Karnaugh Map & Don't-Care Conditions

2016/10/24

范真瑋

Ex1

實驗內容：

Exercise 1

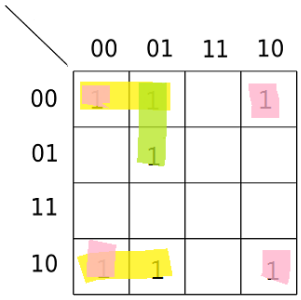
- Simplify the Boolean function $F(w,x,y,z) = \Sigma\,(0,1,2,5,8,9,10)$ using Karnaugh maps to (1) the simplest sum of products (*F1*), and (2) the simplest product of sums (*F2*)
- Verify *F1* = *F2* (using Verilog Dataflow modeling)

程式碼：

```
module Ex1(F1, F2, w, x, y, z);
output F1, F2;
input w, x, y, z;

wire F1, F2, w, x, y, z;

assign F1 = ~x&~z | ~x&~y | ~w&~y&z;
assign F2 = (~y|~z) & (~w|~x) & (~x|z);

endmodule
```

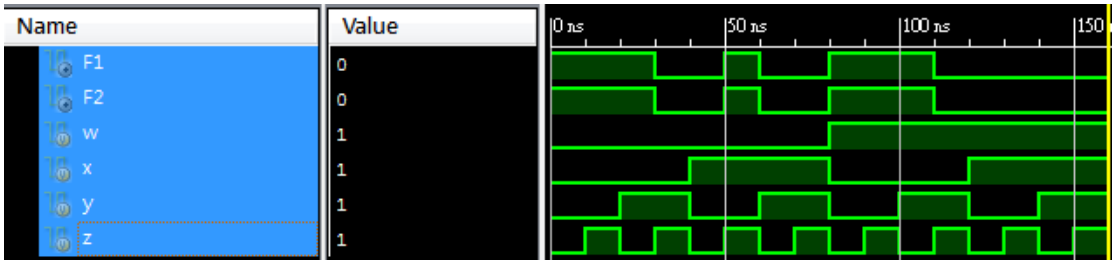
```
module Ex1_tb;
    reg w;
    reg x;
    reg y;
    reg z;

    // Outputs
    wire F1;
    wire F2;

    // Instantiate the Unit Under Test (UUT)
    Ex1 uut (
        .w(w),
        .x(x),
        .y(y),
        .z(z),
        .F1(F1),
        .F2(F2)
    );

    initial begin
        // Initialize Inputs
        w=0;x=0;y=0;z=0;
        #10 w=0;x=0;y=0;z=1;
        #10 w=0;x=0;y=1;z=0;
        #10 w=0;x=0;y=1;z=1;
        #10 w=0;x=1;y=0;z=0;
        #10 w=0;x=1;y=0;z=1;
        #10 w=0;x=1;y=1;z=0;
        #10 w=0;x=1;y=1;z=1;
        #10 w=1;x=0;y=0;z=0;
        #10 w=1;x=0;y=0;z=1;
        #10 w=1;x=0;y=1;z=0;
        #10 w=1;x=0;y=1;z=1;
        #10 w=1;x=1;y=0;z=0;
        #10 w=1;x=1;y=0;z=1;
        #10 w=1;x=1;y=1;z=0;
        #10 w=1;x=1;y=1;z=1;
        #10 $finish;

    end
endmodule
```

實驗結果及分析：



F1 = ~x&~z | ~x&~y | ~w&~y&z

F2 = (~y&~z) & (~w|~x) & (~x&z)



由實驗結果圖中的波形得知 F1 = F2

Ex2

實驗內容：

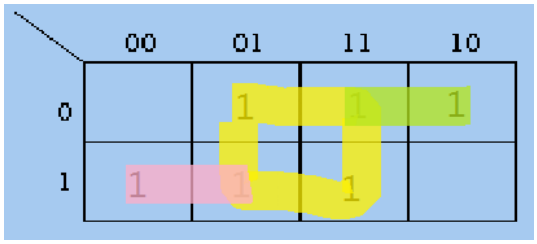
Exercise 2

- Simplify the following Boolean function using Karnaugh maps: $F(x,y,z) = \Sigma\,(1,2,3,4,5,7)$
- Find the simplest sum of products (*F1*) and draw its logic diagram (two-level implementation)
  - Implement *F1* with two-level NAND gate circuit (denoted as *F2*)
  - Verify *F1* = *F2* (using Verilog Structural level modeling)
- Find the simplest product of sums (*F3*) and draw its logic diagram (two-level implementation)
  - Implement *F3* with two-level NOR gate circuit (denoted as *F4*)
  - Verify *F3* = *F4* (using Verilog Structural level modeling)

