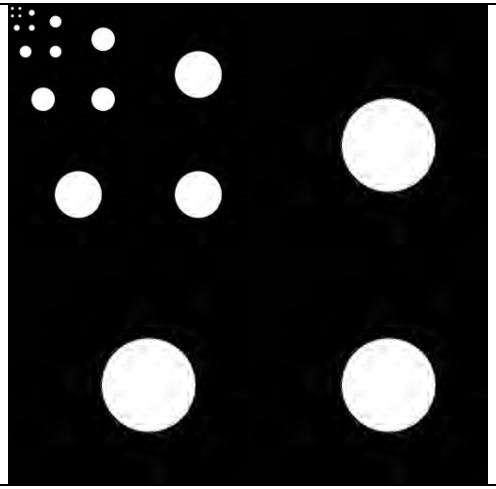
Ako sa konkrétne kódujú regióny v v JPEG 2000?

- Metóda MAXSHIFT (JP2K part 1)
  - Netreba kódovať hranicu regiónu, lebo:
  - Hodnoty v spektre, ktoré patria nejakému regiónu sa v zmysle bitových rovín jednoducho vysunú o nad ostatné spektrálne hodnoty
    - T.j. najnižšia absolútna hodnota s regiónu bude vačsia ako ľubovoľná hodnota mimo regiónu
  - Takto získajú hodnoty spektrálnych koeficientov "absolútnu prednosť" pri progresívnom prenose informácií
  - Ako ale presne vyzerá "maska", ktorá hovorí aké koeficienty vysunút? Viď príklad masky v (c)

a) Originálny obraz	b) spektrum	c) Ako vyzerá spektrálna maska	d) Rekonštruovaný obraz
			

- Metóda general scaling (JP2K part 2)
  - Polohy regiónov sa pamätajú (používajú sa štvorcové a elipsovité oblasti)
  - Za odmenu môžeme použiť ľubovoľný škálovací faktor a príslušné spektrálne hodnoty môžeme prenásobiť ľubovoľným faktorom dôležitosti

JPEG 2000

Kompresný postup

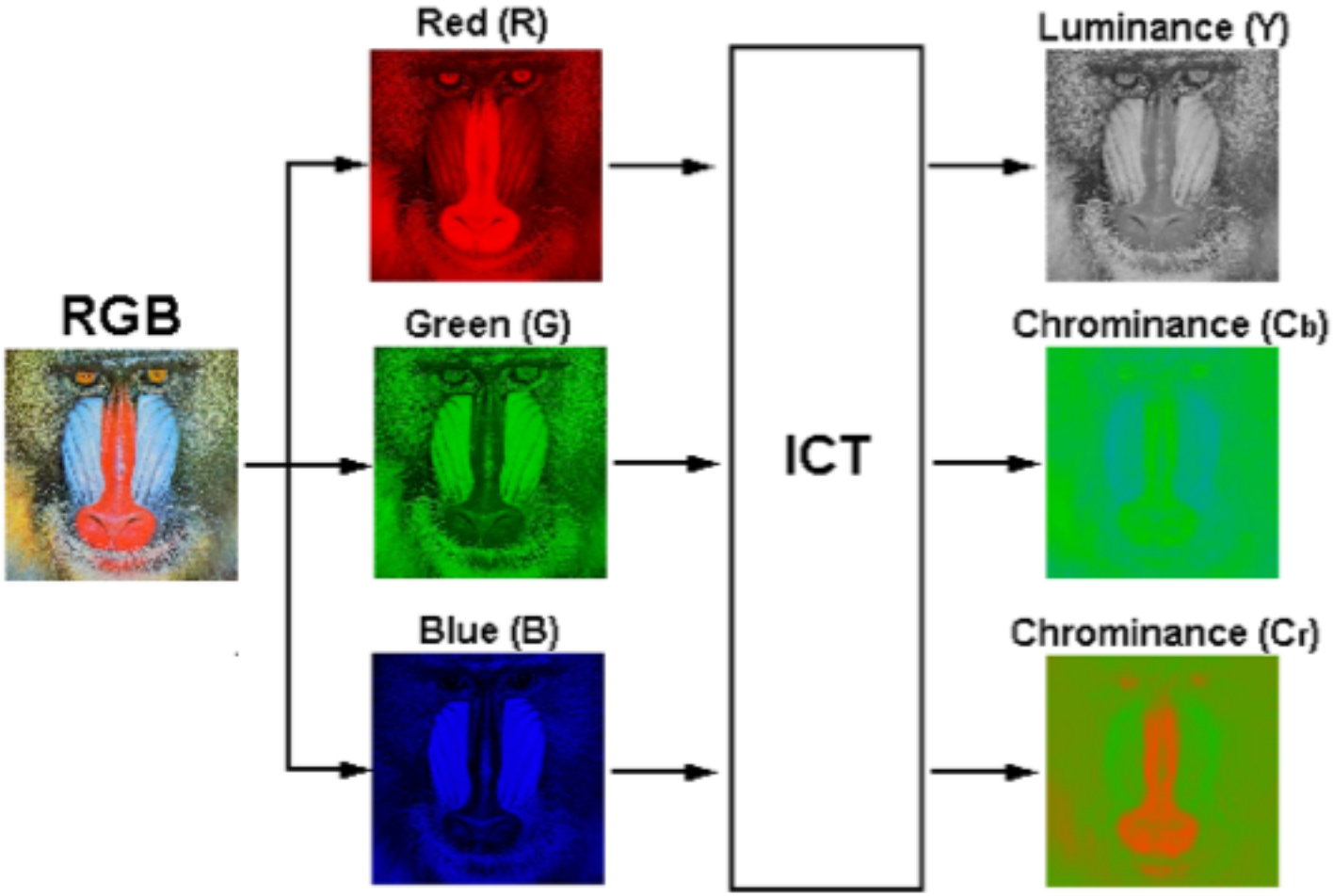
1. Transformácia farebného priestoru
    a. na Y, C_b, C_r
    b. v bezstratovom móde sa uvedený proces iba aproximuje
2. Rozdelenie na diely (tiles) (napr. 128x128,… ktoré sa transformujú úplne nezávisle)
3. Waveletová transformácia
4. Kvantizácia – redukcia počtu bitových rovín
    a. Skalárna kvantizácia s mŕtvou zónou (jednoduché orazanie nižších biových rovín)
        i. JPEG 2000 podporuje oddelené kvantizátory pre každé subpásmo
    b. Trelllisova kvantizácia
5. Po kvantizáci
    a. Každé subpásmo je rozdelené na bloky
    b. Bloky budú kódované nezávisle
6. Entropické kódovanie EBCOT (Embedded Block Coding with Optimised Truncation)
    a. TIER1 - kódovanie zdroja
    b. TIER2 – generovanie výstupného streamu

