Lecture 1: Convolutional Codes

- Block codes and convolutional codes are two major class of codes for error correction.
- From a viewpoint, convolutional codes differ form block codes in that the encoder contains memory.
- For convolutional codes, the encoder outputs at any given time unit depend not only on the inputs at that time unit but also on some number of previous inputs:

$$v_t = f(u_{t-m}, \cdots, u_{t-1}, u_t),$$

where $v_t \in F_2^n$ and $u_t \in F_2^k$.

- A rate $R = \frac{k}{n}$ convolutional encoder with memory order m can be realized as a k-input, n-output linear sequential circuit with input memory m.

- Convolutional codes were first introduced by Elias in 1955.
- The information and codewords of convolutional codes are of infinite length, and therefore they are mostly referred to as information and code sequence.
- In practice, we have to truncate the convolutional codes by zero-biting, tailbiting, or puncturing.
- There are several methods to describe a convolutional codes.
  1. Sequential circuit: shift register representation.
  2. MIMO LTI system: impulse response encoder
  3. Algebraic description: scalar G matrix in time domain
  4. Algebraic description: polynomial G matrix in Z domain,
  5. Combinatorial description: state diagram and trellis

- We emphasize differences among the terms: code, generator matrix, and encoder.
    1. Code: the set of all code sequences that can be created with a linear mapping.
    2. Generator matrix: a rule for mapping information to code sequences.
    3. Encoder: the realization of a generator matrix as a digital LTI system.
- For example, one convolutional code can be generated by several different generator matrices and each generator matrix can be realized by different encoder, e.g., controllable and observable encoders.

Summary

- $\cdots u(i-1)u(i)\cdots \longrightarrow$ [Encoding] $\longrightarrow \cdots c(i-1)c(i)\cdots$

  $u(i) = (u_1(i), \ldots, u_k(i))$, $c(i) = (c_1(i), \ldots, c_n(i))$

- $u(D) = \sum_i u(i)D^i \longrightarrow$ [G(D)] $\longrightarrow c(D) = \sum_i c(i)D^i$

  $c(D) = u(D)G(D)$

- There are two types of codes in general
  - Block codes: $G(D) = G \Longrightarrow c(i) = u(i)G$
  - Convolutional codes: $G(D) = G_0 + G_1 D + \cdots + G_m D^m$

