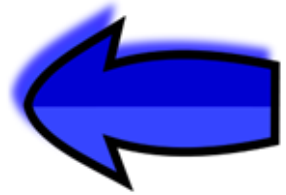
Basic Computer Architecture

Chapter 8: Computer Arithmetic (Part I)

Outline

* Addition
* Multiplication
* Division
* Floating Point Addition
* Floating Point Multiplication
* Floating Point Division



Adding Two 1 bit Numbers

* Let us add two 1 bit numbers – a and b
  * 0 + 0 = 00
  * 1 + 0 = 01
  * 0 + 1 = 01
  * 1 + 1 = 10
* The lsb of the result is known, as the sum, and the msb is known as the carry

Sum and Carry

$$\begin{array}{r} + \quad a \\ \quad\quad b \\ \hline \end{array}$$

carry | sum

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

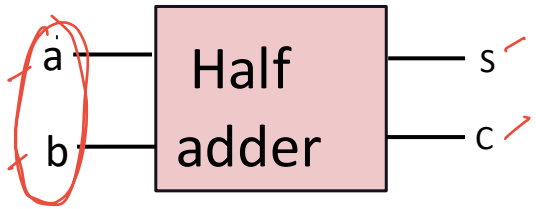
Truth Table

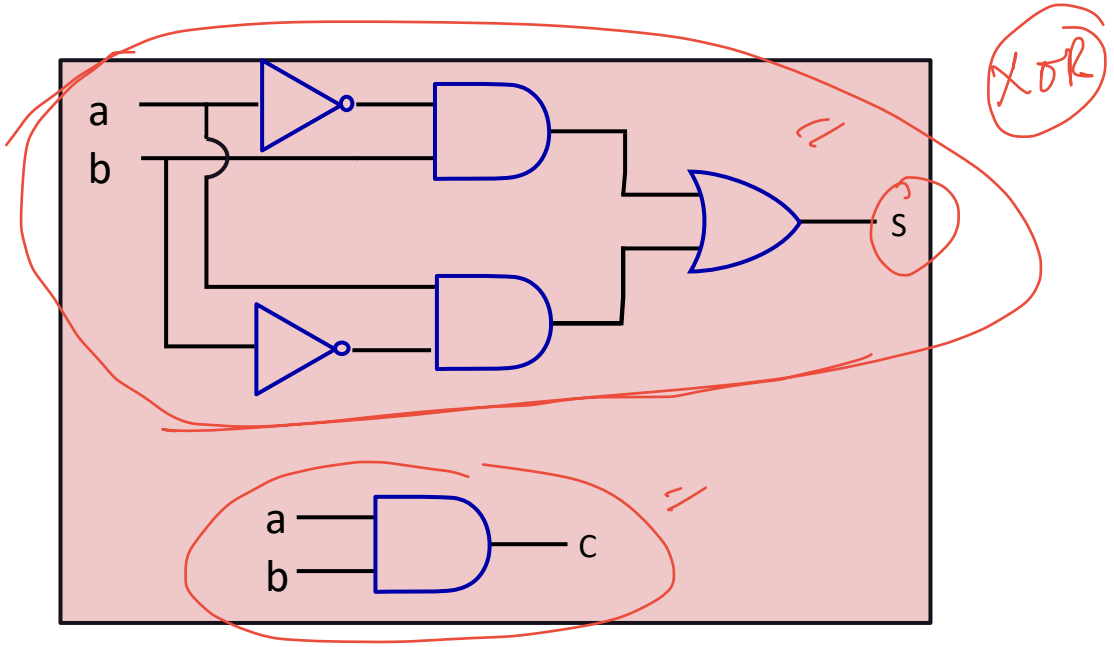
$$S = a \oplus b = \bar{a}.b + a.\bar{b}$$

$$c = a.b$$

Half Adder

* Adds two 1 bit numbers to produce a 2 bit result



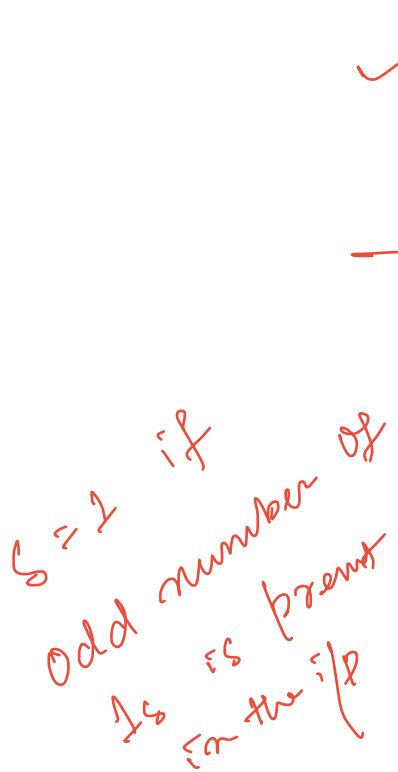


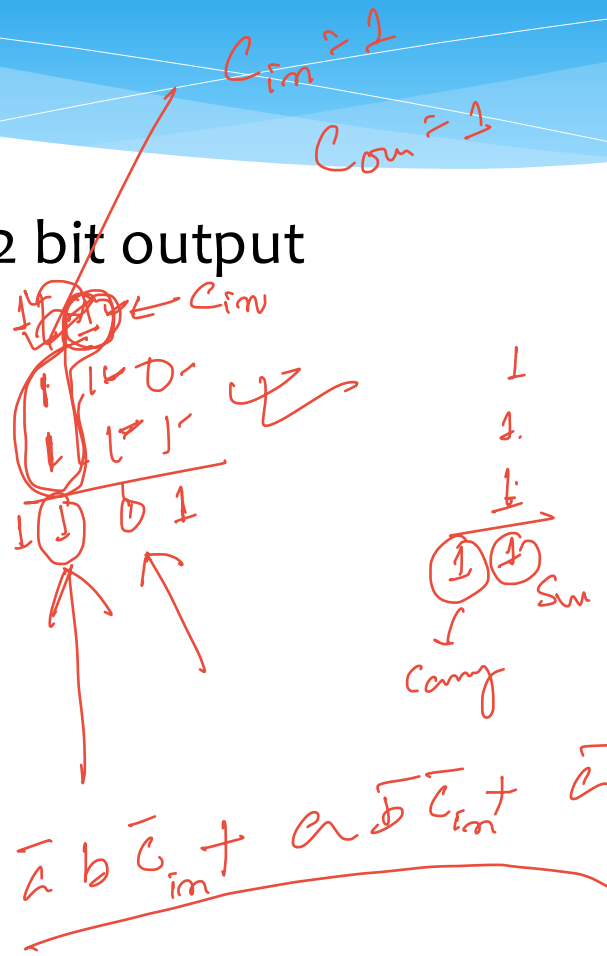
Full Adder

Add three 1 bit numbers to produce a 2 bit output

$$a + b + c_{in} = 2 * c_{out} + s$$

a	b	c_{in}	s	c_{out}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1





S=1 if odd no of 1s in a, b, c_in

Equations for the Full Adder

$$s = a \oplus b \oplus c_{in}$$

$$= \left(a.\bar{b} + \bar{a}.\mathrm{b}\right) \oplus c_{in}$$

$$= \left(a.\bar{b} + \bar{a}.\mathrm{b}\right).\overline{c_{in}} + \overline{a.\bar{b} + \bar{a}.\mathrm{b}}.c_{in}$$

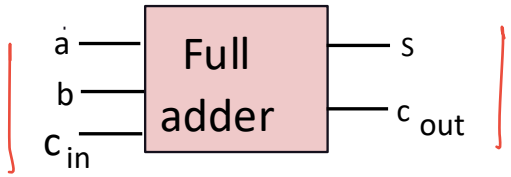
$$= a.\bar{b}.\overline{c_{in}} + \bar{a}.\mathrm{b}.\overline{c_{in}} + \overline{(a.\bar{b})}.\overline{(\bar{a}.b)}.c_{in}$$

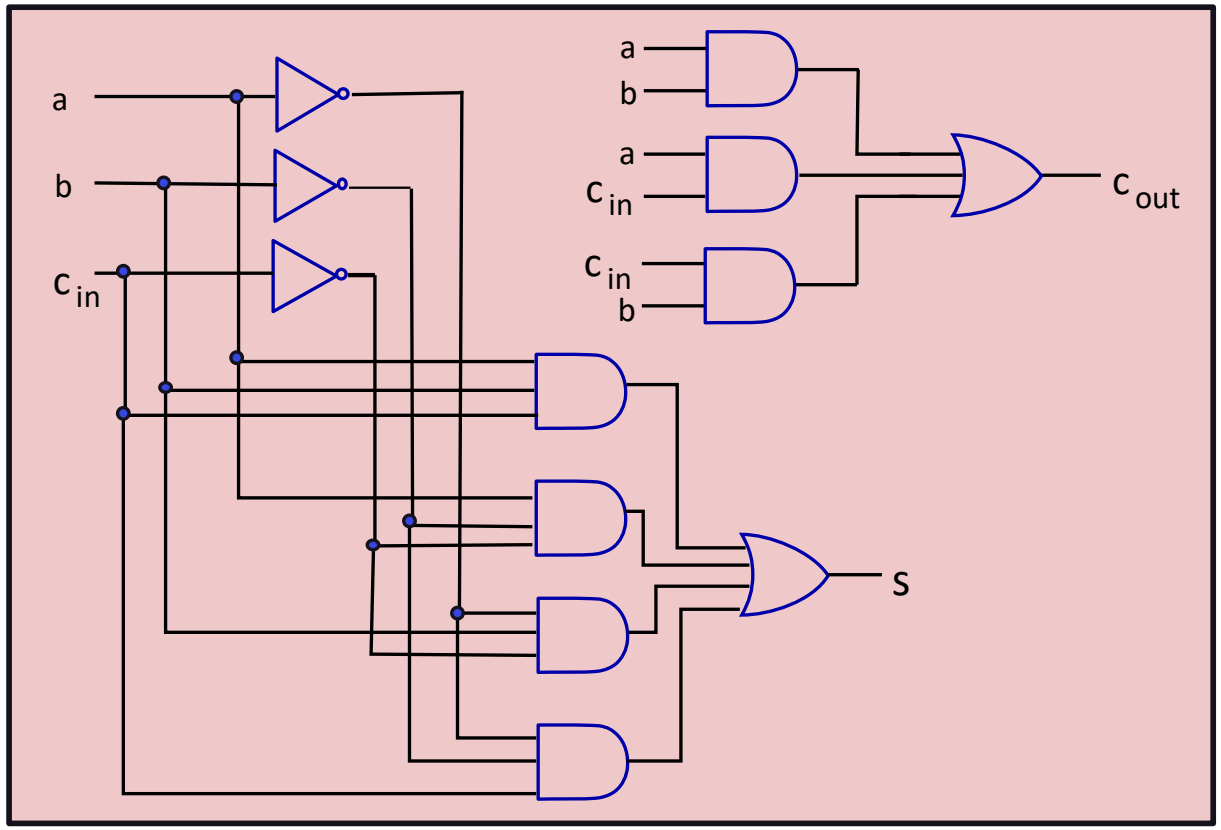
$$= a.\bar{b}.\overline{c_{in}} + \bar{a}.\mathrm{b}.\overline{c_{in}} + (\bar{a} + b).\left(a + \bar{b}\right).c_{in}$$

$$= a.\bar{b}.\overline{c_{in}} + \bar{a}.\mathrm{b}.\overline{c_{in}} + \bar{a}.\bar{b}.c_{in} + a.b.c_{in}$$

