4D ENCODING IN LEVEL-ONE'S PROPOSAL FOR 1000BASE-T

by

Jaime E. Kardontchik

Advanced Micro Devices

Sunnyvale, CA 94088

August 21, 1997 - Rev B

The purpose of this tutorial is to fill-in the missing steps that lead to the final definition of the 8 subsets of the 4D encoding, as presented many times by Sailesh Rao, from Level-One, and that I reproduce here for convenience from his last presentation in Maui:

SUBSET MAPPING

Subset	members
D0	XXXX + YYYY
D1	XXXY + YYYX
D2	XXYY + YYXX
D3	XXYX + YYXY
D4	XYYX + YXXY
D5	XYYY + YXXX
D6	XYXY + YXYX
D7	XYXX + YXYY

Members of the Task Force come with different backgrounds and expertise, and it is sometimes difficult to follow and understand all the technical details of one proposal. The objective of this tutorial is to increase the understanding of the 4D coding within the Task Force.

A simple editor (vi) was chosen to save a lot of time and allow an easy and wide distribution of the tutorial.

....

1.- ONE-DIMENSIONAL VIEW AND NOMENCLATURE

If we look at one transmitter, we have basically a PAM-5 system, with levels:

$$\{-2, -1, 0, 1, 2\}$$

and the minimum distance between levels is 1 Volt.

We define now Y and X subsets as follows:

$$Y = \{-2, 0, 2\}$$

$$X = \{-1, 1\}$$

So, when we talk, for example, about an X symbol we refer either to a -1 or +1 level. Similarly, a Y symbol represents a -2, or 0 or +2 level.

....

2.- TWO DIMENSIONAL VIEW

With two transmitters, each one sending PAM-5 levels, we obtain a two dimensional constellation, PAM5x5, consisting of 25 points:

```
              (-2,2)  (-1,2)  (0,2)   (1,2)   (2,2)
                *       *       *       *       *



                *       *       *       *       *



                *       *       *       *       *



                *       *       *       *       *


                                              (2,-2)
                *       *       *       *       *
```

Fig 1: two-dimensional PAM5x5 constellation

where I added in parenthesis the levels represented by some points, so the meaning becomes clear. For example, the point (-1,2) represents transmitter # 1 sending a level -1 V and, at the same time, transmitter # 2 sending a level +2 V. The minimum distance between points in this 2D space is again 1 Volt:

....

3.- TRELLIS CODING

If we do not introduce any additional coding, the receiver will have to recover the transmitted data at a symbol-per-symbol basis, where the distance between symbols is 1. The idea of Trellis-Coding is to introduce an additional structure in the transmitter so that the basic units sent are sequences instead of individual symbols. In addition, not any arbitrary sequence will be allowed, and the allowed sequences will be such that the Euclidean distance between them will be greater than 1, making easier for the receiver to distinguish different sequences (distance > 1) than to distinguish different individual symbols (distance = 1). This will minimize the probability of error in the receiver. In Level-One's proposal the minimum squared distance between valid sequences is 4. We do not explain here how these sequences (or trellises) are generated. We only explain in this tutorial how this additional structure can be generated and its properties.

....

4.- PARTITIONING - STEP ONE

The first step towards creating these allowable sequences is to divide the 2D constellation into two subsets by assigning alternate points to each subset, i.e., according to the pattern

A	B	A	B	A
B	A	B	A	B
A	B	A	B	A
B	A	B	A	B
A	B	A	B	A

Fig 2: subset partitioning of the PAM5x5 constellation

(see Forney et al, Reference 1, page 638). The most important points to notice is:

a) the points in each subset lie again on a square grid (rotated 45 degrees with respect to the original grid)
b) the minimum squared distance between points within a subset (= 2) is twice the minimum squared distance between points in the original constellation (= 1).
c) because of the first property, the partitioning can be repeated to yied smaller subsets with minimum squared distance between points of 4, 8, 16, ... and so on

We can redraw this pattern as follows using our X-Y nomenclature for the PAM-5 levels:

YY	XY	YY	XY	YY
YX	XX	YX	XX	YX
YY	XY	YY	XY	YY
YX	XX	YX	XX	YX
YY	XY	YY	XY	YY

Fig 3: as Fig 2, but using the X-Y nomenclature to identify the different points of the constellation

A point XY in Fig 3, means that transmitter # 1 sent a type X level (that could be either -1 or +1) and transmitter # 2 sent a type Y level (that could be either -2 or 0 or +2).

