**Универзизитет у Бањој Луци**

**Електротехнички факултет**

**Катедра за електронику**

**Небојша Марковић**

**ИМПЛЕМЕНТАЦИЈА ВИДЕО СКАЛЕРА**

**дипломски рад**

**Бања Лука, октобар 2021.**

**Тема: Имплементација видео скалера**

**Кључне ријечи: видео скалер**

**резолуција**

**обрада слике**

**рад у реалном времену**

**билинеарна трансформација**

**ФПГА**

**Комисија: проф. др Бранко Блануша, предсједник**

**проф. др Жељко Ивановић, ментор**

**проф. др Младен Кнежић, члан**

**Кандидат:**

**Небојша Марковић**

Универзитет у Бањој Луци

Eелектротехнички факултет

Катедра за електронику

Предмет: Пројектовање Дигиталних система

Тема: Имплементација видео скалера

Задатак: Савремени мултимедијални системи подржавају велики број видео формата. Од продукције до приказивања видео садржаја користи се већи број уређаја који користе различите видео формате. Један од уређаја који се користе за прилагођење резолуције видео формата је видео скалер. Улога видео скалера је да видео једне резолуције прилагоди на видео друге резолуције. Скалер може да повећа или да смањи резолуцију видео формата. У раду је потребно имплементирати видео скалер који омогућава скалирање улазног видео садржаја. Модул треба да подржава увећање и смањење резолуције, као и рад у реалном времену. За практичан дио рада потребно је имплементирати модул на ФПГА колу.

Ментор: проф. др Жељко Ивановић

Кандидат: Небојша Марковић (1241/11)

Бања Лука, септембар 2021.

Садржај

[ЛИСТА ОЗНАКА И СКРАЋЕНИЦА I](#_Toc132609403)

[ЛИСТА СЛИКА II](#_Toc132609404)

[ЛИСТА ТАБЕЛА IV](#_Toc132609405)

[1 УВОД 5](#_Toc132609406)

[2 ФОРМУЛАЦИЈА ПРОБЛЕМА 6](#_Toc132609407)

[3 СКАЛИРАЊЕ 8](#_Toc132609408)

[3.1 Преглед метода за скалирање слике 10](#_Toc132609409)

[3.1.1 особине 10](#_Toc132609410)

[3.1.2 најближи сусјед (енг. Nearest Neighbour) 10](#_Toc132609411)

[3.1.3 билинеарна интерполација 12](#_Toc132609412)

[3.1.4 бикубична интерполација 13](#_Toc132609413)

[4 Дигитални електронски уређаји 14](#_Toc132609414)

[4.1 Методологија пројектовања дигиталних кола 14](#_Toc132609415)

[4.2 Програмабилни логички уређаји и њихов развој кроз историју 17](#_Toc132609416)

[4.3 ФПГА 20](#_Toc132609417)

[4.3.1 Област кориштења ФПГА уређаја 21](#_Toc132609418)

[5 Реализација видео скалера у ФПГА технологији 23](#_Toc132609419)

[5.1 Протокол за комуникацију видео скалера са околином, Акси Стрим (*енг.* AXI STREAM) протокол 24](#_Toc132609420)

[5.2 Протокол за комуникацију међу модулима у дизајну 25](#_Toc132609421)

[5.3 Прилагођење улазног тока података на филтер (енг. Stream to rows) модул 27](#_Toc132609422)

[5.4 Билинеарни филтар 28](#_Toc132609423)

[5.4.1 Одабир алгоритма за реализацију 30](#_Toc132609424)

[5.4.2 Рачунање координата оригинала за пикселе на резултујућој слици 30](#_Toc132609425)

[5.4.3 Рачунање резултујућег пиксела на основу познавања вриједности и координата оригинала 34](#_Toc132609426)

[5.4.4 Изведба билинеарног филтра 36](#_Toc132609427)

[5.5 Меморијскe ФИФО (енг. FIFO – First In First Out) банкe 40](#_Toc132609428)

[5.6 Прилагођење података на хоризонтални филтар 43](#_Toc132609429)

[5.7 Излазни ФИФО 43](#_Toc132609430)

[5.8 Прослијеђивање података на Акси стрим (*енг. Axi stream*) 44](#_Toc132609431)

[6 ЗАКЉУЧАК 45](#_Toc132609432)

[7 ЛИТЕРАТУРА 46](#_Toc132609433)

# ЛИСТА ОЗНАКА И СКРАЋЕНИЦА

***Skraćenice:***

1. „Personal Computer“ - PC
2. „Random Access Memory“ – RAM
3. „No operation“ – NOP
4. „Little endian“ – LE

***Oznake:***

1. Gb/s – Gigabit po sekundi
2. MB/s – Megabajt po sekundi
3. GHz – Gigaherc
4. KB – Kilobajt
5. MB – Megabajt
6. B – bajt
7. s – sekunda
8. ms – milisekunda

# ЛИСТА СЛИКА

[Слика 2.1: Стандардизоване димензије слика [2] 7](#_Toc132609337)

[Слика 3.1: Дигиталне репрезентација сиве слике [4] 8](#_Toc132609338)

[Слика 3.2: Пресликавање унапријед 9](#_Toc132609339)

[Слика 3.3: Пресликавање уназад 10](#_Toc132609340)

[Примјеном ове методе, тачки чију вриједност интерполирамо додјељује се вриједност најближе познате сусједне тачке. На дијаграмима испод приказана је апроксимација функције једне промјенљиве кориштењем ове методе (*Слика 3.4*) и област у којој је свака од познатих тачака најближи сусјед за дводимензионалне сигнале. 10](#_Toc132609341)

[Слика 3.5: Апроксимација функције једне промјенљиве методом најближи сусјед 11](#_Toc132609342)

[Слика 3.6: Интерполација дводимензионалне функције методом најближи сусјед 11](#_Toc132609343)

[Слика 3.7: Интерполација методом најближи сусјед са задршком 12](#_Toc132609344)

[Билинеарна интерполација је проширење линеарне интерполације за дводимензионалне промјенљиве, гдје се линеарна интерполација изводи по једној, а затим по другој промјенљивој. За прорачун непознате вриједности у некој тачки потребно је познавати вриједности сусједних тачака и удаљености од њих. На слици испод (*Слика 3.8*) приказана је линеарна интерполација функције f(x) на основу вриједности познатих у одређеном броју тачака. 12](#_Toc132609345)

[Слика 3.9: Апроксимација функције једне промјенљиве примјеном линеарне интерполације 12](#_Toc132609346)

[Слика 3.10: Билинеарна интерполација 13](#_Toc132609347)

[Слика 4.1: Подјела уређаја са дигиталном логиком 14](#_Toc132609348)

[САСИЦ кола се производе у различитим нивоима сложености и ту разликујемо[6][7] (*слика 4.2*): 16](#_Toc132609349)

[Слика 4.3 Подјела програмабилниј логичких уређаја са стаховишта сложености 17](#_Toc132609350)

[Слика 5.1: Видео скалер, блок дизајн 23](#_Toc132609351)

[Слика 5.2: Редослијед слања пиксекла преко стрим интерфејса 24](#_Toc132609352)

[Слика 5.3: Приказ сигнала стрим протокола, исправан податак на линији(valid), спреман за преузимање податка (ready), задњи податак из тренутног реда(last): 25](#_Toc132609353)

[Слика 5.4: Као сигнализација да се шање последњи ред слике користи се сигнал "user". Последњи податак је послан када су и user и valid на високом нивоу 25](#_Toc132609354)

[Слика 5.5: Протокол за комуникацију међу модулима кориштеним за реализацију видео скалера 26](#_Toc132609355)

[Слика 5.6: Информација о позицији реда слике 27](#_Toc132609356)

[Слика 5.7: Модул за прилагођење стрима на билинеарни филтар 28](#_Toc132609357)

[Слика 5.8: Улази, излази и параметри модула који обавља функцију билинеарне интерполације или филтрирања 29](#_Toc132609358)

[Слика 5.11: Проналажење координате оригинала за пиксел резултујуће слике 30](#_Toc132609359)

[Слика 5.12: Растојање међу пикселима са оригиналне слике и растојање уметнутих позиција, позиција на којима се налази оригинал за конкретан резултујући пиксел 31](#_Toc132609360)

[Слика 5.13: Рачунање оригинала за пиксел са резултујуће слике. Информација о најближем пикселу са оригиналне слике и удаљености оригинала од њега 32](#_Toc132609361)

[Слика 5.14: Билинеарни филтр са улазним и излазним сигналима: i - позиција оригинала посматраног резултујућег пиксела, orig\_pix - вриједност пиксела са оригиналне слике, res\_pix- резултујући пиксел, pix\_valid - резултујући пиксел је важећи 33](#_Toc132609362)

[Слика 5.15: Схема модула за рачунање удањености оригинала од пиксела са оригиналне слике. Избор коефицијента филтра 34](#_Toc132609363)

[Слика 5.16: Рачунање резултујућег пиксела на основу утицаја сусједних пиксела оригиналне слике на … 35](#_Toc132609364)

[Слика 5.17: Модел дизајна за рачунање резултујућег пиксела на позицији ј, 36](#_Toc132609365)

[Слика 5.18: Изведба билинеарног филтра из два основна модула 37](#_Toc132609366)

[Слика 5.19: Ћелија за рачунање индекса за одабир коефицијената филтера 38](#_Toc132609367)

[Слика 5.20: Модул за избор коефицијената у билинеарном филтру 39](#_Toc132609368)

[Слика 5.21: Модул за рачунање позиције следећег резултантног пиксела 40](#_Toc132609369)

[Слика 5.9: Колекција ФИФО банака 42](#_Toc132609370)

[Слика 5.10: Прилагођење тока података на џоризонтални филтар 43](#_Toc132609371)

# ЛИСТА ТАБЕЛА

**No table of figures entries found.**

# УВОД

# ФОРМУЛАЦИЈА ПРОБЛЕМА

Видео скалер је блок или урећај за промјену димензија оквира слике (*frame-a*) улазног видео садржаја. Да би се могло приступити проблему израде самог дизајна, потребно је прво размотрити захтјеве који су пред нама, да би се могло одговорити на исте.

У сврху анализе захтјева, навешћемо још једном добијену дефиницију задатка, у којој се каже: „Савремени мултимедијални системи подржавају велики број видео формата. Од продукције до приказивања видео садржаја користи се већи број уређаја који користе различите видео формате. Један од уређаја који се користе за прилагођење резолуције видео формата је видео скалер. Улога видео скалера је да видео једне резолуције прилагоди на видео друге резолуције. Скалер може да повећа или да смањи резолуцију видео формата. У раду је потребно имплементирати видео скалер који омогућава скалирање улазног видео садржаја. Модул треба да подржава увећање и смањење резолуције, као и рад у реалном времену. За практичан дио рада потребно је имплементирати модул на ФПГА (FPGA – Field Programmable Gate Arrays) колу“.

Директно се захтијева:

* скалер може да увећава димензије улазне слике (видеа)
* скалер може да умањи димензије улазне слике (видеа)
* потребно је да уређај има могућност рада у реалном времену, те
* Сама промјена димензија слике треба да се одвија помоћу алгоритма имплементираног на ФПГА колу.

Због честе потребе да се слика са улаза на излаз модула само „проусти“, а с обзиром да се то не противи наведеним захтјевима, претпоставићемо да је и то један од захтјева које треба испунити.

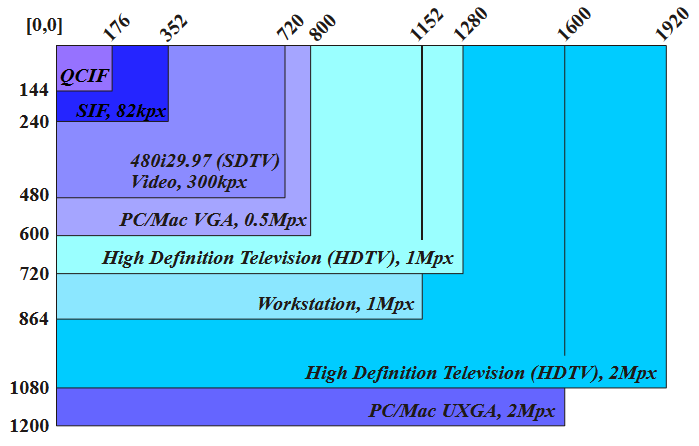
Осим наведених захтјева постоје и друге ствари које значајно могу да утичу на коначно рјешење, као што су:

* Формат улазног видеа
* Захтјевани квалитет излазног видеа
* Минималне и максималне димензије улазне/излазне слике (видеа)
* Цијена представљеног рјешења итд.

Остали фактори који могу утицати на коначно рјешење нису ексшлицитно наведени, па их треба накндадно договорити или усвојити.

Међу могућим улазним форматима одлучујемо се за цтно-бијелу слику(*gray-scale*) гдје су вриједности свјетлина једне тачке представљене 8-битним цјелобројним вриједностима, тј. вриједностима од 0 до 255. Разлог за одабир овог формата јесте релативно лака реализација у односу на друге формате, релативно лако проширење развијеног дизајна на рад са сликама у боји, које су у „RGB“ формату, очекивање најмање потребних ресурса (што директно утиче на избор ФПГА кола које би се користило и могућност смањења трошкова).

За минималне димензије слике на улазу усвојићемо димензије најмање стандардне видео слике (144х176), док ћемо за максималну величину изабрати (1200х1920), како би смо покрили све формате са *Слика 2.5.*



Слика 2.1: Стандардизоване димензије слика [2]

На *слици 2.5.* приказани су само некиод стандардних димензија видео формата.

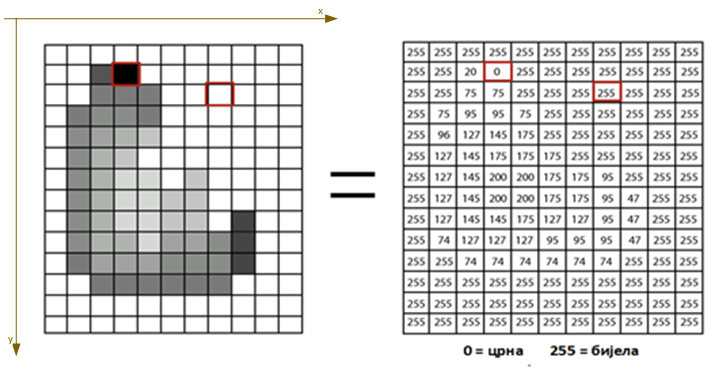
Такође, и ако се ради о видеу, фокусираћемо се само на промјену резолуције слике, док ћемо рад са звуком изоставити у овом раду.

# СКАЛИРАЊЕ

Видео се може дефинисати као покретна слика.

По природи, слика је аналогна, па је за употребу у дигиталним системима потребно направити њену дигиталну репрезентацију. При кориштењу појма дигитална слика обично се мисли на растерску слику, тј дводимензионалну дискретизовану слику, која се састоји из коначног броја дигиталних вриједности, тачака (*pixel-a)*. Пиксели су најмањи елементи слике и садрже вриједности освјетљења за сваку тачку слике. Дигитална слика садржи коначан број пиксела.

У дигиталном домену слика ће бити посматрана као функција двије реалне промјенљиве, f(x,y), гдје су (x,y) просторне координате које одређују положај тачке у равни слике (*Слика 5*).



Слика 3.1: Дигиталне репрезентација сиве слике [4]

Репрезентација вриједности свјетлине појединих пиксела може бити различита за различите формате слике.

Дигитална обрада слике огледа се у:

* мијењању вриједности интензитета тачака слике и
* мијењању просторних односа тачака на слици.

Трансформације засноване на промјени просторних односа тачака слике називају се геометријским трансфорамацијама. Овакве трансформације могу се свести на двије основне операције:

* Дводимензионалну просторну трансформацију, која одређује како се тачке помјерају при формирању резултујуће слике и
* Процјену вриједности свјетлина тачака на резултујућој слици, тзв. интерполацију.

Скалирање слике потпада под геометријске трансформације. Резултат операције скалирања јесте увећана или умањена слика, или у специјалном случају слика истих димензија као и улазна. Примјеном операције скалирања на слику мијењају се дужине линија на слици, али се чува њихова паралелност и углови између њих. Због ове особине скалирање спада у специјалну групу геометријских трансформација, тзв. афине трансформације.

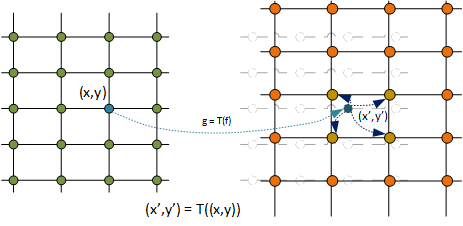
Скалирање, као геометријска трансформација, може се свести на одређивање положаја тачака приликом трансформације и на интерполацију њиховог интензитета. Ако усвојимо следеће ознаке:

* Т оператор пресликавања
* f, оригинална слика
* (x, y), координате тачака у оригиналној слици f
* g, резултујућа слика
* (x’, y’), координате тачака у резултујућој слици g

Резултујућа слика се добија као примјена трансформације Т на оригиналну слику, па је:

Трансформацијом слике се тачке (x, y), улазне слике f, пресликавају на тачке (x’, y’), тачке излазне слике g, што се може представити као:

Примјеном просторне трансформације на овај начин за сваку тачку улазне слике израчунавају се координате и вриједности интензитета свјетлине тачака излазне слике (*Слика 6*).

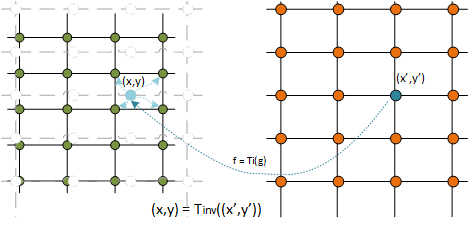


Слика 3.2: Пресликавање унапријед

Како координате тачака треба да буду цијели бројеви, док се наведеним једначинама не могу увијек добити цијели бројеви (тачке оригиналне слике се пресликавају између тачака резултантне слике, тј. вриједност улазне тачке утиче на више тачака резултујуће слике, па би се коначна вриједност интензитета сваке од тачака добијала интерполацијом), а и узимајући у обзир чињеницу да се неке тачке могу пресликавати и ван оквира излазне слике, овакво пресликавање може се сматрати неефикасним [3].

