Trellis Coded Modulation

(TCM)

DESPOINA GEORGIADOU

Chania 2008

Outline

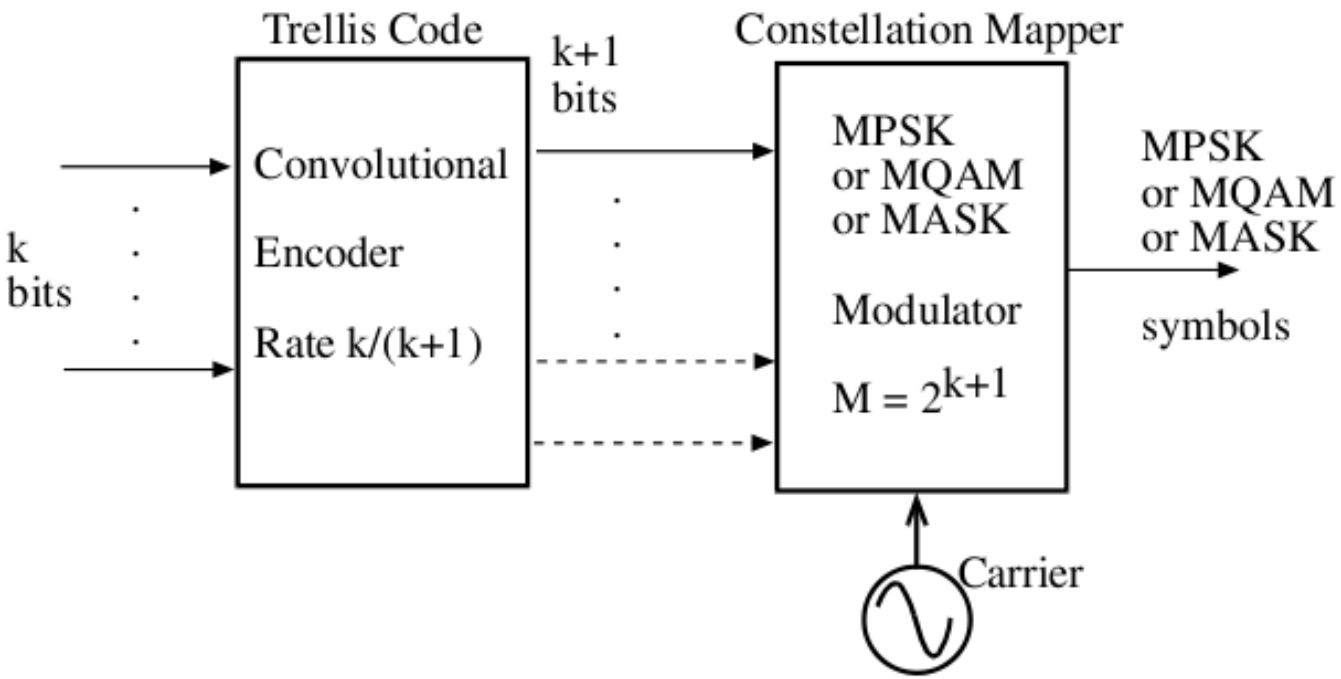
Introduction

Coding is a digital function and **modulation is an analog function**. These functions are done separately and independently in most common modulation schemes.

In Trellis Coded Modulation (TCM), however the two are combined in one function, hence it's name. The word trellis stands for the use of trellis (also called convolutional) codes.

General trellis coded modulation

The functions of a TCM consist of a Trellis code and a constellation mapper as shown in Figure 3. TCM combines the functions of a convolutional coder of rate R = k / k + 1 and a M-ary signal mapper that maps $M = 2^k$ input points into a larger constellation of $M = 2^{k+1}$ constellation points.



