Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Blaž Kelbl

# Analiza energijske učinkovitosti vgrajenega mikroprocesorja BA20

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Patricio Bulić

Ljubljana, 2014

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Analizirajte energijsko učinkovitost različnih implementacij vgrajenega mikroprocesorja Beyond Semiconductors BA20. Ugotovite vpliv frekvence urinega signala ter arhitekture in organizacije procesorja na porabo moči. Kot testni program za vrednotenje energijske učinkovitosti uporabite sintetični program Dhrystone, ki ga prevedite z uporabo dveh naborov stikal (optimizacija velikosti in optimizacija hitrosti) ter tako ocenite vpliv optimizacij kode na porabo moči. Sintezo procesorja ter analizo porabe energije opravite z uporabo programskega paketa Cadence Encounter.

# Izjava o avtorstvu diplomskega dela

Spodaj podpisani Blaž Kelbl, z vpisno številko **63980062**, sem avtor diplomskega dela z naslovom:

*Analiza energijske učinkovitosti vgrajenega mikroprocesorja BA20*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Patricia Bulića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 18. septembra 2014 Podpis avtorja:

*Zahvaljujem se sodelavcem v podjetju Beyond Semiconductors, še posebej Mihu Dolencu in Gyorgyu Jeneyu za vse predano znanje, Branku Dervenšku in Mitji Pufiču za nenadomestljivo pomoč pri obvladovanju vsega programskega in konzolnega, Matjažu Breskvarju za kritični pogled ter Juretu Cigliču za strpnost. Poleg tega gre moja zahvala profesorjem ter asistentom s področja logike in sistemov, v prvi vrsti Urošu Lotriču, Branku Šteru za priložnost demonstriranja, Dušanu Kodeku za odlično knjigo in predavanja ter Patriciu Buliću in Veselku Guštinu za priložnost dela v Laboratoriju za računalniško arhitekturo.*

*Seveda se zahvaljujem tudi družini za potrpljenje.*

Maji in čunkam.

# Kazalo

# Povzetek

Številne tehnike izboljševanja energijske učinkovitosti procesorjev so bile predlagane in uporabljene v različnih procesorjih. Načrtovalci se pogosto soočijo z dilemo, katero izmed tehnik uporabiti, kolikšno je izboljšanje in katere kompromise zahteva. V procesorju se srečujeta strojna in programska oprema, zato mora biti vsaka optimizacija ocenjena v kontekstu izvedljivosti programske opreme in implementacije strojne opreme. Izpostavljenemu izzivu sistemske kompleksnosti se posvečamo s pomočjo primerjalne sistemske analize procesorja Beyond BA20 v razumnih omejitvah načrtovanja programske in strojne opreme. Vsako izmed implementacij oziroma arhitekturnih izbir ocenimo v luči učinka na porabo moči.

**Ključne besede:** nizka poraba moči, mikroprocesor, procesor, sinteza, standardne celice, VLSI, arhitektura.

# Abstract

Many techniques to achieve better energy efficiency have been proposed and applied in various processors. Designers are often faced with dilemma which technique to implement and what are the improvements and tradeoffs that certain technique brings. The processor is the point where software and hardware meet, and any potential optimization has to be evaluated in context of feasibility in regard to software and hardware implementation constraints. This is exposing systemic complexity challenges that we address through systemic analysis of Beyond BA20 in comparison with reasonable practical limitations imposed by hardware and software design. Each implementation or architectural choice, identified to be important, is evaluated in terms of power consumption impact.

**Keywords:** low power, microprocessor, processor, synthesis, standard cell, VLSI, architecture.

# Poglavje 1

# Uvod

Leta 1965 je Gordon Moore za Electronic Magazine zapisal [6]: "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will remain nearly constant for at least 10 years." Skoraj 50 let kasneje njegovo opažanje še vedno drži, le čas podvojitve kompleksnosti je nekoliko drugačen. Navadno Moorov zakon razumemo, da se, ob nespremenjenih stroških, število tranzistorjev na enaki površini podvoji vsakih 18 mesecev. Povečanje števila tranzistorjev pa na drugi strani implicira njihovo zmanjšanje v enakem razmerju. Zgodovinsko je bila glavna implikacija tega zakona nenehno povečevanje števila logičnih elementov na integriranem vezju in tako vse zmogljivejša logična vezja. Včasih spregledan vidik Moorovega zakona je, da se z zmanjšanjem tranzistorjev zmanjšuje tudi poraba moči posameznega tranzistorja.

Z mobilnimi in pametnimi napravami ter v zadnjem času priljubljenim internetom stvari se pozornost trga elektronskih naprav vse bolj usmerja v vgrajene sisteme. Ti sistemi, ki opravljajo naloge okoljskih merilnikov, srčnih spodbujevalnikov, označevalnih čipov, kuhinjskih pripomočkov in zabavne elektronike, so pogosto baterijsko napajani. Čeprav imajo navadno ozek nabor nalog, je njihovo srce vseeno splošno-namenski mikroprocesor. Ti procesorji so javnosti manj znani kot visokocenovni procesorji za namizne računalnike, toda njihovo prisotnost je čutiti povsod. Medtem ko procesorji namenjeni namiznim računalnikom porabijo na desetine in stotine vatov, posebno hlajenje pa je zahtevano že v skromnih strežniških

sobah, vgrajeni mikroprocesorji delujejo v območju milivatov in milimetrov.

Sistem z vgrajenim mikroprocesorjem večino časa prebije v stanju mirovanja. V tem načinu večini vezja odvzamemo urin signal in tako zagotovimo, da se ohranja stanje. Ob prihodu zahtev za procesiranje se mikroprocesor zbudi, obdela naloge, nato pa spet vrne v stanje mirovanja. Poraba moči sistema v stanju mirovanja je nekaj redov manjša od porabe moči med aktivnim delovanjem. Da bi minimirali povprečno porabo moči, moramo zagotoviti, da je čas obdelovanja zahtev čim krajši, a hkrati pazljivo dodajati procesne kapacitete, saj te povečujejo kompleksnost in velikost logičnega vezja ter njegovo skupno porabo.

Mikroprocesorji izvajajo programe, shranjene v glavnem pomnilniku, ki je v primeru vgrajenih sistemov na istem čipu. Statični pomnilniki v primerjavi z mikroprocesorji zasedajo velik del površine čipa ter porabijo nezanemarljiv delež moči. Zato je pomembno, da so pomnilniki čim manjši. Pogosto je razpoložljiv pomnilnik ena izmed glavnih težav, s katero se mora spopasti programer vgrajenega sistema, vendar tudi učinkovitosti reševanja nalog ne moremo zanemariti. Med pristopi, ki povečujejo zmogljivost programske opreme, so optimizacije pri prevajanju, kot sta razvijanje zank in funkcijsko vstavljanje. Izbira takih optimizacij pa poveča velikost prevedene kode ter stimulira potrebo po večjih, energijsko potratnejših pomnilnikih.

