Computer Organization and Architecture

Course Design

A parallel output controller

Jiang Yuxuan (04016616)
Liu Yuhan (04016602)

School of Information Science and Engineering
Southeast University

2019.3

1. Purpose

The purpose of this project is to design and simulate a parallel output controller (POC), which acts an interface between system bus and printer.

2. Introduction and requirements

POC is one of the most common I/O modules, namely the parallel output controller. It plays the role of an interface between the computer system bus and the peripheral (such as a printer or other output devices).

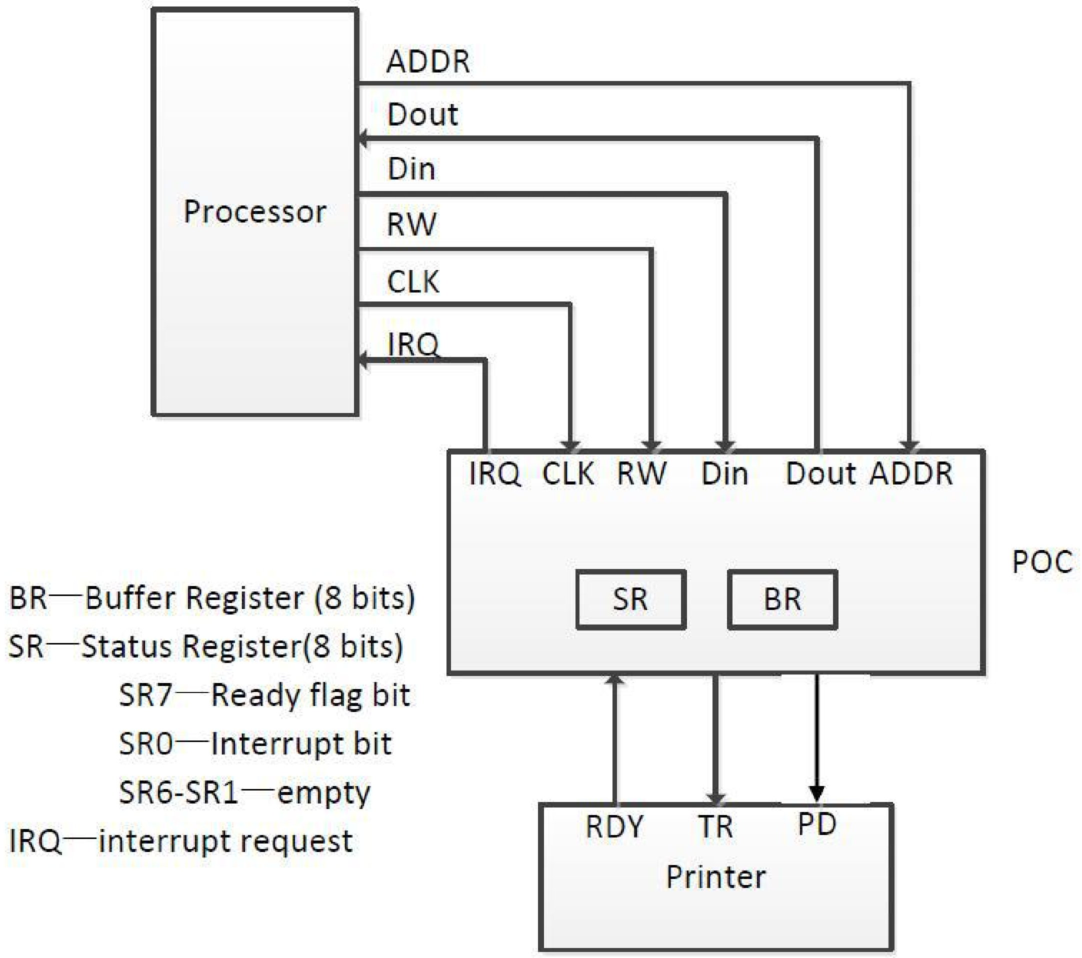


Figure 1 Printer Connection

Figure 1 shows the connecting of a printer to the system bus through the POC. The communication between POC and the printer is controlled by a "handshake" protocol illustrated in **Figure 2**.

