

# Postavljanje i povezivanje

---

Stefan Nikolić

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

13.11.2024.

## Podsetnik sa prošlog časa

Nakon tehnološkog mapiranja, dobili smo graf u kom svaki čvor predstavlja jednu lukap tabelu (eng. **LOOK-UP TABLE** (LUT))

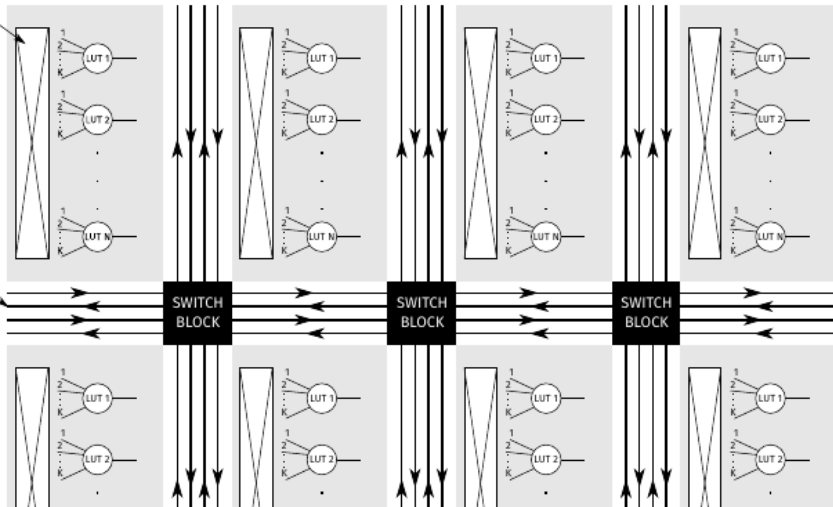
U osnovi, FPGA čip je velika kolekcija LUTova koje je moguće programabilno povezati

Zadatak poslednje dve faze dizajna (eng. **PHYSICAL DESIGN**) je da svakom od čvorova grafa kola odrede fizičku lokaciju na čipu (postavljanje, eng. **PLACEMENT**) i da implementiraju svaku granu grafa pomoću putanje sačinjene od prefabrikovanih žica i programabilnih prekidača (povezivanje, eng. **ROUTING**)

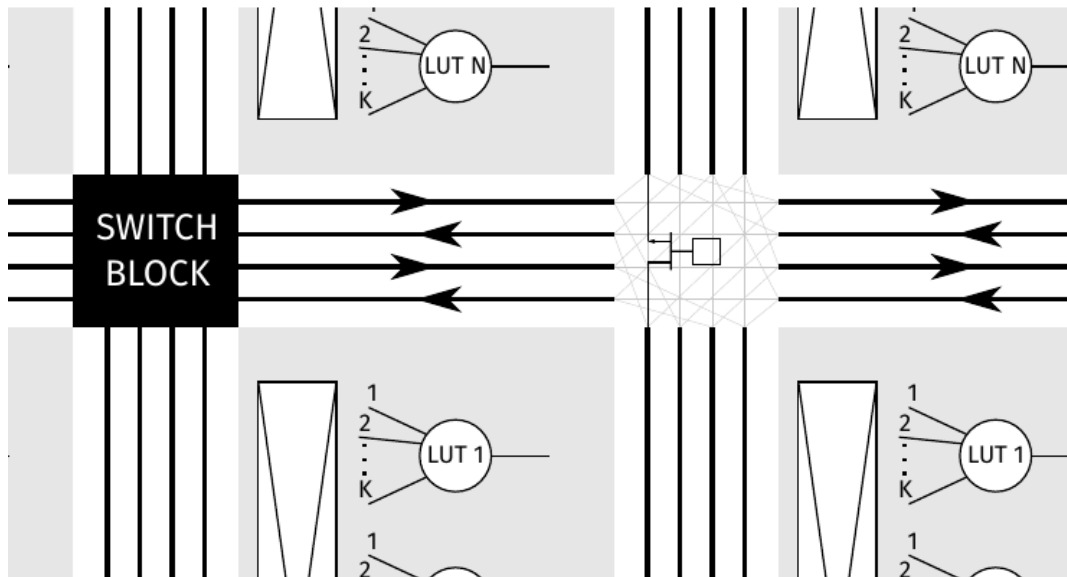
# FPGA arhitektura u kratkim crtama

local interconnect  
(roughly approximates a clique)

routing channel



# FPGA arhitektura u kratkim crtama



## Pakovanje (eng. PACKING/CLUSTERING)

Pošto svi savremeni FPGA čipovi sadrže klastere, u nekim slučajevima pre postavljanja vršimo klasterizaciju (pakovanje) čvorova grafa kola u klastere koje može da implementira fizički klaster datog FPGA čipa

