SZÉCHENYI ISTVÁN
EGYETEM
GYŐR

KÓDOLÁSELMÉLET

Nagy Szilvia

3. Konvolúciós kódolás

2009.

Keret, kódszókeret, kódsebesség

A konvolúciós kódolás során a tömörített $b_1, b_2, b_3, \ldots$ üzenetet k bites szakaszokra – üzenetszegmensekre – bontjuk, és $m+1$ egymás utáni üzenetszegmensből alakítjuk ki a kódoló aktuális kimenetét.

A kódoló mindig m darab k hosszúságú üzenetszegmenst – **keret**et – tárol, és egy van a bemenetén.

Keret, kódszókeret, kódsebesség

Egy lépés során a kódoló

- beolvas egy üzenetszegmenst
- az $m+1$ darab, k hosszúságú bitsorozatból létrehoz egy n hosszúságú kimeneti bitsorozatot, a **kódszókeret**et.
  A **kódsebesség**: $R = k/n$.
- eldobja a legrégebben tárolt keretet és elraktározza az újonnan beolvasottat

Egy üzenetszegmens $m+1$ lépés során befolyásolja a kimenetet, utána tűnik csak el a tárolókból. A kódoló **kényszerhossz**a $K = k\cdot(m+1)$ bit.

Kényszerhossz, fa-kód, trellis

Ugyanez az üzenetszegmens a kimeneten $n\cdot(m+1)$ bit kialakításában vesz részt, az $N = n\cdot(m+1)$ mennyiség a kódoló **blokkhossz**a.

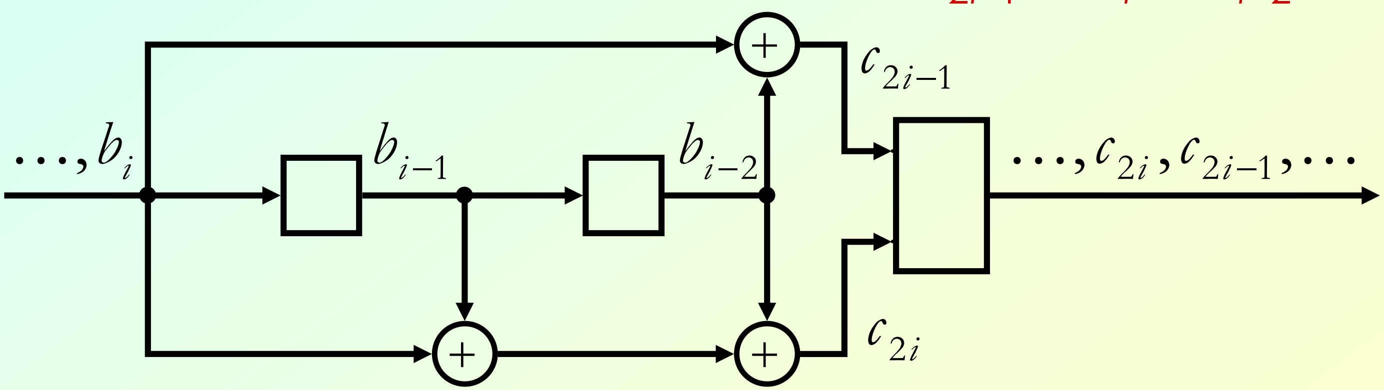
A K kényszerhosszú, N blokkhosszú, lineáris, időinvariáns trellis kódokat **(*N*, *K*) paraméterű konvolúciós kód**oknak nevezzük.

- A konvolúciós kódok tervezése sok esetben lehetőséget nyújt a moduláció tervezésére is.

Kódparaméterek

- A konvolúciós kódolók létrehozhatók léptetőregiszterekkel, például: k=1, n=2, m=2, K=3, N=6-ra

$$c_{2i-1} = b_i + b_{i-2}$$

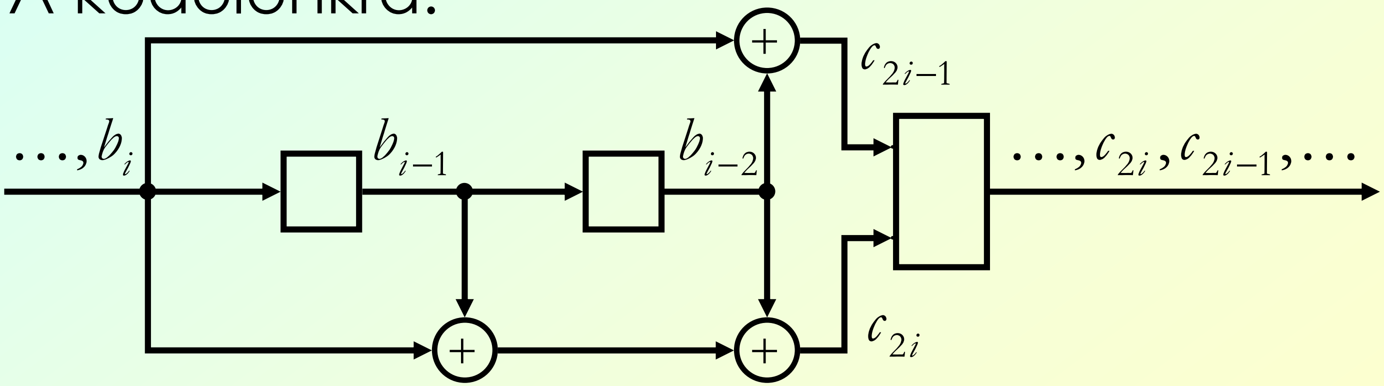


$$c_{2i-1} = b_i + b_{i-1} + b_{i-2}$$

Konvolúció

Jellemezzük a konvolúciós kódoló *p*-edik ágát egy olyan félig végtelen bitsorozattal, amelyiknek a *j*-edik eleme akkor és csak akkor 1, ha a *p*-edik ágban 1-es együtthatóval jelenik meg b_{i-j} .

A kódolónkra:



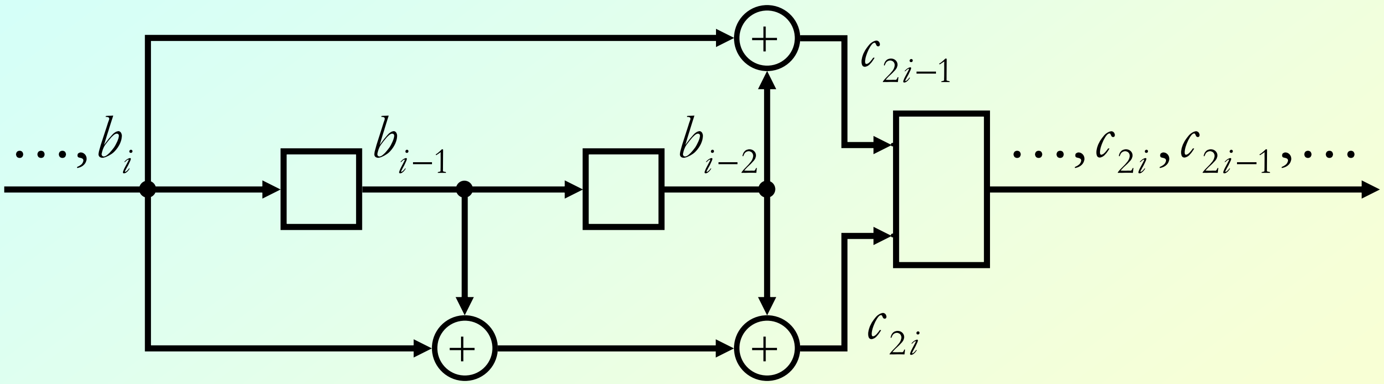
az első ág bitsorozata: g_1=1 0 1 0 0 0 …

a második ág bitsorozata: g_2=1 1 1 0 0 0 …

Konvolúció

az első ág bitsorozata: g_1=1 0 1 0 0 0 ...

a második ág bitsorozata: g_2=1 1 1 0 0 0 ...



Az első ág kimenete:

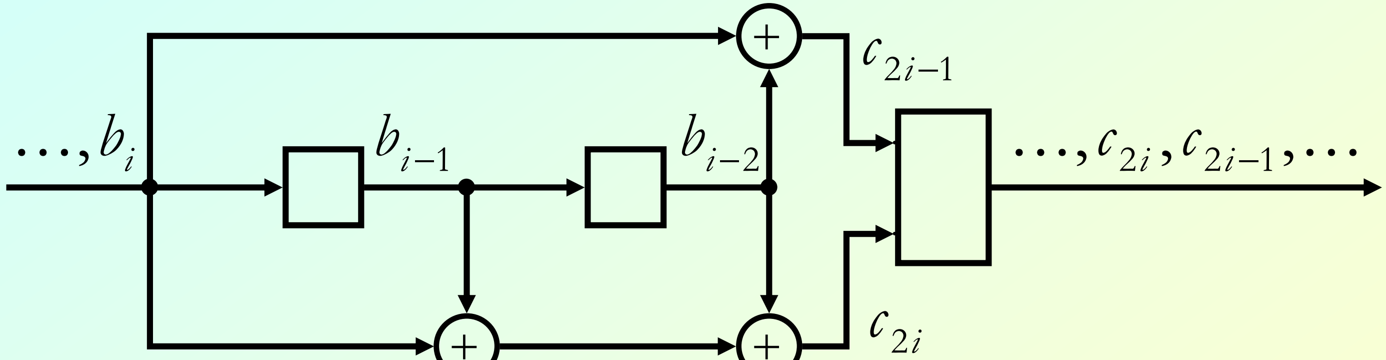
$b_1,\ b_2,\ b_3+b_1,\ b_4+b_2,\ b_5+b_3,\ \ldots=$
$= g_1 * b$

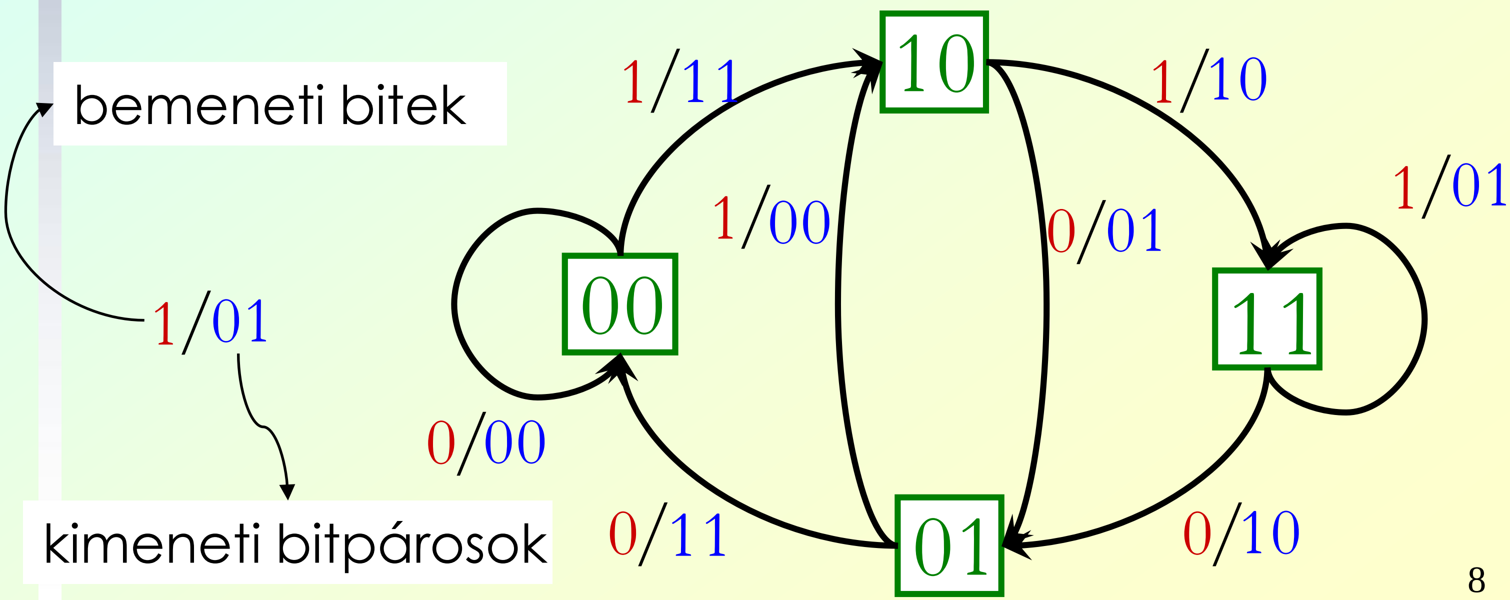
A második ág kimenete:

$b_1,\ b_2+b_1,\ b_1+b_2+b_3,\ b_2+b_3+b_4,\ b_3+b_4+b_5, \ldots=$
$= g_2 * b$

Állapotátmenet-gráf

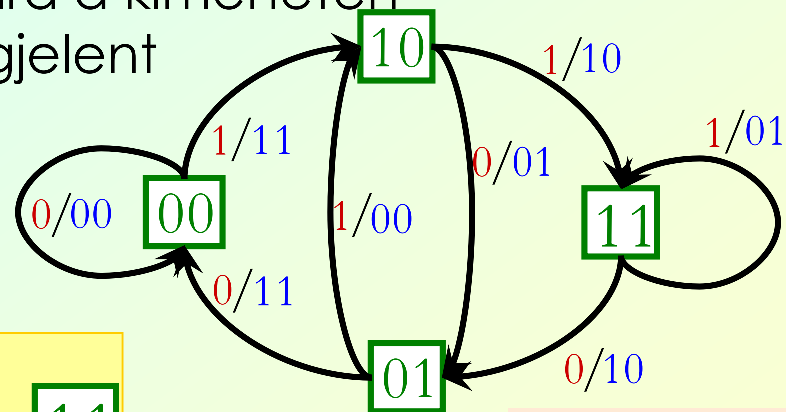
Nézzük a kódolónk tárolóinak állapotai közötti átmeneteket gráfon ábrázolva:



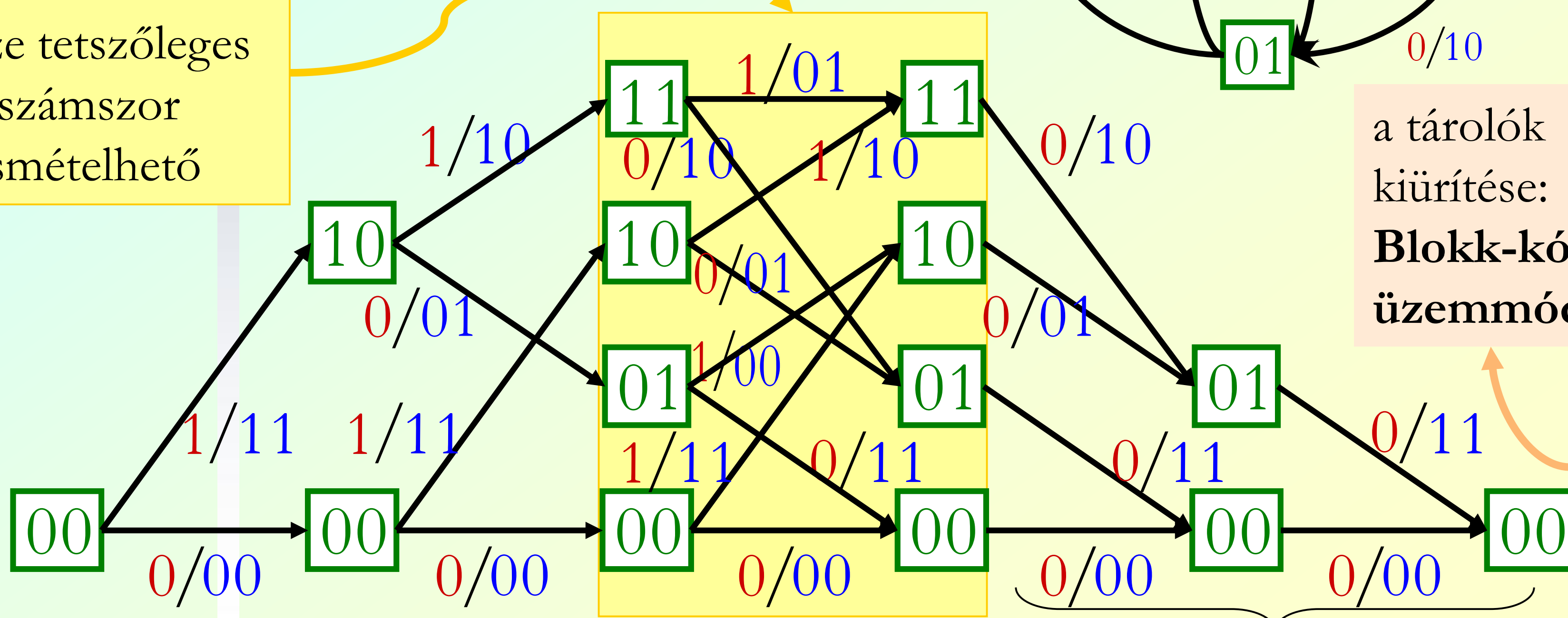


Trellis diagram

Az állapotátmenet-gráfról, vagy az áramkör blokkvázlatáról leolvashatók a különféle bemeneti kombinációk hatására a kimeneten és a tárolókban megjelent bitek, melyeket a **trellisen** lehet ábrázolni:



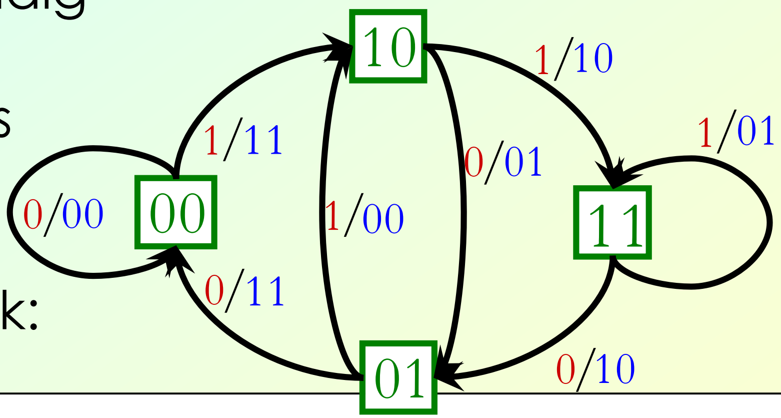
a trellisnek ezen része tetszőleges számszor ismételhető

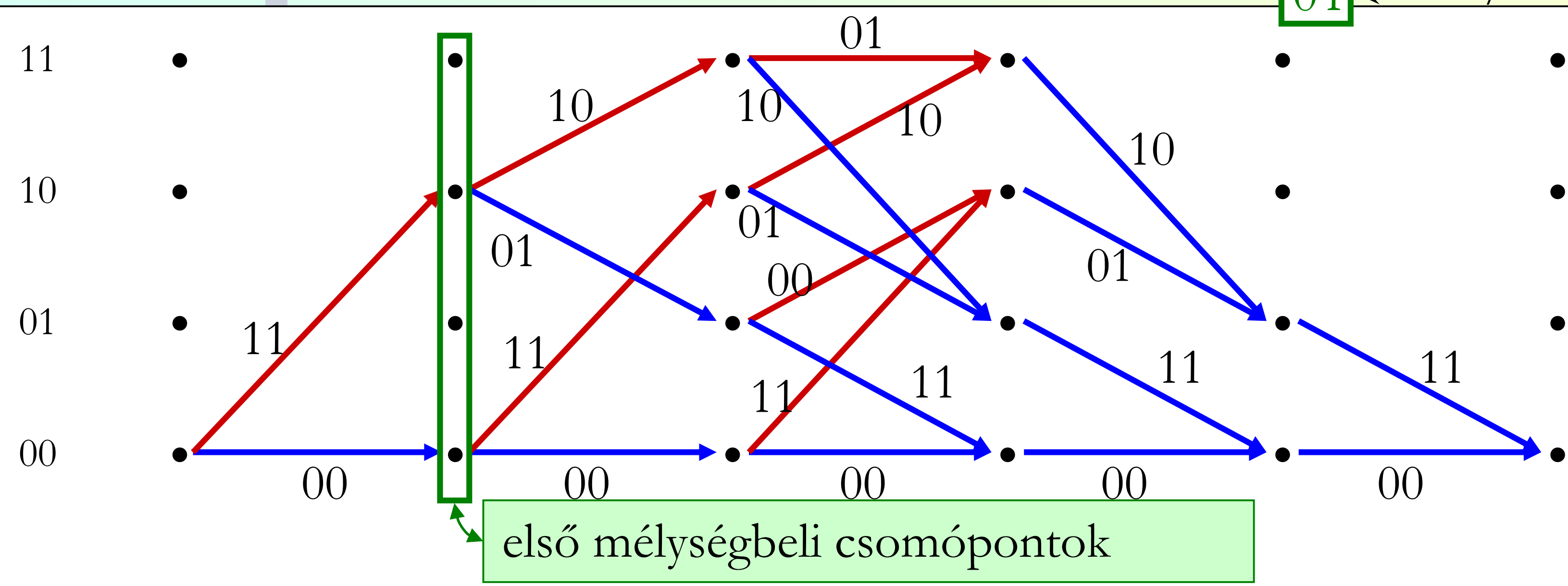


a tárolók kiürítése: **Blokk-kód üzemmód**

Trellis diagram

Az **egyszerűsített trellis**en az azonos tárolóállapotok, mint egy sorban lévő pontok szerepelnek, és a bemenetet sem mindig tüntetik fel az élekre (a felfelé menő piros élek az 1 bemeneti bitet, a lefelé muta-tó kékek a 0-t jelentik:

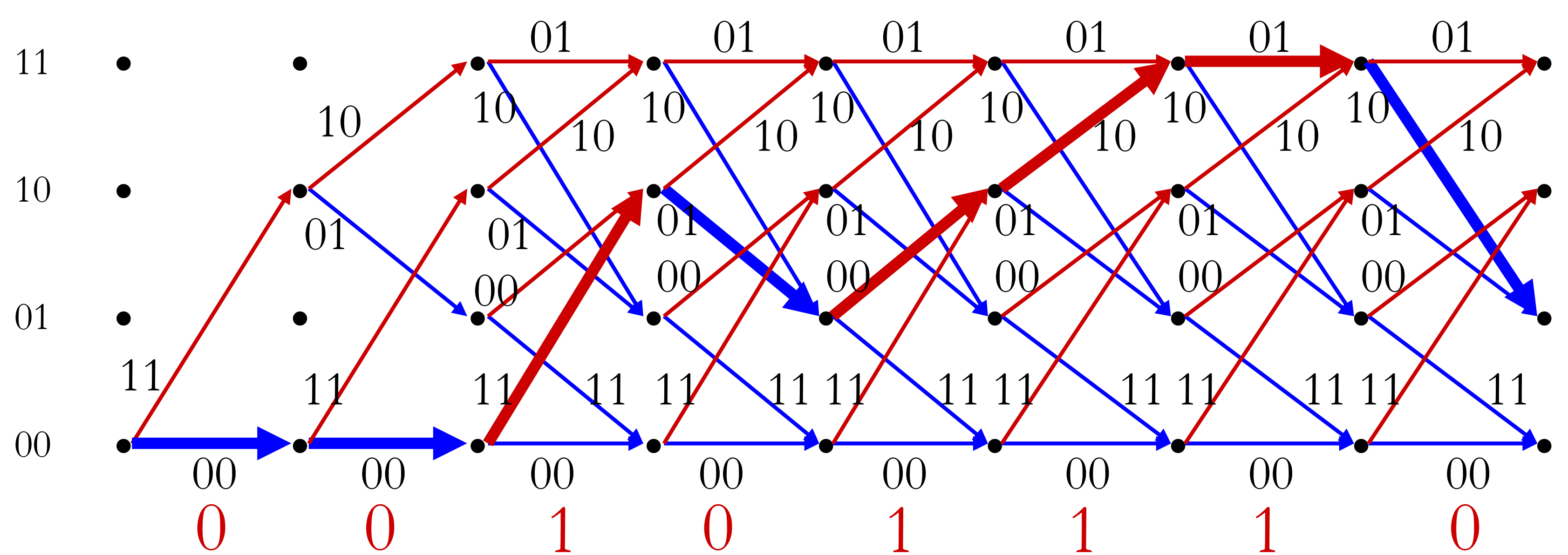




Trellis diagram

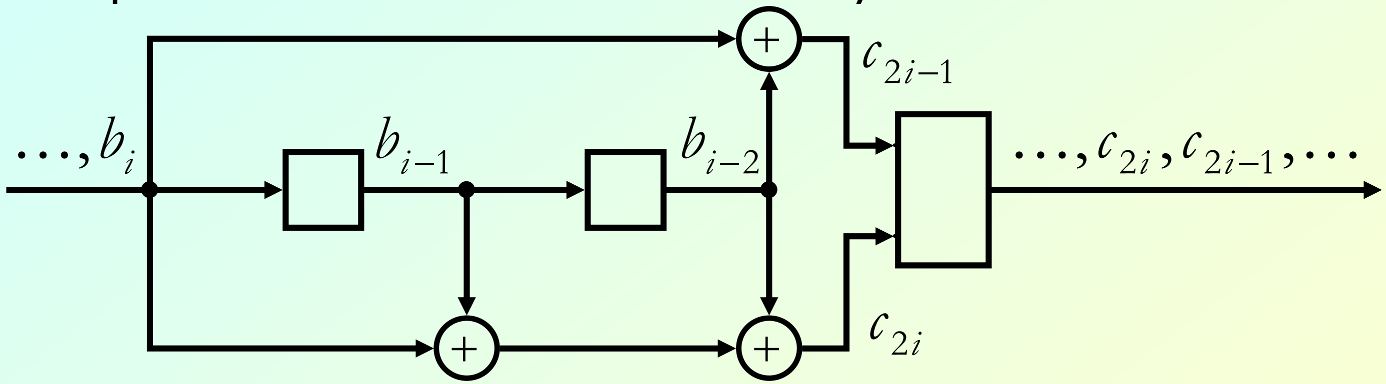
Az egyszerűsített trellisen a különböző üzenetek kódolását lehet nyomon követni: Legyen a kódolandó üzenetszakasz **0 0 1 0 1 1 1 0**

(trellis – rácsozat)



Polinom reprezentáció

Az áramkörünk tulajdonképpen két polinomszorzó eredményeit fésüli össze:



A bináris polinomjaink:

$$g_{11}(t) = 1 + t^2$$

$$g_{12}(t) = 1 + t + t^2$$

Az üzenethez rendelhető polinom:

$$b_1(t) = b_1 + b_2 t + b_3 t^2 + b_4 t^3 + b_5 t^4 + \ldots$$

Polinom reprezentáció

Az ágakat jellemző bináris polinomok:

$$g_{11}(t) = 1 + t^2$$

$$g_{12}(t) = 1 + t + t^2$$

Az üzenethez rendelhető polinom:

$$b_1(t) = b_1 + b_2 t + b_3 t^2 + b_4 t^3 + b_5 t^4 + \dots$$

Az kimenet szétosztható 2 darab külön bitfolyamra, melyekhez a következő polinomok rendelhetők:

$$c_1(t) = c_1 + c_3 t + c_5 t^2 + c_7 t^3 + c_9 t^4 + \dots$$

$$c_2(t) = c_2 + c_4 t + c_6 t^2 + c_8 t^3 + c_{10} t^4 + \dots$$

Polinom reprezentáció

Ha hosszabb az üzenetkeret, az üzenetet is *k* darab különálló bitfolyamra kell bontani, és a generáló polinomoknak is két fontos indexe lesz. Ezek a polinomok mátrixba rendezhetők:

$$\mathbf{G}(t)=\begin{pmatrix} g_{11}(t) & g_{12}(t) & \cdots & g_{1n}(t) \\ g_{21}(t) & g_{22}(t) & \cdots & g_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1}(t) & g_{k2}(t) & \cdots & g_{kn}(t) \end{pmatrix}$$

A mátrix első sora jellemzi az első bemeneti bit hatását az n darab kimenetre, a második sor a második bemeneti bitét, …

Polinom reprezentáció

A kódoló polinom-mátrixa:

$$\mathbf{G}(t)=\begin{pmatrix} g_{11}(t) & g_{12}(t) & \dots & g_{1n}(t) \\ g_{21}(t) & g_{22}(t) & \dots & g_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1}(t) & g_{k2}(t) & \dots & g_{kn}(t) \end{pmatrix}$$