Ефикаснији метод би био тзв. пресликавање уназад, гдје би се за сваку тачку резултујуће слике израчунавале координате њеног оригинала. Да би ово било могуће потребно је познавати инверзну трансформацију поменуте трансформације Т, гдје би важило:

,



Слика 3.3: Пресликавање уназад

Сада се вриједност улазне тачке рачуна интерполацијом вриједности тачака улазне слике. На овај начин свакој тачки резултујуће слике приступа се само једном и не постоји могућност да вријеност неке од њих буде недефинсана[3].

## Преглед метода за скалирање слике

### особине

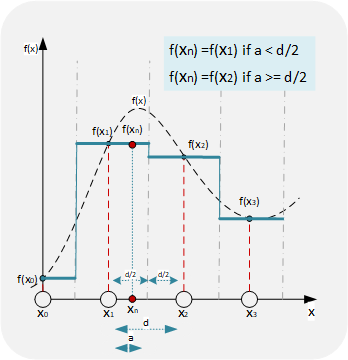
* сепаабилност
* диференцијабилност
* инверзибилност

### најближи сусјед (енг. Nearest Neighbour)

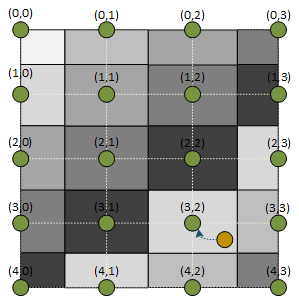
Примјеном ове методе, тачки чију вриједност интерполирамо додјељује се вриједност најближе познате сусједне тачке. На дијаграмима испод приказана је апроксимација функције једне промјенљиве кориштењем ове методе (*Слика 3.4*) и област у којој је свака од познатих тачака најближи сусјед за дводимензионалне сигнале.

Ово је алгоритам који је најједноставнији и најлакши за реализацију, јер је након одређивања позиције улазне тачке (пресликавање уназад) потребно наћи најближу познату тачку оригиналне слике, те њену вриједност додијелити резултантној тачки. Због тога је наведени алгоритам погодан за рад у реалном времену.

Међутим, примјена овог алгоритма може знатно нарушити глаткоћу ивица на сликама, па се због тога чешће примјењује за бирање боја текстуре при 3D рендеровању у реалном времену.

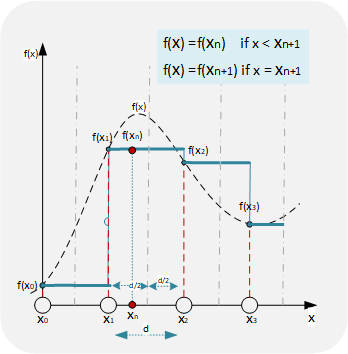


Слика 3.5: Апроксимација функције једне промјенљиве методом најближи сусјед



Слика 3.6: Интерполација дводимензионалне функције методом најближи сусјед

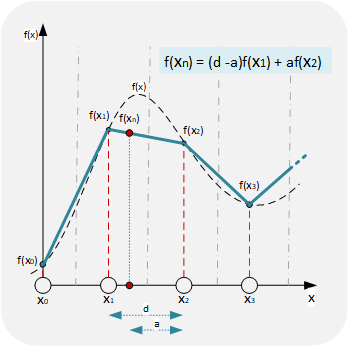
За рад у реалном времену често се овај алгоритам користи у мало измјењеном облику. Наиме, свакој тачки чију вриједност требамо интерполирати додјељује се вриједност претходне познате тачке (*Слика 3.7*), чиме се постиже каузалност функције.



Слика 3.7: Интерполација методом најближи сусјед са задршком

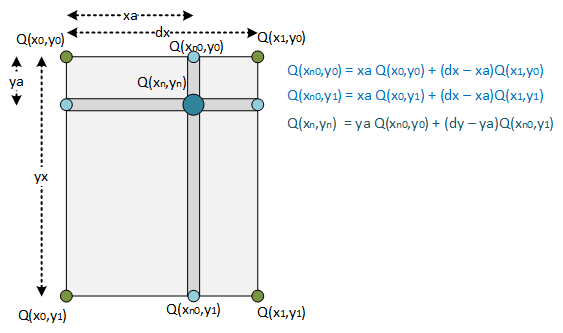
### билинеарна интерполација

Билинеарна интерполација је проширење линеарне интерполације за дводимензионалне промјенљиве, гдје се линеарна интерполација изводи по једној, а затим по другој промјенљивој. За прорачун непознате вриједности у некој тачки потребно је познавати вриједности сусједних тачака и удаљености од њих. На слици испод (*Слика 3.8*) приказана је линеарна интерполација функције f(x) на основу вриједности познатих у одређеном броју тачака.



Слика 3.9: Апроксимација функције једне промјенљиве примјеном линеарне интерполације

Као што се може видјети, линеарном интерполацијом вриједности функције између двије тачке са познатим вриједностима, интерполиране вриједности смјештене су на праву која повезује те двије тачке познатих вриједности.



Слика 3.10: Билинеарна интерполација

Билинеарна интерполација користи се за интерполацију непознатих вриједности дводимензионалних функција. Изводи се примјеном линеарне интерполације у једном, па затим у другом смјеру (*Слика 3.9*). Како је приказано на слици, прво се помјера по х оси гдје се интерполирају вриједности у тачкама (xn0, y0) и (xn0, y1), што резултује вриједностима Q(xn0, y0) и Q(xn0, y1). Затим се, помјерањем по y оси, на основу добијених резултата врши интерполацоија вриједности у тачки (xn, yn).

И ако се добија линеарном операцијом у оба смјера, сама билинеарна трансформација није линеарна, него више квадратна функција.

Билинеарна трансформација није инверзибилна.

Билинеарна трансформација је сепарабилна, може се вршити по једној, па затим по другој промјенљивој произвољним редослиједом, што је поред реалативно малог броја потребних познатих тачака чини додатно погодном за имплементацију у хардверу.

### бикубична интерполација

# Дигитални електронски уређаји

Дигитална кола су електронска кола која су базирана на ограниченом броју дискретних напонских нивоа. Припадају групи импулсних електронских кола која су заснована на прекидачком режиму рада транзистзора. Већина их користи бинарни систем и најчешће су репрезентација Булове алгебре.

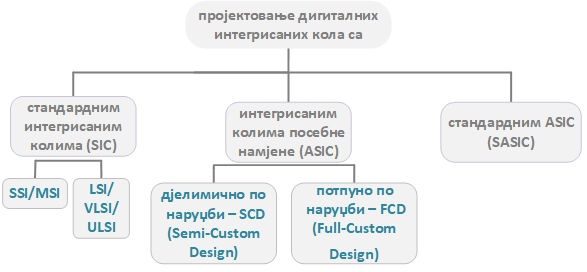
Изводе се као ТТЛ (*енг. Transistor-Transistor Logic*) кола, ЦМОС (*енг. CMOS - Complementary Metal Oxide Semiconductor*) или Би-ЦМОС кола, као комбинација ТТЛ и ЦМОС технологија, гдје се добијају кола високе улазне отпорности (утиче на мању потрошњу) као ЦМОС и мале излазне отпорности (већа максимална брзина рада) као ТТЛ кола [7].

## Методологија пројектовања дигиталних кола

Према методологији пројектовања, дигитална интегрисана кола можемо подијелити на[6][7]:

* стандардна интегрисана кола – SIC (*енг. Standard Integrated Circuits*)
* програмабилна логичка кола (ПЛД, ФПГА, ЦПЛД)
* интегрална кола посебне намјене - ASIC (*енг. Application Specific Integrated circuits*) или - USIC (*енг. User Specific Integrated circuits*)
* стандардна интегрисана кола са посебном намјеном – SASIC (*eng. Standard ASIC*)

Наведена подјела приказана је на слици испод (*Слика 4.1*).



Слика 4.1: Подјела уређаја са дигиталном логиком

Неки од критеријума који утичу на за избор кола које ће се користити су[6][7]:

* економска иплативост
* расположиве технологије
* намјена кола

За различите намјене коло се бира на основу:

* цијене по комаду
* брзине рада
* потрошње
* броја пинова кола
* да ли је коло познате функције или не
* да ли захтјева фабрикацију
* нивоа сложености тестирања

Пројектант треба добро да познаје особине уређаја који пројектује, као и особине интегрисаних кола која користи.

*Стандардна логичка кола* остварују стандардне дигиталне функције. Производе се у великим серијама, због чега им је и цијена релативно ниска. С обзиром да је функционаолност коју остварују стандардна и позната, производе се независно од крајњег корисника, тј. крајњи корисник нема утицај на процес производње ових кола. Значајан напредак у погледу испуњавања потреба крајњег корисника, када су у се ради о интегрисаним колима са стандардним логичким колима, постигнут је 1971. године, производњом микропроцесора, гдје сам корисник програмирањем врши прилагођење својим уређаја својим потребама. Могу бити произведњни као кола са малим степеном интеграције - SSI (*енг*. *Small Scale of Integration*), средњим степеном - MSI (*енг*. *Middle Scale of Integration*), високим степеном - MSI (*енг*. *High Scale Integration*), веома високим - VLSI (*енг*. *Very Large Scale of Integration*) или ултра високим - ULSI (*енг*. *Ultra Large Scale of Integration*) степеном интеграције.

*Интегрисана кола посебне намјене - ASIC (енг. Application Specific Integrated circuits)* настала су да би се рјешио или избјегао проблем са „софтверским уским грлом“, којe се појавњује код SIC кола. Наиме, без обзира на моћне софтверске алате за програмирање микропроцесора, потребе корисника некада превазилазе могућности које ти алати пружају, што доводи до тзв. „софтверског уског грла“. Интегрисана кола посебне намјене пројектована су намјенски, тако да одговарају посебним потребама корисника. Управо због обликовања кола према захтјеву корисника, ASIC кола често се називају и кола по наруџби, гдје опет можемо извршити подјелу на:

* кола потпуно по наруџби (*енг. full-custom IC*) и
* кола дјелимично по наруџби (*енг. semi-custom IC*)

Данас АСИЦ чипови могу да имају и преко 5000 логичких гејтова, чак и РАМ, РОМ, ЕПРОМ или флеш меморије, као и друге компоненте, што их чини потпуним дигиталним системом на једном чипу (*енг. System On Chip, SoC*).

Кола потпуно по наруџби производе се по шеми са потпуно описаним улазним и излазним карактеристикама, које добавља крајњи корисник. Оваквим начином пројектовања обезбеђују се најбоље електричне карактеристике кола (јер се пројектовање изводи за тачно одређену примјену и на нивоу појединачних елемената). Такође, оваквим приступом остварује се и потпуна тајност рјешења, док му је недостатак релативно висока цијена (чак и ако је поступак пројектовања и производње исти као и за SIC кола), која долази као последица малог броја уређаја који се производе (за разлику од SIC који се производе у великим серијама).

Кола дјелимично по наруџби припадају групи такозваних претпројектованих или прерпроцесних кола[7]. Добијају се повезивањем више стандардних модула које произвођач има на располагању, а на основу захтјева корисника. С обзиром да се пројектовање започиње од скоро завршеног кола које се само прилагођава крајњем кориснику, па коло може да буде лако прилагођено већем броју корисника, и цијена овако пројектованих кола је нижа него цијена потпуно по наруџби израђених кола. Ипак, у овом случају тајност рјешења је мања.

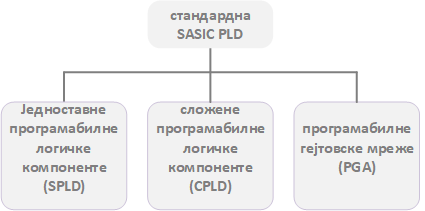
Дјелимично по наруџби дигитална логичка кола могу бити пројектована користећи се[7]:

* стандардним ћелијама (модули дефинисани на нивоу основних логичких и меморијских кола – „НЕ“, „НИ“, „НИЛИ“, флип-флопови итд.). Ово су углавном кола малог степена интеграције
* макроћелијама (моудли средњег и високог степена интеграције који се пројектују кориштењем стандардних комбинационих и секвенцијалних модула – мултиплексора, сабирача, множача, регистара, РОМ/РАМ меморија или чак и микропшроцесора) или
* гејтовским мрежама (*енг. Gate Array*) или морем гејтова (*енг. Sea Of Gate*). Гејтовске мреже- ћелије на нивоу основних гејтова су уграђени на силицијумској плочици, али нису међусобно повезани – тзв. претпроизведена кола. За повезивање су уграђени транзистори распоређени у размакнутим низовима.  
  Море гејтова – транзистори за повезивање су распоређени у неразмакнутим низовима

Стандардна интегрисана кола посебне намјене – SASIC (*енг. Standard ASIC*) су кола чија израда не зависи од крајњег корисника, тј. производе се за непознатог корисника, што је особина SIC кола, док су опет кола за посебну примјену јер их крајњи корисник програмира независно од произвођача, што је, условно речено, особина ASIC кола. У суштини ово су програмабилне логичке компоненте – ПЛД (*енг. PLD - Programmabile Logic Device*). Једноставно речено, програмабилна логички модулису дигиталне електронске компоненте имају недефинисану функцију у тренутку производње и прије употребе морају се испрограмирати да би им се дефинисала функција. Програмабилност ових кола, или прецизније речено реконфигурација, омогућена је мос транзисторима (прекидачки, меморије), осигурачима или антиосигурачима који се налазе у колу. Могу бити програмабилни једном или више пута.

САСИЦ кола се производе у различитим нивоима сложености и ту разликујемо[6][7] (*слика 4.2*):

* једноставне програмабилне логичке компоненте – СПЛД (*енг. SPLD - Simple Programmable Logic*)
* сложене програмабилне логичке мреже – ЦПЛД (*енг. CPLD – Complex Programmable Logic Devices*)
* програмабилне гејтовксе мреже – ПГА (*енг. FPGA – Programmable Gate Arrays*)



Слика 4.3 Подјела програмабилниј логичких уређаја са стаховишта сложености

ПЛД кола, за разлику од соталих АСИЦ кола, програмира сам корисник. За програмирање оваквих кола потребан је само рачунар, чиме је омогућено да сам корисник дефинише функцију и намјену кола, што га чини независним од произвођача кола.

## Програмабилни логички уређаји и њихов развој кроз историју

Програмабилни логички уређаји су интегрисана кола која корисник може да програмира према захтјевима за примјену, због чега се и сврставају у групу АСИЦ (*енг. Application specific Integrated Circuits*). Ипак, производе се за непознатог корисника, као и стандардна интегрисана кола. Због наведених особина ПЛД кола, каже се да су то стандардна интегрисана кола посебне намјене – САСИЦ (*енг. Standard ASIC*).

„Програмабилна логика је само лист на стаблу уређаја са дигиталом логиком“ [5].

Програмабилни логички моули су интегрисана кола која садрже скуп комбинационих или комбиновано- комбинационих и секвенцијалних кола, са могућношћу прорамирања њихове функције. У зависности од тога ко конфигурише (програмира) крајњу функцију модула условно их можемо подијелити у двије групе[6]:

* самоочитљиве меморије РОМ (*енг. Read Only Memory*) које конфигурише произвођач интегрисаних кола и
* програмабилне логичке компоненте – ПЛД (*енг. Programmable Logic Devices*) које програмира крајњи коросник.

Подијела није стриктна. ПРОМ уређаје програмира крајњи корисник, због чега се могу сврстати у ПЛД, међутим, области примјене су као код РОМ компонената, због ћега се често и сврставају у ту групу.

РОМ-ови су комбинациони модули и често се користе за генерисање константи, комплексних табела, аритметичких функција,микропрограмских инструкција (управљачке меморије), специјалних програма разних аутомата (машине за прање итд.) и слично[6].

ПЛД програмира крајњи корисник и у зависности од технологије програмирања дијеле се на репрограмабилне и једном програмабилне[6].

Први уређаји са програмабилном логиком били су ПРОМ (*енг.*  *Programmable Read Only Memory, PROM*), који су изумљени још 1956. године, али свој пут на широко тржиште налазе тек 1969. Убрзо се на тржишту појављује и ЕПРОМ (*енг. Erasable Programmable Read Only Memory*) 1971. Године, а затим 1975. ПЛА (*енг.* Programmable Logic Arrays, PLAs) и 1978. ПАЛ (*енг.* *Programmable Array Logic*, PAL) компоненте[8].

The PROM has a fixed AND plane or product terms which is the address decoding logic.

The OR plain or the sum terms is programmable through the change of memory contents.

This is an eight by four PROM depicted here.

Using this, we could implement the logic function output one

equals ABC by storing in the first column of the PROM the eight bit word,

0 0 0 0 0 0 0 0 0 0 1.

Then, only when A, B and C are all ones where the output, output one be a one.

In the same way, we can make the OR plane fixed and make the AND plain programmable.

In fact, this is a more efficient architecture since most of the logic functions we are interested in have a limited number of products and terms.

The following figure illustrates the general approach of a PAL which has a fixed OR plane but has a programmable AND plane.

PAL devices are very popular and they're still used in many designs, with common part numbers like 22V10 or 16R8.

Further development led to CPLDs,

which were devices with multiple PALs in

the same package with registered outputs and interconnecting programmable fabric.

A CPLD is a Complex Programmable Logic Device.

You can look at the internal link on your screen for a more complete history of the CPLD.

Here's a brief history of the FPGA.

The FPGA industry sprouted from

the programmable read-only memory (PROMs) and other programmable logic devices (PLDs).

In the 1980s, the Naval Service Warfare Center funded an experiment proposed by

Steve Castleman to develop a computer that would implement 600000 reprogrammable gates.

