УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

РАДОМИР М. ЈАКОВЉЕВИЋ

ПАРАЛЕЛНИ МЕМОРИЈСКИ ПОДСИСТЕМИ ЗА ПРИМЕНУ У ОБРАДИ СЛИКЕ И ВИДЕА У МОБИЛНИМ УРЕЂАЈИМА

ДОКТОРСКА ДИСЕРТАЦИЈА

БЕОГРАД, 2015.

UNIVERSITY OF BELGRADE
SCHOOL OF ELECTRICAL ENGINEERING

RADOMIR M. JAKOVLJEVIĆ

PARALLEL MEMORY SUBSYSTEMS FOR IMAGE AND VIDEO PROCESSING ON MOBILE DEVICES

DOCTORAL DISSERTATION

BELGRADE, 2015

Ментор:

др Драган Милићев, редовни професор

Универзитет у Београду – Електротехнички факултет

Чланови комисије:

др Драган Милићев, редовни професор

Универзитет у Београду – Електротехнички факултет

др Зоран Јовановић, редовни професор

Универзитет у Београду – Електротехнички факултет

др Душан Старчевић, редовни професор

Универзитет у Београду – Факултет организационих наука

др Лазар Сарановац, ванредни професор

Универзитет у Београду – Електротехнички факултет

др Мило Томашевић, редовни професор

Универзитет у Београду – Електротехнички факултет

Датум одбране: _____ год.

Овај рад посвећујем супрузи Драгани и нашој кћерки Анки.

ЗАХВАЛНИЦЕ

Најпре бих желео да се захвалим свом ментору, проф. др Драгану Милићеву за изванредну сарадњу током свих година мојих докторских студија, беспрекорно вођење вишегодишњег истраживачког пројекта у сарадњи факултета са компанијом „Силикон Хајв", као и помоћ током писања научних радова и ове дисертације. Истакао бих изузетну ажурност и велику брзину професора Милићева, како приликом прегледа и рецензије текста ове дисертације и научних радова, тако и при извршавању свих административних обавеза које је имао као ментор. Велику захвалност осећам и према проф. др Лазару Сарановцу, који ми је несебично помагао и давао веома корисне савете за време докторских студија, а посебно током писања научног рада објављеног у часопису, за чији квалитет је професор Сарановац у великој мери заслужан. На прегледу и оцени ове дисертације захвалан сам проф. др Зорану Јовановићу, проф. др Душану Старчевићу и проф. др Милу Томашевићу.

Успеху истраживања које стоји иза ове дисертације у великој мери допринела је сарадња са компанијама „Силикон Хајв" и „Интел", односно њихови запослени који су истраживање омогућили и помагали. Стога бих желео да се захвалим Александру Берићу, свом дугогодишњем ментору из индустрије, на помоћи током истраживања и развоја техничког решења представљеног у дисертацији, пријатном времену проведеном на разним деловима света, као и усмеравању и сталном подстицању да напредујем у професионалном смислу и каријери. Поред Александра, захвалност дугујем и великом броју колега из „Силикон Хајва" и „Интела", а посебно онима из некадашњег српског огранка ових компанија.

За постојање ове дисертације заслужна је и моја супруга Драгана, којој сам неизмерно захвалан за огромну подршку, стрпљење и разумевање за време које нисам провео са њом током докторских студија, а посебно током писања научних радова и дисертације. Захвалност за разумевање дугујем и свим осталим мени драгим људима, са којима нисам провео времена колико сам желео у последњих неколико година.

У Београду, 2015. године Радомир Јаковљевић

ПАРАЛЕЛНИ МЕМОРИЈСКИ ПОДСИСТЕМИ ЗА ПРИМЕНУ У ОБРАДИ СЛИКЕ И ВИДЕА У МОБИЛНИМ УРЕЂАЈИМА

Радомир М. Јаковљевић

РЕЗИМЕ

Добро је познато да уско грло векторских процесора за дигиталну обраду слике и видеа, које ограничава њихове перформансе, јесте приступ пикселима у меморији. Ова теза предлаже ново решење паралелног меморијског подсистема на чипу, што укључује нове функционалности и одговарајућу архитектуру, које омогућава већу брзину обраде на оваквим процесорима и троши мање енергије при обради истог броја пиксела од најефикаснијих постојећих подсистема.

У тези су најпре представљене нове функционалности паралелног меморијског подсистема, односно нови начини приступа блоку и реду пиксела, који су боље прилагођени савременим методама обраде слике и видеа него функционалности постојећих подсистема. Њиховом применом значајно се смањује укупан број операција читања и уписа у паралелни меморијски подсистем и на тај начин се убрзавају примитиве обраде које су од интереса у овом раду: естимација кретања суб-пиксел упаривањем блокова, интерполација пиксела у компензацији кретања и филтрирање у просторном домену применом прозорских функција. Основна идеја нових функционалности јесте да се искористи унапред познато просторно преклапање блокова и редова пиксела којима се приступа у меморији и да се више операција читања из меморијског подсистема споји у једну, тако што се у паралели прочита нешто већи број пиксела него код постојећих паралелних подсистема. Да би се избегла потреба за широм, а самим тим и скупљом путањом података векторског процесора, предложене су нове операције читања из меморијског подсистема које деле тај већи број пиксела, прочитаних у паралели, на више блокова или редова ширине постојеће путање података, на начин погодан за даљу обраду. Поред убрзања обраде, применом нових начина

приступа и нових операција читања смањује се утрошена енергија меморијског подсистема за исту обраду, јер се смањује број приступа истим пикселима у подсистему.

Као доказ изводљивости нових идеја и у циљу комплетности предложеног решења, описана је параметризована, скалабилна и економична архитектура која реализује нове и функционалности постојећих паралелних меморијских подсистема. Архитектура је заснована на: 1) скупу од више паралелних меморијских модула, односно банака, таквих да једна адресибилна реч једне банке складишти више пиксела, 2) искошеном обрасцу распоређивања пиксела у банке који омогућава приступ свим пикселима једног блока или реда у паралели и 3) управљачкој логици која реализује искошени образац распоређивања пиксела у банке и нове начине приступа, омогућавајући на тај начин ефикасно и једноставно програмирање приступа меморијском подсистему. Коришћени скуп банака и искошени образац распоређивања пиксела у банке познати су из раније објављених радова у литератури.

Предности предложеног решења паралелног меморијског подсистема демонстриране су аналитички и експериментално на примеру студије случаја реализације познате 3DRS методе за естимацију кретања суб-пиксел упаривањем блокова. Ова метода изабрана је као репрезентативна за студију случаја јер се веома често примењује за повећање броја фрејмова у секунди видеа, у реалном времену при његовој репродукцији. Реализовани естиматор кретања, заснован на предложеном паралелном меморијском подсистему, може да обрађује видео резолуције 3840∗2160 пиксела брзином од 60 фрејмова у секунди, радећи на такту од 600 мегахерца. У поређењу са реализацијама заснованим на других шест подсистема, који су у време овог истраживања били на највишем степену развоја и истој векторској путањи података процесора, предложено решење омогућава од 40 до 70 процената већу брзину обраде, трошећи притом од 17 до 44 процента мање енергије, а уз слично заузеће површине на чипу и исти број приступа меморији ван чипа. Тиме је постигнута од 1,8 до 2,9 пута већа ефикасност од других подсистема, израчуната као количник брзине обраде и производа свих цена подсистема: потрошње енергије, заузећа површине на чипу и броја приступа меморији ван чипа. Овако висока ефикасност резултат је примене нових начина приступа и економичне архитектуре, којима је број приступа паралелном меморијском подсистему на чипу смањен од 1,6 до 2,1 пута.

Захваљујући доказано високој ефикасности, предложено решење паралелног меморијског подсистема примењује се у најсавременијим Интеловим процесорима и системима на чипу за обраду слике и видеа у мобилним уређајима. Истовремено, настављен је развој предложеног подсистема ради примене у наредним генерацијама Интелових процесора.

Кључне речи
функционалност паралелног меморијског подсистема, архитектура паралелног меморијског подсистема, векторски процесори, обрада слике, обрада видеа, упаривање блокова, естимација кретања, компензација кретања, интерполација пиксела, филтрирање у просторном домену

Научна област: Електротехника и рачунарство
Ужа научна област: Рачунарска техника и информатика
УДК број: 621.3:004.272

PARALLEL MEMORY SUBSYSTEMS FOR IMAGE AND VIDEO PROCESSING ON MOBILE DEVICES

Radomir M. Jakovljević

ABSTRACT

Accessing pixels in memory is a well-known performance bottleneck of SIMD (Single-Instruction Multiple-Data) processors for image and video processing. This thesis proposes a new solution of a parallel on-chip memory subsystem, including new functionalities and an enabling architecture, which enables a higher processing throughput and consumes less energy per processed pixel than the other state-of-the-art subsystems.

The thesis first presents new functionalities of a parallel memory subsystem, i.e. new block and row access modes, which are better adjusted to the needs of image and video processing algorithms than the functionalities of existing parallel memory subsystems. The new access modes significantly reduce the number of on-chip memory read and write accesses, and thereby accelerate the imaging/video kernels that are in focus of this work: sub-pixel block-matching motion estimation, pixel interpolation for motion compensation, and spatial window-based filtering. The main idea of the new access modes is to exploit spatial overlaps of blocks/rows accessed in the memory subsystem, which are known at the subsystem design-time, and merge multiple accesses into a single one by accessing somewhat more pixels at a time than with other parallel memories. To avoid the need for a wider, and therefore more costly SIMD datapath, this work proposes new memory read operations that split all pixels accessed at a time into multiple SIMD-wide blocks/rows, in a convenient way for further processing. In addition to a higher processing throughput, the new access modes reduce the energy consumed by the parallel memory subsystem for the same amount of processed pixels, by reducing the number of repeated accesses of the same pixels.

Second, as a proof of concept and for the completness of the proposed solution, the thesis describes a parametric, scalable, and cost-efficient architecture that supports the new access modes, as well as access modes of existing parallel memory subsystems. The architecture is based on: 1) a previously proposed set of memory banks with multiple pixels per addressable word of a bank, 2) a previously proposed shifted scheme for arranging pixels in the banks that enables parallel access to all pixels of a block/row, and 3) control logic that implements the shifted scheme in the memory banks and the access modes, and thus provides a convenient and efficient programming interface.

The advantages of the proposed solution of a parallel memory subsystem are analytically and experimentally demonstrated on a case study of well-known 3DRS sub-pixel block-matching motion estimation algorithm, commonly used for real-time frame-rate conversion during video playback. The implemented block-matcher, based on the proposed parallel memory subsystem, is able to process 3840∗2160 video at the rate of 60 frames per second, while clocked at 600 MHz. Compared to the block-matcher implementations based on the six state-of-the-art subsystems and the same SIMD datapath, this work enables 40 – 70% higher throughput, consumes 17 – 44% less energy, and has similar silicon area and off-chip memory bandwidth costs. That is 1.8 – 2.9 times more efficient than the prior state-of-the-art, calculated as the ratio between the achieved processing throughput and the product of all costs: energy consumption, silicon area, and off-chip memory bandwidth. Such a high efficiency is the result of the new access modes and the cost-effective architecture, which reduced the number of on-chip memory accesses by 1.6 – 2.1 times.

Thanks to its proven efficiency, the proposed parallel memory subsystem is used in the most recent and most advanced Intel processors and systems-on-chip for imaging and video processing on mobile devices. Furthermore, it is being continuously improved for use in future generations of Intel processors.

Keywords
functionality of parallel memory subsystem, architecture of parallel memory subsystem, SIMD processors, image processing, video processing, block-matching, motion estimation, motion compensation, pixel interpolation, spatial filtering

Scientific field: Electrical and Computer Engineering
Scientific subfield: Computer Science and Engineering
UDC number: 621.3:004.272

КРАТАК САДРЖАЈ

САДРЖАЈ

„Крени, сине, у сребрни поток – да те он до извора доведе."

Из народне бајке

1 УВОД

Слика говори више од хиљаду речи, каже народна изрека. Заиста, човек као визуелно биће одувек има потребу да комуницира путем слике. У почетку је руком цртао на зидовима станишта, дрвету и платну, како би забележио важне догађаје и особе. Много времена касније, у савременом технолошком добу, креирање и прегледање дигиталних слика и видео снимака део је свакодневног живота. Традиционални уређаји за креирање су дигитални фотоапарати и видео камере, а за прегледање персонални рачунари и телевизори. Међутим, у ове сврхе све више се користе савремени мобилни уређаји попут паметних телефона и таблет рачунара илустрованих на слици 1.1. Њихова предност је у комбинацији преносивости, бежичног приступа интернету и разноврсних функција у једном уређају. То кориснику омогућава да креира слике и видео снимке где год да се налази и да их одмах подели са другима преко интернета. Такође, где год се налазио, корисник може да прегледа слике и видео снимке, било да су смештени у меморији уређаја или на интернет сервисима попут Јутјуба, Фејсбука и сличних.

Услед велике популарности мобилних уређаја постоји и потреба за сталним унапређивањем њихових функција везаних за слику и видео. Како су ове функције засноване на дигиталној обради слике и видеа, потребно је, са једне стране,

Слика 1.1 – Савремени мобилни уређаји, таблет (лево) и паметни телефон (десно)

унапређење метода обраде у циљу постизања бољег визуелног квалитета слике и видеа, а са друге, унапређење хардверско-софтверске реализације тих метода да би се убрзала обрада, продужило трајање батерије и смањило загревање уређаја. Овај рад се бави аспектом реализације метода обраде и архитектурама процесора специјализованим за ту намену. Притом, фокус је на меморијском подсистему за податке, који најчешће представља уско грло перформанси и један је од највећих потрошача енергије код таквих процесора.

Савремени процесори за обраду слике и видеа заснивају се на паралелним архитектурама, са циљем да се искористе различите врсте паралелизма у методама обраде и тако постигне већа брзина обраде. Притом се, за мобилне уређаје, примењују оне врсте паралелних архитектура које омогућују и малу потрошњу енергије уз велику брзину обраде. Једна од таквих је векторска архитектура процесора, заснована на функционалним јединицама које једном инструкцијом обрађују низ података у паралели, што омогућава вишеструко убрзање у односу на скаларне процесоре где се једном инструкцијом врши обрада једног податка. Истовремено, потрошња енергије мања је него код скаларних процесора јер се инструкције дохватају и декодују мањи број пута по обрађеном податку. Међутим, поред таквих функционалних јединица, за перформансе векторског процесора кључан је и паралелни меморијски подсистем. Он треба да обезбеђује податке функционалним јединицама на такав начин да се најефикасније искористи могућност паралелне обраде у њима. Паралелни меморијски подсистем кључан је и за потрошњу, јер је меморија један од највећих потрошача енергије у савременим технологијама израде чипова.

Предмет овог рада и претходно обављеног истраживања су функционалности и архитектура паралелног меморијског подсистема за векторске процесоре

намењене обради слике и видеа. Циљ истраживања јесте да се унапређењем функционалности и архитектуре паралелног меморијског подсистема убрза обрада и смањи потрошња енергије таквих процесора, а да се притом не повећа заузеће површине на чипу. Како би се постигао тако комплексан циљ, истраживање је ограничено на неке од кључних примитива у савременој обради слике и видеа: естимацију кретања упаривањем блокова, интерполацију пиксела у компензацији кретања и филтрирање у просторном домену. Наведене примитиве представљају основу и у највећој мери одређују рачунску сложеност у многим применама обраде слике и видеа. Неке од таквих примена су повећање броја фрејмова у секунди и скалирање резолуције видеа у реалном времену при његовој репродукцији, потискивање или редукција шума, повећање динамичког опсега боја, дигитална стабилизација видеа и супер-резолуција.

Приликом истраживања коришћене су следеће научне методе: анализа постојећих функционалности и архитектура паралелног меморијског подсистема са становишта реализације посматраних метода обраде на векторском процесору, затим извођење закључака о недостацима постојећих функционалности и архитектура и на крају синтеза нових функционалности и архитектуре којима се превазилазе ти недостаци. Евалуација предложеног решења извршена је мерењем релевантних параметара. Убрзање обраде мерено је на прецизном симулатору предложеног подсистема и одговарајућег векторског процесора. Потрошња енергије и заузеће површине одређени су реализацијом предложеног подсистема у језику високог нивоа за опис хардвера и његовом синтезом у савременој шездесет пет нанометарској CMOS технологији за израду чипова. Ефикасност предложеног решења потврђена је и његовом применом у више Интелових чипова за мобилне уређаје, доступних на тржишту.

Истраживање је спроведено делом на Електротехничком факултету у Београду, а делом у компанији „Силикон Хајв” из Холандије, која се бави развојем процесора и система на чипу за обраду слике и видеа, а која је током истраживања постала део Интел корпорације. Сарадња са овим компанијама донела је најновије захтеве индустрије у вези са предметом истраживања и омогућила примену предложеног решења у најсавременијим Инteловим процесорима за мобилне уређаје. На тај начин истраживање је добило на значају и подигнуто је на врх таласа технолошког развоја.

У наставку овог поглавља, у секцији 1.1, наведени су најважнији доприноси истраживања. Потом су, у секцији 1.2, набројани објављени научни радови и

патентне апликације у вези са овим истраживањем. Кратак преглед осталих поглавља тезе дат је у секцији 1.3.

1.1 Доприноси истраживања

У овом раду предложено је ново решење паралелног меморијског подсистема на чипу за примену у векторским процесорима слике и видеа. Најважнији доприноси овог рада су:

1. Анализа одабраних метода обраде слике и видеа које се често примењују у пракси. Анализа је извршена са становишта њихове реализације на векторском процесору, а са фокусом на приступе подацима у меморијском подсистему. Анализом су уочена ограничења брзине обраде, као и неефикасности у потрошњи енергије које намећу функционалности постојећих паралелних меморијских подсистема. Ова анализа је започета у поглављу 2, а разрађена и довршена у поглављу 3.

2. Нове функционалности паралелног меморијског подсистема, потребне да се превазиђу уочена ограничења и на тај начин омогуће бржа обрада на векторском процесору и мања потрошња енергије меморијског подсистема. Нове функционалности дефинисане су у поглављу 3.

3. Преглед постојећих решења паралелног меморијског подсистема, који представљају највиши степен развоја[1] у области у време реализације овог истраживања. Преглед је дат у поглављу 4.

4. Опис архитектуре предложеног решења паралеленог меморијског подсистема која реализује нове и функционалности постојећих подсистема. Показано је да за постизање брже обраде и мање потрошње енергије описана архитектура не захтева већу површину на чипу од архитектура других подсистема, што предложено решење чини ефикаснијим у сваком погледу. Резултат тога је његова примена у неким од данас најнапреднијих процесора слике и видеа за мобилне уређаје. Архитектура предложеног решења описана је у поглављу 5.

5. Студија случаја која, прво аналитички, а потом и експериментално, потврђује предности предложеног решења на примеру реализације добро

[1] енг. state-of-the-art

познате и често примењиване 3DRS методе за естимацију кретања субпиксел упаривањем блокова. Студија случаја је дата у поглављу 6.

1.2 Научни радови и патентне апликације

На основу постигнутих резултата овог истраживања објављена су три научна рада, један у врхунском међународном часопису категорије М21 са фактором утицаја 2,02, а два на признатим међународним конференцијама:

[1] **Jakovljević, R.**, Berić, A., van Dalen, E., Milićev, D.: New access modes of parallel memory subsystem for sub-pixel motion estimation, *Journal of Real-Time Image Processing*, pp. 1–18, 2014, http://dx.doi.org/10.1007/s11554-014-0481-3, (**IF=2.02**) (ISSN 1861–8200).

[2] **Jakovljević, R.**, Berić, A.: "N-meander scanning trace a method for the on-chip bandwidth reduction", *Proceedings of the 15th IEEE International Conference on Image Processing*, San Diego, USA, pp. 1404–1407, 2008.

[3] **Jakovljević, R.**, Berić, A.: "A method for improving the efficiency of a two-level memory hierarchy", *Proceedings of IEEE Workshop on Signal Processing Systems*, Washington DC, USA, pp. 37–42, 2008.

Као признање иновативности, ефикасности и практичној применљивости предложеног решења паралелног меморијског подсистема, Патентном заводу Сједињених Америчких Држава поднете су следеће патентне апликације од стране Интел корпорације:

[4] **Jakovljević R.**, Berić A., van Dalen E., Milićev D.: Electronic apparatus having parallel memory banks, *Patent Application WO/2013/106210 A1*, July 18th 2013.

[5] **Jakovljević R.**, Berić A., van Dalen E., Milićev D.: Intelligent parametric scratchap memory architecture, *Patent Application US 20140149657 A1*, May 29th 2014.

У вези са практичном применом резултата овог истраживања, Интел корпорација је поднела следећу патентну апликацију, такође Патентном заводу Сједињених Америчких Држава:

[6] Berić, A., Pantić, Z., Kovačević, V., **Jakovljević, R.**, Marković, M.: Motion estimation using hierarchical phase plane correlation and block matching, *Patent Application WO/2013/793029 A1*, September 11th 2014.

Аутор је одржао више предавања у вези са облашћу истраживања на Електротехничком факултету у Београду, у компанијама „Силикон Хајв" и „Интел", Истраживачкој станици „Петница", као и на домаћим и међународним научним конференцијама.

1.3 Преглед рада

Поглавље 2 прво наводи изазове у савременим применама обраде слике и видеа, а затим описује контекст и дефинише проблем истраживања. На крају поглавља, након дефинисања проблема, још једном су наведени најважнији доприноси овог рада, али са знатно више детаља него у секцији 1.1.

Поглавље 3 описује методе обраде слике и видеа које су од интереса у овом раду, анализирајући њихову реализацију на векторском процесору, а посебно приступе паралелном меморијском подсистему. На основу ове анализе, дефинишу се функционалности паралелног меморијског подсистема које су потребне за ефикасну реализацију посматраних метода. Неке од дефинисаних функционалности су познате и подржане од стране постојећих паралелних меморијских подсистема, а неке су нове и представљају најважнији иновативни допринос овог рада.

Поглавље 4 даје преглед постојећих решења паралелног меморијског подсистема, анализирајући њихове функционалности и архитектуру у циљу утврђивања предности и недостатака са становишта реализације посматраних метода обраде слике и видеа.

Поглавље 5 детаљно описује све аспекте архитектуре предложеног решења паралелног меморијског подсистема, која реализује нове и постојеће функционалности дефинисане у поглављима 2 и 3.

Поглавље 6 аналитички и експериментално евалуира предложено решење, дајући комплетну студију случаја реализације суб-пиксел упаривања блокова. Евалуација укључује поређење са шест других паралелних меморијских подсистема који тренутно представљају највиши степен развоја у овој области.

Поглавље 7 закључује рад и даје смернице за даља истраживања у области паралелних меморијских подсистема за обраду слике и видеа.

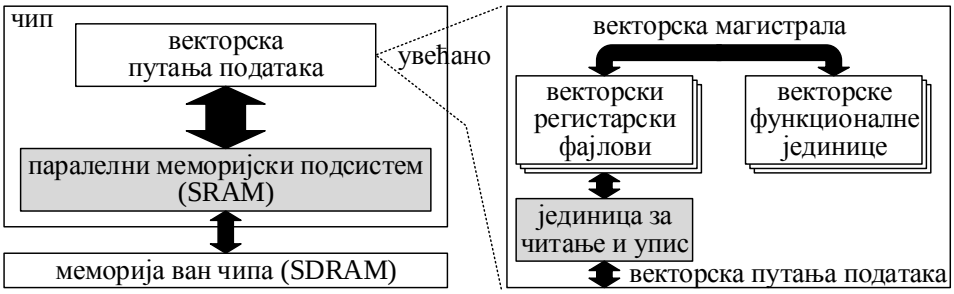
2

ДЕФИНИЦИЈА ПРОБЛЕМА

Ово поглавље дефинише проблем истраживања. Прво су, у секцији 2.1, наведени савремени трендови у применама обраде слике и видеа, који реализацију те обраде чине рачунски све сложенијом и захтевају употребу паралелних и енергетски ефикасних архитектура процесора, као што је векторска. Потом је, у истој секцији, образложена потреба за ефикасним паралелним меморијским подсистемом за примену у векторским процесорима. Недостаци постојећих решења паралелног меморијског подсистема, којима се овај рад бави, описани су у секцији 2.2. На крају, у секцији 2.3, наведени су доприноси овог рада са више детаља него у уводном поглављу. Наведени доприноси представљају основу решења паралелног меморијског подсистема предложеног у овом раду, којима се превазилазе описани недостаци постојећих решења.

2.1 Потреба за ефикасним паралелним меморијским подсистемом

Стални тренд у применама дигиталне слике и видеа јесте повећање њихове резолуције и броја фрејмова у секунди, како би се постигао бољи визуелни угођај корисника. Овај тренд додатно је убрзан са порастом популарности мо-



Слика 2.1 – Паралелни меморијски подсистем на чипу.

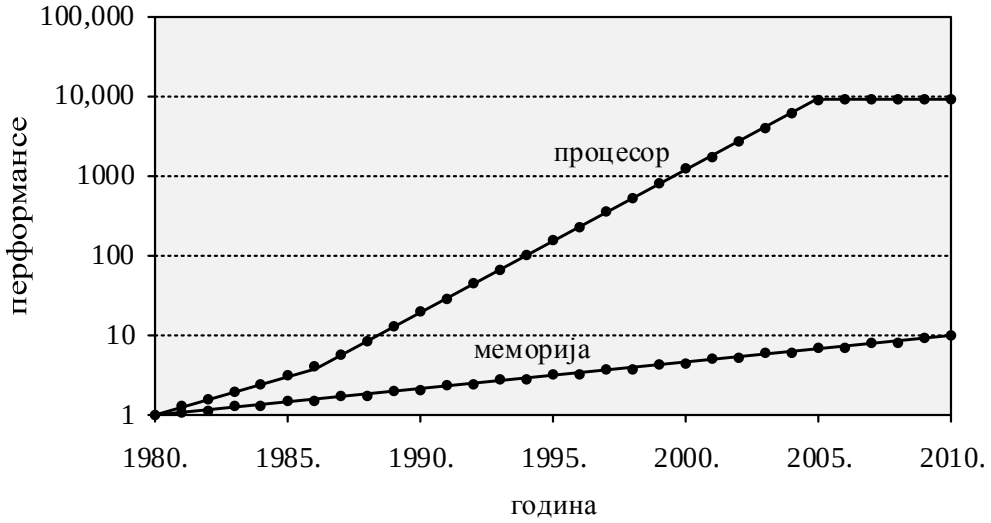
билних уређаја и брзим развојем технологије екрана. На пример, наредна, а чак и текућа генерација мобилних уређаја треба да омогући снимање и репродукцију видеа резолуције 1920∗1080 или 3840∗2160[1] пиксела брзином од 30 или 60 фрејмова у секунди, као и креирање од 15 до 30 слика у секунди резолуције 13, 16, 24 или више мегапиксела. Истовремено, потребно је да визуелни квалитет слике и видеа буде све бољи, трајање батерије дуже, а загревање уређаја мање. Да би се постигао најбољи визуелни квалитет користе се софистициране и рачунски сложене методе за естимацију и компензацију кретања, упаривање блокова, интерполацију пиксела и филтрирање у просторном домену, а у применама као што су повећање броја фрејмова у секунди и скалирање резолуције видеа у реалном времену при његовој репродукцији, компресија и кодовање видеа, потискивање шума, изоштравање слике, повећање динамичког опсега боја, дигитална стабилизација видеа и супер-резолуција [1–9]. Узимајући све наведене чиниоце у обзир, процесор слике и видеа треба да омогући брзину, односно пропусност обраде, од неколико гигапиксела у секунди и више, уз потрошњу која не прелази неколико стотина миливата [10, 11].

Како би се остварила овако велика брзина обраде и мала потрошња, односно како би енергетска ефикасност обраде била веома висока, тежи се искоришћењу свих врста паралелизама у методама обраде. За паралелизам на новоу података користи се векторска архитектура процесора[2]. Она се заснива на путањи података[3], што је скуп векторских функционалних јединица, регистара и магистрала, која је способна да једном инструкцијом обради низ од N пиксела у паралели, нпр. 16, 32, 64 или више. Међутим, уз одговарајућу векторску

[1] 1920∗1080 и 3840∗2160 су ознаке за формате видео фрејма ширине 1920 и 3840 пиксела, а висине 1080 и 2160 линија без прореда, респективно.

[2] енг. Single-Instruction-Multiple-Data (SIMD)

[3] енг. datapath



Слика 2.2 – Јаз између перформанси процесора и меморије ван чипа (извор [12]).

путању података, за велику брзину обраде неопходан је и ефикасан паралелни меморијски подсистем на чипу, чија је позиција у систему приказана на слици 2.1. На овој слици SRAM означава статичку меморију са случајним приступом, а SDRAM синхрону динамичку меморију са случајним приступом.

Улога паралелног меморијског подсистема на чипу јесте да премости добро познати јаз између брзине процесора и меморије ван чипа, односно јаз између брзине којом процесор може да обрађује пикселе и брзине којом меморија те пикселе може да обезбеди процесору [12–16]. Тај јаз илустрован је на слици 2.2. Стога, да би паралелни меморијски подисистем био ефикасан, он треба да:

- Обезбеђује потребне пикселе векторским функционалним јединицима уз минималан број меморијских приступа по обрађеном вектору од N пиксела. Тако се омогућава да степен искоришћења функционалних јединица буде максималан, што је предуслов за максималну брзину обраде.

- Минимизује број приступа дељеној меморији ван чипа, како би се ограничила њена потрошња енергије. Такође, једнако важно је да се на тај начин минимизује и интерференција са осталим процесорима и уређајима у систему на чипу који конкурентно приступају истој дељеној меморији ван чипа.

- Буде економичан у потрошњи енергије и заузећу површине на чипу.

Како су прва два захтева у супротности са трећим, реализација оваквог паралелног меморијског подсистема представља велики изазов. Са друге стране,

захваљујући значају проблема, извршен је велики број истраживања и предложени су многобројни паралелни меморијски подсистеми [15–36]. У овом раду предлаже се ново и ефикасније решење паралелног меморијског подсистема, које отклања недостатке постојећих решења описане у наредној секцији. Фокус је на потребним функционалностима, односно начинима приступа меморијском подсистему, као и на економичној архитектури која омогућава потребне начине приступа.

2.2 Недостаци постојећих решења

Недостаци постојећих решења паралелног меморијског подсистема објашњени су на примеру упаривања блокова[4] [1]. Прво се анализира основна функција у упаривању блокова и њена реализација на векторском процесору, затим се наводе основне особине постојећих решења паралелног меморијског подсистема и на крају се описују и илуструју њихови недостаци. Анализа других метода обраде које су од интереса у овом раду дата је у поглављу 3.

У методама које користе упаривање блокова функција која се извршава убедљиво највећи број пута и због тога одређује брзину обраде јесте израчунавање сличности између два блока пиксела, најчешће од 4∗4, 8∗8 или 16∗16 пиксела. Пример једне такве, често коришћене функције је сума апсолутних разлика, у ознаци SAD[5], која је дефинисана следећом формулом:

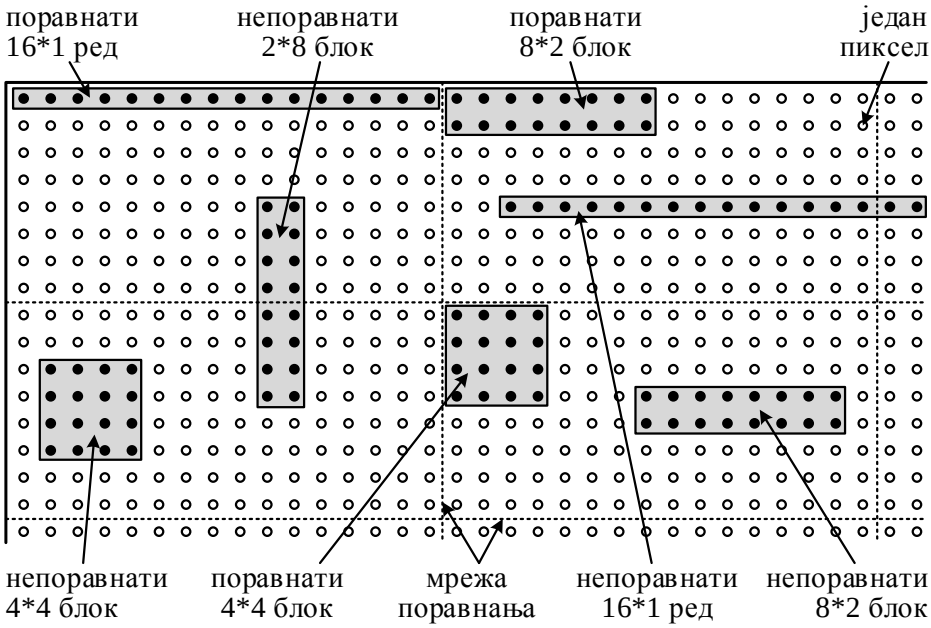
$$SAD(B1, B2) \;=\; \sum_{k=1}^{K} \sum_{m=1}^{M} \mid B1(k,m) \;-\; B2(k,m) \mid . \tag{2.1}$$

У овој формули $B1$ и $B2$ означавају блокове који се упоређују, а који су ширине K пиксела и висине M линија. Притом, $B1(k,m)$ и $B2(k,m)$ представљају вредности пиксела на позицији (k,m) у блоку $B1$ и $B2$, респективно. Из ове формуле може се закључити да су блокови $B1$ и $B2$ сличнији што је SAD вредност мања.

Израчунавање овако дефинисане SAD функције може се извршити у само неколико циклуса такта на стандардној векторској путањи података, као што ће касније бити илустровано. Под стандардном векторском путањом података подразумева се она која има више векторских аритметичко-логичких јединица, тако да може да изврши више векторских операција у паралели, где свака век-

[4] енг. block-matching

[5] енг. sum of absolute differences (SAD)



Слика 2.3 – Начини приступа паралелном меморијском подсистему, прилагођени обради слике и видеа.

торска операција истовремено обрађује 16, 32, 64 или више пиксела. Да би се подржала велика брзина обраде векторске путање података, потребно је да се читање блокова пиксела из меморијског подсистема такође обавља у минималном броју циклуса такта, односно уз минималан број меморијских приступа.

Међу постојећим решењима паралелног меморијског подсистема [15–36], више њих је развијено специјално за примену у обради слике и видеа [24–36], а неколико решења је посебно оптимизовано за методе упаривања блокова [27,33,36]. Како би се минимизовао број приступа меморији потребних за израчунавање једне SAD вредности, најновија решења паралелног меморијског подсистема омогућавају:

- Смештање дводимензионалних низова пиксела, односно делова слика или фрејмова.
- Приступ поднизовима од N суседних пиксела у облику реда или блока различитих пропорција, а на локацији унутар дводимензионалног низа која се дефинише у време извршавања обраде. Притом, за све облике подниза подржани су поравнати и непоравнати приступи[6], на основу локација, односно адреса у оквиру дводимензионалног низа којима се може

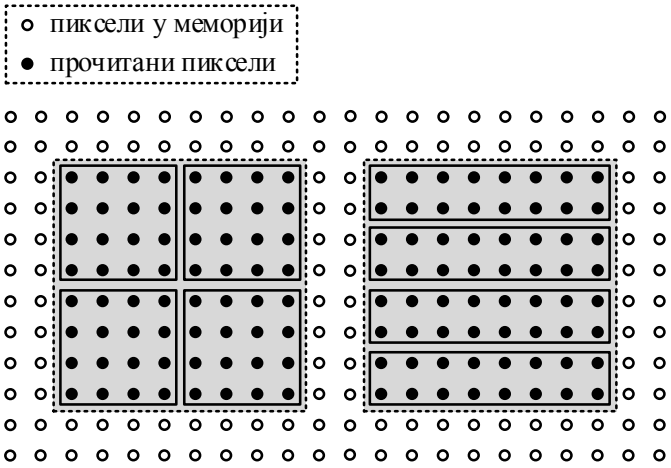
[6] енг. aligned and unaligned accesses

приступити. Поравнати приступи су оганичени тако да морају припадати некој мрежи крупније грануларности, нпр. мрежи блокова пиксела, а неповравнати су са мање ограничења јер припадају мрежи пиксела. Све ове могућности за приступ пикселима, који се у даљем излагању називају **начинима приступа**[7] и који предстаљају функционалност паралелног меморијског подсистема, приказане су на слици 2.3 за случај када је $N = 16$.

Највећи број нових решења, попут оних предложених у радовима [27–35], за реализацију оваквих начина приступа користи N меморијских банака, односно N независних меморијских модула, таквих да једна адресибилна реч једне банке складишти по један пиксел. Због тога се такви подсистеми у даљем тексту називају N-банковним подсистемима. Основна разлика између њих је у специфичним обрасцима за распоређивање пиксела у банке, који се користе како би се осигурало да се у паралели може приступити свим пикселима једног подниза, односно блока или реда на произвољној локацији у дводимензионалном низу. Како би се омогућио приступ поднизовима у облику дијагонала од N пиксела, додатно уз приступе блоку и реду пиксела, подсистем предложен у раду [23] користи $2 * N$ банака са једним пикселом по једној речи банке и стога се у овом раду назива 2N-банковним подсистемом. $2 * N$ банака је коришћено због погодности реализације управљачке логике, иако то није минималан број банака потребан да се омогуће наведени начини приступа [17]. Подсистем назван „BilisPM" и предложен у [26] користи N меморијских банака са два пиксела по једној речи банке и омогућава приступ реду, блоку и колони пиксела. Са друге стране, решење представљено у публикацији [36] засновано је на два нивоа меморијске хијерархије на чипу. Нулти ниво хијерархије, означен са L0 и најближи векторској путањи података, користи мање од N банака, таквих да једна реч једне банке складишти више пиксела, а такође примењује и специфични образац распоређивања пиксела у банке. Предложена реализација L0 нивоа меморије омогућава читање непоравнатих и упис поравнатих блокова од N пиксела. Улога првог нивоа хијерархије, означеног са L1, јесте да обезбеди да број приступа меморији ван чипа буде близу теоријског минимума – један приступ по пикселу у меморији ван чипа. Детаљнији преглед ових и других постојећих решења паралелног меморијског подсистема дат је у поглављу 4.

Ова решења паралелног меморијског подсистема максимално су ефикасна за

[7] енг. access modes

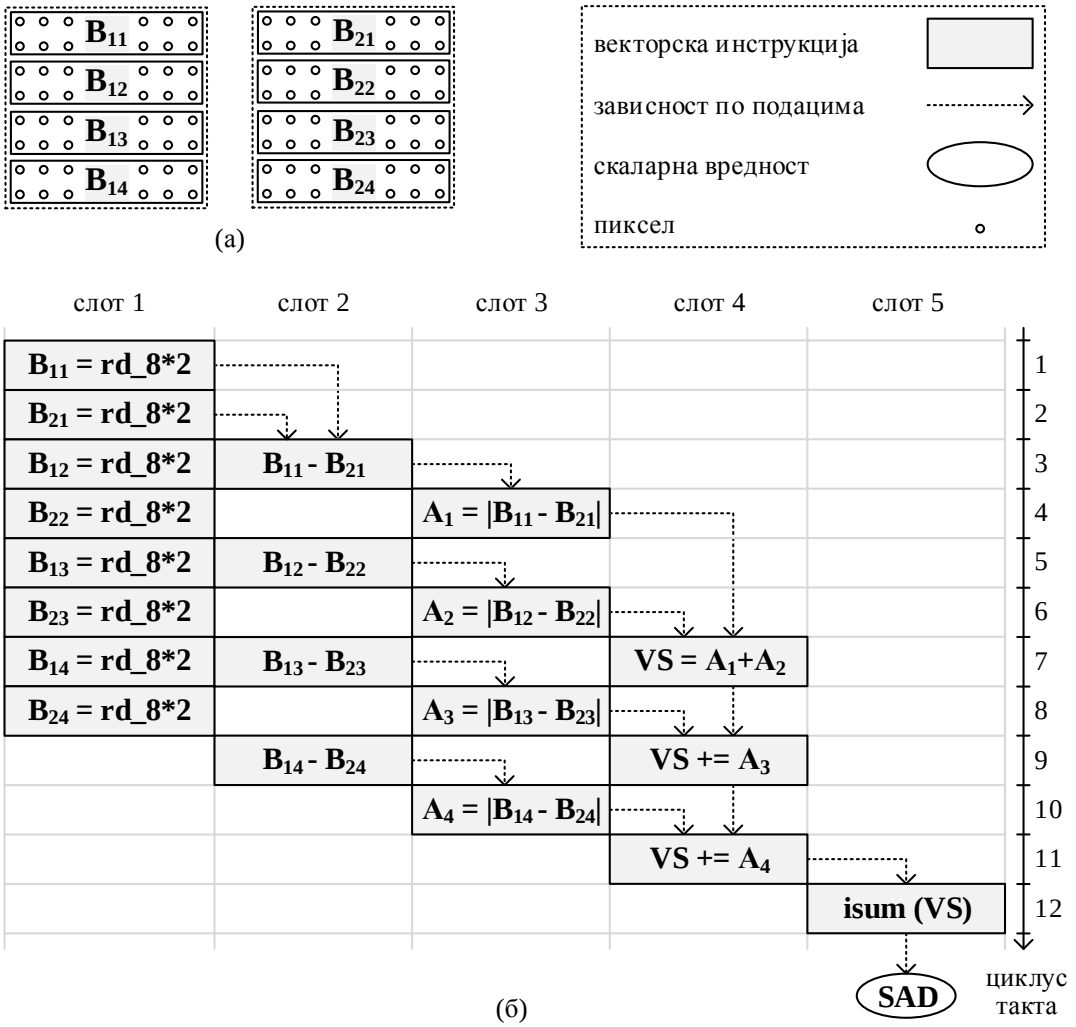


Слика 2.4 – Читање блока од 8∗8 пиксела из меморијског подсистема извршавањем четири просторно непреклопљена 4∗4 или 8∗2 приступа.

упаривање блокова са прецизношћу од једног пиксела[8]. Код оваквог упаривања блокова, за израчунавање једне 8∗8 SAD вредности читају се два блока од 8∗8 пиксела из меморијског подсистема. Притом се изврши четири 4∗4 или 8∗2 приступа подсистему по једном 8∗8 блоку, за $N = 16$, као што илуструје слика 2.4. Исто тако, за израчунавање једне 4∗4 или 16∗16 SAD вредности, потребно је извршити један или шеснаест 4∗4 приступа подсистему, респективно. Ови бројеви приступа су теоријски минимум за дату вредност N и величину SAD блока, јер се сваки пиксел једног SAD блока прочита само једном из подсистема. Стога се може рећи да је у овом случају ефикасност паралелног меморијског подсистема и постојећих начина приступа максимална.

Илустрације ради, слика 2.5 приказује израчунавање једне 8∗8 SAD вредности на векторској путањи података која паралелно, у пет функционалних јединица, односно слотова, може да изврши пет векторских операција. Притом, слот 1 реализује операције читања и уписа у паралелни меморијски подсистем, подржавајући начине приступа који су приказани на слици 2.3. Како се једном векторском операцијом обрађује $N = 16$ пиксела, сваки 8∗8 блок дели се на четири 8∗2 блока, као што је приказано на слици 2.5а, који се један по један читају и користе као операнди векторских операција. Распоред векторских операција, којима се реализује SAD израчунавање на векторској путањи података, приказан је на слици 2.5б. Само израчунавање SAD вредности реализовано је у слотовима 2 – 5, траје десет циклуса такта и завршава се операцијом „isum”

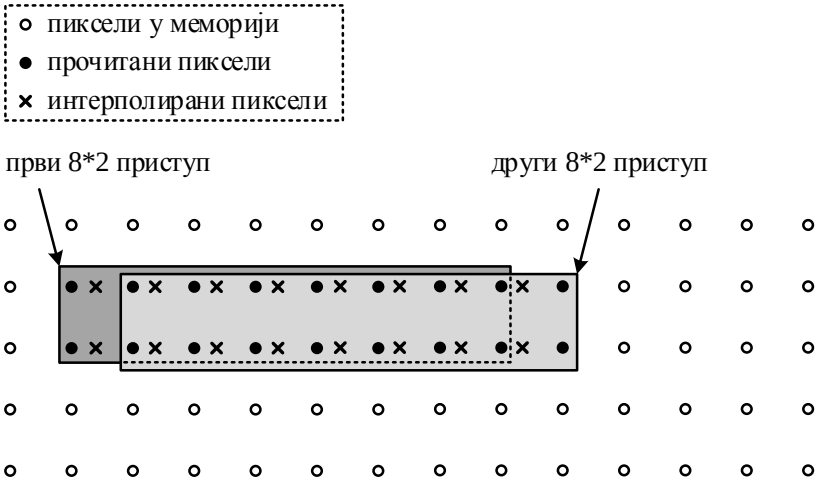
[8] енг. full-pixel block-matching



Слика 2.5 – Израчунавање једне 8∗8 SAD вредности са прецизношћу од једног пиксела на векторској путањи података и паралелном меморијском подсистему.

која сумира све пикселе улазног вектора и враћа скаларну SAD вредност као излаз. Читање пиксела из меморијског подсистема операцијом „rd_8∗2”, приказано у слоту 1, максимално је ефикасно јер се у једном такту прочита један 8∗2 блок, односно један операнд векторске операције. Укупно време потребно за израчунавање једне SAD вредности, које укључује и читање пиксела, износи дванаест циклуса такта. Ради јасноће, илустровано је извршавање у пет слотова иако се израчунавање може обавити у истом броју циклуса такта и са само три слота, јер се векторске операције у слотовима 2 и 3, као и оне у слотовима 4 и 5, извршавају наизменично.

Међутим, постојећа решења паралелног меморијског подсистема и постојећи начини приступа нису тако ефикасни за реализацију такозваних суб-пиксел ме-



Слика 2.6 – Читање блока од 9∗2 пиксела, потребних за линеарну интерполацију блока од 8∗2 пиксела, извршавањем два преклопљена 8∗2 приступа.

тода упаривања блокова[9] [1, 4], где је прецизност упаривања блокова финија од једног пиксела и најчешће износи четвртину или осмину пиксела. Наиме, код суб-пиксел метода, блокови који су улаз SAD функције не могу се директно прочитати из меморијског подсистема, већ се сваки блок интерполира на основу вредности пиксела у меморији. Притом се најчешће користе линеарна и билинеарна интерполација, јер их одликује релативно ниска рачунска комплексност, што је погодно за извршавање у реалном времену, уз довољно добар квалитет интерполације за потребе упаривања блокова [37–39]. Ово захтева да се прочита више пиксела по једном SAD израчунавању него код упаривања блокова са прецизношћу од једног пиксела. На пример, за линеарну интерполацију 8∗2 блока потребно је прочитати блок од 9∗2 пиксела извршавањем два приступа просторно преклопљеним 8∗2 блоковима у меморијском подсистему, као што приказује слика 2.6. Слично томе, за линеарну интерполацију једног 8∗8 или 16∗16 блока, подељену у четири или шеснаест интерполација блокова од 8∗2 пиксела, потребно је извршити осам или тридесет два 8∗2 приступа, респективно. Оваква реализација није ефикасна јер су 8∗2 приступи међусобно преклопљени и велики број пиксела се чита два пута. Као последица тога, број приступа меморијском подсистему извршених за израчунавање једне SAD вредности двоструко је већи у односу на упаривање блокова са прецизношћу од једног пиксела. То даље за последицу има два пута мању брзину обраде, одно-

[9] енг. sub-pixel block-matching

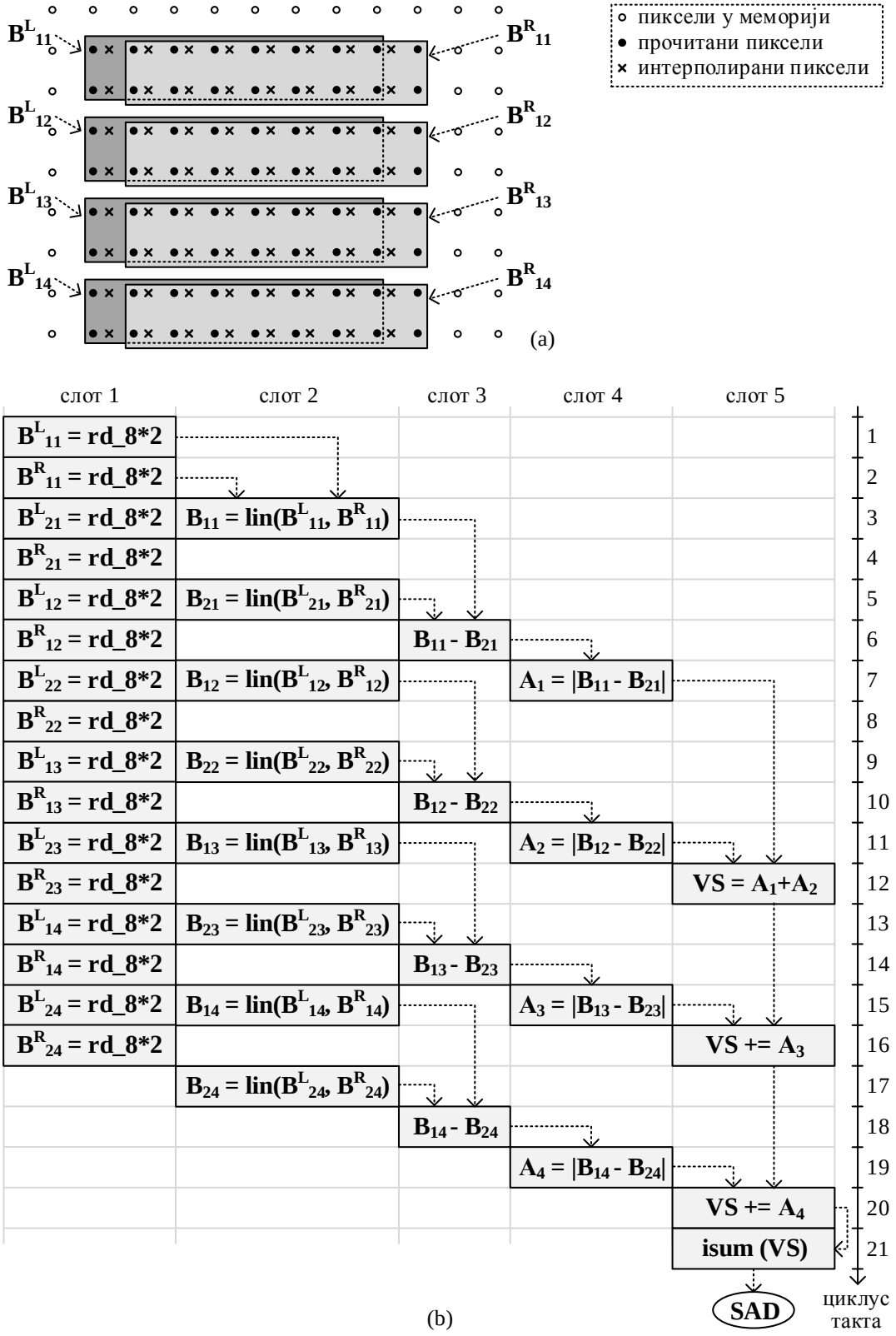
сно два пута мањи број израчунатих SAD вредности у јединици времена, као и два пута више утрошене енергије меморијског подсистема за израчунавање једне SAD вредности.

Слика 2.7 илуструје израчунавање једне 8∗8 SAD вредности уз линеарну интерполацију улазних 8∗8 блокова. Интерполација се реализује операцијом „lin" у слоту 2, чији су операнди два преклопљена 8∗2 блока прочитана извршавањем два приступа меморијском подсистему у слоту 1. Укупно време за израчунавање SAD вредности је двадесет један циклус такта, што је скоро два пута више него у примеру без линеарне интерполације. Од тога се шеснаест циклуса троши на читање из меморијског подсистема, тачно два пута више него на слици 2.5. Притом се педесет шест од седамдесет два пиксела потребних за интерполацију једног 8∗8 SAD блока, односно преко 75 процената, прочита два пута из меморијског подсистема.

Иста врста неефикасности постоји и код билинеарне интерполације, која се састоји од две одвојене линеарне интерполације, једне у хоризонталном и друге у вертикалном правцу, али и у случају других функција за интерполацију пиксела и метода из широке класе просторних филтара са прозорском функцијом, који такође захтевају више преклопљених приступа меморијском подсистему на N обрађених пиксела. Детаљна анализа неефикасности при реализацији ових метода дата је у поглављу 3.

Формално и уопштено дефинисано, основни недостатак постојећих решења паралелног меморијског подсистема јесте недовољна прилагођеност њихових функционалности, односно начина приступа, методама обраде које за реализацију на векторској путањи података захтевају да се за обраду N суседних пиксела прочитају групе са нешто више од N пиксела. Применом постојећих начина приступа за реализацију оваквих метода извршава се више просторно преклопљених приступа, што доводи до вишеструког читања истих пиксела. То за последицу има спорију обраду и већу потрошњу енергије за дату обраду, јер меморијски подсистем најчешће представља уско грло перформанси и једног од највећих потрошача енергије код савремених векторских процесора.

На основу наших сазнања, ова врста неефикасности није била предмет ни једног до сада предложеног решења паралелног меморијског подсистема. Из тог разлога, циљ овог истраживања јесте да се описане неефикасности превазиђу и на тај начин омогуће бржа обрада на векторском процесору и мања потрошња енергије паралелног меморијског подсистема.



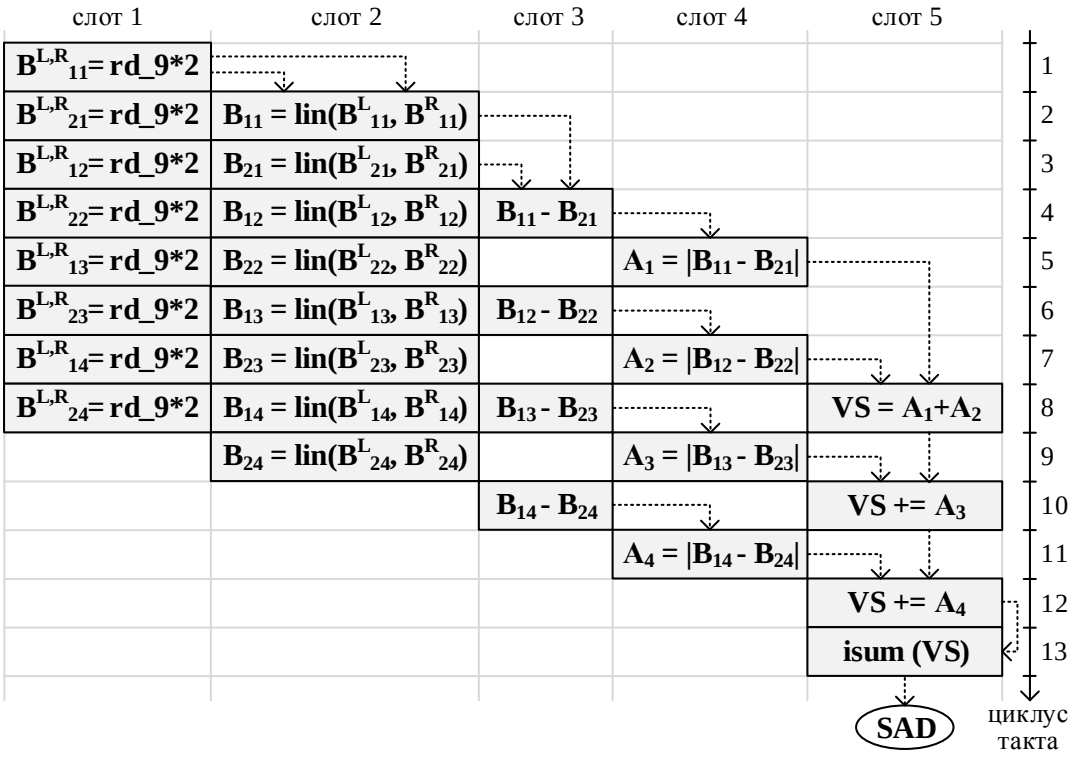
Слика 2.7 – Израчунавање једне 8∗8 SAD вредности уз линеарну интерполацију улазних блокова.

2.3 Детаљни опис доприноса истраживања

Како је у претходним секцијама овог поглавља дефинисан проблем истраживања, као потреба за ефикасним решењем паралелног меморијског подсистема, а потом су наведени и илустровани недостаци постојећих решења, сада је могуће дати више детаља о најважнијим доприносима овог рада.

У овом раду предложено је ново и ефикасније решење паралелног меморијског подсистема, које превазилази описане недостатке постојећих решења. У односу на постојеће, предложено решење омогућава бржу обраду и притом троши мање енергије у методама суб-пиксел упаривања блокова, интерполације пиксела у компензацији кретања и филтрирања у просторном домену. Истовремено, задржана је висока ефикасност у осталим методама обраде слике и видеа, једнака најефикаснијим постојећим подсистемима. Доприноси овог рада који такву ефикасност омогућавају и експериментално показују су:

1. Нове функционалности, односно нови начини приступа, дефинисани у поглављу 3 на основу анализе кључних метода обраде и њихове реализације на векторском процесору, која је изложена у истом поглављу. Нови начини приступа спајају више приступа просторно преклопљеним блоковима или редовима од N пиксела у један приступ блоку или реду са више од N пиксела. Ово је могуће када је образац преклапања одређен и познат унапред, у време развоја меморијског подсистема, као што је то случај са линеарном и билинеарном интерполацијом. На пример, два преклопљена $8*2$ приступа илустрована на слици 2.6 спајају се у један коришћењем новог начина приступа блоку од $9*2$ пиксела. Како би се омогућила даља обрада на истој векторског путањи података, која обрађује N пиксела у паралели, прочитани блок или ред са више од N пиксела дели се на више блокова или редова од N пиксела. То се врши одмах по пристизању прочитаног блока или реда на путању података, односно у оквиру операције читања из меморијског подсистема. Операције читања прилагођене су одговарајућој намени и у складу са тим врше поделу једног блока или реда на више њих. На примеру линеарне интерполације, прочитани $9*2$ блок дели се на два $8*2$ блока, хоризонтално померена за један пиксел, онако како је то приказано на слици 2.6. На тај начин, више преклопљених блокова од N пиксела, који су потребни за даљу обраду, прочитају се извршавањем једног уместо више приступа меморијском подсистему, омогућавајући бржу



Слика 2.8 – Израчунавање једне 8∗8 SAD вредности уз линеарну интерполацију улазних блокова, применом новог 9∗2 начина приступа меморијском подсистему.

обраду и елиминишући поновљена читања истих пиксела.

Слика 2.8 илуструје израчунавање једне 8∗8 SAD вредности уз линеарну интерполацију улазних блокова, коришћењем новог 9∗2 начина приступа меморијском подсистему. Укупан број меморијских приступа у овом случају је осам, односно два пута је мањи у односу на постојеће архитектуре и пример илустрован на слици 2.7. Услед тога укупно време за израчунавање SAD вредности смањено је са двадесет један на тринаест циклуса такта, а сваки од седамдесет два пиксела, потребних за интерполацију једног 8∗8 SAD блока, прочита се само једном из меморијског подсистема, што минимизује утрошену енергију подсистема по једној SAD вредности.

Предложени начини приступа су параметризовани, као што је описано у секцији 5.3, како би се могли прилагодити паралелизму векторске путање података N и потребама конкретних метода упаривања блокова, интерполације пиксела и филтрирања у просторном домену.

2. Параметризована, скалабилна и економична архитектура предложеног ре-

шења паралелног меморијског подсистема, детаљно описана у поглављу 5, која реализује како нове, тако и постојеће начине приступа. Архитектура укључује искошени образац распоређивања пиксела у меморијске банке, затим скуп са типично мање од N меморијских банака, где адресибилна реч једне банке складишти више пиксела, као и интерну управљачку логику која реализује искошени распоред у меморијским банкама и начине приступа, омогућавајући ефикасно и једноставно програмирање приступа меморијском подсистему. Притом су искошени образац распоређивања пиксела и коришћени скуп банака познати из литературе.

Цене описане архитектуре, односно заузеће површине на чипу и потрошња енергије по једном приступу мање су или једнаке ценама најефикаснијих постојећих архитектура, као што је анализирано и закључено у секцији 5.6. Стога, поред повећања брзине обраде, предложено решење паралелног меморијског подсистема троши и мање енергије од постојећих подсистема за обраду истог броја пиксела, захваљујући мањем укупном броју извршених приступа подсистему, што је експериментално показано у секцији 6.4.

3. Експериментална студија случаја, описана у поглављу 6, која јасно показује предности предложеног решења паралелног меморијског подсистема на примеру реализације добро познате 3DRS методе за естимацију кретања суб-пиксел упаривањем блокова [40]. За поређење са предложеним решењем изабрани су подсистеми који тренутно представљају највиши степен развоја у области [23, 26, 33–36].

   Применом предложеног решења паралелног меморијског подсистема постигнута је брзина естимације кретања од 60 фрејмова у секунди за видео резолуције 3840∗2160 пиксела, што је од 40 до 70 процената више од постојећих решења, а уз исту фреквенцију такта од 600 мегагерца и исту векторску путању података. Притом је број приступа меморијском подсистему смањен од 1,6 до 2,1 пута, смањујући укупну потрошњу енергије подсистема од 17 до 44 процента.

3

АНАЛИЗА МЕТОДА ОБРАДЕ И СИНТЕЗА НАЧИНА ПРИСТУПА

У овом поглављу прво се анализирају методе обраде слике и видеа које су од интереса за истраживање, са аспекта ефикасности њихове реализације на векторском процесору и паралелном меморијском подсистему. На основу анализе дефинишу се начини приступа паралелном меморијском подсистему, потребни да би подсистем омогућио брзу обраду на векторској путањи података и притом био економичан у потрошњи енергије.

Анализиране су следеће области обраде: естимација кретања упаривањем блокова, интерполација пиксела у компензацији кретања и филтрирање у просторном домену, у секцијама 3.1, 3.2 и 3.3, респективно. На почетку сваке од ових секција наведени су потребни начини приступа за посматрану област обраде, добијени као резултат потом изложене анализе. Након тога је за сваку од посматраних области дат кратак преглед, што обухвата циљ обраде, најважније примене и најпознатије методе. Затим су описане кључне функције које у највећој мери одређују рачунску сложеност тих метода. На основу тога су одређени потребни начини приступа, што укључује како оне који су претходно предложени и које подржавају постојеће архитектуре паралелног меморијског

подсистема, тако и нове начине приступа који представљају иновативни допринос овог рада. На крају сваке секције описане су реализације кључних функција на векторској путањи података, са и без употребе нових начина приступа, како би се илустровао њихов утицај на убрзање обраде и смањење потрошње енергије паралелног меморијског подсистема.

