# CAD CA3

Mahdy Mokhtari, Reza Chehreghani
810101515, 810101401

در این پروژه به ساخت پروژه اول که یک ضرب کننده تقریبی بود پرداختیم اما با تفاوت اینکه ما یک کتابخانه داشتیم به این صورت که 4 block که هرکدام شامل تعدادی گیت بود. 2 تا از circuit block ها عملا دونوع مدار ترکیبی (combinational) و دو نوع مدار ترتیبی (sequantial) بودند. ما می بایست کل دیتاپث و کنترلری که طراحی کرده بودیم در پروژه اول را بااین 4 تا بلاک c1, c2, s1, s2 می ساختیم و از هیچ گونه گیت منطقی حتی not هم استفاده نکردیم.
در برخی جاها خیلی محدود مجبور میشدیم یک گیت and یا or داشته باشیم که آن را هم با 2 بلاک ترکیبی c1 , c2 درست کردیم.

## یک نکته راجب s1 , s2:

این دو عملا تعداد یکسانی گیت (13) مصرف کرده بودند. و اگر در بلاک s2 ما به سیگنال ورودی B0 سیگنال ورودی clr را هم بدهیم عملا همان s1 میشود . لذا از آنجایی که بلاک s1 عملا زیرمجموعه بلاک s2 و انرا میتونستیم با s2 بسازیم همزمان تعداد گیت های مصرفی انها هم یکسان بود یعنی عملا s1 هیچ مزیت اضافه تری ندارد پس ما فقط از s2 برای بخش های ترتیبی استفاده کردیم.

## پیاده سازی کنترلر (controller):

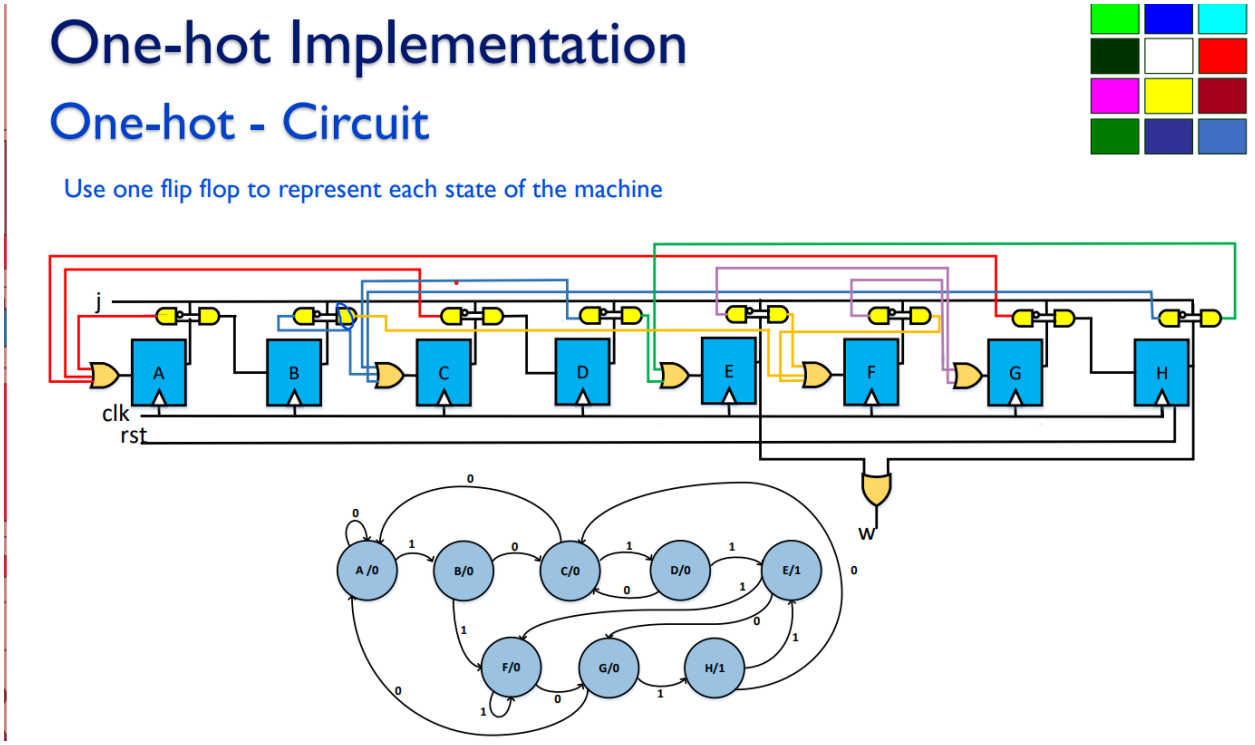
ما کنترلر را به صورت **one hot** پیاده سازی کردیم. 8 استیت داشتیم که عملا یعنی 8 رجیستر نیاز داشتیم و هرکدام تعدادی استیت ورودی و تعدادی استیت خروجی داشت که برای خروجی ها نیاز به and کردن و برای ورودی ها نیاز به or داشتیم و خود منطق رجسیتر که همه اینها بسته به 3ورودی 2 خروجی یا مثلا 2ورودی 2 خروجی یا ... به صورت module control block طراحی شدند که هرکدام از تعدادی and, or, وs2 استفاده میکرد (که خود and , or ماژول هایشان با c1 درست شده بوند) .

