Lecture 25

EE 421 / CS 425

Digital System Design

Fall 2025

Shahid Masud

Topics

QUIZ 5 TODAY

- Data **Scramblers**
- Data **Un-Scramblers**
- Scrambler Applications
- Scrambler Examples
- Binary Rate Multipliers – Maybe next lecture

Rescheduled Lectures

- Rescheduled lecture: Academic Block SS Auditorium has been reserved for Make-up Class of EE 421 from

10:00 AM to 11:30 AM on Saturday, 6th December 2025.

- As per RO's below email, Academic Block-SS Auditorium has been reserved for reserved class of EE 421L/CS 425L on

Monday, 8th December 2025 from 2:00 PM to 4:50 PM.

- **Details will be announced**
- **Quiz 6 in Last lecture**

What is a Scrambler?

- Scramblers → circuits that pseudo-randomly change the values of some bits in a data stream with the purpose of 'whitening' its spectrum to dilute strong spectral components.
- This will reduce electromagnetic interference and introduce data security as part of an encryption algorithm.
- In natural signals, e.g. voice, image, transducer, energy is concentrated in some bands. Scrambling → spread the energy to reduce interference and spikes.
- An ALFSR forms backbone of scrambling operation.
- There are two main types:
  - Additive Scrambler
  - Multiplicative Scrambler

Example Applications of LFSR

Ref: from Wikipedia page

Digital broadcasting systems that use linear-feedback registers:

- ATSC Standards (digital TV transmission system – North America)
- DAB (Digital Audio Broadcasting system – for radio)
- DVB-T (digital TV transmission system – Europe, Australia, parts of Asia)
- NICAM (digital audio system for television)

Other digital communications systems using LFSRs:

- INTELSAT business service (IBS)
- Intermediate data rate (IDR)
- HDMI 2.0
- SDI (Serial Digital Interface transmission)
- Data transfer over PSTN (according to the ITU-T V-series recommendations)
- CDMA (Code Division Multiple Access) cellular telephony
- 100BASE-T2 "fast" Ethernet scrambles bits using an LFSR
- 1000BASE-T Ethernet, the most common form of Gigabit Ethernet, scrambles bits using an LFSR
- PCI Express
- SATA[13]
- Serial attached SCSI (SAS/SPL)
- USB 3.0
- IEEE 802.11a scrambles bits using an LFSR
- Bluetooth Low Energy Link Layer is making use of LFSR (referred to as whitening)
- Satellite navigation systems such as GPS and GLONASS. All current systems use LFSR outputs to generate some or all of their ranging codes (as the chipping code for CDMA or DSSS) or to modulate the carrier without data (like GPS L2 CL ranging code). GLONASS also uses frequency-division multiple access combined with DSSS.

Other uses [edit]

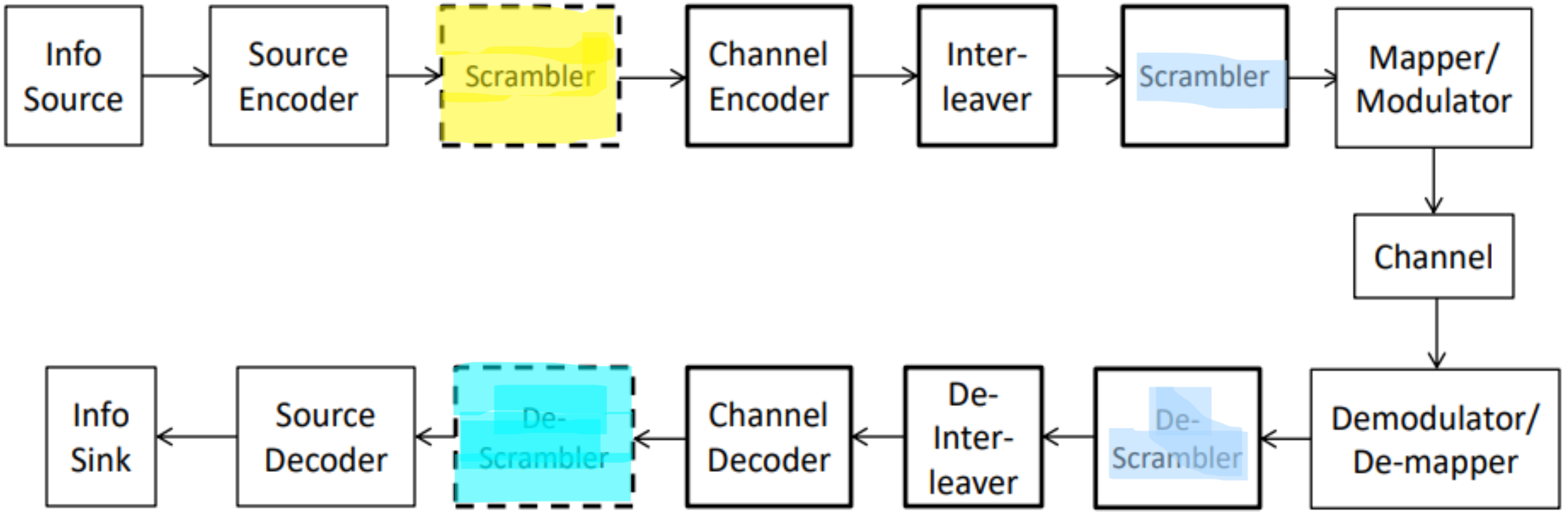
- LFSRs are also used in radio jamming systems to generate pseudo-random noise to raise the noise floor of a target communication system.