3.1 Упаривање блокова у естимацији кретања

Да би омогућио брзо и енергетски ефикасно упаривање блокова на векторској путањи података чији паралелизам износи N пиксела, паралелни меморијски подсистем треба да подржи следеће начине приступа:

1. Неповравнати приступ блоку од N пиксела у више различитих пропорција, односно више различитих односа његове ширине и висине.
2. **Непоравнати приступ блоку са више од N пиксела у више различитих пропорција.**
3. Поравнати приступ реду од N и више пиксела.

Друга врста начина приступа је иновативна идеја овог рада, док су прва и трећа већ познате и подржане од стране постојећих архитектура паралелног меморијског подсистема. Следи детаљнија анализа у наредним секцијама.

3.1.1 Преглед примена

Техника упаривања блокова највећу и најпознатију примену има у естимацији кретања. У овој области упаривање блокова се користи више од три деценије као основа великог броја метода [40–100].

Естимација кретања израчунава математички модел, који за сваки фрејм видео секвенце и за сваку групу пиксела у тренутно посматраном фрејму дефинише позицију најсличније групе у неком другом, односно референтном фрејму. Тренутни фрејм се најпре подели на групе пиксела, а потом се за сваку групу пронађе најсличнија у референтном фрејму и додели јој се одговарајући вектор помераја. На исти начин се врши естимација кретања и за секвенцу слика, које су снимљене непосредно једна за другом. Овако добијени математички модел кретања неопходан је у многим савременим применама обраде слике и видеа.

Најпознатија примена јесте у кодовању и компресији видеа, где служи за уклањање временске редундансе предиктивним кодовањем фрејмова са компен-

зацијом кретања [4]. Основна идеја јесте да се искористи сличност између суседних фрејмова у секвенци и да се уместо самих пиксела кодују разлике између њих. На тај начин се постиже компресија видеа, односно заузима се мање простора на медијуму за смештање и захтева се мањи капацитет комуникационог канала за пренос у реалном времену. Притом, како би разлике између пиксела биле мање, а степен компресије већи, естимацијом кретања се проналазе најсличнији пиксели у референтном фрејму и у односу на њих се врши кодовање. На овом принципу се заснивају савремени стандарди за кодовање и компресију видеа, као што су MPEG-4 и H.264 [4], а естимација кретања представља један од рачунски најсложенијих делова [101]. Приметићемо да у овој примени резултат естимације кретања не мора да одговара стварном кретању објеката у видео сцени, већ је довољно да се минимизују разлике између пиксела, односно минимизује грешка предиктивног кодовања [1].

Са друге стране, област примене у којој се захтева да резултат естимације представља стварно кретање у сцени јесте конверзија видео формата [1]. Ова област обухвата повећање броја фрејмова у секунди, скалирање резолуције и уклањање прореда[1] у реалном времену при репродукцији видеа. Циљ ових обрада јесте да се формат снимљеног видеа при приказивању прилагоди карактеристикама екрана како би се гледаоцу пружио што бољи визуелни утисак. Услед брзог пораста броја и врста уређаја који снимају видео у разноврсним форматима и различитог квалитета, а уз истовремену експанзију екрана разних величина, резолуција и брзина освежавања, значај конверзије формата је све већи. Повећањем броја фрејмова у секунди при репродукцији прилагођава се брзина снимљеног видеа, од најчешће 24, 25 или 30 фрејмова у секунди, брзини освежавања савремених екрана, која износи 60, 120 или више фрејмова у секунди. Повећање се врши креирањем нових фрејмова и њиховим уметањем на временске позиције између постојећих фрејмова. Притом, најбољи визуелни квалитет се добија применом естимације, а потом и компензације кретања при интерполацији нових фрејмова, јер се тако избегава визуелно непријатан ефекат „подрхтавања" слике[2] услед брзог померања камере и објеката у видео сцени [1]. Скалирањем се прилагођава резолуција снимљене слике или видеа резолуцији екрана на коме се врши репродукција, а савремене методе, попут оних заснованих на супер-резолуцији, такође користе естимацију кретања за регистрацију

[1] енг. frame-rate up-conversion, resolution scaling, and deinterlacing

[2] енг. motion judder

фрејмова, односно њихово међусобно поравнавање[3] пре даље обраде [6, 9]. У телевизијском емитовању се користи формат видеа са проредом[4], где суседни фрејмови у секвенци наизменично садрже само парне или само непарне линије пиксела и називају се полу-фрејмови, односно полу-слике. Данашњи екрани приказују потпуне фрејмове и стога је потребно да се изврши реконструкција недостајућих линија пиксела, а најбољи резултати се постижу такође применом естимације и компензације кретања са суб-пиксел прецизношћу [1, 102]. Естимација стварног кретања неопходна је и за дигиталну стабилизацију приликом снимања видеа, односно за компензовање нежељеног подрхтавања уређаја којим се снимање врши [5].

Естимација кретања упаривањем блокова користи се и за потискивање шума[5], која је једна од основних и широко распрострањених обрада слике и видеа. Услед велике резолуције и мале површине сензора слике у мобилним уређајима, површина сензора која прикупља светлост за један пиксел је веома мала, што резултује великом количином шума [103–105]. То је посебно изражено када се слика или видео снимају у условима слабе осветљености сцене[6]. Из тог разлога потискивање шума има посебно велики значај у домену мобилних уређаја.

Неке од најбољих метода за потискивање шума, које су публиковане у последњих десетак година [7, 106–112], користе упаривање блокова при проналажењу сличних блокова пиксела у оквиру једне или више слика, односно фрејмова. Потом се слични блокови групишу и заједно филтрирају да би се шум потиснуо у излазном блоку пиксела. Ове методе се заснивају на претпоставци да је шум случајна величина и да стога није корелисан за различите блокове у слици. На основу тога се очекује да се заједничким филтрирањем сличних блокова шум може потиснути уз очување детаља и ивица на слици [7].

3.1.2 Преглед метода

Према подели датој у књизи [1], естимација кретања може бити пиксел-рекурзивна [113], блоковска [41] и објектна[7] [114], а на основу величине групе пиксела којој се додељује један вектор помераја. Код пиксел-рекурзивне ести-

[3] енг. image registration and alignment

[4] енг. interlaced video format

[5] енг. denoising

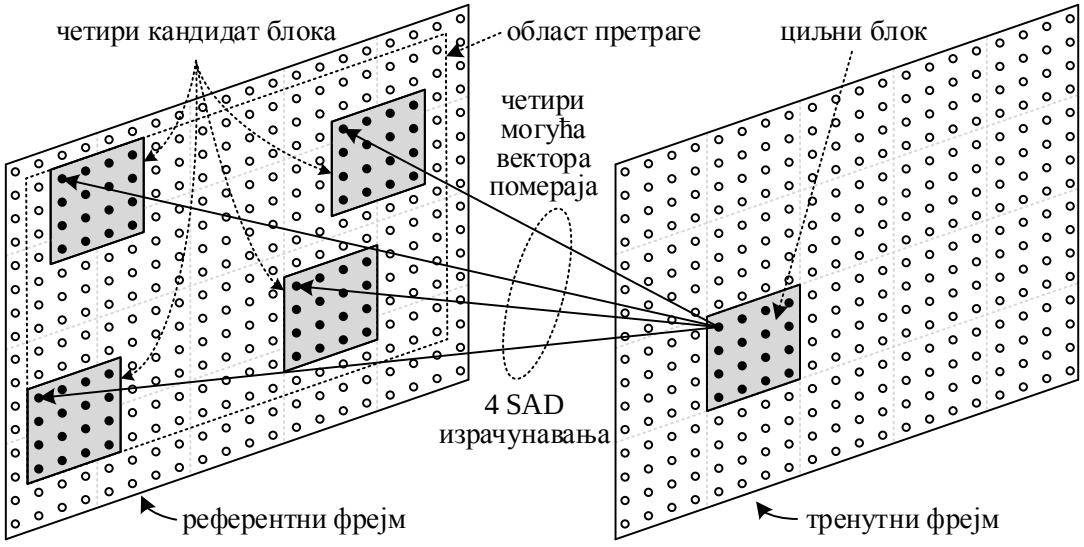
[6] енг. low-light imaging

[7] енг. pel-recursive, block-based, and object-based

мације, вектор помераја се одређује за сваки пиксел у фрејму. Овакав приступ се, осим великом рачунском сложеношћу, одликује и малом отпорношћу на шум и због тога има ограничену практичну примену [1]. Блоковске методе естимације кретања, код којих се вектор помераја одређује за сваки блок пиксела у тренутном фрејму, отпорније су на шум јер су објекти који се крећу у реалној видео сцени већи од једног пиксела, а најчешће су већи и од једног блока. Поред тога, блоковска природа ових метода је погодна за хардверско-софтверску реализацију, што је пресудно утицало на њихову велику популарност и широку практичну примену [1]. Објектни приступ естимацији кретања иде корак даље покушавајући да пронађе објекте који се крећу у видео сцени и за њих одреди помераје. Иако овакав приступ има перспективу, постојеће објектне методе су превише рачунски сложене за практичну примену [1]. Због свега наведеног се у пракси углавном користе блоковске методе естимације кретања, којима се бавимо у наставку.

Блоковске методе естимације кретања заснивају се на упаривању блокова. Њихов улаз је секвенца видео фрејмова, а излаз је један скуп вектора помераја за сваки фрејм, односно једно поље вектора помераја[8] по фрејму. Фрејмови се обрађују један по један и притом се за сваки тренутно обрађивани фрејм користи један или више референтних фрејмова. Тренутни фрејм се подели на непреклопљене блокове од најчешће 4∗4, 8∗8 или 16∗16 пискела и за сваки се проналази најсличнији блок у референтном фрејму и додељује му се одговарајући вектор помераја. Како би се смањила рачунска сложеност естимације кретања и омогућила практична примена, за један циљни блок из тренутног фрејма претражује се само ограничени скуп кандидат блокова, који су одређени скупом могућих вектора помераја. Да би се одабрао најбољи вектор помераја, за сваки кандидат блок се израчунава мера његове сличности са циљним блоком. Функција која представља меру сличности два блока се притом најчешће извршава и одређује рачунску сложеност естимације кретања. Из тог разлога је уложено доста истраживачког труда да се пронађе једноставна, а истовремено поуздана функција [1]. Најпознатија и готово увек примењивана функција за меру сличности два блока је сума апсолутних разлика, означена са SAD и дефинисана једначином 2.1 у поглављу 2. Вектор помераја који резултује најмањом SAD вредношћу означава се као најбољи и додељује се циљном блоку. Слика 3.1 илуструје естимацију кретања упаривањем блокова, где се за један циљни

[8] енг. motion vector field



Слика 3.1 – Естимација кретања уаривањем блокова.

блок претражује скуп од четири могућа вектора помераја и израчунавају се четири SAD вредности.

Директан метод блоковске естимације кретања назива се упаривање блокова са потпуном претрагом[9], где се за један циљни блок као кандидати узимају сви блокови из области претраге око њега. Како би се ограничила меморијска захтевност естимације кретања, уводи се област претраге мања од читавог фрејма. Потпуна претрага се одликује веома великом рачунском сложеношћу, јер је скуп могућих вектора помераја пропорционалан величини области претраге и садржи више од хиљаду кандидата већ за област претраге од скромних 40∗40 пиксела. Да би се омогућила практична примена блоковске естимације кретања, истраживања у овој области углавном иду у правцу изналажења брзих метода, са што мањим скупом могућих вектора помераја и истовременим задржавањем високе поузданости естимације [1]. Неке од најпознатијих брзих метода за примену у кодовању и компресији видеа су дводимензионална логаритамска претрага [41], претрага у три корака[10] [42], претрага заснована на коњугованим правцима[11] [43], нова претрага у три корака[12] [55] и претрага у четири корака[13] [56], чији скупови претраге садрже неколико десетина могућих вектора

[9] енг. full-search block-matching (FSBM)

[10] енг. three-step search (3SS)

[11] енг. one-at-a-time search (OTS)

[12] енг. new three-step search (N3SS)

[13] енг. four-step search (4SS)

помераја [115]. За примену у конверзији видео формата, где је потребно одредити стварно кретање у видео сцени, најбоље резултате дају методе засноване на рекурзивној претрази [1]. Код оваквих метода, изабрани вектор помераја једног блока се додаје у скуп могућих вектора помераја њему суседних блокова, јер је претпоставка да суседни блокови, као део једног објекта, имају исто кретање. Друга претпоставка је да објекти у видео сцени не мењају нагло своје кретање и стога се у скуп могућих вектора помераја додају вектори помераја суседних блокова из претходног фрејма. Ове претпоставке чине основу најпознатијих метода за естимацију стварног кретања: тродимензионалне рекурзивне претраге[14] (3DRS) са прецизношћу од једног пиксела [51] и са суб-пиксел прецизношћу [40]. 3DRS је међу најбржим методама јер њен скуп претраге типично садржи мање од десет могућих вектора помераја. Унапређењем првобитних верзија ове методе, којима су постављени темељи естимације стварног кретања, настало је више деривата попут оних описаних у радовима [80, 93, 94]. Неке од новијих метода естимације стварног кретања су: временски компензована естимација са једноставном блоковском предикцијом[15] [72], естимација са више пролаза и пропагацијом вектора помераја[16] [88], брза естимација стварног кретања са промељивом величином блока за примену у конверзији видео формата са проредом[17] [92], естимација стварног кретања мале рачунске сложености за примену у повећању броја фрејмова у секунди[18] [99].

3.1.3 Потребни начини приступа

Од интереса за даљу анализу јесте SAD функција, која одређује рачунску сложеност у наведеним применама упаривања блокова. Стога, ова секција анализира обрасце приступа пикселима при реализацији SAD функције и одређује потребне начине приступа паралелном меморијском подсистему.

При израчунавању SAD функције, као мере сличности два блока, пиксели циљног блока се читају директно из тренутног фрејма. Такође, у упаривању блокова са прецизношћу од једног пиксела блокови кандидати се читају директно из референтног фрејма, јер могући вектори помераја имају такве вред-

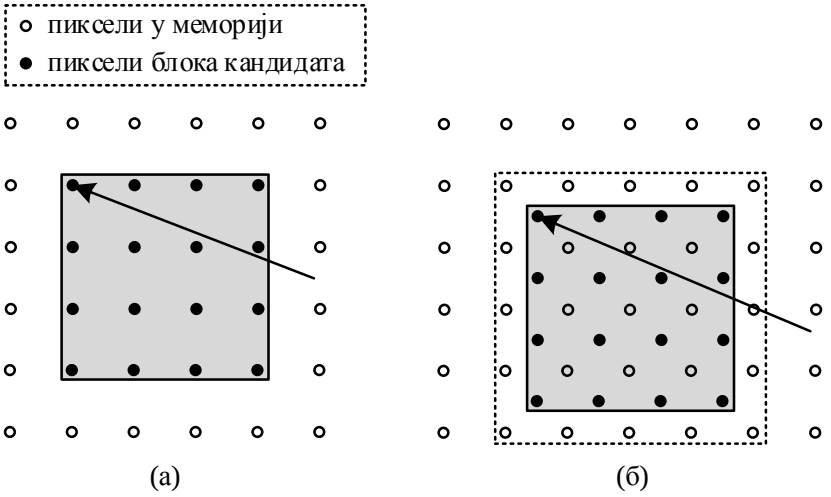
[14] енг. 3D recursive search block-matching (3DRS)

[15] енг. temporally compensated motion estimation with simple block-based prediction (TC-SBP)

[16] енг. multi-pass and motion vector propagation (MPMVP)

[17] енг. variable block size true motion estimation for a translated motion model (VBTME)

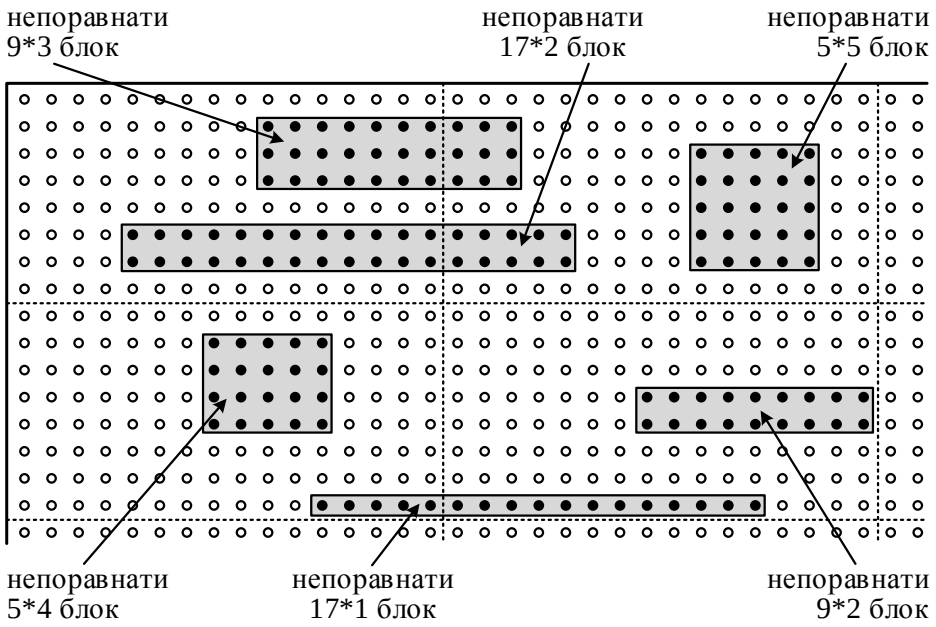
[18] енг. low-complexity true-motion estimation (TME)



Слика 3.2 – Могући вектори помераја (а) са прецизношћу од једног пиксела и (б) са суб-пиксел прецизношћу.

ности да показују тачно на неки од пиксела, као што је илустровано на слици 3.2а. Са друге стране, суб-пиксел упаривање блокова користи прецизније, односно суб-пиксел векторе помераја, који показују између пиксела у фрејму, као што приказује слика 3.2б. Због тога се пиксели сваког блока кандидата интерполирају на основу вредности њима суседних пиксела у фрејму. Узимајући све ово у обзир, можемо уочити три обрасца приступа пикселима и три потребна начина приступа паралелном меморијском подсистему:

1. **Образац приступа**: Читање блокова пиксела који се директно користе као операнди за израчунавање SAD функције, односно читање циљних и блокова кандидата при упаривању са прецизношћу од једног пиксела, као и читање циљних блокова у суб-пиксел упаривању. Број пиксела ових блокова је умножак од N у типичним случајевима када $N \in \{16, 32, 64\}$.
   Начини приступа: Неповезани приступ реду и блоку од N пиксела у више пропорција, као што су 4∗4, 8∗2 и 16∗1 за $N = 16$, затим 8∗4, 16∗2 и 32∗1 за $N = 32$ и 8∗8, 16∗4, 32∗2 и 64∗1 за $N = 64$. Коришћењем оваквих начина приступа, циљни или блок кандидат се подели на непреклопљене блокове или редове од N пиксела, који се прочитају извршавањем одговарајућег броја приступа меморијском подсистему, као што је раније приказано на слици 2.4 у секцији 2.2. Тај број извршених меморијских приступа је теоријски минимум за дату вредност N и величину циљног и блока кандидата, јер се сваки пиксел прочита тачно једном и сваки прочи-

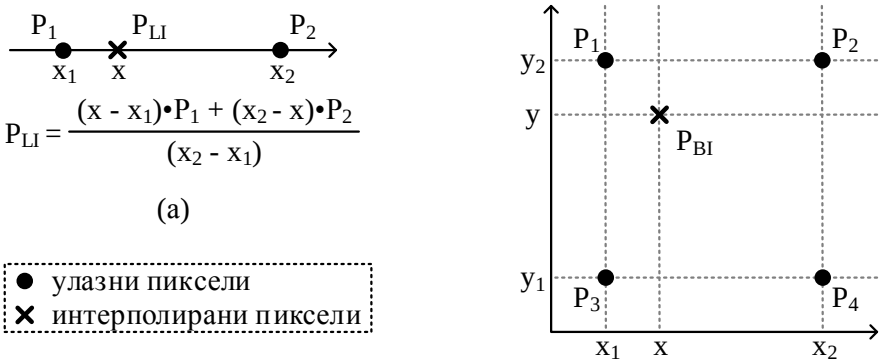


Слика 3.3 – Примери нових начина приступа, прилагођених суб-пиксел упаривању блокова са линеарном и билинеарном интерполацијом.

тани пиксел је потребан за даљу обраду. Ови начини приступа су познати и омогућени од стране постојећих архитектура паралелног меморијског подсистема.

2. **Образац приступа**: Читање блокова пиксела који се користе као операнди за интерполацију блокова кандидата у суб-пиксел упаривању блокова. Број пиксела једног блока кандидата је такође умножак од N као и циљни блок, а његова интерполација је подељена на интерполације блокова или редова од N пиксела, као што је приказано раније на слици 2.7 у секцији 2.2. Како интерполација једног пиксела захтева више суседних пиксела, тако интерполација блока од N пиксела захтева блок са више од N суседних пиксела.

   Нови начини приступа: Непоравнати приступ блоку или реду са више од N пиксела, односно блоку или реду са свим пикселима који су потребни за интерполацију блока или реда од N пиксела. Број пиксела блока или реда коме се приступа зависи од функције интерполације. На пример, у случају линеарне интерполације, блок или ред коме се приступа је шири за један пиксел од интерполираног блока или реда од N пиксела: $5*4$, $9*2$ и $17*1$ за $N = 16$, затим $9*4$, $17*2$ и $33*1$ за $N = 32$ и $9*8$, $17*4$, $33*2$ и $65*1$

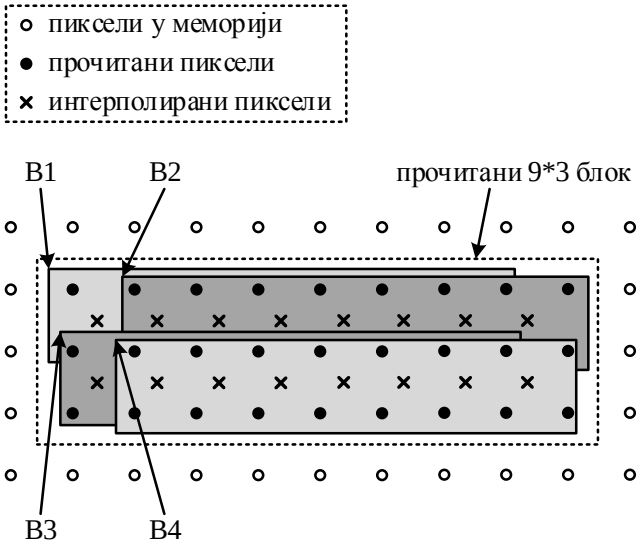


Слика 3.4 – Линеарна (а) и билинеарна (б) интеполација.

за $N = 64$. У случају билинеарне интерполације, која се састоји од хоризонталне и вертикалне линеарне интерполације, блок коме се приступа је шири и виши за један пиксел од интерполираног блока или реда: 5∗5, 9∗3 и 17∗2 за $N = 16$, затим 9∗5, 17∗3 и 33∗2 за $N = 32$ и 9∗9, 17∗5, 33∗3 и 65∗2 за $N = 64$. Овакви начини приступа за $N = 16$ приказани су на слици 3.3. У општем случају, величина и пропорције блока или реда коме се приступа зависи од функције интерполације. **Ови начини приступа представљају један од иновативних доприноса овог рада**.

Нове операције читања и уписа: Када прочитани блок или ред са више од N пиксела стигне на векторску путању података, односно у јединицу за читање и упис у паралелни меморијски подсистем, он се дели на више блокова или редова од N пиксела, који се прослеђују на даљу обраду другим јединицама чији је паралелизам N пиксела. На тај начин, довољан је један приступ меморијском подсистему за интерполацију блока или реда од N пиксела, док векторска путања података не треба да буде шира од N пиксела, а самим тим ни скупља у смислу заузећа површине на чипу. Модификација путање података може бити потребна само у смислу додавања нових функционалних јединица, како би се искористио већи инструкцијски паралелизам добијен новим операцијама читања из меморијског подсистема.

Начин на који се врши подела блока или реда зависи од функције интерполације, тако да добијени блокови или редови од N пиксела буду погодни за даљу обраду. У примеру линеарне интерполације, приказане за један



Слика 3.5 – Читање блока од 9∗3 пиксела и подела на четири преклопљена 8∗2 блока, означена са B1, B2, B3 и B4, који су потребни за билинеарну интерполацију 8∗2 блока.

пиксел на слици 3.4а, један 9∗2 блок се дели на два 8∗2 блока померена за једну пиксел колону, онако како је приказано на слици 2.6 у секцији 2.2. Ова два блока се потом директно користе као операнди за израчунавање линеарне интерполације, односно нису потребне додатне операције за припрему операнада. Слично томе, у случају билинеарне интерполације, илустроване на слици 3.4б, један 9∗3 блок се дели на четири 8∗2 блока, онако како је приказано на слици 3.5.

Овакве операције читања и уписа такође представљају иновативни допринос овог рада.

Слично решење се може применити и на друге функције интерполације, просторне филтре и остале методе обраде које читају групе са нешто више од N суседних пиксела на N обрађених пиксела, као што ће бити објашњено у наредним секцијама овог поглавља.

3. **Образац приступа**: Преноси дводимензионалних низова пиксела, односно делова фрејмова и слика из меморије ван чипа у паралелни меморијски подсистем на чипу, где су фрејмови и слике у меморији ван чипа смештени у поретку по линијама[19] и где меморија ван чипа омогућава само поравнати приступ реду пиксела. Ови преноси се обављају паралелно са обрадом на векторској путањи података и надмећу се са преносима

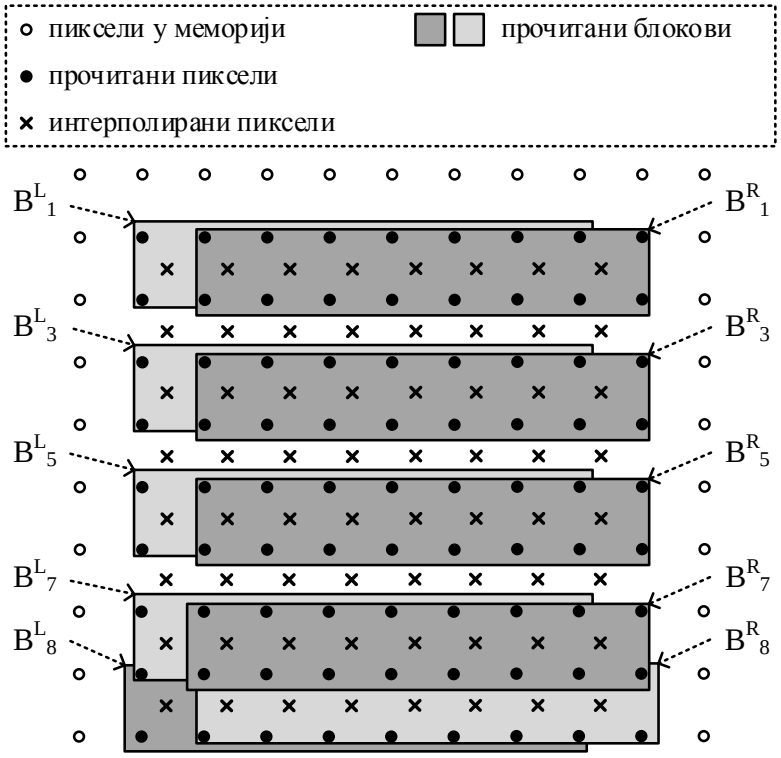
[19]енг. row-major

других уређаја у систему који такође приступају дељеној меморији ван чипа. Из тог разлога су очекивани конфликти између читања и уписа у паралелни меморијски подсистем, као и између такмаца за приступ меморији ван чипа. Како би се ограничило успорење обраде због ових конфликата, преноси треба да се обаве уз минималан број приступа, како паралелном меморијском подсистему на чипу, тако и меморији ван чипа. Минимизовање броја ових приступа такође минимизује потрошњу енергије меморија. То је посебно важно у случају меморије ван чипа, која значајно доприноси укупној потрошњи енергије читавог система. За минимизовање броја приступа меморијама потребно је да број пренесених пиксела по једном приступу буде максималан и да се минимизује број поновљених преноса истих пиксела.

Начини приступа: Поравнати приступ реду пиксела, што је подударно са начином приступа меморији ван чипа. Ови начини приступа су познати и омогућени од стране постојећих архитектура паралелног меморијског подсистема. **Иновативни допринос овог рада у овом погледу јесте могућност приступа поравнатом реду од** $2 * N$ **пиксела**. То омогућава два пута брже преносе од постојећих архитектура паралелног меморијског подсистема, а уз исту потрошњу енергије и исто заузеће површине на чипу, као што ће бити показано у наредним поглављима рада. Поновљени преноси истих пиксела последица су ограниченог капацитета паралелног меморијског подсистема на чипу. Овај проблем се најчешће решава софтверским путем, применом метода као што је покретни-L1 [36], чији је циљ да више пута искористи пикселе једном уписане у паралелни меморијски подсистем и тако избегне њихове поновљене преносе. Архитектура предложена у овом раду подржава покретни-L1 и сличне софтверске методе тако што омогућава да се у време извршавања обраде користе два начина приступа истом дводимензоналном низу: приступ блоку пиксела за читање од стране векторске путање података и приступ реду пиксела за пренос од или ка меморији ван чипа.

3.1.4 Реализације критичне функције

Ова секција илуструје и упоређује три варијанте реализације 8∗8 SAD функције на векторској путањи података, коришћењем три начина приступа паралелном меморијском подсистему за билинеарну интерполацију блока кандидата:



Слика 3.6 – Подела блока од 9∗9 пиксела на 8∗2 блокове, који се читају за билинеарну интерполацију 8∗8 блока.

1. постојећег 8∗2,

2. новог 9∗2 и

3. новог 9∗3 начина приступа.

Циљ је да се прикажу предности нових у односу на постојеће начине приступа.

У све три варијанте се читање циљног 8∗8 блока врши помоћу четири 8∗2 приступа, као што илуструје слика 2.4 у секцији 2.2. За билинеарну интерполацију 8∗8 блока, као што је већ речено, потребан је блок од 9∗9 пиксела. У првој варијанти, која користи познати 8∗2 начин приступа, потребно је извршити десет операција читања, као што приказује слика 3.6. Псеудокод програма који на овај начин реализује билинеарну интерполацију блока кандидата, а потом израчунава вредност SAD функције, приказан је на слици 3.7а. Након што се прочитају потребни 8∗2 блокови врши се хоризонтална линеарна интерполација операцијом „lin”, као први корак билинеарне интерполације. На тај начин се добијају међурезултати B_1, B_3, B_5, B_7 и B_8, који се користе као операнди за вертикалну линеарну интерполацију. На основу ових међурезултата креирају

```
rd_8*2 (B^L_1);
rd_8*2 (B^R_1);
rd_8*2 (B^L_3);
rd_8*2 (B^R_3);
rd_8*2 (B^L_5);
rd_8*2 (B^R_5);
rd_8*2 (B^L_7);
rd_8*2 (B^R_7);
rd_8*2 (B^L_8);
rd_8*2 (B^R_8);

B_1 := lin (B^L_1, B^R_1);
B_3 := lin (B^L_3, B^R_3);
B_5 := lin (B^L_5, B^R_5);
B_7 := lin (B^L_7, B^R_7);
B_8 := lin (B^L_8, B^R_8);

B_2 := shuffle (B_1, B_3);
B_4 := shuffle (B_3, B_5);
B_6 := shuffle (B_5, B_7);

C_1 := lin (B_1, B_2);
C_3 := lin (B_3, B_4);
C_5 := lin (B_5, B_6);
C_7 := lin (B_7, B_8);

rd_8*2 (D_1);
rd_8*2 (D_3);
rd_8*2 (D_5);
rd_8*2 (D_7);

S_1 := sub (C_1, D_1);
S_3 := sub (C_3, D_3);
S_5 := sub (C_5, D_5);
S_7 := sub (C_7, D_7);

A_1 := abs (S_1);
A_3 := abs (S_3);
A_5 := abs (S_5);
A_7 := abs (S_7);

SAD := isum (A_1);
SAD += isum (A_3);
SAD += isum (A_5);
SAD += isum (A_7);
```

(а)

```
rd_9*2 (B^L_1, B^R_1);
rd_9*2 (B^L_3, B^R_3);
rd_9*2 (B^L_5, B^R_5);
rd_9*2 (B^L_7, B^R_7);
rd_9*2 (B^L_8, B^R_8);

B_1 := lin (B^L_1, B^R_1);
B_3 := lin (B^L_3, B^R_3);
B_5 := lin (B^L_5, B^R_5);
B_7 := lin (B^L_7, B^R_7);
B_8 := lin (B^L_8, B^R_8);

B_2 := shuffle (B_1, B_3);
B_4 := shuffle (B_3, B_5);
B_6 := shuffle (B_5, B_7);

C_1 := lin (B_1, B_2);
C_3 := lin (B_3, B_4);
C_5 := lin (B_5, B_6);
C_7 := lin (B_7, B_8);

rd_8*2 (D_1);
rd_8*2 (D_3);
rd_8*2 (D_5);
rd_8*2 (D_7);

S_1 := sub (C_1, D_1);
S_3 := sub (C_3, D_3);
S_5 := sub (C_5, D_5);
S_7 := sub (C_7, D_7);

A_1 := abs (S_1);
A_3 := abs (S_3);
A_5 := abs (S_5);
A_7 := abs (S_7);

SAD := isum (A_1);
SAD += isum (A_3);
SAD += isum (A_5);
SAD += isum (A_7);
```

(б)

Слика 3.7 – Реализација 8∗8 SAD функције, чији су операнди: директно прочитани блок D_{1-7} и билинеарно-интерполирани блок C_{1-7}, уз читање (а) 8∗2 и (б) 9∗2 блокова.

се још три 8∗2 блока, означена са B_2, B_4 и B_6, који су такође потребни за вертикалну линеарну интерполацију. Њихово креирање се врши операцијом „shuffle", чији се излазни 8∗2 блок добија мешањем два улазна 8∗2 блока, тако да је прва

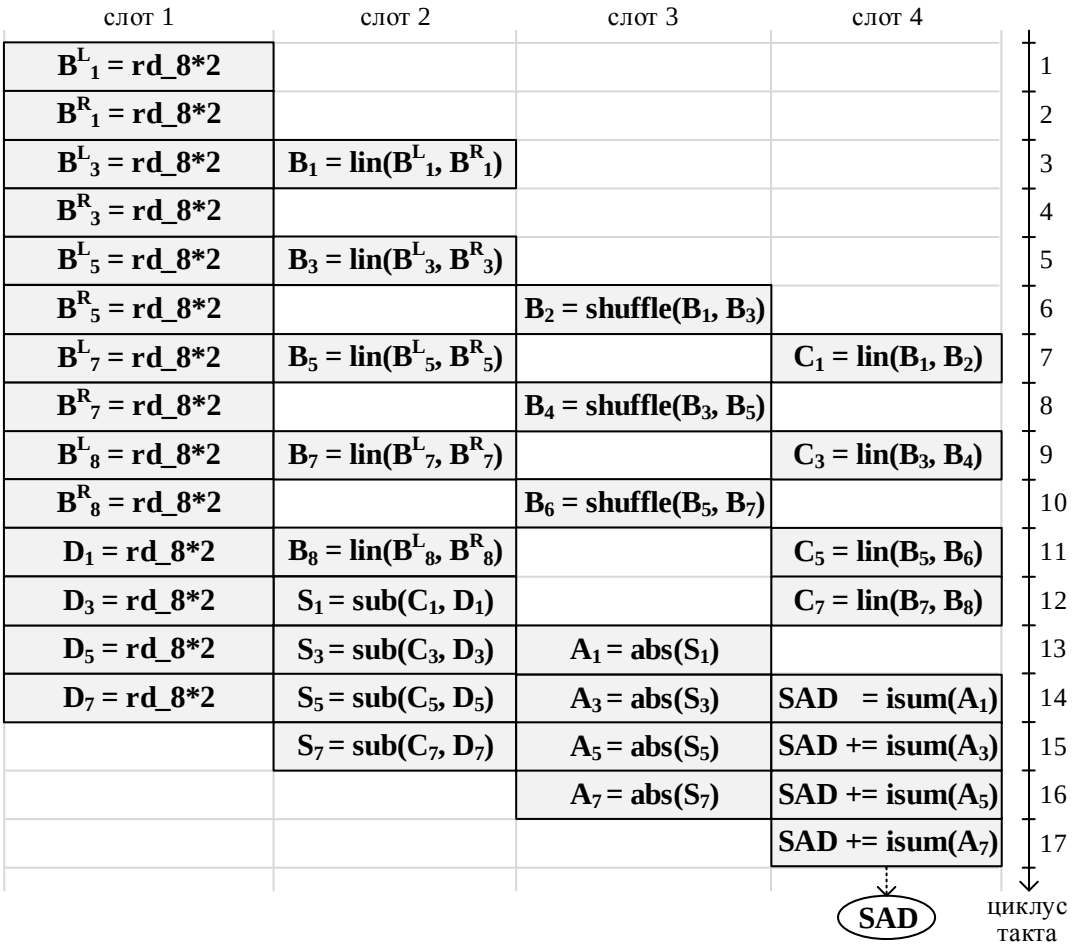
линија излазног блока једнака другој линији првог улазног блока, а друга линија излазног блока једнака је првој линији другог улазног блока. Вертикалном линеарном интерполацијом добија се 8∗8 блок кандидат кога чине C_1, C_3, C_5 и C_7 8∗2 блокови. Након билинеарне интерполације блока кандидата прочитају се 8∗2 делови циљног 8∗8 блока, означени са D_1, D_3, D_5 и D_7. На крају се израчуна скаларна вредност SAD функције, тако што се прво изврше векторске операције одузимања и апсолутне вредности, означене са „sub" и „abs", респективно, а потом се сумирају сви елементи вектора операцијом „isum".

Распоред наведених операција при извршавању на векторској путањи података, која има четири слота са функционалним јединицама[20], приказан је на слици 3.8. Са ове слике се може закључити да извршавање билинеарне интерполације блока кандидата траје дванаест, а израчунавање SAD функције укупно седамнаест циклуса такта. Притом се, за билинеарну интерполацију блока кандидата, изврши десет 8∗2 операција читања из паралелног меморијског подсистема и тиме прочита сто шездесет пиксела, од чега осамдесет један различитих. Када се у обзир узме и циљни блок, за израчунавање једне SAD вредности прочита се укупно двеста двадесет четири пиксела.

Друга варијанта реализације, коришћењем новог 9∗2 начина приступа, елиминише пет операција читања из паралелног меморијског подсистема, као што приказује псеудокод на слици 3.7б. Захваљујући томе што нова операција читања дели 9∗2 блок на погодан начин на два 8∗2 блока, остатак псеудокода је непромењен у односу на слику 3.7а.

Распоред операција којима се овај псеудокод реализује на истој векторској путањи са четири слота илустрован је на слици 3.9. Извршавање билинеарне интерполације блока кандидата траје осам циклуса такта, што је за четири циклуса мање, односно педесет процената брже у односу на реализацију коришћењем постојећег 8∗2 начина приступа. Убрзање обраде искључиво је резултат новог 9∗2 начина приступа и нове операције читања из паралелног меморијског подсистема јер је остатак псеудокода и векторске путање података непромењен. Укупно време за израчунавање једне SAD вредности је дванаест циклуса такта, што представља убрзање од око четрдесет два процента. Извршавањем пет 9∗2 операција читања за билинеарну интерполацију блока кандидата, прочита се

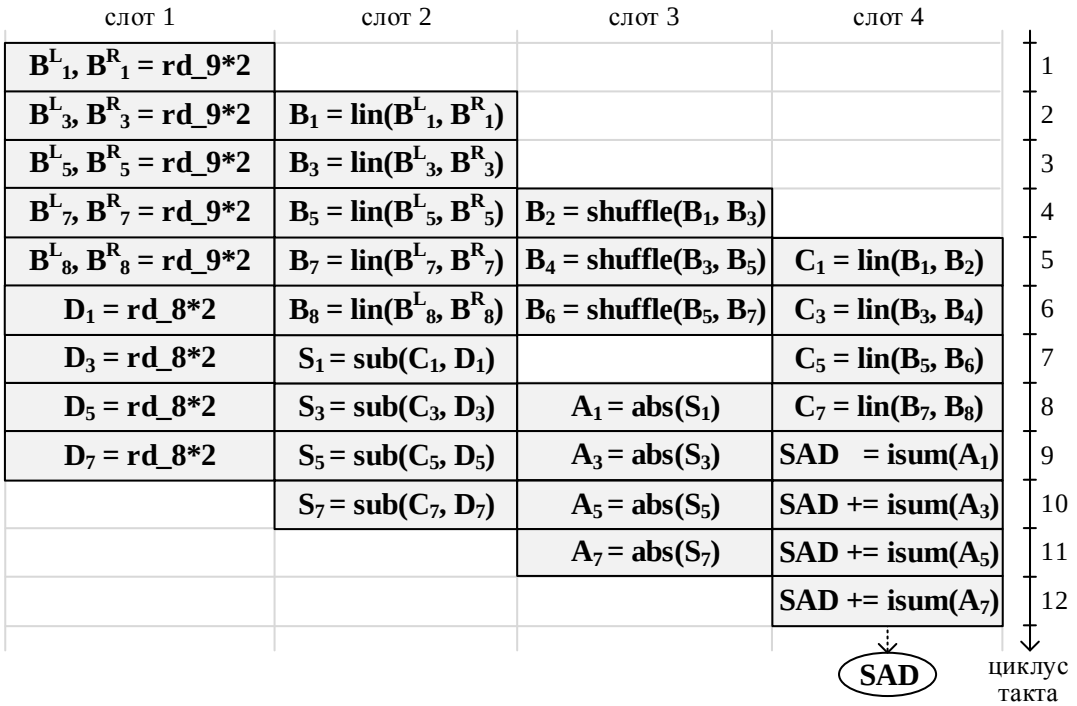
[20] Број слотова одређен је тако да их има таман довољно да аритметичко-логичке операције не представљају уско грло, а под претпоставком да је у савременим технологијама израде чипова цена логике довољно ниска да се слотови могу приуштити у потребном броју.



Слика 3.8 – Израчунавање једне 8∗8 SAD вредности, коришћењем 8∗2 начина приступа за билинеарну интерполацију блока кандидата C_{1-7}.

деведесет пиксела из паралелног меморијског подсистема, што је око 1,8 пута мање у односу на реализацију коришћењем 8∗2 начина приступа. Узимајући у обзир и циљни блок, укупно се прочита сто педесет четири пиксела, што је око 1,5 пута мање у односу на реализацију са 8∗2 начином приступа. На тај начин, поред убрзања обраде, нови 9∗2 начин приступа значајно смањује потрошњу енергије паралелног меморијског подсистема по једном израчунавању SAD функције.

Трећа варијанта илуструје реализацију SAD функције коришћењем новог 9∗3 начина приступа. Псеудокод ове варијанте је приказан на слици 3.10. Осим по коришћењу новог 9∗3 начина приступа, ова варијанта се од претходних разликује и по томе што су три операције „shuffle” замењене операцијама „lin”. То је урађено јер се блокови B_i^L и B_i^R са парним индексима, захваљујући новом 9∗3 начину приступа, читају у истом тренутку када и одговарајући блокови са



Слика 3.9 – Израчунавање једне 8∗8 SAD вредности, коришћењем 9∗2 начина приступа за билинеарну интерполацију блока кандидата C_{1-7}.

непарним индексима. Стога се линеарна интерполација блокова B_2, B_4 и B_6 може извршити у паралели са интерполацијом B_1, B_3 и B_5 блокова, респективно, што је за два циклуса такта раније него што би блокови B_2, B_4 и B_6 били добијени одговарајућим „shuffle” операцијама.

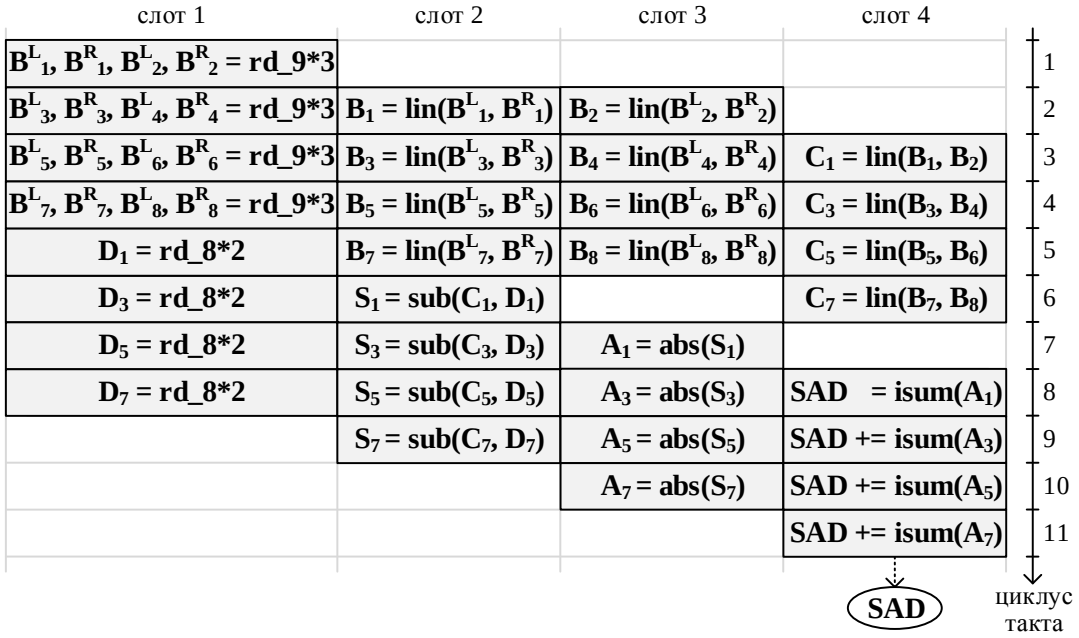
Распоред операција, којима се овај псеудокод реализује на векторској путањи са четири слота, која уместо „shuffle” има још једну „lin” функционалну јединицу у слоту три, илустрован је на слици 3.11. Извршавање билинеарне интерполације блока кандидата траје шест циклуса такта, што је два пута брже у односу на реализацију коришћењем постојећег 8∗2 начина приступа. Истовремено, то је за два циклуса такта мање и двадесет пет процената брже у односу на реализацију коришћењем новог 9∗2 начина приступа. Ово убрзање је директна последица новог 9∗3 начина приступа и нове операције читања, који су омогућили да се линеарна интерполација блокова B_2, B_4 и B_6 изврши четири и два циклуса такта раније у односу на реализације коришћењем 8∗2 и 9∗2 начина приступа, респективно. Укупно време за израчунавање SAD функције је једанаест циклуса такта, што је око педесет пет процената брже у односу на 8∗2 реализацију и десет процената брже у односу на 9∗2 реализацију. За билинеарну интерполацију кандидат блока прочита се сто осам пиксела, 1,5 пута мање

```
rd_9*3 (B^L_1, B^R_1, B^L_2, B^R_2);
rd_9*3 (B^L_3, B^R_3, B^L_4, B^R_4);
rd_9*3 (B^L_5, B^R_5, B^L_6, B^R_6);
rd_9*3 (B^L_7, B^R_7, B^L_8, B^R_8);

B_1 := lin (B^L_1, B^R_1);
B_2 := lin (B^L_2, B^R_2);
B_3 := lin (B^L_3, B^R_3);
B_4 := lin (B^L_4, B^R_4);
B_5 := lin (B^L_5, B^R_5);
B_6 := lin (B^L_6, B^R_6);
B_7 := lin (B^L_7, B^R_7);
B_8 := lin (B^L_8, B^R_8);

C_1 := lin (B_1, B_2);
C_3 := lin (B_3, B_4);
C_5 := lin (B_5, B_6);
C_7 := lin (B_7, B_8);

rd_8*2 (D_1);
rd_8*2 (D_3);
rd_8*2 (D_5);
rd_8*2 (D_7);

S_1 := sub (C_1, D_1);
S_3 := sub (C_3, D_3);
S_5 := sub (C_5, D_5);
S_7 := sub (C_7, D_7);

A_1 := abs (S_1);
A_3 := abs (S_3);
A_5 := abs (S_5);
A_7 := abs (S_7);

SAD := isum (A_1);
SAD += isum (A_3);
SAD += isum (A_5);
SAD += isum (A_7);
```

Слика 3.10 – Реализација 8∗8 SAD функције, чији су операнди: директно прочитани блок D_{1-7} и билинеарно-интерполирани блок C_{1-7}, уз читање 9∗3 блокова.

у односу на 8∗2 реализацију, али 1,2 пута више него у случају 9∗2 реализације. Уколико се у обзир узме и циљни блок, укупно се прочита сто седамдесет два пиксела по једном израчунавању SAD функције. То је 1,3 пута мање него у случају 8∗2 реализације и 1,1 пута више у односу на 9∗2 реализацију. На основу наведеног, може се закључити да 9∗3 начин приступа омогућава бржу обраду, али је енергетски мање ефикасан од 9∗2 начина приступа.

Табела 3.1 резимира претходно излагање у овој секцији, приказујући потребан број циклуса такта и број прочитаних пиксела за билинеарну интерполацију



Слика 3.11 – Израчунавање једне 8∗8 SAD вредности, коришћењем 9∗3 начина приступа за билинеарну интерполацију блока кандидата C_{1-7}.

Табела 3.1 – Број циклуса такта и прочитаних пиксела за билинеарну интерполацију једног 8∗8 блока кандидата и једно израчунавање 8∗8 SAD функције.

Варијанта реализације	Број циклуса такта	Број прочитаних пиксела
	Интерполација блока кандидата	
са познатим 8∗2	12	160
са новим 9∗2	8 (50% мање)	90 (1,8 пута мање)
са новим 9∗3	6 (100% мање)	108 (1,5 пута мање)
	Израчунавање SAD функције	
са познатим 8∗2	17	224
са новим 9∗2	12 (42% мање)	154 (1,5 пута мање)
са новим 9∗3	11 (55% мање)	172 (1,3 пута мање)

једног 8∗8 блока кандидата и једно израчунавање 8∗8 SAD функције, за сваку од три посматране варијанте реализације. У заградама је приказано умањење вредности посматраних параметара код 9∗2 и 9∗3 у односу на 8∗2 реализацију.

Као што је наведено у претходним секцијама овог поглавља, при естимацији кретања упаривањем блокова за један циљни блок се израчунава више SAD вредности, реда величине од десет до хиљаду у зависности од примењене методе естимације. Стога се у ефикасним реализацијама циљни блок чита само једном за сва израчунавања, а интерполације различитих блокова кандидата, као и израчунавања SAD вредности, изрвшавају се делимично преклопљено,

односно у режиму проточне обраде. Распоред операција при израчунавању K SAD вредности у режиму проточне обраде, уз коришћење 8∗2, 9∗2 и 9∗3 начина приступа за билинеарну интерполацију блокова кандидата, приказан је на сликама 3.12, 3.13 и 3.14, респективно.

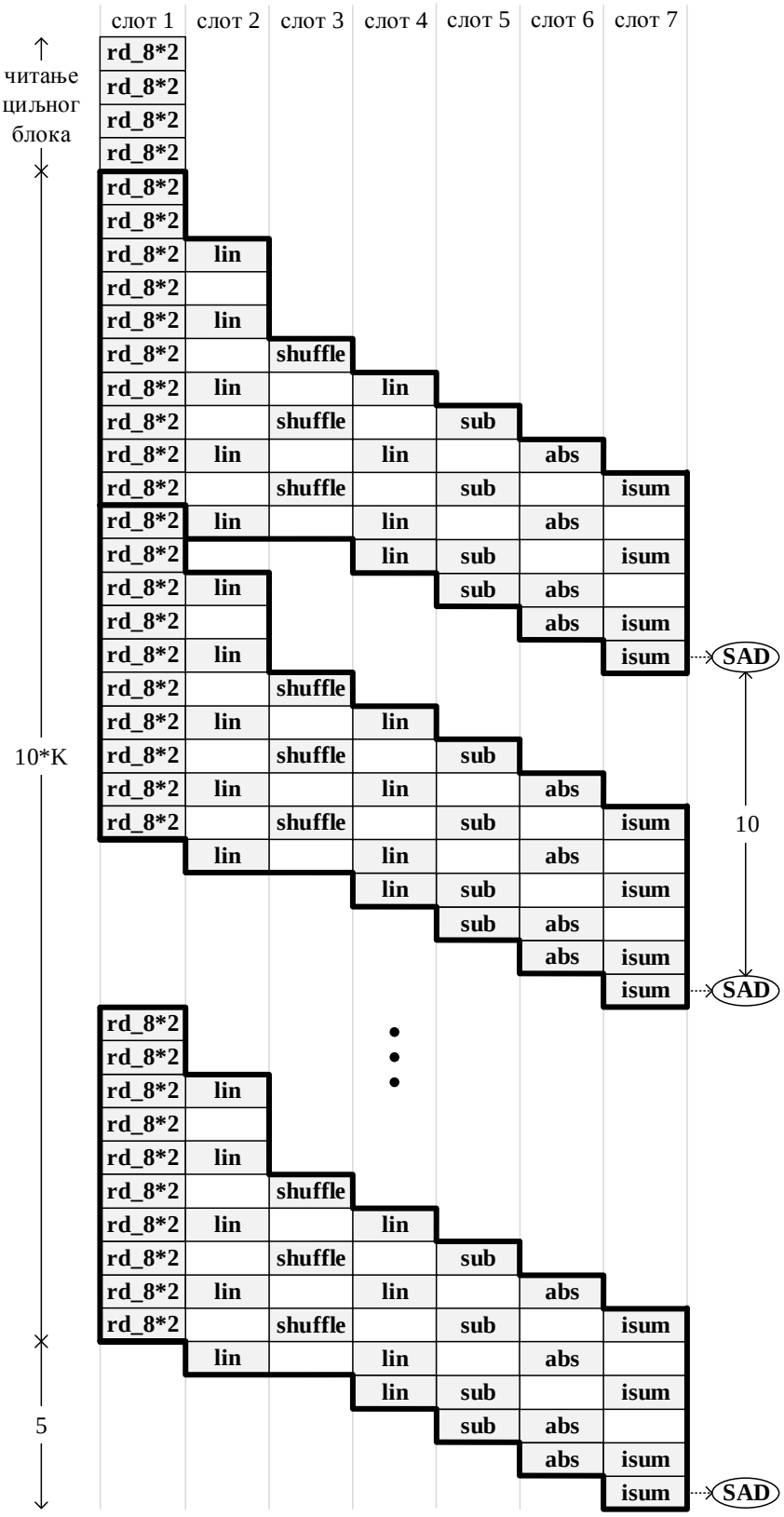
У овим случајевима векторска путања података садржи три додатна слота са функционалним јединицама „sub", „abs" и „isum", како би се у слотовима два, три и четири извршавале само операције „lin" и „shuffle". На тај начин се може боље искористити додатни инструкцијски паралелизам, настао као последица преклопљеног израчунавања више SAD вредности.

Ради прегледности слика, дебљом линијом су уоквирене операције које чине једно израчунавање SAD функције. На почетку сваке од три варијанте се прочита циљни блок извршавањем четири 8∗2 приступа меморијском подсистему. Потом се врши билинеарна интерполација блокова кандидата на идентичан начин као што је представљено на сликама 3.8, 3.9 и 3.11. Извршавање „sub", „abs" и „isum" операција у свакој од три варијанте почиње нешто раније у односу на билинеарну интерполацију јер је циљни блок већ доступан у тренутку када се заврши билинеарна интерполација блока кандидата.

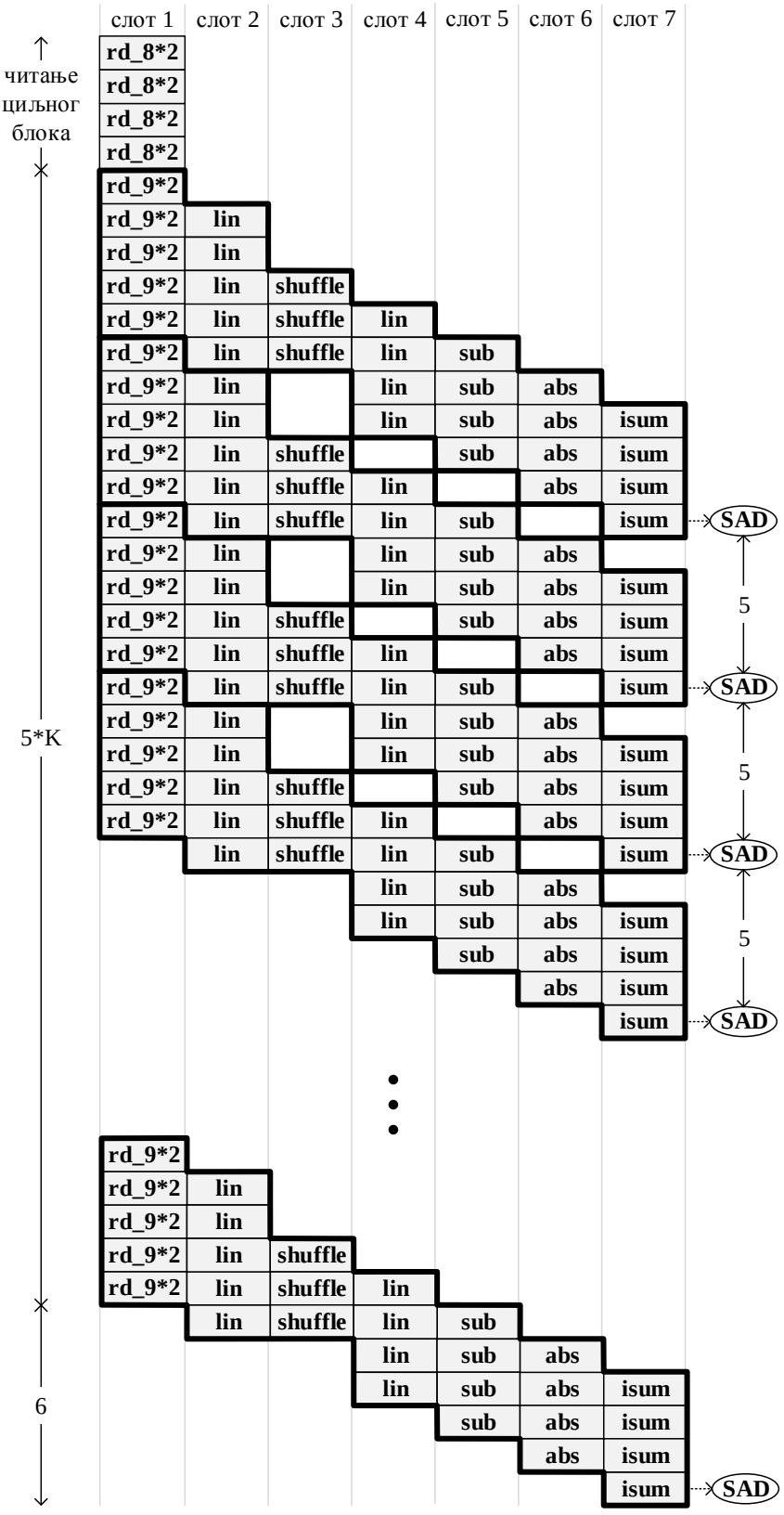
Дате слике показују да се меморијском подсистему приступа у сваком циклусу такта, извршавањем одговарајућих операција читања у слоту један. Притом, како се број приступа по једном SAD израчунавању смањује са десет на слици 3.8, преко пет на слици 3.9, до четири на слици 3.11, тако су операције у другим слотовима гушће распоређене, функционалне јединице су боље искоришћене, а брзина обраде је већа. То нам говори да приступ меморијском подсистему представља уско грло које одређује брзину обраде у све три варијанте. Тиме се потврђује наша почетна претпоставка да се смањењем броја приступа меморијском подсистему може убрзати обрада. У наставку ове секције изводимо једначине које математички потврђују тачност почетне претпоставке и дајемо графиконе које то илуструју.

Анализом распореда операција на сликама 3.8, 3.9 и 3.11 долазимо до следећих једначина, које представљају укупан број циклуса такта за израчунавање K SAD вредности, респективно:

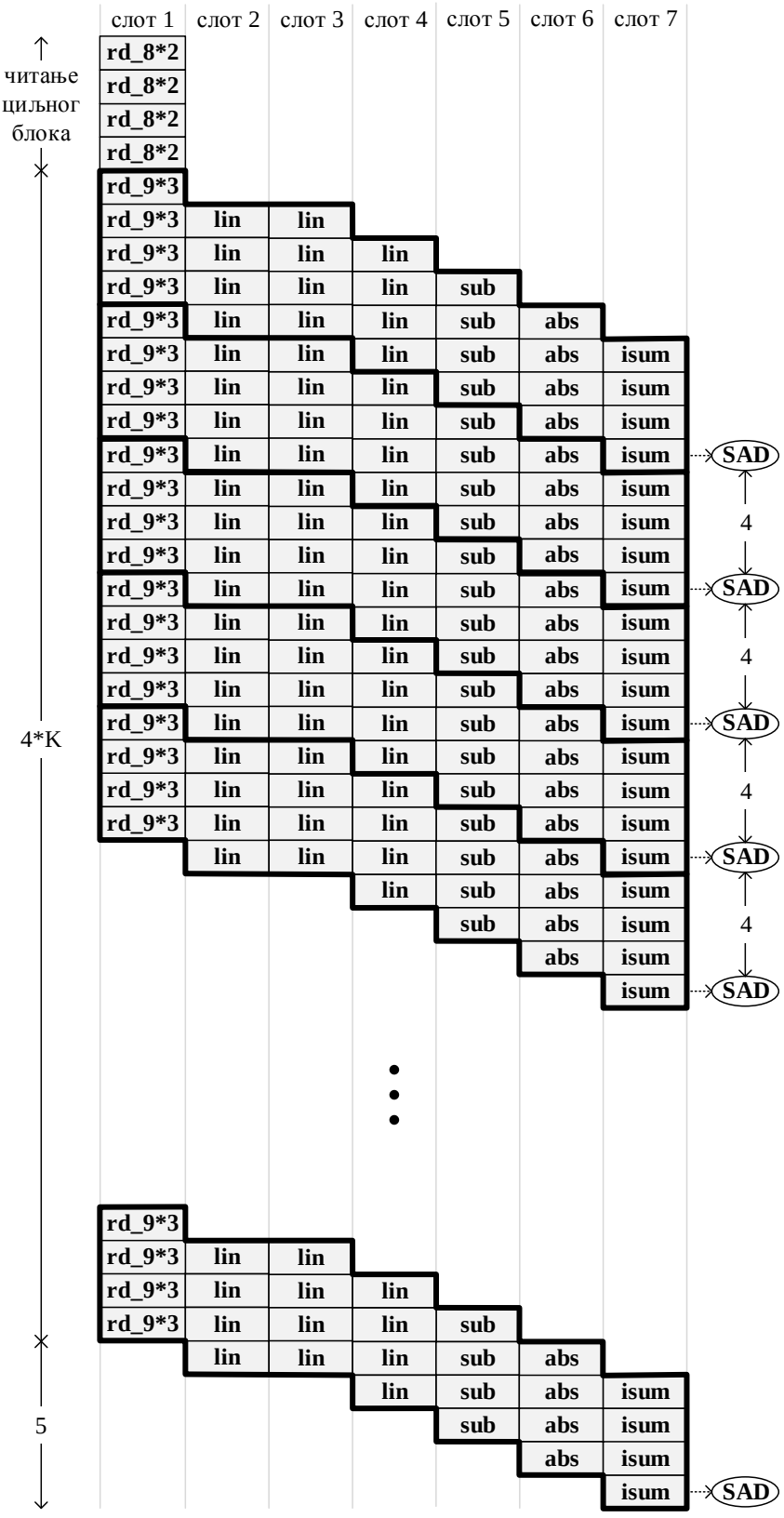
$$\begin{aligned} C_t(8*2) &= 4+10*K+5 = 10*K+9, \\ C_t(9*2) &= 4+5*K+6 = 5*K+10, \\ C_t(9*3) &= 4+4*K+5 = 4*K+9. \end{aligned} \tag{3.1}$$



Слика 3.12 – Израчунавање K 8∗8 SAD вредности, делимично преклопљено, коришћењем 8∗2 начина приступа за билинеарну интерполацију блока кандидата.