This began the interest in programmable logic. Altera was founded in 1983 and delivered

the industry's first reprogrammable logic device in 1984.

The EP300 which featured a quartz window in the package that allowed users to

shine an ultraviolet lamp on the dye to erase the EPROM cells that held the device configuration.

For many years, Altera touted the benefits of the CPLDs architecture over that of FPGAs,

but later relented and began making FPGAs as well. Xilinx co-founders, Ross Freeman and Bernard von der Schmidt,

invented the first commercially viable fill programmable gate array in 1985, the XC2064.

The XC2064 have programmable gates, programmable interconnects between the gates and therefore, the beginnings of a new technology and a new market.

The XC2064 had 64 Configurable Logic Blocks or CLBs with two three-input lookup tables or LUTS which are still important in FPGA architecture today.

More than 20 years later,

Freeman was entered into the National Inventors Hall of Fame for this invention.

In the early 1990s,

FPGAs were primary used in telecommunications and networking.

By the end of the decade, FPGAs found their way into consumer, automotive and industrial applications.

Programmable logic devices constitute a six billion dollar a year business, that's expected to grow to 10 billion dollars a year by 2020.

Programmable logic is just one part of the pantheon of digital logic devices.

This depiction is from Hamilton's book on the recommended reading list.

Standard logic has much more breadths than depicted here.

It includes additional TTL families like LVTTL,

additional CMOS families like HC, LVC, et cetera and biCMOS,

BCT, HCT and so on.

As well as differential logic families like ECL and LVDS.

Incidentally, all the logic types I just listed can be found in I/O interfaces in FPGAs.

And that's another example of how flexible FPGAs are.

Standard logic, programmable logic,

full custom devices are also known as general purpose integrated circuits.

In between ASICs and full custom are

devices now known as ASSPs which I would describe in a moment.

So FPGAs are simpler in concept than other PLDs.

They consist of only three elements.

A wire, a gate and a register or flip-flop.

The chip is made of an array of gates and

flip-flops with wires that can connect them together

in patterns and these patterns create the logic for larger functions like counters,

timers, state machines, ALUs,

and even whole CPUs.

Creating the interconnecting pattern is the heart of

FPGA design which we will learn in this course.

An ASSP or Application-Specific Standard Product is

a semiconductor device integrated circuit product that is dedicated to

a specific application market and sold to more than one user and thus standard.

ASSP is marketed to multiple customers just as a general purpose product

is but it's to a smaller number of customers since this is for a specific application.

Like an ASIC or an Application-Specific Integrated Circuit,

the ASSP is for

a specific application that is sold to any number of companies to an entire market.

An ASIC is designed and built to order for a specific company.

An ASSP generally offers

the same performance characteristics and has the same die size as an ASIC.

The ultimate expression of logic devices is a system on a chip.

Here are two definitions for an SOC or System on a Chip

which are becoming more and more common as time goes on.

A System on a Chip or a System on Chip (SOC) is an integrated circuit

that integrates all components of a computer into

an electronic system into a single chip.

It may contain digital, analog,

mixed-signal and other radio frequency functions on a single chip substrate.

SOCs are very common in

the mobile electronics market because of their low power consumption.

Another typical application is in the area of embedded systems.

Another definition for a System on a Chip or System on Chip is

an integrated circuit that integrates

more than one component into a single chip along with a CPU.

Typical component types are GPUs,

communication interfaces, analog functions, and radios.

If it includes programmable logic then it is a programmable SoC,

or an SoC FPGA.

The higher integration of an SoC provides lower cost,

smaller size, and lower power than alternatives.

Field Programmable Gate Arrays (FPGAs) are programmable logic devices made of gates,

registers, and routing wires connected together in

a pattern that can be programmed after the device is deployed.

Creating the interconnecting pattern is the heart of FPGA design.

Programmable Logic Devices (PLDs) include simple PLDs like PROMs and PALs,

complex PLDs (CPLDs), FPGAs, and SoC FPGAs.

PLDs are a subset of all logical devices and FPGAs are a subset of PLDs.

FPGAs compete with Application-Specific Integrated Circuits, or ASICs,

and Application-Specific Standard Products,

or ASSPs, successfully displacing them in many applications.

When it comes to digital devices,

FPGAs appear to be the future

Интерес за програмабилну логику започео је експерименталним пројектом америчке ратне морнарице (*енг. naval surface warfare center*), гдје се радило на развоју рачунара са 600000 репрограмабилних гејтова. (дај јос неки детаљ 'ѕауѕимао пола собе, мјау мјау... направио га цико сто пеца са мацком у циѕмама и тд ...)

Фирме које су се од самог почетка укључиле у трку развоја оваквих уређаја су Ксајлинкс и Алтера.

Алтера је основана 1983. године. Бла бла ... први уређај итд

Ксајлинкс ... бла бла ...

Данас су ово баш баш модерни уређаји који чуда чине.

-Убаци раѕвој кроѕ године 'коликјо гејтова, ко кад гдје ... (нека буде табела)

-убаци информацију колики се кеш обрце и колико су фирме вриједне данас ... и ова удруѕиванја помени ... као и то је ѕанимлјиво ;)

## ФПГА

ФПГА (*енг.* ***F****ield* ***P****rogrammable* ***G****ate* ***A****rray)* или низови „гејтова“ програмабилних у пољу је програмабилни дигитални уређај чијим програмирањем добијамо

**What is an FPGA?**

**Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing[https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html – posjeceno 20.03.2022].**

reprogrammability to desired application or functionality requirements after manufacturing distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks

### Област кориштења ФПГА уређаја

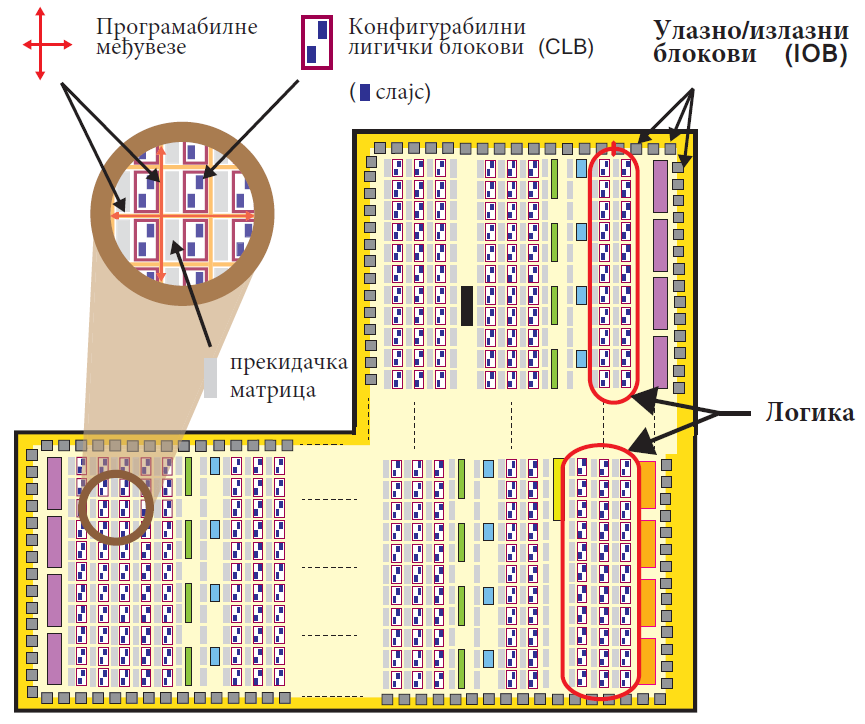
**[https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html – posjeceno 20.03.2022]. преуѕими поглавлје 2 и мјау мјау**

Due to their programmable nature, FPGAs are an ideal fit for many different markets. As the industry leader, Xilinx provides comprehensive solutions consisting of FPGA devices, advanced software, and configurable, ready-to-use IP cores for markets and applications such as:

* [Aerospace & Defense](https://www.xilinx.com/applications/aerospace-and-defense.html) - Radiation-tolerant FPGAs along with intellectual property for image processing, waveform generation, and partial reconfiguration for SDRs.
* [ASIC Prototyping](https://www.xilinx.com/applications/emulation-prototyping.html) - ASIC prototyping with FPGAs enables fast and accurate SoC system modeling and verification of embedded software
* [Automotive](https://www.xilinx.com/applications/automotive.html) - Automotive silicon and IP solutions for gateway and driver assistance systems, comfort, convenience, and in-vehicle infotainment. - [Learn how Xilinx FPGA's enable Automotive Systems](https://www.xilinx.com/training/automotive-fpga-training.htm)
* [Broadcast & Pro AV](https://www.xilinx.com/applications/broadcast.html) - Adapt to changing requirements faster and lengthen product life cycles with Broadcast Targeted Design Platforms and solutions for high-end professional broadcast systems.
* [Consumer Electronics](https://www.xilinx.com/applications/consumer-electronics.html) - Cost-effective solutions enabling next generation, full-featured consumer applications, such as converged handsets, digital flat panel displays, information appliances, home networking, and residential set top boxes.
* [Data Center](https://www.xilinx.com/applications/data-center.html) - Designed for high-bandwidth, low-latency servers, networking, and storage applications to bring higher value into cloud deployments.
* [High Performance Computing and Data Storage](https://www.xilinx.com/applications/high-performance-computing.html) - Solutions for Network Attached Storage (NAS), Storage Area Network (SAN), servers, and storage appliances.
* [Industrial](https://www.xilinx.com/applications/industrial.html) - Xilinx FPGAs and targeted design platforms for Industrial, Scientific and Medical (ISM) enable higher degrees of flexibility, faster time-to-market, and lower overall non-recurring engineering costs (NRE) for a wide range of applications such as industrial imaging and surveillance, industrial automation, and medical imaging equipment.
* [Medical](https://www.xilinx.com/applications/medical.html) - For diagnostic, monitoring, and therapy applications, the Virtex FPGA and Spartan® FPGA families can be used to meet a range of processing, display, and I/O interface requirements.
* Security - Xilinx offers solutions that meet the evolving needs of security applications, from access control to surveillance and safety systems.
* [Video & Image Processing](https://www.xilinx.com/applications/video-and-imaging.html) - Xilinx FPGAs and targeted design platforms enable higher degrees of flexibility, faster time-to-market, and lower overall non-recurring engineering costs (NRE) for a wide range of video and imaging applications.
* [Wired Communications](https://www.xilinx.com/applications/wired-wireless.html) - End-to-end solutions for the Reprogrammable Networking Linecard Packet Processing, Framer/MAC, serial backplanes, and more
* [Wireless Communications](https://www.xilinx.com/applications/wired-wireless.html) - RF, base band, connectivity, transport and networking solutions for wireless equipment, addressing standards such as WCDMA, HSDPA, WiMAX and others.

## Програмабилна логика

Програмабилна логика је базирана на Artix®-7 или Kintex®-7 FPGA серији. PL је претежно састављена од опште намјенске FPGA логике, која се састоји конфигурабилних логичких блокова (енг. *Configurable Logic Blocks -* CLB) и улазно/излазних блокова (енг. *Input/Output Blocks -* IOB) за повезивање.



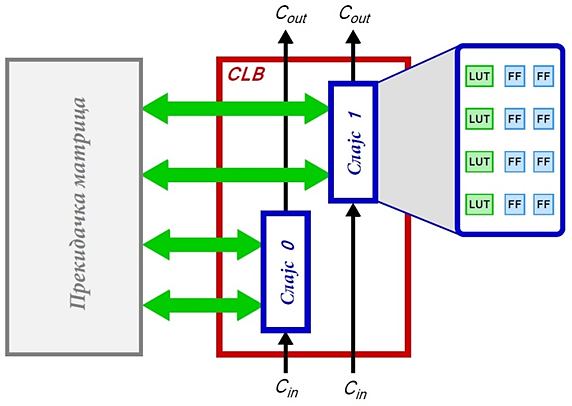
Слика 4.4 – Програмабилна логика са саставним елементима

Карактеристике PL (приказане на слици 3.5) могу се резимирати на следећи начин:

* Конфигурабилни логички блокови – CLB-ови су мале, правилне групације логичких елементи који су постављени у дводимензионалном низу на PL, и повезани са другим ресурсима преко програмабилних међувеза. Сваки CLB је постављен поред прекидачке матрице и садржи два логичка слајса, као што је приказано на слици 3.6.
* Слајс (*slice*) - Подјединица унутар CLB-а, која садржи ресурсе за имплементацију комбинационих и секвенцијални логички кола. Zynq слајс састоји се од 4 Lookup табеле, 8 флип-флопова и друге логике, као што је приказано на слици 3.6.
* *Lookup Table* (LUT) - Флексибилни ресурс који може да имплементира:
  + логичку функција до шест улаза,
  + малу ROM меморију,
  + малу RAM меморију и
  + помјерачки регистар.

LUT се могу заједно комбиновати да формирају веће логичке функције, меморије или помјерачке регистре, по потреби.

* Флип-флоп (FF) - Секвенцијални елсмент који имплементира 1-битни регистар, са ресет функцијом. Један од FF-ова може се опционо користити за имплементацију леча.
* Прекидачка Матрица - прекидачка матрица налази се поред сваког CLB-а, и пружа флексибилност способности рутирања за остваривање веза: између елмената унутар CLB-а и једног CLB-а према другим ресурсима PL-a.
* Преносна логика - Аритметичка кола захтевају да међусобни сигнали пропагирају између суседних слајсева, а то се постиже преносном логиком. Преносна логика садржи ланац рута и мултиплексере за повезивање слајсева у колони.
* Улазни/излазни блокови – IOB-ови су ресурси који омогућавају повезивање између PL логичких ресурсa и физичких приступа (енг. *peds*) који се користе за повезивање са спољним колима. Сваки IOB може да оствари 1-битни улазни или излазни сигнал. IOB-ови се обично налазе око периметра кола.



Слика 4.5 – Блок шема конфигурабилног логичког блока

### Специјални ресурси: DSP48E1 и блок RAM

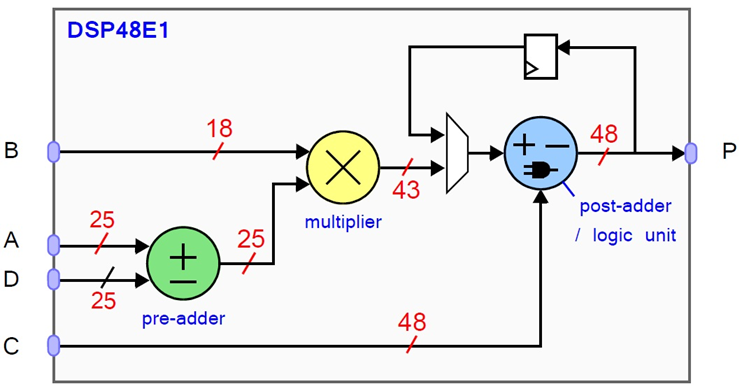
Поред опште логике, постоје и двије компоненте посебне намјене: блок RAM меморије за густе меморијске захтјеве и DSP48Е1 елементи за брзу аритметику. Блок RAM меморије у *Zynq-7000* колима су еквивалентни онима на *Xilinx* 7 серијама FPGA, и они могу имплементирати RAM меморију, ROM меморију и FIFO бафере.

Блок RAM може да сачува до 36 *kb* информација и може бити конфигурисан као један 32 *kb* RAM или два независна 18 *kb* RAM-a. Подразумијевана дужина ријечи је 18 бита и у овој конфигурацији сваки RAM се састоји од 2048 меморијски елемената. RAM такође може бити "преобликован" тако да садржи више мањих елемената (на примјер 4096 елемената × 9 бита или 8192×4 бита), или алтернативно, мање дужих елемената (нпр. 1024 елемента × 36 бита, 512×72 бита). Меморије већег капацитета могу се формирати комбиновањем заједно два или више блокова RAM-ова. Коришћење блок RAM омогућује да се велика количина података може чувати у малом физичком простору на колу, у оквиру намјенског и оптимизованог меморијског елемента. Блок RAM може да ради на максималној фреквенцији коју дато *Zynq* коло подржава.

Алтернатива блок RAM-у је дистрибуирани RAM, који је конструисан од LUT елемената. Значајан број LUT-ова (који су распоређени на већој површину) су потребни да формирају меморију сличне величине као блок RAM меморија, а резултирајућа имплементација има мање перформансе услијед логичког и пропагационог кашњења услијед дужих веза. С друге стране, често је корисно мале меморије користећи дистрибуирани RАМ, како због ефикасност ресурса, тако и због тога што је њихово постављање флексибилније (дистрибуирана меморија може бити смјештена у близини компоненти које комуницирају с њима, што може довести до бољих временских перформанси).

*Lookup* табле се могу користити за имплементацију аритметичких оператора било које поизвољне дужине, али су најпогодније за аритметичке операторе са кратким ријечима (аритметичка кола за дугачке ријечи могу заузети велики број ресурса тј. велику површину кола). DSP48E1 су специјални елементи за имплементацију брзе аритметике над сигналима средње и велике дужине. Они се првенствено састоје од сабирача, одузимача, множача и логичких елемената, као што је приказано на слици 3.7, гдје су означене максималне дужине сигнала.

На слици 3.7 представљен је поједностављени дијаграм DSP48E1 елемента. DSP48E1 користи кола за мултиплексирање како би омогућила флексибилнија употребу регистара и подржала динамичку промјену рачунања (тј. функција се може мијењати из такта у такт по потреби). Могуће су различите операције које укључују један, два или све ове аритметичке оператере, а оне се бирају путем OPMODE улаза који конфигурише интерне мултиплексере (који нису у потпуности приказани на дијаграму) и одређује аритметичке функције које се имплементирају. Улазни сигнали означени су са А, B, C и D а излазни са P. Јединица може израчунати функције P = (А + D) \* B, или P = P' + C, или заправо многе друге. Такође је способна за обраду SIMD-а, примјењујући 2 или 4 краће операције додавања/одузимања/акумулације од 24 или 12 бита, респективно.