Types of Scramblers

Standard	Scrambler type
V.34 (33 Kbps Modem)	Multiplicative
56 Kbps Modem	Multiplicative
DVB	Additive
LTE	Additive
IEEE 802.11 a	Additive
IEEE 802.16 a	Multiplicative

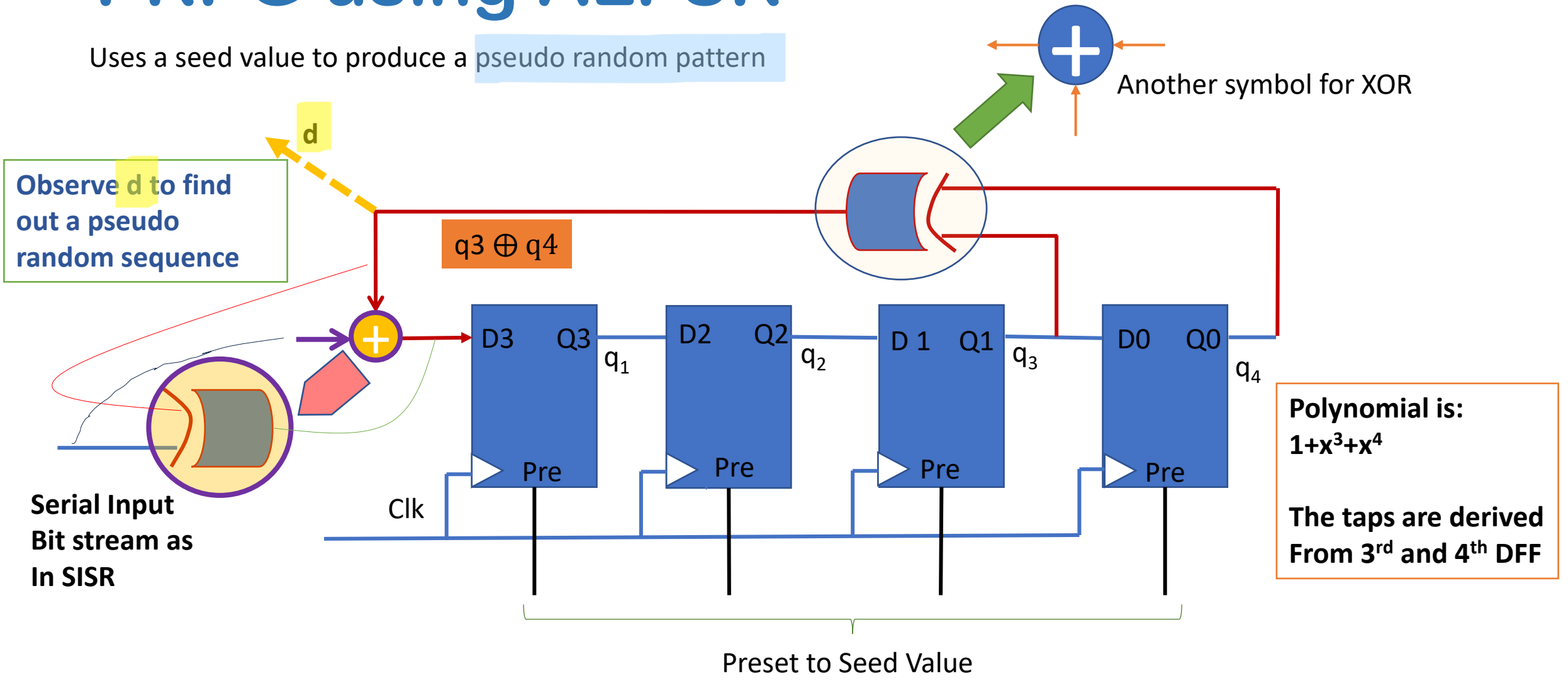
$1+x^4+X^7$ is polynomial for 802.11

Typical Communication System



PRPG using ALFSR

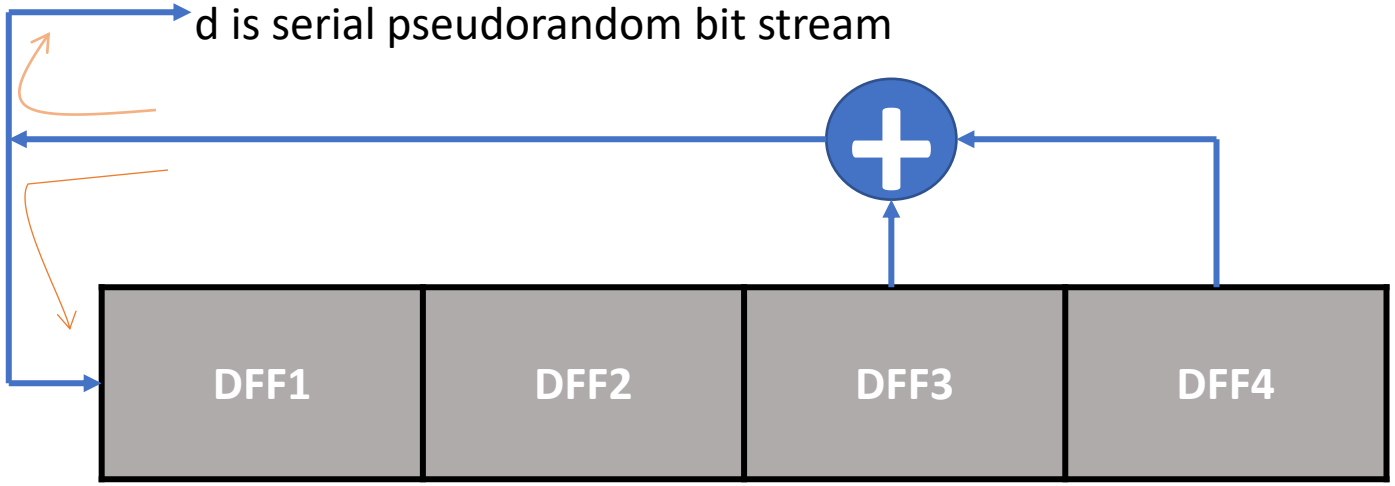
Uses a seed value to produce a pseudo random pattern



Polynomial is:
$1+x^3+x^4$

The taps are derived From 3rd and 4th DFF

Simplified Representation



Preset

q1	q2	q3	q4	d
1	1	1	1	0
0	1	1	1	0
0	0	1	1	0
0	0	0	1	1
1	0	0	0	0
0	1	0	0	0
0	0	1	0	1
1	0	0	1	1
1	1	0	0	0
0	1	1	0	1
1	0	1	1	0
0	1	0	1	1
1	0	1	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Characteristic Polynomial
$1+x^3+x^4$

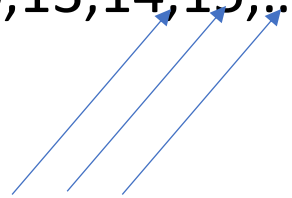
The sequence observed at q3q2q1q0 = {15,7,3,1,8,4,2,9,12,6,11,5,10,13,14,15,.....}

Changing seed value with same polynomial

Question:
What happens when seed value is changed from ‘1111’ to ‘1000’?

Answer:
With seed value of ‘1111’
The sequence observed at q3q2q1q0 = {15,7,3,1,8,4,2,9,12,6,11,5,10,13,14,15,….}

With seed value of ‘1000’
The sequence observed at q3q2q1q0 = {8,4,2,9,12,6,11,5,10,13,14,15,7,3,1,8,….}



