

组成原理实验课程第 五 次实验报告

实验名称	存储器实现			班级	李涛老师
学生姓名	辛浩然	学号	2112514	指导老师	董前琨
实验地点	实验楼 A 区 304			实验时间	2023/5/23

1 实验目的

1. 了解只读存储器 ROM 和随机存取存储器 RAM 的原理。
2. 理解 ROM 读取数据及 RAM 读取、写入数据的过程。
3. 理解计算机中存储器地址编址和数据索引方法。
4. 理解同步 RAM 和异步 RAM 的区别。
5. 掌握调用 xilinx 库 IP 实例化 RAM 的设计方法。
6. 熟悉并运用 verilog 语言进行电路设计。
7. 为后续设计 cpu 的实验打下基础。

2 实验内容说明

1. 调用 xilinx 库 IP 实例化同步 RAM 和 ROM ；
2. 自行搭建的异步 RAM 和 ROM 。

3 实验原理图

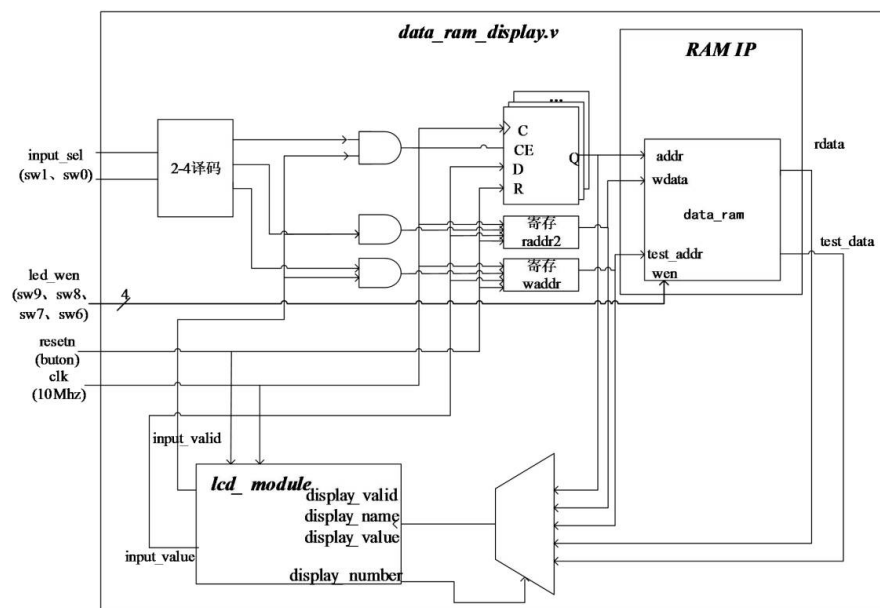


图 1: data_ram 参考设计的顶层模块框图

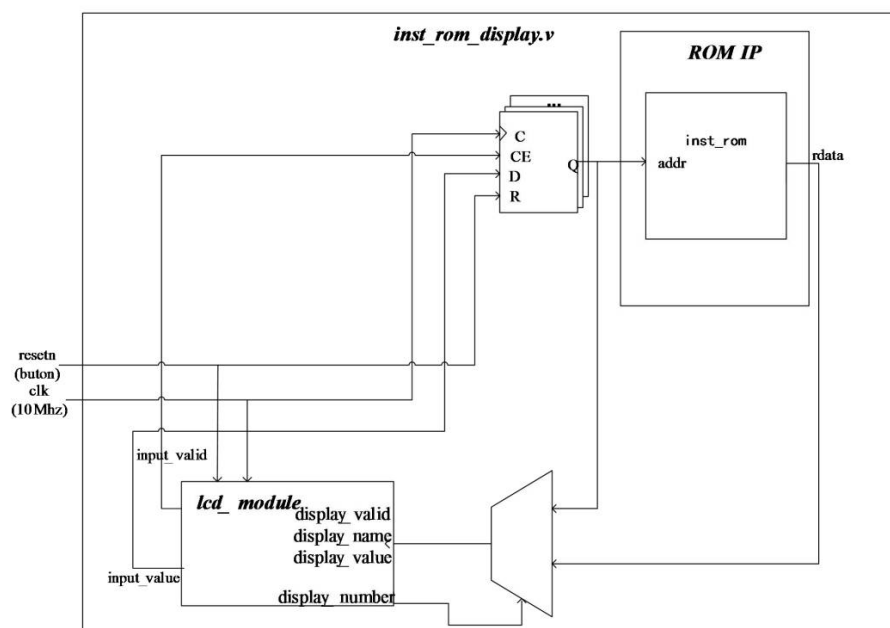


图 2: inst_rom 参考设计的顶层模块框图

4 实验步骤

4.1 同步 ROM

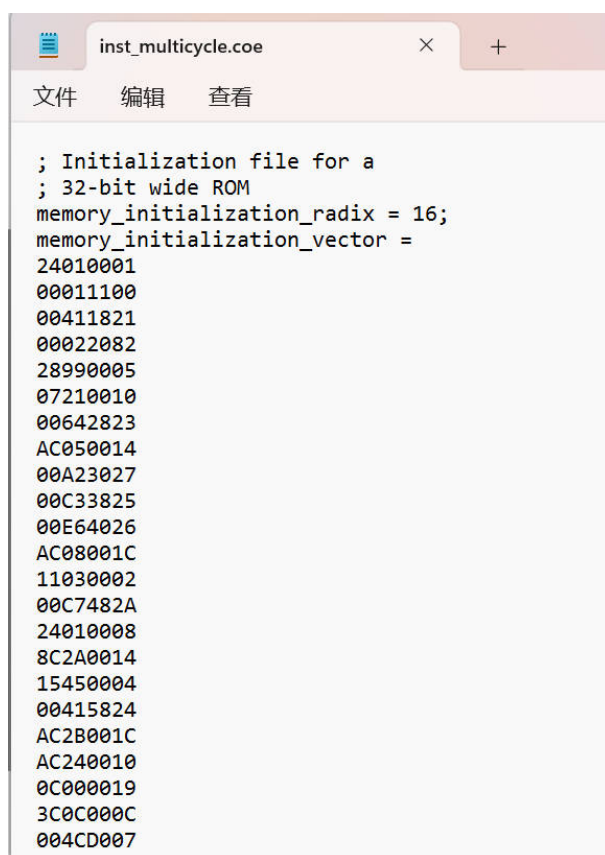
ROM 为只读存储器，需要初始化内部数据，可作为指令存储器。