Figures 4a and 4b show the two subsets of 'A' and 'B' type points separately:

```
                YY              YY              YY


                        XX              XX


                YY              YY              YY


                        XX              XX


                YY              YY              YY
```

        Fig 4a: subset of 'A' points

```
                        XY              XY


                YX              YX              YX


                        XY              XY


                YX              YX              YX


                        XY              XY
```

        Fig 4b: subset of 'B' points

We will call the subset shown in Fig 4a the EVEN subset, and the subset shown in Fig 4b the ODD subset. The reason for this naming is that the sum of the level values of any point belonging to the EVEN subset is an even number, and the sum of the level values of any point belonging to the ODD subset is an odd number.

Notice that the minimum squared distance between points in each subset is 2.

....

5.- PARTITIONING - STEP TWO

5.a) Partitioning of the EVEN subset

We will further subdivide the EVEN and ODD partitions into two subsets each. I will show in great detail how this is done, step by step. For instance, let us take the EVEN subset of Fig 4a. The first step is to look at Fig 4a at an angle of 45 degrees, in which case it will look as shown in Fig 5:

```
 .       .       YY      .       .


 .       YY      XX      YY      .


 YY      XX      YY      XX      YY


 .       YY      XX      YY      .


 .       .       YY      .       .
```

Fig 5: EVEN subset from Fig 4a after 45 degrees rotation

From Fig 5 we see that we have again a square lattice, that can be subdivided into two subsets, 'A' and 'B', as we did before. The results are shown in Figs 6a and 6b:

```
                .       .       YY      .       .

                .       YY              YY      .

                YY              YY              YY

                .       YY              YY      .

                .       .       YY      .       .
```

Fig 6a: Subset A of the EVEN subset

```
                .       .               .       .

                .               XX              .

                        XX              XX

                .               XX              .

                .       .               .       .
```

Fig 6b: Subset B of the EVEN subset

And now we can rotate Fig 6a and 6b back 45 degrees to get the subsets look more normally, as shown in Fig. 6c and 6d:

```
                YY              YY              YY



                YY              YY              YY



                YY              YY              YY
```

Fig 6c: Subset A of the EVEN subset, after another
45 degree rotation: (YY) subset

The points represented in Fig 6c are:

```
                (-2,2)          (0,2)           (2,2)


                (-2,0)          (0,0)           (2,0)


                (-2,-2)         (0,-2)          (2,-2)
```

And here is the back to normal subset of Fig 6b:

```
XX              XX




XX              XX
```

Fig 6d: Subset B of the EVEN subset, after another 45 degree rotation: (XX) subset

The points represented in Fig 6d are:

```
(-1,1)          (1,1)


(-1,-1)         (1,-1)
```

It is natural to call the subset A of the EVEN subset (Fig 6c) the (YY) subset, and the subset B of the EVEN subset (Fig 6d) the (XX) subset.

Notice that the minimum squared distance between points in each subset is now 4.

....