خود کنترلر عملا متشکل از 8 تا control block است . البته به علت اینکه one hot استیت ابتدایی ان با 1000000 شروع میشود و ما چون در ابتدا نیاز به rst کردن و صفر کردن تمامی مقادیر داشتیم, استیت Idle را به صورت control block نگذاشتیم و عملا حالت 0000000 را Idle استیت درنظر گرفتیم تا به مشکلی در شروع کار نداشته باشیم و حتما وقتی rst میشود یعنی عملا در استیت Idle هستیم و بعد از دیدن سیگنال start = 1 که ورودی مسقتیم رجیتسر اول یعنی Init است مقداری که رجسیتر اول ما نگه میدارد مقدار یک است و از این به بعد بسته به سیگنال های

کنترلری که به استیت دیگری برویم این استیت که رجیستر ان مقدار صفر میگیرد و استیت بعدی که به ان میرویم مقدار 1 را در خود ذخیره میکند.

منطق اینکه به کدام استیت بعدی برویم با and های خاص منظوره طراحی شده که سیگنال و خروجی رجیستر خود همان استیت را میگیرد و خرجی انها مشخص میکند به کدام استیت خواهم رفت و یک خواهیم شد. در ابتدا بعد Init با start=0 شدن به استیت ZeroCntA میرویم که عملا به صورت 0100000 نمایش داده میشود.

دلیل 7 رقم بودن این است که 7 تا یک روان نشان گر 7 استیت و تمام صفر نشانن گر استیت Idle است.




کنترلر ما از این تصویر الهام گرفته شده است

## پیاده سازی دیتاپث (datapath):

دیتاپث ما شامل 2 تا شیفت رجیستر 16 بیت, 1 ضرب کننده, 1 شیفت رجسیتر 32 بیت و 2 تا counter است.

که با سیم ها (wire) به هم دیگر وصل هستند با سیگنال های کنتلری کار میکنند به طوری که یک ضرب کننده تقریبی را درست میکنند.

به جز این ها هم یک سری موارد خیلی جزعی بودند که استفاده شدند که در کد و تصاویر انتهایی مشخص شده است.

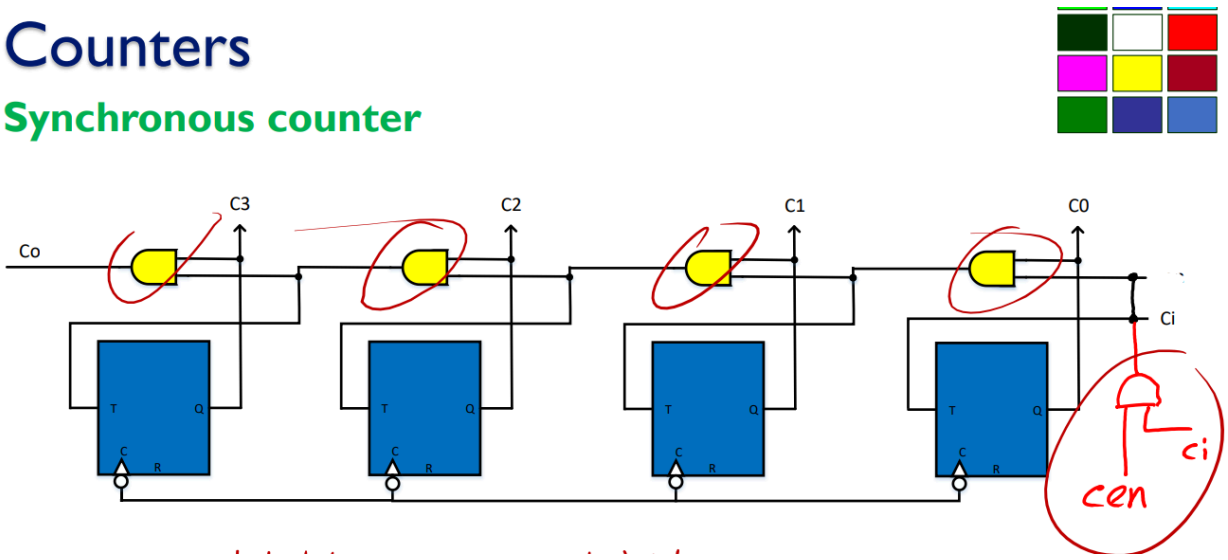
شیفت رجیستر ها (shift register):

