# پروژه درس معماری کامپیوتر

## پیاده سازی حافظه کامپیوتر با زبان vhdl

گروه 1: محمد اصولیان – محمد حسین عباسپور

## مقدمه:

در سیستم های دیجیتال و کامپیوتر ها، حافظه نقش اساسی را ایفا میکند. بدون حافظه انجام هیچ یک اس فعالیت‌های روزانه با کامپیوتر امکان پذیر نیست. قطعات متعددی در کامپیوتر مانند چرخ دنده در کنار هم قرار گرفته‌اند تا بتوان به کمک این قطعات از حافظه استفاده کرد. از این قطعات میتوان hard disk, memory, cache, tlb, page table را نام برد. با هر بار دسترسی به حافظه که توسط cpu انجام می‌شود، فرایندی پیچیده میان این قطعات طی میشود تا بالاخره بتوان از یک آدرس، به داده موجود در آن آدرس رسید.

در این پروژه قصد داشتیم تا با استفاده از زبان سخت افزار نویسی vhdl، حافظه یک کامپیوتر را شبیه سازی کنیم.

## نحوه پیاده سازی:

برای پیاده سازی حافظه، ابتدا هر قطعه از قطعات درگیر با حافظه را به صورت جداگانه پیاده سازی کردیم. هر قطعه ورودی ها و خروجی ها مخصوص به خود را دارد. هر قطعه functionality خاصی دارد که ما این functionality ها را درون هر قطعه پیاده سازی کردیم.

وقتی قطعات مختلف درگیر با حافظه را به صورت جداگانه داشته باشیم تنها کاری که لازم است انجام دهیم این است که این قطعات را به هم دیگر متصل و کنترل کنیم. متصل کردن قطعات به کمک سیم ها انجام میشود و در vhdl این کار با map کردن پورت های قطعه ها انجام می‌شود. برای کنترل فرایند دسترسی به حافظه در کامپیوتر از io کمک گرفته می‌شود. در این پروژه قصد نداشتیم تمام io را شبیه سازی کنیم بنابراین قطعه ای به نام io طراحی کردیم که تنها کنترل قطعات طراحی شده دیگر را بر عهده دار.

در انتها برای بررسی عملکرد قطعات به صورت جداگانه برایشان تست بنچ طراحی شده و برای قطعه io که هسته اصلی پروژه هست نیز test bench طراحی شده تا صحت عملکرد پروژه را بررسی کند.

در ادامه به بررسی هر کدام از قطعات طراحی شده میپردازیم.

## شرح قطعات:

### 1- Hard disk:

این قطعه حافظه اصلی کل پروژه است که تمام اطلاعات را در بر دارد. هارد دیسک طراحی شده در واقع یک آرایه بزرگ از 0 ها و 1 هاست که برای سادگی کار با آن، این ارایه بزرگ به page ها قسمت قسمت شده است. هارد پروژه به طور رندوم با 0 و 1 پر شده و word 48 آن برای نمونه پر شده تا بر روی آن تست اعمال شود.

#### ورودی ها:

R_bit: هرگاه میخواستیم عمل read کردن انجام دهیم این بیت 1 میشود.

Storage_add: آدرسی از هارد که میخواهیم آن را بخوانیم

#### خروجی ها:

Page: محتوای ادرس مشخص شده خوانده میشود و در این خروجی ریخته میشود.

### 2- Memory:

این قطعه همان مموری کامپیوتر است. شامل قسمتی از اطلاعات هارد دیسک است که به هنگام نیاز درون مموری قرار داده میشوند تا دسترسی سریع تر صورت گیرد. مموری پیاده سازی شده در این پروژه میتواند اطلاعات را بخواند و بنویسد. برای نوشتن اطلاعات در مموری در خانه ها به ترتیب پیش میرویم و در هر بار عمل نوشتن، خانه ای overwrite میشود که قدیمی ترین خانه است.

