Turbo Codes

Applications of Turbo Codes

Worldwide Applications & Standards

- Data Storage Systems
- DSL Modem / Optical Communications
- 3G (Third Generation) Mobile Communications
- Digital Video Broadcast (DVB)
- Satellite Communications
- IEEE 802.16 WiMAX
- …

Basic Concept of Turbo Codes

Invented in 1993 by Alian Glavieux, Claude Berrou and Punya Thitimajshima

Basic Structure:

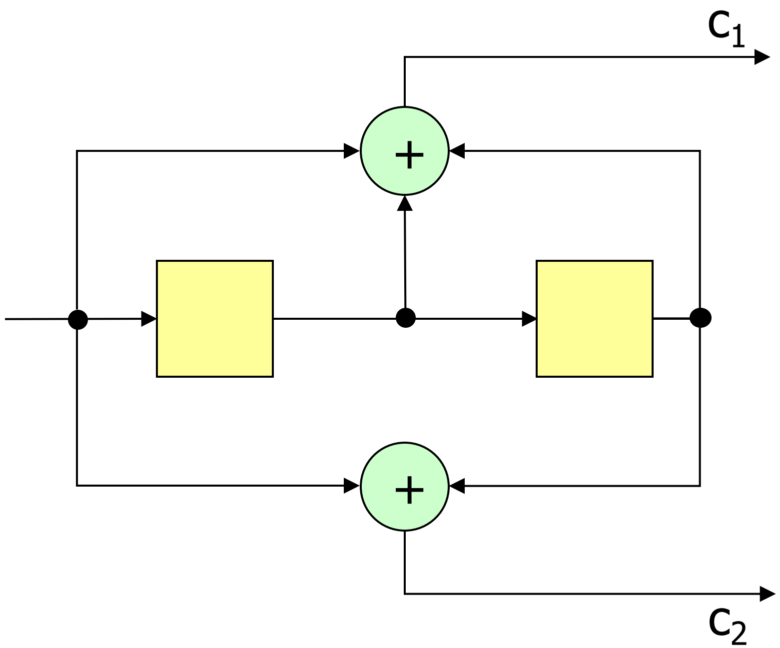
a) Recursive Systematic Convolutional (RSC) Codes

