# Digital Communication Systems
# ECS 452

**Asst. Prof. Dr. Prapun Suksompong**

prapun@siit.tu.ac.th

**5.2 Binary Convolutional Codes**

## Binary Convolutional Codes

- Introduced by Elias in 1955
  - There, it is referred to as convolutional parity-check symbols codes.
  - Peter Elias received
    - Claude E. Shannon Award in 1977
    - IEEE Richard W. Hamming Medal in 2002
      - for "fundamental and pioneering contributions to information theory and its applications
- The encoder **has memory**.
  - In other words, the encoder is a **sequential circuit** or a **finite-state machine**.
    - Easily implemented by shift register(s).
    - The **state** of the encoder is defined as the contents of its memory.

## Binary Convolutional Codes

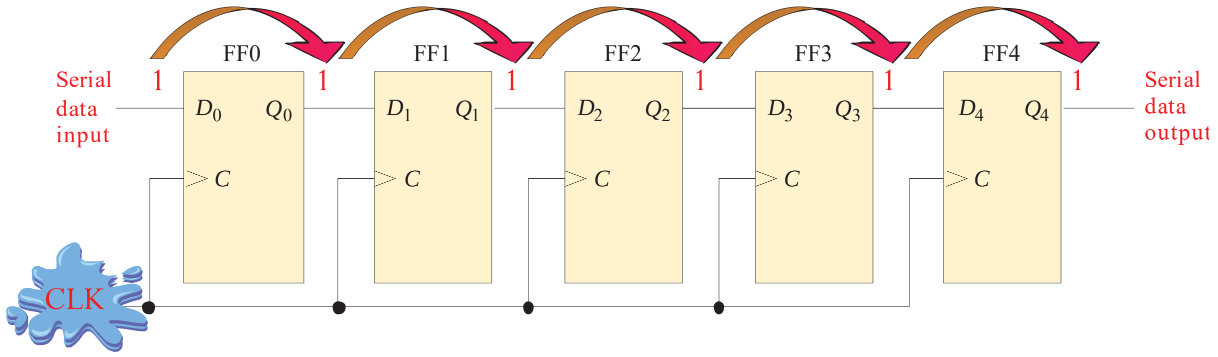
- The encoding is done on a **continuous** running basis rather than by blocks of $k$ data digits.
  - So, we use the terms **bit streams** or **sequences** for the input and output of the encoder.
  - In theory, these sequences have infinite duration.
  - In practice, the state of the convolutional code is periodically forced to a known state and therefore code sequences are produced in a block-wise manner.

## Binary Convolutional Codes

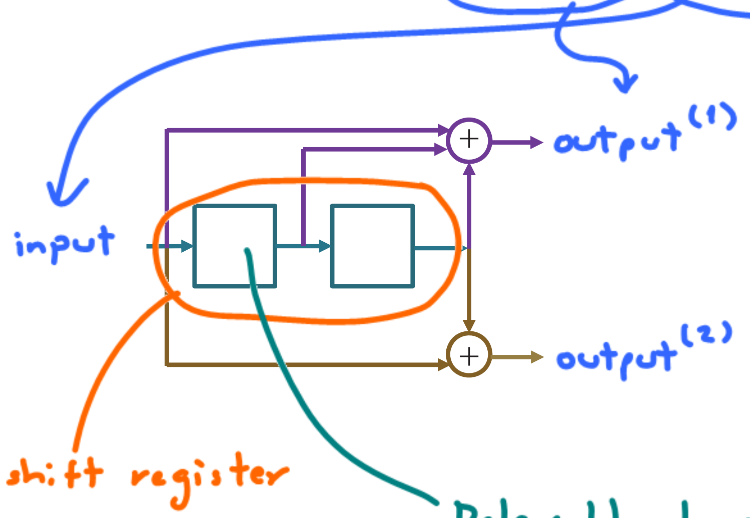
- In general, a **rate-$\frac{k}{n}$ convolutional encoder** has
  - $k$ shift registers, one per input information bit, and
  - $n$ output coded bits that are given by linear combinations (over the binary field, of the contents of the registers and the input information bits.
- $k$ and $n$ are usually small.
- For simplicity of exposition, and for practical purposes, only **rate-$\frac{1}{n}$** binary convolutional codes are considered here.
  - $k = 1$.
  - These are the most widely used binary codes.

