ELEKTROTEHNIŠKI VESTNIK 79(1-2): 55-60, 2012
EXISTING SEPARATE ENGLISH EDITION

Razvoj namenskih mikroprocesorjev za vezja FPGA

Andrej Trost, Andrej Žemva

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana, Slovenija
E-pošta: andrej.trost@fe.uni-lj.si

Povzetek. Programirljiva vezja v tehnologiji FPGA omogočajo razvoj in prototipno izdelavo kompleksnih digitalnih sistemov. Digitalni sistemi izvajajo naloge z uporabo centralnoprocesnih enot in logičnih vezij, ki so namenjena za učinkovito izvedbo algoritmov.

V prispevku je predstavljen razvoj namenskega mikroprocesorskega sistema v vezju FPGA. Osnovna zgradba temelji na centralnoprocesni enoti z akumulatorjem, ki omogoča preprosto nadgradnjo in dodajanje ukazov. Procesor in zbirniška koda programa, ki ga izvaja, sta v celoti opisana v jeziku VHDL, tako da za uporabo ne potrebujemo dodatnih prevajalnikov. Centralno procesno enoto smo optimizirali za tehnologijo vezij FPGA in predstavili rezultate sinteze 4- do 32-bitnih enot. Opisana sta nadgradnja s perifernimi enotami in vodilom Wishbone ter primer uporabe v grafičnem krmilniku.

Ključne besede: digitalni sistemi, programirljiva vezja, procesor, grafični krmilnik

Design of Custom Processors for the FPGA Devices

Programmable devices in the Field Programmable Gate Array (FPGA) technology enable design and prototyping implementation of complex digital systems executing their tasks on central processing units (CPU) and application-specific logic components. The paper presents development of a custom processor system in the FPGA device. The system is based on a CPU with an accumulator easily extendable with additional instructions. The processor core and program memory are described in a VHDL language and no additional compiler is required. The CPU is optimized for the FPGA devices and synthesis results of 4- to 32-bit cores are presented. An upgrade of the CPU by using peripheral units, Wishbone compatible bus and case with a graphical controller is presented.

1 UVOD

Tehnologija programirljivih vezij FPGA omogoča razvoj in prototipno izdelavo kompleksnih digitalnih sistemov. Digitalni sistemi izvajajo naloge z uporabo centralnoprocesnih enot (CPE) in logičnih vezij, ki so namenjena za učinkovito izvedbo algoritmov [1].

Izdelovalci programirljivih vezij ponujajo razvojna orodja za sintezo logičnih vezij iz visokonivojskega opisa, npr. v jeziku VHDL [2], ki so v osnovni obliki prosto dostopna in primerna za izobraževalne namene [3]. Za uporabo mikroprocesorjev v vezjih FPGA potrebujemo procesorsko jedro in razvojna orodja s prevajalniki in generatorji strojne in programske kode. V nekaterih družinah vezij FPGA so vgrajeni standardni procesorji (npr. ARM, PowerPC), ki so del integriranega vezja. V vseh drugih družinah lahko s programirljivo logiko naredimo lasten procesor. Izdelovalec Xilinx ponuja zastonj 8-bitni procesor Picoblaze [4], za zmogljivejši 32-bitni Microblaze pa potrebujemo komercialna orodja.

Mikroprocesorski sistem vsebuje večino digitalnih gradnikov, ki jih najdemo v tipičnem sistemu: registre, pomnilnik, kombinacijske dekodirnike in računske enote ter sekvenčni stroj oz. avtomat. Zato je razvoj lastnega mikroprocesorskega sistema dobra naloga za študente, ki se učijo snovanja kompleksnih digitalnih sistemov [5]. Namenski procesorji v vezjih FPGA so predmet številnih raziskav [6-8].

Optimalna izvedba CPE je odvisna od tehnologije, zato v FPGA vgrajeni procesorji največkrat niso združljivi z naborom ukazov katerega izmed komercialnih vgrajenih procesorjev in je treba razviti tudi lastne prevajalnike.