同步 ROM 的读取操作是与时钟信号同步的。当时钟信号上升沿到达时，ROM 从指定的地址

读取数据，并将其输出。只能对数据进行读取，不能对数据进行修改。

实际读取数据的地址的规则是：将输入的 32 位数的前 30 位与 00 拼接，拼接得到的 32 位数即为待读取数据存放的地址。如输入 00000005h，实际读取地址为 00000004h。

在用 IP 核生成的同步 ROM 中，通过 inst_multicycle.coe 文件初始化内部数据，如图所示。

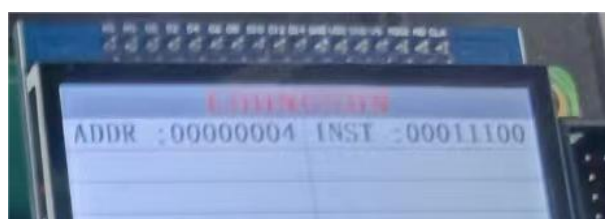


```
; Initialization file for a
; 32-bit wide ROM
memory_initialization_radix = 16;
memory_initialization_vector =
24010001
00011100
00411821
00022082
28990005
07210010
00642823
AC050014
00A23027
00C33825
00E64026
AC08001C
11030002
00C7482A
24010008
8C2A0014
15450004
00415824
AC2B001C
AC240010
0C000019
3C0C000C
004CD007
```

