

Contact Information

Douglas Maskell

Email: asdouglas@ntu.edu.sg

Office: N4-02C-98



SC1005

Digital Logic

Recap and Discussion

Lecture 13

Combinational Circuits

Lecture 14

Introduction to Verilog

Summary of Lecture 13

- Combinational Design Process for Simple Circuits
- Combinational Circuits
 - Seven-Segment Decoder
 - Decoder (One-Hot)
 - Multiplexer
- Timing Diagrams

Summary of Lecture 14

- Introduction to Verilog
 - Module Declaration
 - Instantiating Gate-Level Primitives
 - Module Instantiation

- Combinational Circuit

- Outputs depend solely on the **present combination** of the input values



- Combinational Design Process

- **Step 1: Capture the function**
 - Use truth-table or equations, depending on what makes sense for the application
- **Step 2: Convert to a circuit**

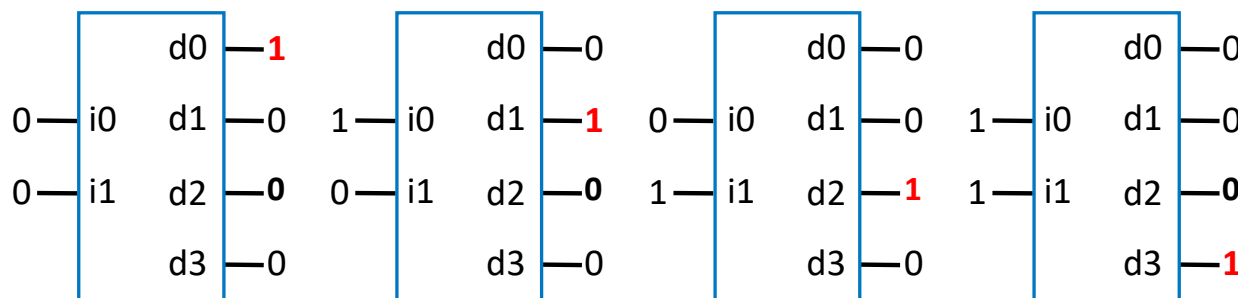
This is what you have been doing in the 1st half of the course

Decoder

- A decoder takes a binary input and produces a corresponding **one-hot output**
 - Only one bit of the output is high (hence one-hot), its position corresponds to the input value
 - Output width is always 2 to the power of the input width
 - N-input Decoder: 2^N outputs

Input	Output
00 →	0001
01 →	0010
10 →	0100
11 →	1000

Source: Frank Vahid, "Digital Design", Wiley, 2nd Edition, pg. 84

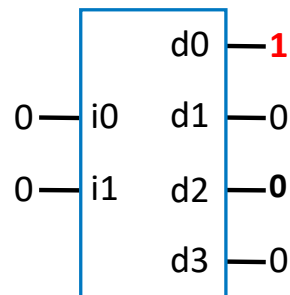


Example: A two-input decoder will have four outputs (2-4 Decoder)

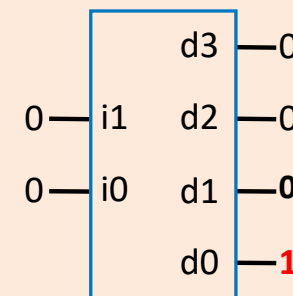
Decoder

- A decoder takes a binary input and produces a corresponding **one-hot output**
 - Only one bit of the output is high (hence one-hot), its position corresponds to the input value
 - Output width is always 2 to the power of the input width
 - N-input Decoder: 2^N outputs

<u>Input</u>	<u>Output</u>
00 →	0001
01 →	0010
10 →	0100
11 →	1000



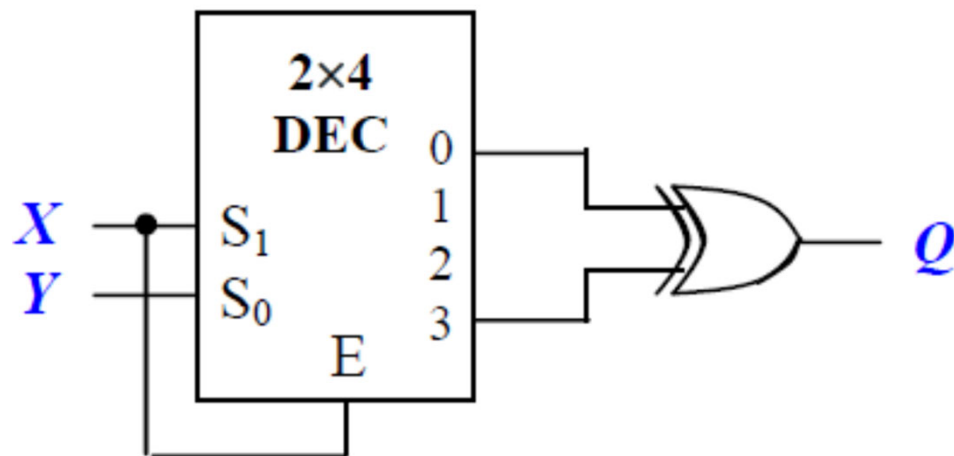
Note these two circuits are identical



Example: A two-input decoder will have four outputs (2-4 Decoder)

Exercise 1

- What is the Boolean expression of Q?



- A. 1
- B. X
- C. $X.Y$
- D. $X' + Y'$
- E. $X'Y + X.Y'$

Ans: C

Consider just the decoder (and ignore enable):

$XY=00 \rightarrow \text{out0} = X' Y'$;

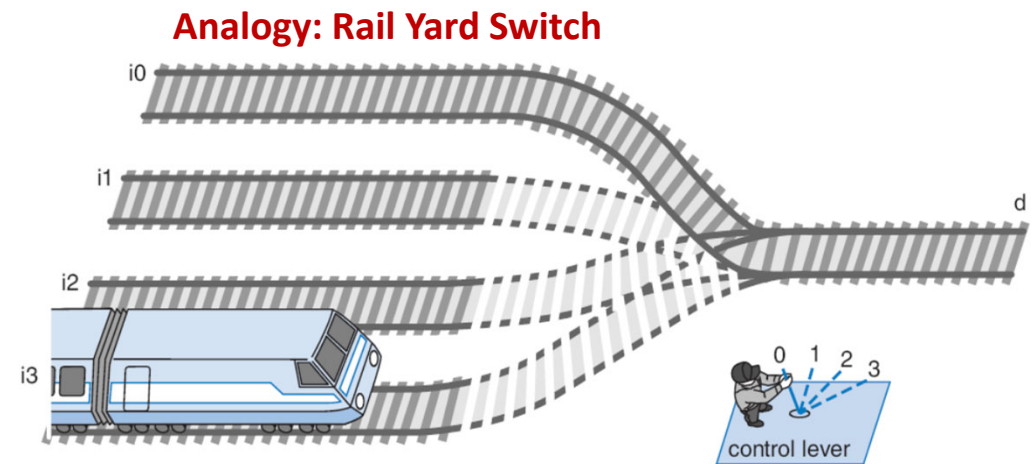
$XY=11 \rightarrow \text{out3} = X Y$;

However, when enable is considered, $\text{out0} = 0$ always.