Слика 4.6 – Архитектура DSP48E1 елемента

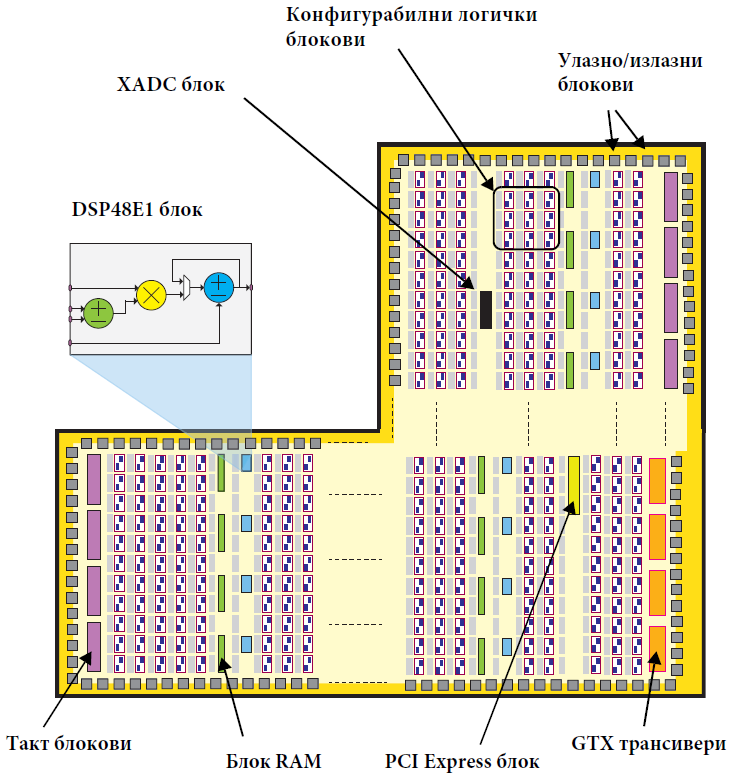
Са слике 3.7 може се примјетити да послије-сабирача/одузимач има додатну способност логичке јединице. Када се користи у логичком режиму, може вршити логичке функције уместо аритметичких и подржава све основне битске операције: НЕ, И, ИЛИ, НИ, НИЛИ, ЕКСИЛИ и ЕКСНИЛИ. Стандардне дужине аритметичких ријечи, означене на слици 3.7, су адекватне за већину захтјева, али се такођер могу проширити комбиновањем више DSP48E1-ова. Дужина ријечи је такође погодна за имплементацију аритметике са покретним зарезом. Заједно са предностима рада на високим фреквенцијама (исто као и RAM блокови, DSP48E1-ови се могу користити на максималној фреквенцији такта кола) и ниске потрошње енергије, ови DSP48E1 елементи су атрактивни и за имплементацију рачунски захтијевних аритметичких кола.

### Опште намјенски улази/излази

IOB на *Zynq* колу су колективно названи SelectIO ресурси, и они су организовани у банке од по 50 IOB-ова. Сваки IOB садржи један приступ који обезбјеђује физичку везу са спољним свијетом за један улазни или излазни сигнал. I/O банке су категоризоване као банке високе перформансе (енг. *High Performance* - HP) или банке високог ранга (енг. *High Range* - HR) и оне подржавају различите I/O стандарде и напонске нивое. HP интерфејси су ограничени на напоне од 1.8V и обично се користе као интерфејси велике брзине за везу са меморијом и другим колима, док HR интерфејси омогућавају напоне до 3.3V и пружају шири избор I/O стандарда. Подржано је и диференцијално сигнализирање, које захтијева 2 IOB по конекцији. Сваки IOB такође садржи IOSERDES ресурс за програмабилну конверзију између паралелних и серијских података (серијализација и десеријализација) од 2 до 8 бита.

### Комуникациони интерфејси

Одређена *Zynq* кола посједују и GTX трансивере (енг. *Gigabit Serial Transceiver*). То су комуникациони модули велике брзине који су уграђени у PL и повезани са одређеним пиновима кола. Ови модули омогућавају серијску комуникацију од неколико Gbps. Коришћењем ових модула омогућава се имплементација различитих комуникационих интерфејса: PCI Express, Serial RapidIO, SCSI и SATA. За PCI Express комуникацију користе се још посебни уграђени блокови (слика 3.8).



Слика 4.7 – Програмабилна логика

### Други програмабилни логички спољни интерфејси

Програмабилна логика садржи и аналогно дигиталне конверторе, XADC блок. Овај блок садржи два одвојена 12-битна ADC-а, оба способна за одмјеравање екстерних аналогних улазних сигнала брзином од 1 *Msps*. Управљање XADC-ом се постиже коришћењем PS-XADC интерфејсног блока лоцираног унутар PS-а, а PS-XADC управљачки блок се може програмирати из софтвера који се извршава на APU.

Као извор такт сигнала за програмабилну логику може се користити четири сигнала која долазе од PS система. Такође такт сигнал се може генерисати коришћењем уграђених PLL (енг. *Phase Locked Loop*) модула који референтни такт могу да добију кроз одређене пинове кола.

За конфигурисање и отклањање грешака (debugging) у програмабилној логици користи се посебан модул који је преко JTAG (енг. *Join Test Action Group*) интерфејса повезан на развојини рачунар.

# Реализација видео скалера у ФПГА технологији

Као компромис између квалитета резултујуће слике и потребних ресурса, између алгоритама погодних за хардверску реализацију и рад у реалном времену, за реализацију видео скалера одабрано је да се користи интерполација слике кориштењем алгоритма билинеарне трансформације са пресликавањем уназад, *Слика 3.2, Слика 3.9*. Уређај је пројектован тако да може лако да се увеже у ланац других уређаја, гдје би се, на примјер, лако надовезао на сличан модул за обраду. Одабрано је да се слика добавља и прослијеђује кориштењем Акси стрим протокола (*енг.* Axi Stream), који је често кориштен у пракси и може да омогући слање и примање потребне количине података довољно великом брзином (за рад са сликама димензија које се користе у раду).

Diagram

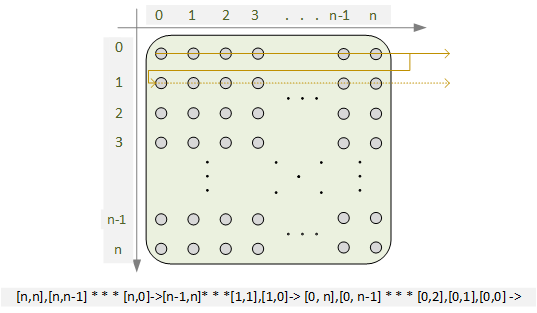
Description automatically generated

Слика 5.1: Видео скалер, блок дизајн

Комуникација између модула у дизајну обављаће се користећи хендшејк (*енг.* Handshake) принцип или принцип руковања, гдје се информација од пошиљаоца према примаоцу шаље тек након што се добије потврда о пријему претходно послане информације, *Слика 5.2*.

## Протокол за комуникацију видео скалера са околином, Акси Стрим (*енг.* AXI STREAM) протокол

За рад скалера битно је изабрати одговарајући начин довођења података на уређај. По угледу на сличан дизајн за ФПГА платформу фирме „Xilinx“ [1], усвојићемо да слика (*енг.* *frame*) на скалер долази преко стрим (*енг.* *stream*) интерејса, пиксел по пиксел, почевши од горњег лијевог угла слике, крећући се у десно до краја (први ред), а затим се прелази други ред и на исти начин до краја последње тачке слике, како је приказано на слици *Слика 5.2*.

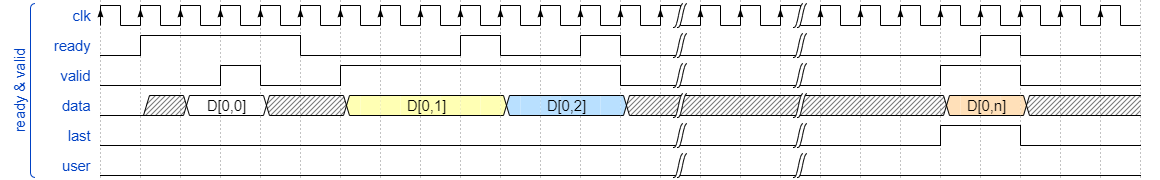


Слика 5.2: Редослијед слања пиксекла преко стрим интерфејса

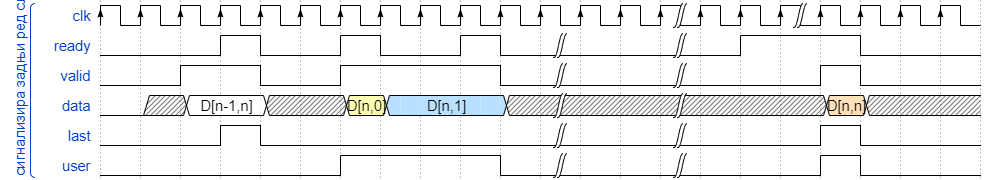
За слање слика користи се стрим протокол који садржати информације опикселу:

* вриједност свјетлине или интензитет пиксела
* сигнал за почетак нове слике
* информацију да ли се ради о последњем податку у реду слике
* информацијуо ваљаности пиксела

Са пријемне стране очекује се информација да ли је модул спреман да преузме податке који се тренутно налазе на линији.



Слика 5.3: Приказ сигнала стрим протокола, исправан податак на линији(valid), спреман за преузимање податка (ready), задњи податак из тренутног реда(last):

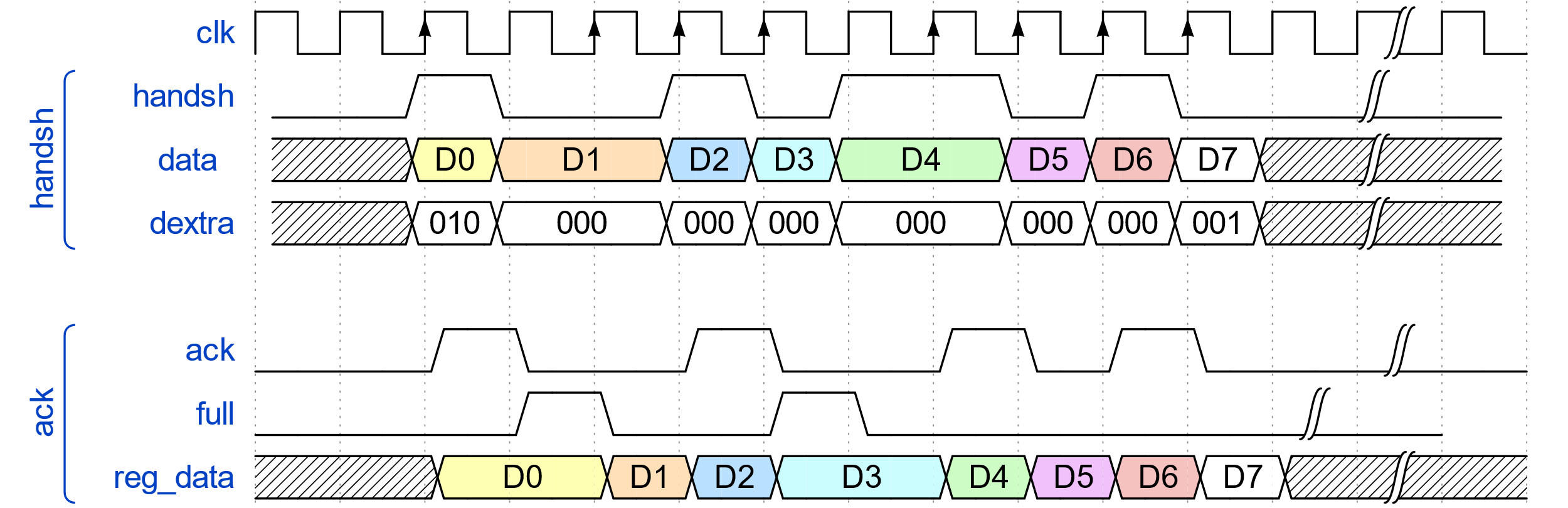


Слика 5.4: Као сигнализација да се шање последњи ред слике користи се сигнал "user". Последњи податак је послан када су и user и valid на високом нивоу

Податак се преутима када су оба сигнала , и „ready“ и „valid“, на високом нивоу. Након што је „valid“ једном постављен на висок логички ниво, податак се не смије мијењати све док га друга страна не преузме, тј. све док сигнал „ready“ не буде на високом логичком нивоу[9]. Ако се шаље задњи податак из реда, сигнал „last“ ће бити постављен на висок логички ниво до преузимања податка (*Слика 5.2*)., док ће сигнал „user“ бити кориштен да сигнализација да се ради о новој слици.

## Протокол за комуникацију између модула у дизајну

Комуникација међу модулима остварује се користећи хенд-шејк (енг. Handshake) начин комуникације. То је начин комуникације заснован на принципу руковања, гдје пошиљалац након слања податка очекује потврду да је податак успјешно прихваћен, а онда, прије слања наредног податка, очекује сигнал да је пријема страна спремна да га прихвати. Комуникацију започиње пошиљалац постављањем податка на линију података и мијењањем тренутног стања хандшејк сигнала. На пријемној страни, као потврда да је податак примљен, сигнал ацк се поставља на вриједност коју има сигнал хендшејк, али за наставак преноса података потребна је не само потврда да је податак примљен, него и потврда да је пријемна страна спрема да прихвати нови податак. Да би се остварио пренос наредног податка, осим што сигнал потврде („ack“) треба да преузме вриједност „handsh“ сигнала и сигнал фул (*енг. full*), који означава да је пријемна страна заузета ако је на високом нивоу, треба да има вриједност 0.



Слика 5.5: Протокол за комуникацију међу модулима кориштеним за реализацију видео скалера

handsh – сигнали којима управља пошиљалац:

* handsh – наредба пријемнику да преузме податак са линије података
* data – линија података
* dextra – додатне информације битне заа обраду (почетак нове слике, крај реда слике, важећи податак)

ack – сигнали обавјештења:

* ack - прималац обавештава да је податак примљен
* full - информација пошиљаоцу да је претходни податак прослијеђен даље (full = 0) и да је пријемник спреман за следећи податак

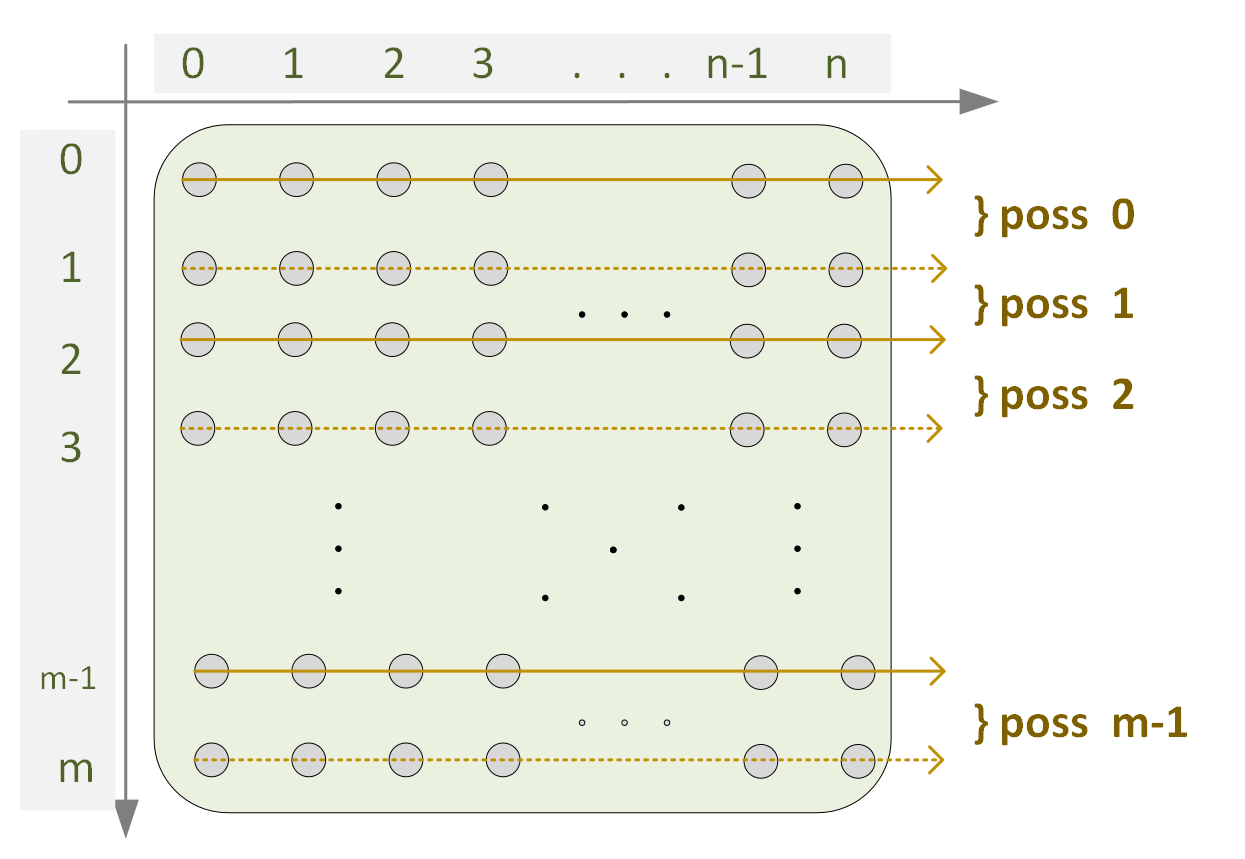
Заједно са подацима и наредбом за пријем податка, хенд-шејк сигнал, пошиљалац прослијеђује и информације о слици (*сигнал dextra*), корисне за њену даљу обраду, као што су:

* информације о почетку нове слике
* информације о последњем пикселу у једном реду слике
* информацију да је пиксел важећи

По потреби, користећи „dextra“ и друге информације могу бити пренешене заједно са наведеним подацима, а које ће то информације бити зависи од тога гдје се у дизајну користи описани начин преноса података.

