展頻通訊
(Spread Spectrum Communications)

國立清華大學電機系暨通訊工程研究所
蔡育仁
台達館 821 室
Tel: 62210
E-mail: yrtsai@ee.nthu.edu.tw

Chapter 3
Binary Shift-Register Sequences

Contents

- Introduction
- Definitions
- Finite-Field Arithmetic
- Sequence Generator
- State-Machine Representation
- Maximal-Length Sequences
- Gold Codes
- Nonlinear Code Generators
- Some Other Types of Spreading Codes

Introduction

Spreading and Despreading Sequences

- The spreading and despreading waveform $c(t)$ is
  - Usually generated using a **shift register**
  - The contents during each time interval is some **linear or nonlinear** combination of the register contents
- For SS systems to operate efficiently, the **phase** (**code phase**) of the received $c(t - T_d)$ must be **initially determined** and then **tracked** by the receiver
  - Choose $c(t)$ to have a **two-valued auto-correlation** function
    - Correlated: a large value; Un-correlated: a small value
    - The two-valued property is preferred but not necessary
  - **Code acquisition** (initial synchronization) (Chapter 5)
  - **Code tracking** (Chapter 4)

Spreading and Despreading Sequences (Cont.)

- When SS systems is used for **multiple access**
  - Sets of waveforms $c_1(t)$, $c_2(t)$, …, $c_m(t)$ must be found which have **good (small) cross-correlation properties**
  - To assure small **multiple access interference** (MAI)
- When **jamming resistance** is a major concern
  - The waveform $c(t)$ must
    - Have an **extremely long period** and
    - Be **difficult for the jammer to generate**
  - If the jammer can generate and track the waveform $c(t)$, it can **perfectly** jam the signal

Binary Shift-Register Sequences

- A **spreading code** is
  - The output of the binary shift-register generator
  - Having logical value: '**0**' or '**1**'
- A **spreading waveform** (signal) is
  - The function $c(t)$ actually input to the spreading or despreading modulator
  - Taking on values of ± 1 **('0' → +1; '1' → –1)**
- The **ideal** spreading code is
  - An **infinite** sequence of equally likely **random** binary digits
  - It is not feasible for practical applications
  - **Why?**

Binary Shift-Register Sequences (Cont.)

- The **periodic pseudorandom** codes (PN codes) are always employed
  - Periodic spreading codes with **noise-like properties**
    - Easy to generate
    - Good random property
  - **Maximum-length codes** and **Gold codes**

Walsh Codes – An Orthogonal Code Set

- The **cross-correlation** between different codes is **zero**
  - An orthogonal code set
- The code period of Walsh codes must be **a power of 2**
  - The code length must be 2, 4, 8, 16, …
- The number of available Walsh codes for a code period is **limited**
  - If the code length is $P = 2^n$, the number of Walsh codes is P
  - A P-dimension space is spanned by a basis with P orthogonal vectors

Walsh Codes – An Orthogonal Code Set (Cont.)

- The Walsh codes can be easily generated based on a recursive approach
- The initial layer (with a length $2^0 = 1$) is '**0**' $\Rightarrow \mathbf{H}_1 = [0]$
- The n-th layer Walsh codes (with a code length 2^n) can be generated based on the $(n-1)$-th layer Walsh codes

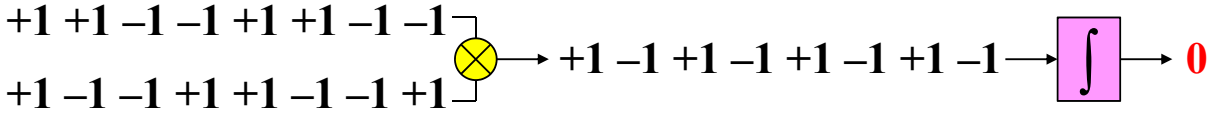
$$\mathbf{H}_{2^n} = \begin{bmatrix} \mathbf{H}_{2^{(n-1)}} & \mathbf{H}_{2^{(n-1)}} \\ \mathbf{H}_{2^{(n-1)}} & \overline{\mathbf{H}_{2^{(n-1)}}} \end{bmatrix}$$

$$\mathbf{H}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}; \quad \mathbf{H}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix};$$

$$\mathbf{H}_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix};$$

Walsh Codes – An Orthogonal Code Set (Cont.)

- The **orthogonality** between any two codes is evaluated based on the corresponding **waveforms**
- Code length $2^3 = 8$: codes **0 0 1 1 0 0 1 1** and **0 1 1 0 0 1 1 0**
  - The corresponding waveforms:



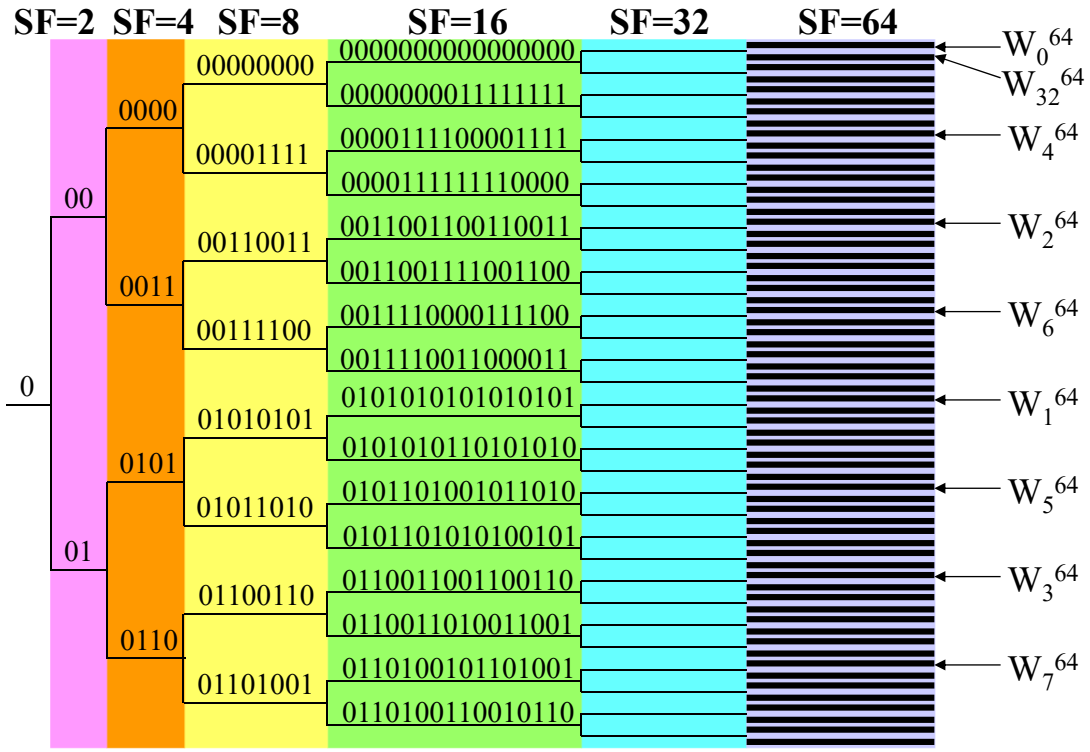
- One of the critical problems of using Walsh codes is
  - The **orthogonality** between any two codes may be **destroyed** when the two codes are **not** perfectly **synchronized**

Synchronous	Asynchronous
+1 +1 –1 –1 +1 +1 –1 –1	+1 +1 –1 –1 +1 +1 –1 –1
+1 –1 –1 +1 +1 –1 –1 +1	+1 –1 –1 +1 +1 –1 –1 +1 **+1**

Variable Length Walsh Codes

- In 3G CDMA systems, the Walsh codes are used for orthogonal **channelization**
  - **Orthogonal variable spreading factor** (OVSF) codes
  - Each code corresponds to a **code channel**
- The chip rate (spreading code rate) is **fixed** for different codes
- The **spreading factor** is the number of chips per symbol
  - It can be 2, 4, 8, 16, 32, or 64
  - Use the Walsh codes with code length 2, 4, 8, 16, 32, or 64
  - A **smaller** spreading factor implies a **higher** symbol rate
- Different codes of the same code length are **orthogonal**
- The number of available codes is **limited**
  - The code resource is **limited**

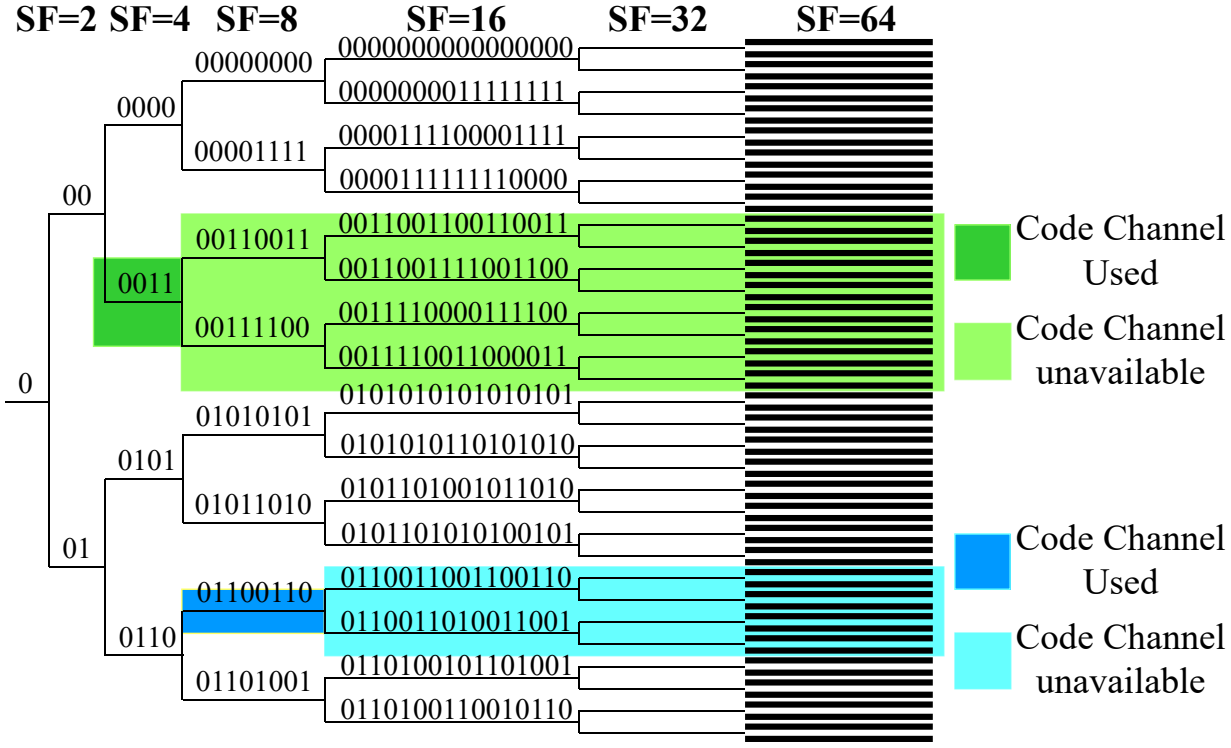
Variable Length Walsh Codes (Cont.)



Variable Length Walsh Codes (Cont.)

- Different codes with the **ancestor-descendant relationship** are **not orthogonal**
- If the OVSF code **0 0 1 1** (with code length 4) is used
  - The spreading sequence is **0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1** …
- If the OVSF code **0 0 1 1 1 1 0 0** (with code length 8) is used
  - The spreading sequence is **0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0** …
- There is strong **mutual interference** between the two OVSF codes '**0 0 1 1**' and '**0 0 1 1 1 1 0 0**'
- If an **ancestor** code is used, the use of the corresponding **descendant** codes is forbidden

Variable Length Walsh Codes (Cont.)



Definitions

Definitions & Mathematical Background

- A spreading code is a **periodic** sequence of ones and zeros with a period N
- A sequence of binary digits …, b_{-2}, b_{-1}, b_0, b_1, b_2, … from the alphabet {0, 1} can be represented as a **polynomial**

$$b(D) = \cdots + b_{-2}D^{-2} + b_{-1}D^{-1} + b_0 + b_1D^1 + b_2D^2 + \cdots$$

  - Because the code is periodic

$$b_n = b_{N+n} \quad \text{for any } n$$

- The spreading waveform $c(t)$ is periodic with a period $T = NT_c$

$$c(t) = \sum_{n=-\infty}^{\infty} a_n p(t - nT_c)$$

- $a_n = (-1)^{b_n} \Rightarrow a_n = \pm 1$, and $p(t)$ is a unit pulse **between 0 and T_c**
  - '0' $\rightarrow +1$; '1' $\rightarrow -1$

Definitions – Correlation

- The autocorrelation function of $c(t)$ is

$$R_c(\tau) = \frac{1}{T}\int_0^T c(t)c(t+\tau)\,dt$$

  - $R_c(\tau)$ is also **periodic** with a period T
- The cross-correlation function of $c(t)$ and $c'(t)$ is

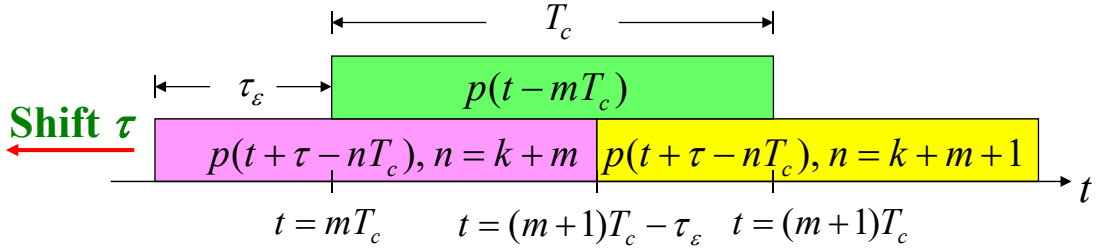
$$R_{cc'}(\tau) = \frac{1}{T}\int_0^T c(t)c'(t+\tau)\,dt$$

  - If both waveforms have the **same** period T, the cross-correlation function is also periodic with a period T
- Substituting $c(t) = \sum_{n=-\infty}^{\infty} a_n p(t - nT_c)$ into $R_{cc'}(\tau)$:

$$R_{cc'}(\tau) = \frac{1}{T}\sum_m \sum_n a_m a'_n \int_0^T p(t - mT_c)\,p(t + \tau - nT_c)\,dt$$

Definitions – Correlation (Cont.)

- Assume that $\tau = kT_c + \tau_\varepsilon$, where $0 \le \tau_\varepsilon < T_c$
  - The integral is nonzero when $p(t - mT_c)$ and $p(t + \tau - nT_c)$ **overlap** $\Rightarrow$ only for $n = k + m$ and $n = k + m + 1$



- Define $\lambda = t - mT_c$:

$$R_{cc'}(\tau) = R_{cc'}(k,\tau_\varepsilon) = \frac{1}{N}\sum_{m=0}^{N-1} a_m a'_{k+m} \frac{1}{T_c}\int_0^{T_c-\tau_\varepsilon} p(\lambda)p(\lambda+\tau_\varepsilon)\,d\lambda$$
$$+\frac{1}{N}\sum_{m=0}^{N-1} a_m a'_{k+m+1} \frac{1}{T_c}\int_{T_c-\tau_\varepsilon}^{T_c} p(\lambda)p(\lambda-T_c+\tau_\varepsilon)\,d\lambda$$

Definitions – Correlation (Cont.)

- The **discrete periodic cross-correlation** function of two codes $b(D)$ and $b'(D)$ is defined as

$$\theta_{bb'}(k) = \frac{1}{N}\sum_{n=0}^{N-1} a_n a'_{n+k}$$

- The cross-correlation function becomes

$$R_{cc'}(k,\tau_\varepsilon) = \left(1 - \frac{\tau_\varepsilon}{T_c}\right)\theta_{bb'}(k) + \frac{\tau_\varepsilon}{T_c}\theta_{bb'}(k+1)$$

  - The discrete periodic cross-correlation functions $\theta_{bb'}(k)$ and $\theta_{bb'}(k+1)$ should be calculated for $R_{cc'}(\tau)$

Definitions – Correlation (Cont.)

- We represent a delay of k time units of the original sequence by $\mathbf{b}(k)$
- The discrete periodic cross-correlation is $\theta_{bb'}(k) = (N_A - N_D)/N$
  - N_A is the number of places in which $\mathbf{b}(0)$ **agrees** $\mathbf{b}'(k)$ $(0 \oplus 0 = 0$ or $1 \oplus 1 = 0)$
  - N_D is the number of places in which $\mathbf{b}(0)$ **disagrees** $\mathbf{b}'(k)$ $(0 \oplus 1 = 1$ or $1 \oplus 0 = 1)$
- The discrete periodic **autocorrelation** function of code $b(D)$ is

$$\theta_b(k) = \frac{1}{N}\sum_{n=0}^{N-1} a_n a_{n+k}$$

- The **autocorrelation** function is

$$R_c(k, \tau_\varepsilon) = \left(1 - \frac{\tau_\varepsilon}{T_c}\right)\theta_b(k) + \frac{\tau_\varepsilon}{T_c}\theta_b(k+1)$$

Finite-Field Arithmetic

Definition of Field

- A **field**, denoted by (**S**, $\oplus$, $\odot$) or **S**, is a set of **at least two** elements with two **binary operations** $\oplus$ and $\odot$, which we call **addition** and **multiplication**, defined on **S** such that the following axioms are satisfied:
  - For example, $\mathbf{S} = \{e_0, e_1, \ldots, e_{M-1}\}$ has M elements

(1) The set is **closed** under the operation $\oplus$:

$$a \oplus b \in \mathbf{S}, \quad \forall a, b \in \mathbf{S}$$

(2) The associative law holds for $\oplus$:

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c, \quad \forall a, b, c \in \mathbf{S}$$

(3) The commutative law holds for $\oplus$:

$$a \oplus b = b \oplus a, \quad \forall a, b \in \mathbf{S}$$

Definition of Field (Cont.)

(4) There is a special (**zero**) element $0 \in \mathbf{S}$, called the **additive identity** of **S**, such that

$$a \oplus 0 = 0 \oplus a = a, \quad \forall a \in \mathbf{S}$$

(5) For each $a \in \mathbf{S}$, there is a corresponding element $-a \in \mathbf{S}$, called the **additive inverse** of a, such that

$$a \oplus (-a) = 0, \quad \forall a \in \mathbf{S}$$

(6) The set is **closed** under the operation $\odot$:

$$a \odot b \in \mathbf{S}, \quad \forall a, b \in \mathbf{S}$$

(7) The associative law holds for $\odot$:

$$a \odot (b \odot c) = (a \odot b) \odot c, \quad \forall a, b, c \in \mathbf{S}$$

(8) The commutative law holds for $\odot$:

$$a \odot b = b \odot a, \quad \forall a, b \in \mathbf{S}$$

Definition of Field (Cont.)

(9) The operation $\odot$ is distributive with respect to $\oplus$:

$$a \odot (b \oplus c) = a \odot b \oplus a \odot c, \quad \forall a, b, c \in \mathbf{S}$$

$$(a \oplus b) \odot c = a \odot c \oplus b \odot c, \quad \forall a, b, c \in \mathbf{S}$$

(10) There is an element $1 \in \mathbf{S}$, called the **multiplicative identity** of **S**, such that $1 \neq 0$ and

$$a \odot 1 = a, \quad \forall a \in \mathbf{S}$$

(11) For each nonzero element $a \in \mathbf{S}$, there is a corresponding element $a^{-1} \in \mathbf{S}$, called the **multiplicative inverse** of a, such that

$$a \odot a^{-1} = 1, \quad \forall a \neq 0, a \in \mathbf{S}$$

- The operations of **subtraction** and **division** are equivalent to
  - **Subtraction**: the addition of the additive inverse
  - **Division**: the multiplication by the multiplicative inverse

Definition of Finite-Field (Cont.)

- A finite field is a field that has a **finite** number of elements in it
  - We call the number as the **order** of the field
- The fundamental results on finite fields was first proved by Évariste Galois

Évariste Galois (October, 1811 – May, 1832) was a French mathematician famous for his contributions to the part of higher algebra now known as **group theory**. His theory provided a solution to the long-standing question of determining **when an algebraic equation can be solved by radicals** (a solution containing square roots, cube roots, and so on but no other nonalgebraic functions). His work laid the fundamental foundations for **Galois theory**. He died in a duel at the age of twenty.



Galois Fields

- There exists a field of order q if and only if q is a **prime power** (i.e., $q = p^m$) with p prime and $m \in \mathbf{N}$.
- Moreover, if q is a prime power, then there is only **one** field of that order.
- The finite fields are often referred to as **Galois fields**: GF(q) for the field having q elements
  - **Prime fields**: having any prime number p of elements
  - **Extension fields**: having any integral power of a prime number p^m of elements
  - For example, the binary number field GF(2) and its extensions GF(2^m)

Prime Fields

- For **prime fields**, addition and multiplication are carried out by using the **modulo-*M*** operations

$\mathbf{S} = \{0, 1\}$

Modulo-2 Addition

+	**0**	**1**
0	0	1
1	1	0

Modulo-2 Multiplication

•	**0**	**1**
0	0	0
1	0	1

$\mathbf{S} = \{0, 1, 2\}$

Modulo-3 Addition

+	**0**	**1**	**2**
0	0	1	2
1	1	2	0
2	2	0	1

Modulo-3 Multiplication

•	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	1

Finite-Field Arithmetic – Addition

- Consider a polynomial of degree m over GF(2)

$$f(D) = f_0 + f_1 D + f_2 D^2 + \cdots + f_m D^m$$

  - where f_j is an element of GF(2)
- The **addition** of $f(D)$ and $g(D)$ yields $h(D)$

$$g(D) = g_0 + g_1 D + g_2 D^2 + \cdots + g_m D^m$$

$$h(D) = h_0 + h_1 D + h_2 D^2 + \cdots + h_m D^m$$

  - where (with **modulo-2** additions)

$$\begin{aligned} h_0 &= f_0 + g_0; \\ h_1 &= f_1 + g_1; \\ h_2 &= f_2 + g_2; \\ &\vdots \\ h_m &= f_m + g_m \end{aligned}$$

Finite-Field Arithmetic – Multiplication

- The **multiplication** of $f(D)$ and $g(D)$ yields $h(D)$

$$h(D) = h_0 + h_1 D + h_2 D^2 + \cdots + h_{2m} D^{2m}$$

  - where (with **modulo-2** additions and multiplications)

$$\begin{aligned} h_0 &= f_0 g_0 \\ h_1 &= f_0 g_1 + f_1 g_0 \\ h_2 &= f_0 g_2 + f_1 g_1 + f_2 g_0 \\ &\vdots \\ h_m &= f_0 g_m + f_1 g_{m-1} + \cdots + f_m g_0 \\ h_{m+1} &= f_1 g_m + f_2 g_{m-1} + \cdots + f_m g_1 \\ &\vdots \\ h_{2m} &= f_m g_m \end{aligned}$$

Finite-Field Arithmetic – Division

- The **division** of one polynomial over GF(2) by another yields a **quotient** $q(D)$ and a **remainder** $r(D)$:

$$f(D)=q(D)g(D)+r(D)$$

- For example, $f(D)=1+D^5;\ \ g(D)=1+D+D^3+D^4$

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

$$\begin{array}{r|l} & D+1 \\ \hline D^4+D^3+D+1 & D^5 \qquad\qquad +1 \\ & D^5+D^4 \qquad +D^2+D \\ \hline & D^4 \qquad +D^2+D+1 \\ & D^4+D^3 \qquad +D+1 \\ \hline & D^3+D^2 \end{array}$$

Finite-Field Arithmetic – Division (Cont.)