5.b) Partitioning of the ODD subset

Let us take now the ODD subset of Fig 4b and rotate it by 45 degrees. It will look as shown in Fig 7:

```
.       XY      YX      .

XY      YX      XY      YX

YX      XY      YX      XY

.       YX      XY      .
```

Fig 7: ODD subset from Fig 4b after 45 degrees rotation

From Fig 7 we see that we have again a square lattice, that can be subdivided into two subsets, 'A' and 'B', as we did before. The results are shown in Figs 7a and 7b:

```
.       .       YX      .

.       YX      .       YX

YX      .       YX      .

.       YX      .       .
```

Fig 7a: Subset A of the ODD subset

```
                .       XY      .       .


                XY      .       XY      .


                .       XY      .       XY


                .       .       XY      .
```

Fig 7b: Subset B of the ODD subset

Now we can rotate Figs 7a and 7b back 45 degrees to get the subsets look more normally, as shown in Figs. 7c and 7d:

```
                YX              YX              YX




                YX              YX              YX
```

Fig 7c: Subset A of the ODD subset, after rotating back 45 degrees: (YX) subset

Explicitly, the points in Fig 7c represent:

```
                (-2,1)          (0,1)           (2,1)

                (-2,-1)         (0,-1)          (2,-1)
```

```
                        XY              XY



                        XY              XY



                        XY              XY
```

Fig 7d: Subset B of the ODD subset, after rotating
back 45 degrees: (XY) subset

Explicitly, the points in Fig 7d represent:

```
                        (-1,2)          (1,2)

                        (-1,0)          (1,0)

                        (-1,-2)         (1,-2)
```

It is natural to call the subset A of the ODD subset (Fig 7c) the (YX) subset, and the subset B of the ODD subset (Fig 7d) the (XY) subset.

Notice that the minimum squared distance between points in each subset is now 4.

....

We have now finished the subdivision of the original PAM5x5 set into four subsets, and this is all what we need to continue to the next step: the building of the 4D constellation. Notice that we could have continue subdividing the subsets to get further increases in the minimum squared distance (8, 16, ...), but for our purposes this is not needed: it would increase the total number of states of the encoder and make more complex the decoder at the receiver.

Let us summarize what we have: our original PAM5x5 constellation was divided into an EVEN and an ODD subsets, which were further subdivided into two subsets each:

(YY), (XX), (YX) and (XY)

The minimum squared distance between points in each of these four subsets is 4. These subsets are shown in Figs. 6c, 6d, 7c and 7d.

....

6.- MULTIDIMENSIONAL CONSTELLATIONS

The 4D coding can be derived from the four 2D subsets, (YY), (XX), (YX) and (XY) following the steps suggested either by Ungerboeck (Ref 2, page 17) or by Wei (Ref 3, page 484). I decided, arbitrarily, to follow more closely Wei's steps and nomenclature.

....

7.- DEFINE 4D 'TYPES'

Using now the four transmitters we can define sixteen 4D 'types' (I use Wei's nomenclature) by concatenating pairs of 2D subsets. These 'types' are shown in the following Table:

Table I: 4D TYPES

type #	content
1	YYYY
2	YYXX
3	YYYX
4	YYXY
5	XXYY
6	XXXX
7	XXYX
8	XXXY
9	YXYY
10	YXXX
11	YXYX
12	YXXY
13	XYYY
14	XYXX
15	XYYX
16	XYXY

For example, if at time t = k*T, where k is an integer and T is the baud period,

```
        Tx # 1 sends    -1V
        Tx # 2 sends     2V
        Tx # 3 sends     1V
        Tx # 4 sends     1V
```

the quartet (-1,2,1,1) is of type XYXX. We can look at these points as points in a four-dimensional space. Within a given 'type', the minimum squared distance between points is still 4. For instance, the points:

(-1,2,1,1) and (-1,0,1,1)

belong both to type XYXX. The Euclidean squared distance between these points is:

[(-1) - (-1)]^2 + [2 - 0]^2 + [1 - 1]^2 + [1 - 1]^2 = 4

....

