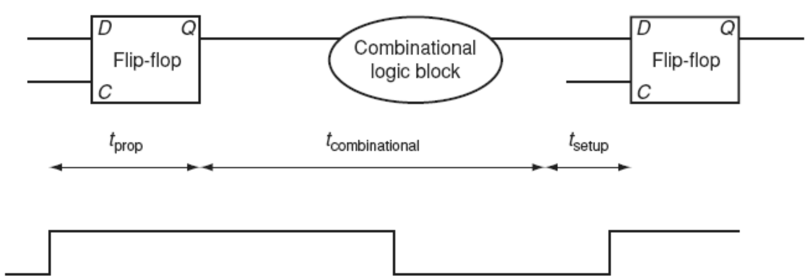
CS 316: Binary Arithmetic

Kavita Bala

Fall 2007

Computer Science

Cornell University



References

- Look at Appendix B in textbook
- Look in Chapter 3 in textbook for today

Recap

- We can build combinatorial circuits
  - Gates, Karnaugh maps, minimization
- We can build stateful circuits
  - Record 1-bit values in latches and flip-flops
- Powerful combination
  - We can build real, useful devices
  - But we will often need to perform arithmetic

Binary Arithmetic

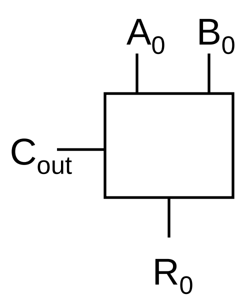
$$\begin{array}{r} 12 \\ +\ 25 \\ \hline 37 \end{array}$$

- Arithmetic works the same way regardless of base
  - – Add the digits in each position
  - – Propagate the carry

$$\begin{array}{r} 001100 \\ +\ 011010 \\ \hline 100110 \end{array}$$

- Unsigned binary addition is pretty easy
  - – Combine two bits at a time
  - – Along with a carry

1-bit Adder

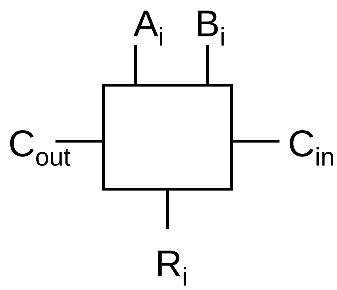


A_0	B_0	C_{out}	R_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- Adds two 1-bit numbers, computes 1-bit result and carry out
- Useful for the rightmost binary digit, not much else

Kavita Bala, Computer Science, Cornell University

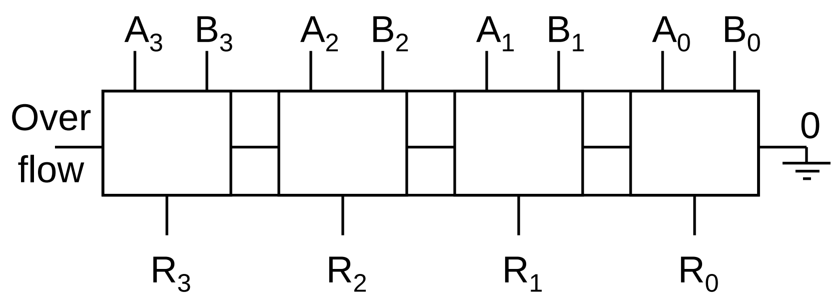
1-bit Adder with Carry



C_{in}	A_i	B_i	C_{out}	R_i
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

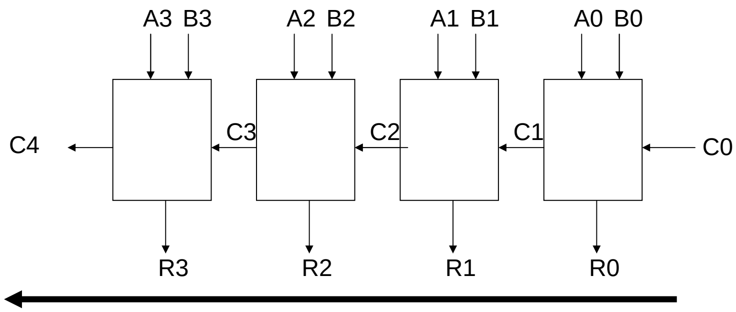
- Adds two 1-bit numbers, along with carry-in, computes 1-bit result and carry out
- Can be cascaded to add N-bit numbers

4-bit Adder



- Adds two 4-bit numbers, along with carry-in, computes 4-bit result and overflow
- Overflow indicates that the result does not fit in 4 bits

4-bit Ripple Carry Adder



Carry ripples from right to left

- First adder, 2 gate delay
- Second adder, 2 gate delay

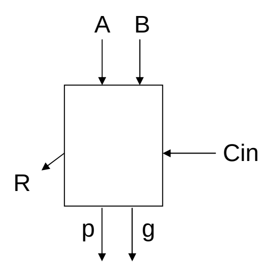
Observations

- Have to wait for Cin
- Can we compute in parallel in some way?
- CLA carry look-ahead adder