- For example, $f(D) = 1 + D^6$

$$\begin{array}{r|l} & D^2+D+1 \\ \hline D^4+D^3+D+1 & D^6 \qquad\qquad +1 \\ & D^6+D^5 \qquad +D^3+D^2 \\ \hline & D^5 \qquad +D^3+D^2 \qquad +1 \\ & D^5+D^4 \qquad +D^2+D \\ \hline & D^4+D^3 \qquad +D+1 \\ & D^4+D^3 \qquad +D+1 \\ \hline & 0 \end{array}$$

$$\begin{aligned} (1+D+D^2)(1+D+D^3+D^4) &= 1+D \qquad +D^3+D^4 \\ &\quad +D+D^2 \qquad +D^4+D^5 \\ &\quad +D^2+D^3 \qquad +D^5+D^6 \\ &= 1 \qquad\qquad\qquad +D^6 = f(D) \end{aligned}$$

Extension Field GF(2^m)

- The extension field has $\mathbf{2^m}$ elements
- Consider **all** the **polynomials** of **degree $m-1$** over GF(2)
  - There are 2^m such polynomials
    - **Degree 0**: 0; 1;
    - **Degree 1**: D; $1+D$;
    - …;
    - **Degree $m-1$**: $1+D^{m-1}$; …; $1+D+D^2+\cdots+D^{m-1}$
  - Each polynomial can be used to represent a single element of the extension field GF(2^m)
- Example of $m = 2$:
  - The extension field contains $2^2 = 4$ elements
  - There are four polynomials of degree 1: 0, 1, D and $1+D$

Extension Field GF(2^m) – Addition

- The **addition** of two elements of GF(2^m):
  - The normal modulo-2 polynomial addition of the two polynomials
  - The addition of any two elements of the field yields **another element** of the field

  $\Rightarrow$ The field is **closed under addition**
  - The **additive identity** element is '0'
  - The **additive inverse** of any element is **the element itself**

$$(D)+(D+1)=1;$$
$$(1)+(D)=D+1;$$
$$(D)+(D)=0;$$
$$(D+1)+(D+1)=0;$$

Extension Field GF(2^m) – Multiplication

- The **multiplication** of two elements of GF(2^m):
  - Must use a special **primitive** **polynomial** of **degree *m***
- A primitive polynomial is a polynomial that **generates all elements** of an **extension field** from a base field.
  - Primitive polynomials are also **irreducible polynomials** ⇒ **Not** the product of any two polynomials of lower degrees
  - An irreducible polynomial **may not** be a primitive polynomial
- A polynomial $h(D)$ of degree m is said to be **primitive** if
  - $h(D)$ **divides** $D^n + 1$
    - where the **smallest** integer of n is $n = 2^m - 1$
  - Primitive polynomials of **any degree *m*** are known to exist

Extension Field GF(2^m) – Multiplication (Cont.)

- The **multiplication** of two elements is defined as
  - The **remainder** of the normal polynomial product divided by the chosen **primitive polynomial**
  - **Modulo-*h*(*D*)** multiplication
  - The multiplication rules **depend on** the chosen primitive polynomial $h(D)$
  - The remainder has degree **at most *m* – 1**, so it is another element of GF(2^m)

  ⇒ The field is **closed** **under multiplication**
  - Polynomial multiplication is associative and commutative

  ⇒ Multiplication in GF(2^m) is also associative and commutative

Extension Field GF(2^m) – Multiplicative Inverse

- Determine the **multiplicative inverse** elements
- Assume the chosen primitive polynomial is

$$h(D)=1+h_1D+h_2D^2+\cdots+h_{m-1}D^{m-1}+D^m$$

- Consider the sequence of nonzero elements of GF(2^m)
  - Beginning with 1

$$\begin{aligned}&1\\&1\cdot D=D;\\&D\cdot D=D^2;\\&D^2\cdot D=D^3;\\&\vdots\end{aligned}$$

Extension Field – Multiplicative Inverse (Cont.)

- For all products where the normal polynomial product has degree **less than *m***
  - The remainder $r(D)$ of modulo-$h(D)$ (modulo-$h(D)$ product) is the **normal polynomial product**

$$\begin{aligned}&1 \text{ modulo-}h(D)=1;\\&D \text{ modulo-}h(D)=D;\\&D^2 \text{ modulo-}h(D)=D^2;\\&D^3 \text{ modulo-}h(D)=D^3;\\&\vdots\\&D^{m-1} \text{ modulo-}h(D)=D^{m-1};\end{aligned}$$

Extension Field – Multiplicative Inverse (Cont.)

- When $\boldsymbol{D^m}$ appear, the modulo-$h(D)$ product is

$$\begin{array}{r} 1 \\ D^m + h_{m-1}D^{m-1} + \cdots + h_2 D^2 + h_1 D + 1 \overline{)\, D^m} \\ \qquad\qquad\qquad\

Extension Field – Multiplicative Inverse (Cont.)

- A **primitive** polynomial of degree m divides $D^{2^m-1}+1$, so that

$$D^{2^m-1}+1 = q(D)h(D) \Rightarrow D^{2^m-1} = q(D)h(D)+1$$

  - The remainder of dividing D^{2^m-1} by $h(D)$ is 1
  - The **smallest** integer n for which $h(D)$ divides $D^n + 1$ is $n = 2^m - 1$
  - The multiplication rule repeats **exactly** $2^m - 1$ distinct elements
    - $D^0 = 1; D^1; D^2; D^3; \ldots; D^{2^m-2}; D^{2^m-1} = 1; D^1; \ldots$
- There are two ways of representing the elements of GF(2^m):
  - A polynomial of degree $m - 1$ over GF(2)
  - **Power of *D***

Extension Field – Multiplicative Inverse (Cont.)

- Using the second representation (power of D), $D^{2^m-1} = 1$ leads the **multiplicative inverse** of any element of GF(2^m)
  - The multiplicative inverse of D^j is

$$\left(D^j\right)^{-1} = 1/D^j = D^{2^m-1}/D^j = D^{2^m-1-j}$$

$$D^j \cdot \left(D^j\right)^{-1} = D^j \cdot D^{2^m-1-j} = D^{2^m-1} = 1$$

- For example,

$$h(D) = D^4 + D + 1$$

$$n = 2^4 - 1 = 15$$

  - Then, we have the multiplicative inverse as follows

Extension Field – Multiplicative Inverse (Cont.)

Element		Inverse Element	
1	1	1	1
D	D	D^{14}	$1+D^3$
D^2	D^2	D^{13}	$1+D^2+D^3$
D^3	D^3	D^{12}	$1+D+D^2+D^3$
D^4	$1+D$	D^{11}	$D+D^2+D^3$
D^5	$D+D^2$	D^{10}	$1+D+D^2$
D^6	D^2+D^3	D^9	$D+D^3$
D^7	$1+D+D^3$	D^8	$1+D^2$
D^8	$1+D^2$	D^7	$1+D+D^3$
D^9	$D+D^3$	D^6	D^2+D^3
D^{10}	$1+D+D^2$	D^5	$D+D^2$
D^{11}	$D+D^2+D^3$	D^4	$1+D$
D^{12}	$1+D+D^2+D^3$	D^3	D^3
D^{13}	$1+D^2+D^3$	D^2	D^2
D^{14}	$1+D^3$	D	D

$$\boxed{\begin{aligned} D^7\left(D^7\right)^{-1} &= D^7 \cdot D^8 \\ &= \left(1+D+D^3\right)\cdot\left(1+D^2\right) \\ &= 1+D+D^2+D^5 \end{aligned}}$$

$$\begin{array}{l} 1+D \qquad\qquad +D^3 \\ \qquad\quad +D^2+D^3+D^5 \\ \hline 1+D+D^2 \qquad\qquad +D^5 \end{array}$$

$$\boxed{\left(1+D+D^2+D^5\right)/\left(1+D+D^4\right)=1}$$

$$\begin{array}{r|l} & D \\ \hline D^4+D+1 & D^5+ \qquad D^2+D+1 \\ & D^5+ \qquad D^2+D \\ \hline & \qquad\qquad\qquad 1 \end{array}$$

Sequence Generator

Sequence Generator Fundamentals

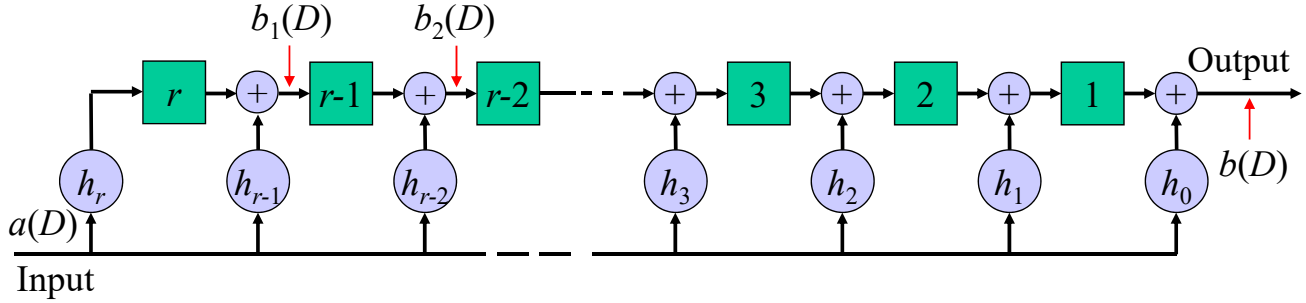
- The **shift registers** are used to generate PN codes
- The shift registers with **feedback** and/or **feedforward** connections can be used to **multiply** and **divide** polynomials over GF(2)
- The sequences are assumed to begin at **time zero**, and a code sequence $a(D)$ contains only **positive powers** of the delay operation D

$$a(D) = \sum_{j=0}^{\infty} a_j D^j$$

- In the logic circuit:
  - If the coefficient is a **'1'**: **represents a connection**
  - If the coefficient is a **'0'**: **represents no connection**
  - Circles containing a **'+'**: **represents modulo-2 adders** or **exclusive-OR** gates

Circuit for Multiplying Polynomials

$$b(D) = h(D)a(D)$$
$$h(D) = h_0 + h_1 D + h_2 D^2 + \cdots + h_r D^r$$



= modulo-2 adder

h_j = modulo-2 multiplication of input by h_j

= single stage shift register

Circuit for Multiplying Polynomials (Cont.)

- The output of the j-th modulo-2 adder **(from left to right** starting with 1), denoted $b_j(D)$, is
  - For $j = 2, \ldots, r$: $b_j(D) = b_{j-1}(D)D + a(D)h_{r-j}$
  - For $j = 1$: $b_1(D) = a(D)Dh_r + a(D)h_{r-1}$
- By iteration:

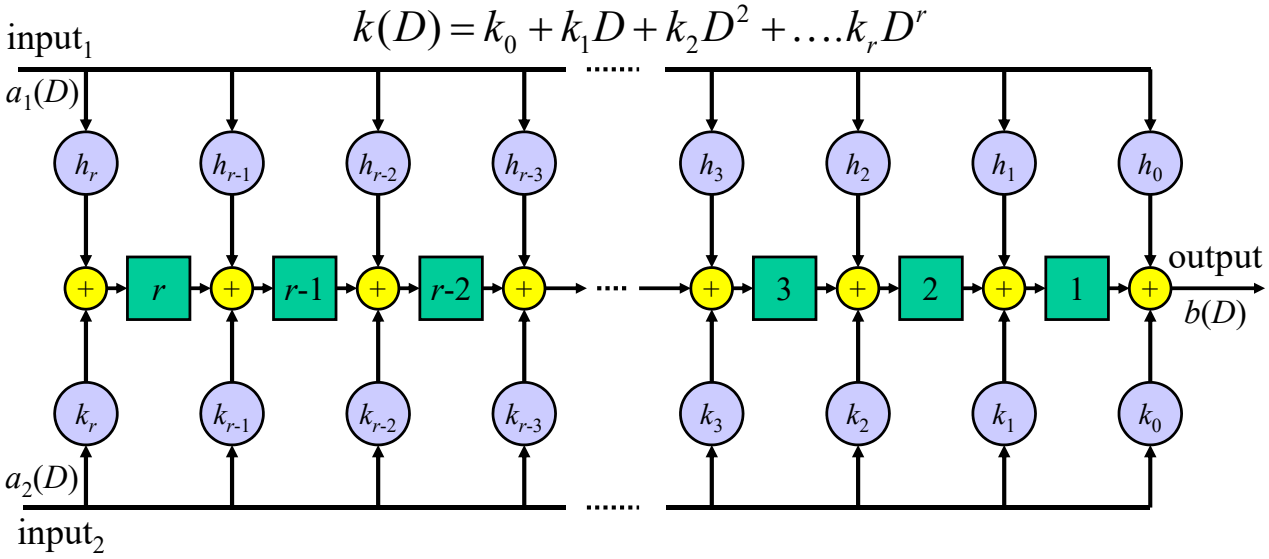
$$\begin{aligned} b(D) = b_r(D) &= b_{r-1}(D)D + a(D)h_0 \\ &= b_{r-2}(D)D^2 + a(D)Dh_1 + a(D)h_0 \\ &\vdots \\ &= b_1(D)D^{r-1} + a(D)D^{r-2}h_{r-2} + \cdots + a(D)h_0 \\ &= a(D)D^r h_r + a(D)D^{r-1}h_{r-1} + \cdots + a(D)h_0 \\ &= \sum_{k=0}^{r}\left[a(D)D^k\right]h_k = h(D)a(D) \end{aligned}$$

  - The circuit performs the **normal polynomial multiplication** of the input sequence $a(D)$ and the transfer polynomial $h(D)$

Circuit for Multiplying Polynomials (Cont.)

- For a two input multiplier:

$$b(D) = h(D)a_1(D) + k(D)a_2(D)$$
$$h(D) = h_0 + h_1D + h_2D^2 + \ldots.h_rD^r$$
$$k(D) = k_0 + k_1D + k_2D^2 + \ldots.k_rD^r$$



Circuit for Multiplying/Dividing Polynomials

- The output sequence: $b(D) = h(D)a_1(D) + k(D)a_2(D)$
  - $h(D)$ is the transfer function for the first input $a_1(D)$
  - $k(D)$ is the transfer function for the second input $a_2(D)$
- Suppose that:
  - $\boldsymbol{k_0 = 0}$ in the second transfer function $k(D)$, and
  - $a_2(D)$ is taken from the output
    - $a_2(D) = b(D)$
- Since $k_0 = 0$, we define $k(D) = g(D) + 1$
  - $g(D)$ is a transfer function with $\boldsymbol{g_0 = 1}$ ($k_0 = g_0 + 1 = 0$)

Circuit for Multiplying/Dividing Polynomials(Cont.)

- The input-output relationship becomes:

$$b(D) = a_1(D)h(D) + b(D)[g(D) + 1]$$

$$b(D) + b(D)g(D) + b(D) = a_1(D)h(D) + b(D)[g(D) + 1] + b(D)g(D) + b(D)$$

$$b(D)g(D) = a_1(D)h(D)$$

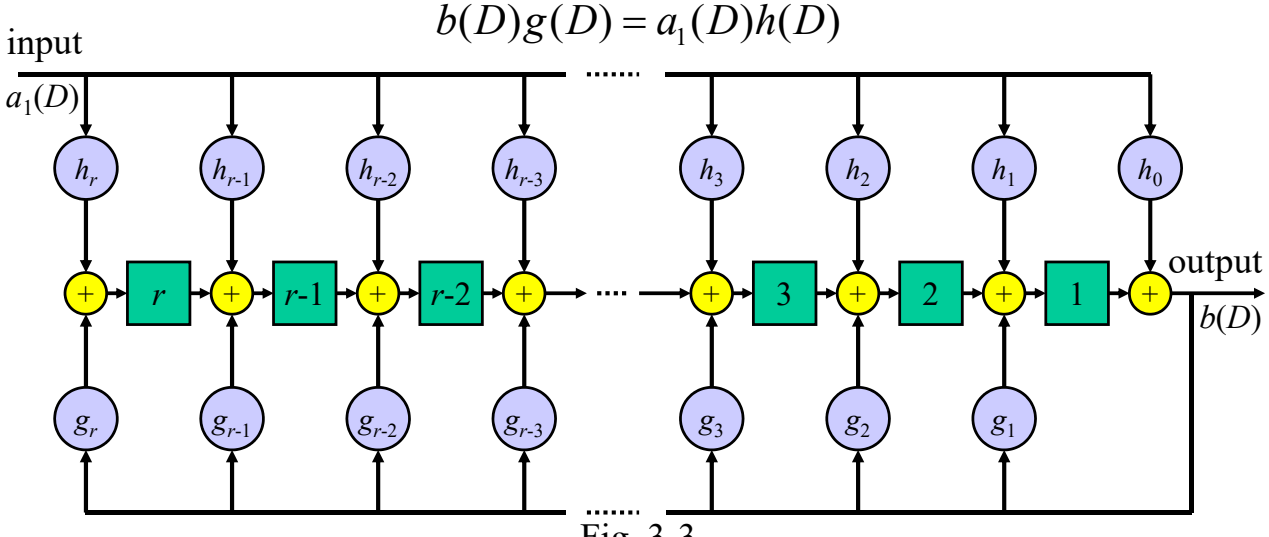


Fig. 3-3

Circuit for Multiplying/Dividing Polynomials(Cont.)

- If a polynomial $c(D)$ (multiplicative inverse of $g(D)$) can be found
  - $g(D)c(D) = 1$
  - Then, we have $b(D) = a_1(D)h(D)c(D)$
- The coefficients of $c(D)$ must satisfy:

$$g_0 c_0 = 1$$

$$\sum_{l=0}^{\min\{j,r\}} g_l c_{j-l} = 0, \quad j = 1, 2, \cdots \quad \text{coefficient of degree } j$$

$$c(D) = 1/g(D)$$

$$b(D)g(D) = a_1(D)h(D)$$

$$b(D)g(D)c(D) = a_1(D)h(D)c(D)$$

$$b(D) = a_1(D)\frac{h(D)}{g(D)}$$

Circuit for Multiplying/Dividing Polynomials(Cont.)

- Consider the polynomial long division of 1 by $g(D)$

$$\begin{array}{r} c(D) \quad 1 + g_1 D + (g_2 + g_1^2)D^2 + \cdots \\ \hline 1 + g_1 D + \cdots + g_r D^r \;\big)\; 1 \end{array}$$

$$\begin{array}{l} 1 + g_1 D + \quad g_2 D^2 + \quad g_3 D^3 + \cdots + \quad g_r D^r \\ \hline \quad g_1 D + \quad g_2 D^2 + \quad g_3 D^3 + \cdots + \quad g_r D^r \\ \quad g_1 D + \quad g_1^2 D^2 + \quad g_1 g_2 D^3 + \cdots + \; g_1 g_{r-1} D^r + g_1 D^{r+1} \\ \hline \quad (g_2 + g_1^2)D^2 + (g_3 + g_1 g_2)D^3 + \cdots \\ \quad (g_2 + g_1^2)D^2 + g_1(g_2 + g_1^2)D^3 + \cdots \\ \hline \quad \cdots \end{array}$$

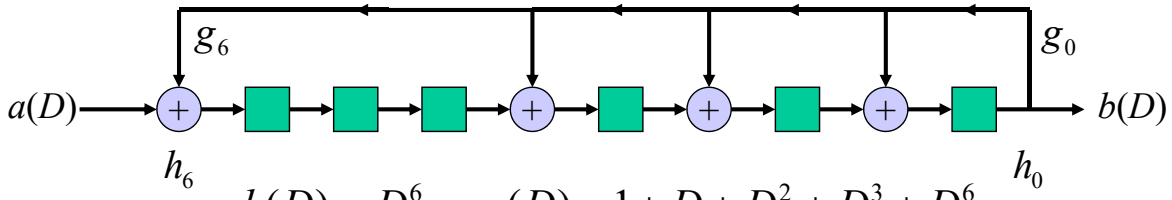
$$\begin{aligned} c_0 &= 1 \\ c_1 &= g_1 \\ c_2 &= g_1^2 + g_2 \\ &\vdots \end{aligned}$$

Example

- The initial contents of the shift register are set to be all '**0**'
- We assume that the input sequence is $a(D) = 1$
  - A '**1**' at time zero followed by an infinite string of '**0**'
- The output is obtain by

$$b(D) = \frac{D^6}{1 + D + D^2 + D^3 + D^6}$$

- The output can also be verified by manually calculating the contents of the shift register



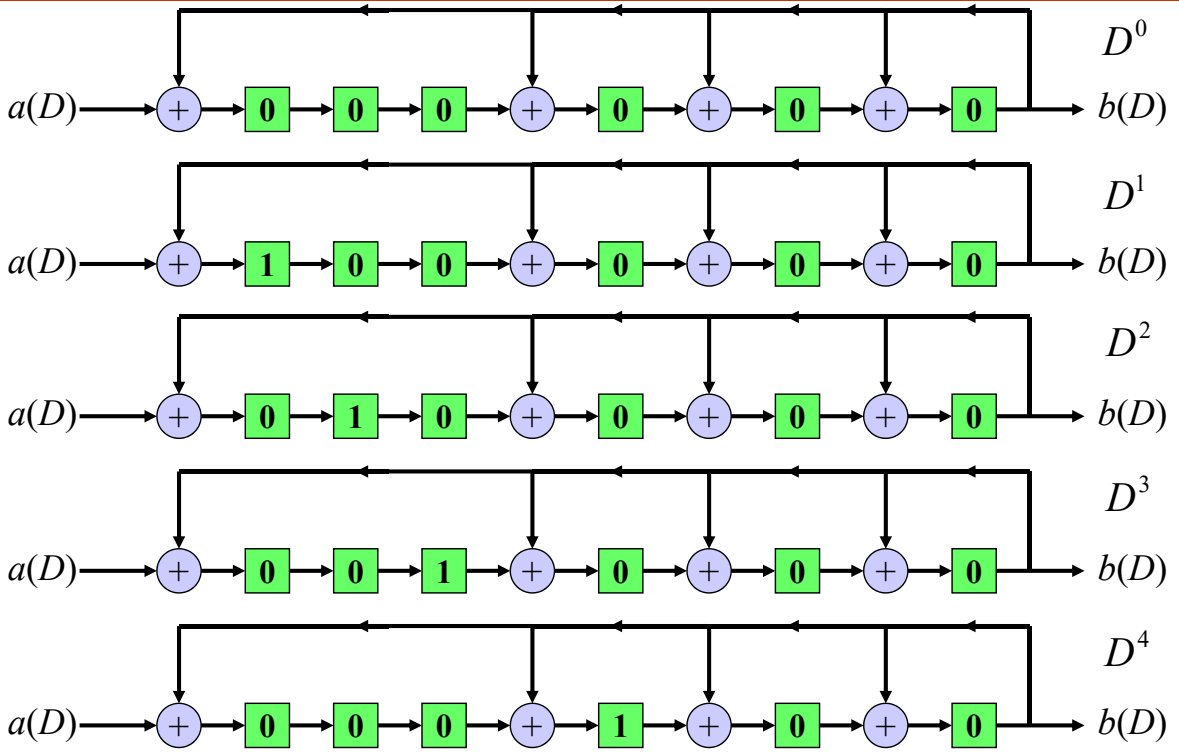
$$h(D) = D^6; \quad g(D) = 1 + D + D^2 + D^3 + D^6$$

Example (Cont.)

