X Lecture 12, June 9, 2011

## Cyclic Codes

A code is called a cyclic code if the cyclic shift of each codeword is another codeword. That is if

$$c_0, c_1 \cdots c_{n-1}$$

is a codeword $c_{n-1}, c_0, c_1, \cdots, c_{n-2}$ is also a codeword.

The advantage of cyclic codes is that they can be treated algebraically. In specific terms they can be represented as algebraic polynomials and generate by polynomial multiplication.

A codeword $c_0, c_1, \cdots c_{n-1}$ is represented as

$$C(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$$

Note that with X substituted by 2, we have the decimal representation of a binary vector.

For cyclic codes the codeword polynomial $c(x)$ is equal to $c(x) = a(x)\,g(x)$ where,

$$g(x) = g_0 + g_1 x + \cdots + g_{n-k} x^{n-k}$$

is a polynomial of degree $n-k$ called the generator polynomial that divide $x^n + 1$.

Using the generator polynomial to find the codewords (encoding)

Note that

$$\underline{c} = (c_0, c_1, \cdots, c_{n-1}) = (b_0, b_1, \cdots, b_{n-k+1}, m_0, \cdots m_{k-1})$$

or

$$c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} = b_0 + b_1 x + \cdots b_{n-k+1} x^{n-k+1} + m_0 x^{n-k} + \cdots + m_{k-1} x^{n-1}$$

or

$$c(x) = b(x) + x^{n-k} m(x)$$

where $m(x) = m_0 + m_1 x + \cdots + m_{k-1} x^{k-1}$ is the message polynomial and $b(x) = b_0 + b_1 x + \cdots + b_{n-k+1} x^{n-k+1}$ is the parity polynomial.

let $c(x) = a(x)\,g(x)$ then,

$$x^{n-k} m(x) = a(x)\,g(x) + b(x)$$

That is, $b(x)$ is the <u>remainder</u> resulting from dividing $x^{n-k} m(x)$ by $g(x)$.

So, the encoding procedure is as follows:

1) Shift $m(x)$ by $n-k$ bits, i.e., multiply it by $x^{n-k}$.

2) Divide $x^{n-k} m(x)$ by $g(x)$, the remainder $b(x)$ is the parity polynomial.

3) Add $b(x)$ to $x^{n-k} m(x)$ to form $c(x)$.

Example: The generator polynomial for (7,4) Hamming Code is $g(x) = x^3 + x + 1$.

Find the codeword for the message $\underline{m} = [0\,1\,0\,1]$

$$m(x) = x^3 + x$$

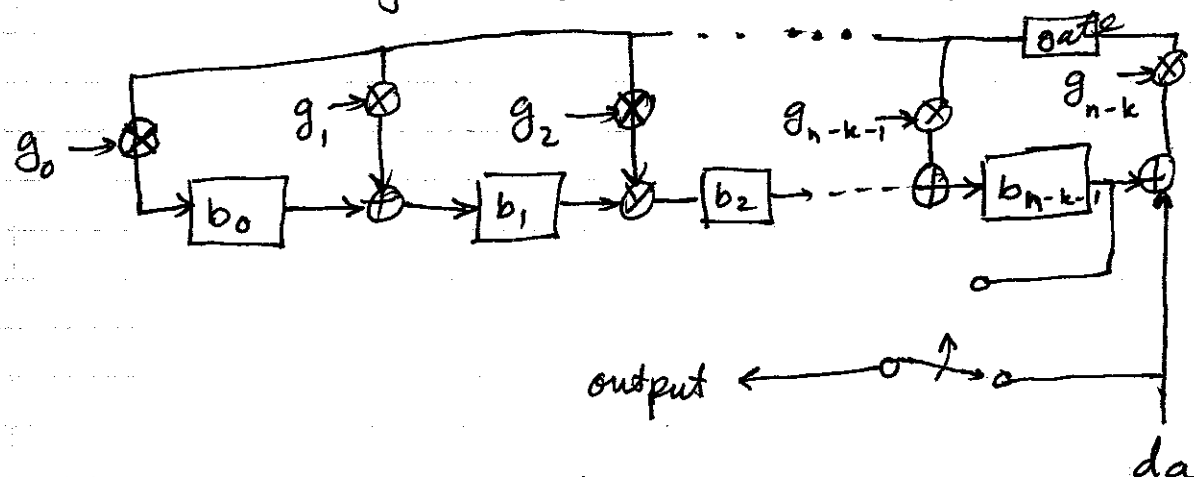
$$x^{n-k} m(x) = x^{7-4}(x^3 + x) = x^3(x^3 + x) = x^6 + x^4$$

