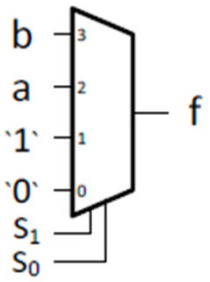
SC1005

Tut 6

Tut 6

Q1 Determine the minimized sum-of-products expression for the 4-1 multiplexer below, and hence sketch the equivalent circuit using just AND and OR gates. Bubble inputs are allowed.



Q2. (a) Draw the circuit represented by the following Verilog module. Label the gates and wires.

```
module whatisit (input a, b, c, d, e,
                 output x, y);
    not  n1  (nb, b);
    not  n2  (ne, e);
    and  a1  (w1, a, b);
    and  a2  (w2, nb, c);
    nor  no1 (w3, d, e);
    nand na1 (w4, w2, w3);
    or   o1  (x, w1, w4);
    and  a3  (y, ne, w1);
endmodule
```

(b) Write down a logic expression for each of the outputs.

Q3. You are required to design a ferry boarding system to direct cars to one of four boarding ramps. The input to the circuit is a 2-bit binary number representing which ramp to use, and a 1-bit signal which is high when all ramps are full. These signals are generated for you (possibly by the human gatekeeper). The circuit has outputs that control four green lights, one above each of the ramps, with only one active at any time. If all ramps are full, no lights should be illuminated indicating that vehicles should not proceed to join any ramp.

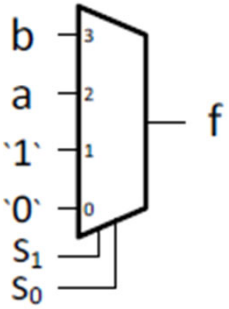
(a) Sketch a block diagram for the circuit using a single 2-4 decoder (with enable). A 2-4 decoder takes a 2-bit binary input and produces a one-hot 4-bit output if enable is TRUE.

(b) Write a Verilog module to implement the boarding system that instantiates a predefined decoder module with the following declaration:

module dec2to4 (**input** [1:0] a, **input** enable, **output** [3:0] one-hot);

Q4. Rewrite the Verilog module of Q2 using a single assign statement for each output.

Q1: Determine min SOP



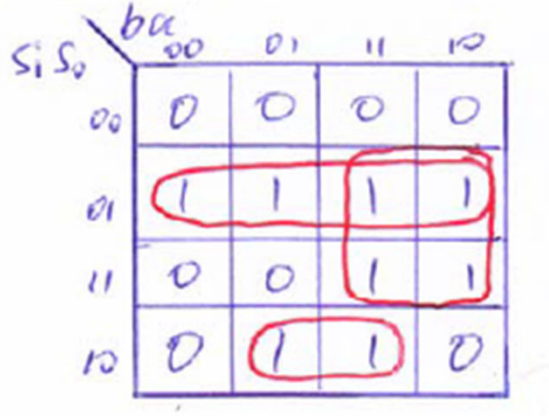
- We can write the expression directly, as f =:

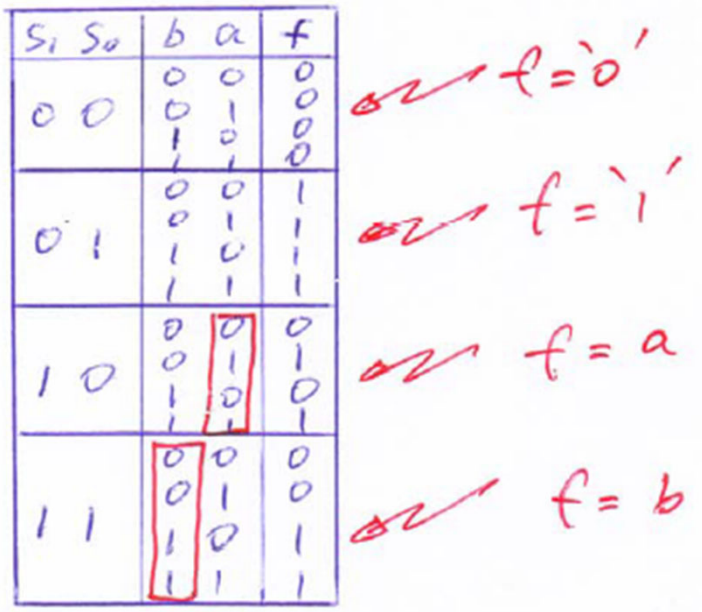
  (S1'.S0').0 + (S1'.S0).1 + (S1.S0').a + (S1.S0).b