V prispevku bomo predstavili razvoj generičnega mikroprocesorskega jedra in optimizacijo za izvedbo v tehnologiji FPGA. Procesor in zbirniška koda programa, ki ga izvaja, sta v celoti opisana v jeziku VHDL, tako da za uporabo ne potrebujemo dodatnih prevajalnikov. V zadnjem poglavju bomo prikazali primer uporabe procesorja v grafičnem krmilniku.

2 RAZVOJ MIKROPROCESORSKEGA JEDRA

Zgradbo mikroprocesorja določa nabor ukazov (angl. Instruction Set Architecture, ISA), ki se izvajajo v mikroarhitekturi procesorja. Vsak mikroprocesorski

Prejet 24. april, 2012
Odobren 9. maj, 2012

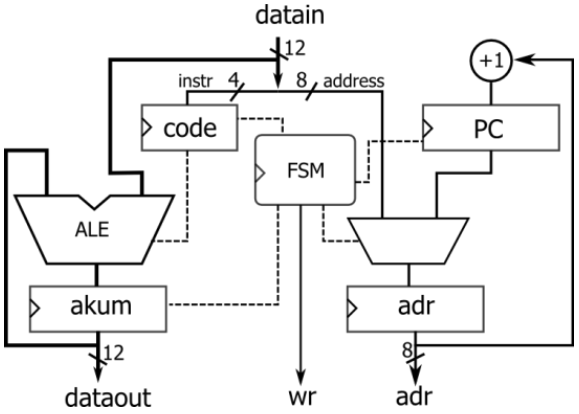
sistem vsebuje pomnilnik z ukazi in podatki ter enoto za izvedbo različnih operacij. Rezultat izračunov se shrani v registrih ali v pomnilniku. Vgrajeni procesorji z manjšim naborom ukazov – RISC (angl. Reduced Instruction Set Computer) imajo večje število podatkovnih registrov. Podatkovni registri omogočajo zelo hitro izvedbo operacij, saj je dostop do pomnilnika po navadi ozko grlo. Če namesto registrov uporabimo kar pomnilniške bloke, ki so na voljo v vezju FPGA, se zgradba procesorja nekoliko poenostavi.

Izbrali smo preprosto mikroarhitekturo procesorja z enim samim podatkovnim registrom, akumulatorjem, v katerem se shranijo rezultati operacij [9]. Ukazi z dvema operandoma, kot je npr. vsota dveh vrednosti, imajo vedno en operand v akumulatorju, drugega pa preberejo iz pomnilnika.

Strojni ukazi procesorja so binarne kode, ki so sestavljene iz dveh delov: kode ukaza in pomnilniškega naslova. Izbrali smo 4-bitne ukazne kode, s katerimi opišemo največ 16 različnih ukazov. Ukazne kode definiramo kot konstante v paketu jezika VHDL:

```
subtype  koda is unsigned(3 downto 0);
constant lda:   koda := "0001";        --  a = [M]
constant sta:   koda := "0010";        --  [M] = a
constant add:   koda := "0100";        --  a = a + [M]
constant sub:   koda := "0101";        --  a = a - [M]
constant anda:  koda := "0110";        --  a = a and [M]
constant ora:   koda := "0111";        --  a = a or [M]
constant jmp:   koda := "1000";        --  jump
constant jze:   koda := "1001";
```

Pomnilniški naslovi procesorja so 8-bitni, kar je več kot dovolj za demonstracijo delovanja v učne namene. Ves strojni ukaz sestavlja 12 bitov, ki jih procesor zajame iz pomnilnika v enem ciklu. Pomnilniki v tehnologiji vezij FPGA imajo poljubno širino podatkovnega vodila, zato so lahko tudi operandi 12-bitne vrednosti. Slika 1 prikazuje blokovno shemo zgradbe 12-bitne procesne enote.