Слика 3.13 – Израчунавање K 8∗8 SAD вредности, делимично преклопљено, коришћењем 9∗2 начина приступа за билинеарну интерполацију блока кандидата.



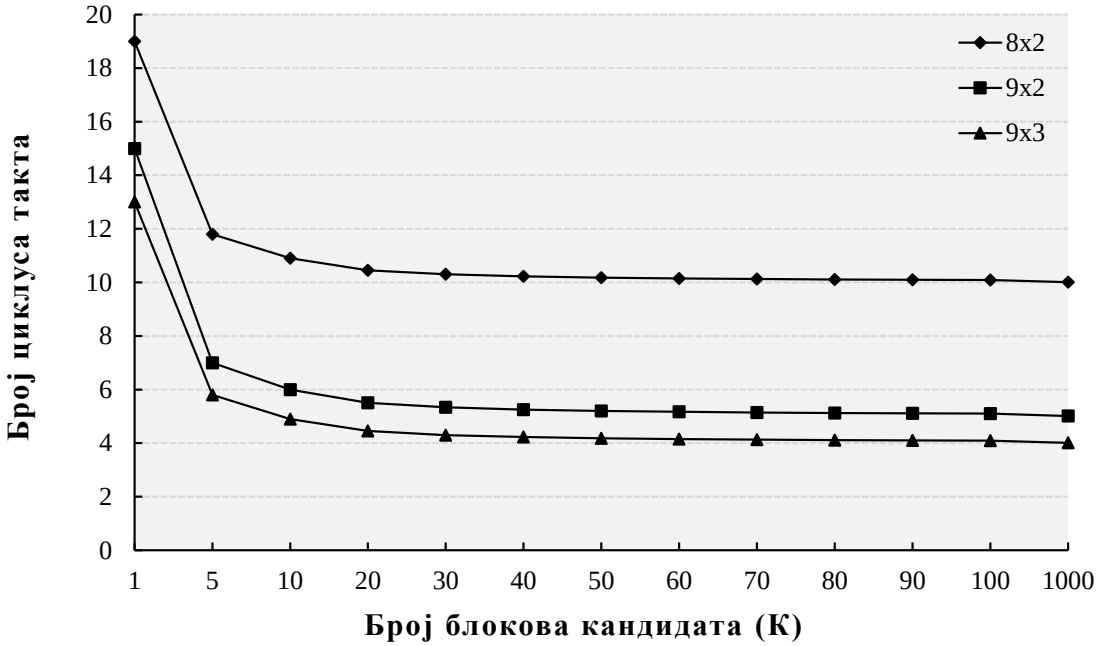
Слика 3.14 – Израчунавање K 8∗8 SAD вредности, делимично преклопљено, коришћењем 9∗3 начина приступа за билинеарну интерполацију блока кандидата.

На основу претходних израза можемо одредити просечан број циклуса такта потребних за једно SAD израчунавање:

$$\begin{aligned} C_a(8*2) &= \frac{C_t(8*2)}{K} = \frac{10*K+\ 9}{K} = 10+\frac{9}{K}, \\ C_a(9*2) &= \frac{C_t(9*2)}{K} = \frac{5*K+10}{K} = 5+\frac{10}{K}, \\ C_a(9*3) &= \frac{C_t(9*3)}{K} = \frac{4*K+\ 9}{K} = 4+\frac{9}{K}. \end{aligned} \tag{3.2}$$

Добијени изрази показују да број приступа меморијском подсистему већински одређује просечну брзину SAD израчунавања, док број блокова кандидата K утиче у мањој мери. Овакав закључак важи под претпоставком да постоји само један слот за приступ меморијском подсистему и да је број слотова за обраду довољан да они не представљају уско грло. Наведена претпоставка одговара реалности јер у савременим архитектурама процесора по правилу постоји само један слот за приступ једном меморијском подсистему, због тога што стандардно коришћена меморија може да одговори на само један захтев у једном тренутку и стога већи број слотова за приступ не би допринео брзини обраде. Такође, савремене архитектуре процесора најчешће садрже велики број јединица за обраду, захваљујући томе што су логичка кола, од којих се те јединице састоје, релативно јефтина у савременим технологијама израде чипова, у смислу заузећа површине [12]. Узимајући ово у обзир, допринос нових начина приступа је и у томе што омогућују већи инструкцијски паралелизам и боље искоришћење доступних јединица за обраду, као што илуструју слике 3.8, 3.9 и 3.11.

Слика 3.15 илуструје просечан број циклуса такта по једном SAD израчунавању за репрезентативни низ вредности параметра K и за све три варијанте реализације. Са слике се може видети да вредност просечног броја циклуса такта по SAD израчунавању врло брзо конвергира ка вредности која представља број приступа меморијском подсистему извршених за интерполацију блока кандидата.



Слика 3.15 – Просечан број циклуса такта за израчунавање SAD функције.

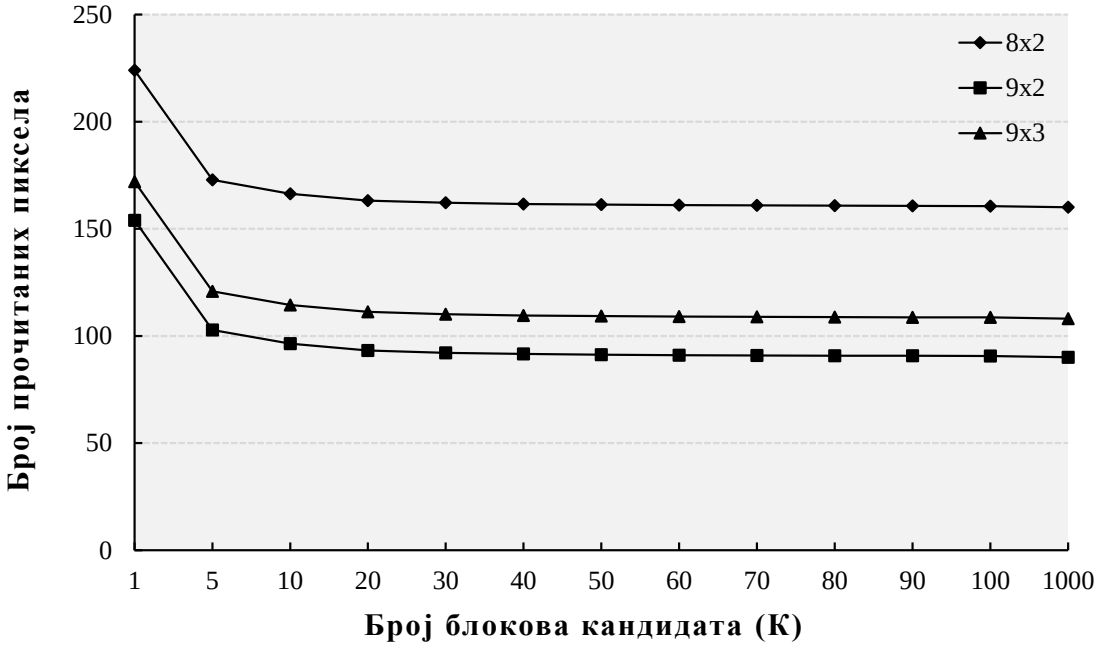
Други параметар од интереса јесте просечан број прочитаних пиксела по SAD израчунавању, што се такође може одредити на основу претходне анализе:

$$\begin{aligned} R_a(8*2) &= \frac{160*K+64}{K} = 160+\frac{64}{K}, \\ R_a(9*2) &= \frac{90*K+64}{K} = 90+\frac{64}{K}, \\ R_a(9*3) &= \frac{108*K+64}{K} = 108+\frac{64}{K}. \end{aligned} \tag{3.3}$$

Овај параметар је значајан јер представља добру апроксимацију потрошње енергије меморијског подсистема за дату обраду. Наиме, број прочитаних пиксела пропорционалан је броју приступа меморијским ћелијама, а потрошња енергије ћелија у највећој мери одређује потрошњу подсистема, док потрошња логичких кола, која се налазе око ћелија, утиче у мањој мери, без обзира на архитектуру подсистема и коришћени начин приступа.

Претходни изрази показују да просечан број прочитаних пиксела по SAD израчунавању у највећој мери одређује интерполација блока кандидата. Графикони вредности ових израза приказани су на слици 3.16.

Трећи параметар који анализирамо јесте просечно искоришћење функционалних јединица векторске путање података током израчунавања K SAD вредности. Овај параметар представља количник укупног броја извршених операција



Слика 3.16 – Просечан број прочитаних пиксела за израчунавање SAD функције.

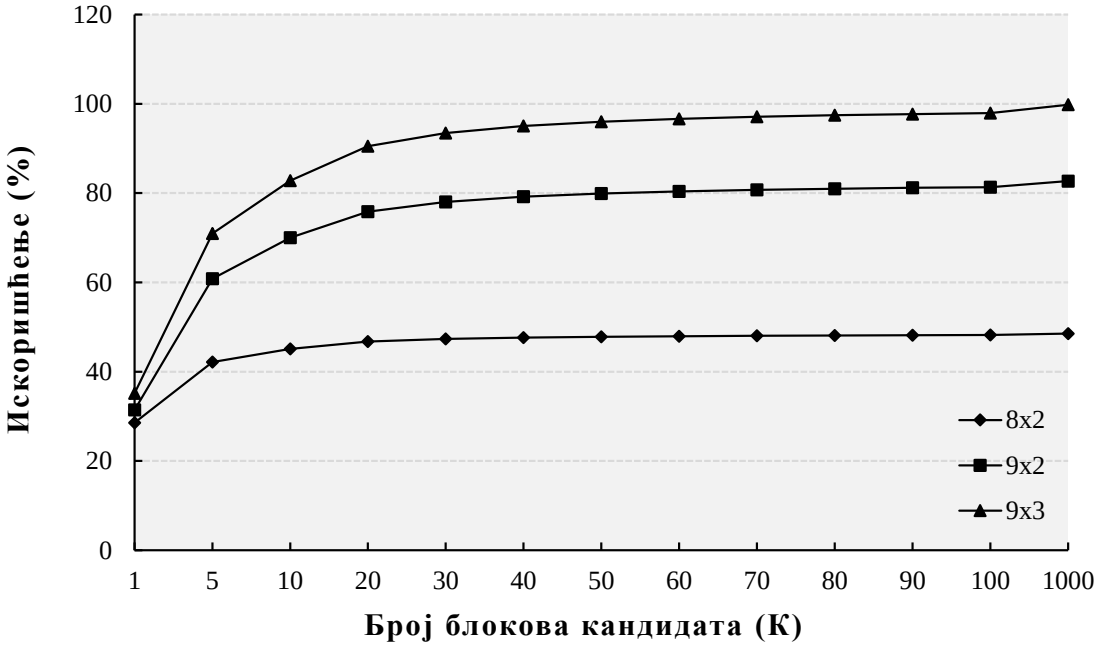
и максималног броја операција који се може извршити на датој векторској путањи података у датом броју циклуса такта. Укупан број извршених операција представљен је следећим изразима:

$$\begin{aligned} O_t(8*2) &= 4+34*K, \\ O_t(9*2) &= 4+29*K, \\ O_t(9*3) &= 4+28*K. \end{aligned} \tag{3.4}$$

Максималан број операција се добија као производ броја слотова и број циклуса такта, што у нашем случају износи $7*C_t$. Просечно искоришћење векторске путање података одређено је следећим изразима:

$$\begin{aligned} U_a(8*2) &= \frac{34*K+4}{7*(10*K+9)} = \frac{34*K+4}{70*K+63} \quad \left(\underset{K=1}{=29\%} \wedge \underset{K\to\infty}{=49\%}\right), \\ U_a(9*2) &= \frac{29*K+4}{7*(5*K+10)} = \frac{29*K+4}{35*K+70} \quad \left(\underset{K=1}{=31\%} \wedge \underset{K\to\infty}{=83\%}\right), \\ U_a(9*3) &= \frac{28*K+4}{7*(4*K+9)} = \frac{28*K+4}{28*K+63} \quad \left(\underset{K=1}{=35\%} \wedge \underset{K\to\infty}{=100\%}\right). \end{aligned} \tag{3.5}$$

Слика 3.17 илуструје просечно искоришћење векторске путање података у процентима, у зависности од броја блокова кандидата K.



Слика 3.17 – Просечно искоришћење функционалних јединица векторске путање података при израчунавању SAD функције.

На сличан начин се може одредити и просечан инструкцијски паралелизам, као количник укупног броја извршених операција O_t и броја циклуса такта C_t током којих се операције извршавају. То је једнако производу броја слотова и просечног искоришћења функционалних јединица U_a. Стога, карактеристика инструкцијског паралелизма има исти облик као просечно искоришћење функционалних јединица на слици 3.17, а дефинисана је следећим изразима:

$$
\begin{aligned}
I_a(8*2) &= \frac{O_t(8*2)}{C_t(8*2)} = \frac{34*K+4}{10*K+9} \quad \left(\underset{K=1}{=2,0} \ \wedge \ \underset{K\to\infty}{=3,4}\right), \\
I_a(9*2) &= \frac{O_t(9*2)}{C_t(9*2)} = \frac{29*K+4}{5*K+10} \quad \left(\underset{K=1}{=2,2} \ \wedge \ \underset{K\to\infty}{=5,8}\right), \\
I_a(9*3) &= \frac{O_t(9*3)}{C_t(9*3)} = \frac{28*K+4}{4*K+9} \quad \left(\underset{K=1}{=2,5} \ \wedge \ \underset{K\to\infty}{=7,0}\right).
\end{aligned}
\tag{3.6}
$$

3.2 Интерполација са компензацијом кретања

Како би омогућио брзу и енергетски ефикасну интерполацију пиксела са компензацијом кретања, на векторској путањи података чији паралелизам износи N пиксела, паралелни меморијски подсистем треба да подржи следеће начине приступа:

1. Непоравнати приступ блоку од N пиксела у више различитих пропорција, односно више различитих односа његове ширине и висине.
2. **Непоравнати приступ блоку са више од N пиксела у више различитих пропорција.**
3. Поравнати приступ реду од N и више пиксела.

Друга врста начина приступа је иновативна идеја овог рада, док су прва и трећа већ познате и подржане од стране постојећих архитектура паралелног меморијског подсистема. Следи детаљнија анализа у наредним секцијама.

3.2.1 Преглед примена

Компензација кретања представља технику интерполације нових пиксела и фрејмова у видео секвенци или секвенци слика, која за њихово креирање, поред референтних фрејмова и постојећих пиксела, користи и информације о кретању у сцени. Најпознатије примене компензације кретања су конверзија формата видеа [1] и кодовање и компресија видеа [4].

У области конверзије видео формата, у применама као што су повећање броја фрејмова у секунди, скалирање резолуције и уклањање прореда[21] при репродукцији видеа, које су раније описане у секцији 3.1.1, потребно је интерполирати нове фрејмове или недостајуће линије у полу-фрејмовима, на основу постојећих, односно референтних фрејмова и полу-фрејмова. Потешкоће приликом интерполације јављају се услед тога што се камера и објекти у сцени крећу. Уколико се то кретање не узме у обзир при интерполацији, долази до појаве визуелних недостатака као што су „замрљаност" слике, „испрекиданост" објеката, „назубљеност" ивица и „хало" ефекат, односно појава „духова" око објеката. Како би се овакви недостаци избегли и постигао висок визуелни квалитет, неопходно је извршити компензацију кретања приликом интерполације [1].

У кодовању и компресији видеа, компензација кретања представља важан део свих тренутно актуелних стандарда, као што су H.263 [116], H.264/MPEG-4 [4, 117] и H.265/HEVC [118]. Компензација кретања у овим стандардима јесте део концепта који се назива предиктивно кодовање или предикција између фрејмова[22]. У предиктивном кодовању, на страни кодера врши се естимација

[21] енг. frame-rate up-conversion, resolution scaling, and deinterlacing

[22] енг. inter-frame prediction

кретања и кодују се разлике између одговарајућих пиксела и вектори помераја уместо самих пиксела, чиме се уклања редунданса између суседних фрејмова у секвенци и тако остварује компресија видеа [4]. Аналогно томе, на страни декодера врши се интерполација са компензацијом кретања, којом се реконструишу пиксели оригиналних фрејмова на основу постојећих пиксела референтних фрејмова, кодованих разлика и кодованих вектора помераја.

3.2.2 Преглед метода

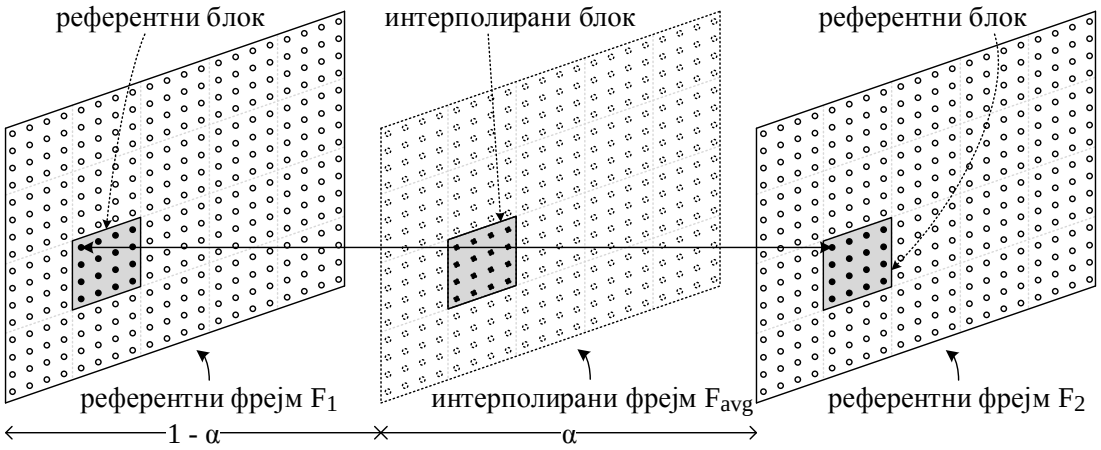
Као што је у секцији 3.1.2 наведено за естимацију кретања, компензација кретања такође може бити пиксел-рекурзивна, блоковска и објектна, на основу величине групе пиксела која се интерполира са истим параметрима кретања. Како се за естимацију кретања најчешће користе блоковске методе, тако је најчешћа и блоковска компензација кретања и стога ће посебна пажња у овом раду бити посвећена таквим методама. Гледано из другог угла, а опет у складу са извршеном естимацијом кретања, компензација кретања може бити са субпиксел или са прецизношћу од једног пиксела.

Ради јасноће даљег излагања, на слици 3.18 приказана је интерполација једног блока пиксела без компензације кретања, на основу два референтна блока. У приказаном случају нема компензације кретања, јер се за сваки интерполирани блок, као референтни узимају блокови на истој позицији у референтним фрејмовима. Вредности пиксела интерполираног блока добијају се тежинским усредњавањем одговарајућих пиксела референтних блокова, а таква метода се назива тежинско усредњавање фрејмова без компензације кретања[23] [1] и описана је следећом једначином:

$$F_{avg}(x,y) \;=\; \alpha * F_1(x,y) \;+\; (1-\alpha) * F_2(x,y),\ \alpha \in [0,1], \tag{3.7}$$

где је F_{avg} фрејм који се интерполира, F_1 и F_2 су референтни фрејмови, α и $(1-\alpha)$ су тежински фактори, а x и y су координате пиксела унутар фрејма. Тежински фактор за сваки од референтних блокова зависи од временске позиције интерполираног у односу на референтне фрејмове, тако да је вредност фактора већа уколико је интерполирани фрејм ближи одговарајућем референтном фрејму. Претходна једначина показује да подела на блокове у овој методи није неопходна, јер интерполација зависи само од координата пиксела и тежинских

[23] енг. non-motion-compensated weighted frame averaging



Слика 3.18 – Интерполација блока пиксела без компензације кретања.

фактора, али је илустрована блоковски ради аналогије са наредним методама које врше компензацију кретања.

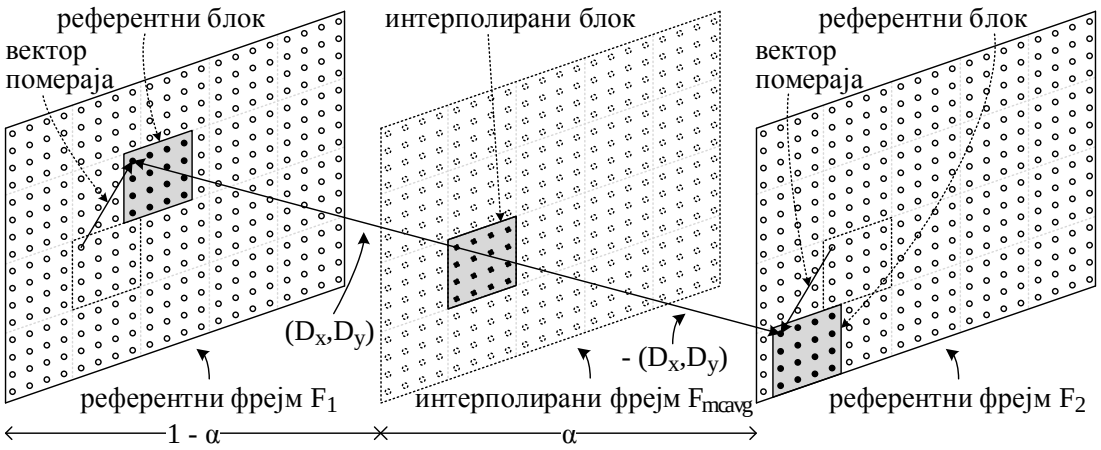
Описана метода интерполације даје визуелно добре резултате само у случајевима када се не крећу ни камера ни објекти у сцени. У супротном, када кретање постоји, постоји и велика вероватноћа да референтни блокови једног интерполираног блока припадају различитим објектима у сцени и да због тога интерполирани блок садржи визуелне недостатке попут „замрљаности" слике.

Како би се избегли овакви визуелни недостаци, неопходно је да се интерполација врши дуж трајекторије кретања објекта у сцени [1]. Другим речима, потребно је извршити интерполацију, не само у временском домену коришћењем референтних блокова из фрејмова на различитим временским позицијама у видео секвенци, већ у временско-просторном домену, тако да референтни блокови буду и са различитих просторних позиција које одговарају трајекторији кретања одговарајућег објекта између референтних фрејмова. Оваква интерполација јесте интерполација са компензацијом кретања.

Слика 3.19 приказује методу интерполације тежинским усредњавањем фрејмова са компензацијом кретања[24] [1], дефинисану следећом једначином:

$$F_{mcavg}(x,y) = \frac{1}{2} * \Big(F_1((x,y) + \alpha * (D_x, D_y)) + F_2((x,y) - (1-\alpha) * (D_x, D_y))\Big), \quad (3.8)$$
$$\alpha \in [0,1],$$

[24]енг. motion-compensated weighted frame averaging



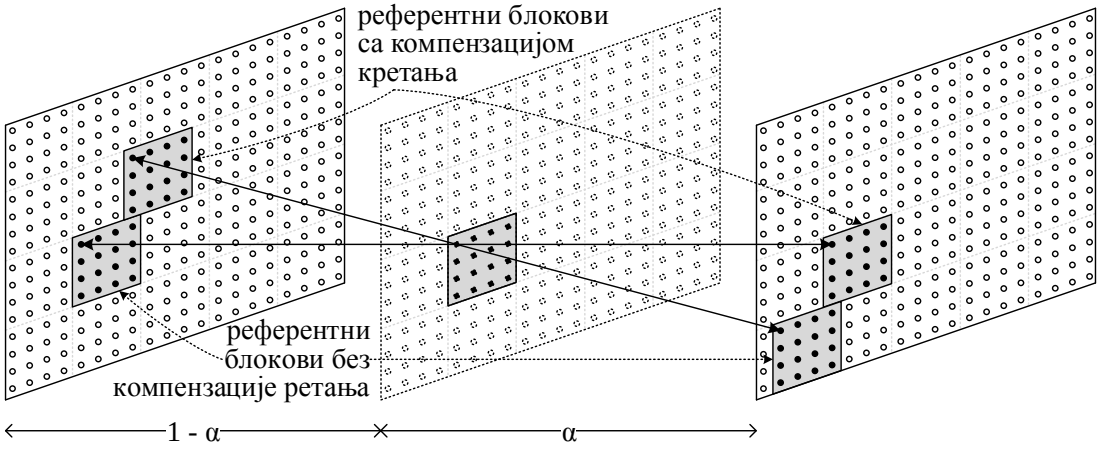
Слика 3.19 – Интерполација блока пиксела са компензацијом кретања.

где је (D_x, D_y) вектор помераја естимиран на позицији блока који се интерполира. У овом случају, референтни блокови се узимају са позиција на које показују одговарајући вектори помераја добијени блоковском естимацијом кретања и стога је подела на блокове неопходна. Овакав начин интерполације омогућава бољи визуелни квалитет у односу на усредњавање без компензације кретања, јер се сваки референтни блок узима са позиције у референтном фрејму на којој се налази одговарајући покретни објекат.

Уколико је прецизност вектора помераја један пиксел, као што приказује слика 3.2а, тада се референтни блокови директно читају из референтних фрејмова и таква компензација кретања је такође са прецизношћу од једног пиксела. У случајевима суб-пиксел вектора помераја, односно када су вектори помераја прецизнији од једног пиксела, као што илуструје слика 3.2б, референтни блокови се креирају, најчешће линеарном или билинеарном интерполацијом, на основу пиксела из референтног фрејма суседних позицији на коју показује субпиксел вектор помераја.

Метода усредњавања фрејмова са компензацијом кретања даје добре резултате у случају једноставних транслаторних кретања. Међутим, њен недостатак јесте у томе што се потпуно ослања на тачност естимације кретања, односно што није отпорна на грешке у естимацији кретања, а које су неизбежне у реалним видео сценама са сложеним кретањем [1]. Примери сложеног кретања су: објекти који се при кретању преклапају[25] тако да су они или њихови делови видљиви само у једном референтном фрејму, затим објекти који се ротирају или

[25]енг. occlusion



Слика 3.20 – Нелинаерна интерполација блока пиксела са компензацијом кретања.

на други начин мењају свој облик и покретни објекти чији се осветљај значајно мења између два фрејма.

Визуелни недостаци интерполације, који се јављају као последица грешака у естимацији кретања, такође сметају оку посматрача као и недостаци интерполације без компезације кретања [1]. Како би се избегле обе врсте недостатака и добио визуелно бољи резултат, напредније методе комбинују интерполацију са и без компензације кретања. Наиме, за сваки интерполирани блок одреди се низ блокова кандидата, „компензованих" и „некомпензованих", а потом се примењује одговарајућа функцију статистичког филтрирања којом се одабира најбољи пиксел за сваку пиксел позицију у интерполираном блоку. Одабирање се врши у зависности од естимираног кретања и поузданости естимације. Оваквe методе интерполације називају се нелинеарним, јер је сваки интерполирани пиксел нелинеарна функција више референтних пиксела.

Једна од основних нелинеарних метода, заснована на медијан функцији за филтрирање са три кандидата, назива се статичко медијан филтрирање са компензацијом кретања[26] [119, 120]. У овој методи низ кандидата медијан филтра чине два референтна „некомпензована" блока и један блок добијен тежинским усредњавањем два референтна блока са компензацијом кретања. Слика 3.20 приказује пример са сва четири референтна блока која се користе при оваквој интерполацији.

Статичко медијан филтрирање даје значајно боље резултате од тежинског усредњавања са компензацијом кретања [120], јер успешно компензује кретање

[26]енг. static median filtering

чувајући притом изглед непокретних структура, као што је на пример текст превода у филмовима. Резултати су посебно добри у сценама са мало детаља. Међутим, недостаци ове методе јављају се у случају сцена са доста детаља и текстура.

Како би се ови недостаци избегли, развијена је метода динамичког медијан филтрирања са компензацијом кретања[27] [119, 120]. Попут статичког филтрирања, динамичко филтрирање такође користи медијан функцију са три кандидата и четири референтна блока, као што је приказано на слици 3.20. Међутим, у случају динамичког филтрирања, кандидати медијан функције су два „компензована" блока и један блок добијен тежинским усредњавањем референтних блокова без компензације кретања. На тај начин фаворизују се „компензовани" кандидати.

Динамичко медијан филтрирање добре резултате даје у сценама са доста детаља и текстура, а посебно када је контраст у осветљености објеката висок [120]. Недостаци ове методе јесу тенденција да се креирају „назубљене" ивице објеката у сцени, као и ограничен број кандидата, који не могу дати добре резултате у широком опсегу реалних видео сцена.

Како су динамичко и статичко медијан филтрирање комплементарне методе у смислу њихових предности и мана, наредни корак у унапређењу интерполације јесте каскадно медијан филтрирање са компензацијом кретања[28] [120]. Улазни кандидати за каскадно медијан филтрирање јесу резултати статичког, динамичког и других сличних филтара, а циљ је да се њиховом комбинацијом омогући одабир најбољег кандидата у различитим ситуацијама које се јављају у реалним видео сценама.

Друга класа метода за интерполацију са компензацијом кретања [121–123], такође заснована на филтрирању, кандидате узима са позиција у референтном фрејму из блиског окружења позиције на коју показује вектор помераја. На тај начин тежи се постизању веће отпорности на непрецизност естимације кретања. Новије методе интерполацију углавном врше на неки од претходно описаних или њима сличних начина, а истраживачки труд се улаже у развој метода за обраду вектора помераја добијених естимацијом кретања, како би се унапред елиминисали погрешни и непоуздани кандидати [124–131].

Претходно описане методе интерполације пиксела углавном се користе у

[27] енг. dynamic median filtering

[28] енг. cascaded median filtering

области конверзије видео формата. У наставку ове секције описана је метода која се примењује за кодовање и компресију видеа, у оквиру актуелних H.264 и MPEG-4 стандарда [132].

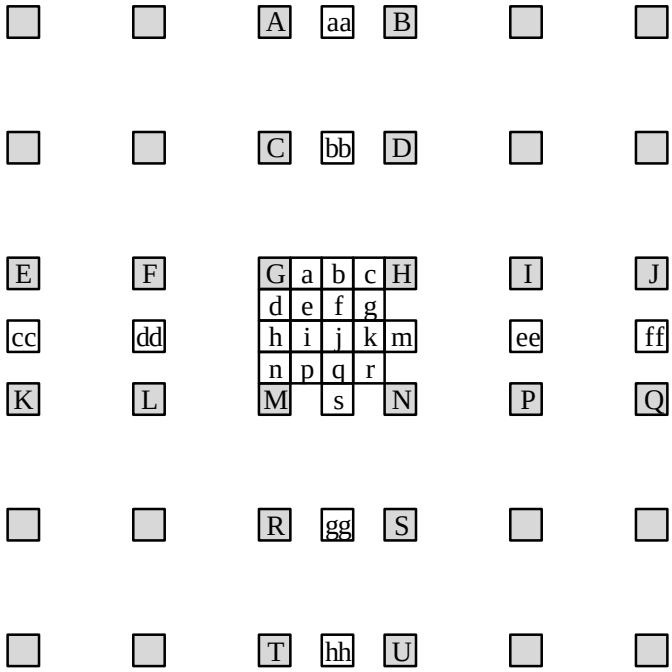
У свим тренутно актуелним стандардима за кодовање и компресију видеа, укључујући и H.264/MPEG-4, користи се блоковска интерполација са компензацијом кретања [4]. Притом се, у терминологији ових стандарда, уместо појма блока користи појам макроблока. Величина једног макроблока јесте 16∗16 пиксела, односно макроблок сачињавају један блок од 16∗16 одбирака луминансе и два блока од 8∗8 одбирака хроминансе[29].

За разлику од конверзије видео формата, где непоузданост естимације кретања представља најчешћи разлог за појаву визуелних недостатака у интерполираном фрејму, основна потешкоћа компензације кретања у кодовању и компресији видеа јесте то што се границе објеката у сцени не поклапају са границама макроблокова. Стога се према H.264/MPEG-4 стандарду, сваки макроблок може поделити на мање блокове, 16∗8, 8∗16 или 8∗8 за луминансу и 8∗4, 4∗8 или 4∗4 за хроминансу, како би се блокови боље уклопили са границама објеката. Подела на мање блокове врши се само у деловима фрејма са доста детаља, текстура и ивица, док се у осталим деловима фрејма задржава максимална величина макроблока у циљу остваривања најбољег компромиса између визуелног квалитета и рачунске сложености, који расту са смањењем блока.

При интерполацији једног блока уз компензацију кретања, видео декодер прво одређује референтни блок на основу кодованог вектора помераја и референтног фрејма. Потом се „компензовани" блок израчунава сабирањем референтног блока са блоком разлика, који је претходно одређен и кодован на страни видео кодера. У H.264/MPEG-4 стандарду естимација и компензација кретања врше се са суб-пиксел прецизношћу, што значи да се референтни блок мора креирати на основу постојећих пиксела у референтном фрејму, као што је раније већ поменуто. Прецизност вектора помераја једнака је једној четвртини пиксела, што значи да вектор помераја може да покаже на саме пикселе или на једну од три позиције између два суседна пиксела [4, 117].

За креирање референтног блока хроминансе користи се билинеарна интерполација без обзира на коју позицију показује вектор помераја [4, 117]. За креирање референтног блока луминансе примењује се сложенија функција, илустро-

[29] луминанса (Y) представља осветљај, а хроминанса (U и V) боју у YUV формату пиксела, који је погоднији за обраду од RGB (енг. Red, Green, Blue) формата.



Слика 3.21 – Интерполација пиксела у H.264/MPEG-4 стандарду. (извор [4]).

вана за један пиксел на слици 3.21. Пиксели референтног фрејма осенчени су и означени великим словима, могуће позиције интерполирације пиксела означене су малим словима, а са два слова означене су помоћне вредности коришћене при интерполацији. За креирање блока чији се пиксели налазе на половини растојања између два пиксела у референтном фрејму, као што се пиксел b налази између G и H, а пиксел h између G и M, користи се једнодимензионални просторни филтар шестог реда[30] у хоризонталном и вертикалном правцу [4,117]. За пикселе на позицијама b и h филтар је дефинисан следећим једначинама [4,117]:

$$\begin{aligned} b_1 &= (E - 5 * F + 20 * G + 20 * H - 5 * I + J) \\ h_1 &= (A - 5 * C + 20 * G + 20 * M - 5 * R + T) \\ b &= (b_1 + 16) : 32 \\ h &= (h_1 + 16) : 32, \end{aligned} \tag{3.9}$$

где су b_1 и h_1 помоћне вредности. Блокови чији се пиксели налазе на позицији означеној са j на слици 3.21, добијају се такође применом филтра шестог реда [4, 117]:

$$\begin{aligned} j_1 &= (cc - 5 * dd + 20 * h_1 + 20 * m_1 - 5 * ee + ff) \\ j &= (j_1 + 512) : 1024, \end{aligned} \tag{3.10}$$

[30]енг. 6-tap FIR (Finite Impulse Response) filter

где се вредности cc, dd, m_1, ee и ff добијају на исти начин као h_1. Блокови чији се пиксели налазе на четвртини растојања између два пиксела у референтном фрејму, на позицијама означеним са a, c, d, n, f, i, k и q добијају се усредњавањем два најближа суседна пиксела [4, 117]. На пример, пиксел на позицији означеној са a, одговара следећем изразу:

$$a \;=\; (G + b + 1) : 2. \tag{3.11}$$

Блокови чији се пиксели налазе на некој од преосталих позиција, e, g, p или r, добијају се усредњавањем два најближа дијагонално суседна пиксела [4, 117], као што је на пример:

$$e \;=\; (b + h + 1) : 2. \tag{3.12}$$

3.2.3 Потребни начини приступа

У описаним методама интерполације пиксела са компензацијом кретања за примену у конверзији видео формата и кодовању и компресији видеа, рачунску сложеност у највећој мери одређују функције за креирање референтног блока. Стога су ове функције од посебног интереса за даљу анализу у наставку поглавља. Фокус даље анализе јесу обрасци приступа пикселима, а циљ је одређивање потребних начина приступа паралелном меморијском подсистему. Прво се анализира област конверзије видео формата, а потом и H.264/MPEG-4 стандард за кодовање и компресију видеа.

У случају конверзије видео формата и компензације кретања са прецизношћу од једног пиксела, вектори помераја показују на неки од пиксела из референтног фрејма, као што је илустровано на слици 3.2а. Сваки референтни блок се стога чита директно из одговарајућег референтног фрејма. У супротном, уколико је компензација кретања са суб-пиксел прецизношћу, вектори помераја могу показивати између пиксела референтног фрејма, као што приказује слика 3.2б. Том приликом се референтни блок креира на основу пиксела из референтног фрејма, најчешће линеарном или билинеарном интерполацијом.

На основу претходно реченог можемо закључити да се обрасци приступа пикселима у компензацији кретања за конверзију видео формата поклапају са обрасцима приступа у естимацији кретања упаривањем блокова. Као што је наведено у секцији 3.1.3, ти обрасци су:

1. Читање блокова пиксела који се користе као референтни блокови у компензацији кретања са прецизношћу од једног пиксела.

2. Читање блокова пиксела који се користе за креирање референтних блокова у компензацији кретања са суб-пиксел прецизношћу.

3. Преноси дводимензионалних низова пиксела, односно делова фрејмова из меморије ван чипа у паралелни меморијски подсистем на чипу.

Оваквим обрасцима приступа одговарају исти начини приступа меморијском подсистему и исте операције читања из подсистема као и у случају естимације кретања упаривањем блокова, који су детаљно описани у секцији 3.1.3, а овде поново наведени:

1. Непоравнати приступ блоку и реду од N пиксела у више пропорција, где N означава паралелизам векторске путање података. Примери таквих начина приступа су 4∗4, 8∗2 и 16∗1 за $N = 16$, затим 8∗4, 16∗2 и 32∗1 за $N = 32$ и 8∗8, 16∗4, 32∗2 и 64∗1 за $N = 64$. Овакви начини приступа су познати и подржани од стране постојећих паралелених меморијских подсистема.

2. Непоравнати приступ блоку и реду са више од N пиксела, односно блоку и реду са свим пикселима који су потребни за интерполацију блока и реда од N пиксела. Примери оваквих начина приступа у случају хоризонталне линеарне интерполације су 5∗4, 9∗2 и 17∗1 за $N = 16$, затим 9∗4, 17∗2 и 33∗1 за $N = 32$ и 9∗8, 17∗4, 33∗2 и 65∗1 за $N = 64$. У случају билинеарне интерполације то су 5∗5, 9∗3 и 17∗2 за $N = 16$, затим 9∗5, 17∗3 и 33∗2 за $N = 32$ и 9∗9, 17∗5, 33∗3 и 65∗2 за $N = 64$. Оваквим начинима приступа одговарају операције читања из подсистема које прочитани блок или ред са више од N пиксела поделе на више блокова или редова од N пиксела, на начин погодан за даљу обраду од стране стандардних функционалних јединица чији паралелизам јесте N пиксела. **Овакви начини приступа и операције читања из паралеленог меморијског подсистема представљају један од иновативних доприноса овог рада**.

3. Поравнати приступ реду од N или више пиксела, подударан са начином приступа меморији ван чипа, који је такође познат и подржан од стране постојећих паралелених меморијских подсистема.

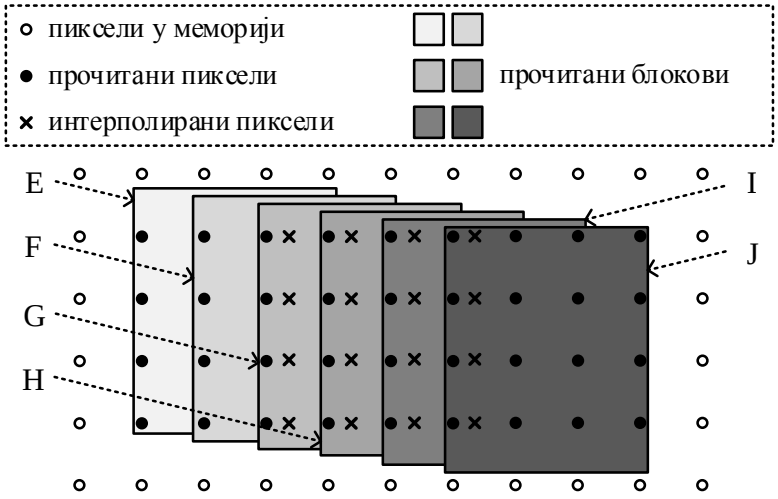
У случају H.264/MPEG-4 стандарда за кодовање и компресију видеа, референтни блокови се креирају на основу пиксела из референтних фрејмова.

Референтни блокови хроминансе креирају се билинеарном интерполацијом, а обрасци приступа пикселима и потребни начини приступа паралелном меморијском подсистему исти су као за билинеарну интерполацију у конверзији видео формата. Са друге стране, референтни блокови луминансе креирају се функцијом заснованом на просторном филтру шестог реда, који је описан једначинама 3.9 и 3.10 и функцији усредњавања, која је дефинисана једначинама 3.11 и 3.12. У наставку анализирамо обрасце приступа пикселима и дефинишемо начине приступа меморијском подсистему, потребне за ефикасно креирање луминансе референтних блокова.

1. **Образац приступа**: Читање блокова пиксела који се користе као операнди за креирање референтних блокова луминансе. Број пиксела једног референтног блока јесте умножак од N и стога се његово креирање дели на креирање више непреклопљених блокова или редова од N пиксела[31]. Како је за креирање једног пиксела потребно прочитати шест хоризонтално или вертикално суседних пиксела према једначинама 3.9 и 3.10, тако је за креирање блока или реда од N пиксела потребно прочитати блок или ред који има пет колона или линија више у односу на креирани блок или ред од N пиксела, респективно. За реализацију једначина 3.11 и 3.12 потребно је само извршити одговарајуће аритметичке операције, без читања додатних пиксела.

   Нови начини приступа: Непоравнати приступ блоку или реду са више од N пиксела, односно блоку или реду са свим пикселима који су потребни за интерполацију блока или реда од N пиксела. Примери оваквих начина приступа у случају хоризонталног филтра, који дефинише вредност b_1 у једначини 3.9, су 9∗4, 13∗2 и 21∗1 за $N = 16$, затим 13∗4 и 21∗2 за $N = 32$, као и 13∗8 и 21∗4 за $N = 64$. У случају вертикалног филтра, који дефинише вредност h_1 у једначини 3.9, примери потребних начина приступа су 4∗9, 2∗13 и 1∗21 за $N = 16$, затим 4∗13 и 2∗21 за $N = 32$, као и 8∗13 и 4∗21 за $N = 64$. У случају хоризонталног и вертикалног филтра, који дефинише вредност j_1 у једначини 3.10 потребни начини приступа су 9∗9, 13∗7 и 21∗6 за $N = 16$, затим 13∗9 и 21∗7 за $N = 32$, као и 13∗13 и 21∗9 за $N = 64$. **Ови начини приступа паралелном меморијском подсистему представљају један од иновативних доприноса овог рада**.

[31] Референтни блокови могу бити величине 16∗16, 16∗8, 8∗16 или 8∗8 пиксела, а стандардне вредности N су 16, 32 и 64.



Слика 3.22 – Подела блока од 9∗4 пиксела на 4∗4 блокове, који се читају за интерполацију 4∗4 блока хоризонталним филтром шестог реда у H.264/MPEG-4 стандарду.

Нове операције читања и уписа: Када прочитани блок или ред са више од N пиксела стигне на векторску путању података, односно у јединицу за читање и упис у паралелни меморијски подсистем, он се дели на више блокова или редова од N пиксела, који се прослеђују на даљу обраду другим јединицама чији је паралелизам N пиксела. На тај начин, довољан је један приступ меморијском подсистему за интерполацију блока или реда од N пиксела, док векторска путања података не треба да буде шира од N пиксела, а самим тим ни скупља у смислу заузећа површине на чипу. Модификација путање података може бити потребна само у смислу додавања нових функционалних јединица, како би се искористио већи инструкцијски паралелизам добијен новим операцијама читања из меморијског подсистема.

Подела блока или реда зависи од филтра и врши се тако да добијени блокови или редови од N пиксела буду погодни за даљу обраду. На пример, у случају хоризонталног филтра, један 9∗4 блок дели се на шест 4∗4 блокова померених хоризонтално за по једну пиксел колону, онако како је приказано на слици 3.22. Тако добијени блокови директно се користе као операнди за израчунавање вредности хоризонталног филтра, односно нису потребне додатне операције за припрему операнада. Слично томе, у случају вертикалног филтра, један 4∗9 блок дели се на шест 4∗4 блокова, померених вертикално за по једну пиксел линију. У случају хоризонта-

лног и вертикалног филтра један 9∗9 блок дели се на тридесет шест 4∗4 блокова, померених хоризонтално за по један пиксел и вертикално за по једну пиксел линију.

Овакве операције читања из паралелног меморијског подсистема такође представљају један од иновативних доприноса овог рада.

2. **Образац приступа**: Читање блокова разлика који се сабирају са референтним блоковима, како би се добили интерполирани блокови фрејма који се креира.
   Начини приступа: Поравнати приступ блоку и реду од N пиксела у више пропорција, као што су 4∗4, 8∗2 и 16∗1 за $N = 16$, затим 8∗4 и 16∗2 за $N = 32$ и 8∗8 и 16∗4 за $N = 64$.
   Као што је раније наведено, овакви начини приступа су познати и подржани од стране постојећих паралелених меморијских подсистема.

3. **Образац приступа**: Преноси дводимензионалних низова пиксела, односно делова фрејмова из меморије ван чипа у паралелни меморијски подсистем на чипу.
   Начини приступа: Поравнати приступ реду од N или више пиксела, подударан са начином приступа меморији ван чипа, који је такође познат и подржан од стране постојећих паралелених меморијских подсистема.
   Детаљни описи овог обрасца и начина приступа дати су раније у секцији 3.1.3.

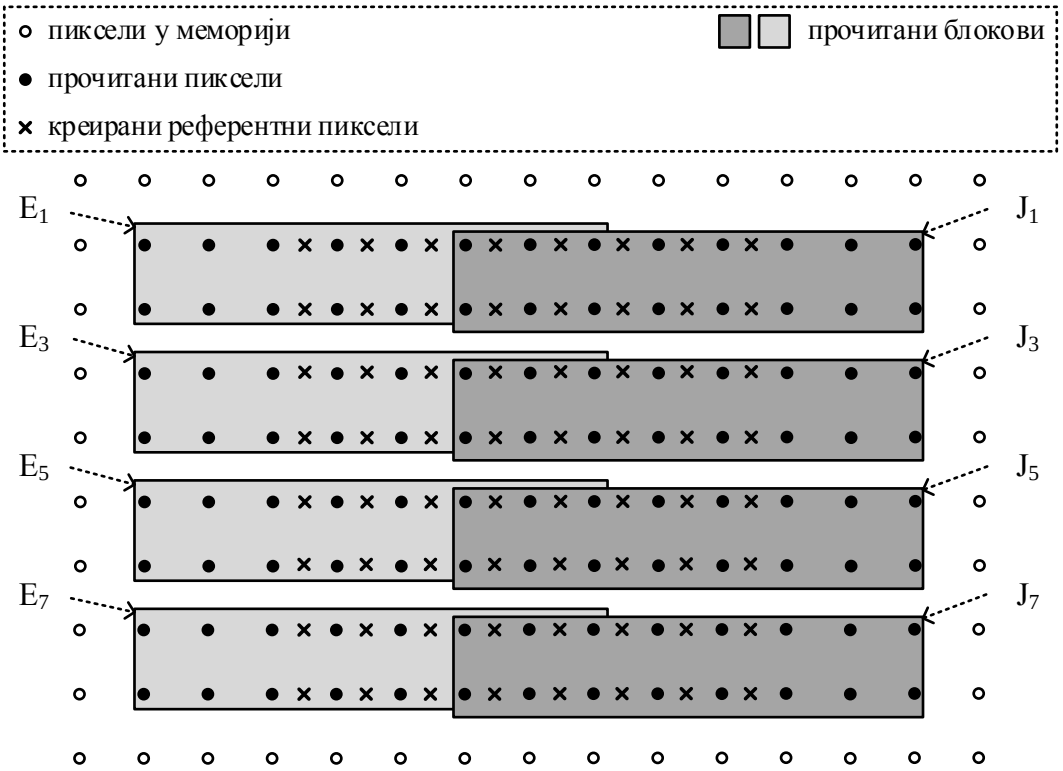
3.2.4 Реализације критичне функције

Ова секција илуструје и упоређује неколико варијанти реализације функције за H.264/MPEG-4 интерполацију 8∗8 блокова луминансе, коришћењем следећих начина приступа паралелном меморијском подсистему:

1. постојећих 8∗2 и 2∗8 и
2. нових 13∗2 и 2∗13.

Циљ је да се прикажу предности нових у односу на постојеће начине приступа, као и нових у односу на постојеће операције читања и уписа у паралелни меморијски подсистем, за примену у H.264/MPEG-4 стандарду.

За почетак, посматрају се две варијанте реализације, у којима се референтни блок интерполира на позицији b приказаној на слици 3.21, односно посматрају



Слика 3.23 – Подела блока од 13∗8 пиксела на осам 8∗2 блокова, који се читају за креирање референтног 8∗8 блока хоризонталним филтром шестог реда.

се две реализације хоризонталног филтра шестог реда дефинисаног једначином 3.9. У обе варијанте, 8∗8 блок разлика чита се извршавањем четири 8∗2 приступа, као што је раније илустровано на слици 2.4 у секцији 2.2.

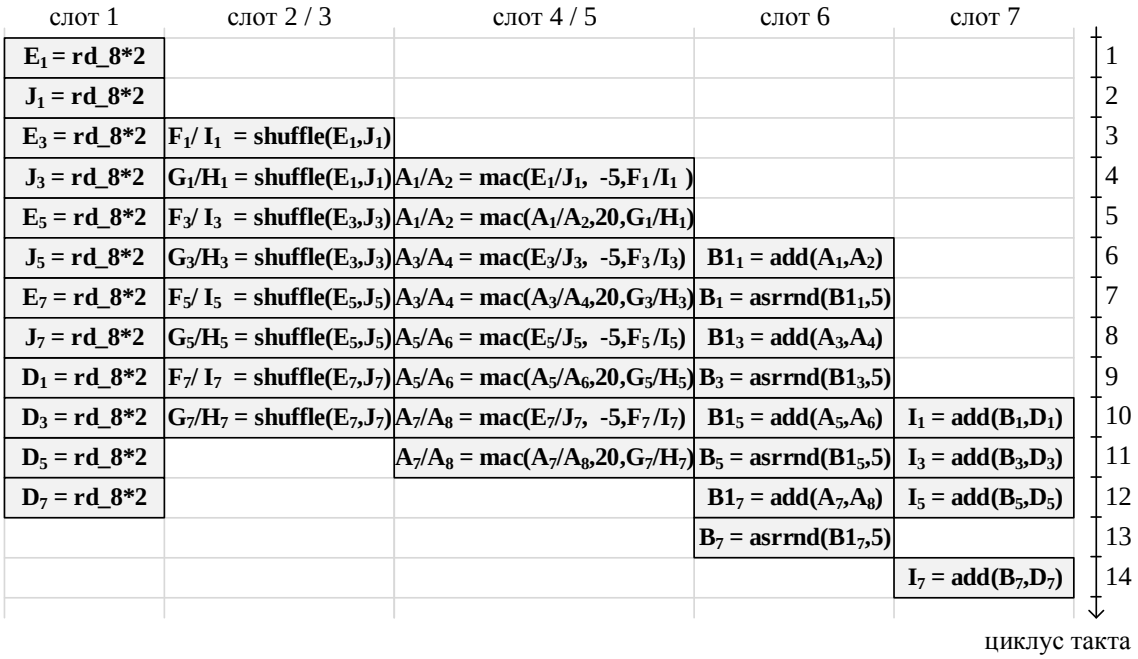
У првој варијанти, која користи познати 8∗2 начин приступа, потребно је извршити осам операција читања за креирање референтног блока, на начин који приказује слика 3.23. Псеудокод програма којим се реализује креирање референтног блока, а потом и израчунавање вредности интерполираног блока, приказан је на слици 3.24. Прво се прочитају потребни 8∗2 блокови E_i и J_i[32], $i \in \{1, 3, 5, 7\}$, операцијама „rd_8∗2" и креирају блокови F_i, G_i, H_i и I_i операцијама „shuffle" чији се излазни 8∗2 блок добија мешањем два улазна 8∗2 блока. Додатни операнди „shuffle" операције, који дефинишу како се мешају улазни блокови, изостављени су ради једноставности илустрације. Референтни блокови B_i добијају се низом „mac", „add" и „asrrnd" операција. Стандардна операција „mac" сабира први операнд са производом другог и трећег, операција „add" враћа збир своја два операнда, а операција „asrrnd" врши аритметичко по-

[32]Блокови у псеудокоду означени су у складу са позицијама њихових горњих левих пиксела, а према слици 3.21.

```
rd_8*2 (E1);
rd_8*2 (J1);
rd_8*2 (E3);
rd_8*2 (J3);
rd_8*2 (E5);
rd_8*2 (J5);
rd_8*2 (E7);
rd_8*2 (J7);

F1 := shuffle (E1, J1);   I1 := shuffle (E1, J1);
G1 := shuffle (E1, J1);   H1 := shuffle (E1, J1);
F3 := shuffle (E3, J3);   I3 := shuffle (E3, J3);
G3 := shuffle (E3, J3);   H3 := shuffle (E3, J3);
F5 := shuffle (E5, J5);   I5 := shuffle (E5, J5);
G5 := shuffle (E5, J5);   H5 := shuffle (E5, J5);
F7 := shuffle (E7, J7);   I7 := shuffle (E7, J7);
G7 := shuffle (E7, J7);   H7 := shuffle (E7, J7);

A1 := mac (E1, -5, F1);   A2 := mac (J1, -5, I1);
A1 := mac (A1, 20, G1);   A2 := mac (A2, 20, H1);
A3 := mac (E3, -5, F3);   A4 := mac (J3, -5, I3);
A3 := mac (A3, 20, G3);   A4 := mac (A4, 20, H3);
A5 := mac (E5, -5, F5);   A6 := mac (J5, -5, I5);
A5 := mac (A5, 20, G5);   A6 := mac (A6, 20, H5);
A7 := mac (E7, -5, F7);   A8 := mac (J7, -5, I7);
A7 := mac (A7, 20, G7);   A8 := mac (A8, 20, H7);

B11 := add (A1, A2);
B13 := add (A3, A4);
B15 := add (A5, A6);
B17 := add (A7, A8);

B1 := asrrnd (B11, 5);
B3 := asrrnd (B13, 5);
B5 := asrrnd (B15, 5);
B7 := asrrnd (B17, 5);

rd_8*2 (D1);
rd_8*2 (D3);
rd_8*2 (D5);
rd_8*2 (D7);

I1 := add (B1, D1);
I3 := add (B3, D3);
I5 := add (B5, D5);
I7 := add (B7, D7);
```

Слика 3.24 – Псеудокод програма за интерполацију једног 8∗8 блока луминансе на позицији *b* дефинисаној H.264/MPEG-4 стандардом, коришћењем познатог 8∗2 начина приступа меморијском подсистему при креирању референтног блока B_{1-7}.

мерање у десно, тако да се збир прва два операнда помера за вредност трећег. Након што се одреде референтни блокови, прочитају се 8∗2 блокови разлика D_i и израчунају вредности интерполираних блокова I_i сабирањем B_i и D_i.



Слика 3.25 – Распоред операција при интерполацији једног 8∗8 блока луминансе на позицији *b* дефинисаној H.264/MPEG-4 стандардом, коришћењем познатог 8∗2 начина приступа меморијском подсистему при креирању референтног блока B_{1-7}.

Распоред наведених операција при извршавању на векторској путањи података, која има седам слотова са функционалним јединицама[33], приказан је на слици 3.25. Ради смањења ширине слике, „shuffle" операције које се извршавају у паралели у слотовима два и три, као и „mac" операције које се извршавају у паралели у слотовима четири и пет, приказане су једним правоугаоником. На пример, $A_1/A_2 := mac\ (E_1/J_1, -5, F_1/I_1)$ означава две „mac" операције: $A_1 := mac\ (E_1, -5, F_1)$ у слоту четири и $A_2 := mac\ (J_1, -5, I_1)$ у слоту пет, које се извршавају у паралели у четвртом циклусу такта.

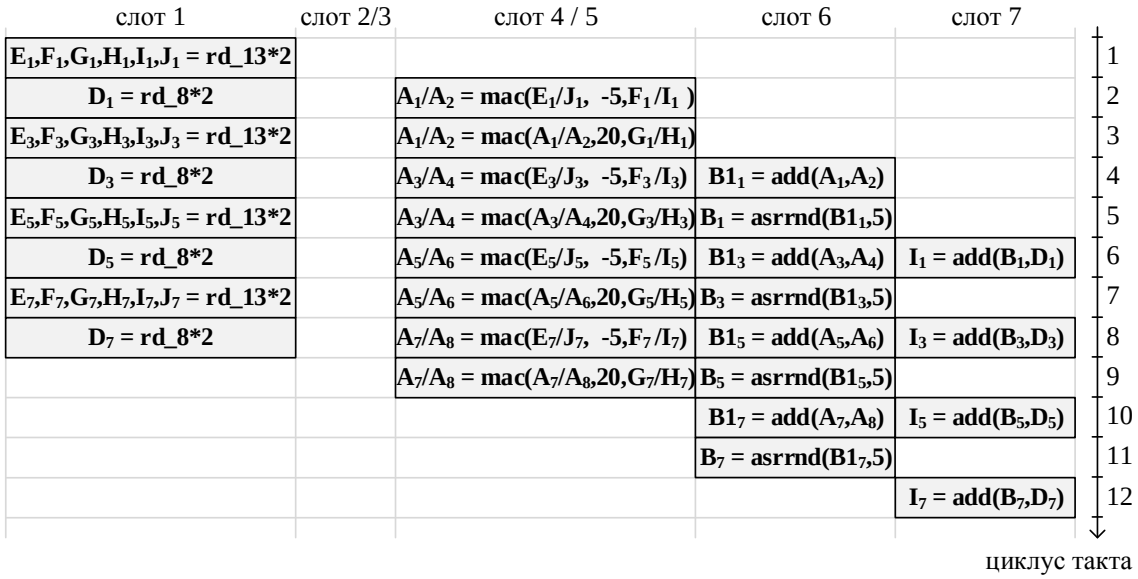
На основу ове слике може се закључити да креирање референтног 8∗8 блока траје тринаест, а интерполација 8∗8 блока луминансе четрнаест циклуса такта. Притом се, за креирање референтног 8∗8 блока, изврши осам 8∗2 операција читања из паралелног меморијског система и прочита сто двадесет осам пиксела, од чега су сто четири различита, а двадесет четири пиксела се прочитају два пута. Уколико се у обзир узму и блокови разлика, за интерполацију једног 8∗8 блока прочита се укупно сто деведесет два пиксела.

[33]Као и у претходној секцији, број слотова одређен је тако да их има таман довољно да аритметичко-логичке операције не представљају уско грло, а под претпоставком да је у савременим технологијама израде чипова цена логике довољно ниска да се слотови могу приуштити у потребном броју.

```
rd_13*2 (E1, F1, G1, H1, I1, J1);
rd_13*2 (E3, F3, G3, H3, I3, J3);
rd_13*2 (E5, F5, G5, H5, I5, J5);
rd_13*2 (E7, F7, G7, H7, I7, J7);

A1 := mac (E1, -5, F1);   A2 := mac (J1, -5, I1);
A1 := mac (A1, 20, G1);   A2 := mac (A2, 20, H1);
A3 := mac (E3, -5, F3);   A4 := mac (J3, -5, I3);
A3 := mac (A3, 20, G3);   A4 := mac (A4, 20, H3);
A5 := mac (E5, -5, F5);   A6 := mac (J5, -5, I5);
A5 := mac (A5, 20, G5);   A6 := mac (A6, 20, H5);
A7 := mac (E7, -5, F7);   A8 := mac (J7, -5, I7);
A7 := mac (A7, 20, G7);   A8 := mac (A8, 20, H7);

B11 := add (A1, A2);
B13 := add (A3, A4);
B15 := add (A5, A6);
B17 := add (A7, A8);

B1 := asrrnd (B11, 5);
B3 := asrrnd (B13, 5);
B5 := asrrnd (B15, 5);
B7 := asrrnd (B17, 5);

rd_8*2 (D1);
rd_8*2 (D3);
rd_8*2 (D5);
rd_8*2 (D7);

I1 := add (B1, D1);
I3 := add (B3, D3);
I5 := add (B5, D5);
I7 := add (B7, D7);
```

Слика 3.26 – Псеудокод програма за интерполацију једног 8∗8 блока луминансе на позицији b дефинисаној H.264/MPEG-4 стандардом, коришћењем новог 13∗2 начина приступа меморијском подсистему при креирању референтног блока B_{1-7}.