Carry Look Ahead Logic

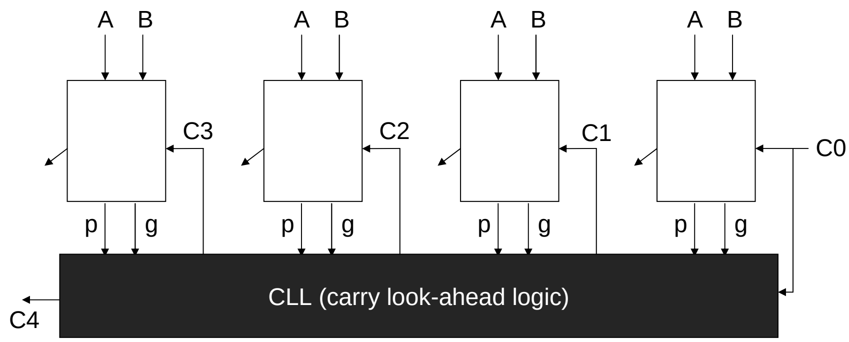
- Can we reason independent of Cin?
  - Just based on (A,B) only
- When is Cout == 1, irrespective of Cin
- If Cin == 1, when is Cout also == 1

1-bit CLA adder



- Create two terms: propagator, generator
- g = 1, generates Cout: g = AB
  - Irrespective of Cin
- p = 1, propagates Cin to Cout: p = A + B
- p and g generated in 1 cycle delay
- R is 2 cycle delay after we get Cin

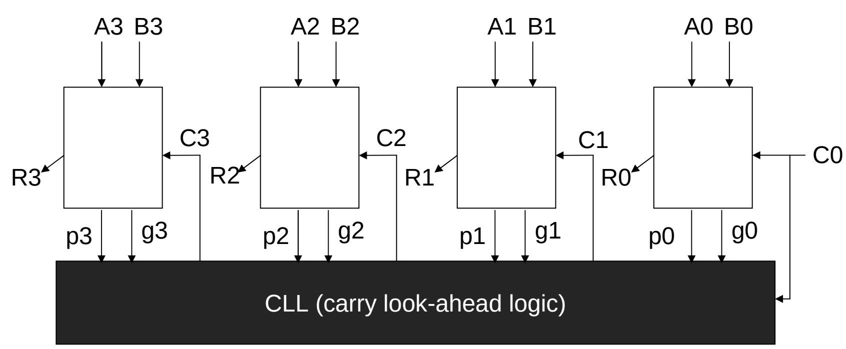
4-bit CLA



- CLL takes p,g from all 4 bits, C0 and generates all Cs
  - 2 gate delay
- C1=g0+p0C0,
- C2=g1+p1(g0+p0C0) = g1+p1g0+p1p0C0
- C3=g2+p2g1+p2p1g0+p2p1p0c0,
- C4=g3+p3g2+p3p2g1+p3p2p1g0+p3p2p1p0c0

Kavita Bala, Computer Science, Cornell University

4-bit CLA



- Given A,B's, all p,g's are generated in 1 gate delay in parallel.
- Given all p,g's, all C's are generated in 2 gate delay in parallel.
- Given all C's, all R's are generated in 2 gate delay in parallel.

- Sequential operation in RCA is made into parallel operation!!

Kavita Bala, Computer Science, Cornell University

Bigger CLAs?

- Can't do it for all 32 bits though

- So use hierarchical construction

Arithmetic with Negative Numbers

- Negative numbers complicate arithmetic

- Recall that for addition and subtraction, the rules are:
  – Both positive => add, result positive
  – One +, one - => subtract small number from larger one
  – Both negative => add, result negative

Arithmetic with Negative Numbers

- We could represent sign with an explicit bit
  - the “sign-magnitude form”
  - But arithmetic would be much easier to perform in hardware if we did not have to examine the operands’ signs
- Two’s complement representation enables arithmetic to be performed without examining the operands

Two’s Complement

- Nonnegative numbers are represented as usual
  - 0 = 0000
  - 1 = 0001
  - 3 = 0011
  - 7 = 0111
- To negate a number, flip all bits, add one
  - -1: 1 ⇒ 0001 ⇒ 1110 ⇒ 1111
  - -3: 3 ⇒ 0011 ⇒ 1100 ⇒ 1101
  - -7: 7 ⇒ 0111 ⇒ 1000 ⇒ 1001
  - -8: 8 ⇒ 1000 ⇒ 0111 ⇒ 1000
  - -0: 0 ⇒ 0000 ⇒ 1111 ⇒ 0000 (this is good, -0 = +0)

Two’s Complement Facts

- Negative numbers have a leading 1
  - Similar to signed magnitude form
  - Largest negative=1000…0
  - Largest positive=0111…1
  - Top most bit: sign bit
- N bits can be used to represent
  - unsigned: the range $0..2^N-1$
  - ex: 8 bits $\Rightarrow$ 0..255
  - two’s complement: the range $-(2^{N-1})..(2^{N-1})-1$
  - ex: 8 bits $\Rightarrow$ (10000000)..(01111111) $\Rightarrow$ -128..127

Kavita Bala, Computer Science, Cornell University

Want to change bit size of number?