  = S1'.S0 + S1.S0'.a + S1.S0.b

  = S1' S0 + S1 S0' a + S1 S0 b

- Or we could use a truth table as:
- Either way, we then get the K-map as:

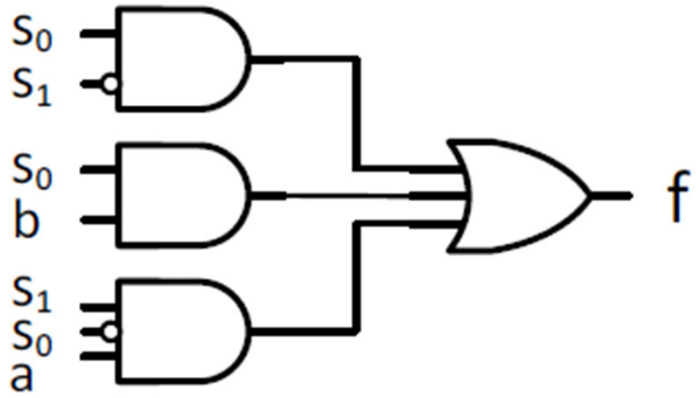




Which gives: f = S1' S0 + S0 b + +S1 S0' a

Q1: Sketch the equivalent circuit

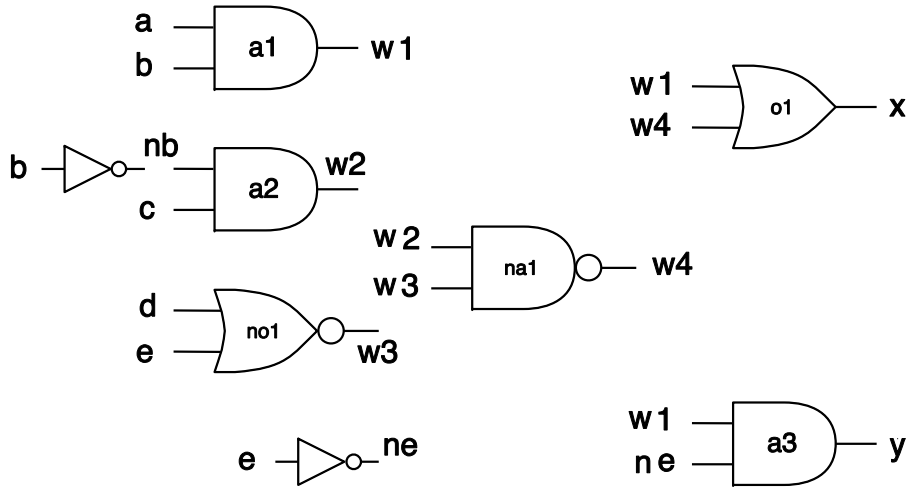
- From the min SOP, f = S1’ S0 + S0 b + S1 S0’ a
- We can directly get the equivalent circuit as:



Q2: Given the Verilog module, draw the circuit

```
module whatisit (input a, b, c, d, e,
                 output x, y);
    not  n1  (nb, b);
    not  n2  (ne, e);
    and  a1  (w1, a, b);
    and  a2  (w2, nb, c);
    nor  no1 (w3, d, e);
    nand na1 (w4, w2, w3);
    or   o1  (x, w1, w4);
    and  a3  (y, ne, w1);
endmodule
```

