HW#2 Tianpu Zhao

#1 (1) $f(a,b,c,d) = S(0,5,6,10,11,13) + d(4,8,14)$

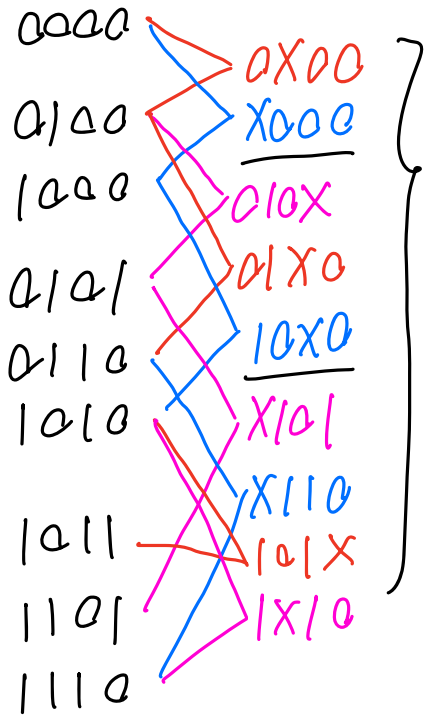
Implication table:

	decimal	binary
S	0	0000
	5	0101
	6	0110
	10	1010
	11	1011
	13	1101
d	4	0100
	8	1000
	14	1110

group by # of 1's →

0000
0100
1000
0101
0110
1010
1011
1101
1110

Find prime implicants:



all are prime implicants

Find unate covering:

Minterms \ Prime implicants	0X00	X000	01X	0X0	10X0	X101	X110	10X	1X10
0000	1	1							
0100	1		1	1					
1000		1			1				
0101			1			1			
0110				1			1		
1010					1			1	1
1011								1	
1101						1			
1110							1		1

○: essential prime implicants & corresponding minterms

↓ eliminate essential prime implicants & corresponding rows

Minterms \ Prime implicants	0X00	X000	01X	0X0	10X0	X110	1X10
0000	1	1					
0100	1		1	1			
1000		1			1		
0110				1		1	
1110						1	1

0X0 column dominates 0X0X, XOOO column dominates 1OXO, XII0 column dominates 1XI0, eliminate dominated columns:

Minterms \ Prime implicants	0X00	XOOO	0X0	XII0
0000	1	1		
0100	1		1	
1000		1		
0110			1	1
1110				1

0110 dominates 1110, 0000 dominates 1000, eliminate 0110, 0000

Minterms \ Prime implicants	0X00	XOOO	0X0	XII0
0100	1		1	
1000		1		
1110				1

X110, X000 are essential prime implicants; either 0X00 or 01X0 can fulfill 0100. Thus the simplified expression is:

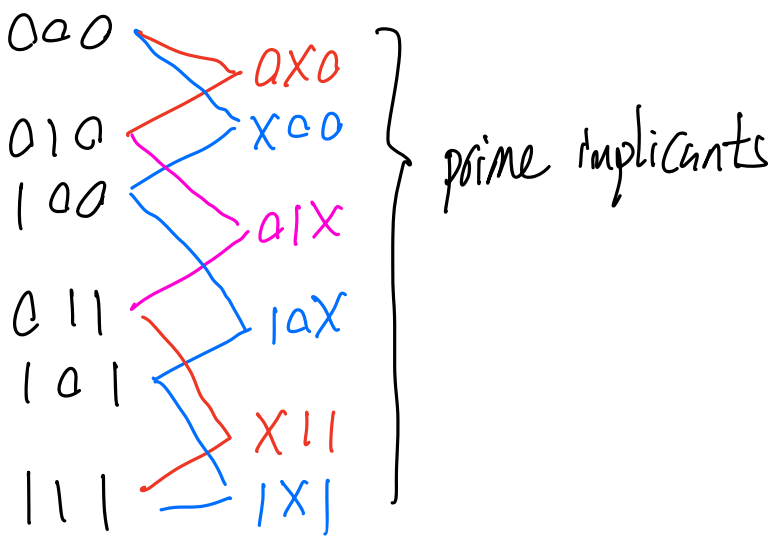
$$f(a,b,c,d) = a'c'd' + b'c'd' + bcd' + bc'd + ab'c$$

Note: 0100 is a don't care minterm. So there is no need for including $a'c'd'$. In fact, following KMap the simplest expression includes exactly the rest 4 terms.

