ساخت ALU

در متلب

رحمت اله انصاری

نیما مظلوم

درس معماری کامپیوتر

دکتر ملک زاده

دانشگاه حکیم سبزواری

رشته مهندسی کامپیوتر

خرداد 1401

بسم الله الرحمن الرحیم

مشخصات

عنوان پروژه : ساخت ALU با شش operation در متلب

نویسندگان:

نام و نام خانوادگی : رحمت اله انصاری

شماره دانشجویی : 9912377331

ایمیل : Rahmat2022a@gmail.com

نام و نام خانوادگی : نیما مظلوم

شماره دانشجویی : 9922377499

ایمیل : mazloomnima79@yahoo.com

درس معماری کامپیوتر

استاد : دکتر ملک زاده

دانشگاه حکیم سبزواری، سبزوار

رشته مهندسی کامپیوتر

دانشکده مهندسی کامپیوتر و برق

نوشته شده در خرداد سال ۱۴۰۱

فهرست مطالب

مقدمه 5

ساختار فایل پروژه 5

فصل اول آموزش نصب متلب 7

دانلود 8

فایل فشرده 13

درایو مجازی 15

شروع رسمی نصب 16

لایسنس 17

فایل اجرایی 23

کرک 24

اطمینان از نصب درست 28

فصل دوم آموزش کار با سیمولینک 29

رفتن به سیمولینک 30

کتابخانه ها 32

ساخت یک مولتی پلکسر 41

کتابخانه Math Operations 44

تبدیل مبنای اعداد با گیت های سیمولینک 49

فصل سوم تشریح فایل متلب پروژه 53

شکل پروژه 54

بلوک ALU 55

ورودی اول AND 56

ورودی دوم OR 56

ورودی سوم و چهارم جمع و تفریق 57

ورودی پنجم و ششم 60

طریقه کارکردن مولتی پلکسر 61
بلوک mul_div 63
حالت کلی 70
فصل چهارم پروژه با گیت های دستی 71
شکل پروژه 72
بلوک ALU 73
بلوک Full Adder 74
بلوک mul_div 74
بلوک mul 75
جمع زننده ۴ بیتی 78
بلوک div 78
بلوک SB 81
تمام تفریق گر 81
بلوک مولتی پلکسر 84
فصل پنجم پروژه تماما بلوکی 85
شکل پروژه 86
بلوک ALU 87
تشریح عملیات ضرب 87
تشریح عملیات تقسیم 89
بلوک CAS 92
فصل ششم نمونه ورودی و خروجی 93
نمونه های پروژه با گیت های آماده و دارای بلوک تقسیم ضرب 94
بررسی AND 94
بررسی OR 96
بررسی جمع 98

بررسی تفریق 100
بررسی ضرب 102
بررسی تقسیم 104
نمونه های پروژه با گیت های دستی و دارای بلوک تقسیم ضرب 107
بررسی AND 107
بررسی OR 109
بررسی جمع 111
بررسی تفریق 113
بررسی ضرب 115
بررسی تقسیم 117
نمونه های پروژه تماما بلوکی با گیت های دستی 119
بررسی AND 119
بررسی OR 121
بررسی جمع 123
بررسی تفریق 125
بررسی ضرب 127
بررسی تقسیم 129
پیوست ها 131

مقدمه

سلام، وقت بخیر . . .

ما در این مستند با استفاده از نرم افزار MATLAB R2018b اقدام به طراحی یک ALU یا واحد محاسباتی ریاضی کرده ایم که با توجه به داده های ورودی (دو عدد ۴ بیتی) و مقدار operation code خروجی مورد نظر (که ۴ بیتی است) را به ما می دهد.

جدول operation code را در مقابل می بینیم . . .

operation	Code
AND	000
OR	001
Addition	010
Subtraction	011
Multiplication	100
Division	101

البته با توجه به گفته ها که در بخش پیوست ها موجود است گفته شده که ورودی استاندارد خواهد بود لذا به جز مواردی خاص مانند تقسیم بر صفر از ایجاد خطا جلوگیری نشده . . .

در صورت وجود هرگونه سوال و ابهام در خصوص این پروژه میتوانید با ایمیلی که در قسمت مشخصات است در ارتباط باشید.

با آرزوی موفقیت . . .

انصاری

ساختار فایل پروژه

Name	Date modified	Type	Size
other	6/17/2022 6:33 AM	File folder	
Record	6/17/2022 10:53 AM	File folder	
word	6/17/2022 11:19 AM	File folder	
Documentary.pdf	6/17/2022 10:13 AM	Microsoft Edge PD...	12,701 KB
project.slx	6/15/2022 5:43 PM	Simulink Model	40 KB

درون فایل فشرده به صورت بالا می باشد که پروژه اصلی فایل project.slx است. فایل pdf مستند هم Documentary.pdf می باشد. فایل ورد مستند در پوشه word می باشد. جهت اطمینان فونت های استفاده شده در word کنار این فایل گذاشته شده است. فیلم مستند به همراه فایل txt نمونه ورودی ها که در این رکورد از آن استفاده شد هم در پوشه Record می باشند.

دو فایل دیگر متلب مورد استفاده در این مستند هم در پوشه other می باشند.

About Word file:

Pages: 134

Words: 8,884

فصل اول

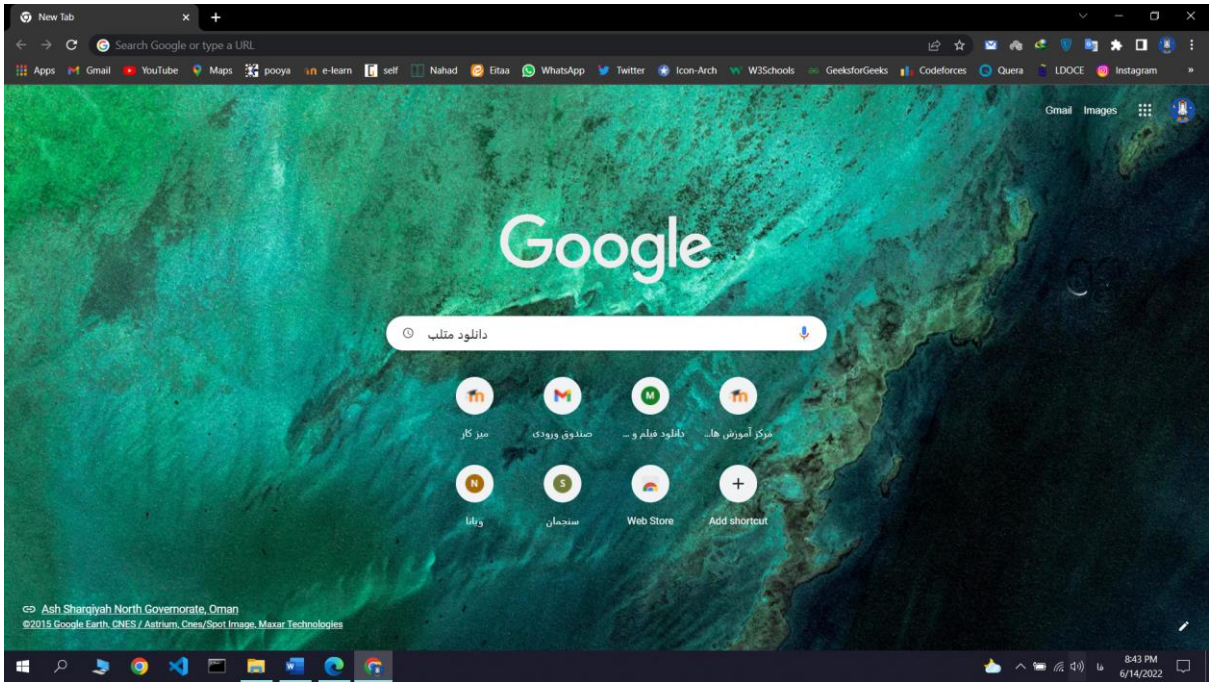
آموزش نصب متلب

در ساخت این پروژه ابزار ما نرم افزار متلب است. متلب (MATLAB) یک محیط نرم‌افزاری برای انجام محاسبات عددی و یک زبان برنامه‌نویسی نسل چهارم است. واژه متلب هم به معنی محیط محاسبات رقمی و هم به معنی خود زبان برنامه‌نویسی مربوطه است که از ترکیب دو کلمه MAT یعنی Matrix یا ماتریس و LAB یا Laboratory یا آزمایشگاه ایجاد شده‌است. این نام حاکی از رویکرد ماتریس محور برنامه است، که در آن حتی اعداد منفرد هم به عنوان ماتریس در نظر گرفته می‌شوند.

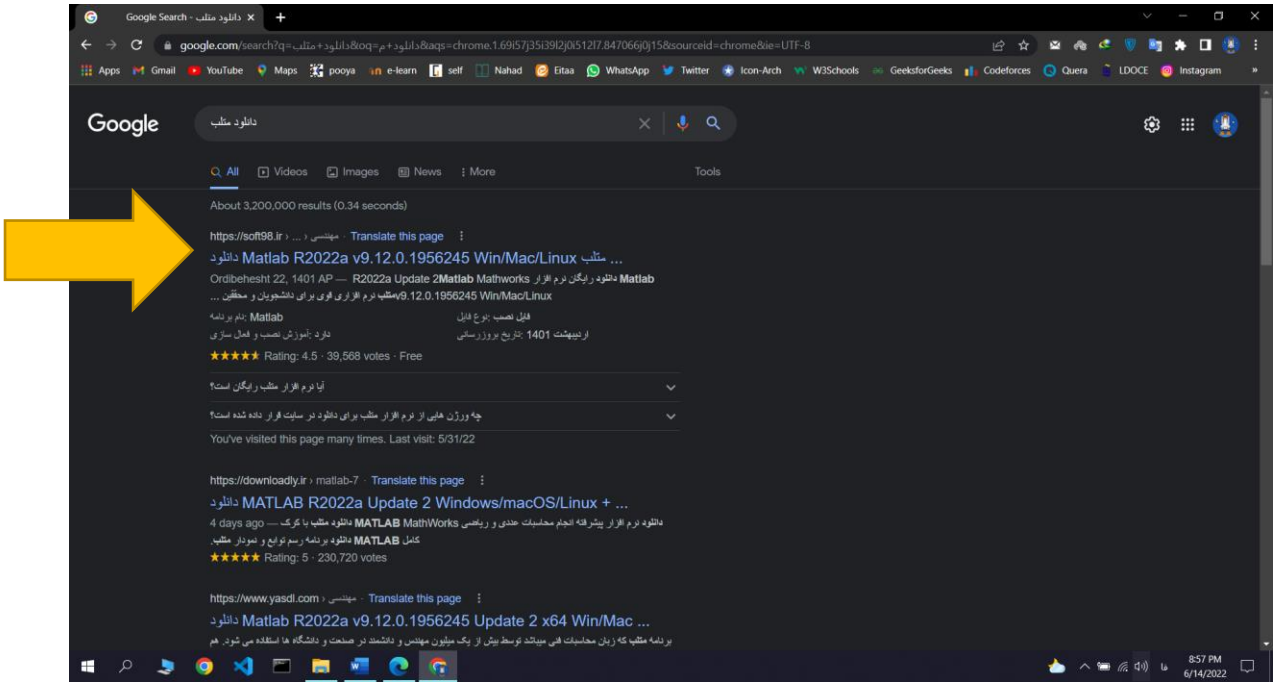
در این فصل به نصب برنامه متلب از دانلود تا اجرا خواهیم پرداخت.

برای نصب آسان تر بهتر است چند نرم افزار جانبی از جمله وین رار و آی دی ام را هم نصب کنید. اگر ویندوز شما ۱۰ یا ۱۱ نیست لازم است نرم افزار درایو مجازی Mount را هم نصب کنید.

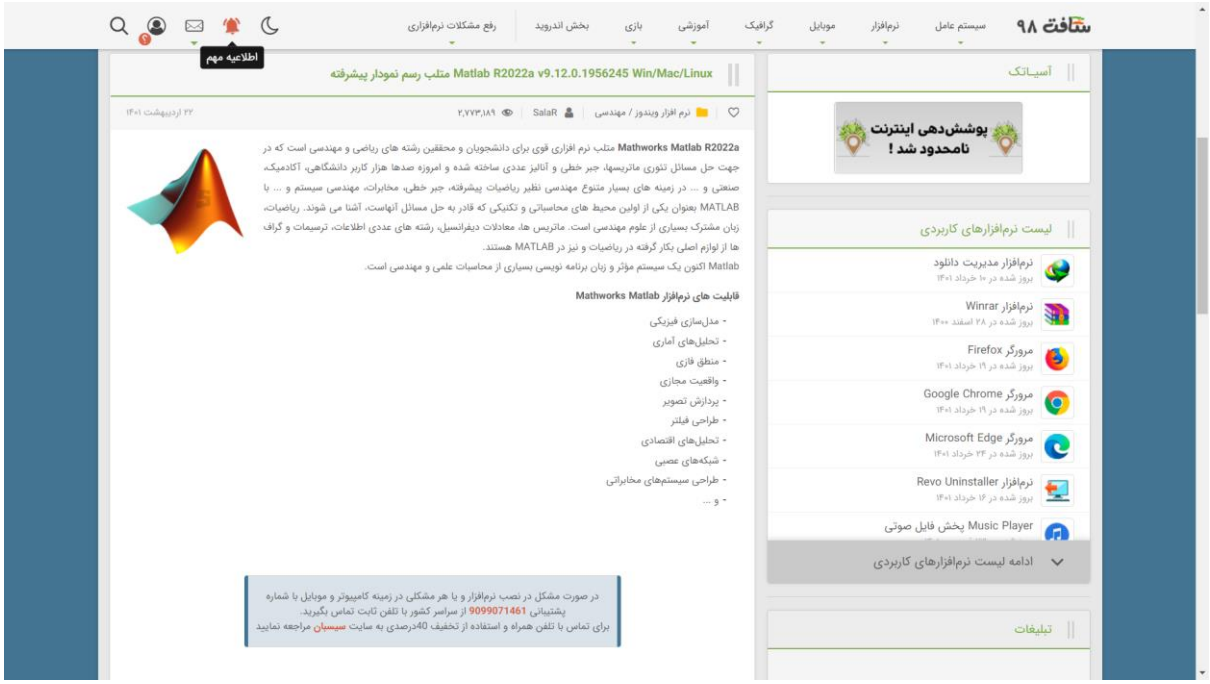
دانلود



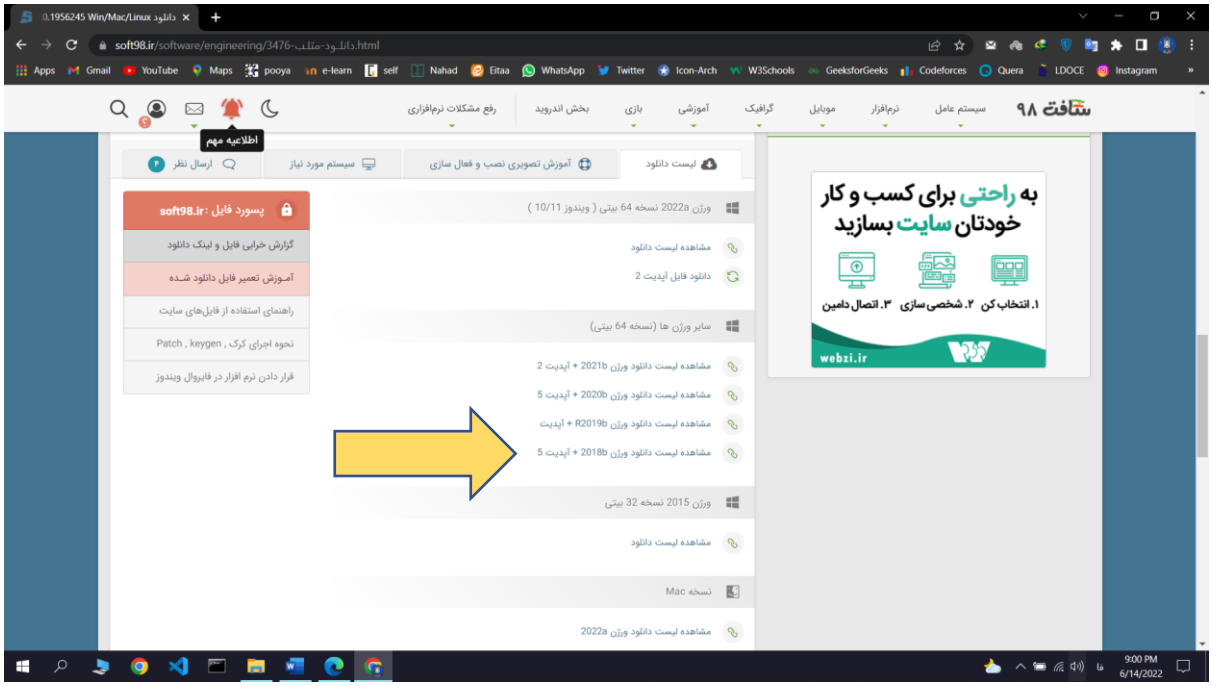
ابتدا در گوگل همانند بالا به دنبال دانلود متلب میرویم.



به سایت اول که سافت ۹۸ است می رویم.



با اسکرول صفحه به محتوای اصلی می رسیم.



ما در این آموزش نسخه ۲۰۱۸ بتای متلب را دانلود و نصب می کنیم پس فایل مربوطه را دانلود می کنیم.

با زدن روی لینک به صفحه ی بالا بر میخوریم که لینک دانلود های متلب ۲۰۱۸ را به ما نشان می دهد.

دانلود فایل کرک یا آپدیت اختیاری است و ما در این مقاله از آن ها استفاده نمی کنیم.

همانطور که نوشته شده حجم کامل فایل فشرده 12.2GB خواهد بود.

ترجیها با نرم افزار آی دی ام (دانلود منیجر) اقدام به دانلود کنید. این نرم افزار را می توانید از لینک زیر دانلود کنید.

IDM (internet download manager) download link: http://p30.in/44201

اگر با این نرم افزار دانلود کنید این فایل ها در پوشه دانلود ها و در بخش فایل های فشرده نگه داری می شود

C:\Users\[Your system name]\Downloads\Compressed

خود سایت سافت ۹۸ آموزشی به صورت زیر برای نصب داده است:

- *دسترسی به اینترنت را قطع کنید.*
- *فایل را از حالت فشرده خارج کنید.*
- *فایل با فرمت iso را توسط نرم افزارهای درایور مجازی Mount و نصب کنید.*
- *فایل Setup را اجرا کنید و Use a file instalation key را بزنید و نرم افزار را به کمک سریال زیر نصب کنید*

09806-07443-53955-64350-21751-41297

- *فایل کرک جدید را از این لینک دانلود کنید.*
- *پوشه R2018b موجود در پوشه Crack را در مسیر*

C:|Program Files|MATLAB کپی و جایگزین پوشه قبلی کنید.

- *نرم افزار را اجرا کنید و گزینه Active Manually Without internet را بزنید.*
- *در این مرحله دکمه Browse را بزنید فایل license_standalone.lic موجود در پوشه Crack را انتخاب کنید.*

- *نرم افزار را اجرا کنید.*

همچنین برای آپدیت هم به صورت زیر داریم:

- *آپدیت سایر ورژن ها به همین صورت میباشد، تنها ورژن و شماره آپدیت باید تغییر کند*
- *فایل R2020b_Update_3_Windows_package از حالت فشرده خارج کنید و در داریو C: کپی کنید.*

  مثال (C:|R2020b_Update_3_Windows_package)

- *به Run در منوی Start بروید (کلید ترکیبی Win+R) و دستور cmd را تایپ کنید.*
- *در پنجره Command Prompt دستور زیر را تایپ یا کپی کنید*

```
cd C:\Program Files\Polyspace\R2020b\bin\win64
```

- *و بعد دستور زیر را تایپ کنید (در صورتیکه بعد از تایپ دستور زیر، پیام خطایی دریافت نکردید، نرم افزار در حال بروزرسانی میباشد)*

```
update_installer.exe -updatepackage
C:\R2020b_Update_3_Windows_package\
```

- *بعد از بروزرسانی فایل libmwlmgrimpl.dll موجود در پوشه Crack را در مسیر زیر کپی کنید*

```
C:\Program Files\Polyspace\R2020b\bin\win64\matlab_startup_plugins\lmgrimpl
```

- *نرم افزار را اجرا کنید.*

System Requirements 2018

- OS: Windows 8.1 x64, Windows 8 x64, Windows 7 Service Pack 1 x64, Windows Vista Service Pack 2 x64, Windows XP x64 Edition Service Pack 2, Windows Server 2012, Windows Server 2008 R2 Service Pack 1, Windows Server 2008 Service Pack 2, Windows Server 2003 R2 Service Pack 2
- Any Intel or AMD x86 processor supporting SSE2 instruction set
- 1 GB for MATLAB only, 3–4 GB for a typical installation
- 1024 MB (At least 2048 MB recommended)

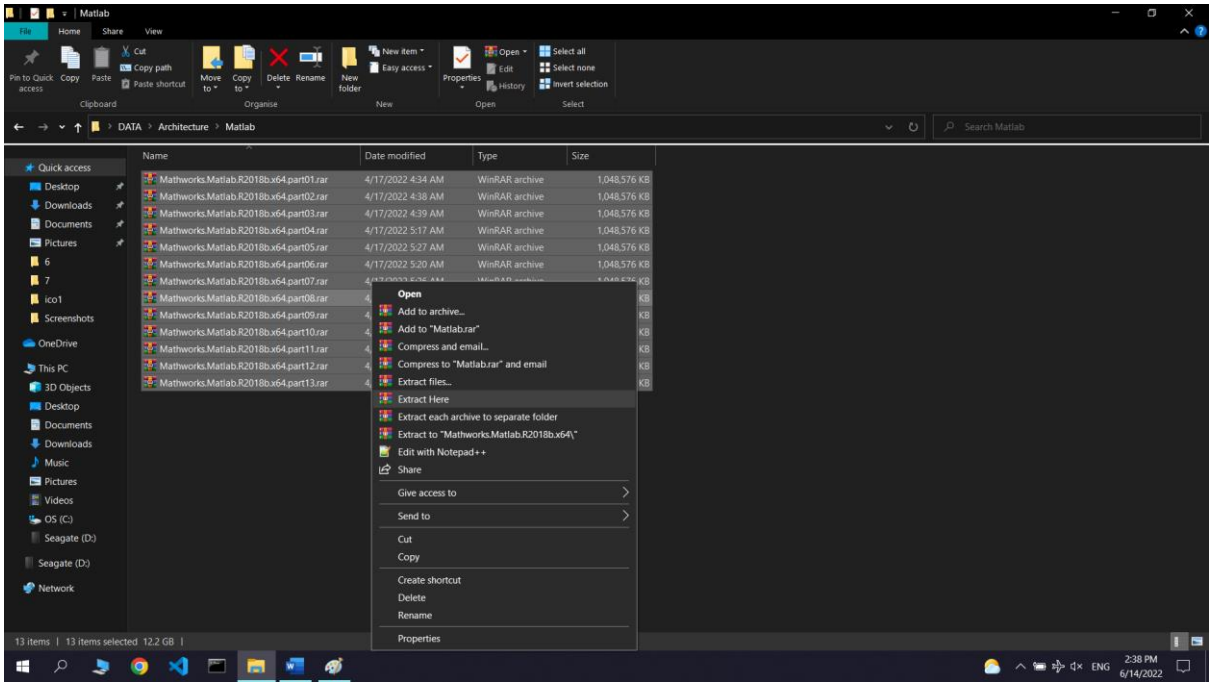
مشخصات سیستم مورد نیاز متلب هم به صورت بالاست.

فایل فشرده

Name	Date modified	Type	Size
Mathworks.Matlab.R2018b.x64.part01.rar	4/17/2022 4:34 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part02.rar	4/17/2022 4:38 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part03.rar	4/17/2022 4:39 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part04.rar	4/17/2022 5:17 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part05.rar	4/17/2022 5:27 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part06.rar	4/17/2022 5:20 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part07.rar	4/17/2022 5:26 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part08.rar	4/17/2022 5:29 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part09.rar	4/17/2022 5:29 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part10.rar	4/17/2022 5:20 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part11.rar	4/17/2022 5:14 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part12.rar	4/17/2022 5:13 AM	WinRAR archive	1,048,576 KB
Mathworks.Matlab.R2018b.x64.part13.rar	4/17/2022 5:00 AM	WinRAR archive	258,807 KB

پس از دانلود همانند بالا فایل های فشرده را داریم.

همانطور که میبینید متلب ۲۰۱۸ شامل ۱۳ فایل فشرده است.

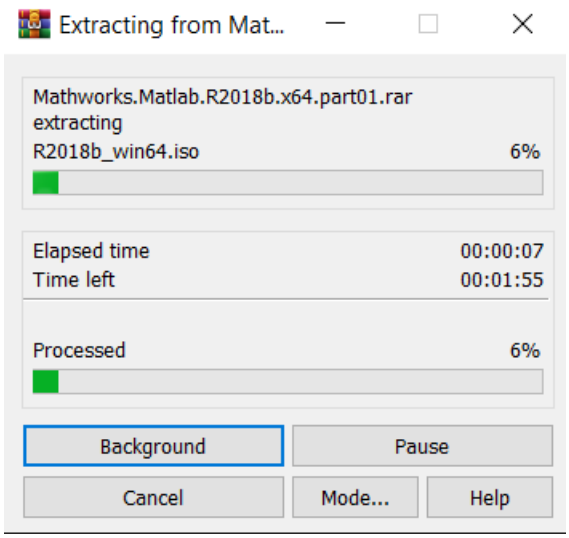


ما برای باز کردن فایل های فشرده از نرم افزار وین رار استفاده کردیم. همه فایل های فشرده را انتخاب و گزینه استخراج به اینجا یا Extract Here را میزنیم.

لینک دانلود نرم افزار وین رار را در پایین قرار داده ام.

WinRar download link: https://soft98.ir/software/compress/21-winrar-full.html

در پایین استخراج فایل فشرده با این نرم افزار را می بینیم.



Name	Date modified	Type	Size
Crack.rar	9/20/2018 8:13 AM	WinRAR archive	62 KB
R2018b_win64.iso	9/20/2018 1:17 PM	Disc Image File	12,610,954 KB
readme.txt	9/20/2018 1:16 PM	Text Document	2 KB
Soft98.iR	3/11/2013 4:02 AM	Internet Shortcut	1 KB

پس از استخراج فایل های بالا را داریم. همچنین به فایل های قبلی نیازی نداریم.

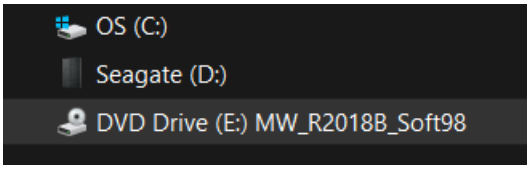
درایو مجازی

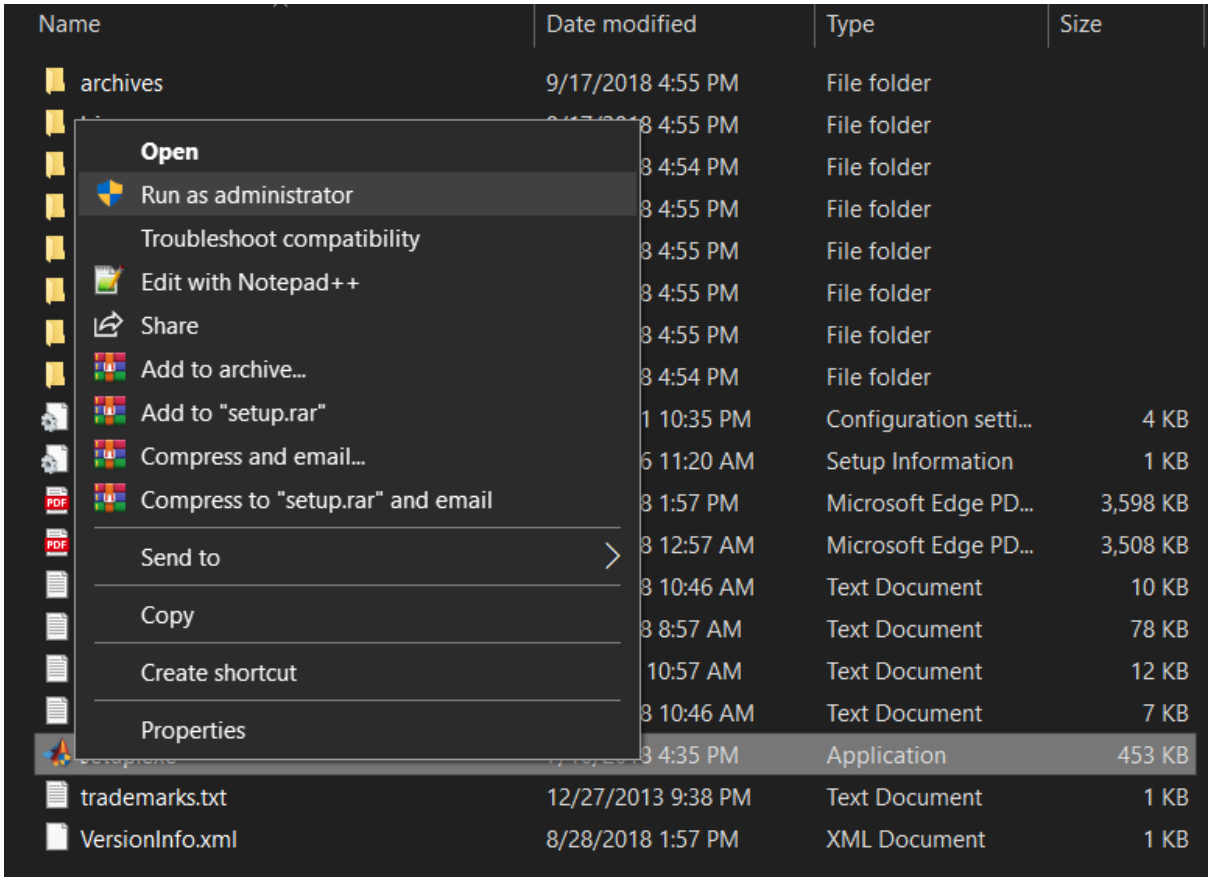
Name	Date modified	Type	Size
archives	9/17/2018 4:55 PM	File folder	
bin	9/17/2018 4:55 PM	File folder	
help	9/17/2018 4:54 PM	File folder	
java	9/17/2018 4:55 PM	File folder	
resources	9/17/2018 4:55 PM	File folder	
sys	9/17/2018 4:55 PM	File folder	
ui	9/17/2018 4:55 PM	File folder	
utils	9/17/2018 4:54 PM	File folder	
activate.ini	3/20/2011 10:35 PM	Configuration setti...	4 KB
autorun.inf	6/16/2006 11:20 AM	Setup Information	1 KB
install_guide.pdf	8/21/2018 1:57 PM	Microsoft Edge PD...	3,598 KB
install_guide_ja_JP.pdf	8/20/2018 12:57 AM	Microsoft Edge PD...	3,508 KB
installer_input.txt	7/27/2018 10:46 AM	Text Document	10 KB
license_agreement.txt	7/27/2018 8:57 AM	Text Document	78 KB
patents.txt	8/6/2018 10:57 AM	Text Document	12 KB
readme.txt	7/27/2018 10:46 AM	Text Document	7 KB
setup.exe	7/10/2018 4:35 PM	Application	453 KB
trademarks.txt	12/27/2013 9:38 PM	Text Document	1 KB
VersionInfo.xml	8/28/2018 1:57 PM	XML Document	1 KB

اگر ویندوز ۱۰ یا ۱۱ داشته باشیم با دوبار کلیک کردن روی فایل ایزو این فایل باز شده و درایو مجازی هم به طور خودکار ساخته می شود.

اگر با اینکار باز نشد یا نسخه دیگری از ویندوز را دارید باید فایل را با استفاده از نرم افزارهای ساخت درایو مجازی Mount کنید.

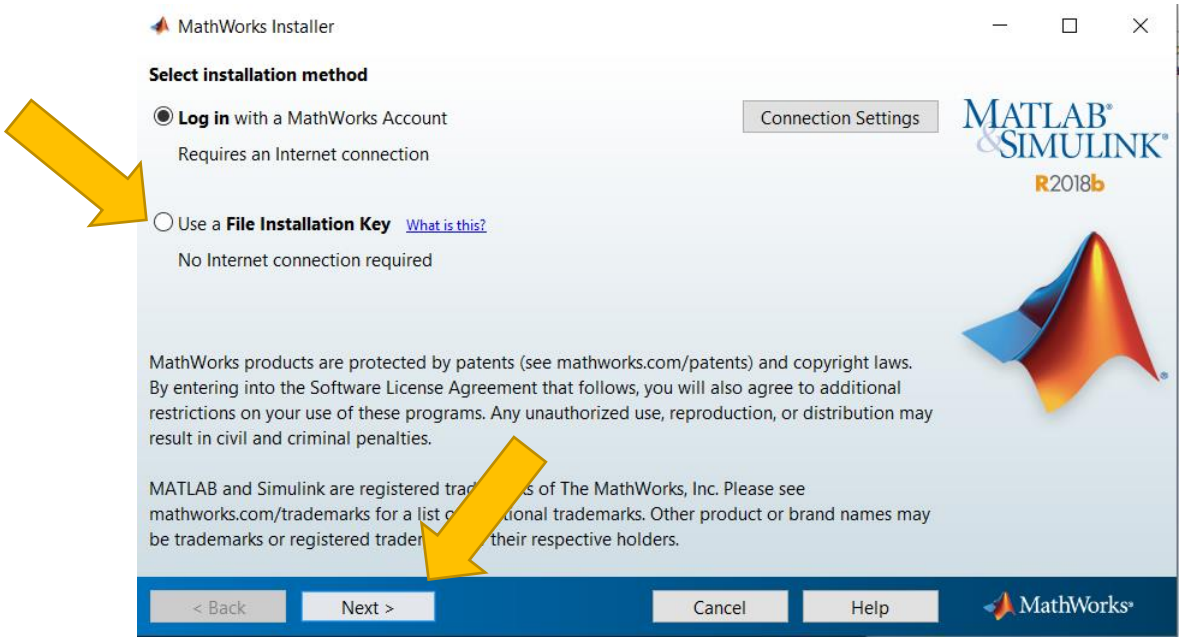
در قسمت درایوها پس از باز کردن فایل ایزو درایو مجازی ساخته شده را میبینیم.





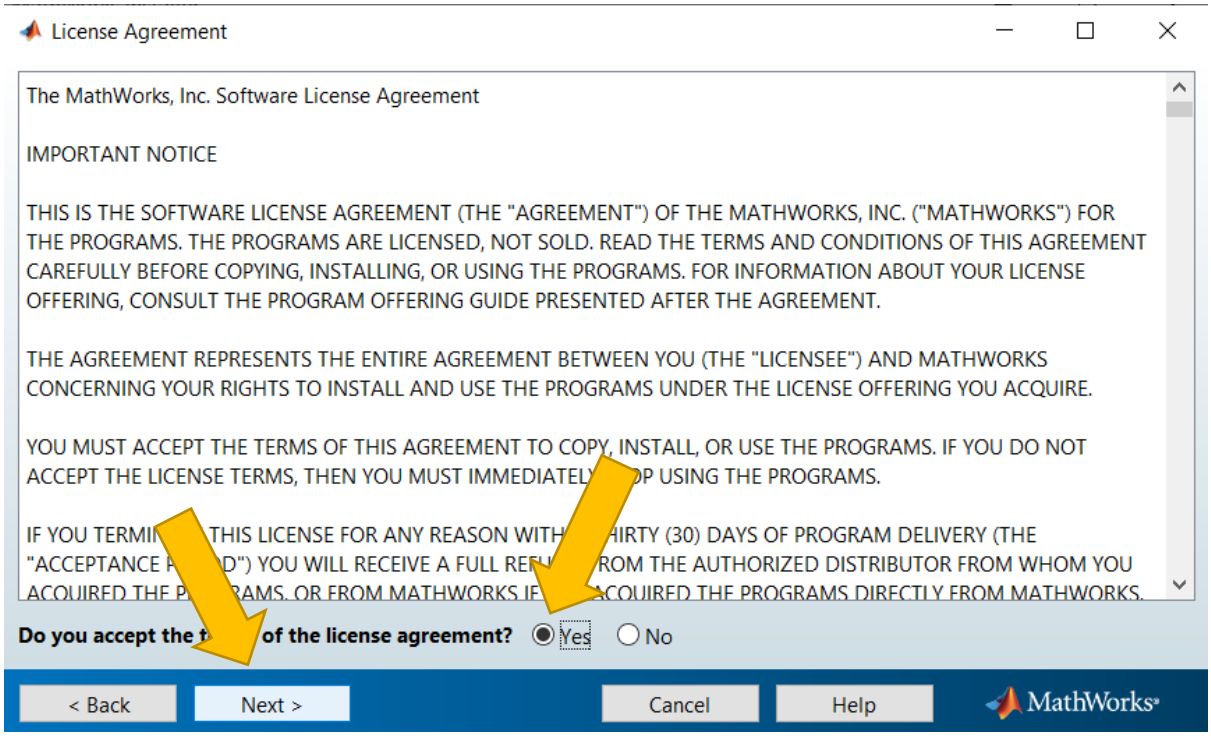
شروع رسمی نصب

پس از باز کردن فایل ایزو بر روی setup.exe راست کلیک می کنیم و بر روی Run as administrator کلیک می کنیم تا شروع رسمی نصب انجام شود.



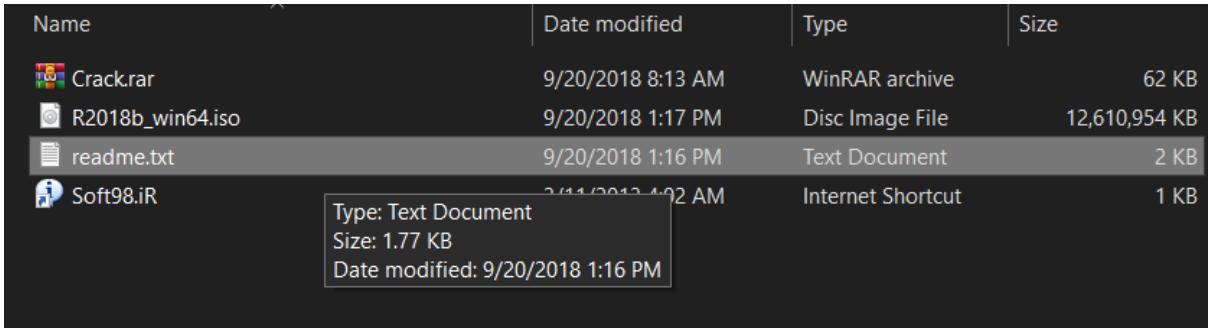
در مرحله بعد ویندوز ممکن است در مورد یک مجوز امنیتی تایید ما را بگیرد که با زدن Yes اجازه این کار را به آن می دهیم.

سپس به پنجره بالا میرسیم که ابتدا Use a File Installation Key را میزنیم و سپس Next را میزنیم.

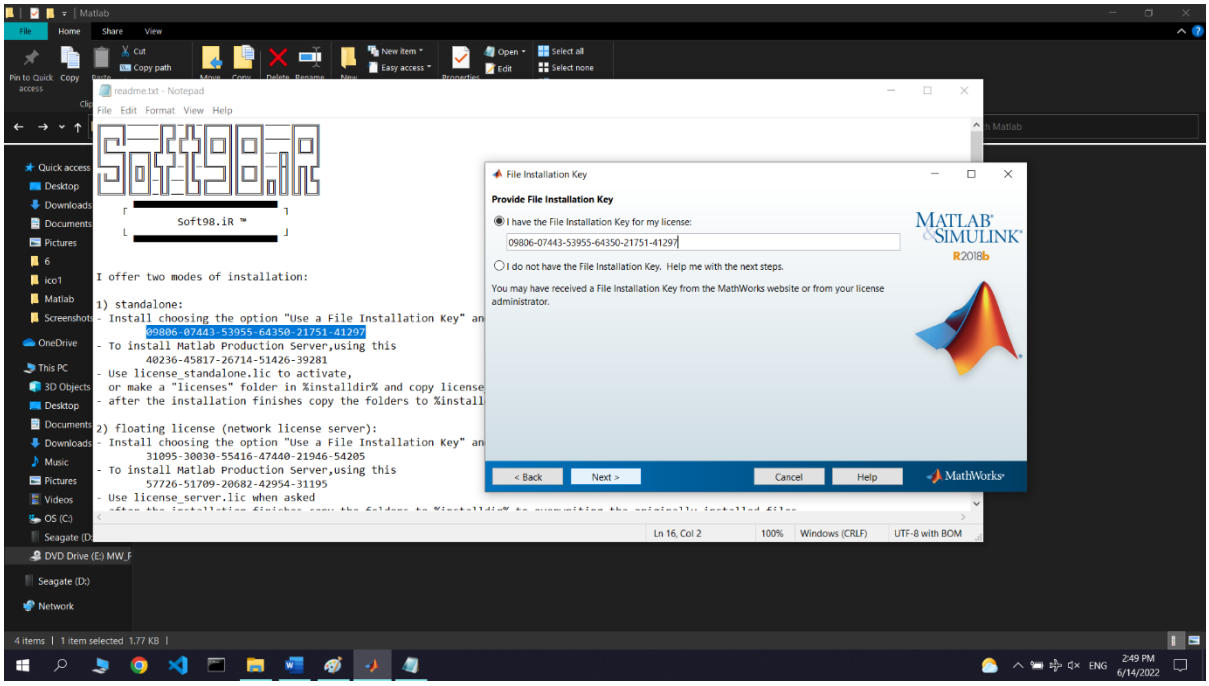


در مرحله بعدی هم با قبول کردن License Agreement و زدن next به مرحله بعد میرویم.

لایسنس



سپس فایل متنی بالا را باز میکنیم.

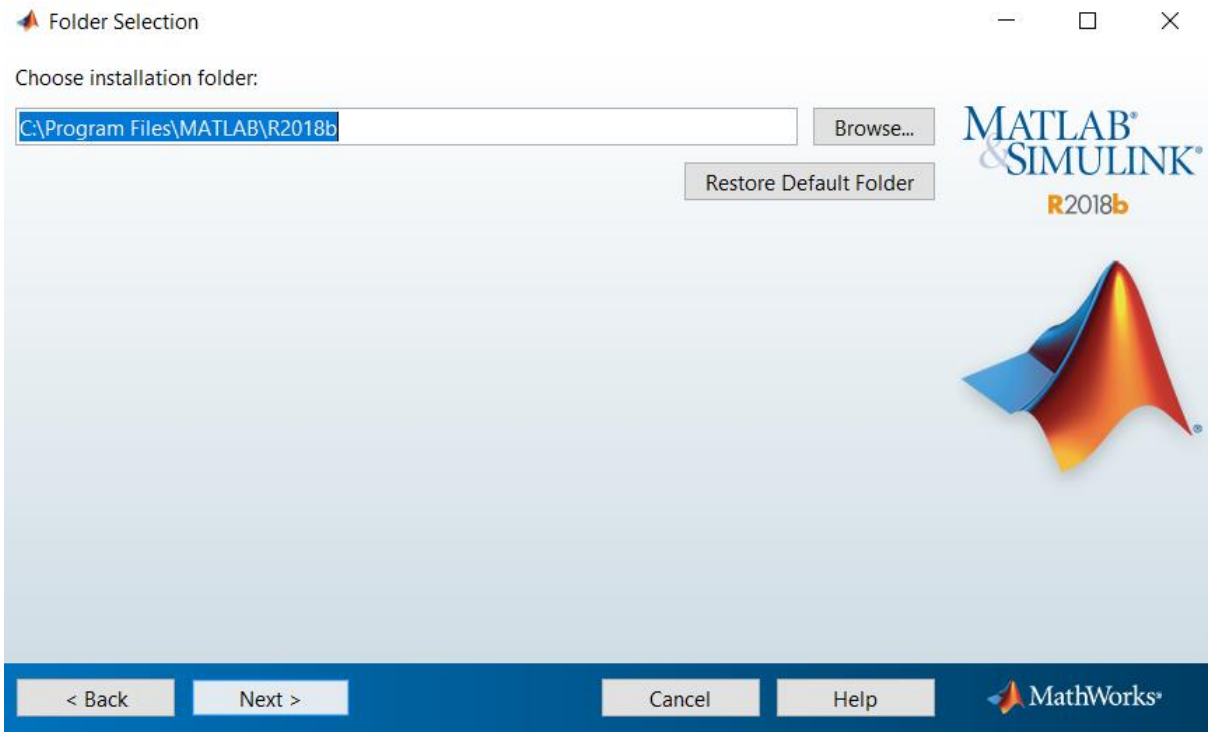


در این مرحله با توجه به مکان نشان داده شده شماره سریال را وارد و next را میزنیم.

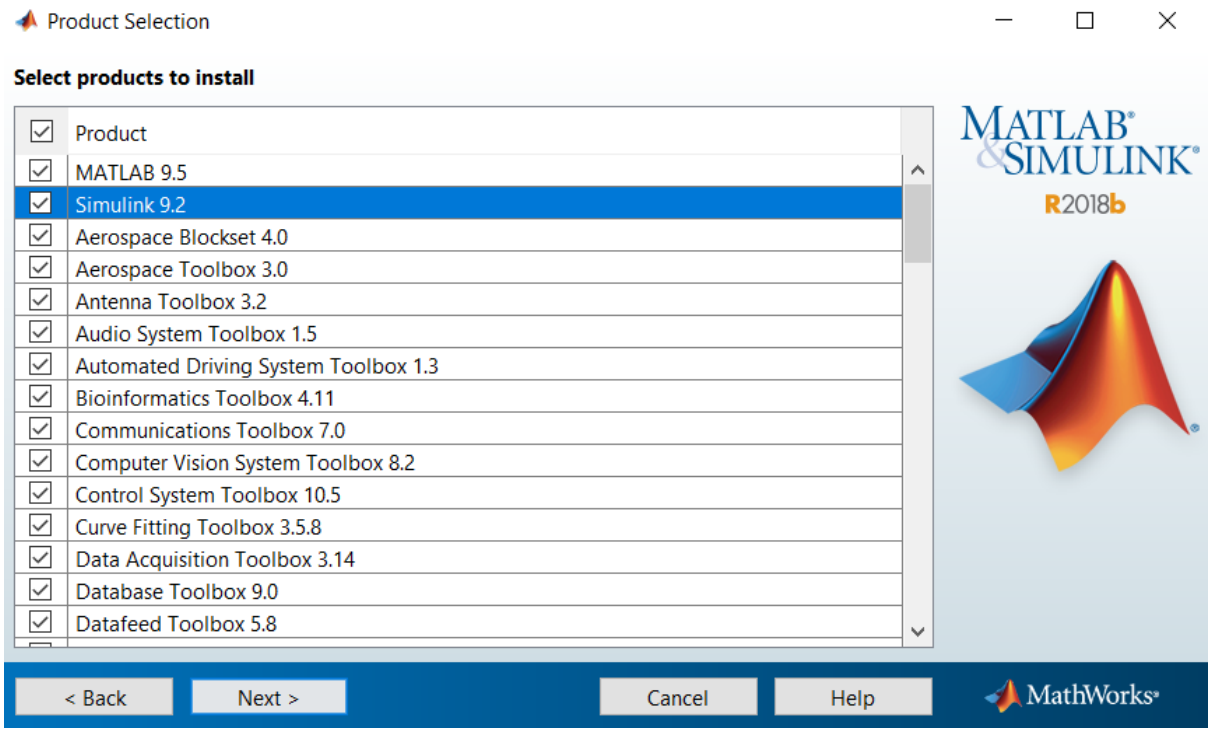
در فایل readme.txt به دنبال لایسنس برنامه می گردیم. (برنامه ما برای سرور نیست)

- Install choosing the option "Use a File Installation Key" and supply the following FIK

09806-07443-53955-64350-21751-41297



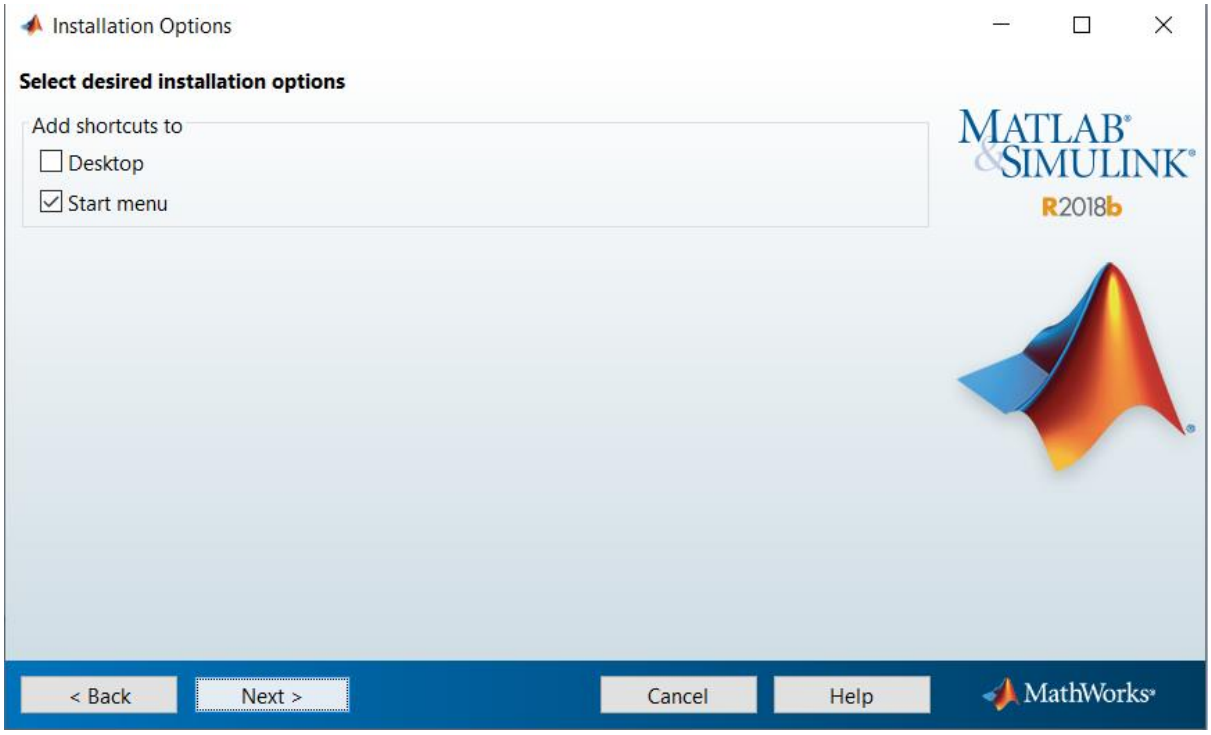
در مرحله بعد هم مکان نصب متلب را انتخاب میکنیم. درایوی را انتخاب کنید که فضای موجود برای نصب را داشته باشد.



در این مرحله هم زیر برنامه هایی که میخواهیم نصب کنیم را باید انتخاب کنیم. اگر product را انتخاب کنیم تمامی زیر برنامه ها نصب می شوند.

سپس next را وارد می کنیم.

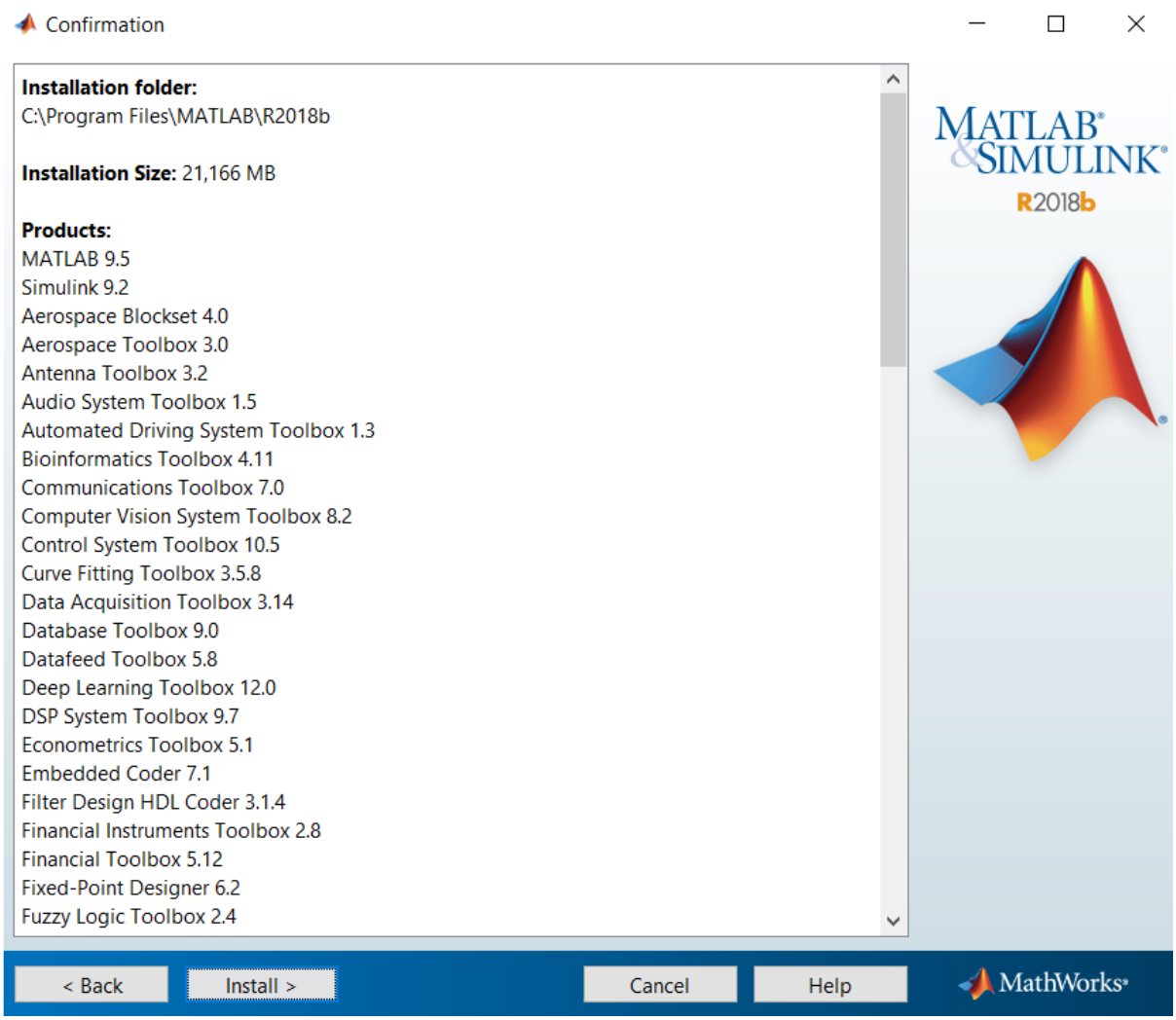
مهم ترین زیر برنامه برای ما Simulink 9.2 است.



در اینجا هم اگر بخواهیم نرم افزار در دسکتاپ هم shortcut داشته باشد تیک آن را فعال میکنیم.

گزینه دیگر Start menu است که پیشنهاد می شود آن را برنداریم چرا که نرم افزار در منوی استارت نمایش داده شود.

