实验二　总线、RAM、汇编与仿真

姓名：　黄梓豪　　学号：　2023310103008

本文使用 LaTeX 构建。

1　实验目的

1. 搭建一个简单的 SoC。掌握在总线上添加外设的方法，理解 SoC 系统的启动过程，总线读写操作的原理。
2. 能编写简单的汇编程序，实现对 CPU 寄存器的控制，理解 CPU 运行的过程。

2　实验原理

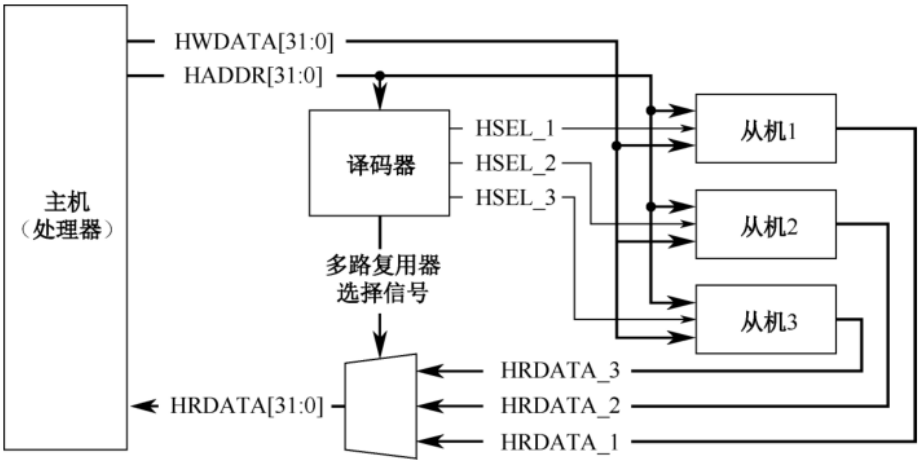

图 1: AHB-Lite 总线示意图

本实验搭建了一个基于 Cortex-M0 内核的简化 SoC。Cortex-M0 作为唯一总线主机，通过 AHB-Lite 总线访问片上存储器 RAMCODE 以及预留的外设地址空间。总线扩展模块 `bus_ext` 内部包含地址译码器和从机多路选择器：地址译码器根据 `HADDR` 产生各从机片选信号 `HSELx`，从机多路选择器根据片选结果选择对应从机的 `HRDATA`、`HRESP`、`HREADYOUT` 返回给 CPU。RAMCODE 被映射到统一地址空间的低地址区域，在仿真开始时通过 `$readmemh` 将 `code.hex` 预加载到 RAMCODE 中，作为 CPU 的指令和数据存储器。

实验采用 AHB-Lite 协议，一次传输分为地址阶段和数据阶段，并支持流水线。在地址阶段，CPU 在时钟上升沿输出 HADDR、HWRITE、HTRANS、HSIZE 等信号，同时由地址译码器产生相应的 HSELx。在数据阶段，写操作由 CPU 在 HWDATA 上提供写数据，读操作由从机在 HRDATA 上返回读数据，当 HREADY 为 1 时当前传输完成。若从机未准备好，可通过将 HREADYOUT 拉低插入等待周期，此时 HREADY 为 0，总线保持当前传输状态，直到 HREADY 再次为 1。

仿真开始时，先将 code.hex 加载到 RAMCODE，对指令存储空间初始化。释放复位后，Cortex-M0 按复位向量从 RAMCODE 中取出入口地址，设置 PC 并开始执行程序。在运行过程中，CPU 通过 AHB-Lite 不断从 RAMCODE 取指，完成取指、译码和执行循环；执行 LDR/STR 等指令时，CPU 通过总线对 RAMCODE 或其它映射区域发起读写访问。通过观察仿真波形中 HADDR、HWRITE、HWDATA、HRDATA、HREADY 等信号变化，可以清晰看出 CPU 复位启动、取指以及数据读写的时序过程。