Polling mode:

SR0 is always 0.

a. RW=0, ADDR=0. The CPU selects the SR register by selecting the appropriate address to query the status of SR7, which is sent to CPU through Dout.
b. If SR7=1, ADDR=1 and RW=1. The CPU selects the BR register and writes one byte of data to be printed through Din to BR.
c. ADDR=0, RW=1. It means that the CPU writes to SR7 and sets the SR7 register to 0 via Din. This indicates that the CPU has written new data which has not yet been processed.
d. If the POC detects that the SR7 register is set to 0, it starts handshaking with the printer. Then, POC sets the SR7 register to 1, which is the "ready" state.

Interrupt mode:

SR0 is always 1.

a. Let IRQ=not(SR7 and SR1). POC sends the interrupt request IRQ signal after sending the data to the printer and setting SR7 to 1 (ready).
b. After the CPU receives the IRQ signal, it makes ADDR = 1, RW = 1, which means CPU will no longer query SR7, but will select the BR directly and write the data to BR through Din.
c. ADDR=0, RW=1. It means that the CPU writes to SR7 and sets the SR7 register to 0 via Din. This indicates that the CPU has written new data which has not yet been processed.
d. If the POC detects that the SR7 register is set to 0, it starts handshaking with the printer. Then, POC sets the SR7 register to 1. Since SR0 = 1, the IRQ signal is pulled low to a low level 0, which means that an interrupt request is issued.

The communication between POC and the printer is controlled by a “handshake” protocol illustrated in Figure 2.

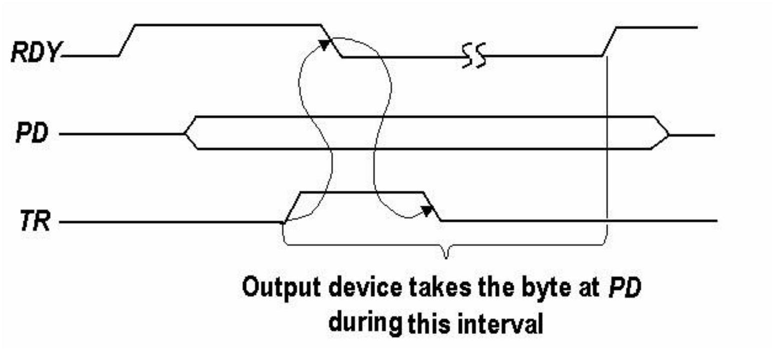
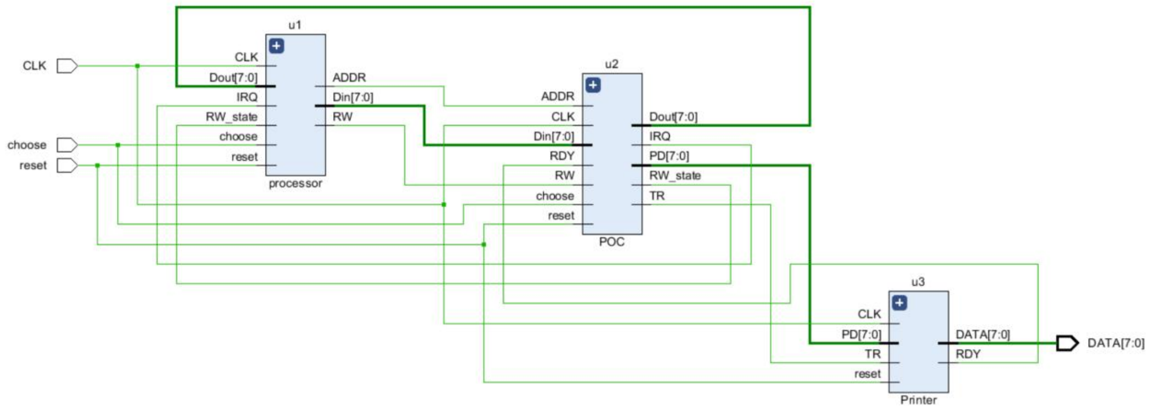


Figure 2 The handshake-timing diagram between POC and the printer

The handshaking process is described as follows: When the printer is ready to receive a character, it holds RDY=1.The POC must then hold a character at PD (parallel data) port and produce a pulse at the terminal TR (transfer request). The printer will change RDY to 0, take the character at PD and hold the RDY at 0 until the character has been printed (e.g. 5 or 10ms), then set RDY=1 again when it is ready to receive the next character. (Suppose the printer has only a one character "buffer" register, so that each character must be printed before the next character is sent).

