

计算机组成原理

四路组相连 cache

设

计

报

告

一、总体设计

设计语言：Verilog 硬件描述语言

仿真工具：Vivado 2020.1

作图工具：亿图图示

映射机制：四路组相连映射、直接相连映射

输入输出：

输入端口	
clk	时钟信号
rst	复位信号
addr	物理地址

输出端口	
r_data	读取数据
r_addr	读地址
cout	命中次数
hit	命中标志

二、实现原理

设计规格：cache 共 128 槽，每一槽为 16B，采用 4 路组相连，数据位宽为 32bit。故 cache 数据区大小为 2KB。四路组相连映射和直接相连映射的区别体现在 cache 地址上主要是各部分（index，tag

和偏移地址) 的位数划分不同。下面对两种方式分别进行说明。

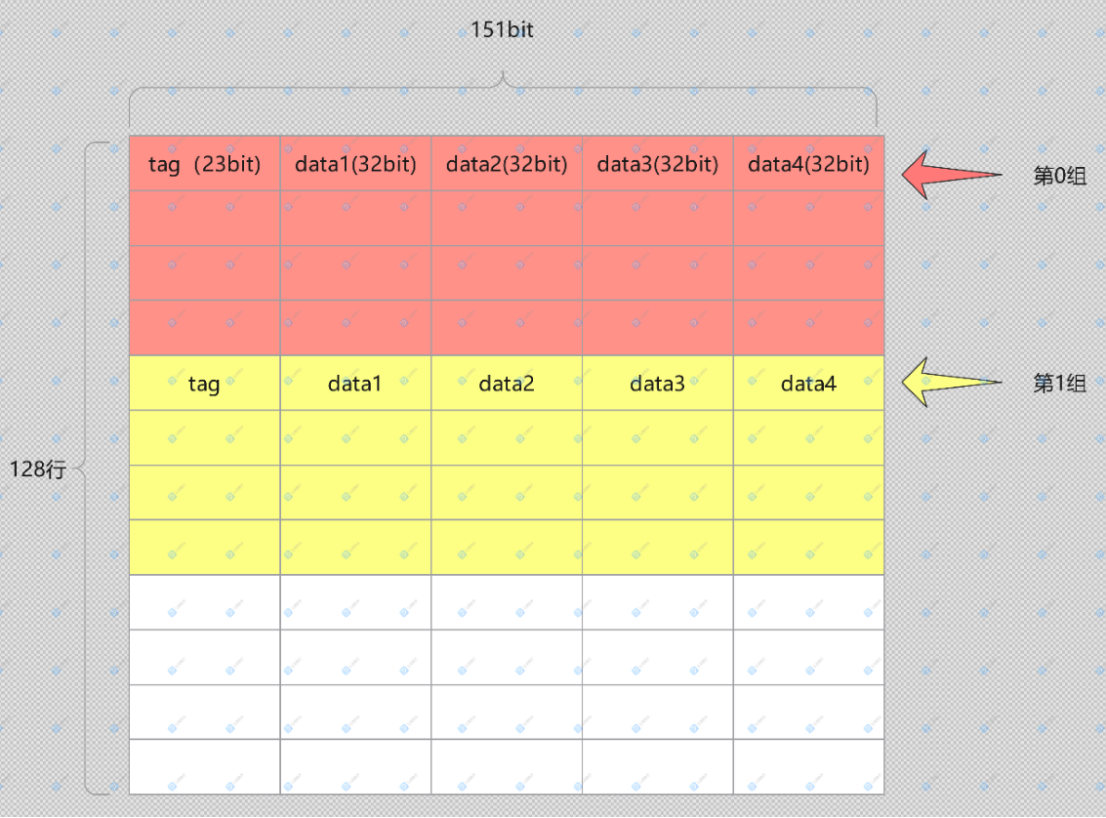
(一) 四路组相连映射

组数: $128/4=32$ 组, 故 index 位宽为 5bit

每一槽为 16B, 故组内偏移量为 4bit

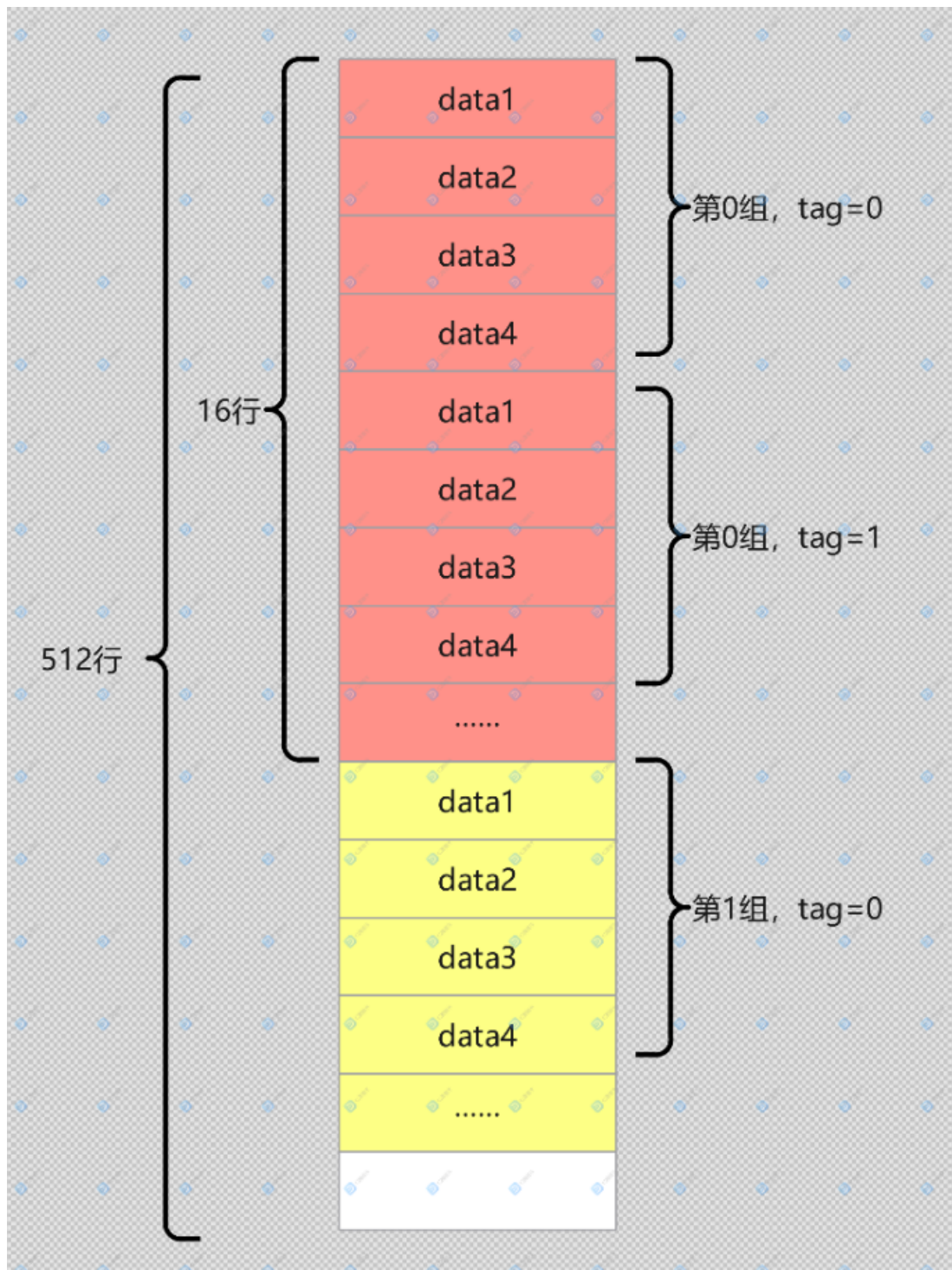
计算出 tag 位宽为: $32-5-4=23\text{bit}$