b) Parallel concatenation and Interleaving

c) Iterative decoding

Recursive Systematic Convolutional Codes

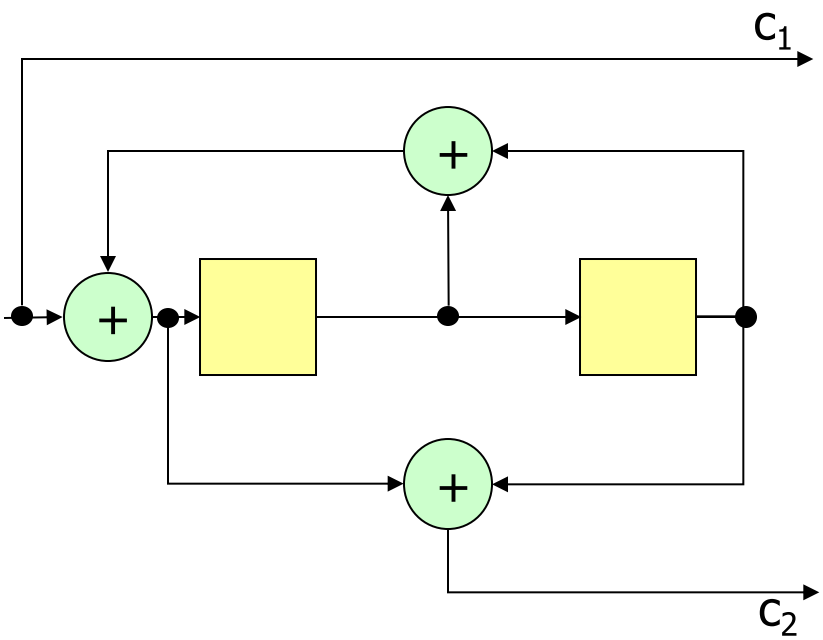
Non Systematic Convolutional Code



$g_1=[1\ 1\ 1]$ $g_2=[1\ 0\ 1]$

$G=[g_1 \quad g_2]$

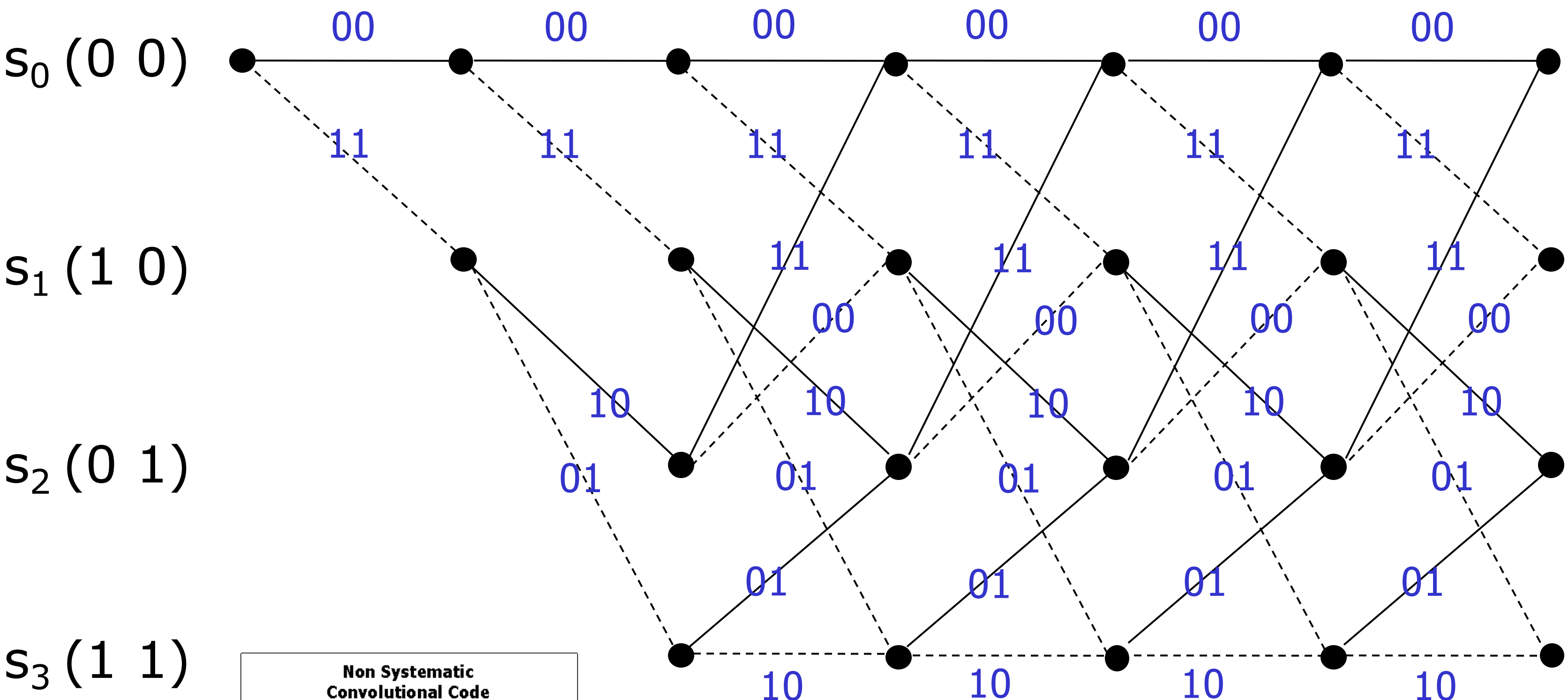
Recursive Systematic Convolutional Code

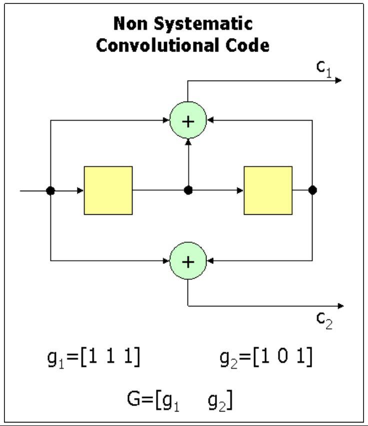


$g_1=[1\ 1\ 1]$ $g_2=[1\ 0\ 1]$

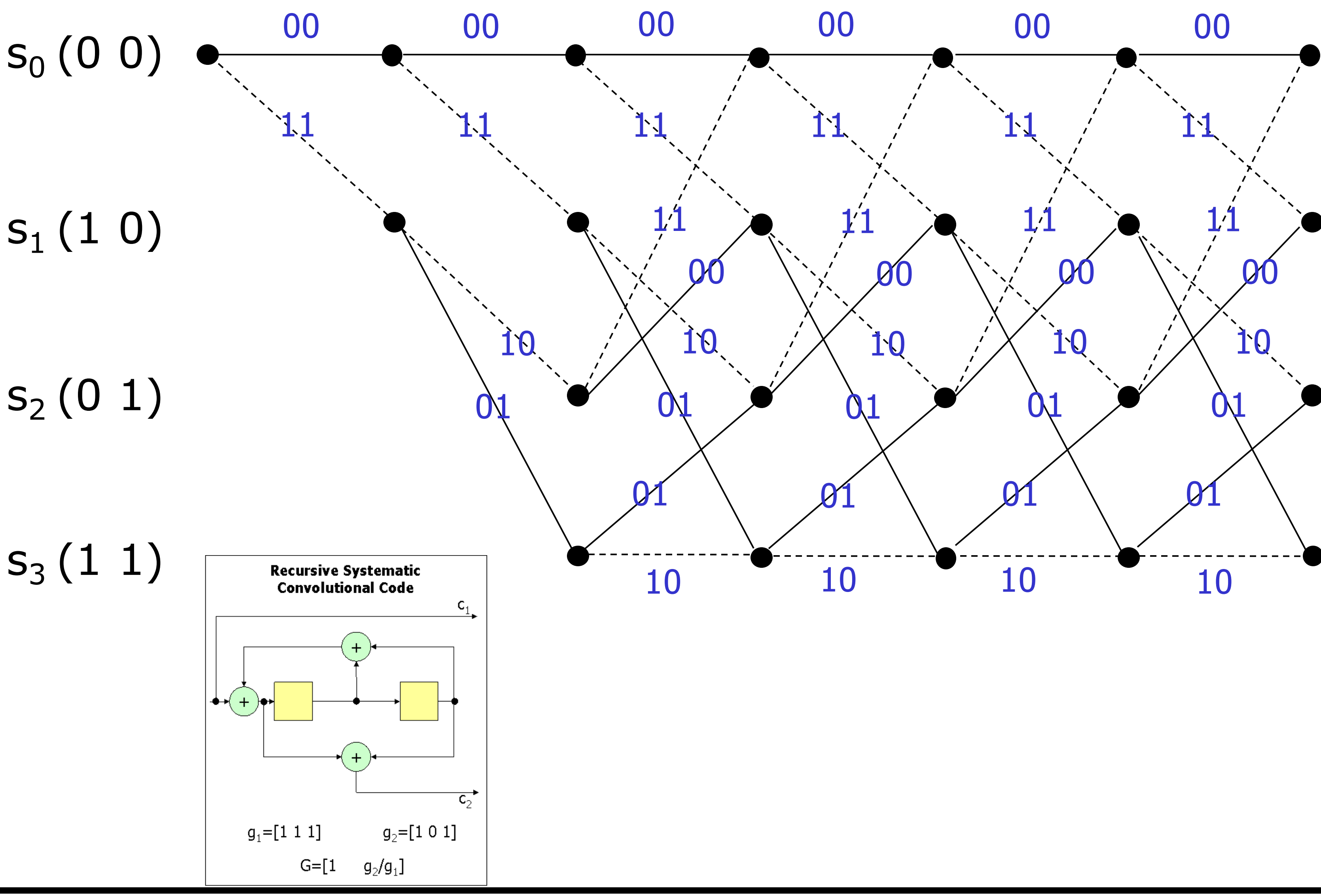
$G=[1 \quad g_2/g_1]$

Non-Recursive Encoder Trellis Diagram



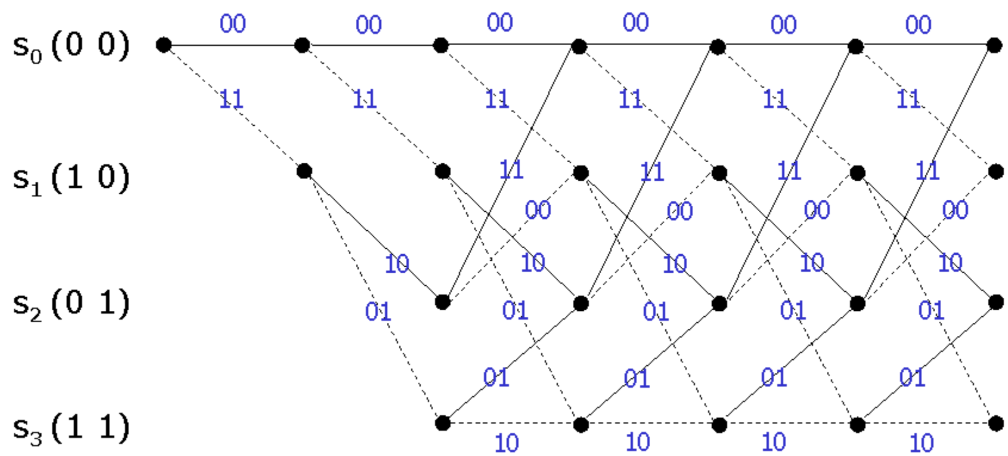


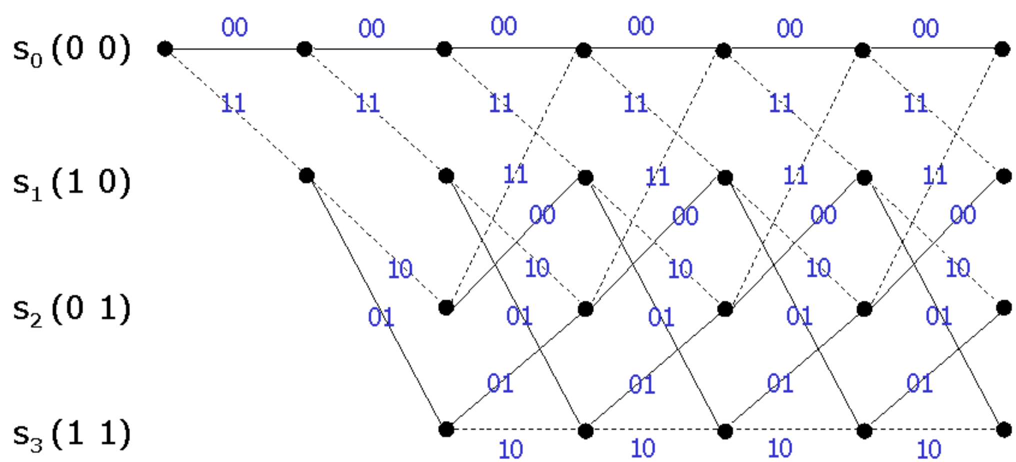
Systematic Recursive Encoder Trellis Diagram