$$\Longrightarrow c(i) = u(i)G_0 + u(i-1)G_1 + \cdots u(i-m)G_m$$

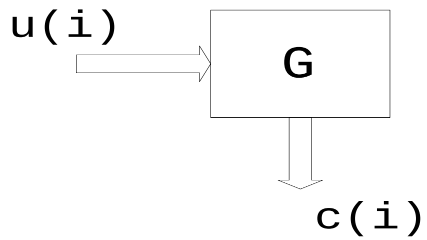


Figure 1: Encoder of block codes

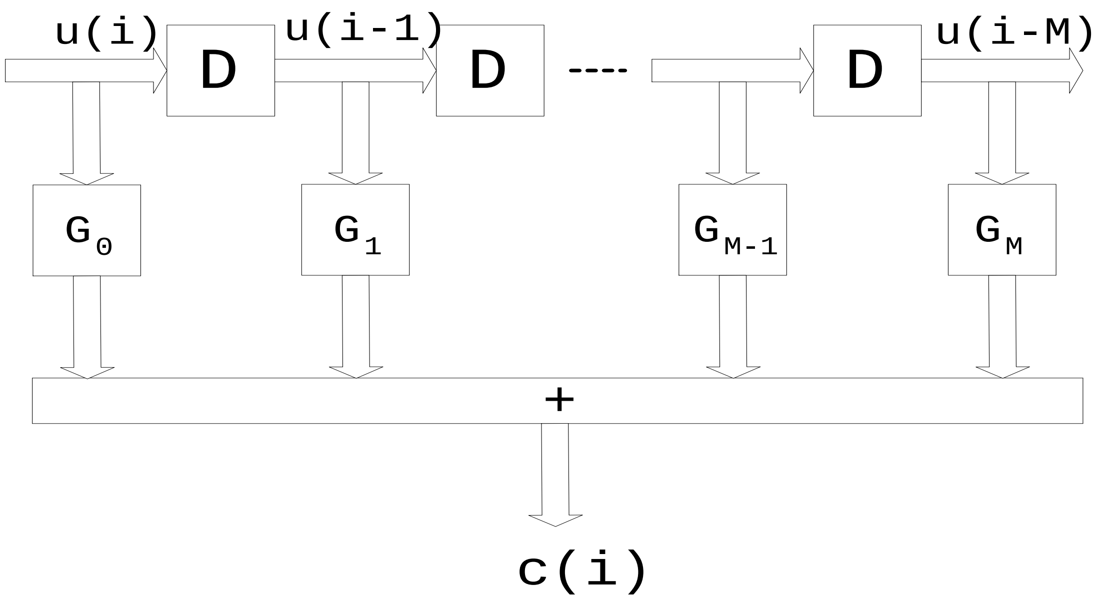


Figure 2: Encoder of convolutional codes

Shift register representation

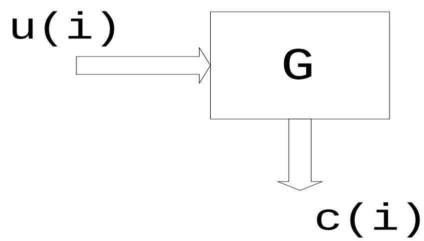


Figure 3: Encoder of block codes

- Use combinatorial logic to implement block codes.
- A information block $u(i)$ of length k at time i is mapped to a codeword $c(i)$ of length n at time i by a $k \times n$ generator matrix for each i, i.e., no memory.

$$c(i) = u(i) \cdot G$$

- We denote this linear block code by $C[n, k]$, usually, n and k are large.

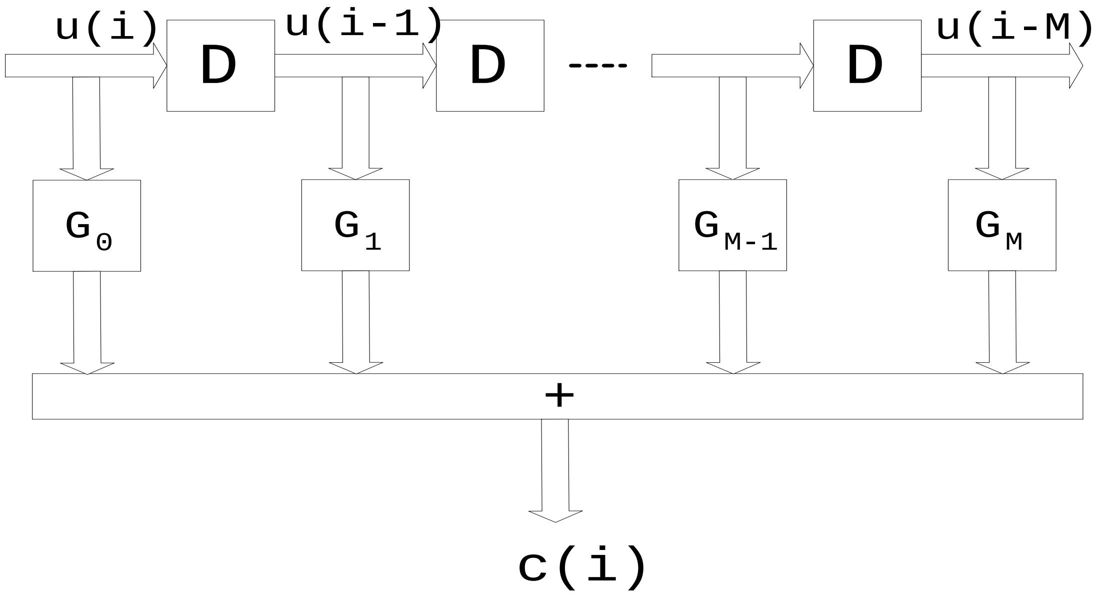


Figure 4: Encoder of convolutional codes

- Use sequential logic to implement convolutional codes.
- A information sequence u of infinite length is mapped to a codeword v of infinite length. In practice, we will output the convolutional codes by termination, truncation, or tailbiting.

- Assume we use feed forward encoder with memory m, then the codeword $c(i)$ of length n at time i is dependent on the current input $u(i)$ and previous m inputs, $u(i-1), \cdots, u(i-m)$.
- We need $m+1$ matrices $G_0, G_1, \cdots, G_m$ of size $k \times n$:

$$c(i) = u(i)G_0 + u(i-1)G_1 + u(i-2)G_2 + \cdots + u(i-m)G_m$$

- We denote this (linear) convolutional code by $C[n, k, m]$, usually, n and k are small.

Relation between block and convolutional codes

- A Convolutional code maps information blocks of length k to code blocks of length n. This linear mapping contains memory, because the code block depends on m previous information blocks.
- In this sense, block codes are a special case of convolutional codes, i.e., convolutional codes without memory.

- In practical application, convolutional codes have code sequences of finite length. When looking at the finite generator matrix of the created code in time domain, we find that it has a special structure.
- Because the generator matrix of a block code with corresponding dimension generally dose not have a special structure, convolutional codes with finite length can be considered as a special case of block codes.
- The trellis structure of convolutional codes is time-invariant, but the trellis structure of block codes is usually time-varying.

