Kako nastaje računarski hardver?

Stefan Nikolić

Departman za matematiku i informatiku
Prirodno-matematički fakultet, Novi Sad

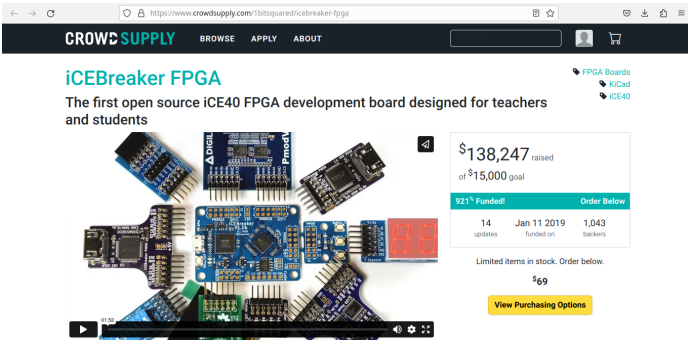
16.10.2024.

Razvojem hardvera se ne bave samo velika, poznata preduzeća

Razvojem hardvera se ne bave samo velika, poznata preduzeća

prof. dr Žan-Daniel Niku objašnjava deci principe projektovanja hardvera

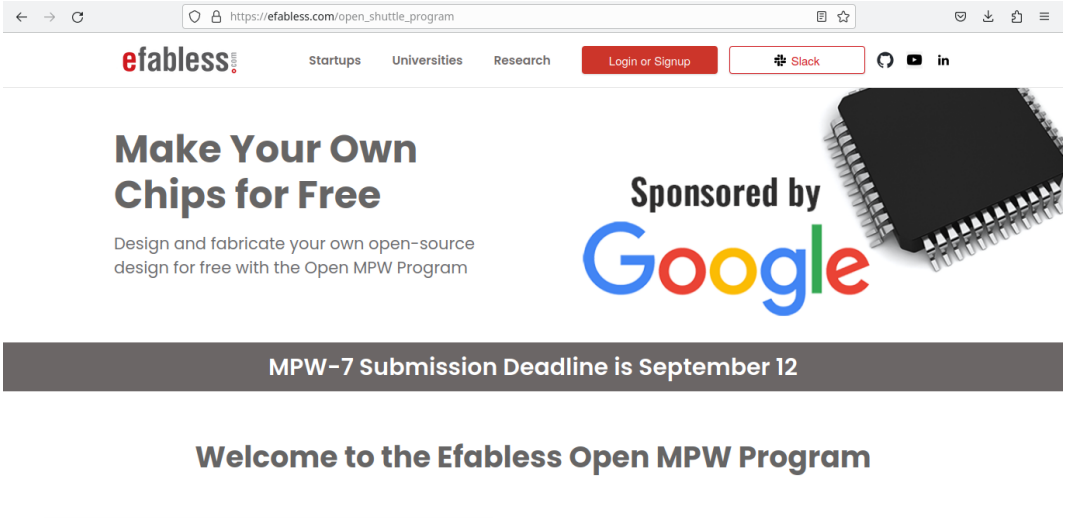
FPGA čipovi omogućuju svima nama da „proizvedemo” svoj hardver



Yosys+nextpnr: an Open Source Framework from Verilog to Bitstream for Commercial FPGAs

David Shah*†, Eddie Hung†*, Clifford Wolf*, Serge Bazanski*, Dan Gisselquist* and Miodrag Milanović*
*Symbiotic EDA; Vienna, Austria; {david,clifford}@symbioticeda.com
†Dept. of Electrical and Electronic Engineering; Imperial College London, UK
‡Dept. of Electrical and Computer Engineering; University of British Columbia, Canada; eddieh@ece.ubc.ca

Odnedavno je i proizvodnja čipova postala šire dostupna



Samostalni razvoj hardvera opšte namene nema mnogo smisla
(osim radi učenja, zabave, ili zbog bezbednosti)

Samostalni razvoj hardvera specijalizovanog za neki konkretan problem ili klasu problema često (i sve češće) ima smisla

Kada nam je potreban specijalizovan hardver?

Ukoliko nam je potrebno kraće vreme obrade podataka i/ili manja potrošnja električne energije

Primer: Poravnanje sekvenci u genomici

2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)

Darwin-WGA: A Co-processor Provides Increased Sensitivity in Whole Genome Alignments with High Speedup

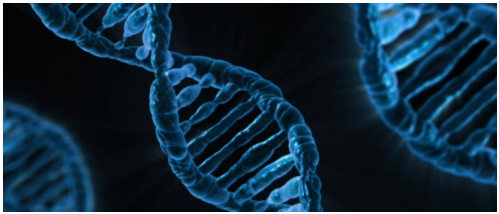
Yatish Turakhia* *Stanford University yatisht@stanford.edu*

Sneha D. Goenka* *Stanford University gsneha@stanford.edu*

Gill Bejerano *Stanford University bejerano@stanford.edu*

William J. Dally *Stanford University NVIDIA Research dally@stanford.edu*

Figure 6: Hardware-software partitioning of the different stages of Darwin-WGA.



"I do 1500 puta veće performanse po vatu utrošene snage od postojećeg softverskog rešenja"

Primer: Obrada podataka u eksperimentima čestične fizike

FERMILAB-CONF-20-622-CMS-SCD

Accelerated Charged Particle Tracking with Graph Neural Networks on FPGAs

Aneesh Heintz*
Cornell University
Ithaca, NY 14850, USA

Vesal Razavimaleki*, **Javier Duarte**
University of California San Diego
La Jolla, CA 92093, USA

Gage DeZoort, Isobel Ojalvo, Savannah Thais
Princeton University
Princeton, NJ 08544, USA

Markus Atkinson, Mark Neubauer
University of Illinois at Urbana-Champaign
Champaign, IL 61820, USA

Lindsey Gray, Sergo Jindariani, Nhan Tran
Fermi National Accelerator Laboratory
Batavia, IL 60310, USA

Philip Harris, Dylan Rankin
Massachusetts Institute of Technology
Cambridge, MA, 02139, USA

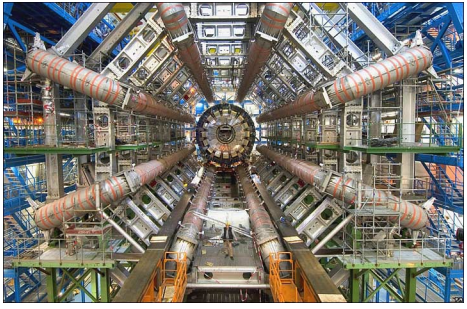
Thea Aarrestad, Vladimir Loncar, Maurizio Pierini, Sioni Summers
European Organization for Nuclear Research (CERN)
CH-1211 Geneva 23, Switzerland

Jennifer Ngadiuba
California Institute of Technology
Pasadena, CA 92115, USA

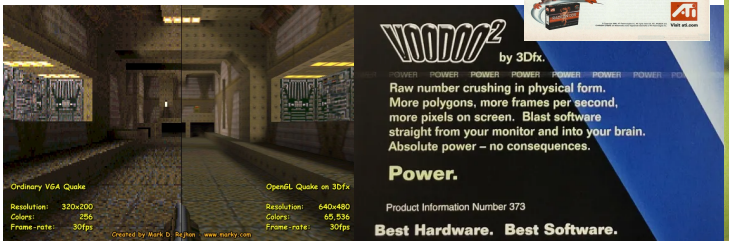
Mia Liu
Purdue University
West Lafayette, IN 47907, USA

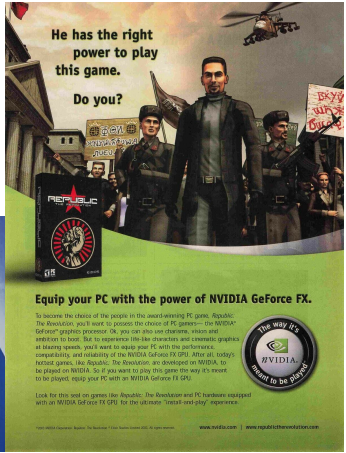
Edward Kreinar
HawkEye360
Herndon, VA 20170, USA

Zhenbin Wu
University of Illinois at Chicago
Chicago, IL 60607, USA



Primer: Računarska grafika





Zašto je specijalizacija hardvera korisna?

Jedan konkretan primer: skalarni proizvod dva vektora

$$s = \sum_{i=1}^{n} a_i b_i$$

Implementacija u C-u

```
unsigned dot_product(unsigned* a, unsigned* b, unsigned n)
{
    unsigned i = 0;
    unsigned s = 0;
    for(i = 0; i < n; ++i)
    {
        s += a[i] * b[i];
    }

    return s;
}
```

Kako izgleda asembli?

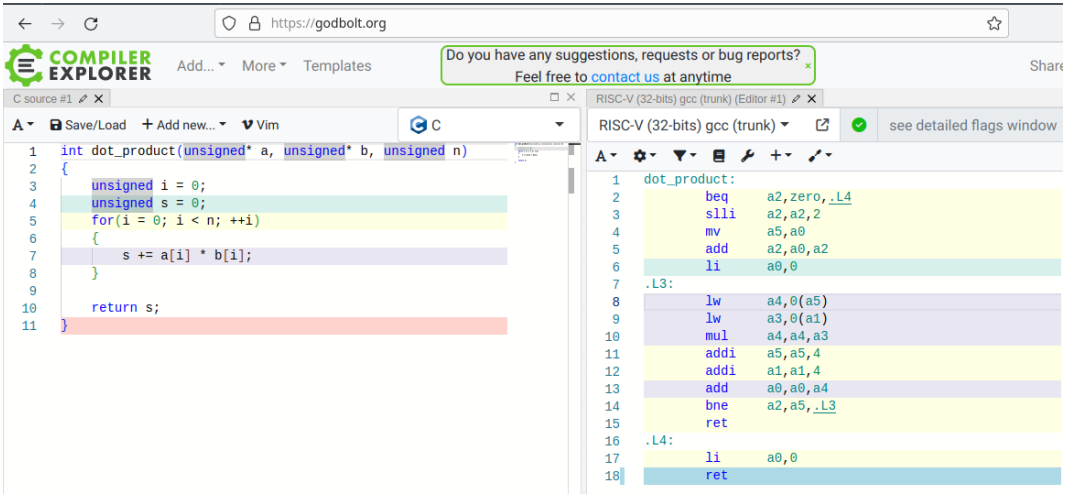
- pozivajuća funkcija kopira argumente u registre a0, a1, a2, redom

- povratna vrednost mora biti u registru a0 ili a1

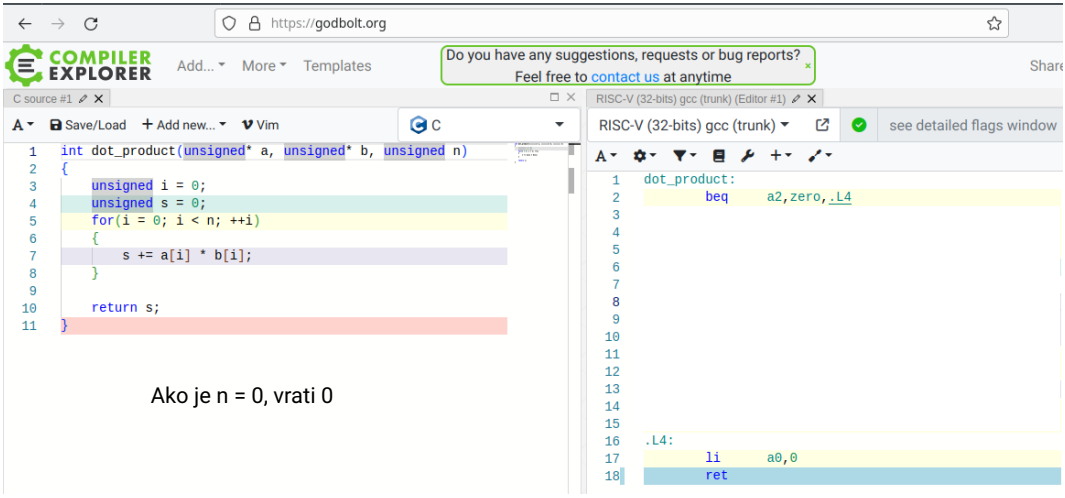
- unsigned zauzima 4 adrese

Kompajliramo najnovijom verzijom gcc-a sa -O3 opcijom

Kako izgleda asembli?

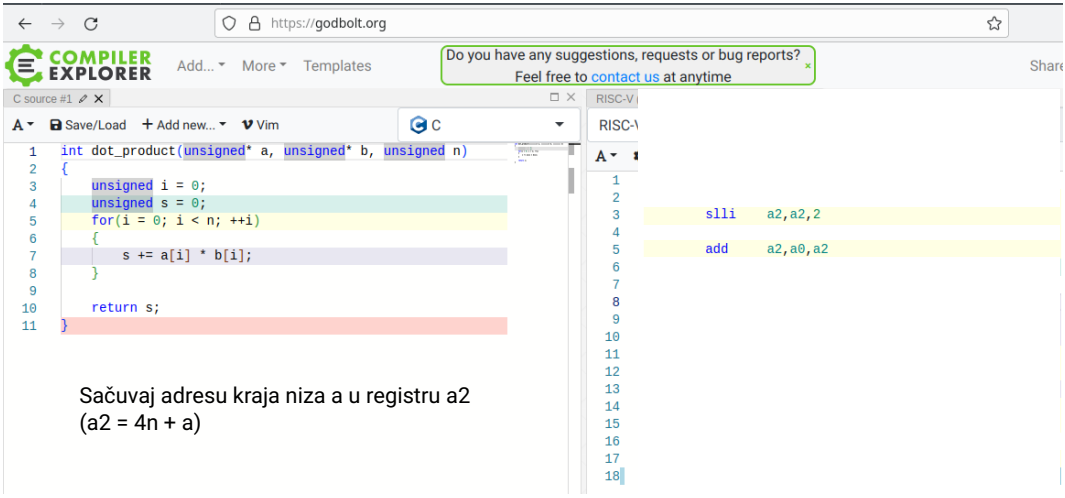


Kako izgleda asembli?



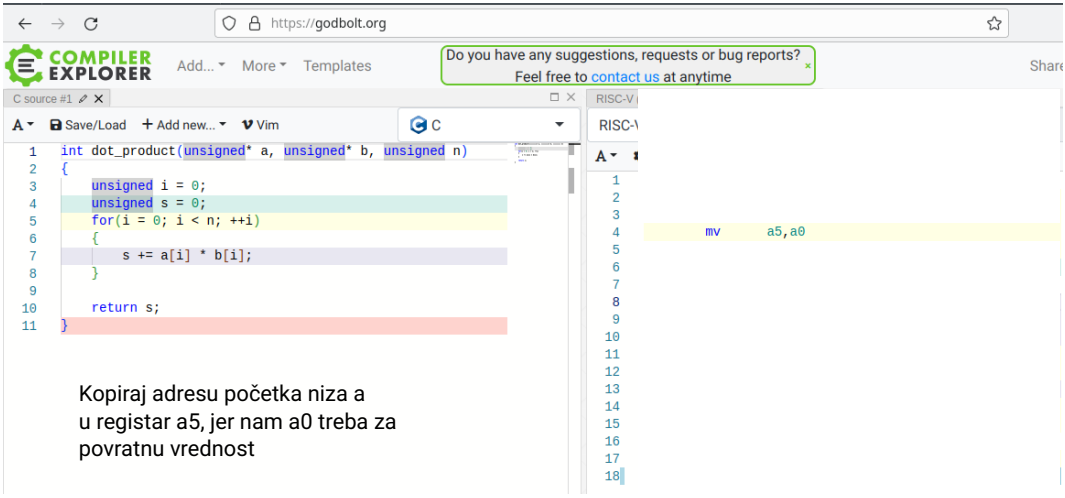
Ako je n = 0, vrati 0

Kako izgleda asembli?

