Digital Logic Design: a rigorous approach ©

Chapter 15: Addition

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

May 21, 2020

Book Homepage:
http://www.eng.tau.ac.il/~guy/Even-Medina

Definition of a binary adder

Definition

$\textsc{adder}(n)$ - a binary adder with input length n is a combinational circuit specified as follows.

Input: $A[n-1:0], B[n-1:0] \in \{0,1\}^n$, and $C[0] \in \{0,1\}$.

Output: $S[n-1:0] \in \{0,1\}^n$ and $C[n] \in \{0,1\}$.

Functionality:

$$\langle \vec{S} \rangle + 2^n \cdot C[n] = \langle \vec{A} \rangle + \langle \vec{B} \rangle + C[0]. \quad (1)$$

Addition terminology:

- addends: $\langle \vec{A} \rangle = \sum_{i=1}^{n-1} A[i] \cdot 2^i$, and $\langle \vec{B} \rangle = \sum_{i=1}^{n-1} B[i] \cdot 2^i$
- carry-in bit : $C[0]$
- sum: $\langle \vec{S} \rangle$
- carry-out bit: $C[n]$

binary adder definition (cont)

Definition

$\text{ADDER}(n)$ - a binary adder with input length n is a combinational circuit specified as follows.

Input: $A[n-1:0], B[n-1:0] \in \{0,1\}^n$, and $C[0] \in \{0,1\}$.

Output: $S[n-1:0] \in \{0,1\}^n$ and $C[n] \in \{0,1\}$.

Functionality:

$$\langle \vec{S} \rangle + 2^n \cdot C[n] = \langle \vec{A} \rangle + \langle \vec{B} \rangle + C[0]. \quad (2)$$

Claim ($\text{ADDER}(n)$ is well defined)

For every $A[n-1:0], B[n-1:0] \in \{0,1\}^n$, *and* $C[0] \in \{0,1\}$, *there exist* $S[n-1:0] \in \{0,1\}^n$ *and* $C[n] \in \{0,1\}$ *such that*

$$\langle \vec{S} \rangle + 2^n \cdot C[n] = \langle \vec{A} \rangle + \langle \vec{B} \rangle + C[0]$$

Full Adder

An ADDER(1) is called a full adder.

Definition (Full-Adder)

FA - a Full-Adder is a combinational circuit with 3 inputs $x, y, z \in \{0, 1\}$ and 2 outputs $c, s \in \{0, 1\}$ that satisfies:

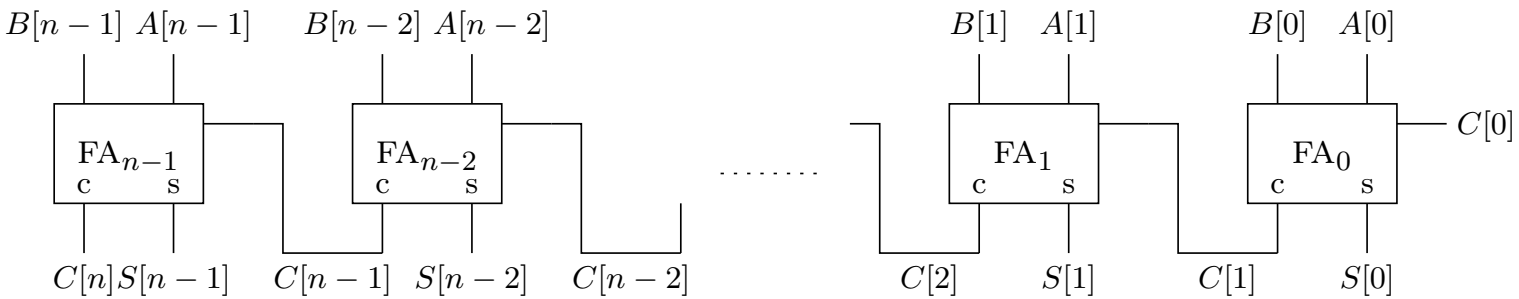
$$2c + s = x + y + z.$$

Terminology: s -sum output, c -carry-out output.

Claim

$$s = x \oplus y \oplus z,$$
$$c = (x \cdot y) \vee (y \cdot z) \vee (x \cdot z).$$

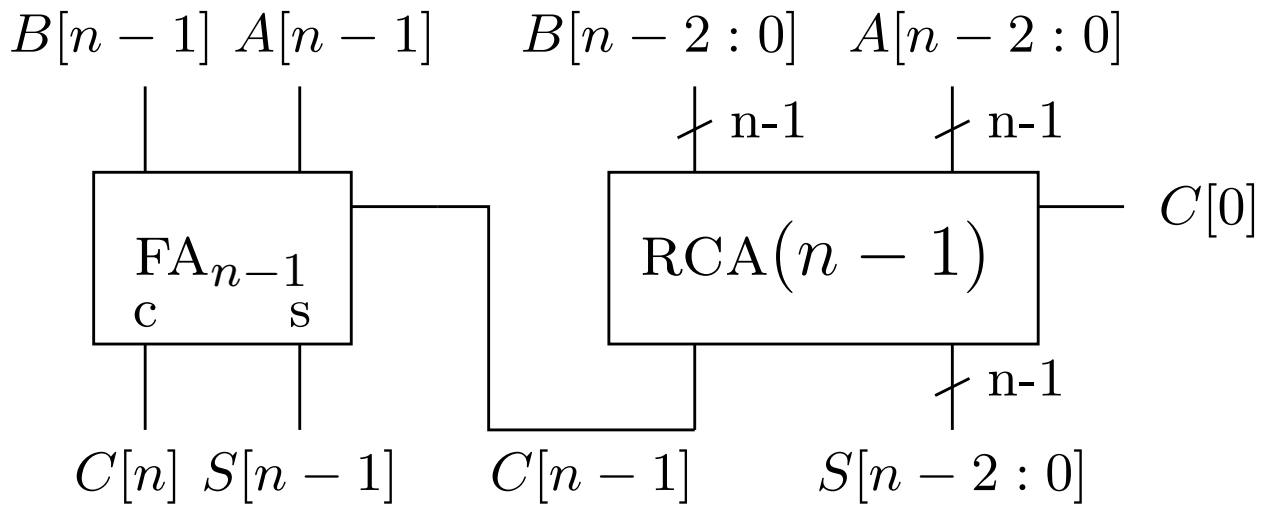
Ripple Carry Adder $\mathrm{RCA}(n)$



- same addition algorithm that we use for adding numbers by hand.
- row of n Full-Adders connected in a chain.
- the weight of every signal is two to the power of its index. (Do not confuse weight here with Hamming weight. Weight means here the value in binary representation.)