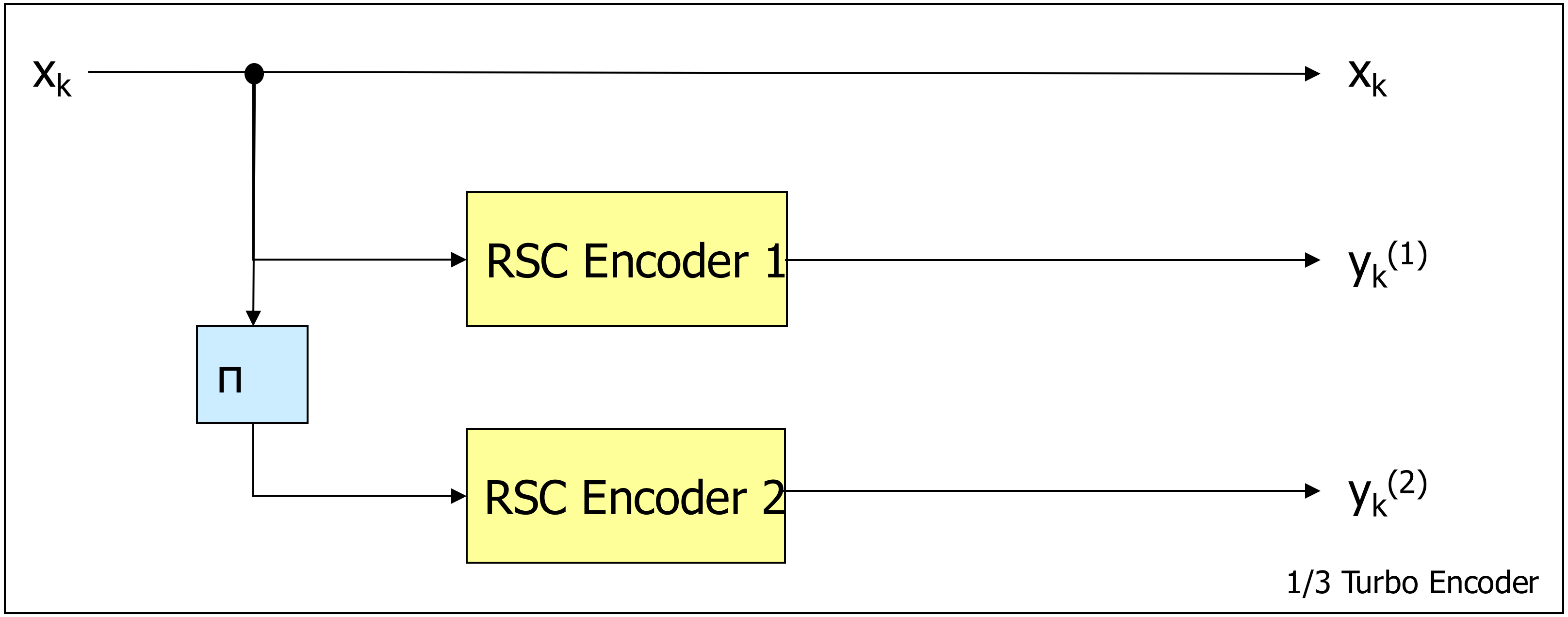
Non-Recursive & Recursive Encoders

- Non Recursive and recursive encoders both have the same trellis diagram structure
- Generally Recursive encoders provide better weight distribution for the code
- The difference between them is in the mapping of information bits to codewords





Parallel Concatenation with Interleaving



- Two component RSC Encoders in parallel separated by an Interleaver
- The job of the interleaver is to de-correlate the encoding process of the two encoders
- Usually the two component encoder are identical

x_k : Systematic information bit

$y_k^{(1)}$: Parity bit from the first RSC encoder

$y_k^{(2)}$: Parity bit from the second RSC encoder

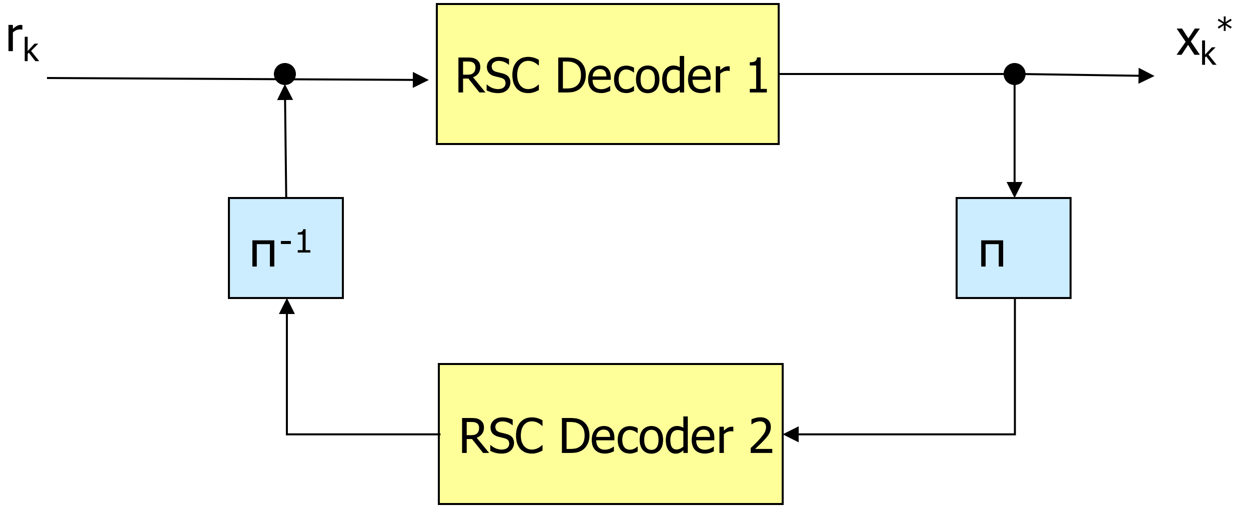
Interleaving

Permutation of data to de-correlate encoding process of both encoders

If encoder 1 generates a low weight codeword then hopefully the interleaved input information would generate a higher weight codeword

By avoiding low weight codewords the BER performance of the turbo codes could improve significantly

Turbo Decoder



Iteratively exchanging information between decoder 1 and decoder 2 before making a final deciding on the transmitted bits