8.- DEFINE 4D SUBLATTICES

The sixteen types may be grouped into eigth 4D 'sublattices', as shown in Table II:

Table II: 4D SUBLATTICES

sublattice #	content	# of points	pruned to
D0	XXXX + YYYY	97	64
D1	XXXY + YYYX	78	64
D2	XXYY + YYXX	72	64
D3	XXYX + YYXY	78	64
D4	XYYX + YXXY	72	64
D5	XYYY + YXXX	78	64
D6	XYXY + YXYX	72	64
D7	XYXX + YXYY	78	64
	total:	625	512

It is easily shown that with this type of grouping the minimum distance between any pair of points within a given sublattice is still 4. For example, let us take the following points in D1:

(1,-1,1,2) (belonging to XXXY)

and

(2,0,2,1) (belonging to YYYX)

The Euclidean distance between these two points is:

$$[1 - 2]^2 + [-1 - 0]^2 + [1 - 2]^2 + [2 - 1]^2 = 4$$

In general, it is clear that if one point belongs to XXXY and the other to YYYX, the distance between the components of the points in any dimension must be equal or larger than 1, since, the minimum distance

between X and Y points is 1. Remember that X = {-1, 1} and Y = {-2, 0, 2}.
Hence, the mininum squared distance between any pair of four dimensional points in a given sublattice must be equal or greater than 4.

And the same can be said for all the eight sublattices: notice that the sixteen 'types' have been grouped in such a way that if the component of one point on one axis is X, then the component of the second point on the same axis (or dimension) is Y, and viceversa.
For example, looking at the two 'types' that belong to the D7 sublattice:

```
        D7:     X   Y   X   X

                |   |   |   |
                |   |   |   |

                Y   X   Y   Y
```

The minimum squared distance, min_squar_dist, between any two points belonging to sublattice D7, one belonging to type XYXX and the other belonging to type YXYY, is:

```
        min_squar_dist = min[(X-Y)^2] + min[(Y-X)^2] +

                         min[(X-Y)^2] + min[(X-Y)^2] = 1+1+1+1 = 4
```

The reason for grouping the sixteen 'types' into pairs to form eight sublattices is to simplify the coding and decoding: we have created in this way a 4D 8-state encoder, instead of a 4D 16-state encoder.

The total number of points in each sublattice is shown in a separate column. The way it is calculated is very simple. For example, for sublattice D3 the total number of points is:

```
        # of points in D3 = 2*2*3*2 + 3*3*2*3 = 78
```

....

9.- DEFINE 4D FAMILIES

The eight 4D sublattices may be further grouped into two 4D families, with minimum squared distance of 2 between points in a family. These two families are shown in Table III:

Table III: 4D FAMILIES

family	content
EVEN	D0 + D2 + D4 + D6
ODD	D1 + D3 + D5 + D7

The reason for naming these families 'EVEN' and 'ODD' is that any point in the EVEN family has an even number of X's, and any point in the ODD family has an odd number of X, or, any other words, the sum modulo-2 of the coordinates of any point belonging to the EVEN family is 0, and the sum modulo-2 of the coordinates of any point belonging to the ODD family is 1.

....

10.- HIERARCHY OF THE PARTITIONING

We may now summarize the 4D partitioning as shown in Fig. 8:

```
                        |---    D0
                        |
                        |---    D2
        ----- EVEN -----|
        |               |---    D4
        |               |
        |               |---    D6
        |
4D -----|
        |
        |               |---    D1
        |               |
        |               |---    D3
        ----- ODD ------|
                        |---    D5
                        |
                        |---    D7

lattice     families            sublattices
```

Fig. 8: partitioning of the 4D lattice

At the top level, the minimum squared distance between points in the lattice is 1. At the second level, the minimum squared distance between points that belong to the same family is 2. At the third level, the minimum squared distance between points that belong to the same sublattice is 4.

The Parity bit in Level-One's Trellis Encoder selects the family (EVEN or ODD). The bits T_D[1] and T_D[0] select one out of the four sublattices within the chosen family. The other six bits, T_D[7:2], select a particular point within the chosen sublattice (a sublattice contains 64 points). The transmitter sends this point to the wires (remember that this is a 4D point, i.e., one component, or voltage level, per transmitter, per clock period).

