实验八：LC3 系统-UART 介绍

1. 实验目的：

1.1. 掌握 LC3 输入输出机制

1.2. 了解 UART 通信协议

2. 实验内容：

2.1. 学习 LC3 系统的输入输出机制

2.2. 学习 UART 通信协议

2.3. 学习如何通过 UART 协议实现 LC3 中的输入输出

2.4. UART 协议在仿真中如何实现

3. 实验步骤：

3.1. LC3 系统输入输出机制

在 LC3 的数据通路中，有着 4 个特殊的寄存器：KBDR(KeyBoard Data Register)，KBSR(KeyBoard State Register)，DDR(Display Data Register)，DSR(Display State Register)，它们都是 2Bytes 宽的寄存器，其中 KBDR 和 KBSR 负责 LC3 系统的输入，而 DDR 和 DSR 负责 LC3 系统的输出。之所以 4 个寄存器的位宽都是 2Bytes，是由于这 4 个寄存器是以内存映射的方式被 LC3 系统读写的，即将这 4 个寄存器映射到内存中特定的地址上，在使用 load/store 指令访问对应的地址时，实际上是对相应的寄存器做读写，因此 4 个寄存器的位宽需要和内存的读写位宽相同。但实际上有效的数据并没有 2Bytes，以 KBDR 与 KBSR 距离，KBDR 保存从外部输入的数据，每次可以输入 1Byte，保存在 KBDR 寄存器的低 8 位，KBSR 的最高位保存 1 位有效位，当这一位为 1 时，代表有新的数据输入，数据保存在 KBDR 中，否则代表没有数据输入。由此可以看出，KBDR 的高 8 位和 KBSR 的低 15 位都是冗余的，其中的数据不含任何意义。DDR 和 DSR 类似，DDR 的低 8 位保存需要输出的 1Byte 数据，DSR 的最高位表示有没有数据需要输出。

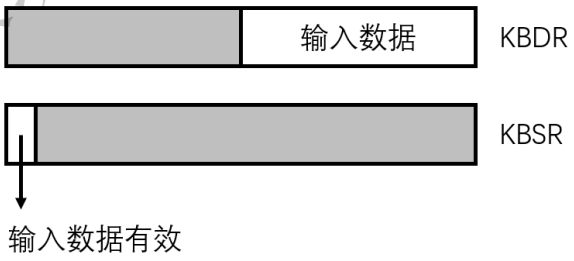

图 8.1 白色表示 KBDR 与 KBSR 的有效数据范围

LC3 需要输入输出时，需要执行相对应的汇编程序代码。

以下是 LC3 完整的输入流程：

程序需要获取一个输入，于是进入一个循环，在循环中不断地检测 KBSR 最高位是否为 1，如果为 1 则结束循环，否则保持循环。当外部的输入数据写入到 KBDR 后，会由输入模块将 KBSR 高位置 1。

循环结束后，使用 load 指令将 KBDR 低 8 位的数据取出，保存到其他寄存器中，或用于计算等操作。

使用 store 指令，向 KBSR 寄存器中写入全零，清除输入有效位。如果不清除 KBSR 的有效位，就无法正常接收下一次的输入。

以下是 LC3 完整的输出流程：

程序需要输出一个数据，于是进入一个循环，在循环中不断地检测 DSR 最高位是否为 1，如果为 1 则结束循环，否则保持循环。当外部准备好从 DDR 接收输出后，会由输出模块将 DSR 高位置 1。

循环结束后，使用 store 指令将需要输出的数据存入 DDR 低 8 位。同时使用 store 指令，将 DSR 的高位置 0。

外部接收完 DDR 中的数据后，就会将 DSR 高位再次置 1，准备好接收下一次输出。

输入输出的标准流程会作为 trap 指令，初始化到内存中，在汇编程序中可以直接调用。

3.2. UART 通信协议简述

在硬件设计中不同于 C 语言，C 中需要输入输出只要使用 scanf 和 printf 函数即可，相关的逻辑都已经包装好了，在硬件设计中输入输出都要自己实现，由于最终我们希望在 FPGA 板上运行 LC3 系统，但是在 FPGA 上使用显示器显示输出和使用键盘进行输入也需要相应的程序来实现，但实际上可以使用更简单的方法来替代，就是使用 FPGA 版上的 UART 接口，通过 UART 协议实现数据的接收和发送，然后在电脑上使用串口调试助手，来实现类似键盘输入和显示器输出的效果。UART(Universal Asynchronous Receiver/Transmitter, 通用异步收发器)是一个在硬件设计中使用十分广泛的协议，UART 的好处在于协议非常简单，FPGA 开发板广泛支持，但缺点在于传输效率较低，传输距离近，且可靠性不够高，但这些对于 LC3 来说已经足够。接下来简单介绍下 UART 通信协议。

