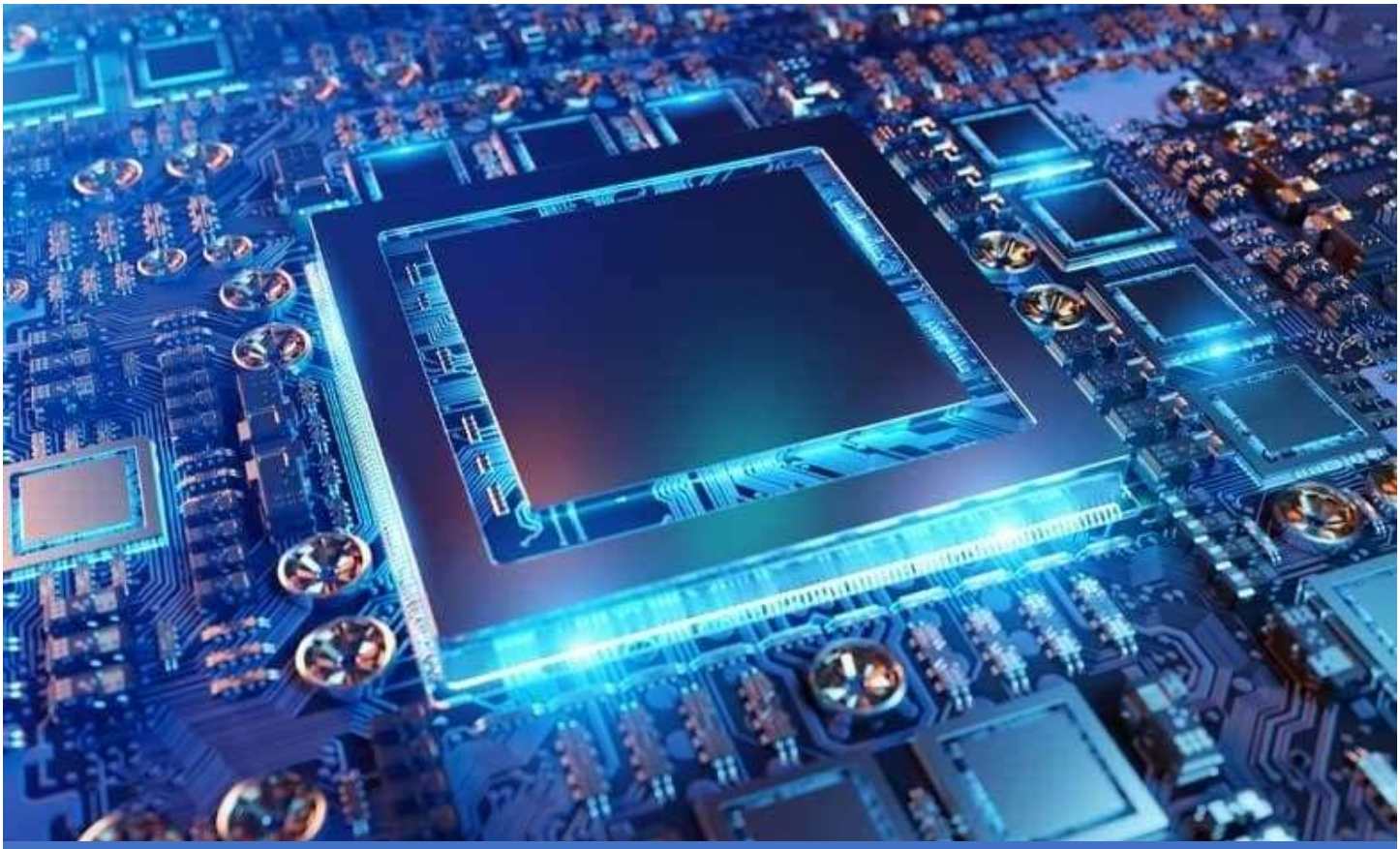
به نام خدا

عرفان تیموری و محمدرضا منعمیان

شماره دانشجویی : ۴۰۲۱۰۶۶۰۴ و۴۰۲۱۰۵۸۱۳



Practical2

Erfan Teymouri Mohammadreza Monemain

3/11/25

Computer Architecture

سوال اول)

زبان RTL زیر برای طراحی مدار در اختیار ما قرار گرفته شده است. ابتدا به تحلیل دستور ها و کارکرد آن می‌پردازیم.

$$\begin{aligned}
&\mathrm{R2} \leftarrow -R1 \\
&\mathrm{R2} \leftarrow R1 \& R2 \\
&\mathrm{R2} \leftarrow -R2 \\
&\mathrm{R2} \leftarrow R1 + R2
\end{aligned}$$

عملکرد مدار

در این مدار ابتدا با ۱ شدن سیگنال load مقدار in1 در رجیستر R1 ذخیره می‌شود.سپس مقدار ذخیره شده در R1 قرینه شده و در مرحله دوم با R1 and می‌شود. در نهایت پاسخ به دست آمده قرینه شده و با مقدار اولیه R1 جمع می‌شود.