Друга варијанта реализације, заснована на новом 13∗2 начину приступа, елиминише четири од осам операција читања при креирању референтног блока, као што приказује псеудокод на слици 3.26. Захваљујући томе што нова опе-



Слика 3.27 – Распоред операција при интерполацији једног 8∗8 блока луминансе на позицији b дефинисаној H.264/MPEG-4 стандардом, коришћењем новог 13∗2 начина приступа меморијском подсистему при креирању референтног блока B_{1-7}.

рација читања дели 13∗2 блок на шест 8∗2 блокова, аналогно подели која је приказана на слици 3.22, елиминише се и свих шеснаест „shuffle" операција из слотова два и три, тако да се ови слотови не користе. Остатак псеудокода је непромењен у односу на претходну варијанту реализације коришћењем 8∗2 начина приступа.

Распоред операција којима се овај псеудокод реализује на истој векторској путањи података са укупно седам слотова[34] приказан је на слици 3.27. За креирање референтног блока потребно је једанаест циклуса такта, што је за два циклуса мање, односно осамнаест процената брже у односу на реализацију коришћењем познатог 8∗2 начина приступа. Убрзање обраде искључиво је резултат новог 13∗2 начина приступа и нове операције читања из паралелног меморијског подистема јер су остатак псеудокода и архитектура векторске путање непромењени. За интерполацију једног 8∗8 блока укупно је потребно дванаест циклуса такта, што је такође за два циклуса мање у односу на другу варијанту реализације и представља убрзање обраде од седамнаест процената.

Процентуално гледано, убрзање H.264/MPEG-4 интерполације, које доноси нови 13∗2 у односу на постојећи 8∗2 начин приступа мање је него убрзање које 9∗2 начин приступа доноси у случају упаривања блокова, што је анализирано у

[34] У овом случају узета је путања података са истим бројем слотова као и код реализације коришћењем 8∗2 начина приступа.

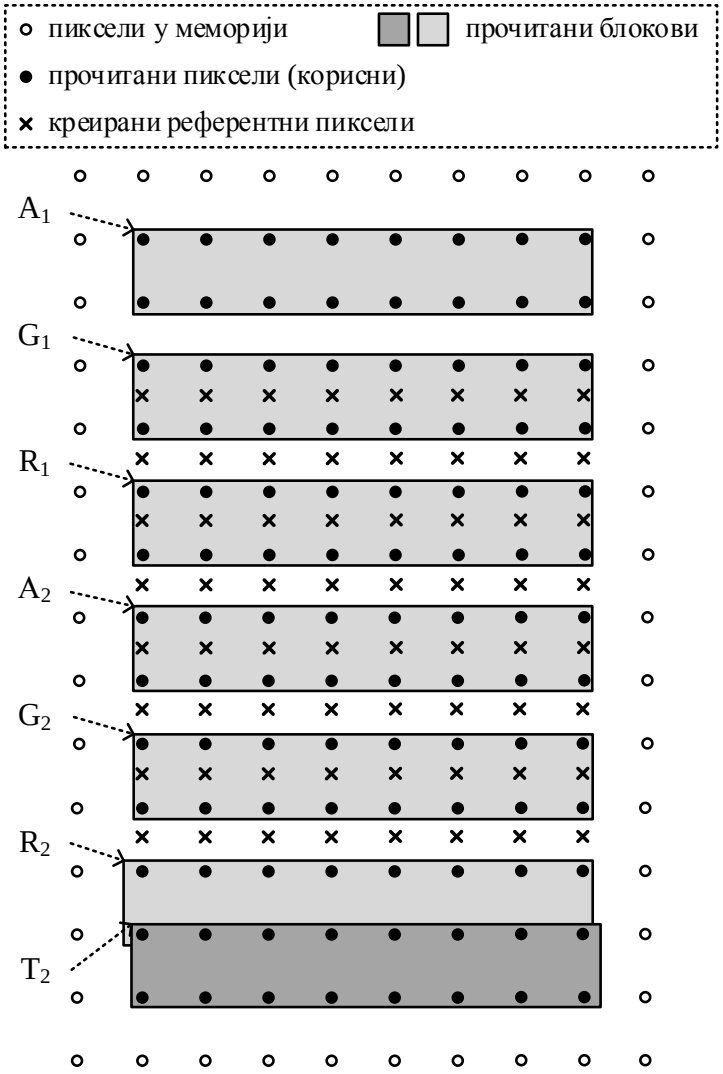
Табела 3.2 – Број циклуса такта и прочитаних пиксела за креирање једног 8∗8 референтног блока и интерполацију једног 8∗8 блока луминансе на позицији *b* дефинисаној H.264/MPEG-4 стандардом.

Варијанта реализације	Број циклуса такта	Број прочитаних пиксела
	Креирање референтног блока	
са познатим 8∗2	13	128
са новим 13∗2	11 (18% мање)	104 (23% мање)
	Интерполација блока луминансе	
са познатим 8∗2	14	192
са новим 13∗2	12 (17% мање)	168 (14% мање)

секцији 3.1.4. Разлог томе је мањи удео операција читања из меморијског подсистема у укупном броју операција у случају H.264/MPEG-4 интерполације. Са слика 3.24 и 3.7а може се закључити да удео операција читања износи дванаест од педесет шест и четрнаест од тридесет осам, односно двадесет један и тридесет седам процената, респективно. Због тога, половљење броја приступа меморијском подсистему потребних за креирање референтног блока у H.264/MPEG-4 интерполацији мање доприноси смањењу укупног број циклуса такта у односу на половљење броја приступа за креирање блока кандидата код реализације упаривања блокова.

Извршавањем четири 13∗2 операције читања при креирању референтног блока, прочита се сто четири пиксела из меморијског подсистема, што је око двадесет три процента мање у односу на реализацију коришћењем 8∗2 начина приступа. Узимајући и блокове разлика у обзир, укупно се прочита сто шездесет осам пиксела, односно четрнаест процената мање у односу на реализацију са 8∗2 начином приступа. Како је потрошња енергије меморијског подсистема пропорционална броју прочитаних и уписаних пиксела, што ће касније бити детаљније објашњено, може се рећи да, поред убрзања обраде, нови 13∗2 начин приступа и нова операција читања значајно смањују потрошњу меморијског подсистема по интерполираном блоку луминансе.

Табела 3.2 резимира закључке изведене до сада у овој секцији, приказујући потребан број циклуса такта и број прочитаних пиксела за креирање једног референтног 8∗8 блока и интерполацију једног 8∗8 блока луминансе, за обе посматране варијанте реализације. У заградама је приказано процентуално умањење вредности посматраних параметара код 13∗2 у односу на 8∗2 реализацију.



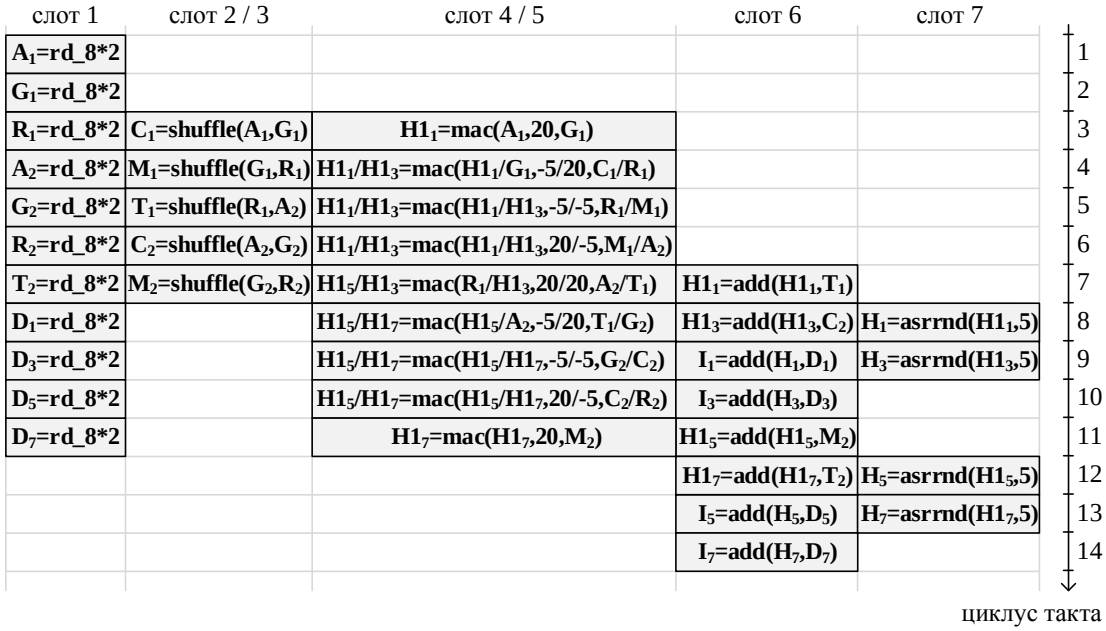
Слика 3.28 – Подела блока од 8∗13 пиксела на седам 8∗2 блокова, који се читају за креирање референтног 8∗8 блока вертикалним филтром шестог реда.

До сада је у овој секцији анализирано креирање 8∗8 референтног блока на позицији b приказаној на слици 3.21, односно реализација хоризонталног филтра шестог реда који захтева приступ блоку од 13∗8 пиксела у меморији. Са друге стране, креирање референтног блока на позицији h врши се вертикалним филтром шестог реда, дефинисаним једначином 3.9. Овакав филтар захтева приступ блоку од 8∗13 пиксела. За илустрацију реализације вертикалног филтра, у овој секцији користе се постојећи 8∗2 и нови 2∗13 начини приступа. Реализација коришћењем 2∗8 начина приступа идентична је 8∗2 реализацији хоризонталног филтра приказаној на слици 3.25, јер је разлика између ових филтара само то што су међусобно заротирани за деведесет степени. Исто важи и за 2∗8 реализацију хоризонталног у односу на 8∗2 реализацију вертикалног филтра.

```
rd_8*2 (A1);
rd_8*2 (G1);
rd_8*2 (R1);
rd_8*2 (A2);
rd_8*2 (G2);
rd_8*2 (R2);
rd_8*2 (T2);

C1 := shuffle (A1, G1);
M1 := shuffle (G1, R1);
T1 := shuffle (R1, A2);
C2 := shuffle (A2, G2);
M2 := shuffle (G2, R2);

H11 := mac (A1,  20, G1);
H11 := mac (H11, -5, C1);
H11 := mac (H11, -5, R1);
H11 := mac (H11, 20, M1);
H11 := add (H11, T1);

H13 := mac (G1,  20, R1);
H13 := mac (H13, -5, M1);
H13 := mac (H13, -5, A2);
H13 := mac (H13, 20, T1);
H13 := add (H13, C2);

H15 := mac (R1,  20, A2);
H15 := mac (H15, -5, T1);
H15 := mac (H15, -5, G2);
H15 := mac (H15, 20, C2);
H15 := add (H15, M2);

H17 := mac (A2,  20, G2);
H17 := mac (H17, -5, C2);
H17 := mac (H17, -5, R2);
H17 := mac (H17, 20, M2);
H17 := add (H17, T2);

H1 := asrrnd (H11, 5);   H3 := asrrnd (H13, 5);
H5 := asrrnd (H15, 5);   H7 := asrrnd (H17, 5);

rd_8*2 (D1);   rd_8*2 (D3);
rd_8*2 (D5);   rd_8*2 (D7);

I1 := add (H1, D1);   I3 := add (H3, D3);
I5 := add (H5, D5);   I7 := add (H7, D7);
```

Слика 3.29 – Псеудокод програма за интерполацију једног 8∗8 блока луминансе на позицији h дефинисаној H.264/MPEG-4 стандардом, коришћењем познатог 8∗2 начина приступа меморијском подсистему при креирању референтног блока H_{1-7}.

Уз коришћење 8∗2 начина приступа потребно је седам операција читања из паралелног меморијског подсистема за дохватање једног 8∗13 блока, према подели коју приказује слика 3.28. На слици 3.29 приказан је псеудокод про-



Слика 3.30 – Распоред операција при интерполацији једног 8∗8 блока луминансе на позицији h дефинисаној H.264/MPEG-4 стандардом, коришћењем познатог 8∗2 начина приступа меморијском подсистему при креирању референтног блока H_{1-7}.

грама који реализује креирање референтног блока вертикалним филтром, а затим и израчунавање вредности интерполираног блока. Поред седам операција читања из паралелног меморијског подсистема, потребно је извршити и пет „shuffle" операција, како би се припремило свих дванаест 8∗2 блокова потребних за креирање једног 8∗8 референтног блока. Креирање референтног блока H_{1-7} врши се низом „mac", „add" и „asrrnd" операција, на сличан начин као код хоризонталног филтра. На крају се прочита 8∗8 блок разлика D_{1-7} и његовим сабирањем са референтим блоком одреди се вредност интерполираног 8∗8 блока I_{1-7}.

Слика 3.30 приказује распоред операција којима се овај псеудокод реализује на истој векторској путањи података са седам слотова која је коришћена за реализацију хоризонталног филтра. Према овој слици, за креирање референтног 8∗8 блока потребно је тринаест циклуса такта. Притом се, извршавањем седам 8∗2 операција читања из меморијског подсистема, прочита сто дванаест пиксела, од чега сто четири различита, а осам пиксела се прочита два пута. Укупно је за интерполацију 8∗8 блока луминансе потребно четрнаест циклуса такта, исто као за интерполацију хоризонталним филтром на позицији b уз коришћење 8∗2 начина приступа. Узимајући у обзир четири 8∗2 операције читања блока разлика, укупно се прочита сто седамдесет шест пиксела.

Табела 3.3 – Број циклуса такта и прочитаних пиксела за креирање једног 8∗8 референтног блока и интерполацију једног 8∗8 блока луминансе на позицији h дефинисаној H.264/MPEG-4 стандардом.

Варијанта реализације	Број циклуса такта	Број прочитаних пиксела
	Креирање референтног блока	
са познатим 8∗2	13	112
са новим 2∗13	11 (18% мање)	104 (8% мање)
	Интерполација блока луминансе	
са познатим 8∗2	14	176
са новим 2∗13	12 (17% мање)	168 (5% мање)

На основу претходне анализе може се закључити да је реализација вертикалног филтра, коришћењем 8*2 начина приступа, нешто ефикаснија по потрошњи енергије од реализације хоризонталног филтра, јер захтева читање шеснаест пиксела мање.

Реализација вертикалног филтра коришћењем 2∗13 начина приступа такође је аналогна реализацији хоризонталног филтра коришћењем 13∗2 начина приступа. Уколико се посматрају псеудокод, распоред операција, број циклуса такта и број прочитаних пиксела, једина разлика између ове две реализације јесте коришћење „rd_2∗13” и „rd_2∗8” уместо „rd_13∗2” и „rd_8∗2” операција, респективно. Узимајући ово у обзир, може се рећи да псеудокод приказан на слици 3.26 и распоред операција на слици 3.27, одговарају реализацији вертикалног филтра коришћењем 2∗13 начина приступа. Што се броја циклуса такта и прочитаних пиксела тиче, вредности из табеле 3.2, које се односе на реализацију са 13∗2 начином приступа, важе и за реализацију вертикалног филтра коришћењем 2∗13 начина приступа. Табела 3.3 приказује те вредности за обе варијанте реализације H.264/MPEG-4 интерполације на позицији h.

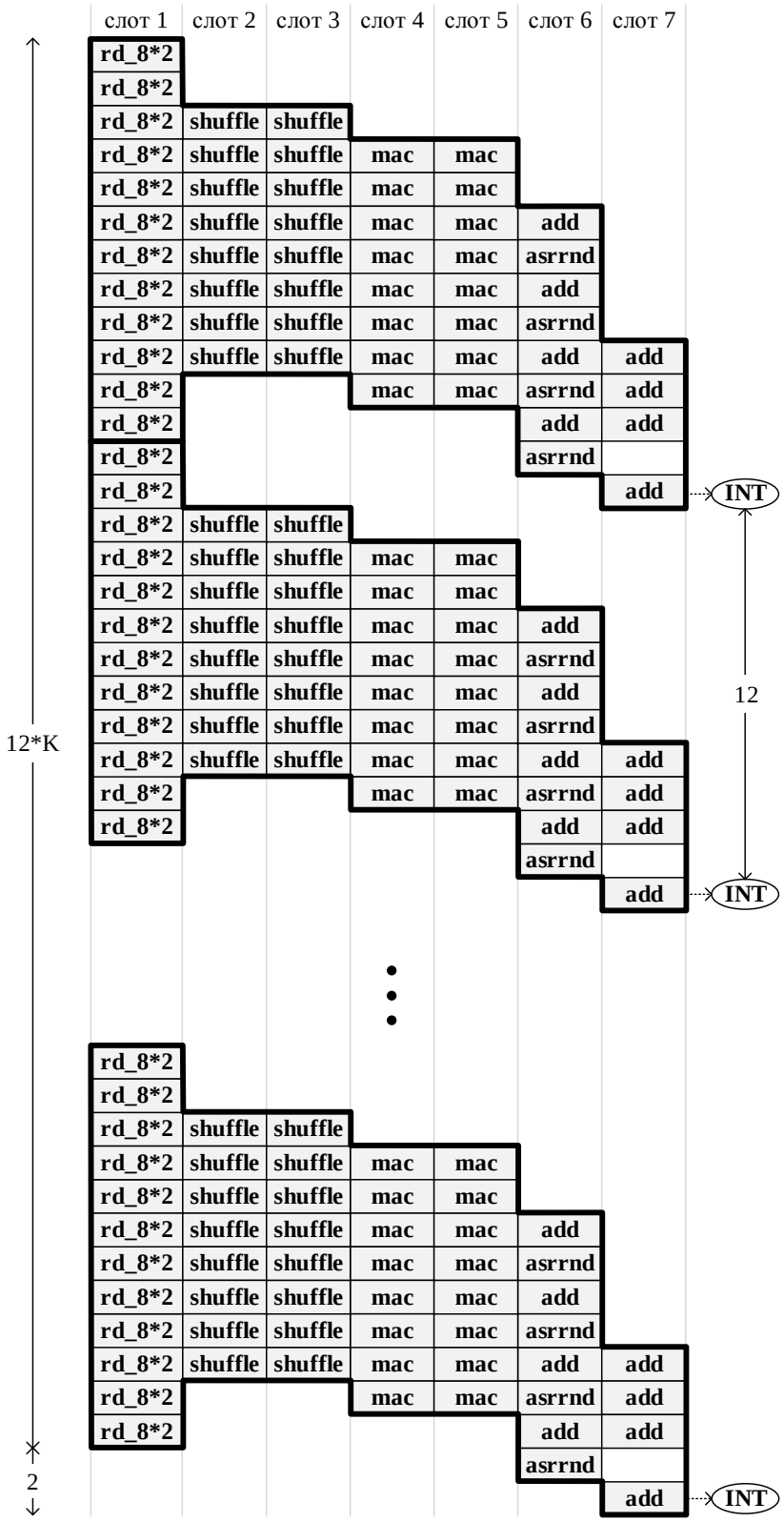
Реализација интерполације на позицији j, дефинисаној на слици 3.21, своди се на комбинацију хоризонталног и вертикалног филтра, тако да се хоризонталним филтром израчунају блокови на позицијама aa, bb, b_1, s_1, gg и hh, а потом се примени вертикални филтар. У овом случају потребно је најпре из меморије прочитати блок од укупно 13∗13 пиксела, потом хоризонталним филтром креирати 8∗13 блок и коначно креирати 8∗8 референтни блок вертикалним филтром. За ефикасну реализацију овакве обраде предлаже се коришћење 13∗2 или 13∗13 начина приступа, у зависности од жељеног компромиса између брзине обраде и цене меморијског подсистема и векторске путање података.

Интерполација на преостале две позиције дефинисане H.264/MPEG-4 стандардом, означене са a и e на слици 3.21, захтева по једну додатну операцију „add" и „asrrnd", односно два додатна циклуса такта у односу на интерполацију на позицијама b и h, респективно. Како нема додатних операција приступа меморијском подсистему, број прочитаних пиксела је исти као код интерполације на позицијама b и h.

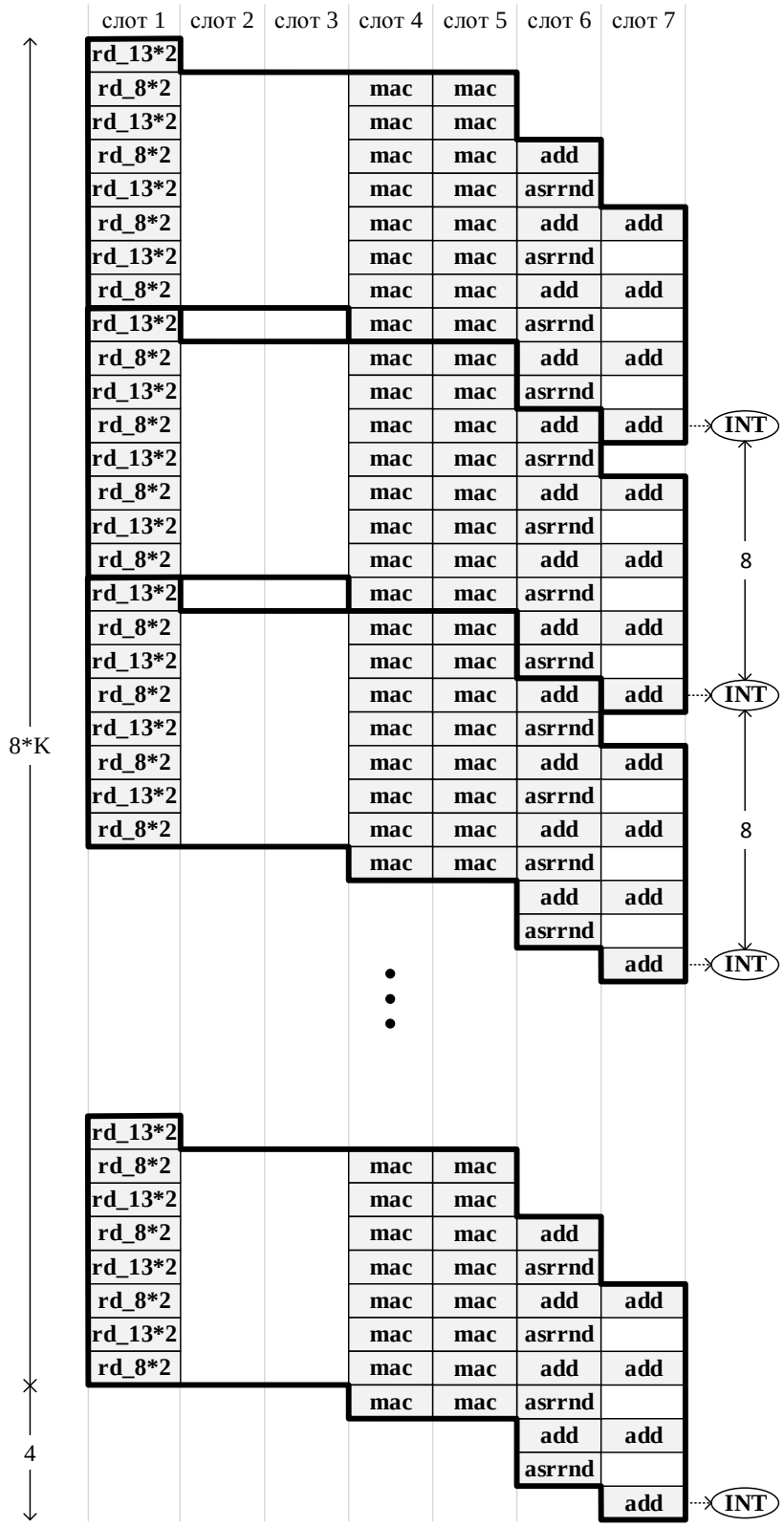
Уколико посматрамо интерполације више блокова луминансе, на пример интерполације четири 8∗8 блока који чине један 16∗16 макроблок, њихово извршавање може бити делимично преклопљено како би се постигло боље искоришћење функционалних јединица и већа пропусност обраде. Распореди операција за интерполацију K 8∗8 блокова на позицијама b, b и h, уз коришћење 8*2, 13*2 и 8*2 начина приступа, приказани су на сликама 3.31, 3.32 и 3.33, респективно. У циљу боље прегледности слика, операције које чине интерполацију једног 8∗8 блока уоквирене су дебљом линијом.

Слике приказују да се операције читања из паралелног меморијског подсистема извршавају у сваком циклусу такта и представљају уско грло у свим илустрованим варијантама. Тиме се потврђује наша полазна претпоставка да се смањењем броја приступа меморијском подсистему може убрзати обрада. Као што смо у претходној секцији навели, овакав закључак је валидан под претпоставком да процесор садржи само један меморијском подсистем и један слот за приступ меморијском подсистему, а да је број слотова са аритметичко-логичким јединицама довољан да они не представљају уско грло, већ да могу реализовати максималну брзину обраде коју омогућава меморијски подсистем. У пракси је ова претпоставка најчешће испуњена, односно могуће је приуштити један меморијски подсистем, један слот за приступ меморијском подсистему и довољан број аритметичко-логичких јединица, захваљујући томе што је цена логичких кола, којима се те јединице реализују, релативно ниска, а цена меморије висока у савременим технологијама израде чипова [12].

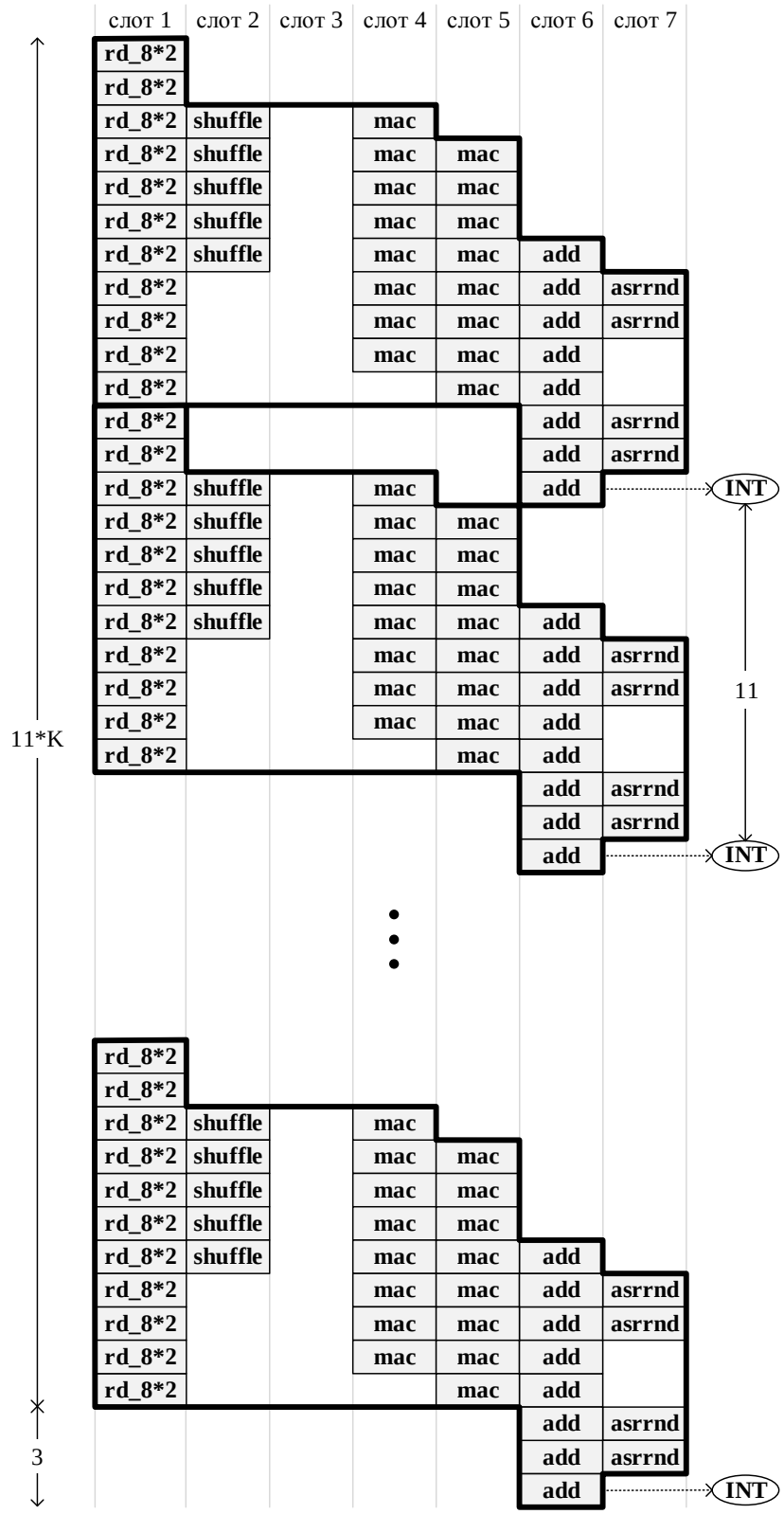
Узимајући претходно у обзир, допринос нових начина приступа је и то што омогућују боље искоришћење доступних аритметичко-логичких јединица. То се јасно може видети упоређивањем искоришћења „mac" јединица на сликама 3.31 и 3.32. Коришћењем 13∗2 начина приступа омогућено је потпуно искоришћење „mac" јединица, док су при коришћењу 8∗2 начина приступа „mac" јединице заузете у осам од једанаест циклуса такта, односно њихово искоришћење је седамдесет три процента.



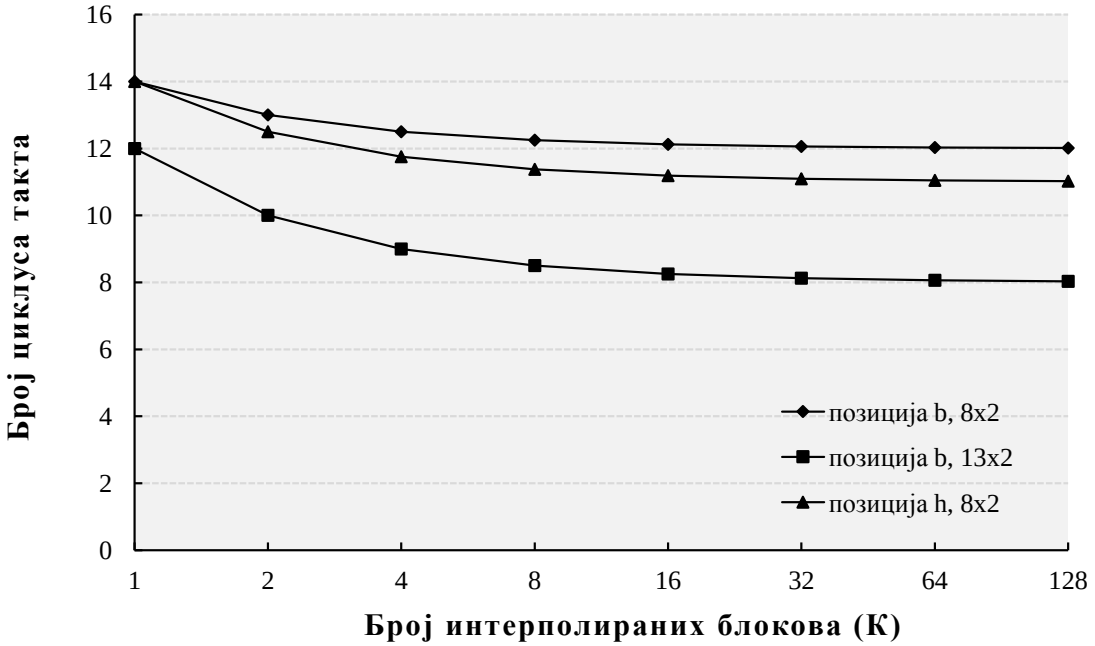
Слика 3.31 – Интерполација K 8∗8 блокова луминансе на позицији b дефинисаној H.264/MPEG-4 стандардом, делимично преклопљено, коришћењем 8∗2 начина приступа за креирање референтног блока.



Слика 3.32 – Интерполација K 8∗8 блокова луминансе на позицији b дефинисаној H.264/MPEG-4 стандардом, делимично преклопљено, коришћењем 13∗2 начина приступа за креирање референтног блока.



Слика 3.33 – Интерполација K 8∗8 блокова луминансе на позицији h дефинисаној H.264/MPEG-4 стандардом, делимично преклопљено, коришћењем 8∗2 начина приступа за креирање референтног блока.



Слика 3.34 – Просечан број циклуса такта за H.264/MPEG-4 интерполацију једног 8∗8 блока.

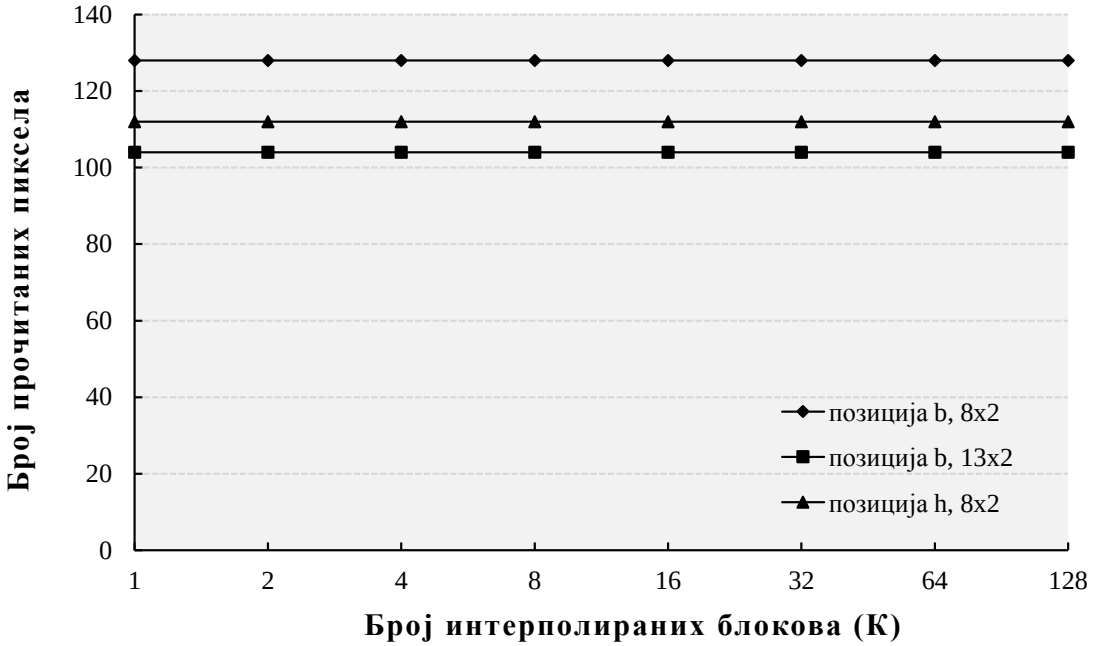
Анализом распореда операција на сликама 3.31, 3.32 и 3.33 долазимо до следећих једначина за укупан број циклуса потребних за интерполацију K 8∗8 блокова, респективно:

$$\begin{aligned} C_t^b\ (8*2) &= 12*K+2, \\ C_t^b(13*2) &= 8*K+4, \\ C_t^h\ (8*2) &= 11*K+3. \end{aligned} \tag{3.13}$$

На основу претходних израза, просечан број циклуса такта потребних за интерполацију једног 8∗8 блока је:

$$\begin{aligned} C_a^b\ (8*2) &= \frac{C_t^b\ (8*2)}{K} = 12+\frac{2}{K}, \\ C_a^b(13*2) &= \frac{C_t^b(13*2)}{K} = 8+\frac{4}{K}, \\ C_a^h\ (8*2) &= \frac{C_t^h\ (8*2)}{K} = 11+\frac{3}{K}. \end{aligned} \tag{3.14}$$

Добијени изрази показују да укупан број приступа меморијском подсистему у највећој мери одређује брзину интерполације. Исто то показују и графикони на слици 3.34, који илуструју једначине 3.14 за низ вредности параметра K, где се види да просечан број циклуса такта брзо конвергира ка броју приступа меморијском подсистему извршених за интерполацију једног 8∗8 блока.



Слика 3.35 – Просечан број прочитаних пиксела за H.264/MPEG-4 интерполацију једног 8∗8 блока.

Други значајан параметар јесте просечан број пиксела прочитаних из меморијског подсистема за интерполацију једног 8∗8 блока. Овај параметар представља добру апроксимацију потрошње енергије подсистема за дату обраду. Број прочитаних пиксела пропорционалан је броју читања из меморијских ћелија, а потрошња енергије меморијских ћелија представља највећи део укупне потрошње подсистема, док потрошња логичких кола који се налазе око ћелија чини много мањи део без обзира на архитектуру подсистема и коришћени начин приступа. На основу претходно извршене анализе, број пиксела прочитаних из подсистема за интерполацију једног 8∗8 блока дефинисан је следећим изразима:

$$\begin{aligned} R_a^b\ (8*2) &= \frac{128*K}{K} = 128, \\ R_a^b(13*2) &= \frac{104*K}{K} = 104, \\ R_a^h\ (8*2) &= \frac{112*K}{K} = 112. \end{aligned} \tag{3.15}$$

Претходни изрази показују да број прочитаних пиксела не зависи од броја интерполираних блокова, јер се за сваки блок чита исти број пиксела. Слика 3.35 графички илуструје једначине 3.15, како би се визуелно приказао добитак коришћењем новог у односу на постојећи начин приступа.

Трећи параметар, који је важно анализирати, јесте просечно искоришћење функционалних јединица векторске путање података. Просечно искоришћење јединица добија се као количник укупног броја извршених операција у одређеном броју циклуса такта и максималног броја операција који се може извршити у истом броју циклуса такта на датој векторској путањи података. За илустроване случајеве интерполације, укупан број извршених операција дат је следећим једначинама:

$$\begin{aligned} O_t^b\ (8*2) &= 56*K, \\ O_t^b(13*2) &= 36*K, \\ O_t^h\ (8*2) &= 46*K. \end{aligned} \tag{3.16}$$

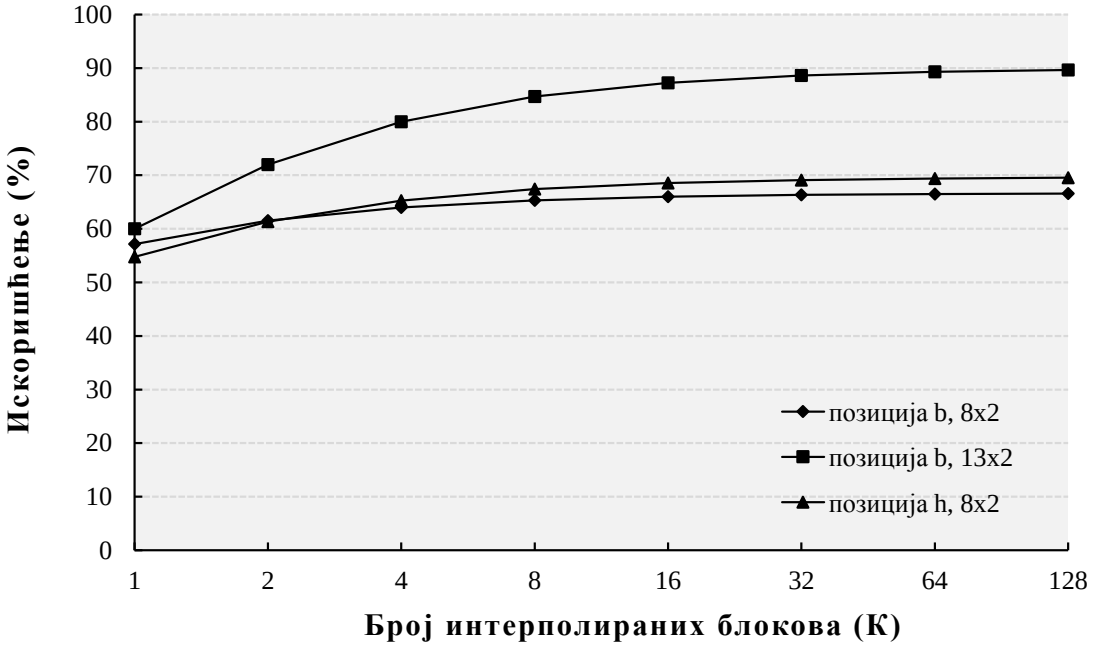
Максималан број операција који се може извршити у истом броју циклуса такта добија се као производ броја коришћених слотова и броја циклуса такта. У случају интерполације на позицији b уз 8∗2 и 13∗2 начине приступа коришћено је седам и пет слотова, респективно, а у случају интерполације на позицији h уз 8∗2 начин приступа коришћено је шест слотова. Стога, максималан број операција за три илустрована случаја износи:

$$\begin{aligned} O_m^b\ (8*2) &= 7*C_t^b\ (8*2) = 7*(12*K+2) = 84*K+14, \\ O_m^b(13*2) &= 5*C_t^b(13*2) = 5*(\ 8*K+4) = 40*K+20, \\ O_m^h\ (8*2) &= 6*C_t^h\ (8*2) = 6*(11*K+3) = 66*K+18. \end{aligned} \tag{3.17}$$

На основу једначина 3.16 и 3.17, добија се просечно искоришћење функционалних јединица векторске путање података:

$$\begin{aligned} U_a^b\ (8*2) &= \frac{O_t^b(8*2)}{O_m^b(8*2)} = \frac{56*K}{84*K+14} \quad \left(\underset{K=1}{=}57\% \ \wedge\ \underset{K\to\infty}{=}67\%\right), \\ U_a^b(13*2) &= \frac{O_t^b(13*2)}{O_m^b(13*2)} = \frac{36*K}{40*K+20} \quad \left(\underset{K=1}{=}60\% \ \wedge\ \underset{K\to\infty}{=}90\%\right), \\ U_a^h\ (8*2) &= \frac{O_t^h(8*2)}{O_m^h(8*2)} = \frac{46*K}{66*K+18} \quad \left(\underset{K=1}{=}55\% \ \wedge\ \underset{K\to\infty}{=}70\%\right). \end{aligned} \tag{3.18}$$

Слика 3.36 илуструје просечно искоришћење функционалних јединица векторске путање података у зависности од параметра K, изражено у процентима. Са слике се јасно види да нови начин приступа омогућује веће искоришћење јединица, посебно у случају већих вредности параметра K, у односу на постојеће начине приступа.

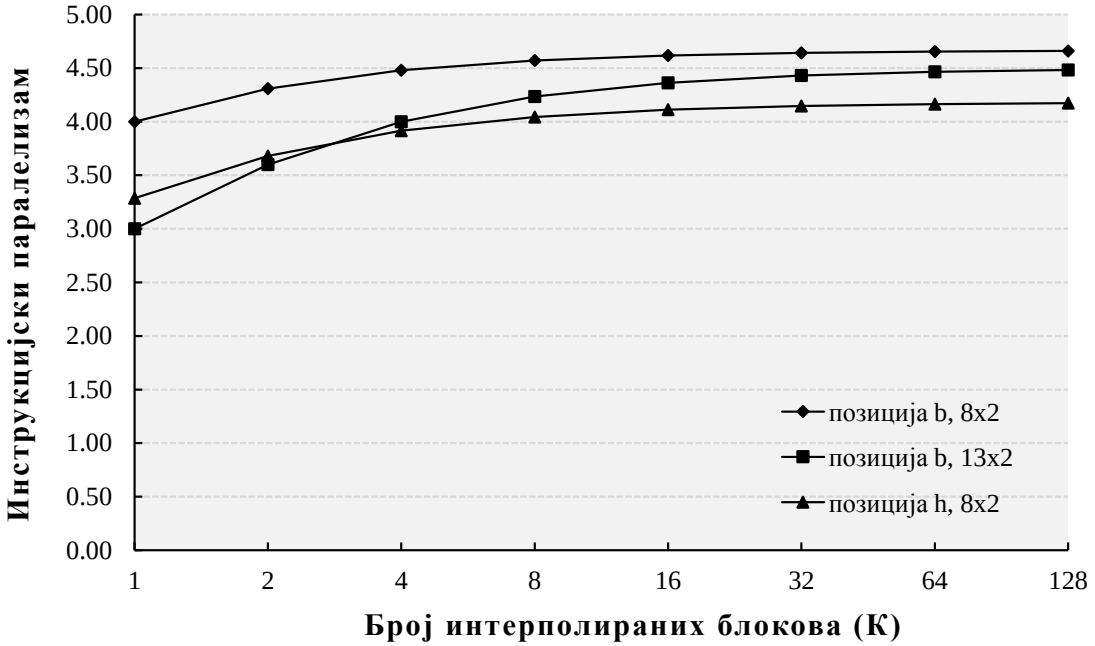


Слика 3.36 – Просечно искоришћење функционалних јединица векторске путање података за H.264/MPEG-4 интерполацију једног 8∗8 блока.

На сличан начин може се одредити и просечан инструкцијски паралелизам, односно просечан број операција које се извршавају у паралели, као однос укупног броја извршених операција O_t и броја циклуса такта потребних за извршавање тих операција C_t:

$$
\begin{aligned}
I_a^b\ (8*2) &= \frac{O_t^b(8*2)}{C_t^b(8*2)} = \frac{56*K}{12*K+2} \quad \left(\underset{K=1}{=4,0} \ \wedge \ \underset{K\to\infty}{=4,7}\right), \\
I_a^b(13*2) &= \frac{O_t^b(13*2)}{C_t^b(13*2)} = \frac{36*K}{8*K+4} \quad \left(\underset{K=1}{=3,0} \ \wedge \ \underset{K\to\infty}{=4,5}\right), \\
I_a^h\ (8*2) &= \frac{O_t^h(8*2)}{C_t^h(8*2)} = \frac{46*K}{11*K+3} \quad \left(\underset{K=1}{=3,3} \ \wedge \ \underset{K\to\infty}{=4,2}\right).
\end{aligned}
\tag{3.19}
$$

Слика 3.37 илуструје изразе дате једначином 3.19 за више вредности параметра K. Слике 3.36 и 3.37 потврђују раније донет закључак да нови 13∗2 начин приступа и нова операција читања из меморијског подсистема омогућују веће искоришћење функционалних јединица, иако инструкцијски паралелизам није увек већи у односу на реализацију коришћењем 8∗2 начина приступа. Ово је последица боље прилагођености нових начина приступа и операција читања из меморијског подсистема карактеристикама методе обраде, чиме је укупан број операција на критичној путањи смањен, а обрада убрзана.



Слика 3.37 – Просечан инструкцијски паралелизам при H.264/MPEG-4 интерполацији једног 8∗8 блока.

3.3 Филтрирање у просторном домену

Како би омогућио брзо и енергетски ефикасно филтрирање слике и видеа у просторном домену, на векторској путањи података чији паралелизам износи N пиксела, паралелни меморијски подсистем треба да подржи следеће начине приступа:

1. **Непоравнати приступ реду са више од N пиксела.**
2. Паралелни приступ табели пресликавања[35].
3. Поравнати приступ реду од N и више пиксела.

Прва врста начина приступа представља једну од иновативних идеја овог рада, док су друга и трећа већ познате и подржане од стране постојећих архитектура паралелног меморијског подсистема. Детаљнија анализа, на основу које су изведени претходни закључци, дата је у наредним секцијама.

3.3.1 Преглед примена

Филтрирање у просторном домену представља технику обраде слике и видеа којом се мењају вредности постојећих пиксела на основу њихових почетних

[35]енг. Look-up table (LUT)

вредности, а најчешће и на основу почетних вредности пиксела из њихове непосредне околине. Начин на који се врши измена вредности пиксела дефинисан је преносном функцијом филтра.

Филтрирање у просторном домену има многобројне примене и представља важан део система за обраду слике и видеа. Две основне примене јесу побољшање визуелног квалитета[36] и рестаурација[37], које подразумевају обраду у циљу отклањања визуелних недостатака насталих услед грешака приликом креирања и преноса слике и видеа. У област побољшања визуелног квалита спадају потискивање шума[38], изоштравање[39] и умекшавање слике[40], док редукција замућености слике[41] припада области рестаурације [1–3]. Поред ових примена, филтрирање у просторном домену користи се и као основна техника у многим другим, сложенијим и софистицираним применама обраде слике и видеа [2, 3]. Примери таквих примена су издвајање ивица и других атрибута из слике, као и препознавање облика и објеката на слици.

Шум у слици и видеу јесте случајна величина и настаје услед несавршености сензора камере, квантизације током аналогно-дигиталне конверзије у процесу креирања слике и грешака при њеном преносу кроз комуникациони канал. Филтрирање у просторном домену ефикасно је за потискивање такозваног „со и бибер" шума, који представља случајну појаву белих и црних пиксела у слици, затим импулсног шума, односно случајног појављивања белих пиксела, као и адитивног Гаусовог шума, који се назива тако јер је његова вредност додата на вредност пиксела, а вероватноћа вредности шума одговара Гаусовој расподели.

Изоштравање слике представља обраду која за циљ има истицање ивица и финих детаља у слици [3], односно појачавања контраста између суседних пиксела који се значајно разликују. Оваква модификација слике прија људском оку јер је оно осетљиво на ивице и детаље у слици, а може се постићи једноставним филтрирањем у просторном домену.

Обрада супротна изоштравању јесте умекшавање слике. Умекшавање се користи када је потребно ублажити или елиминисати ситне детаље на слици. На

[36] енг. image and video enhancement

[37] енг. image and video restoration

[38] енг. denoising

[39] енг. sharpening

[40] енг. smoothing

[41] енг. deblurring

пример, врши се као први корак у препознавању и издвајању великих објеката из слике, јер тада ситни детаљи представљају сметњу [2].

Замућење настаје као последица релативног кретања између камере и сцене, као и услед нефокусираности оптичких елемената камере приликом креирања слике и видеа [3]. Филтрирањем у просторном домену може се поништити замућење, односно може се рестаурирати слика која боље одговара оригиналној сцени, уколико је познат математички модел замућења.

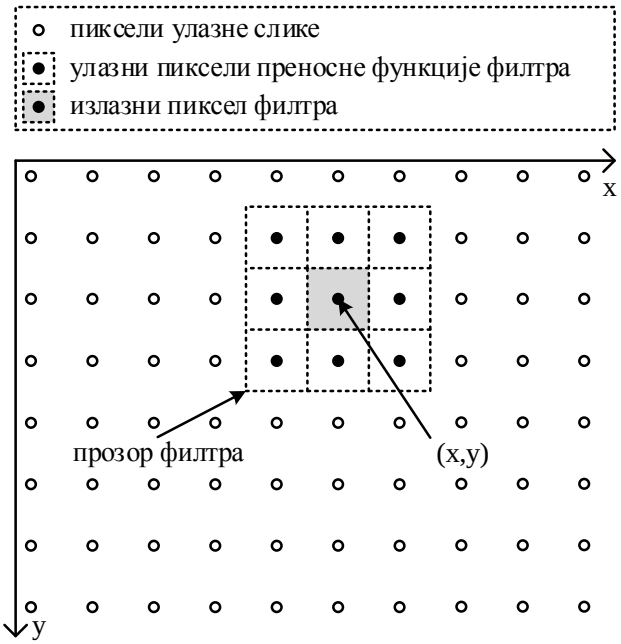
3.3.2 Преглед метода

У обради слике филтрирањем постоје два основна приступа, обрада у просторном и обрада у трансформационом домену, као што је фреквенцијски. Специфичност филтрирања у просторном, у односу на филтрирање у трансформационом домену, јесте то што се обрада врши директно на пикселима слике. У наставку ове секције анализирају се основне методе филтрирања у просторном домену.

У општем случају, филтрирање у просторном домену дефинише се функцијом преноса T, која, на основу улазне слике f и позиције у излазној слици (x, y), одређује вредност пиксела излазне слике $g(x, y)$, односно одређује одзив филтра на датој позицији [2]:

$$g(x, y) \;=\; T(f, x, y),\; x \in [0, X-1] \;\wedge\; y \in [0, Y-1]. \tag{3.20}$$

У претходној једначини X и Y представљају хоризонталну и вертикалну резолуцију слике, респективно. Притом, за одређивање вредности излазног пиксела $g(x, y)$, поред улазног пиксела $f(x, y)$, преносна функција T користи и пикселе из његове непосредне околине. Ради једноставније реализације просторних филтара, у пракси се користи околина квадратног или правоугаоног облика, непарних димензија и са центром у (x, y), као што је приказано на слици 3.38. Оваква околина назива се и прозором филтра. Стандардне димензије прозора у практичним применама су 3∗3, 5∗5, 11∗11 и веће. Укупан број улазних пиксела који се користе за одређивање једног излазног пиксела представља ред филтра и у овом раду је означен са K. Уз овако дефинисане појмове, филтрирање једне слике у просторном домену може се описати и као померање центра филтарског прозора од тачке до тачке (x, y) у улазној слици и израчунавање излазног пиксела $g(x, y)$ на свакој од тих тачака [2].



Слика 3.38 – Филтар у просторном домену са прозором димензија 3∗3.

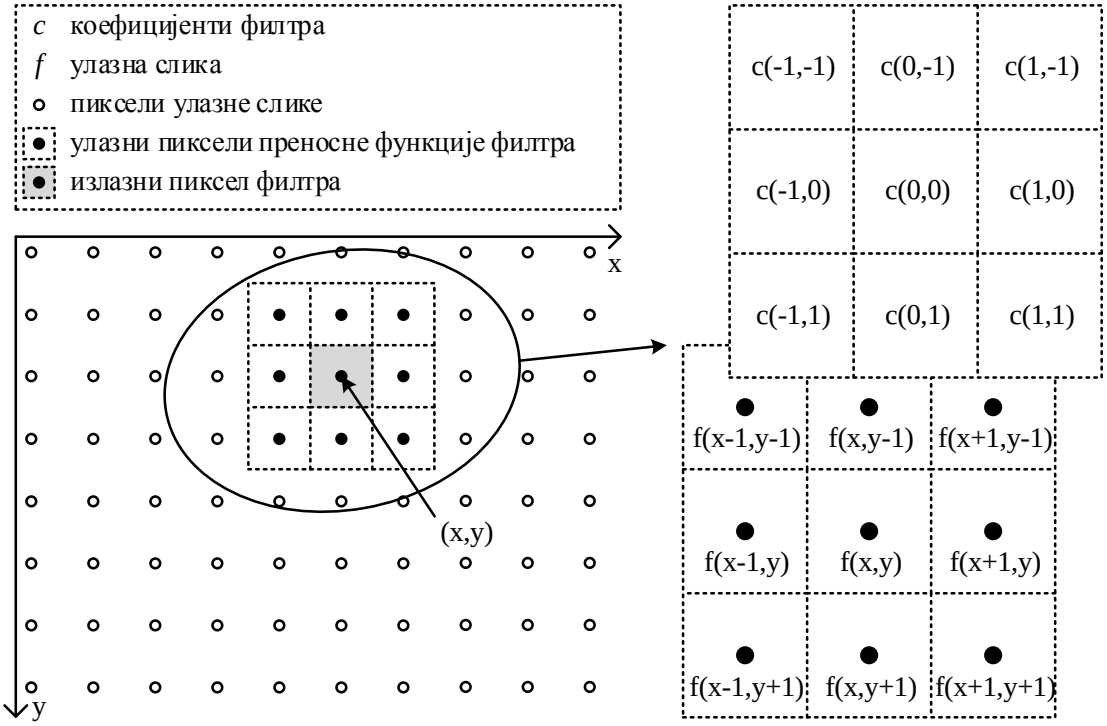
У зависности од преносне функције T, филтар може бити линеаран или нелинеаран. Код линеарног филтра, преносна функција представља линеарну комбинацију, односно суму производа пиксела из филтарског прозора са одговарајућим коефицијентима филтра:

$$g(x,y) = \sum_{m=-(K_x-1)/2}^{(K_x-1)/2} \sum_{n=-(K_y-1)/2}^{(K_y-1)/2} c(m,n) * f(x+m, y+n), \qquad (3.21)$$

где су K_x и K_y хоризонтална и вертикална димензија филтарског прозора, респективно, а $c(m,n)$ представља коефицијент на позицији (m,n) у филтарском прозору, који се у случају линеарних филтара назива и матрицом коефицијената. Слика 3.39 илуструје линеарни филтар димензија 3∗3 дефинисан матрицом коефицијената c.

Може се приметити да једначина 3.21 не дефинише одзив филтра на ивицама улазне слике, где $x+m$ односно $y+n$ има вредност мању од нуле или већу од $X-1$ односно $Y-1$. У пракси се овај проблем решава проширивањем улазне слике потребним бројем пиксела са сваке стране. Стога се специфичност филтрирања ивичних пиксела занемарује у даљој анализи, а фокус се задржава на најчешћем и стога најважнијем случају обраде у средини слике.

Линеарни филтри имају многобројне примене, посебно у области побољшања визуелног квалитета слике, а ефекат који се жели постићи на излазној



Слика 3.39 – Линеарни 3∗3 филтар, дефинисан матрицом коефицијената *c*.

0	1	0
1	-4	1
0	1	0

0	-1	0
-1	4	-1
0	-1	0

1	1	1
1	-8	1
1	1	1

-1	-1	-1
-1	8	-1
-1	-1	-1

Слика 3.40 – Лапласови линеарни филтри за изоштравање слике (извор: [2]).

слици одређују величина матрице и вредности коефицијената. Стога се приликом описа линеарног филтра често даје само матрица коефицијената, која га једнозначно одређује. Примери четири матрице линеарног Лапласовог 3∗3 филтра, којим се врши изоштравање слике, приказане су на слици 3.40.

Линеарни филтри који врше тежинско усредњавање пиксела из филтарског прозора користе се за умекшавање слике [2]. Пример једног таквог 3∗3 филтра приказан је на слици 3.41. Иако су на сликама 3.40 и 3.41 илустроване само 3∗3 матрице, за изоштравање и умекшавање слике користе се и филтри већих димензија, у зависности од јачине ефекта изоштравања и умекшавања који се жели постићи. Такође, може се приметити да је у свим илустрованим примерима филтара за изоштравање односно умекшавање сума коефицијената нула

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

Слика 3.41 – Линеарни филтри за умекшавање слике (извор: [2]).

односно један. На тај начин се постиже да осветљај филтриране слике буде једнак осветљају улазне слике.

Линеарни филтри који врше тежинско усредњавање могу се користити и за потискивање шума. Међутим, недостатак оваквог потискивања шума јесте у нежељним ефектима замућења, умекшавања ивица и губитка ситних детаља у слици [2, 3]. Стога се за потискивање шума користе нелинеарни филтри у просторном домену којима се чувају оштрина, ивице и детаљи [3]. Нелинерни филтри су ефикасни и за многе друге примене, попут изоштравања слике у присуству шума и детекције ивица.

Као што им само име каже, нелинеарне филтре одликује нелинеарна преносна функција, а једна од најчешће коришћених јесте медијан [3]. Медијан функција најпре сортира низ улазних пиксела обухваћених филтарским прозором, а потом као резултат враћа пиксел који се налази на средини сортираног низа. На овај начин одбацују се пиксели који се много разликују од пиксела из своје околине и ефикасно се елиминишу „со и бибер" и импулсни шум [2, 3].

Како би се повећала робусност потискивања шума, користи се тежинско медијан филтрирање као софистицираниja варијанта основног медијана. Код тежинског медијан филтрирања, за сваки пиксел из прозора филтра дефинише се тежински фактор који представља број понављања тог пиксела у сортираном низу медијан функције. На тај начин се подешавањем тежинских фактора подешава утицај одређених пиксела у медијан функцији и може се са већом поузданошћу постићи жељени ефекат филтрирања. Тежински медијан филтри дефинисани су следећим изразом:

$$g(x,y) = \text{MEDIAN}\left(\bigcup_{m=-(K_x-1)/2}^{(K_x-1)/2} \bigcup_{n=-(K_y-1)/2}^{(K_y-1)/2} w(m,n) \diamond f(x+m, y+n) \right), \quad (3.22)$$

где $w(m,n)$ представља матрицу тежинских фактора, а $\diamond$ је оператор пона-

1	1	1	1	1
1	1	1	1	1
1	1	15	1	1
1	1	1	1	1
1	1	1	1	1

Слика 3.42 – Тежински медијан филтар за потискивање шума (извор: [3]).

вљања, који свој десни операнд понавља онолико пута колико износи вредност његовог левог операнда. Аналогно линеарним филтрима, нелинеарни медијан филтри могу се једнозначно описати матрицом тежинских фактора w. Као пример, слика 3.42 илуструје матрицу тежинских фактора медијан филтра који се користи за потискивање шума.

Једна од веома често коришћених техника у обради слике и видеа, а која се може сматрати специјалном врстом филтрирања у просторном домену, јесте пресликавање улазних у излазне пикселе помоћу претходно дефинисане табеле пресликавања[42]. За сваку вредност улазног пиксела $f(x, y)$, табела пресликавања садржи одговарајућу излазну вредност $g(x, y)$, а пресликавање се реализује тако што се улазни пиксел користи директно као адреса или за израчунавање адресе којој се приступа у табели пресликавања. Са једне стране, табеле пресликавања се примењују у методама филтрирања које користе прозор димензија $1*1$, као што су промена осветљаја, контраста и поравнање хистограма слике. Поред тога, овом техником се комплексна и спора аритметичко-логичка израчунавања, попут дељења, кореновања и степеновања, врло често замењују једноставним и брзим приступом табели смештеној у меморији. На тај начин се обрада вишеструко убрзава на рачун нешто већег заузећа меморије.

3.3.3 Потребни начини приступа

Ова секција најпре анализира обрасце приступа пикселима при реализацији филтара у просторном домену на векторској путањи података чији паралелизам

[42] енг. lookup table

износи N пиксела. Потом се, на основу извршене анализе, одређују потребни начини приступа паралелном меморијском подсистему.

