实验六 综合设计

2024春

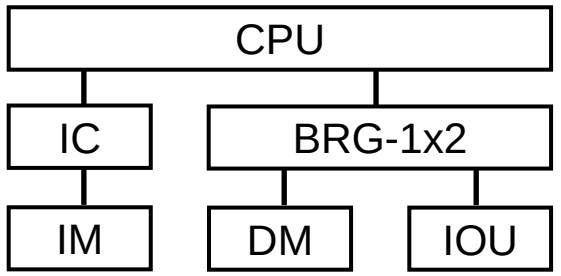
zjx@ustc.edu.cn

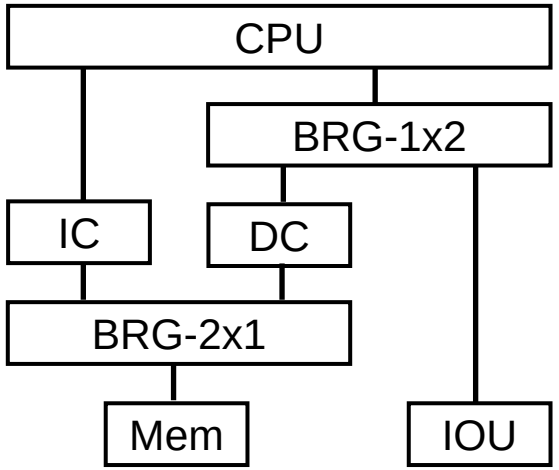
实验目标

- 掌握计算机片上系统 (SOC) 的设计和调试方法
- 掌握计算机输入/输出 (I/O) 的接口技术
- 了解计算机软硬件系统的优化方法

实验内容

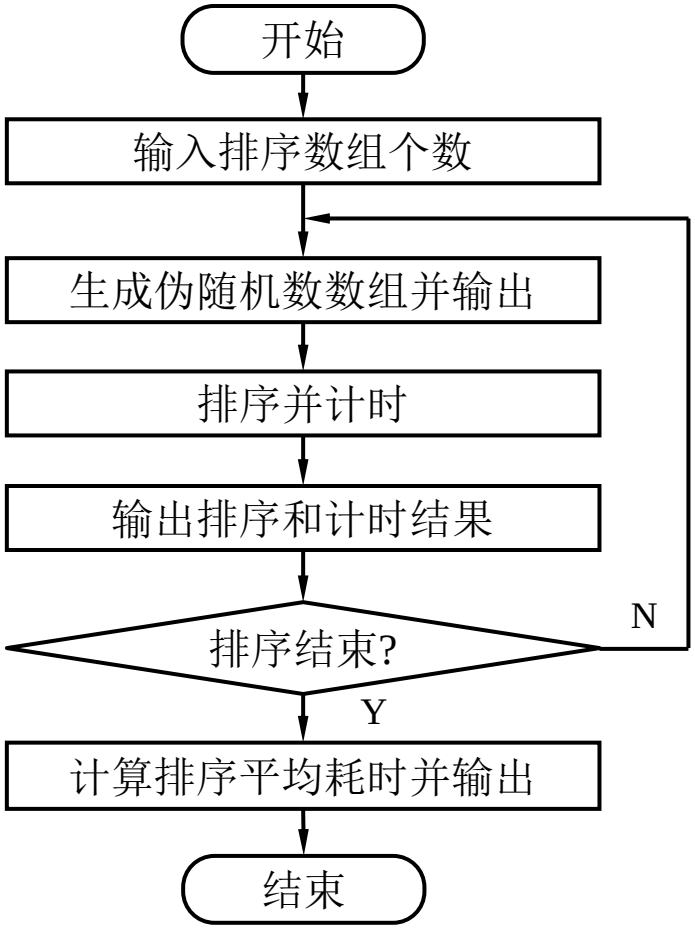
- **设计LA32R CPU的片上系统 (System on Chip, SOC)**
  - IC/DC：指令/数据Cache
  - IM/DM：指令/数据存储器
  - Mem：指令和数据混合的存储器
  - BRG：仲裁/转接桥
  - IOU：Input/Output Unit，输入/输出单元
- **编写应用程序，评估排序耗时**
  - 数组数据：1024 x 32位无符号数
  - 排序算法：自选