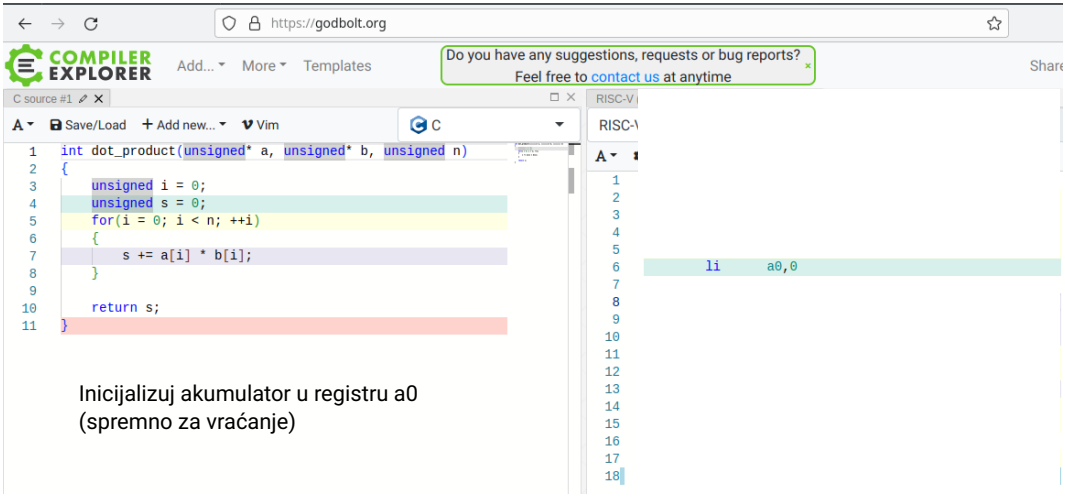


Sačuvaj adresu kraja niza a u registru a2
(a2 = 4n + a)

Kako izgleda asembli?



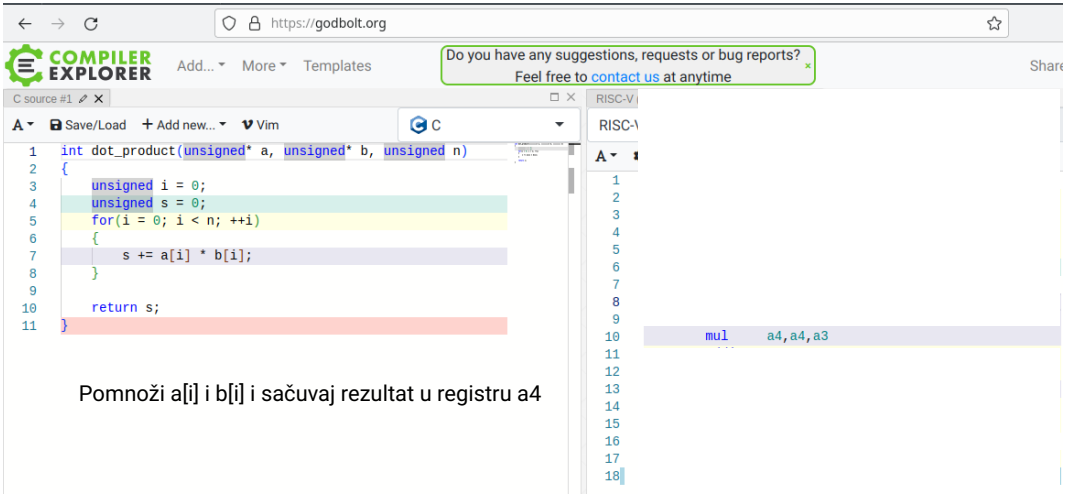
Kako izgleda asembli?



Kako izgleda asembli?

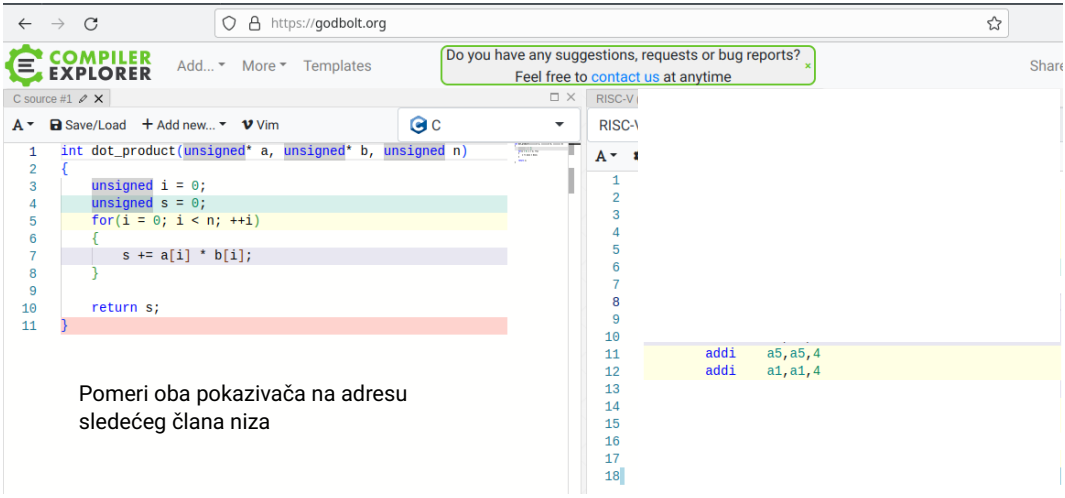
Učitaj iz memorije a[i] i b[i] u registre a4 i a3, redom

Kako izgleda asembli?



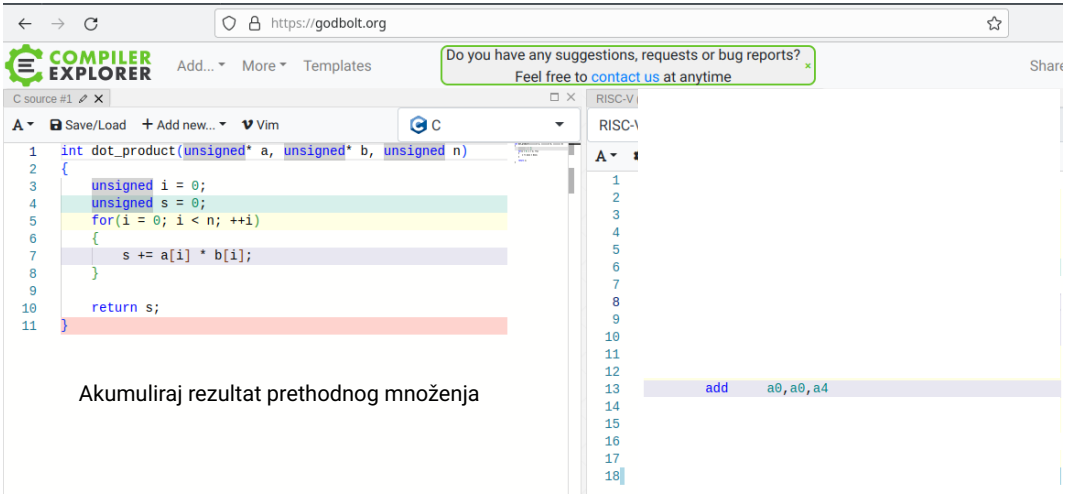
Pomnoži a[i] i b[i] i sačuvaj rezultat u registru a4

Kako izgleda asembli?



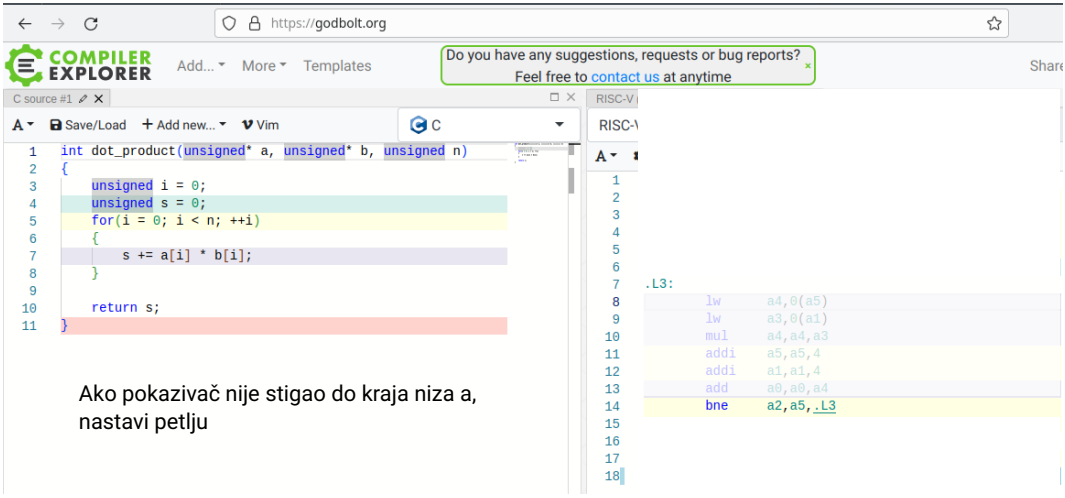
Pomeri oba pokazivača na adresu sledećeg člana niza

Kako izgleda asembli?



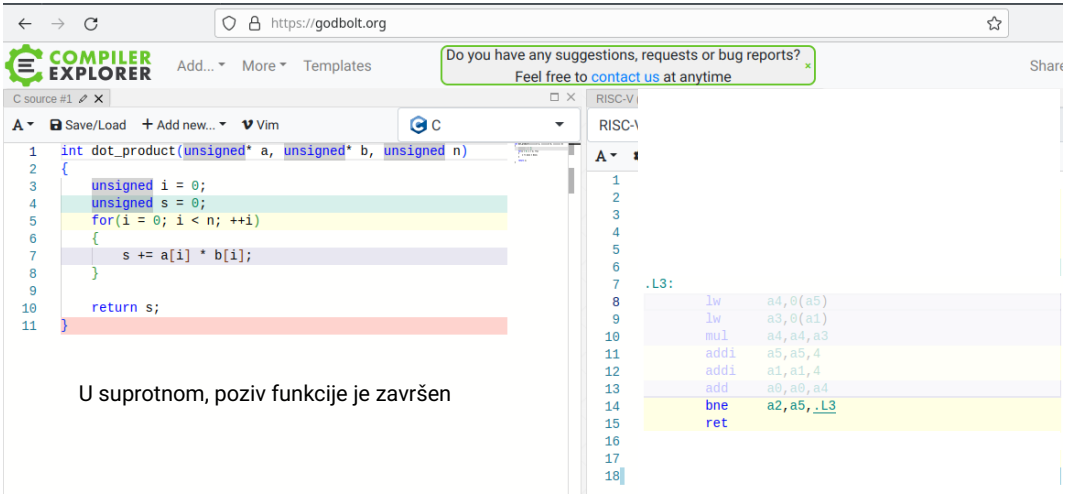
Akumuliraj rezultat prethodnog množenja

Kako izgleda asembli?



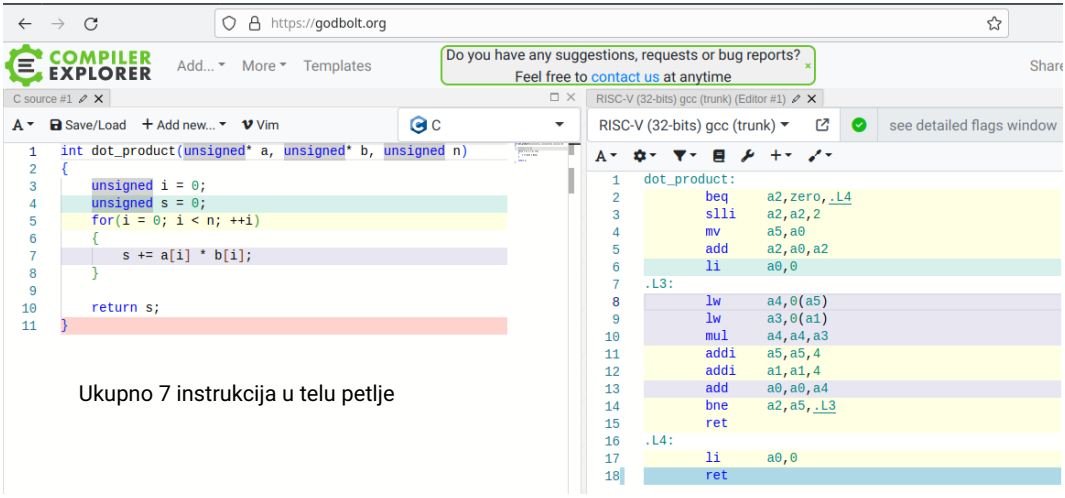
Ako pokazivač nije stigao do kraja niza a, nastavi petlju

Kako izgleda asembli?



U suprotnom, poziv funkcije je završen

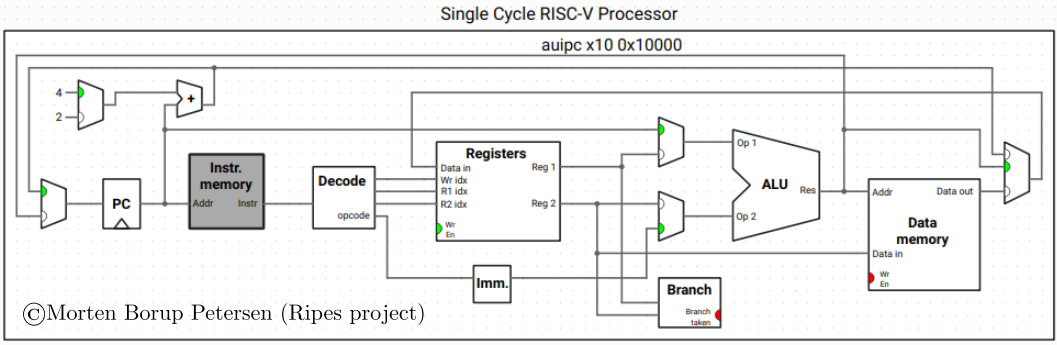
Kako izgleda asembli?



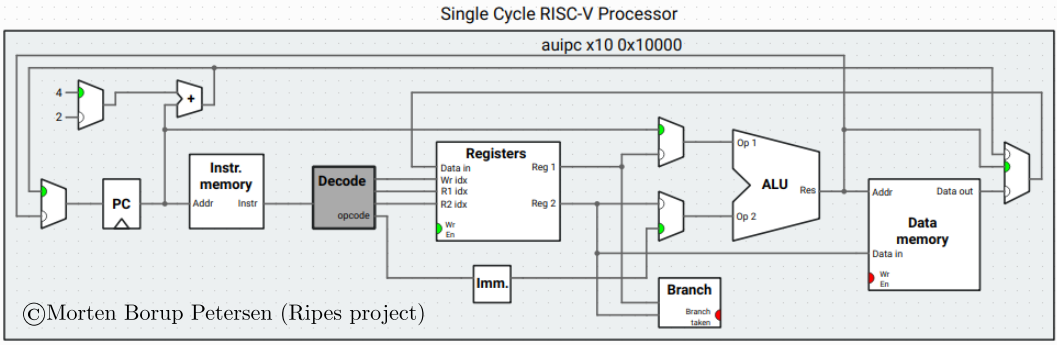
Ukupno 7 instrukcija u telu petlje

U čemu je problem?

