

# Multimedejske arhitekture i sustavi (dio 1)

Prof.dr.sc. Mario Kovač  
Prof.dr.sc. Hrvoje Mlinarić  
Prof.dr.sc.Josip Knezović

## Multimedija

- Detaljnija definicija mogla bi glasiti:

Integracija dva ili više naprednih tipova informacija (audio, video, grafika, ...) sa ciljem stvaranja, grupiranja, prijenosa, pohrane, obrade i prezentacije sadržaja.

## Što je to MULTIMEDIJA?

- Neki autori koriste izraz 'višemedija'
- Prema rječniku:  
*multimedijalan – (višemedijalan), koji se istodobno služi sa nekoliko tehnika reprodukcije i prenošenja komunikacija*

## Multimedijijski računalni sustav

- Integrirani računalni, komunikacijski i informacijski sustav koji omogućuje obradu, upravljanje, zaštitu, slanje i korištenje te prezentaciju sinkroniziranih multimedijijskih informacija

# Multimedijski sustavi

## ■ Interaktivnost

- zahtjevi za dvosmjerni tijek podataka koji može omogućiti korisničko upravljanje informacijama

## ■ Sinkronizacija

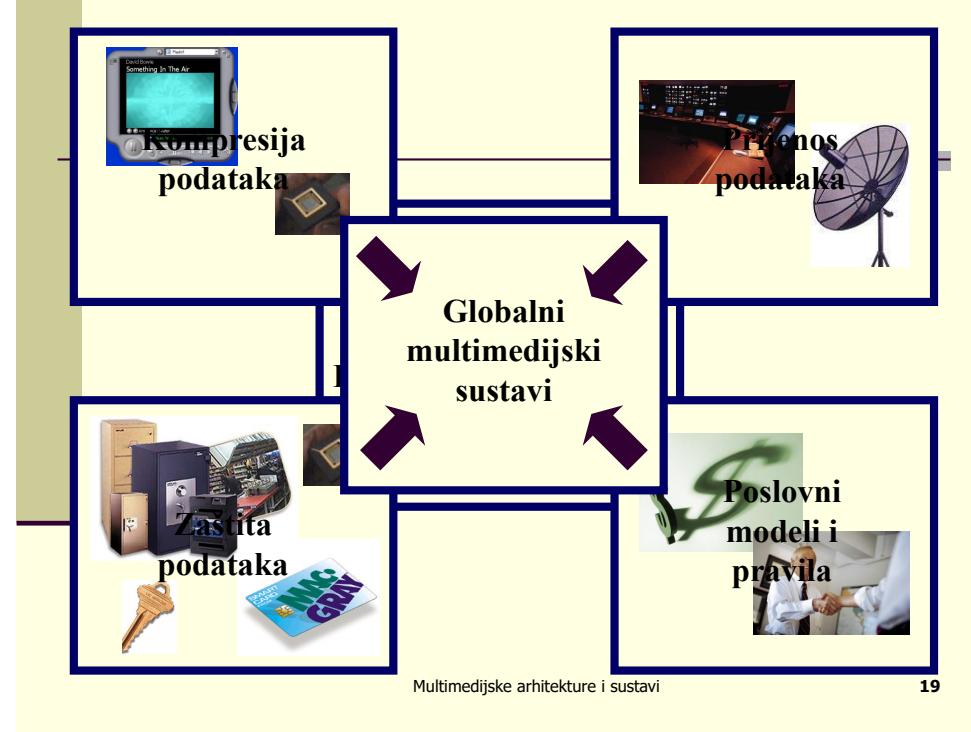
- održavanje vremenskih odnosa između pojedinih dijelova poruke kao i između različitih ali međusobno ovisnih poruka

## ■ Normizacija

- omogućuje interoperabilnost različitih tipova informacija, različitih sustava, mrežnih protokola, proizvoda,...

Multimedijiske arhitekture i sustavi

18



# Multimedijiske arhitekture i sustavi

## ■ Pozadina i potrebe:

- U današnje vrijeme značajni dio razvoja računalne industrije vezan je za multimediju:
  - Komunikacijska industrija (obične i mobilne mreže): nove potrebe za ulaganjima u infrastrukturu
  - Računalna industrija (serveri, stolna i mobilna računala): procesorska snaga
  - Procesori (desktop, mobilni): nove arhitekture
  - Aplikacije: nove i nove mogućnosti

Multimedijiske arhitekture i sustavi

20

MAS

Neke analize i predviđanja

Multimedijiske arhitekture i sustavi

23

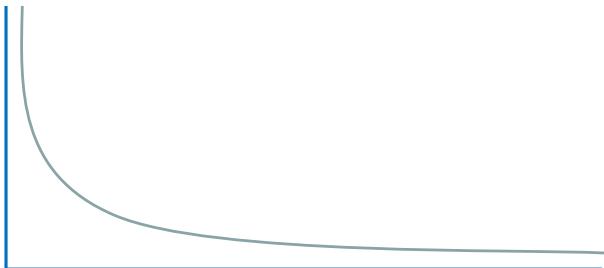
## **Global challenges**

- IP video traffic will be 82 percent of all global Internet traffic

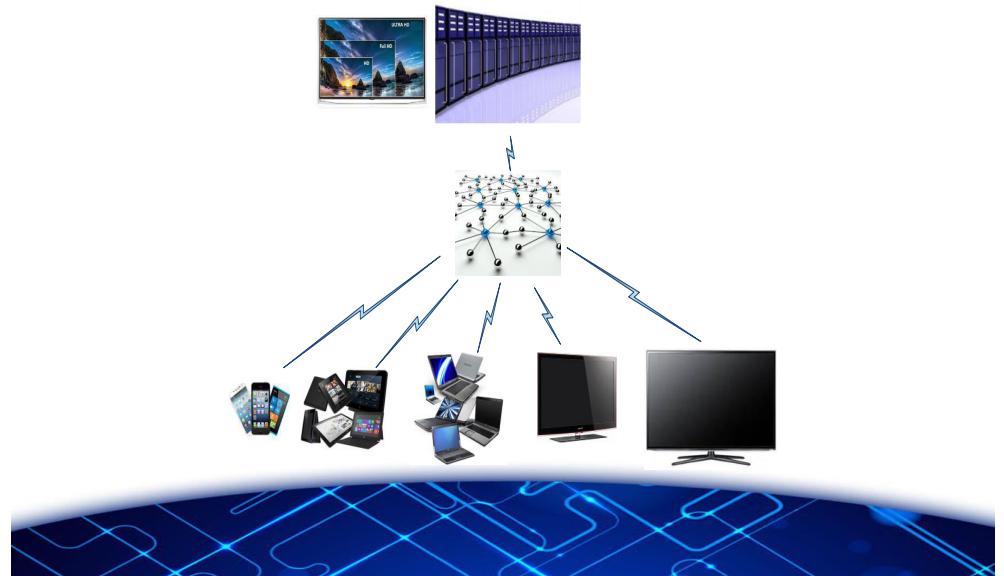


## **The "Long tail"**

- "Forget squeezing millions from a few megahits at the top of the charts. The future of entertainment is in the millions of niche markets at the shallow end of the bitstream." – Chris Anderson, *Wired magazine*



## **Plethora of novel devices**



## **The QoS issue, the Power issue,...**

- MM content processing is almost inherently tied to QoS requirements
- Power constraints are among most important ones in todays computing systems



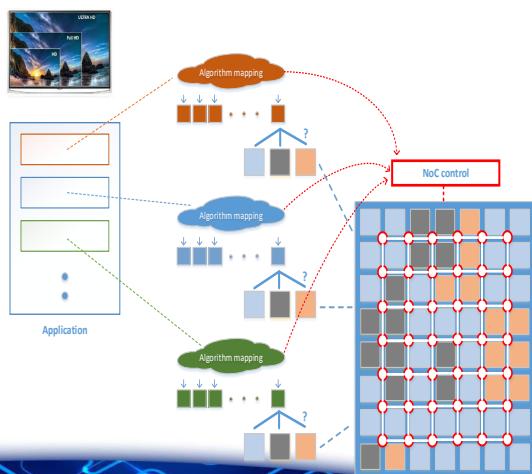
## *The medical imaging case..*

- Another important application case is medical imaging
- Challenges are related to huge medical image sizes that need to be processed in short periods of time to allow novel and advanced medical procedures or medical consultations
- Delivery of medical imaging to various locations and device types shows same problems as previously described



## *Research directions*

- Mapping of algorithms to SW, GPU, SIMD, FPGA
- Low level optimisations
- Various processing constraints:
  - Real time
  - Processing power
  - Throughput
  - ...



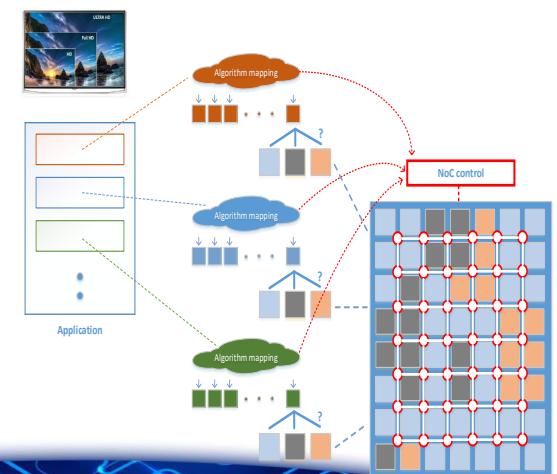
## *Efficient MM processing on HPC*

- Theory shows that MM algorithms can be designed to efficiently exploit various computing architectures
- Based on our previous research and SoA we aim at theoretical decomposition of MM algorithm space and mapping of decomposed units to optimal underlying architecture



## *New challenges*

- Number of cores and volume of data require novel system architecture designs
- NoC communication algorithms that can adapt to MM
- Allocation of cores/core types on manycore system



# Multimedejske arhitekture i sustavi (dio 2)

Prof.dr.sc. Mario Kovač  
Prof.dr.sc. Hrvoje Mlinarić  
Prof.dr.sc.Josip Knezović

## Informacija

- $i(A)$  nam formalno definira ono što je logično:
  - ako se nešto događa često (sa velikom vjerojatnošću) tada nam takav događaj ne donosi puno informacija (npr. suma brojeva na dvije kockice je 7)
  - rijetki događaji donose puno informacija (npr. suma brojeva na dvije kockice je 2)
- ako izaberemo  $\log_2$  tada je količina informacija izražena u bitovima

## Informacija

- Osnovna ideja: pronaći i ukloniti redundanciju u podacima
- Svojstvena (vlastita informacija)
  - Kvantitativna mjera za količinu informacije

$$i(A) = \log_b (1/P(A)) = -\log_b P(A)$$

gdje je:  $P(A)$  vjerojatnost pojavljivanja događaja A

- primjer:    za  $P(A)=1$      $i(A)=0$   
               za  $P(A)=0$      $i(A)=\infty$

## Entropija

- $S = \{a_1, a_2, a_3, \dots, a_N\}$  – skup svih mogućih simbola sa vjerojatnostima  $P(a_1), P(a_2), \dots, P(a_N)$
- $H(S) = \sum P(a_n) i(a_n) = - \sum P(a_n) \log_b P(a_n)$
- $H$  predstavlja *prosječni* vlastiti sadržaj informacije za izvor  $S$  i naziva se Entropija.
- Gornji izraz predstavlja pojednostavljenje stvarnog izraza za entropiju i naziva se entropija nultog reda i biti će dovoljno dobra aproksimacija za naša predavanja

## Entropija

- Ako izaberemo  $\log_2$  tada vrijednost entropije definira najmanji prosječan broj bitova potreban za kodiranje pojedinog simbola iz ulaznog niza
- Stvarnu entropiju slučajnog izvora za općeniti slučaj nikada ne možemo dozнати

## MAS – Entropijsko kodiranje

- Entropijsko kodiranje:
  - Huffmanovo kodiranje
  - RLE (engl. run length encoding)
  - Aritmetičko kodiranje
  - Shannon-Fano, Golomb-Rice, ...



## MAS – Osnove kompresije

- Kompresija podataka:



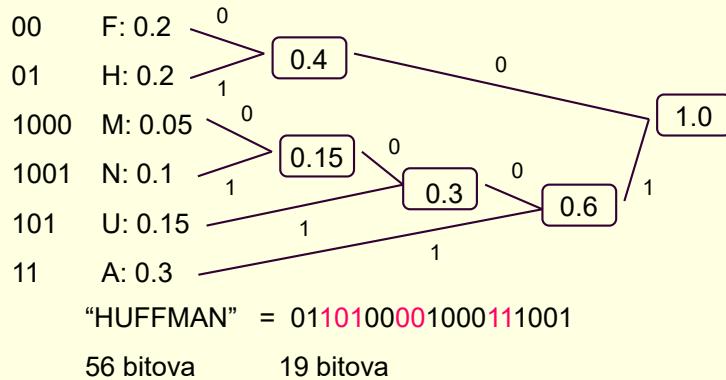
- Entropijski koder (statističko kodiranje): uklanja statističku redundanciju iz toka podataka

## MAS – Osnove kompresije: Huffmanovo kodiranje

- Huffman, 1952:
  - Huffman-ovo kodiranje
  - Optimalni cjelobrojni kodovi (najbliži entropiji modela)
  - Prefiks kodovi (nijedan kod nije početak nekog drugog koda)
  - Često korištena metoda

## MAS – Huffmanovo kodiranje

- Jednostavna prezentacija pomoću binarnog stabla



Multimedijiške arhitekture i sustav

9

## MAS – Osnove kompresije: RLE kodiranje

- Zasniva se na uzastopnom ponavljanju jednog simbola
  - Jednostavna izvedba
  - Primjer:
    - linije očitane sa dokumenta u fax uređaju  
1 linija (75 dpi, 8")=600 bita=75 B

00000000000000000000011111111111111111111000000000000.....000111111111.....11111000000

17,24,3,211,22,188,77,54,4 = 9 B

Multimedijiške arhitekture i sustav

11

## MAS – Huffmanovo kodiranje

- #### ■ Generiranje Huffmanovih kodova:

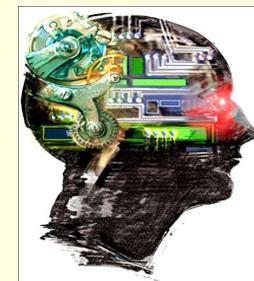
- Statički – dva prolaza po podatcima, prvi skupljanje vjerojatnosti, drugi kodiranje, moraju se prenijeti i huffmanovi kodovi
  - Adaptivno – jedan prolaz, model (Huffmanovo stablo) se adaptivno izgrađuje tijekom prijelaza po podatcima, računski zahtjevnije jer i kompresor i dekompresor adaptivno izgrađuju model

Multimedijiške arhitekture i sustavi

10

## MAS – Osnove kompresije

- Teorija kodiranja (teorija informacija) je relativno opsežno obrađeno područje
  - Umnogo zanimljiviji dio u kompresiji predstavlja iznalaženje adekvatnog MODELA!



Multimedijiške arhitekture i sustavi

12

# Modeli

- Da li možemo na neki način još više ‘*smanjiti*’ vrijednost entropije? (entropija je za izabrani izvor uvek ista ali mi pokušavamo smanjiti našu *procjenu* entropije i time smanjiti količinu podataka)
- Ako na neki način možemo predvidjeti ponašanje izvora (tada govorimo o **modelu** tog izvora) tada možemo bolje predvidjeti entropiju (i u idealnom slučaju smanjiti procjenu)

## Osnovni modeli

- Vjerojatnosni model
  - često korišten model, osnova za neke vrlo efikasne modele
  - pretpostavka je da su svi simboli generirani iz izvora potpuno neovisni
- Dvije osnovne varijante
  - ‘slijepi’ model: dodatno prepostavljamo da je za sve simbole vjerojatnost pojavljivanja ista
  - statistički model: na neki način izračunati učestalost pojavljivanja simbola te na temelju toga definirati vjerojatnosti
- Problem: mogućnost POVEĆANJA ENTROPIJE!!

## Osnovni modeli

- Fizikalni model
  - ako poznajemo fizikalna svojstva izvora
  - obično je ovo prekompleksan model te se u takvim slučajevima pokušava koristiti neki alternativni model

## Osnovni modeli

- Markovljev model (A.A. Markov 1856-1922)
  - ovim modelom opisuje se izvor ‘s pamćenjem’ tj izvor kod kojeg vjerojatnost pojavljivanja određenog simbola ovisi o svim simbolima koji su se pojavili prethodno u nizu
  - tako za Markovljev model prvog reda vrijedi
$$P(A_i|A_{i-1}, A_{i-2}, A_{i-3}, \dots) = P(A_i|A_{i-1})$$
  - ovaj model je izuzetno efikasan za određene tipove podataka (npr model “predviđanja” spada u ovaj model)
  - ponovo postoji opasnost od povećanja entropije!!

## Osnovni modeli

- vrlo često samo jedan model ne može iskoristiti sve činjenice koje znamo o izvoru te se stoga koriste višestruki modeli kojima se značajno poboljšava opis izvora
- Tijekom ovih predavanja vidjeti ćemo konkretnе primjere za većinu prije navedenih modela

## MAS

Algoritam za kompresiju slike

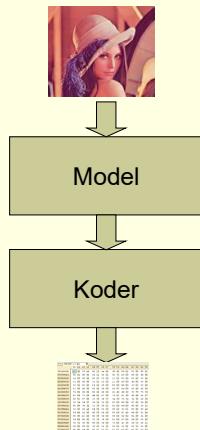
## MAS – neki modeli

- Primjeri modela:
  - Slikovni podatci – prostorna korelacija u 2D prostoru, HVS
  - Video podatci – prostorna i vremenska korelacija podataka, HVS
  - Financijski podatci – korelacija u 1D, predviđanje

## Model za slikovne podatke

- Podaci bi se mogli obraditi (kompresirati) i bez upotrebe posebno odabranog modela
- Npr. ZIP, RAR, ...
- Iz iskustva znamo da time nećemo ostvariti značajne omjere kompresije
- Upotrebljava se model koji omogućuje znatno veće omjere kompresije

# Arhitektura sustava za kompresiju slika/videa

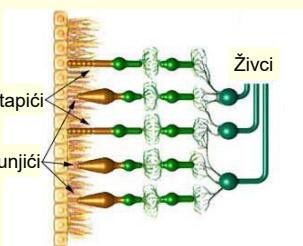


Multimedijiske arhitekture i sustavi

21

## Model zasnovan na našem vizualnom sustavu

- fotoreceptori mrežnice
  - štapići (*rods*) i čunjići (*cones*)
  - sadrže kemijske tvari osjetljive na svjetlost
- čunjići i štapići nisu jednako osjetljivi na cijeli spektar vidljive svjetlosti
  - različite vrste monokromatskog svjetla podražuju ili čunjiće ili štapiće
- Ljudski vizualni sustav – HVS



Multimedijiske arhitekture i sustavi

23

## Ljudski vizualni sustav

HVS model

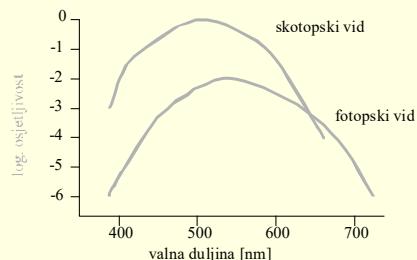
## HVS – spektralna osjetljivost

- štapići
  - osjetljivi na svjetlo i pri niskim razinama luminancije ispod  $1\text{cd}/\text{m}^2$  ("noćno" gledanje ili skotopski vid)
  - mogu razlikovati samo promjene u luminanciji, a nisu osjetljivi na boju
- čunjići
  - doprinose osjetu i razlikovanju boja, a postaju aktivni pri višim razinama luminancije
  - kod razina luminancije između  $1\text{cd}/\text{m}^2$  i  $100\text{cd}/\text{m}^2$  aktivni su i štapići i čunjići (fotopski vid)
  - pri razinama luminancije većim od  $100\text{ cd}/\text{m}^2$  štapići postaju zasićeni i aktivni su samo čunjići

Multimedijiske arhitekture i sustavi

24

## HVS – spektralna osjetljivost



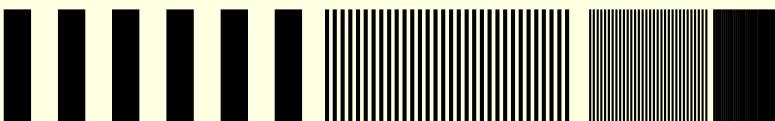
Ljudsko oko ima cca 10-20 puta više štapića nego čunjića

ZAKLJUČAK #1:

**Znatno smo osjetljiviji na promjene u luminanciji nego u boji**

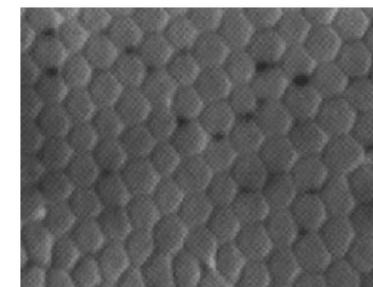
## HVS – prostorna osjetljivost

- Zbog konačne fizičke gustoće receptora, oko može razaznati detalje ako su zrake svjetlosti pod kutem upada većim od cca 1 minute
- Prostorna frekvencija

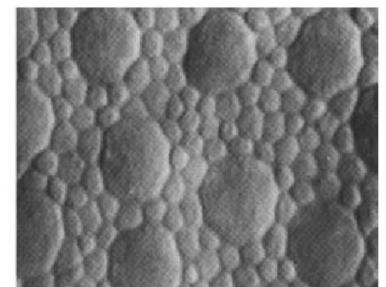


## HVS – prostorna osjetljivost

- Gustoća receptora

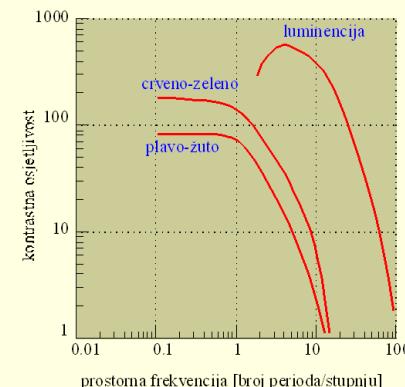


(a) Fovea



(b) Periphery

## HVS – prostorna osjetljivost



ZAKLJUČAK #2:

prostorna frekvencija [broj perioda/stupnju]

Oko je neosjetljivo na visoke prostorne frekvencije , a posebno to vrijedi za komponente boje.

## HVS Model za slike

- Kako iskoristiti prethodna saznanja o ljudskom vizualnom sustavu da bi omogućili veće omjere kompresije
- Cilj bi nam bio:
  - Zadržati što više informacija o luminantnoj komponenti
  - Izbaciti prostorne frekvencije koje oko ne vidi

Multimedejske arhitekture i sustavi

29

## Promjena prostora boja

RGB ↔ YUV

## Korištenje karakteristika HVS

- Kako? Moramo transformirati podatke
  - 1. Prebaciti u prostor boja što sličnije oku
  - 2. Prebaciti u frekvencijsku domenu
- 1. Promjena prostora boja
- 2. Transformacija
- Fizički model !!!

Multimedejske arhitekture i sustavi

30

## Promjena prostora boja

- Ulazni podaci: uglavnom RGB
- Prikladan prostor boja: YUV
- RGB-YUV konverzija vrlo se lako može obaviti jednostavnom matričnom operacijom:
$$Y = (0,257 * R) + (0,504 * G) + (0,098 * B) + 16$$
$$U = -(0,148 * R) - (0,291 * G) + (0,439 * B) + 128$$
$$V = (0,439 * R) - (0,368 * G) - (0,071 * B) + 128$$
- Kompleksnost (za kasnije analize): za izračun svakog piksela treba 9 množenja i 9 zbrajanja (+dohvat, spremanje)

Multimedejske arhitekture i sustavi

32

## Promjena prostora boja

Postoji i nešto drugačija formula za konverziju koja se koristi u JPEG-u a definirana je u JFIF dokumentu:

$$Y = 0.299 R + 0.587 G + 0.114 B$$

$$Cb = -0.1687 R - 0.3313 G + 0.5 B + 128$$

$$Cr = 0.5 R - 0.4187 G - 0.0813 B + 128$$

## Poduzorkovanje

- “subsampling”
- Jedna od najjednostavnijih metoda kako nakon konverzije u YUV prostor boja možemo smanjiti količinu podataka je poduzorkovanje:
  - Oko nije toliko osjetljivo na prostornu frekvenciju komponenata boje te se one ne prenose za svaki piksel
  - 4:4:4, 4:2:2, ...