شیفت رجیستر ها ما به چپ شیفت میدهند و با مقدار 0 پر میکنند. برای ساخت شیفت رجیستر های 32 و 16 بیتی ما نیاز داشتیم در ابتدا شیفت رجیستر 1 بیتی را بسازیم. بعد از ساخت شیفت رجیستر 1 بیت با عملا کنار هم گذاشتنشان به هدف خود میرسیم که عملا با generate block ها ان را زدیم.

برای ساخت شیفت رجیستر 1 بیت که عملا ما اسم ان را shift_ff گذاشتیم با استفاده از s2 ان را ساختیم که در انتها میتوانید تصویر ان را ببینید که چگونه ورودی های s2 مقدار دهی شده اند تا منطق یک shift flip flop را به وجود بیاورند با سیگنال های مورد نیاز برای پروژه ما.

شمارنده (counter):

از انجایی که 3 بیت نیاز داشتیم با 3 عدد t flip flop و 3 عدد and آن را ساختیم. که خود t flip flop را با استفاده از s2 ساختیم که در تصاویر زیر میتوانید نحوه مقدار دهی ان و منطق آن را ببینید.

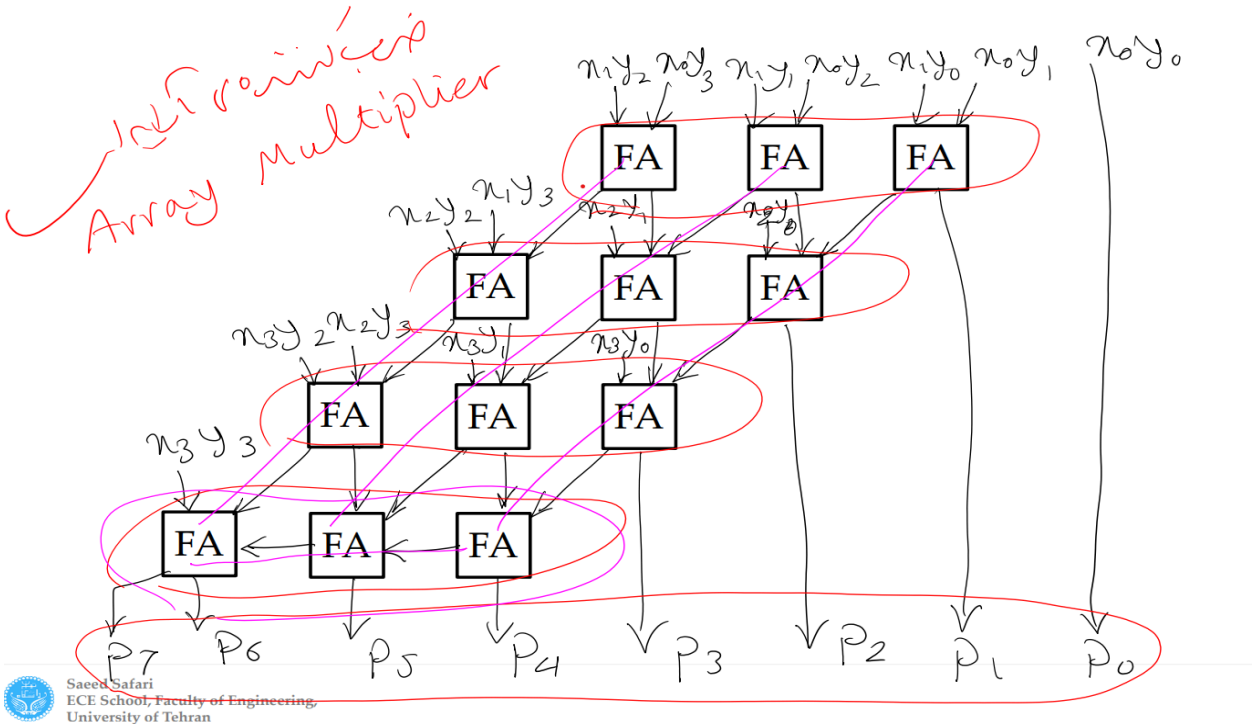



شمارنده ما از این تصویر الهام گرفته است.