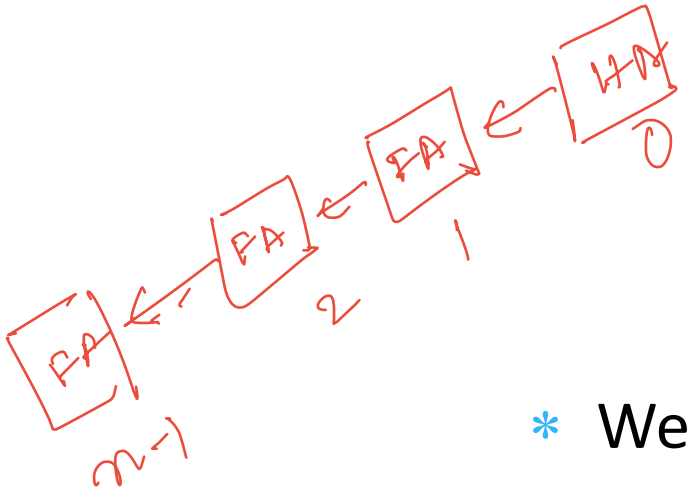
$$c_{out} = a.b + a.c_{in} + b.c_{in}$$

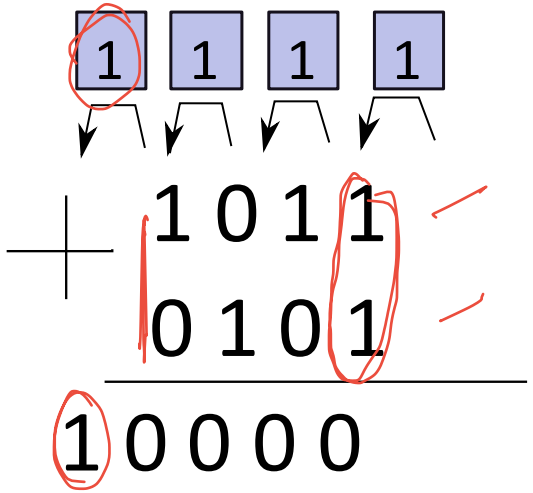
Circuit for the Full Adder





Addition of two *n* bit numbers



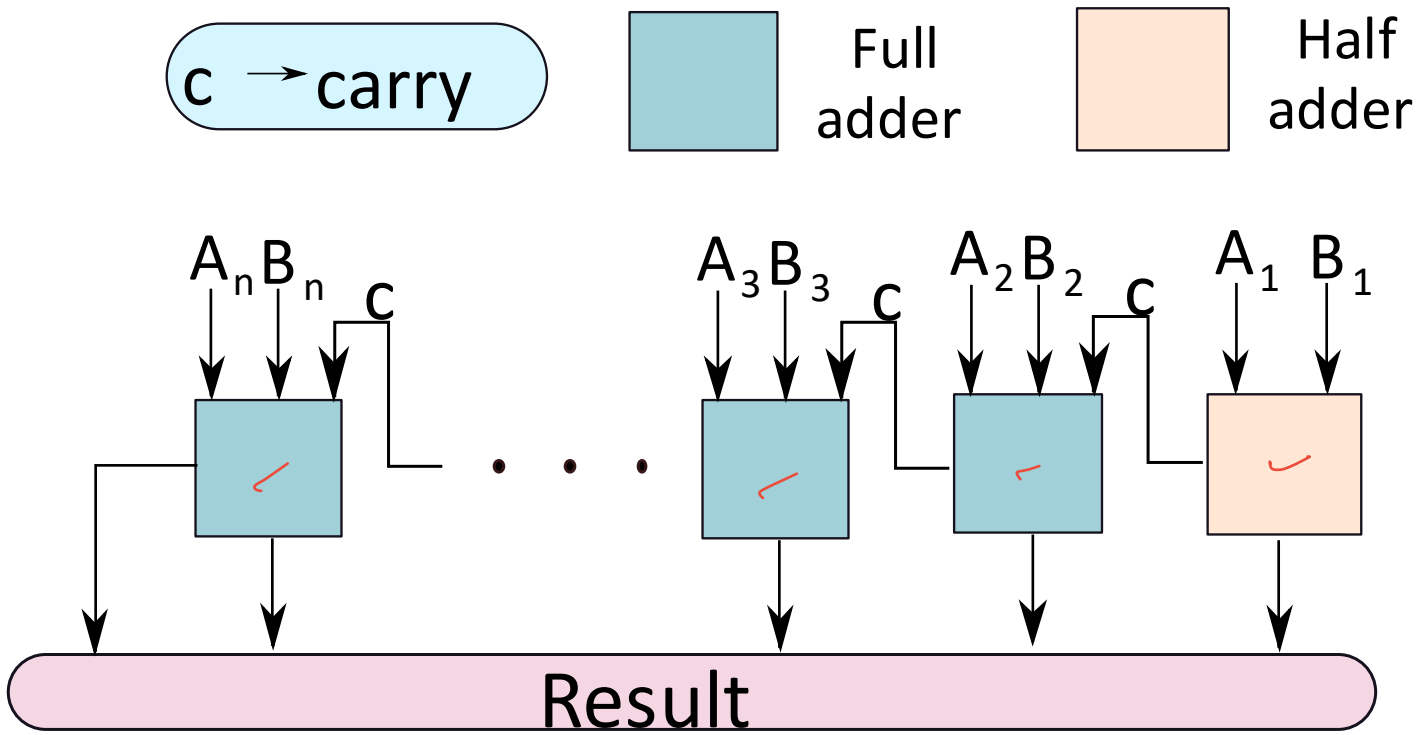


- We start from the lsb
- Add the corresponding pair of bits and the carry in
- Produce a sum bit and a carry out

Observations

* We keep adding pairs of bits, and proceed from the lsb to the msb
* If a carry is generated, we add it to the next pair of bits
* At the last step, if a carry is generated, then it becomes the msb of the result
* The carry effectively ripples through the bits

Ripple Carry Adder



t_w

t_f

$(n-1)\,t_f + t_h$

$= O(n)$

Operation of the Ripple Carry Adder

* Problem : Add A + B
* Number the bits : A_1 to A_n and B_1 to B_n
  * lsb → A_1 and B_1
  * msb → A_n and B_n
* Use a half adder to add A_1 and B_1
* Send the carry(c) to a full adder that adds : $A_2 + B_2 + c$
* Proceed in a similar manner till the msb

How long does the Ripple Carry Adder take ?

