Програмчлагддаг Логик Чип Ашигласан 32 Битийн Хөвөгч Цэгтэй Тооны Нэмэгчийн Хэрэгжүүлэлт

Т.Зулхүү, М.Баярпүрэв
Монгол Улсын Их Сургууль
Хэрэглээний шинжлэх ухаан, инженерчлэлийн сургууль
Электроник, Холбооны инженерчлэлийн тэнхим
zulkhuu@hotmail.com, bayarpurev@num.edu.mn

Хураангуй **- Хөвөгч цэгтэй тоо нь тооцооллын үр дүнг нарийвчлалтай сайтай болгодог учраас тооцоолон бодох системүүдэд өргөн хэрэглэдэг. Энэхүү судалгааны ажлаар IEEE 754 стандартын хөвөгч цэгтэй тооны нэмэгчийн дизайныг хийхийг зорьсон. Нэмэгчийн хэсгийг хийхдээ Когге Стоны нэмэгчийг ашигласан. Нэмэгчийн оролтонд IEEE 754 стандартын нормчлогдсон хоёр тоо өгөх ба гаралтандаа мөн уг стандартад нийцсэн нормчлогдсон хөвөгч цэгтэй тоог гаргана. Нэмэгчийн хэсгийг хийхдээ Когге Стоны нэмэгчийг ашигласан. Санал болгож буй нэмэгчийн дизайныг ISE Xilinx 12.2 програмыг ашиглан симуляци хийсэн бөгөөд Spartan 3E FPGA сургалтын хавтан дээр синтез хийсэн.**

Түлхүүр үг—Когге Стоны нэмэгч., IEEE-754 стандарт, LZA, LZC хэлхээ

I. Удиртгал

Дүрс болон дуу боловсруулах маш олон салбарт том хэмжээний өгөгдлийг харьцангуй өндөр нарийвчлалтайгаар боловсруулах шаардлага гардаг. Тоон дохио боловсруулах салбарт удаан хугацаанд микропроцессорт суурилсан боловсруулалтууд зонхилж байсан. Дохио боловсруулах алгоритмыг хэрэглэсэн бүтээгдэхүүнүүд маш олон хэрэглээнд өргөн хэрэглэгдэж байна. Электроникйн дизайныг автоматжуулах програмууд болон програмчлагддаг логик чип гарч ирснээр эдгээр дохиог илүү хурдтай, үр ашигтай боловсруулах боломжууд нээгдсэн [1].

Энэхүү ажлаар хийсэн нэмэгчийн дизайн нь оролтондоо IEEE 754 стандартын нормчлогдсон хоёр тоо өгөх ба гаралтандаа мөн уг стандартад нийцсэн нормчлогдсон хөвөгч цэгтэй тоог гаргана. Нэмэгчийн дизайн нь таван үе шаттайгаар нэмэх үйлдлийг гүйцэтгэх ба шат бүр нь хамгийн ихдээ 10 логик гэйтийн хугацааны хоцрогдолтой байна. Энэ нь хэрвээ нэг логик гэйтийн хугацааны хоцрогдлыг 1 наносекунд гэж үзвэл манай нэмэгч маань хамгийн ихдээ 100 МегаГерцийн давтамжинд ажиллана гэсэн үг юм [2]. Нэмэгчийн хэсгийг хийхдээ Когге Стоний нэмэгчийг ашигласан.

Энэхүү өгүүллийн дараагийн бүлэг болох II бүлэгт IEEE-754 стандартын хөвөгч цэгтэй тооны форматын тухай, III бүлэгт хөвөгч цэгтэй тооны нэмэгчийн архитектур дизайныг хэрхэн шийдсэн болоод түүний дэд хэсгүүдийн шийдлийг хэрхэн гаргасан тухай, IV бүлэгт симуляцийн болон синтезийн үр дүнг харуулсан.