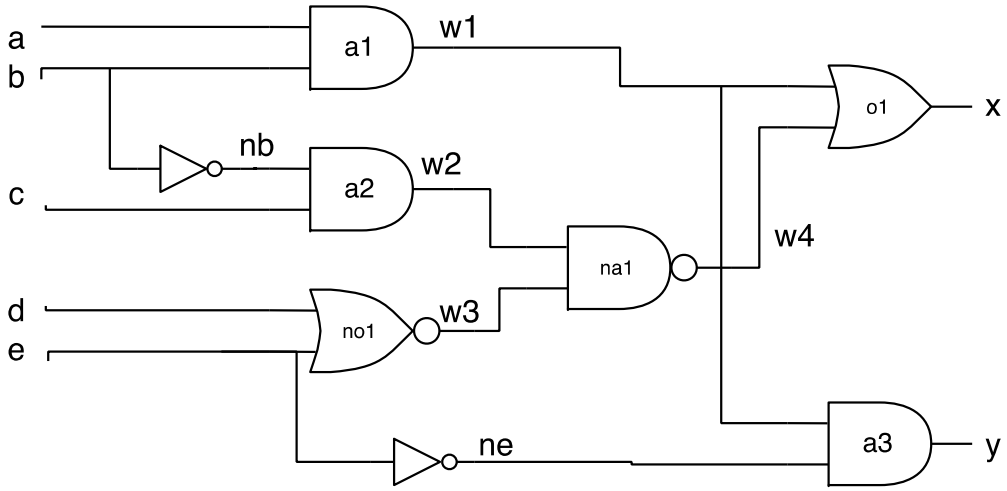
- Just place the gates



Q2: Given the Verilog module, draw the circuit

```
module whatisit (input a, b, c, d, e,
                 output x, y);
    not  n1  (nb, b);
    not  n2  (ne, e);
    and  a1  (w1, a, b);
    and  a2  (w2, nb, c);
    nor  no1 (w3, d, e);
    nand na1 (w4, w2, w3);
    or   o1  (x, w1, w4);
    and  a3  (y, ne, w1);
endmodule
```

- Just place the gates
- and draw the interconnections, as



Q2(b): The logic expression for each output is:

x = ab + ((b'c) (d+e)')'

y = abe'

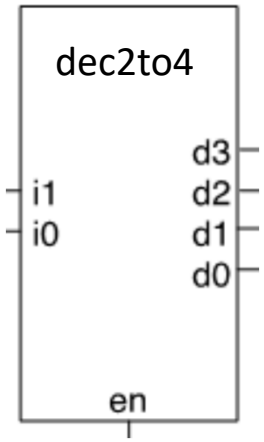
Q3 (a)

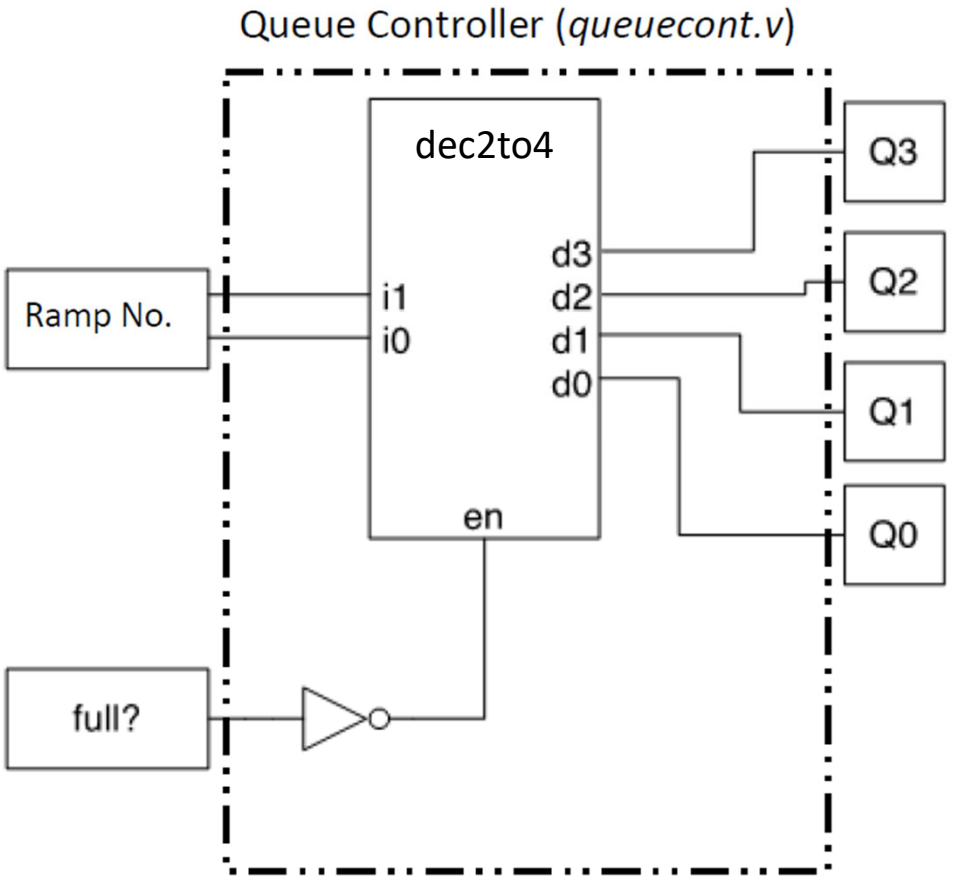
Q3. You are required to design a ferry boarding system to direct cars to one of four boarding ramps. The input to the circuit is a 2-bit binary number representing which ramp to use, and a 1-bit signal which is high when all ramps are full. These signals are generated for you (possibly by the human gatekeeper). The circuit has outputs that control four green lights, one above each of the ramps, with only one active at any time. If all ramps are full, no lights should be illuminated indicating that vehicles should not proceed to join any ramp.

