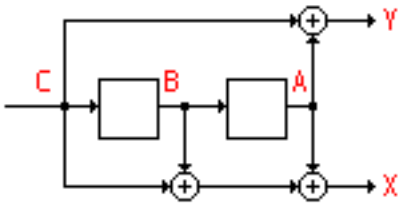
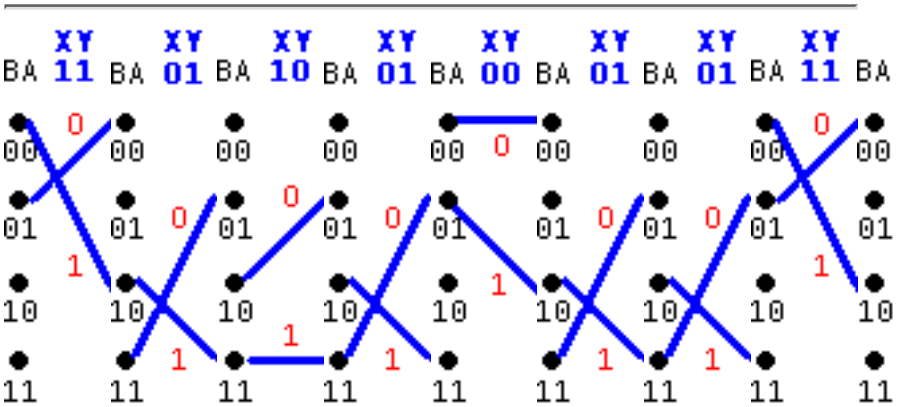
EE4253 Digital Communications

Convolutional Coder

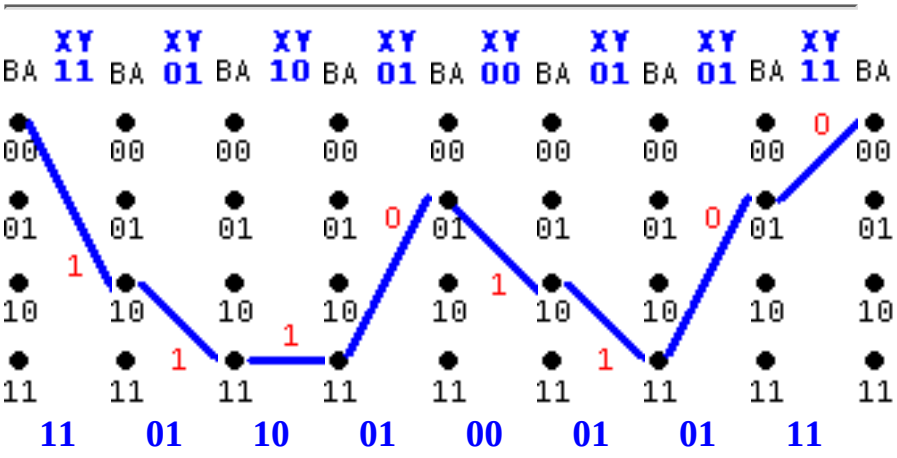
This online tool accepts data assumed to come from a (5,7) convolutional coder, and applies Viterbi decoding to find the original data stream, despite occasional errors in the received bits.



Viterbi Decoder



Input Symbols (XY)



Best-Fit Path

1 1 1 0 1 1 0 0

Recovered Data

Received Symbols: 11-01-10-01-00-01-01-11 Decode This!

EXAMPLES

Representing the input data **1 1 1 0 0 1 1 0 0**.

1. 11-01-10-01-00-01-01-11 (no errors)
2. 11-01-10-X1-00-01-01-11 (one error - corrected)
3. 11-01-10-X1-00-01-X1-11 (two isolated errors - corrected)
4. 11-01-10-01-XX-01-01-11 (two bit error, one symbol - corrected)
5. 11-01-1X-X1-00-01-01-11 (two bit errors, adjacent symbols - corrected)
6. 11-0X-10-X1-00-01-X1-11 (three isolated errors - corrected)
7. 11-01-1X-XX-00-01-01-11 (three bit errors, adjacent symbols - two potential solutions)
8. 11-01-XX-X1-00-01-01-11 (three bit errors, adjacent symbols - not properly corrected)

Wed Feb 27 05:32:11 AST 2002

---

16 OCT 01 - tervo@unb.ca [EE4253]
University of New Brunswick, Department of Electrical and Computer Engineering

EE4253 Digital Communications

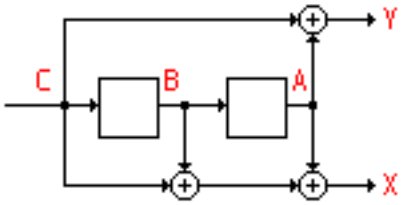
Convolutional Coding

Forward Error Correction (FEC) techniques such as the Hamming code are called *block coding*, because a "block" of data bits is examined to compute the required error control bits. For other applications, a method is required which is suitable for a continuous data stream. The technique of *convolutional coding* is well suited for long bit streams in noisy channels and is readily implemented in hardware.

Related references to this error control approach include "Trellis Coding", "Viterbi Decoding", and "Turbo Coding".

A Convolutional Coder

The circuit below illustrates a simple convolutional coder suitable for incorporating forward error correction into a transmitted message.