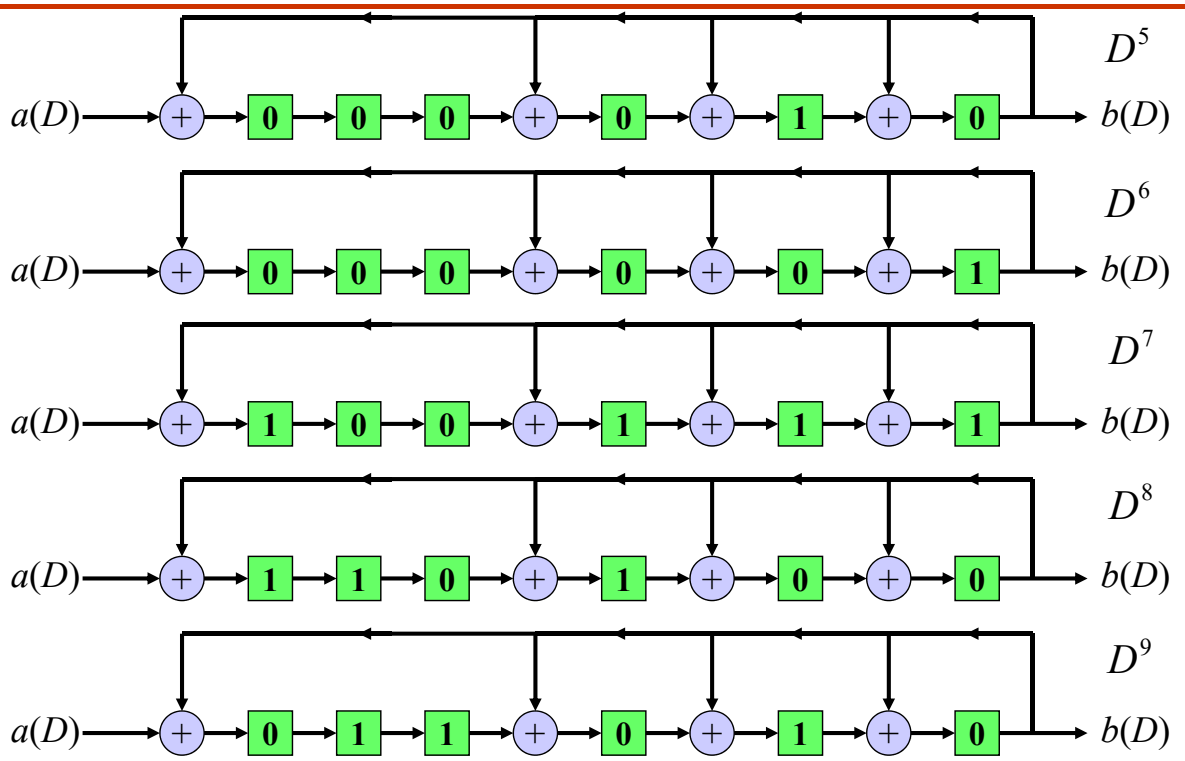
$$\begin{array}{r} D^6 + D^7 + \qquad\qquad D^{10} + D^{11} + D^{12} + \cdots \\ 1 + D + D^2 + D^3 + D^6 \,\big)\, D^6 \\ \hline D^6 + D^7 + D^8 + D^9 + \qquad\quad + D^{12} \\ D^7 + D^8 + D^9 + \qquad\quad + D^{12} \\ \hline D^7 + D^8 + D^9 + D^{10} + \qquad\qquad + D^{13} \\ D^{10} \qquad\quad + D^{12} + D^{13} \\ \hline D^{10} + D^{11} + D^{12} + D^{13} + \qquad\qquad + D^{16} \\ D^{11} \qquad\qquad\qquad\quad + D^{16} \\ \hline D^{11} + D^{12} + D^{13} + D^{14} + \qquad\qquad + D^{17} \\ D^{12} + D^{13} + D^{14} \qquad\quad + D^{16} + D^{17} \end{array}$$

$$b(D) = D^6 + D^7 + D^{10} + D^{11} + D^{12} + \cdots$$
$$\Rightarrow \mathbf{0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,1\,1}\cdots$$

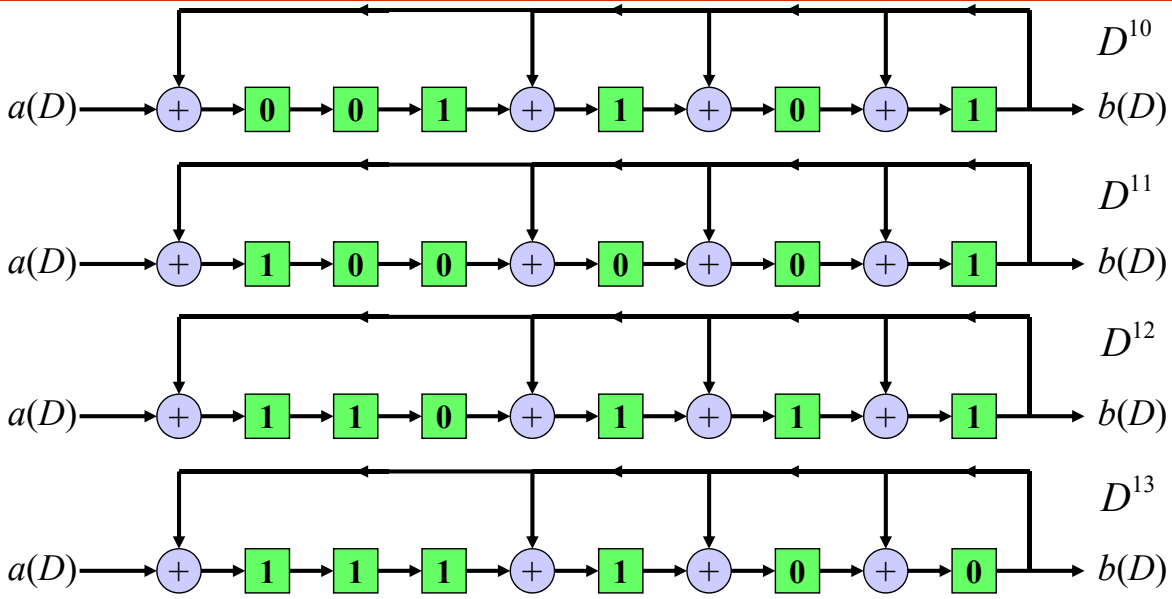
Example (Cont.)



Example (Cont.)



Example (Cont.)

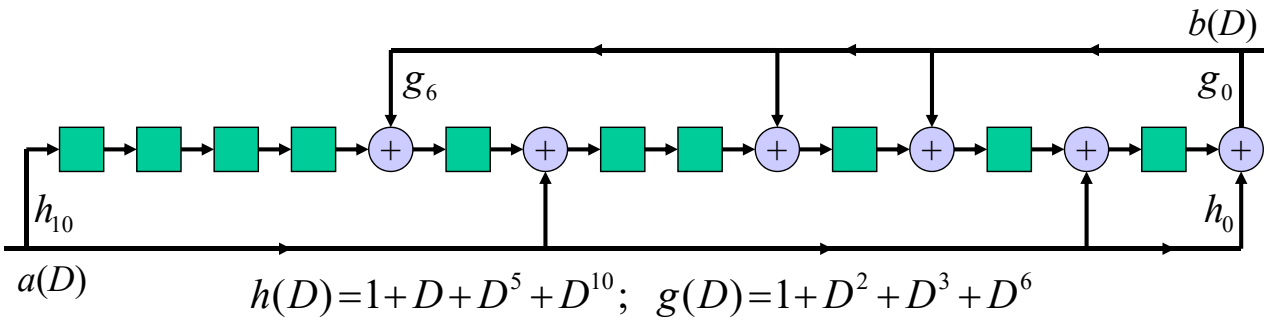


$$b(D) = D^6 + D^7 + D^{10} + D^{11} + D^{12} + \cdots$$

Example

- The initial contents of the shift register are set to be all '**0**'
- We assume that the input sequence is $a(D) = 1$
  - A '**1**' at time zero followed by an infinite string of '**0**'
- The output is obtain by

$$b(D) = \frac{1 + D + D^5 + D^{10}}{1 + D^2 + D^3 + D^6}$$



$$h(D) = 1 + D + D^5 + D^{10}; \quad g(D) = 1 + D^2 + D^3 + D^6$$

Linear Feedback Shift-Register Generator (G)

- Consider a input sequence with a **finite length**:
  - After the sequence ends, the circuit is equivalent to the **feedback shift register**
- This configuration is known as the **Galois configuration**

Galois Configuration

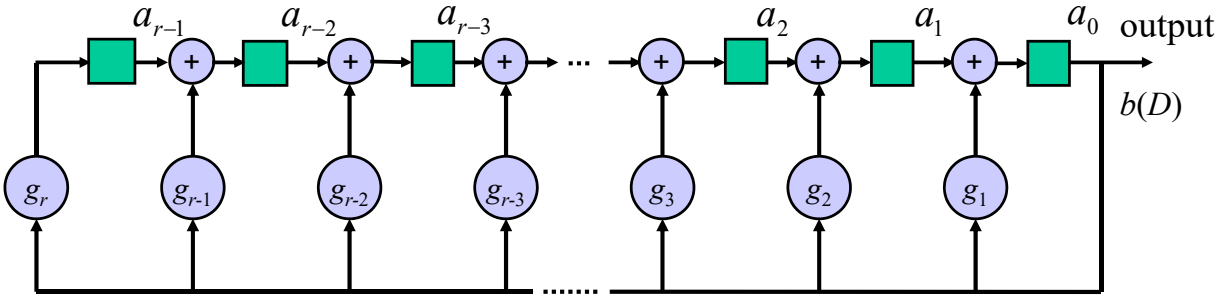


Fig. 3-5 High-speed linear feedback shift-register generator

Linear Feedback Shift-Register Generator

- Suppose that the input sequence $a_1(D)$ **ends at time *j* (degree *j*)**
  - The **highest power** of D in $a_1(D)h(D)$ is D^{j+r}
  - The coefficient of any power of D greater than $j+r$ is zero
  - According to $b(D)g(D) = a_1(D)h(D)$

$$\sum_{m=0}^{r} g_m b_{i-m} = 0, \quad \text{for } i > j+r$$

- Since $g_0 = 1 \Rightarrow b_i + \sum_{m=1}^{r} g_m b_{i-m} = 0, \quad \text{for } i > j+r$

$g_0 b_i = b_i$

$$b_i = \sum_{m=1}^{r} g_m b_{i-m}, \quad \text{for } i > j+r \longleftarrow$$

A feedback system

- Based on this relationship, we have another configuration – the **Fibonacci configuration**

Fibonacci Sequence

- The **Fibonacci numbers** form a sequence defined by the following recurrence relation:

$$F(n) = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$

- That is, after two starting values, each number is the **sum** of the two preceding numbers.
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025…

Linear Feedback Shift-Register Generator (F)

- A **second circuit configuration** for linear feedback shift-register generator – the **Fibonacci configuration**
  - The output is

$$b(D) = g_1 Db(D) + g_2 D^2 b(D) + \cdots + g_r D^r b(D)$$

Fibonacci Configuration

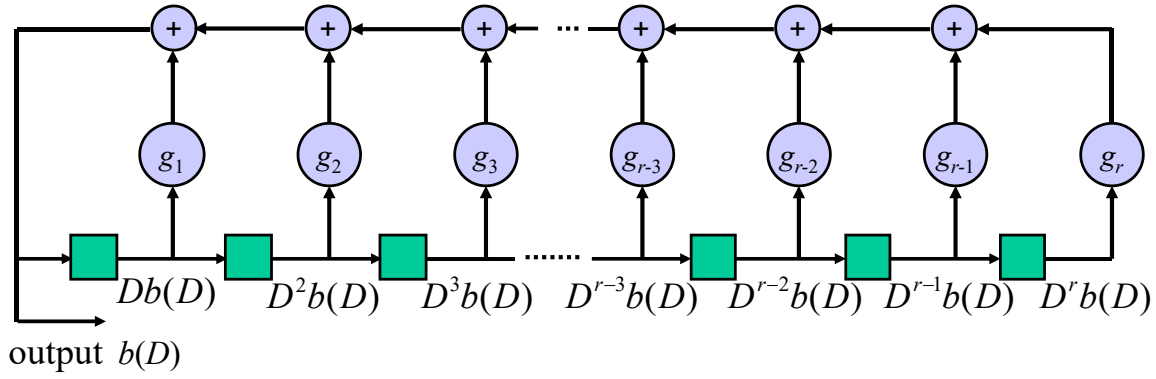


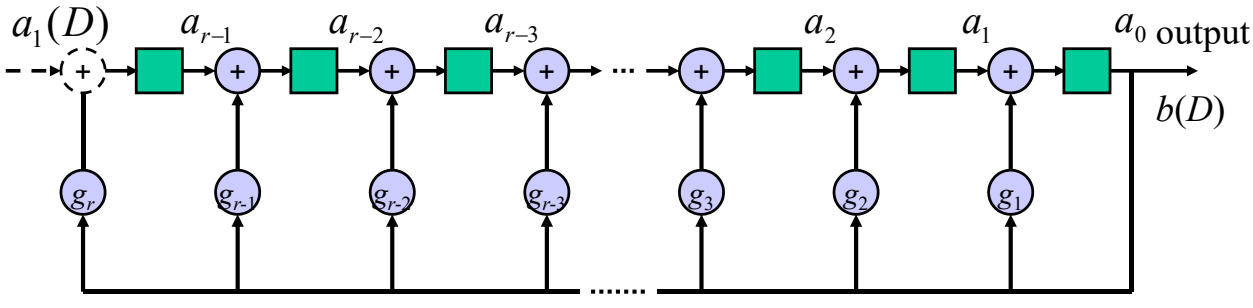
Fig. 3-6 Linear feedback shift-register generator

Linear Feedback Shift-Register Generator (Cont.)

- The configuration of **Fig. 3-6** is commonly used:
  - **Delayed outputs** for all delays up to r are **available**
- For the configuration of **Fig. 3-5**:
  - Delayed outputs are **not available**
  - Function at **higher speeds** than that of Fig. 3-6
    - Since there is **less propagation delay** in the feedback path

Output with Initial Contents (**G**)

- We need to determine the output of **Fig. 3-5** given the **initial contents** of the shift register
- The output $b(D)$ of **Fig. 3-5 (without input)** is identical to
  - The output $b'(D)$ beginning **at time *r*** of **Fig. 3-3** (with input)
    - $h(D) = D^r$ and
    - An input $a_1(D) = a_0 + a_1D + a_2D^2 + \ldots + a_{r-1}D^{r-1}$



Output with Initial Contents (**G**) (Cont.)

- At time r, the input have been **completely loaded** as the initial contents of the shift register
  - After time r, **there will be no input**

  $\Rightarrow$ Fig. 3-3 will converge to Fig. 3-5
- The output of Fig. 3-3: $b'(D) = \dfrac{h(D)a_1(D)}{g(D)} = \dfrac{D^r a_1(D)}{g(D)}$
- For the first r time units, the output $b'(D)$ is **zero**
  - The initial contents of shift register are all '0'
  - The output beginning at time r is simply $b'(D)$ **shifted by r time units** (D^j in $b'(D)$ is D^{j-r} in $b(D)$, for $j > r$)

  $\Rightarrow$ Beginning at time r: $b(D) = D^{-r}b'(D) = \dfrac{a_1(D)}{g(D)}$

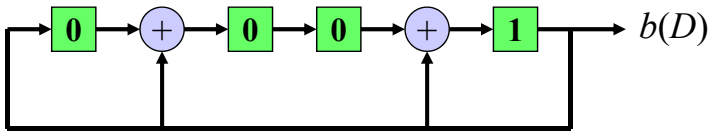
Example (**G**)

- Assume that the initial shift-register load is '**0001**' (a_3, a_2, a_1, a_0)
- The transfer function (generator polynomial) is

$$g(D) = 1 + D + D^3 + D^4$$

- The initial load: $a_1(D) = 1$
- The output:

$$b(D) = \frac{a_1(D)}{g(D)} = \frac{1}{1 + D + D^3 + D^4}$$

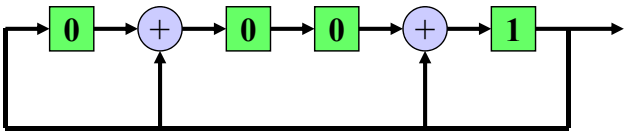


Example (**G**) (Cont.)

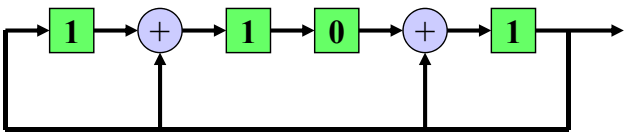
$$
\begin{array}{r}
1+D+D^2 \qquad\qquad +D^6+D^7+\cdots \\
1+D+D^3+D^4 \,\big)\, 1 \\
\underline{1+D+\quad D^3+D^4} \\
\quad D+\quad D^3+D^4 \\
\underline{D+D^2+\quad D^4+D^5} \\
\quad D^2+D^3 \quad +D^5 \\
\underline{D^2+D^3 \quad +D^5+D^6} \\
\quad D^6 \\
\underline{D^6+D^7+\quad D^9+D^{10}} \\
\quad D^7+\quad D^9+D^{10}
\end{array}
$$

$$
\begin{aligned}
b(D) &= 1+D+D^2+D^6+D^7+\cdots \\
&= \underbrace{111000}\ \underbrace{111000}\ \cdots
\end{aligned}
$$

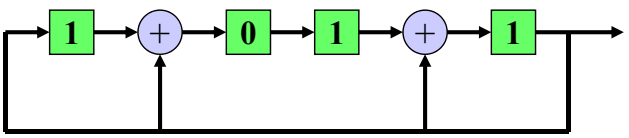
Example (**G**) (Cont.)



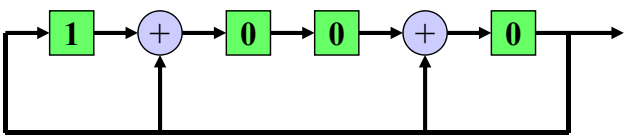
$\Rightarrow b_0 = 1$



$\Rightarrow b_1 = 1$

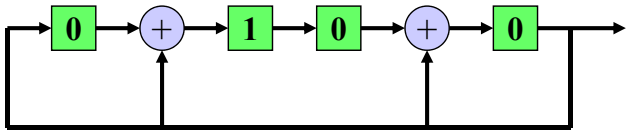


$\Rightarrow b_2 = 1$

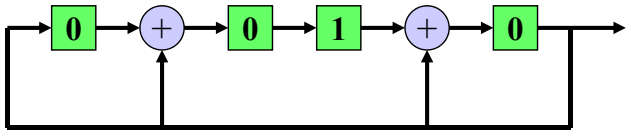


$\Rightarrow b_3 = 0$

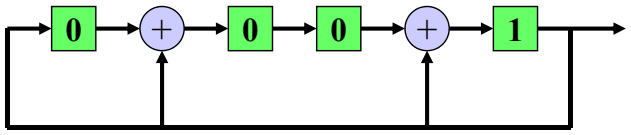
Example (G) (Cont.)



$\Rightarrow b_4 = 0$



$\Rightarrow b_5 = 0$



$\Rightarrow b_6 = 1$

$$b(D) = 1 + D + D^2 + D^6 + D^7 + \cdots$$
$$= \underbrace{111000}\ \underbrace{111000}\ \cdots$$

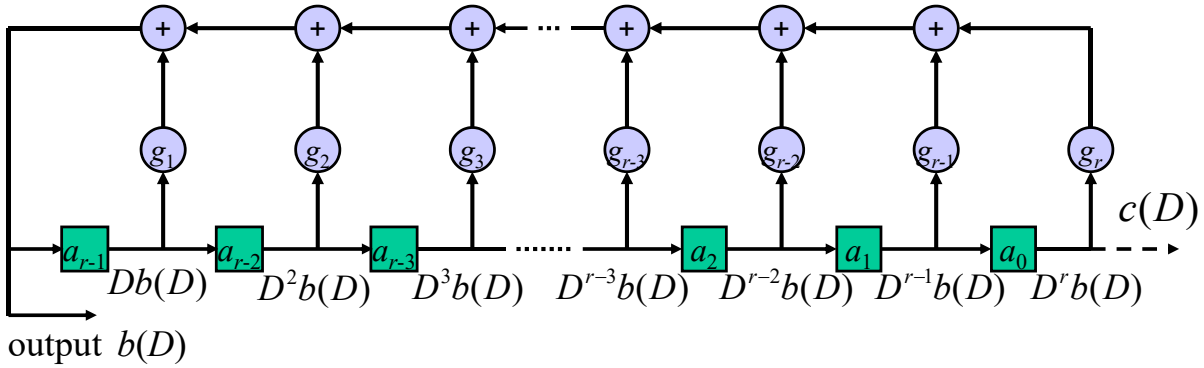
Output with Initial Contents (F)

- We need to determine the output of **Fig. 3-6** given the **initial contents** of the shift register
- Suppose that the initial contents for the circuit of Fig. 3-6 is

$$a(D) = a_0 + a_1 D + a_2 D^2 + \cdots + a_{r-1} D^{r-1}$$

- Define $c(D)$ to be the output of the **rightmost** shift register of Fig. 3-6

$$c(D) = D^r b(D)$$



Output with Initial Contents (**F**) (Cont.)

- Then, the first r elements of $c(D)$ are $a_0, a_1, a_2, \ldots, a_{r-1}$
- Determine directly based on Fig. 3-6 is hard
  - Determine through **Fig. 3-5**

Fig. 3-6: $a(D) \dashrightarrow b(D) = D^{-r}c(D)$

$$\begin{array}{ccc} c(D) & & c(D) \\ \| & & \| \\ b'(D) & & b'(D) \end{array}$$

$\Downarrow$ $\Uparrow$

Fig. 3-5: $a'(D) \Rightarrow b'(D)$

- Since the circuits of Fig. 3-5 and Fig. 3-6 are equivalent
  - The initial load of **Fig. 3-5** can be chosen such that
    - $b'(D)$ (of Fig. 3-5) = $c(D)$ (of Fig. 3-6)

Output with Initial Contents (**F**) (Cont.)

- Given an initial load $a(D)$ of the circuit in **F**-Config.
- Determine the first r output elements in $c(D)$ (**F**-Config.) based on $a(D)$
- Determine the first r output elements in $b'(D)$ (**G**-Config.) based on $c(D)$
- Find the initial load $a'(D)$ of the circuit in **G**-Config. based on
  - The first r output elements in $b'(D)$
  - The polynomial $g(D)$
- Find the complete output sequence $b'(D)$ in **G**-Config.
  - $b'(D) = a'(D)/g(D)$
- Determine the complete sequence $c(D)$ in **F**-Config.
- Determine the complete sequence $b(D)$ in **F**-Config.

Output with Initial Contents (**F**) (Cont.)

- Let the initial load of **Fig. 3-5** that accomplishes this ($b'(D)$) be
$$a'(D) = a'_0 + a'_1 D + a'_2 D^2 + \cdots + a'_{r-1} D^{r-1}$$
- Because the **first *r* elements** of $c(D)$ are (a_0, a_1, a_2, …, a_{r-1})
  - The **first *r* output elements** of $b'(D)$ should also be (a_0, a_1, a_2, …, a_{r-1})
  - Because $c(D) = D^r b(D)$ and $b'(D) = c(D)$, $b'(D)$ becomes
$$a_0 + a_1 D + a_2 D^2 + \cdots + a_{r-1} D^{r-1} + b'_r D^r + b'_{r+1} D^{r+1} + \cdots$$
- According to $b(D) = \dfrac{a_1(D)}{g(D)} \Rightarrow b'(D) = \dfrac{a'(D)}{g(D)}$
$$a_0 + a_1 D + a_2 D^2 + \cdots + a_{r-1} D^{r-1} + b'_r D^r + b'_{r+1} D^{r+1} + \cdots = \frac{a'_0 + a'_1 D + a'_2 D^2 + \cdots + a'_{r-1} D^{r-1}}{g_0 + g_1 D + g_2 D^2 + \cdots + g_r D^r}$$

Output with Initial Contents (**F**) (Cont.)

- Thus the initial state of Fig. 3-5 ($a'(D)$) , which produces the output sequence $b'(D)$ **as Fig. 3-6 with the initial load of *a*(*D*)**,
  - Can be found by **equating the first *r* coefficients**
$$\left[a(D) + b'_r D^r + b'_{r+1} D^{r+1} + \cdots\right] g(D) = a'(D)$$
- **None** of the coefficients $b'_j, j \geq r$, affect the calculation of a'_j ($a'(D)$ has **degree *r* – 1** and $a(D)$ has degree $r - 1$)
- The desired initial load a' is the **first *r*** coefficients of $a(D)g(D)$
  - The output $b(D)$ is
$$b(D) = D^{-r} c(D) = D^{-r} b'(D) = \frac{D^{-r} a'(D)}{g(D)}$$
  - That is, the elements of the $a'(D)/g(D)$ **except** the **first *r* output elements**

Example (**F**)

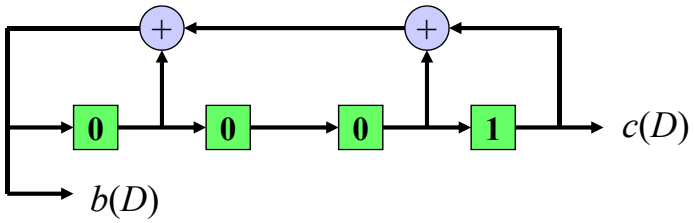
- Initial shift-register load: '**0001**' (a_3, a_2, a_1, a_0)
- The transfer function (generator polynomial) is

$$g(D) = 1 + D + D^3 + D^4$$

- The initial load: $a_1(D) = 1$
- The equivalent initial load for **Fig. 3-5**:

$$g(D) + \left(b_4' D^4 + b_5' D^5 + \cdots\right) g(D) = a_0' + a_1' D + a_2' D^2 + a_3' D^3$$

$$\Rightarrow \quad a_0' = g_0 = 1$$
$$a_1' = g_1 = 1$$
$$a_2' = g_2 = 0$$
$$a_3' = g_3 = 1$$



Example (**F**) (Cont.)