ضرب کننده (multiplier) :

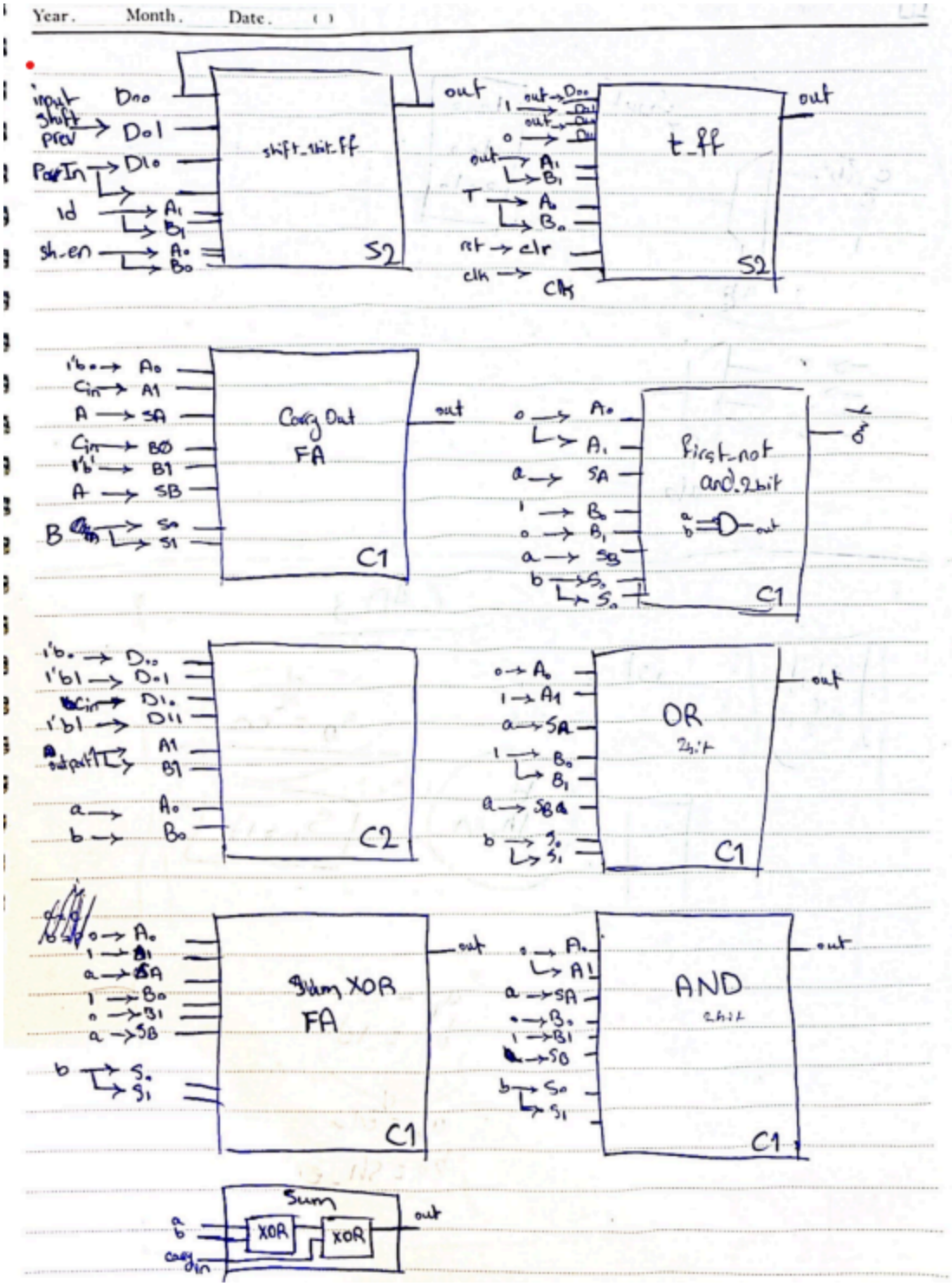
برای ساخت ضرب کننده 8 بیت در 8 بیت که خروجی 16 بیت میدهد از ضرب کننده ارایه ای (array multiplier) استفاده کردیم. پس یعنی عملا به تعدادی full adder و عبارت and نیاز داشتیم که هرکدام از این FA ها را با استفاده از 2 تا xor (که با c1 درست کردیم از قبل) برای بخش sum و برای بخش carry out از c1 استفاده کردیم که منطق ان و مقدار دهی های ورودی ان در تصویری که در زیر آمده مشخص شده است.

