**计算机组织与结构II**

**实验一**

**POC设计**

|  |  |
| --- | --- |
| **学号** | **姓名** |
|  |  |

**2024年3月6日**

**School of Information Science and Engineering**

**Southeast University**

1. **工作步骤**

系统实现了一个基于UART的简单打印功能，由处理器、外设输出控制器(POC)和打印模块三部分组成。整个系统工作在50MHz时钟下，以9600波特率进行UART通信。系统支持两种工作模式：查询模式和中断模式，由SR0寄存器决定（0代表查询模式，1代表中断模式）。

**处理器**

负责系统初始化、数据准备和状态控制；外设输出控制器管理数据缓冲、就绪状态与模式切换；打印模块执行串行数据发送。

查询模式状态转移流程：

1. 初始化状态：系统复位后，处理器设置SR0=0，进入查询模式，准备打印字符串“Hello, World! (polling)”。
2. 查询就绪状态：处理器反复读取POC的SR7寄存器；SR7为1时，处理器准备发送下一个字符；为0时，处理器继续等待。
3. 数据写入状态：检测到SR7=1后，处理器向POC数据寄存器写入字符，并将SR7置为0，通知POC数据已准备好；完成后返回查询就绪状态。

所有字符发送完毕后，处理器切换到中断模式。

中断模式状态转移流程：

1. 中断模式启动状态：处理器将SR0设置为1，进入中断模式；POC检测到SR0变为1，进入中断请求状态。
2. 中断等待状态：处理器等待来自POC的中断请求（irq信号）；irq被拉低时，进入中断响应阶段。
3. 中断响应状态：检测到中断信号后，处理器准备字符数据并写入POC；数据传输完成后，POC取消中断请求（irq恢复为高电平）。

上述过程重复，直到发送完“Hello, World! (interrupt)”后，处理器进入完成状态，所有工作完成。

**POC**

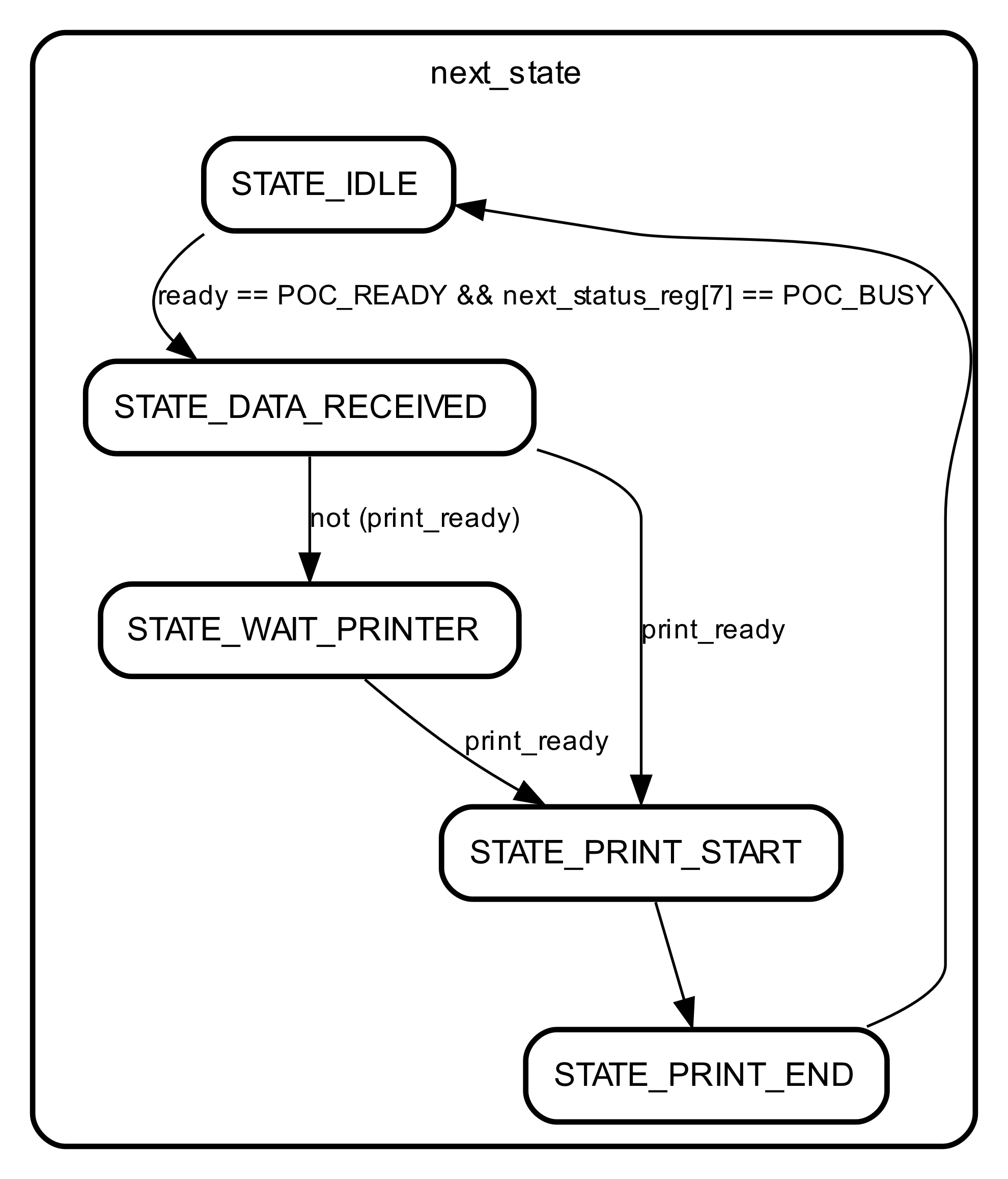
1. STATE\_IDLE（POC空闲状态）： POC等待处理器发送数据；  
   如果是中断模式，且POC空闲(SR7=1)、打印模块准备好(print\_ready=1)，则主动发出中断请求(irq=0)；  
   若检测到SR7寄存器从1变为0，表示处理器已发送数据，POC进入STATE\_DATA\_RECEIVED。
2. STATE\_DATA\_RECEIVED（数据接收状态）： POC接收到处理器发来的字符数据，等待打印模块就绪；  
   若打印模块准备好(print\_ready=1)，进入打印启动状态；  
   若未准备好，进入等待打印机状态。
3. STATE\_PRINT\_WAIT（等待打印机状态）： POC持续检测print\_ready信号；  
   当print\_ready变为高电平(1)时，转入打印启动状态。
4. STATE\_PRINT\_START（打印启动状态）： POC将数据传输至打印模块；  
   同时发出pulse\_request脉冲通知打印模块接收数据；  
   完成后转回POC空闲状态，更新SR7寄存器为1，表示POC空闲。