So, $Q = 0 \text{ XOR } (X Y) = X Y$;

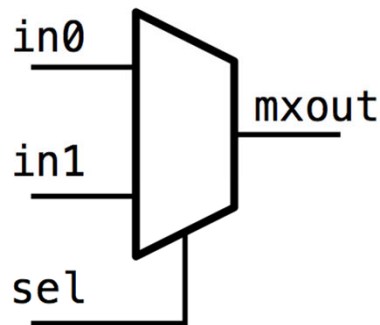
Multiplexer (Mux)

- Consists of multiple inputs, and a single output
- A select input determines which input should be connected to the output
- 2 input multiplexer (mux) needs a 1-bit select input
- 4 input mux requires a 2-bit select input
- An **n** input mux requires a **$\log_2(n)$** -bit select

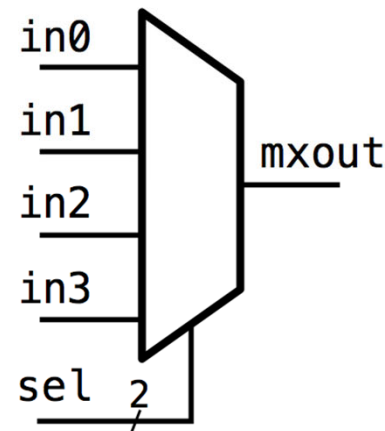


Source: Frank Vahid, "Digital Design", Wiley, 2nd Edition, pg. 87

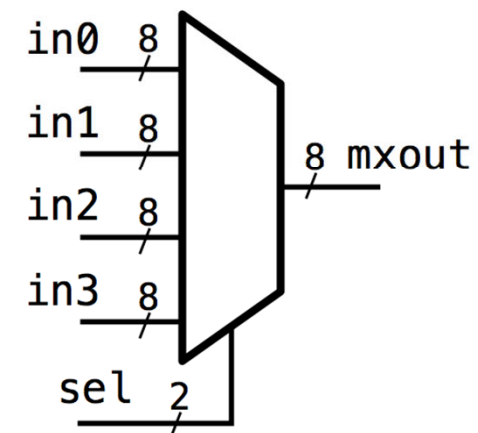
1-bit 2x1 mux



1-bit 4x1 mux

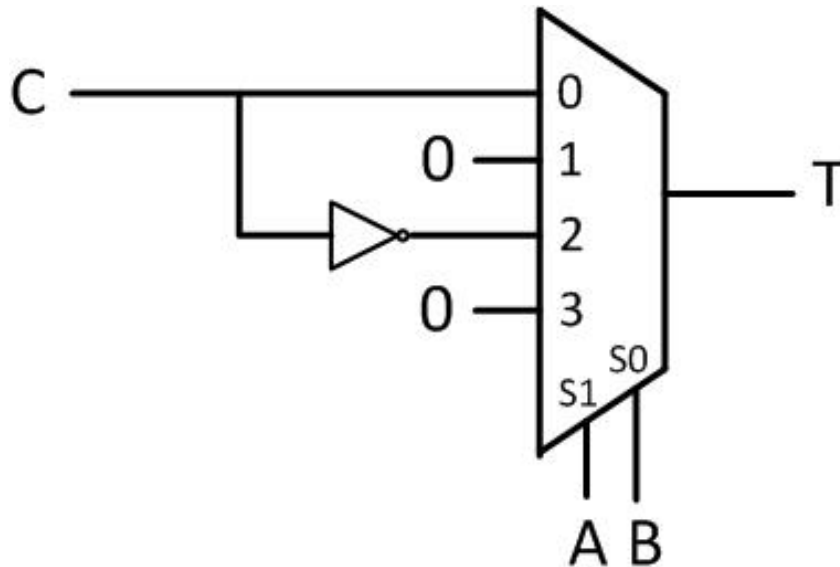


8-bit 4x1 mux



Exercise 2

- Find the SOP expression for function $T(A,B,C)$



- A. $A'.B'.C + A.B'.C'$
- B. $A'.B'.C' + A.B'.C'$
- C. $A'.B'.C + A.B'.C$
- D. $A'.B'.C' + A.B'.C$
- E. None of the above

Ans:

The 0th input (C) will be transferred to T when $A = B = 0$ ($A' B'$). Hence, $T = A' B' C$

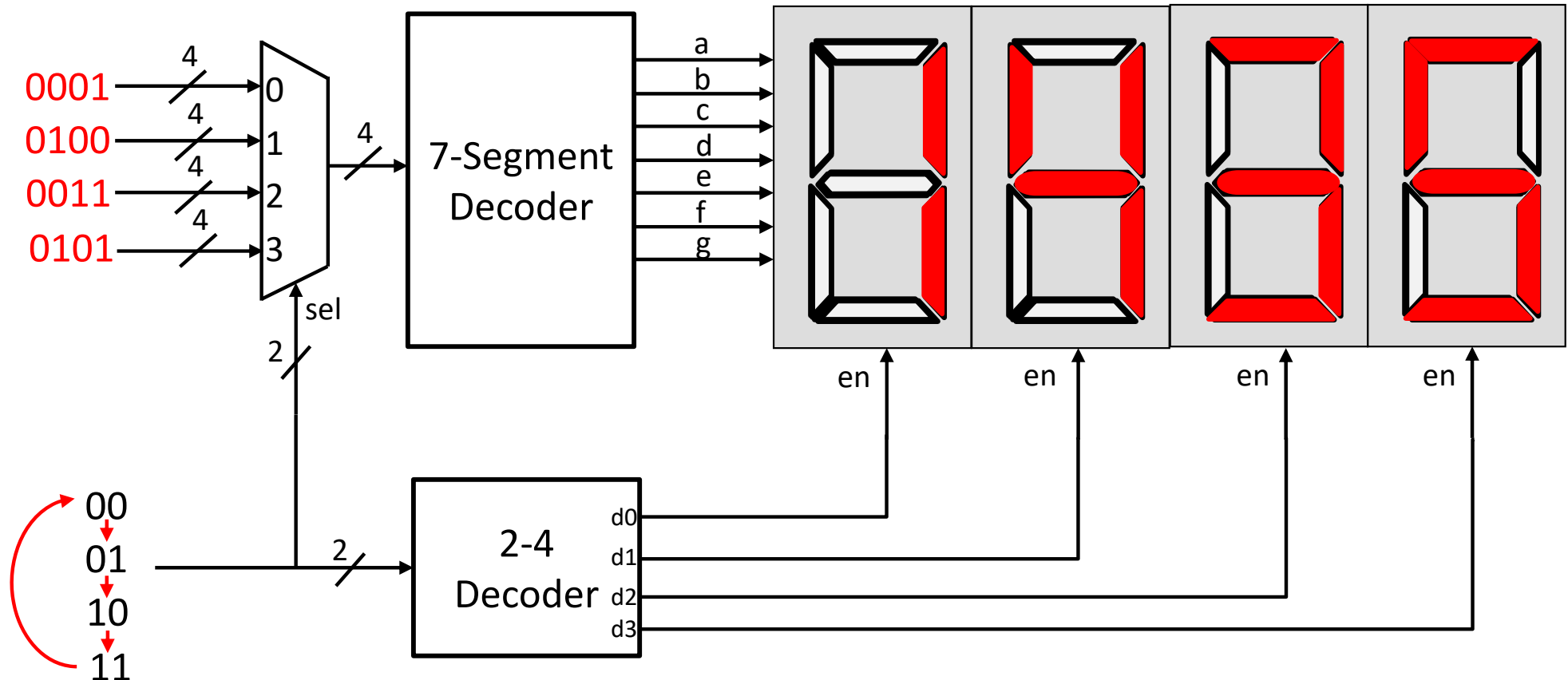
The 2nd input (C') will be transferred to T when $A = 1$ and $B = 0$ ($A B'$). Hence, $T = A B' C'$

So, $T = A' B' C + A B' C'$ (Answer A)