Slika 1: Zgradba jedra 12-bitnega mikroprocesorja

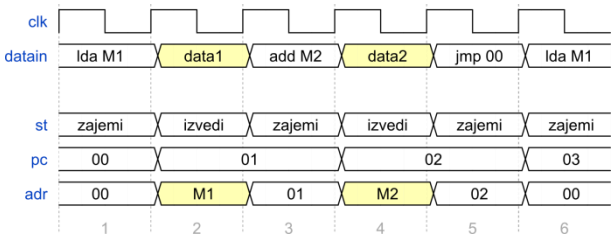
V podatkovnem delu procesne enote sta dva registra: 4-bitni register za ukazno kodo in 12- bitni akumulator. Ukazi in operandi prihajajo iz pomnilnika prek vodila dat_i, prek vodila dat_o pa rezultate shranimo oz. prenesemo nazaj v pomnilnik. Oba registra in operacije, ki shranijo rezultat v pomnilnik, opišemo v obliki sinhronega vezja z izbirnim stavkom:

```
if st=zajemi then                    -- shrani kodo ukaza
   code <= instr;
elsif st=izvedi then          -- izvedi ukaz z akumulatorjem
   case code is
     when lda                          => akum <= dat_i;
     when add                    => akum <= akum + dat_i;
     when sub                    => akum <= akum - dat_i;
     when anda                => akum <= akum and dat_i;
     when ora                  => akum <= akum or dat_i;
     when others => null;
   end case;
end if;
```

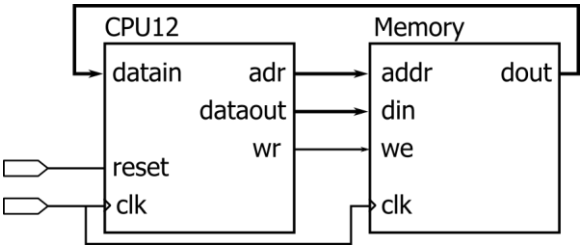
Zapis v registre je odvisen od stanja procesorja: zajemi ali izvedi, ki ga določa sekvenčni stroj stanj (angl. Finite State Machine, FSM). Krmilni del procesorja vsebuje še dva registra: naslovni (adr) in programski števec (PC), v katerem je pomnilniški naslov naslednjega ukaza. Tudi opis krmilnega dela je lahko zelo kompakten:

```
-- Krmilni signali, PC in naslovni register
wr_o <= '0';
if st = zajemi then
   st <= izvedi;
   pc <= adr + 1;   -- naslov naslednjega ukaza
   adr <= address;  -- naslov za podatek
   if instr=sta then
     wr_o <= '1';
   elsif instr=jmp or (instr=jze and akum=0) then
     st <= zajemi;
   end if;
else
   st <= zajemi;
   adr <= pc;
end if;
```

Ukazi se večinoma izvedejo v dveh ciklih: v stanju zajemi se iz pomnilnika prenese ukazna koda in naslov, v stanju izvedi pa se prenese operand in shrani rezultat. Izjema so le skočni ukazi, ki se izvedejo v enem ciklu. Slika 2 prikazuje časovni potek izvajanja treh ukazov. Procesor iz naslova 0 zajame ukaz "lda M1", v naslednjem urnem ciklu pa še iz naslova M1 operand, ki se bo shranil v akumulator. Podobno se izvede naslednji ukaz "add M2", le da se bo operand prištel k akumulatorju. Skočni ukaz "jmp 00" povzroči, da se naslednji ukaz zajame iz naslova 0.



Slika 2: Časovni potek izvajanja ukazov: lda, add in jmp



Slika 3: 12-bitna procesna enota s programskim pomnilnikom

Slika 3 prikazuje povezavo procesne enote in programskega pomnilnika. Pomnilnik v vezju FPGA ima ločeno vhodno in izhodno podatkovno vodilo. Uporabili smo pomnilnik z asinhronim branjem in sinhronim zapisovanjem podatkov. Podatke zapisuje v pomnilnik le ukaz sta, ki nastavi krmilni signal wr_o.

Model pomnilnika naredimo v jeziku VHDL z zbirko podatkov, ob inicializaciji zbirke pa vpišemo strojno kodo:

```
signal m : memory := (
  lda & x"03",
  add & x"03",
  jmp & x"00",
  x"005"
);
```

Ker smo za ukaze definirali simbolične konstante, jih uporabimo tudi pri opisu programa, tako da je koda zelo podobna zbirniku. Za uporabo procesorja tako ne potrebujemo dodatnega križnega prevajalnika in zbirnika v strojno kodo, kar precej poenostavi uporabo procesorja v učne namene.

