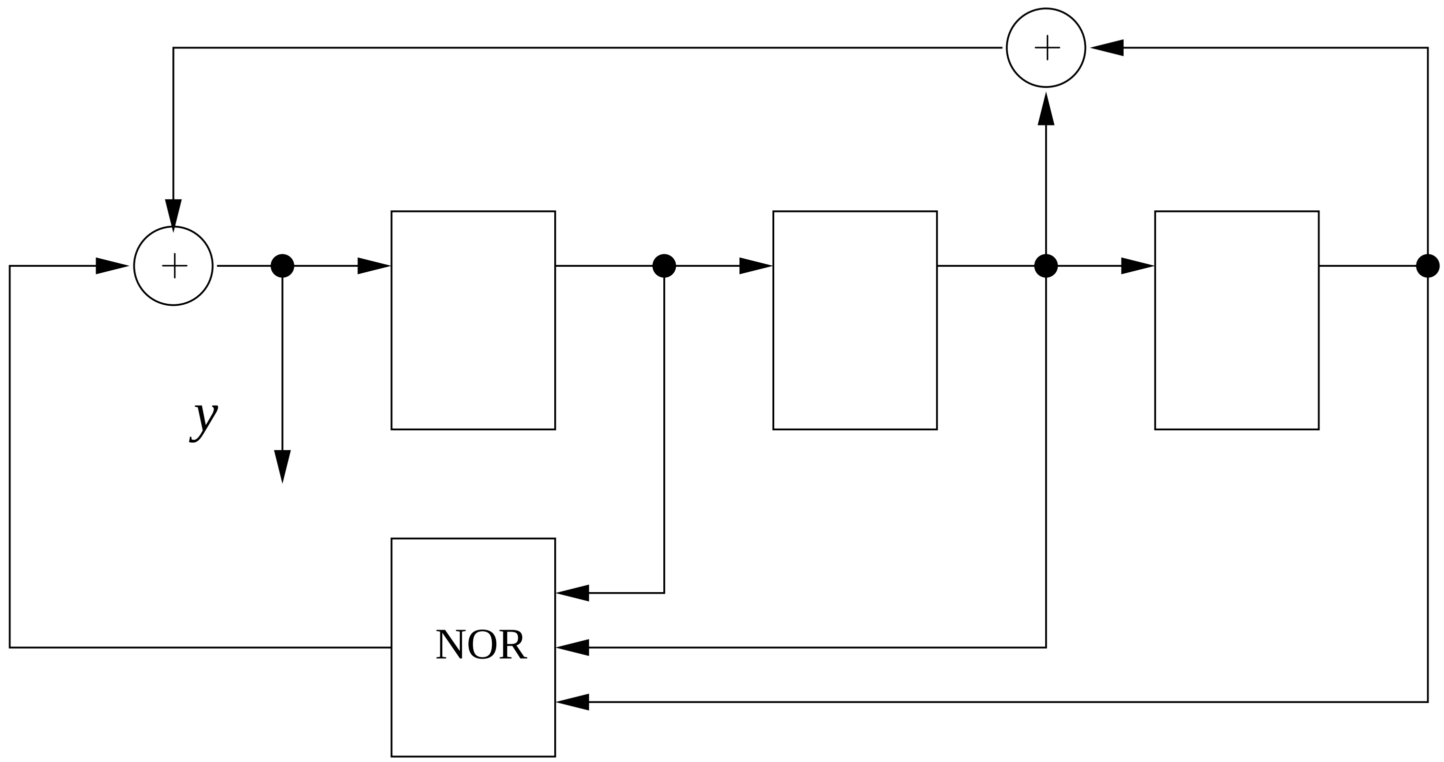
$$1 + D + D^2 \quad ; \quad 1 + D + D^4 \quad ; \quad 1 + D^3 + D^4 \quad ; \quad 1 + D^2 + D^5 \quad ; \quad 1 + D^3 + D^5$$

• Exemplos de polinômios que não são primitivos

$$1 + D^2 = (1 + D)(1 + D) \quad ; \quad D + D^2 + D^3 = D(1 + D + D^2)$$

✔ Polinômios primitivos podem ser usados para gerar corpos de Galois a partir da extensão de uma base de representação.