Tako procesor kot glavni pomnilnik v vgrajenem sistemu zahtevata, da poiščemo ustrezno ravnotežje v načrtovanju in izvedbi integriranega vezja ter programske opreme, saj vpliva enega na drugega ne moremo zanemariti. V diplomskem delu smo raziskovali vpliv funkcijskega nabora ter takta delovanja vezja in tehnik prevajanja programske opreme na povprečno porabo moči. Poskušali smo odkriti pravi nabor parametrov, oziroma ponuditi podatke in pravila, ki bi načrtovalcu sistema pomagala najti pravo razmerje.

# Poglavje 2

# Mikroprocesor Beyond BA20

Uporabili smo mikroprocesor podjetja Beyond Semiconductors BA20. BA20 je najmanjši predstavnik družine mikroprocesorjev, ki implementirajo Beyond Architecture 2. Cilj arhitekture BA2 je zagotovitev najmanjše velikosti prevedenih programov, kar je vgrajenim sistemom še posebej privlačna lastnost.

## 2.1 Arhitektura BA2

Beyond Architecture 2 je izpeljava in nadgradnja BA1 ukaznega nabora s poudarkom na doseganju minimalne velikosti prevedenih programov [18]. Ortogonalen RISC ukazni nabor je razširjen s kompozitnimi ukazi, s katerimi procesor opravi pogoste in prostorsko potratne naloge, kot so funkcijski prologi in epilogi. Arhitektura definira izvedenke posameznih ukazov s takojšnjimi operandi v 16-, 24-, 32- in 48-bitnih inačicah, prevajalnik pa izbere najkrajšo obliko ukaza, v katero lahko zapišemo takojšnji operand. Na ta način so programi, prevedeni v strojno kodo 10 do 40 odstotkov manjši od programov prevedenih v običajen 32-bitni RISC nabor [18].

Večina ukazov deluje nad operandi, shranjenimi v setu 32 ali 16 splošnonamenskih registrov. Izvedenke arhitekture obsegajo ukaze za delo z operandi v plavajoči vejici, celoštevilski aritmetiki, namenjeni digitalni obdelavi signalov, in uporabniško definirane ukaze.

## 2.2 Organizacija

Jedro procesorja tvori enota za izvajanje preprostih operacij v celoštevilski aritmetiki. Enoti za podatkovne dostope in prevzem ukazov v harvardski arhitekturi izvajata dostope do vgrajenega pomnilnika v enem ciklu. Kompleksnejše operacije, kot sta deljenje in množenje, se izvajajo v posebnih cevovodih, strojno pa je podprto ugotavljanje in premoščanje cevovodnih nevarnosti. Družino mikroprocesorjev BA2 sestavljajo še [19]:

- Beyond BA25 Advanced Application Processor, namenjen zahtevnim sistemom, ki poganjajo Linux in Android operacijske sisteme.
- Beyond BA22, namenjen zmogljivim aplikacijam, ki so občutljive na porabo in velikost.
- Beyond BA21, namenjen aplikacijam s poudarjeno pozornostjo nizki porabi energije.

# Poglavje 3

# Postopek določitve porabe moči

Podatke o porabi moči mikroprocesorja BA20 smo pridobili na podlagi preklopne aktivnosti logičnih vrat in pomnilnih celic sinteznih implementacij mikroprocesorja med izvajanjem testnega programa, prevedenega z uporabo več naborov optimizacijskih možnosti.

## 3.1 Priprava implementacij

Izvorno Verilog Register Transfer Language (RTL) kodo mikroprocesorja BA20 smo implementirali s pomočjo programskega paketa Cadence Encounter Digital Implementation System [21] (Encounter). Programski paket Encounter obsega vse korake, od načrtovanja in verifikacije vezja v RTL, do izdelave datotek, ki jih dostavimo proizvajalcem integriranih vezij [21]. Naš postopek implementacije smo omejili na naslednje korake:

1. Določitev parametrov mikroprocesorja, frekvence urinega signala ter izbire knjižnic, ki opisujejo tehnologijo integriranega vezja. Sintezi priskrbimo tudi podatke o kapacitivnosti in upornosti povezav v izbranem tehnološkem procesu.

2. Logična sinteza (*Logic Synthesis*), oziroma prevajanje abstraktnega RTL modela v RTL datoteko netlist [9], ki vsebuje zgolj logična vrata, pomnilne elemente ter tokovne ojačevalnike, ki so na voljo v knjižnici (*technology library*).

3. Razporeditev logičnih vrat (*Placement*) na neprekrivajoče lokacije na shemi integriranega vezja. Postopek poskuša minimirati razdalje med povezanimi logičnimi vrati ter čim bolj enakomerno pokriti dodeljeno površino integriranega vezja [9]. Površino integriranega vezja, ki je na voljo, smo določili samodejno, tako da logična vrata v datoteki netlist zasedajo 60 odstotkov površine.

4. Sintetiziranje urinega signala (*Clock Insertion*). V tem koraku orodje poveže urin signal s pomnilnimi celicami ter z vstavljanjem ustreznega števila ojačevalnikov in razcepov oblikuje drevesno strukturo [9], tako da zakasnitve urinega signala ne ogrozijo časovne diskretnosti logičnega vezja. Drevesna struktura je oblikovana tako, da se oblikujejo skupine pomnilnih celic, ki jim na podlagi funkcijskega opisa v RTL lahko selektivno odvzamemo urin signal [9]. Na ta način se skrči obseg drevesa, na katerem se v vsaki urini periodi dogajajo prehodi, kar zniža porabo energije ter omogoča uporabo preprostejših pomnilnih celic. Naprednejše nastavitve tega koraka omogočajo izkoriščanje zakasnitev urinega signala, kjer so zakasnitve logičnih vrat take, da znotraj zahtevane urine periode ni mogoče vzpostaviti stabilnega stanja vhodov pomnilnih celic (*Clock skew scheduling*) [9]. Razliko v zakasnitvah ure na izvornem in ciljnem sinhronem elementu lahko prištejemo minimalni urini periodi [5]. S časovno razporeditvijo prehodov urinega signala poleg tega dosežemo, da se vsi prehodi stanj pomnilnih celic ter nivojev urinega signala ne zgodijo sočasno, kar poveča odpornost na presluh ter enakomernejšo porabo energije.

5. Povezovanje logičnih vrat (*Routing*), tako da je zagotovljeno, da zakasnitve ne presegajo omejitev, podanih v prvem koraku oziroma, modificranih omejitev, ki so rezultat predhodnega koraka.

6. Shranjevanje stanja implementacije ter datoteke netlist, ki po koraku sintetiziranja urinega signala poleg logičnih vrat vsebuje tudi strukturo urinega drevesa.

Mikroprocesor BA20 smo implementirali v vseh 24 kombinacijah vrednosti parametrov, navedenih v tabeli 3.1. V implementacijah smo uporabili nabor dveh 65-nanometerskih knjižnic standardnih celic – visoko pragovne (*High voltage threshold*, HVT) in knjižnice s standardnim pragom (*Standard voltage threshold*, SVT).

| Parameter | Vrednosti | | |
|---|---|---|---|
| Frekvenca delovanja | 25 MHz | 50 MHz | 75 MHz |
| Množilnik | Prisoten | Ni prisoten | |
| Delilnik | Prisoten | Ni prisoten | |
| Število splošno-namenskih registrov | 16 | 32 | |

Tabela 3.1: Nabor parametrov implementacije.

Logična vrata knjižnice HVT imajo nižjo statično porabo moči kot SVT, toda večje zakasnitve [2]. Program Encounter nastavimo tako, da samodejno izbira elemente tako, da minimira skupno statično porabo [15], ter zadosti zahtevani urini periodi. Pričakujemo, da bodo izvedbe s krajšo zahtevano urino periodo vsebovale več logičnih vrat knjižnice SVT. Tehnologiji CMOS (*Complementary Metal–Oxide–Semiconductor*) integriranih vezij ter izvorom porabe moči se podrobneje posvečamo v posebnem poglavju.

## 3.2 Določitev preklopne aktivnosti

Datoteko netlist mikroprocesorja, ki smo jo izvozili po koraku povezovanja, smo vključili v testno okolje (*test bench*) ter ga povezali z dvema modeloma RTL vgrajenega pomnilnika v harvardski arhitekturi. Pomnilnik, priključen na ukazno vodilo mikroprocesorja, smo organizirali 64-bitno, pomnilnik, povezan na podatkovno vodilo, pa 32-bitno. Testnem okolju smo dodali števce pomnilniških dostopov ter dodelili del naslovnega prostora navidezni napravi, ki procesorju omogoča komunikacijo s testnim okoljem. V pomnilnik smo naložili binarno datoteko s programom.

Kot testni program smo izbrali program za vrednotenje procesorske zmogljivosti Dhrystone. Program Dhrystone prvenstveno ni namenjen vrednotenju zmogljivosti vgrajenih procesorskih sistemov [13], vendar proizvajalci vgrajenih mikorprocesorjev v publikacijah in promocijskih gradivih pogosto navajajo ravno na ta način izmerjeno zmogljivost [17]. Pomemben razlog pri izbiri je igral kratek čas izvajanja ter njegova pozornost celoštevilski aritmetiki [13].

Izvorno kodo smo prevedli z Beyond Architecture izvedbo prevajalnika Gnu

| Stikalo | Hitrost | Velikost |
|---|---|---|
| Optimizacija | -O3 | -Os |
| Razvijanje zank | da | ne |
| Poravnava operandov | da | da |
| Vstavljanje funckcij | ne | ne |

Tabela 3.2: Nabora optimizacijskih možnosti prevajalnika.

Compiler Collection (GCC) z uporabo dveh naborov stikal. S prvim naborom smo optimirali velikost prevedene kode, z drugim pa hitrost izvajanja. Oba nabora navajamo v tabeli 3.2.

S simulacijo prevedenih programov smo ustvarili datoteko, ki beleži prehode stanj vseh logičnih vrat v datoteki netlist (*Value Change Dump*, VCD). Ob tem smo zabeležili število dostopov do pomnilnika ter število urinih period izvajanja desetih ponovitev zanke programa Dhrystone. Na podlagi povprečnega števila urinih period $C$, potrebnih za izvedbo ene ponovitve Dhrystone zanke ter takta delovanja $f$, smo izračunali zmogljivost sistema v količini Dhrystone MIPS [13]:

$$DMIPS = \frac{f * I}{1757 * C}. \tag{3.1}$$

## 3.3 Določanje porabe moči

Datoteko s preklopno aktivnostjo vezja, datoteko netlist, knjižnice, ki opisujejo električne lastnosti logičnih vrat ter nastavitev urine periode, s katero smo omejili implementacijo mikroprocesorja, smo uvozili v Cadence Encounter Power System (EPS). Orodje EPS omogoča oceno porabe moči s statičnim pristopom ter dinamičnim z uporabo testnih vektorjev [20]. Statična analiza določi porabo moči na podlagi podane vrednosti preklopne aktivnosti - verjetnosti, s katero se na logičnem elementu zgodi prehod. Odločili smo se za dinamično analizo, s katero porabo moči določamo z vektorji v datoteki s preklopno aktivnostjo. Sestavili smo poročila, iz katerih je moč razbrati porabo moči glede na vrsto logičnih vrat oziroma pomnilnih celic, moči porabljene na drevesu urinega signala in porabo moči,

ki pripada preklopom logičnih stanj ter statični porabi.

# Poglavje 4

# Viri porabe moči v tehnologiji CMOS

V tehnologiji CMOS obstajajo trije glavni izvori porabe moči, ki ustrezajo trem kategorijam tokov [1]:

- polnjenje in praznjenje kapacitivnih bremen
- kratkostični tok
- tokovi mirovnega stanja

## 4.1 Polnjenje in praznjenje kapacitivnosti bremen

Moč, ki se troši na izhodnem vozlišču med polnjenjem in praznjenjem kapacitivnosti, oziroma prehodom med stanjema logične ničle in enice, definiramo kot dinamično, $P_d$. Kapacitivnost bremena $C_{load}$ je sestavljena iz kapacitivnosti izhoda logičnih vrat, kapacitivnosti povezav izhoda vrat z vhodi povezanih vrat ter kapacitivnosti povezanih vrat. Moč je tako podana z [1]:

$$P_d = \frac{1}{2}\alpha C_{load} * V_{dd}^2 * f \tag{4.1}$$

kjer je:

- $P_d$: dinamična poraba moči logičnih vrat
- $\alpha$: faktor preklopne aktivnosti
- $C_{load}$: kapacitivnost bremen logičnih vrat
- $V_{dd}$: napajalna napetost
- $f$: frekvenca urinega signala

Faktor $\alpha$ določa preklopno aktivnost signala v času ene urine periode, ta pa v idealiziranem sistemu leži na intervalu [1]:

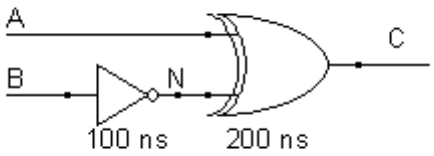
$$0 \leq \alpha \leq 1. \tag{4.2}$$

V obravnavi idealiziranega sistema zanemarimo prehode, ki lahko nastanejo zaradi neuravnoteženih struktur logičnih vrat z zakasnitvami. Slika 4.1 ilustrira tako vezje. Prehod na vhodu B zakasni negator; tako sta na vhodu logičnih vrat XOR $100ns$ prisotni logični enici, kar povzroči neželeni prehod izhoda v logično ničlo ter nazaj. Te prehode imenujemo logični hazardi. Raziskave različnih struktur seštevalnikov spodbujanih z naključno generiranimi števili izkazujejo preklopno aktivnost, ki je med 10 in 20 odstotki višja od pričakovane. Sistemi s faktorjem $\alpha > 6$ pa so bili identificirani v vezjih z občutno raznolikimi podatkovnimi potmi, kot so množilniki [3].