$$x^6 + x^4 \quad | \; x^3 + x + 1$$
$$x^6 + x^4 + x^3 \quad x^3 + 1$$
$$x^3$$
$$x^3 + x + 1$$
$$x + 1$$

So, $c(x) = b^{n-k} m(x) + b(x) = x^6 + x^4 + x + 1$

$\Rightarrow \quad c = [1\ 1\ 0\ 0\ 1\ 0\ 1]$

Circuit Diagram of the encoder:

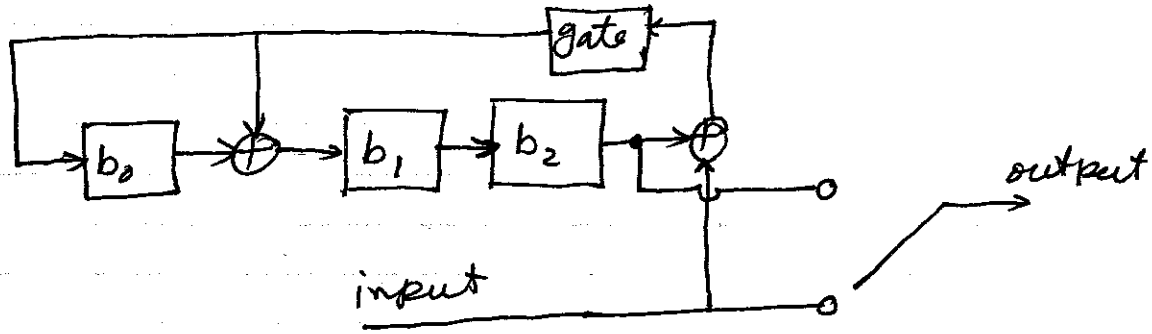



1) Data is shifted to the register.

- It is both shifted n-k places and divided by g(x), after entering all k data bits the parities $b_0, b_1, \ldots b_{n-k-1}$ are formed. Switch is in lower position so message bits are transmitted.

Now, the gate is opened so that the parities do not change and switch is flipped up so that the parities are transmitted.

For the (7,4) Hamming code the encoder is:




Other cyclic codes:

Hamming codes in general:

Block Length $n = 2^m - 1$

Number of message bits: $k = 2^m - m - 1$

Number of parity bits: $n - k = m$

We saw the example of (7,4) code corresponding to $m = 3 \Rightarrow n = 2^3 - 1 = 7$ and $k = 2^3 - 3 - 1 = 4$.

For $m = 4 \Rightarrow n = 15$ and $k = 11 \Rightarrow (15, 11)$ code.

Hamming codes all have $d_{min} = 3$ so correct

one bit of error.

BCH Codes have length $n = 2^m - 1$, but the information bits and parity bits, k and n−k, respectively can be chosen so that the error correcting capability be varied.

Reed-Solomon (RS) Codes:

These codes are non-binary. They consist of $n = 2^m - 1$ symbol and each symbol is m bits long. The number of information symbols k can be varied. The maximum number of symbols that an (n, k) RS Code can correct is $\lfloor \frac{n-k}{2} \rfloor$, i.e.,

$$t \leq \lfloor \frac{n-k}{2} \rfloor$$

As an example take $m = 8$. This means that each symbol is 8 bits (a byte). The total length of a codeword is $n = 2^m - 1 = 255$ So the Code is (255, k) where k can be selected anywhere from 1 to 255 to

make the code stronger or weaker.

