Načrtovanje in vrednotenje ASIC digitalnih vezij z orodjem OpenLANE

Patricio Bulić[1], **Ratko Pilipović**[1], **Uroš Lotrič**[1]

[1]*Univerza v Ljubljani, Fakulteta za računalništvo in informatiko*
E-pošta: pa3cio@fri.uni-lj.si

Design and evaluation of ASIC digital circuits with OpenLANE

This short article presents the OpenLane open source design flow for the physical implementation of digital circuits. OpenLANE is a tape-out-hardened flow that attacks the barriers of cost and expertise. OpenLane aims at democratizing ASIC design. Individuals and SMEs can use it to make their chips. Besides, it can replace expensive industrial tools in designing and evaluating digital circuits in an academic environment. We use three digital multipliers to compare the OpenLane design flow with the Cadence tool.

1 Uvod

Industrijska orodja za fizično implementacijo ASIC vezij so zeo draga in tako pogosto nedostopna raziskovalnim ustanovam, manjšim podjetjem in posameznikom. Namen tega članka je, da nas seznani z načrtovanjem digitalnih integriranih vezij z uporabo odprtokodnega orodja OpenLane ASIC VLSI [1] in procesa Skywater 130nm PDK [2]. OpenLane je ASIC VLSI načrtovalski proces, zgrajen okoli odprtokodnih orodij . OpenLANE je avtomatiziran načrtovalski proces iz RTL v GDSII, ki temelji na več odprtokodnih orodij (OpenROAD, Yosys, Magic, Netgen, Fault, OpenSTA, SPEF-Extractor, ...) ter skriptah, ki ta orodja poganjajo v pravilnem vrstnem redu, po potrebi preoblikujejo njihove vhode in izhode ter organizirajo rezultate in poročila [3], [4]. Cilj OpenLANE je ustvariti GDSII datoteko iz RTL opisa brez človeškega posredovanja. OpenLANE je prilagojen za Skywater 130nm odprtokodni PDK (Process Design Kit) in se lahko uporablja za izdelavo namenskih makro modulov in čipov. OpenLane in 130nm PDK sta rezultat usklajenega prizadevanja različnih akterjev (univerz in industrije) z namenom demokratizacije ASIC načrtovanja vezij.

2 Proces načrtovanja RTL v GDSII

Od zasnove digitalnega vezja do končnega izdelka uporabljamo ponavljajoč se proces načrtovanja. Podrobnosti samega procesa se lahko spremenijo glede na zahteve (npr. velikost, časovne omejitve, ipd.), vendar osnovni koncepti ostajajo enaki. Proces načrtovanja RTL v GDSII razdelimo na 11 korakov:

1. Načrtovanje RTL - načrtovanje digitalnega vezja na ravni prenosov signalov med registri. Pri načrtovanju RTL praviloma uporabljamo jezike za opis hardverja (HDL), ki zagotavljajo abstrakcijo digitalnega vezja z uporabo kombinacijske in sekvenčne logika (registrov). Orodje OpenLane zahteva v tem koraku načrtovanje z uporabo jezika Verilog.

2. Verifikacija RTL - omogoča simulacijo in verifikacijo RTL modula.

3. Logična sinteza (Logic Synthesis) – Logična sinteza preslika RTL modul v standardne logične celice. Postopek običajno poteka v dveh korakih:

