# به نام خالق یکتا

پروژه میانی درس معماری کامپیوتر  نیمسال دوم سال تحصیلی ۹۴-۹۳

***موعد تحویل: ۲۵ اسفندماه ۹۳***

هدف از این پروژه، طراحی و پیاده‌سازی یک حافظه نهان[1] ساده برای یک حافظه اصلی[2]، با استفاده از زبان Verilog و یا VHDL می‌باشد. انتخاب نرم‌افزار بر عهده دانشجویان است. می‌توانید از نرم‌افزارهای Modelsim و یا Xilinx استفاده کنید.

ویژگی‌های حافظه اصلی:

- حجم حافظه: ۴ کیلوبایت (۲۰۴۸ سطر ۲ بایتی)
- نیازی به پیاده‌سازی حافظه اصلی نیست (خط نوشتن، خواندن، داده و ...)
- حافظه اصلی را با داده‌های تصادفی ثابت پرکنید.

ویژگی‌های حافظه نهان:

- حجم حافظه: ۱ کیلوبایت (۲۵۶ سطر، هر سطر ۴ کلمه ۲ بایتی)
- حافظه نهان موردنظر از نوع نگاشت مستقیم[3] است.
  - **امتیازی ۱**: در صورت پیاده‌سازی حافظه نهان مجموعه انجمنی[4] در حالت k=2، نمره مثبت در نظر گرفته خواهد شد (تا سقف ۰/۵ نمره).
- سیگنال‌های ورودی/خروجی و عملکرد آن‌ها در *جدول ۱-سیگنال‌های ورودی/خروجی* آمده است.

---

[1] Cache

[2] Main memory

[3] Direct mapped cache

[4] K-way set associative cache (KWSA)

جدول ۱-سیگنال‌های ورودی/خروجی

| نام سیگنال | خروجی / ورودی | پهنای بیتی | توضیحات |
|---|---|---|---|
| enable | ورودی | ۱ | حالت آماده‌به‌کار. از نوع active high. اگر در حالت low قرار بگیرد، سیگنال‌های comp و write بی‌تأثیر و مقدار تمام خروجی‌ها ۰ خواهد بود. |
| index | ورودی | ۸ | بیت‌های آدرس که برای مشخص کردن شماره سطر حافظه نهان استفاده می‌شوند. |
| word | ورودی | ۲ (۳) | معین می‌کند به کدام کلمه (word) در سطر مشخص‌شده توسط index در حافظه نهان دسترسی خواهیم داشت. |
| comp | ورودی | ۱ | سیگنال مقایسه (compare). اگر comp=1، حافظه نهان، مقدار سیگنال tag_in و مقدار tag خط انتخاب‌شده را مقایسه می‌کند. اگر hit رخ‌داده باشد data از مکان موردنظر از حافظه نهان *خوانده*[5] یا در آن *نوشته*[6] می‌شود؛ اما در صورت رخداد miss *نوشتن* انجام نمی‌شود. |
| write | ورودی | ۱ | اگر در لبه بالارونده کلاک مقدار ۱ داشته باشد، داده تعیین‌شده توسط index و word در محل تعیین‌شده توسط قسمت tag از سیگنالindex *نوشته* خواهد شد (مشروط بر comp=0) |
| tag_in | ورودی | ۵ | اگرcomp=1 ، برای تشخیص رخداد hit، این سیگنال با tag ذخیره‌شده در حافظه مقایسه می‌شود. اگر comp=0 و write=1 این سیگنال در بخش tag آرایه نوشته خواهد شد. |
| data_in | ورودی | ۱۶ | داده‌ای که در زمان نوشتن در محل تعیین‌شده توسط index و word نوشته خواهد شد. |
| valid_in | ورودی | ۱ | در زمان نوشتن درصورتی‌که comp=0، این بیت نشان‌دهنده مقداری است که باید در بیت valid آن سطر قرار داده شود. |
| clk | ورودی | ۱ | سیگنال کلاک. حساس به لبه بالارونده. |
| rst | ورودی | ۱ | اگر در لبه بالارونده کلاک، مقدار سیگنال rst، برابر ۱ باشد، بیت valid تمام سطرها مقدار false خواهد گرفت. (باقیمانده فضای initialize نشده حافظه نهان احتمالاً مقدار X(z high )خواهد داشت.) |
| hit | خروجی | ۱ | هنگام مقایسه، اگر مقدار tag خانه مشخص‌شده با index، با مقدار tag_in برابر باشد، مقدار این سیگنال high خواهد شد. |