Scalar generator matrix in the time domain

G matrix of block codes

$$[u(0), u(1), u(2), \cdots] \begin{bmatrix} G & & & \\ & G & & \\ & & G & \\ & & & \ddots \end{bmatrix}$$

$$= [c(0), c(1), c(2), \cdots]$$

G matrix of convolutional codes

$$[u(0), u(1), u(2), \cdots] \begin{bmatrix} G_0 & G_1 & G_2 & \cdots & G_m & & & \cdots \\ & G_0 & G_1 & \cdots & G_{m-1} & G_m & & \cdots \\ & & G_0 & \cdots & G_{m-2} & G_{m-1} & G_m & \cdots \\ & & & \cdots & \vdots & \vdots & \vdots & \cdots \\ & & & & G_1 & G_2 & G_3 & \cdots \\ & & & & G_0 & G_1 & G_2 & \cdots \\ & & & & & G_0 & G_1 & \cdots \\ & & & & & & G_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$= [c(0), c(1), c(2), \cdots\cdots, c(m), c(m+1), \cdots]$$

- Here, we assume that the initial values in the m memories are all set to zeros.
- For example, the scalar matrix shows that

$$\begin{aligned} c(0) &= u(0)G_0 + u(-1)G_1 + u(-2)G_2 + \cdots + u(-m)G_m \\ &= u(0)G_0 \end{aligned}$$

since $u(-1) = u(-2) = \cdots u(-m) = 0$.

- Similarly

$$\begin{aligned} c(1) &= u(1)G_0 + u(0)G_1 + u(-1)G_2 + \cdots + u(-m+1)G_m \\ &= u(1)G_0 + u(0)G_1. \end{aligned}$$

- In general, we have

$$\begin{aligned} c(i) &= u(i)G_0 + u(i-1)G_1 + u(i-2)G_2 + \cdots + u(i-m)G_m \\ &= u(i) \otimes G_i. \end{aligned}$$

Impulse response of MIMO LTI systems

- From the scalar G matrix representation, we have

$$\begin{aligned} c(i) &= u(i)G_0 + u(i-1)G_1 + u(i-2)G_2 + \cdots + u(i-m)G_m \\ &= u(i) \otimes G_i \end{aligned}$$

- This is the form of discrete time convolutional sum, i.e., the output $c(i)$ is the convolutional sum of input sequence $u(i)$ and the finite impulse response (FIR) $(G_0, \cdots, G_m)$.
- In the undergraduate course of signal and system, we deal with SISO.
- Here, we have MIMO LTI systems with k inputs and n outputs and thus we need $k \times n$ impulse responses

$$g_i^{(j)} = g_i^{(j)}(l), \quad 0 \le l \le m, \;\; 1 \le i \le k, \text{ and } 1 \le j \le n,$$

which can be obtained from $m + 1$ matrices $\{G_0, \cdots, G_M\}$.

- Correspondingly, we can associate each $g_i^{(j)}$ with its Fourier transform $G_i^{(j)}(D)$ and form a $k \times n$ matrix $G(D)$ by

$$G(D) = [G_i^{(j)}(D)] = G_0 + G_1 D + G_2 D^2 + \cdots + G_m D^m$$

- This is the polynomial matrix representation of a convolutional code.
- The $k \times n$ matrix $g(l)$ consisting of impulse responses $g_i^{(j)}(l)$ and the $k \times n$ matrix $G(D)$ consisting of $G_i^{(j)}(D)$ form a Fourier pair.

Example

A (3,2) convolutional code with impulse response $g(l)$ and transfer function $G(D)$:

$$g(l) = \begin{pmatrix} 110 & 111 & 100 \\ 010 & 101 & 111 \end{pmatrix}$$

$$G(D) = \begin{pmatrix} 1+D & 1+D+D^2 & 1 \\ D & 1+D^2 & 1+D+D^2 \end{pmatrix}$$

LTI system representation

- Let us use u_i, c_i, (instead of $u(i)$, $c(i)$) to denote the information sequence, code sequence respectively, at time i.
- I.e., the infinite information and code sequence is

$$\begin{cases} u &= u_0 u_1 u_2 \cdots u_i \cdots \\ v &= v_0 v_1 v_2 \cdots v_i \cdots \end{cases}$$

- The u_i consists of k bits and v_i consists of n bits denoted by

$$\begin{cases} u_i &= u_i^{(1)} u_i^{(2)} u_i^{(3)} \cdots u_i^{(k)} \\ v_i &= v_i^{(1)} v_i^{(2)} v_i^{(3)} \cdots v_i^{(n)} \end{cases}$$