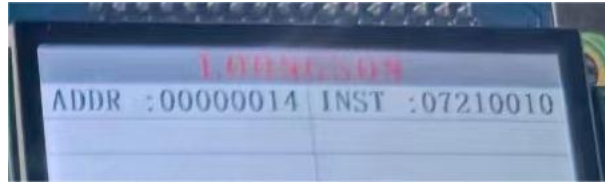
第一行指定了初始化数据格式，此处为 16 进制，也可以设置为 2 进制。第二行说明从第三行开始为初始化的数据向量，由于 ROM 宽度为 32 位，故一个初始化向量为 32 位数据。初始化向量之间必须用空格或换行符隔开，此处使用换行符，故一行为一个初始化向量。初始化数据会从 ROM 中的 0 地址处开始依次填充。

以下是上箱验证：

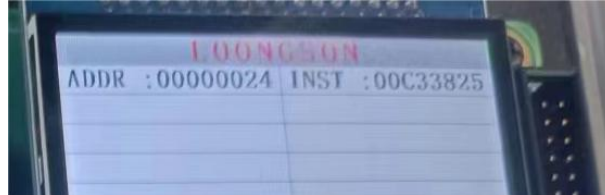
读取地址为 4h 处的数据，为第 1 个数据，结合装载的初始化文件，数据确为 00011100h，读取正确。



读取地址为 14h 处的数据，为第 5 个数据，结合装载的初始化文件，数据确为 07210010h，读取正确。



读取地址为 24h 处的数据，为第 9 个数据，结合装载的初始化文件，数据确为 00C33825h，读取正确。



4.2 异步 ROM

自行搭建异步 ROM，异步 ROM 同样需要初始化内部数据，可作为指令存储器。

与同步 ROM 不同，异步 ROM 的读取操作是异步的，不需要时钟信号。读取操作由外部电路触发，并且在触发信号到达时立即输出所请求的数据。由于异步读取的特性，异步 ROM 的输出具有延迟。在读取请求发出后，ROM 需要一定的时间来访问并输出所请求的数据。但在实验验证中延迟几乎观察不到。

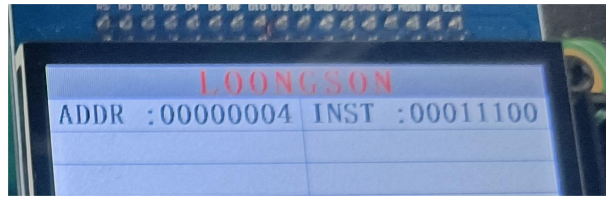
同样异步 ROM 只能对数据进行读取，不能对数据进行修改。实际读取数据的地址的规则与同步 ROM 相同：将输入的 32 位数的前 30 位与 00 拼接，拼接得到的 32 位数即为待读取数据存放的地址。