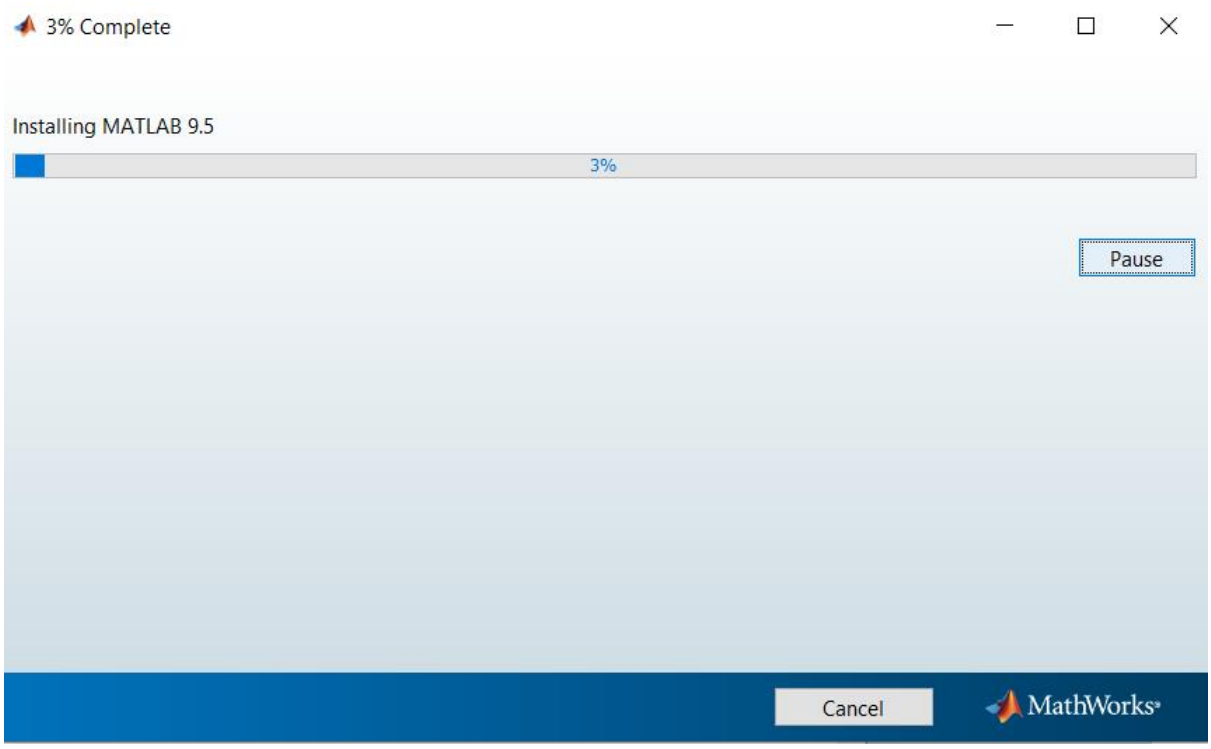
سپس next را میزنیم.



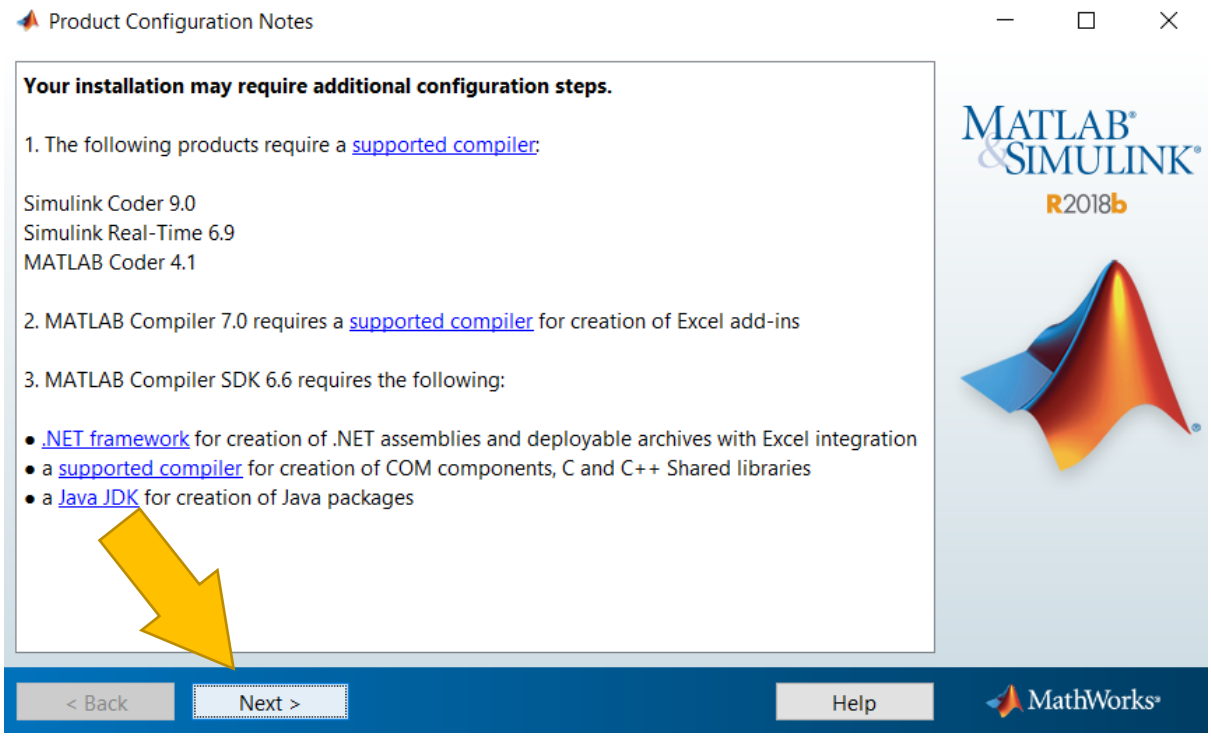
در اینجا هم نمایی از تنظیماتمان که تا اینجا شامل مکان نصب و سایز اشغالی و محصولات نصب شده را نشان می دهد.

همانطور که می بینید فضای مورد نیاز برای نصب کامل متلب 21,166 MB خواهد بود.

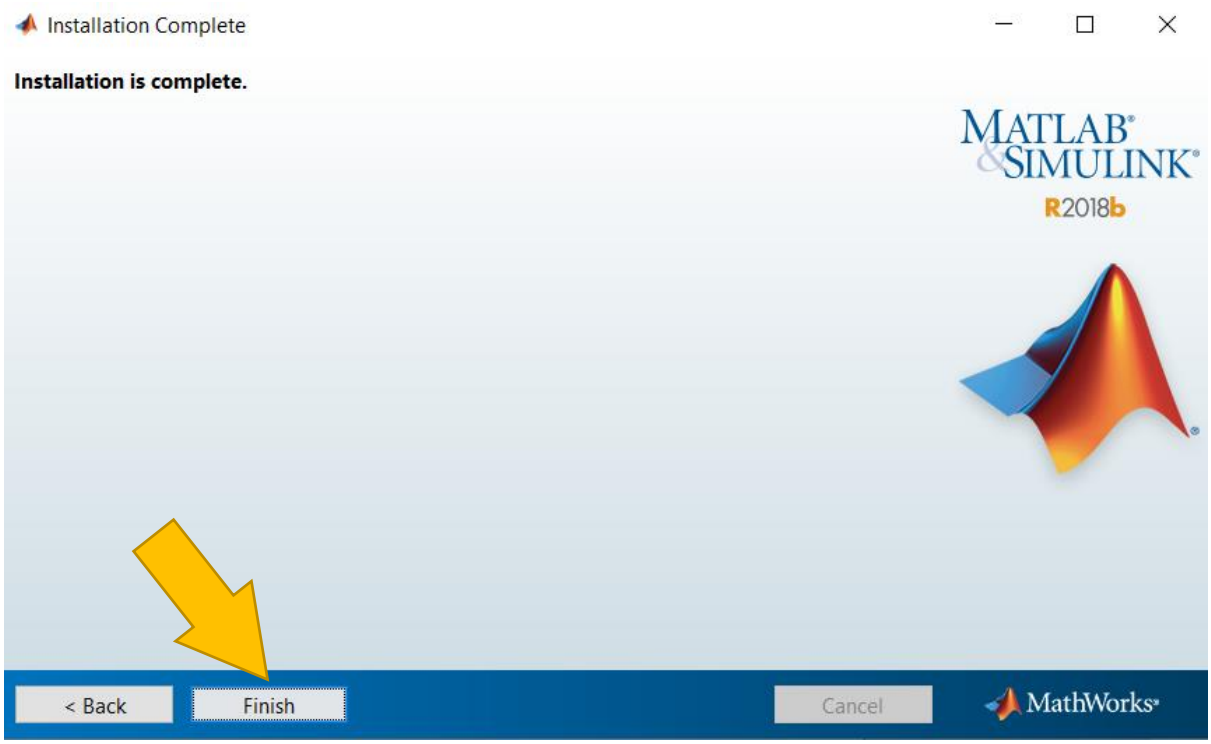
با زدن install برنامه نصب خواهد شد.



این مرحله ممکن است کمی طول بکشد.

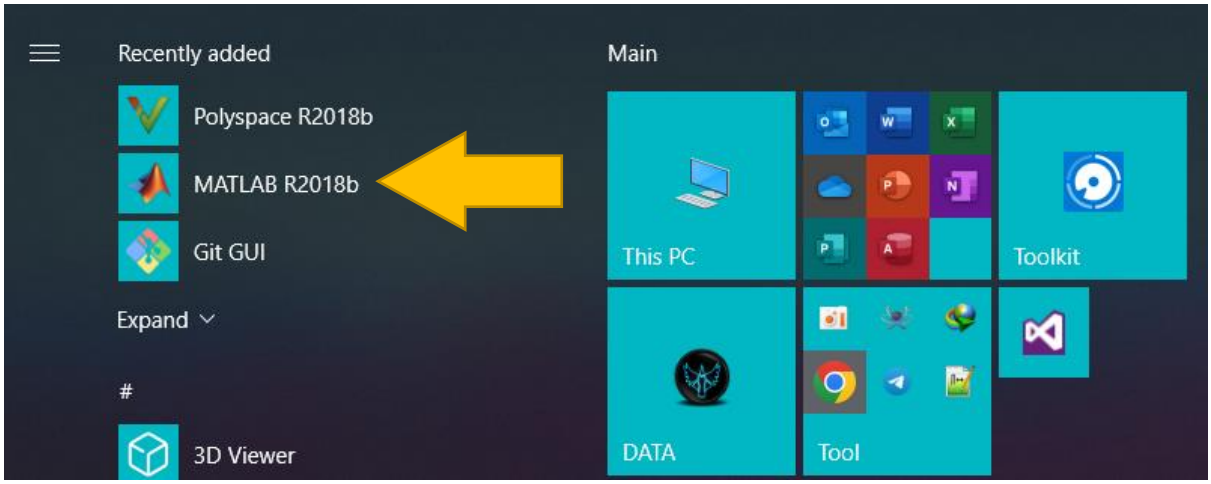


پس از پایان مرحله قبل next را میزنیم.



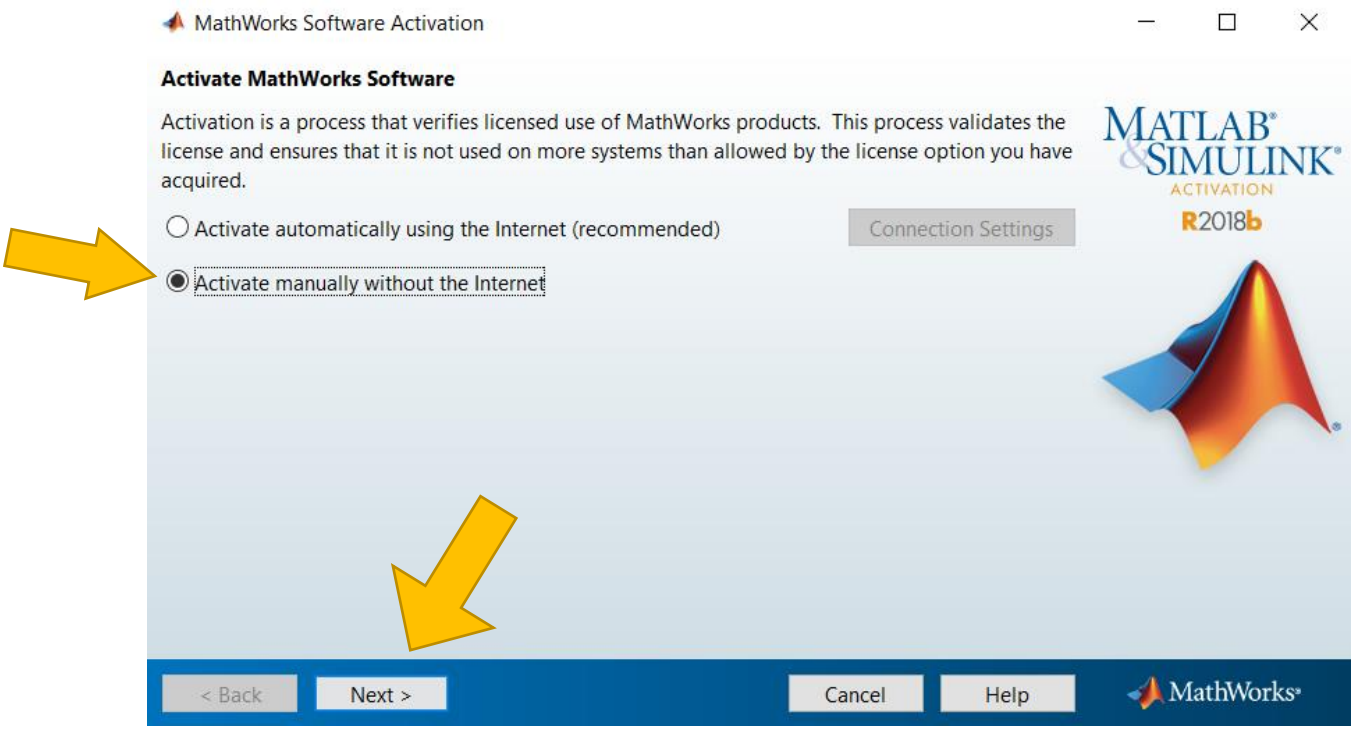
در این قسمت هم finish را می زنیم.

فایل اجرایی



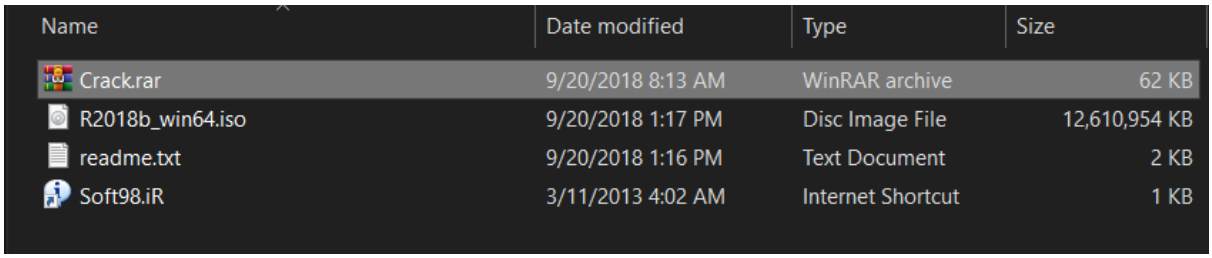
در اینجا فایل اجرایی را (MATLAB R2018b) پیدا و اجرا می کنیم.

من از منوی استارت این کار را کردم. اگر تیک Desktop را زده بودید در دسکتاپ نیز همچین آیکونی در دسترس بود.



در اینجا هم گزینه Active manually without the Internet را میزنیم و سپس next را.

کرک



Name	Date modified	Type	Size
Crack.rar	9/20/2018 8:13 AM	WinRAR archive	62 KB
R2018b_win64.iso	9/20/2018 1:17 PM	Disc Image File	12,610,954 KB
readme.txt	9/20/2018 1:16 PM	Text Document	2 KB
Soft98.iR	3/11/2013 4:02 AM	Internet Shortcut	1 KB

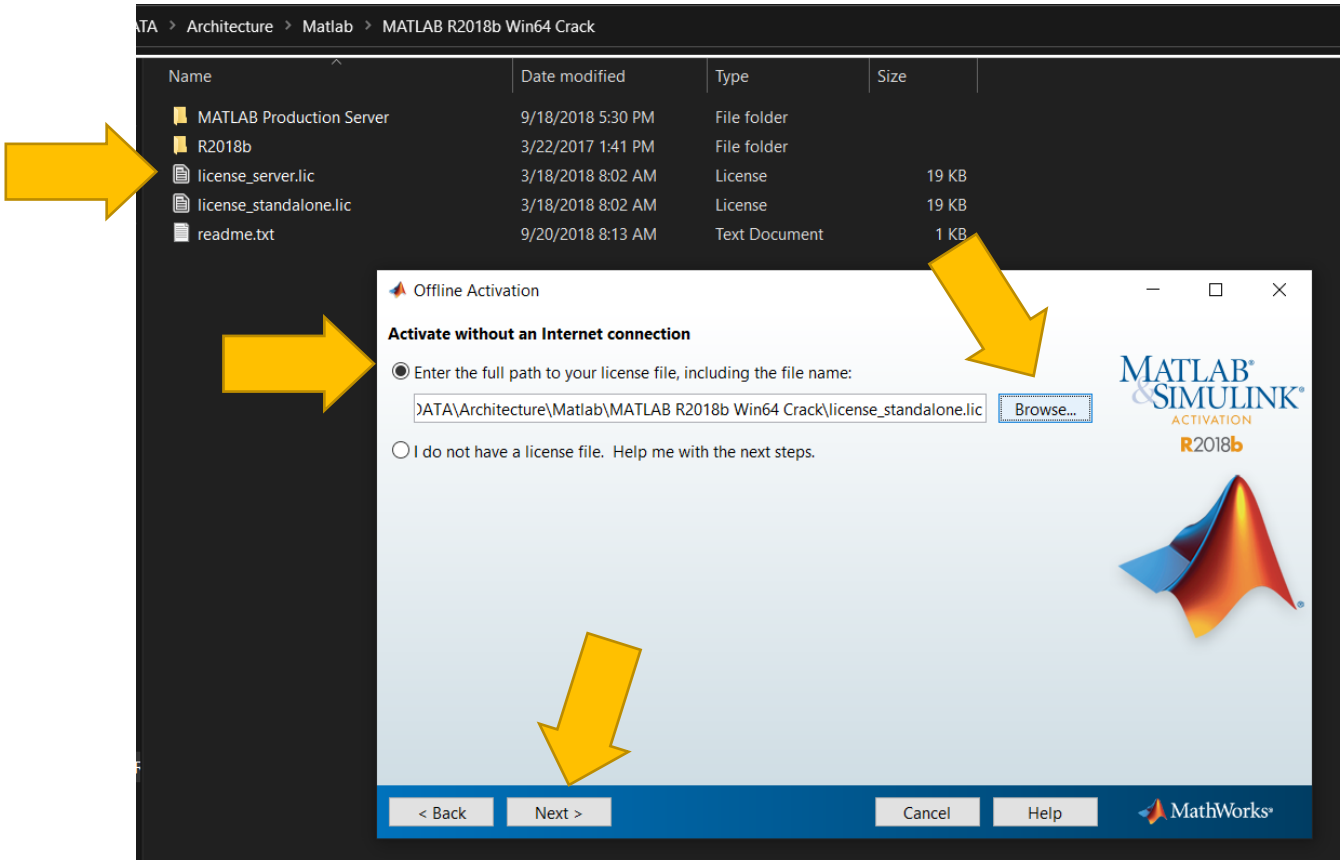
دوباره به فایل هایمان برمیگردیم. فایل Crack.rar را پیدا و آن را استخراج می کنیم تا کرک را آغاز کنیم.

ممکن است آنتی ویروس به محتویات این فایل مشکوک شود پس آن را موقتا خاموش میکنیم.

نحوه خاموش کردن موقت آنتی ویروس ویندوز در لینک زیر آمده است:

https://www.zoomit.ir/computer-learning/337214-disable-windows-defender-antivirus/

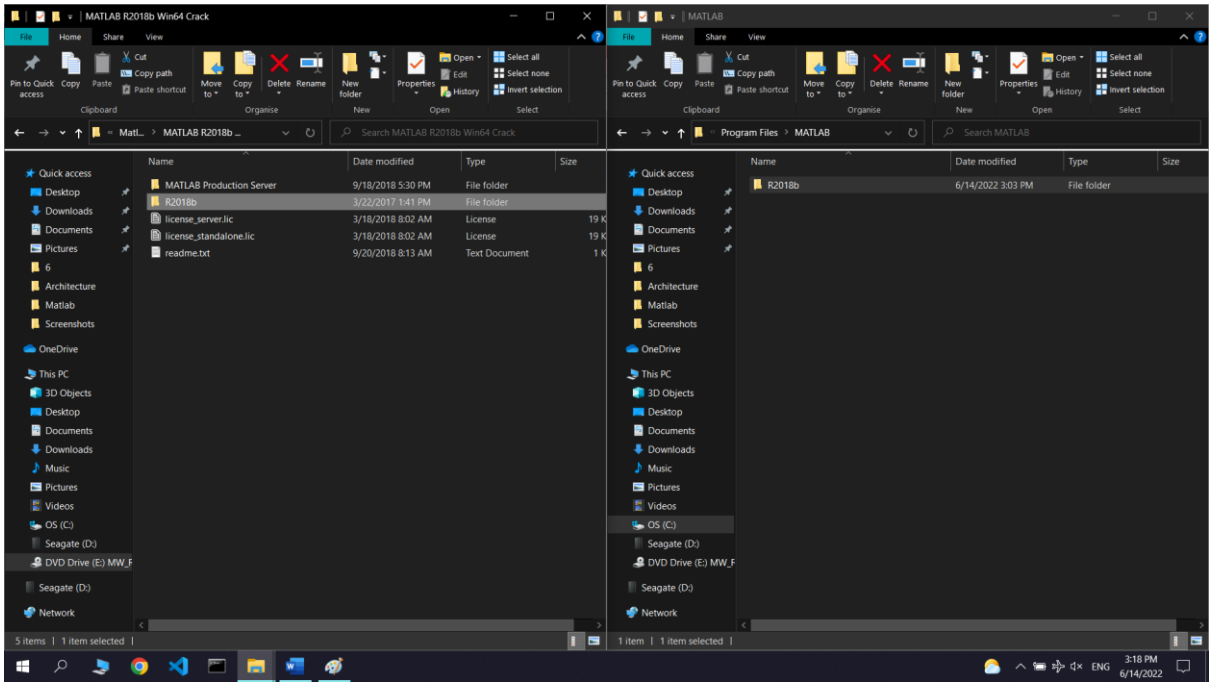
در لینک بالا ویندوز ۱۰ مورد بررسی قرار گرفته. اگر نسخه دیگری از ویندوز بر روی کامپیوتر شما نصب است با جستجو در گوگل میتوانید راه های غیرفعال سازی را پیدا کنید.



در این مرحله گزینه اول یعنی Enter the full path to your license file, including the file name را انتخاب و سپس گزینه Browse... را میزنیم. سپس فایل license_standalone.lic را پیدا و انتخاب می کنیم.

در نهایت next را میزنیم.

در این مرحله کار اکتیو (Active) کردن برنامه تمام است. بر روی finish میزنیم.



در این مرحله باری دیگر به فایل کرک خود باز میگردیم. فایل R2018b را کپی می کنیم و با توجه به مکان نصب خود در کنار فایلی به همین نام آن را کپی میکنیم.

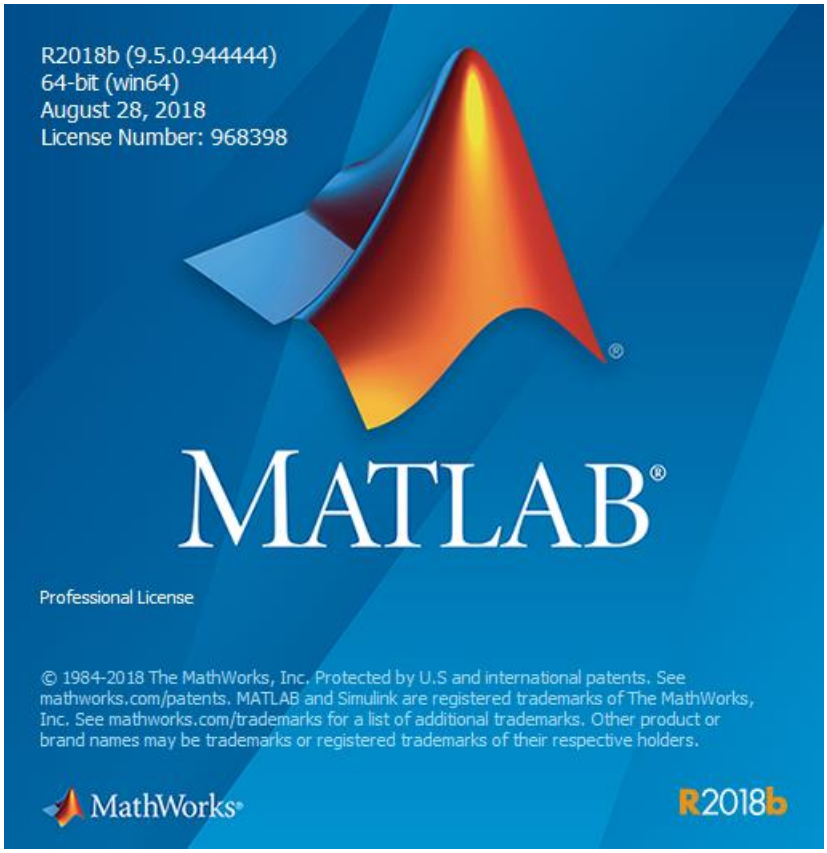
اگر توانستیم همانند بالا در دو پنجره دو مکان (فایل کرک و مکان نصب متلب) را پیدا کنیم میتوانیم با درگ اند دراپ کردن (Drag and Drop) کار را به اتمام برسانیم.

پوشه دیفالت نصب متلب در مکان زیر است:

C:\Program Files\MATLAB

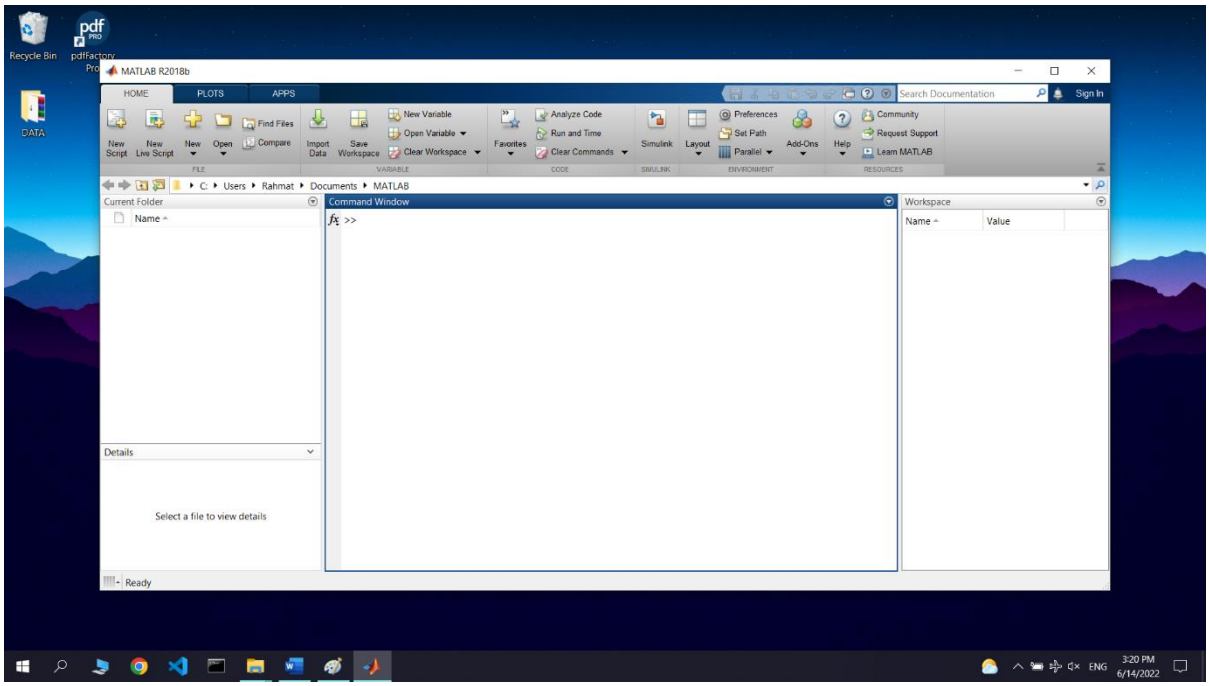
ممکن است ویندوز از ما بخواهد اینکار را تایید کنیم که این کار را می کنیم.

با اینکار فایل کرک ما که netapi.dll است در مکان درست خود قرار می گیرد و کار ما تمام است.

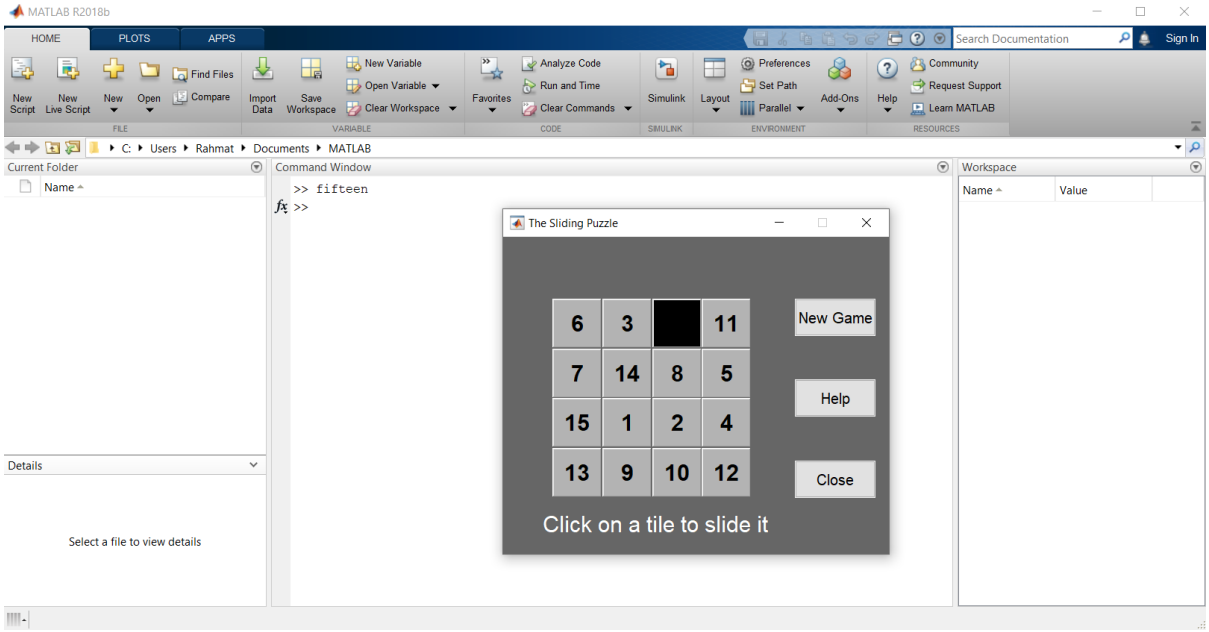


مراحل نصب تمام شد. حال میتوانیم نرم افزار را با خیال راحت اجرا کنیم. (MATLAB R2018b)

پنجره اولیه متلب همانند زیر است:



اطمینان از نصب درست



برای اطمینان از نصب درست و درست کار کردن متلب میتوانیم کد بازی fifteen را بنویسیم.

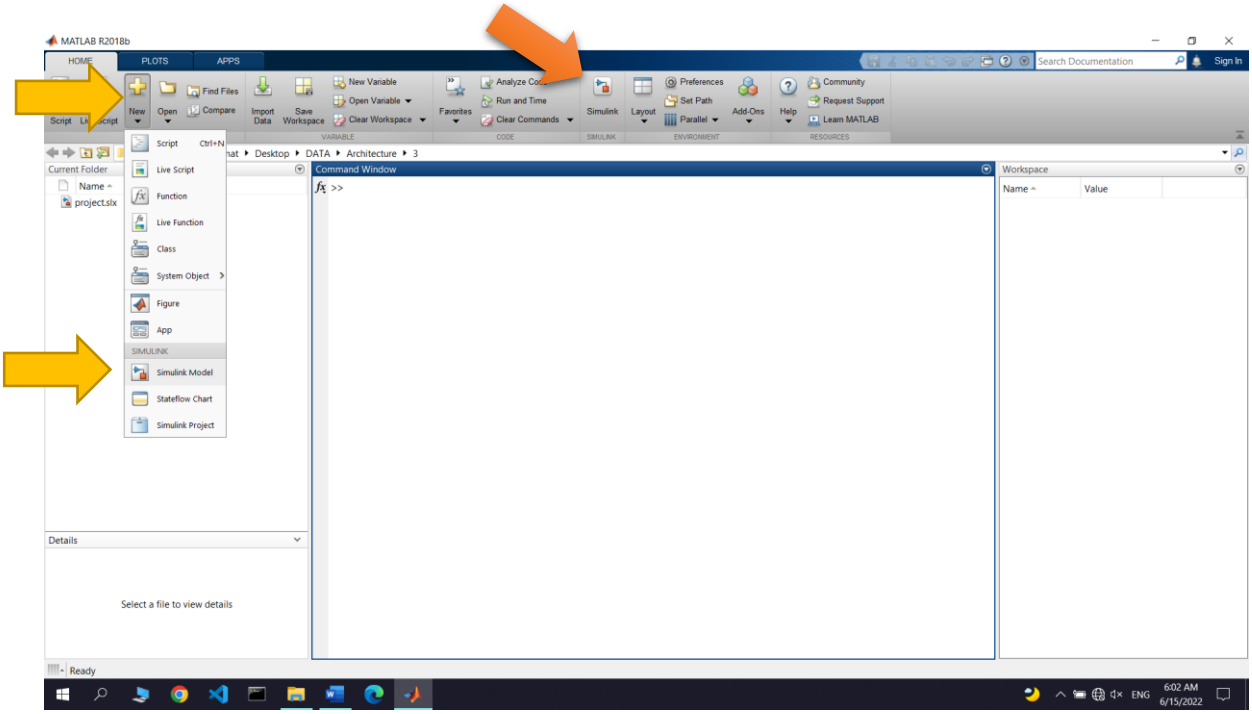
اگر با زدن اینتر بازی اجرا شد یعنی نصب به خوبی انجام شده است.

فصل دوم

آموزش کار با سیمولینک

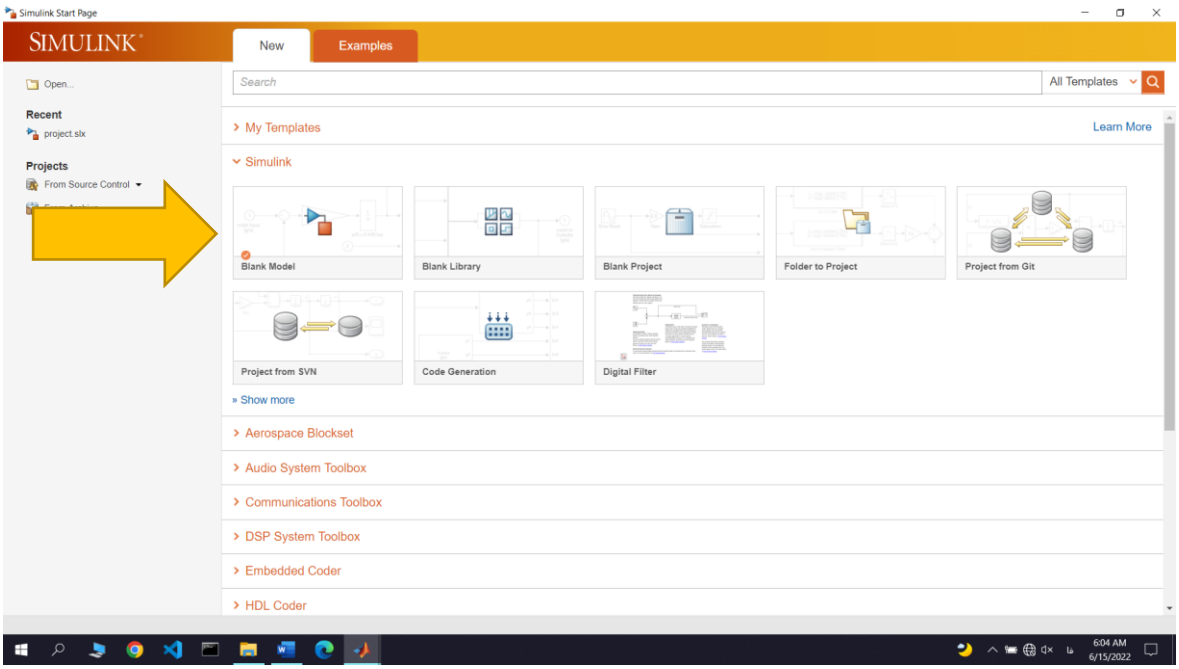
رفتن به سیمولینک

در این فصل به آموزشی مقدماتی از Simulink می پردازیم.

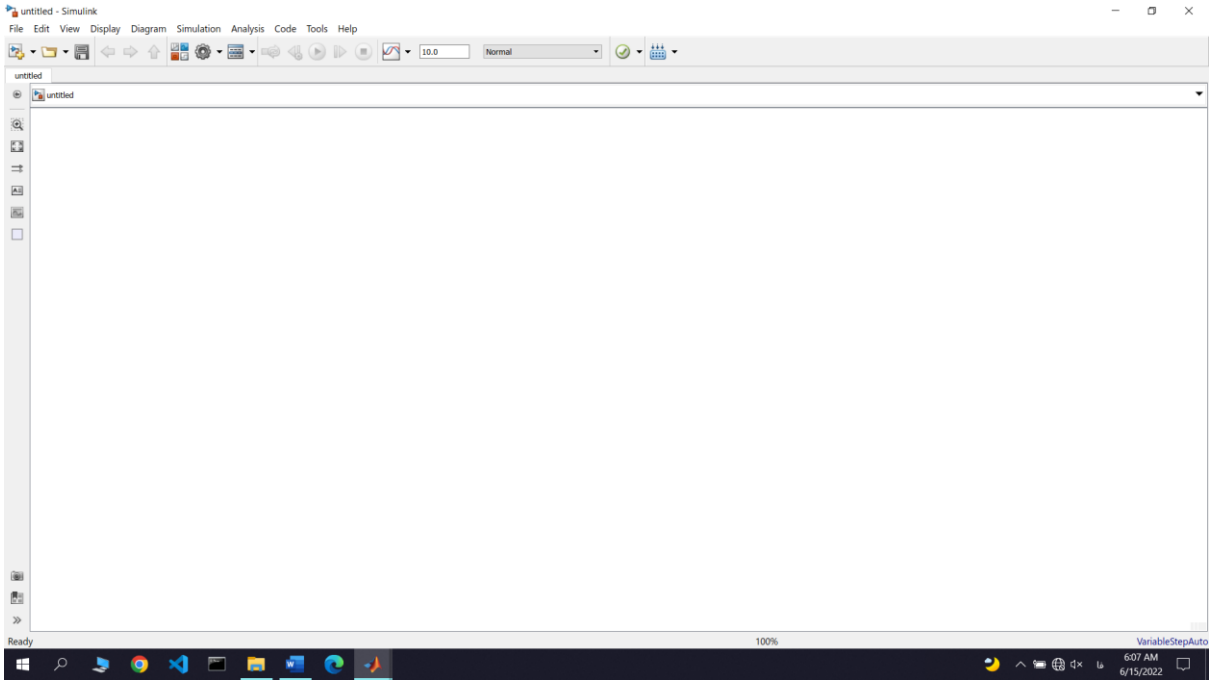


برای رفتن به بخش سیمولینک ابتدا new را می زنیم و سپس Simulink Model را انتخاب می کنیم.

همچنین میتوانیم مستقیم به Simulink برویم. (فلش نارنجی)

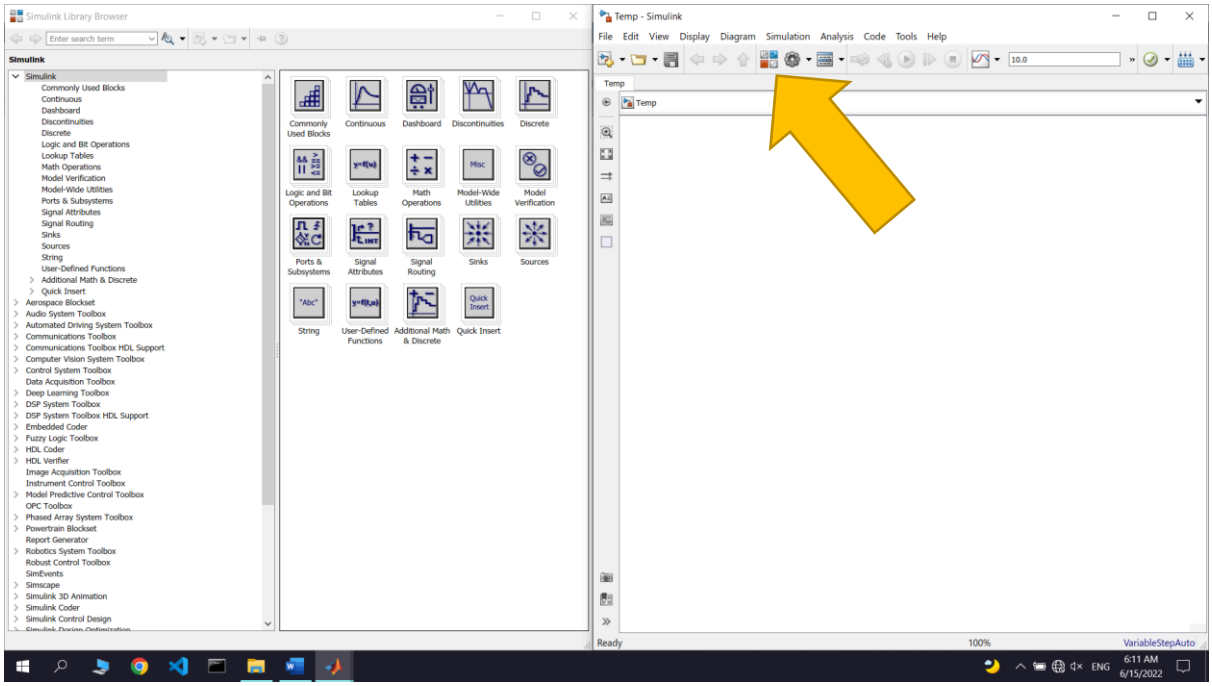


در این پنجره هم Blank Model را انتخاب می کنیم.



صفحه همانند بالا را خواهیم داشت.

برای ذخیره آن میتوانیم کلید های ترکیبی ctrl+s را بزنیم یا از تب File گزینه Save را بزنیم.



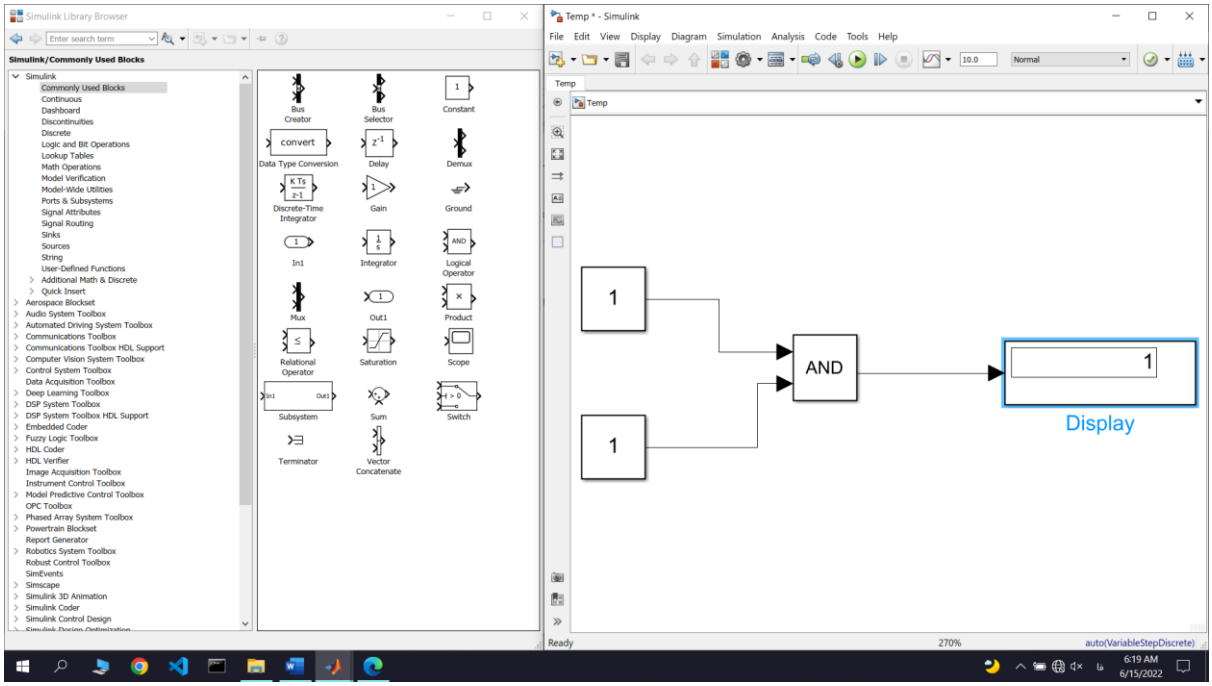
برای دسترسی به قطعات گزینه مشخص شده یعنی Library Browser را میزنیم تا پنجره ای همانند سمت چپ عکس بالا باز شود.

کتابخانه ها

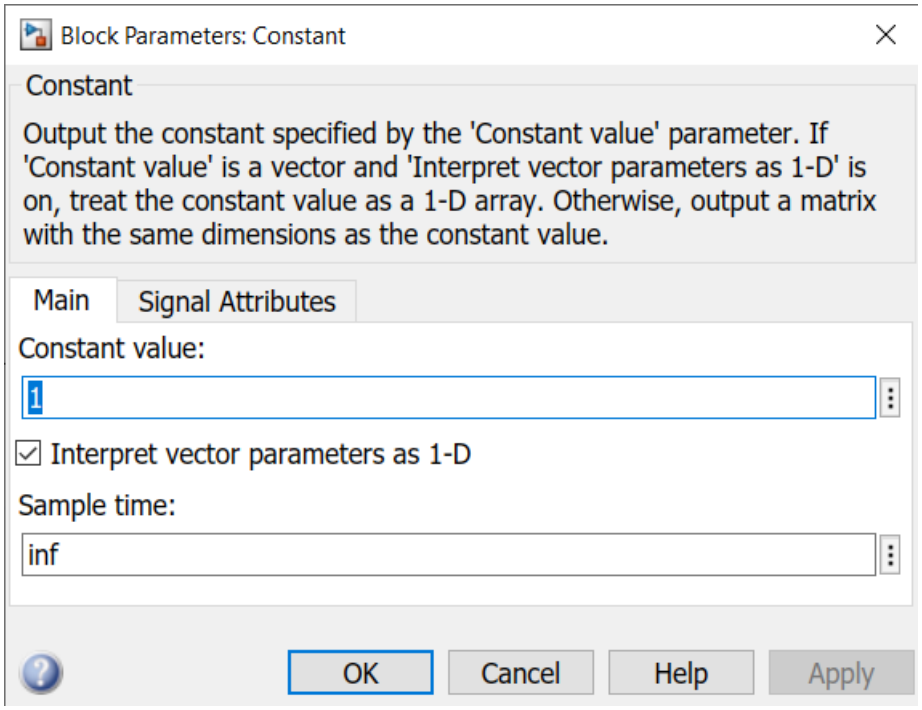
در این جا ما کتابخانه های مورد استفاده خود را میبینیم.

تمام بلوکهای سیمولینک در این پنجره و در پوشه هایی ذخیره شده اند. بلوک های موجود در پوشه اول یعنی پوشه Simulink ، بلوکهای عمومی هستند و پوشه های دیگر حاوی بلوکهای تخصصی هستند .پوشه Simulink خودش حاوی چند زیرپوشه است. مثلا زیرپوشه Sources حاوی بلوک های منابع تولید سیگنال و زیرپوشه Sinks حاوی بلوکهای نمایشگر است. پوشه اول یعنی Blocks Used Commonly حاوی پراستفاده ترین بلوک ها است که از پوشه های دیگر گلچین شده اند.

هر قطعه ای که بخواهیم کافی است از پنجره Simulink Library Browser بکشیم و وارد پنجره اصلی سیمولینک خود کنیم. (درگ اند دراپ)

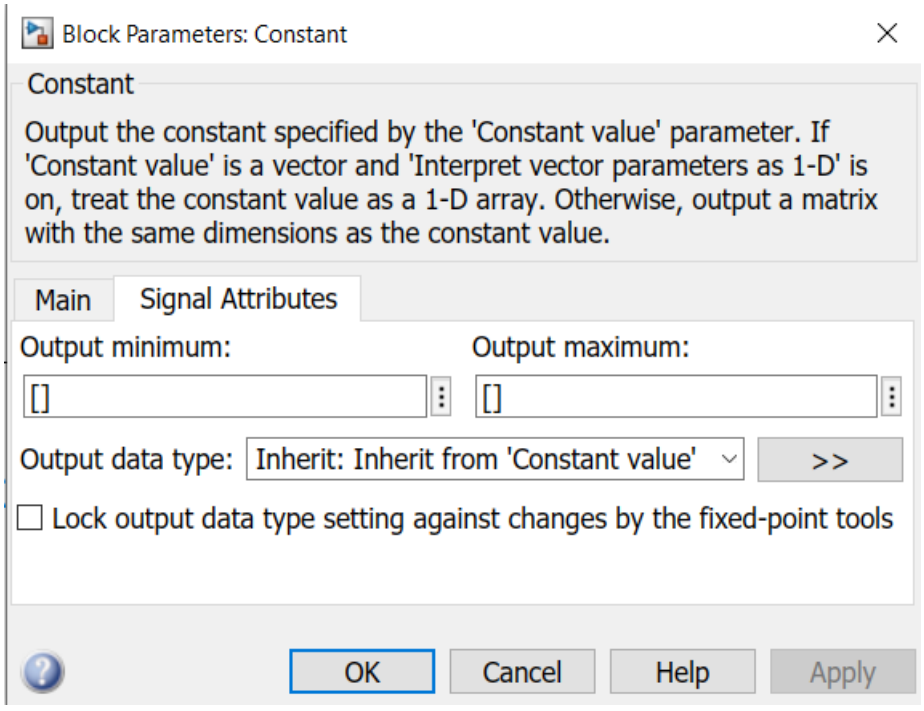


بلوک Constant برای دادن ورودی می باشد.

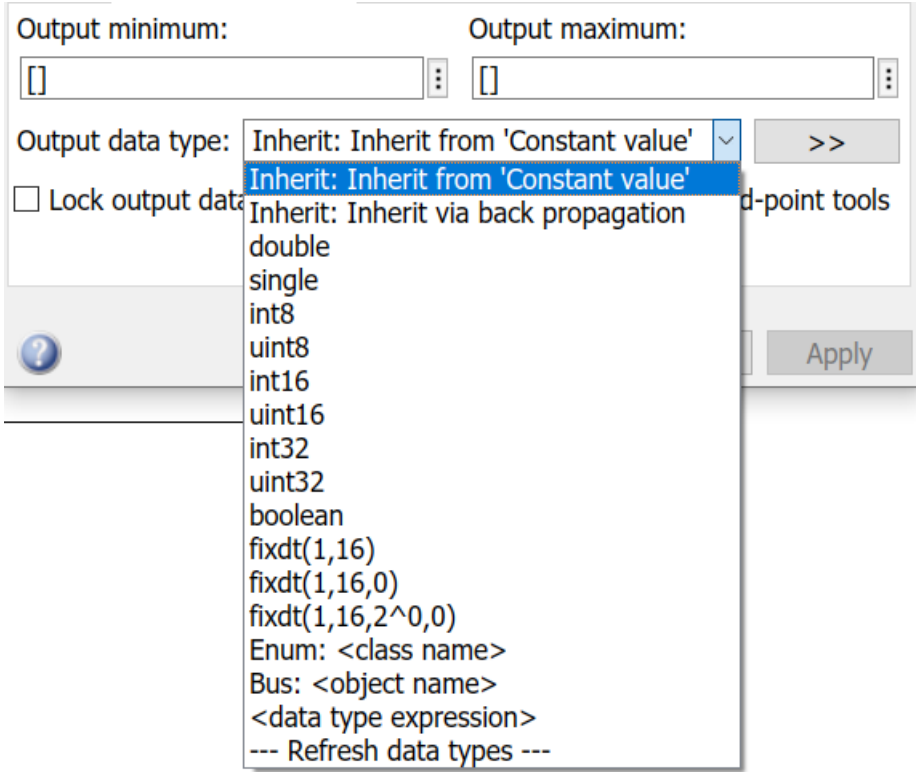


اگر بر روی آن دبل کلیک کنیم میتوانیم در Constant value مقدار ورودی را تنظیم کنیم.

اگر بر روی تب single Attributes بزنیم شکل زیر را داریم:



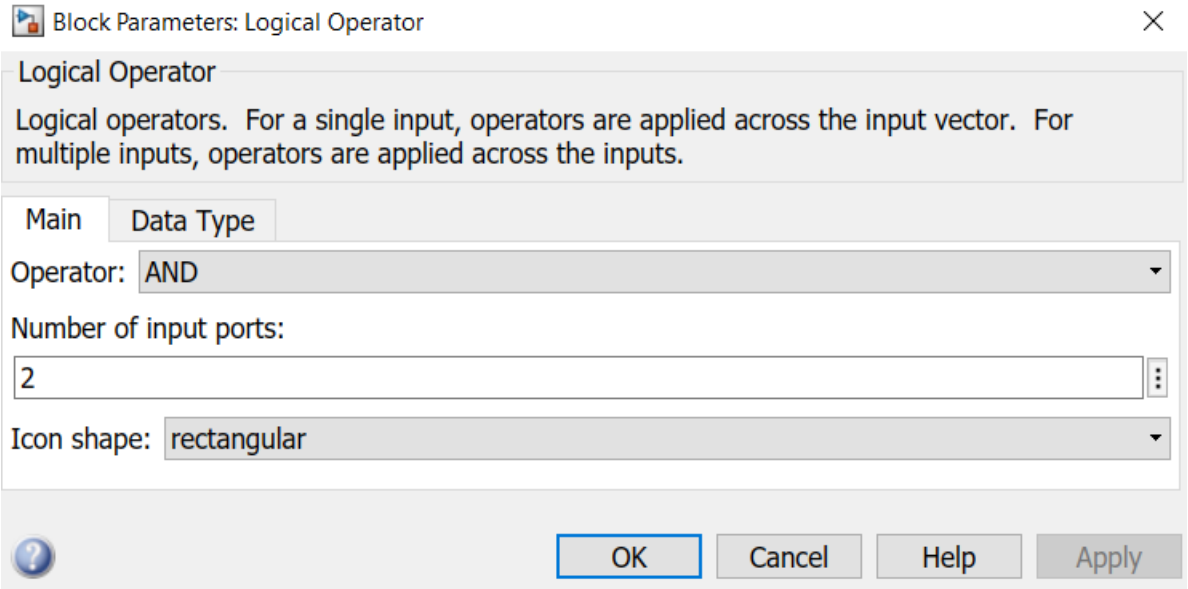
در اینجا هم میتوانیم نوع خروجی را مشخص کنیم. (در Output data type)



نوع بولین که شامل منطق صفر و یک است را در اینجا میتوانیم انتخاب کنیم اما فعلا نیازی به آن نداریم.