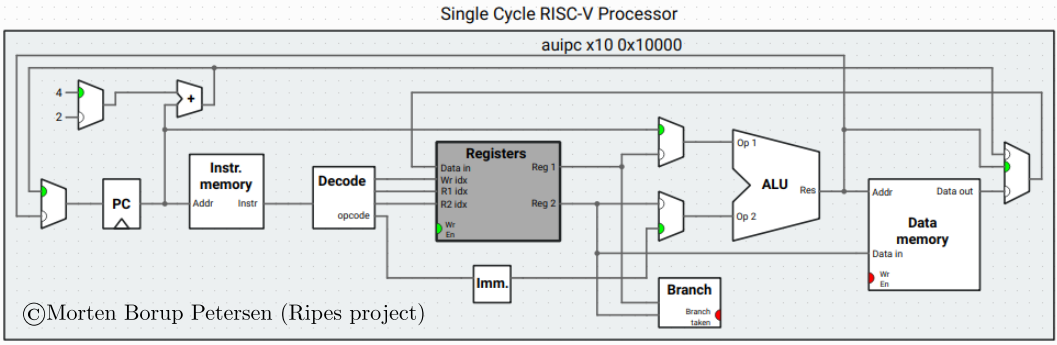
Instrukcijski ciklus



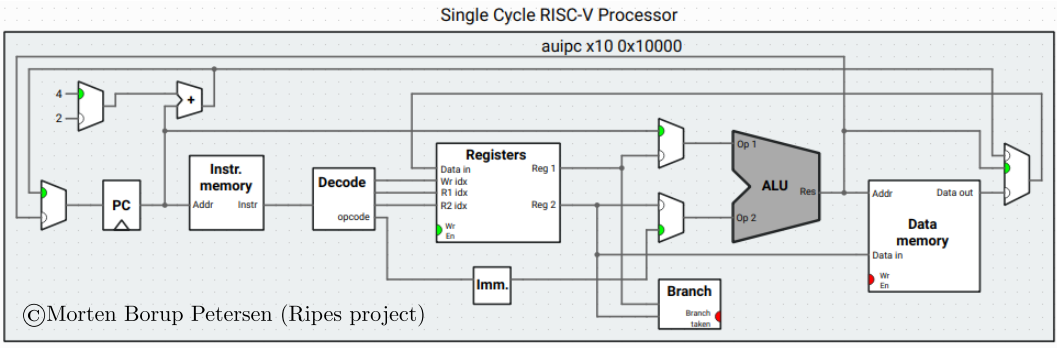
Instrukcijski ciklus



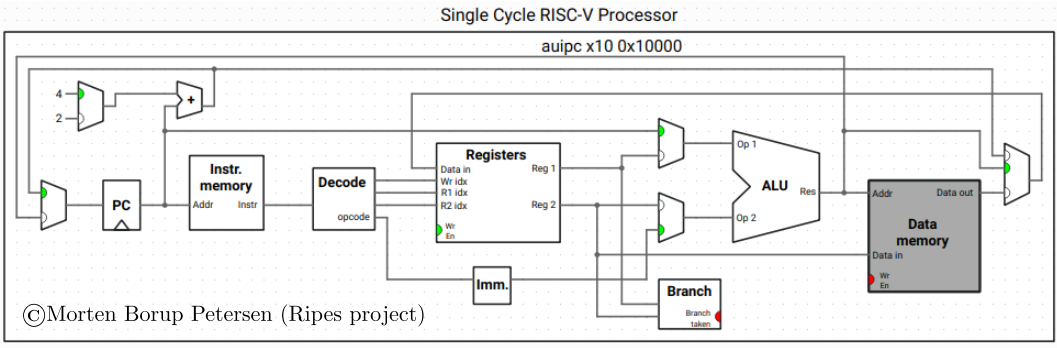
Instrukcijski ciklus



Instrukcijski ciklus



Instrukcijski ciklus



Nije u pitanju samo vreme

ISSCC 2014 / SESSION 1 / PLENARY / 1.1

1.1 Computing's Energy Problem (and what we can do about it)

Mark Horowitz

Departments of Electrical Engineering and Computer Science,
Stanford University, Stanford, CA

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

Memory	
Cache	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

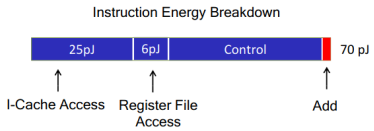


Figure 1.1.9: Rough energy costs for various operations in 45nm 0.9V.

Nije u pitanju samo vreme

ISSCC 2014 / SESSION 1 / PLENARY / 1.1

1.1 Computing's Energy Problem (and what we can do about it)

Mark Horowitz

Departments of Electrical Engineering and Computer Science,
Stanford University, Stanford, CA

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

Memory	
Cache	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

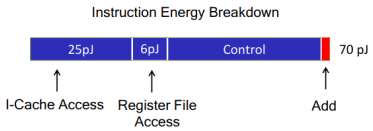
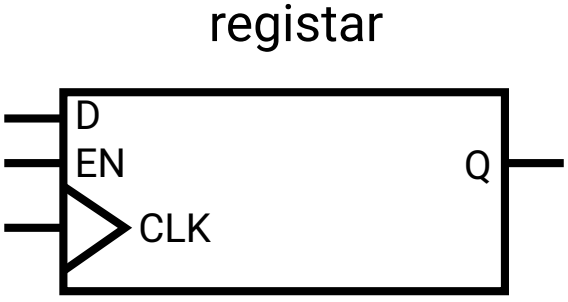


Figure 1.1.9: Rough energy costs for various operations in 45nm 0.9V.

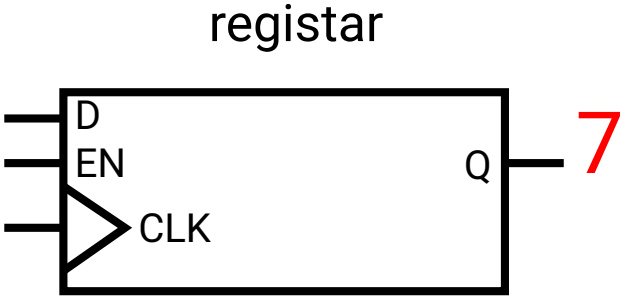
Samo računanje je zanemarljivo spram pomeranja instrukcija i podataka i upravljačkog podsistema

Šta možemo da unapredimo, ako znamo da je jedini zadatak sistema da računa skalarni proizvod dva vektora?

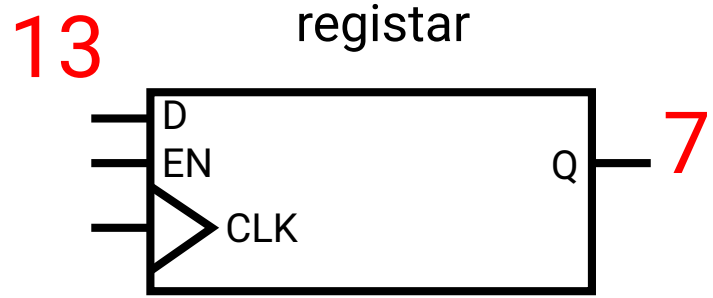
Mali podsetnik: registri



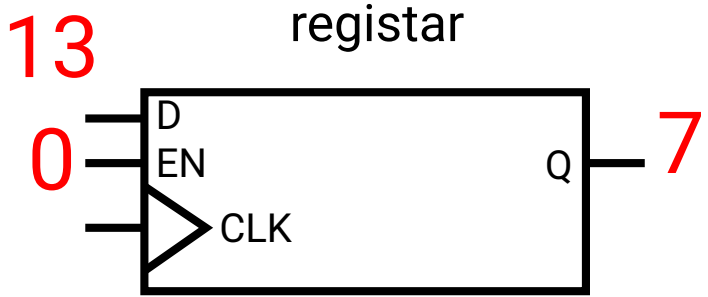
Mali podsetnik: registri



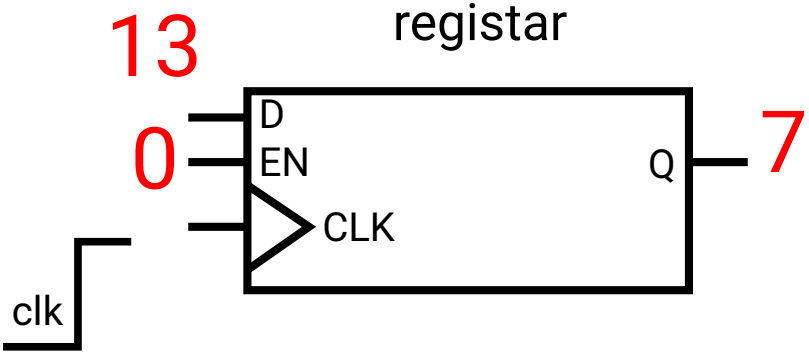
Mali podsetnik: registri



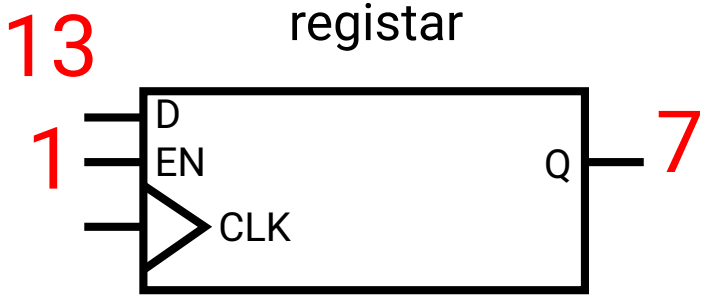
Mali podsetnik: registri



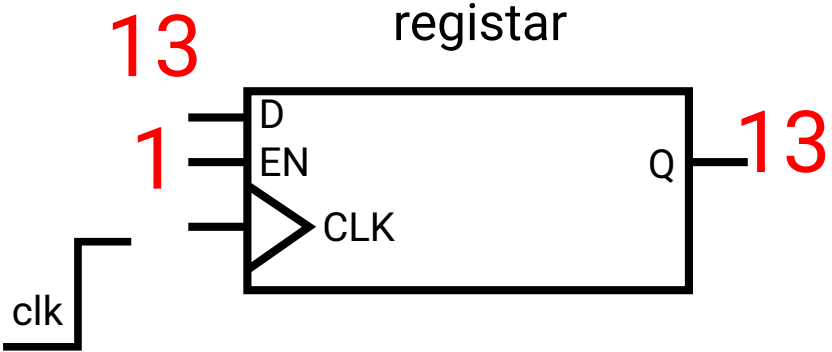
Mali podsetnik: registri



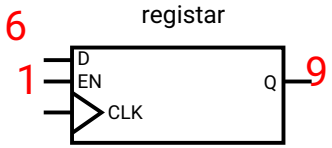
Mali podsetnik: registri

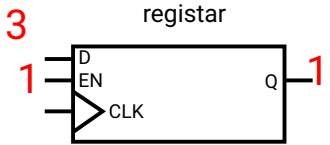


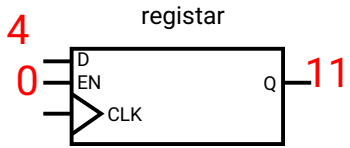
Mali podsetnik: registri



Mali podsetnik: registri

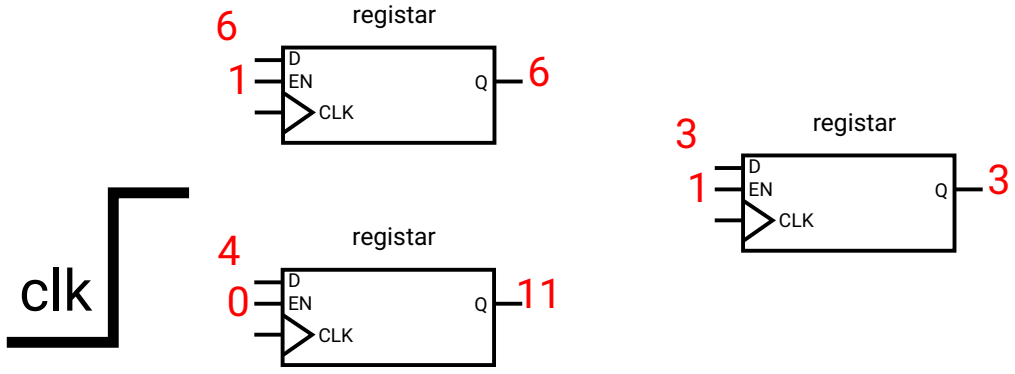






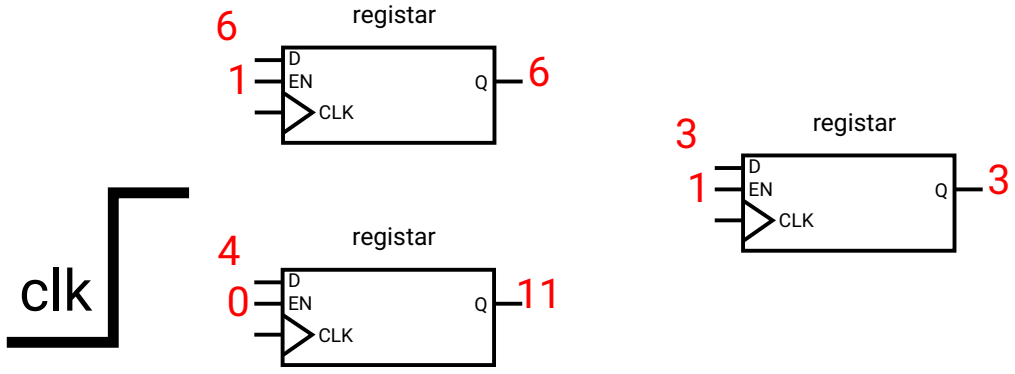
Svi registri na kojima je EN = 1 menjaju vrednost pri nailasku takta

Mali podsetnik: registri



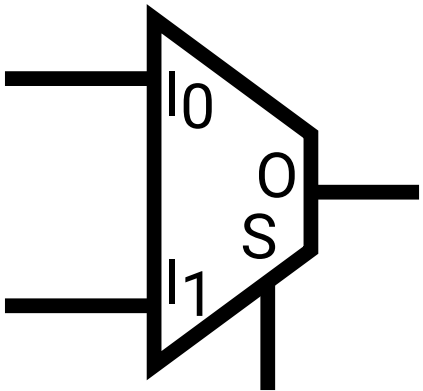
Svi registri na kojima je EN = 1 menjaju vrednost pri nailasku takta

Mali podsetnik: registri

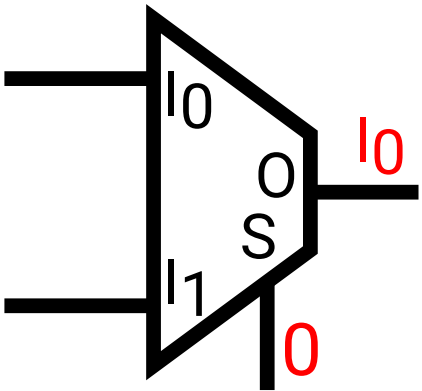


EN signale generiše upravljački podsistem

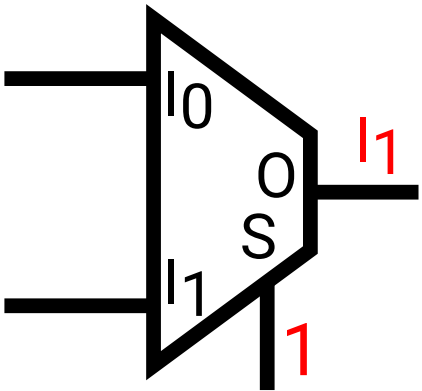
Mali podsetnik: multiplekseri



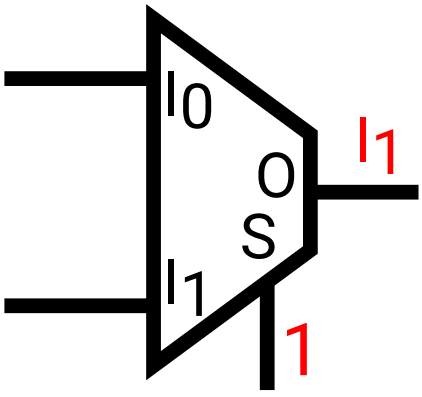
Mali podsetnik: multiplekseri



Mali podsetnik: multiplekseri



Mali podsetnik: multiplekseri



S signale generiše upravljački podsistem

Implementacija direktno u hardveru

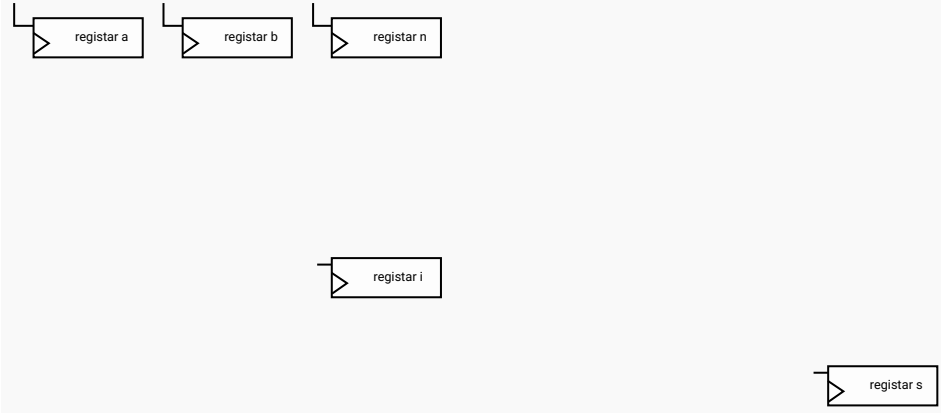
```
unsigned dot_product(unsigned* a, unsigned* b, unsigned n)
{
    unsigned i = 0;
    unsigned s = 0;
    for(i = 0; i < n; ++i)
    {
        s += a[i] * b[i];
    }

    return s;
}
```

Koliko nam je registara potrebno?

