

Combinational Logic Circuits

CIT 595
Spring 2007

CIT 595

4 - 1

Computer Components

- Computer components are made from both combinational and sequential logic circuits
- We will apply the knowledge of Boolean Algebra to realize these circuits
- First we will look at Combinational Logic Circuit

CIT 595

4 - 2

Combinational Logic Circuits

- Always gives the same output for a given set of inputs
- Do not store any information (memoryless)
- Examples: adder, decoder, multiplexer (mux), shifter
 - These are combined to form larger units such as ALU

CIT 595

4 - 3

Bit Shifter (Unsigned Integer)

- Lets see the design of a 2-bit shifter
- To determine 1-bit shift to left or right, assume that this decided by control variable/signal input called S
- If $S = 0$, then we do 1 bit left shift
- Else, we do 1 bit right shift
- Lets say the input is 2-bit value $D(D_1, D_0)$ where D_1 is the most significant bit
- Lets call the output of the shift be $O(O_1, O_0)$

CIT 595

4 - 4

Creating Logic for 2-bit Shifter

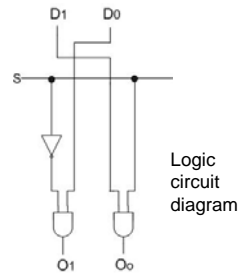
Inputs		Outputs	
S	D1 D0	O1	O0
0	0 0	0 0	0 0
0	0 1	1 1	0 0
0	1 0	0 0	0 0
0	1 1	1 1	0 0
1	0 0	0 0	0 0
1	0 1	1 0	0 0
1	1 0	0 0	0 0
1	1 1	0 0	1 1

O ₁	D ₁ D ₀	S
0	00	0
1	01	0
1	10	0
0	11	0
0	00	1
0	01	1
0	10	1
0	11	1

$O_1 = \overline{S}D_0$

O ₀	D ₁ D ₀	S
0	00	0
0	01	0
0	10	0
1	11	0
0	00	1
0	01	1
1	10	1
1	11	1

$O_0 = SD_1$

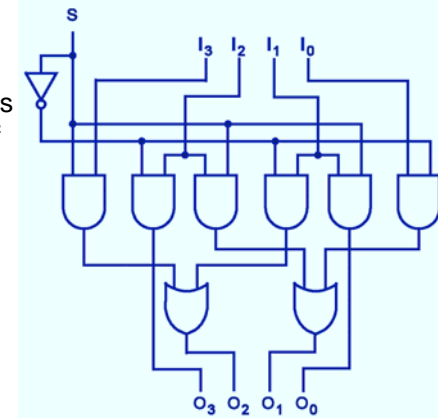


CIT 595

4 - 5

4-Bit Bit Shifter

- Similarly, we can make 4-bit shifter that moves the bits of a nibble (half of a byte) one position to the left or right
- How many rows will the truth table contain?



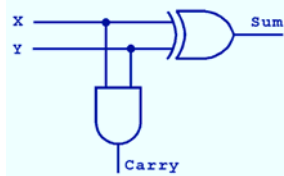
CIT 595

4 - 6

1 Bit Addition Unit (Half Adder)

- The truth table summarizes the outputs of the 1 bit addition based on values of x and y
- The two outputs are sum and carry
- As we see, the sum can be found using the "XOR" operation and the carry using the "AND" operation
- This circuit is known as half adder
 - Does half the job – does not account for carry-in input

Inputs		Outputs	
X	Y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



CIT 595

4 - 7

1 Bit Addition Unit (Full Adder)

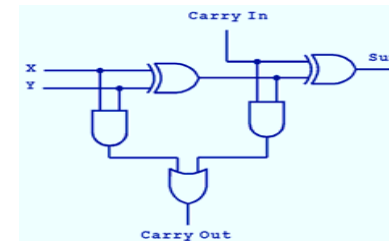
Inputs			Outputs	
X	Y	Carry In	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\begin{aligned} \text{Sum} &= \overline{x}y\overline{\text{Cin}} + \overline{x}y\text{Cin} + x\overline{y}\overline{\text{Cin}} + xy\text{Cin} \\ &= \overline{\text{Cin}}(\overline{x}y + x\overline{y}) + \text{Cin}(\overline{x}y + xy) \\ &= \overline{\text{Cin}}(x \oplus y) + \text{Cin}(x \oplus y) \\ &= (x \oplus y) \oplus \text{Cin} \end{aligned}$$