Recursive definition of $\mathrm{RCA}(n)$

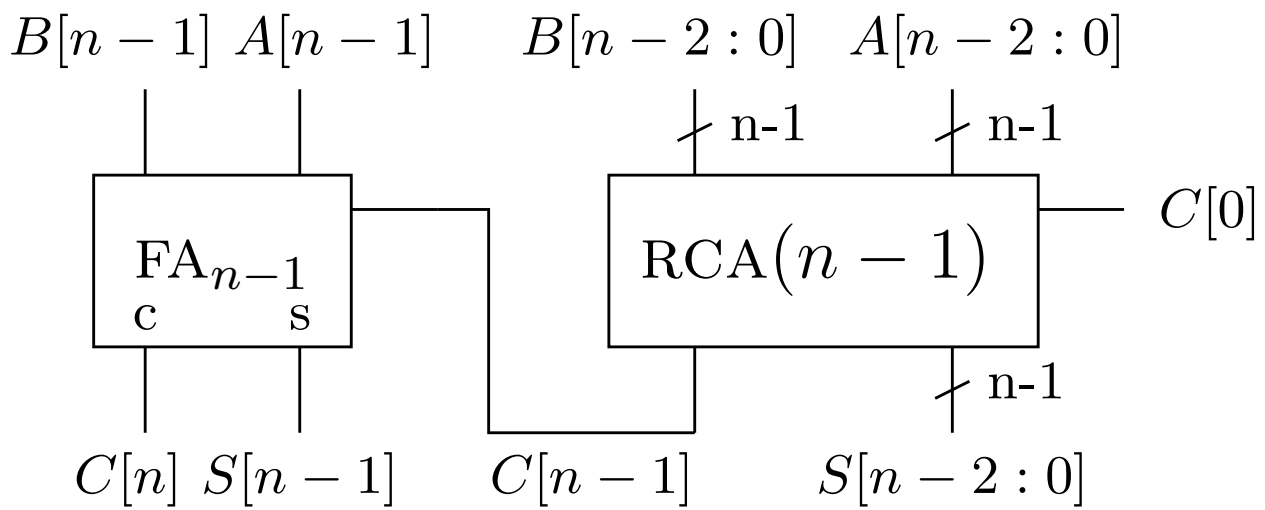
Basis: an $\mathrm{RCA}(1)$ is simply a Full-Adder.
Reduction Step:



$\mathrm{RCA}(n)$ - correctness

Claim

$\mathrm{RCA}(n)$ *is a correct implementation of* $\mathrm{ADDER}(n)$.



Delay and cost analysis

The cost of an $\mathrm{RCA}(n)$ satisfies:

$$c(\mathrm{RCA}(n)) = n \cdot c(\mathrm{FA}) = \Theta(n).$$

The delay of an $\mathrm{RCA}(n)$ satisfies

$$d(\mathrm{RCA}(n)) = n \cdot d(\mathrm{FA}) = \Theta(n).$$

Clock rates in modern microprocessors correspond to the delay of 15-20 gates (in more aggressive designs, the critical paths are even shorter). Most microprocessors easily add 32-bit numbers within one clock cycle (high-end microprocessors even add 100-bit number in a cycle). Obviously, adders in such microprocessors are not Ripple Carry Adders.

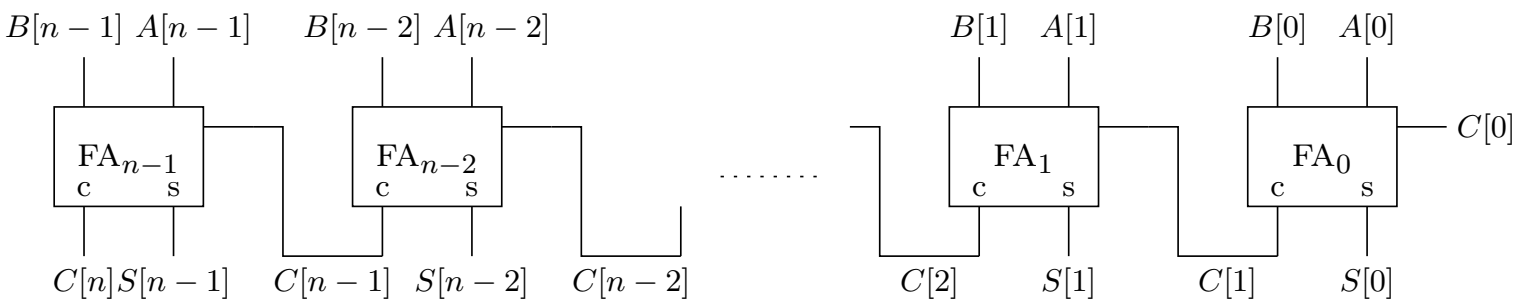
Carry bits

We now define the carry bits associated with the addition

$$\langle A[n-1:0]\rangle + \langle B[n-1:0]\rangle + C[0] = \langle S[n-1:0]\rangle + 2^n \cdot C[n]$$

Definition

The carry bits $C[n:0]$ are defined as the values of the stable signals $C[n:0]$ in an $\mathrm{RCA}(n)$.



This definition is well defined in light of the Simulation Theorem of combinational circuits.

Cone of adder outputs

The correctness proof of $\mathrm{RCA}(n)$ implies that, for every $0 \leq i \leq n-1$,

$$\langle A[i:0] \rangle + \langle B[i:0] \rangle + C[0] = 2^{i+1} \cdot C[i+1] + \langle S[i:0] \rangle.$$

Hence, for every $0 \leq i \leq n-1$:

$$C[i+1] = 1 \iff \langle A[i:0] \rangle + \langle B[i:0] \rangle + C[0] \geq 2^{i+1}$$
$$\langle S[i:0] \rangle = \mathrm{mod}(\langle A[i:0] \rangle + \langle B[i:0] \rangle + C[0], 2^{i+1}).$$

Claim

For each $0 \leq i \leq n-1$, the cone of Boolean functions corresponding to $C[i+1]$ and $S[i]$ consists of $2i+3$ inputs corresponding to $A[i:0], B[i:0]$, and $C[0]$.

Lower bounds

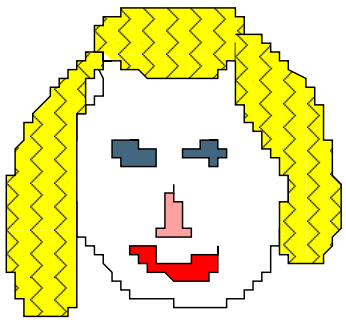
Claim

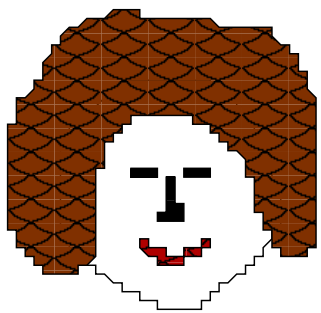
Let A denote a combinational circuit that implements an $\textsc{adder}(n)$. *If the fan-in in C is at most 2, then*

$$c(A) \geq 2n,$$
$$d(A) \geq \log_2(2n+1).$$

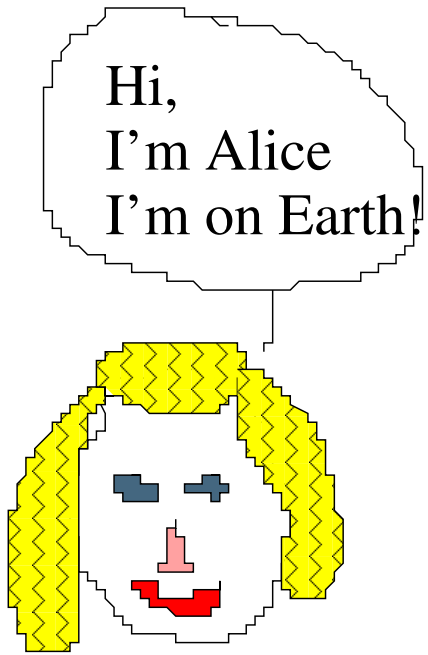
Compare with the cost and delay of $\textsc{rca}(n)$.

