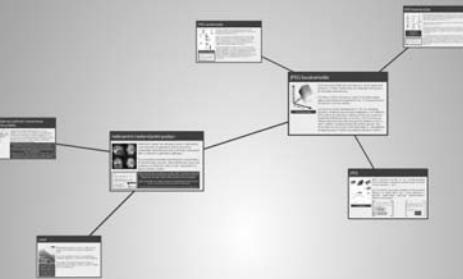


8. Kompresija slika

Dr.sc. Martin Žagar
Multimedejske arhitekture i sustavi



8. Kompresija slika

Dr.sc. Martin Žagar
Multimedejske arhitekture i sustavi

Uvod



Vizualni tipovi podataka, kao što su slike, video i grafika su sveprisutni u današnjem digitalnom svijetu.

Sve je više podataka, sve je veća potražnja za pristupom, prijenosom i pohranom takvih podataka.

Tehnologija kompresije slika je glavni uvjet za ostvarivanje ovih zahtjeva, koji se očituje u mnogim različitim aplikacijama.

Na primjer, digitalna fotografija ne bi bila moguća bez kompresije slike jer i najobičnije kamere na mobilnim telefonima danas imaju prosječno 10 megapikselski CCD (što generira 30 MB sirovih RGB podataka) po slici. Većina kamera za kompresiju slike koristi JPEG standard, gdje se prosječna slika od 10 megapiksela sažima na 1 MB - 2MB.

Redundancija (zalihost) i relevantnost slikovnih podatka



Tehnike kompresije slika mogu biti i sa i bez gubitaka, ali najčešće su kompresije slike hibridne, kombinirajući učinkovito korištenje algoritama sa i bez gubitaka za sažimanje slikovnih podataka. Takve tehnike se temelje na analizi slikovnih podataka u skladu sa dva važna aspekta:

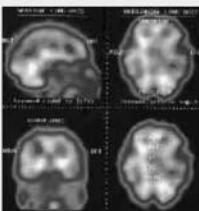
- Uklanjanje irelevantnih podatka – prvenstveno vizualno irelevantnih (na primjer kad je rezolucija slike veća od rezolucije uređaja za prikazivanje), ali i aplikacijsko specifičnih irelevantnosti (na primjer u medicinskim aplikacijama kada je određeno područje slike potpuno nebitno za daljnju obradu).
- Uklanjanje zalihosti – slikovni podaci (kao i svaki drugi podaci), imaju statistički zalihost, jer vrijednosti piksela nisu slučajne, nego visoko povezane, bilo u lokalnim područjima i bilo na razini cijele slike.

Na primjer, digitalna fotografija ne bi bila moguća bez kompresije slike jer i najobičnije kamere na mobilnim telefonima danas imaju prosječno 10 megapikselski CCD (što generira 30 MB sirovih RGB podataka) po slici. Većina kamera za kompresiju slike koristi JPEG standard, gdje se prosječna slika od 10 megapiksela sažima na 1 MB - 2MB.

kombinirajući učinkovito korištenje algoritama sa i bez gubitaka za sažimanje slikovnih podataka. Takve tehnike se temelje na analizi slikovnih podataka u skladu sa dva važna aspekta:

- Uklanjanje irelevantnih podatka – prvenstveno vizualno irelevantnih (na primjer kad je rezolucija slike veća od rezolucije uređaja za prikazivanje), ali i aplikacijsko specifičnih irelevantnosti (na primjer u medicinskim aplikacijama kada je određeno područje slike potpuno nebitno za daljnju obradu).
- Uklanjanje zalihosti – slikovni podaci (kao i svaki drugi podaci), imaju statistički zalihost, jer vrijednosti piksela nisu slučajne, nego visoko povezane, bilo u lokalnim područjima i bilo na razini cijele slike.

Irelevantni i redundantni podaci



Redundancija može biti klasificirana na sljedeći način:

- prostorna redundancija – vrijednosti piksela u manjim područjima su vrlo slične
- spektralna redundancija – kad se podaci preslikavaju u frekvencijsku domenu, nekoliko frekvencija dominira nad drugima.

Kod video postoji još jedna vrsta redundancije:

Irelevantni podaci se uklanjaju pomoću algoritama za kompresiju sa gubicima budući da postoji nedostatak relevantnosti (važnosti) bilo u percepciji bilo u važnosti za potrebne aplikacije.

To se postiže primjenom kvantizacije u prostornoj i/ili frekvencijskoj domeni. Takve distorzije moraju biti svedene na minimum, kako bi bile neopažene za ljudski vizualni sustav.

Zajednička karakteristika većine slika je da su susjedni pikseli u korelaciji koja nastaje zbog toga što slika nije slučajna zbirka piksela, nego koherentna struktura koja se sastoji od objekata. Osim toga, ako se vrijednost piksela promjeni, promjena je postupna.

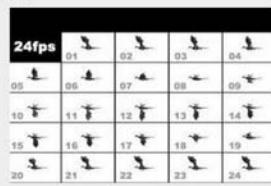
Ukupno se svi pikseli u svim regijama zajedno ne bi mogli prikazati u korelaciji, ali kada se razmatra manje područje, manje je odstupanje između piksela.

Ova lokalna sličnost odnosno redundancija se analizira u tehnikama kompresije slike.

Redundancija može biti klasificirana na sljedeći način:

- **prostorna redundancija** – vrijednosti piksela u manjim područjima su vrlo slične
- **spektralna redundancija** – kad se podaci preslikavaju u frekvencijsku domenu, nekoliko frekvencija dominira nad drugima.

Kod video postoji još jedna vrsta redundancije:
vremenska redundancija



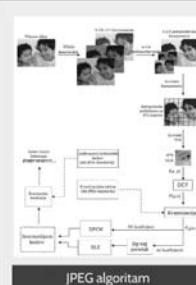
To se postiže primjenom kvantizacije u prostornoj i/ ili frekvencijskoj domeni. Takve distorzije moraju biti svedene na minimum, kako bi bile neopažene za ljudski vizualni sustav.

Zajednička karakteristika većine slika je da su susjedni pikseli u korelaciji koja nastaje zbog toga što slika nije slučajna zbirka piksela, nego koherentna struktura koja se sastoji od objekata. Osim toga, ako se vrijednost piksela promjeni, promjena je postupna.

Ukupno se svi pikseli u svim regijama zajedno ne bi mogli prikazati u korelaciji, ali kada se razmatra manje područje, manje je odstupanje između piksela.

Ova lokalna sličnost odnosno redundancija se analizira u tehnikama kompresije slike.

JPEG



JPEG standard temelji se na transformacijskoj tehnici Diskretni kosinusne transformacije (Discrete Cosine transform - DCT).

DCT je izabran zbog dobre distribucije frekvencijske domene za realne slike, kao i zbog efikasnosti u prilikom aritmetičkih operacija implementiranih direktno na hardveru.

JPEG kompresija počinje transformacijom u prostor boja zastupljen sa YCbCr komponentama.		
Prostor boja YCbCr je vrlo sličan YUV prostoru boja koji smo prije objasnili.		

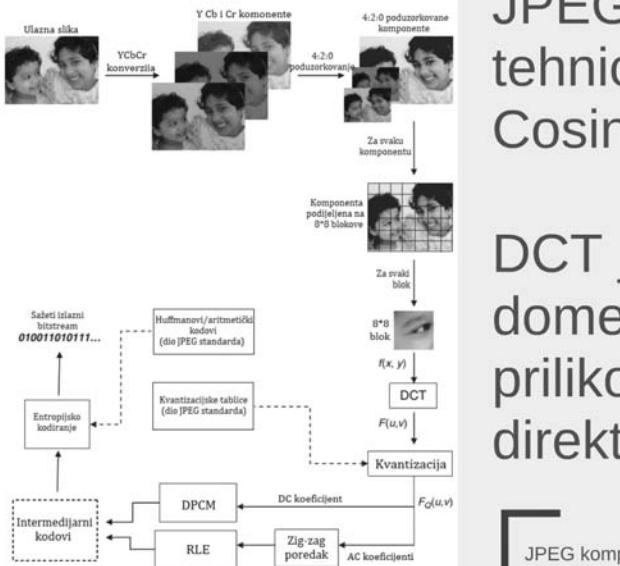
Stavka od Y, Cr i Cb komponenta se odnosi resursima raspolaživima u svakoj bloku. U ovom slučaju nemačkih redova blokovi su oblikovani u brojeve u liniju na desno i razmatraju prema desno i nisu usredotočeni prema DCT transformatoru, a vremenjska koherencija i kompatibilnost kodiranja.

nesavetne karakteristike slike kodne prikazane su u sljedećem poglavju



JPEG technic Cosir

DCT dome prilik direkt



JPEG kompresija počinje transformacijom u prostor boja zastupljen sa YCbCr komponentama.

Prostor boja YCbCr je vrlo sličan YUV prostoru boja koji smo prije objasnili.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

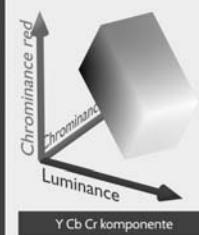
$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Svaka od Y, Cb i Cr komponenta se računa neovisno razbijanjem u 8×8 blokove. U osnovnom načinu rada, blokovi su skenirani u linije s lijeva na desno i odozgora prema dolje i nad svakim se provodi DCT transformacija, kvantizacija koeficijenata i entropijsko kodiranje.

Istaknute karakteristike svakog koraka prikazane su u sljedećem popisu:



JPEG karakteristike

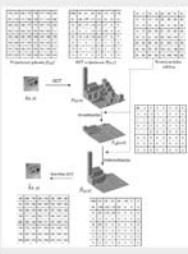


(1) Svaka slika može se uzeti kao ulaz, ali se uvijek prvo pretvara u YCbCr format da bi se razdvojile luminantna i krominantne komponente.

(2) Slika u YCbCr formatu se zatim 4:2:0 poduzorkuje, gdje su krominantne komponente Cb i Cr poduzorkovane na četvrtinu izvorne veličine.

(3) Zatim se svaka komponenta (Y, Cb, Cr) obraduje posebno. Svaka komponenta je podijeljena u 8×8 blokova. (8×8) veličina je optimalna veličina prozora za prostorne i spektralne korelacije koje se koriste u DCT kvantizaciji. Manje veličine prozora povećavaju broj blokova u slici, a veće veličine smanjuju korelaciju između piksela. Ako širina slike (ili visina) nije višekratnik broja 8, rubni blokovi se popunjavaju s nulama da postignu potrebnu veličinu.

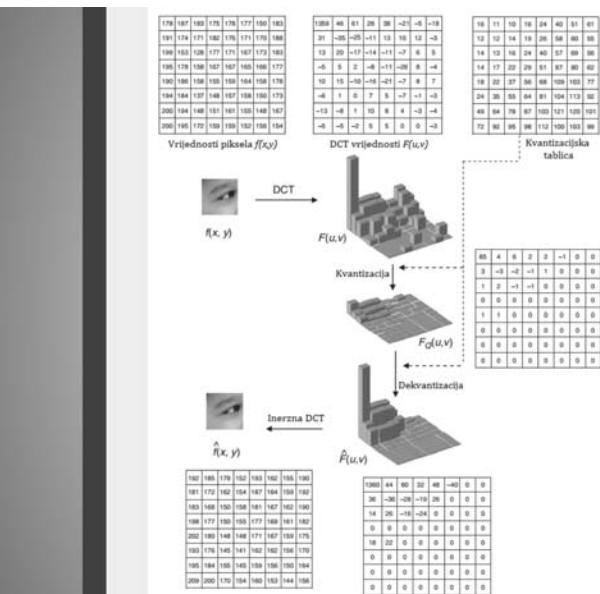
JPEG karakteristike



(4) Svaki 8×8 blok (za sve komponente) prolazi DCT transformaciju koja uzima blokove slike $f(x,y)$ i izračunava koeficijente frekvencija $F(u,v)$.

Slika $f(x,y)$ i pripadne numeričke vrijednosti intenziteta prikazani su gore lijevo. Srednji dio pokazuje izračunatu DCT transformaciju $F(u,v)$. Iako su vrijednosti realni brojevi, tablica pokazuje DCT vrijednosti zaokružene na najbliži cijeli broj za kompaktnost u prikazu.

3D crtež pokazuje DCT vrijednosti u 3D. Prvi koeficijent, odnosno koeficijent na mjestu s indeksom (0,0) ima obično najvišu vrijednost i zove se DC koeficijent. Poseban status za DC koeficijent je namjerno naglašen jer je većina energije u prirodnim fotografijama koncentrirana u najnižim frekvencijama. Preostali koeficijenti se nazivaju AC koeficijenti.



Slika: DCT izračun 8×8 bloka

(5) Zatim se DCT koeficijenti $F(u,v)$ kvantiziraju pomoću kvantizacijske tablice definirane JPEG standardom. Ova tablica je prikazana u gornjem desnom kutu na slići. Svaki broj na položaju (u,v) daje kvantizacijski interval za odgovarajući $F(u,v)$ vrijednost.

Vrijednosti u kvantizacijskoj tablici se temelje na eksperimentalnim procjenama koje su pokazale da su niske frekvencije dominantne u slikama, a ljudski vizualni sustav je više osjetljiv na gubitak informacija u području niskih frekvencija. Sukladno tome, brojevi u nisko-frekvenčnom području (gornji lijevi kut tablice) su manji i povećavaju se kako se krećemo prema visoko-frekvenčnim koeficijentima u ostala tri kuta. Koristeći ovu tablicu možemo izračunati

$$F_Q(u,v) = \begin{bmatrix} F(u,v) \\ Q(u,v) \end{bmatrix}$$

gdje je $Q(u,v)$ pripadna vrijednost u kvantizacijskoj tablici. Izračunate vrijednosti su prikazane u srednjoj tablici.

JPEG karakteristike



Nakon kvantizacije, gotovo sve visoke frekvencije $F_Q(u,v)$ su nula, dok je nekoliko nisko-frekvenčnih vrijednosti ostalo. Za potrebe vrednovanja, također je prikazan proces dekodiranja koji $F_Q(u,v)$ i dekvantizira vrijednosti za izračun $F'(u,v)$.

Gubitak podataka u koeficijentima frekvencija je prilično očit kada se usporede $F(u,v)$ i $F'(u,v)$.

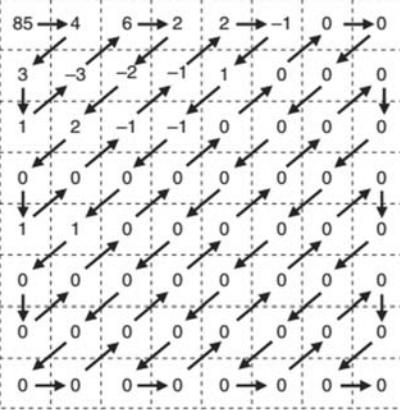
(6) Kvantizirani koeficijenti $F_Q(u,v)$ se zatim kodiraju korištenjem određenog uzorka. DC na poziciji $F_Q(0,0)$ odgovara najvišoj energiji i se obradjuje drugačije u usporedbi s drugim AC koeficijentima viših frekvencija.

DC koeficijenti blokova su kodirani pomoću diferencijalne puls-kodne modulacije (PCM). AC koeficijenti se prvo skeniraju u zig-zag poretku da bi se nul-vrijednosti poredale prema kraju, odnosno time smo dobili visoke frekvencije grupirane prema kraju poruke. To stvara manju entropiju skeniranih AC koeficijenata, koji se kodiraju RLE algoritmom.

(4) Svaki transformacijski koeficijent

Slika $f(x,y)$ prikazuje DCT vrijednosti. DCT transformacija brojevi su najveći u blizini koeficijenta (0,0) i povećavaju se prema stranama. Najveći brojevi su DC koeficijenti.

3D crtež prikazuje odnos



Slika: Zig-zag poredak

DC koeficijent = 85

Niz AC koeficijenata:

4 3 1 -3 6 2 -2 2 0 1 0 -1 -1 2 -1 1 -1 0 1
0 ...

Nakon kvantizacije dok je nekoliko vrednovanja, tako dekvantizira vr

Gubitak podataka usporede F (u,

(6) Kvantiziran određenog uzc i se obrađuje d frekvencija.

DC koeficijenti

1 3 1 3 6 2 2 2 8 1 0 1 1 2 1 1 1 0 1
0 ...

(7) Sljedeći korak je da entropijsko kodiraju DC i AC koeficijenata. Prije entropijskog kodiranja, ovi se koeficijenti pretvaraju u intermedijarne kodove. Za prikaz DC koeficijenta, to je razlika između DPCM vrijednosti tekućeg i prethodnog bloka. Razlika je prikazana kao par koji pokazuje veličinu u bitovima koji se koriste za kodiranje DPCM razlike i amplitudu DPCM razlike.

U našem primjeru, kvantizirana vrijednost DC koeficijenta je 85. Uz pretpostavku da je vrijednost DC koeficijenta prethodnog bloka bila 82, DPCM razlika je 3, koji treba dva bita za kodiranje. Dakle, intermedijarni kod za DC koeficijent je <2><3>.



Uvod u kompresiju

Za AC koeficijente, intermedijarni kod se koristi samo za vrijednosti različite od nule. Svaki AC koeficijent različit od nule je predstavljen parom simbola – kombinacijom duljine niza i veličine, te amplitude AC koeficijenta. Prvi simbol je kombinirana vrijednost duljine niza uzastopnih nula koji nastaju zig-zag poretkom i broja bitova koji se koriste za kodiranje amplitude AC koeficijenta. Drugi simbol kodira amplitudu prvog AC koeficijenta različitog od nule.

U našem slučaju, prvi AC koeficijent različit od nule ima vrijednost 4 i nema nula koje mu prethode, tako da je duljina niza 0. Amplituda iznosa 4 treba za kodiranje 3 bita. Dakle, intermedijarni kod je <0,3><4>. Na slici su prikazani svi intermedijarni kodovi za sve AC koeficijente različite od nule.

dnosti
vljen
ude
za
ji se
dira

ost 4 i
da
<0,3>

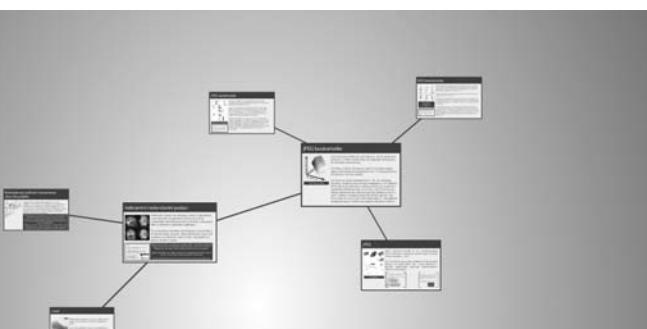
Zapis DC koeficijenta:
Simbol 1 <VELIČINA>
Simbol 2 <AMPLITUDA>

Zapis AC koeficijenta:
Simbol 1 <AMPLITUDA>
<AMPLITUDA>
Simbol 2 <VELIČINA>

Niz intermedijarnih kodova:
<2><3> <0,3><4> <0,2><3> <0,1><1> <0,2><3> <0,3><6>
<0,2><2> <0,2><2> <0,2><2> <1,1><1> <1,1><1> <0,1><-1>
<0,2><2> <0,1><-1> <0,1><1> <0,1><-1> <1,1><1> EOB

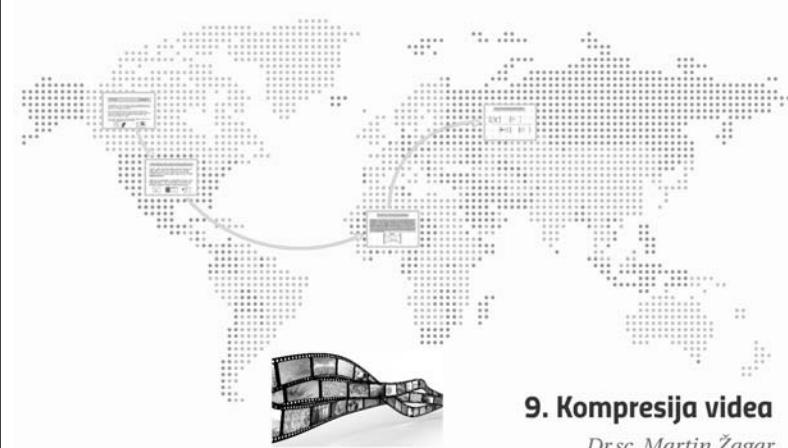
Intermedijarni kod	Binarni zapis prvog simbola (Huffmanov kod)	Binarni zapis drugog simbola (varijabilni cijelobrojni kodovi)
<2><3>	011	11
<0,3><4>	100	100
<0,2><3>	01	11
<0,1><1>	00	1
<0,2><-3>	01	00
<0,3><6>	100	110
<0,2><2>	01	10
<0,2><-2>	01	01
<0,2><2>	01	10
<1,1><1>	11	1
<1,1><-1>	11	0
<0,1><-1>	00	0
<0,2><2>	01	10
<0,1><-1>	00	0
<0,1><1>	00	1
<1,1><1>	11	1
EOB	1010	

Binarni niz:
011111001000111001010010011001100101011011110000011000001000111010



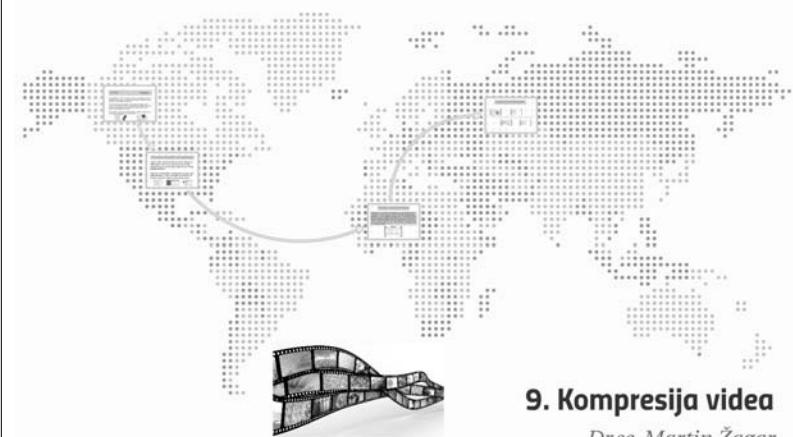
8. Kompresija slika

Dr.sc. Martin Žagar
Multimedijiske arhitekture i sustavi



9. Kompresija videoa

Dr.sc. Martin Žagar
Multimedijiske arhitekture i sustavi



9. Kompresija videa

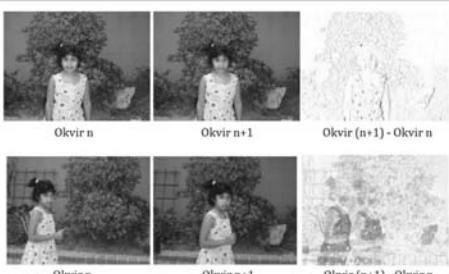
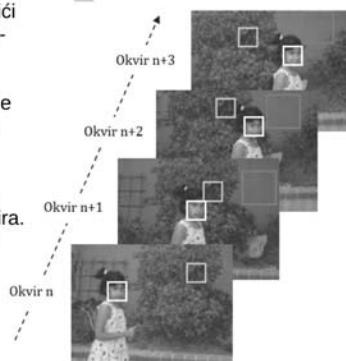
Dr.sc. Martin Žagar
Multimedijks arhitekture i sustavi

ir u video sekvenci

Značajno veći omjer kompresije može se postići uklanjanjem druge vrste redundancije u videu - vremenske redundancije.

Baš kao što su vrijednosti piksela koji čine pozadinu ne razlikuje mnogo od okvira do okvira. Područja koja se mijenjaju su ili objekti koji se pomiču ili se pomiče kamera, ali ti pokreti su kontinuirani i stoga predvidljivi.

Na primjer, u videu se većina piksela koji čine pozadinu ne razlikuje mnogo od okvira do okvira. Područja koja se mijenjaju su ili objekti koji se pomiču ili se pomiče kamera, ali ti pokreti su kontinuirani i stoga predvidljivi.



Gornji niz slika pokazuje dva slikovna okvira iz videa gdje je pozadina gotovo nepromjenjena, a osoba prvom planu se blago pomaknula. Razlikovni okvir (Okvir (n+1) - Okvir n) sadrži mnogo manje informacija nego Okvir n+1. Dekoder u ovom slučaju rekonstruira Okvir n+1 dodavanjem razlikovnog okvira na prethodno rekonstruirani Okvir n.

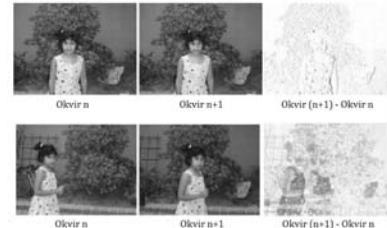
Donji niz slika pokazuje dva slikovna okvira iz videa gdje su prisutni veći pokreti, a također se miče i kamera. Razlikovni okvir ovdje ima mnogo veću entropiju u usporedbi s razlikovnim okviriom u prvom slučaju.

Uvod

U kompresiji slika, analiziramo jedan statičan okvir ispitujući prostornu i spektralnu redundanciju. Ta redundancija postoji jer je slika koherentna struktura, a ne slučajna zbirka piksela.

Budući da je video slijed slikovnih okvira, od kojih svaki ima svoje prostorne redundancije, možemo pretpostaviti da je prilikom kompresije videa potrebno primijeniti neki algoritam za kompresiju slike (npr. JPEG) na svaki slikovni okvir videa.

Ovaj algoritam je poznat kao Motion JPEG (M-JPEG). M-JPEG sažima svaki kadar u video sekvenci kao JPEG sliku.



Postojanje vremenskih zavisnosti rezultira time da razlikovni okvir između dva slikovna okvira sadrži manje podataka od samog slikovnog okvira.

Zato je učinkovitije kodirati video kodiranjem razlikovnih okvira. Primjer razlikovnog okvira prikazan je na slici.

Gornji niz slika pokazuje dva slikovna okvira iz videa gdje je pozadina gotovo nepromjenjena, a osoba prvom planu se blago pomaknula. Razlikovni okvir (Okvir (n+1) - Okvir n) sadrži mnogo manje informacija nego Okvir n+1. Dekoder u ovom slučaju rekonstruira Okvir n+1 dodavanjem razlikovnog okvira na prethodno rekonstruirani Okvir n.

Donji niz slika pokazuje dva slikovna okvira iz videa gdje su prisutni veći pokreti, a također se miče i kamera. Razlikovni okvir ovdje ima mnogo veću entropiju u usporedbi s razlikovnim okviriom u prvom slučaju.

Kad god postoji pokret zbog kretanja nekog objekta, vremenske redundancije mogu se bolje ukloniti predviđanjem kretanja objekta ili određenih blokova između okvira, a ne okvira u cijelini.

Predviđanje na temelju kretanja blokova

Trenutni okvir slike, koji treba biti sažet naziva se **ciljni okvir**. Ciljni okvir se predviđa na temelju kretanja bloka iz prethodnog okvira, koji se naziva **referentni okvir**.

Ciljni okvir je podijeljen u blokove koji se zovu još i **makroblokovi** i svaki makroblok se predviđa na temelju najsličnije regije iz referentnog okvira.



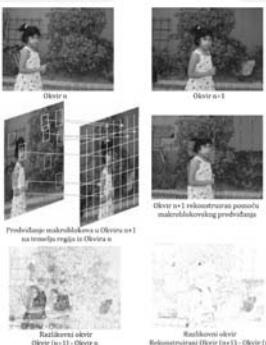
Predviđanje pokreta na temelju makroblokova služi iz dva razloga.

Prvo, vremenska zavisnost se lakše analizira na temelju blokova koji se kreću od okvira do okvira, nego cijelih okvira.



Druge, učinkovitije je podijeliti okvir na blokove i generirati vektor pomaka za svaki blok umjesto vektora pomaka za svaki piksel.

yyijc k tcttci



Predviđanje podrazumijeva pronađenje vektora pokreta (dx , dy) za svaki blok. Kad se pokret može točno predviđeti s vektorom pokreta, regija u okviru n je dobro uskladena i može se koristiti za predikciju odgovarajućeg makrobloka. U takvim slučajevima, razlika između stvarnog i predviđenog bloka je vrlo mala.

To ne mora uvijek biti slučaj zbog različitih razloga: npr. područje okvira n+1 je novo i ne može se točno predviđeti iz okvira n. To je slučaj kad se objekti kreću u ili iz scene u okviru n+1. Na gornjim slikama se makroblokovi koji se odnose na djevojku u prvom planu i neki od središnjih pozadinskih makroblokova mogu dobro predviđjeti te razlike između tih makroblokova imaju malu entropiju.

Međutim područja koja su nova u Okviru n+1 i nisu vidljiva u Okviru n ne mogu biti dobro predviđena na temelju područja iz Okvira n te posljedice imaju veću entropiju. Primjeri takvih makroblokova se mogu vidjeti na donjem desnom dijelu Okvira n+1. Usporedba na dnu slike pokazuje da je sveukupna razlika s predviđanjem pokreta mnogo manja nego bez predviđanja pokreta.

Proces predviđanja ciljnog okvira na temelju referentnog okvira pomoću makroblokova se naziva **kompenzacija pokreta**.

Proces kompresije okvira u ovom načinu rada uključuje sljedeće:

- Pronalaženje najboljeg vektora pomaka za svaki makroblok
- Stvaranje predviđenog okvira ili okvira nastalog kompenzacijom pokreta
- Računanje razlikovnog okvira
- Kompresiju razlikovnog okvira pomoću JPEG algoritma (s gubicima) i vektora pomaka pomoću entropijskog kodiranja (bez gubitaka)

Pronalaženje najboljeg vektora pomaka za makroblok mora biti točno za bolje predviđanje i kompresiju.

Nakon što su vektori pomaka dostupni za svaki makroblok, ciljni okvir može se izračunati pomoću vektora pomaka i referentnog okvira.

Izračun vektora pomaka

U prosjeku, 60% do 80% od ukupnog vremena kodiranja troši se samo na tražnje vektora pomaka. Područje pretraživanja je specifično za svaki algoritam pretraživanja. Cilj traženja vektora pomaka je pronaći najviše podudarno područje u referentnom okviru za ciljni makroblok. Da bi to mogli kvantificirati moramo uvesti mjeru poremećaja koja će biti minimalna za područje koje se najviše podudara.



Mjere poremećaja

Ako definiramo gornji lijevi kut kao ishodište (0,0) makrobloka širine m i visine n. $C(n+1)$ (x, y), gdje x ima vrijednost $[0, m]$ i y ima vrijednost $[0, n]$, definira vrijednosti za svaki piksel u makrobloku.

Pikseli u referentnom okviru koji se razmatraju u odnosu na makroblok iz ciljnog okvira (primjetite da referentni okvir nije podijeljen na blokove) mogu se definirati kao $C_n(x, y)$, gdje x i y imaju isti raspon.