DESIGNING A DIGITAL CLOCK DISPLAY

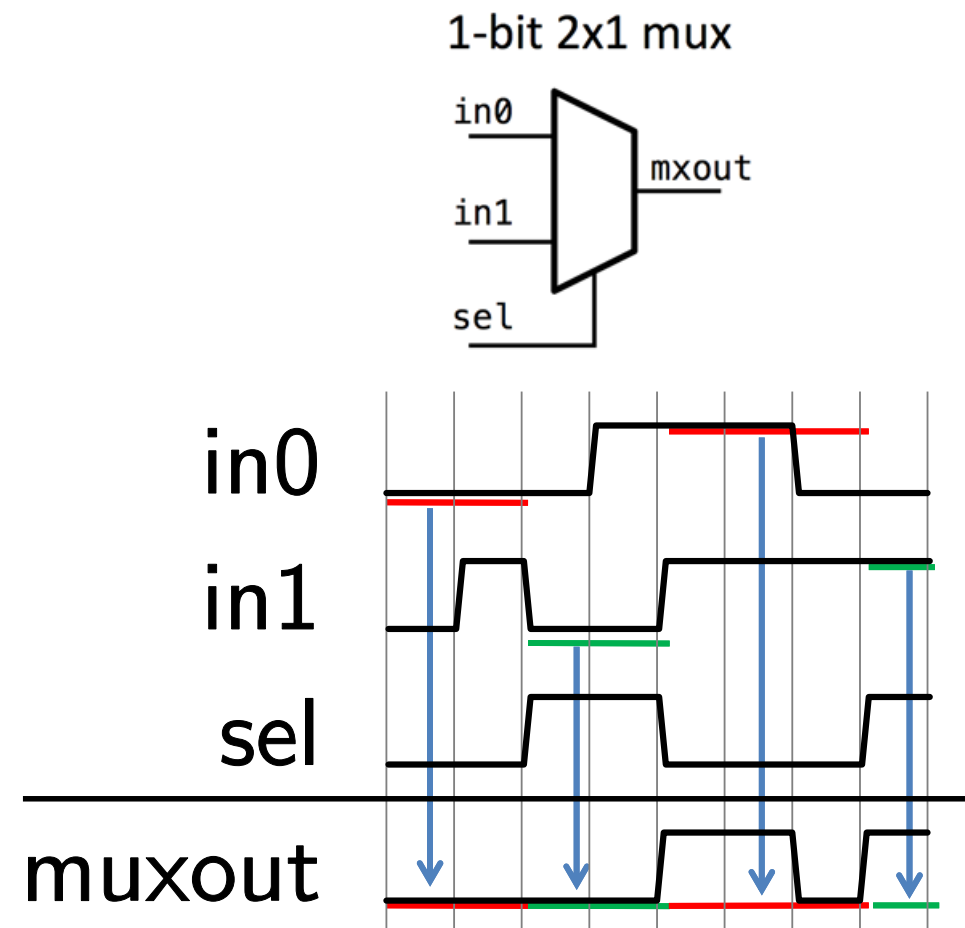
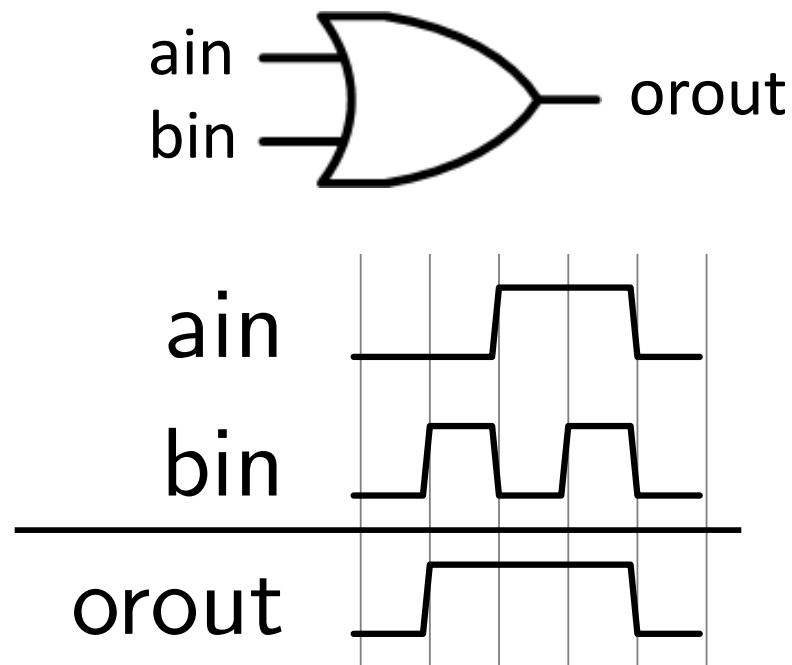


- Four seven segment displays
- One seven segment decoder
- One 4-bit 4x1 Multiplexer
- One 2-4 Decoder



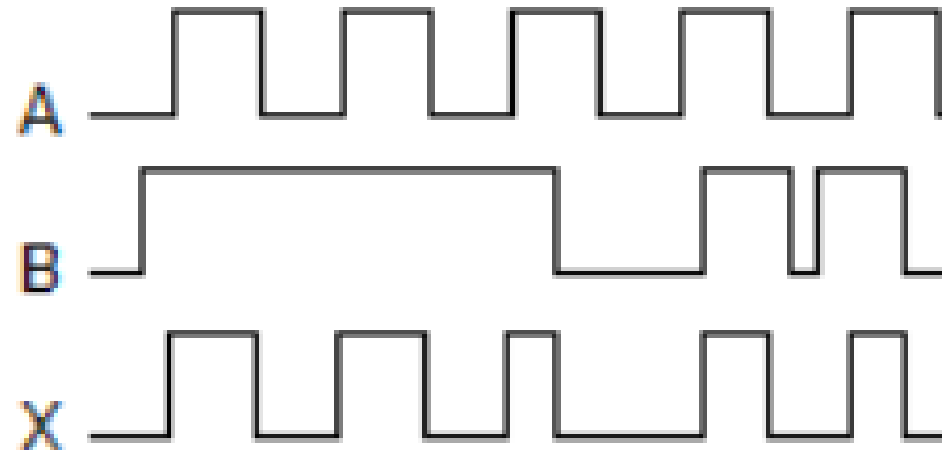
Timing Diagrams

- Timing diagrams show the behaviour of a circuit with **progression of time**
- We can show the timing diagram for any combinational circuit



Exercise 4

- The following waveform for inputs A,B and output X is for:



- A. 2-input AND gate
- B. 2-input OR gate
- C. Exclusive-OR gate
- D. None of the above

Ans:

X is high when both A and B are high.

Hence, $X = A.B$ (Answer A)

So far, you have covered...