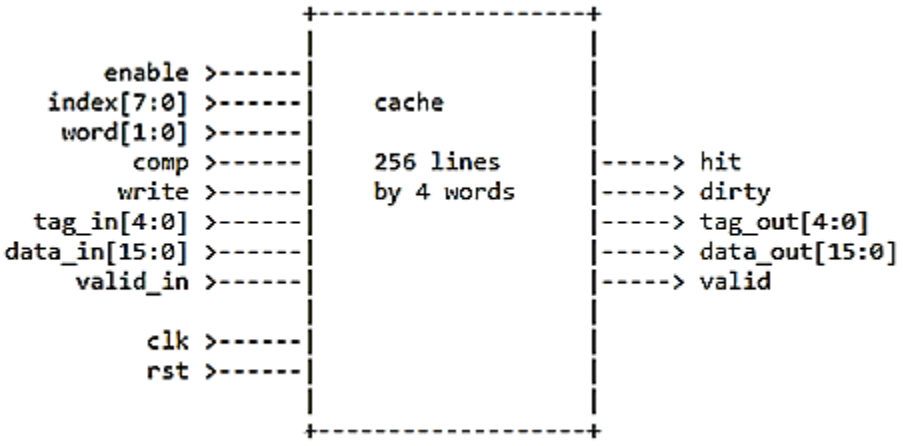
[5] Read

[6] Write

| | | | |
|---|---|---|---|
| dirty | خروجی | ۱ | این بیت نشان‌دهنده این است که آیا بر داده مستقر در این خانه از حافظه نهان تغییری اعمال‌شده است یا خیر. (در یک عمل نوشتن در حالت comp=1، حافظه نهان بیت dirty را برابر یک قرار می‌دهد و در عمل نوشتن در حالت comp=0 حافظه نهان بیت dirty را برابر با ۰ قرار می‌دهد. صفحه ۴) |
| tag_out | خروجی | ۵ | زمانی که مقدار سیگنال write برابر ۰ باشد، مقدار tag موجود در سطر تعیین‌شده توسط index در این خروجی قرار می‌گیرد. |
| data_out | خروجی | ۱۶ | زمانی که مقدار سیگنال write برابر ۰ باشد، data موجود در خانه مشخص‌شده توسط سیگنال‌های index و word در این خروجی قرار داده می‌شود. |
| valid | خروجی | ۱ | در طول یک عمل *خواندن*، نشان‌دهنده اعتبار برای خط مشخص‌شده است. |

## ۱) ساختار حافظه نهان:

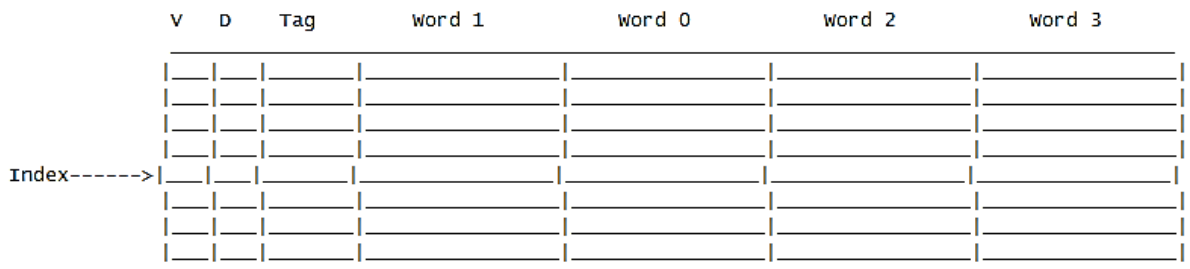
حافظه نهان شامل ۲۵۶ خط و هر خط شامل یک بیت معتبر[7]، یک بیت کثیف[8]، یک برچسب ۵ بیتی و چهار کلمه ۱۶ بیتی است.




