**中国科学技术大学计算机学院**

**《数字电路实验》报告**



实验题目：\_\_\_\_综合实验\_\_\_\_

学生姓名：\_\_\_\_\_\_徐奥\_\_\_\_\_\_

学生学号：\_\_\_PB20061343\_\_\_

完成日期：2021年12月19日

计算机实验教学中心制

2020年09月

【实验题目】

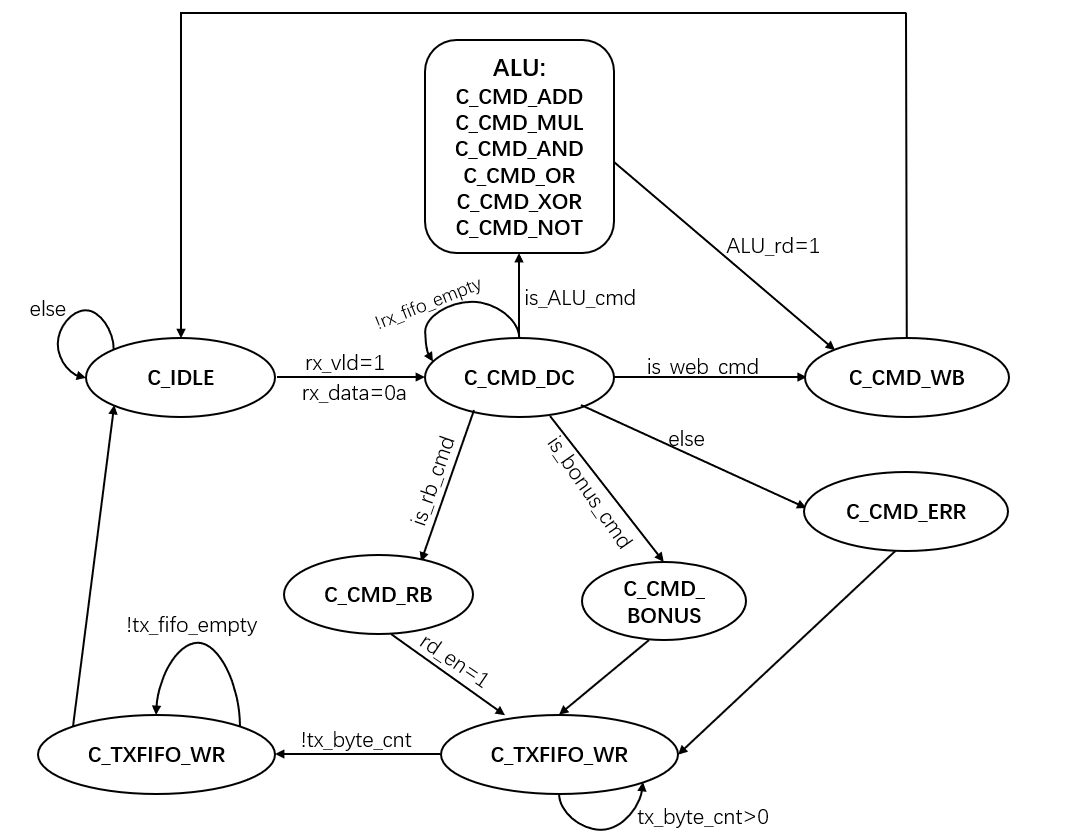
1. 在FPGAOL平台上，利用串口终端等外设，实现简单的Shell功能，例如：在串口协议基础上，实现一个读写命令解析功能，功能电路接收以ASCII码格式发来的命令，并根据命令类型做出合适的响应。
2. 在Logisim中或者在FPGA开发板上实现逻辑电路，通过LED点阵实现汉字的循环显示。要求至少循环显示十个汉字，汉字内容及机内码的形式保存在ROM中，控制电路顺序读取数据，完成机内码到区位码的转换，通过查询字库，获取16\*16的像素数据，最终显示在LED点阵上。