3 NADGRADNJA IN OPTIMIZACIJA

Predstavljena mikroarhitektura procesorja omogoča preprosto nadgradnjo z novimi ukazi in enotami. V podatkovno pot lahko vključimo nove ukaze z enim (npr. pomik bitov akumulatorja) ali dvema operandoma. Dodamo lahko vhodna in izhodna vrata, prek katerih povežemo procesor v digitalni sistem.

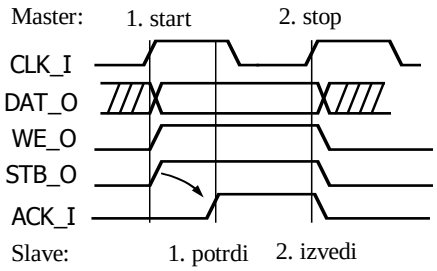
3.1 Vhodna in izhodna vrata

Vhodna in izhodna vrata omogočajo prenos podatkov med procesorjem in zunanjim svetom. Naredimo jih bodisi z deljenjem obstoječega pomnilniškega prostora bodisi z uporabo novih ukazov za dostop do posebnega, vhodno/izhodnega prostora. Prva rešitev ne zahteva posega v zgradbo procesne enote, druga pa z majhnim posegom doda še enkrat več naslovnega prostora in poenostavi povezovanje v sistem.

3.1.1 Vodilo Wishbone

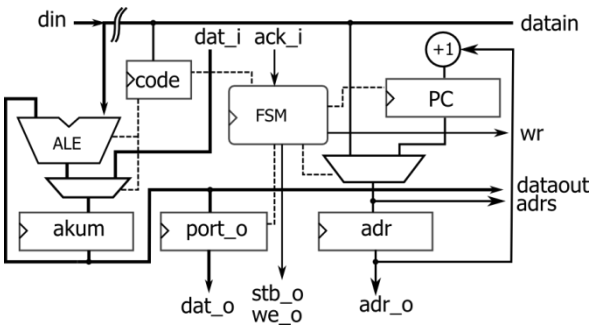
Vodilo Wishbone je sinhrono vzporedno vodilo s protokolom vrste Master/Slave (M/S) [10]. Standard predvideva različne povezovalne arhitekture: točka s točko, podatkovni tok, deljeno vodilo in stikalno. Vse povezovalne linije so enosmerne.

Osnovni protokol M/S je zelo preprost, tako da pri načrtovanju gradnikov za vodilo nimamo veliko redundance. Slika 4 prikazuje postavitev glavnih krmilnih signalov pri pisalnem ciklu.



Slika 4: Pisalni cikel na vodilu Wishbone

Slika 5 prikazuje novo zgradbo procesorja z vhodnimi in izhodnimi vrati. Podatek se iz vhodnih vrat prenese v akumulator z novim ukazom inp. Izhodna vrata so v obliki registra, v katerega se prepiše vrednost akumulatorja ob ukazu outp. V obeh primerih se na naslovnem vodilu prikaže naslov, ki je argument ukaza, in nastavi kontrolni signal za signalizacijo branja oziroma pisanja.



Slika 5: Nadgradnja procesorja z vodilom Wishbone

3.1.2 Sinhroni pomnilnik

Programirljiva vezja FPGA vsebujejo različne gradnike za izvedbo pomnilnika. Pomnilnik z asinhronim branjem in sinhronim vpisom podatkov naredimo s porazdeljenimi pomnilnimi bloki (angl. distributed RAM), ki so na voljo v matriki logičnih blokov. Poleg pomnilnika v matriki imajo FPGA na voljo dodatne bloke sinhronega pomnilnika z oznako BRAM. Ti bloki imajo pisanje in branje podatkov sinhronizirano z uro, kar pomeni, da dobimo podatek en cikel po postavitvi naslova.

Sinhroni bloki že vsebujejo naslovni register za hitrejši prenos podatkov. Sprememba v zgradbi procesorja, ki omogoča uporabo sinhronega pomnilnika, je relativno preprosta: treba je le odstraniti naslovni register in povezati signale iz izbiralnika na naslovne vhode pomnilnika.