## (Serial-in/Serial-out) Shift Register

- Accept data serially: one bit at a time on a single line.
- Each clock pulse will move an input bit to the next FF. For example, a 1 is shown as it moves across.
- Example: five-bit serial-in serial-out register.



## Example 1: $n = 2$, $k = 1$



D-transform

Delay block : D, $z^{-1}$

by one (discrete) time unit

from z-transform

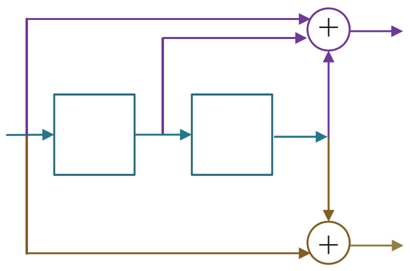
## Graphical Representations

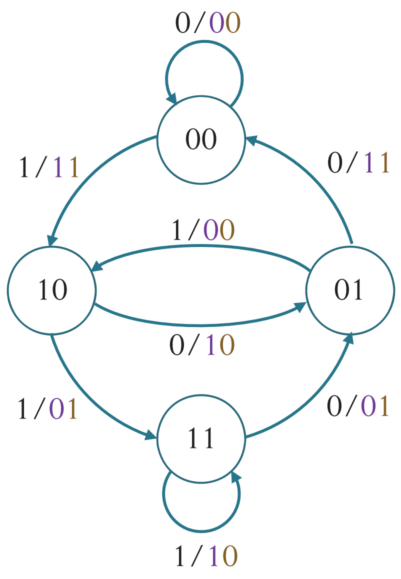
- Three different but related graphical representations have been devised for the study of convolutional encoding:

1. the state diagram
2. the code tree
3. the trellis diagram

## Ex. 1: State (Transition) Diagram

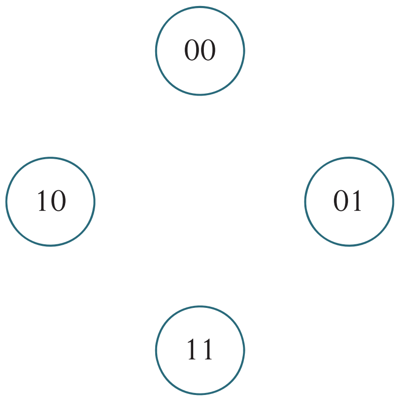
- The encoder behavior can be seen from the perspective of a finite state machine with its state (transition) diagram.





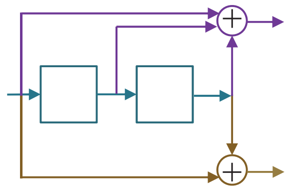
A four-state directed graph that uniquely represents the input-output relation of the encoder.

## Drawing State Diagram

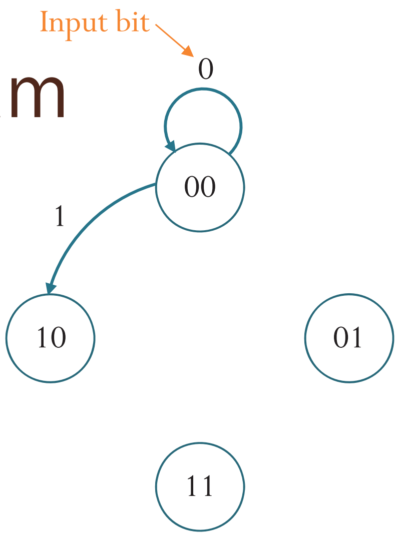


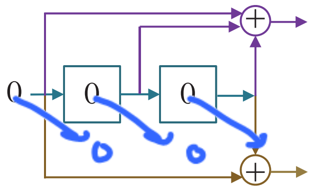
There are two boxes (FFs).
So, there are two bits in the memory.