Az üzenethez és a kimenethez rendelhető polinomok vektorokba rendezhetők:

$$\mathbf{b}(t)=\begin{pmatrix} b_1(t) & b_2(t) & \dots & b_k(t) \end{pmatrix}$$

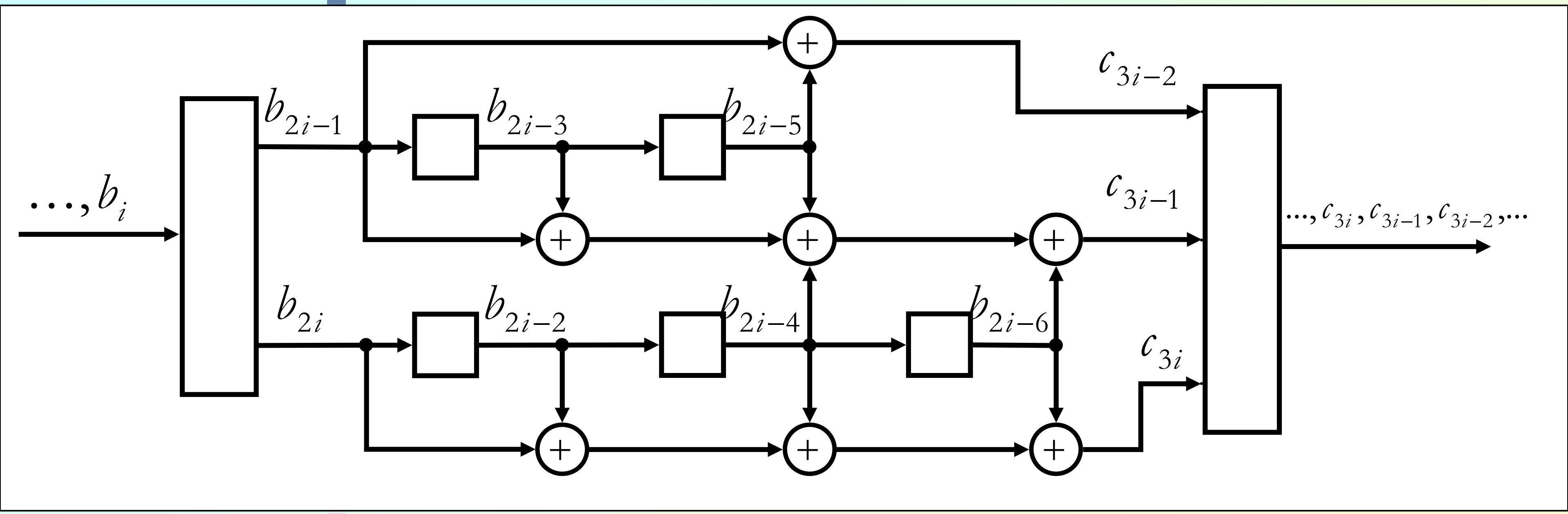
$$\mathbf{c}(t)=\begin{pmatrix} c_1(t) & c_2(t) & \dots & c_n(t) \end{pmatrix}$$

Így

$$\mathbf{c}(t)=\mathbf{b}(t)\cdot\mathbf{G}(t)$$

Konvolúciós kódok

Példa: Nézzük a következő kódoló áramkört:

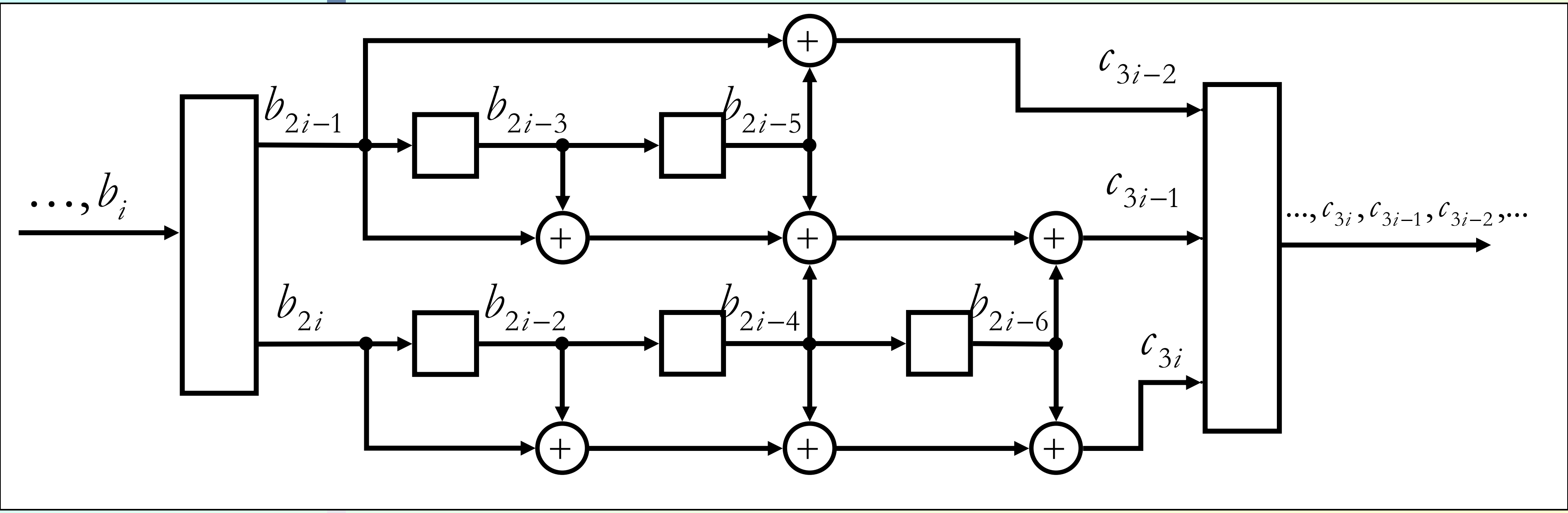


A kódoló paraméterei: k=2, n=3, m=3, K=8, N=12.

A tárolóknak összesen $2^5 = 32$-féle állapota lehetséges, az állapotátmeneti gráf 32 csúccsal rendelkezik, a trellis 32 sorral.

Konvolúciós kódok

<u>Példa</u>: Nézzük a következő kódoló áramkört:



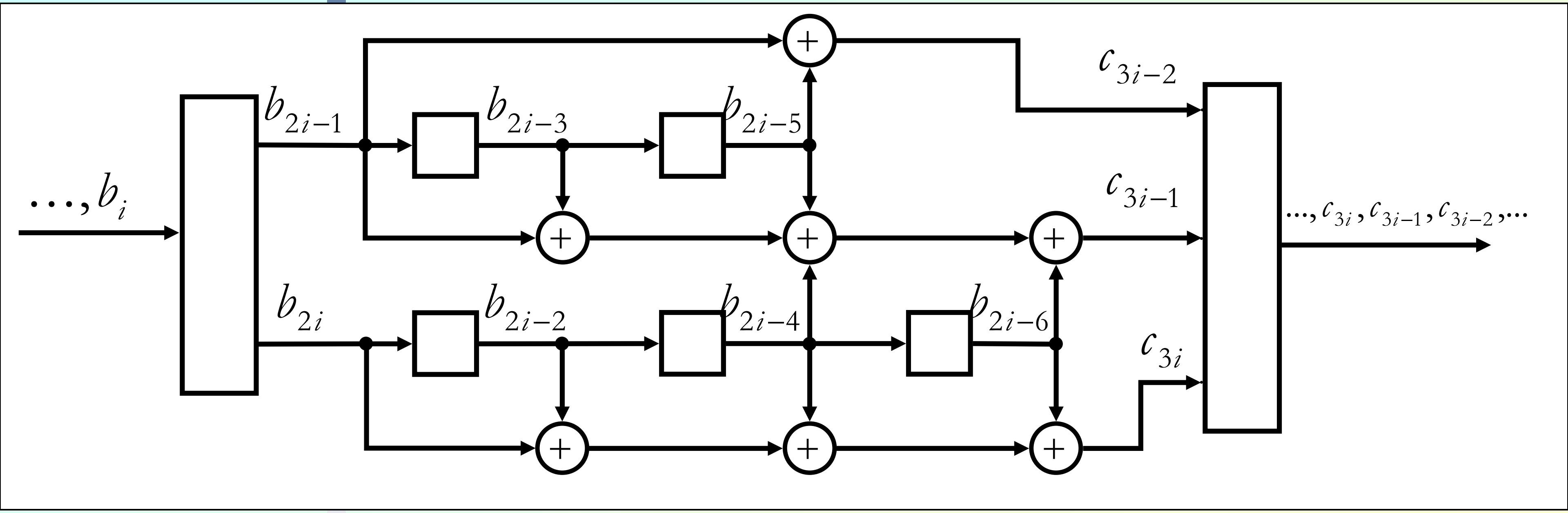
A generáló polinomok:

$$g_{11}(t) = 1 + t^2 \quad g_{12}(t) = 1 + t + t^2 \quad g_{13}(t) = 0$$

$$g_{21}(t) = 0 \quad g_{22}(t) = t^2 + t^3 \quad g_{23}(t) = 1 + t + t^2 + t^3$$

Konvolúciós kódok

Példa: Nézzük a következő kódoló áramkört:

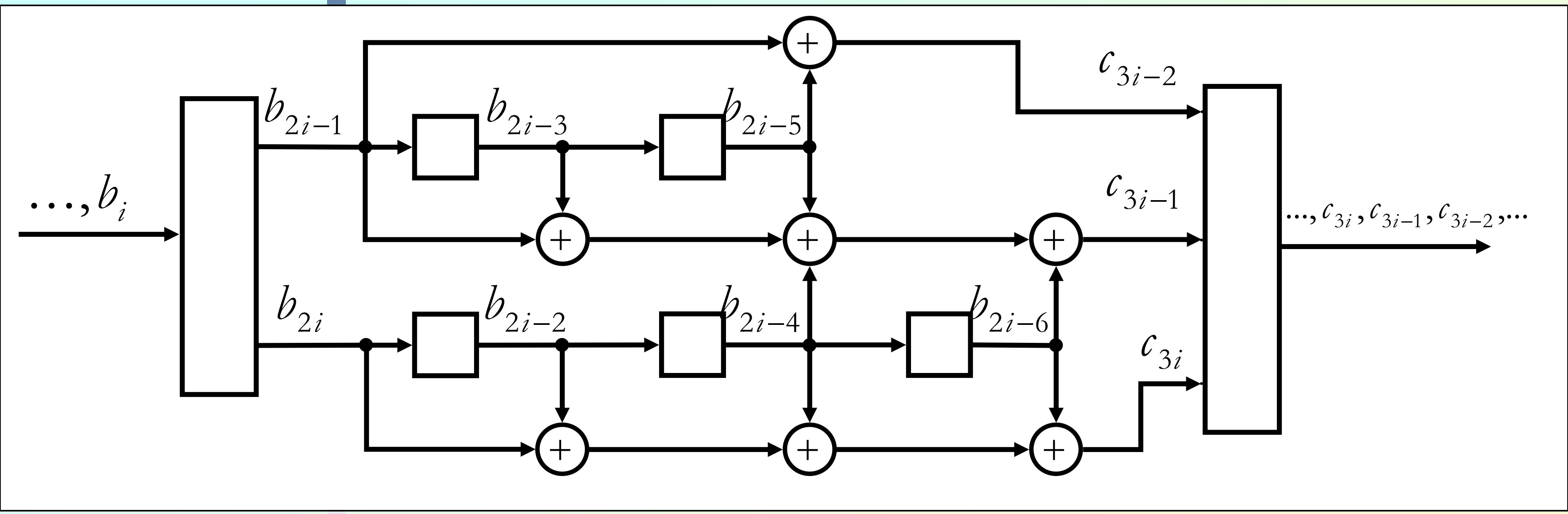


A generáló polinom-mátrix:

$$\mathbf{G}(t)=\begin{pmatrix} g_{11}(t) & g_{12}(t) & g_{13}(t) \\ g_{21}(t) & g_{22}(t) & g_{23}(t) \end{pmatrix}=\begin{pmatrix} 1+t^2 & 1+t+t^2 & 0 \\ 0 & t^2+t^3 & 1+t+t^2+t^3 \end{pmatrix}$$

Konvolúciós kódok

Példa: Nézzük a következő kódoló áramkört:



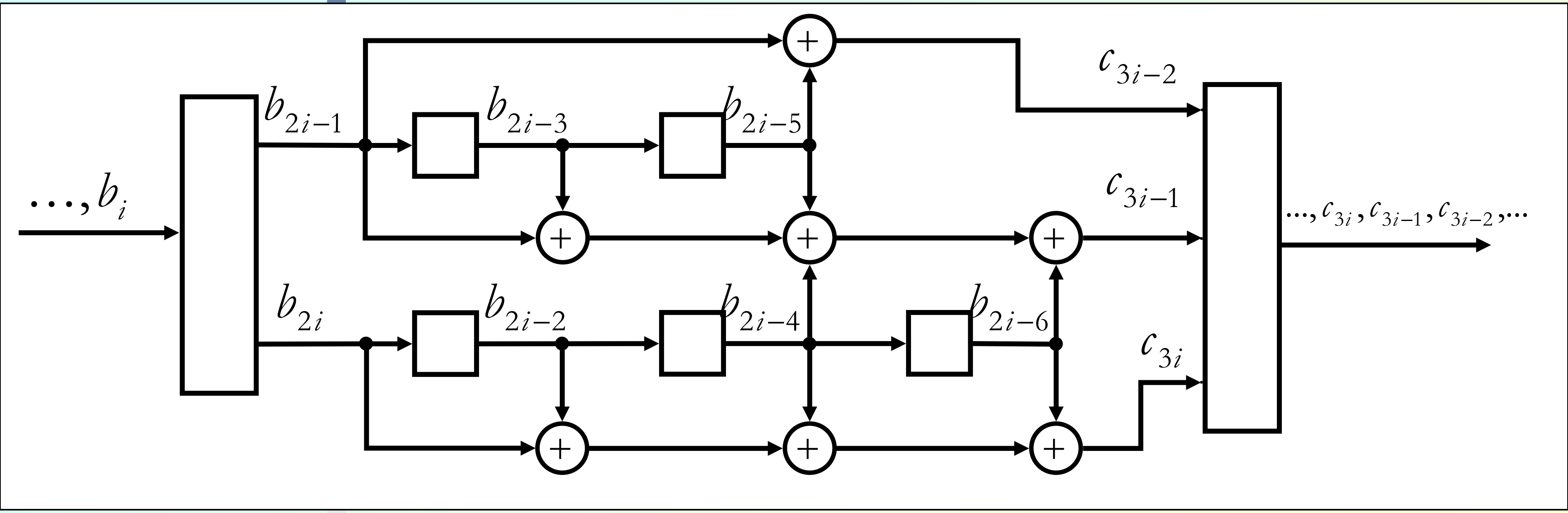
Az üzenet és kimenet felbontása:

$$b_1(t) = b_1 + b_3 t + b_5 t^2 + b_7 t^3 + \ldots$$
$$b_2(t) = b_2 + b_4 t + b_6 t^2 + b_8 t^3 + \ldots$$

$$c_1(t) = c_1 + c_4 t + c_7 t^2 + c_{10} t^3 + \ldots$$
$$c_2(t) = c_2 + c_5 t + c_8 t^2 + c_{11} t^3 + \ldots$$
$$c_3(t) = c_3 + c_6 t + c_9 t^2 + c_{12} t^3 + \ldots$$

Konvolúciós kódok

Példa: Nézzük a következő kódoló áramkört:



Az üzenethez két-, a kimenethez három komponensű polinom-vektor tartozik:

$$\mathbf{b}(t) = \begin{pmatrix} b_1(t) & b_2(t) \end{pmatrix}$$

$$\mathbf{c}(t) = \begin{pmatrix} c_1(t) & c_2(t) & c_3(t) \end{pmatrix}$$

Konvolúciós kódok

<u>Példa</u>: A polinom-mátrix:

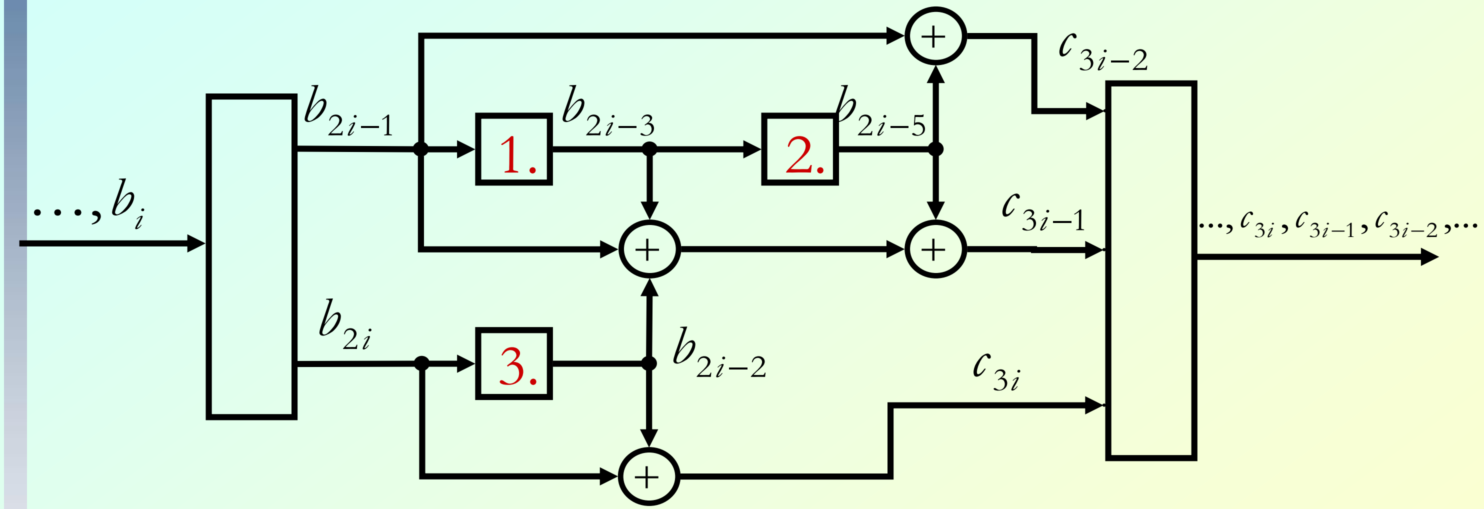
$$\mathbf{G}(t)=\begin{pmatrix} 1+t^2 & 1+t+t^2 & 0 \\ 0 & t^2+t^3 & 1+t+t^2+t^3 \end{pmatrix}$$

A kód komponensei:

$$c_1(t)=b_1(t)\cdot g_{11}(t)+b_2(t)\cdot g_{21}(t)=$$
$$=\left(1+t^2\right)\cdot\left(b_1+b_3t+b_5t^2+b_7t^3+\ldots\right)$$
$$c_2(t)=b_1(t)\cdot g_{12}(t)+b_2(t)\cdot g_{22}(t)=$$
$$=\left(1+t+t^2\right)\cdot\left(b_1+b_3t+b_5t^2+b_7t^3+\ldots\right)+$$
$$+\left(t^2+t^3\right)\cdot\left(b_2+b_4t+b_6t^2+b_8t^3+\ldots\right)$$
$$c_3(t)=b_1(t)\cdot g_{13}(t)+b_2(t)\cdot g_{23}(t)=$$
$$=\left(1+t+t^2+t^3\right)\cdot\left(b_2+b_4t+b_6t^2+b_8t^3+\ldots\right)$$

Konvolúciós kódok

Példa: Nézzük a következő, kicsit egyszerűbb kódoló áramkört:

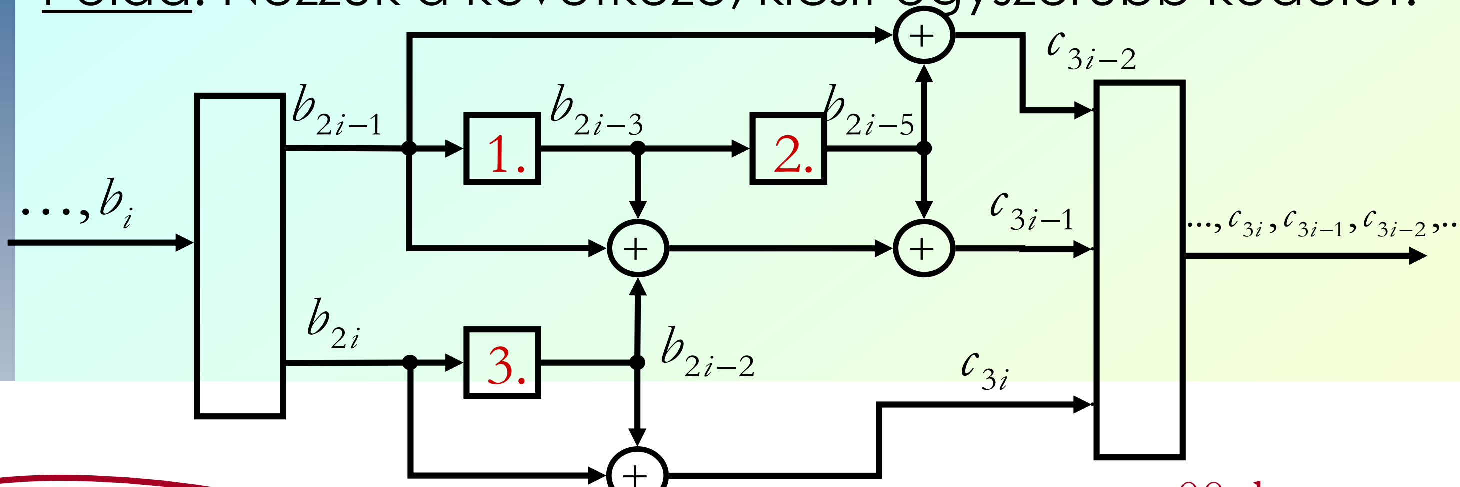


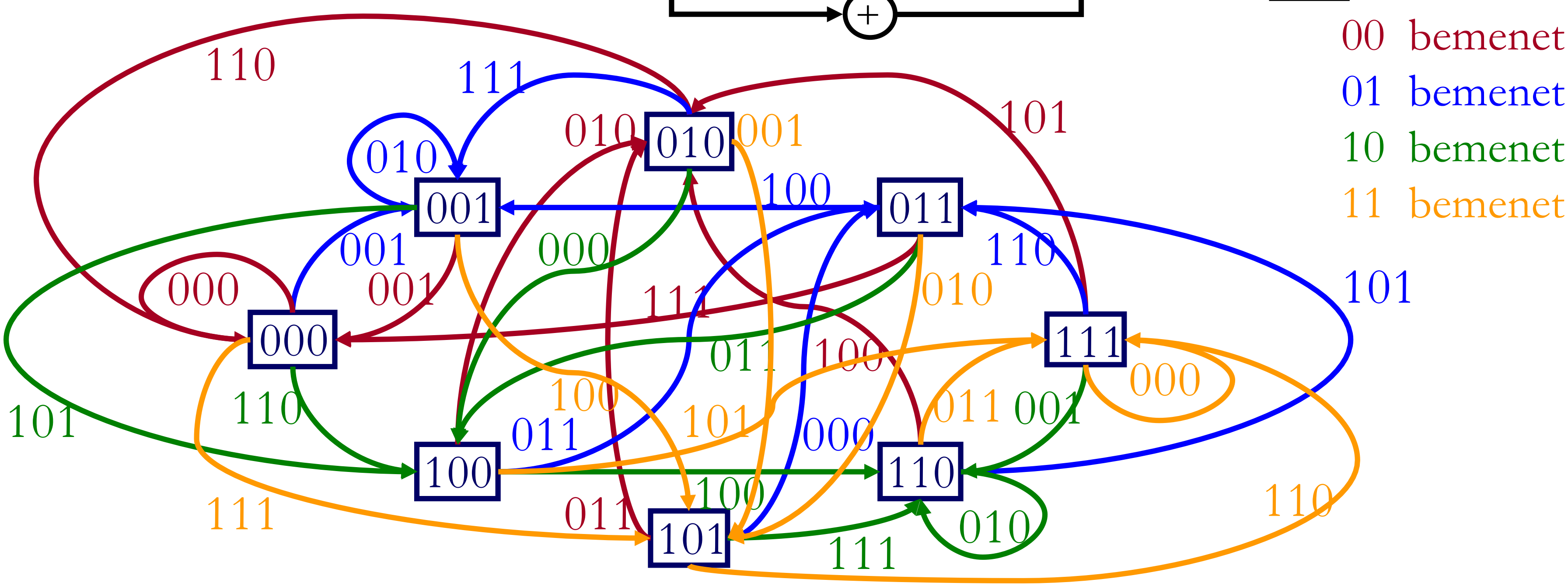
A kódoló paraméterei: $k=2$, $n=3$, $m=2$, $K=6$, $N=9$.

A tárolóknak összesen $2^3 = 8$-féle állapota lehetséges, az állapotátmeneti gráf 32 csúccsal rendelkezik. Ezeket fogjuk most felrajzolni. $2^2 = 4$-féle bemeneti kombináció lehetséges, így minden csúcsból négy ág indul ki.

Konvolúciós kódok

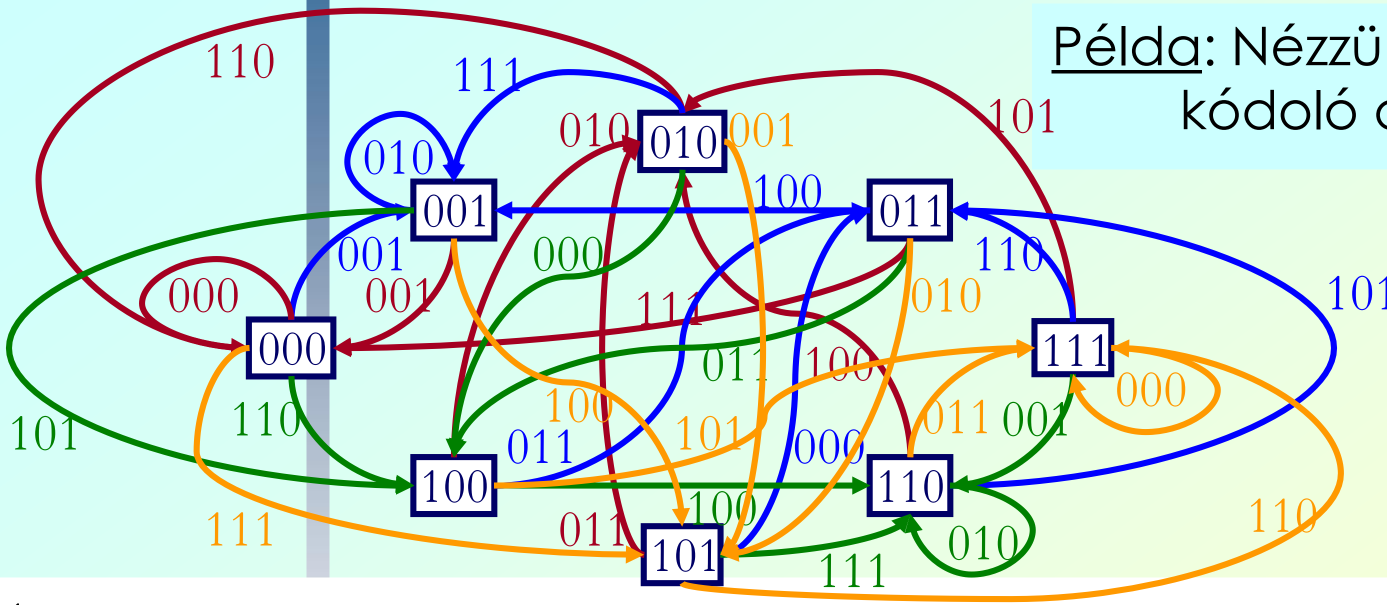
Példa: Nézzük a következő, kicsit egyszerűbb kódolót:



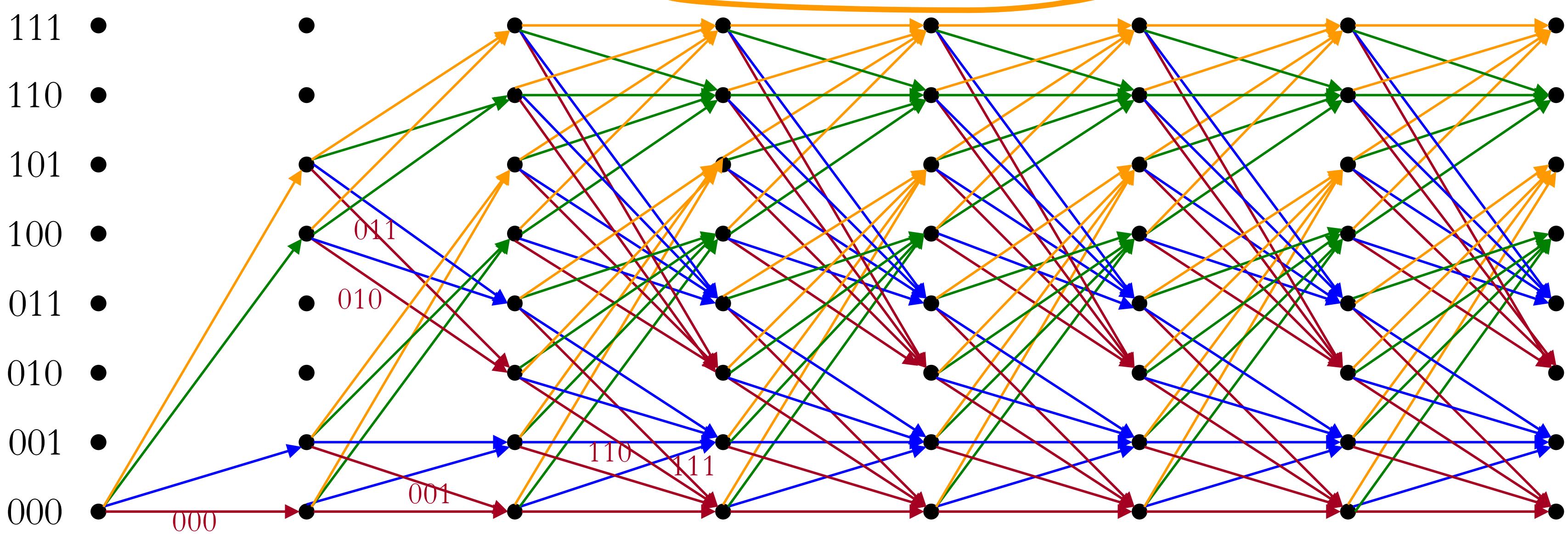


Konvolúciós kódok

Példa: Nézzük a következő kódoló áramkör trellisét:

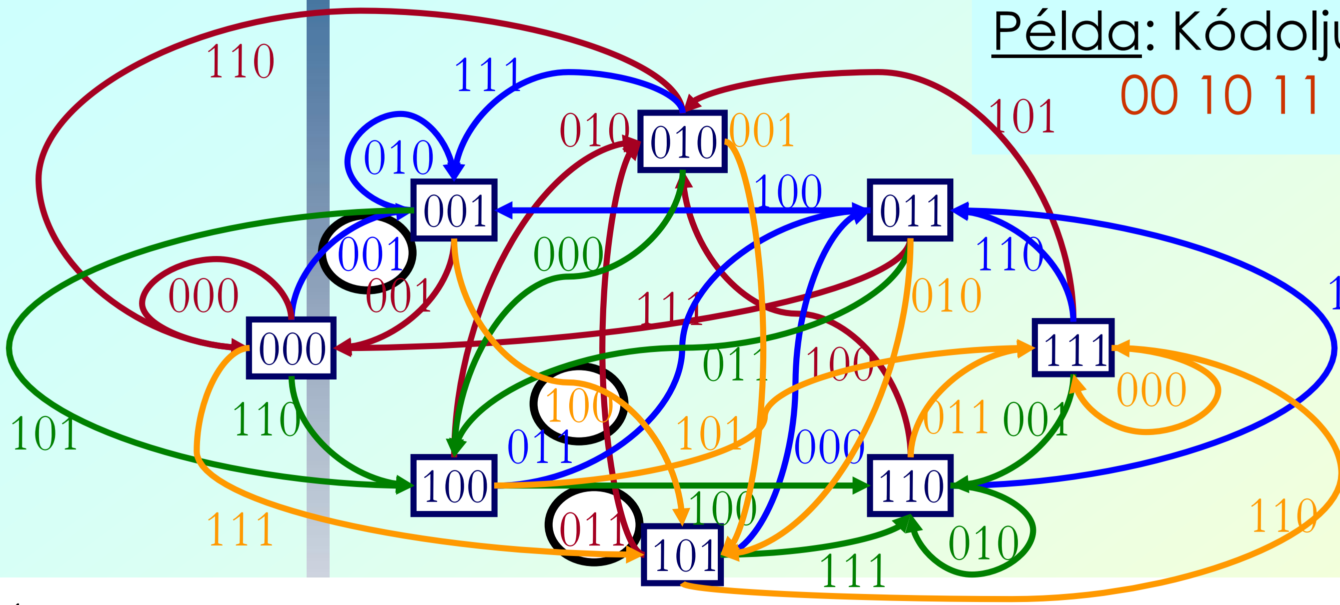


00 bemenet
01 bemenet
10 bemenet
11 bemenet

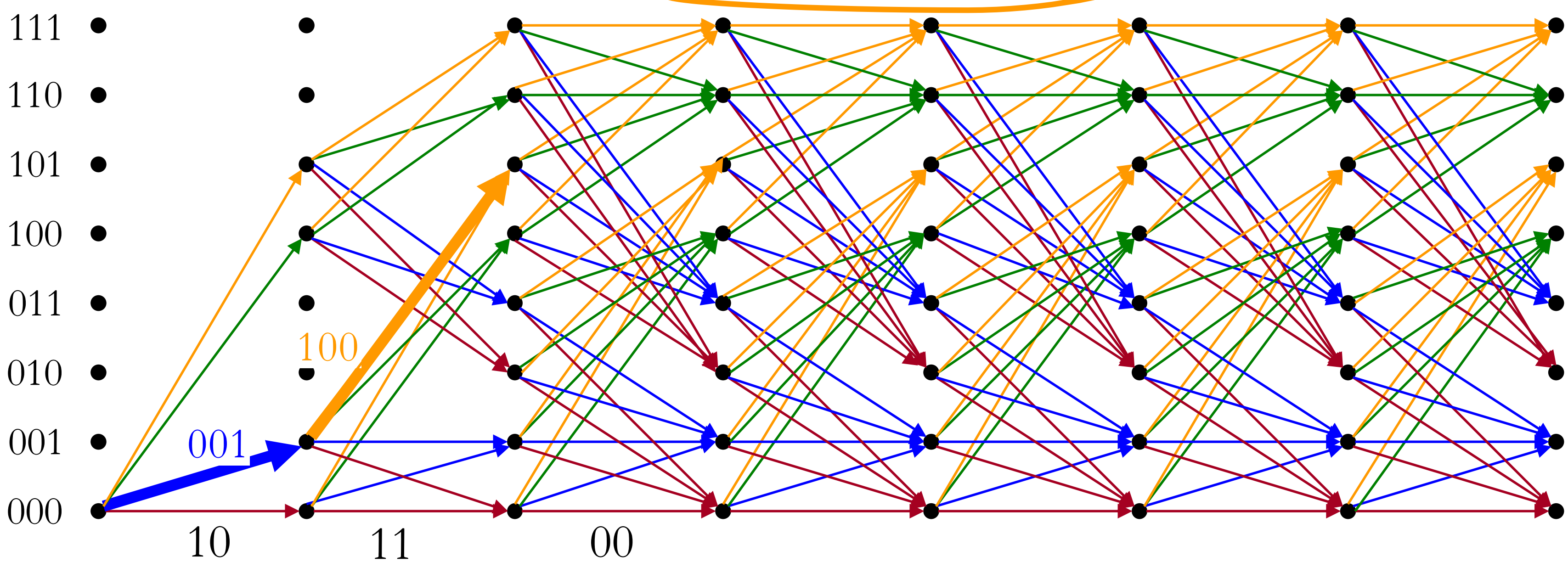


Konvolúciós kódok

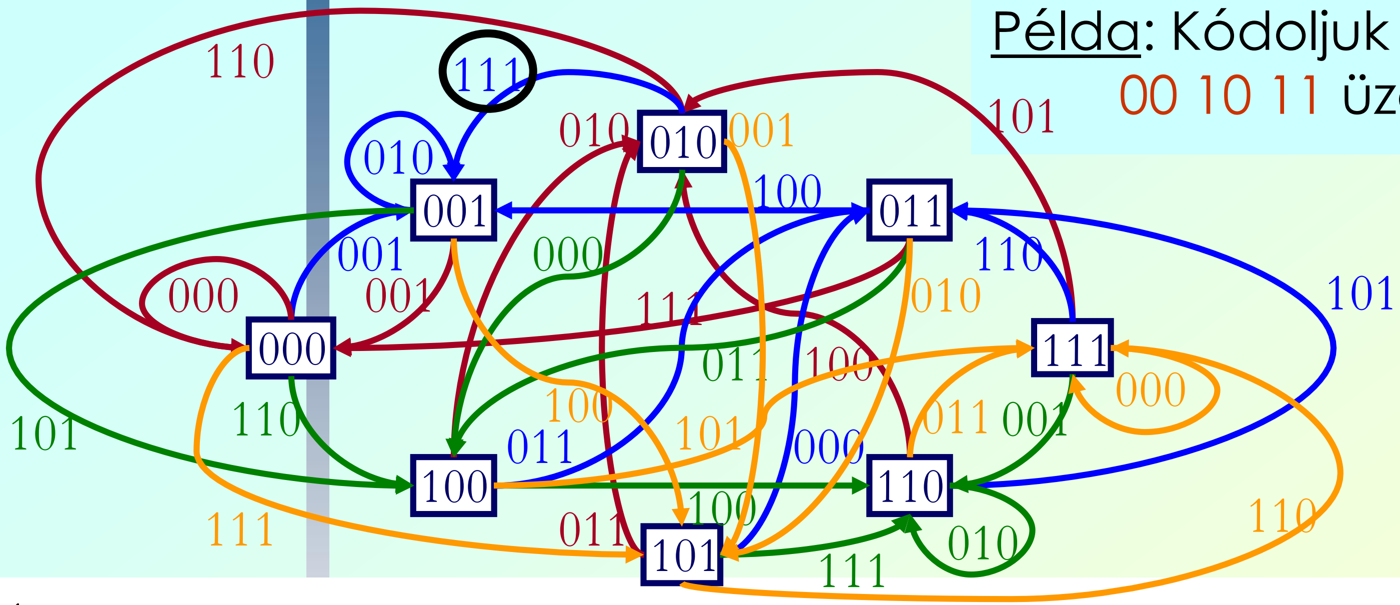
<u>Példa</u>: Kódoljuk a 01 11 00 01 00 10 11 üzenetet:



00 bemenet
01 bemenet
10 bemenet
11 bemenet

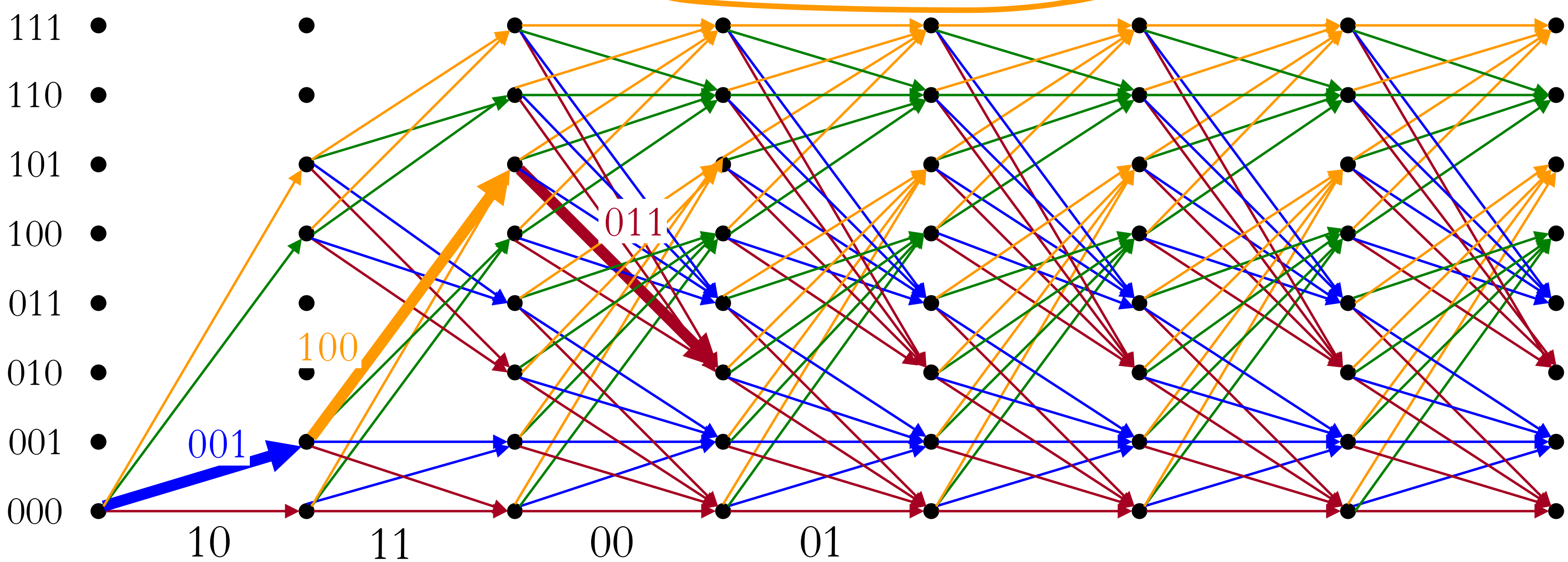


Konvolúciós kódok



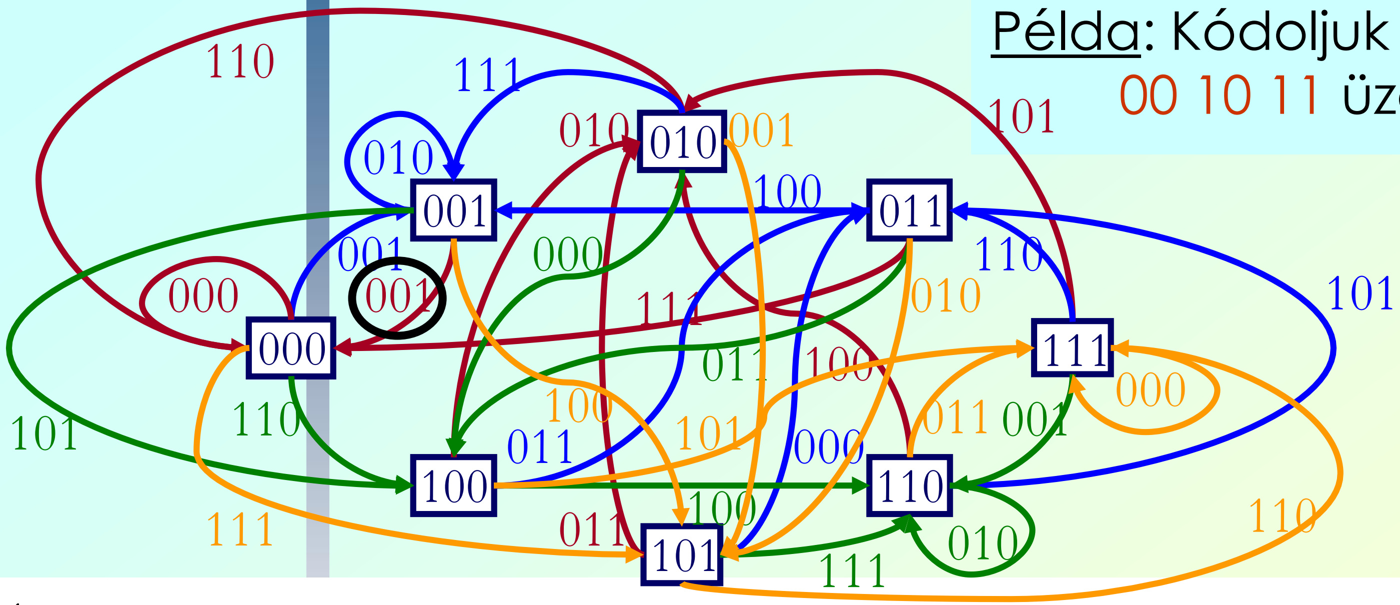
Példa: Kódoljuk a 01 11 00 01 00 10 11 üzenetet:

00 bemenet
01 bemenet
10 bemenet
11 bemenet

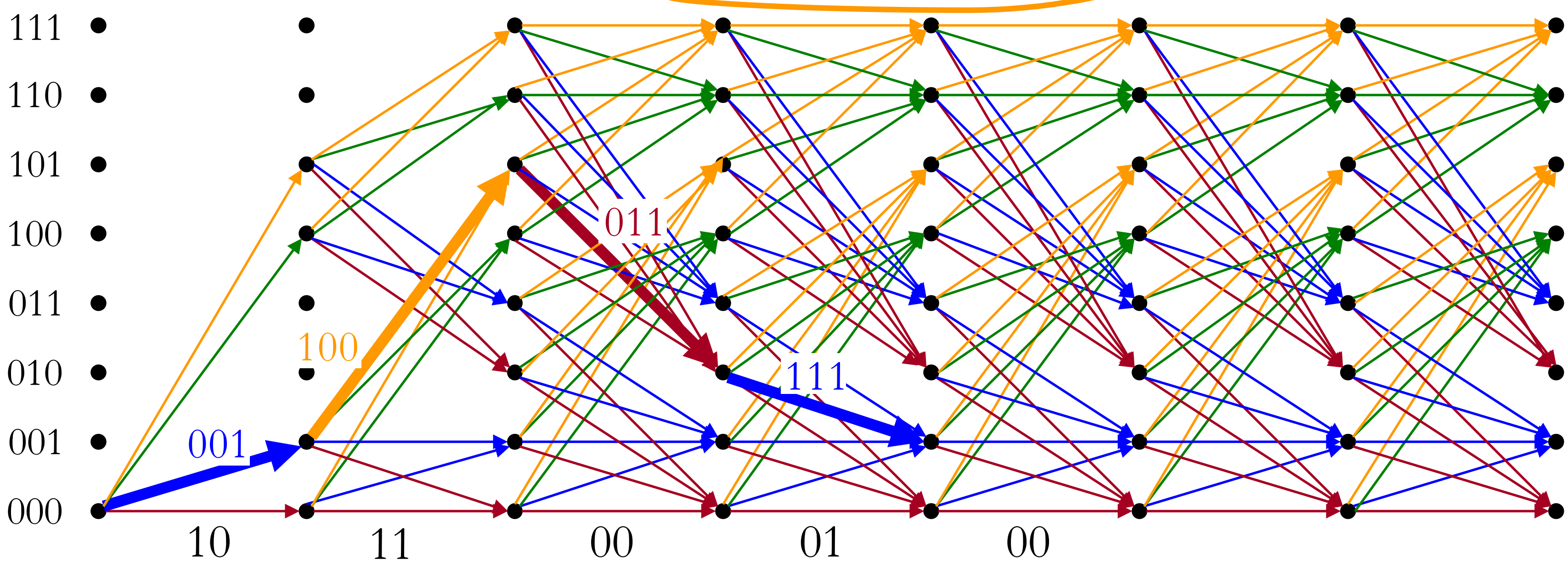


Konvolúciós kódok

Példa: Kódoljuk a 01 11 00 01 00 10 11 üzenetet:



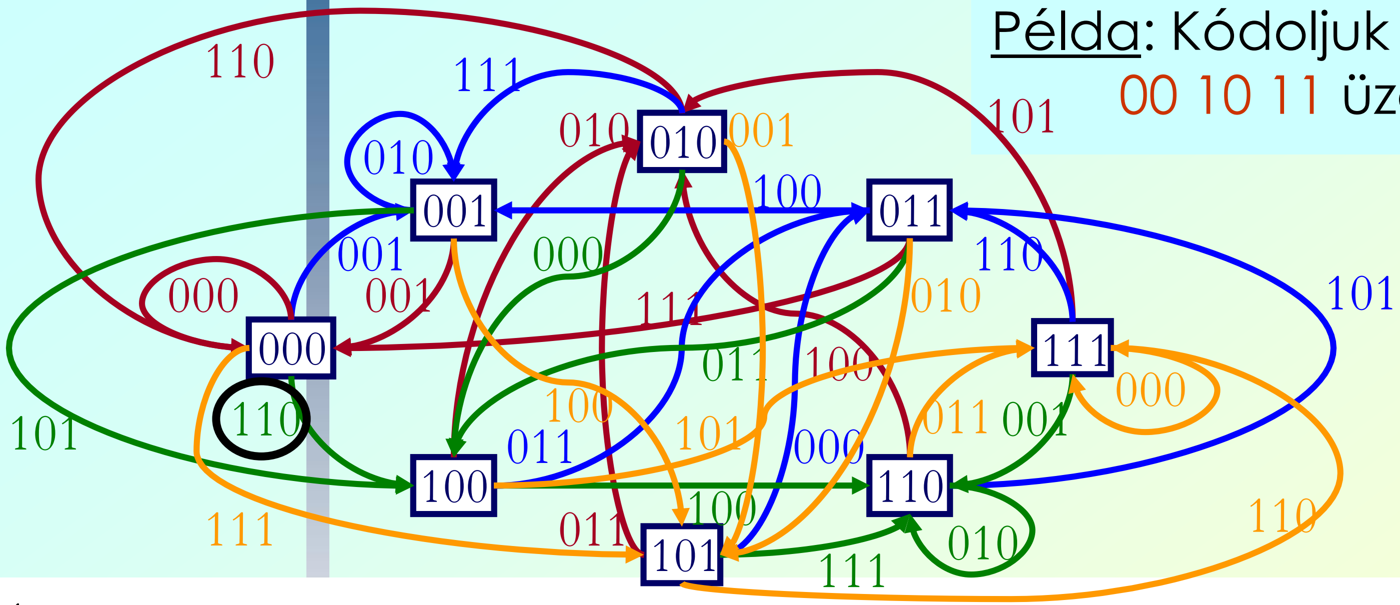
00 bemenet
01 bemenet
10 bemenet
11 bemenet

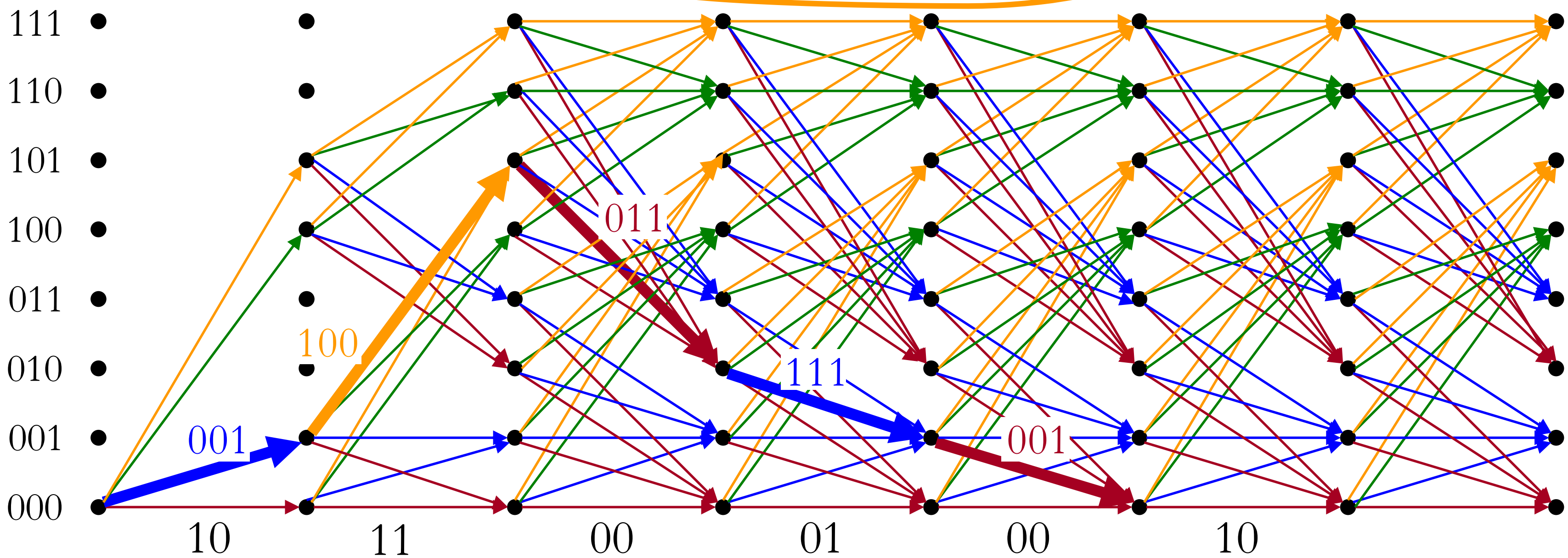


Konvolúciós kódok

Példa: Kódoljuk a 01 11 00 01 00 10 11 üzenetet:

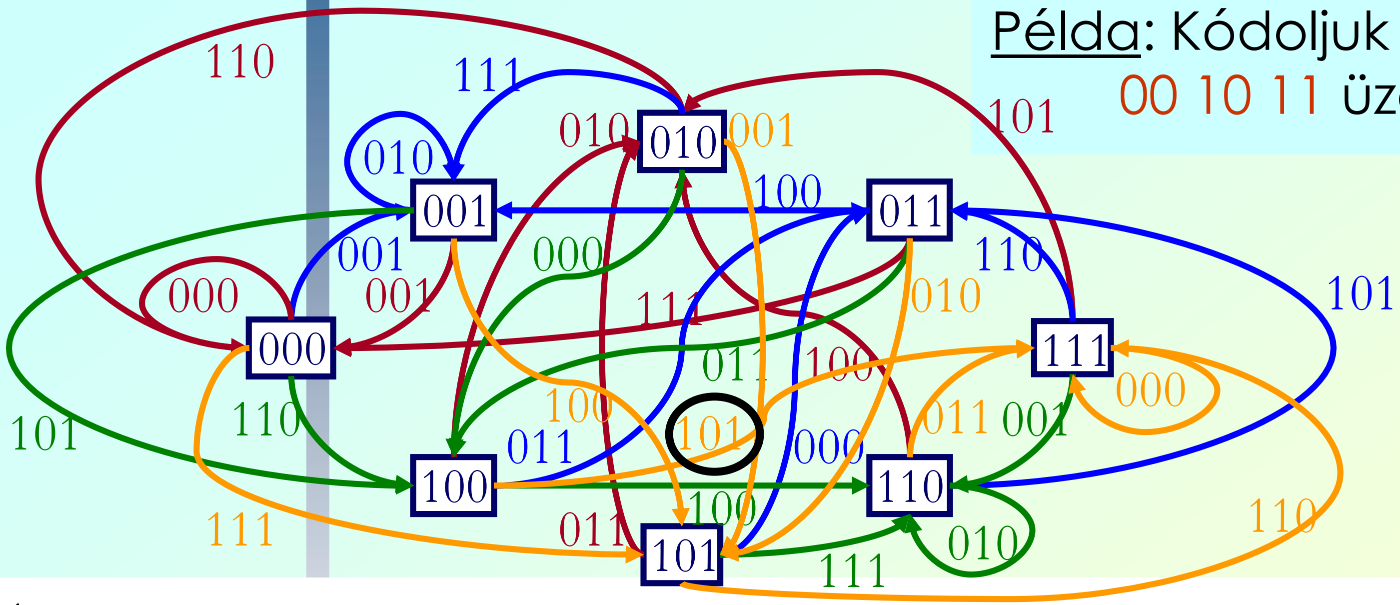
00 bemenet
01 bemenet
10 bemenet
11 bemenet



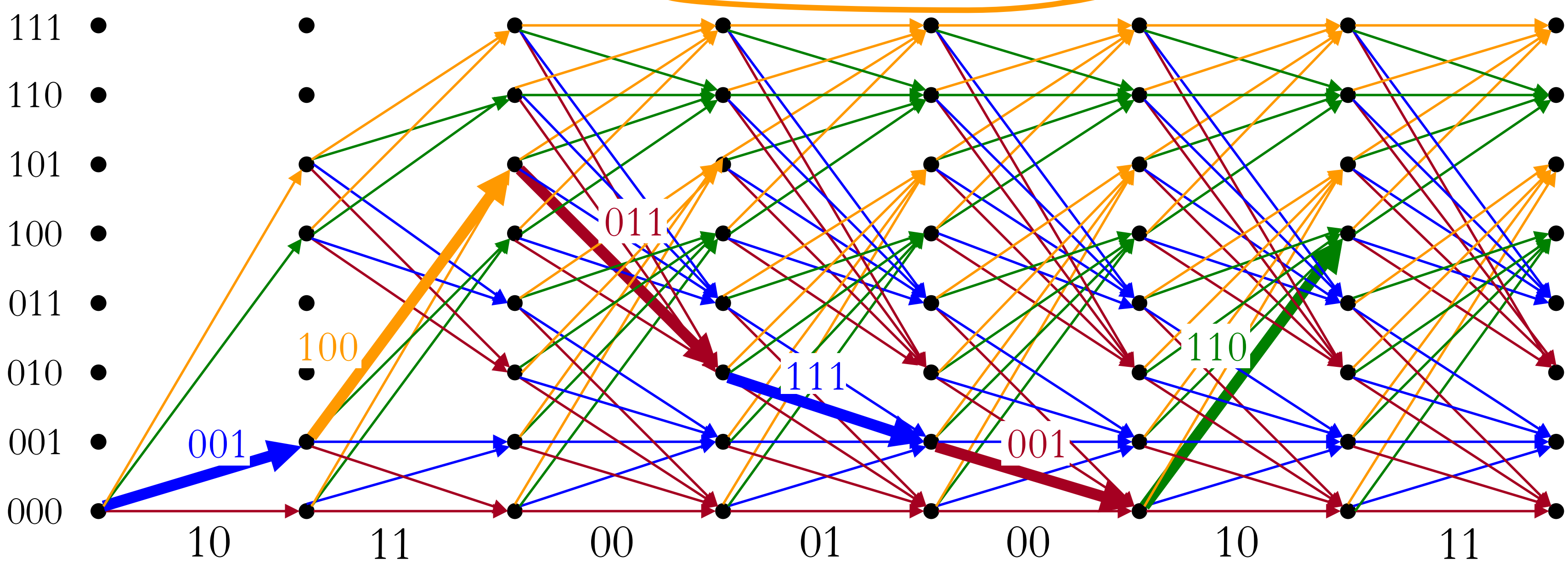


Konvolúciós kódok

Példa: Kódoljuk a 01 11 00 01 00 10 11 üzenetet:



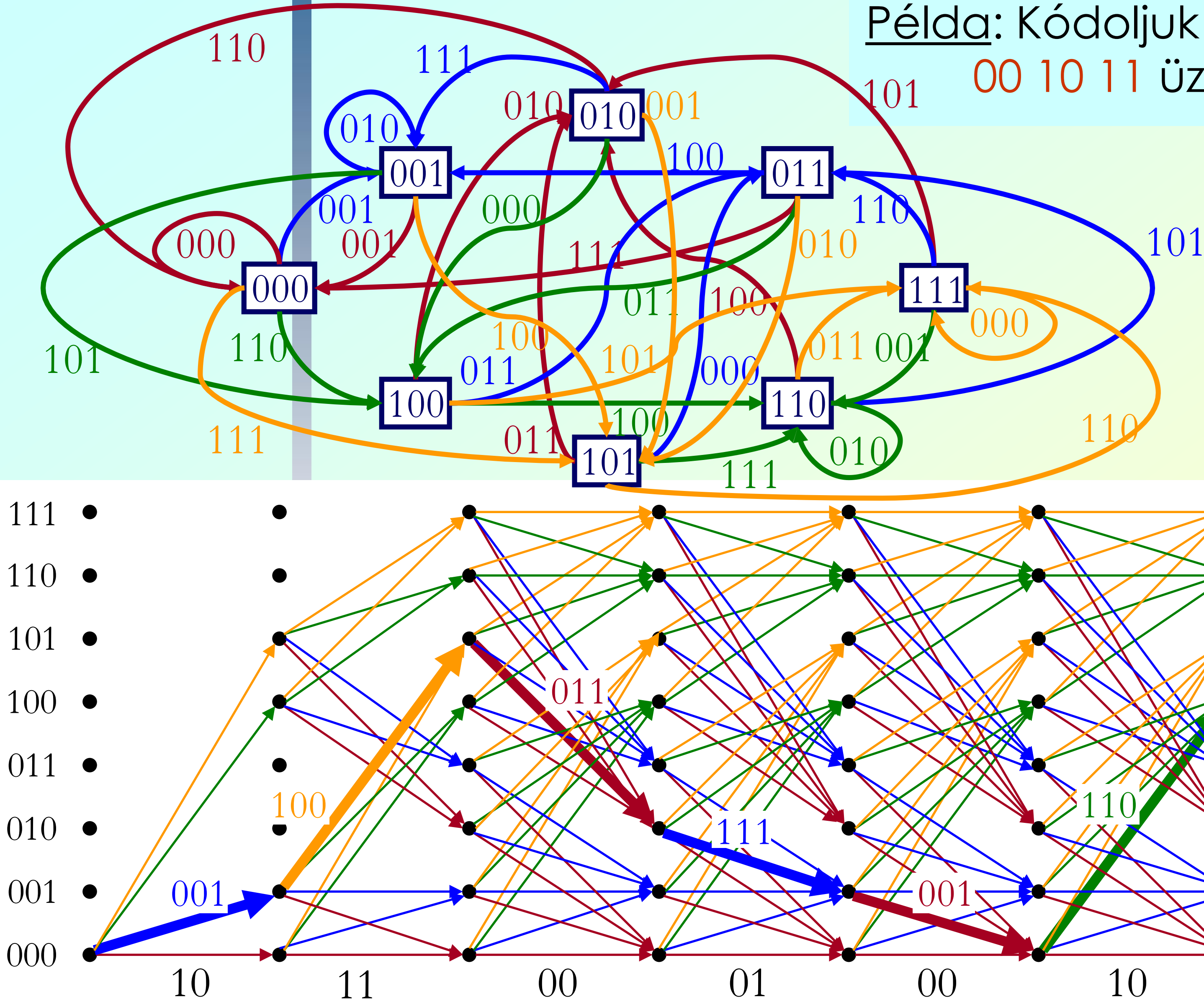
00 bemenet
01 bemenet
10 bemenet
11 bemenet



Konvolúciós kódok

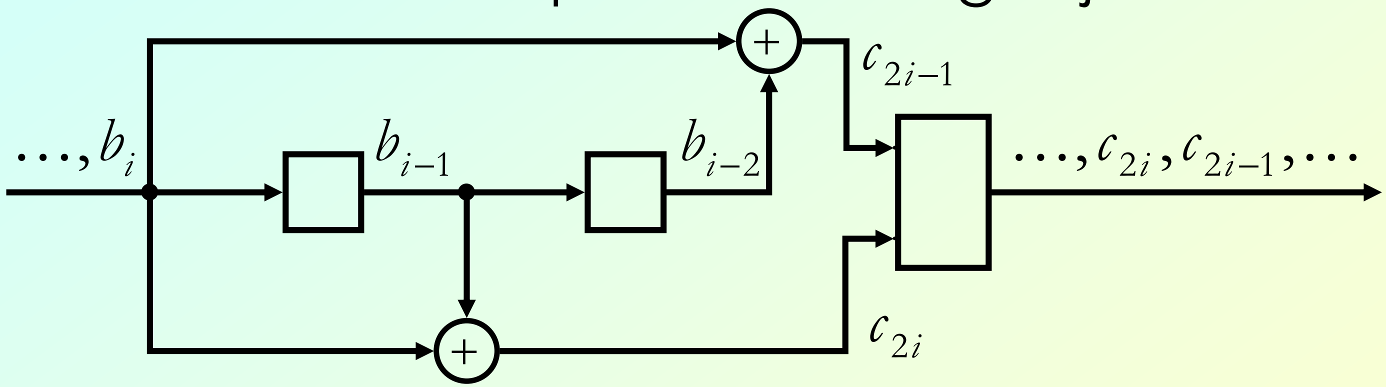
<u>Példa</u>: Kódoljuk a 01 11 00 01 00 10 11 üzenetet:

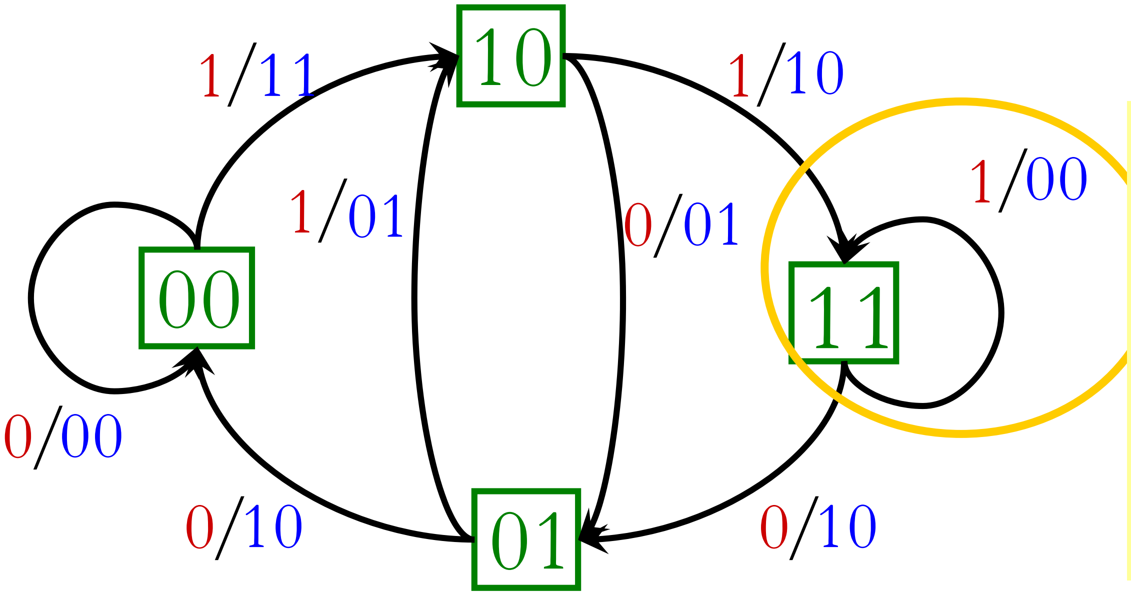
00 bemenet
01 bemenet
10 bemenet
11 bemenet



Katasztrofális kódoló

Módosítsuk kicsit az első kódolónkat, és nézzük az állapotátmeneti gráfját:



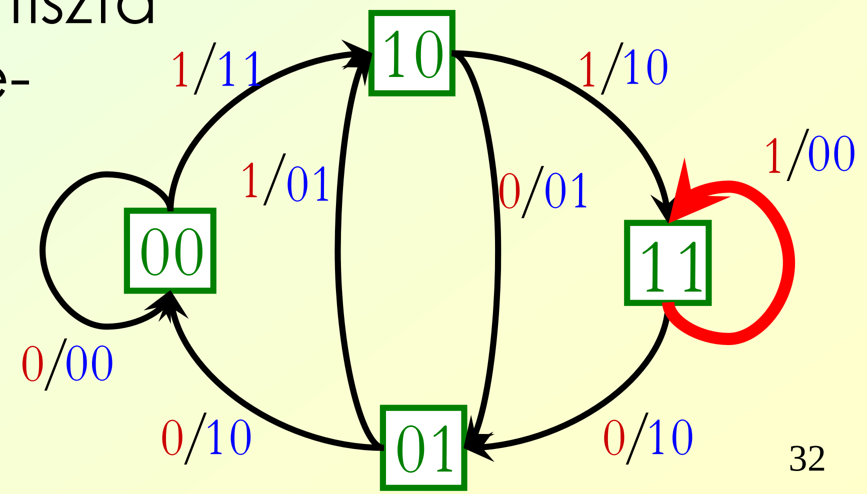


Engedjünk csupa 1-est a bemenetre. Ekkor rövid tranziens (két lépés, 11 10 kimenet) után a kimenet csupa nulla lesz.

Katasztrofális kódoló

Az olyan kódolókat amelyek nem csak tiszta nulla bemenetre, hanem valamilyen más, periodikus bemeneti sorozatra is tiszta nulla kimenetet adnak, katasztrofális kódolóknak nevezik.

A katasztrofális kódolónak az állapotátmeneti gráfján mindig van egy olyan hurok, amelyik nem a tiszta nulla állapotból indul és mégis tiszta nulla a kimenete, azaz nulla a hurok kimenetének Hamming-súlya.



Katasztrofális kódoló

A katasztrofális kódoló pontos definíciója: Az olyan kódolók, melyek tetszőlegesen nagy Hamming-súlyú bemenetre is korlátos Hamming-súlyú maradhat a kimenet. (I. a kódolónkra bármennyi 1-est engedünk, a kimenet lehet 3-nál nem nagyobb súlyú.)

Szeretnénk elkerülni a katasztrofális kódolókat, ezért szeretnénk egy olyan feltételt, amely az állapotátmenet-gráf (hosszadalmas) felrajzolása nélkül is megmondja, hogy katasztrofális-e a kódoló vagy nem.

Katasztrofális kódoló

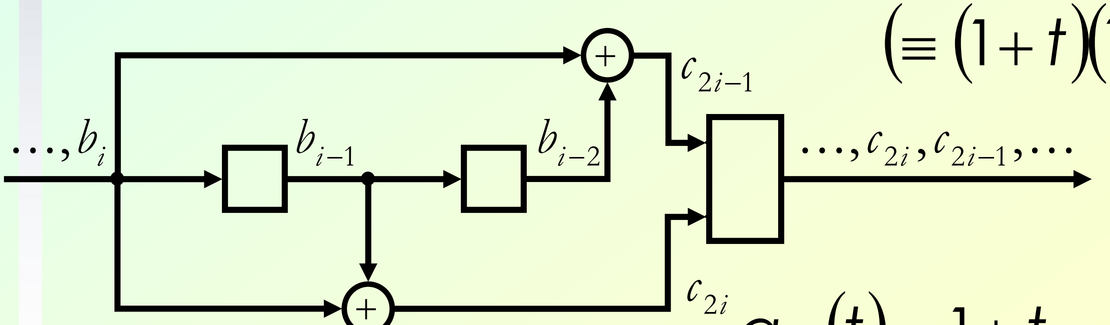
Ha k=1, azaz a kódsebesség 1/n, akkor létezik ilyen feltétel: a kódoló akkor és csak akkor nem katasztrofális, ha az ágait jellemző polinomok legnagyobb közös osztója 1:

$$\mathrm{lnko}\{g_{11}(t), g_{12}(t), \ldots g_{1n}(t)\} = 1$$

A katasztrofális kódolónkra:

$$g_{11}(t) = 1 + t^2$$

$$\left(\equiv (1+t)(1+t) \mod 2\right)$$



$$g_{12}(t) = 1 + t$$

Távolságprofil, szabad távolság

Vizsgáljuk a kódoló lehetséges kimeneteiből mindig az első *r* kódszókeretet. Az így kapott $r \cdot n$ hosszúságú vektoroknak értelmezhető a Hamming-távolsága. Jelöljük ezek közül a minimálisat d_r^*-gal. Ekkor *r* növelésével ez a minimális távolság nem fog csökkenni:

$$d_1^* \le d_2^* \le d_3^* \le \ldots$$

A $d_1^*, d_2^*, d_3^*, \ldots$ sorozatot a kódoló **távolságprofil**jának nevezik.

A d_r^*-okból álló monoton növekvő sorozat határértéke, avagy ezen értékek maximuma a kód **szabad távolsága**:

$$d_\infty = \max_r d_r^*$$

Kódolási nyereség

A csatornakódolás során ezesetben is a kódtávolság növelésére törekedünk.

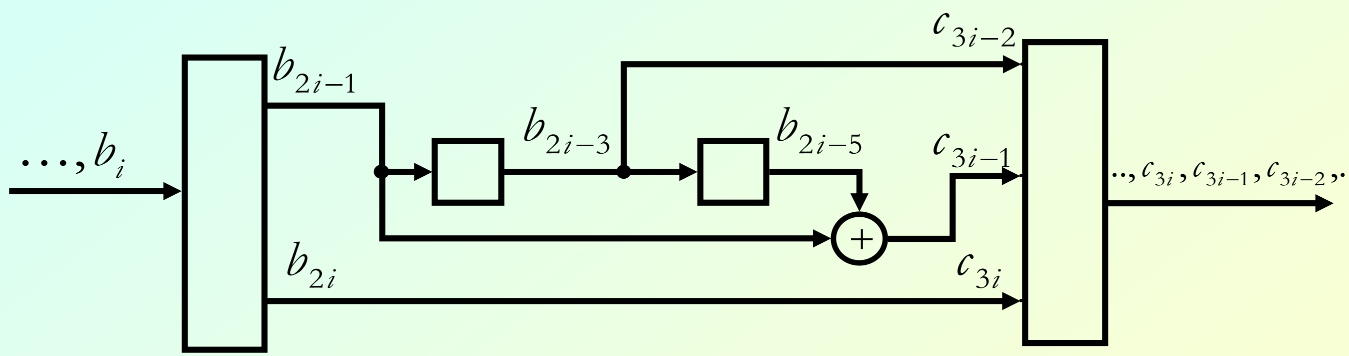
Vegyük a kódolatlan üzenet d_{ref} szabad távolságát referenciának.
A **kódolási nyereség**:

$$G = 20 \lg \frac{d_\infty}{d_{ref}}$$

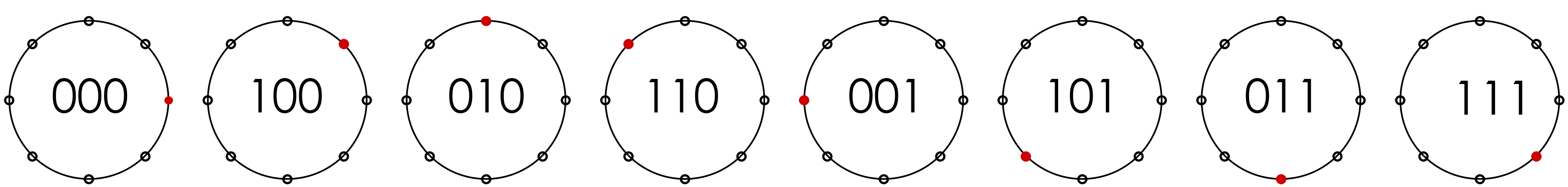
a kódolt üzenet szabad kódtávolságá-nak és a referenciatávolság aránya decibelskálán.

Komplex kódábécé

A konvolúciós kódolók kimenetét nem csak *n* bitként lehet értelmezni, hanem komplex számként is: például a



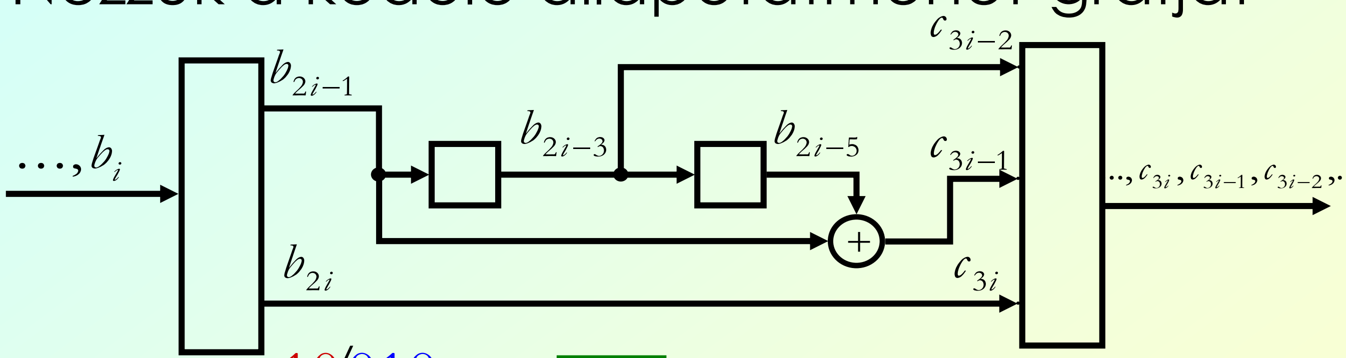
áramkör három bitnyi kimenete értelmezhető 8PSK modulátor fázisszögeként is ($2^3=8$):

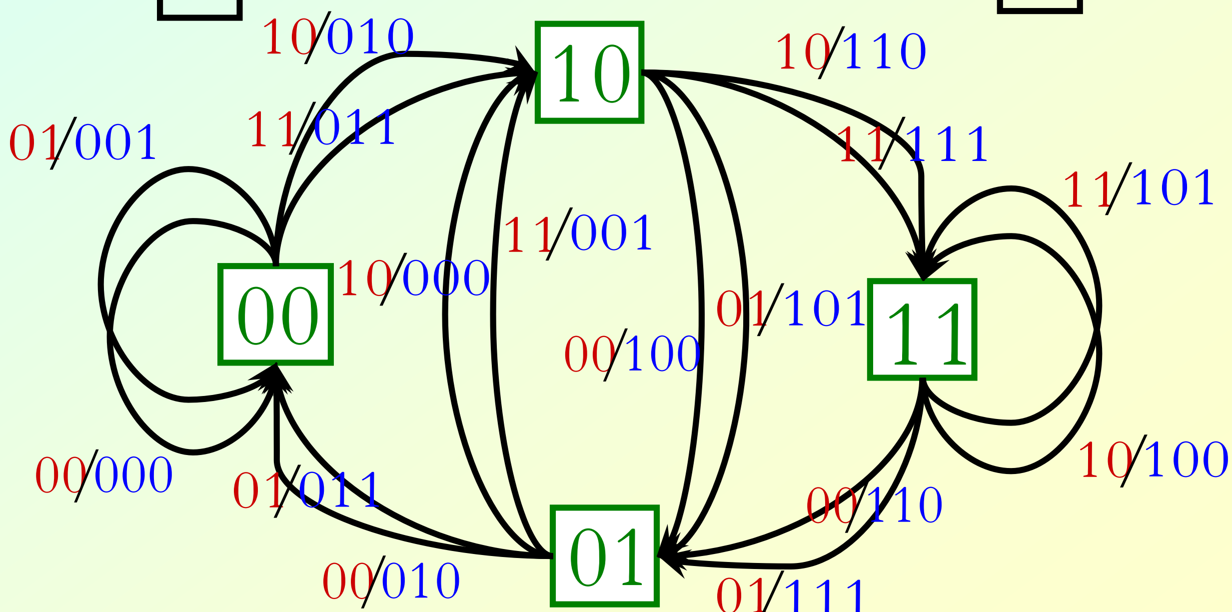


Komplex kódábécé

Komplex kódábécével jelentős kódolási nyereség érhető el. Lássuk:

Nézzük a kódoló állapotátmenet-gráfját





Komplex kódábécé

Konvolúciós kódok

Alapfogalmak

Állapotátmeneti gráf, trellis

Polinom-reprezentáció

Katasztrofális kódoló

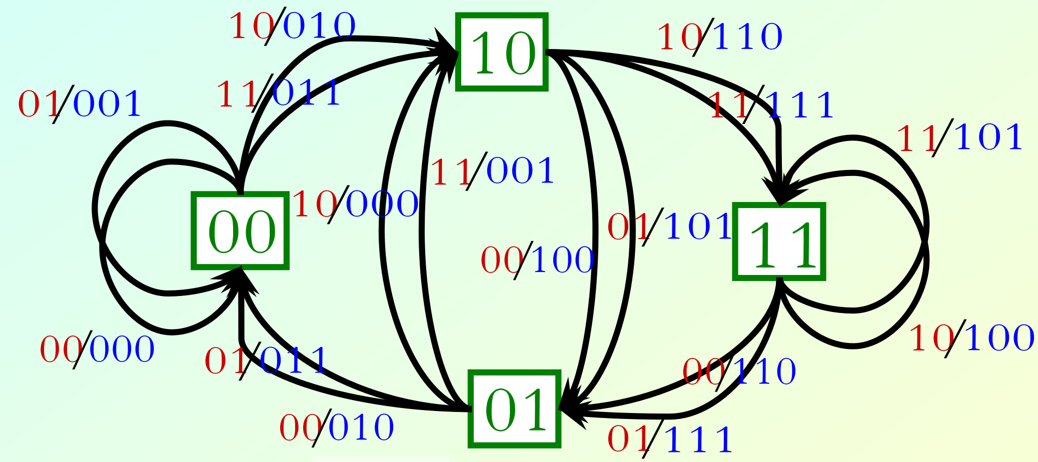
Szabad távolság

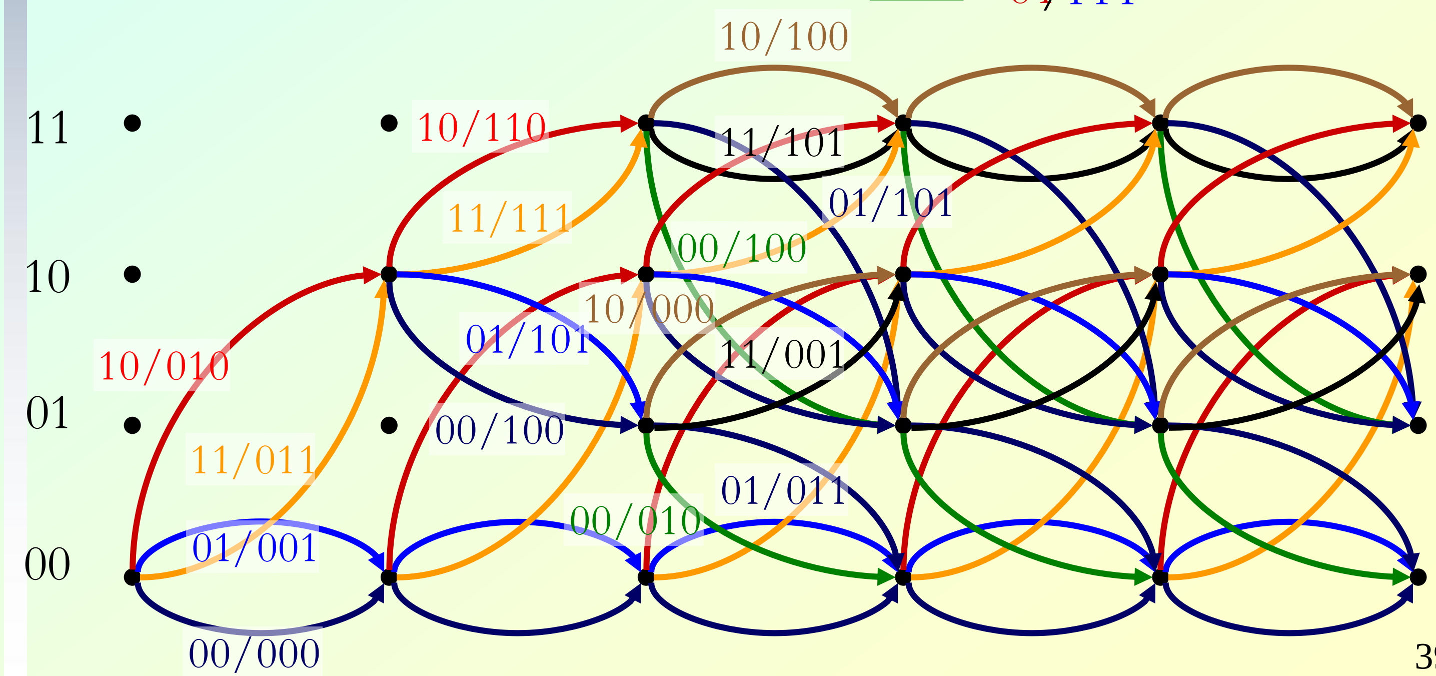
Maximum likelihood dekódolás

Visszacsatolt konvolúciós kódolók

Turbó kódok

Nézzük a kódoló trellisét:

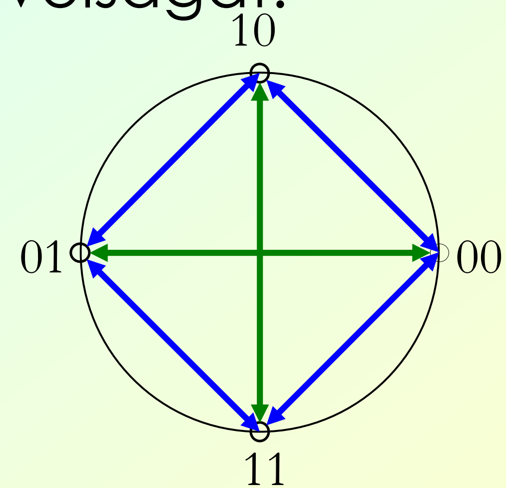




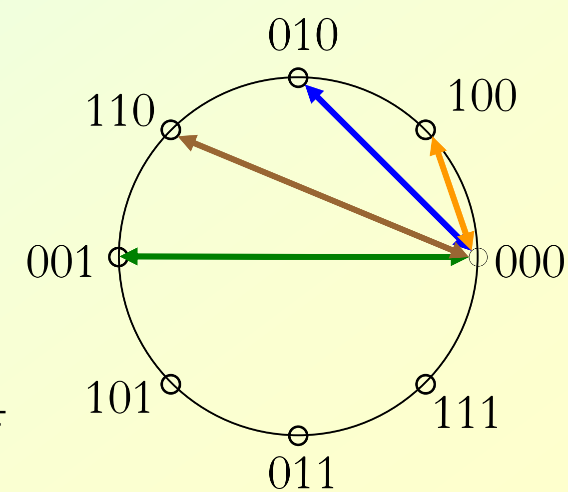
Komplex kódábécé

Nézzük az egyes állapotok távolságát:

- a kiindulási bitpárokat, mint komplex számokat értelmezve a távolságaik euklideszi mértékkel:
  - 2
  - $\sqrt{2}$

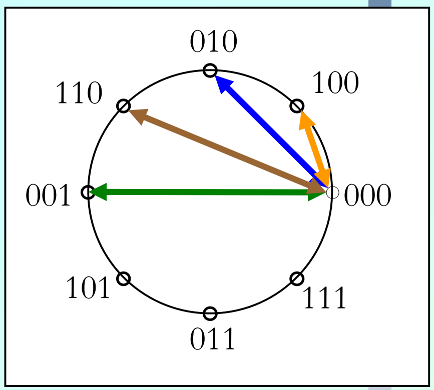


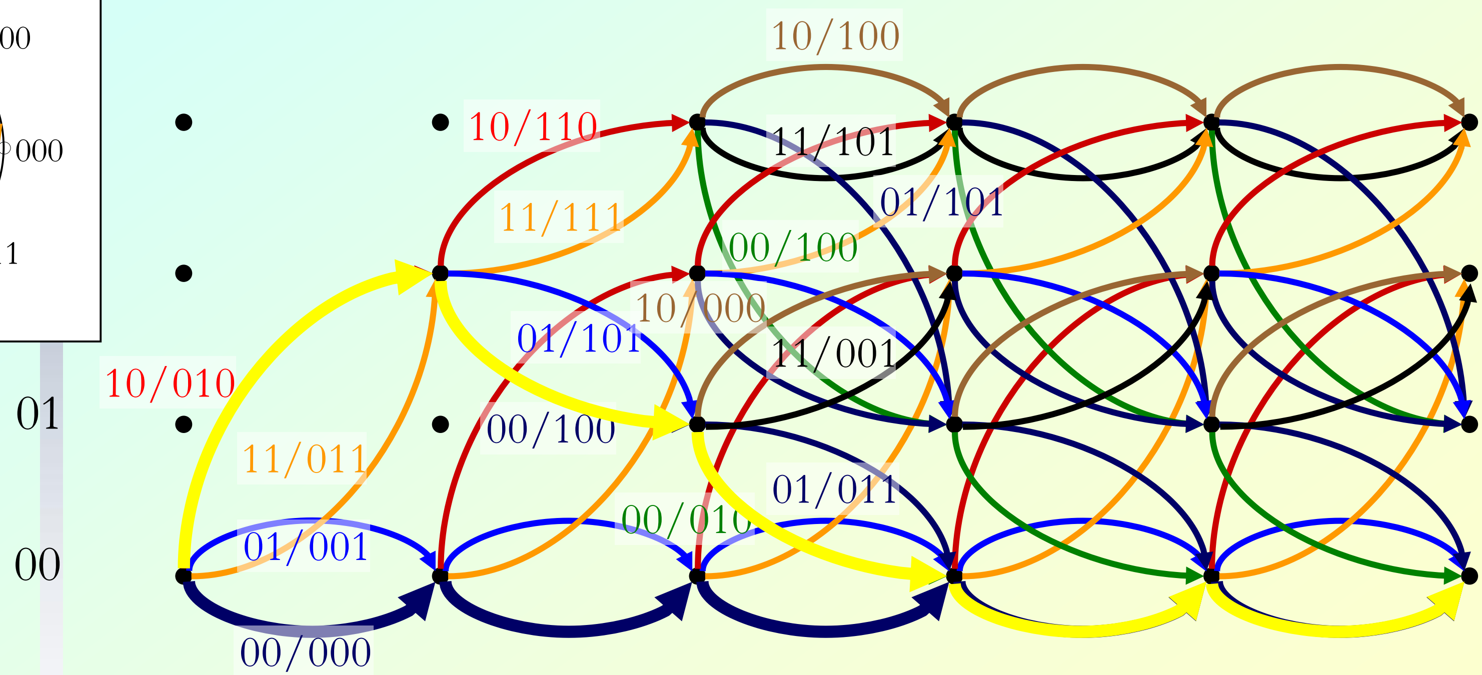
- a kimeneti bittrióknak, mint komplex számoknak a távolsága:
  - 2
  - $\sqrt{2}$
  - $\sqrt{2-\sqrt{2}}$
  - $\sqrt{2+\sqrt{2}}$