In order to ease your design work, the further explanations of the POC operations and some design hints are given as follows: The buffer register BR is used to hold a character that has been sent via the system bus while that the character is being transferred to the printer. The status register SR is used for two control functions: SR7 serves as a ready flag for system bus transfers to BR (like the printer RDY signal for transfers from POC to the printer), and SR0 is used to enable or disable interrupt requests from POC. If SR0=1, then POC will interrupt when it is ready to receive a character (i.e., when SR7=1). If SR0=0, then POC will not interrupt. The other bits of SR are not used and empty.

3. Top module



4. Meaning of Symbols:

Processor:

Port name	in or out	Functional description
CLK	in	Clock signal
Dout	in	Data from POC to Processor
IRQ	in	Interrupt request signal from POC to Processor, 0 for interrupt
choose	in	Choose working mode, 0 for interrupt and 1 for polling

reset	in	Reset signal, 1 for reset
ADDR	out	Target address indicator, 0 for SR and 1 for BR
Din	out	Data from Processor to POC
RW	out	Read or write control signal, 0 for read and 1 for write
RW_state	in	The status of data exchange between CPU and POC

POC:

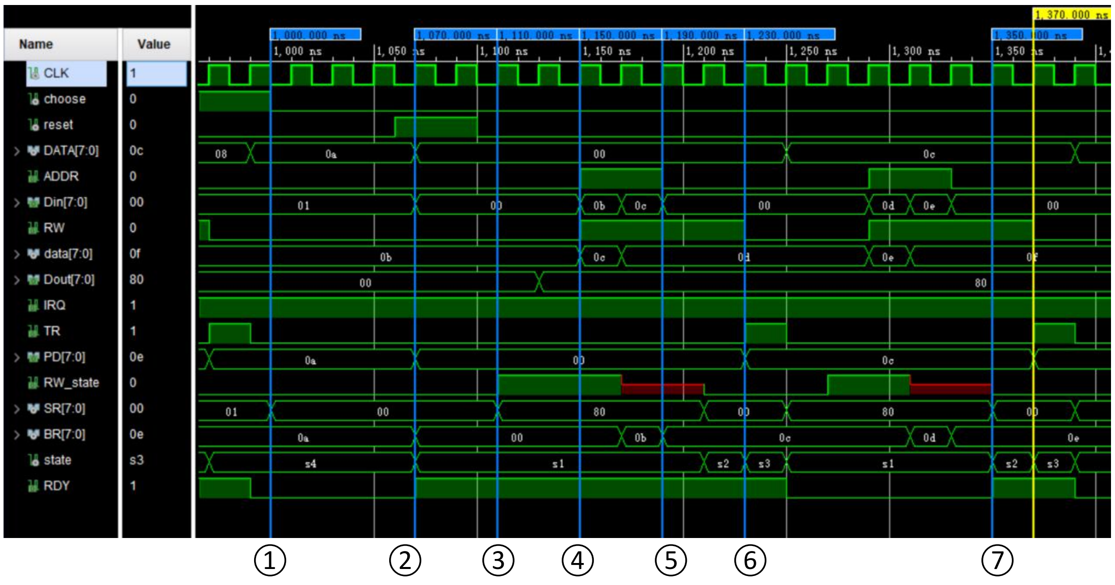
Port name	in or out	Functional description
ADDR	in	Target address indicator
Dout	out	Data from POC to Processor
Din	in	Data from Processor to POC
RW	in	Read or write control signal
reset	in	Reset signal
CLK	in	Clock signal
IRQ	out	Interrupt request signal from POC to Processor
RDY	in	Determine if the printer is ready to receive data
TR	out	Transfer request signal from poc to printer
PD	out	Send data from POC to printer
choose	in	Choose working mode
RW_state	out	The status of data exchange between CPU and POC: when polling mode, '1' means processor selects SR and reads SR, 'X' means processor selects BR and writes BR, '0' means processor selects SR and writes SR. when interrupt mode, '1' means processor selects BR and writes BR, '0' means processor selects SR and writes SR.