هر کدام از این عملیات ها در یک پالس ساعت انجام می‌شود و بنابراین برای رسیدن به پاسخ نهایی نیاز به ۵ کلاک داریم.

اما عملکرد اصلی مدار چه خواهد بود؟

در دو مرحله اول یعنی قرینه کردن مقدار in1 و and کردن این قرینه با خودش نتیجه ای که حاصل می‌شود عدد۳۲ بیتی است که تنها بیت یک آن اولین بیت ۱ از سمت راست در in1 است. برای مثال اگر بیت i ام in1 اولین بیت ۱ از سمت آن باشد در پایان مرحله دوم در رجیستر R2 همه بیت ها به جز بیت i ام صفر است .

حالا در دو مرحله بعدی در عمل R1-R2 انجام می‌شود پس در پایان ۴ کلاک عددی که در R2 ذخیره میشود همان عدد R1 است با این تفاوت که اولین بیت سمت راست یک آن به صفر تبدیل شده است.

برای مثال اگر ورودی ۱۱۱ باشد خروجی ۱۱۰ خواهد شد.البته توجه داریم که ورودی و خروجی ۳۲ بیتی هستند و مقدار out1 خروجی نهایی است.

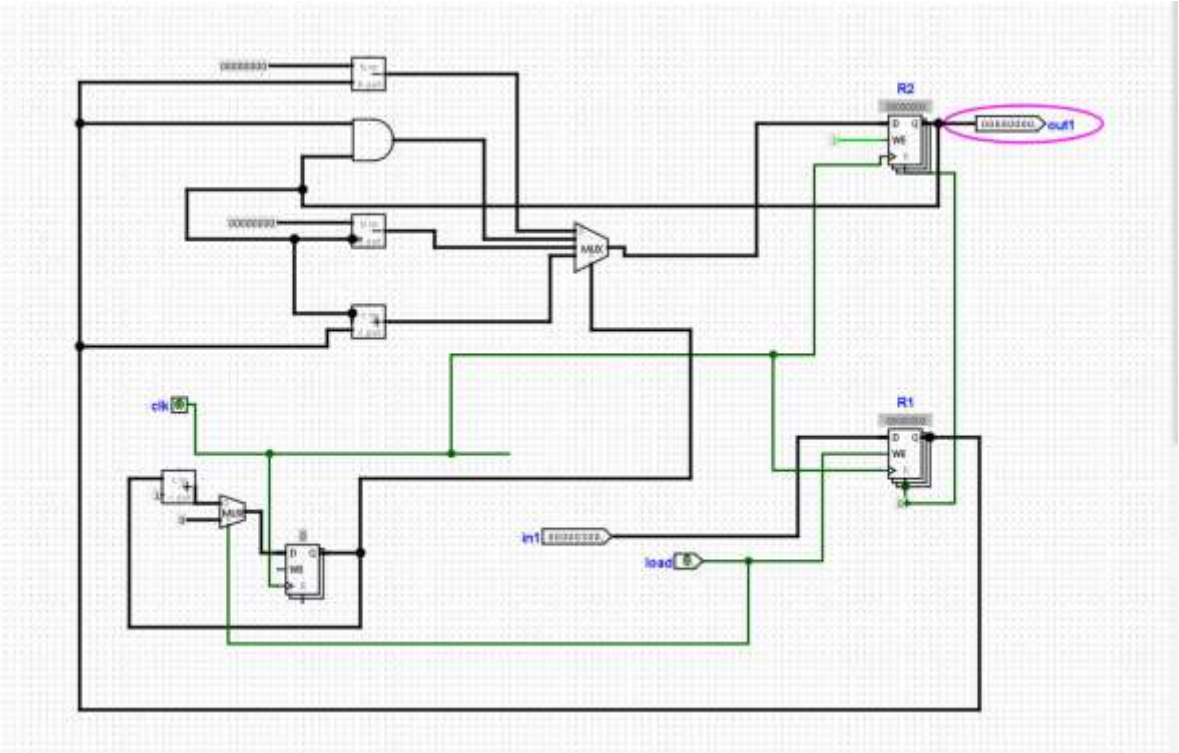
طراحی مدار

برای طراحی کردن مدار به خاطر اینکه دستورات باید در ۴ کلاک انجام شوند نیاز به یک شمارنده داریم که از ۰ تا ۳ را بشمرد و با استفاده از یک MUX اینکه چه چیزی وارد R2 شود انتخاب شود.