## Dve vrste pakovanja

1. Partitionisanjem (top-down): rekurzivno delimo graf dok ne dobijemo particije veličine klastera koji postoji na FPGA čipu
2. Zasejavanjem (eng. **SEED-BASED, BOTTOM-UP**): započnemo novi klaster čvorom  $u$ , a zatim ga popunjavamo čvorovima koji imaju najviše zajedničkih veza sa čvorovima koji su već u datom klasteru, sve dok ne popunimo klaster

## Dve vrste pakovanja

Prednost prvog pristupa je što može da sagleda čitav graf odjednom, a prednost drugog pristupa je veća fleksibilnost (možemo da formulišemo različite cene dodavanja čvorova, kao i da proverimo veoma složene uslove koje klaster mora da zadovolji da bi bio legalan)

Danas se eksplicitno pakovanje uglavnom ne radi, već se arhitekture dizajniraju na taj način da su ograničenja u pogledu legalnosti klastera vrlo mala, pa je pakovanje moguće preskočiti

Ipak, iz edukativnih razloga, čitaćemo dva rada o pakovanju



# Postavljanje

---

# Postavljanje

Zadatak postavljanja je da svaki čvor bude mapiran na fizički blok na čipu (ili na lokaciju u slučaju ASIC-a)

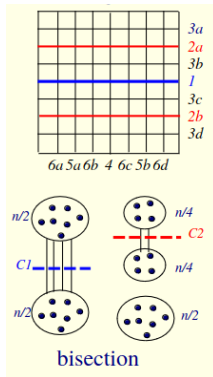
Pritom ne sme biti preklapanja, a cilj nam je uglavnom da smanjimo dužine razdaljina među granama koje je potrebno realizovati

U novije doba je cilj i direktna minimizacija predviđenog zagušenja

# Tri osnovna pristupa

1. Min-cut
2. simulirano kaljenje
3. analitički algoritmi

# Min-cut



©Sung Kyu Lim

Prednosti: velika brzina

Mane: teško je modelovati tajming; velike varijacije u rezultatima

# Odlični algoritmi za particionisanje grafova

`https://karypis.github.io/glaros/software/metis/overview.html`

# Simulirano kaljenje

```
S = RandomPlacement ();
T = InitialTemperature ();
Rlimit = InitialRlimit ();

while (ExitCriterion () == False) {      /* “Outer loop” */
    while (InnerLoopCriterion () == False) { /* “Inner loop” */
        Snew = GenerateViaMove (S, Rlimit);
        ΔC = Cost (Snew) - Cost (S);
        r = random (0,1);
        if (r < e-ΔC/T) {
            S = Snew;
        }
    } /* End “inner loop” */
    T = UpdateTemp ();
    Rlimit = UpdateRlimit ();
} /* End “outer loop” */
```

# Simulirano kaljenje

Prednosti: jako velika fleksibilnost (opšta metaheuristika)

Mane: veoma sporo za velika kola

Ipak, još uvek je veoma zastupljen pristup u istraživanju, a do skoro i u industriji; mi ćemo ga takođe koristiti



## HeAP: Heterogeneous Analytical Placement for FPGAs

FPL 2012

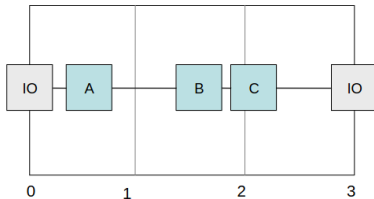
Marcel Gort and Jason Anderson



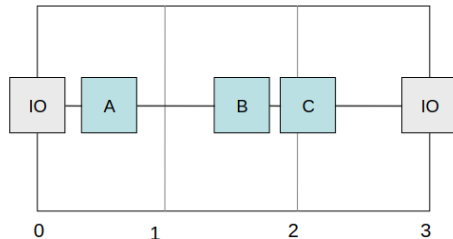
## Analytical Placement (AP)

- Objective function: Half-Perimeter Wirelength (HPWL)
- Minimizing objective function:
  - Solve system of linear equations generated from connections between cells.
  - 1. Convert multi-pin nets to 2-pin nets.
  - 2. Create system of linear equations to solve **weighted sum of squared distances between cells**.
  - 3. Solve system using off-the-shelf linear systems solver.

