FPGA系統設計\_第十章\_系統設計

**10867024 廖育賢**

# **objective- THE PROBLEM AND PURPOSE**

建立一個泛用的輸入輸出界面，包括32點輸入輸出組合，並透過印表機界面由微電腦直接使用軟體控制。

# **procedure – DESIGN methods**

## 外部電路配置:

1. XA(3~0): 4為圓固定輸出阜。

2. XB(7~0): 8位元可程式輸入輸出阜，當輸出設為1時即可當輸入使用。

3. XC(3~0): 4位元固定輸入阜。

4. YA(3~0): 4位元固定輸出阜。

5. YB(7~0): 8位元可程式輸入輸出阜，當輸出設為1時即可當成輸入使用。

6. YC(3~0): 4位元固定輸入阜。

## 內部電路設計:

此電路分成五個控制模組，包括:

1. 印表機界面電路 : 由元件PRINTER.vhd來執行

2. 固定輸出電路 : 由元件OUTPORT4.vhd來執行，為4位元簡單輸出阜

3. 固定輸入電路 : 由元件INPORT4.vhd來執行，為4位元簡單輸入阜

4. 可程式輸入輸出 : 由元件IOPORT8.vhd來執行，為8位元可程式輸入輸出阜

5. 位置解碼電路 : 執行位址解碼工作

# **simulation results**

## program codes

### **ch010.vhd**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

library SYNOPSYS;

use SYNOPSYS.attributes.all;

entity INPORT4 is

port

( DB : inout STD\_LOGIC\_VECTOR (3 downto 0); --data bus

IP : in STD\_LOGIC\_VECTOR (3 downto 0); --input port

CS : in STD\_LOGIC; --chip select

RD : in STD\_LOGIC --read signal

);

end INPORT4;

architecture INPORT4\_ARCH of INPORT4 is

signal T : STD\_LOGIC; --tri-state signal

begin

T <= not RD and CS; --tri-state control

DB <= IP when T='1' else --read to data bus

"ZZZZ";

end INPORT4\_ARCH;

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

library SYNOPSYS;

use SYNOPSYS.attributes.all;

entity INPORT8 is

port

( DB : inout STD\_LOGIC\_VECTOR (7 downto 0);

IP : in STD\_LOGIC\_VECTOR (7 downto 0);

CS : in STD\_LOGIC;

RD : in STD\_LOGIC

);

end INPORT8;

architecture INPORT8\_ARCH of INPORT8 is

signal T : STD\_LOGIC; --tri-state signal

begin

T <= not RD and CS; --tri-state control

DB <= IP when T='1' else --read to data bus

"ZZZZZZZZ";

end INPORT8\_ARCH;

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

--This package is a version of the Synopsys package and has been

-- optimized for use with the Express compiler.

library SYNOPSYS;

use SYNOPSYS.attributes.all;

entity IOPORT8 is

port

( DB : inout STD\_LOGIC\_VECTOR (7 downto 0); --data bus

IO : inout STD\_LOGIC\_VECTOR (7 downto 0); --input/output

CS : in STD\_LOGIC; --chip select

RD : in STD\_LOGIC; --read signal

SWR : in STD\_LOGIC; --sync. write

CLK : in STD\_LOGIC --system clock

);

end IOPORT8;

architecture IOPORT8\_ARCH of IOPORT8 is

signal Q : STD\_LOGIC\_VECTOR (7 downto 0);

signal T,EC : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then --CLK rising

if EC='1' then --EC enable

Q <= DB; --write from data bus

end if;

end if;

end process;

GEN: for I in 0 to 7 generate

IO(I) <= Q(I) when Q(I)='0' else --open-collect output

'Z';

end generate;

DB <= IO when T='1' else --read to data bus

"ZZZZZZZZ";

T <= not RD and CS; --tri-state control

EC <= SWR and CS; --write enable

RT8\_ARCH;