- The complete output sequence is equivalent to

$$b'(D) = \frac{a'(D)}{g(D)} = \frac{1 + D + D^3}{1 + D + D^3 + D^4}$$

$$\begin{array}{r|l}
 & 1 \qquad\qquad + D^4 + D^5 + D^6 \qquad\qquad + D^{10} + \cdots \\
\hline
1 + D + D^3 + D^4 & 1 + D + \quad D^3 \\
 & 1 + D + \quad D^3 + D^4 \\
\hline
 & \qquad\qquad D^4 \\
 & \qquad\qquad D^4 + D^5 + \quad D^7 + D^8 \\
\hline
 & \qquad\qquad\quad D^5 + \quad D^7 + D^8 \\
 & \qquad\qquad\quad D^5 + D^6 + \quad D^8 + D^9 \\
\hline
 & \qquad\qquad\qquad D^6 + D^7 + \quad D^9 \\
 & \qquad\qquad\qquad D^6 + D^7 + \quad D^9 + D^{10} \\
\hline
 & \qquad\qquad\qquad\qquad\qquad\qquad D^{10}
\end{array}$$

$$b'(D) = 1 + D^4 + D^5 + D^6 + D^{10} + \cdots$$
$$= \underbrace{100011}\ \underbrace{100011}\ \cdots$$

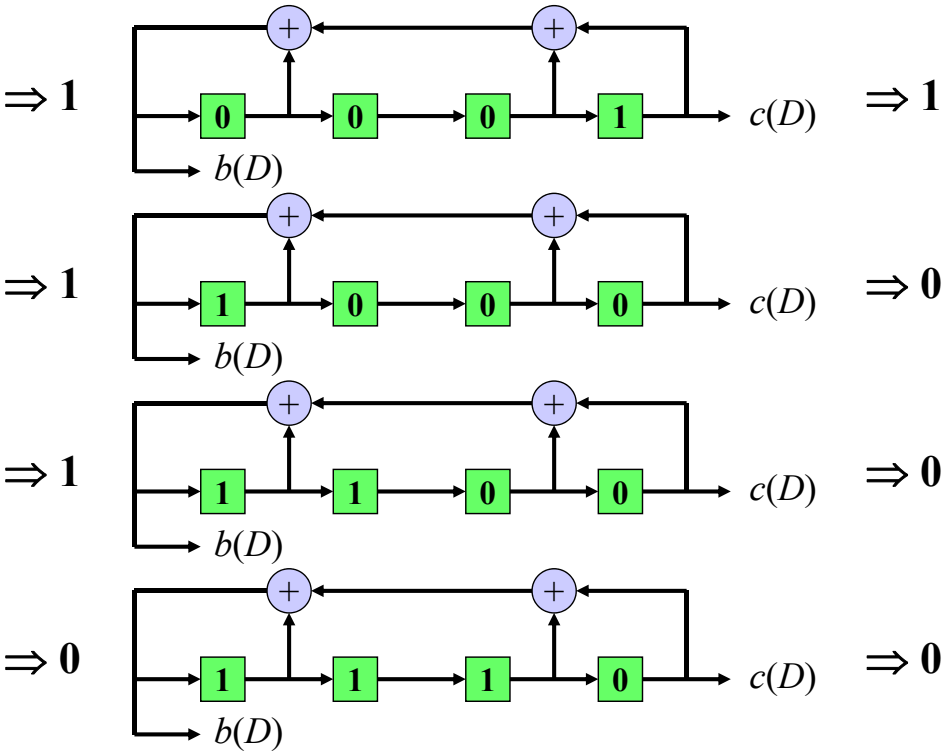
Example (F) (Cont.)

- Multiplying by $D^{-r} = D^{-4}$

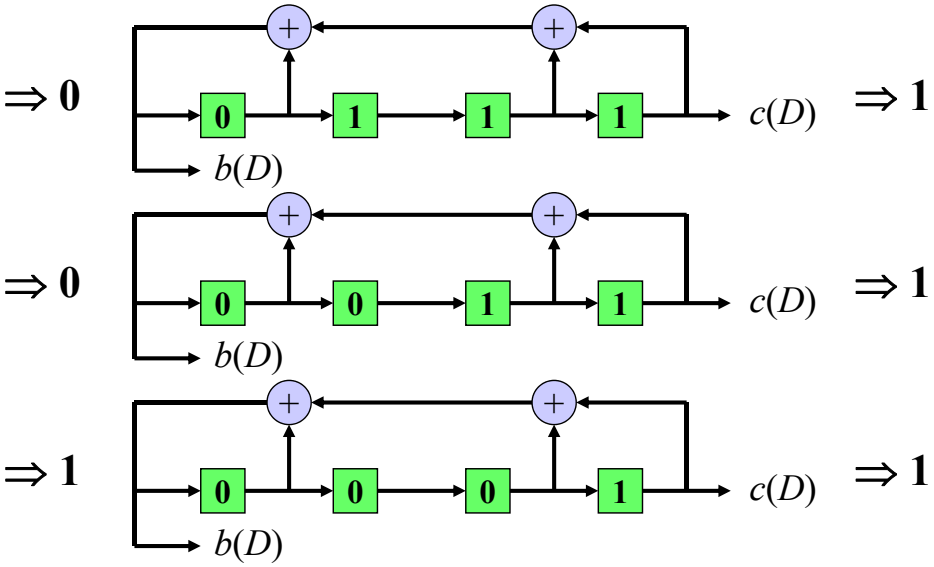
$$b(D) = D^{-4}b'(D) = D^{-4} + 1 + D + D^2 + D^6 + D^7 + D^8 + \cdots$$
$$= \underbrace{111000}\quad \underbrace{111000}\quad \cdots$$

- The preceding output '1000' is the initial load of the shift register
- When the same polynomial $g(D)$ is used, different generator configurations (Galois and Fibonacci) yield **the same sequence**, but **the initial contents may be different**.

Example (F) (Cont.)



Example (**F**) (Cont.)



$$b(D) = 1 + D + D^2 + D^6 + D^7 + \cdots$$
$$= \underbrace{111000}\ \underbrace{111000}\ \cdots$$

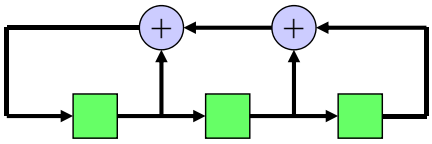
Linear Feedback Shift-Register Generator (Cont.)

- Several observations about **G**-Config. and **F**-Config. are made:
  - Given **nonzero** initial conditions, **neither** of the registers will ever reach an **all-zero state**
    - If an all-zero state occurs, the output becomes all-zero
  - Since the register contains r stages and an r-stage shift register has at most $2^r - 1$ nonzero states

  $\Rightarrow$ The output must be periodic with a period of **at most** $2^r - 1$
  - The period can be significantly less than $2^r - 1$
  - The same circuit may generate many different output sequences
    - The output depends on the **initial state** of the shift register

Example

- Based on the following circuit, there are **four** different sets of shift-register states
- These four possible cycles have periods of 1, 1, 2, and 4
- All possible shift-register states are included in one of the four cycles ($1+1+2+4 = 8 = 2^3$)



Cycle 1	Cycle 2	Cycle 3	Cycle 4
0 0 0	1 1 1	0 1 0	1 0 0
0 0 0	1 1 1	1 0 1	1 1 0
0 0 0	1 1 1	0 1 0	0 1 1
⋮	⋮	1 0 1	0 0 1
		⋮	1 0 0
			1 1 0
			0 1 1
			0 0 1
			⋮

Linear Feedback Shift-Register Generator (Cont.)

- The **maximum** possible period for an arbitrary feedback shift-register connection defined by $g(D)$:
  - Can be found by the **reciprocal polynomial** of $g(D)$:

$$g_r(D) = D^r g\left(1/D\right)$$

- The **maximum** possible period is the **smallest** possible integer N for which $D^N + 1$ is **divisible** by $g_r(D)$

  $\Rightarrow$ A polynomial $h_r(D)$ such that

$$g_r(D)h_r(D) = D^N + 1$$

Example

- Consider the generator polynomial

$$g(D) = 1 + D + D^3 + D^4$$

$$\begin{aligned} g_r(D) &= D^4 g\left(1/D\right) \\ &= D^4\left(1 + D^{-1} + D^{-3} + D^{-4}\right) \\ &= D^4 + D^3 + D + 1 \end{aligned}$$

- The smallest possible integer N for which $D^N + 1$ is divisible by $g_r(D)$ is
  - $N = 6$
- The maximum possible period is 6 ($< 2^4 - 1 = 15$)

State-Machine Representation

State-Machine Representation

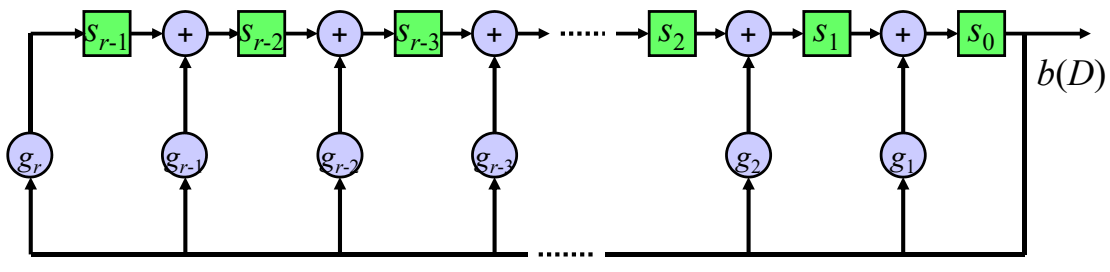
- The shift-register generator may be viewed as a **state machine**
  - Whose state at time n is the **contents** of the shift register represented by a column vector
  - Define the state at time n by

$$\mathbf{S}_n = \begin{bmatrix} s_{0,n} \\ s_{1,n} \\ s_{2,n} \\ \vdots \\ s_{r-2,n} \\ s_{r-1,n} \end{bmatrix}$$

State-Machine Representation (**G**)

- Consider **G**-Config.
  - The contents of the shift register at time $n+1$

$$\begin{aligned} s_{0,n+1} &= s_{1,n} + g_1 \cdot s_{0,n} \\ s_{1,n+1} &= s_{2,n} + g_2 \cdot s_{0,n} \\ &\vdots \\ s_{r\text{-}2,n+1} &= s_{r\text{-}1,n} + g_{r\text{-}1} \cdot s_{0,n} \\ s_{r\text{-}1,n+1} &= g_r \cdot s_{0,n} \end{aligned}$$



State-Machine Representation (**G**) (Cont.)

- Define the **Galois state transition matrix G** as the $r \times r$ square matrix

$$\mathbf{S}_{n+1} = \begin{bmatrix} s_{0,n+1} \\ s_{1,n+1} \\ s_{2,n+1} \\ \vdots \\ s_{r-2,n+1} \\ s_{r-1,n+1} \end{bmatrix} = \begin{bmatrix} g_1 & 1 & 0 & 0 & \cdots & 0 \\ g_2 & 0 & 1 & 0 & \cdots & 0 \\ g_3 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ g_{r-1} & 0 & 0 & 0 & \cdots & 1 \\ g_r & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \times \begin{bmatrix} s_{0,n} \\ s_{1,n} \\ s_{2,n} \\ \vdots \\ s_{r-2,n} \\ s_{r-1,n} \end{bmatrix} = \mathbf{G} \times \mathbf{S}_n$$

- The state at any time n can be found as

$$\begin{aligned} \mathbf{S}_1 &= \mathbf{G} \times \mathbf{S}_0 \\ \mathbf{S}_2 &= \mathbf{G} \times \mathbf{S}_1 = \mathbf{G} \times \mathbf{G} \times \mathbf{S}_0 \\ &\vdots \\ \mathbf{S}_n &= \mathbf{G}^n \times \mathbf{S}_0 \end{aligned}$$

State-Machine Representation (**G**) (Cont.)

- Find the state at time n from the state at time $n + 1$:

$$\mathbf{G}^{-1} \times \mathbf{S}_{n+1} = \mathbf{G}^{-1} \times \mathbf{G} \times \mathbf{S}_n = \mathbf{S}_n$$

- The state at any time **in the past**:

$$\mathbf{S}_{n-k} = \left[\mathbf{G}^{-1} \right]^k \times \mathbf{S}_n$$

- The output of the shift-register generator at time n:

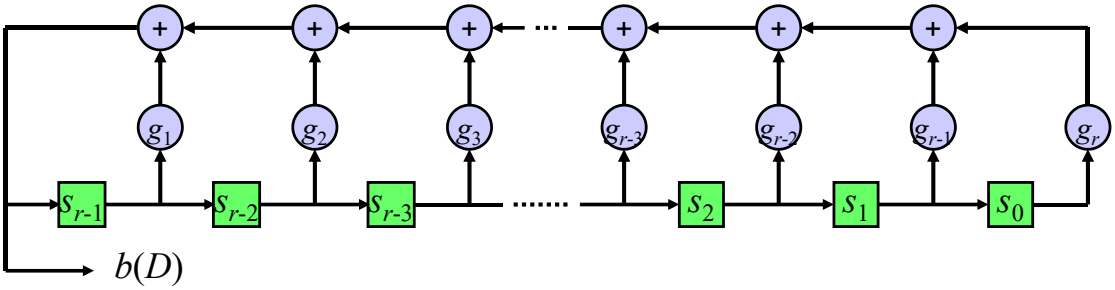
$$\begin{aligned} b_n &= \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \times \begin{bmatrix} s_{0,n} \\ s_{1,n} \\ \vdots \\ s_{r-2,n} \\ s_{r-1,n} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \times \mathbf{G}^n \times \mathbf{S}_0 \end{aligned}$$

State-Machine Representation (**F**)

- Similar results for **F**-Config.

$$\mathbf{S}_{n+1} = \begin{bmatrix} s_{0,n+1} \\ s_{1,n+1} \\ s_{2,n+1} \\ \vdots \\ s_{r-1,n+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ g_r & g_{r-1} & g_{r-2} & \vdots & g_1 \end{bmatrix} \times \begin{bmatrix} s_{0,n} \\ s_{1,n} \\ s_{2,n} \\ \vdots \\ s_{r-1,n} \end{bmatrix} = \mathbf{F} \times \mathbf{S}_n$$

$$b_n = \begin{bmatrix} g_r & g_{r-1} & g_{r-2} & g_{r-3} & \cdots & g_1 \end{bmatrix} \times \mathbf{F}^n \times \mathbf{S}_0$$



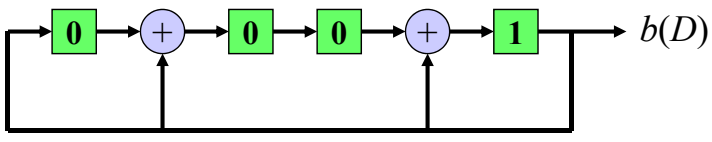
Example

- The Galois state-transition matrix is

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

- The state at time $n = 0$ is

$$\mathbf{S}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



Example (Cont.)

- At any time n, the output is

$$b_n = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\end{bmatrix}^n \times \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

- For $n = 0$

$$b_0 = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\end{bmatrix}^0 \times \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = 1$$

- For $n = 1$

$$b_1 = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\end{bmatrix}^1 \times \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1 & 1 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = 1$$

Example (Cont.)

- For $n = 2$

$$b_2 = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

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

- For $n = 3$

$$b_3 = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1 & 1 & 1 & 0\\ 1 & 0 & 0 & 1\\ 0 & 1 & 0 & 0\\ 1 & 1 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

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

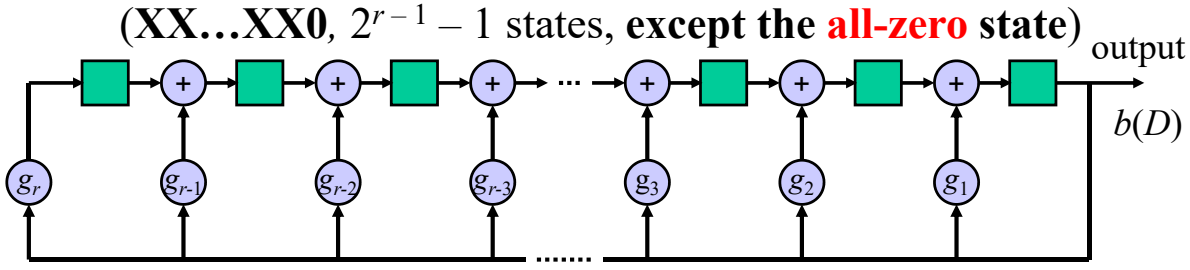
Maximal-Length Sequences

Maximal-Length Sequences

- The maximum possible period is the smallest possible integer N for which $D^N + 1$ is divisible by the reciprocal polynomial $g_r(D)$
- Consider the linear feedback shift-register generators (Fig. 3-5 or Fig. 3-6) with $g(D)$ a **primitive polynomial**
  - The smallest N for $g(D)$ of degree r: $N = 2^r - 1$
  - Result in a cycle with period $N = 2^r - 1$
  - There are a total of $2^r - 1$ nonzero states $\Rightarrow$ **all nonzero states** are passed through in this cycle
  - There is **only one** possible cycle
- **Maximal-length sequence** or ***m*-sequence**:
  - Shift-register sequence have the **maximum possible** period
  - A different initial condition results in a **different code phase**

Properties of *m*-Sequences

- **Property I:** A maximal-length sequence contains one more '1' than '0'. The number of ones in the sequence is $\frac{1}{2}(N+1) = 2^{r-1}$
  - Consider the generator of **G**-Config., the **rightmost symbol** of the shift-register state is the **output symbol**
  - The shift-register passes through all nonzero states
  - $2^{r-1} = \frac{1}{2}(N+1)$ states have a '1' in the **rightmost** symbol (**XX…XX1**, $r-1$ symbols can be '1' or '0' $\Rightarrow 2^{r-1}$ states)
  - $2^{r-1} - 1$ states have a '0' in the **rightmost** symbol (**XX…XX0**, $2^{r-1} - 1$ states, **except the all-zero state**)



Properties of *m*-Sequences (Cont.)

- **Property II:** The modulo-2 sum of an *m*-sequence and any **phase shift** of the same sequence is **another phase** of the same *m*-sequence (**shift-and-add property**)
  - Consider the generator of **G**-Config., the **output symbol** is given by

$$b(D) = \frac{a_1(D)}{g(D)}$$

  - A different initial condition results in a different phase of the same *m*-sequence.
  - Let $b(D)$ and $b'(D)$ are two different phases

$$b(D) = \frac{a(D)}{g(D)}; \quad b'(D) = \frac{a'(D)}{g(D)}$$

    - where $a(D)$ and $a'(D)$ are distinct initial conditions ($\neq 0$)

Properties of *m*-Sequences (Cont.)

– The modulo-2 sum

$$b(D)+b'(D)=\frac{[a(D)+a'(D)]}{g(D)}=\frac{a''(D)}{g(D)}$$

– The modulo-2 sum of any two distinct initial conditions is a third distinct initial conditions

$$a''(D)\neq a(D);\quad a''(D)\neq a'(D)$$

– The output sequence of $a''(D)$ is

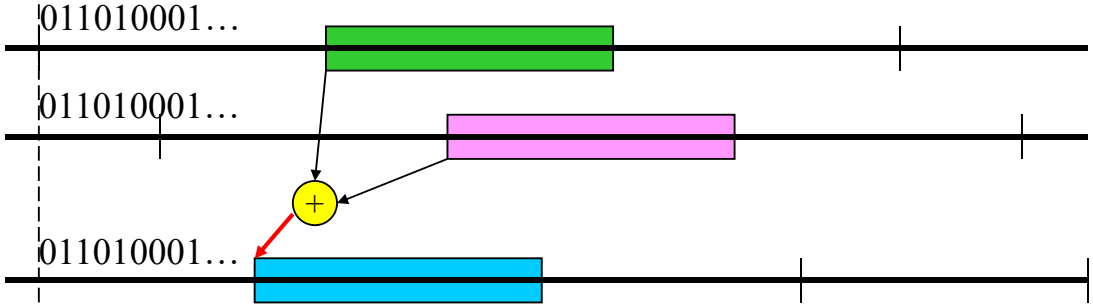
$$\frac{a''(D)}{g(D)}=b''(D)$$

– The resultant is a third distinct phase of the original sequence $b(D)$

$$b''(D)\neq b(D);\quad b''(D)\neq b'(D)$$

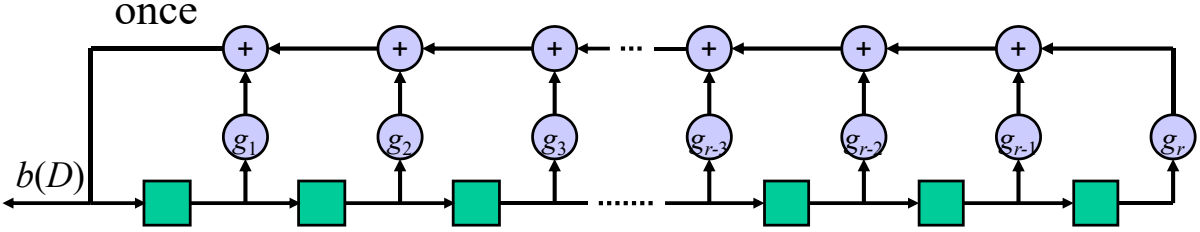
Properties of *m*-Sequences (Cont.)

- Consider a primitive polynomial $g(D) = 1+D+D^4$
  – Initial condition $a(D) = 1$
- **b**(0) = 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0
  – The number of '1' is 8 and the number of '0' is 7;
- **b**(0) = 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0
- **b**(6) = 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1
- **b**(0) + **b**(6) = 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 = **b**(8)

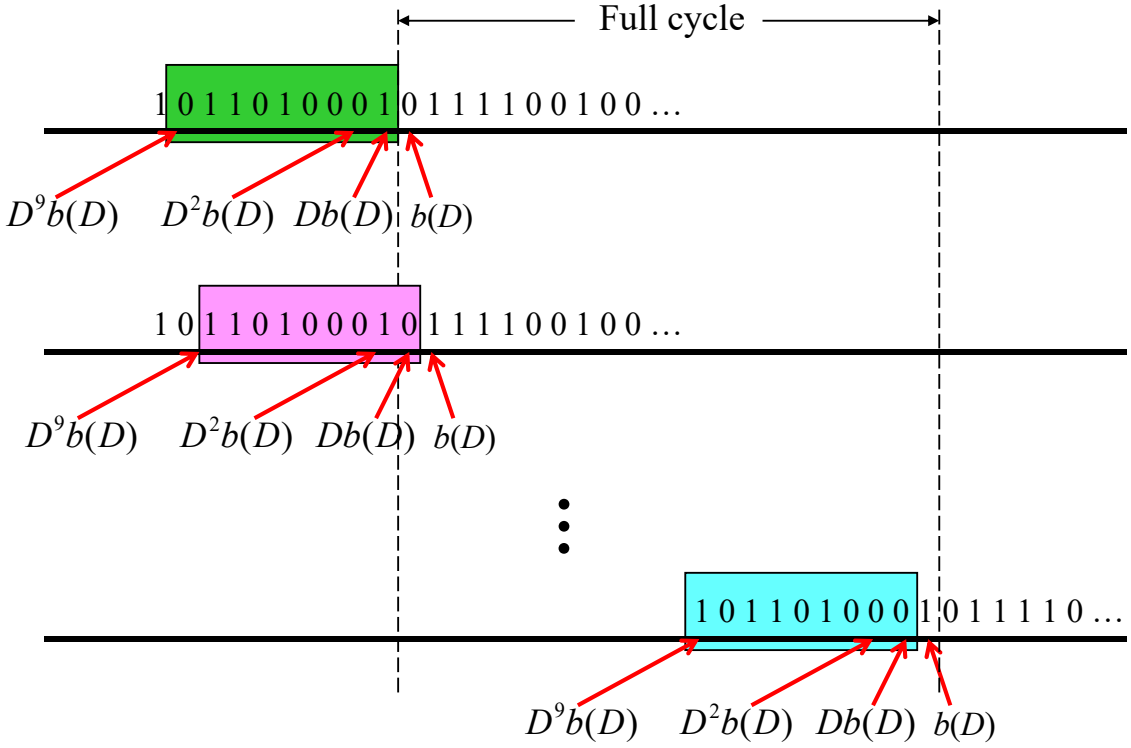


Properties of *m*-Sequences (Cont.)

- **Property III:** If a window of width r is slide along the sequence for N shifts, each ***r*-tuple** expect the all-zero r-tuple will appear **exactly once**
  - Consider the generator of **F**-Config., the **output sequence** $b(D)$ passes through the shift register of this generator
  - The window of width r is the **state** of the shift register
  - For ***N* shifts**, it will be a **full cycle** of the sequence
  - The shift register passes through all **nonzero states** exactly once



Properties of *m*-Sequences (Cont.)



Properties of *m*-Sequences (Cont.)

- **Property IV:** The periodic autocorrelation function $\theta_b(k)$ is **two-valued** and is given by

$$\theta_b(k) = \begin{cases} 1.0, & k = lN \\ -1/N, & k \neq lN \end{cases}$$

  - where l: any integer, N: sequence period
  - The value of the periodic autocorrelation function $\theta_b(k)$ is
    - In the **modulo-2 sum** of the sequence **b** and the k-th cyclic shift of **b**, calculate the value

$$\frac{(N_A - N_D)}{N}$$

      - N_A is the number of zeros (agree)
      - N_D is the number of ones (disagree)

Properties of *m*-Sequences (Cont.)

  - For $k = lN$:
    - The k-th cyclic shift of **b** is identical to **b**

    $\Rightarrow N_A = N$, $N_D = 0$, and $\theta_b(lN) = N/N = 1.0$
  - For $k \neq lN$:
    - The modulo-2 sum is **another phase** of the original sequence (by **Property II**)
    - By **Property I**, there is one more '1' than '0' in the modulo-2 sum

    $\Rightarrow N_A - N_D = -1$, and $\theta_b(k) = -1/N$

Properties of *m*-Sequences (Cont.)

- Consider a primitive polynomial $g(D) = 1+D+D^4$
  - Initial condition $a(D) = 1$
- $\mathbf{b}(0) =$ 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0
- $\mathbf{b}(15) =$ 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 $= \mathbf{b}(30) = \mathbf{b}(45) = \ldots$
- $\mathbf{b}(0) + \mathbf{b}(15) =$ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $\Rightarrow \theta_b(15) = 1.0$

- $\mathbf{b}(0) =$ 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0
- $\mathbf{b}(6) =$ 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1
- $\mathbf{b}(0) + \mathbf{b}(6) =$ 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 $= \mathbf{b}(8)$

$\Rightarrow \theta_b(6) = -1/15$

Properties of *m*-Sequences (Cont.)