بلوک Logical Operator هم شامل انواع گیت های منطقی است.

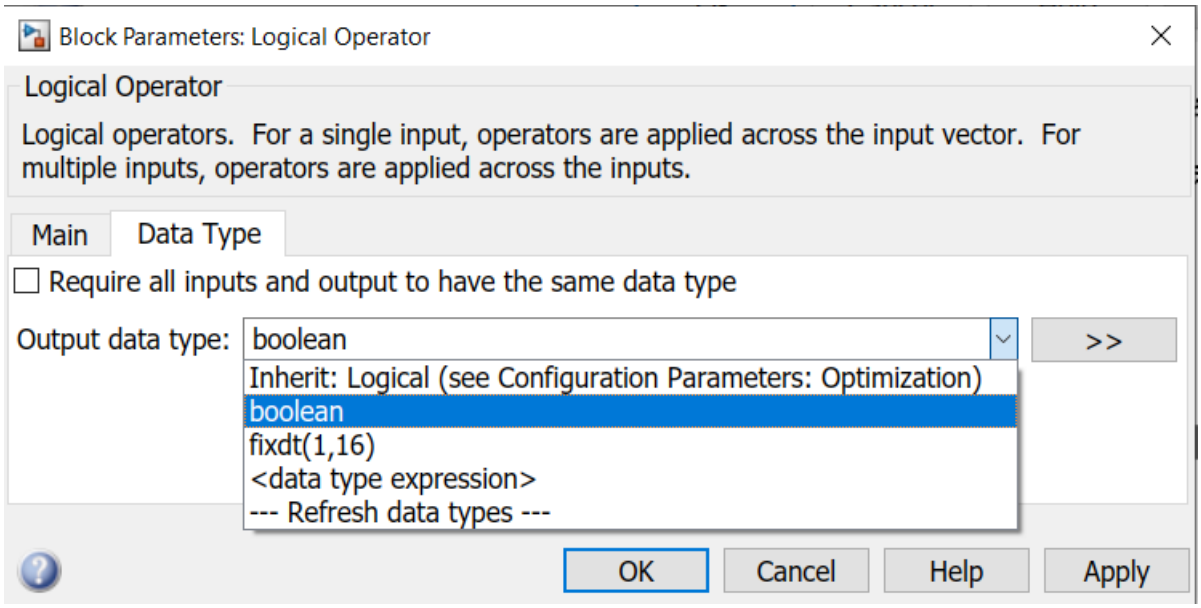
(شامل AND, OR, NOT, XOR, NOR, NAND, XNOR)



پنجره بالا با دبل کلیک روی گیت logical operator بدست آمد. میتوان تعداد ورودی را تنظیم کرد.

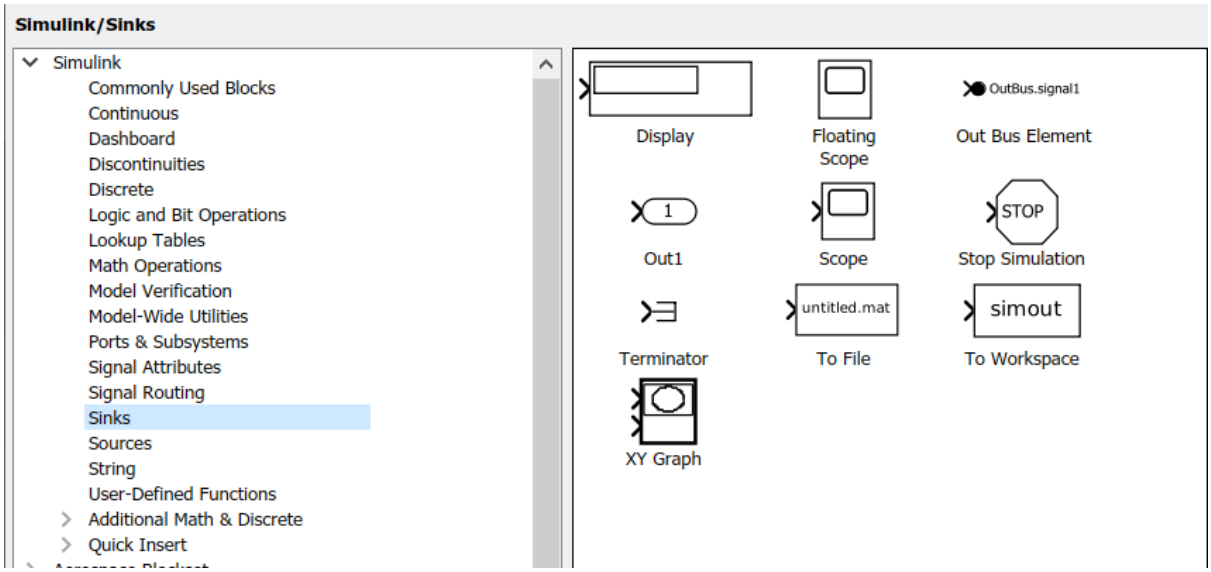
همچنین نوع گیت را هم مشخص کرد.

گزینه ای هم تحت عنوان Icon shape وجود دارد که شکل آیکون گیت را مشخص میکند. Rectangle (مستطیل) باشد یا distinctive (شکل نرمال گیت)

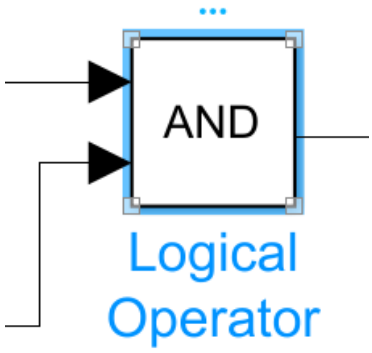


با رفتن در تب Data type هم میتوانیم نوع خروجی گیت را تنظیم کنیم.

بلوک Display از پوشه sinks هم برای دادن خروجی است.

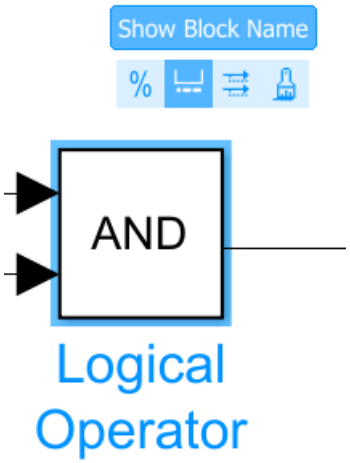


برای متصل کردن گیت ها به هم کافی است موس را به جای وروی و خروجی ها ببریم تا شکل کرسر موس عوض شود.

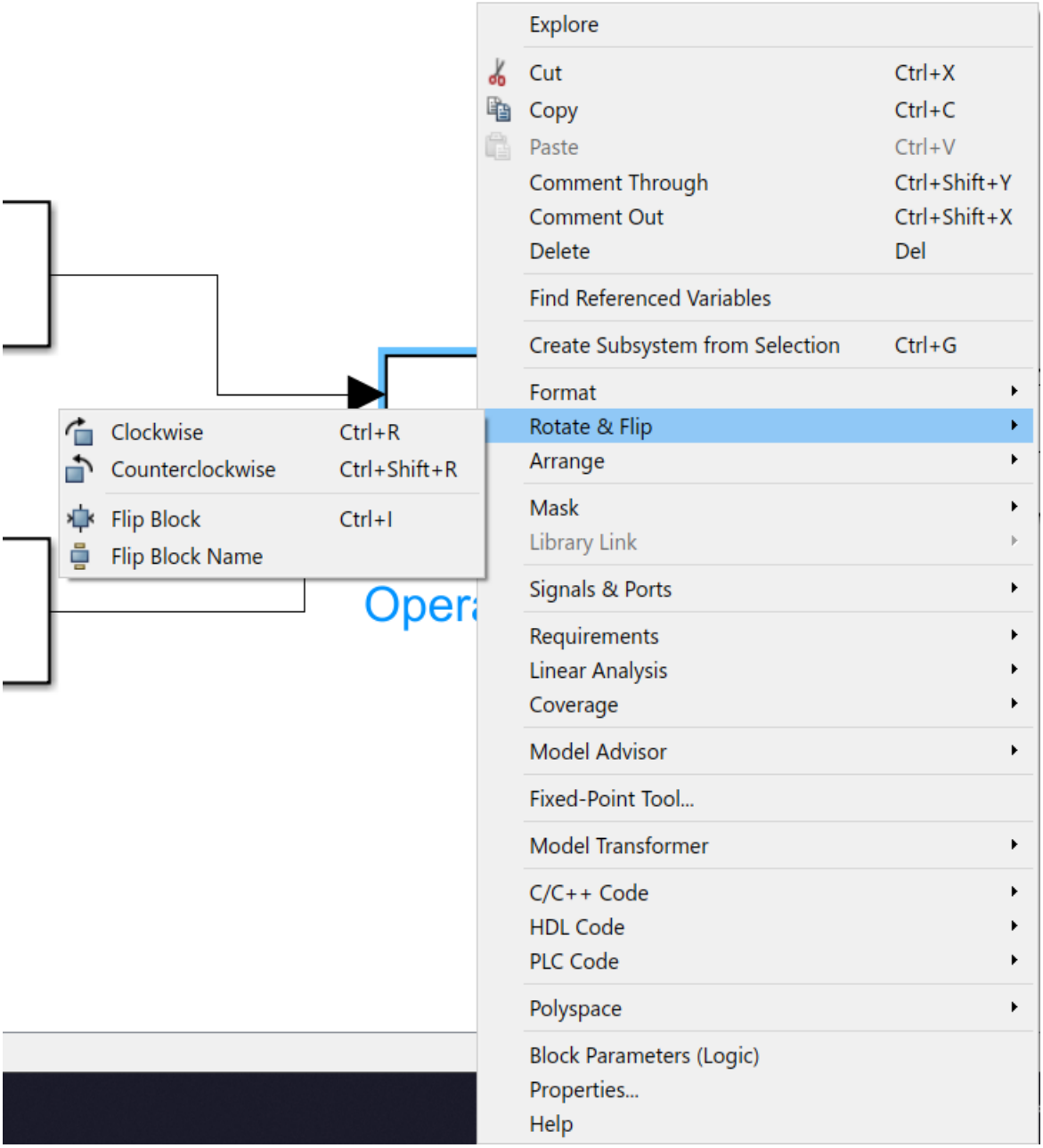


با کلیک روی یک قطعه میتوانیم اسم آن را عوض کنیم. (کافی است روی اسم بزنیم تا قابل ویرایش شود.)

همچنین برای برخی ویرایش ها میتوانیم روی سه نقطه ی بالای آن بزنیم.

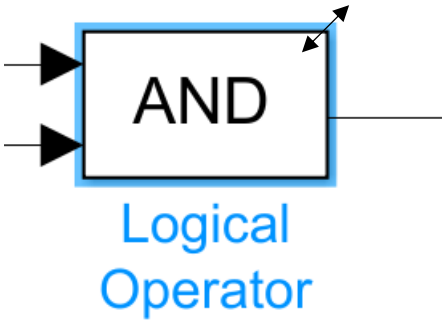


همانطور که میبینیم میتوان نام پوشه را نمایش داد یا نداد.

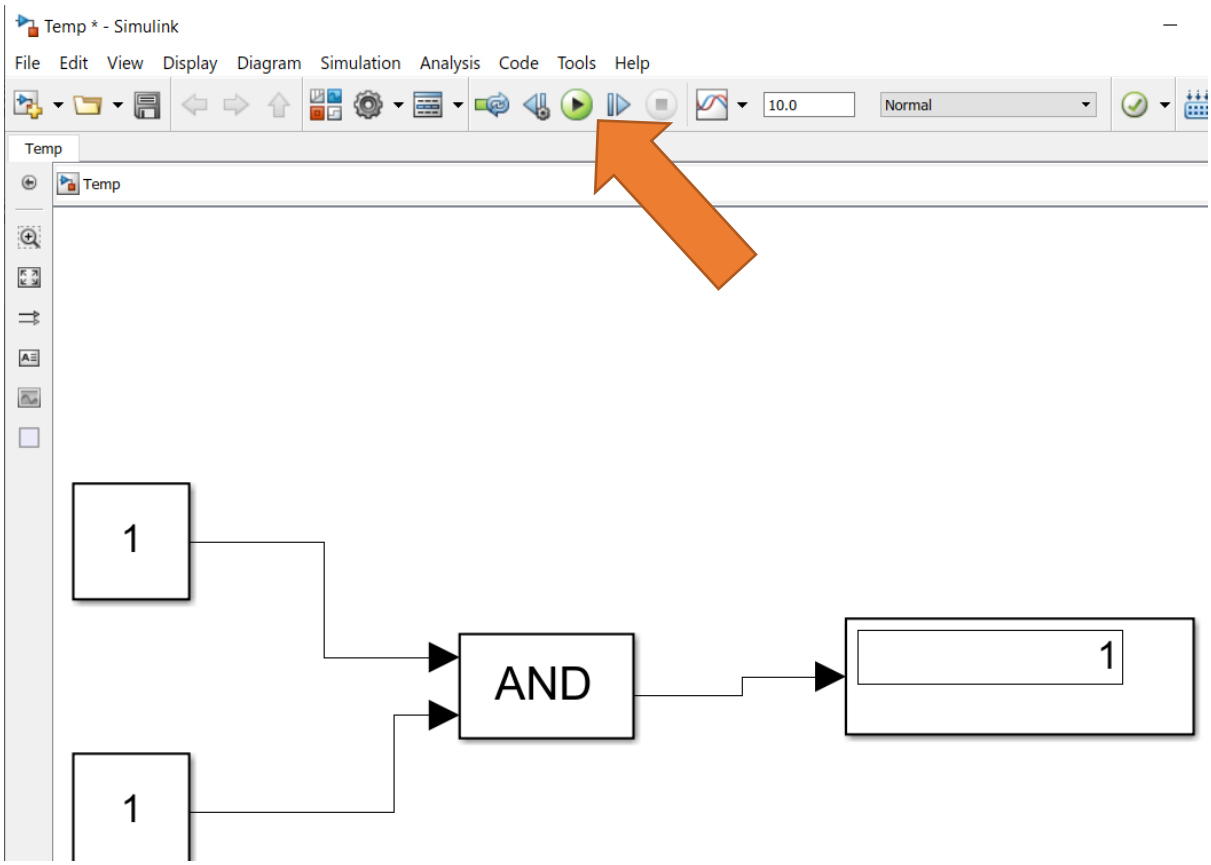


با راست کلیک روی یه قطعه میتوانیم آن را منتقل (cut) یا کپی (copy) و یا حذف (Delete) کنیم.

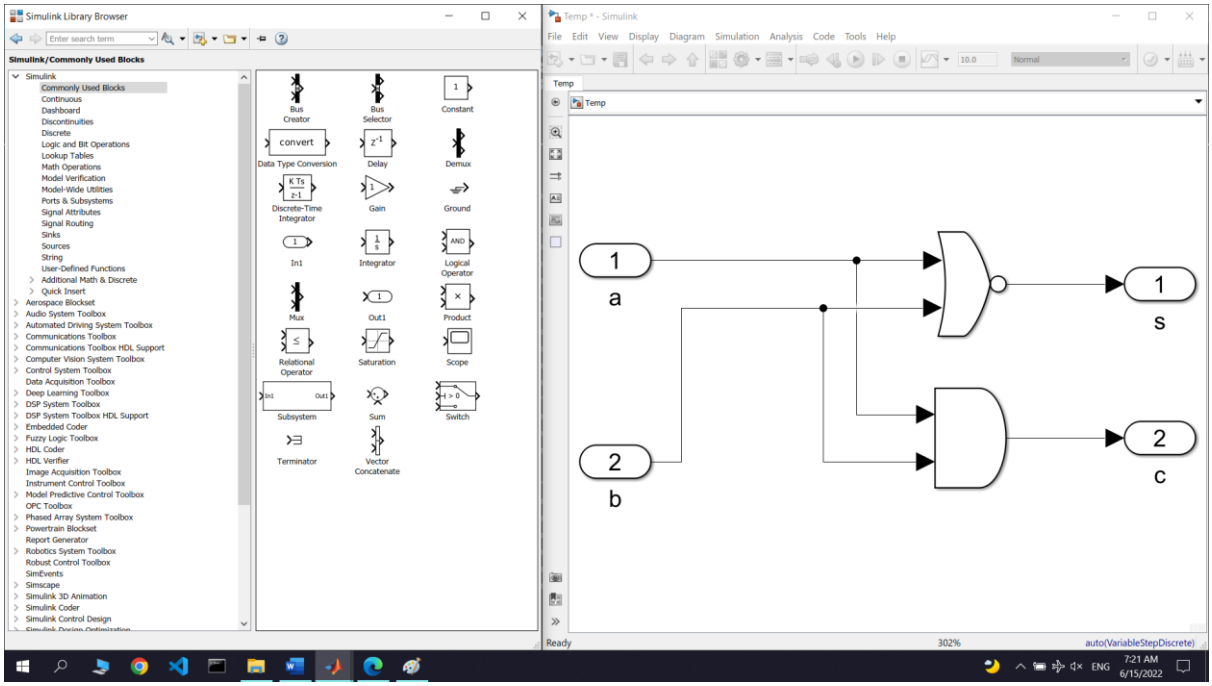
همچنین میتوان Rotate & Flip را انتخاب و آن را چرخاند.



برای تغییر اندازه بلوک هم میتوان موس را بر روی گوشه ای از بلوک برد تا شکل کرسر آن عوض شود و قابل تغییر اندازه باشد.

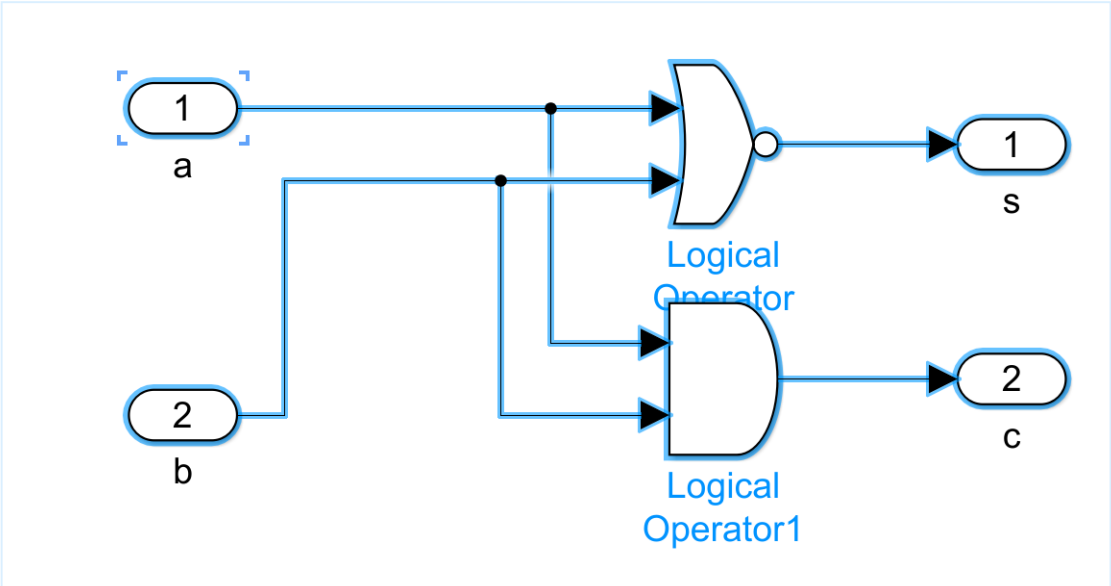


برای خروجی گرفتن از مدار هم کافی است علامت مشخص شده (Run) را بزنیم.

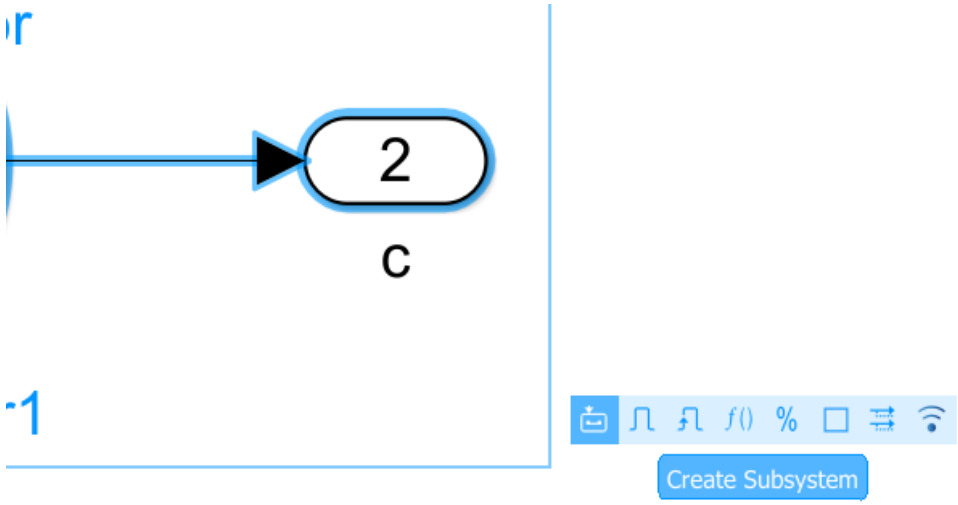


همچنین با گیت های in1 و out1 را داریم که همان ورودی و خروجی ما هستند.

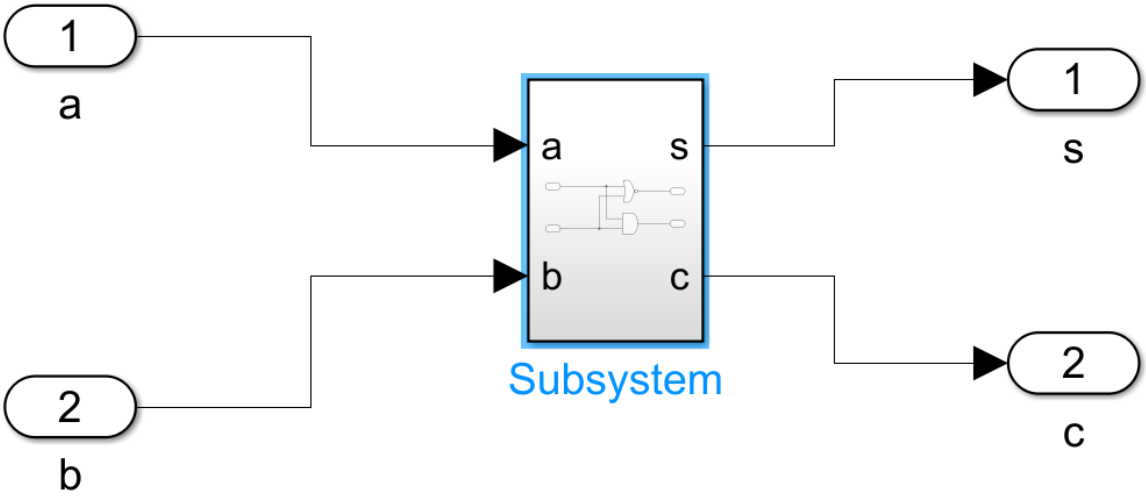
شماره درون آن ها مربوط به port number آن هاست. اگر مدار را تبدیل به یک بلوک کنیم بر اساس port number پورت ها در دیواره بلوک قرار می گیرند.



با کشیدن موس میتوان مدار یا قسمتی از آن را انتخاب کرد. پس از انتخاب میتوان آن را کپی، حذف و یا منتقل کنیم. همچنین باز هم در گوشه مستطیل ایجاد شده یک سه نقطه ظاهر می شود.



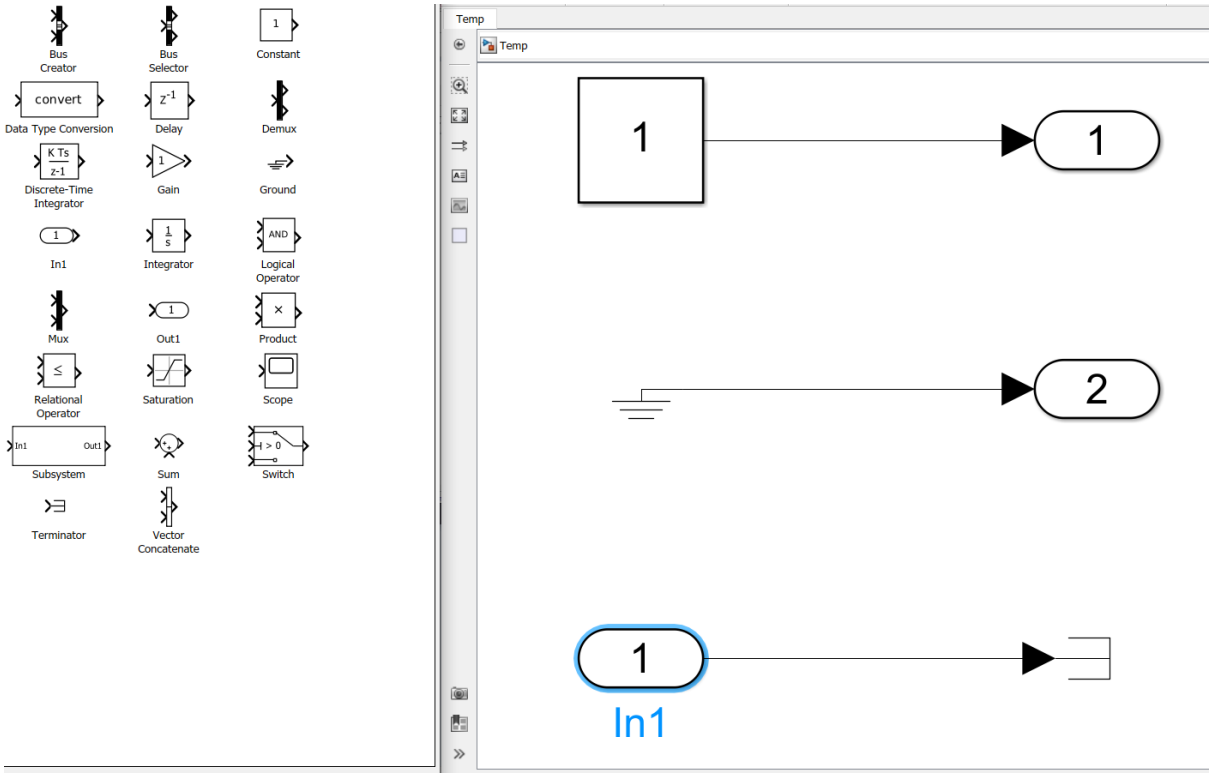
همانطور که میبینید یکی از گزینه ها create Subsystem است که برای ساخت یک بلوک جدید با ورودی ها و خروجی های گفته شده و منطقی که پیاده شده به کار می رود.



پس از زدن این گزینه بلوک ساخته می شود. در اینجا ما یک نیم جمع زننده (Half Adder) ساختیم. برای استفاده از آن هم میتوانیم از constant و display استفاده کنیم یا حتی از آن در ساخت یک بلوک دیگر استفاده کنیم.

برای این بلوک هم میتوانیم نام انتخاب کنیم یا نام آن را پنهان سازیم.

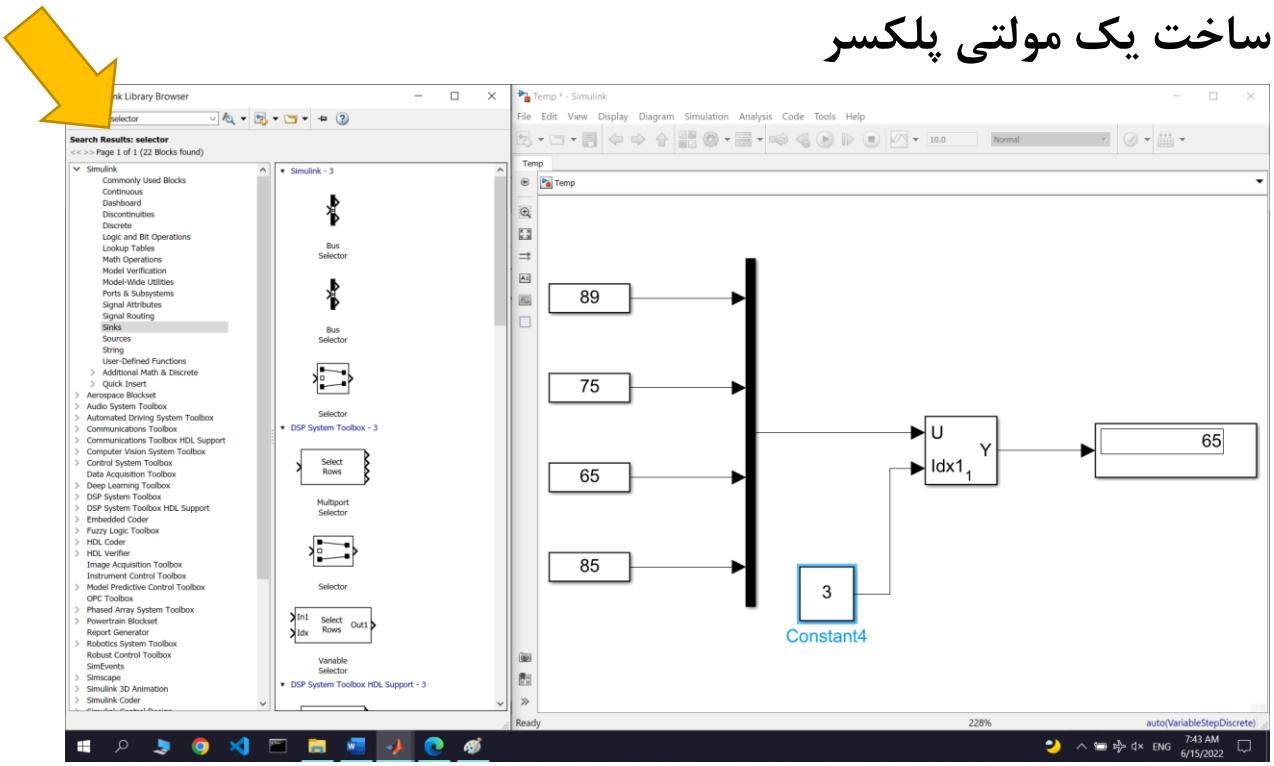
هم چنین اگر نام ورودی و یا خروجی های درون بلوک را پنهان می کردیم (in1, out1) بر روی بلوک هم نمایش داده نمی شد. همچنین میتوانیم ببینیم که ترتیب ورودی ها و خروجی ها بر اساس port number آن است.



از دیگر گیت های استفاده شده در این پروژه زمین یا Ground است که منطق 0 منطقی است.

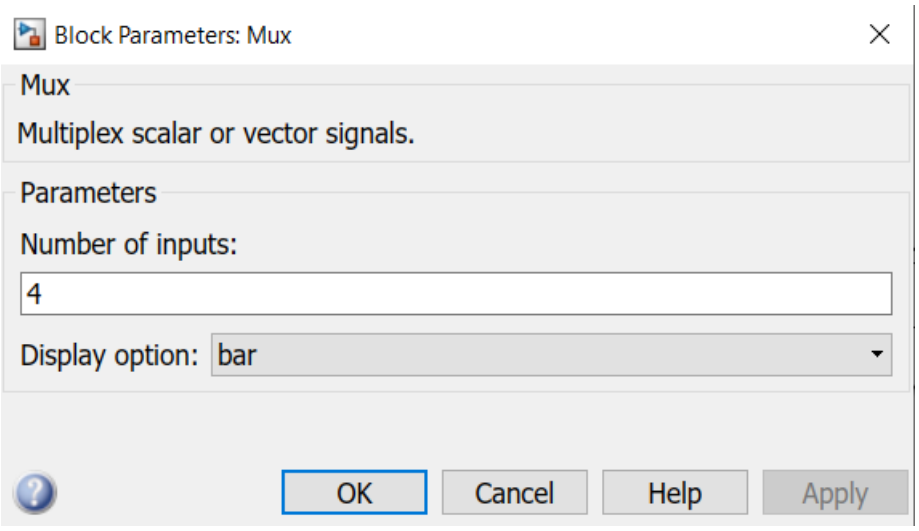
همچنین از ترمینال هم استفاده شده که برای خروجی های بلا استفاده متوانیم استفاده کنیم.

ساخت یک مولتی پلکسر



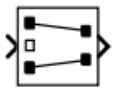
در اینجا ما منطق یک مولتی پلکسر را با گیت های آماده ی متلب پیاده کردیم.

برای اینکار ابتدا به یک گیت mux داریم که در همان کتابخانه Commonly used Blocks هم موجود است.
پس از انتخاب تعداد پورت های آن را ۴ می کنیم.



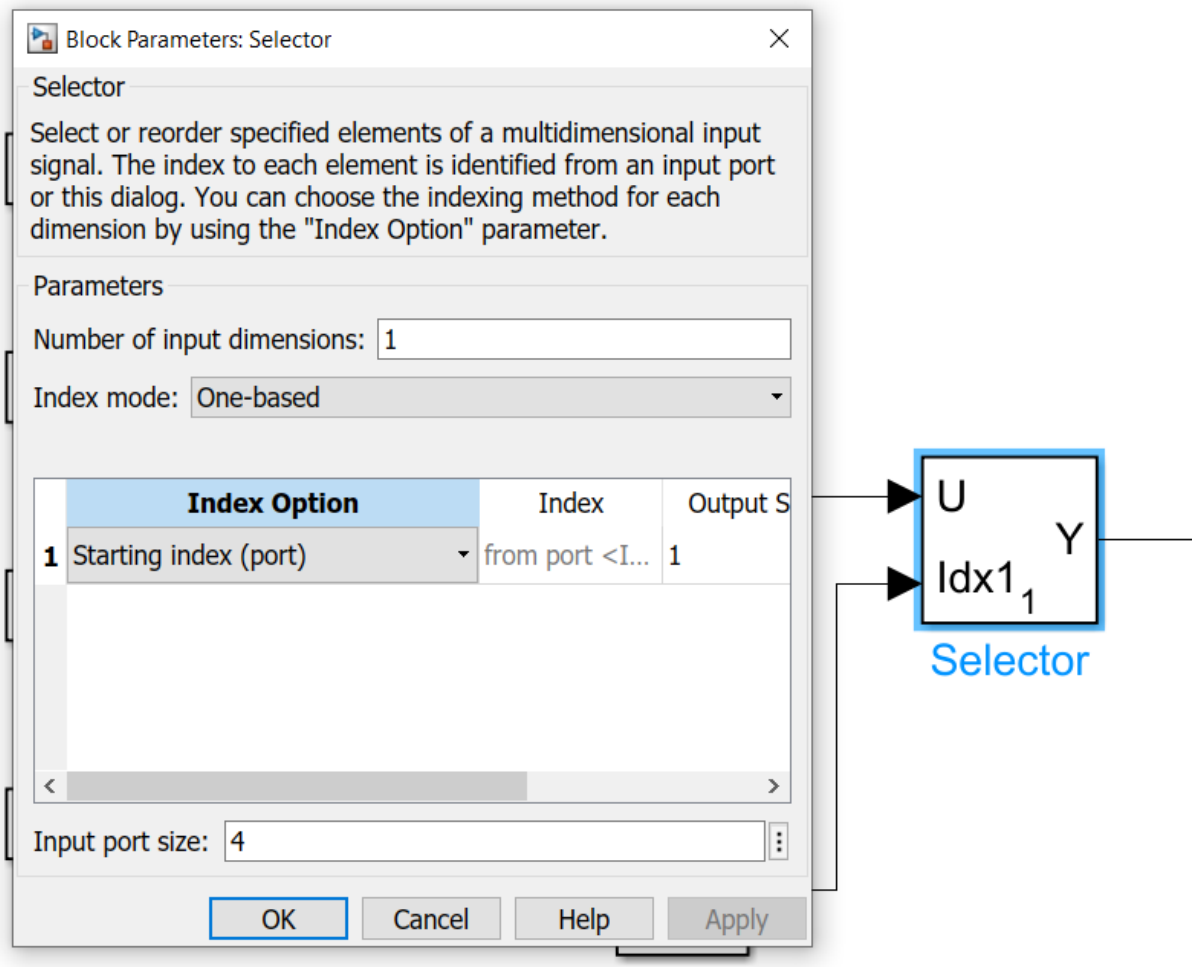
با دبل کلیک روی آن پنجره بالا باز می شود.

همچنین نیاز به یک سلکتور (selector) داریم که با سرچ در قسمت Enter search term پنجره Simulink Library Browser (قسمت مشخص شده در تصویر صفحه قبل با فلش زرد رنگ) میتوانیم به راحتی آن را پیدا کنیم.



Selector

این قطعه را همچنین در کتابخانه signal routing نیز میتوانید پیدا کنید. (جای Mux)



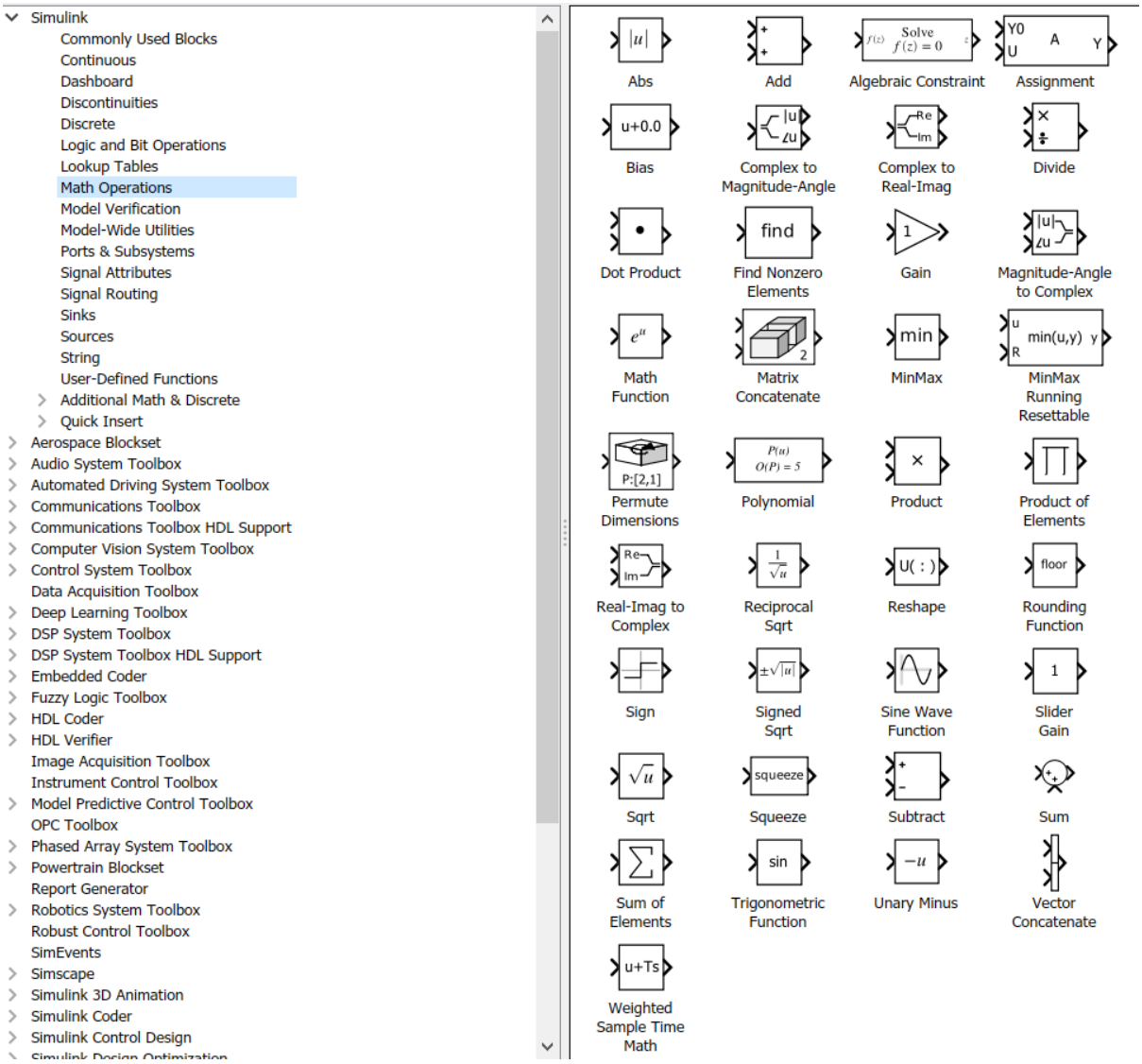
اگر بر روی سلکتور دبل کلیک کنیم پنجره بالا را مشاهده می کنیم.

در قسمت Input port size تعداد پورت هایی که به Mux متصل است را مشخص می کنیم.

قسمت Index Option را هم باید روی Starting index (Port) قرار دهیم.

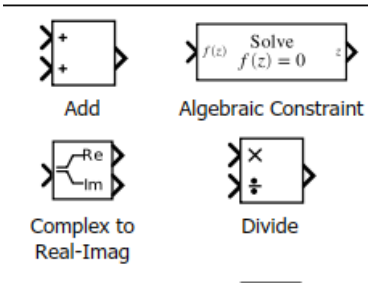
با اینکار ما یک مولتی پلکسر داریم. البته **اندیس های این مولتی پلکسر از یک شروع می شوند** و هم چنین ورودی کنترل این مولتی پلکسر باید از نوع double باشد.

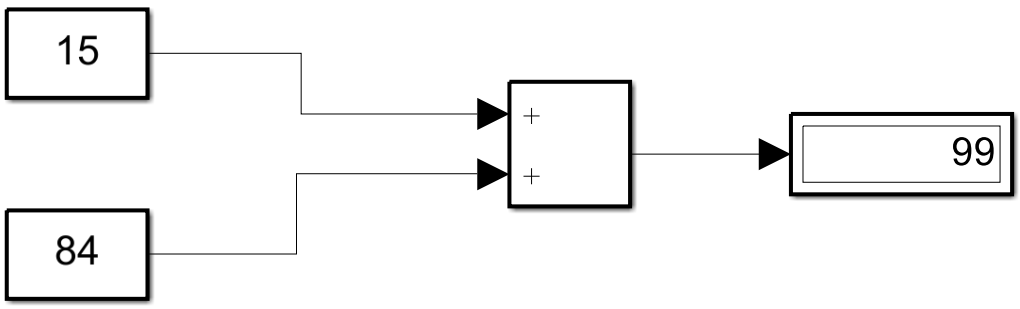
کتابخانه Math Operations

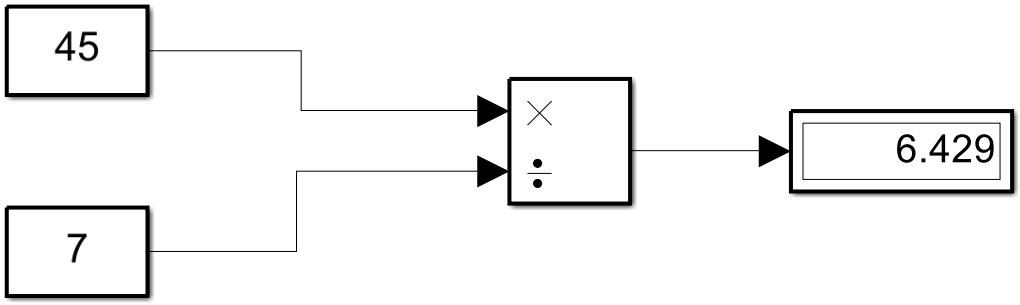


در این کتابخانه عملگر های ریاضی را داریم. از جمله آن ها که در این پروژه استفاده می شوند بلوک های زیر است:

بلوک Add و Divide



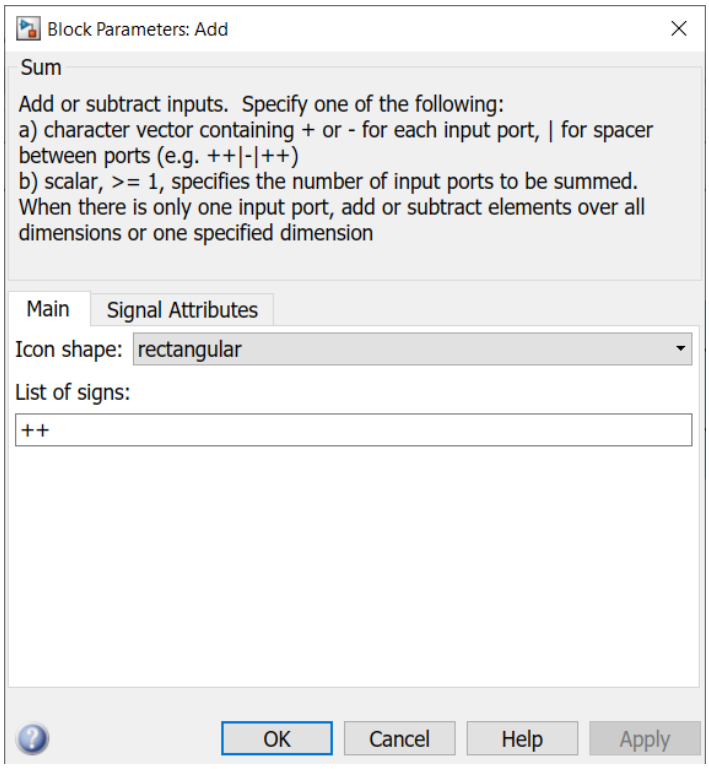




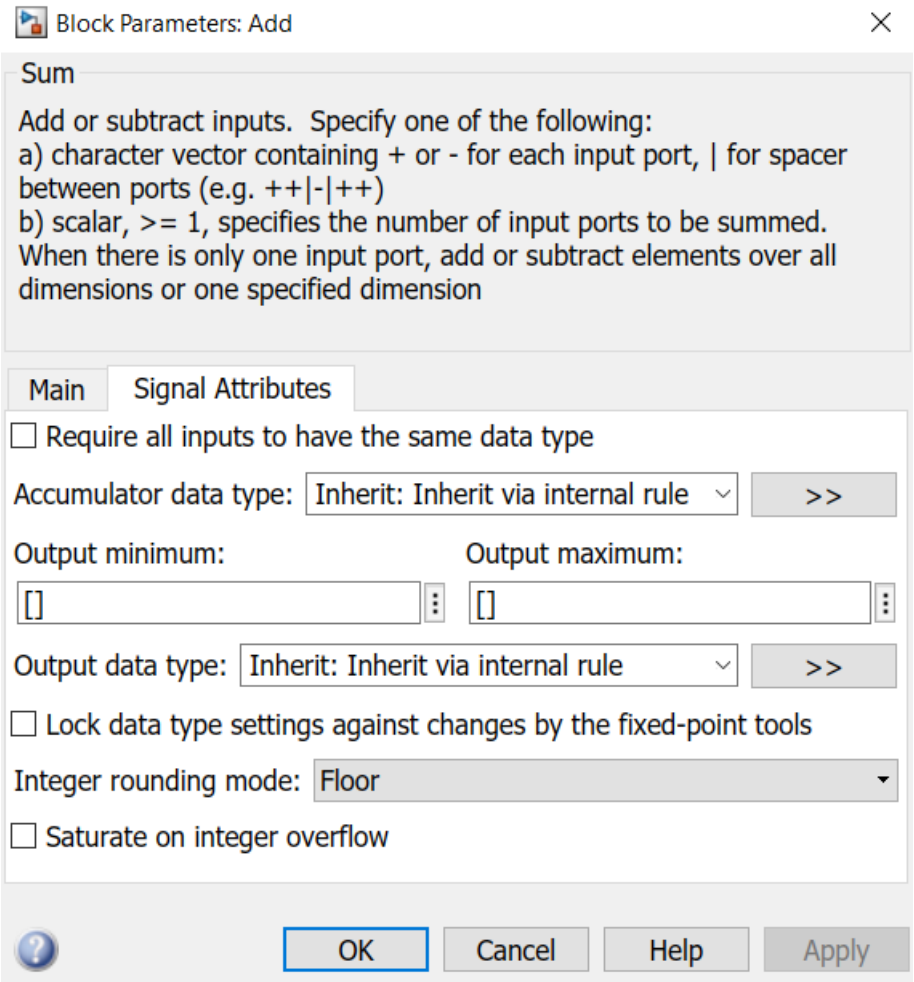
کار بلوک Add جمع زدن ورودی ها است.

با دبل کلیک روی بلوک آن پنجره زیر باز می شود که میتوانیم تعداد ورودی ها را کنترل کنیم. برای افزودن ورودی کافی است به List of signs علامت به علاوه اضافه کنیم.

برای مثال اگر ۳ ورودی نیاز داشتیم باید در List of signs محتوای **+++** باشد.

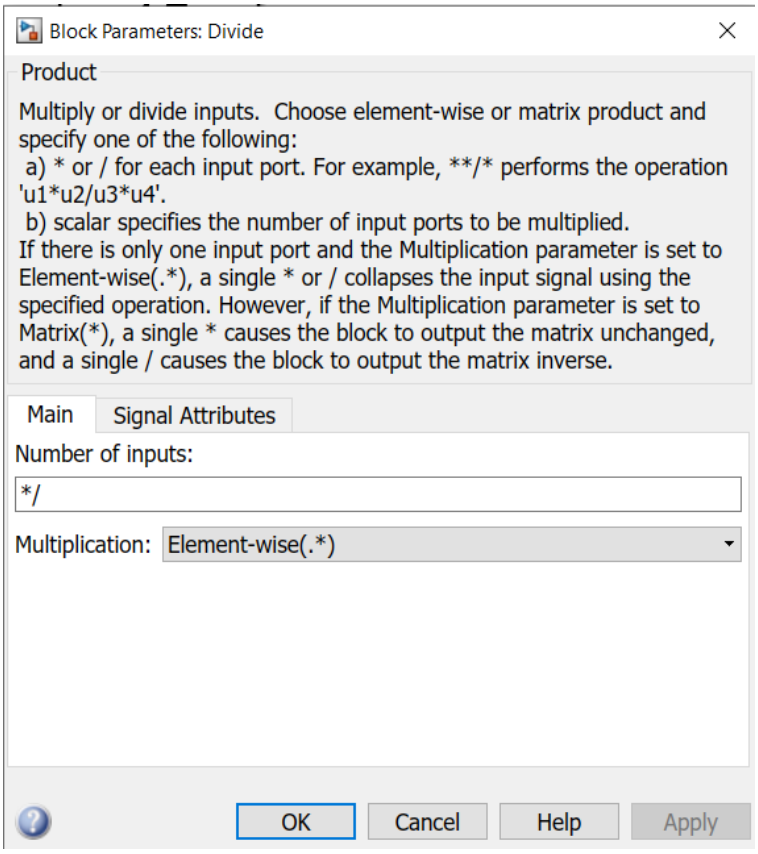


در تب Signal Attributes هم به شکل زیر داریم:



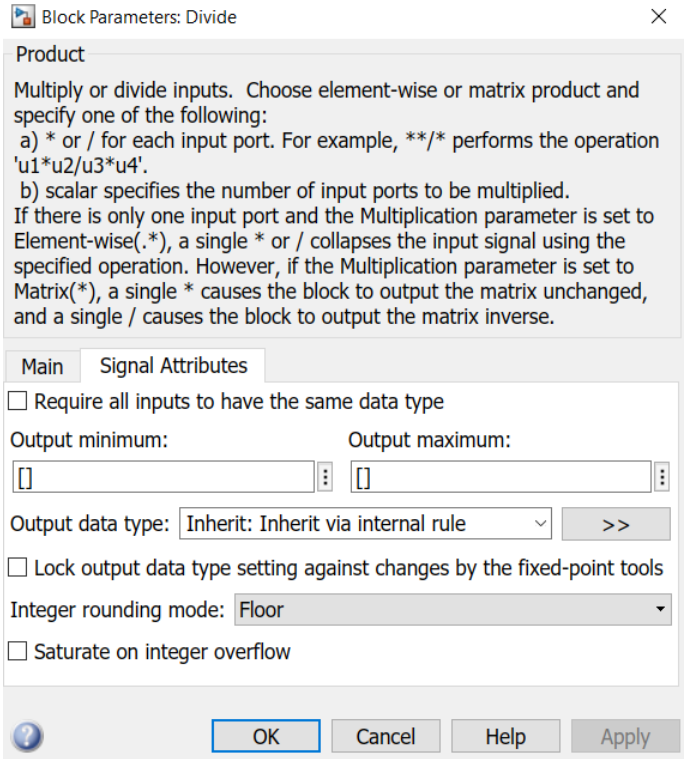
در اینجا در قسمت Accumulator data type نوع داده پروسه را مشخص می کنیم. در قسمت output data type که مهمترین قسمت این صفحه است نوع داده خروجی را تعیین می کنیم. در قسمت Integer rounding mode هم نوع تقریب را معین می کنیم.

در بلوک تقسیم یا Divide هم در صورت دبل کلیک پنجره زیر را میبینیم:



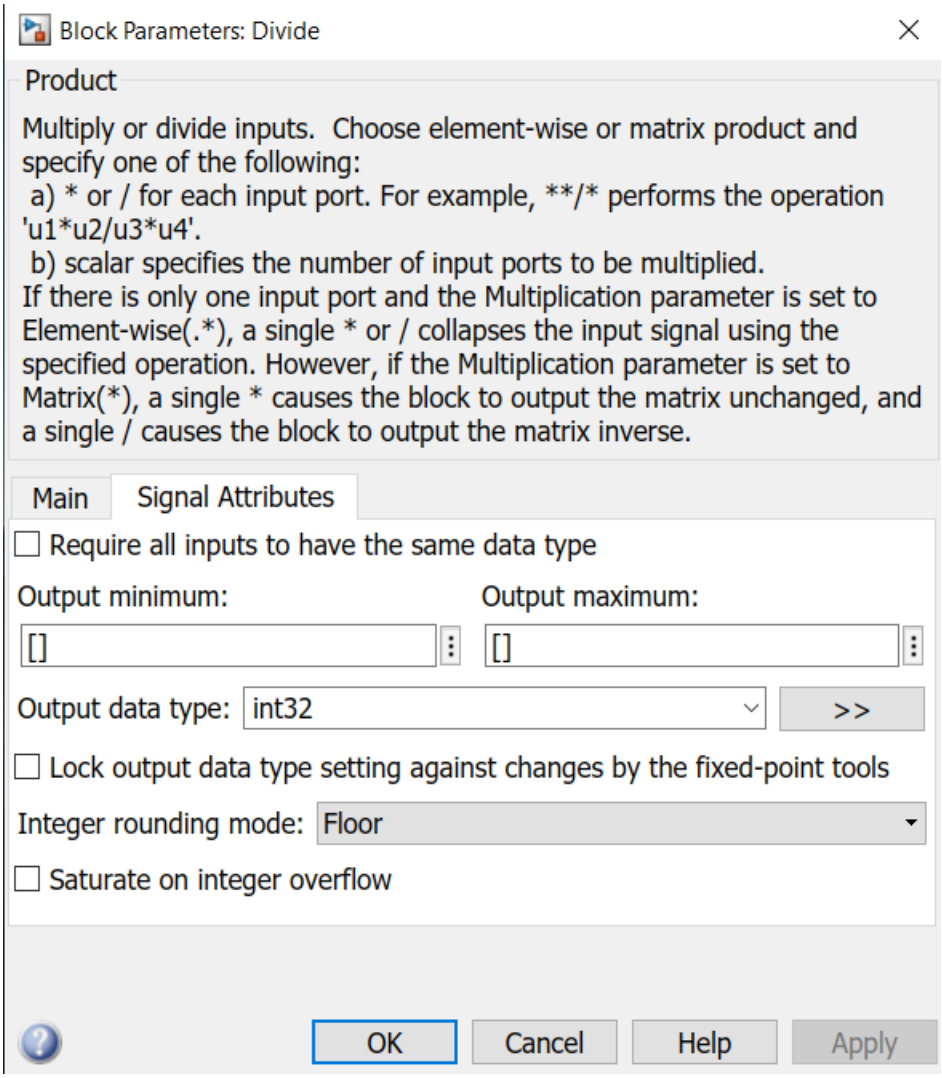
در اینجا اگر ورودی */ باشد آن دیتا که به * وصل است تقسیم بر دیتایی که به / وصل است می شود.