Printer:

Port name	in or out	Functional description
CLK	in	Clock signal
reset	in	Reset signal
TR	in	Receive transfer request signal from POC
PD	in	Receive data from POC
RDY	out	Printer ready to receive data from POC, 1 for ready
DATA	out	Print data

5. Simulation results

1、In polling mode



①When choose=’0’, the system enter into polling mode, so SR0 is set ‘0’.

②When reset is set ‘1’, ADDR、Din、RW、SR、BR、Dout、TR、PD、RW_state、DATA are set ‘1’ and RDY、IRQ are set ‘1’, state is set s1.

③After reset SR7 is set ‘1’. Processor selects SR and reads SR, so ADDR=’0’ and RW=’0’, and the value of SR is sent to Dout.

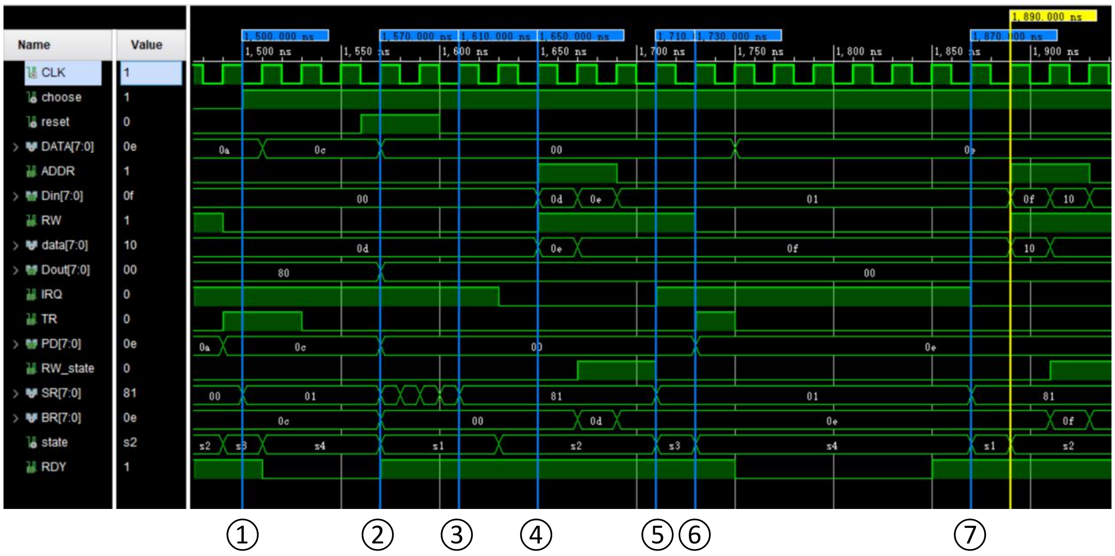
④If SR7=1, the processor selects BR and writes a character into BR. At this period, processor sets ADDR and RW to 1. And BR is changed.

⑤Then processor clears SR7 to indicate that the new character has been written into BR and not printed yet.

⑥ When POC detects that SR7 is set to 0, POC then proceeds to start the handshaking operations with the printer, and the data is passed to the printer through the data line PD and it and the printing time lasts for 5 clock cycles. During the handshaking operations between POC and printer, the processor continues to fetch and execute instructions. It happens to read SR, it finds SR7=0 and hence dose not attempt to send another character to the POC.

⑦After sending character to printer, POC sets the SR7 to 1, which indicates POC is ready to receive another character from the processor. The transfer cycle can now repeat.

2、In interrupt mode,



①When choose=’1’, the system enter into polling mode, so SR0 is set ‘1’.
②When reset is set ‘1’, ADDR、Din、RW、SR、BR、Dout、TR、PD、RW_state、DATA are set ‘1’ and RDY、IRQ are set ‘1’, state is set s1.
③After sending character to printer, POC sets the SR7 to 1, since SR0=1, the interrupt request signal (IRQ) is set to 0, which indicate an effective interrupt signal to the processor.
④When the processor detects the effective IRQ signal, the processor directly selects BR and writes a character into BR.
⑤And then the processor sets the SR7 to 0, which indicates that the new character has been written into BR and not printed yet.
⑥When POC detects that SR7 is set to 0, POC then proceeds to start the handshaking operations with the printer. It lasts for 5 clock cycles.
⑦After sending character to printer, POC sets the SR7 to 1, which indicates POC is ready to receive another character from the processor. The transfer cycle can now repeat. During the handshaking operations between POC and printer, the processor does not try to access POC until it receives the interrupt request signal.