Truncation = kódovanie každého bloku sa optimalizuje v zmysle bitovej náročnosti verzus skreslenie (rate/distrotion)

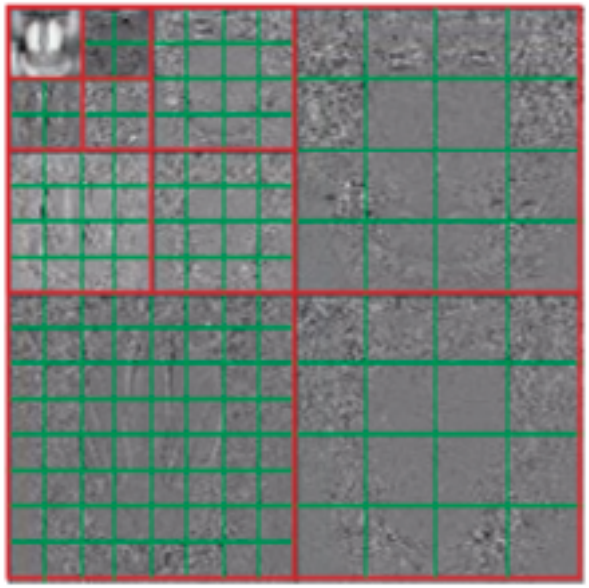
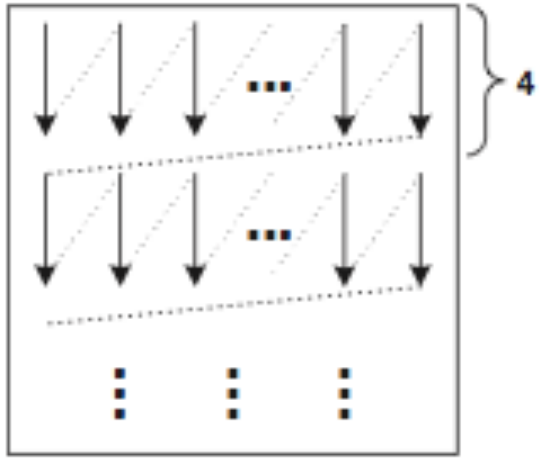
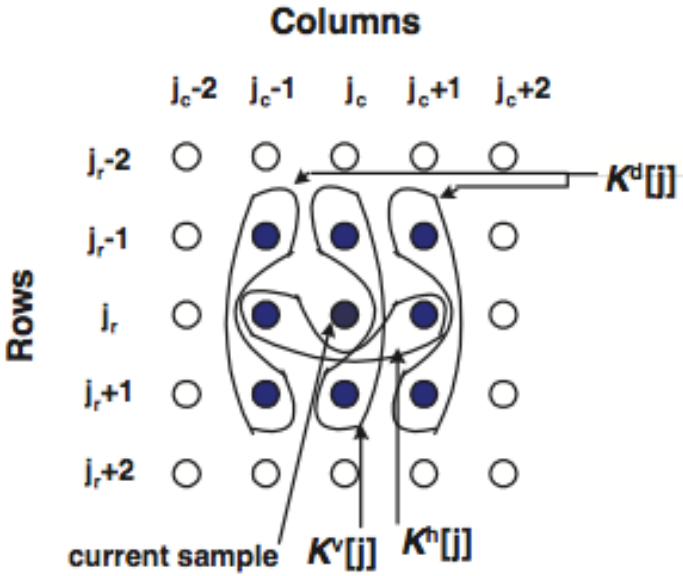
Transformácia farebného priestoru