* Time :
  * Time of half adder : t_h
  * Time of full adder : t_f
  * Total Time : $t_h + (n-1)t_f$

Asymptotic Time Complexity

* Most of the time, we are primarily interested in the order of the function
* For example : we are only interested in the n^2 term in $(2n^2 + 3n + 4)$
* We do not care about the constants, and terms with smaller exponents
  * 3n and 4
* We can thus say that :
  * $2n^2 + 3n + 4$ is order of (n^2)

The O notation

* Formally :
  * We say that: f(n) = O(g(n))
  * if, $|f(n)| \leq c|g(n)|$, for all $n > n_0$. Here *c* is a positive constant.
* In simple terms:
  * Beyond a certain *n* , g(n) is greater-than-equal to a certain constant times f(n)
    * *For example, beyond 15, $(n^2 + 10n + 16) \leq 2n^2$*

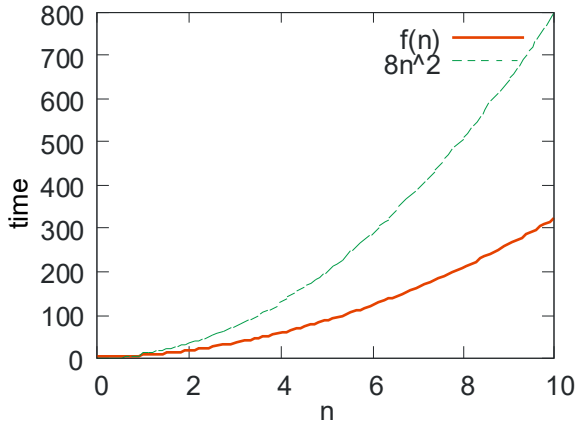
Example of the big O Notation

$f(n) = 3n^2 + 2n + 3$. *Find its asymptotic time complexity.*

Answer:

$$f(n) = 3n^2 + 2n + 3$$
$$\leq 3n^2 + 2n^2 + 3n^2 \ (n > 1)$$
$$\leq 8(n^2)$$

Hence, $f(n) = O(n^2)$.