故 cache 深度为 128, 宽度为 151bit



由于 tag 的数值生成以及赋值、比对等操作较为复杂, 这里将每一组的 tag 简化为 0~3, 用于区分每个组内的四个数据。

因此 cache 逻辑图转变为下图, 其思想和二维数组的实现类似, 对一维存储空间进行不同层次的划分实现逻辑上的二维存储。



根据此逻辑可得出物理地址映射的算法：

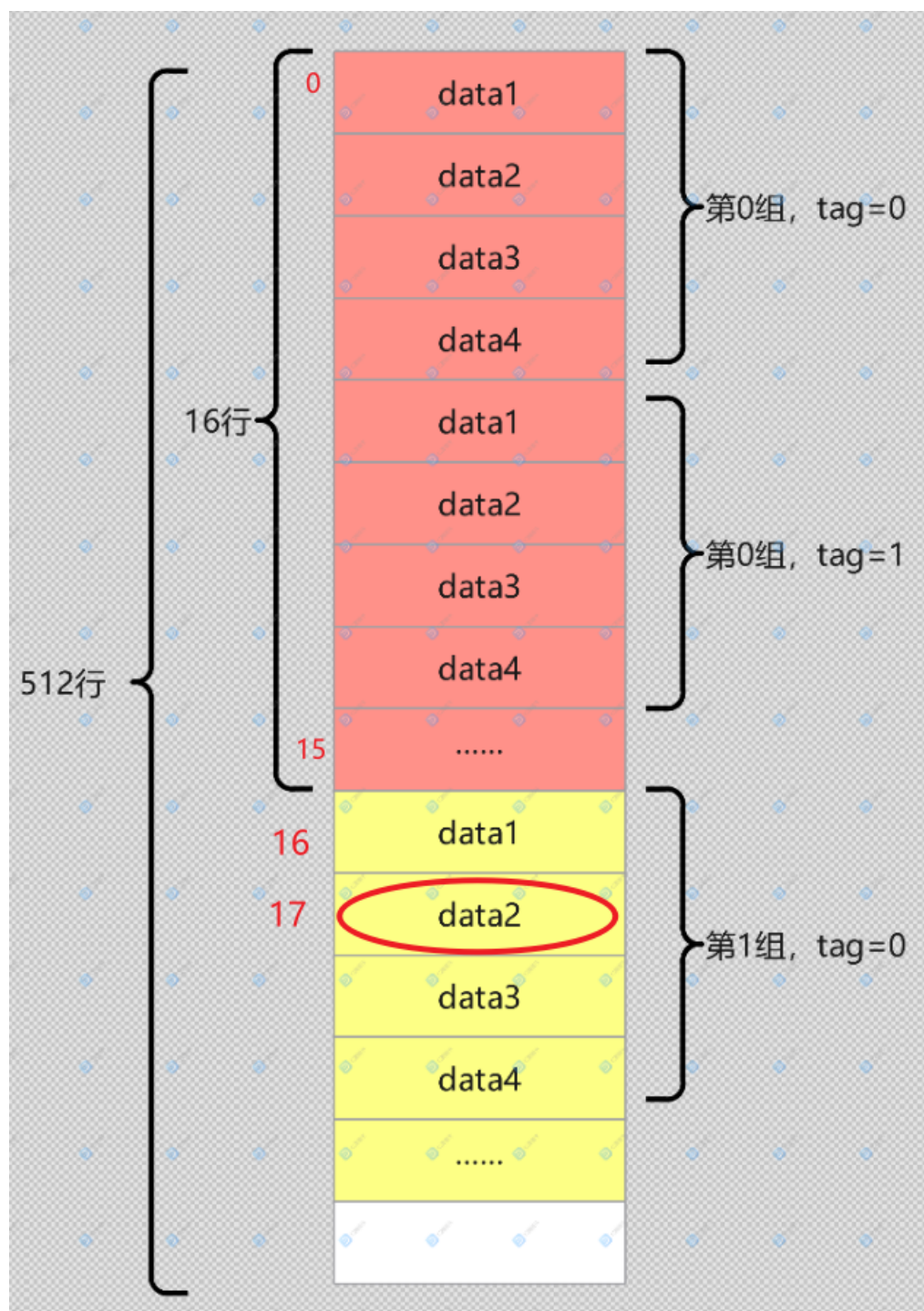
$$\text{cache_addr} = \text{index} * 16 + \text{tag} * 4 + \text{块内偏移} / 4$$