ICT=Irreversible color transform

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.586 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

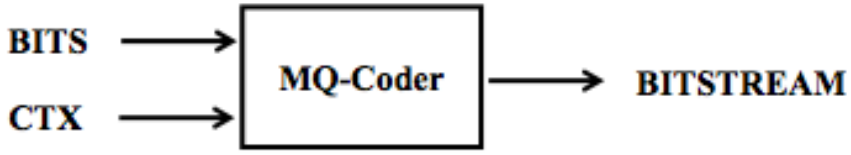


Kódové bloky a ich spracovanie

Rozdelenie spektra na kódové bloky	Poradie spracovania koeficientov v kódovom bloku	Definície okolia spracovávaného bodu
		

Ako funguje EBCOT

- Bitové roviny kódových blokoch sa kódujú pomocou 3 kódovacích prechodov (v uvedenom poradí):
  - Kódovania okolia významných (significance propagation)
  - Spresnenie magnitúdy významných (magnitude refinement)
  - Kódovanie nevýznačných (cleanup)
- Začína sa najvyššu bitovou rovinou a postupuje sa k nižším
  - Jednotlivé koeficieny sa pri prekročení hodnoty chategorizujú postupne ako signifikantné (významné) pričom ich model využíva pravdepodobnosť významnosti v ich susedstve
- Používajú sa 4 kódovacie postupy (primitívy) kvoli lepšiemu modelovaniu
  - RL – run length
  - ZC – zero coding
  - MR – magnitude refinement
  - SC – sign coding
- V rámci týchto postupov existujú ešte rôzne kontexty
- Informácia o kontexte a jednotlivé bity idú do MQ aritmetického kódera:



Ako funguje EBCOT (2)

- Kódovanie SC:

$\bar{\chi}^h$	$\bar{\chi}^v$	κ^{SC}	$\hat{\chi}$
1	1	SC4	1
1	0	SC3	1
1	-1	SC2	1
0	1	SC1	1
0	0	SC0	1
0	-1	SC1	-1
-1	1	SC2	-1
-1	0	SC3	-1
-1	-1	SC4	-1

Pri pásmach LL, LH, HH platí tabuľka, pri HL je výstup invertovaný.

- Vstup χ je 1, ak susedia su kladní, alebo jeden je kladný a druhý ešte nie je význačný
- Vstup χ je 0 aj ani jeden sused nie je význačný, alebo majú opačné znamienka
- Vstup χ je -1 aksuedia sú záporní, alebo jeden je záporný a druhý ešte nie je význačný

Výstup je predikcia hodnoty. Ak predikcia sedí kóduje sa 0, ake nesedí, tak 1

- Kódovanie ZC:

LL and LH subbands			HL subband			HH subband		κ^{ZC}
κ^h	κ^v	κ^d	κ^h	κ^v	κ^d	κ^d	$\kappa^h + \kappa^v$	
0	0	0	0	0	0	0	0	ZC0
0	0	1	0	0	1	0	1	ZC1
0	0	≥ 2	0	0	≥ 2	0	≥ 2	ZC2
0	1	x	1	0	x	1	0	ZC3
0	2	x	2	0	x	1	1	ZC4
1	0	0	0	1	0	1	≥ 2	ZC5
1	0	≥ 1	0	1	≥ 1	2	0	ZC6
1	≥ 1	x	≥ 1	1	x	2	≥ 1	ZC7
2	x	x	x	2	x	≥ 3	x	ZC8

Vyhodnocuje sa počet signifikantných susedov susedov κ (všetkých 8, 2x κ^h, 2x κ^v, 4x κ^d), v závislosti od toho sa volí kontext a dáva sa 1bit na výstup 0 ak bit je 0, 1 ak bit je 1

- Kódovanie MR:

$\bar{\sigma}$	$\kappa^h + \kappa^v$	κ^{MR}
0	0	MR0
0	$\neq 0$	MR1
1	x	MR2

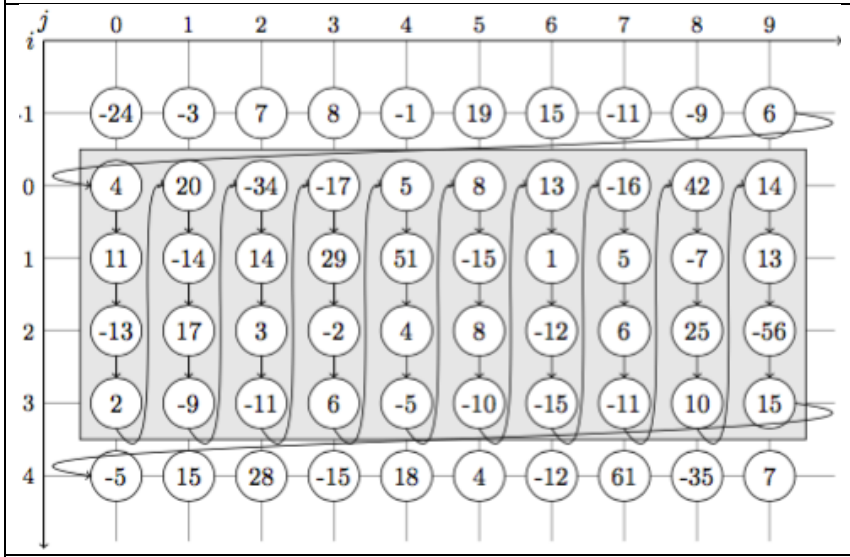
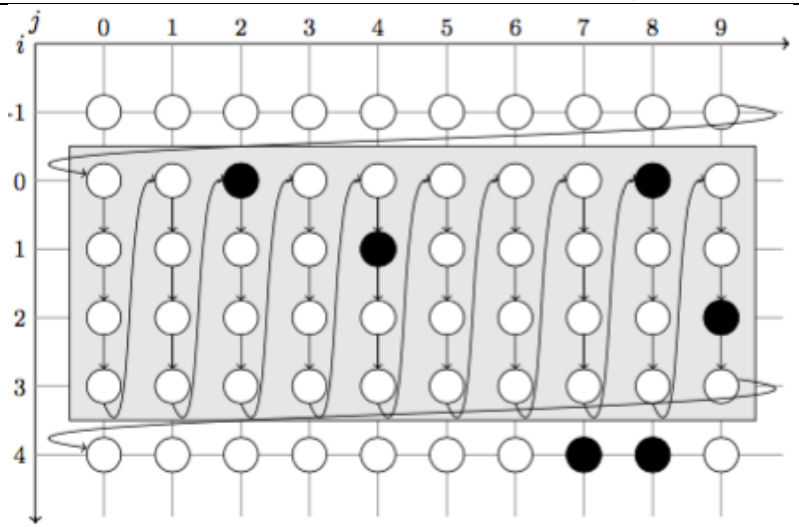
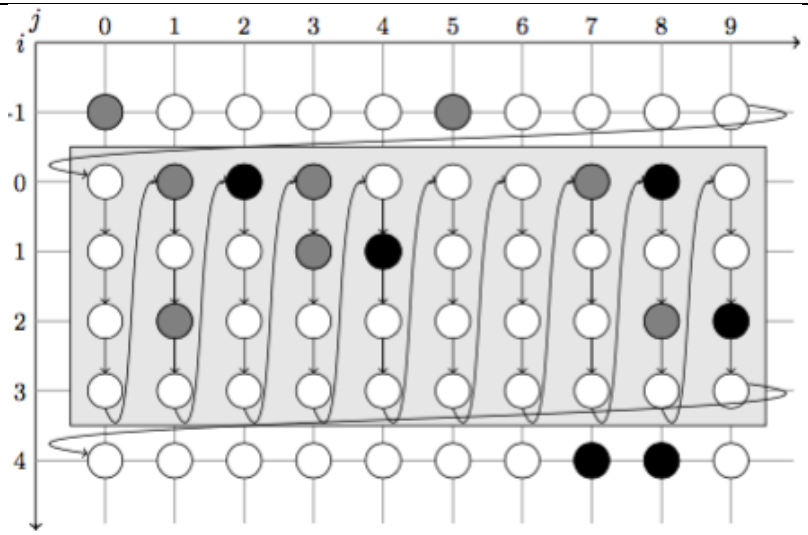
Vstup σje 0 ak v kroku MR sme pre daný koeficient po prvý krát, ináč je vstup 1, kontext závisí aj od signifikantnosti susedov κ

Algoritmus kódovania

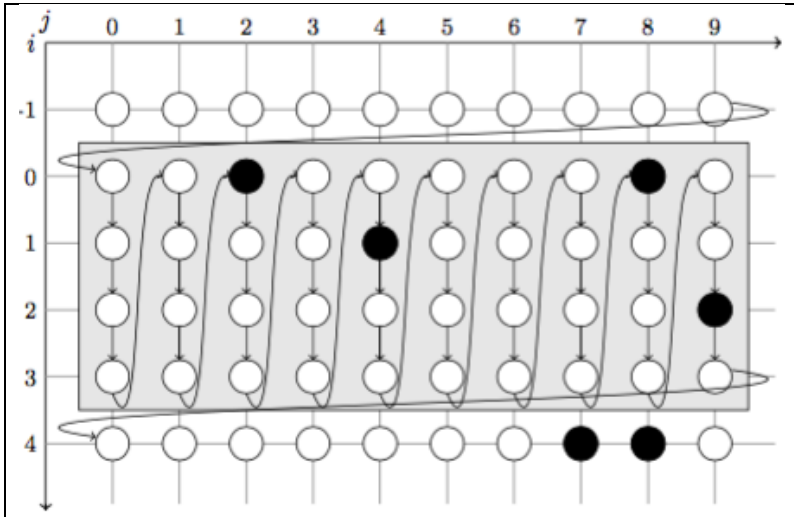
- Každý bit sa prechádza iba raz, a to podľa toho, do akej množiny patrí jeho koeficient
  - Na začiatku sú všetky koeficieny nevýznačné (v cleanup množine)

