Lecture
0_7_7

Linear Feedback Shift Registers: An overview



Paolo PRINETTO
Politecnico di Torino (Italy)
University of Illinois at Chicago, IL (USA)

Paolo.Prinetto@polito.it
prinetto@uic.edu

www.testgroup.polito.it
www.comitato-girotondo.org

License Information

This work is licensed under the Creative Commons BY-NC License



To view a copy of the license, visit:
http://creativecommons.org/licenses/by-nc/3.0/legalcode

Rel. 26.04.2017
© Prinetto - 2017

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Rel. 26.04.2017
© Prinetto - 2017

Goal

– This lecture presents a brief overview of Linear Feedback Shift Registers and their applications.

© Prinetto - 2017

Outline

- Association bit-vectors ↔ polynomials
- Operations on polynomials
- Primitive Polynomials
- Arithmetic Circuits for polynomials
  - Multipliers
  - Dividers
- Autonomous LFSRs
- Applications
  - Pseudo-Random Number Generators
  - Signature Generators
    - Serial
    - Parallel

© Prinetto - 2017

Outline

- Association bit-vectors ↔ polynomials
- Operations on polynomials
- Primitive Polynomials
- Arithmetic Circuits for polynomials
  - Multipliers
  - Dividers
- Autonomous LFSRs
- Applications
  - Pseudo-Random Number Generators
  - Signature Generators
    - Serial
    - Parallel

© Prinetto - 2017

Associazione bitvector □□polinomi

Un qualsiasi vettore binario

$$R = a_m\ a_{m-1} \ldots a_2\ a_1\ a_0$$

può essere espresso tramite un *polinomio binario* di grado m

$$P(x) = a_m x^m + a_{m-1} x^{m-1} \ldots + a_1 x + a_0$$

essendo x una qualsiasi variabile di comodo.

Esempio: $10111 \rightarrow x^4 + x^2 + x + 1$

Rel. 26.04.2017
© Prinetto - 2017

Associazione bitvector □ □ polinomi

Un qualsiasi vettore binario

$$R = a_m\ a_{m-1} \ldots a_2\ a_1\ a_0$$

può essere espresso tramite un ***polinomio binario*** di grado **m**

$$P(x) = a_m x^m + a_{m-1} x^{m-1} \ldots + a_1 x + a_0$$

essendo **x** una qualsiasi variabile di comodo.

Esempio: 10111 → x

spesso chiamato
funzione generatrice
del vettore R.

Rel. 26.04.2017
© Prinetto - 2017

Outline

- Association bit-vectors ↔ polynomials
- Operations on polynomials
- Primitive Polynomials
- Arithmetic Circuits for polynomials
  - Multipliers
  - Dividers
- Autonomous LFSRs
- Applications
  - Pseudo-Random Number Generators
  - Signature Generators
    - Serial
    - Parallel

© Prinetto - 2017

Operazioni sui polinomi

In modo del tutto analogo a quanto avviene nell'algebra degli interi, è possibile definire sui polinomi le usuali operazioni aritmetiche.

N.B. :

Tutte le operazioni elementari sui coefficienti devono essere effettuate | 2 |

© Prinetto - 2017

"Congruenza" tra polinomi

È possibile definire anche tra polinomi l'operazione di "modulo".

Due polinomi r(x) ed d(x) sono detti *congruenti modulo* n(x), scritto

$$r(x) = d(x) \bmod n(x)$$

se esiste un polinomio q(x) tale che

$$d(x) = n(x) \cdot q(x) + r(x)$$

dove

- **d(x) è il dividendo**
- **n(x) è il divisore**
- **q(x) è il quoziente**
- **r(x) è il resto**

Rel. 26.04.2017
© Prinetto - 2017

Esempi

- **Dati:**
  - $Q(x) = x^2 + x + 1$
  - $P(x) = x^3 + x^2 + 1$
- **si ha:**
  - $P(x) + Q(x) = x^3 + 2x^2 + x + 2 = x^3 + x$
  - $P(x) - Q(x) = x^3 - x = x^3 + x$
  - $P(x) \cdot Q(x) = x^5 + x + 1$
  - $P(x) / Q(x) = x$
  - $P(x) \bmod Q(x) = x + 1$

© Prinetto - 2017

Esempio di calcolo

Dati:

- $Q(x) = x^2 + x + 1$
- $P(x) = x^3 + x^2 + 1$

si ha:

- $P(x) / Q(x) = x$
- $P(x) \bmod Q(x) = x + 1$

$$\begin{array}{l} x^3 + x^2 \quad +1 \\ x^3 + x^2 + x \\ \hline \qquad\quad x + 1 \end{array} \qquad \begin{array}{l|l} & x^2 + x + 1 \\ \hline x & \end{array}$$

Rel. 26.04.2017
© Prinetto - 2017

Outline

- Association bit-vectors ↔ polynomials
- Operations on polynomials
- **Primitive Polynomials**
- Arithmetic Circuits for polynomials
  - Multipliers
  - Dividers
- Autonomous LFSRs
- Applications
  - Pseudo-Random Number Generators
  - Signature Generators
    - Serial
    - Parallel

Rel. 26.04.2017
© Prinetto - 2017

Irreducible Polynomial

A polynomial is said to be ***irreducible*** if it cannot be factored into nontrivial polynomials.

Examples of Irreducible Polynomials:

$x^4 + x^3 + x^2 + x + 1$

$x^4 + x^3 + 1$

$x^4 + x + 1$

$x^2 + x + 1$

$x + 1$

© Prinetto - 2017

Irreducible Polynomial (cont'd)

But $x^2 + 1$ is not irreducible

In fact:

$x = 1$ is a solution, since $1^2 + 1 = 0$

Thus

$x^2 + 1$ can be divided by $x + 1$:

$$x^2 + 1 = (x+1)\,(x+1)$$

Rel. 26.04.2017

© Prinetto - 2017

Primitive polynomials

Un polinomio di grado m è *primitivo* se divide

$$(1 + x^k) \quad \text{dove} \quad k = 2^m - 1$$

ma *non* divide

$$(1 + x^l) \quad \forall\, i < k$$

Primitive polynomials are also irreducible polynomials

Rel. 26.04.2017
© Prinetto - 2017

For any positive integer n, there exists a primitive polynomial of degree n

Problema

Come riconoscere o generare i polinomi primitivi?

Soluzione

Non banale ⇒ si ricorre sempre a delle tavole.

© Prinetto - 2017

n	Esempio di Pol. Prim.	# di Pol. Prim.
1	$x + 1$	1
2	$x^2 + x + 1$	1
3	$x^3 + x + 1$	2
4	$x^4 + x + 1$	2
5	$x^5 + x^2 + 1$	6
6	$x^6 + x + 1$	6
7	$x^7 + x^3 + 1$	18
8	$x^8 + x^4 + x^3 + x^2 + 1$	16
9	$x^9 + x^4 + 1$	48
10	$x^{10} + x^3 + 1$	60
11	$x^{11} + x^2 + 1$	176
12	$x^{12} + x^6 + x^4 + x + 1$	144

Rel. 26.04.2017
© Prinetto - 2017

n	Esempio di Pol. Prim.	# di Pol. Prim.
13	$x^{13} + x^4 + x^3 + x + 1$	630
14	$x^{14} + x^{10} + x^6 + x + 1$	756
15	$x^{15} + x + 1$	1800
16	$x^{16} + x^{12} + x^3 + x + 1$	2048
17	$x^{17} + x^3 + 1$	131072
18	$x^{18} + x^7 + 1$	262144
19	$x^{19} + x^5 + x^2 + x + 1$	524288
20	$x^{20} + x^3 + 1$	1048576
21	$x^{21} + x^2 + 1$	
22	$x^{22} + x + 1$	
23	$x^{23} + x^5 + 1$	
24	$x^{24} + x^7 + x^2 + x + 1$	

Rel. 26.04.2017
© Prinetto - 2017

Polinomi reciproci

Il polinomio *reciproco* A*(x) di un polinomio A(x) di grado m è dato da:

$$A^*(x) = x^m A(x^{-1})$$

Proprietà

Se A(x) rappresenta una sequenza di n bit S_n allora A*(x) rappresenta la sequenza di bit inversa S_n^*.