## Što smo postigli konverzijom?

- Dobili smo dva skupa komponenata
  - Y: luminancija, oko je znatno osjetljivije na ovu komponentu
  - U,V: boja, oko manje osjetljivo
- Ideja je da prethodna dva skupa obrađujemo RAZLIČITO
- No ovime još uvijek nismo postigli apsolutno nikakvu promjenu u količini podataka

## Transformacija

DCT

# Transformacija

- Prebacivanjem podataka u frekvencijsku domenu želimo dobiti informacije o frekvencijskim karakteristikama svake komponente
- Koju transformaciju izabrati?
  - Želja nam je da se nakon transformacije većina informacija zadrži u što manjem broju što nižih frekvencijskih elemenata
  - Prema teoriji: Karhunen-Loëve (KLT) je idealna (ali je potpuno nepraktična za primjenu)

## DCT

$$F(u,v) = \frac{2}{\sqrt{M \cdot N}} \cdot C(u) \cdot C(v) \cdot \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f(i,j) \cdot \cos\left[\frac{(2 \cdot i + 1) \cdot u \cdot \pi}{2 \cdot M}\right] \cdot \cos\left[\frac{(2 \cdot j + 1) \cdot v \cdot \pi}{2 \cdot N}\right]$$

$F(u,v)$  - transformacijski koeficijent

$f(i,j)$  - amplitude elemenata slike u bloku

$u, v$  - koordinate u području transformacije (prostorne frekvencije)

$i, j$  - koordinate u području elemenata slike

$C(u)=C(v) = (1/2)^{1/2}$ , za  $u,v = 0$

$C(u)=C(v) = 1$ , za  $u = 1,2,\dots,M-1, v = 1,2,\dots,N-1$

# Transformacija: DCT

- Iz teorije se može vidjeti da je diskretna kosinusna transformacija (DCT) vrlo bliska idealnoj
- Prednost DCT:
  - Može se jednostavnije izračunati (brzi algoritmi)

## DCT za slike

- DCT kod obrade slika uglavnom se obavlja nad blokovima podataka veličine 8x8:

$$F(u,v) = \frac{1}{4} \cdot C(u)C(v) \cdot \sum_{i=0}^7 \sum_{j=0}^7 f(i,j) \cdot \cos\left[\frac{(2 \cdot i + 1) \cdot u \cdot \pi}{16}\right] \cdot \cos\left[\frac{(2 \cdot j + 1) \cdot v \cdot \pi}{16}\right]$$

■ Poduzorkovanje je vrlo "primitivna" i nekvalitetna metoda

- DC koeficijent:

$$F(0,0) = \frac{1}{8} \cdot \sum_{i=0}^7 \sum_{j=0}^7 f(i,j)$$

(DC=8 x srednja vrijednost elemenata bloka)

## IDCT

- Na sličan način definirana je i inverzna DCT:

$$f(i,j) = \frac{1}{4} \cdot \sum_{v=0}^7 \sum_{u=0}^7 C(u)C(v)F(v,u) \cdot \cos\left[\frac{(2 \cdot i + 1) \cdot u \cdot \pi}{16}\right] \cdot \cos\left[\frac{(2 \cdot j + 1) \cdot v \cdot \pi}{16}\right]$$

- Vidimo da DCT koeficijenti  $F(v,u)$  u stvari predstavljaju faktore kojima množimo bazne valne oblike pri restauraciji signala

## Pomak kod DCT

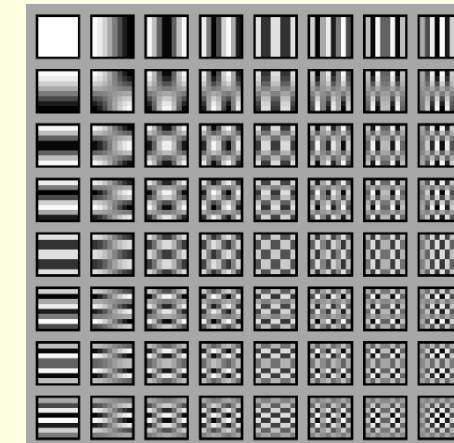
- Slikovni podaci na ulazu su pozitivni cijeli brojevi (npr. 0-255)
- S obzirom da je DCT definirana i za pozitivno i negativno područje na ovaj način bi se polovica ulaznog prostora izgubila ( a time i znatno smanjila efikasnost)
- Zato se prije DCT sve vrijednosti na ulazu translatiraju za polovicu opsega tj. -128
- Prema tome umjesto da ulazni elementi budu u opsegu [0-255] biti će u opsegu [-128 – 127]

## Opseg podataka

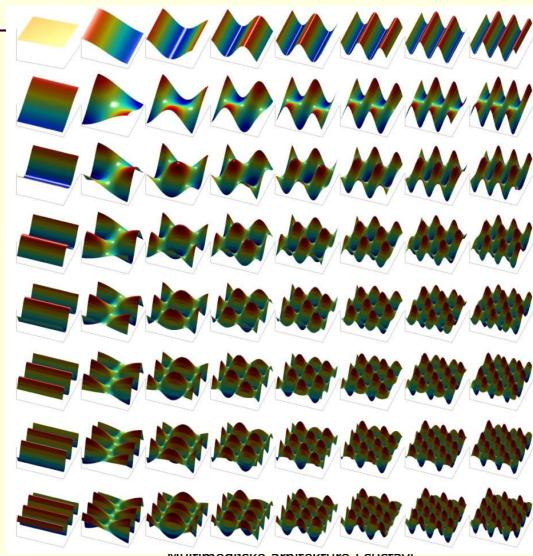
- DCT dovodi do proširenja opsega podataka (dinamički opseg za 8x8 2D DCT je  $2^3$  puta veći u odnosu na ulaz)
  - Ako na ulazu imamo 8 bitovne podatke nakon 2D DCT imati ćemo 11 bitovne koeficijente!!
- Još uvijek su SVI podaci sačuvani i moguće je obaviti perfektnu rekonstrukciju (uz uvažavanje nepreciznosti matematičkih izračuna)

## 2D-DCT bazne funkcije

bijelo pozitivne vrijednosti, crno negativne vrijednosti

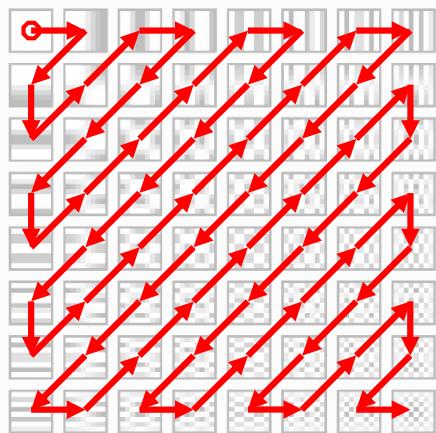


## Prikaz u 3D



46

## Cik-cak (Zig-zag) reorganizacija

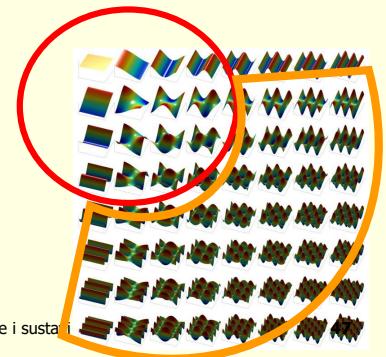


Multimedijiske arhitekture i sustavi

48

## Što smo dobili sa DCT?

- Energija signala koncentrirana u nižim frekvencijama
- Značajniji koeficijenti
- Manje značajni koef.



Multimedijiske arhitekture i sustavi

## Kako je to na stvarnoj slici....



Multimedijiske arhitekture i sustavi

49

## Kako je to na stvarnoj slici....

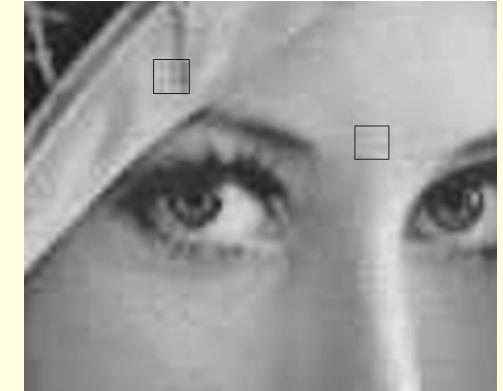


## Kako iskoristiti DCT?

- Nakon DCT, uz poznavanje prosječnih osjetljivosti oka možemo smanjivati količinu podataka koja opisuje frekvencije na koje naše oko nije osjetljivo
- To ne možemo učiniti potpunim odbacivanjem visokofrekventnih komponenata već smanjenjem njihove preciznosti (problemi naglih prijelaza !)

## Primjer

### ■ Primjer bloka



## Kvantizacija

# Što je kvantizacija

- Postupak kojim se smanjuje dinamički opseg ulaznih vrijednosti (a time i potreban broj bita)
- Ulazni podatak dijeli se sa zadanim brojem (kvantizacijski korak)
- Kod inverznog postupka podatak se množi sa kvantizacijskim korakom
- Primjer (Q=4):
  - Ulagani niz 5,11,4,17,1 (potrebno 5 bitova za prikaz)
  - Izlazni niz 1,3,1,4,0 (potrebno 3 bita za prikaz)
  - Restaurirani niz 4,12,4,16,0 (**GUBITAK PODATAKA**)
- **KVANTIZACIJOM DOLAZI DO NEPOVRATNOG GUBITKA INFORMACIJA !!!**

# Kvantizacija nakon DCT

- Utjecaj pojedinih prostornih frekvencija može se proizvoljno kontrolirati postupkom kvantizacije
- S obzirom da smo razdvojili Y i U, V komponente sada možemo Y DCT frekvencijske koeficijente kvantizirati sa različitim kvantizacijskim vrijednostima od U,V komponenti (kvant. koef. za U,V biti će veći zbog HVS)
- Također kvantizacijski koeficijenti za visoke frekvencije mogu biti veći

# Kvantizacija nakon DCT

- kvantizacijska tablica sa 64 vrijednosti  $q(u,v)$  određenih na temelju HVS (koraci kvantizacije)
- Svaki DCT koeficijent  $F(u,v)$  dijeli se sa pripadnim (skaliranim) faktorom  $q(u,v)$ , a rezultat se zaokružuje na najbližu cijelobrojnu vrijednost
- Skaliranjem sa faktorom  $S$  se pojednostavljeno određuje stupanj kompresije i "kvaliteta" slike
- vrijednosti nastale kvantizacijom  $S(u,v)$  su:

$$S(u,v) = \text{round}\left(\frac{F(u,v)}{Q(u,v)}\right) = \text{round}\left(\frac{F(u,v)}{q(u,v)S}\right)$$

# Tipične kvantizacijske tablice

Table K.1 – Luminance quantization table

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

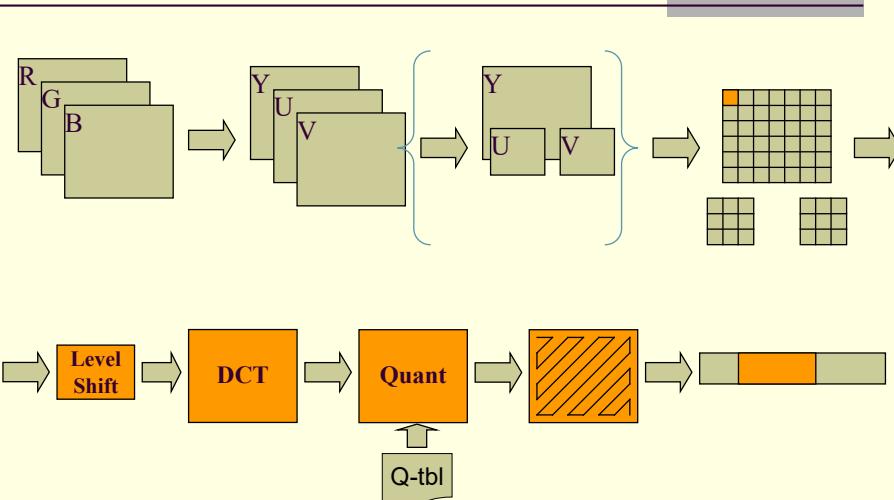
Table K.2 – Chrominance quantization table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

# Što se postiže kvantizacijom

- Dijeljenje viših frekvencija sa većim kvantizacijskim faktorima rezultira time da veliki broj koeficijenata postaje 0
- Nakon zig-zag reorganizacije koeficijenti iz 2D se prebacuju u 1D na način da važni koeficijenti dolaze prvi u nizu a manje važni koeficijenti koji imaju veliku vjerojatnost da su jednaki nuli nakon njih
- To rezultira nizovima od 64 koeficijenta sa velikim brojem nula na kraju

## JPEG koderski model



## JPEG koderski model

Pregled svih koraka

## JPEG koderski model

- Prethodnim postupcima iskorištena je većina karakteristika ljudskog vizualnog sustava
- Pogledajmo rezultate ovih koraka na primjeru bloka iz stvarne slike (blok iz lenna.pgm)



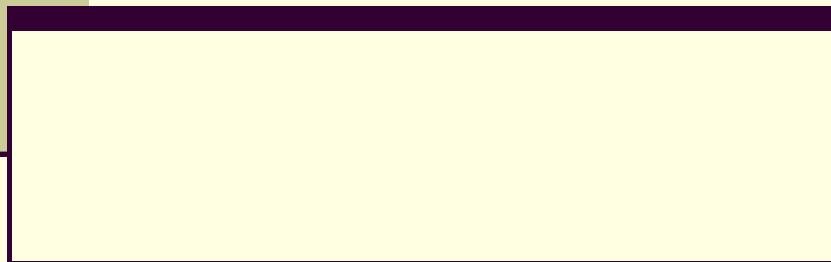
## Što nakon HVS modela

- Statističkom analizom podataka nakon obrade u koderskom modelu može se vidjeti da je njihova raspodjela izvrsna za daljnju kompresiju algoritmima zasnovanim na statističkim modelima
- Drugi dio JPEG kodera radi upravo to

Multimedijске arhitekture i sustavi

62

## Statistički/entropijski koder



## DZ1

- Prije nego predemo na statistički koder definirajmo zadatak za DZ
  - ... Nakon ovog predavanja mislim da DZ ne bi trebala biti problem...
- Natuknice: ne prepisujte, isprobajte, usporedite,...
- MAS\_DZ1

Multimedijске arhitekture i sustavi

63

## Od čega krećemo

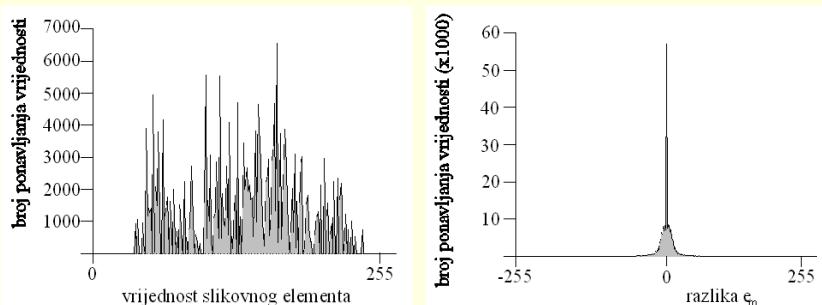
- Entropijski koder prima kvantizirane koeficijente presložene po cik-cak algoritmu u nizove od po 64 koeficijenta za svaku komponentu bloka
- Koje karakteristike možemo uočiti? (koristite DZ i proučite rezultantne vrijednosti za nekoliko slika)

Multimedijске arhitekture i sustavi

65

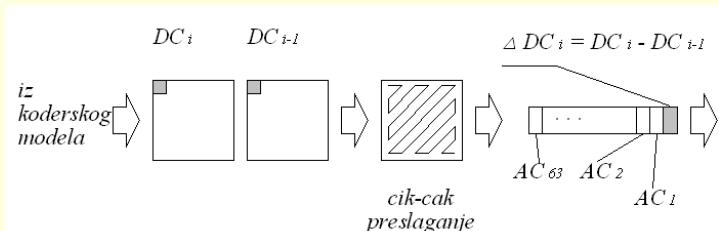
# DC koeficijenti

- Podloga: analiza statistike DC komponenata



## DC razlika

- Iz prethodnoga odlučujemo da ne želimo kodirati vrijednost DC elementa već razliku između trenutnog i prethodnog
- Znatna ušteda u bitovima (malo po malo...)



# DC koeficijenti

- Teorijska podloga: DC koeficijent predstavlja srednju vrijednost svih 64 elementa bloka
  - dva susjedna bloka obično imaju vrlo slične srednje vrijednosti

## Rezultat:

- Distribucija DC vrijednosti je prilično jednolika i nepogodna za kodiranje
- Distribucija RAZLIKE dva susjedna DC koeficijeta je vrlo gusta oko nule i idealna za kodiranje

## Kodiranje duljine niza nula (ZRL)

- Slijedeći korak u našoj analizi proizlazi iz postupka kvantizacije nakon koje je mnogo elemenata viših frekvencija postalo nula.
- Za ovakve nizove izuzetno je pogodna metoda kodiranja duljine niza (ZRL coding)
- Metoda radi na način da umjesto da se kodira svaki koeficijent koji je nula u nizu, da se kodira broj uzastopnih nula koji se nalazi u nizu

## DC/AC simboli

- Dodatno se u statističkom modelu određuje pripadnost pojedinog ulaznog elementa određenoj kategoriji prema apsolutnoj vrijednosti njegove amplitude
- Određivanje kategorije povezano je sa postupkom modificiranog Huffmanovog kodiranja o čemu će biti više riječi kasnije. Na izlazu iz statističkog modela svaka ulazna DC razlika i svaki ulazni AC koeficijent različit od nule biti će zamjenjeni sljedećim simbolima:
  - DC simbol: **[kategorija] [amplituda]**
  - AC simbol: **[[duljina niza nula],[kategorija]] [amplituda]**
- DC simbol nema komponentu koja određuje duljinu niza nula što je logično jer JPEG zasebno obraduje svaki slikovni blok a DC razlika uvijek je prvi element bloka.
- Za AC komponente duljina niza nula opisuje koliko koeficijenata ima vrijednost nula prije nekog koeficijenta koji je različit od nule.

## Kategorija

Kategorija	DC razlika	AC koeficijent
0	0	
1	-1,1	-1,1
2	-3,-2,2,3	-3,-2,2,3
3	-7,...,-44,...,7	-7,...,-44,...,7
4	-15,...,-8,8,...,15	-15,...,-8,8,...,15
5	-31,...,-16,16,...,31	-31,...,-16,16,...,31
6	-63,...,-32,32,...,63	-63,...,-32,32,...,63
7	-127,...,-64,64,...,127	-127,...,-64,64,...,127
8	-255,...,-128,128,...,255	-255,...,-128,128,...,255
9	-511,...,-256,256,...,511	-511,...,-256,256,...,511
10	-1023,...,-512,512,...,1023	-1023,...,-512,512,...,1023
11	-2047,...,-1204,1024,...,2047	

## ZRL/EOB

- Veličina polja koje opisuje duljinu niza normom je ograničena na 4 bita kojima se mogu opisati nizovi od 0 do 15 nula. U stvarnom nizu ulaznih elemenata može se pojaviti i niz koji sadrži i više od 15 nula te se u tom slučaju koristi specijalni simbol nazvan ZRL (Zero Run Length).
- ZRL simbol označava niz od 16 nula a nakon tog simbola nule se počinju brojati iz početka.
- Drugi, i zadnji, specijalni simbol označava da su od trenutnog elementa do kraja bloka svi elementi jednaki nuli. Ovaj simbol ima oznaku EOB (End Of Block).
- Slikovni blok sastoji se od 64 ulazna elementa te se nakon obrade u statističkom modelu na izlazu može naći najviše tri simbola ZRL. U normi je također definirano kako se niz ZRL simbola nakon kojih slijedi EOB simbol mora izbaciti iz izlaznog niza.
  - Razlog je prirođan. EOB simbol označava sve nule od trenutnog mjesto do kraja bloka pa su prema tome ZRL simboli koji eventualno prethode EOB simbolu nepotrebni.

## Kategorija

- Neke vrijednosti se rijetko pojavljuju pa nije imalo smisla svakom broju pridjeljivati kod kategorije
- Kategorije su određene prema vjerojatnosti pojavljivanja ulaznih elemenata
- Kako se kategorijom ne može točno odrediti vrijednost elementa poslije kategorije mora se poslati još podatak [amplituda] koji unutar kategorije definira koji je to element.
- Amplituda ima različit broj bitova za svaku kategoriju (koji je jednak rednom broju kategorije)

## Tablica simbola za AC komponente

Duljina niza	Kategorija				
	0	1	2	...	10
0	EOB	0/1	0/2	...	0/10
1	x	1/1	1/2	...	1/10
:	x	:	:		:
14	x	14/1	14/2	...	14/10
15	ZRL	15/1	15/2	...	15/10

## Huffman-ova tablica za DC simbole

Kategorija	Duljina koda	Kodna riječ
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

## Amplituda

- Za kodiranje amplitude koristi se sljedeće pravilo:
- Pretpostavimo da je koeficijent C zapisan u formatu dvojnog komplementa, a K je kategorija kojoj taj koeficijent pripada.
  - Ako je C pozitivan broj tada će se Huffmanovom kodu, kao proširenje, dodati K nižih bitova od C.
  - Ako je C negativan tada će se Huffmanovom kodu dodati K nižih bitova od vrijednosti koeficijenta C umanjenog za jedan, tj. (C-1).

## Huffman-ova tablica za AC simbole

Duljina niza/Kategorija	Duljina koda	Kodna riječ
0/0 (EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	111111110000010
0/10	16	1111111110000011
1/1	4	1100
1/2	5	11011
...	...	...

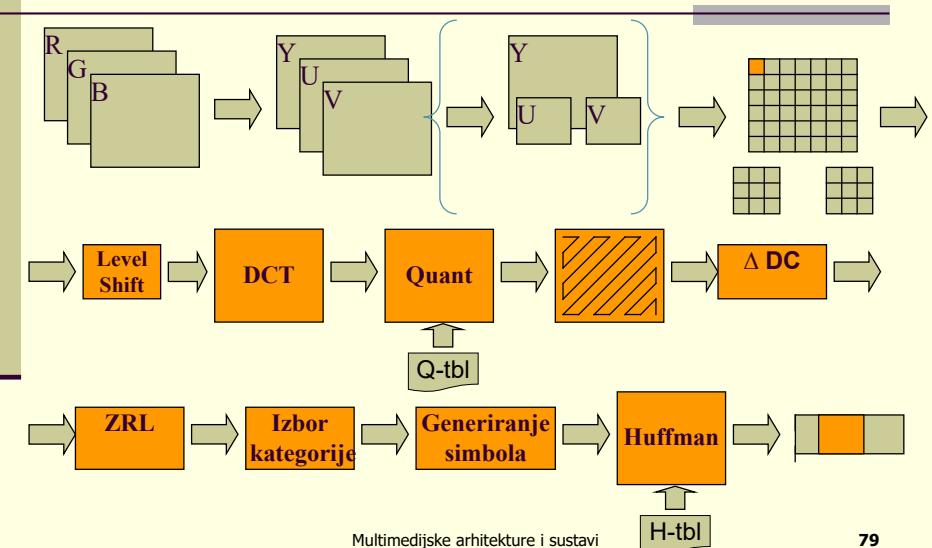
## JPEG koder



## JPEG grayscale

Potpuni primjer kodiranja jednog bloka

## JPEG koder



Multimedijiske arhitekture i sustavi

79

## Pomak (level shift)

$$\begin{bmatrix} 107 & 113 & 122 & 122 & 122 & 139 & 137 & 139 \\ 107 & 113 & 122 & 122 & 122 & 139 & 133 & 139 \\ 107 & 122 & 122 & 122 & 126 & 133 & 133 & 148 \\ 107 & 113 & 122 & 115 & 122 & 131 & 131 & 148 \\ 107 & 113 & 122 & 127 & 130 & 139 & 139 & 148 \\ 107 & 113 & 113 & 115 & 122 & 131 & 133 & 147 \\ 109 & 107 & 122 & 122 & 131 & 131 & 131 & 133 \\ 103 & 109 & 118 & 115 & 122 & 122 & 122 & 140 \end{bmatrix} \xrightarrow{\text{Pomak}} \begin{bmatrix} -21 & -15 & -6 & -6 & -6 & 11 & 9 & 11 \\ -21 & -15 & -6 & -6 & -6 & 11 & 5 & 11 \\ -21 & -6 & -6 & -6 & -2 & 5 & 5 & 20 \\ -21 & -15 & -6 & -13 & -6 & 3 & 3 & 20 \\ -21 & -15 & -6 & -1 & 2 & 11 & 11 & 20 \\ -21 & -15 & -15 & -13 & -6 & 3 & 5 & 19 \\ -19 & -21 & -6 & -6 & 3 & 3 & 3 & 5 \\ -25 & -19 & -10 & -13 & -6 & -6 & -6 & 12 \end{bmatrix} \xrightarrow{\text{DCT}}$$

Multimedijiske arhitekture i sustavi

81

## 2D DCT, kvantizacija

$$\begin{array}{l} \text{DCT} \\ \rightarrow \begin{bmatrix} -31 & -84 & 2 & -16 & -5 & -10 & 13 & -3 \\ 12 & -2 & 1 & 1 & -9 & 2 & 1 & -7 \\ -11 & 7 & -6 & 3 & -1 & 4 & 1 & 1 \\ 5 & -6 & -2 & 7 & -2 & 6 & -1 & -2 \\ -1 & -2 & 1 & -4 & 0 & 0 & 2 & 1 \\ -1 & 3 & -0 & 1 & -0 & 1 & 0 & 4 \\ -2 & -2 & 9 & -4 & 1 & -6 & -4 & -3 \\ 12 & 0 & -8 & 1 & 1 & 2 & -3 & 0 \end{bmatrix} \xrightarrow{\text{Kvantizacija.}} \begin{bmatrix} -2 & -8 & 0 & -1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{Cik-cak}} \end{array}$$

Multimedijiske arhitekture i sustavi

82

## JPEG+JFIF

Kako je zapisana slika u računalu

## Cik-cak + entropijsko kodiranje

-2 -8 1 -100 -10 10 0 0 0 00 -10 ....0

[011:01][1011:0111][00:1][00:0][11100:0][1100:1][1111011:0][1010]

01101101101110010001110001100111101101010

Multimedijiske arhitekture i sustavi

83

## Kako se slika zapisuje i prenosi

- Niz binarnih podataka pohranjen ne može se interpretirati...
  - koja je dimenzija, boje/cb, kako je kvantizirano,...
- Upravo zato JPEG norma definira način zapisa podataka vezanih za kompresiju slike

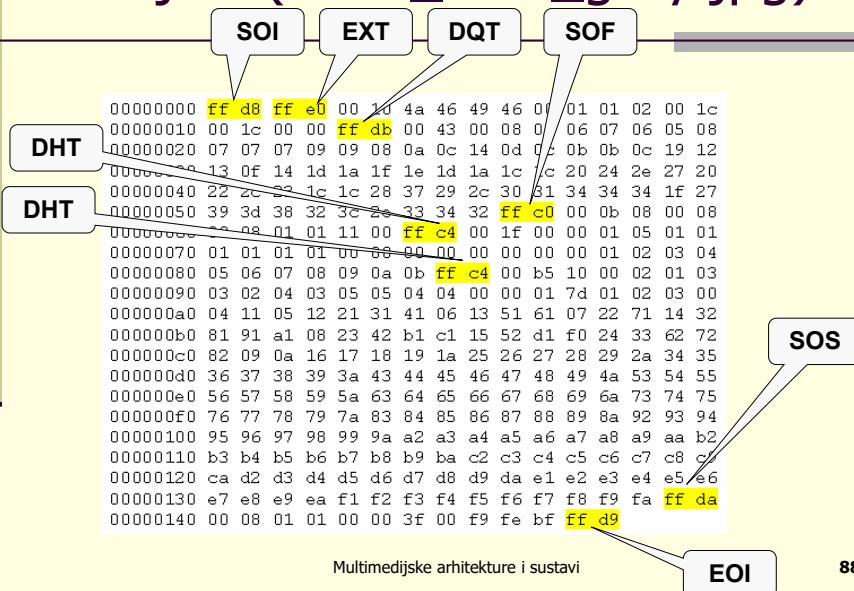
Multimedijiske arhitekture i sustavi

85

# Markeri

- Sve počinje definiranjem MARKERA
- Markeri su specijalni dvo-bajtni podaci koji počinju sa podatkom 0xFF kojeg slijedi podatak različit od 0 ili 0xFF a kojim se identificiraju različiti strukturalni dijelovi kompresiranog niza podataka

## Primjer: (test1\_crno\_gray.jpg)



# Minimalna struktura JPEG datoteke

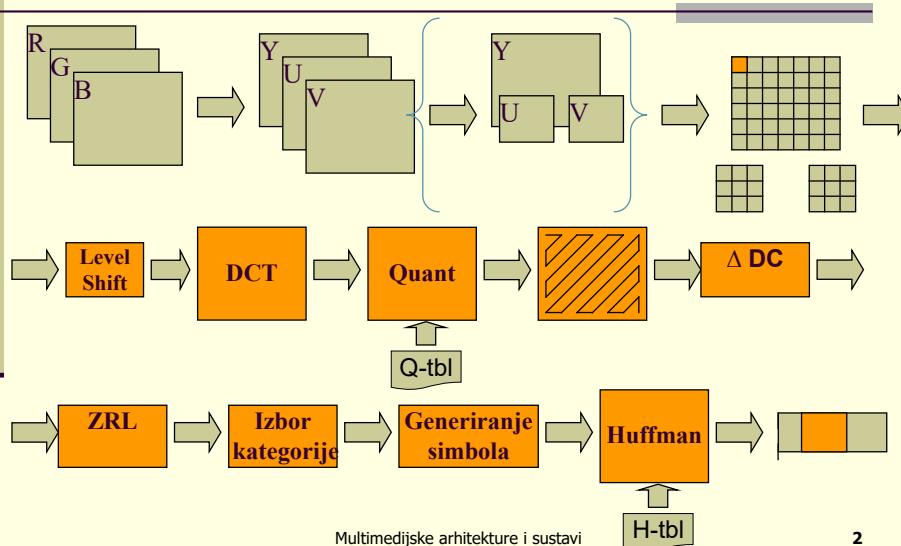
SOI	FFD8	Start of Image
DQT	FFDB	Quantization table(s)
DHT	FFC4	Huffman table(s)
SOF	FFC0	Frame header
SOS	FFDA	Scan header
Kompresirani podaci		
EOI	FFD9	End of Image

# Multimedijijske arhitekture i sustavi

Prof.dr.sc. Mario Kovač  
Prof.dr.sc. Hrvoje Mlinarić



## JPEG koder



## Količine podataka

- Jednostavan izračun daje nam okvirne količine nekih tipičnih formata koje ste upoznali

Rezolucija slike	Bita/pikselu	Veličina
640x480	8	307kB
1280x1024	24	3,9MB

Video rezolucija	Slika/s	Veličina/s
640x480 (8bpp)	10	3 MB/s
1280x1024 (24bpp)	30	120 MB/s

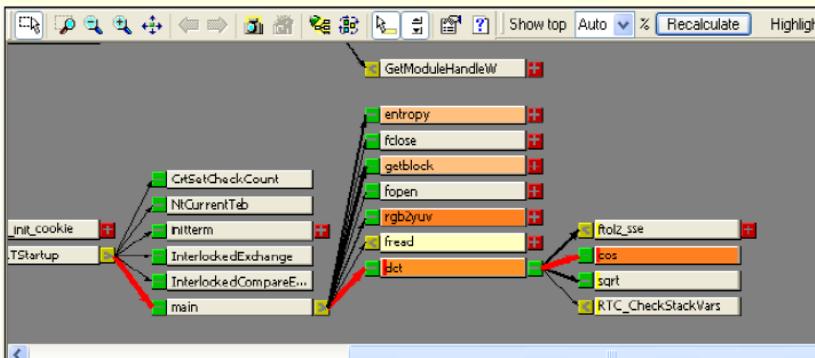
## Cilj...