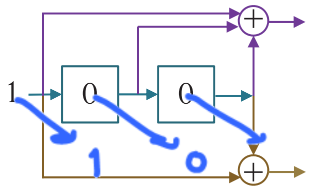
$\underbrace{\text{two bits}}$ ↳ state

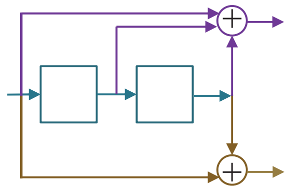


## Drawing State Diagram

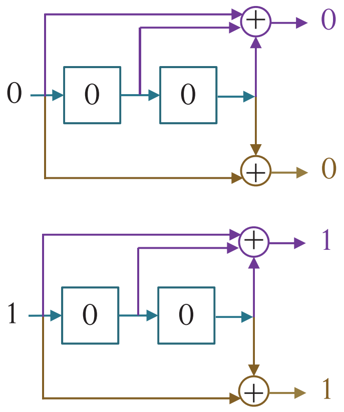


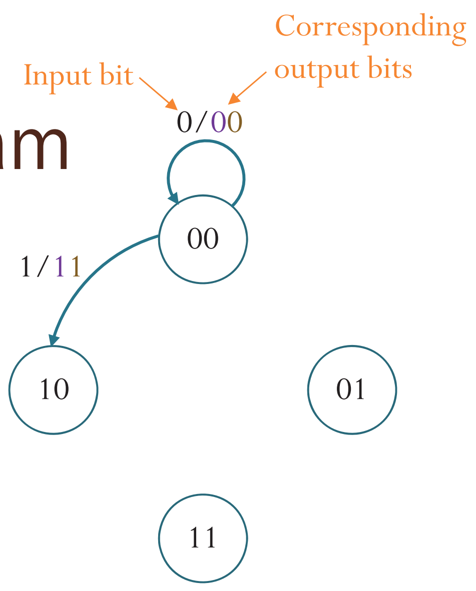


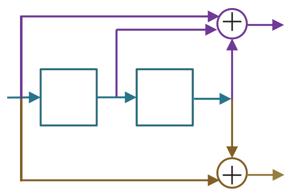




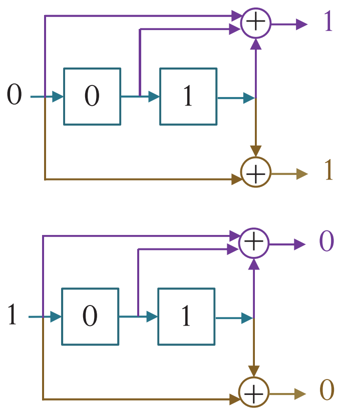
## Drawing State Diagram

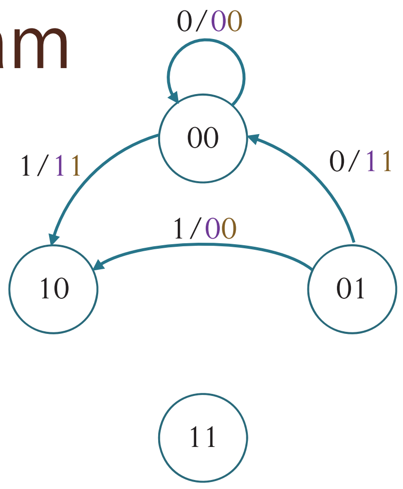


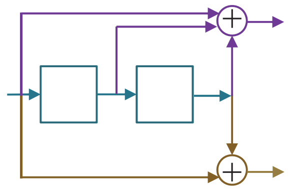