Esempio

$$A(x) = x^5 + x^3 + x + 1 \quad \Rightarrow \quad S_n = 101011$$
$$A^*(x) = x^5 + x^4 + x^2 + 1 \quad \Rightarrow \quad S_n^* = 110101$$

Rel. 26.04.2017
© Prinetto - 2017

Outline

- Association bit-vectors ↔ polynomials
- Operations on polynomials
- Primitive Polynomials
- Arithmetic Circuits for polynomials
  - Multipliers
  - Dividers
- Autonomous LFSRs
- Applications
  - Pseudo-Random Number Generators
  - Signature Generators
    - Serial
    - Parallel

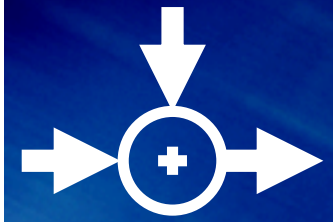
© Prinetto - 2017

Premessa

- **Nel seguito vengono presentate alcune strutture in grado di eseguire le operazioni di moltiplicazione e di divisione tra due polinomi binari, dei quali:**
  - **uno noto di grado r**
  - **uno qualsiasi, i cui coefficienti vengono trasmessi in modo seriale, a partire dal bit più significativo.**

© Prinetto - 2017

Dispositivi logici utilizzati



sommatore modulo 2



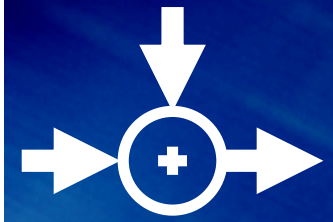
dispositivo in grado di memorizzare un bit



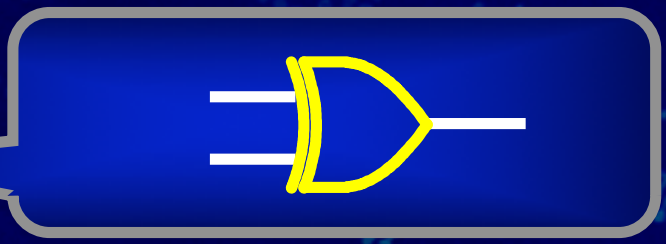
dispositivo in grado di moltiplicare per b

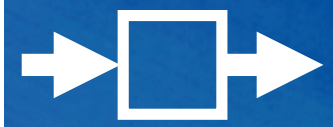
© Prinetto - 2017

Dispositivi logici utilizzati

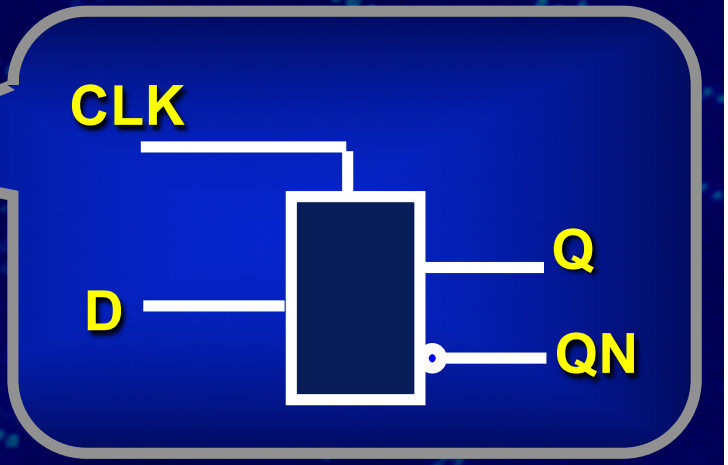


sommatore modulo 2



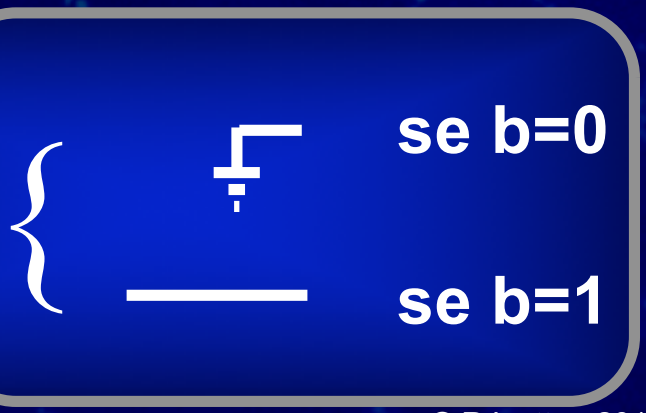


dispositivo in grado di memorizzare un bit





dispositivo in grado di moltiplicare per b



Rel. 26.04.2017
© Prinetto - 2017

Moltiplicazione di due polinomi

Un qualsiasi polinomio

$$a(x) = a_0 + a_1x + \ldots + a_{k-1}x^{k-1} + a_kx^k$$

può essere moltiplicato per un dato polinomio fisso

$$h(x) = h_0 + h_1x + \ldots + h_{r-1}\ x^{r-1} + h_r\ x^r$$

tramite uno qualsiasi dei due circuiti seguenti.

Rel. 26.04.2017
© Prinetto - 2017

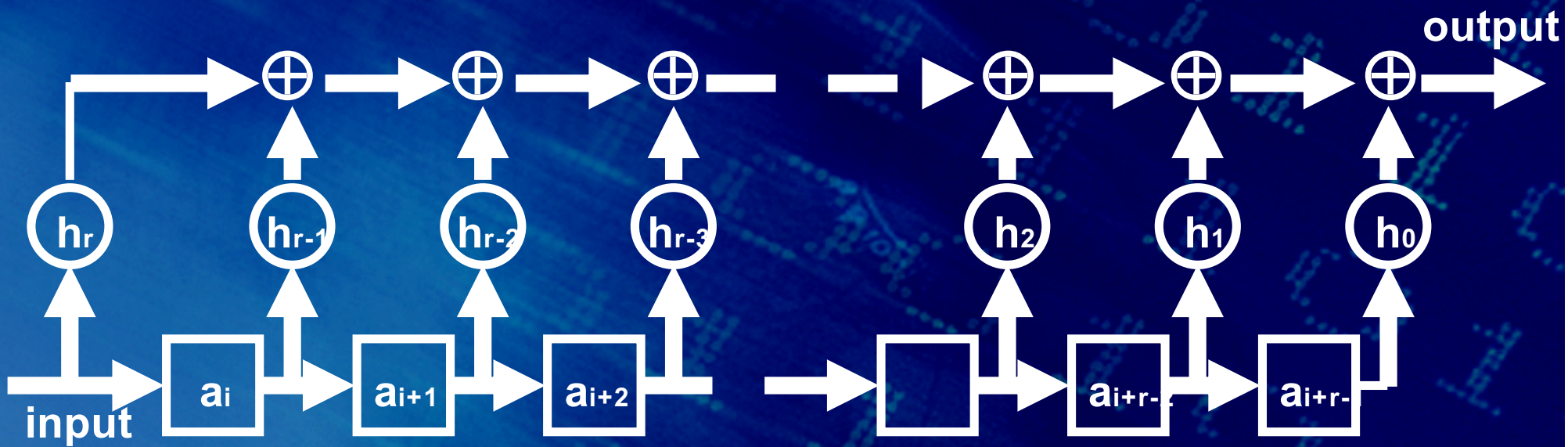
Moltiplicazione di due polinomi
(segue)

Il prodotto vale:

$$
\begin{aligned}
a(x) \cdot h(x) \quad &= a_0 h_0 + \\
&+ (a_0 h_1 + a_1 h_0)\, x + \\
&+ (a_0 h_2 + a_1 h_1 + a_2 h_0)\, x^2 + \\
&+ \ldots + \\
&+ (a_{k-2} h_r + a_{k-1} h_{r-1} + a_k h_{r-2})\, x^{k+r-2} + \\
&+ (a_{k-1} h_r + a_k h_{r-1})\, x^{k+r-1} + \\
&+ a_k h_r\, x^{k+r}
\end{aligned}
$$

Rel. 26.04.2017
© Prinetto - 2017

Prima soluzione



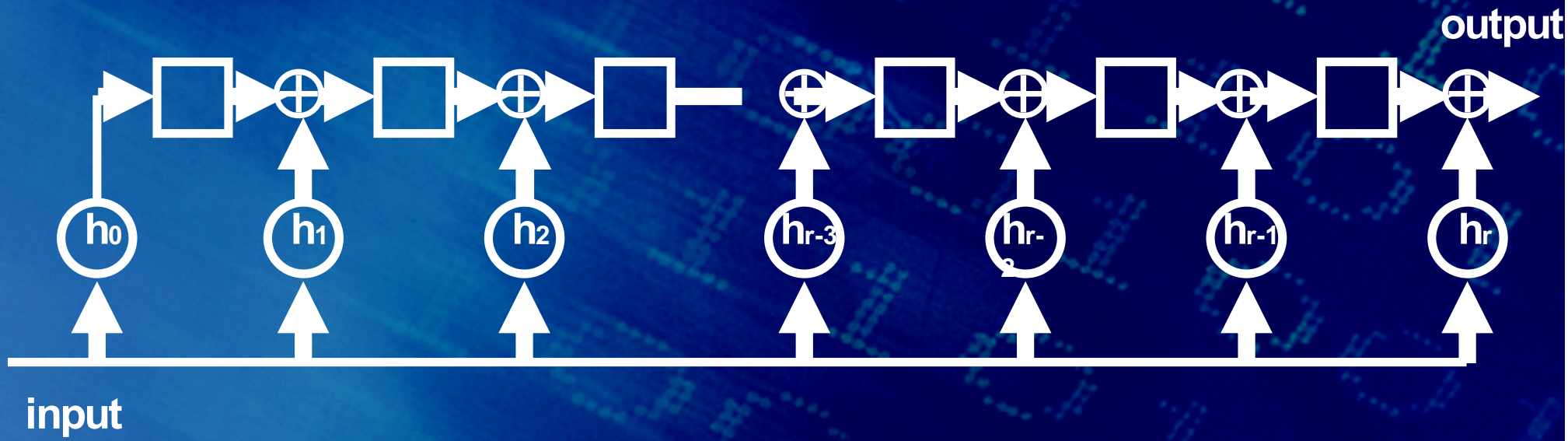
© Prinetto - 2017

Andamento temporale

input	flip flop	output
	0 0 0…0 0 0	
a_k	0 0 0…0 0 0	$a_k h_r$
a_{k-1}	a_k 0 0…0 0 0	$a_{k-1} h_r + a_k h_{r-1}$
a_{k-2}	a_{k-1} a_k 0…0 0 0	$a_{k-2} h_r + a_{k-1} h_{r-1} + a_k h_{r-2}$
0	0 0 0 …0 $a_0 a_1$	$a_0 h_1 + a_1 h_0$
0	0 0 0 …0 0 a_0	$a_0 h_0$