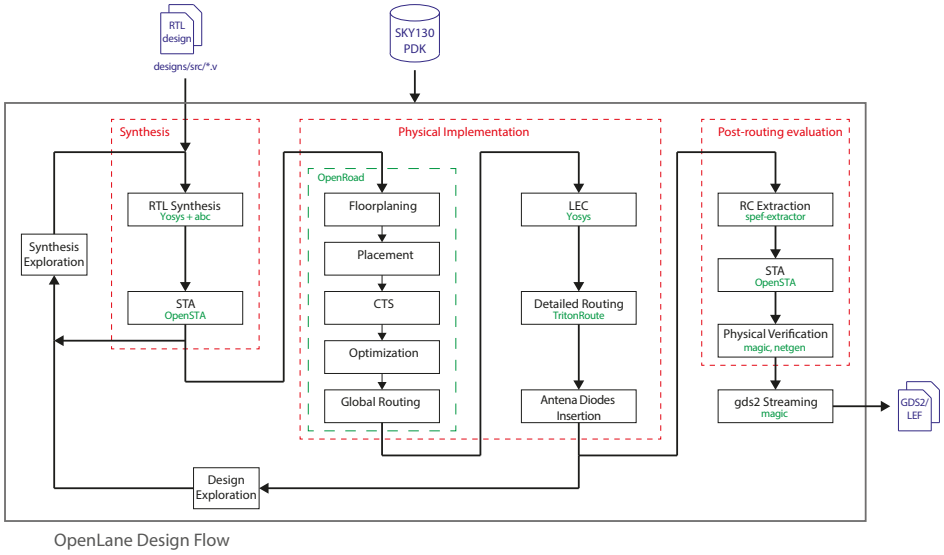
   (a) preslikovanje HDL opisa digitalnega vezja v generična logična vrata,

   (b) preslikovanje iz generičnih logičnih vrat v standardne logične celice izbranega procesa PDK. Standardne celice imajo fiksno višino, njihova širina pa je praviloma večkratnik osnovne širine. Vsaka standardna celica v PDK ima datoteke tipa SPICE, HDL, LIBERTY, LEF in DEF z abstraktnimi in fizičnimi postavitvami, ki jih uporabljajo različna orodja na različnih stopnjah v procesu fizične implementacije vezij.

4. Statična časovna analiza (Static Timing Analysis - STA) po sintezi. Statična časovna analiza je metoda verifikacije časovnih zahtev s preverjanjem vseh možnih poti in iskanju morebitnih kršitev časovnih zahtev. STA razdeli dizajn na časovne poti, izračuna zakasnitev širjenja signala vzdolž vsake poti in preveri kršitve časovnih omejitev.

5. Načrtovanje tlorisa (Floorplanning) - gre za določitev površine na siliciju, kjer bodo postavljene logične celice vezja ter makro bloki. V tej fazi se določi tudi postavitev pinov ter napajalnih linij (Power distribution Network) na čipu.

6. Postavitev (Placement)– v tem koraku se celice postavijo v posamezne vrstice znotraj tlorisa. Celice so medseboj poravnane glede na opis priključkov v datoteki LEF. Postavitev poteka v dveh korakih: globalno in podrobno. Pri globalni postavitvi poskušamo najti optimalen položaj celic. Podrobna



Slika 1: Potek procesa načrtovanja iz RTL v GDSII v okolju OpenLane.

umestitev nato postavi celice tako, da skuša minimizirati dolžine povezovalnih žic, zakasnitve, porabo, ter pustiti dovolj prostora za povezovalne in napajalne linije.

7. Sinteza urinega drevesa (Clock Tree Synthesis - CTS) – v tem koraku se ustvari distribucijsko omrežje urinega signala, ki se uporablja za dostavo ure vsem sinhronim celicam. Glavni cilj je ustvariti distribucijsko omrežje, ki zagotavlja minimalen zdrs urinega signala (clock skew).

8. Usmerjanje (Routing) – v tem koraku se določijo povezave med standardnimi celicami z uporabo razpoložljivih kovinskih plasti po generiranju urinega drevesa in napajalnega omrežja. Na koncu tega koraka se tvori datoteka GDSII, ki vsebuje 3D nabor poligonov iz postavitve integriranega vezja in jo uporabimo za generiranje mask. GDSII je binarna datoteka, ki je de facto industrijski standard za izmenjavo podatkov pri načrtovanju integriranih vezij. Datoteke GDSII so končni izhodni izdelek procesa načrtovanja integriranega vezja in se predajo proizvajalcem integriranih vezij.