Turbo Codes Performance

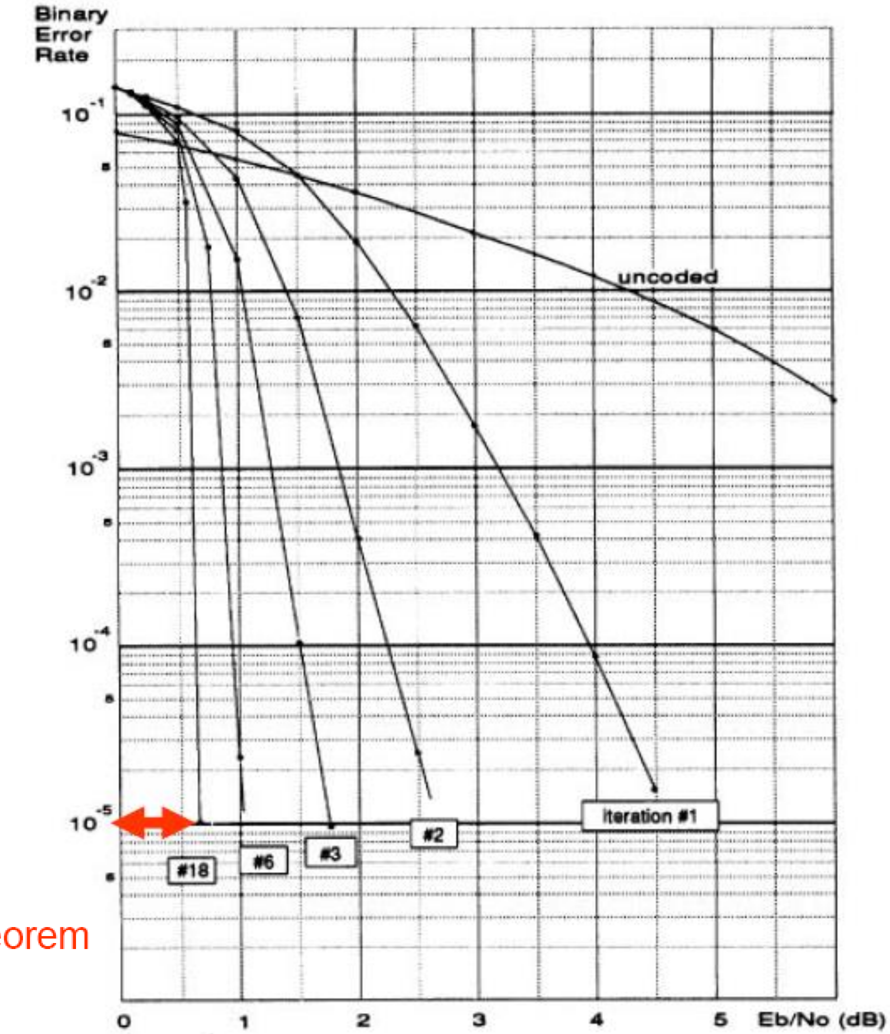
Modulation: *BPSK Signal,*

Turbo Encoder: *G=[37,21], R=1/2,*

Turbo Decoder: *Modified BCJR Algorithm*

Interleave: *Pseudo Random Interleave*

Block size: *65,536 Bit*



0.7 dB from Shannon's limit theorem

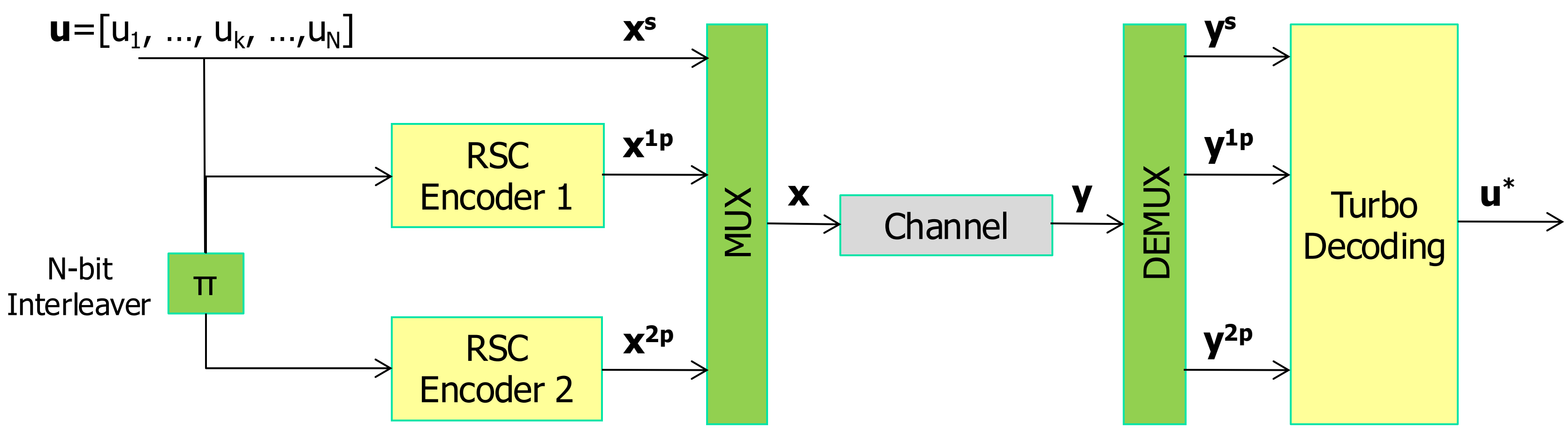
Turbo Decoding

- Turbo decoders rely on probabilistic decoding of component RSC decoders

- Iteratively exchanging ***soft-output*** information between decoder 1 and decoder 2 before making a final deciding on the transmitted bits

- As the number of iteration grows, the decoding performance improves

Turbo Coded System Model



u = $[u_1, \ldots, u_k, \ldots, u_N]$: vector of N information bits

$\mathbf{x}^s$ = $[x_1^s, \ldots, x_k^s, \ldots, x_N^s]$: vector of N systematic bits after turbo-coding of **u**

$\mathbf{x}^{1p}$ = $[x_1^{1p}, \ldots, x_k^{1p}, \ldots, x_N^{1p}]$: vector of N parity bits from first encoder after turbo-coding of **u**

$\mathbf{x}^{2p}$ = $[x_1^{2p}, \ldots, x_k^{2p}, \ldots, x_N^{2p}]$: vector of N parity bits from second encoder after turbo-coding of **u**

x = $[x_1^s, x_1^{1p}, x_1^{2p}, \ldots, x_N^s, x_N^{1p}, x_N^{2p}]$: vector of length 3N of turbo-coded bits for **u**

y = $[y_1^s, y_1^{1p}, y_1^{2p}, \ldots, y_N^s, y_N^{1p}, y_N^{2p}]$: vector of 3N received symbols corresponding to turbo-coded bits of **u**

$\mathbf{y}^s$ = $[y_1^s, \ldots, y_k^s, \ldots, y_N^s]$: vector of N received symbols corresponding to systematic bits in $\mathbf{x}^s$

$\mathbf{y}^{1p}$ = $[y_1^{1p}, \ldots, y_k^{1p}, \ldots, y_N^{1p}]$: vector of N received symbols corresponding to first encoder parity bits in $\mathbf{x}^{1p}$

$\mathbf{y}^{2p}$ = $[y_1^{2p}, \ldots, y_k^{2p}, \ldots, y_N^{2p}]$: vector of N received symbols corresponding to second encoder parity bits in $\mathbf{x}^{2p}$

$\mathbf{u}^*$ = $[u_1^*, \ldots, u_k^*, \ldots, u_N^*]$: vector of N turbo decoder decision bits corresponding to **u**

$$u_k^* = \begin{cases} +1 & \Pr\left[u_k = +1 \middle| \mathbf{y}\right] > \Pr\left[u_k = -1 \middle| \mathbf{y}\right] \\ -1 & \text{otherwise} \end{cases}$$

$\Pr[u_k|\mathbf{y}]$ is known as the ***a posteriori*** probability of k^{th} information bit

Log-Likelihood Ratio (LLR)