Adding two signed
binary numbers

$$\begin{array}{r} +6 \\ + -3 \\ \hline +3 \end{array} \Rightarrow \begin{array}{r} 0110 \\ + 1101 \\ \hline 0011 \end{array}$$



X	Y	C _i	C _o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth table for
Full Adder
(1-bit)

Minterm (or Maxterm)
expression

$$C_o = \sum (3, 5, 6, 7)$$

$$S = \sum (1, 2, 4, 7)$$

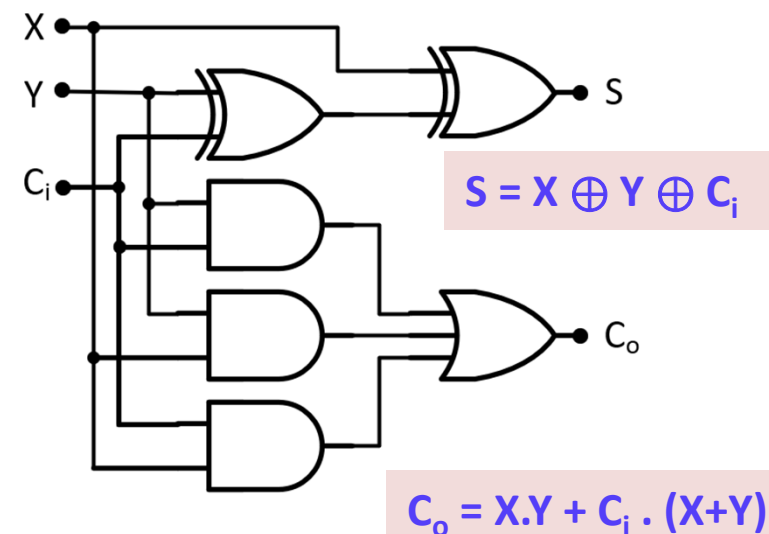


$$S = X \oplus Y \oplus C_i$$

$$C_o = X.Y + X.C_i + Y.C_i$$

Minimize using Boolean
Algebra or K-Maps

Implementation of Full
Adder using basic logic gates



- Number Systems and Digital Arithmetic
- Basic logic gates and Boolean Algebra
- Truth Tables and K-Map
- Gate level combinational circuits

So far, you have covered...

Adding two signed
binary numbers

$$\begin{array}{r} +6 \\ + -3 \\ \hline +3 \end{array} \Rightarrow \begin{array}{r} 0110 \\ + 1101 \\ \hline 0011 \end{array}$$

This is everything we need to start building digital circuits.
So, let's look at how modern digital circuits are designed.
By using Hardware Description Languages (HDLs) for digital design.

The HDL we will use is Verilog.



X	Y	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth table for
Full Adder
(1-bit)

Minterm (or Maxterm)
expression

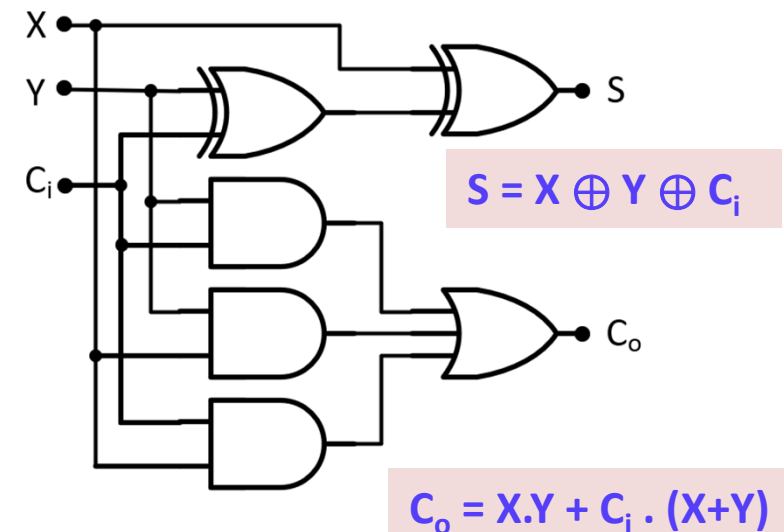
$$C_o = \sum (3, 5, 6, 7)$$
$$S = \sum (1, 2, 4, 7)$$



$$S = X \oplus Y \oplus C_i$$
$$C_o = X.Y + X.C_i + Y.C_i$$

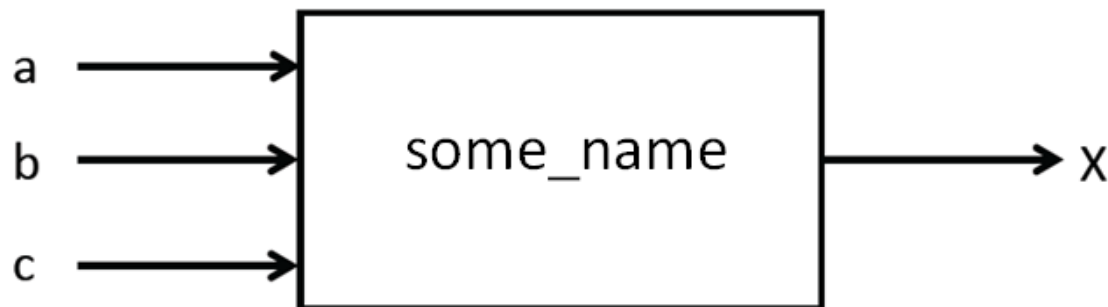
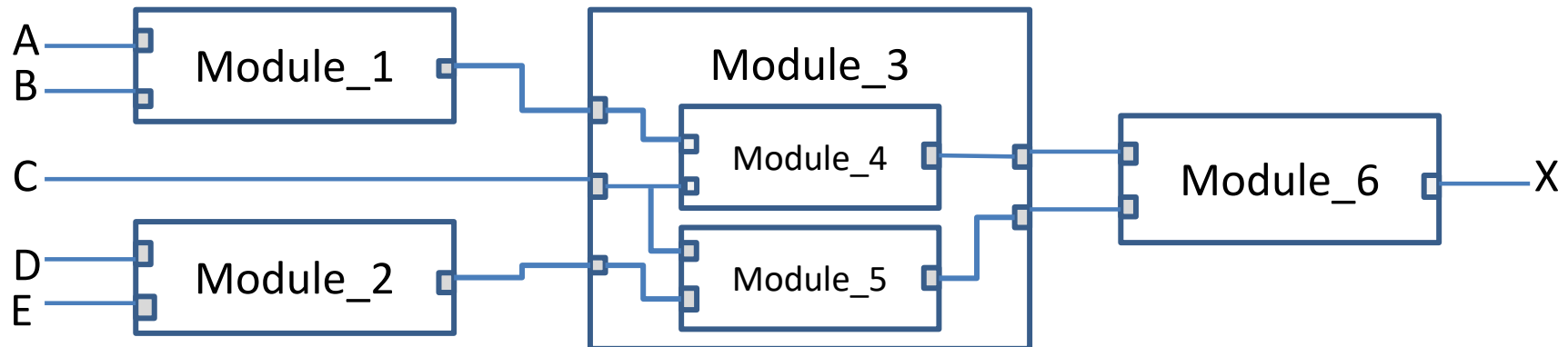
Minimize using Boolean
Algebra or K-Maps

Implementation of Full
Adder using basic logic gates



Verilog Modules

- In Verilog, designs are broken down into **modules**
- At each level in the hierarchy, a module instance is treated as a “black-box” – the internals are unknown
 - A module name cannot start with a number (a letter or underscore is OK). It cannot contain reserved symbols such as * or /, etc.
 - Verilog is case sensitive