3 实验过程

3.1 vis_r0_o

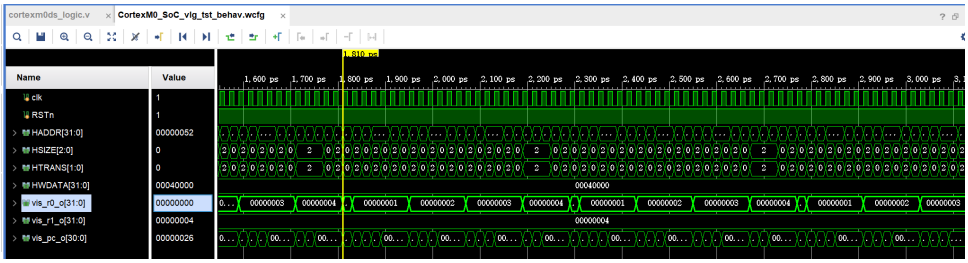

图 2: vis_r0_o 实验结果

由图可见，vis_r0_o 变化范围为 0 到 4

3.2 vis_pc_o

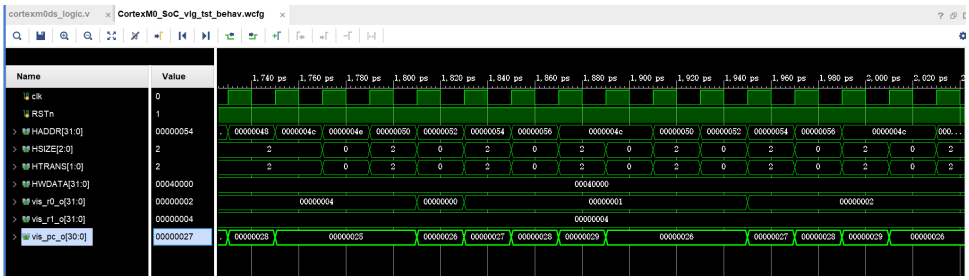

图 3: vis_pc_o 实验结果

由图可见，vis_r0_o 变化范围为 25 到 29。变化顺序如下：25-26–27-28-29-26-27-28-29-26-27-28-29-26-27-28。

在“code.txt”的.text 中有这样一段:

```
1     start
2         0x00000048:     2104          .!        MOVS      r1,#4
3         0x0000004a:     2000          .         MOVS      r0,#0
4         0x0000004c:     1c40          @.        ADDS      r0,r0,#1
5         0x0000004e:     4288          .B        CMP       r0,r1
6         0x00000050:     d0fb          ..        BEQ       0x4a ; start + 2
7         0x00000052:     d1fb          ..        BNE       0x4c ; start + 4
```

这表明, 程序运行时, 取指地址在 0x0000004a-0x00000052 之间循环, 不难发现, vis_pc_o 恰好是取指地址去除 LSB 后的结果.

3.3 AHB Lite 总线波形

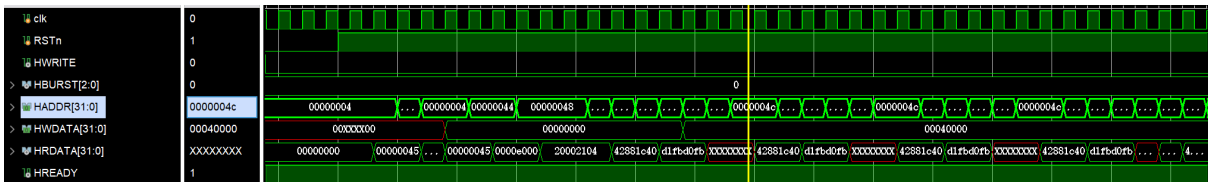

图 4: AHB Lite 总线波形

由图4，HWRITE 一直为低电平，说明一直是读操作；HBURST 一直为低电平，说明一直是单次传输；尤其注意到光标所指的位置，可见 HADDR $= 0x4c$ 时长两个 cycle，但是第一次上升沿结束后 HRDATA 仍然是不定态，说明此时从 RAM 读取数据还未准备好，直到第二个上升沿时 HRDATA 才变为有效数据 0x1c40，这说明 RAM 有一个 cycle 的读延迟. 这跟图5所强调的两个阶段：Address phase & Data phase 是一致的.