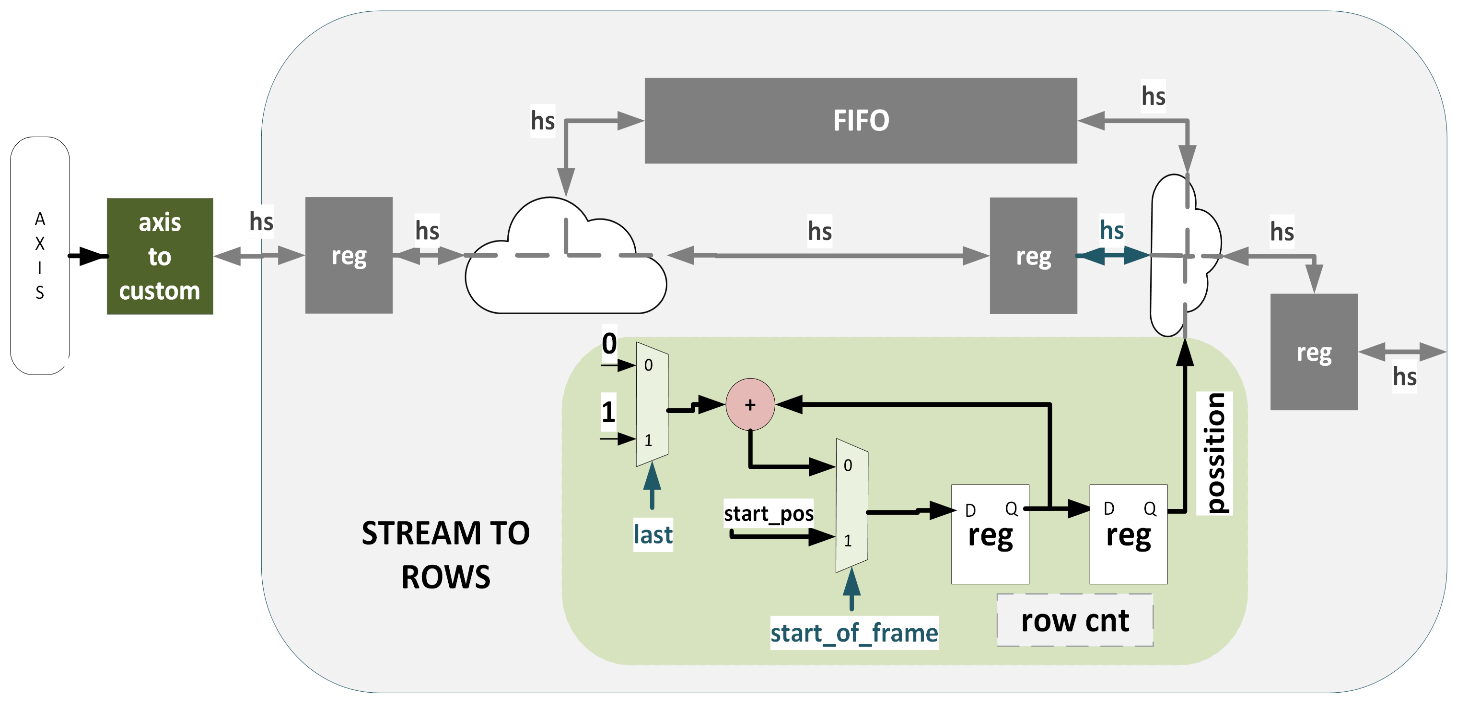
## Прилагођење улазног тока података на филтер (енг. Stream to rows) модул

Прилагодни модул ради накупљање (*енг. buffering*) података, како би ток прилагодио наредном степену, што је у овом случају модул за бертикалну промјену димензије слике, тј. билинеарни вертикални филтар,. Од прилагодног степена затијева се да наредни степен снабдијева са по два пиксела, сусједна по вертикалној оси, уз информацију о позицији коју заузима посматрани пар пиксела, *Слика* *5.6*.



Слика 5.6: Информација о позицији реда слике

С обзиром да се један ред почетне слике користи у креирању парова за двије позиције излазу, потребно је сваки од редова почетне слике (од 0 до m) пропусти као члана у креирању једне позиције и складишти, да би могао учестовати у креирању парова за следећу. Нпр, ред 1 почетне слике пропушта се на излаз гдје има учешће у пару пиксела чија је позиција 0, а истовремено се складишти да би заједно са редом 2 учетовао у креирању парова пиксела чија је позиција 1. Дизајн који обавља описану улогу приказан је на *Слика 5.7*. Модул назван „axis\_to\_custom“ прихвата податке са акси стирм (*енг. Axi Stream*) интерфејса, препакује их и послијеђује даље по протоколу који се користи у дизајну за комуникацију међу модулима. Модул „axis\_to\_custom“ издвојен је и лако замјенљив, да би се омогућила лака и једноставна прилагодба видео скалера у случају потреба да са окружењм комуницира користећи неки други протокол умјесто акси стрима. Подаци се прихватају у регистар који је смјештен на улаз, а из кога се прослијеђују истовремено на „FIFO“ модул, у ком се складиште, и на следећи регистар. Додатна логика спаја излаз ФИФО модула и регистра у један податак, те заједно са информацијом о вриједности свјетлине пиксела, позицији пара пиксела који се прослијеђује, прослијеђују се и додатним информацијама везане за тај пар, као што су информација о почетку нове слике или информација о последњем пару пикселу за позицију који се тренутно шаље ка наредном модулу. Рачунање позиције пара пиксела обавља логика издвојена зеленом подлогом од осталих модула на истој слици.



Слика 5.7: Модул за прилагођење стрима на билинеарни филтар

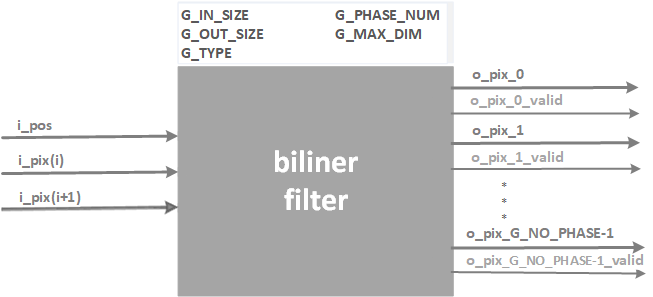
Позиција се увећава за 1 уколико прихватамо последњи пиксел реда оригиналне слике, који није нулти ред (лијеви мултиплексер на слици), док се враћа на почетну вриједност у случају сигнала за почетак нове слике (десни мултиплексер).

## Билинеарни филтар

Билинеарни филтер обавља рачунање вриједности интензитета тачака резултујуће слике, примјеном билинеарне интерполације са пресликавањем уназад, на основу вриједности пара пиксела са почетне слике и информација о позицији тих пиксела. За исправан рад модула потребне су следеће информације:

* вриједности интензитета двије сусједне тачке по посматраној оси почетне слике
* позиција реда/колоне из кога потичу пиксели које имамо на располагању (ако се филтер користи за вертикалну промјену димензија потребна је информација о реду из којег добијамо пикселе, док у случају хоризонталне промјене димензија требамо информацију о колони)
* информације о димензијама реда или колоне (у зависности да ли се филтер користи за вертикалну или хоризонталну промјену димензије слике) оригиналне и резултујуће слике. Добијају се као параметри, а служе да би се могао израчунати фактор скалирања по посматраној оси
* минимална и максимална димензија резултујуће слике (потребна да би се поставиле одговарајуће ширине података унутар модула)
* максималан број фаза модула (максималан број излазних пиксела које у једном такту можемо прорачунати на основу тренутних улаза и добити на излазу модула, а истовремено и максималан број тачака на које можемо процијенити позицију оригиналног пиксела, за резултујући пиксел чија се вриједност рачуна)

Да би операција промјене димензија слике могла да се извршава довољно брзо, да се при томе не успорава нормалан ток података, а да се истовремено избјегне велико повећање радне фракевенције уређаја, филтар је реализован тако да на свом излазу може да да више од једног резултујућег пиксела за пар пиксела доведених на његов улаз. Максималан број пиксела које je могуће добити истовременона излазу модула одређен је параметром филтра, означеним са G\_PH\_NUM. Улазни и излазни портови модула, те параметри, прикани су на *Слици 5.8.*



Слика 5.8: Улази, излази и параметри модула који обавља функцију билинеарне интерполације или филтрирања

Параметри филтра су:

* величина оригиналне слике по посматраној оси, G\_IN\_SIZE
* величина резултујуће слике по посматраној оси, G\_OUT\_SIZE
* тип филтра, вертикални или хоризонтални, G\_TYPE
* број фаза, одговара максималном броју излазних пиксела филтра у једном такт циклусу, а истовремено и максималном броју тачака између два пиксела оригиналне слике које се могу процијенити као тачка оригинала за резултујући пиксел - G\_PH\_NUM
* максимална димензија резултујуће слике, потребна за рачунање броја такт циклуса потребних да се добију све резултујуће вриједности које потичу од једног пара пиксела на улазу модула, G\_MAX\_DIM

У случају да улазни пар пиксела није потпуно експлоатисан у једном такт циклусу, тј. да се од пиксела на улазу добија више од G\_PH\_NUM резултујућих пиксела, претходном модулу се сигнализира да је информација примљена („ack“ сигнал на пријемној страни поприма вриједност „handsh“ сигнала, али „full“ остаје постављен на вриједност 1 све док се пиксели на улазу не експлоатишу у потпуности). Да би се, гледајући са излазне стране модула, знало да се ради о пикселима који потичу од истог пара на улазу информација о редном броју пакета који је у тренутку доступан на излазу, напиши име сигнала, прослијеђује се наредном степену. Сигнал има јединицу на најнижем биту када је у питању прво читање, а мијења своју позицију помјерајући се ка вишим битима на свако наредно читање резултата ако се ради о пикселима који потичу од истог пара. Сигнал о ваљаности сваког од резултујућих пиксела, „o\_pix\_valid“, прослијеђује се као сингал хендшејк интерфејса, гдје је постављен на 3. позицију у „dextra“ податку, док је на слици посебно истакнут јер носи битну информацију за наредни степен у дизајну. Суштински, вертикални и хоризонтални филтар не разликују се у начину на који дају резултантне вриједности пиксела. Разлика почива у постављању додатних сигнала, као што су сигнал за почетак нове слике, сигнал за последњи пиксел у једном реду слике, те наилазимо на разлику у начину рада уколико пиксели са улаза нису у потпуности искориштени у једном циклусу, што ће бити описано у даљем раду.

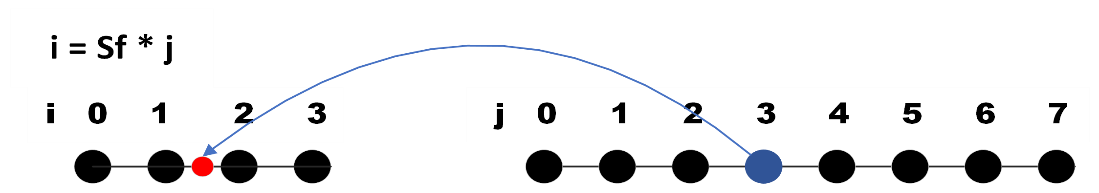
### Одабир алгоритма за реализацију

* Opisi downscale
* Пиши може се раставити
* Брз
* Не много ресурса
* Опиши банк сел као хандсејк изнад

Билинеарна трансформација Реализација растављањем на вертикални и хоризонтални филтер по угледу на ксајклинксов (*енг.* Xilinx) видео скалер, верзија 8.1[1].

### Рачунање координата оригинала за пикселе на резултујућој слици

До изведбе билинеарне трансформације са пресликавањем уназад у ФПГА технологији, анализe и пресликавањa теоријских захтјева у дигитални дизајн, долази се кроз неколико корака. Најприје, потребно је пронаћи начин да се пронађе оригинал, тј. позиција или координате пиксела на оригиналној слици који се пресликава у пиксел резултујуће слике. Позиција оригинала не мора да буде цијелобројна вриједност. Она може да буде реалан број, који предстаља позицију између пиксела оригиналне слике, *Слика 5.9*, а за коју се вриједност свијетлине пиксела процјењује на основу његовог окружења. Позицију са које „потиче” пиксел на посматраним координатама резултујуће слике добија се множењем његове координате са фактором скалирања по одговарајућој оси, *Слика 5.11*.



Слика 5.10: Проналажење координате оригинала за пиксел резултујуће слике

Ако се усвоји да је растојање између било која два сусједна пиксела на оригиналној слици једнако d, *Слика 5.12*, тада се позиција оригинала добија као збир цјелобројне координате пиксела са оригиналне слике, i (пиксел који на посматраној оси заузима позицију са мањом координатом од два посматрана пиксела) и удаљености од њега, d0. Према наведеном, координата позиције оригинала за резултујући пиксел на позицији ј може се представити као:

гдје је фактор скалирања по одређеној оси, једнак односу димензије оригиналне и резултујуће слике по посматраној оси:

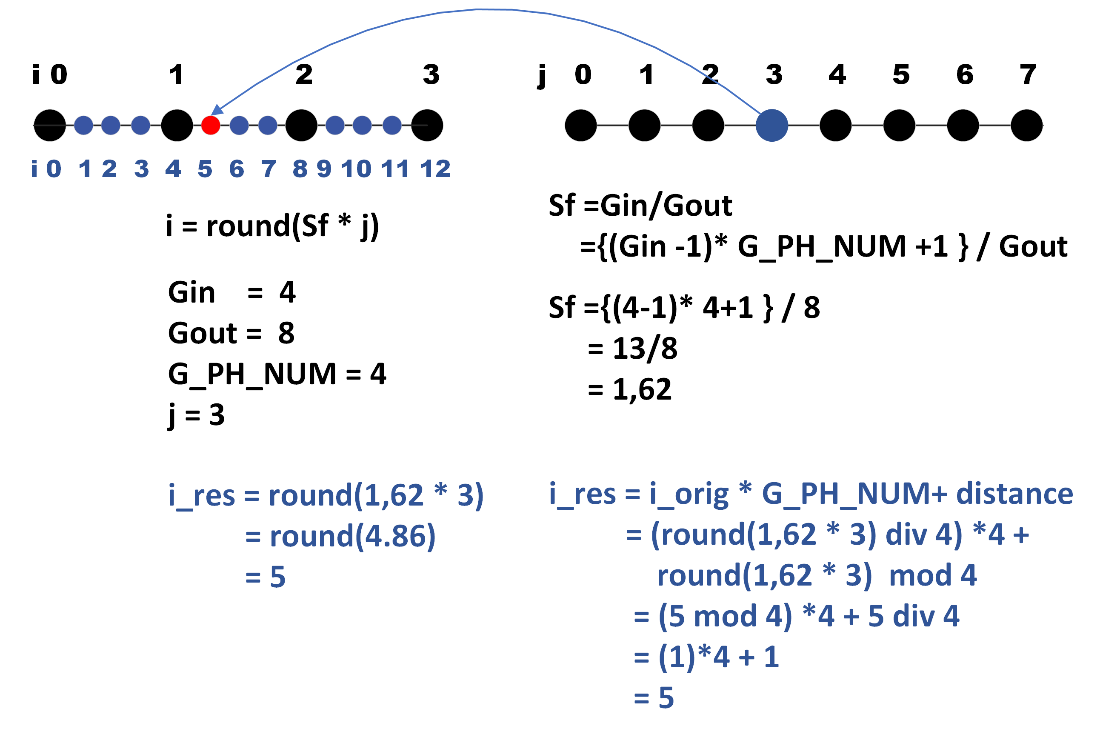
Graphical user interface

Description automatically generated with medium confidence

Слика 5.11: Растојање међу пикселима са оригиналне слике и растојање уметнутих позиција, позиција на којима се налази оригинал за конкретан резултујући пиксел

Позиција пиксела оригиналне слике са мањим индексом (од два најближа пиксела на посматраној оси) и одстојање од њега, респективно, добијају се на следећи начин:

Како се између двије тачке, на реалној бројној оси, налази бесконачан број тачака, док у хардверској представи броја таква представа није потпуно могућа, а често није ни потребна, те због уштеде ресурса, а по угледу на Xilinx-ов дизајн[1] уређаја сличне намјене, број тачака између два сусједна пиксела оригиналне слике биће ограничен на коначан број, G\_PH\_NUM. У том случају фактор скалирања једнак је односу броја пиксела оригиналне и резултујуће слике по посматраној оси, илустровано на слици *Слика 1.11,* па имамо да је:

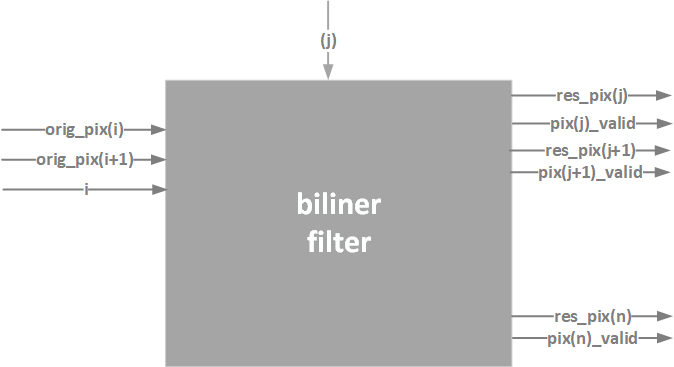


Слика 5.12: Рачунање оригинала за пиксел са резултујуће слике. Информација о најближем пикселу са оригиналне слике и удаљености оригинала од њега

На *Слика 5.5* приказан је примјер проналажења позиције оригиналног пиксела за пиксел резултујуће слике, гдје значајне величине имају следеће вриједности:

* координата посматраног пиксела на посматраној оси резултујуће слике, ј = 3
* димензија оригиналне слике по посматраној оси, Gin = 4
* димензија резултујуће слике посматраној оси, Gout = 8 и
* број могућих удаљености позиције оригинала резултујућег пиксела од сусједног пиксела оригиналне слике, G\_PH\_NUM = 4 (удаљености 0, 1, 2 и 3)

Пиксел који се у наведеном примјеру пресликава на координату 3 резултујуће слике налази се на координати 5 проширене мреже оригиналне слике, гдје је реална оса на којој би добили резултат дискретизована и свака 4. позиција почевши од нулте садржи оригинални пиксел, док су остале позиције убачене и вриједност пиксела на њима добија се интерполацијом сусједних пиксела оригиналне слике. Такође, процјењена позиција оригинала за пиксел са координатом 3 на резултујућој оси може се посматрати и као позиција на удаљености 1 од пиксела са координатом 1 на оригиналној слици. С обзиром да код тока података какав се захтијева од дизајна на његовом улазу постоји инфорација о вриједности пиксела са оригиналне слике и његове координате, па на основу тога да се провјери на којим позицијама резултујуће слике пиксели потичу од пара пиксела на улазу, те да се израчунају њихове вриједности, поменута представа процјењене позиције оригиналног пиксела ће бити од значаја за реализацију дизајна.