- **Kódovania okolia význačných (significance propagation)**
  - Pre každý koeficient z okolia význačných (označené ako význačné pri predchádzajúcich cleanup prechodoch) kóduj bity pomocou ZC kontextu
- **Spresnenie magnitúdy význačných (magnitude refinement)**
  - Používa sa výhradne MR kódovanie (MR0, MR1, MR2)
- **Kódovanie nevýznačných (cleanup)**
  - stĺpec kóduje pomocou RL ak v jeho okolí nie sú žiadne signifikantné koeficienty. Ak sú tak kóduj stĺpec pomocou UNI kontextu
  - Kódovanie RL
    - ak stĺpec nemá ani jeden bit rovný 1,
      - kóduj 0 v kontexte RL
    - ak má aspoň jeden bit rovný 1,
      - kóduj 1 v kontexte RL,
      - prepni sa do UNI kontextu oznám polohu bitu 1 (hodnoty 00-11)
      - oznám znamienko pomocou kontextu SC
      - pokračuje kódovaním zvyšných bity v stĺpci pomocou kódovania ZC
        - ak je niektorý bit jednodkový, kóduj aj znamienko pomocou kontextu SC
      - odpovedanúce koeficienty sú odteraz vedené ako význačné

Príklad:

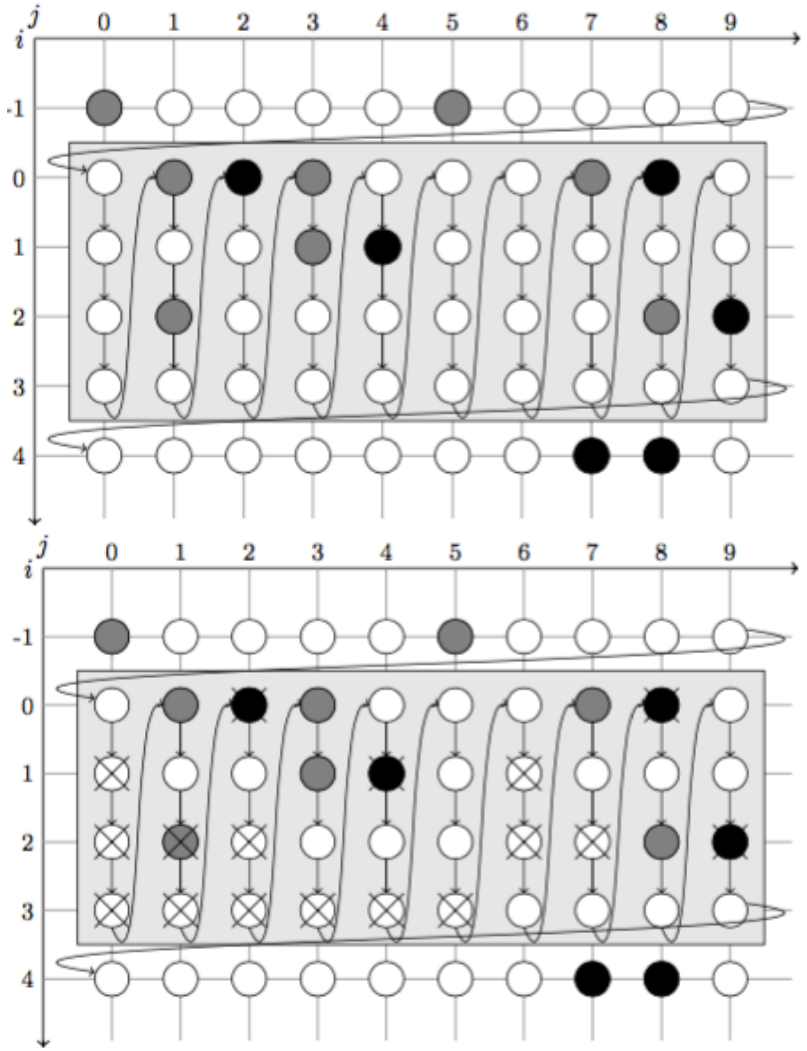
Numerické hodnoty kódového bloku v LH pásme po kvantizácii	Označené sú koeficienty s jednotkovým bitom v bitovej rovine n=5 ($2^5 = 32$)	
		Čiernou sú označené koeficienty MSB v bitovej rovine n=5 ($2^5 = 32$)
		Sivou sú označené koeficienty s MSB v bitovej rovine n=4 ($2^4 = 16$)

1) kódovanie začneme cleanup prechodom pri bitovej rovine 5



$j = 0$	0(RL)
$j = 1$	0(RL)
$j = 2$	1(RL) 00(UNI) 1(SC0) 0(ZC3) 0(ZC0) 0(ZC0)
$j = 3$	0(ZC5) 0(ZC1) 0(ZC0) 0(ZC0)
$j = 4$	1(RL) 01(UNI) 0(SC0) 0(ZC3) 0(ZC0)
$j = 5$	0(ZC1) 0(ZC5) 0(ZC1) 0(ZC0)
$j = 6$	0(RL)
$j = 7$	0(RL)
$j = 8$	1(RL) 00(UNI) 0(SC0) 0(ZC3) 0(ZC0) 0(ZC0)
$j = 9$	0(ZC5) 0(ZC1) 1(ZC0) 1(SC0) 0(ZC3)