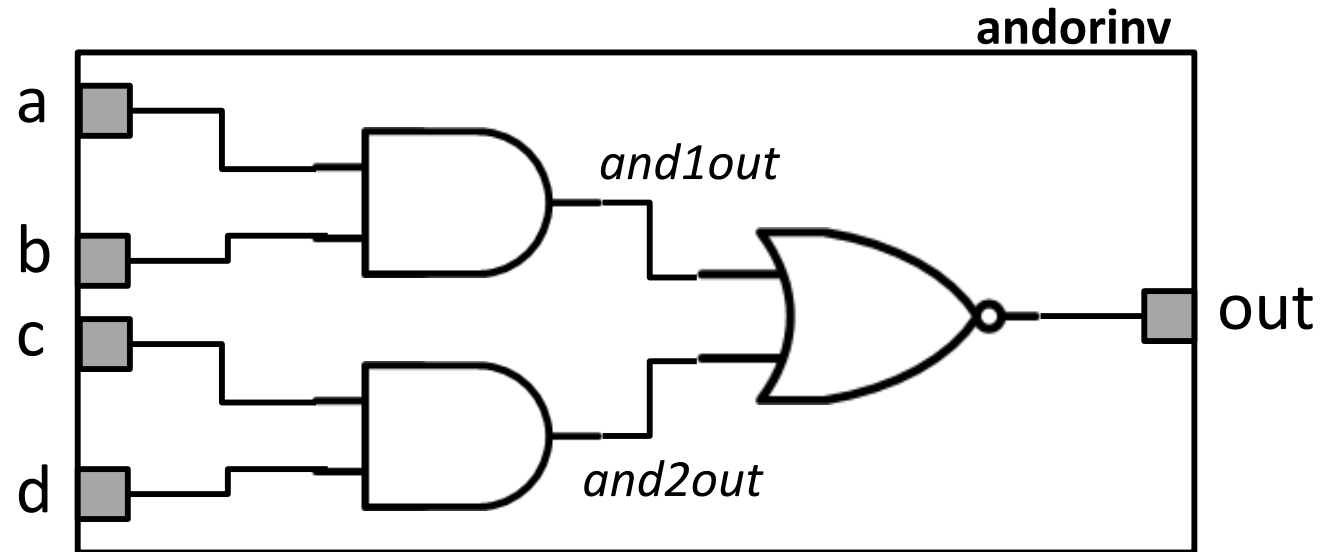
```
module some_name (
    input a, b, c,
    output X);
    // Describe your circuit here
endmodule
```

Verilog Module with Gate Level Primitives

- **Structural design** approach describes a circuit by its internal structure

Boolean equation:

$$\text{out} = ((a \bullet b) + (c \bullet d))'$$

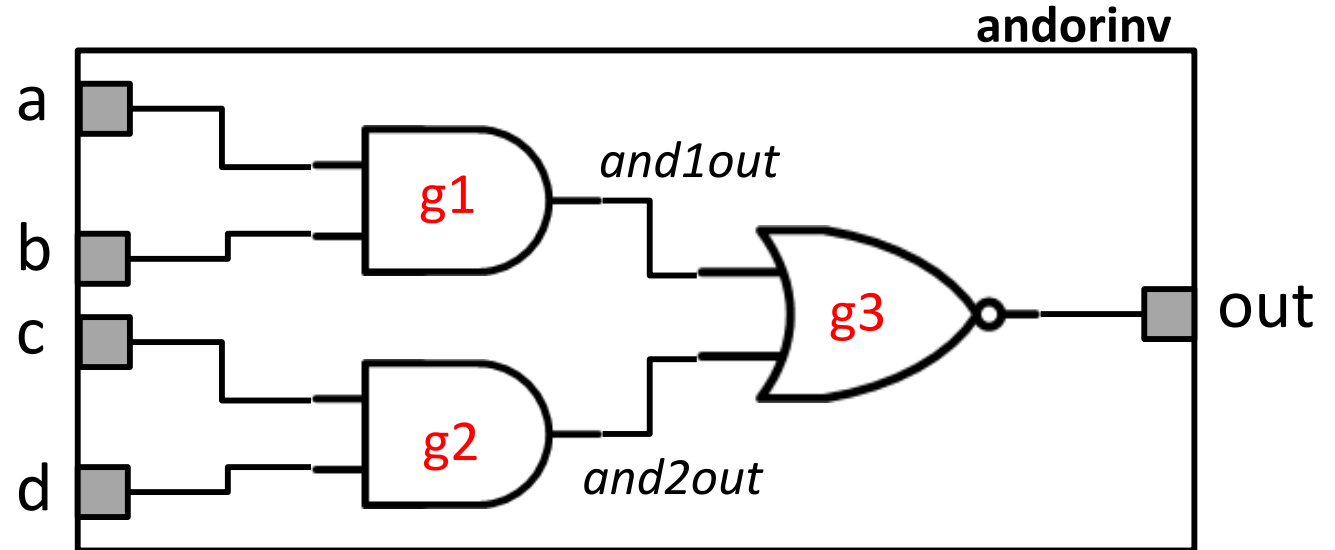


```
module andorinv (input a, b, c, d,  
                 output out);  
  
    wire and1out, and2out;  
  
    and (and1out, a, b);  
    and (and2out, c, d);  
    nor (out, and1out, and2out);  
  
endmodule
```


Verilog Module with Gate Level Primitives

- **Structural design** approach describes a circuit by its internal structure (Drawing the circuit using codes)

Boolean equation:
$$\text{out} = ((a \bullet b) + (c \bullet d))'$$

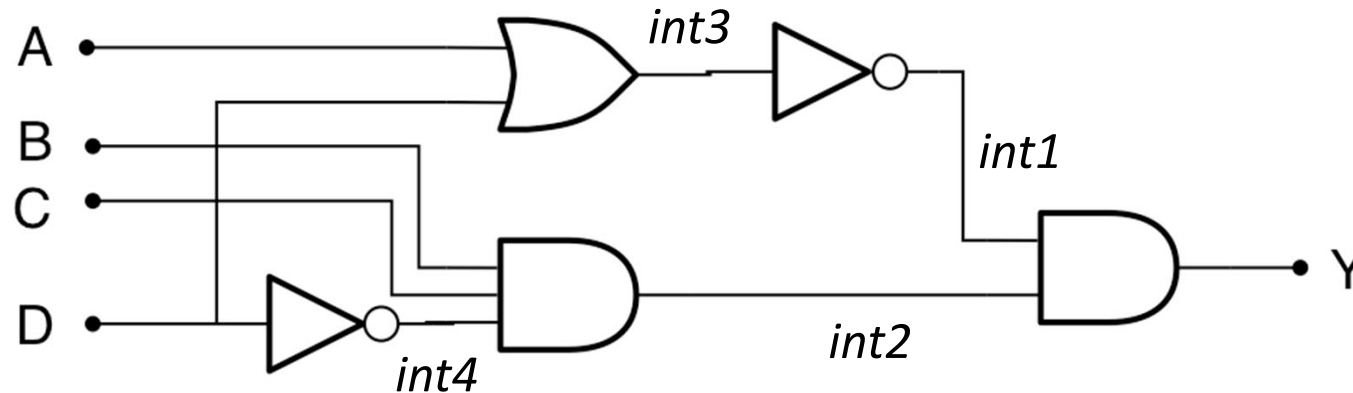


Gate
Identifiers

```
module andorinv (input a, b, c, d,
                  output out);
    wire and1out, and2out;
    and g1 (and1out, a, b);
    and g2 (and2out, c, d);
    nor g3 (out, and1out, and2out);
endmodule
```

Lecture 14 (Task 1)

