**2.6 Cache高速缓冲**

**2.6.1 设计过程**

这部分只实现了cache的控制器模块，实际上还没有设计专门的cache寄存器模块。

首先定义cache行的格式和内存地址的格式：内存地址为12位，其中[1:0]是块内偏移，[6:2]是索引，[11:7]是Tag。Cache行为cache=V+D+Tag+Data=1+1+5+128=135位。且Cache缓存中cache大小为32块，主存大小为1024块，1块=4字，1字=32bit。

cache控制状态机分别有4个状态，IDLE为空闲状态， CompareTag为通过Tag比较判断是否命中，执行两种情况的状态，Allocate为没有命中时的情况从内存取出数据放到cache里面，然后更新修改位D和有效位V。在WriteBack阶段执行将cache的修改行写入到内存中。

其中写回的时候，只有当没有命中且查询cache的那一行的修改位为1时，才将原本的cache行写入到内存里。然后去内存里面通过cpu发送过来的tag和index寻找cache行放入cache中，再次命中cpu即可取出。

测试方面，在这个部分中需要测试6种情况，分别是写未命中（writemiss），写命中（writehit），读命中（readhit），读未命中（readmiss），读写回（readdirtymiss），写写回（writedirtymiss）。在test.v里面测试。

这里我们主要是使用cache第一行和第二行以及cache第一行对应于内存里面的第二块进行测试。且设置cache第二行第一个对应的内存里面存的是1，其余内存里面都是0，也就是mem[4]=32’d1。首先是writemiss部分，此时cache行未命中，首先通过cpu那边给出的tag和index将内存里面的那一行数据都取出，然后放到cache行里，并将有效位设置为1，修改位设置为0，然后将cpu那边给出的数据写入对应cache行里面，并且将修改位设置为1.，先使用以下代码测试。

|  |
| --- |
| WriteMiss:begin                cpu\_req\_rw<=1'b1;                cpu\_req\_addr<=12'd0;                cpu\_data\_write<=32'd8;                cpu\_req\_valid<=1'b1;  end |

cpu\_req\_rw表示写有效， cpu\_req\_addr表示CPU发过来的内存访问地址，cpu\_data\_write表示CPU写入内存的数据，cpu\_req\_valid<=1'b1表示从cpu发过来的这些数据和信号有效，即cache控制器中状态机开始从IDLE空闲状态转变为下一状comparetag。

然后是writehit部分：

|  |
| --- |
| WriteHit:begin                cpu\_req\_rw<=1'b1;                cpu\_req\_addr<=12'd0;                cpu\_data\_write<=32'd9;                cpu\_req\_valid<=1'b1;               End |

接下来是readmiss：

|  |
| --- |
| ReadMiss:begin                cpu\_req\_rw<=1'b0;                cpu\_req\_addr<=12'd4;                cpu\_data\_write<=cpu\_data\_write;                cpu\_req\_valid<=1'b1;              End |

readhit 部分：

|  |
| --- |
| ReadHit:begin                cpu\_req\_rw<=1'b0;                cpu\_req\_addr<=12'd0;                cpu\_data\_write<=cpu\_data\_write;                cpu\_req\_valid<=1'b1;              End |

writedirtymiss部分：

|  |
| --- |
| WriteDirtyMiss:begin                         cpu\_req\_valid<=1'b1;                         cpu\_req\_rw<=1'b1;                         cpu\_data\_write<=32'd6;                         cpu\_req\_addr<=12'd128;                     end |

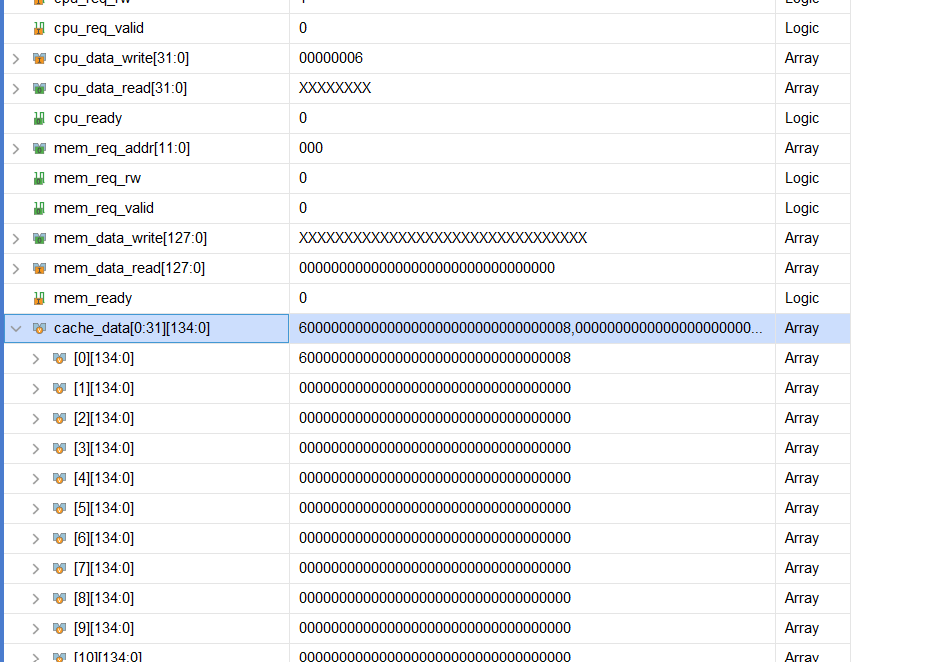
readDirtyMiss部分：

|  |
| --- |
| ReadDirtyMiss:begin                         cpu\_req\_valid<=1'b1;                         cpu\_req\_rw<=1'b0;                         cpu\_req\_addr<=12'd0;                         cpu\_data\_write<=cpu\_data\_write;                    End |

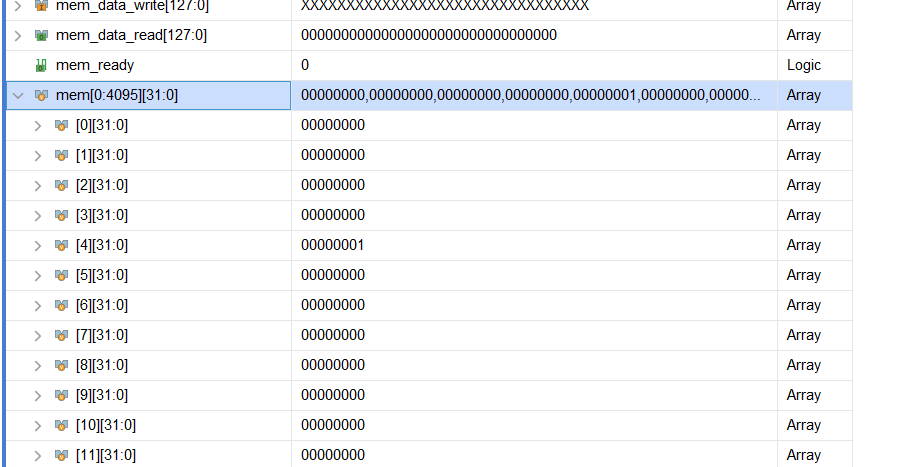
**2.6.2 实验现象及分析**

（一）readmiss的情况

（1）Cache部分



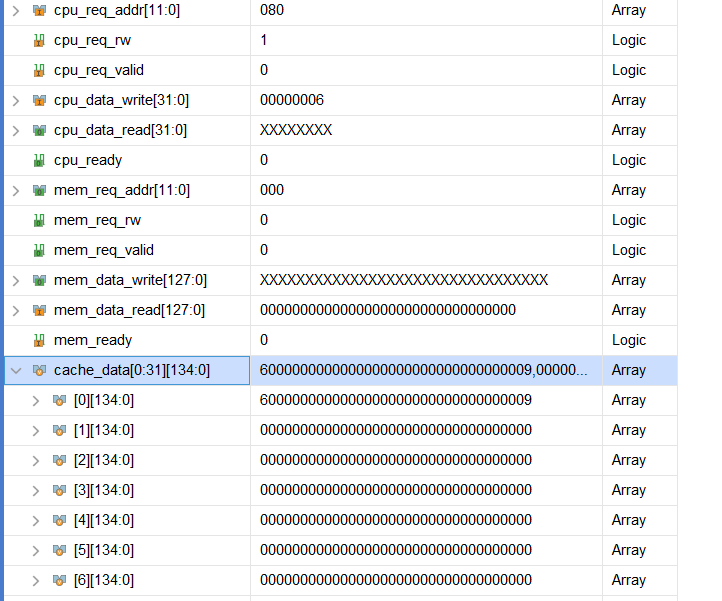
（2）Mem部分



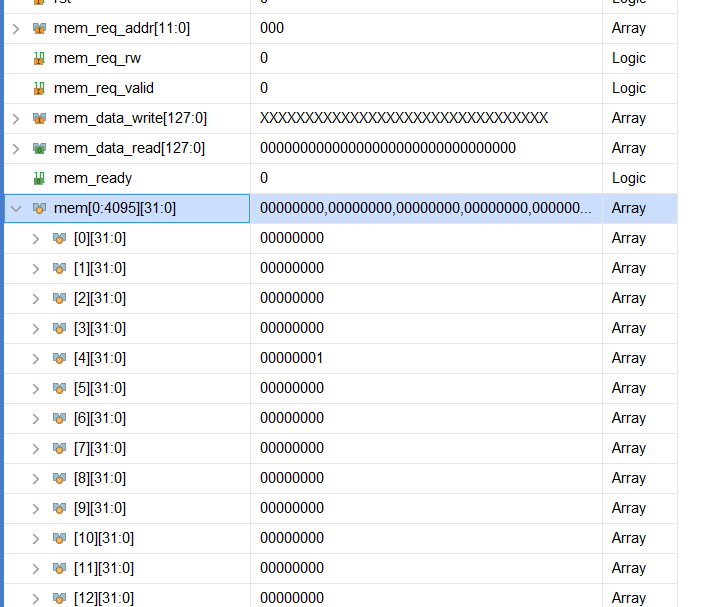
此时cache第一行中6位0110，由于我们是135位cache行所以最高位0无效，第一位为1表示V=1,有效，第二位为1表示D=1，已修改，最后为8表示数据8以及写入，此时cache行第一个字对应与mem[0]，此时并未写入。