## Drawing State Diagram

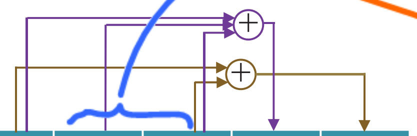






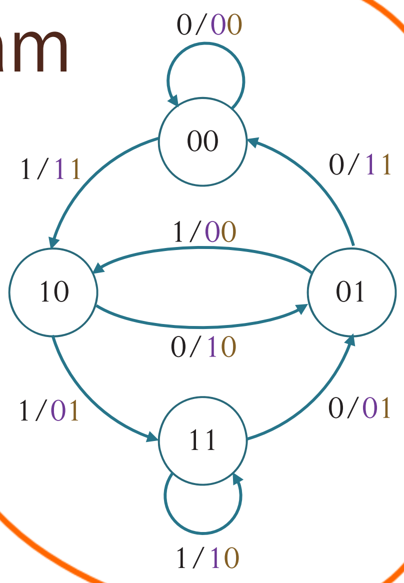
## Drawing State Diagram

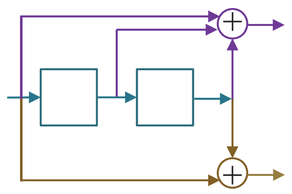
current state



| $b$ | $s_1$ | $s_2$ | $x^{(1)}$ | $x^{(2)}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

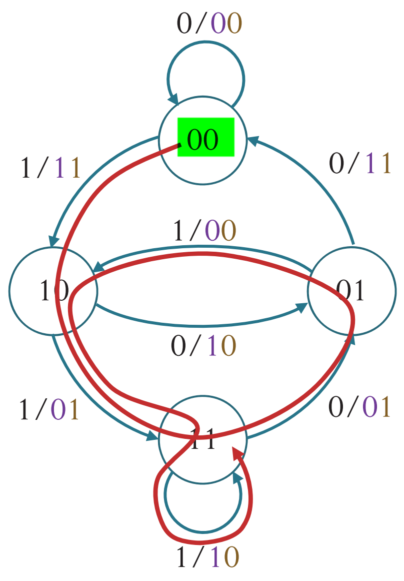
$\underbrace{\quad}$ ↑ next state

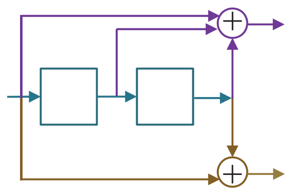




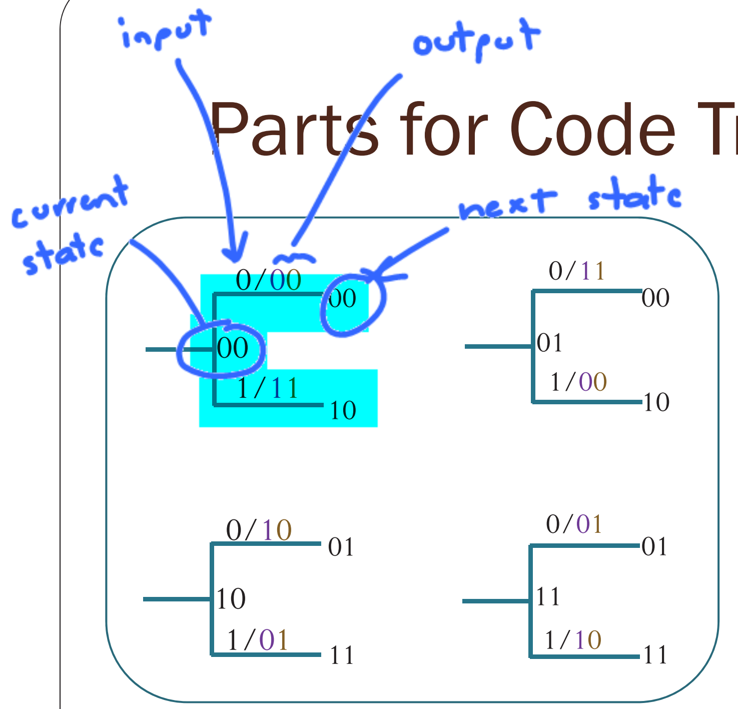
## Tracing the State Diagram to Find the Outputs

