Introduction to Binary Convolutional Codes [1]

Yunghsiang S. Han

Shenzhen Institute for Advanced Study, UESTC

E-mail: yunghsiangh@gmail.com

电子科技大学

University of Electronic Science and Technology of China

Binary Convolutional Codes

1. A binary convolutional code is denoted by a three-tuple (n, k, m).
2. n output bits are generated whenever k input bits are received.
3. The current n outputs are linear combinations of the present k input bits and the previous $m \times k$ input bits.
4. m designates the number of previous k-bit input blocks that must be memorized in the encoder.
5. m is called the *memory order* of the convolutional code.

Encoders for the Convolutional Codes

1. A binary convolutional encoder is conveniently structured as a mechanism of shift registers and modulo-2 adders, where the output bits are modular-2 additions of selective shift register contents and present input bits.

2. n in the three-tuple notation is exactly the number of output sequences in the encoder.

3. k is the number of input sequences (and hence, the encoder consists of k shift registers).

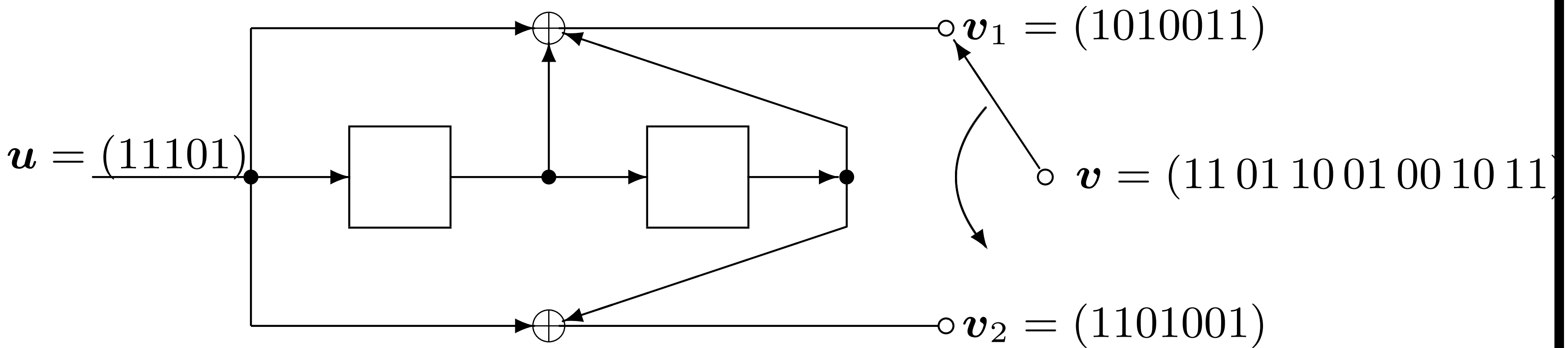
4. m is the maximum length of the k shift registers (i.e., if the number of stages of the jth shift register is K_j, then $m = \max_{1 \le j \le k} K_j$).

5. $K = \sum_{j=1}^{k} K_j$ is the total memory in the encoder (K is sometimes called the *overall constraint lengths*).

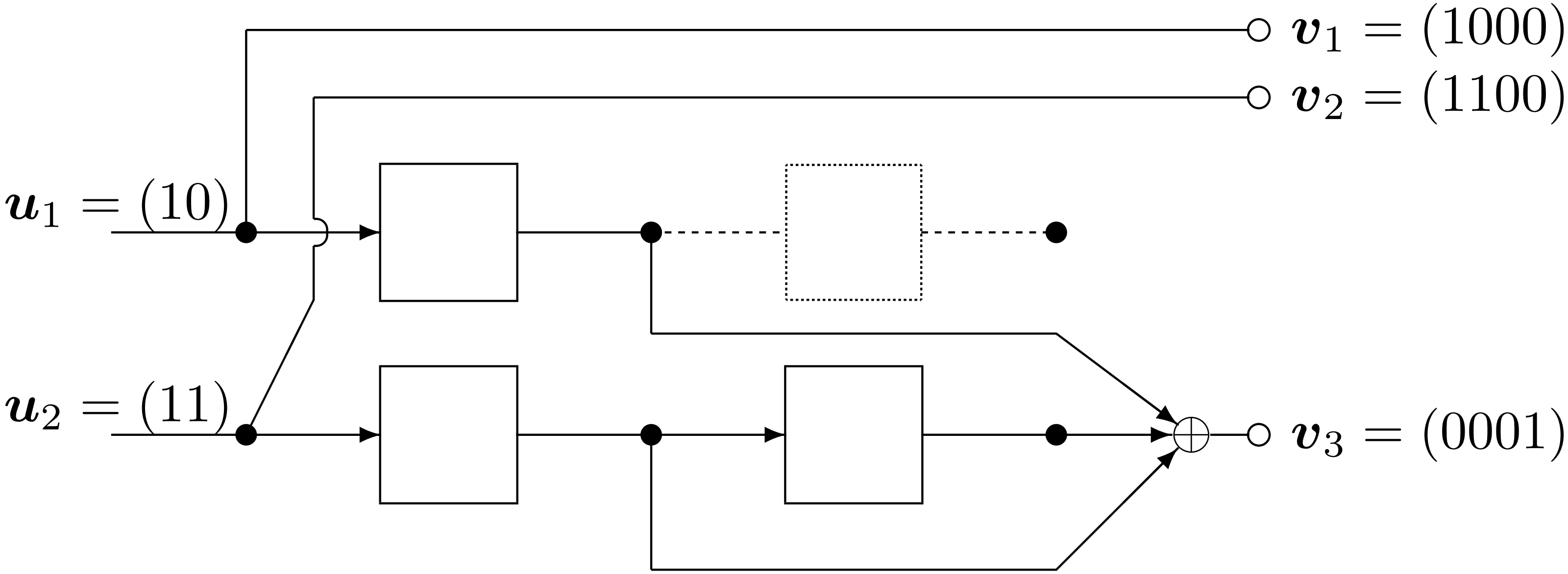
6. The definition of constraint length of a convolutional code is defined in several ways. The most popular one is $m + 1$.