9. Fizično preverjanje (Physical verification) - v tem koraku preverimo pravilnost postavitve in povezav določenih z usmerjanjem, tako da se preveri, ali je dizajn skladen z vsemi tehnološkimi zahtevami (Design Rule Checking - DRC), skladen z rezultatom logične sinteze (Layout Versus Schematic - LVS), da ni antenskih efektov ter da dizajn ustreza vsem električnim zahtevam (Electrical Rule Checking - ERC). V primeru kakršnihkoli napak, se ponovi enega ali več predhodnih korakov.

10. generiranje mask (Layout Post Processing) - to je postopek pretvorbe datoteke GDSII v fizične maske.

Slika 1 prikazuje potek procesa načrtovanja iz RTL v GDSII.

3 Uporaba orodja OpenLane

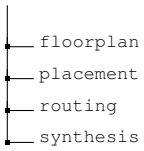
3.1 Ustvarjanje novega dizajna

Vsak dizajn mora imeti lastno konfiguracijsko datoteko s spremenljivkami, ki bodo določale potek načrtovalskega procesa ter nekatere robne pogoje, kot npr. velikost dizajna na siliciju ali širina napajalnih linij ipd.). Orodje OpenLane vsebuje ukaz za ustvarjanje novega dizajna in njegove konfiguracijske datoteke [5]. Ta ukaz ustvari naslednjo imeniško strukturo:

```
designs/<design_name>
├── config.tcl
└── src
```

Priporočljivo je, da Verilog datoteke hranimo v imeniku src in da ima vrhnji modul našega dizajna ime `<design_name>.v`. Ko bomo zagnali načrtovalski proces, se bo ustvarila naslednja imeniška struktura:

```
<design_name>
├── config.tcl
├── src
│   └── <design_name>.v
└── runs
    └── <time_stamp>
        ├── config.tcl
        ├── {logs, reports, tmp}
        │   ├── cts
        │   ├── signoff
        │   ├── floorplan
        │   ├── placement
        │   ├── routing
        │   └── synthesis
        └── results
            ├── final
            ├── cts
            └── signoff
```

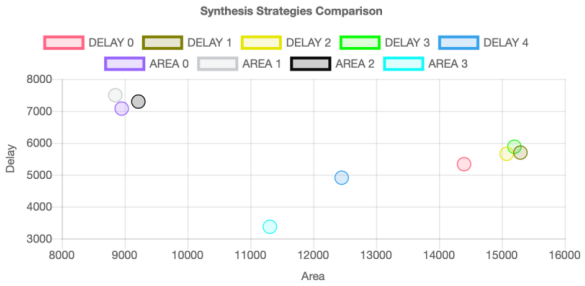


V mapi `results` bodo rezultati po posameznih korakih, v mapi `reports` pa bodo poročila po posameznih korakih.

3.2 Sinteza in sintezno preiskovanje

Best Area	Best Gate Count	Best Delay
8850.99	898.0	3368.1
AREA 1	AREA 1	AREA 3

Strategy	Gate Count	Area (um^2)	Delay (ps)	Gates Ratio	Area Ratio	Delay Ratio
DELAY 0	1435.0	14398.81	5331.08	1.597	1.626	1.582
DELAY 1	1550.0	15295.92	5695.01	1.726	1.728	1.69
DELAY 2	1539.0	15076.96	5656.02	1.713	1.703	1.679
DELAY 3	1509.0	15199.58	5880.06	1.68	1.717	1.745
DELAY 4	1306.0	12451.94	4906.86	1.454	1.406	1.456
AREA 0	919.0	8952.34	7076.12	1.023	1.011	2.1
AREA 1	898.0	8850.99	7494.71	1.0	1.0	2.225
AREA 2	955.0	9217.59	7294.53	1.063	1.041	2.165
AREA 3	1366.0	11309.6	3368.1	1.521	1.277	1.0



Slika 2: Zbirno poročilo sinteznega preiskovanja.