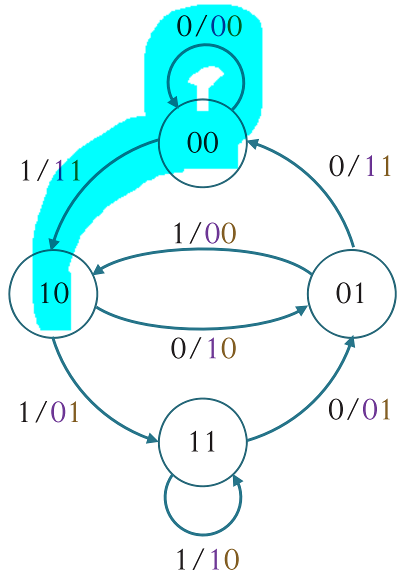
| | | | | | | |
|---|---|---|---|---|---|---|
| $b$ = Input = | 1 | 1 | 0 | 1 | 1 | 1 |
| $x$ = Output | 11 | 01 | 01 | 00 | 01 | 10 |



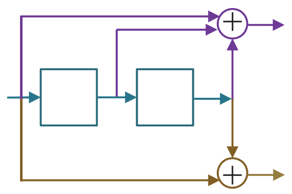


## Parts for Code Tree



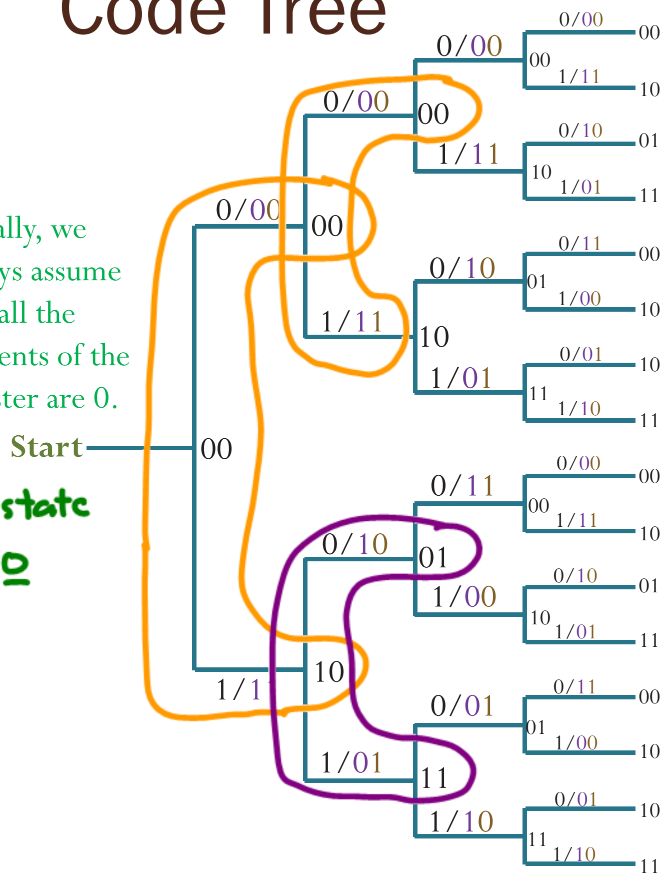


Two branches initiate from each node, the upper one for 0 and the lower one for 1.