Encoder for the Binary $(2, 1, 2)$ Convolutional Code

$\boldsymbol{u}$ is the information sequence and $\boldsymbol{v}$ is the corresponding code sequence (codeword).



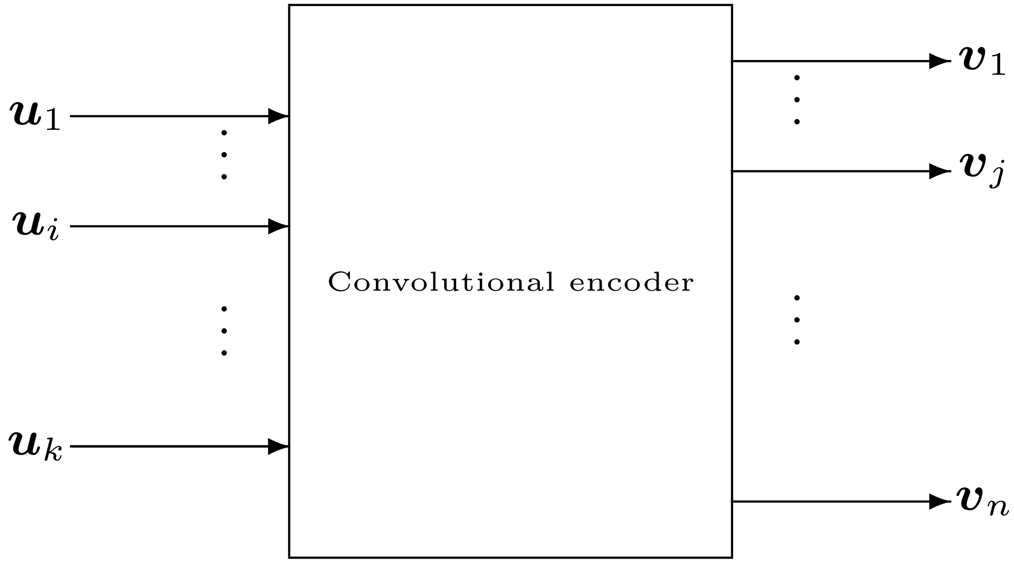
Encoder for the Binary $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$

$\boldsymbol{v} = (110\ 010\ 000\ 001)$

Impose Response and Convolution



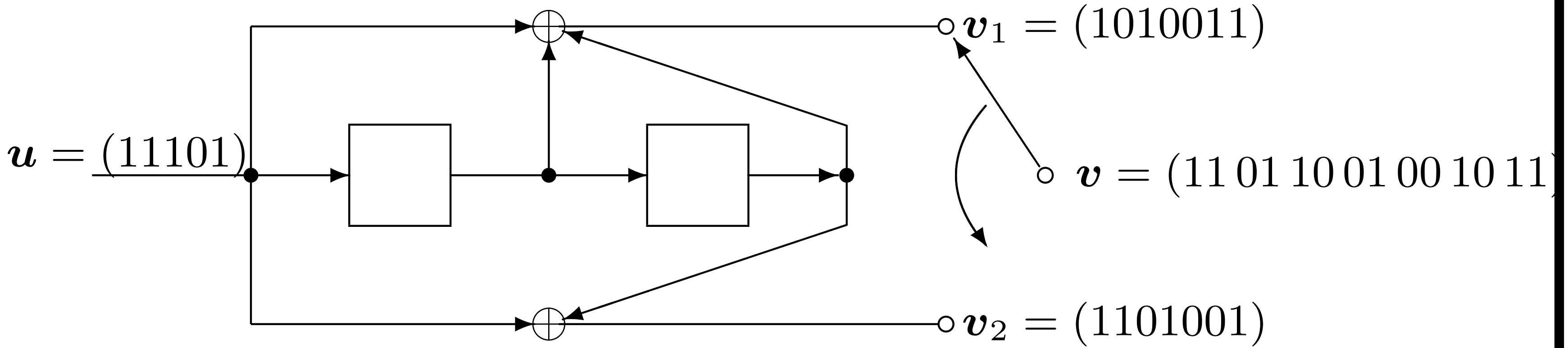
1. The encoders of convolutional codes can be represented by *linear time-invariant* (LTI) systems.
2. $$\boldsymbol{v}_j = \boldsymbol{u}_1 * \boldsymbol{g}_j^{(1)} + \boldsymbol{u}_2 * \boldsymbol{g}_j^{(2)} + \cdots + \boldsymbol{u}_k * \boldsymbol{g}_j^{(k)} = \sum_{i=1}^{k} \boldsymbol{u}_i * \boldsymbol{g}_j^{(k)},$$

where $*$ is the convolutional operation and $g_j^{(i)}$ is the impulse response of the ith input sequence with the response to the jth output.

3. $\boldsymbol{g}_j^{(i)}$ can be found by stimulating the encoder with the discrete impulse $(1, 0, 0, \ldots)$ at the ith input and by observing the jth output when all other inputs are fed the zero sequence $(0, 0, 0, \ldots)$.