Circuit Description The circuit consists of two D-Flip-Flops connected in series with a common clock (not shown). Data arrives at point C and each bit passes through points C and B and A in three clock periods. The output bits X and Y are formed by exclusive-OR gates as shown, such that X=C+B+A and Y=C+A.

Defined by Connections The way in which the output bits are generated defines a particular convolutional coding circuit. In this case, the connections defining X and Y can be described as binary values 111 for X=C+B+A and 101 for Y=C+A. This circuit would be called a (5,7) coder.

A 1/2 Rate Coder For each bit arriving at point C, two bits (XY) are transmitted. In practice, a circuit at the output would switch quickly between X and Y to transmit an output bitstream at twice the rate of the input bit stream. Twice as many bits emerge from the circuit as enter at point C. In common with other error control techniques, convolutional coding involves increasing the length of the original message. A coder such as this which doubles the number of bits is called a "1/2 rate" (half rate) coder. In general, a 1/N rate code leads to a proportional length increase by N times the data bits.

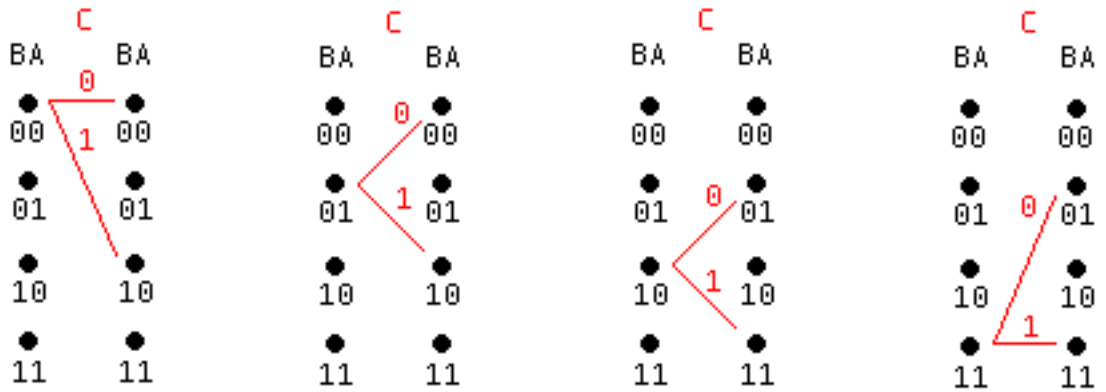
Combinatorial Circuit Analysis There are eight possible conditions for the three bits (CBA). Each combination leads to a unique pair of values for the outputs (XY), as summarized in the table below.

C	B	A	=	X	Y
0	0	0	=	0	0
0	0	1	=	1	1
0	1	0	=	1	0
0	1	1	=	0	1
1	0	0	=	1	1
1	0	1	=	0	0
1	1	0	=	0	1
1	1	1	=	1	0

Since the bits (CBA) represent the most recent three bits to enter the circuit, the outputs (XY) produced whenever a new bit arrives depend on the new bit (C) and the two previous bits (BA). This observation is key to the performance of the circuit as an error correction scheme.

Sequential Circuit Analysis Since the bits at B and A are defined by a shift register circuit fed from input C, the stored bits (BA) cannot jump arbitrarily from one state to another. For example, if the state (BA) of the flip flops is currently 00, it will change to 10 if the new bit is '1' (00 --1-> 10), and will remain 00 if the new bit is '0' (00 --0-> 00). Significantly, *the flip flops could never change directly from 00 to any state except 00 or 10.* Similarly, for each of the other states of the two flip flops, only two transitions are possible.

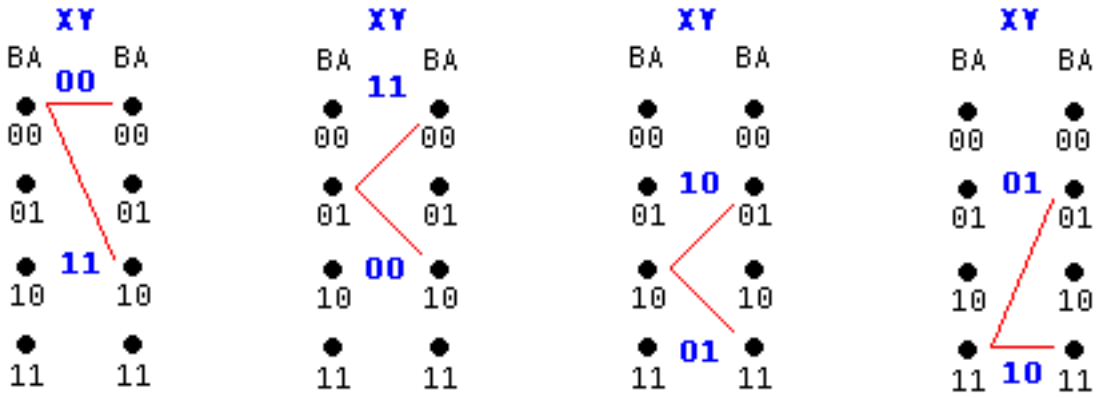
The behaviour of the two bit shift register can be summarized graphically below in a *trellis diagram* showing the four possible states (BA) and the two possible transitions from each state when a new bit (C) arrives. In each case, the lower path is followed for a input '1', and the upper path for an input '0'. (Note the exclusive-OR gates do not play any role in this trellis diagram.)