ورودی ها:

R_bit: هر گاه بخواهیم عمل read را انجام دهیم این بیت، یک میشود.

W_bit: هر گاه بخواهیم در موری بنویسیم این بیت 1 میشود.

Ph_add: آدرسی که میخواهیم محتوای آن را بخوانیم

Write_data_from_disk: داده ای که میخواهیم درون موری بنویسیم را از این ورودی وارد میکنیم. در صورتی که بخواهیم read کنیم، این ورودی don’t care است.

## خروجی ها:

Read_data: داده خوانده شده در این خروجی ریخته میشود.

Ppnout: وقتی داده جدیدی در موری نوشته میشود این خروجی بیان میکند که این داده در چه آدرسی از موری نوشته شده.

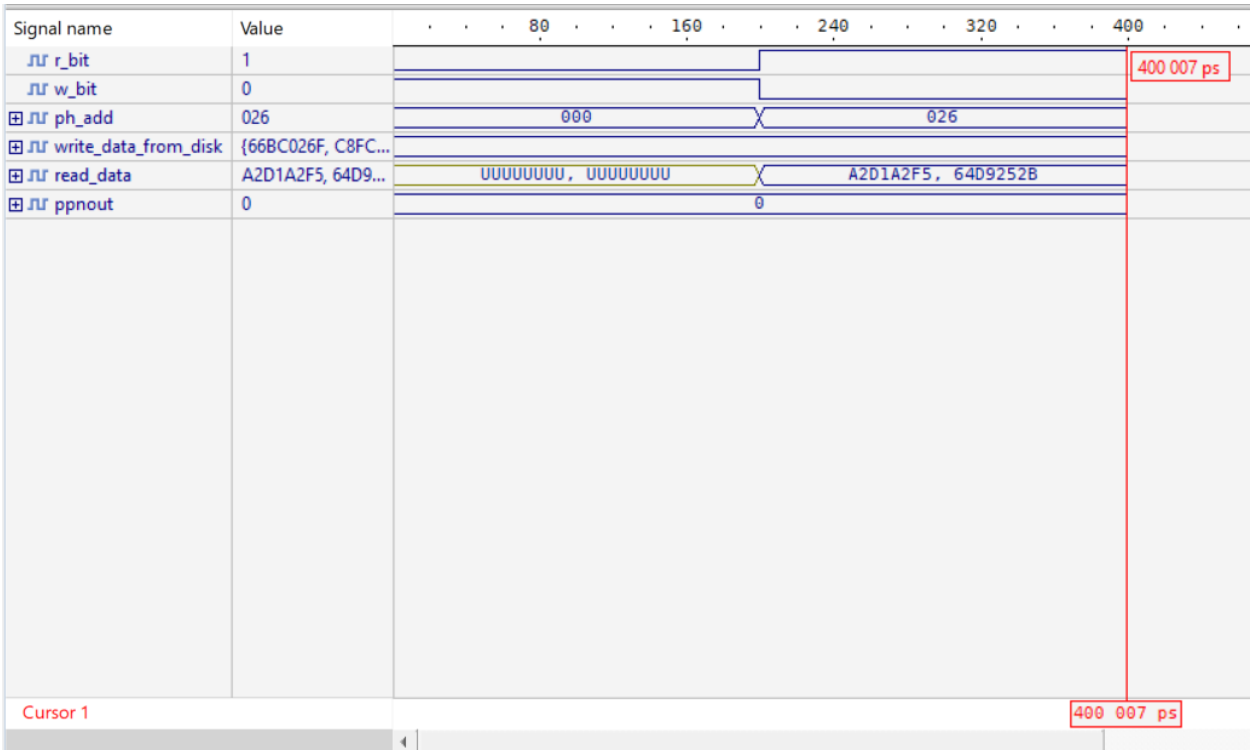


*Figure 1 - نحوه عملکرد memory و تایید صحت عملکرد آن در test bench*

## 3- Cache و two-way cache

این قطعه کش کامپیوتر می‌باشد. کش در کامپیوتر یک حافظه سریع تر اما کم حجم تر از مموری است. کش پیاده سازی شده در این پروژه هم قابلیت read و write دارد. هر دو نوع دایرکت کش و 2 way پیاده سازی شده و برای استفاده ازهر کدام تنها کافیست که port map قطعات را در قطعه io تغییر دهیم.