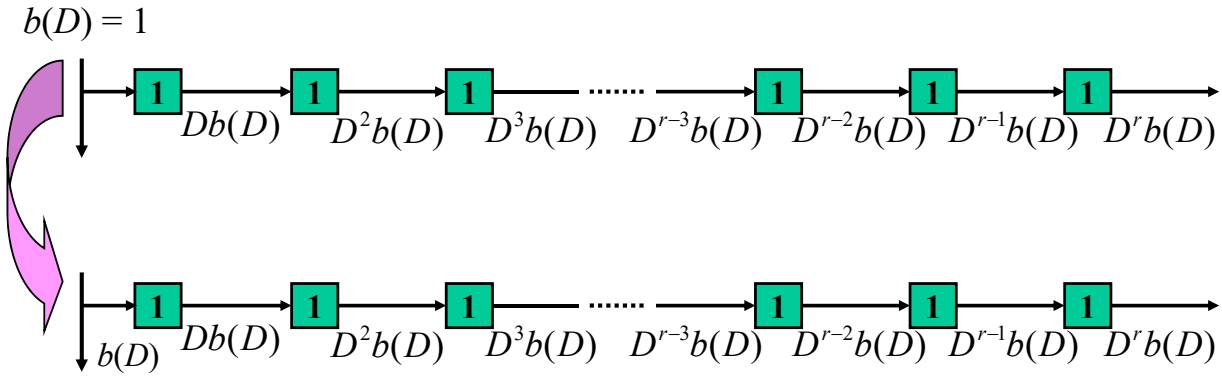
- **Property V:** Define a **run** as a **subsequence** of **identical symbols** within the *m*-sequence. The **length** of this subsequence is the length of the run. Then, for any *m*-sequence, there is
  1. 1 run of **ones** of length r
  2. 1 run of **zeros** of length $r - 1$
  3. 1 run of **ones** and 1 run of **zeros** of length $r - 2$
  4. 2 runs of **ones** and 2 runs of **zeros** of length $r - 3$
  5. 4 runs of **ones** and 4 runs of **zeros** of length $r - 4$

  …

  r. 2^{r-3} runs of **ones** and 2^{r-3} runs of **zeros** of length 1

Properties of *m*-Sequences (Cont.)

- ***Proof of Property V*:**
  - Consider the generator of **F**-Config., there can be **no** run of ones having **length $l > r$**
    - Since it requires that the all-ones state is followed by another all-ones state
    - This cannot occur since each state occurs **only once**



Properties of *m*-Sequences (Cont.)

Length r

1 1 1 … 1 1 1

Length r

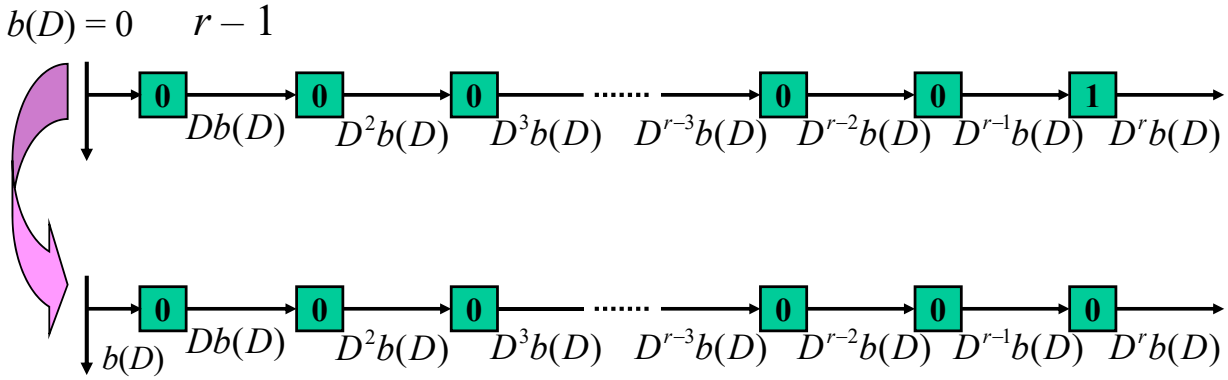
Two all-one states ⇒ Impossible

  - There is **a single run of r consecutive ones**, and this run is preceded by a zero and followed by a zero
    - State 11…11 occurs only once

Properties of *m*-Sequences (Cont.)

– 1 run of zeros of length $r-1$

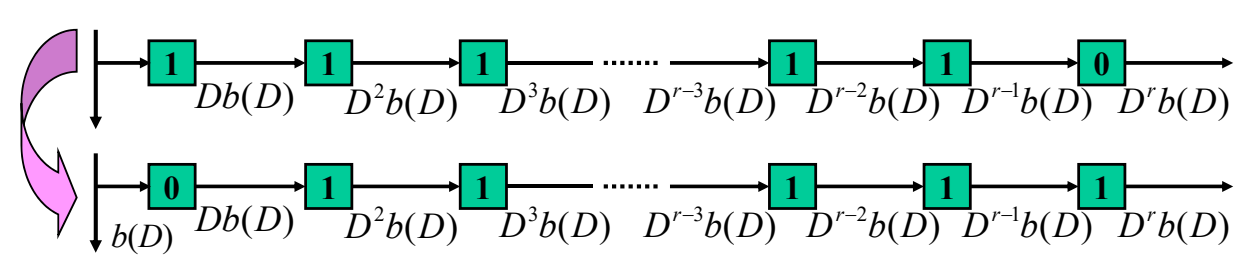
- A run of $r-1$ zeros must be preceded by and followed by a one; otherwise the state **00…00** will occur
  - The state 00…01 is followed immediately by the state 10…00 **(00…01 → 10…00)**
- Each state occurs **only once** ⇒1 run of zeros of length $r-1$



Properties of *m*-Sequences (Cont.)

– No run of ones of length $r-1$

- A run of $r-1$ ones must be preceded by and followed by a zero ⇒ The state 11…10 is followed immediately by the state 01…11 **(11…10 → 01…11)**
- These two states are also passed through in generating **the run of *r* ones** (**11…10** → 11…11 → **01…11**)
- Each state occurs only once ⇒ **no run** of ones of length $r-1$



Properties of *m*-Sequences (Cont.)

– Consider a run of k ones or zeros where $1 \le k \le r-2$

- A run of k ones (zeros) must be preceded by and followed by a zero (one)
- The state must be X…X**011…110**X…X with the $r-k-2$ remaining positions taking on **arbitrary values**

$\Rightarrow$ There are 2^{r-k-2} possible ways

- There are 2^{r-k-2} runs of k ones or zeros

Power Spectrum of *m*-Sequences

- The **power spectrum** of $c(t)$ is the Fourier transform of the autocorrelation function $R_c(\tau)$ $(0 \le \tau_\varepsilon \le T_c,\ \ \tau = kT_c + \tau_\varepsilon)$

$$R_c(k,\tau_\varepsilon) = (1-\frac{\tau_\varepsilon}{T_c})\theta_b(k) + \frac{\tau_\varepsilon}{T_c}\theta_b(k+1) \qquad \text{(Eq. (3-8))}$$

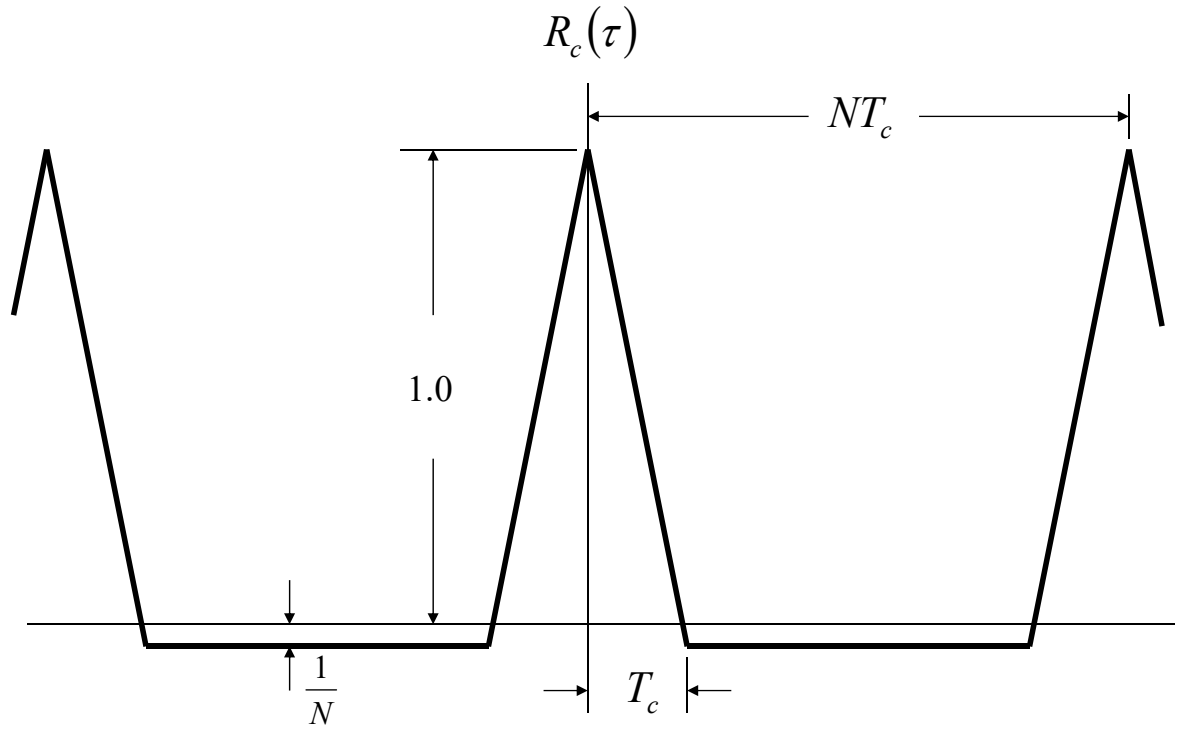
$$R_c(\tau) = (1-\frac{\tau}{T_c})\times 1.0 + \frac{\tau}{T_c}\times(-\frac{1}{N}) = 1-\frac{\tau}{T_c}(1+\frac{1}{N}),\ 0 \le \tau \le T_c$$

$$R_c(\tau) = (1-\frac{\tau_\varepsilon}{T_c})\times(-\frac{1}{N}) + \frac{\tau_\varepsilon}{T_c}\times(-\frac{1}{N}) = -\frac{1}{N},\ T_c \le \tau \le (N-1)T_c$$

$$R_c(\tau) = (1-\frac{\tau_\varepsilon}{T_c})\times(-\frac{1}{N}) + \frac{\tau_\varepsilon}{T_c}\times 1.0 = \frac{\tau_\varepsilon}{T_c}(1+\frac{1}{N}) - \frac{1}{N},\ (N-1)T_c \le \tau \le NT_c$$

- Since $\theta_b(k)$ is **periodic**, $R_c(\tau)$ is also **periodic** and has a period $T = NT_c$

Power Spectrum of *m*-Sequences (Cont.)

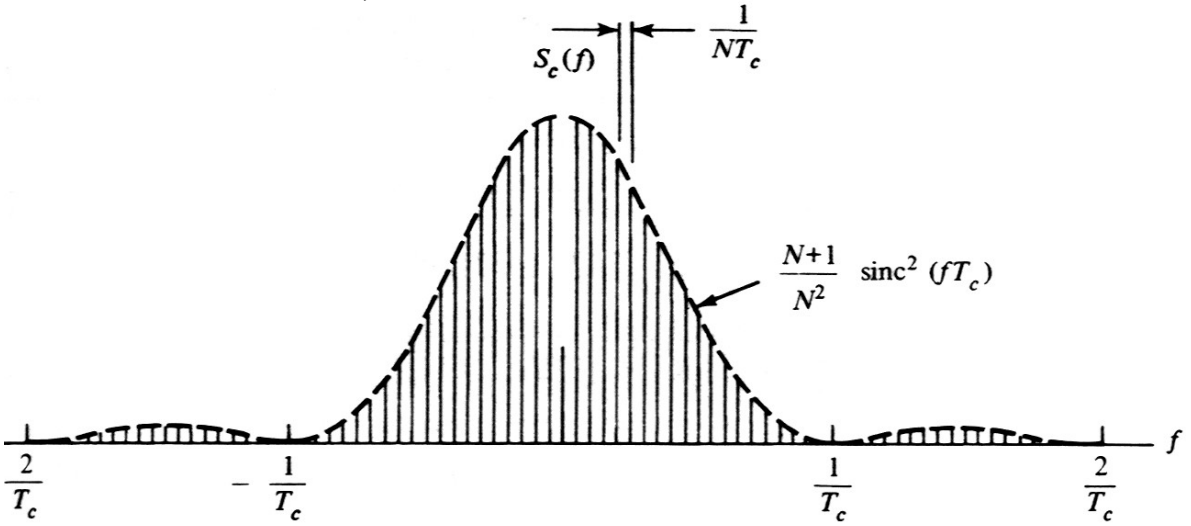


Power Spectrum of *m*-Sequences (Cont.)

- By taking the Fourier transform:

$$S_c(f) = \sum_{m=-\infty}^{\infty} P_m \delta(f - mf_0)$$

where $P_0 = 1/N^2$, $P_{m,m\neq 0} = [(N+1)/N^2]\,\mathrm{sinc}^2(m/N)$, and $f_0 = 1/NT_c$



Power Spectrum of *m*-Sequences (Cont.)

- Suppose the *m*-sequence $c(t)$ is **biphase** (–1, +1) and modulates a sinusoidal carrier having power P and frequency f_c

$$s(t) = \sqrt{2P}c(t)\cos(2\pi f_c t)$$

  - The power spectrum of this modulated carrier is

$$S_s(f) = S_c(f) * \frac{P}{2}\delta(f - f_c) + S_c(f) * \frac{P}{2}\delta(f + f_c)$$

Convolution

  - The resultant power spectrum is a translation of the discrete spectrum $S_c(f)$ upward and downward by a frequency f_c
  - In most SS systems, the carrier is **randomly modulated by data** as well as the spreading code

$\Rightarrow$ The transmitted spectrum is **continuous and not discrete**

Polynomials Yielding *m*-Sequences

- Table 3-5 presents the primitive polynomials used to generate m-sequences
  - All polynomials are specified by an **octal** number that defines the coefficients of $g(D)$
  - Beginning with g_0 on the **right** and proceeding to g_r in the **last nonzero** position on the **left**
- Example 3-13:
  - [367]:

octal		binary		coefficient
3	=	(011)	=	(g_8, g_7, g_6);
6	=	(110)	=	(g_5, g_4, g_3);
7	=	(111)	=	(g_2, g_1, g_0);

$$g(D) = 1 + D + D^2 + D^4 + D^5 + D^6 + D^7$$

Polynomials Yielding *m*-Sequences (Cont.)

- Asterisk(∗): with only **two** feedback connections

Degree	Octal Representation of Generator Polynomial
2	[7]*
3	[13]*
4	[23]*
5	[45]*, [75], [67]
6	[103]*, [147], [155]
7	[211]*, [217], [235], [367], [277], [325], [203]*, [313], [345]
8	[435], [551], [747], [453], [545], [537], [703], [543]
9	[1021]*, [1131], [1461], [1423], [1055], [1167], [1541], [1333], [1605], [1751], [1743], [1617], [1553], [1157]
10	[2011]*, [2415], [3771], [2157], [3515], [2773], [2033], [2443], [2461], [3023], [3543], [2745], [2431], [3177]

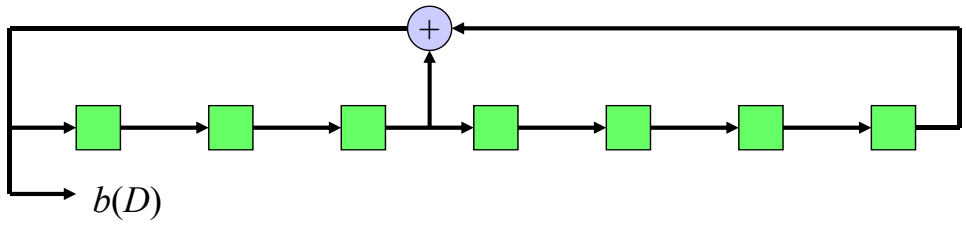
Polynomials Yielding *m*-Sequences (Cont.)

- The entries followed by an asterisk correspond to circuit implementation with only **two** feedback connections
  - Very useful for **high-speed** applications (**less propagation delay**)
  - [211]*:

octal binary coefficient

$$211 = (010001001) = (g_8, g_7, g_6, g_5, g_4, g_3, g_2, g_1, g_0)$$

$$g(D) = 1 + D^3 + D^7$$



Polynomials Yielding *m*-Sequences (Cont.)

- The list of primitive polynomials in Table 3-5 is not complete
- The number of **primitive polynomials** of degree r is

$$N_p = \frac{2^r - 1}{r} \prod_{i=1}^{J} \frac{p_i - 1}{p_i}, \quad 2^r - 1 = \prod_{i=1}^{J} p_i^{e_i}$$

  - where p_i are the **prime factors** of $2^r - 1$, e_i are positive integers
- For example $r = 9$:
  - The prime factors of $2^9 - 1 = 511$ are 7 and 73

  $\Rightarrow N_p = (511/9)(6/7)(72/73) = 48$

Polynomials Yielding *m*-Sequences (Cont.)

r	N_P	r	N_P
2	1	16	2,048
3	2	17	7,710
4	2	18	8,064
5	6	19	27,594
6	6	20	24,000
7	18	21	84,672
8	16	22	120,032
9	48	23	356,960
10	60	24	276,480
11	176	25	1,296,000
12	144	26	1,719,900
13	630	27	4,202,496
14	756	28	4,741,632
15	1,800	29	18,407,808

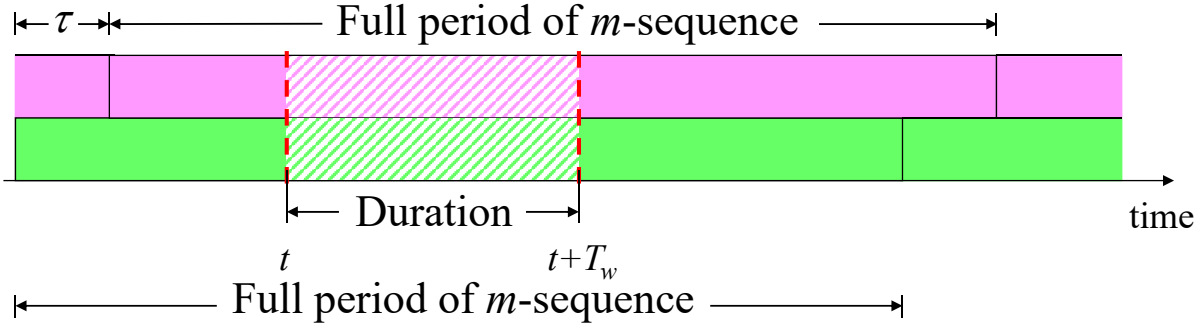
Partial Autocorrelation of *m*-Sequences

Partial Autocorrelation of *m*-Sequences

- **Partial autocorrelation**: a correlation over a **partial** **period**
  - The partial autocorrelation of $c(t)$ is defined by

$$R_c(\tau, t, T_w) = \frac{1}{T_w} \int_t^{t+T_w} c(\lambda) c(\lambda + \tau)\, d\lambda$$

  - where T_w is the **duration** of the correlation and t is the **starting time** of the correlation
  - It depends on the **size** and the **starting time** of the duration



Partial Autocorrelation of *m*-Sequences (Cont.)

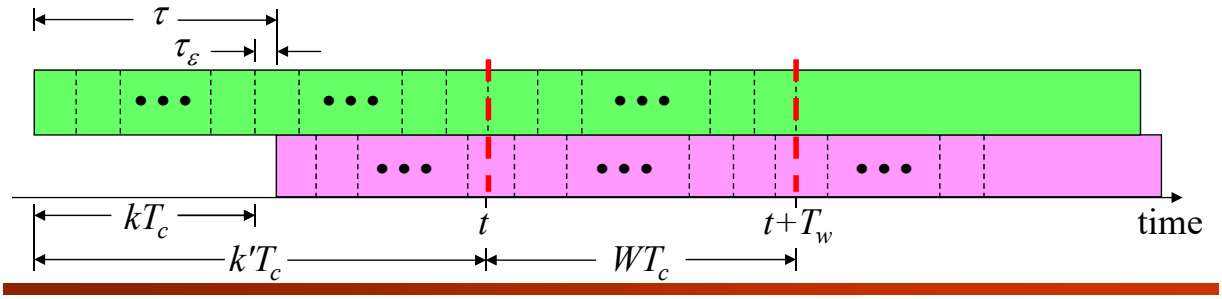
- Using $c(t) = \sum_{n=-\infty}^{\infty} a_n p(t - nT_c)$ and letting $\gamma = \lambda - t$

Change variable

- We have $R_c(\tau, t, T_w) = \frac{1}{T_w} \int_t^{t+T_w} c(\lambda) c(\lambda + \tau)\, d\lambda$

$$= \frac{1}{T_w} \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} a_m a_n \int_0^{T_w} p(\gamma + t - mT_c)\, p(\gamma + t + \tau - nT_c)\, d\gamma$$

- Let $\tau = \boldsymbol{k}T_c + \tau_\varepsilon$, $T_w = WT_c$, and assume that $t = \boldsymbol{k'}T_c$ (the **starting time** aligns to the chip interval)



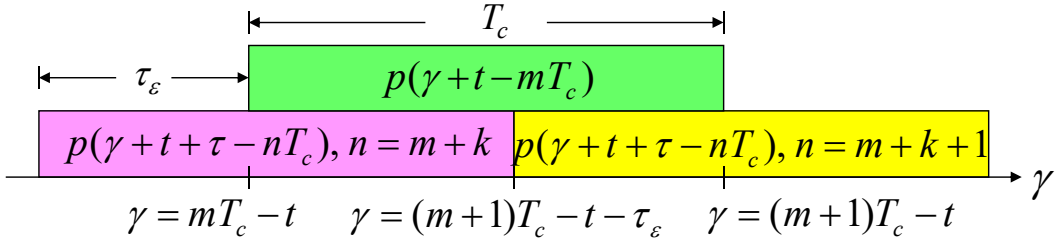
Partial Autocorrelation of *m*-Sequences (Cont.)

- Then, we have

$$R_c(\tau, t, T_w) \triangleq R_c(\tau_\varepsilon, k, k', W)$$

$$= \frac{1}{WT_c} \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} a_m a_n \int_0^{WT_c} p(\gamma - (m - k')T_c)\, p(\gamma - (n - k' - k)T_c + \tau_\varepsilon)\, d\gamma$$

- Because $p(t)$ takes value only for $0 \le t \le T_c$, and $0 \le \tau_\varepsilon \le T_c$
  - The integral is **nonzero** only for $n - k' - k = m - k'$ or $n - k' - k = m - k' + 1 \Rightarrow \boldsymbol{n = m + k}$ or $\boldsymbol{n = m + k + 1}$

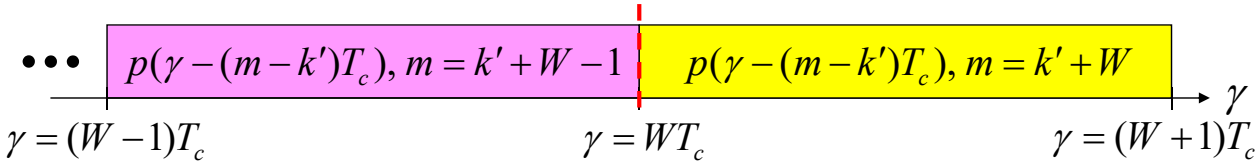


Partial Autocorrelation of *m*-Sequences (Cont.)

- Hence, we have

$$R_c(\tau_\varepsilon, k, k', W)$$
$$= \frac{1}{WT_c} \sum_{m=-\infty}^{\infty} a_m a_{m+k} \int_0^{WT_c} p(\gamma - (m-k')T_c)\, p(\gamma - (m-k')T_c + \tau_\varepsilon)\, d\gamma$$
$$+ \frac{1}{WT_c} \sum_{m=-\infty}^{\infty} a_m a_{m+k+1} \int_0^{WT_c} p(\gamma - (m-k')T_c)\, p(\gamma - (m-k'+1)T_c + \tau_\varepsilon)\, d\gamma$$

- Because $0 \le \gamma \le WT_c$ and $p(t)$ takes value only for $0 \le t \le T_c$
  - The integrand is **nonzero** when $0 \le (m-k')T_c \le (W-1)T_c$
  - The **limits** of m are reduced to $k' \le m \le W + k' - 1$



Partial Autocorrelation of *m*-Sequences (Cont.)