Za sva moguća područja s pomakom (i, j) u odnosu na razmatrani makroblok moramo izračunati srednju apsolutnu razliku (MAD – Mean Absolute Difference) na sljedeći način:

$$MAD(i, j) = \frac{\sum_{p=1}^m \sum_{q=1}^n |C_{n+1}[p, q] - C_n[p + i, q + j]|}{mn}$$

Cilj je naći vektor pomaka (i, j) takav da je pripadni MAD (i, j) minimalan. U prethodnoj formuli koristili smo srednju apsolutnu razliku kao mjeru pronašla najbolje vektora pomaka.

Većina komercijalnih kodera koristi mjeru poremećaja sumu apsolutnih razlika (SAD - Sum of Absolute Differences) zbog niže cijene. Računalno je SAD sličan MAD ali bez dijeljenja sa $(m \cdot n)$.

Algoritmi pretraživanja



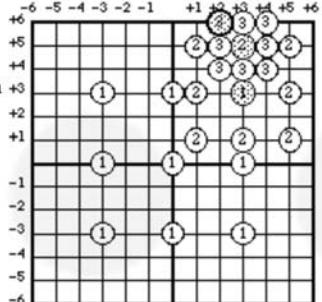
Three Step Search (TSS)

Ovaj algoritam je popularan zbog svoje jednostavnosti i robusnosti i vrlo dobrih performansi. Algoritam ima sljedeće korake:

Korak 1: Početna veličina koraka pretraživanja je 3. Osam blokova s udaljenošću koraka pretraživanja od središta pretraživanja se uspoređuju.

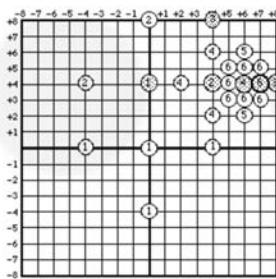
Korak 2: Veličina koraka se smanjuje za 1. Središte se pomiče na mjesto s minimalnom mjerom poremećaja.

Koraci 1 i 2 se ponavljaju dok korak pretraživanja ne bude jednak 1.



Problem koji se može javiti kod TSS algoritma je da koristi ravnomerno razmještene točke provjere u prvom koraku, što može biti neučinkovito za malu procjenu pomaka.

Two Dimensional Logarithmic Search (TDL)



Ovaj algoritam zahtijeva više koraka nego TSS, te je u principu precizniji, pogotovo kada je prozor pretraživanja velik. Algoritam ima sljedeće korake:

Korak 1: Ako je prozor pretraživanja d, onda se početni korak pretraživanja računa kao $2^{\lceil \log_2 d \rceil} - 1$. Uspoređuju se 4 bloka u udaljenosti koraka pretraživanja od središta i središte pretraživanja.

Korak 2: Ako je najmanja mjera poremećaja u središtu, preploviti veličinu koraka. Ako nije, novo središte postaje točka s najmanjom mjerom poremećaja i ponavlja se korak 1.

Korak 3: Kad veličina koraka postane 1, svih devet blokova oko središta se uspoređuju i odabire se onaj s najmanjom mjerom poremećaja.

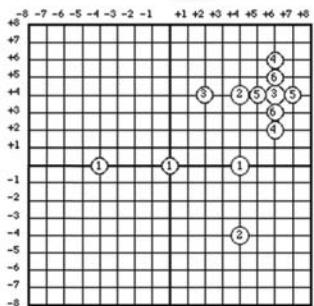
Orthogonal Search Algorithm (OSA)

Ovaj algoritam je hibrid TSS i TDL algoritama. Specifičan je po tome što najprije ispituje horizontalne blokove, a potom blokove u vertikalnom smjeru. Algoritam ima sljedeće korake:

Korak 1: Početna veličina koraka se računa kao $(d + 1) \div 2$, gdje je d veličina prozora pretraživanja. Uspoređuju se dva bloka na udaljenosti od koraka pretraživanja u vodoravnom smjeru od središta (i središte) te novo središte postaje (ili ostaje) blok s najmanjom mjerom poremećaja.

Korak 2: Potom se to isto radi u vertikalnom smjeru.

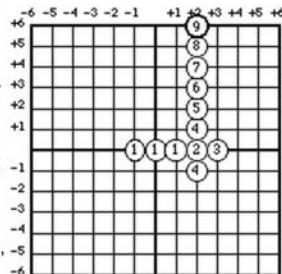
Korak 3: Preploviti veličinu koraka pretraživanja, te ponavljati sve dok je korak pretraživanja veći od 1.



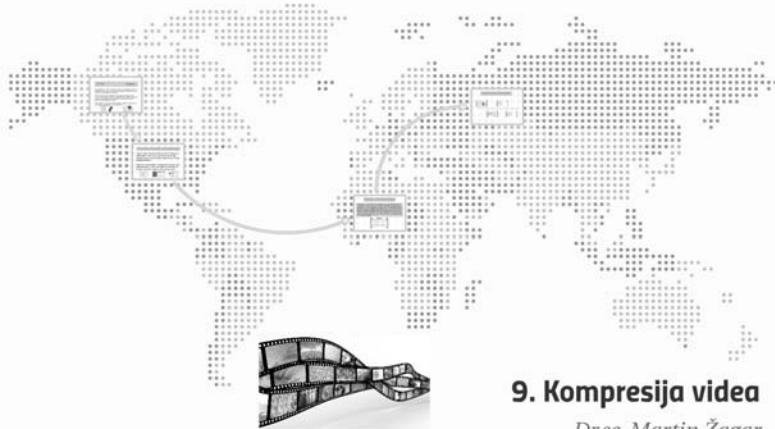
One at a Time Algorithm (OTA)

Ovo je najjednostavniji algoritam pretraživanja optimalnog bloka. Ovaj algoritam pretraživanja zahtijeva vrlo malo vremena, no kvaliteta pretraživanja je najslabija od navedenih algoritama. Algoritam ima sljedeće korake:

Korak 1: Uspoređujemo dva susjedna horizontalna bloka oko središta za pretraživanja i samo središte.



Korak 2: Novo središte postaje blok s najmanjom mjerom poremećaja. Ukoliko je to staro središte pretraživanja, algoritam kreće u vertikalnom smjeru, inače skačemo na korak 1.



9. Kompresija video

Dr.sc. Martin Žagar
Multimedijiske arhitekture i sustavi

Multimedejske arhitekture i sustavi (dio 1)

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

Multimedejske arhitekture i sustavi

1

Multimedejske arhitekture i sustavi

- Nastavnici
 - Prof.dr.sc. Mario Kovač
 - Prof.dr.sc. Hrvoje Mlinarić
 - Doc.dr.sc. Josip Knezović
- Suradnici
 - Dr.sc. Martin Žagar
- Zavod ZARI, Grupa Računalni sustavi i procesi (RASIP), 11. kat
- Administracija: tajnica Blanka Gott (9. kat)

Multimedejske arhitekture i sustavi

2

Organizacija

- Predavanja
- Domaće zadaće
- Međuispit
- Predmet Laboratorij računalnog inženjerstva (LRI) nadovezuje se na gradivo

Multimedejske arhitekture i sustavi

3

Predavanja

- Pokrivaju teorijska znanja i primjere
- Upućuju na znanja koja se trebaju samostalno proučiti
- Komentiraju se i analiziraju različiti pristupi rješavanju problema

Multimedejske arhitekture i sustavi

4

DZ

- Zadaci za samostalno (!! rješavanje radi dubljeg upoznavanja sa specifičnostima pojedinog problema
- Prepostavka da zadaci proširuju građu s predavanja

Multimedejske arhitekture i sustavi

5

MI/ZI

- Pokrivaju teorijsko i praktično znanje s naglaskom na razumijevanje tematike

Multimedejske arhitekture i sustavi

6

Bodovanje

- DZ 25%
- MI 30%
- ZI 45%

- Prolaz: min 50% bodova

- Na pismenom roku računaju se bodovi iz dz

Multimedijске arhitekture i sustavi

7

Literatura

- Prvenstveno dokumentacija s WEB-a
 - JPEG
 - JFIF
 - Xilinx
 - Intel
 - NXP
 - ...
- Materijali dostupni na stranicama predmeta

Multimedijске arhitekture i sustavi

8

MAS

Laboratorij

Multimedijске arhitekture i sustavi

9

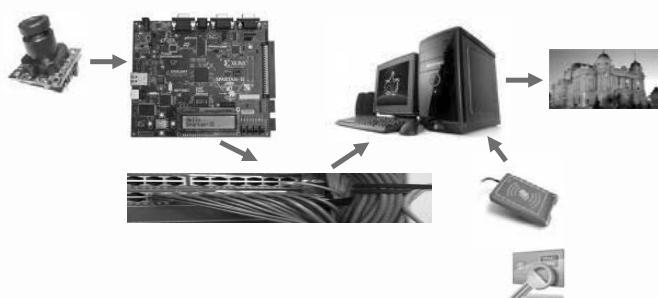
Laboratorij RI - MAS

- Obavijesti o početku i terminima LRI-MAS biti će stavljeni na WEB
- Predavanja LRI-MAS: više u obliku konzultacija vezanih za izvođenje zadatka
- LRI: 2/3 laboratorija izvodi se kod kuće dok će cca. 1/3 (zadnja) biti u okviru FER labosa (zbog licenciranog Intel SW)

Multimedijске arhitekture i sustavi

10

LRI – MAS Projekt



Multimedijске arhitekture i sustavi

11

Dodatno: FRISC3

- Istinski inženjerski problem:
 - FRISC3 .. Implementirana proširenja
 - .. Dodatna proširenja FRISC3 CPU i sustava

Multimedijске arhitekture i sustavi

12

LRI

- Izvode se u GRUPAMA
- Svaka grupa sastoji se od 6(do 7) članova:
 - Voditelj
 - 5 projektnih cjelina čiji raspored izvođenja određuje Voditelj:
 - Čitanje slike s dig. kamere
 - MicroBlaze/FRISC programiranje (kripto, lib)
 - JPEG encoder (lib, jezik C)
 - JPEG decoder (lib, jezik C)
 - PC aplikacija (kriptografija, lib, Smart Card)

Multimedijiske arhitekture i sustavi

13

LRI ocjenjivanje

- Ocjenjuje se REZULTAT projekta uz PRAĆENJE MEĐUREZULTATA u nekoliko točaka tijekom semestra
- Intelektualno vlasništvo
- Projekt mora raditi
 - Voditelj projekta mora prezentirati projekt (i dobre i loše strane)
 - Prezentacija je dio ocjene
 - Projekti će se međusobno uspoređivati ("tržišna utakmica")

Multimedijiske arhitekture i sustavi

14

MAS

Ukratko...

Multimedijiske arhitekture i sustavi

15

Što je to MULTIMEDIJA?

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

Multimedijiske arhitekture i sustavi

16

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.

Multimedijiske arhitekture i sustavi

17

Multimedijiski računalni sustav

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

Multimedijiske arhitekture i sustavi

18

Multimedijski sustavi

■ Interaktivnost

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

■ Sinkronizacija

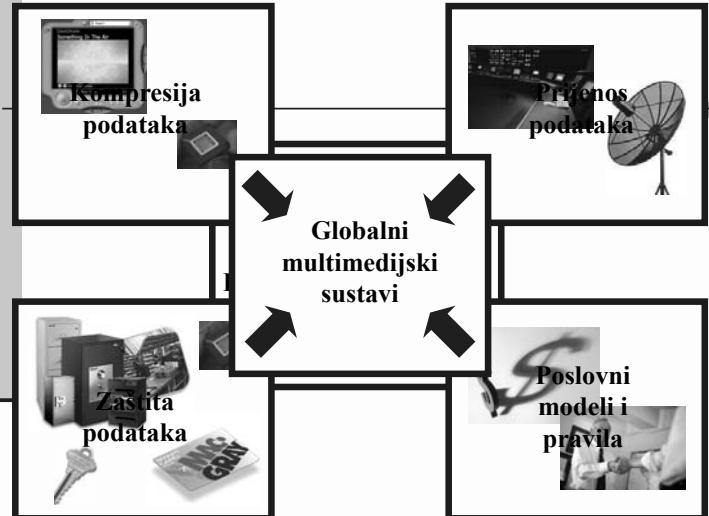
- održavanje vremenskih odnosa između pojedinih dijelova poruke kao i između različitih ali međusobno ovisnih poruka

■ Normizacija

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

Multimedijski arhitekture i sustavi

19



Multimedijski arhitekture i sustavi

20

Multimedijске 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

Multimedijski arhitekture i sustavi

21

Multimedijске arhitekture i sustavi

■ Cilj predmeta

- Sagledavanje problema iz pogleda RAČUNARSTVA
- HW/SW co-design
- Rješavanje problema srednje/visoke kompleksnosti
- Sagledavanje i rješavanje CJELOKUPNOG problema iz stvarne okoline
- Timski rad

Multimedijski arhitekture i sustavi

22

Načelni sadržaj predavanja

- Uvod, Lab, Osnovno
- Kompresija, JPEG, JFIF, transformacije, kodiranje mm podataka x 3
- FPGA, Soft procesori x 3
- Smart Card, kriptografija
- MM ubrzanja (SW, HW,...), SIMD, multicore, GPGPU, performanse
- MM sustavi i aplikacije x 2
- Analize

Multimedijski arhitekture i sustavi

23

MAS

Neke analize i predviđanja

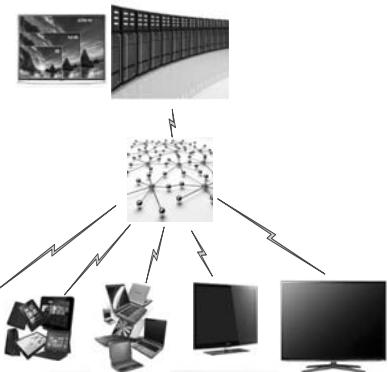
Multimedijski arhitekture i sustavi

24

Global challenges

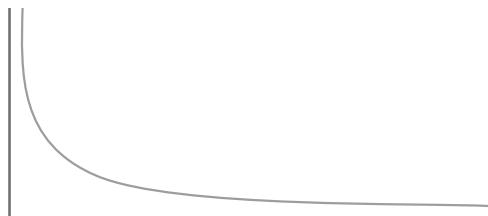
- Annual global Internet protocol traffic will surpass the zettabyte (1000 exabytes) threshold in 2016. Global IP traffic will reach 1.1 zettabytes per year or 91.3 exabytes (one billion gigabytes) per month in 2016.
- By 2018, global IP traffic will reach 1.6 zettabytes per year, or 131.6 exabytes per month.
- Globally, IP video traffic will be 79 percent of all consumer Internet traffic in 2018, up from 66 percent in 2013. This percentage does not include video exchanged through peer-to-peer (P2P) file sharing.
- The sum of all forms of video (TV, video on demand [VoD], Internet, and P2P) will be in the range of 80 to 90 percent of global consumer traffic by 2018.

Plethora of novel devices



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*



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..

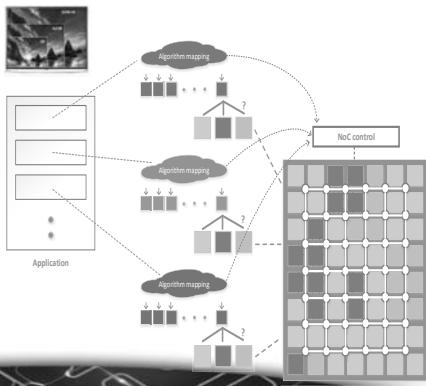
- Another important application case is medical imaging
- Challenges are related to huge medical image sizes that need to be processed in short periods of time to allow novel and advanced medical procedures or medical consultations
- Delivery of medical imaging to various locations and device types shows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

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

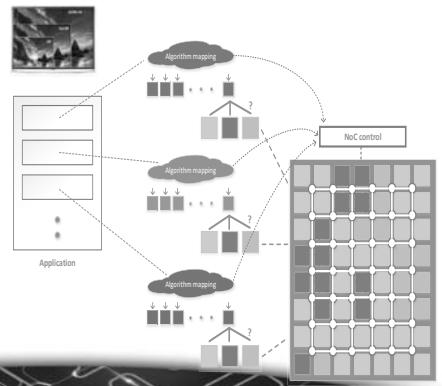
Research directions

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



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



MAS

Važni preuvjeti

Norme

- Pogledajte: www.hzn.hr
- Norme NISU besplatne
- Norme su potrebne
- Norme na područu MM aplikacija znaju imati česte promjene

Norme

Službena definicija:

- dokument donesen konsenzom i odobren od priznatoga tijela, koji za opću i višekratnu uporabu daje pravila, upute ili značajke za djelatnosti ili njihove rezultate s ciljem postizanja najboljeg stupnja uređenosti u danome kontekstu
NAPOMENA: Norme bi se trebale temeljiti na provjerениm znanstvenim, tehničkim i iskustvenim rezultatima, i biti usmjerene promicanju najboljih prednosti za društvo.

Intelektualno vlasništvo

- Autorsko djelo
- Patent
- Žig
- Topografija poluvodičkih proizvoda
- Pogledajte: www.dziv.hr

IV

- Na IV se zasniva cijelokupna industrija
- U okviru ovog predmeta zahtjevati će se poštivanje IV
- "na Internetu se sve može naći"....
- Pravni i ekonomski rizici

Multimedejske arhitekture i sustavi

37

Multimedejske arhitekture i sustavi (dio 2)

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

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 $P(A)=0$ $i(A)=\infty$

Multimedejske arhitekture i sustavi

2

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

Multimedejske arhitekture i sustavi

3

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

Multimedejske arhitekture i sustavi

4

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 doznati

Multimedejske arhitekture i sustavi

5

MAS – Osnove kompresije

- #### ■ Kompresija podataka:



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

MAS – Entropijsko kodiranje

- #### ■ Entropijsko kodiranje:

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



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

- #### ■ 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

MAS – Osnove kompresije: RLE kodiranje

- Zasniva se na uzastopnom ponavljanju jednoq simbola

- #### ■ Jednostavna izvedba

- #### ■ Primjer:

- linije očitane sa dokumenta u fax uređaju
1 linija (75 dpi, 8")=600 bita=75 B

17.24.3.211.22.188.77.54.4 = 9 B

MAS – Osnove kompresije

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



Multimedejske arhitekture i sustavi

12

Modeli

- Da li možemo na neki način još više ‘smanjiti’ vrijednost entropije? (entropija je za izabrani izvor uvijek 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)

Multimedejske arhitekture i sustavi

13

Osnovni modeli

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

Multimedejske arhitekture i sustavi

14

Osnovni modeli

- Vjerovatnosni 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 vjerovatnost pojavljivanja ista
 - statistički model: na neki način izračunati učestalost pojavljivanja simbola te na temelju toga definirati vjerovatnosti
- Problem: mogućnost POVEĆANJA ENTROPIJE!!

Multimedejske arhitekture i sustavi

15

Osnovni modeli

- Markovljev model (A.A. Markov 1856-1922)
 - ovim modelom opisuje se izvor ‘s pamćenjem’ tj izvor kod kojeg vjerovatnost 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!!

Multimedejske arhitekture i sustavi

16

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 konkretne primjere za većinu prije navedenih modela

Multimedejske arhitekture i sustavi

17

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

Multimedijске arhitekture i sustavi

18

MAS

Algoritam za kompresiju slike

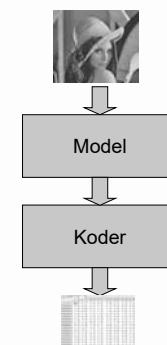
Model za slikovne podatke

- 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

Multimedijске arhitekture i sustavi

20

Arhitektura sustava za kompresiju slika/videa



Multimedijске arhitekture i sustavi

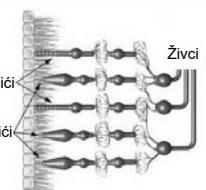
21

Ljudski vizualni sustav

HVS model

Model zasnovan na našem vizualnom sustavu

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



Multimedijске arhitekture i sustavi

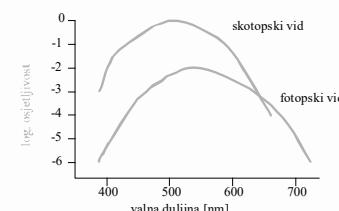
23

HVS – spektralna osjetljivost

- štapići
 - osjetljivi na svjetlo i pri niskim razinama luminancije ispod 1cd/m² ("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 1cd/m² i 100cd/m² aktivni su i štapići i čunjići (fotopski vid)
 - pri razinama luminancije većim od 100 cd/m² štapići postaju zasićeni i aktivni su samo čunjići

Multimedejske arhitekture i sustavi

24



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

ZAKLJUČAK #1:

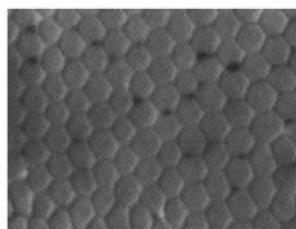
Znatno smo osjetljiviji na promjene u luminanciji nego u boji

Multimedejske arhitekture i sustavi

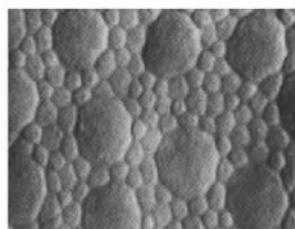
25

HVS – prostorna osjetljivost

- Gustoća receptora



(a) Fovea



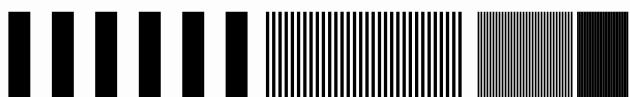
(b) Periphery

Multimedejske arhitekture i sustavi

26

HVS – prostorna osjetljivost

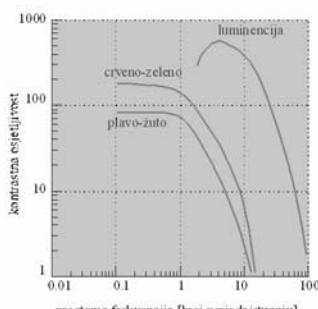
- Zbog konačne fizičke gustoće receptora, oko može razaznati detalje ako su zrake svjetlosti pod kutem upada većim od cca 1 minute
- Prostorna frekvencija



Multimedejske arhitekture i sustavi

27

HVS – prostorna osjetljivost



ZAKLJUČAK #2:

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

Multimedejske arhitekture i sustavi

28

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

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 !!!

Multimedijiske arhitekture i sustavi

30

Promjena prostora boja

RGB ↔ YUV

Promjena prostora boja

- Ulazni podaci: uglavnom RGB
- Prikidan 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)

Multimedijiske 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:

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

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

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

Multimedijiske arhitekture i sustavi

33

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

Multimedijiske arhitekture i sustavi

34

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,...

Multimedijiske arhitekture i sustavi

35

Transformacija

DCT

Transformacija

- Prebacivanjem podataka u frekvencijsku domenu želimo dobiti informacije o frekvencijskim karakteristikama svake komponente
- Koju transformaciju izabratи?

 - 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)

Multimedijiske arhitekture i sustavi

37

Transformacija: DCT

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

Multimedijiske arhitekture i sustavi

38

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

$f(i,j)$ - amplitudo 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 $u = 1,2,\dots,M-1, v = 1,2,\dots,N-1$

Multimedijiske arhitekture i sustavi

39

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)

Multimedijiske arhitekture i sustavi

40

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

Multimedijiske arhitekture i sustavi

41

Opseg podataka

- DCT dovodi do proširenja opsega podataka (dinamički opseg za 8×8 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)

Multimedejske arhitekture i sustavi

42

Pomak kod DCT

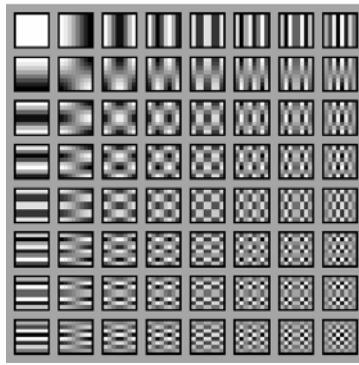
- Slikovni podaci na ulazu su pozitivni cijeli brojevi (npr. 0-255)
- S obzirom da je DCT definirana i za pozitivno i negativno područje na ovaj način bi se polovica ulaznog prostora izgubila (a time i znatno smanjila efikasnost)
- Zato se prije DCT sve vrijednosti na ulazu translatiraju za polovicu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

Multimedejske arhitekture i sustavi

43

2D-DCT bazne funkcije

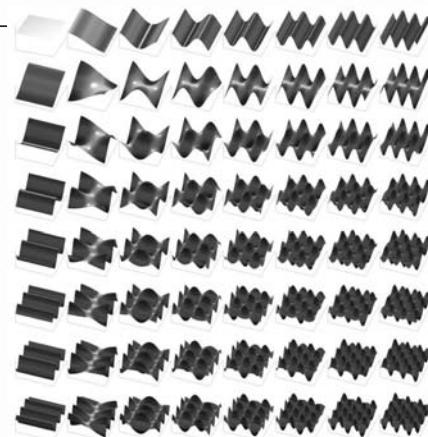
bijelo pozitivne vrijednosti, crno negativne vrijednosti



Multimedejske arhitekture i sustavi

45

Prikaz u 3D

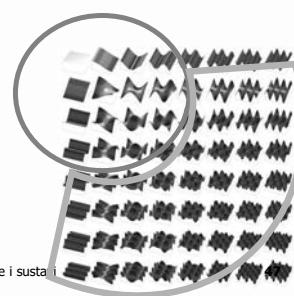


Multimedejske arhitekture i sustavi

46

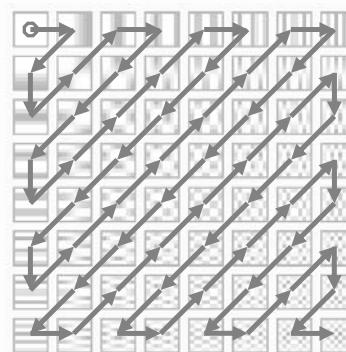
Što smo dobili sa DCT?

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



Multimedejske arhitekture i sustavi

Cik-cak (Zig-zag) reorganizacija



Multimedejske arhitekture i sustavi

48

Kako je to na stvarnoj slici....



Multimedijiske arhitekture i sustavi

49

Kako je to na stvarnoj slici....

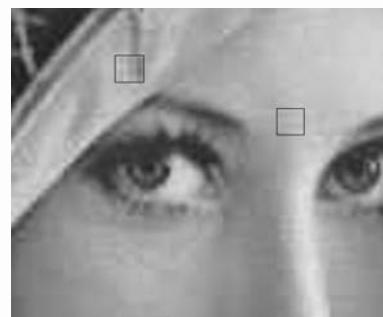


Multimedijiske arhitekture i sustavi

50

Primjer

■ Primjer bloka



Multimedijiske arhitekture i sustavi

51

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 !)

Multimedijiske arhitekture i sustavi

52

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):
 - Uzalni 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 !!!

Multimedijiske arhitekture i sustavi

54

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:

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

Multimedejske arhitekture i sustavi

55

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

Multimedejske arhitekture i sustavi

56

Tipične kvantizacijske tablice

Table K.1 – Luminance quantization table

16	11	10	16	24	40	51	61
12	12	14	19	26	38	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

Multimedejske arhitekture i sustavi

57

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

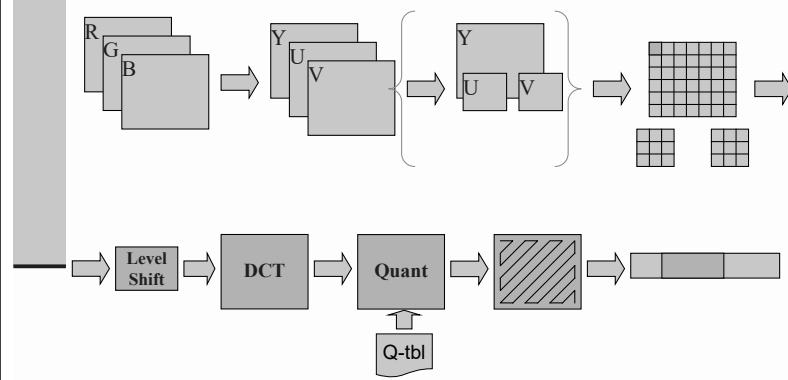
Multimedejske arhitekture i sustavi

58

JPEG koderski model

Pregled svih koraka

JPEG koderski model



Multimedejske arhitekture i sustavi

59

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)



Multimedijiske arhitekture i sustavi

61

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

Multimedijiske arhitekture i sustavi

62

DZ1

- Prije nego pređemo 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.ppt](#)

Multimedijiske arhitekture i sustavi

63

Statistički/entropijski koder

Od čega krećemo

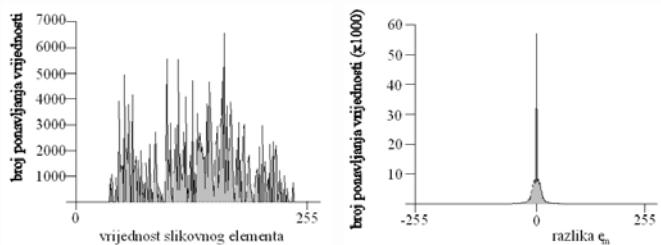
- Entropijski koder prima kvantizirane koeficijente presložene po cik-cak algoritmu u nizove od po 64 koeficijenta za svaku komponentu bloka
- Koje karakteristike možemo uočiti? (koristite DZ i proučite resultantne vrijednosti za nekoliko slika)

Multimedijiske arhitekture i sustavi

65

DC koeficijenti

- Podloga: analiza statistike DC komponenata



Multimedijiske arhitekture i sustavi

66

DC koeficijenti

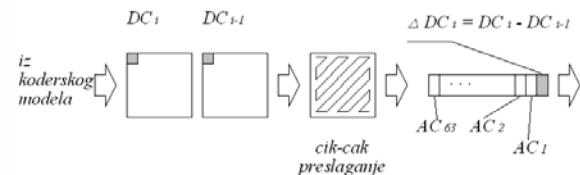
- Teorijska podloga: DC koeficijent predstavlja srednju vrijednost svih 64 elementa bloka
 - dva susjedna bloka obično imaju vrlo slične srednje vrijednosti
- Rezultat:
 - Distribucija DC vrijednosti je prilično jednolika i nepogodna za kodiranje
 - Distribucija RAZLIKE dva susjedna DC koeficijeta je vrlo gusta oko nule i idealna za kodiranje

Multimedejske arhitekture i sustavi

67

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...)



Multimedejske arhitekture i sustavi

68

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

Multimedejske arhitekture i sustavi

69

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 Huffmannovog 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 zamijenjeni 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 obrađuje 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.

Multimedejske arhitekture i sustavi

70

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.