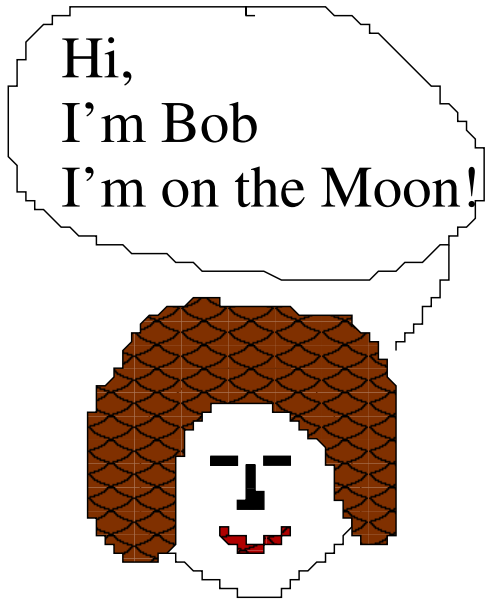
Conditional Sum Adder - motivation



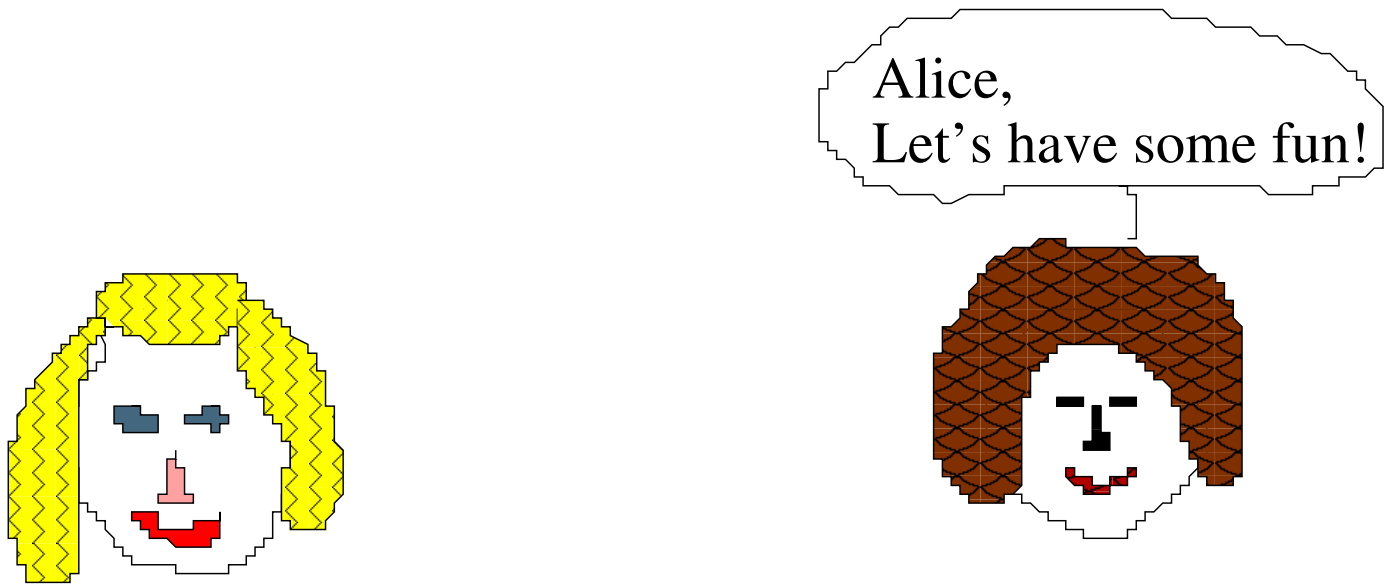


Conditional Sum Adder - motivation

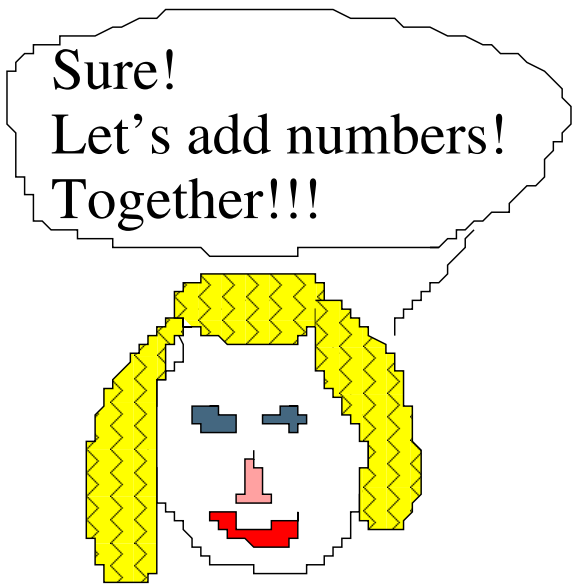


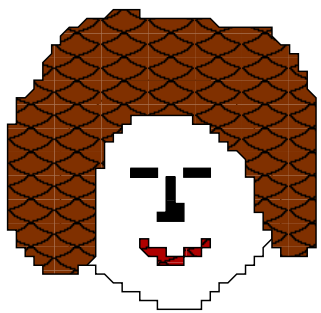


Conditional Sum Adder - motivation

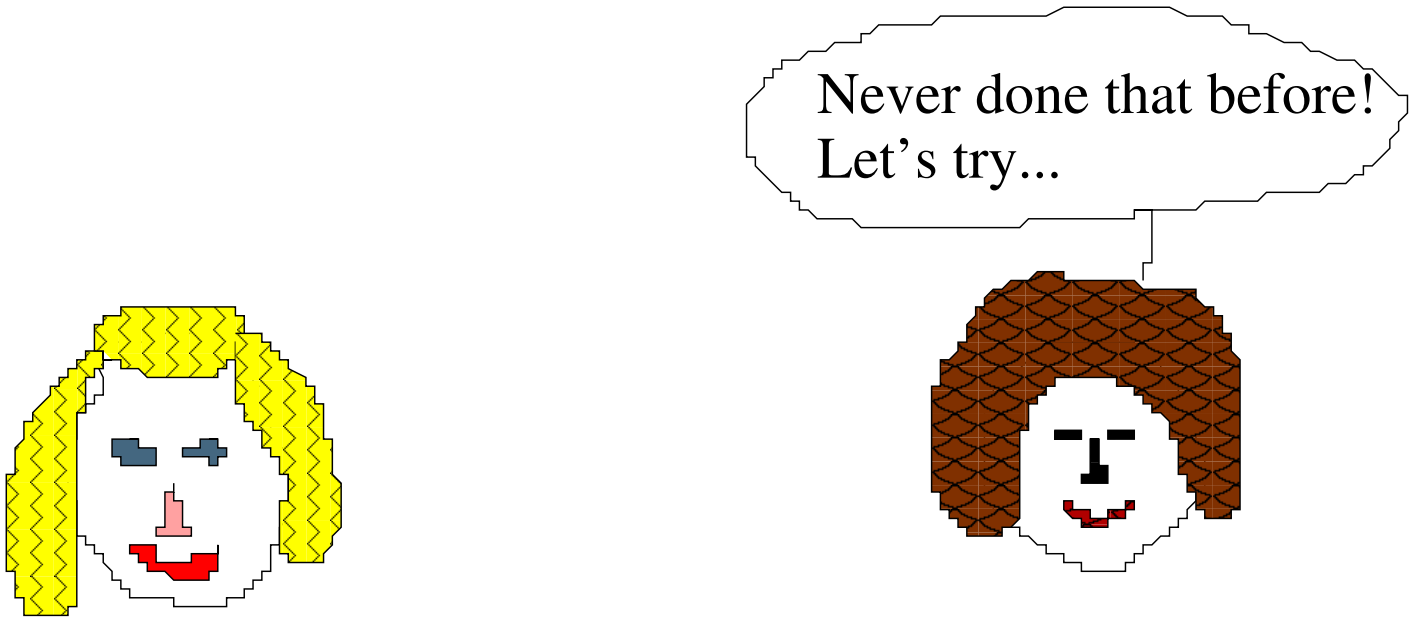


Conditional Sum Adder - motivation

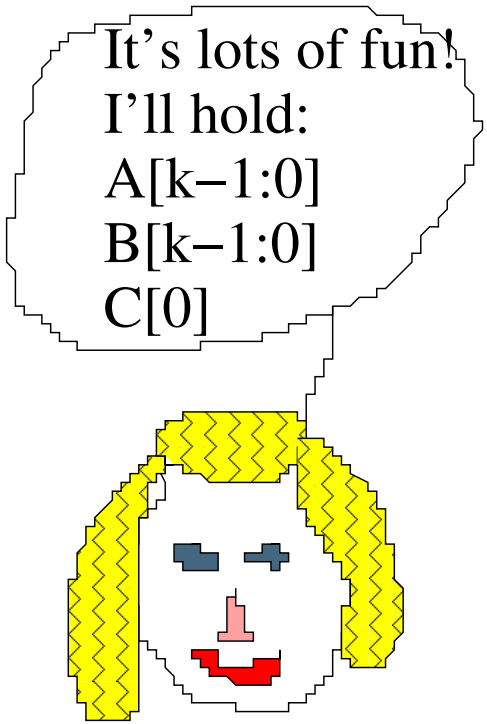


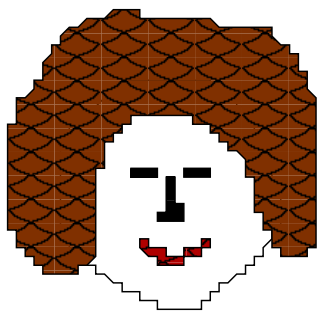


Conditional Sum Adder - motivation

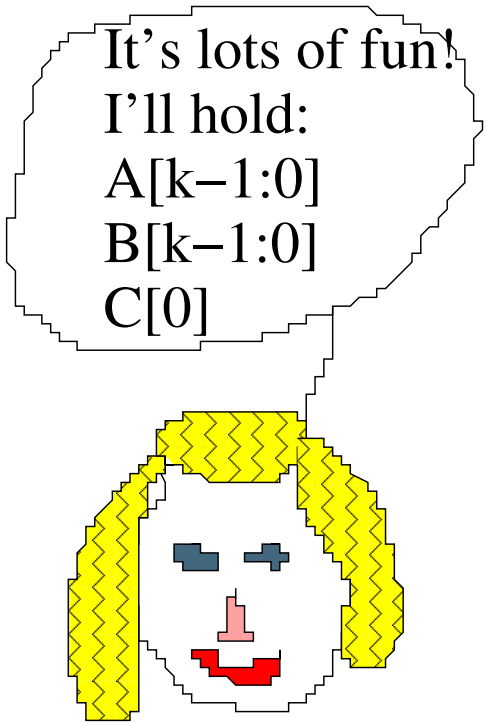