II. Судалгаа

Бодит тоог дүрслэх хамгийн үр ашигтай арга бол хөвөгч цэгтэй тооны дүрслэл юм. Хөвөгч цэгтэй тооны дүрслэл нь мөш өргөн хязгаарт тоог дүрслэж чаддаг бөгөөд өндөр нарийвчлалтай тооцоолол хийх боломжийг олгодог давуу талтай. Хөвөгч цэгтэй тоо нь суурь(base), мантис(mantissa) болон экспонент(exponent) гэсэн хэсгүүдээс бүрддэг. (1)Ту-хайн тоо нь хэд гэсэн тоо болохыг дараах илэрхийллээр олно:

$$value = mantissa * base^{exponent} \quad (1)$$

IEEE-с гаргасан ANSI/IEEE Std 754–1985 стандарт нь хоёртын суурьтай хөвөгч цэгтэй тоог дүрслэх түүнийг ашиглан арифметикийн үйлдлүүд хэрхэн хийхийг зааж өгсөн. IEEE 754 стандартад зааснаар хөвөгч цэгтэй тоо нь тэмдэгийн бит(S), мантиссын бит(M) болон экспонентийн бит(E)-үүдээр илэрхийлэгдэнэ. (2)

$$value = (-1)^S * M * 2^E \quad (2)$$

Энэхүү стандарт маань хөвөгч цэгтэй тоог 32 битээр дүрслэх дан нарийвчлал болон 64 битээр дүрслэх давхар нарийвчлал гэсэн сонголтуудтай байдаг. Бидний хэрэгжүүлэх нэмэгчийн дизайн маань зөвхөн дан нарийвчлалтай тоог дэмжин ажиллахаар хийгдсэн

болно. Тухайн нарийвчлал бүрт тоог ямар форматаар дүрслэдэгийг доорх хүснэгтээс харж болно. IEEE 754 стандартаар дүрслэгдсэн тооны илэрхийлж буй утга нь 2 томьёогоор илэрхийлэгддэг гэж үзсэн. Гэвч дээрх илэрхийлэлд захирагддаггүй онцгой тохиолдлуудыг IEEE 754 стандартад тодорхойлсон байдаг. Эдгээрийг дурдвал:

- Хэрвээ экспонент нь 255-тай тэнцүү бөгөөд мантис нь тэгээс ялгаатай байвал NaN(Not a number) буюу алдаатай байна гэж үзнэ.
- Хэрвээ экспонент нь 255-тай тэнцүү бөгөөд мантис нь тэгтэй тэнцүү байвал тухайн тоо тэмдэгийн битээс хамааран нэмэх эсвэл хасах хязгааргүй байна.
- Хэрвээ экспонент болон мантис нь тэгтэй тэнцүү байвал тухайн тоо тэмдэгийн битээс хамааран нэмэх эсвэл хасах тэг байна
- Хэрвээ экспонент нь тэгтэй тэнцүү бөгөөд мантис нь тэгээс ялгаатай байвал тухайн тооны илэрхийлж буй утгыг дараах томьёогоор олно

$$value = (-1)^S * (0.M) * 2^{E-126} \quad (3)$$

- Хэрвээ экспонент нь 0<E<255 хооронд байвал тухайн тооны илэрхийлж буй утгыг дараах томьёогоор олно
- $$value = (-1)^S * (1.M) * 2^{E-127} \quad (4)$$

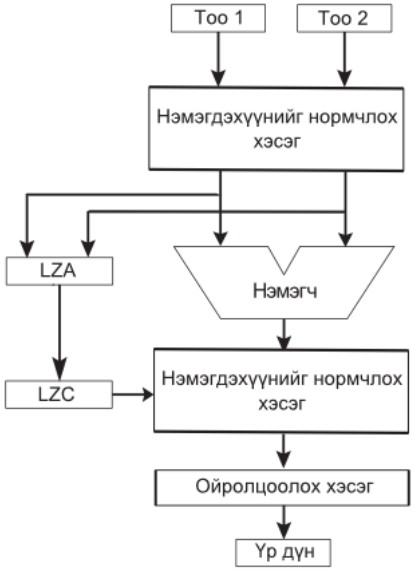
Нарийвчлал	Дан	Давхар
Мантиссын өргөн битээр	24	53
Экспонентийн өргөн битээр	8	11
Экспонентийн хамгийн их утга	+127	+1023
Экспонентийн хамгийн бага утга	-126	-1022
Нийт өргөн	32	64