(a) Sketch a block diagram for the circuit using a single 2-4 decoder (with enable). A 2-4 decoder takes a 2-bit binary input and produces a one-hot 4-bit output if enable is TRUE.

(b) Write a Verilog module to implement the boarding system that instantiates a predefined decoder module with the following declaration:

module dec2to4 (**input** [1:0] i, **input** en, **output** [3:0] d);





Note the *full* signal needs to be negated to drive *en*.

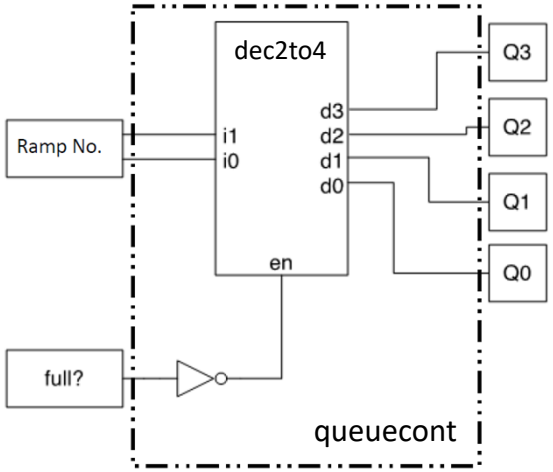
Q3 (b)

Q3. You are required to design a ferry boarding system to direct cars to one of four boarding ramps. The input to the circuit is a 2-bit binary number representing which ramp to use, and a 1-bit signal which is high when all ramps are full. These signals are generated for you (possibly by the human gatekeeper). The circuit has outputs that control four green lights, one above each of the ramps, with only one active at any time. If all ramps are full, no lights should be illuminated indicating that vehicles should not proceed to join any ramp.

(a) Sketch a block diagram for the circuit using a single 2-4 decoder (with enable). A 2-4 decoder takes a 2-bit binary input and produces a one-hot 4-bit output if enable is TRUE.

(b) Write a Verilog module to implement the boarding system that instantiates a predefined decoder module with the following declaration:

module dec2to4 (**input** [1:0] i, **input** en, **output** [3:0] d);




```
module queuecont (input [1:0]  ramp_num,
                  input        full,
                  output [3:0] Q);

   dec2to4 u1 (.i(ramp_num), .en(~full),
               .d(Q));

endmodule
```

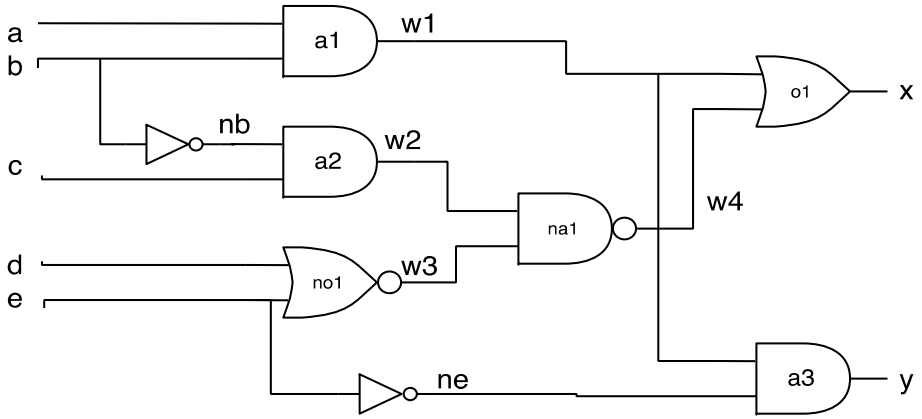
Note the use of the negated *full* signal in the port connection. Thus there is no need for an inverter.

Q4

Q4. Rewrite the Verilog module of Q2 using a single assign statement for each output.

x = ab + ((b'c) (d+e)')'

y = abe'




```
module whatisit2 (input a,b,c,d,e, output x, y);

    assign x = (a & b) | ~((~b & c) & ~(d | e));
    assign y = a & b & ~e;

endmodule
```

Note: The &, |, and ~ operators are bitwise, whereas &&, ||, ! are logical (they evaluate to T or F). They are interchangeable for 1-bit signals.

Also note the operator precedence (~ & | ^ ~^)

So, can we remove any of the brackets in X?

```
assign x = a & b | ~(~b & c & ~(d | e));
```