假设传入的物理地址为 000000000000000000000000 00001 0100

则该数据的 index=1, tag=0, 块内偏移=4

所取数据为第 1 组 tag=0 中的第 1 个数据（注：第 x，x 均从 0 开始计数）

对应 cache 中的第 17 行

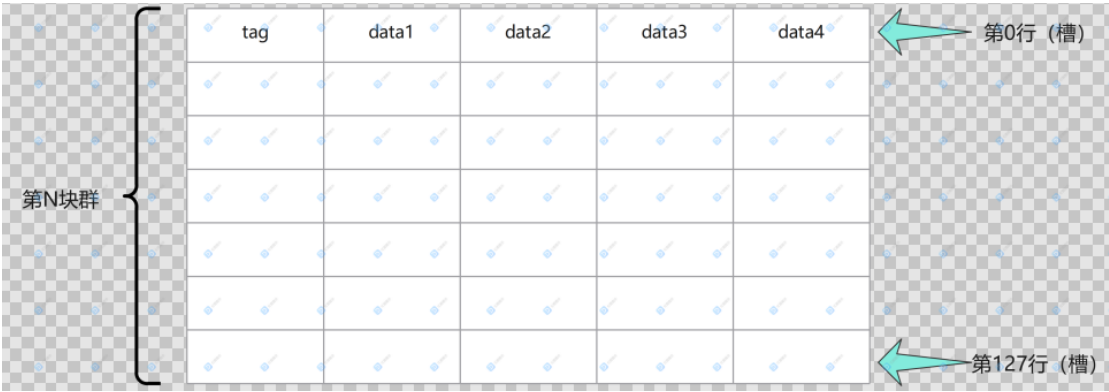


（二）直接相连映射

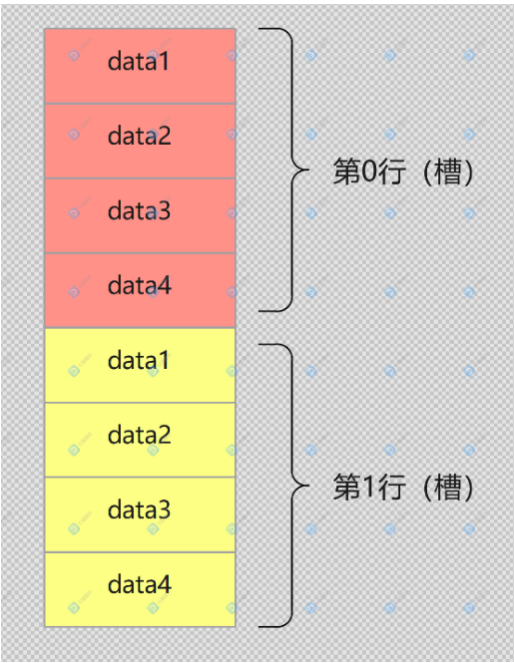
与四路组相连映射一样，块内偏移地址为 4bit，因为 cache 有

128 槽，故 index 有 7bit，剩余 tag 位有 $32-4-7=21\text{bit}$ 。

实际上，tag 即为主存根据块大小划分出的第 N 块群（N 从 0 开始计数），和四路组相连类似，默认 tag 为 $0\sim 127$ 顺序排列，故可省略对 tag 的比对操作。由于这里没有考虑主存的限制，故实现时对非负整数的 tag 的范围不作限制。