Хүснэгт 1 IEEE-754 формат

III. Хөвөгч цэгтэй тооны нэмэгчийн архитектур дизайн

Хөвөгч цэгтэй тоонуудыг нэмэх/хасах үйлдэл нь бусад арифметик үйлдлүүдээсээ илүү төвөгтэй олон үйлдэл шаарддаг. Нэмэх үйлдэл гүйцэтгэхийн өмнө ба гүйцэтгэсний дараа тоонуудаа нормчлох хэрэгтэй ба экспонентийг мөн тооцож олох шаардлагатай. Хөвөгч цэгтэй тоонуудыг нэмэхэд дараах үйлдлүүдийг хийдэг.

- Экспонентүүдийн зөрүү буюу ялгаврыг олох. Экспонентүүдийн ялгавар нь мантиссыг нормчлоход хэрэглэгдэнэ.
- Мантисыг нормчлох. Аль бага мантисыг экспонентүүдийн ялгаврын хэмжээгээр баруун шифт хийнэ. Ингэснээр хоёр тоо ижил зэрэгтэй болж нэмэх үйлдэл гүйцэтгэх боломжтой болно.
- Нэмэх буюу хасах үйлдэл. Тэмдэгтэй хоёр тоог нэмэх үйлдэл гүйцэтгэнэ. Нэмэгдэхүүнүүдийн тэмдэг ялгаатай бол хасах үйлдэл хийнэ. Хасах үйлдлийг хоёртын гүйцээлтийг нь нэмэх замаар гүйцэтгэнэ.
- Нийлбэрийг нормчлох. Хоёр мантисын нийлбэр/ялгавар нь орон ахьсан эсвэл орон зээлсэн байх тохиолдлуудад нийлбэрийн экспонент өөрчлөгдөх ба мантисыг дахин нормчлох шаардлагатай.
- Ойролцоолох. Гарсан үр дүнг IEEE-ийн стандартад нийцүүлэхийн тулд зарим тодорхойгүй битүүдийг ойролцоолж авах шаардлагатай.



Зураг 1 Нэмэгчийн архитектур

Эдгээр үйлдлүүдийг гүйцэтгэх блокууд нь хоорондоо хэрхэн холбогдож ажиллахыг нэмэгчийн блок диаграмаас үзэж болно. Хөвөгч цэгтэй тооны нэмэгчийн блок диаграмыг 1-р зурагт үзүүллээ.

A. *Дэд хэсгүүдийн шийдэл*

Нэмэгчийн ажиллагаа хурдан байх нь нэмэгчийн блок диаграмд агуулагдсан дэд хэсгүүдийг хэр сайн хийсэн бэ гэдгээс шууд хамаарна. Нэг л дэд хэсэг их хугацааны хоцрогдолтой байвал системийн нийт хугацааны хоцрогдол төдий чинээгээр нэмэгдэх учраас нэмэгчийн дизайнд агуулагдаж буй дэд хэсэг бүрийг маш хурдан ажиллагаатай байхаар шийдэх хэрэгтэй. Зураг 1-д үзүүлсэн нэмэгчийн архитектурын чухал гэсэн блокууд буюу дэд хэсгүүдийг дараагийн бүлгүүдэд тайлбарлана.