Predpostavki 4.2 idealiziranega sistema v simulaciji zadostimo tako, da zanemarimo zakasnitve logičnih vezij. Tako se vsi prehodi zgodijo sočasno in največ enkrat v eni urini periodi. Našo analizo smo omejili na tak sistem. To poenostavitev smo izbrali zaradi občutno manjše računske zahtevnosti takega procesa ter načrtne uporabe tehnik, ki zmanjšujejo možnosti hazardnih preklopov pri načrtovanju RTL mikroprocesorja. Take tehnike so na primer ohranjanje podatkovnih vhodov v znanih nespremenljivih stanjih. S primerjavo porabe moči idealiziranega sistema s sistemom, v katerem upoštevamo zakasnitve, bi dobili oceno, kako učinkovito so bile metode izogibanja neželenim prehodom uporabljene.

## 4.2 Kratkostični tok

Na sliki 4.2 [8] je prikazan negator v tehnologiji CMOS. Komplementarni par tranzistorjev je povezan med napajalno napetost $V_{DD}$ in ozemljitvijo $V_{SS}$, tako da




Slika 4.1: Vezje, občutljivo na logične hazarde.

je glede na napetost na njunem vhodu eden odprt, drugi pa zaprt. Ko je vhodna napetost višja od pragovne napetosti n-kanalnega MOS tranzistorja

$$V > V_{Tn} \tag{4.3}$$

je odprt spodnji n-kanalni tranzistor, tako da je sklenjena pot med VSS in izhodom, kar ustreza stanju logične ničle. V primeru, ko pa je napetost

$$V < V_{Tp} \tag{4.4}$$

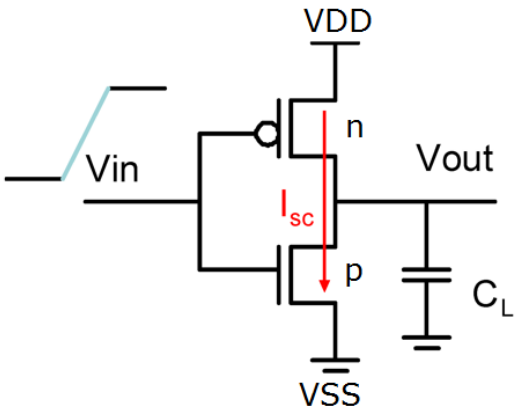
pa je odprt zgornji, p-kanalni tranzistor in povezava med VDD in izhodom, kar ustreza logični enici. Ob prehodu napetosti, ko je ta med pragovnima napetostima, sta sočasno odprta oba tranzistorja in med VDD in VSS steče kratkostični tok.

$$V_{Tp} < V < V_{Tn} \tag{4.5}$$

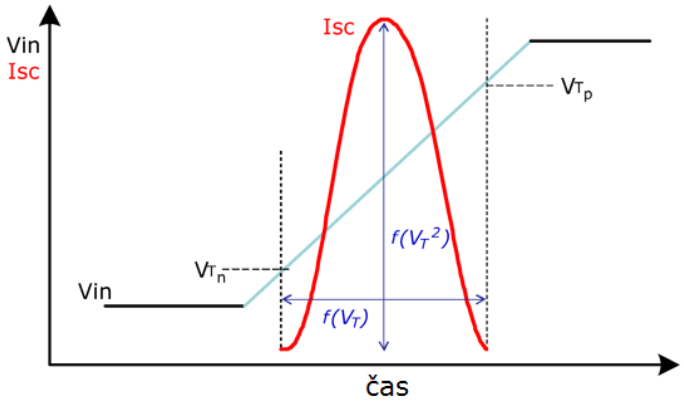
Karakteristiko kratkostičnega toka med prehodom logične ničle v enico ilustriramo na sliki 4.3 [8]. Največji tok teče, ko je vhodna napetost na polovici napajalne. Daljši kot je prehod vhodne napetosti med $V_{Tn}$ in $V_{Tp}$, več energije se zaradi kratkostičnega toka porabi. Hitrejši prehodi porabo energije zmanjšajo, vendar zahtevajo potratnejše tranzistorje MOSFET v izhodih logičnih vrat in tako posledično večjo porabo za njih [3].

## 4.3 Tokovi mirovnega stanja

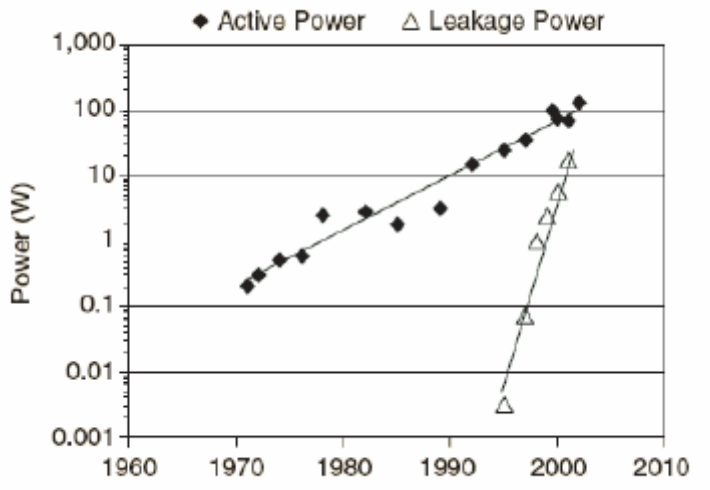
Čeprav je bilo obvladovanje tokov mirovnega stanja v tehnologiji izdelave dinamičnih pomnilnikov neobhodno pomembno že od nekdaj, so bili ti tokovi pri




Slika 4.2: Negator v tehnologiji CMOS.




Slika 4.3: Kratkostični tok v tehnologiji CMOS.




Slika 4.4: Naraščanje deleža statične moči.

izdelavi logičnih vrat CMOS zanemarljivi. Z uvedbo 250-nanometerskih in manjših tehnologij pa so ti tokovi postali pomembnejši [3], njihov vpliv se z vsako novo generacijo integriranih vezij CMOS povečuje, kot je moč razbrati s slike 4.4 [7].