在介绍 UART 协议前，首先需要了解串行通信和并行通信的区别。串行通信指数据在单条 1 位宽的传输线上，1bit 接 1bit 地按顺序传送的方式。而在并行通信中 1Byte（8bit）数据是在 8 条并行传输线上同时由源端传到目的地，也可以说有多个数据线（几根就是几 bit）。形象的说，把线路（通道）比作道路，能并排开几辆车的就可以说是“并行”，只能一辆一辆开的就属于“串行”了。

很明显，并行通信的速度要比串行通信的速度快得多，效率更高，费时更少。不过这些都是早期 I/O 速率都不高的情况下的理论理解，随着信息技术的飞速发展，之前的理解放在现在来看已经过时了，现在大部分通信协议都是以串行为主，这是由于在高速状态下，并行口的几根数据线信号之间存在互相干扰，而并行口需要信号同时发送同时接收，任何一根数据线的延迟都会引起问题。而串行只有一根数据线，不存在信号线之间的干扰，而且串行还可以采用低压差分信号，可以大大提高它的抗干扰性，所以可以实现更高的传输速率，尽管并行可以一次传多个数据位，但是由于干扰问题，时钟远远低于串行，所以目前串行传输是高速传输的首选。

UART 作为异步串行通信协议的一种，工作原理是将传输数据的每个二进制位一位接一位地传输。在 UART 通信协议中信号线上的状态为高电平时代表‘1’，信号线上的状态为低电平时代表‘0’。比如使用 UART 通信协议进行一个字节数据的传输时就是在信号线上产生八个高低电平的组合。

异步通信以一个 Byte 为传输单位，通信中两个 Byte 间的时间间隔多少是不固定的，然而在同一个 Byte 中的两个相邻位间的时间间隔是固定的。通俗说是两个 UART 设备之间通信的时候不需要时钟线，这时候就需要在两个 UART 设备上指定相同的传输速率，以及空闲位、起始位、校验位、结束位，也就是遵循相同的协议。

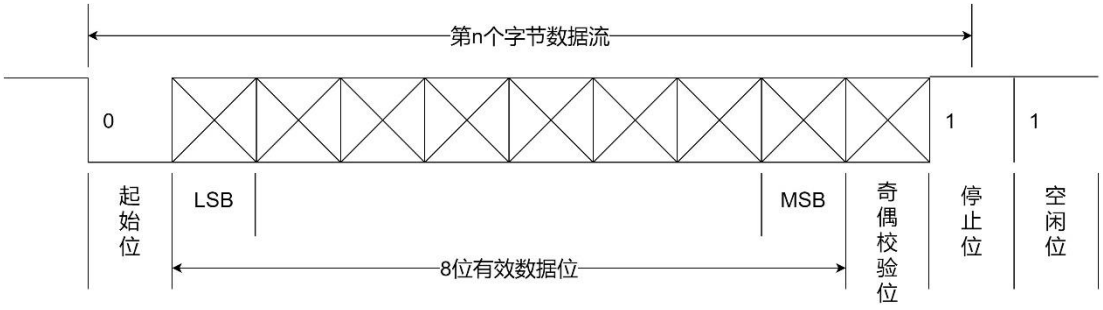

图 8.2 UART 数据传输格式

UART 的数据传输格式如图 8.2 所示。其中各位的意义如下：

空闲位：

UART 协议规定，当总线处于空闲状态时信号线的状态为‘1’即高电平，表示当前线路上没有数据传输。

起始位：

每开始一次通信时发送方先发出一个逻辑‘0’的信号（低电平），表示传输字符的开始。因为总线空闲时为高电平所以开始一次通信时先发送一个明显区别于空闲状态的信号即低电平。

数据位：

起始位之后就是我们所要传输的数据，数据位可以是 5、6、7、8，9 位等，构成一个字符（一般都是 8 位）。如 ASCII 码（7 位），扩展 BCD 码（8 位）。先发送最低位，最后发送最高位，使用低电平表示‘0’高电平表示‘1’完成数据位的传输。