B. *Нийлбэрийн тэмдэгийг тодорхойлох*

Нийлбэрийн буюу ялгаврын тэмдэг нь аль их тооныхоо тэмдэгийг дагаж гарна. Тиймээс хоёр тооны мантис болон экспонентийг харьцуулж аль тоо нь их болохыг олох хэрэгтэй. Мантисуудын харьцаа нь зөвхөн экспонентүүд тэнцүү үед л хэрэглэгдэнэ. Энэ хэсэг нь мөн хоёр тооны тэмдэг ялгаатай эсэхийг шалгаж гаралтандаа гаргана. Хоёр тоо ялгаатай эсэхийг логик XOR үйлдэл ашиглан хялбар шалгаж болно. Хоёр тооны тэмдэгүүд ялгаатай байх нь нэмэх үйлдэл хийх үү хасах үйлдэл хийх үү гэдэгийг шийднэ. Экспонентүүд болон мантисуудын харьцаанаас нийлбэрийн тэмдэг хэрхэн хамаарах вэ гэдэгийг хүснэгт II-ээс харж болно.

Экспонентүүдийн харьцаа	Мантисуудын харьцаа	Нийлбэрийн тэмдэг
$E_1>E_2$	X	S_1
$E_1<E_2$	X	S_2
$E_1=E_2$	$M_1>M_2$	S_1
$E_1=E_2$	$M_1<M_2$	S_2

Хүснэгт 2 Нийлбэрийн тэмдэг экспонент болон мантисын харьцаанаас хамаарах нь

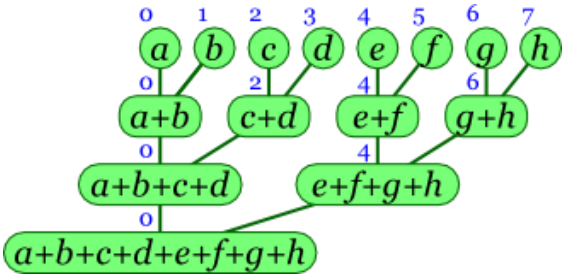
C. *Нэмэгдэхүүнийг нормчлох*

Мантисуудыг нэмэхийнхээ өмнө нормчлох шаардлагатай. Нормчлоход хийх хамгийн эхний үйлдэл нь жиших үйлдэл бөгөөд аль тоо нь их вэ гэдэг нь экспонентүүдийн харьцаанаас мэдэгдэнэ. Аль их тооныхоо мантисыг нэмэгчийн оролтонд шууд холбож өгөх хэрэгтэй. Үүний дараа бага тооныхоо мантисыг нормчлох үйлдэл хийж их тооныхоо мантистай ижил зэрэгтэй болгох ёстой. Ингэж бага тооны мантисыг нь их тооны мантистай зэрэгцүүлэхийн тулд баррел шифтер(barrel shifter)-ийн хэлхээгээр оруулна. Баррел шифтер нь нэг клок циклд олон шифт хийдэгэрээ бусад шифт хийдэг хэлхээнүүдээс давуу талтай. Хэр хэмжээгээр шифт хийх вэ гэдэг нь экспонентүүдийн ялгавартай тэнцүү байна.

Эцэст нь хэрвээ хасах үйлдэл хийх гэж байгаа бол мантисуудаа зэрэгцүүлсний дараа бага тооны мантисаасаа хоёртын гүйцээлт авах үйлдэл хийнэ. Нэмэх үйлдэл хийх гэж байгаа бол энэ үйлдлийг алгасна(тэмдэгийн битүүд ялгаатай байвал хасах үйлдэл хийнэ гэдэгийг дээр дурдсан).

D. *Нэмэгч*

Олон төрлийн нэмэгч байдгаас Carry Look Ahead төрлийн нэмэгчүүд хамгийн хурдан нь байдаг. Энэ төрлийн нэмэгчүүдийн дундаас хамгийн хурдан бөгөөд цөөн тооны логик гэйт шаарддаг нэмэгч бол Когге Стоны нэмэгч юм. Когге Стоны нэмэгчийн гэйтүүдийн хоорондын холболт нь бусад нэмэгчийг бодвол илүү төвөгтэй байдгаараа сул талтай. Дөрвөн битийн Когге



Зураг 2 Когге Стоний нэмэгч

Стоний нэмэгчийн бүтцийг Зураг 2-т харууллаа.