异步 ROM 通过.v 文件初始化内部数据，如图所示。

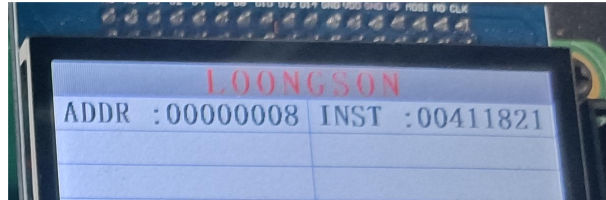
```
assign inst_rom[ 0] = 32'h24010001;//00H: addiu $1,$0,#1 | $1 = 0000_0001H
assign inst_rom[ 1] = 32'h00011100;//04H: sll   $2,$1,#4 | $2 = 0000_0010H
assign inst_rom[ 2] = 32'h00411821;//08H: addu  $3,$2,$1 | $3 = 0000_0011H
assign inst_rom[ 3] = 32'h00022082;//0CH: srl   $4,$2,#2 | $4 = 0000_0004H
assign inst_rom[ 4] = 32'h00642823;//10H: subu  $5,$3,$4 | $5 = 0000_000DH
assign inst_rom[ 5] = 32'hAC250013;//14H: sw    $5,#19($1)| Mem[14H] = DH
assign inst_rom[ 6] = 32'h00A23027;//18H: nor   $6,$5,$2 | $6 = FFFF_FFE2H
assign inst_rom[ 7] = 32'h00C33825;//1CH: or    $7,$6,$3 | $7 = FFFF_FFF3H
assign inst_rom[ 8] = 32'h00E64026;//20H: xor   $8,$7,$6 | $8 = 0000_0011H
assign inst_rom[ 9] = 32'hAC08001C;//24H: sw    $8,#28($0)| Mem[1CH] = 11H
assign inst_rom[10] = 32'h00C7482A;//28H: slt   $9,$6,$7 | $9 = 0000_0001H
assign inst_rom[11] = 32'h11210002;//2CH: beq   $9,$1,#2 | 跳转到指令 34H
assign inst_rom[12] = 32'h24010004;//30H: addiu $1,$0,#4 | 不执行
assign inst_rom[13] = 32'h8C2A0013;//34H: lw    $10,#19($1)| $10 = 0000_000DH
assign inst_rom[14] = 32'h15450003;//38H: bne   $10,$5,#3 | 不跳转
assign inst_rom[15] = 32'h00415824;//3CH: and   $11,$2,$1 | $11 = 0000_0000H
assign inst_rom[16] = 32'hAC0B001C;//40H: sw    $11,#28($0)| Mem[1CH] = 0H
assign inst_rom[17] = 32'hAC040010;//44H: sw    $4,#16($0)| Mem[10H] = 4H
assign inst_rom[18] = 32'h3C0C000C;//48H: lui   $12,#12 | [R12] = C_0000H
assign inst_rom[19] = 32'h08000000;//4CH: j     00H      | 跳转指令 00H
```

以下是上箱验证：

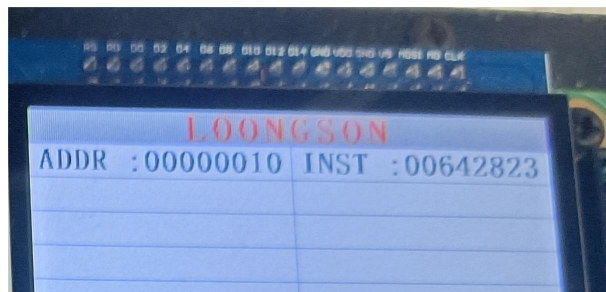
读取地址为 4h 处的数据，为第 1 条指令，结合初始化的内部数据，数据确为 00011100h，读取正确。



读取地址为 8h 处的数据，为第 2 条指令，结合初始化的内部数据，数据确为 00411821h，读取正确。



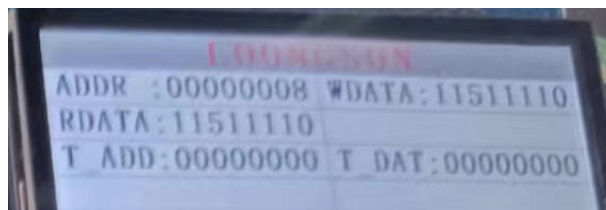
读取地址为 10h 处的数据，为第 3 条指令，结合初始化的内部数据，数据确为 00642823，读取正确。



4.3 同步 RAM

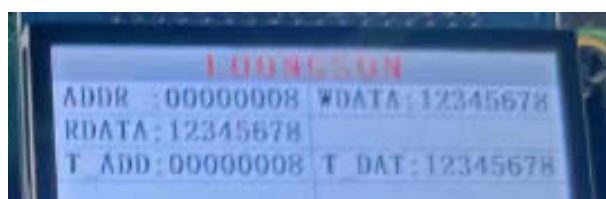
RAM 是一种随机存储器，它可以被读取和写入，用户可以在其中存储和修改数据。

通过 IP 核生成同步 RAM，在写使能信号有效的情况下，输入写地址和相应数据，可以将数据写入地址。同时可以在 RDATA 中可以读到写入的数据。经过上箱验证，实现了读取、写入数据的功能：