Rel. 26.04.2017
© Prinetto - 2017

Seconda soluzione



© Prinetto - 2017

Andamento temporale

input	flip flop	output
	0 0 0 … 0 0	$a_k h_r$
a_k	$a_k h_0$, $a_k h_1$, $a_k h_2$, … ,$a_k h_{r-2}$, $a_k h_{r-1}$	$a_{k-1} h_r + a_k h_{r-1}$

© Prinetto - 2017

Esempio

Progettare un circuito che moltiplichi per

$$h(x) = 1 + x^3 + x^4 + x^5 + x^6$$

© Prinetto - 2017

Esempio

Progettare un circuito che moltiplichi per

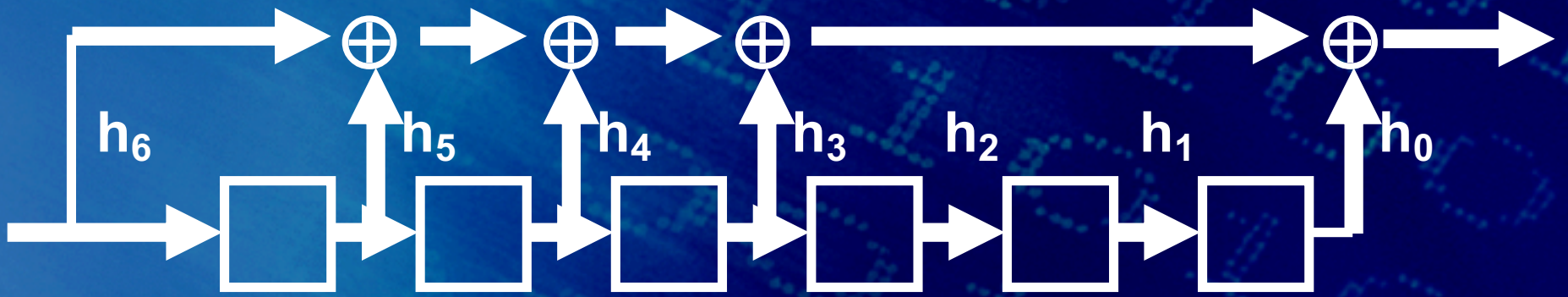
$$h(x) = 1 + x^3 + x^4 + x^5 + x^6$$

$h_0 = 1$
$h_1 = 0$
$h_2 = 0$
$h_3 = 1$
$h_4 = 1$
$h_5 = 1$
$h_6 = 1$

© Prinetto - 2017

Prima soluzione

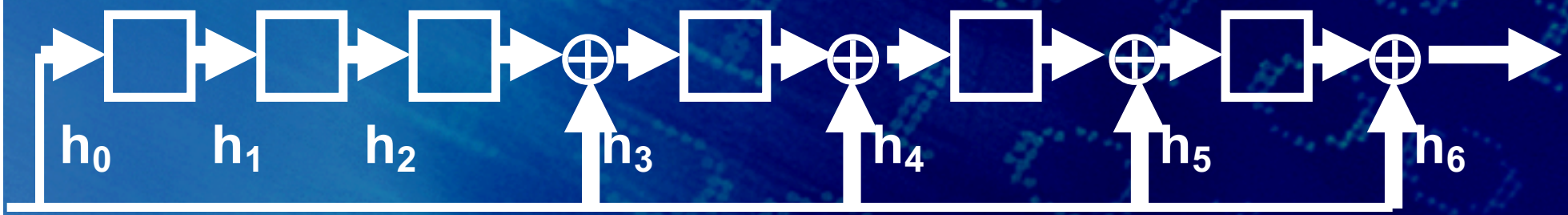
$h_0 = 1$
$h_1 = 0$
$h_2 = 0$
$h_3 = 1$
$h_4 = 1$
$h_5 = 1$
$h_6 = 1$



Rel. 26.04.2017
© Prinetto - 2017

Seconda soluzione

$h_0 = 1$
$h_1 = 0$
$h_2 = 0$
$h_3 = 1$
$h_4 = 1$
$h_5 = 1$
$h_6 = 1$



© Prinetto - 2017

Divisione di due polinomi

Per dividere un polinomio qualsiasi

$$d(x) = d_0 + d_1 x + d_2 x^2 + \ldots + d_n x^n$$

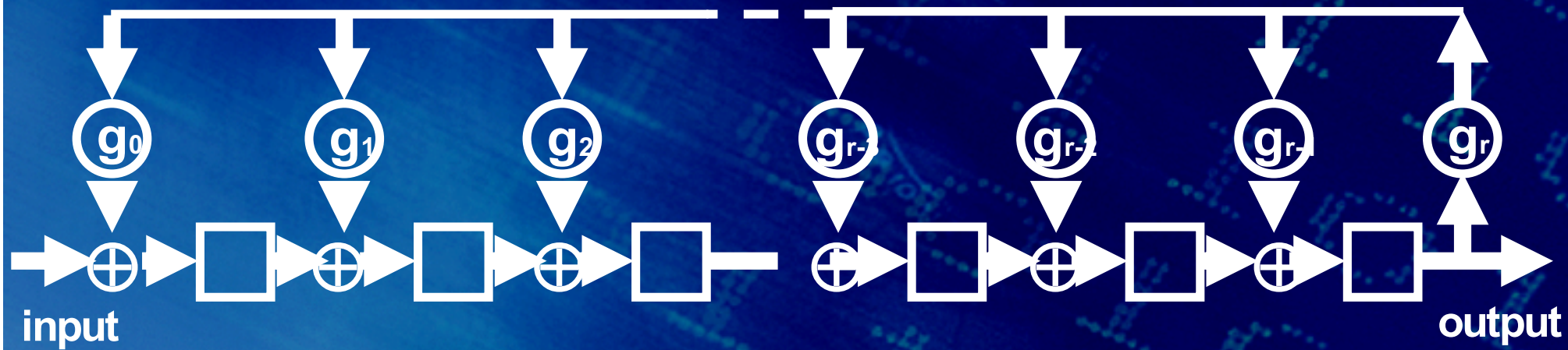
per un polinomio fisso

$$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots + g_{r-1} x^{r-1} + g_r x^r$$

si può utilizzare uno dei seguenti circuiti.

Rel. 26.04.2017
© Prinetto - 2017

Prima soluzione



I flip flop devono essere inizialmente azzerati.

© Prinetto - 2017

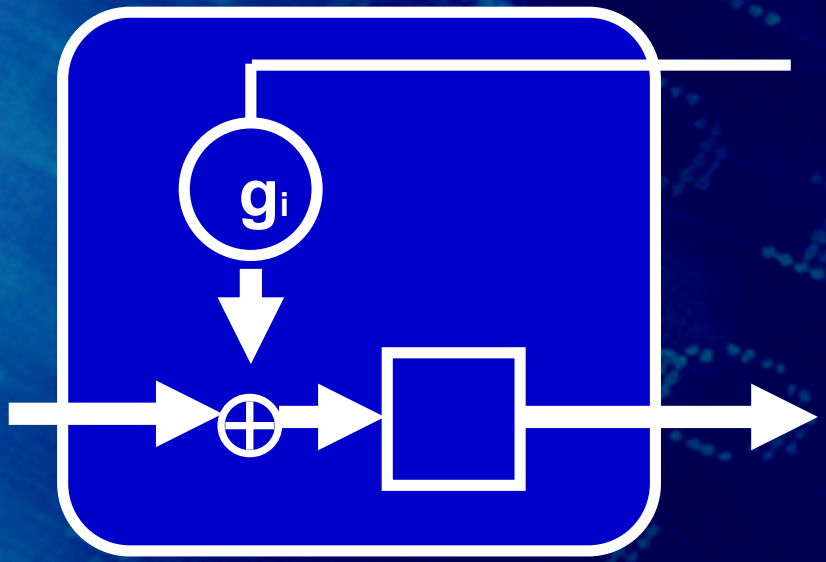
Proprietà

- Durante le operazioni, i bit forniti in uscita costituiscono il *quoziente* della divisione
- Al termine delle operazioni, nel registro è memorizzato il *resto* della divisione