# Analitički algoritmi



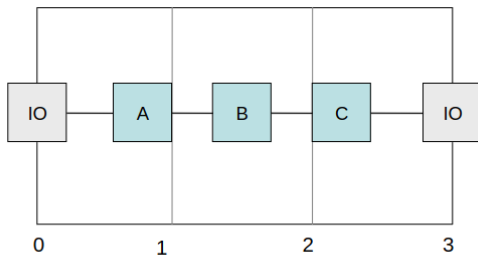
$$\Phi = \text{Min} [(X_B - X_A)^2 + (X_C - X_B)^2 + (X_A - 0)^2 + (3 - X_C)^2]$$



$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \times \begin{bmatrix} X_A \\ X_B \\ X_C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}$$

$$X_A = 0.75, X_B = 1.5, X_C = 2.25$$

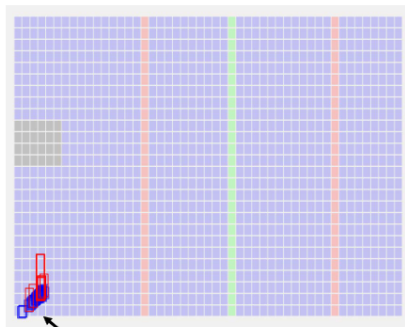
# Analitički algoritmi



$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \times \begin{bmatrix} X_A \\ X_B \\ X_C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}$$

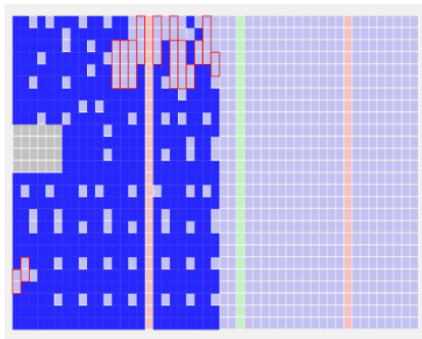
$$X_A = 0.75, X_B = 1.5, X_C = 2.25$$

## Actual Solved Solution



Need to spread cells!

## Legalized Solution



Kako da izbegnemo preklapanja?



## Spreading

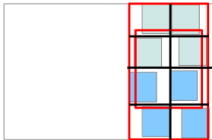
- Adapted from SimPL [2].
- Find over-utilized area.
- Find a larger surrounding area that can accommodate all cells within it.
- Split the cells into two sets.



13

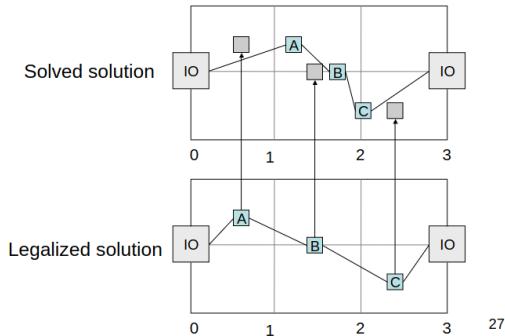
## Spreading

- Assign an area to each cell which is proportional to the total area of the cells.
- Spread each set of cells separately, within the area assigned to it.
- Alternate x and y spreading directions until solution is legal.

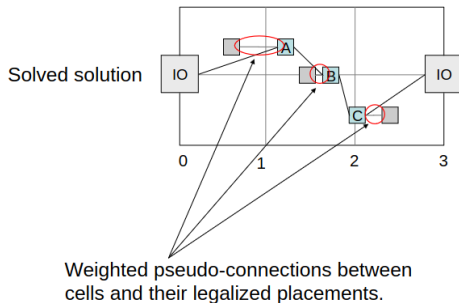


17

## Pseudo-connections



## Pseudo-connections



# Opšti recept

Kada nas neka ograničenja legalnosti sprečavaju da efikasno rešimo problem:

1. Privremeno zaboravimo na ta ograničenja i rešimo uprošćen problem
2. Legalizujemo dobijeno rešenje (time odbacujemo optimalnost rešenja uprošćenog problema)
3. Iz legalizovanog rešenja naučimo nešto što će rešenje uprošćenog problema privesti bliže poznatom legalnom rešenju
4. Iteriramo dok ne konvergiramo

## RePLace: Advancing Solution Quality and Routability Validation in Global Placement

Chung-Kuan Cheng, *Fellow, IEEE*, Andrew B. Kahng, *Fellow, IEEE*,  
Ilgweon Kang, *Member, IEEE*, and Lutong Wang, *Student Member, IEEE*

## elfPlace: Electrostatics-based Placement for Large-Scale Heterogeneous FPGAs

Wuxi Li, Yibo Lin, and David Z. Pan  
ECE Department, University of Texas at Austin, Austin, Texas, USA  
{wuxi.li, yibolin}@utexas.edu; dpan@ece.utexas.edu