همینطور اگر بخواهیم بلوک ضرب بسازیم کافی است در Number of inputs قرار دهیم **



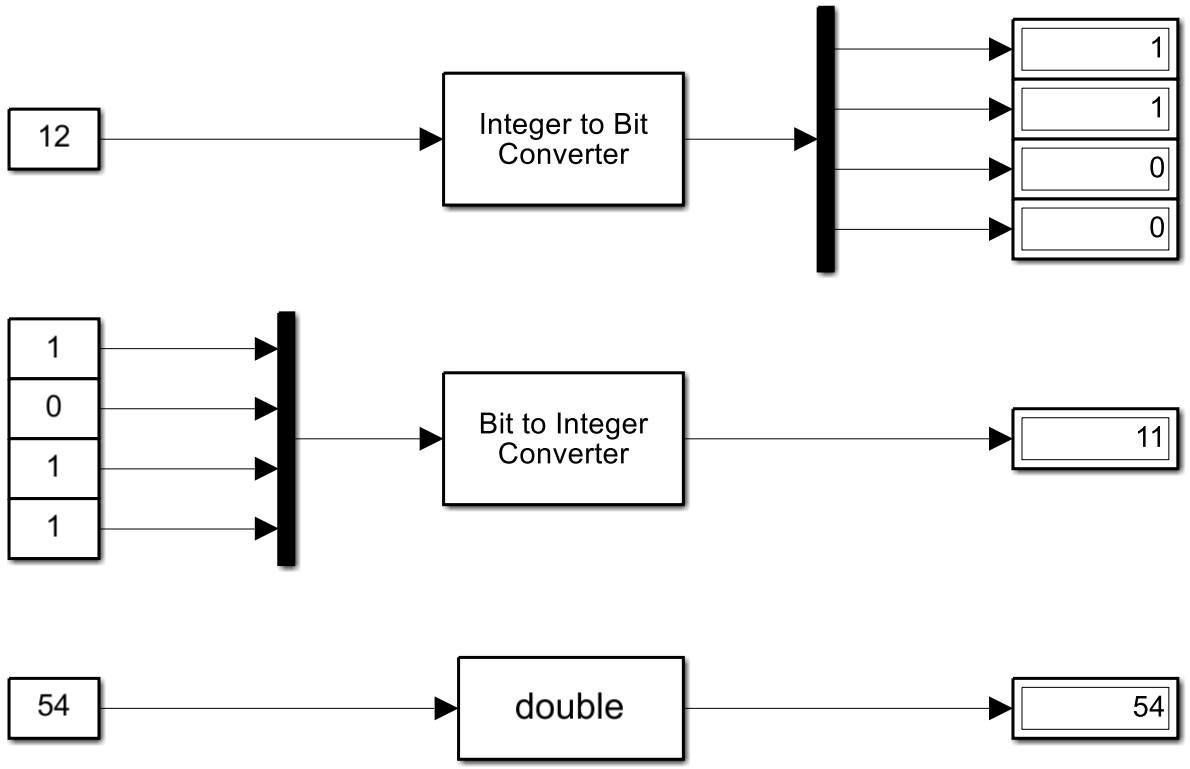
در تب signal Attributes هم صفحه ی بالا را داریم که نوع خروجی و نوع تقریب را مشاهده می کنیم.

حال اگر بخواهیم مثلا اعشاری نداشته باشیم باید همانند زیر تنظیم کنیم:



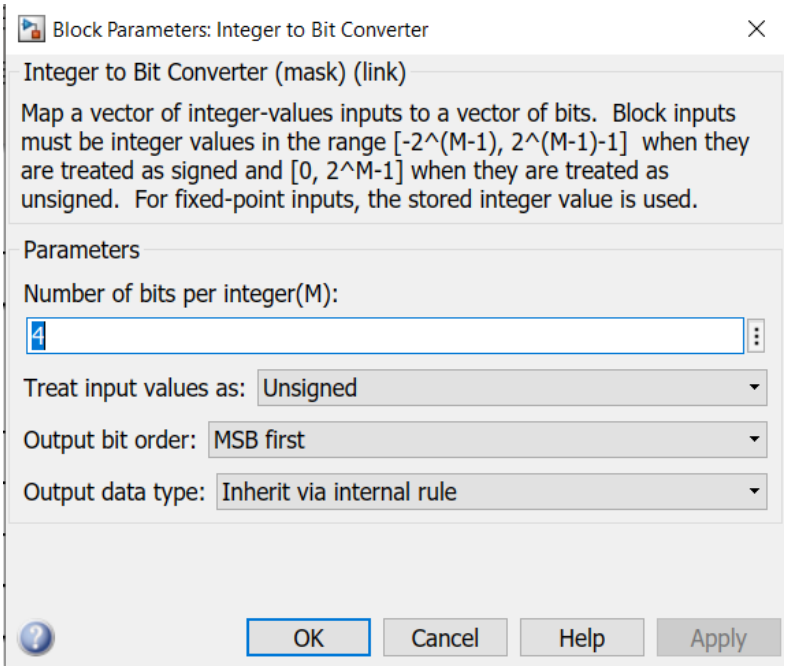
با تنظیم output data type به int32 میتوانیم از باقی مانده تقسیم صرف نظر کنیم.

تبدیل مبنای اعداد با گیت های سیمولینک



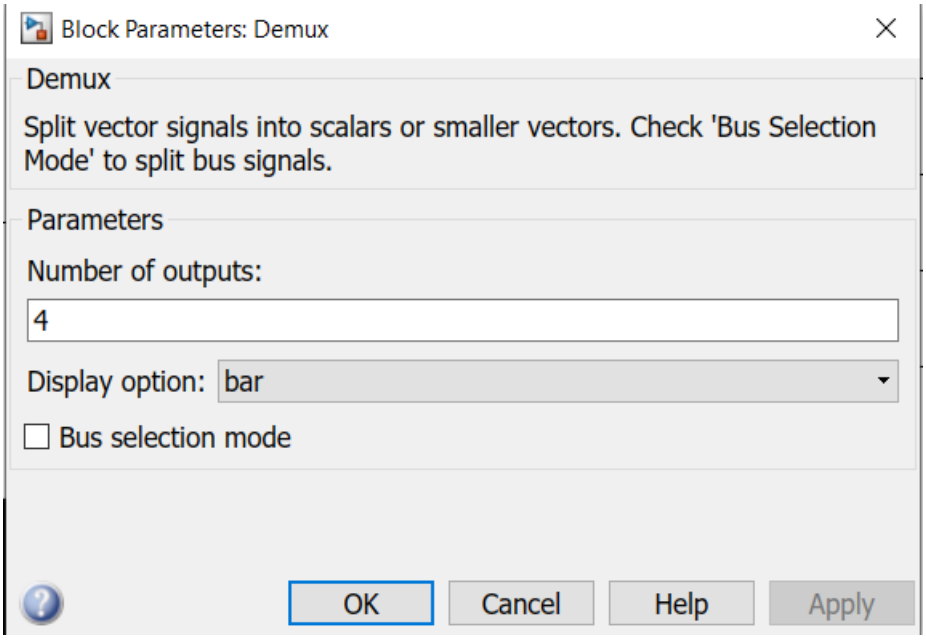
برای تبدیل عدد دسیمال به باینری به یک دیماکس (Demux) و یک گیت تبدیل Integer to Bit Converter نیاز داریم. برای پیدا کردن گیت Integer to Bit Converter کافی است این عبارت را یا حتی عبارت کوتاه شده to bit را در قسمت سرچ جستجو کنیم.

سپس با توجه به اینکه عدد ورودی تبدیل به حداکثر چند بیت خواهد شد بر روی گیت تبدیل عدد به بیت دبل کلیک کرده و سپس تعداد ورودی ها را مشخص می کنیم.



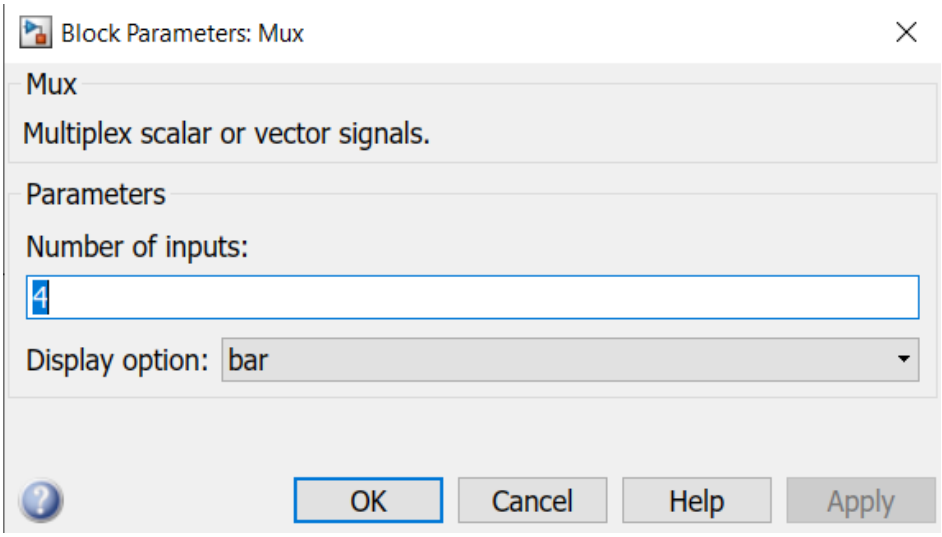
اگر Number of bits per integer(M) ما ۴ باشد ورودی ما میتواند بازه ای بین 0 تا 15 را شامل شود.

همچنین در دیماکس هم باید تعداد خروجی ها را ۴ بکنیم.

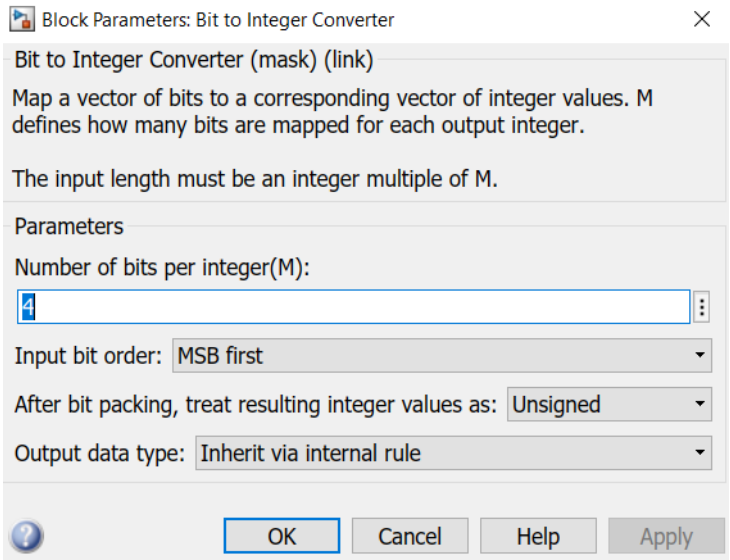


سپس به هر خروجی دیماکس یک Display متصل می کنیم.

برای تبدیل مبنای باینری به دسیمال برعکس عمل می کنیم. یعنی تعداد ورودی های Mux و مبدل باینری به دسیمال را با توجه به بیت های عدد باینری ورودی تنظیم می کنیم.



پنجره پس از دبل کلیک روی Mux



پنجره پس از دبل کلیک روی Bit to integer converter . همانطور که میبینید با توجه به اینکه ورودی ما ۴ بیت است پس باید Number of bits per integer ما هم ۴ باشد.

اگر قضیه تعداد بیت های ورودی و یا خروجی و تعداد ورودی ماکس و خروجی دی ماکس و Number of bits per integer مبدل ها را رعایت نکنیم در نهایت به خطا برمیخوریم.

نکته حائز اهمیتی وجود دارد که ارزش بیت ها چه در ورودی و چه در خروجی از پایین به بالا زیاد می شود. در نتیجه پایین ترین بیت ماکس یا دی ماکس همواره کم ارزش ترین بیت است.

در نهایت به بلوک Cast to double میرسیم که یک مستطیل است که یک ورودی و یک خروجی دارد و روی آن double نوشته شده است. کاررد این بلوک هنگامی است که بخواهیم مثلا یک خروجی int32 را به دبل تبدیل کنیم.

برای این به این تبدیل نیاز داریم چرا که در مولتی پلکسر ممکن نیست ورودی های متفاوتی وارد Mux شوند و همه باید همانند هم باشند.

حال با استفاده از مطالب این فصل و کمی دانش پایه مدار منطقی میتوانیم مدارهای فصل های بعد را تحلیل کنیم.

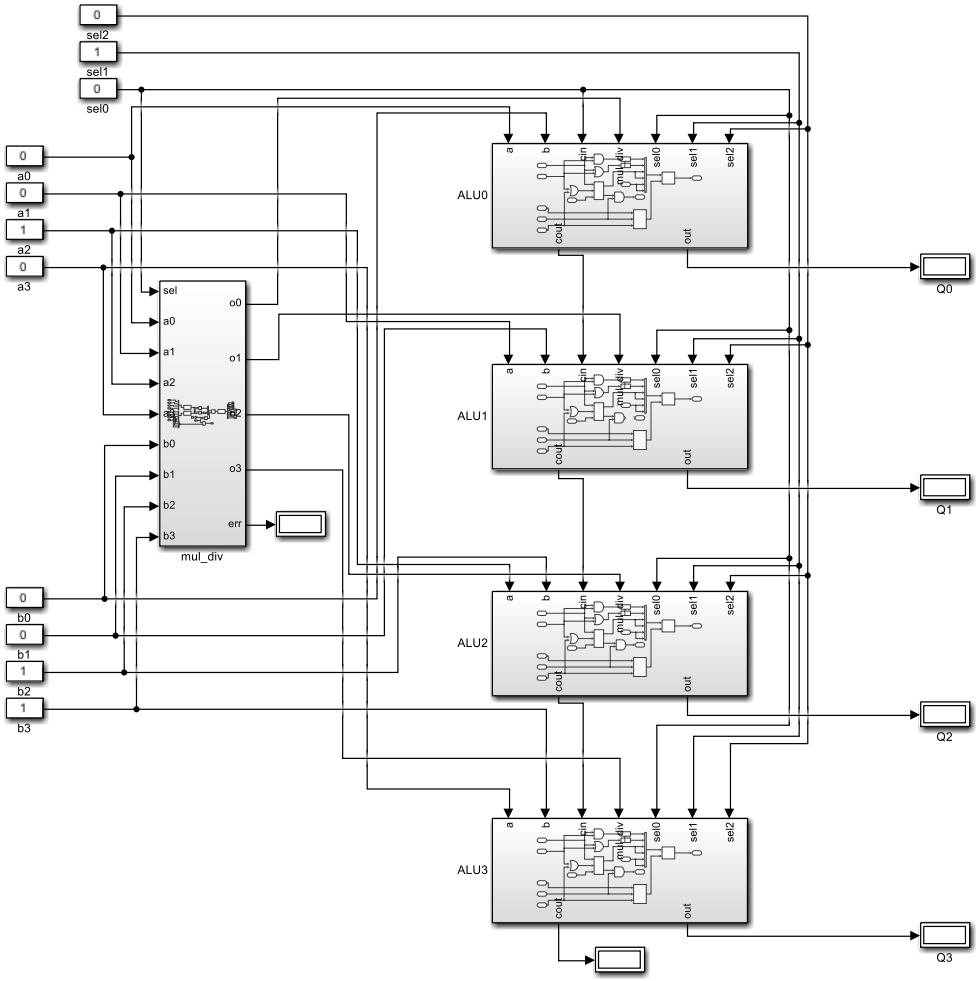
فصل سوم

تشریح فایل متلب پروژه

در این فصل به تشریح پروژه با استفاده از گیت های آماده می پردازیم.

شکل پروژه

پروژه به شکل زیر است:



همانطور که می دانیم این پروژه باید ۶ عمل را انجام دهد. Operation table ما به صورت زیر است:

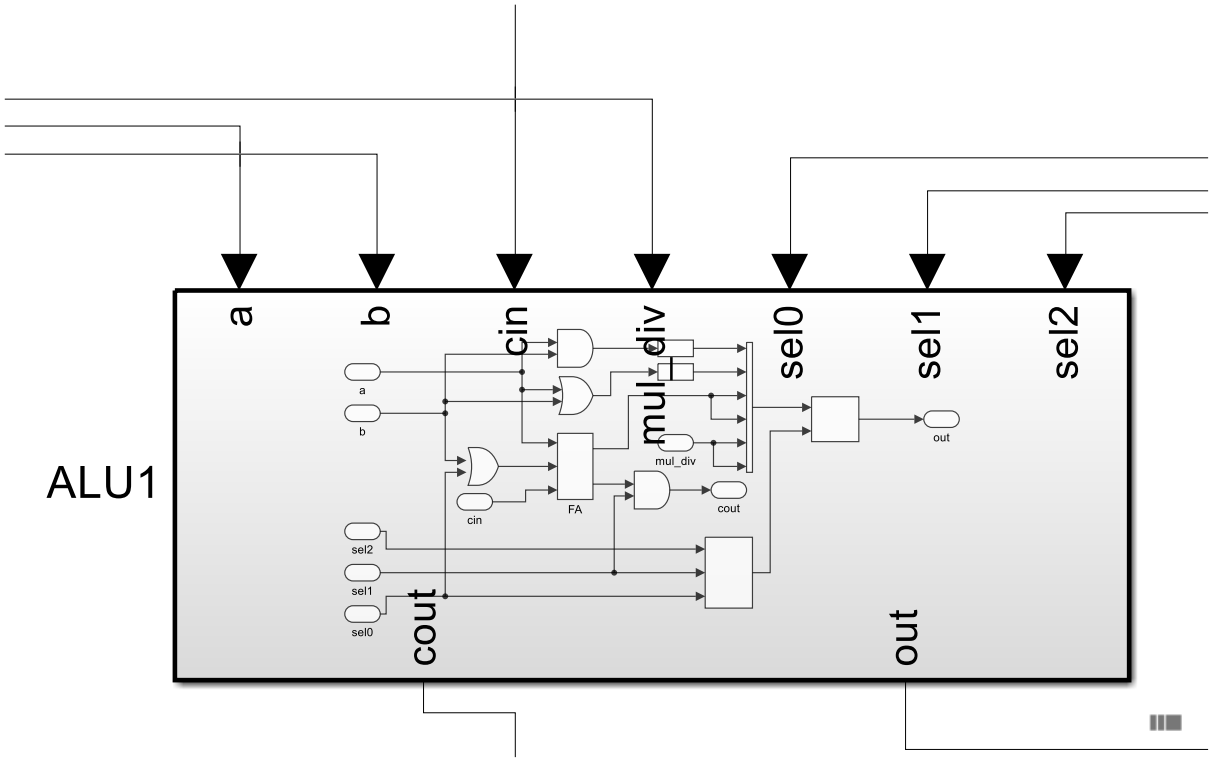
operation	Code
AND	000
OR	001
Addition	010
Subtraction	011
Multiplication	100
Division	101

همانطور که میبینیم دو بلوک اصلی ALU و mul_div را داریم. در بلوک ALU چهار عملیات AND و OR و جمع و تفریق را انجام می دهد. ضرب و تقسیم هم در بلوک mul_div صورت می گیرد.

همچنین حاصل ضرب یا تقسیم به بلوک ALU تزریق می شود.

در ادامه به توضیح این دو بلوک اصلی می پردازیم:

بلوک ALU



بلوک ALU ما به شکل بالاست. همانطور که میبینید در قسمت ورودی ها ما:

یک ورودی برای عدد اول (a)

یک ورودی هم برای عدد دوم (b)

یک کری ورودی برای عملیات های جمع و تفریق

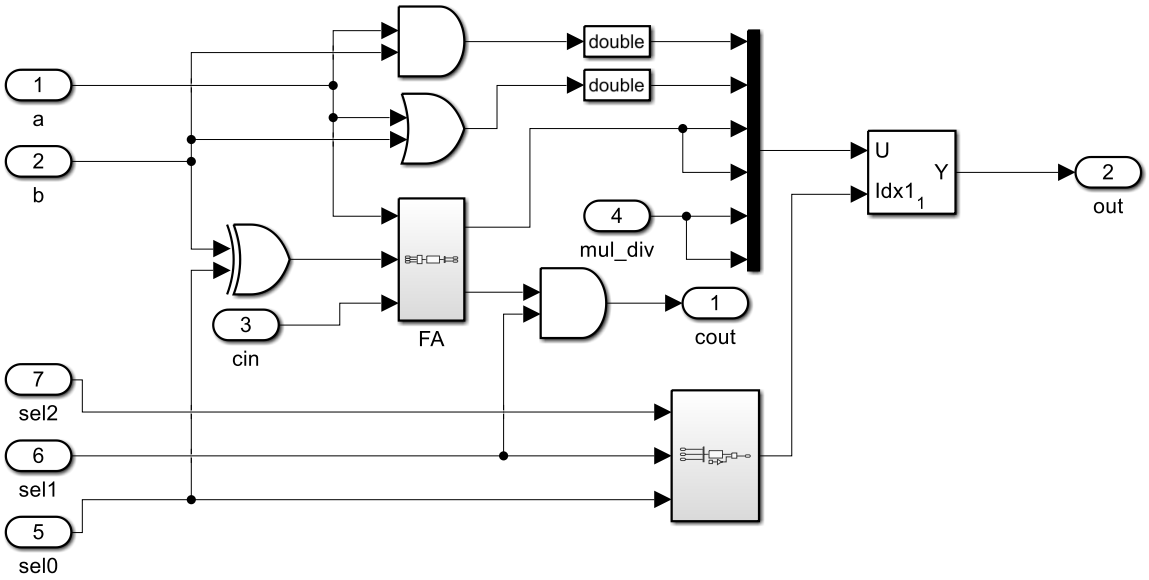
ورودی ضرب و یا تقسیم

و همینطور ۳ ورودی سلکت برای انتخاب operation داریم.

در قسمت خروجی ها هم دو خروجی به صورت زیر داریم:

یک کری خروجی که به کری ورودی بلوک ALU بعدی متصل می شود و همینطور خروجی اصلی که به گیت display متصل است.

درون این بلوک به صورت زیر است:



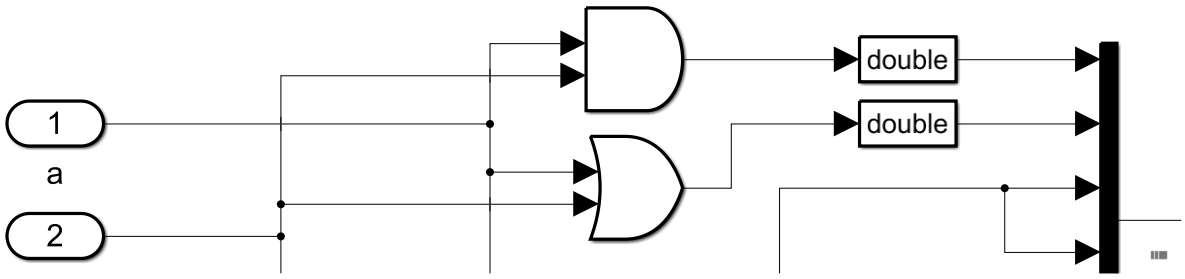
در اینجا ما یه مولتی پلکسر با ۶ ورودی داریم. البته به راحتی میشد شامل ۸ ورودی ساخت که دو ورودی آخر صفر باشند تا وقتی در پایه های سلکت 111 داشتیم به ارور بر نخوریم.

ورودی اول AND

برای این ورودی تنها کافی بود همانطور که در جزوه گفته شد از یک گیت AND استفاده کنیم که ورودی هایش ورودی های اصلی ما یعنی a و b باشد. البته با توجه به مطالب فصل پیشین باید بدانیم که تمام ورودی های مولتی پلکسر باید از یک جنس باشند. پس با استفاده از گیت cast to Double خروجی Boolean را تبدیل به double می کنیم.

ورودی دوم OR

برای این ورودی هم کافی است از یک گیت OR استفاده و ورودی های اصلی alu را به ورودی های OR متصل کنیم. خروجی آن را هم همانند AND به cast to double و خروجی این مبدل را هم به ورودی دوم مولتی پلکسر متصل می کنیم.



ورودی سوم و چهارم جمع و تفریق

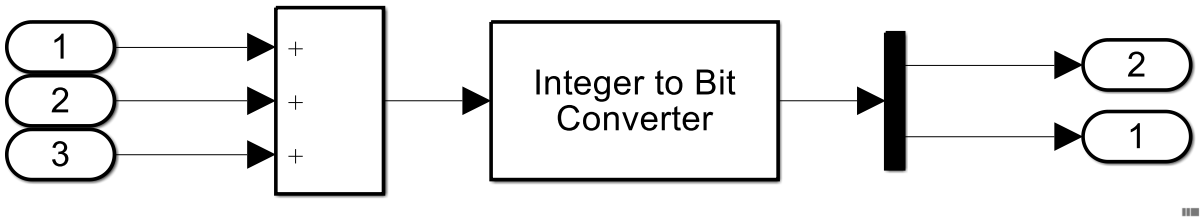
برای اینکار از یک فول ادر استفاده کردیم. همانطور که میدانید فول ادر سه ورودی دارد و دو خروجی.

حاصل جمع این سه ورودی به شکل باینری در فول ادر ما قابل مشاهده است.

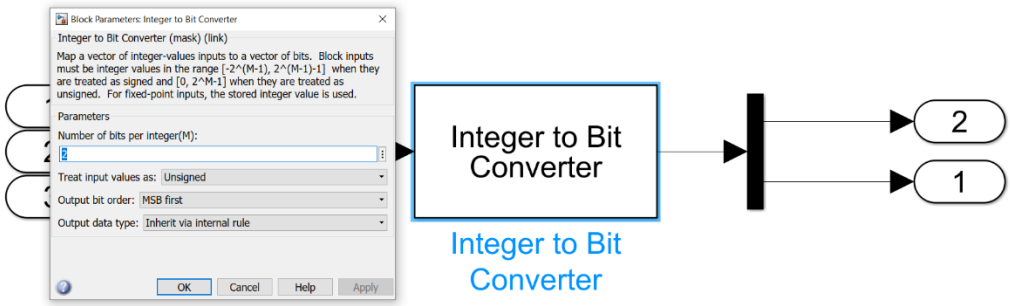
Truth table فول ادر ما به شکل زیر است:

Input 1	Input 2	Input 3	carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

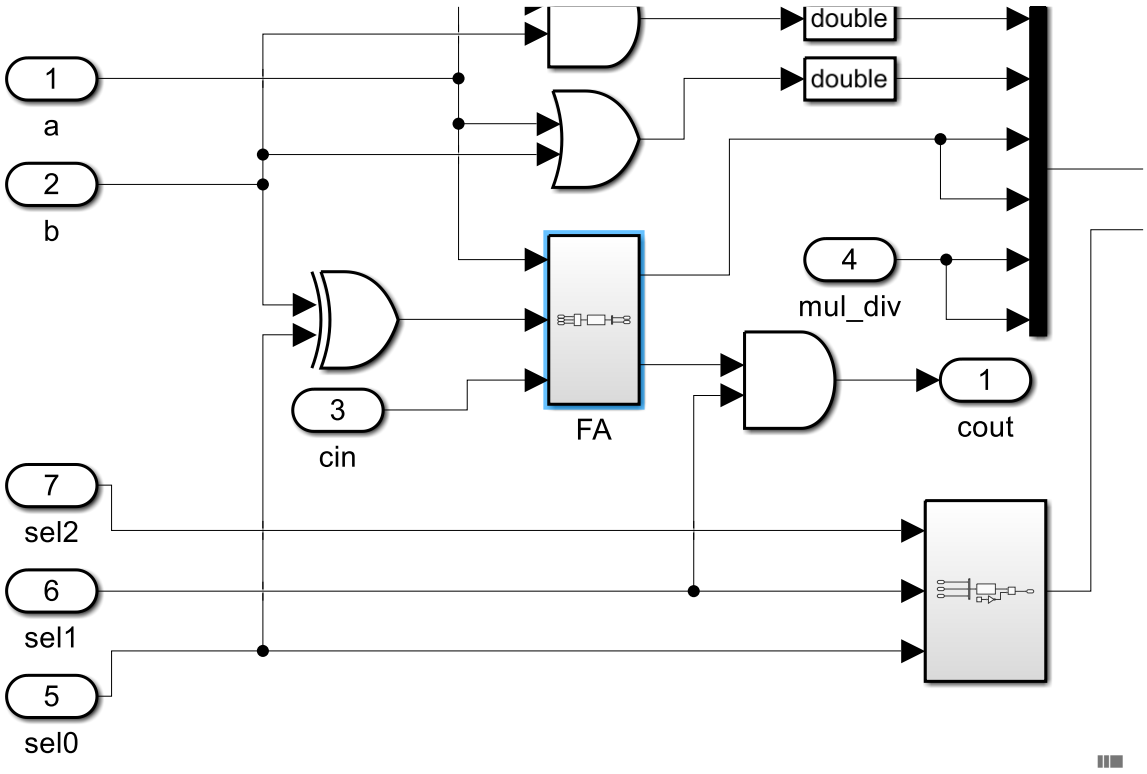
درون مدار فول ادر ما هم به شکل زیر است:



همانطور که پیداست ابتدا سه بیت (عدد معمولی – فرقی ندارد) را با هم جمع کردیم. جواب که بین صفر تا دو است را تبدیل به یک عدد باینری کردیم و با استفاده از دی ماکس آن را به خروجی ها بردیم. البته شماره خروجی ها را طوری انتخاب کردیم که در بلوک بیت پایینی پر ارزش تر باشد که همان کری ماست.



تعداد خروجی های ما ۲ تاست پس number of bits per integer را هم ۲ قرار می دهیم.

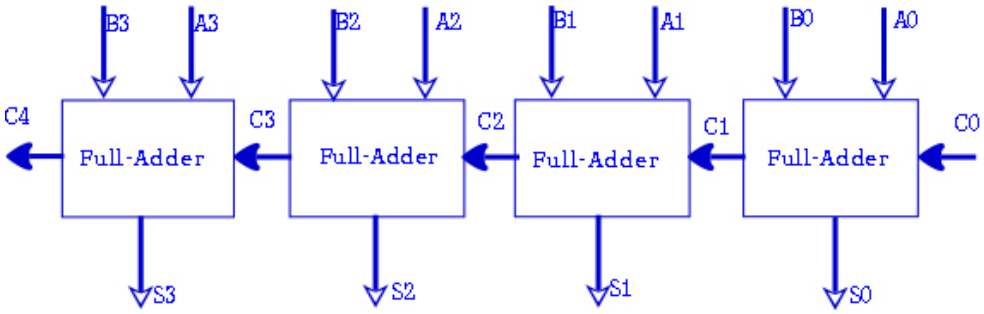


همانطور که میبینید خروجی اصلی ما به ورودی سوم و چهارم مولتی پلکسر متصل می شوند.

کری فول ادر هم با یک and به کری خروجی alu متصل می شود. دیگر پایه and متصل به پایه دوم سلکتور است. با توجه به اینکه کری ما فقط در جمع مورد استفاده است و تنها در جمع و تفریق کری ما مورد استفاده قرار می گیرد.

با توجه به اینکه operation code ما در جمع 010 است و در تفریق هم 011 پس کافی است تنها وقتی بیت دوم operation code ما یک باشد از کری استفاده شود.

می دانیم که در جمع دو عدد ۴ بیتی مدار زیر به درستی جواب را به ما می دهد:



در این مدار ما هم همین مدار را با استفاده از کری ورودی و خروجی بلوک های ALU پیاده سازی کرده ایم.

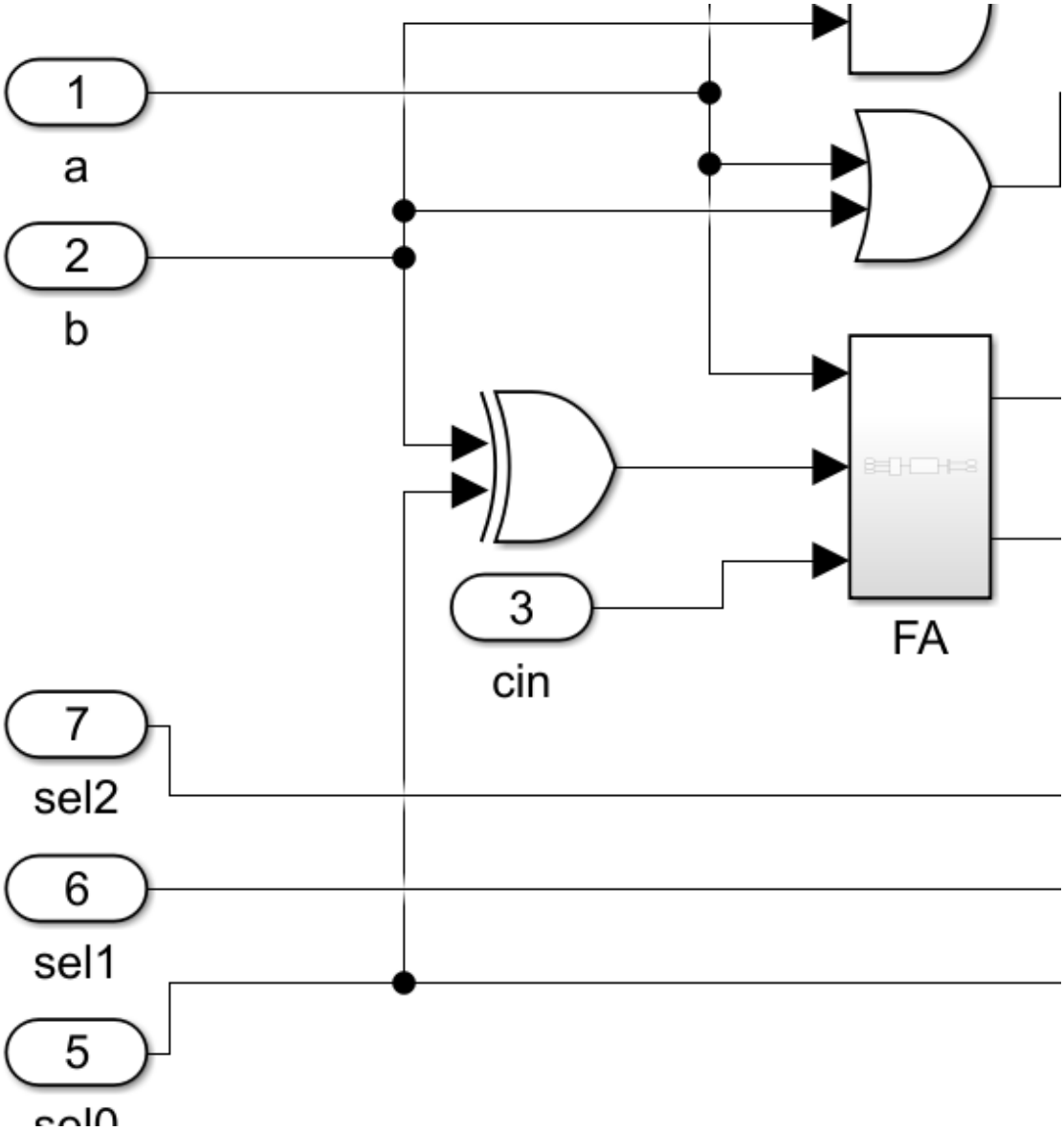
اما تفریق چگونه کار می کند؟؟

همانطور که میدانیم دو عبارت زیر کاملا برابر است:

$$A - B = A + (-B)$$

و برای بدست آوردن حاصل منفی برابر یک عدد کافی است (در اعداد باینری) یک ها را صفر و صفرها را یک کنیم. در مدار منطقی گیت xor ای که یک پایه اش به یک وصل باشد اینکار را برای ورودی دیگر انجام می دهد.

حال در اینجا هم همینکار را انجام میدهیم:

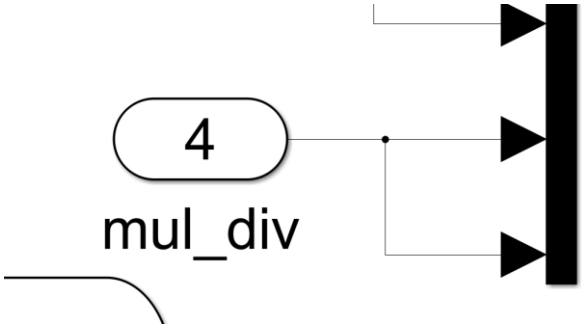


این xor را با بیت اول سلکتور انجام میدهیم. چرا که در اینجا کافی است دو عملیات جمع (010) و تفریق (011) را از هم تمیز دهیم و تفاوت این دو تنها در بیت اول است پس از بیت اول سلکتور پایه ی دوم xor را میگیریم.

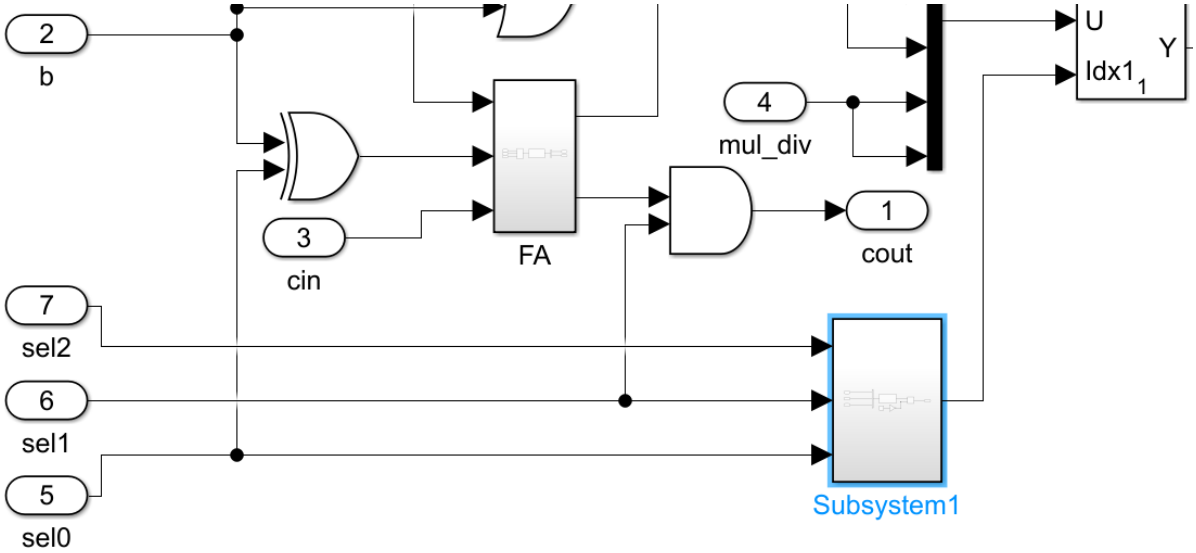
با این منطق تنها هنگامی که تفریق داشته باشیم عددی متمم b وارد فول ادر می شود.

ورودی پنجم و ششم

* در ورودی پنجم و ششم مولتی پلکسر هم حاصل ضرب و تقسیم وارد آن ها می شود. این دو ورودی در خارج از بلوک alu تولید می شوند پس فعلا به طریقه بدست آوردن این دو قسمت نمی پردازیم.

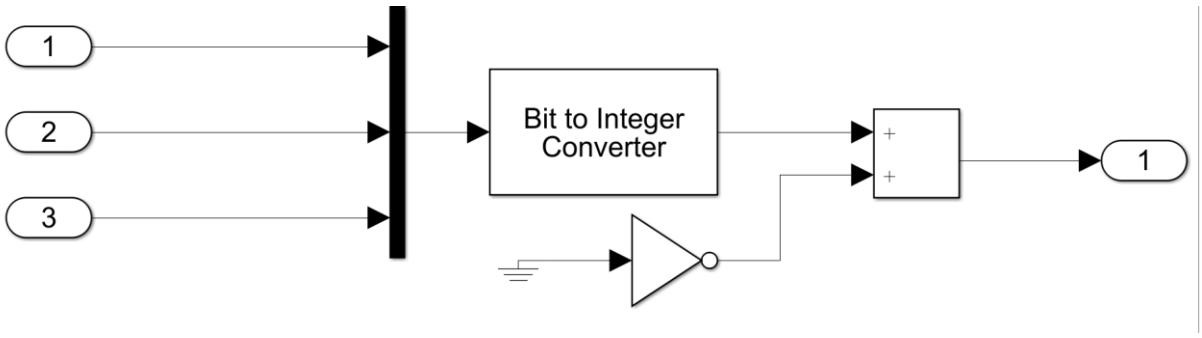


طریقه کارکردن مولتی پلکسر



قبلا با نحوه کارکرد مولتی پلکسر آشنا شده بودیم. در این جا ما ۳ پایه سلکت داریم که حاوی یک عدد به صورت باینری هستند. این سه پایه وارد یک باکس می شوند که خروجی اش یک عدد دسیمال است.

درون باکس به صورت زیر است:



همانطور که میبینید یک عدد باینری تبدیل به دسیمال شده (قبلا نحوه تبدیل گفته شده بود) و عدد حاصل هم با یک جمع شده چرا که اندیس های مولتی پلکسر از یک شروع می شود نه صفر.

عدد یک از not کردن زمین که همیشه صفر است بدست آمده.

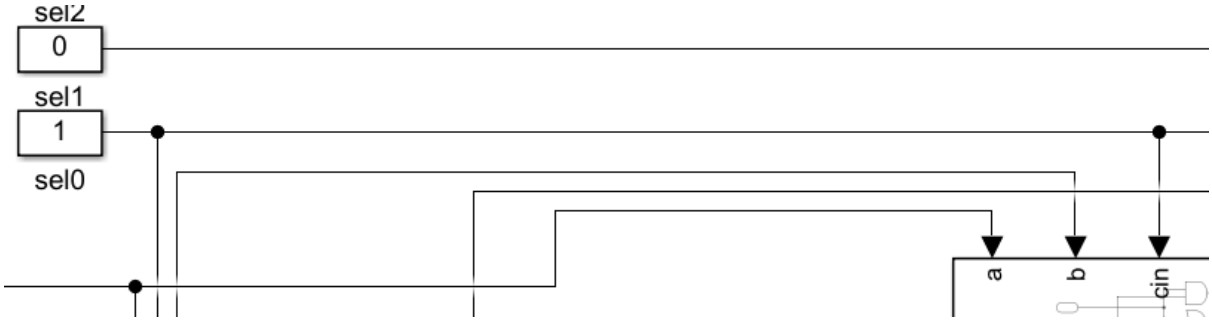
همچنین ورودی دارای ترکیبی همانند استاندارد است و طوری port number ها چیده شده که با ارزش ترین بیت در بالاترین مکان بلوک باشد.

حاصل این بلوک به ورودی کنترل مولتی پلکسر وارد می شود.

همانطور که قبلا گفته شده بود تمامی ورودی های مولتی پلکسر باید از یک جنس باشد. در اینجا همه از نوع double هستند.

همچنین عملگرهای and و or با توجه به اینکه مقدار carry ندارند دارای استقلال بوده و مقادیر این دو عملگر در بلوک های ALU به هم مرتبط نیستند. اما در مورد جمع و تفریق اینطور نیست و با یک کری به هم مرتبط گشته اند. کری های ورودی هر ALU کری خروجی ALU قبلی متصل شده اند.

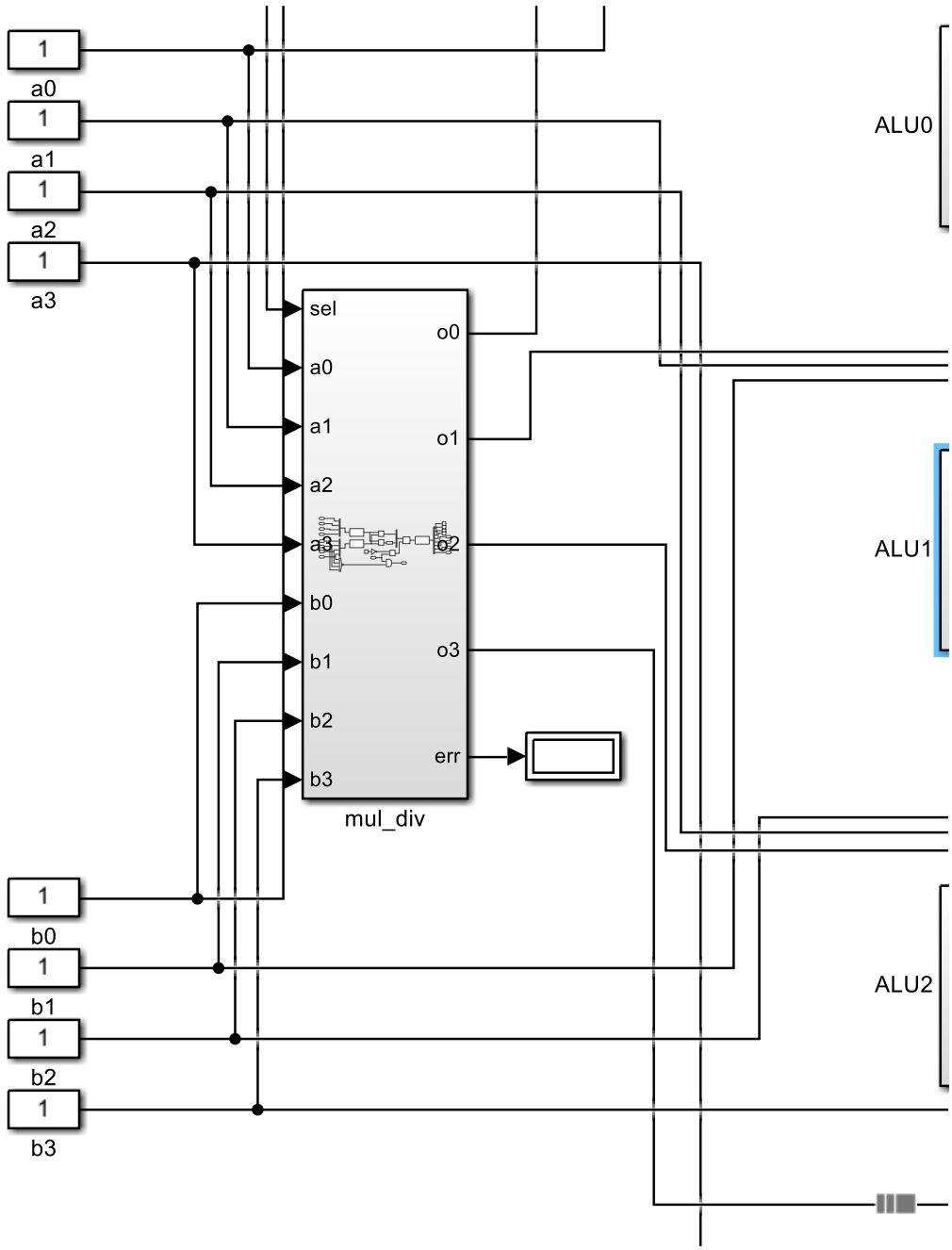
همچنین ورودی کری ورودی بلوک اول به بیت کم ارزش سلکتور متصل شده ...



این مقدار در تفریق یک و در جمع صفر خواهد بود و در دیگر operator ها بی تاثیر خواهد بود.

همچنین مقادیر ضرب و تقسیم هر ALU هم کاملا وابسته به دیگر ALU ها هستند. اگر میخواستیم ساختاری داشته باشیم که ضرب و تقسیم به صورت بیرون ALU نباشد نیاز به چندین کری ورودی و خروجی دیگر داشتیم.

بلوک mul_div

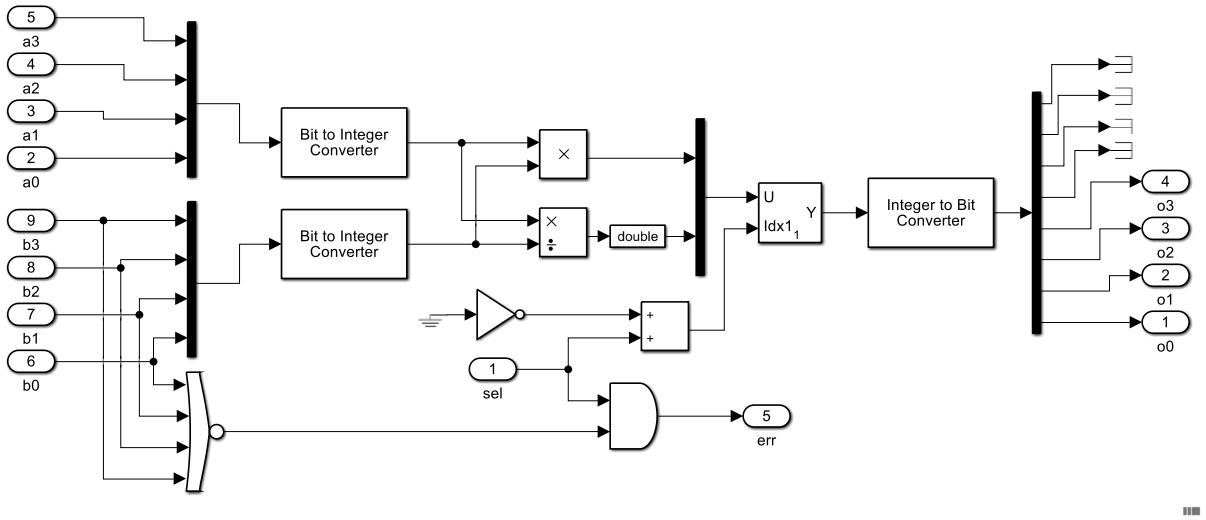

1
a0
1
a1
1
a2
1
a3
sel
a0
a1
a2
a3
b0
b1
b2
b3
o0
o1
o2
o3
err
mul_div
ALU0
ALU1
ALU2
1
b0
1
b1
1
b2
1
b3

این بلوک هم شامل 9 ورودی است که دو ۴ بیت داریم که هر یک از ورودی ها به ۴ تای آن وارد می شوند. همچنین یک ورودی سلکت هم داریم. در خروجی هم ۴ خروجی اصلی داریم که این خروجی ها به بلوک های ALU هم ایندکس خود تزریق می شوند.

یک خروجی هم تحت عنوان ارور داریم که تنها وقتی یک می شود که یک عدد تقسیم بر صفر شده باشد و به ما وجود این خطا را میرساند.

خروجی تقسیم دو عدد ۴ بیتی نهایتا ۴ بیت است اما این برای ضرب درست نیست و ما در برخی ورودی ها به ۸ بیت برای خروجی هم نیاز داریم ولی طبق فرض مسئله ما تنها ۴ بیت کم ارزش تر جواب را در نظر میگیریم.

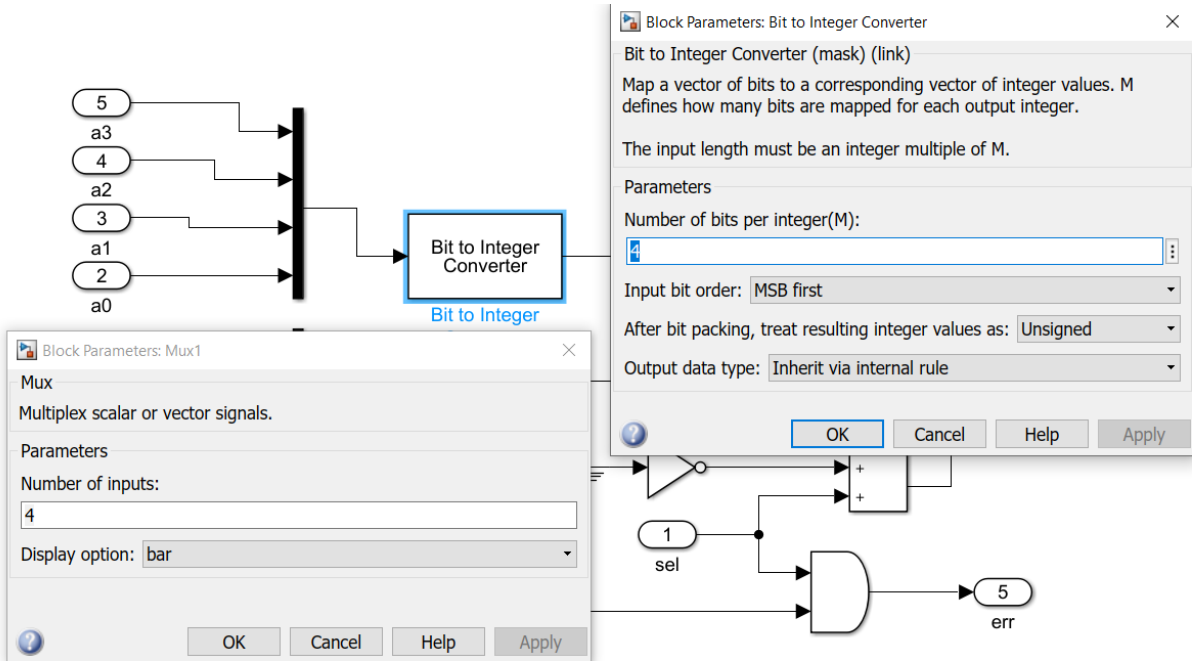
درون این بلوک به شکل زیر است:



همانطور که مشخص است باز هم یک مولتی پلکسر داریم که حاصل تقسیم و ضرب را از هم تمیز میکنیم.

