Elektrotehniški vestnik 74(4): 165-170, 2007
Electrotechnical Review: Ljubljana, Slovenija

Zasnova in izvedba algoritma JPEG v integriranem vezju

Simon Šrot[1], **Andrej Žemva**[2]

[1] *InSilica d.o.o., Vodovodna 99, 1000 Ljubljana, Slovenija*
[2] *Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana, Slovenija*
E-pošta: simons@insilica.si

Povzetek. V prispevku sta prikazani zasnova in izvedba algoritma JPEG v integriranem vezju ASIC z uporabo 90 nm tehnologije. Predstavljena je izvedba osnovnega sekvenčnega zgoščevanja JPEG, ki je najbolj razširjena metoda za zgoščevanje slik. Za izvedbo je bil uporabljen programski jezik Verilog HDL. Predstavljeni modul JPEG izvaja zgoščevanje sinhrono z zajemanjem slike iz senzorja. Pri tem načinu slike ni treba shranjevati v medpomnilnik, kar zagotavlja velik prihranek pri velikosti integriranega vezja in pri porabi energije, kar je ključnega pomena pri izvedbi kamere za vgradnjo v mobilne telefone, kjer so produkcijske količine velike in kjer prihranek pri velikosti vezja pomeni prihranek pri ceni.

Ključne besede: zgoščevanje digitalnih slik, JPEG, integrirana vezja ASIC, verifikacija vezij.

Design and implementation of the JPEG algorithm in integrated circuit

Extended abstract. The JPEG compression algorithm is widely used in applications demanding transfer or storage of digital images and also in cameras of mobile phones. There are rigorous constraints imposed on integrated circuits of mobile phones in terms of their price and power consumption. Software implementations of the JPEG algorithm normally require storing the image in an uncompressed form before processing. In case of a 3- Mpixel image size, this requires a memory capacity of 6 MB.

The hardware implementation of the JPEG algorithm described in this paper performs the compression simultaneously with the image captured from the image sensor. As this eliminates the need of the memory, and consequently reduces the chip area, the cost and power consumption of the integrated circuit, the proposed hardware implementation proves to be an optimal solution for the use in mobile phones.

The designed JPEG integrated circuit, realized in the 90 nm technology with ISP (Image Signal Processing) and JPEG functionality, performs the complete image processing and sensor control as well as image compression inside the JPEG module.

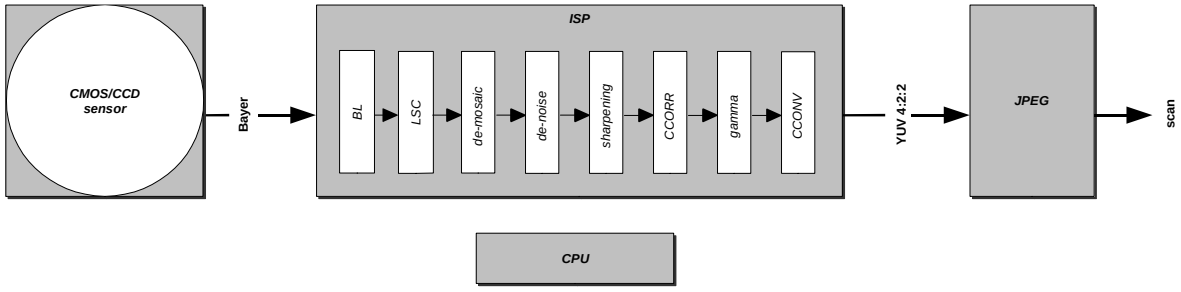
Keywords: digital image compression, JPEG, ASIC, circuit verification