3.2 Razširitev nabora ukazov ALE

V jeziku VHDL preprosto dodamo nove ukaze v podatkovno pot z razširitvijo izbirnega stavka. Dodamo lahko še zastavico za prenos (angl. Carry), ki se postavi, kadar pri seštevanju ali odštevanju pride do prenosa na najvišjem bitu in zato rezultat v akumulatorju ni pravilen. K ukazom aritmetično logične enote bi v tem primeru dodale vsaj še ukaz za seštevanje s prenosom, ki izvede operacijo:

$$akum = akum + [M] + Carry$$

in odštevanje z izposojo:

$$akum = akum - [M] - Carry$$

Vse te spremembe pa zahtevajo dodatne seštevalnike in izbiralnike, ki precej povečajo zasedenost vezja FPGA. Zato velja razmisliti o posebni izvedbi ALE, ki bi bila optimizirana izvedba v arhitekturi vezja FPGA.

3.3 Optimizacija ALE

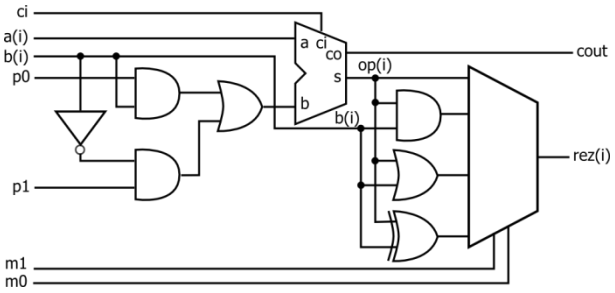
Računsko jedro procesorja je aritmetično-logična enota (ALE), ki ob nastavljenih parametrih izvede izbrano operacijo in shrani rezultat v akumulator. Enota izvaja operacije nad večbitnimi vrednostmi in ima regularno strukturo. Osnovna operacija seštevanja je narejena s povezavo enobitnih polnih seštevalnikov. Če na vhodu vsakega dodamo nekaj logike, dobimo že kar uporaben nabor osnovnih računskih operacij.

V zasnovi vezja razdelimo ALE na celice za obdelavo posameznih bitov, ki jih bomo imenovali aritmetične celice. Če bi obravnavali operacije skupaj z izhodnim registrom (akumulatorjem), bi se ukvarjali z razvojem registrskih celic [11]. Delitev na celice nam olajša načrtovanje in optimizacijo kombinacijske logike. Aritmetična celica z dvobitnim parametrom *p* izvaja naslednje funkcije:

$$op = \begin{cases} a + ci, & p = 00 \\ a + b + ci, & p = 01 \\ a + \bar{b} + ci, & p = 10 \\ a - 1 + ci, & p = 11 \end{cases}$$

Funkcija celice je odvisna od parametra p in vhodnega prenosa ci. Pri p=00 je izhod enak prvemu vhodu, ali pa vhodu, povečanem za 1, ki pomeni operacijo inkrement. Pri p=01 dobimo vsoto s prenosom, kadar je p=10, dobimo razliko z izposojo. Kombinacija p=11 pa tedaj, ko je vhodni prenos enak 0, zmanjša vhod za 1, kar pomeni operacijo dekrement.

Logične operacije dodamo z izbiralnikom in dvema dodatnima bitoma parametra. V izbiralnik in logične operatorje »in«, »ali« ter »ex-ali« povežemo izhod aritmetične celice in drugi operand. Celotno shemo osnovne celice ALE prikazuje slika 6.



Slika 6: Zgradba osnovne celice aritmetično logične enote

Sodobna programirljiva vezja vsebujejo logične celice s 4-vhodnimi tabelami (LUT) in dodatno prenosno logiko. Vezje smo zasnovali tako, da za vsak bit potrebujemo natanko 2 logični celici: prvo za izvedbo aritmetične celice in drugo za izbiralnik z logičnimi operatorji. Programirljivo vezje je tako optimalno izkoriščeno.