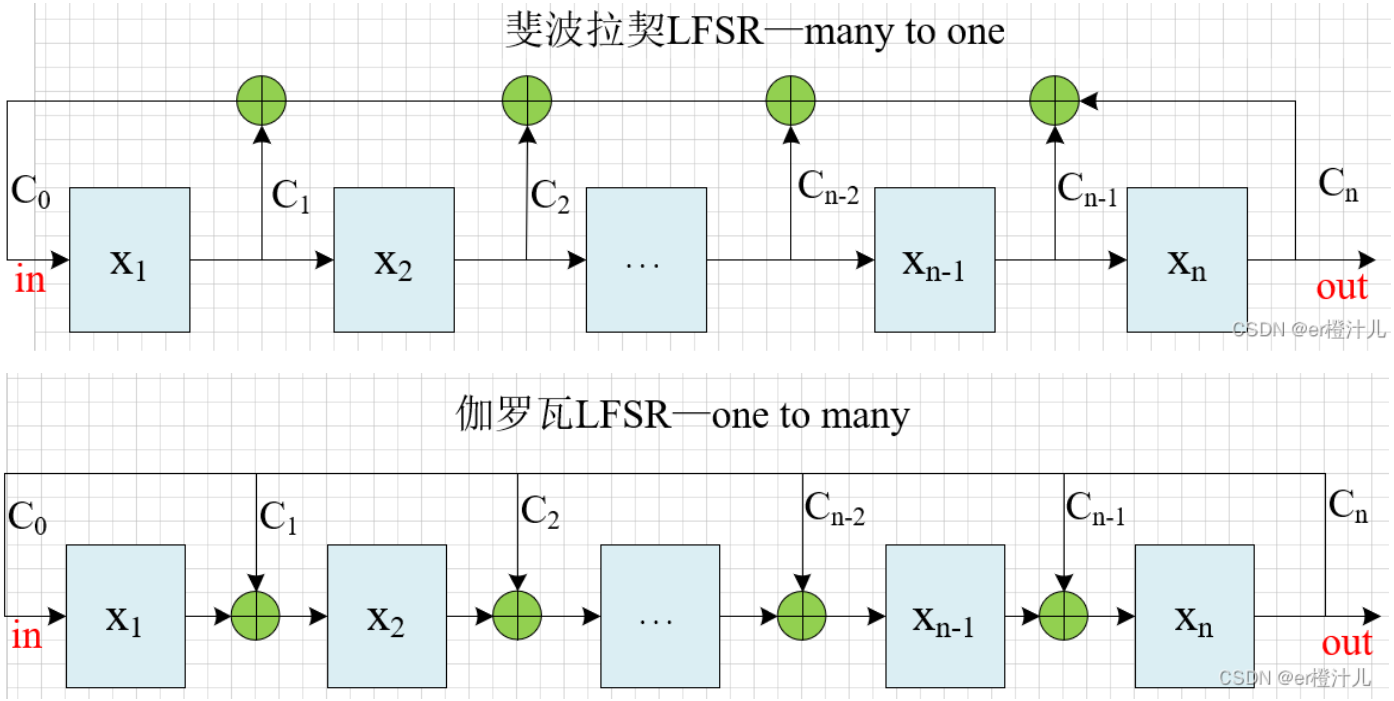
应用程序

- **输入排序数组个数**
  - 数量限定为 2^N, N = 0 ~ F
  - 通过开关或者串口输入N
- **生成伪随机数算法**
  - 伽罗瓦LFSR，从右到左递增编号，初值为 0x1234_5678
- **输出数组和计时结果**
  - 通过按钮控制的LED指示灯和数码管输出
  - 或者通过串口输出，数组数据项间tab分隔，数组间以及数组与计时结果间空行分隔



伪随机数生成算法

- **Linear Feedback Shift Register, LFSR**
  - 由移位寄存器和异或门组成，主要应用于伪噪声序列、伪随机数、计数器、数据加密和CRC校验等



本原多项式

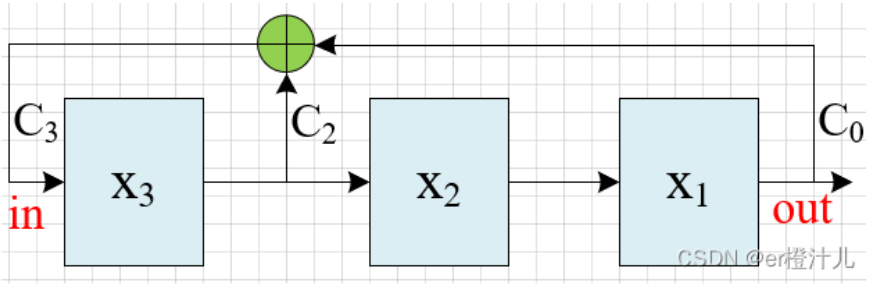
触发器的个数 (n)	反馈多项式
2	x^2+x+1
3	x^3+x^2+1
4	x^4+x^3+1
5	x^5+x^3+1
6	x^6+x^5+1
7	x^7+x^6+1
8	$x^8+x^6+x^5+x^4+1$
9	x^9+x^5+1
10	$x^{10}+x^7+1$
11	$x^{11}+x^9+1$
12	$x^{12}+x^{11}+x^{10}+x^4+1$
13	$x^{13}+x^{12}+x^{11}+x^8+1$
14	$x^{14}+x^{13}+x^{12}+x^2+1$
15	$x^{15}+x^{14}+1$
16	$x^{16}+x^{14}+x^{13}+x^{11}+1$
17	$x^{17}+x^{14}+1$

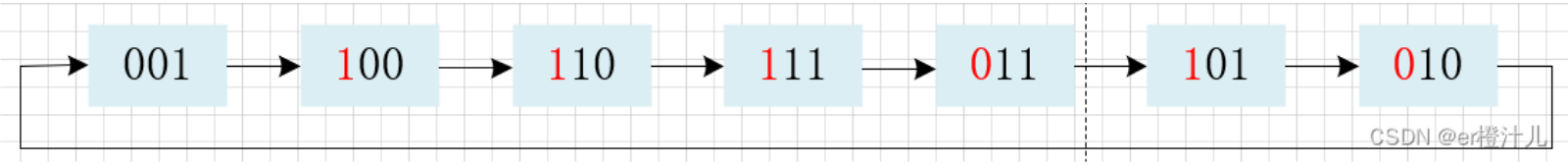
No of Taps	Generator Polynomial	No of Taps	Generator Polynomial
2	[2 1 0]	28	[28 25 0]
3	[3 2 0]	29	[29 27 0]
4	[4 3 0]	30	[30 29 28 7 0]
5	[5 3 0]	31	[31 28 0]
6	[6 5 0]	32	[32 31 30 10 0]
7	[7 6 0]	33	[33 20 0]
8	[8 6 5 4 0]	34	[34 15 14 1 0]
9	[9 5 0]	35	[35 2 0]
10	[10 7 0]	36	[36 11 0]
11	[11 9 0]	37	[37 12 10 2 0]
12	[12 11 8 6 0]	38	[38 6 5 1 0]
13	[13 12 10 9 0]	39	[39 8 0]
14	[14 13 8 4 0]	40	[40 5 4 3 0]
15	[15 14 0]	41	[41 3 0]
16	[16 15 13 4 0]	42	[42 23 22 1 0]
17	[17 14 0]	43	[43 6 4 3 0]
18	[18 11 0]	44	[44 6 5 2 0]
19	[19 18 17 14 0]	45	[45 4 3 1 0]
20	[20 17 0]	46	[46 21 10 1 0]
21	[21 19 0]	47	[47 14 0]
22	[22 21 0]	48	[48 28 27 1 0]
23	[23 18 0]	49	[49 9 0]
24	[24 23 22 17 0]	50	[50 4 3 2 0]
25	[25 22 0]	51	[51 6 3 1 0]
26	[26 25 24 20 0]	52	[52 3 0]
27	[27 26 25 22 0]	53	[53 6 2 1 0]