（二）命中的情况

（1）Cache部分



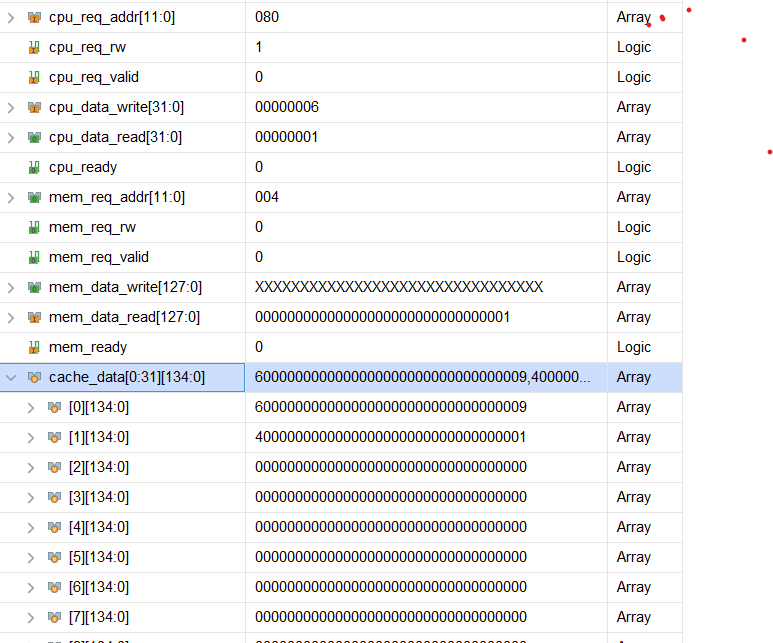
（2）Mem部分



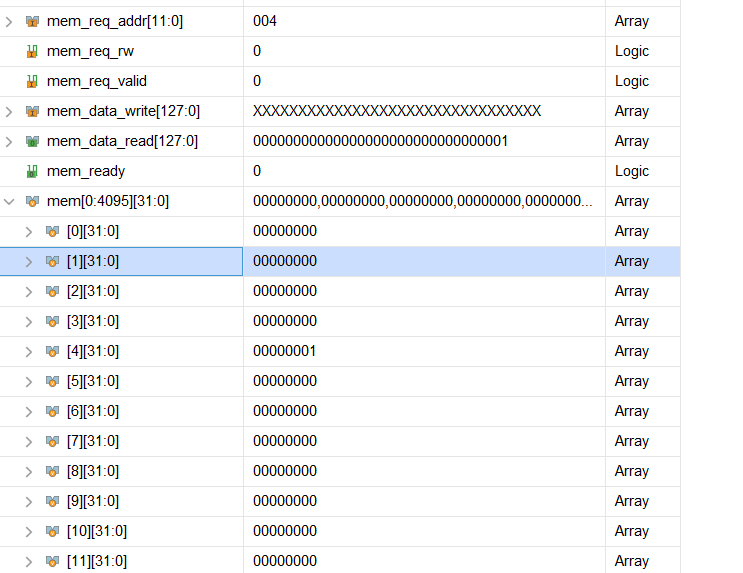
此时cache中的数据部分值修改为了9，此时V和D均为1，但是没有写回到mem中，因为此时tag是命中的。只有当不命中的时候且cache那一行是修改过的时候才写回。

（三）Readmiss的情况

（1）Cache部分



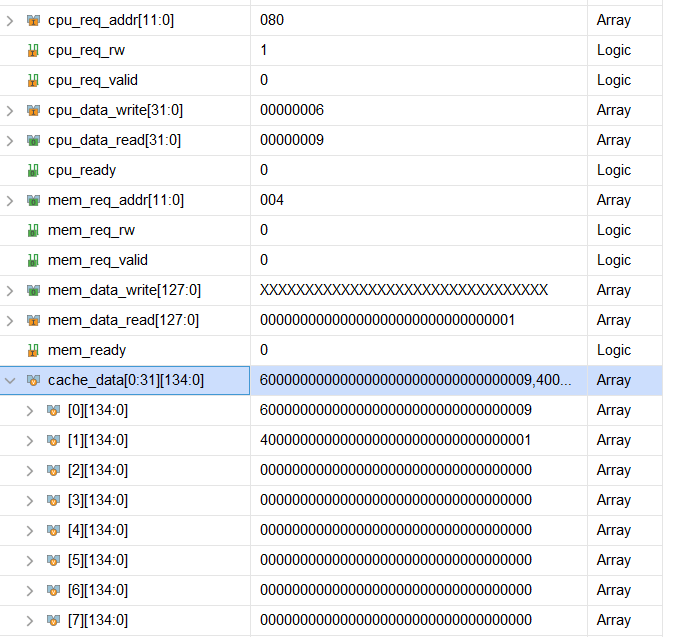
（2）Mem部分



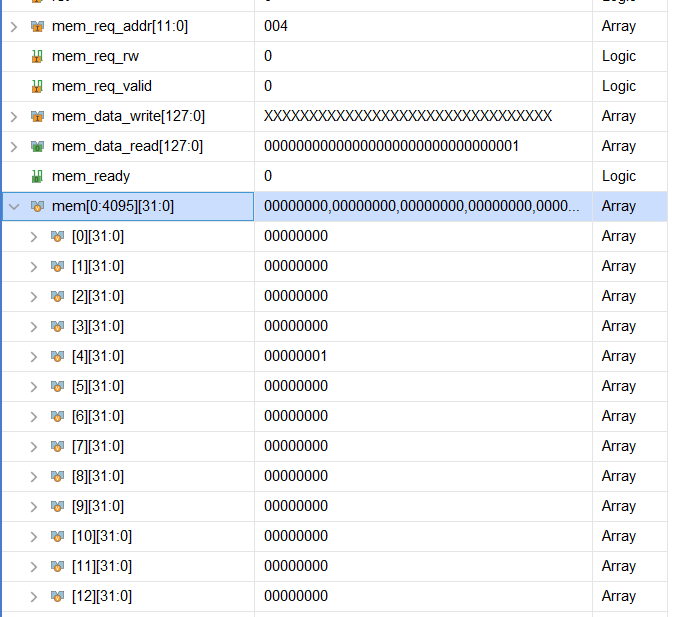
此时从mem[4]那里将数据读取到了cache第二行，4表示0100，即有效位为1，修改位为0。