Gerador Pseudo-Aleatório Autoinicializável



✔ Se o estado, por alguma razão, ficar na condição inicial 000, a presença do NOR força a entrada de um 1 no primeiro pulso de *clock*. A partir daí, o gerador entra no ciclo de máxima seqüência (que não contém o 000)

Códigos

- Tópico relevante em comunicações

- O foco aqui é restrito a códigos binários gerados de maneira linear

  $\rightsquigarrow$ Por operações envolvendo matrizes (com adição $=$ ou-exclusivo)

  $\rightsquigarrow$ Ou por sistemas dinâmicos formados por *flip-flops* tipo D e ou-exclusivos

  $\rightsquigarrow$ Palavra $000\cdots0$ faz parte do código

- O número de *bits* distintos entre duas palavras de um código é definido como a *distância de Hamming* entre as palavras

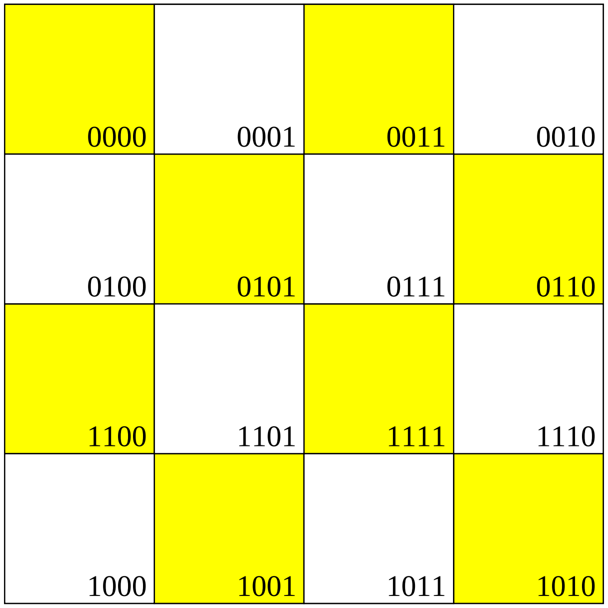
  $\rightsquigarrow$ Códigos com distância mínima igual a 2 permitem a deteção de erros simples (códigos de paridade)

  $\rightsquigarrow$ Códigos com distância mínima igual a 3 permitem a correção de erros simples (códigos de Hamming)

Códigos — Exemplos

- Código com 4 *bits* e distância mínima igual a 2 (3 *bits* de informação mais 1 *bit* de paridade, que completa um número par de *bits* iguais a 1)

Palavras do Código			
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

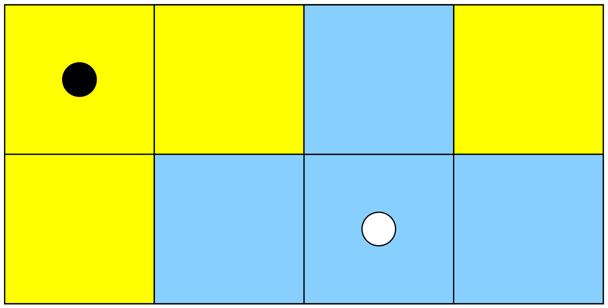


$\rightsquigarrow$ Erros em um *bit* produzem palavras que não pertencem ao código e podem ser detetados

Códigos — Exemplos

• Código com 3 *bits* e distância mínima igual a 3 $(1$ *bit* de informação $\{0, 1\}$ e 2 *bits* de redundância ou paridade).

0 0 0
1 1 1



⇝ Erro simples pode ser detetado e corrigido

⇝ Este é um código de Hamming $(3, 1)$, isto é, palavras de $3 = 2^2 - 1$ *bits* sendo 1 de informação e 2 de paridade.

Códigos de Hamming (n, k)

- São códigos com distância 3, palavras de $n = 2^m - 1$ *bits* sendo $k = n - m$ de informação e $m = n - k$ de redundância

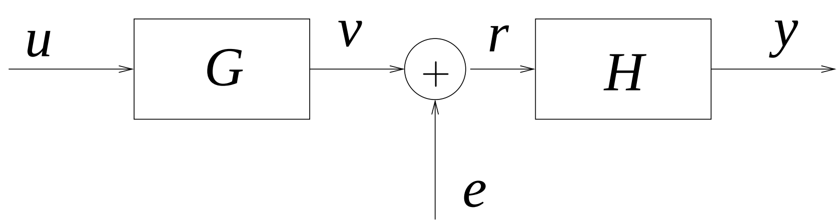
  $\rightsquigarrow$ Podem ser gerados por polinômios primitivos $P(D)$ de grau m