Prejet 31. marec, 2007
Odobren 10. maj, 2007

1 Uvod

Zgoščevanje digitalnih slik se uporablja v večini aplikacij, ki zahtevajo shranjevanje ali prenos slik, in je nepogrešljiv del današnjih digitalnih kamer in drugih sistemov, ki uporabljajo zajem slike v digitalni obliki. Algoritem JPEG (angl.: Joint Photographic Experts Group) je danes gotovo najbolj razširjen algoritem za zgoščevanje naravnih slik [1], ki se uporablja tudi v kamerah mobilnih telefonov. Njegova zasnova omogoča zgoščevanje slike sinhrono z zajemom slike iz senzorja, kar je primerno za izvedbo algoritma v integriranem vezju majhne velikosti, ki deluje v resničnem času. Za vezja, ki se vgrajujejo v mobilne telefone, namreč veljajo zelo stroge zahteve glede cene in porabe energije. Programske izvedbe algoritma JPEG, uporabljene v mobilnih telefonih, navadno zahtevajo, da se slika najprej shrani v medpomnilnik in šele nato obdela. Pri sliki s tremi milijoni slikovnih točk bi to zahtevalo pomnilnik velikosti 6 MB.

Predstavljena strojna izvedba algoritma obdela sliko sinhrono z zajemom slike iz slikovnega senzorja in s tem odpravi zahtevo po pomnilniku. To prinese precejšen prihranek pri velikosti integriranega vezja in pri porabi energije, kar je bistvenega pomena pri izvedbi kamere za vgradnjo v mobilne telefone, kjer so produkcijske količine zelo velike in kjer prihranek pri velikosti vezja pomeni prihranek pri ceni.



Slika 1: Digitalna kamera s procesiranjem v realnem času
Figure 1: Digital camera with real-time image processing

Integrirano vezje ASIC je bilo izdelano v 90 nm tehnologiji s funkcijo ISP (Image Signal Processing) in JPEG. Vezje, prikazano na sliki 1, zajema na vhodu sliko iz slikovnega senzorja CMOS ali CCD v rastrskem zapisu. Vzorci slike se obdelajo v modulu ISP. Obdelava vzorcev vsebuje korekcijo nivoja črne barve (BL), korekcijo napake prosojnosti sistema leč LSC, interpolacijo manjkajočih barvnih komponent (de-mosaic), sistem za odstranitev šuma (de-noise), sistem za poudarjanje ostrine (sharpening), sistem za korekcijo barvnih komponent (CCORR), sistem za korekcijo intenzitete (gamma) in sistem za pretvorbo barvnih komponent (CCONV).

Modul ISP vsebuje tudi sistem za zajemanje raznih statističnih informacij o sliki, ki služijo za ugotavljanje najprimernejše osvetlitve, ugotavljanje barve svetlobe in korekcijo sivih tonov, ugotavljanje fokusne razdalje, ugotavljanje prisotnosti utripajočega svetlobnega izvora in korekcijo drugih efektov. Ti algoritmi so programsko realizirani in se izvajajo na vgrajenem namenskem procesorju (CPU). Algoritmi na podlagi statistike predhodne slike definirajo nastavitve modula ISP za obdelavo naslednje slike.

Slika se na koncu zgosti v modulu JPEG, katerega izvedba je podrobneje opisana v nadaljevanju.

2 Algoritem JPEG

Ideja zgoščevalnega algoritma JPEG s transformacijo DCT (Discrete Cosine Transform) temelji na dejstvu, da je človeško oko bolj občutljivo na globalne, grobe spremembe intenzitet barvnih komponent v sliki kot pa na lokalne in majhne spremembe in bi zatorej izguba informacije o majhnih lokalnih spremembah najmanj poslabšala kakovost slike [2].

Prostorske spremembe intenzitet v sliki se najpreprosteje ugotovijo s pretvorbo slike v prostorski frekvenčni prostor [3], za kar se uporabi transformacija DCT [4]. Algoritem JPEG temelji na pretvorbi enega bloka slike hkrati. Blok slike je delček slike velikosti 8x8 vzorcev posamezne barvne komponente. Enačba za izračun koeficientov DCT enega bloka je (1) [5]:

$$S_{uv} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{xy} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16} \quad (1),$$

pri čemer je s_{xy} vzorec barvne komponente v bloku z odštetim polovičnim razponom vrednosti (pri 8-bitnih vzorcih je polovični razpon 128), C_u in C_v pa sta:

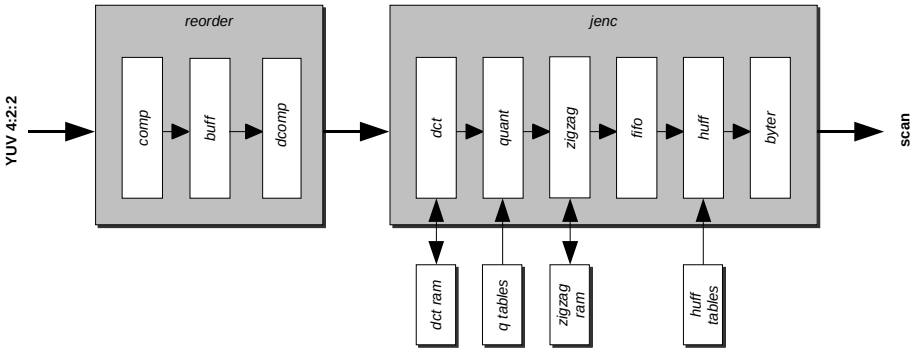
$$C_u, C_v = \frac{1}{\sqrt{2}} \text{ če sta } u, v = 0$$

$$C_u, C_v = 1 \text{ za druge vrednosti } u, v$$

S_{uv} pomeni 64 koeficientov spektra posameznega bloka. Zaradi prej omenjene lastnosti človeškega očesa koeficienti višjih frekvenc spektra slike niso tako pomembni za vizualno kakovost slike, zato se lahko število prostostnih stopenj teh koeficientov zmanjša. To je funkcija kvantizacije. Koeficienti DCT se delijo s kvantizacijskimi koeficienti iz kvantizacijske tabele. Izbira kvantizacijskih koeficientov določa kakovost rekonstruirane slike, hkrati pa tudi kompresijsko razmerje. Navadno so kvantizacijski koeficienti višjih frekvenc večji, se pa razlikujejo tudi glede na barvno komponento slike. Oko je bolj občutljivo na spremembe intenzitete slike kot pa na spremembo barve, zato se slika pred zgoščevanjem z algoritmom JPEG praviloma pretvori v barvni prostor YUV, kjer Y pomeni intenziteto, U in V pa sta barvni komponenti. Tako lahko bloke slike komponent U in V kvantiziramo z večjimi vrednostmi kvantizacijskih koeficientov kot pa bloke komponente Y, ne da bi bila vizualna kakovost rekonstruirane slike bistveno zmanjšana.

Pri naravnih slikah je veliko kvantiziranih koeficientov višjih komponent enakih nič, saj je v naravnih slikah veliko ploskev, na katerih se intenziteta posameznih barv zelo malo spreminja, kar pomeni, da bo veliko koeficientov majhne vrednosti. Koeficienti se uredijo v zaporedje cikcak, kjer si koeficienti sledijo od najnižjih frekvenčnih komponent do najvišjih. Tako zaporedje je primerno za kodiranje po postopku RLE (Run Length Encoding).

Kodiranje RLE temelji na predpostavki, da je v vhodnem zaporedju veliko zaporednih vzorcev enake



Slika 2: Zasnova algoritma JPEG
Figure 2: Design of JPEG algorithm

vrednosti. Kodiranje RLE zaporedje enakih vzorcev pretvori v unikatno sekvenco bitov (kodo), ki nosi informacijo o številu in vrednosti enakih zaporednih vzorcev [4]. Pričakujemo, da koda RLE obsega manj bitov kot prvotno zaporedje vzorcev, sicer kodiranje RLE ni smiselno.

Na koncu se dobljeno zaporedje kodira še po Huffmanovem postopku. Huffmanovo kodiranje določa kode vzorcev tako, da so vzorci, ki se pogosteje pojavljajo v zaporedju, predstavljeni s krajšimi kodami kot pa vzorci, ki se redkeje pojavljajo v zaporedju. Tako bo zaporedje Huffmanovih kod manjše kot vhodno zaporedje vzorcev.