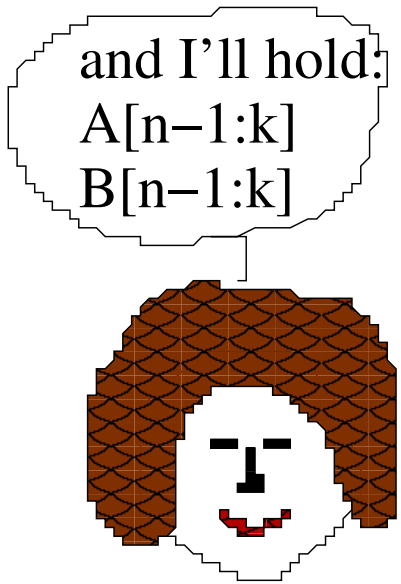
Conditional Sum Adder - motivation



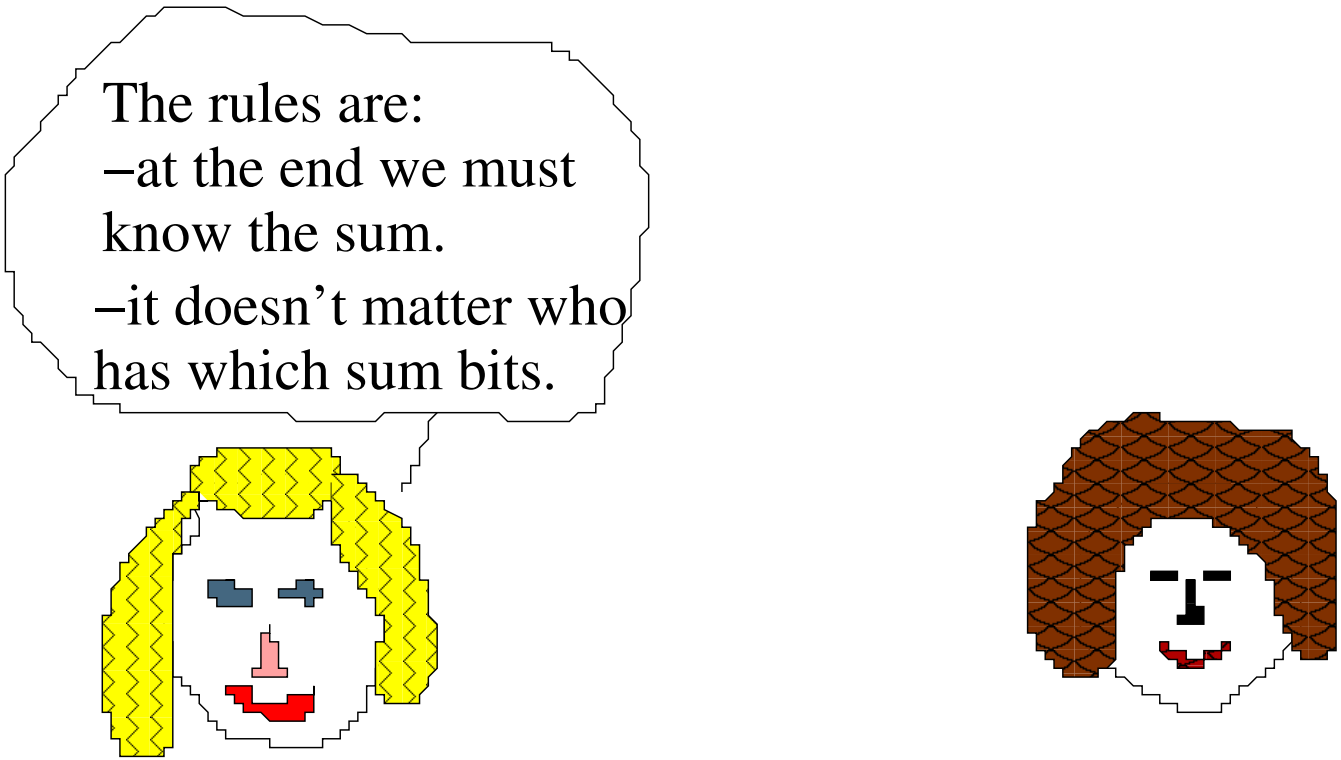


Conditional Sum Adder - motivation

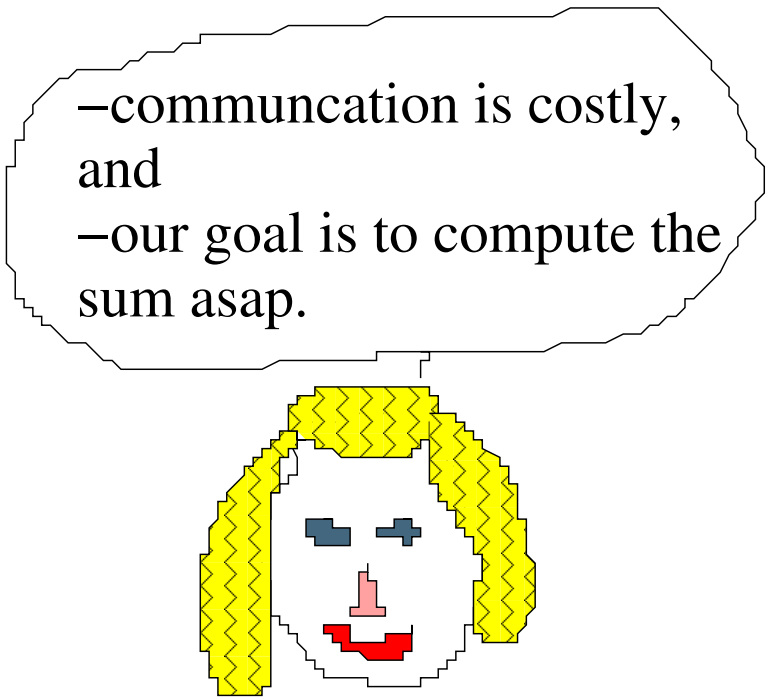


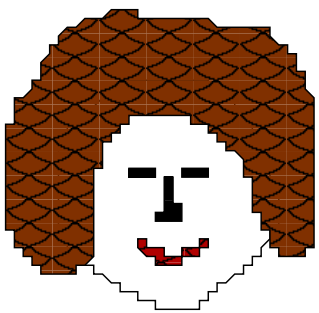


Conditional Sum Adder - motivation

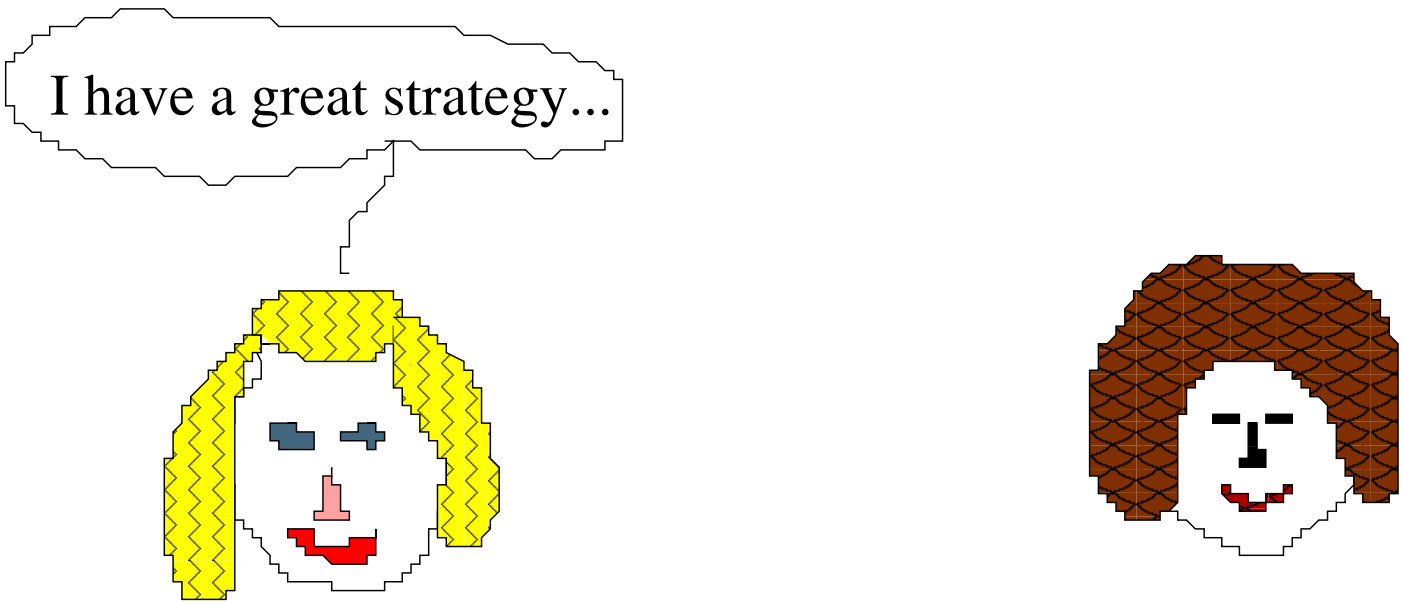


Conditional Sum Adder - motivation

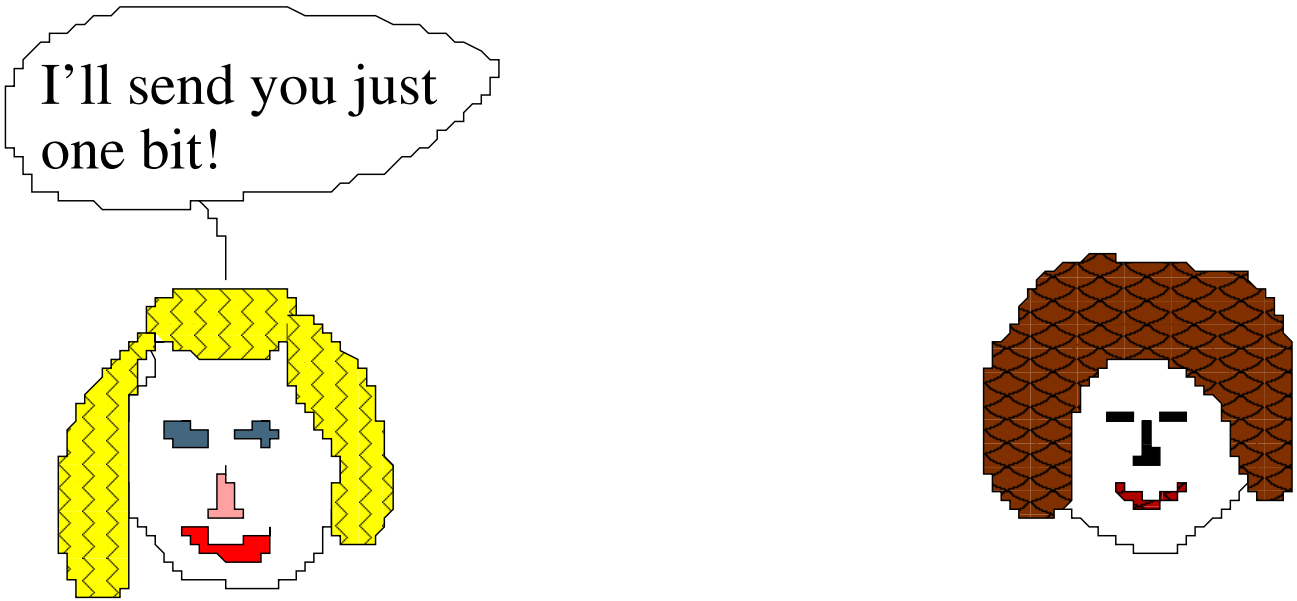




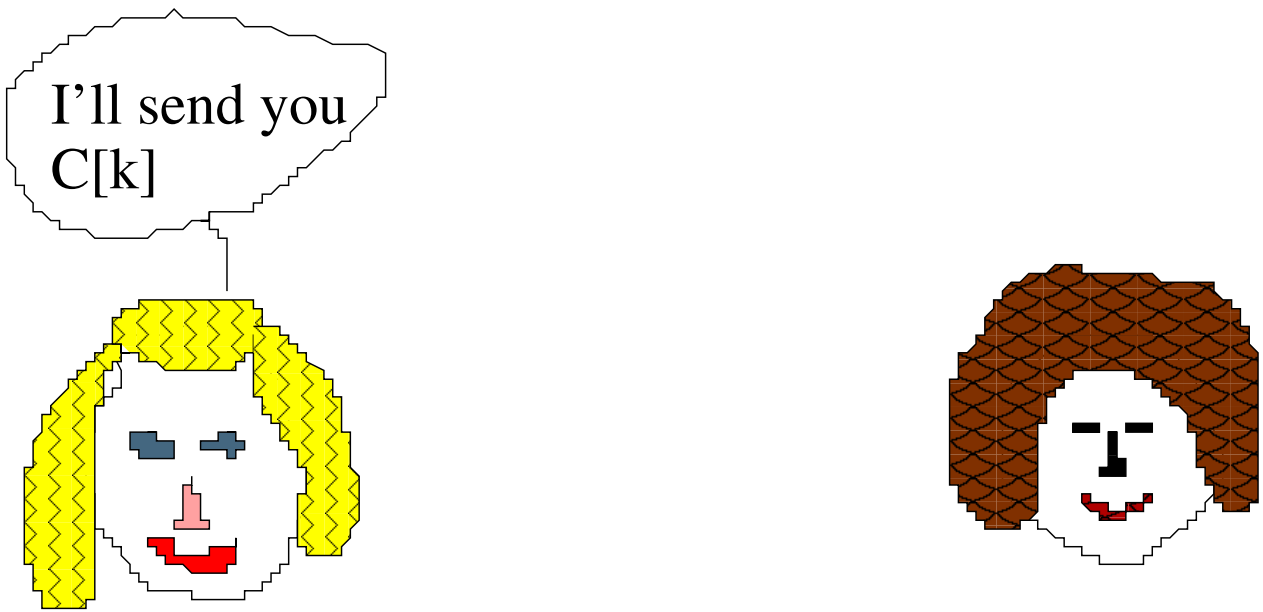
Conditional Sum Adder - motivation



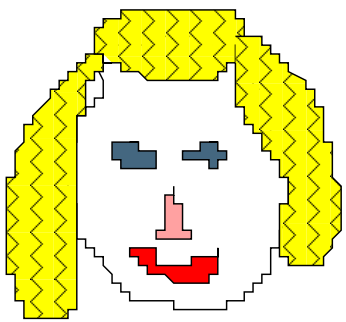