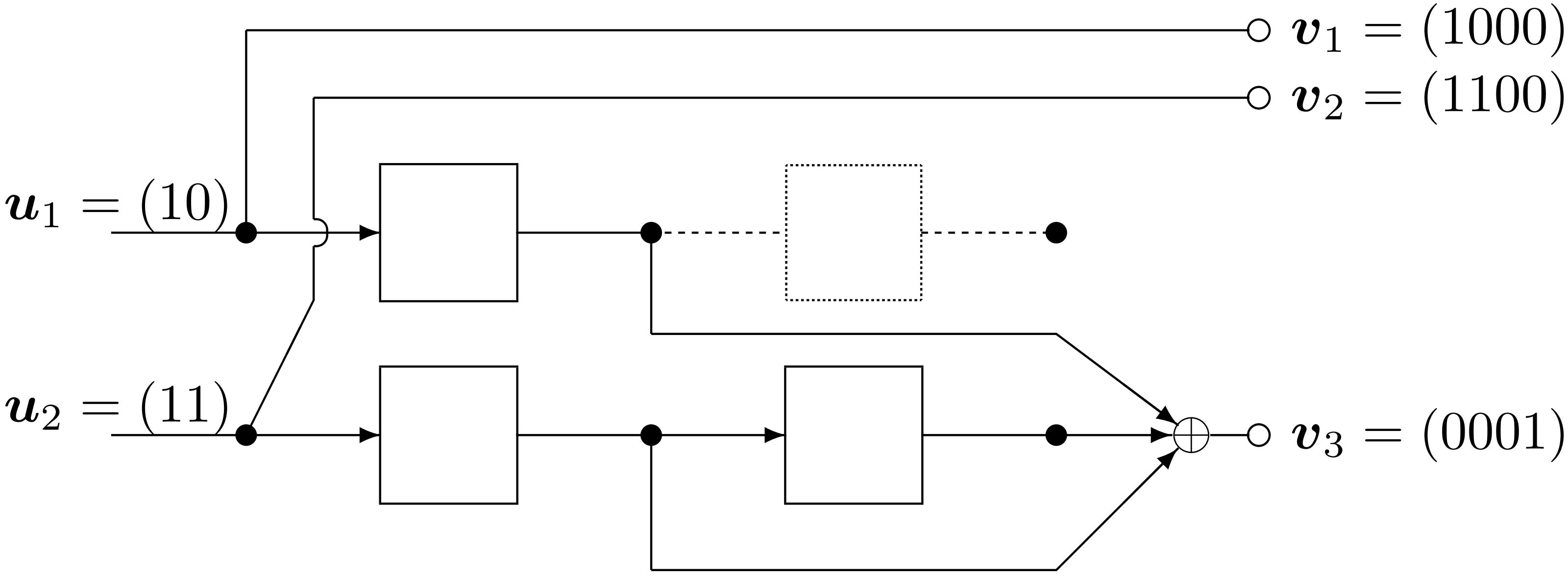
4. The impulse responses are called *generator sequences* of the encoder.

Impulse Response for the Binary $(2, 1, 2)$ Convolutional Code



$$
\begin{aligned}
\boldsymbol{g}_1 &= (1, 1, 1, 0, \ldots) = (1, 1, 1), \ \boldsymbol{g}_2 = (1, 0, 1, 0, \ldots) = (1, 0, 1) \\
\boldsymbol{v}_1 &= (1, 1, 1, 0, 1) * (1, 1, 1) = (1, 0, 1, 0, 0, 1, 1) \\
\boldsymbol{v}_2 &= (1, 1, 1, 0, 1) * (1, 0, 1) = (1, 1, 0, 1, 0, 0, 1)
\end{aligned}
$$

Impulse Response for the $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$ $\boldsymbol{v} = (110\ 010\ 000\ 001)$

$\boldsymbol{g}_1^{(1)} = 4$ (octal)$= (1, 0, 0)$, $\boldsymbol{g}_1^{(2)} = 0$ (octal), $\boldsymbol{g}_2^{(1)} = 0$ (octal), $\boldsymbol{g}_2^{(2)} = 4$ (octal), $\boldsymbol{g}_3^{(1)} = 2$ (octal), $\boldsymbol{g}_3^{(2)} = 3$ (octal)

Generator Matrix in the Time Domain

1. The convolutional codes can be generated by a generator matrix multiplied by the information sequences.
2. Let $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k$ are the information sequences and $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ the output sequences.
3. Arrange the information sequences as

$$\begin{aligned}\boldsymbol{u} &= (u_{1,0}, u_{2,0}, \ldots, u_{k,0}, u_{1,1}, u_{2,1}, \ldots, u_{k,1},\\ &\quad \ldots, u_{1,\ell}, u_{2,\ell}, \ldots, u_{k,\ell}, \ldots)\\ &= (\boldsymbol{w}_0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_\ell, \ldots),\end{aligned}$$

and the output sequences as

$$\begin{aligned}\boldsymbol{v} &= (v_{1,0}, v_{2,0}, \ldots, v_{n,0}, v_{1,1}, v_{2,1}, \ldots, v_{n,1},\\ &\quad \ldots, v_{1,\ell}, v_{2,\ell}, \ldots, v_{n,\ell}, \ldots)\\ &= (\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_\ell, \ldots)\end{aligned}$$

4. $\boldsymbol{v}$ is called a codeword or code sequence.
5. The relation between $\boldsymbol{v}$ and $\boldsymbol{u}$ can characterized as

$$\boldsymbol{v} = \boldsymbol{u} \cdot \boldsymbol{G},$$

where $\boldsymbol{G}$ is the generator matrix of the code.