The effect of the exclusive-OR may now be included. Each three-bit condition (CBA) is associated with a unique two bit "symbol" (XY), the behavior of the circuit in time can be summarized in the table below. As as each new bit arrives one of two possible two bit "symbols" is generated as a function of the bit (C) and the shift register state (BA).

```
THIS  ------------> NEXT
STATE               STATE
BA -C--> CBA=(XY) next BA
-------------------------
00 -0--> 000=(00) ===> 00
00 -1--> 100=(11) ===> 10
-------------------------
01 -0--> 001=(11) ===> 00
01 -1--> 101=(00) ===> 10
-------------------------
10 -0--> 010=(10) ===> 01
10 -1--> 110=(01) ===> 11
-------------------------
11 -0--> 011=(01) ===> 01
11 -1--> 111=(10) ===> 11
-------------------------
```

The table can be mapped onto the trellis diagram to show the overall effect of the convolutional coder. In this figure the arrival of each new bit is now associated with a symbol (XY) to be transmitted. Recall that in each case, the lower path is followed for an input '1', and the upper path for an input '0'.



Coding Example

With the aid of the above table, a message can be coded and later decoded. The following bitstream is to be transmitted.

100111011

Assume the circuit starts in the state 00. When the first '1' arrives, the bits CBA=100 produce the bit pair XY=11. When the circuit is clocked, the new state BA is 10. This process can be shown as

```
BA -C--> CBA=(XY) next BA
00 -1--> 100=(11) ---> 10
```

The two bit "symbol" 11 will be transmitted in place of the single incoming data bit. This is a 1/2 rate encoder since two bits result from each incoming bit. (Two bits together are often called a "dibit")

The state BA is now 10 when the next bit arrives.

The entire process is summarized below to show the state of the circuit (BA) and the output symbols (XY) in time as each data bit (C) arrives. After the final data bit, the circuit can be put back into the state 00 by sending extra zeros. This also ensures that an error could be corrected in one of the final data bits.

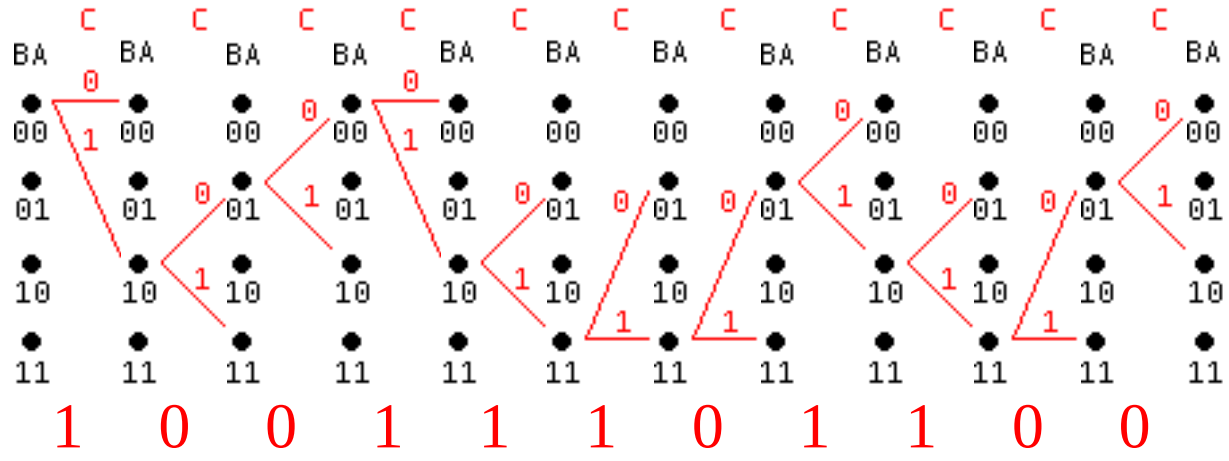
```
BA -C--> CBA=(XY) next BA

00 -1--> 100=(11) ===> 10

10 -0--> 010=(10) ===> 01

01 -0--> 001=(11) ===> 00

00 -1--> 100=(11) ===> 10

10 -1--> 110=(01) ===> 11

11 -1--> 111=(10) ===> 11

11 -0--> 011=(01) ===> 01

01 -1--> 101=(00) ===> 10

10 -1--> 110=(01) ===> 11

--- flush with zeros at the end of data...

11 -0--> 011=(01) ===> 01

01 -0--> 001=(11) ===> 00
```

The resulting message is transmitted as:

11 10 11 11 01 10 01 00 01 01 11

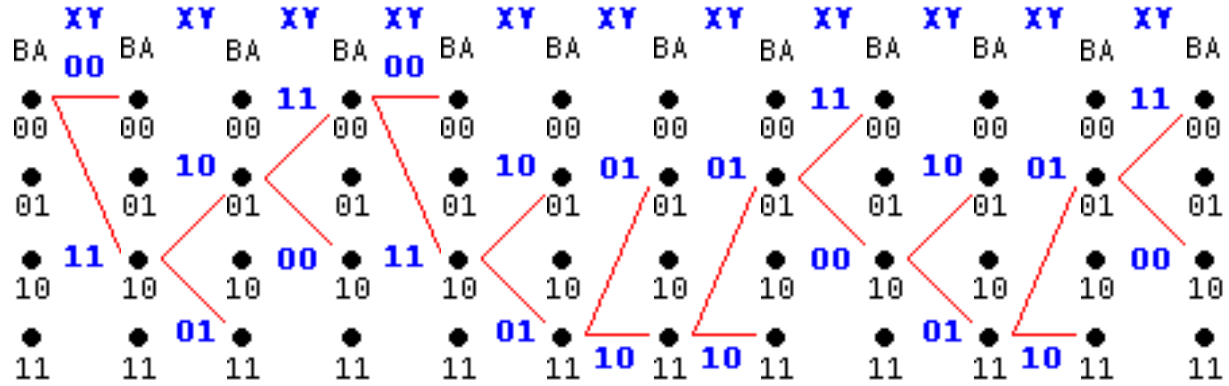
(*Spaces are left between each bit pair for clarity.*)

A Trellis Diagram It can be observed that during the above encoding process, the circuit passes through states (BA) shown on the *Trellis Diagram* shown below. Since for each incoming bit there are only two possible branches to a new state, the trellis diagram illustrates how the convolutional coder serves to constrain potential paths as time moves from left to right. A distinct path through the trellis can be traced as a function of the incoming bits.



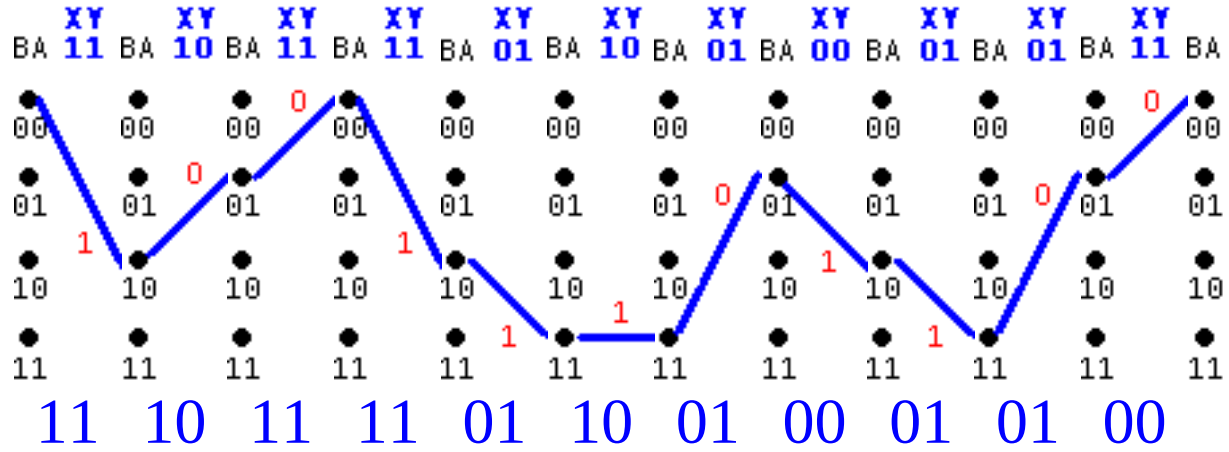
Incoming Bits

The exclusive-OR circuitry maps a two-bit symbol (XY) onto each of the above state transitions. The trellis diagram can now be redrawn to show the symbols associated with the above states.



The continuous path through the trellis defines the transmitted symbols.

By tracing the continous path through the trellis, the dibit symbols are determined. Note the the transmitted data does *not* directly incorporate the original data bits. Instead, the new bit stream with double the number of bits describes a distinct path through the trellis from which the original data bits can be extracted.



Transmitted Symbols

Upon reception, it is necessary to reconstruct the original bitstream essentially by retracing the above path from the symbols which are received. Error correction is possible because an errored symbol will cause a detectable deviation from the correct path. The remaining symbols may be then used to reconstruct the most likely correct path and consequently, the correct bitstream. This process is explored in the next section.

The EE4253 Online Convolutional Coder is available to explore encoding short messages using convolutional coding.

Convolutional coding is fast, efficient, and readily implemented in hardware. It is the method of choice for error control in many applications.

---

12 OCT 01 - tervo@unb.ca
University of New Brunswick, Department of Electrical and Computer Engineering

EE4253 Digital Communications

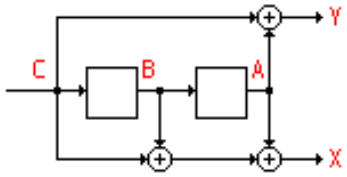
Convolutional Coding - Viterbi Decoding

The technique of convolutional coding transforms a binary message into a sequence of symbols to be transmitted. Upon reception, the received information must be related back to the original message bits. If there are no errors, the process of decoding is readily accomplished. It has already been shown that single bit errors can be corrected by simple inspection of the received symbols.

In general, convolutional coding techniques are applied to very long messages, such as the continuous stream of data from a satellite television transmiter. A fast approach to applying bit correction dynamically during reception is required. The method of *Viterbi Decoding* is widely used to decode and to correct errors in convolutionally encoded data.