- Define the input sequence due to the ith stream, $1 \leq i \leq k$, as

$$u^{(i)} = u_0^{(i)} u_1^{(i)} u_2^{(i)} u_3^{(i)} \cdots$$

and the output sequence due to the jth stream, $1 \leq j \leq n$, as

$$v^{(j)} = v_0^{(j)} v_1^{(j)} v_2^{(j)} v_3^{(j)} \cdots$$

- A $[n, k, m]$ convolutional code can be represented as a MIMO LTI system with k input streams

$$(u^{(1)}, u^{(2)}, \cdots, u^{(k)}),$$

and n output streams

$$(v^{(1)}, v^{(2)}, \cdots, v^{(n)}),$$

and a $k \times n$ impulse response matrix $g(l) = \{g_i^{(j)}(l)\}$.

- The jth of the n output sequence $v^{(j)}$ is obtained by convolving the input sequence with the corresponding system impulse response

$$v^{(j)} = u^{(1)} \otimes g_1^{(j)} + u^{(2)} \otimes g_2^{(j)} + \cdots u^{(k)} \otimes g_k^{(j)} = \sum_{i=1}^{k} u^{(i)} \otimes g_i^{(j)}$$

- This is the origin of the name convolutional code.
- The impulse response $g_i^{(j)}$ of the ith input with the response to the jth output is found by stimulating the encoder with the discrete impulse $(1000\cdots)$ at the ith input and by observing the jth output when all other inputs are set to $(0000\cdots)$.

Polynomial generator matrix in frequency domain

Now introduce the delay operator D in the representation of input sequence, output sequence, and impulse response, i.e.,

1. Use z transform

$$u^{(i)} = u_0^{(i)} u_1^{(i)} u_2^{(i)} u_3^{(i)} \cdots \longleftrightarrow U_i(D) = \sum_{t=0}^{\infty} u_t^{(i)} D^t$$

$$v^{(j)} = v_0^{(j)} v_1^{(j)} v_2^{(j)} v_3^{(j)} \cdots \longleftrightarrow V_i(D) = \sum_{t=0}^{\infty} v_t^{(j)} D^t$$

$$g_i^{(j)} = (g_i^{(j)}(0), \cdots, g_i^{(j)}(m)) \longleftrightarrow G_i^{(j)}(D) = \sum_{l=0}^{m} g_i^{(j)}(l) D^l$$

2. $z\{u * g\} = U(D)G(D) = V(D)$

3. $V_j(D) = \sum_{i=1}^{k} U_i(D) \cdot G_i^{(j)}(D)$

We thus have

$$V(D) = U(D) \cdot G(D)$$

, where

$$\begin{aligned} U(D) &= (U_1(D), U_2(D), \ldots, U_k(D)) \\ V(D) &= (V_1(D), V_2(D), \ldots, V_n(D)) \\ G(D) &= \left(\quad G_i^{(j)}(D) \quad \right) \end{aligned}$$

Example 1

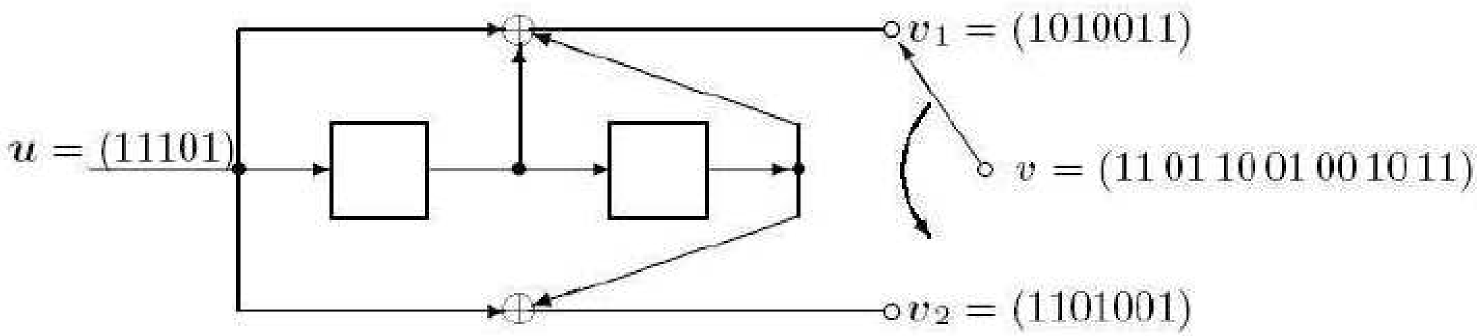


Figure 5: (2,1,2) convolutional code encoder

Input: $u = (1, 1, 1, 0, 1)$ in time domain

In z domain:

$$\begin{aligned}
G^{(1)}(D) &= 1 + D + D^2 \\
G^{(2)}(D) &= 1 + D^2 \\
G(D) &= [1 + D + D^2, 1 + D^2] \\
U(D) &= 1 + D + D^2 + D^4 \\
V(D) &= U(D) \bullet G(D) \\
V_1(D) &= 1 + D^2 + D^5 + D^6 \\
V_2(D) &= 1 + D + D^3 + D^6
\end{aligned}$$