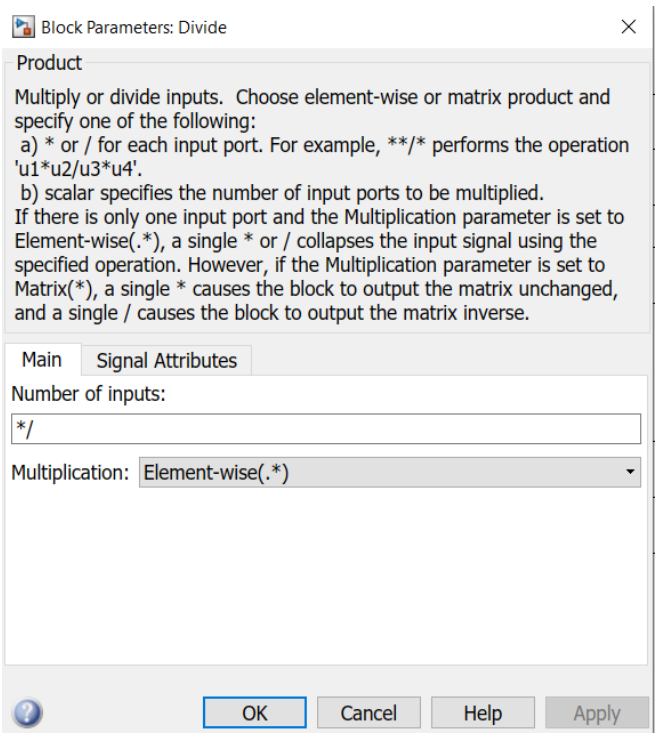
ورودی ها و خروجی ها هر دو برعکس حالت استاندارد است و با part number کاری کردیم که در بلوک خارجی بیت با ارزش پایین ترین بیت باشد، چه در ورودی، چه در خروجی.

ظرفیت mux و converter ما ۴ است:

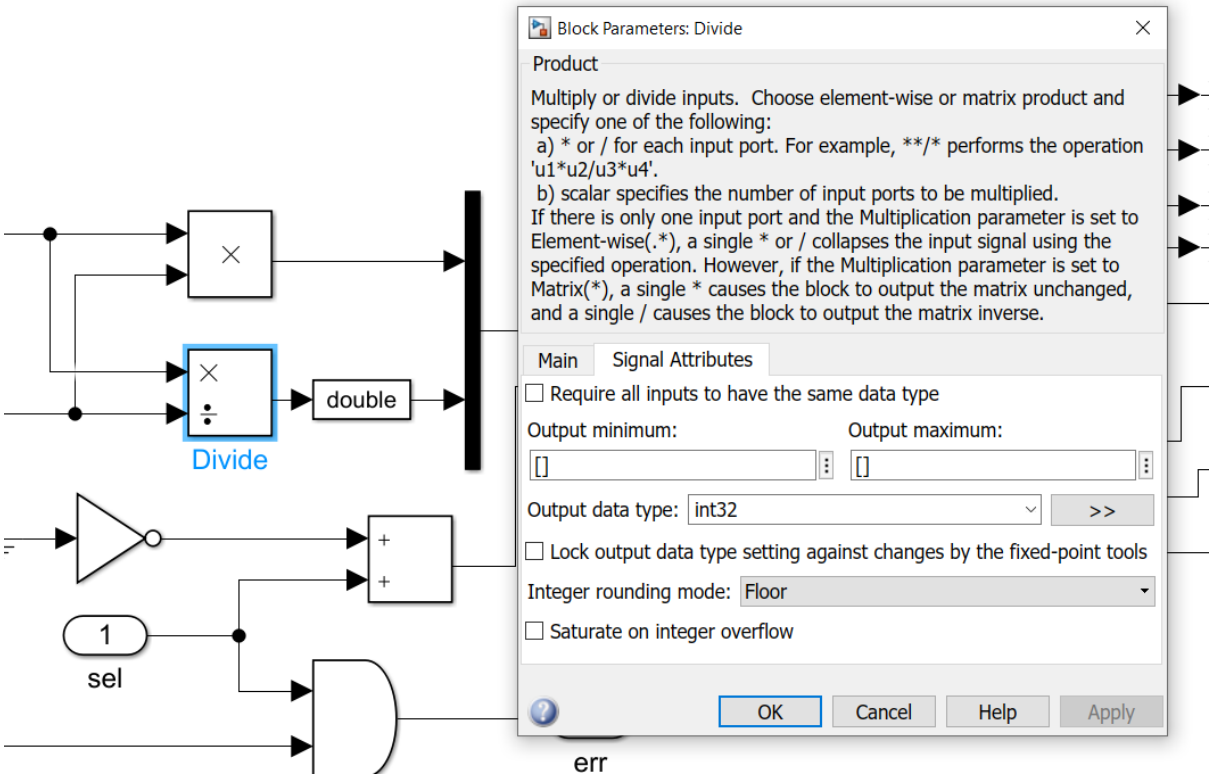


ابتدا با استفاده از یک mux برای هر ورودی عدد را به بلوک bit to integer Converter میبریم تا ورودی ها به اعداد دسیمال تبدیل شوند. (double) سپس هم حاصل ضرب و هم حاصل تقسیم را حساب می کنیم. این عملیات ها با بلوک Divide انجام می شود که در کتابخانه Math Operators می باشد.

برای ضرب ورودی ** است و در تقسیم هم /* می باشد.

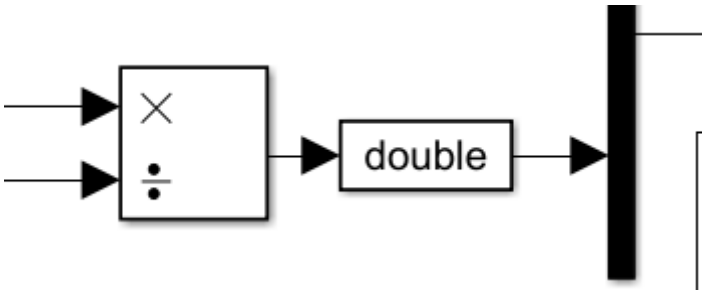


برای تقسیم به این علت که اعداد به هنگام بخش پذیر نبودن به صورت اعشاری نشوند باید خروجی این بلوک را به صورت int32 گذاشت. (تنها تفاوت int32 و int16 و int8 در مقدار ماکسیممی است که میتوانند تقسیم کنند)

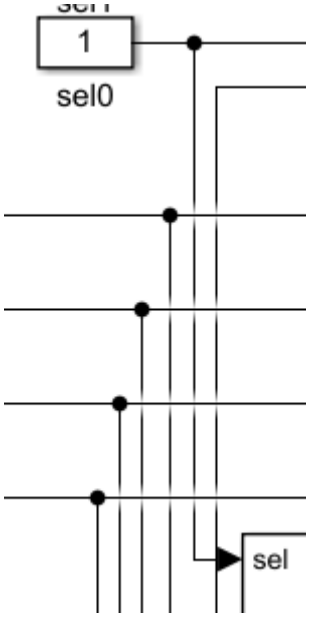


همچنین به علت این که همه ورودی های مولتی پلکسر باید یکسان باشند پس یا باید خروجی ضرب را int32 کرد یا حاصل تقسیم را double کرد.

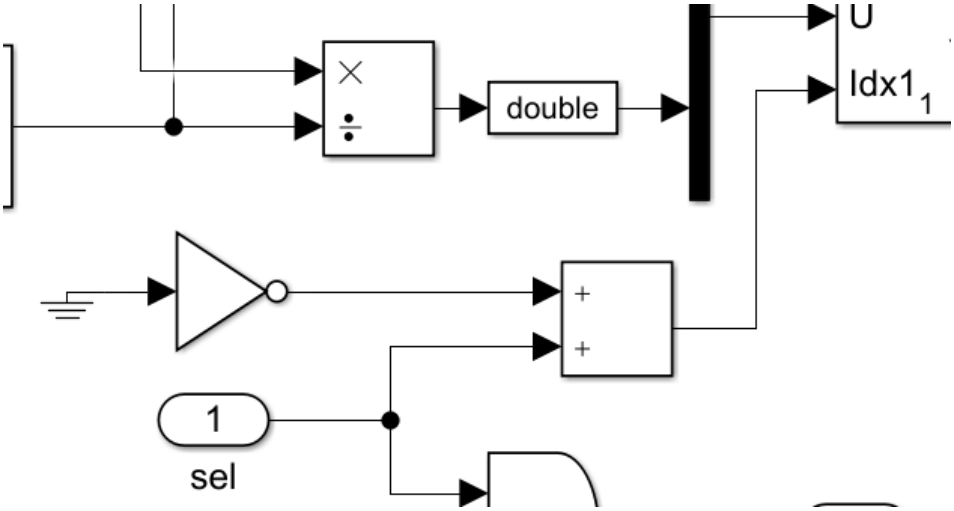
با توجه به اینکه در بلوک ALU تمام ورودی های مولتی پلکسر double بودند پس در این دو راهی double را انتخاب می کنیم و حاصل تقسیم را که int32 است به double تغییر می دهیم. (با استفاده از بلوک cast to double)



با توجه به اینکه operation code ضرب 100 است و تقسیم 101 است پس کافی است بیت سلکت اول را که در این دو متفاوت است را به عنوان سلکت بلوک mul_div بفرستیم.

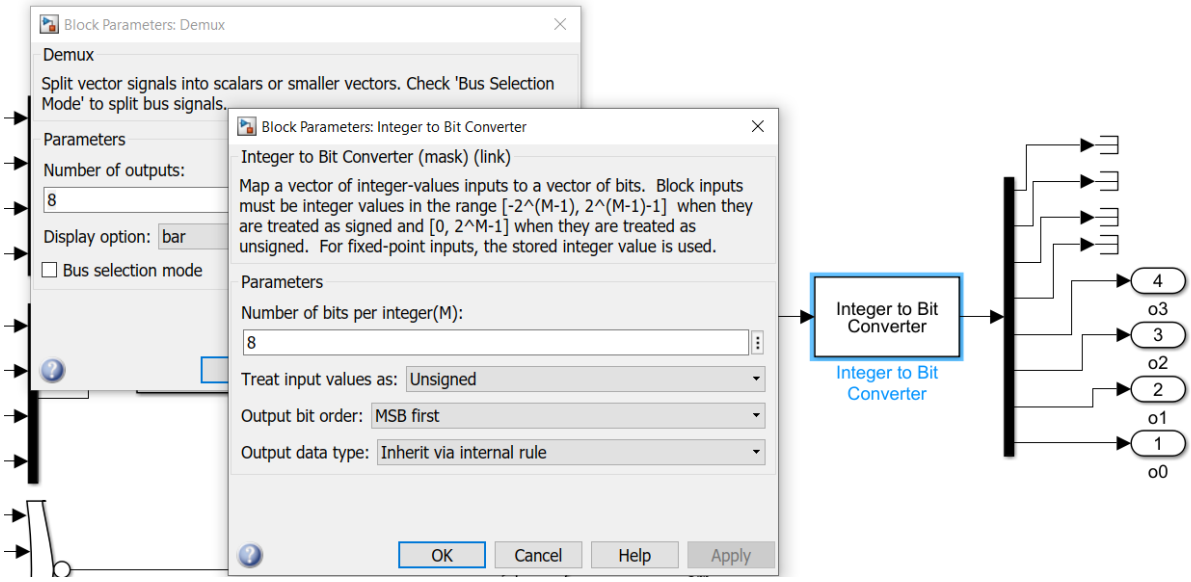


با توجه به اینکه اندیس های مولتی پلکسر آماده از یک شروع می شود لذا بیت سلکت را با یک جمع می کنیم. (این یک از not گرفتن زمین یا Ground بدست آمده)



در انتها هم حاصل تقسیم یا ضرب (با توجه به انتخاب ما) به بلوک integer to bit Converter می رود تا باز تبدیل به باینری شود.

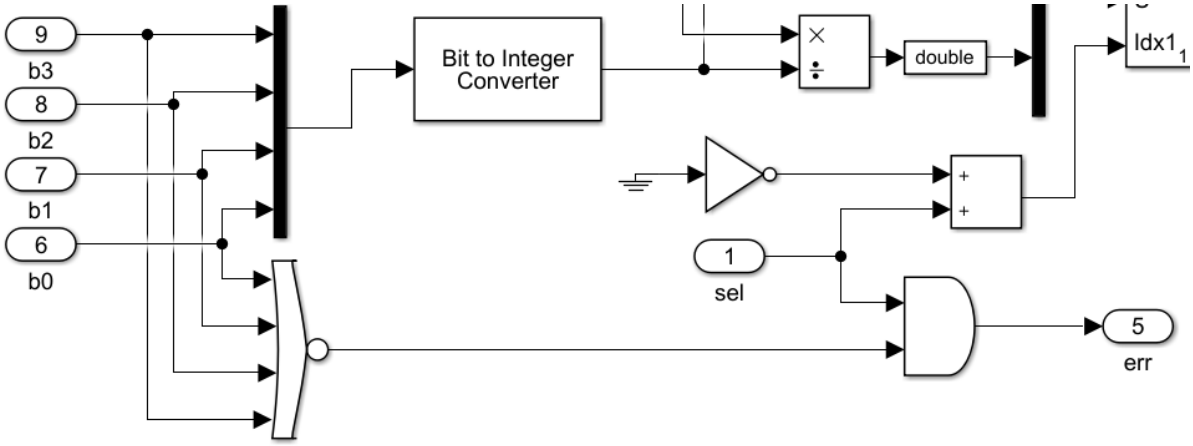
در اینجا با توجه به اینکه خروجی ضرب تا ۸ بیت میتواند باشد پس ظرفیت converter و demux را 8 انتخاب می کنیم:



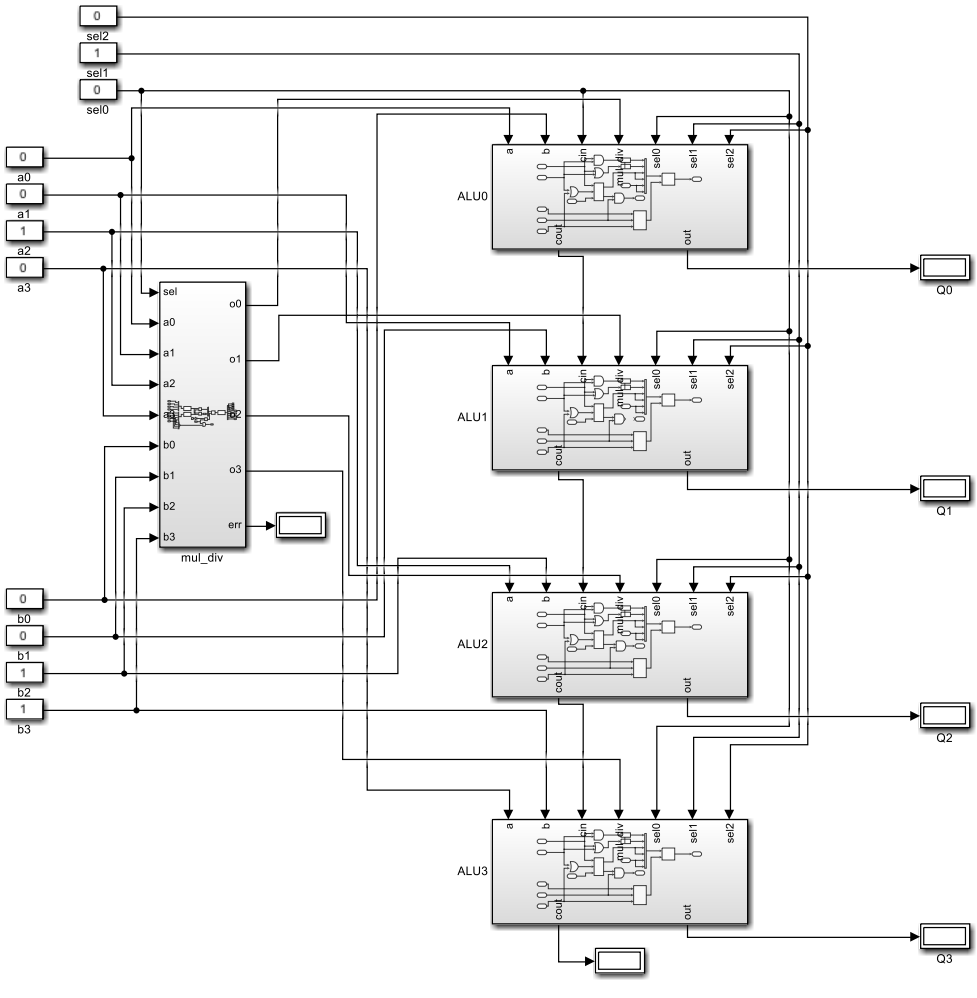
البته با توجه به قراردادی که داشتیم تنها بیت های کم ارزش برای ما اهمیت داشتند پس برای چهار خروجی با ارزشمان از ترمینال استفاده میکنیم تا این خروجی ها تبدیل به خروجی برای بلوک ما نشوند.

برای گرفتن ارور هنگامی که در تقسیم عددی بر صفر تقسیم می شود نیاز است که همه بیت های b صفر باشند. برای بررسی چنین وضعیتی میتوانیم از تمام بیت ها not گرفته و خروجی همه not ها را به and متصل کنیم. البته ترکیب این not ها و and به گیت nor را میدهد. (حباب های ورودی and به خروجی منتقل می شوند و and به or تبدیل می شود.)

شرط دوم هم این است که عملیات ما تقسیم باشد، پس کافی است پایه سلکت ما یک باشد. برای بررسی این وضعیت کافی است از حاصل nor با پایه سلکت and بگیریم. در نهایت به خروجی ای به نام err خروجی آخر را متصل می کنیم:



در این قسمت هم به اتمام بررسی بلوک mul_div میرسیم.



حالت کلی

در حالت کلی ما 3 پایه سلکت داریم که با آن یکی از ۶ عملگر را انتخاب می کنیم.

همچنین دو عدد ۴ بیتی هم در ورودی هم داریم. هر بیت از هر یک از ورودی ها یک بار مستقیم به بلوک ALU و بار دیگر به mul_div متصل می شود.

دو خروجی کمکی هم داریم که یکی کری برای جمع و دیگری ارور برای تقسیم است. در خروجی هم ۴ خروجی داریم.

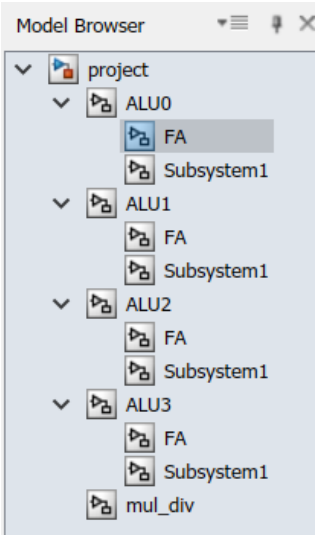
در and و or این چهار بیت تنها خروجی های قابل توجه ما خواهند بود.

در جمع کری هم جزو خروجی هاست.

در تفریق هم این چهار خروجی، خروجی های اصلی اند که در صورتی که a از b بزرگ تر باشد حاصل مثبت است. اگر b از a بزرگ تر باشد هم حاصل منفی است.

البته اگر در نمونه ورودی را طوری بدهیم که خروجی حاصل منفی هم بدست بیاورد باید طبق استانداردی حاصل نهایی بین 8- تا 7 باشد چرا که اعداد منفی را بر مبنای متمم دو نشان میدهد و این استاندارد برای نمایش اعداد بزرگ تر از ۷ و کوچکتر از منفی ۸ دچار خطا می شود.

در ضرب و تقسیم هم خروجی های ما همین ۴ بیت هستند، هر چند که در تقسیم بیت اعلام خطا هم داریم.



خروجی های این مدار را برای مقادیری استاندارد و رندم در فصل نمونه ورودی و خروجی میتوانید ببینید.

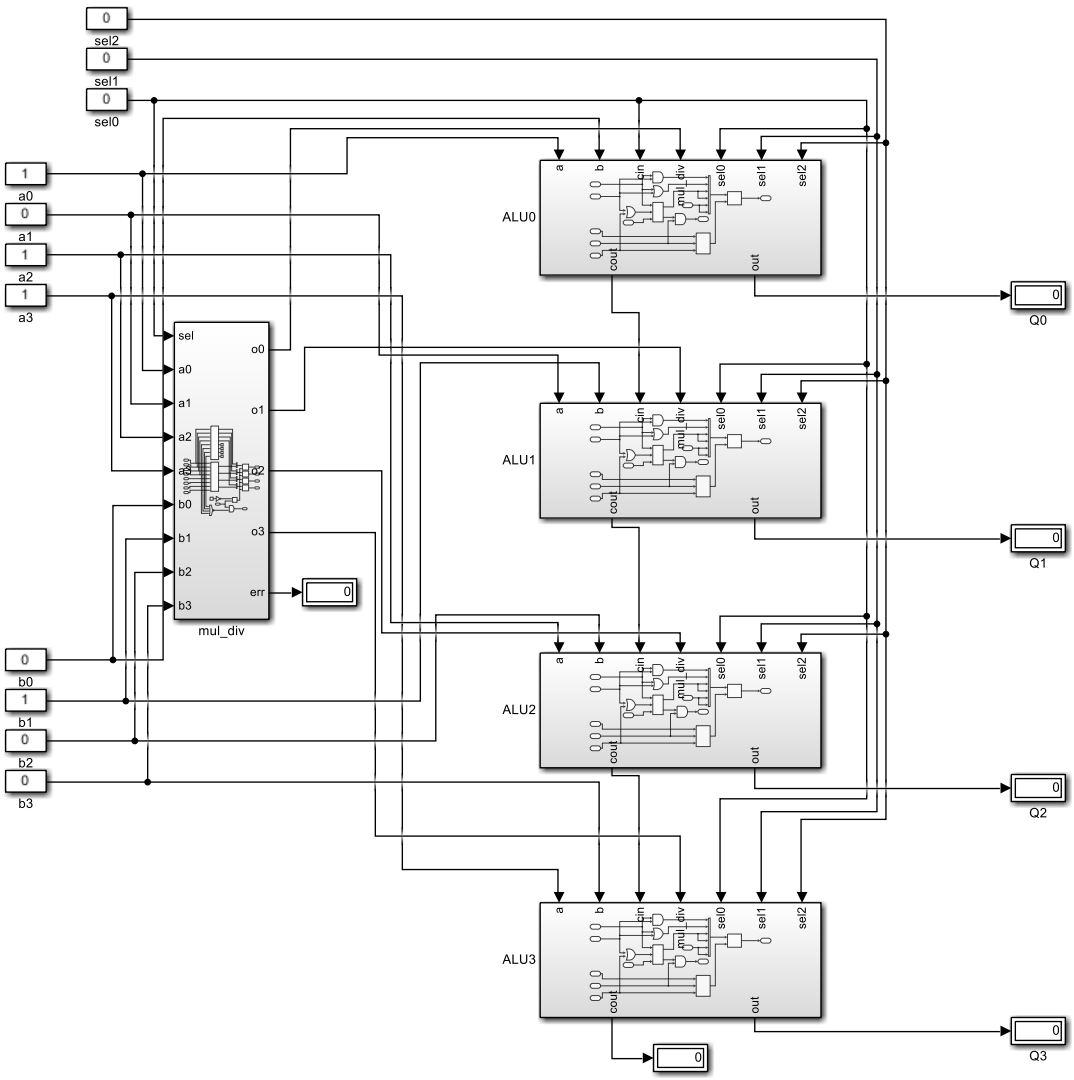
فصل چهارم

پروژه با گیت های دستی

در این فصل به بررسی فایل دوم پروژه که در تقسیم و ضرب و جمع و تفریق از گیت های آماده ی کتابخانه math operation استفاده نکرده می پردازیم.

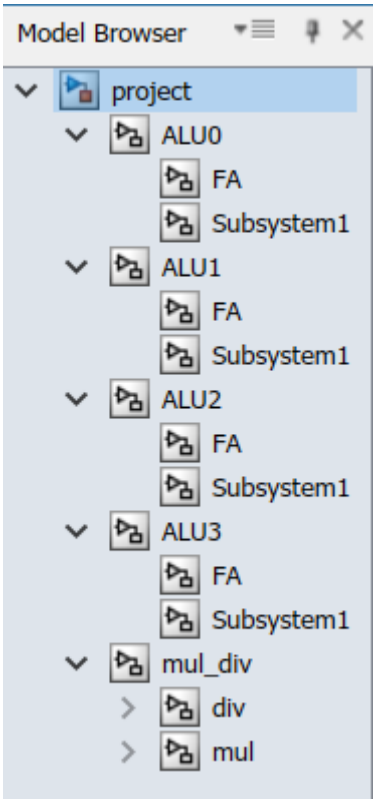
شکل پروژه

شکل کلی پروژه به شکل زیر است:



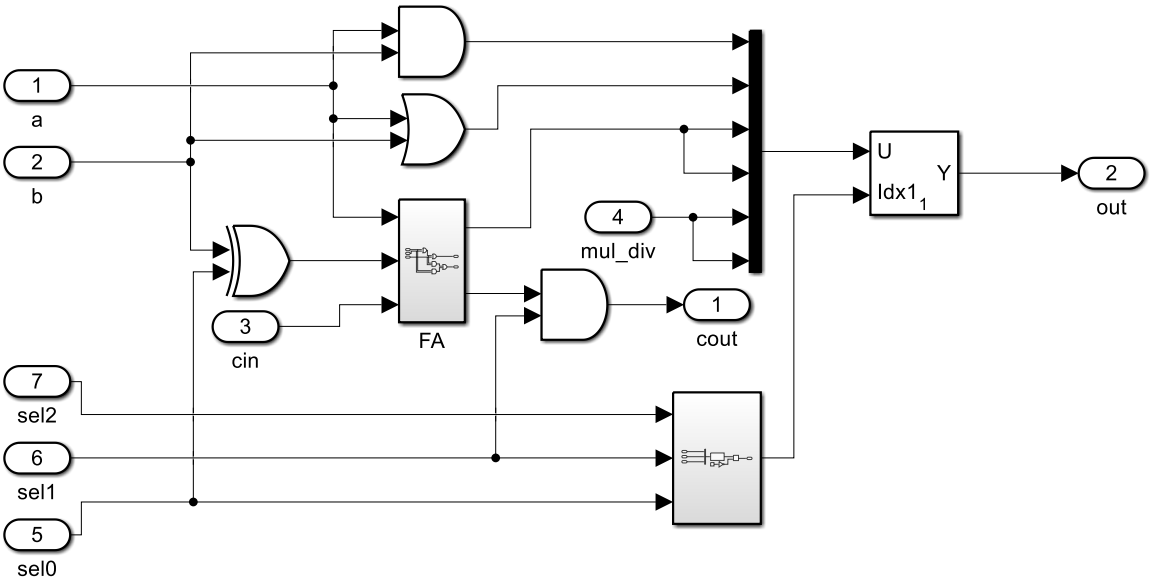
همانطور که می بینید نمای کلی این فایل هم همانند قبلی است.

ساختار این فایل به صورت زیر است:



حال به بررسی این بلوک ها می پردازیم:

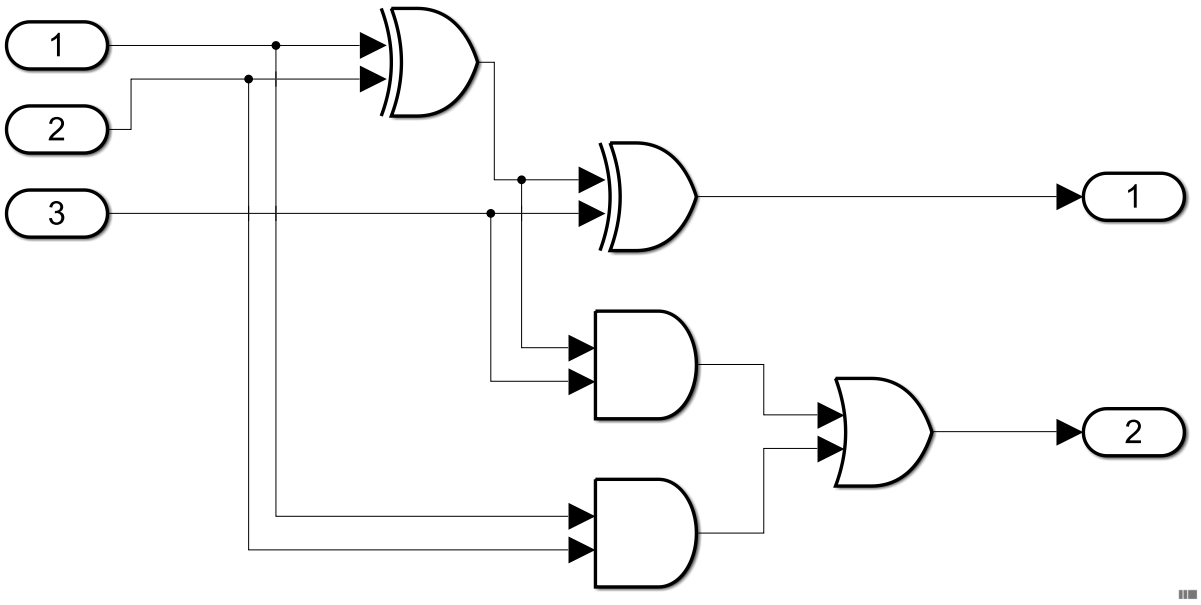
بلوک ALU



باز هم دقیقا شکل فایل پیشین می باشد. البته با توجه به اینکه اینبار خروجی به صورت Boolean است نه double پس از بلوک های cast to double صرف نظر کردیم.

بلوک Full Adder

بیرون این فول ادر همانند فایل پیش است اما درونی متفاوت دارد.



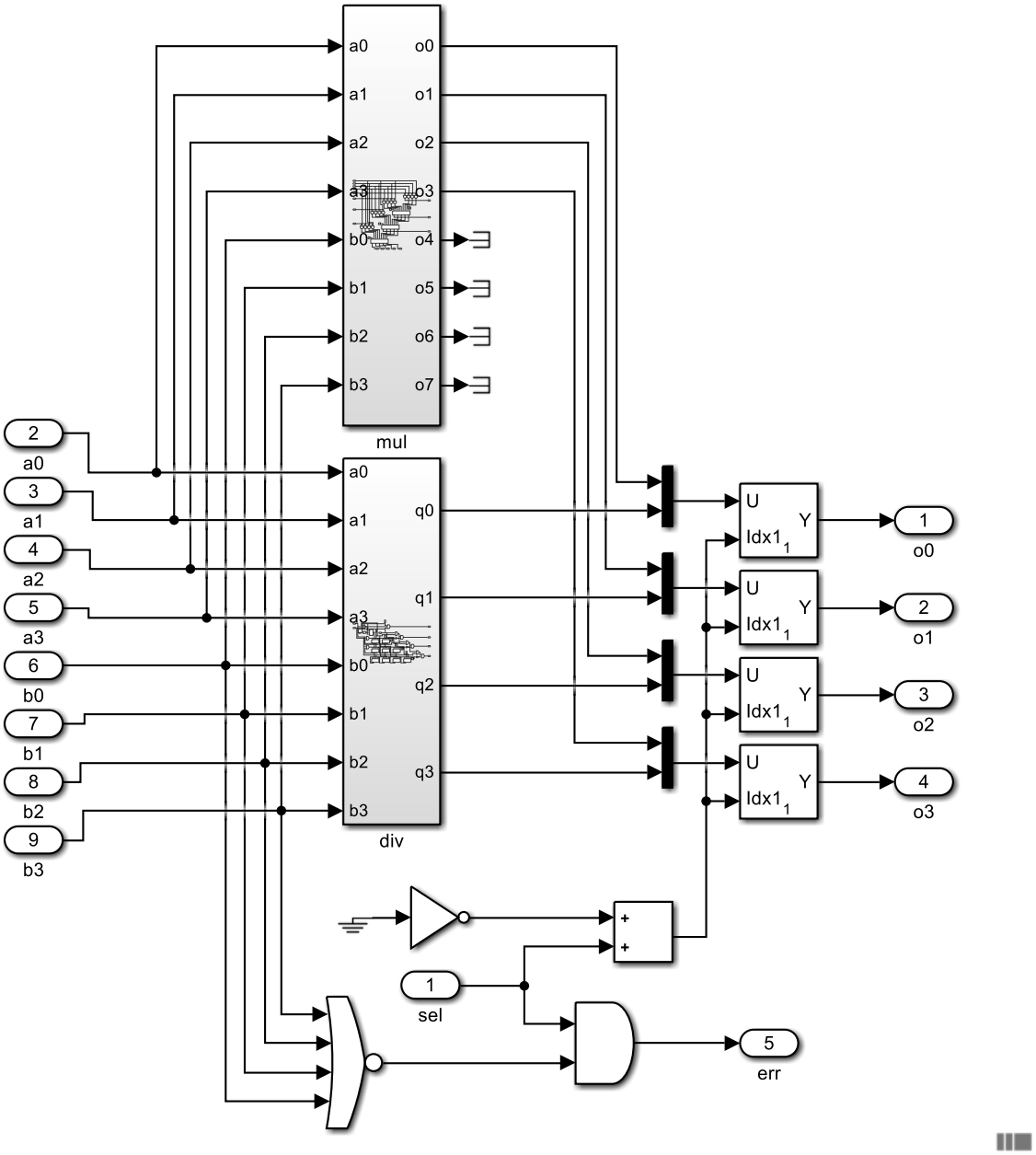
قبلا با truth table فول ادر آشنا شدیم و نیازی به توضیح اضافی نیست.

بلوک mul_div

در این بلوک باز هم ما برای جدا سازی ضرب و تقسیم از مولتی پلکسر استفاده کرده ایم.

در اینجا دو بلوک اصلی mul برای ضرب و div برای تقسیم داریم. ورودی های هر دو ۲ عدد ۴ بیتی می باشد. خروجی ضرب ۸ بیت است که طبق مطالب گفته شده تنها ۴ بیت کم ارزش آن برای ما مهم است و در تقسیم هم ۴ بیت خروجی داریم. این دو ۴ بیت خروجی با کمک مولتی پلکسر های آماده از هم جدا می شوند.

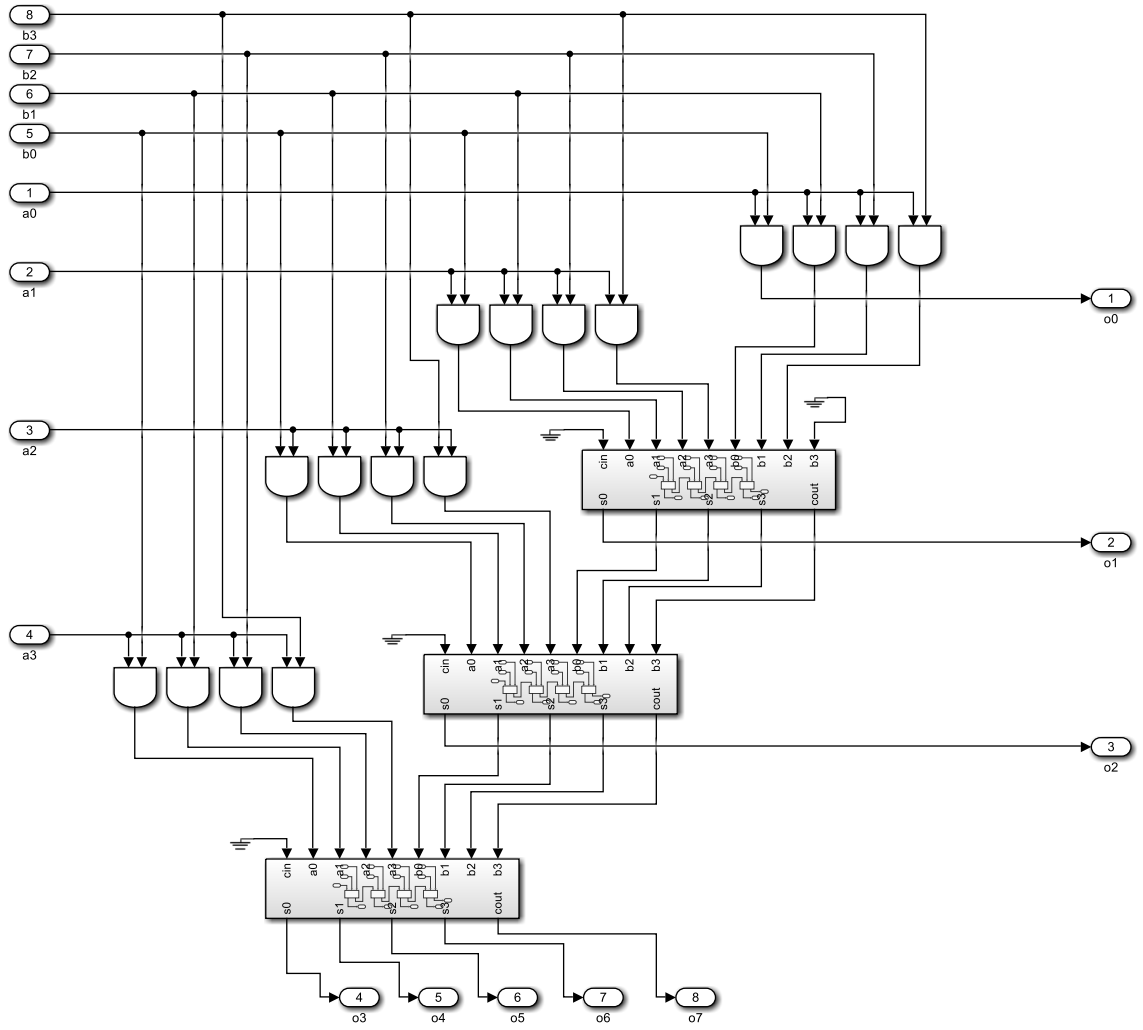
میتوانید نمایی از این بلوک را مشاهده کنید:



در اینجا هم یک بیت برای گزارش ارور داریم که همانند قبلی است و نیازی به توضیح نیست.

بلوک mul

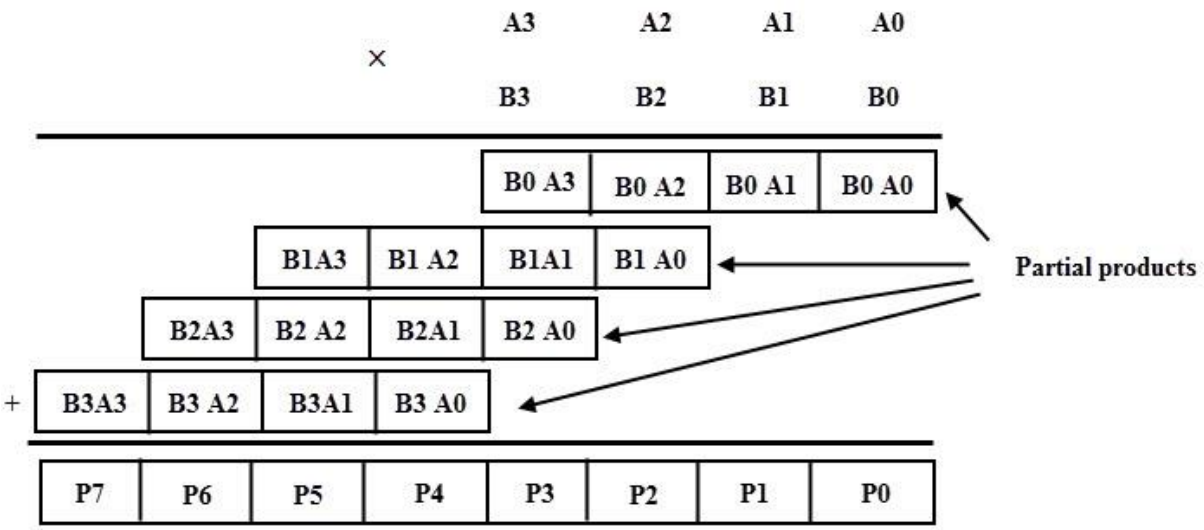
برای ساخت این بلوک از رویه استاندارد ضرب بهره برده شده است.



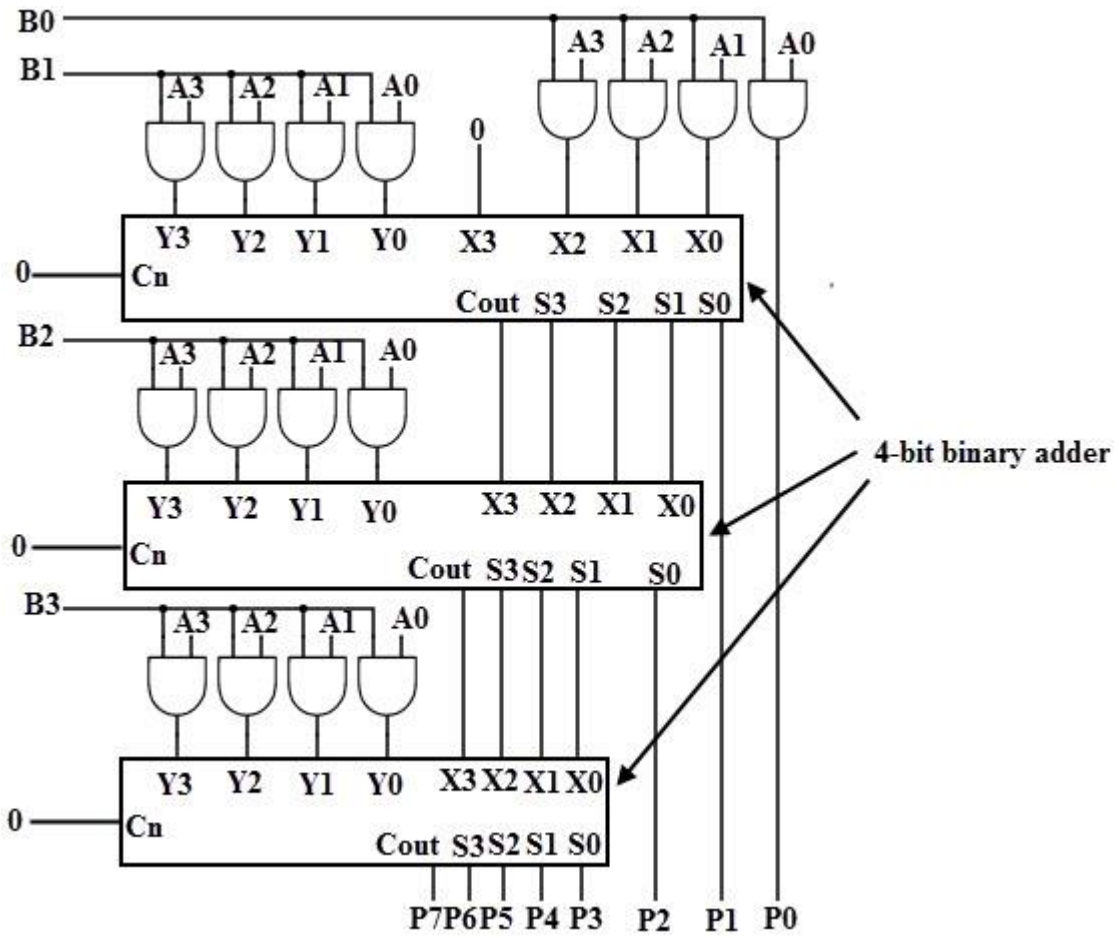
ضرب بالا از قانون زیر پیروی می کند:

1 0 1 0	→	Multiplicand
× 1 0 1 1	→	Multiplier
1 0 1 0	→	Partial product 1
1 0 1 0	→	Partial product 2
0 0 0 0	→	Partial product 3
1 0 1 0	→	Partial product 4
1 1 0 1 1 1 0		

که به صورت زیر استفاده کردیم:

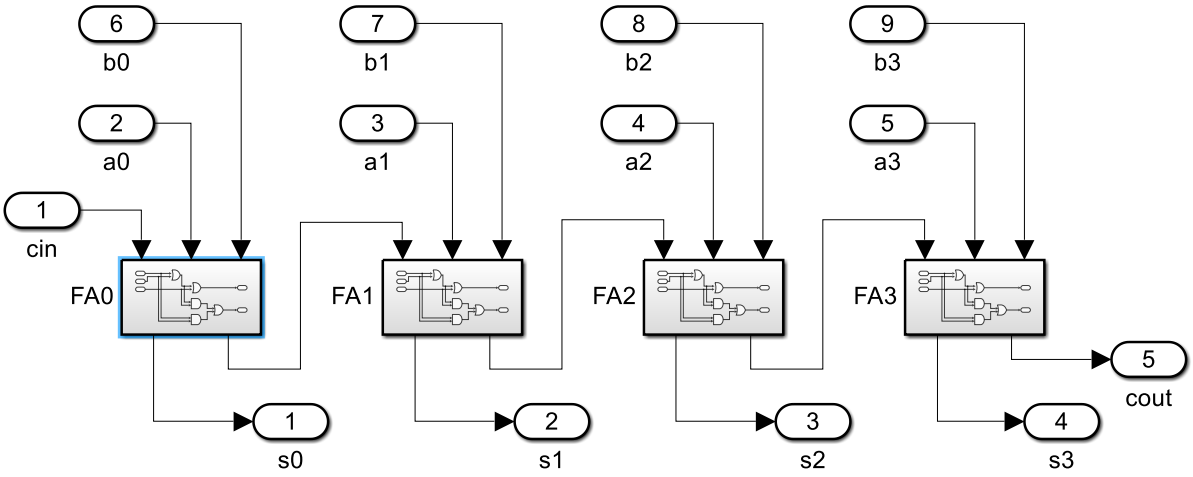


شکل مدار ضرب به شکل زیر است:



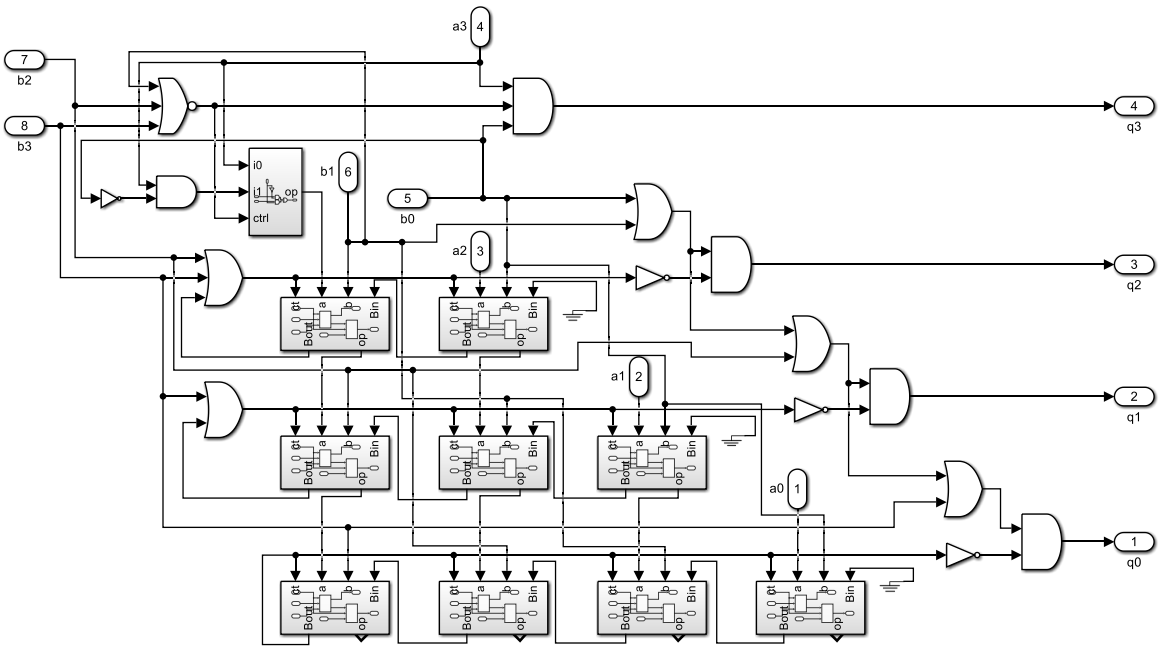
چون ضرب دو عدد ۴ بیتی است لذا به 4*4 گیت and و 4-1 جمع زننده 4 بیتی نیازداریم.

جمع زننده ۴ بیتی



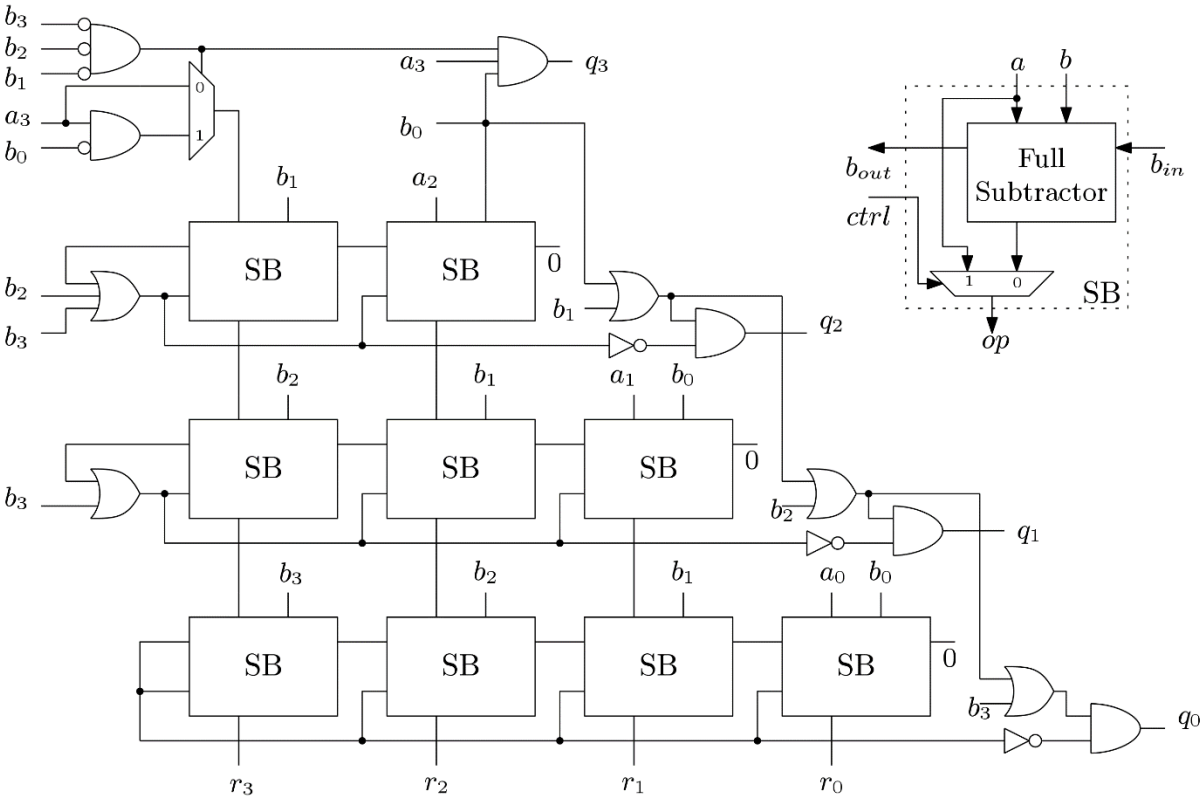
قبلا با فول ادر آشنا شدیم پس نیازی به توضیح نیست.

بلوک div



نمای درون بلوک به شکل بالاست.

این مدار از شکل زیر گرفته شده است:



طبق شکل بالا رابطه زیر همواره برقرار است:

A = B * Q + R

که در آن r باقی مانده است و q هم حاصل تقسیم است. البته ما در این پروژه به باقی مانده نیازی نداشتیم پس خروجی ای از آن ها نمی گیریم.

توضیحات سایت منبع در مورد تقسیم این چنین است:

Our objective is to divide A by B where both A and B can be positive or negative. The division operation can be expressed as by the following equation by considering 4 bit data width.

where Q is the quotient and R is the remainder. The division operation is carried out by subsequent stages where each stage determines a quotient bit. Thus it is called an array divider.
In the first stage to find the bit Q3, B is left shifted by 3 bits and subtracted from A. The quotient bit Q3 is 1 if the difference is zero or positive. If the difference is negative, Q3 is 0 and the value of A is passed to the next stage. This technique is simply the restoring division discussed in the previous post. The division operation is explained by an

example where A = 1100 and B = 0010. The unsigned divider for 4-bit data width is shown below.

$$\begin{array}{r} 0001100 \\ -\ 0010000 \\ \hline 1\ 1111100 \end{array} \quad \begin{array}{r} 0001100 \\ -\ 0001000 \\ \hline 0\ 0000100 \end{array} \quad \begin{array}{r} 0000100 \\ -\ 0000100 \\ \hline 0\ 0000000 \end{array} \quad \begin{array}{r} 0000000 \\ -\ 0000010 \\ \hline 1\ 1111110 \end{array}$$

$b_{out} = 1, Q_3 = 0$ $\quad b_{out} = 0, Q_2 = 1$ $\quad b_{out} = 0, Q_1 = 1$ $\quad b_{out} = 1, Q_0 = 0$

4-bit Unsigned Array Divider

From the above example it is clear that to compute Q3, B0 is subtracted from A3 only when other bits of B are zero. This logic is implemented in the first stage. Similarly to compute Q2, bits B1 and B0 are needed to be subtracted thus two sub blocks are used. The block diagram of the sub block is also given in the above figure. It consists of a full subtractor and a Mux.

The above example is shown for division by unsigned numbers. The signed division can be done easily by adopting this division method. The input signed operands (Two's compliment representation) are needed to converted to unsigned operands and at the output the quotient and residual are also needed to be converted to signed operands depending on the sign of input operands.

The above divider produces a quotient (Q) and remainder (R) but the practical use of divider involve generation of fractional bits. A Verilog code of a complete pipelined 18-bit signed divider is given in this bit. 8-bits are used for integer part and 10-bits are used for fractional part. To generate fractional result, 10 more stages are added to the previous stages. This divider can be modified accordingly to requirement of quotient and fractional bits.

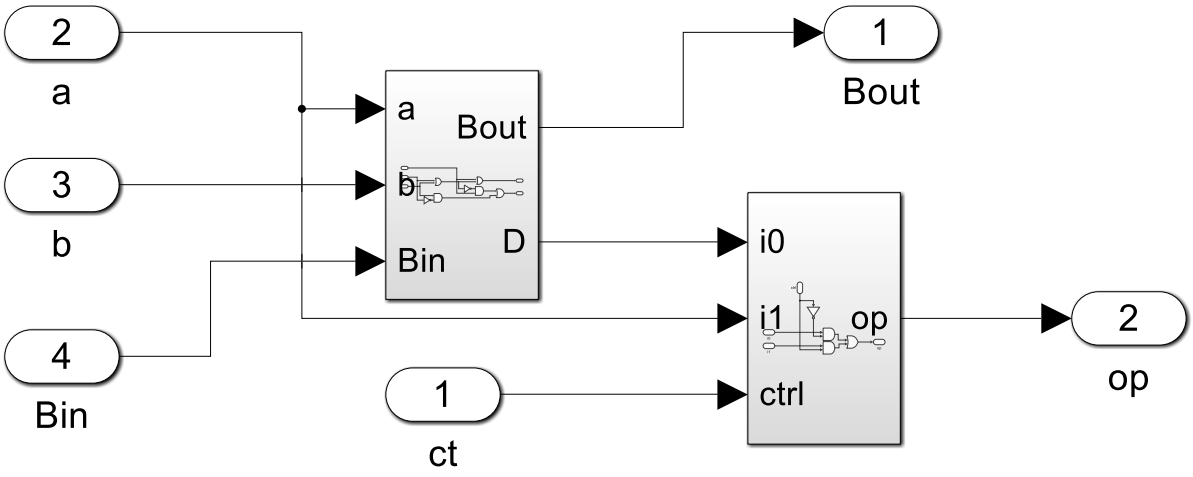
همانطور که میتوان دید ابتدا در این مدار Q3 محاسبه می شود و همینطور به سمت بیت های کم ارزش تر پیش میرویم.