Both sequences are equal in a circle, just the starting point is different

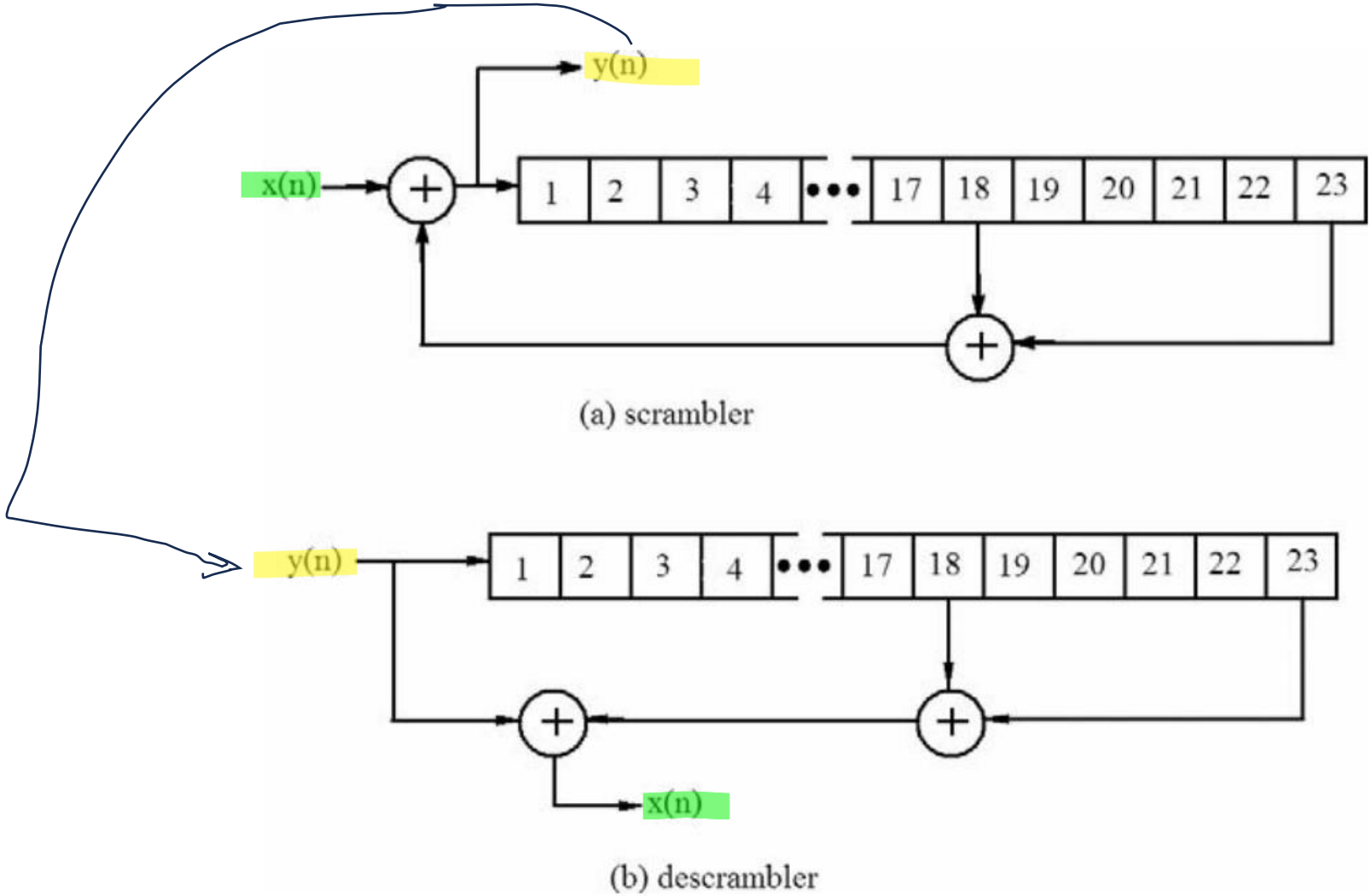
PRPG Cycle length vs Register length

# of Bits	Length of Loop		Taps
2	3	*	[0,1]
3	7	*	[0,2]
4	15		[0,3]
5	31	*	[1,4]
6	63		[0,5]
7	127	*	[0,6]
8	255		[1,2,3,7]
9	511		[3,8]
10	1,023		[2,9]
11	2,047		[1,10]
12	4,095		[0,3,5,11]
13	8,191	*	[0,2,3,12]
14	16,383		[0,2,4,13]
15	32,767		[0,14]
16	65,535		[1,2,4,15]
17	131,071	*	[2,16]
18	262,143		[6,17]
19	524,287	*	[0,1,4,18]
20	1,048,575		[2,19]
21	2,097,151		[1,20]
22	4,194,303		[0,21]
23	8,388,607		[4,22]
24	16,777,215		[0,2,3,23]
25	33,554,431		[2,24]
26	67,108,863		[0,1,5,25]
27	134,217,727		[0,1,4,26]
28	268,435,455		[2,27]
29	536,870,911		[1,28]
30	1,073,741,823		[0,3,5,29]
31	2,147,483,647	*	[2,30]
32	4,294,967,295		[1,5,6,31]