In time domain:

$$\begin{aligned}
v_1 &= (1, 0, 1, 0, 0, 1, 1) \\
v_2 &= (1, 1, 0, 1, 0, 0, 1)
\end{aligned}$$

Example 2

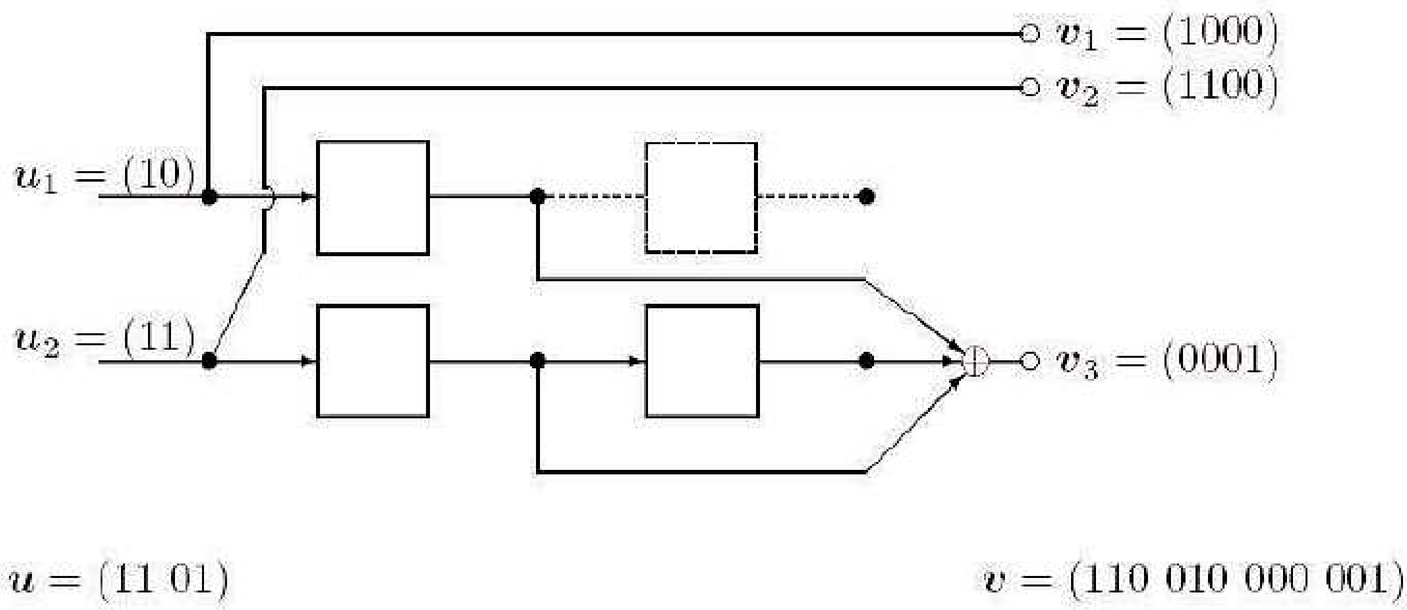


Figure 6: (3,2,2) convolutional code encoder

$$V_1(D) = U_1(D)$$

$$V_2(D) = U_2(D)$$

$$V_3(D) = U_1(D) \bullet D + U_2(D) \bullet (D + D^2)$$

$$\begin{bmatrix} V_1(D) & V_2(D) & V_3(D) \end{bmatrix} = \begin{bmatrix} U_1(D) & U_2(D) \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & D \\ 0 & 1 & D + D^2 \end{bmatrix}$$

$$G_1(D) = \begin{bmatrix} 1 & 0 & D \\ 0 & 1 & D + D^2 \end{bmatrix}$$

$$U_1 = 1 \quad U_2 = 1 + D$$

$$V = \begin{bmatrix} 1 & 1 + D \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & D \\ 0 & 1 & D + D^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 + D & D^3 \end{bmatrix}$$

$$v_1 = (1, 0, 0, 0) \quad v_2 = (1, 1, 0, 0) \quad v_3 = (0, 0, 0, 1)$$

State diagram, tree, and trellis

State diagram

- Convolutional code 的編碼器，可看成一個 finite state machine
- 輸入(shift register)的內容，可用來描述 states，輸出 v_t 在時間 t 的值，由當時所在的 state σ_t 與 輸入 u_t 來決定
- 在 state diagram 上， nodes 為可能的 state ，路徑上標示著輸入與輸出 (u_t/v_t)