Слика 5.13: Билинеарни филтр са улазним и излазним сигналима

i - позиција оригинала посматраног резултујућег пиксела, orig\_pix - вриједност пиксела са оригиналне слике, res\_pix- резултујући пиксел, pix\_valid - резултујући пиксел је важећи

Посматрајући промјену димензија слике са улаза у модул, на основу пара пиксела са оригиналне слике, чији је број реда/колоне , рачунају се вриједности резултујућих пиксела на позицијама за које вриједи да је:

гдје је:

* ј - позиција првог пиксела резултујуће слике чија вриједност се рачуна полазећи од пара пиксела на улазу модула
* к – број из скупа природних бројева проширеног са нулом
* – фактор промјене димензије
* број могућих удаљености позиције оригинала резултујућег пиксела од сусједног пиксела оригиналне слике

Ако се усвоји да се вриједности координата слике по посматраној оси крећу од 0 до Gdim -1, гдје је Gdim димензија слике по посматраној оси, да би се процјена вриједности пиксела на почетној координати у резултујућем координатном систему радила од пара пиксела са позицијом 0, а процјена вриједности пиксела са највишом координатом на посматраној оси резултујуће слике радила на основу пара пиксела са оригинала чија је позиција m -1 (*Слика 5.6*.), потребно је извршити прилагођење фактора за промјену димензије слике:

Прорачун координата оригинала за резултујући пиксел своди се на то да се на основу познате позиције резултујућег пиксела и фактора за промјену димезије слике ради процјена да ли он потиче од пиксела на улазу у модул, чија је позиција на оригиналној слици такође позната. Пиксел на посматраној позицији ј резултујуће слике потиче од пара пиксела позиције i на улазу ако вриједи да је:

Ако је наведени услов испуњен удаљеност позиције, на коју се процјењује оригинал за посматрани пиксел, од пиксела оригиналне слике са позицијом i, добија се као:

#### Примјер прорачуна позиције оригинала у случају увећавања димензије по посматраној оси

Примјер прорачуна позиције оригинала за резултујући пиксел на позицији ј, када је Gin = 3, Gout = 7 и G\_PH\_NUM = 4, приказан је табеларно, у *Табела 5.1*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| j | rnd(j \* Sf)  Sf = 1.17 | Бинарна представа резултата 8b | | Координата пиксела на оригиналној слици | Удаљеност од пиксела на оригиналној слици |
| div G\_PH\_NUM | mod G\_PH\_NUM |
| 0 | 0 | 000000 | 00 | 0 | 0 |
| 1 | 1 | 000000 | 01 | 0 | 1 |
| 2 | 2 | 000000 | 10 | 0 | 2 |
| 3 | 3 | 000001 | 00 | 1 | 0 |
| 4 | 5 | 000001 | 01 | 1 | 1 |
| 5 | 6 | 000001 | 10 | 1 | 2 |
| 6 | 7 | 000001 | 11 | 1 | 3 |

Табела 5.1: Примјер рачунања оригинала конкретног пиксела резултујуће слике

Из табеле се види да се пиксели на резултујућим позицијама 0,1 и 2 добијају од пара пиксела чија је позиција 0 на оригиналној слици (пиксели са координатама 0 и 1), док се резултујући пиксели са координатама 3, 4, 5 и 6 добијају од пара пиксела чија је позиција 1 (пиксели са координатама 1 и 2). Уколико усвојимо да је G\_PH\_NUM увијек степен броја 2, тј. G\_PH\_NUM = 2k, изведба операције цјелобројног дијељења и остатка при дијељењу се много поједностављује.

Diagram

Description automatically generated

Слика 5.14: Схема модула за рачунање удањености оригинала од пиксела са оригиналне слике. Избор коефицијента филтра

У том случају резултат операције цјелобројног дијељења и остатка при дијељењу своде се само на подјелу бинарног записа дјљеника на виши и нижи дио. У горњем дијелу, од бита са највишом позицијом до позиције log2(G\_PH\_NUM) садржан је резултат цјелобројног дијељења са G\_PH\_NUM дјелиоцем, док је у доњем дијелу садржана информација о остатку при дијељењу.Из табеле се такођевиди да су оригинали пиксела који потичу од пара са позицијом 0 на улазу на удаљеностима 0,1 и 2, респективно, од пиксела који чини пар пиксела на улазу и има мању координату по посматраној оси на оригиналној слици. Пиксели 3, 4, 5 и 6 потичу од пиксела чија је позицијоа на оригиналној слици 1, а налазе се на удаљеностима 0, 1, 2 и 3 од пиксела који припада пару на улазу модула, а има мању вриједност координате на оригинаној слици.

#### Примјер прорачуна позиције оригинала у случају смањивања димензије по посматраној оси

Примјер прорачуна позиције оригинала за резултујући пиксел на позицији ј, када је Gin = 7, Gout = 3 и G\_PH\_NUM = 4, приказан је табеларно, у *Табела 5.2*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| j | rnd(j \* Sf)  Sf = 11.5 | Бинарна представа резултата 8b | | Координата пиксела на оригинлној слици | Удаљеност од пиксела на оригиналној слици |
| div G\_PH\_NUM | mod G\_PH\_NUM |
| 0 | 0 | 000000 | 00 | 0 | 0 |
| 1 | 11 | 000011 | 00 | 3 | 0 |
| 2 | 23 | 000101 | 11 | 5 | 3 |

Табела 5.1: Примјер рачунања оригинала конкретног пиксела резултујуће слике

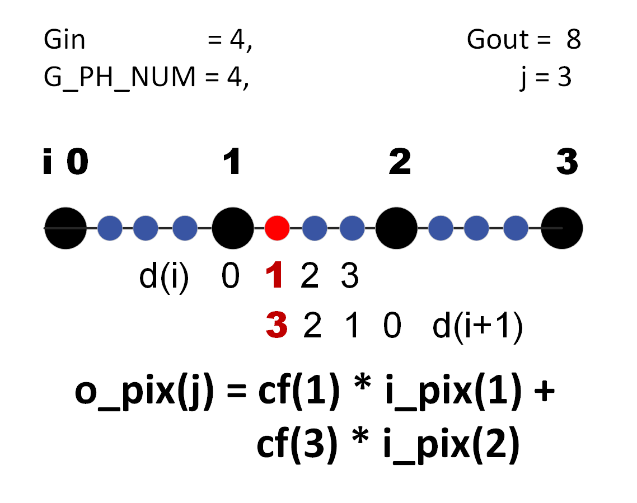
Из табеле се види да је оригинал резултујућег пиксела на позицији 0 пар пиксела чија је позиција на оригиналној слици такође 0. Пиксел са координатом 1 резултат је обраде пара пиксела са позицијом 3, а његов оригинал процјењен је да се налази на удаљености 0 од пиксела чија је координата 3 на оригиналној слици. Резултујући пиксел на позицији 2 резултујуће слике потиче од пара пиксела чија је позиција 5, а његов оригинал налази се на удаљености 3 од пиксела оригиналне слике са координатом 5. Када се на улазу појаве пиксели са позицијом која није наведена у табели, проналазимо да нити један пиксел на излазу се не прорачунава на основу пиксела који су у том тренутку на улазу, па се они не користе, него се претходном степену сигнализира да припреми седећи пар пиксела.

### Рачунање резултујућег пиксела на основу познавања вриједности свјетлине и координата оригиналног пиксела

Да би се дошло до вриједности свјетлине пиксела на резултујућој слици потребно је:

* процјенити позицију оригиналног пиксела на почетној слици
* процијенити вриједност свјетлине оригиналног пиксела, која је резултат утицаја сусједних пиксела са оригиналне слике
* примјенити функцију пресликавања на оригинални пиксел, да се добије вриједност свјетлине резултујућег пиксела

Процјена позиције оригинала описана је раније, док када се ради о скалирању слике, израчуната вриједност свјетлине оригиналног пиксела директно се додјељује резултујућем пикселу, па остаје да се опише корак рачунања вриједности свјетлине оригиналног пиксела (а то је уједно и вриједност резултујућег). Kада се ради о билинеарној трансфорамцији oна се рачуна процјеном утицаја два пиксела сусједна оригиналном. Вриједности свјетлине сусједних пиксела множе се фактором њиховог утицаја на посматрану позицију, па је коначно вриједност свјетлине оригиналног пиксела једнака збиру помножених вриједности. Фактор утицаја зависи од растојања пиксела који припадају оригиналној слици и оригинала, гдје утицај сваког од пиксела опада линеарно са растојањем од њега, а престаје наиласком на следећи пиксел који постоји на оригиналној слици. Максималан утицај који један пиксел може да има на оригинал остварује се ако се позиција оригиналног пиксела поклопи са позицијом пиксела на почетној, оригиналној, слици, а његов коефицијент утицаја једнак је 1 у том случају. Такође, у том случају утицај свих осталих пиксела на вриједност свјетлине оригиналног пиксела једнак је нули.



Слика 5.15: Рачунање резултујућег пиксела на основу утицаја сусједних пиксела оригиналне слике

Примјер израчунавања резултујућег пиксела приказан је на *Слика 5.16*, за случај када:

* оригинална слика има 4 пиксела на посматраној оси
* максимално 4 позиције између од једног до другог сусједног пиксела оригиналне слике, G\_PH\_NUM=4
* оригинал се налази на растојању 1 од пиксела 1 на оригиналној слици, тј. на растојању 2 од пиксела са координатом 2 на оригиналној слици

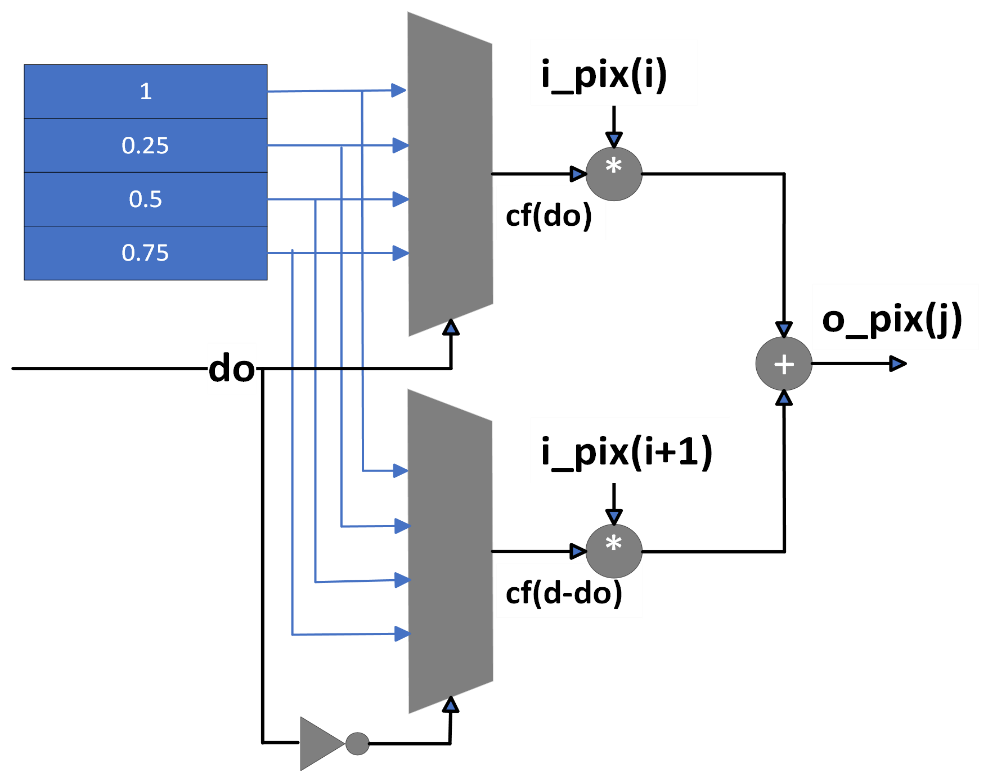
Ако се узме да је удаљеност два пиксела на оригиналној слици d=1, и усвојимо да утицај сваког пиксела опада линеарно са растојањем од њега, тада се утицај пиксела са оригиналне слике на позицију на удаљености do од њега, добија као:

Утицај пиксела са оригиналне слике или кооефицијент утицаја, за случај када је G\_PH\_NUM = 4, дат је у следећој табели:

|  |  |
| --- | --- |
| do | cf |
| 0 | 1.0 |
| 1 | 0.66 |
| 2 | 0.33 |
| 3 | 0.0 |

Табела 5.2:Утицај пиксела са оригиналне слике или кооефицијент утицаја, за случај када је G\_PH\_NUM = 4

Модел дизајна који на основу удањености оригинала од пиксела са лијеве стране на оригиналној слици, do, користећи кооефицијенте утицаја који су смјештени у меморији, рачуна вриједност резултујућег пиксела, приказан је на *Слика 5.17*.



Слика 5.16: Модел дизајна за рачунање резултујућег пиксела на позицији ј,

на основу познавања позиције оригинала - i, вриједности пиксела са оригиналне слике који се налазе са лијеве и десне стране оригинала на посматраној оси, i\_pix(i), i\_pix(i+1), удаљености оригинала од пиксела са лијеве стране do и познавања фактора утицаја пиксела са оригиналне слике за одговарајућу удаљеност

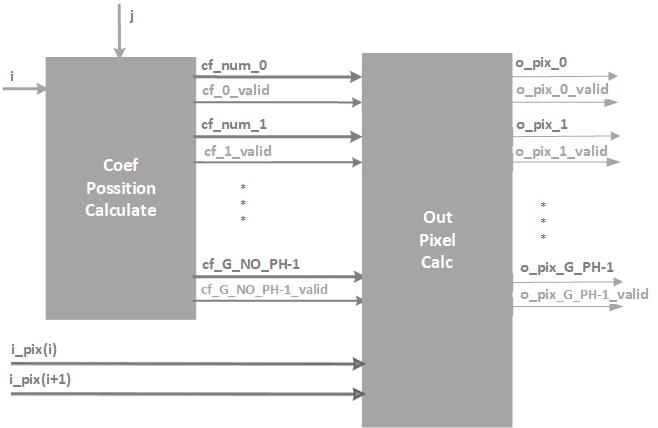
Ако је параметар G\_PH\_NUM изабран тако да буде степен броја два, тада се растојање оригинала од пиксела i\_pix(i) може представити користећи log2(G\_PH\_NUM) бинарних цифара. У том случају (d-d0), потребно за избор коефицијента са којим се множи i\_pix(i+1), добија једноставним инвертовањем бинарног записа d0. Узимајући у обзир да је инвертовање бита операција која се лакше изводи од операције одузимања у дигиталној техници, овакав избор G\_PH\_NUM параметара и у овом случају доприноси уштеди ресурса при реализацији оваквог модула.

### Изведба билинеарног филтра

Реализација билинеарног филтра заснива се на реализацији два основна модула за остварење операција описаних на *Слика 5.10*, операција за израчунавање

* везано за избор коефицијената филтра
* вриједности резултујућих пиксела

Избор коефицијента заснва се на провјери да ли се оригинал резултујућег пиксела налази у сусједству пиксела са координатама које имају пиксели оригиналне слике доведени на улаз модула, координатама *i* и *i+1*. У случају увећавања слике, од пара пиксела са оригиналне слике може да потиче више од једног резултујућег пиксела. Посматрано са стране брзине рада филтра, пожељано је да се истовремено може добити што већи број резултујућих пиксела на излазу модула. Истовремено са становишта потребних ресурса за имплементацију, с обзиром да се за добијање једног резултујућег пиксела ради избор коефицијената, двије операције множења и сабирање, усваја се компромисхо рјешење, гдје се у једном циклусу на излазу филтра може добити максимално G\_PH\_NUM резултујућих пиксела. У случају потребе да се од једног пара пиксела оригиналне слике добије више од G\_PH\_NUM резултујућих пиксела, позиција ј увећава се за G\_PH\_NUM, а пиксели на улазу задржавају се и користе и у следећем циклусу. На описан начин постиже се уштеда ресурса у односу на дизајн који би могао да у једном циклусу прорачуна све резултујуће пикселе за један пар на улазу филтра, али могуће да је потребно повећати брзину рада, тј. повећати фреквенцију такта на ком ради дизајн, да не би дошло до успоравања тока података који пристижу на улаз модула за промјену димензија слике, што негативно утиче на потрошњу и носи са собом одређене изазове. G\_PH\_NUM је параметар дизајна, који може да се изабере прије покретања фаза анализе и синтезе дизајна.