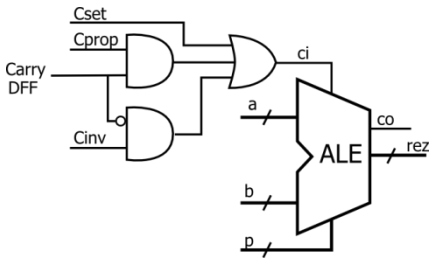
Tabela 1: Dekodirna tabela za ukaze aritmetično-logične enote

sel(3:0)	p(1:0)	m(1:0)	Cprop	Cinv	Cset	ukaz
0000	00	00	0	0	0	lda
0001	01	00	0	0	0	add
0010	10	00	0	0	1	sub
0011	11	00	0	0	0	dec
0100	00	00	0	0	1	inc
0101	01	00	1	0	0	adc
0110	10	00	0	1	0	sbc
1x01	00	01	0	0	0	anda
1x10	00	10	0	0	0	ora
1x11	00	11	0	0	0	xora

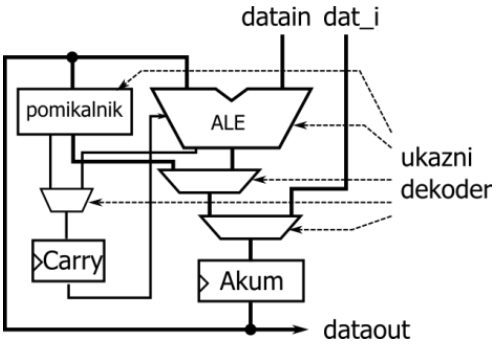
Izhodni prenos ALE shranimo v flip-flop, ki pomeni prenosno zastavico. Vrednost vhodnega prenosa naj bo odvisna od prenosne zastavice in dodatnih parametrov:

- *Cprop* določa propagacijo prenosa na *ci*, ki jo potrebuje ukaz seštevanja s prenosom (adc),
- *Cinv* določa invertiranje in propagacijo prenosa, ki jo potrebujemo za odštevanje z izposojo (sbc),
- *Cset* pa postavi *ci* na 1, kar potrebujemo za operaciji povečevanja (inc) in odštevanja (sub).

Vezje za izračun vhodnega prenosa, ki je na sliki 7 vezano med flip-flop in aritmetično enoto (AE), zasede eno samo logično celico. Dekodirnik strojnih ukazov mora nastaviti vrednosti vseh parametrov v odvisnosti od trenutnega ukaza.



Slika 7: Izračun vhodnega prenosa v ALE



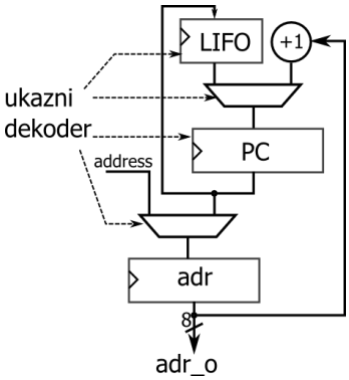
Slika 8: Podatkovna pot z aritmetično-logično enoto (ALE), pomikalnikom, akumulatorjem in zastavico prenosa

3.4 Podprogrami in prekinitve

V osnovnem naboru ukazov manjkajo skočni ukazi za izvedbo podprogramov in prekinitev. Ob skoku na podprogram ali prekinitveno rutino je treba na sklad shraniti vrednost programskega števca, ki se ob vrnitvi postavi v prejšnje stanje.

Sklad je rezerviran prostor v delu glavnega pomnilnika. Procesorji, pri katerih ne pričakujemo veliko gnezdenja skokov, imajo sklad v namenskem pomnilniku.

Namenski sklad dodamo v krmilni del procesorja k programskemu števcu, kot prikazuje slika 5. Velikost sklada je odvisna od pričakovane globine gnezdenja skokov, za manjše procesorje pa je tipično od 4 do 32 lokacij.



