تمرین دوم - عملی

سوال ۱)

۱. در این سوال می‌خواهیم که یک کامپیوتر ساده را به کمک دانش RTL طراحی کنیم. کامپیوتری که طراحی می‌کنید باید ویژگی‌های زیر را داشته باشد:

- ۴ ثبات ۸ بیتی داشته باشد
- چهار عملیات زیر را بتواند انجام دهد:
  ۱. جمع یک ثبات با ثبات دیگر
  ۲. زیاد کردن عدد یک ثبات به اندازه‌ی ۱ واحد
  ۳. کم کردن عدد یک ثبات به اندازه‌ی ۱ واحد
  ۴. ریختن یک ثبات در یک ثبات دیگر
- به صورت point-to-point طراحی شده باشد.

پیاده‌سازی را به صورت شماتیک در نرم‌افزار Quartus انجام دهید و سپس به سوالات زیر پاسخ دهید:

آ) طراحی شما چند سیگنال کنترلی دارد؟

ب) برای هر یک از عملیات ذکر شده بگویید که سیگنال‌های شما باید چه مقادیری داشته باشند.

ج) به زبان RTL مخصوص سیستمی که طراحی کردید یک برنامه بنویسید که دنباله‌ی فیبوناچی را تا عدد ۴ام حساب کند. فرض کنید که مقدار اولیه تمام ثبات‌ها ۰ است.

جواب سه سوال بالا را در یک فایل PDF تایپ کنید و سپس به همراه فایل مدار خود در سامانه CW آپلود کنید. همچنین در گزارش خود تمامی دستوراتی که تعریف کردید را تست کنید و نتیجه‌ی تست آن را بنویسید. دقت کنید که لازم نیست که به mux و adder را خودتان از صفر بسازید و می‌توانید از قطعات آماده استفاده کنید.

آ) در پیاده‌سازی‌ای که مطابق طراحی point-to-point انجام داده‌ایم، از ۴ data bus ۳ بیتی (۱۲ سیگنال کنترلی) برای سلکت کردن مقادیر ورودی muxها استفاده کرده‌ایم. همچنین برای مشخص کردن رجیسترهای مربوطه برای عملیات add و inc و dec از دو mux استفاده شده‌است و هر mux برای سلکت کردن ورودی مربوطه، از یک data bus ۳ بیتی استفاده می‌کند (۶ سیگنال کنترلی). پس در مجموع از ۱۸ سیگنال کنترلی یک‌بیتی استفاده کرده‌ایم.

البته در ادامه می‌توان آن‌ها را به طور اتوماتیک توسط control unit مشخص کرد، اما اینجا برای نشان دادن کارکرد data path طراحی‌شده، سیگنال‌ها به صورت دستی ست می‌شوند.

ب) ۴ دستور بیان‌شده در صورت سوال را به ترتیب add و inc و dec و mov می‌نامیم. از طرفی muxهای متعلق به رجیسترها به ترتیب R0 و R1 و R2 و R3 و result را سلکت می‌کنند، که result همان نتیجه‌ی ALU است. همچنین muxهای مشخص‌کننده‌ی ورودی‌های ALU به ترتیب R0 و R1 و R2 و R3 و 1 و 1- را انتخاب می‌کنند که 1 و 1- برای انجام عملیات inc و dec استفاده می‌شوند.

- mov: برای اجرای این عملیات کافی است mux متعلق به رجیستر مقصد، رجیستر مبدا را انتخاب کند.

- inc: برای اجرای این عملیات کافی است mux متعلق به رجیستر انتخاب‌شده result ر انتخاب کند و muxهای ALU به ترتیب رجیستر انتخاب‌شده و 1 را انتخاب کنند.
- dec: برای اجرای این عملیات کافی است mux متعلق به رجیستر انتخاب‌شده result ر انتخاب کند و muxهای ALU به ترتیب رجیستر انتخاب‌شده و 1- را انتخاب کنند.
- add: برای اجرای این عملیات کافی است mux متعلق به رجیستر مقصد result را انتخاب کند و muxهای ALU رجیسترهای عملوند را انتخاب کنند.

ج) رجیسترها را R0 و R1 و R2 و R3 می‌نامیم که مقدار اولیه‌ی آن‌ها صفر است و با دستورات بالا این طور عمل می‌کنیم:

$$R1 \leftarrow R1 + 1$$

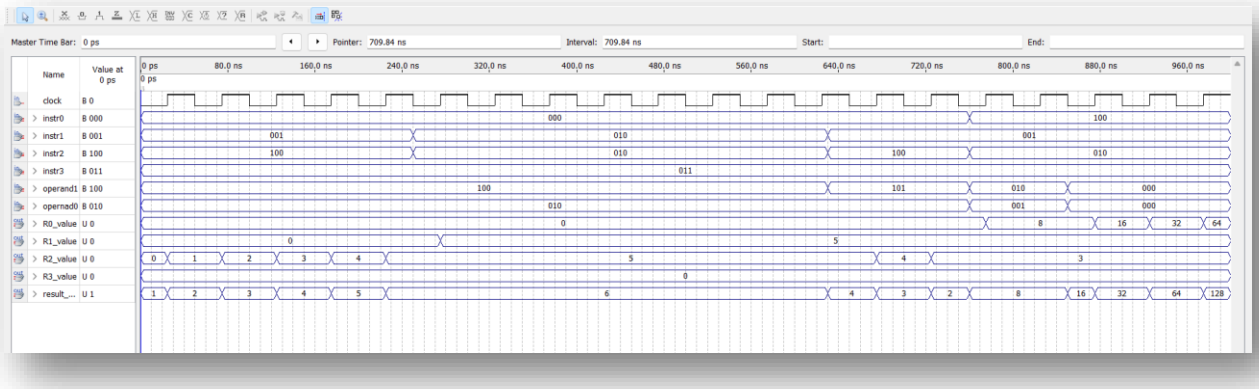
$$R2 \leftarrow R0 + R1$$

$$R3 \leftarrow R1 + R2$$

توضیح: ابتدا همه‌ی رجیسترها صفر هستند. برای ساخت جملات پایه، ابتدا R1 را یکی زیاد می‌کنیم. سپس با جمع R0 و R1 و ذخیره‌ی نتیجه در R2 جمله‌ی سوم را می‌سازیم. در نهایت نیز با جمع R1 و R2 و ذخیره‌ی نتیجه در R3 جمله‌ی چهارم را می‌سازیم. حالا جملات اول تا چهارم دنباله‌ی فیبوناچی با مقادیر 0 و 1 و 1 و 2 در رجیسترهای R0 تا R3 ذخیره شده‌اند.

نمونه (توضیح دستورات):

همه‌ی دستورات در این نمونه گنجانده شده‌اند:



در این نمونه، ابتدا مقدار رجیستر R2 ۵ بار یکی زیاد شده‌است (inc) و از صفر به ۵ رسیده‌است، سپس مقدار رجیستر R2 در رجیستر R1 ریخته شده‌است (mov). پس از آن مقدار رجیستر R2 ۲ بار یکی کاهش یافته‌است (dec) و از ۵ به ۳ رسیده‌است. در ادامه مجموع مقادیر رجیسترهای R1 و R2 (۸) در رجیستر R0 ریخته شده‌است (add). در نهایت نیز چند مرتبه مقدار رجیستر R0 با خودش جمع شده‌است و در خودش ریخته شده‌است و مقدار ۱۶، ۳۲ و ۶۴ را تولید و ذخیره کرده‌است.