*شکل ۱- شمای کلی حافظه نهان از منظر ورودی/خروجی*

[7] Valid bit

[8] Dirty bit




*شکل ۲- شمای منطقی حافظه نهان*

## ۲) عملکرد حافظه نهان نگاشت مستقیم:

اگرچه سیگنال‌های زیادی در این حافظه نهان وجود دارد، اما عملکرد آن‌ها بسیار ساده است. هنگامی‌که سیگنال enable مقدار ۱، (high) است، مقادیر دو خط کنترل اصلی comp و write چهار حالت به شرح زیر به وجود می‌آورند:

۱- *مقایسه و خواندن (comp=1, write=0)*

این حالت زمانی رخ می‌دهد که دستور load اجرا می‌شود. سیگنال‌های tag_in و index و word باید معتبر باشند. اگر hit رخ دهد، سیگنال data_out دارای مقدار معتبر خواهد بود. در صورت رخداد miss، خروجی سیگنال valid، مشخص‌کننده اعتبار بلوکی است که خط معین‌شده توسط index اشغال کرده است. خروجی سیگنال dirty، مقدار بیت dirty آن خط از حافظه است.

۲- *مقایسه و نوشتن (comp=1, write=1)*

این حالت زمانی رخ می‌دهد که دستورالعمل store اجرا شود. سیگنال‌های data_in و tag_in و index و word، باید valid باشند. اگرmiss رخ دهد، تغیری در وضعیت حافظه نهان رخ نمی‌دهد (داده برای نوشته شدن باید به حافظه اصلی فرستاده شود). در صورت رخداد hit، سیگنال word در لبه بالارونده کلاک در حافظه، *نوشته* می‌شود و مقدار **بیت** dirty آن خط از حافظه نهان، ۱ می‌شود (درواقع چون در سیکل *نوشتن داده هستیم*، خروجی **سیگنال** dirty بی‌معنا خواهد بود). زمانی که hit رخ دهد، لازم است که به سیگنال خروجی valid نیز توجه کرد. اگر hit رخ دهد اما داده valid نبود، سیستم باید مانند حالت miss رفتار کند.

اگر miss اتفاق بیفتد، خروجی سیگنال valid مشخص‌کننده اعتبار بلوکی است که آن خط از حافظه را اشغال کرده است. به‌علاوه مقدار سیگنال dirty نیز نمایش‌دهنده وضعیت بیت dirty آن خانه از حافظه نهان، خواهد بود. برعکس اگر مقدار سیگنال hit، ۱ باشد و سیگنال‌های write و comp نیز مقادیر ۱ داشته باشند، خروجی **سیگنال** dirty، ۰ باقی خواهد ماند (زیرا حافظه نهان در حین انجام عمل *نوشتن* است).

۳- *دسترسی و خواندن (comp=0, write=0)*

زمانی رخ می‌دهد که بخواهیم به tag و data ای، خارج از حافظه نهان (از حافظه اصلی) دسترسی داشته باشیم. مورداستفاده آن زمانی است که خطی از حافظه نهان، به‌عنوان قربانی برای جایگزینی انتخاب‌شده است (اگر بیت dirty آن خط از حافظه نهان دارای مقدار ۱ باشد، باید در حافظه اصلی، *بازنویسی* شود).

درواقع حافظه نهان در حالت comp = 0، مانند RAM عمل می‌کند. سیگنال‌های index و word باید معتبر باشند تا بتوانند داده موردنظر برای *خواندن* را انتخاب کنند. در این سیکل، سیگنال‌های data_out و valid و dirty نیز معتبر هستند.