اگر بخواهیم میتوانیم با این مدار نتیجه را به صورت کسری هم ببینیم به شرط اینکه باقی مانده را با شیفت دادن دوباره به ورودی ببریم.

همانطور که میبینید برای یک شدن Q3 باید ۳ بیت با ارزش b ما صفر و بیت کم ارزش یک باشد. همچنین با ارزش ترین بیت a هم یک باشد.

در محاسبه بیت های بعدی از بلوکی تحت عنوان SB استفاده شده که یک تمام تفریق گر را با یک مولتی پلکسر استفاده کرده، اگر پایه کنترل صفر باشد خروجی ما خروجی تمام تفریق گر خواهد بود وگرنه خروجی ورودی اول تمام تفریق گر خواهد بود. یک کری ورودی و خروجی هم داریم که مستقیما به کری ورودی و خروجی تمام تفریق گر متصل می شوند.

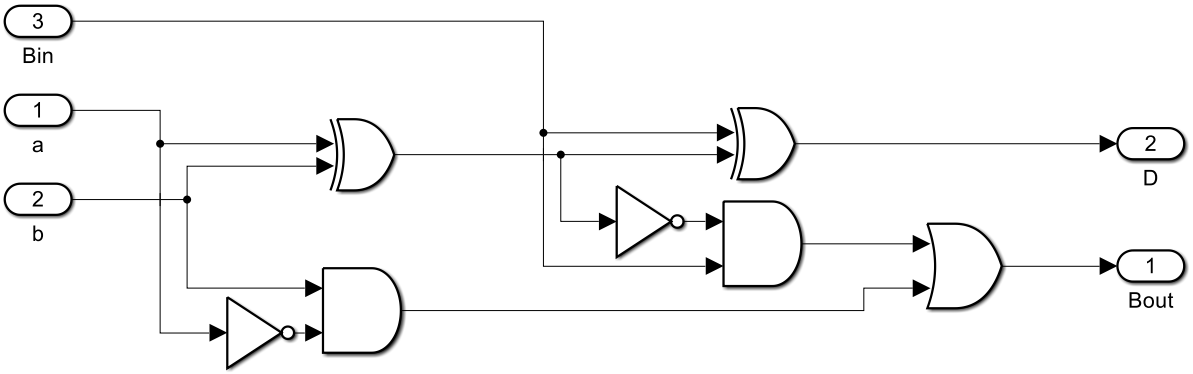
بلوک SB



شکل بالا بلوک SB می باشد.

در واقع وقتی کنترل ما ۱ باشد داده هایی که به عنوان بیت a وارد می شوند به عنوان خروجی به بیرون می روند.

تمام تفریق گر

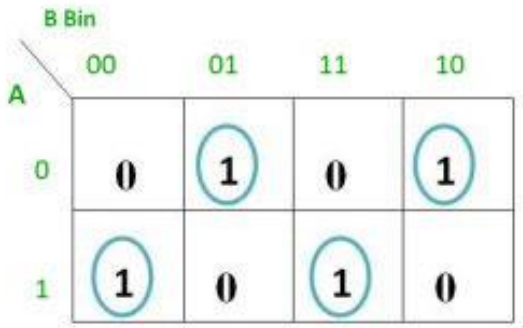


تمام تفریق گر ما به شکل بالاست.

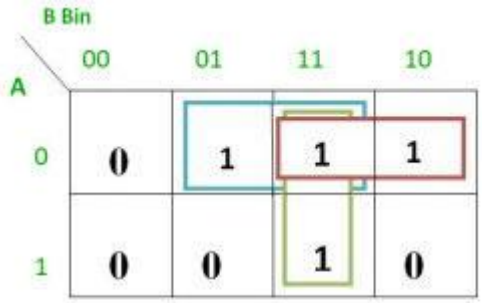
جدول درستی مدار بالا به شکل زیر است:

INPUT			OUTPUT	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

برای بدست آوردن مدار به شکل زیر ابتدا جدول کارنو را رفتیم . . .



D=A'B'Bin + AB'Bin' + A'BBin' + ABBin



Bout=A'Bin + A'B + BBin

همچنین ساده سازی های زیر را هم انجام می دهیم:

Logical expression for difference –

```
D    = A'B'Bin + A'BBin' + AB'Bin' + ABBin
     = Bin(A'B' + AB)  + Bin'(AB' + A'B)
     = Bin( A XNOR B) + Bin'(A XOR B)
     = Bin (A XOR B)'  +  Bin'(A XOR B)
     = Bin XOR (A XOR B)
     = (A XOR B) XOR Bin
```

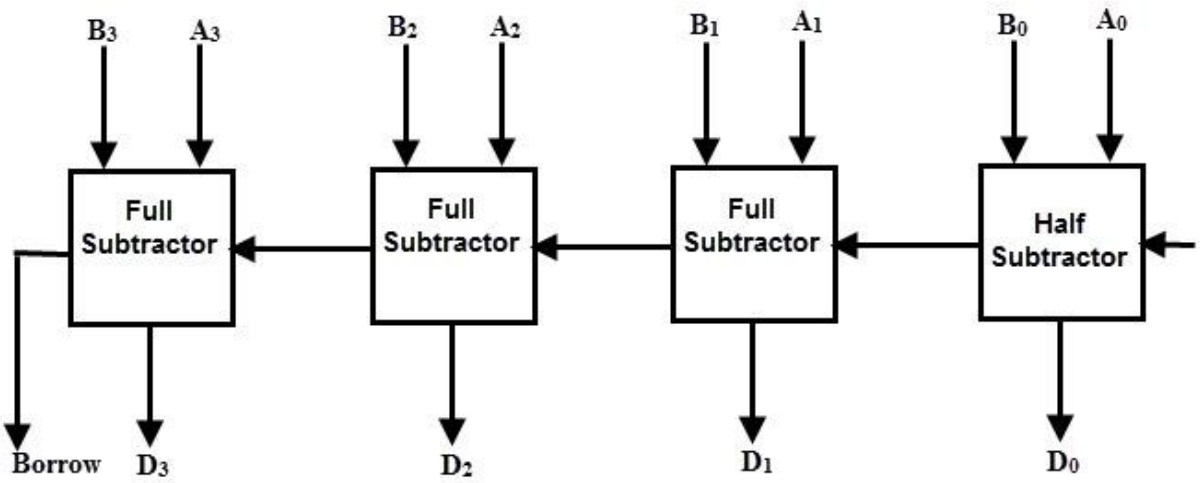
Logical expression for borrow –

```
Bout = A'B'Bin + A'BBin' + A'BBin + ABBin
      = A'B'Bin +A'BBin' + A'BBin + A'BBin + A'BBin + ABBin
      = A'Bin(B + B') + A'B(Bin + Bin') + BBin(A + A')
```

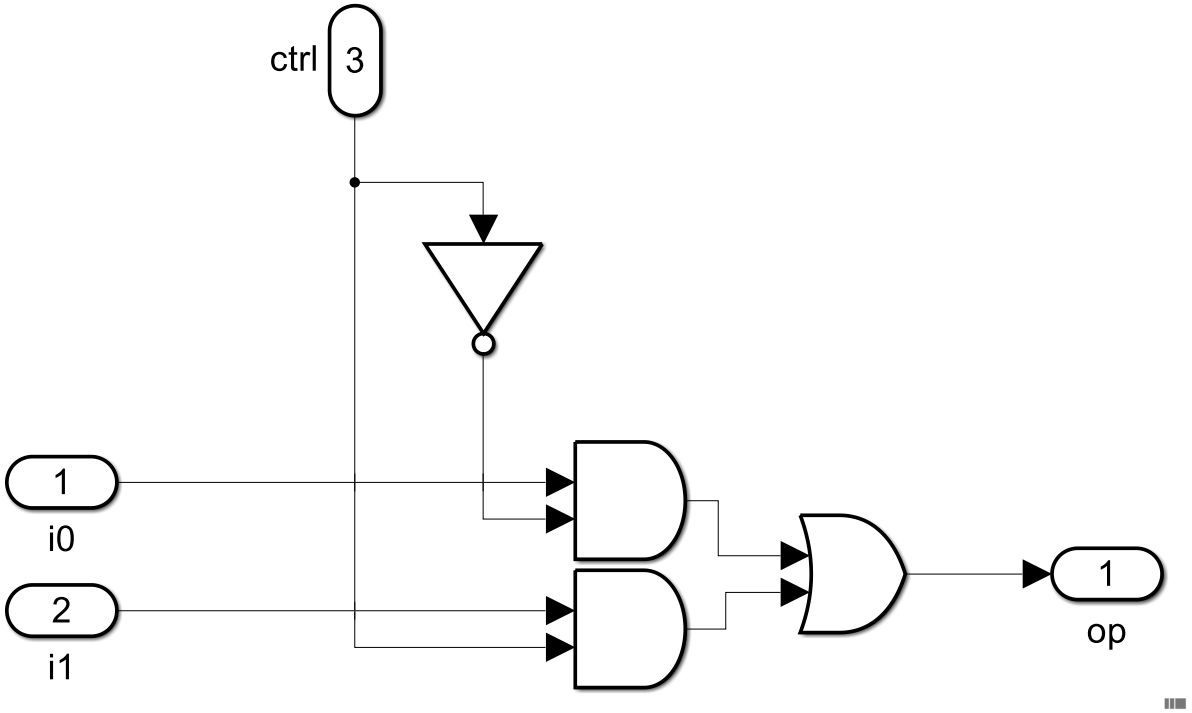
```
      = A’Bin + A’B + BBin

OR

Bout = A’B’Bin + A’BBin’ + A’BBin + ABBin
      = Bin(AB + A’B’) + A’B(Bin + Bin’)
      = Bin( A XNOR B) + A’B
      = Bin (A XOR B)’ + A’B
```

همچنین تمام تفریق گرهای متوالی در جا هایی مثل تمام تفریق گرهای چند بیتی کاربرد دارند.

در پایین یک تمام تفریق گر ۴ بیتی داریم:



بلوک مولتی پلکسر



مولتی پلکسر مورد استفاده در تفریق گر به شکل بالا است که به علت کم بودن حجم از مولتی پلکسر آماده متلب استفاده نکردیم.

با توضیح این بخش به پایان این فصل میرسیم.

برای دیدن نمونه های این فایل میتوانید به فصل نمونه ورودی و خروجی و بخش نمونه های پروژه با گیت های دستی و دارای بلوک تقسیم ضرب بروید.

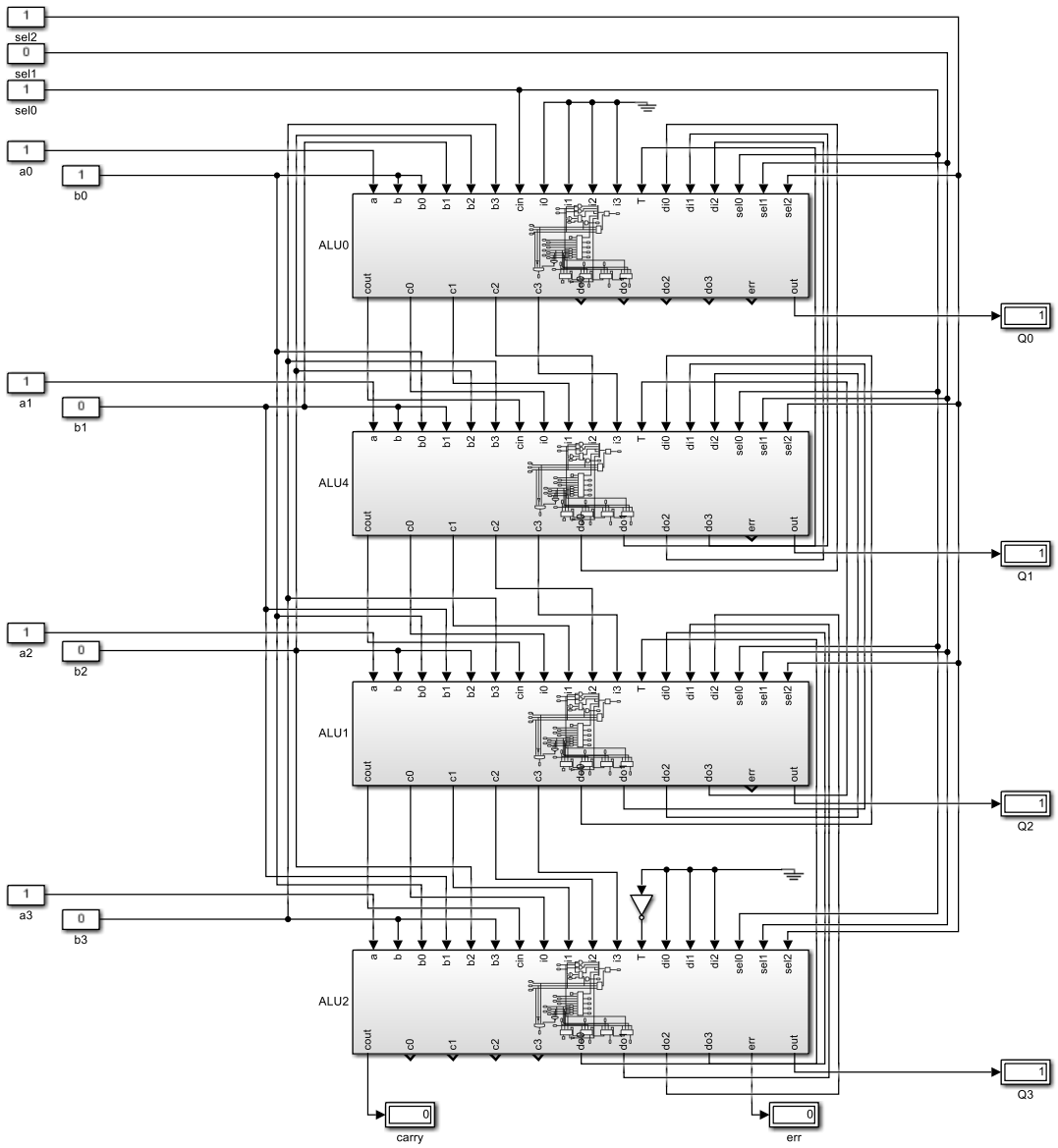
فصل پنجم

پروژه تماما بلوکی

در این فصل قصد داشتیم تا با بلوک هایی که اکثرا دست ساز هستند پروژه ای با دقیقا 4 بیت ALU بسازیم.

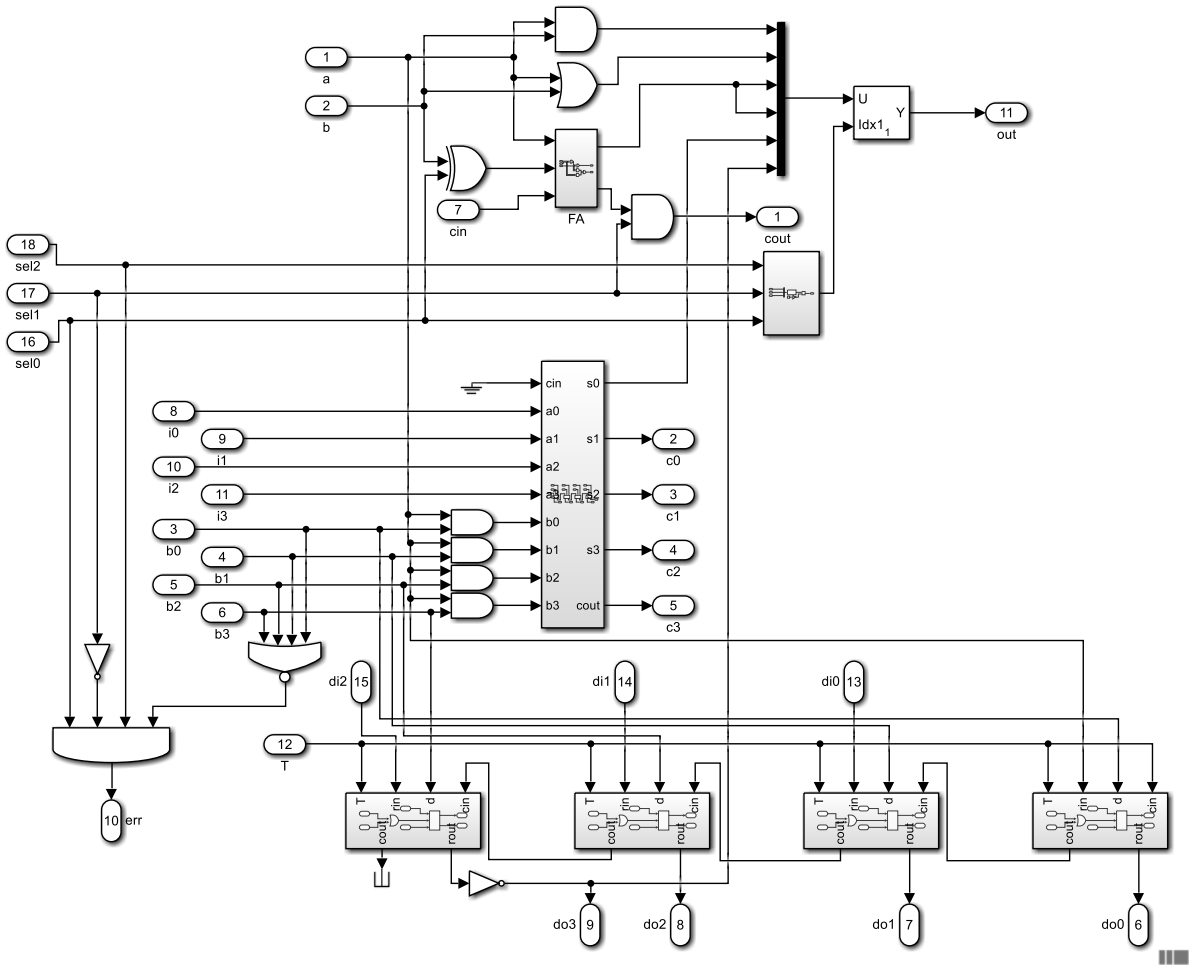
شکل پروژه

نمای کلی این فایل به صورت زیر است:



همانطور که می بینید حجم کری ها در این روش به شدت بیشتر است.

بلوک ALU

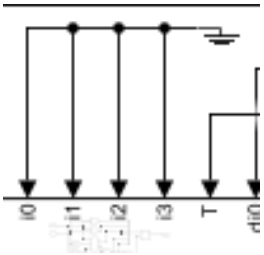


درون این بلوک ها به شکل بالا می باشد. همانطور که می بینید تنها قسمت هایی که برای ضرب و تقسیم است به بلوک ALU که از فصول پیش می شناسیم اضافه شده است.

تشریح عملیات ضرب

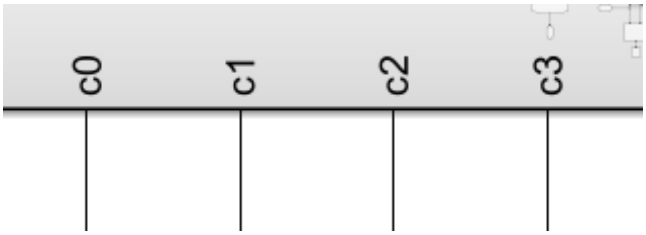
در اینجا مدار ضرب که در فصل گذشته داشتیم را به ۴ قسمت تقسیم کردیم. نکته حائز اهمیت این است که یک جمع زننده ۴ بیتی به مدار اضافه کردیم که برای بدست آوردن بیت اول به کار می رود. برای اینکه بلوک ها کاملا مشابه هم باشند دست به این اقدام زدیم وگرنه لازم نداشتیم.

برای بی اثر کردن این جمع زننده در ALU اول ۴ کری وارد شده مخصوص ضرب را مقدار صفر دادیم.

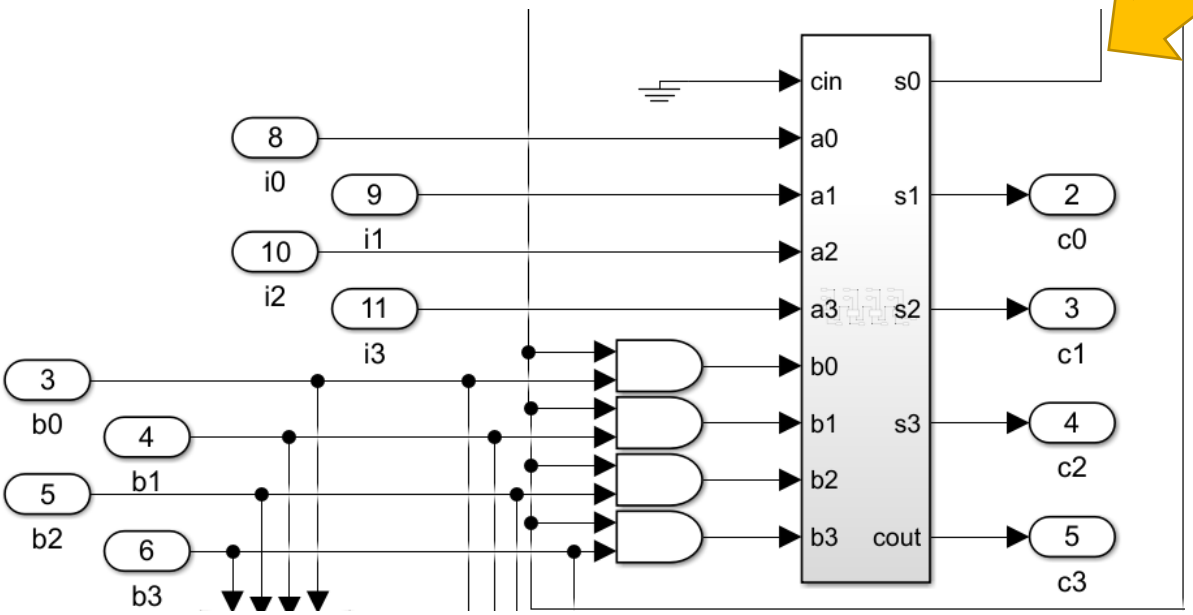


همانطور که میبینید کری های ورودی ضرب این ها هستند.

کری های خروجی هم به شکل زیر است که به جز ALU اول به ورودی های کری ورودی ALU بعدی خود متصل می شوند.



نمای ضرب به شکل زیر است:



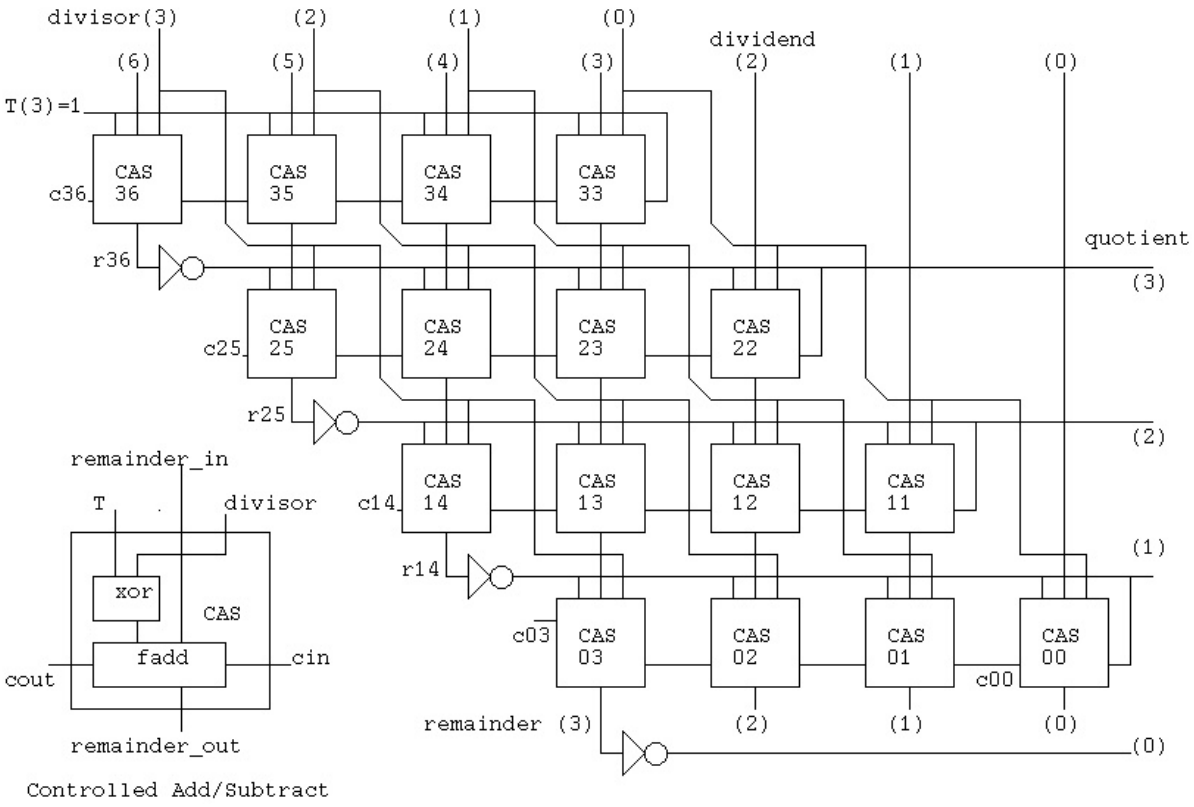
بلوکی که در بالا قرار دارد جمع زننده ۴ بیتی است.

پایه ای که با فلش مشخص شده پایه خروجی ماست.

طبق این شیوه پیاده سازی نیاز است تا تمامی b ها به تمامی ALU ها وارد شود.

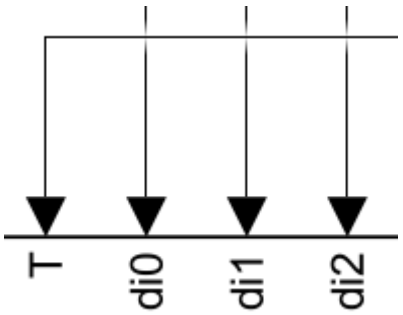
تشریح عملیات تقسیم

برای طراحی این قسمت از مدار زیر استفاده کرده ایم:

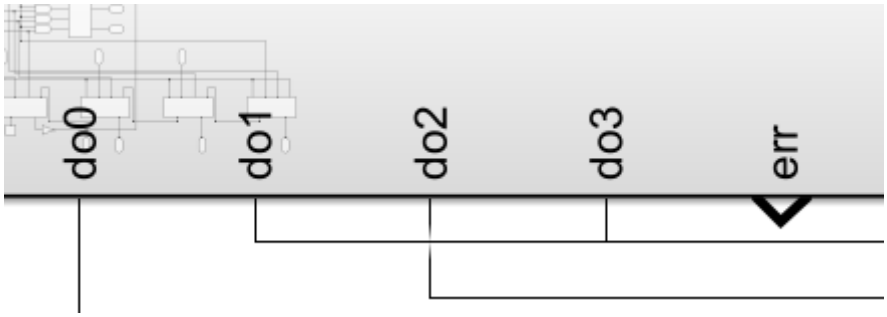


برای بلوکی کردن هم به صورت افقی بلوکی کردیم تا به هر بلوک یک a و ۴ تا b وارد شود. دیگر ورودی ها را به عنوان کری ورودی دریافت می کنیم.

ورودی های مربوط به تقسیم ما به صورت زیر است:

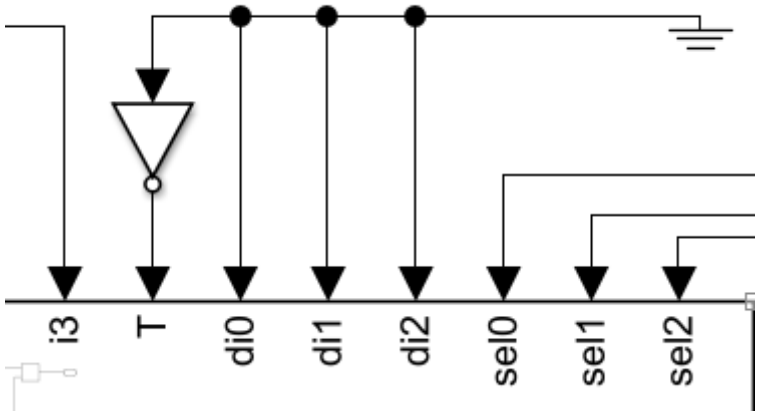


همینطور خروجی های مربوط به تقسیم هم به شکل زیر است:

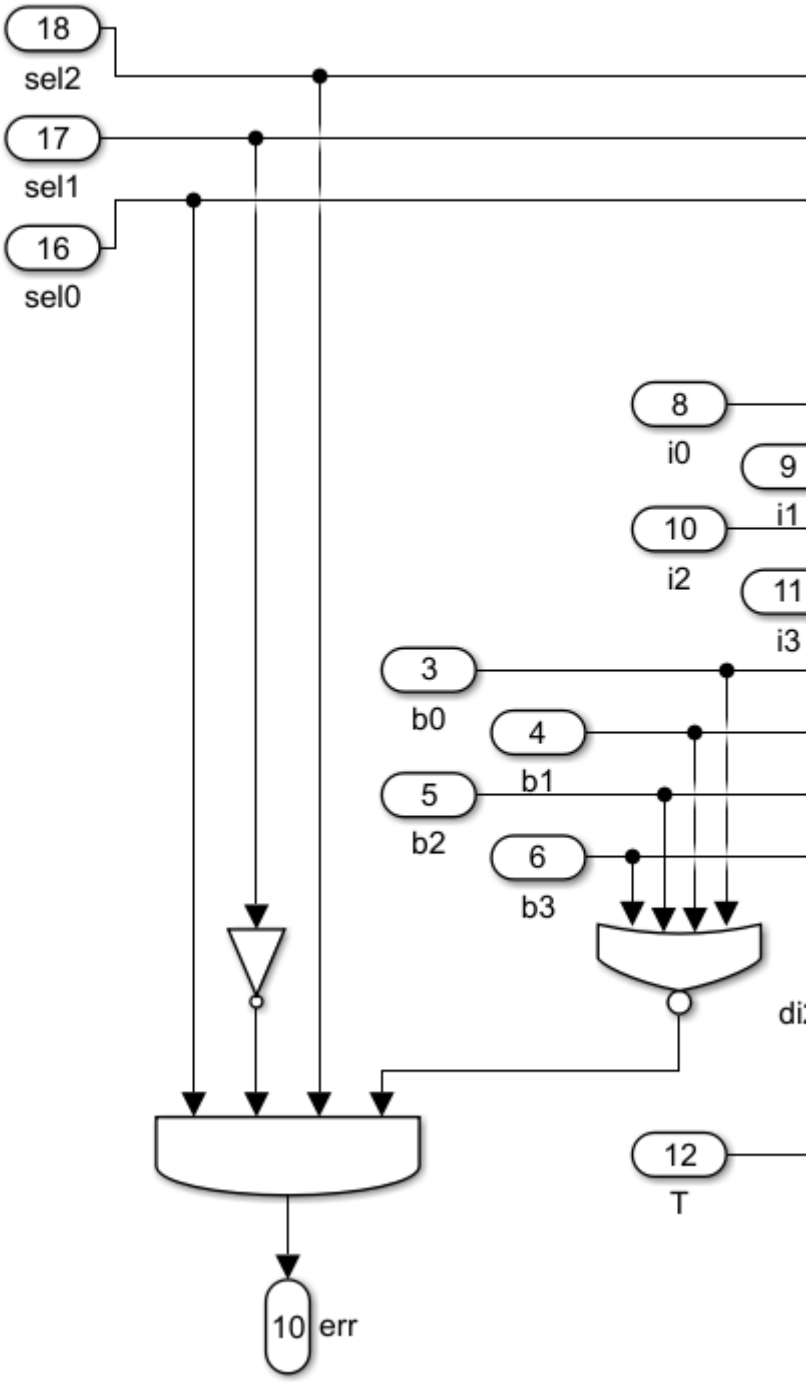


که نوع اتصال آن ها شاید کمی متفاوت با ضرب باشد چرا که ورودی های یک ALU در خصوص تقسیم را باید از خروجی های تقسیم ALU **بعدی** دریافت کنید نه قبلی.

چون همینطور که در تصویر می بینید ابتدا خروجی چهارم محاسبه شده نه اول. پس مقادیر اولیه ورودی ها را هم باید به ALU چهارم بدهیم:

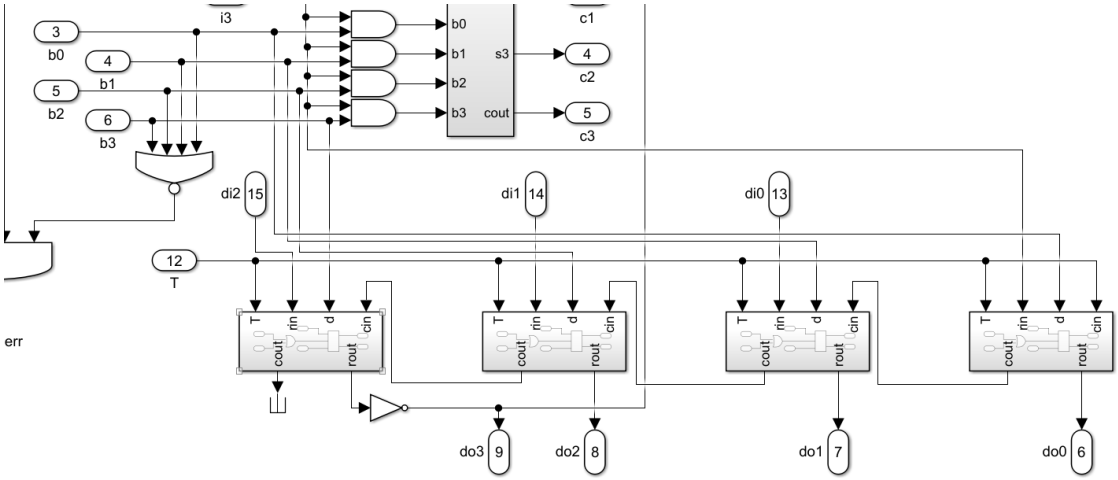


بیت ارور ولی در این جا مستقل است و نیازی به ALU های دیگر ندارد چرا که تمامی b ها دریافت شده اند.



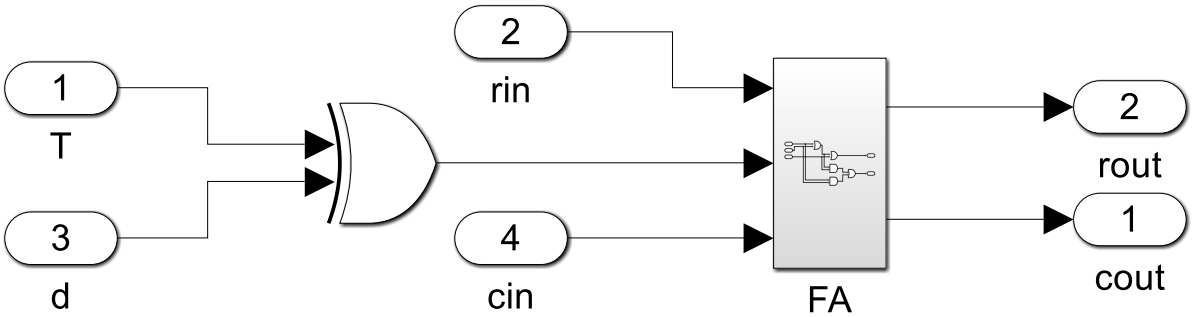
ابتدا شرط صفر بودن تمامی b ها با استفاده از یک گیت nor چهار ورودی چک شده و سپس با استفاده از and هم شرط تقسیم بودن operator code چک می شود. (101)

شکل بلوک تقسیم ما به شکل زیر است:



به ۴ بلوکی که در اینجا کار ما را انجام میدهند به اصطلاح CAS گفته می شود.

بلوک CAS



که در اینجا یک فول ادر داریم و یک xor جهت متمم کردن یا نکردن ورودی d که خود این کار با پایه ی T مشخص می شود.

کلمه CAS مخفف شده controlled Add/subtract است که هسته اصلی مدار تقسیم ماست.

خروجی های کری مربوط به تقسیم در ALU اول که در محاسبه تقسیم آخر است مهم نیستند!

با این توضیح این فصل هم به پایان میرسد. شما میتوانید در فصل نمونه ورودی و خروجی در قسمت نمونه های پروژه تماما بلوکی با گیت های دستی نمونه های حل شده با این فایل را ببینید.

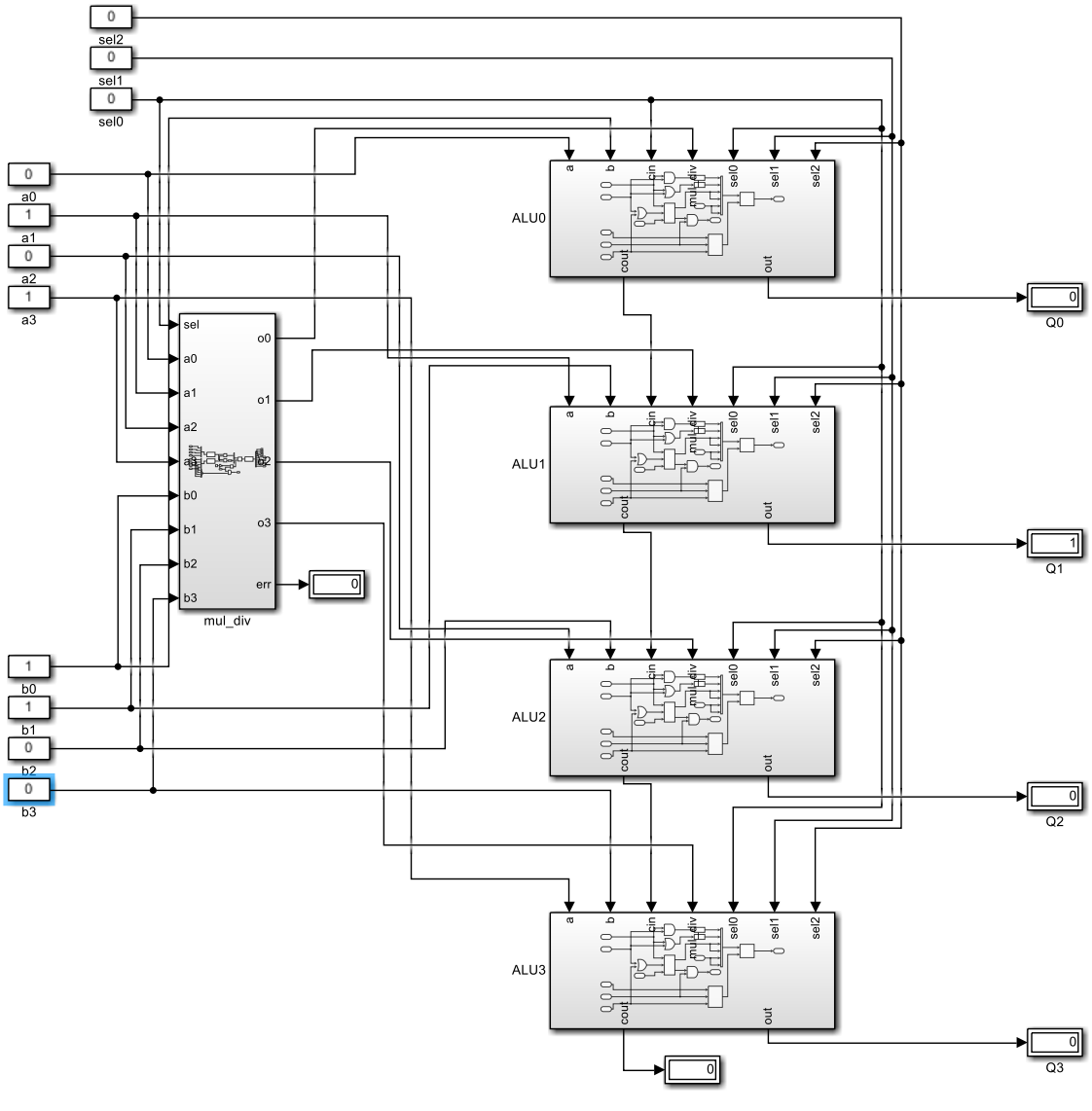
فصل ششم

نمونه ورودی و خروجی

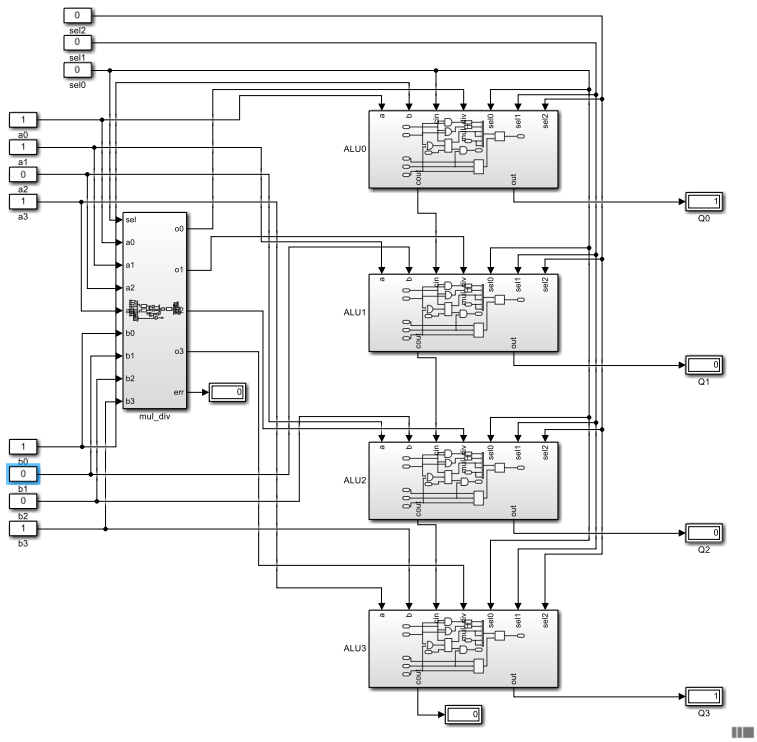
در این فصل به خروجی برخی از نمونه ها میپردازیم:

نمونه های پروژه با گیت های آماده و دارای بلوک تقسیم ضرب

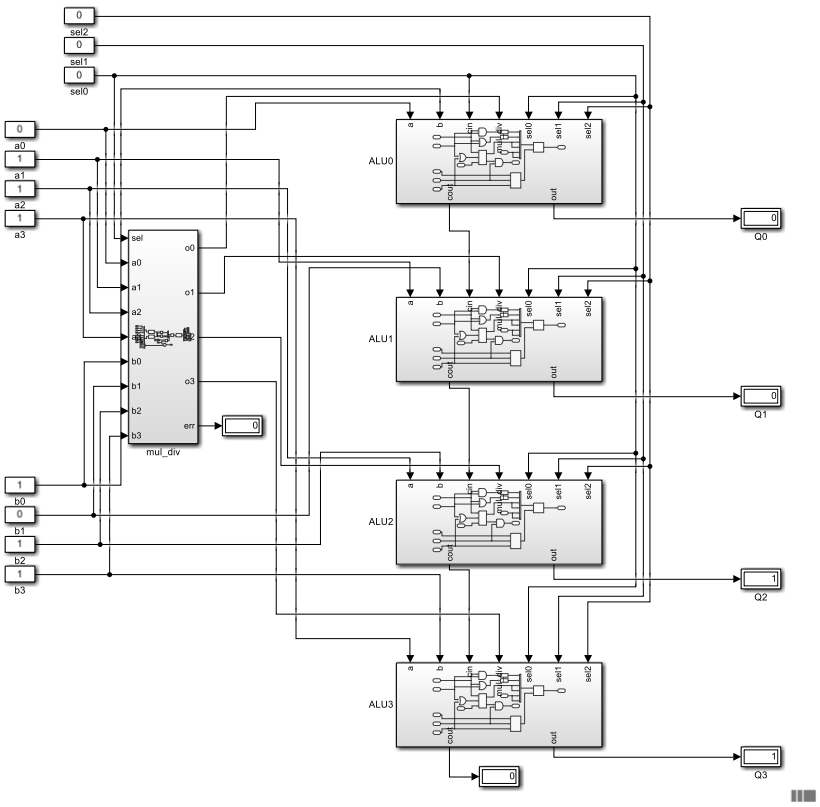
بررسی AND



ورودی a برابر 1010 و b برابر 0011 است که حاصل and آنها 0010 است.