示例：斐波那契LFSR

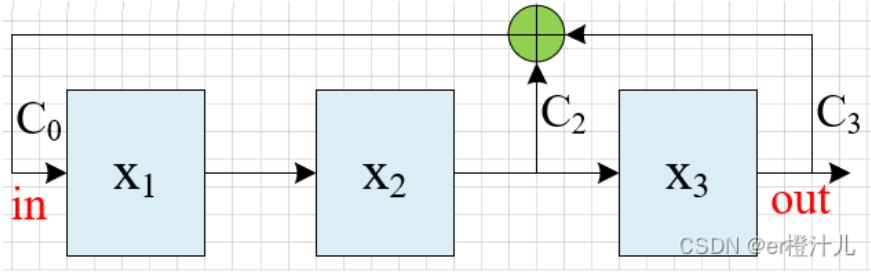
- **n = 3, $g(x) = x^3 + x^2 + 1$**

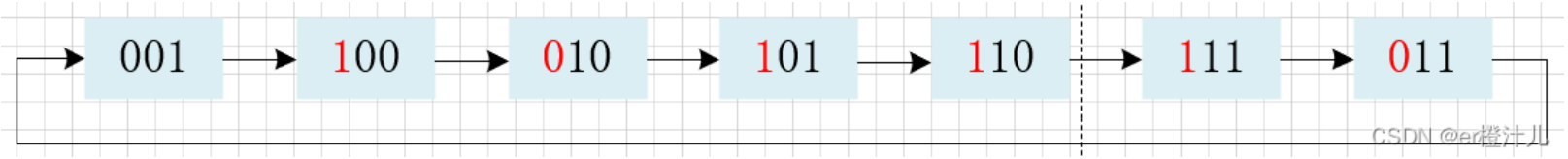
  从右到左依次递增编号





从左到右依次递增编号

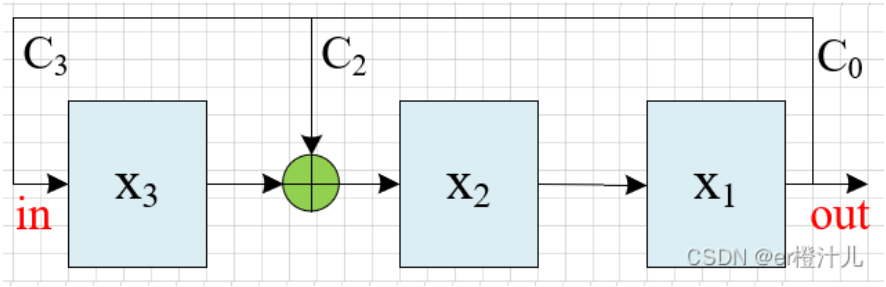


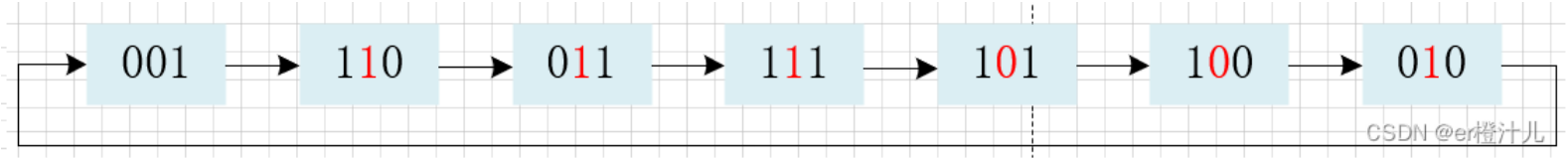


示例：伽罗瓦LFSR

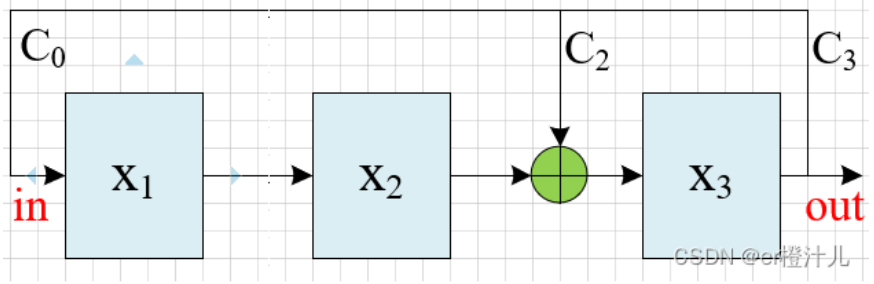
- **n = 3, $g(x) = x^3 + x^2 + 1$**

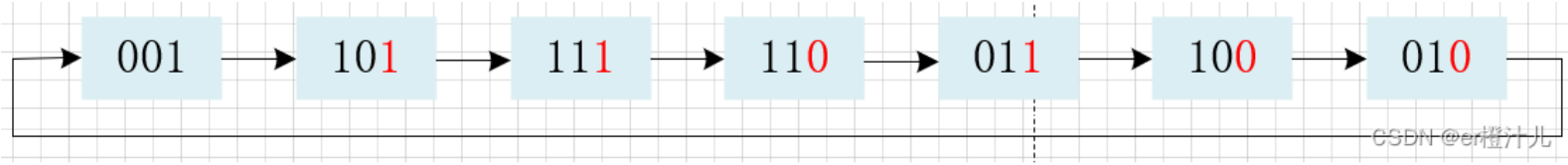
从右到左依次递增编号





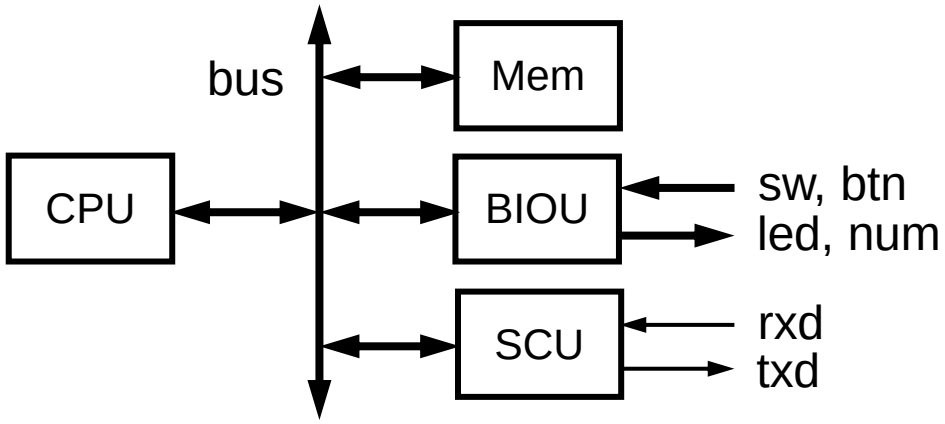
从左到右依次递增编号





输入/输出

- **输入/输出 (Input/Output, I/O) 设备，也称外设**
  - 例如，开关、按钮、指示灯、数码管，计时器、串行通信等
- **I/O设备编址方式：存储器映射I/O (**Memory-Mapped I/O, **MMIO)，统一编址**
- **总线 (bus) 信号**
  - 地址：addr
  - 写数据：wdata
  - 读数据：rdata
  - 写使能：we
  - 使能：en