程式碼：

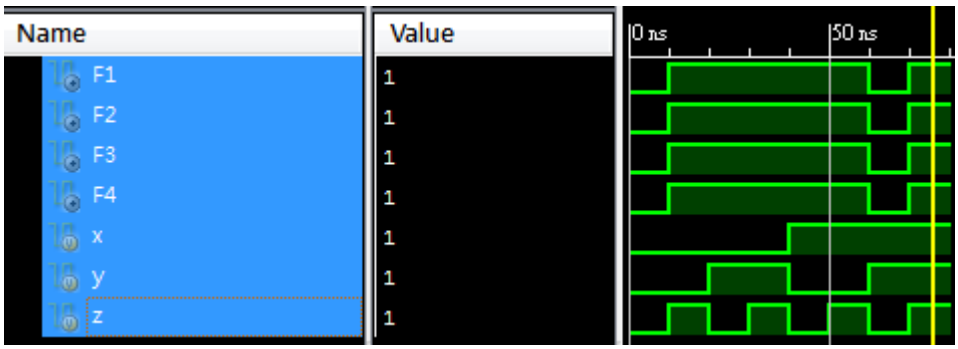
```
module Ex2(F1, F2, F3, F4, x, y, z);
output F1, F2, F3, F4;
input x, y, z;

wire F1, F2, F3, F4, x, y, z, t1, t2, t3, t4;

assign F1 = z | x&~y | ~x&y;
nand(t1, x, ~y);
nand(t2, ~x, y);
nand(F2, ~z, t1, t2);

assign F3 = (x|y|z) & (~x|~y|z);
nor(t3, x, y, z);
nor(t4, ~x, ~y, z);
nor(F4, t3, t4);

endmodule

module Ex2_tb;

   // Inputs
   reg x;
   reg y;
   reg z;

   // Outputs
   wire F1;
   wire F2;
   wire F3;
   wire F4;

   // Instantiate the Unit Under Test (UUT)
   Ex2 uut (
      .F1(F1),
      .F2(F2),
      .F3(F3),
      .F4(F4),
      .x(x),
      .y(y),
      .z(z)
   );

   initial begin
      // Initialize Inputs
      x = 0;y = 0;z = 0;
      #10 x = 0;y = 0;z = 1;
      #10 x = 0;y = 1;z = 0;
      #10 x = 0;y = 1;z = 1;
      #10 x = 1;y = 0;z = 0;
      #10 x = 1;y = 0;z = 1;
      #10 x = 1;y = 1;z = 0;
      #10 x = 1;y = 1;z = 1;
      #10 $finish;
   end
endmodule
```

實驗結果及分析：



F1 = z | x&~y | ~x&y

F3 = (x|y|z) & (~x|~y|z)



由實驗結果圖中的波形得知 F1 = F2，F3 = F4，且 F1 = F2 = F3 = F4

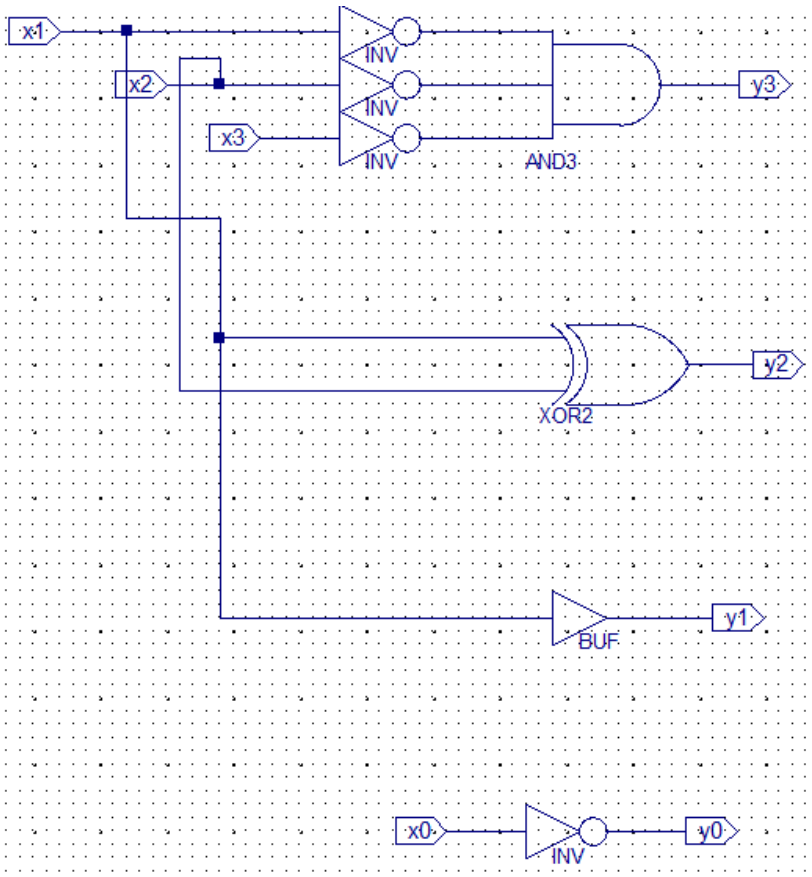
Ex3

實驗內容：

Exercise 3: Nine's Complement Converter (2/2)