6. The generator matrix is

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{G}_0 & \boldsymbol{G}_1 & \boldsymbol{G}_2 & \cdots & \boldsymbol{G}_m & & \\ & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_{m-1} & \boldsymbol{G}_m & \\ & & \boldsymbol{G}_0 & \cdots & \boldsymbol{G}_{m-2} & \boldsymbol{G}_{m-1} & \boldsymbol{G}_m \\ & & & \ddots & & & \ddots \end{pmatrix},$$

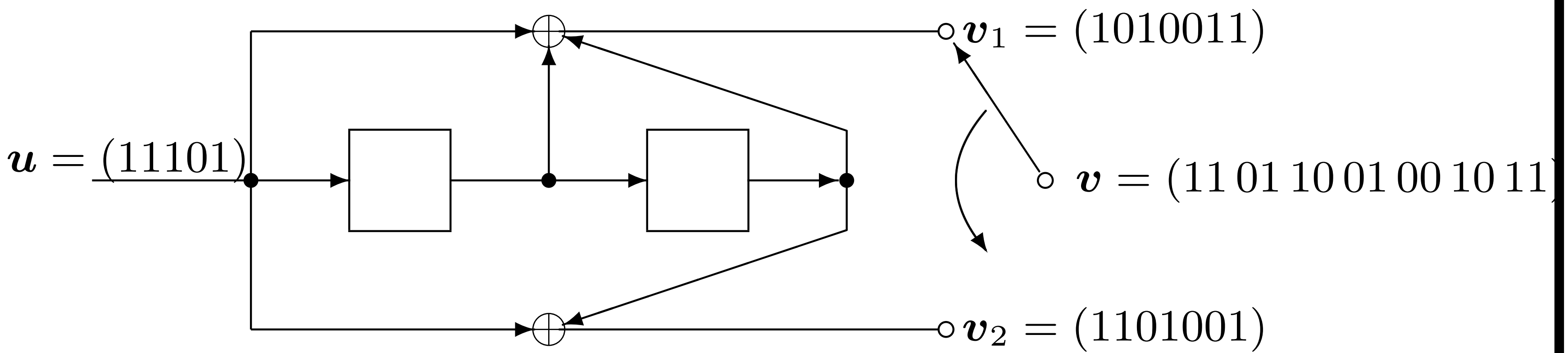
with the $k \times n$ submatrices

$$\boldsymbol{G}_\ell = \begin{pmatrix} g_{1,\ell}^{(1)} & g_{2,\ell}^{(1)} & \cdots & g_{n,\ell}^{(1)} \\ g_{1,\ell}^{(2)} & g_{2,\ell}^{(2)} & \cdots & g_{n,\ell}^{(2)} \\ \vdots & \vdots & & \vdots \\ g_{1,\ell}^{(k)} & g_{2,\ell}^{(k)} & \cdots & g_{n,\ell}^{(k)} \end{pmatrix}.$$

7. The element $g_{j,\ell}^{(i)}$, for $i \in [1,k]$ and $j \in [1,n]$, are the impulse response of the ith input with respect to jth output:

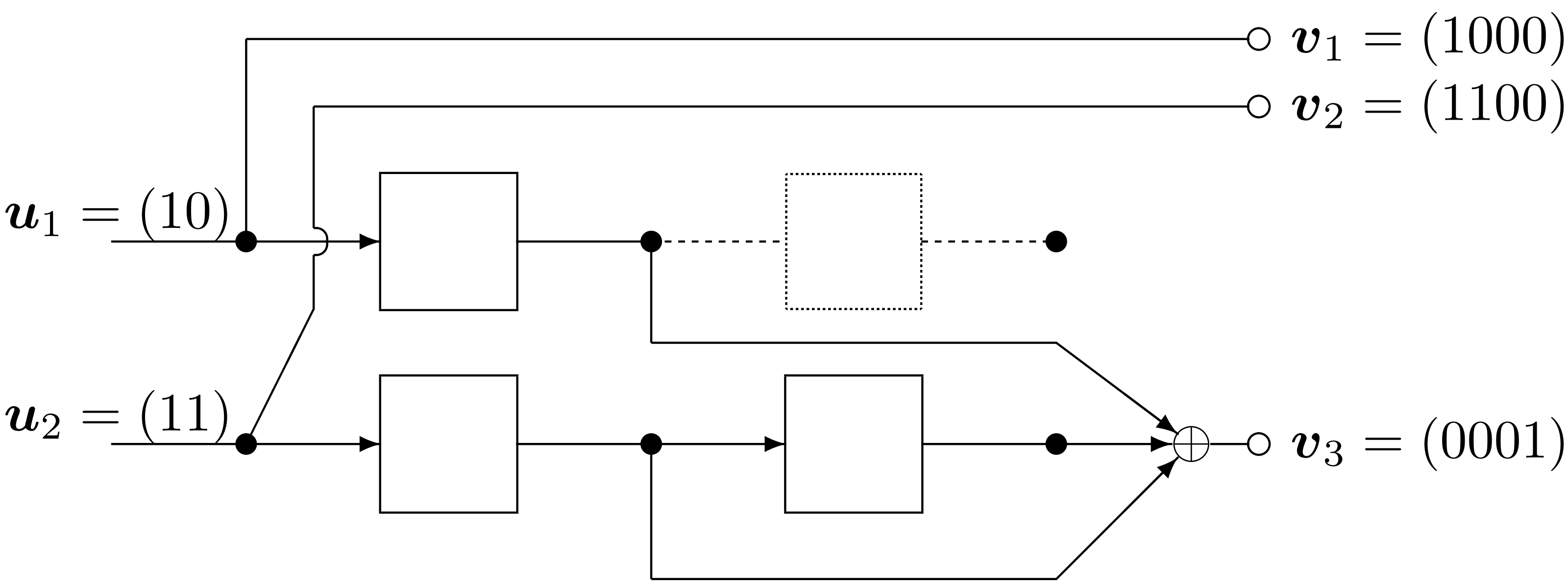
$$\boldsymbol{g}_j^{(i)} = (g_{j,0}^{(i)}, g_{j,1}^{(i)}, \ldots, g_{j,\ell}^{(i)}, \ldots, g_{j,m}^{(i)}).$$