The meaning and use of the Parity bit and T_D[7:0] are further clarified in Fig 9, taken from Level-One's presentations:

```
             -----
             | S |
             |   |-------------------------------- T_D[7] --
---          | C |                                          |
  |          |   |-------------------------------- T_D[6]  |
G |          | R |                                          |
  |          |   |-------------------------------- T_D[5]  | select
M |Tx_D[7:0] | A |                                          \ point
  |----------|   |-------------------------------- T_D[4]  / in
I |          | M |                                          | subset
  |          |   |-------------------------------- T_D[3]  |
I |          | B |                                          |
  |          |   |-------------------------------- T_D[2] --
---          | L |                                         --
             |   |-------------------------------- T_D[1]  |
             | E |     |                           |
             |   |-----(-------------------------- T_D[0]  |
             | R |     |          |                         \ select
             -----     |          |                         / subset
                       |  |                        |
                ----   |   ----   |   ----                  |
            ----|  |--(+)--|  |--(+)--|  |-------- PARITY  |
            |   ----       ----       ----  |               |
            |                               |              --
            ---------------------------------
```

Fig 9: Trellis encoder used in Level-One's proposal
subset = sublattice

Notice that the encoder has three flip-flops, or memory units: the selected subset will then depend on T_D[1:0] and on the values stored in these flip-flops, or, in other words, it will depend on the state of the encoder. This is a simple 8-state machine, where the next state will depend on the present state and on the value of T_D[1:0].

....

11.- WHO NEEDS ALL THIS ?

The Viterbi decoder.

The Viterbi decoder in the receiver compares all the paths diverging from one node (or state) A of the encoder and merging after some time later into another node B. The measure of comparison is the Euclidean distance between the different paths. The more different these paths are (in terms of Euclidean distance) the easier it will be to find and choose the actual path followed by the encoder.

If we follow the following five rules the minimum Euclidean distance between any two different paths will be 4:

a) transitions from a given state A at time k*T to a state B at time (k+1)*T

rule 1: when the encoder goes from A to B,

```
state:         A ----- B

at time:       k       (k+1)
```

all the possible output symbols must belong to the same sublattice (hence, there are only 64 possible different transitions from A to B).

From rule # 1, the minimum squared Euclidean distance between two different paths,

```
path # 1:       A --- B

path # 2:       A --- B
```

```
            at time:k     (k+1)

            has to be 4.


b) transitions emerging from a given state A


        rule 2: when the encoder is in a given state A, the symbols
                he can sent can only be chosen from the EVEN
                family OR (EXCLUSIVE) the ODD family

        rule 3: let the encoder be in state A at time kT, and let
                N1 and N2 be two possible states of the encoder,
                or nodes of the Trellis, at time (k+1)*T.

                We already know, by rule 1, that for the transition
                A ---> N1 all the possible output symbols must
                belong to only one sublattice, and for the transition
                A ---> N2 all the output symbols must belong to only
                one sublattice. Rule # 3 states that the two
                sublattices must be different. And, as a reminder,
                from Rule # 2, these two different sublattices
                must be both EVEN or (exclusive) both ODD.

                From rules # 2 and 3, the minimum squared Euclidean
                distance between two paths,

                path # 1:       A --- N1

                path # 2:       A --- N2

                at time:k     (k+1)

                has to be 2.


c) transitions merging into a given state B

        rule 4: all the paths that merge into a given state B, must
                have the last symbol sent belong either to the EVEN
                family OR (EXCLUSIVE) the ODD family.

        rule 5: let N1 and N2 be two possible states of the encoder
                at time k*T, and let B be a possible state of the
                encoder at time (k+1)*T.

                We already know, by rule 1, that for the transition
                N1 ---> B all the possible output symbols must
                belong to only one sublattice, and for the transition
                N2 ---> B all the output symbols must belong to only
                one sublattice. Rule # 5 states that the two
                sublattices must be different. And, from Rule # 4,
                these two different sublattices must be both EVEN
                or (exclusive) both ODD.

                From rules # 4 and 5, the minimum squared Euclidean
                distance between two paths,

                path # 1:       N1 --- B

                path # 2:       N2 --- B

                at time:k      (k+1)

                has to be 2.

                        ............
```

The following two examples describe the only two cases that can exist when two different paths begin at a common state A and end, some time later, in a common state B. Two paths will be called different if at least one output symbol is different. In the first example, the two paths follow the same sequence of intermediate states. In the second example, the two paths diverge at a given time (they follow different branches) and merge together at a later time. In both cases, by following the above five rules, we end up with a minimum squared Euclidean distance of 4 between the two output sequences.

