Lecture 5
Basic Addition and Counting

Half Adders and Full Adders

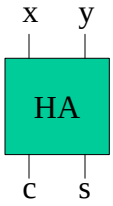
Basic building blocks for arithmetic circuits

Half Adder

Inputs: x, y

Outputs: $s = x \oplus y$

$c = x \cdot y$

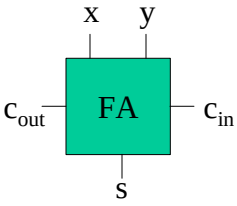


Full Adder

Inputs: x, y, c_{in}

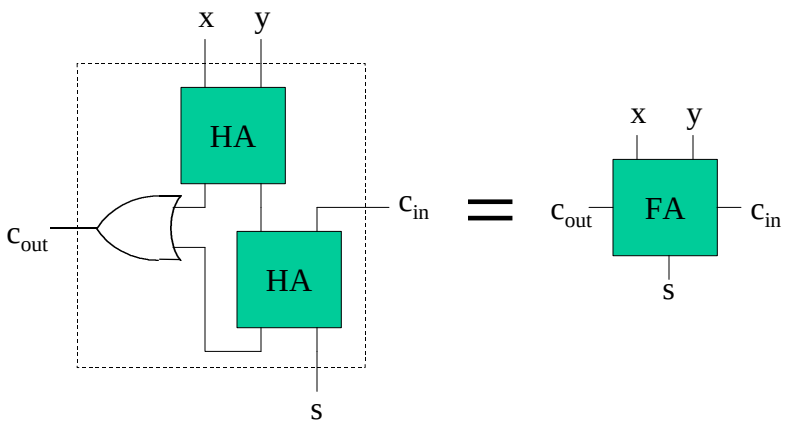
Outputs: $s = x \oplus y \oplus c_{in}$

$c_{out} = x \cdot y + (x + y) \cdot c_{in}$



Also called a (3,2) Counter

Full Adder



Mixed Positive and Negative Binary Full Adders

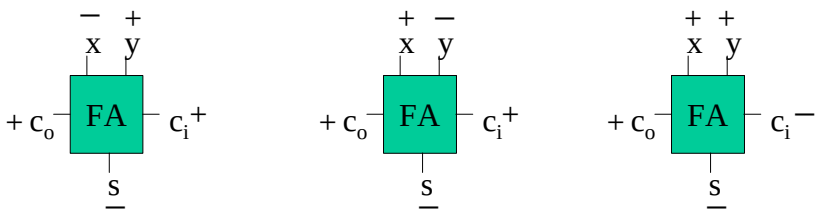
+ Digit Set = {0,1} − Digit Set = {-1,0}