（四）readhit的情况

（1）Cache部分



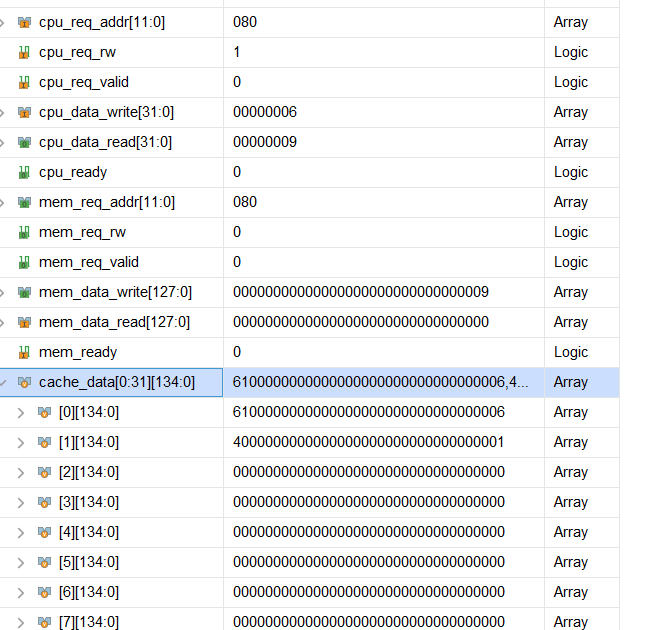
（2）Mem部分



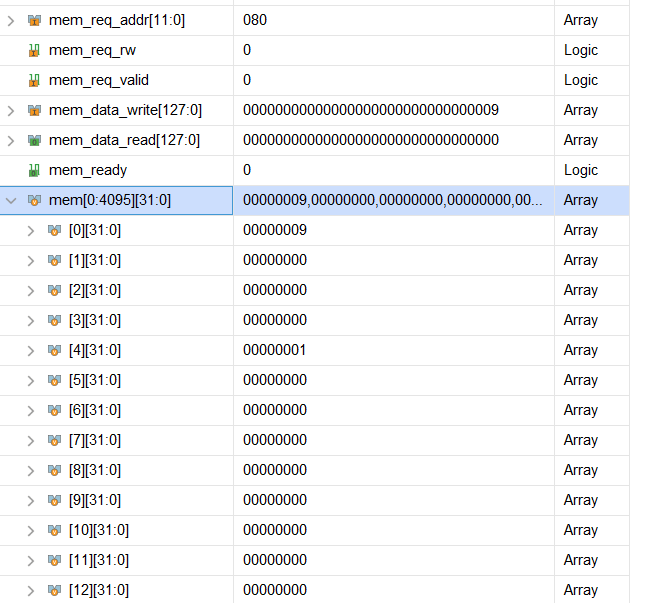
和上一次的结果一样，其实这里不太好看是否命中，不过可以从cpu\_data\_read那得到cpu接受的数据是9，即命中。

