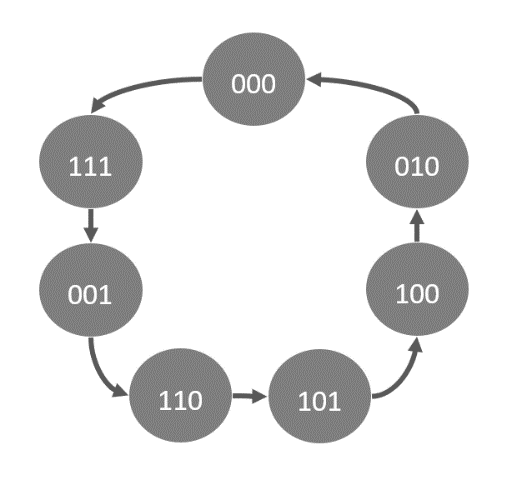
電機一乙 數位邏輯實習

第十週實習作業報告

組員：02葉峻呈 15邱宇柔

I.實驗目的與原理

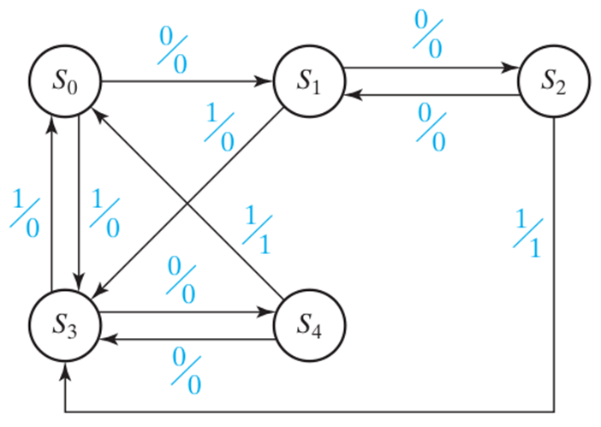
*A.作業1*

 利用J-K正反器推導出下圖狀態圖的計數器電路，並以狀態圖及布林代數式兩種方式在VHDL上實作。

圖一：計數器狀態圖

*B.作業2*

利用T正反器推導出下圖狀態圖的狀態機電路，並同樣以狀態圖及布林代數式兩種方式在VHDL上實作。



圖二：狀態機狀態圖

II.實驗過程

*A.作業1*

*B.作業2*

方法1(布林代數式)：

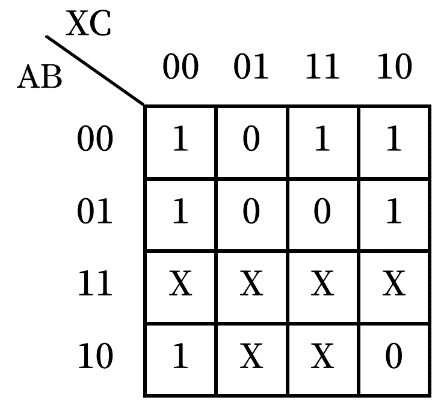
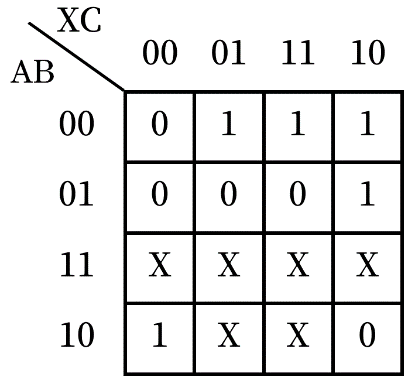
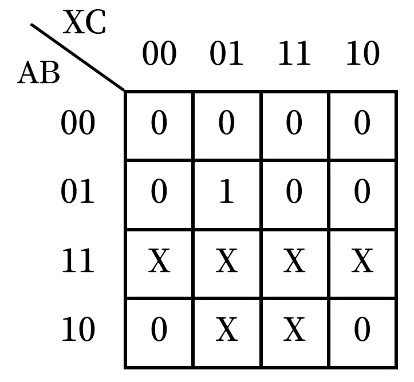
首先，我們將圖二的狀態圖轉為狀態表，並把圖中的S0~S4改為二進制，如下表。

表一：狀態表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Present State | Next State | | State Output | |
| X = 0 | X = 1 | X = 0 | X = 1 |
| 000(S0) | 001(S1) | 011(S3) | 0 | 0 |
| 001(S1) | 010(S2) | 011(S3) | 0 | 0 |
| 010(S2) | 001(S1) | 011(S3) | 0 | 1 |
| 011(S3) | 100(S4) | 000(S0) | 0 | 0 |
| 100(S4) | 011(S3) | 000(S0) | 0 | 1 |