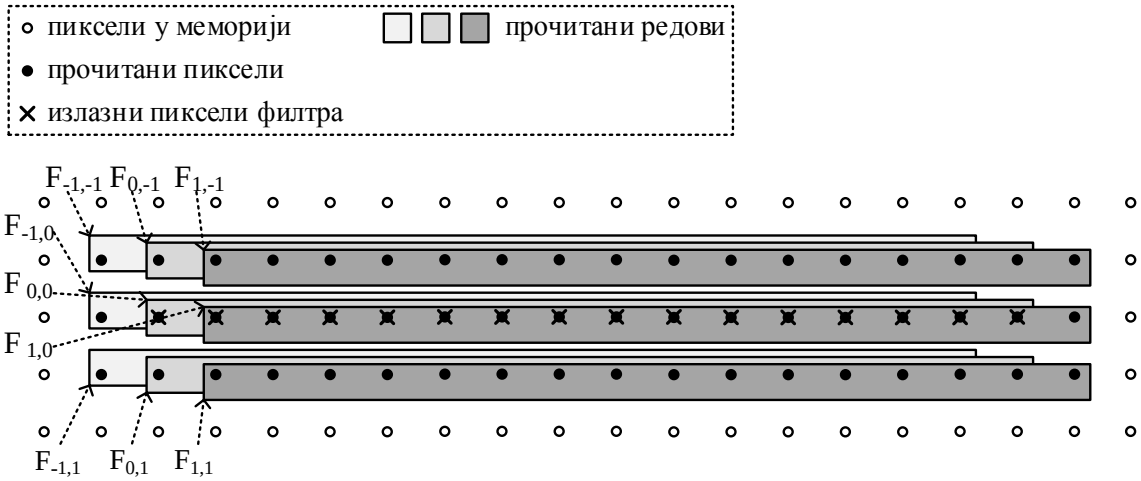
1. **Образац приступа**: Читање улазних пиксела из филтарског прозора потребних за креирање излазних пиксела. Заједничка особина филтара у просторном домену, како линеарних тако и нелинеарних, јесте да је за креирање једног излазног пиксела потребно више суседних улазних пиксела обухваћених филтарским прозором. Како је за креирање једног излазног пиксела потребно прочитати $K_x * K_y$ улазних пиксела из прозора, тако је за креирање N суседних излазних пиксела, на векторској путањи података, потребно прочитати $(K_x + N - 1) * K_y$ суседних улазних пиксела.

   Нови начини приступа: Непоравнати приступ реду са више од N пиксела, односно реду од $K_x + N - 1$ пиксела, који су потребни за креирање излазног реда од N пиксела. Примери оваквих начина приступа у случају 3∗3 филтара су 18∗1 за $N = 16$, 34∗1 за $N = 32$ и 66∗1 за $N = 64$. У случају 5∗5 филтара, то су 20∗1 за $N = 16$, 36∗1 за $N = 32$ и 68∗1 за $N = 64$. За филтре димензија 11∗11, потребни начини приступа су 26∗1 за $N = 16$, 42∗1 за $N = 32$ и 74∗1 за $N = 64$. **Овакви начини приступа паралелном меморијском подсистему представљају један од иновативних доприноса овог рада**.

   Нове операције читања и уписа: Када прочитани ред са више од N пиксела стигне на векторску путању података, односно у јединицу за читање и упис у паралелни меморијски подсистем, он се подели на више редова од N пиксела који се користе за даљу обраду у аритметичко-логичким јединицама чији паралелизам износи N пиксела. Прецизније, један ред од $K_x + N - 1$ пиксела подели се на K_x редова од N пиксела, међусобно померених хоризонтално за по један пиксел, на начин који је илустрован на слици 3.43, на примеру 3∗3 филтра и за $N = 16$.

   Захваљујући описаној подели на редове од N пиксела, нови начини приступа и операције читања не захтевају да векторска путања података буде шира од N пиксела, а самим тим ни скупља у смислу заузећа површине на чипу. Модификација путање података може бити потребна само у смислу додавања нових аритметичко-логичких јединица, како би се искористио већи инструкцијски паралелизам, који је резултат нових операција читања из меморијског подсистема.

   Овакве операције читања из паралелног меморијског подсистема



Слика 3.43 – Подела три реда од 18∗1 пиксела на девет 16∗1 редова, који су потребни за креирање једног 16∗1 реда као излаза 3∗3 просторног филтра.

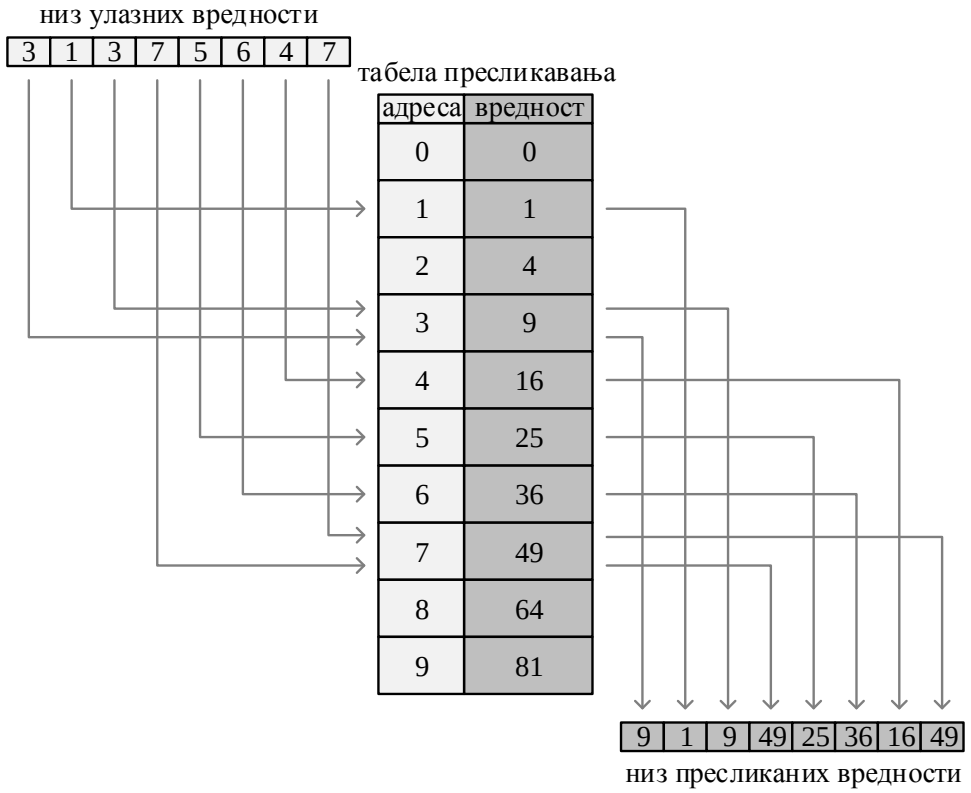
такође представљају један од иновативних доприноса овог рада.

2. **Образац приступа**: Читање група од N пиксела, односно N вредности, са међусобно независних локација у табели пресликавања смештеној у меморији. У обради заснованој на табели пресликавања, улазни пиксел пресликава се у излазни тако што се вредност излазног пиксела прочита из табеле пресликавања, са адресе која одговара вредности улазног пиксела. Како би се извршило пресликавање низа од N пиксела, при обради на векторској путањи података, потребно је прочитати N вредности из табеле пресликавања са N међусобно независних адреса, које одговарају улазном низу од N пиксела.

   Начини приступа: Паралелни приступ табели пресликавања[43], односно истовремено читање L вредности из табеле, извршавањем једног приступа којим се специфицира L међусобно независних адреса, где је $1 \leq L \leq N$. Слика 3.44 илуструје паралелни приступ табели пресликавања од десет локација за случај $L = 8$.

   Употребом овог начина приступа меморијском подсистему, потребно је извршити $N : L$ приступа табели за пресликавање N улазних пиксела. На тај начин пресликавање се убрзава L пута у односу на праволинијску реализацију засновану на N секвенцијалних приступа којима се прочита по једна вредност из табеле пресликавања.

[43] енг. parallel lookup access



Слика 3.44 – Пресликавање $L = 8$ улазних у излазне вредности техником табеле пресликавања са десет локација.

Параметризовањем броја вредности које се читају из табеле једним приступом омогућава се компромис између брзине пресликавања и цене реализације овог начина приступа у смислу заузећа површине. Наиме, брзина пресликавања одређена је количником N и L, а цена је директно пропорционална вредности параметра L, јер је потребно L независних меморијских модула да би се омогућило истовремено читање L вредности са произвољних адреса из табеле, одређених у време извршавања програма. Концепт паралелног приступа табели пресликавања предложен је у патенту [133], где се користи N меморијских модула за читање N вредности из табеле, без могућности компромиса између брзине пресликавања и цене реализације. **Стога, иновативни допринос нашег рада, у овом сегменту, јесте параметризовање паралелног приступа табели пресликавања и омогућавање компромиса између брзине и цене, као и иновативна архитектура за реализацију овог начина приступа, која је описана у поглављу 5**.

3. **Образац приступа**: Преноси једнодимензионалних и дводимензионалних низова пиксела, односно делова слика и видео фрејмова, из меморије ван чипа у паралелни меморијски подсистем на чипу.

   Начини приступа: Поравнати приступ реду од N или више пиксела, подударан са начином приступа меморији ван чипа. Овакав начин приступа је познат и подржан од стране постојећих паралелених меморијских подсистема.

   Детаљни описи овог обрасца и начина приступа дати су у секцији 3.1.3.

3.3.4 Реализација критичне функције

Ова секција илуструје и упоређује две варијанте реализације линеарног 3∗3 филтра у просторном домену, који је дефинисан једначином 3.21. Варијанте реализације разликују се у томе што користе различите начине приступа паралелном меморијском подсистему:

1. постојећи 16∗1 и
2. нови 18∗1.

Циљ поређења јесте да се на практичном примеру илуструју предности нових у односу на постојеће начине приступа паралелном меморијском подсистему, за примену у области филтрирања у просторном домену.

За израчунавање излаза линеарног 3∗3 филтра на векторској путањи података чији паралелизам износи $N = 16$, потребно је прочитати улазни блок од 18∗3 пиксела. У првој варијанти, која користи познати 16∗1 начин приступа, праволинијска реализација захтева извршавање девет операција читања како би се прочитало девет 16∗1 редова означених са $F_{i,j}, i, j \in \{-1, 0, 1\}$, на слици 3.43. Потом се израчунава излаз филтра стандардним операцијама множења „mul” и множења са сабирањем „mac”, која свој први операнд сабира са производом другог и трећег. Псеудокод програма који реализује прву варијанту 3∗3 филтра приказан је на слици 3.45а.

Распоред операција приказаног псеудокода при извршавању на векторској путањи података, која садржи два слота са функционалним јединицама[44], приказан је на слици 3.46. Са слике се може видети да је за израчунавање излаза филтра потребно десет циклуса такта. Притом се изврши девет 16∗1 опера-

[44] Као и у претходним секцијама, број слотова одређен је тако да их има таман довољно да аритметичко-логичке операције не представљају уско грло.

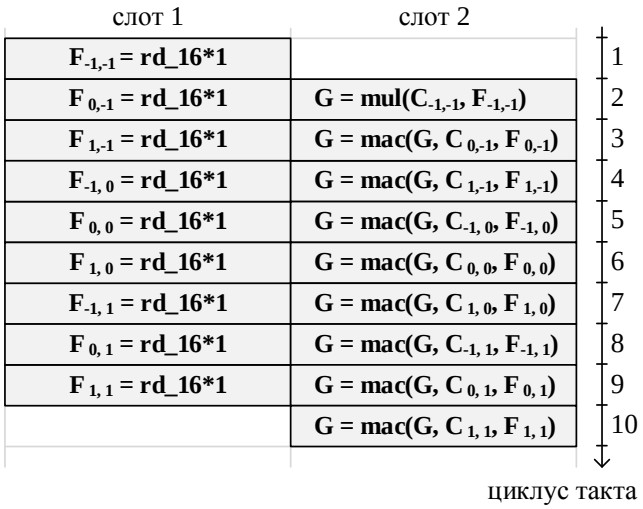
```
rd_16∗1 (F−1,−1);
rd_16∗1 (F 0,−1);
rd_16∗1 (F 1,−1);
rd_16∗1 (F−1, 0);
rd_16∗1 (F 0, 0);
rd_16∗1 (F 1, 0);
rd_16∗1 (F−1, 1);
rd_16∗1 (F 0, 1);
rd_16∗1 (F 1, 1);

G := mul (C−1,−1, F−1,−1);
G := mac (G, C 0,−1, F 0,−1);
G := mac (G, C 1,−1, F 1,−1);
G := mac (G, C−1, 0, F−1, 0);
G := mac (G, C 0, 0, F 0, 0);
G := mac (G, C 1, 0, F 1, 0);
G := mac (G, C−1, 1, F−1, 1);
G := mac (G, C 0, 1, F 0, 1);
G := mac (G, C 1, 1, F 1, 1);
```

(а)

```
rd_18∗1 (F−1,−1, F 0,−1, F 1,−1);


rd_18∗1 (F−1, 0, F 0, 0, F 1, 0);


rd_18∗1 (F−1, 1, F 0, 1, F 1, 1);



G1 := mul (C−1,−1, F−1,−1);
G2 := mul (C 0,−1, F 0,−1);
G3 := mul (C 1,−1, F 1,−1);
G1 := mac (G1, C−1, 0, F−1, 0);
G2 := mac (G2, C 0, 0, F 0, 0);
G3 := mac (G3, C 1, 0, F 1, 0);
G1 := mac (G1, C−1, 1, F−1, 1);
G2 := mac (G2, C 0, 1, F 0, 1);
G3 := mac (G3, C 1, 1, F 1, 1);

G := add (G1,G2);
G := add (G ,G3);
```

(б)

Слика 3.45 – Псеудокод програма линеарног 3∗3 филтра у просторном домену, уз коришћење (а) 16∗1 и (б) 18∗1 начина приступа паралелном меморијском подсистему.



Слика 3.46 – Распоред операција при креирању једног 16∗1 реда линеарним 3∗3 филтром, коришћењем постојећег 16∗1 начина приступа меморијском подсистему.

ција читања и прочита сто четрдесет четири пиксела, од чега педесет четири различита.

Претходно илустрована варијанта реализације, коришћењем 16∗1 начина приступа, може се оптимизовати применом принципа софтверске проточности у циљу смањења броја циклуса такта за израчунавање излаза филтра и броја

Слика 3.47 – Оптимизовано читање операнада једне линије 3∗3 филтра, читањем два реда од 16∗1 пиксела и креирањем два реда од 16∗1 пиксела „shuffle" операцијом.

операција читања из меморијског подсистема. Оптимизација подразумева другачији образац читања пиксела, употребу „shuffle" операција и поновну употребу прочитаних 16∗1 редова у наредној итерацији. У првој итерацији сваке линије пиксела, уместо да се извршавањем три приступа директно прочитају 16∗1 редови $F_{-1,-1}$, $F_{0,-1}$ и $F_{1,-1}$, прочитају се два реда $F_{-1,-1}$ и $F^{+1}_{-1,-1}$, где је $F^{+1}_{-1,-1}$ први наредни 16*1 ред у истој линији слике који се не преклапа са $F_{-1,-1}$, као што илуструје слика 3.47. Након ове две операције читања изврше се две „shuffle" операције које креирају излазне редове $F_{0,-1}$ и $F_{1,-1}$ тако да њихов почетак чини последњих $N-1$ односно $N-2$ пиксела реда $F_{-1,-1}$, а крај први односно прва два пиксела реда $F^{+1}_{-1,-1}$, респективно. Аналогно томе, потребно је извршити још четири операције читања и четири „shuffle" операције, којима се прочитају $F_{-1,0}$ и $F^{+1}_{-1,0}$, односно $F_{-1,1}$ и $F^{+1}_{-1,1}$ редови, а на основу њих потом креирају $F_{0,0}$ и $F_{1,0}$, односно $F_{0,1}$ и $F_{1,1}$. Укупно је потребно извршити шест операција читања и шест „shuffle" операција за израчунавање првог 16∗1 излаза филтра у једној линији слике.

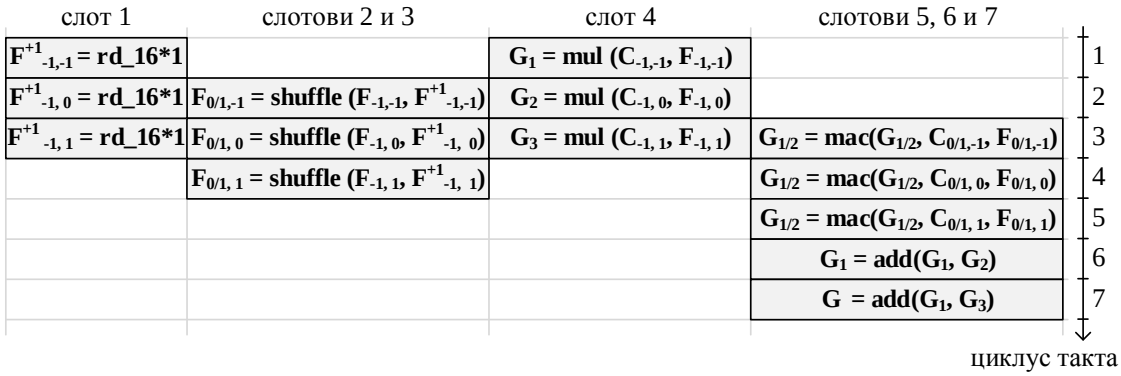
У наредним итерацијама у оквиру исте линије слике, редовима означеним са $F_{-1,-1}$, $F_{-1,0}$ и $F_{-1,1}$ одговарају редови $F^{+1}_{-1,-1}$, $F^{+1}_{-1,0}$ и $F^{+1}_{-1,1}$ прочитани у претходној итерацији, респективно. Стога се за сваку од наредних итерација ови редови поново користе, а додатно се изврше три операције читања, које одговарају наредним позицијама $F^{+1}_{-1,-1}$, $F^{+1}_{-1,0}$ и $F^{+1}_{-1,1}$, као и шест „shuffle" операција. Псеудокод оптимизоване варијанте реализације приказан је на слици 3.48. Распоред операција при извршавању на векторској путањи података са седам слотова приказан је на слици 3.49. Притом, ради смањења ширине илустрације, „shuffle" операције које се извршавају паралелно у слотовима два и три, као и „mac" операције у слотовима пет и шест, приказане су једним правоугаоником. На пример, $F_{0/1,-1} := shuffle\ (F_{-1,-1}, F^{+1}_{-1,-1})$ означава операције $F_{0,-1} :=$

```
if (is_1st_out_in_line == true)
then
  rd_16*1 (F_{-1,-1});
  rd_16*1 (F_{-1, 0});
  rd_16*1 (F_{-1, 1});
else
  F_{-1,-1} := F^{+1}_{-1,-1};
  F_{-1, 0} := F^{+1}_{-1, 0};
  F_{-1, 1} := F^{+1}_{-1, 1};
end if

rd_16*1 (F^{+1}_{-1,-1});
rd_16*1 (F^{+1}_{-1, 0});
rd_16*1 (F^{+1}_{-1, 1});

F_{0,-1} := shuffle (F_{-1,-1}, F^{+1}_{-1,-1});
F_{1,-1} := shuffle (F_{-1,-1}, F^{+1}_{-1,-1});
F_{0, 0} := shuffle (F_{-1, 0}, F^{+1}_{-1, 0});
F_{1, 0} := shuffle (F_{-1, 0}, F^{+1}_{-1, 0});
F_{0, 1} := shuffle (F_{-1, 1}, F^{+1}_{-1, 1});
F_{1, 1} := shuffle (F_{-1, 1}, F^{+1}_{-1, 1});

G_1 := mul (C_{-1,-1}, F_{-1,-1});
G_2 := mul (C_{-1, 0}, F_{-1, 0});
G_3 := mul (C_{-1, 1}, F_{-1, 1});
G_1 := mac (G_1, C_{0,-1}, F_{0,-1});
G_2 := mac (G_2, C_{1,-1}, F_{1,-1});
G_1 := mac (G_1, C_{0, 0}, F_{0, 0});
G_2 := mac (G_2, C_{1, 0}, F_{1, 0});
G_1 := mac (G_1, C_{0, 1}, F_{0, 1});
G_2 := mac (G_2, C_{1, 1}, F_{1, 1});

G_1 := add (G_1, G_2);
G   := add (G_1, G_3);
```

Слика 3.48 – Псеудокод оптимизованог програма линеарног 3∗3 филтра, уз коришћење 16∗1 начина приступа паралелном меморијском подсистему.

$shuffle\ (F_{-1,-1}, F^{+1}_{-1,-1})$ у слоту четири и $F_{1,-1} := shuffle\ (F_{-1,-1}, F^{+1}_{-1,-1})$ у слоту пет. Из истог разлога су „add" операције, које се извршавају у слоту седам, приказане у истој вертикали као и „mac" операције. Ради једноставности илустрације, нису приказани операнди „shuffle" операција који дефинишу како се врши прерасподела пиксела из два улазна реда у један излазни.

Са слике се може видети да је потребно седам циклуса такта за израчунавање једног 16∗1 излаза филтра, на позицијама које нису на самом почетку линије. Притом се изврши три операције читања из меморијског подсистема и прочита четрдесет осам различитих пиксела. За израчунавање излаза филтра на почетку линије потребно је још три циклуса такта, а укупно десет, за из-



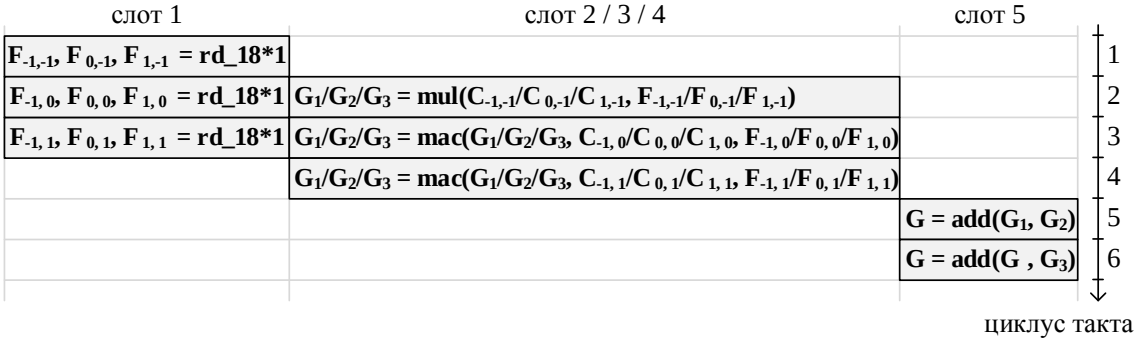
Слика 3.49 – Распоред операција при оптимизованом креирању једног 16∗1 реда линеарним 3∗3 филтром, коришћењем 16∗1 начина приступа меморијском подсистему.

вршавање додатне три операције читања из меморијског подсистема. У овом случају се прочита укупно деведесет шест пиксела. Под претпоставком да је слика довољне ширине, просечан број циклуса такта и просечан број пиксела прочитаних из меморијског подсистема по једном израчунавању излаза филтра теже ка седам и четрдесет осам, респективно.

На основу претходне анализе може се закључити да је, у односу на праволинијску реализацију, број циклуса такта оптимизоване реализације мањи за четрдесет три процента, а број прочитаних пиксела три пута мањи, чиме је постигнуто значајно убрзање обраде и значајна уштеда у потрошњи енергије меморијског подсистема. Мана оптимизоване варијанте јесте у томе што је псеудокод програма мање интуитиван и тежи за разумевање.

Трећа варијанта реализације заснована је на новом 18∗1 начину приступа меморијском подсистему. Ова варијанта извршава три операције читања, са вертикално суседних позиција, као што приказује псеудокод на слици 3.45б. Сваком операцијом прочита се три 16∗1 реда, а остатак израчунавања чини исти број „mul”, „mac” и „add” операција као у случају оптимизоване 16∗1 реализације. Притом је псеудокод остао интуитиван и лак за разумевање попут псеудокода праволинијске 16∗1 варијанте.

Распоред операција којима се овај псеудокод извршава на векторској путањи података приказан је на слици 3.50. Са слике се може видети да је потребно пет слотова са функционалним јединицама, што је за два слота мање у односу на оптимизовану 16∗1 реализацију. За израчунавање једног 16∗1 излаза филтра потребно шест циклуса такта, односно за један циклус мање и седамнаест процената брже у односу на оптимизовану 16∗1 варијанту. Како обе варијанте



Слика 3.50 – Распоред операција при креирању једног 16∗1 реда линеарним 3∗3 филтром, коришћењем новог 18∗1 начина приступа меморијском подсистему.

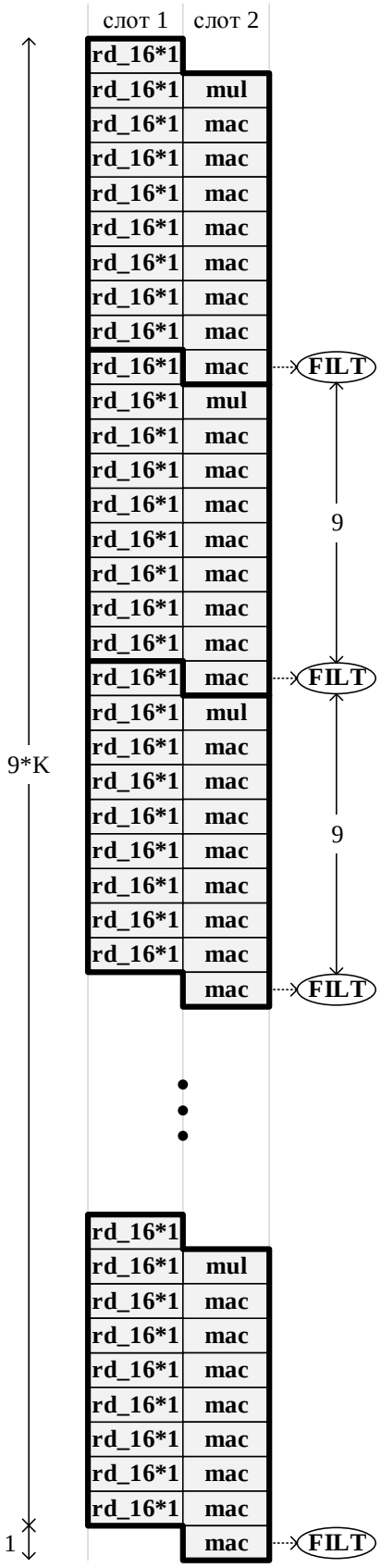
Табела 3.4 – Број циклуса такта и прочитаних пиксела за израчунавање једног 16∗1 излаза 3∗3 линеарног филтра.

Варијанта реализације	Број циклуса такта	Број прочитаних пиксела
праволинијска са познатим 16∗1	10	144
оптимизована са познатим 16∗1	7 (43%)	48 (3 пута мање)
са новим 18∗1 начином приступа	6 (67%)	54 (2,7 пута мање)

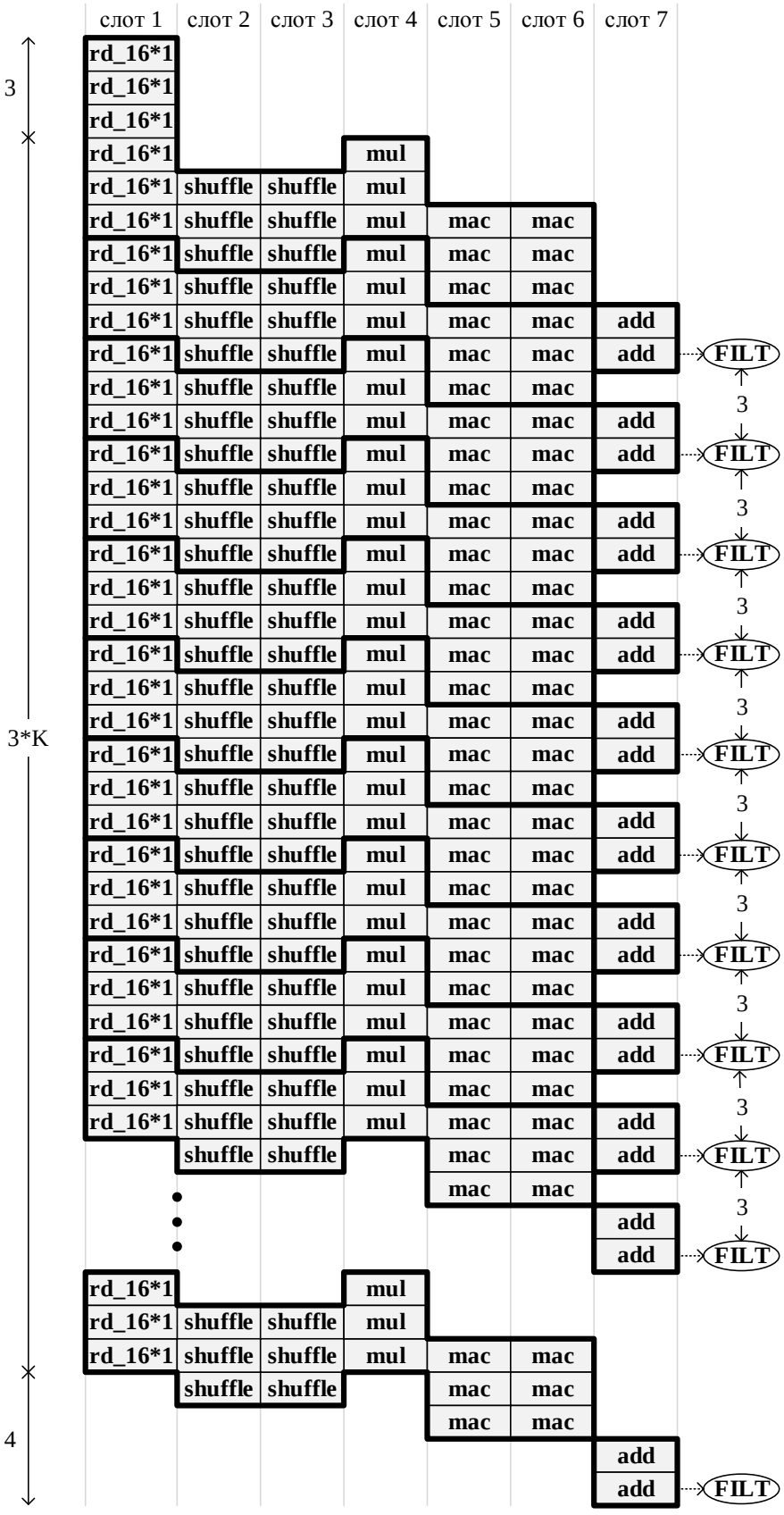
извршавају исти број „mul”, „mac” и „add” операција, убрзање обраде резултат је коришћења новог 18∗1 начина приступа и нове операције читања из меморијског подсистема, којима се истовремено прочитају три операнда „mul” операција и којима се елиминише потреба за постојањем два слота са „shuffle” операцијама.

Приликом израчунавања једног излаза филтра прочита се педесет четири пиксела, што је у просеку за шест пиксела, односно за тринаест процената више него у случају оптимизоване 16∗1 варијанте. Стога, у овом случају постојећи 16∗1 начин приступа омогућава нешто мању потрошњу енергије меморијског подсистема, јер је она пропорционална броју прочитаних пиксела. Табела 3.4 сажето приказује све релевантне бројке за варијанте реализације анализиране у овој секцији. У заградама је приказано умањење вредности посматраних параметара у односу на праволинијску 16∗1 реализацију.

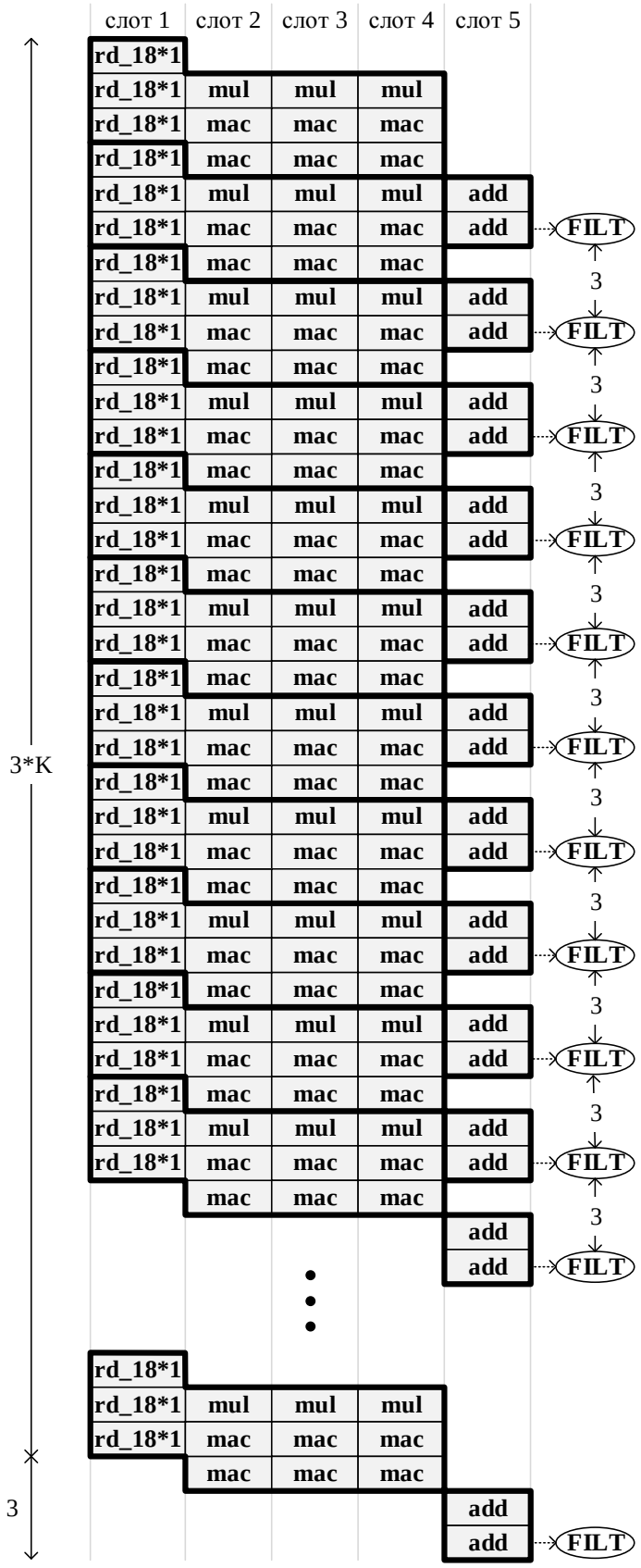
Уколико се посматра израчунавање више излаза филтра, њихово извршавање може се делимично преклопити у циљу постизања бољег искоришћења функционалних јединица и веће пропусности обраде, односно већег броја излазних редова у јединици времена. Распореди операција за преклопљено израчунавање K 16∗1 излаза 3∗3 филтра, у праволинијској 16∗1, оптимизованој 16∗1 и 18∗1 варијанти, приказани су на сликама 3.51, 3.52 и 3.53, респективно.



Слика 3.51 – Креирање K 16∗1 редова пиксела линеарним 3∗3 филтром, уз коришћење познатог 16∗1 начина приступа меморијском подсистему.



Слика 3.52 – Оптимизовано креирање K 16∗1 редова пиксела линеарним 3∗3 филтром, уз коришћење 16∗1 начина приступа меморијском подсистему.



Слика 3.53 – Креирање K 16∗1 редова пиксела линеарним 3∗3 филтром, уз коришћење новог 18∗1 начина приступа меморијском подсистему.

Са слика се може видети да се паралелном меморијском подсистему приступа у сваком циклусу такта у све три варијанте, односно да читање из меморијског подсистема представља уско грло које одређује пропусност обраде, у овом случају заједно са „mul", „mac" и „shuffle" операцијама. На тај начин је још једном потврђена полазна претпоставка овог рада, да је број приступа меморијском подсистему критичан параметар и да се обрада може убрзати његовим смањењем. Притом, као и у секцијама 3.1 и 3.2, претпоставља се најчешћи случај из праксе, а то је да процесор садржи један меморијски подсистем и један слот за приступ подсистему, а већи број слотова са аритметичко-логичким јединицима. Овакав случај је у пракси најчешћи захваљујући вишеструко нижој цени логичких кола у односу на цену меморије, у смислу заузећа површине у савременим технологијама израде чипова [12]. Другим речима, у архитектурама савремених процесора могуће је приуштити довољан број аритметичко-логичких јединица, тако да један меморијски подсистем постаје уско грло.

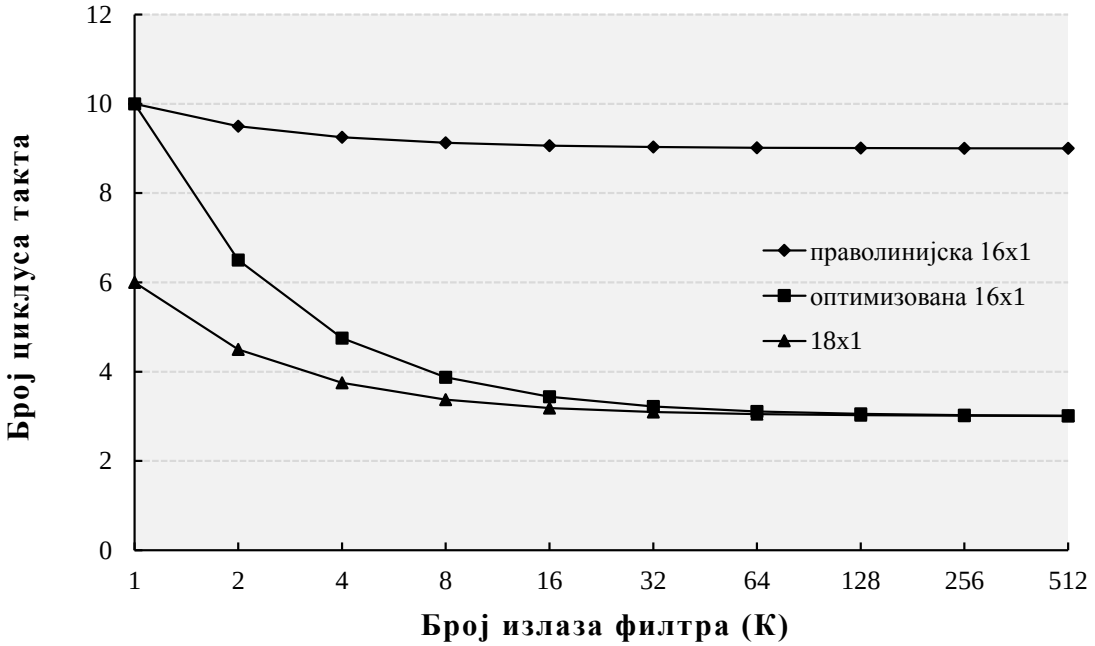
Анализом слика 3.51, 3.52 и 3.53, добијају се следеће једначине за укупан број циклуса такта потребних за израчунавање K излаза филтра:

$$\begin{aligned} C_t^s(16*1) &= 9*K+1, \\ C_t^o(16*1) &= 3*K+7, \\ C_t(18*1) &= 3*K+3, \end{aligned} \tag{3.23}$$

где C_t^s и C_t^o означавају праволинијску и оптимизовану 16∗1 варијанту. На основу претходних израза, просечан број циклуса такта по једном излазу филтра јесте:

$$\begin{aligned} C_a^s(16*1) &= \frac{C_t^s(16*1)}{K} = 9+\frac{1}{K}, \\ C_a^o(16*1) &= \frac{C_t^o(16*1)}{K} = 3+\frac{7}{K}, \\ C_a(18*1) &= \frac{C_t(18*1)}{K} = 3+\frac{3}{K}. \end{aligned} \tag{3.24}$$

Слика 3.54 илуструје просечан број циклуса такта по једном излазу филтра, за репрезентативан низ вредности параметра K и за све три варијанте реализације.



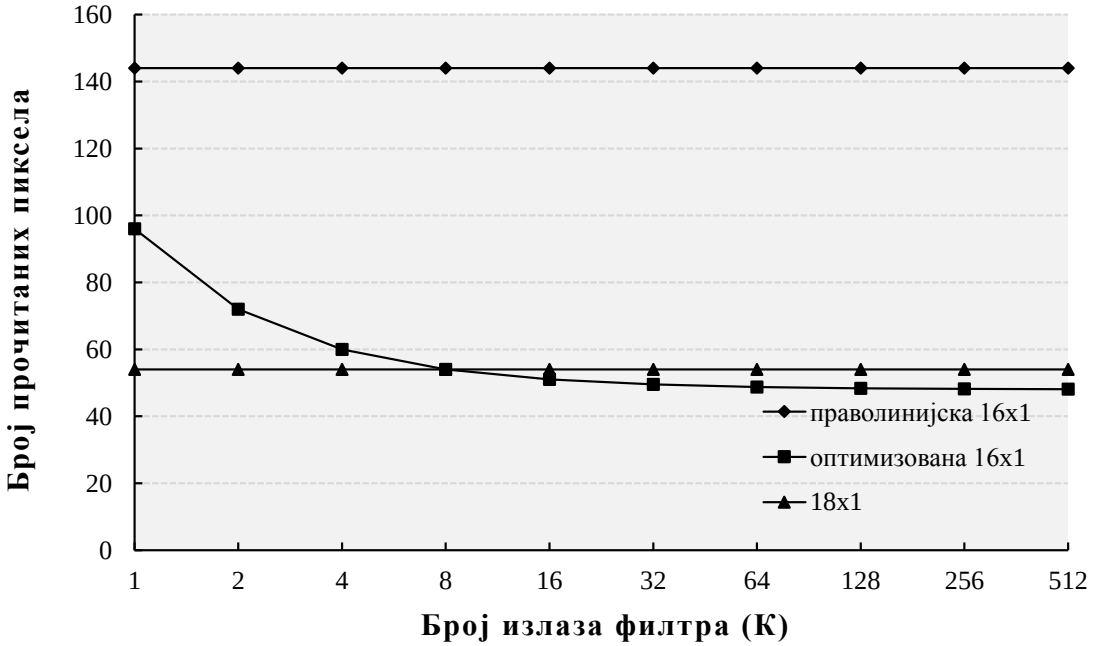
Слика 3.54 – Просечан број циклуса такта за израчунавање једног 16∗1 излаза линеарног 3∗3 филтра.

Други параметар од интереса јесте просечан број прочитаних пиксела при израчунавању једног излаза филтра, који је дат следећим изразима:

$$\begin{aligned} R_a^s(16*1) &= \frac{144*K}{K} = 144, \\ R_a^o(16*1) &= \frac{48*K+48}{K} = 48+\frac{48}{K}, \\ R_a(18*1) &= \frac{54*K}{K} = 54. \end{aligned} \tag{3.25}$$

Овај параметар представља једноставну и добру меру потрошње енергије меморијског подсистема. Потрошња подсистема у највећој мери одређена је сумом потрошњи сваке од меморијских ћелија које се читају, а та сума је пропорционална броју приступа, односно броју прочитаних пиксела из подсистема. Слика 3.55 илуструје просечан број прочитаних пиксела по једном излазу филтра.

Трећи параметар који се посматра јесте просечно искоришћење функционалних јединица векторске путање података, које се добија као однос укупног броја извршених операција у одређеном броју циклуса такта и максималног броја операција који се може извршити у истом броју циклуса такта. Како је максимални број операција једнак производу броја слотова и броја циклуса



Слика 3.55 – Просечан број прочитаних пиксела при израчунавању једног 16∗1 излаза линеарног 3∗3 филтра.

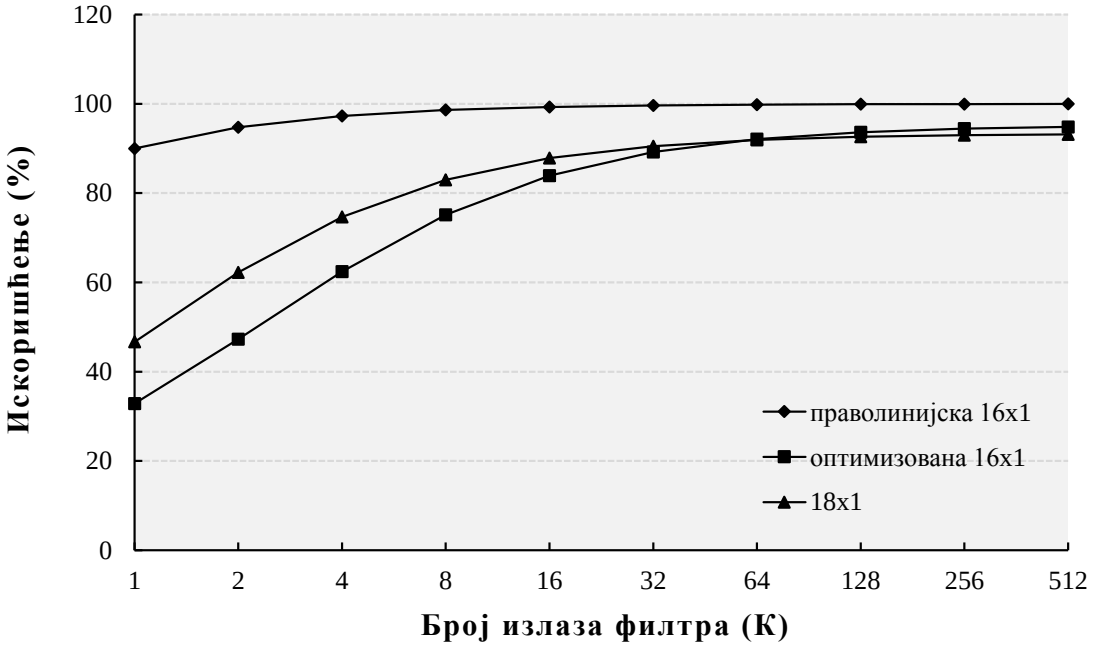
такта, просечно искоришћење функционалних јединица дато је изразима:

$$
\begin{aligned}
U_a^s(16*1) &= \frac{18*K}{2*(9*K+1)} = \frac{18*K}{18*K+2} \quad \left(\underset{K=1}{=90\%} \ \wedge \ \underset{K\to\infty}{=100\%}\right), \\
U_a^o(16*1) &= \frac{20*K+3}{7*(3*K+7)} = \frac{20*K+3}{21*K+49} \quad \left(\underset{K=1}{=33\%} \ \wedge \ \underset{K\to\infty}{=95\%}\right), \\
U_a(18*1) &= \frac{14*K}{5*(3*K+3)} = \frac{14*K}{15*K+15} \quad \left(\underset{K=1}{=47\%} \ \wedge \ \underset{K\to\infty}{=93\%}\right).
\end{aligned} \tag{3.26}
$$

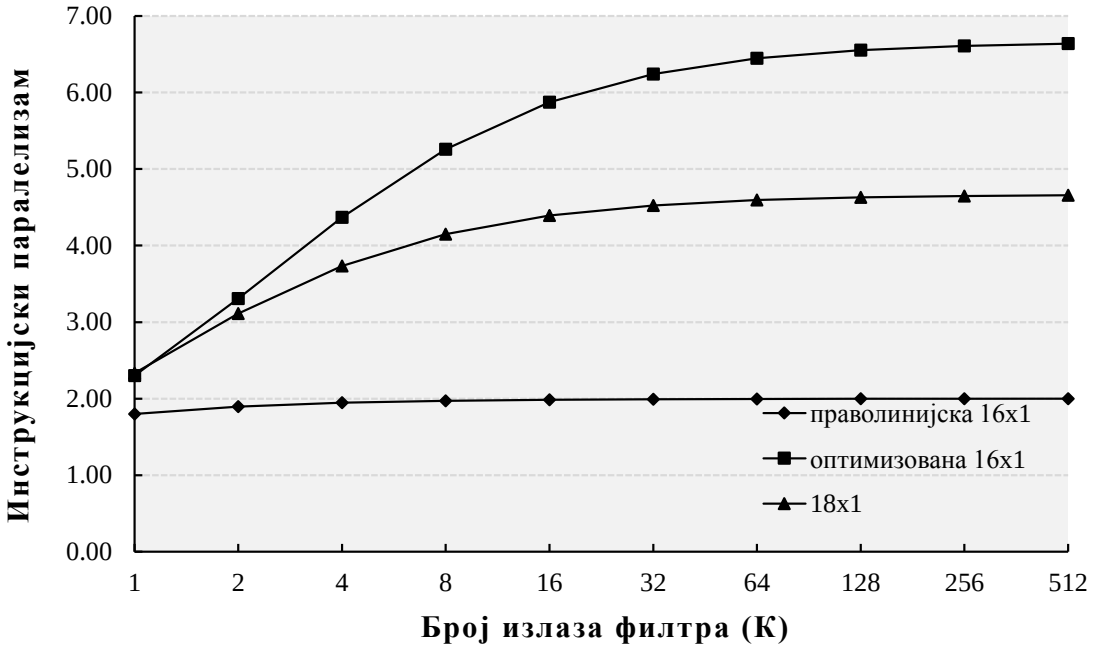
Слика 3.56 приказује просечно искоришћење функционалних јединица у процентима, за различите вредности параметра K.

Слично искоришћењу функционалних јединица, може се одредити и просечан инструкцијски паралелизам, односно просечан број операција које се извршавају у паралели. Просечан инструкцијски паралелизам представља количник између укупног број извршених операција и броја циклуса такта у којима се оне извршавају:

$$
\begin{aligned}
I_a^s(16*1) &= \frac{18*K}{9*K+1} \quad \left(\underset{K=1}{=1,8} \ \wedge \ \underset{K\to\infty}{=2,0}\right), \\
I_a^o(16*1) &= \frac{20*K+3}{3*K+7} \quad \left(\underset{K=1}{=2,3} \ \wedge \ \underset{K\to\infty}{=6,7}\right), \\
I_a(18*1) &= \frac{14*K}{3*K+3} \quad \left(\underset{K=1}{=2,3} \ \wedge \ \underset{K\to\infty}{=4,7}\right).
\end{aligned} \tag{3.27}
$$



Слика 3.56 – Просечно искоришћење функционалних јединица векторске путање података при израчунавању једног 16∗1 излаза линеарног 3∗3 филтра.



Слика 3.57 – Просечан инструкцијски паралелизам при израчунавању једног 16∗1 излаза линеарног 3∗3 филтра.

Слика 3.57 приказује вредности просечног инструкцијског паралелизма за низ вредности параметра K и за све три варијанте реализације.

Анализом графикона приказаних на сликама 3.54 – 3.57, може се доћи до следећих закључака. У односу на друге две варијанте, праволинијску 16∗1 ре-

ализацију одликује највиши степен искоришћења функционалних јединица од преко деведесет процената и најнижи инструкцијски паралелизам, за све вредности параметра K. Низак инструкцијски паралелизам последица је великог броја приступа меморијском подсистему, који се извршавају секвенцијално. Истовремено, низак инструкцијски паралелизам узрок је мале брзине обраде, која се огледа кроз велики просечан броја циклуса такта потребних за израчунавање једног излаза филтра.

Оптимизована 16∗1 варијанта смањује просечан број приступа меморијском подсистему са порастом вредности параметра K и тиме омогућава већи инструкцијски паралелизам, што је донело вишеструко убрзање обраде. Цена таквог убрзања обраде јесте највећи број потребних слотова у односу на друге две варијанте, чије је искоришћење притом најниже.

Варијанта реализације коришћењем новог 18∗1 начина приступа омогућава најбољи баланс инструкцијског паралелизма и искоришћења функционалних јединица. Стога ова варијанта постиже једнаку или већу брзину обраде у односу на оптимизовану 16∗1 реализацију, али коришћењем два слота мање. То је резултат добре прилагођености новог начина приступа и нове операције читања методи обраде. У прилог томе говори и задржавање интуитивности псеудокода на истом нивоу као код праволинијске 16∗1 реализације, а за разлику од оптимизоване 16∗1 варијанте. Једини недостатак ове варијанте реализације јесте нешто већа потрошња енергије меморијског подсистема за вредности параметра K веће од осам.

4

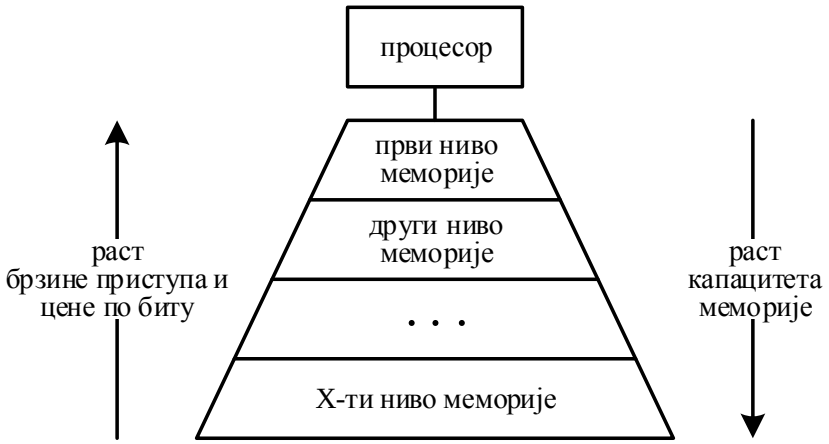
ПРЕГЛЕД ПОСТОЈЕЋИХ РЕШЕЊА

У овом поглављу дат је преглед постојећих решења паралелног меморијског подсистема. Преглед укључује опис функционалности и архитектура најважнијих подсистема кроз историју ове области. Поред тога, наводе се њихови недостаци за примену у посматраним методама обраде слике и видеа, које су анализиране у поглављу 3. Ради комплетности овог рада, прво је у секцији 4.1 дат кратак увод о меморијама и меморијским подсистемима уопште, а сам преглед паралелних меморијских подсистема дат је у секцији 4.2.

4.1 Основе меморијских подсистема

Меморија представља један од основних и најважнијих делова сваког рачунарског система. У идеалном случају, са становишта развоја и извршавања програма, рачунарски систем треба да садржи неограничено много брзе меморије [12], која је притом и ценовно прихватљива у смислу заузећа површине на чипу. Међутим, таква врста меморије не постоји на данашњем нивоу развоја технологије. Од савремених врста меморија, синхрона динамичка меморија са случајним приступом, у ознаци SDRAM[1], ценовно је прихватљива и може се приуштити велики капацитет овакве меморије. Њен недостатак је недовољна

[1] енг. Synchronous Dynamic Random Access Memory

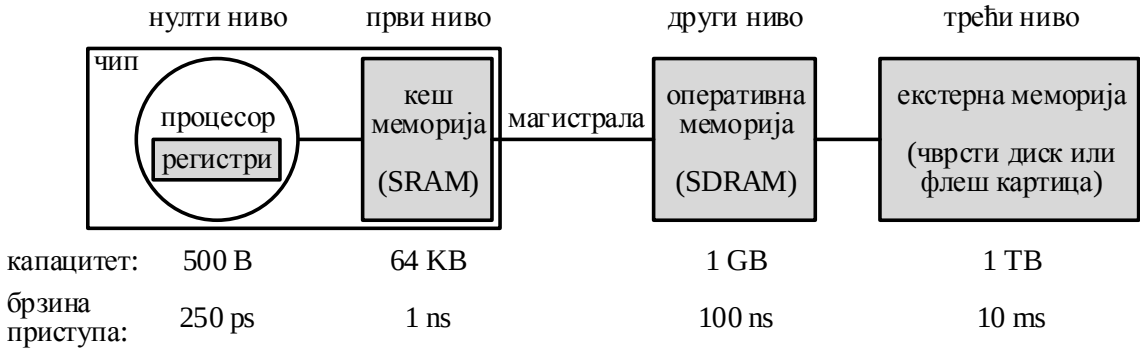


Слика 4.1 – Концепт меморијске хијерархије, приказан са X нивоа меморије.

брзина, услед чега постоји велики јаз између брзине приступа SDRAM меморији и брзине обраде података на процесору, као што је раније приказано на слици 2.2. Са друге стране је статичка меморија са случајним приступом, у ознаци SRAM[2], којој се може приступати истом брзином којом се подаци обрађују на процесору. Поред тога, SRAM троши мање енергије по биту него SDRAM, јер не постоји потреба за периодичним освежавањем меморијских ћелија како би се сачувао њихов садржај. Недостатак SRAM меморија је висока цена по биту, што ограничава капацитет који се може приуштити. Илустрације ради, у савременим применама, капацитет меморија типа SRAM износи од неколико стотина килобајта до неколико мегабајта, док је капацитет SDRAM меморија у гигабајтима [12].

Традиционално решење, које на економичан начин тежи ка идеалном случају брзе меморије неограниченог капацитета, јесте концепт меморијске хијерархије приказан на слици 4.1. Меморијска хијерархија заснива се на добро познатим особинама временске и просторне локалности програма [12], као и на компромису између брзине меморије са једне стране и цене по биту и капацитета меморије са друге стране. Меморијска хијерархија садржи више нивоа, углавном различитих врста меморије. Први ниво, најближи процесору, представља најбржу меморију најмањег капацитета, како би се процесору омогућио брз приступ, односно висока пропусност приступа меморији. Што су меморијски нивои у хијерархији даље од процесора, то је њихов капацитет већи, а брзина приступа мања. Притом, при реализацији сваког од нивоа хијерархије

[2] енг. Static Random Access Memory



Слика 4.2 – Пример меморијске хијерархије (извор: [12]).

и веза између њих, циљ је да, са становишта процесора, брзина приступа меморијској хијерархији буде једнака брзини приступа првом и најбржем нивоу, а да укупна цена меморијске хијерархије по биту буде блиска цени последњег и најјефтинијег, односно по капацитету највећег нивоа.

У зависности од реализације меморијске хијерархије, одређени број нивоа налази се на истом чипу као и путања података процесора, док су остали нивои ван чипа. Нивои који се налазе на чипу називају се меморијским подсистемом процесора и представљају тему овог рада. Већ помињана, дељена меморија ван чипа представља први ниво у хијерархији који се налази ван чипа. Поред ових, хијерархија садржи и меморије чији се садржај не губи по укидању напајања.

Пример једне меморијске хијерархије приказан је на слици 4.2, где кеш и оперативна меморија представљају меморијски подсистем на чипу и дељену меморију ван чипа, респективно, а екстерна меморија представља меморију која чува садржај и по укидању напајања. Регистри процесора, иако реализовани помоћу логичких кола, а не технологијом меморија, могу се сматрати нултим нивоом у меморијској хијерархији. Слика приказује и однос капацитета и брзина приступа за све нивое у хијерархији. Оваква меморијска хијерархија стандардна је за процесоре опште намене, како у персоналним рачунарима тако и у мобилним уређајима.

У наставку ове секције пажња ће бити усмерена на меморијски подсистем процесора, односно на нивое хијерахије на чипу, који представљају тему овог рада у ширем смислу. Код процесора опште намене, на нивоима меморијске хијерархије на чипу стандардно се користи кеш меморија [12]. Коришћењем кеш меморије сакрива се постојање меморијске хијерархије од програмера и омогућава једноставнији и бржи развој програма, захваљујући томе што радом кеш меморије у потпуности управља хардвер.

Међутим, за примене где је битна ниска цена у смислу заузећа површине и ниска потрошња енергије, концепт кеш меморија није најефикаснији. Такође, динамичке особине кеш меморија чине их непогодним за примене где је потребно да се обрада врши у реалном времену, јер је тешко предвидети време приступа и гарантовати перформансе. Из тих разлога се у оваквим применама чешће користе такозване скречпед[3] меморије [134]. Садржајем оваквих меморија експлицитно управља програмер или програмски преводилац, који воде рачуна о томе да подаци буду расположиви у меморији онда када су потребни за обраду. Пренос података из или у меморију ван чипа врши се програмирањем контролера за директан приступ меморији[4]. Захваљујући оваквој организацији меморије, не постоји потреба за комплексном управљачком логиком која би при сваком приступу проверавала да ли су адресирани подаци у меморијском подсистему или их је потребно пренети из меморије на вишем нивоу хијерархије. Из тог разлога су скречпед меморије једноставније у односу на кеш меморије, заузимају око тридесет четири процента мање површине на чипу, троше око четрдесет процената мање енергије и имају предвидиво време приступа [134, 135]. Са друге стране, недостатак скречпед меморија јесте потреба за програмирањем преноса и организацијом података у меморијском подсистему, што захтева већи труд при развоју програма.

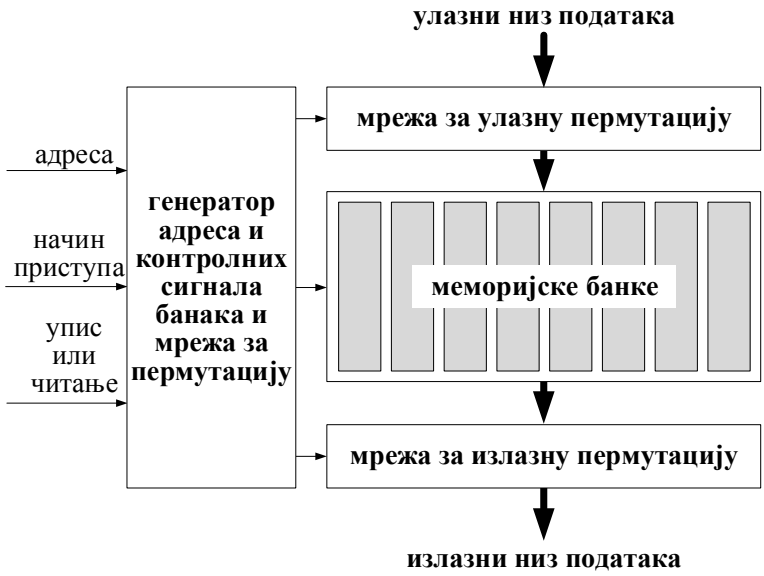
У случају векторских процесора за обраду слике и видеа у мобилним уређајима, од великог значаја су предвидивост перформанси и потрошња енергије. Стога се у тим применама најчешће користе паралелни меморијски подсистеми реализовани као скречпед меморије, као што је и решење предложено у овом раду. Наредна секција даје преглед најважнијих постојећих решења паралелног скречпед меморијског подсистема предложених у литератури.

4.2 Паралелни меморијски подсистеми

Као што је већ речено, паралелни меморијски подсистем јесте меморијски подсистем који омогућава читање и упис више података паралелно. Функционалност паралелног меморијског подсистема зависи од области примене, а дефинисана је структурама података које се у њега могу сместити и начинима приступа подацима у тим структурама. На пример, у случају обраде слике

[3] енг. scratchpad

[4] енг. Direct Memory Access controller (DMA)

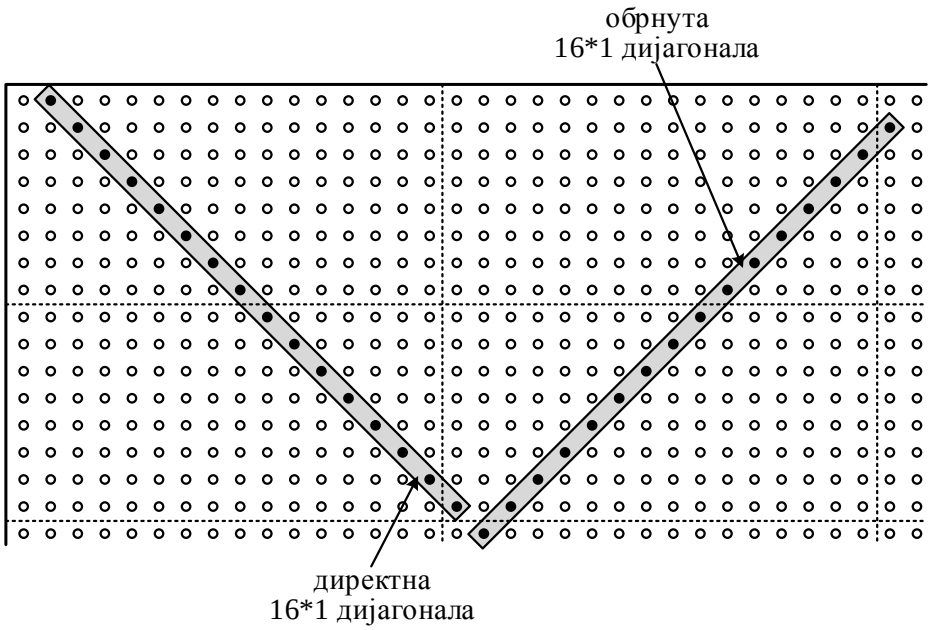


Слика 4.3 – Уопштена микроархитектура паралелног меморијског подсистема.