NOTE: $A \oplus B = \overline{A}B + A\overline{B}$

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

$\overline{x}y + xy = x \oplus y$

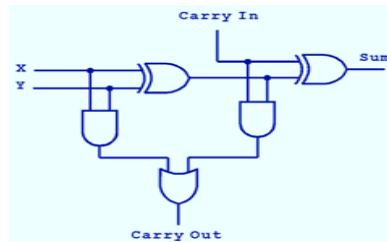


CIT 595

4 - 8

1 Bit Addition Unit (Full Adder) contd..

Inputs			Outputs	
X	Y	Carry In	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$$\text{Carry Out} = x\bar{y}C_{in} + xyC_{in} + \bar{x}yC_{in} + xy\bar{C}_{in}$$

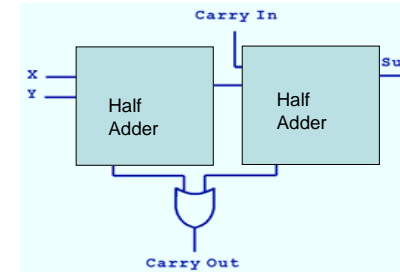
$$C_{in}(x\bar{y} + \bar{x}y) + xy(C_{in} + \bar{C}_{in})$$

$$C_{in}(x \oplus y) + xy$$

CIT 595

4 - 9

1 Bit Full Adder



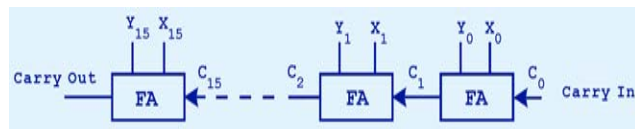
Two half adders make a full adder

CIT 595

4 - 10

N-bit Adder

- Just as we combined half adders to make a full adder, full adders can be connected in series
- The carry bit “ripples” from one full adder to the next; hence, this configuration is called a *ripple-carry adder*



16-bit Ripple Carry Adder

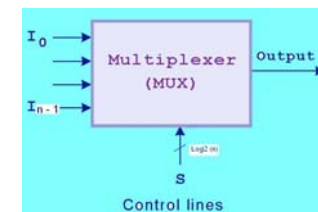
Co is assumed to be 0

CIT 595

4 - 11

Multiplexer

- A multiplexer sets its single output to the same value as one of its many inputs
- Output is determined by the value of the multiplexer’s control lines (a.k.a selector)
- To be able to select among n inputs, $\log_2 n$ control lines are needed



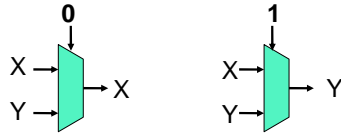
This is a block diagram for a multiplexer

CIT 595

4 - 12

2-to-1 MUX

- Selects between two inputs



- What is the logic behind selection?
 - Is for you to find out (Homework I)

CIT 595

4 - 13

Subtraction

- The adder logic circuit seen before does only addition
- Recall that $X - Y = X + (-Y)$
 - We find 1's complement of Y and add 1 to get negative value of Y i.e. $-Y$
 - Then we add X and $-Y$

$$\begin{array}{r}
 01101000 \text{ (104)} \\
 - 00010000 \text{ (16)} \\
 \hline
 01101000 \text{ (104)} \\
 + 11110000 \text{ (-16)} \\
 \hline
 01011000 \text{ (88)}
 \end{array}
 \qquad
 \begin{array}{r}
 11110110 \text{ (-10)} \\
 - 11110111 \text{ (-9)} \\
 \hline
 11110110 \text{ (-10)} \\
 + 00001001 \text{ (9)} \\
 \hline
 11111111 \text{ (-1)}
 \end{array}$$