- 4 bit to 8 bit
  - For positive number, just 0’s in new 4 bits
  - For negative number, 1’s in new 4 bits
- What about shifting
  - sll (shift left logical)
  - sra (shift right arithmetic), srl (shift right logical)

Kavita Bala, Computer Science, Cornell University

Two's Complement Addition

- Perform addition as usual, regardless of sign
  - 1 = 0001, 3 = 0011, 7 = 0111, 0 = 0000
  - -1 = 1111, -3 = 1101, -7 = 1001
- Examples
  - 1 + -1 = 1111 + 0001 = 0000 (0)
  - -3 + -1 = 1111 + 1101 = 1100 (-4)
  - -7 + 3 = 1001 + 0011 = 1100 (-4)

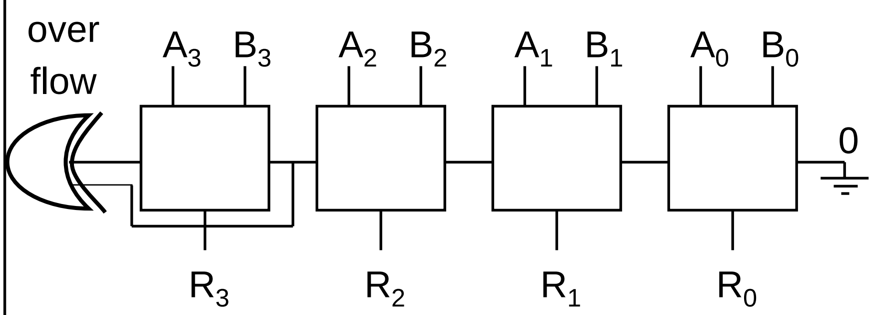
Overflow

- When can it occur?
  - If you add a negative and positive number
    - Cannot occur (Why?)
  - If you add two negatives or two positives
    - Can occur (Why?)
  - Add two positives, and get a negative number
  - Or, add two negatives, get a positive number
    - Overflow!
  - Overflow when
    - Carry into most significant bit (msb) != carry out of msb

Kavita Bala, Computer Science, Cornell University

Two's Complement Adder

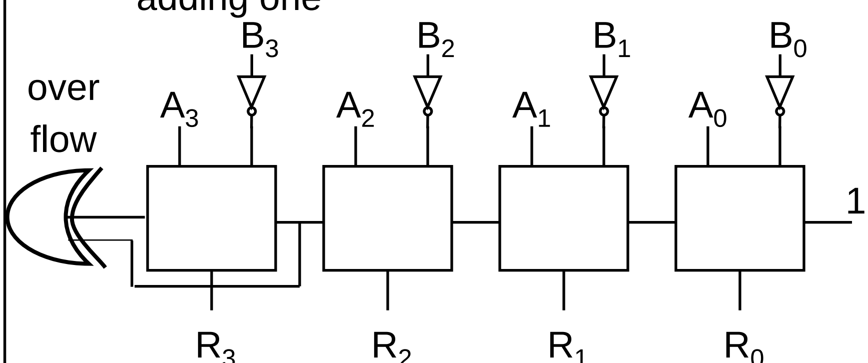
- Let's build a two's complement adder



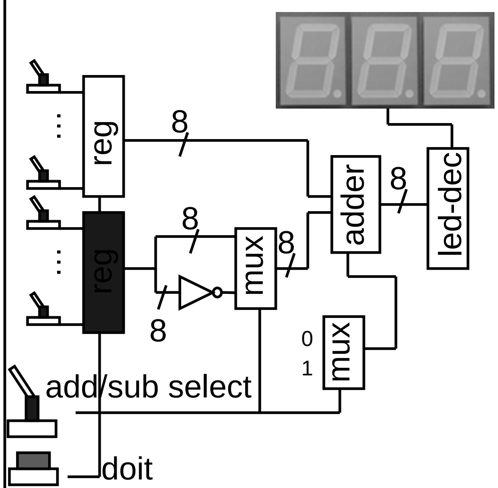
- Already built, just needed to modify overflow checking

Two's Complement Subtraction

- Subtraction is simply addition, where one of the operands has been negated
  - Negation is done by inverting all bits and adding one



A Calculator



- Enter numbers to be added or subtracted using toggle switches
- Select: ADD or SUBTRACT
- Muxes feed A and B,or A and –B, to the 8-bit adder
- The 8-bit decoder for the hex display is straightforward

Summary

- We can now perform arithmetic
  - – And build basic circuits that operate on numbers