A general trellis coded modulation

General information about TCM

- TCM is a bandwidth efficient modulation based on convolutional coding.
- It conserves bandwidth by doubling the number of constellation points of the signal. This way the bit rate increases but the symbol rate stays the same.
- Unlike a true Convolutional code, not all incoming bits are coded and only 1 extra bit is always added.
- Increasing the constellation size reduces Euclidean distances between the constellation points but sequence coding offers a coding gain that overcomes the power disadvantage of going to the higher constellation.
- The decoding metric is the Euclidean distance and not the Hamming distance.
- TCM uses set-partitioning and small number of states.

TCM CODE CONSTRUCTION

The main steps in designing a TCM system are:

1. Signal set selection
2. Labeling of the signal set
3. Code selection

Specifically:

- We start with a given bandwidth B from which we determine the maximum possible symbol rate (never more than 2B).
- Then we determine the size of the alphabet that can deliver the needed signal BER at the given available power.
- Last, we assign binary labels, representing encoder output blocks, to the signal points in such a way that squared Euclidean distance d is maximized (set partitioning).

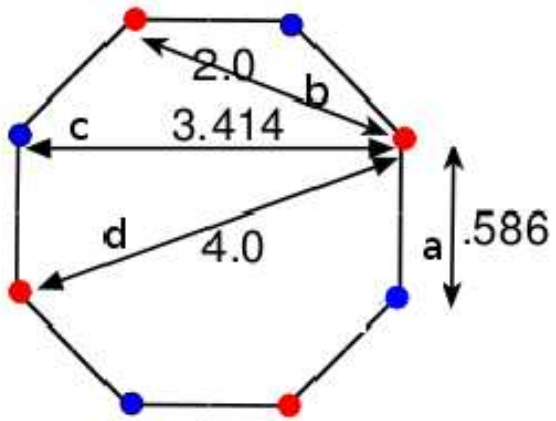
Mapping by set partitioning

- We can map k information bits to 2^{k+1} constellation points such that the signals get further apart increasing the Euclidean distance between the signals in that set.

- This mapping follows from successive partitioning of a channel-signal set into subsets with increasing minimum distances $d_0 < d_1 < d_2$... between the signals of these subsets.

Mapping by set partitioning - Example

Consider 8-PSK channel signals a_n with $E|a_n^2| = 1$.



$$a=[1/\sqrt{2}]^2+[1-(1/\sqrt{2})]^2=.586$$
$$b=1^2+1^2=2$$
$$c=[1/\sqrt{2}]^2+[1+(1/\sqrt{2})]^2=3.414$$
$$d=2^2=4$$

1

- The 8 points are successively partioned into disjoint cosets such that the SEDs are increasing at each level.
- There are total of four partitions counting the first unpartitioned set. At top-most level, the SED is 0.586.
- At the next level, where there are only four points in each of the two cosets, the SED has increased to 2.0 and at the last level, the SED is 4.0.

[1]The distances above are Squared Euclidean and came up from the geometry of the constellation.