и видеа, најчешће коришћене структуре података јесу једнодимензионални и дводимензионални низ пиксела, а начини приступа су читање и упис редова, блокова и колона пиксела, као што је дефинисано у поглављу 3.

Архитектуре паралелних меморијских подсистема одређене су бројем меморијских банака, обрасцем распоређивања података у банке и управљачком логиком. Под меморијском банком подразумева се низ меморијских локација, односно речи, које се могу адресирати независно од речи других банака. Притом, свака адресибилна реч складишти одговарајући број података, односно ширина банке одговара том броју података. Образац распоређивања дефинише у којој меморијској банци, којој речи банке и којој позицији унутар те речи се налази посматрани податак. Циљ употребе специфичних образаца распоређивања јесте да се омогући паралелан приступ свим подацима које дефинише одговарајући начин приступа, узимајући притом у обзир да се у једном тренутку у једној банци може уписати или прочитати само једна реч[5]. Управљачка логика управља радом банака, што укључује генерисање адреса и селекцију банака, као и улазну и излазну пермутацију података. Улазном пермутацијом се при упису распореде подаци улазног низа у банке у складу са коришћеним обрасцем, а при читању из подсистема излазном пермутацијом се изврши инверзно распоређивање и обезбеди одговарајући распоред података у излазном низу. Слика 4.3 приказује уопштену микроархитектуру паралелног меморијског подсистема.

[5] Претпоставка је да се користе ценовно економичне банке са једним улазно-излазним портом.



Слика 4.4 – Начини приступа директној и обрнутој дијагонали за случај $N = 16$.

Концепт паралелног меморијског подсистема први пут се помиње у шездестим годинама двадесетог века у супер-рачунарима заснованим на векторским процесорима, као што је Illiac IV [136], са циљем да се паралелизам приступа меморији усклади са паралелизмом обраде на векторском процесору. Први паралелни меморијски подсистеми представљали су једноставан низ независних меморијских банака, чија адресибилна реч складишти један податак, а подаци су се програмски распоређивали у банке на жељени начин [17].

Будник и Кук су међу првима, у раду [17], истакли потребу за приступом подацима у облику реда, блока, колоне, директне и обрнуте дијагонале од N елемената, а потом и разматрали обрасце распоређивања елемената дводимензионалног низа у банке који омогућују такве приступе. Начини приступа дијагоналама илустровани су на слици 4.4, а остали поменути начини илустровани су раније на сликама 2.3 и 3.3. Будник и Кук су притом увели појам искошеног обрасца распоређивања и предложили више његових варијанти, како би се омогућили различити начини приступа. Уопштено гледано, образац распоређивања дефинише распоред елемената дводимензионалног низа у меморијским банкама. Под искошеним обрасцем подразумева се да је дводимензионални низ смештен у поретку по врстама у банке, а да свака врста, односно линија дводимензионалног низа почиње од различите банке, како би се омогућио паралелни приступ вертикално суседним елементима низа. Варијанта искошеног обрасца

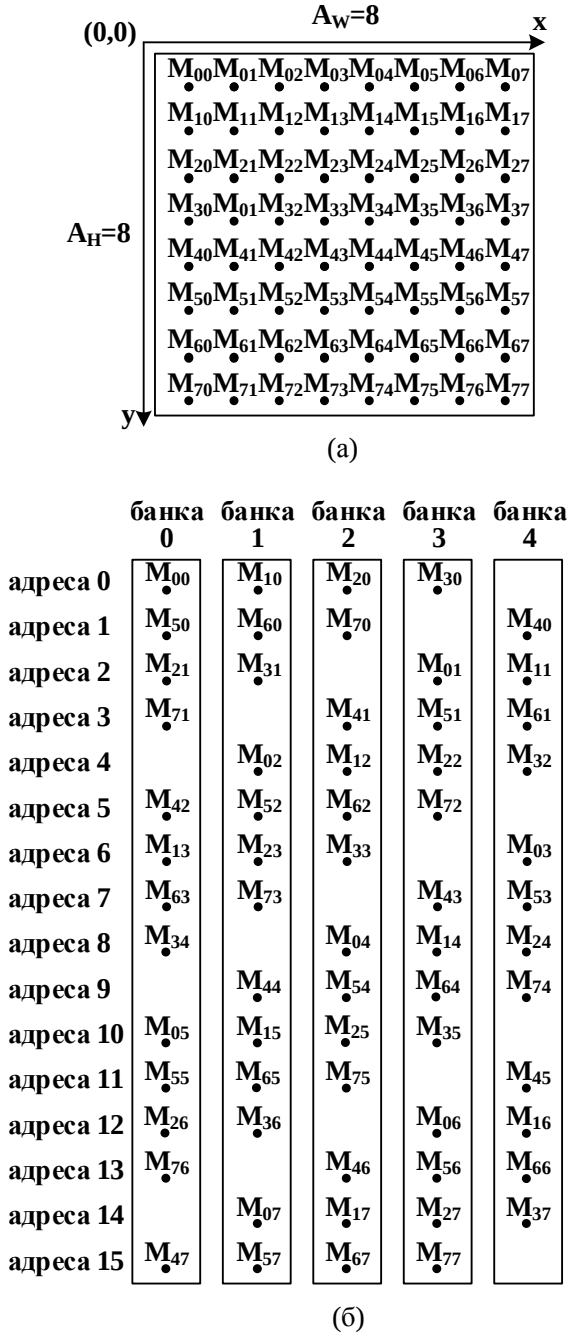
распоређивања, која представља део архитектуре решења предложеног у овом раду, детаљно је описана и илустрована у секцији 5.1.2.

Будник и Кук су даље анализирали, у истом раду [17], подржане начине приступа са датим бројем меморијских банака и различитим варијантама искошеног обрасца распоређивања. Анализом су показали да постоје одређена ограничења у подржаним начинима приступа уколико је број банака једнак неком степену броја два. Наиме, у том случају и у зависности од коришћене варијанте искошеног обрасца распоређивања, подржана су два или три од наведених пет начина приступа. Стога су анализирали и више случајева када је број банака мало већи и мало мањи од датог степена броја два и закључили да је потребан прост број банака већи од тог степена броја два, како би сви наведени начини приступа били подржани.

Иако су закључци о броју банака веома важни, то није једини проблем који је требало решити у почетку развоја паралелних меморијских подсистема. Распоређивање података у банке при упису и њихово адресирање и инверзно распоређивање при читању није било погодно вршити програмским путем, већ је била потребна реализација у хардверу како би се олакшало коришћење меморијског подсистема. Притом, у случају меморијског подсистема са простим бројем банака, управљачка логика за адресирање је комплексна јер захтева реализацију целобројног дељења и модуло операције са простим бројем. Лаври и Вора су анализирали ове проблеме у раду [18] и предложили решење паралелног меморијског подсистема које се заснива на простом броју меморијских банака, реализује адресирање у хардверу и подржава свих пет, раније наведених, начина приступа. Образац распоређивања података у банке у поретку по колонама, који су Лаври и Вора предложили, описан је следећим једначинама:

$$\begin{aligned} bank &= (y * A_W + x + base) \bmod B, \\ word &= \frac{(y * A_W + x + base)}{N}, \end{aligned} \tag{4.1}$$

где x и y представљају координате елемента у дводимензионалном низу коме се жели приступити. Редни број банке и адреса речи унутар те банке, у којима се налази жељени елемент, означени су са $bank$ и $word$, респективно. Ширина дводимензионалног низа, смештеног у меморијски подсистем, је A_W елемената, $base$ означава почетну адресу од које је низ смештен у меморијски подсистем, а B и N су број банака и паралелизам векторског процесора, респективно. Обра-



Слика 4.5 – (а) Пример дводимензионалног низа података димензија 8∗8. (б) Образац распоређивања елемената датог дводимензионалног низа у меморијске банке, предложен у [18].

зац распоређивања, описан претходним једначинама, илустрован је на слици 4.5 за случај $B = 5$, $N = 4$, $base = 0$ и $A_W = 8$. Може се приметити да су неке локације у меморијским банкама неискоришћене. То је последица поједностављења другог израза у једначини 4.1, где се дељење врши са N, што је степен броја два, а не са B који је прост број. Непотпуно искоришћење меморијских

локација представља један од недостатака решења које су предложили Лаври и Вора. Други велики недостатак овог решења јесте потреба за реализацијом комплексне модуло операције са простим бројем B. Стога је у литератури предложен низ паралелних меморијских подсистема код којих је поједностављено израчунавање адреса или је изабран број банака једнак неком степену броја два, уз компромис да се не подржи свих пет начина приступа које су Будник и Кук навели.

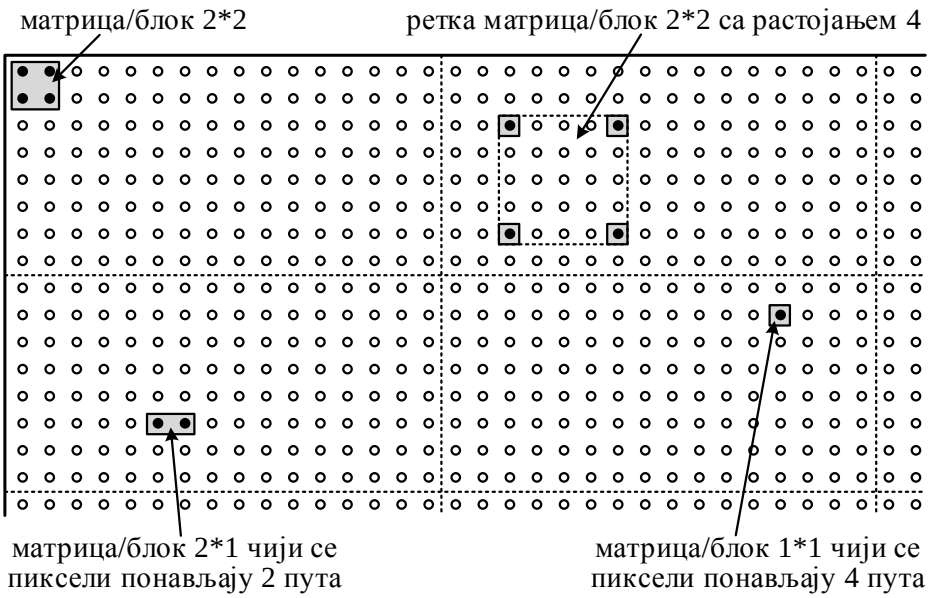
Исте проблеме као Лаври и Вора анализирали су нешто раније Вурхис и Морин у раду [24]. За примену у обради слике они су предложили низ меморијских подсистема са $B = p * q$ и више од $p * q$ банака и управљачком логиком за израчунавање адреса у банкама, где су p и q параметри подсистема. У зависности од броја банака, предложени подсистеми подржавају одређене или све начине приступа блоку од $p * q$, реду од $pq * 1$ и колони од $1 * pq$ пиксела.

Један од првих паралелних меморијских подсистема за примену у обради слике предложио је Парк у раду [25]. Парков меморијски подсистем заснива се на $B = p * q + 1$ меморијских банака и подржава приступ блоку од $p * q$, реду од $pq * 1$ и колони од $1 * pq$ пиксела, где су p и q параметри подсистема. У односу на претходно предложене подсистеме, допринос Парковог решења јесте једноставнија управљачка логика за израчунавање адреса пиксела унутар меморијских банака. Једноставнија логика постигнута је раздвајањем генерисања адреса од њиховог прослеђивања одговарајућим банкама. Образац распоређивања пиксела у банке који је Парк предложио дефинисан је следећим једначинама:

$$\begin{aligned} bank &= (x * q + y) \bmod (p * q + 1), \\ word &= \frac{x}{p} * S + \frac{y}{q}, \end{aligned} \tag{4.2}$$

где је S цео број такав да важи $S \geq \lceil A_W : q \rceil$ и да вредност $S * \lceil A_H : p \rceil$ није већа од капацитета једне меморијске банке. Притом, A_H представља висину дводимензионалног низа смештеног у меморијски подсистем, изражену у линијама. Како $p * q + 1$ није степен броја два, недостатак овог, као и претходно наведених решења, јесте потреба за реализацијом комплексне модуло операције у оквиру управљачке логике за адресирање банака.

У свом каснијем раду [21] Парк је предложио убрзање адресне логике применом принципа табеле пресликавања. Наиме, предложио је израчунавање адреса у банкама унапред и смештање у додатну меморијску банку. Приступ тој банци



Слика 4.6 – Подржани начини приступа дељене матричне меморије предложене у [20], у случају варијанте процесора који обрађује четири пиксела у паралели.

у једном циклусу такта у потпуности замењује израчунавање адреса у више циклуса такта. Овај меморијски подсистем предложен је за примену у области рачунарске графике и подржава свих пет начина приступа које су навели Будник и Кук у [17].

Хајнрикс и коаутори рада [20] предложили су векторски процесор дигиталног сигнала високих перформанси за примену у обради слике, који обрађује шеснаест пиксела у паралели и садржи паралелни меморијски подсистем. Подсистем назван „дељена матрична меморија"[6] састоји се од двадесет пет меморијских банака и омогућава приступ матрицама, односно блоковима од шеснаест пиксела. Притом матрице могу бити стандардне, ретке и са поновљеним пикселима. Код ретких матрица, између елемената постоји задато растојање. У случају матрица са поновљеним пикселима, из меморије се чита мање од шеснаест пиксела, а пиксели се понављају како би се попунио вектор процесора. Аутори рада [20] реализовали су прототип процесора који обрађује четири пиксела у паралели, а одговарајућа матрична меморија састоји се од девет меморијских банака. Подржани начини приступа за тај случај илустровани су на слици 4.6. Приступи ретким матрицама и матрицама са поновљеним пикселима погодни

[6] енг. shared matrix memory

су за примену у трансформацијама слике као што су брза Фуријеова[7] и дискретна косинусна трансформација[8]. Израчунавање адреса банака подељено је на израчунавање хоризонталне и вертикалне адресе. На тај начин је логика за адресирање поједностављена. Међутим, као недостатак и даље остаје потреба за реализацијом модуло и операције целобројног дељења са пет и три у случајевима паралелизма процесора од шеснаест и четири пиксела, респективно.

У циљу додатног поједностављења логике за адресирање банака, Ли је предложио изокренути образац распоређивања елемената[9] у раду [19]. У складу са изокренутим обрасцем, дводимензионални низ од $N * N$ елемената смешта се у $N = 2^n$ банака, где је n паран и позитиван цео број, а N паралелизам векторског процесора. Захваљујући томе што је број банака степен броја два, управљачка логика за генерисање адреса значајно је једноставнија него код претходних подсистема и састоји се од n ексклузивно-или логичких кола. Овај меморијски подсистем омогућава приступ реду, колони, блоку и ретком блоку попут оног који је илустрован на слици 4.6. Како користи N меморијских банака, Лијев подсистем представља један од првих N-банковних подсистема.

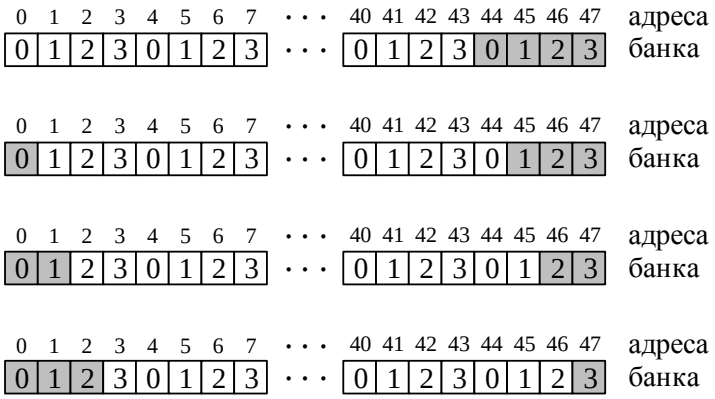
Поред Лијевог подсистема, предложен је читав низ N-банковних подсистема, где је вредност N једнака неком степену броја два, за примену у области обраде слике и видеа [27–35]. Танскан и коаутори су, у раду [27], предложили паралелни меморијски подсистем за примену у кодовању и компресији видеа, посебно прилагођен естимацији кретања упаривањем блокова. Подсистем је заснован на N меморијских банака ширине једног пиксела од осам бита, које притом имају два порта за приступ, један за приступе од стране процесора и други за преносе података из и у меморију ван чипа. Подсистем [27] омогућава смештање једнодимензионалних низова и приступ непоравнатом реду од N пиксела. Аутори су анализирали случајеве када је вредност N једнака два, четири и осам. Допринос овог подсистема јесте што омогућава адресирање по модулу, односно реализацију кружног бафера у подсистему, као што је илустровано на слици 4.7. Образац распоређивања пиксела у банке дефинисан је следећим једначинама:

$$\begin{aligned} bank &= i \bmod N, \\ word &= \frac{i}{N}, \end{aligned} \tag{4.3}$$

[7] енг. Fast Fourier Transform (FFT)

[8] енг. Discrete Cosine Transform (DCT)

[9] енг. scrambled storage scheme



Слика 4.7 – Адресирање по модулу предложено у [27]. Осенчени квадрати означавају приступе.

Како је N степен броја два, реализација управљачке логике на основу претходних једначина врло је једноставна.

У свом наредном раду [28], Тансканен је са коауторима предложио подсистем који омогућава смештање дводимензионалних низова и приступе блоку, реду, колони, ретком блоку, ретком реду и реткој колони, за примену у H.263 и MPEG-4 стандардима за кодовање и компресију видеа. Притом је коришћен исти број и ширина банака као и у претходном раду Тансканена, али уз комплекснији образац распоређивања пиксела у банке. Аутори су додатно укључили и анализу случаја $N = 16$.

Кузманов и коаутори су, у раду [30], предложили скалабилни образац за распоређивање пиксела у меморијске банке. Предложени образац омогућава приступ блоковима пиксела и намењен је за примену у обрадама попут естимације кретања и дискретне косинусне трансформације. Аутори су претпоставили смештање низа од $A_W * A_H$ пиксела у $N = a * b$ банака, где је $1 \leq a \leq A_W$ и $1 \leq b \leq A_H$, са циљем да се омогући приступ блоку B од a∗b пиксела на произвољној локацији (x, y) у посматраном низу. Блок B дефинисан је следећом једначином:

$$\begin{gathered} B(x,y) = I(x+p, y+q), \\ 0 \leq p < a,\ 0 \leq q < b,\ 0 \leq x \leq A_W - a,\ 0 \leq y \leq A_H - b, \end{gathered} \tag{4.4}$$

где I означава посматрани низ. N банака је организовано у виду матрице димензија a∗b, а локација сваког пиксела блока B дефинисана је следећим јед-

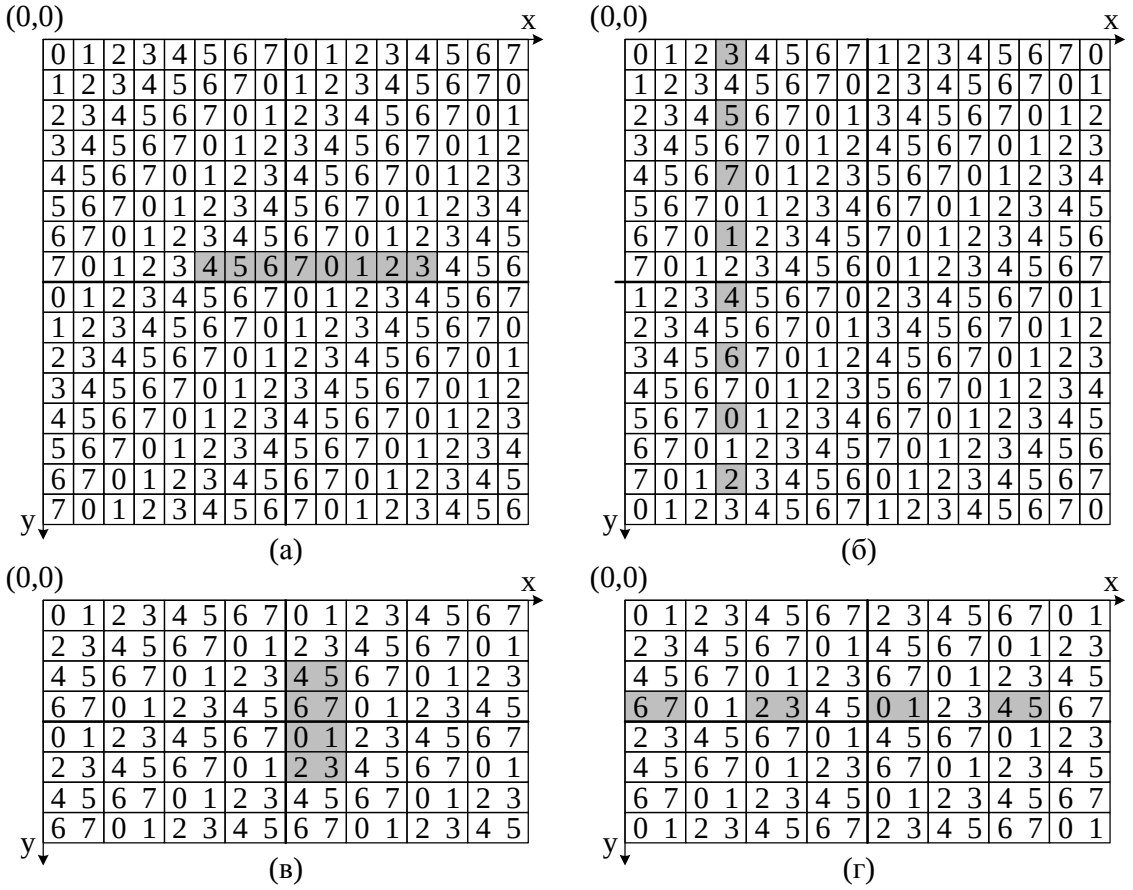
начинама:

$$
\begin{aligned}
bank_x &= (x - p) \bmod a, \\
bank_y &= (y - q) \bmod b, \\
word &= \left(\frac{y}{b} + c_y\right) * \frac{A_W}{a} + \frac{x}{a} + c_x, \\
c_x &= \begin{cases} 1, & x \bmod a > p \\ 0, & x \bmod a \leq p \end{cases} \\
c_y &= \begin{cases} 1, & y \bmod b > q \\ 0, & y \bmod b \leq q. \end{cases}
\end{aligned}
\tag{4.5}
$$

Предложени образац поседује неколико добрих особина. Израчунавање редног броја банке раздвојено је на хоризонталну и вертикалну компоненту, што доприноси једноставнијој реализацији. Уколико су изабране вредности a и b степени броја два, управљачка логика је релативно једноставна, лако се реализује и омогућава брзо израчунавање локације пиксела у банкама. Притом, у време када је предложен, образац распоређивања захтевао је једнак или мањи број меморијских банака у односу на друге обрасце.

Пенг и коаутори су, у раду [32], предложили архитектуру паралеленог меморијског подсистема за примену у кодовању и компресији видеа. Предложена архитектура се заснива на N меморијских банака, где је N степен броја два, а у раду су анализирани случајеви $N = 4$, 8 и 16, као релевантни за циљну област примене. Иновативни допринос представљају два обрасца распоређивања пиксела у банке, засновани на померању и ротирању и названи A–образац и C–образац. Ови обрасци подржавају смештање и приступање елементима различитог броја бита, односно пикселима од осам и коефицијентима од шеснаест бита, при чему је број шеснаестобитних елемената којима се може приступити у паралели двоструко мањи од броја осмобитних елемената. Од начина приступа, A-образац подржава неповравнати ред и колону, а C–образац подржава неповравнати ретки ред и ретку колону са дистанцом која је једнака неком степену броја два. Слика 4.8 илуструје A–образац и C–образац распоређивања осмобитних пиксела и шеснаестобитних коефицијената у осам меморијских банака. Осенчени су примери редова и колона којима се може приступити у сваком од четири илустрована случаја.

N-банковни подсистем који су предложили Ване и коаутори у раду [33] опти-



Слика 4.8 – Образац распоређивања коришћењем А–обрасца и С–обрасца, предложених у [32], у случају када је $N = 8$. (а) 16∗16 блок пиксела коришћењем А–обрасца, (б) 16∗16 блок пиксела коришћењем С–обрасца, (в) 8∗8 блок коефицијената коришћењем А–обрасца, (г) 8∗8 блок коефицијената коришћењем С–обрасца.

мизован је за естимацију кретања упаривањем блокова, а за примену у кодовању и компресији видеа. Њихов подсистем је дуалан, односно састоји се од два паралелна меморијска подсистема. Први подсистем, у ознаци M_C, смешта део циљног фрејма и омогућава читање поравнатог блока, а други, у ознаци M_R, смешта област претраге из референтног фрејма и омогућава читање неповравнатог блока пиксела. Поред тога, оба подсистема, M_C и M_R, омогућавају поравнати упис реда пиксела, чиме се подржавају ефикасни преноси из меморије ван чипа. Како M_C подсистем треба да омогући само поравнате приступе, он садржи четири банке ширине четири пиксела, док је M_R N-банковни подсистем и садржи шеснаест банака ширине једног пиксела. Захваљујући наведном скупу подржаних начина приступа, аутори су постигли и у раду објавили убрзање уписа у подсистем од четири пута и убрзање читања од педесет пет процената при естимацији кретања, у односу на друге подсистеме који су тада

0	4	8	C	1	5	9	D	2	6	A	E	3	7	B	F	4	8	C	0
1	5	9	D	2	6	A	E	3	7	B	F	4	8	C	0	5	9	D	1
2	6	A	E	3	7	B	F	4	8	C	0	5	9	D	1	6	A	E	2
3	7	B	F	4	8	C	0	5	9	D	1	6	A	E	2	7	B	F	3
4	8	C	0	5	9	D	1	6	A	E	2	7	B	F	3	8	C	0	4
5	9	D	1	6	A	E	2	7	B	F	3	8	C	0	4	9	D	1	5
6	A	E	2	7	B	F	3	8	C	0	4	9	D	1	5	A	E	2	6
7	B	F	3	8	C	0	4	9	D	1	5	A	E	2	6	B	F	3	7
8	C	0	4	9	D	1	5	A	E	2	6	B	F	3	7	C	0	4	8
9	D	1	5	A	E	2	6	B	F	3	7	C	0	4	8	D	1	5	9
A	E	2	6	B	F	3	7	C	0	4	8	D	1	5	9	E	2	6	A
B	F	3	7	C	0	4	8	D	1	5	9	E	2	6	A	F	3	7	B
C	0	4	8	D	1	5	9	E	2	6	A	F	3	7	B	0	4	8	C
D	1	5	9	E	2	6	A	F	3	7	B	0	4	8	C	1	5	9	D
E	2	6	A	F	3	7	B	0	4	8	C	1	5	9	D	2	6	A	E
F	3	7	B	0	4	8	C	1	5	9	D	2	6	A	E	3	7	B	F
0	4	8	C	1	5	9	D	2	6	A	E	3	7	B	F	4	8	C	0
1	5	9	D	2	6	A	E	3	7	B	F	4	8	C	0	5	9	D	1

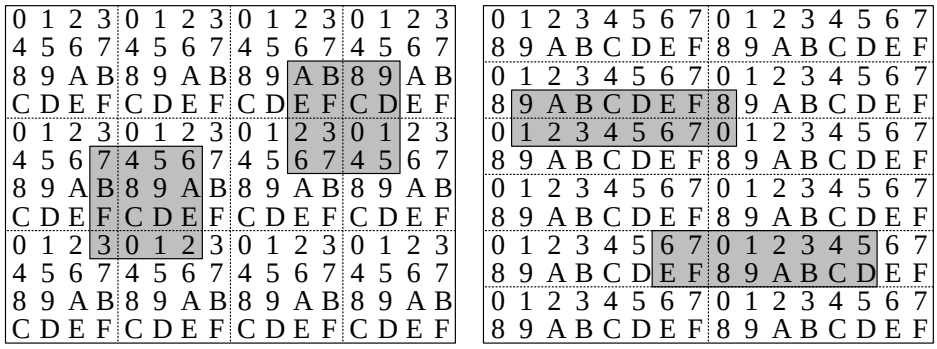
Слика 4.9 – Образац распоређивања низа од 20∗18 пиксела у шеснаест меморијских банака, предложен у [34], за случај $p = q = 4$.

били на највишем степену развоја.

За примену у области рачунарске графике, Лентарис и Реисис [34] предложили су паралелни меморијски подсистем заснован на нелинеарном обасцу распоређивања пиксела у банке и на искоришћавању корелације између узастопних приступа подсистему. Подсистем омогућава приступе реду, колони, блоку и ретком блоку од $N = p * q$ пиксела и садржи $p * q$ банака, при чему $p * q$ може бити неки степен броја два или неки други број у зависности од потреба примене. Образац распоређивања пиксела у банке дефинисан је следећом једначином и илустрован на слици 4.9 за случај $p = q = 4$:

$$\begin{aligned} bank &= \left(x * q + y + \left\lfloor \frac{x}{p} \right\rfloor \right) \bmod (p * q), \\ word &= \left\lfloor \frac{x}{p} \right\rfloor + \left\lfloor \frac{y}{q} \right\rfloor * \left\lceil \frac{A_W}{p} \right\rceil. \end{aligned} \tag{4.6}$$

Винг-Ји и коаутори предложили су процесор за обраду видеа, конкретно за примену у кодовању и компресији видеа, у раду [35]. Једна од кључних карактеристика тог процесора јесте N-банковни паралелни меморијски подсистем, који подржава начине приступа блоку различитих димензија. Поред различитих димензија блока, ширина пиксела такође може бити различита: осам, шеснаест или тридесет два бита. Притом, за приступ блоку пиксела од осам, шеснаест и тридесет два бита потребно је један, два и четири циклуса такта, респективно.



Слика 4.10 – Образац распоређивања низа од 16∗12 пиксела у шеснаест меморијских банака, предложен у [35] како би се омогућио приступ (а) 4∗4 и (б) 8∗2 блоку.

Образац распоређивања, који су предложили Винг-Ји и коаутори, илустрован је на слици 4.10, а примери приступа 4∗4 и 8∗2 блоковима су осенчени.

Лиу, Јан и Кин су, у раду [23], предложили паралелни меморијски подсистем заснован на $B = 2 * N = 2 * p * q$ банака ширине једног пиксела и који је стога означен као 2N-банковни подсистем у овом раду. Допринос рада [23] јесте оптимизовани линеарни образац распоређивања пиксела у банке, који омогућава приступ блоку, реду, колони, директној и обрнутој дијагонали, као и њиховим ретким варијантама од $p * q$ пиксела. На тај начин су омогућени сви начини приступа које су Будник и Кук навели у свом раду, али уз искоришћење свих меморијских локација у банкама. Притом је подржано адресирање по модулу и реализација кружног бафера. Предложени образац распоређивања примењује линеарно искошење у хоризонталном и нелинеарно у вертикалном правцу, а дефинисан је следећим једначинама:

$$bank = \left(\frac{y}{2 * p} * (q + 1) + (y \bmod (2 * p)) * q + x \right) \bmod (2 * p * q),$$
$$word = \frac{y}{2 * p} * \frac{A_W}{q} + \frac{x}{q}. \tag{4.7}$$

Слика 4.11 илуструје оптимизовани линеарни образац распоређивања низа од $32 * 16$ пиксела за случај када је $2 * N = 16$, односно $p = 2$ и $q = 4$, као и неке од подржаних начина приступа.

У раду [26], Лиу Шенг и коаутори предложили су такозвани „BilisPM" паралелни меморијски подсистем, који је заснован на $N = p * q$ меморијских банака ширине два пиксела и двоструко линеарном искошеном обрасцу распоређивања

y \ x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3
2	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
0	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B
4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4
5	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8
6	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C
7	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0
8	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9
9	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D
A	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1
B	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5
C	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
D	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2
E	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6
F	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A

Слика 4.11 – Оптимизовани линеарни образац распоређивања низа од 32∗16 пиксела у шеснаест меморијских банака, предложен у [23].

пиксела у банке – у хоризонталном и у вертикалном правцу. „BilisPM" омогућава приступе блоку, реду и колони од $p*q$ пиксела, њихове ретке варијанте, као и адресирање по модулу у оба правца, омогућавајући на тај начин реализацију кружног бафера. Наредне једначине дефинишу двоструко линеарни искошени образац распоређивања пиксела:

$$bank = \left((p+1) * \frac{x}{2} + y\right) \bmod (p*q),$$

$$word = \frac{x}{2} + \frac{y}{p*q} * \frac{A_W}{2}, \tag{4.8}$$

$$elem = x \bmod 2.$$

Слика 4.12 илуструје двоструко линеарни искошени образац распоређивања за случај низа од 16∗16 пиксела и $p*q = 4*2$. Осенчени су примери неких начина приступа.

Као другачији приступ у овој области, Берић и коаутори предложили су меморијски подсистем на чипу заснован на два нивоа меморијске хијерархије [36]. Подсистем са два нивоа предложен је за примену у обради видеа заснованој на естимацији и компензацији кретања, као што су области побољшања квалитета и конверзије формата видеа. Ниво хијерархије означен са L0, који је најближи векторској путањи података, заснива се на архитектури са мање од N меморијских банака, где је свака банка ширине више пиксела. Притом, банке су

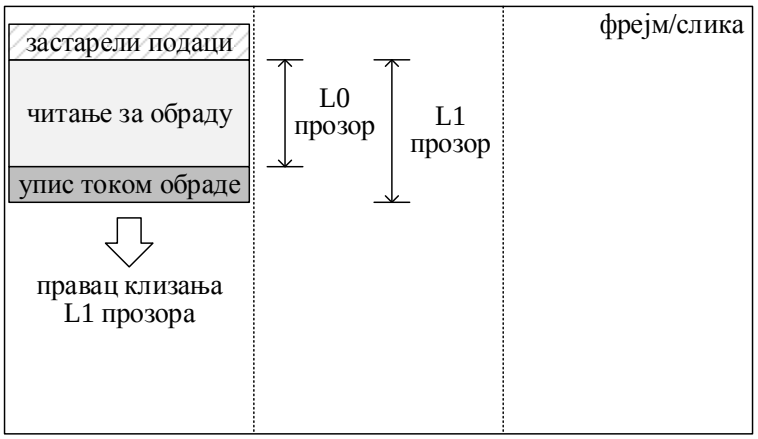
y \ x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	3	6	1	4	7	2	5								
1	1	4	7	2	5	0	3	6								
2	2	5	0	3	6	1	4	7								
0	3	6	1	4	7	2	5	0								
4	4	7	2	5	0	3	6	1								
5	5	0	3	6	1	4	7	2								
6	6	1	4	7	2	5	0	3								
7	7	2	5	0	3	6	1	4								
8	0	3	6	1	4	7	2	5								
9	1	4	7	2	5	0	3	6								
A	2	5	0	3	6	1	4	7								
B	3	6	1	4	7	2	5	0								
C	4	7	2	5	0	3	6	1								
D	5	0	3	6	1	4	7	2								
E	6	1	4	7	2	5	0	3								
F	7	2	5	0	3	6	1	4								

Слика 4.12 – Двоструко линеарни искошени образац распоређивања низа од 16∗16 пиксела у осам меморијских банака ширине два пиксела, предложен у [26].

организоване у сетове, где се један сет користи за приступ једној линији блока. Поред тога, у случају L0 нивоа меморије примењује се искошени образац распоређивања пиксела у банке, како би се омогућило читање непоравнатих и упис поравнатих блокова од N пиксела. Ниво хијерархије означен са L1, који је удаљенији од процесора, а ближи меморији ван чипа, има улогу да обезбеди да број приступа меморији ван чипа буде близу теоријског минимума, који износи један приступ по пикселу смештеном у меморији ван чипа. Стога, L1 ниво омогућава само поравнати приступ реду од N пиксела, а број и ширина меморијских банака одређују се тако да њихова цена у смислу заузећа површине и потрошње енергије буде минимална за дату примену и технологију израде чипова. Захваљујући врло једноставном начину приступа који подржава L1 ниво, његово адресирање се реализује потпуно програмски. Са друге стране, L0 ниво садржи управљачку логику која омогућава једноставније програмирање приступа.

Поред архитектуре меморијског подсистема са два нивоа, у истом раду предложен је и клизајући–L1 метод[10], којим се омогућава постизање минималног броја приступа меморији ван чипа уз прихватљив капацитет L1 нивоа меморије. Овај метод описује клизање L1 меморијског прозора, односно дводимензионалног бафера, кроз видео фрејм или слику, као што је илустровано на слици 4.13. Фрејм се најпре подели на вертикалне регионе ширине L1 прозора, а потом се један по један регион обради уз клизање L1 прозора од врха ка дну. Висина

[10]енг. sliding–L1 method



Слика 4.13 – Клизајући–L1 метод, предложен у [36].

L1 прозора већа је од висине L0 прозора за вертикални померај прозора у сваком кораку. Стога се на свакој позицији L1 прозора допуњава само та разлика, уписивањем из меморије ван чипа током обраде.

Недостатак меморијског подсистема са два нивоа хијерархије јесте потреба за преносом пиксела из L1 у L0 ниво. Ови преноси захтевају велики број читања из L1 нивоа и уписа у L0 ниво, чиме се ограничава број приступа у јединици времена, односно меморијска пропусност, која је доступна за обраду [137, 138]. Поред тога, при преносу из L1 у L0 ниво потребно је препаковати пикселе из реда у блок на векторској путањи података, јер се из L1 нивоа читају редови, а у L0 ниво уписују блокови. Ово препакивање оптерећује векторску путању података извршавањем додатних операција. Други недостатак јесте неоптимално искоришћење меморијског простора, јер се копија сваког пиксела смештеног у L0 ниво налази и у L1 нивоу меморије.

Поред наведених, заједнички недостатак свих претходно поменутих меморијских подсистема јесте непостојање подршке за читање више од N пиксела у паралели, које је потребно за ефикасну реализацију метода обраде из посматране области, као што је анализирано у поглављу 3. Из тог разлога се у наредном поглављу описује архитектура предложеног решења, која омогућава начине приступа са више од N пиксела.

5

АРХИТЕКТУРА ПРЕДЛОЖЕНОГ РЕШЕЊА

Ово поглавље описује параметризовану архитектуру паралелног меморијског подсистема која подржава и реализује потребне начине приступа, како нове тако и већ познате, као што је дефинисано у поглављу 3. После описа врши се анализа цене описане архитектуре, у смислу заузећа површине на чипу и потрошње енергије по једном приступу. Уз анализу, пореди се цена описане архитектуре са ценама других архитектура паралелног меморијског подсистема које представљају највиши степен развоја у овој области. На основу резултата поређења може се рећи да описана архитектура представља доказ да се нови начини приступа, предложени у поглављу 3, могу реализовати уз цену сличну цени других савремених архитектура. Другим речима, нови начини приступа омогућавају убрзање обраде, као што је показано у поглављу 3, без потребе за већим заузећем површине меморијског подсистема на чипу, захваљујући ефикасности описане архитектуре. Поред убрзања обраде, у највећем броју случајева добитак је и мања укупна потрошња енергије меморијског подсистема, као последица тога што описана архитектура има сличну потрошњу по једном приступу, а новим начинима се укупан број приступа вишеструко смањује.

Архитектура је потпуно параметризована и скалабилна. Параметризованост подразумева да се у време реализације меморијског подсистема може изабрати жељена вредности параметра N, који представља паралелизам путање података векторског процесора, скуп начина приступа који треба да буду подржани и укупан капацитет подсистема. Поред тога, у време развоја програма за процесор, бира се један од подржаних начина приступа, у складу са потребама конкретне методе обраде. Обе врсте избора одређују компромис између перформанси са једне и површине и потрошње меморијског подсистема са друге стране. Скалабилност је потврђена анализом цене подсистема, где је показано да површина и потрошња равномерно прате раст перформанси подсистема са порастом N, броја подржаних начина приступа и укупног капацитета.

Описана архитектура заснива се на коришћењу B независних меморијских банака, где свака банка садржи W адресибилних речи, а свака адресибилна реч банке садржи E пиксела. Сличан скуп меморијских банака, са више од једног пиксела по речи банке, коришћен је у дисертацији [138], али за реализацију само познатих начина приступа од N пиксела, као што је описано у поглављу 4. Вредности параметара B, W и E одређују се у време развоја процесора, односно у време креирања меморијског подсистема, а у складу са вредностима улазних параметара подсистема. Секција 5.2 детаљно описује потребан скуп меморијских банака и даје једначине за B, W и E.

Поред меморијских банака, описана архитектура користи искошени образац распоређивања пиксела, описан у секцији 5.1, како би се омогућило да сви пиксели произвољног блока или реда могу да се прочитају или упишу у исто време, односно паралелно. Искошени обрасци распоређивања такође су познати из литературе и коришћени су у радовима као што су [138] и [139].

Сви начини приступа блоку и реду пиксела које подржава описана архитектура, односно максималне димензије блока и реда, наведени су у секцији 5.3 у функцији од параметара меморијског подсистема. У истој секцији илустрован је максималан број пиксела који се може паралелно прочитати или уписати, за више репрезентативних вредности сваког од параметара подсистема.

Секција 5.4 описује микроархитектуру управљачке логике која се налази унутар меморијског подсистема, управља радом меморијских банака, распоређује пикселе у банке по искошеном обрасцу и омогућава једноставно приступање подсистему без потребе за познавањем унутрашње архитектуре и организације пиксела.

Проширење описане архитектуре којим се реализује концепт паралелног приступа табели пресликавања описано је у секцији 5.5.

На крају овог поглавља, секција 5.6 анализира и илуструје цену описане архитектуре, односно укупно заузеће површине на чипу и потрошњу енергије по једном приступу. Анализа је извршена за више вредности сваког од параметара меморијског подсистема, а цена је потом упоређена са ценом других архитектура које представљају највиши степен развоја.

5.1 Образац распоређивања пиксела

Под обрасцем распоређивања пиксела у меморијске банке подразумева се једнозначно одређивање локације сваког пиксела дводимензионалног низа, попут слике или фрејма, у банкама. Под претпоставком да је на располагању B банака са W адресибилиних речи и E пиксела по једној речи, локација сваког пиксела одређена је са три једначине, које дефинишу у којој банци, у којој речи банке и на којој позицији унутар адресиране речи се налази посматрани пиксел.

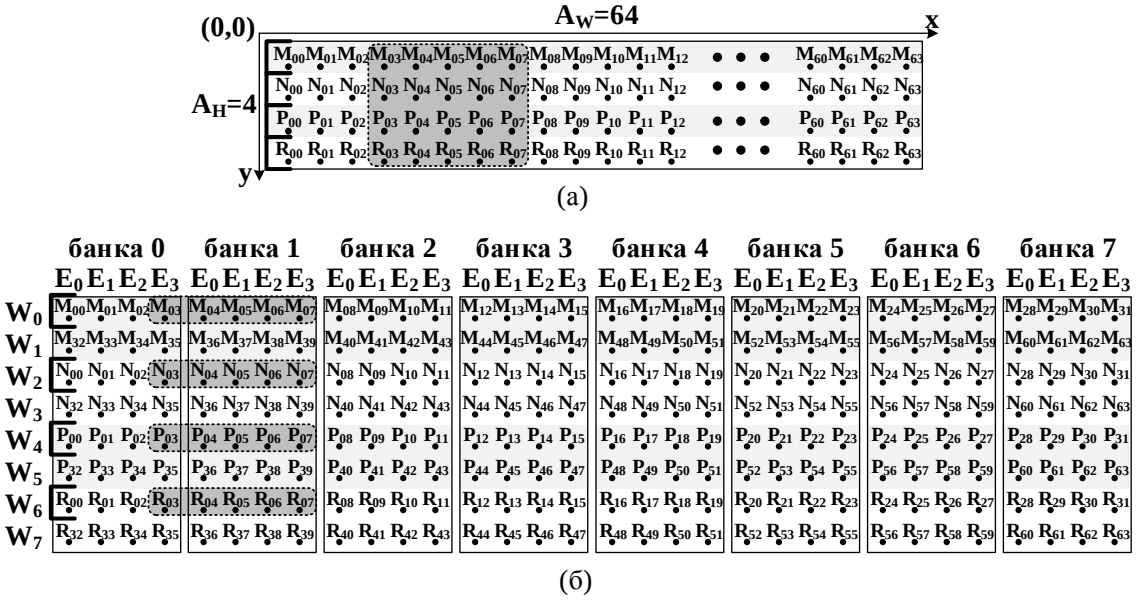
У наставку ове секције уводе се основни појмови из области образаца распоређивања пиксела у банке, помоћу једноставног праволинијског обрасца, а потом се описује искошени образац распоређивања коришћен као део архитектуре паралелног меморијског подсистема предложеног у овом раду.

5.1.1 Праволинијски образац распоређивања

Како би се описали основни појмови, једноставан праволинијски образац распоређивања пиксела дводимензионалног низа од $A_H = 4$ линије и $A_W = 64$ пиксела по линији у осам меморијских банака са осам речи у свакој банци и четири пиксела по једној речи банке приказан је на слици 5.1. По овом обрасцу, дводимензионални низ смештен је у банке у поретку по врстама[1] почевши од банке са редним бројем нула. Праволинијси образац распоређивања дефинисан је следећим једначинама:

$$\begin{aligned} bank &= \left\lfloor \frac{x}{E} \right\rfloor \bmod B, \\ word &= y * \left\lceil \frac{A_W}{B * E} \right\rceil + \left\lfloor \frac{x}{B * E} \right\rfloor, \\ elem &= x \bmod E, \end{aligned} \tag{5.1}$$

[1] енг. row-major order



Слика 5.1 – (а) Пример дводимензионалног низа пиксела. (б) Праволинијски образац распоређивања пиксела датог дводимензионалног низа у меморијске банке.

где (x, y) представља локацију пиксела у дводимензионалном низу, рачунато од горњег левог угла низа, који представља локацију $(0, 0)$ приказану на слици 5.1а, *bank* означава редни број меморијске банке, *word* је адресирана реч банке, а *elem* представља једну од E позиција пиксела у оквиру адресиране речи банке. У примеру илустрованом на слици 5.1б је $bank \in \{0, 1, ..., 7\}$, $word \in \{0, 1, ..., 7\}$ и $elem \in \{0, 1, 2, 3\}$.

Локација пиксела унутар дводимензионалног низа (x, y) може се изразити и као померај a, изражен у пикселима и рачунат у поретку по врстама почевши од локације $(0, 0)$. Притом важе следећи изрази: $a = y * A_W + x$, $y = \lfloor a : A_W \rfloor$ и $x = a \bmod A_W$. Заменом ових израза у једначине 5.1 добија се:

$$
\begin{aligned}
bank &= \left\lfloor \frac{a \bmod A_W}{E} \right\rfloor \bmod B, \\
word &= \left\lfloor \frac{a}{A_W} \right\rfloor * \left\lfloor \frac{A_W}{B * E} \right\rfloor + \left\lfloor \frac{a \bmod A_W}{B * E} \right\rfloor, \\
elem &= (a \bmod A_W) \bmod E.
\end{aligned}
\tag{5.2}
$$

Са слике 5.1 може се приметити да праволинијски образац распоређивања смешта све пикселе исте колоне дводимензионалног низа у различите речи исте меморијске банке. Како су смештени у истој банци, њима се не може приступити паралелно, под претпоставком да се користе ценовно прихватљиве банке

са једним приступним портом. Пример таквих пиксела на слици 5.1 су M_{03}, N_{03}, P_{03} и R_{03}. На основу тога се може закључити да праволинијски образац распоређивања подржава паралелан приступ само хоризонтално суседним пикселима дводимензионалног низа, односно подржава приступ реду, али да не подржава приступ блоку пиксела. Из тог разлога, проволинијски образац се не може користити у овом раду, већ представља основу за касније описани искошени образац распоређивања пиксела.

Број пиксела реда којима се може приступити у паралели зависи од поравнања приступа[2], односно од локације a_{row} крајњег левог пиксела реда коме се приступа. У случајевима када је a_{row} умножак од E, односно када се први пиксел реда налази на крајње левој позији у речи банке, за приступ се каже да је поравнат на ивицу банке и тада се може прочитати или уписати $B * E$ пиксела паралелно. Ово је максималан број пиксела који подржава праволинијски образац распоређивања примењен на дати скуп меморијских банака. Пример једног поравнатог приступа јесте приступ реду од тридесет два пиксела који почиње пикселом M_{04}.

У случајевима када је приступ непоравнат, односно када a_{row} није умножак од E, број пиксела којима се може приступити мањи је од максималног и зависи од вредности $a_{row} \bmod E$. Минималан број пиксела може се прочитати или уписати у случајевима када је $a_{row} = k * E - 1, k = \{1, 2, 3, \dots\}$, односно када је крајњи леви пиксел реда коме се приступа истовремено и крајњи десни пиксел у адресираној речи меморијске банке. Пример оваквог пиксела на слици 5.1б јесте M_{03} за вредност $a_{row} = 3$. У меморијској банци у којој се налази крајњи леви пиксел реда може се приступити само том једном пикселу, а у свакој од преосталих $B - 1$ банака може се приступити до свих E пиксела. Стога је минималан број пиксела једног реда, којима се може приступити паралелно, једнак $1 + (B - 1) * E$, у случају праволинијског обрасца распоређивања.

5.1.2 Искошени образац распоређивања

Како би се превазишло ограничење праволинијског обрасца распоређивања пиксела у меморијске банке и омогућио приступ блоку пиксела, пиксели дводимензионалног низа треба да буду распоређени тако да за непоравнати блок на произвољној локацији унутар низа не постоје два пиксела смештена у различи-

[2] енг. access alignment

Слика 5.2 – Искошени образац распоређивања пиксела дводимензионалног низа датог на слици 5.1а у меморијске банке.

тим речима исте меморијске банке. Да би се тај услов испунио, у овом раду се користи искошени образац распоређивања приказан на слици 5.2.

У случају искошеног обрасца распоређивања, дводимензионални низ пиксела смештен је у банке у поретку по врстама, као и у случају праволинијског обрасца, али су суседне линије низа међусобно померене да би истовремени приступ до више линија блока био могућ. Прецизније речено, линија n дводимензионалног низа смешта се почевши од банке $(n*S) \bmod B$, где S представља степен искошења изражен у банкама, а $n \in [0, A_H - 1]$.

У примеру илустрованом на слици 5.2 је $B = 8$, $S = 2$ и $A_H = 4$. Стога су нулта, прва, друга и трећа линија дводимензионалног низа, означене са M, N, P и R, смештене почевши од нулте, друге, четврте и шесте банке, респективно, а почетак сваке линије означен је дебљом линијом. Уколико би низ имао и пету линију, онда би она била смештена почевши поново од нулте банке. Код приказаног обрасца распоређивања, сваке четири суседне линије низа почињу од различите банке и стога се паралено може приступити блоку од четири линије. Притом, гарантована ширина непоравнатог блока, без обзира на локацију унутар низа на којој се блок налази, јесте пет пиксела[3]. Пример таквог блока, који садржи пикселе $M_{03} - M_{07}$, $N_{03} - N_{07}$, $P_{03} - P_{07}$ и $R_{03} - R_{07}$, приказан је тамнијом бојом на слици 5.2.

У општем случају искошеног обрасца распоређивања, може се приступити непоравнатом блоку од максимално $B : S$ линија у паралели, где је за приступ једној линији доступно S банака, односно једна линија може да садржи максимално $1 + (S - 1) * E$ пиксела када је приступ непоравнат или $S * E$ пиксела у случају поравнатог приступа. Максималне гарантоване димензије блока у

[3] У случају поравнатог блока и илустрованог обрасца гарантована ширина је осам пиксела.

случају непоравнатог приступа дефинисане су следећим изразима:

$$\begin{aligned} Blk_H &= \frac{B}{S}, \\ Blk_W &= 1 + (S-1) * E, \end{aligned} \tag{5.3}$$

где Blk_H представља висину блока изражену у линијама, а Blk_W ширину блока у пикселима. У случају поравнатог приступа максималне димензије блока су:

$$\begin{aligned} Blk_H^a &= Blk_H = \frac{B}{S}, \\ Blk_W^a &= S * E. \end{aligned} \tag{5.4}$$

Поред приступа блоку пиксела, искошени образац распоређивања омогућава непоравнати и поравнати приступ реду пиксела. Притом, приступи реду пиксела могу се извршавати конкурентно са приступима блоку, односно без промене распореда пиксела у банкама. То је могуће захваљујући чињеници да су пиксели сваке линије дводимензионалног низа распоређени континуално у банке у поретку по врстама, као код праволинијског обрасца. Максималан број пиксела реда коме се може приступити је:

$$\begin{aligned} Row_W &= 1 + (B-1) * E, \\ Row_W^a &= B * E, \end{aligned} \tag{5.5}$$

где Row_W означава непоравнати, а Row_W^a поравнати приступ, респективно. У илустрованом примеру искошеног обрасца распоређивања, на слици 5.2, подржани су приступи непоравнатом 29∗1 и поравнатом 32∗1 реду уз приступе непоравнатом 5∗4 и поравнатом 8∗4 блоку.

Једначине *bank*, *word* и *elem*, које дефинишу искошени образац распоређивања пиксела у меморијске банке, добијају се проширивањем једначина 5.2 параметрима S и Blk_H и дате су следећим изразима:

$$\begin{aligned} bank &= \left(\left\lfloor \frac{a \bmod A_W}{E} \right\rfloor + \left(\left\lfloor \frac{a}{A_W} \right\rfloor \bmod Blk_H \right) * S \right) \bmod B, \\ word &= \left\lfloor \frac{a}{A_W} \right\rfloor * \left\lceil \frac{A_W}{B * E} \right\rceil + \left\lfloor \frac{\left\lfloor \frac{a \bmod A_W}{E} \right\rfloor + \left(\left\lfloor \frac{a}{A_W} \right\rfloor \bmod Blk_H \right) * S}{B} \right\rfloor \bmod \left\lfloor \frac{A_W}{B * E} \right\rfloor, \\ elem &= a \bmod E. \end{aligned} \tag{5.6}$$

Важно је напоменути да искошени образац распоређивања, који подржава приступ блоку од Blk_H линија, не подржава конкурентно и приступ блоку са другим бројем линија, односно других димензија. Да би се промениле димензије блока коме се приступа, потребно је да се промени вредност S и распоред пиксела у банкама, као што је касније детаљно описано у секцији 5.3.

5.2 Архитектура меморијских банака

Ова секција одређује минималан скуп меморијских банака, дефинисан бројем банака B, бројем адресибилних речи банке W и ширином једне адресибилне речи у пикселима E, потребан да се реализује искошени образац распоређивања пиксела и омогући непоравнати приступ блоку задатих димензија.

Минималан број и ширина меморијских банака, као и вредност степена искошења S, одређују се на основу једначине 5.3 и максималне висине блока коју је потребно подржати. Максимална висина блока Blk_H и N су улазни параметри, чија се вредност одређује у време развоја процесора, односно у време креирања паралелног меморијског подсистема, док се ширина блока максималне висине, означена са Blk_W, одређује као количник N и Blk_H. Како се вредности B и E фиксирају у време креирања меморијског подсистема, потребно је да S буде променљива која се дефинише у време развоја или извршавања програма на процесору, односно онда када се бирају жељене димензија блоковског приступа, што ће у секцији 5.3 бити детаљније анализирано. Поред наведеног, следеће претпоставке и ограничења се узимају у обзир како би се омогућила ценовно прихватљива и практична реализација меморијског подсистема.

Да би се омогућила једноставна, брза и ценовно прихватљива хардверска реализација интерне контролне и управљачке логике меморијског подсистема, која, између осталог, реализује једначине 5.6, претпоставља се да су вредности B, E, S, и A_W степени броја два. На тај начин, множење, дељење и модуло операције могу се реализовати померањем улево, померањем удесно и логичком конјункцијом, респективно. Узимајући то и једначине 5.3 у обзир, долази се до закључка да Blk_H такође мора бити степен броја два. На крају, претпоставка је да је N степен броја два, што одговара пракси и савременим архитектурама векторских процесора.

Начини приступа са више од N пиксела, који треба да буду подржани, одређују везу између Blk_W, Blk_H и N. Како би се ограничио број потребних ме-

моријских банака и цена архитектуре, одлука у овом раду јесте да се омогући приступ блоку чија је ширина, у случају максималне висине Blk_H, за један пиксел већа од блока исте висине и укупно N пиксела. Другим речима, веза између горе наведених параметара дефинисана је изразом $Blk_W = 1 + N : Blk_H$. Као што ће у секцији 5.3 бити показано и прецизно дефинисано, у случају овакве архитектуре и блокова чија је висина мања од Blk_H, могуће је приступити до више од једног додатног пиксела по линији блока. На тај начин омогућена је ефикасна подршка, не само за хоризонталну линеарну интерполацију, већ и за H.264/MPEG-4 интерполацију и многобројне просторне филтре, за које су потребни начини приступа одређени у поглављу 3.

Уколико се у обзир узму израз $Blk_W = 1 + N : Blk_H$ и други израз у једначини 5.3, $Blk_W = 1 + (S - 1) * E$, долази се до закључка да наредна неједначина мора бити испуњена како би се подржао приступ блоку од Blk_H линија и Blk_W пиксела по линији:

$$1 + (S - 1) * E \geq 1 + \frac{N}{Blk_H}. \tag{5.7}$$

У циљу минимизовања броја потребних банака B, а узимајући у обзир први израз у једначини 5.3, $Blk_H = B : S$, вредност S треба да буде минимална могућа за дату вредност Blk_H. Минимална вредност S, таква да неједначина 5.7 има решење за све могуће вредности осталих параметара, јесте:

$$S_{min} = 2. \tag{5.8}$$

Заменом S са S_{min} у првом изразу једначине 5.3, добија се минималан број потребних меморијских банака:

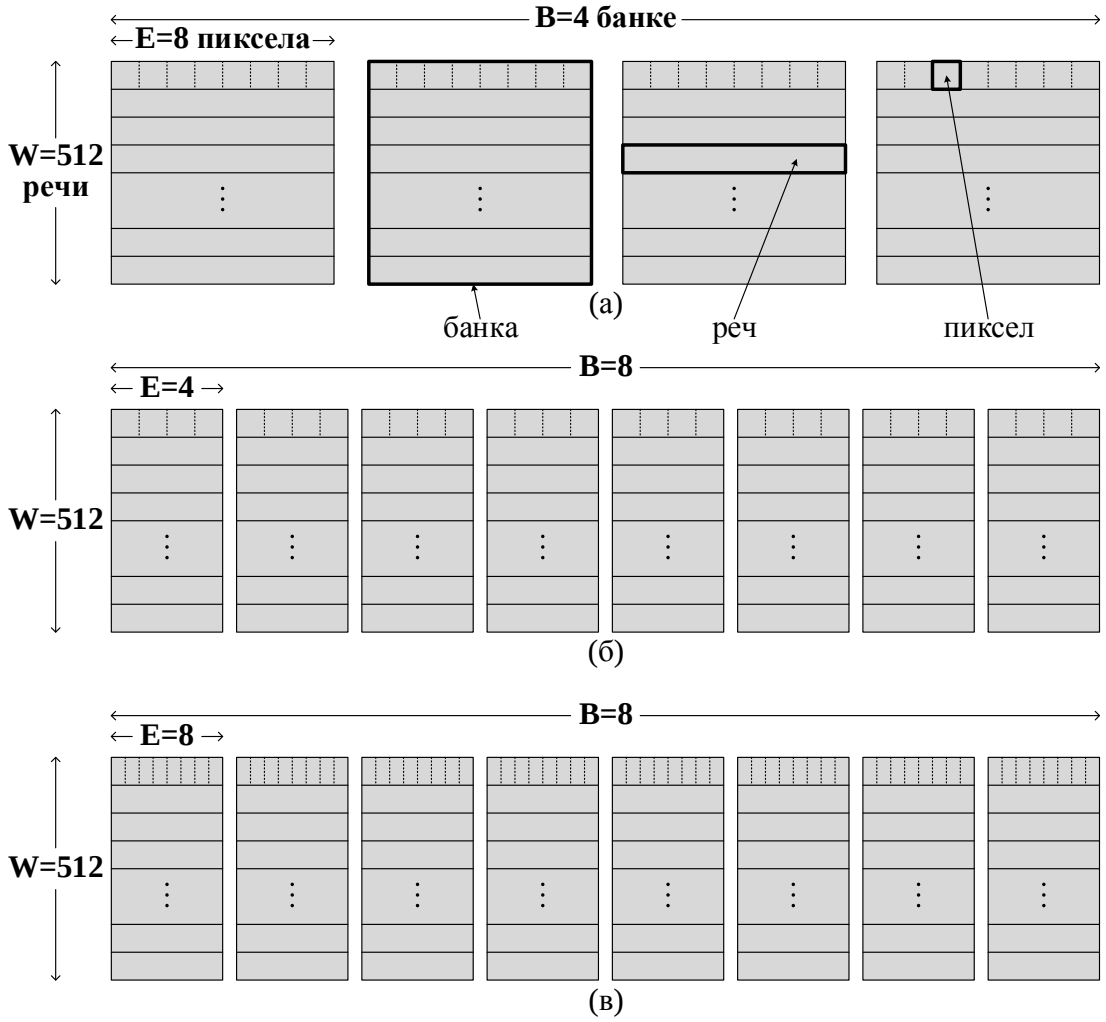
$$B = 2 * Blk_H. \tag{5.9}$$

На основу једначине 5.8, добија се да је минимална вредност E која задовољава неједначину 5.7:

$$E = \frac{N}{Blk_H}. \tag{5.10}$$

Број адресибилних речи меморијских банака W добија се на основу укупног капацитета C меморијског подсистема, израженог у речима од N пиксела:

$$W = \frac{C * N}{B * E} = \frac{C}{2}. \tag{5.11}$$



Слика 5.3 – Три архитектуре меморијских банака дефинисане укупним капацитетом од $C = 1024$ речи од N пиксела и параметрима: а) $N = 16$ и $Blk_H = 2$, б) $N = 16$ и $Blk_H = 4$ и в) $N = 32$ и $Blk_H = 4$.