A Convolutional Coder

The circuit below illustrates a simple convolutional coder suitable for incorporating forward error correction into a transmitted message.



In this section, it is assumed that a data stream has passed through the above circuit to yield a series of two bit (dibit) symbols representing the data. It is now necessary to examine the incoming bits and to extract the original data. Of course, the symbols received may not be exactly the same as those which were originally transmitted.

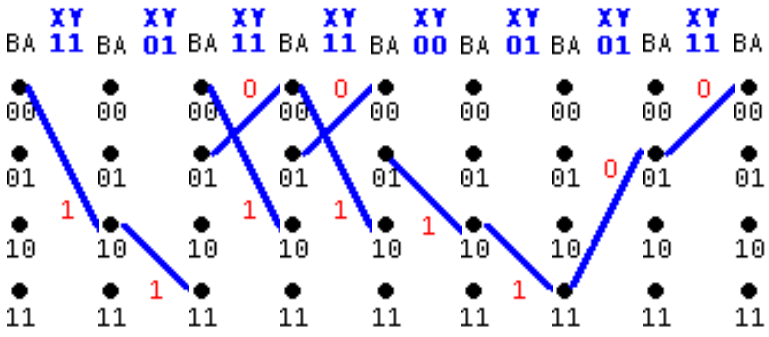
Error Correction Example

With the aid of the trellis diagram, a received message can be related back to the original bitstream even if the received data contains errors. Assume the circuit starts at 00, and the following short symbol stream is received.

> Note there are errors in two adjacent bits spanning two symbols, as shown highlighted below. *Of course, this fact is not yet known to the decoder!*

11 01 11 11 00 01 01 11

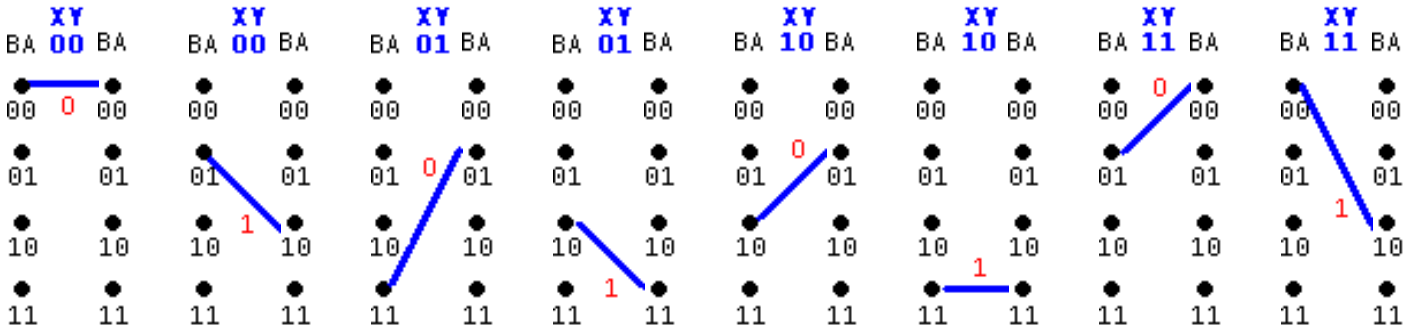
The problem is obvious if the symbols are simply accepted as received:



1 1 ? ? 1 1 0 0

Errored Data Bits (no continuous path)

It is necessary to reconstruct a continuous path from the above information. The "best fit" path will be the one which best matches the above data, deviating only as required to establish continuity. The available resources to establish this path are the four possible symbols and the corresponding eight possible state transitions:

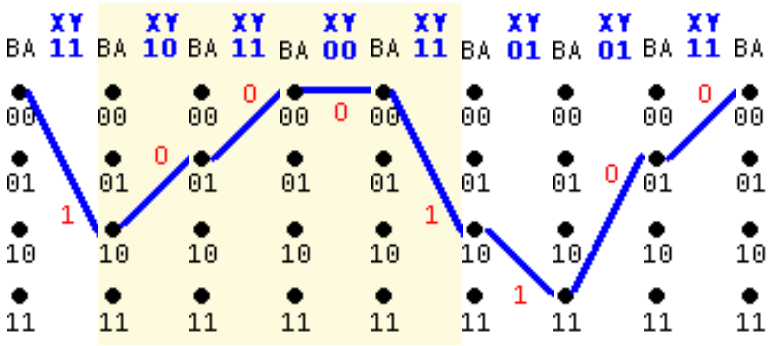


Available elements to establish a continuous path

Proceeding by trial and error to find the "missing symbols" presupposes that the location of the bad symbols was known and greatly oversimplifies the decoding process. *Perhaps the symbols leading up to the "obvious" break in the path are themselves incorrect.* In general, **the exact location of any bad symbols cannot be known in advance**. In practice, a systematic approach to identifying the best path through the trellis can be employed to correct errors automatically as they occur.