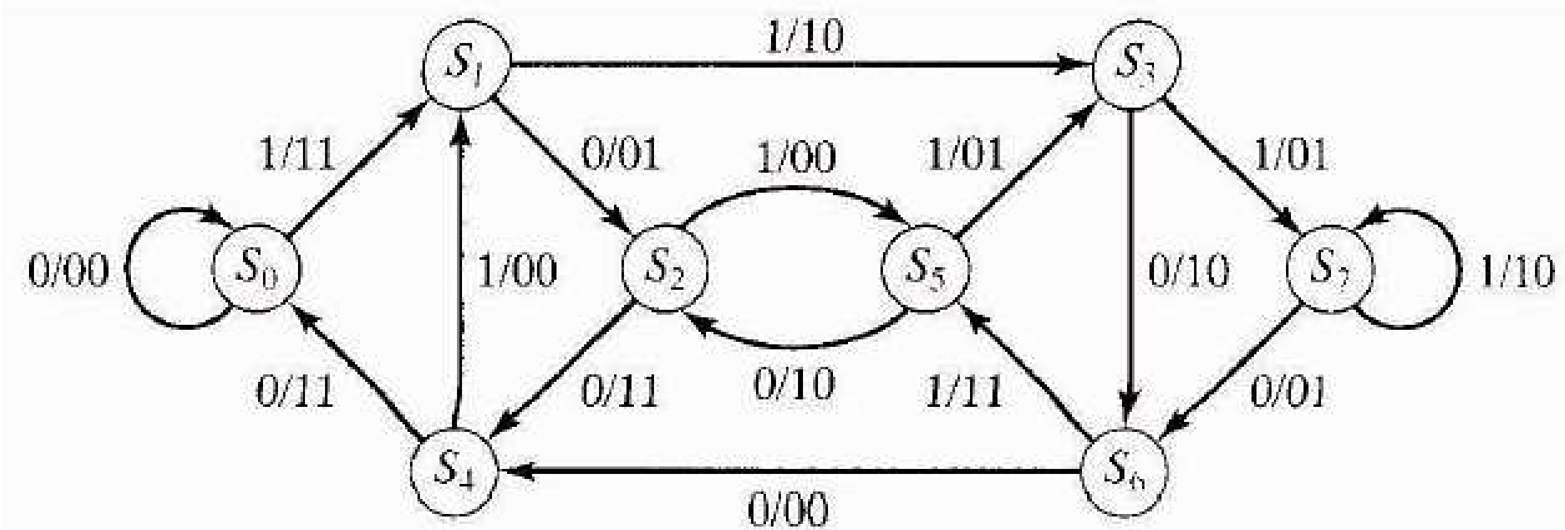


Figure 7: state diagram of (2,1,2) Convolutional code

Code Tree of Convolutional code

- 一個 (n,k,m) Convolutional code 的 codeword 可視為 code tree 上面的一個路徑
- 輸入的長度為 h ， code tree 會有 $(h+m+1)$ 個 level ，最左邊的 node(level 0) 稱為 origin node
- 在最先的 h levels， 每個 node 存在 2^k 個 branch ，位在 level (h+m)，最右邊的那 2^{hk} 個 nodes ，稱為 terminal nodes
- 從 origin node 到 terminal 的路徑稱為 code path，可用來表示一個 code word