برای مثال اگر شمارنده صفر بود باید قرینه R1 اگر شمارنده برابر ۱ شد R1&R2 همچنین اگر شمارنده ۲ شد قرینه R2 و اگر شمارنده ۳ شد R1 + R2 باید درR2 ذخیره شود.

همچنین سیگنال load برای ورودی دادن به R1 استفاده می‌شود و وقتی دوباره صفر شد با کلاک زدن که عملا یک پین است که ۱ و صفر می‌شود عملیات ها پشت سر هم انجام می‌شوند.

توجه داریم که به خاطر درست نبودن شمارنده خود Logisim یک شمارنده دو بیتی با استفاده از MUX و رجیستر طراحی کریم.



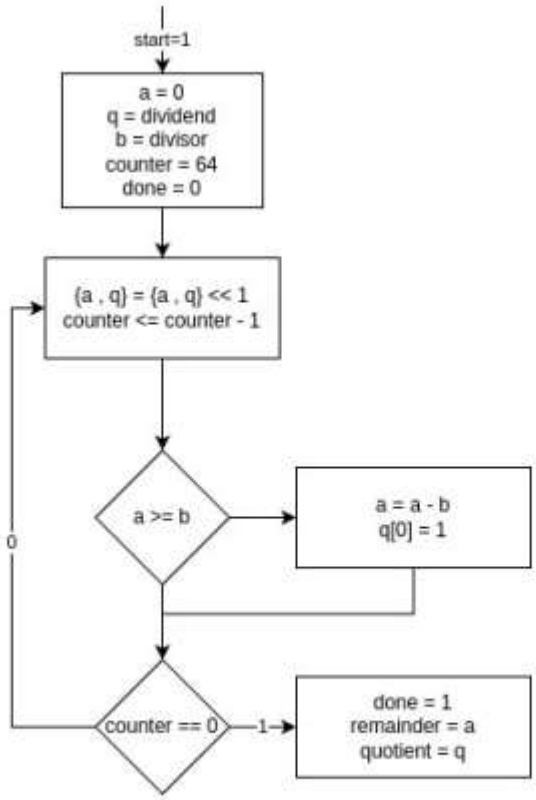
در ادامه خروجی تست مدار توسط وریلاگ داده شده را مشاهده می‌کنیم:

```
monitoring 0 0 10001011100101000000100100010111 11111111111111111111111111111111
monitoring 1 0 10001011100101000000100100010111 10001011100101000000100100010110
monitoring 0 1 01010000111101010111100010100001 10001011100101000000100100010110
monitoring 1 1 01010000111101010111100010100001 01110100011010111111011011101001
monitoring 0 0 01010000111101010111100010100001 01110100011010111111011011101001
monitoring 1 0 01010000111101010111100010100001 10101111000010101000011101011111
monitoring 0 0 01010000111101010111100010100001 10101111000010101000011101011111
monitoring 1 0 01010000111101010111100010100001 00000000000000000000000000000001
monitoring 0 0 01010000111101010111100010100001 00000000000000000000000000000001
monitoring 1 0 01010000111101010111100010100001 11111111111111111111111111111111
monitoring 0 0 01010000111101010111100010100001 11111111111111111111111111111111
monitoring 1 0 01010000111101010111100010100001 01010000111101010111100010100000
ACCEPTED
       200 /       200
monitoring 0 0 01010000111101010111100010100001 01010000111101010111100010100000
mohamandreza@mylove:~/CA/SUT_CA_4032_ProfAsadi_Judgement_System$
```

سوال دوم)

عملکرد مدار

در این سوال به پیاده سازی یک تقسیم کننده پرداختیم. البته نه به روش عادی تفریق متوالی بلکه با توجه به فلوچارت زیر:



شکل ۱: Flowchart تقسیم‌کننده

همانطور که مشاهده می‌کنیم در این روش ابتدا با ۱ شدن سیگنال start رجیستر های a,b,q مقدار دهی خواهند شد. توجه داریم که باید مقسوم را در رجیستر b و مقسوم علیه را در رجیستر q قرار داد و همچنین در ابتدا مقدار رجیستر a را برابر صفر کرد.