- U prethodna dva predavanja uzeli smo primjer jednog tipičnog mm algoritma
- U nastavku cilj će nam biti vidjeti kako se neki od algoritama implementiraju u stvarnom svijetu : IZVEDBENI POGLED NA TEHNOLOGIJE
- S obzirom da je cilj diplomskog studija omogućiti rješavanje kompleksnih problema...

## Najzahtjevnije operacije

- Obrađivati takve podatke se može na različite načine
- Teoretski kad razmatramo kompresija npr slika ili video je nužna....no ako je želimo izvesti onda se susrećemo sa tipičnim inženjerskim problemima
- Pokušajmo predvidjeti kompleksnost nekih funkcijskih blokova a onda se uvjerimo kakva je uistinu kompleksnost

## I sada počinje ozbiljna analiza..



Multimedijiske arhitekture i sustavi

6

## Analiza izvedbe JPEG algoritma

- Iz tablica koje smo ranije proučili vidljivo je da je podataka kod slika i videa puno... no mi mislimo da su naši današnji procesori brzi...
- NAPOMENA: u okviru ovog predmeta nećemo ulaziti u detaljne analize efikasnosti prevoditelja, OS-a i razvojne okoline. No zapamtite da efikasnost rješenja može ZNAČAJNO ovisiti o ovim a i nekim dodatnim uvjetima

Multimedijiske arhitekture i sustavi

8

## Kako to implementirati u SW ili HW

### Najzahtjevnije operacije pri obradi slike/videa

- Procjena/predviđanje pokreta !!!!!....!!!
- DCT
- RGB-YUV transformacija
- Entropijsko kodiranje

Multimedijiske arhitekture i sustavi

7

## Konverzija prostora boja

- Kao što ste naučili RGB način zapisa podataka za sliku nije dobar za kompresiju podataka već se za to koristi YUV (YCrCb) prostor
- Prije pokretanja kompresije slika u RBG zapisu konvertira se u YUV
- Inverzna transformacija obavlja se nakon dekompresije a prije prikaza na zaslonu

Multimedijiske arhitekture i sustavi

9

## Konverzija prostora boja

- RGB-YCbCr konverzija vrlo se lako može obaviti jednostavnom matričnom operacijom:

$$Y = 0.299 R + 0.587 G + 0.114 B$$

$$Cb = -0.1687 R - 0.3313 G + 0.5 B + 128$$

$$Cr = 0.5 R - 0.4187 G - 0.0813 B + 128$$

- Vidimo da za izračun svakog piksela treba 9 množenja i 8 zbrajanja (+dohvat,spremanje)

## Konverzija prostora boja

- Verzija 2: osnovna approximacija

$$Y = 0.299 R + 0.587 G$$

$$Cb = -0.3313 G + 0.5 B + 128$$

$$Cr = 0.5 R - 0.4187 G + 128$$

- Verzija 3: poboljšana approximacija

```
data[i++] = (byte)(R/4 + G/2);
```

```
data[i++] = (byte)(-(G/4) + B/2 + 128);
```

```
data[i++] = (byte)(R/2 - (G / 2) + 128);
```

## Konverzija prostora boja

- Iako matematički trivijalna, konverzija boja jedan je od računalno zahtjevnijih zadataka pri kompresiji
- Ako izračunamo koliko je potrebno da se obradi jedna slika 1280x1024:
  - 11,8M množenja
  - 10,5M zbrajanja
  - +dohvat, spremanje

## Analiza

- Za jednu vrlo zahtjevnu operaciju uspjeli smo značajno smanjiti procesorske zahtjeve
- No to smo postigli uz **degradaciju** kvalitete izračuna podataka
- Da li je kvaliteta zadovoljavajuća ili ne vrlo je teško empirijski definirati te je zato potrebno napraviti mnogo testova i subjektivnih provjera

## Zadatak za vježbu

- neobavezno
- Korištenjem bilo kojeg alata (Visual Studio, Matlab, Mathematica,...) usporedite kvalitetu tri ranije opisane metode aproksimacije konverzije prostora boja
- Postupak:
  - Napišite funkciju koja učitava sliku u boji u RGB
  - Napišite tri različite funkcije transformacije boja (od kojih je jedna po punim formulama)
  - Izračunajte MSE za svaku komponentu (Y,Cb,Cr) za jednu testnu sliku za dva aproksimativna rješenja

## Načini optimizacije algoritama

- Neke osnovne grupe optimizacija:
  - Smanjenje preciznosti izračuna
  - Razvoj ekvivalentnih matematičkih algoritama sa manjom kompleksností
  - Promjena programske razvojne okoline
  - Promjena programske izvedbene okoline
  - Promjena arhitekture sustava za izvođenje

## Načini optimizacije algoritama

- Vidjeli smo prvi primjer jedne jednostavne metode za optimizaciju nekog algoritma
- Iako se može činiti da je ovo dobar pristup njega na žalost ne možemo primjeniti u većini situacija
- Ponekad nije dozvoljeno unošenje ovako značajnih grešaka u izračune no ipak se računalni zahtjevi moraju značajno smanjiti

## Načini optimizacije algoritama

- U projektiranju visokoefikasnih proizvoda morati ćemo se vrlo često poslužiti SVIM dostupnim metodama i njihovim kombinacijama
- U nastavku ćemo proučiti na koji način se može pristupiti optimizaciji i izvedbi nekih ključnih dijelova multimedijskih algoritama

## Načini optimizacije algoritama

- Da bi mogli razmotriti moguće načine optimiranja MORAMO DOBRO POZNAVATI:
  - Algoritme koje optimiramo
  - Programska rješenja koja koristimo
  - Arhitekturu sustava na kojem se algoritmi izvode

## JPEG

### ■ 2D-DCT, 2D-IDCT

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos((2x+1)u\pi)/2N \cos((2y+1)v\pi)/2N$$

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) C(u, v) \cos((2x+1)u\pi)/2N \cos((2y+1)v\pi)/2N$$

- pokušajmo proučiti koliko je to operacija po 2D-DCT elementu...

## JPEG

- RGB2YCbCr:
  - Definitivno velika količina podataka
  - Osnovna metoda: 9\*, 8+ po pikselu
  - $(1280 \times 1240) \approx 10^7$  množenja,  $10^7$  zbrajanja
- Vidjeli smo jednu metodu za smanjenje kompleksnosti ove konverzije

## JPEG - DCT

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos(((2x+1)u\pi)/2N) \cos(((2y+1)v\pi)/2N) \quad \alpha(u) = \begin{cases} \sqrt{1/N} & \text{for } u = 0 \\ \sqrt{2/N} & \text{for } u = 1, 2, \dots, N-1 \end{cases}$$

- Za svaki element:

- $\alpha() \alpha() \sum f^* \cos()^* \cos()$
- Teoretski 66 množenja, 63 zbrajanja, 128 izračuna  $\cos()$  funkcije (koja samo u parametrima ima 1 dijeljenje, 3 množenja, 1 zbrajanje)
- Pokušajte ovo izračunati u bilo kojem programskom jeziku.....

## JPEG - DCT

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos(((2x+1)u\pi)/2N) \cos(((2y+1)v\pi)/2N) \quad \alpha(u) = \begin{cases} \sqrt{1/N} & \text{for } u = 0 \\ \sqrt{2/N} & \text{for } u = 1, 2, \dots, N-1 \end{cases}$$

- Primjer približne kompleksnosti računanja po teorijskoj formuli...
- Očito je da ovo ovako nije iskoristivo ni za kakve primjene....

## JPEG – 2D-DCT odvojivost

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos(((2x+1)u\pi)/2N) \cos(((2y+1)v\pi)/2N) \quad \alpha(u) = \begin{cases} \sqrt{1/N} & \text{for } u = 0 \\ \sqrt{2/N} & \text{for } u = 1, 2, \dots, N-1 \end{cases}$$

- Jedna od najvažnijih karakteristika 2D DCT je odvojivost (engl. separability).
- Koristeći tu karakteristiku, rezultat transformacije moguće je izračunati preko 1D transformacije nad svim redovima nakon čega slijedi 1D transformacija nad svim stupcima.
- Kao rezultat ovoga, u literaturi se mogu pronaći dva pristupa rješavanju 2-D DCT: "potpunim" 2D postupkom ili korištenjem dva 1D postupka
- 2D pristup: nešto efikasniji, znatno kompleksniji

## JPEG - DCT

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos(((2x+1)u\pi)/2N) \cos(((2y+1)v\pi)/2N) \quad \alpha(u) = \begin{cases} \sqrt{1/N} & \text{for } u = 0 \\ \sqrt{2/N} & \text{for } u = 1, 2, \dots, N-1 \end{cases}$$

- Postoji mnogo različitih matematičkih i računalnih metoda kako efikasno izračunati ovakvu funkciju
- Pogledati ćemo neke najvažnije
- Odvojivost (separability), lookup tablice,...

## 2D DCT izvedba

- Za izvedbu DCT, ali i svake druge operacije programski ili na integriranom sklopu, od velike je važnosti procijeniti ukupnu efikasnost s potrebnom količinom resursa/logike (što na kraju rezultira u vremenu obrade ili površini silicija, a time i cijeni).

## 1D DCT

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos(((2x+1)u\pi)/2N)$$

- Ako pogledamo kompleksnost vidimo da je približna kompleksnost ove formule za svaki element:
  - $\alpha() \sum f^* \cos()$
  - Teoretski 9 množenja, 7 zbrajanja, 8 izračuna cos() funkcije (koja u parametrima ima 1 dijeljenje, 3 množenja, 1 zbrajanje)

## 2D DCT

- 2D DCT za blok 8x8:
  - 4096 množenja, 4032 zbrajanja
- Samo za jednu sliku 1280x1024
  - 83886080 množenja, 82575360 zbrajanja

## 2D DCT preko 1D DCT

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos(((2x+1)u\pi)/2N)$$

- Priličan problem je cos() funkcija
- S obzirom na diskretizirane vrijednosti parametara cos() faktori se NE računaju već se koriste unaprijed izračunate vrijednosti pohranjene u tablicu (tzv.lookup tablica) te se kao takvi oni obično i kombiniraju sa  $\alpha()$  parametrom.
- Samo ovim postupkom ZNAČAJNO smo smanjili kompleksnost računanja
- No i sa tako izvedenim cos() kompleksnost je velika

## 2D DCT preko 1D DCT

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos(((2x+1)u\pi)/2N)$$

### Koristeći svojstvo odvojivosti

- 1D DCT za 8 podataka
  - 64 množenja, 56 zbrajanja
- 2D DCT preko 1D DCT
  - 1024 množenja, 896 zbrajanja
- Samo za jednu sliku 1280x1024
  - 20971520 množenja, 18350080 zbrajanja

## 2D DCT

- No i ovo što smo do sada predložili nije dovoljno da u stvarnom svijetu izvedete neki algoritam za kompresiju uz razumnu potrošnju resursa (procesora, vremena, energije,...)
- Potreban je novi pristup računanju....
- BRZI ALGORITMI

...

- Na taj način mogu se identificirati elementi:
  - $C_k$
  - $S_k$
  - $s_{jk}$  i  $d_{jk}$

$$C_k = \cos(k\pi/16)$$
$$S_k = \sin(k\pi/16)$$

$$C_1 = S_7 = 0.9808$$
$$C_2 = S_6 = 0.9239$$
$$C_3 = S_5 = 0.8315$$
$$C_4 = S_4 = 0.7071$$
$$C_5 = S_3 = 0.5556$$
$$C_6 = S_2 = 0.3827$$
$$C_7 = S_1 = 0.1951$$

$$s_{jk} = s(j) + s(k)$$
$$s_{07} = s(0) + s(7)$$
$$s_{16} = s(1) + s(6)$$
$$s_{25} = s(2) + s(5)$$
$$s_{34} = s(3) + s(4)$$
$$s_{0734} = s_{07} + s_{34}$$
$$s_{1625} = s_{16} + s_{25}$$

$$d_{jk} = s(j) - s(k)$$
$$d_{07} = s(0) - s(7)$$
$$d_{16} = s(1) - s(6)$$
$$d_{25} = s(2) - s(5)$$
$$d_{34} = s(3) - s(4)$$
$$d_{0734} = s_{07} - s_{34}$$
$$d_{1625} = s_{16} - s_{25}$$

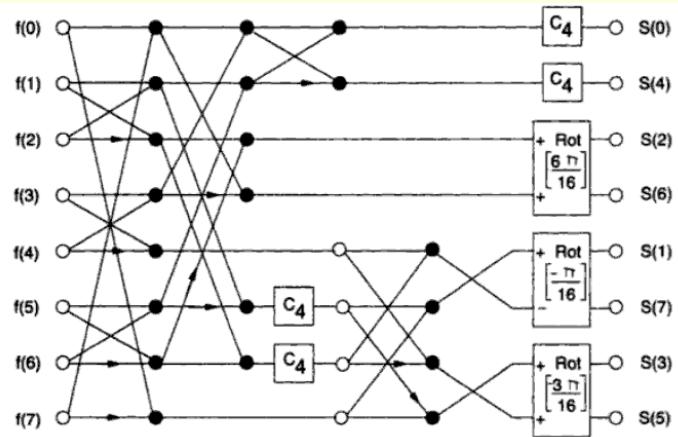
## Simetrije, tablice,..., i još ...

- Kad razmatramo kako implementirati neki algoritam moramo pokušati proučiti problem sa mnogih strana
- U slučaju DCT analizom formula za 1D DCT mogu se uočiti neke simetrije članova koji se računaju te uz malo trigonometrije reducirati broj izračuna  $\cos()$  funkcije
- Isto tako mogu se uočiti neki parovi elemenata koji se zbrajaju i oduzimaju

## Ligtenberg, Vetterli

- Na temelju prethodnih analiza Ligtenberg i Vetterli su izveli formule za 1D DCT:
  - $2S(0)=C4((s07+s12)+(s34+s56))$
  - $2S(1)=C1d07 + C3d16 + C5d25 + C7d34$
  - ...
- Također se može vidjeti da ako dva podatka x i y želimo rotirati za kut  $k\pi/16$  tada nove vrijednosti iznose:
  - $X=Ck*x+Sk*y$
  - $Y=-Sk*x+Ck*y$

# Ligtenberg, Vetterli



## Brzi DFT....

- Bitno drugačiji pristup računanju DCT predložili su neki autori koji su se intenzivno bavili i proučavanjem brzih algoritama za računanje diskretne Fourierove transformacije (DFT). Tako je u svom radu Haralick pokazao da se DCT s N točaka može izračunati koristeći dvije brze Fourierove transformacije (FFT) s N točaka koristeći se simetrijom ulaza.
- Kroz brojne članke (npr. Tseng i Miller) pokazalo se kako se DCT može efikasnije izračunati na način da se, uz simetrično raspoređene ulaze, koriste samo realni dijelovi prvih N koeficijenata iz DFT s 2N točaka.

# Ligtenberg, Vetterli

- Kao rezultat ovog razmatranja Ligtenberg i Vetterli su izveli algoritam koji zahtjeva samo 13 operacija množenja i 29 operacija zbrajanja za računanje 1-D DCT.
- Već ovaj algoritam pokazuje drastično smanjenje broja matematičkih operacija u usporedbi s konzervativnim načinom računanja.

## Skalirana DCT

- Ovi radovi pored toga uvode dodatnu prednost:
  - Naime kod konzervativnog pristupa ili kod primjera Lightenbergovog i Vetterljevog algoritma, kvantizacija koeficijenata koja slijedi transformaciju uvodi dodatne matematičke operacije u postupak.
  - Algoritam Tsenga i Millera naziva se tzv. Skaliranim algoritmom jer za dobivanje DCT koeficijenata rezultati DFT se moraju pomnožiti s određenim konstantama (skalirati).
  - Ako nakon postupka transformacije odmah slijedi kvantizacija tada se kvantizacijski koraci mogu prije pomnožiti s konstantama skaliranja. Na taj način na izlazu će se bez dodatnih operacija dobiti kvantizirani DCT koeficijenti. Koristeći se ovim načinom razmišljanja, DCT s 8 točaka koja se koristi u JPEG normi može biti zamijenjena DFT-om s 16 točaka i operacijom skaliranja.

## Brzi algoritmi – što se koristi

- Tipično se za izračun 1D DCT preko 1D FFT koristi teorija:
  - AAN algoritam za skalirani 1D DCT (8 točaka):
    - 5 množenja, 29 zbrajanja, 16 dvojnih komplementa
  - Kovač, Ranganathan algoritam za skalirani 1D DCT (8 točaka):
    - 5 množenja, 29 zbrajanja, 12 dvojnih komplementa

## Uštede ....

- Korištenjem ovih brzih algoritama za jednu sliku 1280x1024 potrebno je približno:
  - 1638400 množenja
  - 9502720 zbrajanja
  - 3932160 dvojnih komplementa
- Najvažnije je da je broj operacija množenja (SPORO u procesoru) smanjen cca. 15 puta

## KR algoritam

**Korak 1:**

$$\begin{array}{llll} b_0 = a_0 + a_7; & b_1 = a_1 + a_6; & b_2 = a_3 - a_4; & b_3 = a_1 - a_6; \\ b_4 = a_2 + a_5; & b_5 = a_3 + a_4; & b_6 = a_2 - a_5; & b_7 = a_0 - a_7; \end{array}$$

**Korak 2:**

$$\begin{array}{llll} c_0 = b_0 + b_5; & c_1 = b_1 - b_4; & c_2 = b_2 + b_6; & c_3 = b_1 + b_4; \\ c_4 = b_0 - b_5; & c_5 = b_3 + b_7; & c_6 = b_3 + b_6; & c_7 = b_7; \end{array}$$

**Korak 3:**

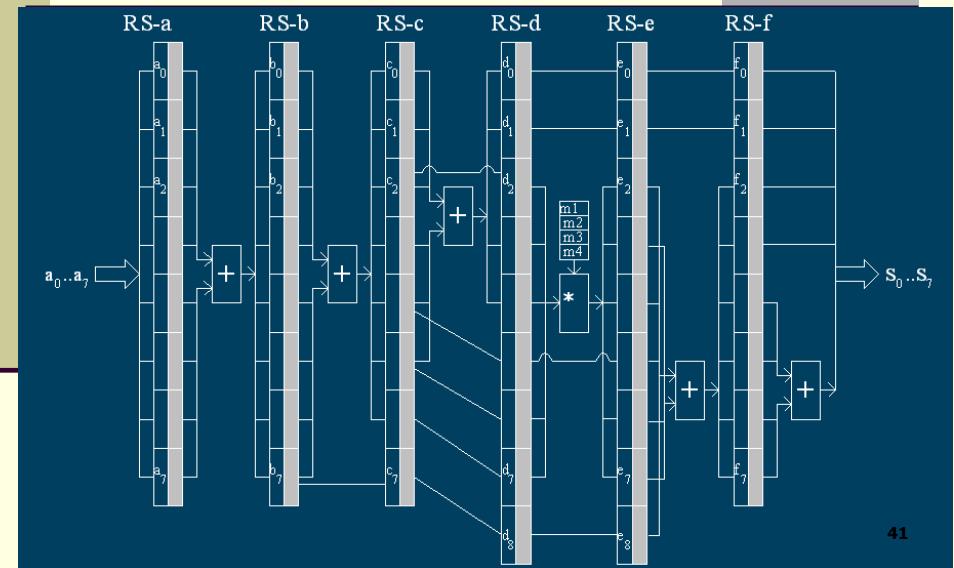
$$\begin{array}{llll} d_0 = c_0 + c_3; & d_1 = c_0 - c_3; & d_2 = c_2; & d_3 = c_1 + c_4; \\ d_4 = c_2 - c_5; & d_5 = c_4; & d_6 = c_5; & d_7 = c_6; & d_8 = c_7; \end{array}$$

**Korak 4:**

$$\begin{array}{llll} e_0 = d_0; & e_1 = d_1; & e_2 = m_3 * d_2; & e_3 = m_1 * d_7; \\ e_4 = m_4 * d_6; & e_5 = d_5; & e_6 = m_1 * d_3; & e_7 = m_2 * d_4; & e_8 = d_8; \end{array}$$

..... I tako dalje

I što nam to još omogućuje... (o tome će riječi biti kasnije)



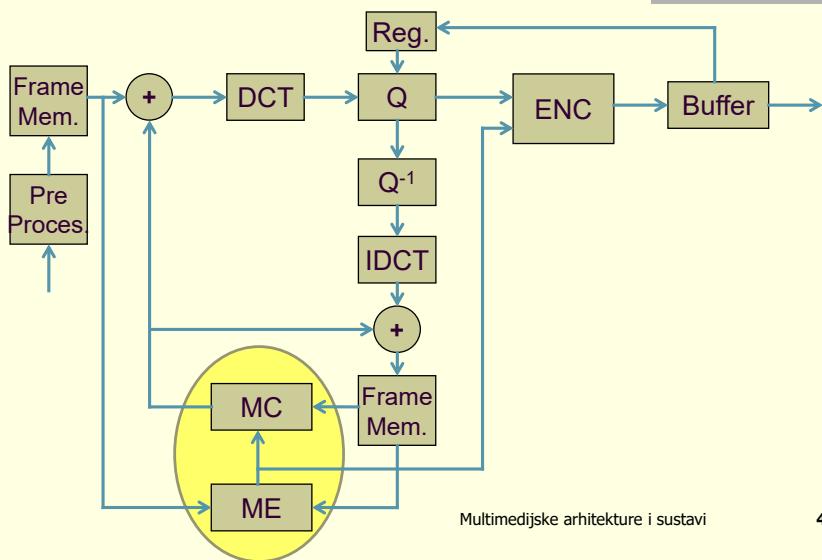
## Zadatak za vježbu (neobavezno)

- Napišite program koji će računati 2D DCT za neku ulaznu sliku koristeći osnovni (teorijski) algoritam i koristeći odvojivost te 1D DCT

## MPEG

Procjena kompleksnosti

## MPEG koder



## Intra i inter-blokovska kompresija

- **Unutar-blokovska kompresija**
  - Uklanjanje informacija visokih frekvencija koje ljudsko oko ne može prepoznati
  - Isto kao kod kompresije slika (JPEG) što je prije objašnjeno
- **Među-blokovska kompresija**
  - Smanjivanje vremenske redundancije
- Obje se metode zasnivaju na karakteristikama ljudskog vizualnog sustava (HVS)

## Međublokova kompresija

- Često se pojedini dijelovi slike unutar niza vrlo malo mijenjaju ili su čak nepromijenjeni
- Pozadina slike često se ne mijenja



## Vremenska redundancija

AUTO UZ

## Vremenska redundancija

CARJUMP

## Vremenska redundancija

BRZI AUTO

# Procjena i kompenzacija pokreta

## ■ Procjena pokreta (motion estimation, ME):

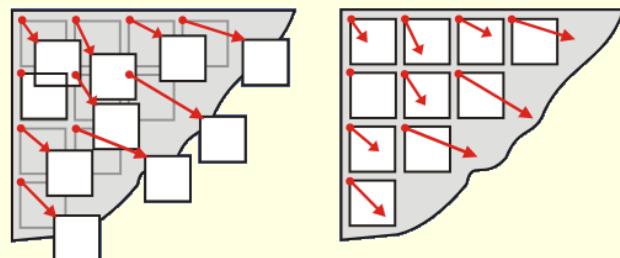
- Izdvajanje dijelova slike (segmentacija pokreta)
- Opisivanje pokreta (procjena)
- Izvodi se u koderu