(2) $f(a,b,c) = S(0,2,3,4,5,7)$

decimal	binary		
0	000 ·		00 0
2	0 1 0 ·		0 1 0
3	0 1 1 ·	→ group	1 0 0
4	1 0 0 ·		0 1 1
5	1 0 1 ·		1 0 1
7	1 1 1		1 1 1

prime implicants:



Minterms \ prime implicants	0X0	X00	01X	10X	X11	1X1
000	1	1				
010	1		1			
100		1		1		
011			1		1	
101				1		1
111					1	1

Unate covering: no dominant, perform branch & bound.

Disjoint rows = 3 (pick 111, 100, 010, for example)

⇒ the lower bound $L = 3$.

Pick one path for round (1):

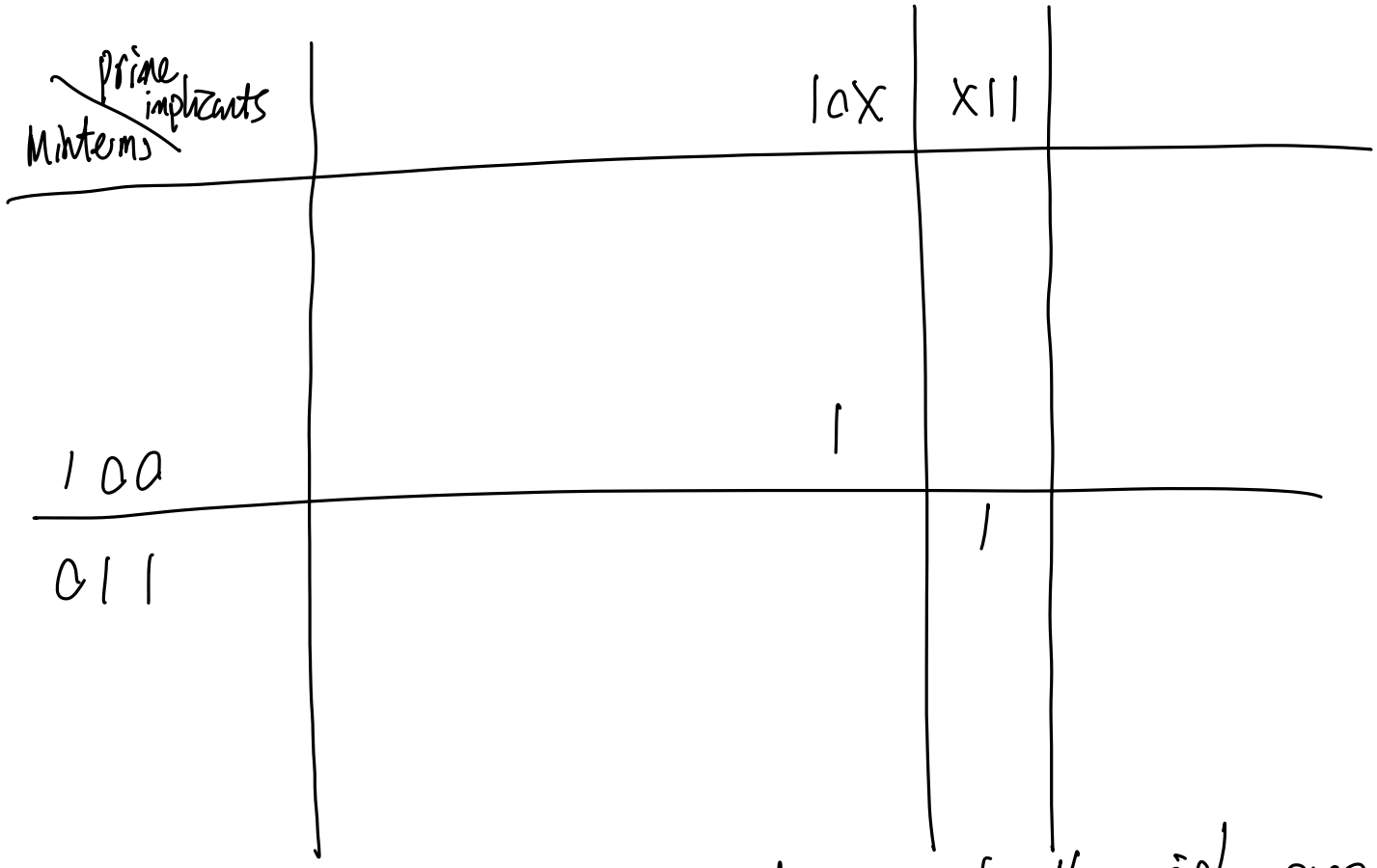
* pick 0X0, erase rows 000, 010

Minterms \ Prime implicants	X00	0IX	1IX	XI1	1XI
100	1		1		
011		1		1	
1 0 1			1		1
111				1	1

1IX dominates X00, erase X00, XI1 dominates 0IX, erase 0IX

Minterms \ Prime implicants	1IX	XI1	1XI
100	1		
011		1	
1 0 1	1		1
111		1	1

101 dominates 100, remove 101, 111 dominates 011, remove 111



Left 1ax and X11 as prime implicants. together with 0x0 forms the expression. The sdution matches the lower bound thus is an optimal solution.

$$f(a,b,c) = a'c' + ab' + bc.$$

#2

Signed = S
Unsigned = U

Decimal = D
Binary = B
Octal = O
Hex = H

	Signed Or Unsigned	Width	Base: Decimal, Binary, Octal, Hexadecimal, etc.	Equivalent Integer Value
8'b00001000	U	8	B	8
16'hABCD	U	16	H	43981
4'sb1110	S	4	B	-2
4'd12	U	4	D	12
4'b1100	U	4	B	12
8'shFF	S	8	H	-1