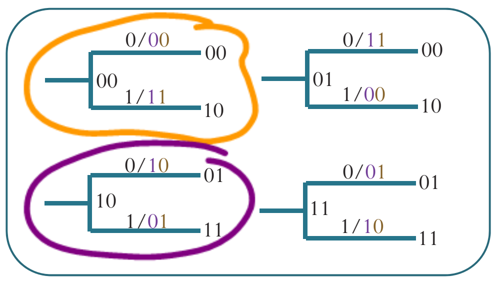
Show the coded output for any possible sequence of data digits.

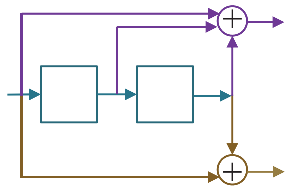
## Code Tree



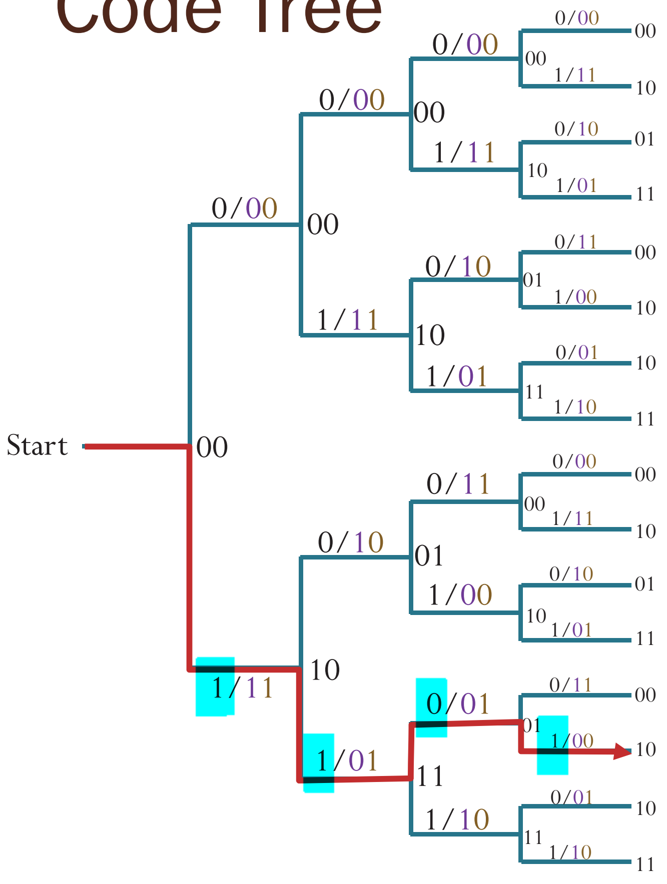
Initially, we always assume that all the contents of the register are 0.

Initial state = 0

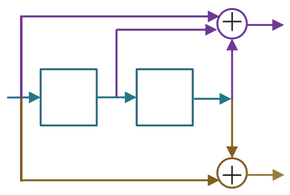




## Code Tree

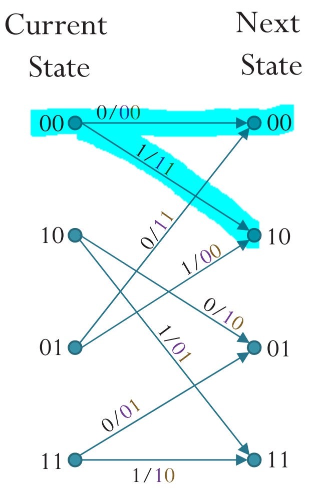


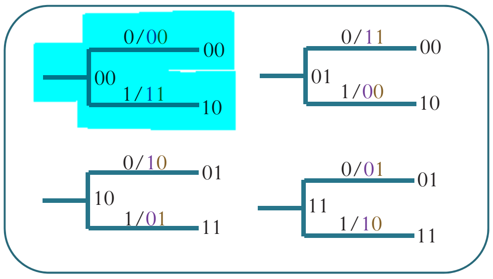
| Input | 1 | 1 | 0 | 1 |
|---|---|---|---|---|
| Output | 11 | 01 | 01 | 00 |