Multimedejske arhitekture i sustavi

71

Kategorija

Kategorija	DC razlika	AC koeficijent
0	0	-1,1
1	-1,1	-1,1
2	-3,-2,2,3	-3,-2,2,3
3	-7,-4,4,-7	-7,-4,4,-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	

Multimedejske arhitekture i sustavi

72

Kategorija

- Neke vrijednosti se rijetko pojavljuju pa nije imalo smisla svakom broju pridjeljivati kod
- Kategorije su određene prema vjerovatnosti 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)

Multimedijijske arhitekture i sustavi

73

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

Multimedijijske arhitekture i sustavi

74

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).

Multimedijijske arhitekture i sustavi

75

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

Multimedijijske arhitekture i sustavi

76

Huffman-ova tablica za AC simbole

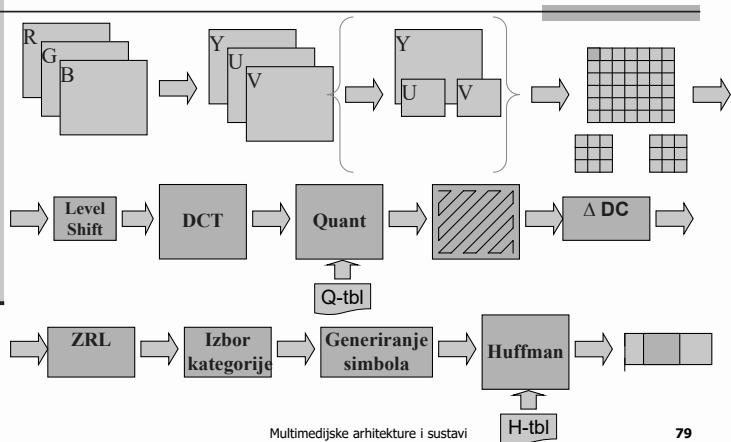
Duljina niza/Kategorija	Duljina koda	Kodna riječ
0/0 (EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/10	16	11111111110000011
1/1	4	1100
1/2	5	11011
...

Multimedijijske arhitekture i sustavi

77

JPEG koder

JPEG koder



JPEG grayscale

Potpuni primjer kodiranja jednog bloka

Pomak (level shift)

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

Multimedejske arhitekture i sustavi

81

2D DCT, kvantizacija

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

Multimedejske arhitekture i sustavi

82

Cik-cak + entropijsko kodiranje

-2 -8 1 -1 0 0 -1 0 1 0 0 0 0 0 -1 00

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

01101101101110010001110001100111101101010

Multimedejske arhitekture i sustavi

83

JPEG+JFIF

Kako je zapisana slika u računalu

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

Multimedijiske arhitekture i sustavi

86

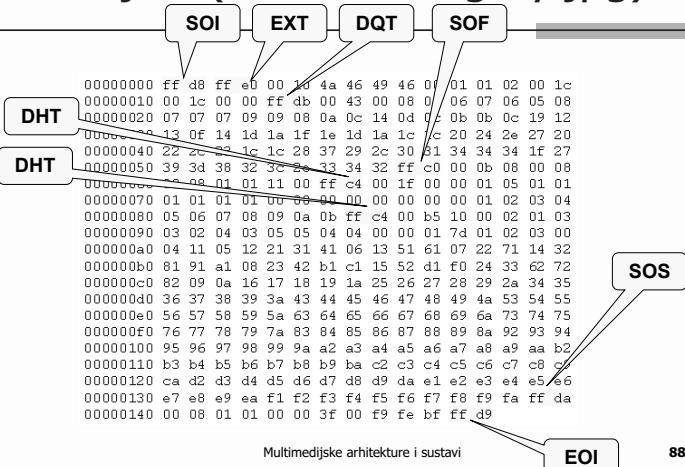
Minimalna struktura JPEG datoteke

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

Multimedijiske arhitekture i sustavi

87

Primjer: (test1_crno_gray.jpg)



Multimedijiske arhitekture i sustavi

88

Multimedijiske arhitekture i sustavi

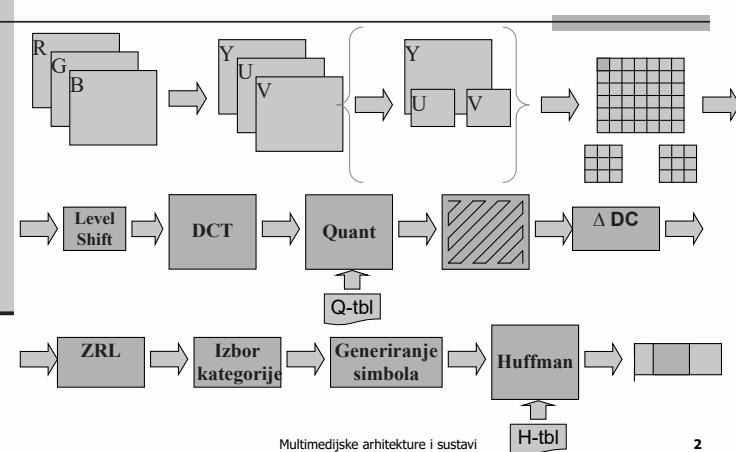
prof.dr.sc. Mario Kovač
izv.prof.dr.sc. Hrvoje Mlinarić



Multimedijiske arhitekture i sustavi

1

JPEG koder



Multimedijiske arhitekture i sustavi

2

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...

Multimedejske arhitekture i sustavi

3

Količine podataka

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

Rezolucija slike	Bita/piksela	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

Multimedejske arhitekture i sustavi

4

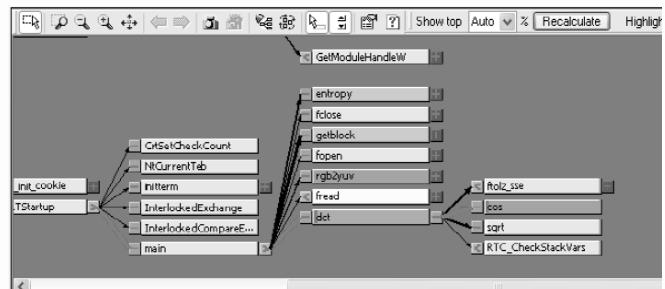
Najzahtjevnije operacije

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

Multimedejske arhitekture i sustavi

5

I sada počinje ozbiljna analiza..



Multimedejske arhitekture i sustavi

6

Kako to implementirati u SW ili HW

- Najzahtjevnije operacije pri obradi slike/videa
 - Procjena/predviđanje pokreta !!!!!....!!!
 - DCT
 - RGB-YUV transformacija
 - Entropijsko kodiranje

Multimedejske arhitekture i sustavi

7

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

Multimedejske arhitekture i sustavi

8

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-YUV 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)

Multimedijiske arhitekture i sustavi

10

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
- Ako napravimo jednostavan program za test ([Primjer1.exe](#)) možemo dobiti vrlo jednostavnu usporedbu efikasnosti:
 - RGB2YUV v.1. AVG ≈ 160

11

Konverzija prostora boja

- Verzija 2: osnovna aproksimacija

$$Y = 0.299 R + 0.587 G$$

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

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

- RGB2YUV v.2. AVG ≈ 138

- Verzija 3: poboljšana aproksimacija

$$\text{data}[i\text{++}] = (\text{byte})(R/4 + G/2);$$

$$\text{data}[i\text{++}] = (\text{byte})(-(G/4) + B/2 + 128);$$

$$\text{data}[i\text{++}] = (\text{byte})(R/2 - (G/2) + 128);$$

- RGB2YUV v.3. AVG ≈ 35!!!!

- Ako se napiše u asembleru: za red veličine brže

Multimedijiske arhitekture i sustavi

12

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

13

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,U,V) za jednu testnu sliku za dva aproksimativna rješenja

Multimedijiske arhitekture i sustavi

13

Multimedijiske arhitekture i sustavi

14

Načini optimizacije algoritama

- Vidjeli smo prvi primjer jedne jednostavne metode za optimizaciju nekog algoritma
- Iako se može ciniti 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

Multimedijiske arhitekture i sustavi

15

Načini optimizacije algoritama

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

Multimedijiske arhitekture i sustavi

16

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 multimedijiskih algoritama

Multimedijiske arhitekture i sustavi

17

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

Multimedijiske arhitekture i sustavi

18

JPEG

- RGB2YUV:
 - 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

Multimedijiske arhitekture i sustavi

19

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...

Multimedijiske arhitekture i sustavi

20

JPEG - DCT

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

- 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

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

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

JPEG - DCT

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

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

JPEG – 2D-DCT odvojivost

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

- 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

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).
- Vojne primjene

1D DCT

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

- 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 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 α() parametrom.
- Samo ovim postupkom ZNAČAJNO smo smanjili kompleksnost računanja
- No i sa tako izvedenim cos() kompleksnost je velika

Multimedijiske arhitekture i sustavi

27

2D DCT

- 2D DCT za blok 8x8:

- 4096 množenja, 4032 zbrajanja
- Samo za jednu sliku 1280x1024
- 83886080 množenja, 82575360 zbrajanja

Multimedijiske arhitekture i sustavi

28

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

Multimedijiske arhitekture i sustavi

29

2D DCT

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

Multimedijiske arhitekture i sustavi

30

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

Multimedijiske arhitekture i sustavi

31

- Na taj način mogu se identificirati elementi:

- C_k
- S_k
- s_{jk} i d_{jk}

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

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

$$C_1 = S_7 = 0.9808$$

$$C_2 = S_6 = 0.9239$$

$$C_3 = S_5 = 0.8315$$

$$C_4 = S_4 = 0.7071$$

$$C_5 = S_3 = 0.5556$$

$$C_6 = S_2 = 0.3827$$

$$C_7 = S_1 = 0.1951$$

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

Multimedijiske arhitekture i sustavi

32

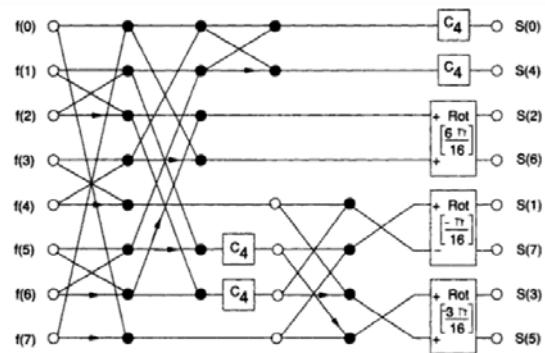
Ligtenberg, Vetterli

- 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 $\pi/16$ tada nove vrijednosti iznose:
 - $X=Ck^*x+Sk^*y$
 - $Y= -Sk^*x+Ck^*y$

Multimedejske arhitekture i sustavi

33

Ligtenberg, Vetterli



Multimedejske arhitekture i sustavi

34

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.

Multimedejske arhitekture i sustavi

35

Brzi DFT....

- Bitno drugačiji pristup računaju DCT predložili su neki autori koji su se intenzivno bavili i proučavanjem brzih algoritama za računanje diskretnе 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.

Multimedejske arhitekture i sustavi

36

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.

Multimedejske arhitekture i sustavi

37

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

Multimedejske arhitekture i sustavi

38

KR algoritam

Korak 1:

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

Korak 2:

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

Korak 3:

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

Korak 4:

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

..... i tako dalje

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

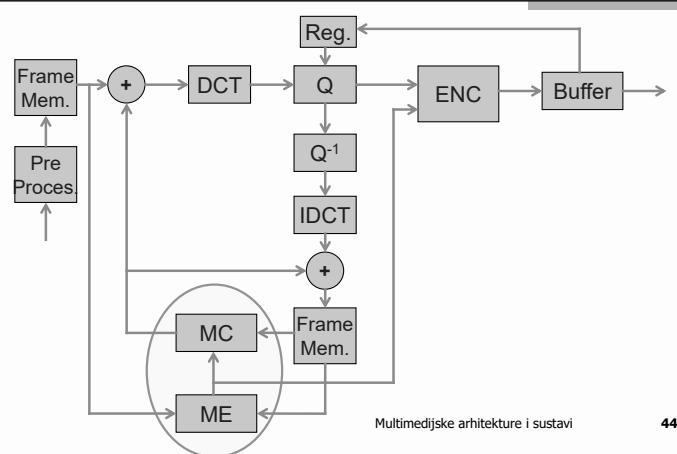
Zadatak za vježbu (neobavezno)

■ 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)

Multimedijiske arhitekture i sustavi

45

Međublokova kompresija

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



Multimedijiske arhitekture i sustavi

46

Vremenska redundancija

CARJUMP

Multimedijiske arhitekture i sustavi

47

AUTO UZ

Multimedijiske arhitekture i sustavi

48

Vremenska redundancija

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

Vremenska redundancija

BRZI AUTO

Multimedijiske arhitekture i sustavi

49

Multimedijiske arhitekture i sustavi

50

Procjena i kompenzacija pokreta

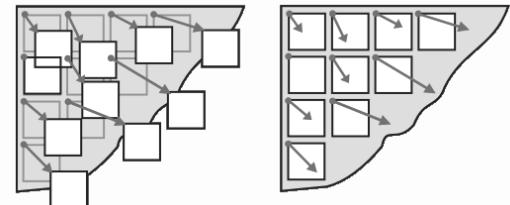
- Kako bi omogućili jednostavniju implementaciju nameću se potrebna pojednostavljenja:
 - Segmentacija na pravokutne dijelove unaprijed znanih dimenzija (blok)
 - Svaki je pokret translacijski (vektor pomaka)

Multimedijiske arhitekture i sustavi

51

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



Multimedijiske arhitekture i sustavi

52

Procjena i kompenzacija pokreta

CITY VIEW

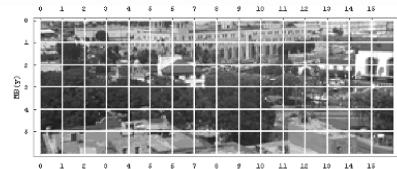
PROCJENA

Multimedijiske arhitekture i sustavi

53

Slika kao pojedinačni objekt pri kompresiji

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



Multimedijiske arhitekture i sustavi

54

Mjera poremećaja

- Mjera sličnosti dvaju blokova jest **mjera poremećaja** slike od jednog trenutka do drugog
- Procjena pokreta svodi se na **optimizaciju** mjere 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

Multimedijiske arhitekture i sustavi

55

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)

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

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

Multimedijiske arhitekture i sustavi

56

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**

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:
 - $(2^8+1)*(2^8+1)*(256+\text{dohvat})=73984 + \text{dohват}$

- Pronalaženje minimuma

- Za jednu sliku 1280x1024
 - $5120*73984=378.798.080 + \text{dohват}$

- Za 30 slika u sekundi: $1,1*10^{10}$ zbrajanja/s

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 blizu potpunom pretraživanju
 - Problemi sa tipovima pokreta (objašnjeno kasnije)

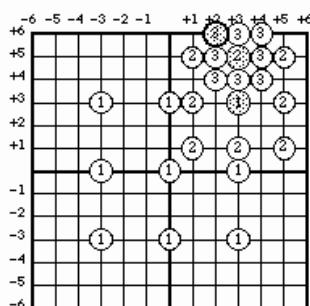
Brzi algoritmi

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

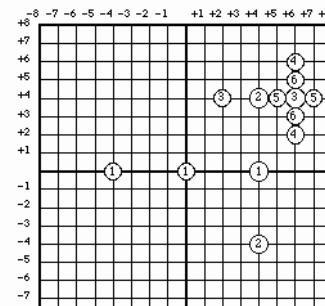
Neki primjeri:

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

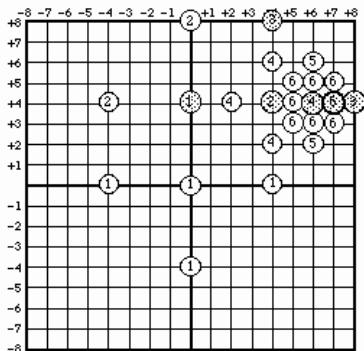
Algoritmi – primjer (3SS)



Algoritmi – primjer (ORT)



Algoritmi – primjer (LOG)



Multimedijiske arhitekture i sustavi

63

Algoritmi – primjer (LOG)

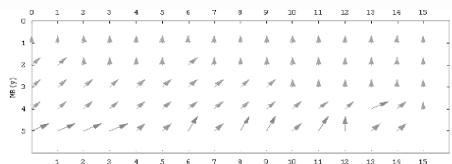
LOG

Multimedijiske arhitekture i sustavi

64

Raspodjela vektora pomaka

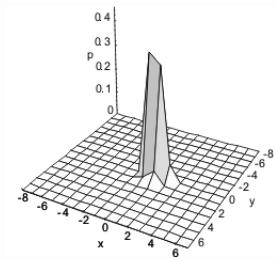
- Prikaz upućuje na ravnomjerno raspoređenepokrete između dvije slike sekvence
- Vektori su većinom kratki – centrirana raspodjela vektora - tipična za sekvence iz stvarnog svijeta



65

Raspodjela vektora (2)

- Prebrojavanje vektora za sve moguće parove komponenti (x,y) tijekom cijele sekvence 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

66

Primjeri usporedbe algoritama

- Parametri koji određuju učinkovitost algoritma:
 - Ukupna **mjera poremećaja** nakon procjene pokreta treba biti što manja (bolja kompresija)
 - Ukupni broj **ispitnih točaka** također treba biti što manji (veća brzina)
- Potreban je kompromis **kompresija \leftrightarrow 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

Multimedijiske arhitekture i sustavi

67

Implementacija – primjer

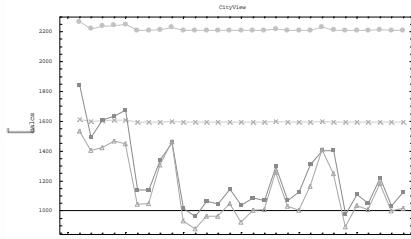
- Norma za kompresiju **ne specificira** postupke procjene pokreta
- Kvaliteta aplikacije znatno ovisi o načinu procjene pokreta

Multimedijiske arhitekture i sustavi

68

Rezultati (1)

- "City View" sekvenca – broj ispitnih točaka:



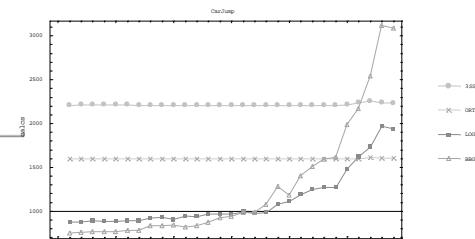
- Centrirana raspodjela: BBGDS najbolji
- ORT i 3SS konstantan i relativno velik broj ispitnih točaka

Multimedijiske arhitekture i sustavi

69

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

Multimedijiske arhitekture i sustavi

70

Rezultati (3)

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

Multimedijiske arhitekture i sustavi

71

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

Multimedijiske arhitekture i sustavi

72

Optimizacije

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

Multimedijiske arhitekture i sustavi

73

Osnovni načini SW podrške za MM



Multimedijiske arhitekture i sustavi

74

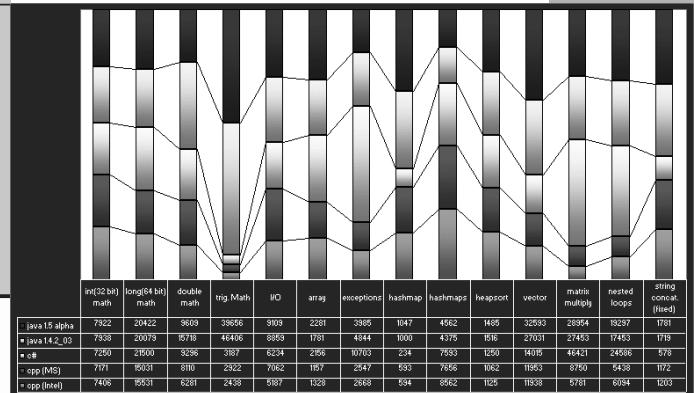
Performanse

- Morate biti svjesni da se program pisan u Javi ili npr. C# sporije izvodi od programa koji su prevedeni u kod za ciljni procesor
- Dodatno: viši programski jezici generiraju sporiji kod od onoga pisanih u kombinaciji sa asemblerom

Multimedijiske arhitekture i sustavi

75

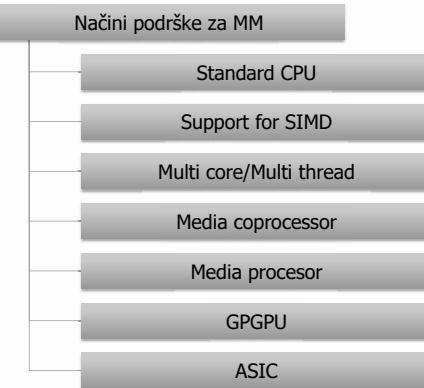
Usporedba Java, C#, C++



www.tommi-systems.de/main-Dateien/reviews/languages/benchmarks.html

76

Osnovni načini HW podrške za MM



Multimedijiske arhitekture i sustavi

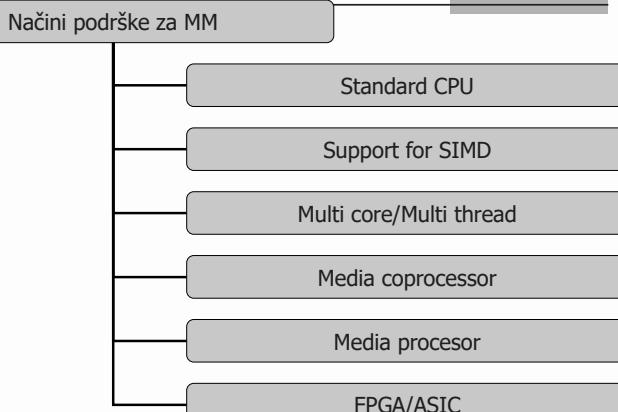
77

Multimedijiske arhitekture i sustavi (dio 4)

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



Osnovni načini CPU podrške za MM



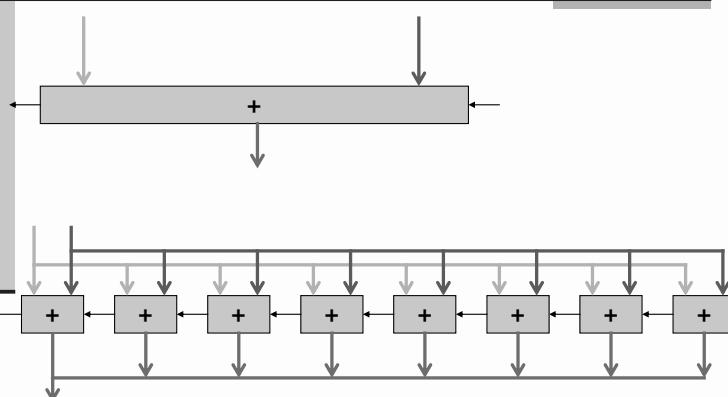
Standardni CPU

- 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



- AVX-512 instruction are encoded with the new EVEX prefix. It allows 4 operands, 7 new 64-bit opmask registers, scalar memory mode with automatic broadcast, explicit rounding control, and compressed displacement memory addressing mode. The width of the register file is increased to 512 bits and total register count increased to 32 (registers ZMM0-ZMM31) in x86-64 mode.

AVX-512 Subset	F	CD	ER	PF	VL	BW	DQ	IFMA	VBMI
Xeon Phi x200 co-processor (2016)	Yes	Yes	Yes	No					
Skylake EP/EPX Xeon "Purley" processors (expected in H2 2017)	Yes	Yes	Yes	No	Yes				No
Cannonlake processors (expected in 2017)			Yes	No			Yes		

Podrška za SIMD

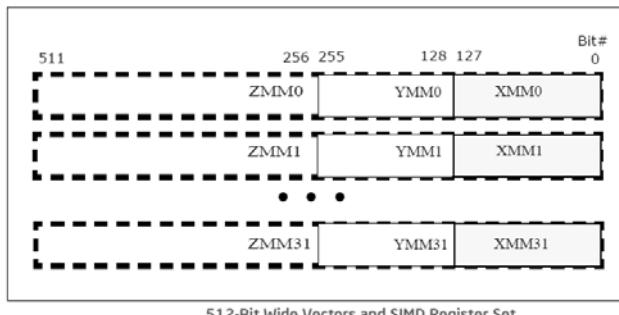
- SIMD (Single Instruction Multiple Data)
 - Arhitektura puta podataka u procesoru koja omogućuje obradu više podataka (u načelu manje preciznosti) sa jednom naredbom
- Osnovna ideja:
 - Ako imamo npr 64 bitovnu ALU onda je potpuno neefikasno s njom obrađivati 8 bitovne podatke
 - Reorganizirati ALU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

MM proširenja

- Ovo (osnovno) načelo služi kako bi se obrada multimedijiskih algoritama značajno ubrzala
- Primjeri:
 - Prvi: VIS, PA-RISC
 - Stari: MMX, 3DNow!
 - Ne tako stari: SSE4
 - Novi: AVX, AVX2, NEON (ARM)
 - Najnoviji: 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/SSE2.1 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.

AVX-512



Multimedjiske arhitekture i sustavi

10

Primjer: VDBPSADBW

VDBPSADBW—Double Block Packed Sum-Absolute-Differences (SAD) on Unsigned Bytes

Opcode/ Instruction	Op / En	64/32 bitMode	CPuid Feature Support	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.

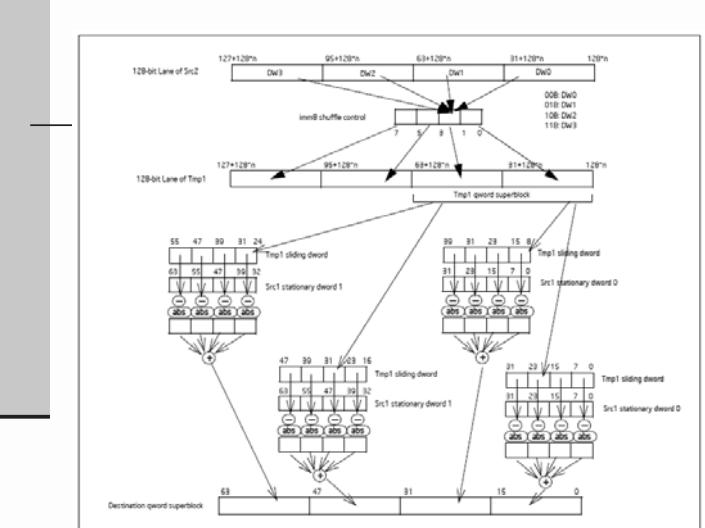
Within each super block of packed word results, the SAD results from two 32-bit dword elements are calculated as follows:

- 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 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 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 PSHUFD.

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.