Слика 5.17: Изведба билинеарног филтра из два основна модула

модула за рачунање удаљености оригинала од пиксела са почетне слике, тј. Коефицијената утицаја пиксела са оригиналне слике и модула за рачунање вриједности резултујућих пиксела

#### Реализација модула за избор коефицијената билинеарног филтра

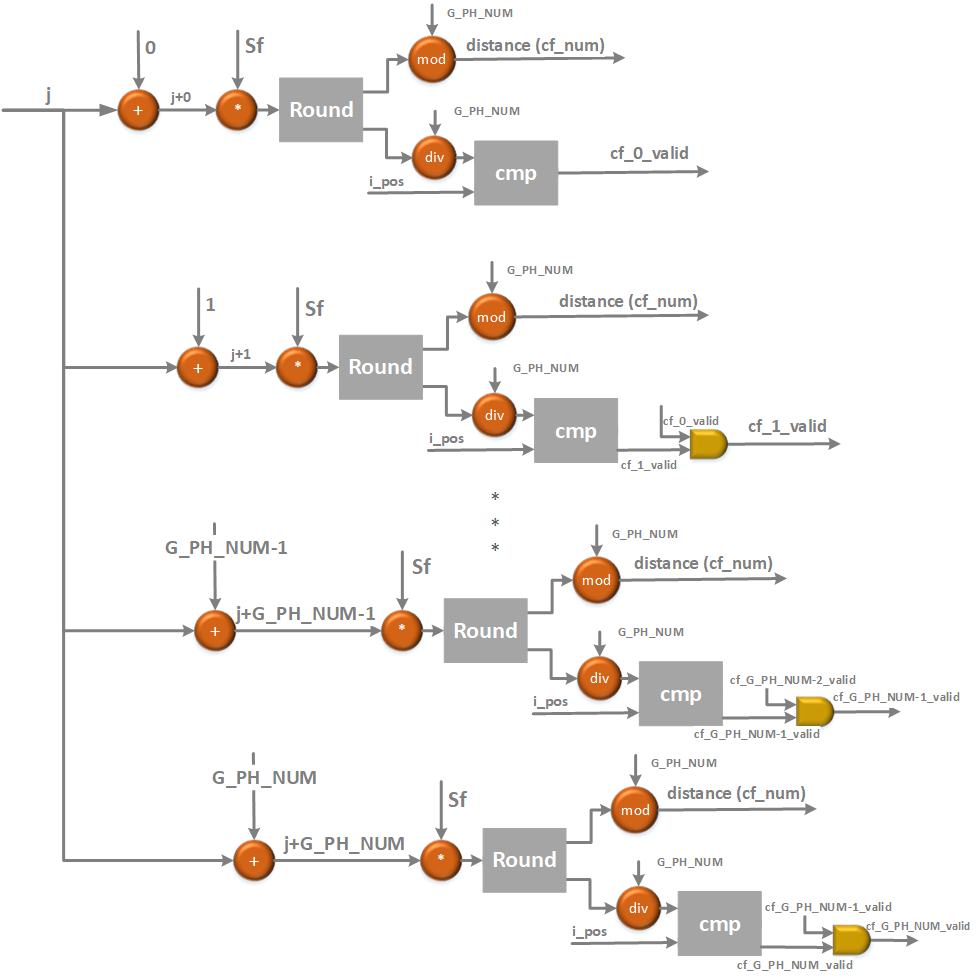
Потребно је на основу позиције пиксела на улазу филтра, те параметара као што су величине резултујуће и оригиналне слике по посматраној димензији, израчунати које позиције резултујућег пиксела потичу од пара пиксела на улазу, те позицију оригинала у односу на један и други пиксел. Позиција оригинала, која се налази између пиксела на улазу, даје информацију колики је њихов утицај, понаособ, на резултујући пиксел, тј. на основу ове информације бирају се коефицијенти којима се множе пиксели са оригиналне слике у циљу добијања пиксела на резултујућој слици. Схема модула који врши ову функцију приказана је на *Слика 5.19*. Када се на улазу појави ваљан пар пиксела, чија је позиција i\_pos, провјерава се да ли резултујући пиксел на позицијама ј до ј + G\_PH\_NUM -1 потичу од пиксела на улазу и ако потичу колика је удаљеност оригинала од пиксела са мањом координатом по посматраној оси. Почетна вриједност ј, позиција резултујућег пиксела који се рачуна, јесте 0. Максималан број коефицијената који се може одредити у једном циклусу јесте G\_PH\_NUM. Ако је на једној од излазних позиција „valid“ буде постављен на 0, пиксели на улазу су потпуно искориштени (прорачунати су коефицијенти за све пикселе који потичу од пара на улазу), претходном модулу се, постављајући хендшејк сигнал full на 0, сигнализира да може да пошаље следећи пар пиксела са оригиналне слике. Позиција следећег резултујућег пиксела, „j\_next“ на слици, јесте прва позиција за коју на излазу немамо ваљан сигнал за избор коефицијената „cf\_num“. Ако су сви коефицијенти ваљани, тј. пиксели на улазу нису до краја искориштени, хендшејк сигнал full има вриједност 1, а у следећем циклусу позиција резултујућег пиксела од које почињемо рачунање броја за избор коефицијента постаје ј + G\_PH\_NUM. Логика за избор следеће позиције ј јесте једна од ствари која чини разлику између вертикалног и хоризоналног филтра и биће детаљно описана у даљем раду.

Diagram, timeline

Description automatically generated

Слика 5.18: Ћелија за рачунање индекса за одабир коефицијената филтера

Да би се омогућио истовремени избор коефицијената за рачунање више од једног резултујућег пиксела у једном циклусу, ћелија за рачунање индекса коефицијената, раније описана и приказана на *Слика 5.15*. умножена је G\_PH\_NUM +1 пута и на улаз сваке доведена је позиција резултујућег пиксела за један увећана у односу на позицију доведену на претходној ћелији, *Слика 5.20*. Последња ћелија служи само за провјеру да ли резултујући пиксел на позицији ј + G\_PH\_NUM потиче од пиксела на улазу, јер ако претходни пиксели потичу, посматрани не потиче, а посматрану ћелија не буде дио дизајна, хендшејк сигнал full би непотребно задржао вриједност 1 и још један циклус би се покушавало искористити пикселе на улазу, који су већ потпуно искориштени.



Слика 5.19: Модул за избор коефицијената у билинеарном филтру

Као што је речено, наредна позиција резултујућег пиксела за коју ће се рачунати индекси коефицијената бира се на основу сигнала који говоре о ваљаности коефицијената. Дигитални дизајн намјењен за избор следеће позиције за коју се рачуна резултујући пиксел приказан је на *Слика 5.21*. У случају да коефицијент 0 није ваљан, ради се о смањењу слике по посматраној димензији, те позиција резултујућег пиксела остаје ј, док се на улаз доводи нови пар пиксела са оригиналне слике. Уколико се ипак ћелији G\_PH\_NUM нађе сигнал о ваљаности постављен на јединицу, то говори да се пиксели на улазу користе и у наредном циклусу за прорачун резултујућих пиксела, а позиција резултујућег пиксела од кога се започиње прорачун у наредном циклусу јесте ј + G\_PH\_NUM.

Diagram

Description automatically generated

Слика 5.20: Модул за рачунање позиције следећег резултантног пиксела

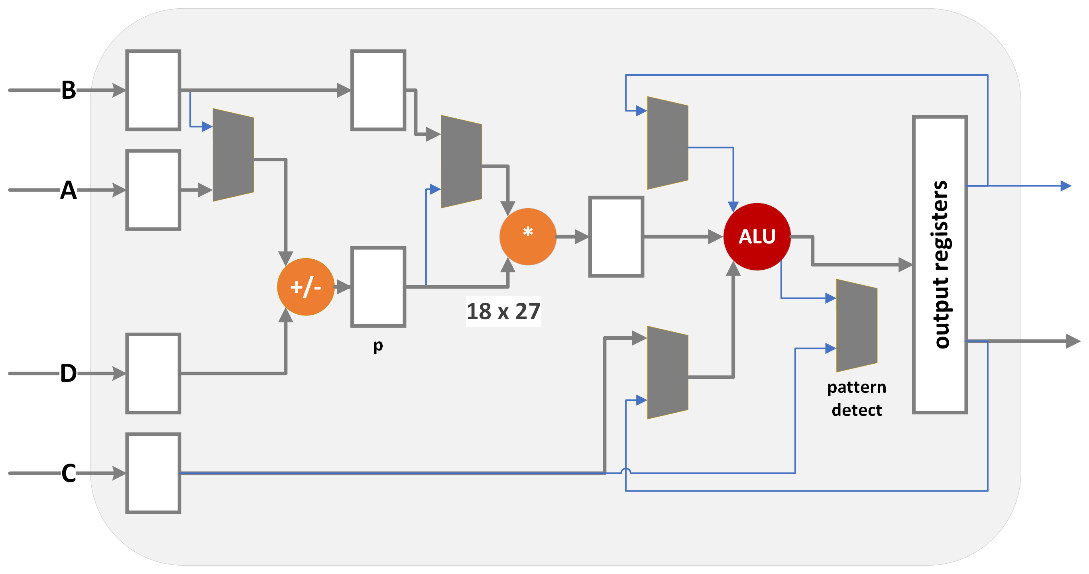
Због другачијег мијењања позиције улазних пиксела при довођењу на вертикални и на хоризонтални филтар постоје разлике и у избору ј\_next вриједности која се узима као позиција наредног резултујућег пиксела који се рачуна. Наиме, на вертикални филтар, све до појаве сигнала за последњи сигнал у реду, доводе се парови пиксела који имају исту позицију, па с обзиром на то ј\_next се користи само ако пиксели на улазу нису у потпуности искориштени или ако смо искористили последњи пар пиксела из посматраног реда, док ако јесу у потпуности искориштени поново се полази од вриједности ј која се користила на почетку. У случају да се ради о хоризонталном филтру, позиција пиксела се мијења све до последњег пиксела у реду, када се прелази на следећи пар пиксела који потиче из следећег реда, па се сходно томе и ј\_next користи све до последњег пиксела у реду, након кога се поново почиње од нуле.

#### Управљање вриједностима додатних сигнала

Уколико је на улазу вертикалног филтра доведен пар пиксела, са додатним сигналном за почетак нове слике, та информација прослијеђује се само једном резултујућем пикселу и то оном са најнижом вриједности ј. Информација о последњем пикселу у реду прослијеђује се свим резултујућим пикселима, јер сви и чине последње пикселе у насталим редовима. Хоризонтални филтар има благо другачије понашање. На хоризонталном филтру информација о почетку нове слике такође се прослијеђује само једном пикселу и то оном на најнижој позицији, у овом случају на позицији нула, док се информација о последњем пикселу у реду прослијеђује само последњем пикселу који настаје од посматраног пара пиксела на улазу.

#### Изведба модула за рачунање вриједности међурезултата и резултујућих пиксела

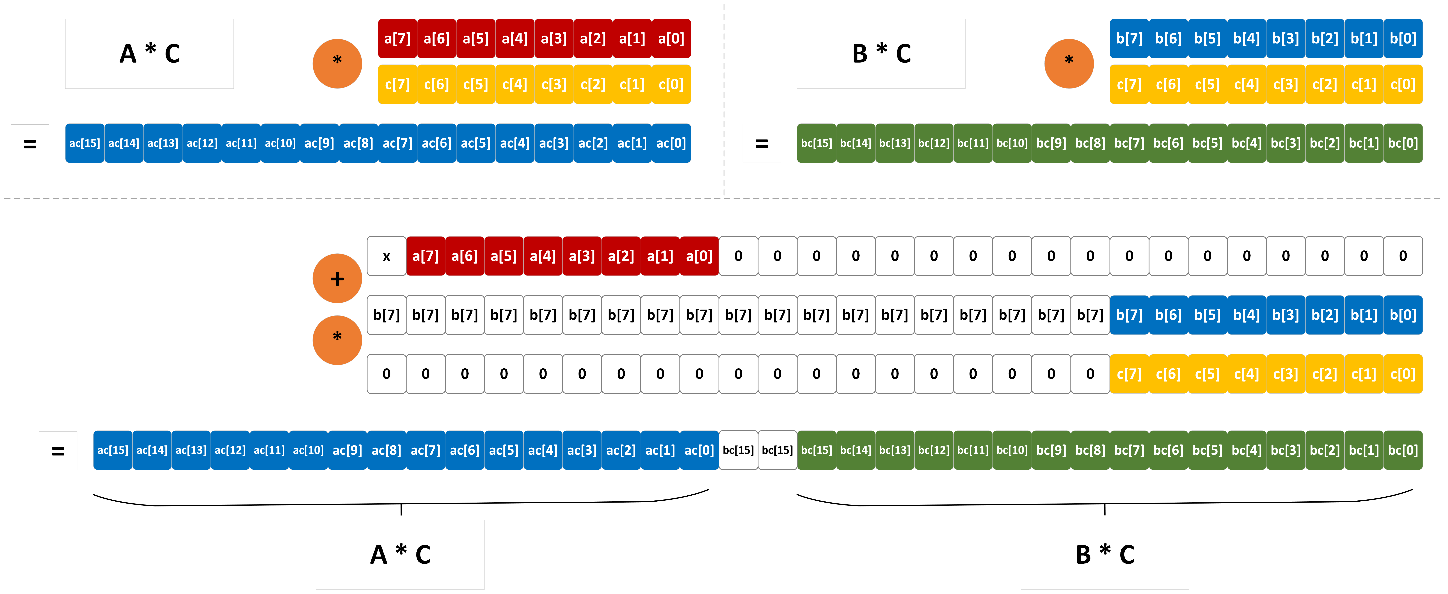
Након избора коефицијената приступа се израчунавању међурезултата (уколико се ради о вертикалном филтру) или резултујућих пиксела (уколико је ријеч о хоризонталном филтру), *Слика 5.16*. Као што је приказано на поменутој слици, да би се добио међурезултат или резултат на филтру, потребно је извршити множење два пиксела са одговарајућим коефицијентима, те сабирање резултата множења. Због намјере да се дизајн користи у ланцу са другим модулима за обраду слике, тежи се коришћењу што мањег броја ресурса и што бољој искориштености кориштених ресурса. У том циљу дизајн је описан тако да се извођење операције множења синтетизује користећи ДСП48Е2 компоненту, често присутну у ФПГА чиповима новије производње, *Слика 5.21*, док се искоришћеност компоненте постиже паковањем улазних података тако да се на једном искоришћеном ДСП-ју изводе два множења и сабирање[10] [11].



Слика 5.21: Поједностављен приказ Xilinx FPGA DSP, DSP48E2

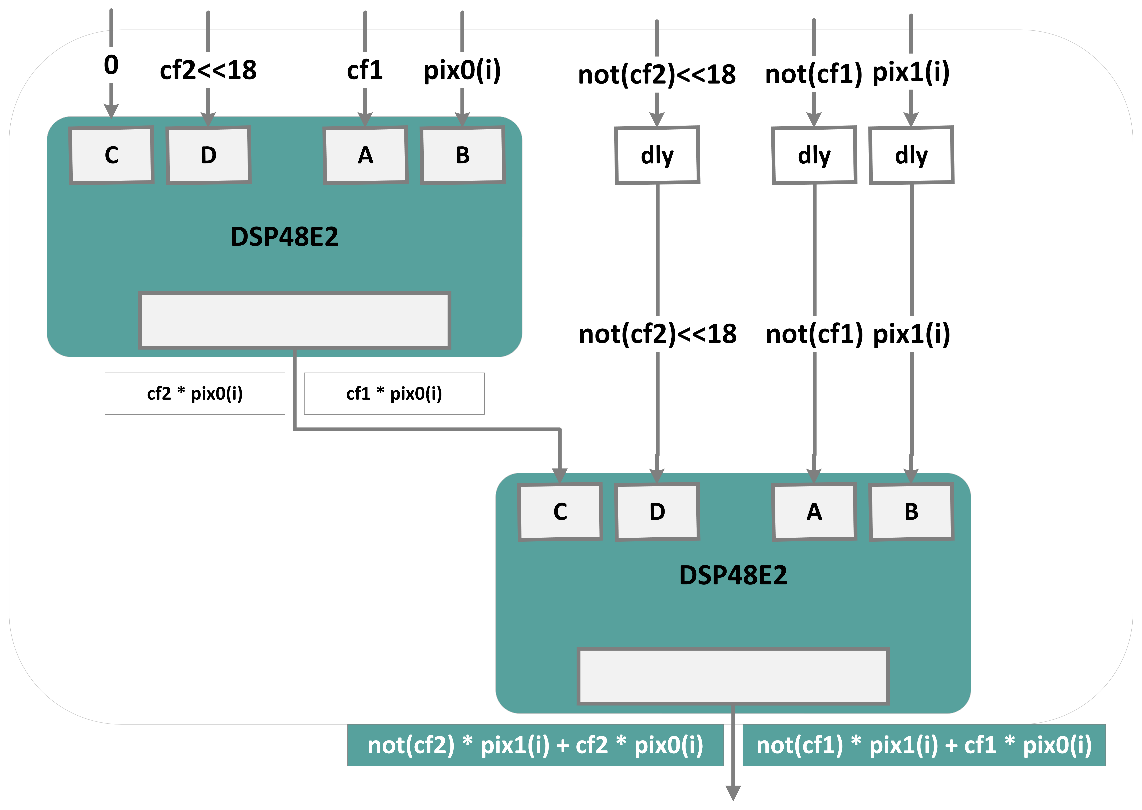
Ако се користе 8-битне представе операнада, операција A\*B + D\*B, коришћењем компоненте ДСП48Е2, остварује се, *Слика 5.22*:

* довођењем заједничког множиоца на улаз B ДСП компоненте
* на улаз D доводи се 27-битни вектор, гдје је 8-битни операнд смјештен тако да заузима 8 његових позиција са најнижим вриједностима. У случају рада са означеним вриједностима остале позиције у вектору узимају вриједност највишег бита операнда, док се у случају рада са неозначеним бројевима, што је и овдје случај, оне постављају на вриједност нула
* на улаз А доводи се 27-битни вектор, гдје је највиши бит резервисан за предзнак, уколико се користе означени бројеви, док је у реализованом дизајну постављен на вриједност нула, затим се у вектор смјешта 8-битни операнд, а вриједност осталих бита у вектору поставља се на вриједност нула



Слика 5.22: Добијање А\*C и B\*C у једној операцији множења, кориштењем технике паковања 8бит података и DSP48E2

Описано паковање операнада у дизајну користи се да би се један пиксел помножио са два коефицијента и резултат оба множења добио истовремено. Тако се на улазе А и D добављају коефицијенти филтра, запаковани у 27-битне вриједности, како је раније описано, док се вриједност свјетлине пиксела поставља на улаз B. Да би се добиле крајње вриједности свјетлине међурезултата или резултујућег пиксела (што зависи од тога да ли се ради о вертикалном или хоризонталном филтру), ради се упаривање две ДСП компоненте, како је приказано на *Слика 5.23*.