E. *Нийлбэрийг нормчлох*

Мантисуудыг нэмээд эсвэл хасаад гарсан үр дүн нь орон ахьсан эсвэл орон зээлсэн байвал түүнийг дахин нормчлох шаардлагатай. Хэрвээ орон ахьсан байх юм бол экспонентийг нэгээр нэмэгдүүлж мантисын тасалыг нэг орноор ахиулах хэрэгтэй. Үүнийг нэмэгчийн гаралтанд carry гаралт гарсан эсэхийг шалгаж хялбар мэдэж болно. Харин хасах үйлдлийн дараа гарсан ялгавар нь орон зээлсний улмаас урдаа тэгтэй гарах юм бол мантисыг орон ахиулж нормчлох хэрэгтэй. Доорх хүснэгтэнд нэмэгчийн гаралтаас хамаарч нийлбэрийг нормчлох хэсгийн хийх үйлдлүүдийг харууллаа.

Carry	Экспонент болон ахлах орны тэгүүдийн харьцаа	Нийлбэрийн мантис	Нийлбэрийн экспонент
1	X	M=M » 1	E = E + 1
0	LZ < E	M=M « LZ	E = E – LZ
0	LZ > E	M=M « E	E = 0

Хүснэгт 3 Нийлбэрийг нормчлох нөхцлүүд

Нэмэх үйлдлийн үр дүнд орон ахьсан байвал мантисыг нормчлохын тулд баруун тийш нэг удаа шифт хийх хэрэгтэй. Таслал баруун тийш нэг орноор

шилжиж байгаа учраас экспонентийг нэгээр нэмэгдүүлнэ. Хэрвээ ялгаврын үр дүн нь тэгээр эхлэсэн байвал таслалын урд нэг гарах хүртэл мантисыг зүүн шифт хийх хэрэгтэй. Таслал зүүн тийш шилжиж байгаа учраас шилжсэн битийн тоогоор экспонентийг хорогдуулах хэрэгтэй. Хэрвээ Ахлах орны тэгүүдийн тоо нь экспонентийн утгаас бага байвал экспонентийн утгыг тэг болтол мантисын орныг ахиулна. IEEE 754 стандартад зааснаар экспонентийн хоёртын утга тэгтэй(-126) тэнцүү болсон үед мантис нь нормчлогдоогүй байдаг гэдэгийг үзсэн. Хэрвээ үүнээс их утгаар мантисыг орон ахиулвал алдаа гарна.

Зээлсэн оронгийн хэмжээ нь ялгаврын эхэнд байгаа тэгүүдийн тоотой тэнцүү байдаг тиймээс зээлсэн оронг тооцохдоо ахлах орны тэгүүдийг тоолдог хэлхээ (LZC - Leading zero counter)-г ашигладаг. LZC хэлхээ нь ялгавар бодогдож гарсны дараа ажиллах учраас нийт хэлхээний хугацааны хоцрогдлыг их болгодог. Үүнээс зайлсхийхийн
тулд ахлах орны тэгүүдийг таадаг хэлхээ (LZA - Leading zero anticipator)-г нэмэгчтэй зэрэгцээгээр холбож өгдөг. Энэ хэлхээ нь нэмэгчтэй харьцуулвал бага хугацааны хоцрогдолтойгоор ахлах орны тэгүүдийн тоог гаргаж өгөх бөгөөд нэмэгчтэй параллель ажилласнаар нийт хэлхээний
хугацааны хоцрогдлыг багасгах боломжийг олгодог.