Example # 1:

For example, assume two possible paths from state A at time 3*T to state B at time 8*T:

```
        path # 1A --- N1 --- N2 --- N3 --- N4 --- B

        path # 2A --- N1 --- N2 --- N3 --- N4 --- B

        at time         3     4      5      6      7      8
```

Following rule # 1 the output symbols could belong, for example,
to the following sublattices:

```
        path # 1   D0      D3      D1      D1      D5

        path # 2   D0      D3      D1      D1      D5

        at time            3       4       5       6       7
```

For the two paths to be different, at least at one sampling time,
say t = 6*T, they should have had different output symbols. And,
since the two symbols belong to the same sublattice D1, the minimum
squared distance between them is 4. Hence, the minimum squared
distance between the two sequences is 4.

Example # 2:

In the previous example we had two different paths going through
the same sequence of encoder states. Now we treat the second
possible case, where at some time k*T the two paths "diverge",
that is they go to different states at time (k+1)*T, and at
some later time, (k+n)*T, n > 1, they converge again.

For example, assume the following two possible paths from state
A at time 3*T to state B at time 8*T:

```
        path # 1A --- N1 --- N2 --- N5 --- N4 --- B

        path # 2A --- N1 --- N3 --- N1 --- N4 --- B

        at time         3     4      5      6      7      8
```

We see that at time t = 4*T the two paths diverge (they go to
different states at time t = 5*T, states N2 and N3) and at time
t = 7*T the two paths converge again into the same state N4.

When the paths diverge at time 4*T, we know, by rules # 2 and 3,
that the ouput symbols must belong to different sublattices
of the same parity. Hence the symbols outputed at time 4*T must
have a minimum squared distance of 2.

When the paths converge again at time 7*T, we know, by rules
4 and 5, that the two last symbols outputted before converging
must have belonged to different sublattices of the same parity.
Hence, the minimum squared distance between the two symbols

outputted at time 6*T must have been at least 2.

Therefore, the two paths that begin at state A of the encoder at time t = 3*T and end later in state B at time t = 8*T, have a guaranteed accumulated minimum squared Euclidean distance between sequences of 2 + 2 = 4.

....

12.- AND WHAT IF WE DO NOT WANT TO USE VITERBI DECODING ?

Suppose we want to reduce the complexity and latency of the receiver, by using symbol-by-symbol decoding, instead of sequence-by-sequence decoding. Do we still can gain anything from this division between X and Y symbols, and the 4D structure we have built defining families and sublattices ?

The answer is: yes.

Without encoding, the minimum squared Euclidean distance between paths would have been only 1, the minimum distance between symbols in the 4D lattice.

But even without using Trellis encoding, we can force the Parity bit to be always zero, Parity = 0, transmitting then always 4D EVEN symbols, whose minimum squared distance between them is 2, corresponding to a coding gain of 3 dB.

....

REFERENCES

1) G. David Forney et al
"Efficient modulation for band-limited channels"
IEEE Journal on Selected Areas in Commun, vol SAC-2, pp 632-647, Sept 1984

2) Gottfried Ungerboeck
"Trellis-coded modulation with redundant signal sets, Part I and II
IEEE Communications Magazine, vol 25, Feb 1987

3) Lee-Fang Wei
"Trellis-coded modulation with multidimensional constellations"
IEEE Trans. on Information Theory, vol IT-33, pp 483-501, July 1987

....