## ■ Kompenzacija pokreta (motion compensation, MC):

- Korištenje rezultata procjene pokreta
- Izvodi se u dekoderu

# Procjena i kompenzacija pokreta

- Algoritmi procjene pokreta za svaki blok računaju pripadni **vektor pomaka**
- Vektor pokazuje izvorište bloka u prethodnoj slici prije pomaka na poziciju u trenutnoj slici



# Procjena i kompenzacija pokreta

- Kako bi omogućili jednostavniju implementaciju nameću se potrebna pojednostavljenja:

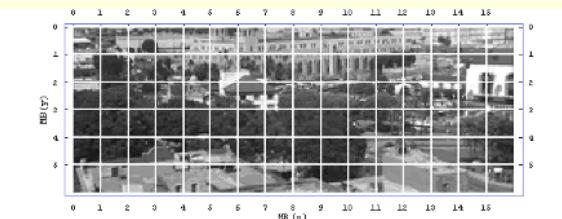
- Segmentacija na pravokutne dijelove unaprijed znanih dimenzija (blok)
- Svaki je pokret translacijski (vektor pomaka)

# Procjena i kompenzacija pokreta

PROCJENA

## Slika kao pojedinačni objekt pri kompresiji

- Radi efikasnije obrade za procjenu pokreta dovoljna je **komponenta Y**



Multimedijске arhitekture i sustavi

54

## Mjera poremećaja – primjer

- Neke moguće mjere poremećaja dvaju blokova:
  - Srednja kvadratna pogreška (MSE) – računski zahtjevna
$$MSE(x, y; d_x, d_y) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [Y(x+i, y+j, t_1) - Y(x+d_x+i, y+d_y+j, t_0)]^2$$
  - Srednji absolutni poremećaj (MAD) – široko prihvaćena
$$MAD(x, y; d_x, d_y) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |Y(x+i, y+j, t_1) - Y(x+d_x+i, y+d_y+j, t_0)|$$
  - Broj podudarajućih slikovnih elemenata (MPC)

$$MPC(x, y; d_x, d_y) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} T[Y(x+i, y+j, t_1), Y(x+d_x+i, y+d_y+j, t_0)]$$

$$T(\alpha, \beta) = \begin{cases} 1, & \text{za } |\alpha - \beta| \leq T_0 \\ 0, & \text{inace} \end{cases}$$

Multimedijске arhitekture i sustavi

56

## Mjera poremećaja

- Mjera sličnosti dvaju blokova jest **mjera poremećaja** slike od jednog trenutka do drugog
- Procjena pokreta svodi se na **optimizaciju** mjerne poremećaja kao kriterijske funkcije
- Algoritmi u definiranom **području pretraživanja** nastoje pronaći minimum kriterijske funkcije
  - Promjena slike na tom je mjestu najmanja
  - proglašavamo da se blok pomakao duž vektora

Multimedijске arhitekture i sustavi

55

## Algoritmi

- Ovisno o načinu pretraživanja razlikujemo:
  - Algoritam potpunog pretraživanja**
    - pretražuje sve točke unutar područja pretraživanja
    - računski vrlo zahtjevan
    - Daje najbolji mogući rezultat
  - Nepotpuni (brzi, napredni) algoritmi**

Multimedijске arhitekture i sustavi

57

# Algoritam potpunog pretraživanja

- Veoma zahtjevan algoritam za računalne resurse:
  - Izračun jedne vrijednosti poremećaja (MAD):
    - Dohvat podataka (vrlo zahtjevno)
    - Za blok 16x16: 256 oduzimanja + 1 pomak (dijeljenje sa 256)
- Ako područje pomaka iznosi +8 u obje dimenzije:
  - $(2^8+1) * (2^8+1) * (256+\text{dohvat}) = 73984 + \text{dohvati}$
- Pronalaženje minimuma
  - Za jednu sliku 1280x1024
  - $5120 * 73984 = 378.798.080 + \text{dohvati}$
- Za 30 slika u sekundi:  $1,1 * 10^{10}$  zbrajanja/s

# Brzi algoritmi

- **Brzi, napredni algoritmi pretraživanja** usmjeravaju pretraživanje ovisno o vrijednosti poremećaja slike

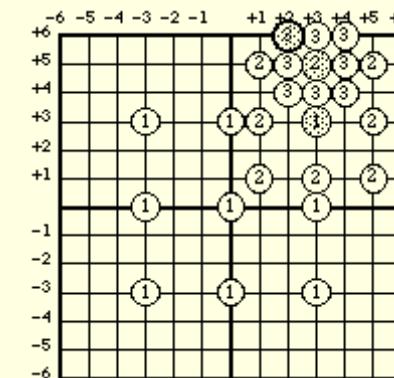
Neki primjeri:

- Logaritamsko pretraživanje (LOG)
- Pretraživanje u tri koraka (3SS)
- Ortogonalno pretraživanje (ORT)
- Algoritam gradijentnog spusta temeljen na blokovima (BBGDS)

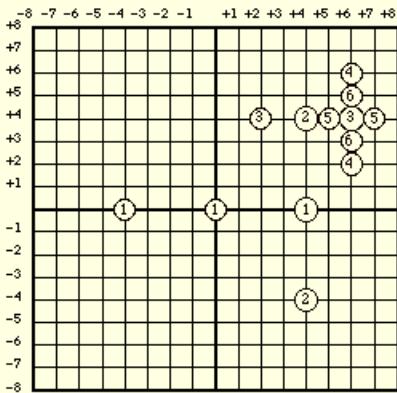
# Algoritam potpunog pretraživanja

- Vidljivo je da algoritam potpunog pretraživanja daje optimalni rezultat ali je nažalost u praksi teško primjenjiv
- Koriste ga uglavnom samo HW koderi
- Upravo zato potreba za brzim algoritmima
  - Prednosti: manje potrebnih operacija
  - Nedostaci: veće razlike koje se trebaju kodirati a time je i izlazna količina podataka veća
  - Kvaliteta nekih algoritama zadovoljavajuća te su blizupotpunom pretraživanju
  - Problemi sa tipovima pokreta (objašnjeno kasnije)

# Algoritmi – primjer (3SS)



## Algoritmi – primjer (ORT)

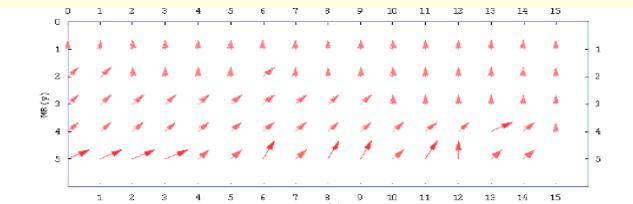


Multimedijiske arhitekture i sustavi

62

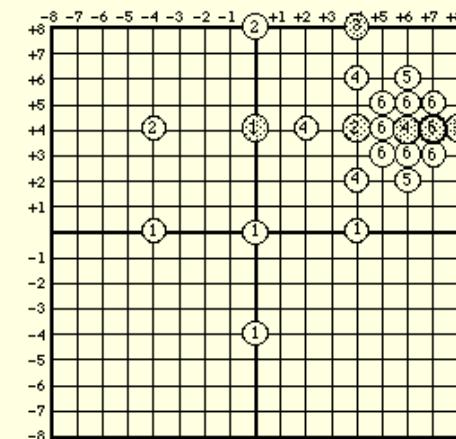
## Raspodjela vektora pomaka

- Prikaz upućuje na ravnomjerno raspoređenepokrete između dvije slike sekvence
- Vektori su većinom kratki – centrirana raspodjela vektora - tipična za sekvence iz stvarnog svijeta



64

## Algoritmi – primjer (LOG)

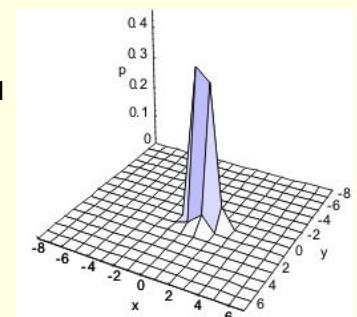


Multimedijiske arhitekture i sustavi

63

## Raspodjela vektora (2)

- Prebrojavanje vektora za sve moguće parove komponenti  $(x,y)$  tijekom cijele sekvene daje relativne frekvencije vektora
- Učinkovitost algoritma moguće je povećati ako se u obzir uzme uočena centrirana raspodjela vjerojatnosti vektora
- središtu područja pretraživanja treba posvetiti više pažnje



Multimedijiske arhitekture i sustavi

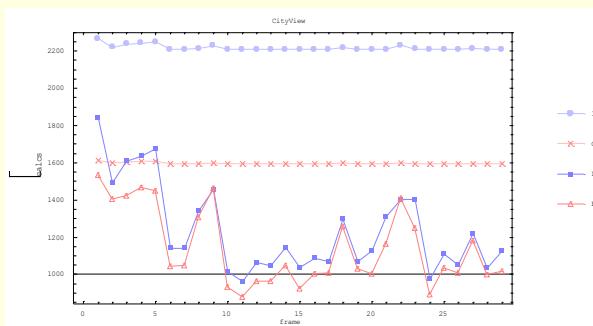
65

# Primjeri usporedbe algoritama

- Parametri koji određuju učinkovitost algoritma:
  - Uкупna **mjera poremećaja** nakon procjene pokreta treba biti što manja (bolja kompresija)
  - Uкупni broj **ispitnih točaka** također treba biti što manji (veća brzina)
- Potreban je kompromis **kompresija ↔ brzina**
- Primjer sekvenci koje sadrže različite vrste pokreta:
  - “*City View*” - ujednačeno raspoređen i malen pokret
  - “*Car Jump*” - lociran i velik pokret
  - “*Troops*” - ujednačeno raspoređen i velik pokret

## Rezultati (1)

- “*City View*” sekvenca – broj ispitnih točaka:



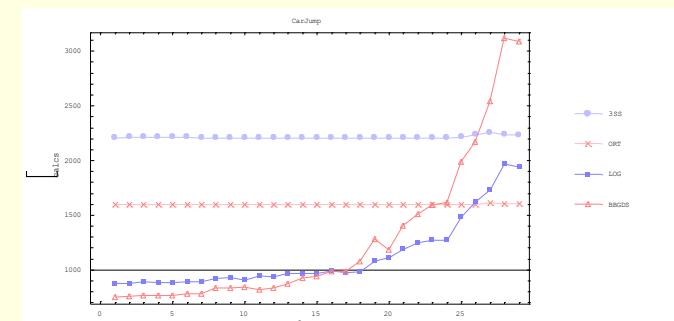
- Centrirana raspodjela: BBGDS najbolji
- ORT i 3SS konstantan i relativno velik broj ispitnih točaka

## Implementacija – primjer

- Norma za kompresiju **ne specificira** postupke procjene pokreta
- Kvaliteta aplikacije znatno ovisi o načinu procjene pokreta

## Rezultati (2)

- “*Car Jump*” sekvenca – broj ispitnih točaka:



- Lokalizirana rastuća promjena: BBGDS i LOG daju slabije rezultate kako poremećaj raste
- ORT bolji od 3SS, konstantni unatoč poremećaju

## Rezultati (3)

- Učinkovitost pojedinog algoritma ovisi o vrsti sadržanog pokreta
- Učinkovit postupak će na temelju vrste pokreta heuristički odabrati najprikladniji algoritam: **adaptivni algoritmi**

## Optimizacije

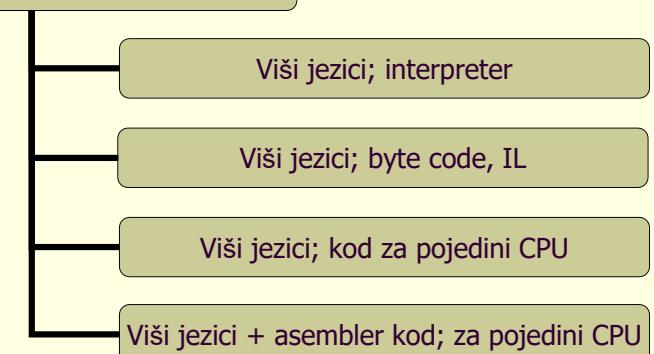
- Neke osnovne grupe optimizacija:
  - Smanjenje preciznosti izračuna
  - Razvoj ekvivalentnih matematičkih algoritama sa manjom kompleksnošću
  - Promjena programske razvojne okoline
  - Promjena programske izvedbene okoline
  - Promjena arhitekture sustava za izvođenje

## Izvedba

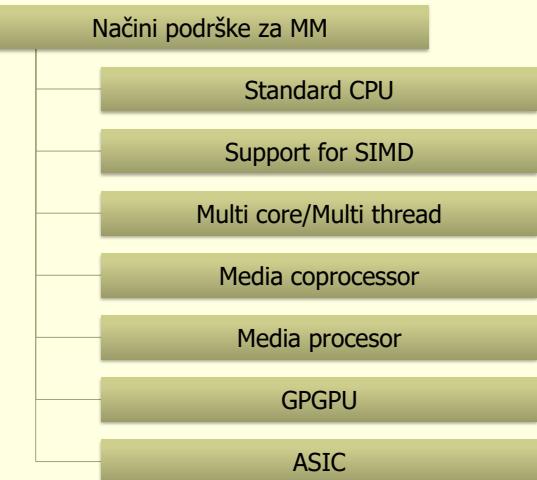
- Ovime smo ukratko analizirali neke najzahtjevnije algoritme pri obradi i kompresiji multimedijiskih podataka
- Vidjeli smo i načelu kompleksnost izračuna bez ulazeњa u previše detalja
- Postavlja se pitanje kako efikasno izvesti takve algoritme

## Osnovni načini SW podrške za MM

### Načini programske podrške za MM



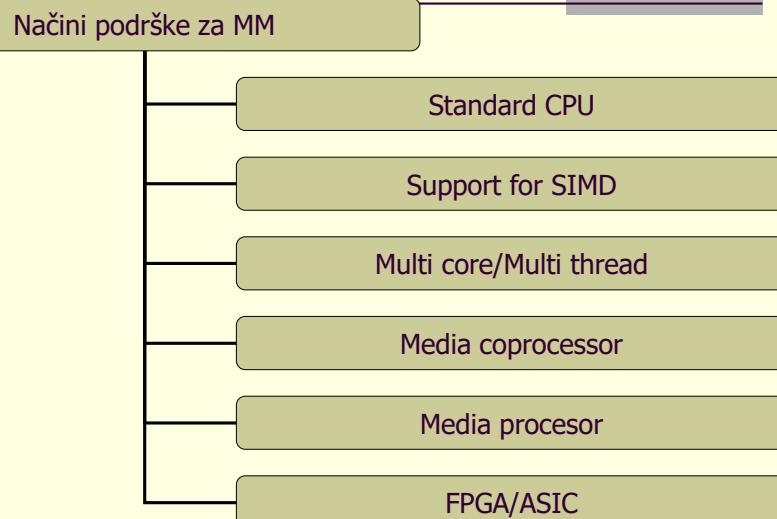
## Osnovni načini HW podrške za MM



74

Multimedijске arhitekture i sustavi

## Osnovni načini CPU podrške za MM



# Multimedijске arhitekture i sustavi (dio 4)

prof.dr.sc. Mario Kovač  
prof.dr.sc. Hrvoje Mlinarić



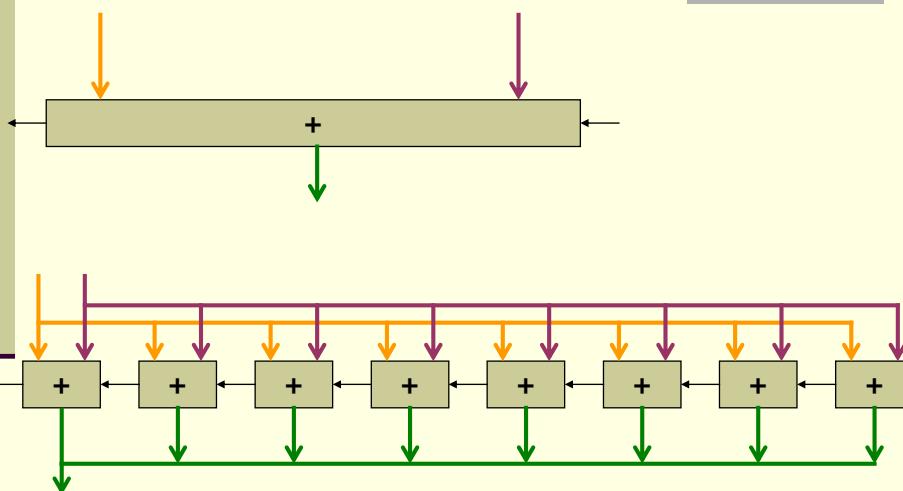
## Standardni CPU

- Ako se prisjetimo Arhitekture računala onda znamo da obični procesori mogu izvesti jednu ALU operaciju po periodu
- Operacija je širine koju ima ALU
- Kako bi ubrzali izvođenje moramo mnogo pažnje posvetiti dohvatu podataka te optimizacijama petlji

## Standardni CPU

- Za DSP operacije (npr DCT) izuzetno je korisno ako procesor ima sklopovski izvedeno množenje i pripadnu naredbu
- Dodatna značajna prednost ako postoji naredba množenja sa zbrajanjem (Multiply Accumulate)
  - Kod primjera računanja DCT, DFT i slično imamo većinu "leptir" operacija kod kojih je MLA osnovna karika

## ALU – obična i SIMD



## Podrška za SIMD

- SIMD (Single Instruction Multiple Data)
  - Arhitektura puta podataka u procesoru koja omogućuje obradu više podataka (u načelu manje preciznosti) sa jednom naredbom
- Osnovna ideja:
  - Ako imamo npr 64 bitovnu ALU onda je potpuno neefikasno s njom obrađivati 8 bitovne podatke
  - Reorganizirati ALU na način da se može "podijeliti"

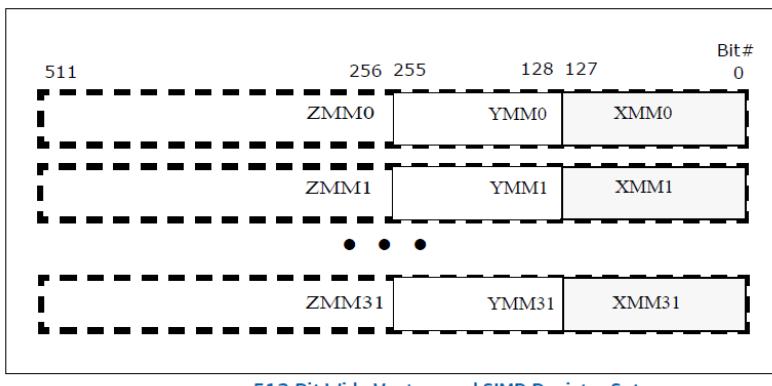
## MM proširenja

- Ovo (osnovno) načelo služi kako bi se obrada multimedijskih algoritama značajno ubrzala
- Primjeri:
  - Prvi: VIS, PA-RISC
  - Stari: MMX, 3DNow!
  - Ne tako stari: SSE4
  - Novi: AVX, AVX2, NEON (ARM)
  - Najnoviji: Intel AVX-512, ARM SVE

# AVX-512

CPUs with AVX-512 [edit]																		
AVX-512 Subset	F	CD	ER	PF	4FMAPS	4VNNIW	VPOPCNTDQ	VL	DQ	BW	IFMA	VBMI	VNNI	VBM12	BITALG	VPCLMULQDQ	GFNI	VAES
Knights Landing (Xeon Phi x200) <sup>[18]</sup>	AVX-512 F, CD, ER, PF								No									
Knights Mill (Xeon Phi x205) <sup>[19]</sup>	AVX-512 F, CD, ER, PF, 4FMAPS, 4VNNIW, VPOPCNTDQ								No									
Skylake-SP, Skylake-X <sup>[19][20]</sup>	AVX-512 F, CD, VL, DG, BW								No									
Cannon Lake <sup>[21]</sup>	AVX-512 F, CD, VL, DQ, BW, IFMA, VBMI								No									
Ice Lake <sup>[21]</sup>	AVX-512 F, CD, VL, DQ, BW, IFMA, VBMI, VBM12, VPOPCNTDQ, BITALG, VNNI, VPCLMULQDQ, GFNI, VAES								No									
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	

# AVX-512



- AVX-512 consists of multiple extensions not all meant to be supported by all processors implementing them. The instruction set consists of the following:
- AVX-512 Foundation – adds several new instructions and expands most 32-bit and 64-bit floating point SSE-SSE4.1 and AVX/AVX2 instructions with EVEX coding scheme to support the 512-bit registers, operation masks, parameter broadcasting, and embedded rounding and exception control
- AVX-512 Conflict Detection Instructions (CDI) – efficient conflict detection to allow more loops to be vectorized, supported by Knights Landing[1]
- AVX-512 Exponential and Reciprocal Instructions (ERI) – exponential and reciprocal operations designed to help implement transcendental operations, supported by Knights Landing[1]
- AVX-512 Prefetch Instructions (PFI) – new prefetch capabilities, supported by Knights Landing[1]
- AVX-512 Vector Length Extensions (VL) – extends most AVX-512 operations to also operate on XMM (128-bit) and YMM (256-bit) registers (including XMM16-XMM31 and YMM16-YMM31 in x86-64 mode)[21]
- AVX-512 Byte and Word Instructions (BW) – extends AVX-512 to cover 8-bit and 16-bit integer operations[21]
- AVX-512 Doubleword and Quadword Instructions (DQ) – enhanced 32-bit and 64-bit integer operations[21]
- AVX-512 Integer Fused Multiply Add (IFMA) - fused multiply add for 52-bit integers.[22]:746
- AVX-512 Vector Byte Manipulation Instructions (VBMI) adds vector byte permutation instructions which are not present in AVX-512BW.
- Only the core extension AVX-512F (AVX-512 Foundation) is required by all implementations; desktop processors will additionally support CDI, VL, and BW/DQ, while computing coprocessors will support CDI, ERI and PFI.

## Primjer: VDBPSADBW

VDBPSADBW—Double Block Packed Sum-Absolute-Differences (SAD) on Unsigned Bytes				
Opcode/ Instruction	Op / En	64/32 bitMode Support	CPUID Feature Flag	Description
EVEX.NDS.128.66.0F3A.W0 42 /r ib VDBPSADBW xmm1 {k1}{z}, xmm2, xmm3/m128, imm8	FVM	V/V	AVX512VL AVX512BW	Compute packed SAD word results of unsigned bytes in dword block from xmm2 with unsigned bytes of dword blocks transformed from xmm3/m128 using the shuffle controls in imm8. Results are written to xmm1 under the writemask k1.
EVEX.NDS.256.66.0F3A.W0 42 /r ib VDBPSADBW ymm1 {k1}{z}, ymm2, ymm3/m256, imm8	FVM	V/V	AVX512VL AVX512BW	Compute packed SAD word results of unsigned bytes in dword block from ymm2 with unsigned bytes of dword blocks transformed from ymm3/m256 using the shuffle controls in imm8. Results are written to ymm1 under the writemask k1.
EVEX.NDS.512.66.0F3A.W0 42 /r ib VDBPSADBW zmm1 {k1}{z}, zmm2, zmm3/m512, imm8	FVM	V/V	AVX512BW	Compute packed SAD word results of unsigned bytes in dword block from zmm2 with unsigned bytes of dword blocks transformed from zmm3/m512 using the shuffle controls in imm8. Results are written to zmm1 under the writemask k1.

### Description

Compute packed SAD (sum of absolute differences) word results of unsigned bytes from two 32-bit dword elements. Packed SAD word results are calculated in multiples of qword superblocks, producing 4 SAD word results in each 64-bit superblock of the destination register.

Within each super block of packed word results, the SAD results from two 32-bit dword elements are calculated as follows:

- The lower two word results are calculated each from the SAD operation between a sliding dword element within a qword superblock from an intermediate vector with a stationary dword element in the corresponding qword superblock of the first source operand. The intermediate vector, see "Tmp1" in Figure 5-18, is constructed from the second source operand and the imm8 byte as shuffle control to select dword elements within a 128-bit lane of the second source operand. The two sliding dword elements in a qword superblock of Tmp1 are located at byte offset 0 and 1 within the superblock, respectively. The stationary dword element in the qword superblock from the first source operand is located at byte offset 0.
- The next two word results are calculated each from the SAD operation between a sliding dword element within a qword superblock from the intermediate vector Tmp1 with a second stationary dword element in the corresponding qword superblock of the first source operand. The two sliding dword elements in a qword superblock of Tmp1 are located at byte offset 2 and 3 within the superblock, respectively. The stationary dword element in the qword superblock from the first source operand is located at byte offset 4.
- The intermediate vector is constructed in 128-bits lanes. Within each 128-bit lane, each dword element of the intermediate vector is selected by a two-bit field within the imm8 byte on the corresponding 128-bits of the second source operand. The imm8 byte serves as dword shuffle control within each 128-bit lanes of the intermediate vector and the second source operand, similarly to PSHUFUD.

The first source operand is a ZMM/YMM/XMM register. The second source operand is a ZMM/YMM/XMM register, or a 512/256/128-bit memory location. The destination operand is conditionally updated based on writemask k1 at 16-bit word granularity.