- Codificadores: circuitos lineares que transformam uma seqüência de *bits* (mensagem mais *bits* nulos de redundância) em uma palavra código

- Essa operação linear pode ser descrita por matrizes e vetores ou por sistemas dinâmicos com *flip-flops* D e ou-exclusivos

- Decodificadores: circuitos lineares que recuperam a informação transmitida, detetando a ocorrência de erros simples e eventualmente realizando a correção

Codificadores e Decodificadores



- Mensagem u, palavra código v, erro e, palavra recebida $r = v + e$, saída do decodificador y, matriz geradora G e matriz de paridade H

$$v = uG \quad , \quad s = rH' \quad \text{síndrome}$$

$\leadsto$ Por construção, quando não há erro

$$s = vH' = 0 \quad \Rightarrow \quad GH' = 0$$

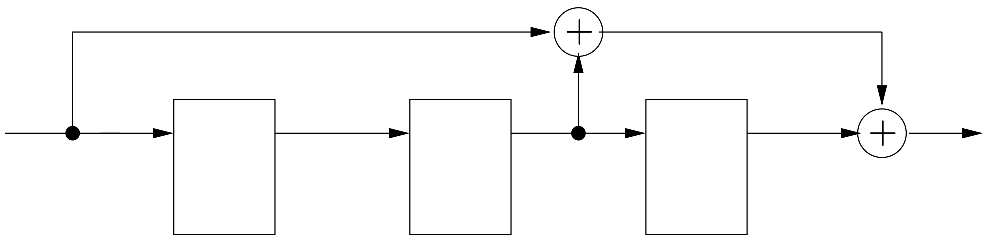
Exemplo de Código de Hamming $(7,4)$

- Gerado a partir do polinômio

$$P(D) = 1 + D^2 + D^3$$

$\rightsquigarrow$ Palavras de 7 *bits* (4 de informação e 3 de redundância)

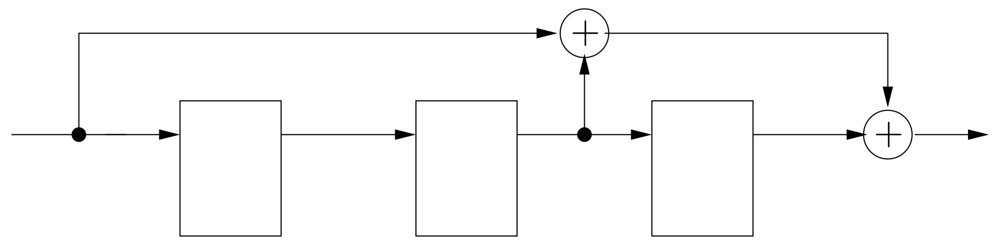
- Circuito codificador



- Com condições iniciais nulas, a seqüência de entrada $u_0u_1u_2u_3000$ produz, após 7 pulsos de *clock* a palavra código correspondente $v_0v_1v_2v_3v_4v_5v_6$