8'sb00011100	S	8	B	28
12'h2A8	U	12	H	680
6'sb111101	S	6	B	-3
12'o3456	U	12	O	1838

#3

Codes:

```
// Q3_dec_2_to_4_output_pol_ctrl.v
`timescale 1ns/1ps
module dec_2_to_4_output_pol_ctrl (
    input A0, A1, A2,
    output Y0, Y1, Y2, Y3
);
    // inverters and the and gates
    wire d0, d1, d2, d3;
    assign d0 = (~A1 & ~A0);
    assign d1 = (~A1 &  A0);
    assign d2 = ( A1 & ~A0);
    assign d3 = ( A1 &  A0);

    // xors
    assign Y0 = d0 ^ A2;
    assign Y1 = d1 ^ A2;
    assign Y2 = d2 ^ A2;
    assign Y3 = d3 ^ A2;
endmodule
```

Testbench:

```
// Q3_tb_dec_2_to_4.v
`timescale 1ns/1ps

module tb;
  // DUT input signals
  reg  A0, A1, A2;
  // DUT output signals
  wire Y0, Y1, Y2, Y3;
  // index for loop
  integer i;

  // DUT
  dec_2_to_4_output_pol_ctrl dut (
    .A0(A0), .A1(A1), .A2(A2),
     .Y0(Y0), .Y1(Y1), .Y2(Y2), .Y3(Y3)
  );

  // print table
  initial begin
  $display("A2 A1 A0 | Y3 Y2 Y1 Y0");
    $display("----------------------");

    // create all combinations for the input
    for (i = 0; i < 8; i = i + 1)
      begin
         {A2, A1, A0} = i[2:0];
         #1; // wait 1ns
         $display(" %b   %b  %b | %b  %b  %b  %b",
                A2, A1, A0, Y3, Y2, Y1, Y0);
      end
    $display("----------------------");
    $finish;
  end
endmodule
```