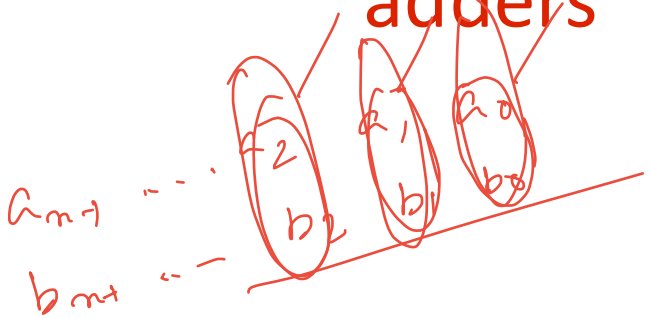
$8n^2$ *is a strict upper bound on* $f(n)$ *as shown in the figure.*

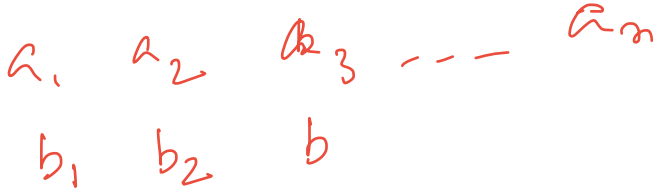
Big O Notation - II

Example:
$f(n) = 0.00001n^{100} + 10000n^{99} + 234344$. Find its asymptotic time complexity.

Answer: $f(n) = O(n^{100})$

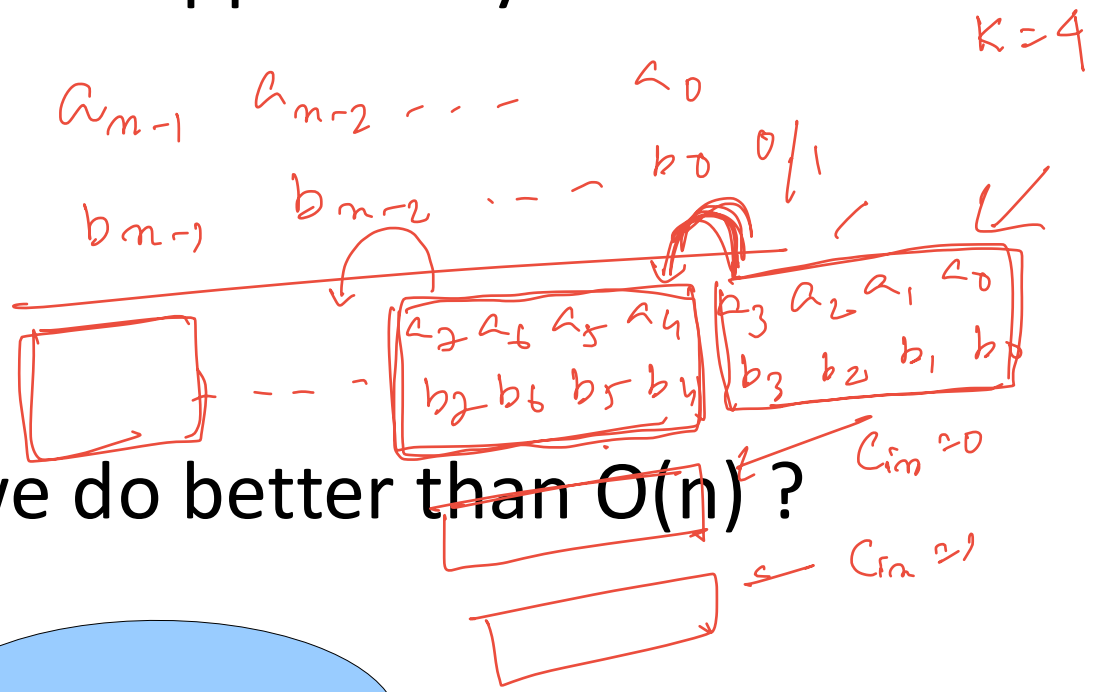
* We shall use the asymptotic time complexity metric (big O notation) to characterize the time taken by different adders





Ripple Carry Adders and Beyond

* Time complexity of a ripple carry adder :
  * O(n)

