Arithmetic

Kevin Walsh
CS 3410, Spring 2010
Computer Science
Cornell University

See: P&H Chapter 3.1-3, C.5-6
• Adds two 4-bit numbers, along with carry-in
• Computes 4-bit result and carry out
Addition with negatives:

- $\text{pos} + \text{pos} \rightarrow \text{add magnitudes, result positive}$
- $\text{neg} + \text{neg} \rightarrow \text{add magnitudes, result negative}$
- $\text{pos} + \text{neg} \rightarrow \text{subtract smaller magnitude, keep sign of bigger magnitude}$

$A + B$
First Attempt: Sign/Magnitude Representation

- 1 bit for sign (0=positive, 1=negative)
- N-1 bits for magnitude

\[ \begin{align*}
0\ 1\ 1\ 1\ 1 &= +7 \\
1\ 1\ 1\ 1\ 1 &= -7
\end{align*} \]
Better: Two’s Complement Representation

- Leading 1’s for negative numbers
- To negate any number:
  - complement all the bits
  - then add 1, ignore carry out.

\[ +6 = 0110 \quad +20 = 00010100 \]
\[ \sim 6 = 1001 \quad \sim 20 = 11101000 \]
\[ +1 \]
\[ -6 = 1010 \quad -20 = 11101100 \]
Non-negatives (as usual):
+0 = 0000
+1 = 0001
+2 = 0010
+3 = 0011
+4 = 0100
+5 = 0101
+6 = 0110
+7 = 0111
+8 = 1000

Negatives (two’s complement: flip then add 1):
\(~0 \neq 1111\)
\(~1 \neq 1110\)
\(~2 \neq 1101\)
\(~3 \neq 1100\)
\(~4 \neq 1011\)
\(~5 \neq 1010\)
\(~6 \neq 1001\)
\(~7 \neq 1000\)
\(~8 \neq 0111\)
Signed two’s complement

- Negative numbers have leading 1’s
- zero is unique: $+0 = -0$
- wraps from largest positive to largest negative

N bits can be used to represent

- unsigned:
  - eg: 8 bits $\Rightarrow 0 \ldots 2^8 - 1 = 255$
- signed (two’s complement):
  - ex: 8 bits $\Rightarrow -128 \ldots 0 \ldots +127$
  - $-\left(\frac{2^N}{2}\right) \ldots \left(\frac{2^N}{2}\right) - 1$
Extending to larger size

\[ 0110 \]

\[-1 = 1111\]

Truncate to smaller size

Drop leading sign bits, as long as sign doesn’t change

\[ 00001111 \]

\[ 0111 = 7 \]
Addition with two’s complement signed numbers

- Perform addition as usual, regardless of sign (it just works)

```
\[
\begin{array}{cccccc}
\text{Cout} & \text{R}_3 & \text{R}_2 & \text{R}_1 & \text{R}_0 \\
\end{array}
\]
```

\[+3 \quad +(-6) = -3\]
How does that work?

\[
\begin{array}{c}
-154 \\
+283 \\
\hline
\end{array}
\]

\[
\begin{array}{c}
999 \\
-154 \\
\hline
845 \\
283 \\
\hline
0
\end{array}
\]

10's complement
Overflow

- adding a negative and a positive?

- adding two positives?

- adding two negatives?

Rule of thumb:

Overflow happened iff carry into msb := carry out of msb
Two’s Complement Adder with overflow detection

\[ \begin{align*}
O(111) + 1 & \quad \text{(Overflow)} \\
\downarrow & \\
0 & \quad \text{(Result)}
\end{align*} \]
Two's Complement Subtraction

Lazy approach

\[ A - B = A + (-B) = A + (\overline{B} + 1) \]

\[ B = -8 = \underbrace{1111}_{(7)_{10}} - (-8) \]

Q: What if (-B) overflows?
A Calculator

A

B

S

0=add
1=sub

decoder
if (b == five)
then --
• Is this design fast enough?
• Can we generalize to 32 bits? 64? more?
Speed of a circuit is affected by the number of gates in series (on the *critical path* or the *deepest level of logic*)

\[ t_{\text{combinational}} \]
• First full adder, 2 gate delay
• Second full adder, 2 gate delay
• ...