## **DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement**

Yibo Lin

ECE Department, UT Austin  
yibolin@utexas.edu

Haoxing Ren

Nvidia, Inc., Austin  
haoxingr@nvidia.com

Shounak Dhar

ECE Department, UT Austin  
shounak.dhar@utexas.edu

Brucek Khailany

Nvidia, Inc., Austin  
bkhailany@nvidia.com

Wuxi Li

ECE Department, UT Austin  
wuxi.li@utexas.edu

David Z. Pan

ECE Department, UT Austin  
dpan@ece.utexas.edu

# Neki drugi pristupi

1. ILP, SAT, SMT: odlični za lokalna unapređenja; loše se skaliraju
2. razne metode mašinskog učenja: obećavaju kao pomoć klasičnim algortmima; samostalno se još nisu pokazali



← → ↻ <https://www.nature.com/articles/y41586-021-03544-w>

nature

Explore content ▾ About the journal ▾ Publish with us ▾ Subscribe

nature > articles > article

Article | Published: 09 June 2021

## A graph placement methodology for fast chip design

Azalia Mirhoseini , Anna Goldie , Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azadeh Nazki, Jiwon Pak, Andy Tong, Karya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter & Jeff Dean

Nature 594, 207–212 (2021) | [Cite this article](#)

55k Accesses | 189 Citations | 2094 Altmetric | [Metrics](#)

**20 September 2023** Editor's Note: Readers are alerted that the performance claims in this article have been called into question. The Editors are investigating these concerns, and, if appropriate, editorial action will be taken once this investigation is complete.

**1** An [Author Correction](#) to this article was published on 31 March 2022

## Stronger Baselines for Evaluating Deep Reinforcement Learning in Chip Placement

### Abstract

Deep Reinforcement Learning (DRL) has demonstrated stunning success in game-playing and several applied optimization problems. Improving chip designs is an attractive next application, as illustrated by a recently proposed DRL technique for chip layout optimization based on EdgesGNNs. To evaluate this technique, we develop stronger baselines by enhancing established techniques. Compared to the DRL implementation reported earlier, our stronger baselines produce consistently superior design layouts both on the chip designs used in the DRL paper and on public benchmarks. We also produce competitive layouts with computational resources smaller by orders of magnitude. Ablation studies help explain these wins and point to weaknesses in how DRL is applied. Furthermore, our stronger baselines and our empirical results suggest that the work of human chip designers cannot serve as a strong baseline in scientific settings.

Bloomberg

Live Now Markets Economics Industries Tech AI Politics Wealth Pursuits Opinion Businessweek Equality Green

US Edition

Technology

## Fired Google AI Engineer's Whistleblower Lawsuit Moves Ahead

- Judge tentatively rejects company's request to toss complaint
- Satrajit Chatterjee claims research paper overhyped technology



## Neka korisna predavanja

# Neka korisna predavanja

→ ↺ 🔒 https://www.youtube.com/watch?v=rAJ7-KXlpuQ&t=3487s

YouTube RS cass rio grande patrick

IEEE CASS RS Talks 2023  
UFGRS, Porto Alegre, Brazil  
1:30 PM (Brasilia Time, GMT-3), August 4, 2023

YouTube Live @ IEEE CASS Rio Grande do Sul Chapter  
<https://www.youtube.com/cassriograndesul>

UFGRS  
PGMICRO  
CAS

Dr. Patrick Groeneveld, Stanford University, USA  
**What Drives the Enduring Success of Electronic Design Automation?**

**Abstract:**  
Today's chips have reached a truly astronomical scale, with some boasting more components than there are stars in the Milky Way or neurons in the human brain. Designing with such extreme complexity as a chip the size of a fingernail remains feasible, thanks to successful Electronic Design Automation tools. In this presentation, we will explore the methodologies and algorithms that enable design at an ever-increasing scale, and discuss the challenges that remain open. IC design is a process that consists of hundreds of algorithmic steps, gradually transforming a functional description into the layout of an operational circuit. The appropriate placement and interconnection of billions of components on the IC is crucial for controlling cost, performance, reliability, and power consumption. We will delve into how these conflicting objectives are optimally managed and demonstrate how this approach meets the extreme computing requirements of Machine Learning.