Output

```
xcelium>
xcelium> source /vol/cadence2018/XCELIUM1809/tools/xcelium/files/xmsimrc
xcelium> run
A2 A1 A0 | Y3 Y2 Y1 Y0
-----------------------
 0   0  0 |  0  0  0  1
 0   0  1 |  0  0  1  0
 0   1  0 |  0  1  0  0
 0   1  1 |  1  0  0  0
 1   0  0 |  1  1  1  0
 1   0  1 |  1  1  0  1
 1   1  0 |  1  0  1  1
 1   1  1 |  0  1  1  1
-----------------------
Simulation complete via $finish(1) at time 8 NS + 0
./Q3_tb_dec_2_to_4.v:31      $finish;
xcelium>
```

#4

Codes:

// Q4_full_adder.v

```
`timescale 1ns/1ps
module full_adder (
  input  A,
  input  B,
  input  CIN,
  output S,
  output COUT
);
  assign S    = A ^ B ^ CIN;
  assign COUT = (A & B) | (A & CIN) | (B & CIN);
endmodule

module ripple_adder4 (
  input  [3:0] A,
  input  [3:0] B,
  input  CIN,
  output [3:0] S,
  output COUT
);
  wire C1, C2, C3;

  assign S[0] = A[0] ^ B[0] ^ CIN;
  assign C1 = (A[0] & B[0]) | (A[0] & CIN) | (B[0] &
CIN);
  assign S[1] = A[1] ^ B[1] ^ C1;
  assign C2 = (A[1] & B[1]) | (A[1] & C1) | (B[1] &
C1);
  assign S[2] = A[2] ^ B[2] ^ C2;
  assign C3 = (A[2] & B[2]) | (A[2] & C2) | (B[2] &
C2);
```

Testbench

```
// Q4_tb_full_adder.v

`timescale 1ns/1ps
module tb_ripple_adder4;
  reg [3:0] A, B;
  reg CIN;
  wire [3:0] S;
  wire COUT;

  ripple_adder4 dut (.A(A), .B(B), .CIN(CIN), .S(S), .COUT(COUT));

  task run_vec;
    input [3:0] a, b, cin;
    begin
      A = a; B = b; CIN = cin;
      #1;
      $display("A=%0d (%b)  B=%0d (%b)  CIN=%0d -> COUT=%0b, S=%04b, {COUT, S}=%0d",
        a, a, b, b, cin, COUT, S, {COUT, S});
    end
  endtask

  initial begin
    $display("---- 4-bit Ripple Carry Adder Test ----");

    // initialize
    A=0; B=0; CIN=0;
    #1;

    run_vec(4'd1 , 4'd3 , 1'b0);
    run_vec(4'd11, 4'd9 , 1'b0);
    run_vec(4'd7 , 4'd13, 1'b0);
    run_vec(4'd15, 4'd1 , 1'b0);

    $display("All tests completed.");
    $finish;
  end
endmodule
```

Output

```
xcelium>
xcelium> source /vol/cadence2018/XCELIUM1809/tools/xcelium/files/xmsimrc
xcelium> run
---- 4-bit Ripple Carry Adder Test ----
A=1 (0001)  B=3 (0011)  CIN=0 -> COUT=0, S=0100, {COUT, S}=4
A=11 (1011)  B=9 (1001)  CIN=0 -> COUT=1, S=0100, {COUT, S}=20
A=7 (0111)  B=13 (1101)  CIN=0 -> COUT=1, S=0100, {COUT, S}=20
A=15 (1111)  B=1 (0001)  CIN=0 -> COUT=1, S=0000, {COUT, S}=16
All tests completed.
Simulation complete via $finish(1) at time 5 NS + 0
./Q4_tb_full_adder.v:33     $finish;
xcelium>
```