和四路组相连的思想一致，直接相连也采用一维存储方式进行实现



直接映射又叫模映射，映射关系如下：

$$\text{cache 行号} = \text{主存块号} \bmod \text{cache 行数}$$

根据此逻辑可得出物理地址映射的算法：

$$\text{cache_addr} = (\{\text{tag:index}\} \% 4) * 4 + \text{块内偏移} / 4,$$

实际上，因为 index 已是 cache 行号，故上式可简化为：

$$\text{cache_addr} = \text{index} * 4 + \text{块内偏移} / 4, \text{ 通过验证，两者的结果一致。}$$

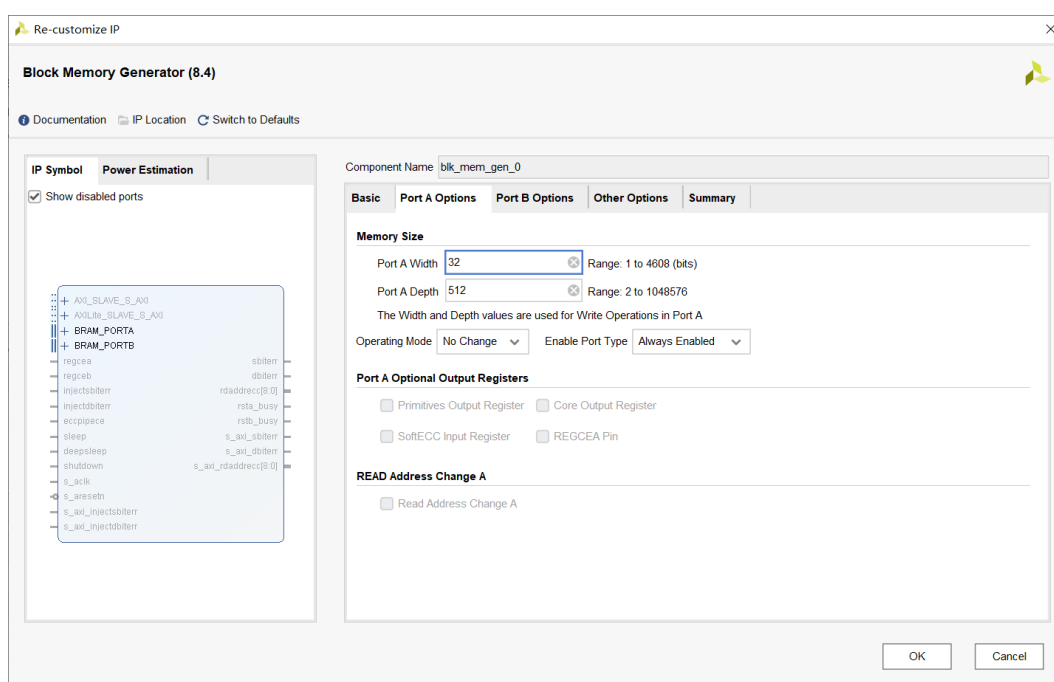
三、具体实现

在 cache 文件中实例化 IP 核 BRAM，输入端口有时钟信号 (clk)，复位信号 (rst)，以及物理地址 (addr)，通过对物理地址 addr 操作得出 cache 中需要进行读写的地址 r_addr 和 w_addr。

将得到的读写地址以及控制信号传入 BRAM 中。中间对是否命中 hit 进行判断并记录命中次数 cout。

1. IP 核设置

BRAM 的 Port Width 设置为 32bit，Port 的 Depth 设置为 512，cache 的内容使用 coe 文件进行写入。



其中，coe 文件中的数据使用 java 程序生成相应数量的二进制


数并将其写入文件。

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

public class Verilog_tb {

    public static void main(String[] args) throws IOException {
        Random rand = new Random();
        int data;
        String bs; //转化为二进制字符串