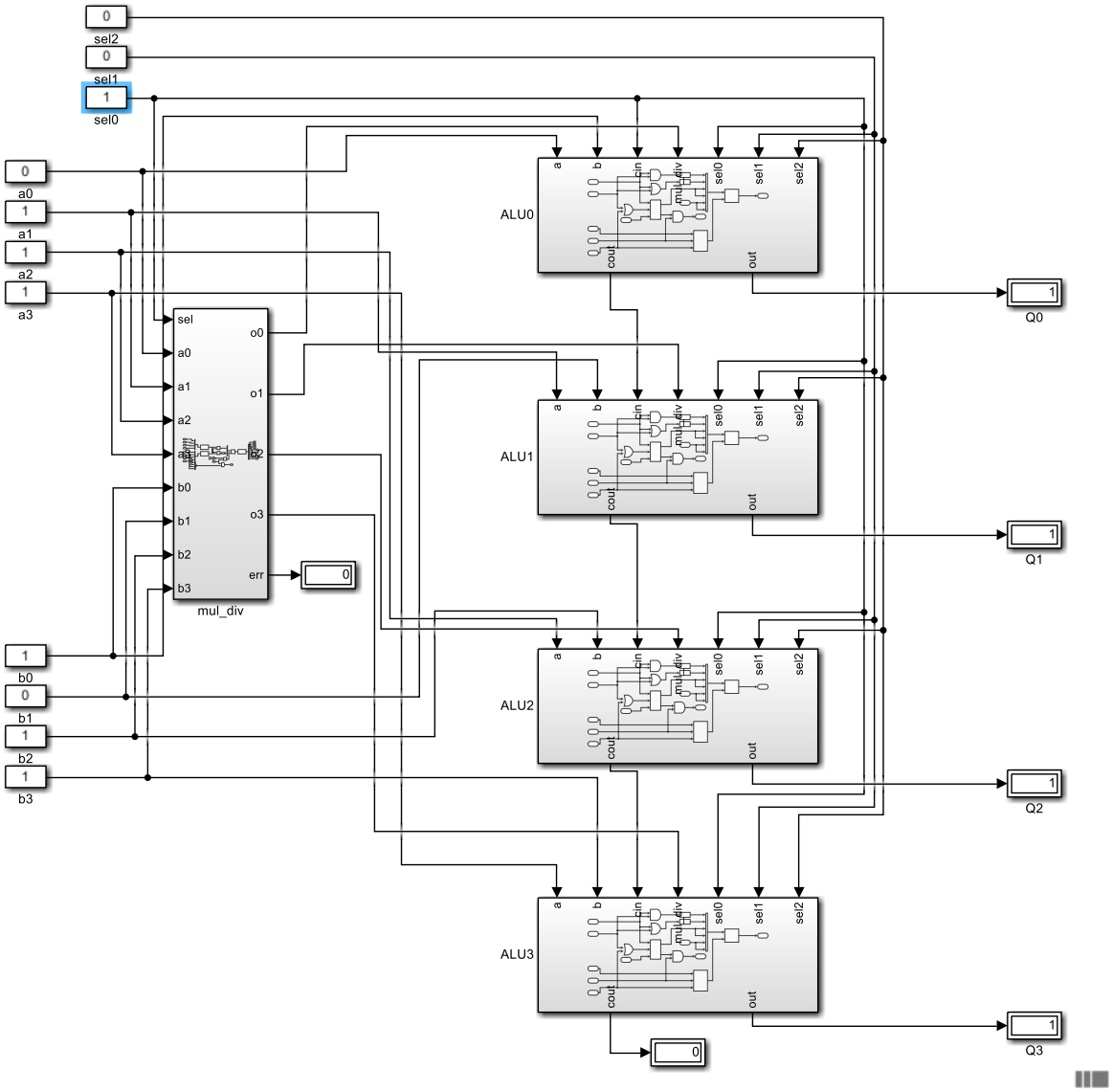


1011 and 1001 = 1001

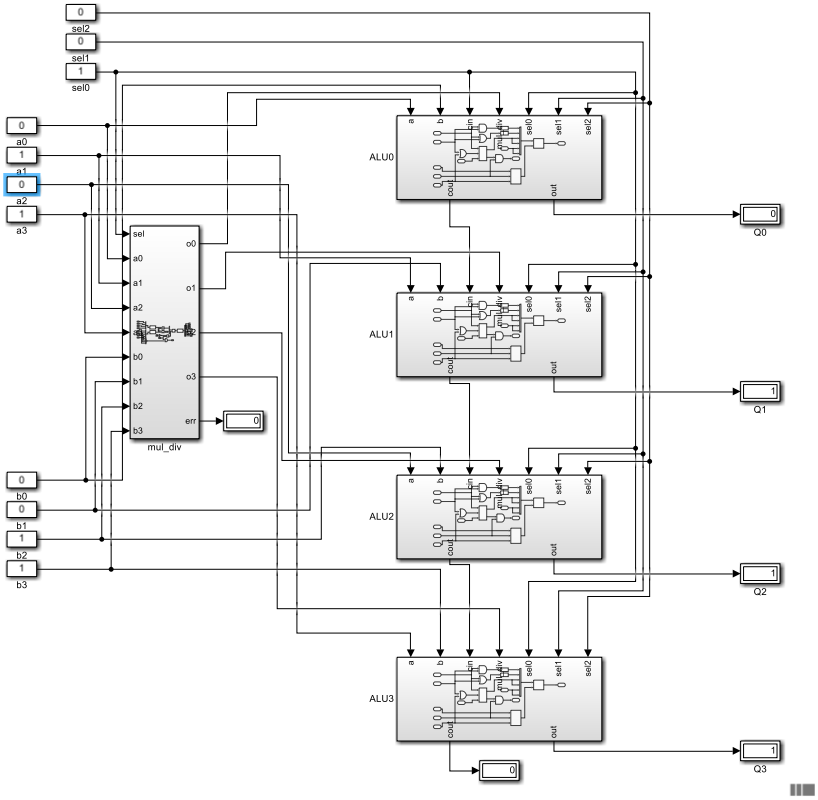


1110 and 1101 = 1100

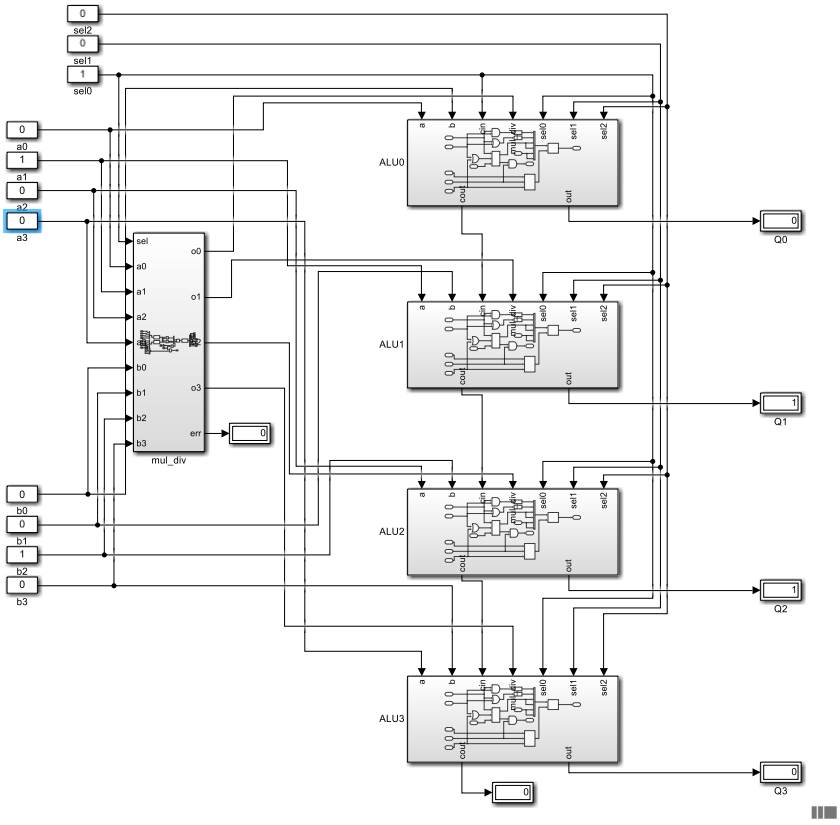
بررسی OR



1110 or 1101 = 1111

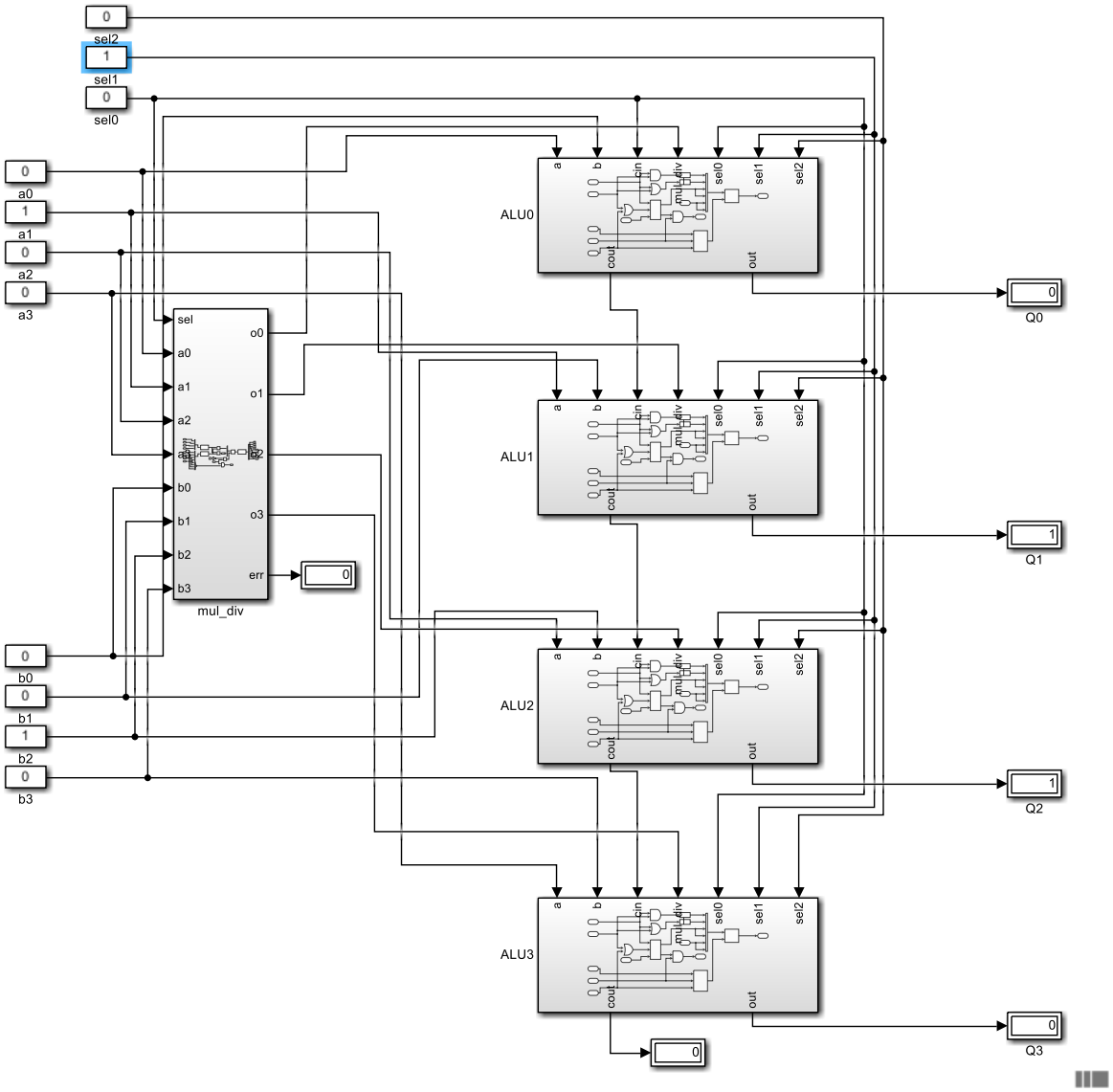


1010 or 1100 = 1110



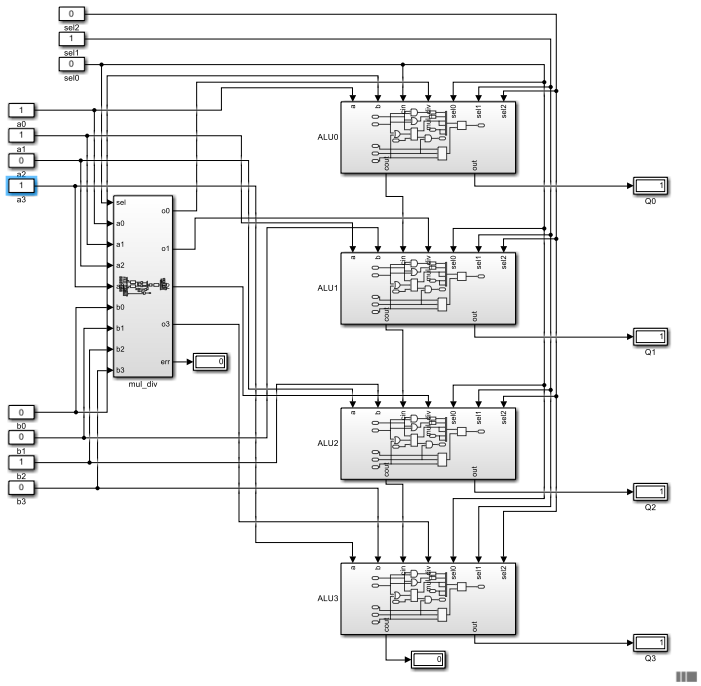
0010 or 0100 = 0110

بررسی جمع



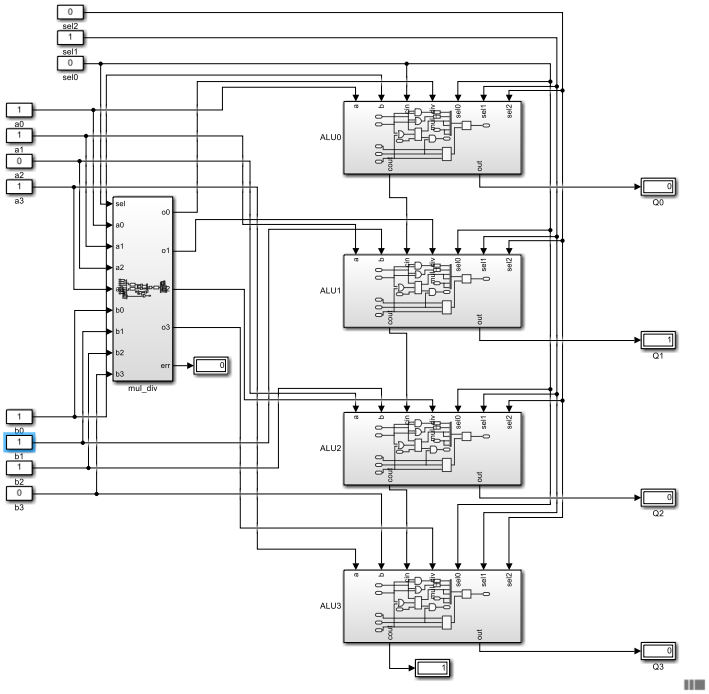
0010 + 0100 = (0)0110

2 + 4 = 6



1011 + 0100 = (0)1111

11 + 4 = 15

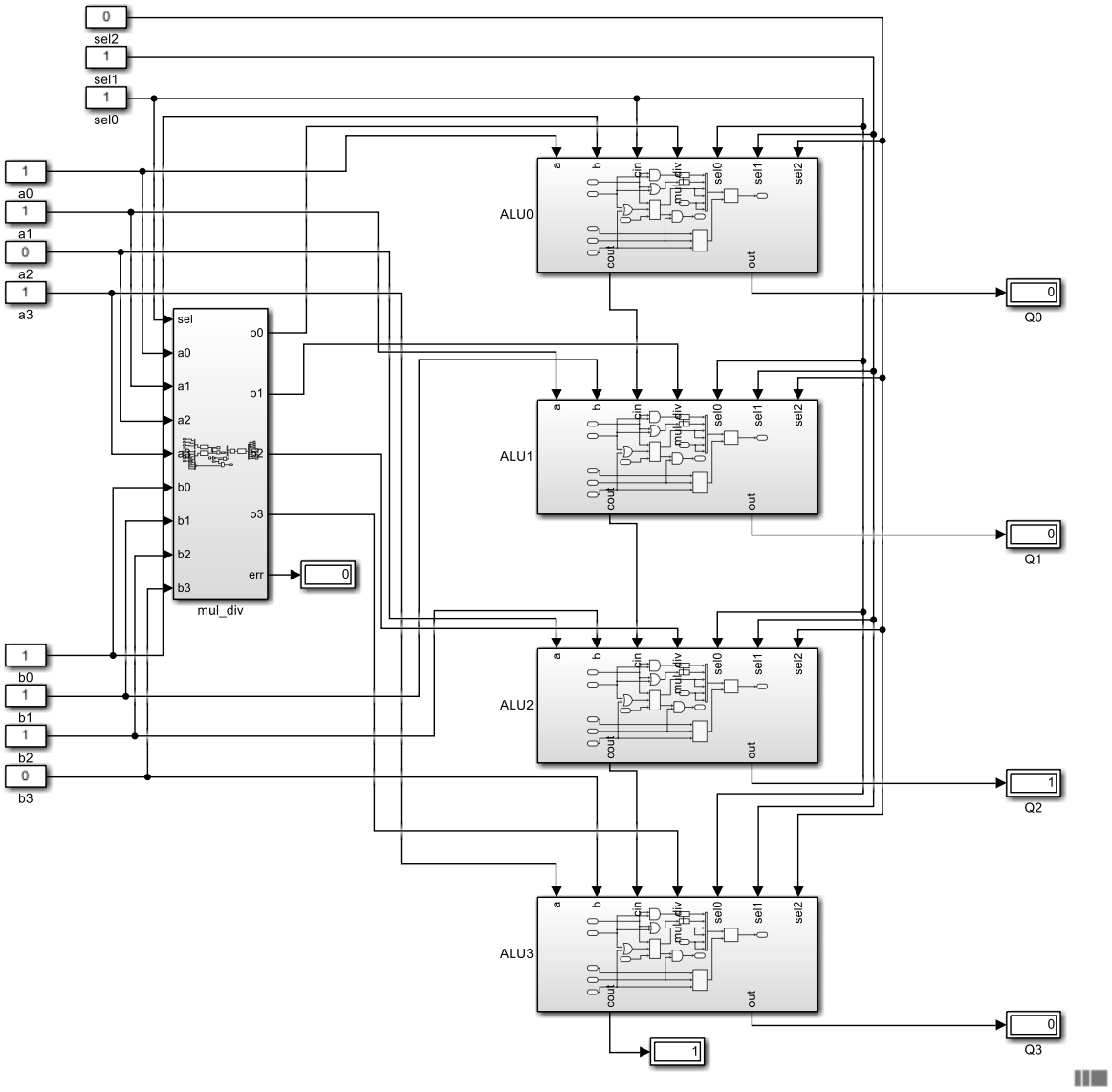


1011 + 0111 = (1)0010

11 + 7 = 18

در اینجا از خروجی کری هم استفاده شده . . .

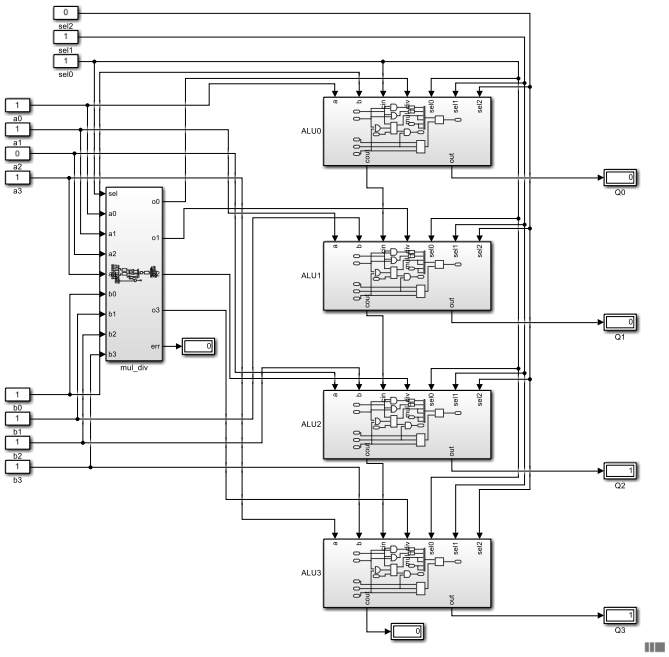
بررسی تفریق



1011 – 0111 = 0100

11 – 7 = 4

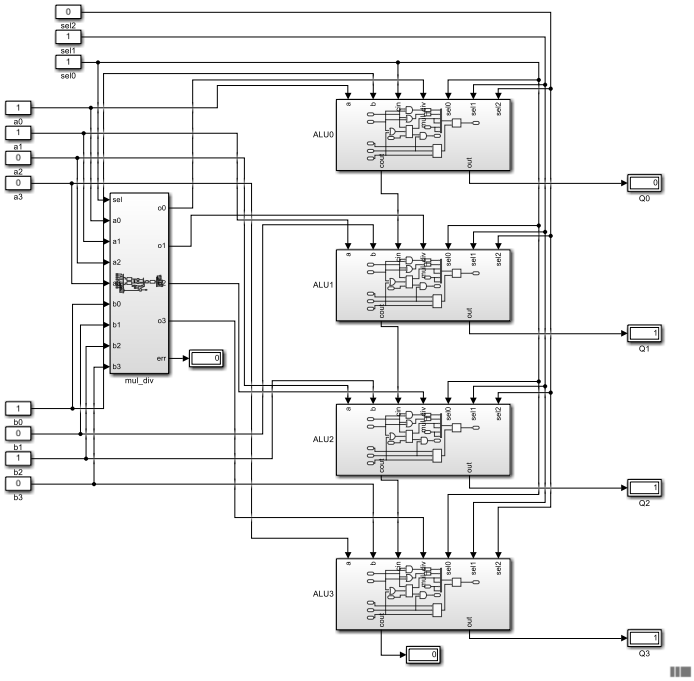
بیت کری در اینجا اضافی است.



1011 – 1111 = 1100

11 – 15 = -4

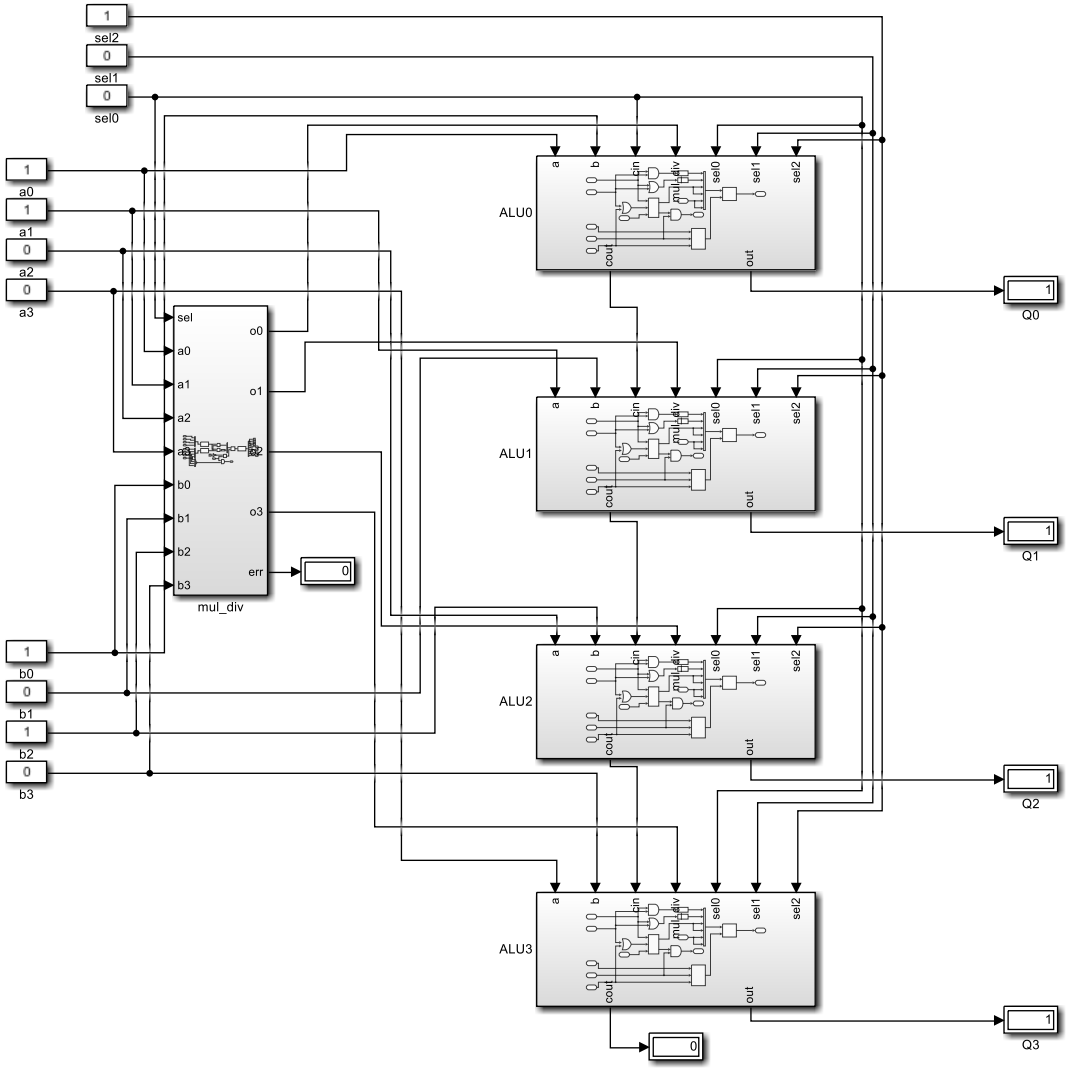
خروجی در اینجا بر مبنای متمم دو نوشته شده است.



0011 – 0101 = 1110

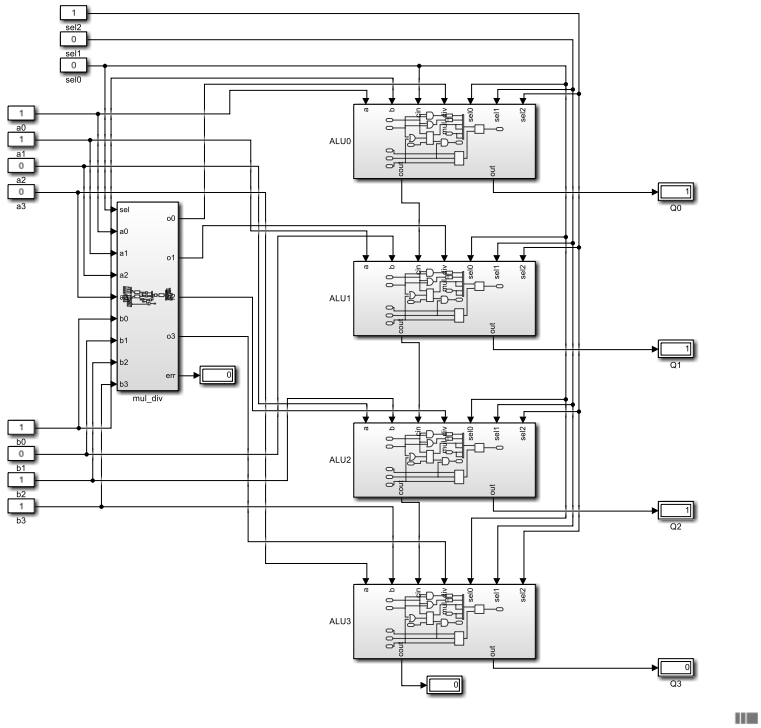
3 – 5 = -2

بررسی ضرب



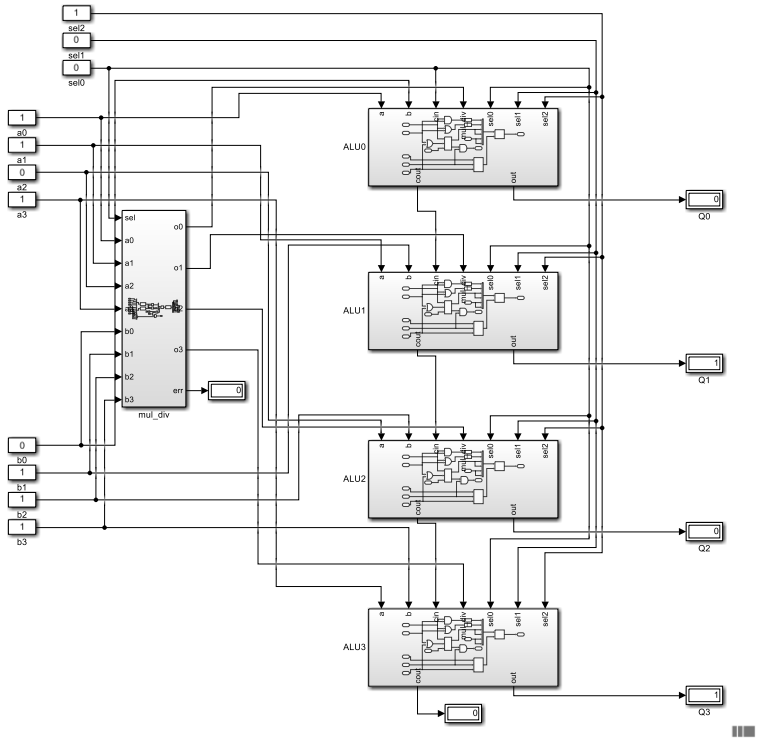
0011 * 0101 = (0000)1111

3 * 5 = 15



0011 * 1101 = (0010)0111

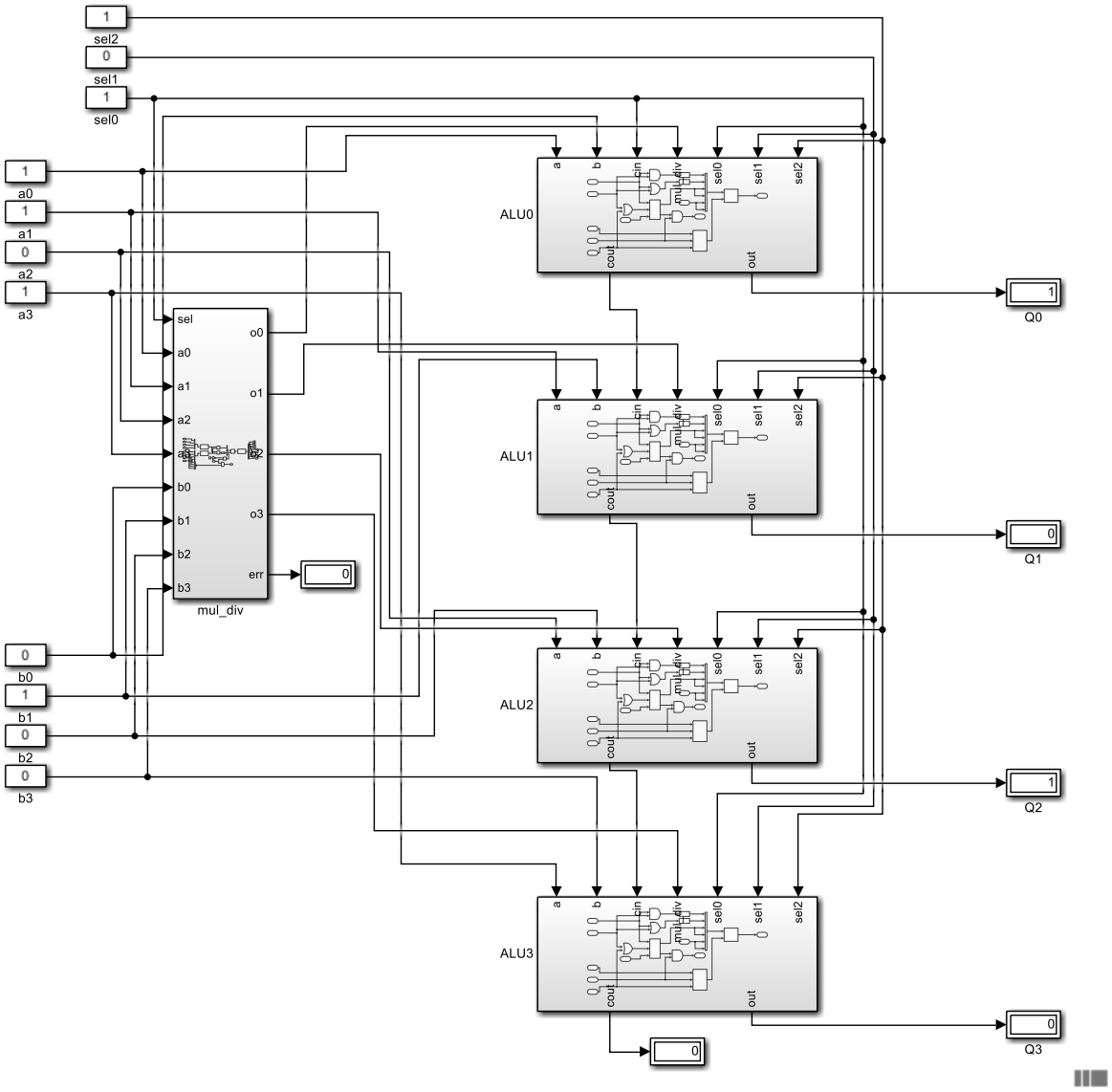
3 * 13 = 39



1011 * 1110 = (1001)1010

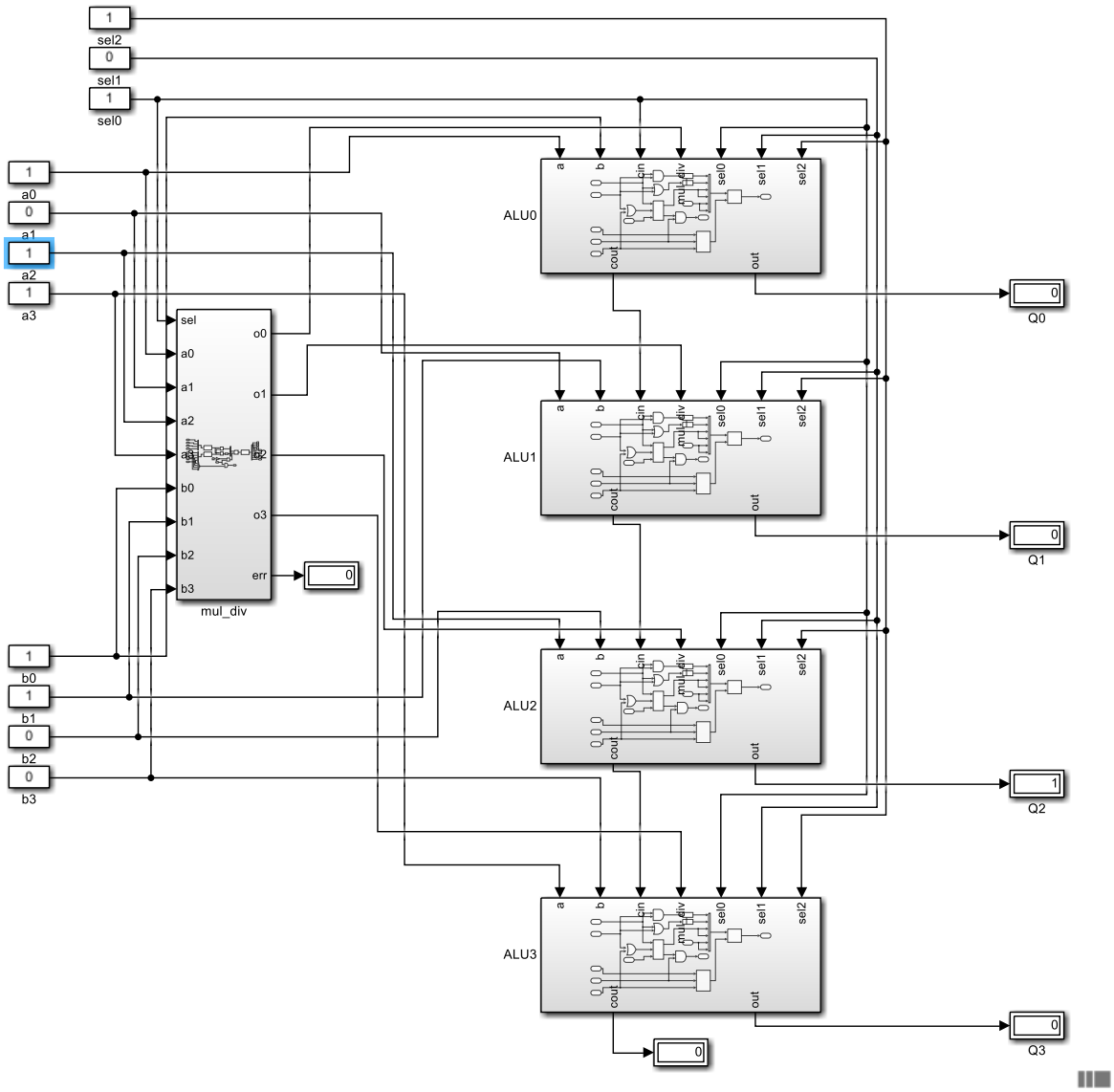
11 * 14 = 154

بررسی تقسیم



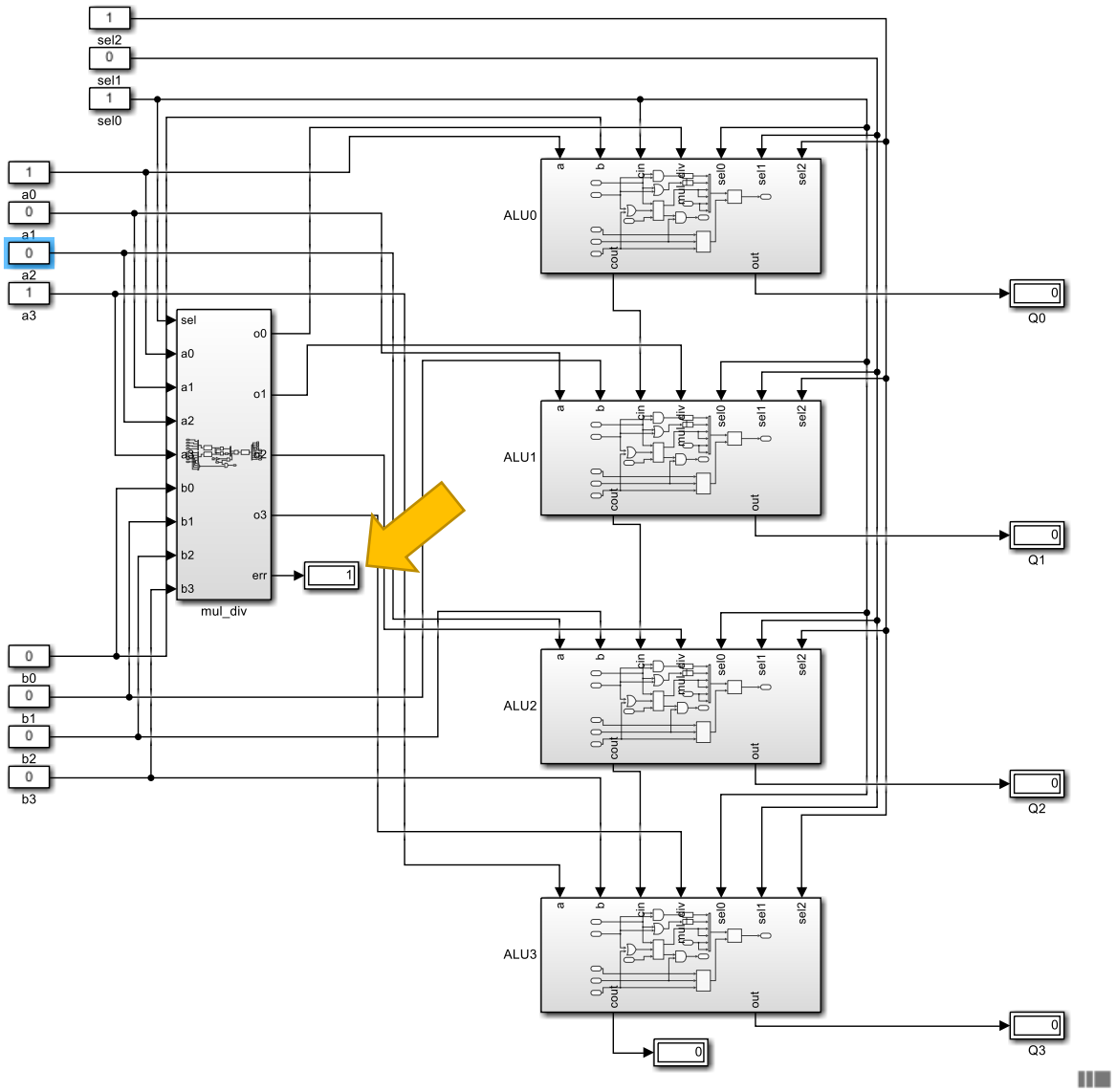
1011 * 0010 = 0101

11 / 2 = 5



1101 / 0011 = 0100

13 / 3 = 4



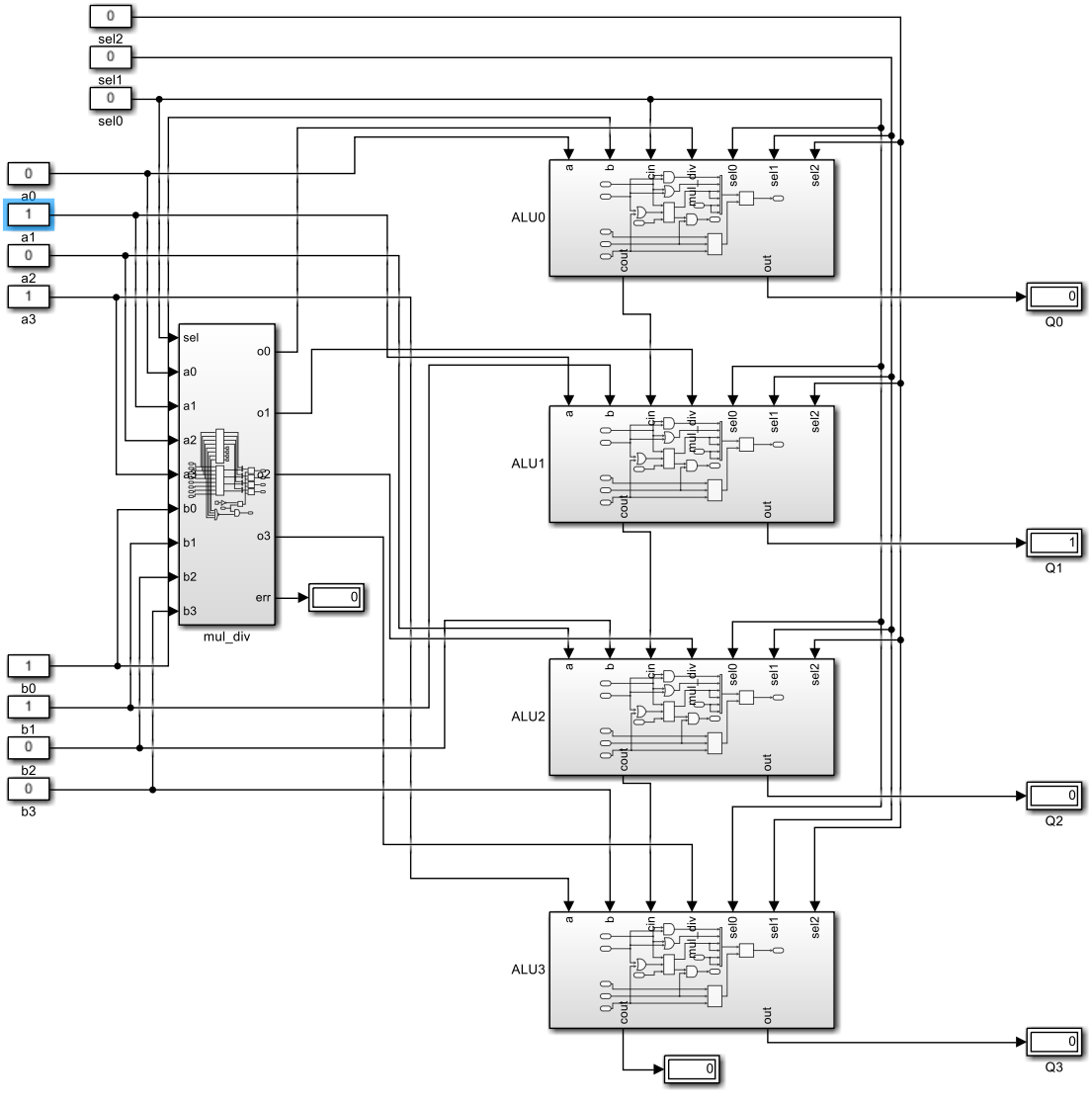
1001 / 0000 = 0000 (error flag is true!)

9 / 0 = inf

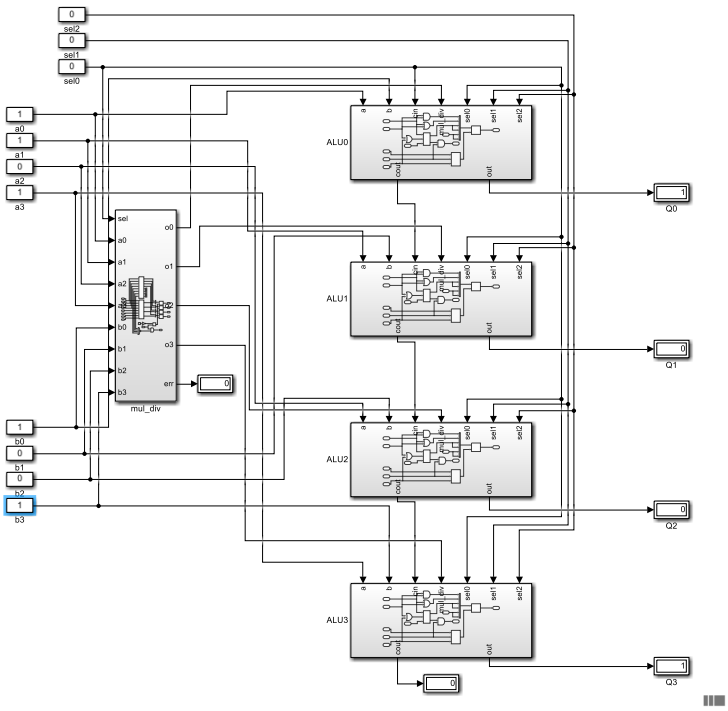
در اینجا حاصل 9 تقسیم بر 0 را که نتیجه اشتباه است را داریم و بیت ارورمان یک شده است.

نمونه های پروژه با گیت های دستی و دارای بلوک تقسیم ضرب

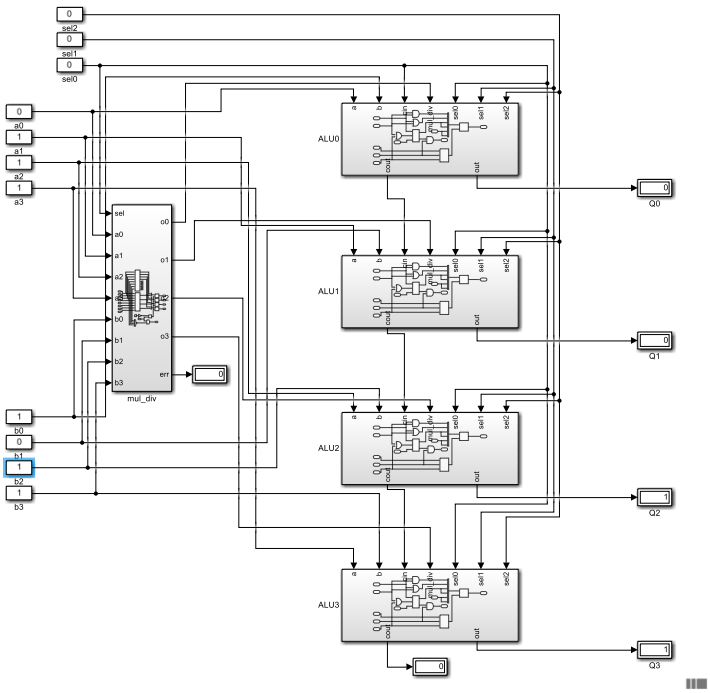
بررسی AND



ورودی a برابر 1010 و b برابر 0011 است که حاصل and آنها 0010 است.

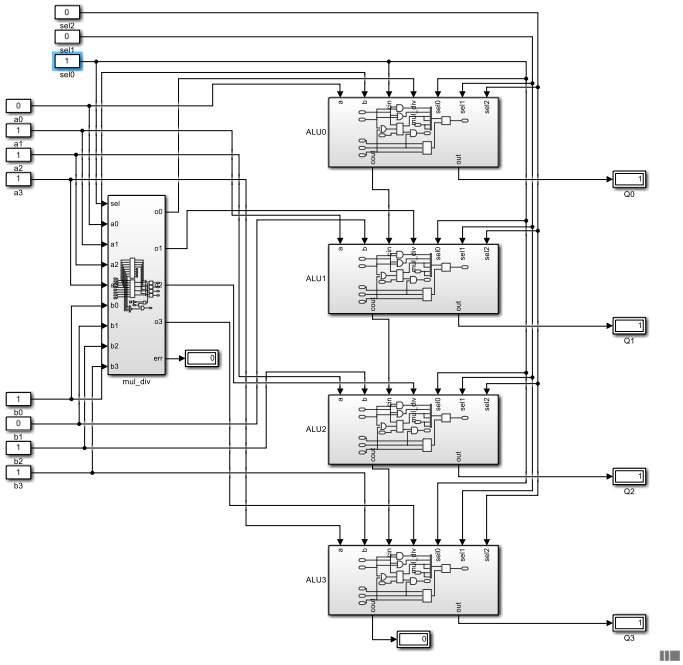


1011 and 1001 = 1001

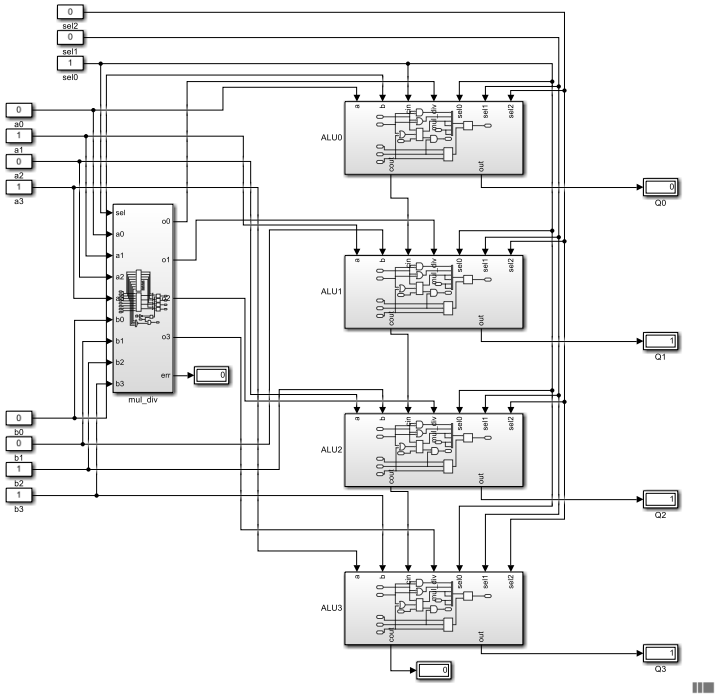


1110 and 1101 = 1100

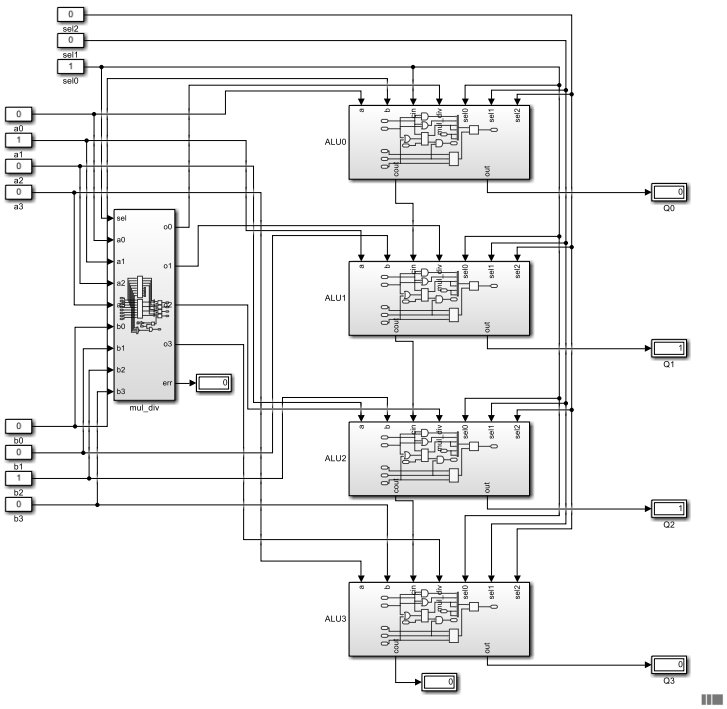
بررسی OR



1110 or 1101 = 1111

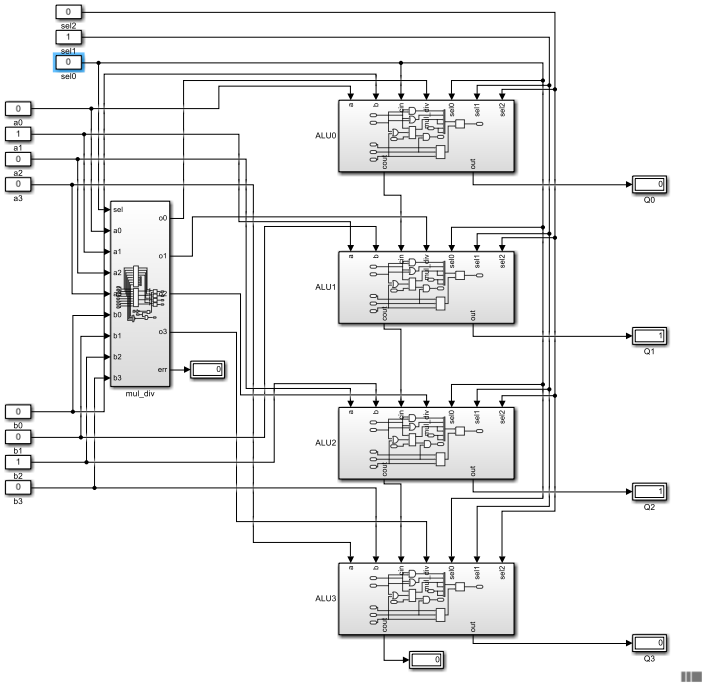


1010 or 1100 = 1110



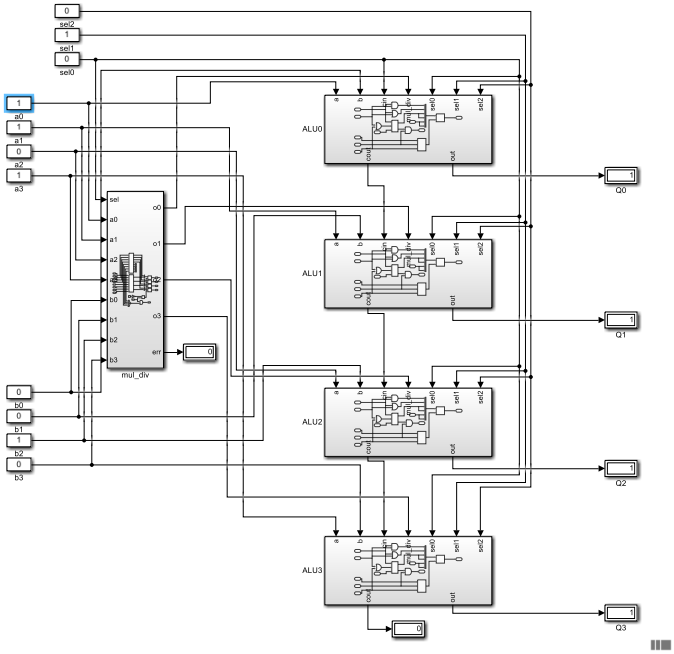
0010 or 0100 = 0110

بررسی جمع



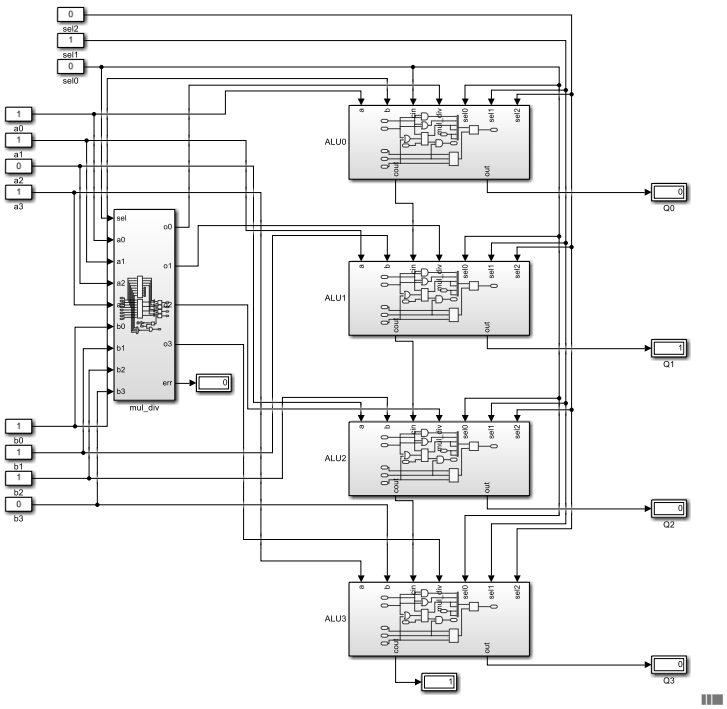
0010 + 0100 = (0)0110

2 + 4 = 6



1011 + 0100 = (0)1111

11 + 4 = 15

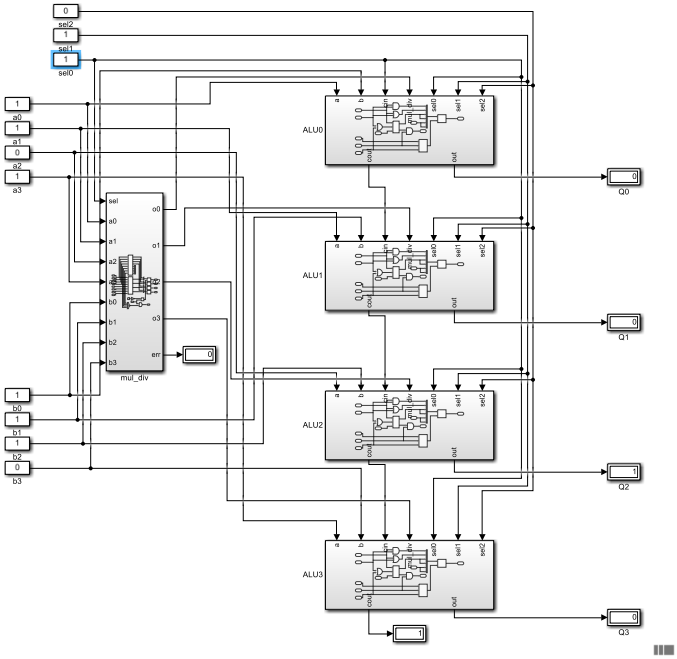


1011 + 0111 = (1)0010

11 + 7 = 18

در اینجا از خروجی کری هم استفاده شده . . .

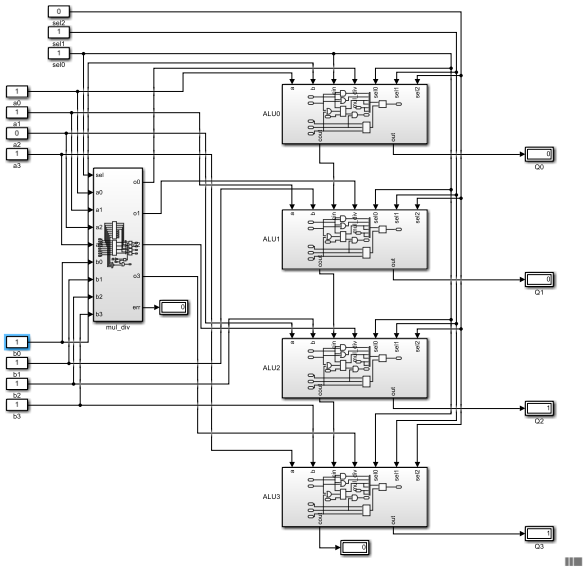
بررسی تفریق



1011 – 0111 = 0100

11 – 7 = 4

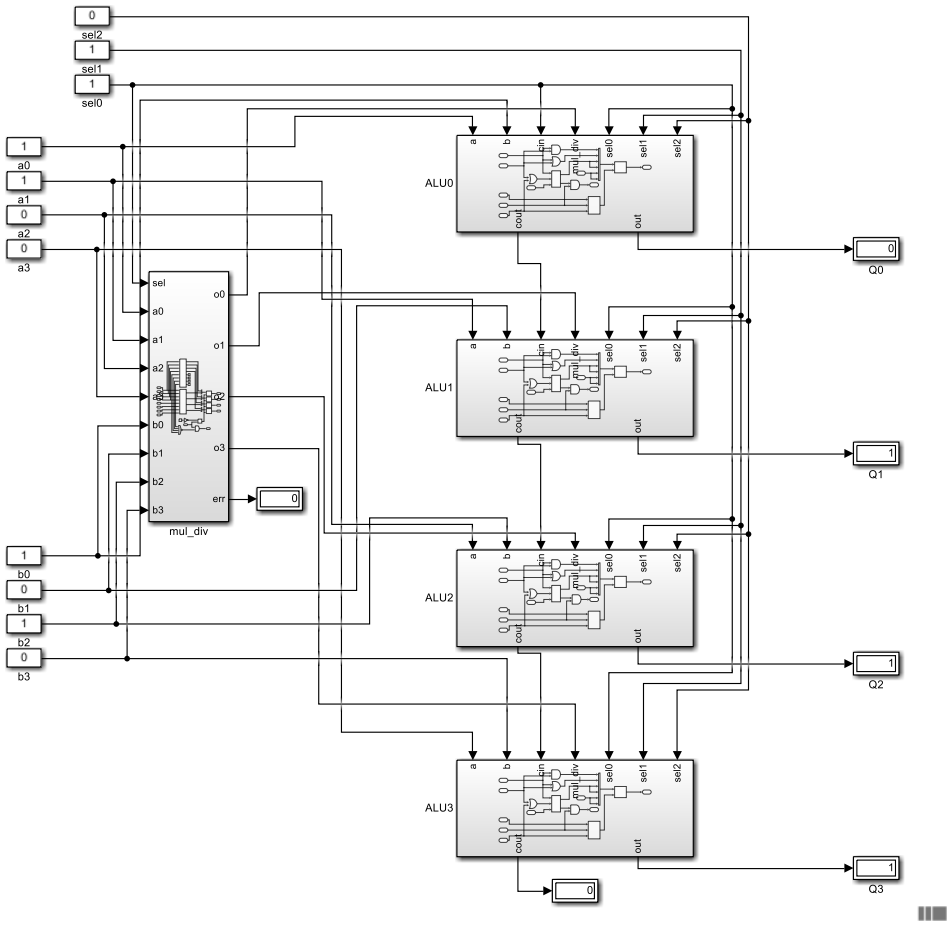
بیت کری در اینجا اضافی است.



1011 – 1111 = 1100

11 – 15 = -4

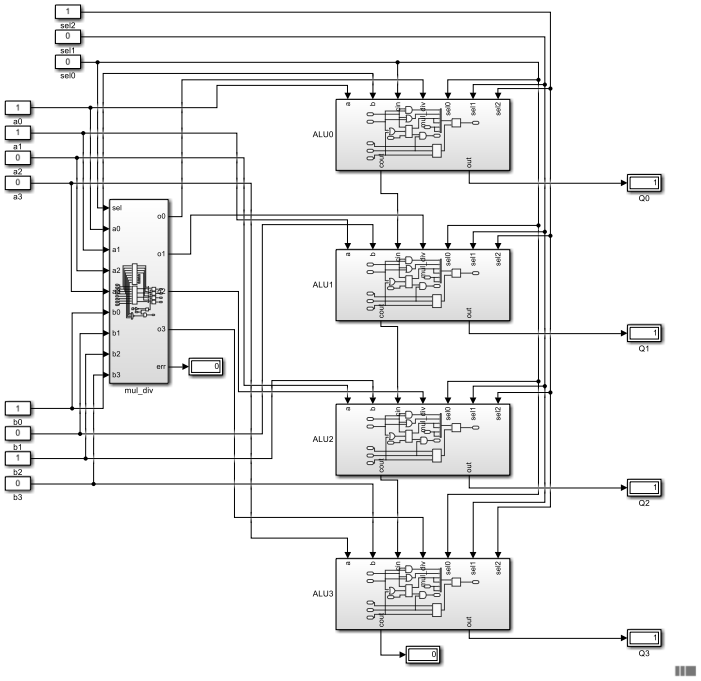
خروجی در اینجا بر مبنای متمم دو نوشته شده است.