Say, if we let $k=239$, we have a (255, 239) code. This code takes 239 bytes (8 bit symbols), i.e., $239 \times 8 = 1912$ and adds $255-239=16$ parity bits and create a 255 bytes (2040 bits) codeword. This code can correct $\frac{255-239}{2} = 8$ bytes of error. The good thing about RS code is that it corrects a symbol (several bits) so, it is suitable for burst errors (errors occuring consecutively). For example, the above code (255, 239) can correct any consecutive 7 bytes, i.e., any burst of upto $7 \times 8 = 56$ bits

## Performance of Coded Systems

When we use error control coding schemes we hope that we get lower bit error rate than uncoded system. The difference between a coded system and an uncoded version at a given $\frac{E_b}{N_0}$ is called the coding gain.

To clarify the idea consider a simple example using BPSK and (7,4) Hamming Code.

Example: Find the performance of a system using BPSK modulation and (7,4) code at $\frac{E_b}{N_0} = 5dB$. Compare with uncoded case.

Solution: $\frac{E_b}{N_0} = 5dB \Rightarrow \frac{E_b}{N_0} = 10^{0.5} = 3.16$

$$P_{unc}(e) = Q\left(\sqrt{2\frac{E_b}{N_0}}\right) = Q\left(\sqrt{2\times 3.16}\right) \simeq 6\times 10^{-3}$$

For coded case for each 4 bits we send 7 bits so energy is reduced by $\frac{4}{7}$. So

$$\frac{E_c}{N_0} = \frac{4}{7}\times 3.16 = 1.807$$

So, the error probability before decoding (before correcting errors) is:

$$p = Q\left(\sqrt{2\frac{E_c}{N_0}}\right) = Q(1.9) = 0.0287$$

The code can correct 1 error in a codeword (in 7 bits). So, the probability of error after decoding is the probability that more than one error exists in 7 bits.

$$P_{CW}(e) = \sum_{i=2}^{7} \binom{7}{i} p^i (1-p)^{7-i}$$

$$= 1 - \left(\sum_{i=0}^{1} \binom{7}{i} p^i (1-p)^{7-i}\right)$$

$$= 1 - (1-p)^7 - 7p(1-p)^6 = 0.0157$$

This is the probability that a codeword is decoded erroneously. Assume that, on the average, the probability that a bit is in error is half this, i.e., one the average half the bits are in error the

$$P_b = 0.00785$$

which is almost the same as uncoded case.

Even when $\frac{E_b}{N_0} = 10\ dB \rightarrow \frac{E_b}{N_0} = 10$

$$P_{unc} = Q(\sqrt{20}) = Q(4.47) = 4 \times 10^{-6}$$

and with coding $\frac{E_c}{N_0} = 10 \times \frac{4}{7} = 5.7$

and $$P = Q(\sqrt{2 \times 5.7}) = 3.4 \times 10^{-4}$$

and

$$P_B = \frac{1}{2}\left[1 - p^7 - (1-p)^6 p\right] = 1.2 \times 10^{-6}$$

which is only about one third of uncoded case. The following figure shows the performance of Hamming (7,4) code and some other codes.

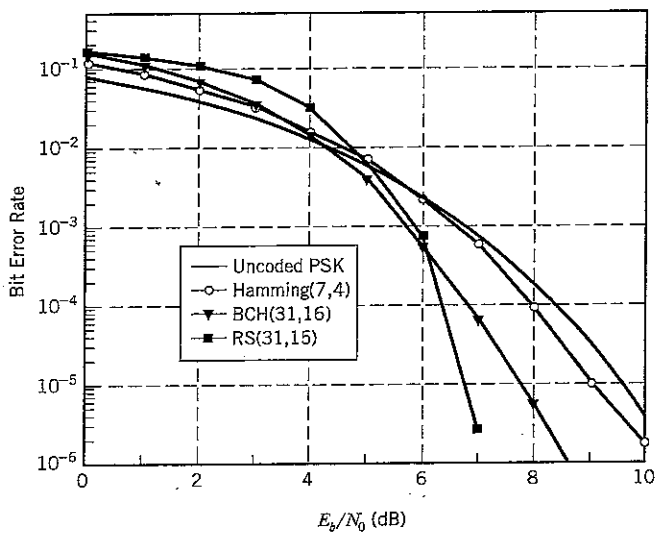



**FIGURE 10.25** Simulated BER performance of uncoded and FEC-encoded PSK transmitted over a Gaussian channel.

Now consider using Hamming (15, 11) code. This code corrects 1 bit of error in 15 bits.