Multimedjiske arhitekture i sustavi

12



13

Operation

VDBPSADBW (EVEX encoded versions)

(KL, VL) = (8, 128), (16, 256), (32, 512)

Selection of quadruplets:

```
FOR I=0 TO VL step 128
    TMP1[+3:1] <- select(SRC2[+127:I], imm8[1:0])
    TMP1[+63:I+32] <- select(SRC2[+127:I], imm8[3:2])
    TMP1[+95:H64] <- select(SRC2[+127:I], imm8[5:4])
    TMP1[+127:I+96] <- select(SRC2[+127:I], imm8[7:6])
END FOR
```

SAD of quadruplets:

```
FOR I=0 TO VL step 64
    TMP_DEST[+15] <- ABS(SRC1[+I:7] - TMP1[+I:7]) +
        ABS(SRC1[+I+8] - TMP1[+I+8]) +
```

```
ABS(SRC1[+I+23:I+16] - TMP1[+I+23:I+16]) +
    ABS(SRC1[+I+31:I+24] - TMP1[+I+31:I+24])
```

```
TMP_DEST[+I+16] = ABS(SRC1[+I:7] - TMP1[+I+16]) +
    ABS(SRC1[+I+15:I+9] - TMP1[+I+23:I+16]) +
    ABS(SRC1[+I+23:I+16] - TMP1[+I+31:I+24]) +
    ABS(SRC1[+I+31:I+24] - TMP1[+I+39:I+32])
```

```
TMP_DEST[+I+47:I+32] = ABS(SRC1[+I+31:I+32] - TMP1[+I+23:I+16]) +
    ABS(SRC1[+I+47:I+40] - TMP1[+I+31:I+32]) +
    ABS(SRC1[+I+55:I+48] - TMP1[+I+47:I+40]) +
    ABS(SRC1[+I+63:I+56] - TMP1[+I+55:I+48])
```

ENDIF

FOR J=0 TO KL-1

I <= J*16

IF k1[j] OR "no writemask"

THEN DEST[+15:j] <- TMP_DEST[+15:j]

ELSE

IF "merging-masking" ; merging-masking

THEN DEST[+15:j] remains unchanged*

ELSE ; zeroing-masking

DEST[+15:j] <- 0

FI

ENDFOR

DEST[MAX_VL-1:VL] <- 0

Multimedjiske arhitekture i sustavi

14

Intel C/C++ Compiler Intrinsic Equivalent

```
VDBPSADBW_mm512_mm512_dbsad_epu8_mm512_a_mm512_b;
VDBPSADBW_mm512_mm512_mm512_mm512_dbsad_epu8_mm512_a_mm512_b;
VDBPSADBW_mm512_mm512_mm512_mm512_dbsad_epu8_mm512_a_mm512_b;
VDBPSADBW_mm256_mm256_dbsad_epu8_mm256_a_mm256_b;
VDBPSADBW_mm256_mm256_mm256_mm256_dbsad_epu8_mm256_a_mm256_b;
VDBPSADBW_mm256_mm256_mm256_mm256_dbsad_epu8_mm256_a_mm256_b;
VDBPSADBW_mm128_mm128_dbsad_epu8_mm128_a_mm128_b;
VDBPSADBW_mm128_mm128_mm128_dbsad_epu8_mm128_a_mm128_b;
VDBPSADBW_mm128_mm128_mm128_mm128_dbsad_epu8_mm128_a_mm128_b;
```

Multimedjiske arhitekture i sustavi

15

Multicore/multithread

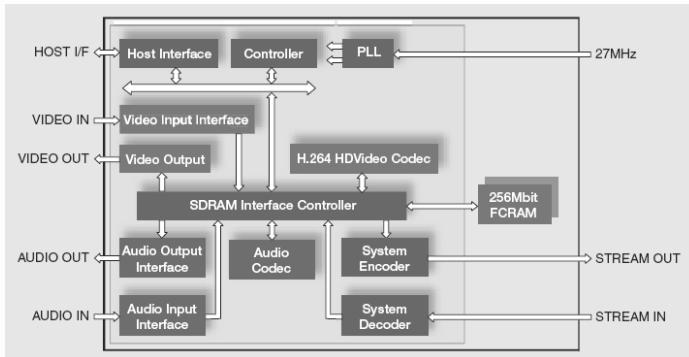
- U prethodnim predavanjima:
 - jedan CPU
- Danas: vrlo zahtjevni zadaci (MM je jedan od njih) mogu se na još jedan način ubrzati koristeći više dretvi (thread) ili više jezgri (core)
- Bez ulazeњa u teoriju, jasno je da se djelomičnom paralelizacijom procesa može postići veća brzina

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

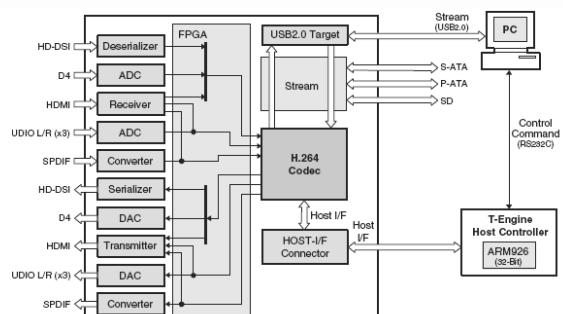
Primjeri: Fujitsu

- MB86H50: H.264 Video Processing IC



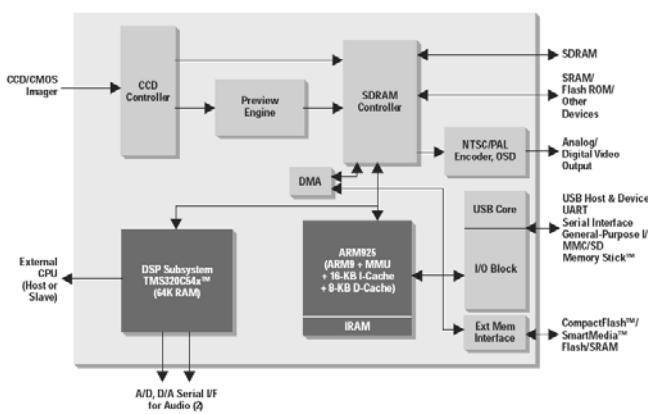
Fujitsu

- Primjer sustava

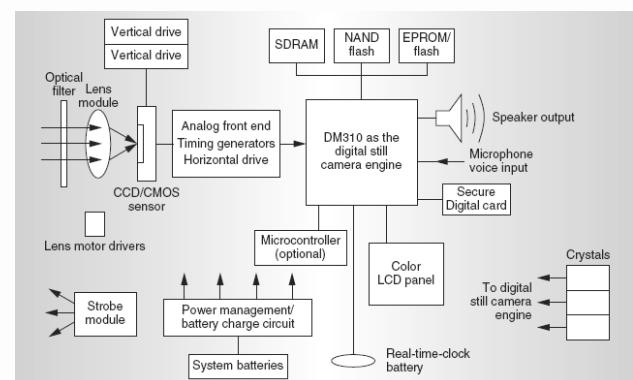


Primjer: TI

TMS320DM310 Functional Block Diagram

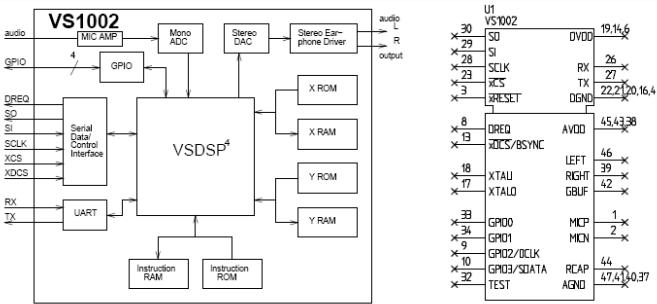


TI

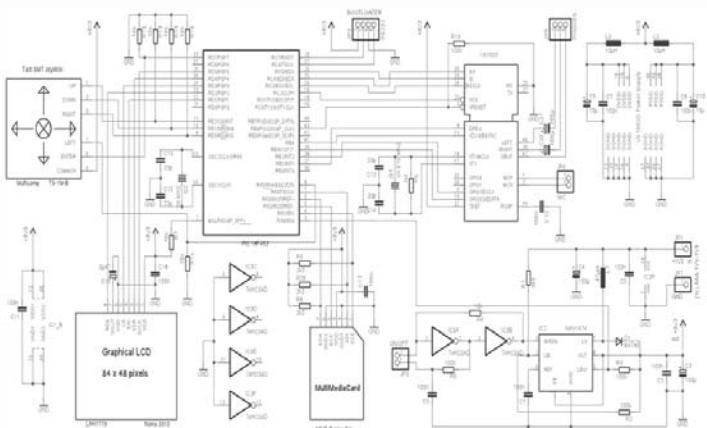


VLSI Solution

■ 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

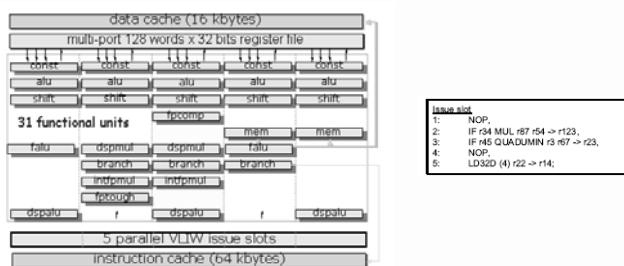
- U dosadašnjim primjerima procesor je bio prvenstveno projektiran za "opće" zadatke
- Za obradu multimedije koristile su se "poboljšanja"
- Novi pristup: procesor se projektira PRVENSTVENO za obradu multimedijiskih podataka, a ostale stvari može obradjavati ali tek sporedno
 - -> Medijski procesor

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

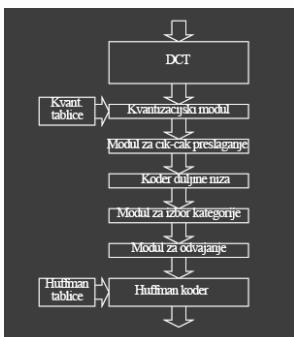
- Izuzetno visoke performanse (veće od GPP, medijskih koprocесora)
- Složen postupak programiranja
- Paralelno izvođenje operacija unutar jedne naredbe
- Još uvijek programabilan
- Podrška za opće zadatke mora biti osigurana od dodatnog procesora

FPGA/ASIC

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

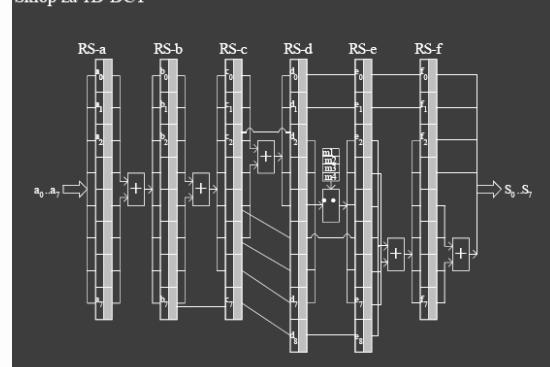
JAGUAR

■ HW JPEG encoder



JAGUAR

Sklop za 1D-DCT

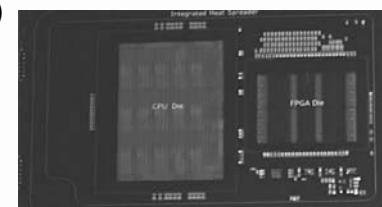


VLSI/ASIC

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

Heterogeni procesori

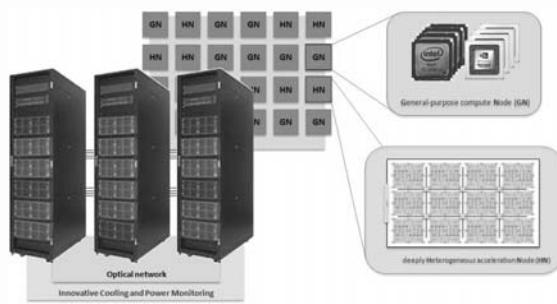
- Najnoviji pristup poboljšanju arhitekture
- CPU + GPU
- CPU + FPGA
- CPU + GPU + FPGA
- **Intel Broadwell Xeon with a built-in FPGA (2016)**



35

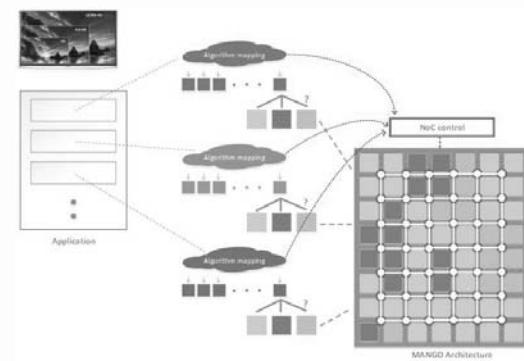
MANGO Arhitektura

- Deeply heterogeneous QoS aware architecture



36

Pogled sa strane aplikacija



37

Zaštita podataka

Kripto algoritmi

- Simetrični
 - Kriptiranje i dekriptiranje podataka obavlja se sa istim ključem
- Asimetrični
 - Različiti ključevi za kriptiranje i dekriptiranje

Računalna kompleksnost

- Kompleksnost za 3DES je 8091 operacija za 64 bitovni blok (cca 1000 op. po bajtu)

Step #	operation	#Times	Equiv Total*	Notes
1,19	64 bit transposition	2	64x2	
2,17	32 bit COPY	16	16	
2,17	32 bit XOR	16	16	16steps
2,17	48 bit transposition	16	48x16	
2,17	48 bit XOR	16	16	f function
2,17	48 bit SWAP	8x16	3x128	
Dimensional Table Mapping				
2,17	32 bit transposition	16	32x16	
1	56 bit transposition	1	56	
2,17	26 bit LEFT SHIFT	2x16	32	
2,17	48 bit transposition	16	48x16	
18	32 bit SWAP	1	1	Pre-out DES Total 3DES Total 8091 8091

Multimedijiske arhitekture i sustavi

40

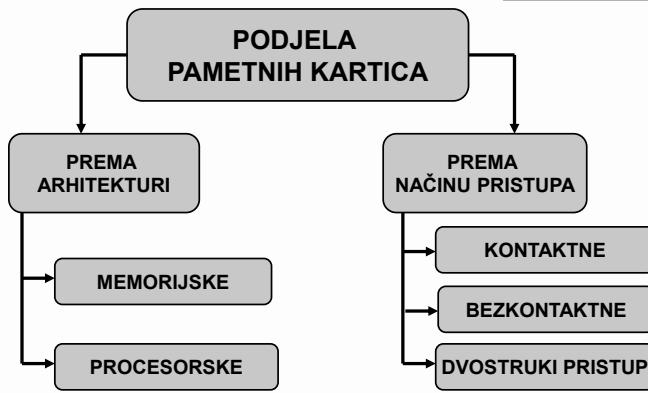
Multimedijiske arhitekture i sustavi

41

Pametne kartice



Podjela



Multimedijiske arhitekture i sustavi

44

Diskusija....

- Kako najefikasnije zaštititi digitalni sadržaj ?
- Koji su Vaši prijedlozi.....
- Analiza sigurnosnih rizika...

Multimedijiske arhitekture i sustavi

41

Svrha

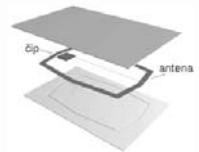
- Mobilnost
- Pohrana ili obrada podataka
- Zaštita podataka

Multimedijiske arhitekture i sustavi

43

Fizička izvedba

- Pametne kartice
 - Danas su još kartice ali uskoro više ne...
- Osnovni parametri: ISO 7816-1,2,3,4...
- Elektronički sklop umetnut u sloj plastike
- Uspostava komunikacije sa vanjskim svjetom preko metalnih kontakata ili induktivnom vezom



Multimedijiske arhitekture i sustavi

45

Dimenziije

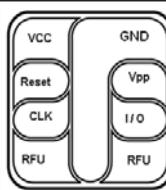
- Kartice – tri formata
 - ID-1 (85.6mm x 54mm)
 - ID-00 (66mm x 33mm)
 - ID-000 (25mm x 15mm)



Multimedijске arhitekture i sustavi

46

Kontaktne kartice



- Vcc, GND
- Vpp
- Reset
- I/O
- CLK

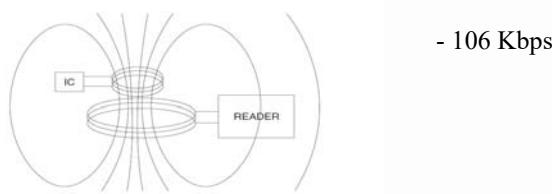
napajanje
V programiranje
podaci
takt

Multimedijске arhitekture i sustavi

47

Beskontaktne kartice

- komunikacija na 13,56 MHz
- čitač emitira elektromagnetsko polje
- val nosilac na 847,5 kHz



Multimedijске arhitekture i sustavi

48

Napajanje

- Postoje 3 metode napajanja pametnih kartica:
- Iz vanjskog izvora napajanja preko kontakata
 - Iz vanjskog izvora napajanja induktivno
 - Iz baterije koja je ugrađena u karticu (rijetko)

Multimedijске arhitekture i sustavi

49

Memorijske kartice



Memorijske kartice

- Vrlo jednostavne kartice
- Niska cijena
- Cjelokupna logika izvedena je korištenjem kombinacijskih sklopova

Multimedijске arhitekture i sustavi

51

Memorijske kartice

- Funkcionalnosti:
 - Kartice s pohranjenom vrijednošću
 - Logički zaštićena memorija (ireverzibilna)
 - Kartice s običnom memorijom
 - Pohrana podataka
 - Bez zaštitnih mehanizama
 - Kartice s zaštićenom memorijom
 - Jednostavni mehanizmi kontrole pristupa podacima

Multimedejske arhitekture i sustavi

52

Procesorske kartice

Procesorske kartice

- Mali računalni sustavi zasnovani na jednostavnijim, uglavno 8-bitovnim procesorima
- Pored procesora: ROM (za OS), EEPROM (za aplikacije i trajne varijable), RAM (za varijable)
- Napredne inačice imaju i kripto koprocesor
- Procesor izvodi Card Operating System (COS)
 - Razni proizvođači nude različite COS

Multimedejske arhitekture i sustavi

54

Multimedejske arhitekture i sustavi

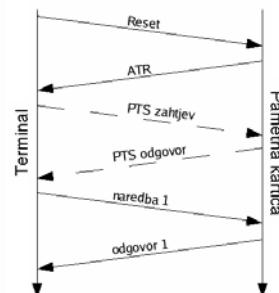
55

Procesorske kartice

- Procesor omogućuje pohranu i obradu podataka
- Procesor komunicira sa vanjskim svijetom serijskom vezom definiranom ISO normom
- Komunikacija ne ovisi o OS-u ili procesoru

Upostavljanje komunikacije

- ATR
 - Answer to Reset
 - Definiraju se parametri komunikacije
- PTS
 - Protocol Type Selection
 - Definiraju se protokoli
- naredbe



Multimedejske arhitekture i sustavi

56

Komunikacija s računalom

- Dvosmjerna komunikacija preko čitača
- Half duplex – ne istovremeno u oba smjera
- Komunikacija podatkovnim paketima
- APDU protokol (Application Protocol Data Unit)
 - Razmjena naredbi i odgovora
- Odnos "gospodar – sluga" (engl. "master – slave")

Multimedejske arhitekture i sustavi

57

Komunikacijski protokoli

■ APDU naredba

zaglavlje				tijelo		
CLA	INS	P1	P2	Lc	Podatkovno polje	Le

- CLA: oznaka klase instrukcije
- INS: instrukcija
- P1, P2: parametri instrukcije, proširenja
- Lc: Određuje duljinu podatkovnog polja
- Podatkovno polje
- Le: Očekivani broj bajtova u odgovoru

Multimedejske arhitekture i sustavi

58

Komunikacijski protokoli

■ APDU odgovor

tijelo	statusna riječ	
Podatkovno polje	SW1	SW2

- Podatkovno polje
- SW1, SW2
 - Statusna riječ
 - Stanje kartice nakon izvršenja APDU naredbe

Multimedejske arhitekture i sustavi

59

Način obrade podataka

- Procesor prima naredbu i eventualne podatke
- Naredba se izvodi, međurezultati se spremaju u internu memoriju
- Eventualni povratni rezultati šalju se natrag u odgovoru na poruku

Multimedejske arhitekture i sustavi

60

Kripto-koprocesor

- Kripto algoritmi su izuzetno zahtjevni i te ako postoji potreba za bržom obradom pored generalnog procesora dodaje se kriptografski procesor
- Kriptografski procesor ima ponekad i cilj zaštiti generatora slučajnih brojeva (!) te nekih master-ključeva

Multimedejske arhitekture i sustavi

61

Primjeri

MIFARE

O MIFARE-u

- ISO 14443: **tip A** i tip B
- NXP Semiconductors (Philips)
- RF komunikacijska tehnologija
- Memorijske kartice
- Doseg očitanja do cca 10 cm
- Najraširenija tehnologija beskontaktnih kartica u svijetu
- 500 mil. kartica i oko 5 mil. čitača u upotrebi



Multimedejske arhitekture i sustavi

64

Organizacija podataka

Sector	Block	Byte Number within a Block															Description
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	3																Sector Trailer 15
	2																Data
	1																Data
	0																Data
14	3																Sector Trailer 14
	2																Data
	1																Data
	0																Data
:	:																
1	3																Sector Trailer 1
	2																Data
	1																Data
	0																Sector Trailer 0
0	3																Manufacturer Block
	2																
	1																
	0																

Multimedejske arhitekture i sustavi

65

Organizacija kartice

- 16 sektora sa 4 bloka
- 0. sektor → serijski broj
- 0. sektor → MAD
- Svaki sektor zasebni par ključeva (A i B)
- Ostalih 15 sektora je rezervirano za aplikacije

Multimedejske arhitekture i sustavi

66

Organizacija kartice

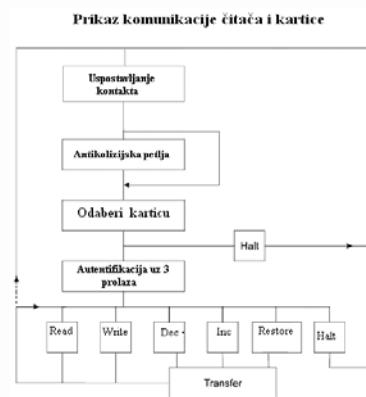
- MAD (direktorij) definira zajedničku strukturu podataka za sve aplikacije koje se nalaze na kartici. MAD time omogućuje terminalu koji očitava karticu da u što kraćem vremenu i što učinkovitije pronađe odgovarajuću aplikaciju
- Svaka MIFARE kartica sadrži integrirani čip sa jedinstvenom identifikacijom (UID) koja je dodijeljena još za vrijeme procesa proizvodnje
- Serijski broj nije kriptiran i nalazi se u sektoru 0 (blok 0) i ne može se ništa preko njega upisati u to područje.

Multimedejske arhitekture i sustavi

67

Način komunikacije

- Uspostavljanje kontakta
- Antikolizijska petlja
- Odaberi karticu
- Autentifikacija sa 3 prolaza
- Nakon autentifikacije:**
- READ WRITE
- INCREMENT DECREMENT
- TRANSFER RESTORE



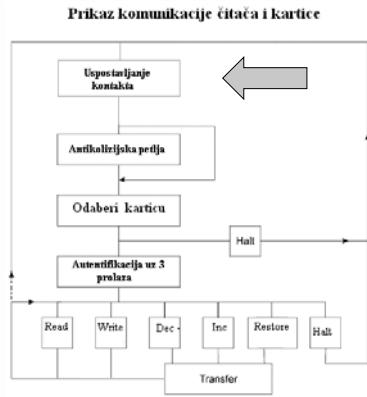
Multimedejske arhitekture i sustavi

68

Način komunikacije

- Uspostavljanje kontakta
- Čitač ostvaruje komunikaciju sa karticama

Nakon registriranja nastavlja se komunikacija odgovarajućim protokolima.



Multimedejske arhitekture i sustavi

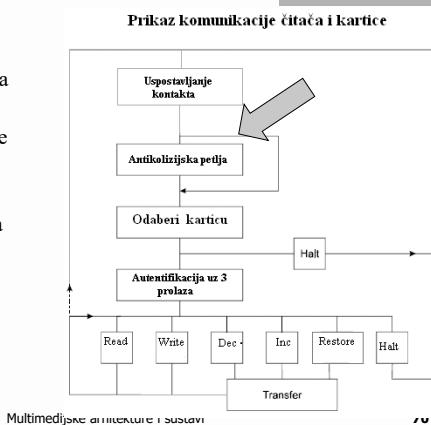
69

Način komunikacije

■ Antikolizijska petlja:

- Čitanje serijskog broja kartice
- Mogućnost pojave više kartica u dosegu

Odabire se samo jedna sa Odaberi karticu
Ostale u stand by mode



Multimedijiske arhitekture i sustavi

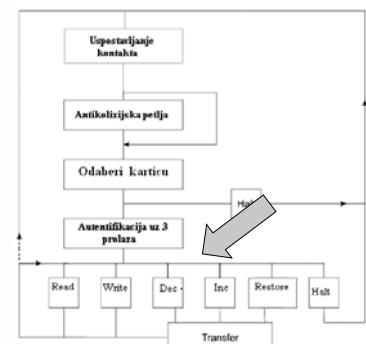
70

Način komunikacije

■ Operacije:

- READ
- WRITE
- DECREMENT
- INCREMENT
- TRANSFER
- RESTORE

Prikaz komunikacije čitača i kartice



Multimedijiske arhitekture i sustavi

71

Podatkovni blokovi

- Svaki sektor sadrži 3 bloka za pohranu podataka. Ti podatkovni blokovi mogu sa pristupnim bitovima biti različito konfigurirani:
 - ▶ Read/write blokovi
 - Blokovi vrijednosti kod kojih su omogućene dodatne naredbe kao increment ili decrement za kontrolu i obradu pohranjenih podataka
 - Nad određenom vrstom blokova se mogu izvršavati samo operacije definirane za te blokove

Multimedijiske arhitekture i sustavi

72

Sektorske karakteristike

Svaki sektor ima poseban dio koji sadrži:

- Tajne ključeve A i B, koji vraćaju logičke nule pri čitanju
- ▶ Pristupne uvjetne za 4 bloka tog sektora
- ▶ Ukoliko se ne koristi jedan od ključeva jer nema potrebe za njim, njegovi bitovi se mogu pretvoriti u podatkovne bitove.

SEKTOR	BLOK	BROJ BYTE-a U BLOKU															OPIS	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
15	3																	SEktorski dodatak 15
		KLJUČ A																Podaci
			2															Podaci
				1														Podaci
					0													

Funkcijske karakteristike

Uvjeti za pristup podacima:

- Sector trailer – poseban dio sektora u kojem se nalaze pristupni uvjeti u obliku 3 bita
- Kontroliraju razine memorijskog pristupa koristeći ključeve A i B
- Sa svakim memorijskim pristupom, unutarnja logika provjerava format pristupnih uvjeta.

Pristupni bitovi	Naredbe	BLOK	OPIS
C ₁ ₃ C ₂ ₃ C ₃ ₃	read, write	3	sekt. dodatak
C ₁ ₂ C ₂ ₂ C ₃ ₂	read, write, increment, decrement, transfer, restore	2	podaci
C ₁ ₁ C ₂ ₁ C ₃ ₁	read, write, increment, decrement, transfer, restore	1	podaci
C ₁ ₀ C ₂ ₀ C ₃ ₀	read, write, increment, decrement, transfer, restore	0	podaci

Pristupna konfiguracija za podatke

Table 4. Access conditions for data blocks

Access bits			Access condition for			Application		
C1	C2	C3	read	write	increment	decrement,	transfer,	restore
0	0	0	key A B ¹¹	key A B1	key A B1	key A B1	key A B1	transport configuration
0	1	0	key A B ¹¹	never	never	never	never	read/write block
1	0	0	key A B ¹¹	key B ¹	never	never	never	read/write block
1	1	0	key A B ¹¹	key B ¹	key B ¹	key A B1	key A B1	value block
0	0	1	key A B ¹¹	never	never	key A B ¹	key A B ¹	value block
0	1	1	key B ¹¹	key B ¹	never	never	never	read/write block
1	0	1	key B ¹¹	never	never	never	never	read/write block
1	1	1	never	never	never	never	never	read/write block

Za više detalja:

<http://www.nxp.com/products/identification/mifare/classic/#rel>

Multimedijiske arhitekture i sustavi

75

Primjer:

- Korištenje MIFARE kartice za dvije jednostavne aplikacije:
 - Čitanje podatka
 - Elektronički novčanik

Multimedejske arhitekture i sustavi

76

Java Card

Osobine Java Card OS-a

- temelji se na normi ISO 7816-4
- neovisnost kartične aplikacije o platformi na kojoj se izvodi
- učitavanje aplikacija nakon proizvodnje kartice
- osnovne namjene operacijskog sustava:
 - upravljanje prijenosom podataka
 - kontrola izvođenja naredbi
 - izvođenje kriptografskih algoritama
 - stvaranje digitalnih potpisa

Multimedejske arhitekture i sustavi

78

Arhitektura Java Card tehnologije

- najmanja računalna platforma za Java
- tipična konfiguracija memorije: 1K RAM-a, 32K EEPROM-a i 24K ROM-a
- podržan samo podskup osobina Java jezika

Multimedejske arhitekture i sustavi

79

Arhitektura Java Card tehnologije



Multimedejske arhitekture i sustavi

80

Java Card platforma

- platforma za izvršavanje Java appleta
- postojanje više aplikacija na kartici
- neovisnost i zaštićenost aplikacija
- dijelovi Java Card platforme:
 - Java Card Virtual Machine (JCVM)
 - Java Card Runtime Environment (JCRE)
 - Java Card Application Programming Interface (API)

Multimedejske arhitekture i sustavi

81

Svojstva Java Card jezika

- podržan samo podskup osobina Java jezika
- očuvana objektno-orientirana svojstva Java jezika
- razlika između Java Card appleta i Java appleta

podržane osobine Java jezika	nepodržane osobine Java jezika
primitivni tipovi podataka: byte, short, boolean, int (kod nekih platformi)	primitivni tipovi podataka: char, long, float, double
Java paketi	dretve, dinamičko upravljanje memorijom, kloniranje objekata
klase, iznimke	znakovi i stringovi
jednodimenzionalna polja	višedimenzionalna polja
sučelja	upravljanje sigurnošću
objektno-orientirane osobine	dinamičko učitavanje klasa

Multimedijijske arhitekture i sustavi

82

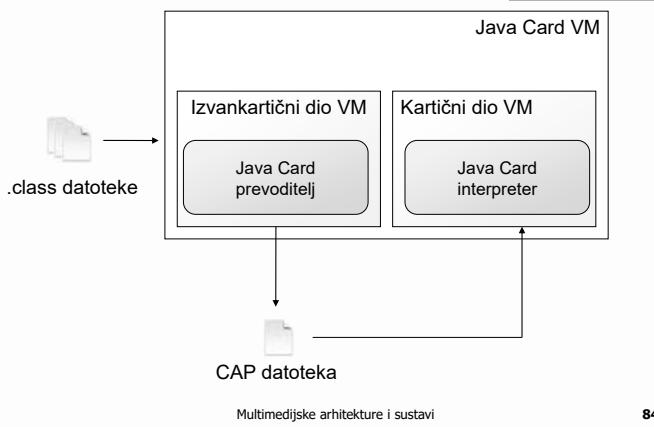
Java Card stogovni stroj

- podijeljen na dva dijela
- izvankartični dio (Java Card prevoditelj)
 - procesiranje i kreiranje CAP datoteke
 - inicijalizacija statičkih varijabli
 - optimiziranje Java bytecode-a
 - kreiranje podatkovnih struktura
- kartični dio (Java Card interpreter)
 - izvedba bytecode naredbi
 - upravljanje memorijom
 - kreiranje objekata

Multimedijijske arhitekture i sustavi

83

Java Card stogovni stroj (2)



Multimedijijske arhitekture i sustavi

84

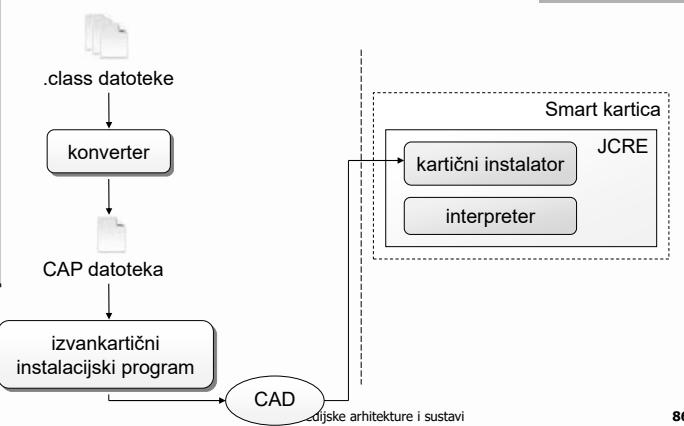
Java Card instalator (1)

- Java Card interpreter – izvršava CAP datoteke
- instalator – mehanizmi preuzimanja i instaliranja
- CAD uređaj – prihvata karticu
- CAP datoteka sadrži binarni kod
- podjela funkcionalnosti između interpretera i instalatora
 - smanjena veličina instalatora
 - fleksibilnost implementacije instalatora

Multimedijijske arhitekture i sustavi

85

Java Card instalator (2)



Multimedijijske arhitekture i sustavi

86

Java Card izvršna okolina (1)

- sistemske komponente – izvršavaju se unutar kartice
- upravljanje kartičnim resursima
- upravljanje mrežnom komunikacijom
- izvršavanje appleta
- sigurnost appleta
- najvažnija namjena izvršne okoline – uloga operacijskog sustava
- appleti se mogu učitavati na karticu nakon proizvodnje

Multimedijijske arhitekture i sustavi

87

Java Card izvršna okolina (2)

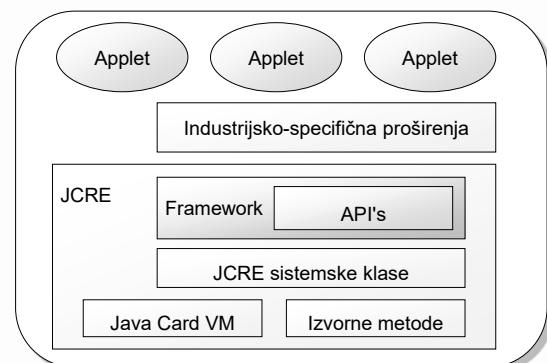
- dijelovi izvršne okoline:

- Java Card stogovni stroj
- Java Card aplikacijske klase
- industrijsko-specifična proširenja
- JCRE sistemske klase

Multimedijске arhitekture i sustavi

88

Java Card izvršna okolina (3)



Multimedijске arhitekture i sustavi

89

Java Card API

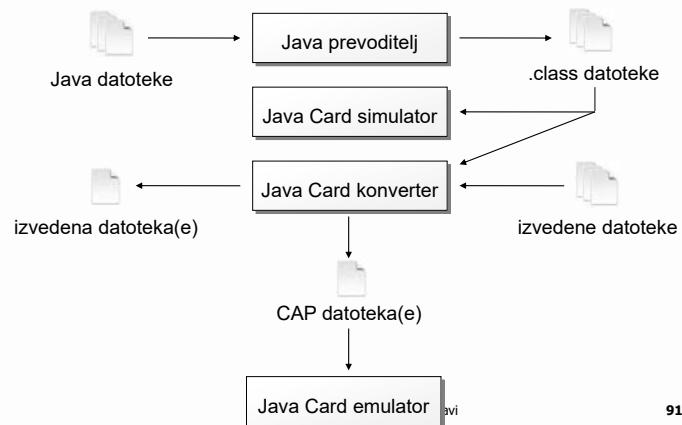
- podržan samo podskup Java paketa
- prilagođeni razredi
- vlastiti jezgreni razredi
- osnovni Java Card API (2.1.1.) paketi:

Ime paketa	Opis paketa
java.lang	osnove dizajna Java Card tehnologije
javacard.framework	osnovni skup klasa i sučelja potrebnih za izgradnju i rad s Java Card appletima.
javacard.security	razredi i sučelja koja su potrebna za implementaciju sigurnosnih alata
javacardx.crypto	dodatni paket koji sadrži sigurnosne klase koje pružaju podršku za kriptografske alate

Multimedijске arhitekture i sustavi

90

Razvojni proces appleta



91

Sigurnosne osobine Java Card platforme

- implementirane osobine Java Card platforme:
 - integritet transakcije
 - perzistentni i tranzientni objekti
 - aplikacijski vatrosid
 - sigurnosne i kriptografske klase
 - izvorne metode u appletima

Multimedijске arhitekture i sustavi

92

Primjer

- Jednostavna aplikacija za izračun kripto funkcije

Multimedijске arhitekture i sustavi

93

Multimedejske arhitekture i sustavi

Prof.dr.sc. Mario Kovač

Prof.dr.sc. Hrvoje Mlinarić



Multimedejske arhitekture i sustavi

1

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

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 !!!