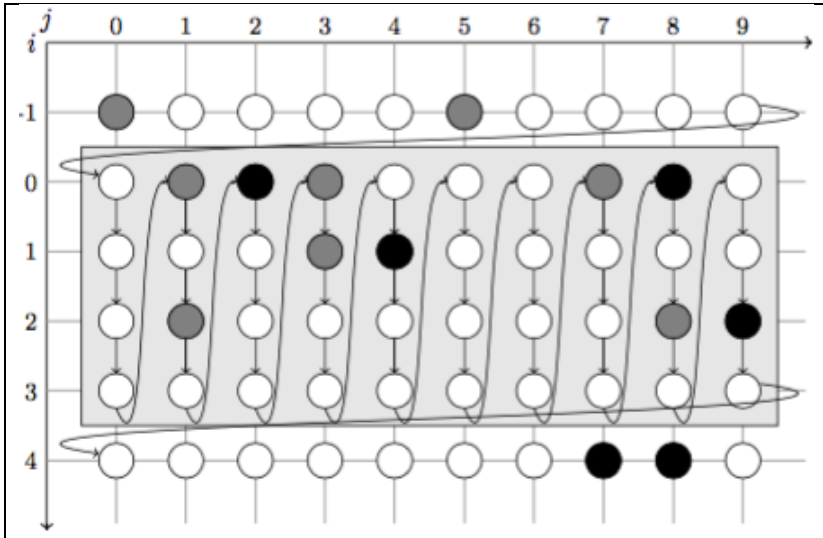
2) pokračujeme kódovaním significance propagation pri bitovej rovine 4



kódujeme iba koeficienty v okolí čiernych bodov →
Kódovanie okolia význačných (preškrtnuté nekódujeme)

$j = 0$	0(ZC3)
$j = 1$	1(ZC6) 1(SC3) 0(ZC3)
$j = 2$	0(ZC3)
$j = 3$	1(ZC6) 0(SC3) 1(ZC7) 0(SC2) 0(ZC3)
$j = 4$	0(ZC7) 0(ZC3)
$j = 5$	0(ZC3) 0(ZC5) 0(ZC1)
$j = 6$	0(ZC1) 0(ZC1)
$j = 7$	1(ZC5) 1(SC3) 0(ZC3) 0(ZC3)
$j = 8$	0(ZC3) 1(ZC5) 1(SC3) 0(ZC4)
$j = 9$	0(ZC5) 0(ZC3) 0(ZC3)

3) Pokračujem prechodom “Spresnenie magnitúdy význačných” (magnitude refinement)

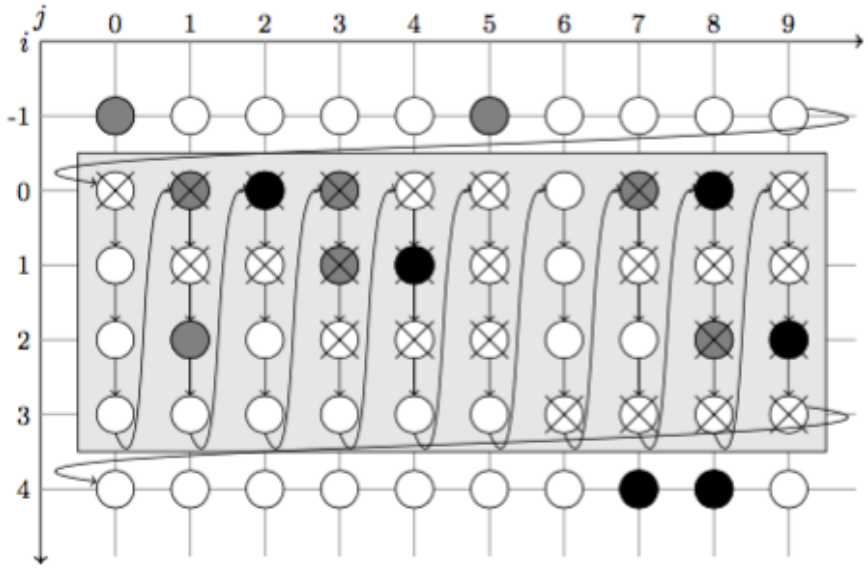


kódujeme spresňujúce bity z hodnôt, ktoré sa stali signifikantnými pri cleanupe pri n=5
(ale pozor, už aj šedé body z roviny n=4 sú signifikantné, takže preto všade vychádza MR1)

Kódujeme hodnoty pre 4 koeficienty:
(0,2) → 0(MR1)
(1,4) → 1(MR1)
(0,8) → 0(MR1)
(2,9) → 1(MR1)

4) Kódovanie bitovej roviny pri n=4 zakončíme prechodom Kódovanie nevýznačných (cleanup)