0011 – 0101 = 1110

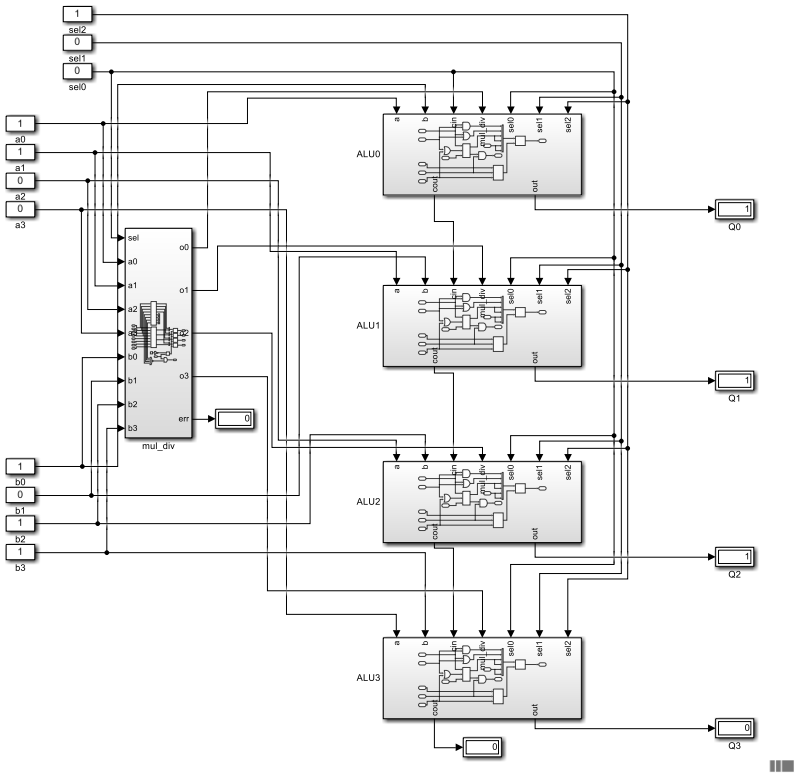
3 – 5 = -2

بررسی ضرب



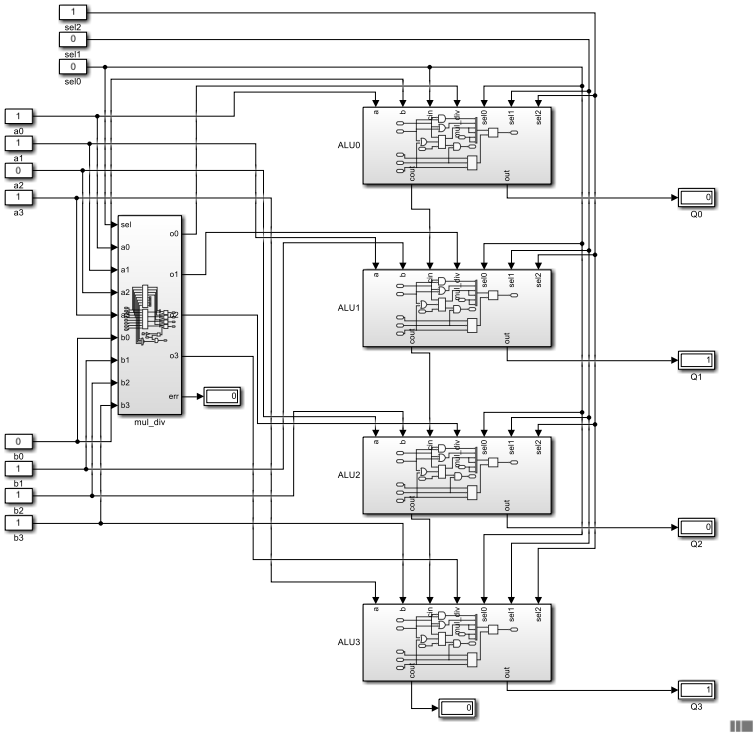
0011 * 0101 = (0000)1111

3 * 5 = 15



0011 * 1101 = (0010)0111

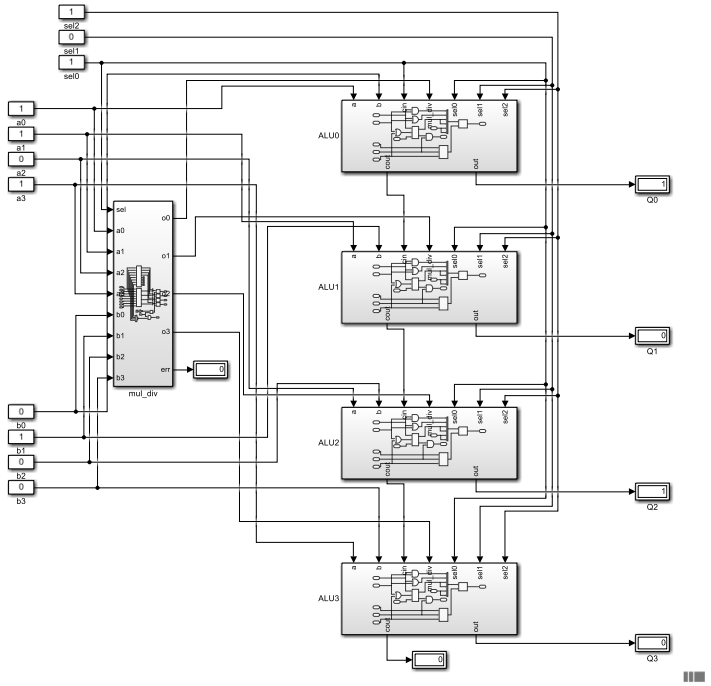
3 * 13 = 39



1011 * 1110 = (1001)1010

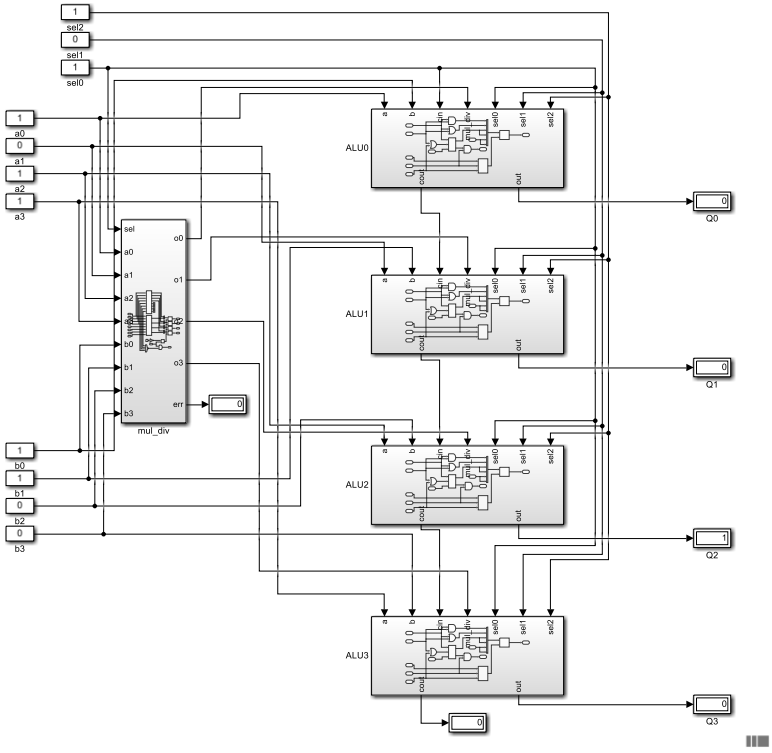
11 * 14 = 154

بررسی تقسیم



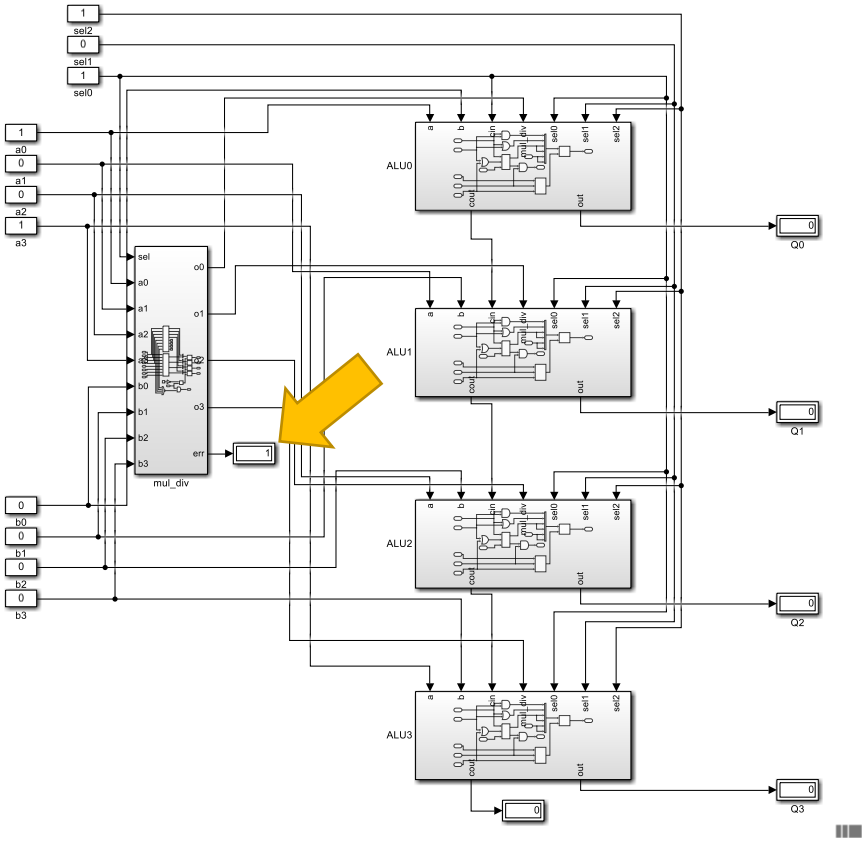
1011 * 0010 = 0101

11 / 2 = 5



1101 / 0011 = 0100

13 / 3 = 4



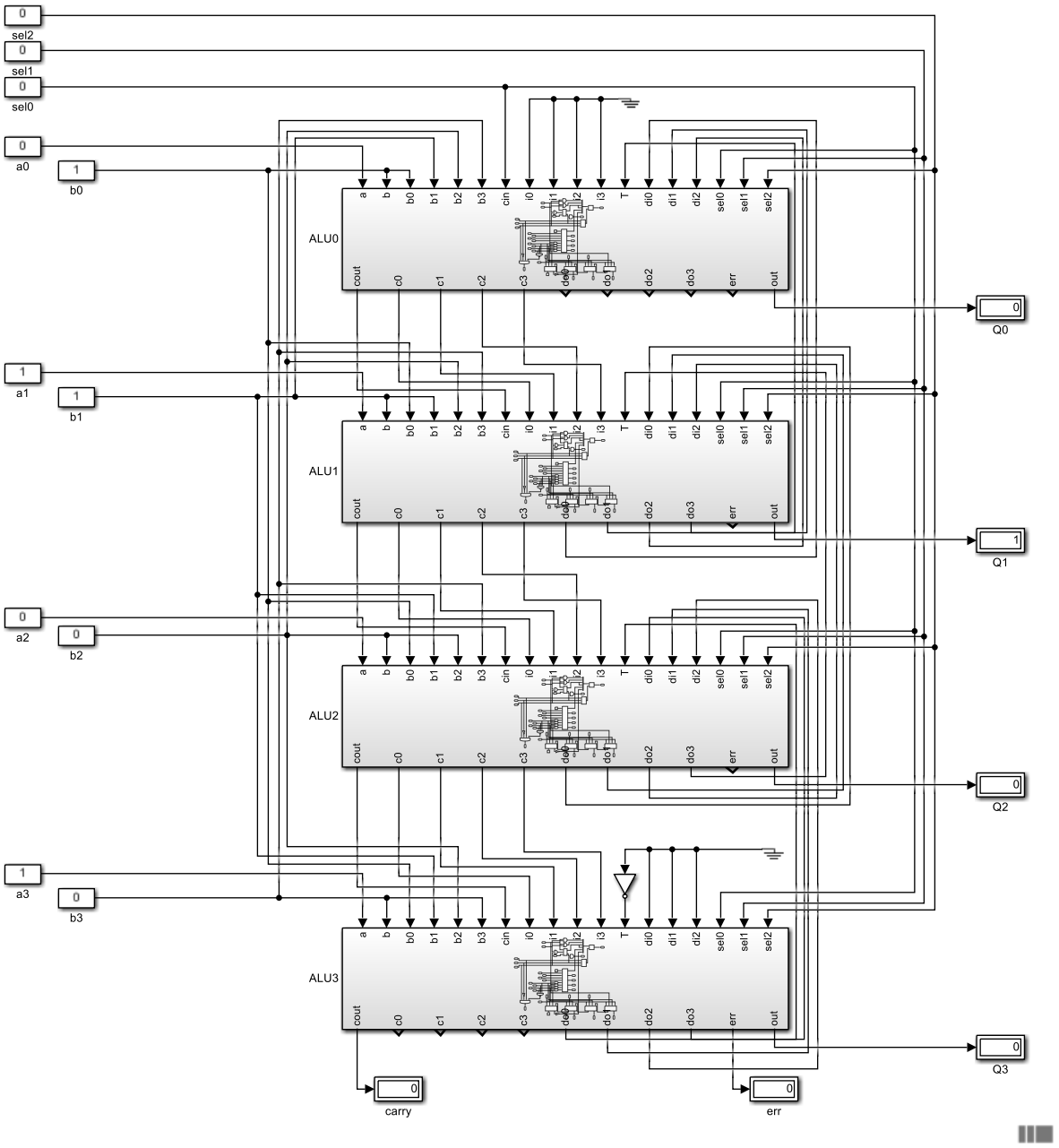
1001 / 0000 = 0000 (error flag is true!)

9 / 0 = inf

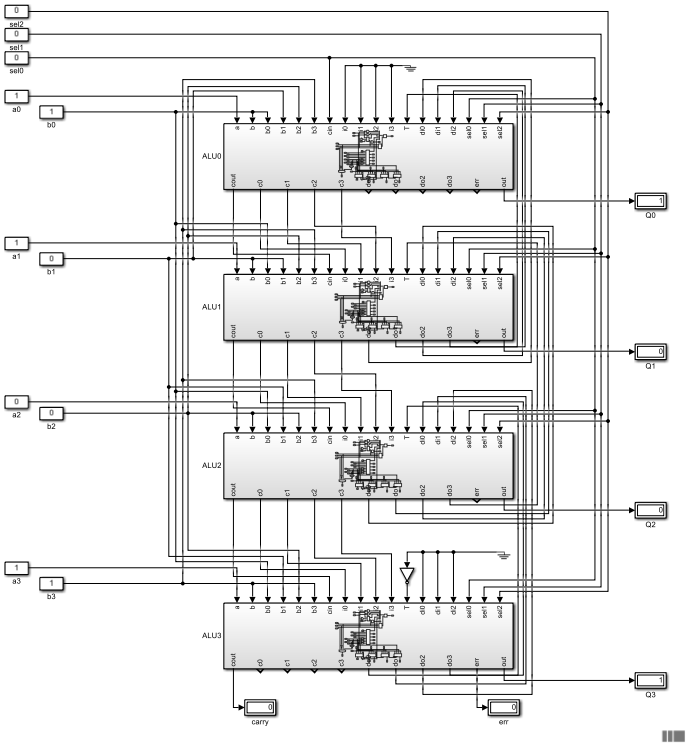
در اینجا حاصل 9 تقسیم بر 0 را که نتیجه اشتباه است را داریم و بیت ارورمان یک شده است.

نمونه های پروژه تماما بلوکی با گیت های دستی

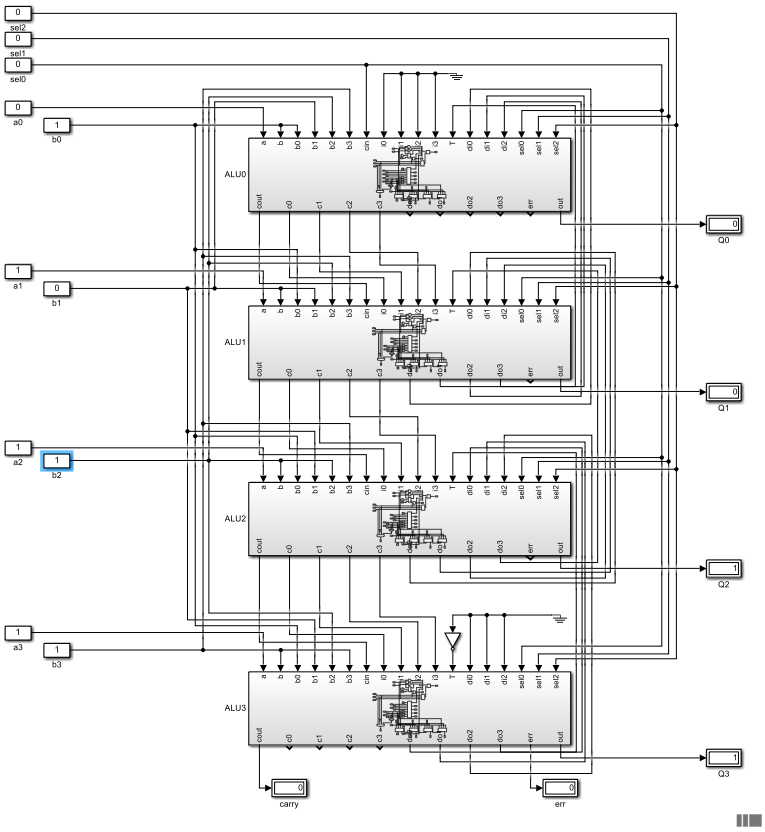
بررسی AND



ورودی a برابر 1010 و b برابر 0011 است که حاصل and آنها 0010 است.

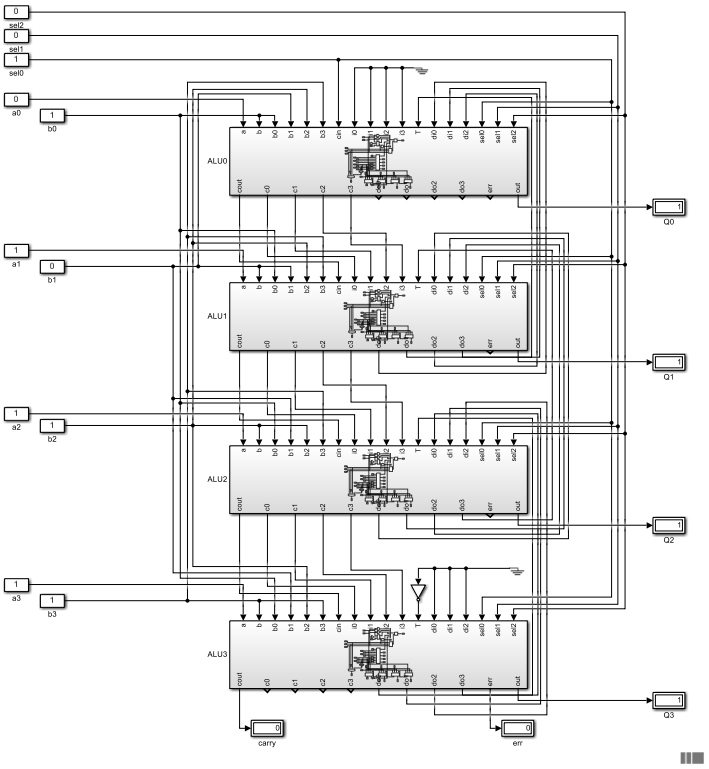


1011 and 1001 = 1001

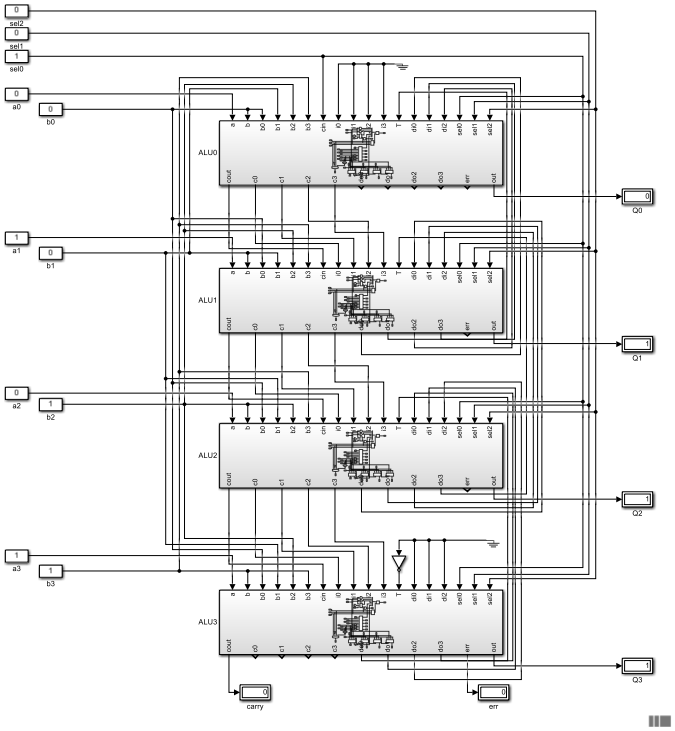


1110 and 1101 = 1100

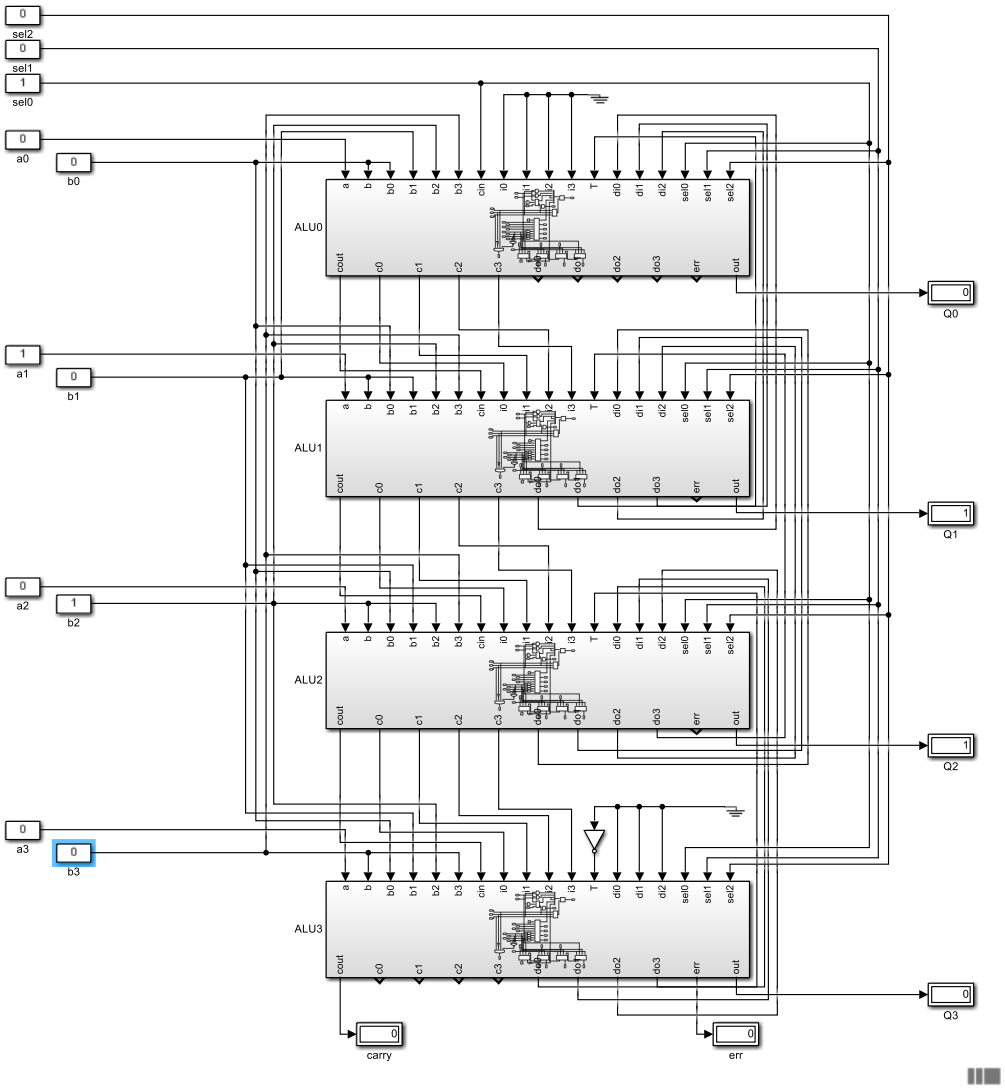
بررسی OR



1110 or 1101 = 1111

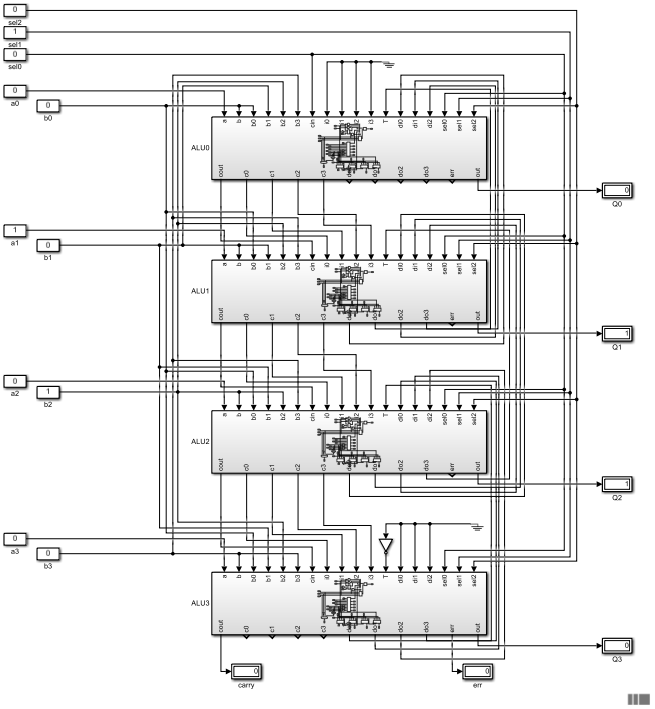


1010 or 1100 = 1110



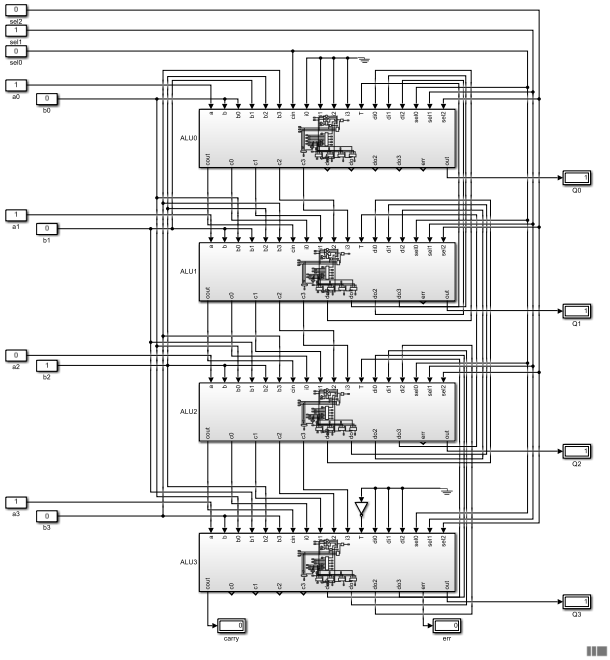
0010 or 0100 = 0110

بررسی جمع



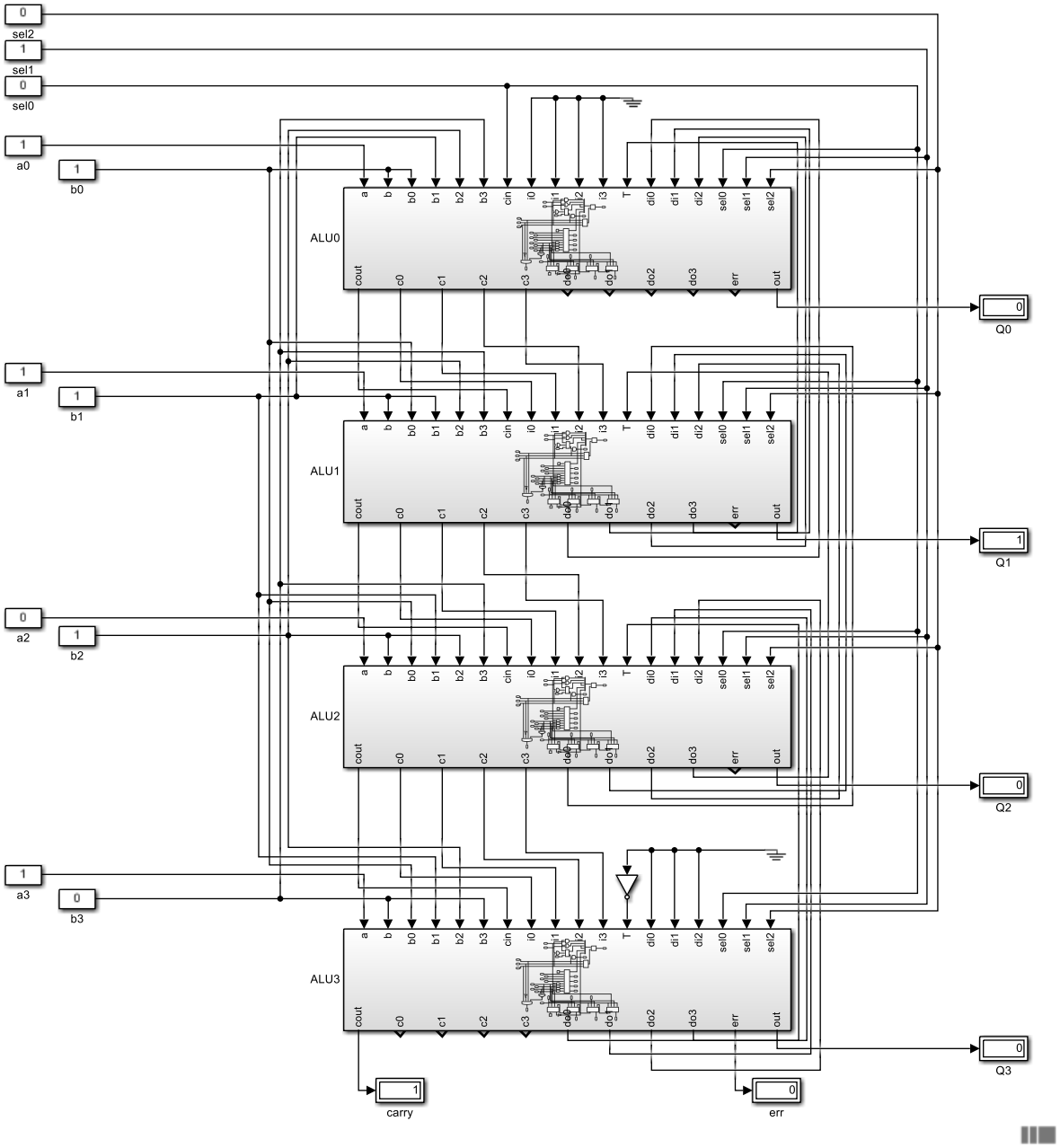
0010 + 0100 = (0)0110

2 + 4 = 6



1011 + 0100 = (0)1111

11 + 4 = 15

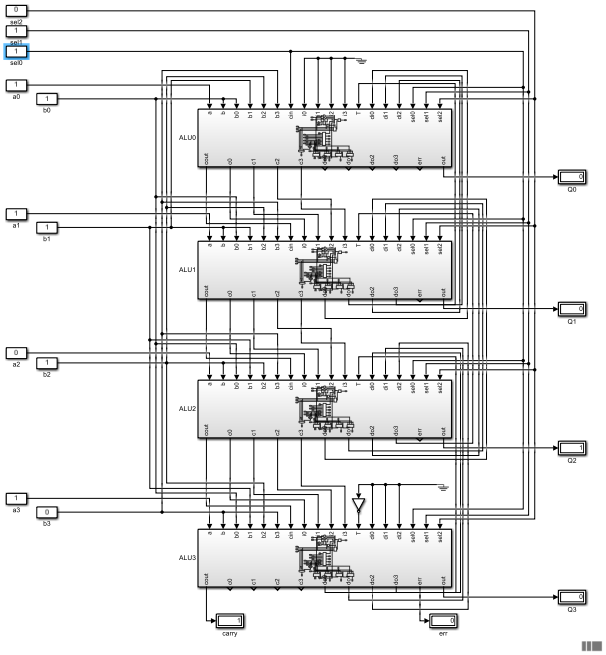


1011 + 0111 = (1)0010

11 + 7 = 18

در اینجا از خروجی کری هم استفاده شده . . .

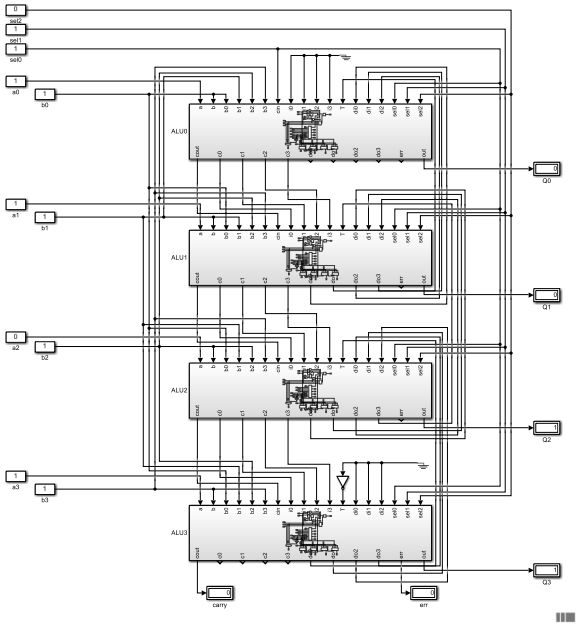
بررسی تفریق



1011 – 0111 = 0100

11 – 7 = 4

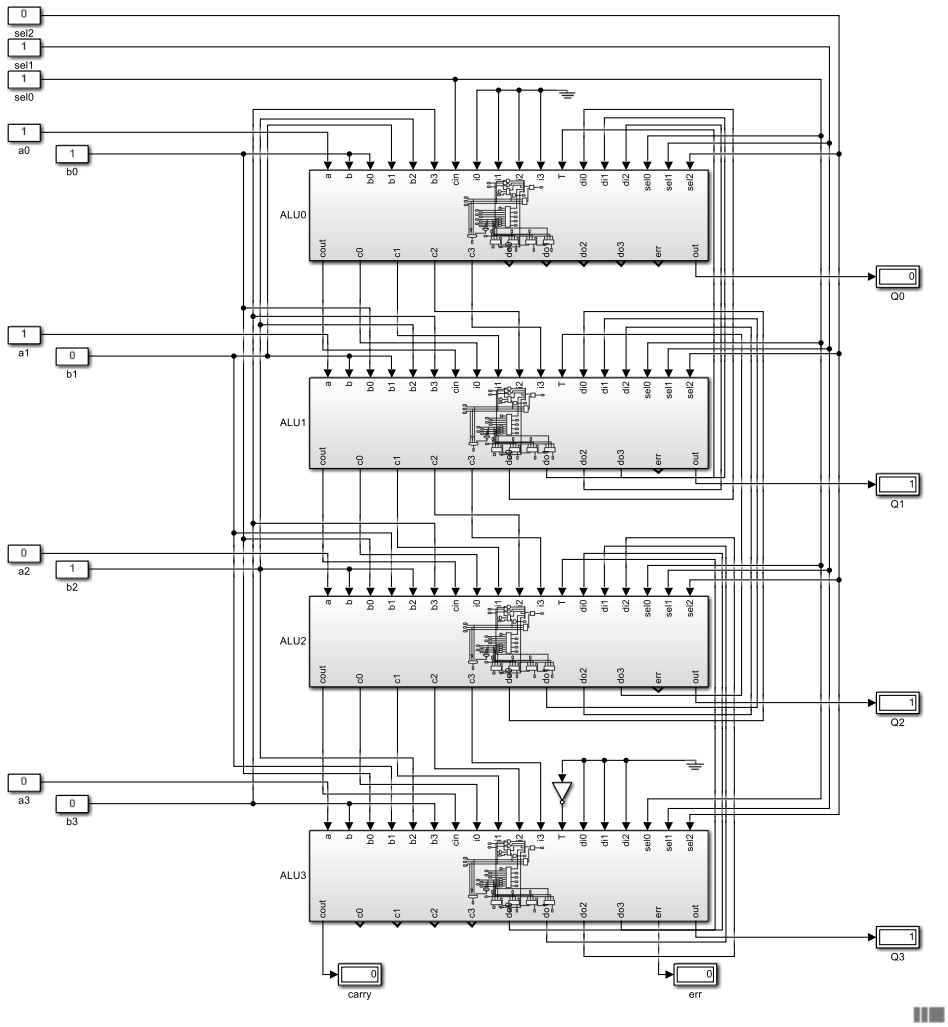
بیت کری در اینجا اضافی است.



1011 – 1111 = 1100

11 – 15 = -4

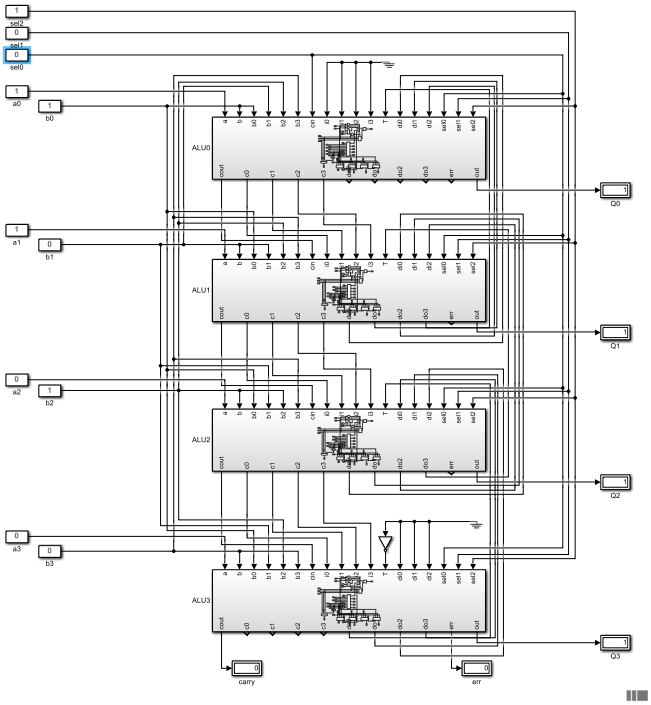
خروجی در اینجا بر مبنای متمم دو نوشته شده است.



0011 – 0101 = 1110

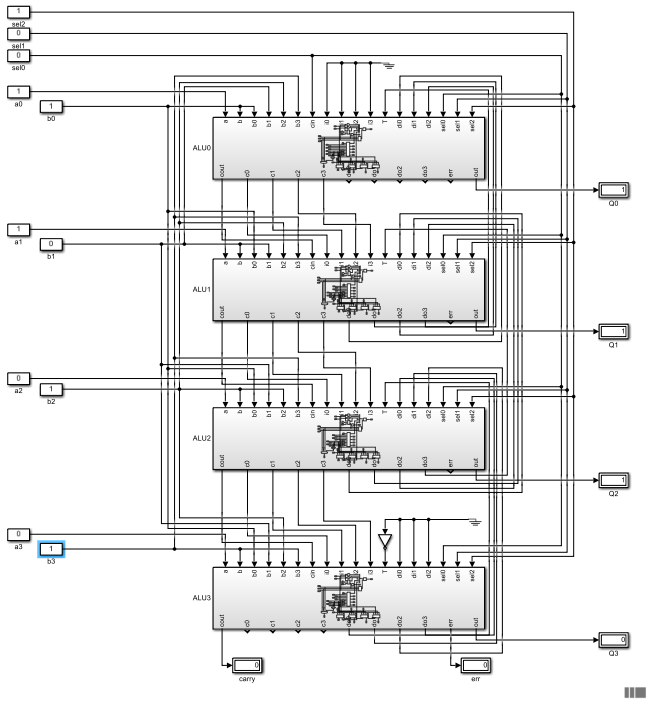
3 – 5 = -2

بررسی ضرب



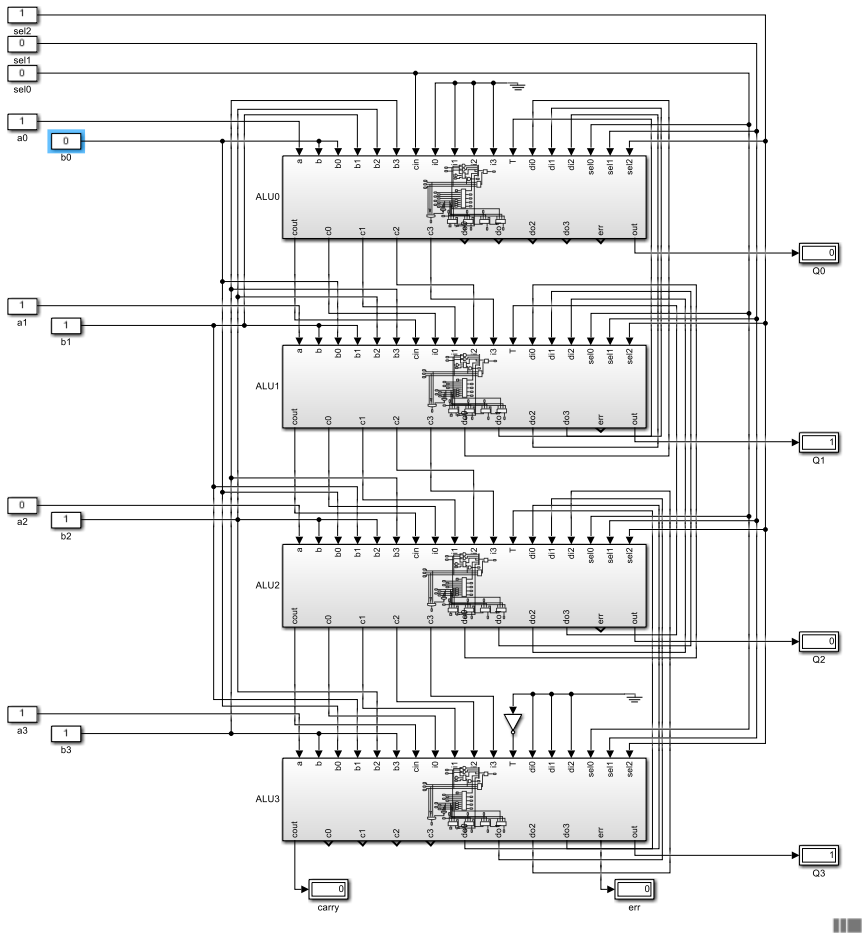
0011 * 0101 = (0000)1111

3 * 5 = 15



0011 * 1101 = (0010)0111

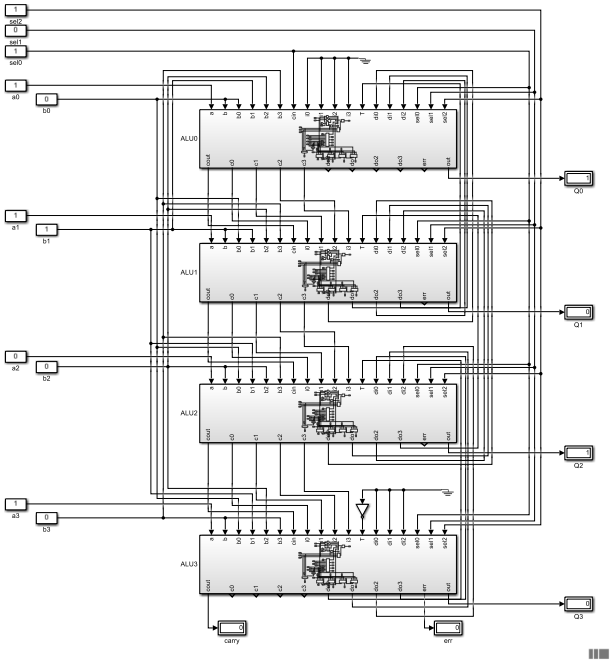
3 * 13 = 39



1011 * 1110 = (1001)1010

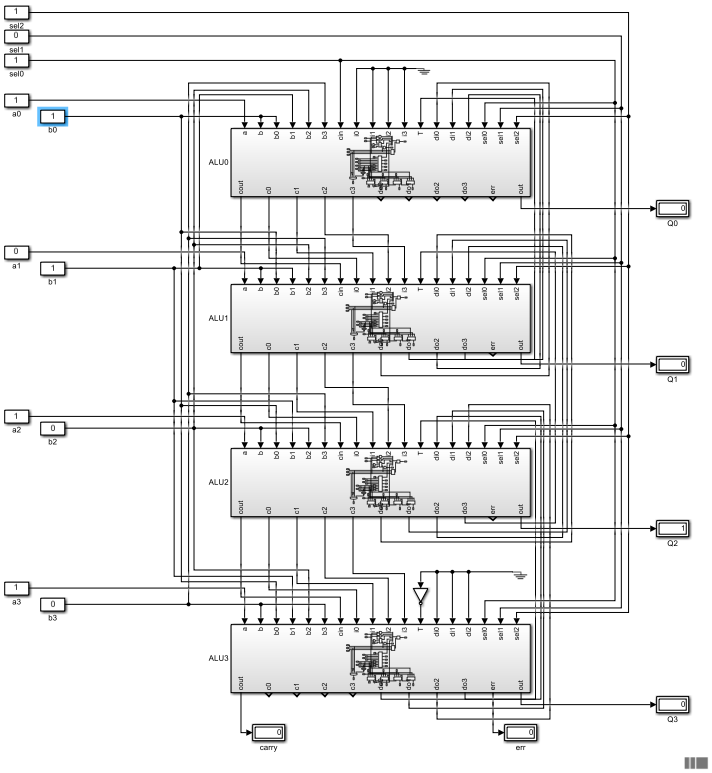
11 * 14 = 154

بررسی تقسیم



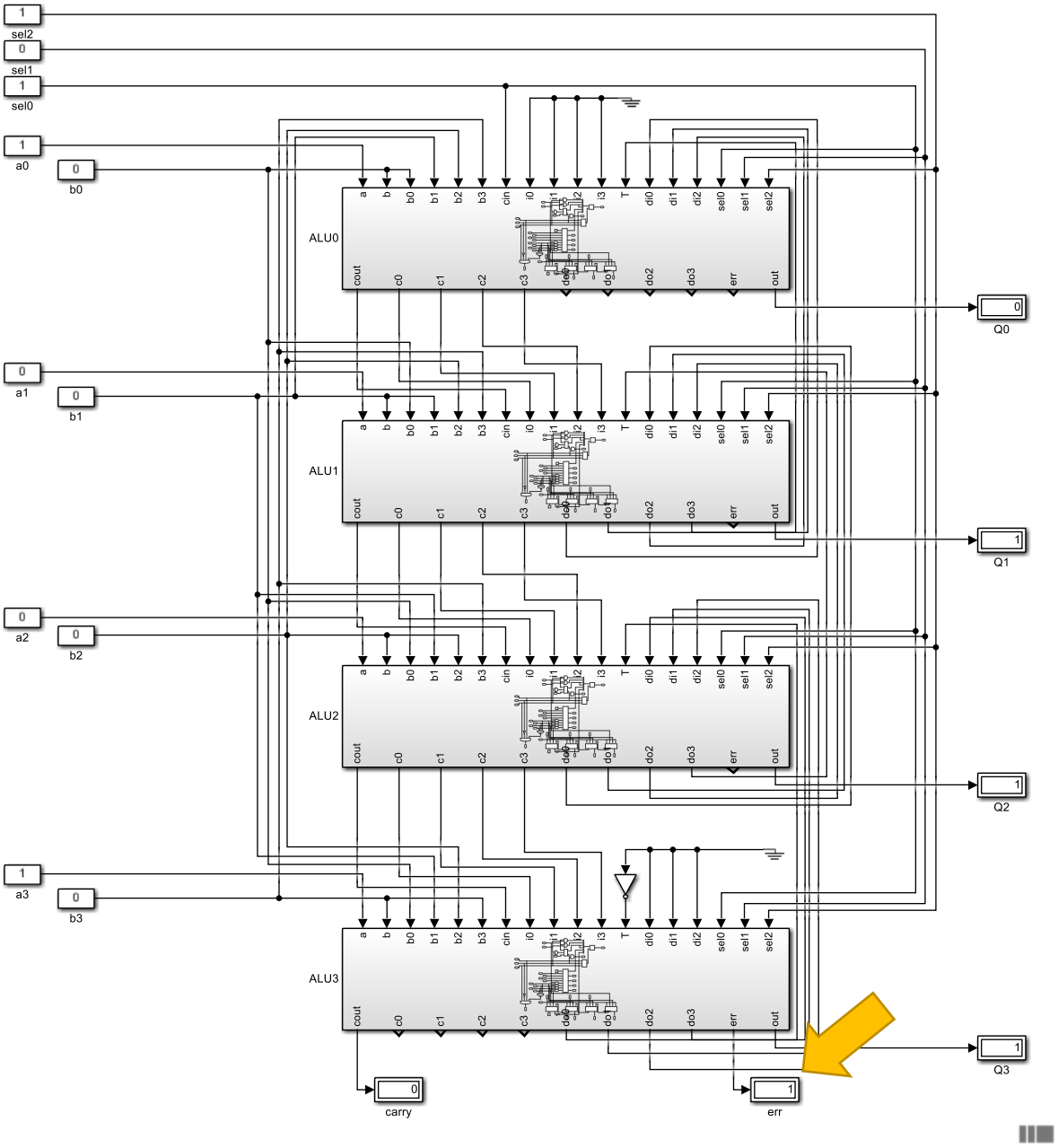
1011 * 0010 = 0101

11 / 2 = 5



1101 / 0011 = 0100

13 / 3 = 4



1001 / 0000 = 1110 (error flag is true!)

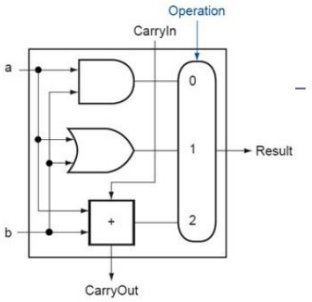
9 / 0 = inf

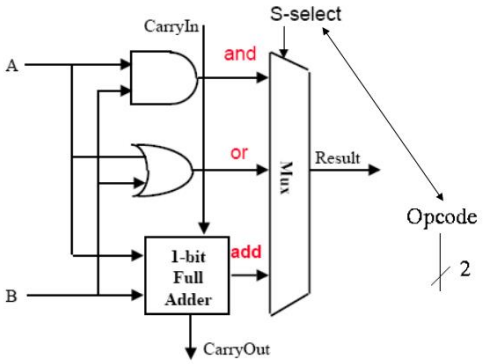
در اینجا حاصل 9 تقسیم بر 0 را که نتیجه اشتباه است را داریم و بیت ارورمان یک شده است.

پیوست ها

Designing a 1-Bit ALU

- This is an one-bit ALU which can do Log
  ADD operation (2 bits opcode).
  - Result = a AND b when opcode = 0 (
  - Result = a OR b when opcode = 1 (01
  - Result = a ADD b when operation = :



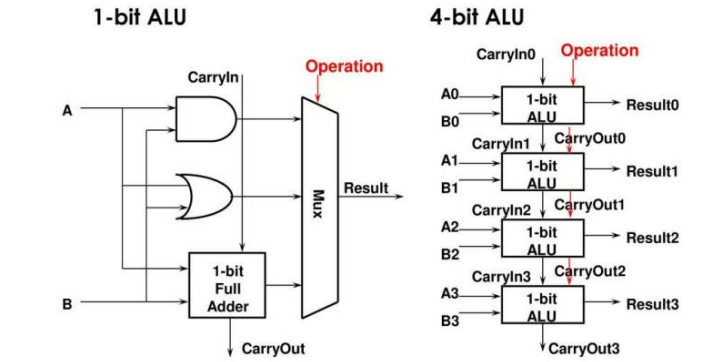


There are three results while we need only one output. So use 3-1 **MUX with Opcode as its select line.** So Opcode is now 2 bits.

Connect Cin (from previous bit) and Cout (to next bit).

92

A 4-bit ALU



National Tsing Hua University
COMPUTER ARCHITECTURE
8

سوال پروژه معماری صندوق ورودی

Rahmat Ansari <rahmat2022a@gmail.com>
به مینا

سلام وقتتون بخیر استاد
میخواستم بدونم **آیا باید بلوک های alu مون (مثلا بلوک اول و آخر) کاملا مشابه هم باشه یا نه لازم نیست؟**
مشکل اصلی ضرب کننده و تقسیم کننده هستند چراکه اگر alu بلوکی باشه و بخواهیم طبق روال مرسوم به ضرب کننده طراحی کنیم برای بدست آوردن بیت چهارم خروجی به بیت اول یکی از ورودی ها نیاز پیدا میکنیم.
ممنون میشم جواب بدید

سه‌شنبه ۷ ژوئن، ۲۲:۳۹ (۱۰ روز پیش)

مینا ملک زاده
به من

چهارشنبه ۸ ژوئن، ۴:۴۰ (۹ روز پیش)

سلام

بله باید مشابه باشند

From: Rahmat Ansari <rahmat2022a@gmail.com>
Sent: Tuesday, June 7, 2022 10:39:18 PM
To: مینا ملک زاده
Subject: سوال پروژه معماری

Back | Archive | Move | Delete | Spam

(No Subject) 5 — Yahoo/Inbox

Nima Mazloom <mazloomnima79@yahoo.com>
To: m.malekzadeh@hsu.ac.ir — Sun, Jun 5 at 6:14 PM

سلام استاد ببخشید من نیما مظلوم هستم میخواستم ببینم برای پروژه ای که گفتید انجام بدین گفتین از mux 4 استفاده کنیم ولی برای ضرب به 8 تا ماکس نیاز داریم ؟

Nima Mazloom <mazloomnima79@yahoo.com>
To: m.malekzadeh@hsu.ac.ir — Sun, Jun 5 at 6:15 PM

باید چیکار کنیم؟

Show original message

مینا ملک زاده <m.malekzadeh@hsu.ac.ir>
To: Nima Mazloom — Mon, Jun 6 at 7:59 AM

نگفتم 4 تا ماکس استفاده کنید گفتم ۱ ال یو چهار بیتی

Show original message

Back | Archive | Move | Delete | Spam

(No Subject) 2 — Yahoo/Inbox

Nima Mazloom <mazloomnima79@yahoo.com>
To: مینا ملک زاده — Sat, Jun 11 at 6:34 PM

سلام استاد ببخشید من همه قسمت های پروژه رو جز ضرب و تقسیم رقم میخواستم ببینم برای ضرب هم اگر با ای سی حاضر برم مثل شکل میشه که مثل جمع و تفریق بلوکی نمیشه میخواستم ببینم باید همینجوری برم چون بلوکی نمیشه اگر بخوام بلوکی برم باید ضرب رو کلا کاملا طراحی کنم

photo_2022-... .jpg
57.6kB

مینا ملک زاده <m.malekzadeh@hsu.ac.ir>
To: Nima Mazloom — Sun, Jun 12 at 3:39 AM

سلام
مشکلی نیست

Show original message

Reply, **Reply All** or **Forward**

Innovation distinguishes between a leader and a follower.
Steve Jobs 1955-2011

نوآوری آن چیزی است که تفاوت میان رهبر و دنبال کنندگان رقم میزند.

استیو جابز