【实验练习】

**一、在FPGAOL平台上实现Shell功能**

**1. 建立状态转换图**

本程序会在以下状态中循环。初始状态为C\_IDLE，当读取到来自Shell的指令时，进入C\_CMD\_DC状态进行指令解码，根据不同指令进入对应的状态，执行完毕后回到初始状态。状态转换图如下：

**图1**

**2. 命令格式说明**

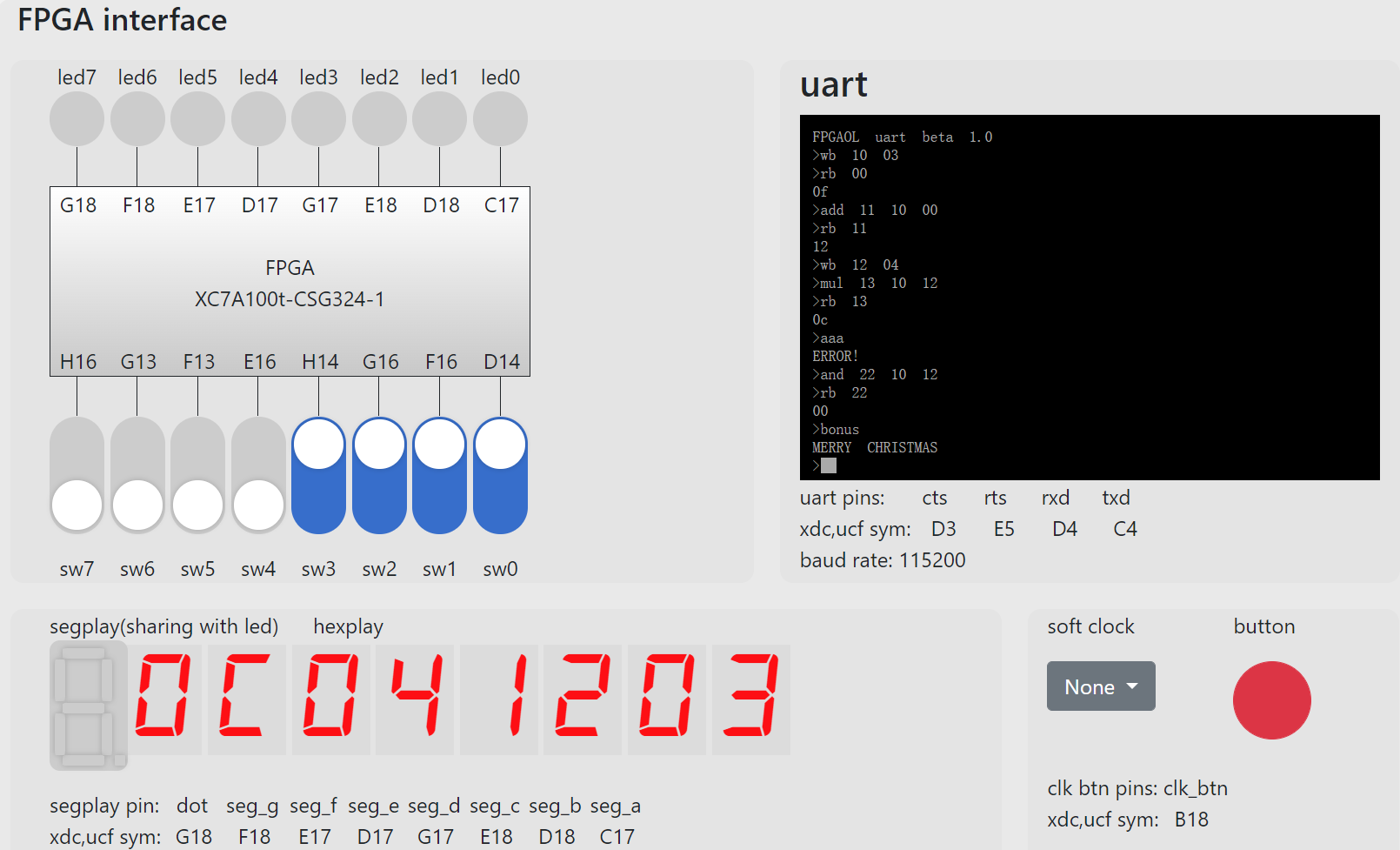
|  |  |  |  |
| --- | --- | --- | --- |
| 命令功能 | 格式 | 举例 | 说明 |
| 写字节 | wb [addr] [data] | wb 11 1a | 向11地址写入字节1a |
| 读字节 | rb [addr] | rb 10 | 从10地址读取一个字节，并以ASCII码格式显示在串口终端 |
| 加运算 | add [addr] [addr] [addr] | add 12 11 10 | 将地址11和10的值相加，结果存储在地址12中 |
| 乘运算 | mul [addr] [addr] [addr] | mul 12 11 10 | 将地址11和10的值相乘，结果存储在地址12中 |
| 与运算 | and [addr] [addr] [addr] | and 12 11 10 | 将地址11和10的值相与，结果存储在地址12中 |
| 或运算 | or [addr] [addr] [addr] | or 12 11 10 | 将地址11和10的值相或，结果存储在地址12中 |
| 非运算 | not [addr] [addr] | not 12 11 | 将地址11的值取非，结果存储到地址12中 |
| 异或运算 | xor [addr] [addr] [addr] | xor 12 11 10 | 将地址11和10的值相异或，结果存储在地址12中 |
| 彩蛋 | bonus | bonus | 在串口终端上显示  ”MERRY CHRISTMAS” |
| 其他 |  |  | 无效命令，串口终端打印”ERROR!”字样 |