این الگوریتم تقسیم با کنار هم قرار دادن a,q و شیفت به چپ دادن آنها در هر مرحله عمل می‌کند. توجه داریم که در مداری که ما طراحی کردیم در ابتدا شمارنده نه با ۶۴ بلکه با ۳۲ مقدار دهی می‌شود به این خاطر که در پیاده سازی مدار هر حلقه در یک کلاک انجام شده و برای انجام تقسیم دو عدد ۳۲ بیتی ۳۲ بار شیفت کافی است پس باید ۳۲ بار کلاک زد. احتمالا منظور فلوچارت سوال این بوده که هر حلقه در دو کلاک انجام شود یعنی در یک کلاک شیفت به چپ و در یک کلاک بررسی شرط و نتیجه اش که ما این موارد را در یک کلاک انجام دادیم.

الگوریتم در هر مرحله a,q را کنار هم قرار داده حال این عدد ۶۴ بیتی را یکی به چپ شـیفت داده و در مرحله بعد اگر مقدار a از b بزرگتر بود a = a – b و همچنین بیت کم ارزش q برابر ۱ خواهد شـد و با ۳۲ بار انجام دادن این مراحل در آخر می‌توانیم مقدار باقی مانده تقسیم را در a و مقدار خارج قسمت را در رجیستر q مشاهده کنیم.

طراحی مدار

حال به نحوه طراحی مدار می‌پردازیم:

از یک شمارنده استفاده کردیم و وقتی سیگنال start یا done یک شد مقدار مناسب را به صورت انتخابی توسط یکMUX در رجیستر load می‌کنیم یعنی اگر start برابر ۱ شد مقدار ۳۲ و اگر done ۱ شد و start صفر بود که به معنی اتمام فرآیند تقسیم است صفر را در شمارنده load می‌کنیم.

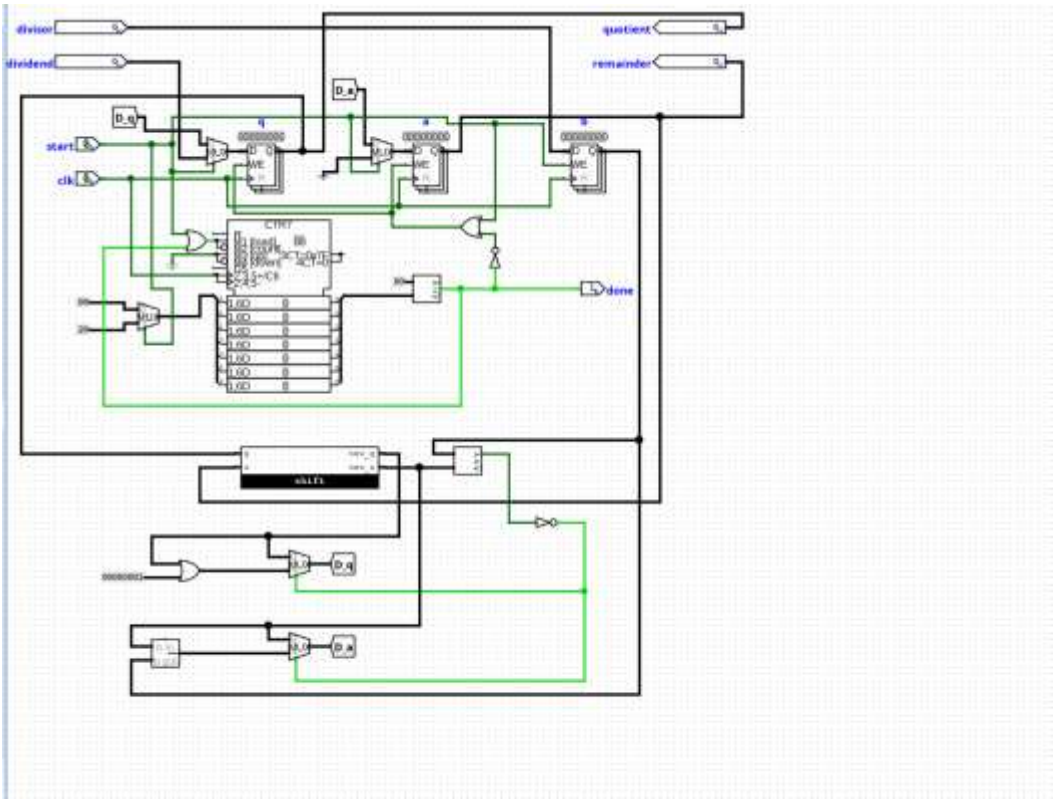
و همچنین با متصل کردن صفر به up/down شمارنده به صورت نزولی می‌شمارد.