- Design the nine's complement converter using Karnaugh maps and don't-care conditions, draw the logic diagram of the nine's complement converter and verify the circuit (using Schematic)

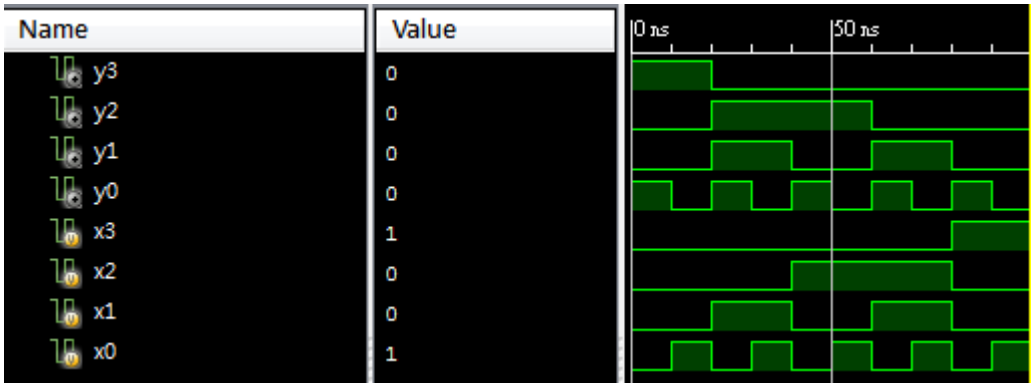
電路圖：



程式碼：

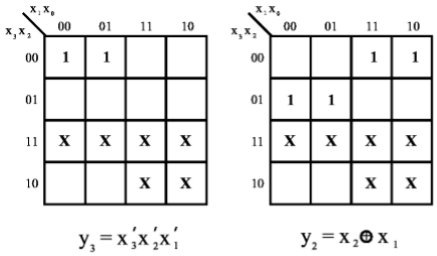
```
module Ex3_Ex3_sch_tb();

// Inputs
    reg x3;
    reg x0;
    reg x1;
    reg x2;

// Output
    wire y3;
    wire y2;
    wire y0;
    wire y1;

// Bidirs

// Instantiate the UUT
    Ex3 UUT (
        .x3(x3),
        .x0(x0),
        .y3(y3),
        .y2(y2),
        .y0(y0),
        .x1(x1),
        .y1(y1),
        .x2(x2)
    );
// Initialize Inputs
         initial begin
        x3=0;x2=0;x1=0;x0=0;
        #10  x3=0;x2=0;x1=0;x0=1;
        #10  x3=0;x2=0;x1=1;x0=0;
        #10  x3=0;x2=0;x1=1;x0=1;
        #10  x3=0;x2=1;x1=0;x0=0;
        #10  x3=0;x2=1;x1=0;x0=1;
        #10  x3=0;x2=1;x1=1;x0=0;
        #10  x3=0;x2=1;x1=1;x0=1;
        #10  x3=1;x2=0;x1=0;x0=0;
        #10  x3=1;x2=0;x1=0;x0=1;
        #10  $finish;
        end
endmodule
```

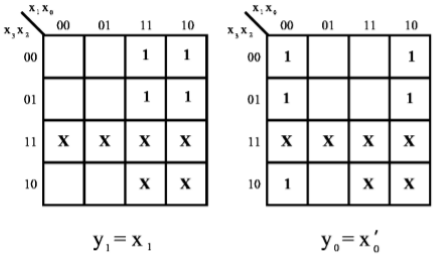
實驗結果及分析：



Digits		Nine's Complements	
Decimal	BCD	Decimal	BCD
	$x_3x_2x_1x_0$		$y_3y_2y_1y_0$
0	0 0 0 0	9	1 0 0 1
1	0 0 0 1	8	1 0 0 0
2	0 0 1 0	7	0 1 1 1
3	0 0 1 1	6	0 1 1 0
4	0 1 0 0	5	0 1 0 1
5	0 1 0 1	4	0 1 0 0
6	0 1 1 0	3	0 0 1 1
7	0 1 1 1	2	0 0 1 0
8	1 0 0 0	1	0 0 0 1
9	1 0 0 1	0	0 0 0 0

Nine's-complement table





電路圖依據此圖完成九補數轉換器，兩數相加為 $9_{10}=(1001)_2$