4.4 异步 RAM

异步 RAM 自行搭建。在写使能信号有效的情况下，输入写地址和相应数据，可以将数据写入地址。同时可以在 RDATA 中可以读到写入的数据。经过上箱验证，实现了读取、写入数据的功能：



5 总结感想

1.ROM 和 RAM 的区别：

功能：

- ROM 是只读存储器，数据初始化后，用户无法修改或者删除其中的内容。ROM 可以用于存储固定的指令集、固件和不经常变化的数据，如启动程序、操作系统指令等。
- RAM 是随机存储器，它可以被读取和写入，用户可以在其中存储和修改数据。RAM 可以用于临时存储正在运行的程序、数据和临时信息。

可写性：

- ROM 的数据是只读的，用户不能修改。
- RAM 的数据可以被用户读取和写入。

易失性：

- ROM 中的数据是非易失性的，它的数据是固定的，不可更改的。也就是说，即使断电，其中的数据也会被保留。
- RAM 其中的数据是易失性的，也就是说，一旦断电，其中的数据就会丢失。

访问速度：

- ROM 的访问速度通常较慢，因为它使用的是非易失性存储技术，其访问时间较长。
- RAM 的访问速度通常较快，因为它使用的是易失性存储技术，可以更快地读取和写入数据。

2. 分析一下同步存储器和异步存储器的特点，思考说明一下何时需要使用同步存储器，何时需要使用异步存储器。

同步存储器特点：

- **时钟同步：**同步存储器与系统时钟同步，操作在时钟的上升沿或下降沿进行，以确保数据的正确读写。
- **顺序读写：**同步存储器按照时钟信号进行顺序读写操作，每个操作需要等待时钟周期。
- **高速操作：**由于同步存储器按照时钟信号同步工作，减少了访问延迟其操作速度较快，适用于高速计算机系统。
- **容量较小：**同步存储器的容量相对较小，一般用于缓存或寄存器等需要快速访问的场景。
- **复杂性高：**同步存储器的设计和控制较为复杂，需要时钟同步和时序控制电路。
- 同步存储器具有较高的时序精确度和可靠性。

异步存储器特点：

- **无需时钟同步：**异步存储器不依赖系统时钟信号进行操作，可以根据输入信号的变化瞬时读写数据。
- **并行读写：**异步存储器支持并行读写操作，可以同时进行多个读写操作。
- **速度较慢：**由于异步存储器没有严格的时钟同步，读写操作可能需要更长的时间完成，因此相对于同步存储器来说，异步存储器的读写速度较慢。
- **容量较大：**异步存储器的容量相对较大，可以存储更多的数据。
- **简单性高：**异步存储器的设计和控制相对简单，无需时钟同步和复杂的时序控制电路。

同步存储器的适用情况：

- 同步存储器通常具有较快的读写速度和响应时间，适用于对存储器访问速度要求较高的应用，如高性能计算、实时系统等。
- 同步存储器可以确保数据在特定时钟边沿进行读写，保证数据的一致性。可适用于需要精确同步和协调操作的场景，如多处理器系统、并行计算等。

异步存储器的适用情况：

- 异步存储器相对于同步存储器来说，通常具有更低的功耗，因此在**低功耗要求**的系统中比较适用。例如，移动设备和嵌入式系统通常使用异步存储器以降低功耗。
- 异步存储器的控制电路相对简单，不需要复杂的同步协议。适用于对**控制复杂度要求较低**的应用，如简单嵌入式系统、低成本设备等。
- 由于异步存储器的设计相对简单，生产成本通常较低。适用于对**成本敏感**的应用，如消费电子产品、大规模存储系统等。