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

library SYNOPSYS;

use SYNOPSYS.attributes.all;

entity OUTPORT4 is

port

( DB : inout STD\_LOGIC\_VECTOR (3 downto 0); --data bus

OP : out STD\_LOGIC\_VECTOR (3 downto 0); --output port

CS : in STD\_LOGIC; --chip select

SWR : in STD\_LOGIC; --sync write

CLK : in STD\_LOGIC --system clock

);

end OUTPORT4;

architecture OUTPORT4\_ARCH of OUTPORT4 is

signal EC : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then --CLK rising

if EC='1' then --CLK enable

OP <= DB; --latch output

end if;

end if;

end process;

EC <= SWR and CS; --CLK control

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

library SYNOPSYS;

use SYNOPSYS.attributes.all;

entity OUTPORT8 is

port

( DB : inout STD\_LOGIC\_VECTOR (7 downto 0);

OP : out STD\_LOGIC\_VECTOR (7 downto 0);

CS : in STD\_LOGIC;

WR : in STD\_LOGIC

);

end OUTPORT8;

architecture OUTPORT8\_ARCH of OUTPORT8 is

begin

process (WR)

begin

if WR'event and WR='1' then --WR rising edge

if CS='1' then --CS enable

OP <= DB; --write from data bus

end if;

end if;

end process;

end OUTPORT8\_ARCH;

--The IEEE standard 1164 package, declares std\_logic, rising\_edge(), etc.

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

library SYNOPSYS;

use SYNOPSYS.attributes.all;

entity PRINTER is

port

( CLK : in STD\_LOGIC; --4MHz clock

DB : inout STD\_LOGIC\_VECTOR (7 downto 0); --data bus

A : out STD\_LOGIC\_VECTOR (7 downto 0); --address bus

AD : in STD\_LOGIC\_VECTOR (7 downto 0); --data input

DI3,DI4,DI5,DI7 : out STD\_LOGIC; --status output

WR, RD, SEL,ALE : in STD\_LOGIC; --control input

SWR,SRD,SALE : out STD\_LOGIC; --sync. control

CS : in STD\_LOGIC --chip select

);

end PRINTER;

architecture PRINTER\_ARCH of PRINTER is

signal RD1 : STD\_LOGIC;

signal CNT : STD\_LOGIC;

signal DQ : STD\_LOGIC\_VECTOR (7 downto 0);

signal DM : STD\_LOGIC\_VECTOR (3 downto 0);

signal RD0,ALE0 : STD\_LOGIC;

begin

SIGNAL\_FILTER : block

signal D1,D2,DIN,FLT,DLY : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then --clock rising

D1<=DIN; D2<=D1; --delay

DLY <= FLT; --delay and RS flip-flop

FLT <= (D1 and D2) or ((D1 or D2) and FLT);

end if;

end process;

DIN <= (not RD or not WR or not ALE) and SEL;

RD0 <= not RD and FLT and not DLY;

ALE0 <= not ALE and FLT and not DLY;

SWR <= not WR and FLT and not DLY;

SALE <= ALE0;

SRD <= RD0;

end block SIGNAL\_FILTER;

ADDRESS\_LATCH : block

begin

process (CLK)

begin

if CLK'event and CLK='1' then --CLK rising

if ALE0='1' then --SALE enable

A <= AD; --address latch

end if;

end if;

end process;

end block ADDRESS\_LATCH;

DATA\_WRITE : block

begin

DB <= AD when WR='0' else --enable data bus

"ZZZZZZZZ";

end block DATA\_WRITE;

READ\_LATCH : block

signal EC : STD\_LOGIC;

begin

process (CLK)

begin

if CLK'event and CLK='1' then --CLK rising

if EC='1' then --CLK enable

DQ <= DB; --data latch

end if;

end if;

end process;

EC <= RD0 and not CNT;

end block READ\_LATCH;

READ\_MULTIPLEXER : block

begin

DM <= DQ(3 downto 0) when CNT='1' else --multiplexer

DQ(7 downto 4);

end block READ\_MULTIPLEXER;

READ\_DATA : block

begin

DI3 <= DM(0) when CS='1' else 'Z'; --tri-state output

DI4 <= DM(1) when CS='1' else 'Z';