© Prinetto - 2017

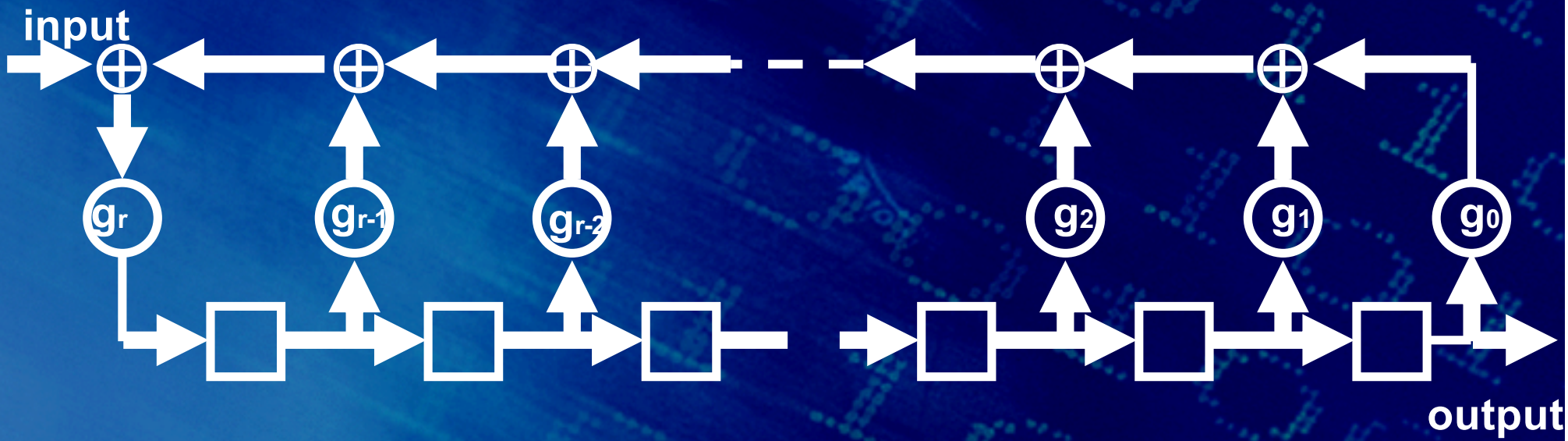
Realizzazione modulare

Questa prima soluzione prende il nome di *realizzazione modulare* in quanto è una cascata di m-1 celle del tipo:



© Prinetto - 2017

Seconda soluzione

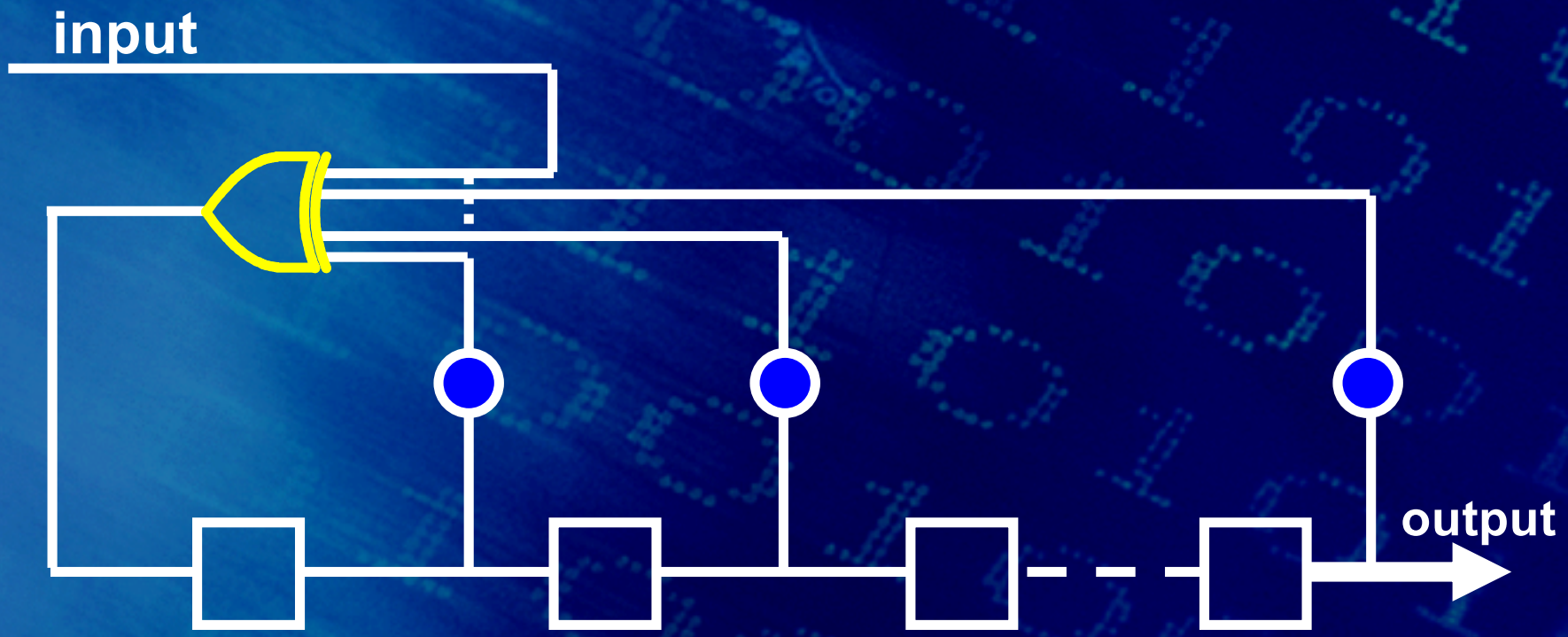


I flip flop devono essere inizialmente azzerati.

© Prinetto - 2017

Realizzazione standard

La forma più generale, che prende di solito il nome di *realizzazione standard*, è la seguente:



© Prinetto - 2017

Esempio

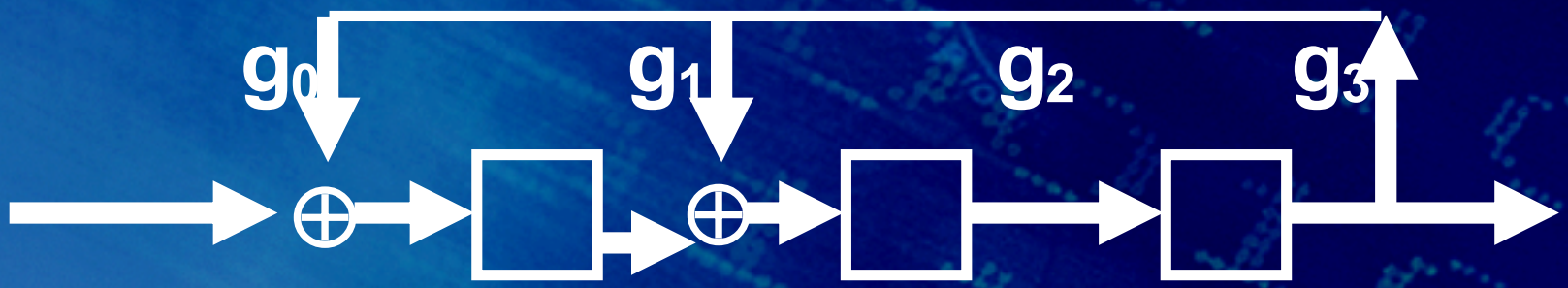
Progettare un dispositivo che divida per
$g(x) = 1 + x + x^3$

$g_0 = 1$
$g_1 = 1$
$g_2 = 0$
$g_3 = 1$

© Prinetto - 2017

$g_0 = 1$
$g_1 = 1$
$g_2 = 0$
$g_3 = 1$

Soluzione modulare



© Prinetto - 2017

$g_0 = 1$
$g_1 = 1$
$g_2 = 0$
$g_3 = 1$

Soluzione standard



© Prinetto - 2017

LFSR

Sia la struttura modulare sia quella standard, essendo composte esclusivamente da:

- **ritardatori unitari (Flip Flop di tipo D)**
- **sommatori modulo 2 (Exor)**
- **moltiplicatori modulo 2 (Shift)**

vengono solitamente chiamate *LFSR* (Linear Feedback Shift Resister).

Rel. 26.04.2017
© Prinetto - 2017

LFSR

Si può dimostrare che qualsiasi circuito formato solo da questi elementi è *lineare*, ossia soddisfa il principio di sovrapposizione:

$$f(i_1 + i_2) = f(i_1) + f(i_2)$$

- **moltiplicatori modulo 2 (Shift)**

vengono solitamente chiamate *LFSR* (Linear Feedback Shift Resister).

Rel. 26.04.2017
© Prinetto - 2017

Analisi comparativa

Area massima

struttura	Flip Flop	exor
standard	r	1 exor a r ingressi
modulare	r	(r-1) exor a 2 ingressi

Ritardo

Modulare: ritardo di un exor a 2 ingressi

standard: ritardo di un exor a (r-1) ingressi.