3 Zasnova vezja

Izvedba algoritma JPEG obsega generiranje izhodnega zaporedja vzorcev Huffmanovih kod iz vhodnega rastrskega zaporedja vzorcev slike. Izvedba samega algoritma JPEG je sicer neodvisna od vhodnega formata slike, vendar pa modul za pretvorbo slike iz rastrskega zaporedja v zaporedje blokov, primernih za zgoščevanje, pričakuje vhodno zaporedje slike v formatu YUV 4:2:2. Osnovno vodilo pri načrtovanju vezja je bilo doseči čim manjšo velikost vezja in porabo energije.

Izvedba zgoščevalnega algoritma JPEG je sestavljena iz dveh glavnih delov: modula za pretvorbo rastrskega zaporedja vzorcev v zaporedje blokov, primernih za zgoščevanje, *reorder*, in modula za zgoščevanje *jenc*. Modul za zgoščevanje je nato razdeljen še na modul za diskretni kosinusni transform *dct*, modul za kvantizacijo *quant*, modul za pretvorbo zaporedja vzorcev v zaporedje cikcak *zigzag*, kodirni medpomnilnik *fifo*, modul za Huffmanovo kodiranje *huff* in modul za formatiranje izhodnega oktetnega zaporedja *byter*.

Modul *reorder* je primeren za razvrščanje vzorcev slik širine do 2048 slikovnih pik v načinu 422 in 400 oziroma slik širine 1024 v načinu 420. Modul shrani vhodno zaporedje vzorcev v medpomnilnik in nato prebere vzorce iz medpomnilnika v zaporedju, primernem za zgoščevanje. Ker je izhodno zaporedje sestavljeno iz blokov, ki sestavljajo minimalne kodirne enote, je treba v medpomnilnik shraniti toliko vrstic, kot je višina minimalne kodirne enote. V načinu 422 je višina minimalne kodirne enote 8 vrstic, v načinu 420 je višina 16 vrstic in v načinu 400 je višina 8 vrstic. V najpreprostejši izvedbi modula bi bil potreben medpomnilnik z zmogljivostjo 16 vrstic v načinu 422 in 400, oziroma 32 vrstic v načinu 420, pri čemer bi bila ena polovica medpomnilnika uporabljena za vpis novih podatkov, druga polovica pa za branje že vpisanih podatkov. Polovici bi se med seboj zamenjali na vsakih 8 oziroma 16 vrstic.

V modulu *reorder* je uporabljena najbolj optimalna izvedba, ki potrebuje medpomnilnik velikosti 8 oziroma 16 vrstic. Pri tej izvedbi se nove podatke vpisuje na lokacije, kjer se nahajajo ravnokar prebrani podatki. Taka realizacija potrebuje bolj kompleksen način vpisovanja in branja podatkov, vendar pa je zaradi tega zmogljivost medpomnilnika prepolovljena. Za širino slike 2048 slikovnih pik v načinu 420 to pomeni prihranek 32 KB spomina. Povečanja vezja za vpis in branje zaradi povečane kompleksnosti je ocenjeno na 3000 logičnih vrat, kar v izbrani tehnologiji pomeni velikost približno 18000 um^2, velikost pomnilnika z zmogljivostjo 32 KB pa je približno 400000 um^2.

Da bi še dodatno zmanjšali zahtevano velikost medpomnilnika, vsebuje modul *reorder* preprost zgoščevalni algoritem, ki zmanjša število oktetov, potrebnih za vpis v medpomnilnik. Na izhodni strani je izveden razgoščevalni algoritem. Algoritem spada med izgubne zgoščevalne algoritme, torej rekonstruirane vrednosti niso popolnoma enake vhodnim vrednostim. Algoritem temelji na predpostavki, da je človeško oko manj občutljivo na hitre in velike spremembe v sliki, kar je enako kot pri zgoščevalnem algoritmu JPEG. Ker se tovrstne informacije v sliki deloma zavržejo med samim zgoščevanjem JPEG, je torej smiselno te informacije zavreči že prej in s tem zmanjšati velikost vezja *reorder*,

ker to ne bo zmanjšalo vizualne kakovosti slike. Zgoščevalno razmerje je stalno 2:1, s čimer se zahtevana velikost medpomnilnika zmanjša še za polovico, kar pomeni prihranek 16 KB oziroma 200000 um^2. Vezje za zgoščevanje in razgoščevanje je veliko približno 4000 logičnih vrat, torej 24000 um^2.