说明：本程序对命令格式有严格要求，命令、 地址、数据之间有且仅有一个空格。

1. 地址空间分配

|  |  |  |  |
| --- | --- | --- | --- |
| 读地址空间 | | 写地址空间 | |
| 00 | 8个拨动开关所表示的字节数据 | 00 | 8个LED所对应的字节数据 |
| 10 | 数码管1~0位所表示的字节数据 | 10 | 数码管1~0位所表示的字节数据 |
| 11 | 数码管3~2位所表示的字节数据 | 11 | 数码管3~2位所表示的字节数据 |
| 12 | 数码管5~4位所表示的字节数据 | 12 | 数码管5~4位所表示的字节数据 |
| 13 | 数码管7~6位所表示的字节数据 | 13 | 数码管7~6位所表示的字节数据 |
| 20~2f | 内部存储空间 | 20~2f | 内部存储空间 |

说明：20至2f共计16个地址空间，每个对应的内容空间为8bits

1. 实测演示截图
2. 扩展部分
3. 扩展了指令个数，引入了加、乘、与、或、非、异或操作
4. 扩展了内存个数，由原来的5个扩展为21个
5. 命令实现过程

**（1）读入并解析来自Shell的命令：**

通过模块rx将来自串口的数据进行转化，约定数据收发频率为115200，运用分频计数器对串口接收信号进行采样，保存到rx\_data,每读完8bits数据，置读取完成的标志位rx\_vld为1.

rx\_vld作为rx\_fifo写入的使能标志，当一次8bits数据读入后，调用fifo模块将该数据存储到rx\_fifo中。

当来自串口的信号读取到换行符时，说明来自Shell的一行命令输入完毕，此时状态机进入命令解码状态。从rx\_fifo中读取数据并且存入rx\_byte\_buff，更新命令标志变量is\_add\_cmd，is\_wb\_cmd等.