Rel. 26.04.2017

© Prinetto - 2017

Outline

- Association bit-vectors ↔ polynomials
- Operations on polynomials
- Primitive Polynomials
- Arithmetic Circuits for polynomials
  - Multipliers
  - Dividers
- **Autonomous LFSRs**
- Applications
  - Pseudo-Random Number Generators
  - Signature Generators
    - Serial
    - Parallel

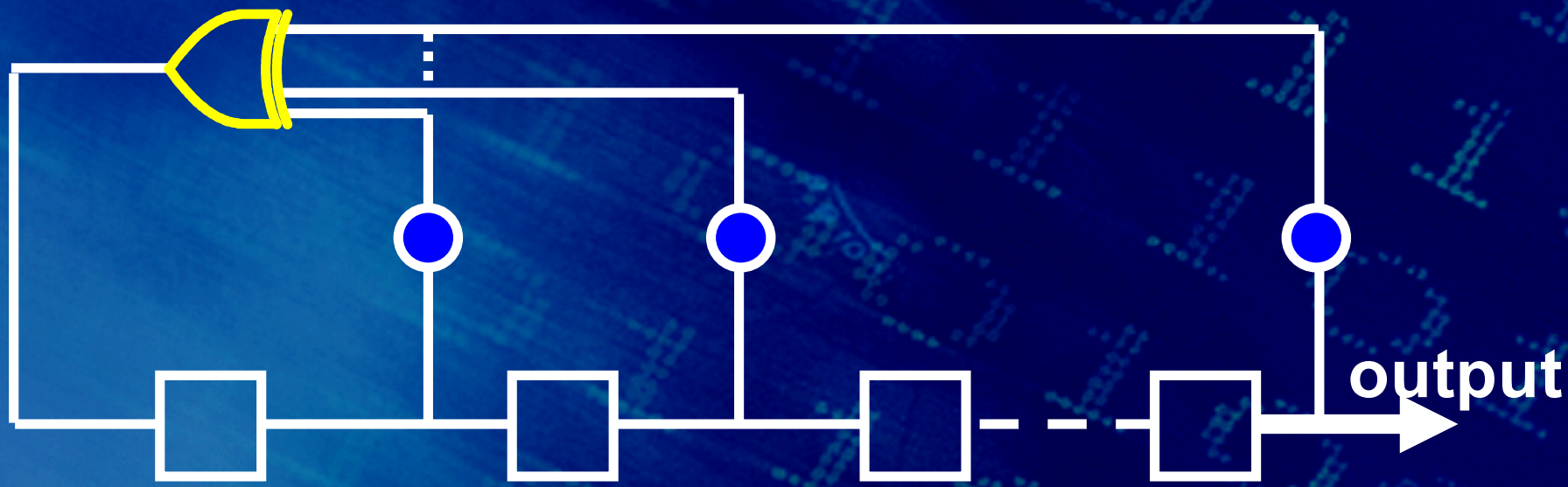
Rel. 26.04.2017
© Prinetto - 2017

ALFSRs

- Gli LFSR privi di ingressi esterni vengono detti *ALFSR (Autonomus LFSR).*
- Analogamente a quanto visto per i circuiti che eseguono la divisione tra polinomi, anche gli ALFSR possono essere realizzati:
  - *in modo standard*
  - *in modo modulare.*

Rel. 26.04.2017
© Prinetto - 2017

Esempio di ALFSR in forma standard



© Prinetto - 2017

Spazio degli stati di un ALFSR

Un ALFSR di n bit è una macchina a stati finiti che può raggiungere, al più,

$$2^n - 1$$

stati (lo stato 00…0 è detto *stato assorbente* e, se raggiunto, non viene più abbandonato).

© Prinetto - 2017

Teorema

Da un ALFSR di n bit, cambiando opportunamente il polinomio caratteristico, è possibile ottenere tutti i periodi compresi tra 1 e 2^n - 1.

© Prinetto - 2017

ALFSR a lunghezza massima

Un ALFSR di n bit in grado di raggiungere tutti i 2^n - 1 stati possibili viene detto *a lunghezza massima*

Rel. 26.04.2017
© Prinetto - 2017

Teorema

Il polinomio caratteristico di un ALFSR a lunghezza massima è un *polinomio primitivo*

© Prinetto - 2017

Outline

- Association bit-vectors ↔ polynomials
- Operations on polynomials
- Primitive Polynomials
- Arithmetic Circuits for polynomials
  - Multipliers
  - Dividers
- Autonomous LFSRs
- Applications
  - Pseudo-Random Number Generators
  - Signature Generators
    - Serial
    - Parallel

© Prinetto - 2017

Caratteristiche delle Sequenze di Uscita degli ALFSR a lunghezza massima

- **Partendo da uno stato diverso da 00…0, vengono raggiunti *tutti* i possibili stati prima di avere ripetizioni**
- **Se una finestra di ampiezza *n* è fatta scorrere sulla sequenza degli *m* bit in uscita, ognuna delle 2^n - 1 possibili *n*-uple di bit è selezionata una ed una sola volta**
- **Il numero degli 1 in una qualsiasi sequenza completa differisce da quello degli 0 al massimo per un'unità**

Rel. 26.04.2017
© Prinetto - 2017

Caratteristiche delle Sequenze di Uscita degli ALFSR a lunghezza massima (segue)

- **In ogni sequenza completa di *m* bit:**
  - **vi sono tante sottosequenze di 1 quante di 0**
  - **metà delle sottosequenze ha lunghezza 1, un quarto ha lunghezza 2, un ottavo ha lunghezza 3, e così via**
  - **vi sono (*m* + 1) / 2 transizioni (da 0 a 1 o vice-versa).**

© Prinetto - 2017

Conclusione

- **Le sequenze di bit generate da un ALFSR a lunghezza massima sono *pseudo-casuali* in quanto:**
  - **sono ripetibili deterministicamente**
  - **hanno proprietà analoghe a quelle delle sequenze definite *casuali*.**



- **Gli ALFSR sono degli ottimi generatori di sequenze pseudo-casuali.**

© Prinetto - 2017

Polinomio Reciproco

- **Due ALFSR,**
  - **uno con polinomio caratteristico P(x)**
  - **uno con polinomio caratteristico P*(x), reciproco di P(x)**
- **producono, in uscita, due sequenze S_n e S_n^* che sono, rispettivamente, l'una l'*inverso* dell'altra.**