        FileWriter fw = new FileWriter("test.txt");
        for(int i = 0; i < 503; i++) { //cache共装入512个数据, coe文件中有9个, 这里生成503个就好
            data = rand.nextInt(1000)+1; //生成1~1000的随机数
            bs = Integer.toBinaryString(data);
            fw.write(bs+", "+"\\n"); //写入文件, 每个数字用英文逗号分隔
            System.out.println(bs);
            fw.flush();
        }
        fw.close();
    }
}
```

 test.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
100110000,
1011101100,
101011111,
10111,
100001100,
101010011,
1010011001,
110000010,
11100000,
10010000,
1011000111,
110001010,
101100111,
1010001000,
1001101101,
```

2. 物理地址映射到 cache 地址

四路组相连

当触发信号到来时进行条件判断:

复位信号是否有效;

addr[31:11]即 tag, 因该设计中 tag 的取值为 0,1,2,3, 故将

其 ≥ 4 作为判断条件;

addr[3:0]即块内偏移地址,每槽大小为 16B,数据位宽为 32bit,
故每槽有 4 个数据,因此块内地址偏移量的取值只能为 0,4,8,12

根据物理地址映射的方法得到 cache 的读写地址。

```
always @(*) begin
    if(!rst || addr[31:11]>=4 || (addr[3:0]!=0&&addr[3:0]!=4&&addr[3:0]!=8&&addr[3:0]!=12))begin
        w_addr<=0;
        r_addr<=0;
        hit<=0;
    end
    else begin
        w_addr<=addr[10:4]*16+addr[31:11]*4+addr[3:0]/4;
        r_addr=w_addr;
        hit<=1;
    end
end
```

直接相连

同样判断复位信号是否有效,不同的是,这里的判断条件中没有 tag 进行限制。

```
always @(*) begin
    if(!rst || (addr[3:0]!=0&&addr[3:0]!=4&&addr[3:0]!=8&&addr[3:0]!=12))begin
        w_addr<=0;
        r_addr<=0;
        hit<=0;
    end
    else begin
        w_addr<=addr[10:4]*4+addr[3:0]/4;
        r_addr=w_addr;
        hit<=1;
    end
end
```

该计算方法与下面的计算方法效果相同

```

always @(*) begin
    if(!rst || (addr[3:0]!=0&&addr[3:0]!=4&&addr[3:0]!=8&&addr[3:0]!=12))begin
        w_addr<=0;
        r_addr<=0;
        hit<=0;
    end
    else begin
        w_addr<=(addr[31:4]%4)*4+addr[3:0]/4;
        r_addr=w_addr;
        hit<=1;

    end
end

```

3. 生成 cout 和 hit 信号

使用寄存器 cout 进行命中次数的计数，因 cache 共 512 槽，故 cout 的数据位宽为 9bit，首先利用复位信号，当复位信号的下降沿到来时，将计数初始化为 0

```

reg [8:0] cout;
always@(negedge rst) begin
    cout<=0;
end

