

Some Common Combinational Circuits

MUX (Multiplexor)

Decoder

Adder

CompOrg - Combinational Circuits

1

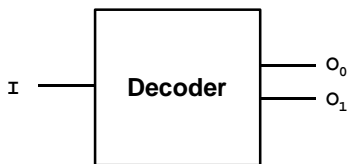
Some commonly used components

- Decoders: n inputs, 2^n outputs.
– *the inputs are used to select which output is turned on. At any time exactly one output is on.*
- Multiplexors: 2^n inputs, n selection bits, 1 output.
– *the selection bits determine which input will become the output.*
- Adder: $2n$ inputs, $2n$ outputs.
– *Computer Arithmetic.*

CompOrg - Combinational Circuits

2

1 input Decoder

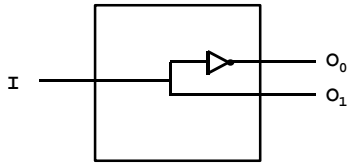


Treat I as a 1 bit integer i . The i^{th} output will be turned on ($O_i=1$), the other one off.

CompOrg - Combinational Circuits

3

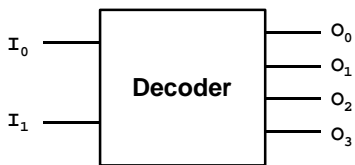
1 input Decoder



CompOrg - Combinational Circuits

4

2 input Decoder

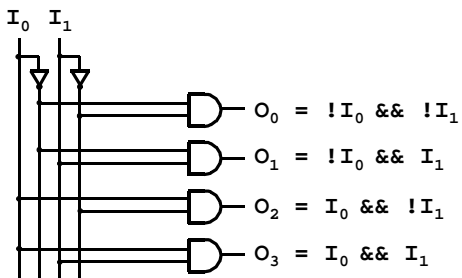


Treat I_0I_1 as a 2 bit integer i . The i^{th} output will be turned on ($O_i=1$), all the others off.