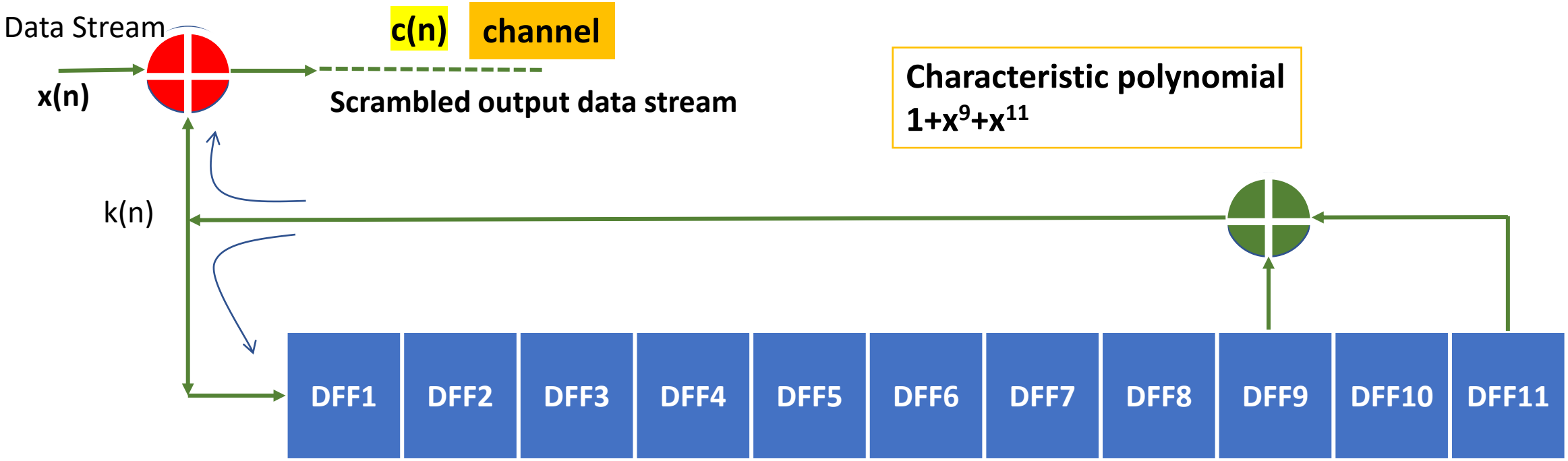
Additive Scrambler and Descrambler

- **Synchronous scramblers** because they require the initial state of the scrambler and descrambler to be the same
- They are **non-recursive** because they do not have any feedback loop in the process
- LFSR is connected to the data stream by means of an additional modulo-2 adder (XOR) gate
- Used in 100Base-TX interface which repeats its sequence after 2^N-1 = 2047 bits; with N=11
- Usually employed where a fixed size frame is to be transmitted

Additive Scrambler Eg. from Web

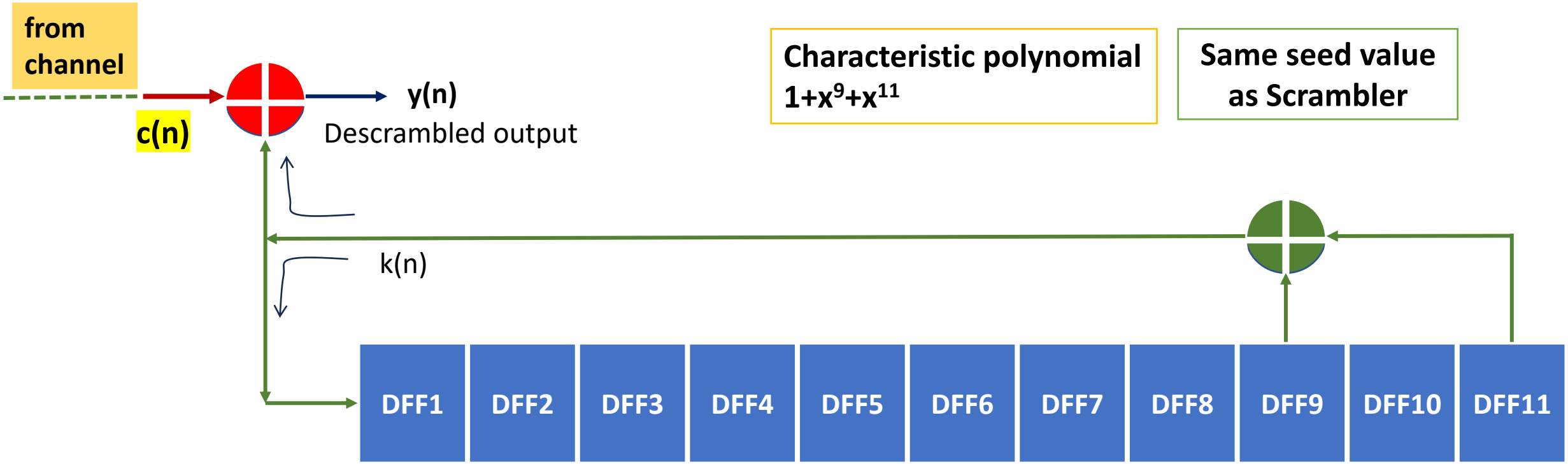


Additive Scrambler Example



x(n) represents the data to be scrambled at time n
k(n) represents the 'key' produced by the LFSR
c(n) represents the scrambled code word