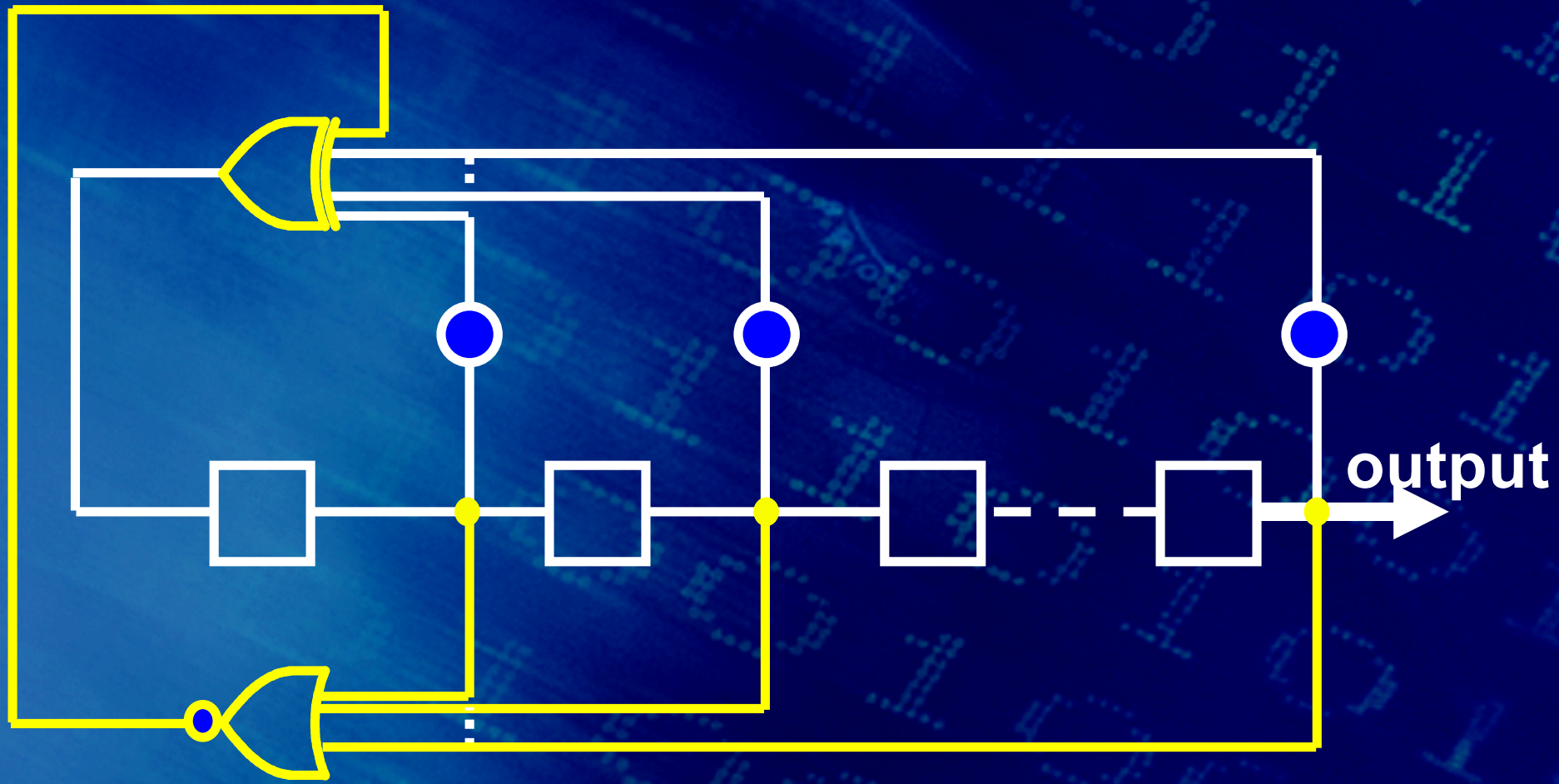
© Prinetto - 2017

ALFSR modificati per eliminare lo stato assorbente

Per garantire che l'ALFSR possa evolvere attraverso tutti i 2^n stati possibili, occorre aggiungere alcune porte.

Rel. 26.04.2017
© Prinetto - 2017

Esempio di ALFSR in forma standard senza stato assorbente



© Prinetto - 2017

Nota

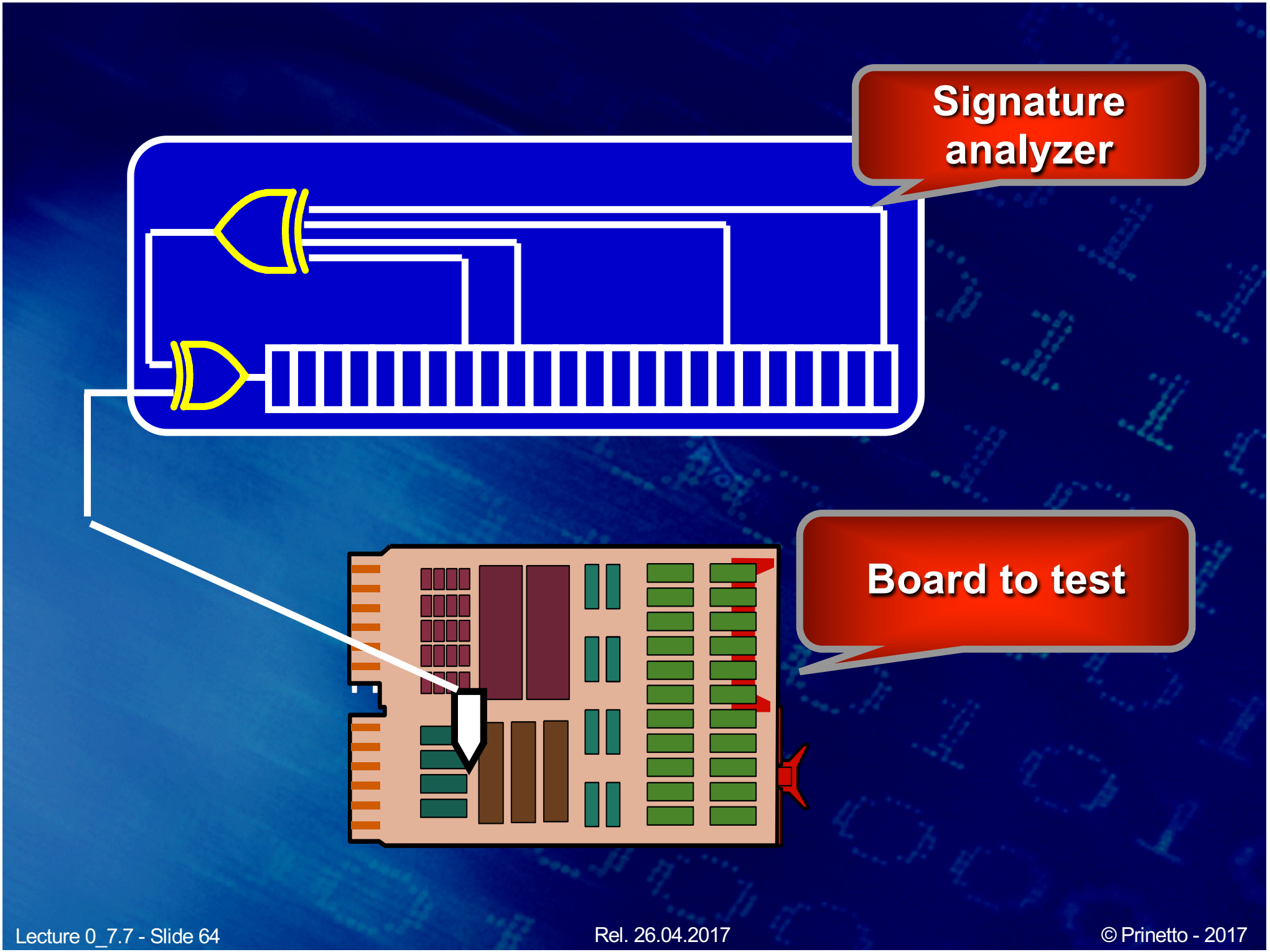
- **Il circuito ottenuto non è più lineare.**

Rel. 26.04.2017
© Prinetto - 2017

LFSRs as compactors

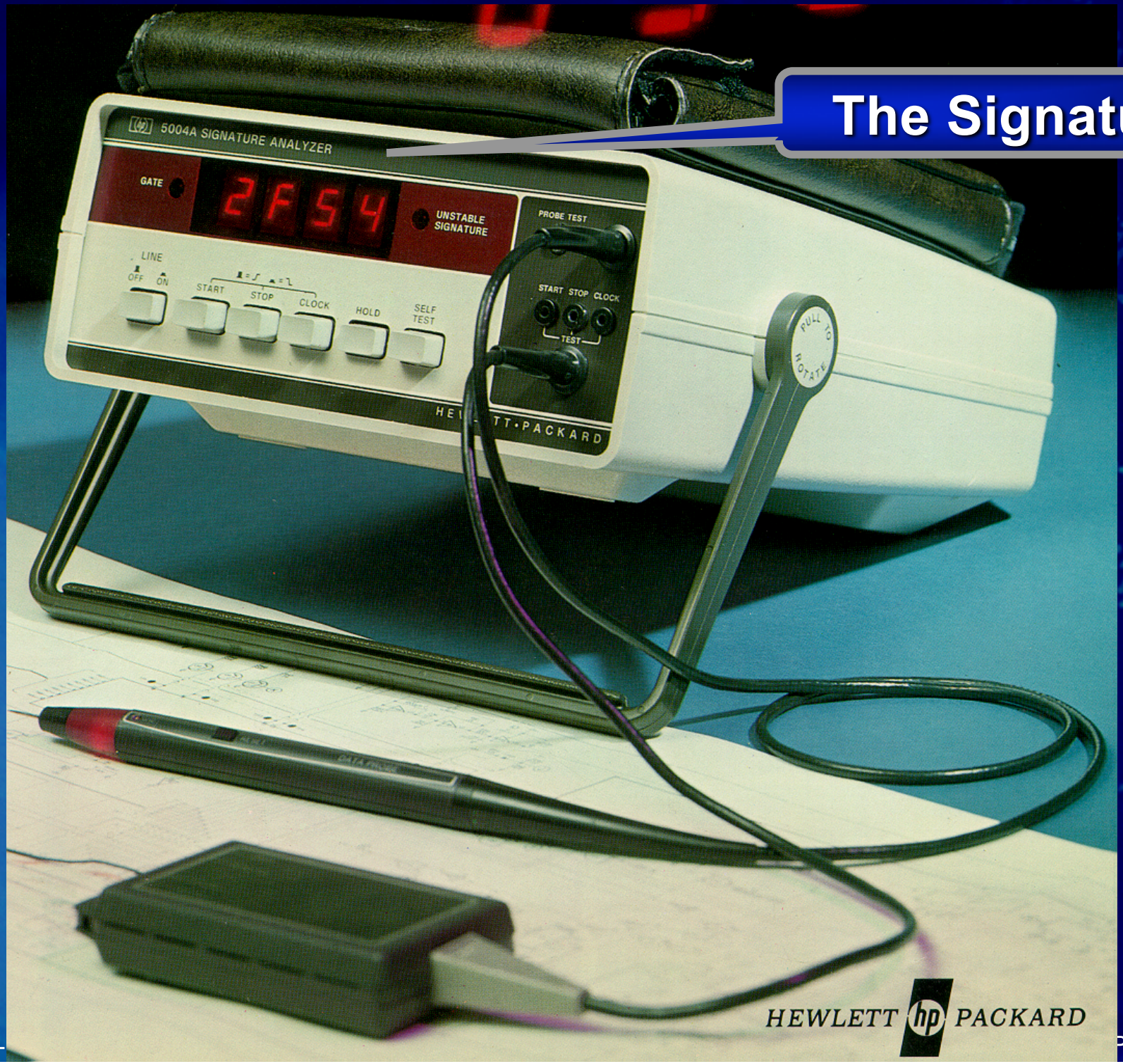
- **LFSRs were first proposed as compactors by HP, that in 1977 used them inside the portable tester 5004A *Signature Analyzer*.**

