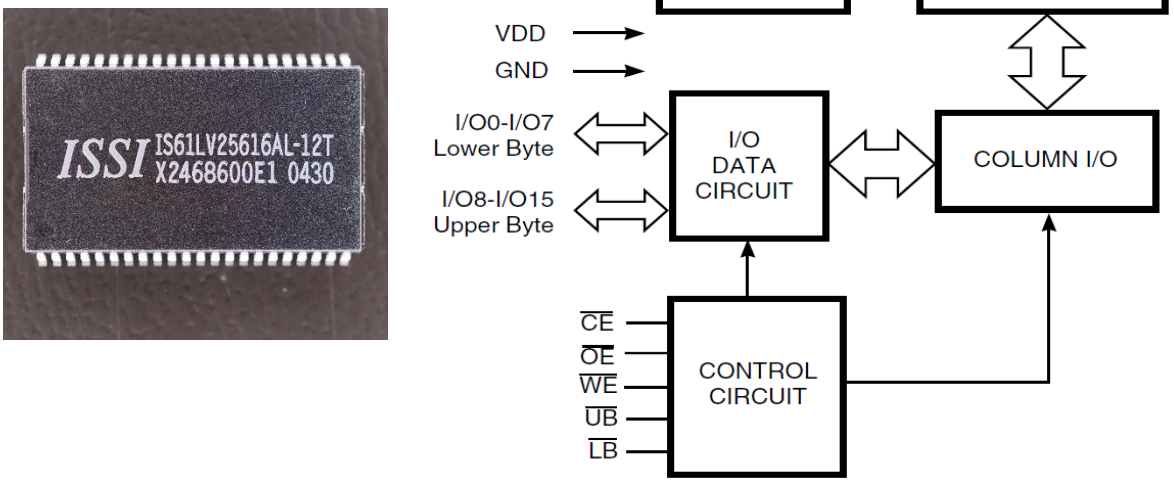
حافظه آسنکرون و کنترل‌کننده سنکرون

در سیستم‌های دیجیتال، حافظه یکی از اجزای اصلی به شمار می‌رود و حافظه‌های SRAM به‌دلیل سرعت بالا و ساختار ساده، کاربرد فراوانی دارند. بسیاری از انواع این حافظه‌ها به‌صورت آسنکرون ساخته می‌شوند، در حالی‌که بیشتر مدارهای دیجیتال به‌صورت سنکرون طراحی می‌شوند.

در این تمرین ابتدا یک حافظه SRAM آسنکرون طراحی و شبیه‌سازی می‌شود. در ادامه یک کنترل‌کننده سنکرون برای این حافظه طراحی می‌گردد تا امکان استفاده از آن در سیستم‌های دیجیتال سنکرون فراهم شود.

بخش اول) شبیه‌سازی حافظه SRAM آسنکرون

حافظه‌ی **SRAM** مخفف (Static Random Access Memory) نوعی حافظه‌ی تصادفی است که برای ذخیره‌ی هر بیت از بیت‌های خود از یک سلول حالت پایدار (معمولاً شامل 4 تا 6 ترانزیستور) استفاده می‌کند و تا زمانی که تغذیه‌ی برق وصل باشد، داده‌ها را بدون نیاز به تازه‌سازی (refresh) حفظ می‌کند. برخلاف **DRAM** که برای نگهداری داده‌ها باید به‌صورت دوره‌ای تازه‌سازی شود، در SRAM داده‌ها به‌طور ثابت و با تأخیر کمتر در دسترس هستند، اما به‌دلیل تعداد بالاتر ترانزیستور در هر سلول، هزینه و مصرف فضا بیشتر است. به خاطر سرعت بالا و تأخیر کم، SRAM معمولاً به‌عنوان حافظه‌ی کش (Cache) در پردازنده‌ها یا به‌عنوان حافظه‌ی جانبی پرسرعت در بردهای FPGA به‌کار می‌رود.



ماژول SRAM

در این بخش، هدف شما پیاده‌سازی و شبیه‌سازی یک ماژول حافظه SRAM آسنکرون مشابه با چیپ **IS61LV25616AL** شرکت ISSI است. این حافظه دارای ظرفیت 16 × 256k بیت است و با سیگنال‌های کنترلی متنوعی کار می‌کند اما به منظور سهولت شبیه‌سازی، ظرفیت ماژول حافظه خود را 16 × 512 بیت در نظر بگیرید.

