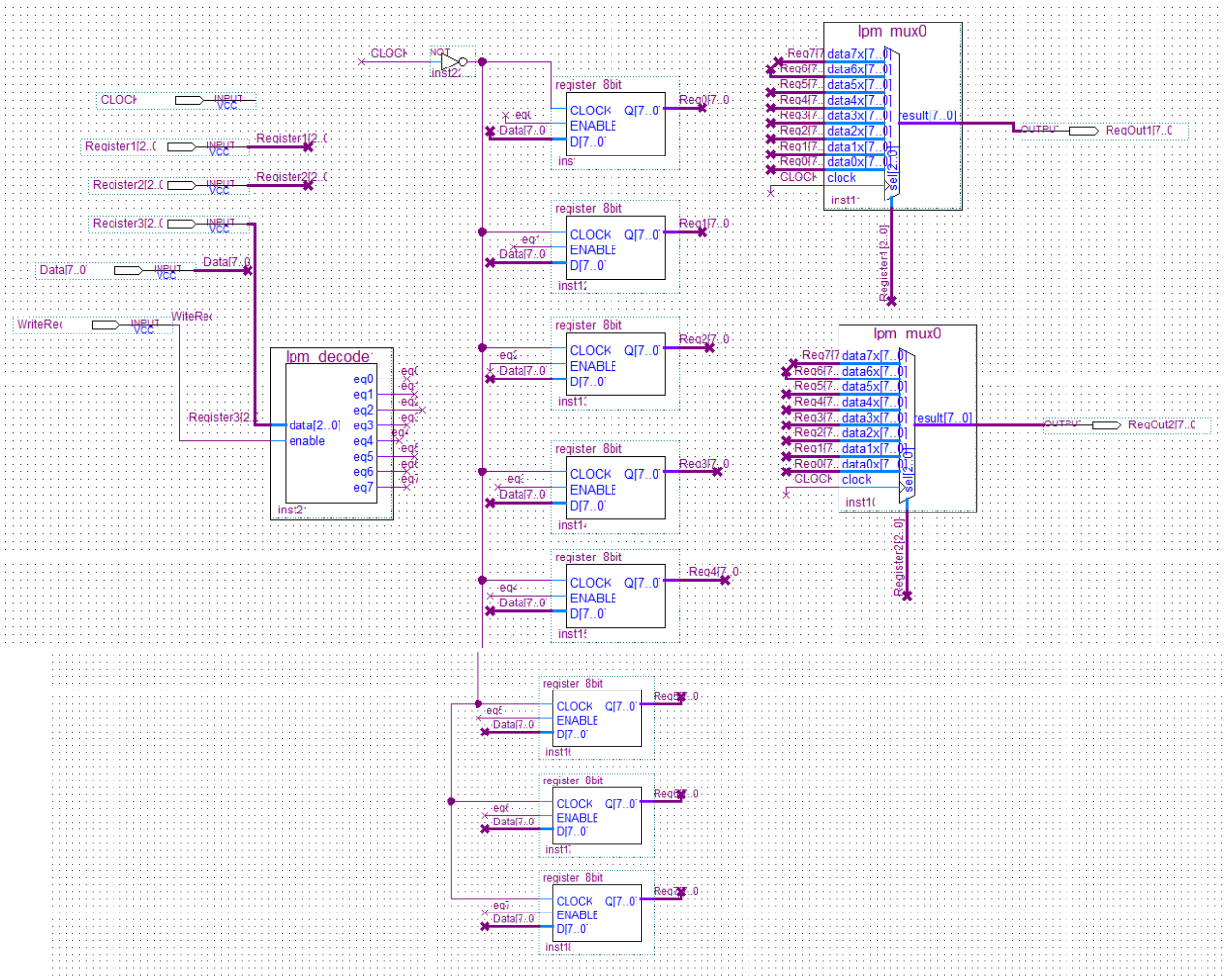
تمرین عملی معماری کامپیوتر دکتر اسدی، سری چهارم

آریا همتی ۴۰۱۱۱۰۵۲۳ | فرزام کوهی ۴۰۱۱۰۶۴۰۳

برای این تمرین از ۲ بخش جدا گونه (register files, memory) درست شده است.

ابتدا از register files شروع کرده و توضیح و تست میکنیم.

RegisterFile:



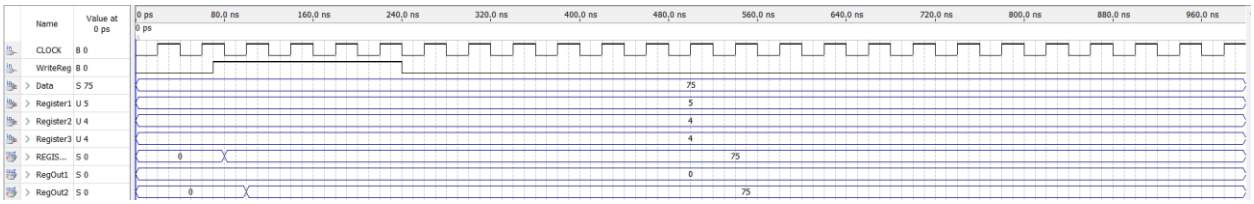
از ۸ رجیستر ۸ بیتی که در تمرین های قبلی ساخته استفاده کرده تا رجیسترفایل را بسازیم.