Alternative Continuous Paths

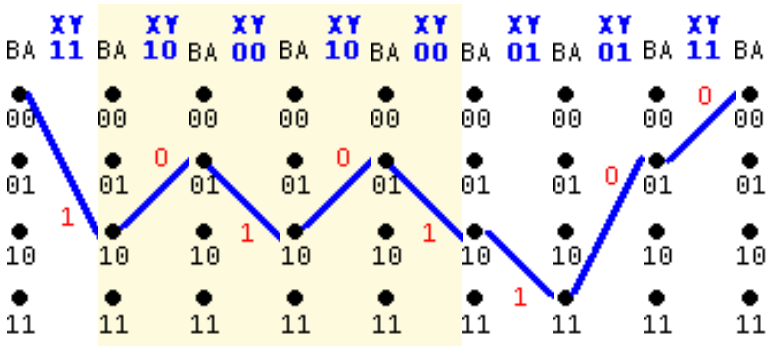
In fact, there are four paths of interest which can be considered in correcting the problem identified above. Each choice reconstructs a continuous path through the trellis in the region around the affected symbols, and one will be chosen as the best solution.



Candidate Path #1

```
RECEIVED DATA:11 01 11 11 00 01 01 11
PROPOSED PATH:11 10 11 00 11 01 01 11
                =======================
   DIFFERENCE:00 11 00 11 11 00 00 00
```

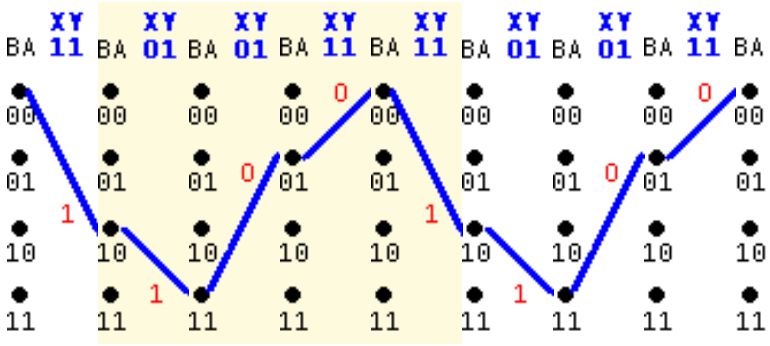
Giving **total path error = 6**.



Candidate Path #2

```
RECEIVED DATA:11 01 11 11 00 01 01 11
PROPOSED PATH:11 10 00 10 00 01 01 11
                =======================
   DIFFERENCE:00 11 11 01 00 00 00 00
```

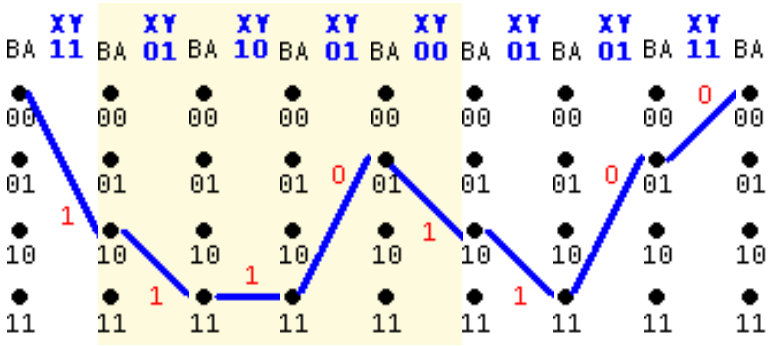
Giving **total path error = 5**.



Candidate Path #3

```
RECEIVED DATA:11 01 11 11 00 01 01 11
PROPOSED PATH:11 01 01 11 11 01 01 11
                ======================
   DIFFERENCE:00 00 10 00 11 00 00 00
```

Giving **total path error = 3**.



Candidate Path #4

```
RECEIVED DATA:11 01 11 11 00 01 01 11
PROPOSED PATH:11 01 10 01 00 01 01 11
                ======================
   DIFFERENCE:00 00 01 10 00 00 00 00
```

Giving **total path error = 2**.

Other paths beyond the range of the known problem area can only lead to higher differences. (All four of the above paths have zero error in the final three symbols and there is no point proposing any additional paths.)

Of the above choices, Path #4 gives the lowest error, and this correction is chosen. A two bit error has been sucessfully corrected.

Viterbi Decoding

It is impractical to identify all possible paths through a very long message and then evaluate the Hamming distance between each of them against the received symbols.

Fortunately, the properties of the trellis make possible an iterative approach in which all possible paths are explored, but only the best paths are pursued.

At each time interval, the trellis can have only four possible states (BA), although many alternate paths may have led to those four states. Each state in the circuit can lead to one of two states in the next time interval. Conversely, a given state can be arrived at from one of exactly two states from the previous time interval.

In other words, exactly two incoming paths always lead to each of the four states. In general, one of those two incoming paths will have the lowest error *up to that point in the trellis*. By choosing the 'best path so far' and moving on to the next stage, the overall best path will be the one with the smallest 'cumulative error' at it traverses the trellis.