Modul *dct* pretvarja slikovne bloke višine in širine 8 vzorcev posamezne barvne komponente v prostorski frekvenčni prostor z diskretno kosinusno transformacijo. Pri tem se enačba za izračun koeficientov DCT razdeli na horizontalni in vertikalni del. Enačba za izračun delnih koeficientov v horizontalni smeri je (2):

$$Shor_{uy} = \sum_{x=0}^{7} s_{xy} \cos\frac{(2x+1)u\pi}{16} \qquad (2)$$

$u = 0\ldots7,\ y = 0\ldots7$

Tako dobimo za vsako od osmih vrstic v bloku osem delnih koeficientov za osem različnih vrednosti *u*, skupaj torej 64 delnih koeficientov. Nato delne koeficiente uporabimo za izračun končnih koeficientov DCT s kosinusno transformacijo v vertikalni smeri po enačbi (3):

$$S_{uv} = \frac{1}{4} C_u C_v \sum_{y=0}^{7} Shor_{uy} \cos\frac{(2y+1)v\pi}{16} \qquad (3)$$

$u = 0\ldots7,\ v = 0\ldots7$

Modul *dct* je tako sestavljen iz dveh enakih modulov, enega za izračun koeficientov v horizontalni smeri in drugega za izračun koeficientov v vertikalni smeri. Modula sta zasnovana tako, da ne izračunata točnih vrednosti koeficientov DCT, pač pa le približke. Tako se vezje poenostavi, poleg tega pa so bili pri načrtovanju vezja uporabljeni postopki za skupno rabo virov. Tako se v vsakem modulu uporablja le 6 seštevalnikov in en sam množilnik. S tem je velikost integriranega vezja še dodatno zmanjšana.

Približki koeficientov DCT se nato kvantizirajo v modulu za kvantizacijo *quant*. Modul vsebuje delilnik za deljenec širine 15 bitov in delitelj širine 12 bitov. Delilnik je realiziran v štirih stopnjah, torej traja eno deljenje 4 urne cikle, vendar pa zaporedna vezava registrov omogoča eno deljenje na urni cikel. Deljenje se uporablja tudi za korekcijo natančnosti koeficientov DCT, ki so posledica poenostavitev, narejenih v modulu *dct*. Kvantizacijska tabela se pred uporabo v modulu *quant* pomnoži s korekcijsko tabelo. Tako je napaka koeficientov DCT enaka 0 za prvi koeficient, za preostale koeficiente pa narašča proti koeficientom višjih frekvenc. Ker je oko manj občutljivo na lokalne spremembe, ta napaka ne zmanjšuje vizualne kakovosti slike.

4 Verifikacija vezja

Testiranje izvedbe algoritma JPEG je razdeljeno na dva dela: na verifikacijo modula *reorder* in na verifikacijo modula *jenc*. Kot osnovno orodje smo uporabili simulator RTL za programski jezik Verilog, ki simulira delovanje modulov *reorder* in *jenc*. Poleg tega je testno okolje sestavljeno še iz programskih modelov modula *reorder* in *jenc*, ki so funkcionalno ekvivalentni izvedbi v jeziku Verilog. Ti modeli se uporabljajo za generiranje vmesnih in končnih rezultatov, ki se nato primerjajo z izhodnimi podatki simulacije RTL. Osnovni potek testiranja je določen v glavnem zapisu, napisanem v skriptnem jeziku Bourne-Again Shell.