```

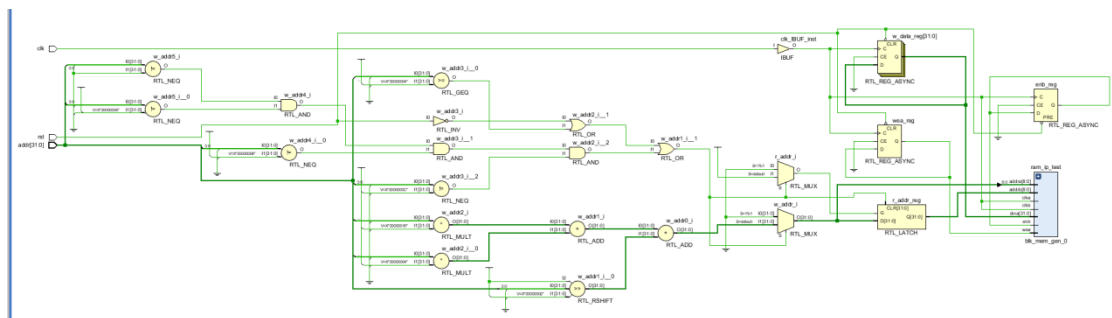
把输入地址作为触发信号，当地址发生变化时，判断复位信号是否有效，若复位无效，则进行是否命中判断，若命中 (hit==1)，则 cout+1

```

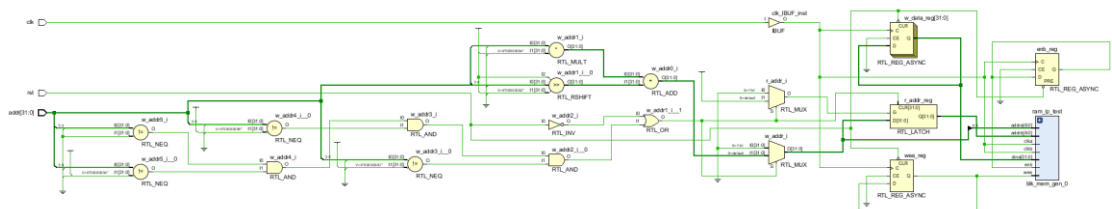
always@(addr) begin      //posedge hit or negedge clk or negedge rst
    if(!rst)
        begin
            cout<=0;
        end
    else
        begin
            if(hit)
                begin
                    cout<=cout+1;
                end
            end
        end
    end
end

```

总体电路图如下



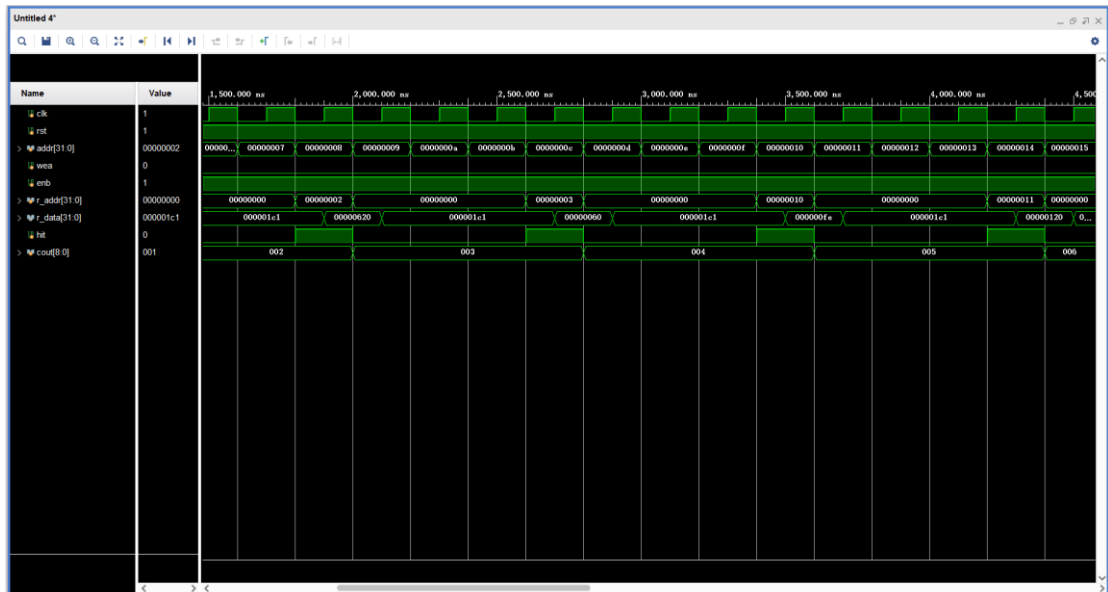
(四路组相连映射)