Implementacija direktno u hardveru

```
unsigned dot_product(unsigned* a, unsigned* b, unsigned n)
{
    unsigned i = 0;
    unsigned s = 0;
    for(i = 0; i < n; ++i)
    {
        s += a[i] * b[i];
    }

    return s;
}
```

Implementacija direktno u hardveru



Implementacija direktno u hardveru

```
1  unsigned dot_product(unsigned* a, unsigned* b, unsigned n)
2  {
3      unsigned i = 0;
4      unsigned s = 0;
5      for(i = 0; i < n; ++i)
6      {
7          s += a[i] * b[i];
8      }
9
10     return s;
11 }
```

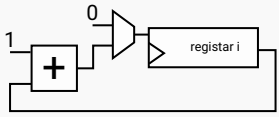
inicijalizacija (upis 0)

Implementacija direktno u hardveru

```
1  unsigned dot_product(unsigned* a, unsigned* b, unsigned n)
2  {
3      unsigned i = 0;
4      unsigned s = 0;
5      for(i = 0; i < n; ++i)
6      {
7          s += a[i] * b[i];
8      }
9
10     return s;
11 }
```

inkrement (upis i + 1)

Implementacija direktno u hardveru



Implementacija direktno u hardveru

```
1  unsigned dot_product(unsigned* a, unsigned* b, unsigned n)
2  {
3      unsigned i = 0;
4      unsigned s = 0;
5      for(i = 0; i < n; ++i)
6      {
7          s += a[i] * b[i];
8      }
9
10     return s;
11 }
```

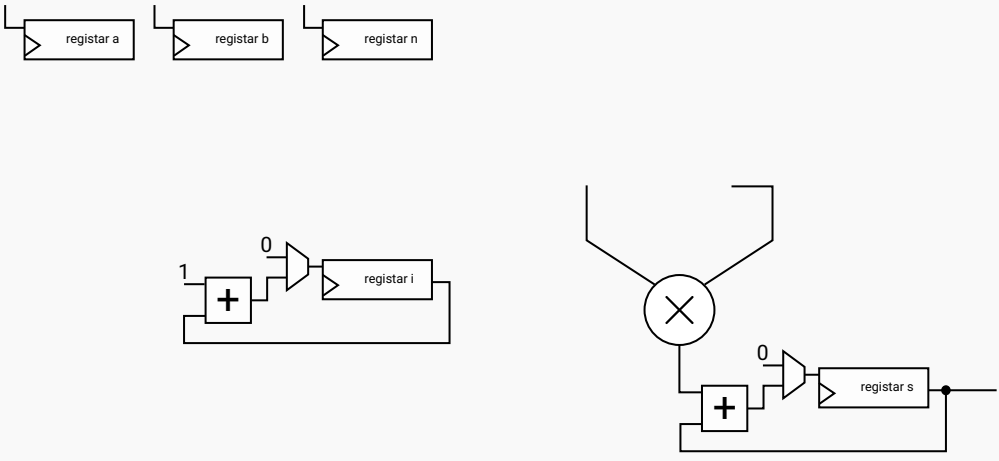
inicijalizacija (upis 0)

Implementacija direktno u hardveru

```
1  unsigned dot_product(unsigned* a, unsigned* b, unsigned n)
2  {
3      unsigned i = 0;
4      unsigned s = 0;
5      for(i = 0; i < n; ++i)
6      {
7          s += a[i] * b[i];
8      }
9
10     return s;
11 }
```

akumulacija (upis s + a[i]*b[i])

Implementacija direktno u hardveru

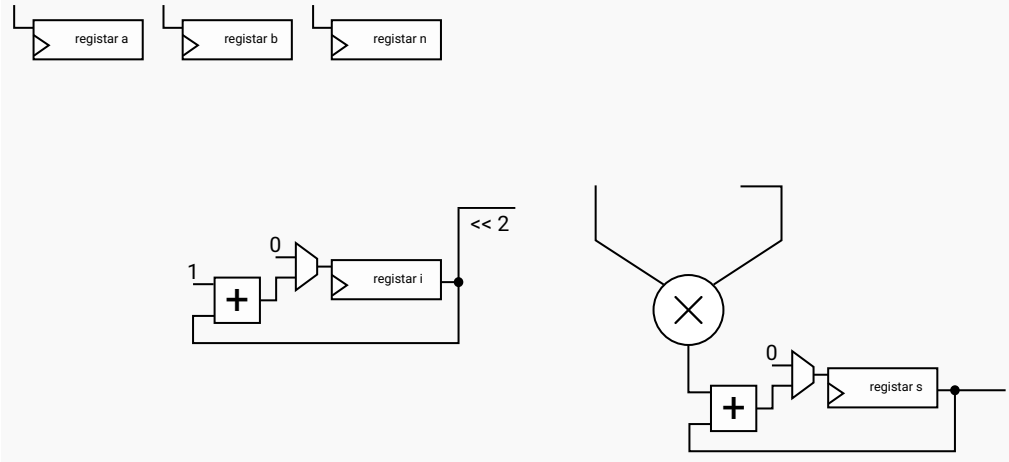


Implementacija direktno u hardveru

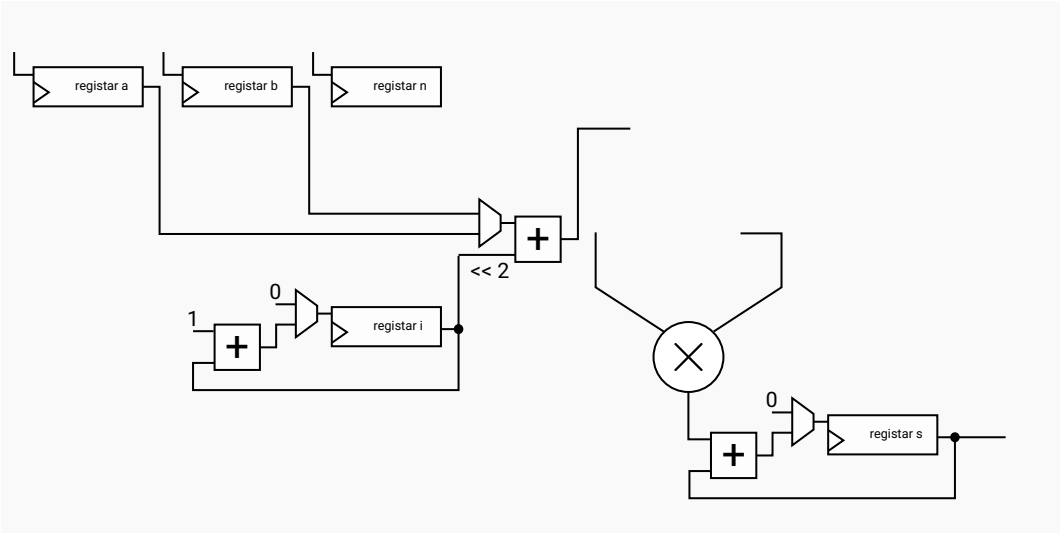
```
unsigned dot_product(unsigned* a, unsigned* b, unsigned n)
{
    unsigned i = 0;
    unsigned s = 0;
    for(i = 0; i < n; ++i)
    {
        s += a[i] * b[i];
    }

    return s;
}
```

(*(a + 4i)) * (*(b + 4i))

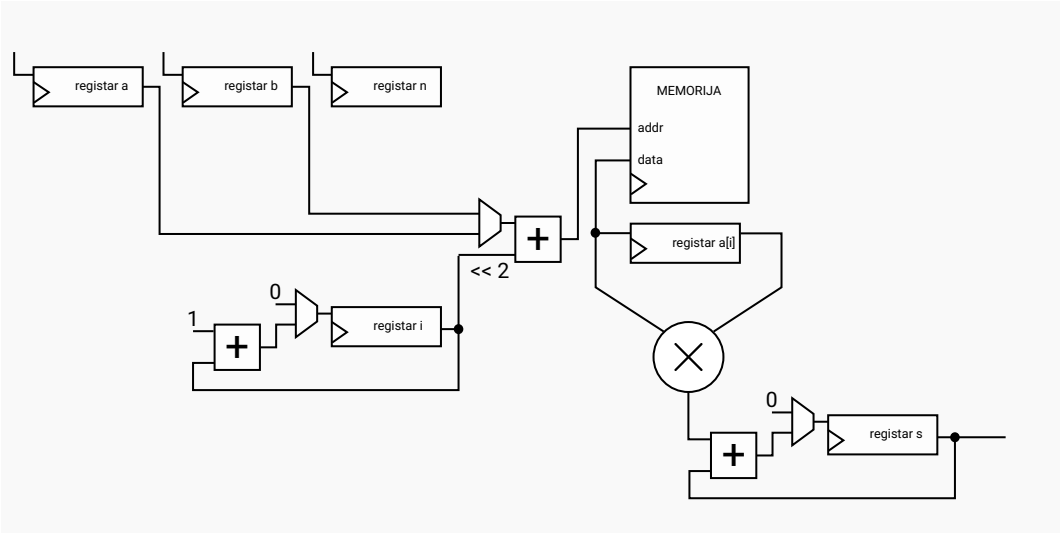
Implementacija direktno u hardveru



Implementacija direktno u hardveru



Implementacija direktno u hardveru

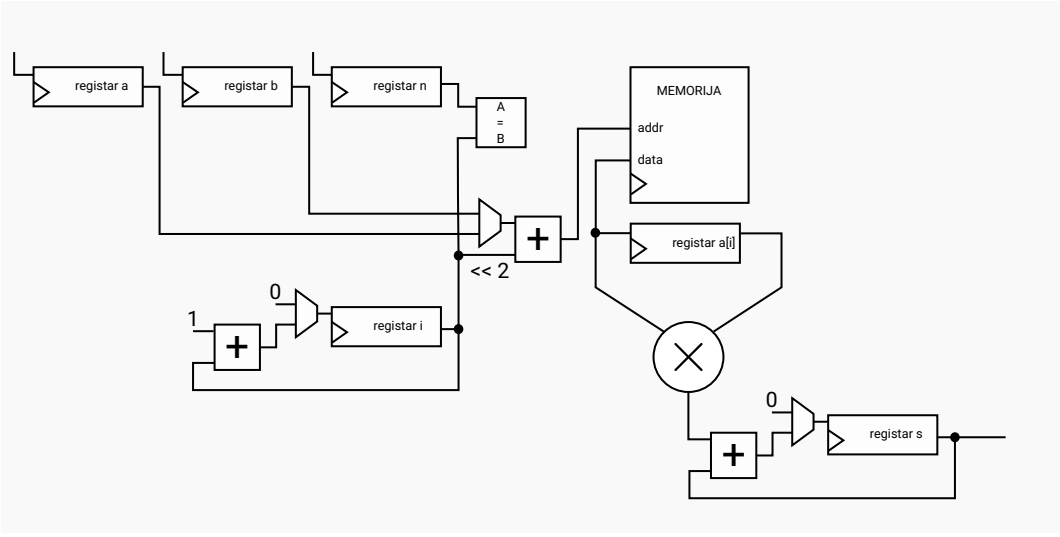


Implementacija direktno u hardveru

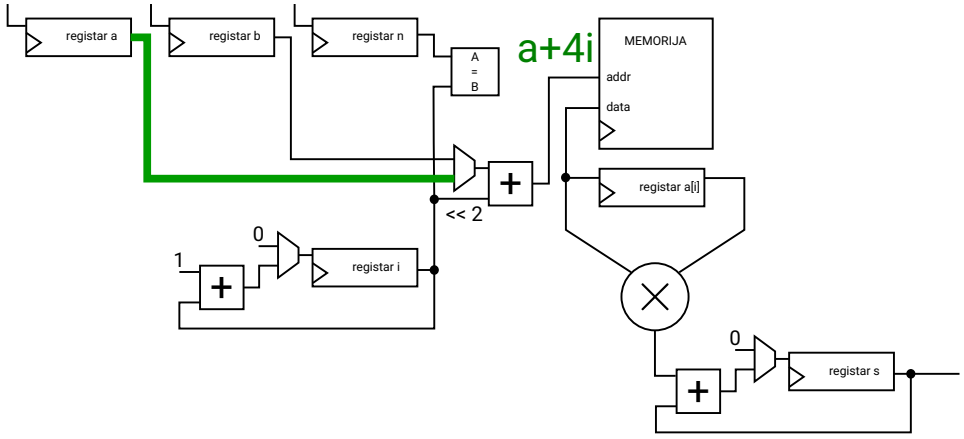
```
1  unsigned dot_product(unsigned* a, unsigned* b, unsigned n)
2  {
3      unsigned i = 0;
4      unsigned s = 0;
5      for(i = 0; i < n; ++i)
6      {
7          s += a[i] * b[i];
8      }
9
10     return s;
11 }
```

poređenje (bitno za upravljački podsistem)