Mapping by set partitioning - Example

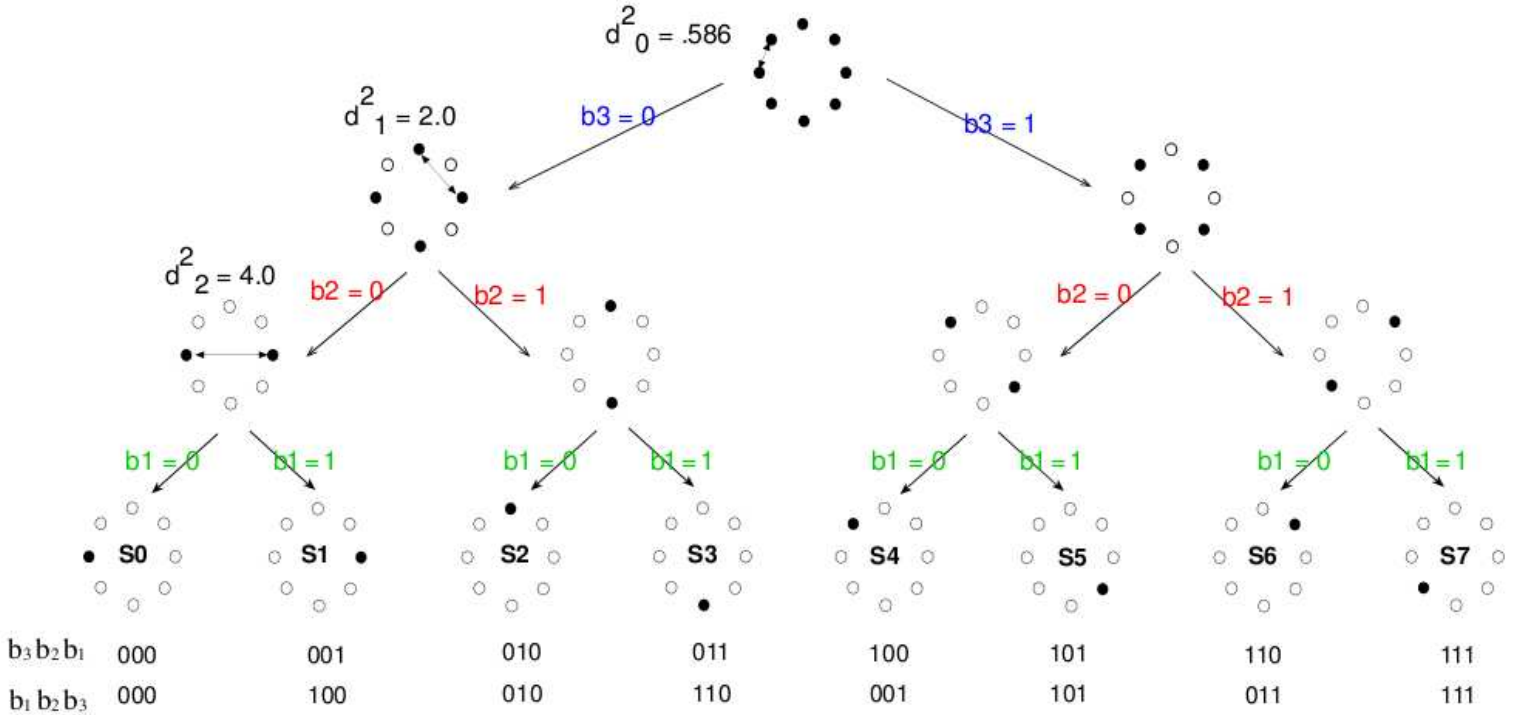
Explanation

There are 3 incoming bits from the code:

1. b1: the uncoded bit
2. b2, b3: the coded bits

- Since the top levels have smaller distances and the errors at this level are more likely, we will use the coded bits to traverse through this part.
- So, with b3 and b2 we decide which partition to choose and then we can use the uncoded bit at the last level to pick the signal transmitted.
- Two symbols at last level that differ only at bit b1 have a large Euclidean distance and would require an error of 180 degrees to be corrupted.

Mapping by set partitioning - Example Result



- There is no partition of 8-PSK into two equal-size subsets that achieves a larger SED.
- Each subset is partitioned into two isomorphic subsets (the one is obtained by rotating the other by x^o). (the second property is not always feasible!)

Multi-dimensional TCM

Generally about multi-dimensional TCM

We are about to consider the more general case of designing TCM systems that use multidimensional signal sets. Let L the dimension of modulations (for 8-PSK L=2). With L = 1, we transmit just one TCM symbol. With L = 2, we transmit 2 symbols, so that the number of symbols transmitted is equal to L.