Капацитет C меморијског подсистема такође је параметар задат у време креирања меморијског подсистема као N и Blk_H. Илустрације ради, слика 5.3 приказује три случаја архитектуре меморијских банака дате за три комбинације вредности параметара N, Blk_H и C.

5.3 Подржани начини приступа блоку и реду

Ова секција наводи све начине приступа које подржава описана архитектура, односно искошени образац распоређивања пиксела и скуп меморијских банака описани у секцијама 5.1 и 5.2, респективно. Како су образац распоређивања и скуп банака параметризовани, једначине које описују подржане начине

приступа такође су параметризоване. Ради јасноће, ова секција наводи примере подржаних начина приступа за репрезентативне вредности параметара.

За дати скуп меморијских банака, одређен у време креирања меморијског подсистема на основу једначина 5.9, 5.10 и 5.11 и конкретних вредности параметара N, Blk_H и C, димензије блока коме се приступа могу се изабрати у време развоја или извршавања програма, односно пре него што се пиксели упишу у меморијске банке. Избор димензије блока врши се подешавањем степена искошења S обрасца распоређивања пиксела у меморијске банке, у складу са једначином 5.3. Вредност параметра S може се подесити на сваки степен броја два између минималне и максималне вредности, где минимална износи два у складу са једначином 5.8, а максимална износи B меморијских банака[4]:

$$S_i = S_{min} * 2^i = 2^{i+1}, \quad i \in [0, \log_2(Blk_H)]. \tag{5.12}$$

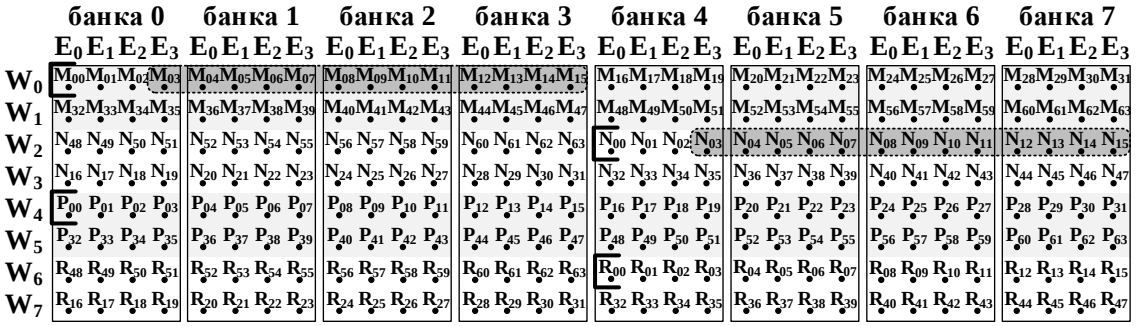
Свака вредност S_i омогућава одређени број нових начина приступа непоравнатом блоку са више од N пиксела. Као прво, свака вредност S_i омогућава неповравнати приступ блоку максималне висине Blk_{Hi} линија и максималне ширине Blk_{Wi} пиксела по линији:

$$\begin{aligned} Blk_{Hi} &= \frac{B}{S_i} = \frac{2 * Blk_H}{2^{i+1}} = \frac{Blk_H}{2^i}, \\ Blk_{Wi} &= 1 + (S_i - 1) * E = 1 + (2^{i+1} - 1) * \frac{N}{Blk_H}. \end{aligned} \tag{5.13}$$

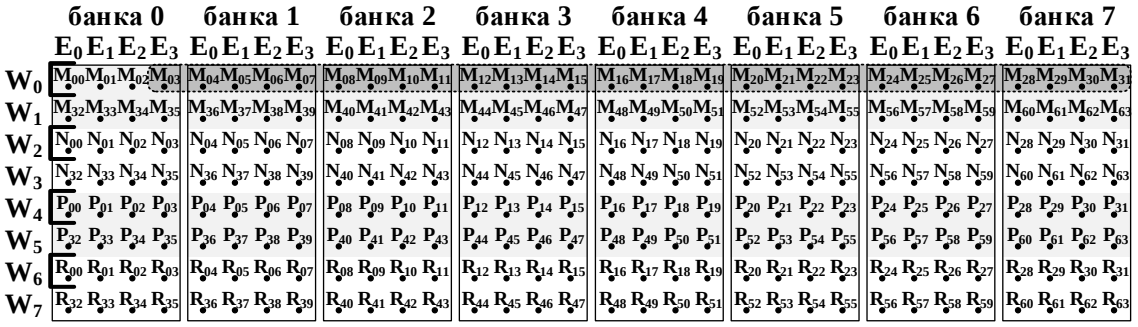
На пример, када су вредности $N = 16$ и $Blk_H = 4$, што одговара архитектури банака на слици 5.3б, степен искошења S_i може бити подешен на два, четири или осам. Одговарајуће максималне димензије блока коме се може приступити $Blk_{Wi} * Blk_{Hi}$ су 5∗4, 13∗2 и 29∗1. Илустрација меморијских банака, распореда пиксела дводимензионалног низа димензија 64∗4 пиксела и приступа блоку максималних димензија за ова три случаја дата је на сликама 5.2, 5.4а и 5.4б.

Поред блокова максималних димензија, вредности S_i веће од $S_{min} = 2$ подржавају и приступе ужим блоковима који такође садрже више од N пиксела. У илустрованом примеру такви начини приступа су 12∗2, 11∗2, 10∗2 и 9∗2 за $S_i = 4$ и 28∗1, 27∗1, ..., 17∗1 за $S_i = 8$. За овакве начине приступа распоред пиксела у меморијским банкама идентичан је распореду у случају блокова максималних димензија, као што је илустровано на слици 5.4.

[4] Степен искошења од B банака еквивалентан је степену искошења од нула банака.



(а)



(б)

Слика 5.4 – Меморијске банке и распоред пиксела дводимензионалног низа датог на слици 5.1а за случај $N = 16$, $Blk_H = 4$ и подешавање степена искошења (а) $S_i = 4$ и (б) $S_i = 8$ што омогућава приступ блоку од максимално (а) 13∗2 и (б) 29∗1 пиксела.

Изрази 5.13 такође укључују и познате начине приступа непоравнатом блоку од N пиксела, чије су димензије дате следећим једначинама:

$$\begin{aligned} Blk^N_{Hi} &= Blk_{Hi} = \frac{Blk_H}{2^i}, \\ Blk^N_{Wi} &= \frac{N}{Blk^N_{Hi}} = 2^i * \frac{N}{Blk_H}. \end{aligned} \tag{5.14}$$

У случају када је $N = 16$ и $Blk_H = 4$ подржани начини приступа непоравнатом блоку од N пиксела су 4∗4, 8∗2 и 16∗1 за $S_i = 2$, $S_i = 4$ и $S_i = 8$, респективно.

Из једначина 5.13 и 5.14 може се видети да је висина блока иста и константна за начине приступа са више од N пиксела и начине од N пиксела. Са друге стране, ширина блока је различита, а једна вредност S_i подржава све вредности ширине блока у опсегу између Blk^N_{Wi} и Blk_{Wi}. Стога је број различитих димензија блока подржан једном вредношћу S_i, односно једним распоредом пиксела у меморијским банкама једнак $Blk_{Wi} - Blk^N_{Wi} + 1 = 2 + (2^i - 1) * (N : Blk_H)$. То у илустрованом примеру износи два, шест и четрнаест начина приступа за $i = 0$, $i = 1$ и $i = 2$, односно $S_i = 2$, $S_i = 4$ и $S_i = 8$, респективно.

Поред приступа непоравнатом блоку, свака вредност S_i омогућава и приступ поравнатом блоку. На основу једначина 5.4, максималне димензије у случају поравнатог блока су Blk^a_{Hi} линија и Blk^a_{Wi} пиксела по линији:

$$\begin{aligned} Blk^a_{Hi} &= \frac{B}{S_i} = Blk_{Hi}, \\ Blk^a_{Wi} &= S_i * E = 2^{i+1} * \frac{N}{Blk_H}. \end{aligned} \tag{5.15}$$

У поређењу са једначином 5.14, када је приступ поравнат могуће је приступити два пута ширем блоку и истом броју линија. Притом је укупан број пиксела блока једнак $2 * N$. За посматрани случај $N = 16$ и $Blk_H = 4$, подржани су приступи поравнатом блоку од 8∗4, 16∗2 и 32∗1 пиксела подешавањем S_i на вредности два, четири и осам, респективно.

Уз блоковске начине приступа, свака вредност S_i омогућава непоравнати и поравнати приступ реду пиксела, чије се максималне димензије добијају заменом израза за B и E из једначина 5.9 и 5.10 у једначину 5.5. Максималне димензије приступа непоравнатом и поравнатом реду Row_W и Row^a_W, респективно, износе:

$$\begin{aligned} Row_W &= 1 + (2 * Blk_H - 1) * \frac{N}{Blk_H}, \\ Row^a_W &= 2 * N. \end{aligned} \tag{5.16}$$

За посматрани пример $N = 16$ и $Blk_H = 4$ вредности Row_W и Row^a_W су 29 и 32, респективно. На основу израза 5.16 може се приметити да максималне димензије реда не зависе од подешавања S_i, већ само од параметара одређених у време креирања меморијског подсистема. Другим речима, без обзира на начин приступа блоку, који је изабран у време развоја или извршавања програма, подржане димензије приступа реду су константне. Како се приступ реду углавном користи за преносе података из меморије ван чипа, ово даље значи да су преноси независни од избора блоковског начина приступа.

Табела 5.1 илуструје више комбинација вредности улазних параметара задатих у време креирања подсистема, параметре који се рачунају на основу улазних параметара, као и подржане начине приступа и параметар S_i који се подешава у време развоја или извршавања програма. Свака група од три суседне комбинације разликује се само по капацитету подсистема, што не утиче на архитектуру и подржане начине приступа. Приказ начина приступа у облику 25∗1–16∗1 означава да су подржане све ширине блока или реда од 25 до 16.

Табела 5.1 – Вредности параметара подсистема описаног у овом раду и подржани начини приступа за низ репрезентативних случајева.

Улазни параметри који се задају у време креирања подсистема				Параметри који се рачунају у време креирања подсистема				Параметри који се подешавају у време развоја или извршавања програма и подржани начини приступа		
N (пиксела)	Blk_H (линија)	C (речи)	Бита по пикселу	B (банака)	E (пиксела)	W (речи)	Капацитет (килобајта)	S_i (банака)	Непоравнати начини приступа	Поравнати начини приступа
16	2	1024	8	4	8	512	16	2, 4	9∗2, 8∗2, 25∗1–16∗1	16∗2, 32∗1
		2048				1024	32			
		4096				2048	64			
	4	1024		8	4	512	16	2, 4, 8	5∗4, 4∗4, 13∗2–8∗2, 29∗1–16∗1	8∗4, 16∗2, 32∗1
		2048				1024	32			
		4096				2048	64			
	8	1024		16	2	512	16	2, 4, 8, 16	3∗8, 2∗8, 7∗4–4∗4, 15∗2–8∗2, 31∗1–16∗1	4∗8, 8∗4, 16∗2, 32∗1
		2048				1024	32			
		4096				2048	64			
32	2	1024	8	4	16	512	32	2, 4	17∗2, 16∗2, 49∗1–32∗1	32∗2, 64∗1
		2048				1024	64			
		4096				2048	128			
	4	1024		8	8	512	32	2, 4, 8	9∗4, 8∗4, 25∗2–16∗2, 57∗1–32∗1	16∗4, 32∗2, 64∗1
		2048				1024	64			
		4096				2048	128			
	8	1024		16	4	512	32	2, 4, 8, 16	5∗8, 4∗8, 13∗4–8∗4, 29∗2–16∗2, 61∗1–32∗1	8∗8, 16∗4, 32∗2, 64∗1
		2048				1024	64			
		4096				2048	128			
64	4	1024	8	8	16	512	64	2, 4, 8	17∗4, 16∗4, 49∗2–32∗2, 113∗1–64∗1	32∗4, 64∗2, 128∗1
		2048				1024	128			
		4096				2048	256			
	8	1024		16	8	512	64	2, 4, 8, 16	9∗8, 8∗8, 25∗4–16∗4, 57∗4–32∗2, 121∗1–64∗1	16∗8, 32∗4, 64∗2, 128∗1
		2048				1024	128			
		4096				2048	256			
	16	1024		32	4	512	64	2,4,8,16,32	5∗16, 4∗16, 13∗8–8∗8, 29∗4–16∗4, 61∗2–32∗2, 125∗1–64∗1	8∗16, 16∗8, 32∗4, 64∗2, 128∗1
		2048				1024	128			
		4096				2048	256			

Закључак ове секције јесте да свака вредност S_i омогућава непоравнати и поравнати приступ блоку у више димензија, као и непоравнати и поравнати приступ реду пиксела, који се могу конкурентно реализовати у време извршавања програма.

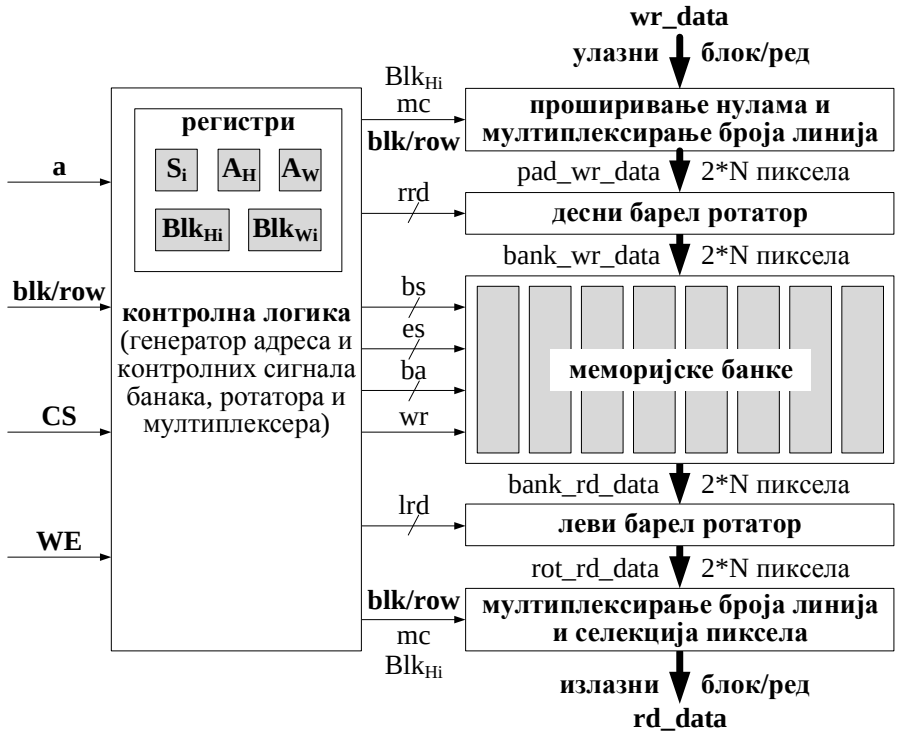
5.4 Управљачка логика

Поред одговарајућег скупа меморијских банака, архитектура паралеленог меморијског подсистема садржи и управљачку логику која се налази око меморијских банака и управља њиховим радом. Основни задаци управљачке логике јесу да при упису распореди пикселе у банке у складу са искошеним обрасцем, а да при читању из подсистема изврши инверзно распоређивање и на излазу обезбеди формат који се користи на путањи података процесора. На тај начин се омогућава једноставно приступање подацима у подсистему без потребе за познавањем обрасца распоређивања пиксела и архитектуре банака унутар подсистема. Слика 5.5 приказује микроархитектуру управљачке логике развијену за меморијски подсистем предложен у овом раду.

Управљачка логика заснива се на два барел ротатора и два мултиплексера који померају улазне или излазне пикселе меморијских банака удесно или улево, у складу са искошеним обрасцем распоређивања. При упису у меморијски подсистем се, помоћу улазног мултиплексера и десног ротатора, пиксели свих линија блока постављају у жељени распоред који одговара искошеном обрасцу. Излазни леви ротатор и мултиплексер врше инверзну операцију при читању из меморијског подсистема, односно пикселе прочитане из меморијских банака, који су рапоређени по искошеном обрасцу, постављају у редослед по врстама, што одговара формату који користи векторска путања података процесора. Притом су дистанце ротирања rrd и lrd одређене тако да се крајњи леви пиксел блока постави на жељену позицију. Управљачки сигнали mc, којима се контролише рад мултиплексера, зависе од броја линија блока коме се приступа. Проширивање нулама[5] односно селекција пиксела[6] врше се при упису односно читању, респективно, јер је укупна ширина низа меморијских банака $B * E = 2 * N$ већа или једнака броју пиксела улазног односно излазног блока или реда.

[5] енг. zero padding

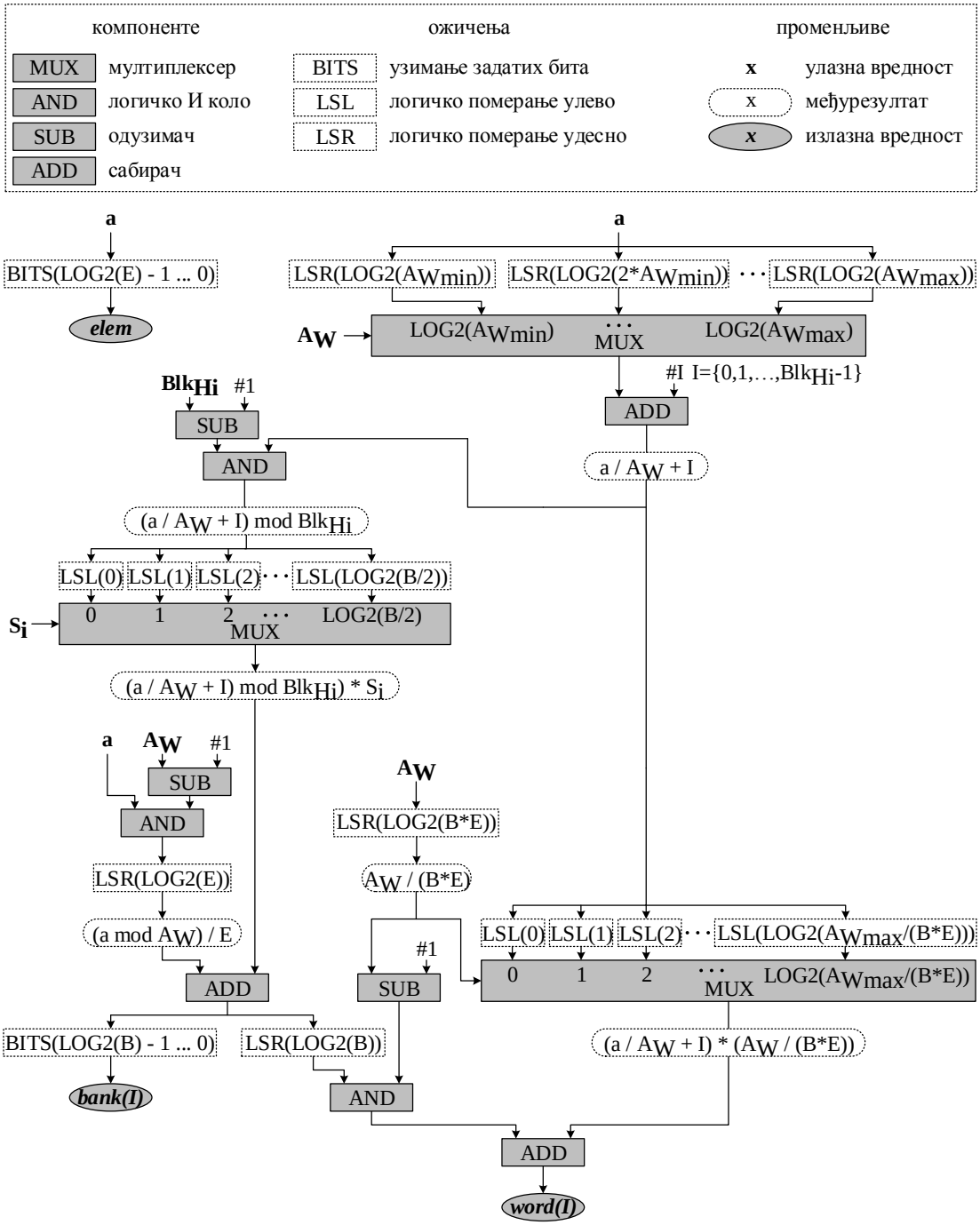
[6] енг. pixel selection



Слика 5.5 – Микроархитектура управљачке логике унутар паралелног меморијског подсистема.

Меморијске банке селектују се помоћу B bs управљачких сигнала. Притом, како би потрошња енергије била минимална, селектују се само оне банке у које је потребно уписати или из којих је потребно прочитати један или више пиксела у оквиру реализације приступа. Низ управљачких сигнала означених са es селектује пикселе у оквиру адресиране речи сваке банке који ће бити прочитани или уписани. На тај начин омогућавају се непоравнати уписи, где је потребно уписати вредности у одређене локације унутар једне речи банке, а без преписивања садржаја других локација у истој речи. Низ управљачких сигнала ba представља адресе меморијских речи унутар банака, а сигнал wr одређује да ли се врши упис или читање.

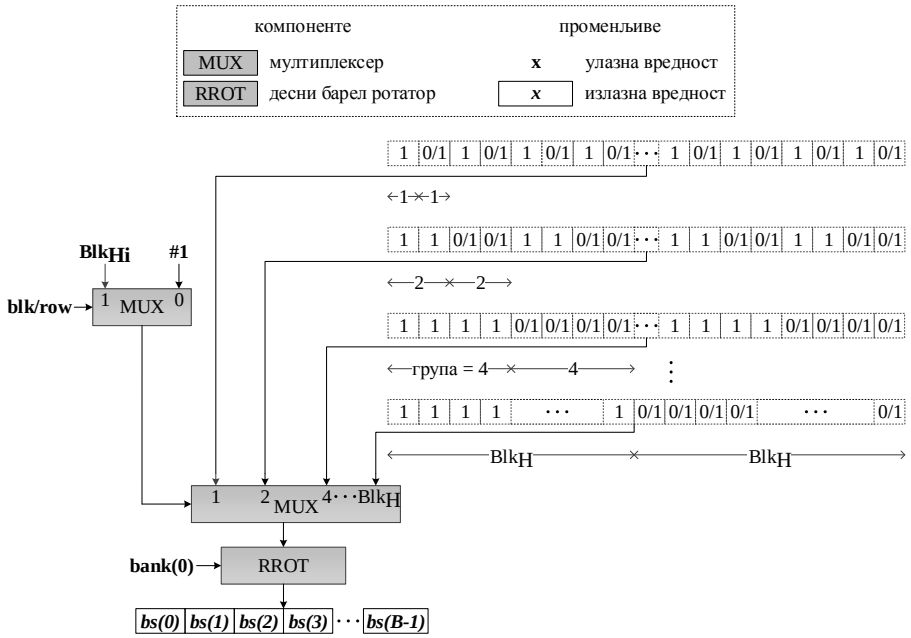
Вредности адреса банака и контролних сигнала, односно излаза блока „контролна логика", генеришу се на основу једначина 5.6, вредности S_i, ширине блока Blk_{Hi} и ширине дводимензионалног низа A_W уписаних у истоимене регистре, затим сигналом blk/row којим се споља врши избор између приступа блоку и реду, CS сигнала којим се споља селектује подсистем и сигнала WE којим се споља врши избор између уписа и читања у подсистем. Сигнал a представља адресу којој се приступа у подсистему. Као што је раније објашњено,



Слика 5.6 – Микроархитектура логике која израчунава вредности *bank*, *word* и *elem* дате једначинама 5.6.

адреса одговара померају горњег левог пиксела блока или реда коме се приступа у односу на горњи леви пиксел дводимензионалног низа. Померај се рачуна у поретку по врстама и изражава у пикселима.

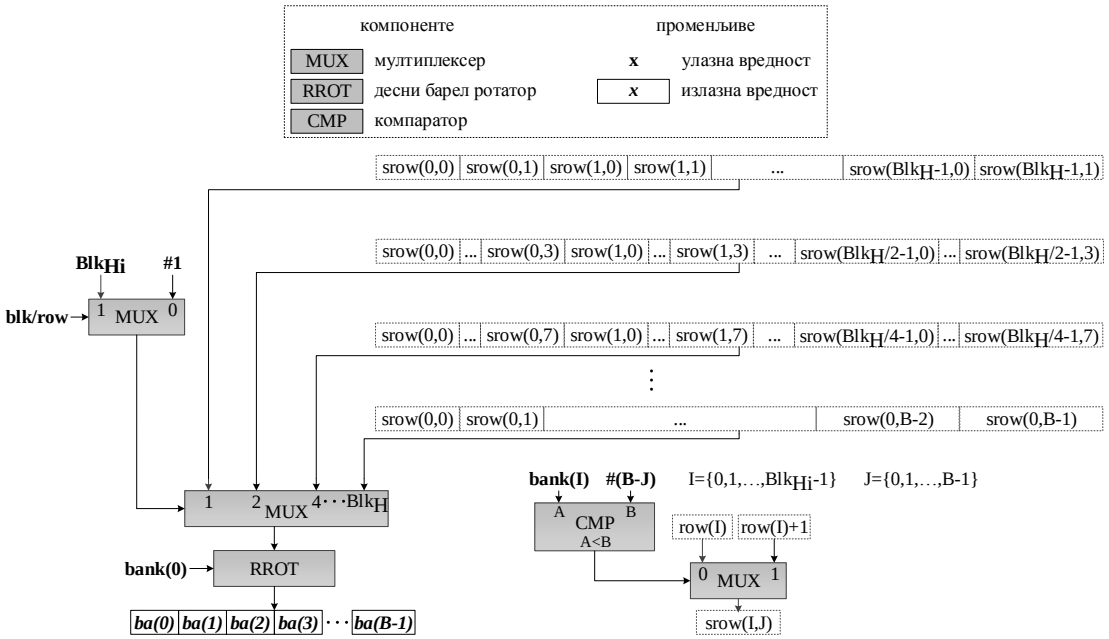
Микроархитектура логике којом се реализују једначине 5.6, у оквиру блока „контролна логика”, приказана је на слици 5.6. Приказана логика одређује вред-



Слика 5.7 – Микроархитектура логике која генерише селекционе сигнале *bs* меморијских банака.

ности $bank(I)$ и $word(I)$ за линију I блока коме се приступа, док је вредност $elem$ иста за све линије блока. За израчунавање вредности $bank$ и $word$ за сваку линију блока потребно је Blk_H инстанци приказане логике где се параметар I замењује вредностима из скупа $\{0, 1, 2, ...Blk_H - 1\}$. Са слике се може приметити да су коришћене једноставне компоненте попут сабирача, одузимача и кола логичке конјункције, као и нешто комплекснији мултиплексери чији број улаза зависи од минималне и максималне ширине дводимензионалног низа које подсистем подржава, у ознаци A_{Wmin} и A_{Wmax} респективно.

Слика 5.7 приказује микроархитектуру логике којом се одређује B селекционих сигнала меморијских банака, у ознаци bs. Најважније компоненте јесу мултиплексер са Blk_H улаза ширине B бита и десни барел ротатор ширине такође B бита и са B позиција ротирања. Мултиплексер бира један од претходно генерисаних образаца селекционих сигнала у зависности од броја линија блока коме се приступа, а ротатор позиционира $bs(0)$ сигнал на банку са редним бројем $bank(0)$ у којој се налази горњи леви пиксел блока. Мултиплексер са два улаза прослеђује Blk_{Hi} или 1 као селекциони сигнал првог мултиплексера у зависности од тога да ли се захтева приступ блоку или реду сигналом blk/row. Као што је већ наведено, у циљу минималне потрошње енергије свака од банака селектује се само онда када је неопходно, што зависи од начина приступа,

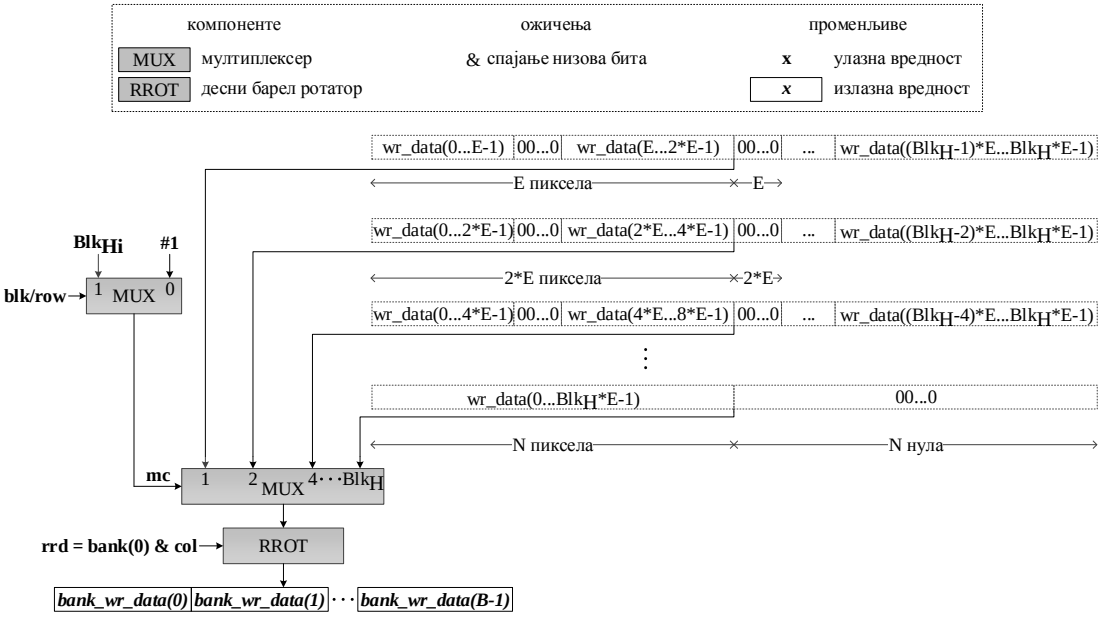


Слика 5.8 – Микроархитектура логике која генерише адресе речи *ba* меморијских банака.

поравнања адресе и броја линија и пиксела блока коме се приступа. Слика приказује селекцију банака у зависности од броја линија блока, док су мултиплексери који одређују селекционе сигнале у зависности од поравнања и броја пиксела изостављени ради једноставности илустрације. Притом су одговарајуће позиције у обрасцима селекционих сигнала које зависе од поравнања и броја пиксела означене са 0/1.

Генерисање сигнала *es* врши се на готово идентичан начин мултиплексирањем у зависности од броја линија блока коме се приступа и ротирањем удесно у зависности од позиције горњег левог пиксела блока. Разлика у односу на генерисање *bs* сигнала јесте у томе што су унапред генерисани обрасци селекционих сигнала *es* ширине $B * E$ бита и садрже Blk_{Hi} група од по Blk_{Wi} јединица, а ротирање удесно врши се за $bank(0) * E + elem$ позиција. Стога су мултиплексер и ротатор ширине $B * E$ бита и са B позиција ротирања.

На слици 5.8 приказана је микроархитектура логике која одређује B адреса меморијских банака, означених са *ba*. Мултиплексирање у зависности од броја линија блока коме се приступа и ротирање удесно у зависности од редног броја банке $bank(0)$ врши се на исти начин као и при генерисању сигнала *bs*. Специфичност ове логике јесте у томе што је потребно одредити исправне адресе банака на којима се линија дводимензионалног низа прелама услед примене

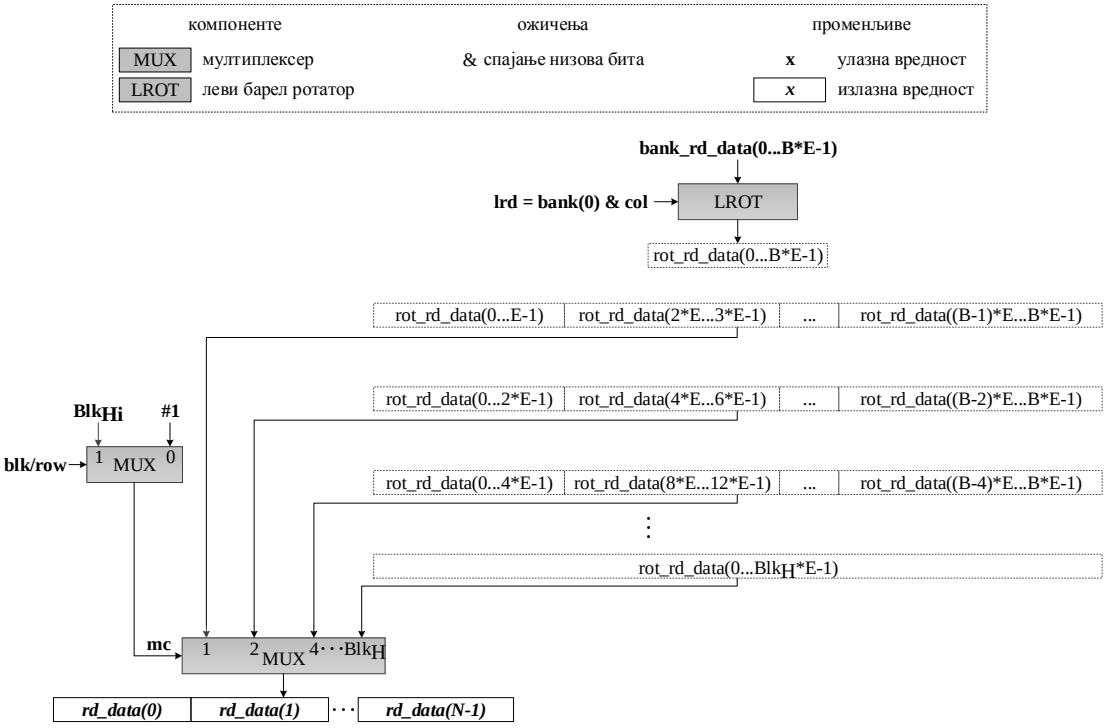


Слика 5.9 – Микроархитектура логике која проширује нулама, мултиплексира и ротира удесно пикселе за упис у меморијске банке.

искошеног обрасца распоређивања. У том случају се одговарајући пиксели налазе на адреси $row + 1$ уместо row. Како би се одредила исправна адреса сваке банке J, за сваку линију I блока потребан је низ парова компаратора и мултиплексера који врше селекцију између вредности $row(I)$ и $row(I) + 1$. Резултат ове селекције су вредности $srow(I, J)$ које се потом мултиплексирају и ротирају удесно на претходно описан начин.

Припрема података за упис у меморијске банке врши се помоћу логике приказане на слици 5.9. Улазни блок или ред означен са wr_data који се жели уписати у подсистем и који је у формату путање података, прво се проширује убацивањем нула на начин који зависи од броја линија блока, а потом ротира удесно за $bank(0) * E + elem$ пиксела како би се горњи леви пиксел блока или реда поставио на одговарајућу позицију. Добијени низ од $B * E$ пиксела $bank\_wr\_data$ шаље се на улазе меморијских банака, тако да се $bank\_wr\_data(I)$ уписује у банку са редним бројем I.

Подаци прочитани из меморијских банака припремају се за слање на путању података на начин који је приказан на слици 5.10. Прочитани низ од $B * E$ пиксела означен са $bank\_rd\_data$ прво се ротира улево како би се горњи леви пиксел блока или реда коме се приступа поставио на нулту позицију у низу rot_rd_data. Потом се из ротираног низа селектују потребни пиксели у зависности од броја линија блока или реда коме се приступа и у складу са



Слика 5.10 – Микроархитектура логике која ротира улево, мултиплексира и селектује пикселе прочитане из меморијских банака.

изабраним начином приступа. Тако добијени низ пиксела rd_data шаље се назад на путању података као резултат читања из меморијског подсистема. Ради једноставности илустрације слика приказује случај када је подржан приступ блоку или реду од N пиксела. У случајевима када су подржани додатни начини приступа излазни мултиплексер садржи више улаза на које су доведени низови пиксела селектовани у складу са жељеним начинима приступа.

Приказана микроархитектура меморијског подсистема потпуно је проточна, што значи да се нови приступ подсистему гарантовано може започети у сваком циклусу такта и да ће резултат читања гарантовано бити доступан после константног броја циклуса такта, а који одговара укупном кашњењу подсистема[7]. Потпуна проточност је неопходна како би се могла гарантовати висока пропусност подсистема, а самим тим и висока пропусност обраде на процесору.

[7] енг. memory subsystem latency

5.5 Паралелни приступ табели пресликавања

Ова секција дефинише проширење претходно описане архитектуре које се односи на реализацију паралелног читања из табеле пресликавања, као још једне потребне врсте начина приступа паралелном меморијском подсистему.

Овакав начин приступа подсистему познат је у литератури, а најновија реализација, нама позната у време писања овог рада, описана је у патенту [133]. Ова реализација заснива се на N-банковној архитектури, односно употреби N меморијских банака и смештању N копија табеле пресликавања у банке, чиме се омогућава паралелно читање N вредности из табеле – по једна вредност из сваке банке. Предност овакве архитектуре јесте што омогућава максималну пропусност приступа табели пресликавања, односно захтева само један приступ меморијском подсистему за пресликавање вектора од N вредности. Са друге стране, цена овакве пропусности јесте постојање N копија табеле, односно N пута веће заузеће меморије у односу на величину табеле. Притом, до сада објављене реализације не предлажу компромис између пропусности и цене приступа табели пресликавања.

Да би се омогућио такав компромис, овај рад предлаже реализацију варијанте паралелног приступа табели пресликавања која је описана у секцији 3.3.3, односно паралелно читање L вредности из табеле пресликавања, где се L може изабрати у време креирања подсистема или у време развоја или извршавања програма. Како се предложена реализација заснива на архитектури банака описаној у секцији 5.2, важи да је $1 \leq L \leq B$ и $L = 2^i, i \in \{0, log_2(B)\}$. Уз овакав начин приступа потребно је $N : L$ приступа за пресликавање вектора од N вредности, при чему је цена постојање L копија табеле у подсистему.

Оваква реализација паралелног приступа табели пресликавања представља један од иновативних доприноса овог рада, а детаљи су дати у наставку. Секције 5.5.1 и 5.5.2 описују праволинијски образац и образац распоређивања са груписањем банака, респективно, који се користе за реализацију паралелног приступа табели пресликавања.

5.5.1 Праволинијски образац распоређивања

Свака меморијска банка из раније предложеног скупа од B банака, дефинисаног једначинама 5.9, 5.10 и 5.11, може се адресирати независно од других банака. Стога, B банака омогућава паралелно пресликавање $L = B$ вредности.

T(0)	T(1)	T(2)	T(3)	T(4)	T(5)	T(6)	T(7)	T(8)	T(9)	T(10)	T(11)	T(12)	T(13)	T(14)	T(15)

(а)

	банка 0		банка 1		банка 2		банка 3		банка 4		банка 5		банка 6		банка 7	
	E_0	E_1	E_0	E_1	E_0	E_1	E_0	E_1	E_0	E_1	E_0	E_1	E_0	E_1	E_0	E_1
W_0	T(0)	T(1)	T(0)	T(1)	T(0)	T(1)	T(0)	T(1)	T(0)	T(1)	T(0)	T(1)	T(0)	T(1)	T(0)	T(1)
W_1	T(2)	T(3)	T(2)	T(3)	T(2)	T(3)	T(2)	T(3)	T(2)	T(3)	T(2)	T(3)	T(2)	T(3)	T(2)	T(3)
W_2	T(4)	T(5)	T(4)	T(5)	T(4)	T(5)	T(4)	T(5)	T(4)	T(5)	T(4)	T(5)	T(4)	T(5)	T(4)	T(5)
W_3	T(6)	T(7)	T(6)	T(7)	T(6)	T(7)	T(6)	T(7)	T(6)	T(7)	T(6)	T(7)	T(6)	T(7)	T(6)	T(7)
W_4	T(8)	T(9)	T(8)	T(9)	T(8)	T(9)	T(8)	T(9)	T(8)	T(9)	T(8)	T(9)	T(8)	T(9)	T(8)	T(9)
W_5	T(10)	T(11)	T(10)	T(11)	T(10)	T(11)	T(10)	T(11)	T(10)	T(11)	T(10)	T(11)	T(10)	T(11)	T(10)	T(11)
W_6	T(12)	T(13)	T(12)	T(13)	T(12)	T(13)	T(12)	T(13)	T(12)	T(13)	T(12)	T(13)	T(12)	T(13)	T(12)	T(13)
W_7	T(14)	T(15)	T(14)	T(15)	T(14)	T(15)	T(14)	T(15)	T(14)	T(15)	T(14)	T(15)	T(14)	T(15)	T(14)	T(15)

(б)

Слика 5.11 – (а) Пример табеле пресликавања. (б) Праволинијски образац распоређивања вредности из дате табеле пресликавања у меморијске банке.

За реализацију паралелног читања из табеле пресликавања потребно је да се у банке смести B копија табеле. Слика 5.11 приказује праволинијски образац распоређивања, где су вредности једне копије табеле распоређене у једну банку, а $T(i)$ означава вредност на адреси i у табели. Према праволинијском обрасцу, вредности из табеле распоређене су у банкама у поретку по врстама, где једна реч банке садржи E узастопних вредности из табеле. Локација вредности $T(i)$ у свакој банци дефинисана је следећим изразима:

$$word = \left\lfloor \frac{i}{E} \right\rfloor, \qquad elem = i \bmod E. \tag{5.17}$$

Уз овакав образац распоређивања и дати скуп меморијских банака постиже се максимална пропусност пресликавања од $B : N$ вектора у једном циклусу такта. Притом је и цена, у смислу заузећа меморије, максимална, јер се свака вредности из табеле смешта у сваку банку. Стога је максимална величина табеле пресликавања, која се може сместити у меморијски подсистем, B пута мања од укупног капацитета подсистема и износи $W * E$ вредности.

5.5.2 Образац распоређивања са груписањем банака

Како би се омогућио компромис између пропусности пресликавања и цене у смислу заузећа меморије, предлаже се образац распоређивања са груписањем меморијских банака, који је илустрован на слици 5.12. Овај образац подразумева постојање L копија табеле у подсистему и стога омогућава паралелно

Слика 5.12 – Образац за распоређивање вредности из дате табеле пресликавања у меморијске банке са груписањем банака.

читање L вредности из табеле. Избором вредности L у време креирања меморијског подсистема, развоја или извршавања програма, може се одредити поменути компромис.

По овом обрасцу, B банака дели се у L група од $B : L$ банака по групи. Свака група посматра се логички као једна меморијска банка од W речи и $E * B : L$ вредности по једној речи и садржи једну копију табеле. У оквиру једне групе банака, вредности из табеле смештају се у поретку по врстама, аналогно праволинијском обрасцу.

По обрасцу распоређивања са груписањем банака, локација вредности $T(i)$ у свакој групи банака дефинисана је следећим једначинама:

$$word = \left\lfloor \frac{i}{E * \frac{B}{L}} \right\rfloor, \qquad elem = i \bmod E. \tag{5.18}$$

У овом случају пропусност пресликавања јесте $L : N$ вектора у једном циклусу такта, а максимална величина табеле пресликавања, која се може сместити у меморијски подсистем, износи $W * E * B : L$ вредности. На основу ових израза може се закључити да избор вредности L одређује компромис између пропусности пресликавања и заузећа меморије.

За упис вредности табеле пресликавања у меморијски подсистем предлаже се коришћење начина приступа реду. Притом, сваки ред је потребно уписати два пута јер је $B = 2 * N$, а сваки уписани ред садржи $E * B : L$ суседних вредности из табеле, поновљених $N : (E * B : L)$ пута, као што илуструје слика 5.13 за случај $L = 4$, $B = 8$, $E = 2$ и $N = 8$.

Слика 5.13 – Упис вредности дате табеле пресликавања у банке меморијског подсистема коришћењем начина приступа реду.

За подршку паралелном читању из табеле пресликавања, потребно је унапредити управљачку логику приказану на слици 5.5. То подразумева измену блока „контролна логика", у смислу додавања новог регистра L, додавања улазног сигнала којим се захтева извршавање приступа табели пресликавања уместо приступа блоку или реду, као и исправног генерисања контролних сигнала за овај начин приступа. Поред тога, потребно је унапредити блок „мултиплексирање броја линија и селекција пиксела", како би се подржао посматрани начин приступа. Измена других блокова управљачке логике није потребна.

5.6 Анализа цене архитектуре

У овој секцији анализира се цена, односно трошкови описане архитектуре, у смислу заузећа површине на чипу и потрошње енергије, а потом се врши поређење са ценама других архитектура које представљају тренутно највиши степен развоја у овој области.

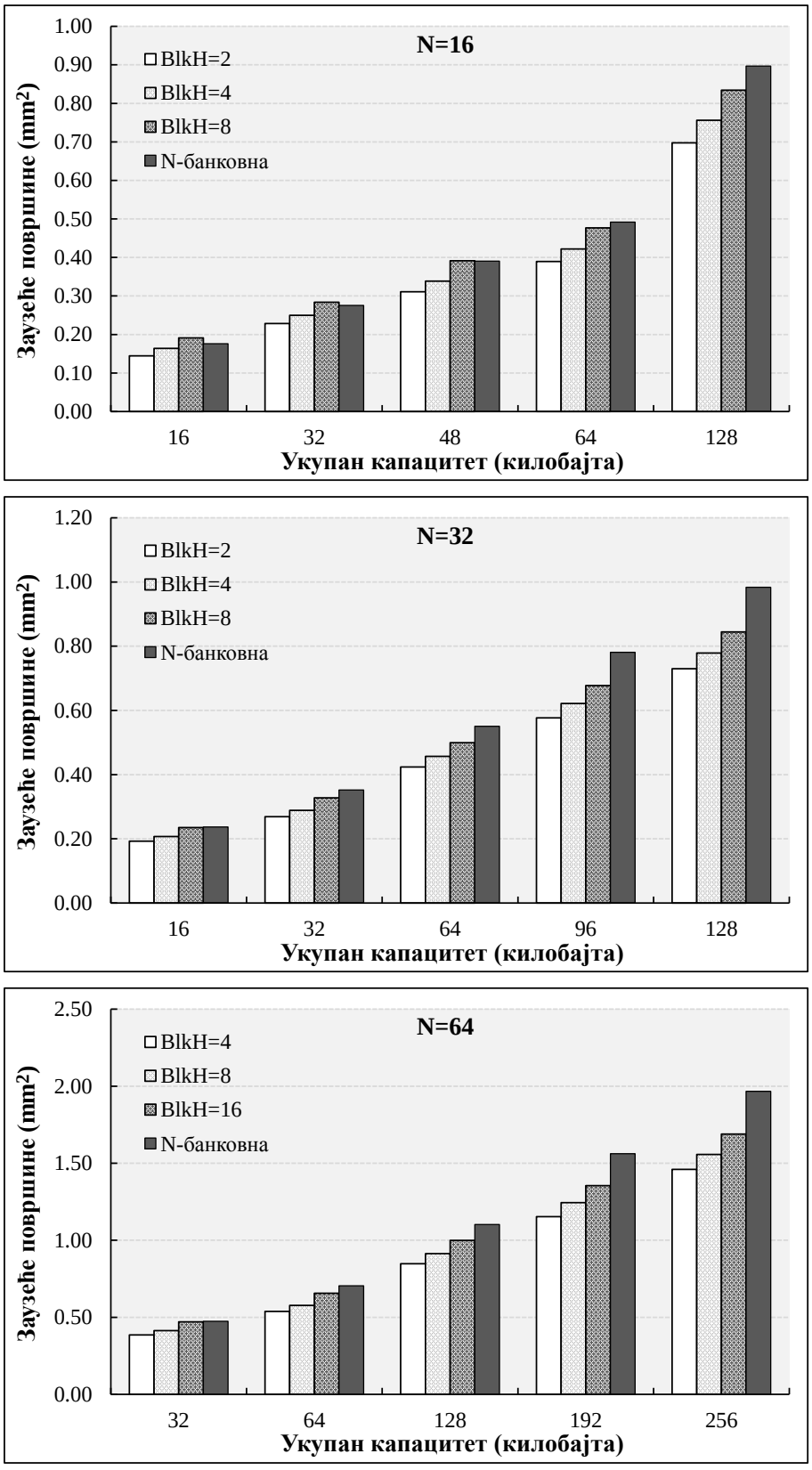
Цена паралелног меморијског подсистема укључује цену меморијских банака и цену управљачке логике унутар подсистема. У овој секцији посматра се само цена меморијских банака, јер је она многоструко већа у односу на цену логике и чини највећи део укупне цене, као што је експериментално показано у радовима из литературе [23,26,27,32,33,138], као и у оквиру студије случаја у поглављу 6. У складу са тим, укупно заузеће површине меморијског подсистема

одговара суми површина B меморијских банака и одређено је у време креирања подсистема. Слично томе, утрошена енергија за један приступ једнака је суми поторошњи свих банака селектованих при реализацији тог приступа, а укупна потрошња енергије једнака је суми потрошњи свих реализованих приступа. Притом, како број селектованих банака зависи од поравнања приступа и коришћеног начина приступа, односно димензија блока или реда, тако је и потрошња одређена у време извршавања програма. У овој секцији подразумева се приступ максималном броју пиксела, што захтева селектовање свих меморијских банака и резултује максималном потрошњом енергије по једном приступу, која је једнака суми потрошњи свих B банака.

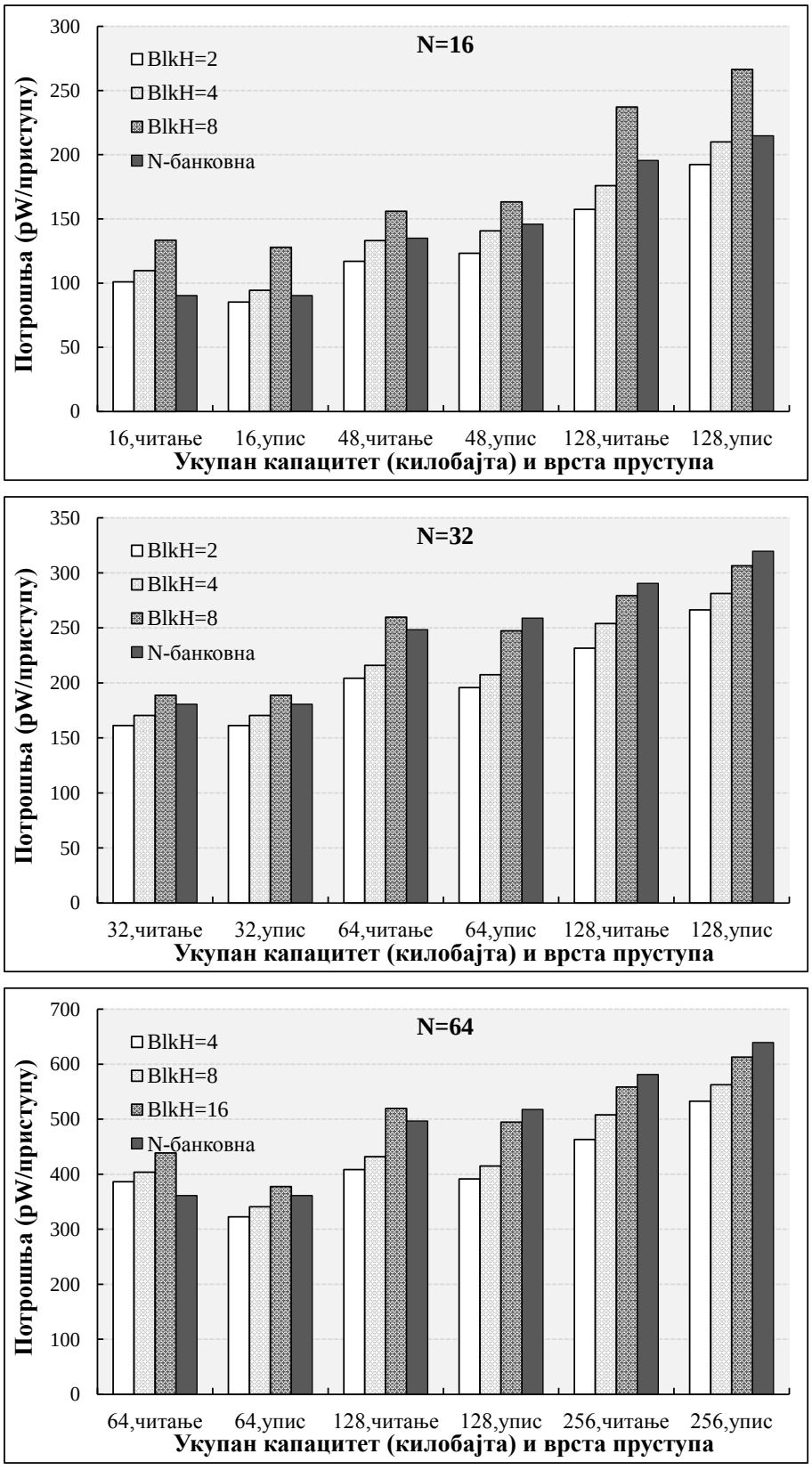
Слике 5.14 и 5.15 приказују површину и максималну потрошњу по једном приступу, респективно, за репрезентативне вредности параметара N и Blk_H, као и за више случајева укупног капацитета меморијског подсистема израженог у килобајтима. Претпостављена је ширина једног пиксела од осам бита, што дефинише ширину речи меморијске банке у битима. За прорачун цена коришћене су SRAM банке које омогућавају приступ једној меморијској речи у једном циклусу такта, реализоване у шездесет пет нанометарској CMOS технологији и оптимизоване за ниску потрошњу. Иако се анализа врши само за шездесет пет нанометарску технологију, сличан релативан однос цена очекује се и за друге CMOS технологије, јер технологија не утиче на укупан број меморијских ћелија који већински одређује цену. Поређења ради, поред цена описане архитектуре графикони на сликама 5.14 и 5.15 приказују и цене N-банковне архитектуре. Подсећања ради, N-банковна архитектура саджи N банака чије су речи ширине једног пиксела од осам бита и двоструко већи број речи у односу на архитектуру овог рада.

Са слика 5.14 и 5.15 може се закључити да површина описане архитектуре равномерно расте са повећањем максималне подржане висине блока Blk_H, укупног капацитета и параметра N. Притом је заузеће површине описане архитектуре нешто мање у односу на N-банковну архитектуру у готово свим илустрованим случајевима. Изузетак представљају случајеви високих вредности Blk_H, на пример осам, и малог укупног капацитета у случају $N = 16$, као што су шеснаест и тридесет два килобајта. У ова два случаја, заузеће површине описане архитектуре веће је за три до осам процената.

Максимална потрошња енергије по једном приступу такође равномерно расте са повећањем вредности параметара подсистема. У највећем броју случа-



Слика 5.14 – Заузеће површине меморијских банака, дато за више варијанти описане архитектуре, као и за N-банковну архитектуру.



Слика 5.15 – Максимална потрошња енергије меморијских банака, дата за више варијанти описане архитектуре, као и за N-банковну архитектуру.

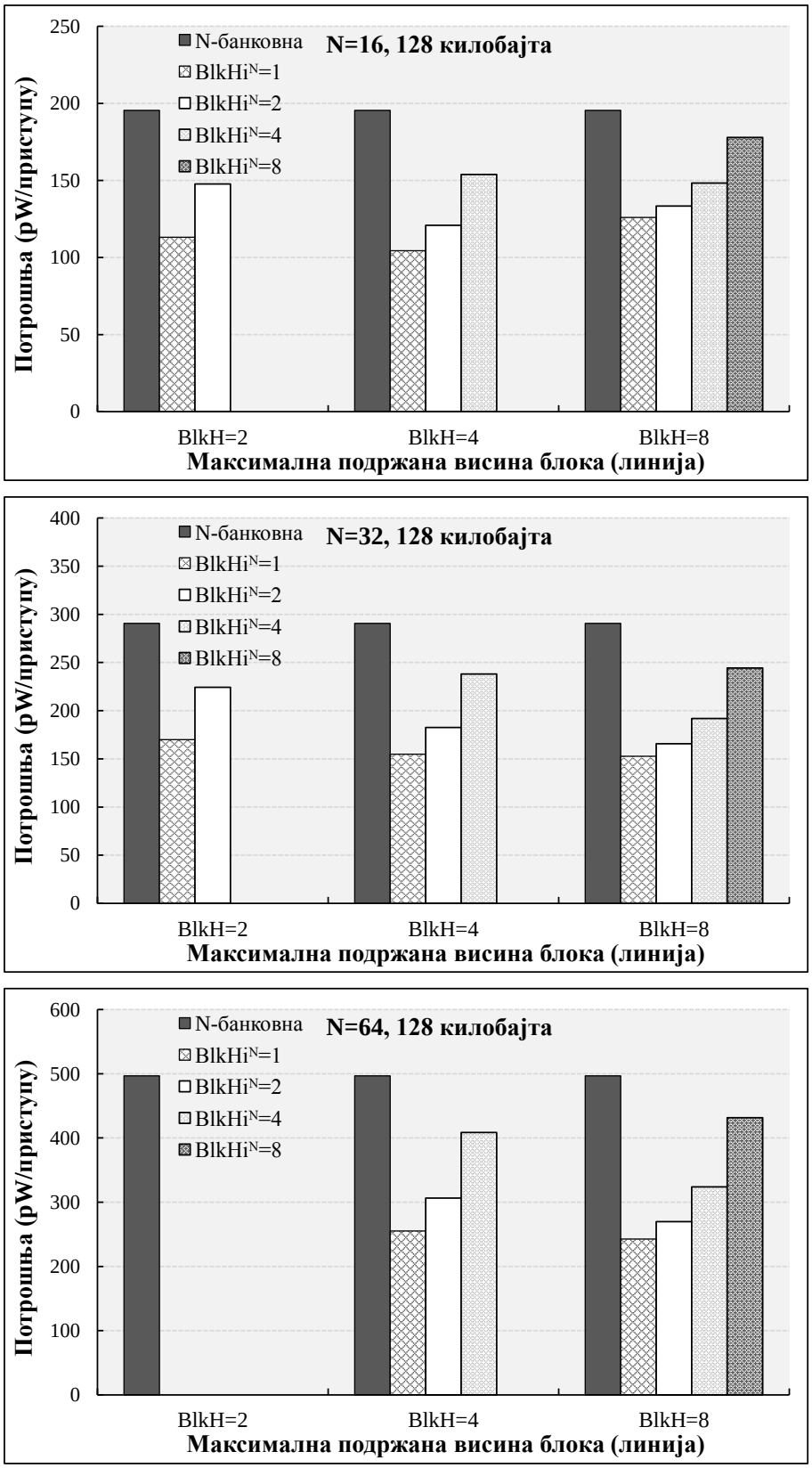
јева потрошња је мало мања од потрошње N-банковне архитектуре. Изузетак су поново случајеви високе вредности Blk_H и малог укупног капацитета од шеснаест килобајта. У овим случајевима, потрошња по једном приступу описане архитектура већа је за пет до двадесет пет процената. Без обзира на то, укупна потрошња у методама обраде, као што је суб-пиксел упаривање блокова, значајно је нижа захваљујући новим начинима приступа и вишеструко мањем укупном броју приступа, што је експериментално показано у поглављу 6.

Када се приступа блоковима и редовима од N пиксела, у просеку се селектује мање од B банака по једном приступу. Стога је и просечна потрошња мања од максималне, под претпоставком да банка троши енергију само када је селектована, а да се у супротном потрошња енергије може занемарити. У случају поравнатих приступа, број банака који се селектује по једном приступу је $(Blk^N_{Wi} : E) * Blk^N_{Hi}$. У случају неповравнатих приступа, селектује се по једна додатна меморијска банка за сваку линију блока или реда. Укупан број селектованих банака у том случају је $(Blk^N_{Wi} : E + 1) * Blk^N_{Hi}$. Под претпоставком да се током обраде врши велики број приступа и да је расподела адреса којима се приступа равномерна, просечно ће се извршити $1 : E * 100$ процената поравнатих и $(E - 1) : E * 100$ процената непоравнатих приступа. На основу тога и једначина 5.14, просечан број селектованих банака по једном приступу је:

$$\begin{aligned} B^a_i &= \frac{1}{E} * \frac{Blk^N_{Wi}}{E} * Blk^N_{Hi} + \frac{E-1}{E} * \left(\frac{Blk^N_{Wi}}{E} + 1\right) * Blk^N_{Hi} \\ &= Blk_H + \left(1 - \frac{Blk_H}{N}\right) * Blk^N_{Hi}. \end{aligned} \tag{5.19}$$

Множењем B^a_i са потрошњом једне банке, која је константна у време извршавања програма, односно одређена је у време креирања меморијског подсистема, добија се просечна потрошња по једном приступу од N пиксела. Узимајући то и једначину 5.19 у обзир, може се закључити да је просечна потрошња делимично одређена у време креирања подсистема, параметрима Blk_H и N, а делимично вредношћу Blk^N_{Hi} у време извршавања програма. Како је потрошња директно пропорционална вредности Blk^N_{Hi}, може се рећи да описана архитектура нуди компромис између броја линија блока и потрошње енергије у случају приступа од N пиксела.

Слика 5.16 приказује просечну потрошњу по једном приступу за репрезентативне вредности N и Blk_H и различите вредности Blk^N_{Hi} дефинисане једначи-



Слика 5.16 – Потрошња енергије меморијских банака при реализовању приступа од N пиксела, дата за више варијанти описане архитектуре, као и за N-банковну архитектуру.

ном 5.14. Поређења ради, на графиконима је приказана и просечна потрошња N-банковне архитектуре. У свим приказаним случајевима описана архитектура троши мање енергије него N-банковна архитектура и стога је ефикаснија у применама које користе приступе блоковима и редовима од N пиксела.

6

ЕКСПЕРИМЕНТАЛНА СТУДИЈА СЛУЧАЈА

Ово поглавље представља резултате студије случаја којом је реализована естимација кретања методом суб-пиксел упаривања блокова на паралелном меморијском подсистему предложеном у овом раду и одговарајућем векторском процесору. Циљ студије случаја јесте демонстрација предности предложеног подсистема на практичном примеру, у поређењу са шест изабраних паралелних меморијских подсистема који представљају највиши степен развоја у време истраживања представљеног у овом раду [23, 26, 33–36].