Generator Matrix of the Binary $(2, 1, 2)$ Convolutional Code



$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \boldsymbol{G} \\
&= (1, 1, 1, 0, 1) \cdot \begin{pmatrix}
11 & 10 & 11 & & & & \\
 & 11 & 10 & 11 & & & \\
 & & 11 & 10 & 11 & & \\
 & & & 11 & 10 & 11 & \\
 & & & & 11 & 10 & 11
\end{pmatrix} \\
&= (11, 01, 10, 01, 00, 10, 11)
\end{aligned}
$$

Generator Matrix of the Binary $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$ $\boldsymbol{v} = (110\ 010\ 000\ 001)$

$\boldsymbol{g}_1^{(1)} = 4$ (octal)$= (1, 0, 0)$, $\boldsymbol{g}_1^{(2)} = 0$ (octal), $\boldsymbol{g}_2^{(1)} = 0$ (octal),

$\boldsymbol{g}_2^{(2)} = 4$ (octal), $\boldsymbol{g}_3^{(1)} = 2$ (octal), $\boldsymbol{g}_3^{(2)} = 3$ (octal)

$$\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \boldsymbol{G} \\
&= (11, 01) \cdot \begin{pmatrix} 100 & 001 & 000 & \\ 010 & 001 & 001 & \\ & 100 & 001 & 000 \\ & 010 & 001 & 001 \end{pmatrix} \\
&= (110, 010, 000, 001)
\end{aligned}$$

Generator Matrix in the Z Domain

1. According to the Z transform,

$$\begin{aligned}
\boldsymbol{u}_i \quad &\multimap \quad U_i(D) = \sum_{t=0}^{\infty} u_{i,t} D^t \\
\boldsymbol{v}_j \quad &\multimap \quad V_j(D) = \sum_{t=0}^{\infty} v_{j,t} D^t \\
\boldsymbol{g}_j^{(i)} \quad &\multimap \quad G_{i,j}(D) = \sum_{t=0}^{\infty} g_{j,t}^{(i)} D^t
\end{aligned}$$

2. The convolutional relation of the Z transform $Z\{\boldsymbol{u} * \boldsymbol{g}\} = U(D)G(D)$ is used to transform the convolution of input sequences and generator sequences to a multiplication in the Z domain.

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

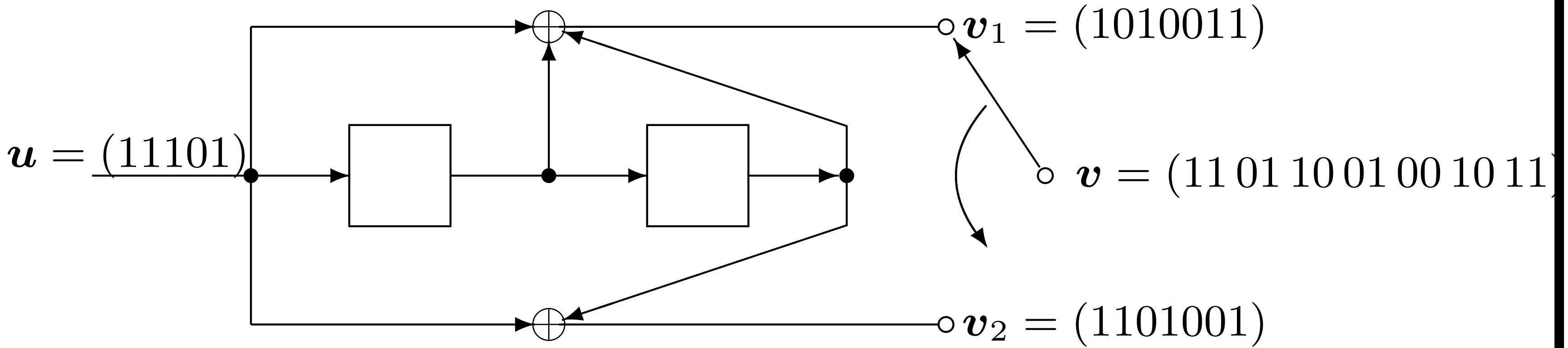
4. We can write the above equations into a matrix multiplication:

$$\boldsymbol{V}(D) = \boldsymbol{U}(D) \cdot \boldsymbol{G}(D),$$

where

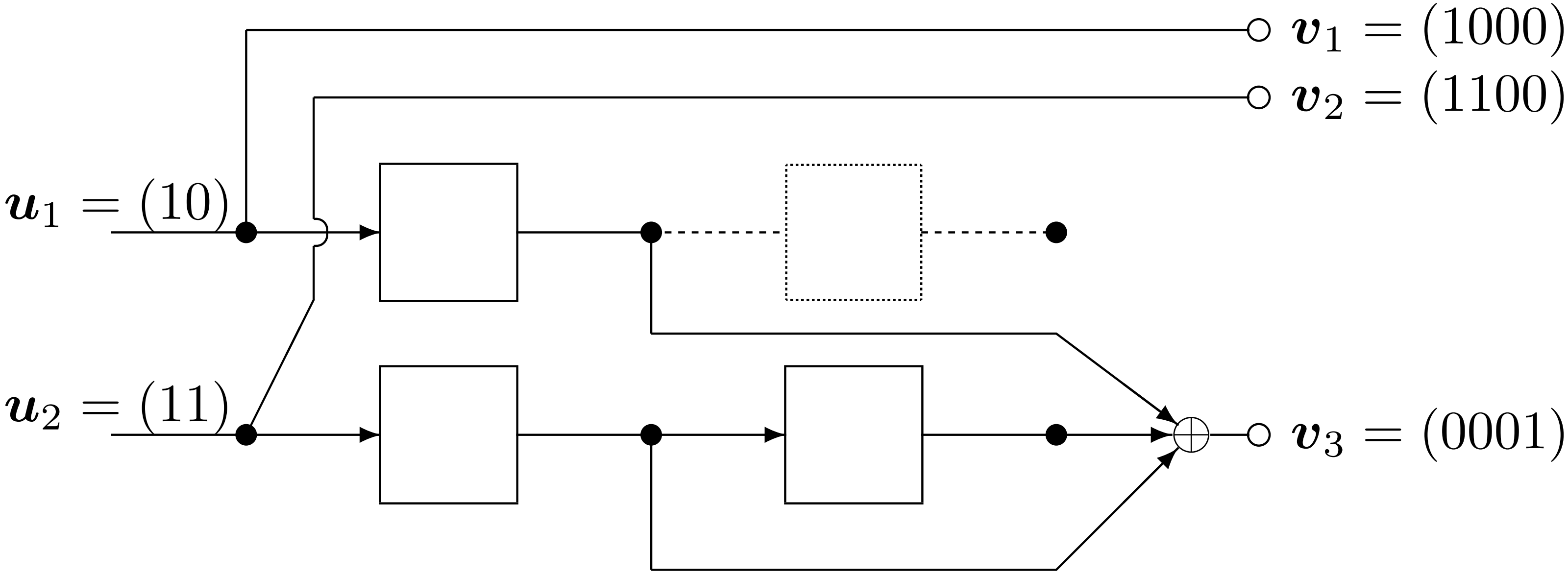
$$\begin{aligned}
\boldsymbol{U}(D) &= (U_1(D), U_2(D), \ldots, U_k(D)) \\
\boldsymbol{V}(D) &= (V_1(D), V_2(D), \ldots, V_n(D)) \\
\boldsymbol{G}(D) &= \left(\begin{array}{c} \\ G_{i,j} \\ \\ \end{array} \right)
\end{aligned}$$