- For a fixed value of m,
  - First integral is nonzero: $(m-k')T_c \le \gamma \le (m+1-k')T_c - \tau_\varepsilon$
  - Second integral is nonzero: $(m+1-k')T_c - \tau_\varepsilon \le \gamma \le (m+1-k')T_c$

$$R_c(\tau_\varepsilon, k, k', W)$$
$$= \frac{1}{WT_c} \sum_{m=k'}^{W+k'-1} a_m a_{m+k} \int_{(m-k')T_c}^{(m+1-k')T_c - \tau_\varepsilon} p\left(\gamma - (m-k')T_c\right) p\left(\gamma - (m-k')T_c + \tau_\varepsilon\right) d\gamma$$
$$+ \frac{1}{WT_c} \sum_{m=k'}^{W+k'-1} a_m a_{m+k+1} \int_{(m+1-k')T_c - \tau_\varepsilon}^{(m+1-k')T_c} p\left(\gamma - (m-k')T_c\right) p\left(\gamma - (m-k'+1)T_c + \tau_\varepsilon\right) d\gamma$$
$$= \frac{1}{W} \sum_{m=k'}^{W+k'-1} a_m a_{m+k} \times \left(1 - \frac{\tau_\varepsilon}{T_c}\right) + \frac{1}{W} \sum_{m=k'}^{W+k'-1} a_m a_{m+k+1} \times \left(\frac{\tau_\varepsilon}{T_c}\right)$$
$$= \left(1 - \frac{\tau_\varepsilon}{T_c}\right) \theta_b(k, k', W) + \left(\frac{\tau_\varepsilon}{T_c}\right) \theta_b(k+1, k', W)$$

Partial Autocorrelation of *m*-Sequences (Cont.)

- The **discrete partial autocorrelation** function of a sequence $b(D)$ is defined by

$$\theta_b(k,k',W)=\frac{1}{W}\sum_{m=k'}^{W+k'-1}a_m a_{m+k}$$

- The partial autocorrelation function is

$$R_c(\tau_\varepsilon,k,k',W)=(1-\frac{\tau_\varepsilon}{T_c})\theta_b(k,k',W)+\frac{\tau_\varepsilon}{T_c}\theta_b(k+1,k',W)$$

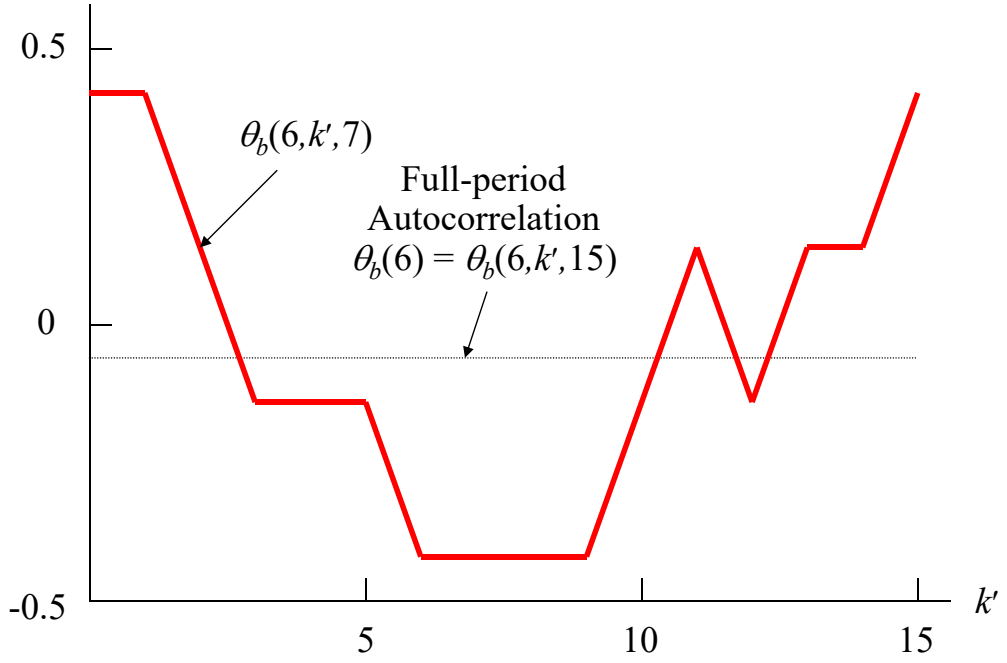
- $R_c(\tau, t, T_w)$ can be calculated from the knowledge of $\theta_b(k, k', W)$
  - The value of $\theta_b(k, k', W)$ is the number of **agreements** N_A minus the number of **disagreements** N_D between **b**(0) and **b**(k)
    - over the window **beginning** at k' and **ending** at $k'+W-1$

Example

- Consider a primitive polynomial $g(D) = 1+D+D^4$
  - Initial condition $a(D) = 1$
- Evaluate $\theta_b(k, k', W) = \theta_b(6, k', 7)$ for the 15-bit *m*-sequence
- **b**(0) = 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0
- **b**(6) = 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1
- **b**(0) + **b**(6) = 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1
  - 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 $\Rightarrow \theta_b(6, 1, 7) =(5 – 2)/7 = 3/7$
  - 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 $\Rightarrow \theta_b(6, 7, 7) =(2 – 5)/7 = –3/7$
- The partial autocorrelation function is **not two-valued** and its **variation** is a function of the **window size** and the **window placement**

Example (Cont.)

- The discrete partial autocorrelation function $\theta_b(6, k', 7)$



Statistics of Partial Autocorrelation

$$\theta_b(k,k',W) = \frac{1}{W}\sum_{m=k'}^{W+k'-1} a_m a_{m+k}$$

- By **Property II** of *m*-sequences, the modulo-2 sum of **b**(0) and **b**(k) is **another phase b**(q) of the same sequence

$$\theta_b(k,k',W) = \frac{1}{W}\sum_{i=0}^{W-1} a_{i+q+k'}, \quad a_i = (-1)^{b_i}, (0 \to +1, 1 \to -1)$$

- Averaging over all k':

$$\overline{\theta_b(k,k',W)} = \frac{1}{N}\sum_{k'=0}^{N-1}\frac{1}{W}\sum_{i=0}^{W-1} a_{i+q+k'} = \frac{1}{WN}\sum_{i=0}^{W-1}\sum_{k'=0}^{N-1} a_{i+q+k'}$$

- By **Property I** of *m*-sequences, the inner summation is –1 for all i and q
  - The mean is $\overline{\theta_b(k,k',W)} = -\frac{1}{N}$

Statistics of Partial Autocorrelation (Cont.)

- The second moment of $\theta_b(k,k',W)$ is:

$$\overline{\theta_b^2(k,k',W)} = \frac{1}{N}\sum_{k'=0}^{N-1}\theta_b^2(k,k',W)$$

$$\theta_b(k,k',W) = \frac{1}{W}\sum_{m=k'}^{W+k'-1} a_m a_{m+k} = \frac{1}{W}\sum_{i=0}^{W-1} a_{i+q+k'},$$

$$= \frac{1}{N}\sum_{k'=0}^{N-1}\left[\frac{1}{W^2}\sum_{m=0}^{W-1} a_{m+q+k'}\sum_{n=0}^{W-1} a_{n+q+k'}\right]$$

$$= \frac{1}{W^2 N}\sum_{m=0}^{W-1}\sum_{n=0}^{W-1}\sum_{k'=0}^{N-1} a_{m+q+k'} \times a_{n+q+k'}$$

  - When $\boldsymbol{m = n}$, $a_{m+q+k'} \times a_{n+q+k'} = 1$ for all q and $k' \Rightarrow$ the inner summation is equal to N **(W terms)**
  - When $\boldsymbol{m \neq n}$, $a_{m+q+k'} \times a_{n+q+k'}$, $k' = 0 \sim N-1$, is the modulo-2 sum of **b**(0) and **b**(k) (another phase of the same sequence) $\Rightarrow$ the inner summation is equal to -1 **($W^2 - W$ terms)**

Statistics of Partial Autocorrelation (Cont.)

$$\overline{\theta_b^2(k,k',W)} = \frac{1}{W^2 N}\left[W \times N + (W^2 - W)\times(-1)\right] = \frac{1}{W}\left(1 - \frac{W-1}{N}\right)$$

- The variance over k' of $\theta_b(k,k',W)$ is

$$\mathrm{Var}\left[\theta_b(k,k',W)\right] = \overline{\theta_b^2(k,k',W)} - \left[\overline{\theta_b(k,k',W)}\right]^2 = \frac{1}{W}\left(1 - \frac{W-1}{N}\right) - \frac{1}{N^2}$$

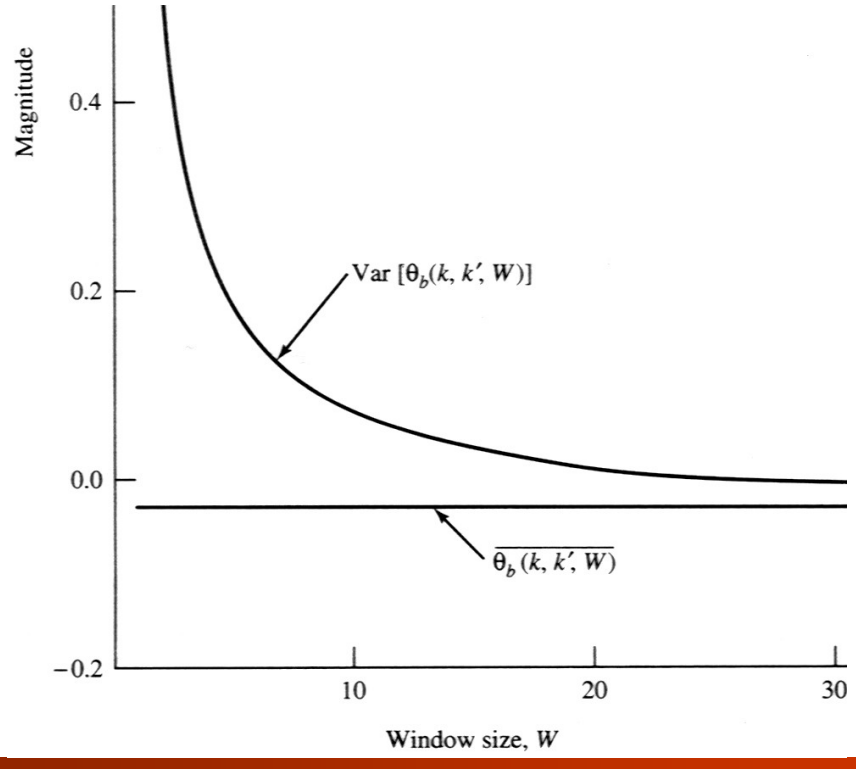
  - When $W = N$

$$\mathrm{Var}\left[\theta_b(k,k',W)\right] = \frac{1}{N}\left(1 - \frac{N-1}{N}\right) - \frac{1}{N^2}$$

$$= \frac{1}{N^2} - \frac{1}{N^2} = 0$$

  - $W \to N \Rightarrow \mathrm{var}[\theta_b(k,k',W)] \to 0$ as expected

Statistics of Partial Autocorrelation (Cont.)



Power Spectrum of *m*-Sequences with Timing Offset

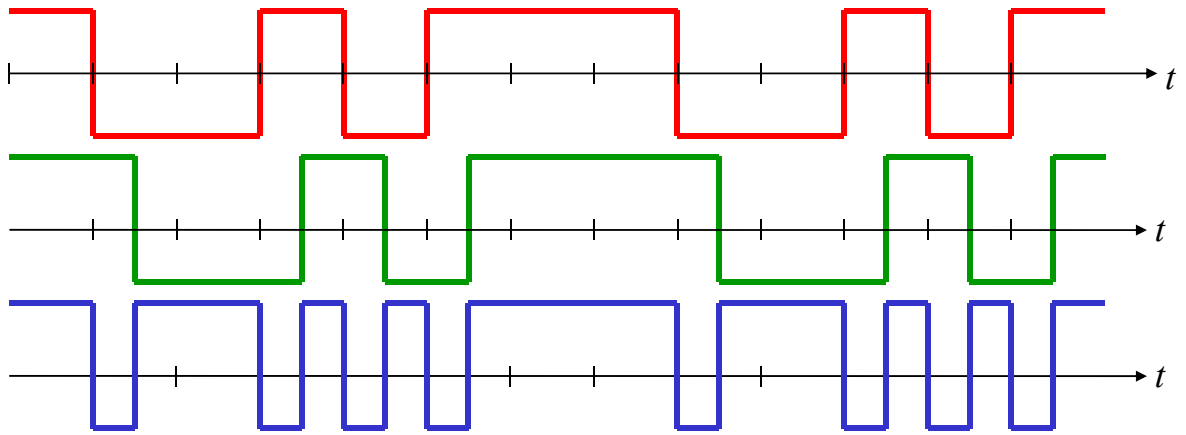
Power Spectrum of $c(t)c(t+\varepsilon)$

- The **despreading** operation is accomplished by correlating the received signal with a replica of $c(t)$
- The replica of $c(t)$ may be **offset** in phase by some fraction of a code period
- The power spectrum of the output $b(t,\varepsilon) = c(t)c(t+\varepsilon)$ of the despreading correlator for $|\varepsilon| \le T_c$ is (Appendix D)

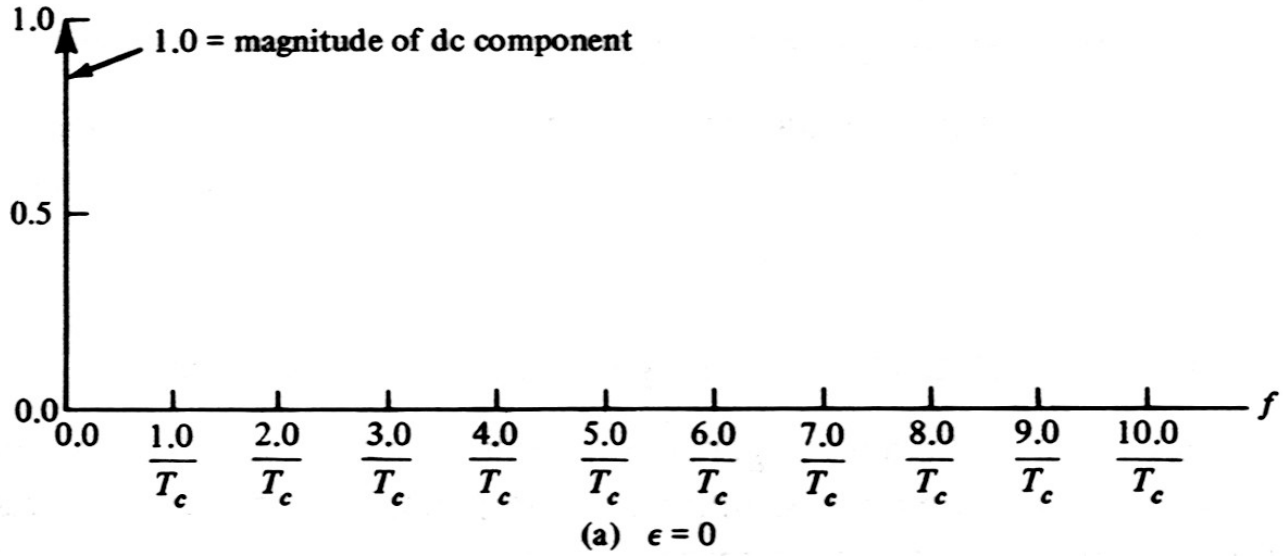
$$S_b(f,\varepsilon) = \left[1-\left(1+\frac{1}{N}\right)\frac{|\varepsilon|}{T_c}\right]^2 \delta(f)$$
$$+\left(1+\frac{1}{N}\right)\left(\frac{|\varepsilon|}{T_c}\right)^2 \sum_{n=-\infty,n\neq 0}^{\infty} \operatorname{sinc}^2\left(nf_c|\varepsilon|\right)\delta\left(f-nf_c\right)$$
$$+\frac{N+1}{N^2}\left(\frac{|\varepsilon|}{T_c}\right)^2 \sum_{m=-\infty,m\neq 0}^{\infty} \operatorname{sinc}^2\left(\frac{mf_c}{N}|\varepsilon|\right)\delta\left(f-\frac{mf_c}{N}\right)$$

Power Spectrum of $c(t)c(t+\varepsilon)$ (Cont.)

- For $\varepsilon = 0$: a single spectral line at **zero frequency**
- For $\varepsilon = T_c$: $b(t,\varepsilon)$ is simply a phase-shifted replica of $c(t)$, so $S_b(f, \varepsilon) = S_c(f)$
- For $\varepsilon \neq 0$ or T_c: significantly **wider** than the spectrum of $c(t)$
  - The $b(t, \varepsilon)$ **transits more rapidly** than $c(t)$

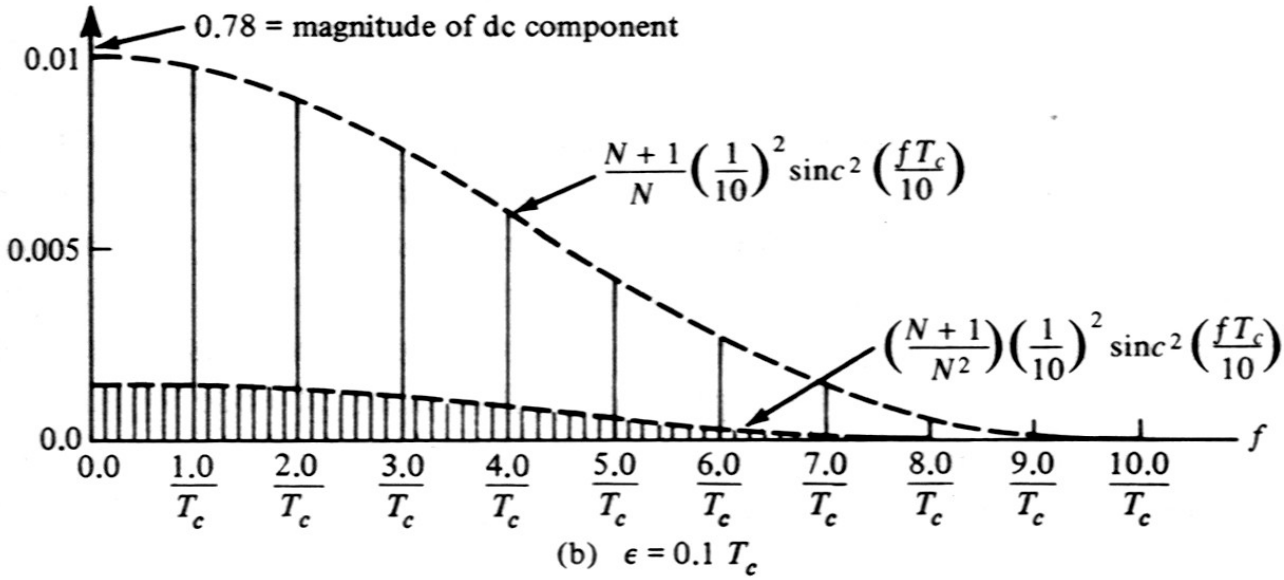


Power Spectrum of $c(t)c(t+\varepsilon)$ (Cont.)



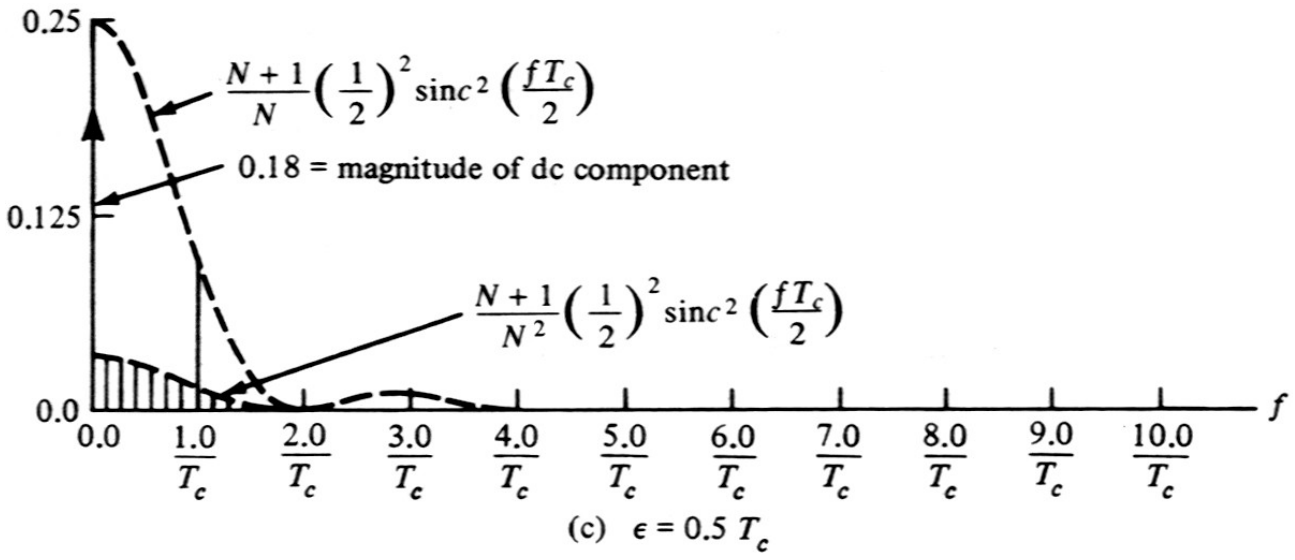
(a) $\epsilon = 0$

Power Spectrum of $c(t)c(t+\varepsilon)$ (Cont.)



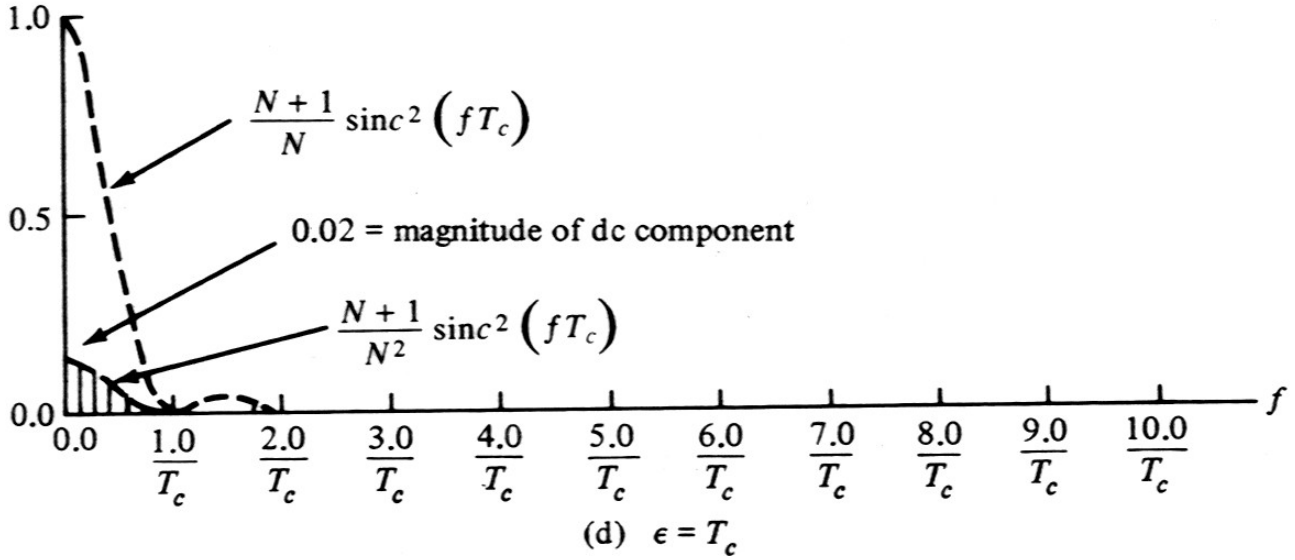
(b) $\epsilon = 0.1\ T_c$

Power Spectrum of $c(t)c(t+\varepsilon)$ (Cont.)



(c) $\epsilon = 0.5\,T_c$

Power Spectrum of $c(t)c(t+\varepsilon)$ (Cont.)