**（2）wb命令：**

若已将rx\_fifo中的数据读取完毕，且is\_wb\_cmd标志变量为1，则进入写命令状态。根据rx\_byte\_buff更新wr\_addr和wr\_data，前者为要写入数据的目标地址，后者为要写入的数据内容。根据wr\_addr，更新对应的地址数据。

**（3）rb指令**

若已将rx\_fifo中的数据读取完毕，且is\_rb\_cmd为1，则进入读命令状态。根据rx\_byte\_buff更新rd\_addr，即要读取数据的地址。然后根据rd\_addr到对应地址读取数据，存储到rd\_data.

之后进入C\_TXFIFO\_WR和C\_TXFIFO\_WAIT状态，将要显示到串口终端的数据写入tx\_fifo，并逐个tx\_fifo转化为tx\_data，再将tx\_data通过调用tx模块转化为输出到串口的数据。每当tx模块转化完其接收到的数据，则置tx\_fifo的读使能有效，即再从tx\_fifo中读取数据送入tx模块。

tx模块，实现将读入的8bits数据按照数据收发频率115200转化为输出数据，输出到串口终端。

**（4）add命令**

若已将rx\_fifo中的数据读取完毕，且is\_add\_cmd标志位为1，那么进入加命令状态。根据rx\_byte\_buff更新ALU\_addr\_0, ALU\_addr\_1, ALU\_addr\_2，第一个为结果存储的目标地址，后两个为进行加法运算的操作数地址。

根据ALU\_addr\_1和ALU\_addr\_2获取对应位置的操作数，存储到ALU\_operand\_1和ALU\_operand\_2，调用8bits加法器，求得加法结果，存入ALU\_result。

用ALU\_addr\_0和ALU\_result分别更新wr\_addr和wr\_data，进入写命令状态，根据wr\_addr，用wr\_data更新目标地址的值。

**（5）mul命令**

若已将rx\_fifo中的数据读取完毕，且is\_mul\_cmd标志位为1，那么进入乘命令状态。根据rx\_byte\_buff更新ALU\_addr\_0, ALU\_addr\_1, ALU\_addr\_2，第一个为结果存储的目标地址，后两个为进行乘法运算的操作数地址。

根据ALU\_addr\_1和ALU\_addr\_2获取对应位置的操作数，存储到ALU\_operand\_1和ALU\_operand\_2，调用8bits乘法器，求得乘法结果，存入ALU\_result。

用ALU\_addr\_0和ALU\_result分别更新wr\_addr和wr\_data，进入写命令状态，根据wr\_addr，用wr\_data更新目标地址的值。

**（6）逻辑运算命令**

本程序实现的逻辑运算包括与、或、非、异或，所有逻辑运算执行过程类似，只是在部分细节上存在微小差异，故归到一类来说明。

若已将rx\_fifo中的数据读取完毕，且根据更新后的逻辑运算命令标志位，判定为逻辑运算命令，则进入对应的逻辑运算命令执行状态。

根据rx\_byte\_buff更新ALU\_addr\_0, ALU\_addr\_1, ALU\_addr\_2，第一个为结果存储的目标地址，后两个为进行逻辑运算的操作数地址。

根据ALU\_addr\_1和ALU\_addr\_2获取对应位置的操作数，存储到ALU\_operand\_1和ALU\_operand\_2，进行逻辑运算，将结果存储到ALU\_result，用ALU\_addr\_0和ALU\_result分别更新wr\_addr和wr\_data，进入写命令状态，根据wr\_addr，用wr\_data更新目标地址的值。

注：非运算只有一个操作数，故非运算在上述执行过程中只会获得一个操作数地址，相应的，只会获得一个有效操作数。

**（7）命令不合法，输出REEOR信息**

若已将rx\_fifo中的数据读取完毕，且更新后的各命令标志位均为零，即读取到的命令非法，则进入错误命令处理状态。

置tx\_byte\_buff为 ”ERROR!”, 进入C\_TXFIFO\_WR和C\_TXFIFO\_WAIT状态，将要tx\_byte\_buff写入tx\_fifo，并逐个tx\_fifo转化为tx\_data，再将tx\_data通过调用tx模块转化为输出到串口的数据。