- BIOU：Basic I/O Unit
- SCU：Serial Communication Unit

- **I/O信息交换方式：程序查询方式**

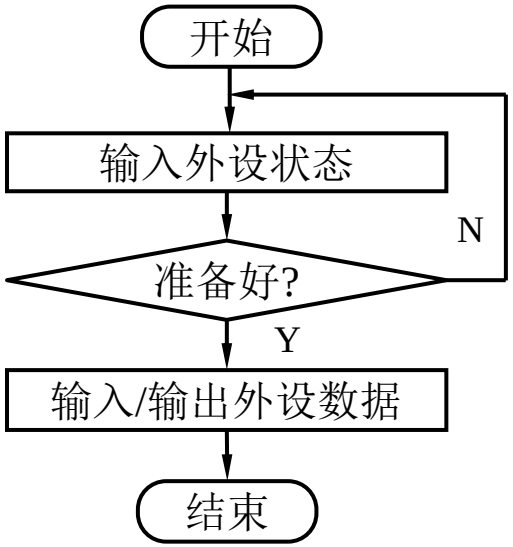
I/O端口地址

```
// mycpu_env\soc_verify\soc_dram\rtl\confreg\confreg.v
// mycpu_env\func\include\cpu_cde.h

#define TIMER_ADDR          0xbfafe000      // 计时器

#define LED_ADDR            0xbfaff020      // 16个LED指示灯
#define LED_RG0_ADDR        0xbfaff030      // 双色LED指示灯0
#define LED_RG1_ADDR        0xbfaff040      // 双色LED指示灯1
#define NUM_ADDR            0xbfaff050      // 8个7段数码管
#define SWITCH_ADDR         0xbfaff060      // 16个开关
#define BTN_STEP_ADDR       0xbfaff080      // 2个按钮

#define UART_ADDR           0xbfafff10      // 串口
```