Viterbi Decoding Algorithm

As above, the following message is received with two errors:

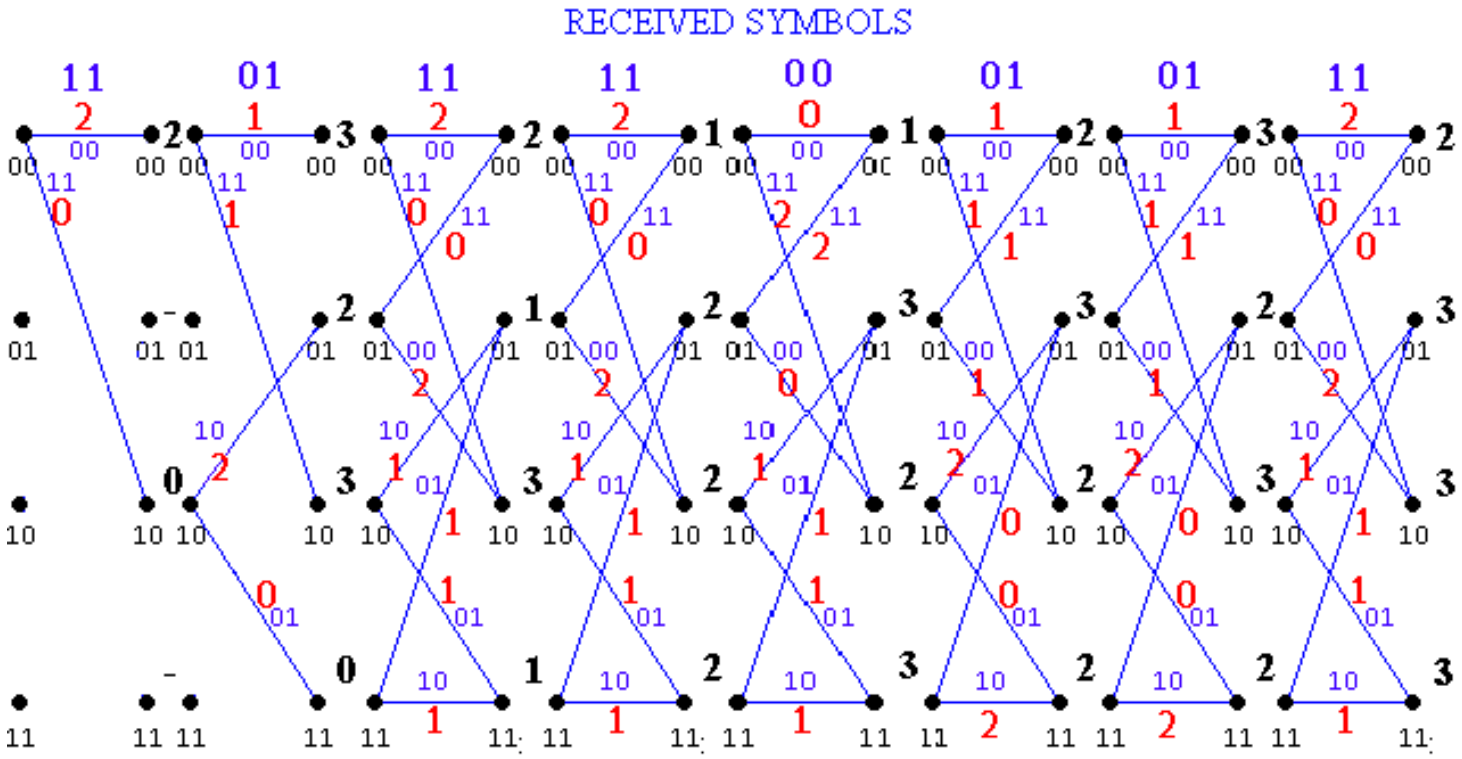
11 01 1<u>1</u> <u>1</u>1 00 01 01 11

Step 1 - The process begins at the start of the path, where it is always assumed the circuit begins at state 00. From here, only two paths are possible. Since the first received symbol is ' 11', the lower path (11) involves an error of 0, while the upper path (00) involves an error of 2. These two values are recorded on the table for the next step. Any further progress along the upper path will include and possibly add to this error 2.

Step 2 - Four paths are defined leading to the next stage, two from each of the incoming states. The error associated with a given path segment depends on the actual symbol ' 01' received, as compared to the symbol defined for that segment. Here, the errors are shown superimposed on each path segment, and the *cumulative error* is shown to begin the next stage. Any further progress along the upper path will include and possibly add to the error at state 00 which is now 3. The best path so far appears to be the lower path which still has zero error.

Step 3 - Eight paths are defined leading to the next stage, two from each of the four incoming states. The error associated with a given path segment depends on the actual symbol '11' received, as compared to the symbol defined for that segment. Again, the errors are shown superimposed on each path segment, and the *cumulative error* is shown at the end of this stage; however, *only the best (smallest) error of the two paths leading to that state is shown*. Whenever two paths arrive at the same state, the one with the lowest cumulative error is retained, and the other is discarded. For example, the path along the top of the table (00-00-00) arrives at state 00 with a cumulative error of 5. On the other hand, another path (symbols 11-10-11) leads to state 00 with a cumulative error of 2. The second path is retained with an error of 2 representing the best path to arrive at this state. The best overall path is uncertain at this point since there are two states with error 1 (of course, an error in the received data has just occurred).

Continuing... - At each successive stage, the four best path errors are kept, and the overall correct path is expected to emerge from the computations...