Šibek tok teče tudi, ko je tranzistor v izklopljenem stanju. Ta tok imenujemo podpražno puščanje (*leakage*) in je odvisen od pragovne napetosti. Skaliranje tehnologije zahteva zniževanje napajalne napetosti, tako da se omeji dinamična poraba. Zniževanje napajalne napetosti pa zahteva zniževanje pragovne napetosti ter posledično povečevanje podpragovnih tokov [11]:

$$I_{sub} = KWe^{\frac{-V_{th}}{nT}}\left(1 - e^{\frac{-V_{dd}}{T}}\right) \tag{4.6}$$

V tej enačbi smo z $W$ označili širino vrat, s $K$ in $n$ konstanti, s $T$ temperaturo ter z $V_{th}$ in $V_{dd}$ pragovno in napajalno napetost. Statična poraba moči je torej v nasprotni eksponentni odvisnosti $V_{th}$. Izbira višje pragovne napetosti sicer zniža podpragovne tokove, vendar doseganje višjega napetostnega praga zahteva več časa in sprememba logičnega nivoja se zgodi kasneje. Tako lahko ugotovimo, da bodo visoko pragovni logični elementi, ob isti napetostni karakteristiki, zahtevali močnejše izhode logičnih vrat. Iskanje ustreznih kombinacij visoko in nizko pragovnih logičnih vrat z mislijo na minimalno porabo moči torej zahteva izbiro

elementov z visokim pragom, da bi omejili statično porabo, a hkrati izbiro elementov z nizkim pragom, ko poti preko velikega števila vrat zahtevajo hitrejše preklope.

# Poglavje 5

# Analiza podatkov

Podatke, pridobljene v postopku, opisanem v poglavju 3, smo primerjali znotraj vrste parametra, da bi izluščili njihove posamične vplive.

Definirali smo breme $L$ kot količino procesiranja, ki ga procesor mora opraviti, in ga izrazili z Dhrystone MIPS, tako kot zmogljivost $D$, ki smo jo izmerili s 3.1. Za analizo povprečne porabe moči je uporaba bremena kot neodvisne spremenljivke ključna, saj razmerje bremena in zmogljivosti določa razmerje dolžine časovnih intervalov, v katerih je procesor aktiven $t_a$ oziroma v mirovnem stanju $t_s$.

$$\frac{L}{D} = \frac{t_a}{t_a + t_s} \tag{5.1}$$

Vrste porabe moči razvrstimo glede na stanji mikroprocesorja, v katerih se trošijo. V mirovnem stanju se troši zgolj statična moč $P_l$, v aktivnem pa poleg tega še dinamična moč $P_d$ ter moč, ki izvira iz kratkostičnih tokov $P_s$.

Povprečno moč $P$ torej lahko izrazimo kot seštevek prispevkov posameznih kategorij moči v deležu, v katerem se trošijo:

$$P = \frac{t_a}{t_a + t_s}(P_d + P_s + P_l) + \frac{t_s}{t_a + t_s}P_l \tag{5.2}$$

$$= \frac{L}{D}(P_d + P_s + P_l) + \left(1 - \frac{L}{D}\right)P_l \tag{5.3}$$

$$= \frac{L}{D}P_a + P_l \tag{5.4}$$

| Frekvenca delovanja | Delež HVT | Delež SVT | Skupna površina logičnih vrat |
|---|---|---|---|
| 25 MHz | 97.1 % | 2.9 % | 41 063 µm$^2$ |
| 50 MHz | 86.6 % | 13.4 % | 44 798 µm$^2$ |
| 75 MHz | 72.1 % | 27.9 % | 52 521 µm$^2$ |

Tabela 5.1: Deleži vrste logičnih vrat v odvisnosti od minimalnega takta.

kjer smo s $P_a$ označili moč, ki se troši v stanju aktivnosti:

$$P_t = P_d + P_s, \tag{5.5}$$

označimo pa še skupno moč $P_t$, kot seštevek vseh vrst moči:

$$P_t = P_d + P_s + P_l. \tag{5.6}$$

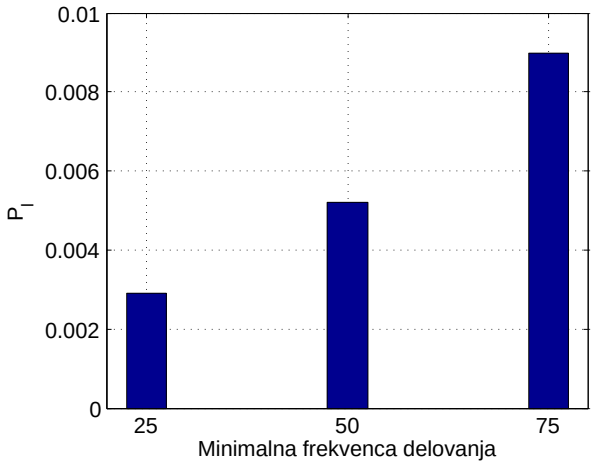
## 5.1 Vpliv frekvence delovanja

Frekvenca delovanja v vgrajenem sistemu je najpomembnejši faktor, ki določa zmogljivost sistema. Pomnilnik je s procesorjem tesno povezan in dostopen v enem urinem ciklu. Vgrajeni enoizstavitveni mikroprocesor BA20 se tako izogne Von Neumanovemu ozkemu grlu in s skaliranjem frekvence dosežemo linearno povečanje zmogljivosti.

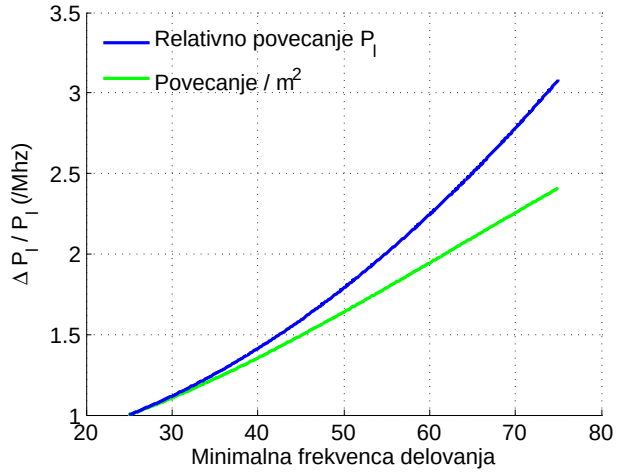
Pregled poročil uporabljenih logičnih vrat potrjuje našo domnevo, da imajo konfiguracije, ki smo jih implementirali z višjim taktom delovanja, večji delež logičnih vrat knjižnice s standardnim napetostnim pragom, kot navajamo v tabeli 5.1.