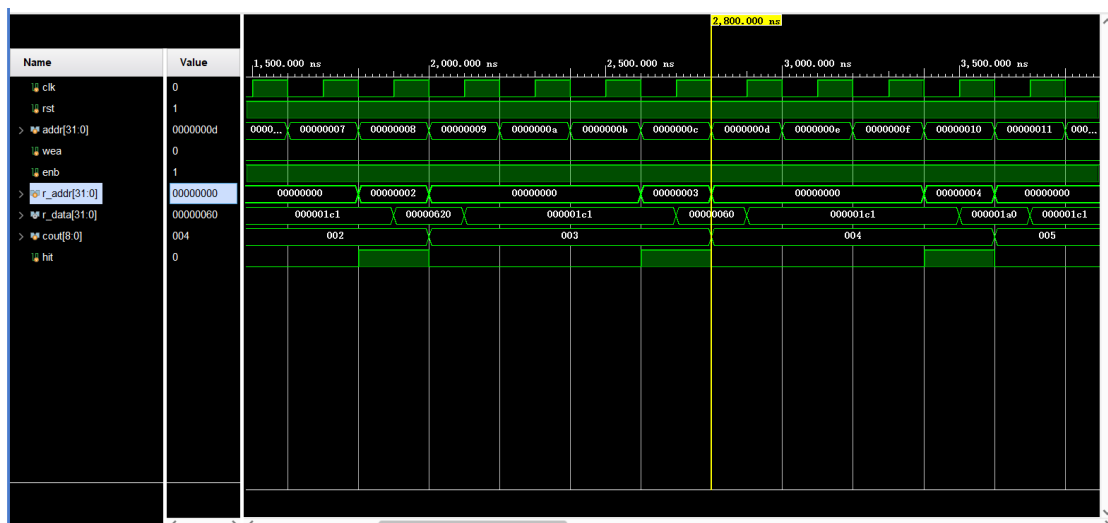
(直接相连映射)

四、仿真实现

行为仿真：

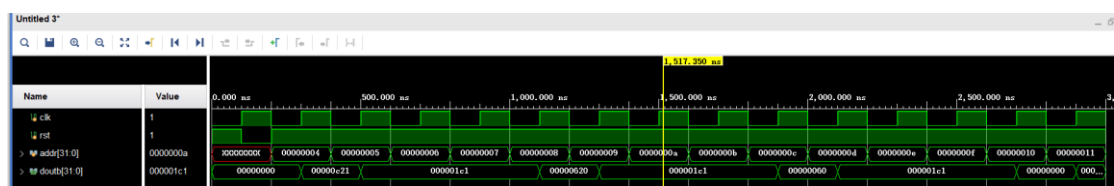


(四路组相连映射)

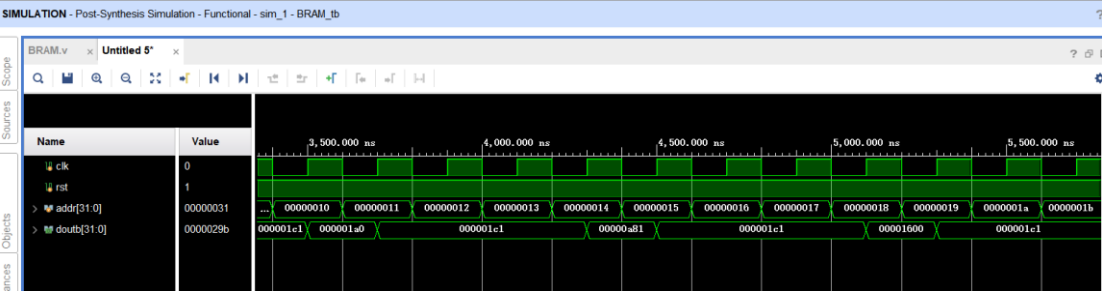


(直接相连映射)

功能仿真：



(四路组相连映射)



(直接相连映射)