程序查询方式I/O

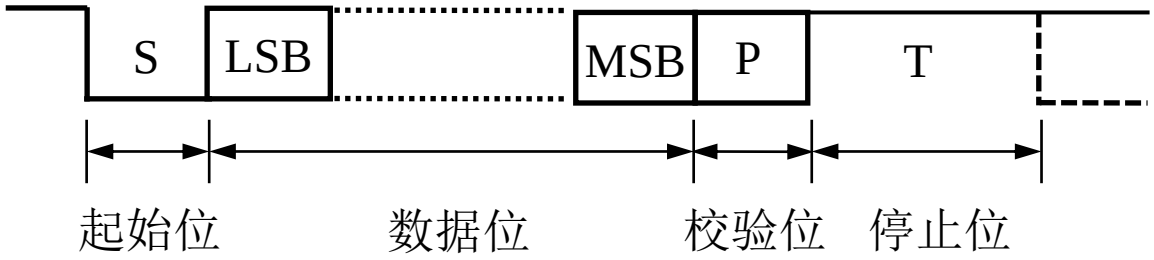
```
# nscscc2023-personal\source\lab2\code\lab2.S
info:
   .asciz "Fib Finish."
WRITESERIAL:
   lu12i.w    s0, -0x40300   # s0 = 0xbfd00000 (io base address)
   la.local   s1, info               # s1 = string pointer
   ld.b       a0, s1, 0x0            # a0 = char
loop1:
   addi.w     s1, s1, 0x1
.TESTW:
   ld.b       t0, s0, 0x3fc          # read tx status (0xbfd0-03fc)
   andi       t0, t0, 0x001
   beq        t0, zero, .TESTW       # check if ready(1)
   st.b       a0, s0, 0x3F8          # write tx data (0xbfd0-03f8)
   ld.b       a0, s1, 0x0
   bne        a0, zero, loop1
```

```
READSERIAL:
.TESTR:
   ld.b       t0, s0, 0x3fc
   andi       t0, t0, 0x002
   beq        t0, zero, .TESTR
   ld.b       a0, s0, 0x3F8
```



RS-232协议

- **数据传输格式**
  - 起始位S：1位，为0
  - 校验位P：1位，可选
  - 数据位D：5~8位，低位在前
  - 停止位T：1、1.5或2位，为1

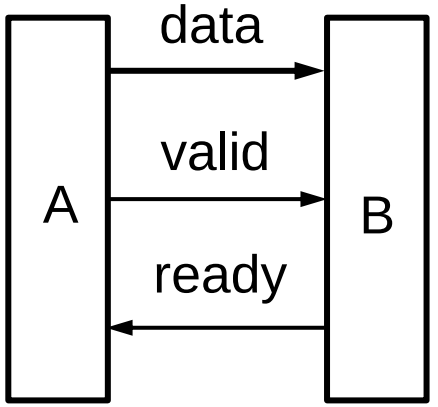


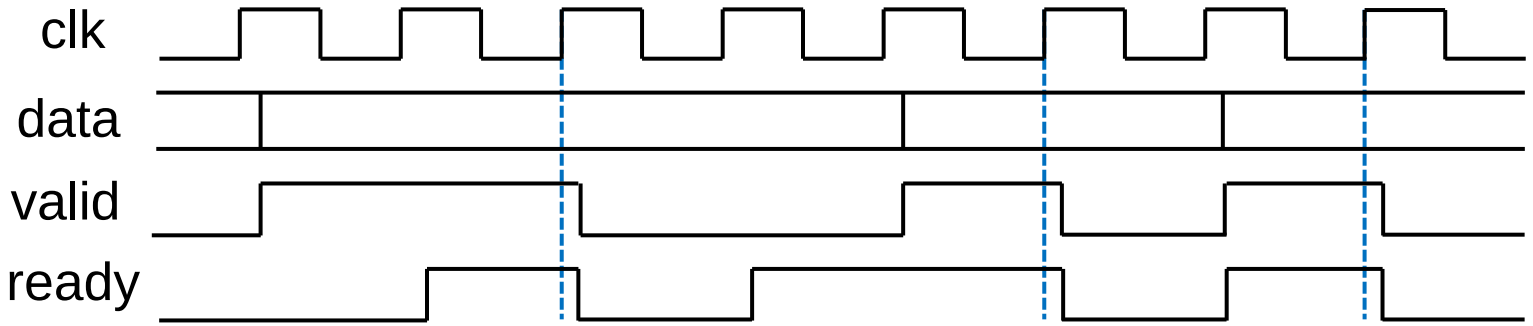
- **数据传输速率**
  - 波特率：每秒传输位数。常用波特率有1200、2400、9600、19.2K、38.4K、56K、115.2K等

实现：8位数据位，无校验位，1位停止位，波特率9600

Valid-Ready握手协议

- 数据源端A：数据准备好，则置valid有效
- 数据目标端B：准备好接收数据，则置ready有效
- 在时钟采样沿valid和ready均有效时，完成数据传输



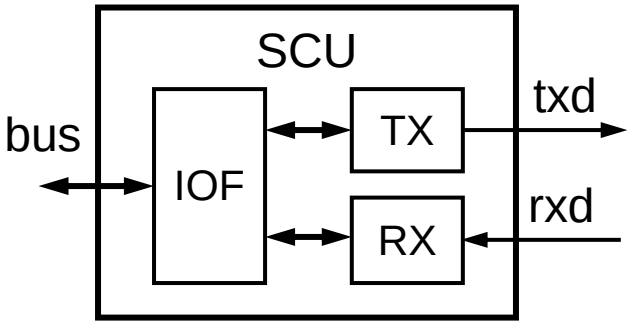


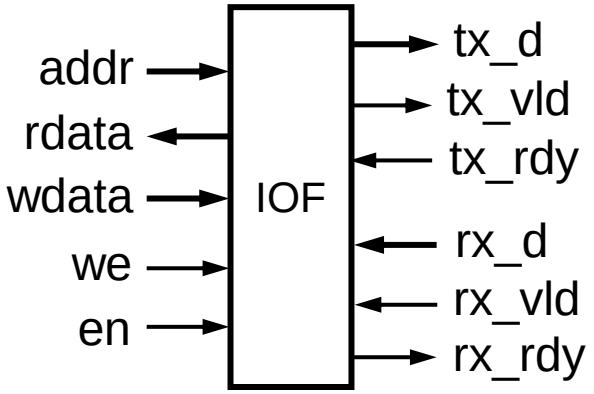
Valid-Ready握手协议 (续)