（五）writedirty的情况

（1）Cache部分



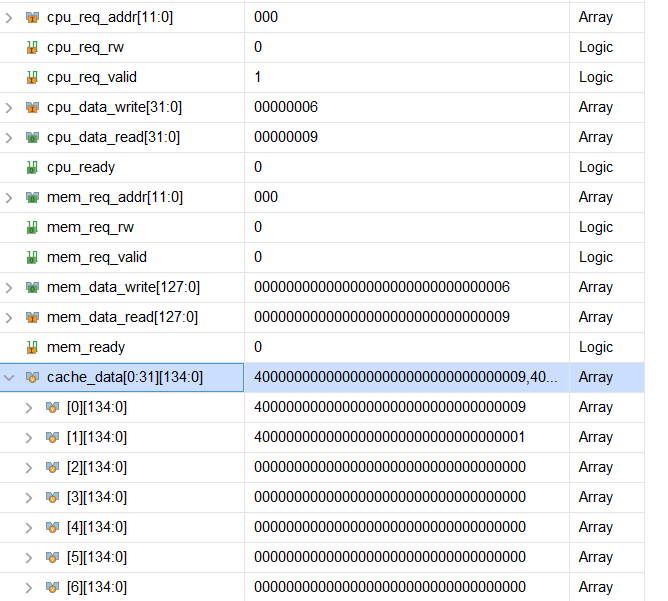
（2）Mem部分



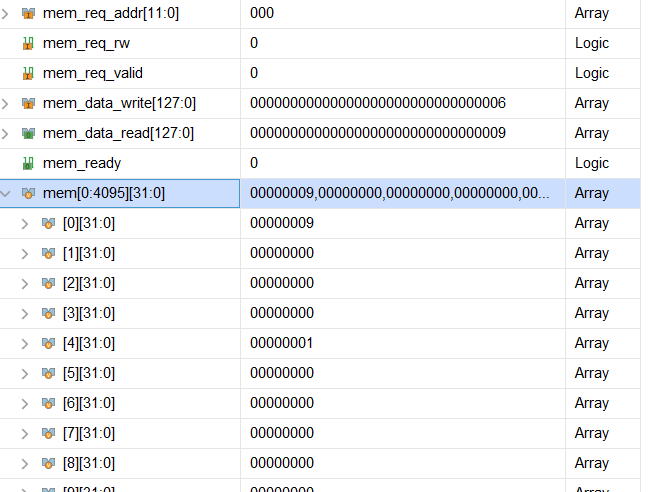
此时cache第一行的数据写回了mem中，第一个字的数据为9，且cache第一行的数据为6，为新写入的数据，此时写入的cache第一行的第一个字的数据对应于mem[128]。

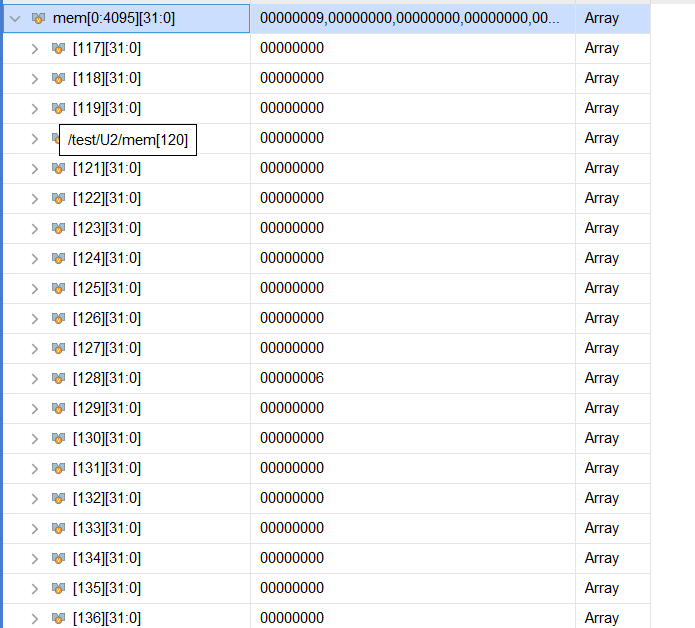
（六）readdirtymiss的情况

（1）Cache部分



（2）Mem部分





首先此时cache第一行写的数据对应于mem[128]，由于我们此时读取的数据地址为12’d0即mem[0]的数据，且上一次即在writedirtymiss那写入过，此时修改位为1，所以得需要将cache第一行写入到mem[128:131],再从内存里面取出mem[0]=9,然后将它放到cache第一行，并将修改位设置为1，所以cache第一行此时的数据中4表示只有有效位为1,9表示将mem[0]取出。mem[128]=6表示已经将数据写入到内存中。