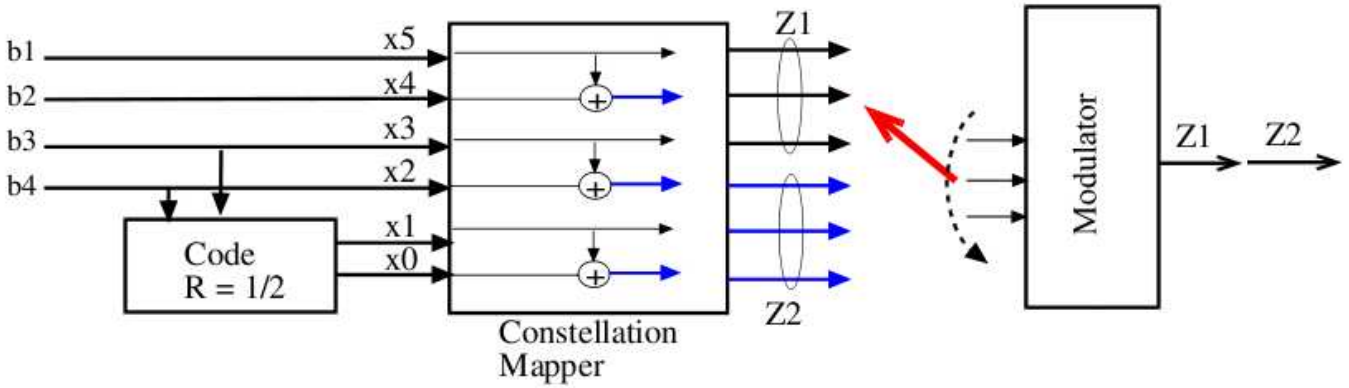
The main concept in multi-dimensionality is increasing the number of symbols created in one processing period. The transmitted symbols are generated together and this co-generation creates dependence and allows better performance.

Multi-dimensional TCM

The main advantages of multi dimensional TCM are:

1. We can transmit fractional information rates. Instead of the effective code rate being 2/3 as it is in 1 x 8PSK, here it can be higher. We can reduce the code overhead by effecting more than one symbol so we can use code rates like 5/6, 8/9 and 11/12.
2. Better bit efficiency is possible. We define bit efficiency as number of input information bits divided by the number of symbols transmitted in one processing period.
3. No additional hardware complexity, we can use standard rate 1/2 codes.

Multi-dimensional TCM

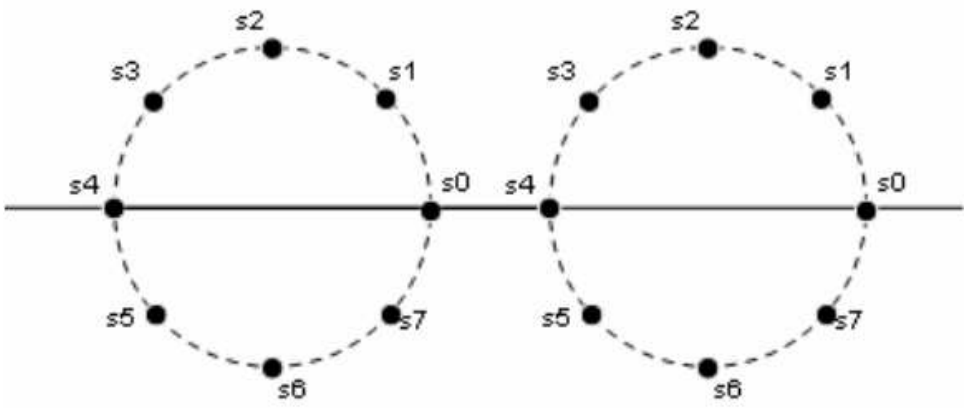


Example of generating a 2 x 8PSK signal.

- 4 bits come in and 2 of these go into a rate 1/2 encoder and generate 2 parity bits, for a total of 6 bits.
- These six bits are then mapped in a special way by the constellation mapper.
- It is this mapping function that creates the symbol inter-dependency.