Хэрвээ нэмэгчийн гаралтанд ахлах орны тэгүүдийг тоолдог хэлхээг холбох юм бол нийт хэлхээний хугацааны хоцрогдол энэ хоёр хэлхээний нийлбэр хугацааны хоцрогдлоор илэрхийлэгдэнэ. Харин ахлах орны тэгүүдийг таах хэлхээг нэмэгчтэй зэрэгцээгээр холбож өгөх юм бол нийт хэлхээний хугацааны хоцрогдол зэрэгцээ холбогдсон хоёр хэлхээний аль их хугацааны хоцрогдолтой хэлхээний хугацааны хоцрогдолтой ижил болно.

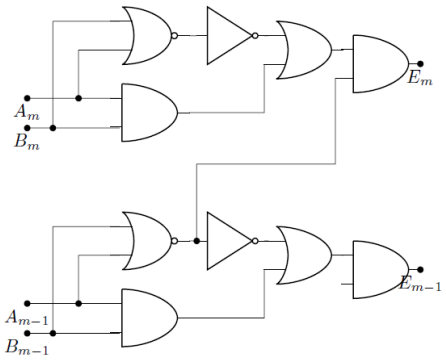
F. Leading Zero Anticipator - LZA хэлхээ

Ахлах орны тэгүүдийг таамаглах хэлхээ нь нэмээд гарсан үр дүнгийн эхний хэдэн бит нь тэг гарахыг нэмэгчээс хурднаар илрүүлэх зорилготой. Гаргах үр дүн нь нэмэгчтэй төстэй байх боловч LZA хэлхээ нь нэмэгчийг бодвол харьцангуй бага логик гэйт хэрэглэдэг. Энэ ажилд Хироаки Сузуки, Хироюки Моринака, Хироши Макино нарын дэвшүүлсэн LZA логик хэлхээг ашигласан. (3)Энэ хэлхээний оролтонд нэмэх гэж буй хоёр нэмэгдэхүүнээ холбож өгнө. Их нэмэгдэхүүнийг A, бага нэмэгдэхүүн буюу хасах үйлдлийн хувьд хоёр дахь нэмэгдэхүүний хоёртын гүйцээлтийг B, гаралтыг E гэж тэмдэглэвэл LZA хэлхээний гаралтын i-дүгээр бит нь дараах логикоор илэрхийлэгдэнэ.

$$E_i = \overline{A_i \oplus B_i}(A_{i-1} + B_{i-1}) \quad (3)$$

$$E_i = (A_i B + \bar{A}_i \bar{B}_i)(A_{i-1} + B_{i-1}) \quad (4)$$

Энэхүү илэрхийллийг хоёртын нэмэгчийн илэрхийллээс (4)гаргаж ирсэн бөгөөд •, + болон _ тэмдэглэгээнүүд нь харгалзан логик AND, OR болон XOR илэрхийлж байгаа болно. Энэхүү илэрхийллийг гүйцэтгэх хэлхээний нэг хэсгийг Зураг 3-т үзүүллээ.



Зураг 3 LZA хэлхээний хэсэг

G. Leading Zero Counter - LZC хэлхээ

LZA хэлхээ нь нэмэгчээс илүү хурднаар ахлах орны тэгүүдийг гаргаж өгдөг. LZA хэлхээ нь ахлах оронд байгаа тэгүүдийг тоолдог хэлхээ буюу Leading Zero Counter(LZC) хэлхээтэй заавал хамт хэрэглэгдэнэ (5). LZC-ийн хэлхээ нь ахлах орны тэгүүдийн тоонд харгалзах хоёртын тоон утгыг гаралтандаа гаргадаг комбинацын логик хэлхээ байна. Гиоргос Димитракопулосын дэвшүүлсэн ахлах орны тэгүүдийг тоолох хэлхээ нь бусад LZC хэлхээнүүдийг бодвол илүү хурдан болсон байна. Түүний дэвшүүлсэн 4 битийн LZC хэлхээний гаралтын битүүд нь дараах логик илэрхийллээр илэрхийлэгдэнэ.