Exemplo de Código de Hamming $(7,4): P(D) = 1 + D^2 + D^3$

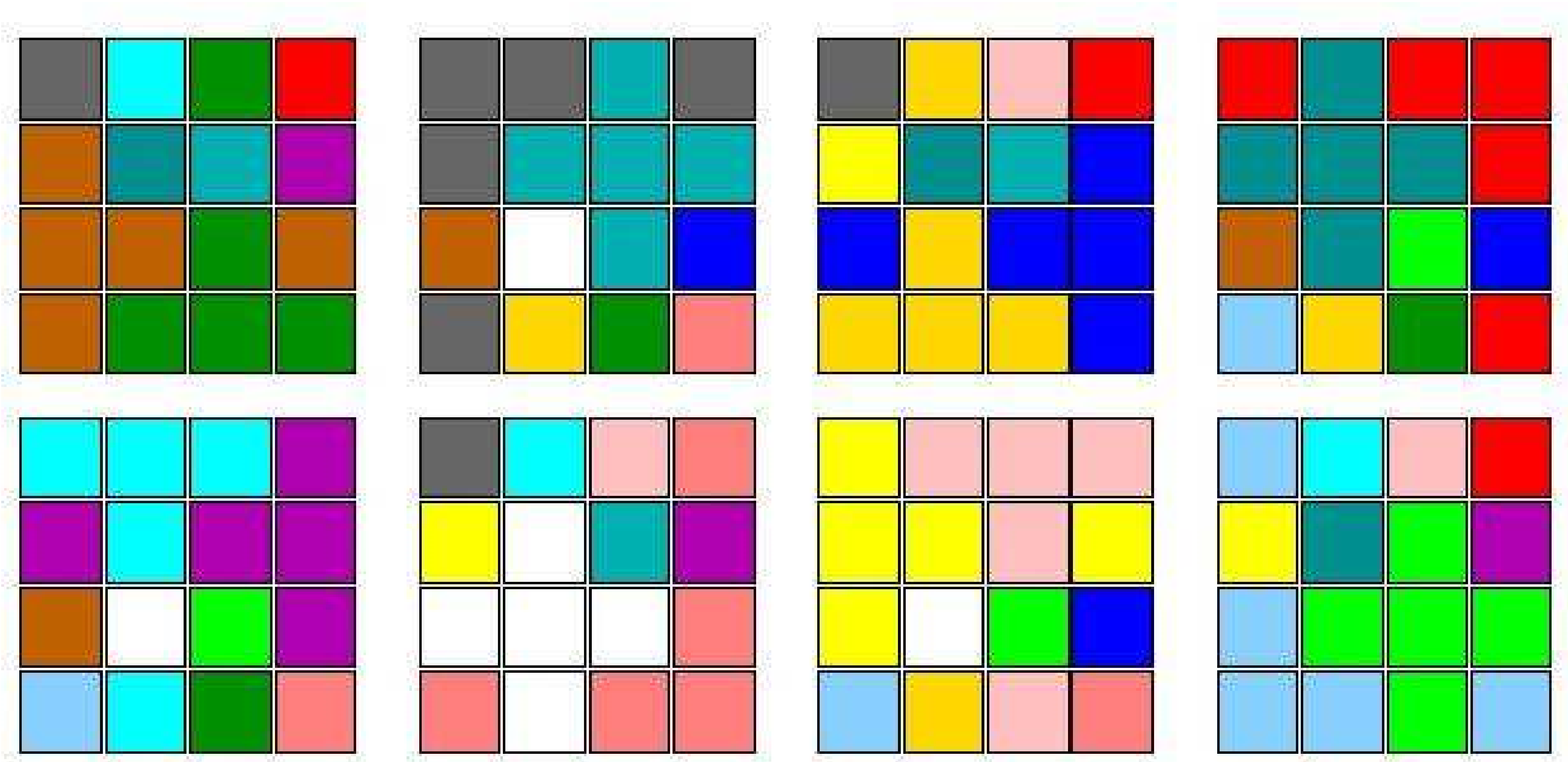
Circuito Codificador



$$
\begin{array}{c|ccc|c}
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 &
\end{array}
\qquad
\begin{array}{c|ccc|c}
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 &
\end{array}
\qquad
\begin{array}{c|ccc|c}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 &
\end{array}
\qquad
\begin{array}{c|ccc|c}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 \\
 & 0 & 0 & 0 &
\end{array}
\qquad
\begin{array}{c|ccc|c}
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
 & 0 & 0 & 0 &
\end{array}
$$

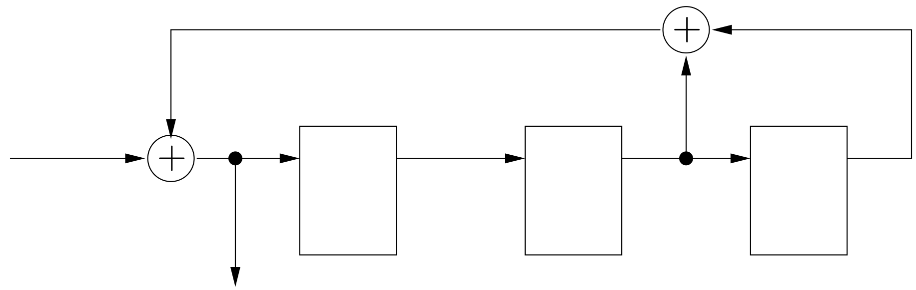
- As demais palavras do código são geradas por superposição

