به نام خدا

گزارش پروژه معماری کامپیوتر

محمدمهدی آقاجانی

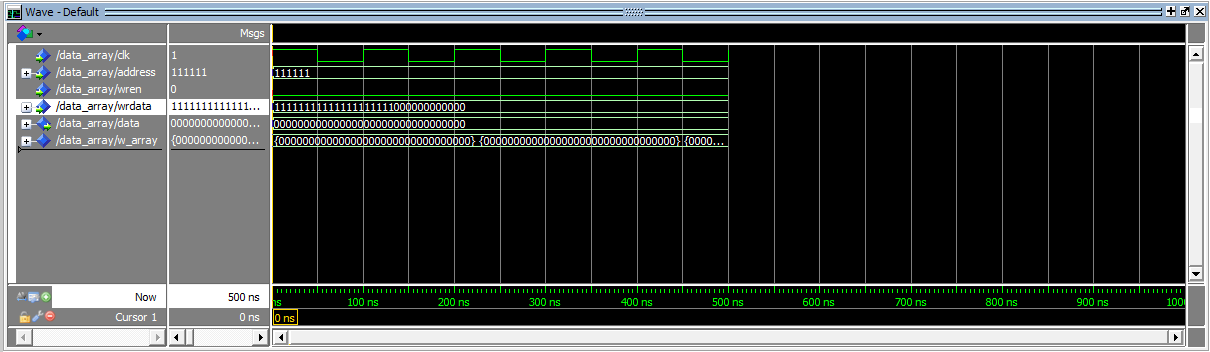
9331056

آرایه داده ای (data array )

آرایه ای از std logic vector ها ست که در واقع 64 عنصر دارد. این ماژول حساس به لبه بالارونده کلاک است و اگر سیگنال wren فعال باشد اقدام به نوشتن می کنده در غیر این صورت از همین آرایه داده مربوطه را می خواند و در خروجی قرار میدهد.

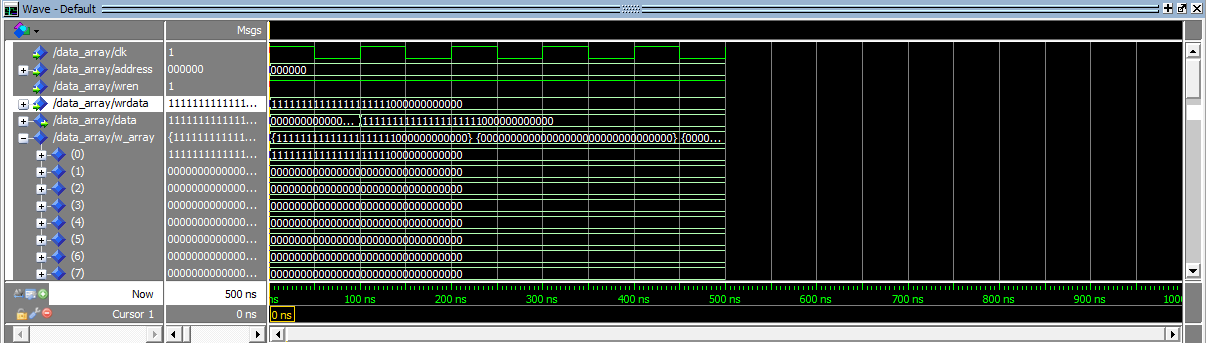
شکل موج خروجی وقتی که wren = 0 :

در این حالت داده در آدرس 111111 که در واقع 00000000000000000000000000000000 است در خروجی قرار گرفته است :



شکل موج خروجی وقتی wren = 1 است :

در این حالت داده wrdata در خانه آدرس که 000000 است قرار گرفته همچنین به خاطر این جایگزینی موفق داده در خروجی نیز قرار گرفته است :