**Short CV:**  
Patrick is a lecturer in the Department of Electrical Engineering at Stanford University. He has a long career in Electronic Design Automation, working at both Cadence and Synopsys. He was Chief Technologist at Pegasus Design Automation where he was part of the team that developed a groundbreaking RTL-to-GDSII synthesis product. Patrick also worked at AI hardware startups and was a Full Professor of Electrical Engineering at Eindhoven University. He serves as Branch chair in the Executive Committee of the Design Automation Conference. Patrick received his MSc and PhD degrees from Delft University of Technology in the Netherlands.

Play (k)

0:44 / 1:06:53

UFGRS INF ufrgs CAS IEEE CAS

CASS Talks 2023 - Patrick Groeneveld, Stanford University, USA - August 4, 2023

IEEE CASS Rio Grande do...  
2.76K subscribers

Subscribed

28

Share

Clip

# Neka korisna predavanja

https://www.ispd.cc/slides/2021/protected/3\_1\_Kahng.pdf


— + Automatic Zoom ▾

## Advancing Placement

ISPD-2021  
March 22, 2021

---

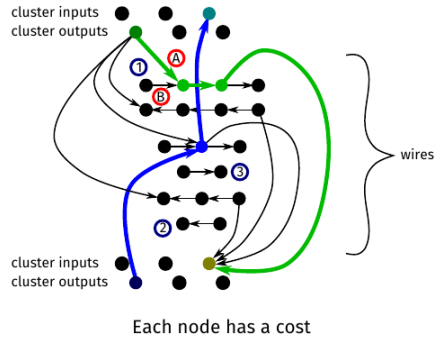
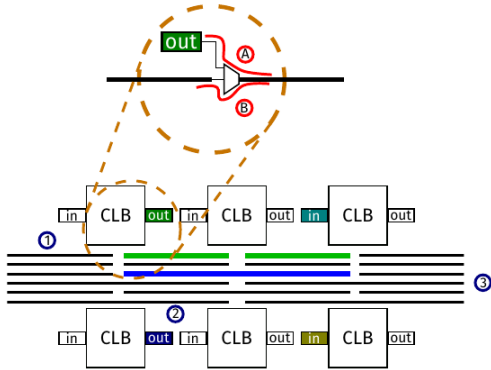
Andrew B. Kahng  
Depts. of CSE and ECE  
UC San Diego  
[abk@ucsd.edu](mailto:abk@ucsd.edu)  
<https://vlsicad.ucsd.edu>

 Andrew B. Kahng, 210322 ISPD-21 Advancing Placement

# Rutiranje

---

# RR-graf



---

**Algorithm 5.1** Simplified PathFinder [McM95; Bet99]
 

---

**Input:**  $G = (V, E)$ —rr-graph,  $E_c \subseteq V \times V$ —all connections to be routed;

**Output:** A routing tree of each signal

```

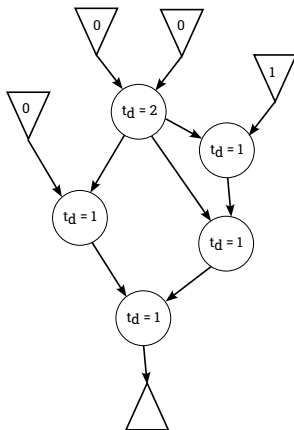
1: function CONGESTION_COST( $u, s$ )  $\triangleright$  computes the congestion cost of node  $u$  when routing signal  $s$ 
2:   if  $u \in RT(s)$  then return 0  $\triangleright$  if  $u$  is already used by one connection of  $s$ , others can freely use it
3:   return  $b(u) \times (1 + p_{fac} \times O(u)) \times (1 + h_{fac} \times C_h(u))$   $\triangleright$  otherwise, compute congestion
4: for  $u \in V$  do
5:    $O(u) = 0; C_h(u) = 0$   $\triangleright$  set occupancy and historical congestion of all nodes to 0
6:   if  $(\exists v \in V) ((u, v) \in E_c)$  then
7:      $RT(u) = \{u\}$   $\triangleright$  initialize the routing tree of each signal
8:    $i = 0; p_{fac} = p_{fac}^{init}$ 
9:   do
10:    if  $i \geq \text{max\_iter}$  then return UNROUTABLE
11:     $\triangleright$  no congestion-free routing was found in max_iter iterations
12:    for  $s \in \{u \in V : (\exists v \in V) ((u, v) \in E_c)\}$  do
13:       $\triangleright$  all signals are ripped up and rerouted in each iteration; modern incremental routers deviate from this [Mur20]
14:      for  $u \in RT(s)$  do
15:         $O(u) = O(u) - 1$   $\triangleright$  reduce the occupancy of all nodes used by the signal  $s$  that is ripped up
16:       $RT(s) = \{s\}; O(s) = O(s) + 1$   $\triangleright$  rip up the signal
17:      for  $t \in V : (s, t) \in E_c$  do
18:         $P = \text{SHORTEST\_PATH}(s, t, \forall u \in V : \text{cong}(u) = \text{CONGESTION\_COST}(u, s))$ 
19:         $\triangleright$  (re)route the connection  $s \rightarrow t$ 
20:      for  $u \in P$  do
21:        if  $\neg(u \in RT(s))$  then
22:           $O(u) = O(u) + 1$   $\triangleright$  increase the occupancy of all nodes not already used by the signal  $s$ 
23:         $RT(s) = RT(s) \cup P$   $\triangleright$  add the connection route to the routing tree of  $s$ 
24:      for  $u \in V$  do
25:         $C_h(u) = C_h(u) + \max(0, O(u) - 1)$   $\triangleright$  update historical congestion
26:       $i = i + 1$ 
27:       $p_{fac} = p_{fac} \times p_{fac}^{mult}$   $\triangleright$  increase the penalty of using occupied nodes;  $p_{fac}^{mult} > 1$ 
28:    while  $\exists u \in V : O(u) > 1$   $\triangleright$  finish if there is no congestion
29:  return  $\forall RT$   $\triangleright$  return all routing trees
    
```