The LLR of an information bit u_k is given by:

$$L[u_k] = \ln\left(\frac{\Pr[u_k = +1]}{\Pr[u_k = -1]}\right)$$

Given that $\Pr[u_k=+1]+\Pr[u_k=-1]=1$

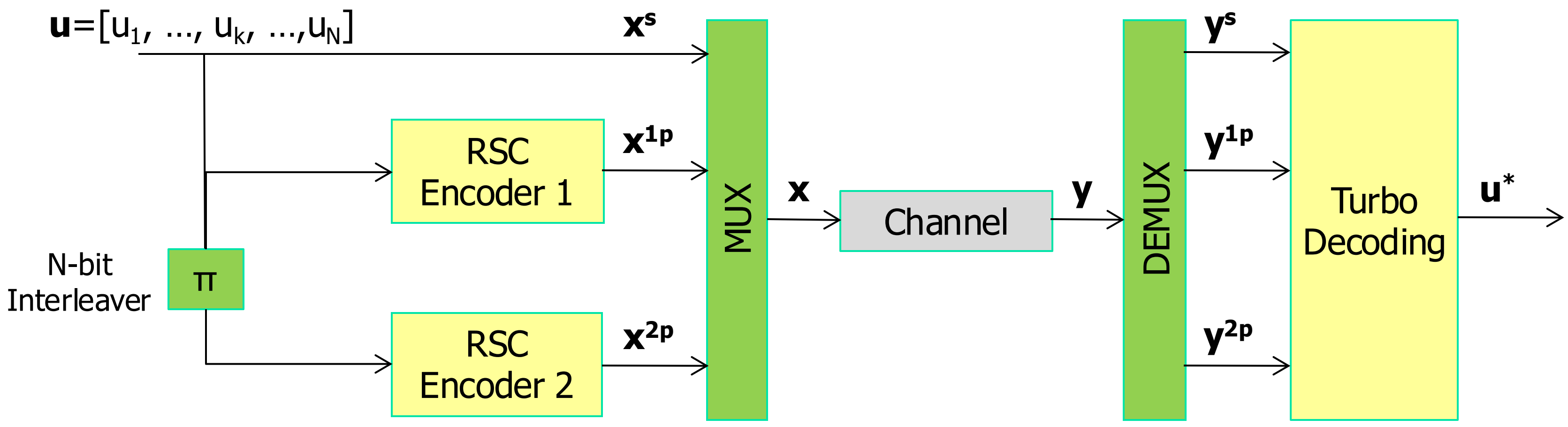
$$\frac{\Pr[u_k = +1]}{1-\Pr[u_k = +1]} = e^{L[u_k]}$$

$$\Pr[u_k = +1] = e^{L[u_k]}\left[1-\Pr[u_k = +1]\right]$$

$$\Pr[u_k = +1] = \frac{e^{L[u_k]}}{1+e^{L[u_k]}} = \frac{1}{1+e^{-L[u_k]}}$$

$$\Rightarrow \Pr[u_k = -1] = \frac{e^{-L[u_k]}}{1+e^{-L[u_k]}}$$

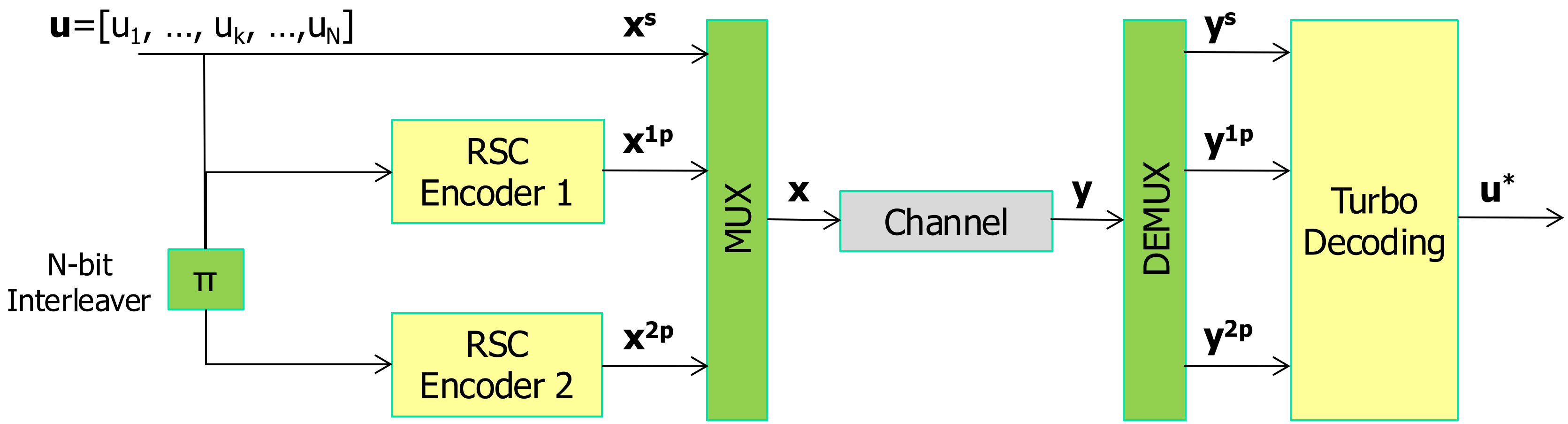
Maximum *A Posteriori* Algorithm



$$L\left[u_k \middle| \mathbf{y}\right] = \ln\left(\frac{\Pr\left[u_k = +1 \middle| \mathbf{y}\right]}{\Pr\left[u_k = -1 \middle| \mathbf{y}\right]}\right)$$

$$u_k{}^* = \operatorname{sign}\left(L\left[u_k \middle| \mathbf{y}\right]\right) = \begin{cases} +1 & L\left[u_k \middle| \mathbf{y}\right] > 0 \\ -1 & \text{otherwise} \end{cases}$$

Some Definitions & Conventions

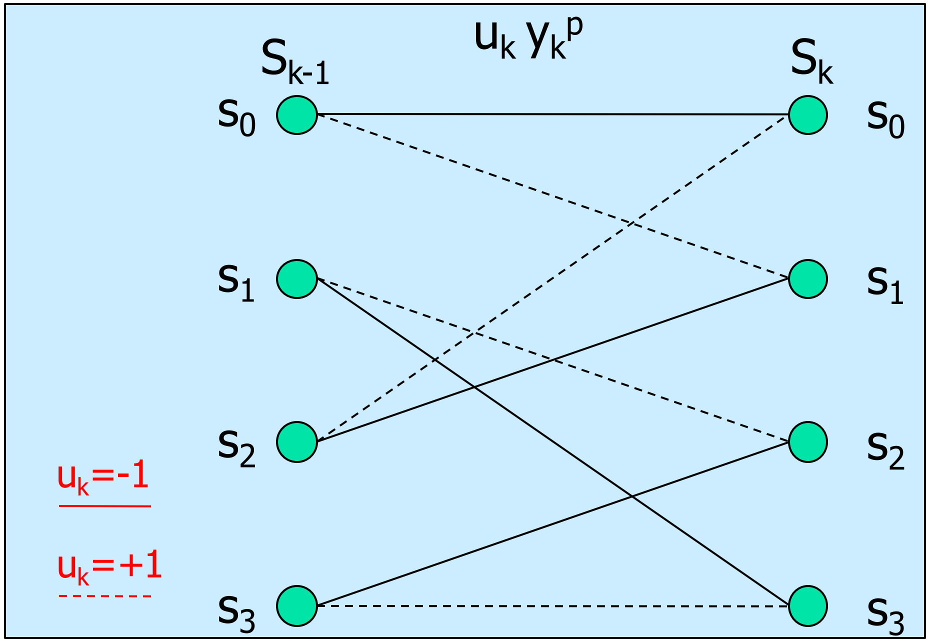


$$\mathbf{y}_{\mathbf{a}}^{\mathbf{b}} = [y_a{}^{s}, y_a{}^{1p}, y_a{}^{2p}, \ldots, y_b{}^{s}, y_b{}^{1p}, y_b{}^{2p}]$$