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)

Multimedejske arhitekture i sustavi

3

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 pomoći matematike.
 - Problem znanja i primjene

Multimedejske arhitekture i sustavi

4

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

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

Multimedejske arhitekture i sustavi

5

■ RGB to YUV cijelobrojni

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

■ RGB to YUV cijelobrojni pojednostavljeni

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

Multimedejske arhitekture i sustavi

6

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

Multimedijiske arhitekture i sustavi

7

Kako riješiti problem?

- Aproksimacija funkcija polinomom n-tog reda...
- Prevođenje algoritama iz realne u cjelobrojnu 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.

Multimedijiske arhitekture i sustavi

8

Kako riješiti problem

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

Multimedijiske arhitekture i sustavi

9

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

Multimedijiske arhitekture i sustavi

10

Multimedijiske arhitekture i sustavi

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

Multimedijiske arhitekture i sustavi

11

Hardware Software codesign

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

Multimedijiske arhitekture i sustavi

12

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

Multimedejske arhitekture i sustavi

13

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, ...

Multimedejske arhitekture i sustavi

14

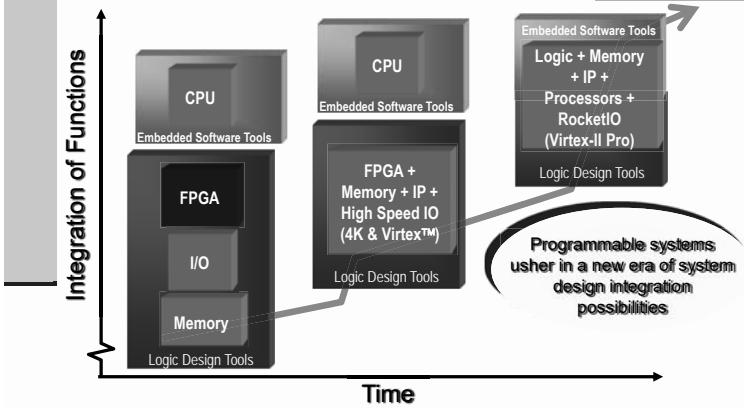
Postojeća tehnologija

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

Multimedejske arhitekture i sustavi

15

Integration in System Design



Multimedejske arhitekture i sustavi

16

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)

Multimedejske arhitekture i sustavi

17

Što je u planu za ovaj dio semestra?

- Dobiti neki osnovni uvid u problematiku i metode rješavanja 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

Multimedejske arhitekture i sustavi

18

Primjer 22 Mpixel digitalni aparat

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



Multimedijiske arhitekture i sustavi

19

- 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

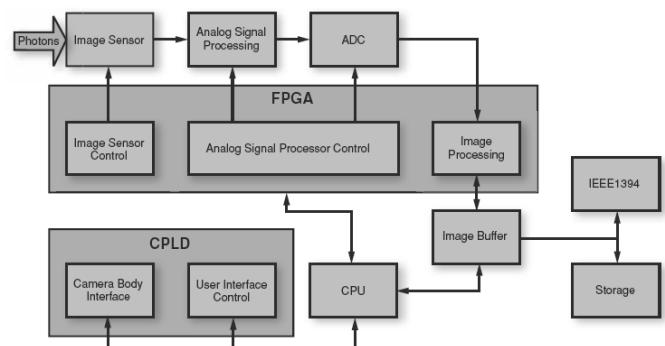


- Danas:
- CCD Full frame
- Rezolucija: 80 mega pixels
- CCD size effective 49.1 x 36.8 mm
- Pixel size 6.8 x 6.8 micron
- Image ratio 4:3
- Cijena: \$49,990

Multimedijiske arhitekture i sustavi

21

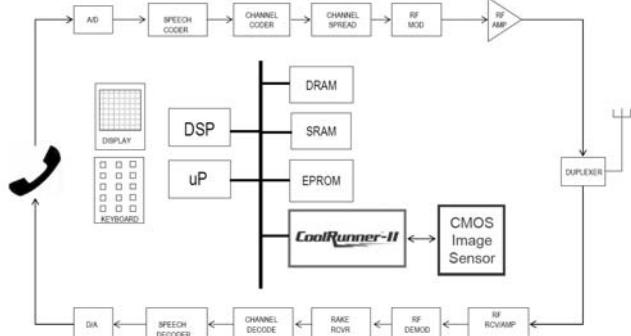
Phase One



Multimedijiske arhitekture i sustavi

22

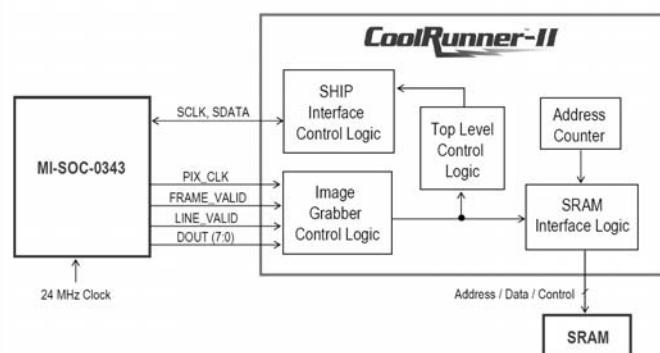
Mobilni telefoni - primjer



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 funkcione 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 novo 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 sklopova 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.

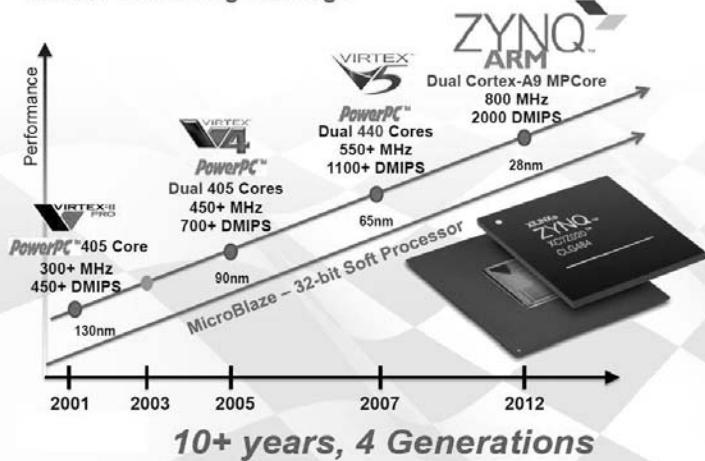
Soft – Hard procesor

- Soft procesor – koristi FPGA blokove
- Hard procesori – Izvedeni 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.

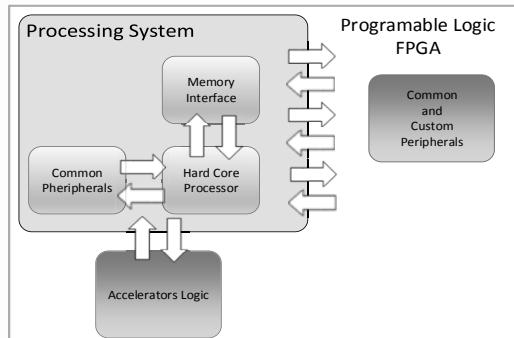
FPGA embedded processor

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

Xilinx Processing Heritage



Što imamo danas?



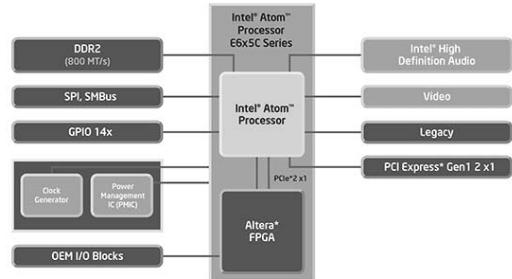
What We Have Today ?

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

Xilinx Zynq 7000 family The All Programmable SoC

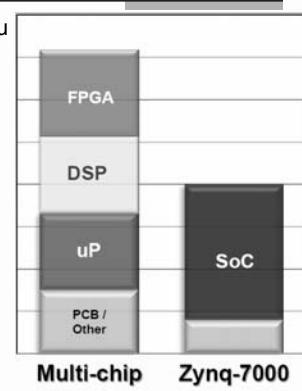
- **Nije običan FPGA!**
- **Nije običan mikroprocesor!**
- **Jedinstvena simbioza i jednog i drugog**
 - Dual ARM Cortex™-A9 procesor+ caches + robusni sustav periferija
 - Velika povezanost između procesora i logike
- **To više nije FPGA ili procesor to je procesorski sustav s programibilnom logikom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

Intel Atom Processor E6x5C Series



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

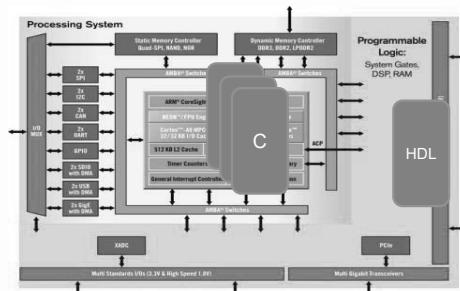
Meet HW and SW Processing Performance Needs

- 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

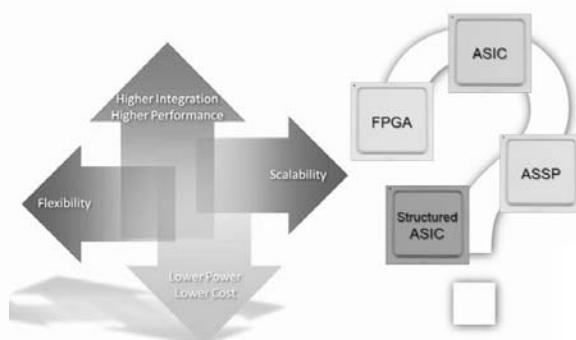
Povećanje performansi

Optimized & Simplified HW/SW Partitioning



Današnji zahtjevi

Koju tehnologiju izabrati?



Što odabrati?

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

Što odabrati?

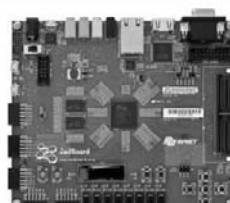
	ASIC	ASSP	2 Chip Solution
Performance	+	+	■
Power	+	+	~
Unit Cost	+	+	~
TCO	■	+	+
Risk	~	+	+
TTM	~	+	+
Flexibility	■	~	+
Scalability	~	■	+

Što odabrati?

	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



Multimedijске arhitekture i sustavi

49

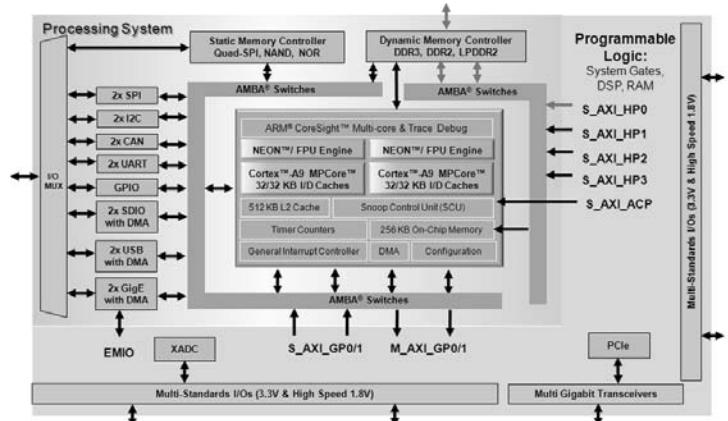
Komponente sustava

- 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

Multimedijске arhitekture i sustavi

50

ZYNQ 7000



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

Multimedijске arhitekture i sustavi

51

ZYNQ 7000

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

Multimedijске 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)

Multimedijске 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

Multimedijiske arhitekture i sustavi

55

ZYNQ – dual core procesor

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

Multimedijiske arhitekture i sustavi

56

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

Multimedijiske arhitekture i sustavi

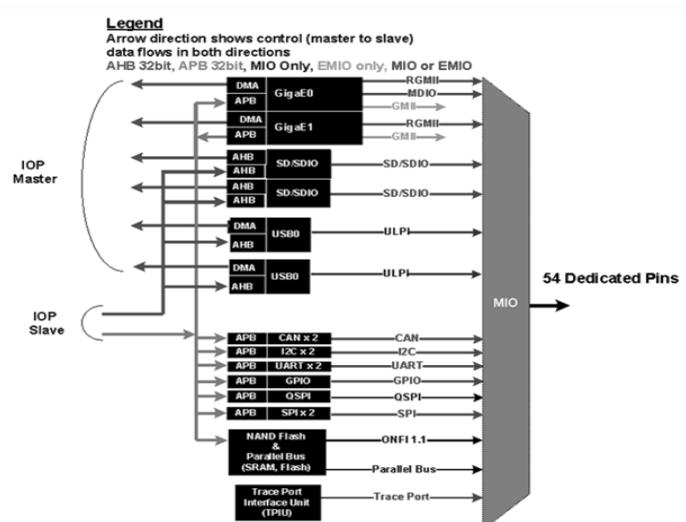
58

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

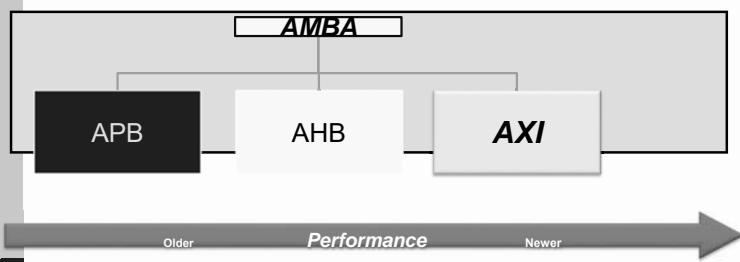
61

- 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

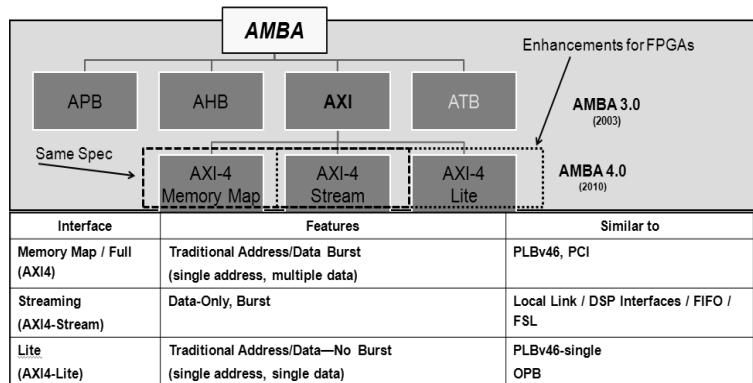


AMBA: Advanced Microcontroller Bus Architecture
AXI: Advanced Extensible Interface

Multimedijiske arhitekture i sustavi

63

AXI



Multimedijiske arhitekture i sustavi

64

Programabilna 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

Multimedijiske arhitekture i sustavi

65

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,

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

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

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	XC6SLX25T	XC6SLX45T	XC6SLX75T	XC6SLX100T	XC6SLX150T
Logic Resources																		
Slice ⁽¹⁾	600	1,450	2,278	3,758	6,822	11,692	15,822	23,038	3758	6,822	11,692	15,822	23,038	401	692	976	1,355	
Logic Cells ⁽²⁾	3,840	9,152	14,979	24,051	43,661	74,537	101,261	147,443	24,051	43,661	74,537	101,261	147,443	401	692	976	1,355	
CLB Flip-Flops	4,600	11,440	18,224	30,064	54,576	93,296	128,576	184,304	30,064	54,576	93,296	128,576	184,304	401	692	976	1,355	
Memory Resources																		
Maximum Distributed RAM (Kbit)	75	90	136	229	401	692	976	1,355	229	401	692	976	1,355	401	692	976	1,355	
Block RAM (16K bit each)	12	32	82	92	116	172	268	368	52	116	172	268	368	1	1	1	1	
Total Block RAM (Kbit)	216	576	576	936	2,088	3,096	4,824	6,824	936	2,088	3,096	4,824	6,824	1	1	1	1	
Clock Resources																		
Clock Manager Tiles (CMT) ⁽³⁾	2	2	2	2	4	6	6	6	2	4	6	6	6	1	1	1	1	
I/O Resources																		
Maximum Single-Ended Pins	120	200	292	296	358	400	480	570	250	296	320	400	530	125	148	160	245	
Maximum Differential Pairs	60	100	116	133	170	200	240	285	125	148	160	245	285	38	58	132	180	
Embedded Hard IP Resources																		
DSP48A1 Slice ⁽⁴⁾	8	16	32	38	58	132	180	180	38	58	132	180	180	1	1	1	1	
PCI Express® Endpoint Block	—	—	—	—	—	—	—	—	—	—	—	—	—	1	1	1	1	
Memory Controller Blocks	0	2	2	2	4	4	4	4	2	2	4	4	4	2	2	4	4	
GTP Low-Power Transceivers	—	—	—	—	—	—	—	—	2	4	8	8	8	2	4	8	8	
Speed Grades																		
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	-2,-3	-2,-3	-2,-3	-2,-3	
Industrial	-41,-2	-41,-2	-41,-2	-41,-2	-41,-2	-41,-2	-41,-2	-41,-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	
Configuration	27	27	27	44	77	106	171	280	44	77	106	171	280	—	—	—	—	

Spartan-6 Family FPGAs



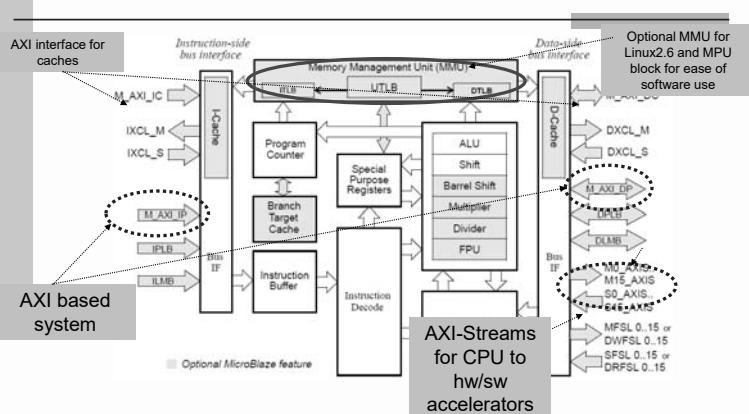
Spartan-6 LX FPGAs Optimized for Lowest Cost Logic, DSP, and Memory (1.2 Volt, 1.0 Volt)										Spartan-6 LXT FPGAs Optimized for Low Cost Logic, DSP, and Memory with High Speed Serial Connectivity (1.2 Volt)									
Part Number	XC6SLX4	XC6SLX9	XC6SLX16	XC6SLX25	XC6SLX45	XC6SLX75	XC6SLX100	XC6SLX150	XC6SLX25T	XC6SLX45T	XC6SLX75T	XC6SLX100T	XC6SLX150T	XC6SLX25T	XC6SLX45T	XC6SLX75T	XC6SLX100T	XC6SLX150T	
Package																			
Chip Scale Package (CSP) Pb-free wire-bond BGA (0.5 mm ball spacing)	CPG106	8 x 8 mm	100	100															
TQFP Package (TQF) Pb-free thin CFP (0.5 mm lead spacing)	TOG144	20 x 20 mm	100	102															
Chip Scale Package (CSP) Pb-free wire-bond chip scale BGA (0.8 mm ball spacing)	CSG225	13 x 13 mm	120	160	160														
	CSG324	15 x 15 mm	200	232	226	218													
	CSG484	19 x 19 mm			310	310	320	330											
FQI256		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"
 - Potrebo 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

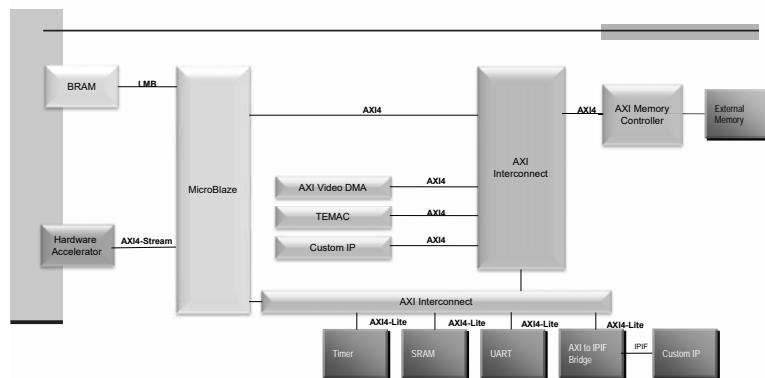
MicroBlaze Block Diagram



Hardware Design 13-73

© Copyright 2012 Xilinx

AXI4 System



Hardware Design 13-74

© 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

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

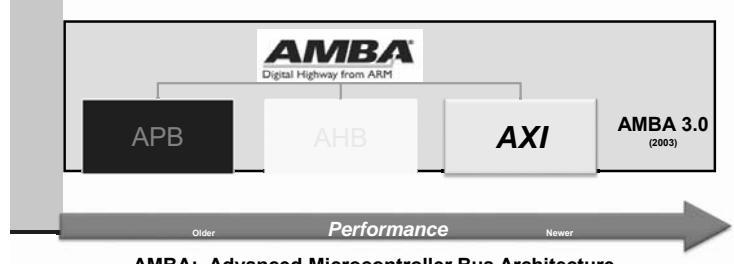
© Copyright 2012 Xilinx

Hardware Design 13-76

Podrška za više jezgri

- Multicore arhitektura
 - Mailbox: komunikacija između dvije jezgre
 - Podrška AXI4-Lite, AXI4-Stream and FSL
 - Mutex core: Sincronizacija 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

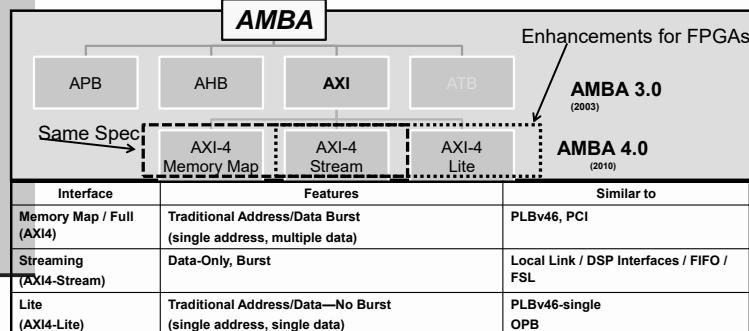
AXI is Part of ARM's AMBA



Hardware Design 13-78

© Copyright 2012 Xilinx

AXI is Part of AMBA: Advanced Microcontroller Bus Architecture

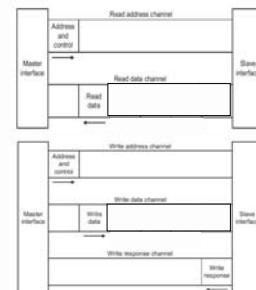


Hardware Design 13- 79

© Copyright 2012 Xilinx

The AXI Interface—AXI4-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)



AXI4-Lite Read

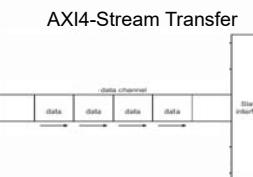
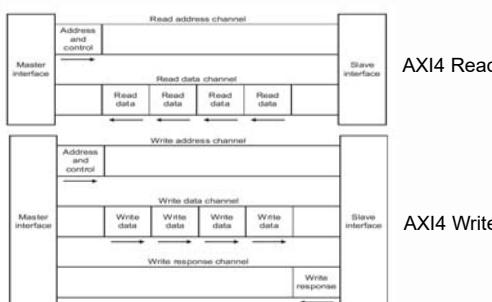
AXI4-Lite Write

© Copyright 2012 Xilinx

Hardware Design 13- 80

The AXI Interface—AXI4-Stream

- Sometimes called "Full AXI" or "AXI Memory Mapped"
- Not ARM-sanctioned name
- Single address multiple data
 - Burst up to 256 data beats
- Data Width parameterizable
 - 1024 bits



© Copyright 2012 Xilinx

Hardware Design 13- 82

Processor Local Bus (PLB)

- Ne preporuča se. Kompatibilnost sa stariim 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 3rd 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

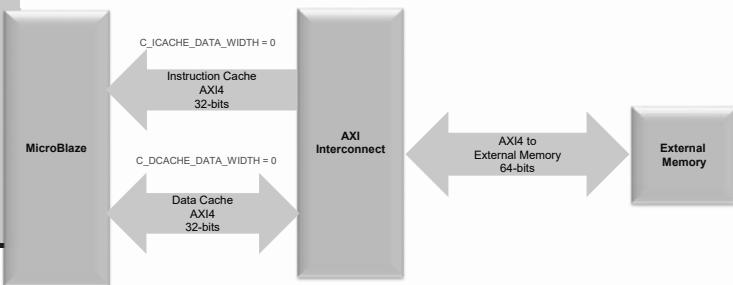
© Copyright 2012 Xilinx

Hardware Design 13- 85

- Local Memory Bus (LMB)
 - Za povezivanje BRAM memorije
- Fast Simplex Links (FSL)
 - Za dodavanje koprocesorskih 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- 87

© Copyright 2012 Xilinx

Multimedijске arhitekture i sustavi

Prof.dr.sc. Mario Kovač

Izv.prof.dr.sc. Hrvoje Mlinarić



Multimedijске arhitekture i sustavi

1

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

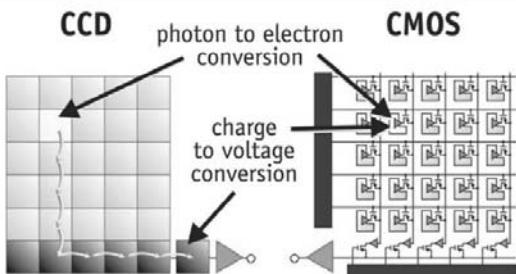
Video Senzor

- 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 dohvataju se istovremeno i uniformno se obrađuju (osnovni uvjet kvalitetne slike).

Video Senzor



Multimedijiske arhitekture i sustavi

4

Video Senzor

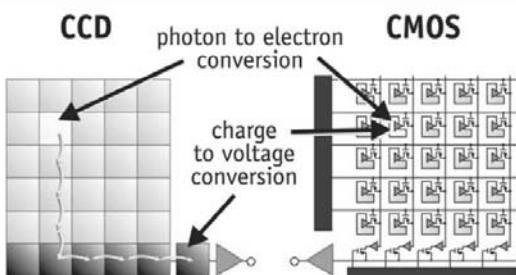
■ CMOS senzor:

- 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



Multimedijiske arhitekture i sustavi

6

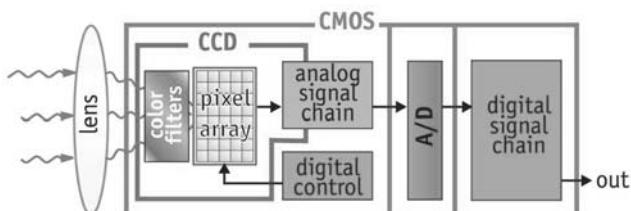
Video Senzor

- 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

Video Senzor

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



Multimedijiske arhitekture i sustavi

9

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

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

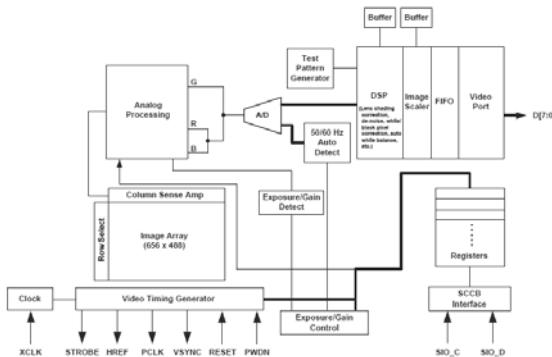
OV7670

- 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

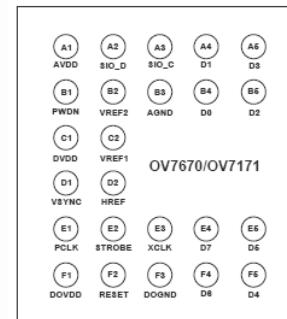
OV7670



Multimedijiske arhitekture i sustavi

14

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.

Multimedijске arhitekture i sustavi

16

OV7670

■ Dva komunikacijska sučelja

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

Multimedijске arhitekture i sustavi

17

SCCB - Serial Camera Control Bus

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

Multimedijске arhitekture i sustavi

18

I2C

- I2C - Inter-Integrated Circuit
- Početkom '80ih Philips Semiconductors razvio je dvosmernu "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 primjenjen i u drugim uređajima.

Multimedijске arhitekture i sustavi

19

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.