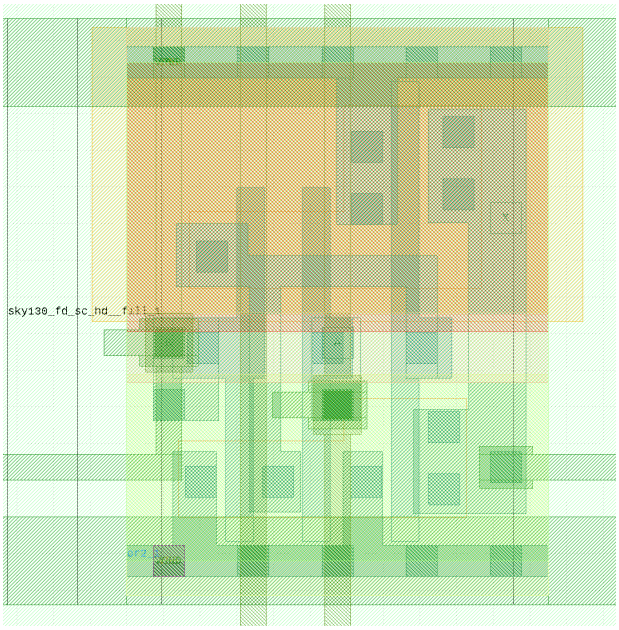
Priporočljivo je, da najprej zaženemo t.i. sintezno preiskovanje, s katerim bomo zagnali vse razpoložljive strategije sinteze (trenutno jih je devet). Vsaka strategija skuša optimizirati vezje za določene zahteve glede prostora in zakasnitev. Izhod tega koraka bodo devet možnih netlist na ravni vrat v mapi `results/synthesis` in zbirno poročilo v mapi `reports/synthesis`. Za sintezo na ravni vrat se uporablja orodje yosys [6], statična časovna analiza pa se izvede na netlisti z uporabo OpenSTA [3], [4]. STA se izvede na vsaki točki v preiskanem prostoru. Slika 2 prikazuje zbirno poročilo sinteznega preiskovanja.

3.3 Fizična implementacija

Ko s pomočjo sinteznega preiskovanja izberemo sintezno strategijo, ki ustreza našim zahtevam, lahko poženemo fizično implementacijo. Čeprav orodje OpenLane opravi vse avtomatično korake pri fizični implementaciji, moramo pred fizično implementacijo nastaviti konfiguracijsko datoteko. Le-ta bo vodila avtomatiziran proces fizične implementacije (iz RTL v GDSII). Konfiguracijska datoteka vsebuje množico spremenljivk, za vsak posamičen korak procesa. Navodila za pisanje konfiguracijskih datotek ter pomen posamičnih spremenljivk najdemo v [7], [8]. Primer konfiguracijske datoteke uporabljene za dizajne v tem članku je:

```
set ::env(DESIGN_NAME) LOBO12_12_8
set ::env(VERILOG_FILES) [glob $::env(DESIGN_DIR)/src/*.v]

set ::env(CLOCK_PERIOD) "10.0"
```



Slika 3: Prikaz postavitve vezja in celic v Klayout (slika prikazuje ena ALI vrata).

```
set ::env(CLOCK_PORT) "clk"
set ::env(SYNTH_CAP_LOAD) {33.442}
set ::env(SYNTH_ADDER_TYPE) {CSA}
set ::env(SYNTH_STRATEGY) {DELAY 4}

# Set the core area and the placement density
set ::env(FP_CORE_UTIL) 2
set ::env(PL_TARGET_DENSITY)
      [expr ($::env(FP_CORE_UTIL)+5)/100.0 ]

# power lines : larger pitch requires larger core area
set ::env(FP_PDN_VPITCH) 153.6
set ::env(FP_PDN_HPITCH) 153.18
set ::env(FP_PDN_VOFFSET) 16.32
set ::env(FP_PDN_HOFFSET) 16.65

# Specifies the insertion strategy of diodes
set ::env(DIODE_INSERTION_STRATEGY) 3
```

V zgornji konfiguracijski datoteki nastavimo periodo urinega signala, velikost kapacitivnega bremena, tip seštevalnika pri sintezi, sintezno strategijo, velikost področja in gostoto celic za načrtovanje tlorisa in postavitev celic [8], razmik med napajalnimi linijami ter njihov odmik od roba jedra in nazadnje še strategijo vstavljanja diod za odpravljanje antenskega učinka med izdelavo čipa.