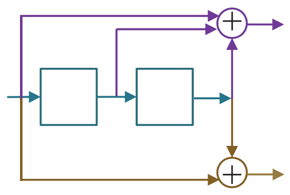


## Code Trellis

[Carlson & Crilly, 2009, p. 620]

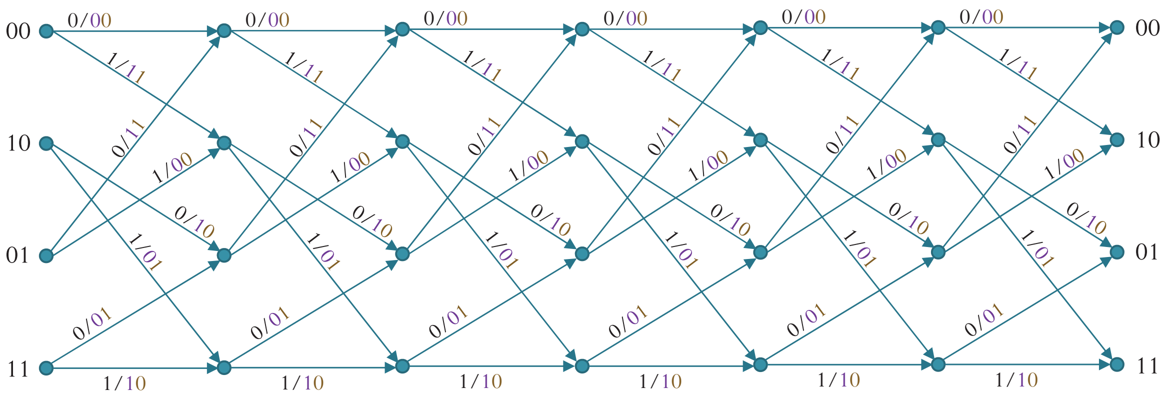


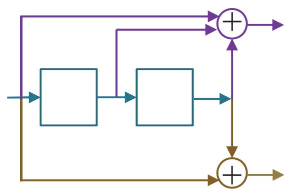




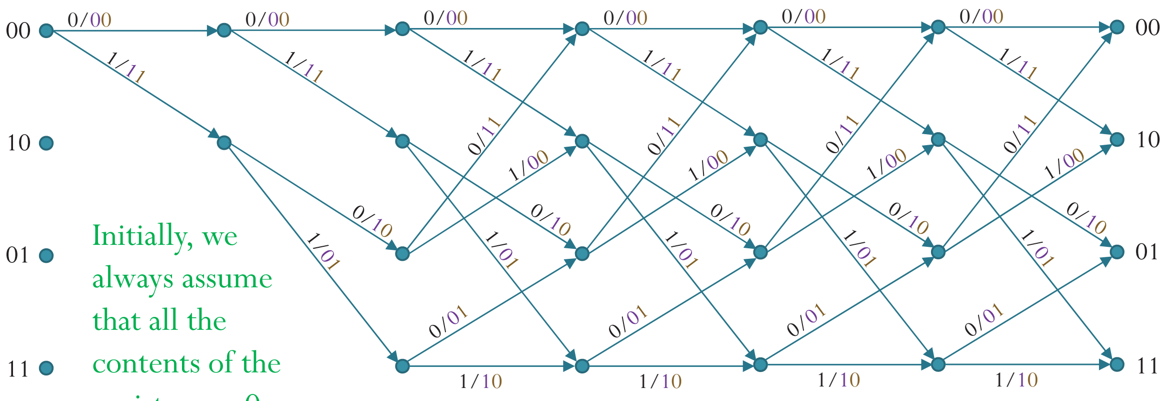
## Towards the Trellis Diagram

Another useful way of representing the code tree.