Komplex kódábécé

Keressük meg a tiszta nulla bemenetre adott tiszta nulla kimenethez euklideszi távolságban legközelebb eső kódolt üzenetet:





A hozzá tartozó résztávolságok:

$$\underbrace{\sqrt{2}\qquad \sqrt{2-\sqrt{2}}\qquad \sqrt{2}\qquad 0\qquad 0}$$

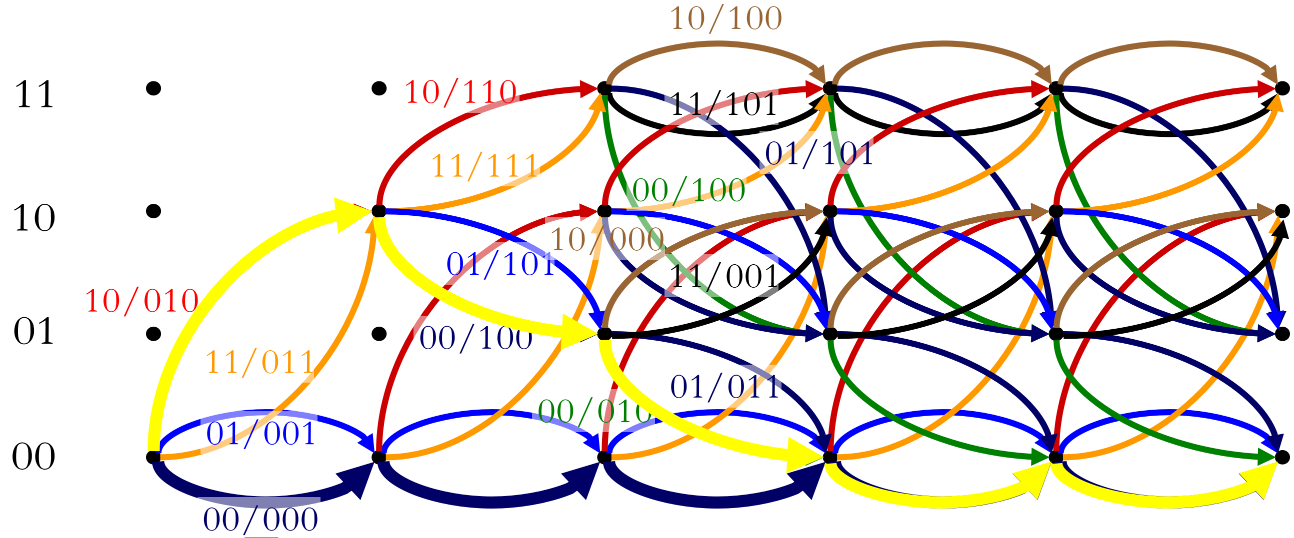
$$\sqrt{d_1^{\,2}+d_2^{\,2}+d_3^{\,2}+d_4^{\,2}+\ldots}=\sqrt{2+\left(2-\sqrt{2}\right)+2}=\sqrt{6-\sqrt{2}}=2{,}14$$

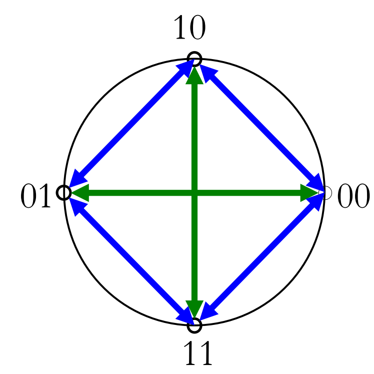
Komplex kódábécé

A tiszta nulla bemenetre adott tiszta nulla kimenet euklideszi távolsága a hozzá legközelebb eső kódolt üzenettől:

$$\sqrt{d_1^{\,2}+d_2^{\,2}+d_3^{\,2}+d_4^{\,2}+\ldots}=\sqrt{2+\left(2-\sqrt{2}\right)+2}=2{,}14$$

Az eredeti üzenet távolsága a tiszta nulla üzenettől





$\sqrt{2}$ 0 0 0 0 $\Longrightarrow$ $d_{\text{ref}}=\sqrt{2}$

Így a kódolási nyereség: $G=20\lg\left(\dfrac{2{,}14}{1{,}41}\right)=3{,}6\,\text{dB}$

Komplex kódábécé

A valós számokkal és Hamming-távolsággal ugyanez a kódolási nyereség:

az üzenet: 10 00 00 00 00 …

Hamming-távolsága a 00 00 00 00 00 … üzenettől: 1

kódolt üzenet: 010 100 010 000 000

Hamming-távolsága a 000 000 000 000 000 … kódolt üzenettől: 3

Így a kódolási nyereség:

$$G = 20\lg\left(\frac{3}{1}\right)$$

Maximum likelihood dekódolás

A Viterbi-dekódolás olyan algoritmus amelyet a trellis kódok maximum likelihood dekódolására fejlesztettek ki és optimalizáltak.

Szemléltetés: Bináris szimmetrikus csatorna $p < 1/2$ paraméterrel. Legyen a csatornára adott vektor $\mathbf{c} = (c_1, c_2, \ldots, c_N)$, a kimeneten észlelt vektor $\mathbf{v} = (v_1, v_2, \ldots, v_N)$. A $p(\mathbf{v} \mid \mathbf{c})$ feltételes valószínűség:

$$p(\mathbf{v} \mid \mathbf{c}) = (1-p)^N \cdot \left(\frac{p}{1-p}\right)^{d(\mathbf{v},\mathbf{c})},$$

Maximum likelihood dekódolás

$$p(\mathbf{v}|\mathbf{c}) = (1-p)^N \cdot \left(\frac{p}{1-p}\right)^{d(\mathbf{v},\mathbf{c})},$$

mivel a csatorna mindkét bit esetén $1-p$ valószínűséggel továbbít helyes jelet és p valószínűséggel hibáz; N darab szimbólum van és ebből $d(\mathbf{v}, \mathbf{c})$ helyen tér el a két vektor egymástól, azaz ennyi helyen rontott a csatorna.

Maximum likelihood dekódolás

Maximum likelihood döntésnél tehát a

$$p(\mathbf{v}|\mathbf{c}) = (1-p)^N \cdot \left(\frac{p}{1-p}\right)^{d(\mathbf{v},\mathbf{c})},$$

valószínűséget kell maximalizálni, azaz, mivel

$$\left(\frac{p}{1-p}\right) < 1,$$

a $d(\mathbf{v}, \mathbf{c})$ Hamming-távolságot kell minimalizálni.

Maximum likelihood dekódolás

$$p(\mathbf{v}|\mathbf{c}) = (1-p)^N \cdot \left(\frac{p}{1-p}\right)^{d(\mathbf{v},\mathbf{c})},$$

Tehát a $d(\mathbf{v}, \mathbf{c})$ Hamming-távolságot kell minimalizálni.

Jelöljük az *i*-edik kódszókeretet $\mathbf{c}_i$-vel, a belőle a csatorna kimenetén kapott szimbólumsorozatot $\mathbf{v}_i$-vel, távolságukat $d(\mathbf{v}_i, \mathbf{c}_i)$-vel. ekkor a

$$d(\mathbf{v},\mathbf{c}) = \sum_i d(\mathbf{v}_i,\mathbf{c}_i)$$

Hamming-távolság minimumát kell keresni ($d(\mathbf{c},\mathbf{v})$ a fenti szorzat kitevőjében van).

Maximum likelihood dekódolás

A trellisben minden kódszókeretet egy-egy él reprezentál. Rendeljünk hozzá minden élhez egy olyan súlyfaktort avagy **metrikát** (mértéket), amely arányos ezzel a mennyiséggel, járjunk végig minden lehetséges utat, és válasszuk ki közülük a maximális súlyút. (Ha a Hamming-távolság a metrika akkor a minimumot kell keresni) Ez a Viterbi-dekódolás alapötlete.

A legegyszerűbb, ha a vett bitsorozattól mért Hamming-távolságot nevezzük ki metrikának. (BSC esetén jó is)

Maximum likelihood dekódolás

Más típusú diszkrét, emlékezet nélküli csatornánál is lehet a szorzatként előálló $p(\mathbf{v}\,|\,\mathbf{c})$ likelihood helyett olyan mennyiséget találni, amelyet egy-egy kódszókerethez tartozó részmennyiségek összegeként lehet meghatározni:

$$\log p(\mathbf{v}|\mathbf{c}) = \sum_{i=1}^{N} \log p(\mathbf{v}_i|\mathbf{c}_i).$$

A Viterbi-algoritmus

A Viterbi-dekódolás A következő ciklust hajtja végre amíg el nem fogy a vett szimbólumsorozat:

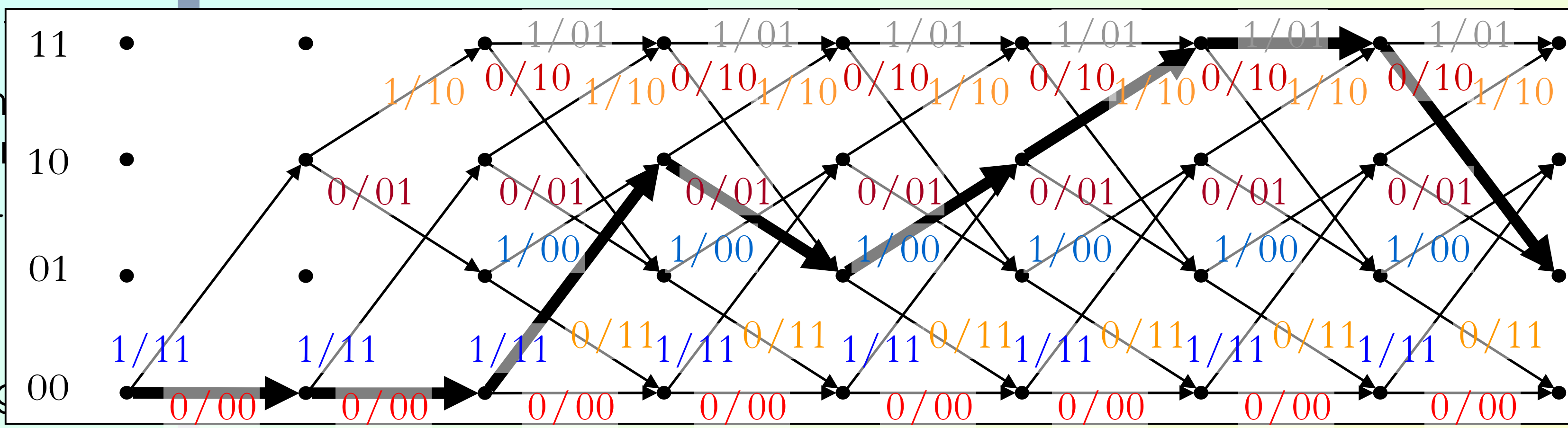
1. beolvas egy kódszókeretet, az *i*-ediket: $\mathbf{c}_i$-t
2. kiszámolja a trellis *i*-edik és *i*+1-edik mélységi csomópontjai közötti ágak súlyát a $\mathbf{c}_i$ ismeretében
3. előhívja az *i*-edik mélységi csomópontokig vezető utak metrikáját, ezekhez hozzáadja az újakat, így minden *i*+1-edik csomóponthoz kap több útvonalat különféle metrikával;

A Viterbi-algoritmus

3. előhívja az *i*-edik mélységi csomópontokig vezető utak metrikáját, ezekhez hozzáadja az újakat, így minden *i*+1-edik csomóponthoz kap több útvonalat különféle metrikával; ezeket az útvonalakat hozzárendeli azokhoz az *i*+1-edik mélységi csomópontokhoz, amelyekbe mutatnak.
4. az állapotokhoz rendelt útvonalak közül kiválasztja a maximális súlyút, azt elraktározza az adott csomóponthoz, ez lesz a **túlélő útvonal**, a többit törli. (Hamming-távolság esetén a maximális súly a minimális Hamming-távolság.)

A Viterbi-dekódolás működése

Nézzük az egyik egyszerű korábbi kódolónkat, és a belőle a **0 1 0 1 1 1 0** üzenet hatására kapott kódot:



00 00 11 01 00 10 01 10.

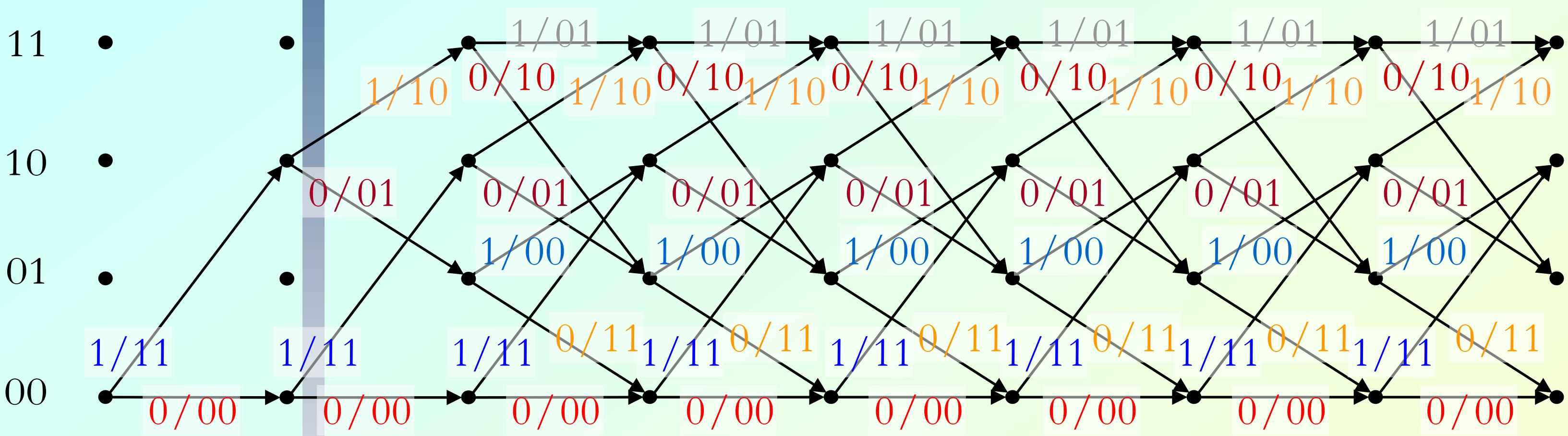
Hibázzon a csatorna a második és az ötödik bitben, így a vett bitsorozat:

01 00 01 01 00 10 01 10.

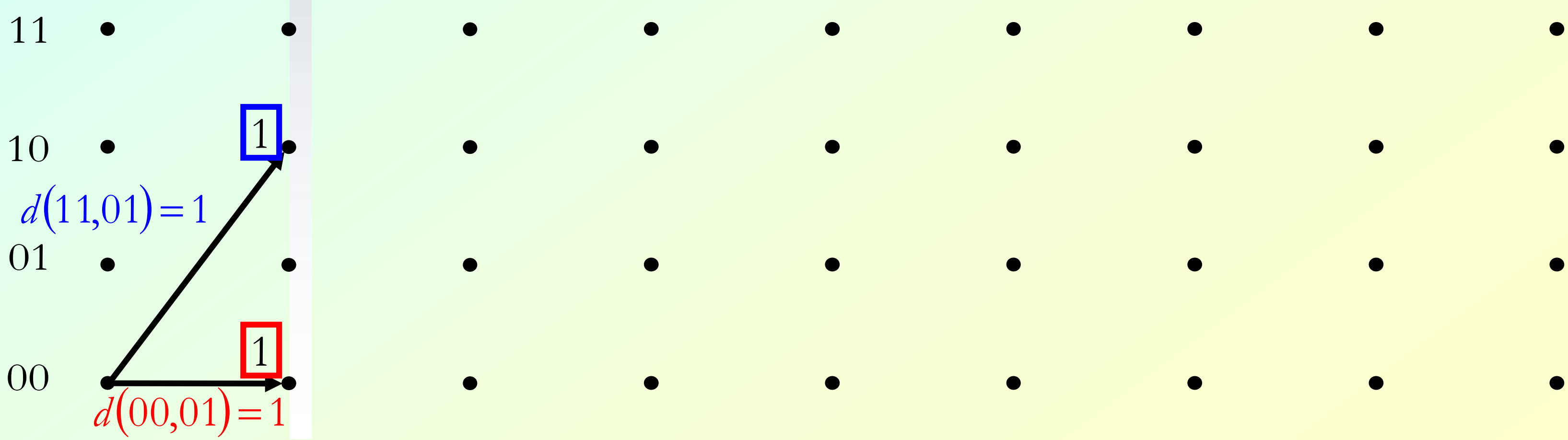
Hajtsuk végre a Viterbi-dekódolást:

A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

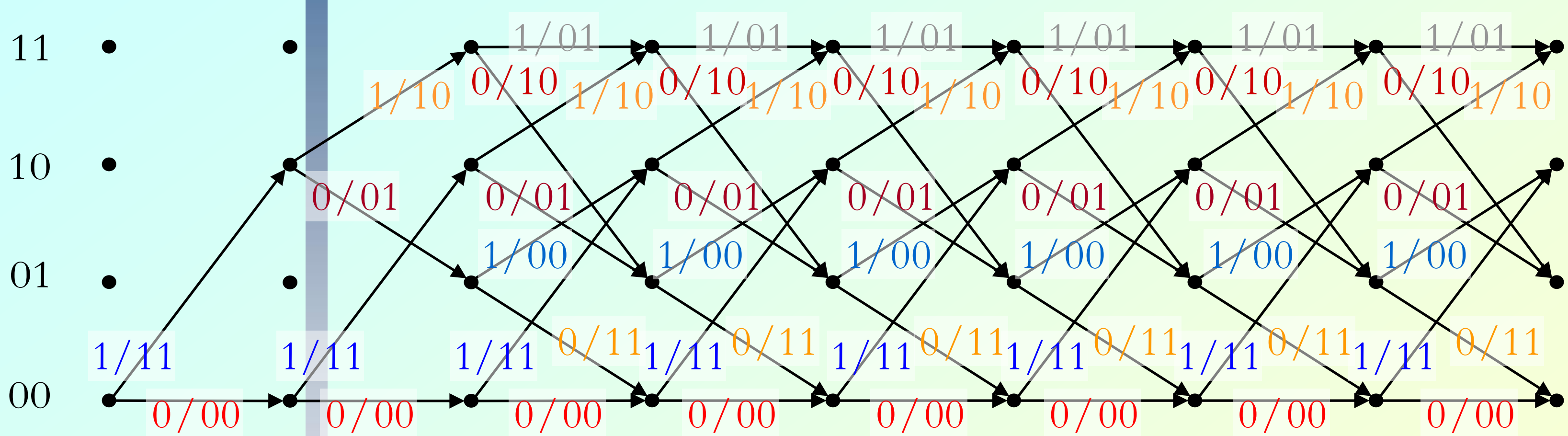


A vett bitsorozat: 01 00 01 01 00 10 01 10. 1. lépés: az 1. élek súlya

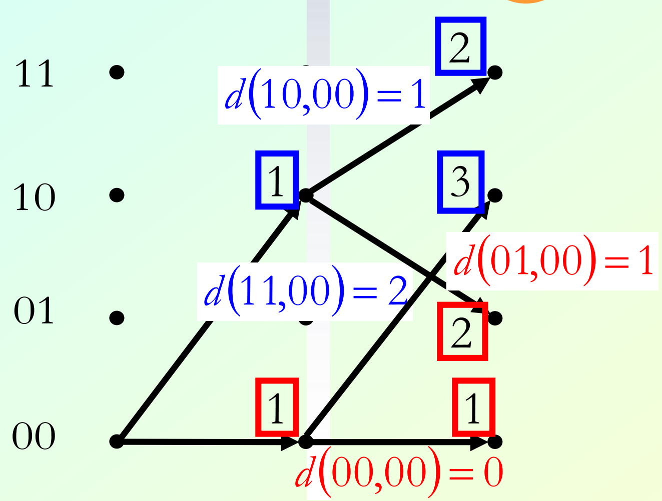


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:



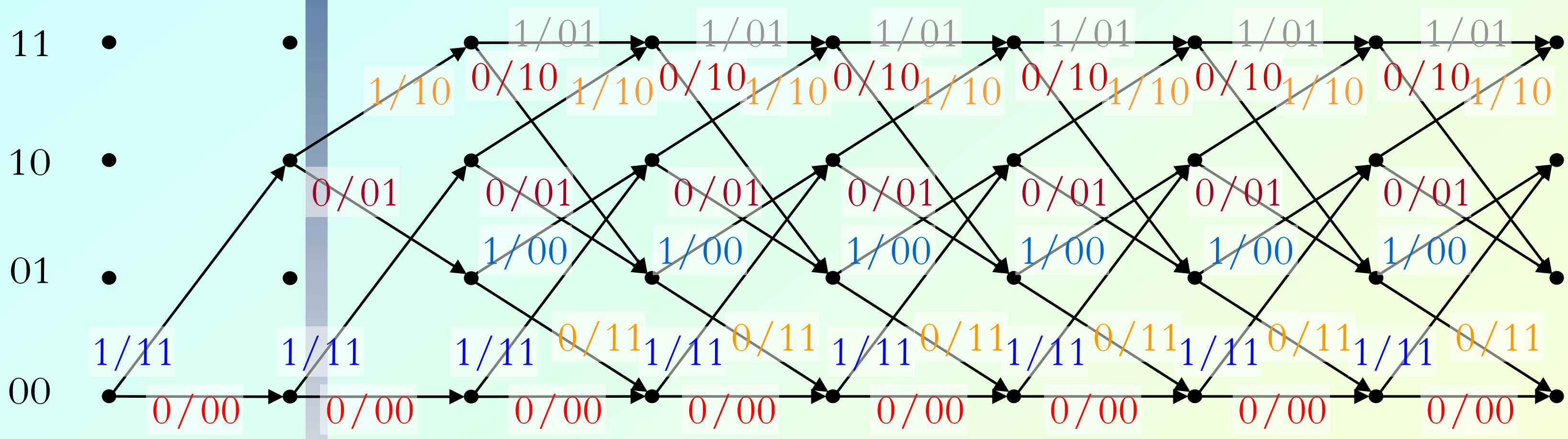
A vett bitsorozat: 01 00 01 01 00 10 01 10.   2. lépés: a 2. élek súlya



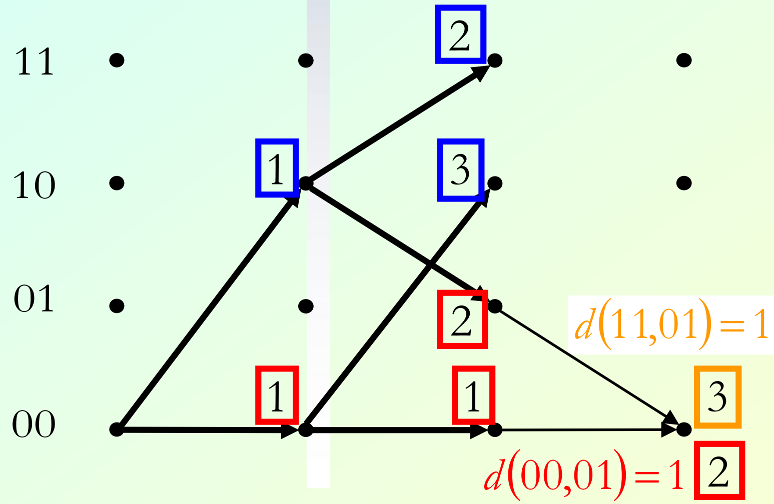
Eddig minden csúcshoz csak egy él érkezett, nem kellett választani közülük

A Viterbi-dekódolás működése

Emlékeztetőül a trellis:



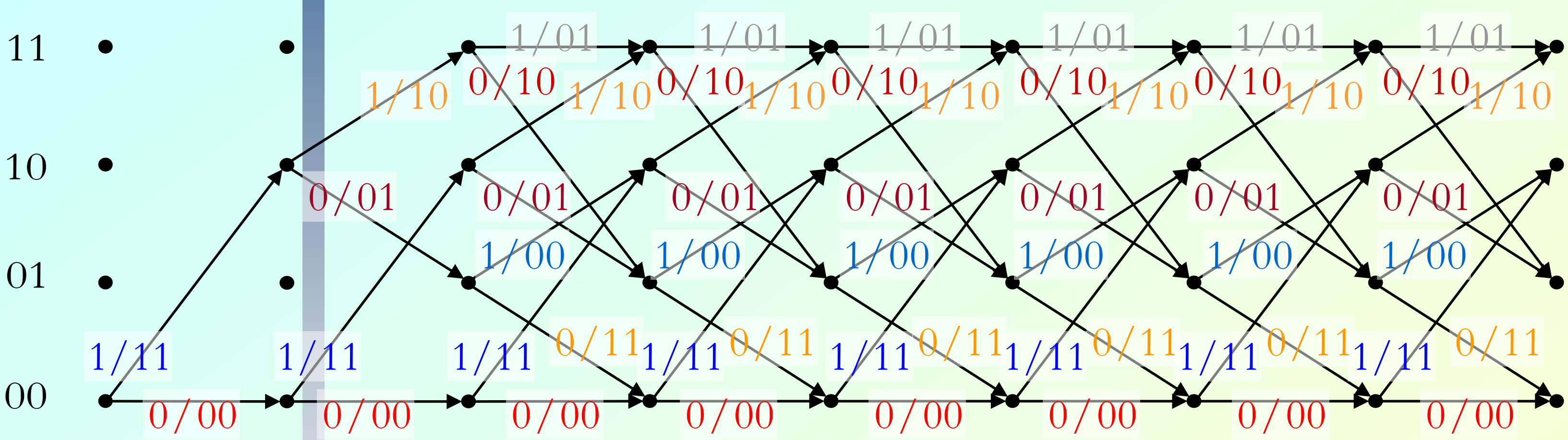
A vett bitsorozat: 01 00 01 01 00 10 01 10.



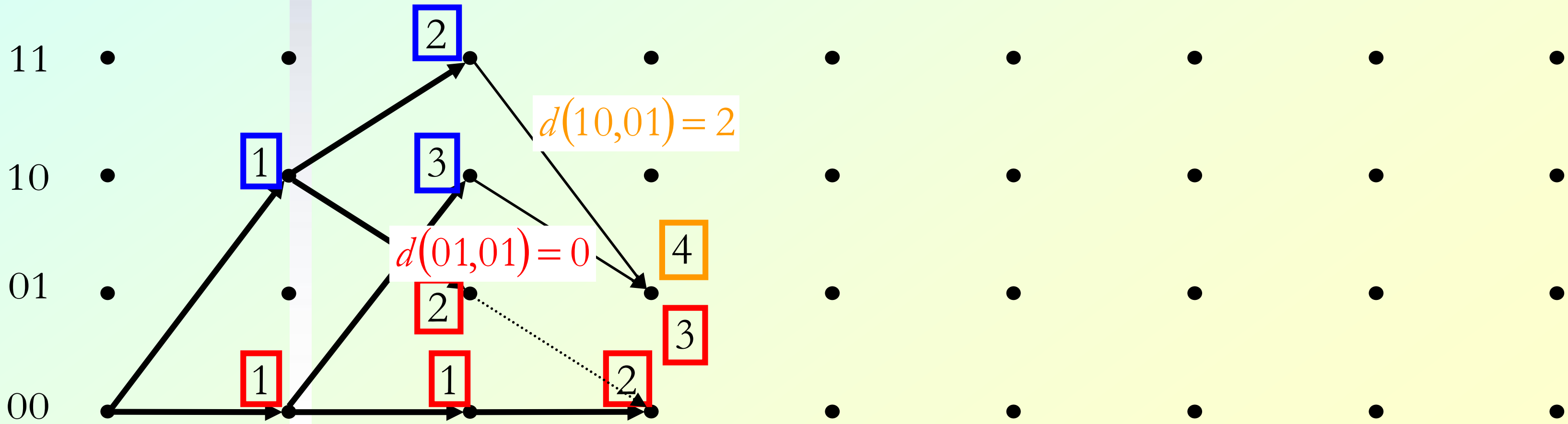
a két él közül kisebb súlyút (a pirosat) választjuk, az lesz a **túlélő él**, a másikat töröljük

A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

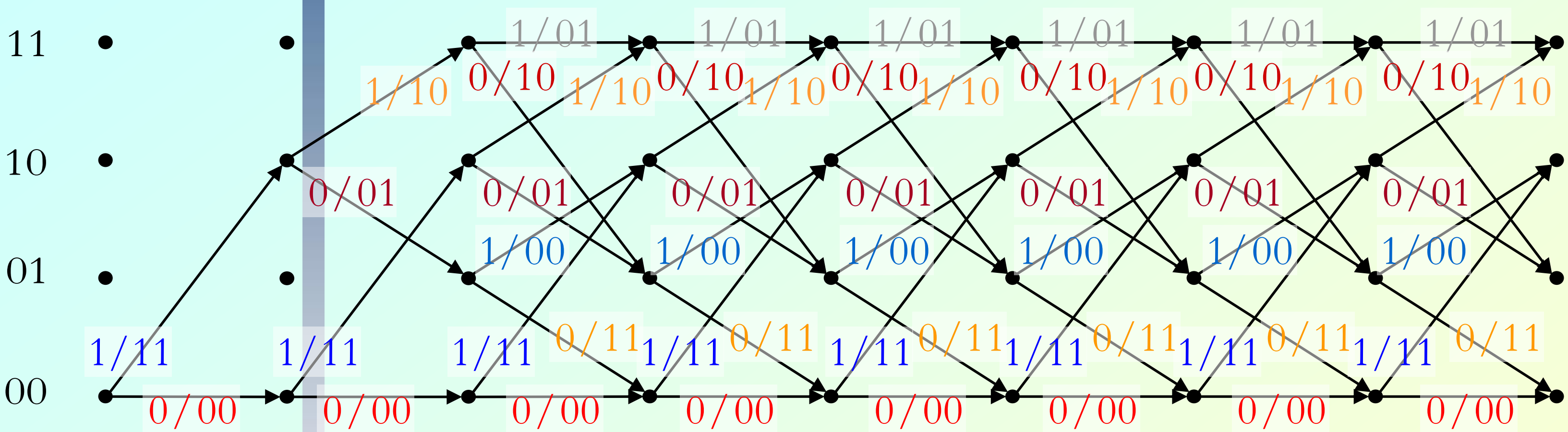


A vett bitsorozat: 01 00 01 01 00 10 01 10.

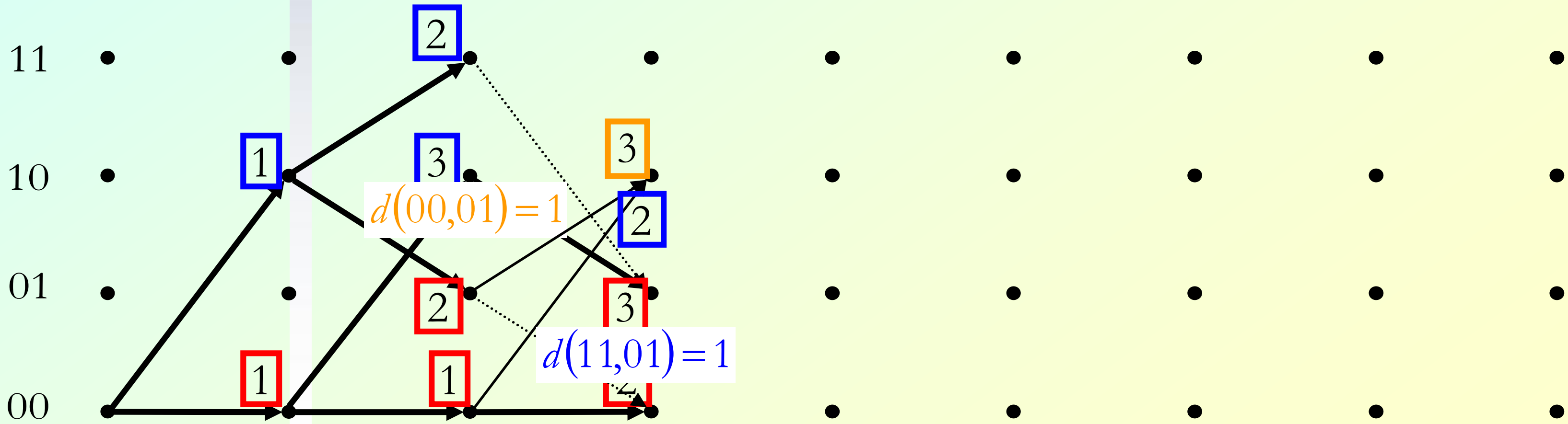


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

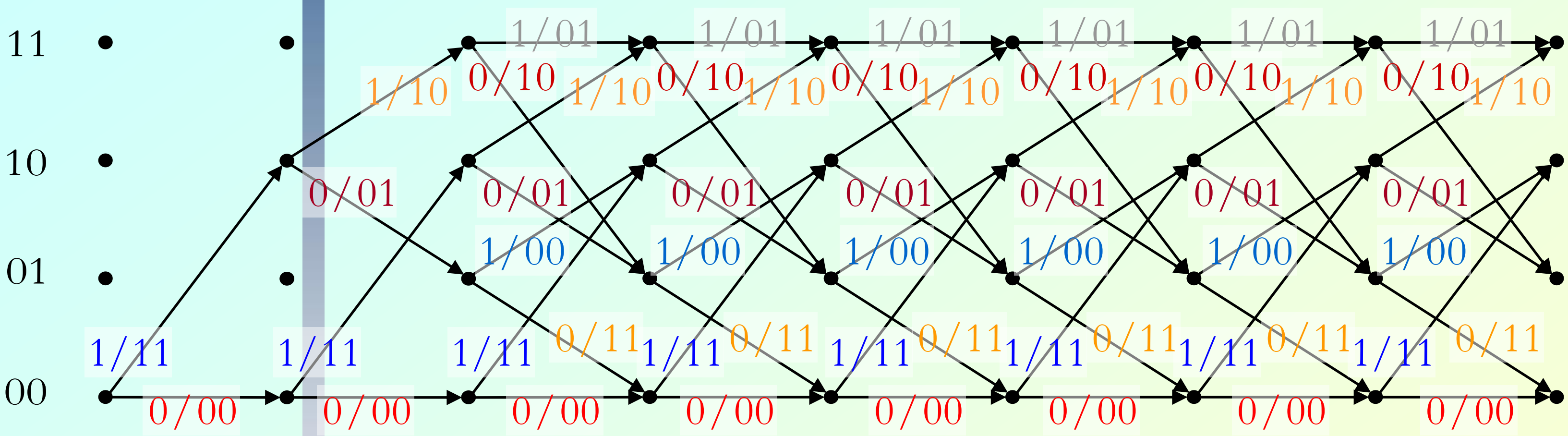


A vett bitsorozat: 01 00 01 01 00 10 01 10.

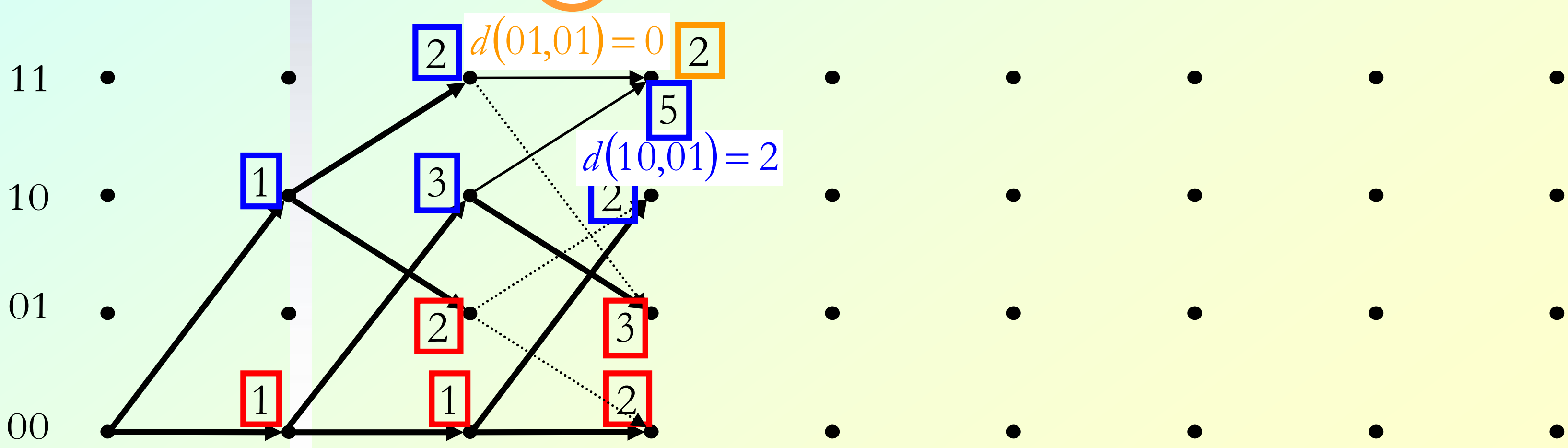


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

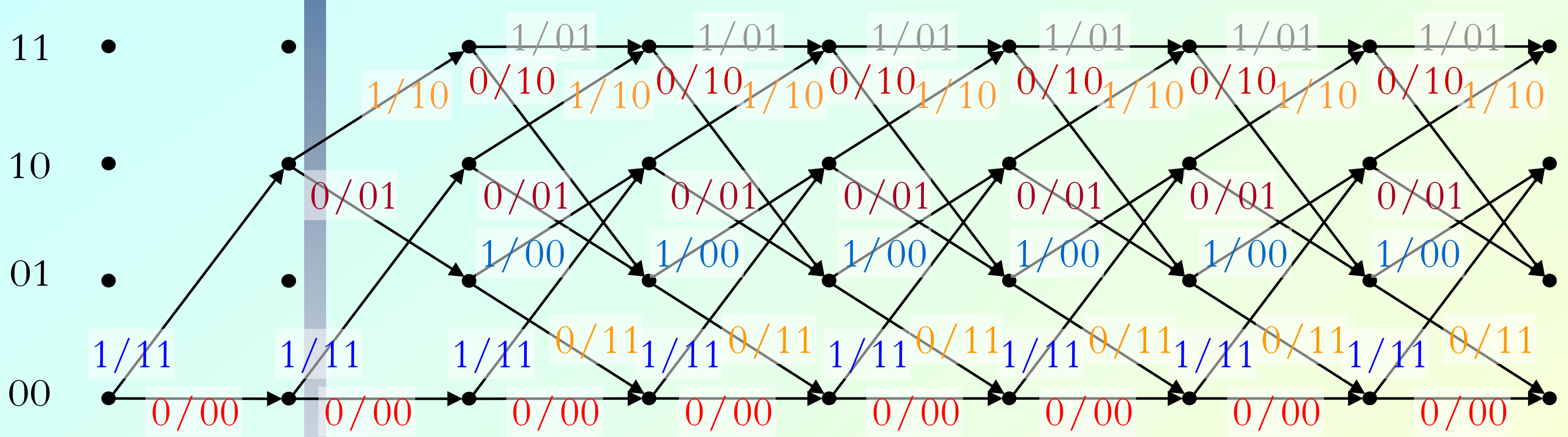


A vett bitsorozat: 01 00 01 01 00 10 01 10.

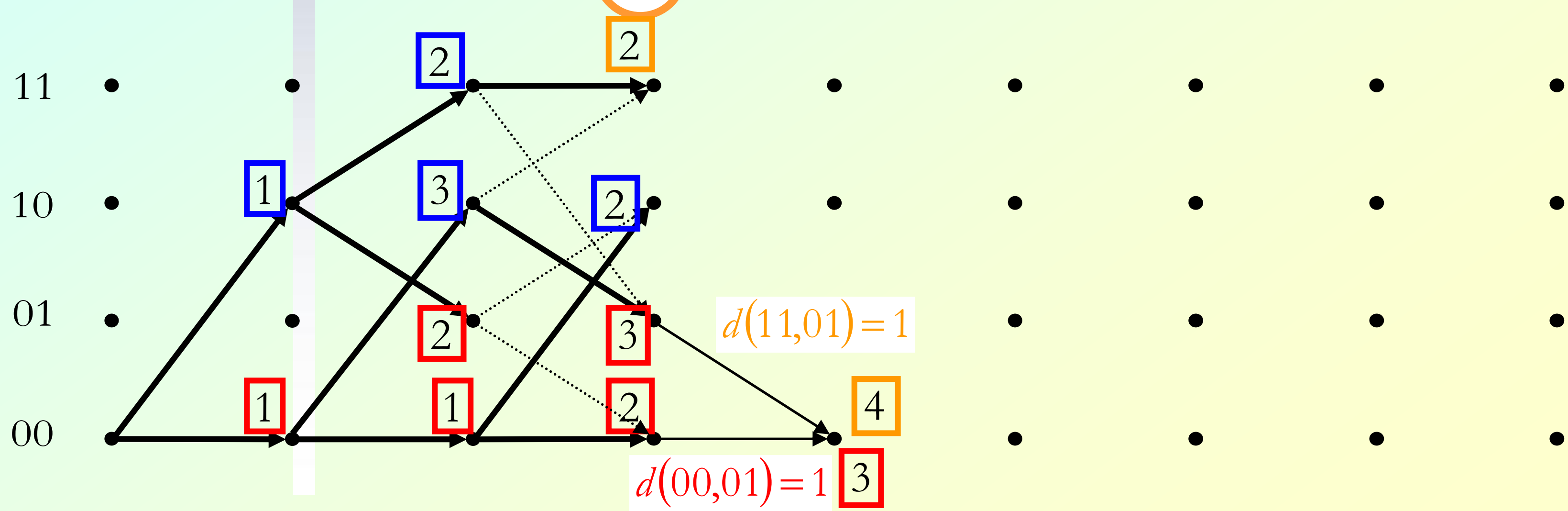


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

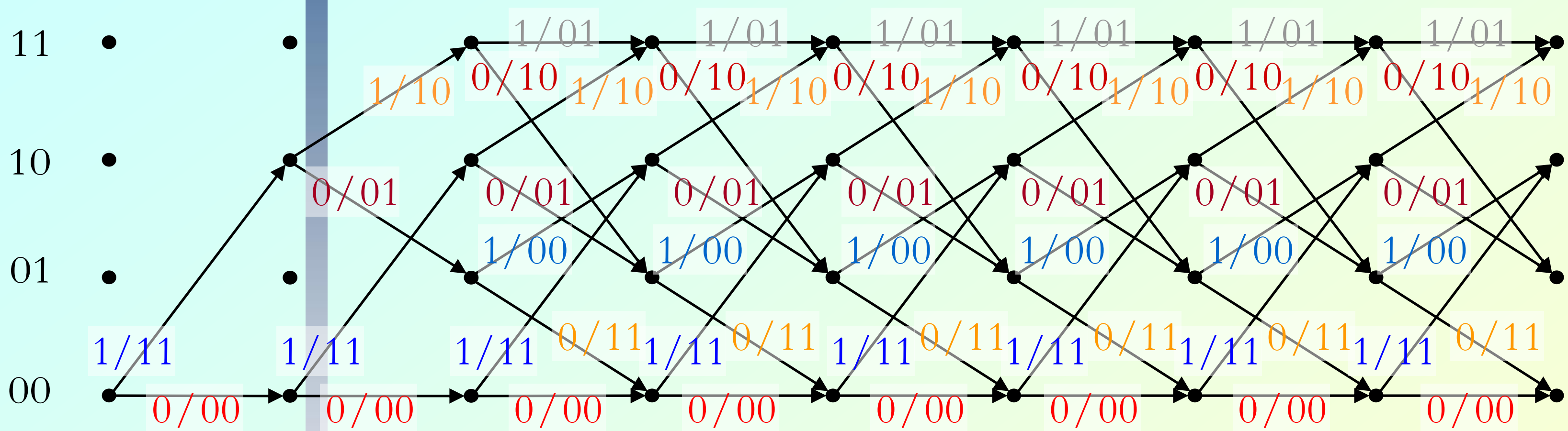


A vett bitsorozat: 01 00 01 01 00 10 01 10.

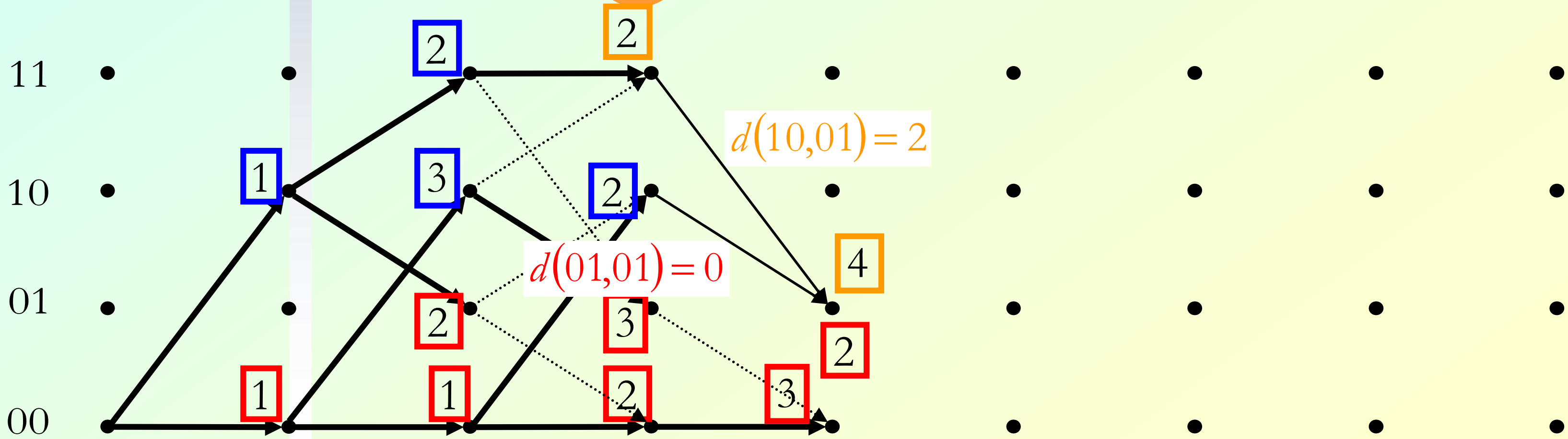


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

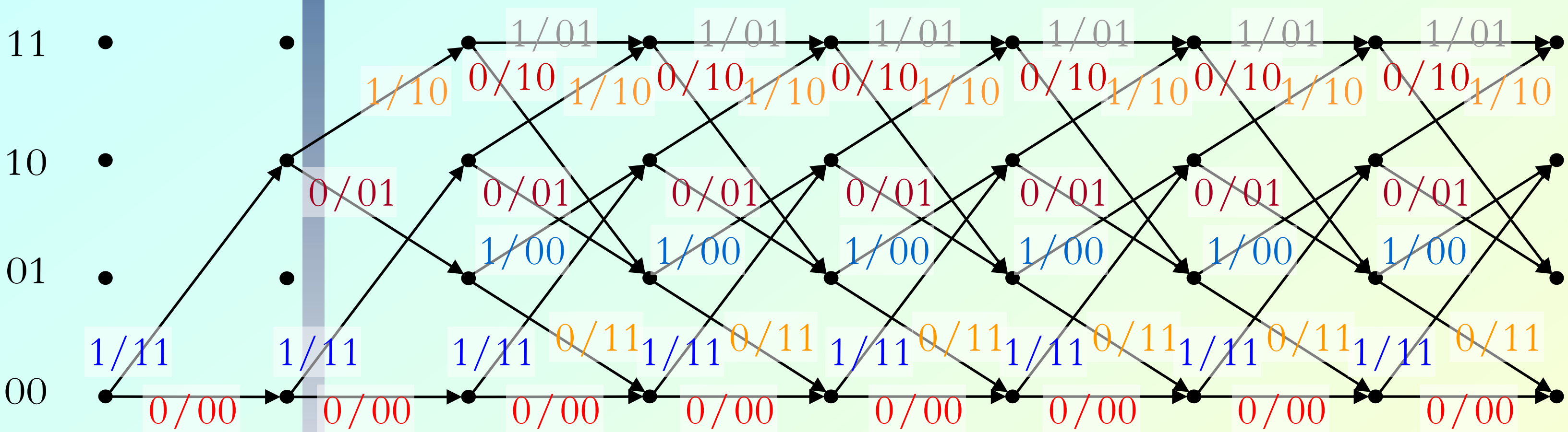


A vett bitsorozat: 01 00 01 01 00 10 01 10.

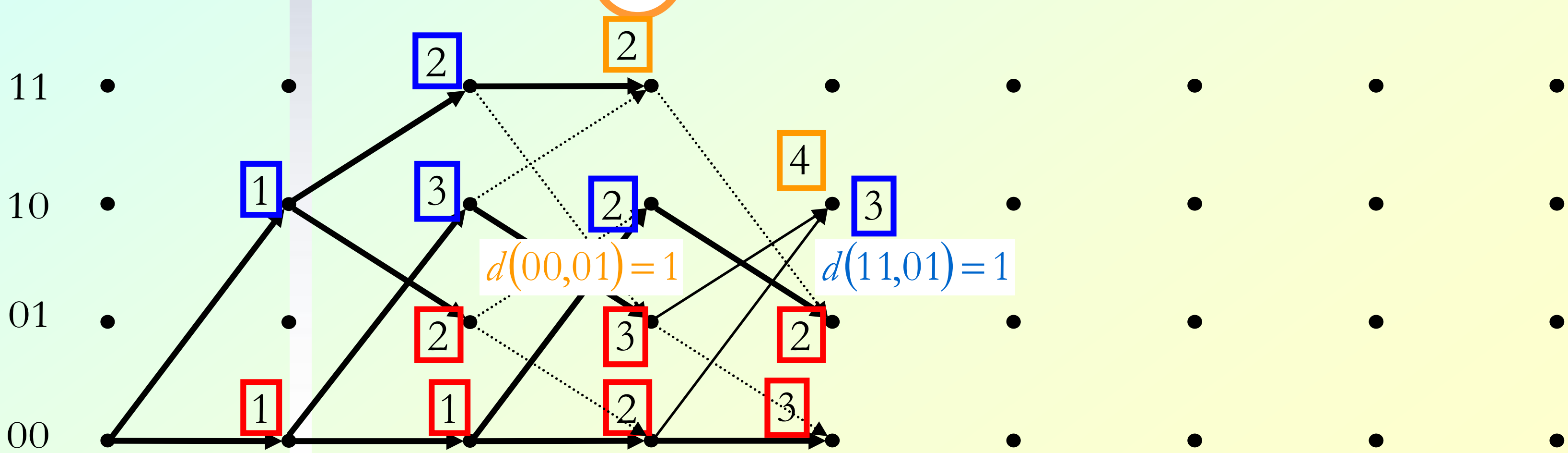


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

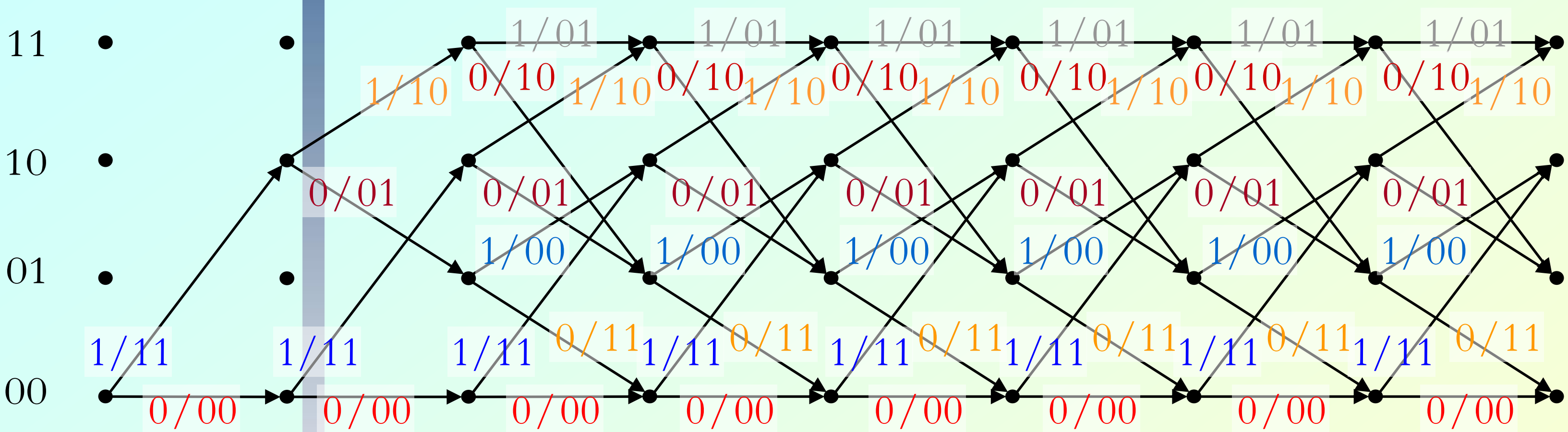


A vett bitsorozat: 01 00 01 01 00 10 01 10.

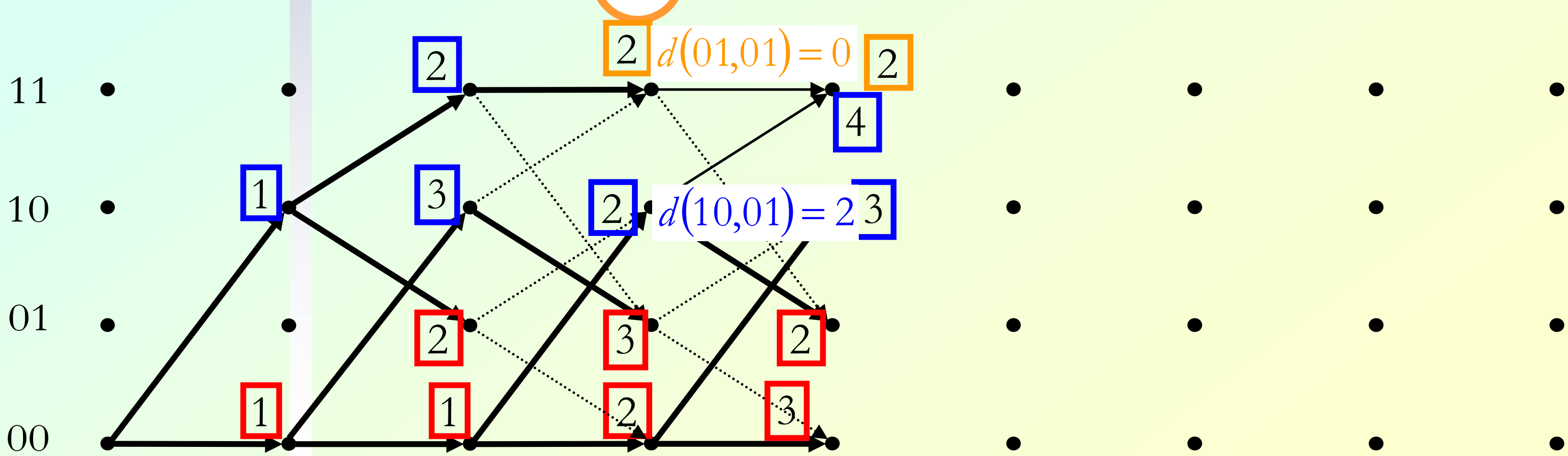


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

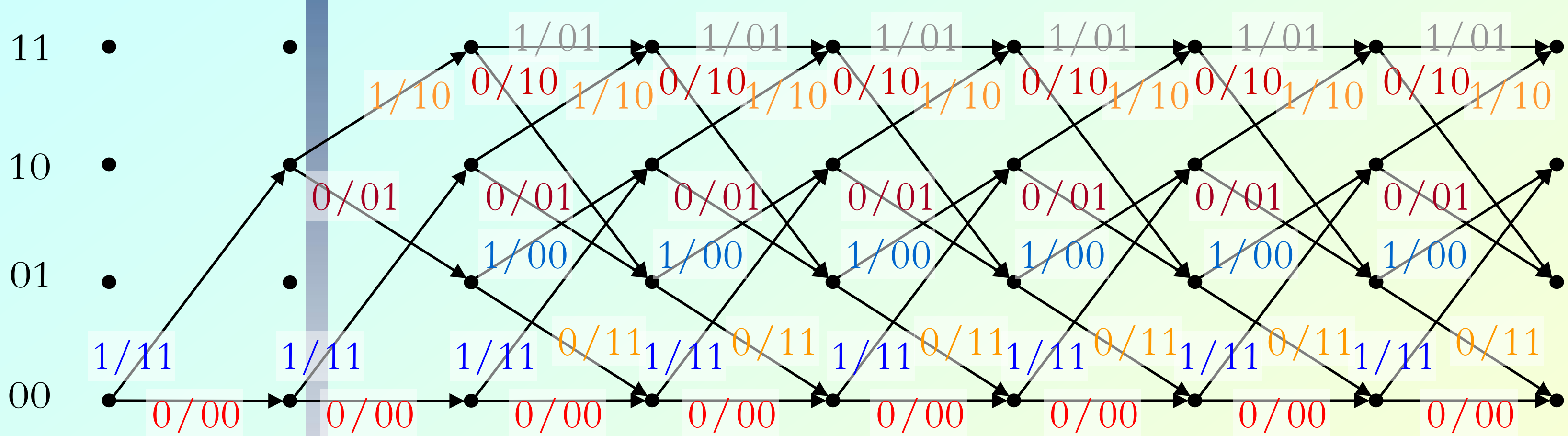


A vett bitsorozat: 01 00 01 01 00 10 01 10.