+	
0	0
1	1

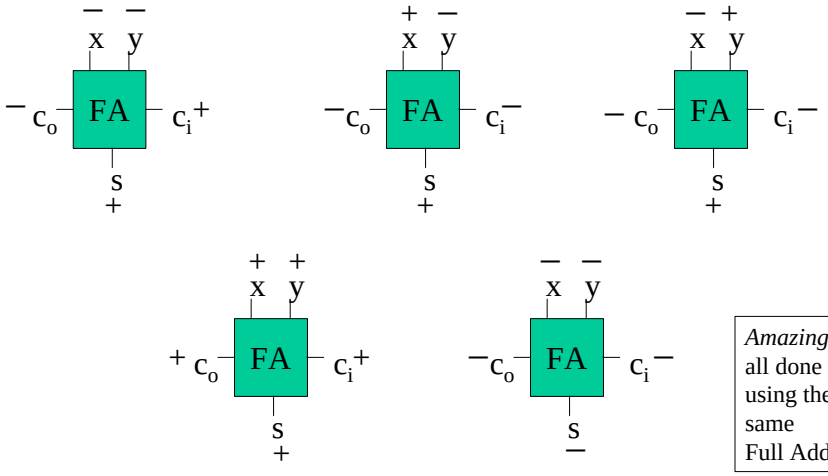
−	
−1	0
0	1

Excess-1 encoding

Digit Encodings

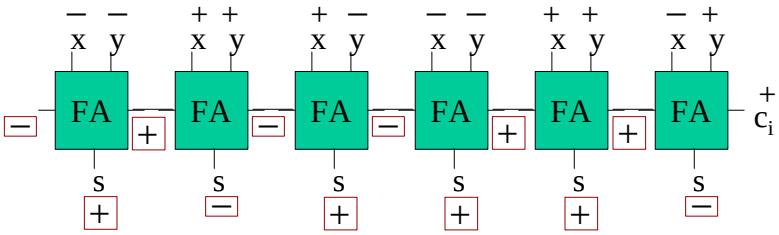


More Mixed Binary Full Adders



Amazingly all done using the same Full Adder

Mixed Binary Additions

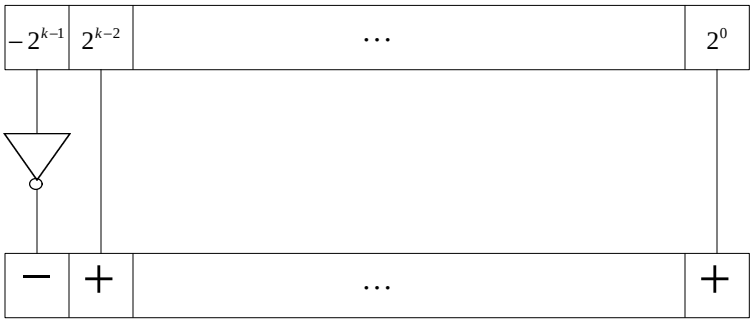


Propagate the digit sets.