Corresponding Additive Descrambler



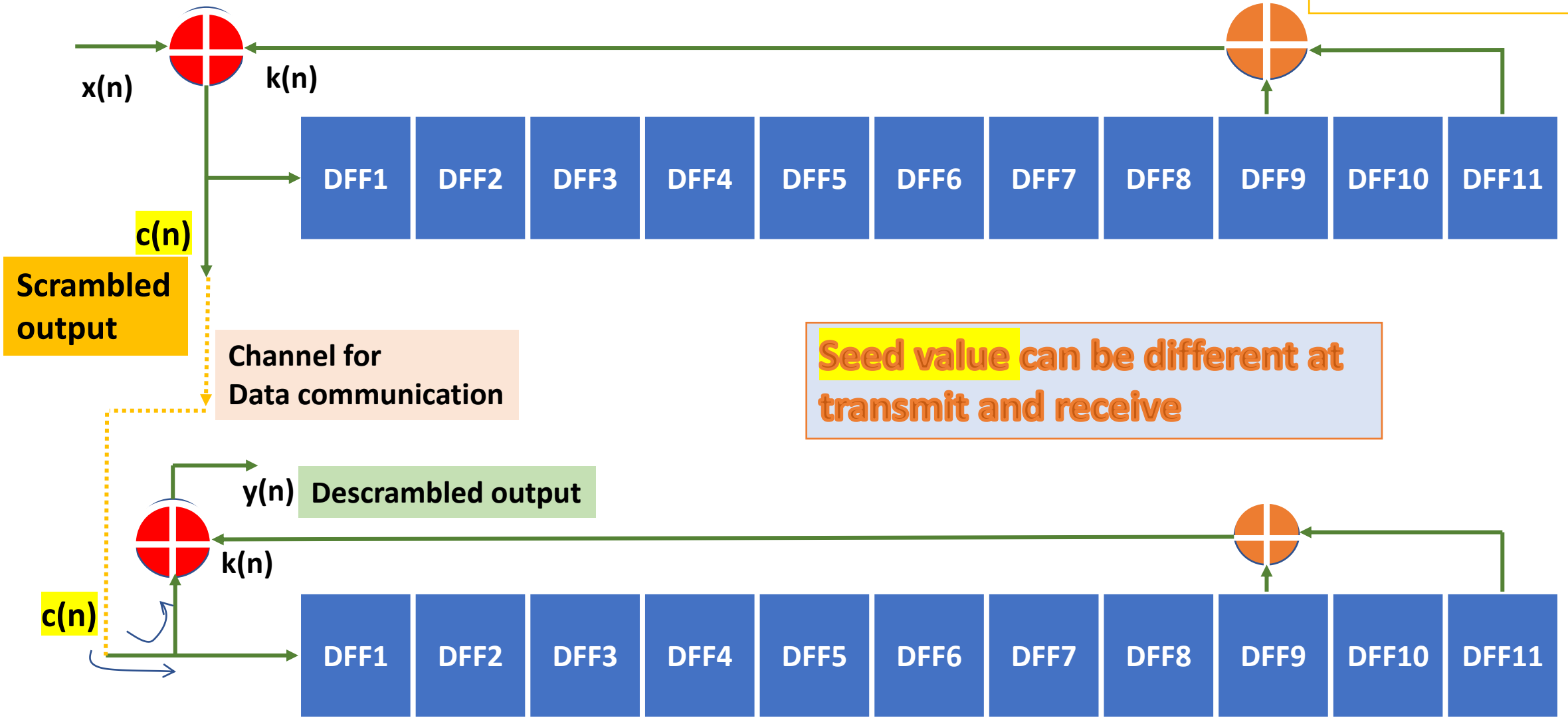
y(n) represents the data de-scrambled at time n
k(n) represents the 'key' produced by the LFSR
c(n) represents the scrambled code word

Multiplicative Scramblers

- They are **Asynchronous** because they do not require LFSR synchronization.
- They are **recursive** as they have feedback loops.
- The pair of scrambler-descrambler is **self-synchronizing**. They do not need to start from same initial value.
- The self-synchronizing process could take up to N bits (or N clock cycles) so the first N values of y(n) should be discarded.
- With this approach, the transmission errors are multiplied by T+1, where T is the number of taps used in LFSR. Thus if one bit is flipped due to some error, the descrambler will result in 3 bit error.