CompOrg - Combinational Circuits

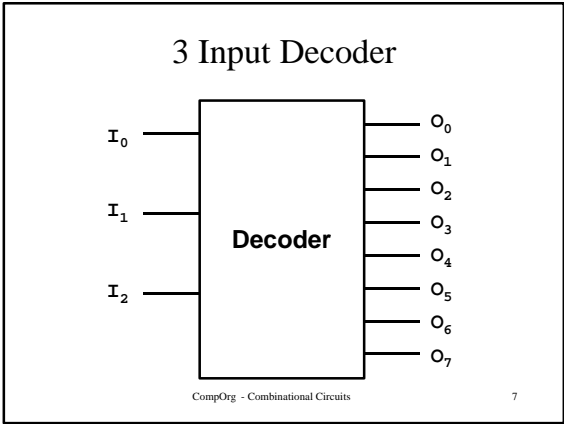
5

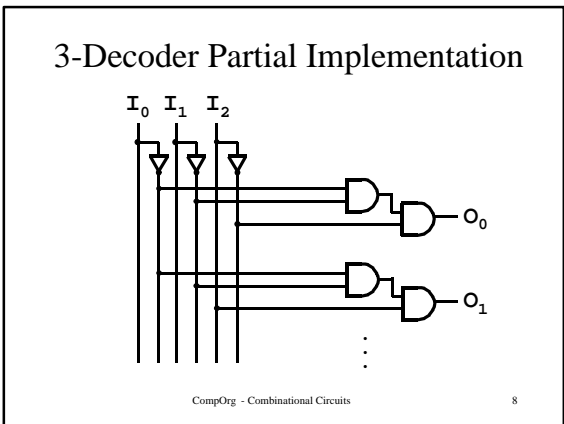
2 input Decoder

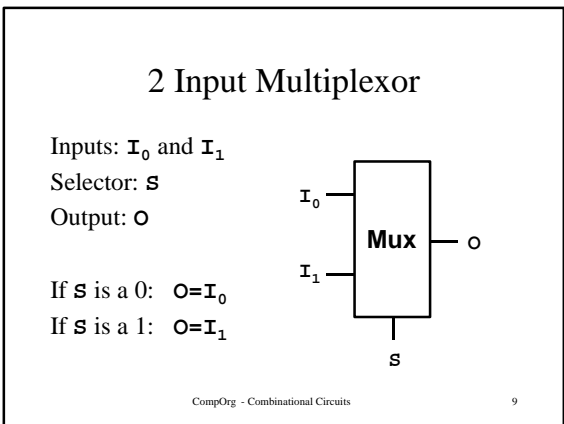


CompOrg - Combinational Circuits

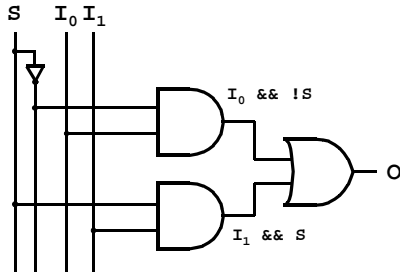
6







2-Mux Logic Design



CompOrg - Combinational Circuits

10

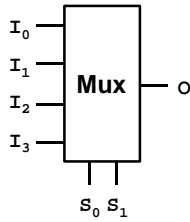
4 Input Multiplexor

Inputs: I_0 I_1 I_2 I_3

Selectors: S_0 S_1

Output: O

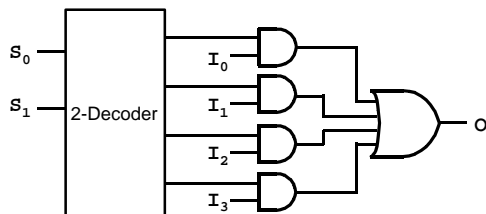
S_0	S_1	O
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



CompOrg - Combinational Circuits

11

One Possible 4-Mux



CompOrg - Combinational Circuits

12

Adder

- We want to build a box that can add two 32 bit numbers.
 - Assume 2s complement representation
- We can start by building a *1 bit adder*.

CompOrg - Combinational Circuits

13

Addition

- We need to build a 1 bit *adder*
 - compute binary addition of 2 bits.
- We already know that the result is 2 bits.

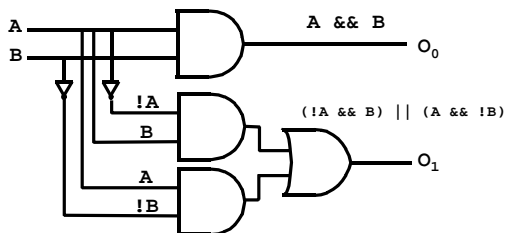
A	B	O ₀	O ₁
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

This is addition! $\begin{array}{r} A \\ + B \\ \hline O_0 \ O_1 \end{array}$

CompOrg - Combinational Circuits

14

One Implementation



CompOrg - Combinational Circuits

15

Binary addition and our *adder*

$$\begin{array}{r}
 1 1 \leftarrow \text{Carry} \\
 01001 \\
 + 01101 \\
 \hline
 10110
 \end{array}$$

What we really want is something that can be used to implement the binary addition algorithm.

- O_0 is the *carry*
- O_1 is the *sum*

CompOrg - Combinational Circuits

16

What about the second column?

$$\begin{array}{r}
 1 \leftarrow \text{Carry} \\
 01001 \\
 + 01101 \\
 \hline
 10110
 \end{array}$$

- We are adding 3 bits
 - new bit is the *carry* from the first column.
 - The output is still 2 bits, a *sum* and a *carry*

CompOrg - Combinational Circuits

17

Truth Table for Addition

A	B	Carry In	Carry Out	Sum
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

CompOrg - Combinational Circuits

18

1 bit adder (3 inputs!)

- We can come up with a logic design:

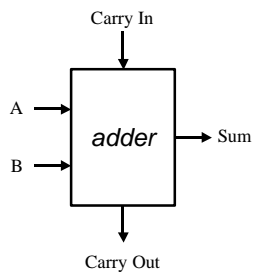
```
Carry Out = (A&&B) || (A&&CarryIn) || (B&&CarryIn)
```

```
Sum = (!A && !B && CarryIn) ||  
      (!A && B && !CarryIn) ||  
      ( A && !B && !CarryIn) ||  
      ( A && B && CarryIn)
```