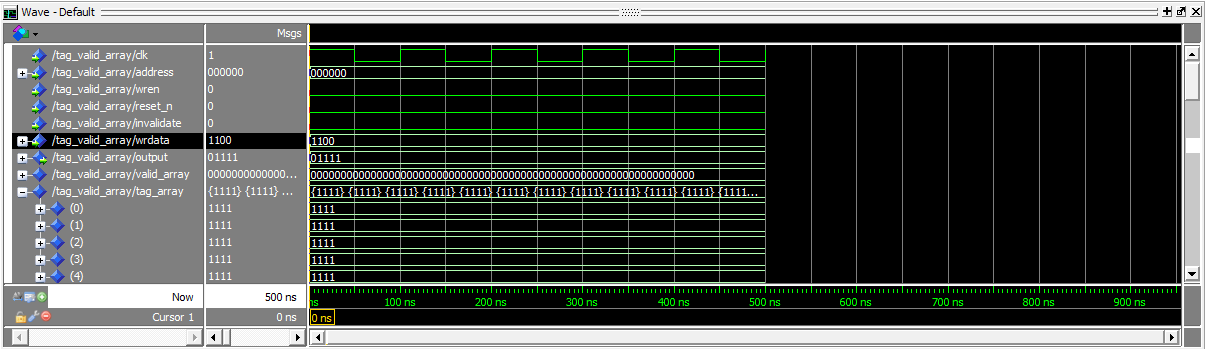


Tag valid array :

نحوه ساخت این ماژول مانند آرایه داده ای می باشد و حساس به لبه بالارونده کلاک است توجه کنید که در ابتدا به صورت پیش فرض تمامی داده های tag ها برابر 1111 تمامی داده های valid برابر صفر قرار داده شده است

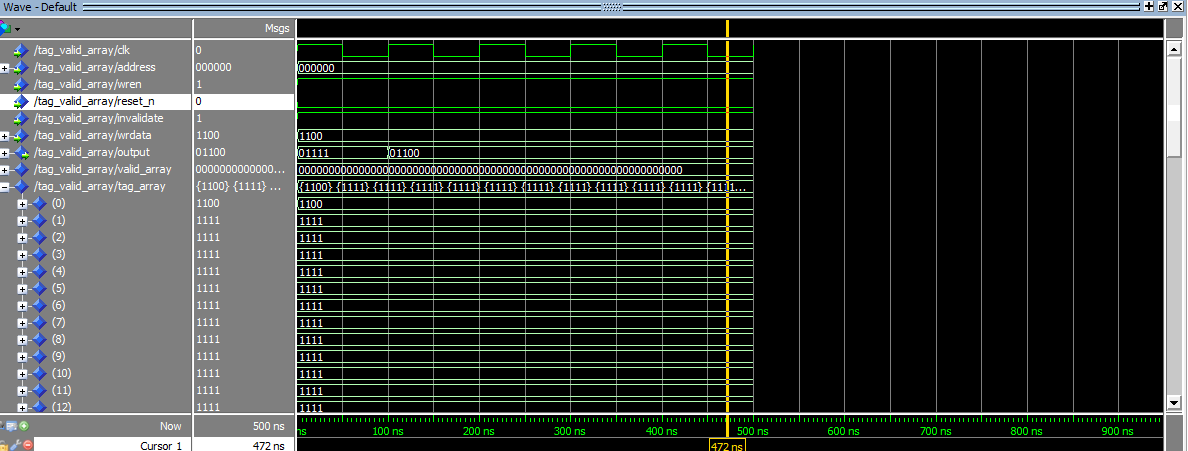
در زیر شکل موج به ازای wren = 0 , reset = 0 را مشاهده می کنید :

در این حالت بدلیل اینکه سیگنال invalidate برابر صفر است valid خانه مورد نظر تغییری نکرده و برابر همان مقدار قبلی ست به همین دلیل در خروجی سیگنال 01111 را مشاهده میکنیم