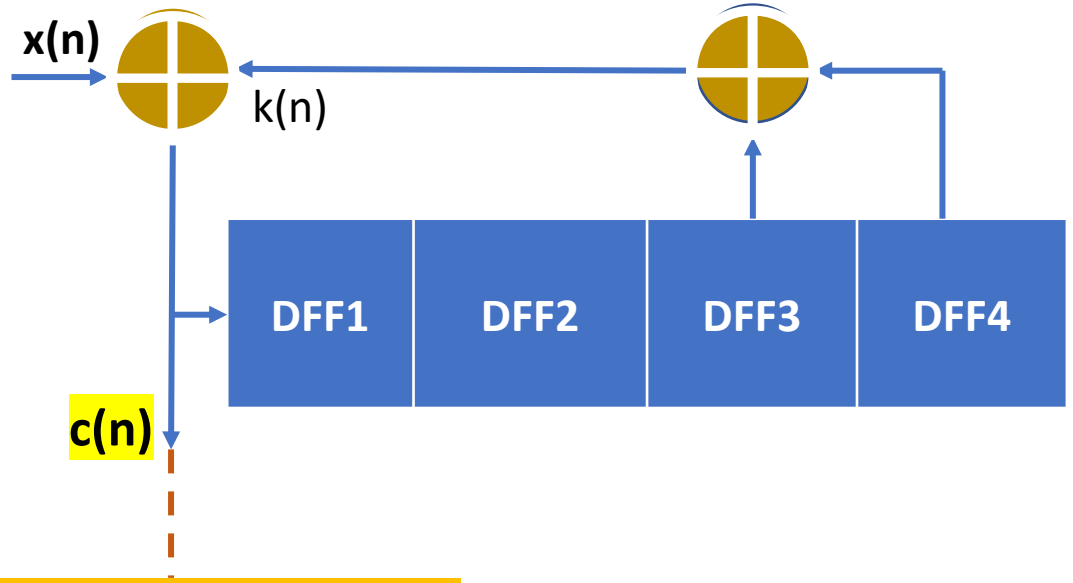
Multiplicative Scrambler - Descrambler

Characteristic polynomial
$1+x^9+x^{11}$



Seed value can be different at transmit and receive

Example – Multiplicative Scrambler

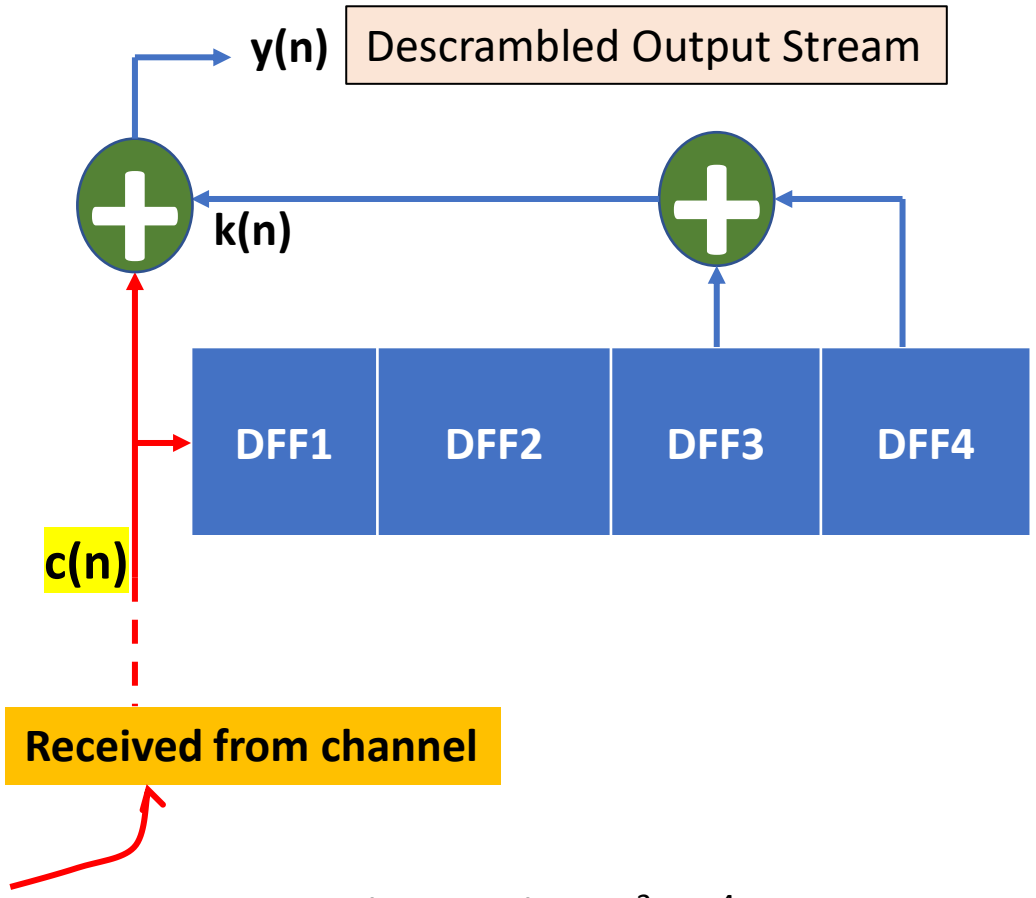


Transmitted Scrambled Stream to Receiver

LFSR Using polynomial $1 + x^3 + x^4$
Scrambler Seed Value "0000"
Input Data stream:
"101100010110"

Registers output "q1q2q3q4"				k	x	c
0	0	0	0	0	1	1
1	0	0	0	0	0	0
0	1	0	0	0	1	1
1	0	1	0	1	1	0
0	1	0	1	1	0	1
1	0	1	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	1	0
0	1	1	1	0	0	0
0	0	1	1	0	1	1
1	0	0	1	1	1	0
0	1	0	0	0	0	0