CompOrg - Combinational Circuits

19

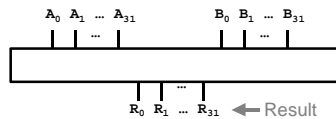
New Component: 1 Bit Adder



CompOrg - Combinational Circuits

20

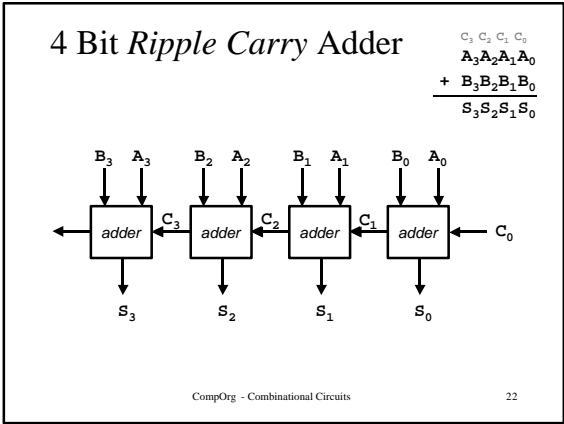
Building a 32 bit Adder

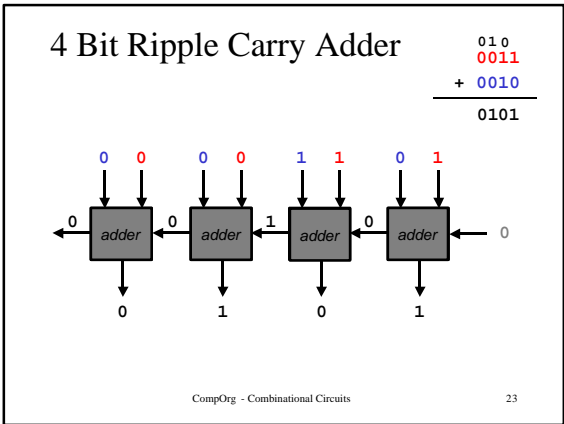


- 64 inputs
- 32 bit output

CompOrg - Combinational Circuits

21

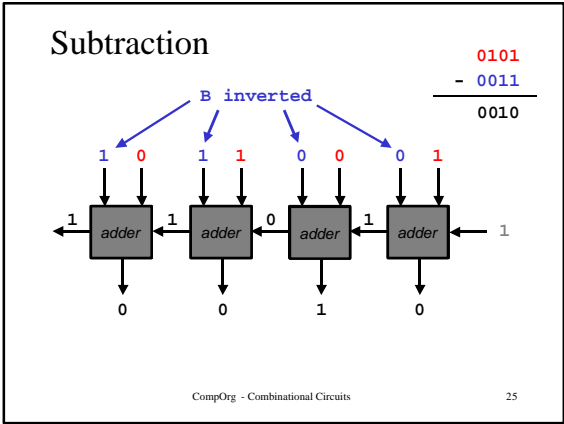




Subtraction

- Compute $A-B$ as $A + (-B-1) + 1$
- $-B-1$ is just all the bits of B inverted.
- Add the $+1$ by setting C_0 to 1

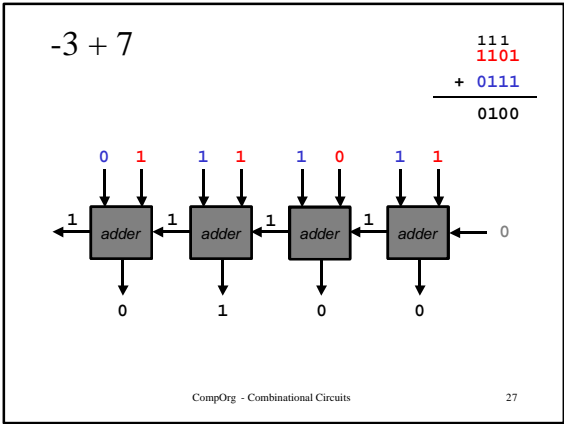
CompOrg - Combinational Circuits 24



Two's Complement Numbers

- Nothing is different!
 - This is the advantage of using 2's complement representation.
- Overflow:
 - For addition: sign of the result is different than the sign of the operands (and they have the same sign).

CompOrg - Combinational Circuits 26



$-3 + -7$

```

      001
      1101
    + 1001
    -----
      0110
  
```

Overflow!

CompOrg - Combinational Circuits 28

Ripple Carry Timing

- All the adders are actually operating all the time (they are just combinational circuits).
- We wait long enough (until the last carry has been computed) and then pay attention to the complete answer.
- It is likely that there are intermediate values that are wrong!

CompOrg - Combinational Circuits 29

Carry Look-ahead

- Compute the carry bits right away.
 - As a function of the inputs A and B.
- Not possible for a large adder (32 bit), but realistic for a 4 bit adder.

CompOrg - Combinational Circuits 30

4 Bit Carry Look-ahead

$$\begin{array}{r}
 010 \\
 0011 \\
 + 0010 \\
 \hline
 0101
 \end{array}$$