- 为防止死锁 (Deadlock)，源端不允许在valid置位前等待ready置位，而目标端允许在ready置位前等待valid置位，即目标可以等待源，而源不可以等待目标
- valid置位后必需保持，直至握手完成，即时钟采样沿时valid和ready均置位，而ready置位后可以在valid置位前取消置位
- 建议目标端准备好就置位ready，这样在valid置位后仅需一个时钟周期即可完成数据传输，从而提高效率

串行通信单元

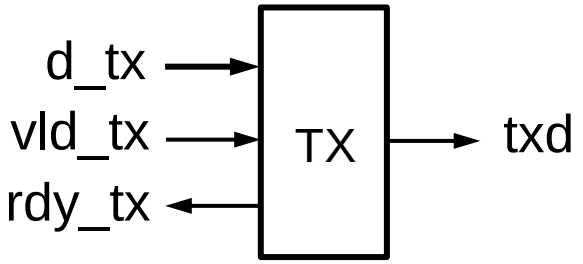
- **SCU：Serial Communication Unit**
- **发送器 (Transmitter, TX)**
  - 并行输入数据→串行输出数据
- **接收器 (Receiver, RX)**
  - 串行输入数据→并行输出数据
- **I/O接口 (I/O Interface, IOF)**
  - CPU侧：存储器接口
  - TX和RX侧：Valid-Ready协议
  - 输出缓冲寄存器：OBR
  - 输入缓冲寄存器：IBR
  - 状态寄存器：SR



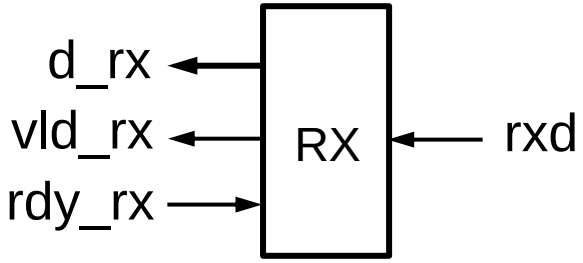


串行通信单元 (续)

- **TX：发送器**
  - d_tx：8位并行发送数据
  - vld_tx：发送有效 (valid)
  - rdy_tx：发送准备好 (ready)
  - txd： 1位串行发送数据

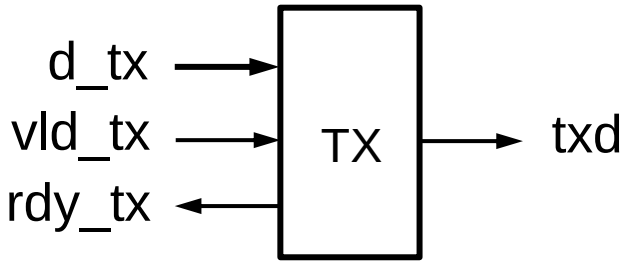


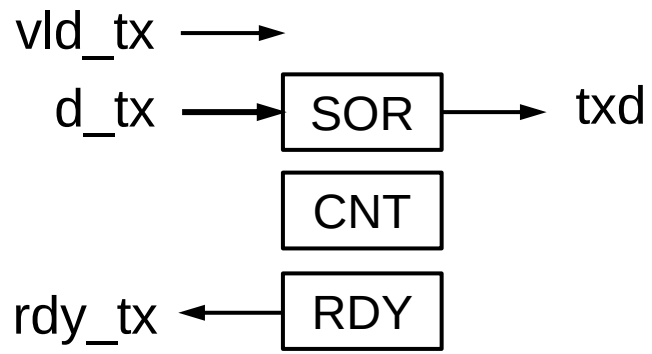
- **RX：接收器**
  - rxd：1位串行接收数据
  - d_rx：8位并行接收数据
  - vld_rx：接收有效
  - rdy_rx：接收准备好