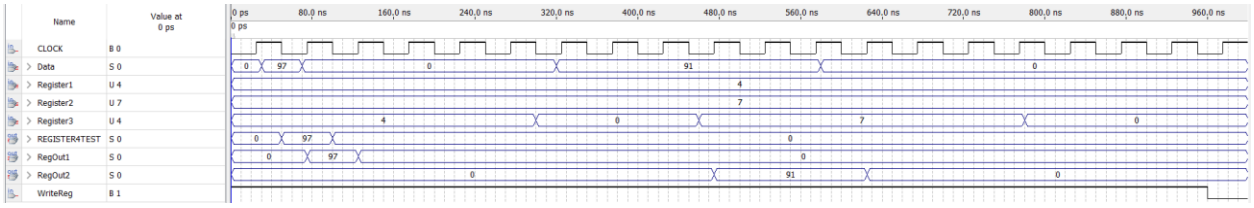
برای مشخص کردن RegisterOut , 2RegisterOut1 از 2 تا MUX با 8 بیت ورودی و خروجی استفاده کرده که در select شان از input های Register , 2Register1 استفاده کرده ایم.

برای مشخص کردن رجیستر مقصد که به عنوان registerWrite استفاده می‌شود از سیگنال 3register که ۳ بیت است استفاده کرده‌ایم. Data نیز با توجه به 3register و یک decoder که ورودی enable آن WriteReg است برای write کردن در رجیستر مقصد استفاده شده است.

طبق waveform های زیر همانطور که در سوال خواسته شده بود read در لبه ی بالای clock و write در لبه ی پایین صورت میگیرد. (برای رعایت انجام write در لبه پایین clock، سیگنال clock، نات شده است.)

Testing of RegisterFile



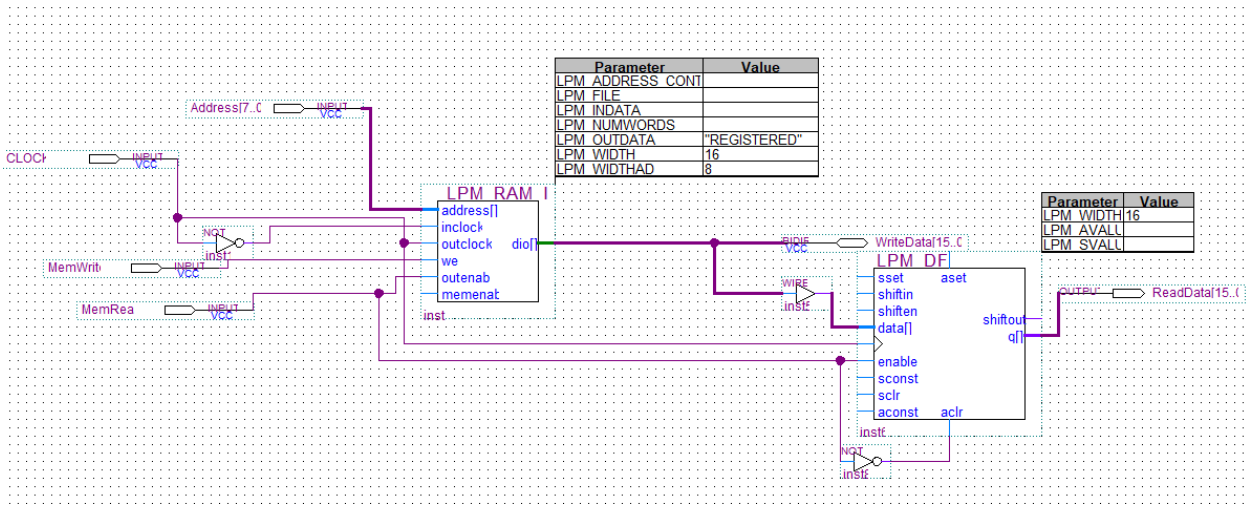


نکته: در دو مثال بالا، برای نمایش دادن زمان WRITE که در لبه پایین clock است، به طور آزمایشی مقدار رجیستر4 به صورت یک pin مجزا نمایش داده شده است. زمان READ نیز در لبه بالای clock است.