برای ساخت این ضرب کننده از بلاک های generate استفاده کردیم تا FA ها را به هم وصل کنیم و تشکیل یک ضرب کننده بدهند. با محساباتی که ما کردیم نوع ارایه ای ضرب کننده یک ترکیب خوبی از سادگی و استفاده کمتر از gate ها را به همراه داشت.

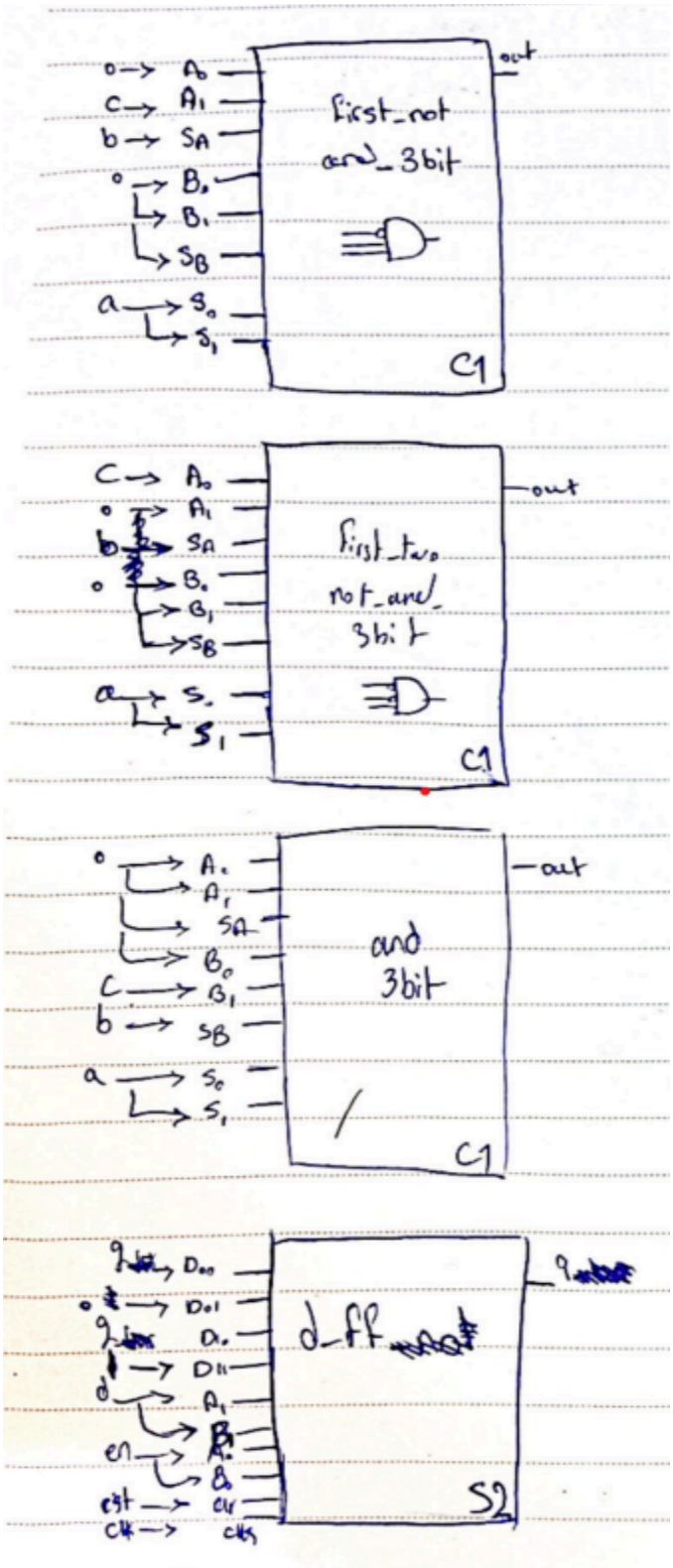



Saeed Safari
ECE School, Faculty of Engineering,
University of Tehran

ضرب کننده ارایه ای ما از این تصویر الهام گرفته شده است.