You can add any combination of +/– to any other combination of +/–

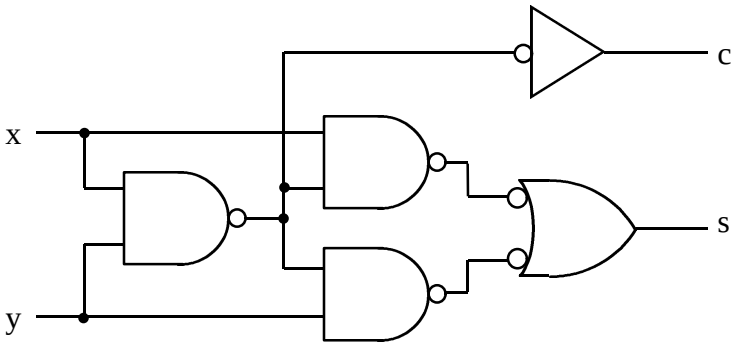
Converting Two's Comp to +/–

Two's complement number



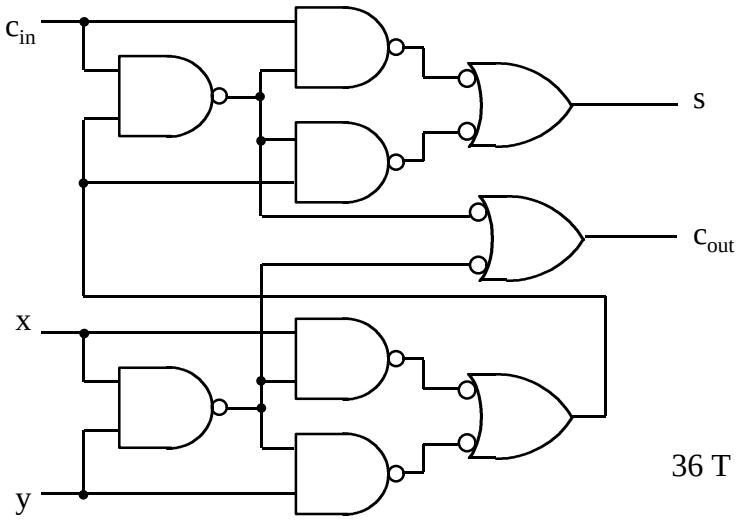
Mixed +/– digit set number

Half Adder



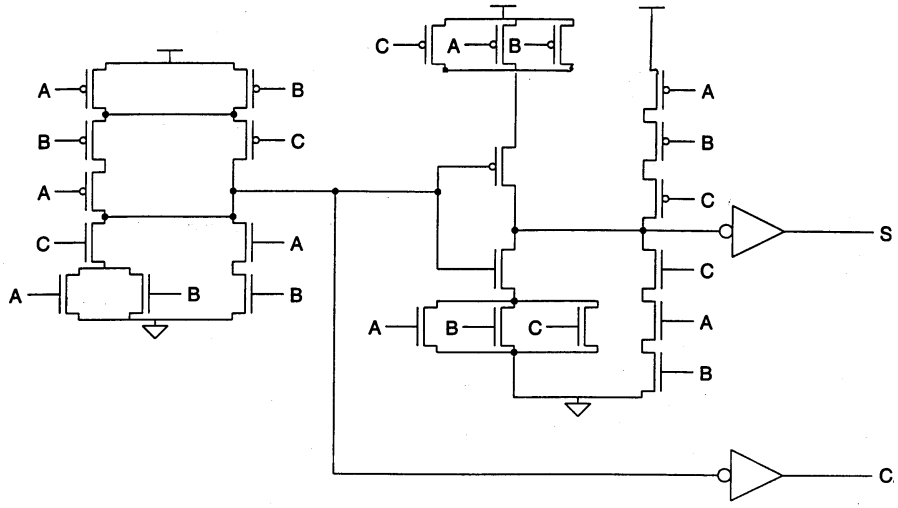
18 T

Full adder

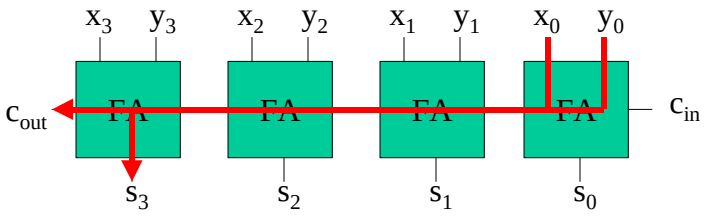


36 T

CMOS Full Adder

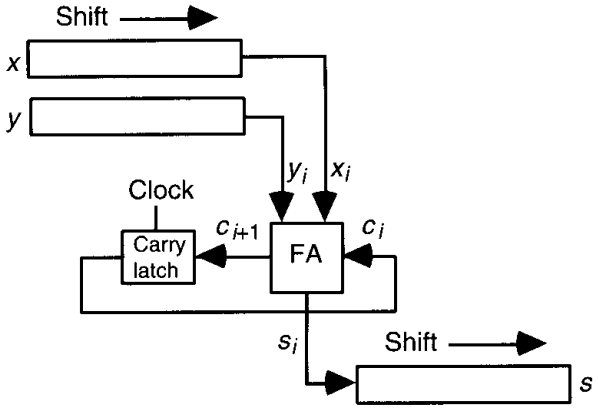


Ripple Carry Adder



Worst case delay path

Serial Addition



(a) Bit-serial adder.

Conditions, Flags, and Exceptions

- Overflow - The output cannot be represented in the format of the result.
- Sign - 1 if the result is negative, 0 if the result is positive.
- Zero - The result is zero.

Signed Overflow

- $\text{Overflow}_{\text{two's-comp}}$ = sign of result is wrong

$$= x_{k-1}y_{k-1}\bar{s}_{s-1} + \bar{x}_{k-1}\bar{y}_{k-1}s_{s-1}$$

when $c_{k-1} = 1$

$$= \underbrace{x_{k-1}y_{k-1}\bar{s}_{s-1}}_{0} + \underbrace{\bar{x}_{k-1}\bar{y}_{k-1}s_{s-1}}_{\bar{C}_k}$$

when $c_{k-1} = 0$

$$= \underbrace{x_{k-1}y_{k-1}\bar{s}_{s-1}}_{C_k} + \underbrace{\bar{x}_{k-1}\bar{y}_{k-1}s_{s-1}}_{0}$$