S povečevanjem deleža logičnih vrat SVT se povečuje delež statične moči (slika 5.1), ki ni odvisna od števila preklopov. Del povečane porabe pripada večjemu številu logičnih vrat oziroma večji skupni površini, ki jo logična vrata zasedajo, vendar slika 5.2 ilustrira, da pretežni del povečanja prispeva višji delež logičnih vrat SVT.

Povečanje porabe, ki izvira iz dinamične in kratkostične moči enake logične strukture vrat (4.1), je sorazmerno številu preklopov v vezju, vendar se implementacije procesorja na višjih frekvencah razlikujejo (tabela 5.1). Povečanje površine




Slika 5.1: Statična moč konfiguracij, implementiranih z različnimi minimalnimi frekvencami delovanja (mW).




Slika 5.2: Povečanje statične porabe moči v odvisnosti od minimalne frekvence delovanja ter povečanje, normirano s skupno površino logičnih vrat.

| $f$ | $P_d$ | $P_s$ | $P_a$ | $P_l$ |
|---|---|---|---|---|
| 25 | 7.41 µW/MHz | 2.45 µW MHz | 9.87 µW MHz | 2.6 mW |
| 50 | 7.83 µW/MHz | 2.84 µW MHz | 10.67 µW MHz | 4.2 mW |
| 75 | 9.03 µW/MHz | 3.97 µW MHz | 13.00 µW MHz | 9.2 mW |

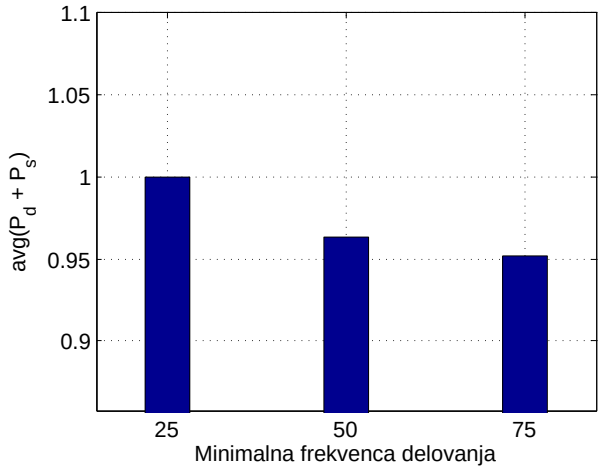
Tabela 5.2: Vrste porabe moči konfiguracij, implementiranih na različnih frekvencah delovanja.

lahko pripišemo kompleksnejšim izvedbam posameznih podstruktur, kot so seštevalniki z vnaprejšnjim izračunom prenosa in potrebi po hitrejših preklopih, ki jih dosežemo z uporabo večjih inačic vrat z močnejšimi izhodi.

Razmerje elementov HVT in SVT vpliva tudi na dinamično porabo, vendar z nasprotnim učinkom kot na statično porabo. Povečan delež vrat knjižnice s standardnim napetostnim preklopnim pragom zmanjša relativni delež dinamične moči, če primerjavo normiramo s skupno površino in frekvenco delovanja. Slika 5.3 ilustrira, da je moč povprečnega preklopa posamičnih vrat v implementacijah procesorja z minimalnim taktom delovanja 75 MHz, približno pet odstotkov nižja kot v 25 MHz implementacijah.

## 5.2 Vpliv funkcijskih enot

Prisotnost množilnika poveča število logičnih vrat, zato je povečanje porabe moči v vseh kategorijah pričakovano, po drugi strani pa množilnik izboljša zmogljivost sistema. Program Dhrystone je izvedel 40 množenj v desetih iteracijah, kar predstavlja 1.2 odstotka vseh izvedenih ukazov. Pospešitev izvajanja programa je 7.5-odstotna. To je več kot je razlika v skupni porabi moči, ki ne presega 4 odstotkov. Tako lahko obstaja procesno breme, pri katerem bi implementacija brez množilnika dosegla povprečno moč implementacije z množilnikom, če le to breme ni večje od zmogljivosti ali manjše od 0. V prvem primeru velja, da prva izmed implementacij vedno troši manj moči, v drugem pa, da je izenačitveno breme večje od zmogljivosti. Takega bremena seveda ni mogoče doseči. To točko lahko izračunamo, če rešimo enačbo, v kateri izenačimo povprečno moč obeh konfiguracij (5.4) in rešimo




Slika 5.3: Primerjava povprečnih dinamičnih moči logičnih elementov. Kot osnovo smo vzeli povprečno moč, ki se troši na logičnih elementih implementacije procesorja pri 25 MHz.

za $L_0$:

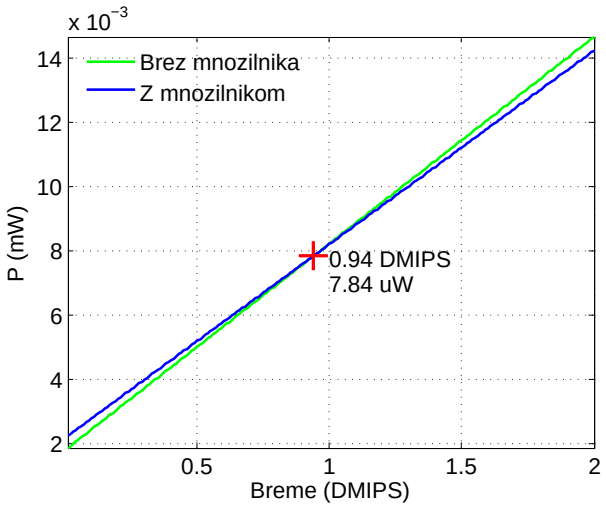
$$0 = \frac{L_0}{D_1} P_{a1} + P_{l1} - \frac{L_0}{D_2} P_{a2} + P_{l2} \tag{5.7}$$

$$L_0 = \frac{(D_1 D_2 (P_{l1} - P_{l2}))}{(D_1 P_{a2} - D_2 P_{a1})} \tag{5.8}$$

oziroma grafično poiščemo presečišče poltrakov, ki opisujeta povprečno moč $P$ ene in druge implementacije, kot je ilustrirano na sliki 5.4.

V tabeli 5.4 podajamo vrednosti izenačitve bremen za nabor implementacij, sestavljenih s 16 in 32 registri ter množilnkom in brez njega ter delilnikom in 25 MHz taktom. Opazimo lahko, da konfiguracije z 32 registri trošijo več moči pri vseh bremenih. Razlika v zmogljivosti ni zadostna, da bi odtehtala višjo statično moč in porabo aktivnega stanja.

Hitrost izvajanja programa Dhrystone z delilnikom je med 60 in 70 odstotkov višja kot brez. V tabeli 5.5 so navedena bremena, do katerih so konfiguracije procesorja brez delilnika močnostno učinkovitejše. Nizke vrednosti so posledica prevladujočega vpliva prisotnosti delilnika na zmogljivost, izmerjeno z programom Dhrystone.