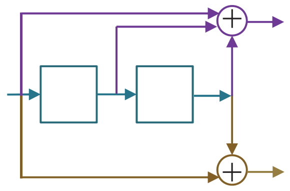


## Trellis Diagram

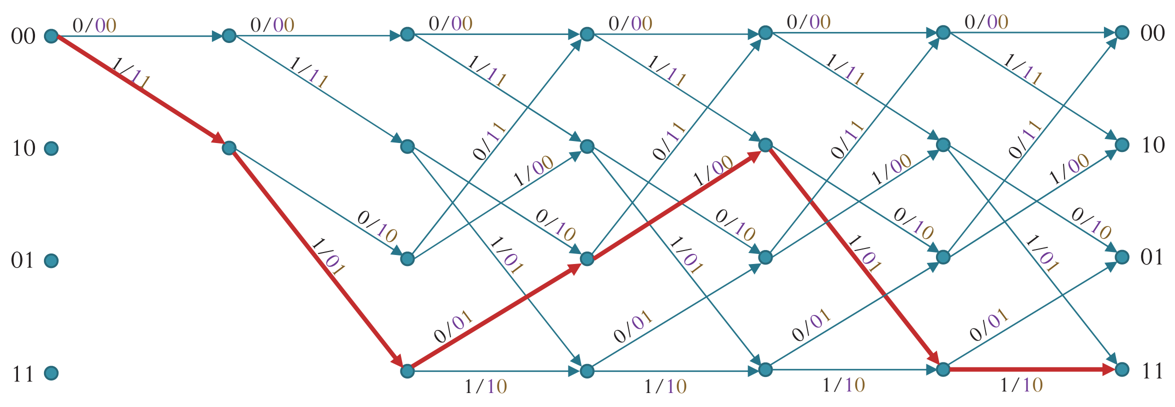


Initially, we always assume that all the contents of the register are 0.

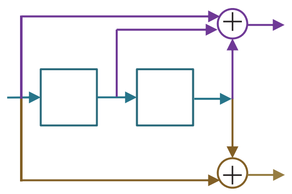
Each path that traverses through the trellis represents a valid codeword.



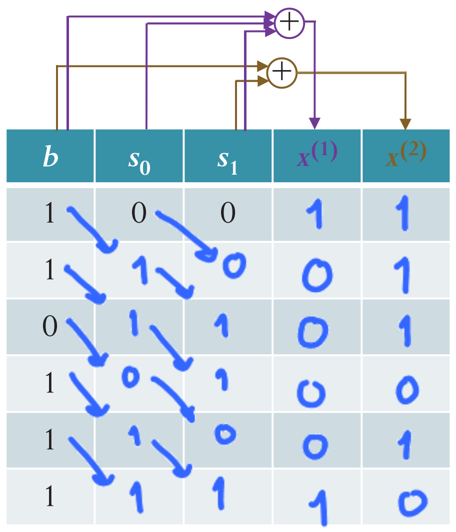
## Trellis Diagram



| b = Input = | [1 | 1 | 0 | 1 | 1 | 1] |
|---|---|---|---|---|---|---|
| x = Output = | [11 | 01 | 01 | 00 | 01 | 10] |

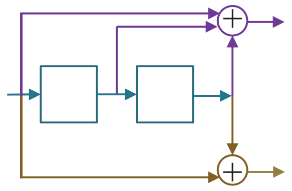


## Directly Finding the Output

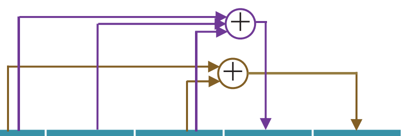


| $b$ | $s_0$ | $s_1$ | $x^{(1)}$ | $x^{(2)}$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| Input | 1 | 1 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|
| Output | 11 | 01 | 01 | 00 | 01 | 10 |



# Directly Finding the Output



| $b$ | $s_0$ | $s_1$ | $x^{(1)}$ | $x^{(2)}$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| Input | 1 | 1 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|
| Output | 11 | 01 | 01 | 00 | 01 | 10 |