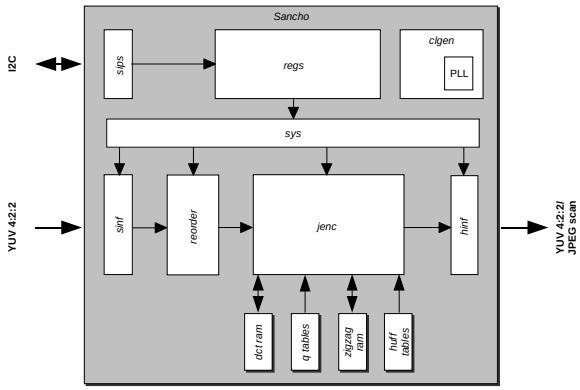
V zapisu so na preprost in pregleden način določeni vsi testni primeri in njihovi parametri, kot na primer število slik, velikost slike, razmik med okvirji slik in razmik med okvirji vrstic slike, vrsta zgoščevanja, kvantizacijske in Huffmanove tabele, poleg tega pa se za vsak testni primer uporablja tudi ena od vhodnih slik. Vhodne slike so lahko naravne ali pa so zgrajene posebej za testiranje robnih primerov posameznih modulov. Za vsak testni primer se najprej izvrši programski model. Model sprejme vhodno sliko in nabor vhodnih parametrov in na podlagi tega zgradi vmesne in končne rezultate v obliki datotek, ki so primerne za uporabo v simulacijskem okolju RTL. Enaka vhodna slika in nabor vhodnih parametrov se nato uporabita še pri izvršitvi simulacije RTL, ki simulira izvedbo modula *reorder* in *jenc* v jeziku Verilog. Pri tem se vsi vmesni in končni rezultati programskega modela uporabijo za preverjanje pravilnosti delovanja izvedbe v jeziku Verilog.

Osnovni zapis generira vhodne parametre posameznih testnih primerov. Najpreprostejši način za to je uporaba vgnezdenih zank, pri čemer vsaka zanka izbira med naborom vrednosti posameznega vhodnega parametra. Tako se zgradi množica testnih primerov, ki vsebujejo vse kombinacije vrednosti vhodnih parametrov in tako kar najbolje zagotovijo učinkovitost verifikacije. Ta se kvalitativno ugotavlja tudi z uporabo posebnega orodja, ki je del simulacijskega okolja. Orodje ugotavlja pokritost vseh vhodnih kombinacij za vsak izraz v opisu Verilog in tako prikaže učinkovitost testiranja. Pri verifikacijskem okolju modula *reorder* in *jenc* je bila pokritost testiranja več kot 98 odstotna.

Poleg testiranja s simulacijskim orodjem, ki je formalni verifikacijski del testiranja [6], je bila izvedba algoritma JPEG preizkušena tudi v resničnem sistemu digitalne kamere, ki je validacijski del testiranja. V tem načinu modula *reorder* in modul *jenc* izvedemo v tehnologiji programirljivih vezij in ju vključimo v celoten sistem digitalne kamere, kot je to prikazano na sliki 1. Tako se vse predpostavke, ki so bile narejene pri izvedbi algoritma JPEG, preverijo tudi v resničnem okolju.

5 Izvedba vezja

Blok shema integriranega vezja je prikazana na sliki 3. Integrirano vezje sprejme na vhodu sliko v rastrskem zaporedju v formatu YUV 4:2:2. Modul *sinf* lahko zajema sliko v različnem zaporedju barvnih komponent Y, U in V in nato pretvori zaporedje v obliko, primerno za zgoščevanje. Poleg tega modul *sinf* prireja velikost slike na dva načina. Prvi je decimacija slike (scale-down), pri čemer se zgradi nova slika tako, da se v originalni sliki zavrže določeno število vrstic in