Fizična implementacija temelji na orodjih znotraj aplikacije OpenRoad [3], [4]. Med fizično implementacijo se po sintezi opravi načrtovanje tlorisa, postavitev celic, usmerjanje povezovalnih linij, postavitev napajalnih linij ter vstavljanje diod na vrata celic z namenom odpravljanja antenskega učinka (glej sliko 1).

3.4 Verifikacija po fizični implementaciji

Po fizični implementaciji se izvede več verifikacij implementiranega dizajna. Najprej se izvedeta preverjanji DRC in LVS. Nato sledi preverjanje antenskih učinkov. Po ekstrakcija parazitskih kapacitivnosti in upornosti sledi še en krog statične časovne analize za natančnejše ocene zakasnitev, ki ustrezajo dejanski fizični postavitvi. Končni rezultat je datoteka GDSII. V mapah po posameznih ko-

rakih najdemo tudi druge rezultate (npr. datoteki LEF in DEF, ki predstavljat fizično postavitev integriranega vezja, SPICE model vezja ipd.). Načrtovalec lahko po potrebi z uporabo orodij Magic ali Klayot pregleda postavitev vezja in posameznih celic (slika 3) ali pa zažene dodatne simulacije z orodjem SPICE (slika 4).

```
.subckt sky130_fd_sc_hd__inv_2 A VGND VNB VPB VPWR Y
X0 Y A VGND VNB sky130_fd_pr__nfet_01v8 w=650000u l=150000u
X1 VPWR A Y VPB sky130_fd_pr__pfet_01v8_hvt w=1e+06u l=150000u
X2 VGND A Y VNB sky130_fd_pr__nfet_01v8 w=650000u l=150000u
X3 Y A VPWR VPB sky130_fd_pr__pfet_01v8_hvt w=1e+06u l=150000u
.ends

X_1403_ _2631_/Q _1403_/B _1403_/C VGND VGND VPWR VPWR _1403_/X sky130_fd_sc_hd__or3_1
XFILLER_233_1425 VGND VPWR VPWR VGND sky130_ef_sc_hd__decap_12
XFILLER_190_1141 VGND VGND VPWR VPWR sky130_fd_sc_hd__decap_6
XFILLER_272_1485 VGND VPWR VPWR VGND sky130_ef_sc_hd__decap_12
X_2383_ _2532_/A VGND VGND VPWR VPWR _2383_/Y sky130_fd_sc_hd__inv_2
XFILLER_111_821 VGND VGND VPWR VPWR sky130_fd_sc_hd__fill_1
XFILLER_229_505 VGND VGND VPWR VPWR sky130_fd_sc_hd__decap_3
```

Slika 4: Izsek datoteke SPICE za načrtovano vezje.

3.5 Raziskovalni zagon

OpenLane ponuja skripto run_designs.py, ki lahko izvede več zagonov fizične implementacije vzporedno z različnimi konfiguracijami. En zagon je sestavljen iz RTL dizajna in konfiguracijske datoteke. Z več zagoni, pri čemer ima vsak zagon svojo konfiguracijo, lahko poiščemo zagon, ki najbolje ustreza našim zahtevam. Skripta zahteva ločeno konfiguracijsko datoteko v kateri za posamezno spremenljivko lahko podamo več vrednoti. Skripta bo tako tvorila kartezični produkt med vsemi spremenljivkami in tako tvorila več konfiguracijskih datotek in naposled zagnala več zagonov. Primer konfiguracije pri kateri poskušamo med zagoni izbrati različne seštevalnike in različne strategije pri sintezi je:

```
SYNTH_ADDER_TYPE=(YOSYS,RCA,CSA)
SYNTH_STRATEGY=(DELAY 0,AREA 3)
extra="
"
```

Na osnovi te konfiguracije bomo tvorili šest različnih zagonov. Na koncu bomo dobili rezultate in poročila za vsak posamičen zagon fizične implementacije.