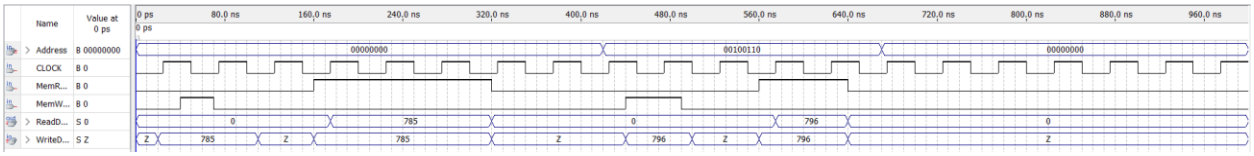
در پارت بعدی که همان memory است ما از lpm-ram-io استفاده کرده و برای ورودی های آن را به ترتیب ۸ بیت آدرس خانه ۱۶ بیتی در نظر گرفته شده است.

حال برای اینکه read , write ها بر اساس داک گفته شده بر روی لبه ی بالارونده و پایین رونده انجام شوند. ما از ۲ تریک استفاده میکنیم. اولین این است که outclock را کلاک عادی و inclock را not شده ی کلاک می‌دهیم. سپس برای خروجی ReadData از یک DFlipFlop استفاده کرده که با تغییر MemRead از یک به صفر، دیتا را clear بکند تا دیگر Read صورت نگیرد و مقدار ReadData برابر صفر شود.

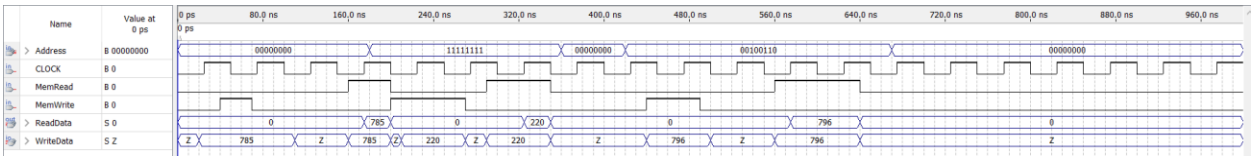
DataMemory:



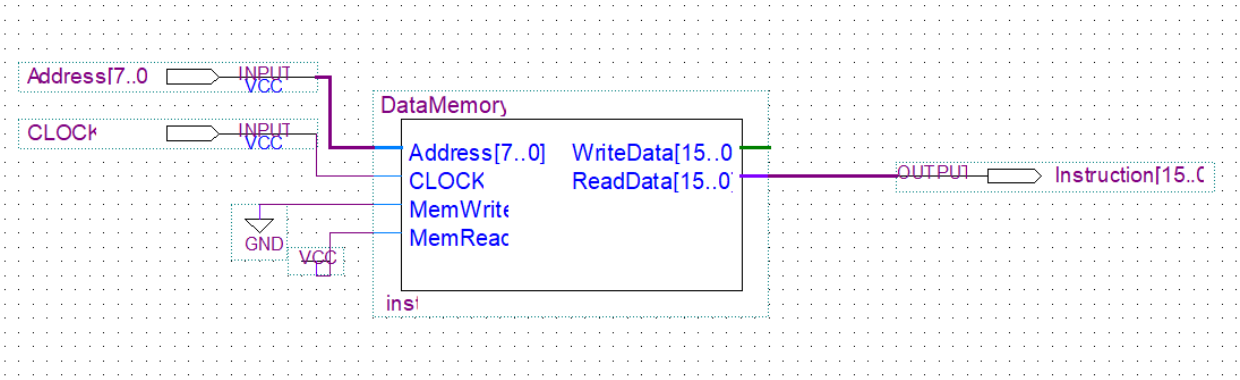
Testing of DataMemory:



همانطور که در عکس معلوم است: ابتدا در آدرس صفر عملیات ها صورت میگیرد و ابتدا یک write صورت میگیرد (با ۱ کردن سیگنال MemWrite) سپس با لبه ی پایین این تغییر اعمال شده. پس از آن عملیات read صورت گرفته و عدد ۷۸۵ چاپ میشود (با لبه ی بالا trigger میشود) و تا لحظه ای که MemRead برابر 1 است، همچنان مقدار نمایش داده می‌شود. سپس آدرس عوض شده و به 00100110 و همین عملیات ها انجام شده ولی تنها تفاوت عوض شدن عدد خواسته شده به ۷۹۶ است. در مثال پایین نیز به ترتیب به خانه‌های 00000000 و 11111111 و 00100110 حافظه مقادیر 785، 220 و 796 اختصاص داده شده است. (مشاهده می‌شود که تمام Read ها در لبه بالا رونده و تمام Write ها در لبه پایین رونده صورت می‌گیرد)



برای instruction memory هم از مدار قسمت قبل (DataMemory) استفاده کرده تا operation های مورد نیاز را پیاده سازی بکنیم. سیگنال های memread همواره به ۱ , memwrite همواره ۰ است و از آن به صورت ROM استفاده میشود. (چون تنها قصد Read را از instruction memory داریم). شکل Instruction memory در صفحه بعد قابل رویت است.



InstructionMemory