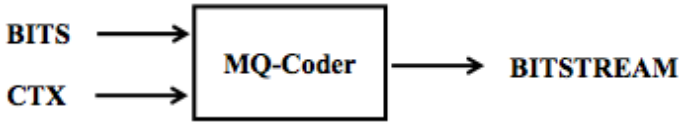
bity z roviny n=4, ktoré v predchádzajúch prechodoch spracovali teraz vynecháme (na obrázku sú preškrtnuté) a v cleanupe prechádzame zvyšné:



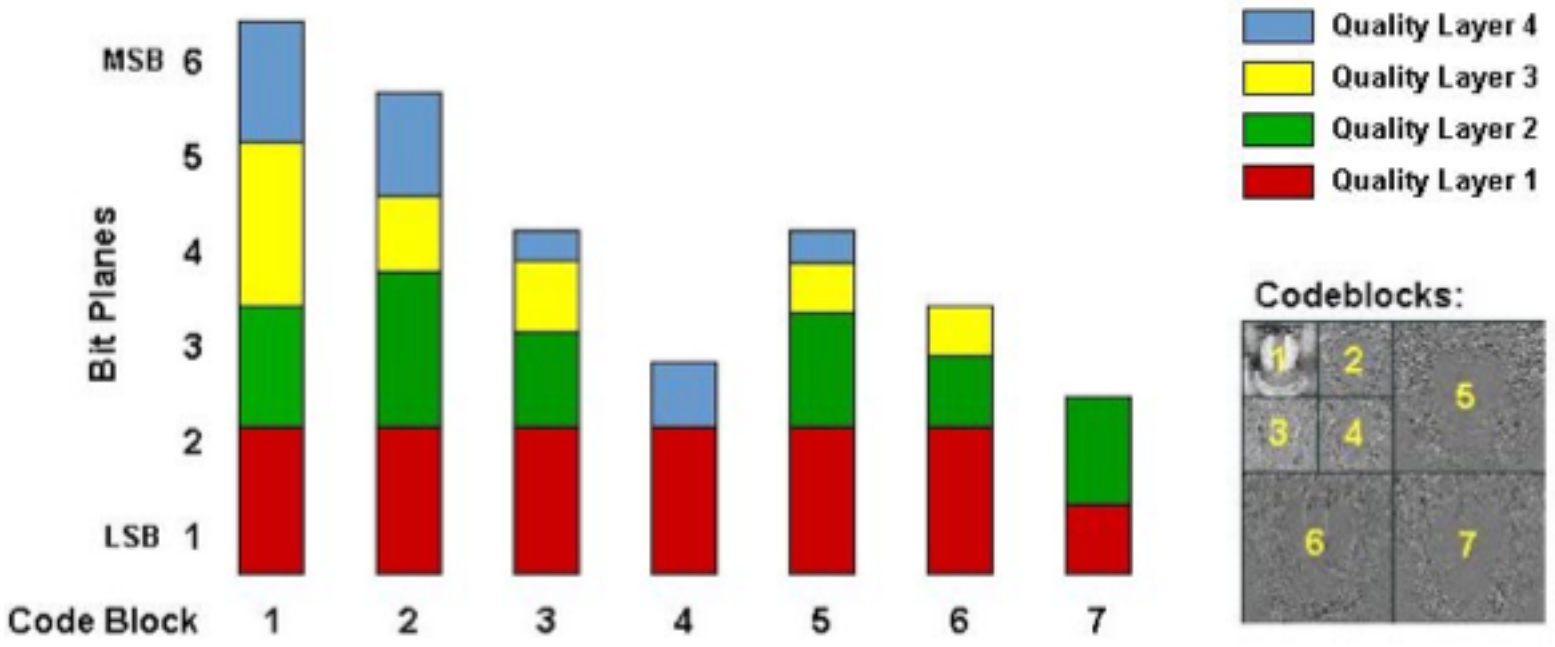
$j = 0$	0(ZC1) 0(ZC0) 0(ZC0)
$j = 1$	1(ZC0) 0(SC0) 0(ZC3)
$j = 2$	0(ZC6) 0(ZC1)
$j = 3$	0(ZC0)
$j = 4$	0(ZC0)
$j = 5$	0(ZC0)
$j = 6$	0(ZC6) 0(ZC1) 0(ZC0)
$j = 7$	0(ZC5)