ورودی/خروجی‌های ماژول

در این بخش هدف، پیاده‌سازی ماژول حافظه و شبیه‌سازی عملکرد آن است. بدین منظور در این ماژول، آرایه 512 عددی از رجیسترهای 16 بیتی تعریف نمایید.

جدول زیر خلاصه‌ای از عملکرد سیگنال‌های این ماژول را نشان می‌دهد و در ادامه شرح کامل عملکرد آن نشان داده شده است.

سیگنال	توضیح
`addr[8:0]`	آدرس حافظه (512 آدرس ممکن) 9 بیتی
`data[15:0]`	گذرگاه داده دوطرفه (inout) 16 بیتی
`CE`	Chip Enable (فعال‌سازی تراشه، فعال با صفر)
`OE`	Output Enable (فعال‌سازی خروجی، فعال با صفر)
`WE`	Write Enable (فعال‌سازی نوشتن، فعال با صفر)
`UB`	Upper Byte Enable (برای فعال‌سازی بیت‌های 15:8)
`LB`	Lower Byte Enable (برای فعال‌سازی بیت‌های 7:0)

این ماژول دارای دو گذرگاه است:

- گذرگاه `addr` یک گذرگاه ورودی 9 بیتی برای مشخص کردن آدرس (با توجه به داشتن 512 خانه حافظه) خانه‌ی مورد نظر از حافظه است.
- گذرگاه `data` یک گذرگاه مشترک ورودی/خروجی 16 بیتی برای تبادل داده است.

به کمک این گذرگاه‌ها و سیگنال‌های کنترلی عملیات خواندن و نوشتن روی حافظه به صورت آسنکرون انجام می‌شود. سیگنال‌های کنترلی همگی به صورت **Active Low** بوده و به شرح زیر هستند:

- سیگنال `WE` مشخص می‌کند قرار است عملیات نوشتن از حافظه انجام شود یا عملیات خواندن روی آن. زمانی که مقدار آن HIGH باشد، عملیات خواندن انجام می‌شود و زمانی که مقدار آن LOW باشد، عملیات نوشتن.

- سیگنال `CE` مشخص می‌کند که ماژول در حالت عملکرد فعال عادی خود است یا در حالت آماده به کار (Standby) قرار دارد تا انرژی کمتری مصرف کند.
- سیگنال `OE` مشخص می‌کند گذرگاه خروجی ماژول فعال است یا در حالت High Impedance قرار دارد.
- با توجه به اینکه خانه‌های حافظه به صورت 16 بیتی (دو بایتی) هستند، با استفاده از سیگنال `UB` می‌توان اجازه‌ی دسترسی به بایت پر ارزش داده جهت انجام عملیات خواندن یا نوشتن داده شود. سیگنال `LB` همین مورد را برای بایت کم‌ارزش داده انجام می‌دهد. زمانی که هر دوی این سیگنال‌ها LOW باشند، عملیات روی تمام 16 بیت داده‌ی خانه‌ی مورد نظر از حافظه انجام می‌پذیرد.

عملکرد ماژول

این ماژول دارای سه حالت کاری است:

1. حالت خواندن (Read):

هنگام خواندن داده، ماژول خارجی آدرس خانه‌ی مورد نظر روی گذرگاه آدرس قرار می‌دهد و مقدار z را روی گذرگاه داده قرار می‌دهد. ماژول حافظه نیز داده‌ی موجود در آدرس مورد نظر را روی گذرگاه داده قرار می‌دهد.

- شرایط فعال‌سازی: `CE = 0, WE = 1, OE = 0`
- اگر:
  - `UB = 0` و `LB = 1` آنگاه فقط بیت‌های بالا (`data[15:8]`)
  - `UB = 1` و `LB = 0` آنگاه فقط بیت‌های پایین (`data[7:0]`)
  - `UB = 0` و `LB = 0` آنگاه کل 16 بیت
- در سایر حالات، گذرگاه `data` باید high impedance (حالت z) باشد.

2. حالت نوشتن (Write):

هنگام نوشتن داده، ماژول خارجی آدرس خانه‌ی مورد نظر روی گذرگاه آدرس قرار می‌دهد و داده‌ی جدید را روی گذرگاه داده قرار می‌دهد. ماژول حافظه نیز مقدار z روی گذرگاه داده قرار داده، داده‌ی موجود در گذرگاه داده را خوانده و در آدرس مورد نظر از حافظه می‌نویسد.