(d) $\epsilon = T_c$

m-Sequences with Specific Delays

Specific Delays of an *m*-Sequences

- To generate two **different** phases of an *m*-sequence
  - The phase difference could be impractical **(too large)** to generate using a **shift register** or a **delay line**
  - For example:
    - Assume that $m = 15 \Rightarrow P = 2^{15} - 1 = 32767$
    - The desired phase difference: $2^{14} = \mathbf{16384}$

$b(D)$ delayed by k chips

$$b(D) = \cdots + b_{-2}D^{-2} + b_{-1}D^{-1} + b_0 + b_1D^1 + b_2D^2 + \cdots$$

$$D^k b(D) = \cdots + b_{-(k+2)}D^{-2} + b_{-(k+1)}D^{-1} + b_{-k} + b_{1-k}D^1 + b_{2-k}D^2 + \cdots$$

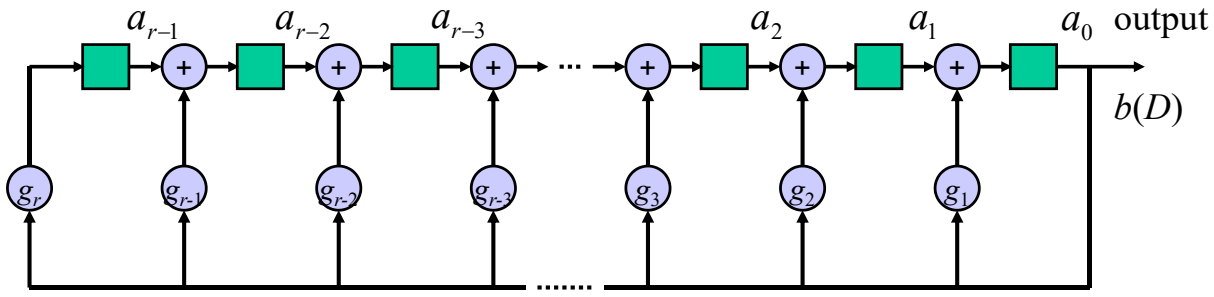
- Two techniques are discussed for generating specific delays of an *m*-sequence
  - Calculate the required shift-register **initial conditions**
  - Use the **shift-and-add property** of *m*-sequence generators

Determining the Initial Condition

- Consider the sequence generator for **G**-Config.:
  - Given an initial condition $a(D)$, the output is

$$b(D) = a(D)/g(D)$$

- Another initial condition $a'(D)$ will produce another output sequence $b'(D) = D^k\, b(D)$
  - $b'(D)$ is the sequence $b(D)$ **delayed by *k* chips**



Determining the Initial Condition (Cont.)

- Note that the shift register states of an *m*-sequence generator are associated with elements of the **extension field** GF(2^r) defined by the **primitive polynomial *g*(*D*)**
- Let $q(D)$ represent an element of GF(2^r)
  - The element ***l* units later**, denoted by $q'(D)$, satisfies

$$D^l q(D) = p(D)g(D) + q'(D)$$

- From **G**-Config., considering the initial condition $a(D)$, the **next shift-register state** $a'(D)$ satisfies (based on the circuit)

$$a'(D) = D^{-1}a(D) + D^{-1}a_0 g(D)$$

or equivalently,

$$D^{-1}a(D) = D^{-1}a_0 g(D) + a'(D)$$

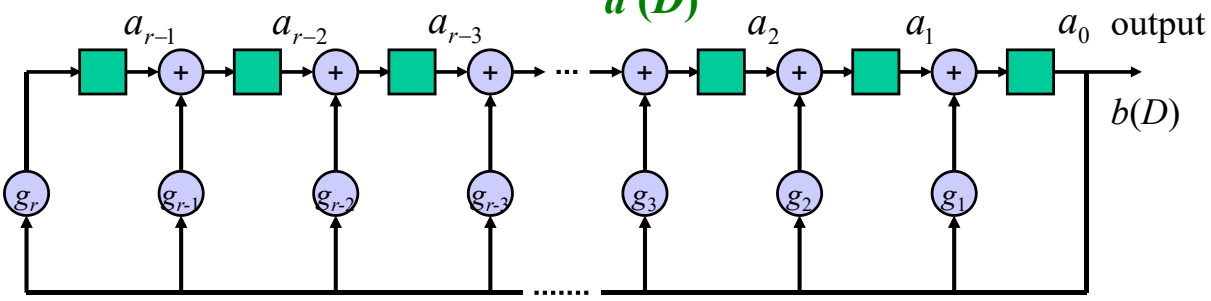
quotient **remainder**

Determining the Initial Condition (Cont.)

$$\because D^{-1}a(D) = \boxed{a_{r-1}D^{r-2} + a_{r-2}D^{r-3} + \cdots + a_2 D + a_1} + a_0 D^{-1}$$

$$D^{-1}a_0 g(D) = a_0 g_r D^{r-1} + a_0 g_{r-1} D^{r-2} + a_0 g_{r-2} D^{r-3} + \cdots + a_0 g_2 D + a_0 g_1 + a_0 D^{-1}$$

$$a'(D)$$



- Because $g(D)$ is a primitive polynomial:

$$D^{2^r-1} \Big/ g(D) = 1 \Rightarrow D^{-1} = D^{2^r-2}$$

The same elements of GF(2^r)

$$D^{2^r-2}a(D) = a_0 D^{2^r-2} g(D) + a'(D)$$

quotient **remainder**

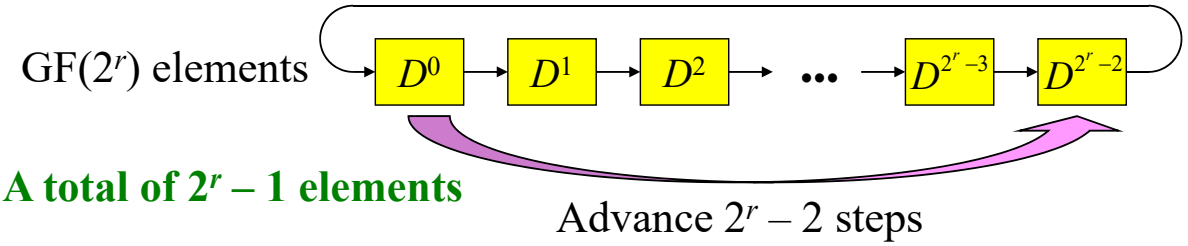
Determining the Initial Condition (Cont.)

- The output sequence **advancing** one unit is equivalent to that the shift-register contents are in the **next state** (after one clock cycle)

Equivalent Elements

$a(D)$ advance $2^r - 2$ steps on GF(2^r) → $D^{2^r-2}a(D) = a_0 D^{2^r-2} g(D) + a'(D)$ ← $a(D)$ after one clock cycle

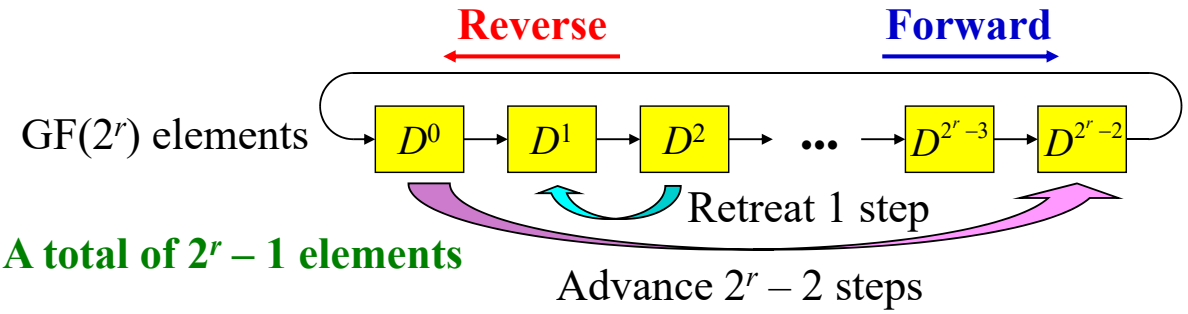
- For each clock cycle, the contents of the shift register **advance $2^r - 2$ steps** through the sequence of **elements of GF(2^r)**



A total of $2^r - 1$ elements

Determining the Initial Condition (Cont.)

- To **advance $2^r - 2$ steps** is equivalent to **retreat 1 step**



A total of $2^r - 1$ elements

- For the output sequence to **advance** one unit (or to **delay** one unit) of time of the sequence
  - **$2^r - 2$ elements shift** (or **1 elements shift**) on GF(2^r)

  $\Rightarrow$Cycle through the elements of GF(2^r) in **reverse** or **forward** order

Example

- Example 3-17

  $g(D) = 1 + D + D^4$
- $2^r - 2 = 2^4 - 2 = 14$
- Advance one cycle

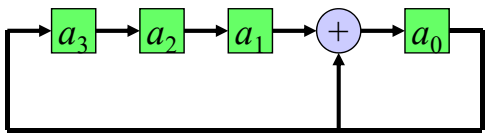
$\Rightarrow$ 14 elements shift on GF(2^m)

$\Rightarrow \boldsymbol{D^{14}}$

$\Rightarrow D^{28} = D^{15} \times D^{13} = \boldsymbol{D^{13}}$

$\Rightarrow \boldsymbol{D^{12}}$

$\Rightarrow \ldots$



Cycle	Register state	$a(D)$	Element of GF(2^4)
0	0 0 0 1	1	D^0
1	1 0 0 1	$1 + D^3$	D^{14}
2	1 1 0 1	$1 + D^2 + D^3$	D^{13}
3	1 1 1 1	$1 + D + D^2 + D^3$	D^{12}
4	1 1 1 0	$D + D^2 + D^3$	D^{11}
5	0 1 1 1	$1 + D + D^2$	D^{10}
6	1 0 1 0	$D + D^3$	D^9
7	0 1 0 1	$1 + D^2$	D^8
8	1 0 1 1	$1 + D + D^3$	D^7
9	1 1 0 0	$D^2 + D^3$	D^6
10	0 1 1 0	$D + D^2$	D^5
11	0 0 1 1	$1 + D$	D^4
12	1 0 0 0	D^3	D^3
13	0 1 0 0	D^2	D^2
14	0 0 1 0	D	D^1

Determining the Initial Condition (Cont.)

- The problem of finding the shift-register **initial conditions** corresponding to a particular **advance** or **delay**
  - Reduce to a problem of manipulating elements of GF(2^r)
- With $a(D)$ defining one initial condition and $a'(D)$ defining the initial condition corresponding to an **advance of *k* units**
  - $a'(D)$ is the remainder of dividing $D^{k(2^r-2)}a(D)$ by $g(D)$
  - $D^{k(2^r-2)}$ can be **reduced** using the fact that $D^{2^r-1}=1$
- The load corresponding to a **delay of *k* units**
  - $a'(D)$ is the **remainder** of dividing $D^k a(D)$ by $g(D)$

Example

- Consider $g(D) = 1 + D + D^4$ with an initial condition $a(D) = 1$
  - Find the initial condition: **advance** 20 units and **delay** 20 units
- Period 15 units, an advance of 20 units $\Rightarrow$ an advance of 5 units
  - $D^{k(2^r-2)} = D^{5\times 14} = D^{70} = D^{10}$. Thus, $a'(D) = a(D)D^{10} = D^{10}$
  - $D^{10} = 1 + D + D^2$
- Period 15 units, a delay of 20 units $\Rightarrow$ a delay of 5 units
  - $a'(D) = a(D)D^5 = D^5 = D + D^2$
- This technique **works only** for the configuration of **G**-Config.
- For the configuration of **F**-Config.: $a(D)$ of **F**-Config. $\Rightarrow$ $a(D)$ of **G**-Config. $\Rightarrow$ $a'(D)$ of **G**-Config. $\Rightarrow$ $a'(D)$ of **F**-Config.

Determining with Shift-and-Add Property

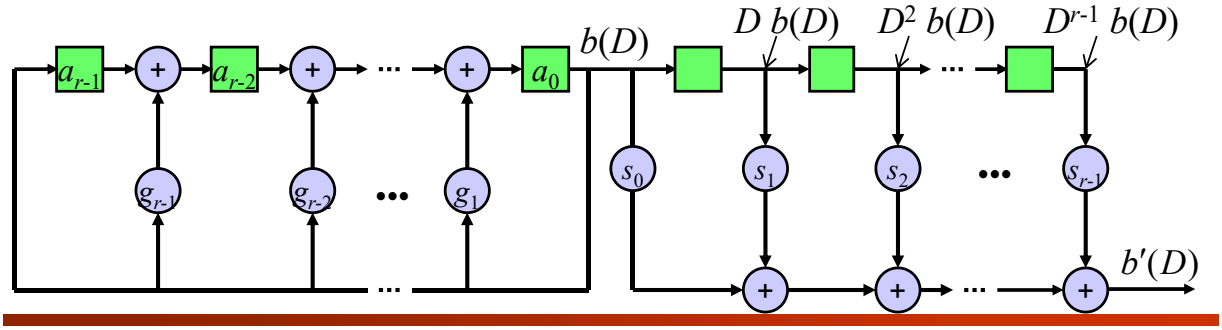
- Consider a r-stage shift-register generator shown as follows:
  - The output $b(D) = a(D)/g(D)$
  - To determine the connection polynomial which is used to obtain $b'(D) = D^k\, b(D)$

No. of possible $s(D) = 2^r - 1 \Rightarrow$ Cover all possible code phases

- The output $b'(D)$ is defined by

$$b'(D) = s_0 b(D) + s_1 D b(D) + s_2 D^2 b(D) + \cdots + s_{r-1} D^{r-1} b(D) = s(D) b(D)$$

  - $s(D)$ is a **connection polynomial** used to obtain $b'(D)$



Determining with Shift-and-Add Property (Cont.)

- Based on the result according to the method of **determining the initial condition**,

$$b'(D) = \frac{a'(D)}{g(D)} = s(D) b(D) = s(D) \frac{a(D)}{g(D)}$$

- The initial conditions are related by $a'(D) = s(D)a(D)$ **(for any pair of $a(D)$ and $a'(D)$)**
  - $a(D)$ can be arbitrarily chosen to be $a(D) = 1$

  $\Rightarrow s(D) = a'(D)$
  - $a'(D)$ can be determined by using the previous technique

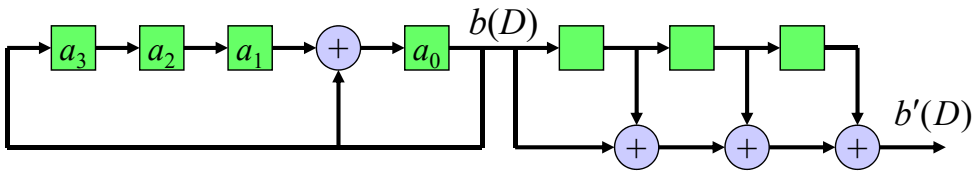
Example

- Consider $g(D) = 1 + D + D^4$
  - To obtain another sequence which is delay by 12 units
- Period 15 units, delay 12 units

  $\Rightarrow a'(D) = a(D)D^{12} = D^{12} = 1 + D + D^2 + D^3$

  $\Rightarrow s(D) = a'(D) = 1 + D + D^2 + D^3$
- It is presumed that the output $b(D)$ is taken from the **rightmost** shift register of the generator (**Galois Configuration**)

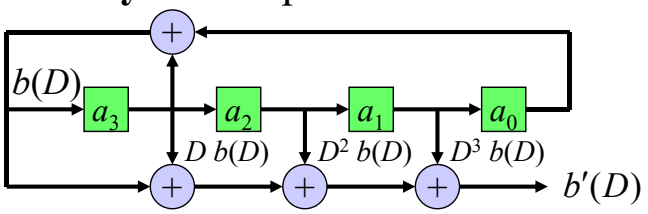
  $\Rightarrow$ **Delays of $b(D)$** must be added



Example (Cont.)

- For the alternative configuration (**Fibonacci Configuration**), the output is taken to be the input to the **leftmost** shift register stage
  - The delays of the sequence $b(D)$ required to produce $b'(D)$ are **available** within the generator

  $\Rightarrow$ **No external delays** are required



$b(D)$ = 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 …
$D\,b(D)$ = 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 …
$D^2\,b(D)$ = 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 …
$D^3\,b(D)$ = 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 …
$b'(D)$ = 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 …

Security Issue

Security of Maximal-Length Sequences

- SS systems are used to protect digital transmissions from being jammed or to preclude unintended reception
  - It can only be met if the jammer or unintended receiver **does not** have knowledge of the spreading waveform $c(t)$
- When the **jammer** or **interceptor** can receive a relatively **noise-free** copy of the transmitted signal
  - The spreading code **feedback connections** and **initial phase** can be determined
- Suppose that the unintended party knows
  - An **uncorrupted version** of the spreading code $b_0, b_1, b_2, \ldots$
  - The **period of the sequence** (by accurately measuring the received power spectrum) $\Rightarrow m$ of m-sequences is obtained

Security of Maximal-Length Sequences (Cont.)

$$
\begin{aligned}
b_i &= b_{i\text{-}1}g_1 + b_{i\text{-}2}g_2 + \cdots + b_{i\text{-}m}g_m \\
b_{i+1} &= b_i g_1 + b_{i\text{-}1}g_2 + \cdots + b_{i\text{-}m+1}g_m \\
b_{i+2} &= b_{i+1}g_1 + b_i g_2 + \cdots + b_{i\text{-}m+2}g_m \\
&\vdots \\
b_{i+m-1} &= b_{i+m-2}g_1 + b_{i+m-3}g_2 + \cdots + b_{i-1}g_m
\end{aligned}
$$

- After m such equations have been obtained
  - The m unknowns, g_1 through g_m, can be **solved**
- The number of symbols which must be received is $2m$
  - $b_{i\text{-}m}, \ldots, b_i, \ldots, b_{i+m\text{-}1}$
  - which is **much shorter** than the **period** $N = 2^m - 1$

Example

- Consider that the sequence '**0 1 1 0 0 1 0 0**' is received
  - The known **period** of the m-sequence is 15 $\Rightarrow m = 4$

$$
\begin{aligned}
(1)\quad & 0 = 0\times g_1 + 1\times g_2 + 1\times g_3 + 0\times g_4 \\
(2)\quad & 1 = 0\times g_1 + 0\times g_2 + 1\times g_3 + 1\times g_4 \\
(3)\quad & 0 = 1\times g_1 + 0\times g_2 + 0\times g_3 + 1\times g_4 \\
(4)\quad & 0 = 0\times g_1 + 1\times g_2 + 0\times g_3 + 0\times g_4
\end{aligned}
$$

- Adding (1) and (4) yields $g_3 = 0$
- Substituting $g_3 = 0$ into (1) yields $g_2 = 0$
- Substituting $g_2 = g_3 = 0$ into (2) yields $g_4 = 1$
- Substituting $g_2 = g_3 = 0$ and $g_4 = 1$ into (3) yields $g_1 = 1$
- The generating polynomial is $g(D) = 1 + D + D^4$

Gold Codes

Cross-correlation Spectrum

- Channel resources may be shared by using spread spectrum techniques **(Code Division Multiple Access, CDMA)**
  - Users are each assigned a **different** spreading code
  - To find a set of codes with as **little** **mutual interference** as possible
- The cross-correlation (mutual interference) between two codes cannot be guaranteed for using two **arbitrary** m-sequences
  - Or using two **arbitrary** segments of the same m-sequence with a specific code phase offset
- Gold codes
  - Exist relatively **large** sets of codes
  - Have **well controlled** cross-correlation properties

Cross-correlation Spectrum (Cont.)

- The **full-period** cross-correlation is

$$\theta_{bb'}(k) = \frac{1}{N}\sum_{n=0}^{N-1} a_n a'_{n+k}$$

- The **cross-correlation spectrum**: a list of **all** possible values of $\theta_{bb'}(k)$ and the **number** of values of k which yield that particular cross-correlation
- For example, the autocorrelation spectrum for an ***m*-sequence** ($b=b'$) is

$$\begin{cases} 1.0 & \text{occurs 1 time;} \\ -\dfrac{1}{N} & \text{occurs } N-1 \text{ times;} \end{cases}$$

Decimation

- Consider an m-sequence **b** of length N, and a second sequence **b′** obtained by **sampling every *q*-th symbol** of **b**
  - The second sequence is said to be a **decimation** of the first, and **b′** = **b**[q]

b = **0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 …**

b′ = b[3] = **0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 …**

b = **0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 …**

b′ = b[4] = **0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 …**

- The decimation of an m-sequence **may** or **may not** yield another m-sequence
  - When does yield an m-sequence ⇒ **proper decimation**
  - **b′** = **b**[q] has a period N if and only if **gcd(*N*, *q*) = 1**

Decimation (Cont.)

- Proper decimation by **odd integers *q*** will give **all** of the *m*-sequences of period N
- **Any pair of *m*-sequences** having the same period N can be related by **b′** = **b**[q] **for some *q***

Cross-correlation Spectrum

- The cross-correlation spectrum of pairs of *m*-sequences can be three-valued, four-valued, or many-valued
- Certain special pairs of *m*-sequences whose cross-correlation spectrum is **three-valued** are referred to as
  - **Preferred pairs** of *m*-sequences

$$\begin{cases} -\dfrac{1}{N}t(n); \\ -\dfrac{1}{N}; \\ \dfrac{1}{N}[t(n)-2]; \end{cases} \qquad t(n)=\begin{cases} 1+2^{0.5(n+1)}, & \text{for } n \text{ odd} \\ 1+2^{0.5(n+2)}, & \text{for } n \text{ even} \end{cases}, \qquad N=2^n-1$$

Cross-correlation Spectrum (Cont.)

- The conditions for defining **a preferred pair b** and **b′** are
  - (1) $n \neq 0 \bmod 4 \Rightarrow n$ is odd or $n = 2 \bmod 4$

$$N = 2^n - 1$$

  - (2) $\mathbf{b}' = \mathbf{b}[q]$, where q is odd and either

$$q = 2^k + 1$$

or

$$q = 2^{2k} - 2^k + 1$$

  - (3) $$\gcd(n,k) = \begin{cases} 1 & \text{for } n \text{ odd} \\ 2 & \text{for } n = 2 \bmod 4 \end{cases}$$

Example

- Consider an *m*-sequence with
  - The period $N = 31 \Rightarrow n = 5$
- By Table 3-5, the entry [45] (100101) may be used to generate an *m*-sequence of length 31
- The decimation $\mathbf{b}' = \mathbf{b}[3]$ is **proper**, (**b**, **b**[3]): a candidate pair
  - First condition: $n = 1 \bmod 4$
  - Second condition: q is odd and $q = 2^k + 1$ for $k = 1$
  - Third condition: gcd(5, 1) = 1

b = 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 …

b′ =**b**[3]= 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 …

- Can you find the generating polynomial ***g′(D)*** ?

Example (Cont.)

$$t(n)=\begin{cases}1+2^{0.5(n+1)} & \text{for } n \text{ odd}\\ 1+2^{0.5(n+2)} & \text{for } n \text{ even}\end{cases},\quad \Rightarrow t(n)=1+2^{0.5(5+1)}=9$$

- For **any phase shift**, the cross-correlation takes on one of the three values: – 9/31, – 1/31 or + 7/31

b = 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 …
b′(3) = 1 0 1 0 1 0 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 1 0 1 …
bb′(3) = 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 1 0 0 0 1 0 1 0 1 …
$\Rightarrow$ + 7/31

b = 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 …
b′(8) = 0 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 1 0 1 …
bb′(8) = 1 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 0 0 1 0 1 …
$\Rightarrow$ – 1/31

Gold Codes

- Let $b(D)$ and $b'(D)$ represent a **preferred pair** of m-sequences having period $N = 2^n - 1$
- The family of codes ($N + 2$ codes) defined by

$$b(D);\quad b'(D);\quad b(D)+b'(D);$$
$$b(D)+Db'(D);$$
$$b(D)+D^2b'(D);$$
$$\vdots$$
$$b(D)+D^{N-1}b'(D);$$

  is called **the set of Gold codes** for this preferred pair of m-sequences
  - $D^j\, b'(D)$ represents a **phase shift** of $b'(D)$ by j units

Gold Codes (Cont.)

- **Any pair of codes** in the set have a **three-valued** cross-correlation spectrum

$$\begin{cases} -\frac{1}{N}t(n); \\ -\frac{1}{N}; \\ \frac{1}{N}[t(n)-2]; \end{cases} \qquad t(n)=\begin{cases} 1+2^{0.5(n+1)}, & \text{for } n \text{ odd} \\ 1+2^{0.5(n+2)}, & \text{for } n \text{ even} \end{cases}, \quad N=2^n-1$$

Another phase of *b*(*D*)

$D^i b(D) \oplus b'(D)$; **Preferred pair**

Another phase of *b′*(*D*)

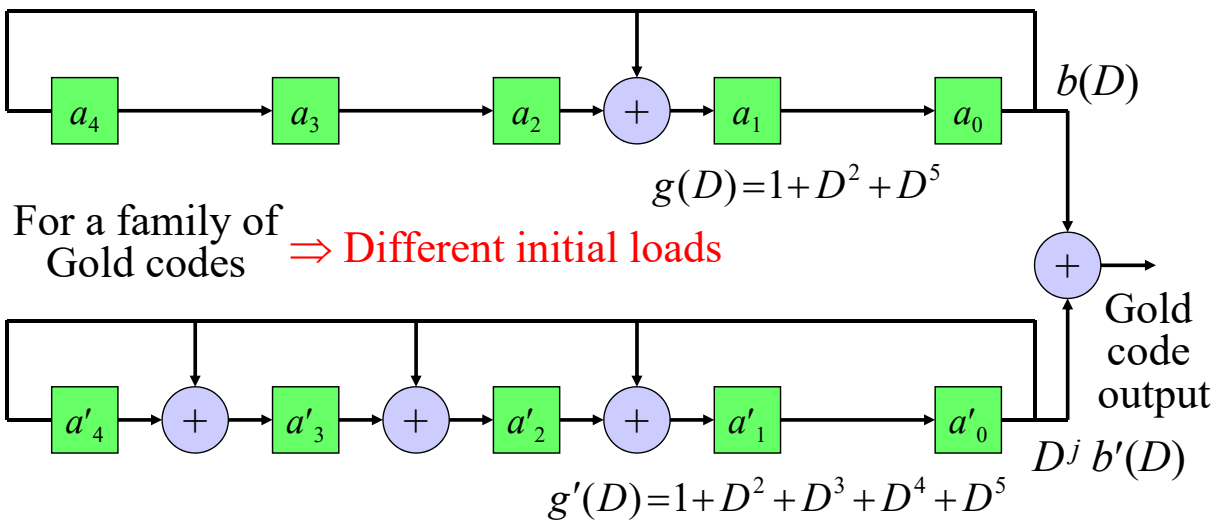
$$D^i b(D) \oplus [b(D) \oplus D^j b'(D)] = D^k b(D) \oplus D^j b'(D);$$

$$D^i b'(D) \oplus [b(D) \oplus D^j b'(D)] = b(D) \oplus D^k b'(D);$$

$$D^i [b(D) \oplus D^j b'(D)] \oplus [b(D) \oplus D^k b'(D)] = D^l b(D) \oplus D^m b'(D);$$

Gold Code Generator

- A typical shift register configuration used to generate a family of Gold codes



Gold Codes (Cont.)