Slika 9: Krmilni del procesorja s strojnim skladom

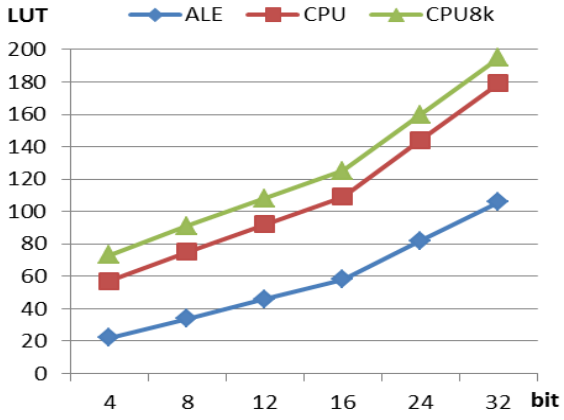
3.5 Rezultati sinteze generične CPU

Osnovna izvedba 8-bitne CPE zasede v vezju FPGA iz družine Xilinx Spartan-3 skupaj 30 flip-flopov in 62 tabel (LUT), 12-bitna CPE pa 34 flip-flopov in 79 LUT.

Visokonivojski opis aritmetično-logične enote z desetimi aritmetičnimi in logičnimi ukazi zasede v FPGA 176 LUT. Optimizirana ALE v strukturnem opisu zasede 31 LUT, kar je le 18 % v primerjavi z visokonivojskim opisom.

Optimizirana 12-bitna ALE z aritmetičnimi, logičnimi ukazi in pomikanjem zasede 46 LUT. Celoten mikroprocesor z 21 ukazi zasede 92 LUT, 57 flip-flopov in deluje pri frekvenci 130MHz.

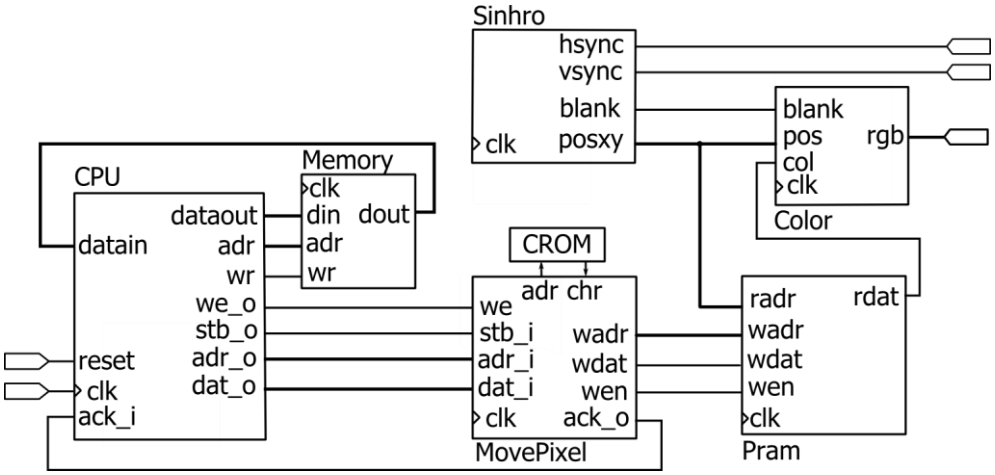
Slika 10 prikazuje rezultate zasedenosti (v enotah LUT) za generično izvedbo procesorja s 4- do 32-bitov podatkovne širine in velikosti programskega pomnilnika 128 besed (CPU) in 8k besed (CPU8k).



Slika 10: Rezultat sinteze procesorskega jedra in ALE

4 Uporaba CPE v grafičnem krmilniku

Programirljiva vezja se uporabljajo predvsem pri obdelavi hitrih signalov, npr. videosignala, kjer se zahteva obdelava podatkov v realnem času [12]. Uporabo CPE bomo predstavili na primeru razvoja krmilnika za računalniško grafiko v načinu VGA pri osnovni ločljivosti 640 x 480 točk [13]. To je grafični način, ki ga še vedno podpirajo vse kartice in monitorji, uporablja pa se tudi v prenosnih napravah.