This may be a weaker code, i.e., correcting one in 15 instead of 1 in 7 errors, but it has less parity and the $E_b$ is only reduced by $\frac{11}{15}$ instead of $\frac{4}{7}$. So,

$$\frac{E_c}{N_0} = 10 \times \frac{11}{15} = 7.33$$

$$P = Q(\sqrt{2 \times 7.33}) = Q(3.83) = 7 \times 10^{-5}$$

and the bit error probability after decoding is:

$$P_B = \frac{1}{2}\left[1 - P^{15} - P(1-P)^{14}\right] = 2.57 \times 10^{-7}$$

which is more than one order of magnitude better than the uncoded bit error rate (BER) of $4 \times 10^{-6}$.

Shortened block codes:

Sometimes, in a system, we may want to have smaller code length in order to reduce latency or to adapt to data at the input of the encoder. In such case an (n, k) code can be shortened by putting L zero's

at the end of k-L information bits (or symbols). After adding n-k parities, we have an (n, k) code whose last L bits (or symbols) are zero. Since the transmitter and receiver know that these bits are zero, it is not needed to transmit them. So, we have a code with length n-L with k-L information symbols.

For example in Digital Video Broadcasting (DVB) Standard, each packet is 188 bytes (MPEG packets). Only 16 bytes of parity is needed in most situations. So, a (255, 239) Reed-Solomon Code is selected. However, since the information length is 188 and not 239, we null 239-188 = 51 bytes and we get (204, 188) Shortened Code.

## Convolutional Codes

In the case of block codes, a usually long block of data is fed to the encoder and a block of encoded data consisting of the message (the input) and the parities is output.

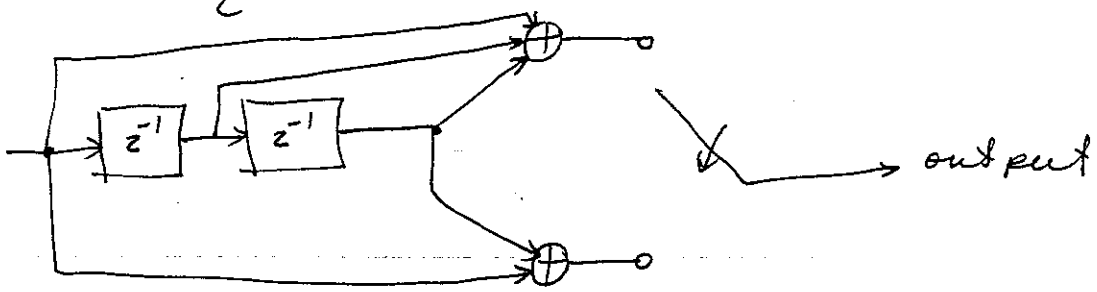
In the case of Convolutional codes a few (most often one) bit is fed to the encoder and n bits are generated based on the input bit (s) and a few previous bits. Assume that the system has k bits input an n bits output and M memory cells to store the previous inputs.

Assume that a block of L symbols, i.e., kL bits is fed to the encoder. For each k-bit input, we have an n-bit output i.e., nL output bits, We need also flush the memory by feeding M null (zero) symbols. So the rate of the code will be

$$\frac{kL}{n(L+M)} \cong \frac{k}{n}$$

Example

A rate $\frac{1}{2}$ Code

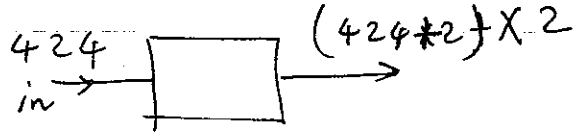



if the initial state of shift registers is 0 0 then the input sequence 1 0 0 1 produces:

1 1, 1 0, 1 1, 1 1, 1 0, 1 1,
Tail

actual rate here is $= \frac{4}{12} = \frac{1}{3}$ not $\frac{1}{2}$

but usually more bits enter encoder ~~at~~ at a given time interval. Say, if you have ATM cell (53 bytes = 424 bits) then




$$\text{rate} = \frac{424}{(424+2)\times 2} \approx \frac{1}{2} \quad (0.4976)$$

12-15

## generator polynomial

$\underline{g}_1 = (1\ 1\ 1)$

$\underline{g}_2 = (1\ 0\ 1)$

/ we can also show this as

$g_1(D) = 1 + D + D^2$

$g_2(D) = 1 + D^2$

two vectors (in general one vector for each input-output pair)

in location $l$ of $m$th vector a 1 represents connection between the output of the shift register $l$ to $m$ th output.

Constraint length $K$ = the number of shift register bits (2+1 latch)

number of state $S = 2^{K-1}$, e.g., for $K = 3$, $S = 4$

---

Using generator polynomial we can generate output by finding

$$c_1(D) = m(D)\, g_1(D)$$

$$c_2(D) = m(D)\, g_2(D)$$

let input be $\underline{m} = (1\ 1\ 0\ 0\ 1) \Rightarrow m(D) = D^4 + D^3 + 1$

then $c_1(D) = (D^4 + D^3 + 1)(D^2 + D + 1) = D^6 + D^3 + D^2 + D + 1$

or $\underline{c}_1 = (1\ D\ D\ 1\ 1\ 1\ 1)$

and $c_2(D) = (D^4 + D^3 + 1)(D^2 + 1) = D^6 + D^5 + D^4 + D^3 + D^2 + 1$

or $\underline{c}_2 = (1\ 1\ 1\ 1\ 1\ 1\ 0\ 1)$

So the output will be (after multiplexing):

11, 10, 10, 11, 11, 01, 11

A typical example:

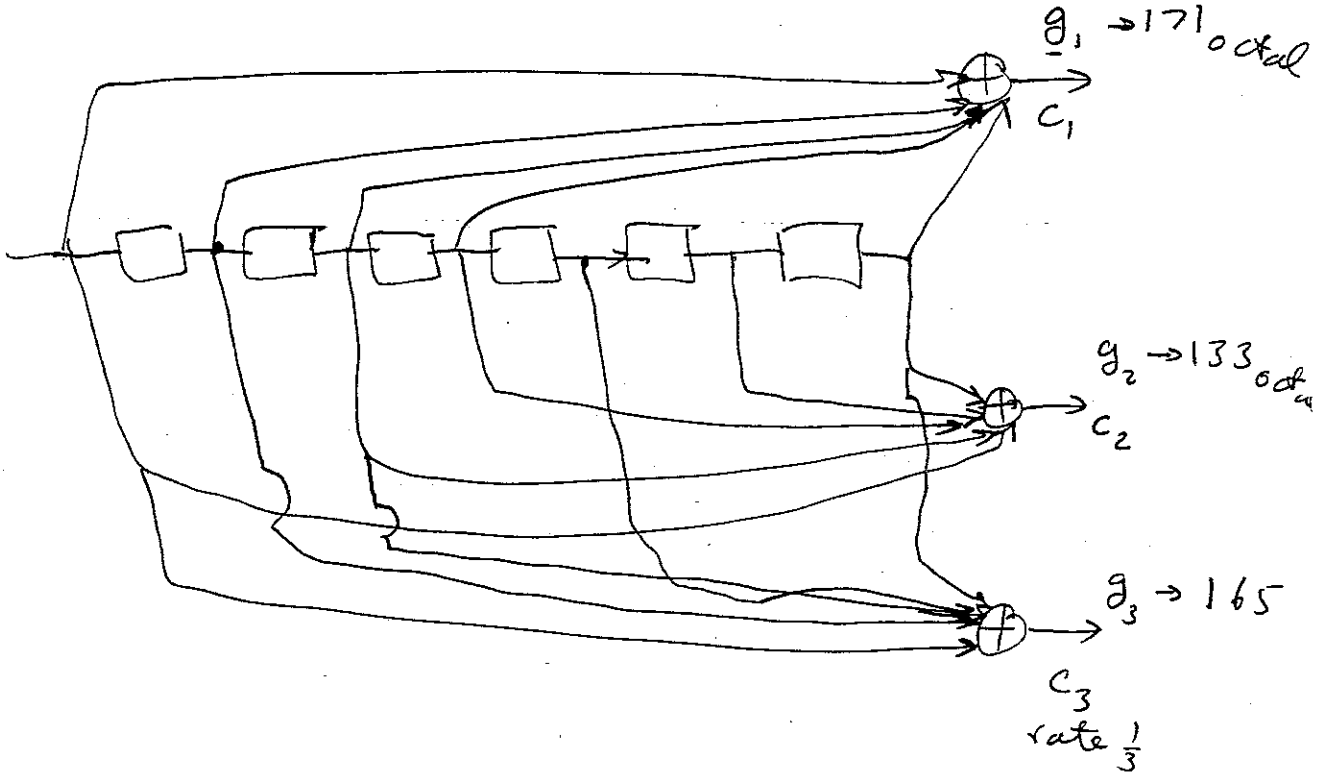
A de facto industrial standard code is a rate $\frac{1}{2}$ or $\frac{1}{3}$ code with $K = 7$ and