Slika 3: Blok shema integriranega vezja
Figure 3: Integrated circuit's schematic

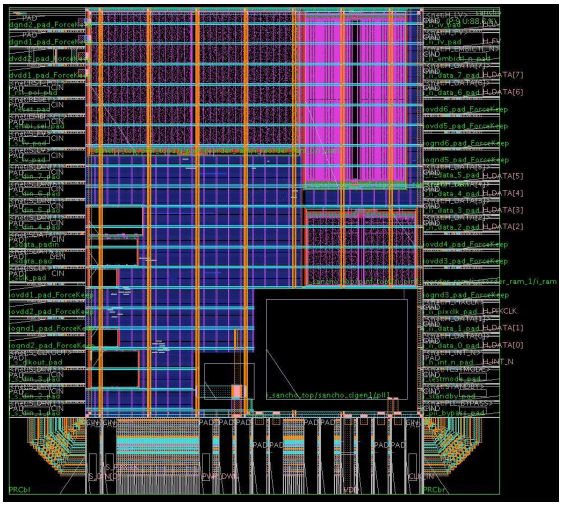
stolpcev ter uporabijo preostale vrstice in stolpci, ki so v originalni sliki enakomerno razporejeni. Drugi način je zmanjšanje vidnega polja slike (crop), pri čemer se nova slika zgradi z odstranitvijo določenega števila vrstic in stolpcev ob robovih originalne slike. Modula *reorder* in *jenc* zgoščujeta sliko oziroma sta neuporabljena in slike ne spreminjata (bypass). Na izhodu je modul *hinf*, ki sliko v zgoščenem ali nezgoščenem formatu pošilja sistemskemu procesorju. Modul *hinf* lahko spreminja zaporedje barvnih komponent, vstavlja zastavice in spreminja hitrost pošiljanja podatkov. Modul *sys* skrbi za pravilno preklapljanje med posameznimi načini delovanja vezja. Sistemski procesor nadzira delovanje integriranega vezja z vpisi v registre prek modula *sips*, ki sprejema vpise po protokolu I2C. Modul *sys* nato poskrbi za sinhronizacijo preklopov med različnimi načini delovanja posameznih modulov, tako da se vsi preklopi izvedejo med zaporednimi okvirji slik, ko so moduli v nedelujočem stanju.

Integrirano vezje je bilo izdelano v tehnologiji standardnih celic v 90 nm tehnološkem procesu CMOS s sedmimi kovinskimi nivoji. Izbrana tehnologija omogoča, da se lahko ista fizična izvedba prenaša med industrijskimi procesi različnih izdelovalcev polprevodniških vezij. Ta tehnologija sicer ni optimalna glede zmogljivosti in velikosti vezja, omogoča pa prilagodljivost pri izbiri izdelovalca produkta [7], kar je pri velikih količinah pomembna strateška prednost.

Izvedba vezja je sestavljena iz naslednjih načrtovalskih korakov [8]:

- sinteza vezja in vstavitev makroblokov izbrane knjižnice v izvorni kodi
- sinteza vezja za izbrano knjižnico standardnih celic
- vstavljanje struktur za testiranje (DFT - Design For Test)
- postavitev ogrodja vezja, makroblokov, vhodno-izhodnih struktur in struktur za napajanje (floorplaning)
- postavitev standardnih celic (placement)
- gradnja distribucije urnih signalov (CTS – Clock Tree Synthesis)
- povezovanje standardnih celic (routing)

Postavitev makroblokov integriranega vezja je prikazana na sliki 4.



Slika 4: Postavitev makroblokov
Figure 4: Placement of macro blocks

Vsako vezje se pred izdelavo maske za posamezen tehnološki korak preveri. Navadno opravimo naslednja preverjanja:

- preverjanje pravilnosti funkcije povezane sheme (FV - Formal Verification)
- preverjanje izpolnjevanja vseh časovnih meril (STA – Static Timing Analysis)
- simulacija vezja s časovno simulacijo (Timing Annotated Simulation)
- preverjanje geometrijske pravilnosti vezja (DRC – Design Rule Check)
- preverjanje enakosti geometrije vezja s povezano shemo vezja (LVS – Layout Versus Schematic)

Preverjanje v osnovi sicer ni potrebno, saj orodja, s katerimi zgradimo vezje, že vsebujejo vse algoritme, ki so navedeni v seznamu preverjanja. Ker pa je izdelava mask za integrirano vezje z današnjim napredkom tehnologije čedalje dražja in tudi ugotavljanje napak pri testiranju resničnega vezja je časovno zamuden postopek, je preverjanje nujen in obvezen korak za zagotavljanje pravilnega delovanja vezja.