---

Da bi naše kolo radilo na odgovarajućoj frekvenciji, moramo optimizovati kašnjenja



U čvorovima su označena njihova kašnjenja u odnosu na aktivnu ivicu takta

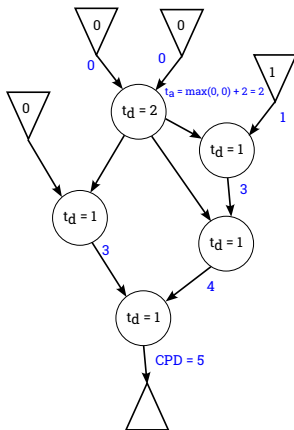


## Vreme dolaska (eng. ARRIVAL TIME)

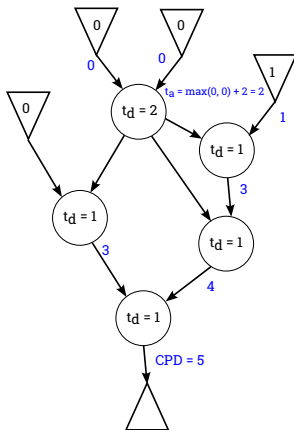
Najpre je potrebno da odgovorimo na pitanje kada će najranije izlaz čvora  $v$  biti dostupan

Za to uvodimo promenljivu  $t_a(v) = \max_{u:(u,v) \in E} t_a(u) + t_d(v)$

## U našem primeru



$\max_{u \in V} t_a(u)$  nazivamo kašnjenjem kritične putanje (eng. CRITICAL PATH DELAY (CPD))



Naš cilj je da CPD ne prekorači neku vrednost

Ona može biti unapred zadata ili je možemo izračunati, obilaskom grafa u topološkom poretku i izračunavanjem najmanjeg vremena dolaska za svaki čvor u kolu

Ovaj način zakazivanja (eng. **SCHEDULING**) nazivamo ASAP (**AS SOON AS POSSIBLE**)

Da bi vrednost CPD-a bila zadovoljena, ne moraju svi čvorovi da proizvedu izlaz najranije što je moguće

Da ne bi došlo do kašnjenja, čvorovi vezani za izlaze moraju obezbediti izlaz tačno u trenutku CPD ili ranije

Njihovo neophodno vreme (eng. **REQUIRED TIME**) je CPD, jer ako bi zakasnili, čitavo kolo bi zakasnilo

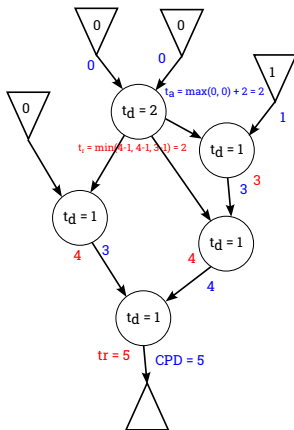
Zakazivanje čvorova u najkasnijem dozvoljenom trenutku nazivamo ALAP (**AS LATE AS POSSIBLE**)