- Implement a Verilog module using structural gate-level primitives for the following circuit:



```
module simple_circ (input A, B, C, D,
                    output Y);

    wire int1, int2, int3, int4;

    not (int4, D);
    or (int3, A, D);
    and (int2, B, C, int4);
    not (int1, int3);
    and (Y, int1, int2);
endmodule
```

```
module simple_circ (input A, B, C, D,
                    output Y);

    wire int1, int2, int3, int4;

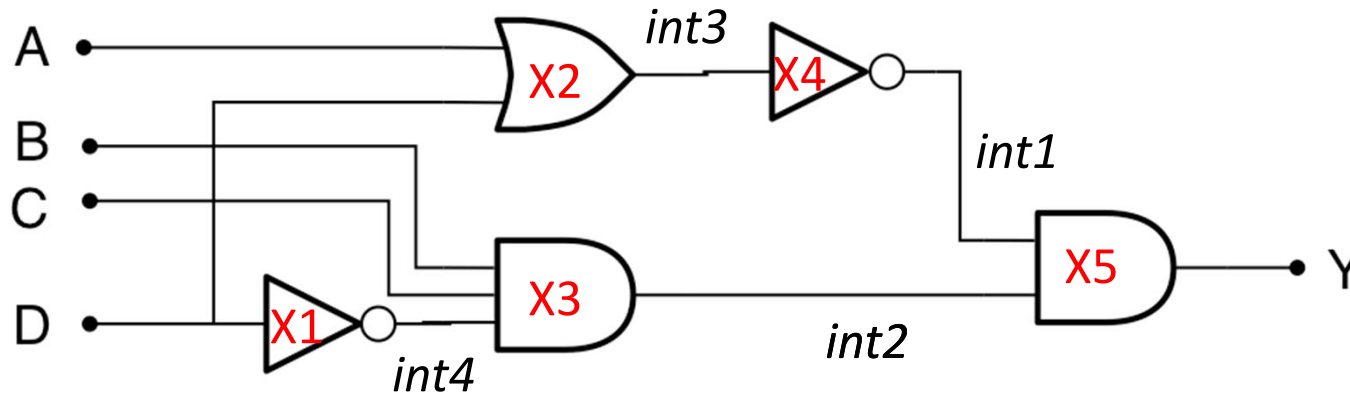
    not (int1, int3);
    not (int4, D);
    and (int2, B, C, int4);
    and (Y, int1, int2);
    or (int3, A, D);
endmodule
```

Both will produce the same circuit.

The order is not important as each is executed concurrently

Lecture 14 (Task 1)

Some Comments: No need to declare **single bit** wires.
Gate identifiers are not needed for gate level primitives, but good to use them.



```
module simple_circ (input A, B, C, D,
                    output Y);

    wire int1, int2, int3, int4;

    not (int4, D);
    or  (int3, A, D);
    and (int2, B, C, int4);
    not (int1, int3);
    and (Y, int1, int2);
endmodule
```



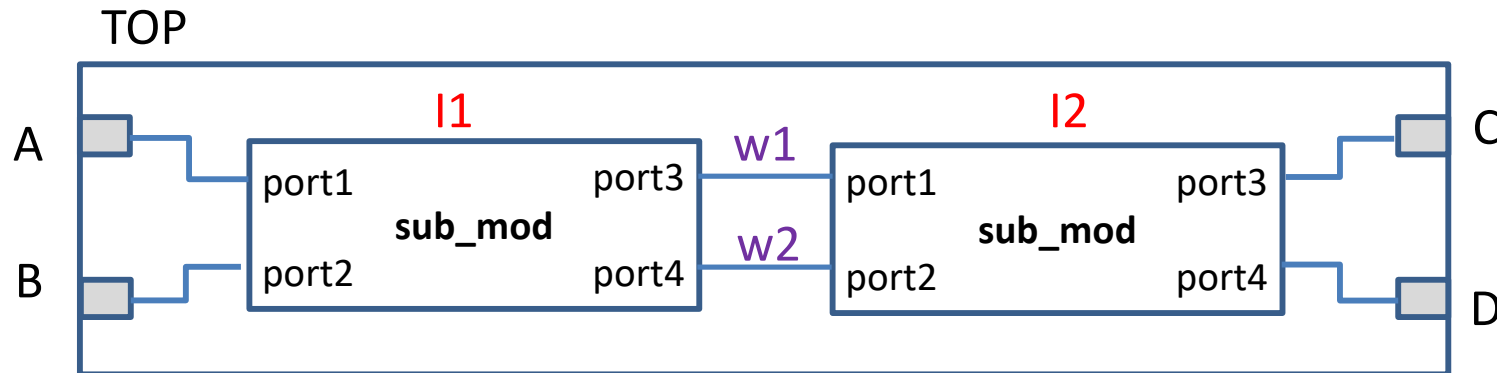
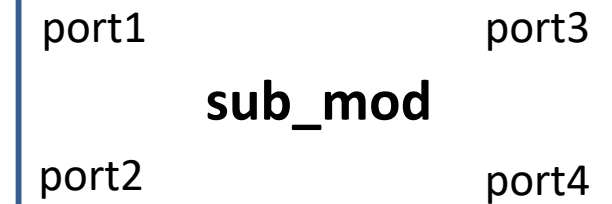
```
module simple_circ (input A, B, C, D,
                    output Y);

    not X1 (int4, D);
    or  X2 (int3, A, D);
    and X3 (int2, B, C, int4);
    not X4 (int1, int3);
    and X5 (Y, int1, int2);
endmodule
```

Identifiers need to be unique within a module

Module Instantiation

```
module sub_mod (input port1, port2,  
                output port3, port4);  
  
    ...  
endmodule
```



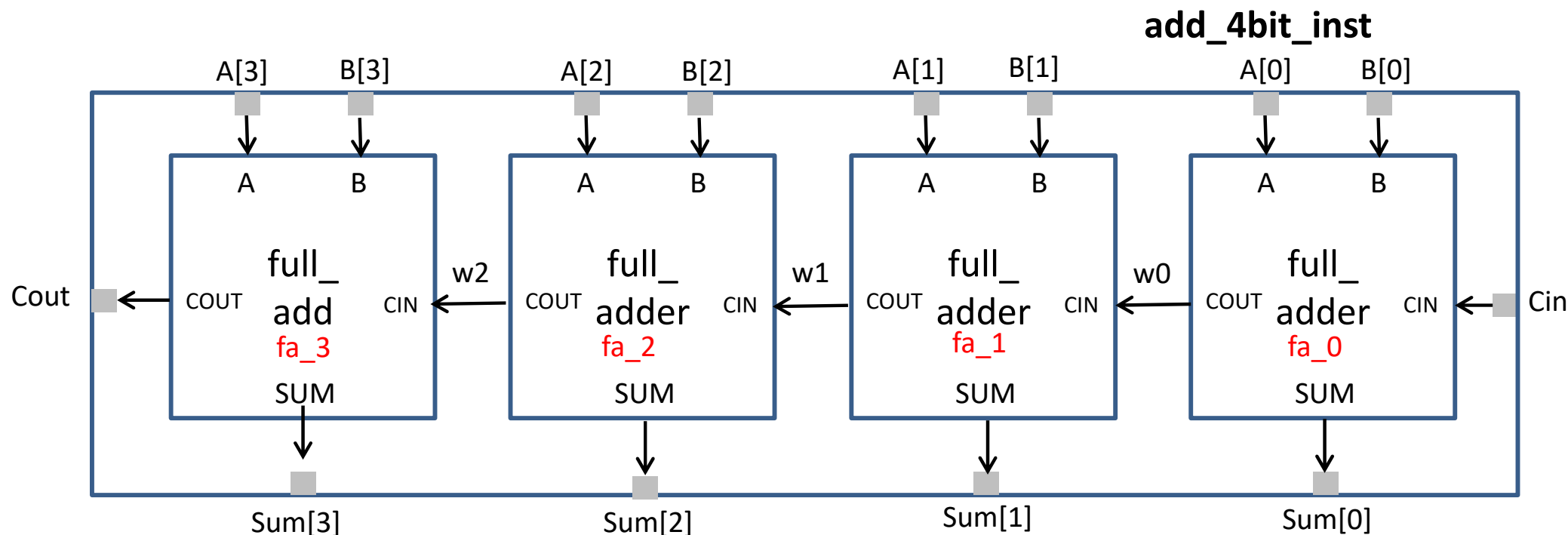
Ordered Instantiation

```
module TOP (input A, B,  
            output C, D);  
    wire w1, w2;  
  
    sub_mod I1 (A, B, w1, w2);  
    sub_mod I2 (w1, w2, C, D);  
  
endmodule
```