奇偶校验位：

数据位加上这一位后，使得“1”的位数应为偶数(偶校验)或奇数(奇校验)，以此来校验数据传送的正确性。校验位其实是调整个数，串口校验分几种方式：

1、无校验（no parity）。

2、奇校验（odd parity）：如果数据位中“1”的数目是偶数，则校验位为“1”，如果“1”的数目是奇数，校验位为“0”。

3、偶校验（even parity）：如果数据为中“1”的数目是偶数，则校验位为“0”，如果为奇数，校验位为“1”。

4、mark parity：校验位始终为 1（不常用）。

5、parity：校验位始终为 0（不常用）。

LC3 由于对数据可靠性要求不高，选择的是无校验位。

停止位：

它是一个字符数据的结束标志。可以是 1 位、1.5 位、2 位的高电平。 由于数据是在传输线上定时的，并且每一个设备有其自己的时钟，很可能在通信中两台设备之间出现了小小的不同步。因此停止位不仅仅是表示传输的结束，

并且提供计算机校正时钟的机会。停止位个数越多，数据传输越稳定，但是数据传输速度也越慢。

UART 的数据传送速率用波特率(Baud)来表示，单位 bps（bits per second），即每秒钟传送的二进制位数。例如数据传送速率为 120 字符/秒，而每一个字符为 10 位（1 个起始位，7 个数据位，1 个校验位，1 个结束位），则其传送的波特率为 10×120＝1200 字符/秒＝1200 波特。

常见的波特率有 9600bps、115200bps 等等，LC3 中使用的是 115200。其他标准的波特率是 1200，2400，4800，19200，38400，57600。举个例子，如果串口波特率设置为 9600bps，那么传输一个比特需要的时间是 1/9600≈104.2us。

3.3. UART 实现以及与 LC3 输入输出机制的连接

LC3 中的 UART 传送和接收模块参考了开源的设计。在 src/main/scala/LC3/Uart.scala 文件中可以看到具体的 Uart 模块设计，但 UART 具体的实现不属于 LC3 系统范畴，这里不过多介绍。重点在于 UART 模块是如何与 LC3 相连接，以及如何在仿真中实现的。

在 LC3 系统的顶层中，有着两个 UART 的模块，分别是 UartRX 模块和 BufferedUartTX 模块，UartRX 模块负责从外部接收输入数据给 LC3 系统，BufferedUartTX 模块负责将 LC3 系统内部的输出传给电脑来显示。这里的输出模块之所以叫做 BufferedUartTX，是因为它内部实现了一个可以存储 1Byte 数据的缓冲 (Buffer)，LC3 输出的数据可以先放在这个缓冲中，这样在送入缓冲之后 LC3 可以继续执行之后的指令，而不用等待数据传输完毕。而接收模块不使用缓冲是因为通常处理器的运行速度是远大于 UART 的传输速度的，因此当 UART 有输入时，可以认为 LC3 处理器已经准备好接收了，所以不需要缓冲。

在 Top 中，这两个模块的信号串行传输/接收信号线分别接在整个 Top 模块的接口上，连接到外部。而它们传输/接收完整 1Byte 数据的接口连接在 DataPath 模块的相对应接口上。而在 DataPath 模块中，UART 的信号分别被映射到 KBDR(KeyBoard Data Register)，KBSR(KeyBoard State Register)，DDR(Display Data Register)，DSR(Display State Register) 4 个寄存器中。在读写内存时，会检测是否是这四个寄存器映射的内存地址，如果时，则直接访问这 4 个寄存器，如此一来就可以将外界的输入输出与 4 个寄存器相连接，实现 LC3 约定的输入输出设计。

在 DataPath.scala 文件中，IOMap 模块负责完成特殊寄存器与内存之间的映射关系，IOMap 接口接收访存的地址，即 MAR 中的数据，以及访存的使能信号 mio_en，还有判断是读还是写的读写信号 r_w，然后输出 7 个 1bit 的布尔信号，代表对 4 个特殊寄存器以及内存的读写使能。IOMap 中的内容比较简单，作为练习由学生完成，下面给出 IOMap 的工作逻辑。

首先，DDR 是 LC3 需要输出的数据，因此没有读的需求，同样的，KBDR 是从外部输入的数据，因此没有写的需求。当 LC3 访存使能 mio_en 为 true，并且是 r_w 为 false 时，代表需要从内存中读数据，此时判断访存的地址 mar 是否是特殊寄存器，如果是特殊寄存器对应的内存映射地址的话，则将对应的 r_寄存器名接口信号设为 true，否则则为 false。如果 mar 中的地址与所有的特殊寄存器的内存映射地址都不相同时，代表访问的是普通内存，此时还需要判断 mar

的访存地址是否小于 0xfe00，保证不会访问到超过内存地址范围的数据。

类似的，当 LC3 访存使能 mio_en 为 true，并且是 r_w 为 true 时，代表需要往内存中写数据，此时判断访存的地址 mar 是否是特殊寄存器，如果是特殊寄存器对应的内存映射地址的话，则将对应的 w_寄存器名接口信号设为 true，否则则为 false。