Conditional Sum Adder - motivation

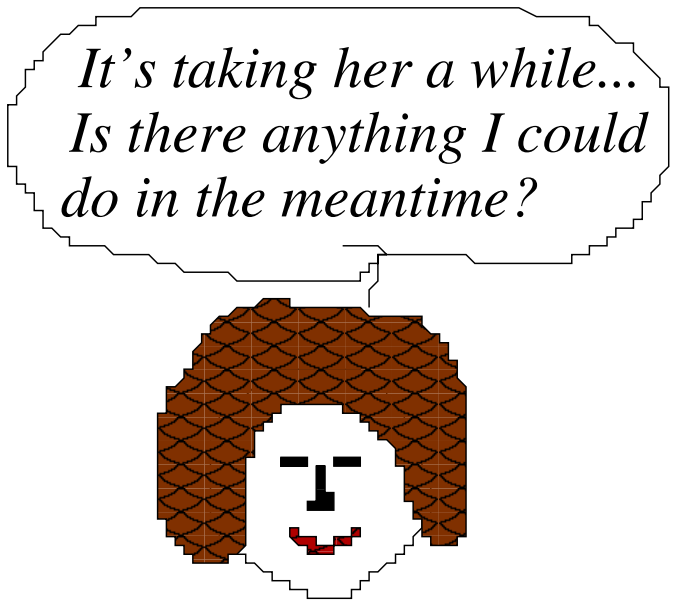


Conditional Sum Adder - motivation



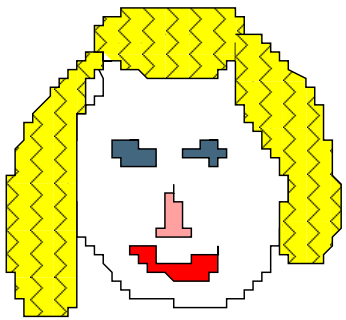
Conditional Sum Adder - motivation

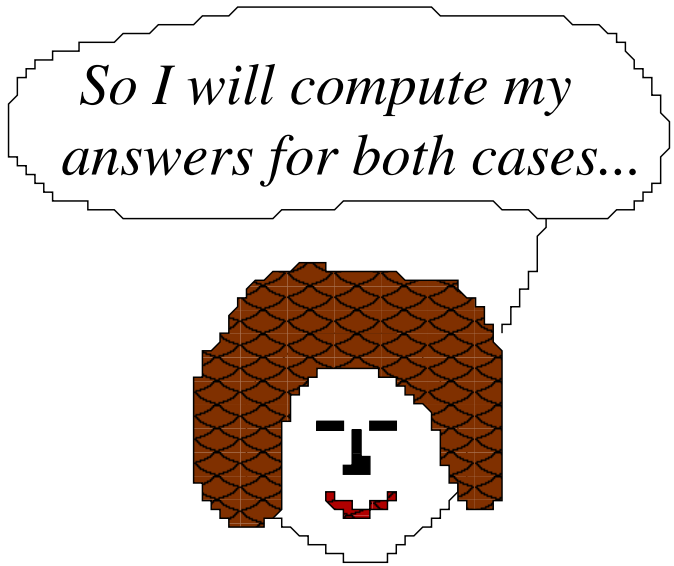




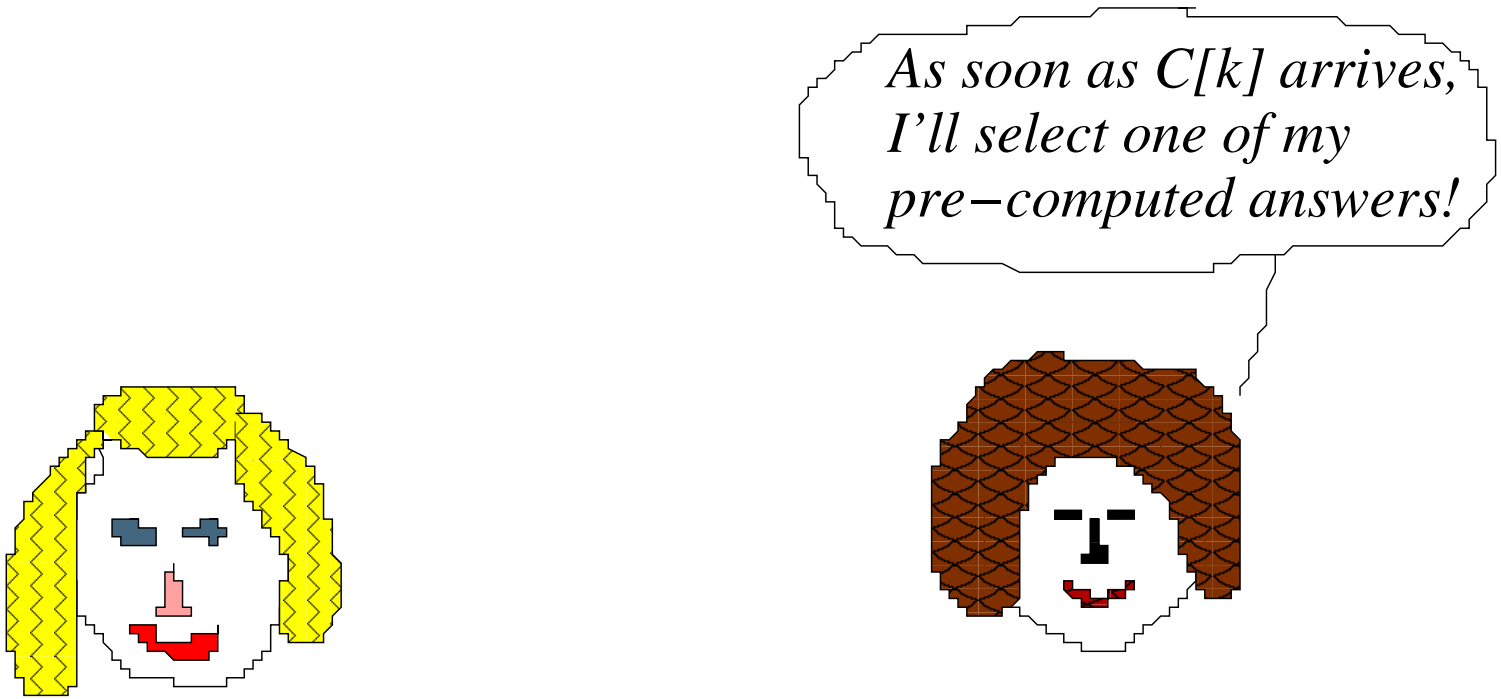
Conditional Sum Adder - motivation

Conditional Sum Adder - motivation

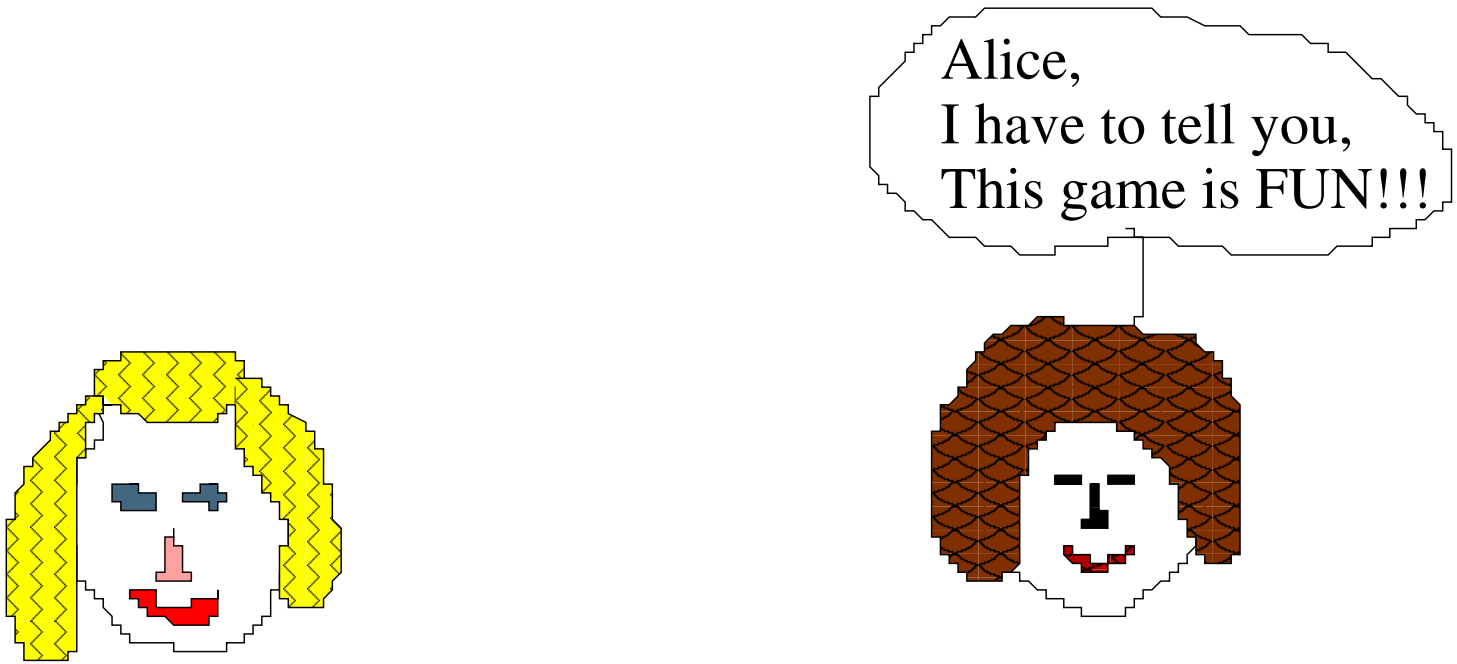




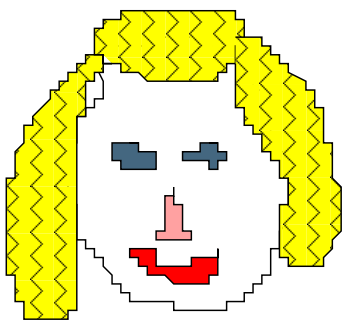
Conditional Sum Adder - motivation

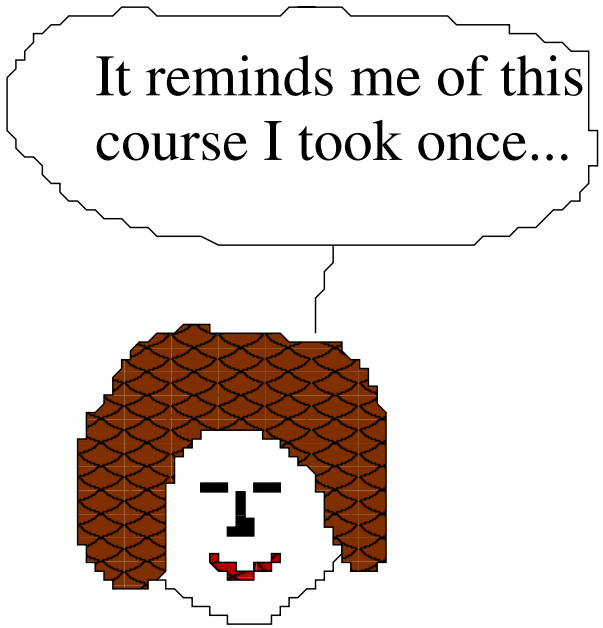


Conditional Sum Adder - motivation