Exemplo de Código de Hamming $(7, 4)$: Mapa de Karnaugh



Exemplo de Código de Hamming $(7,4) : P(D) = 1 + D^2 + D^3$

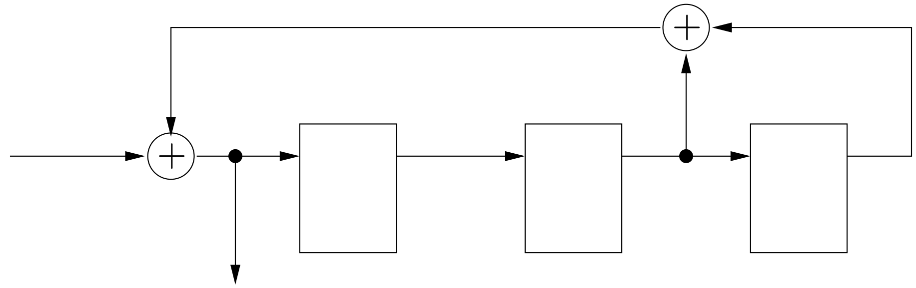
Circuito Decodificador



$$\begin{array}{c|ccc|c} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & \end{array} \qquad \begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & \end{array} \qquad \begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & \end{array} \qquad \begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ & 0 & 0 & 0 & \end{array}$$

- Palavras que pertencem ao código são decodificadas corretamente, produzindo a síndrome 000

Exemplo de Código de Hamming $(7,4): P(D) = 1 + D^2 + D^3$

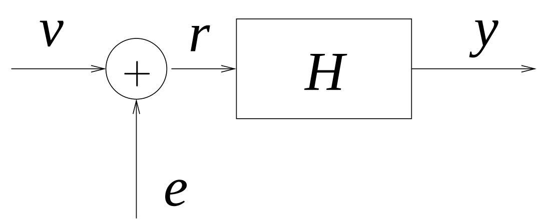


Circuito Decodificador

$$
\begin{array}{c|ccc|l}
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
 & \color{red}{0} & \color{red}{0} & \color{red}{1} & \rightsquigarrow \text{1o.}
\end{array}
\qquad
\begin{array}{c|ccc|l}
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 \\
 & \color{red}{0} & \color{red}{1} & \color{red}{1} & \rightsquigarrow \text{2o.}
\end{array}
\qquad
\begin{array}{c|ccc|l}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 \\
 & \color{red}{1} & \color{red}{1} & \color{red}{1} & \rightsquigarrow \text{3o.}
\end{array}
\qquad
\begin{array}{c|ccc|l}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
 & \color{red}{1} & \color{red}{1} & \color{red}{0} & \rightsquigarrow \text{4o.}
\end{array}
$$

- Cada tipo de erro é identificado por uma síndrome diferente, pois há $2^3 - 1$ estados finais distintos no decodificador (se não houver erro, a síndrome é 000)

Circuito Decodificador



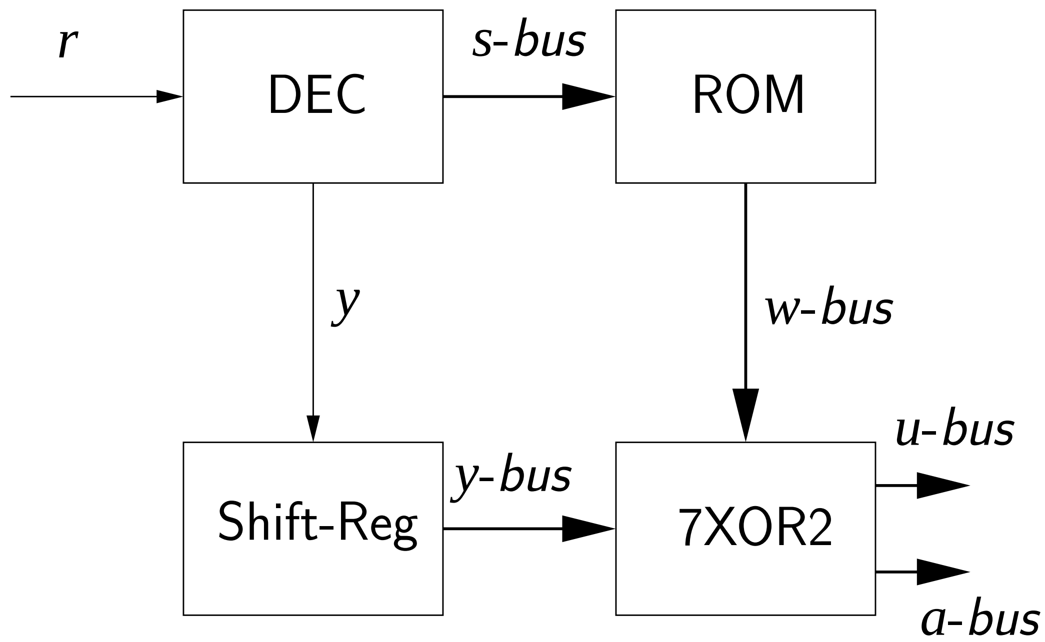
- Por superposição, a saída y possui uma componente devido à palavra do código v e uma componente devido ao erro e