Named Instantiation

```
module TOP (input A, B,  
            output C, D);  
    wire w1, w2;  
  
    sub_mod I1 (.port1(A), .port2(B),  
                .port3(w1), .port4(w2));  
    sub_mod I2 (.port1(w1), .port2(w2),  
                .port3(C), .port4(D));  
  
endmodule
```

Instantiation in Verilog (4-bit Adder)



```
module add_4bit_inst (input  [3:0] A, B, input  Cin
                     output [3:0] Sum, output Cout);
```

```
    wire w0, w1, w2;
```

Do we really need to declare the wires?

```
    full_add fa_0 (.A(A[0]), .B(B[0]), .CIN(Cin), .SUM(Sum[0]), .COUT(w0));
    full_add fa_1 (.A(A[1]), .B(B[1]), .CIN(w0), .SUM(Sum[1]), .COUT(w1));
    full_add fa_2 (.A(A[2]), .B(B[2]), .CIN(w1), .SUM(Sum[2]), .COUT(w2));
    full_add fa_3 (.A(A[3]), .B(B[3]), .CIN(w2), .SUM(Sum[3]), .COUT(Cout));
```

```
endmodule
```

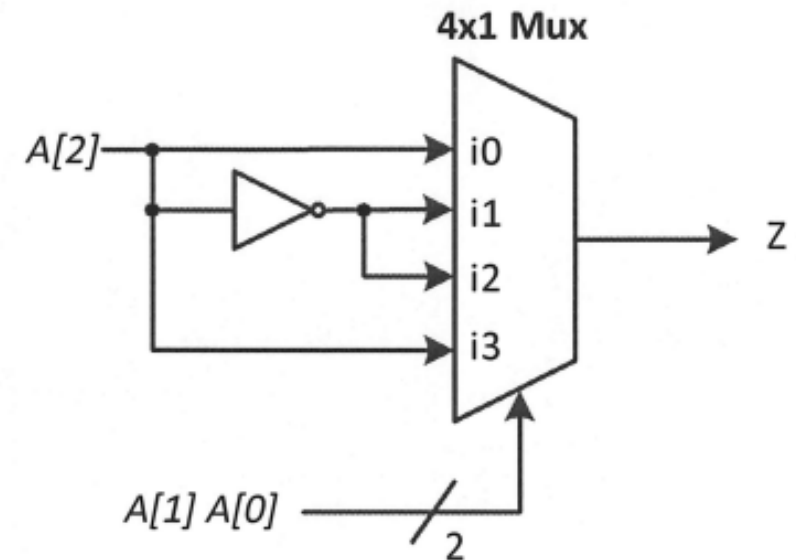
Selected Past Exam Questions

CE/CZ1105 2020-2021 Semester 2 (Apr/May 2021)

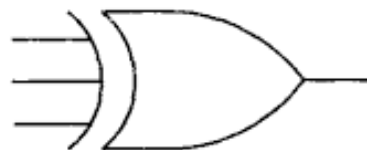
Q3 (b) (iii) Draw the truth table of the combinational circuit (below). What is the functionality of the combinational circuit? Redraw the combinational circuit using a minimal number of logic gates.

ANS: TT is:

<u>A[2:0]</u>	Z	Z
000	A2	0
001	A2'	1
010	A2'	1
011	A2	0
100	A2	1
101	A2'	0
110	A2'	0
111	A2	1



$$Z = A[1] \oplus A[2] \oplus A[3]$$



Selected Past Exam Questions

CE/CZ1005 2018-2019 Semester 2 (Apr/May 2019)

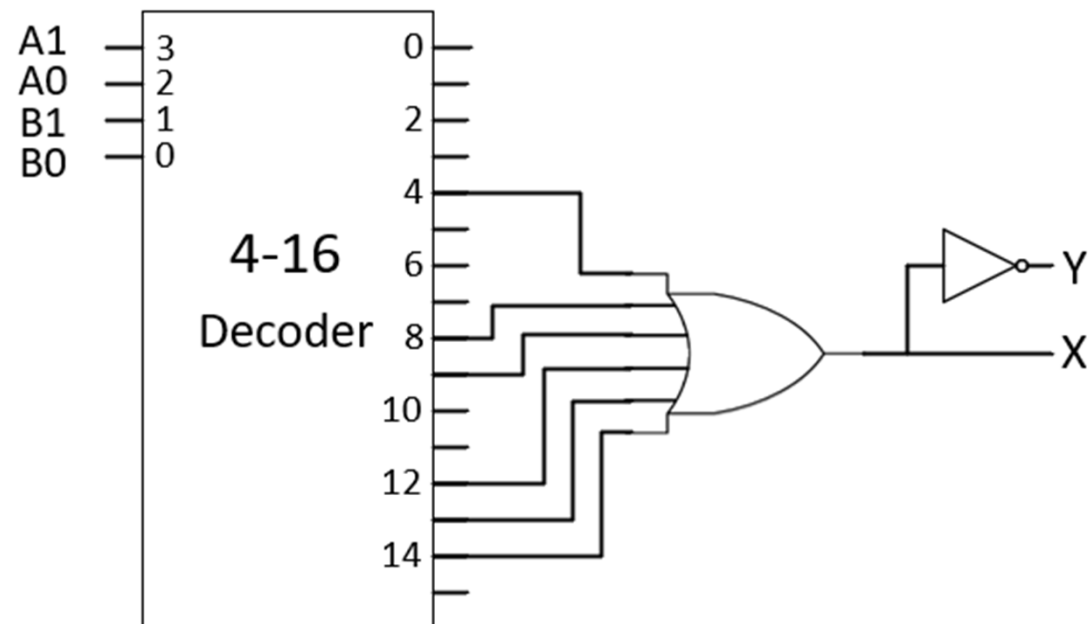
- Q3 (a) Design a 2-bit comparator circuit that takes two inputs, $A[1:0]$ and $B[1:0]$. The outputs are X and Y , where $X=1$ if $A > B$ and $Y=1$ if $A \leq B$.
- (i) Draw the truth table (TT) of the comparator circuit
 - (ii) Draw the circuit diagram (based on the TT). You must use a 4-16 decoder.

ANS:

A1	A0	B1	B0	X	Y
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	1

Firstly, assume unsigned numbers (easier)

Also, Note that $Y = X'$



Selected Past Exam Questions

CE/CZ1005 2018-2019 Semester 1 (Nov/Dec 2018)

Q3 (a) Given the Boolean function $F(a,b,c) = a'b'c' + ab'c + a'bc + abc'$

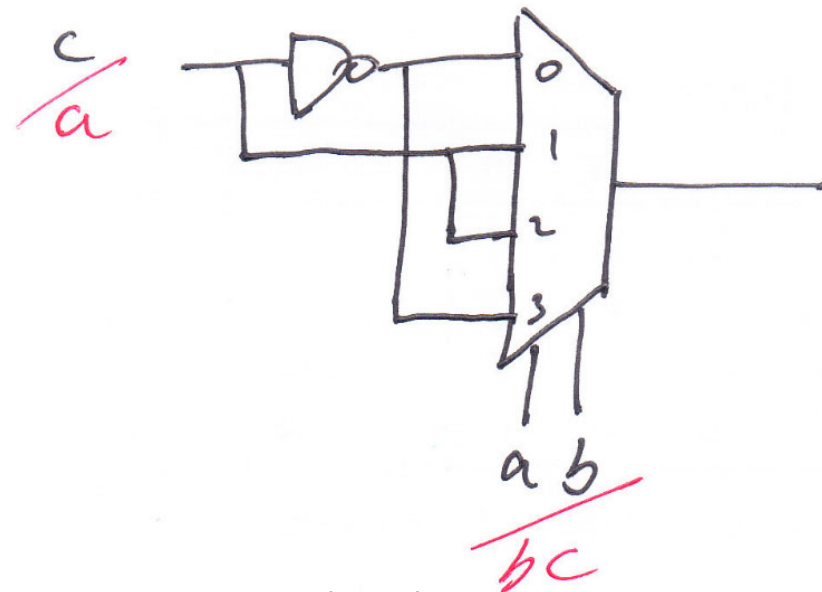
(i) Implement F using a single 4x1 multiplexer and a single inverter.