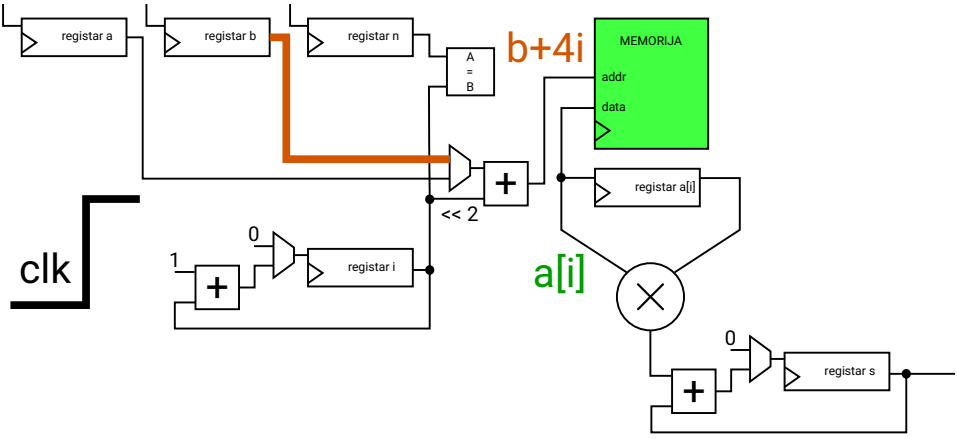
Implementacija direktno u hardveru



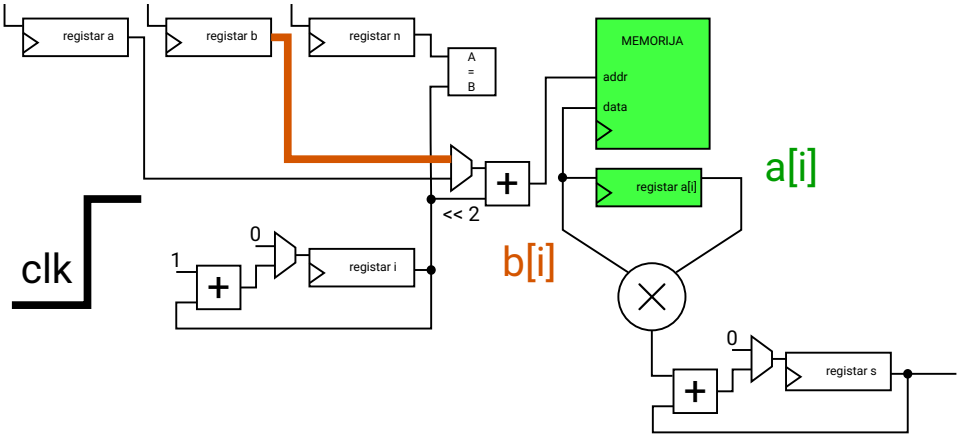
Implementacija direktno u hardveru



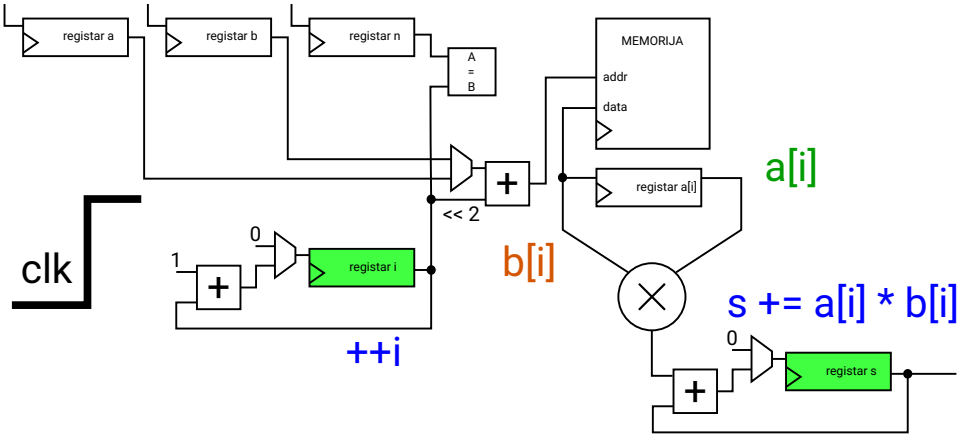
Implementacija direktno u hardveru



Implementacija direktno u hardveru



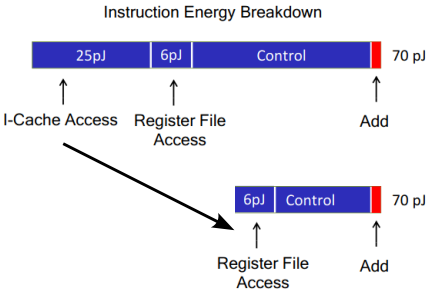
Implementacija direktno u hardveru



Inherentni paralelizam: jedan takt menja tri instrukcije
(addi a5, a5, 4; addi a1, a1, 4; add a0, a0, a4)

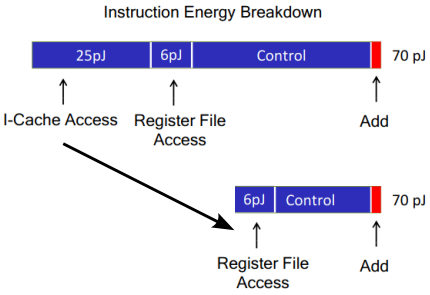
Šta smo do sada postigli?

- 3 periode takta po iteraciji umesto najmanje 7
- kraća minimalna perioda takta (nema zahvata instrukcije, dekodiranja...)



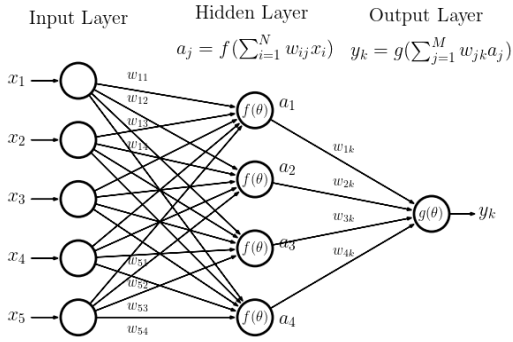
Šta smo do sada postigli?

- 3 periode takta po iteraciji umesto najmanje 7
- kraća minimalna perioda takta (nema zahvata instrukcije, dekodiranja...)



Nismo još završili...

Gde se često koriste skalarni proizvodi?


Input Layer
Hidden Layer
Output Layer
$a_j = f(\sum_{i=1}^{N} w_{ij} x_i)$
$y_k = g(\sum_{j=1}^{M} w_{jk} a_j)$
x_1
x_2
x_3
x_4
x_5
w_{11}
w_{12}
w_{13}
w_{14}
w_{51}
w_{52}
w_{53}
w_{54}
$f(\theta)$
a_1
a_2
a_3
a_4
w_{1k}
w_{2k}
w_{3k}
w_{4k}
$g(\theta)$
y_k

Da li je u neuralnim mrežama neophodno koristiti 32 bita?

Binarized Neural Networks

Itay Hubara[1]*
itayh@technion.ac.il

Matthieu Courbariaux[2]*
matthieu.courbariaux@gmail.com

Daniel Soudry[3]
daniel.soudry@gmail.com

Ran El-Yaniv[1]
rani@cs.technion.ac.il

Yoshua Bengio[2,4]
yoshua.umontreal@gmail.com

(1) Technion, Israel Institute of Technology.
(2) Université de Montréal.
(3) Columbia University.
(4) CIFAR Senior Fellow.
(*) Indicates equal contribution.

Šta bismo dobili smanjenjem broja bita?

Mala pomoć

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

$$\frac{0.2}{8^2} \times 32^2 = 3.2$$

Računska složenost množenja

$$
\begin{array}{r}
5739 \times 4846 \\
\hline
34434 \\
22956 \\
45912 \\
+\,22956 \\
\hline
27811194
\end{array}
$$

$\mathcal{O}(n^2)$ operacija $\propto$ površini na čipu

($\mathcal{O}(n \log n)$ za [Harvey and van der Hoeven, 2021])

Smanjenje širine sa 32 na 8 bita

- na istu površinu možemo smestiti 6-16× više množača[1]

[1] Zanemarujući konstante.

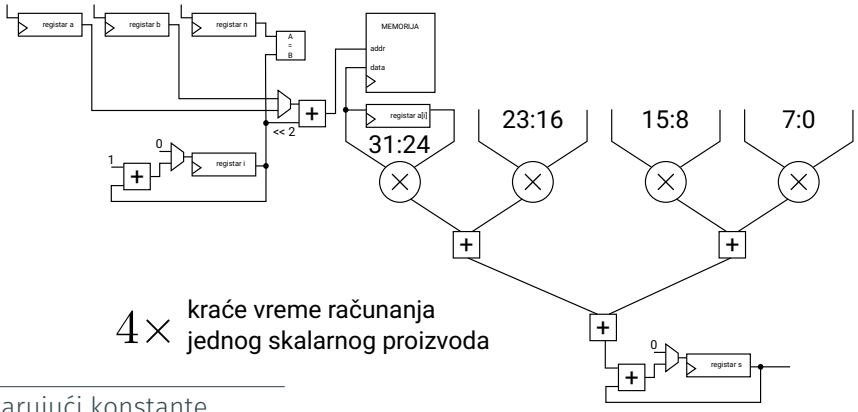
Smanjenje širine sa 32 na 8 bita

- na istu površinu možemo smestiti 6-16× više množača[1]
- sa 4× više nam je još uvek dovoljan 32-bitni pristup memoriji

[1] Zanemarujući konstante.

Smanjenje širine sa 32 na 8 bita

- na istu površinu možemo smestiti 6-16× više množača[1]
- sa 4× više nam je još uvek dovoljan 32-bitni pristup memoriji



$4\times$ kraće vreme računanja jednog skalarnog proizvoda

[1]Zanemarujući konstante.

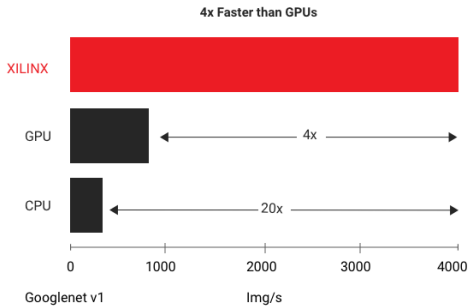
Rezultati u praksi

Vitis AI in the Data Center

AMD delivers the highest throughput at the lowest latency. In standard benchmark tests on GoogleNet V1, the AMD Alveo U250 platform delivers more than 4x the throughput of the fastest existing GPU for real-time inference. Learn more in the whitepaper: Accelerating DNNs with AMD Alveo Accelerator Cards

AI in the Data Center eBook

Download eBook



Zaključci

Prednosti specijalizovanog hardvera:

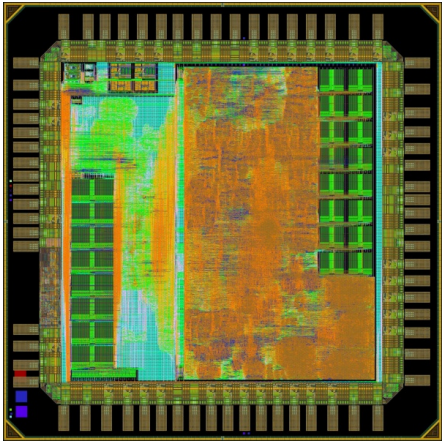
- Nema zahvata instrukcija iz memorije
- Upravljački podsistem je minimalan
- Aritmetika je prilagođena problemu

Sve to dovodi do povećanja brzine izvršenja i/ili smanjenja potrošnje energije

Kako nastaje računarski hardver?

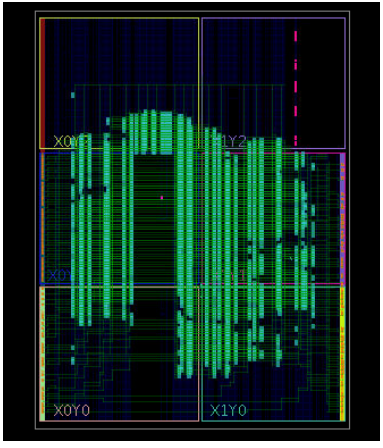
Krajnji ishod

©ETH Zürich



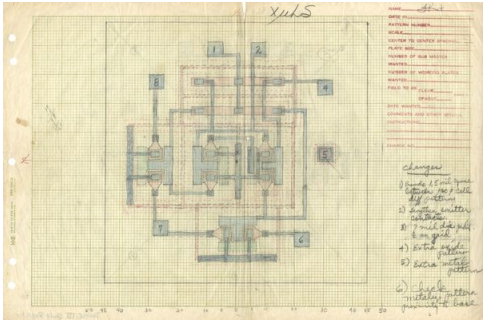
Izgled lejauta čipa "Pulp v3"

©Xilinx, Inc.

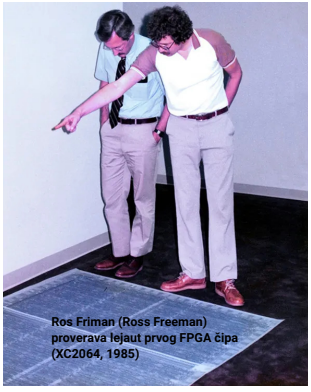


Izgled jedne FPGA konfiguracije

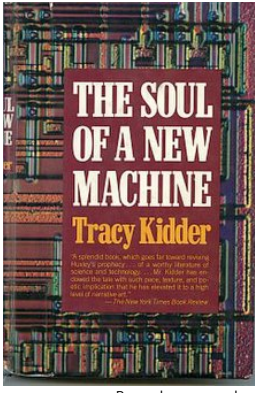
Nekada je gotovo sve bilo ručno



Ručno crtana litografska maska za pomerački registar
(Fairchild Camera and Instrument Corporation, 1960)



Ros Friman (Ross Freeman) proverava lejaut prvog FPGA čipa (XC2064, 1985)



Preporuka za razonodu

No, teorija je bila dobro razvijena

Klod Šenon, 1937.
(u pripremi master rada na MIT-u)

A Symbolic Analysis of Relay and Switching Circuits*

*Claude E. Shannon***

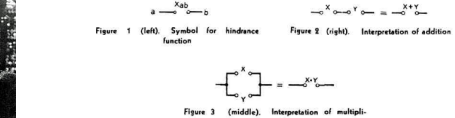


Figure 1 (left). Symbol for hindrance function

Figure 2 (right). Interpretation of addition

Figure 3 (middle). Interpretation of multiplication

It is evident that with the above definitions the following postulates will hold:

Postulates

1. *a.*	$0 \cdot 0 = 0$	A closed circuit in parallel with a closed circuit is a closed circuit.
b.	$1 + 1 = 1$	An open circuit in series with an open circuit is an open circuit.
2. *a.*	$1 + 0 = 0 + 1 = 1$	An open circuit in series with a closed circuit in either order (i.e., whether the open circuit is to the right or left of the closed circuit) is an open circuit.
b.	$0 \cdot 1 = 1 \cdot 0 = 0$	A closed circuit in parallel with an open circuit in either order is a closed circuit.
3. *a.*	$0 + 0 = 0$	A closed circuit in series with a closed circuit is a closed circuit.
b.	$1 \cdot 1 = 1$	An open circuit in parallel with an open circuit is an open circuit.

4. At any given time either $X = 0$ or $X = 1$.

1941 АВТОМАТИКА И ТЕЛЕМЕХАНИКА № 2

АЛГЕБРА ДВУХПОЛЮСНЫХ СХЕМ, ПОСТРОЕННЫХ ИСКЛЮЧИТЕЛЬНО ИЗ ДВУХПОЛЮСНИКОВ (АЛГЕБРА А-СХЕМ)*

Канд. физ.-мат. наук **В. И. Шестаков**

ВЗАИМНО-ОДНОЗНАЧНОЕ СООТВЕТСТВИЕ МЕЖДУ А-СХЕМАМИ И А-ВЫРАЖЕНИЯМИ

А-схемы

Viktor Ivanovič Šestakov, 1935.
(u pripremi doktorata na Lomonosovu)

Prekidačka kola je moguće predstaviti kao Bulove funkcije.