- The complete family of Gold codes is obtained using **different initial loads** of either of the shift registers
  - Code $b(D)$ is obtained by choosing some nonzero $a(D)$ for upper generator and $a'(D) = 0$ for lower generator
  - Code $b'(D)$ is obtained by choosing some nonzero $a'(D)$ for lower generator and $a(D) = 0$ for upper generator
  - The other N codes are obtained using **the same** $a(D)$ used for $b(D)$ with **all possible** nonzero $a'(D)$
  - There are a total of $\boldsymbol{N+2}$ **codes** in any family of Gold codes
- The period of any code in the family is N
  - The same as the period of the m-sequences

Nonlinear Code Generators

Nonlinear Code Generators

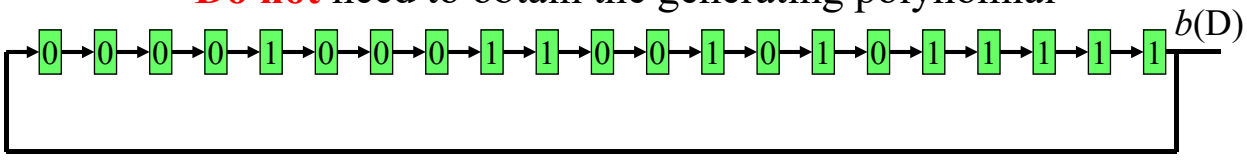
- The feedback connections for an n-stage maximal-length code can be easily determined from the knowledge of $2n$ **successive** code symbols
  - m-sequences are never used when a high degree of **security** is required
- **Nonlinear** spreading codes increase security through increased **complexity**
- There are two approaches:
  - Develop codes that **cannot be described** by a simple linear relationship
  - Develop codes for which ***r* is so large** (computationally impossible): r is the degree of the polynomial

Nonlinear Code Generators (Cont.)

- For a periodic sequence of period $2^r - 1$, it can be generated by a **recirculating shift register** of length $2^r - 1$
  - Consider a periodic sequence:
    1 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0 1 0 0 0 0 …
  - Can be generated using a 21-stage recirculating shift register

$$b(D) = \frac{1 + D + D^2 + D^3 + D^4 + D^6 + D^8 + D^{11} + D^{12} + D^{16}}{1 + D^{21}}$$

  - The ratio is $g(D) = 1 + D^{21}$: **only one connection** from the **last** stage to the input of the **first** stage
    - **Do not** need to obtain the generating polynomial



Nonlinear Code Generators (Cont.)

- The denominator can be factored into two terms and

$$b(D) = \frac{1}{1 + D + D^5}$$

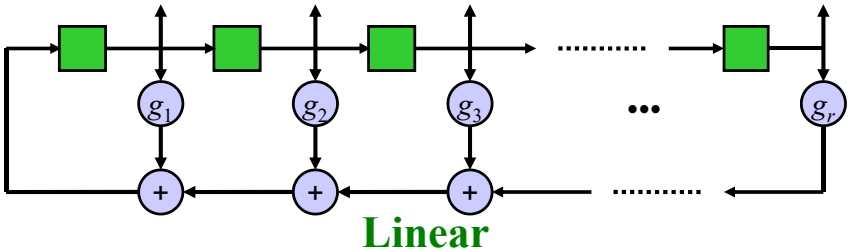
  - It can be simply generated by a **5-stage** shift register
- The denominator can be further factored into two terms:

$$b(D) = \frac{1}{(1 + D + D^2)(1 + D^2 + D^3)}$$

  - It can be generated by **two separate sequence generators** whose outputs are **modulo-2** summed
- **Any periodic sequence** can be generated by a linear feedback shift register
- If the use of **nonlinear elements** is allowed, it may be much more efficiently generated

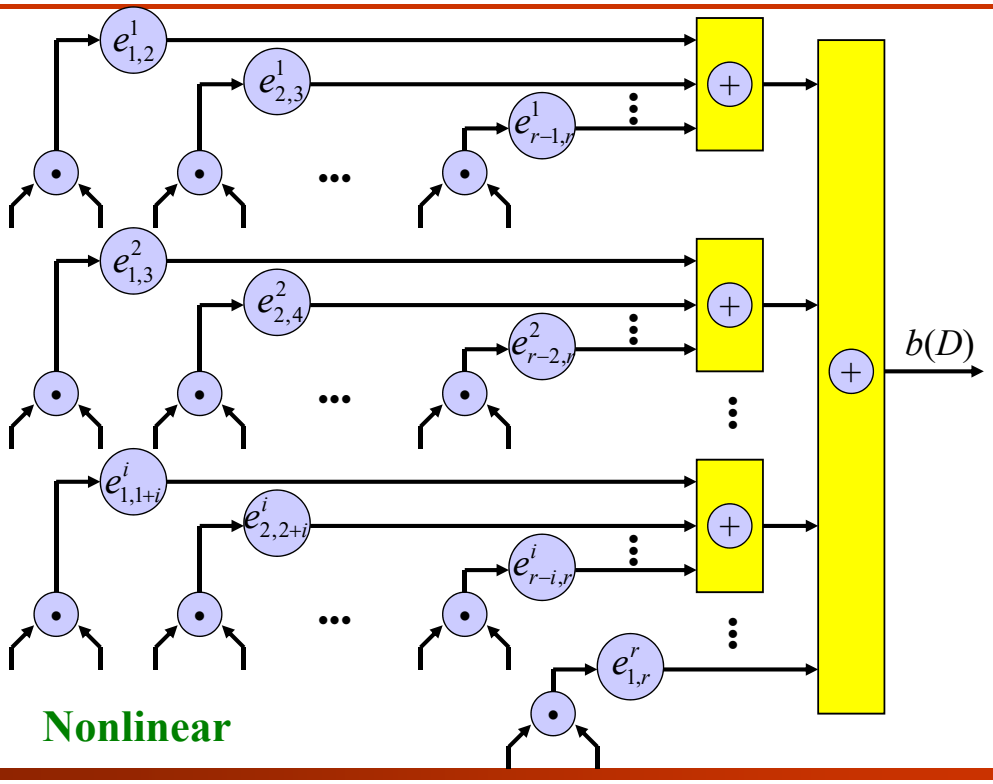
Nonlinear Code Generators (Cont.)

- One type of circuit used to generate high-complexity sequences:
  - Nonlinear **feedforward** logic is added to a conventional linear feedback shift-register generator
  - All binary multipliers have only two inputs
  - $e_{i,j}$ coefficients indicate which shift register stages are connected to the multiplier ($e_{i,j} = 0$: no connections)
  - The final output sequence is the sum of all multipliers' outputs

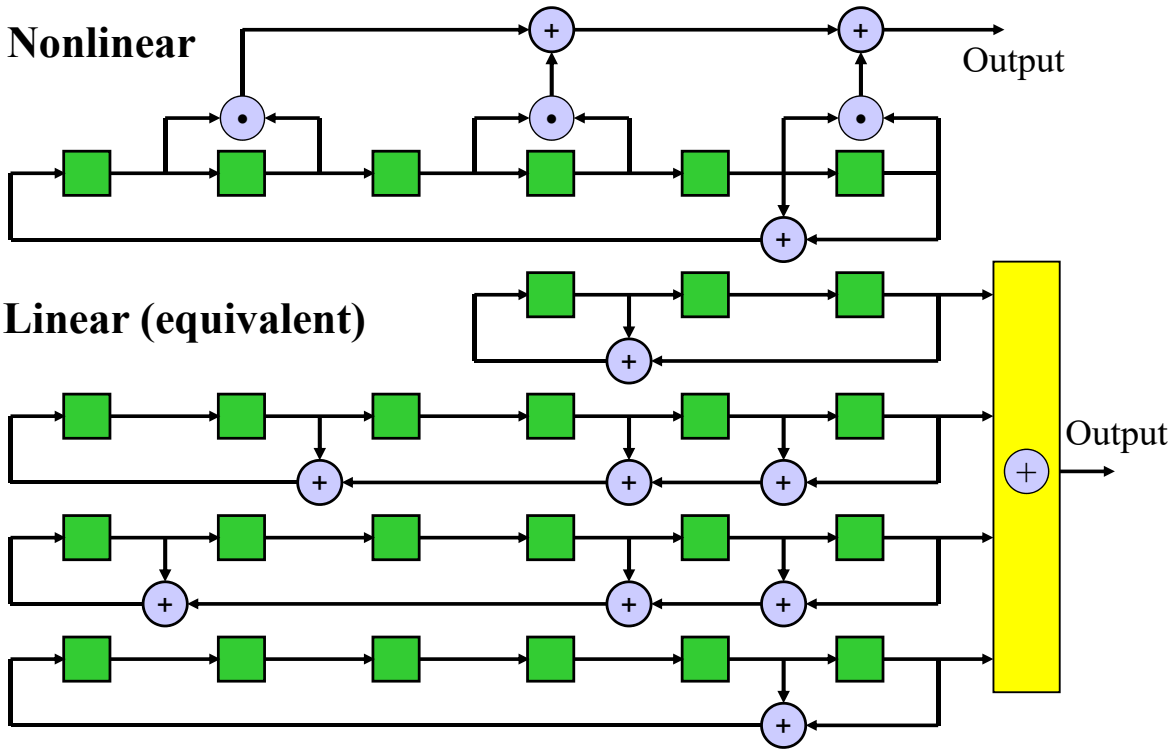


Linear

Nonlinear Code Generators (Cont.)



Nonlinear Code Generators (Cont.)



Some Other Types of Spreading Codes

Kasami Sequences

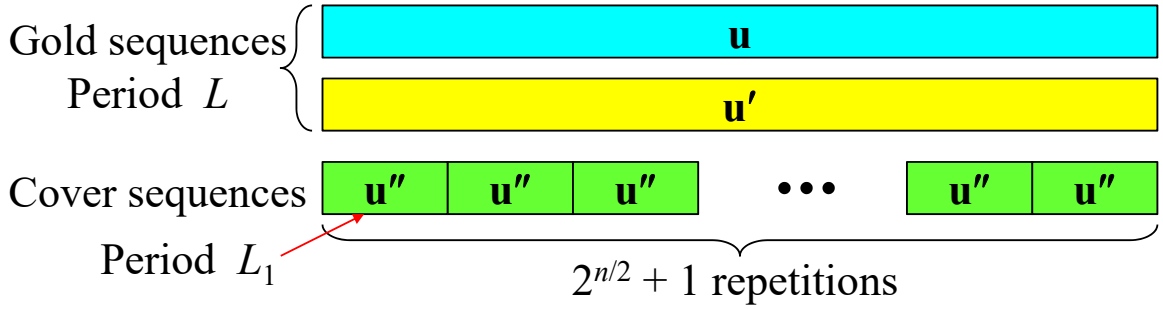
- Kasami code sequences have the advantages, including
  - A very **large code space**
  - **Bounded cross-correlation** between different codes
- Let **u** and **u′** form a preferred pair of binary m-sequence vectors of degree n, where n is an **even** integer
- For $n = 2 \bmod 4$, the code set size is $2^{n/2}(2^n + 1)$
- For $n = 0 \bmod 4$, the code set size is $2^{n/2}(2^n + 1) - 1$
- The length of Kasami sequences is $L = 2^n - 1$

Based on Gold sequences

Based on Gold-like sequences

Kasami Sequences (Cont.)

- Let **u** and **u′** form a preferred pair of binary m-sequence vectors of degree n (with length $L = 2^n - 1$), where
  - **u′** = **u**[$t(n)$]: a **decimation** of **u** with $t(n) \equiv 1 + 2^{(n+2)/2}$
  - **u″** = **u**[$s(n)$]: another **decimation** with $s(n) \equiv 1 + 2^{n/2}$
    - Forms another m-sequence with a period $L_1 = 2^{n/2} - 1$
- The **large set** of Kasami sequences K(**u**): **A shorter period**
  - Composed of **u**, **u′**, and **u″**



Kasami Sequences (Cont.)

- A set of Gold sequences can be obtained:

$$G(\mathbf{u},\mathbf{u}') \equiv \left\{\mathbf{u}, \mathbf{u}', \mathbf{u}\oplus\mathbf{u}', \mathbf{u}\oplus D\mathbf{u}', \mathbf{u}\oplus D^2\mathbf{u}', ..., \mathbf{u}\oplus D^{L-1}\mathbf{u}'\right\}$$

- Define a set of **cover sequences**

$$C(\mathbf{u}'') \equiv \mathbf{0}_L \cup \left[\bigcup_{j=1}^{L_1} D^{j-1}\mathbf{c}\right] = \left\{\mathbf{c}_j,\ j = 0,\cdots,L_1\right\}$$

All-zero sequence **(L_1+1) different sequences**

  - **c** = [c_0, c_1, …, c_{L-1}]: the **repetition** of **u″** by $2^{n/2} + 1$ times
    - where **u″** with a length $L_1 = 2^{n/2} - 1$
- The **large set** of Kasami sequences K(**u**) is composed of **u**, **u′**, and **u″**

$$K(\mathbf{u}) = G(\mathbf{u},\mathbf{u}') \cup \left[\bigcup_{j=1}^{L_1} \left\{D^{j-1}\mathbf{c} \oplus G(\mathbf{u},\mathbf{u}')\right\}\right]$$

Kasami Sequences (Cont.)

- The **large set** of Kasami sequences includes
  - A set of Gold sequences: covered by the cover sequence $\mathbf{0}_L$
  - Other sequences: Gold sequences covered by $D^{j-1}\mathbf{c}$
- The code set size is

$$S_K = 2^{n/2}(2^n + 1)$$

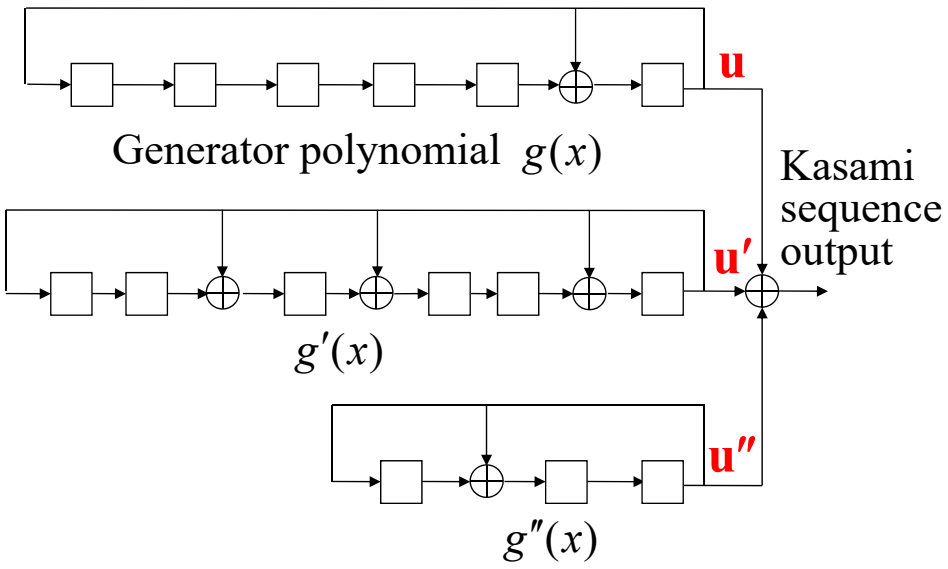
- The **large set** Kasami sequences properties include
  - Polynomial degree: n (even)
  - Sequence length: $L = 2^n - 1$
  - Code set size: $> \approx 2^{1.5n}$
  - Bounded cross-correlations:

$$\{[t(n)-2]/L, [s(n)-2]/L, -1/L, -s(n)/L, -t(n)/L\}$$

Kasami Sequence Generator

- The large set of Kasami sequences generator

$$K(\mathbf{u}) = G(\mathbf{u}, \mathbf{u}') \cup \left[\cup_{j=1}^{L_1} \{ D^{j-1}\mathbf{c} \oplus G(\mathbf{u}, \mathbf{u}') \} \right]$$



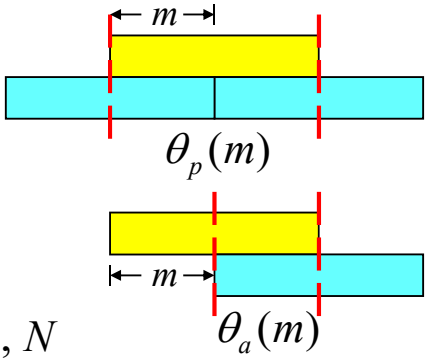
Periodic and Aperiodic Auto-correlation

- For code sequences with complex numbers, the discrete **periodic** auto-correlation function (ACF) of $\mathbf{b} = \{b_0, b_1, \ldots, b_{N-1}\}$ is defined as:

$$\theta_p(m) = \frac{1}{\|\mathbf{b}\|^2} \sum_{n=0}^{N-1} b_n b_{n-m}^*$$

- Define the discrete **aperiodic** auto-correlation function of **b** as

$$\theta_a(m) = \begin{cases} \dfrac{1}{\|\mathbf{b}\|^2} \displaystyle\sum_{n=m}^{N-1} b_n b_{n-m}^*, & m \geq 0 \\ \dfrac{1}{\|\mathbf{b}\|^2} \displaystyle\sum_{n=0}^{N-1+m} b_n b_{n-m}^*, & m < 0 \end{cases}$$



$$\theta_p(m) = \theta_a(m) + \theta_a(m-N), \quad m = 0, 1, \cdots, N$$

Example

- Form some applications, the transmission of a signal may contain only **one period** of the spreading sequence

m	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	$R_a(m)$	$R_p(m)$
0	+	+	+	–	–	+	–	–	+8	+8
1	–	+	+	+	–	–	+	–	+1	0
2	–	–	+	+	+	–	–	+	–2	–4
3	+	–	–	+	+	+	–	–	+1	0
4	–	+	–	–	+	+	+	–	0	0
5	–	–	+	–	–	+	+	+	–1	0
6	+	–	–	+	–	–	+	+	–2	–4
7	+	+	–	–	+	–	–	+	–1	0

Barker Codes

- One requirement of code design is to **minimize** the **maximal sidelobe** of the aperiodic ACF
- For binary sequences, the **rightmost** aperiodic ACF (only **one chip**) has a sidelobe $|\theta_a(N-1)| = 1/N$
  - The maximal sidelobe of a binary sequence is $\theta_{a,\max} \geq 1/N$
- The code sequences attaining this bound are called **Barker codes**

Known Barker codes, including the reversal $(b_0, \ldots, b_{N-1} \Rightarrow b_{N-1}, \ldots, b_0)$ and negation (+ ⇒ –; – ⇒ +)

N	Code
2	+ – ; + +
3	+ + –
4	+ + – + ; + – – –
5	+ + + – +
7	+ + + – – + –
11	+ – + + – + + + – – –
13	+ + + + + – – + + – + – +

Example

- The periodic ACF and aperiodic ACF of the binary Barker code of length 7

m	a_0	a_1	a_2	a_3	a_4	a_5	a_6	$R_a(m)$	$R_p(m)$
0	+	+	+	–	–	+	–	+7	+7
1	–	+	+	+	–	–	+	0	–1
2	+	–	+	+	+	–	–	–1	–1
3	–	+	–	+	+	+	–	0	–1
4	–	–	+	–	+	+	+	–1	–1
5	+	–	–	+	–	+	+	0	–1
6	+	+	–	–	+	–	+	–1	–1

Polyphase Codes

- For some applications, numerous codes with **perfect** periodic ACF are required
  - One approach: use **non-binary** PSK modulation with $M > 2$
- **Zadoff-Chu** (ZC) sequences with two key parameters
  - The length of the sequence N: which is an odd number (and is often a prime number)
  - The root index $q = 1, 2, \ldots, N-1$
- Given these two parameters, **the *q*-th ZC sequence** is defined as

$$S_q[n] = \exp\left[-j\pi qn(n+1)/N\right], \quad n = 0, 1, \cdots, N-1$$

  - Each sequence has length N
  - The **number of sequences** is $N-1$: $S_1, S_2, \ldots, S_q, \ldots, S_{N-1}$

Polyphase Codes (Cont.)

- ZC sequences have several highly desirable properties:
  - **Constant amplitude:** Each element $S_q[n]$ is a unit amplitude complex number.
  - **Zero periodic autocorrelation:** The periodic autocorrelation of a ZC sequence is optimal (i.e., equal to 0 $\Rightarrow$ orthogonal).
  - **Fixed periodic cross-correlation:** When N is a prime number, the normalized periodic cross-correlation of any two distinct ZC sequences of the same length N (e.g., S_{q1} and S_{q2}, for $q1 \neq q2$) is exactly $1/\sqrt{N}$.
  - **DFT or IDFT of a ZC Sequence is a ZC Sequence:** This is particularly useful for OFDM, DFT-Spread OFDM or SC-FDMA waveforms.

Polyphase Codes (Cont.)

- There is another form of ZC sequences defined as (for an arbitrary length N): $a_i = \begin{cases} \exp(j\pi i^2/N), & N \text{ even} \\ \exp(j2\pi i^2/N), & N \text{ odd} \end{cases}$ $\boldsymbol{a_i = a_{i+N}}$
- The size of the phase alphabet **grows linearly** with length, and the **distance** between adjacent phases becomes very **small**
- **Frank codes** exist only for lengths that are squares of integers $N = h^2 = 4, 9, 16, \ldots$. The generation rule is

$$a_i = \exp\left(\frac{j2\pi i}{h}\left\lfloor \frac{i}{h} \right\rfloor\right) = \exp(\phi_i), \quad \lfloor x \rfloor : \text{round-toward-zero (floor)}$$

- $N = 4 \Rightarrow h = 2$, $\boldsymbol{a} = \{+1\ +1\ +1\ -1\}$, $\phi_i = 0, 0, 0, \pi$, $i = 0, \ldots, 3$
- $N = 16 \Rightarrow h = 4$, $\boldsymbol{a} = \{+1\ +1\ +1\ +1\ +1\ +j\ -1\ -j\ +1\ -1\ +1\ -1\ +1\ -j\ -1\ +j\}$, $\phi_i = 0, 0, 0, 0, 0, \pi/2, \pi, 3\pi/2, 0, \pi, 0, \pi, 0, 3\pi/2, \pi, \pi/2$

Ternary Sequences

- **Ternary sequences**: the elements a_i may be the **zero** value in addition to binary values ± 1
  - The alphabet is now ternary **{–1, 0, +1}**
- For $N = 13$, a ternary sequence is
  - **{+1 0 0 +1 0 +1 +1 +1 –1 –1 0 +1 –1}**
- +1 0 0 +1 0 +1 +1 +1 –1 –1 0 +1 –1
- +1 0 0 +1 0 +1 +1 +1 –1 –1 0 +1 –1
  $\Rightarrow \theta_p(m) = +9$ for $m = 0 \bmod N$
- +1 0 0 +1 0 +1 +1 +1 –1 –1 0 +1 –1
- –1 +1 0 0 +1 0 +1 +1 +1 –1 –1 0 +1 $\Rightarrow \theta_p(1) = 0$
- +1 0 0 +1 0 +1 +1 +1 –1 –1 0 +1 –1
- +1 –1 +1 0 0 +1 0 +1 +1 +1 –1 –1 0 $\Rightarrow \theta_p(2) = 0$

Homework

- 3-1
- 3-2 1 + D^2 + D^5
- 3-6
- 3-7
- 3-9
- 3-12
- 3-16
- 3-17
- 3-20
- 3-24
- 3-26