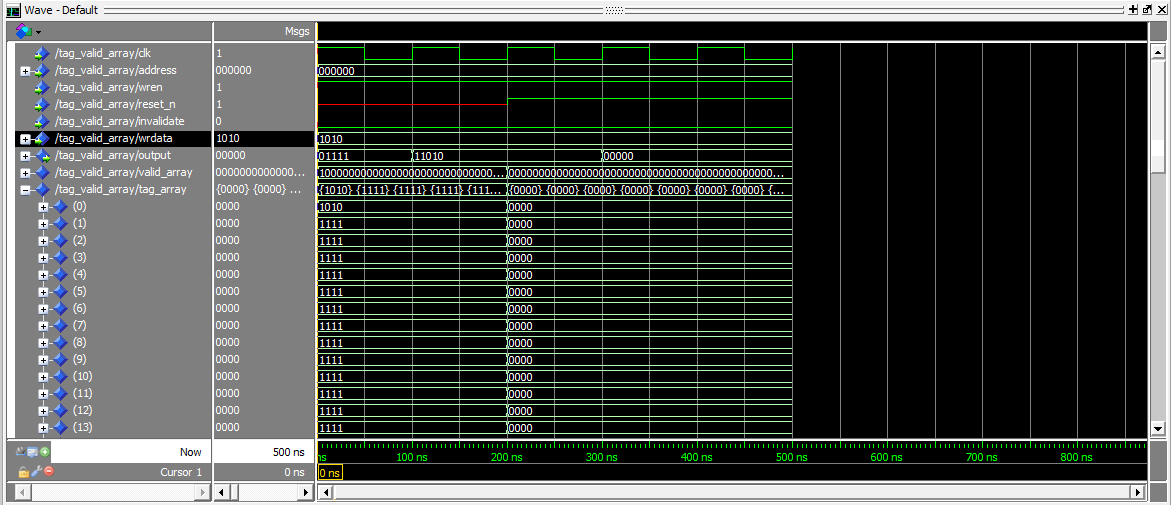
در این حالت شکل موج را به ازای wren = 1 , reset = 0 مشاهده می کنید

دقت کنید که به این علت valid مربوط به آدرس داده شده صفر شده است که invalidate برابر 1 در نظر گرفته شده و الا باید valid مربوط به آدرس داده شده برابر یک قرار میگرفت



در این حالت reset برابر یک قرار داده شده :

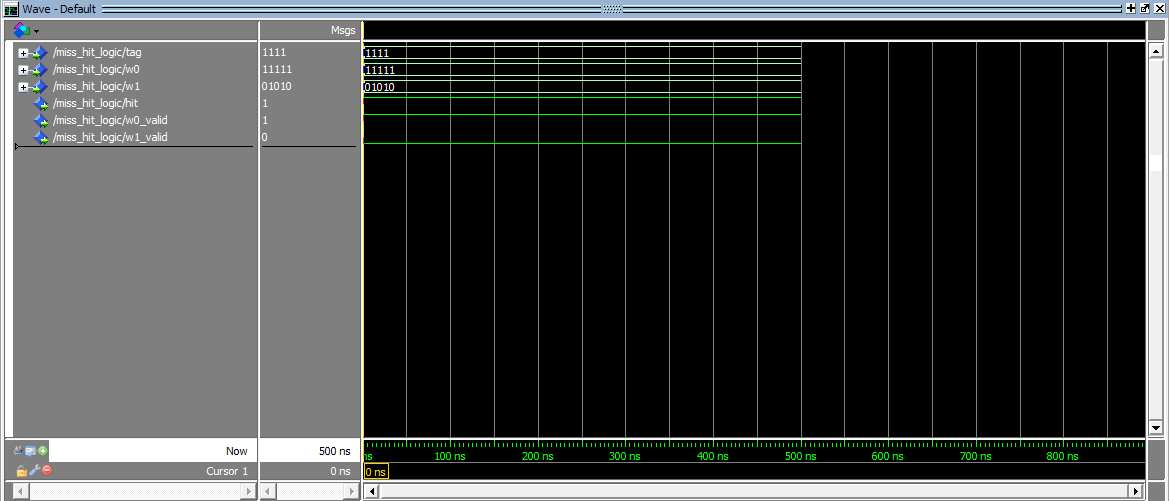
دقت کنید که wren برابر یک است و invalidate برابر صفر است به همین خاطر در ابتدا خروجی برابر داده نوشته شده در آرایه قرار میگیرد و سپس بعد از اینکه ریست برابر یک قرار داده می شود تمامی آرایه برابر سفر قرار میگیرد



منطق مطابقت ( miss-hit logic ) :