Слика 5.23: Кориштење DSP48E2 за извођење операција множења у билинеарном филтру

Резултат множења пиксела означеног са pix0 са одговарајућим коефицијентима, доводи се на улаз C упареног ДСП модула, на који се истовремено са пристизањем резултат на његовом C улазу доводе други пиксел, pix1, са одговарајућим коефицијентима. На излазу другог ДСП модула пристижу истовремено два резултујућа пиксела (или међурезултата), па се тако коришћењем два упарена ДСП модула остварју четири операције множења и двије операције сабирања. Описана оптимизација постиже се кориштењем ФПГА чипова који посједују ДСП48Е2 компоненте, док ако се користе ФПГА чипови са другим ДСП компонентама или чипови који немају ДСП компоненте на располагању, синтеза дизајна била би могућа, али би (и) други ресурси били искориштени за то.

## Меморијскe ФИФО (енг. FIFO – First In First Out) банкe

У фифо меморију прослијеђује се слика чије су димензије промјењене по вертикалној оси. Пиксели смјештени у фифо банке читају се ред по ред и прослијеђују на филтар који мијеља хоризонталну димензију слике. Максималан број резултујућих пиксела који потичу од истих оригинала и који се у једном циклусу могу прослиједити са филтра према посматраном модулу јесте G\_PH\_NUM, док то није укупан максималан број таквих резултујућих пиксела. Максималан број пиксела који потичу од два сусједна пиксела са оригиналне слике одређен је односом димензија излазне и улазне слике, у овом случају димензије по вертикалној оси. Равнајући се према наведеном, колекција фифо банака садржи:

Nfb = Vrez\_dim mod Vin\_dim + K

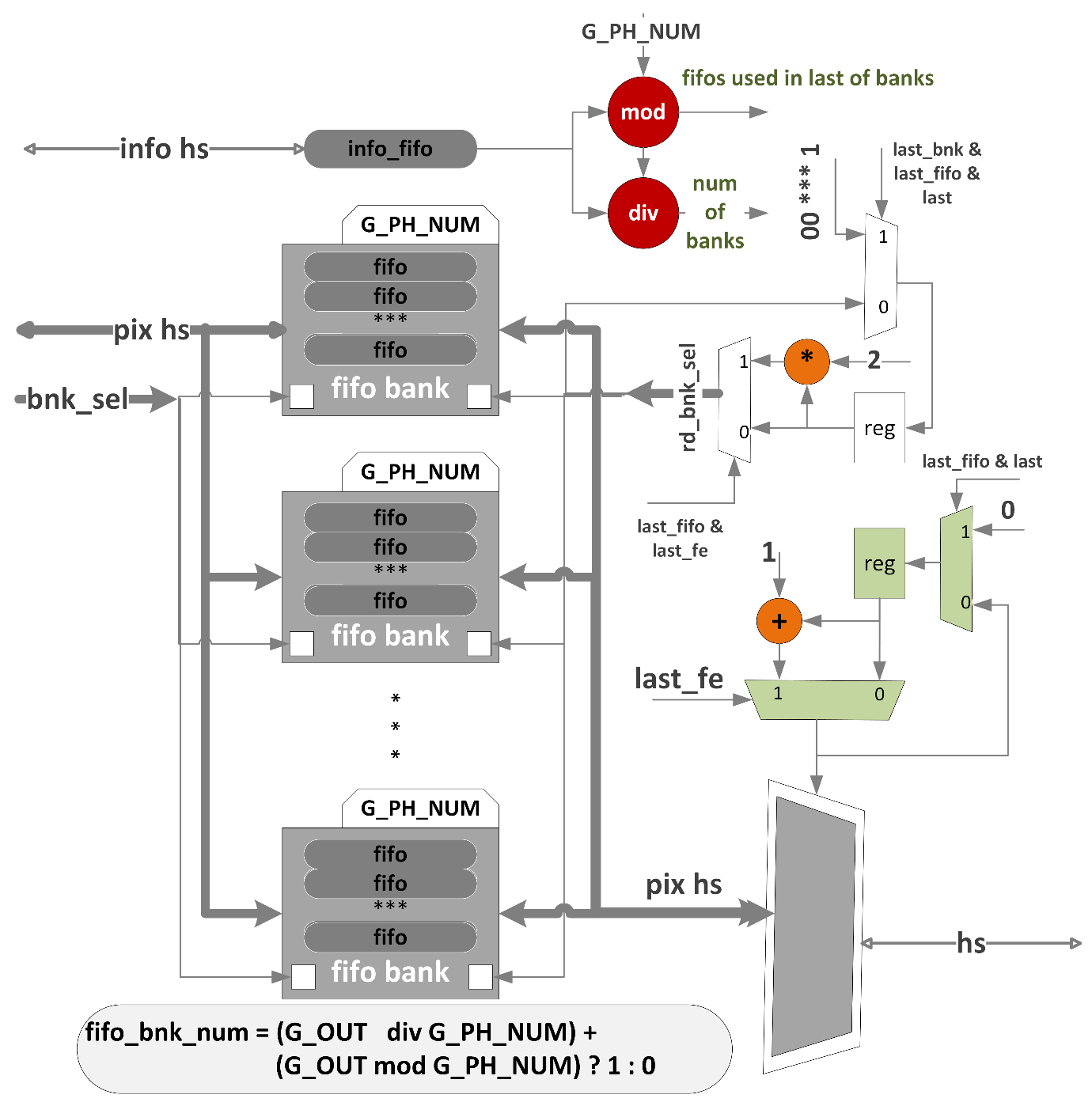
К = 1 ? (Vrez\_dim mod Vin\_dim) : 0

гдје је: Nfb, број фифо банака

Vrez\_dim , Вертикална димензија резултујуће слике

Vin\_dim , Вертикална димензија оригиналне слике

Свака фифо банка садржи G\_PH\_NUM број фифо модула, да би била у могућности да прихвати све податке који у истом тренутку могу доћи са претходног модула. Ако се каже да пиксели који истовремено стижу на модул, тј. потичу од једног пара оригиналних пиксела и долазе у истом такт циклусу, чине један пакет пиксела, онда сигнал „bnk\_sel“ носи информацију о томе који је редни број пакета пиксела пристигао и сходно томе бира се у коју банку се уписује. У тренутку када долази последњи пакет стиже и информација колико је укупно пристигло пиксела и смијешта се у издвојен фифо модул, *Слика 5.9*. Читање увијек креће од фифо модула из нулте банке који има индекс нула (прва фифо банка гледано од горе и први фифо у њој на *Слика 5.9*). Фифо који се чита, чита се све до исчитавања пиксела који је последњи пиксел у реду слике, након чега се прелази на исчитавање следећег фифо модула. Када се исчита пиксел који је последњи у реду, из фифа који је последњи у банци, прелази се на следећу банку или се поново почиње од почетног фифа из почетне банке уколико су исчитани сви редови који потичу из истог реда пиксела са оригиналне слике. Логика офарбана зеленом бојом на *Слика 5.6*. служи да се послије прочитаног последњег пиксела у једном реду пређе на следећи ред, тј. следећи фифо. Када се прочита последњи фифо у банци или последњи фифо у банци са корисном информацијом, бројање креће од нуле. Ако су исчитани подаци из последњег фифоа у банци, али то нису сви подаци пристигли као резултат обраде једног пара пиксела, прелази се на читање следеће банке, а тај дио обавља логика на слици приказана бијелом бојом. Информацију о броју читања које је потребно обавити добијамо из издвојеног фифо модула. Цјелобројним дјељењем информације у наведеном фифу добија се број фифо банака које треба исчитати до краја, а остатак тог дијељења даје информацију колико фифо модула треба исчитати у последњој кориштеној банци.



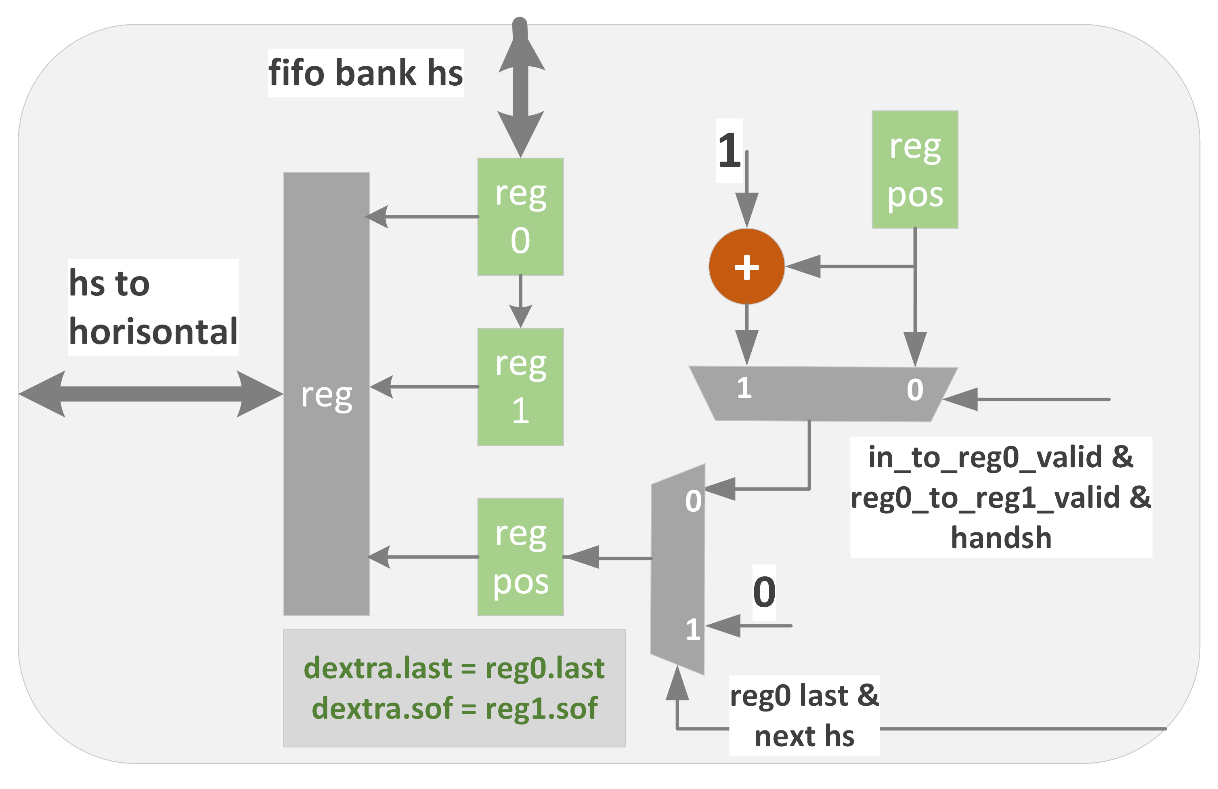
Слика 5.24: Колекција ФИФО банака

С обзиром да је G\_PH\_NUM изабран тако да буде степен броја два, информација о банкама у којима су искориштени сви фифои и броја фифоа искориштених у последњој кориштеној банци добија се једноставним дијељењем бинарног записа из информационог фифа на два дијела, на следећи начин:

* податак о броја фифоа искориштених у последњој кориштеној банци добијамо гледајући бите у распону од log2G\_PH\_NUM -1 до 0
* информација о броју банака у којима су искориштени сви фифои добијамо гледајући бите у распону од највшег бита у податку који добијамо из издвојеног фифо модуле, па до log2G\_PH\_NUM бита

## Прилагођење података на хоризонтални филтар

Као и вертикални филтар и хоризонтални филтар на свом улазу очекује пар пиксела, информацију о њиховој позицији у реду из кога долазе, те додатне сигнале, као што су сигнал за почетак нове слике или сигнал за крај једног реда.



Слика 5.25: Прилагођење тока података на џоризонтални филтар

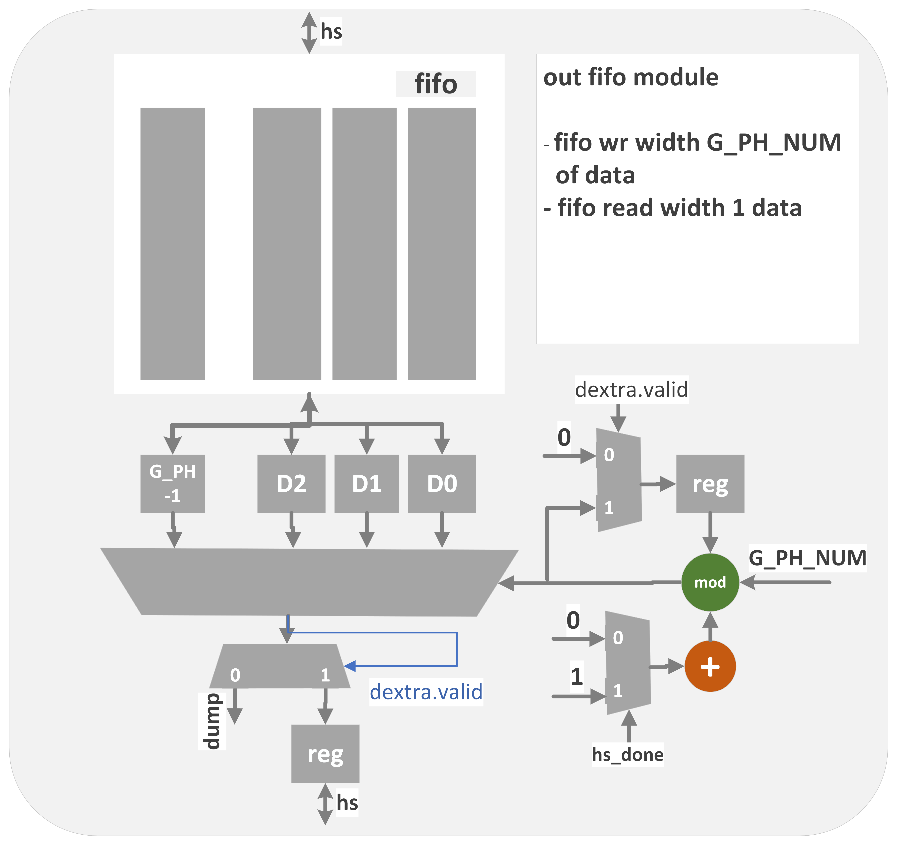
Изведба модула приказана је на *Слика 5.10*. Податак прво пристиже у регистар 0, а на појаву следећег податка прослијеђује се у регистар 1. Уколико у оба регистра постоје ваљани подаци, заједно са позицијом у реду и додатним сигналима уписују се у излазни регистар. Позиција почиње од 0, а увећава се пристизањем сваког новог пиксела све до последњег у реду, када се поново поставља на почетну вриједност. Додатни сигнали прослијеђују се на излазном регистру на следећи начин:

* сигнал за почетак нове слике прослијеђује се узимајући сигнал везан за пиксел у регистру 1
* сигнал за последњи пиксел у низу прослијеђује се узимајући сигнал везан за пиксел у регистру 0

Након што се из регистра 0 прочита пиксел који је поледњи у реду слике, он више није ваљан, тј. не уписује се у регистар 1 или се занемарује и ако је тамо уписан.

## Излазни ФИФО

Последњи модул у дизајну који комуницира са околином по протоколу усвојеном за комуникацију у дизајну јесте излазни фифо модул. Излазни фифо предвиђен је да прима податке са хоризонталног билинеарног филтра и то максималан број који може пристићи у једном тренутку, G\_PH\_NUM, те да их прослијеђује један по један на прилагодни степен, степен који служи за прилагођење уређаја за промјену димензије слике на магистралу преко које комуницира са околином (у овом случају на акси стрим магистралу). Скица дизајна приказана је на *Слика 5.23*. Подаци који пристижу уписују се у фифо модул означен на слици. Подаци из фифа пристигли у једном такту, исчитавају се у регистре чији садржај се провјерава и прослеђује на излазни регистар уколико су подаци ваљани. Када се прочита регистар у ком се налази податак који није вањан исчитава се наредни скуп података из фифоа,и читање регистара креће од почетка (од регистра на слици означеног са Dо).



Слика 5.26: Излазни ФИФО модул

Прочитане информације, уколико су ваљане, шаљу се на излазни степен који их прилагођава и прослијеђује даље.

# ЗАКЉУЧАК

# ЛИТЕРАТУРА

[1] <https://www.xilinx.com/support/documentation/ip_documentation/v_scaler/v8_1/pg009_v_scaler.pdf>

[2] https://dsp.etfbl.net/students/ceko.pdf

[3] Мултумедијални системи, проф. др Владимир Рисојевић

[4] <https://edtech.engineering.utoronto.ca/sites/default/files/M6_2DImageDigitalRepresentation.png>, посјећено 7. октобра 2021. године

[5] FPGA for dummies, 2nd Intel Special Edition, Andrew Moore, Ron Wilson

[6] Дигитална електроника, проф. др Бранко Докић

[7] Импулсна електроника, проф. др Бранко Докић

[8] <https://www.coursera.org/learn/intro-fpga-design-embedded-systems>, посјећено 10. јануара 2022. године

[9] AMBA® 4 AXI4-Stream Protocol, **Version: 1.0**

[10]DSP-Packing: Squeezing Low-precision Arithmetic into FPGA DSP Blocks,

Jan Sommer, Akif O¨ zkan, Oliver Keszocze, Member, IEEE Ju¨rgen Teich, Fellow, IEEE

[11] Deep Learning with INT8 Optimization on Xilinx Devices, *Yao Fu, Ephrem Wu, Ashish Sirasao, Sedny Attia, Kamran Khan, and Ralph Wittig*