$\underline{g}_1 = (171)_{\text{octal}}$, $\underline{g}_2 = (133)$, $\underline{g}_3 = 165$

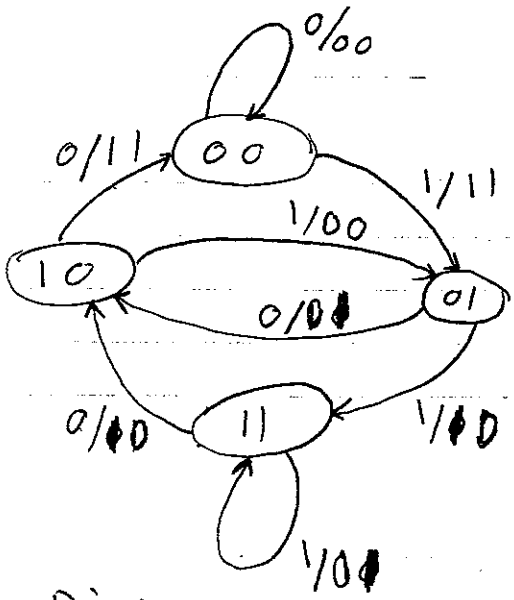
Check: http://people.qualcomm.com/karn/papers/cnc_coding.htm

http://www.istari-design.com/viterbi.htm (speeding up software ~~implementation~~ Simulation).

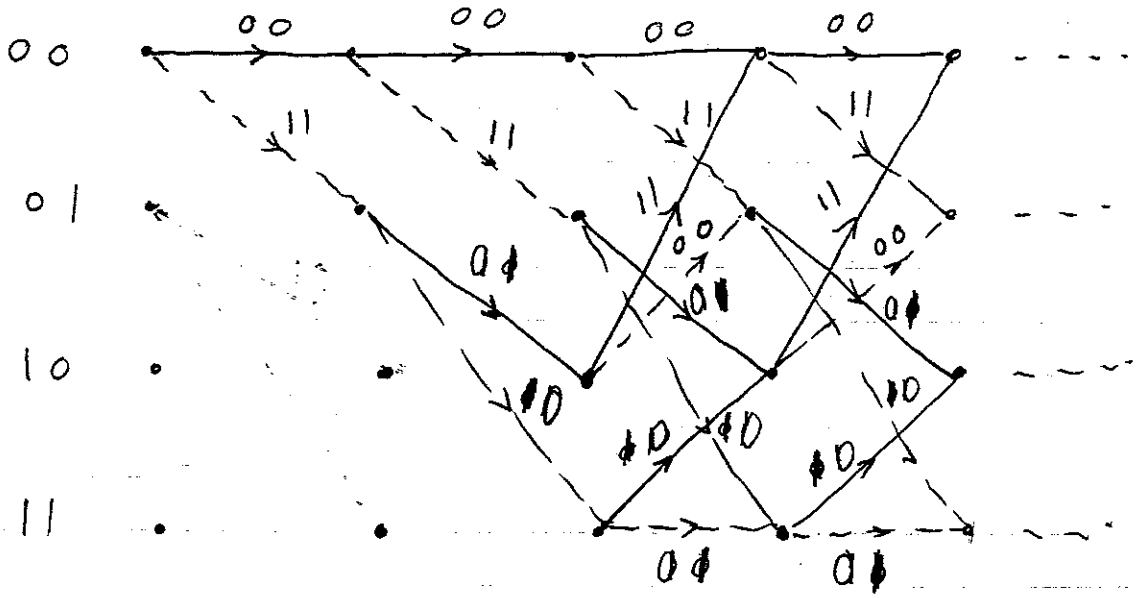
Q1900 / http://www.qualcomm.com/ProdTech/asic/vtdec.html