Multimedijске arhitekture i sustavi

20

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

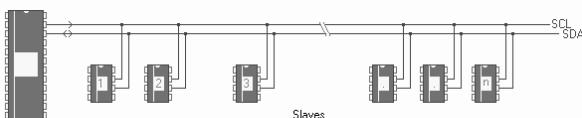
Multimedijске arhitekture i sustavi

21

I2C

Svojstva sabirnice

- Više uređaja može započeti komunikaciju
- Uređaj 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



Multimedijiske arhitekture i sustavi

22

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

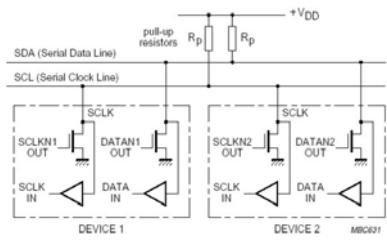
Multimedijiske arhitekture i sustavi

23

I2C

Spojeno i sabirnica

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



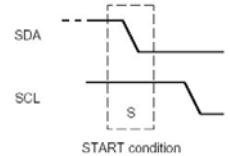
Multimedijiske arhitekture i sustavi

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

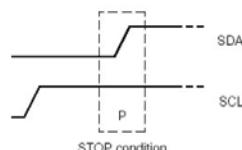
25

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

- Podaci se prenose po bitovima nakon start signala
- Uvijek se šalju podaci 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

Multimedijiske arhitekture i sustavi

28

I2C

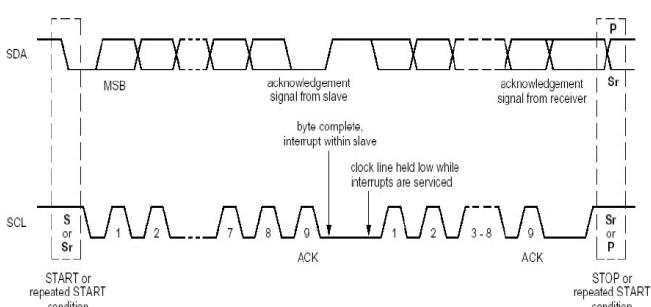
Prijenos podataka

- 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

Multimedijiske arhitekture i sustavi

29

I2C



Multimedijiske arhitekture i sustavi

30

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 primetak podataka
- Nakon toga otpušta SDA signal

Multimedijiske arhitekture i sustavi

31

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 ozičenog i povezivanja

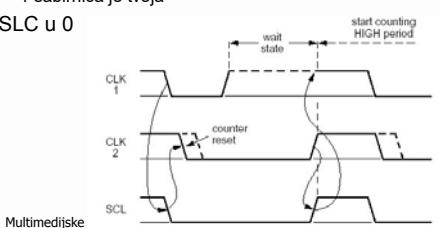
Multimedijiske arhitekture i sustavi

32

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 SCL u 0



I2C

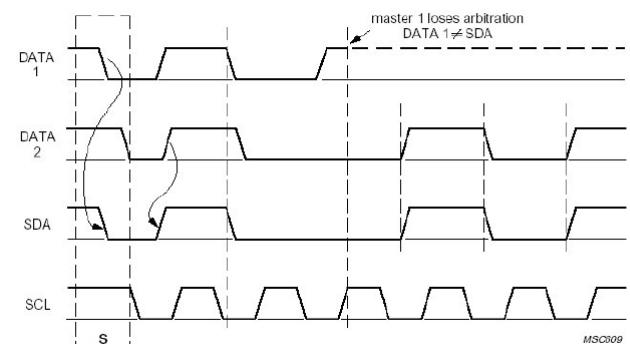
■ Arbitraža na SDA signalu

- 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

Multimedijiske arhitekture i sustavi

34

I2C



Multimedijiske arhitekture i sustavi

35

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 fiksнog dijela i promjenjivog
 - Fiksni dio određuje I2C odbor za dodjelu adresa

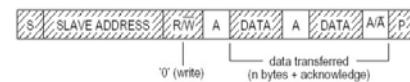
Multimedijiske arhitekture i sustavi

36

I2C

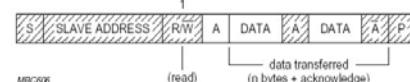
■ Okvir komunikacije

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)



Multimedijiske arhitekture i sustavi

37

I2C

■ Rezervirane adrese za posebnu namijenu:

■ 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** *

Multimedijiske arhitekture i sustavi

38

SCCB

■ Podržava brzinu prijenosa do 400 kbps

■ Imma 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

Multimedijiske arhitekture i sustavi

39

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

Multimedijске arhitekture i sustavi

40

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

Multimedijске arhitekture i sustavi

41

SCCB

- Nama bitni registri unutar senzora:

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

Multimedijске arhitekture i sustavi

42

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

43

OV7670

- Nama bitni registri unutar senzora:

70	SCALING_XSC	4A	RW	<i>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</i> <i>Bit[6:0]: Horizontal scale factor</i>
71	SCALING_YSC	35	RW	<i>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</i> <i>Bit[6:0]: Vertical scale factor</i>

Multimedijске arhitekture i sustavi

44

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

Multimedijске arhitekture i sustavi

45

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)
- Standard 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

Multimedijiske arhitekture i sustavi

46

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 što su MPEG dekoderi za filmove i igrice, TV tuners, live video input and video capture.

Multimedijiske arhitekture i sustavi

47

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

Multimedijiske arhitekture i sustavi

48

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

Multimedijiske arhitekture i sustavi

49

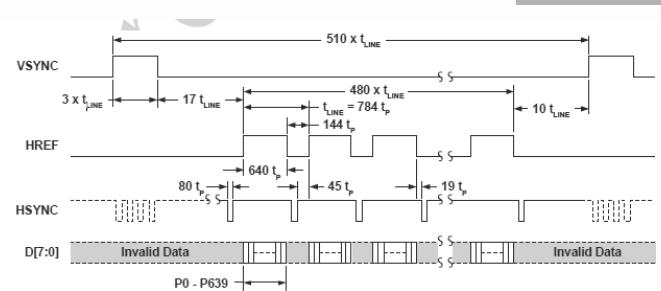
OV7670

- 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 podešiti preko I2C-a

Multimedijiske arhitekture i sustavi

50

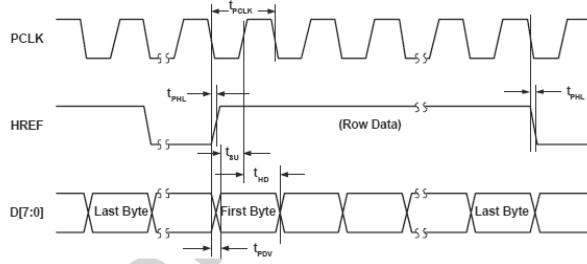
ZV Port



Multimedijiske arhitekture i sustavi

51

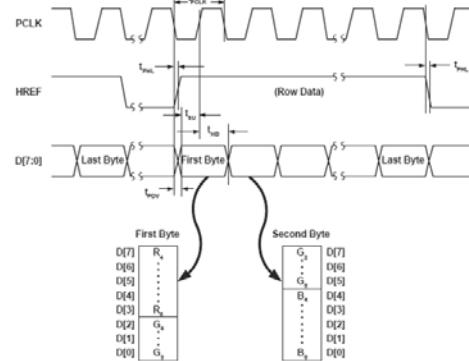
ZV Port



Multimedijiske arhitekture i sustav

52

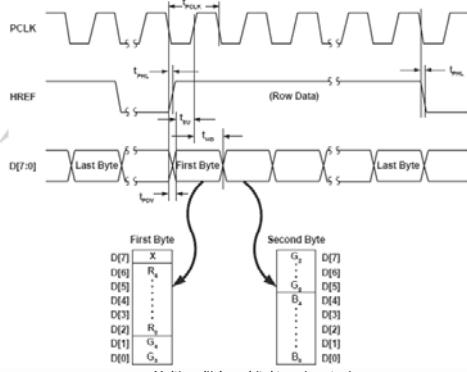
OV7670 – RGB565



Multimedijiske arhitekture i sustavi

53

OV7670 – RGB555



Multimedijiske arhitekture i sustav

54

OV7670

■ Ostali signali koje koristimo

■ RESET

- #### ■ Sklopovalni 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

Multimedijiske arhitekture i sustav

56

OV7670

■ Frame execution mode

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

Multimedijiske arhitekture i sustavi

57

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

Multimedijске arhitekture i sustavi

58

Multimedijске arhitekture i sustavi

59

Multimedijске arhitekture i sustavi

Prof.dr.sc. Mario Kovač

Izv. prof.dr.sc. Hrvoje Mlinarić

Doc. dr.sc. Josip Knezović



Multimedijске arhitekture i sustavi

1

Multimedijске arhitekture i sustavi

2

Osnove profiling-a

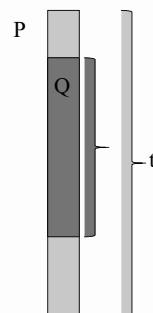
Vrijeme izvođenja programa P je t
Dio programa Q u omjeru p se može ubrzati N puta
Kliko je ubrzanje cijelog programa U?

Amdahlov zakon:

$$U = \frac{t}{t'} = \frac{t}{t * \left(1 - p + \frac{p}{N}\right)} = \frac{1}{\left(1 - p + \frac{p}{N}\right)}$$

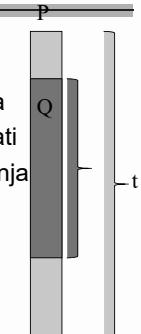
Ako $N \rightarrow \infty$

$$U \approx \frac{1}{1 - p}$$



Osnove profiling-a

- Zašto je Amdahl-ov zakon bitan
- Jer definira realnu situaciju u kojoj se dio programa ne može ubrzati, a postoje i dijelovi koji se mogu ubrzati
- Definira maksimalnu granicu ubrzanja
- Pomaže u definiranju strategije ubrzavanja programa



Zahtjevi

Prihvatljiva funkcionalnost

- ABS: vrijeme odgovora <= hidraulika
- MPEG decoder: propusnost min. 20 fps
- Google: dostupnost rezultata <= 1s

Dodavanje nove funkcionalnosti

- Nove mogućnosti
- Bolja kvaliteta
- Veća veličina problema (data sets)

Mogućnost skaliranja

- Povećanje *workloada*
- Novi zahtjevi (osvježavanja)

Zahtjevi (2)

Potrošnja energije/računskih resursa

Ugradbeni sustavi

Podatkovni centri

Da li moj program postiže maksimum/optimum

Resursi

Memorija, CPU, mrežni promet, energija

Zahtjevi

Propusnost, latencija



Osnove profiling-a

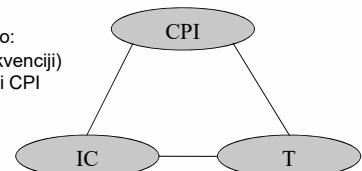
- Cilj profiliranja je pronaći dijelove programa koji troše najviše resursa (usko grlo)
 - Vrijeme
 - Energija
 - Sklopovski: memorija, dma, disk, sabirnica
- Definirati strategiju ubrzavanja identificiranih dijelova
- Strategije ubrzavanja:
 - Asembler (SIMD-izacija)
 - Biblioteke (IPP, MKL, boost, BLAS, LAPACK)
 - Paralelizam (TBB, OpenMP, CUDA, OpenCl, MPI)
 - Novi pristupi (DSL, Go, Cilk+)

Osnove profiling-a

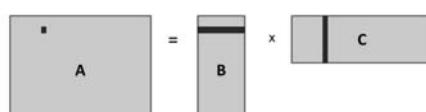
$$T = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clockcycle}} = \text{CPU time}$$

Računske performanse ovise o:

- Vremenskom taktu T (frekvenciji)
- Broju ciklusa po instrukciji CPI
- Broju instrukcija IC



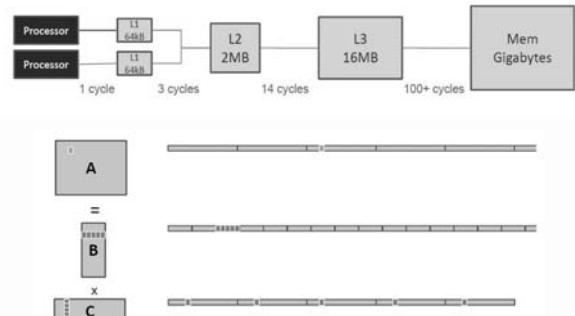
Osnove: Množenje matrica



```
for(int i = 0; i < x; i++)  
for(int j = 0; j < y; j++)  
    for(int k=0; k < z; k++)  
        A[i][j] += B[i][k] * C[k][j];
```

- Matrica B: pristup po retcima
- Matrica C: pristup po stupcima

Osnove: Množenje matrica



Osnove: Množenje matrica

- Transponiranje matrice koja stvara probleme priručnoj memoriji
 - $N^3 \rightarrow N^2$

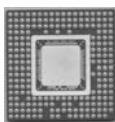
```
#define CITAJ(A, x, y, d) A[(x)*(d)+(y)]
A = (double *)malloc(sizeof(double)*x*y);
B = (double *)malloc(sizeof(double)*x*z);
C = (double *)malloc(sizeof(double)*y*z);
Cx = (double *)malloc(sizeof(double)*y*z);

for(j =0; j < y; j++)
    for(k=0; k < z; k++)
        CITAJ(Cx,j,k,z) = CITAJ(C, k, j, y);

for(i =0; i < x; i++)
    for(j =0; j < y; j++)
        for(k=0; k < z; k++)
            CITAJ(A, i, j, y) += CITAJ(B, i, k, z)*CITAJ(Cx, j, k, z);
```

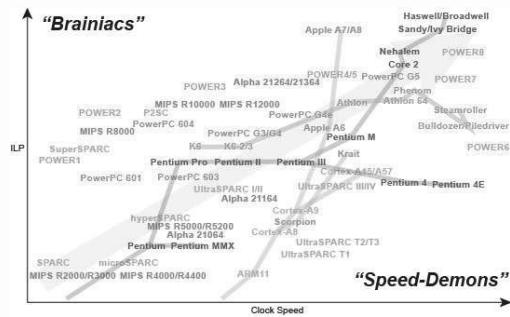
Jednoprocesorski paralelizam

- ILP – Instruction Level Parallelism
 - Arhitekti računala skrivaju paralelizam (2002)
 - Protočna arhitektura (preklapanje dijelova instrukcija)
 - Superskalarno izvođenje (obrada više instrukcija u isto vrijeme)
 - VLIW: Prevoditelj određuje ILP
 - Vektorska obrada: Grupe sličnih nezavisnih operacija (SIMD)
 - "Out of Order Execution": Instrukcije se prividno izvode po redu u toku instrukcija, stvarno ovisno o slobodnim resursima

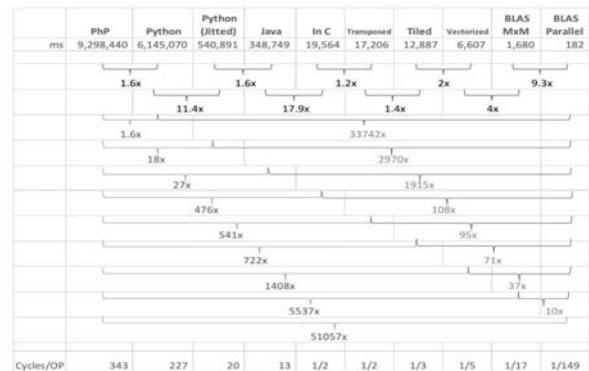


Ograničenja ILP-a

- Brainiacs: ILP u sklopljaju
 - Speed-Demons: ILP u prevoditelju



Osnove: Množenje matrica



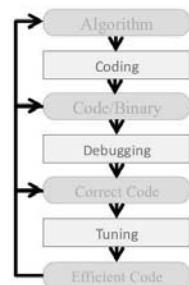
Ograničenja ILP-a

- Hazardi:
 - Strukturni – računski resursi
 - Podatkovni – međuvisnost podataka
 - Kontrolni – Upravljanje tijekom izvođenja
 - Granice (Wall)
 - Power wall
 - ILP wall
 - Memory wall



Performance profiling

- Razvojni ciklus
 - Tuning:
 - Mjerenje performansi
 - Analiza mjerjenja
 - Modifikacija algoritma
 - Ponovno mjerjenje
 - Analiza razlika



Kako mjeriti performanse

- Mjerenje vremena izvođenja
 - Načelno
 - Ne mogu se otkriti kritične točke
- Integracija programskog koda za mjerjenje
 - C/C++: `clock_t, clock()` iz `<time.h>`
 - Nije pogodno za održavanje
- Korištenja alata za profiling
 - Vtune Amplifier, gprof, ...
 - Detaljna analiza
 - Povezanost sa izvornim kodom

Multimedejske arhitekture i sustavi

17

Detalji...

- Identifikacija računski zahtjevnih dijelova
 - Gdje trošim najviše vremena?
 - Moduli (biblioteke)
 - Funkcije
 - Petje
- Utjecaj memorijske hijerarhije
 - Prevelik udio promašaja u priručnoj memoriji?
 - Lokalitet podataka?
- Vanjski resursi
 - I/O?
 - Sistemske biblioteke?



Multimedejske arhitekture i sustavi

18

Tipovi *profiling-a*

- Statističko uzorkovanje (*statistical sampling*)
 - Periodičko prekidanje izvođenja i pamćenje lokacije
 - Analiza statističke distribucije
 - Vremensko skupljanje podataka (vrijeme)
 - Uniforman *overhead*
- Praćenje događaja (*event tracing*)
 - Skupljanje individualnih događaja (pozivi funkcija, razmjena poruka)
 - Detaljna analiza događaja
 - Velika količina podataka i *overhead*

Multimedejske arhitekture i sustavi

19

Gnu profiler: gprof



■ Unix/Linux

- `gcc -pg program.cpp -o program`
- `./program`
- `gprof program > program.prof`

```
Flat profile:
Each sample counts as 0.01 seconds.
% cumulative self           self      total
time   seconds   seconds    calls us/call us/call name
 7  92.63     2.75     2.75    4096  671.68  671.68 dot()
 7  2.69     2.83     0.08   12288   6.51   6.51 pipe(rliststruct*, pestruct*)
 8  1.68     2.88     0.05   4096  12.21  12.21 rgh2yvr()
 9  1.35     2.92     0.04   12288   3.26   3.26 runlen(int*, rliststruct*, int)
10  0.67     2.94     0.02  823296   0.02   0.04 codeinsert(int)
11  0.34     2.95     0.01  823296   0.01   0.01 status(pestruct*, int*, int)
12  0.34     2.96     0.01 113555   0.09   0.09 order(char*, int*, int, int, int*)
13  0.34     2.97     0.01   4096   2.44   2.44 getblock(int)
14  0.00     2.97     0.00 12288   0.00   0.00 zigzag(int*)
15  0.00     2.97     0.00 12288   0.00   0.00 entropy(int*, int)
16
17
```

GNU profiler: gprof



- *Flat profile*
 - Provedeno vrijeme u svakoj funkciji
 - Broj poziva funkcija
 - Jednostavan pronašao vremenski kritičnih funkcija
- *Call graph*
 - Analiza po funkcijama (tko je pozvao, koga sam pozvao, koliko puta)
 - Procjena koliko vremena je provedeno u pozvanim funkcijama
 - Potencijalna mjesta za uklanjanje poziva

Multimedejske arhitekture i sustavi

21

Vtune Amplifier



■ Performanse CPU, GPU

■ Skalabilnost, Propusnost, Dretve

■ Vizualizacija rezultata

■ Hotspot

- Mjesto u programu s izraženom aktivnosti
 - Vrijeme
 - Pristup memoriji

Multimedejske arhitekture i sustavi

22

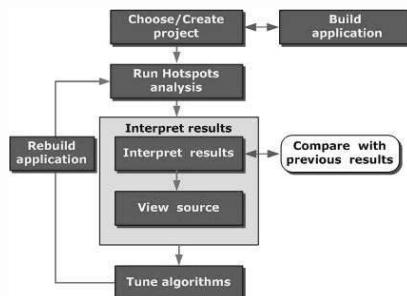
Vtune Amplifier

- Identifikacija *hotspot-ova*
 - Identifikacija neučinkovitih dijelova programa
 - Identifikacija dijelova koji su pogodni za optimizaciju
 - Analiza sinkronizacijskih objekata koji utječu na iskorištenost sustava
 - Analiza I/O operacija
 - Aktivnost dretvi i tranzicije
 - Sklopovalski kritične točke u programu

Multimedijiške arhitekture i sustav

2

Tijek izvođenja



Multimedijijske arhitekture i sustav

2

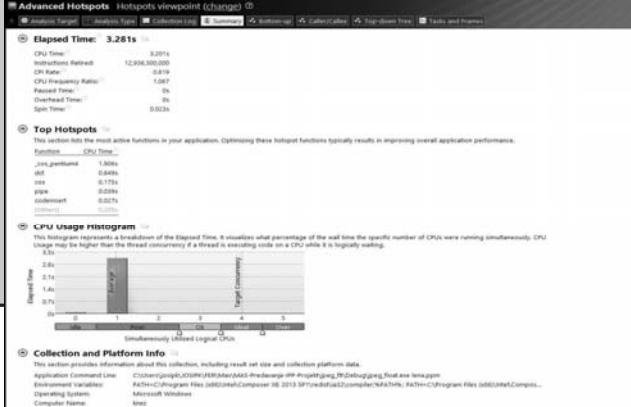
Terminologija

- **Target:** izvršni program koji se analizira
 - **Baseline:** osnovna mjera i model koji se koristi za usporedbu
 - **CPU time:** vrijeme kojeg program izvodi na procesoru (za više dretvi, CPU vremena svih dretvi su zbrojena u programsko CPU vrijeme)
 - **Elapsed time:** ukupno vrijeme (wall clock time) potrebljano za izvođenje programa
 - **Hotspot:** Dio programa sa značajnim doprinosom u ukupnom vremenu izvođenja programa
 - **CPI rate:** broj taktova po instrukciji (*clocks per instruction*)

Multimedijiške arhitekture i sustavi

24

Interpretacija rezultata (1)



Interpretacija rezultata (2)

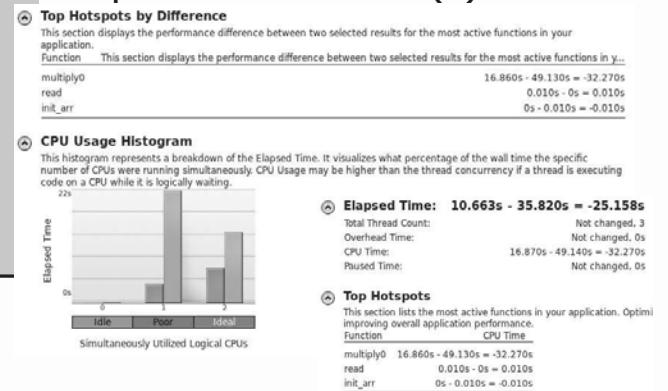


Usporedba rezultata

- Usporedba rezultata prije i poslije optimizacije
 - Kliknuti na 
 - Izabrati rezultate dviju analiza



Usporedba rezultata (2)

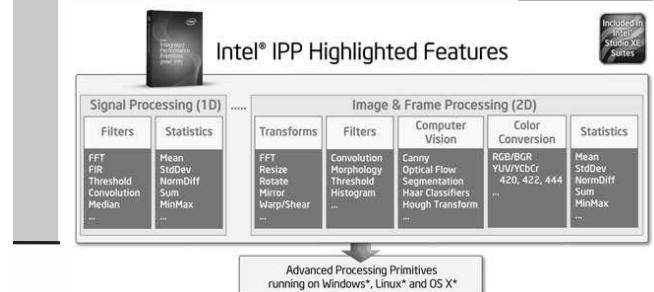


Integrated Performance Primitives IPP

- Biblioteka optimiranih funkcija za multimediju, obradu i kodiranje audio i video podataka, vektorsku manipulaciju, konverziju, kriptografiju itd.
- Zasniva se na apstrakciji procesorskih svojstava kao što su MMX, SIMD, SSE.
- Nedostatak: Ovisnost o arhitekturi procesora i ekstenzija



Integrated Performance Primitives IPP



Integrated Performance Primitives IPP

Code of Domain	Header file	Prefix	Description
CC	ippcc.h	ippi	color conversion
CH	ippeh.h	ipps	string operations
CORE	ippcore.h	ipp	core functions
CP	ippcp.h	ipps	cryptography
CV	ippcv.h	ippi	computer vision
DC	ippdc.h	ipps	data compression
I	ippi.h	ippi	image processing
S	ipps.h	ipps	signal processing
VM	ippvm.h	ipps	vector math
E*	ippe.h	ipps	embedded functionality

* available only within the Intel® System Studio suite

Intel IPP

■ Imenovanje funkcija:

ipp<data-domain><name>_<datatype>[_<descriptor>](<parameters>);

■ Primjer:

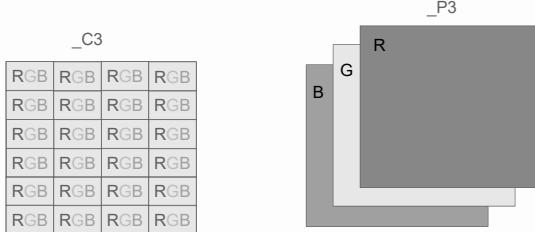
ippiRGBToYUV_8u_C3R();

Naziv funkcije: Pretvorba RGB u YUV
IPP funkcija, i: image processing

Tip podataka
Format zapisa podataka

Intel IPP – Image processing

- Format zapisa: raspored komponenti slike podataka
 - Kanalni raspred (Channel Data Layout)
Oznaka: “_C”
 - Planarni format (Planar Data Layout)
Oznaka: “_P”



IPP – Korisne funkcije

IppStatusippiDCT8x8FwdLS_<mod>()

IppStatusippiDCT8x8Inv_<mod>()

Više u opsežnoj dokumentaciji



<https://software.intel.com/en-us/ipp-9.0-reference-manual>

Jednoprocесорски системи

Problemi sa skaliranjem uniprocesorske arhitekture:

- Disipacija snage
- Efikasnost
- Kompleksnost
- Kašnjenje u interkonekcijama
- Usporen rast u performansama
- ILP – ne može više dodatno povećavati performanse

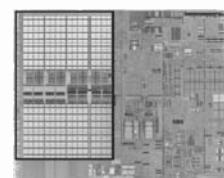
Primjer: RGB → YUV

- IppStatusippiRGBToYUV_<mod>
(const Ipp8u* pSrc, int srcStep, Ipp8u* pDst, int dstStep,
IppiSize roiSize);
 - <mod>: 8u_C3R, 8u_AC4R
- IppStatusippiRGBToYUV_8u_P3R
(const Ipp8u* const pSrc[3], int srcStep, Ipp8u* pDst[3], int
dstStep, IppiSize roiSize);
- IppStatusippiRGBToYUV_8u_C3P3R
(const Ipp8u* pSrc, int srcStep, Ipp8u* pDst[3], int
dstStep, IppiSize roiSize);

Višejezgrene arhitekture

Što se desilo sa uniprocesorskom arhitekturom (Primjer:
Intel Tejas i Jayhawk – Otkazani!)

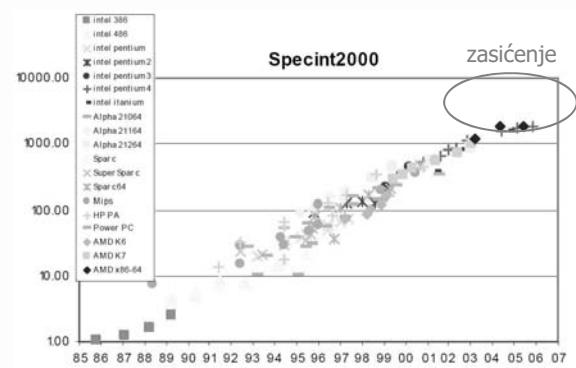
The Inquirer: The immediate successor to Prescott after it tops out at 5.20GHz will be the “Tejas” core, also produced on a 90 nanometer process and delivering 5.60GHz using a 1066MHz system bus. That’s slated to start appearing towards the end of 2004. Tejas will increase in steady increments which appear to be 6GHz, 6.40GHz, 6.80GHz, 7.20GHz, 7.60GHz, 7GHz, 8.40GHz, 8.80GHz and topping out at 9.20GHz. The first Nehalem is supposed to appear at 9.60GHz before Intel succeeds in its goal to produce a 10GHz+ chip, the Nehalem, and using a 1200MHz front side bus.



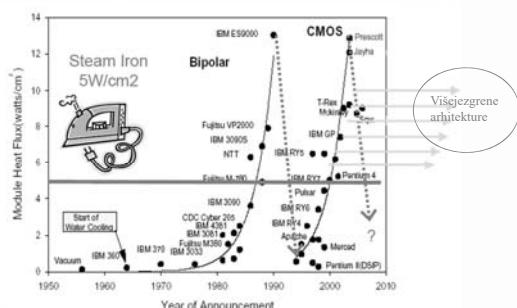
Intel Presscott:

- 30 stupnjeva protočne arhitekture?
- Velika disipacija snage
- Veliki udio L1 i L2 cache memorije u površini sklopa
- Wire delay

Višejezgrene arhitekture

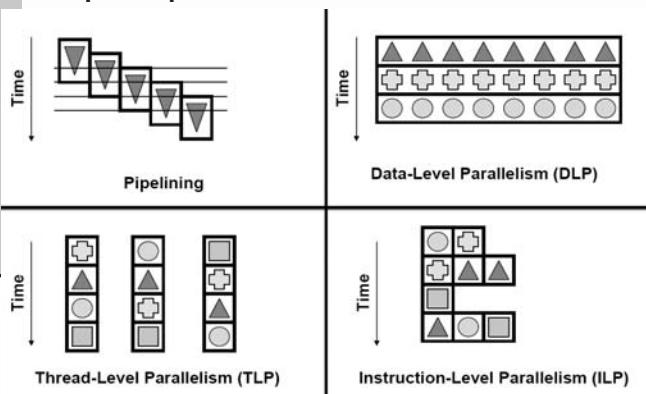


Disipacija



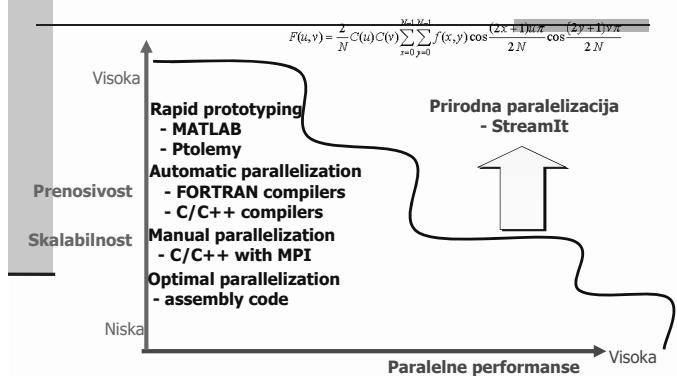
Dr. Michael Perrone – Introduction to the Cell Processor
MIT Class 6.189 Multicore Programming Primer

Tipovi paralelizma



Prof. Saman Amarasinghe, MIT

DSL pristup



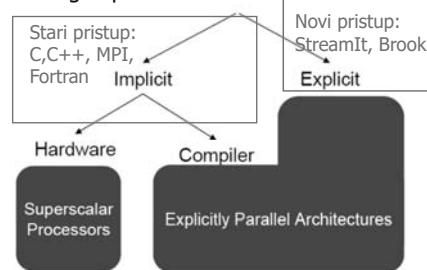
<http://cag.csail.mit.edu/streamit>

Multimedija na višejezgrevim arhitekturama

- PARALELIZAM U MM APLIKACIJAMA !!!
- Podatkovni paralelizam DLP (data level)
- Paralelizam među zadatcima TLP (task, thread level)
- Protočni paralelizam PLP (pipeline level)
- Instrukcijski paralelizam ILP (instruction level)

Paralelizam u MM aplikacijama

Kako ga opisati i iskoristiti



Stari pristup:
• Neprirodno
• Neprenosivost
• Verifikacija!