**（8）将部分地址对应的数据显示到LED或数码管上**

将对应的数据轮换存储到hexplay\_data，利用刷新，显示到FPGA上。

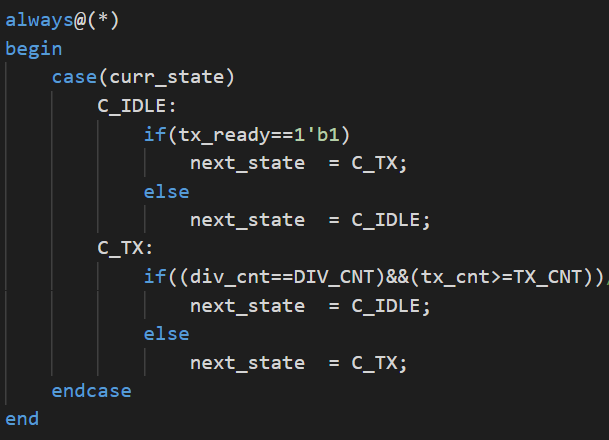
1. 关键代码演示说明

**（1）tx模块**

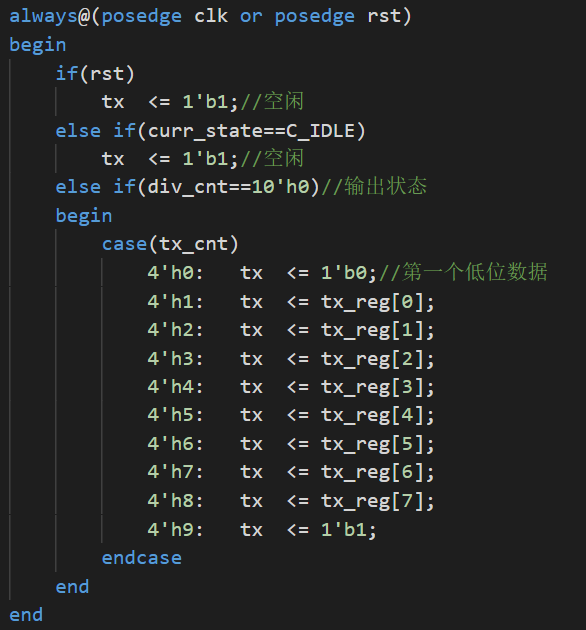
此模块要实现的功能为将8bits input数据转化为串口数据，即一个信号会持续868个时钟周期，再加上开头一位起始低位，结尾一位停止高位。

建立状态机：共包含两个状态，C\_IDLE和C\_TX，前者为空闲状态，后者为输出状态，即将tx\_data的每一位转化为868时钟周期的输出信号tx。当tx\_fifo非空时，状态机进入输出状态。当数据输出数目达到10\*868后，说明当前8bits数目输出完毕，进入空闲状态。

div\_cnt记录当前bit的数据已经输出了多少位，tx\_cnt记录当前输出是第几位数据。更新次态的代码为：



转化输出信号tx的代码为：



（2）rx模块

本模块实现了串口数据的输入，并将输入的数据转化为8bits的rx\_data。与tx模块实现过程类似，只不过数据传输方向相反。

状态机有两个状态，空闲状态和输入状态。由于数据采样是在868bits信号的中间，所以当第一个868时钟周期的低位开始信号输入一半时，状态由空闲装填转化为输入状态。更新次态的代码如下：

*//如无特殊说明，则应完成对应实验手册上的所有练习题目，将过程和结果以图文并茂的形式体现在本报告中，建议实验过程中随手保存各种截图。*

【总结与思考】

*//请填写对于本次实验的总结与思考，鼓励填写对于本实验或者本课程的各种建议和吐槽。*