### ورودی ها:

R_bit: هر گاه بخواهیم عمل read را انجام دهیم این بیت، یک میشود.

W_bit: هر گاه بخواهیم عمل write را انجام دهیم این بیت یک میشود.

ph_add: ادرسی که میخواهیم آن را بخوانیم از این ورودی وارد میشود.

write_data_from_memory: هر گاه بخواهیم داده ای را در کش بنویسیم آن را در این ورودی قرار میدهیم.

### خروجی ها:

Read_data: داده خوانده شده از کش در این خروجی قرار میگیرد.

Hit: در صورتی که داده آدرس وارد شده در کش باشد hit یک میشود و در غیر این صورت 0 می شود تا داده از مموری خوانده شود.

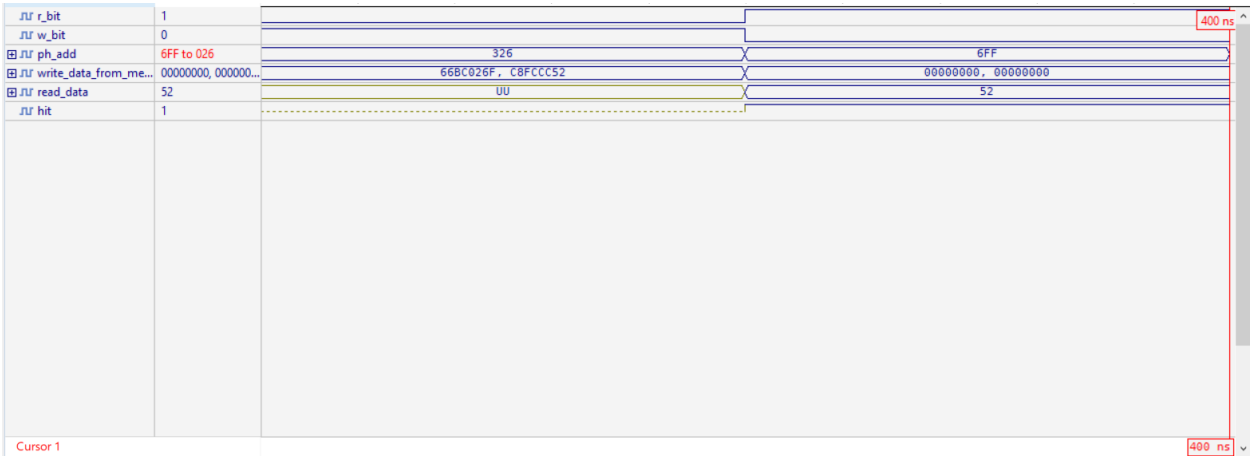


*Figure 2 - نحوه عملکرد Cache و تایید صحت عملکرد آن در test bench*

## 4- TLB و 4 way set TLB

این قطعه مانند یک کش برای page table میباشد و ادرس های پرکاربرد در این قطعه سریع و کم حافظه قرار میگیرند تا دسترسی سریع تر صورت گیرد. این قطعه هم قابلیت خواندن و نوشتن دارد. در حالت خواندن ppn مربوط به vpn داده شده را میخواند و در حالت نوشتن ppn داده شده را با تگ vpn داده شده می‌نویسد.

### ورودی ها:

R_bit: هر گاه بخواهیم read کنیم این بیت یک میشود.

W_bit: هر گاه بخواهیم write کنیم این بیت یک می‌شود.