ANS: Just looking at a and b we see that we have every combination {00, 10, 01, 11}

So, use a and b as s_1 and s_0 , respectively.

Then input c into the appropriate mux input.

That is, c' into 0 and 3; and c into 1 and 2



Note: Could also input b and c into selector inputs (red).

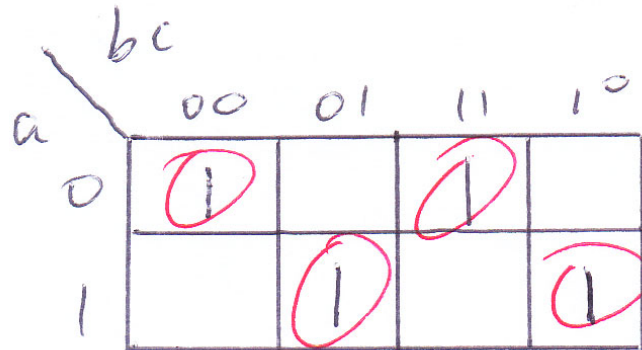
Selected Past Exam Questions

CE/CZ1005 2018-2019 Semester 1 (Nov/Dec 2018)

Q3 (a) Given the Boolean function $F(a,b,c) = a'b'c' + ab'c + a'bc + abc'$

(ii) Determine the minimum sum-of-products expression for F .

ANS: use a K-map, as:



	bc			
	00	01	11	10
a 0	1		1	
1		1		1

So, No change. The existing $F(a, b, c)$ is already a minimum sum-of-products expression .

Selected Past Exam Questions

CE/CZ1005 2017-2018 Semester 2 (Apr/May 2018)

Q3(b) Given the sum-of-minterm expression $F(x, y, z) = \sum m(1, 6)$

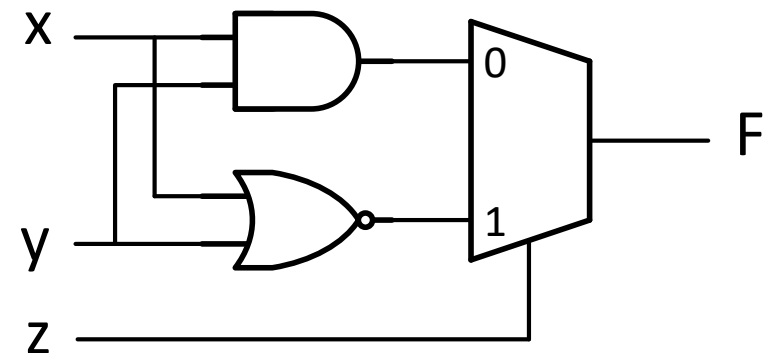
- (i) Draw the schematic diagram showing the implementation of F using one 2x1 multiplexer, one AND gate and one NOR gate.

(5 marks)

ANS: Firstly, $F = x'y'z + xyz'$
$$= (x'y')z + (xy)z' = (x+y)'z + (xy)z'$$

This is a NOR gate & an AND gate, which is selected based on z

Note that you have a xy term which is a AND gate, and a $x'y'$ term which is a NOR gate (use DeMorgans), so use the z term for the multiplexer.



Selected Past Exam Questions

CE/CZ1005 2017-2018 Semester 1 (Nov/Dec 2017)

Q3(b) Draw the schematic diagram of an implementation of a Full-Adder circuit, using one 3x8 decoder and two OR gates. The sum-of-minterm expressions for the output Sum and Carry are given by:

$$\text{Sum}(x, y, z) = \sum m(1, 2, 4, 7)$$

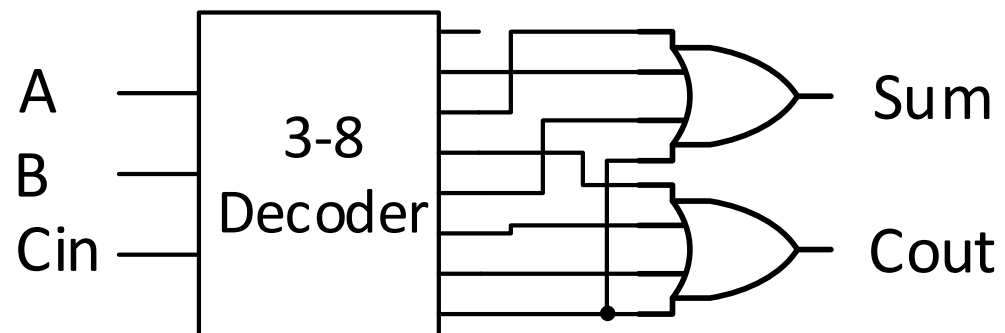
$$\text{Carry}(x, y, z) = \sum m(3, 5, 6, 7)$$

(7 marks)

ANS: The 3-8 Decoder has the following truth table

Input			Output							
2	1	0	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

So just OR the corresponding minterms

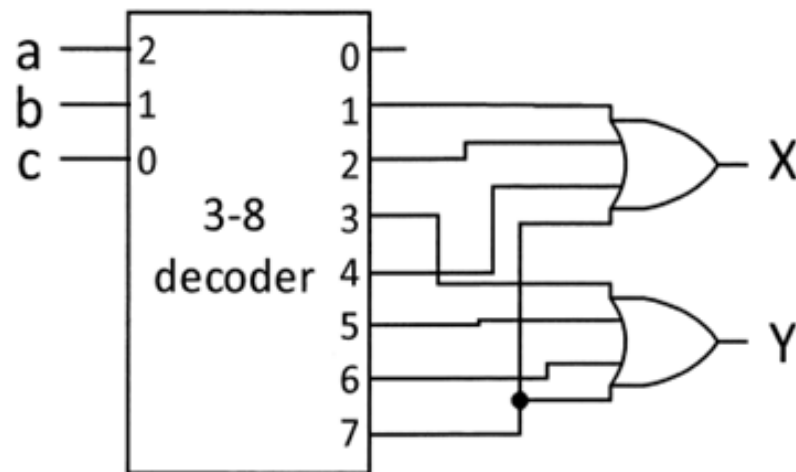


Selected Past Exam Questions

CE/CZ1005 2019-2020 Semester 1 (Nov/Dec 2019)

Q3(a) The circuit given in Figure Q3a uses a 3-8 decoder to implement a common logic function with two outputs, X and Y.

- (i) Using a truth table, or any other means, determine the function of the circuit given in Figure Q3a.



This is the reverse of the previous question

So $X = \sum m(1, 2, 4, 7)$, and

$Y = \sum m(3, 5, 6, 7)$,

as before

Selected Past Exam Questions

CE/CZ1005 2016-2017 Semester 2 (Apr/May 2017)

Q3(c)(i) Draw a schematic diagram to show how the function $F(a,b,c) = a'b'c' + a'bc' + ab'c' + abc$ is implemented using only a 8x1 multiplexer.

(5 marks)

ANS: Thus, $F = \sum m(0, 2, 4, 7)$,
and an 8x1 mux has 8 inputs, 3 select inputs and 1 output.