$$= c_k \cdot \bar{c}_{k-1} + \bar{c}_k \cdot c_{k-1}$$

$$= c_k \oplus c_{k-1}$$

Unsigned Overflow

- $\text{Overflow}_{\text{unsigned}}$ = carry out of last stage
  $= c_k$

Sign

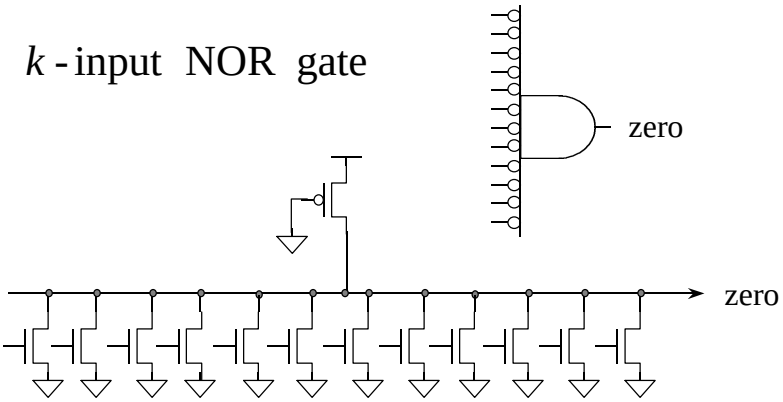
$$\text{Sign}_{signed} = 0 \text{ when positive, } 1 \text{ when negative}$$
$$= s_{k-1} \text{ when Overflow} = 0$$
$$= \bar{s}_{k-1} \text{ when Overflow} = 1$$
$$= s_{k-1} \oplus \text{ Overflow}$$
$$= s_{k-1} \oplus c_k \oplus c_{k-1}$$

$$\text{Sign}_{unsigned} = 0 \quad \text{(always positive!)}$$

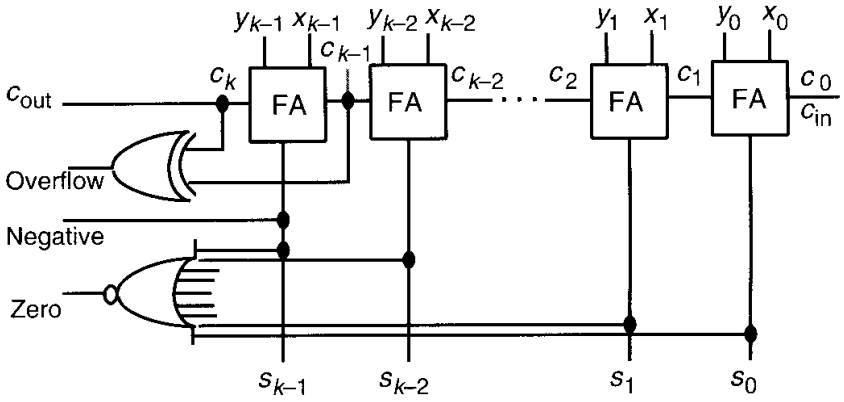
Zero

$\text{Zero} = \overline{s}_{k-1} \cdot \overline{s}_{k-2} \cdot \cdots \cdot \overline{s}_0$ (both signed, unsigned)

k-input NOR gate



Adder with Flag Detection

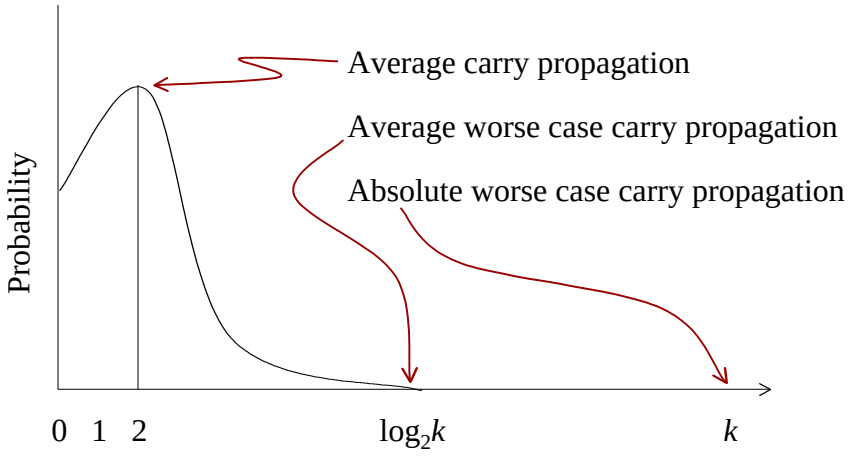


What's wrong with the diagram from the book ?