۴- دسترسی و نوشتن *(comp=0, write=1)*

زمانی که داده‌ای از حافظه اصلی آورده شده است و باید در حافظه نهان ذخیره شود. سیگنال‌های index و tag_in و wordو valid_in و data_in، در این حالت باید معتبر باشند. در لبه بالارونده کلاک مقادیر در خط مشخص‌شده در حافظه نهان نوشته می‌شوند و همچنین مقدار بیت dirty، ۰ قرار داده خواهد شد.

در ابتدا مقدار بیت valid تمام خانه‌های حافظه نهان، ۰ است.

## ۳) امتیازی ۲: پیاده‌سازی یک حافظه نهان مجموعه انجمنی:

پس از ساختن یک حافظه نهان که عملکرد نگاشت مستقیم دارد، می‌توانید واحد[9] حافظه نهان دیگری اضافه کنید که به‌صورت two-way set-associative کار می‌کند.

چهار حالت جدید به‌صورت زیر به وجود می‌آیند:

۱- مقایسه و خواندن (comp=1, write =0)

لازم است که index و word به هر دو واحد (way) حافظه نهان برسد. اگر هرکدام از خروجی‌های hit یک شوند، یک hit خواهیم داشت. از یکی از خروجی‌های hit به‌عنوان خط select تسهیم کننده[10] بین دو خروجی data استفاده کنید. اگر miss رخ داد، بر اساس منطق زیر تصمیم‌گیری کنید کدام بخش باید حذف شود:

- اگر یکی از آن‌ها valid بود دیگری را حذف کنید.
- اگر هیچ‌کدام valid نبودند،way شماره ۰ را انتخاب کنید.
- اگر هر دو valid بودند از *الگوریتم Pseudo-Random replacement* استفاده کنید.

۲- مقایسه و نوشتن (comp =1, write =1)

لازم است که index و word و data به هر دو واحد حافظه نهان فرستاده شوند. اگر هرکدام از خروجی‌های hit یک باشند، hit رخ‌داده است. توجه داشته باشید که در طراحی شما باید این اطمینان وجود داشته باشد که داده تنها دریکی از واحدهای حافظه نهان نوشته شود.

۳- دسترسی و خواندن (comp =0, write = 0)

پس از تصمیم‌گیری درباره واحد قربانی شونده، از بیتِ select برای انتخاب data، valid و dirty bit یکی از واحدها استفاده کنید.

۴- دسترسی و نوشتن (comp = 0, write = 1)

سیگنال‌های index، word، data و بیت valid را برای هر دو واحد تنظیم کنید. اطمینان حاصل کنید که فقط در واحد درست بیت write برابر یک می‌شود.

---

[9] Module

[10] Multiplexer

### ۳,۱ الگوریتم *Pseudo-Random replacement*

۱ یک فلیپ-فلاپ به نام victimway که به‌صورت ۰ مقدار اولیه داده‌شده است، داشته باشید.

۲ در هر بار read یا write کردن در حافظه نهان victimway را برعکس کنید.

۳ پس از یک miss در صورت وجود یک خط *نامعتبر* دریکی از واحدها، خط آورده شده از حافظه اصلی در آن واحد قرار می‌گیرد.

۴ اگر هر دو خط در واحدها معتبر باشند، واحد اول را برای جایگزینی انتخاب می‌کنیم.

## ۴) نکات مهم:

- پروژه شما باید دارای یک محک برای تست[11]، به‌منظور ارائه و ارزیابی تمامی قابلیت‌های پیاده‌سازی شده باشد.
- پیاده‌سازی‌ها در سطح بالای تجرید خواهد بود. لزومی به پیاده‌سازی ثبات و مانند آن نیست.
- پروژه به‌صورت انفرادی انجام خواهد شد.
- تهیه گزارش برای این پروژه الزامی است.
- با هرگونه کپی (از دانشجو و از وب) به‌شدت برخورد خواهد شد.
- کارگاه‌های آموزشی کار با Verilog برگزار خواهد شد، زمان آن به‌زودی اعلام می‌گردد.
- در صورت وجود هرگونه ابهام و سؤال، از طریق مودل پرسش‌های خود را مطرح کنید.

موفق باشید.

[11] Test bench