## Finite State representation




## Trellis Diagram




---

Viterbi Decoding Algorithm (VA)

VA tries to find the sequence closest to the received symbols. If it works on usable original samples at the output of the channel

(the sampled output of the matched filter) it is called soft decision Viterbi decoder.

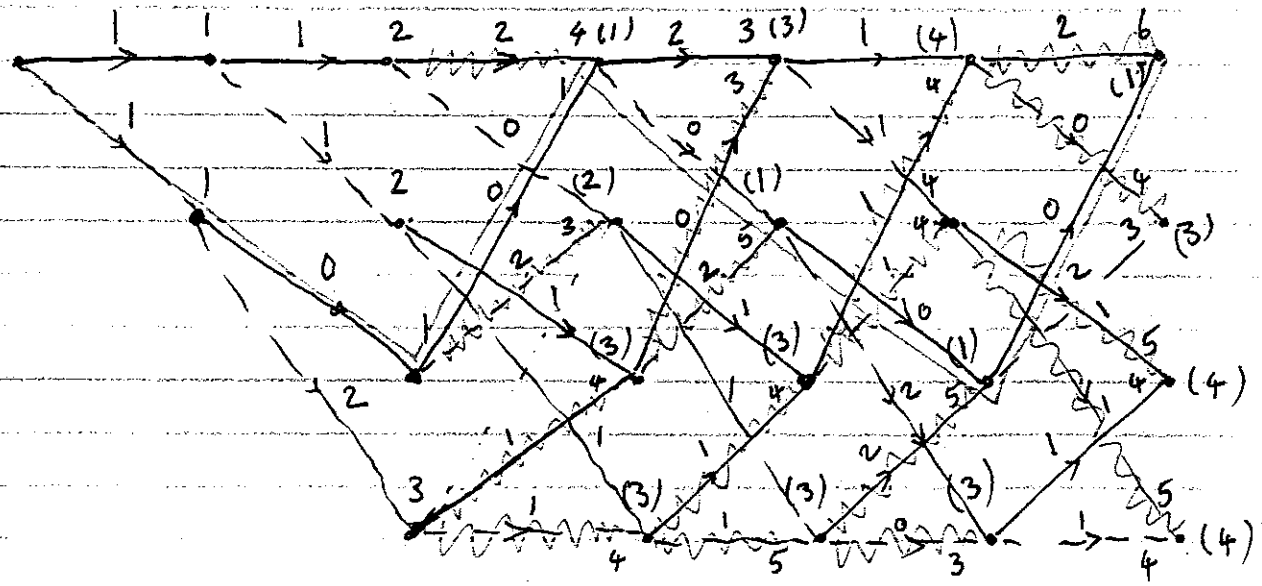
If VA works on the bits at the output of the demodulator, we have hard decision ($\tilde{\sim}$ 3 dB penalty in comparison with soft decision decoding

Example: Assume we had the message 1 0 0 1 and we sent 1 1, 0 0, 1 1, 1 1, 0 0, 1 1 and

(Tail)

received

1 0, 0 0, 1 1, 1 1, 0 0, 1 1



1 0 0 1 0 0