3.6 Poročila

Po vsakem izvedenem koraku procesa sinteze in fizične implementacije nam orodje OpenLane v pripadajoči mapi pripravi kopico poročil. Največkrat nas zanimajo velikost vezja, maksimalna zakasnitev in moč (statična in dinamična). Najbolj zanesljivo oceno velikosti vezja bomo dobili po izvedenem usmerjanju povezav, saj moramo med celice postaviti povezovalne in napajalne linije, zato bo površina po tem koraku zagotovo večja, kot pa po sintezi. Najbolj zanesljivo oceno zakasnitev in moči pa bomo dobili po ekstrakciji parazitskih kapacitivnosti in upornosti, saj se bo zaradi parazitskih kapacitivnosti in upornosti podaljšal čas potovanj signalov in poraba toka. Slike 5, 6 in 7 prikazujejo poročila o površini, zakasnitvi in moči.

4 Vrednotenje vezij

Za prikaz vrednotenja vezij z orodjem OpenLane smo uporabili tri približne 16-bitne množilnike opisane v [9].

```
=========================================================================
 report_design_area
=========================================================================
Design area 17892 u^2 4% utilization.
```

Slika 5: Poročilo o površini vezja po končanem usmerjanju.

```
-------------------------------------------------------------------------
                          10.18   data required time
                          -5.33   data arrival time
-------------------------------------------------------------------------
                           4.85   slack (MET)
```

Slika 6: Poročilo o tipični maksimalni zakasnitvi po ekstrakciji parazitov.

Trije približni množilniki imajo različno napako pri računanju, posledično pa so tisti z večjo napako manjši, hitrejši in porabijo manj energije. V delu [9] smo množilnike sintetizirali in vrednotili z uporabo Cadence Genus 171 in TSMC 180nm PDK. Pri evalvaciji smo uporabili urin signal s period 100ns, kapacitivnost bremena 10fF in in napajalno napetost 1.8V. Verilog implementacije množilnikov so v [10]. Rezultati implementacije in vrednotenja so prikazani v tabeli 1. Množilnike smo vrednotili z vidika maksimalne zakasnitve (*Delay*), dinamične moči(*Power*), površine na čipu (*area*) ter produkta zakasnitve, moči in površine (*Power-Area-Delay-Product, PADP*).

Tabela 1: Rezultati za 16-bitne LOBO množilnike z uporabo Cadence Genus 171 in TSMC 180nm PDK.

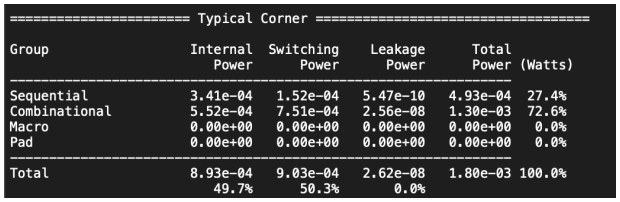
Množilnik	**Delay** [ns]	**Power** [μW]	**Area** [μm^2]	**PADP** [nJ $\times$ μm^2]	**PDP ratio**
LOBO12-0-0	25.14	3150	17386	1376.81	1
LOBO12-8-8	16.69	2620	14024	613.24	2.245
LOBO12-12-8	14.61	2150	12126	380.89	3.615

Ista vezja smo vrednotili tudi z orodjem OpenLane in Skywater 130nm PDK.. Po začetnem sinteznem raziskovanju smo izbrali ustrezno strategijo (DELAY 4) in nato zagnali fizično implementacijo vsakega množilnika. Pri evalvaciji smo uporabili urin signal s periodo 10ns, kapacitivnost bremena 33.442fF in napajalno napetost 1.8V. Površino, ki jo zaseda posamičen množilnik dobimo po usmerjanju v imeniku `reports/routing`, tipično zakasnitev in moč pa po po ekstrakciji parazitov v imeniku `reports/signoff`. Rezultati so prikazani v tabeli 2.