Slika 5.4: Grafično iskanje bremena izenačitve povprečnih moči na primeru implementacij BA20 s taktom delovanja 25 MHz, 16 registri in delilnikom ob izvajanju programa, prevedenega s stikali za povečanje zmogljivosti.

| f | Množilnik | $P_a$ | $P_l$ | $P_t$ | DMIPS/MHz | DMIPS |
|---|---|---|---|---|---|---|
| 25 | ne | 0.23 mW | 1.81 µW | 0.23 mW | 1.45 | 36.2 |
| 25 | da | 0.23 mW | 2.19 µW | 0.24 mW | 1.56 | 38.9 |
| 50 | ne | 0.25 mW | 3.29 µW | 0.25 mW | 1.45 | 72.5 |
| 50 | da | 0.25 mW | 4.28 µW | 0.25 mW | 1.56 | 77.8 |
| 75 | ne | 0.28 mW | 5.32 µW | 0.29 mW | 1.45 | 108.7 |
| 75 | da | 0.29 mW | 7.18 µW | 0.30 mW | 1.56 | 116.8 |

Tabela 5.3: Vpliv množilnika na zmogljivost in porabo moči. Konfiguracije vsebujejo delilnik in 16 splošno-namenskih registrov, zmogljivost pa smo ocenili med izvajajem programa, prevedenega za hitrost.

| Št. reg. | Množ. | $L_0$ | Št. reg | Množ. |
|---|---|---|---|---|
| 16 | ne | 0.94 | 16 | da |
| 16 | ne | $< 0$ | 32 | ne |
| 16 | ne | $< 0$ | 32 | da |
| 16 | da | $< 0$ | 32 | ne |
| 16 | da | $< 0$ | 32 | da |
| 32 | ne | 2.48 | 32 | da |

Tabela 5.4: Največje breme, do katerega je povprečna poraba moči konfiguracij v levih stolpcih manjša od konfiguracije v desnih stolpcih. V vrsticah, v katerih je breme označeno kot $< 0$, konfiguracija na desni troši manj moči pri vseh bremenih. Primerjane konfiguracije so bile implementirane na 25 MHz in vsebujejo delilnik.

| Št. reg. | Množ | $L_0$ |
|---|---|---|
| 16 | ne | 0.11 |
| 16 | da | 0.05 |
| da | ne | 0.09 |
| da | da | 0.08 |

Tabela 5.5: Breme izenačene moči implementacij z in brez delilnika konfiguracij z minimalnim taktom delovanja 25 MHz, med izvajanjem programa, optimiranega za hitrost.

| Kategorija | O3 | Os | Količnik |
|---|---|---|---|
| DMIPS/MHz | 0.65 | 1.56 | 2.39 |
| Velikost zanke | 2354 | 1726 | 1.36 |

Tabela 5.6: Meritve izvajanja programa Dhrystone v O3 in Os. Števila pomnilniških dostopov ne navajamo.

## 5.3 Vpliv prevajalnih optimizacij

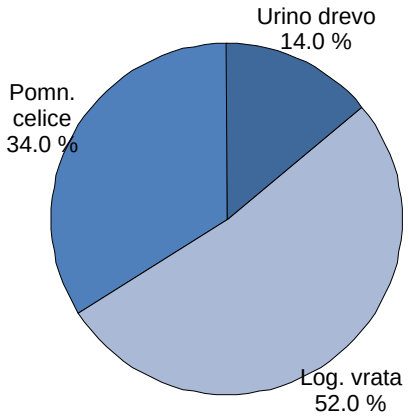
Program Dhrystone smo prevedli z naboroma stikal, navedenih v tabeli 3.2. Ob tem omenimo, da smo upoštevali t.i. "ground-rules" - set pravil prevajanja programa Dhrystone. Ta pravila zahtevajo ločeno prevajanje modulov z izvorno kodo ter prepovedujejo združevanje procedur in optimizacijo v času povezovanja. Prepoved združevanja procedur zaobjema prepoved vnaprejšnjega izračuna klicev funkcij ob prevajanju ter vstavljanje funkcij [12, 14]. Prepoved optimizacij ob povezovanju pa preprečuje, da bi povezovalnik optimiral alokacije registrov ali celo predvidel rezultate funkcijskih klicev med ločeno prevedenima moduloma [12, 16].

Vpliv prevajalnih optimizacij vpliva na eni strani na število pomnilniških lokacij, ki jih program zaseda, na število pomnilniških dostopov ter zmogljivost in s tem čas, v katerem je procesor v aktivnem stanju. Izvajanje programa Dhrystone z obema naboroma optimizacij navajamo v tabeli 5.6. Števila dostopov do pomnilnika v tabeli ne navajamo, zapišemo lahko le, da program, preveden z optimizacijami velikosti, v povprečju 2- do 3-krat več dostopa do pomnilnika (dinamična poraba moči), kar ustreza količniku upočasnitve. Povprečna dinamična pomnilniška poraba moči programa z optimirano velikostjo je tako pri znanem bremenu 2- do 3-krat višja od porabe programa, prevedenega za hitrost.

Vpliv različnih stikal na velikost prevedenih programov v arhitekturi x86 si bralec lahko ogleda v [10].

## 5.4 Pregled porabe po vrstah elementov

Velik del porabe moči v vseh konfiguracijah pripada pomnilnim celicam (slika 5.5). Najizraziteje na število pomnilnih celic vpliva izbira števila splošno-namenskih




Slika 5.5: Dinamična poraba vrst logičnih celic.

registrov. Registrski set s 16 registri vsebuje 480, set z 32 registri pa 992 pomnilnih celic, kar v konfiguracijah brez množilnika in delilnika predstavlja 27.3 in 32.6 odstotkov površine. Povečanje površine, ki jo zasedajo pomnilne celice, je 50.1-odstotno. Primerjava porabe moči pomnilnih elementov razkrije, da je povečanje porabe aktivnega stanja zgolj 8.4-odstotna, medtem ko statična poraba naraste za 47.0 odstotkov. To očitno neravnovesje lahko razložimo, če upoštevamo, da je:

- razlika v zmogljivosti 16- in 32-registrskih konfiguracij zgolj 3.1-odstotna, kar implicira majhno izkoriščenost povečanega nabora registrov. Majhna izkoriščenost registrov pa pomeni malo sprememb stanja in tako majhno povečanje dinamične moči;
- ter, da je razlika v površini, ki jo te celice zasedajo, 50.1-odstotna, statična moč pa ni odvisna od preklopov.