### Operation

**VDBPSADBW (EVEX encoded versions)**  
 $(KL, VL) = (8, 128), (16, 256), (32, 512)$

Selection of quadruplets:

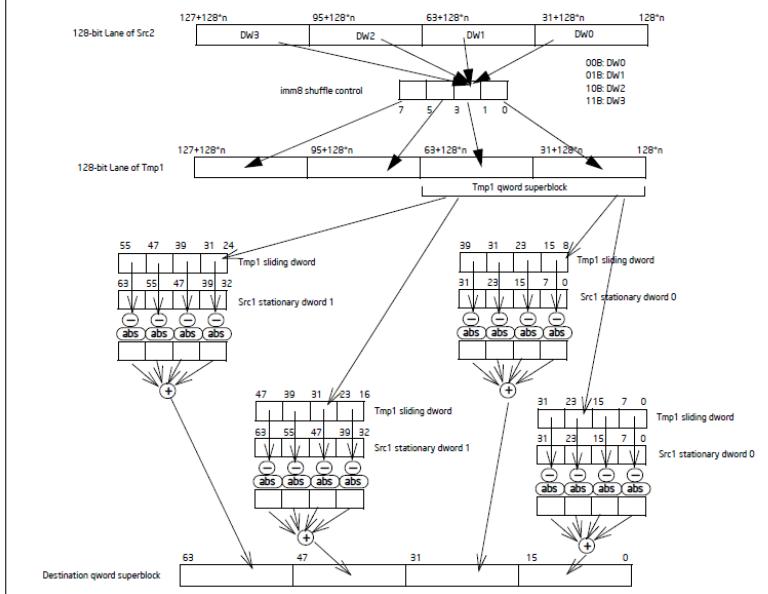
```
FOR I = 0 to VL step 128
    TMP1[I+31:I] ← select (SRC2[I+127:I], imm8[1:0])
    TMP1[I+63:I+32] ← select (SRC2[I+127:I], imm8[3:2])
    TMP1[I+95:I+64] ← select (SRC2[I+127:I], imm8[5:4])
    TMP1[I+127:I+96] ← select (SRC2[I+127:I], imm8[7:6])
END FOR
```

SAD of quadruplets:

```
FOR I=0 to VL step 64
    TMP_DEST[I+15:I] ← ABS(SRC1[I+7:I] - TMP1[I+7:I]) +
        ABS(SRC1[I+15:I] - TMP1[I+15:I]) +
```

```
    TMP_DEST[I+63:I+48] ← ABS(SRC1[I+39:I+32] - TMP1[I+31:I+24]) +
        ABS(SRC1[I+47:I+40] - TMP1[I+39:I+32]) +
        ABS(SRC1[I+55:I+48] - TMP1[I+47:I+40]) +
        ABS(SRC1[I+63:I+56] - TMP1[I+47:I+40])
ENDFOR
```

```
FOR I ← 0 TO KL-1
    i ← j'16
    IF k1[i] OR *no writemask*
        THEN DEST[I+15:i] ← TMP_DEST[I+15:i]
        ELSE
            IF *merging-masking* ; merging-masking
                THEN DEST[I+15:i] remains unchanged*
            ELSE
                ; zeroing-masking
                DEST[I+15:i] ← 0
            FI
    FI;
ENDFOR
DEST[MAX_VL-1:VL] ← 0
```



64-bit Super Block of SAD Operation in VDBPSADBW

### Intel C/C++ Compiler Intrinsic Equivalent

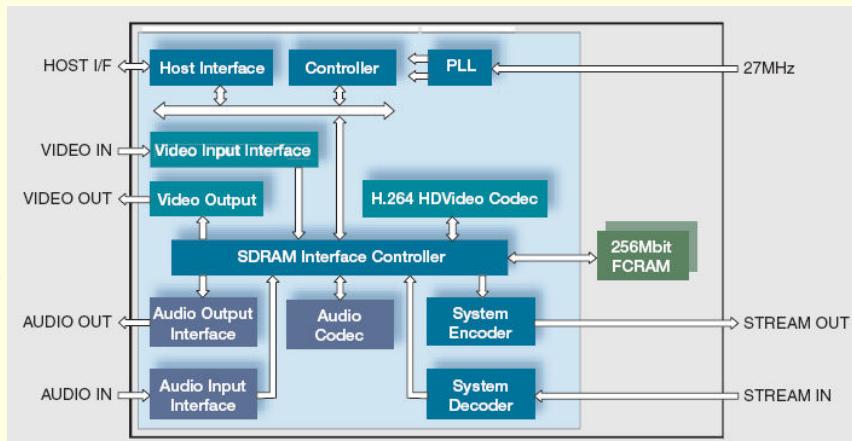
```
VDBPSADBW _m512i_mm512_dbsad_epu8(_m512i a, _m512i b);
VDBPSADBW _m512i_mm512_mask_dbsad_epu8(_m512i s, _mmask32 m, _m512i a, _m512i b);
VDBPSADBW _m512i_mm512_mask2_dbsad_epu8(_m512i s, _mmask32 m, _m512i a, _m512i b);
VDBPSADBW _m256i_mm256_dbsad_epu8(_m256i a, _m256i b);
VDBPSADBW _m256i_mm256_mask_dbsad_epu8(_m256i s, _mmask16 m, _m256i a, _m256i b);
VDBPSADBW _m256i_mm256_mask2_dbsad_epu8(_m256i s, _mmask16 m, _m256i a, _m256i b);
VDBPSADBW _m128i_mm_dbsad_epu8(_m128i a, _m128i b);
VDBPSADBW _m128i_mm_mask_dbsad_epu8(_m128i s, _mmask8 m, _m128i a, _m128i b);
VDBPSADBW _m128i_mm_mask2_dbsad_epu8(_m128i s, _mmask8 m, _m128i a, _m128i b);
```

## Multicore/multithread

- U prethodnim predavanjima:
  - jedan CPU
- Danas: vrlo zahtjevni zadaci (MM je jedan od njih) mogu se na još jedan način ubrzati koristeći više dretvi (thread) ili više jezgri (core)
- Bez ulaženja u teoriju, jasno je da se djelomičnom paralelizacijom procesa može postići veća brzina

## Primjeri: Fujitsu

- MB86H50: H.264 Video Processing IC

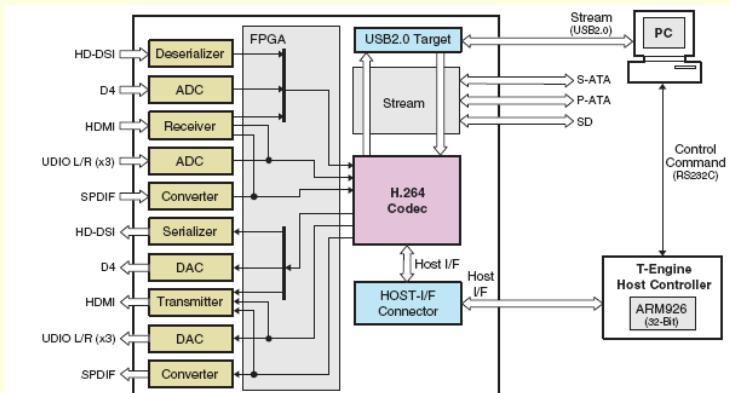


## Što je medijski koprocesor

- Standardni procesori nisu pogodni za mnoge primjene
- Iz dosadašnjeg izlaganja znamo kakva arhitektura je pogodna za visokozahtjevne algoritme i puno podataka
- Moguće rješenje:
  - Zahtjevni dijelovi algoritma obrađuju se u posebnom procesoru koji se stavlja u sustav i služi samo toj funkciji
  - -> Medijski koprocesor

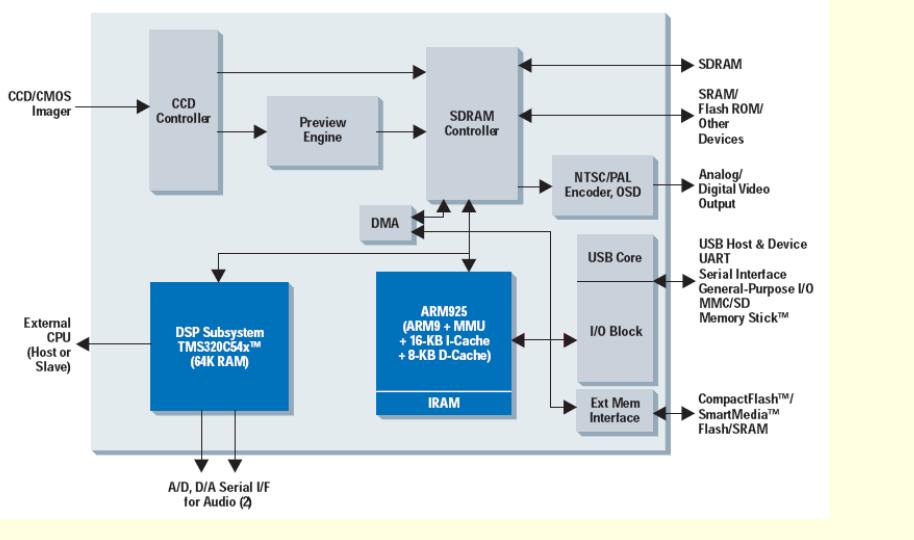
## Fujitsu

- Primjer sustava

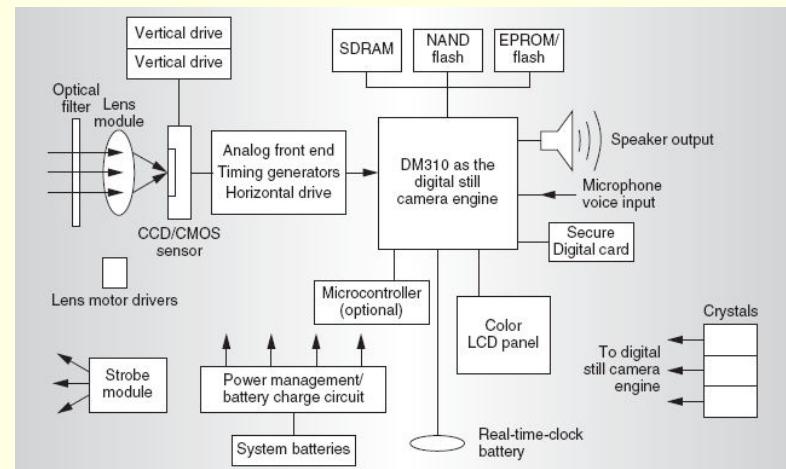


# Primjer: TI

TMS320DM310 Functional Block Diagram

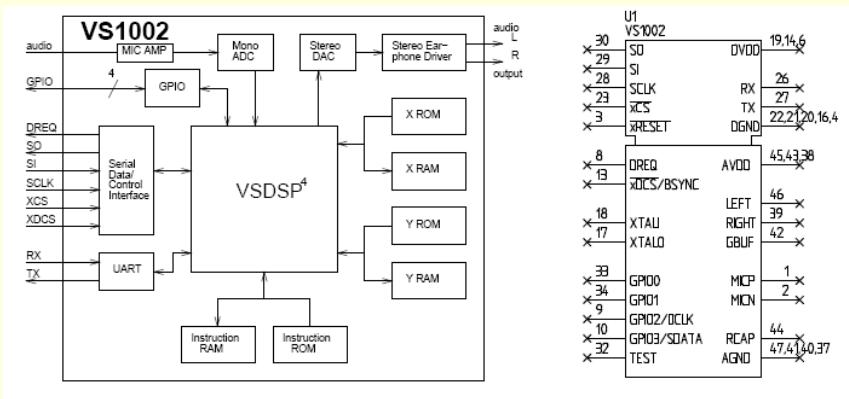


TI

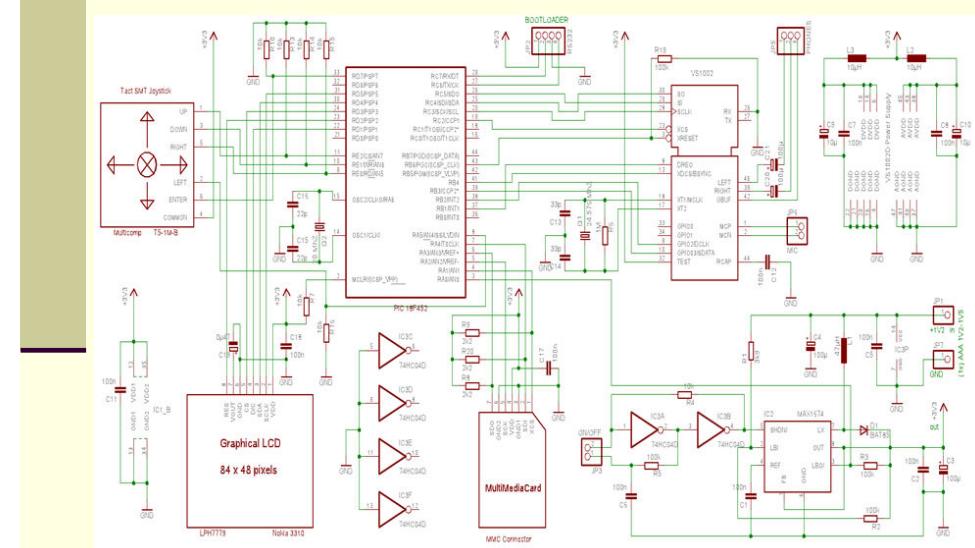


## Medijski koprocесор ne mora biti samo u high-end sustavima

- :VLSI Solution VS 1002: MP3 decoder IC



## Primjer jednostavnog MP3 playera



## VS1002

- Opći procesor:
  - **PIC18LF45x** jednostavan CPU, radi na 20MHz, upravlja radom sustava, upravlja s LCD, tipkama, FLASH karticom,...
  - **VS1002D** MP3 koprocesor, radi samo dekodiranje

## Medijski koprocesor

- U osnovi : DSP sa mnoštvom periferija
- Jeftiniji od procesora opće namjene
- Performanse prilagođene aplikaciji (manje od GPP)
- Manja potrošnja
- Programabilan !! (mogućnost poboljšanja i dodavanja aplikacija)
- Predviđen za porodicu algoritama
- 32-bitovni CPU (npr. ARM) obično dobar za ostale poslove (mreža, user i/f, ...)

## Medijski procesor

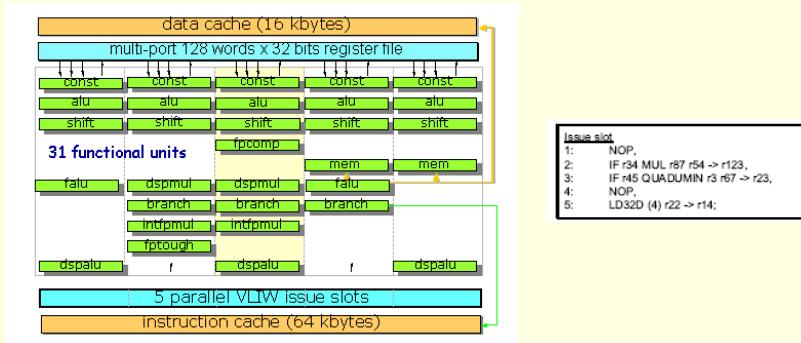
- U dosadašnjim primjerima procesor je bio prvenstveno projektiran za "opće" zadatke
- Za obradu multimedije koristile su se "poboljšanja"
- Novi pristup: procesor se projektira PRVENSTVENO za obradu multimedijskih podataka, a ostale stvari može obrađivati ali tek sporedno
  - -> Medijski procesor

## Medijski procesor

- Procesor se projektira sa ciljem visokih performansi za određen skup algoritama
- No to je još uvijek PROCESOR: može se programirati
  - Prednosti: promjena algoritama, dodavanje funkcionalnosti,....

## Primjer: NXP

### ■ Trimedia TM3270 CPU



## TM

- 31 funkcionalna jedinica
- 5 paralelnih izvedbenih polja
- Very Large Instruction Word (VLIW) arhitektura
- Neke naredbe omogućuju SIMD
- VLIW ima prednosti nad superscalar arhitekturom jer paralelizam ne određuje procesor (scheduling unit on CPU) već programer i compiler tijekom dizajna
  - Procesor jeftiniji i brži

## Medijski procesor

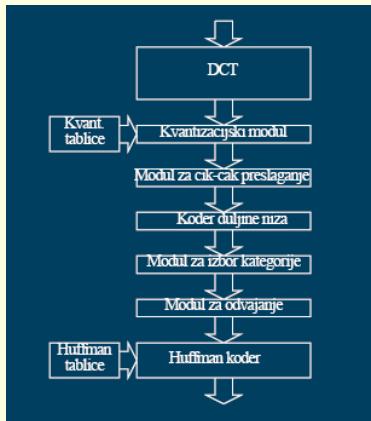
- Izuzetno visoke performanse (veće od GPP, medijskih koprocесora)
- Složen postupak programiranja
- Paralelno izvođenje operacija unutar jedne naredbe
- Još uvijek programabilan
- Podrška za opće zadatke mora biti osigurana od dodatnog procesora

## FPGA/ASIC

- Radi što većih performansi a niže cijene krajnja mogućnost je projektirati sklop koji sklopovski izvodi izabran algoritam
- Dva pristupa
  - FPGA (Field Programmable Gate Array)
  - ASIC (Application Specific IC)

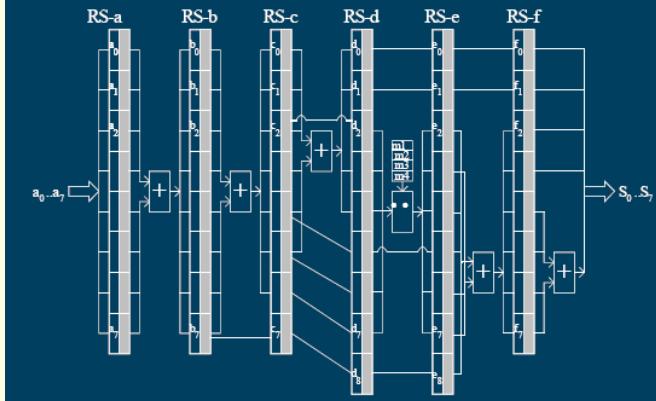
# JAGUAR

## ■ HW JPEG encoder



# JAGUAR

## Sklop za 1D-DCT



# VLSI/ASIC

- Najpogodnije rješenje za zadani algoritam: performanse, potrošnja, cijena,...
- Jednostavni za integraciju i korištenje
- Nemogućnost poboljšanja, nadogradnje
- Samo jedan dobavljač: opasnost

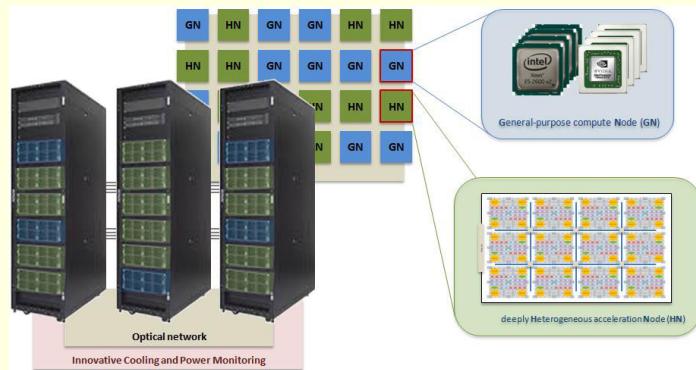
# Heterogeni procesori

- Najnoviji pristup poboljšanju arhitekture
- CPU + GPU
  - **Arm + NVIDIA (2020!?)**
- CPU + FPGA
  - **Intel Broadwell Xeon with a built-in FPGA**
- CPU+GPU+FPGA
- CPU+HW Accelerators



# MANGO Arhitektura

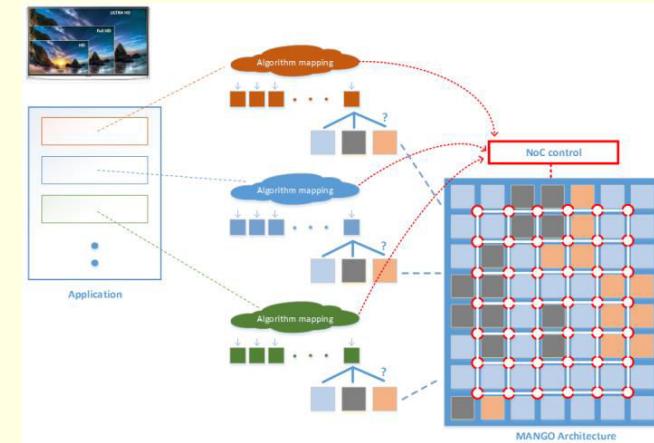
- Deeply heterogeneous QoS aware architecture



Multimedijiske arhitekture i sustavi

36

# Pogled sa strane aplikacija



Multimedijiske arhitekture i sustavi

37

# Multimedijiske arhitekture i sustavi

Prof.dr.sc. Mario Kovač

Prof.dr.sc. Hrvoje Mlinarić



Ova predavanja koriste jedan dio materijal koji je ustupljen od strane XILINX pod XUP uvjetima korištenja.

Multimedijiske arhitekture i sustavi

1

# Algoritmi i njihova izvedba u rač. sustavima

- Do sada smo vidjeli na primjeru JPEG-a kako izvesti neki algoritam
- Jednostavno uzmemu standard i raspišemo ga u nekom višem programskom jeziku
- Stvar će naravno raditi
  - Brzina je upitna !!!

Multimedijiske arhitekture i sustavi

2

# Algoritmi i njihova izvedba u rač. sustavima

## ■ Sljedeći korak: provjeriti usko grlo algoritma:

- Kod JPEG-a
  - DCT/IDCT
  - kvantizacija
  - RGB2YUV i YUV2RGB
  - GetBits (vrlo jednostavna funkcija koja se jako puno puta poziva)

# Kako riješiti problem

## ■ Prvi način programski:

- DCT/IDCT: da li postoji neki drugi pristup od standardnog

$$X_{k_1, k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1, n_2} \cos\left[\frac{\pi}{N_1}\left(n_1 + \frac{1}{2}\right)k_1\right] \cos\left[\frac{\pi}{N_2}\left(n_2 + \frac{1}{2}\right)k_2\right].$$

- Pojednostavljeni algoritam putem FFT ili neki drugi pristup. AAN Algoritam
- Treba nam pomoć matematike.
- Problem znanja i primjene

# Kako riješiti problem

## ■ RGB to YUV Conversion

$$\begin{aligned} Y &= (0.257 * R) + (0.504 * G) + (0.098 * B) + 16 \\ V &= (0.439 * R) - (0.368 * G) - (0.071 * B) + 128 \\ U &= -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128 \end{aligned}$$

## ■ YUV to RGB Conversion

$$\begin{aligned} B &= 1.164(Y - 16) + 2.018(U - 128) \\ G &= 1.164(Y - 16) - 0.813(V - 128) - 0.391(U - 128) \\ R &= 1.164(Y - 16) + 1.596(V - 128) \end{aligned}$$

## ■ RGB to YUV cjelobrojni

$$\begin{aligned} y &= (66 * r + 129 * g + 25 * b + 128) \gg 8 + 16; \\ u &= (-38 * r - 74 * g + 112 * b + 128) \gg 8 + 128; \\ v &= (112 * r - 94 * g - 18 * b + 128) \gg 8 + 128; \end{aligned}$$

## ■ RGB to YUV cjelobrojni pojednostavljeni

$$\begin{aligned} y &= (66 * r + 4 * 32 * g + 25 * b + 128) \gg 8 + 16; \\ u &= (-38 * r - 74 * g + 112 * b + 128) \gg 8 + 128; \\ v &= (3 * 38 * r - 3 * 32 * g - 18 * b + 128) \gg 8 + 128; \end{aligned}$$

## GetBits

- Problem dohvata bitova iz memorije
- Nije zahtjevan potprogram ali se često poziva i zato može izazvati probleme.
- Nije efikasno dohvaćati bit po bit na 32-bitnom procesoru
- Problem protočne arhitekture
- Problem pričuvne memorije

## Kako riješiti problem

- Ne postoji metodologija kako to jednostavnije ubrzati algoritam
- Što onda, kako ubrzati algoritam?
  - Programske
    - Aproksimacija, pojednostavljenje
    - Proučiti arhitekturu procesora
      - Napisati pretvorbu u strojnom jeziku
      - Možda postoje posebne naredbe u strojnom jeziku koje mogu ubrzati algoritam
        - HITACHI SH4 (množenje matrice 4x4 s vektorom) ('97)
        - SIMD, ARM NEON, 3D naredbe
  - Sklopovsko rješenje

## Kako riješiti problem?

- Aproksimacija funkcija polinomom n-tog reda...
- Prevodenje algoritama iz realne u cijelobrojnu domenu (fix-point)
- Dodavanje posebnog sklopolja
  - ARM - XSCALE(200MHz) procesor prespor za dekodiranje MPEG4 video zapisa (320x240). Naknadno dodan dodatni sklop Maratton za pomoć u dekodiranju Video zapisa.

## Zaključak

- Pisanje efikasnog koda zahtjeva poznavanje i programske i sklopovske podrške. Moramo poznavati:
  - Kako radi prevodioci
  - Kako se koristi strojni kod
  - Arhitekturu sklopolja periferija
  - Arhitekturu procesora

## Multimedejske arhitekture i sustavi

- Kako ubrzati rad algoritma korištenjem sklopovlja:
  - Korištenjem postojećih sklopova
  - Izrada vlastitih
  - Sklopovska integracija SoC
    - Hardware software codesign

## Hardware Software codesign

- Ugradbeni sustavi imaju sljedeće karakteristike:
  - Uglavnom imaju jednu funkcionalnost
    - Koja je unaprijed definirana
  - Zadovoljavaju
    - Projektirani da se smanji potrošnja
      - Što manji broj komponenti
    - Moraju zadovoljiti brzinom rada
  - Rad u stvarnom vremenu (Real-time)
    - Mora kontinuirano pratiti događaje i reagirati na promjene
  - Sklopovska programska isprepletenost

## Hardware Software codesign

- Problematika izvedbe cijelokupnog rješenja
- Pogodnost izrade sustava u jednom okruženju
- Jednostavniji dizajn
- Naročito pogodan u ugradbenim sustavima

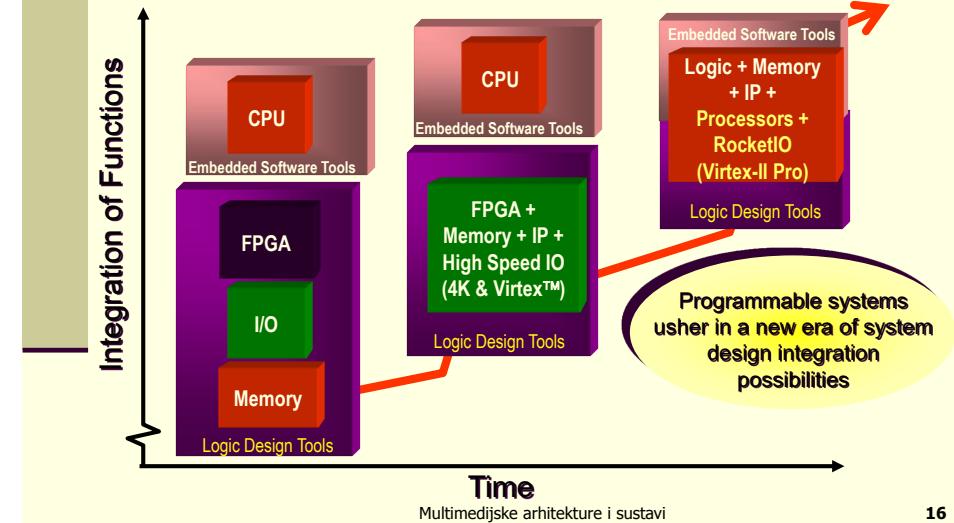
## Ugradbeni računalni sustavi

- Primjeri:
  - Mobilni telefoni
  - Auto aplikacije
    - Sustav kočnica, kontrola proklizavanja, zračni jastuci, i.t.d.
  - Avionska industrija
    - Sustavi kontrole leta, kontrola motora, sustav za samostalno letenje, sustavi za zabavu putnika
  - Obrambeni sustavi
    - Radarski sustavi, sustav za kontrolu borbenih avion, sustavi avion, ...