$$\mathbf{y} = \mathbf{y}_{\mathbf{1}}^{\mathbf{N}}$$

$$(s', s) \equiv S_{k-1} = s', S_k = s$$

$$\Pr\left[S_k = s \middle| S_{k-1} = s'\right] = \Pr\left[u_k\right]_{(s',s)}$$

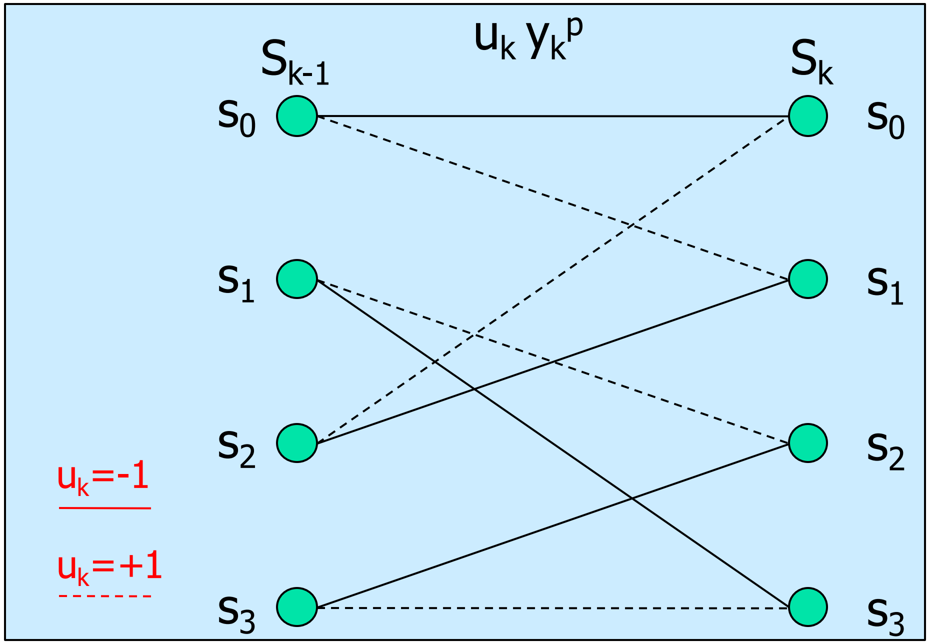


Derivation of LLR

Define $S^{(i)}$ as the set of pairs of states (s',s) such that the transition from $S_{k-1}=s'$ to $S_k=s$ is caused by the input $u_k=i$, $i=0,1$, i.e.,

$S^{(0)} = \{(s_0, s_0), (s_1, s_3), (s_2, s_1), (s_3, s_2)\}$

$S^{(1)} = \{(s_0, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_3)\}$



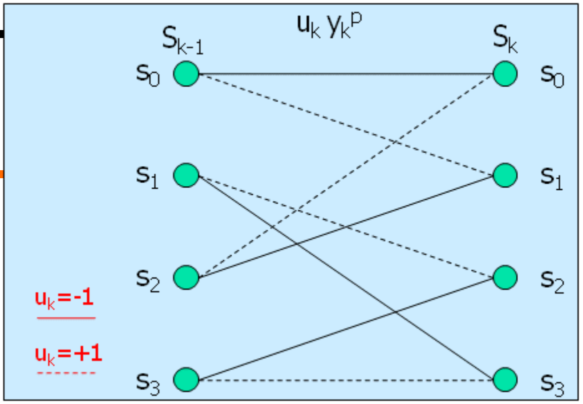
$$\Pr\left[u_k = i \middle| \mathbf{y}_1^N\right] = \sum_{S^{(i)}} \Pr\left[S_{k-1} = s', S_k = s \middle| \mathbf{y}_1^N\right]$$

$$\Pr\left[u_k = i \middle| \mathbf{y}_1^N\right] = \frac{\sum_{S^{(i)}} \Pr\left[S_{k-1} = s', S_k = s, \mathbf{y}_1^N\right]}{\Pr\left[\mathbf{y}_1^N\right]}$$

Remember Bayes' Rule

$$\Pr\left[B \middle| A\right] = \frac{\Pr\left[A,B\right]}{\Pr\left[A\right]}$$

Derivation of LLR



$$\Pr\left[u_k = i \middle| \mathbf{y}_1^N\right] = \frac{\sum_{S^{(i)}} \Pr\left[S_{k-1} = s', S_k = s, \mathbf{y}_1^N\right]}{\Pr\left[\mathbf{y}_1^N\right]}$$

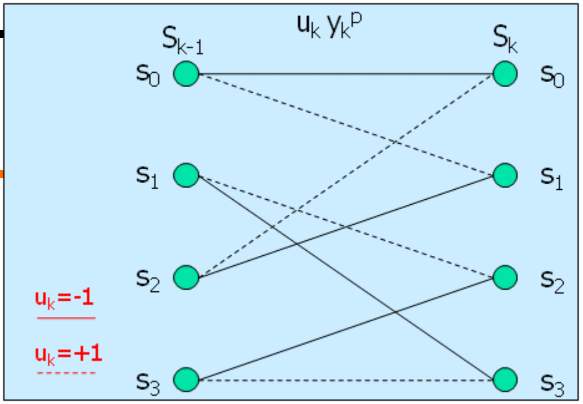
Define

$$\sigma_k(s', s) = \Pr\left[S_{k-1} = s', S_k = s, \mathbf{y}_1^N\right]$$

$$L\left[u_k \middle| \mathbf{y}_1^N\right] = \ln\left(\frac{\Pr\left[u_k = +1 \middle| \mathbf{y}_1^N\right]}{\Pr\left[u_k = -1 \middle| \mathbf{y}_1^N\right]}\right)$$

$$L\left[u_k \middle| \mathbf{y}_1^N\right] = \ln\left(\frac{\sum_{S^{(1)}} \sigma_k(s', s) \Big/ \Pr\left[\mathbf{y}_1^N\right]}{\sum_{S^{(0)}} \sigma_k(s', s) \Big/ \Pr\left[\mathbf{y}_1^N\right]}\right) = \ln\left(\frac{\sum_{S^{(1)}} \sigma_k(s', s)}{\sum_{S^{(0)}} \sigma_k(s', s)}\right)$$

Derivation of LLR



$$\sigma_k(s',s) = \Pr\left[S_{k-1}=s', S_k=s, \mathbf{y}_1^N\right]$$

$$\sigma_k(s',s) = \Pr\left[S_{k-1}=s', S_k=s, \mathbf{y}_1^{k-1}, y_k, \mathbf{y}_{k+1}^N\right]$$

$$\sigma_k(s',s) = \Pr\left[\mathbf{y}_{k+1}^N \middle| S_{k-1}=s', S_k=s, \mathbf{y}_1^{k-1}, y_k\right]\Pr\left[S_{k-1}=s', S_k=s, \mathbf{y}_1^{k-1}, y_k\right]$$