- Tabela de síndromes

Erro	1000000	0100000	0010000	0001000	0000100	0000010	0000001
Síndrome	001	011	111	110	101	010	100
Saída devido ao erro	1011100	0101110	0010111	0001011	0000101	0000010	0000001

$\rightsquigarrow$ Palavras do código (erro $e = 0000000$) produzem síndrome 000

Circuito Decodificador com Correção de Erro



- A palavra r chega seqüencialmente ao decodificador, produz a saída y (que é armazenada em um registro de deslocamento) e uma síndrome s-bus

- A memória ROM armazena a tabela de síndromes, gerando a palavra que deve ser adicionada (ou-exclusivo) à saída y para corrigir o efeito do erro. A mensagem é recuperada em u-bus com a-$bus = 000$

Implementação com Matrizes — Geração

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

- A primeira linha da matriz geradora G é a palavra do código associada à informação 1000, as demais linhas obtêm-se por deslocamento (códigos cíclicos)

$$\begin{bmatrix} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Implementação com Matrizes — Síndrome

$$H = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

- A matriz de paridade H contém todas as possíveis síndromes (exceto 000). A primeira coluna contém a síndrome associada ao erro no primeiro *bit* 1000000, e assim por diante.

$$s = rH' = (v+e)H' = \underbrace{vH'}_{=0} + eH' \qquad \Rightarrow \qquad \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = 0$$

Síndrome

- As funções s_0, s_1 e s_2 (síndrome) são determinadas de maneira biunívoca por erros simples

$$\begin{bmatrix} s_0 & s_1 & s_2 \end{bmatrix} = \begin{bmatrix} e_0 & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$s_0 = e_2 + e_3 + e_4 + e_6$$

$$s_1 = e_1 + e_2 + e_3 + e_5$$

$$s_2 = e_0 + e_1 + e_2 + e_4$$

Geração

- Similarmente, os *bits* que compõem as palavras do código podem ser obtidos da matriz geradora por operações ou-exclusivo

$$v_0 = u_0 \quad , \quad v_1 = u_1 \quad , \quad v_2 = u_0 + u_2 \quad , \quad v_3 = u_0 + u_1 + u_3$$
$$v_4 = u_1 + u_2 \quad , \quad v_5 = u_2 + u_3 \quad , \quad v_6 = u_3$$

$$\begin{bmatrix} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

- Códigos de Hamming $(7,4)$ poderiam ser gerados de outras maneiras

Códigos de Hamming $(7,4)$

- A partir do polinômio primitivo $P(D) = 1 + D + D^3$

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

- Compondo palavras do código $p_0p_1u_0p_2u_1u_2u_3$ a partir da informação $u_0u_1u_2u_3$ e da paridade $p_0p_1p_2$ (código sistemático)

$$p_0 = u_0 + u_1 + u_3 \quad , \quad p_1 = u_0 + u_2 + u_3 \quad , \quad p_2 = u_1 + u_2 + u_3$$

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad \leadsto \text{ não cíclica}$$

Circuitos Seqüenciais — Aspectos de Sistemas Lineares

Circuitos lógicos seqüenciais compostos por *flip-flops* D e ou-exclusivos são sistemas lineares.

Podem ser descritos por:

- ✔ Equações a diferenças;
- ✔ Funções de transferência.

- ➠ Auto-função?
- ➠ Raízes da equação característica?
- ➠ Controlabilidade e observabilidade?

Bibliografia

- Introdução à Análise e Síntese de Circuitos Lógicos, Ivanil Bonatti e Marcos Madureira, Editora da Unicamp, 1990.
- Notas de aula de EA772

  `http://www.dt.fee.unicamp.br/~peres/ea772/ea772.html`
- Computer Aided Logical Design with Emphasis on VLSI, Frederick J. Hill and Gerald R. Peterson, John Wiley & Sons, Inc., Fourth Edition, 1993.