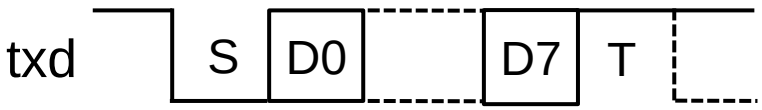


发送器数据通路

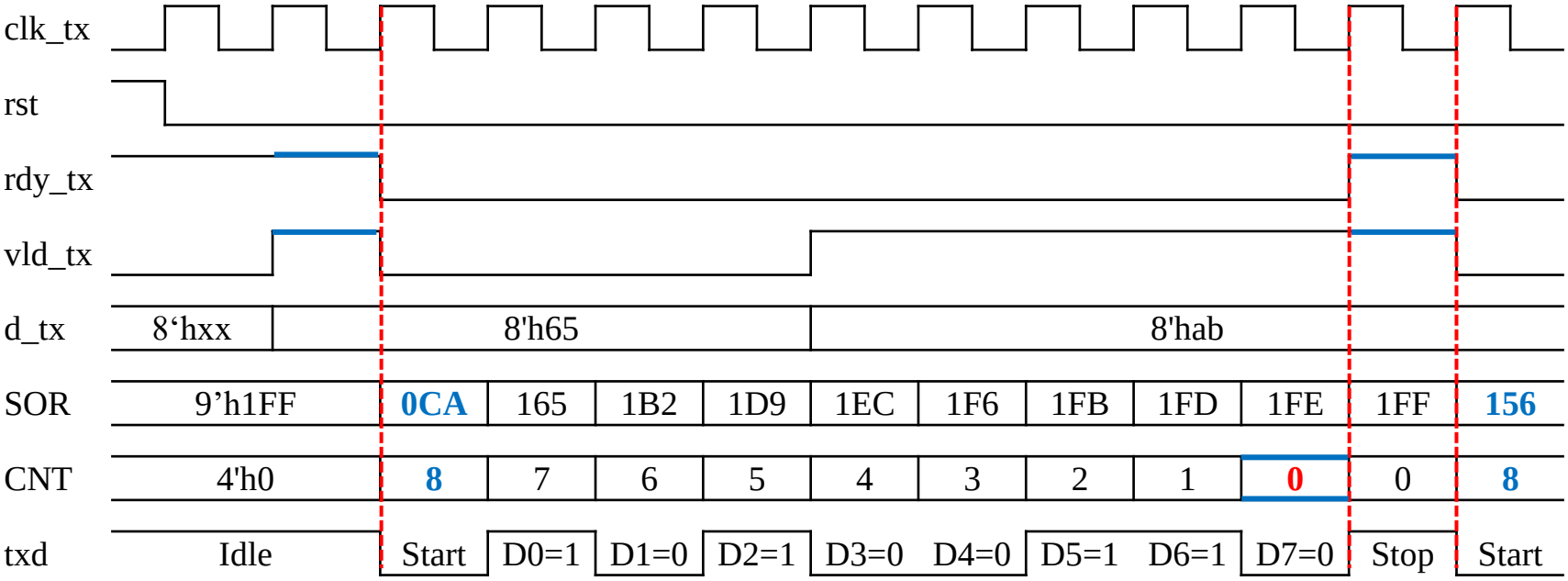
- **SOR：移位输出寄存器，9位**
  - 复位：全部位置1
  - 置数：数据d_tx，起始位0
  - 移位：右移，最高位补1
- **CNT：位计数器，4位**
  - 复位：清零
  - 置数：8
  - 计数：不等于0就递减计数
- **RDY：准备好标志，1位**
  - 复位：置1
  - vld_tx和rdy_tx均为1时清零
  - CNT等于0时置1







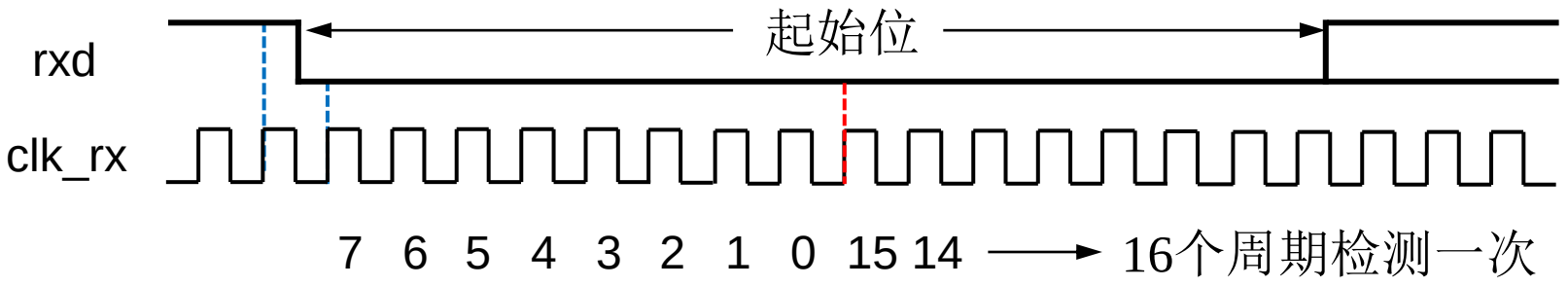
发送时序



- **vld_tx 和 rdy_tx均有效时，发送数据 d_tx 附加起始位0后存入移位输出寄存器SOR，rdy_tx 清零，移位计数器CNT加载常数8；随后SOR右移位和CNT递减计数**
- **CNT等于0时，停止计数，rdy_tx置1**