6. Conclusions and discussions

In this experiment, our group designed a complete system including POC, Processor and the printer. And this system has the function to send the certain data to POC and let POC sent to the printer and print the data out. But there are two issues should be considered.
First of all, the IRQ signal is used to send the data message to CPU, which means the printer is ready to receive the next data. But the bus can connect many controllers.So under this condition, how to identify where a certain IRQ comes from is truly a problem needed to be solved.
Secondly, in the interrupt mode, if the interrupt signal is disabled, POC will do nothing in this process. So solutions must be found to solve the problem. CPU may issue request signal from time to time to make sure that POC is ready to receive the

data. On the other hand, during the handshaking operations between POC and printer, the processor continues to fetch and execute instructions in polling mode. In this way, we can solve it by adding a line between POC and processor.
As a parallel output controller, POC module to act as an interface between CPU and printer. Form the simulation wave, we can see that our program meets the designs requirements. In this task, all three devices work at the same clock frequency. In fact, the operating frequencies of the three devices are different. Among them, the CPU is the highest and the processor is the lowest. Here, we only simulate the timing characteristics of the POC and do not distinguish between operating frequencies. If a more complex system is needed, such as connect multiple printers, it can be achieved by making improvements in this system.

7. Appendix

```
-------------------------------------------------top.vhd-------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TOP is
Port (
      CLK : in STD_LOGIC;       --  输入时钟信号
      choose : in STD_LOGIC; --  工作方式选择信号
      reset : in STD_LOGIC;    --  重置工作方式信号
      DATA : out STD_LOGIC_VECTOR(7 downto 0)
  );
end TOP;

architecture Behavioral of TOP is

component processor is        -- Processor 实体
        Port (
                  CLK : in STD_LOGIC;       --  输入时钟信号
                  Dout : in STD_LOGIC_VECTOR(7 downto 0); --  输入八位数据（由 POC 的 dout 而来）
                  IRQ : in STD_LOGIC; --  输入中断请求信号（由 POC 而来）
                  choose : in STD_LOGIC; --  选择工作方式信号
                  reset : in STD_LOGIC;    --  重置工作方式信号
                  ADDR : out STD_LOGIC; --  地址线
                  Din : out STD_LOGIC_VECTOR(7 downto 0);     --  输出八位数据（由 POC 的 din 而
来）
                  RW : out STD_LOGIC; --  输出读写信号
                  RW_state :in STD_LOGIC-----读写状态
          );
end component;
```

```
component POC is        -- POC 实体
        Port (

              ADDR : in STD_LOGIC;    --  输入地址
              Din : in STD_LOGIC_VECTOR(7 downto 0);      --  输入八位数据
              RW : in STD_LOGIC;                                          --  输入读写信号
              CLK : in STD_LOGIC;        --  输入时钟信号
              reset : in STD_LOGIC;    --  重置信号
              RDY : in STD_LOGIC;        --  输入 Ready 信号，由 Printer 模块传入
              Dout : out STD_LOGIC_VECTOR(7 downto 0); --  输出八位数据（至 Processor）
              IRQ : out STD_LOGIC;      --  输出中断请求信号（至 Processor）
              TR : out STD_LOGIC;        -- Transfer Request 传输请求信号，发给 Printer，表明数据
可以开始传输
              PD : out STD_LOGIC_VECTOR(7 downto 0); --  由 POC 传输数据至 Printer
              RW_state :out STD_LOGIC;-----读写状态
              choose : in STD_LOGIC --  选择工作方式信号
        );
end component;

component printer is
        Port (
              CLK : in STD_LOGIC; --  时钟信号
              reset : in STD_LOGIC; --  重置信号
              TR : in STD_LOGIC; --  传输请求信号
              PD : in STD_LOGIC_VECTOR(7 downto 0); --  打印数据