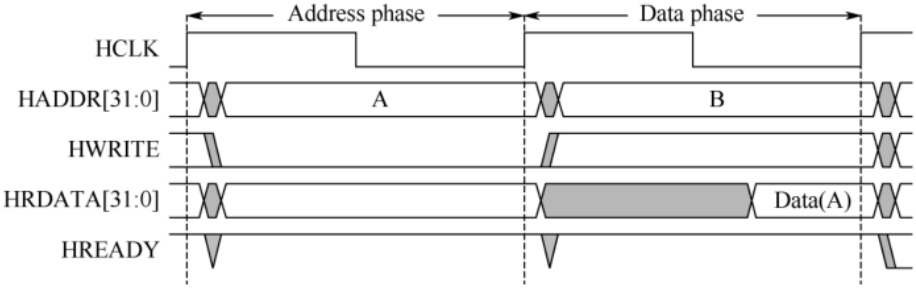

图 5: RAM 读基本波形

4 实验总结

通过本次实验，掌握了 AHB Lite 总线的基本读写操作，理解了总线与 RAM 的交互过程，熟悉了汇编指令的执行流程，并通过仿真验证了设计的正确性，为后续更复杂的 SoC 设计打下了坚实的基础。

同时，最近我的 Modelsim 软件出问题了，在助教的建议下使用 Vivado 自带的仿真工具进行仿真，虽然界面和操作习惯上有些不同，但是也能仿照指导书进行添加波形观察信号的操作，学习到了新方法。之前为了观测已经例化模块里的信号，我都是在 testbench 里添加额外的输出端口，这次我学会了直接在仿真工具里添加信号进行观察，更加方便快捷。

5 思考题

1. **怎样判断 CPU 处在 Reset 异常状态？**

   在 Cortex-M 处理器中，Reset 被视为一种异常，其异常号为 1，对应的异常入口为 `Reset_Handler`。CPU 处在 Reset 异常状态时，实际上就是在执行 `Reset_Handler` 对应的指令序列。

   (1) 从异常号寄存器角度 Cortex-M 内核提供了异常程序状态寄存器 IPSR（Interrupt Program Status Register）。当 CPU 正在执行某个异常服务程序（Exception Handler）时，IPSR 中会保存当前异常的异常号。若

   $$\texttt{IPSR} = 1,$$

   则表示当前处于 Reset 异常服务程序中，即 CPU 正在执行 `Reset_Handler`，此时可以认为 CPU 处在 Reset 异常状态。

   (2) 从 PC 值和向量表角度（结合本实验） Cortex-M 的向量表（Vector Table）通常放在地址 `0x00000000` 起始处：

   - `0x00000000` 处存放主堆栈指针 MSP 初值；
   - `0x00000004` 处存放 Reset 异常入口地址，即 `Reset_Handler` 的起始地址。

   在本实验的仿真中，复位信号 `RSTn` 由 0 变为 1 后，Cortex-M0 首先通过 AHB-Lite 总线读取地址 `0x00000000` 和 `0x00000004`，从 RAMCODE 中取得堆栈初值和 Reset 向量，然后将 PC（Program Counter，程序计数器）装载为 Reset 向量中给出的入口地址。此后的一段时间内，PC 会在 `Reset_Handler` 所在的代码区间内顺序执行。

   因此，在实验波形中若观察到：

- 复位释放后，CPU 首先访问 `0x00000000` 和 `0x00000004` 等向量表地址；
- 随后 `vis_pc_o` 的值被设置为向量表中给出的 Reset 入口地址，并在这一区间内顺序执行；

即可判断此时 CPU 正在执行 Reset 异常服务程序，也就是处于 Reset 异常状态。

2. **如何判断总线操作开始的表示是什么（哪几个信号的值决定了总线操作状态，分别是什么）?**

本实验使用的是 AHB-Lite 片上总线协议（AHB-Lite bus protocol）。在该协议中，一次总线传输（Bus transfer）分为地址阶段（Address phase）和数据阶段（Data phase），并且采用流水线方式工作。当出现**一次新的、有效的总线传输开始**时，需要同时满足以下条件：

(1) 由 HTRANS 表示传输类型 总线传输类型由 `HTRANS[1:0]` 信号决定：

- `HTRANS = 2'b00`：IDLE，无有效传输；
- `HTRANS = 2'b01`：BUSY，占用总线但不发起新传输；
- `HTRANS = 2'b10`：NONSEQ，新的非顺序传输，本实验中使用的主要类型；
- `HTRANS = 2'b11`：SEQ，顺序传输（Burst 传输中的后续传输）。