در یک بلاک مجزا یک شیفتر ساختیم تا برای شیفت به چپ دادن a,q از آن استفاده کنیم بدین صورت که کم ارزش ترین بیت q را صفر کرده و هر بیت q جدید را از بیت قبلی اش می‌گیریم. همچنین بیت ۳۲ ام q نیز به عنوان کم ارزش ترین بیت a قرار داده و بقیه بیت های a را نیز یکی به چپ شیفت می‌دهیم.توجه داریم در ساخت این shifter از splitter بهره بردیم.

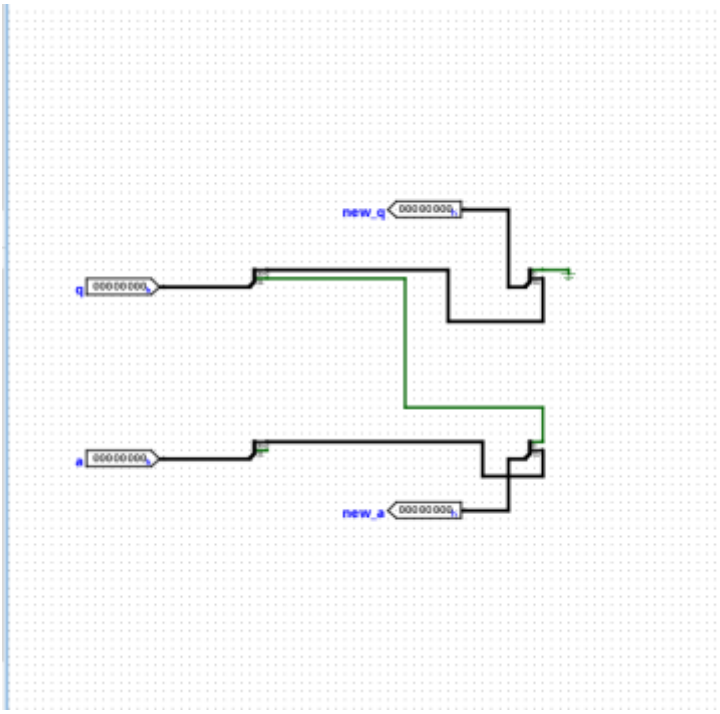
برای مقدار دهی رجیستر های a,q از دو MUX برای هر کدام استفاده کردیم بدین صورت که MUX اول نشان میدهد اگر سیگنال start فعال بود که مقدار صفر در رجیستر a و مقدار dividend در رجیستر q ذخیره شود. اما اگر سیگنال start فعال نبود ابتدا شیفت به چپ ها با شیفتر ساخته شده انجام می‌شود سپس اگر $a \geq b$ بود(که این را با استفاده از یک comparator بررسی می‌کنیم)مقدار a = a – b شده و همچنین برای ۱ کردن بیت آخر q نیز مقدار جدید آن را با 1 OR می‌کنیم. اگر هم که شرط برقرار نبود که مقادیر شیفت یافته شده توسط MUX انتخاب می‌شوند.

مقادیر خروجی نیز با استفاده از tree state buffer وقتی سیگنال done فعال شد از رجیستر های مناسب خوانده می‌شود.

در زیر مدار طراحی شده را مشاهده می‌کنیم:



همچنین shifter طراحی شده:



همچنین خروجی تست مدار را نیز مشاهده می‌کنیم:

```
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Circuit "main" passed DRC check
[main] INFO com.cburch.logisim.fpga.gui.Reporter - The Board ALCHITRY_AU_IO has:
[main] INFO com.cburch.logisim.fpga.gui.Reporter - 6 Button(s)
[main] INFO com.cburch.logisim.fpga.gui.Reporter - 3 DIPSwitch(s)
[main] INFO com.cburch.logisim.fpga.gui.Reporter - 32 Led(s)
[main] INFO com.cburch.logisim.fpga.gui.Reporter - 1 SevenSegmentScanning(s)
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] WARN com.cburch.logisim.fpga.gui.Reporter - Component "Counter" in circuit
[main] WARN com.cburch.logisim.fpga.gui.Reporter - Component "Register" in circuit
[main] WARN com.cburch.logisim.fpga.gui.Reporter - Component "Register" in circuit
[main] WARN com.cburch.logisim.fpga.gui.Reporter - Component "Register" in circuit
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating HDL file: /home/mohamar
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating script file: /home/moh
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating script file: /home/moh
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating script file: /home/moh
[main] INFO com.cburch.logisim.fpga.gui.Reporter - Creating script file: /home/moh

HW2/tb2.v /home/mohamamdreza/logisim_evolution_workspace/Q2p.circ.tmp
ACCEPTED
        200 /        200
mohamamdreza@mylove:~/CA/SUT_CA_4032_ProfAsadi_Judgement_System$
```