Vpn: ادرس ویرچوالی که میخواهیم ppn آن را پیدا کنیم در این ورودی داده می شود.

ppn_from_pt: هر گاه بخواهیم ppn ای را در tlb بنویسیم آن را در این ورودی مینویسیم.

### خروجی ها:

Ppn: ppn یاقت شده برای vpn داده شده در این خروجی نوشته می‌شود.

Hit: خروجی ای که مشخص میکند که ppn ای که دنبال آن هستیم در این tlb موجود هست یا خیر.


*Figure 3 بررسی عملکرد tlb و صحت عملکرد آن در test bench*

## 5- Page table:

پیج تیبل تمام vpn ها را به همراه ppn مربوط به آن ها ذخیره کرده. با وارد شدن یک vpn در صورتی که سطر مربوط به آن valid باشد، ppn مربوط به آن را در خروجی مینویسد.

### ورودی ها:

R_bit: در صورتی که بخواهیم از پیج تیبل بخوانیم این بیت یک میشود.

W_bit: در صورتی که بخواهیم در پیج تیبل بنویسیم این بیت یک میشود.

Vpn: vpn ای که میخواهیم ppn آن را پیدا کنیم در این ورودی نوشته میشود.

Write_ppn: هر گاه بخواهیم یک ppn جدید در پیج تیبل بنویسیم در این ورودی آن را وارد میکنیم.

### خروجی ها:

Read_ppn: ppn ای که میخوانیم در این خروجی قرار میگیرد.

Hit: این بیت مشخص میکند که آیا ppn ای که به دنبال آن هستیم در پیج تیبل بوده یا پیج فالت رخ داده.


*Figure 4 - بررسی عملکرد page table و تایید صحت عملکرد آن در test bench*

## 6- Processor:

پروسسور در کامپیوتر نقش های محاسباتی زیادی دار اما در این پرژه تنها وظیفه دارد که آدرس هایی را تولید کند تا ما مقادیر را از آدرس ها بخوانیم. پروسسور در این پروژه تنها چند ادرس پیشفرض از قبل آماده شده برای نمونه دارد و همان ها را تولی میکند.

### ورودی ها:

R_bit: هرگاه بخواهیم از پروسسور ادرس جدید دریافت کنیم این بیت یک میشود.

Index: برای دریافت ادرس جدید به پروسسور داده میشود.

### خروجی ها:

Vr_add: ادرس تولید شده در این خروجی نوشته میشود.


*Figure 5 - نحوه عملکرد processor و تایید صحت عملکرد آن در test bench*

## 7- Io:

Io کنترل تمام قطعات ذکر شده را بر عهده دارد. ادرس را از پروسسور دریافت کرده و به دنبال آن در Tlb و کش و در صورت لزوم سایر قطعات میگردد. و در نهایت داده یافت شده را در خروجی فطعه مینویسد. نحوه کار این قطعه و الگوریتم اجرا شده در آن مطابق با اسلاید های درس است. برای جزعیات بیشتر به اسلاید ها مراجعه کنید.

### ورودی ها:

Idxin: ایندکسی که به پروسسور میدهیم را در این ورودی مینویسیم تا ادرس تولید شود.

### خروجی ها:

Dataout: داده را از آدرس خوانده و در این خروجی مینویسد.

## جمع بندی:

پیاده سازی این پروژه کمک کرد تا بهتر با ساز و کار حافظه کامپیوتر آشنا شویم و بتوانیم حافظه یک کامپیوتر را از پایه طراحی کنیم. در این مسیر از زبان vhdl کمک گرفته شد تا به جای استفاده از سخت افزار واقعی و گیت های مختلف از این زبان برای نوشتن سخت افزار استفاده کنیم و کار را برای طراحی قطعات ساده تر کرد.

لینک گیت پروژه: https://github.com/MohammadHAbbaspour/CA_Project