特殊寄存器在内存中映射的地址如表 8.1 所示，这些映射地址是在《计算机系统概论》中指定的，由于 LC3 设计中，内存地址范围是 0x0000-0xfdff，因此将特殊寄存器的内存映射地址按顺序放在最大内存地址之后。

表 8.1 特殊寄存器内存映射地址

寄存器名	内存映射地址
KBSR	0xfe00
KDBR	0xfe02
DSR	0xfe04
DDR	0xfe06

在 src/test/scala/IOMaptest.scala 中，有针对 IOMap 模块的单元测试，其中设计了多个测试用例，通过在 chisel_lc3 项目目录下运行以下命令即可执行 IOMap 的单元测试。

```
mill chisel_lc3.test.testOnly LC3.IOMaptest
```

实验八-任务一：根据实验指导，完成 IOMap 模块的代码，并且通过单元测试

3.4. UART 的仿真实现

使用 UART 协议，能够通过电脑与 FPGA 连接，通过串口调试助手实现 LC3 的输入输出，但是在仿真中，没有办法直接调用串口调试助手，并且我们希望像普通的终端程序一样，直接将自己想要输入的数据通过终端输入，而不是像 UART 传输协议一样每次 1bit 传送，这就需要我们再实现一套仿真的 UART 模块，与 LC3 的 UART 模块连接，如图 8.3 所示，先将正常的输入通过仿真 UART 模块转换成 UART 信号，传给 LC3 的 UART 模块，再由 LC3 的 UART 模块将其转换回正常的输入传给 LC3，输出类似。

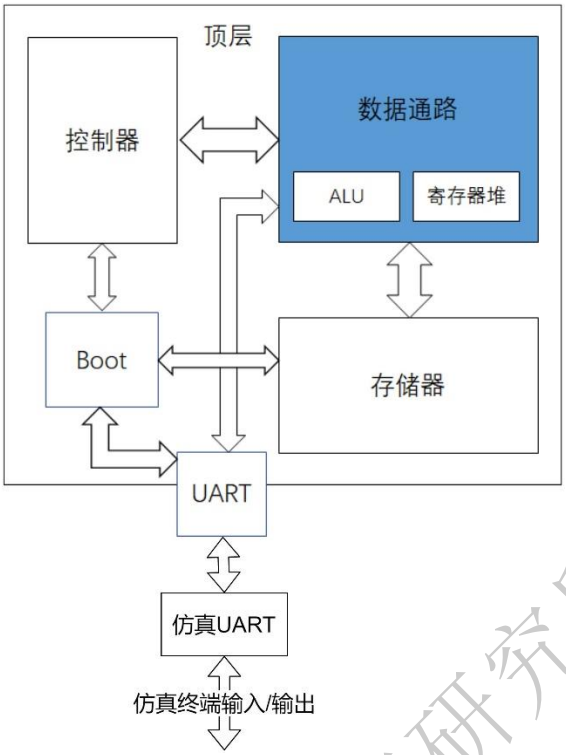

图 8.3 仿真 UART 模块与真实 UART 模块的连接

UART 仿真使用 DPI-C 实现，相关的代码在 src/test/csrc/uart.cpp 中，其核心代码其实就是将 UART 的传输协议使用 C++语言实现了一遍，基本逻辑类似。值得一提的是仿真 UART 模块如何接收终端输入的。在 C++中，输入可以使用 cin 函数，也可以使用类似于 C 中的 scanf 函数，但是这类函数是阻塞输入的，即当程序运行到 scanf 函数时，会将程序暂停下来，直到 scanf 函数返回一个输入的值之后，程序才会再次运行。如果使用 scanf 函数来实现仿真 UART 的输入，由于 LC3 系统会一直监听是否有数据输入，整个系统仿真会被 scanf 函数卡住，无法正常运行。因此需要一个类似于 LC3 系统输入机制的实现，即每隔一段时间就检测终端中有没有输入，如果有就将其接收，否则继续运行仿真。uart.cpp 中实现了一个 get_str 函数，其中使用到了一些 C++的高级特性，稍作了解即可。

3.5. 总结

本节实验重点在于学习 LC3 中的输入输出机制，即 KBDR，KBSR，DDR，DSR 4 个寄存器的含义，以及它们是如何映射到内存中的。其次了解 UART 协议是什么，开发时为什么要选用该协议，以及 LC3 系统中 UART 模块是如何与 LC3 的输入输出策略相结合，实现 FPGA 上的输入输出的。协议具体内容，数据是如何传输的，以及在仿真中的实现，稍作了解即可。