Preverja se ponavadi z orodji, ki ne prihajajo od istega izdelovalca kot orodja, uporabljena pri sami gradnji vezja. Tu ne gre toliko za preverjanje napak samega programskega orodja, ampak za dejstvo, da pri prevedbi vezja v obliko, primerno za drugo programsko orodje, zmanjšamo možnost ponovitve iste napake [9].

6 Sklepne ugotovitve

Zgoščevanje digitalnih slik je nepogrešljivi del današnjih digitalnih kamer in drugih sistemov, ki uporabljajo zajem slike v digitalni obliki. Algoritem JPEG je med vsemi algoritmi najbolj razširjen algoritem za zgoščevanje naravnih slik.

Izvedba, ki smo jo opisali, je primerna za uporabo v aplikaciji digitalne kamere v mobilnem telefonu. Vse uporabljene predpostavke, ki pripomorejo k zmanjšanju velikosti vezja, so bile dokazane kot upravičene z validacijo modula *jenc* v resničnem sistemu digitalne kamere. Kakovost slike je zaradi teh optimizacij le neopazno slabša v primerjavi z najnatančnejšimi izvedbami algoritma JPEG, kar smo preverili z vsemi standardnimi slikovnimi scenami, ki se uporabljajo za digitalno fotografijo. Uporabljene rešitve pri izvedbi medpomnilnika za pretvorbo rastrskega formata slike v format blokov in rešitve pri sami izvedbi algoritma JPEG pomenijo velik prihranek pri velikosti vezja, kar je najpomembnejše merilo tovrstnih aplikacij, tako s stališča cene kot s stališča porabe energije vezja. Majhna površina vezja omogoča vgradnjo neposredno poleg slikovnega senzorja v tako imenovani kamera-modul, ki je sestavljen iz slikovnega senzorja in sistema leč.

Poleg prihranka pri velikosti vezja omogoča strojna izvedba tudi hitrejšo obdelavo slike v primerjavi s programsko. Hitrost obdelave slike je pri strojni izvedbe enaka zajemanju slike, kar omogoča zgoščevanje video slik po metodi M-JPEG (Motion JPEG) [10].

7 Literatura

[1] H. R. Wu, K. R. Rao, Digital video image quality and preceptual coding, CRC Press, Boca Raton, 2000.

[2] John Miano, Compressed image file format, Addison-Wesley, New Jersey, 1999.

[3] Mohammed Ghanbari, Standard codecs: Image compression to advanced video coding, The institution of electrical engineers, London, 2003.

[4] http://www.vectorsite.net, JPEG image compre-ssion (dostopano 2.11.2006).

[5] International Telecommunication Union, Digital compression and coding of continuous-tone still images – Requirements and guidelines T.81, 1993.

[6] Andreas Meyer, Principles of functional verification, Elsevier Science, Burlington, 2004.

[7] Michael Keating, Pierre Bricaud, Reuse methodology manual for System-On-Chip designs, Kluwer Academic Publishers, Massachusetts, 2002.

[8] Michael John Sebastian Smith, Aplication-specific integrated Circuits, Addison-Wesley, New Jersey, 1997.

[9] David Chinnery, Kurt Kreutzer, Closing the gap between ASIC & custom: Tools and techniques for high-performance ASIC design, Kluwer Academic Publishers, Massachusetts, 2003.

[10] Peter Symes, Digital video compression, McGraw-Hill, New York, 2004.

Simon Šrot je diplomiral leta 2000 na Fakulteti za elektrotehniko Univerze v Ljubljani, kjer je leta 2006 tudi magistriral. Trenutno je zaposlen kot razvijalec integriranih vezij ASIC v podjetju Insilica, d.o.o., v Ljubljani. Njegovo delo vključuje celotno področje zasnove kompleksnih digitalnih integriranih vezij v submikronski tehnologiji.

Andrej Žemva je diplomiral, magistriral in doktoriral na Fakulteti za elektrotehniko Univerze v Ljubljani v letih 1989, 1993 in 1996. Njegova raziskovalna in razvojna dejavnost obsega načrtovanje digitalnih elektronskih vezij in sistemov, vgrajene sisteme ter sočasno načrtovanje strojne in programske opreme.