Generator Matrix of the Binary $(2, 1, 2)$ Convolutional Code



$$
\begin{aligned}
\boldsymbol{g}_1 &= (1,1,1,0,\ldots) = (1,1,1), \\
\boldsymbol{g}_2 &= (1,0,1,0,\ldots) = (1,0,1) \\
G_{1,1}(D) &= 1 + D + D^2 \\
G_{1,2}(D) &= 1 + D^2 \\
U_1(D) &= 1 + D + D^2 + D^4 \\
V_1(D) &= 1 + D^2 + D^5 + D^6 \\
V_2(D) &= 1 + D + D^3 + D^6
\end{aligned}
$$

Generator Matrix of the Binary $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$ $\boldsymbol{v} = (110\ 010\ 000\ 001)$

$$
\begin{aligned}
\boldsymbol{g}_1^{(1)} &= (1,0,0),\ \boldsymbol{g}_1^{(2)} = (0,0,0),\ \boldsymbol{g}_2^{(1)} = (0,0,0) \\
\boldsymbol{g}_2^{(2)} &= (1,0,0),\ \boldsymbol{g}_3^{(1)} = (0,1,0),\ \boldsymbol{g}_3^{(2)} = (0,1,1) \\
G_{1,1}(D) &= 1,\ G_{1,2}(D) = 0,\ G_{1,3}(D) = D \\
G_{2,1}(D) &= 0,\ G_{2,2} = 1,\ G_{2,3}(D) = D + D^2 \\
U_1(D) &= 1,\ U_2(D) = 1 + D \\
V_1(D) &= 1,\ V_2(D) = 1 + D,\ V_3(D) = D^3
\end{aligned}
$$

Termination

1. The *effective code rate*, $R_{\text{effective}}$, is defined as the average number of input bits carried by an output bit.
2. In practice, the input sequences are with finite length.
3. In order to terminate a convolutional code, some bits are appended onto the information sequence such that the shift registers return to the zero.
4. Each of the k input sequences of length L bits is padded with m zeros, and these k input sequences jointly induce $n(L+m)$ output bits.
5. The effective rate of the terminated convolutional code is now

$$R_{\text{effective}} = \frac{kL}{n(L+m)} = R\frac{L}{L+m},$$

where $\frac{L}{L+m}$ is called the *fractional rate loss*.

6. When L is large, $R_{\text{effective}} \approx R$.
7. All examples presented are terminated convolutional codes.

Truncation

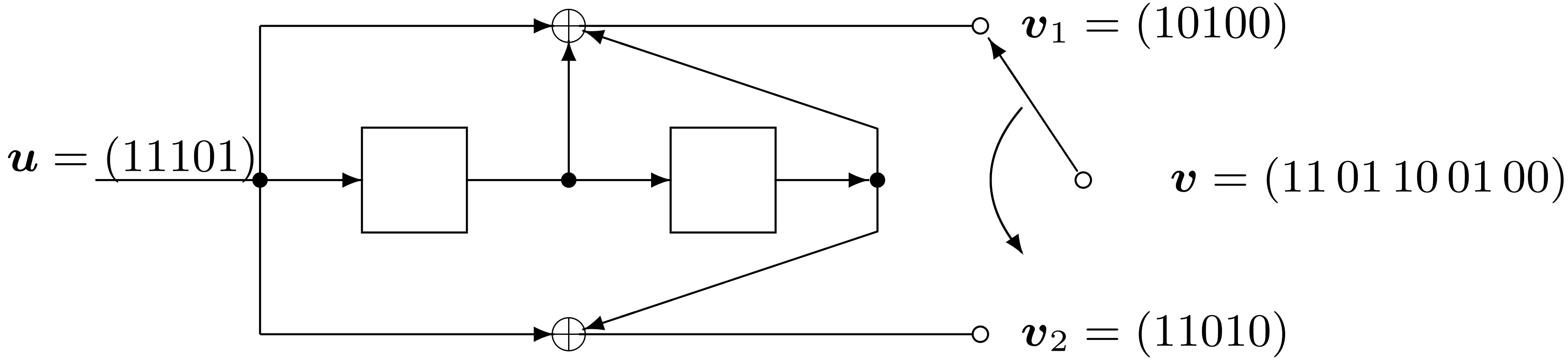
1. The second option to terminate a convolutional code is to stop for $t > L$ no matter what contents of shift registers have.
2. The effective code rate is still R.
3. The generator matrix is clipped after the Lth column:

$$\boldsymbol{G}^c_{[L]} = \begin{pmatrix} \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & & & \\ & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & & \\ & & \ddots & & \ddots & & & \\ & & & \boldsymbol{G}_0 & & & \boldsymbol{G}_m & \\ & & & & \ddots & & \vdots & \\ & & & & & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \\ & & & & & & \boldsymbol{G}_0 & \end{pmatrix},$$

where $\boldsymbol{G}^c_{[L]}$ is an $(k \cdot L) \times (n \cdot L)$ matrix.