این مدار چون حساس به کلاک نیست باید در سطح گیت پیاده شود که از گیت xnor برای مقایسه استفاده شده است و در نهایت با valid مقایسه می شود تا نتیجه را مشخص کند

در این حالت شکل موج را در حالتی که hit رخ داده است مشاهده می کنید:



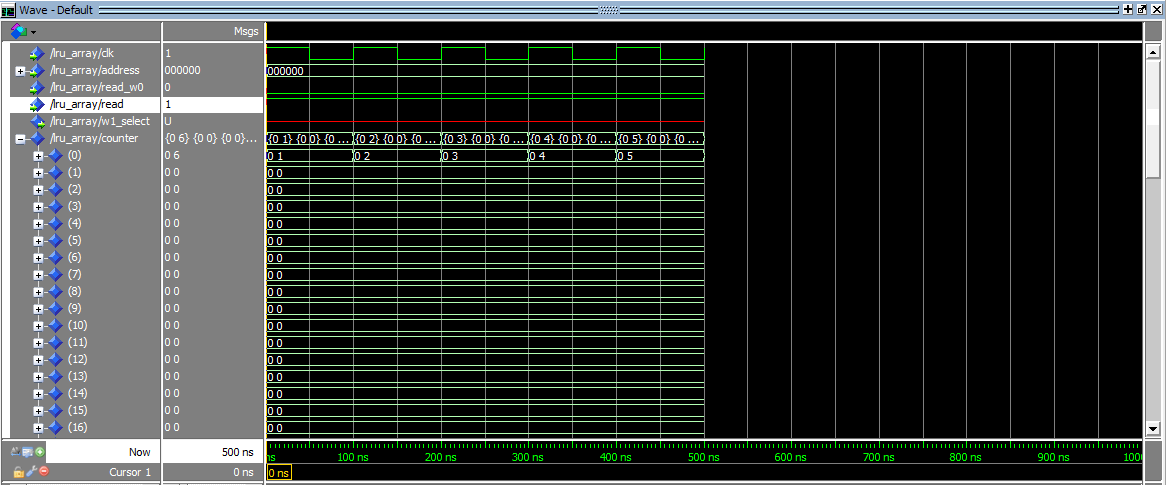
در این حالت miss رخ داده است :



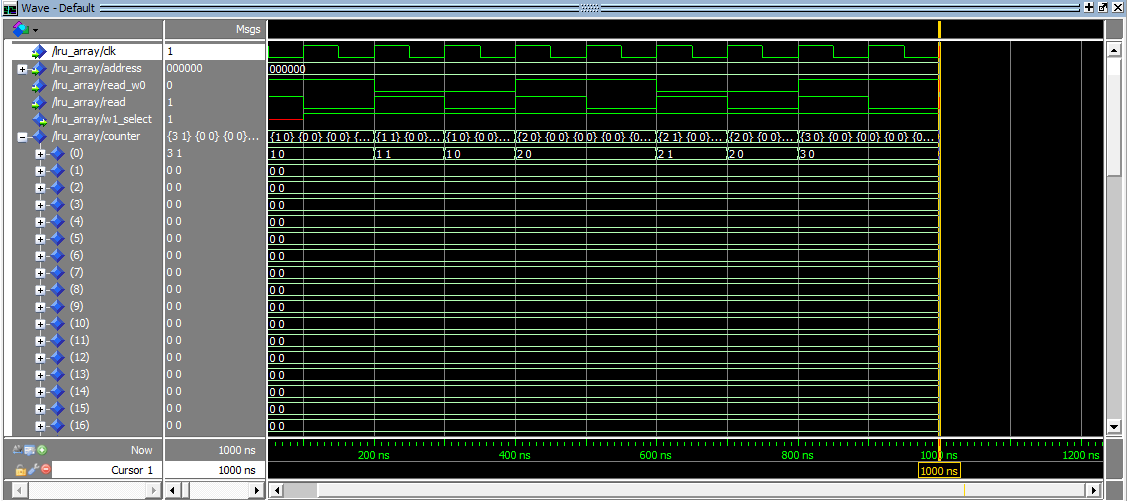
سیاست جایگزینی ( lru array ) :