Поглавље је организовано на следећи начин. Секција 6.1 дефинише карактеристике реализоване методе суб-пиксел упаривања блокова. Особине свих седам паралелних меморијских подсистема који се упоређују дате су у секцији 6.2. Теоријска анализа броја меморијских приступа потребних за реализацију методе упаривања блокова на векторској путањи података дата је у секцији 6.3. На тај начин ова секција одређује максималну пропусност обраде која се може постићи са сваким од наведених меморијских подсистема. Експериментални резултати засновани на практичној реализацији упаривања блокова дати су у секцији 6.4. У истој секцији дата је и анализа добијених резултата.

Табела 6.1 – Карактеристике коришћене 3DRS методе суб-пиксел упаривања блокова.

Број референтних фрејмова	1
Величина циљног блока	8∗8 (пиксела∗линија)
Мера сличности блокова	сума апсолутних разлика
Број кандидат вектора по циљном блоку	7
Прецизност суб-пиксел кандидат вектора	четвртина пиксела
Интерполација референтих блокова	билинеарна
Резолуција улазног видеа	3840∗2160 (пиксела∗линија)
Величина области претраге	224∗96 (пиксела∗линија)

6.1 Коришћени метод упаривања блокова

У овој студији случаја реализује се добро позната 3DRS[1] метода суб-пиксел упаривања блокова [40]. Најважније карактеристике ове методе дате су у табели 6.1. Метода упаривања блокова користи укупно два фрејма, тренутни и референтни. Скуп кандидат вектора садржи седам вектора прецизности од једне четвртине пиксела. Величина циљних и референтих блокова јесте 8∗8 пиксела, а коришћена мера сличности два блока је сума апсолутних разлика (SAD). Референтни блок се добија билинеарном интерполацијом на основу пиксела из референтног фрејма. Кандидат вектори су ограничени унутар области претраге величине 224∗96 пиксела, са центром у циљном блоку. Резолуција улазног видеа је 3840∗2160 пиксела.

6.2 Упоређени меморијски подсистеми

Ова секција наводи и табеларно приказује најважније особине свих седам паралелних меморијских подсистема који се пореде у студији случаја.

Метода упаривања блокова описана у претходној секцији реализована је коришћењем паралелног меморијског подсистема са два нивоа хијерархије [36] описаном у оквиру дисертације [138]. Према објављеним резултатима, постиг-

[1] енг. Three Dimensional Recursive Search

нута је осам пута већа ефикасност обраде у односу на претходни највиши степен развоја, у смислу пропусности по квадратном милиметру површине процесора. Стога се меморијски подсистем са два нивоа хијерархије, или краће двонивовски подсистем [36], као и резултати објављени у дисертацији [138], сматрају релевантним и пореде са меморијским подсистемом предложеним у овом раду.

Меморијски подсистеми предложени у радовима [33], [35] и [34] изабрани су да представљају класу N-банковних подсистема у овој студији случаја, јер омогућавају оптималне начине приступа блоку и реду за упаривање блокова са прецизношћу од једног пиксела. Притом су ово, према нашим сазнањима и у време нашег истраживања, најскорије предложени N-банковни подсистеми и самим тим представљају највиши степен развоја.

2N-банковни подсистем предложен у раду [23] и „BilisPM" подсистем предложен у раду [26] такође омогућавају оптималне начине приступе за примену у упаривању блокова са прецизношћу од једног пиксела, исто као и N-банковни подсистеми. Међутим, они се заснивају на нешто другачијој архитектури меморијских банака од N-банковних подсистема и стога су укључени у студију случаја и поређење.

Овај рад представљају: 1) паралелни меморијски подсистем који омогућава нови 9∗2 начин приступа и 2) одговарајућа нова операција читања из подсистема, која дели прочитани 9∗2 блок на два 8∗2 блока као што је описано у секцији 3.1.3. Притом, меморијски подсистем је реализован на основу архитектуре описане у поглављу 5. Параметри описане архитектуре изабрани су тако да се омогући непоравнати 9∗2 начин приступа, у циљу ефикасне реализације билинеарне интерполације референтних блокова, као и непоравнати 8∗2 и 4∗4 начини приступа за приступ циљним блоковима из тренутног фрејма.

Изабрана меморија ван чипа омогућава поравнати приступ реду од шеснаест пиксела. Стога сви наведени меморијски подсистеми на чипу такође омогућавају поравнати 16∗1 начин приступа, како би подржали ефикасне преносе пиксела из меморије ван чипа. Табела 6.2 приказује најважније одлике свих меморијских подсистема који се пореде у овој студији случаја.

Табела 6.2 – Одлике упоређених паралелних меморијских подсистема.

	Овај рад	Двонивовска [36] (Beric 2008)	N-банковни [33] (Vanne 2008)	N-банковни [34] (Lentaris 2010)	N-банковни [35] (Wing-Yee 2011)	2N-банковни [23] (Liu 2007)	BilisPM [26] (Liu 2012)
Начини приступа (пиксела∗линија)	4∗4, 8∗2, **9∗2**, 16∗1	4∗4, 8∗2, 16∗1	4∗4, 16∗1	4∗4, 8∗2, 2∗8 1∗16, 16∗1	4∗4, 8∗2 2∗8, 16∗1	4∗4, 8∗2, 2∗8, 1∗16 16∗1, дијагонале	4∗4, 8∗2, 2∗8 1∗16, 16∗1
N (пиксела)	16	16	16	16	16	16	16
Blk_H (линија)	4						
C (речи од N пиксела)	8192						
Меморијских банака	8	L0: 8, L1: 4	16	16	16	32	16
Ширина банке (пиксела)	4	L0: 4, L1: 4	1	1	1	1	2
Капацитет банке (речи)	4096	L0:1024, L1:5632	8192	8192	8192	4096	4096
Ширина пиксела (бита)	8	8	8	8	8	8	8
Капацитет (килобајта)	128	L0: 32, L1: 88	128	128	128	128	128

6.3 Анализа меморијских приступа

Ова секција описује и анализира реализације дате методе упаривања блокова са сваким од посматраних меморијских подисистема. Фокус је на потребном броју приступа подсистему, а циљ је да се одреди максимална пропусност обраде која се може постићи са сваким од подсистема.

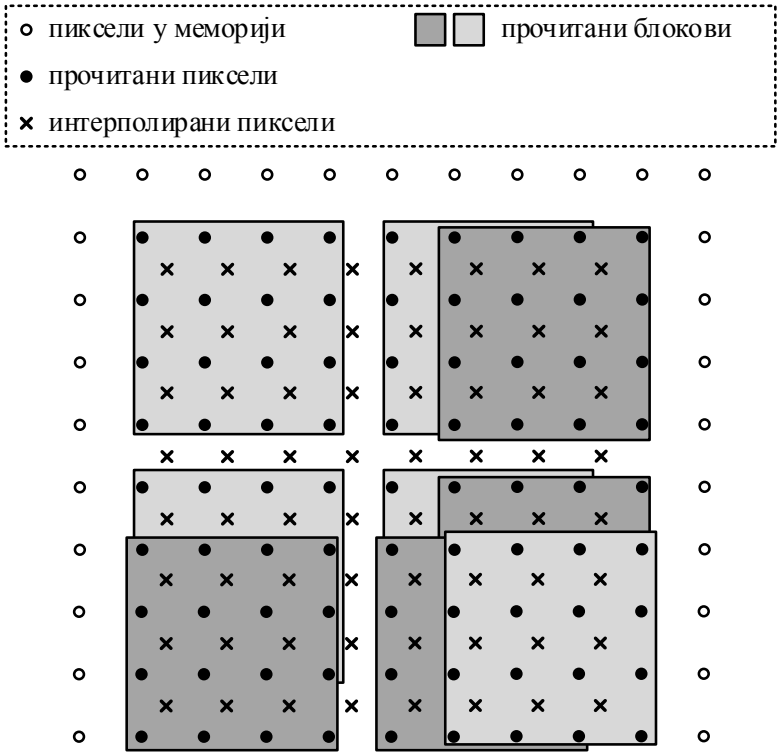
Као што је раније наведено у поглављу 3, за израчунавање једне 8∗8 SAD вредности потребно је прочитати циљни и референтни 8∗8 блок. Циљни блок се чита из меморијског подсистема једном на сваких седам SAD израчунавања, јер постоји седам кандидат вектора по циљном блоку. Притом се у просеку изврши $8 * 8 : N : 7 = 0,6$ приступа по SAD израчунавању. Овај број приступа важи за све меморијске подсистеме. У случају двонивовског подсистема, циљни блок се чита из L1 нивоа меморије, што је у меморијској хијерархији ниво удаљенији од векторске путање података.

Са друге стране, референтни 8∗8 блок креира се билинеарном интерполацијом, што захтева читање једног 9∗9 блока из меморијског подсистема. Читање 9∗9 блока представља критичан део реализације који одређује пропусност обраде код свих посматраних подсистема. Стога се у свим реализацијама обраде тежи минимизовању броја меморијских приступа за читање 9∗9 блокова. Притом се одређене операције приступа подсистему замењују „shuffle" операцијама, које се могу извршити у другом слоту у паралели са меморијским приступима.

Као што је наведено у дисертацији [138], реализација заснована на двонивовском подсистему користи 8∗2 начин приступа за читање 9∗9 блокова. То захтева десет приступа и три „shuffle" операције по једном SAD израчунавању, као што је раније приказано на сликама 3.6, 3.8 и 3.12. Ова читања врше се из L0 нивоа меморије, што је у меморијској хијерархији ниво ближи векторској путањи података.

У случају три N-банковна, 2N-банковног и „BilisPM" подсистема, најефикасније је користити 4∗4 начин приступа за читање 9∗9 блокова, уз који је потребно извршити девет приступа и пет „shuffle" операција по једном SAD израчунавању, као што приказује слика 6.1.

Што се тиче реализације коришћењем меморијског подсистема предложеног у овом раду, потребно је само пет 9∗2 приступа и три „shuffle" операције да се прочита 9∗9 блок и изврши билинеарна интерполација 8∗8 референтног блока, као што је раније приказано на сликама 3.9 и 3.13.



Слика 6.1 – Подела блока од 9∗9 пиксела на 4∗4 блокове, који се читају за билинеарну интерполацију 8∗8 референтног блока.

Поред наведених приступа читања из меморијског подсистема на чипу, врше се и уписи у подсистем, којима се делови тренутног и референтног фрејма преносе из меморије ван чипа. Како би се минимизовао број приступа меморији ван чипа на вредност што ближу теоријском минимуму од једног приступа по пикселу, претпоставка је да се преноси врше коришћењем клизајуће–L1 методе [36, 138]. Ова метода се реализује у програму процесора, а односи се на начин поделе фрејмова у меморији ван чипа на регионе и преноса у меморијски подсистем на чипу. Уколико се претпостави теоријски минимум од једног приступа по пикселу у меморији ван чипа, то значи да се по једном циљном блоку пренесу два 8∗8 блока, један из тренутног, а други из референтног фрејма. То даље значи да је потребно $2 * 8 * 8 : N : 7 = 1,1$ уписа у меморијски подсистем на чипу по једном SAD израчунавању. У случају двонивовског подсистема, ови уписи се врше у L1 ниво меморије.

Додатни приступи потребни су за реализацију засновану на двонивовском меморијском подсистему, како би се пиксели пренели из L1 у L0 ниво меморије. Број читања из L1 једнак је броју уписа у L0 ниво меморије и зависи од вертикалне димензије области претраге изражене у 8∗8 блоковима [36], као и од

редоследа обраде блокова тренутног фрејма [137]. Код реализације засноване на двонивовском подсистему и описане у дисертацији [138], вертикална димензија области претраге једнака је $96 : 8 = 12$ блокова, а редослед обраде блокова је 2-меандер[2], предложен у [137]. Стога се сваком нивоу меморије, односно L0 и L1 нивоима, приступи $8 * 8 * (12 + 2) : N : 7 : 2 = 4$ пута по једном SAD израчунавању, што је детаљно објашњено у [137] и [138].

На основу претходно реченог може се одредити укупан број приступа по једном SAD израчунавању за сваки од подсистема. У случају реализације која представља овај рад, укупан број приступа по SAD израчунавању једнак је 6,7. У случају реализација заснованих на N-банковним, 2N-банковном и „BilisPM" подсистемима, укупан број приступа је 10,7. У случају L0 и L1 нивоа меморије двонивовског подсистема, укупан број приступа је 14 и 5,7, респективно. Као што је и очекивано, потребно је више приступа L0 нивоу меморије, што значи да овај ниво представља уско грло и одређује пропусност обраде.

За обраду једног фрејма резолуције 3840∗2140 пиксела, изврши се 907200 SAD израчунавања, што је израчунато на основу израза: $3840 * 2160 : 8 : 8 * 7$. То захтева 12,7, 9,7, 9,7, 9,7 и 6,1 милиона приступа двонивовском, N-банковном, 2N-банковном, „BilisPM" и подсистему предложеном у овом раду, респективно. Уз фреквенцију циклуса такта од шест стотина мегахерца и практично достижном степену искоришћења путање података процесора од осамдесет процената, максимална пропусност обраде која се може постићи је 38, 49, 49, 49 и 79 фрејмова у секунди, респективно. Притом, пропусности обраде добијене су на основу израза као што је $600 : 6,1 * 0,8 = 79$. У складу са претходном анализом, може се закључити да овај рад омогућава 1,6, 1,6, 1,6 и 2,1 пута већу максималну пропусност обраде у односу на N-банковни, 2N-банковни, „BilisPM" и двонивовски подсистем, респективно. Табела 6.3 приказује резултате добијене анализом у овој секцији.

[2] енг. 2-meander scanning order

Табела 6.3 – Потребан броја приступа подсистему и максимална пропусност.

	Овај рад	Двонивовска [36] (Beric 2008)	N-банковни [33] (Vanne 2008)	N-банковни [34] (Lentaris 2010)	N-банковни [35] (Wing-Yee 2011)	2N-банковни [23] (Liu 2007)	BilisPM [26] (Liu 2012)
			Број приступа подсистему по једном SAD израчунавању				
Читање тренутног блока	0,6	0,6	0,6	0,6	0,6	0,6	0,6
Интерполација референтног блока	5	10	9	9	9	9	9
Пренос из меморије ван чипа	1,1	1,1	1,1	1,1	1,1	1,1	1,1
Читање из L1 у L0 ниво		4					
Упис у L0 из L1 нивоа		4					
Укупно	6,7	L0:14, L1:5,7	10,7	10,7	10,7	10,7	10,7
			Број приступа подсистему по једном обрађеном фрејму				
Укупно (милиона)	6,1	12,7	9,7	9,7	9,7	9,7	9,7
			Максимална пропусност уз фреквенцију такта од 600 мегахерца и степен искоришћења јединице за читање из подсистема од 80%				
Пропусност (фрејмова у секунди)	79	38	49	49	49	49	49
Убрзање које доноси овај рад (пута)		2,1	1,6	1,6	1,6	1,6	1,6

6.4 Експериментални резултати

У циљу практичне потврде претходне анализе и као доказ предности овог рада у односу на подсистеме који представљају тренутно највиши степен развоја, реализована је метода упаривања блокова описана у секцији 6.1 и извршен је низ експеримената.

Прво је реализован меморијски подсистем предложен у овом раду, као параметризовани модул у језику високог нивоа за опис хардвера. Како би се одредило заузеће површине меморијских банака и интерне контролне и управљачке логике, извршена је синтеза модула са параметрима који су дати у табели 6.2. За синтезу је коришћен „Cadence RTL compiler" алат и библиотека логичких кола и меморијских банака реализованих у шездесет пет нанометарској CMOS технологији и оптимизованих за ниску потрошњу. Добијено заузеће површине управљачке логике износи мање од пет процената укупне површине подсистема. Додатно извршене синтезе, за све вредности параметара приказаних на сликама 5.14 и 5.15, показују да заузеће површине управљачке логике износи између три и тринаест процената укупне површине подсистема. Стога, у случају меморијског подсистема предложеног у овом раду, може се закључити да је површина меморијских банака заиста доминантна у односу на површину управљачке логике, као што је претпостављено у секцији 5.6.

За реализацију упаривања блокова развијен је векторски процесор коришћењем технологије компаније „Silicon Hive", која се састоји од библиотеке основних компонената процесора, језика високог нивоа за опис нових компонената, језика за опис архитектуре процесора и алата за генерисање RTL[3] модела и софтверског симулатора процесора [140, 141]. Развијени процесор састоји се од паралелног меморијског подсистема описаног у овом раду, односно варијанте специфициране у табели 6.2 и путање података са векторским функционалним јединицама. Према моделу извршавања операција, архитектура процесора је са веома дугачком програмском речју, означена са VLIW[4]. VLIW векторска путања података процесора састоји се од осам слотова за извршавање операција, од којих су четири са векторским функционалним јединицама ширине $N = 16$ пиксела и четири са скаларним јединицама ширине тридесет два бита. Оба типа функционалних јединица, векторске и скаларне, садрже RISC аритметичко-

[3] енг. Register Transfer Level

[4] енг. Very-Long-Instruction-Word

логичке операције и неколико операција прилагођених упаривању блокова. То су, између осталих, „rd_9*2", „lin", „shuffle", и „isum", које су илустроване на сликама 3.9 и 3.13, као и „subabs", која представља комбинацију операције одузимања „sub" и операције апсолутне вредност „abs". Операције „rd_9*2", „lin", „shuffle" и „subabs" распоређене су у различите слотове путање података, јер представљају критичне ресурсе који одређују пропусност обраде. Иста путања подата првобитно је коришћена и оптимизована за реализацију упаривања блокова уз коришћење двонивовског меморијског подсистема, што је публиковано у дисертацији [138] и где је постигнут степен искоришћења путање података од деведесет три процента. У циљу фер поређења меморијских подсистема, путања података није додатно оптимизована за реализацију коришћењем меморијског подсистема предложеног у овом раду.

За описани процесор развијен је програм који реализује посматрану методу упаривања блокова. Програм је реализован у проширеној варијанти програмског језика C, која има подршку за векторски тип података и векторске операције. За превођење развијеног програма и распоређивање операција у слотове коришћени су „Silicon Hive" VLIW програмски преводилац и распоређивач [142], респективно. Преводилац подржава уобичајене оптимизације кода, као што су софтверска проточна обрада[5] и одмотавање петље[6], које су примењене у циљу постизања вишег степена искоришћења функционалних јединица, регистара и магистрала процесора.

Пропусност обраде мерена је „Silicon Hive" симулатором процесора, чија је прецизност један циклус такта. Пропусност обраде изражена је бројем обрађених фрејмова у секунди. Потрошња паралелног меморијског подсистема на чипу, изражена у миливатима, израчуната је на основу броја извршених приступа подсистему, коришћених начина приступа, поравнања приступа и модела меморијских банака у коришћеној шездесет пет нанометарској CMOS технологији. Број приступа меморији ван чипа по једном пикселу обрађених фрејмова такође је измерен у симулатору. Резултати мерења приказани су у колони „Овај рад" табеле 6.4.

[5] енг. software pipelining

[6] енг. loop unrolling

Табела 6.4 – Експериментални резултати.

Путања података	VLIW са 8 слотова на такту фреквенције 600 мегахерца (4 векторска и 4 скаларна слота)						
Меморијски подсистем	Овај рад	Двонивовски [36] (Beric 2008)	N-банковни [33] (Vanne 2008)	N-банковни [34] (Lentaris 2010)	N-банковни [35] (Wing-Yee 2011)	2N-банковни [23] (Liu 2007)	BilisPM [26] (Liu 2012)
	Заузеће површине (mm^2) у 65 нанометарској CMOS технологији						
Меморијске банке	0,7563	0,7478	0,8967	0,8967	0,8967	0,9834	0,8342
Логика	0,0365	0,0287	0,0083			0,1112	0,0052
Укупно	0,7928	0,7765	0,9050	≈ 0,8967	≈ 0,8967	1,0946	0,8394
Логика/Укупно	5%	4%	1%			10%	1%
Пропусност (фрејмова у секунди)	61	36	43	43	43	43	43
Потрошња (миливата)	47	64	84	84	84	62	56
Приступа по пикселу у меморији ван чипа	1,15	1,32	1,15	1,15	1,15	1,15	1,15
Цена/Пропусност	1	2,6	2,9	2,9	2,9	2,6	1,8

Према нашим сазнањима не постоји објављена реализација упоредиве методе упаривања блокова коришћењем N-банковног, 2N-банковног или „BilisPM" подсистема. Стога, како би се измерила пропусност, потрошња и број приступа меморији ван чипа, реализовано је упаривање блокова за процесор који садржи исту путању података и подсистеме специфициране у табели 6.2. Заузеће површине управљачке логике објављено је у публикацијама [23], [26] и [33] док за подсистеме предложене у [35] и [34] овај податак није објављен. Резултати мерења у случају ових подсистема дати су у колонама „N-банковни", „2N-банковни" и „BilisPM" табеле 6.4.

Колона „двонивовски" у истој табели приказује резултате за реализацију засновану на двонивовском меморијском подсистему, објављене у дисертацији [138]. Како се објављени резултати односе на резолуцију видеа од 1920∗1080 пиксела и деведесет нанометарску CMOS технологију, пропусност обраде и број приступа меморији ван чипа прерачунати су за 3840∗2160 резолуцију, а заузеће површине и укупна потрошња подсистема су прерачунати за шездесет пет нанометарску CMOS технологију.

Као мера ефикасности меморијског подсистема, израчунат је производ свих мера његове цене, односно заузећа површине, потрошње енергије и броја приступа меморији ван чипа и подељен са постигнутом пропусношћу обраде. За сваки подсистем, израчуната мера је нормализована дељењем са мером ефикасности подсистема предложеног у овом раду. Према овој мери, подсистем је ефикаснији што је њена вредност мања. Нормализоване вредности ове мере су дати у колони „Цена/Пропусност" табеле 6.4.

Добијени резултати показују да је меморијски подсистем предложен у овом раду ефикаснији од 1,8 до 2,9 пута у односу на друге посматране подсистеме. Притом, постигнута је од 1,4 до 1,7 пута већа пропусност обраде уз потрошњу енергије подсистема мању од 1,2 до 1,8 пута. Већа пропусност и мања потрошња резултат су вишеструко мањег броја читања из подсистема при интерполацији референтних блокова. Ову разлику приказује табела 6.5 у реду „читање референтних фрејмова". Број читања и потрошња у осталим редовима једнаки су код свих подсистема. Мањој потрошњи доприноси и већа ефикасност предложеног подсистема у случају приступа блоковима од N пиксела. Ово се може приметити у редовима табеле „читање тренутних", „упис референтних" и „упис тренутних" фрејмова, где је број приступа једнак код свих подсистема сем двонивовског, а потрошња значајно мања у случају предложеног подсистема.

Табела 6.5 – Рашчлањени број приступа и потрошња енергије меморијских подсистема.

	Овај рад	Двонивовска [36] (Beric 2008)	N-банковни [33] (Vanne 2008)	N-банковни [34] (Lentaris 2010)	N-банковни [35] (Wing-Yee 2011)	2N-банковни [23] (Liu 2007)	BilisPM [26] (Liu 2012)
				Милиона приступа меморијском подсистему у секунди			
Читање референтних фрејмова	276,70	553,39	498,05	498,05	498,05	498,05	498,05
Читање тренутних фрејмова	31,62	31,62	31,62	31,62	31,62	31,62	31,62
Упис референтних фрејмова	40,85	51,91	40,85	40,85	40,85	40,85	40,85
Упис тренутних фрејмова	31,62	31,62	31,62	31,62	31,62	31,62	31,62
Читање из L1 у L0 ниво		240,59					
Упис у L0 из L1 нивоа		240,59					
Укупно	380,79	L0:793,98, L1:355,74	602,14	602,14	602,14	602,14	602,14
				Потрошња енергије подсистема у миливатима			
Читање референтних фрејмова	36,51	49,40	97,37	97,37	97,37	72,35	66,45
Читање тренутних фрејмова	2,78	3,74	6,18	6,18	6,18	4,59	3,75
Упис референтних фрејмова	4,29	7,00	8,77	8,77	8,77	6,53	5,44
Упис тренутних фрејмова	3,32	4,27	6,79	6,79	6,79	5,05	4,21
Читање из L1 у L0 ниво		28,48					
Упис у L0 из L1 нивоа		14,87					
Укупно при читању	39,29	L0:49,40, L1:32,22	103,55	103,55	103,55	76,94	70,20
Укупно при уписивању	7,61	L0:14,87, L1:11,27	15,56	15,56	15,56	11,58	9,65
Укупно	46,90	107,76	119,11	119,11	119,11	88,52	79,85

Постигнута пропустност је и даље за око двадесет процената мања од максималне коју предложени меморијски подсистем омогућава, а која је прорачуната у секцији 6.3. Ово је резултат суб-оптималног искоришћења функционалне јединице за читање из меморијског подсистема од око шездесет процената. Максимална пропусност може се постићи оптимизацијом путање података за предложени подсистем, што у овој студији случаја није урађено ради фер поређења меморијских подсистема. Конкретно, под оптимизацијом се у овом случају подразумева додавање два векторска слота са „lin" и „isum" операцијама. На тај начин би се избегло надметање у извршавању ове две и других операција на критичној путањи. Даље повећање броја векторских слотова не би донело додатно повећање пропусности јер би меморијски подсистем био скоро потпуно искоришћен већ са шест слотова. Са друге стране, са мање од четири слота, пропусност би била мања јер би се неке од операција на критичној путањи морале извршавати секвенцијално у истом слоту уместо паралелно.

Пропусност постигнута уз предложени меморијски подсистем већа је за двадесет пет фрејмова у секунди у односу на реализацију са двонивовским подсистемом. Две основне разлике између ова два подсистема су број нивоа меморијске хијерархије и нови 9∗2 начин приступа. Упоређујући постигнуту пропусност и разлике, N-банковни, 2N-банковни и „BilisPM" подсистеми налазе се између претходна два, јер имају један ниво меморијске хијерархије, као и предложени подсистем, а омогућавају исте начине приступа од $N = 16$ пиксела као и L0 ниво меморије двонивоовског подсистема. Уколико се погледа постигнута пропусност, може се закључити да архитектура са једним уместо два нивоа хијерархије доноси око двадесет процената већу пропусност. Додатно на такво убрзање, нови 9∗2 начин приступа доноси још четрдесет процената већу пропусност у односу на N-банковне, 2N-банковни и „BilisPM" подсистеме.

Као што је претпостављено у секцији 6.3, остварени број приступа меморији ван чипа близак је теоретском минимуму од једног приступа по пикселу у свим случајевима. У случају двонивовског подсистема, број приступа меморији ван чипа већи је за петнаест процената у односу на остале подсистеме. Разлог томе је за осам килобајта мањи укупан капацитет двонивовског подсистема у односу на остале подсистеме, као и мање ефикасно коришћење меморије на чипу јер у L1 нивоу меморије постоји копија сваког пиксела смештеног у L0 ниво меморије.

7
ЗАКЉУЧАК

У овом раду представљено је ново решење паралелног меморијског подсистема за примену у векторским процесорима слике и видеа за мобилне уређаје. Предложено решење вишеструко је ефикасније од постојећих решења, јер омогућава већу брзину, односно већу пропусност обраде, истовремено трошећи мање енергије при обради истог броја пиксела. Висока ефикасност постигнута је применом нових функционалности меморијског подсистема, односно нових начина приступа блоку или реду пиксела, као и економичној архитектури меморијских банака која реализује те нове функционалности.

Када се користе за реализацију једне од кључних примитива у обради слике и видеа, естимацију кретања суб-пиксел упаривањем блокова, предложени начини приступа два пута смањују укупан број меморијских приступа потребних за линеарну и билинеарну интерполацију и на тај начин омогућавају до два пута већу брзину обраде од најефикаснијих постојећих решења. Притом, за исту количину обрађених пиксела, укупна утрошена енергија предложеног меморијског подсистема двоструко је мања, као последица два пута мањег укупног броја приступа и потрошњи по једном приступу која је слична потрошњи по приступу постојећих подсистема. Показано је да, захваљујући разноврсним и прилагодљивим начинима приступа, предложено решење нуди исте предности

и у многим другим методама обраде као што су компензација кретања, интерполација пиксела и филтрирање у просторном домену. У методама где нови начини приступа не доносе предност, предложено решење једнако је ефикасно као и постојећа решења паралелног меморијског подсистема.

У експертименталној студији случаја, која је представила реализацију добро познате и често примењиване 3DRS методе за естимацију кретања суб-пиксел упаривањем блокова [40], предложено решење омогућило је од 40 до 70 процената бржу обраду, истовремено трошећи од 17 до 44 процента мање енергије од других шест решења паралелног меморијског подсистема који представљају највиши степен технолошког развоја у овој области. Оваква побољшања постигнута су уз практично исто заузеће површине на чипу и исти број приступа меморији ван чипа. Посматрајући укупну ефикасност меморијског подсистема као количник брзине обраде и производа свих трошкова, где спадају потрошња енергије, заузеће површине на чипу и број приступа меморији ван чипа, предложено решење је од 1,8 до 2,9 пута ефикасније од постојећих решења.

Захваљујући овако високој ефикасности, предложено решење нуди велике предности у многим применама обраде слике и видеа, као што су повећање броја фрејмова у секунди и скалирање резолуције при репродукцији видеа, кодовање и компресија видеа, потискивање шума, повећање динамичког опсега боја, стабилизација видеа и супер-резолуција. Резултат тога је примена предложеног решења у најновијим и најнапреднијим Интеловим процесорима и системима на чипу за мобилне уређаје, као и наставак развоја ради примене у наредним генерацијама Интел чипова. У прилог иновативности овог рада говори и чињеница да је Интел корпорација поднела две патентне апликације Патентном заводу Сједињених Америчких Држава, ради заштите права на примену предложеног решења паралелног меморијског подсистема [143, 144].

ЛИТЕРАТУРА

[1] G. de Haan, *Digital Video Post Processing.* Eindhoven, The Netherlands: Royal Philips Electronics, 2010.

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, New Jersey 07458, USA: Pearson Education, Inc, 2007.

[3] A. C. Bovik, *Handbook of image and video processing.* Academic press, 2010.

[4] I. E. Richardson, *H.264 and MPEG-4 video compression: video coding for next-generation multimedia.* Hoboken, NJ, USA: John Wiley & Sons Inc., 2003.

[5] G. Puglisi and S. Battiato, "A Robust Image Alignment Algorithm for Video Stabilization Purposes," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 10, pp. 1390–1400, oct. 2011.

[6] R. Lukac, *Computational Photography: Methods and Applications*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2010.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *Image Processing, IEEE Transactions on*, vol. 16, no. 8, pp. 2080–2095, 2007.

[8] S. H. Keller, "Video upscaling using variational methods," Ph.D. dissertation, Ph. D. thesis, Faculty of Science, University of Copenhagen, 2007.

[9] P. Milanfar, *Super-resolution imaging.* Boca Raton, FL, USA: CRC Press Inc., 2010.

[10] M. Woh, S. Mahlke, T. Mudge, and C. Chakrabarti, "Mobile Supercomputers for the Next-Generation Cell Phone," *Computer*, vol. 43, no. 1, pp. 81–85, jan. 2010.

[11] W. Hwangbo and C. M. Kyung, "A Multitransform Architecture for H.264 / AVC High-Profile Coders," *Multimedia, IEEE Transactions on*, vol. 12, no. 3, pp. 157–167, april 2010.

[12] J. Hennessy and D. Patterson, *Computer architecture: a quantitative approach.* Burlington, MA: Morgan Kaufmann Publishers Inc., 2011.

[13] C. Kozyrakis and D. Patterson, "Scalable, vector processors for embedded systems," *Micro, IEEE*, vol. 23, no. 6, pp. 36–45, nov-dec 2003.

[14] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 1, pp. 61–72, jan 2002.

[15] F. Catthoor, E. d. Greef, and S. Suytack, *Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design.* Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[16] K. Denolf, C. De Vleeschouwer, R. Turney, G. Lafruit, and J. Bormans, "Memory centric design of an MPEG-4 video encoder," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 5, pp. 609–619, may 2005.

[17] P. Budnik and D. Kuck, "The Organization and Use of Parallel Memories," *Computers, IEEE Transactions on*, vol. C-20, no. 12, pp. 1566–1569, dec 1971.

[18] D. Lawrie and C. Vora, "The Prime Memory System for Array Access," *Computers, IEEE Transactions on*, vol. C-31, no. 5, pp. 435–442, may 1982.

[19] D. Lee, "Scrambled storage for parallel memory systems," in *Computer Architecture, 1988. Conference Proceedings. 15th Annual International Symposium on*, may-jun 1988, pp. 232–239.

[20] W. Hinrichs, J. Wittenburg, H. Lieske, H. Kloos, M. Ohmacht, and P. Pirsch, "A 1.3-GOPS parallel DSP for high-performance image-processing applications," *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 7, pp. 946–952, july 2000.

[21] J. W. Park, "An efficient buffer memory system for subarray access," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 12, no. 3, pp. 316–335, mar 2001.

[22] H.-J. Stolberg, M. Berekovic, L. Friebe, S. Moch, S. Flugel, X. Mao, M. Kulaczewski, H. Klussmann, and P. Pirsch, "HiBRID-SoC: a multi-core system-on-chip architecture for multimedia signal processing applications," in *Design, Automation and Test in Europe Conference and Exhibition, 2003*, 2003, pp. 8–13 suppl.

[23] C. Liu, X. Yan, and X. Qin, "An optimized linear skewing interleave scheme for on-chip multi-access memory systems," in *Proceedings of the 17th ACM Great Lakes symposium on VLSI*, ser. GLSVLSI '07. New York, NY, USA: ACM, 2007, pp. 8–13. [Online]. Available: http://doi.acm.org/10.1145/1228784.1228793

[24] D. C. Van Voorhis and T. Morrin, "Memory systems for image processing," *Computers, IEEE Transactions on*, vol. C-27, no. 2, pp. 113–125, Feb 1978.

[25] J. W. Park, "An Efficient Memory System for Image Processing," *Computers, IEEE Transactions on*, vol. C-35, no. 7, pp. 669–674, july 1986.

[26] S. Liu, S. Chen, H. Chen, and Y. Guo, "A novel parallel memory organization supporting multiple access types with matched memory modules," *IEICE Electronics Express*, vol. 9, no. 6, pp. 602–608, 2012.

[27] J. Tanskanen, T. Sihvo, and J. Niittylahti, "Byte and modulo addressable parallel memory architecture for video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, no. 11, pp. 1270–1276, nov 2004.

[28] J. K. Tanskanen, R. Creutzburg, and J. T. Niittylahti, "On Design of Parallel Memory Access Schemes for Video Coding," *Journal of VLSI Signal Processing Systems*, vol. 40, no. 2, pp. 215–237, Jun. 2005.

[29] E. Aho, J. Vanne, and T. Hamalainen, "Parallel Memory Architecture for Arbitrary Stride Accesses," in *Design and Diagnostics of Electronic Circuits and systems, IEEE*, 2006, pp. 63–68.

[30] G. Kuzmanov, G. Gaydadjiev, and S. Vassiliadis, "Multimedia rectangularly addressable memory," *Multimedia, IEEE Transactions on*, vol. 8, no. 2, pp. 315–322, april 2006.

[31] E. Aho, J. Vanne, and T. D. Hamalainen, "Configurable data memory for multimedia processing," *Journal of Signal Processing Systems*, vol. 50, no. 2, pp. 231–249, Feb. 2008.

[32] J.-Y. Peng, X.-L. Yan, D.-X. Li, and L.-Z. Chen, "A parallel memory architecture for video coding," *Journal of Zhejiang University - Science A*, vol. 9, pp. 1644–1655, 2008.

[33] J. Vanne, E. Aho, T. Hamalainen, and K. Kuusilinna, "A Parallel Memory System for Variable Block-Size Motion Estimation Algorithms," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 4, pp. 538–543, april 2008.

[34] G. Lentaris and D. Reisis, "A Graphics Parallel Memory Organization Exploiting Request Correlations," *Computers, IEEE Transactions on*, vol. 59, no. 6, pp. 762–775, june 2010.

[35] W.-Y. Lo, D. Lun, W.-C. Siu, W. Wang, and J. Song, "Improved SIMD Architecture for High Performance Video Processors," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 12, pp. 1769–1783, dec. 2011.

[36] A. Beric, J. van Meerbergen, G. de Haan, and R. Sethuraman, "Memory-Centric Video Processing," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 4, pp. 439–452, april 2008.

[37] F. Kelly and A. Kokaram, "Fast image interpolation for motion estimation using graphics hardware," in *Electronic Imaging 2004*. International Society for Optics and Photonics, 2004, pp. 184–194.

[38] P. Gupta and R. Korada, "Novel algorithm to reduce the complexity of quarter-pixel motion estimation," in *Electronic Imaging 2004*. International Society for Optics and Photonics, 2004, pp. 31–36.

[39] P.-K. Tsung, W.-Y. Chen, L.-F. Ding, C.-Y. Tsai, T.-D. Chuang, and L.-G. Chen, "Single-iteration full-search fractional motion estimation for quad full

HD H.264/AVC encoding," in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, June 2009, pp. 9–12.

[40] G. de Haan and P. W. Biezen, "Sub–pixel motion estimation with 3–D recursive search block-matching," *Signal Processing: Image Communication*, vol. 6, no. 3, pp. 229–239, 1994.

[41] J. Jain and A. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *Communications, IEEE Transactions on*, vol. 29, no. 12, pp. 1799–1808, Dec 1981.

[42] T. Koga, K. Iinuma, A. Hirano, and I. T., "Motion–compensated interframe coding for video conferencing," *Proc. of the National Telecommunications Conference (NTC)*, pp. C5.3.1–C5.3.5, Nov 1981.

[43] R. Srinivasan and K. Rao, "Predictive coding based on efficient motion estimation," *Communications, IEEE Transactions on*, vol. 33, no. 8, pp. 888–896, Aug 1985.

[44] S. Kappagantula and K. Rao, "Motion compensated interframe image prediction," *Communications, IEEE Transactions on*, vol. 33, no. 9, pp. 1011–1015, Sep 1985.

[45] M. Bierling, "Displacement estimation by hierarchical block matching," in *In Proceedings of SPIE: Visual communications and image processing (VCIP'88)*, 1988, pp. 942–951.

[46] B. Girod and F. Joubert, "Motion-compensating prediction with fractional pel accuracy for 64 kbit/s coding of moving video," in *Proc. Internat. Workshop 64kbit/s Coding Moving Video, Hannover, Germany*, 1988.

[47] M. Ghanbari, "The cross–search algorithm for motion estimation," *Communications, IEEE Transactions on*, vol. 38, no. 7, pp. 950–953, Jul 1990.

[48] L.-G. Chen, W.-T. Chen, Y.-S. Jehng, and T.-D. Chiuch, "An efficient parallel motion estimation algorithm for digital image processing," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 1, no. 4, pp. 378–385, Dec 1991.

[49] A. Zaccarin and B. Liu, "Fast algorithms for block motion estimation," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 3, Mar 1992, pp. 449–452.

[50] J.-S. Kim and R.-H. Park, "A fast feature–based block matching algorithm using integral projections," *Selected Areas in Communications, IEEE Journal on*, vol. 10, no. 5, pp. 968–971, Jun 1992.

[51] G. de Haan, P. Biezen, H. Huijgen, and O. Ojo, "True–motion estimation with 3–D recursive search block matching," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 3, no. 5, pp. 368–379, 388, Oct 1993.

[52] B. Girod, "Motion–compensating prediction with fractional–pel accuracy," *Communications, IEEE Transactions on*, vol. 41, no. 4, pp. 604–612, Apr 1993.

[53] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 3, no. 2, pp. 148–157, Apr 1993.

[54] M.-J. Chen, L.-G. Chen, and T.-D. Chiueh, "One–dimensional full search motion estimation algorithm for video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 4, no. 5, pp. 504–509, Oct 1994.

[55] R. Li, B. Zeng, and M.-L. Liou, "A new three–step search algorithm for block motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 4, no. 4, pp. 438–442, Aug 1994.

[56] L.-M. Po and W.-C. Ma, "A novel four–step search algorithm for fast block motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, no. 3, pp. 313–317, Jun 1996.

[57] L.-K. Liu and E. Feig, "A block–based gradient descent search algorithm for block motion estimation in video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, no. 4, pp. 419–422, Aug 1996.

[58] J. Y. Tham, S. Ranganath, M. Ranganath, and A. Kassim, "A novel unrestricted center–biased diamond search algorithm for block motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 8, no. 4, pp. 369–377, Aug 1998.

[59] G. de Haan and P. Biezen, "An efficient true–motion estimator using candidate vectors from a parametric motion model," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 8, no. 1, pp. 85–91, Feb 1998.

[60] S. Borman, M. A. Robertson, and R. L. Stevenson, "Block-matching subpixel motion estimation from noisy undersampled frames: an empirical performance evaluation," in *Electronic Imaging'99*. International Society for Optics and Photonics, 1998, pp. 1442–1451.

[61] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block–matching motion estimation," *Image Processing, IEEE Transactions on*, vol. 9, no. 2, pp. 287–290, Feb 2000.

[62] A. Tourapis, O. Au, M.-L. Liou, G. Shen, and I. Ahmad, "Optimizing the MPEG-4 encoder-advanced diamond zonal search," in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, vol. 3, 2000, pp. 674–677.

[63] V. Christopoulos and J. Cornelis, "A center–biased adaptive search algorithm for block motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 10, no. 3, pp. 423–426, Apr 2000.

[64] O.-C. Chen, "Motion estimation using a one–dimensional gradient descent search," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 10, no. 4, pp. 608–616, Jun 2000.

[65] M. Brunig and W. Niehsen, "Fast full–search block matching," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 2, pp. 241–247, Feb 2001.

[66] J.-H. Lee, K. W. Lim, B. C. Song, and J. B. Ra, "A fast multi–resolution block matching algorithm and its LSI architecture for low bit–rate video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 12, pp. 1289–1301, Dec 2001.

[67] A. Tourapis, O. Au, and M.-L. Liou, "Highly efficient predictive zonal algorithms for fast block–matching motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 10, pp. 934–947, Oct 2002.

[68] H.-Y. Tourapis and A. Tourapis, “Fast motion estimation within the H.264 codec,” in *Multimedia and Expo, 2003. ICME ’03. Proceedings. 2003 International Conference on*, vol. 3, July 2003, pp. III–517–20.

[69] W. I. Choi, B. Jeon, and J. Jeong, “Fast motion estimation with modified diamond search for variable motion block sizes,” in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 2, Sept 2003, pp. II–371–4.

[70] C.-H. Cheung and L.-M. Po, “A novel cross–diamond search algorithm for fast block motion estimation,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 12, pp. 1168–1177, Dec 2002.

[71] Y.-W. Huang, S.-Y. Ma, C.-F. Shen, and L.-G. Chen, “Predictive line search: an efficient motion estimation algorithm for MPEG-4 encoding systems on multimedia processors,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 1, pp. 111–117, Jan 2003.

[72] J. Wang, D. Wang, and W. Zhang, “Temporal compensated motion estimation with simple block-based prediction,” *Broadcasting, IEEE Transactions on*, vol. 49, no. 3, pp. 241–248, 2003.

[73] J.-H. Lee and N. S. Lee, “Variable block size motion estimation algorithm and its hardware architecture for H.264/AVC,” in *Circuits and Systems, 2004. ISCAS ’04. Proceedings of the 2004 International Symposium on*, vol. 3, May 2004, pp. III–741–4.

[74] C. Lam, L. Po, and C. Cheung, “A novel kite–cross–diamond search algorithm for fast video coding and video conferencing applications,” in *In proceeding of IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP’04)*, 2004, pp. 365–368.

[75] X. Li, X. Li, and Y.-K. Chen, “Fast multi–frame motion estimation algorithm with adaptive search strategies in H.264,” in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP ’04). IEEE International Conference on*, vol. 3, May 2004, pp. III–369–72.

[76] Z. Zhou, M.-T. Sun, and Y.-F. Hsu, “Fast variable block–size motion estimation algorithm based on merge and slit procedures for H.264/MPEG-4 AVC,”

in *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, vol. 3, May 2004, pp. III–725–8.

[77] M.-J. Chen, Y.-Y. Chiang, H.-J. Li, and M.-C. Chi, "Efficient multi–frame motion estimation algorithms for MPEG-4 AVC/JVT/H.264," in *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, vol. 3, May 2004, pp. III–737–40.

[78] C.-H. Kuo, M. Shen, and C.-H. Kuo, "Fast inter–prediction mode decision and motion search for H.264," in *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, vol. 1, June 2004, pp. 663–666.

[79] P. Yang, Y.-W. He, and S. Q. Yang, "An unsymmetrical–cross multi–resolution motion search algorithm for MPEG4–AVC/H.264 coding," in *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, vol. 1, June 2004, pp. 531–534.

[80] R. A. Braspenning and G. de Haan, "True–motion estimation using feature correspondences," in *Proc. SPIE*, vol. 5308, 2004, pp. 396–407. [Online]. Available: http://dx.doi.org/10.1117/12.525625

[81] I. Ahmad, W. Zheng, J. Luo, and M. Liou, "A fast adaptive motion estimation algorithm," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 3, pp. 420–438, March 2006.

[82] S. Li, J. Du, D. Zhao, Q. Huang, and W. Gao, "An improved 3DRS algorithm for video de–interlacing," in *Proc. Picture Coding Symp*, 2006.

[83] Y. Su and M.-T. Sun, "Fast multiple reference frame motion estimation for H.264/AVC," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 3, pp. 447–452, March 2006.

[84] W. I. Choi and B. Jeon, "Hierarchical motion search for H.264 variable–block–size motion compensation," *Optical Engineering*, vol. 45, no. 1, pp. 017 002–017 002–9, 2006. [Online]. Available: http://dx.doi.org/10.1117/1.2150214

[85] Y.-W. Huang, C.-Y. Chen, C.-H. Tsai, C.-F. Shen, and L.-G. Chen, "Survey on block matching motion estimation algorithms and architectures with new results," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 42, no. 3, pp. 297–320, 2006.

[86] Y.-J. Wang, C.-C. Cheng, and T.-S. Chang, "A Fast Algorithm and Its VLSI Architecture for Fractional Motion Estimation for H.264/MPEG-4 AVC Video Coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 578–583, May 2007.

[87] M. Ezhilarasan and P. Thambidurai, "Simplified block matching algorithm for fast motion estimation in video compression," *Journal of Computer Science*, vol. 4, no. 4, p. 282, 2008.

[88] S.-C. Tai, Y.-R. Chen, Z.-B. Huang, and C.-C. Wang, "A multi-pass true motion estimation scheme with motion vector propagation for frame rate up-conversion applications," *Journal of display technology*, vol. 4, no. 2, pp. 188–197, 2008.

[89] C.-M. Kuo, Y.-H. Kuan, C.-H. Hsieh, and Y.-H. Lee, "A Novel Prediction-Based Directional Asymmetric Search Algorithm for Fast Block–Matching Motion Estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 6, pp. 893–899, June 2009.

[90] Y.-S. Cheng, Z.-Y. Chen, and P.-C. Chang, "An H.264 spatio–temporal hierarchical fast motion estimation algorithm for high–definition video," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, May 2009, pp. 880–883.

[91] Z. Chen, "Efficient block matching algorithm for motion estimation," *International Journal of Signal Processing*, vol. 5, no. 2, pp. 133–137, 2009.

[92] Y.-R. Chen and S.-C. Tai, "True motion-compensated de-interlacing algorithm," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 10, pp. 1489–1498, 2009.

[93] C. Bartels and G. de Haan, "Smoothness Constraints in Recursive Search Motion Estimation for Picture Rate Conversion," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 10, pp. 1310–1319, Oct 2010.

[94] A. Heinrich, C. Bartels, R. van der Vleuten, C. Cordes, and G. de Haan, "Optimization of Hierarchical 3DRS Motion Estimators for Picture Rate Conversion," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 2, pp. 262–274, April 2011.

[95] A. Saha, J. Mukherjee, and S. Sural, "A Neighborhood Elimination Approach for Block Matching in Motion Estimation," *Image Communication*, vol. 26, no. 8-9, pp. 438–454, Oct. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.image.2011.06.002

[96] J. Cai and W. David Pan, "On Fast and Accurate Block–based Motion Estimation Algorithms Using Particle Swarm Optimization," *Information Sciences: an International Journal*, vol. 197, pp. 53–64, Aug. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2012.02.014

[97] Z. Cui, G. Jiang, S. Yang, and C. Wu, "A New Fast Motion Estimation Algorithm Based on the Loop–epipolar Constraint for Multiview Video Coding," *Image Communication*, vol. 27, no. 2, pp. 172–179, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.image.2011.11.001

[98] E. Cuevas, D. Zaldivar, M. Pérez-Cisneros, and D. Oliva, "Block-matching algorithm based on differential evolution for motion estimation," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 488–498, 2013.

[99] S. Dikbas and Y. Altunbasak, "Novel true-motion estimation algorithm and its application to motion-compensated temporal frame interpolation," *Image Processing, IEEE Transactions on*, vol. 22, no. 8, pp. 2931–2945, 2013.

[100] N. Al-Najdawi, M. Noor Al-Najdawi, and S. Tedmori, "Employing a Novel Cross–diamond Search in a Modified Hierarchical Search Motion Estimation Algorithm for Video Compression," *Information Sciences: an International Journal*, vol. 268, pp. 425–435, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2013.08.009

[101] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: tools, performance, and complexity," *Circuits and Systems Magazine, IEEE*, vol. 4, no. 1, pp. 7–28, Jan 2004.

[102] E. B. Bellers and G. de Haan, *De-interlacing: A Key Technology for Scan Rate Conversion: A Key Technology for Scan Rate Conversion*. Elsevier, 2000.

[103] P. B. Catrysse and B. A. Wandell, "Roadmap for cmos image sensors: Moore meets planck and sommerfeld," in *Electronic Imaging 2005*. International Society for Optics and Photonics, 2005, pp. 1–13.

[104] J. Farrell, F. Xiao, and S. Kavusi, "Resolution and light sensitivity tradeoff with pixel size," in *Electronic Imaging 2006*. International Society for Optics and Photonics, 2006, pp. 60 690N–60 690N.

[105] F. Xiao, J. E. Farrell, P. B. Catrysse, and B. Wandell, "Mobile imaging: the big challenge of the small pixel," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2009, pp. 72 500K–72 500K.

[106] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising with block-matching and 3D filtering," in *Electronic Imaging 2006*. International Society for Optics and Photonics, 2006.

[107] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3d transform-domain collaborative filtering," in *Proc. 15th European Signal Processing Conference*, vol. 1, no. 2, 2007, p. 7.

[108] G. Boracchi and A. Foi, "Multiframe Raw-Data Denoising Based On Block-Matching And 3-D Filtering For Low-Light Imaging And Stabilization," in *Proceedings of LNLA 2008, the International Workshop on Local and Non-Local Approximation in Image Processing, 22–24 August, 2008 Lausanne, Switzerland*, 2008.

[109] S. Zimmer, S. Didas, and J. Weickert, "A rotationally invariant block matching strategy improving image denoising with non-local means," in *Proc. 2008 International Workshop on Local and Non-Local Approximation in Image Processing*, 2008, pp. 135–142.

[110] Q. Chen and D. Wu, "Image denoising by bounded block matching and 3D filtering," *Signal Processing*, vol. 90, no. 9, pp. 2778–2783, 2010.

[111] C. Liu and W. T. Freeman, "A high-quality video denoising algorithm based on reliable motion estimation," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 706–719.

[112] Y. S. Zhang, S. J. Zhu, and Y. J. Li, "BM3D denoising algorithm with adaptive block-match thresholds," *Applied Mechanics and Materials*, vol. 229, pp. 1715–1720, 2012.

[113] J. Driessen, L. Boroczki, and J. Biemond, "Pel-recursive motion field estimation from image sequences," *Journal of Visual Communication and Image Representation*, vol. 2, no. 3, pp. 259–280, 1991.

[114] R. Schutten and G. De Haan, "Real-time 2-3 pull-down elimination applying motion estimation/compensation in a programmable device," *Consumer Electronics, IEEE Transactions on*, vol. 44, no. 3, pp. 930–938, 1998.

[115] L. Manikandan and D. R. K. Selvakumar, "A new survey on block matching algorithms in video coding," *International Journal of Engineering research*, vol. 3, no. 2, pp. 121–125, Feb. 2014.

[116] K. Rijkse, "H.263: video coding for low-bit-rate communication," *Communications Magazine, IEEE*, vol. 34, no. 12, pp. 42–45, Dec 1996.

[117] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560–576, july 2003.

[118] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[119] P. W. Biezen, G. De Haan, H. Huijgen, and O. A. Ojo, "Apparatus for performing motion-compensated picture signal interpolation," US Grant US 5 534 946, 07 09, 1996. [Online]. Available: http://www.google.com/patents/US5534946

[120] O. Ojo and G. de Haan, "Robust motion-compensated video upconversion," *Consumer Electronics, IEEE Transactions on*, vol. 43, no. 4, pp. 1045–1056, Nov 1997.

[121] P. Haavisto, J. Juhola, and Y. Neuvo, "Fractional frame rate up-conversion using weighted median filters," *Consumer Electronics, IEEE Transactions on*, vol. 35, no. 3, pp. 272–278, Aug 1989.

[122] L. Blume, H. Schwoerer and K. Zygis, "Subband based upconversion using complementary median filters," in *Proceedings of. the 7th International Workshop on HDTV*, 1994.

[123] O. Franzen, “Design and application of weighted median filters for poly-phase image interpolation,” Ph.D. dissertation, Dortmund University (in German), Germany, 2005.

[124] S.-H. Lee, Y.-C. Shin, S. Yang, H.-H. Moon, and R.-H. Park, “Adaptive motion-compensated interpolation for frame rate up-conversion,” *Consumer Electronics, IEEE Transactions on*, vol. 48, no. 3, pp. 444–450, Aug 2002.

[125] D. Wang, A. Vincent, and P. Blanchfield, “Hybrid de-interlacing algorithm based on motion vector reliability,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 8, pp. 1019–1025, Aug 2005.

[126] G. Dane and T. Q. Nguyen, “Optimal temporal interpolation filter for motion-compensated frame rate up conversion,” *Image Processing, IEEE Transactions on*, vol. 15, no. 4, pp. 978–991, April 2006.

[127] Q. Huang, W. Gao, D. Zhao, and H. Sun, “An efficient and robust adaptive deinterlacing technique,” *Consumer Electronics, IEEE Transactions on*, vol. 52, no. 3, pp. 888–895, Aug 2006.

[128] S.-J. Kang, D.-G. Yoo, S.-K. Lee, and Y. H. Kim, “Design and implementation of median filter based adaptive motion vector smoothing for motion compensated frame rate up-conversion,” in *Consumer Electronics, 2009. ISCE ’09. IEEE 13th International Symposium on*, May 2009, pp. 745–748.

[129] A.-M. Huang and T. Nguyen, “Correlation-based motion vector processing with adaptive interpolation scheme for motion-compensated frame interpolation,” *Image Processing, IEEE Transactions on*, vol. 18, no. 4, pp. 740–752, April 2009.

[130] H. Liu, R. Xiong, D. Zhao, S. Ma, and W. Gao, “Multiple hypotheses bayesian frame rate up-conversion by adaptive fusion of motion-compensated interpolations,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 8, pp. 1188–1198, Aug 2012.

[131] G. G. Lee, C.-F. Chen, C.-J. Hsiao, and J.-C. Wu, “Bi-directional trajectory tracking with variable block-size motion estimation for frame rate up-convertor,” *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 4, no. 1, pp. 29–42, March 2014.

[132] *ITU-T Recommendation H.264 : Advanced video coding for generic audiovisual services*, ITU-T Video Coding Experts Group (VCEG) Rec. [Online]. Available: http://www.itu.int/rec/T-REC-H.264-201402-P

[133] T. Mimar, "Method and system for parallel histogram calculation in a SIMD and VLIW processor," US Patent Application US 2009/0 276 606 A1, 11 05, 2009. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_2009_0276606_A1/en/

[134] R. Banakar, S. Steinke, B. S. Lee, M. Balakrishnan, and P. Marwedel, "Scratchpad memory: a design alternative for cache on-chip memory in embedded systems," in *Hardware/Software Codesign, CODES. Proceedings of the 10th International Symposium on*, 2002, pp. 73–78.

[135] I. Puaut and C. Pais, "Scratchpad memories vs locked caches in hard real-time systems: a quantitative comparison," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, april 2007, pp. 1–6.

[136] G. H. Barnes, R. M. Brown, M. Kato, D. J. Kuck, D. L. Slotnick, and R. A. Stokes, "The illiac iv computer," *Computers, IEEE Transactions on*, vol. 100, no. 8, pp. 746–757, 1968.

[137] R. Jakovljevic and A. Beric, "A method for improving the efficiency of a two-level memory hierarchy," in *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, oct 2008, pp. 37–42.

[138] A. Beric, "Video post processing architectures," Ph.D. dissertation, Eindhoven University of Technology, The Netherlands, 2008.

[139] E. Jaspers and P. de With, "Bandwidth reduction for video processing in consumer systems," in *Consumer Electronics, 2001. ICCE. International Conference on*, 2001, pp. 72–73.

[140] G. Burns, M. Jacobs, M. Lindwer, and B. Vandewiele, "Silicon Hive's scalable and modular architecture template for high-performance multi-core systems," in *In Proceedings of International Signal Processing Conference and Expo*, 2006.

[141] C. Pinto, A. Beric, S. Singh, and S. Farfade, "HiveFlex-Video VSP1: Video Signal Processing Architecture for Video Coding and Post-Processing," in *Multimedia, 2006. ISM'06. Eighth IEEE International Symposium on*, Dec 2006, pp. 493–500.

[142] L. Augusteijn, "The HiveCC Compiler for Massively Parallel ULIW Cores," in *Embedded Processor Forum*, San Jose, USA, 2004.

[143] R. Jakovljevic, A. Beric, E. van Dalen, and D. Milicev, "Electronic apparatus having parallel memory banks," WO Patent Application WO 2013/106 210 A1, 07 18, 2013. [Online]. Available: http://patentscope.wipo.int/search/en/WO2013106210

[144] R. Jakovljevic, A. Beric, E. Van Dalen, and D. Milicev, "Intelligent parametric scratchap memory architecture," US Patent Application US 14/129 178 A1, 05 29, 2014. [Online]. Available: http://www.google.com/patents/US20140149657

БИОГРАФИЈА АУТОРА

Радомир Јаковљевић је рођен 12. новембра 1982. године у Горњем Милановцу, где је завршио основну и средњу школу са одличним успехом уз запажене резултате на државним такмичењима из електронике. Електротехнички факултет у Београду уписао је 2001, а дипломирао 2008. године на одсеку за рачунарску технику и информатику. Током студија учествовао је у оснивању, развоју и одржавању академске рачунарске мреже Студентског дома „Патрис Лумумба", која је у то време била јединствена у студентским установама у Србији и шире.

Истраживањем паралелних архитектура процесора за обраду слике и видеа почео је да се бави током стручне праксе у компанији „Силикон Хајв" у технолошком кампусу „Филипса" у Холандији, где је провео осам месеци током 2007. и 2008. године. Његов задатак био је да реализује методу естимације кретања упаривањем блокова на векторском процесору и оптимизује архитектуру тог процесора како би се постигле максималне перформансе. На основу добијених резултата објавио је два научна рада на престижним IEEE SiPS'08 и ICIP'08 конференцијама у Сједињеним Америчким Државама, а потом и написао дипломски рад под називом „Метод за повећање ефикасности двонивовске меморијске хијерархије".

По завршетку стручне праксе и дипломирању, уписао је докторске студије на Електротехничком факултету у Београду и тада успоставио сарадњу између факултета и компаније „Силикон Хајв". У оквиру те сарадње, као запослени факултета на месту сарадника, успешно је реализовао трогодишњи истраживачко-развојни пројекат, на ком је самостално осмислио и реализовао иновативно решење паралелног меморијског подсистема за примену у векторским процесорима за обраду слике и видеа. Ово истраживање и његови резултати представљају Радомиров докторски рад, а његова иновативна идеја публикована је у врхунском међународном часопису категорије М21 „*Real–Time*

Image Processing" издавачке куће Спрингер и заштићена у Патентном заводу Сједињених Америчких Држава.

Предложено решење меморијског подсистема уграђено је у најсавременије Интелове чипове за мобилне уређаје, а његово даље усавршавање је настављено ради примене у будућим генерацијама Интелових чипова.

Радомир је од 2011. до средине 2015. године био запослен у развојном центру Интел корпорације у Београду, која је 2011. године преузела компанију „Силикон Хајв". Он се у Интелу бавио савременим архитектурама и реализацијама хардвера и фирмвера за обраду слике и видеа у мобилним уређајима. Од 2013. године водио је тим за развој специфичних хардверских блокова за убрзање обраде и смањење потрошње енергије.

Радомир је тренутно запослен у развојном центру Мајкрософт корпорације у Београду где води тим за развој софтверских система за управљање базама података.

ПРИЛОГ 1

Изјава о ауторству

Потписани Радомир Јаковљевић
број индекса 5006/08

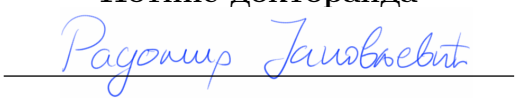
Изјављујем

да је докторска дисертација под насловом „Паралелни меморијски подсистеми за примену у обради слике и видеа у мобилним уређајима”

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, 5.9.2015.



ПРИЛОГ 2

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора:	Радомир Јаковљевић
Број индекса:	5006/08
Наслов рада:	Паралелни меморијски подсистеми за примену у обради слике и видеа у мобилним уређајима
Ментор:	др Драган Милићев, редовни професор

Потписани Радомир Јаковљевић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду**.

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, 5.9.2015.

Радомир Јаковљевић

ПРИЛОГ 3

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић” да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом **„Паралелни меморијски подсистеми за примену у обради слике и видеа у мобилним уређајима”**, која је моје ауторско дело.

Дисертацију са свим прилозима предао сам у електронском формату погодном за трајно архивирање. Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (*Creative Commons*) за коју сам се одлучио.

1. Ауторство
2. Ауторство – некомерцијално
3. Ауторство – некомерцијално – без прераде
4. Ауторство – некомерцијално – делити под истим условима
5. Ауторство – без прераде
6. Ауторство – делити под истим условима

(Кратак опис лиценци дат је на следећој страни)

Потпис докторанда

У Београду, 5.9.2015.

Радомир Јаковљевић

1. Ауторство – Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство — некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство – некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство — без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.