# Postojeća tehnologija

- Mikrokontroler zasnovani sustavi
- DSP procesor zasnovani sustavi
- ASIC tehnologija
- FPGA tehnologija

# Integration in System Design



# Što je u planu za ovaj dio semestra?

- Dobiti neki osnovni uvid u problematiku i metode riješavanja H/S codesigna.
- S obzirom da već od prije poznajete programabilnu logiku, a u prvom dijelu predavanja upoznali ste se s JPEG standardom. Ovaj dio će se pozabaviti problemom izrade sustava digitalnog foto aparata

# Ugradbeni sustavi korištenjem FPGA

- Osnove smo vidjeli na URS-u
- Sastoje se od:
  - FPGA sklopovske arhitekture
  - Generiranje upravljačkih programa i biblioteka
  - Programske aplikacije
    - Potprograma
    - Prekidnog sustava
    - Operating System (OS) ili Real Time Operating System (RTOS) (optional)

## Primjer 22 Mpixel digitalni aparat

- Phase One H25
- Digitalna kamera
- 2004 godina
- 22 MegaPixel
- Cijena:
  - \$29,990



Multimedijiske arhitekture i sustavi

19

## Phase One

- Danas:
  - CCD Full frame
  - Rezolucija: 80 mega pixels
  - CCD size effective 49.1 x 36.8 mm
  - Pixel size 6.8 x 6.8 micron
  - Image ratio 4:3
  - Cijena: \$49,990



Multimedijiske arhitekture i sustavi

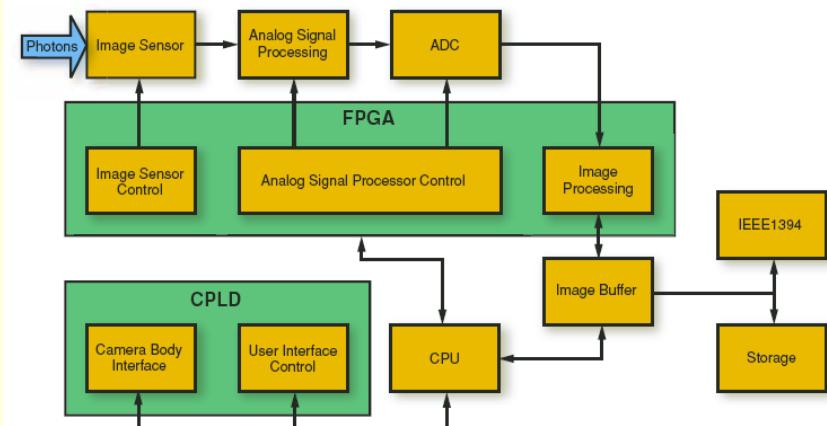
21

- Velika veličina datoteke: 128 MB/slici
- od 16 bits po boji, ukupno 48 bits za red/green/blue (RGB)
- Brzina prijenosa podataka 7 Gbps
- Velika memorija za dugačke burst sequences
- Pohrana nekomprimiranih ili komprimiranih slika
- 30 slike/min u punoj rezoluciji
- 400 Mb IEEE1394 (Firewire)
- Napredni "power management"
- Male dimenzije (relativno gledano)

Multimedijiske arhitekture i sustavi

20

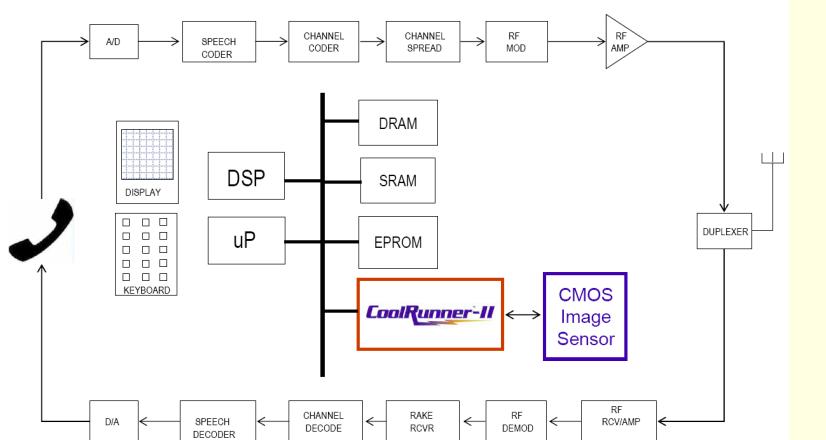
## Phase One



Multimedijiske arhitekture i sustavi

22

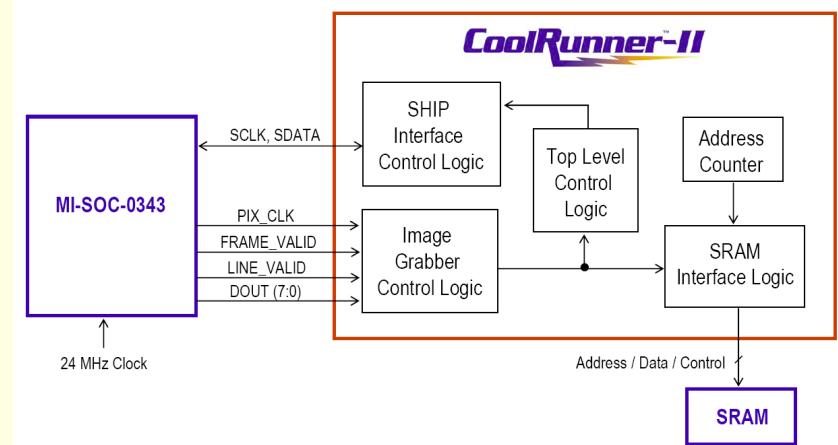
## Mobilni telefoni - primjer



Multimedijiske arhitekture i sustavi

23

## CoolRunner-II



Multimedijiske arhitekture i sustavi

24

## Prototyping

- “A prototype is an early sample or model built to test a concept or process or to act as a thing to be replicated or learned from.”
- Male serije
- Mikroprocesori protiv FPGA

## Mikroprocesori

- Izvršavaju prevedeni kod napisan u asembleri i/ili višem programskom jeziku
  - Program se nalazi na samom mikroprocesoru i/ili vanjskoj memoriji
  - Procesor dohvata naredbe, dekodira ih, obrađuje podatke i kontrolira I/O
    - Slijedno izvođenje
    - Zahtijeva velik broj taktova za obavljanje operacija
- Lagana optimizacija potrošnje (Power save mode)
  - Procesor proradi samo na određene događaje
- Ograničena I/O sučelja

## FPGA

- 'Sea of gates' moguće konfigurirati prema potrebi
- Pogodni za paralelno izvođenje operacija (mreža, multimedija,...)
- Velik broj I/O pinova i podržanih standarda
- Jednostavna podjela u funkcijeske blokove
- Nisu pogodni za tokovne aplikacije i kontrolu

## Mikroporcesor naspram FPGA

- Zašto ne bi koristili najbolje od jedno i drugoga.
- Povežimo ih zajedno u jednu cjelinu.

## FPGA embedded processor

- FPGA računalni sustav ima mnoge prednosti u odnosu na standardni računalni sustav:
  - 1) modularnost i nadogradnja
  - 2) jednostavna migracija
  - 3) smanjena cijena i potrošnja (redundantnost)
  - 4) sklopovsko ubrzanje

## Modularnost i Nadogradnja

- Potpuna fleksibilnost u izradi računalnog sustava.
- Dizajn može zahtijevati potpuno novu nepostojeće sučelje koje je vrlo jednostavno dodat
  - Na primjer ne možete naći procesor sa 10 UART sučelja, ali svaki FPGA možete prilagoditi bez ikakvih problema da ima 10 UART potrova.

## Migracija

- Mnoge kompanije su razvile cijeli sustav kojem je životni vijek puno veći nego, životni vijek pojedinih komponenti
- FPGA soft-procesori su izvanredan odabir jer su opisani pomoću HDL jezika i lako ih je prebaciti i na nove platforme
- Npr. Končar Elektronik

## Smanjena cijena i potrošnja

- Sklop koji je prije zahtjevalo više komponenti danas se može zamijeniti s jednim FPGA sklopolom
- Smanjivanjem broja komponenti, možemo smanjiti veličinu proizvoda i potrebnih komponenti
- Sve to dovodi do smanjenja cijene i potrošnje.

## Sklopovsko ubrzanje

- Perhaps the most compelling reason to choose an FPGA embedded processor is the ability to make tradeoffs between hardware and software to maximize efficiency and performance.
- If an algorithm is identified as a software bottleneck, a custom co-processing engine can be designed in the FPGA specifically for that algorithm.
  - This co-processor can be attached to the FPGA embedded processor through special, low-latency channels, and custom instructions can be defined to exercise the co-processor.
- With modern FPGA hardware design tools, transitioning software bottlenecks from software to hardware is much easier since the software C code can be readily adapted into hardware with only minor changes to the C code.

## Nedostatci

- Za razliku od off-the-shelf procesora, sklopovska arhitektura treba biti opisana za FPGA i testirana.
- Zbog povezanosti sklopovlja i programske podrške alati su složeniji.
- Kako je područje relativno novo alati imaju neke dječje bolesti ☺.
- Cijena samih komponenti.
  - Cijena off-the-shelf višestruko je jeftinija od FPGA sklopova.

## Soft – Hard procesor

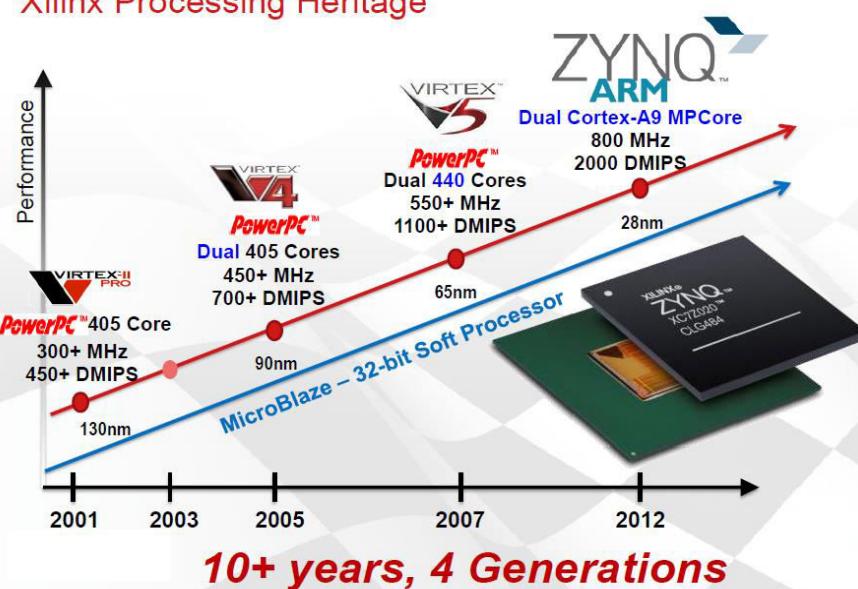
- Soft procesor – koristi FPGA blokove
- Hard procesori – Izveden u siliciju
  - ARM922T -Altera Excalibur
  - PowerPC 405 - Xilinx Virtex-II Pro and, Virtex-4.
- Za razliku od hard procesora, soft procesor mora biti sintetiziran i implementiran u FPGA sklopu.

## FPGA embedded processor

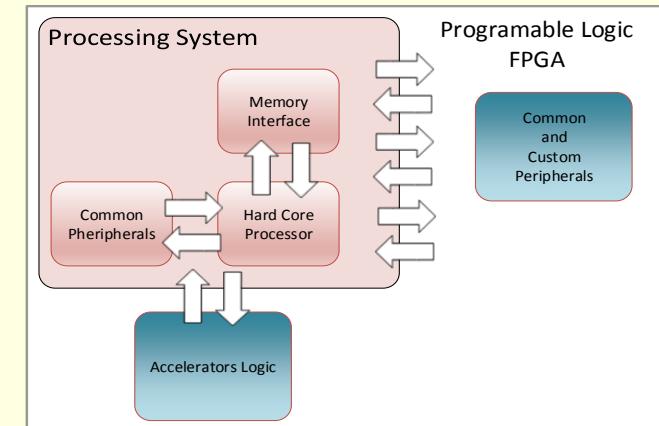
### FPGA

- Soft core processor
  - PicoBlaze, MicroBlaze (Xilinx)
  - Nios II (Altera)
  - LEON3, 8051, C68000, ...
  - ARM Cortex M1
- Hardcore proceor
  - PowerPC 405

### Xilinx Processing Heritage



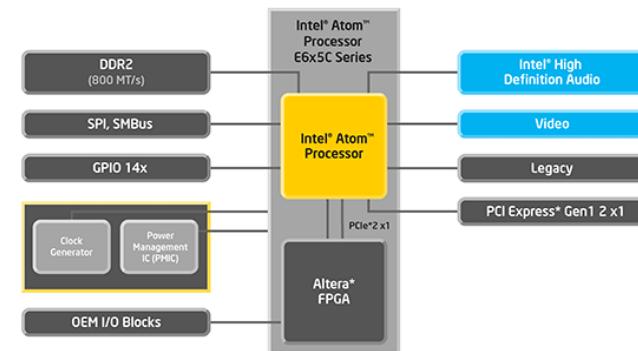
## Što imamo danas?



# What We Have Today ?

- Xilinx
  - Virtex II Pro, Virtex 4, Virtex5 (PowerPC 405)
  - ZYNQ 7000
- Altera
  - Excalibur (ARM922T)
  - Cyclone V SoC, Arria V SoC
- Actel
  - SmartFusion
  - SmartFusion 2

# Intel Atom Processor E6x5C Series

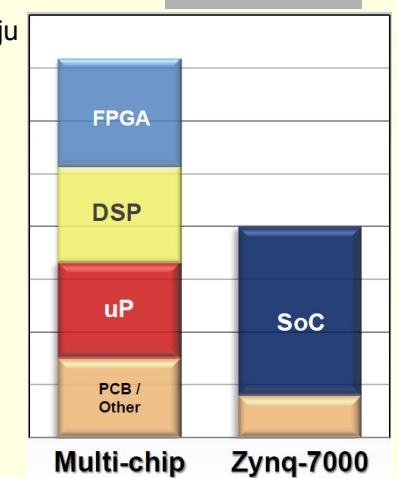


## Xilinx Zynq 7000 family The All Programmable SoC

- Nije običan FPGA!
- Nije običan mikroprocesor!
- Jedinstvena simbioza i jednog i drugog
  - Dual ARM Cortex™-A9 procesor+ caches + robusni sustav periferija
  - Velika povezanost između procesora i logike
- To više nije FPGA ili procesor to je procesorski sustav s programibilnom logikom – “All Programmable SoC”

## Smanjenje troška

- Smanjen broj komponenti po uređaju
  - Procesor
  - Programabilni sklop
  - DSP
  - Napajanje
- Smanjena PCB složenost
  - Manje vodova => manje slojeva
  - Brži dizajn
- Jako bitno
  - In-System Reconfiguration



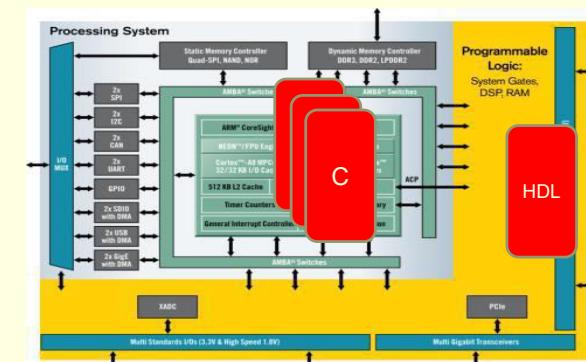
# Povećanje performansi

- Meet HW and SW Processing Performance Needs
  - Programabilna logika
  - Ogromna DSP snaga
  - Sabirnice velike propusnosti –on chip
    - Over 3000 Processing System to Programmable Logic direct connections
  - High performance I/Os
  - Gigabit transceivers

Elements	Performance (up to)
Processors (each)	1 GHz
PL Fabric/ DSP Fmax	741 MHz
DSP (aggregate)	1080 GMACs
Transceivers (each)	12.5Gbps

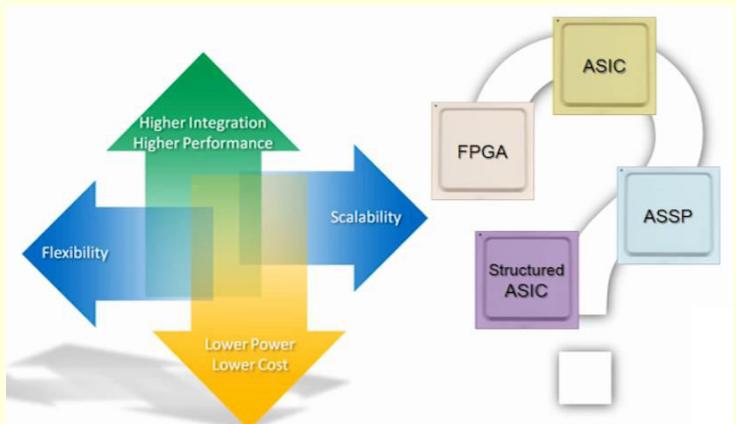
# Povećanje performansi

- Optimized & Simplified HW/SW Partitioning



# Današnji zahtjevi

- Koju tehnologiju izabrati?



# Što odabrat?

	ASIC	ASSP
Performance	+	+
Power	+	+
Unit Cost	+	+
TCO	■	+
Risk	-	+
TTM	-	+
Flexibility	■	-
Scalability	-	■

# Što odabrat?

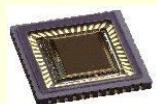
	ASIC	ASSP	2 Chip Solution
Performance	+	+	-
Power	+	+	-
Unit Cost	+	+	-
TCO	-	+	+
Risk	-	+	+
TTM	-	+	+
Flexibility	-	-	+
Scalability	-	-	+

# Što odabrat?

	ASIC	ASSP	2 Chip Solution	FPGA SoC
Performance	+	+	-	+
Power	+	+	-	+
Unit Cost	+	+	-	+
TCO	-	+	+	+
Risk	-	+	+	+
TTM	-	+	+	+
Flexibility	-	-	+	+
Scalability	-	-	+	+

## Naš sustav

- Xilinx Zynq SoC FPGA
- SPARTAN-6 E2LP
- OmniVision OV7670 senzor



## Komponente sustava

- Zynq SoC FPGA
  - ZedBoard - »Zynq™-7000 SoC XC7Z020
    - Dual core ARM Cortex A9
- Spartan-6 E2LP
  - MicroBlaze procesor
- Digitalni video senzor OmniVision OV7670
  - I2C (SCCD)
  - Zoom Video Port – Digital video port

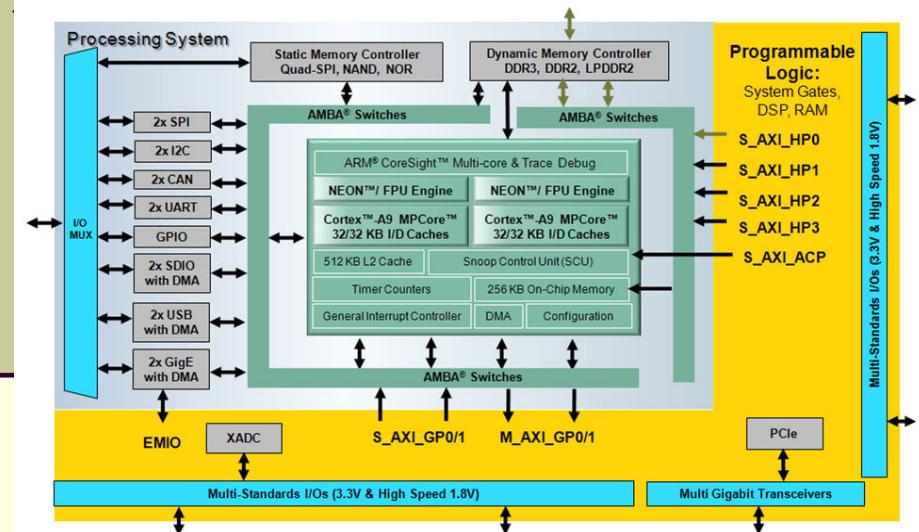
# Zynq-7000 Family Highlights

- ARM procesorski sustav
  - Application Processor Unit (APU)
  - Dual ARM Cortex™-A9
  - Pričuvna memorija
  - Fully integrated memory controllers
  - I/O sučelja
- Povezano s programabilnom logikom
  - Koristi se zaproširivanje funkcionalnosti procesora
  - Skalabilnost i povećanje performansi
- Fleksibilno I/O sučelje
  - Veliki raspon I/O standarda
  - Serijski prijenos visokih performansi
  - Analogno digitalni pretvornici

Multimedijiske arhitekture i sustavi

51

# ZYNQ 7000



# ZYNQ 7000

- The Zynq-7000 AP SoC arhitektura sastoji se od dva dijela
  - PS: Procesorskog sustava
    - Dual ARM Cortex-A9 procesor
    - Mnogobrojne periferije
  - PL: Programabilne logike
    - Dijele arhitekturu najnovijih Xilinx FPGA sklopova
      - Artix™-zasnovani uređaji: Z-7010 i Z-7020
      - Kintex™-zasnovani uređaji: Z-7030, Z-7045, and Z-7100

Multimedijiske arhitekture i sustavi

53

# ZYNQ – dual core procesor

- ARM Cortex-A9 izveden u ARMv7-A arhitekturi
  - ARMv7 definira ARM Instrukcijski set (ISA)
  - ARMv7-A: oznaka A znači da podržava - Memory Management Unit (MMU)
- ARMv7 ISA osim osnovnog seta naredbi podržava sljedeće naredbe
  - Thumb naredbe: 16 bits; Thumb-2 naredbe: 32 bits
  - NEON: ARM's Single Instruction Multiple Data (SIMD)

Multimedijiske arhitekture i sustavi

54

## ZYNQ – dual core procesor

- ARM Advanced Microcontroller Bus Architecture (AMBA®) sabirnica
  - AXI3: ARM sučelje treće generacije
  - AXI4: nadogradnja AXI3 sabirnice
- Cortex je najnovija porodica procesora ARM

## ZYNQ – dual core procesor

- 2.5 DMIPS (Dhrystone)
- Harvardska arhitektura
- 32KB L1 instrukcijske i podatkovne pričuvne memorije
- 512KB L2 pričuvne memorije
- maksimalna frekvencija 1GHz

- Application processing unit (APU)
- I/O sučelja (IOP)
  - Multiplexed I/O (MIO), Extended Multiplexed I/O (EMIO)
- Memorjsko sučelje
- DMA
- Timers
  - Public and private
- General interrupt controller (GIC)
- On-chip memory (OCM): RAM
- Debug controller: CoreSight

## ZYNQ Procesor - memorija

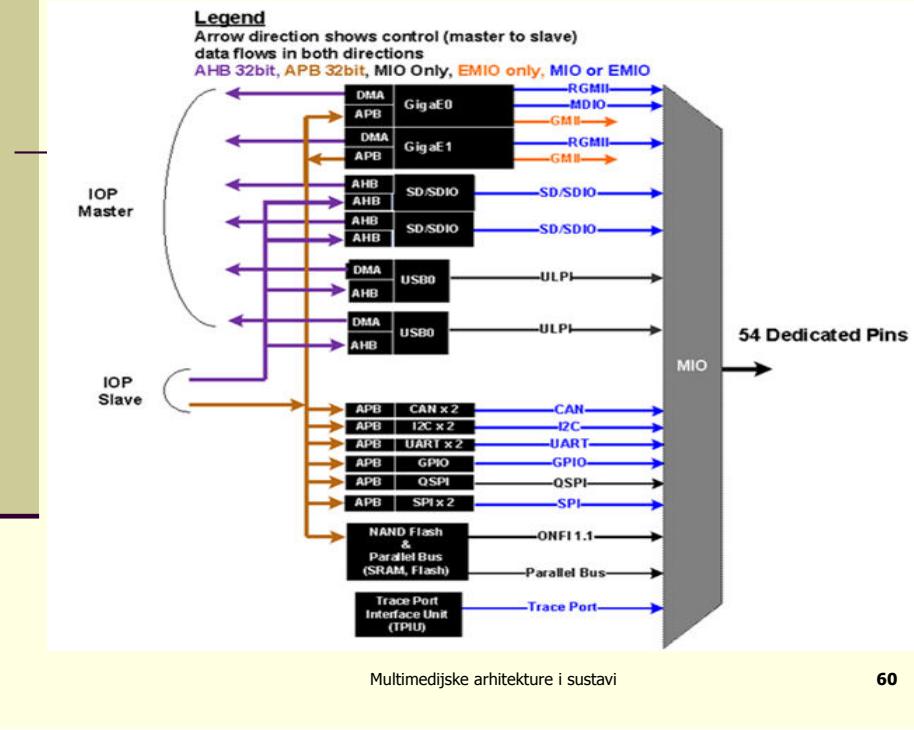
- On-chip memorija(OCM)
  - RAM
  - Boot ROM
- DDRx sučelje
  - podržava LPDDR2, DDR2, DDR3
- Flash/static, memorjsko sučelje
  - Podržava SRAM, QSPI, NAND/NOR FLASH

# ZYNQ – Procesor – IO sučelja

- Dva GigE
- Dva USB
- Dva SPI
- Dva SD/SDIO
- Dva CAN
- Dva I2C
- Dva UART
- Četiri 32-bit GPIOs
- Statičke memorije
  - NAND, NOR/SRAM, Quad SPI

Multimedijiske arhitekture i sustavi

59



Multimedijiske arhitekture i sustavi

60

# PS – PL Sučelje

- AXI high-performance slave ports (HP0-HP3)
  - 32-bit i 64-bit
  - pristup OCM i DDR
  - AXI FIFO sučelje (AFI)
- AXI general-purpose ports (GP0-GP1)
  - Dva “master” PS - PL
  - Dva “slave” PL - PS
  - 32-bit širine