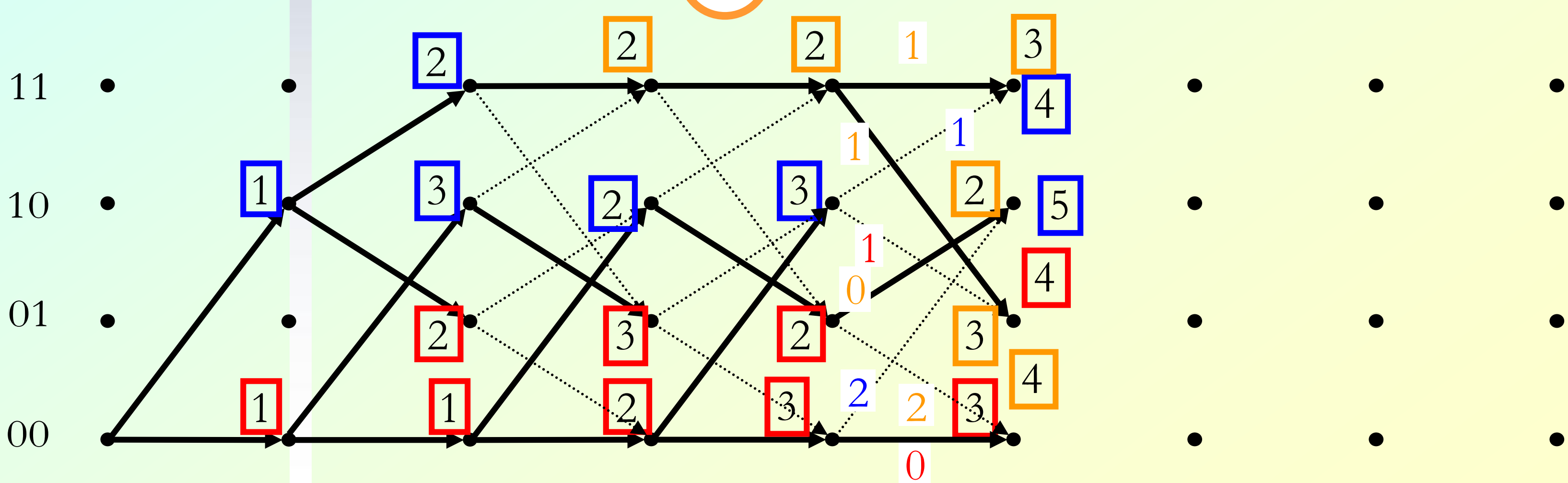


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

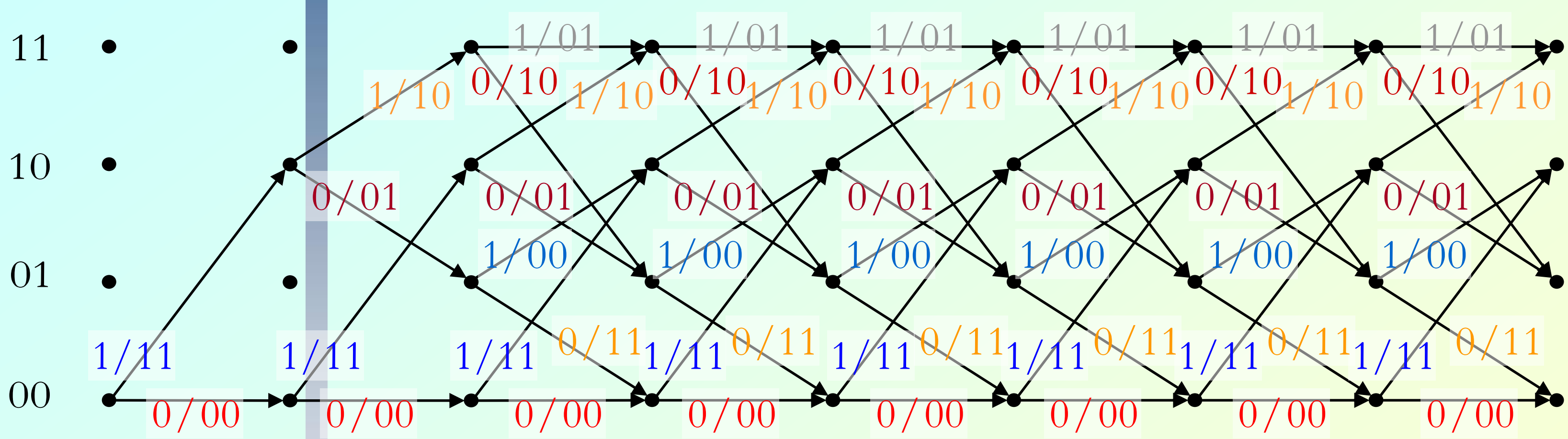


A vett bitsorozat: 01 00 01 01 00 10 01 10.

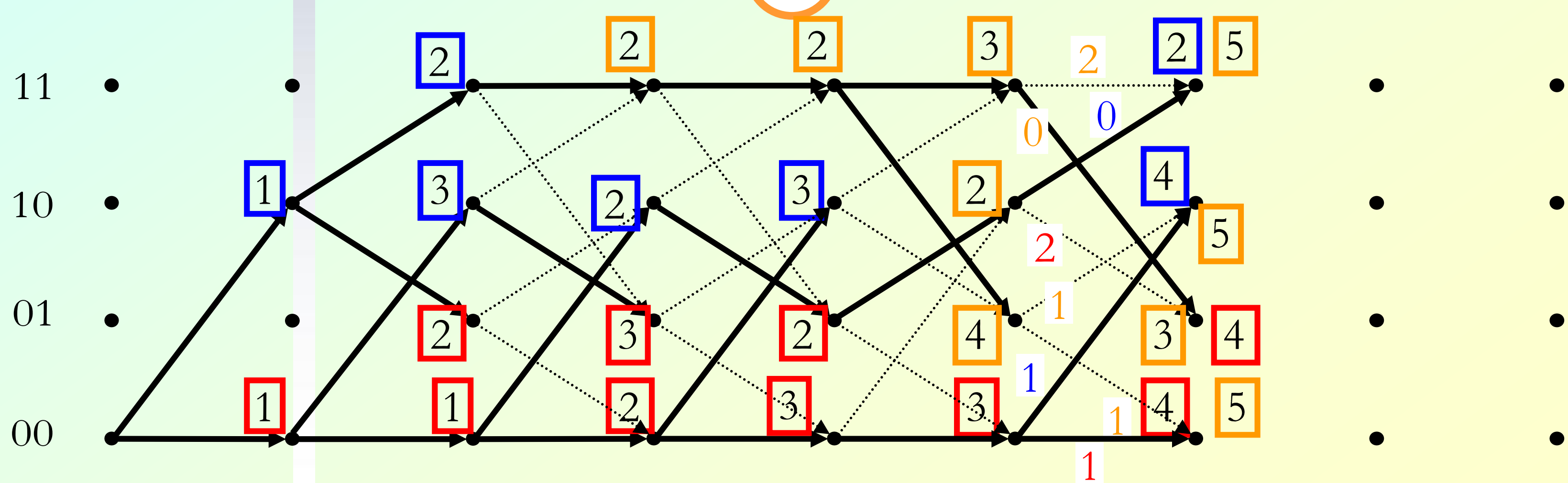


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

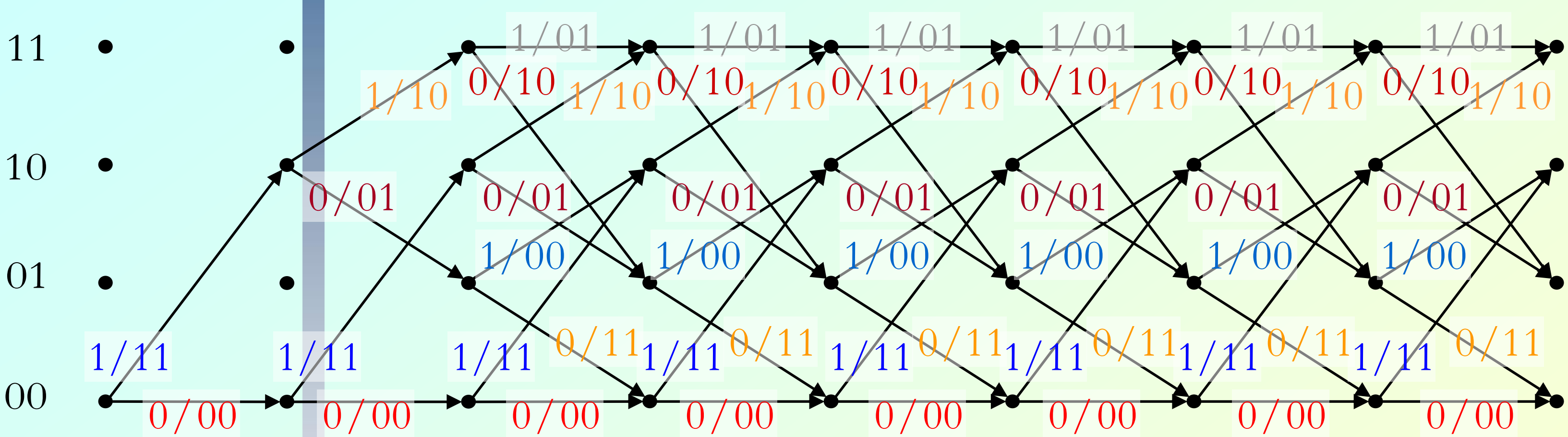


A vett bitsorozat: 01 00 01 01 00 10 01 10.

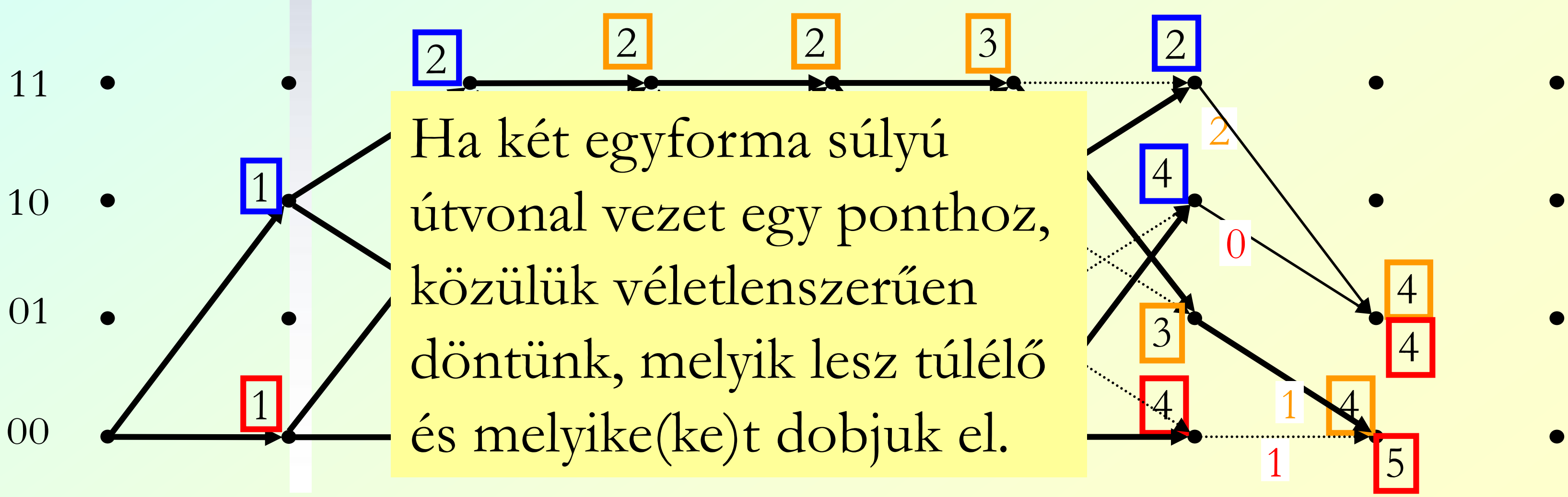


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:



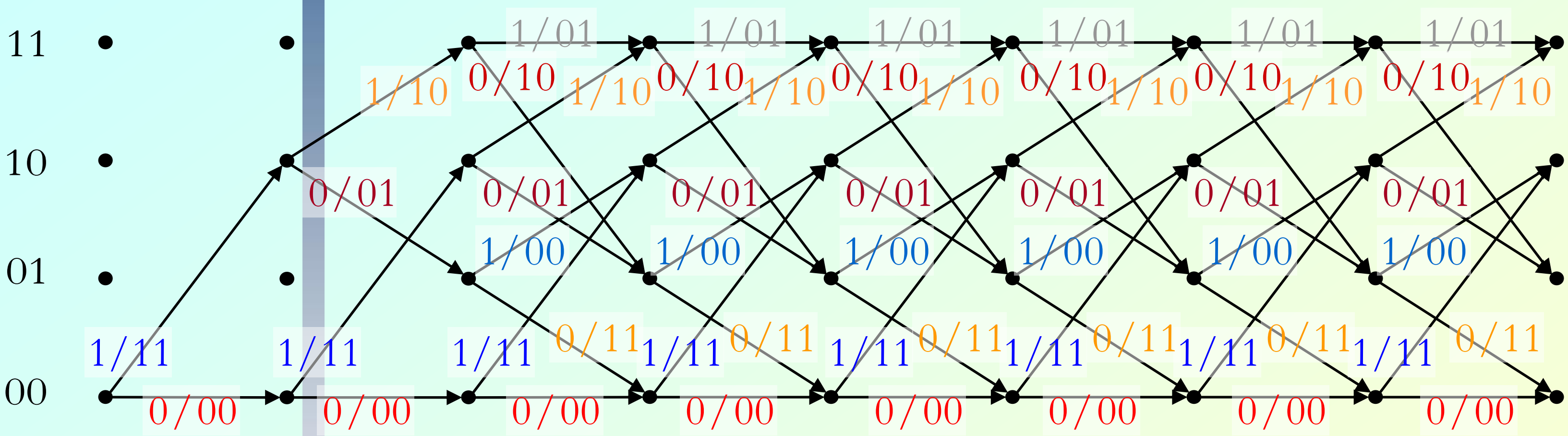
A vett bitsorozat: 01 00 01 01 00 10 01 10.



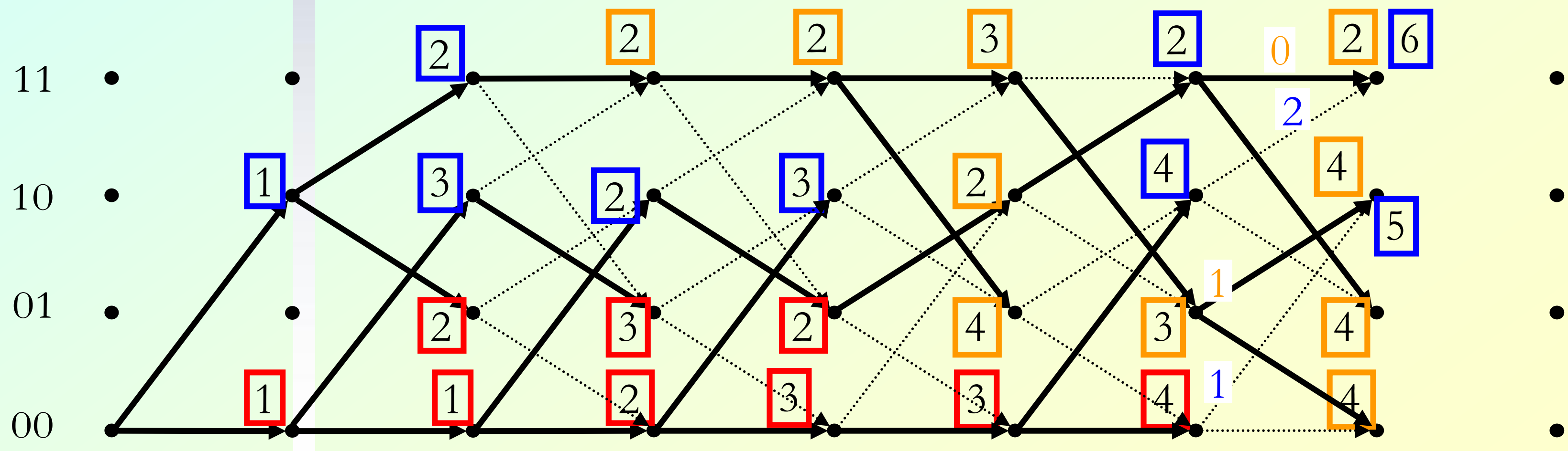
Ha két egyforma súlyú útvonal vezet egy ponthoz, közülük véletlenszerűen döntünk, melyik lesz túlélő és melyike(ke)t dobjuk el.

A Viterbi-dekódolás működése

Emlékeztetőül a trellis:

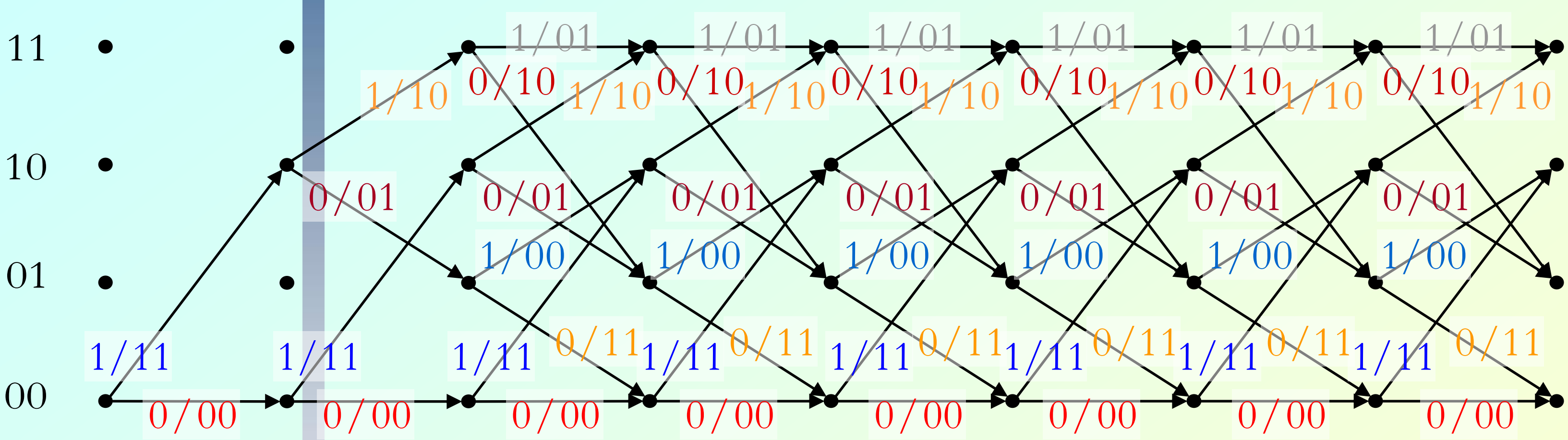


A vett bitsorozat: 01 00 01 01 00 10 01 10.

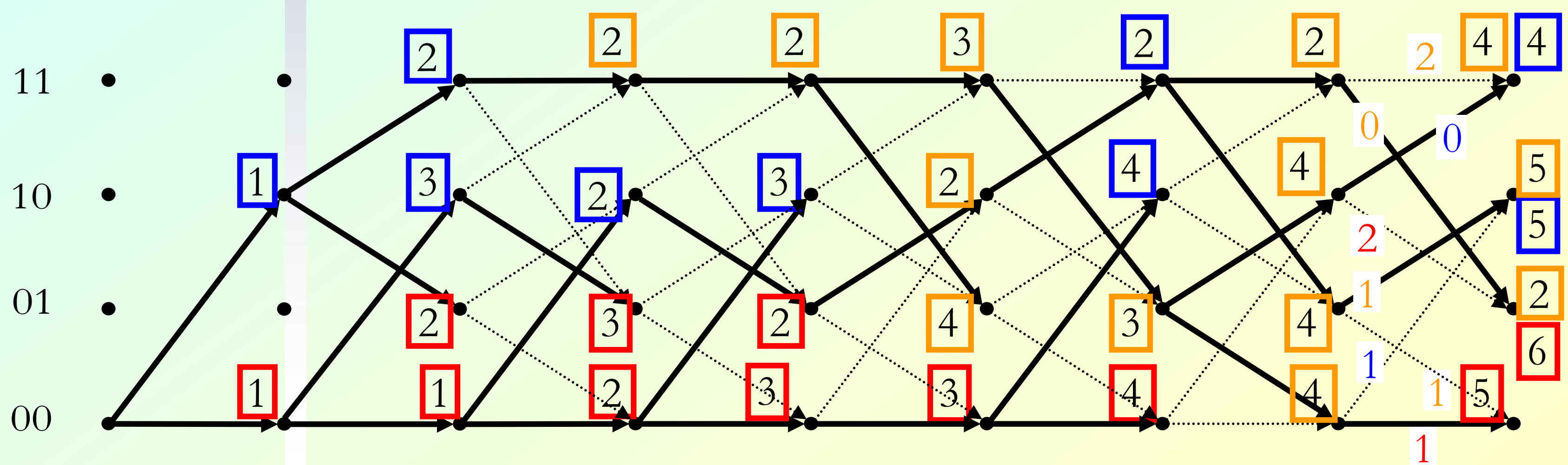


A Viterbi-dekódolás működése

Emlékeztetőül a trellis:



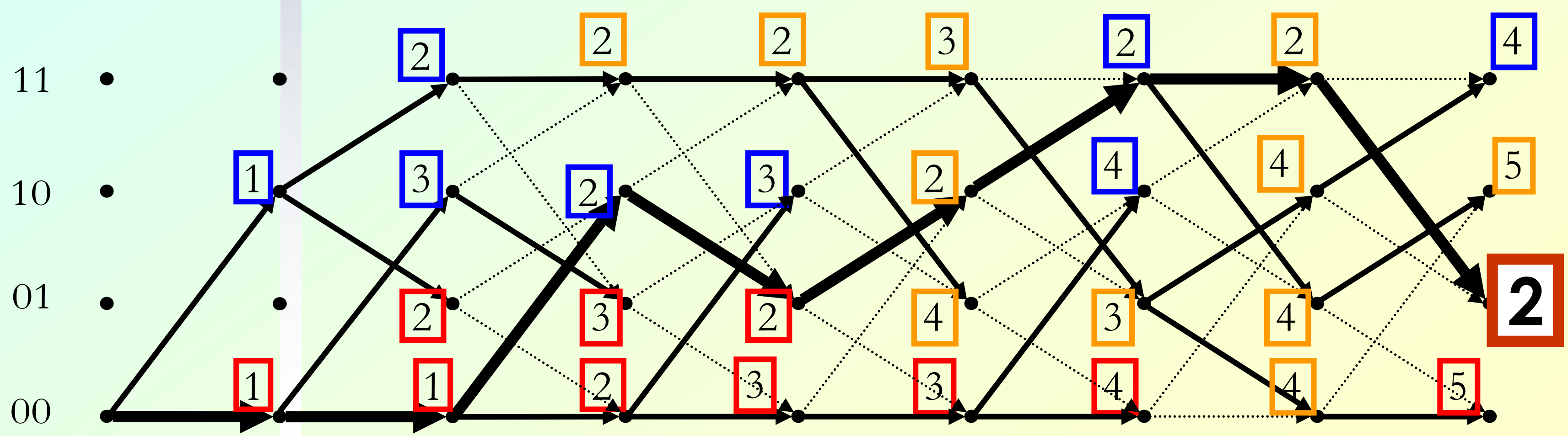
A vett bitsorozat: 01 00 01 01 00 10 01 10.



A Viterbi-dekódolás működése

A kapott túlélő útvonalak közül kiválasztjuk a minimális súlyút:
a 2 súlyú utat. Az útvonal és a trellis ismeretében az üzenet visszakapható:
0 0 1 0 1 1 1 0

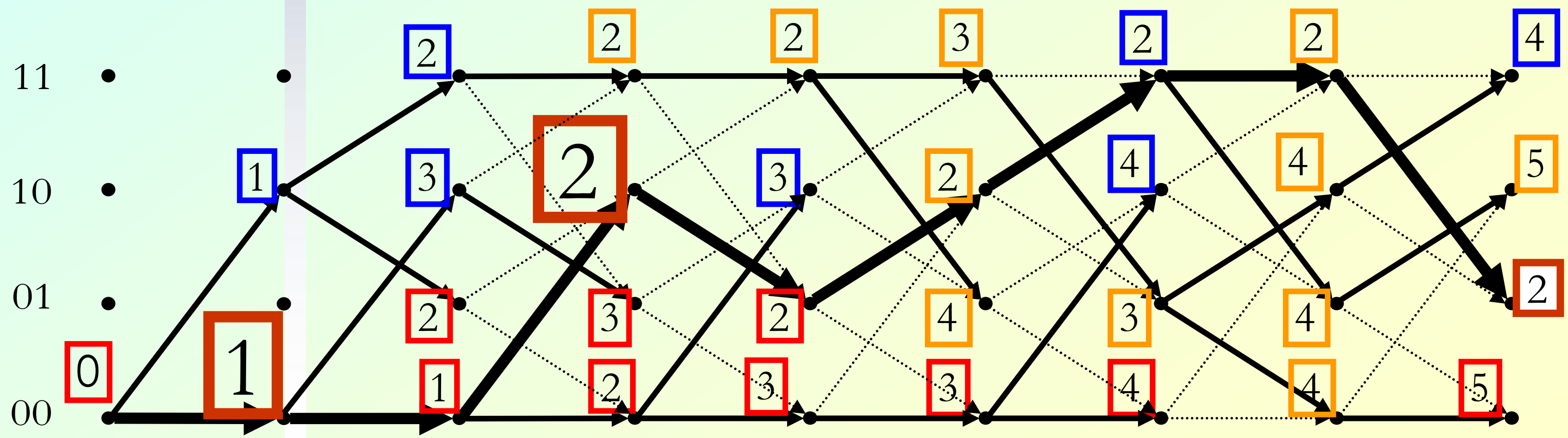
- Mivel minden pontba <u>csak egy túlélő él fut be</u>, a hibajavítás egyértelmű



A Viterbi-dekódolás működése

A kapott túlélő útvonalak közül kiválasztjuk a minimális súlyút:
a 2 súlyú utat. Az útvonal és a trellis ismeretében az üzenet visszakapható:
0 0 1 0 1 1 1 0

- Felderíthető, <u>hol hibázott a csatorna</u>: amely kódszókeretnél nőtt az összsúly, annak továbbításakor volt rontás.



Visszacsatolt konvolúciós kódolók

Ha egy konvolúciós kódoló léptetőregisztereinek kimenetét valamiféle lineáris eszközön keresztül visszacsatoljuk a bemenetre, **rekurzív konvolúciós kód**ot kapunk.

Szintén lineáris, időinvariáns fa-kódok, ám a kényszerhossz többnyire végtelen.

Lehet állapotátmeneti gráfot és trellist készíteni hozzájuk, Viterbi-algoritmussal dekódolhatók.