## Neophodno vreme (eng. REQUIRED TIME)

Neophodno vreme računamo u obrnutom topološkom poretku, na sledeći način:

$$t_r(u) = \min_{v:(u,v) \in E} (t_r(v) - t_d(v))$$

## U našem primeru



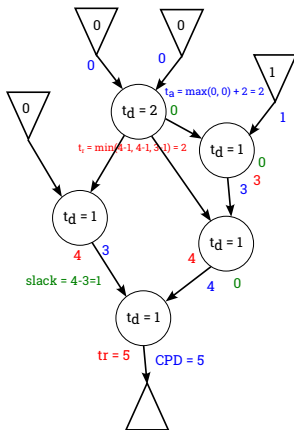


## Sloboda kašnjenja (eng. SLACK)

Ukoliko svaki čvor  $u$  proizvede svoj izlaz bilo gde u rasponu od  $t_a(u)$  i  $t_r(u)$ , kolo neće biti zakašnjeno

Slobodu kašnjenja čvora  $u$  računamo kao  $slack(u) = t_r(u) - t_a(u)$

## U našem primeru



Slekovi opisuju koliko bismo neki čvor mogli da zakasnimo a da ne narušimo uslove tajminga

Međutim, čim zakasnimo neki čvor, vrednosti slekova drugih čvorova se menjaju i moramo ih preračunati

# Kašnjenje veza

U savremenim kolima, veze imaju veće kašnjenje od tranzistora, pa ih je potrebno modelovati i optimizovati

Modelujemo ih pridruživanjem kašnjenja granama:  $t_d(u, v)$

## Kašnjenje veza

Tada vremena dolaska računamo uzimajući u obzir i kašnjenja grana:

$$t_a(v) = \max_{u:(u,v) \in E} (t_a(u) + t_d(u, v)) + t_d(v)$$

Slično je i sa neophodnim vremenima:

$$t_r(u) = \min_{v:(u,v) \in E} (t_r(v) - t_d(v) - t_d(u, v))$$

# Slekovi grana

Slek grane možemo izračunati kao

$slack(u, v) = t_r(v) - t_d(v) - t_a(u) - t_d(u, v)$  i on predstavlja meru dodatnog kašnjenja grane  $(u, v)$  koje možemo uneti tako da ne zakasnimo čitavo kolo

$t_r(v) - t_d(v)$  nam govori kada najkasnije signal iz čvora  $u$  mora da stigne do čvora  $v$ ,  $t_a(u)$  nam govori kada će najranije stići na početak veze, a  $t_d(u, v)$  nam govori koliko je vremena potrebno za prostiranje kroz samu vezu

Veze koje imaju mali slek su blizu toga da određuju kritičnu putanju kola, pa o njima moramo posebno da vodimo računa

Ako normalizujemo sve slekove kašnjenjem kritične putanje, možemo da uvedemo meru koju nazivamo **KRITIČNOST**:

$$crit(u, v) = 1 - \frac{slack(u, v)}{CPD}$$

# Optimizacija kašnjenja veza

Svaka žica na FPGA čipu (svaki čvor RRG-a) ima neko kašnjenje  $t(n)$

Cilj nam je da obezbedimo da kritične veze biraju čvorove RRG-a prevashodno po kašnjenju, a tek potom po zagušenosti, dok bi one koje su manje kritične trebalo da primoramo da brze čvorove ustupe kritičnijim vezama



# Optimizacija kašnjenja veza

To činimo na sledeći način: Pri rutiranju grane kola  $(u, v)$ ,  
 $cost(n) = crit(u, v) \times t(n) + (1 - crit(u, v)) \times cong(n)$ , gde je  $cong(n)$  cena zagušenja koju smo ranije videli

Za kritične veze, drugi deo cene će biti blizak nuli i dominiraće kašnjenje, dok će za manje kritične veze situacija biti obrnuta, što nam je i bio cilj

Kako neki signali ne bi u potpunosti ignorisali zagušenje, maksimalnu kritičnost obično ograničavamo na neku vrednost malo manju od 1 (na primer 0,999)

# Domaći zadatak

---

# Pročitati i recenzirati ova dva rada

← → ↻ https://uns-auph24.hotcrp.com/search?q=&t=active ☆

Seminarski rad C: AUPH 2024 stefan.nikolic@dm.uns.ac.rs

Search (All) Search


(All) in Active Search

Search [Advanced search](#) [Saved searches](#) [View options](#)