Multimedijiske arhitekture i sustavi

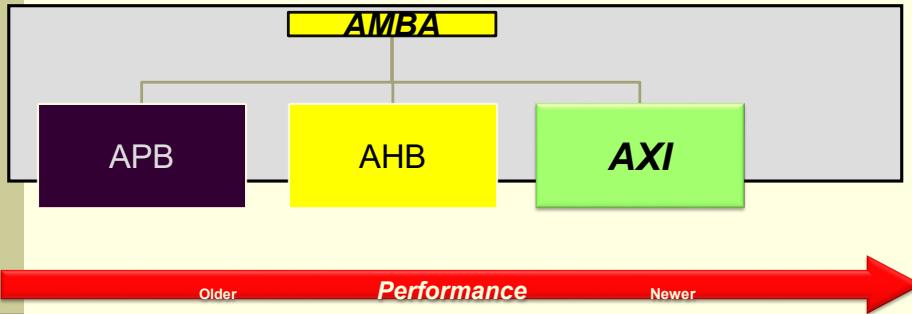
61

- 64-bit accelerator coherence port (ACP) AXI slave interface to CPU memory
- DMA, interrupts, events signals
- Extended multiplexed I/O (EMIO) – omogućava spajanje procesorskih periferija s PL logikom
- Clock i reset
  - Četiri odvojena signala vremenskog vođenja
  - Četiri odvojena reset signala

Multimedijiske arhitekture i sustavi

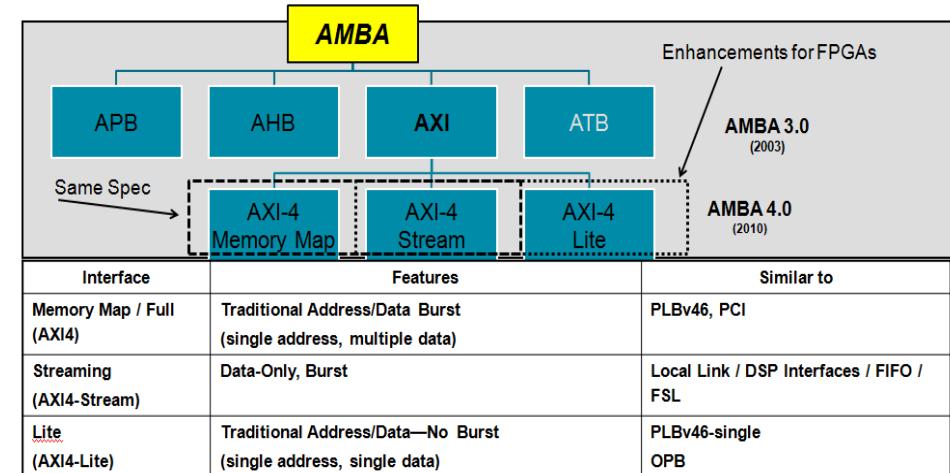
62

# AXI



AMBA: Advanced Microcontroller Bus Architecture  
AXI: Advanced Extensible Interface

# AXI



# Programibilna logika

- Artix-7 FPGA
- 87K logičkih blokova, oko 1.3M logičkih vrata
- 53.200 logičkih vrata
- 106.400 registara (bistabila)
- 560 KB BRAM memorija
- 220 MACC blokova
- 276 GMAC operacija u sekundi
- ADC – 2 x 12 bita – 17 ulaza

# Spartan 6

- Spartan-6:
  - Spartan-6 LX FPGA: velik broj logičkih vrata
  - Spartan-6 LXT FPGA: komunikacijski zahtjevni sklopovi
- Namijenjeni za izradu jeftinih ugradbenih sustava
  - Velik broj različiti blokovi koji imaju različite funkcionalnosti
  - Velik broj IO podržanih normi
  - Jeftina izvedba pakiranja

# Spartan 6

- Različite naponske razine, podrška za različite standarde - SelectIO™
  - Do 1,080 Mb/s podataka
  - Mogućnost konfiguracije izlazne struje, do 24 mA po pinu
  - 3.3V do 1.2V I/O
  - "Hot swap" podrška
- High-speed GTP serijska komunikacija - LXT obitelj
  - do to 3.2 Gb/s
  - High-speed podrška za: Serial ATA, Aurora, 1G Ethernet, PCI Express, OBSAI, CPRI, EPON, GPON, DisplayPort,

# Spartan 6

- CLB
  - Sadrže dvije polovice (Slices)
- Slices
  - Postoje tri tipa
    - SLICEM
    - SLICEL
    - SLICEX
- Upravljanje signalom vremenskog vođenja
  - DCM
  - PLL
- Blok RAM
- Množila, IO blokovi, memorijski sklopovi

# Spatan 6

- DSP48A1 Blokovi
  - Sklopovi pogodni za aritmetičke operacije i digitalnu obradu signala
  - 18 x 18 Množenje i 48-bit MAC operacije
  - Pogodni za protočni ili kaskadni
  - Dodatno pred zbrajalo
- Memory Controller blokovi
  - DDR, DDR2, DDR3 i LPDDR podrška
  - Brzine do 800 Mb/s
  - "Multi-port" sabirnička struktura sa neovisnim FIFO spremnicima

## Spartan-6 Family FPGAs



Spartan-6 LX FPGAs Optimized for Lowest Cost Logic, DSP, and Memory (1.2 Volt, 1.0 Volt)											Spartan-6 LXT FPGAs Optimized for Low Cost Logic, DSP, and Memory with High Speed Serial Connectivity (1.2 Volt)				
	Part Number	XC6SLX4	XC6SLX9	XC6SLX16	XC6SLX25	XC6SLX45	XC6SLX75	XC6SLX100	XC6SLX150	XC6SLX25T	XC6SLX45T	XC6SLX75T	XC6SLX100T	XC6SLX150T	
Logic Resources	Slices <sup>(1)</sup>	600	1,430	2,278	3,768	6,822	11,662	15,822	23,038	3,758	6,822	11,662	15,822	23,038	
	Logic Cells <sup>(2)</sup>	3,840	9,152	14,579	24,051	43,681	74,837	101,261	147,443	24,051	43,681	74,837	101,261	147,443	
	CLB Flip-Flops	4,800	11,440	18,224	30,064	54,576	93,296	125,576	184,304	30,064	54,576	93,296	125,576	184,304	
Memory Resources	Maximum Distributed RAM (Kbits)	75	90	138	229	401	692	976	1,355	229	401	692	976	1,355	
	Block RAM (16K bits each)	12	32	32	52	116	172	268	268	52	116	172	268	268	
	Total Block RAM (Kbits) <sup>(3)</sup>	218	576	576	936	2,088	3,096	4,824	4,824	936	2,088	3,096	4,824	4,824	
Clock Resources	Clock Manager Tiles (CMT) <sup>(4)</sup>	2	2	2	2	4	6	6	6	2	4	6	6	6	
	Maximum Single-Ended Pins	120	200	232	266	358	400	480	570	250	296	320	490	530	
I/O Resources	Maximum Differential Pairs	60	100	116	133	179	200	240	285	125	148	160	245	285	
	Maximum Differential Pairs	60	100	116	133	179	200	240	285	125	148	160	245	285	
Embedded Hard IP Resources	DSP48A1 Slices <sup>(5)</sup>	8	16	32	38	58	132	180	180	38	58	132	180	180	
	PCI Express® Endpoint Block	—	—	—	—	—	—	—	—	1	1	1	1	1	
	Memory Controller Blocks	0	2	2	2	4	4	4	4	2	2	4	4	4	
Speed Grades	GTP Low-Power Transceivers	—	—	—	—	—	—	—	—	2	4	8	8	8	
	Commercial	-2,-3	-2,-3	-2,-3	-2,-3	-2,-3	-2,-3	-2,-3	-2,-3	-2,-3	-2,-3	-2,-3	-2,-3	-2,-3	
	Industrial	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	-4,-1,-2	
Configuration	Configuration Memory (Mbits)	27	27	27	44	77	106	171	280	44	77	106	171	280	



Spartan-6 LX FPGAs			Spartan-6 LXT FPGAs										
Optimized for Lowest Cost Logic, DSP, and Memory (1.2 Volt, 1.0 Volt)			Optimized for Low Cost Logic, DSP, and Memory with High Speed Serial Connectivity (1.2 Volt)										
Part Number	XC6SLX4	XC6SLX9	XC6SLX16	XC6SLX25	XC6SLX45	XC6SLX75	XC6SLX100	XC6SLX150	XC6SLX25T	XC6SLX45T	XC6SLX75T	XC6SLX100T	XC6SLX150T
Package Area Maximum User I/O, SelectIO™ Interface Pins (GTP Transceivers) <sup>10</sup>													
CPG106	8 x 8 mm	100	100	100									
Chip Scale Packages (CSC): Pb-Free wire-bond BGA (0.5 mm ball spacing)													
TOF144	20 x 20 mm	100	102										
Chip Scale Packages (CSC): Pb-free wire-bond chip scale BGA (0.8 mm ball spacing)													
CSG225	18 x 13 mm	120	160	160									
CSG324	15 x 15 mm		200	232	226	218			190 (2)	190 (4)			
CSG484	19 x 19 mm				310	310	320	330		290 (4)	290 (4)	290 (4)	
TQFP Packages (FTG): Pb and Pb-free wire-bond fine-pitch thin BGA (1.0 mm ball spacing)													
FTG256	17 x 17 mm		186	186	186								

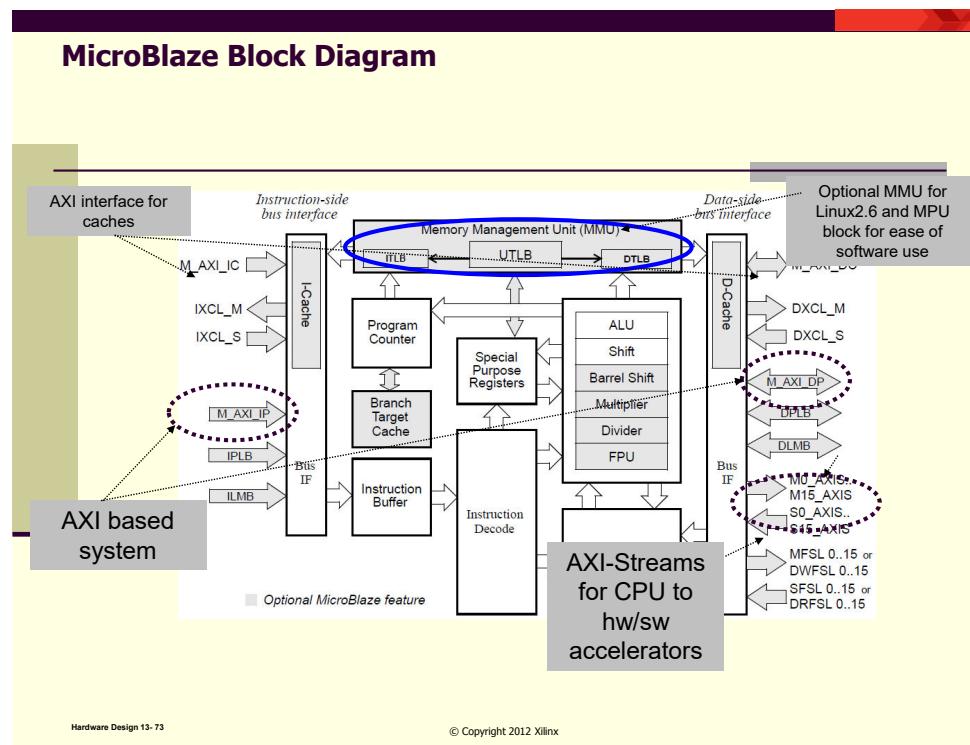
## MicroBlaze Procesor

- 32-bitna prilagodljiva Jezgra
  - Protočna arhitektura
    - 3 stupnja (smanjena količina logike) ili 5 stupnjeva (veća brzina rada)
  - Pričuvna memorija za naredbe i podatke – AXI ili XCL sučelje
    - Direktno mapiranje (1-way asocijativno)
  - Po izboru "Memory Mgt" ili "Memory Protection Unit"
    - Potrebitno za Linux OS (Linux 2.6 podržan)
  - Floating-point unit (FPU)
    - IEEE 754 format
  - Barrel Shifter
  - Sklopovsko množilo
    - 32x32 množilo koje daje 64 bitni rezultat
  - Sklopovsko dijeljenje
  - AXI4 Stream/Lite/Full podrška za direktni pristup programabilnom sklopu
  - Podrška za debagiranje

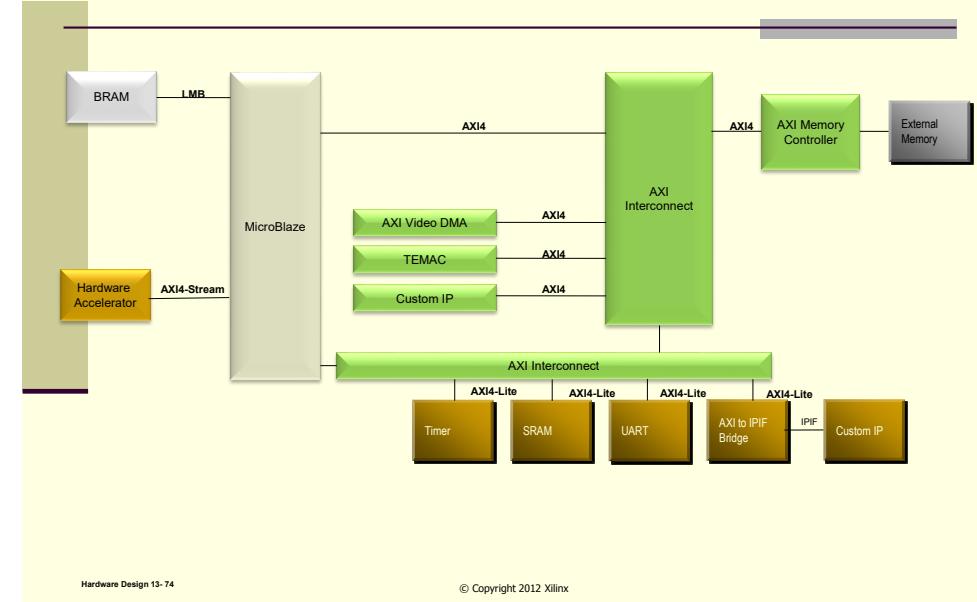
© Copyright 2012 Xilinx

Hardware Design 13- 72

## MicroBlaze Block Diagram



## AXI4 System



Hardware Design 13- 72

© Copyright 2012 Xilinx

## MicroBlaze dodatne funkcije

- Podrška za AXI4 sabirnički sustav
- Memory Management Unit (MMU)
  - PowerPC 405 procesor - MMU compatible
- Procesorska poboljšanja
  - Nove naredbe za konverziju float-integer. Računanje drugog korijena.
- XMD **Xilinx® Microprocessor Debugger**
- AXI4 streaming sučelje

© Copyright 2012 Xilinx

Hardware Design 13- 75

## Podrška za više jezgri

- Multicore arhitektura
  - Mailbox: komunikacija između dvije jezgre
    - Podrška AXI4-Lite, AXI4-Stream and FSL
  - Mutex core: Synchronizacija dva ili više procesora
    - Supports AXI4-Lite and PLBV46
  - Processor Version Register (PVR)
    - Sadrži: Processor ID, configuration/user/processor info (e.g. cache size etc), version number and other internal information

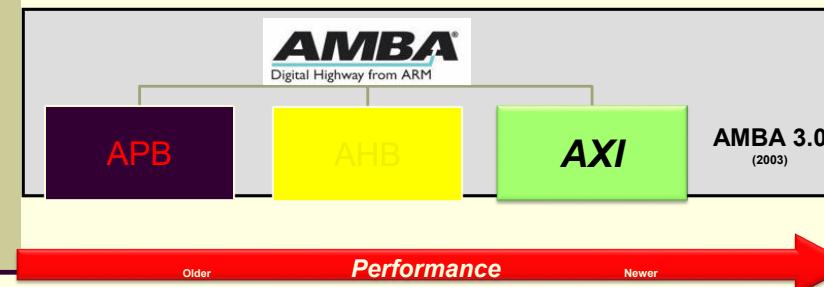
## AXI Streaming Interface

- Jednosmjerna komunikacija, "point-to-point", koristi FIFO spremnike
- Programabilna dubina FIFO spremnika
- Direktna veza s jezgrom procesora
  - Do 16 komunikacijskih kanala
  - Specializirani regitri za čitanje i pisanje u/iz FIFO spremnika

© Copyright 2012 Xilinx

Hardware Design 13- 76

## AXI is Part of ARM's AMBA

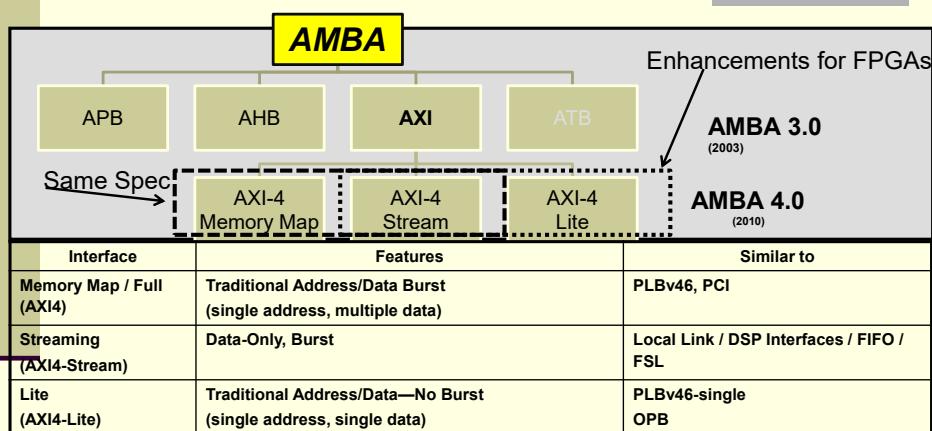


AMBA: Advanced Microcontroller Bus Architecture  
AXI: Advanced Extensible Interface

Hardware Design 13- 78

© Copyright 2012 Xilinx

## AXI is Part of AMBA: Advanced Microcontroller Bus Architecture

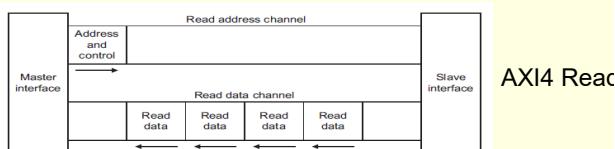


Hardware Design 13- 79

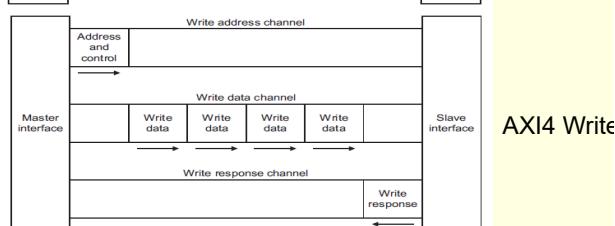
© Copyright 2012 Xilinx

## The AXI Interface—AXI4

- Sometimes called “Full AXI” or “AXI Memory Mapped”
- Not ARM-sanctioned name



- Single address multiple data
- Burst up to 256 data beats

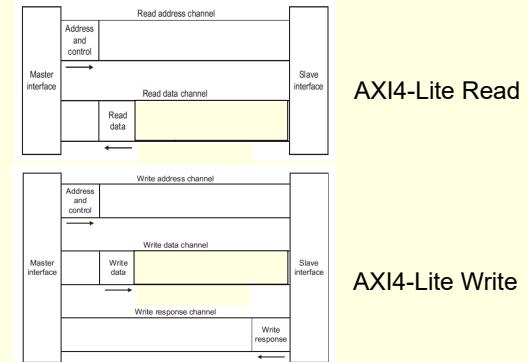


© Copyright 2012 Xilinx

Hardware Design 13- 81

## The AXI Interface—AXI4-Lite

- No burst
- Data width 32 or 64 only
  - Xilinx IP only supports 32-bits
- Very small footprint
- Bridging to AXI4 handled automatically by AXI\_Interconnect (if needed)

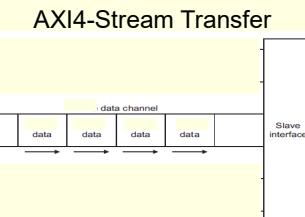


© Copyright 2012 Xilinx

Hardware Design 13- 80

## The AXI Interface—AXI4-Stream

- No address channel, no read and write, always just master to slave
- Effectively an AXI4 “write data” channel



- Unlimited burst length
  - AXI4 max 256
  - AXI4-Lite does not burst

- Virtually same signaling as AXI Data Channels

- Protocol allows merging, packing, width conversion
- Supports sparse, continuous, aligned, unaligned streams

© Copyright 2012 Xilinx

Hardware Design 13- 82

## Processor Local Bus (PLB)

- Ne preporuča se. Kompatibilnost sa starijim sustavima
- Potpuno sinkronizirani protokol
- Centralizirani sklop za arbitražu—PLB arbiter
- 32 ili 64-bitna adresa
- 32, 64 ili 128-bitni podatci
- Mogućnost dijeljenja sabirnice ili point-to-point komunikacija
- Protočna struktura (2 nivoa)

© Copyright 2012 Xilinx

Hardware Design 13- 83

## PLB Most

- PLB-to-PLB most je potreban kada dva PLB segmenta komuniciraju
  - Npr.:
    - Sabirnice različite brzine
    - Sabirnice različite širine

© Copyright 2012 Xilinx

Hardware Design 13- 84

## AXI Most

- AXI\_to\_PLBv46 | PLBv46\_to\_AXI Most
  - Used in system having two standards (Core-Connect and AMBA)
  - Supports multi-master/multi-slave connections
  - Designed to support existing customer PLBv46-based cores in an AXI system
- AXI\_to\_APB Bridge
  - Designed to support 3<sup>rd</sup> party slave IP talking to an AXI4-Lite master
  - The bridge is slave on the AXI4Lite side and master on the APB peripheral side
  - APB3/APB4 peripherals are supported

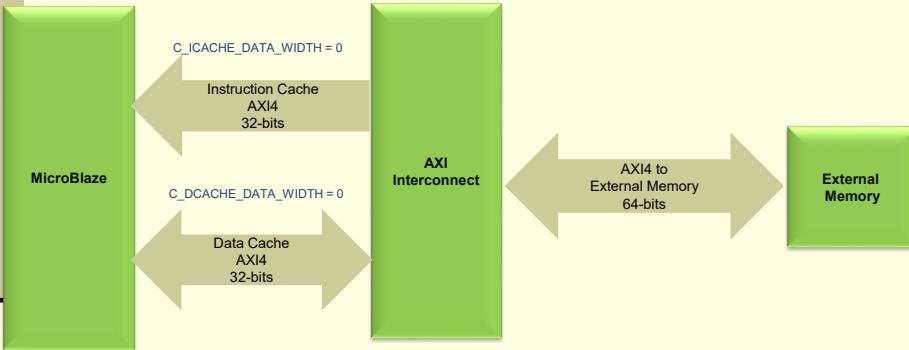
© Copyright 2012 Xilinx

Hardware Design 13- 85

- Local Memory Bus (LMB)
  - Za povezivanje BRAM memorije
- Fast Simplex Links ( FSL)
  - Za dodavanje koprocесorskih naredbi – ne preporuča se
- Xilinx Cache Link
  - Sučelje za pričuvnu memoriju – ne preporuča se
- MicroBlaze AXI Cache Interfaces

## MicroBlaze Cache to External Memory Datapath

Default Configuration, same as pre-AXI



Hardware Design 13- 67

© Copyright 2012 Xilinx

# Multimedijiske arhitekture i sustavi

Prof.dr.sc. Mario Kovač

Izv.prof.dr.sc. Hrvoje Mlinarić



Multimedijiske arhitekture i sustavi

1

Ova predavanja koriste jedan dio materijal koji je ustupljen od strane XILINX pod XUP uvjetima korištenja.

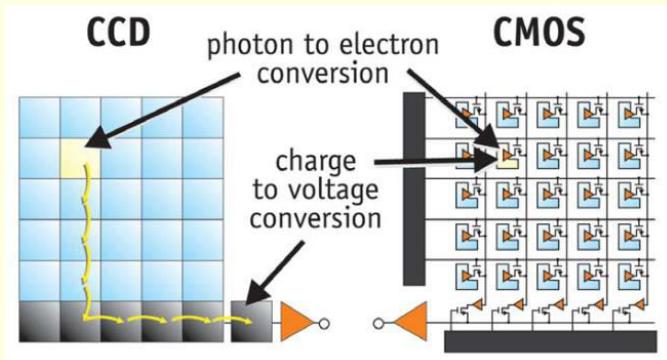
## Video Senzor

- Dvije vrste - tehnološki različite
  - CCD (charge coupled device)
  - CMOS (complementary metal oxide semiconductor)
- Svaka tehnologija ima svoje prednosti i mane u ovisnosti o primjeni. Niti jedan nema superiornu prednost pred drugim iako često možete naći da proizvođači tvrde suprotno.

## Video Senzor

- **CCD senzor**
  - naboj svakog piksela prenosi se vrlo kratkim putovima i kroz mali broj čvorova prije nego se pretvori u naponsku razinu i pošalje izvan čipa kao analogni signal (koristi se samo jedan ili mali broj pretvarača). Svi pikseli dohvaćaju se istovremeno i uniformno se obrađuju (osnovni uvjet kvalitetne slike).

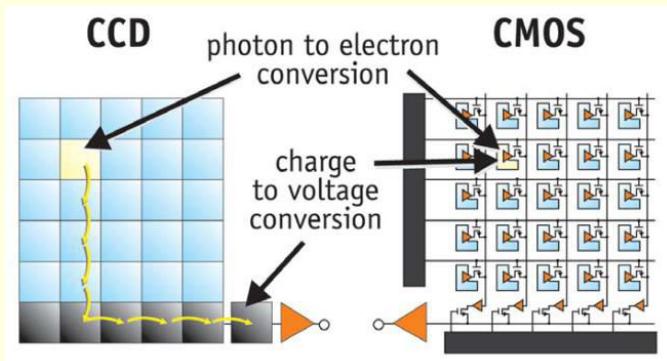
## Video Senzor



Multimedijiske arhitekture i sustavi

4

## Video Senzor



Multimedijiske arhitekture i sustavi

6

## Video Senzor

### ■ CMOS senzor:

- Svaki piksel ima svoj zasebni sklop za pretvorbu naboja u napon. Senzori obično posjeduju i pojačala, filtre šuma, i digitalni sklop sve na jednom čipu.
- Sve navedeno povećava fleksibilnost dizajna.
- Pošto svaki piksel ima svoj pretvarač narušava se uniformnost konverzije, ali zato čip može biti izrađen da treba mali broj vanjskih komponenti.