Novi pristup:
• Prirodno
• Prenosivost
• Verifikacija!

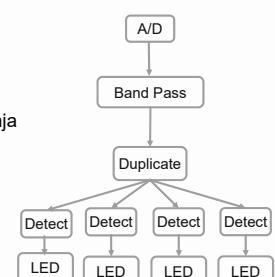
<http://cag.csail.mit.edu/streamit>

StreamIt

(<http://cag.csail.mit.edu/streamit>)

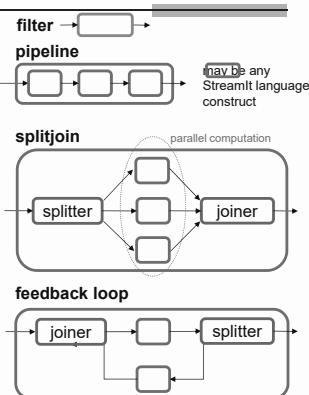
- SDF Model [Lee_87]
- Graf autonomnih aktora
- Aktori imaju lokaliziran adresni prostor
- Komunikacija preko FIFO kanala
- Statički definirana brzina I/O
- Prevodioc odreduje redoslijed izvođenja

StreamIt



Programski jezik StreamIt

- SDF sa dinamičkim proširenjima
 - Paralelizam i komunikacija:
eksplicitno izraženi u programu
 - Neovisnost o arhitekturi
 - Portabilnost
 - Modularnost (lako slaganje filtera
u složenije grafove toka)
 - Skalabilnost
 - Osnovne konstrukcije:
 - Filter (osnovna jedinica)
 - Pipeline (sekvenca filtera)
 - Splitjoin (scatter-gather)
 - Feedback loop



StreamIt - Filter

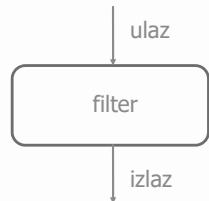
Osnovna računska jedinica

Svaki filter posjeduje work funkciju

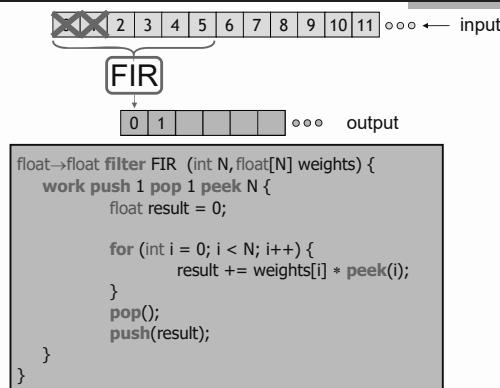
Svaki filter ima ulaznu traku i izlaznu traku

Svaki filter definira brzinu pristupa ulaznoj i izlaznoj traci (pop, push i peek rate)

```
float→float filter sum2elems () {  
    work pop 2 push 1() {  
        int first = pop();  
        int second = pop();  
        push(first + second);
```



StreamIt filter (FIR primjer)



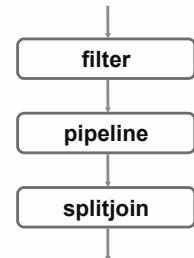
4. StreamIt - pipeline

Slijedni niz komponenti

Izlaz jedne komponente spojen na ulaz sljedeće u nizu

Izražavanje protočnog paralelizma

```
float→float pipeline 2D_iDCT (int N)
{
    add Column_iDCTs(N);
    add Row_iDCTs(N);
}
```



StreamIt – splitjoin

Scatter-gather struktura

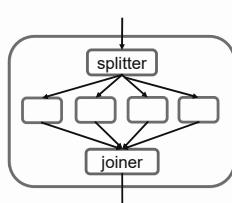
Izražavanje DLP i TLP paralelizma

Postoji više načina razvrstavanja i skupljanja podataka (duplicate, roundrobin)

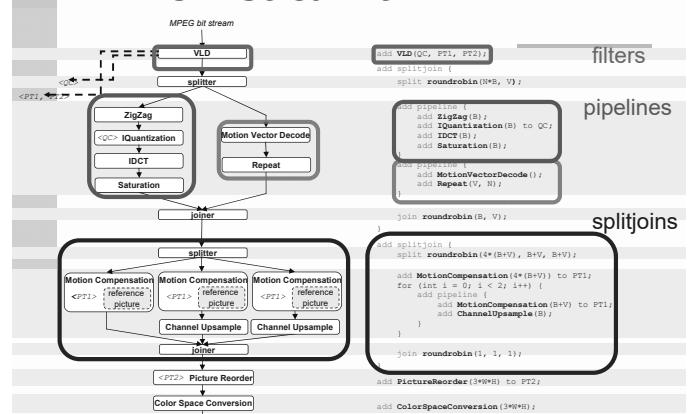
```

float→float splitjoin Row_iDCT (int N)
{
    split roundrobin(N);
    for (int i = 0; i < N; i++)
        add 1D_iDCT(N);
    }
    join roundrobin(N);
}

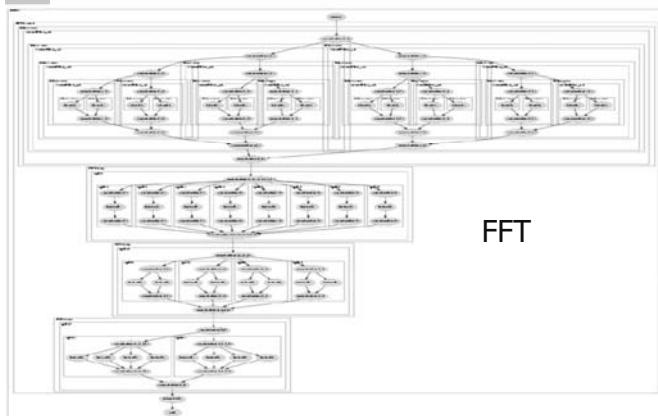
```



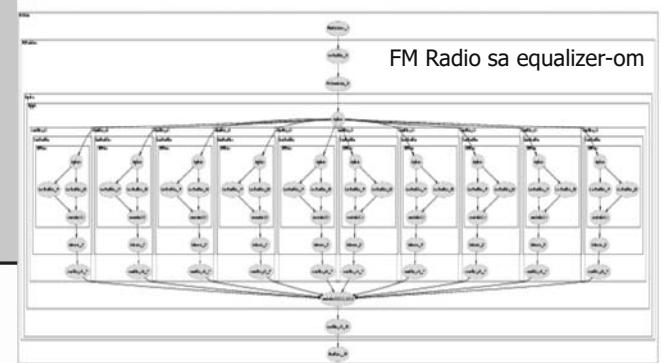
MPEG-2 StreamIt



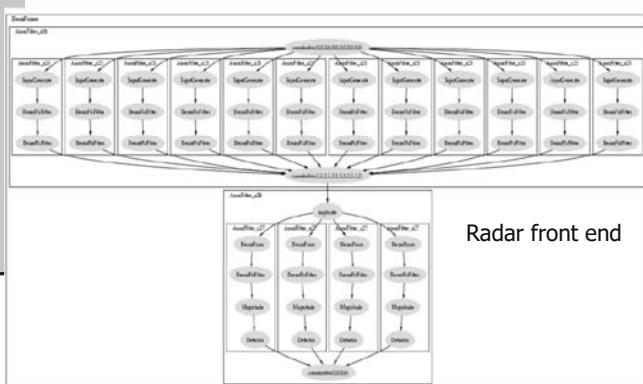
StreamIt – Zaključak



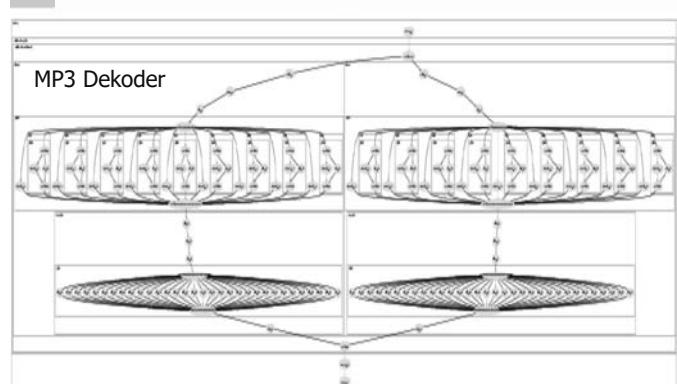
StreamIt – Zaključak



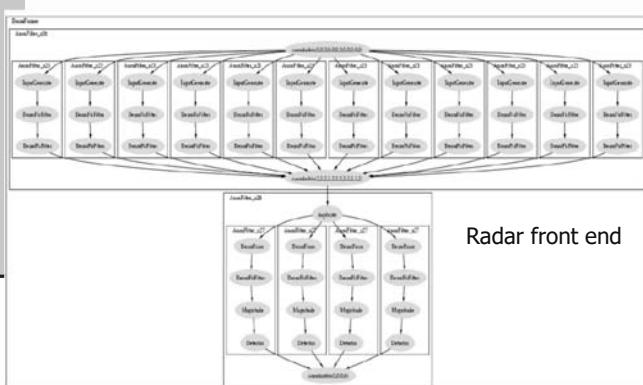
StreamIt – Zaključak



StreamIt – Zaključak



StreamIt – Zaključak

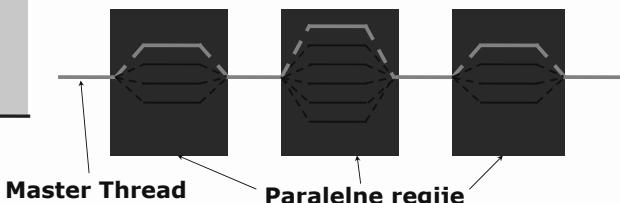


OpenMP

- Skup direkтива i rutina (biblioteka): omogućuje portabilne paralelne aplikacije na SMP arhitekturama
- C/C++, Fortran
- Direktive prevodiocu (compiler directives)
- Data parallelism model i Task parallelism
- Inkrementalni paralelizam
- Kombinira serijski i paralelni kod

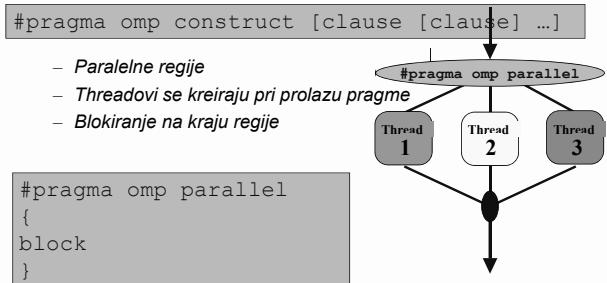
OpenMP biblioteka

- Fork-join paralelizam (zasnovan na multithreadingu)
- Master thread pokreće druge threadove
- Inkrementalno dodavanje paralelizma – sekvenčinalni program evoluira u paralelni



OpenMP biblioteka - sintaksa

Većina OpenMP ključnih riječi je u formi direktiva prevodiocu ili praga



OpenMP biblioteka

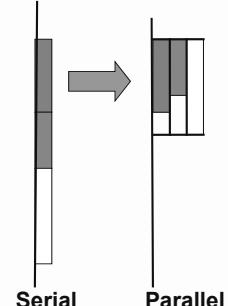
- Broj niti: Environment varijabla OMP_NUM_THREADS=4
- Intel prevodioč: #threadova = #procesora
- Primjer:

```
float dot_prod(float* a, float* b, int N)
{
    float sum = 0.0;
#pragma omp parallel for shared(sum)
    for(int i=0; i<N; i++) {
#pragma omp critical
        sum += a[i] * b[i];
    }
    return sum;
}
```

OpenMP – Paralelne sekcije

Nezavisni dijelovi koda koji se mogu izvoditi paralelno

```
#pragma omp parallel sections
{
    #pragma omp section
    phase1();
    #pragma omp section
    phase2();
    #pragma omp section
    phase3();
}
```



OpenMP sinkronizacija

Međusobna isključivost

- #pragma omp critical

Barijera

- Svaka nit čeka dok sve niti ne dođu
- Implicitne barijere:
 - #pragma omp for – na kraju pragma konstrukcije
 - #pragma omp parallel – na kraju pragma konstrukcije
- Eksplisitne barijere:
 - #pragma omp barrier

OpenCL

Heterogeni multiprocesorski sustavi

- CPU, GPU, akcelerator

Model podatkovnog paralelizma (CUDA)

Model paralelizma među zadacima

- Thread -> Work item
- Thread block -> Work group

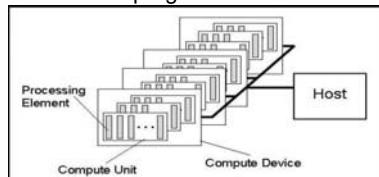
Paralelizam, paralelizam, paralelizam

- Tisuće niti za punu paralelizaciju
- Fokus na “on-chip” memoriju i komunikaciju

OpenCL

Heterogeni multiprocesorski sustavi

- OpenCL device: skup jedne ili više računskih jedinica (compute unit: host + devices (x86 host + GPU device))
- Compute unit: Skup procesnih elemenata (processing elements)
- Processing elements: izvode programski kod



OpenCL izvedbeni model

Kernel (jezgra)

- Osnovna jedinica izvedbenog koda (C funkcija)
- Podatkovno-paralelan, zadatkovno-paralelan Program
- Skup jezgri i drugih funkcija (dll)

Aplikacija

- Kreira kontekst za upravljanje i izvođenje OpenCL jezgri, razmjenu podataka host-device
- Rep instanci OpenCL jezgri

OpenCL program

Dinamički model prevođenja (OpenGL)

- API pozivi koji omogućuju dinamičku optimizaciju za postojeći OpenCL device

Kreiranje jezgre:

1. OpenCL program spremljen u tekstualnom obliku (učitan u memoriju)
2. Kreiranje programa, API poziv `clCreateProgramWithSource()`
3. Prevodenje programa za neki OpenCL uređaj, API poziv `clBuildProgram()`
 - x86: instrukcijski kod
 - GPU: IL (PTX) reprezentacija programa
4. Ekstrakcija jezgre, API poziv `clCreateKernel()`
5. Prijenos argumenata i "dispatching", API `clSetKernelArg()` i `clEnqueueNDRangeKernel()`



Energy-efficient bcrypt cracking

Katja Malvoni

(kmalvoni at openwall.com)

Solar Designer

(solar at openwall.com)

Openwall

<http://www.openwall.com>

August 6, 2014

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

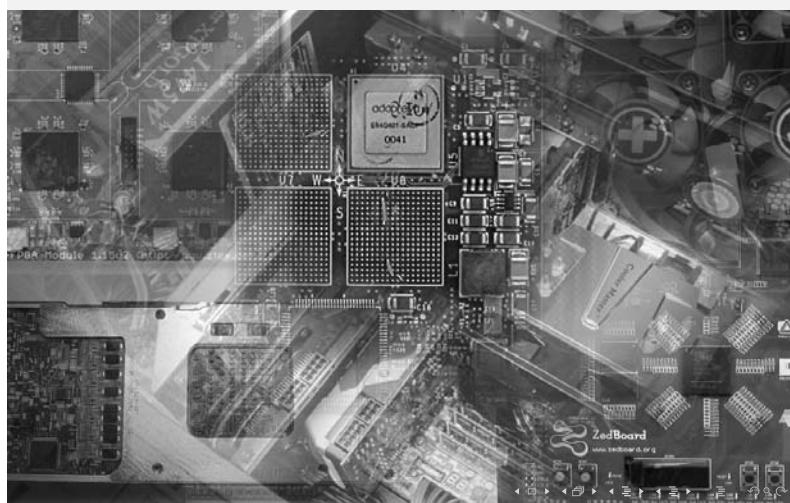
August 6, 2014

1 / 61

Motivation

- Bcrypt is:
 - ▶ Slow
 - ▶ Sequential
 - ▶ Designed to be resistant to brute force attacks and to remain secure despite hardware improvements
- You could almost think why even bother optimizing

But



Outline

① Bcrypt

② Implementation on different hardware

- Parallel/Ephipany
- ZedBoard
- ZC706
- Xeon Phi
- Haswell

③ Power consumption

④ Demo

⑤ Future work

⑥ Takeaways

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 4 / 61

Bcrypt

Bcrypt

- Based on Blowfish block cipher
- Expensive key setup
- User defined cost setting
 - ▶ Cost setting between 4 and 31 inclusive is supported
 - ▶ Cost 5 is traditionally used for benchmarks for historical reasons
 - ▶ All given performance figures are for bcrypt at cost 5
 - ▶ Current systems should use higher cost setting
- Pseudorandom memory accesses
- Memory usage
 - ▶ 4 KB for four S-boxes
 - ▶ 72 B for P-box

Blowfish. Photo source: <http://wallpapers.free-review.net>

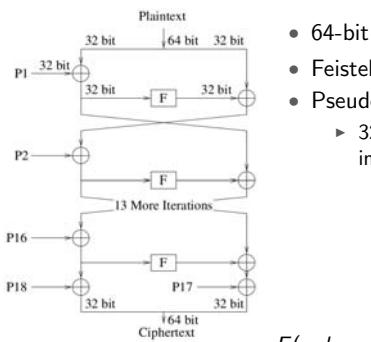
Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 5 / 61

Bcrypt

Blowfish encryption



EksBlowfish

Expensive key schedule Blowfish



Algorithm 1 EksBlowfishSetup(cost, salt, key)

```
1: state ← InitState()
2: state ← ExpandKey(state, salt, key)
3: repeat(2cost)
4:   state ← ExpandKey(state, 0, salt)
5: state ← ExpandKey(state, 0, key)
6: return state
```

• Order of lines 4 and 5 is swapped in implementation

Niels Provos and David Mazieres, "A Future-Adaptable Password Scheme", The OpenBSD Project, 1999

Bcrypt

bcrypt



Algorithm 2 bcrypt(cost, salt, pwd)

```
1: state ← EksBlowfishSetup(cost, salt, key)
2: ctext ← "OrpheanBeholderScryDoubt"
3: repeat(64)
4:   ctext ← EncryptECB(state, ctext)
5: return Concatenate(cost, salt, ctext)
```

mode	cost	salt	hash
\$2a	\$12	\$GhvMmNVjRW29ulnudlLbuAnUtIN/LRfe1JsBm1Xu6LE3059zTr8m	

Niels Provos and David Mazieres, "A Future-Adaptable Password Scheme", The OpenBSD Project, 1999

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 8 / 61

Implementation on different hardware Parallela/Ephiphy

Outline

① Bcrypt

② Implementation on different hardware

- Parallel/Ephipany
- ZedBoard
- ZC706
- Xeon Phi
- Haswell

③ Power consumption

④ Demo

⑤ Future work

⑥ Takeaways

Katja Malvoni and Solar Designer

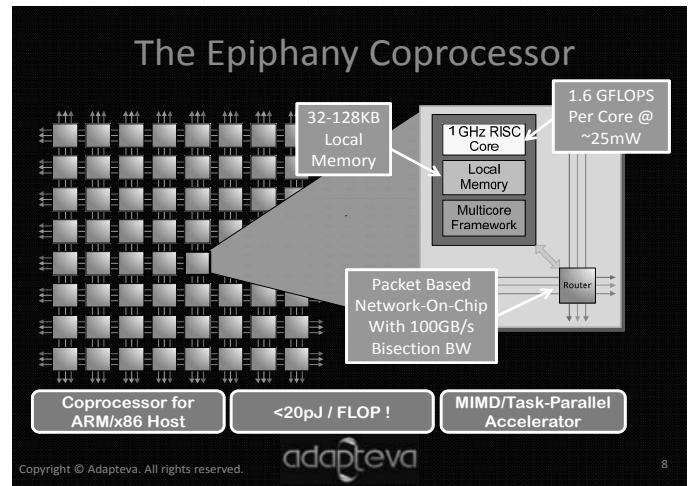
Energy-efficient bcrypt cracking

August 6, 2014 9 / 61

Architecture

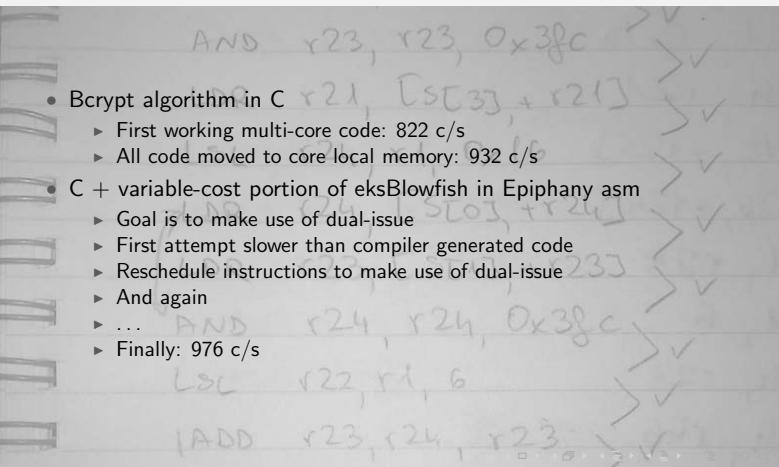
Epiphany

- 16/64 32-bit RISC cores operating at up to 1 GHz/800 MHz
 - ▶ Chips used in our testing operate at 600 MHz
- Pros
 - ▶ **Energy-efficient** - 2 W maximum chip power consumption
 - ▶ 32 KB of local memory per core
 - ▶ 64 registers
 - ▶ FPU can be switched to integer mode
 - ▶ Dual-register (64-bit) load/store instructions
 - ▶ Ability for cores and host to directly address other cores' local memory
- Cons
 - ▶ FPU in integer mode can issue only add and mul instructions
 - ▶ Only simple addressing modes
 - Index scaling would be helpful for S-box lookups



Implementation

One bcrypt instance per core on E16



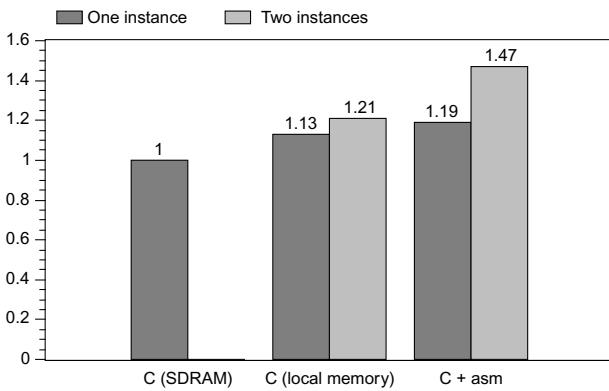
Implementation

Two bcrypt instances per core on E16

- Two instances because:
 - ▶ Instructions executed on FPU have 4 cycles latency
 - ▶ Single bcrypt instance doesn't have enough parallelism to hide this
 - ▶ Adding second instance brings sufficient amount of parallelism down to instruction level to hide those latencies
- Bcrypt algorithm in C
 - ▶ 947 c/s
 - ▶ Preload P-boxes: 996 c/s
- C + variable-cost portion of eksBlowfish in Epiphany asm
 - ▶ 1194 c/s
 - ▶ Transfer keys only when changed: 1207 c/s
 - When cracking multiple hashes, with different salts
- And 227 e-mails on the john-dev mailing list

Implementation

Speedup between different E16 implementations



Implementation

Epiphany asm

```
.macro BF2_2ROUND_B P1, P2, P3, P4
    and tmpa1, L0, c1
    lsr tmpa3, L0, 0xe
    and tmpa3, tmpa3, c2
    lsr tmpa4, L0, 0x1e
    and tmpa4, tmpa4, c2
    imul tmpa1, tmpa1, c3
    ldr tmpa3, [S01, +tmpa3]
    ldr tmpa4, [S00, +tmpa4]
    lsr tmpa2, L0, 6
    and tmpa2, tmpa2, c2
    iadd tmpa3, tmpa4, tmpa3
    ldr tmpa2, [S02, +tmpa2]
    ldr tmpa1, [S03, +tmpa1]
    lsr tmpa1, L1, \px18
    eor R0, R0, \P1
    eor tmpa3, tmpa2, tmpa3
    imul tmpb4, tmpb4, c3
    and tmpb1, L1, c1
    lsr tmpb3, L1, 0xe
    and tmpb3, tmpb3, c2
    iadd tmpa3, tmpa3, tmpa1
    imul tmpb1, tmpb1, c3
    ldr tmpb2, [S11, +tmpb3]
    ldr tmpb4, [S10, +tmpb4]
    imul tmpb3, tmpb3, c3
    lsr tmpb1, L1, 6
    and tmpa1, tmpa1, c2
    eor R0, R0, tmpa3
    iadd tmpb3, tmpb4, tmpb3
    ldr tmpa1, [S12, +tmpa1]
    ldr tmpb1, [S13, +tmpb1]
    lsr tmpa4, R0, 0x18
    eor R1, R1, \P2
    eor tmpb3, tmpa1, tmpb3
    imul tmpa4, tmpa4, c3
    .endm
```

Implementation

Ephipany asm

```

loop2:
    eor L0, P00, L0
    eor L1, P10, L1
    BF2_2ROUND_B P01, P02, P11, P12
    BF2_2ROUND_B P03, P04, P13, P14
    BF2_2ROUND_B P05, P06, P15, P16
    BF2_2ROUND_B P07, P08, P17, P18
    BF2_2ROUND_B P09, P010, P19, P10
    BF2_2ROUND_B P011, P012, P111, P112
    BF2_2ROUND_B P013, P014, P113, P114
    BF2_2ROUND_B P015, P016, P115, P116
    eor tmpa2, R0, P017
    strd tmpa2, [ptr0], +0x1
    eor tmpa3, R1, P117
    strd tmpa3, [ptr1], +0x1
    mov R0, L0
    mov R1, L1
    mov L0, tmpa2
    mov L1, tmpa3
    sub tmpa4, end, ptr0
    bgtu loop2

```

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 16 / 61

Implementation

Summary

- Two bcrypt instances per core
- Optimized in assembly
- Dual-issue
 - ▶ Integer ALU
 - ▶ FPU in integer mode
- Integrated into John the Ripper

```

malvoni@linaro-ubuntu-desktop:~/JohnTheRipper/run$ sudo -E LD_LIBRARY_PATH=$LD_LIBRARY_PATH ./john -test -format:bcrypt-parallella
Benchmarking: bcrypt-parallella, OpenBSD Blowfish ("$2a$05", 32 iterations) [Parallel]... DONE
Raw: 1207 c/s real, 1207 c/s virtual

```

```

linaro@linaro-ubuntu-desktop:~/JohnTheRipper/run$ sudo -E LD_LIBRARY_PATH=$LD_LIBRARY_PATH ./john -test -format:bcrypt-parallella
Benchmarking: bcrypt-parallella, OpenBSD Blowfish ("$2a$05", 32 iterations) [Parallel]... DONE
Raw: 4812 c/s real, 4812 c/s virtual

```

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 17 / 61

Performance

Ephipany 16

- 1207 c/s
- ~600 c/s per Watt
- We achieved 3/4th of the per-MHz per-core speed of a full integer dual-issue architecture



Parallelia Board. Photo (c) Adapteva, reproduced under the fair use doctrine

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 18 / 61

Performance

Ephipany 64

- 4812 c/s
- ~2400 c/s per Watt for E64 chip
- ~1000 c/s per Watt for Parallelia board with E64
 - ▶ When not yet using the ARM cores and FPGA PL for computation
- Scalability
 - ▶ $4812/1207 = 3.987x$ faster
 - ▶ 99.7 % efficiency
 - $4812/1207/4 = 0.9967$

Ephipany 16

#define EPIPHANY_CORES 16

Ephipany 64

#define EPIPHANY_CORES 64

ZedBoard + FMC with E16 or E64 - prototyping

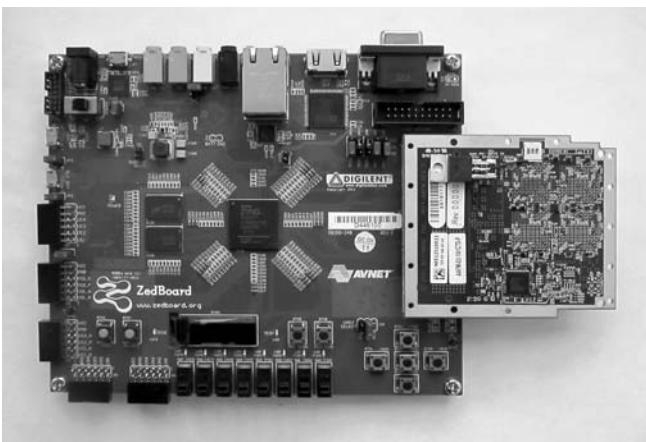


Photo (c) Adapteva, used with permission

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 20 / 61

Outline

- ① Bcrypt
- ② Implementation on different hardware
 - Parallel/Ephipany
 - ZedBoard
 - ZC706
 - Xeon Phi
 - Haswell
- ③ Power consumption
- ④ Demo
- ⑤ Future work
- ⑥ Takeaways

Katja Malvoni and Solar Designer

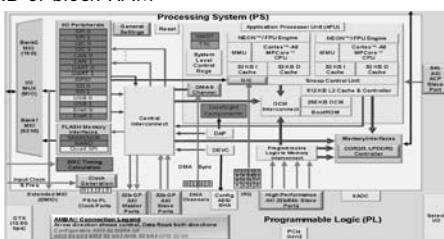
Energy-efficient bcrypt cracking

August 6, 2014 21 / 61

Architecture

Zynq 7020

- Dual ARM Cortex-A9 MPCore
 - ▶ 667 MHz
 - ▶ 256 KB on-chip memory
- Advanced low power 28nm programmable logic
 - ▶ 85 K logic cells
 - ▶ 560 KB of block RAM



Zynq diagram. Screenshot from Xilinx Platform Studio, reproduced under the fair use doctrine