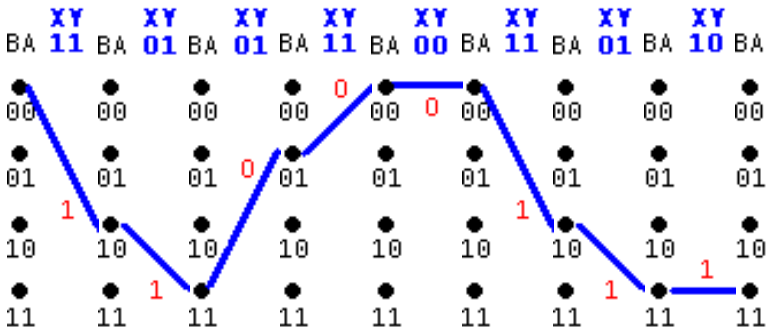


At the end of the table (far righthand side) there is one path which gives the lowest cumulative error (2). In fact, this path corresponds to the correct data. The above algorithm could be easily be carried through if the message were to continue from this point.

It is interesting to note that a better path seemed to be emerging at the received symbols -11-00-, when the uppermost path had a cumulative error of 1. This potential path soon lost out to the actual path which was eventually found to have less error. The promising path was a dead end when it deviated too far from the true path.

Path Analysis

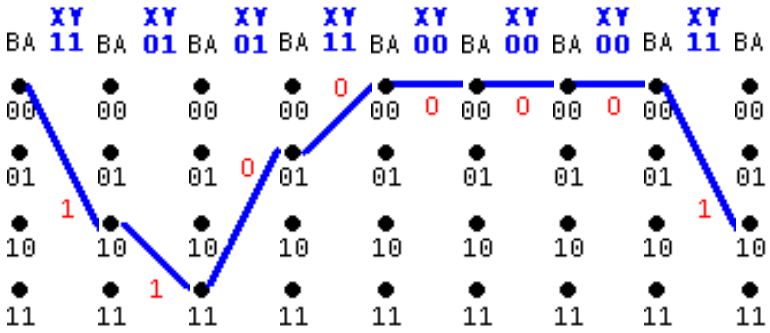
The above algorithm has defined best paths leading to each of the four terminating states. Each choice reconstructs a continuous path through the trellis which has the minimum cumulative error of all other possible paths to that point.



Best Path Ending at State 11

```
RECEIVED DATA:11 01 11 11 00 01 01 11

BEST-FIT PATH:11 01 01 11 00 11 01 10

                ======================

   DIFFERENCE:00 00 10 00 00 10 00 01
```

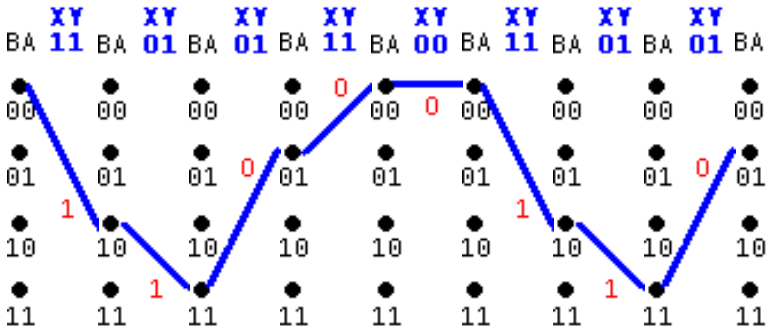
Giving **total path error = 3**.



Best Path Ending at State 10

```
RECEIVED DATA:11 01 11 11 00 01 01 11

BEST-FIT PATH:11 01 01 11 00 00 00 11

                ======================

   DIFFERENCE:00 00 10 00 00 01 01 00
```

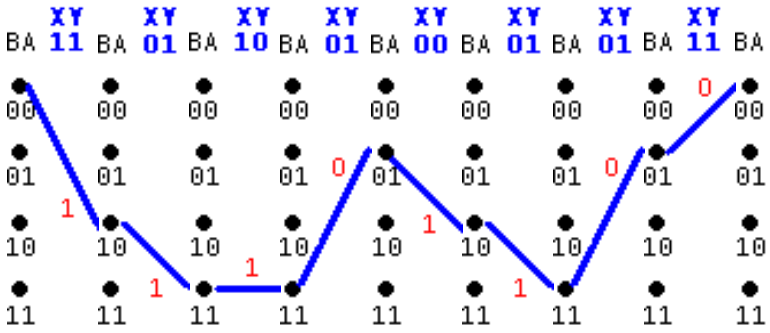
Giving **total path error = 3**.



Best Path Ending at State 01

```
RECEIVED DATA:11 01 11 11 00 01 01 11

BEST-FIT PATH:11 01 01 11 00 11 01 01

                ======================

   DIFFERENCE:00 00 10 00 00 10 00 10
```

Giving **total path error = 3**.



Best Path Ending at State 00

```
RECEIVED DATA:11 01 11 11 00 01 01 11

BEST-FIT PATH:11 01 10 01 00 01 01 11

                ======================

   DIFFERENCE:00 00 01 10 00 00 00 00
```

Giving **total path error = 2**.

The EE4253 Online Viterbi Decoder is available to explore bit correction in short messages using Viterbi decoding.

The Viterbi Decoding algorithm chooses the best-fit path through the trellis, correcting for multiple bit errors as the received data arrives.

---

15 OCT 01 - tervo@unb.ca
University of New Brunswick, Department of Electrical and Computer Engineering