این ماژول حساس به کلاک است و دارای دو سیگنال کنترلی read و read\_w0 است که هر وقت read برابر یک باشد به این معناست که cache در حال خواندن است و شمارنده به ازای آدرس ورودی یکی زیاد می شود که این افزایش برمبنای سیگنال کنترلی دیگری به نام read\_w0 است که اگر یک باشد شمارنده مربوط به w0 زیاد میشود و برعکس

در زیر شکل موج مربوط به حالت read = 1 , read\_w0 = 0 را مشاهده میکنید :



در شکل زیر شکل موج در حالت کلی تر را برای این ماژول مشاهده میکنید :



نحوه کارکرد controller :

این ماژول دارای یک سری خروجی هایی ست که تماما سیگنال های کنترلی بقیه ماژول ها هستند که عبارتند از : invalidate , ram\_write ,ram\_read ,read\_w0,read\_cache,wren برای کارکرد آن یک FSM طراحی کردیم که به صورت زیر است :

Invalidate : i , ram\_write : rw , ram\_read : rr , read\_w0 : rw0 , read\_cache : rc , wren : wr

read\_request = 1

**Start**

i=0 , rw = 0 , rr = 0 ,

rw0 = 0 , rc = 0 , wr = 0

Ram\_ready = 1

Is\_hit = 0

**read**

i=0 , rw = 0 , rr = 1 ,

rw0 = 0 , rc = 0 , wr = 0

**miss**

i=0 , rw = 0 , rr = 1 ,

rw0 = 0 , rc = 0 , wr = 0

**Read\_from\_ram**

i=0 , rw = 0 , rr = 0 ,

rw0 = 0 , rc = 0 , wr = 1

Cache\_ready = 1

Cache\_ready = 1

Is\_hit = 1

Ram\_reaedy = 1

Cache\_ready = 1

write\_request = 1

**write**

i=1 , rw = 1 , rr = 0 ,

rw0 = 0 , rc = 0 , wr = 0

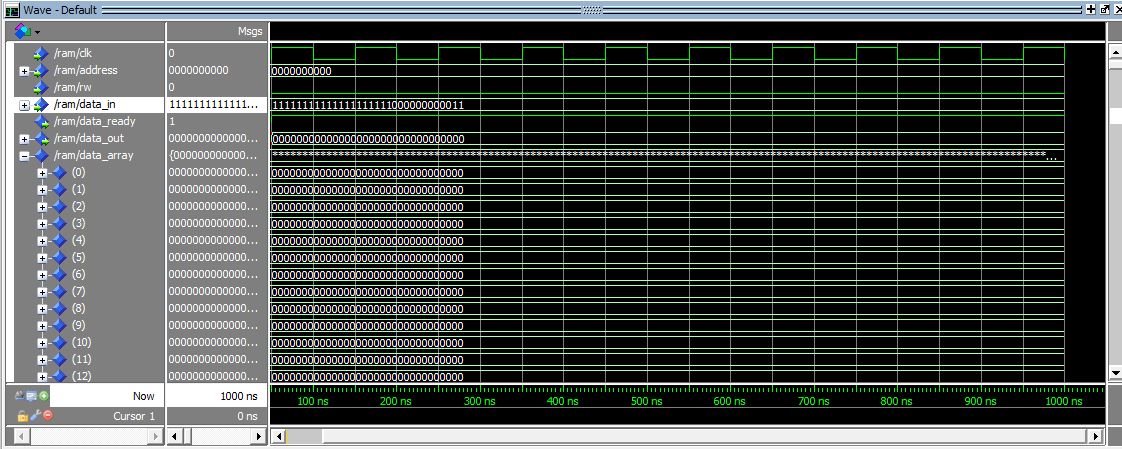
**hit**

i=0 , rw = 0 , rr = 0 ,

rw0 = w0\_valid , rc = 1 ,

wr = 0

کارکرد ram :

این ماؤول همانند data array عمل می کند ولی تنها سیگنال کنترلی آن همان rw است که در زیر شکل موج را به ازای حالت rw = 0 مشاهده میکنید :

در زیر شکل موج به ازای حالت rw = 1 می بینید که باید دیتا در آدرس مربوطه نوشته شود :