**打印模块**

1. STATE\_IDLE（打印模块空闲状态）： 打印模块准备就绪(print\_ready=1)，等待POC请求；  
   收到POC脉冲请求(pulse\_request=1)时，进入数据接收状态。
2. STATE\_RECEIVE\_DATA（数据接收状态）： print\_ready拉低，表示正在接收数据；  
   完成数据接收后立即进入发送数据状态。
3. STATE\_TRANSMIT（UART发送启动状态）： 打印模块启动UART传输；  
   先发送起始位(逻辑0)，随后发送数据位（8位，低位优先），最后发送停止位(逻辑1)。
4. STATE\_WAIT\_COMPLETION（等待UART发送完成状态）： 等待UART发送完毕(tx\_done=1)；  
   UART发送完成后，print\_ready拉高(1)，返回STATE\_IDLE状态，准备接收下一字符。

两种模式比较：查询模式简单易实现但效率较低，占用处理器资源；中断模式效率更高，但需要增加额外的中断处理机制，适合高效率场合使用。

1. **有限状态机（FSM）状态图**



1. **状态机核心代码**

//==========================================

// 计算下一状态和输出

//==========================================

always @(\*) begin

    // 默认：保持当前值

    next\_state = state; next\_status\_reg = status\_reg;

    // 处理CPU寄存器读写操作

    if (rw) case (addr) // CPU写操作

        // ...寄存器写操作...

    endcase

    else // CPU读操作

        // ...寄存器读操作...

    // 主状态机逻辑

    case (state)

        STATE\_IDLE:       // POC空闲状态,在中断模式下生成中断请求，若SR7从1变0则准备打印

        STATE\_DATA\_RECEIVED:  // 数据已接收，检查打印机是否就绪

            // 若就绪，发送脉冲请求；若未就绪，等待打印机

        STATE\_WAIT\_PRINTER:   // 等待打印机就绪

            // 检测print\_ready信号

        STATE\_PRINT\_START:    // 开始打印状态 (脉冲高)

            // 转入打印结束状态

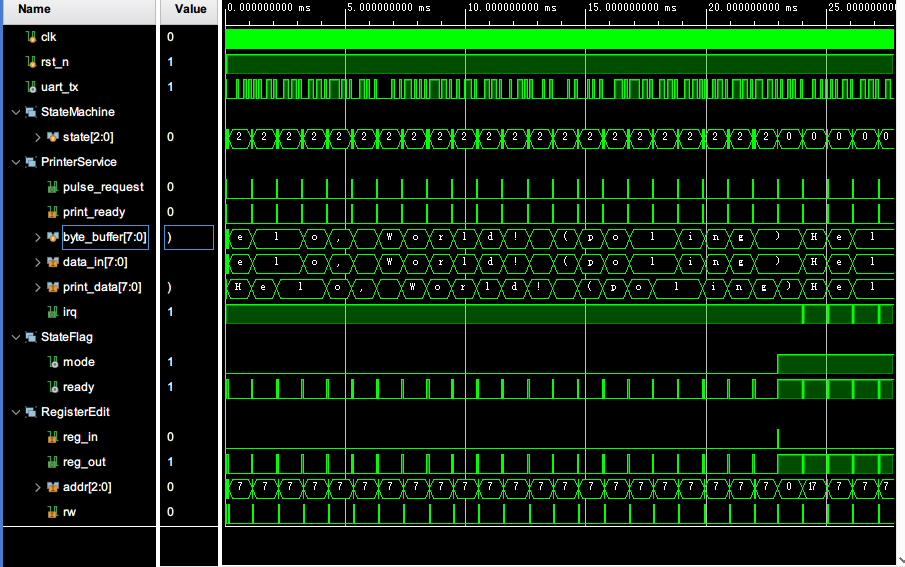
        STATE\_PRINT\_END:      // 结束打印状态 (脉冲低)