接著，將Present State的最高位至最低位設為ABC，Next State為，

State Output為Z。並轉換為Next-State Map，如下圖。

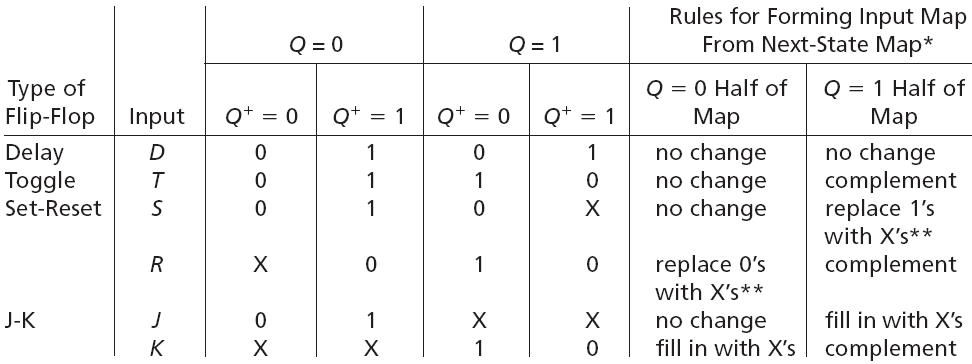


圖：卡諾圖

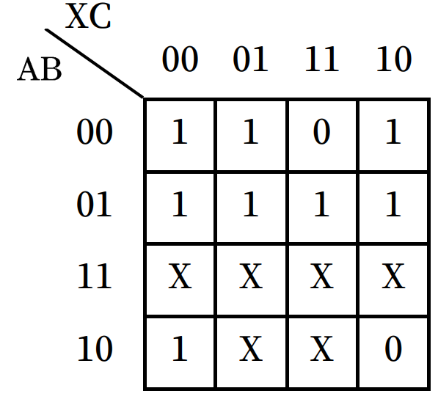
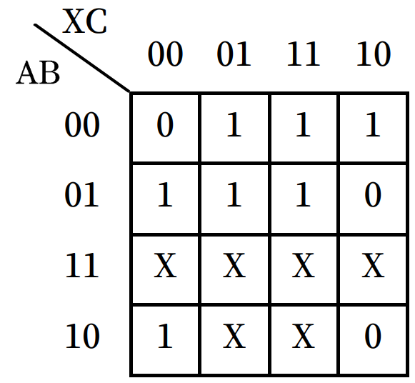
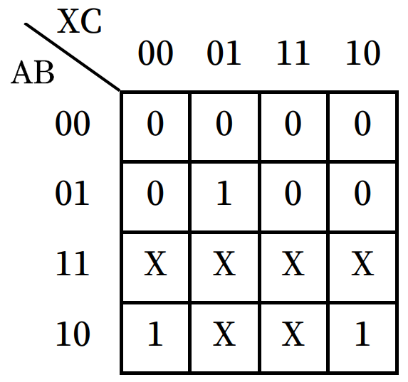
圖：卡諾圖

圖：卡諾圖

透過表二，我們可以再將Next-State Map轉換成T正反器的輸入，其卡諾圖如下。



表二：各正反器特性表



圖：卡諾圖

圖：卡諾圖

圖：卡諾圖

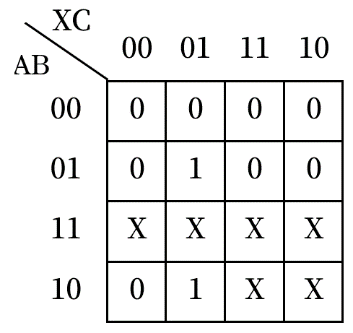
由此我們可以知道各個T正反器的輸入：

= A + BX’C

　　 = C + BX’ + AX’ + A’B’X

　　 = B + X’ + A’C’

根據表一，此狀態機之輸出Z與目前狀態、輸入訊號X有關，為Mealy machine，輸出Z之卡諾圖及布林代數式如下。



圖：Z卡諾圖

Z = AX + BC’X

VHDL如下。

library ieee;

use ieee.std\_logic\_1164.all;

entity test is

port (

clk, x: in std\_logic;

a\_out, b\_out, c\_out, f: out std\_logic);

end test;

architecture test\_boolean\_arch of test is

signal Ta, Tb, Tc : std\_logic;

signal a, b, c : std\_logic := '0';

begin

f <= (a and x) or (b and not c and x);

a\_out <= a;

b\_out <= b;

c\_out <= c;

Ta <= a or (b and not x and c);

Tb <= c or (b and not x) or (a and not x) or (not a and not b and x);

Tc <= b or not x or (not a and not c);

process(clk) is

begin

if(rising\_edge(clk)) then

a <= Ta xor a;

b <= Tb xor b;

c <= Tc xor c;

end if;

end process;

end test\_boolean\_arch;

輸入輸出

輸出Z之布林代數式

正反器輸入之布林代數式

T正反器封包

方法二(狀態圖)：

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.numeric\_std.all;

entity test is

port (

-- Input ports

clk, x: in std\_logic;

-- Output ports

a\_out, b\_out, c\_out, f: out std\_logic);

end test;

architecture state\_graph of test is

signal z : std\_logic\_vector(2 downto 0) := "000";

Begin

a\_out <= z(0);

b\_out <= z(1);

c\_out <= z(2);

f <= (z(2) and x) or (z(1) and not z(0) and x);

process(clk) is

begin

if(rising\_edge(clk)) then

if(z = "000" and x = '0') then z <= "001";

elsif(z = "000" and x = '1') then z <= "011";

elsif(z = "001" and x = '0') then z <= "010";

elsif(z = "001" and x = '1') then z <= "011";

elsif(z = "010" and x = '0') then z <= "001";

elsif(z = "010" and x = '1') then z <= "011";

elsif(z = "011" and x = '0') then z <= "100";

elsif(z = "011" and x = '1') then z <= "000";

elsif(z = "100" and x = '0') then z <= "011";

elsif(z = "100" and x = '1') then z <= "000";

end if;

end if;

end process;

end state\_graph;

III.模擬驗證

*A.作業1*

*B.作業2*

IV.實驗結果與成果討論

1. *作業1*

*B.作業2*

V.實驗心得