© Prinetto - 2017



© Prinetto - 2017

The Signature



Outline

- Association bit-vectors ↔ polynomials
- Operations on polynomials
- Primitive Polynomials
- Arithmetic Circuits for polynomials
  - Multipliers
  - Dividers
- Autonomous LFSRs
- Applications
  - Pseudo-Random Number Generators
  - Signature Generators
    - Serial
    - Parallel

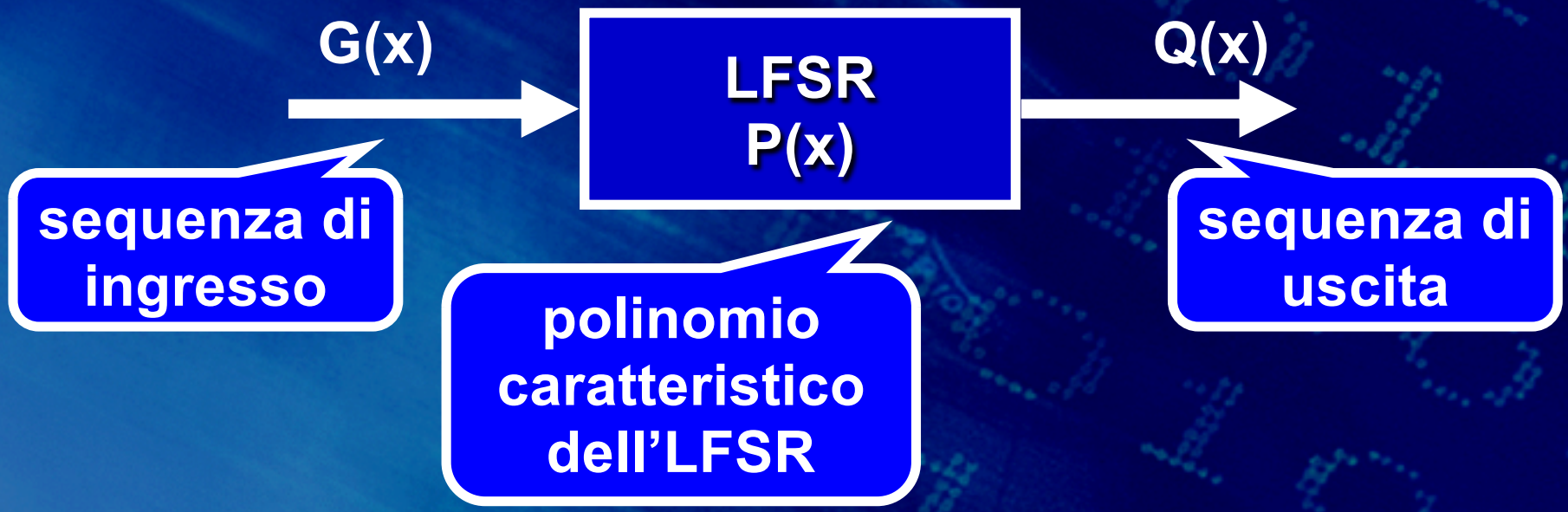
Rel. 26.04.2017
© Prinetto - 2017

Gli LFSR come compattatori

- **Gli LFSR trovano largo impiego come compattatori di sequenze di bit, secondo il seguente approccio:**
  - **considerare la sequenza di bit come coefficienti di un polinomio**
  - **effettuare la divisione per un determinato polinomio divisore**
  - **ignorare il quoziente della divisione**
  - **considerare il resto della divisione come dato compattato (*firma*).**

© Prinetto - 2017

Gli LFSR come compattatori



© Prinetto - 2017

Gli LFSR come compattatori



$$G(x) / P(x) = Q(x) + R(x) / P(x)$$

Rel. 26.04.2017
© Prinetto - 2017

Problema

È possibile che sequenze di ingresso diverse diano origine ad una stessa firma



È importante poter determinare a priori la probabilità di avere perdite di informazione dovute alla compattazione

© Prinetto - 2017

Teorema

Assumendo che:

- **le sequenze di ingresso abbiano lunghezza *m***
- **siano tutte equiprobabili,**

la probabilità di mascheramento P_M di un LFSR con polinomio caratteristico P(x) di grado *n* (vale a dire di *n* bit) è data da

$$P_M = \frac{2^{m-n} - 1}{2^{m-1} - 1} .$$

Rel. 26.04.2017
© Prinetto - 2017

Teorema

Assumendo che:

- **le sequenze di ingresso abbiano lunghezza *m***
- **siano tutte equiprobabili,**

la probabilità di mascheramento P_M di un LFSR con polinomio caratteristico P(x) di grado *n* (vale a dire di *n* bit) è data da

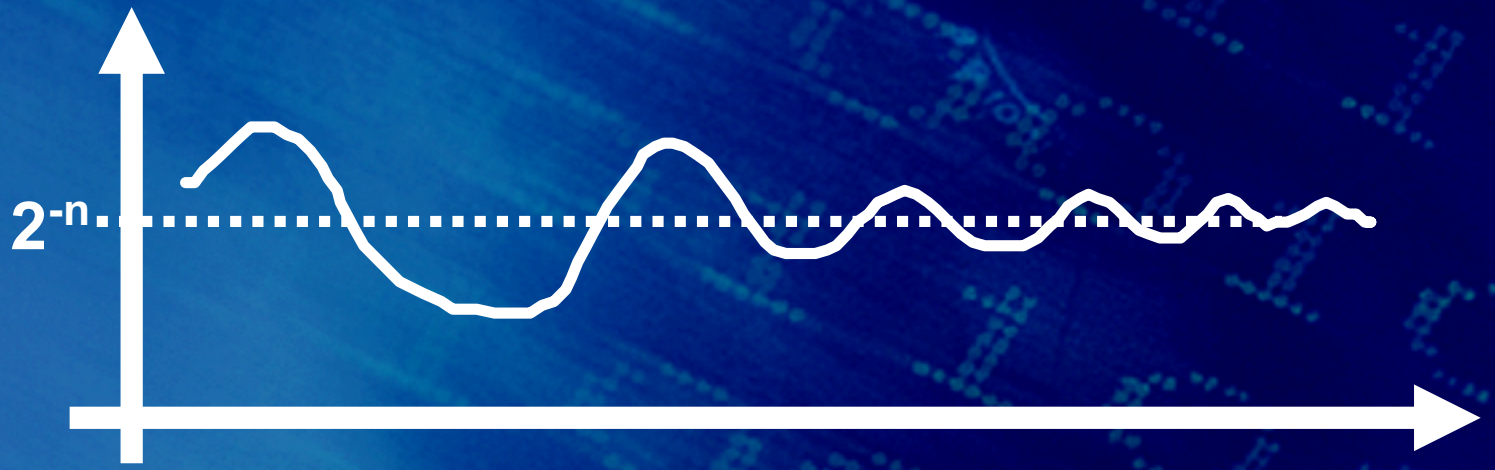
$$P_M = \frac{2^{m-n} - 1}{2^{m-1} - 1} .$$

$$m \gg 1 \quad \Rightarrow \quad P_M \rightarrow 2^{-n}$$

Rel. 26.04.2017
© Prinetto - 2017

Nota

- **Nel caso in cui le sequenze non siano tutte equiprobabili, si ha un comportamento del tipo:**



- **ma il comportamento asintotico è ancora : 2^{-n}**

© Prinetto - 2017

Approccio tipico

- **Per stabilire come debba essere fatto il polinomio divisore P(x) occorre fare delle ipotesi sul tipo di errori che si intendono rilevare.**
- **In pratica si opera nel modo seguente:**
  - **si determina una caratterizzazione probabilistica delle sequenze errate**
  - **si sceglie il P(x) ottimale in funzione di tali distribuzioni probabilistiche.**

© Prinetto - 2017

Approccio pratico

In pratica si utilizzano *polinomi primitivi*

I più utilizzati sono:

$1 + x^5 + x^{12} + x^{16}$: CCITT-16, usato nel protocollo SDLC

$1 + x^7 + x^9 + x^{12} + x^{16}$: usato nella Signature Analysis

$1 + x^2 + x^{15} + x^{16}$: CRC-16, usato nel protocollo BSC

© Prinetto - 2017

È in grado di rilevare:

- tutti gli errori singoli e doppi
- tutti gli errori su un numero dispari di bit
- tutti gli errori di tipo burst di al più 16 bit
- il 99.997 % degli errori di tipo burst di 17 bit
- il 99.998 % degli errori di tipo burst di 18 bit

$1 + x^5 + x^{12} + x$: CCITT-16, usato nel protocollo SDLC

$1 + x^7 + x^9 + x^{12} + x^{16}$: usato nella Signature Analysis

$1 + x^2 + x^{15} + x^{16}$: CRC-16, usato nel protocollo BSC

© Prinetto - 2017

Outline

- Association bit-vectors ↔ polynomials
- Operations on polynomials
- Primitive Polynomials
- Arithmetic Circuits for polynomials
  - Multipliers
  - Dividers
- Autonomous LFSRs
- Applications
  - Pseudo-Random Number Generators
  - Signature Generators
    - Serial
    - Parallel

Rel. 26.04.2017
© Prinetto - 2017

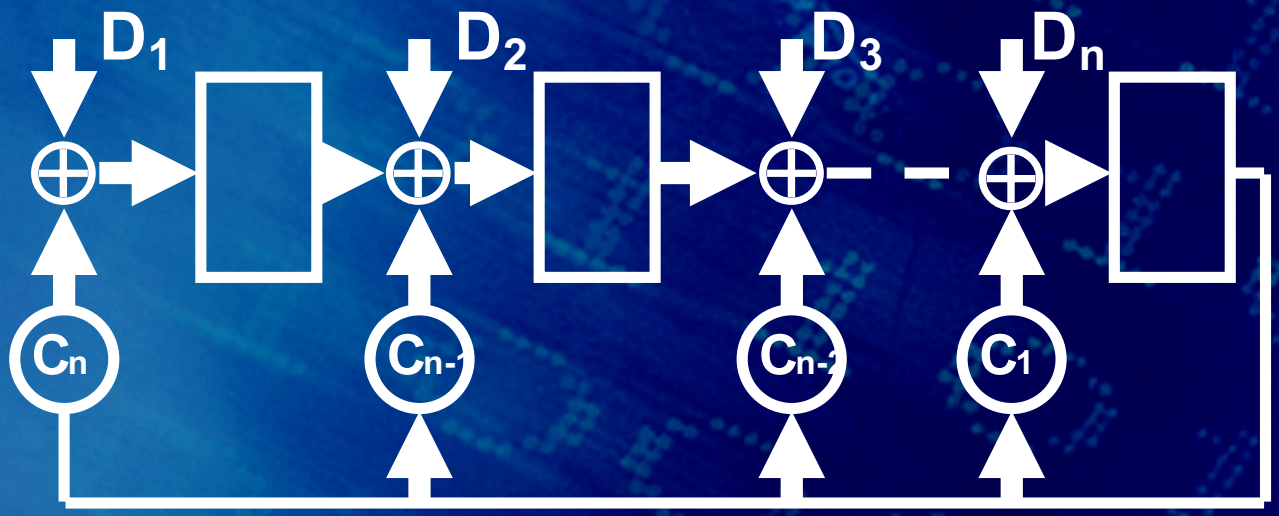
Circuiti multiuscita

- **Le tecniche di compattazione viste sono applicabili anche a circuiti a multiuscita**

Rel. 26.04.2017
© Prinetto - 2017

Soluzione pratica

- Si usa un ***MISR (Multiple Input Signature Register)***:



© Prinetto - 2017

Probabilità di Mascheramento

Si può dimostrare che la probabilità di mascheramento P_M per una sequenza di vettori di lunghezza *m* è data da

$$P_M = \frac{2^{m-1} - 1}{2^{m+n-1} - 1}$$

Se $m >> n$ allora $P_M \cong 2^{-n}$.

Rel. 26.04.2017
© Prinetto - 2017

Probabilità di Cancellazione

Se in una sequenza di vettori vi sono due o più errori, può succedere che un bit errato su un'uscita cancelli l'effetto di un bit errato apparso su un'altra uscita in uno dei colpi di clock precedenti.

La probabilità P_C che questo succeda (detta *Probabilità di Cancellazione*) è data da

$$P_C = 2^{(n-1)(m-1)} - 1 \,/\, 2^{mn} - 1$$

Per $m >> n$ si ha che $P_C \cong 2^{1-m-n}$.

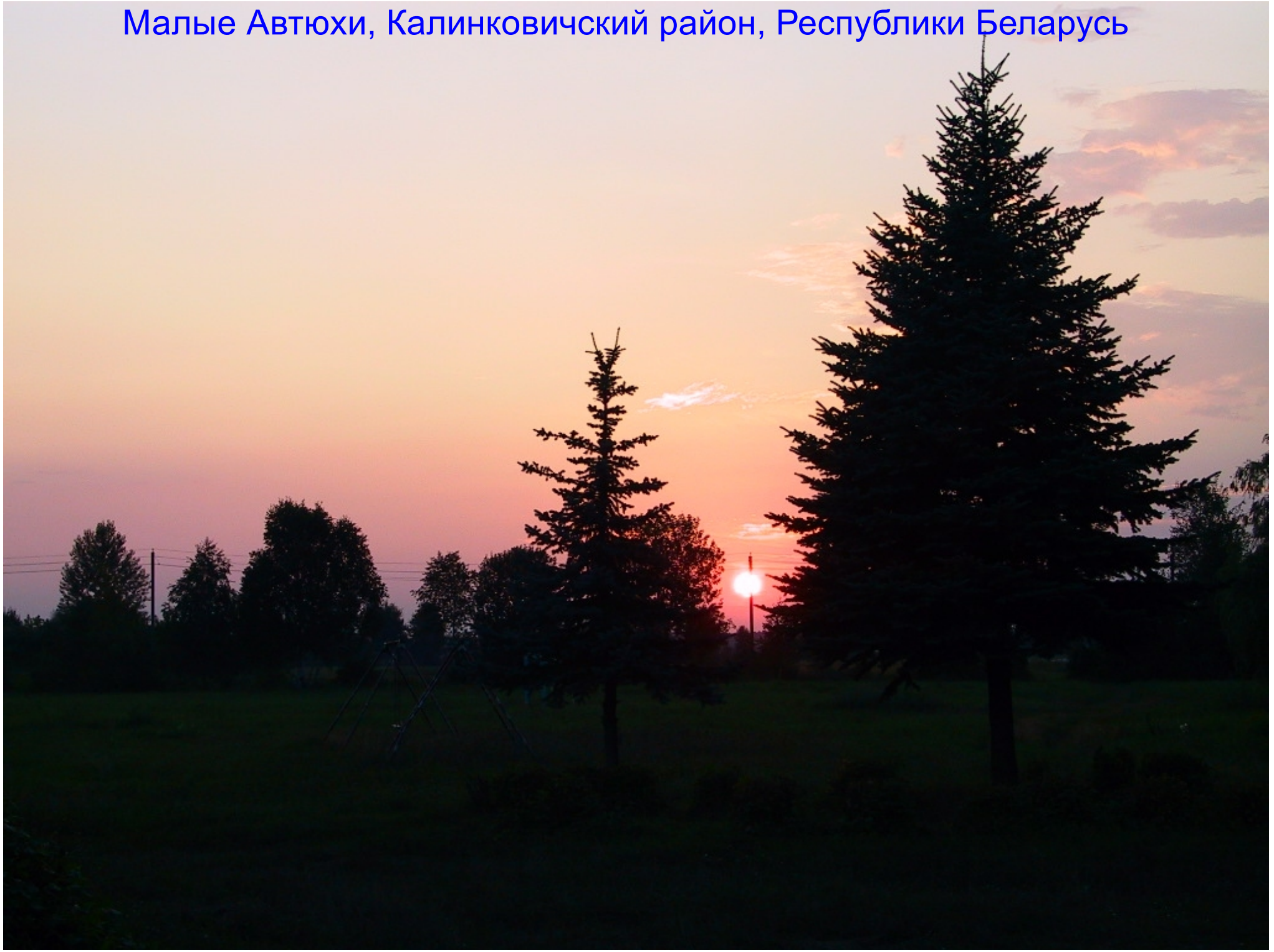
Quindi P_C è ancora minore di P_M .

Rel. 26.04.2017
© Prinetto - 2017

Riferimenti bibliografici

- J.Savir, P.H.Bardell:
  "Built-In Self Test: Milestones and Challenges"
  VLSI Design, Vol.1, n.1, 1993, pp. 23-24
- V.D.Agrawal, C.R.Kime, K.K.Saluja:
  "A tutorial on BIST, Part I: Principles"
  IEEE Design & Test of Computers, Vol 10, n. 1,
  March 1993, pp. 73-82
- V.D.Agrawal, C.R.Kime, K.K.Saluja:
  "A tutorial on BIST, Part II: Applications"
  IEEE Design & Test of Computers, Vol 10, n. 2,
  June 1993, pp. 69-77

© Prinetto - 2017


Малые Автюхи, Калинковичский район, Республики Беларусь