Flag Summary

Flag	*Sign*	*Unsigned*
Overflow	$c_k \oplus c_{k-1}$	c_k
Sign	$c_k \oplus c_{k-1} \oplus s_{k-1}$	0
Zero	$\overline{s}_{k-1} \cdot \overline{s}_{k-2} \cdots \overline{s}_0$	$\overline{s}_{k-1} \cdot \overline{s}_{k-2} \cdots \overline{s}_0$

Analysis of Carry Propagation



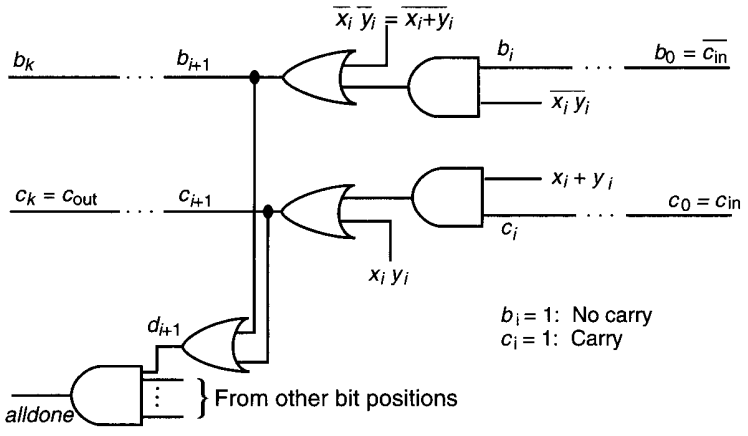
Asynchronous circuits must wait average worst case, synchronous circuits must wait absolute worst case

Carry Completion Detection

- For asynchronous arithmetic, (not useful for synchronous arithmetic)
- Carry Completion Detection gives a *done* signal when carry chain is done.
- Average time $\propto \log_2 k$
- Two rail logic:

b_i c_i	
0 0	Carry not known yet
0 1	Carry known to be 1
1 0	Carry known to be 0

Carry Completion Detection



b_i and c_i all start at 0's

Speeding Up Addition

Making Low Latency Carry Chains

- From point of view of carry propagation
  - computation of *sum* is not important.
  - At each position a carry is either
    - generated : $x_i + y_i \geq r$
    - propagated : $x_i + y_i = r - 1$, or
    - annihilated : $x_i + y_i < r - 1$

$$c_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i \oplus y_i)}_{p_i} \cdot c_i$$

$$c_{i+1} = g_i + p_i \cdot c_i$$
$$s_i = p_i \oplus c_i$$

"Carry Recurrence"

Propagation of Carry

$$\begin{aligned} c_{i+1} &= g_i + p_i c_i \\ &= g_i(1 + c_i) + p_i c_i \\ &= g_i + \underbrace{(g_i + p_i)}_{t_i} \cdot c_i \\ &= g_i + t_i c_i \end{aligned}$$

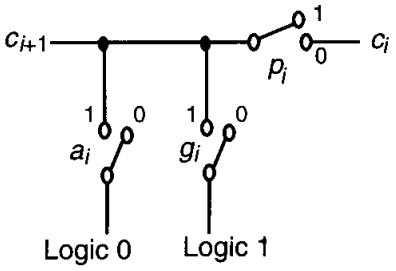
$$t_i = x_i + y_i$$

Propagation of Inverse Carry

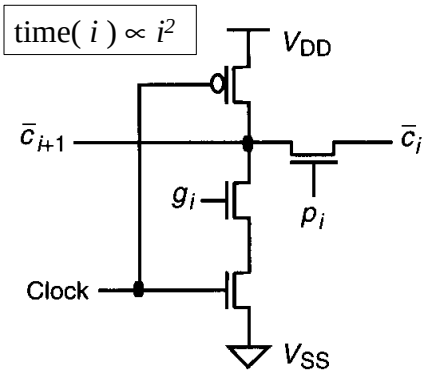
$$\begin{aligned}\overline{c}_{i+1} &= \overline{g_i + p_i c_i} \\ &= \overline{g}_i \cdot (\overline{p}_i + \overline{c}_i) \\ &= \overline{g}_i \overline{p}_i + \overline{g}_i \overline{c}_i \\ &= a_i + (a_i + p_i) \cdot \overline{c}_i \\ &= a_i + p_i \overline{c}_i\end{aligned}$$

Manchester Carry Chains

time(i) $\propto i^2$



(a) Conceptual representation.



(b) Possible CMOS realization.