ID ▼	Title	Status	# Reviews
<input type="checkbox"/>	#1 SAT-Based Exact Synthesis: Encodings, Topology Families, and Parallelism 📄	Submitted	0
<input type="checkbox"/>	#2 Improvements to Technology Mapping for LUT-Based FPGAs 📄	Submitted	0
<input type="checkbox"/>	#3 Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density 📄	Submitted	0
<input type="checkbox"/>	#4 Improving Timing-Driven FPGA Packing with Physical Information 📄	Submitted	0
<input type="checkbox"/>	#5 Timing-Driven Placement for FPGAs 📄	Submitted	0
<input type="checkbox"/>	#6 Improving Annealing Using Directed Moves 📄	Submitted	0
<input type="checkbox"/>	#7 Adaptive FPGA Placement Optimization via Reinforcement Learning 📄	Submitted	0
<input checked="" type="checkbox"/>	#8 A Simple Yet Effective Technique for Global Wiring 📄	Submitted	0
<input checked="" type="checkbox"/>	#9 PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs 📄	Submitted	0
<input type="checkbox"/>	#10 Timing-Driven Pathfinder Pathology and Remediation: Quantifying and Reducing Delay Noise in VPR-Pathfinder 📄	Submitted	0

🔍 Select papers (or [select all 10](#)), then [Download](#) · [Tag](#) · [Assign](#) · [Decide](#) · [Mail](#)

# Instalirati VTR

 Product ▾ Solutions ▾ Resources ▾ Open Source ▾ Enterprise ▾ Pricing

Sign in Sign up




verilog-to-routing / vtr-verilog-to-routing Public







Notifications Fork 392 Star 1k

[Code](#) [Issues 423](#) [Pull requests 57](#) [Discussions](#) [Actions](#) [Security](#) [Insights](#)

master ▾ 118 Branches 13 Tags

[Code](#) ▾

 **duck2** Merge pull request #2807 from verilog-to-routing/fix-flat-sync   113655f · yesterday 21,478 Commits


 .github	[CI] Turned Off Nightly Tests on PR and Merge	2 weeks ago
 abc	Final Commit	last year
 ace2	updated cmake_minimum_required in CMakeLists.txt	last year
 blifexplorer	initial fix.	6 months ago
 cmake	[core] now use clang-format-14 as default in formatting ...	last year
 dev	Merge branch 'master' into routing constraints	6 months ago


## About


Verilog to Routing -- Open Source CAD Flow for FPGA Research


[verilogtorouting.org](https://verilogtorouting.org)

fpga routing eda cad verilog synthesis placement vtr vpr

 Readme

 View license

 Activity

 Custom properties

```
./vpr/vpr vtr_flow/arch/timing/k6_N10_40nm.xml  
vtr_flow/benchmarks/blif/6/clma.blif -disp on -route_chan_width 120
```

# Proveriti da li radi

```
Final critical path delay (least slack): 8.47991 ns, Fmax: 117.926 MHz
Final setup Worst Negative Slack (sWNS): -8.47991 ns
Final setup Total Negative Slack (sTNS): -377.154 ns

Final setup slack histogram:
[ -8.5e-09: -7.7e-09] 4 ( 4.0%) |*****
[ -7.7e-09: -7e-09] 2 ( 2.0%) |***
[ -7e-09: -6.3e-09] 1 ( 1.0%) |**
[ -6.3e-09: -5.5e-09] 1 ( 1.0%) |**
[ -5.5e-09: -4.8e-09] 3 ( 3.0%) |*****
[ -4.8e-09: -4e-09] 28 ( 27.7%) |*****
[ -4e-09: -3.3e-09] 25 ( 24.8%) |*****
[ -3.3e-09: -2.6e-09] 23 ( 22.8%) |*****
[ -2.6e-09: -1.8e-09] 5 ( 5.0%) |*****
[ -1.8e-09: -1.1e-09] 9 ( 8.9%) |*****

Final geomean non-virtual intra-domain period: 8.47991 ns (117.926 MHz)
Final fanout-weighted geomean non-virtual intra-domain period: 8.47991 ns (117.926 MHz)

Incr Slack updates 1 in 0.000668294 sec
Full Max Req/Worst Slack updates 1 in 7.069e-06 sec
Incr Max Req/Worst Slack updates 0 in 0 sec
Incr Criticality updates 0 in 0 sec
Full Criticality updates 1 in 0.00207218 sec
Flow timing analysis took 2.7172 seconds (2.39946 STA, 0.317743 slack) (107 full updates: 77 setup, 0 hold, 30 combined).
VPR succeeded
The entire flow of VPR took 27.22 seconds (max_rss 166.9 MiB)
Incr Slack updates 29 in 0.0174967 sec
Full Max Req/Worst Slack updates 3 in 1.9626e-05 sec
Incr Max Req/Worst Slack updates 26 in 0.000322827 sec
Incr Criticality updates 24 in 0.0228888 sec
Full Criticality updates 5 in 0.0079734 sec
```