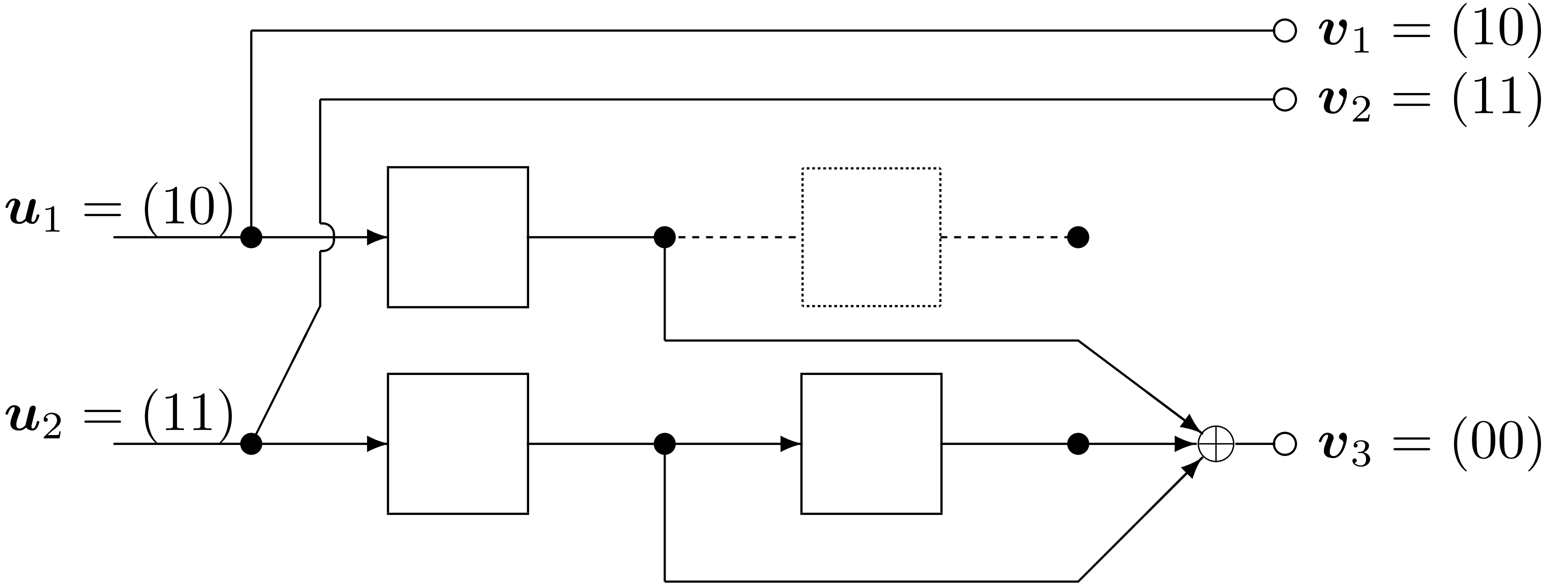
4. The drawback of truncation method is that the last few blocks of information sequences are less protected.

Generator Matrix of the Truncation Binary $(2, 1, 2)$ Convolutional Code



$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \boldsymbol{G}^{c}_{[5]} \\
&= (1,1,1,0,1) \cdot \begin{pmatrix} 11 & 10 & 11 & & \\ & 11 & 10 & 11 & \\ & & 11 & 10 & 11 \\ & & & 11 & 10 \\ & & & & 11 \end{pmatrix} \\
&= (11,01,10,01,00)
\end{aligned}
$$

Generator Matrix of the Truncation Binary $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$ $\boldsymbol{v} = (110\ 010)$

$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \boldsymbol{G}^{c}_{[2]} \\
&= (11, 01) \cdot \begin{pmatrix} 100 & 001 \\ 010 & 001 \\ & 100 \\ & 010 \end{pmatrix} \\
&= (110, 010)
\end{aligned}
$$