$$\overline{Z_2} = A_7 + A_6 + A_5 + A_4 \quad (5)$$

$$\overline{Z_1} = A_7 + A_6 + \overline{A_5 A_4}(A_3 + A_2) \quad (6)$$

$$\overline{Z_0} = A_7 + \overline{A_6} A_5 + \overline{A_6 A_4} A_3 + \overline{A_6 A_4 A_2} A_1) \quad (7)$$

$$\overline{V_0} = A_7 + A_6 + A_5 + A_4 \quad (8)$$

II. Үр дүн

Энэ ажлаар IEEE-754 стандартын хөвөгч цэгтэй тооонуудын нэмэгчийн дизайныг гаргаж туршиж үзсэн. Туршилтыг Spartan 3E FPGA хавтан дээр хийсэн болно. Нэмэгчийн дизайныг VHDL хэл дээр гаргасан бөгөөд Xilinx ISE 12.2 програмыг ашигласан. Нэмэх үйлдлийн эхэнд экспонентүүдийг жишиж их багаар нь ялгаж өгснөөр дараагийн үйлдлүүдийг хурдан хийх боломжтой болсон. Нэмэгдэхүүнүүдийн харьцааг мэдсэнээр нэмэлт мультиплексер, харьцуулагчийн

хэлхээ шаардлагагүй болсон бөгөөд төдий хэмжээгээр хэлхээний хугацааны хоцрогдлыг багасгаж өгсөн. Хөвөгч цэгтэй тооны нэмэгчийг Spartan 3E FPGA хавтан дээр хийхэд хир зай үзүүлснийг Зураг 4-т үзүүллээ.

PostNorm Project Status			
Project File:	Feb13.xise	**Parser Errors:**	No Errors
Module Name:	PostNorm	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 12.2	**• Warnings:**	118 Warnings (0 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	113	4656	2%
Number of 4 input LUTs	214	9312	2%
Number of bonded IOBs	70	232	30%

Зураг 4 Синтезийн үр дүн

Синтезийн үр дүнд гарсан хөвөгч цэгтэй тооны нэмэгч маань Spartan 3E FPGA хавтангийн 4656 slice-ийн 113 буюу 2%, 9312 4 ороллтой LUT-ийн 214 буюу 2%, 232 оролт гаралтын блокын 70 буюу 30%-ийг эзлэж байна.

III. АШИГЛАСАН МАТЕРИАЛ

[1] Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic. **Kahan, Prof. W.** University of California : с.н., 1997

[2] What Every Computer Scientist Should Know About Floating Point Arithmetic. **Goldberg, David.** Computing Surveys., 1991 оны.

[3] Leading-Zero Anticipatory Logic for High-Speed Floating Point Addition. **Hiroaki Suzuki, Hiroyuki Morinaka, Hiroshi Makino, Yasunobu Nakase.** IEEE JOURNAL OF SOLID-STATE CIRCUITS,, 1996

[4] IEEE 754 стандартын хөвөгч цэгтэй тооны нэмэгч. **Т.Зулхүү, Б.Золбоо, М.Баярпүрэв.** Улаанбаатар, 2013.

[5] Програмчлагддаг логик чипэнд суурилсан 32 бит Floating point тооны үржүүлэгчийн дизайн. **Ж.Баттогтох, Б.Зориг, М.Доржжамц.** Улаанбаатар : ММТ 2013

[6] Theory and Practice of Online Learning, Athabasca University 2010. ISBN: 0-919737-59-5. pp 92.

[7] A Very Fast and Low Power Incrementer and Decrementer Circuit. **Samiappa Sakthikumaran, S. Salivahanan, V. S. Kanchana Bhaaskaran, V. Kavinilavu.**

[8] A 40MFLOPS 32-bit floating-point processor. **S. Komori, H. Takata, T. Tamura, F. Asai, T. Ohno, O. Tomisawa, T.** ISSCC Dig. Tech. Papers, Feb. 1989