Formalizacija otvara prostor za automatizaciju

Danas razvoj hardvera ima mnogo sličnosti sa razvojem softvera

```
// Generalized FIR filter parameterized by the convolution coefficients
class FirFilter(bitWidth: Int, coeffs: Seq[UInt]) extends Module {
  val io = IO(new Bundle {
    val in = Input(UInt(bitWidth.W))
    val out = Output(UInt(bitWidth.W))
  })
  // Create the serial-in, parallel-out shift register
  val zs = Reg(Vec(coeffs.length, UInt(bitWidth.W)))
  zs(0) := io.in
  for (i <- 1 until coeffs.length) {
    zs(i) := zs(i-1)
  }

  // Do the multiplies
  val products = VecInit.tabulate(coeffs.length)(i => zs(i) * coeffs(i))

  // Sum up the products
  io.out := products.reduce(_ + _)
}
```

CHISEL

Berkeley
UNIVERSITY OF CALIFORNIA

Umesto crtanja šema, za opis hardvera
se koriste formalni jezici (HDL = Hardware Description Language)

and use and re-use them across designs:

```
val movingSum3Filter = Module(new FirFilter(8, Seq(1.U, 1.U, 1.U)))  // same 3-point moving sum filter as before
val delayFilter = Module(new FirFilter(8, Seq(0.U, 1.U)))  // 1-cycle delay as a FIR filter
val triangleFilter = Module(new FirFilter(8, Seq(1.U, 2.U, 3.U, 2.U, 1.U)))  // 5-point FIR filter with a triangle impulse res
ponse
```

Jezik za opis hardvera "CHISEL", zasnovan na programskom jeziku Scala

Prevođenje izvornog koda u međupredstavu je skoro identično

Code Generation for Silicon

S. C. Johnson

Bell Laboratories
Murray Hill, New Jersey 07974

Autor YACC-a i prvog lintera

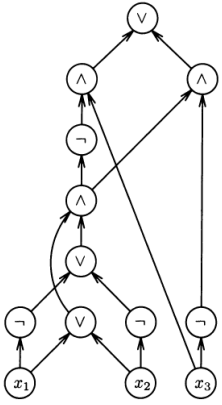
ACM POPL '83

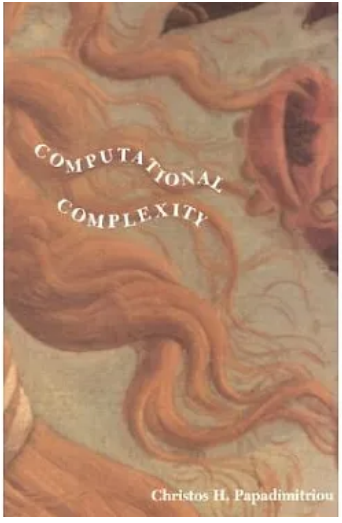
EXTENDED ABSTRACT

Compiler writers face a bleak future—soon everyone will be using Ada on the Nebula Architecture and there will be no further need for our services. Luckily, it appears to be possible to recycle compiler writers by encouraging them to build circuit design tools (Silicon Compilers). Many of the standard techniques used in compilers can be immediately applied in circuit design, especially for LSI circuits. In part, this paper attempts to serve as a tutorial, casting some of the problems of designing in nMOS with the Mead-Conway design rules into the vocabulary of the compiler writer. The paper also discusses experience with a Silicon compiler that has fabricated over two dozen different designs.

Međupredstava je najčešće Bulovo kolo

$$(x_3 \wedge \neg((x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2))) \vee (\neg x_3 \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2))$$





Međupredstava je najčešće Bulovo kolo

Softver na klasičnom procesoru

Specijalizovan hardver

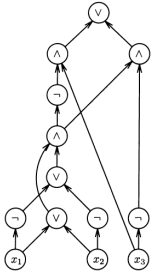
$p \in K,$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
$s,$	0	$(s, 0, \rightarrow)$
$s,$	1	$(s, 1, \rightarrow)$
$s,$	$\sqcup$	$(q, \sqcup, \leftarrow)$
$s,$	$\triangleright$	$(s, \triangleright, \rightarrow)$
$q,$	0	$(q_0, \sqcup, \rightarrow)$
$q,$	1	$(q_1, \sqcup, \rightarrow)$
$q,$	$\sqcup$	$(q, \sqcup, -)$
$q,$	$\triangleright$	$(h, \triangleright, \rightarrow)$
$q_0,$	0	$(s, 0, \leftarrow)$
$q_0,$	1	$(s, 0, \leftarrow)$
$q_0,$	$\sqcup$	$(s, 0, \leftarrow)$
$q_0,$	$\triangleright$	$(h, \triangleright, \rightarrow)$
$q_1,$	0	$(s, 1, \leftarrow)$
$q_1,$	1	$(s, 1, \leftarrow)$
$q_1,$	$\sqcup$	$(s, 1, \leftarrow)$
$q_1,$	$\triangleright$	$(h, \triangleright, \rightarrow)$

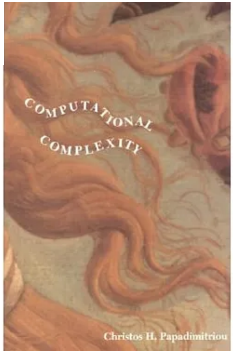
0. $s, \quad \underline{\triangleright}010$
1. $s, \quad \triangleright\underline{0}10$
2. $s, \quad \triangleright0\underline{1}0$
3. $s, \quad \triangleright01\underline{0}$
4. $s, \quad \triangleright010\underline{\sqcup}$
5. $q, \quad \triangleright01\underline{0}\sqcup$
6. $q_0, \quad \triangleright01 \sqcup \underline{\sqcup}$
7. $s, \quad \triangleright01\underline{\sqcup}0$
8. $q, \quad \triangleright0\underline{1} \sqcup 0$
9. $q_1, \quad \triangleright0 \sqcup \underline{\sqcup}0$
10. $s, \quad \triangleright0\underline{\sqcup}10$
11. $q, \quad \triangleright\underline{0} \sqcup 10$
12. $q_0, \quad \triangleright \sqcup \underline{\sqcup}10$
13. $s, \quad \triangleright\underline{\sqcup}010$
14. $q, \quad \underline{\triangleright} \sqcup 010$
15. $h, \quad \triangleright\underline{\sqcup}010$

Figure 2.1. Turing machine and computation.

4.3 Boolean Functions and Circuits 81

$$(x_3 \wedge \neg((x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2))) \vee (\neg x_3 \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2))$$

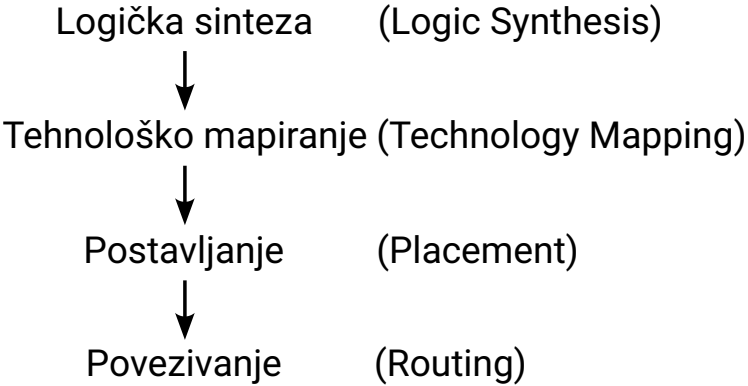




Tjuringova mašina: inherentno sekvencijalan model računanja

Bulovo kolo: inherentno paralelan model računanja

Koraci u generisanju lejauta



Logička sinteza (Logic Synthesis)

- Nakon dobijanja Bulovog kola, proces logičke sinteze ga optimizuje, tako da mu broj čvorova i/ili dubina budu što manji

Logička sinteza (Logic Synthesis)

- Nakon dobijanja Bulovog kola, proces logičke sinteze ga optimizuje, tako da mu broj čvorova i/ili dubina budu što manji
- Korak analogan kompajlerskim optimizacijama nezavisnim od ciljne arhitekture, u kompilaciji softvera

Ključne odlike problema u logičkoj sintezi

- Probleme je najčešće veoma jednostavno formulisati

Ključne odlike problema u logičkoj sintezi

- Probleme je najčešće veoma jednostavno formulisati
- Metrike za ocenu kvaliteta rešenja su jasno definisane

Ključne odlike problema u logičkoj sintezi

- Probleme je najčešće veoma jednostavno formulisati
- Metrike za ocenu kvaliteta rešenja su jasno definisane
- Velik praktični značaj

Ključne odlike problema u logičkoj sintezi

- Probleme je najčešće veoma jednostavno formulisati
- Metrike za ocenu kvaliteta rešenja su jasno definisane
- Velik praktični značaj
- Nalaženje globalnog optimuma je poželjno,
  ali je i poboljšanje lokalnog važno

Ključne odlike problema u logičkoj sintezi

- Probleme je najčešće veoma jednostavno formulisati
- Metrike za ocenu kvaliteta rešenja su jasno definisane
- Velik praktični značaj
- Nalaženje globalnog optimuma je poželjno,
  ali je i poboljšanje lokalnog važno
- Ogromne instance

Ključne odlike problema u logičkoj sintezi

- Probleme je najčešće veoma jednostavno formulisati
- Metrike za ocenu kvaliteta rešenja su jasno definisane
- Velik praktični značaj
- Nalaženje globalnog optimuma je poželjno,
  ali je i poboljšanje lokalnog važno
- Ogromne instance
- Teški kombinatorni problemi (najčešće dokazano NP-teški)

Ključne odlike problema u logičkoj sintezi

- Probleme je najčešće veoma jednostavno formulisati
- Metrike za ocenu kvaliteta rešenja su jasno definisane
- Velik praktični značaj
- Nalaženje globalnog optimuma je poželjno,
  ali je i poboljšanje lokalnog važno
- Ogromne instance
- Teški kombinatorni problemi (najčešće dokazano NP-teški)

Posledica: razvijen je velik broj algoritama i struktura podataka za rešavanje bazičnih kombinatornih problema koji nemaju bolje asimptotsko ponašanje ali nalaze rešenja za velike probleme od praktičnog značaja

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj:

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj:

$$f(a, b) = (\neg a \wedge f(a = 0, b)) \vee (a \wedge f(a = 1, b))$$

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj:

$$f(a,b) = (\neg a \wedge f(a=0,b)) \vee (a \wedge f(a=1,b))$$

$$f(a=0,b) = (\neg b \wedge f(a=0,b=0)) \vee (b \wedge f(a=0,b=1))$$

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj:

$f(a, b) = (\neg a \wedge f(a = 0, b)) \vee (a \wedge f(a = 1, b))$

$f(a = 0, b) = (\neg b \wedge f(a = 0, b = 0)) \vee (b \wedge f(a = 0, b = 1))$

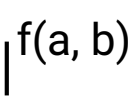
$f(a = 0, b = 1) = const. \in \{0, 1\}$

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj je moguće prikazati grafički pomoću binarnog diagrama odluke:

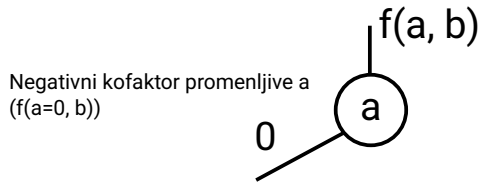
Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj je moguće prikazati grafički pomoću binarnog diagrama odluke:



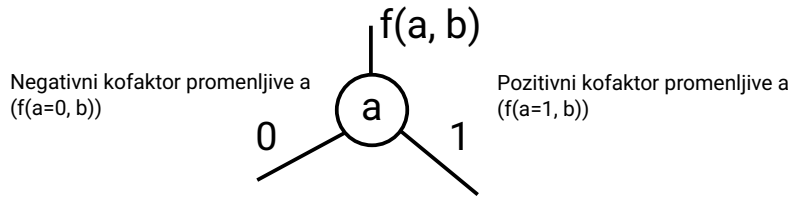
Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj je moguće prikazati grafički pomoću binarnog diagrama odluke:



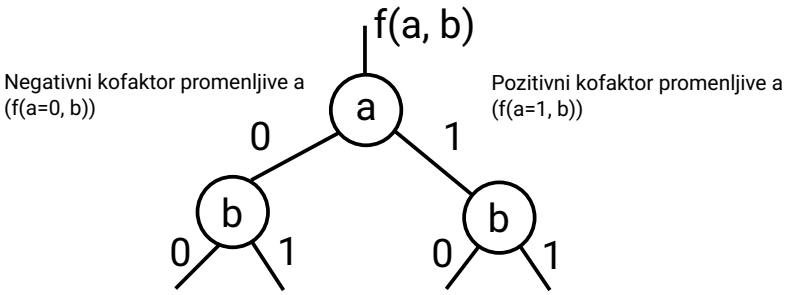
Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj je moguće prikazati grafički pomoću binarnog diagrama odluke:



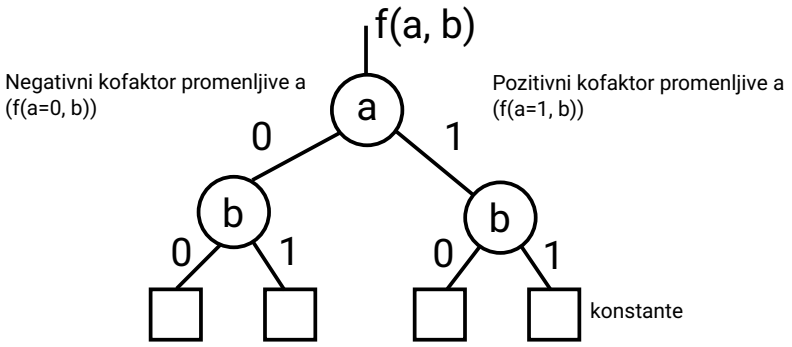
Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj je moguće prikazati grafički pomoću binarnog diagrama odluke:



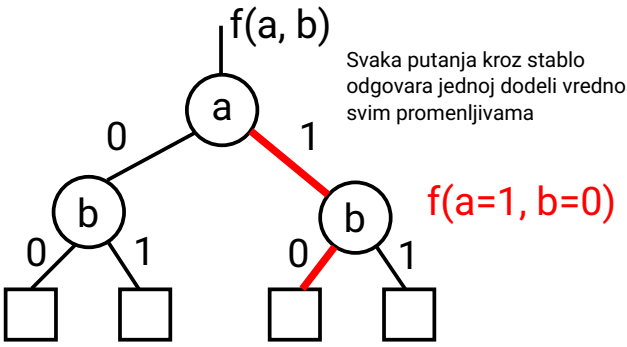
Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj je moguće prikazati grafički pomoću binarnog diagrama odluke:



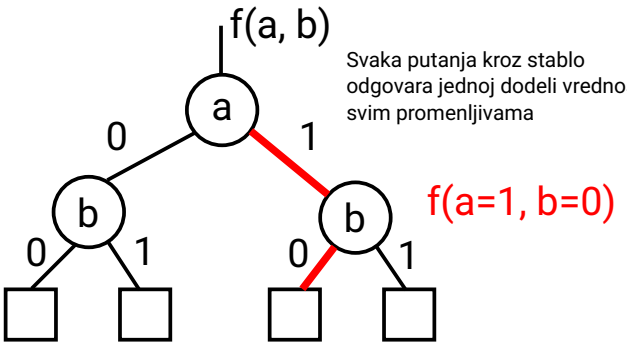
Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj je moguće prikazati grafički pomoću binarnog diagrama odluke:



Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

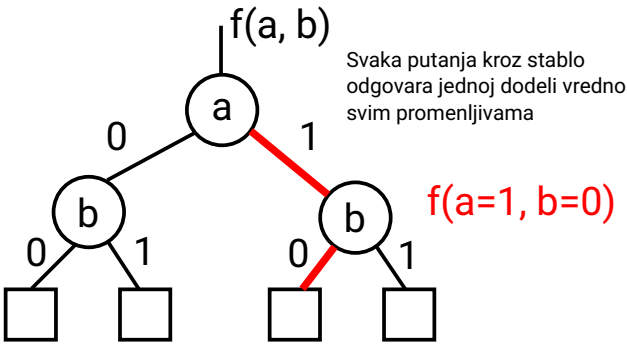
Šenonov razvoj je moguće prikazati grafički pomoću binarnog diagrama odluke:



Ovakav dijagram nije naročito koristan jer mu je veličina eksponencijalna spram broja promenljivih

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Šenonov razvoj je moguće prikazati grafički pomoću binarnog diagrama odluke:



No, moguće ga je redukovati (i to tokom prevođenja izraza)

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[12]

Randal E. Bryant[3]

IEEE Transactions on Computers '86

Abstract

In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[1,2]

Randal E. Bryant[3]

IEEE Transactions on Computers '86

Abstract

In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

"For many of the last five, ten years, his paper was cited more often than anything else in CiteSeer, according to CiteSeer. This was the number one referenced paper."