Multimedijiske arhitekture i sustavi

5

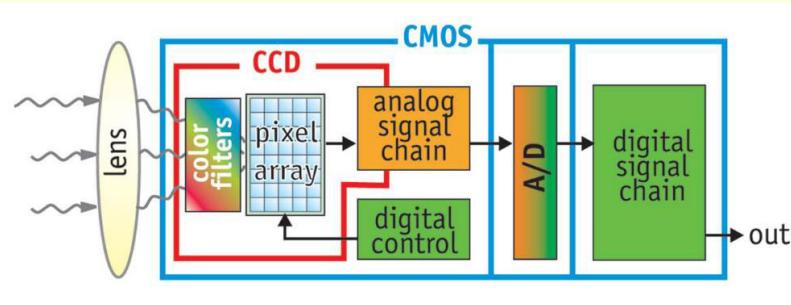
## Video Senzor

- CCD i CMOS izumljeni su kasnih '60 i početkom '70 (osnivač DALSA Dr. Savvas Chamberlain).
- Zbog svoje strukture i jednostavnije izvedbe (čitaj cijene ☺) CCD senzori su postali dominantni.
- Napretkom tehnologije '90 godina ponovo se pojavljuje interes za CMOS senzorima

Multimedijiske arhitekture i sustavi

7

## Video Senzor



Multimedijiske arhitekture i sustavi

8

## OV7670

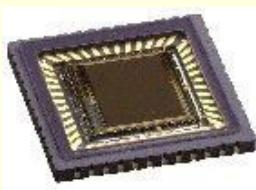
- 640x480 – VGA format
- 24 pina
- Podržan format – YUV 4:2:2, GRB 4:2:2, RGB Raw Data 565/555
- 8 video data: ITU-601, ITU-656, ZV port
- Automatska ekspozicija/gain/kontrola bijele boje (WB)
- Operacije nad slikom – svjetloća, kontrast, gamma, saturation, oštrina, uzimanje dijela slike, i.t.d.
- Vanjska i unutarnja sinkronizacija

Multimedijiske arhitekture i sustavi

10

## Video Senzor

- OmniVision's OV7670 SINGLE-CHIP CMOS VGA COLOR DIGITAL CAMERA



Multimedijiske arhitekture i sustavi

9

## OV7670

- Frame exposure/line exposure option (za foto aparate)
- 3.3V Volt operation, low power dissipation
  - < 80 mW active power
  - < 20 uA in power-save mode
- I2C kontrolno sučelje (400 kb/s):
  - - Color saturation, brightness, contrast, white balance,exposure time, gain

Multimedijiske arhitekture i sustavi

11

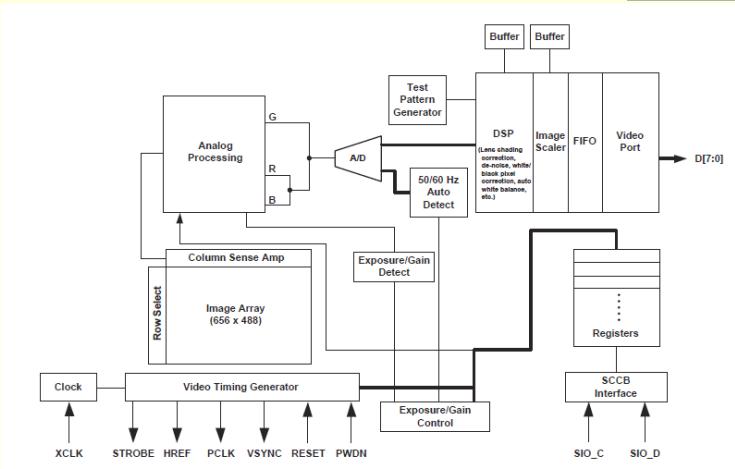
## OV7670

- Postoji i crno-bijela verzija OV7171
- CMOS Video senzor
- Ukupan broj pikslea 656 x 488
- Maksimalno do 60 slika po sekundi

Multimedijiske arhitekture i sustavi

12

## OV7670



Multimedijiske arhitekture i sustavi

14

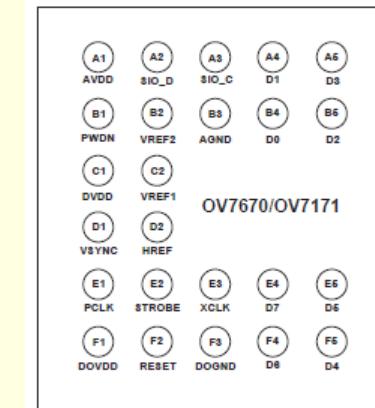
## OV7670

- Predviđena za sljedeće funkcije:
  - Video konferencije
  - Video telefonija
  - Video pošta
  - Foto aparati na mobilnim telefonima
  - PC Multimedia
  - i.t.d.

Multimedijiske arhitekture i sustavi

13

## OV7670



Multimedijiske arhitekture i sustavi

15

# OmniVision

## ■ Image Sensors

- [16-megapixel](#)
- [14-megapixel](#)
- [13-megapixel](#)
- [12-megapixel](#)
- [10-megapixel](#)
- [9-megapixel](#)
- [8-megapixel](#)
- i.t.d.

# SCCB - Serial Camera Control Bus

- OmniVision komunikacijski protokol
- Omogućava komunikaciju jednog master i više slave uređaja
- Sukladno s IIC (I2C) sučeljem

# OV7670

## ■ Dva komunikacijska sučelja

- Kontrolno sučelje SCCB
- Podatkovno sučelje: ITU-601, ITU-656, ZV port

# I2C

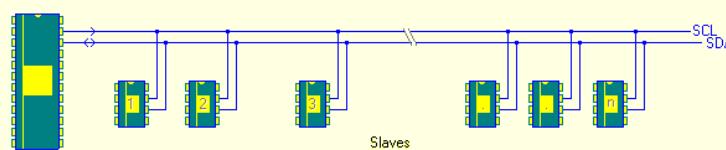
- I2C - Inter-Integrated Circuit
- Početkom '80ih Philips Semiconductors razvio je dvosmjernu "2-wire" komunikacijsku sabirnicu.
- Osnovna namjena joj je bila da omogući jednostavnu komunikaciju između procesora i ostalih komponenti unutar televizora.
- Philips Labs u Eindhoven (Nizozemskoj)
- Danas I2C široko primijenjen i u drugim uređajima.

# I2C

- Generalno je prihvaćena kao standard
- Podržana od mnogih proizvođača:
  - Xicor
  - ST Microelectronics
  - Infineon Technologies
  - Intel
  - Texas Instruments
  - Maxim
  - Atmel
  - Analog Devices
  - i.t.d.

# I2C

- Svojstva sabirnice
  - Više uređaja može započeti komunikaciju
  - Uredaj koji započinje komunikaciju naziva se master na sabirnici
  - Sukladno tome svi ostali u tom trenutku su slave uređaji
  - Master na sabirnici je obično procesor
  - Sabirnica može imati više master uređaja



# I2C

- Fizički je izvedena sa dvije žice (spojna puta)
  - SDA – Serial DAta
  - SCL – Serial CLock
- Obije linije su dvosmjerne
- Svaki uređaj spojen na sabirnicu mora imati svoju jedinstvenu adresu na toj sabirnici
- Svaki od uređaja može slati ili primati podatke

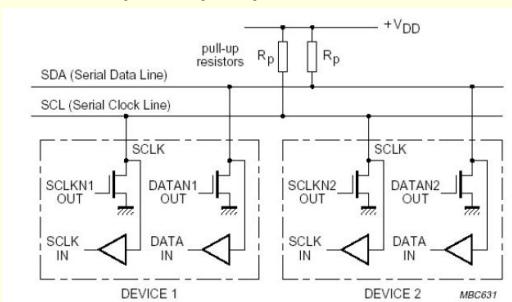
# I2C

- Brzina komunikacije
  - 100 kbps (standardni način rada)
  - 400 kbps(brzi način rada)
  - 3.4 Mbps (vrlo-brzi načina rada)
- Broj uređaja na sabirnici je limitiran s maksimalnim kapacitetom od 400 pF

# I2C

## ■ Spojeno i sabirnica

- Sabirnica je slobodna kada su SDA i SCL u visokom.
  - Koriste se pull-up otpornici



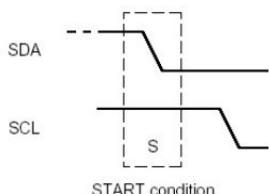
Multimedijiske arhitekture i sustavi

24

# I2C

## ■ Komunikacija

- Master generira START stanje
  - Dojavljuje ostalim na sabirnici da zahtijeva pozornost
  - Postavlja SDA signal u nisko, a zatim SCL signal u nisko

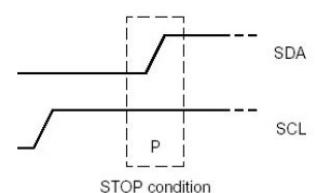


Multimedijiske arhitekture i sustavi

25

# I2C

- Nakon što master primi potvrdu može početi slati podatke.
- Na kraju master šalje STOP
  - Otpušta SCL i SDA signal koji idu u visoko stanje



Multimedijiske arhitekture i sustavi

26

# I2C

## ■ Master

- Kontrolira SCL
- Šalje start i stop sekvencu
- Kontrolira adresu na sabirnici

## ■ Slave

- Adresiran je od strane master-a
- Ako master čita onda šalje bitove

Multimedijiske arhitekture i sustavi

27

# I2C

## Prijenos podataka

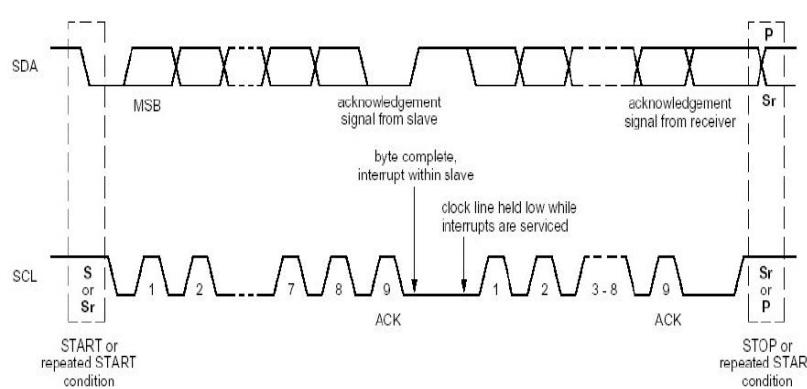
- Podatci se prenose po bitovima nakon start signala
- Uvijek se šalju podatci grupirani u bajtove
- MSB (most significant bit) ide prvi
- Adresa SLAVE uređaja je isto podatak
  - I to prvi podatak koji se prenosi
  - Tijekom prijenosa prvog bajta master šalje a slave prima adresu
  - Dalje sve ovisi o zadnje poslanom bitu

# I2C

## Prijenos podataka

- Master spušta SCL u nisko i počinje generirati impulse za svaki bit
- Generira se 8 pulsova za podatak i jedan puls za potvrdu od slave uređaja
  - Master započinje slanje sljedećeg bajta
  - Slave može odgoditi slanje sljedećeg ili primanje bajta držanjem signala SCL u niskom

# I2C



# I2C

## Prijenos podataka

- Prenosi se osam bitova i potvrda od primatelja
- Uređaj koji šalje podatke nakon 8 pulsova oslobađa SDA
- Onaj koji je primao podatke spušta SDA u nisko da bi potvrdio primitak podataka
- Nakon toga otpušta SDA signal

# I2C

## ■ Multi master sabirnica

- Više uređaja može kontrolirati sabirnicu
- Moguća situacija je da više mastera istovremeno pokrenu start sekvencu
- Potrebna je sinkronizacija na SCL signali
- Potrebna je arbitraža na SDA signali
- Problem se rješava korištenjem ožičenog I povezivanja

# I2C

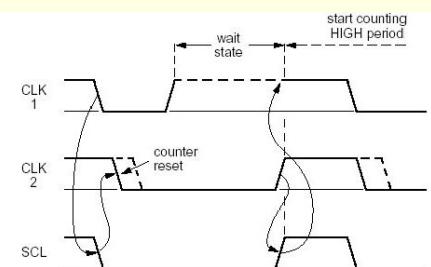
## ■ Arbitraža na SDA signalu

- Započinje komunikacija šalje se START
- Izvrši se sinkronizacija SCL signala i na visoku razinu SCL postavlja se podatak
- Svaki master generira svoj podatak
- Master prestaje slati ako razina na SDA signalu ne odgovara onome što je on postavio
  - Oslobađa SDA i pokušava ponovo poslati podatak kada je sabirnica slobodna

# I2C

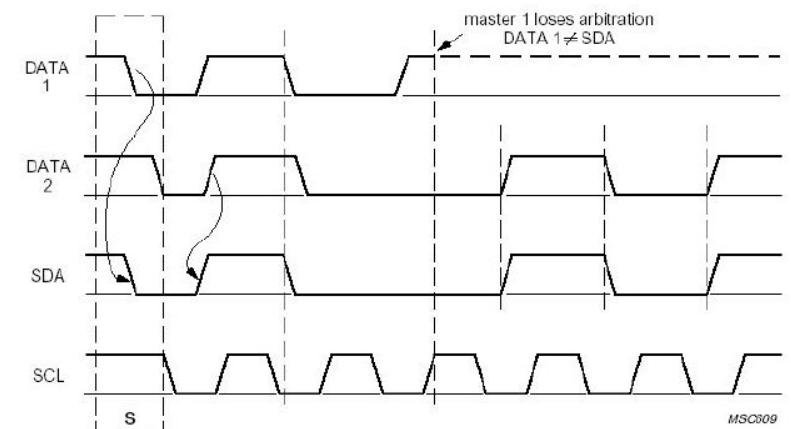
## ■ Sinkronizacija na SCL signalu

- Započinje komunikacija
  - SCL je u 1 i prvo što se radi master postavlja SCL u 0 – START
  - Nakon toga Master otpušta SCL
    - Ako je SCL = 0 netko je još na sabirnici odi u stanje čekanja
    - Ako je SCL = 1 sabirnica je tvoja
  - Master vraća SLC u 0



# I2C

master 1 loses arbitration  
DATA 1 ≠ SDA



# I2C

## ■ Adresiranje:

- Prvi bajt uvijek šalje master
  - 7 bitova čine adresu
  - 1 bit daljnji smjer komunikacije
    - 0 – master piše podatke
    - 1 – master čita podatke
- Prijenos podataka završava STOP stanjem, može se prenijeti više od jednog bajta
- Adresa se sastoji od fiksног dijela i promjenjivog
  - Fiksni dio određuje I2C odbor za dodjelu adresa

# I2C

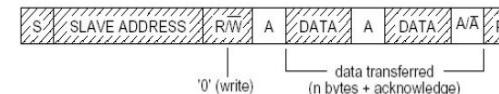
## ■ Rezervirane adrese za posebnu namјenu:

■ general call	0000 000   0
■ start byte	0000 000   1
■ CBUS address	0000 001   *
■ used for cooperation of I2C and CBUS	
■ High-speed mode master code	0000 1**   *
■ 10-bit slave addressing	1111 0**   *

# I2C

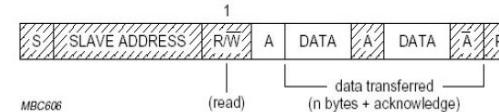
## ■ Okvir komunikacije

master-transmitter



■ from master to slave  
■ from slave to master  
A = acknowledge (SDA LOW)  
Ā = not acknowledge (SDA HIGH)  
S = START condition  
P = STOP condition

master-receiver (since second byte)



# SCCB

- Podržava brzinu prijenosa do 400 kbps
- Ima 7 bitnu adresu

CS[2:0]	000	001	010	011	100	101	110	111
WRITE ID (hex)	C0	C4	C8	CC	D0	D4	D8	DC
READ ID (hex)	C1	C5	C9	CD	D1	D5	D9	DD

## SCCB

- Osim osnovnog adresiranje preko kojeg se odabire senzor s kojim se komunicira SCCB podržava i pod adresiranje internih registara unutar video senzora
- Tijekom slanja podataka nakon adrese senzora u drugom bajtu se šalje adresa registra kojem se želi pristupiti
- Treći bajt koji se šalje zapisuje se u odabrani registar
- U koliko se nastavi dalje slati bajtove podatci se zapisuju u sljedeće registre koji slijede

## SCCB

- Čitanje podataka po I2C protokolu ne omogućava pod adresiranje pojedinih registara unutar video senzora.
- Zato se čitanje vrši sa zadnje adresiranog registra u procesu pisanja
- Moramo koristiti jedan prazan ciklus pisanja u kojem ne upisujemo podatak u registar, odnosno šaljemo samo dva bajta – adresu senzora i adresu registra

## SCCB

- Nama bitni registri unutar senzora:

Register	Address	Default	Description
CLKRC	0x11	0x80	<p><i>Bit[6]: 0: Apply prescaler on input clock 1: Use external clock directly</i></p> <p><i>Bit[0-5]: Clock prescaler <math>F(\text{internal clock}) = F(\text{input clock}) / (\text{Bit}[0-5] + 1)</math> Range [0 0000] to [1 1111]</i></p>
DBLV	0x6B	0x0A	<p><i>Bit[7-6]: PLL control 00: Bypass PLL 01: Input clock x4 10: Input clock x6 11: Input clock x8</i></p> <p><i>Bit[4]: Regulator control 0: Enable internal regulator 1: Bypass internal regulator</i></p>

Register	Address	Default	Description
COM3	0x0C	0x00	<p>0: Nothing 1: Swap the data MSB and LSB.</p> <p><i>Bit[6]: On powdown 0: Tri-state the output clock 1: Do not tri-state the output clock</i></p> <p><i>On powerdown 0: Tri-state the output data 1: Do not tri-state the output data</i></p> <p><i>Bit[3]: Disable scaling 1: Enable scaling</i></p> <p><i>Bit[2]: 0: Disable downsampling, cropping, windowing 1: Enable downsampling, cropping, windowing</i></p>
COM7	0x12	0x00	<p><i>Bit[7]: 0: Nothing 1: Reset all the registers to default values</i></p> <p><i>Bit[5]: 0: Nothing 1: Use CIF format</i></p> <p><i>Bit[4]: 0: Nothing 1: Use QVGA format</i></p> <p><i>Bit[3]: 0: Nothing 1: Use QCIF format</i></p> <p><i>Bit[1]: 0: Disable color bar 1: Enable color bar</i></p> <p><i>Bit[2, 0]: 00: YUV 01: RGB 10: Bayer raw 11: Processed bayer raw</i></p>

# OV7670

- Nama bitni registri unutar senzora:

70	SCALING_XSC	4A	RW	Bit[7]: Test_pattern[0] - works with test_pattern[1] test_pattern (SCALING_XSC[7], SCALING_YSC[7]): 00: No test output 01: Shifting "1" 10: 8-bar color bar 11: Fade to gray color bar Bit[6:0]: Horizontal scale factor
71	SCALING_YSC	35	RW	Bit[7]: Test_pattern[1] - works with test_pattern[0] test_pattern (SCALING_XSC[7], SCALING_YSC[7]): 00: No test output 01: Shifting "1" 10: 8-bar color bar 11: Fade to gray color bar Bit[6:0]: Vertical scale factor

# ITU656

- Pravi naziv BT.656 predložen od strane ITU (International Telecommunication Union) stoga se često naziva ITU656 i CCIR656
- Definira jednostavni video protokol za prijenos ne komprimiranog digitalnog video signala bilo PAL ili NTSC (522 ili 625 linija)
- Standar je nastao na BT.601(CCIR601) koji definira prijenos podataka u formatu 4:2:2 interlaced u YUV (YCbCr)
- Standard definira prijenos 8 i 10 bitnih podataka serijski ili paralelno.
- Koristi se za prijenos podataka u televizorima između čipova

# Video sučelje

- Video senzor podržava

- Digitalno sučelje

- 8 bita
  - Video format CCIR601, CCIR656, ZV port
  - Format podataka – YUV 4:2:2, RGB 4:2:2, RGB row dana 565/555

# ZV Port

- ZV Port je dio PC Card standarda koji definira PCMCIA i PC Express kartice
- ZV (Zoomed Video) je protokol koji služi za povezivanje PC Card (PCMCIA) i host sistema (računala) koji omogućava direktni pristup video memoriji i/ili VGA upravljačkom sklopu. Podatci se prenose bez upotrebe međuspremnika odvojenom sabirnicom tako da ne opterećuju računalo.
- Pogodan je za jednostavno i jeftino povezivanje video uređaja koji zahtijevaju brzi prijenos podataka za aplikacije kao sto su MPEG dekoderi za filmove i igrice, TV tuners, live video input and video capture.

## ZV Port

- Korištenjem ZV porta prijenosna računala dobivaju performanse desktop računala u video svijetu
- ZV Port je jednosmjerna komunikacija između PC Card-a i VGA kontrolera
- U potpunosti je kompatibilno s CCIR601

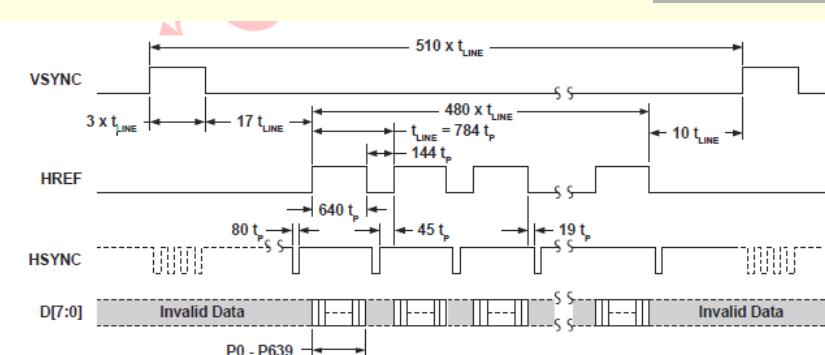
## OV7670

- Signali
  - VSYNC
    - Postavljanjem u visoko dojavljuje se početak okvira slike. Generira se jednom za svaku sliku
  - HREF
    - Postavlja se visoko i ostaje u visokom kada se šalju jedan red slike po završetku slanja jednog reda postavlja se u nisko
  - PCLK
    - Na padajući ili rastući brid signala podatak na sabirnici je valjani
    - Opcija rastući ili padajući brid može se podesiti preko I2C-a

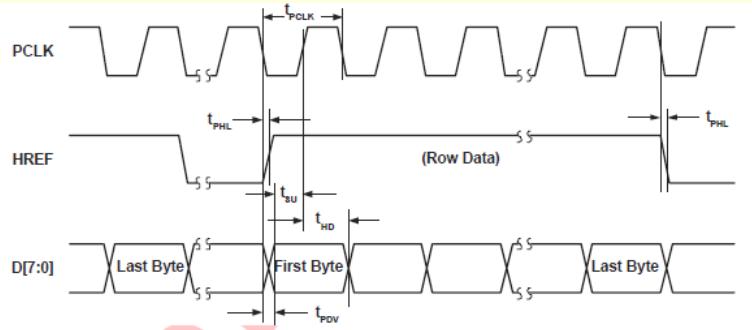
## ZV Port

- Više detalja u PC Card Standard
- Dodano u standard od verzije PC Card Standard 5.04 Update
- Trenutna verzija (koja je ujedno i konačna)
  - PC Card Standard 8.0
- PC Card je napušten
- Trenutno se razvija Express Card koji podržava sve što i PC Card
- Fizički nisu kompatibilni

## ZV Port



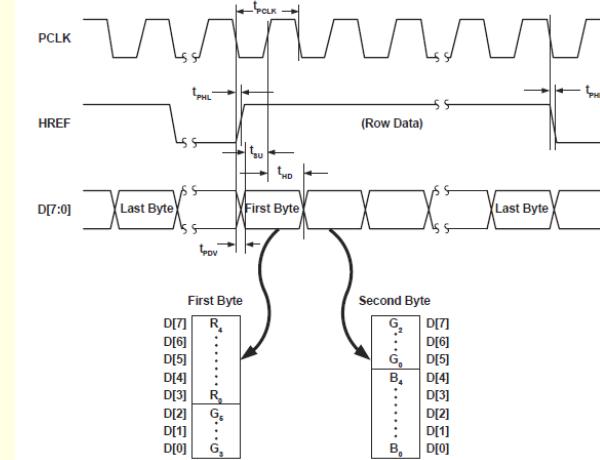
## ZV Port



Multimedijiske arhitekture i sustavi

52

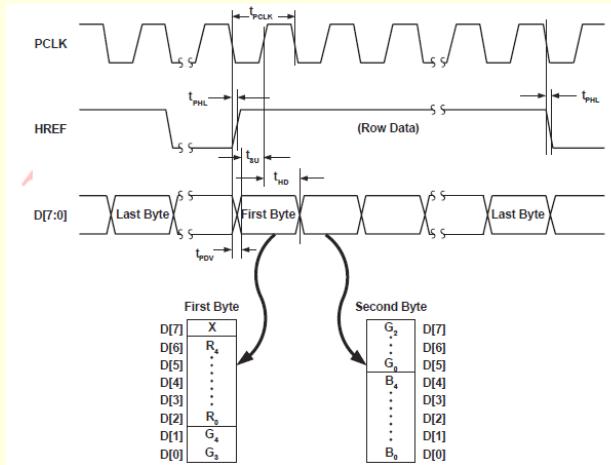
## OV7670 – RGB565



Multimedijiske arhitekture i sustavi

53

## OV7670 – RGB555



Multimedijiske arhitekture i sustavi

54

## OV7670

### Ostali signali koje koristimo

#### RESET

- Sklopovalski reset kamere

#### PWDN

- Postavljanje kamere u "Power Down Mode"
  - Način rada smanjene potrošnje
  - U tom načinu rada senzor zaustavlja obradu video signala.

Multimedijiske arhitekture i sustavi

55

## OV7670

- Video senzor radi kao MASTER uređaj
- Moguće ga je postaviti da radi i kako SLAVE uređaj
- Tada procesor mora kontrolirati i generirati signale CHSYNC, VSYNC, PCLK. On se mora brinuti o horizontalnoj i vertikalnoj sinkronizaciji

## OV7670

- Vaš zadatak
  - Kreirati vaš IP za čitanje sa ZV porta ili koristiti postojeći IP-ove (GPIO)
  - Koristimo jedno od dva I2C sučelje koje se nalazi na ARM procesor
  - Napisati program za čitanje s ZV porta
  - Napisati program za konfiguraciju kamere preko I2C porta
    - moramo usporiti kameru 60 fps je malo prebrzo za nas ☺, a možda i nije – ovisi o izvedbi

## OV7670

- Frame execution mode
  - Predviđen za uređaje koji imaju mehaničku kontrolu otvora blende
  - Digitalni foro aparati

## Xilinx Processor IP Library

- iicps v1\_03\_a
- Uređaj može biti master ili slave
- Omogućeno je korištenje prekida prilikom čitanja i pisanja ili prozivanje
- lako sklopoljje podržava 7 i 10 bitnu adresu izvedeni upravljački programi podržavaju samo 7 bitno adresiranje