DI5 <= DM(2) when CS='1' else 'Z';

DI7 <= not DM(3) when CS='1' else 'Z';

end block READ\_DATA;

READ\_CLOCK : block

begin

process (CLK)

begin

if CLK'event and CLK='1' then --clock rising

RD1<=RD0; --delay

end if;

end process;

end block READ\_CLOCK;

TER : block

signal RST,EC : STD\_LOGIC;

begin

process (RD1,RST)

begin

if RST='1' then --preset

CNT <= '0';

elsif CLK'event and CLK='1' then --CLK rising

if EC='1' then --CLK enable

CNT <= not CNT; --counter+1

end if;

end if;

end process;

RST <= ALE0;

EC <= RD1;

end block READ\_COUNTER;

end PRINTER\_ARCH;

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

library SYNOPSYS;

use SYNOPSYS.attributes.all;

is

port

( CLK : in STD\_LOGIC; --4MHz clock

AD : in STD\_LOGIC\_VECTOR (7 downto 0); --data input

WR, RD, SEL,ALE : in STD\_LOGIC; --control input

DI3,DI4,DI5,DI7 : out STD\_LOGIC; --status output

XA,YA : out STD\_LOGIC\_VECTOR (3 downto 0); --output port

XB,YB : inout STD\_LOGIC\_VECTOR (7 downto 0); --input/output

XC,YC : in STD\_LOGIC\_VECTOR (3 downto 0) --input port

);

end CH10;

architecture CH10\_ARCH of CH10 is

component PRINTER --\*\*\*\*\* printer interface \*\*\*\*\*

port

( CLK : in STD\_LOGIC; --system clock

DB : inout STD\_LOGIC\_VECTOR (7 downto 0); --data bus

A : out STD\_LOGIC\_VECTOR (7 downto 0); --address bus

AD : in STD\_LOGIC\_VECTOR (7 downto 0); --data input

DI3,DI4,DI5,DI7 : out STD\_LOGIC; --status output

WR, RD, SEL,ALE : in STD\_LOGIC; --control input

SWR,SRD,SALE : out STD\_LOGIC; --sync. control

CS : in STD\_LOGIC --chip select

);

end component;

component INPORT4 --\*\*\*\*\* 4-BIT simple input \*\*\*\*\*

port

( DB : inout STD\_LOGIC\_VECTOR (3 downto 0); --data bus

IP : in STD\_LOGIC\_VECTOR (3 downto 0); --data input

CS : in STD\_LOGIC; --chip select

RD : in STD\_LOGIC --read control

);

end component;

component OUTPORT4 --\*\*\*\*\* 4-BIT simple output \*\*\*\*\*

port

( DB : inout STD\_LOGIC\_VECTOR (3 downto 0); --data bus

OP : out STD\_LOGIC\_VECTOR (3 downto 0); --data output

CS : in STD\_LOGIC; --chip select

SWR : in STD\_LOGIC; --write control

CLK : in STD\_LOGIC --system clock

);

end component;

component IOPORT8 --\*\*\*\*\* 8-BIT input/output \*\*\*\*\*

port

( DB : inout STD\_LOGIC\_VECTOR (7 downto 0); --data bus

IO : inout STD\_LOGIC\_VECTOR (7 downto 0); --input/output

CS : in STD\_LOGIC; --chip select

RD : in STD\_LOGIC; --read control

SWR : in STD\_LOGIC; --write control

CLK : in STD\_LOGIC --system clock

);

end component;

signal DB : STD\_LOGIC\_VECTOR (7 downto 0);

signal A : STD\_LOGIC\_VECTOR (7 downto 0);

signal CS,CS0,CS1,CS2,CS3 : STD\_LOGIC;

signal SRD,SWR,SALE : STD\_LOGIC;

begin

SYSTEM\_CONNECT : block

begin

U1: PRINTER port map (CLK,DB,A,

AD,

DI3, DI4, DI5, DI7,

WR, RD, SEL, ALE,

SWR, SRD, SALE,

CS);

U2: OUTPORT4 port map (DB(3 downto 0), XA, CS0, SWR, CLK);