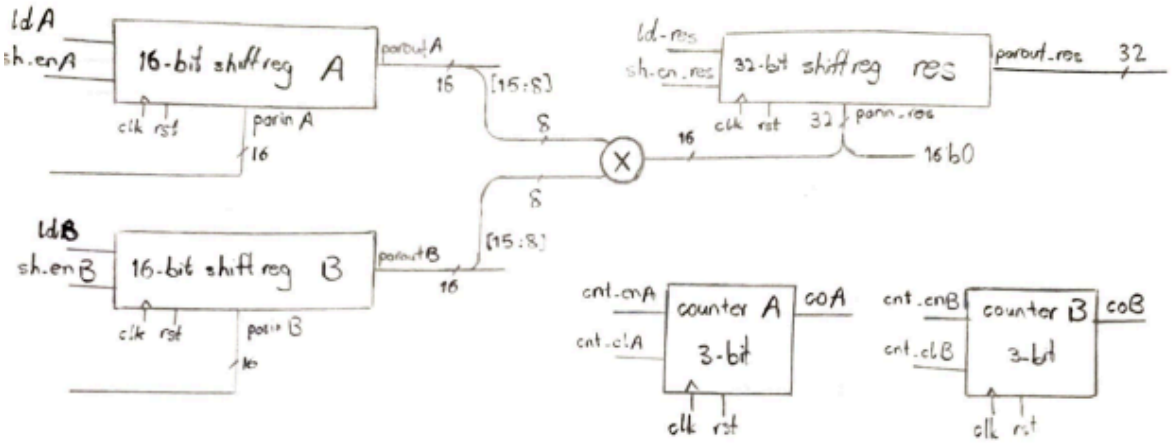

مواردی که با استفاده از c1, c2, s1 درست شدند.

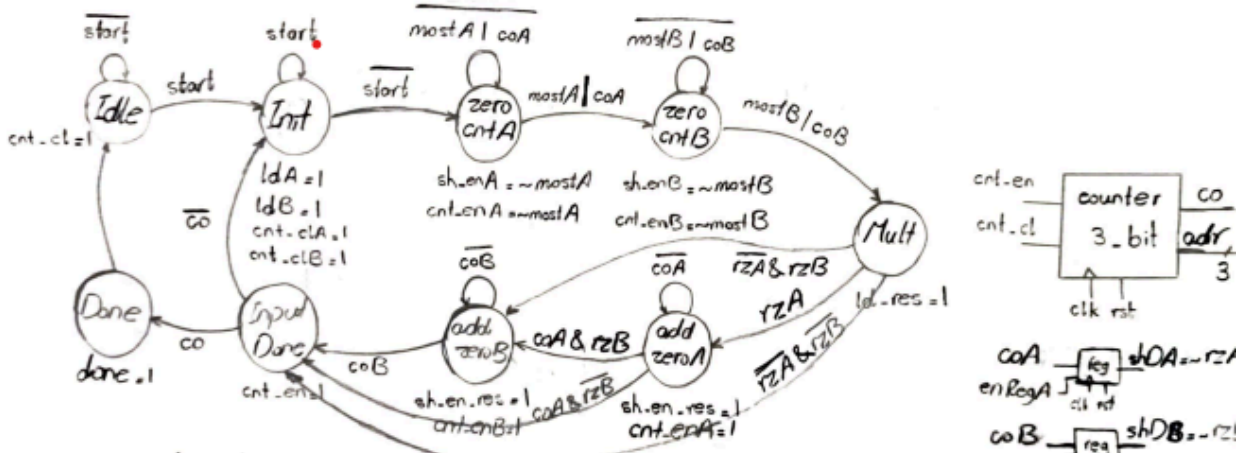



مواردی که با استفاده از c1, c2, s1 درست شدند.

تمرین اول  رضا چرقانی ۸۱۰۱۰۱۴۰۱  مهدی مختاری ۸۱۰۱۰۱۵۱۵

همه‌ی شیفت رجیسترها به سمت چپ شیفت می دهند و بیت راست را با صفر پر می‌کنند







در هر حلقه از خانه شماره adr دو عدد A و B را می‌خواند و مقدار res را در خانه شماره adr می‌نویسیم.