Katja Malvoni and Solar Designer

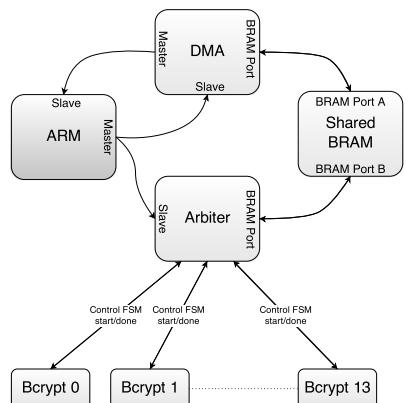
Energy-efficient bcrypt cracking

August 6, 2014 22 / 61

Implementation

CPU/FPGA communication

- General Purpose Master AXI interface
- Accelerator Coherency Port Slave AXI interface
- DMA from on-chip Memory to BRAM
- Cores copy relevant data from Shared BRAM one by one



Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 23 / 61

Implementation

5 cycles per round

- Non-optimal implementation: 13.9 c/s
 - ▶ S-boxes are stored in shared BRAM so only one port is used for load
- Copy S-boxes to another dual port BRAM and use both ports: 23.5 c/s
- Per-cycle summary
 - ▶ Cycle 0: initiate 2 S-box lookups
 - ▶ Cycle 1: wait
 - ▶ Cycle 2: initiate other 2 S-box lookups, compute tmp
 - ▶ Cycle 3: wait
 - ▶ Cycle 4: compute new L, swap L and R
- 14 cores fit: 311 c/s

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 24 / 61

Implementation

2 cycles per round

- Use two dual port BRAMs
 - ▶ Two S-boxes in one BRAM, two in the other
 - ▶ Load all 4 needed values in one cycle
 - ▶ Single core speed: 79 c/s
 - ▶ 14 cores still fit: 780 c/s
 - With no overhead, theoretical performance would be $79*14 = 1106$ c/s
 - With bcrypt cost setting above 5, efficiency is higher
- Per-cycle summary
 - ▶ Cycle 0: compute new R; swap L and R; initiate 4 S-box lookups
 - ▶ Cycle 1: wait

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 25 / 61

Implementation

Utilization summary

- Only most costly loop (Algorithm 1, lines 3, 4 and 5) implemented in FPGA
- 2 cycles per one Feistel network round
- 14 bcrypt cores at 100 MHz
- Utilization
 - ▶ Register: 19%
 - ▶ LUT: 90%
 - ▶ Slice: 98%
 - ▶ RAMB36E1: 11%
 - ▶ RAMB18E1: 5%
 - ▶ DSP48E1: 6%
 - ▶ BUFG: 3%
- Support logic utilizes most of the resources and limits clock rate

Katja Malvoni and Solar Designer

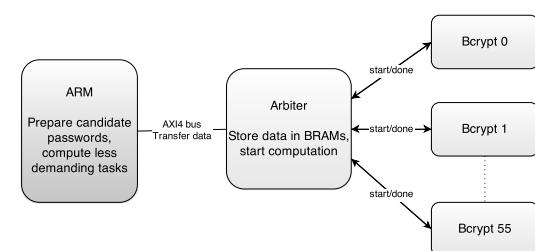
Energy-efficient bcrypt cracking

August 6, 2014 26 / 61

Implementation

Reduce support logic

- Avoid DMA and shared BRAM
- Data transfer via AXI bus directly to bcrypt cores
- Number of bcrypt instances running in parallel limited by available BRAM



Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 27 / 61

Energy-efficient platforms

Platform	Cores	Lithography	Performance	Efficiency	Chip power estimate	TDP	Idle to load delta	Full system
Epiphany 16	16c 600 MHz	65 nm	1207 c/s	600 c/s/W	2 W	2 W	1.3 W	9.1 W ¹
Epiphany 64	64c 600 MHz	28 nm	4812 c/s	2400 c/s/W	2 W	2 W		
Zynq-7020 ²	56c 71.4 MHz	28 nm	4571 c/s	2280 c/s/W	2 W		1 W	7 W ³
Zynq-7020 (emulated with 7045)	56c 71.4 MHz	28 nm	7044 c/s	3522 c/s/W ⁴	2 W			
Zynq-7045	196c 71.4 MHz	28 nm	20538 c/s	4116 c/s/W	5 W			

¹ZedBoard + FMC with E16 chip and glue logic, together simulating a Parallella board. The actual Parallella board should consume less power

²On our modified ZedBoard

³5.2 W consumed by the same ZedBoard, but with the FMC disconnected and FPGA bitstream replaced + 1 W consumed by 12 V fan added on the board + 0.8 W PSU overhead

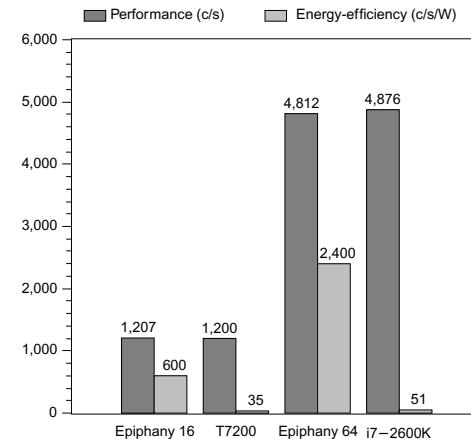
⁴If ZedBoard would be without hardware defects

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 46 / 61

Epiphany vs x86

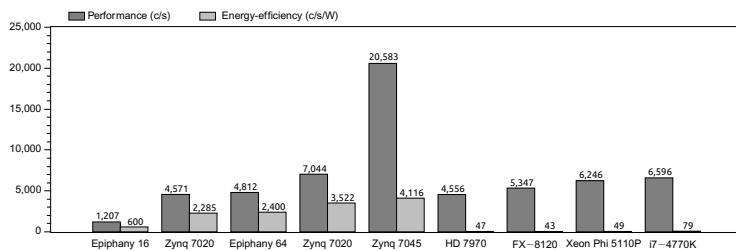


Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 47 / 61

Performance and efficiency comparison

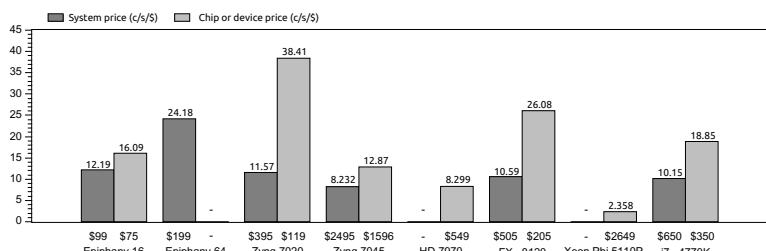


Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 48 / 61

Cost comparison

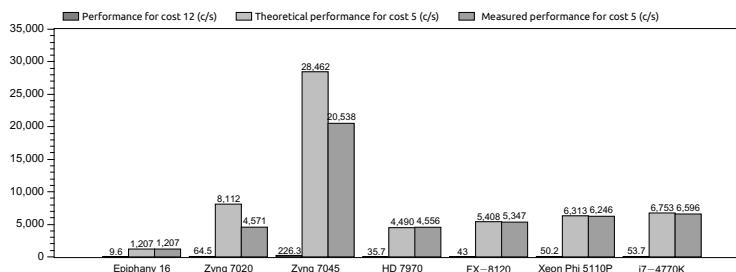


Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 49 / 61

Derived performance from cost 12



```
bcrypt(cost, salt, pwd)
1: state ← InitState()
2: state ← ExpandKey(state, salt, key)
3: repeat(2cost)
4:   state ← ExpandKey(state, 0, salt)
5:   state ← ExpandKey(state, 0, key)
6:   ctxt ← "OrpheanBeholderScryDoubt"
7:   repeat(64)
8:     ctxt ← EncryptECB(state, ctxt)
9:   return Concatenate(cost, salt, ctxt)
```

$$c/s = \frac{(2^{12} * 1024 + 585)}{(2^5 * 1024 + 585)} * performance_{12} \quad (4)$$

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 50 / 61

Theoretical Peak Performance Analysis Theory

$$c/s = \frac{N_{ports} * f}{(2^{cost} * 1024 + 585) * N_{reads} * 16} \quad (5)$$

- N_{ports} - number of available read ports to local memory or L1 cache
- N_{reads} - number of reads per Blowfish round
 - 4 or 5 depending on whether reads from P-boxes go from one of those read ports we've counted or from separate storage such as registers
- $2^{cost} * 1024 + 585$ - number of Blowfish block encryptions in bcrypt hash computation
- f (in Hz) - clock rate

```
bcrypt(cost, salt, pwd)
1: state ← InitState()
2: state ← ExpandKey(state, salt, key)
3: repeat(2cost)
4:   state ← ExpandKey(state, 0, salt)
5:   state ← ExpandKey(state, 0, key)
6:   ctxt ← "OrpheanBeholderScryDoubt"
7:   repeat(64)
8:     ctxt ← EncryptECB(state, ctxt)
9:   return Concatenate(cost, salt, ctxt)
```

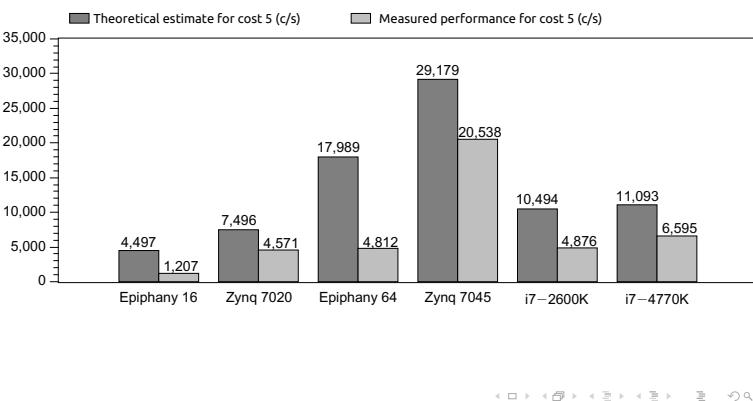
Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014 51 / 61

Theoretical Peak Performance Analysis

Comparison



Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014

52 / 61

Outline

- ① Bcrypt
- ② Implementation on different hardware
 - Parallel/Epiphanys
 - ZedBoard
 - ZC706
 - Xeon Phi
 - Haswell
- ③ Power consumption
- ④ Demo
- ⑤ Future work
- ⑥ Takeaways

Katja Malvoni and Solar Designer

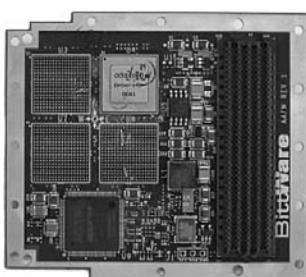
Energy-efficient bcrypt cracking

August 6, 2014

54 / 61

Parallel/Epiphanys

- Using both Epiphany and Zynq 7020 at once
- Chip to chip links for integrating up to 64 chips on a single board
- Scalability of current implementation is promising
- $64 * 64 = 4096$ cores with theoretical performance of 300000 c/s



FMC with E64 chip and pads for 3 more such chips. Photo (c) Adapteva, used with permission.

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014

56 / 61

Related work

- F.Wiemer, R. Zimmermann. Speed and Area-Optimized Password Search of bcrypt on FPGAs
- bcrypt running on ZedBoard at 80 MHz
- 40 parallel instances
- 5208 c/s at cost 5, 41.6 c/s at cost 12

Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014

53 / 61

Outline

- ① Bcrypt
- ② Implementation on different hardware
 - Parallel/Epiphanys
 - ZedBoard
 - ZC706
 - Xeon Phi
 - Haswell
- ③ Power consumption
- ④ Demo
- ⑤ Future work
- ⑥ Takeaways

Katja Malvoni and Solar Designer

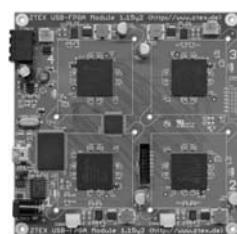
Energy-efficient bcrypt cracking

August 6, 2014

55 / 61

FPGA

- Zynq 7020 and 7045 optimizations
 - ▶ Improve clock rate
 - ▶ Reduce communication overhead
- Targeting bigger FPGAs
- Targeting multi-FPGA boards



ZTEX Board. Photo (c) ZTEX, reproduced under the fair use doctrine



Katja Malvoni and Solar Designer

Energy-efficient bcrypt cracking

August 6, 2014

57 / 61

Outline

- ① Bcrypt
- ② Implementation on different hardware
 - Parallelia/Epiphany
 - ZedBoard
 - ZC706
 - Xeon Phi
 - Haswell
- ③ Power consumption
- ④ Demo
- ⑤ Future work
- ⑥ Takeaways

Takeaways

- Many-core low power RISC platforms and FPGAs are capable of exploiting bcrypt peculiarities to achieve comparable performance and higher energy-efficiency
- Higher energy-efficiency enables higher density
 - ▶ More chips per board, more boards per system
- It doesn't take ASICs to improve bcrypt cracking energy-efficiency by a factor of 45+
 - ▶ Although ASICs would do better yet

Thanks

- Sayantan Datta
- Steve Thomas
- Parallelia project
- Google Summer of Code
- Xilinx
- Faculty of Electrical Engineering and Computing, University of Zagreb



[kmalvoni at openwall.com](mailto:kmalvoni@openwall.com)

Multimedejske arhitekture i sustavi

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

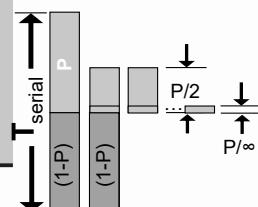


Amdahl's Law

Amdahl's Law:

- Is a law governing the *speedup* of using parallel processors on an application, versus using only one serial processor.
- Describes the upper bound of parallel execution speedup.

$n = 2$



$$T_{\text{parallel}} = \{(1-P) + P/n\} T_{\text{serial}}$$

$n = \text{number of processors}$
 $\text{Speedup} = T_{\text{serial}} / T_{\text{parallel}}$
 $1.0 / 0.5 = 2.00$

Parallel Efficiency

Parallel Efficiency:

- Is a measure of how efficiently processor resources are used during parallel computations.
- Is equal to $(\text{Speedup} / \text{Number of Threads}) * 100\%$.

CONCURRENCY != PARALLELISM



1/3/2009

3

Granularity

Definition:

- An approximation of the ratio of computation to synchronization.

The two types of granularity are:

- **Coarse-grained:** Concurrent calculations that have a large amount of computation between synchronization operations are known as *coarse-grained*.
- **Fine-grained:** Cases where there is very little computation between synchronization events are known as *fine-grained*.

4 1/3/2009

Summary

- A *thread* is a discrete sequence of related instructions that is executed independently. It is a single sequential flow of control within a program.
- The *benefits* of using threads are increased performance, better resource utilization, and efficient data sharing.
- The *risks* of using threads are data races, deadlocks, code complexity, portability issues, and testing and debugging difficulty.
- Every process has at least one thread, which is the main thread that initializes the process and begins executing the initial instructions.
- All threads within a process share code and data segments.
- Concurrent threads can execute on a single processor. Parallelism requires multiple processors.

5 1/3/2009

Summary (Continued)

- *Turnaround* refers to completing a single task in the smallest amount of time possible, whereas accomplishing the most tasks in a fixed amount of time refers to *throughput*.
- Decomposing a program based on the number and type of functions that it performs is called *functional decomposition*.
- Dividing large data sets whose elements can be computed independently, and associating the required computation among threads is known as *data decomposition* in multithreaded applications.
- Applications that scale with the number of independent functions are probably best suited to functional decomposition while applications that scale with the amount of independent data are probably best suited to data decomposition.
- *Race conditions* occur because the programmer assumes a particular order of execution but does not use synchronization to guarantee that order.

6 1/3/2009

Summary (Continued)

- Storage conflicts can occur when multiple threads attempt to simultaneously update the same memory location or variable.
- *Critical regions* are parts of threaded code that access (read or write) shared data resources. To ensure data integrity when multiple threads attempt to access shared resources, critical regions must be protected so that only one thread executes within them at a time.
- *Mutual exclusion* refers to the program logic used to ensure single-thread access to a critical region.
- *Barrier synchronization* is used when all threads must have completed a portion of the code before proceeding to the next section of code.
- *Deadlock* refers to a situation when a thread waits for an event that never occurs. This is usually the result of two threads requiring access to resources held by the other thread.

7 1/3/2009

Summary (Continued)

- *Livelock* refers to a situation when threads are not making progress on assigned computations, but are not idle waiting for an event.
- *Speedup* is the metric that characterizes how much faster the parallel computation executes relative to the best serial code.
- *Parallel Efficiency* is a measure of how busy the threads are during parallel computations.
- *Granularity* is defined as the ratio of computation to synchronization.
- *Load balancing* refers to the distribution of work across multiple threads so that they all perform roughly the same amount of work.

8 1/3/2009

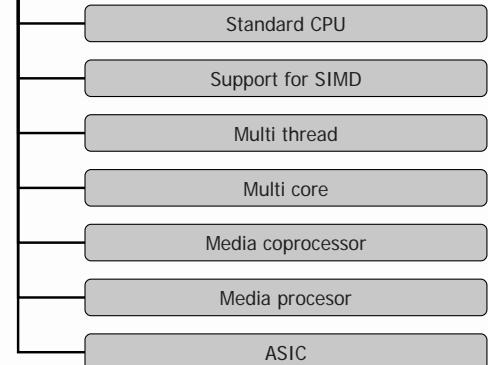
Optimizacije i arhitekture

Kako poboljšati performanse

9

Osnovni načini CPU podrške za MM

Načini podrške za MM



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, MLA)
 - Kod primjera računanja DCT, DFT i slično imamo većinu "leptir" operacija kod kojih je MLA osnovna karika

Standardni CPU ubrzanja

- Efikasno korištenje protočne strukture
 - Loop-unrolling
 - Branch prediction
-
- Nedovoljno !

SIMD

- Analizom mnogih aplikacija ustanovljeno:
 - Koje aplikacije su najzahtjevnije
 - Gdje je efikasnost najmanja
 - Koja su uska grla
 -
- Povećanje brzine nije rješenje
 - P4 fijasko
 - (enormni pipeline: energija, toplina, branch miss)
 - ARH1 !!

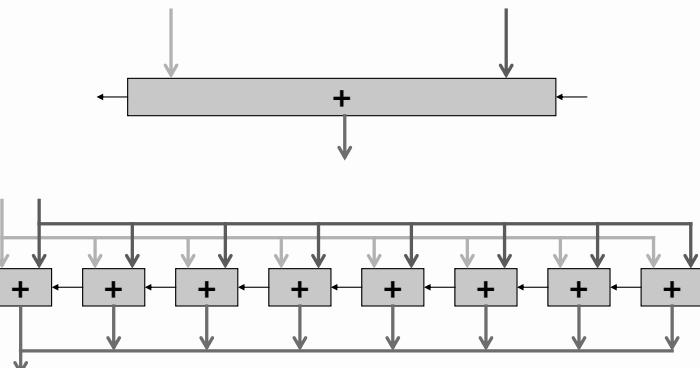
SIMD

- Ispada da su najzahtjevније aplikacije koje koriste vrlo jednostavne operacije i operande
 - 8 bita (video)
 - 16/32 bita audio
- ... Ali puuuuuno podataka

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"

ALU – obična i SIMD



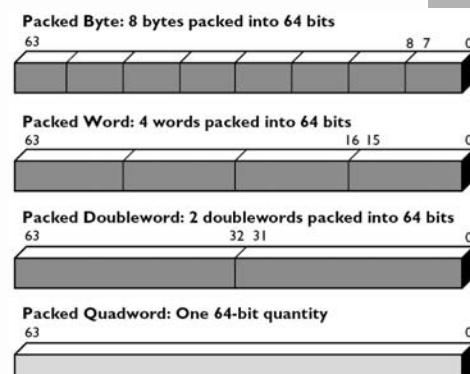
Najpoznatiji primjeri

- MMX
- Stavljen na tržište 1996 (Pentium sa MMX i Pentium II)
- Uveden novi tip podataka: Packed 64 bit
- Zašto 64 bita?
 - zadovoljavajuće ubrzanje
 - Ne treba velike zahvate na arhitekturi

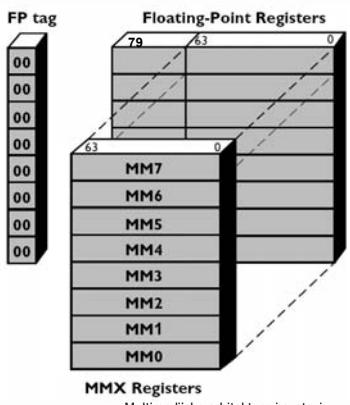
MM proširenja

- Ovo (osnovno) načelo služi kako bi se obrada multimedijiskih algoritama značajno ubrzala
- Primjeri:
 - Stari: VIS, PA-RISC
 - Ne tako stari: MMX, 3DNow!
 - Novi: SSE4

MMX tipovi podataka



Integracija u IA-32



21

MMX naredbe

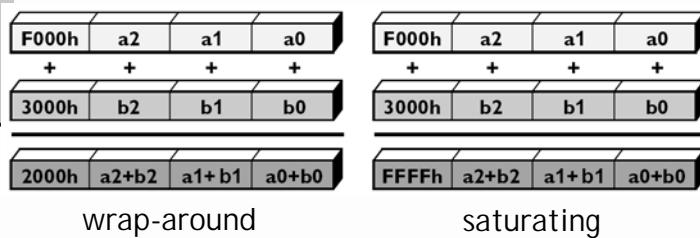
- 57 novih MMX naredaba
- Aritmetičke, logičke, usporedbe, ...
- Sa zasićenjem (signed i unsigned), bez zasićenja
- Sve naredbe osim MOVE rade nad MMX registrima
- Sve naredbe rade nad 16 bitovnim podacima a samo neke nad 8 i 32 bitovnim

Multimedijске arhitekture i sustavi

22

Aritmetika sa zasićenjem

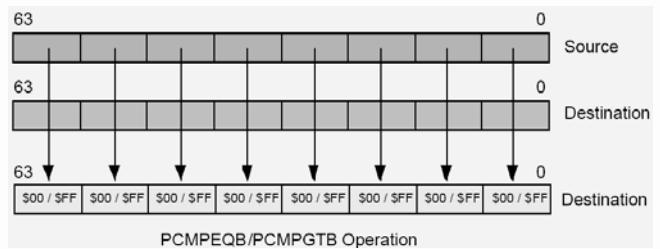
- Izuzetno korisno kod MM algoritama
- Značajno poboljšava performanse



23

Usporedba

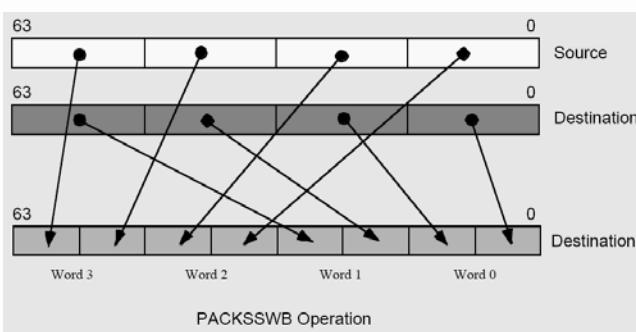
- Ne postavljaju se zastavice (jer ih nema) već rezultat



Multimedijске arhitekture i sustavi

24

Pack and Saturate Signed



Multimedijске arhitekture i sustavi

25

Primjer

Chromakey

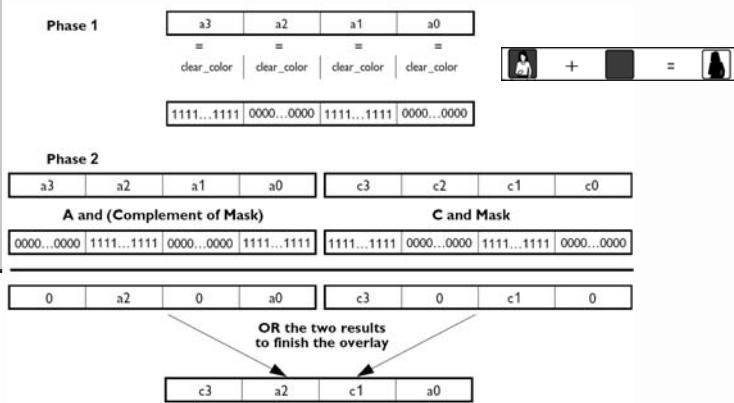
- Standardna funkcija kod TV aplikacija je preklapanje dvije slike
- Ostvaruje se na način da se komentatoru stavi unaprijed definirana pozadina koju računalo zamjenjuje sa drugom slikom
- Običnim računanjem jako sporo



Multimedijске arhitekture i sustavi

26

Primjer: preklapanje slike



Primjer: preklapanje slike

MOVQ mm0, kamera
MOVQ mm2, mm0
MOVQ mm4, karta
MOVQ mm1, plavo



PCMPEQW mm0, mm1

PAND mm4, mm0

PANDN mm0, mm2

POR mm0, mm4



Kompatibilnost sa IA?

- Ne zahtjeva promjene u OS
 - Nema novih stanja
 - OS ne zna za MMX već sve radi kao sa FP

Rezultati...

- Inicijalno velika medijska "buka" ali rezultati nisu zadovoljili na dulje vrijeme..
- Ubrzanja su dobra u odnosu na uloženo ali su uočene potrebe za poboljšanjem
- Nedostaci:
 - Ne mogu se koristiti paralelno FPU i INT
 - Nedovoljna podrška za 8 i 32 bitovne podatke
 - Samo integer aritmetika
 - OS nije problem osvježiti

3DNow!

- AMD-ova proširenja MMX-a
 - Predstavljena 1998.
 - Podrška za 32 bitovni FP
 - Neke od ovih naredaba Intel uveo u SSE
 - Kasnije prošireno na 3DNow!+

SSE

- Streaming SIMD Extensions
- Predstavljeno 1999 sa Pentium III
- Popravljeni najvažniji nedostaci MMX-a
- 8 potpuno novih 128 bitovnih registara XMM0..XMM7 (još 8, ..XMM15 u 64-bit)
- Omogućena 128-bit (4x32bit) SIMD FP podrška: aritmetika, usporedbe, shuffle,...
- Integer SIMD podrška nad 64 bita
- Preko 60 novih naredbi

SSE nedostaci

- Spremanje stanja registara loše izvedeno
- Resursi za izvođenje zajednički sa FPU

Multimedijске arhitekture i sustavi

33

SS.....

SSE2

- Pentium4(2001), 144 nove naredbe, 64 bitovni FP, proširenje MMX naredbi radi i sa XMM, performanse nisu naročito bolje od MMX

SSE3

- Pentium4(2004), horizontalne operacije, konverzije FP-INT jednostavnije,

SSSE3

- Intel Core, 16 novih naredaba, AMD ne podržava

Multimedijске arhitekture i sustavi

34

SSE4

- 54 nove naredbe
 - SSE4.1 (47), SSE4.2 (7)
- Bolja implementacija nekih operacija

Multimedijске arhitekture i sustavi

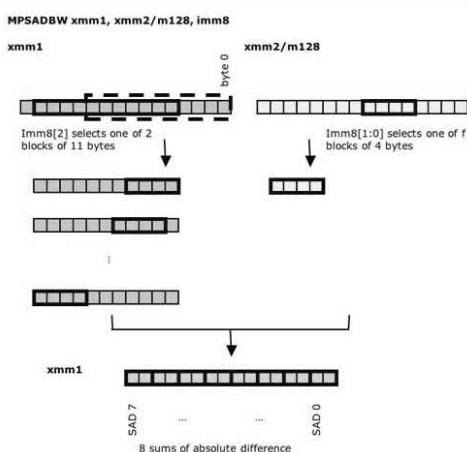
35

Primjer: MPSADBW

MPSADBW

- sums the absolute difference of 4 unsigned bytes, selected by bits [0:1] of the immediate byte (third operand), from the source (second operand) with sequential groups of 4 unsigned bytes in the destination operand.
- The first group of eight sequential groups of bytes from the destination operand (first operand) start at an offset determined by bit 2 of the immediate.
- The operation is repeated 8 times, each time using the same source input but selecting the next group of 4 bytes starting at the next higher byte in the destination.
- Each 16-bit sum is written to dest.

Primjer: MPSADBW



Primjer optimizacije (BlockMatch4x4)

Neoptimirani kod

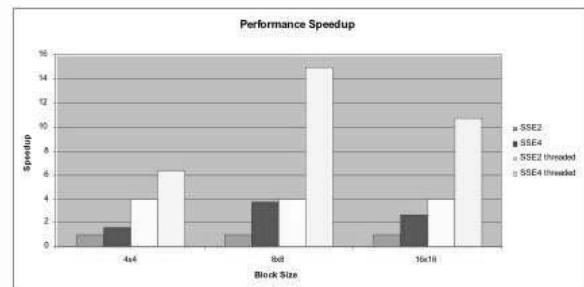
SSE2 optimiran

SSE4 optimiran

Primjer ubrzanja

Code Sample	Cycles / Block SAD	Speedup
4x4 Block		
C++	54.84	
SSE2	4.32	1.00
Intel SSE4	2.71	1.59
8x8 Block		
C++	180.55	
SSE2	25.29	1.00
Intel SSE4	6.73	3.83
16x16 Block		
C++	173.01	
SSE2	71.42	1.00
Intel SSE4	26.86	2.66

Ubrzanje



SIMD Zaključak

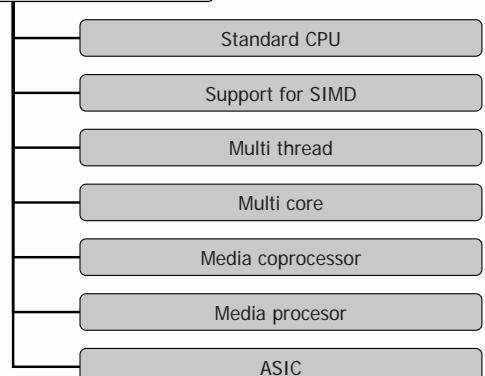
- SIMD proširenja donose značajna poboljšanja performansi
- Nažalost, korištenje SIMD zahtjeva ogromno znanje i puno vremena ali rezultira izuzetno efikasnim kodom
- Nove inačice SIMD dovode do sve boljih rezultata
- No, i SIMD ima svojih granica Što dalje?

Multimedijiske arhitekture i sustavi

41

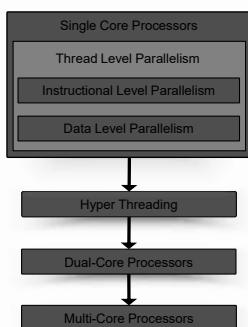
Osnovni načini CPU podrške za MM

Načini podrške za MM



Multi thread/Multi Core....

- Evolucija....



1/3/2009

43

Multi thread/Multi core

- Pregledni demo video...

Multimedijiske arhitekture i sustavi

44



Threading Games for Performance

Introducing the Destroy the Castle Demo

Destroy the Castle

■ Samo kao uvod u sljedeće predavanje:

- Destroy the Castle (serial)
- Destroy the Castle (threaded)