Povečano število registrov zahteva tudi večje urino drevo. Tudi tu opazimo podoben pojav: poraba statične moči urinega drevesa naraste za 33.5 odstotkov, dinamična poraba pa zgolj 10.1 odstotek. Argumenta majhne izkoriščenosti povečanega seta registrov tu ne moremo uporabiti neposredno, če ne upoštevamo metod, ki omejujejo razširjanje urinega signala (Clock Gating). Orodje na povezave urinega signala umesti logične elemente, kot so IN vrata in zapahi (*clock gate*) [4], na en vhod poveže urin signal, na drugega pa signal za omogočanje izhoda. Poleg tega orodje spremeni logiko, ki nadzira spremembe stanj pomnilnih celic, tako da


(a) Brez vrat na urinem signalu. (b) Z vrati na urinem signalu.

Slika 5.6: Zapiranje urinega signala.

jih združi v skupine, ter jih poveže na urin signal, ki izvira iz urinih vrat. Ko sprememba stanja pomnilnih elementov ni potrebna (slika 5.6), se celotni skupini odvzame urin signal, tako da se zapro urina vrata. Na ta način je število vhodov in obseg povezav, na katerih se na urinem signalu dogajajo prehodi, manjši, kar vodi do manjše dinamične porabe moči. Statična poraba moči pa se seveda poveča zaradi večjega urinega drevesa.

# Poglavje 6

# Sklepne ugotovitve in omejitve analize

Izbira majhnega in na prevajalne možnosti močno občutljivega programa Dhyrstone omeji empirične izsledke naših meritev na izvajanje ravno tega programa. Prispevki posameznih funkcijskih enot k zmogljivosti se v primerih praktičnih programov močno razlikujejo, tako lahko na primer pričakujemo večjo občutljivost na prisotnost množilnika v programih, ki obdelujejo digitalne signale.

Zmogljivost procesorja med izvajanjem specifičnega programa lahko izmerimo na podlagi simulacij funkcijskega modela ali celo emulatorja, ki upošteva zgradbo mikroprocesorja. Taka ocena zmogljivosti ne zahteva predhodne implementacije procesorja v elemente knjižnic integriranih vezij in je tako cenejša ter procesno in velikostno veliko manj zahtevna. V ilustracijo naj dodamo, da datoteke s preklopno aktivnostjo izvajanja zgolj desetih iteracij Dhrystone zasedejo več sto megabajtov prostora na disku, implementacije pa trajajo več ur.

V analizi pridobljene podatke o porabi moči sedaj lahko združimo s funkcijsko simulacijo izmerjeno zmogljivostjo ter ponovimo ocene, ki smo jih opravili v poglavju 5. V analizi se nismo dotaknili možnosti izklopa napajanja delov vezja, ki ne vsebujejo arhitekturnega stanja. Tak pristop v veliki meri zmanjša vpliv porabe statične moči na povprečno moč.

Kljub naštetemu lahko za mikroprocesor zapišemo nekatere ugotovitve - v večini konfiguracij se je prisotnost polnega 32-registrskega seta izkazala kot neučinkovita

ne glede na breme. Statična poraba takih konfiguracij je toliko višja, da je minorno povečana zmogljivost ne odtehta. Zdi se, da bi bilo smiselno raziskati konfiguracije s še manj registri. Za prisotnost množilnika in delilnika, čeprav je vpliv slednjega v programu Dhrystone preizrazit, pa lahko zapišemo, da je koristna že ob majhnih bremenih.

# Literatura


[1] J. D. Alexander, "Simulation Based Power Estimation For Digital CMOS", M.S. thesis, Aub. Univ. Auburn, Alabama, USA, 2003, str. 19-24.

[2] H. I. A. Chen, E. K. W. Loo, J. B. Kuo, M. J. Syrzycki, "Triple-Threshold Static Power Minimization in High-Level Synthesis of VLSI CMOS", *Lecture Notes on Computer Science*, Vol. 4464, str. 453-462, 2007.

[3] H. Kaeslin, *Digital Integrated Circuit Design*, New Delhi: Cambridge University Press, 2008, str. 459-471.

[4] J. Kathuria, M. Ayoubkhan, A. Noor, "A Review of Clock Gating Techniques", *MIT International Journal of Electronic and Communication Engineering*, Vol. 1, No. 2, 2011, str. 106-114.

[5] J. Lu, B. Taskin, "Post-CTS Delay Insertion", VLSI Design, Vol. 2010, str. 1-6, 2010.

[6] G. E. Moore, "Cramming more components onto integrated circuits", Electronics, zv. 38, št. 8, str. 2, 1965.

[7] G. E. Moore, "No exponential is forever: but "Forever" can be delayed!", *IEEE Int. conf. Solid State Circuits*, str. 20-23, 2003.

[8] A. Morgenshtein, "Short-Circuit Power Reduction by Using High-Threshold Transistors", *Journal of Low Power Electronics and Applications*, Vol. 2, Issue 2, 2012, str. 1.

[9] L. Scheffer, L. Lavagno, G. Martin, *EDA for IC Implementation, Circuit Design, and Process Technology*, Boca Raton: CRC Press, Taylor & Francis Group, 2006, pogl. 2, 5, 8.

[10] E. Stupache (2013). *GCC x86 code size optimizations*, Dostopno na: https://software.intel.com/en-us/blogs/2013/01/17/x86-gcc-code-size-optimizations

[11] V. Venkatachalam, M. Franz, "Power Reduction Techniques For Microprocessor Systems", *ACM Computing Surveys*, Vol. 37, No. 3, str. 198-205, 2005.

[12] R. P. Weicker, "Dhrystone Benchmark: Rationale for Version 2 and Measurement Rules", *SIGPLAN Notices*, Vol. 23, Issue 8, 1988, str. 49-62.

[13] A. R. Weiss (2002), "Dhrystone Benchmark: History, Analysis, Ščores"and Recomendations", White Paper, EEMBC Certification Laboratories (ECL), LLC, str. 2, 2002.

[14] *Cpplib Internals*, GNU Press, a division of Free Software Foundation, Boston, MA, USA, 2014, str. 11-13.

[15] *Low Power Implementation using RC*, Cadence Design Systems, San Jose, CA 95134, USA.

[16] *Using the GNU Compiler Collection*, GNU Press, a division of Free Software Foundation, Boston, MA, USA, 2014, str. 117-136.

[17] (2014) Cortex M0-Processor. Dostopno na: www.arm.com/products/processors/cortex-m/cortex-m0.php

[18] (2014) Beyond Architectures. Dostopno na: http://www.beyondsemi.com/21/architectures/

[19] (2014) Embedded Processor Cores. Dostopno na: http://www.beyondsemi.com/24/embedded-processor-cores/

[20] (2006) Voltagestorm Power and Power Rail Verification. Dostopno na: http://www.cadence.com/rl/Resources/datasheets/voltage_storm_ds.pdf

[21] (2012) SoC Encounter RTL-to-GDSII System. Dostopno na: http://www.cadence.com/rl/Resources/datasheets/soc_encounter_ds.pdf