Slika 11: Primer digitalnega sistema z mikroprocesorjem in grafičnim krmilnikom v programirljivem vezju

Grafični krmilnik je sestavljen iz sinhronizacijske komponente (Sinhro), ki usklajuje izrisovanje slike, komponente za določanje izhodnih barv (Color), pomnilnika za slikovne točke (Pram) in komponente za premikanje točk in prikaz znakov (MovePixel), kot prikazuje slika 11. Komponente krmilnika so prek vodila Wishbone povezane z 12-bitno centralno procesno enoto.

Za osveževanje slike na zaslonu skrbita sinhronizacijska in barvna komponenta, ki bereta podatke z enih vrat pomnilnika Pram. Na druga vrata so vezani signali komponente MovePixel, v kateri je sekvenčno vezje za premikanje slikovnih točk. Osnovni nalogi tega vezja sta vpisovanje točke na določeno pozicijo v pomnilniku in prenos znakov v obliki sličic velikosti 8x8 točk iz znakovnega pomnilnika (CROM) v točkovni pomnilnik (Pram).

Procesor izvaja algoritme za risanje vektorske slike iz točk v slikovnem pomnilniku. Napisali smo strojno kodo za Bresenhamov algoritem risanja ravnih črt in krožnic [14]. Algoritem obdeluje 12-bitne celoštevilske vrednosti, zato smo za optimalno izvedbo algoritma izbrali 12-bitno CPE.

5 SKLEP

Predstavili smo razvoj generičnega mikroprocesorskega jedra, ki je optimizirano za izvedbo v tehnologiji FPGA. Vezje je prilagojeno za izobraževalne namene, zato ima osnovna izvedba CPE preprosto zgradbo, obenem pa ponuja dovolj možnosti za nadgradnjo.

LITERATURA

[1] W. Wolf, FPGA-Based System Design, Prentice Hall, New Jersey, 2004.

[2] D. L. Perry, VHDL, 3rd ed. McGraw-Hill, 1998.

[3] D. Hanna and R. E. Haskell, “Learning Digital Systems Design in VHDL by Example in a Junior Course,” Proceedings of the ASEE North Central Section Conference, Charleston, West Virginia, marec 2007.

[4] PicoBlaze 8-bit Embedded Microcontroller User Guide, Xilinx inc, 2011, www.xilinx.com.

[5] V. Angelov, Volker L., 2009 “The Educational Processor Sweet-16”, International Conference on Field Programmable Logic and Applications, 2009, Praga, pp. 555–559.

[6] G. Hempel, C. Hochberger, "A resource optimized Processor Core for FPGA based SoCs," *Digital System Design Architectures, Methods and Tools*, pp.51–58, August 2007.

[7] N. Maheshwari, P. K. Jain, D.S. Ajnar: A 16-Bit Fully Functional Single Cycle Processor, International Journal of Engineering Science and Technology, 2011, vol. 3, no. 8, pp. 6219–6226.

[8] M. Schoeberl: Leros: A tiny microcontroller for FPGAs, In Proceedings of the 21st International Conference on Field Programmable Logic and Applications (FPL 2011), September 2011

[9] T Böscke, MCPU - A Minimal 8Bit CPU in a 32 Macro cell CPLD, www.opencores.org.

[10] WISHBONE, Revision B.4 Specification, 2010, http://opencores.org/opencores,wishbone.

[11] F. Vahid, Digital Design. John Wiley & Sons, Inc., 2007.

[12] S. Lapanja, A. Trost, Testno okolje za razvoj vgrajenih naprav obdelave videosignala, Elektrotehniški vestnik, vol 77, no. 2-3, pp. 137–142, 2010.

[13] A. Trost, Načrtovanje digitalnih vezij v jeziku VHDL, založba FE/FRI, 2011.

[14] A. Zingl, The Beauty of Bresenham's Algorithm, http://free.pages.at/easyfilter/bresenham.html, 2012.

Andrej Trost je doktoriral leta 2000 na Fakulteti za elektrotehniko Univerze v Ljubljani in bil habilitiran v naziv docent. Raziskovalno se ukvarja z razvojem vezij v tehnologiji FPGA in načrtovanjem digitalnih sistemov, ki jih razvija tudi za industrijske aplikacije. Visokonivojsko načrtovanje vezij poučuje pri predmetih na dodiplomskem in podiplomskem študiju.

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.