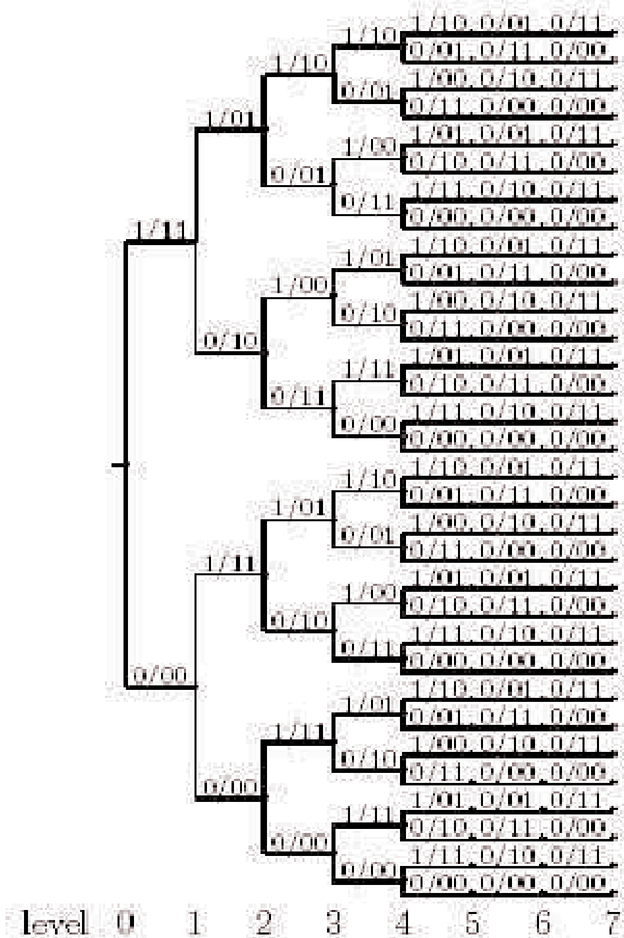


Figure 8: state tree of (2,1,2) Convolutional code

Trellises of Convolutional code

- 將 code tree 上的 node，有結構性的合併在相同的 state ，稱為 Convolutional code 的 code trellis
- 對於一 (n,k,m) 的 Convolutional code，state 的數量在 level m 為 2^K，當 $K = \sum_{j=1}^{k} K_j$,K_j 為第 j 個輸入的長度，在此 level 上，存在 2^K 個 nodes
- terminal node 只有一點，且會回到最初的狀態
- 從 origin node 到 terminal node 的路徑，可用來表示一個 codeword

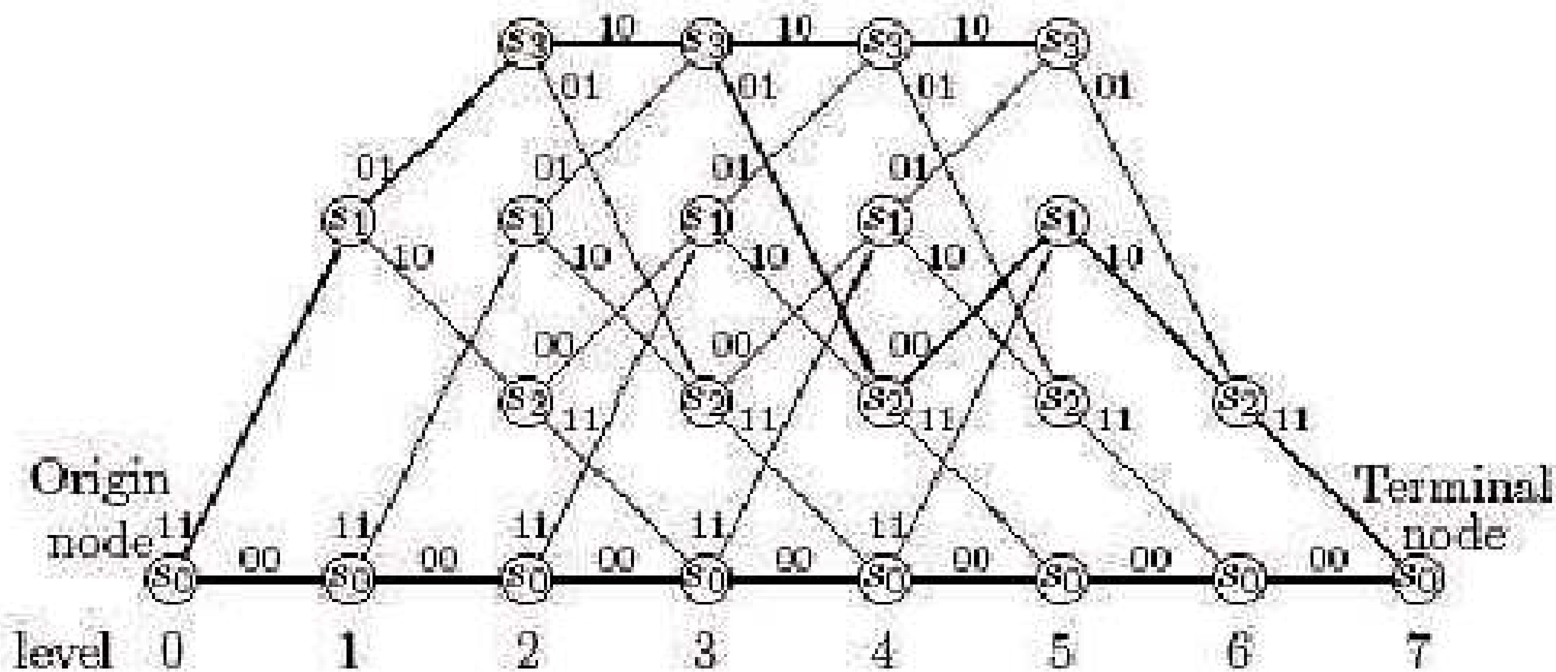


Figure 9: trellis of (2,1,2) Convolutional code

Structural properties of Convolutional codes

Convolutional encoder is a linear sequential circuit, it's operation can be describe by a state diagram.The state of an encoder is defined as its shift register contents.