Čeprav uporabljamo drugi PDK in druga orodja v procesu fizične implementacije vezij, pridemo z odprtokodnim orodjem OpenLane do podobnih ocen velikosti, zakasnitev ter moči kot z dragim industrijskim orodjem Cadence Genus.

5 Zaključek

V članku smo na kratko predstavili orodje odprto kodno OpenLane namenjeno fizični implementaciji vezij. Na primeru treh vezij smo predstavili njegovo uporabo za evalvacijo vezij. Rezultati jasno nakazujejo, da je za potrebe evalvacije orodje popolnoma primerljivo z dragimi



Slika 7: Poročilo o tipični moči po ekstrakciji parazitov.

Tabela 2: Rezultati za 16-bitne LOBO množilnike z uporabo OpenLane in Skywater 130nm PDK.

Množilnik	**Delay** [ns]	**Power** [μW]	**Area** [μm^2]	**PADP** [nJ $\times$ μm^2]	**PDP ratio**
LOBO12-0-0	7.20	1300	29197	273.28	1
LOBO12-8-8	6.39	943	21636	130.37	2.096
LOBO12-12-8	5.33	903	17892	86.11	3.174

industrijski orodji. Edina pomanjkljivost orodja je nekoliko starejši PDK.

6 Zahvala

Uporabo orodja Cadence sta omogočila prof. dr. Dušan Raič in prof. dr. Drago Strle za kar se jima avtorji iskreno zahvaljujemo. Prof. dr. Dušanu Raiču se zahvaljujeno za vso pomoč in razlage. Raziskavo je omogočila Javna agencija za raziskovalno dejavnost Republike Slovenije (ARRS) v okviru raziskovalnih programov P2-0359 Vseprisotno računalništvo in P2-0241 Sinergetika kompleksnih sistemov in procesov.

Literatura

[1] –. "The OpenLane". (), spletni naslov: https://github.com/The-OpenROAD-Project/OpenLane. (accessed: 21.07.2022).

[2] A. A. Ghazy in M. Shalan, "OpenLANE: The Open-Source Digital ASIC Implementation Flow", 2020.

[3] T. Ajayi, D. Blaauw, T.-B. Chan in sod., "OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain", 2019.

[4] T. Ajayi, V. A. Chhabria, M. Fogaça in sod., "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project", v *Proceedings of the 56th Annual Design Automation Conference 2019*, zbirka DAC '19, Las Vegas, NV, USA: Association for Computing Machinery, 2019, ISBN: 9781450367257. DOI: 10.1145/3316781.3326334. spletni naslov: https://doi.org/10.1145/3316781.3326334.

[5] –. "The OpenLane Designs Folder". (), spletni naslov: https://github.com/The-OpenROAD-Project/OpenLane/blob/master/designs/README.md. (accessed: 21.07.2022).

[6] C. Wolf, J. Glaser in J. Kepler, "Yosys-A Free Verilog Synthesis Suite", 2013.

[7] –. "The OpenLane Configuration Files". (), spletni naslov: https://github.com/The-OpenROAD-Project/OpenLane/blob/master/docs/source/configuration_files.md. (accessed: 21.07.2022).

[8] ——, "The OpenLane Variables Information". (), spletni naslov: https://github.com/The-OpenROAD-Project/OpenLane/blob/master/configuration/README.md. (accessed: 21.07.2022).

[9] R. Pilipović in P. Bulić, "On the Design of Logarithmic Multiplier Using Radix-4 Booth Encoding", *IEEE Access*, let. 8, str. 64 578–64 590, 2020. DOI: 10.1109/ACCESS.2020.2985345.

[10] R. Pilipovic. "LOBO multiplier - supplemental materials". (2021), spletni naslov: https://dx.doi.org/10.21227/1fx5-5207.