$\mathbf{y}_{k+1}^N$ depends only on S_k and is independent on S_{k-1}, y_k and $\mathbf{y}_1^{k-1}$

$$\Rightarrow \sigma_k(s',s) = \Pr\left[S_{k-1}=s', S_k=s, \mathbf{y}_1^{k-1}, y_k\right]\Pr\left[\mathbf{y}_{k+1}^N \middle| S_k=s\right]$$

$$\sigma_k(s',s) = \Pr\left[S_{k-1}=s', \mathbf{y}_1^{k-1}\right]\Pr\left[S_k=s, y_k \middle| S_{k-1}=s', \mathbf{y}_1^{k-1}\right]\Pr\left[\mathbf{y}_{k+1}^N \middle| S_k=s\right]$$

S_k , y_k depends only on S_{k-1} and is independent on $\mathbf{y}_1^{k-1}$

$$\sigma_k(s',s) = \Pr\left[S_{k-1}=s', \mathbf{y}_1^{k-1}\right]\Pr\left[S_k=s, y_k \middle| S_{k-1}=s'\right]\Pr\left[\mathbf{y}_{k+1}^N \middle| S_k=s\right]$$

Derivation of LLR

$$\sigma_k(s',s)=\Pr\left[S_{k-1}=s',\mathbf{y}_1^{k-1}\right]\Pr\left[S_k=s,y_k \middle| S_{k-1}=s'\right]\Pr\left[\mathbf{y}_{k+1}^{N} \middle| S_k=s\right]$$

$$\alpha_k(s)=\Pr\left[S_k=s,\mathbf{y}_1^{k}\right]$$

$$\beta_k(s)=\Pr\left[\mathbf{y}_{k+1}^{N} \middle| S_k=s\right]$$

$$\gamma_k(s',s)=\Pr\left[S_k=s,y_k \middle| S_{k-1}=s'\right]$$

$$\sigma_k(s',s)=\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)$$

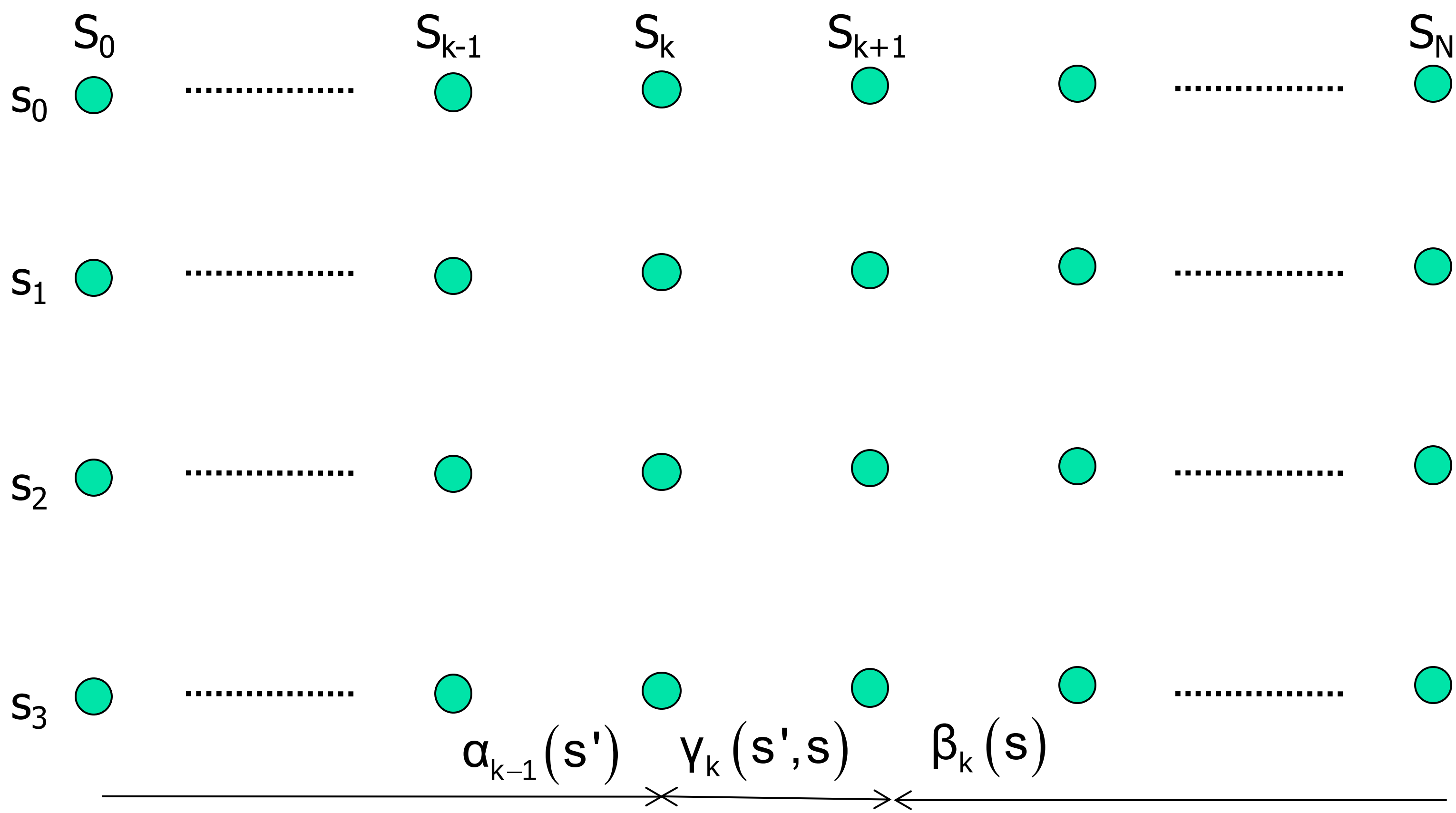
$$L\left[u_k \middle| \mathbf{y}_1^{N}\right]=\ln\left(\frac{\sum_{S^{(1)}}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum_{S^{(0)}}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right)$$

Derivation of LLR

$$\alpha_k(s) = \Pr\left[S_k = s, \mathbf{y}_1^k\right]$$

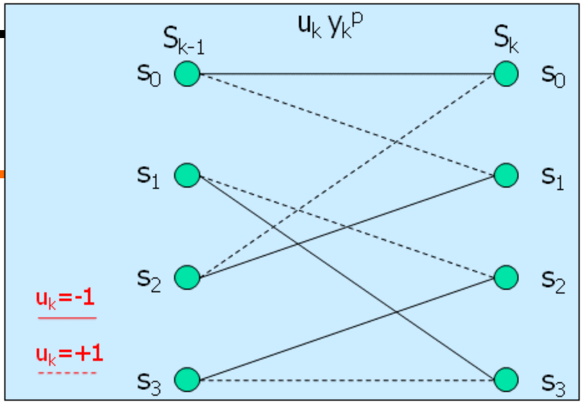
$$\beta_k(s) = \Pr\left[\mathbf{y}_{k+1}^N \middle| S_k = s\right]$$

$$\gamma_k(s',s) = \Pr\left[S_k = s, y_k \middle| S_{k-1} = s'\right]$$



y= y_1 ……………… y_{k-1} y_k y_{k+1} ……………… y_N

Computation of $\alpha_k(s)$



$$\alpha_k(s) = \Pr\left[S_k = s, \mathbf{y}_1^k\right] = \sum_{\text{all } s'} \Pr\left[S_{k-1} = s', S_k = s, \mathbf{y}_1^k\right]$$