- شرایط فعال‌سازی: `CE = 0, WE = 0` بوده و حداقل یکی از `UB` یا `LB` باید صفر باشد.
- داده‌ای که روی گذرگاه `data` قرار گرفته است، در آدرس مشخص شده توسط گذرگاه `addr` نوشته می‌شود.
- بخش‌هایی که توسط `UB` یا `LB` فعال نشده‌اند، نباید تغییر کنند.

3. حالت عدم انتخاب (Deselected):

- هر زمانی که `CE = 1` است، حافظه در حالت غیرفعال است و گذرگاه داده باید high impedance شود.

برای اطلاع بیشتر از نحوه عملکرد سیگنال‌ها به دیتاشیت این ماژول مراجعه نمایید.

نکته‌ی مهم: برای این که شبیه‌سازی شما نزدیک به واقعیت باشد، بایستی تأخیرهای خواندن و نوشتن داده‌ها را لحاظ کنید. بدین منظور حداقل تأخیرهای زیر را در نظر بگیرید:

- تأخیر خواندن از یک آدرس از حافظه: 15 نانوثانیه
- تأخیر نوشتن روی یک آدرس از حافظه: 15 نانوثانیه
- تأخیر تغییر حالت گذرگاه داده از یک مقدار به z و بالعکس: 4 نانوثانیه

نمره امتیازی: در صورتی که مقادیر دقیق تأخیرها را در هر عملیات از دیتاشیت استخراج نمایید و در طراحی خود قرار دهید تا 10٪ نمره‌ی امتیازی دریافت می‌کنید.

تست‌بنچ

برای ماژول حافظه‌ای که طراحی کرده‌اید، یک تست‌بنچ بنویسید و عملکرد بخش‌های مختلف آن را بررسی و ارزیابی نمایید. خروجی شبیه‌سازی را به‌صورت فایل vcd تولید و در فایل ارسالی قرار دهید.

بدین منظور، سناریوی زیر را پیاده‌سازی کنید:

1. در بایت کم‌ارزش مربوط به 10 خانه‌ی اول حافظه، مقدار آدرس هر خانه را به‌عنوان داده در همان خانه بنویسید.
2. در بایت پرارزش مربوط به 10 خانه‌ی دوم حافظه، لگاریتم پایه 2 آدرس هر خانه را با استفاده از تابع `$clog2` بنویسید.
3. در 10 خانه‌ی سوم حافظه، از تابع `$random` برای تولید داده‌های تصادفی استفاده کرده و آن‌ها را در حافظه بنویسید.
4. مقادیر موجود در این 30 خانه از حافظه را خوانده و در خروجی نمایش دهید (با استفاده از دستور `$display`).
5. اکنون ماژول را در حالت **standby** قرار دهید (با `CE=1`) و دوباره همین عملیات را تکرار نمایید. بررسی کنید که در این حالت، حافظه نباید واکنشی نشان دهد (هیچ نوشتنی انجام نشود و خواندن نیز داده‌ای برنگرداند).

بخش دوم) طراحی کنترل‌کننده برای SRAM

امروزه اغلب سیستم‌های دیجیتال به‌صورت سنکرون و مبتنی بر سیگنال کلاک (`clk`) طراحی می‌شوند. در مقابل، بسیاری از حافظه‌های خارجی مانند SRAM عملکردی آسنکرون دارند و مستقل از کلاک سیستم کار می‌کنند.

در بخش اول تمرین، شما یک ماژول SRAM آسنکرون طراحی کرده‌اید. اکنون هدف این بخش، طراحی یک واحد کنترل‌کننده برای این حافظه است، به‌گونه‌ای که یک سیستم دیجیتال 32 بیتی با فرکانس مشخص بتواند به‌صورت سنکرون با آن ارتباط برقرار کند.

ورودی/خروجی‌های ماژول

ماژول کنترل‌کننده‌ی شما باید دارای پارامتر و پورت‌های زیر باشد:

پارامتر:

- `freq`: فرکانس کاری سیستم دیجیتال بر حسب MHz

ورودی‌ها:

- `clk`: سیگنال کلاک دستگاه دیجیتال
- `rst`: سیگنال ریست (آسنکرون، فعال با 1) - با فعال شدن، مقدار خروجی صفر شود.
- `memRead`: فرمان آغاز عملیات خواندن از SRAM (فعال با 1)
- `memWrite`: فرمان آغاز عملیات نوشتن به SRAM (فعال با 1)
- `addrTarget`: آدرس مقصد حافظه (به‌صورت بلوک‌های 2 کلمه‌ای) 9 بیتی
- `dataIn`: گذرگاه داده‌ی ورودی 32 بیتی برای نوشتن