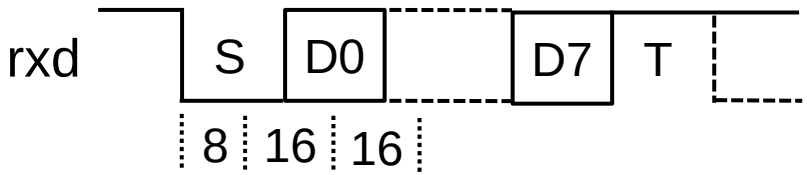
接收过程

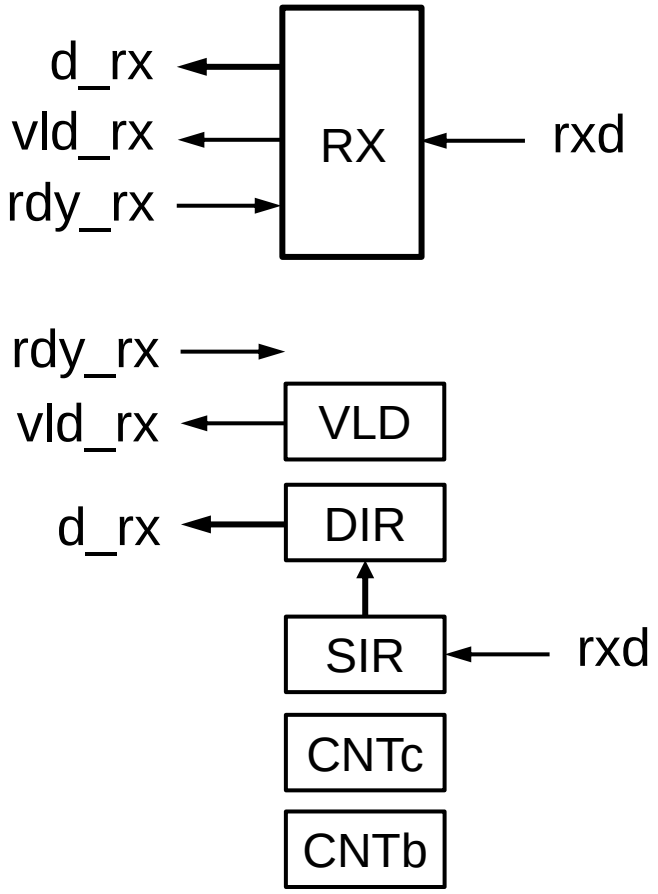
- 当检测到rxd信号由1到0跳变，且8个接收时钟(clk_rx)后检测仍为0，则确认为起始位
- 接着每隔16个接收时钟，对rxd检测一次，依次作为数据位和停止位，存入移位输入寄存器SIR



接收器数据通路

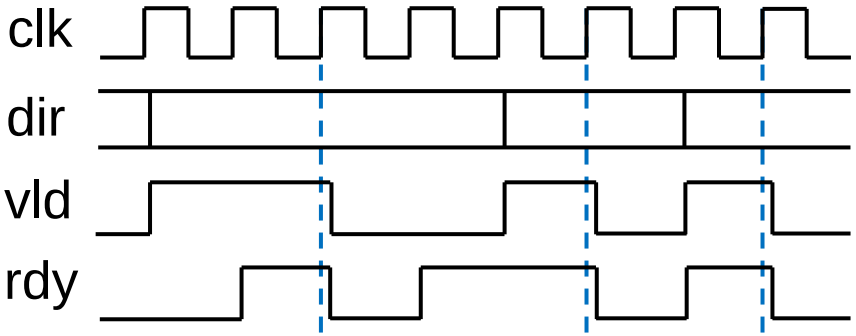
- **DIR：数据输入寄存器，8位**
- **VLD：DIR有效标志，1位**
- **SIR：移位输入寄存器，9位**
  - 左移，保存8位数据和1停止位
- **CNTc：时钟计数器，4位**
  - 7 ~ 0或15 ~ 0递减计数
- **CNTb：位计数器，4位**
  - 8 ~ 0递减计数

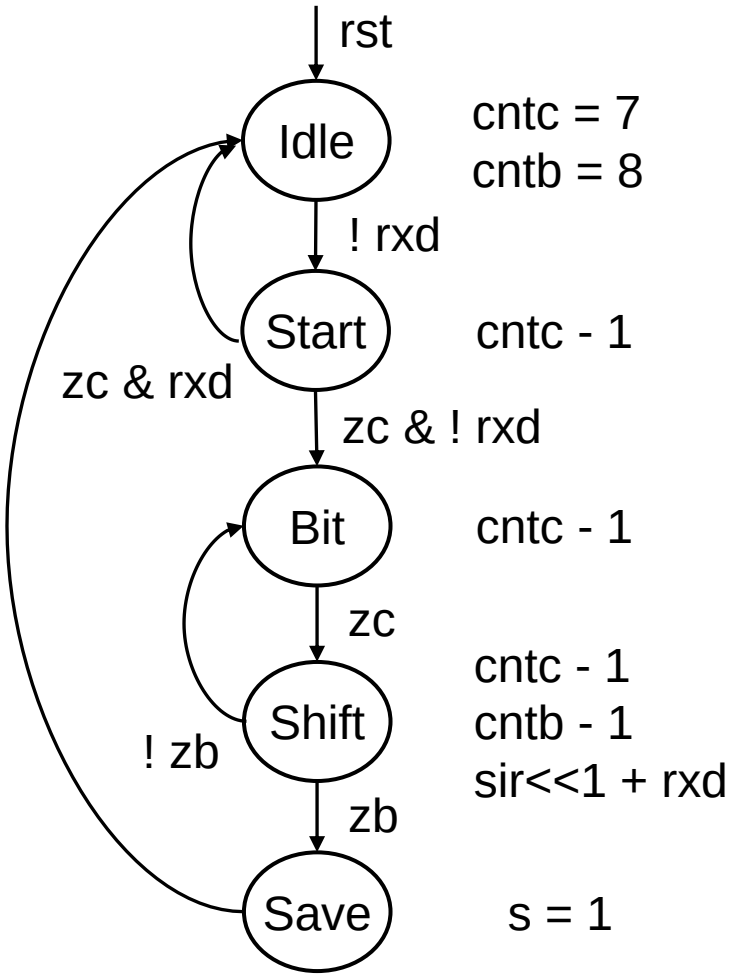




接收器状态图

- **复位时，vld清零**
- **从rxd接收到1个数据时，s = 1**
  - 若vld = 0，则保存该数据 (dir <= sir)，并将vld置1
  - 否则来不及处理，丢弃该数据
- **当vld和rdy均为1时，vld清零**
  - 表示dir数据已被DCP接收





实验要求

- **设计片上系统和应用程序，评估排序性能**
- **优化软硬件系统，提高排序性能**
  - 优化排序算法，降低执行指令条数
  - 增加数据Cache，以及优化Cache，提高Cache命中率
  - 增加指令分支预测，提高流水线执行效率
  - 优化数据通路，提高运行时钟频率

The End