Partitioning in multi-dimensional TCM

In our example we have two 8PSK symbol producers:



If we transmit a pair of symbols, then there are total 64 possible pairs: $= S^L = S*S$ [2], L=2

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77

[2] S = {0,1,2,3,4,5,6,7}

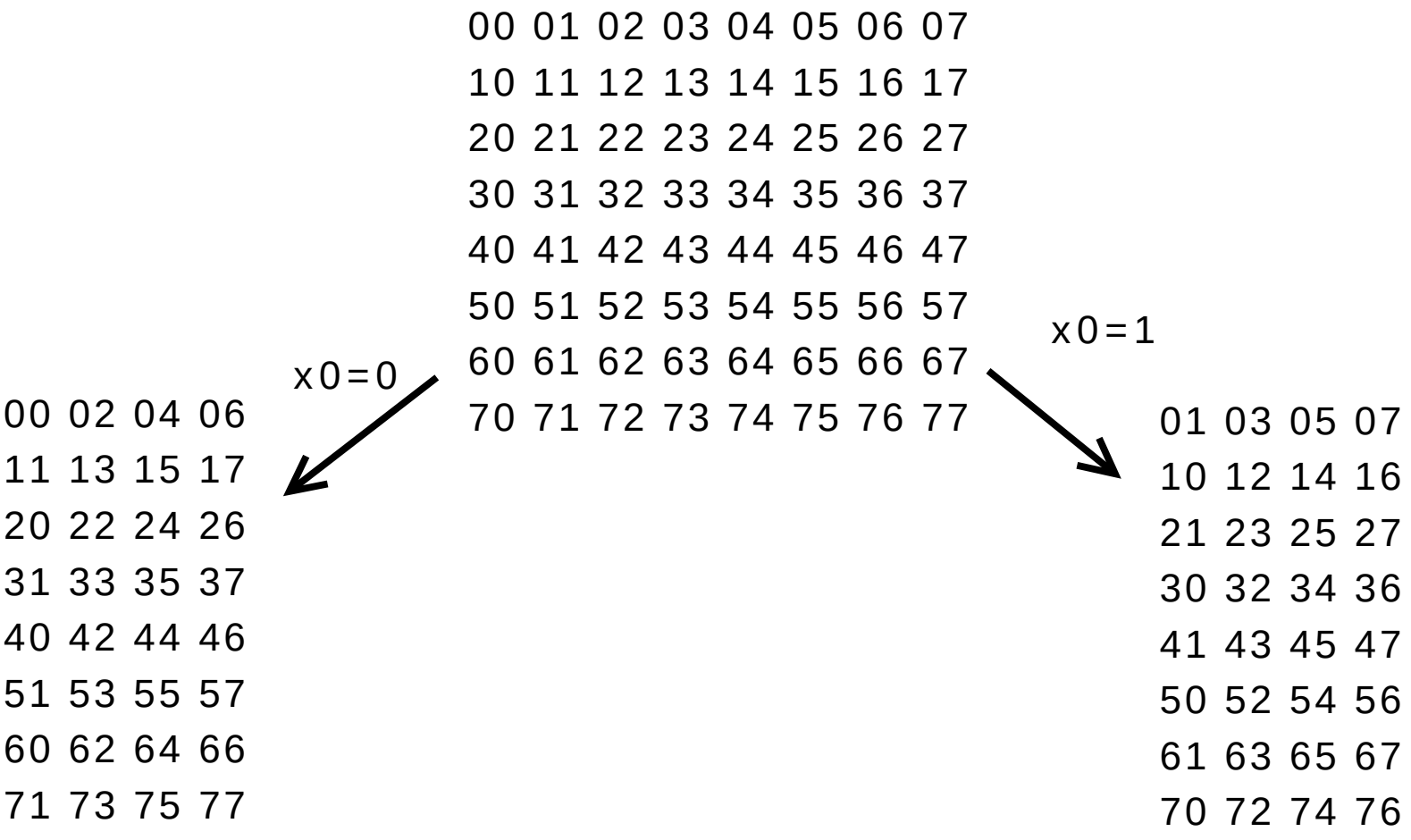
Partitioning in multi-dimensional TCM

Partitioning top level

The previous table is the top level of our partitioning. Now we partition the top level into two cosets by this way: starting from 00 choose the neighbor points that the SED between them is the biggest possible

$$coset0 = \begin{matrix} 00 & 02 & 04 & 06 \\ 11 & 13 & 15 & 17 \\ 20 & 22 & 24 & 26 \\ 31 & 33 & 35 & 37 \\ 40 & 42 & 44 & 46 \\ 51 & 53 & 55 & 57 \\ 60 & 62 & 64 & 66 \\ 71 & 73 & 75 & 77 \end{matrix} \quad coset1 = \begin{matrix} 01 & 03 & 05 & 07 \\ 10 & 12 & 14 & 16 \\ 21 & 23 & 25 & 27 \\ 30 & 32 & 34 & 36 \\ 41 & 43 & 45 & 47 \\ 50 & 52 & 54 & 56 \\ 61 & 63 & 65 & 67 \\ 70 & 72 & 74 & 76 \end{matrix}$$

