علی آراسته ۹۶۱۰۱۱۶۵

گزارش کار فاز چهارم پروژه ساختار کامپیوتر و میکروپروسسور

خواسته‌های پروژه:

۱- پس از وصل کردن دو واحد Processor و Memory System در ماژول Top، با استفاده از تست بنچ TopTB عملکرد این ماژول را مورد بررسی قرار می‌دهیم.

با توجه به نتیجه simulation، ۲۸ Hit رخ می‌دهد، که با Correct Output داده شده مطابقت دارد.

```
The Number of Hits is:          28
** Note: $stop    : A:/modeltech64_10.4c/Cache/TopTb.v(45)
   Time: 1016 ns  Iteration: 0  Instance: /TopTb
Break in Module TopTb at A:/modeltech64_10.4c/Cache/TopTb.v line 45
```

۲- با توجه به صورت پروژه، ازآن جایی که اندازه Cache، ۱۶ بایت و اندازه هر بلوک ۱ بایت است، این Cache دارای ۱۶ بلوک است و چون از نوع 2 way set associative طراحی شده است، دارای ۸ set است. لذا آن را به صورت زیر طراحی می‌کنیم:

U	D1	V1	Tag1	Data1	D0	V0	Tag0	Data0	set
									7
									6
									5
									4
									3
									2
									1
									0

برای تعیین مقدار Hit از دستور زیر استفاده می‌کنیم.

```
assign Hit = ((Address[5:3] == Tag0) && (ValidityBit0)) || ((Address[5:3] == Tag1) && (ValidityBit1));
```

نحوه کار این دستور بدین صورت است که اگر Address[5:3] با Tag یکی از way ها برابر باشد و به طور همزمان Validity Bit مربوط به همان way نیز یک باشد، مقدار Hit یک می‌شود.

برای تعیین مقدار ReadEn از دستور زیر استفاده می‌کنیم.

```
assign ReadEn = ((!RWB) && (!Hit));
```

نحوه کار این دستور بدین صورت است که اگر Processor در حال خواندن یک داده باشد و داده مورد نظر Hit نشود (Miss رخ دهد)، مقدار ReadEn یک می‌شود تا آن داده از RAM خوانده شود.

برای تعیین مقدار WriteEn از دستور زیر استفاده می‌کنیم.

```
assign WriteEn = ((!Hit) && (((!LRUBit) && DirtyBit0 && ValidityBit0) || (LRUBit && DirtyBit1 && ValidityBit1)));
```

نحوه کار این دستور بدین صورت است که اگر داده‌ای Hit نشود (Miss رخ دهد) و Validity Bit هر دو way مربوط به set آن، یک باشد و Dirty Bit مربوط به دستور قدیمی‌تر (بر اساس LUR Bit) یک باشد، مقدار WriteEn یک می‌شود. زیرا این اتفاق بدین معنا است که داده جدید باید در مکان داده‌ای نوشته شود که تغییرات آن در RAM ثبت نشده است و در نتیجه این داده باید در RAM نوشته شود.

نحوه کار کلی Cache بدین صورت است که وقتی داده‌ای از RAM خوانده می‌شود یا داده‌ای قرار است در RAM نوشته شود، در صورتی که این داده Hit نشود (Miss رخ دهد) به ترتیب زیر در Cache نیز نوشته می‌شود:

۱- اگر Validity Bit مربوط به یکی از way ها صفر باشد، داده در آن way نوشته می‌شود و Validity Bit مربوط به آن way یک می‌گردد. اگر داده قرار است در RAM نوشته شود، Dirty Bit مربوط به آن way نیز یک می‌شود.

۲- اگر Validity Bit مربوط به هر دو way یک باشد، ولی Address[5:3] با tag یکی از way ها یکسان باشد، داده در آن way نوشته می‌شود (داده موجود آپدیت می‌شود)

۳- اگر Validity Bit مربوط به هر دو way یک باشد و Address[5:3] با tag هیچ یک از way ها یکسان نباشد، داده در way ای نوشته می‌شود که داده آن قدیمی‌تر است (تصمیم‌گیری بر اساس LRU Bit انجام می‌گیرد). اگر Dirty Bit مربوط به way ای که داده قرار است در آن نوشته شود، یک باشد، ابتدا داده قدیمی در RAM و سپس داده جدید در Cache نوشته می‌شود.

برای پیاده‌سازی سیاست write_Back برای هر یک از way های مربوط به هر set یک بیت Dirty Bit در نظر می‌گیریم. عملکرد این بیت بدین صورت است که هرگاه داده‌ای بخواهد از Processor روی RAM نوشته شود، این داده فقط در Cache نوشته می‌شود و Dirty Bit مربوط به way ای که داده در آن نوشته می‌شود، یک می‌گردد تا مشخص شود که داده‌ای که در Cache وجود دارد با داده موجود در RAM متفاوت است.

برای پیاده‌سازی سیاست جایگزینی LRU برای هر set یک بیت U در نظر می‌گیریم تا داده‌ای که قدیمی‌تر است، مشخص شود.

۳- می‌توان برای کاهش تعداد compulsory miss اندازه هر بلوک و conflict miss تعداد way ها را افزایش داد.