Donald Knuth, 2008
(*Stanford Lecture: Donald Knuth - "Fun With Binary Decision Diagrams (BDDs)*")

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[12]

Randal E. Bryant[3]

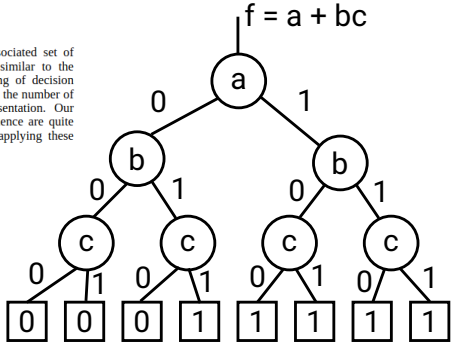
IEEE Transactions on Computers '86

Abstract

In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

"For many of the last five, ten years, his paper was cited more often than anything else in CiteSeer, according to CiteSeer. This was the number one referenced paper."

Donald Knuth, 2008
(*Stanford Lecture: Donald Knuth - "Fun With Binary Decision Diagrams (BDDs)"*)



Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[12]

Randal E. Bryant[3]

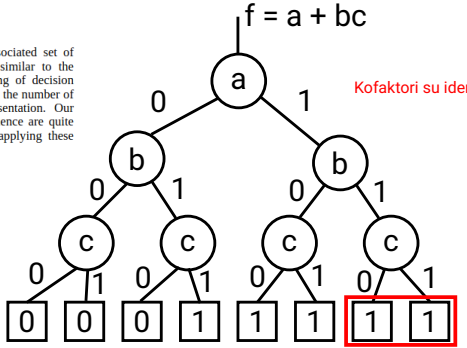
IEEE Transactions on Computers '86

Abstract

In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

"For many of the last five, ten years, his paper was cited more often than anything else in CiteSeer, according to CiteSeer. This was the number one referenced paper."

Donald Knuth, 2008
(*Stanford Lecture: Donald Knuth - "Fun With Binary Decision Diagrams (BDDs)"*)



Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[12]

Randal E. Bryant[3]

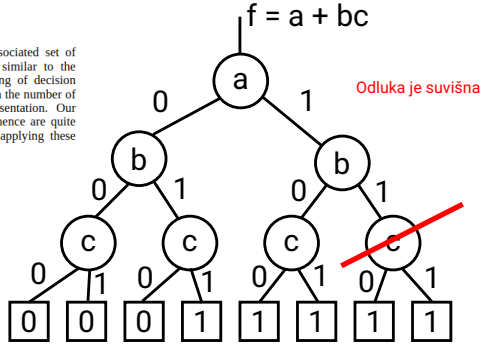
IEEE Transactions on Computers '86

Abstract

In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

"For many of the last five, ten years, his paper was cited more often than anything else in CiteSeer, according to CiteSeer. This was the number one referenced paper."

Donald Knuth, 2008
(*Stanford Lecture: Donald Knuth - "Fun With Binary Decision Diagrams (BDDs)"*)



Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[12]

Randal E. Bryant[3]

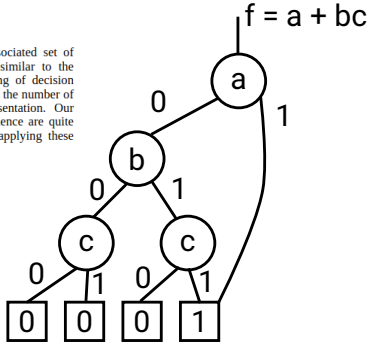
IEEE Transactions on Computers '86

Abstract

In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

"For many of the last five, ten years, his paper was cited more often than anything else in CiteSeer, according to CiteSeer. This was the number one referenced paper."

Donald Knuth, 2008
(*Stanford Lecture: Donald Knuth - "Fun With Binary Decision Diagrams (BDDs)"*)



Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[12]

Randal E. Bryant[3]

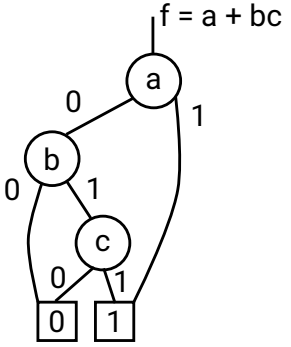
IEEE Transactions on Computers '86

Abstract

In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

"For many of the last five, ten years, his paper was cited more often than anything else in CiteSeer, according to CiteSeer. This was the number one referenced paper."

Donald Knuth, 2008
(*Stanford Lecture: Donald Knuth - "Fun With Binary Decision Diagrams (BDDs)"*)



Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[12]

Randal E. Bryant[3]

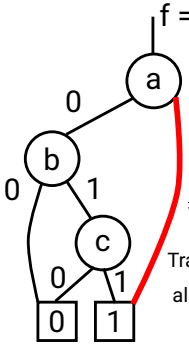
IEEE Transactions on Computers '86

Abstract

In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

"For many of the last five, ten years, his paper was cited more often than anything else in CiteSeer, according to CiteSeer. This was the number one referenced paper."

Donald Knuth, 2008
(*Stanford Lecture: Donald Knuth - "Fun With Binary Decision Diagrams (BDDs)"*)



Postoji putanja do 1 $\Longrightarrow$ funkcija je zadovoljiva

Traženje putanje je $\mathcal{O}(|V|)$, ali $|V|$ može biti i $2^{n+1} - 1$

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[12]

Randal E. Bryant[3]

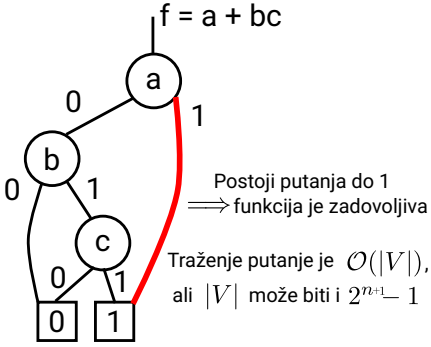
IEEE Transactions on Computers '86

Abstract

In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

"For many of the last five, ten years, his paper was cited more often than anything else in CiteSeer, according to CiteSeer. This was the number one referenced paper."

Donald Knuth, 2008
(*Stanford Lecture: Donald Knuth - "Fun With Binary Decision Diagrams (BDDs)"*)



Redukovani dijagrami su često vrlo mali za funkcije koje se sreću u praksi

Primer uticaja na druge oblasti: (RO)BDD (Reduced Ordered Binary Decision Diagram)

Graph-Based Algorithms for Boolean Function Manipulation[12]

Randal E. Bryant[3]

IEEE Transactions on Computers '86

Abstract

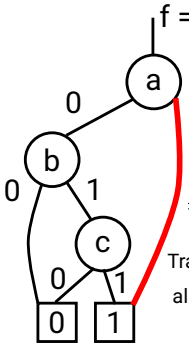
In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

"For many of the last five, ten years, his paper was cited more often than anything else in CiteSeer, according to CiteSeer. This was the number one referenced paper."

Donald Knuth, 2008
(*Stanford Lecture: Donald Knuth - "Fun With Binary Decision Diagrams (BDDs)"*)

Kanonska forma za dati poredak promenljivih

$$\implies f = g \iff \mathrm{ROBDD}(f) = \mathrm{ROBDD}(g)$$



Postoji putanja do 1 $\implies$ funkcija je zadovoljiva

Traženje putanje je $\mathcal{O}(|V|)$, ali $|V|$ može biti i $2^{n+1} - 1$

Redukovani dijagrami su često vrlo mali za funkcije koje se sreću u praksi

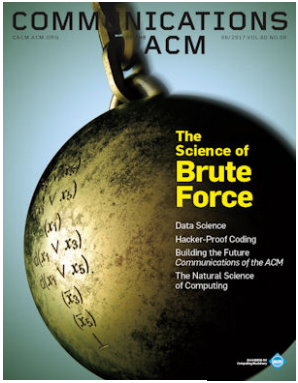
Primer uticaja na druge oblasti: CDCL SAT rešavači

SEARCH ALGORITHMS FOR SATISFIABILITY PROBLEMS
IN COMBINATIONAL SWITCHING CIRCUITS

by

João Paulo Marques da Silva

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering)
in The University of Michigan
1995



Chaff: Engineering an Efficient SAT Solver

Matthew W. Moskewicz
Department of EECS
UC Berkeley
moskewcz@alumni.princeton.edu

Conor F. Madigan
Department of EECS
MIT
cmadigan@mit.edu

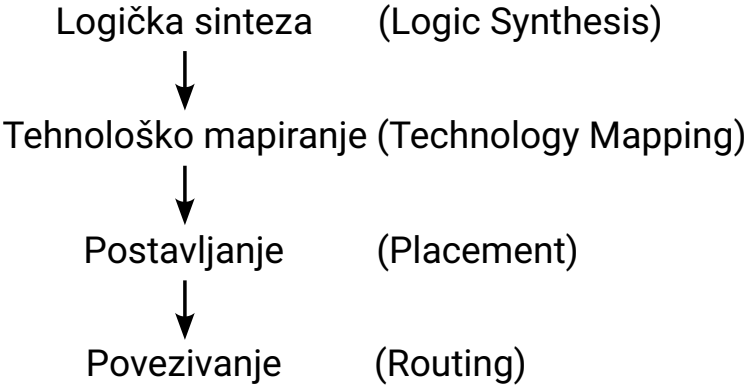
Ying Zhao, Lintao Zhang, Sharad Malik
Department of Electrical Engineering
Princeton University
{yingzhao, lintaoz, sharad}@ee.princeton.edu

ABSTRACT

Boolean Satisfiability is probably the most studied of combinatorial optimization/search problems. Significant effort has been devoted to trying to provide practical solutions to this

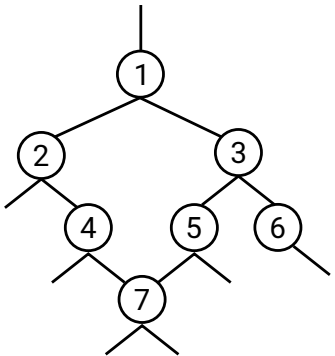
Many publicly available SAT solvers (e.g. GRASP [8], POSIT [5], SATO [13], rel_sat [2], WalkSAT [9]) have been developed, most employing some combination of two main strategies: the Davis-Putnam (DP) backtrack search and heuristic

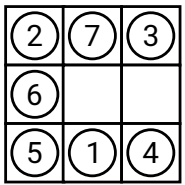
Koraci u generisanju lejauta



Postavljanje (Placement)

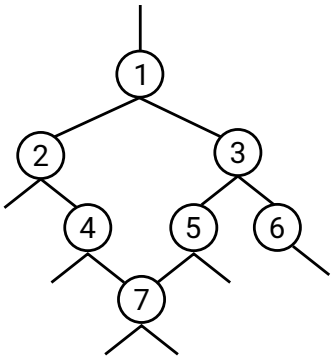
Zadatak: Dodeliti svakom čvoru koordinate u dve (tri) dimenzije, tako da se čvorovi ne preklapaju i da je neka funkcija dužina grana minimizovana

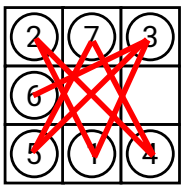




Postavljanje (Placement)

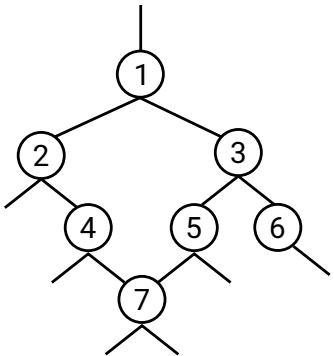
Zadatak: Dodeliti svakom čvoru koordinate u dve (tri) dimenzije, tako da se čvorovi ne preklapaju i da je neka funkcija dužina grana minimizovana

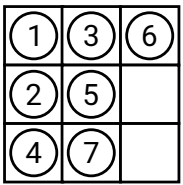




Postavljanje (Placement)

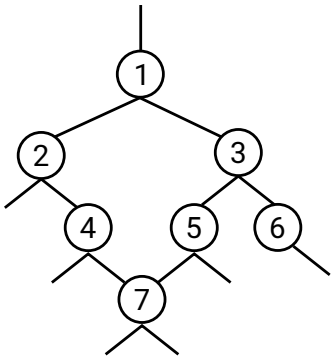
Zadatak: Dodeliti svakom čvoru koordinate u dve (tri) dimenzije, tako da se čvorovi ne preklapaju i da je neka funkcija dužina grana minimizovana

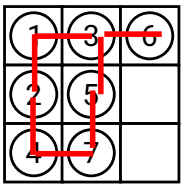




Postavljanje (Placement)

Zadatak: Dodeliti svakom čvoru koordinate u dve (tri) dimenzije, tako da se čvorovi ne preklapaju i da je neka funkcija dužina grana minimizovana





Postavljanje (Placement)

Nekoliko najznačajnijih pristupa:

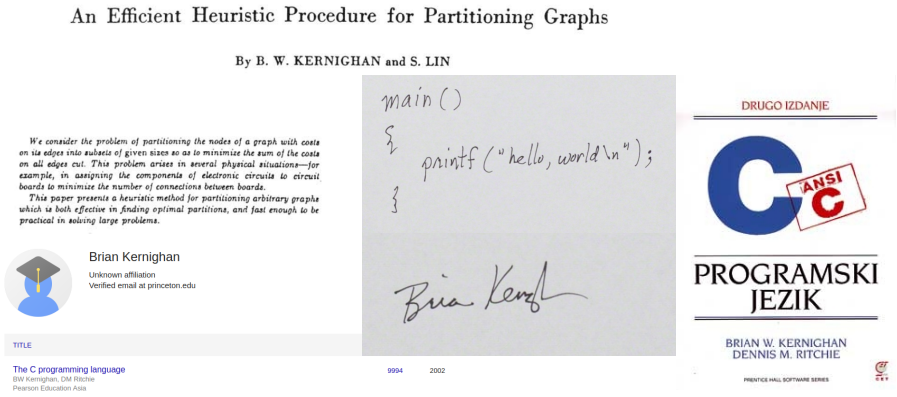
- Rekurzivno biparticionisanje (Recursive Bipartitioning)
- Simulirano kaljenje (Simulated Annealing)
  (metaheuristika nastala baš za rešavanje ovog problema [Kirkpatrick et al., '83])
- Analitički algoritmi

Rekurzivno biparticionisanje

Zadatak: Podeliti skup čvorova u dva podskupa približno jednake veličine, tako da je broj grana čiji su incidentni čvorovi u različitim podskupovima minimizovan.