خروجی‌ها:

- `ready`: نشان‌دهنده‌ی پایان عملیات خواندن یا نوشتن
- `dataOut`: گذرگاه داده‌ی خروجی 32 بیتی حاصل از خواندن دو خانه‌ی حافظه

گذرگاه‌های اتصال به SRAM (خروجی یا دوطرفه):

- `addr`: آدرس حافظه 9 بیتی(متصل به SRAM)

- `data`: گذرگاه داده‌ی دوطرفه (inout) 16 بیتی
- `CE`: (خروجی، همواره صفر)
- `OE`: (خروجی، همواره صفر)
- `WE`: (خروجی، برای کنترل خواندن یا نوشتن روی حافظه)
- `UB`: (خروجی، همواره صفر)
- `LB`: (خروجی، همواره صفر)

عملکرد ماژول

این ماژول که به صورت سنکرون با سیگنال `clk` کار می‌کند، دارای دو حالت کاری است که باید طی آن‌ها سیگنال‌ها و گذرگاه‌ها را به صورت مناسب تنظیم نمایید:

1. حالت خواندن از **SRAM**:

در زمان `memRead=1`

- ابتدا `ready` را صفر کنید.
- دو خانه‌ی پیاپی از SRAM بخوانید (بایت پایین و بالا).
- مقدار نهایی 32 بیت خوانده‌شده را در `dataOut` قرار دهید.
- پس از اتمام عملیات، `ready=1` شود و همین مقدار تا عملیات بعدی باقی بماند.

2. حالت نوشتن روی **SRAM**:

در زمان `memWrite=1`

- `ready` را صفر کنید.
- مقدار دو بایت اول `dataIn` را در خانه اول و دو بایت دوم `dataIn` را در خانه دوم بنویسید.
- سپس `ready=1` شود و همین مقدار تا عملیات بعدی باقی بماند.

توجه: از آنجا که بسیاری از پردازنده‌ها 32 بیتی هستند، عملیات خواندن/نوشتن باید روی دو خانه‌ی متوالی از حافظه SRAM انجام شود. به همین دلیل، آدرس‌ها به‌صورت دوتایی افزایش می‌یابند:

آدرس‌های خانه‌های 0 و 1 ← `addrTarget = 0`

آدرس‌های خانه‌های 2 و 3 ← `addrTarget = 2`

و به همین ترتیب...

توجه: فرض کنید در طول عملکرد معمول سیستم، سیگنال‌های `CE, OE, UB, LB` همواره صفر هستند و تنها `WE` مطابق عملیات خواندن/نوشتن تغییر می‌کند.

نکته مهم: شما باید یک ماشین حالت (**FSM**) طراحی کنید که عملیات خواندن و نوشتن را بر اساس `clk` و `freq` به شکل زمان‌بندی‌شده انجام دهد. برای مثال، اگر فرکانس دستگاه 100 مگاهرتز باشد (دوره زمانی کلاک 10 نانوثانیه)، مراحل عملیات باید در چند سیکل متوالی از `clk` با تأخیر مشخص انجام شوند. طبق بخش اول تمرین ، تأخیر عملیات خواندن و نوشتن روی حافظه SRAM را 15 نانوثانیه در نظر بگیرید.

تست‌بنچ

برای بررسی عملکرد کنترل‌کننده، آن را در کنار SRAM که طراحی کرده بودید، قرار داده و تست‌بنچی طراحی نمایید که موارد زیر را بررسی کند. خروجی شبیه‌سازی را به‌صورت فایل vcd تولید و در فایل ارسالی قرار دهید.

در دو فرکانس متفاوت `freq = 10 MHz` و `freq = 200 MHz` سناریوی زیر را انجام دهید:

1. ده مقدار تصادفی 32 بیتی را با استفاده از تابع `$random` تولید کنید.
2. این مقادیر را به‌ترتیب در ده آدرس مختلف (دو خانه‌ای) از حافظه بنویسید.
3. سپس همان آدرس‌ها را دوباره بخوانید.
4. داده‌های خوانده‌شده را با داده‌های اصلی مقایسه و با `$display` چاپ کنید.
5. رفتار سیگنال `ready` را در حین عملیات بررسی کنید.