For an (n,k,v) encoder

The encoder state σ_l at time unit l is the binary v-tuple

$$\sigma_l = (s_{l-1}^{(1)} s_{l-2}^{(1)} \ldots s_{l-v_1}^{(1)} s_{l-1}^{(2)} s_{l-2}^{(2)} \ldots s_{l-v_2}^{(2)} \cdots s_{l-1}^{(k)} s_{l-2}^{(k)} \ldots s_{l-v_k}^{(k)})$$

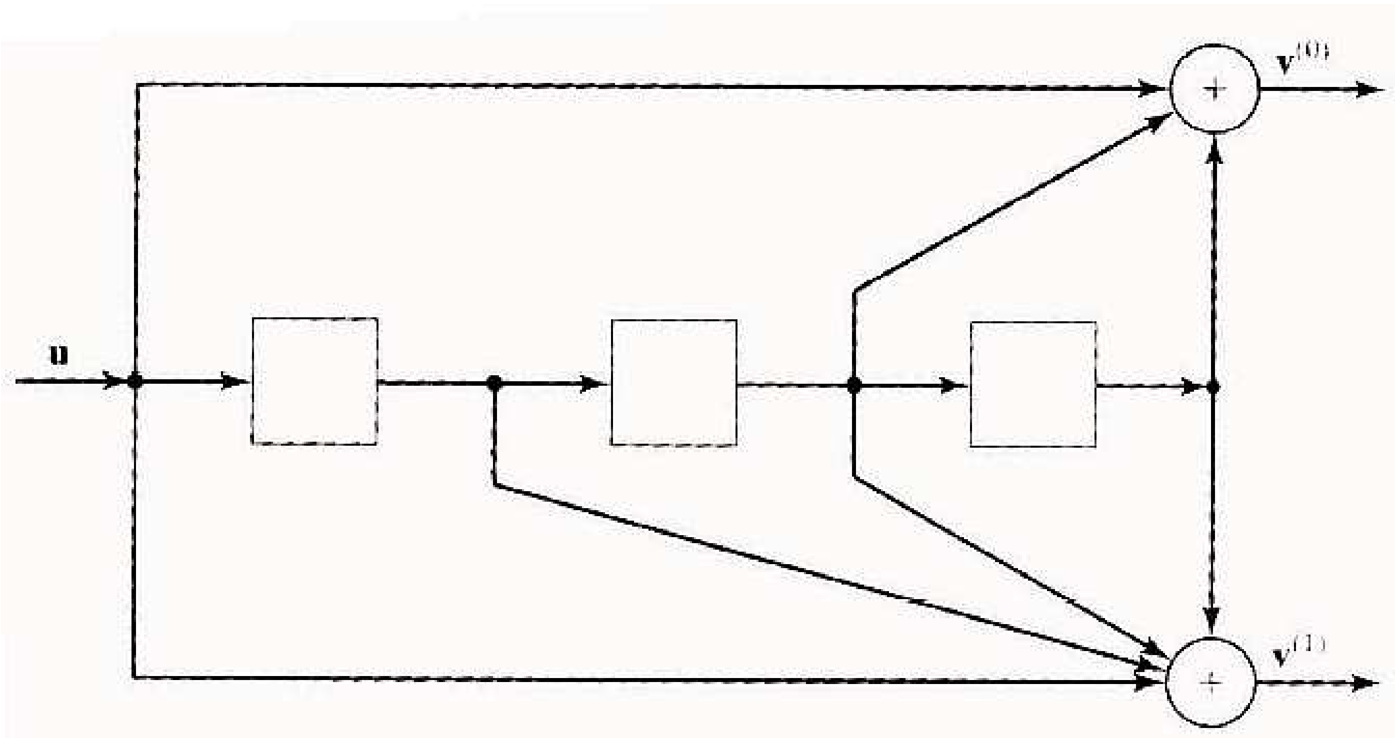
Each branch in the state diagram is labeled with the k inputs

$$(u_l^{(1)}, u_l^{(2)}, \ldots, u_l^{(k)})$$

causing the transition and the n corresponding output

$$(v_l^{(1)}, v_l^{(2)}, \ldots, v_l^{(n-1)})$$

(2,1,3) encoder



State diagrams for (2,1,3) encoder