U3: INPORT4 port map (DB(3 downto 0), XC, CS0, RD);

U4: IOPORT8 port map (DB, XB, CS1, RD, SWR, CLK);

U5: OUTPORT4 port map (DB(3 downto 0), YA, CS2, SWR, CLK);

U6: INPORT4 port map (DB(3 downto 0), YC, CS2, RD);

U7: IOPORT8 port map (DB, YB, CS3, RD, SWR, CLK);

end block SYSTEM\_CONNECT;

CHIP\_SELECT : block

begin

CS <= '1' when A(7 downto 2)="000000" else '0';

CS0 <= CS and not A(1) and not A(0);

CS1 <= CS and not A(1) and A(0);

CS2 <= CS and A(1) and not A(0);

CS3 <= CS and A(1) and A(0);

end block CHIP\_SELECT;

end CH10\_ARCH;

### **CH010.UCF**

NET CLK LOC = B12;

NET XA<0> LOC = N2;

NET XA<1> LOC = T1;

NET XA<2> LOC = T2;

NET XA<3> LOC = U2;

NET XB<0> LOC = M3;

NET XB<1> LOC = M4;

NET XB<2> LOC = W3;

NET XB<3> LOC = W4;

NET XB<4> LOC = Y3;

NET XB<5> LOC = Y4;

NET XB<6> LOC = W6;

NET XB<7> LOC = Y6;

NET XC<0> LOC = V1;

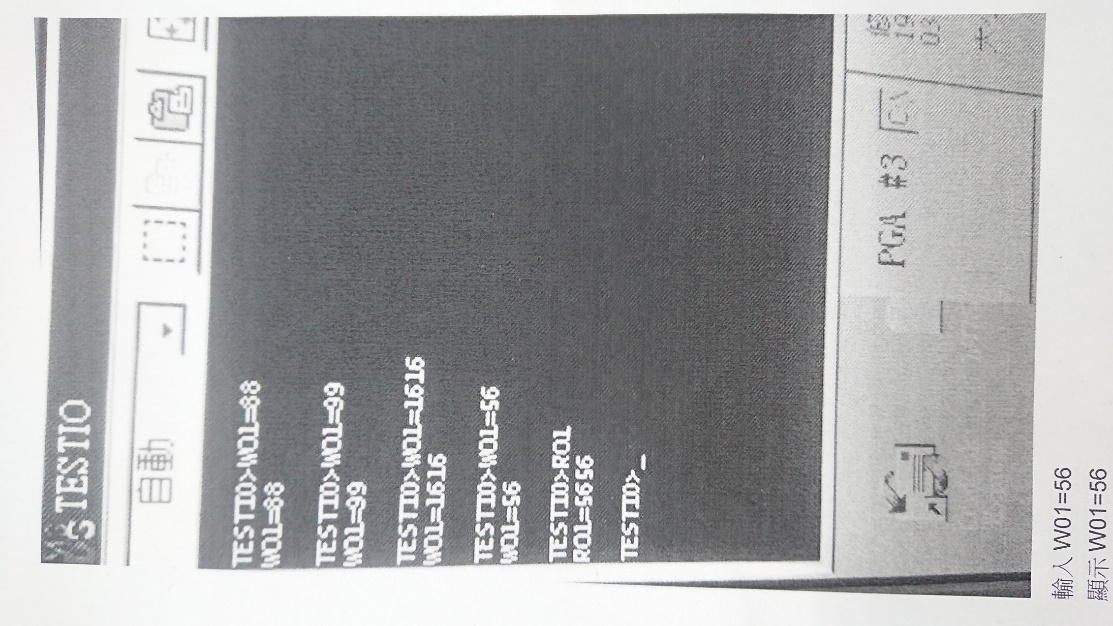
NET XC<1> LOC = V2;

NET XC<2> LOC = W1;

NET XC<3> LOC = W2;

## observations

### 1. 輸入W01=56, 顯示W01=56



### 2. XB0~7=0100(5) 1010(6)， XB7、XB5、XB3、XB0亮。