Mapping by set partitioning - Top level partition



Partitioning in multi-dimensional TCM

Partitioning level-1 and level-2

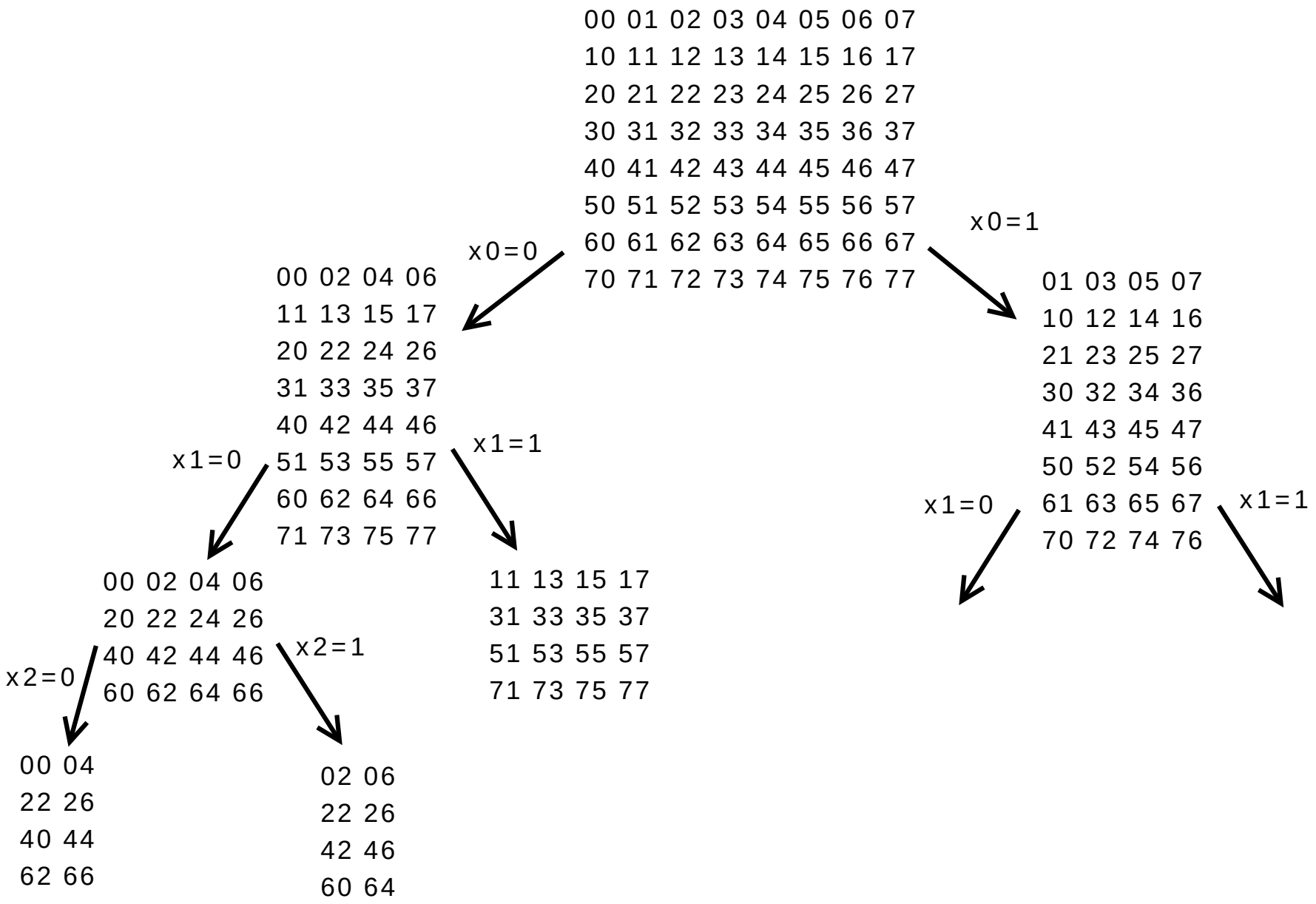
Now we partition the 2 cosets of level-1 into four cosets. As an example, the coset0 is partitioned into:

$$coset00 = \begin{matrix} 00 & 02 & 04 & 06 \\ 20 & 22 & 24 & 26 \\ 40 & 42 & 44 & 46 \\ 60 & 62 & 64 & 66 \end{matrix} \quad coset01 = \begin{matrix} 11 & 13 & 15 & 17 \\ 31 & 33 & 35 & 37 \\ 51 & 53 & 55 & 57 \\ 71 & 73 & 75 & 77 \end{matrix}$$

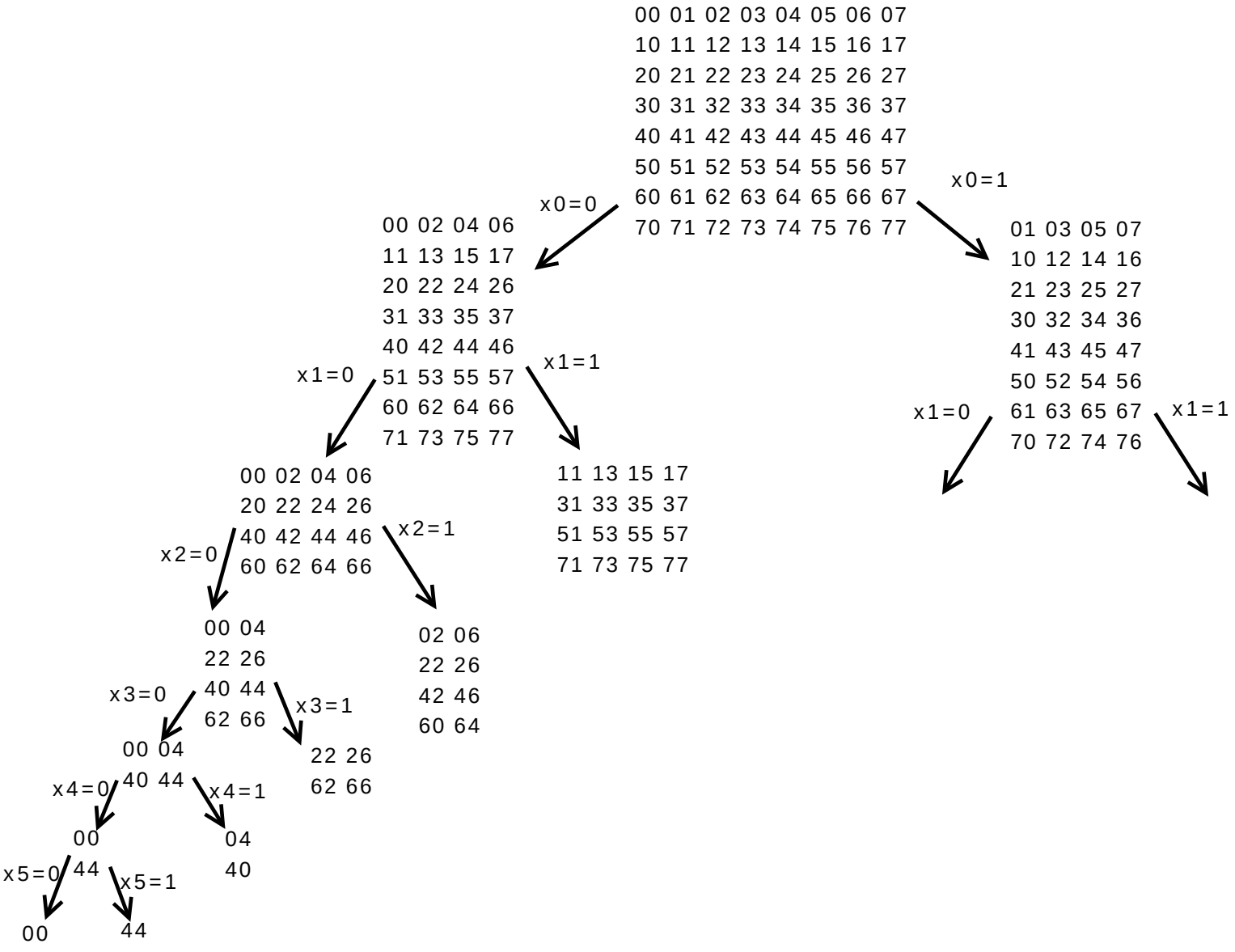
and coset00 into:

$$coset000 = \begin{matrix} 00 & 04 \\ 22 & 26 \\ 40 & 44 \\ 62 & 66 \end{matrix} \text{ and } coset001 = \begin{matrix} 02 & 06 \\ 20 & 24 \\ 42 & 46 \\ 60 & 64 \end{matrix}$$

Mapping by set partitioning - 2 and 3 Level partition



The partitioning goes on until all single pairs consist a new coset.



Partitioning in multi-dimensional TCM

Coset Generators

Each 2 cosets below the same coset can be generated by having only the one of them. The other is generated by adding a signal pair g_i to each member of the coset we have.

Example

coset0 + g_0 = coset1, where g_0=01=
$$\begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{matrix}$$

$$coset0 + 01 = \begin{matrix} 00 & 02 & 04 & 06 \\ 11 & 13 & 15 & 17 \\ 20 & 22 & 24 & 26 \\ 31 & 33 & 35 & 37 \\ 40 & 42 & 44 & 46 \\ 51 & 53 & 55 & 57 \\ 60 & 62 & 64 & 66 \\ 71 & 73 & 75 & 77 \end{matrix} + 01 = \begin{matrix} 01 & 03 & 05 & 07 \\ 10 & 12 & 14 & 16 \\ 21 & 23 & 25 & 27 \\ 30 & 32 & 34 & 36 \\ 41 & 43 & 45 & 47 \\ 50 & 52 & 54 & 56 \\ 61 & 63 & 65 & 67 \\ 70 & 72 & 74 & 76 \end{matrix} = \text{coset1}$$

Partitioning in multi-dimensional TCM

All the coset generators are:

g_0 = (0,1)=[(000)',(001)'] g_1 = (1,1)=[(001)',(001)']
g_2 = (0,2)=[(000)',(010)'] g_3 = (2,2)=[(010)',(010)']
g_4 = (0,4)=[(000)',(100)'] g_5 = (4,4)=[(100)',(100)']

The coset generators are more useful than you may thought:

Take for example an incoming vector x = $(x_5, x_4, x_3, x_2, x_1, x_0)$ = (0,1,0,1,0,1)
We multiply each bit by its coset generator to determine which coset it falls in, let Y.

$Y = x_5 * g_5 + x_4 * g_4 + x_3 * g_3 + x_2 * g_2 + x_1 * g_1 + x_0 * g_0 = x_5*(4,4)+x_4*(0,4)+x_3*(2,2)+x_2*(0,2)+x_1*(1,1)+x_0*(0,1)$ = (0,7) = $[(000)^T, (111)^T]$=$[Y_1, Y_2]$

Partitioning in multi-dimensional TCM

For x = $(x_5, x_4, x_3, x_2, x_1, x_0)$ = (1,0,0,0,0,0) we get:

Y=$x_5 * (4,4) + x_4 * (0,4) + x_3 * (2,2) + x_2 * (0,2) + x_1 * (1,1) + x_0 * (0,1)$=(4,4)=[$(100)^T, (100)^T$]