Example

$$\alpha_k(s_0) = \Pr\left[S_k = s_0, \mathbf{y}_1^k\right] = \sum_{\text{all } s'} \Pr\left[S_{k-1} = s', S_k = s_0, \mathbf{y}_1^k\right]$$

$$\alpha_k(s_0) = \Pr\left[S_k = s_0, \mathbf{y}_1^k\right] = \Pr\left[S_{k-1} = s_0, S_k = s_0, \mathbf{y}_1^k\right] + \Pr\left[S_{k-1} = s_2, S_k = s_0, \mathbf{y}_1^k\right]$$

$$\alpha_k(s) = \sum_{\text{all } s'} \Pr\left[S_{k-1} = s', \mathbf{y}_1^{\mathbf{k-1}}\right] \Pr\left[S_k = s, y_k \middle| S_{k-1} = s', \mathbf{y}_1^{\mathbf{k-1}}\right]$$

$$\alpha_k(s) = \sum_{\text{all } s'} \Pr\left[S_{k-1} = s', \mathbf{y}_1^{\mathbf{k-1}}\right] \Pr\left[S_k = s, y_k \middle| S_{k-1} = s'\right]$$

$$\alpha_k(s) = \sum_{\text{all } s'} \alpha_{k-1}(s') \gamma_k(s', s)$$

Computation of $\alpha_k(s)$

$$\alpha_k(s) = \sum_{\text{all } s'} \alpha_{k-1}(s')\gamma_k(s',s)$$

Forward Recursive Equation

Given the values of $\gamma_k(s',s)$ for all index k, the probability $\alpha_k(s)$ can be forward recursively computed. The initial condition $\alpha_0(s)$ depends on the initial state of the convolutional encoder

$$\alpha_0(s) = \Pr\left[S_0 = s, \mathbf{y}_1^0\right] = \Pr\left[S_0 = s\right]$$

The encoder usually starts at state 0

$$\Rightarrow \alpha_0(s_0) = 1$$

$$\alpha_0(s) = 0 \quad \forall s \neq s_0$$

Computation of $\beta_k(s)$

$$\beta_k(s) = \Pr\left[\mathbf{y}_{\mathbf{k+1}}^{\mathbf{N}} \middle| S_k = s\right]$$

$$\beta_k(s) = \sum_{s''} \Pr\left[S_{k+1} = s'', y_{k+1} \middle| S_k = s\right] \Pr\left[\mathbf{y}_{\mathbf{k+2}}^{\mathbf{N}} \middle| S_k = s, S_{k+1} = s'', y_{k+1}\right]$$

$$\beta_k(s) = \sum_{s''} \Pr\left[\mathbf{y}_{\mathbf{k+2}}^{\mathbf{N}} \middle| S_{k+1} = s''\right] \Pr\left[S_{k+1} = s'', y_{k+1} \middle| S_k = s\right]$$

$$\beta_k(s) = \sum_{s''} \beta_{k+1}(s'')\gamma_{k+1}(s, s'')$$

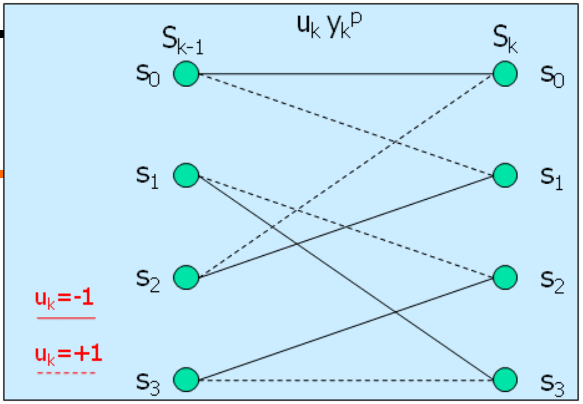
Backward Recursive Equation

Given the values of $\gamma_k(s',s)$ for all index k, the probability $\beta_k(s)$ can be backward recursively computed. The initial condition $\beta_N(s)$ depends on the final state of the trellis

First encoder usually finishes at state s_0 $\Rightarrow \beta_N(s_0) = 1, \ \beta_N(s) = 0 \quad \forall s \neq 0$

Second encoder usually has open trellis $\Rightarrow \beta_N(s) = 1 \quad \forall s$

Computation of $\gamma_k(s',s)$



$$\gamma_k(s',s) = \Pr\left[S_k = s, y_k \middle| S_{k-1} = s'\right]$$

$$\gamma_k(s',s) = \Pr\left[y_k \middle| S_{k-1} = s', S_k = s\right]\Pr\left[S_k = s \middle| S_{k-1} = s'\right]$$

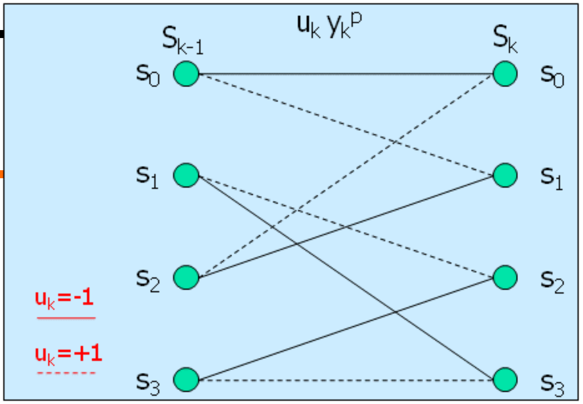
$$\gamma_k(s',s) = \Pr\left[y_k \middle| S_{k-1} = s', S_k = s\right]\Pr^a\left[u_k\right]$$

$$\gamma_k(s',s) = \Pr\left[y_k \middle| x_k\right]\Pr^a\left[u_k\right]$$

Note that $y_k = \left[y_k{}^s \quad y_k{}^p\right], x_k = \left[x_k{}^s \quad x_k{}^p\right]$

$$\Pr\left[y_k \middle| x_k\right] = \Pr\left[y_k{}^s \middle| x_k{}^s\right]\Pr\left[y_k{}^p \middle| x_k{}^p\right]$$

Computation of $\gamma_k(s',s)$



$$\Pr\left[y_k \mid x_k\right] = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(y_k^s - x_k^s\right)^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(y_k^p - x_k^p\right)^2}{2\sigma^2}}$$

$$\Pr\left[y_k \mid x_k\right] = \frac{1}{2\pi\sigma^2} e^{-\frac{\left(y_k^s - x_k^s\right)^2}{2\sigma^2}} \times e^{-\frac{\left(y_k^p - x_k^p\right)^2}{2\sigma^2}}$$

$$L^a\left[u_k\right] = \ln\left(\frac{\Pr\left[u_k = +1\right]}{\Pr\left[u_k = -1\right]}\right)$$

$$\Pr{}^a\left[u_k = +1\right] = \frac{1}{1 + e^{-L^a\left[u_k\right]}}, \Pr{}^a\left[u_k = -1\right] = \frac{e^{-L^a\left[u_k\right]}}{1 + e^{-L^a\left[u_k\right]}}$$

$$\Rightarrow \Pr{}^a\left[u_k\right] = \left(\frac{e^{-L^a\left[u_k\right]/2}}{1 + e^{-L^a\left[u_k\right]}}\right) e^{u_k L^a\left[u_k\right]/2}$$

$$\gamma_k\left(s', s\right) = \left(\frac{1}{2\pi\sigma^2} e^{-\frac{\left(y_k^s - x_k^s\right)^2}{2\sigma^2}} \times e^{-\frac{\left(y_k^p - x_k^p\right)^2}{2\sigma^2}}\right)\left(\frac{e^{-L^a\left[u_k\right]/2}}{1 + e^{-L^a\left[u_k\right]}}\right) e^{u_k L^a\left[u_k\right]/2}$$