              RDY : out STD_LOGIC;      --  打印机准备好信号
              DATA: out STD_LOGIC_VECTOR(7 downto 0):="00000000"
          );
end component;


signal ADDR : STD_LOGIC:='0'; --  地址信号
signal Din : STD_LOGIC_VECTOR(7 downto 0):="00000000";      --  数据传入 POC 信号
signal Dout : STD_LOGIC_VECTOR(7 downto 0):="00000000";    --  数据传出 POC 信号
signal RW : STD_LOGIC:='0';                                        --  表征读写信号
signal IRQ : STD_LOGIC:='0';                                      --  中断请求信号
signal RDY : STD_LOGIC:='0';                                      --  准备好信号
signal TR : STD_LOGIC:='0';                                       --  传输请求信号
signal PD : STD_LOGIC_VECTOR(7 downto 0):="00000000";        --  打印数据选择信号方式信号
signal RW_state : STD_LOGIC:='0';-----读写状态
begin

u1 : processor port map (CLK, Dout, IRQ, choose, reset, ADDR, Din, RW,RW_state);
u2 : POC port map (ADDR, Din, RW, CLK, reset, RDY, Dout, IRQ, TR, PD,RW_state,choose);
```

```
u3 : printer port map (CLK, reset, TR, PD, RDY,DATA);
end Behavioral;

-----------------------------------------------processor.vhd---------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity processor is
Port (
        CLK : in STD_LOGIC;         --  输入时钟信号
        Dout : in STD_LOGIC_VECTOR(7 downto 0); --  输入八位数据（由 POC 的 dout 而来）
        IRQ : in STD_LOGIC; --  输入中断请求信号（由 POC 而来）
        choose : in STD_LOGIC; --  选择工作方式信号
        reset : in STD_LOGIC;     --  重置工作方式信号
        ADDR : out STD_LOGIC; --  地址线
        Din : out STD_LOGIC_VECTOR(7 downto 0);      --  输出八位数据（由 POC 的 din 而来）
        RW : out STD_LOGIC; --  输出读写信号
        RW_state :in STD_LOGIC-----读写状态
  );
end processor;

architecture Behavioral of processor is
signal data : std_logic_vector(7 downto 0 ) := "00000000";

begin

process(CLK,reset,choose,IRQ)
begin

if CLK'event and CLK = '1' then

            if    reset='1'then --  初始化
            ADDR <= '0';
            Din <= "00000000";
            RW <= '0';
else

if choose = '0' then --  选择工作方式为查询,此时 SR0=0
                ADDR <= '0'; --  选择 SR
                RW<='0';---读 SR
if Dout(0)='0' then
```

```
          if Dout(7)='1' then ----如果 SR=1,process select BR and write BR
          if RW_state='0' then
          ADDR <= '0'; --  选择 SR
            RW<='0';---读 SR
            end if;
            if RW_state='1' then
            ADDR <= '1'; --  选择 BR 写 BR,然后 process write SR7<=0
            RW<='1';
            Din<=data;
            data<=data+1;
            end if;

          if    RW_state='X'then
          ADDR <= '0';
          RW<='1';
          Din<="00000000";
          end if;
              end if;
            end if;
    end if;

    if choose='1'then -------选择工作方式为中断，此时 SR0=1
                 ADDR <= '0'; --  选择 SR
                 RW<='0';---读 SR
if IRQ='0' then-----IRQ=0,select BR and write BR
               if RW_state='0'then
                       ADDR<='1';
                       RW<='1';
                       Din<=data;
                       data<=data+1;
                 end if;
                 if RW_state='1'then
                   ADDR<='0';-----SR7<=0
                   RW<='1';
                   Din<="00000001";
                   end if;
      end if;
  end if;
end if;
end if;
end process;
end Behavioral;

----------------------------------------POC.vhd-----------------------------------------------------------
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity processor is
Port (
        CLK : in STD_LOGIC;         --  输入时钟信号
        Dout : in STD_LOGIC_VECTOR(7 downto 0); --  输入八位数据（由 POC 的 dout 而来）
        IRQ : in STD_LOGIC; --  输入中断请求信号（由 POC 而来）
        choose : in STD_LOGIC; --  选择工作方式信号
        reset : in STD_LOGIC;    --  重置工作方式信号
        ADDR : out STD_LOGIC; --  地址线
        Din : out STD_LOGIC_VECTOR(7 downto 0);      --  输出八位数据（由 POC 的 din 而来）