Rekurzivno biparticionisanje

Zadatak: Podeliti skup čvorova u dva podskupa približno jednake veličine, tako da je broj grana čiji su incidentni čvorovi u različitim podskupovima minimizovan.



An Efficient Heuristic Procedure for Partitioning Graphs

By B. W. KERNIGHAN and S. LIN

We consider the problem of partitioning the nodes of a graph with costs on its edges into subsets of given sizes so as to minimize the sum of the costs on all edges cut. This problem arises in several physical situations—for example, in assigning the components of electronic circuits to circuit boards to minimize the number of connections between boards.

This paper presents a heuristic method for partitioning arbitrary graphs which is both effective in finding optimal partitions, and fast enough to be practical in solving large problems.

Brian Kernighan

Unknown affiliation

Verified email at princeton.edu

```
main()
{
    printf("hello, world\n");
}
```

Brian Kernighan

DRUGO IZDANJE

C ANSI C

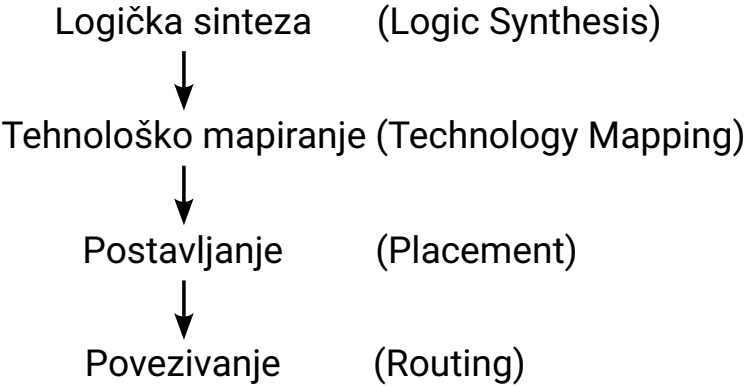
PROGRAMSKI JEZIK

BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

PRENTICE HALL SOFTWARE SERIES

TITLE		
The C programming language BW Kernighan, DM Ritchie Pearson Education Asia	9994	2002
An efficient heuristic procedure for partitioning graphs BW Kernighan, S Lin The Bell system technical journal 49 (2), 291-307	7531	1970

Koraci u generisanju lejauta



Povezivanje

Zadatak: Povezati sve parove čvorova između kojih postoji grana, što kraćom putanjom sačinjenom od horizontalnih i vertikalnih segmenata metala, tako da broj različitih putanja koje se seku u bilo kojoj tački ne prevazilazi broj raspoloživih slojeva metala.

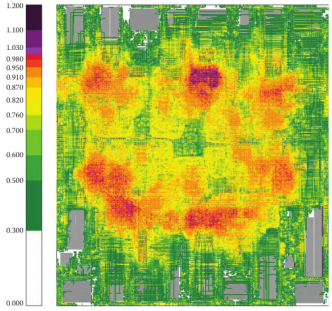


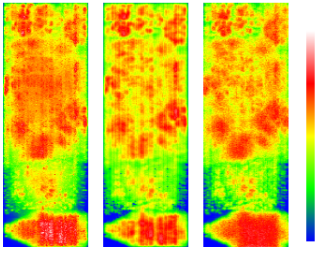
FIGURE 8.10 A congestion map plotting $\eta(e)$ (color scale on left) over a chip floorplan. Gray areas denote macro blocks within the floorplan.

L. Lavagno, G. Martin, I. Markov, L. Scheffer (eds.),
"Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology",
CRC Press, 2016

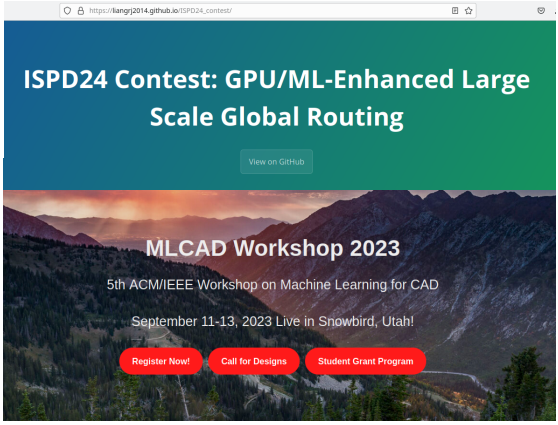
Primena mašinskog učenja

Machine-Learning Based Congestion Estimation for Modern FPGAs

D. Maarouf, A. Alhyari, Z. Abuowaimer, T. Martin, A. Gunter, G. Grewal, S. Areibi, A. Vannelli
School of Engineering/School of Computer Science, University of Guelph
Guelph, Ontario, Canada
{dmaarouf,aalhyari,abuowaiz,tmarti14,agunter,ggrewal,sareibi}@uoguelph.ca, Tony.vannelli@usask.ca



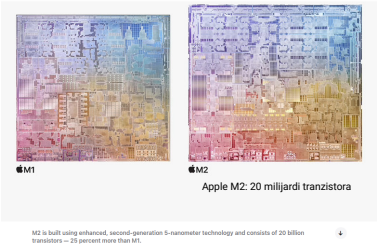
(e) mPFGR (f) G_{M16} (g) Vivado Detailed Router



Neki dodatni aspekti

- Alati za razvoj hardvera su veoma složeni
- Performanse su od izuzetnog značaja
  (generisanje lejauta za najveće čipove može da traje danima)

OpenROAD: jedan od najvećih projekata otvorenog koda.
Započet je 2018, a danas sadrži oko pola miliona linija koda.



Nevertheless, a design database is the heart of the system. While it is possible to build a bad EDA tool or flow on any database, it is impossible to build a good EDA tool or flow on a bad database. Mark Bales (Synopsys Fellow)

Zaključci

Zaključci

- Usporavanje Murovog zakona dovelo je do velike potrebe za specijalizovanim hardverom

Zaključci

- Usporavanje Murovog zakona dovelo je do velike potrebe za specijalizovanim hardverom
- Razvoj hardvera odavno liči na razvoj softvera (nije više bauk), ali je zadržao neke specifičnosti (potrebno je specifično znanje)

Zaključci

- Usporavanje Murovog zakona dovelo je do velike potrebe za specijalizovanim hardverom
- Razvoj hardvera odavno liči na razvoj softvera (nije više bauk), ali je zadržao neke specifičnosti (potrebno je specifično znanje)
- Prodorom velikih softverskih kompanija u razvoj hardvera, sličnosti se dodatno uvećavaju

Velike softverske kompanije uveliko razvijaju sopstveni specijalizovani hardver

Microsoft

Google

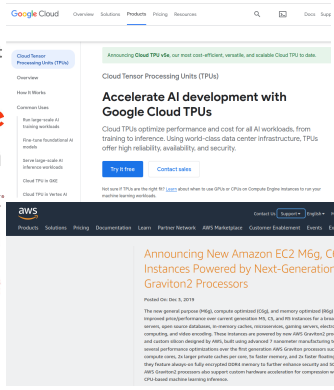
amazon

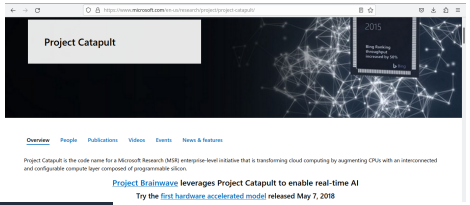
ORACLE

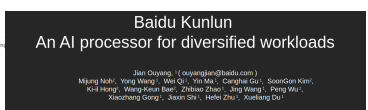
Meta

Baidu 百度

Alibaba Group







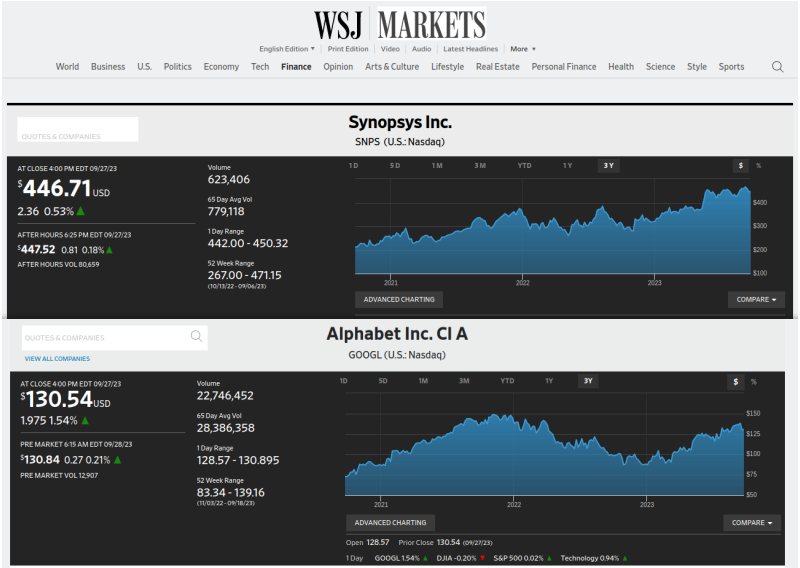
Zaključci

- Usporavanje Murovog zakona dovelo je do velike potrebe za specijalizovanim hardverom
- Razvoj hardvera odavno liči na razvoj softvera (nije više bauk), ali je zadržao neke specifičnosti (potrebno je specifično znanje)
- Prodorom velikih softverskih kompanija u razvoj hardvera, sličnosti se dodatno uvećavaju
- Alati za automatizaciju razvoja hardvera su plodno tle za istraživanje u domenu teorije grafova, programskih jezika, kompajlera, algoritama, baza podataka...

Zaključci

- Usporavanje Murovog zakona dovelo je do velike potrebe za specijalizovanim hardverom
- Razvoj hardvera odavno liči na razvoj softvera (nije više bauk), ali je zadržao neke specifičnosti (potrebno je specifično znanje)
- Prodorom velikih softverskih kompanija u razvoj hardvera, sličnosti se dodatno uvećavaju
- Alati za automatizaciju razvoja hardvera su plodno tle za istraživanje u domenu teorije grafova, programskih jezika, kompajlera, algoritama, baza podataka...
- Povećana potreba za specijalizovanim hardverom povećava i praktičnu potrebu za razvojem takvih alata

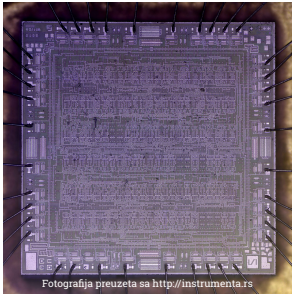
Vrednost akcija najveće kompanije za razvoj EDA alata, Synopys



Nešto malo bliže nama

Neke preteče FPGA čipova su 1980ih razvijane i proizvođene i kod nas

MPGA BMD21 (1987)




БОРБА

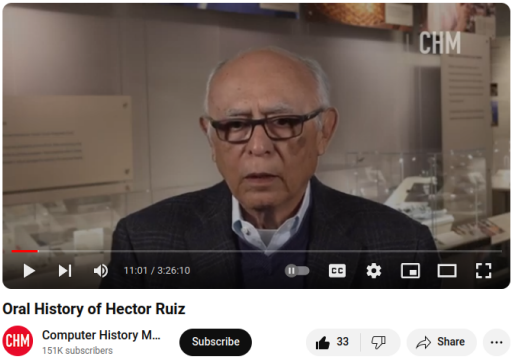
КОМИСИЈА ЗА СТРУЧНО ОБРАЗОВАЊЕ КАДРОВА
РАДНИЧКОГ САВЕТА
ЕЛЕКТРОНСКЕ ИНДУСТРИЈЕ - НИШ
ИЗВЕШТАЈ
О РЕЗУЛТАТУ КОНКУРСА ЗА ДОДЕЛУ СТИПЕНДИЈА ЗА ШКОЛСКУ 1965/66 ГОДИНУ

ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ МАШИНСКИ ФАКУЛТЕТ

129. ЛИТОВСКИ Бориса ВАНЧО
130. МАЛЧИЋ Андрије МИРЈАНА
131. МАНДИЋ Вељка НАДА
132. МАНДИЋ Радивоја МАРИЈА

1203) Litovski, V., Petković, P. **Elektricna karakterizacija celija u gejtovskoj matrici GEM 21** Zbornik radova XII Jugoslovenskog savetovanja o mikroelektronici, MIEL 84, Niš 7-9. maj 1984 pp. 209-214

Intervju sa drugim CEO-om AMD-a



https://www.youtube.com/watch?v=9YzMJi5T3A8

CMOS kod nas

Proizvodnja CMOS integrisanih kola: 1981.

- Za proizvodnju integrisanih kola odabrana je CMOS tehnologija.
- Licencirana je tehnologija kompanije RCA.
- Deo fabrike je u potpunosti adaptiran za novu proizvodnju.
- Nabavljena je najsavremenija proizvodna oprema.
- Proizvodnja digitalnih CMOS integrisanih kola sa aluminijumskim gejtom serije CD4000.
- Proizvodnja visokopouzdanih i komercijalnih integrisanih kola.
- Planarna tehnologija, na pločicama prečnika 3 inča.

Ostvarenju ovog projekta posebno je doprineo prof. dr Radivoje Popović.

Organizacija predmeta

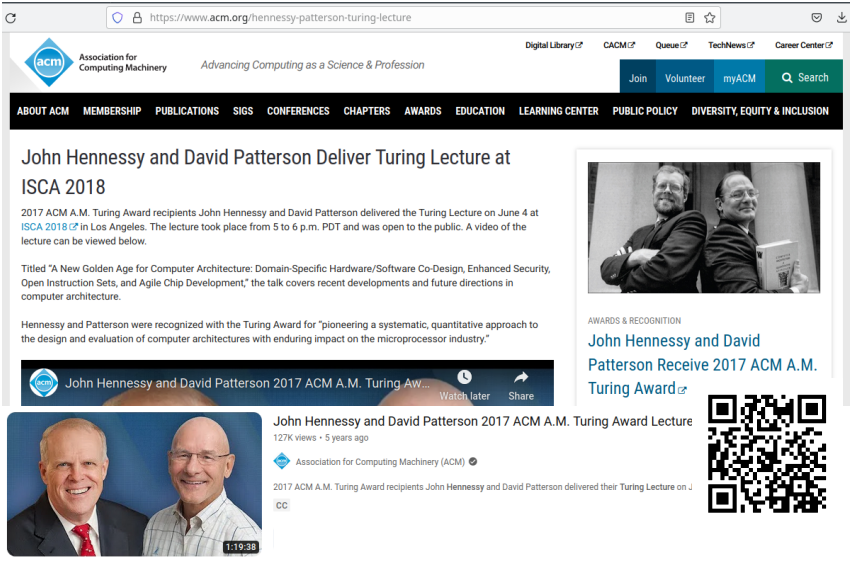
Čitalačka grupa + istraživački projekti

- Dva bloka predavanja za upoznavanje sa osnovnim pojmovima
- Deset naučnih radova (svi čitamo sve, svako prezentuje po dva)
- Recenzije (zapravo, hotCRP, ISFPGA)
- Dva mala domaća (jedan iz sinteze i jedan iz postavljanja)
- Jedan projekat (preporučeno iz rutiranja, postoji spisak tema, ali je moguće i predložiti neku drugu)
- Završna prezentacija i izveštaj

Nakon što završimo sa predavanjima, sve će funkcionisati po principu prezentacija i diskusija zadatih radova, te istraživačkih sastanaka

Domaći zadatak

Pogledati sledeće predavanje



Hvala na pažnji

“Yes, there’s an end to scaling. But there’s no end to creativity.”

Robert Dennard
The inventor of DRAM