AHB-Lite 规定：**只有当 `HTRANS[1] = 1` 时，才表示本周期存在有效传输**，即 `HTRANS` 为 NONSEQ 或 SEQ。本实验中仅使用 NONSEQ，因此可以认为：

$$\texttt{HTRANS} = 2'b10 \quad \Rightarrow \quad \text{本周期为一次新的有效传输的地址阶段。}$$

(2) 由 HREADY 表示上一笔传输是否完成 `HREADY` 为总线就绪信号（Ready signal），由所有从机的 `HREADYOUT` 综合得到，用于指示当前数据阶段是否完成：

- `HREADY = 1`：当前传输在本周期结束，可进入下一次传输；
- `HREADY = 0`：当前传输尚未完成，需要插入等待周期，地址和控制信号保持不变。

因此，只有当 `HREADY = 1` 时，主机在本周期输出的地址和控制信息才能被视作“新传输”的地址阶段。

(3) 对某个具体从机,还需片选信号 HSELx 有效　总线扩展模块中的地址译码器（Address Decoder）根据 HADDR 产生各从机的片选信号 HSELx，表示当前地址是否落在该从机的地址空间。例如 RAMCODE 的地址范围为 0x00000000--0x0000FFFF，译码条件为：

$$\texttt{HADDR[31:16]} = 16'h0000 \quad \Rightarrow \quad \texttt{HSEL\_RAMCODE} = 1.$$

因此，一个具体从机的**一次有效总线访问**需满足：

$$\text{有效访问某从机} \iff (\texttt{HSELx} = 1)\&(\texttt{HTRANS[1]} = 1)\&(\texttt{HREADY} = 1).$$

3. **总线读操作时外设如果需要多个周期（例如 3 个周期）完成数据准备，那么 Slave_Interface 中的哪个信号需要重新描述以及如何实现（简述时序）?**

   如果外设需要多个周期才能完成数据准备，那么在 AHB-Lite 总线的 Slave Interface 中，HREADYOUT（或由其组合得到的总线 HREADY）信号需要被重新描述，以实现等待状态（Wait State）的插入。

   实现方式（时序简述）如下：

   - 初始状态：当主机（Master）发起读操作，地址和控制信号在总线上有效时，从机（Slave）通过检测 HSEL、HTRANS[1] = 1 以及 HREADY = 1 等条件，识别出这是针对自己的有效读请求。
   - 拉低 HREADYOUT：在数据尚未准备好的阶段，从机需要将 HREADYOUT 信号驱动为低电平（0）。这会告诉主机，从机还没有准备好数据，需要主机延长当前数据阶段、暂时不要结束本次传输。
   - 维持等待：如果外设需要 3 个周期来准备数据，那么从机需要在接下来的 2 个时钟周期内保持 HREADYOUT 为低电平。在这段等待期间：
     - 主机必须保持地址和控制信号不变（即保持原来的 HADDR、HTRANS、HWRITE 等值）；
     - 主机在每个时钟上升沿检查 HREADY 信号（HREADY 信号通常是所有从机 HREADYOUT 信号组合后的结果）；
     - 由于 HREADY = 0，本次数据阶段尚未完成，HRDATA 上的数值不会被主机采样，可视为无效或保持上一次的值。
   - 准备就绪：在第 3 个时钟周期，当从机终于准备好数据并将**最终有效**的数据放置在 HRDATA 总线上时，需要在该周期的时钟上升沿到来之前，将 HREADYOUT 信号驱动为高电平。此时要求 HRDATA 在本周期内保持稳定，以满足主机对数据的建立/保持时间要求。

- 完成传输：主机在检测到 HREADY 为高电平的那个时钟上升沿，会采样 HRDATA 总线上的数据，从而在该时钟沿完成这次读操作的数据阶段。随后，主机可以在下一周期发起新的总线传输。

简单来说，HREADYOUT 信号就是从机用来“暂停”总线传输的机制：当它为低时，当前数据阶段被延长，主机不会采样 HRDATA；当它重新变为高电平时，表示从机已经准备好数据，当前数据阶段会在这一时钟上升沿完成，主机在此时刻读取 HRDATA 的有效数据。