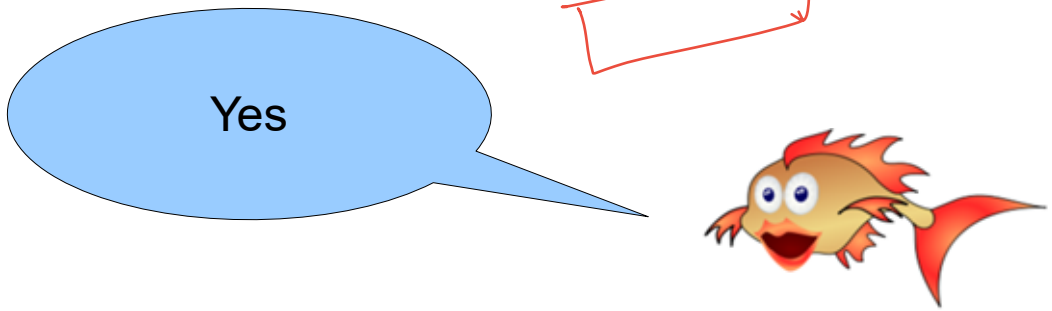


*  Can we do better than O(n) ?

Step 1
O(K)

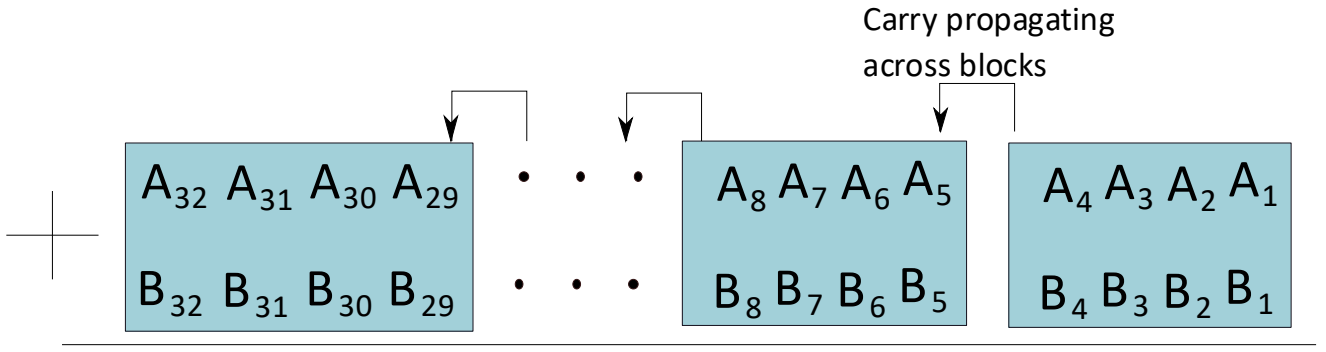
Step 2
$O\left(\frac{n}{K}\right)$

$O(K)+O\left(\frac{n}{K}\right)$



Carry Select Adder O(√n) time

* Group bits into blocks of size (k)
* If we are adding two 32 bit numbers A and B, and k = 4, then the blocks are :



* Produce the result of each block with a small ripple carry adder

Carry Select Adder - II

- In this case, the carry propagates across blocks
- Time complexity is O(n)

Idea :

- Add the numbers in each block in parallel
- Stage I : For each block, produce two results
  - Assuming an input carry of 0
  - Assuming an input carry of 1

Carry Select Adder – Stage II

- For each block we have two results available
- Result → (k sum bits), and 1 carry out bit
- Stage II
  - Start at the least significant block
    - The input carry is 0
    - Choose the appropriate result from stage I
  - We now know the input carry for the second block
    - Choose the appropriate result
    - Result contains the input carry for the third block

Carry Select Adder – Stage II

* Given the result of the second block
  * Compute the carry in for the third block
  * Choose the appropriate result
* Proceed till the last block
* At the last block (most significant positions)
  * Choose the correct result
  * The carry out value, is equal to the carry out of the entire computation.

How much time did we take ?

- Our block size is k
  - Stage I takes k units of time
- There are n/k blocks
  - Stage II takes (n/k) units of time
- Total time : (k + n/k)

$O(\sqrt{n})$

$$\frac{\partial\,(k + n/k)}{\partial k} = 0$$

$$\Rightarrow 1 - \frac{n}{k^2} = 0$$

$$\Rightarrow k = \sqrt{n}$$

Time Complexity of the Carry Select Adder

* $T = O(\sqrt{n} + \sqrt{n}) = O(\sqrt{n})$
* Thus, we have a $\sqrt{n}$ time adder

Can we do better ?