Visszacsatolt konvolúciós kódolók

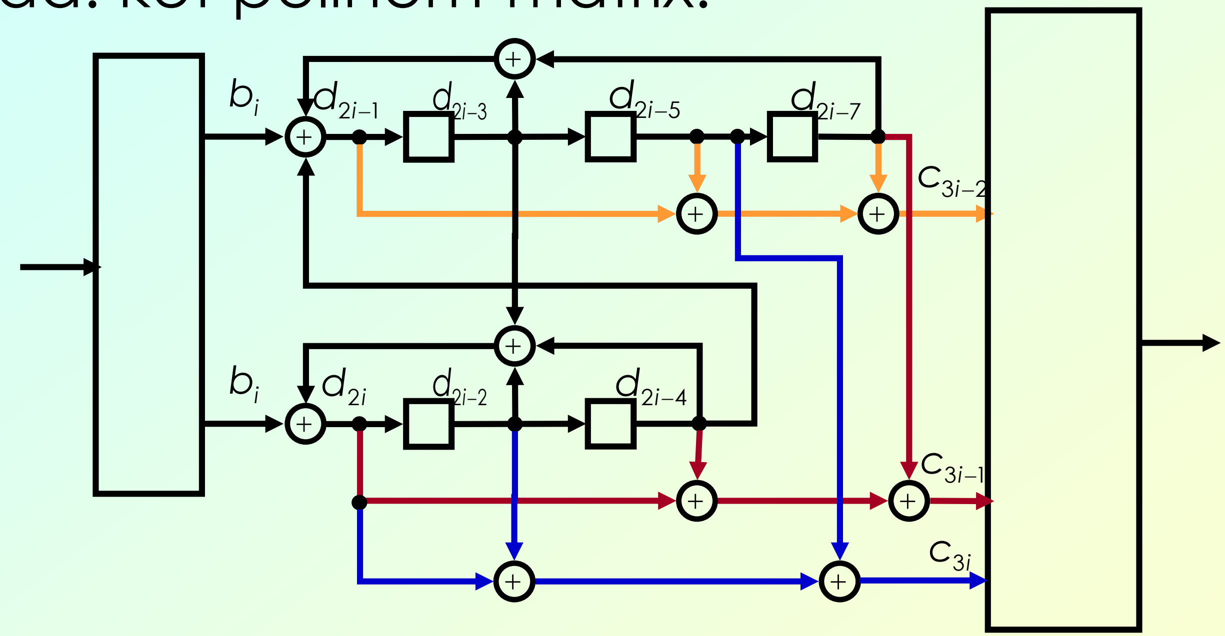
Két polinom-mátrix van, az előre irányú és a visszacsatolt ágnak is egy-egy:

$$\mathbf{G}(t)=\begin{pmatrix} g_{1,1}(t) & g_{1,2}(t) & \cdots & g_{1,n}(t) \\ g_{2,1}(t) & g_{2,2}(t) & \cdots & g_{2,n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,1}(t) & g_{k,2}(t) & \cdots & g_{k,n}(t) \end{pmatrix}$$

$$\mathbf{H}(t)=\begin{pmatrix} h_{1,1}(t) & h_{1,2}(t) & \cdots & h_{1,k}(t) \\ h_{2,1}(t) & h_{2,2}(t) & \cdots & h_{2,k}(t) \\ \vdots & \vdots & \ddots & \vdots \\ h_{k,1}(t) & h_{k,2}(t) & \cdots & h_{k,k}(t) \end{pmatrix}$$

Visszacsatolt konvolúciós kódolók

Példa: két polinom-mátrix:

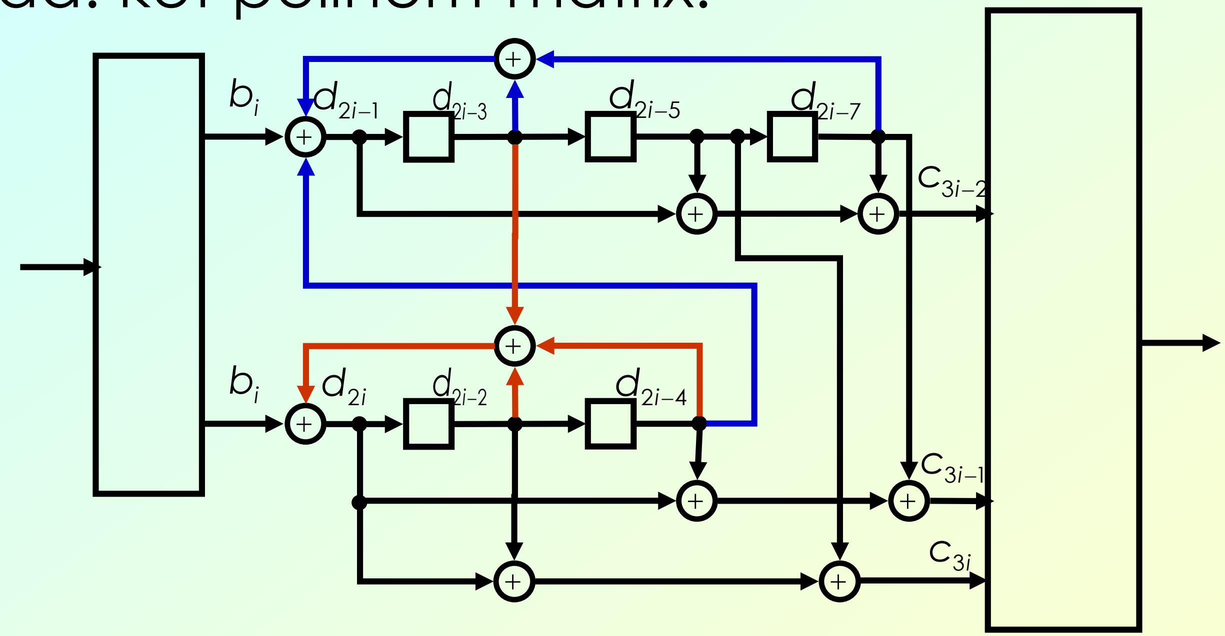


előre:

$$\mathbf{G}(t) = \begin{pmatrix} t^3 + t^2 + 1 & t^3 & t^2 \\ 0 & t^2 + 1 & t + 1 \end{pmatrix}$$

Visszacsatolt konvolúciós kódolók

Példa: két polinom-mátrix:



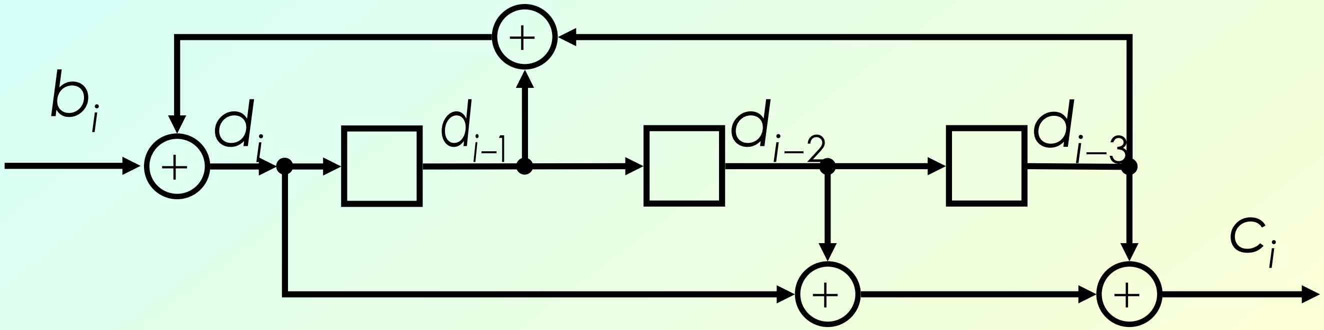
vissza:

$$\mathbf{H}(t) = \begin{pmatrix} t^3 + t & t \\ t^2 & t^2 + t \end{pmatrix}$$

Visszacsatolt konvolúciós kódolók

Példa: a két polinom-mátrix:

$$\mathbf{H}(t) = \left(t + t^3\right)$$



$$\mathbf{G}(t) = \left(1 + t^2 + t^3\right)$$

az együtthatók:

$$h_{1,1,0} = 0 \quad h_{1,1,1} = 1 \quad h_{1,1,2} = 0 \quad h_{1,1,3} = 1$$

$$g_{1,1,0} = 1 \quad g_{1,1,1} = 0 \quad g_{1,1,2} = 1 \quad g_{1,1,3} = 1$$

Visszacsatolt konvolúciós kódolók

A bemenet (üzenetkeret) polinom-vektora:

$$\mathbf{b}(t) = \begin{pmatrix} b_1(t) & b_2(t) & \dots & b_k(t) \end{pmatrix}$$

a kimenet (kódszókeret) polinom-vektora:

$$\mathbf{c}(t) = \begin{pmatrix} c_1(t) & c_2(t) & \dots & c_n(t) \end{pmatrix}$$

Egy adott időpillanatban tárolók állapotait leíró polinom-vektor:

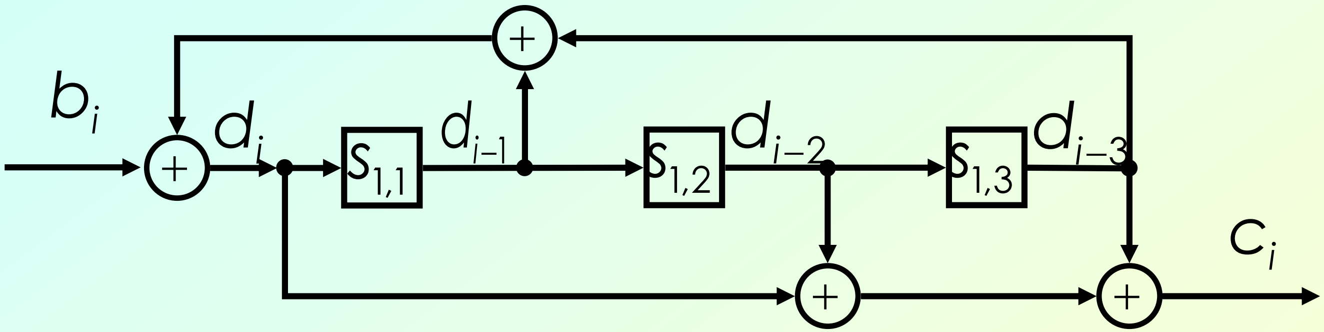
$$\mathbf{s}(t) = \begin{pmatrix} s_1(t) & s_2(t) & \dots & s_k(t) \end{pmatrix}$$

ahol

$$s_j(t) = s_{j,0} + s_{j,1}t + s_{j,2}t^2 + \dots$$

Visszacsatolt konvolúciós kódolók

Példa: az állapotpolinom-vektor



$$\mathbf{s}(t)=\left(s_1(t)\quad s_2(t)\quad \ldots \quad s_k(t)\right)$$

ahol $s_i(t)=\sum_{j=0}^{\infty} s_{i,j}t^j$

azaz, mivel $s_{i,j}=b_{i,j}+\sum_{r=1}^{m}\sum_{p=1}^{k}h_{p,i,r}s_{p,j-r}$ $\quad \Big| \cdot t^j, \sum_{j=0}^{\infty}$

$$s_i(t)=\sum_{j=0}^{\infty}\left(b_{i,j}+\sum_{r=1}^{m}\sum_{p=1}^{k}h_{p,i,r}s_{p,j-r}\right)t^j$$

Visszacsatolt konvolúciós kódolók

Az állapotpolinom-vektor

$$s_i(t) = \sum_{j=0}^{\infty}\left(b_{i,j} + \sum_{r=1}^{m}\sum_{p=1}^{k} h_{p,i,r} s_{p,j-r}\right)t^j =$$

$$= \sum_{j=0}^{\infty} b_{i,j}t^j + \sum_{j=0}^{\infty}\sum_{r=1}^{m}\sum_{p=1}^{k} h_{p,i,r} t^r s_{p,j-r} t^{j-r} =$$

$$= \sum_{j=0}^{\infty} b_{i,j}t^j + \sum_{r=1}^{m}\sum_{p=1}^{k} h_{p,i,r} t^r \underbrace{\sum_{j=0}^{\infty} s_{p,j-r} t^{j-r}}_{s_p(t)} =$$

$$= b_i(t) + \sum_{p=1}^{k}\underbrace{\sum_{r=1}^{m} h_{p,i,r} t^r}_{h_{p,i}(t)} s_p(t)$$

Visszacsatolt konvolúciós kódolók

Az állapotpolinom-vektor

$$s_i(t) = b_i(t) + \sum_{p=1}^{k} h_{p,i}(t) s_p(t)$$

ebből

$$\mathbf{s}(t) = \mathbf{b}(t) \cdot \mathbf{H}^{-1}(t).$$

A kimenet a

$$\mathbf{c}(t) = \mathbf{s}(t) \cdot \mathbf{G}(t)$$

szerint:

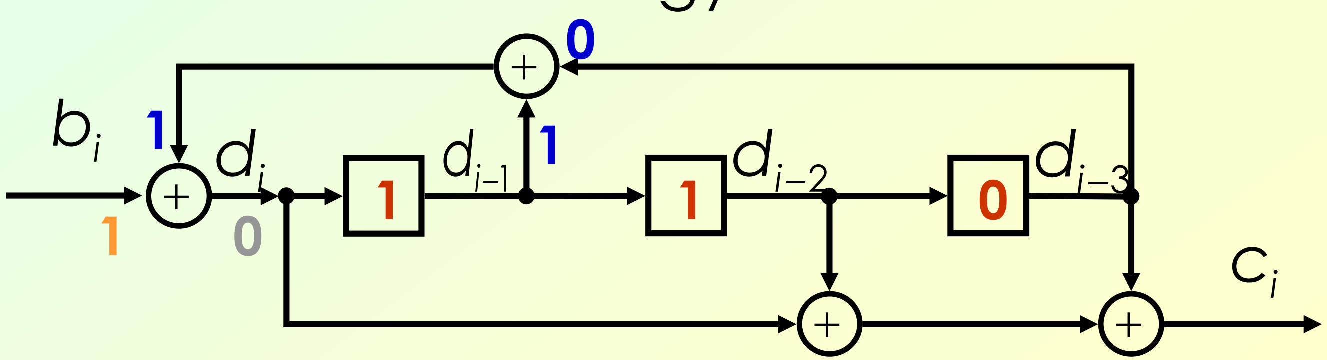
$$\mathbf{c}(t) = \mathbf{b}(t) \cdot \mathbf{H}^{-1}(t) \cdot \mathbf{G}(t)$$

Visszacsatolt konvolúciós kódolók

Ha k=1, akkor

$$\mathbf{c}(t)=\frac{b_1(t)}{h_{1,1}(t)}\cdot\mathbf{G}(t)=\frac{b_1(t)}{h_{1,1}(t)}\cdot\begin{pmatrix}g_{11}(t) & g_{12}(t) & \dots & g_{1n}(t)\end{pmatrix}$$

Blokk-kódoló üzemmód itt is lehetséges, csak nem 0 bitekkel kell kiüríteni a tárolókat, hanem olyanokkal, amelyek az aktuális visszacsatolással együtt adnak 0-t.

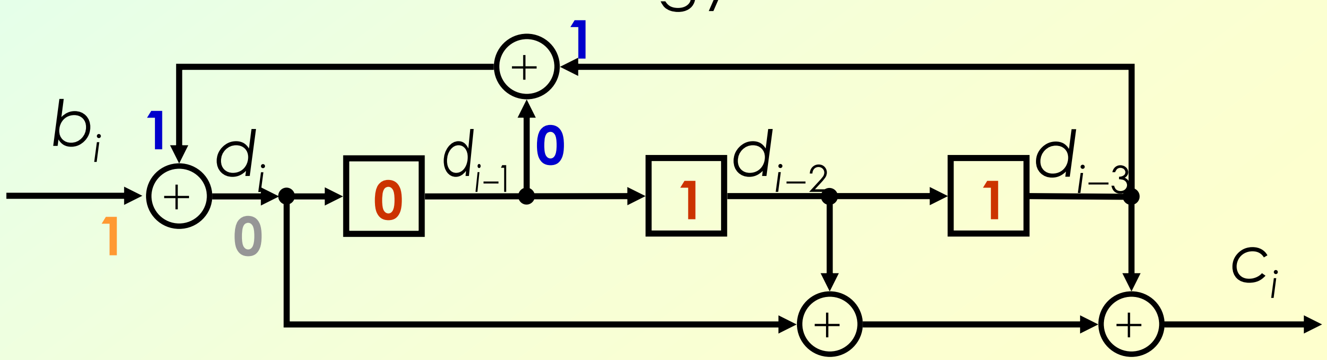


Visszacsatolt konvolúciós kódolók

Ha k=1, akkor

$$\mathbf{c}(t)=\frac{b_1(t)}{h_{1,1}(t)}\cdot\mathbf{G}(t)=\frac{b_1(t)}{h_{1,1}(t)}\cdot\begin{pmatrix}g_{11}(t) & g_{12}(t) & \ldots & g_{1n}(t)\end{pmatrix}$$

Blokk-kódoló üzemmód itt is lehetséges, csak nem 0 bitekkel kell kiüríteni a tárolókat, hanem olyanokkal, amelyek az aktuális visszacsatolással együtt adnak 0-t.

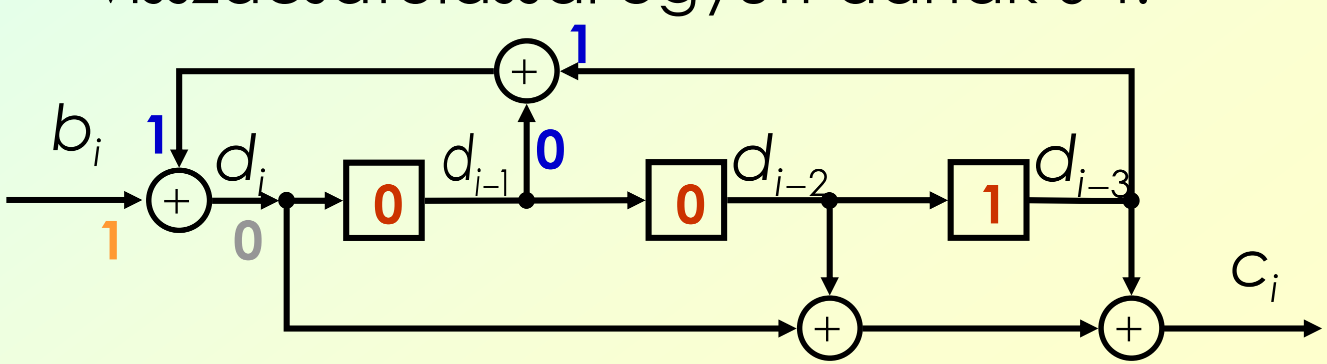


Visszacsatolt konvolúciós kódolók

Ha k=1, akkor

$$\mathbf{c}(t)=\frac{b_1(t)}{h_{1,1}(t)}\cdot\mathbf{G}(t)=\frac{b_1(t)}{h_{1,1}(t)}\cdot\left(g_{11}(t)\quad g_{12}(t)\quad\ldots\quad g_{1n}(t)\right)$$

Blokk-kódoló üzemmód itt is lehetséges, csak nem 0 bitekkel kell kiüríteni a tárolókat, hanem olyanokkal, amelyek az aktuális visszacsatolással együtt adnak 0-t.

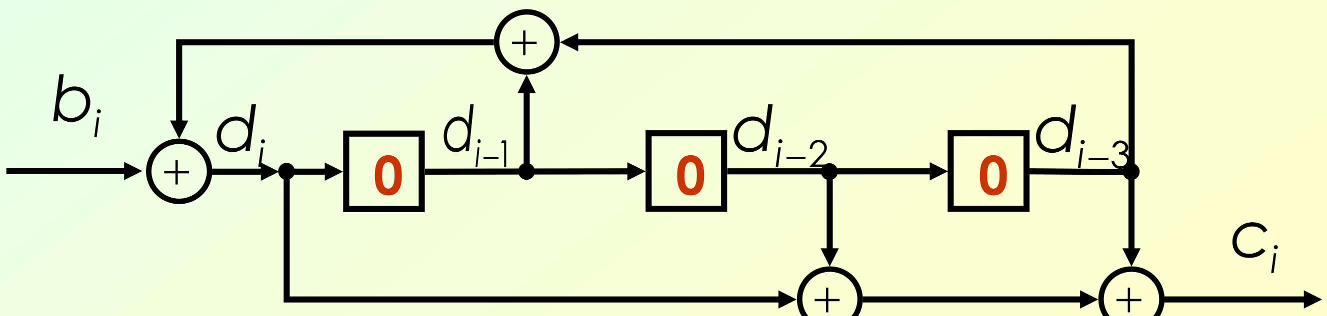


Visszacsatolt konvolúciós kódolók

Ha k=1, akkor

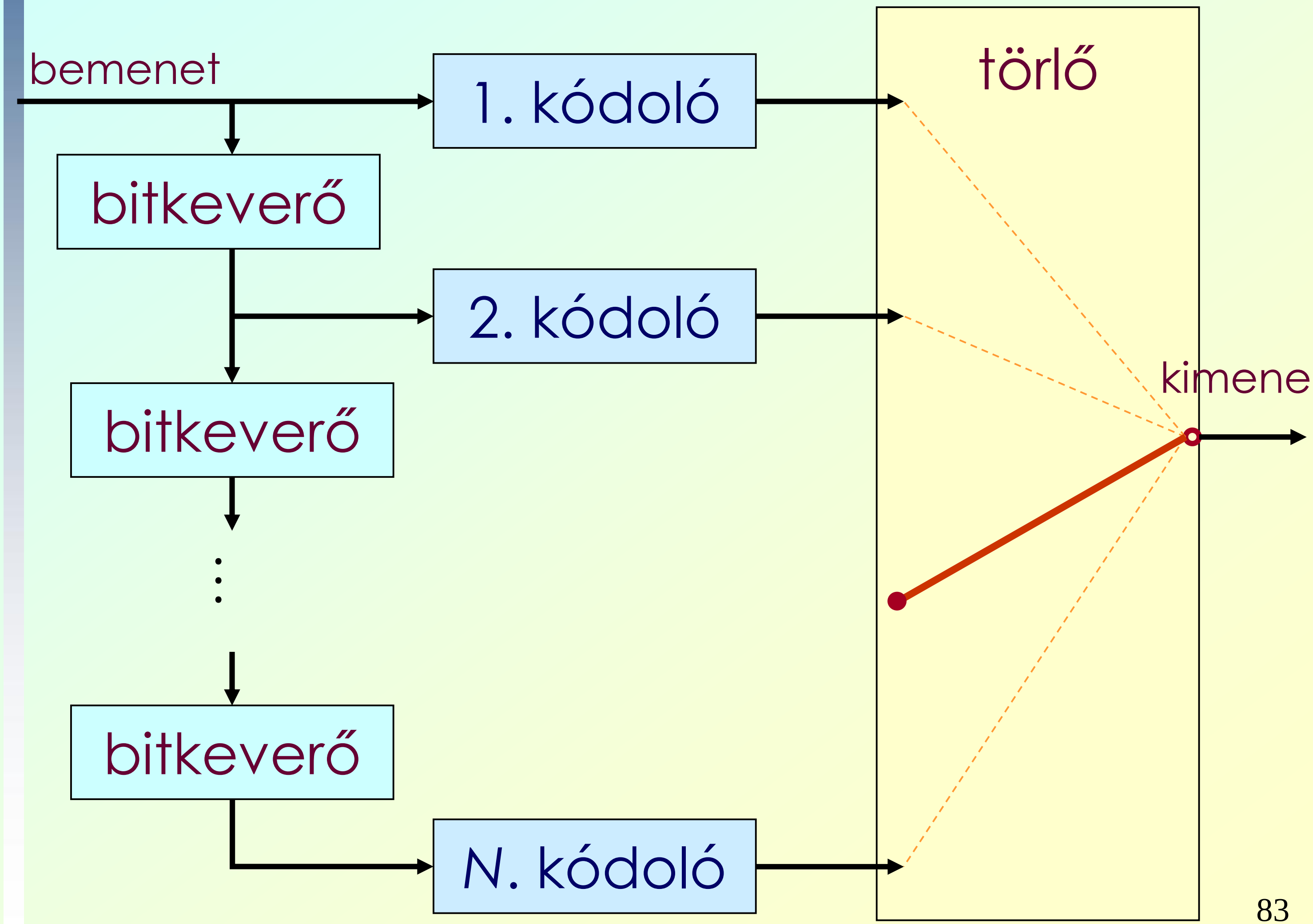
$$\mathbf{c}(t)=\frac{b_1(t)}{h_{1,1}(t)}\cdot\mathbf{G}(t)=\frac{b_1(t)}{h_{1,1}(t)}\cdot\begin{pmatrix}g_{11}(t) & g_{12}(t) & \ldots & g_{1n}(t)\end{pmatrix}$$

Blokk-kódoló üzemmód itt is lehetséges, csak nem 0 bitekkel kell kiüríteni a tárolókat, hanem olyanokkal, amelyek az aktuális visszacsatolással együtt adnak 0-t.



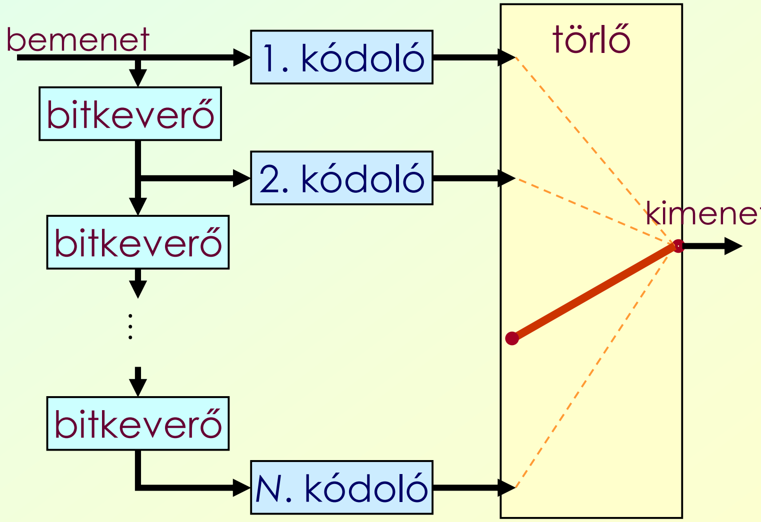
Turbó kódok

A turbó kódolók általános felépítése:



Turbó kódok

- A turbó kódokat Berrou, Glavieux és Thitimajshima fedezte fel.
- Az egyes kódoló ágak általában visszacsatolt konvolúciós kódolók
- 1/N jelsebesség ez a törlővel javítható
- A javított jelsebesség egészen közel van az elméleti maximumához, a csatornakapacitáshoz.



Turbó kódok

A komponenskódokat külön-külön dekódolják, de felhasználják a többi dekódoló eredményeit.

Akár nagyságrendekkel kisebb hibavalószínűség, mint ha csak egyetlen komponens-kódolót használnának

Ha bittörlés történik, annak mintázata a vevőben is ismert, ott vagy törléses hibaként kezelik őket, vagy nullákkal helyettesítik

Ha a komponenskódolók konvolúciós kódolók, akkor is blokk-kódoló üzemmódban működnek, a bitkeverővel megegyező szóhosszal (csak az első)

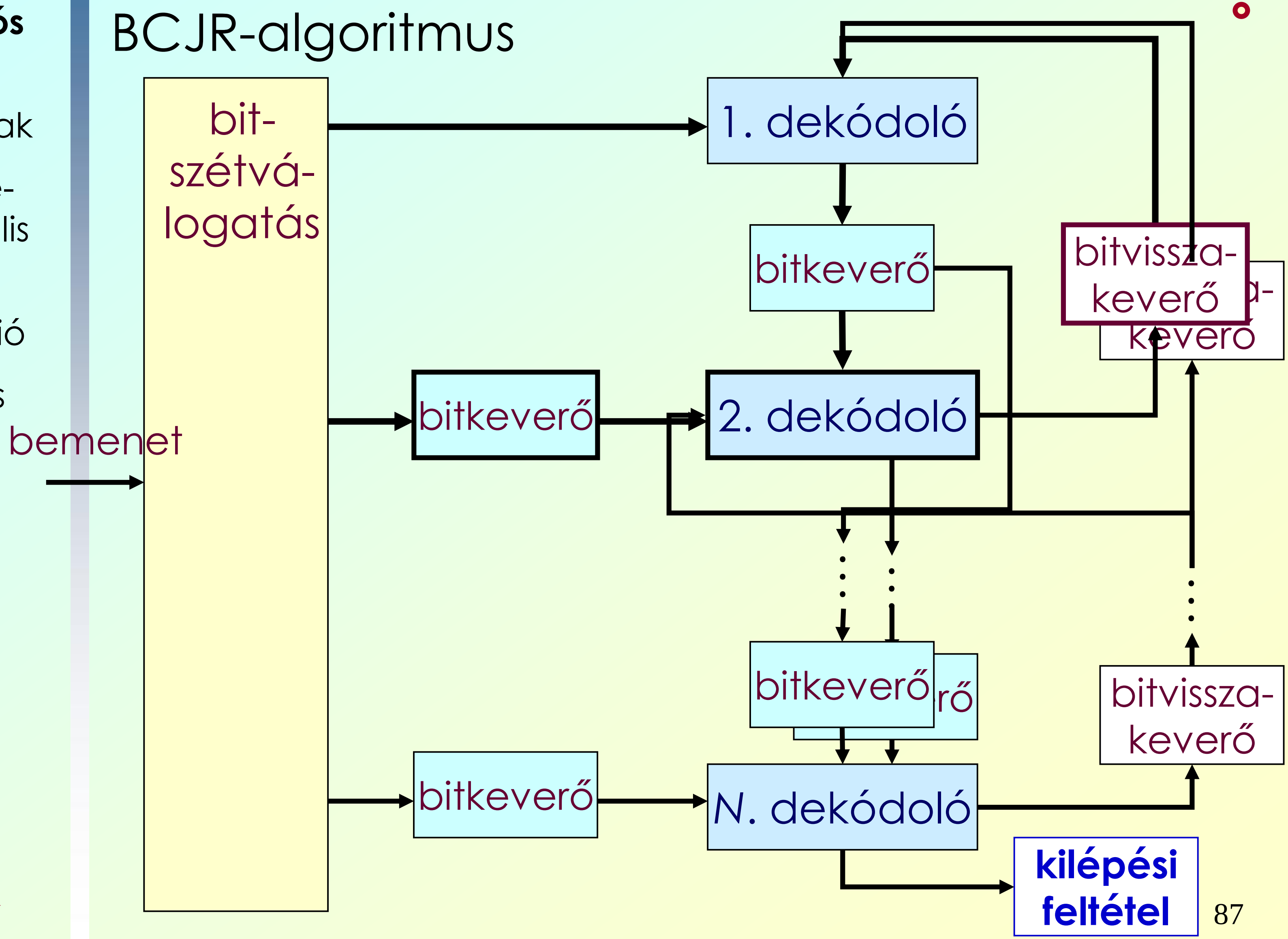
Turbó kódok dekódolása

A maximum likelihood dekódolás kellően nagy blokkhosszra már gazdaságalan.

BCJR-algoritmus (Bahl, Cocke, Jelinek, Raviv) és továbbfejlesztett verziói iterációk során dekódolnak: minden dekódoló megkapja a saját bitjeit a csatornáról és a többi kódoló által dekódolt biteket megfelelő bitkeveréssel és visszakeveréssel. Addig cserélgetik az információt, ameddig konszenzusra nem jutnak

Turbó kódok dekódolása

BCJR-algoritmus



A Viterbi-dekódolás hibaanalízise

Diszkrét, memóriamentes csatorna.

Fontos a kódtávolság, de itt nincsenek kódszavak:

Legyen $a(d,i)$ azon i súlyú üzenetekből származtatott d súlyú kódszavak száma, melynek útvonala az első elágazásnál már eltér a tiszta nulla üzenetétől.

Vezessünk be ezekhez a kétindexes mennyiségekhez egy formális hatványsort

$$T(D,I) = \sum_{i=1}^{\infty}\sum_{d=1}^{\infty} a(d,i) I^i D^d$$