CIT 595

4 - 14

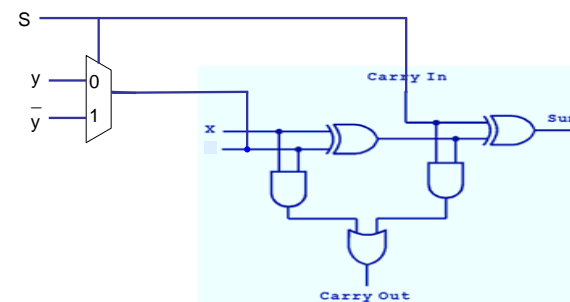
Implementing Subtraction Logic in 1 Bit Adder Unit

- Let Y be the 2nd input
- We need both the Y and Y complement (Y')
- To choose between addition and subtraction we will use a select signal "S" (we will learn later that S is actually generated by the control unit)
 - S = 0, then addition i.e. Y input is chosen
 - S = 1, then subtraction i.e. Y complement is chosen
- If subtraction then we need to add 1 to Y complement (Y') so as to get $-Y$
 - +1 can be achieved by making the first carry C_0 into the adder be 1
 - Hence, $C_0 = S$ (this will allow both operations i.e. add/sub)

CIT 595

4 - 15

Modification to the 1 Bit Adder (w/ Subtraction)



CIT 595

4 - 16

Detecting Arithmetic Overflow

Overflow is said to occur if result is too large to fit in the number of bits used in the representation

Carry into	→ 1	01000 (8)	01000 (-8)
MSB	+	01001 (9)	10111 (-9)
Carry Out	→ 0	10001 (-15)	10111 (+15)

We have overflow if

- Signs of both numbers are the same, and Sign of sum is different
- If Positive number is subtracted from a Negative number, result is positive and vice versa

Another test (easy for hardware)

- Carry into MSB does not equal carry out

CIT 595

4 - 17

Detecting Overflow

- Circuit outputs 1 when the Carry into MSB (MCin) does not equal carry out (Cout)

Cout	MCin	Overflow
0	0	0
0	1	1
1	0	1
1	1	0

- If you observe carefully, the output is equivalent to XOR gate

- Thus to detect overflow we XOR the values of Cout and MCin

- In general, for n-bit adder
Overflow = $C_{n-1} \oplus C_{n-2}$

↓
Carry Out
from MSB

CIT 595

4 - 18

Decoder

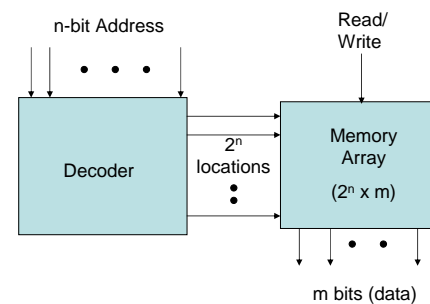
- Decoders are circuits used to decode encoded information
- A binary decoder converts binary information from n-bit coded inputs to a maximum of 2^n unique outputs
 - Decoder logic uses n-bit input value to choose exactly one of the 2^n outputs (only a particular output is active)
 - Example: Memory Address Decoding



CIT 595

4 - 19

Address Decoder Example



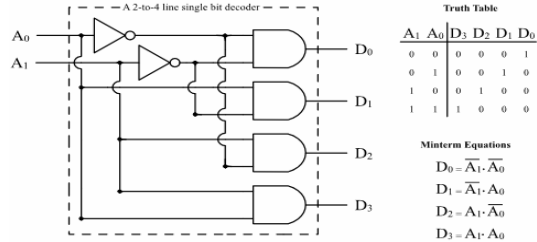
Decoder will select only 1 memory location (row) based on address

CIT 595

4 - 20

2-to-4 Decoder Logic

- A binary number with 2 bits as its input
- Selects exactly 1 of 4 outputs
 - At any time, only 1 output line is "ON" or "1" and all others are "OFF" or "0" (referred to as one-hot encoded)
- The line that is "ON" specifies the desired location



1 – active/asserted/chosen

0 – not active/deasserted/not chosen

CIT 595

4 - 21

Encoder

- Opposite of decoder
- Given information is transformed into more compact form
- Example:
 - In Interrupt Driven I/O – Priority encoder decides which interrupt request should be serviced by the processor
 - See "Handout" provided on how priority encoder works
 - More on this when we visit I/O in chapter 7

CIT 595

4 - 22

Aside: Code Converters

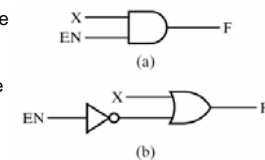
- Convert from one type of coded information (encoding) to another output encoding
 - Example: 7 segment display (calculators and digital number displays etc) – see "Handout" on code converters (Read sec 2.6.1 on chp2 first – pg 74)

CIT 595

4 - 23

Enabling/Gating Outputs

- Combinational logic circuits produce an output based on certain inputs
- However, we may not want to use its output all the time
 - Same inputs are shared amongst different logic units within a component but we want the output from a particular logic unit
 - Some how we only want to make one unit active and disable all others
- Hence we want some sort of control that would temporarily disable the circuit and only enable it if the control is set
 - AND gate - Figure a. EN = 1, F = X
 - OR gate - Figure b. $\bar{EN} = 0$, F = X



Who decides value of EN?

Example on the next slide

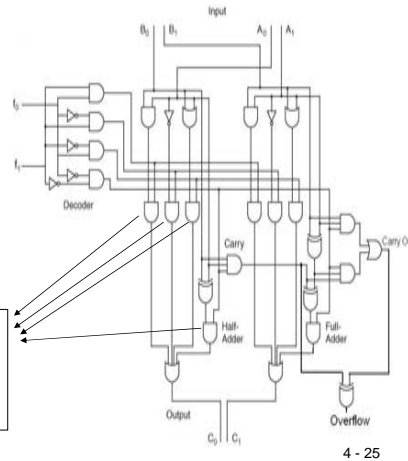
CIT 595

4 - 24

2-Bit ALU

- f_0 and f_1 control lines (generated by control unit)
- The value of control lines determine which operation:
 00 – A + B (Addition)
 01 – NOT A
 10 – A OR B
 11 – A AND B
- Similarly a N-Bit ALU can be made

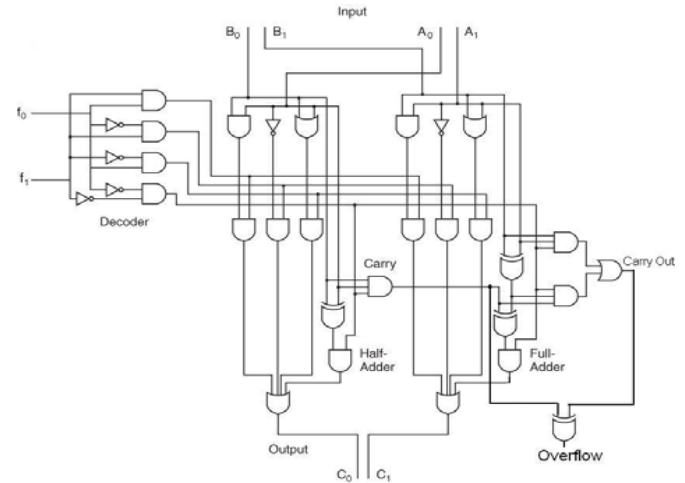
All sub units form their operation, but final output is chosen only if enabled ($EN = 1$). Here EN is decided by the decoder logic.



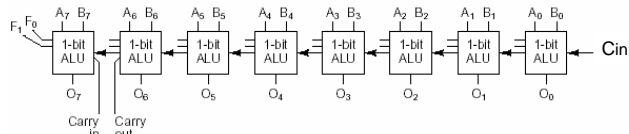
CIT 595

4 - 25

2-Bit ALU



8-bit ALU made from eight "bit slices"



- Bit slices allow designers to build an ALU of any desired bit capacity
- **Carry out** of each bit slice connects to **carry in** of next (more significant bit) slice
- F_0 and F_1 (decoder inputs) connect simultaneously to all slices so that the identical operation is selected in all slices at a given time
- There is a single input to the least significant slice i.e. carry-in (C_{in}) input

CIT 595

4 - 27