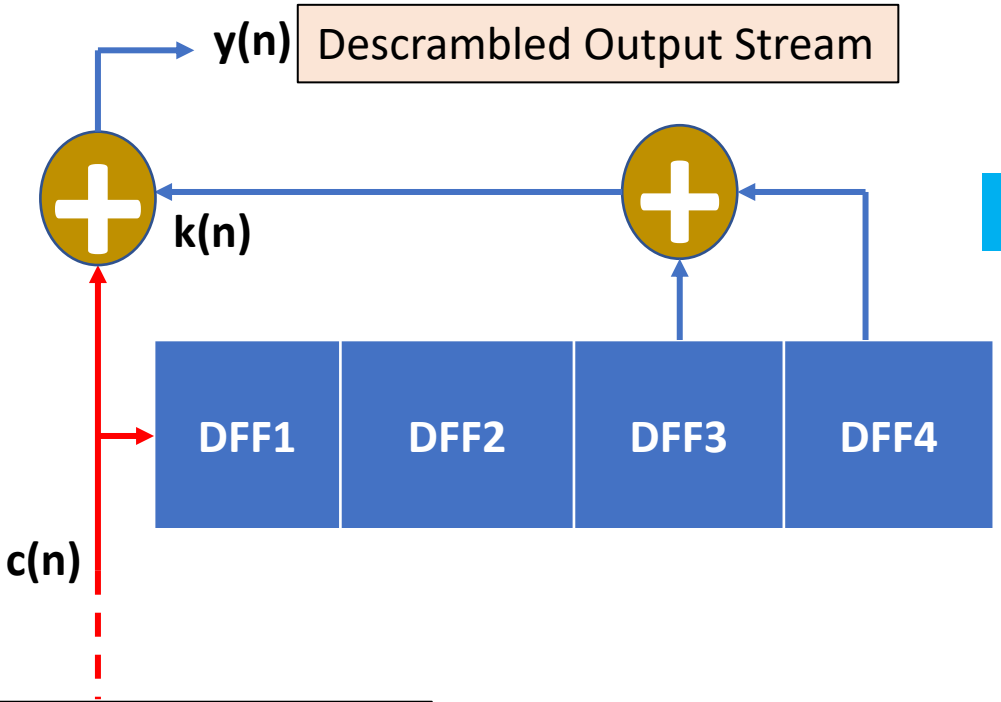
Continue to – Multiplicative De-Scrambler



LFSR Using polynomial $1 + x^3 + x^4$
Scrambler Seed Value "0000"
De-Scrambler Seed Value "1111"
Input Data stream:
"101100010110"

	Registers output "q1q2q3q4"				k	c	y	
Seed	1	1	1	1	0	1	1	N=4
	1	1	1	1	0	0	0	
	0	1	1	1	0	1	1	
	1	0	1	1	0	0	0	
	0	1	0	1	1	1	0	y=x Input String From here
	1	0	1	0	1	1	0	
	1	1	0	1	1	1	0	
	1	1	1	0	1	0	1	
	0	1	1	1	0	0	0	
	0	0	1	1	0	1	1	
	1	0	0	1	1	0	1	
	0	1	0	0	0	0	0	

Multiplicative De-Scrambler with Error Detection



LFSR Using polynomial $1 + x^3 + x^4$
Scrambler Seed Value "0000"
De-Scrambler Seed Value "1111"
Input Data stream:
"101100010110"

T is the number of TAPS used in ALFSR

Seed

Registers output "q1q2q3q4"				k	c	y
1	1	1	1	0	1	1
1	1	1	1	0	0	0
0	1	1	1	0	1	1
1	0	1	1	0	0	0
0	1	0	1	1	1	0
1	0	1	0	1	0	1
1	1	0	1	1	1	0
1	1	1	0	1	0	1
0	1	1	1	0	0	1
0	0	1	1	0	1	0
1	0	0	1	1	0	1
0	1	0	0	0	0	0

N=4

y=x Input String From here

One error in blue was introduced in c(n)
This error resulted in **T+1 = 3** errors in y(n) shown in red

Case Study: Binary Rate Multipliers

Binary Rate Multipliers - BRM

Definition: A circuit module that transforms a stream of input clock pulses into another stream of output clock pulses

Let Ni = no. of input pulses for a particular time period

Let No = no. of output clock pulses for the same time period

$$\textbf{\textit{Binary Fractional Rate Multiplier}} \implies No = \frac{B}{2^n} Ni$$

B = Rate Constant Input to module

B = $(B_{n-1}, B_{n-2}, \ldots\ldots, B_2, B_1, B_0)_2$

n = no. of binary counter stages controlling the module

The clock input drives an n-bit binary counter whose outputs are labelled (Xn,Xn-1,….,X2,X1).

Will do next lecture due to quiz today