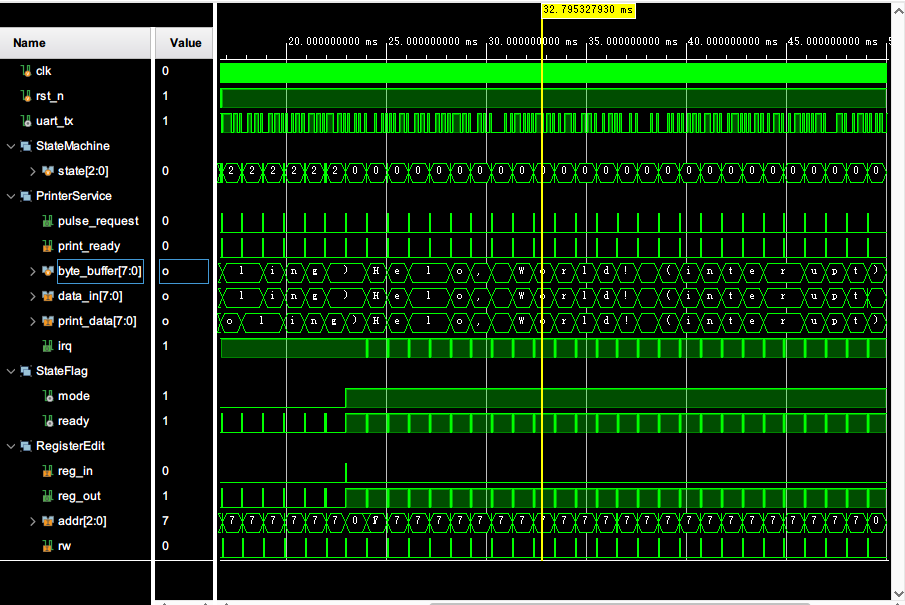
            // 结束脉冲，设置POC为就绪状态，返回空闲状态

        default: next\_state = STATE\_IDLE;

    endcase

end

1. **仿真结果与分析**



从图中可以看到，当mode=0(SR0=0)的时候，poc处于轮询模式打印，irq信号一直维持高电平，当processor通过reg\_out检查到SR7(ready)=1，表明poc可用，便会写入SR7(ready)=0时，byte\_buffer会从data\_in接受数据，并进入打印/等待状态，由于打印速度较慢，可以看出，每次都会等到print\_ready为1后byte\_buffer的数据才会进入print\_data，并且产生一个pulse\_request请求打印，请求后，print\_ready变成0直到打印完成，uart\_tx是print模块输出的串行数据。打印完成后，poc会自动把SR7(ready)重新置为1，告诉processor可以继续请求打印了。State是有限状态机的五个状态，根据情况进行转移。

对于mode=1（SR0=1）的情况也是类似的，首先processor会通过reg\_in写进地址为0的SR0(mode)=1,让模式切换到中断，这个时候，poc自己会在确认print\_ready=1的情况下不断向processor发送低电平有效的中断请求，processor接收到请求后进行轮询类似的步骤，由于printer可用才会发起irq，processor此时无需查询可用性，直接传输数据并把SR7(ready)变为0，接下来的步骤与轮询模式类似，只是打印完成后，ready置为1后，当print\_ready又为1的时候，poc发出中断请求。

1. **问题与解决方案**

波形图上可能无法很直观的表示打印的数据，因此使用了ASCII，让波形更加直观，此外轮询模式和中断模式可能通过波形不够直观，因此，我设计了processor会轮流切换poc的模式，并在不同的工作模式下打印不同的字符串，以此作为区分。

1. **总结**

本系统实现了一个基于POC（外设输出控制器）的简单打印功能，主要包括processor（处理器）、POC和printer（打印模块）三个模块，系统支持两种工作模式：

* **轮询模式**：processor主动检查POC的SR7寄存器状态，确认POC空闲后再发送数据，资源占用较高。
* **中断模式**：POC主动通过中断请求信号（irq）通知processor准备就绪，有效降低processor资源占用。

仿真结果显示：

* 在轮询模式下，processor需主动查询SR7寄存器，当POC空闲时发送数据；POC检测到数据后等待printer就绪，再发出pulse\_request请求打印，完成后自动恢复就绪状态。
* 在中断模式下，POC主动发起中断请求信号通知processor，无需processor主动查询状态。其他步骤与轮询模式类似。

为增强仿真结果的直观性，采用ASCII字符显示打印数据，并设计processor在不同模式下打印不同字符串，以区分两种工作模式。