5) A takto by sme dalej išli postupne po bitových rovinách …

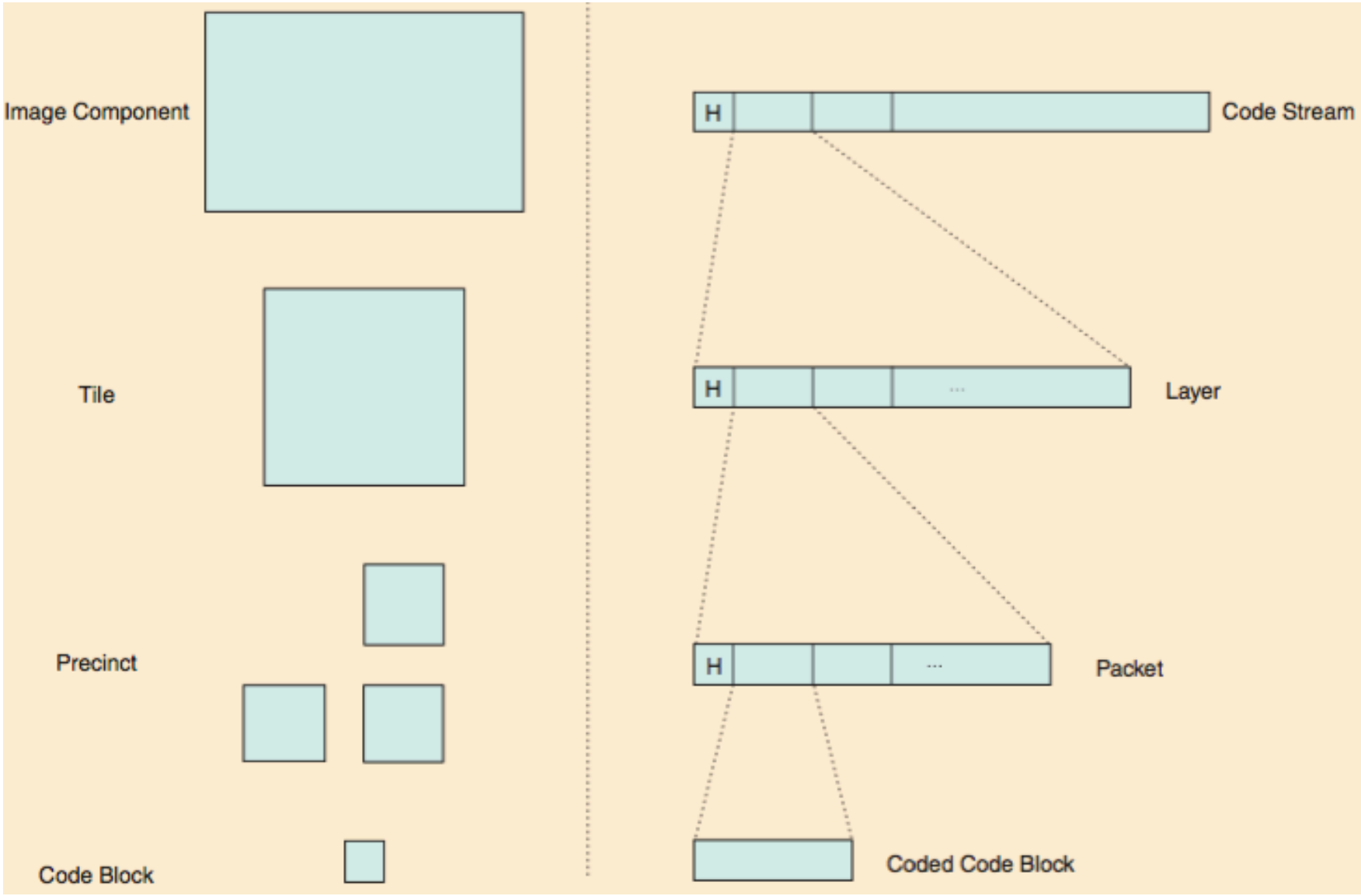
6) … a dávali to všetko na vstup aritmetického MQ kódera



Vzťah medzi kódovými blokmi, bitovými rovinami a vrstvami v JPEG 2000



Priestorové delenie obrázku (vľavo) verzus organizovanie výstupného bitového toku (vpravo)



Aritmetické kódovanie - základ

- entropické kódovanie – používa sa pri stratových aj bezstratových konpresných postupoch
- úloha – zakódovať správu
  - Hufffmanov kód – pristupuje k správe ako k množine symbolov a kóduje každý symbol zvlášť
  - Aritmetický kód – zakóduje celú správu ako jedno jediné číslo
    - Umožnuje sa dostať až k hranici – $\log_2 P$ na symbol

Analýza efektivity

- Predpokladajme 3 symboly s rovnakými pravdepodobnosťami
- Vyšlime správu, kde je po 100 výskytov každého symbolu (t.j. dokopy 300 symbolov)
  - Aký je teoretický limit? $pocet_{bitov} = 300(-\log_2 0.\overline{3}) = 300(1.585)$=475.5 [bit]
  - Ako si s tým poradí huffmanov kód?
    - Symbol 1 = 0
    - Symbol 2 = 10
    - Symbol 3 = 11
    - Správa=100+200+200=500 bitov
  - Ako s s tým poradí aritketický kód?
    - Kódujeme číslo v trojkovej sústave, napr.: $0.000 \ldots 000111 \ldots 111222 \ldots 222_3$
    - T.j. napr 100 symbolov, potom 100 symbolov2, potom 100 symbolov3
    - A toto číslo chceme previesť do dvojkovej sústavy
      - =$0.010010 \ldots 01_2$
      - koľko bitov sme spotrebovali, toľko potrebujeme na výsledok
      - na výsledok spotrebujeme približne $300 * \frac{\ln 3}{\ln 2}$=300(1.585)=475.5 [bit]