        RW : out STD_LOGIC; --  输出读写信号
        RW_state :in STD_LOGIC-----读写状态
  );
end processor;

architecture Behavioral of processor is
signal data : std_logic_vector(7 downto 0 ) := "00000000";

begin

process(CLK,reset,choose,IRQ)
begin

if CLK'event and CLK = '1' then


            if    reset='1'then --  初始化
            ADDR <= '0';
            Din <= "00000000";
            RW <= '0';
else

if choose = '0' then --  选择工作方式为查询,此时 SR0=0
                ADDR <= '0'; --  选择 SR
                RW<='0';---读 SR
if Dout(0)='0' then
          if Dout(7)='1' then ----如果 SR=1,process select BR and write BR
          if RW_state='0' then
          ADDR <= '0'; --  选择 SR
            RW<='0';---读 SR
            end if;
          if RW_state='1' then
          ADDR <= '1'; --  选择 BR 写 BR,然后 process write SR7<=0
          RW<='1';
          Din<=data;
          data<=data+1;
          end if;

         if    RW_state='X'then
         ADDR <= '0';
         RW<='1';
         Din<="00000000";
         end if;
           end if;
          end if;
   end if;

   if choose='1'then -------选择工作方式为中断，此时 SR0=1
             ADDR <= '0'; --  选择 SR
             RW<='0';---读 SR
if IRQ='0' then-----IRQ=0,select BR and write BR
            if RW_state='0'then
                  ADDR<='1';
                  RW<='1';
                  Din<=data;
                  data<=data+1;
             end if;
             if RW_state='1'then
               ADDR<='0';-----SR7<=0
               RW<='1';
               Din<="00000001";
               end if;
     end if;
  end if;
end if;
end if;
end process;
end Behavioral;

-------------------------------------------printer.vhd------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
```

```
entity Printer is
Port (
CLK : in STD_LOGIC; --  时钟信号
reset : in STD_LOGIC; --  重置信号
TR : in STD_LOGIC; --  传输请求信号
PD : in STD_LOGIC_VECTOR(7 downto 0); --  打印数据
RDY : out STD_LOGIC; --  打印机准备好信号
DATA: out STD_LOGIC_VECTOR(7 downto 0):="00000000"
);
end Printer;
architecture Behavioral of Printer is
signal count:integer range 0 to 4:=0;
signal trt:std_logic:='0';
signal ready:std_logic:='1';
begin
process(CLK, TR, reset)
begin
if clk'event and clk = '1' then
if reset= '1' then
RDY <= '1';
DATA<="00000000";
else
if TR = '1' and trt='0' and ready='1' then------
RDY <= '0';
DATA<=PD;------取 PD
ready<='0';
elsif ready='0' then
count <= count + 1;
  if count=4 then---延时 5 个时钟周期，等待打印完成
  RDY<='1';-----RDY 置 1，准备接收下一个字符
  ready<='1';
  count<=0;
  end if;
else
RDY <= '1';
end if;
trt<=TR;
end if;
end if;
end process;
end Behavioral;

--------------------------------------------top_sim.vhd---------------------------------------------------
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;

entity top_sim is
--    Port ( );
end top_sim;

architecture Behavioral of top_sim is
component Top
            Port (
                    CLK : in STD_LOGIC;        --  输入时钟信号
                    choose : in STD_LOGIC; --  工作方式选择信号
                    reset : in STD_LOGIC;    --  重置工作方式信号
                    DATA : out STD_LOGIC_VECTOR(7 downto 0)
             );
        end component;
signal CLK : STD_LOGIC := '0';
signal choose : STD_LOGIC := '0';
signal reset : STD_LOGIC := '0';
signal DATA : STD_LOGIC_VECTOR(7 downto 0):="00000000";
constant clk_period : time :=20 ns;

begin
uut : Top port map (CLK=>CLK, choose=>choose, reset=>reset,DATA=>DATA);
process
begin
    wait for clk_period/2;
    CLK<='1';
    wait for clk_period/2;
    CLK<='0';
end process;

process
begin
        reset<='0';
        wait for 60ns;
        reset<='1';
        wait for 40 ns;
        reset<='0';
        wait for 400ns;
end process;

process
begin
          choose<='0';
          wait for 500 ns;
          choose<='1';
          wait for 500 ns;
end process;
```