Tail Biting

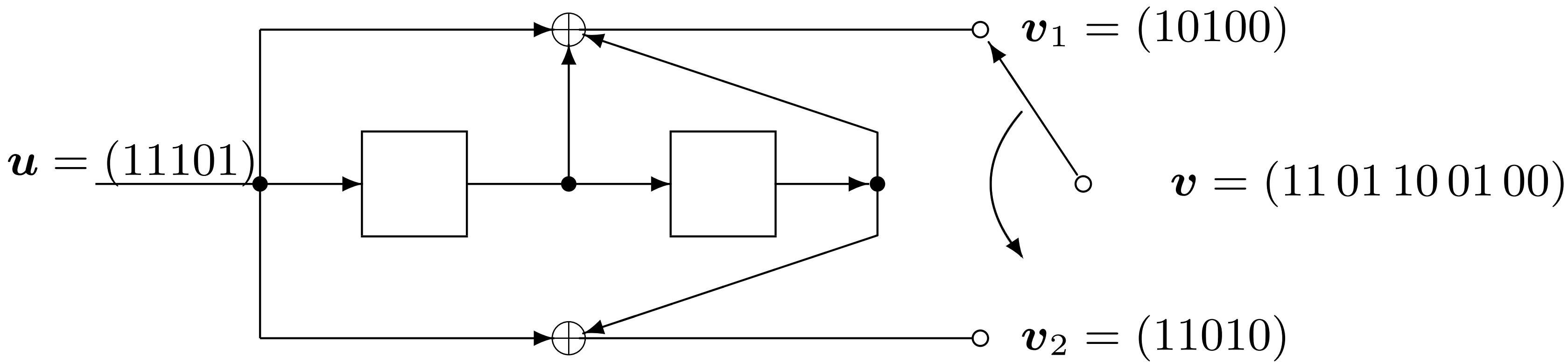
1. The third possible method to generate finite code sequences is called tail biting.
2. Tail biting is to start the convolutional encoder in the same contents of all shift registers (state) where it will stop after the input of L information blocks.
3. Equal protection of all information bits of the entire information sequences is possible.
4. The effective rate of the code is still R.
5. The generator matrix has to be clipped after the Lth column

and manipulated as follows:

$$\tilde{\boldsymbol{G}}^{c}_{[L]} = \begin{pmatrix} \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & & \\ & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & \\ & & \ddots & & \ddots & & \\ & & & \boldsymbol{G}_0 & & & \boldsymbol{G}_m \\ \boldsymbol{G}_m & & & & \ddots & & \vdots \\ \vdots & \ddots & & & & \boldsymbol{G}_0 & \boldsymbol{G}_1 \\ \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & & & \boldsymbol{G}_0 \end{pmatrix},$$

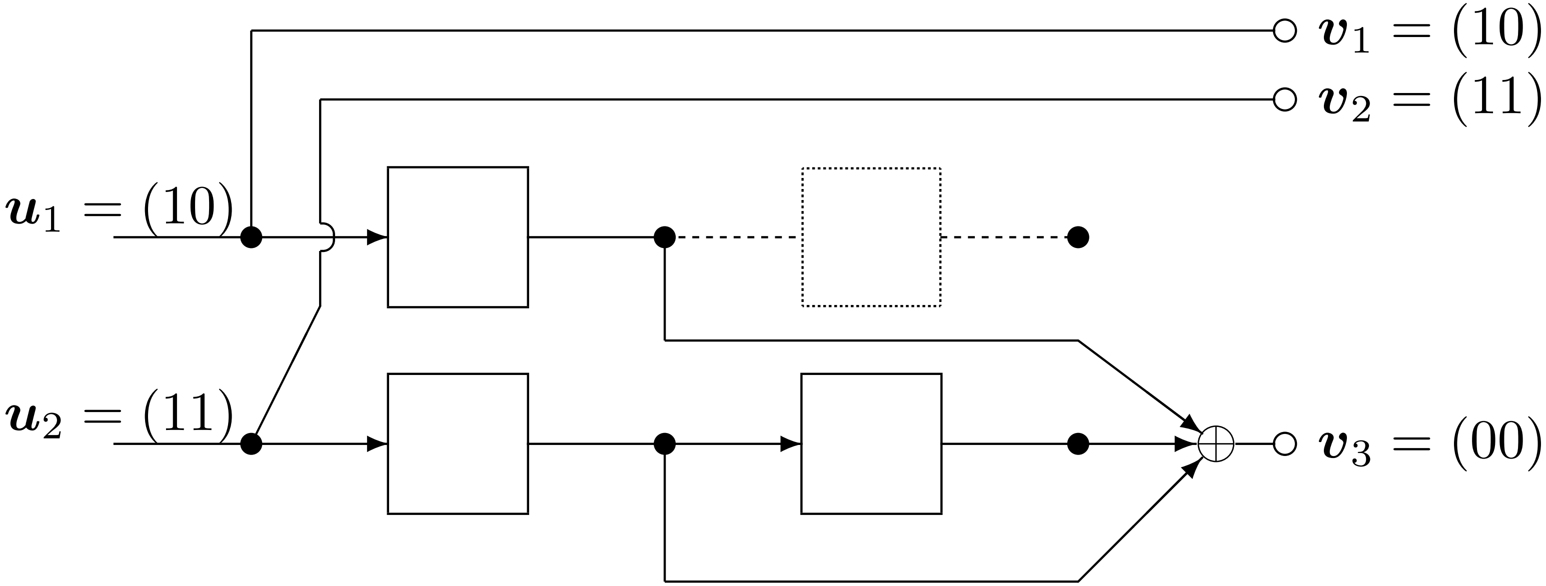
where $\tilde{\boldsymbol{G}}^{c}_{[L]}$ is an $(k \cdot L) \times (n \cdot L)$ matrix.

Generator Matrix of the Tail Biting Binary $(2, 1, 2)$ Convolutional Code



$$\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \tilde{\boldsymbol{G}}^{c}_{[5]} \\
&= (1,1,1,0,1) \cdot \begin{pmatrix} 11 & 10 & 11 & & \\ & 11 & 10 & 11 & \\ & & 11 & 10 & 11 \\ 11 & & & 11 & 10 \\ 10 & 11 & & & 11 \end{pmatrix} \\
&= (01, 10, 10, 01, 00)
\end{aligned}$$

Generator Matrix of the Tail Biting Binary $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$

$\boldsymbol{v} = (110\ 010)$

$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \tilde{\boldsymbol{G}}^{c}_{[2]} \\
&= (11, 01) \cdot \begin{pmatrix} 100 & 001 \\ 010 & 001 \\ 001 & 100 \\ 001 & 010 \end{pmatrix} \\
&= (111, 010)
\end{aligned}
$$

References

[1] M. Bossert, *Channel Coding for Telecommunications*, New York, NY: John Wiley and Sons, 1999.