به نام خدا

گزارش کار آزمایشگاه طراحی سیستم‌های دیجیتال

محمدحسین تاج‌الدینی ۹۴۱۷۰۸۲۸ آزمایش چهارم – توصیف رفتاری
محمدحسین حاجی سید سلیمان ۹۸۱۰۵۶۷۸
محمدعلی محمدخانی ۹۸۱۰۲۲۵۱

بخش صفر – خلاصه کار جلسه

باید یک پشته (Stack) با ظرفیت ۸ و پهنای ۴ بیت بسازیم که قابلیت ریست شدن هم داشته باشد. توصیف رفتاری در حالت کلی به فرم یک بلوک حساسیت و یک سری شرط و مقدار دهی در آن است؛ بر خلاف جریان داده که assign می‌کردیم.

بخش یک – پشته ۴×۸ بیتی

بخش ۱.۱ – پیدا کردن منطق مدار

برای ساخت پشته باید از یک آرایه یک بعدی به اندازه ۸ استفاده کنیم که هر عنصر آن ۴ بیت داده دارد. باید بتوانیم به شکل FILO داده را بنویسیم و بازخوانی کنیم، پس باید یک سیگنال مانند index برای آدرس هم داشته باشیم. در ادامه نکات مهم منطق مدار و توضیحات آنها بیان شده:

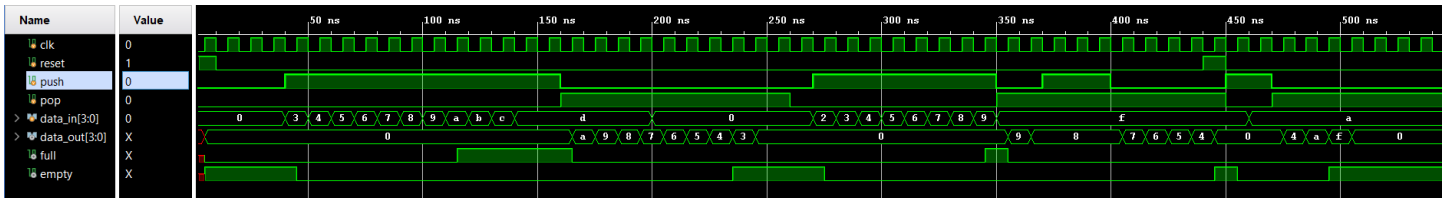
۱. برای مقادیر push و pop باید شرط $\sim(A \wedge B)$ برقرار باشد که این وظیفه بر عهده کاربر ماژول است، پس حالاتی که هم push داشته باشیم هم pop را برای ماژول به منزله حالت idle می‌گیریم یعنی انگار هر دو آنها صفر بوده‌اند.
۲. روش best practice انجام دادن تمامی مراحل در یک بلوک always است. پس از آنجا که کد ما پالس ساعت دارد، تمام عملیات آن در یک بلوک حساسیت به لبه بالا رونده پالس ساعت انجام می‌شود. نکته این است که حالت ها در if بررسی شوند و برای if ها else داشته باشیم که منطق دچار ایراد نشود.
۳. بدیهی است که قبل از شروع به استفاده، کاربر باید یک مرتبه مدار را ریست کند تا برای کار آماده شود. در این پروسه آدرس مقدار ۰ می‌گیرد، سیگنال خالی روشن شده و سیگنال پر خاموش می‌شود.
۴. push کردن تنها در صورتی ممکن است که ماژول پر نباشد. در این صورت مقدار ورودی در خانه به آدرس index نوشته می‌شود و آدرس یکی بیشتر می‌شود. بدیهی است که در صورت push شدن یک مقدار، دیگر ماژول خالی نیست، پس empty صفر می‌شود. این روش باعث می‌شود محدودیت در مقادیر ورودی نداشته باشیم و خالی و پر بودن جدا از مقادیر ذخیره شده باشند.
۵. اگر مقدار index بیشتر از ۶ شود دیگر نباید ورودی دریافت کنیم، حتی اگر سیگنال push روشن باشد، چون حافظه پر شده است.
۶. منطق به نحوی ساخته شده که مقدار index کمتر از ۰ نخواهد شد. پس خالی بودن ماژول زمانی است که index برابر ۰ باشد و empty هم روشن باشد. این برای اطمینان از pop شدن مقداری که در آدرس ۰ داشتیم است.
۷. pop کردن سه حالت دارد:
۷٫۱. در صورتی که مدار پر باشد، باید آدرس اصلاح شود به ۶ (یکی کمتر از آخرین آدرس ماژول) و از آنجا که عملیات موازی است، مقدار آدرس ۷ (آخرین آدرس ماژول) به خروجی داده شود.
۷٫۲. در صورتی که نه پر است و نه خالی، مقدار pop می‌شود و آدرس یکی کم می‌شود.
۷٫۳. در صورتی که به اولین خانه برسیم، باید اول مقدار آن pop شود تا بتوان مدار را خالی اعلام کرد. از آنجایی که اولین باری که به این خانه برسیم مدار empty نبوده، شرط pop شدن را خالی نبودن می‌گذاریم و با pop شدن مدار را empty می‌گیریم.
۸. اگر مدار خالی باشد و pop کنیم، خروجی پیش فرض ۰ می‌گیریم.
۹. خروجی مدار ثابت است و با پالس ساعت و در صورتی که خروجی بخواهیم تغییر می‌کند.

بخش ۱.۲ – ساخت مدار، شبیه سازی و سنتز

از آنجا که کد ماژول و تست بنچ با کامنت گذاری کامل پیوست شده، از تکرار کد پرهیز می‌کنم. برای ساخت دقیقا طبق موارد بخش منطق عمل شد و سپس در حالت‌های زیر مدار تست شد:

- مدار پر شود و سیگنال پر بگیریم.
- در صورت پر بودن آخرین مقدار ثابت مانده و ورودی دریافت نکنیم.
- مدار خالی شود و سیگنال خالی بگیریم.
- با خالی شدن خروجی پیش‌فرض ۰ بگیریم.
- مدار مجددا پر شود.
- مقدار pop کنیم تا کمی خالی شود.
- سیگنال pop و push همزمان روشن شود و صبر کنیم. مدار نباید پر شود.
- سیگنال push را خاموش کنیم تا pop کنیم. نباید در زمانی که pop و push همزمان روشن بودند ورودی یا خروجی گرفته باشیم. چند مقدار دیگر pop کنیم تا این مورد بررسی شود.
- چند مقدار push کنیم که مدار نه پر باشد نه خالی.
- ریست کنیم. باید خالی شود.
- دو مقدار push کنیم.
- سه مقدار pop کنیم. باید دو مقدار داده شده و خروجی پیش فرض بگیریم.

تمامی موارد با موفقیت پاس شد. برای نوشتن این تست بنچ از حلقه for استفاده شد که کد شلوغ نشود. فرم موج (فایل در پیوست):



در نهایت سنتز پذیری را بررسی می‌کنیم که طبق پیام زیر موفقیت آمیز بوده است: