Signed Numbers: Addition/ Subtraction

Decimal	Signed-1's Complement
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
−0	1111
−1	1110
−2	1101
−3	1100
−4	1011
−5	1010
−6	1001
−7	1000
−8	—

One's comp.

0110
1001

no conversion

4 bit compl.

+2	0010	5	0101	1	0001
+3	+ 0011	-6	+ 1001	-7	+ 1000
5	0101	-1	1110	-6	1001

no carry take 1's comp. 0001

5	0101	-5	1010
-3	+ 1100	-2	+ 1101
2	10001	-7	10111
	+0001		+0001
	0010		1000

add

1111
-0011

Signed Numbers: Addition/ Subtraction

Decimal	Signed-1's Complement
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
−0	1111
−1	1110
−2	1101
−3	1100
−4	1011
−5	1010
−6	1001
−7	1000
−8	—

- The addition of 1's complement numbers may or may not be simple. n bit → (n+1) bit ← added (n+1) th bit
- In some cases, a correction is needed, which amounts to an extra addition that must be performed. (2n full adders)
- Consequently, the time needed to add two 1's complement numbers may be twice as long as the time needed to add two unsigned numbers.

Realize in hardware carry is 1 etc

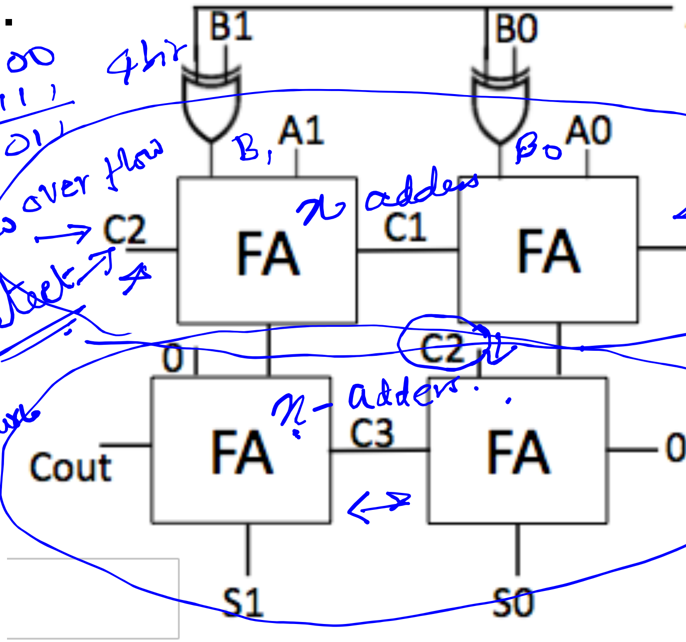
Many realization

0111
0111
(1)110
-ve → X overflow X 0111

0111
0111
(1)110
check

01100
01111
(1)1011
4 bit

overflow
3 bit problem
detect
2n
heavy resource demand



M = 0 add
n bit signed (n-1)
carry in
one more control
set of adders

Signed Numbers: Addition/ Subtraction

Decimal	Signed-2's Complement
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
−0	—
−1	1111
−2	1110
−3	1101
−4	1100
−5	1011
−6	1010
−7	1001
−8	1000

1000

$-(2^3-1)$

0110
1001
1010

2	0010
+3	+ 0011
5	0101

5	0101
-6	+ 1010
-1	1111

-2^3 -1

1	0001
-7	+ 1001
-6	1010

5	0101
-3	+ 1101
2	~~1~~0010

drop it

-5	1011
-2	+ 1110
-7	~~1~~1001

drop it

More comfortable

Addition/Subtraction

Architecture (adder/subtractor):

A ← $A_3A_2A_1A_0$ ← 2's comp. rep.
B ← $B_3B_2B_1B_0$

2's comp.

Preferred

A+B
A−B

n bit

1 → subtract

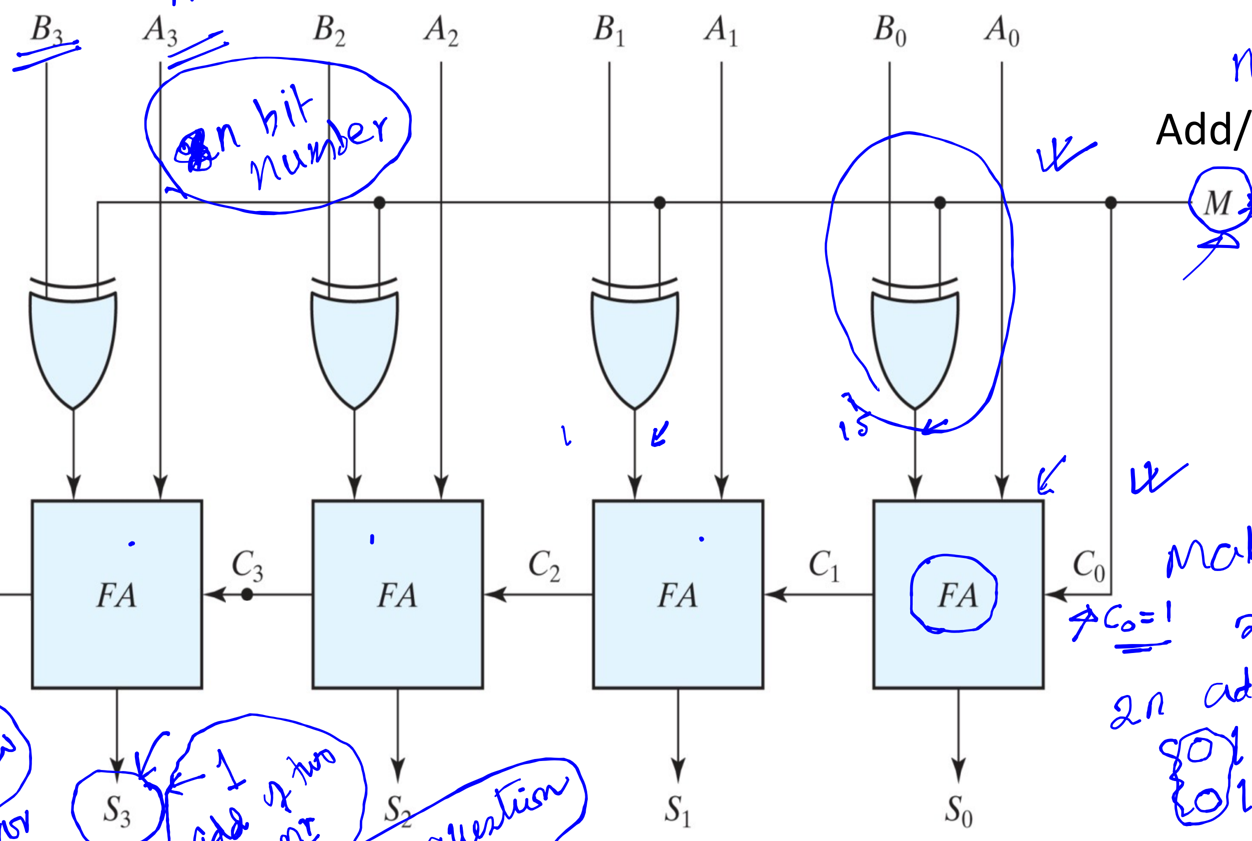
overflow

0110
1001
1

$x-(-y)$
$-x-y$
$-x+y$
$-x+(-y)$ rep.
$-x-(-y)$

n bit

1 sign
(n−1) mag. bits.



Add/Sub

Add: M = 0

Sub: M = 1

no. can be signed
Add or Sub
control input

overflow
M=0 drop it
if M=1 drop it

it is an overflow
overflow error
overflow

1 add of two −ve no.

good question

$\bar{A_3}\cdot\bar{B_3}\cdot\bar{M}$

makes it 2's comp.
$C_0=1$

2's comp.
1's comp
+1

2n adders

0110
0101
× 1011

1110
1101
11011
−5

Examples:

Read this slide & respond

When not indicated the radix is 10.

- Perform addition and subtraction for unsigned numbers:

1) $(AA)_{16} + (12)_{16}$
2) $(23)_8 + (56)_8$
3) $(24)_5 + (12)_5$

$(AA)_{16}$	170	$(23)_8$	19	$(24)_5$	14
$+(12)_{16}$	+ 18	$+(56)_8$	+ 46	$+(12)_5$	+ 7
$(BC)_{16}$	188	$(101)_8$	65	$(41)_5$	21

- Perform addition and subtraction for signed numbers using r's and (r-1)'s representation:

1) $(AA)_{16} - (12)_{16}$
2) $(23)_8 - (56)_8$ ← do this
3) $(24)_5 - (32)_5$

$(AA)_{16}$	$(AA)_{16}$	170	$(23)_8$	$(23)_8$	19
$-(12)_{16}$	$+(EE)_{16}$	- 18	$-(56)_8$	$+(22)_8$	- 46
	$(198)_{16}$	152		$(45)_8$	-27
	$(98)_{16}$			$-(33)_8$	

Examples for Practice:

- Consider X=1010100 and Y=1000011. Find X-Y and Y-X using 1’s complement
- Consider X=1010100 and Y=1000011. Find X-Y and Y-X using 2’s complement
- Using 10’s complement representation, subtract 72532 – 3250.
- Using 10’s complement representation, subtract 3250-72532.

Binary Codes:

alpha numeric

Binary Number Codes

- A group of symbols is called as a code
- A digital data is represented, stored and transmitted as group of binary bits. This group is also called as **binary code**.
- A binary code is used to represent both number as well as alphanumeric letters and special characters.
- Useful in various applications, makes analysis and implementation easy since they are represented using 0 and 1

Binary Codes:

BCD and 2421 Codes are called Weighted Codes:

Decimal Digit	BCD 8421	2421	Excess-3
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100
Unused bit combinations	1010	0101	0000
	1011	0110	0001
	1100	0111	0010
	1101	1000	1101
	1110	1001	1110
	1111	1010	1111

2421 and Excess-3 code are called Self Complimenting Codes.

Excess- 3 Code is called an Unweighted Code.

extensively
conversion
position
convenience
non - weighted
calculator
4 bits
symbols
displays
calculator
machines
answer
Weighted code
BCD
Weighted
89 → 1001
0100
0100 1001
Binary Coded Decimal
6
decimal
decimal number
Code
convenience
Code
machine
Self complementing
8bit
calculator
87 → 1110 1101
complement domain

Binary Codes:
Gray Code:

non-weighted

Gray Code	Decimal Equivalent	Binary Code
0000	0	0000
0001	1	0001
0011	2	0010
0010	3	0011
0110	4	0100
0111	5	0101
0101	6	0110
0100	7	0111
1100	8	1000
1101	9	1001
1111	10	1010
1110	11	1011
1010	12	1100
1011	13	1101
1001	14	1110
1000	15	1111

0000

Also called the Reflected Binary Code and a Cyclic Binary Code.

Successive symbols differ only by one bit, minimum switching and hence minimum noise generated. less power

Binary Code to Gray Code Conversion:

- Consider a given Binary Code and place a zero to the left of MSB.
- Compare the successive two bits, starting from extended MSB (zero). If the two bits are same, then the output is zero, otherwise 1.

XOR

- Repeat till the LSB of Gray Code is reached.

$g_3 = b_3$; $g_2 = b_3 \oplus b_2$; $g_1 = b_2 \oplus b_1$; $g_0 = b_1 \oplus b_0$

Decimal Equivalent	Binary Code	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Gray Code to Binary Code Conversion:

- Consider a given Gray Code and and its Binary Code equivalent, their MSBs will be identical. For 4-bit representation, $b_3 = g_3$

- Other bits of the output binary code can be obtained by checking Gray code bit at that index. If current Gray code bit is 0, then copy previous binary code bit, else copy invert of previous binary code bit. i.e., $b_2 = b_3 \oplus g_2 = g_3 \oplus g_2$. Similarly, $b_1 = b_2 \oplus g_1 = g_3 \oplus g_2 \oplus g_1$ and $b_0 = b_1 \oplus g_0 = g_3 \oplus g_2 \oplus g_1 \oplus g_0$.

Decimal Equivalent	Gray Code	Binary Code
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

Homework:

answer in the next class

not able (1) solve

not able (2) check

not

not solutions to Home work

- Study various applications of Gray codes
- Design the binary to Gray code converter circuit where input is 3-bit binary number and output is 3-bit Gray code.
- Design the Gray to Binary code converter circuit where input is 3-bit Gray code and output is 3-bit binary number.