Conditional Sum Adder - motivation

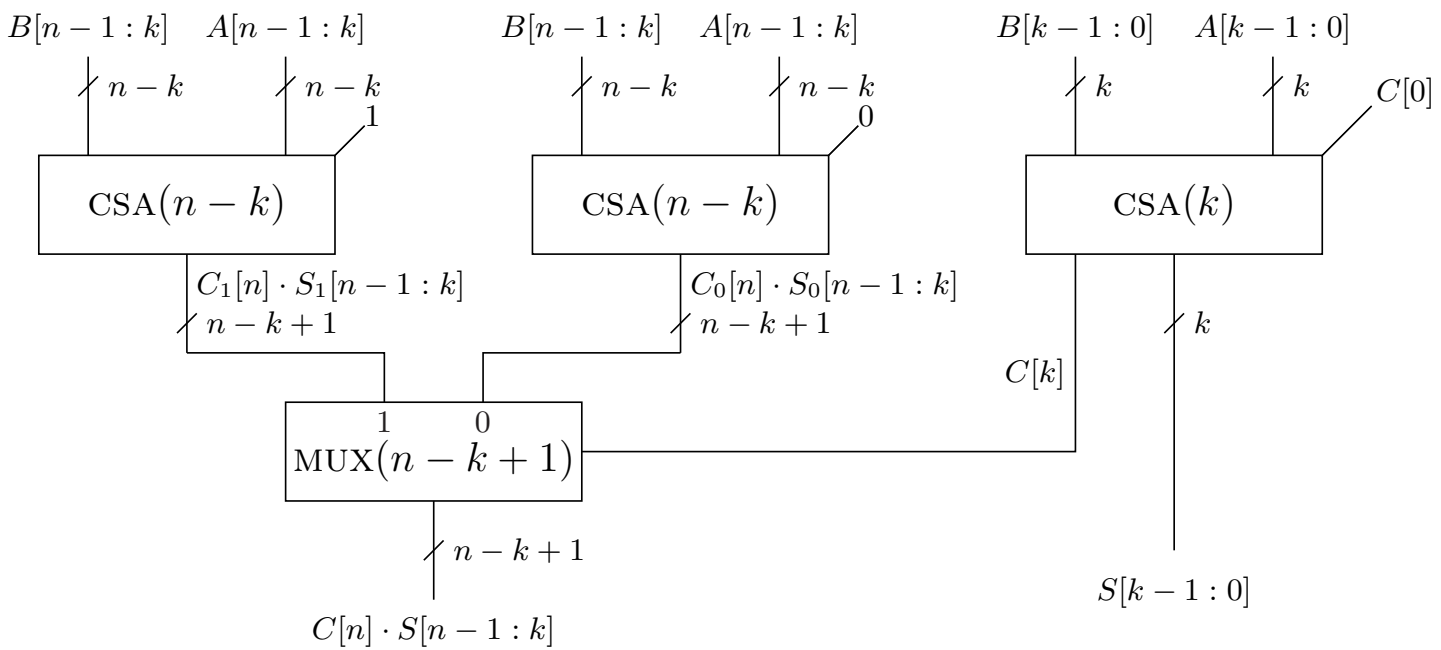




Conditional Sum Adder $\mathrm{CSA}(n)$

basis: A $\mathrm{CSA}(1)$ is simply a Full-Adder.

reduction step:



Claim

The $\mathrm{CSA}(n)$ *is a correct* $\mathrm{ADDER}(n)$ *design.*

Delay analysis

To simplify the analysis we assume that $n = 2^\ell$. To optimize the delay, we use $k = n/2$.
Let $d(\text{FA})$ denote the delay of a Full-Adder. The delay of a $\text{CSA}(n)$ satisfies the following recurrence:

$$d(\text{CSA}(n)) = \begin{cases} d(\text{FA}) & \text{if } n = 1 \\ d(\text{CSA}(n/2)) + d(\text{MUX}) & \text{otherwise.} \end{cases}$$

Hence, the delay of a $\text{CSA}(n)$ is

$$\begin{aligned} d(\text{CSA}(n)) &= \ell \cdot d(\text{MUX}) + d(\text{FA}) \\ &= \Theta(\log n). \end{aligned}$$

Cost analysis.

Let $c(\text{FA})$ denote the cost of a Full-Adder. The cost of a $\text{CSA}(n)$ satisfies the following recurrence:

$$c(\text{CSA}(n)) = \begin{cases} c(\text{FA}) & \text{if } n = 1 \\ 3 \cdot c(\text{CSA}(n/2)) + (n/2 + 1) \cdot c(\text{MUX}) & \text{otherwise.} \end{cases}$$

the solution of this recurrence is $c(\text{CSA}(n)) = \Theta\left(n^{\log_2 3}\right)$.

- $\log_2 3 \approx 1.58$, so a $\text{CSA}(n)$ is costly.
- but delay is logarithmic!
- the $\text{CSA}(n)$ design uses three half-size adders (easy to use).

Compound Adder

Definition

COMP-ADDER(n) - a Compound Adder with input length n is a combinational circuit specified as follows.

Input: $A[n-1:0], B[n-1:0] \in \{0,1\}^n$.

Output: $S[n:0], T[n:0] \in \{0,1\}^{n+1}$.

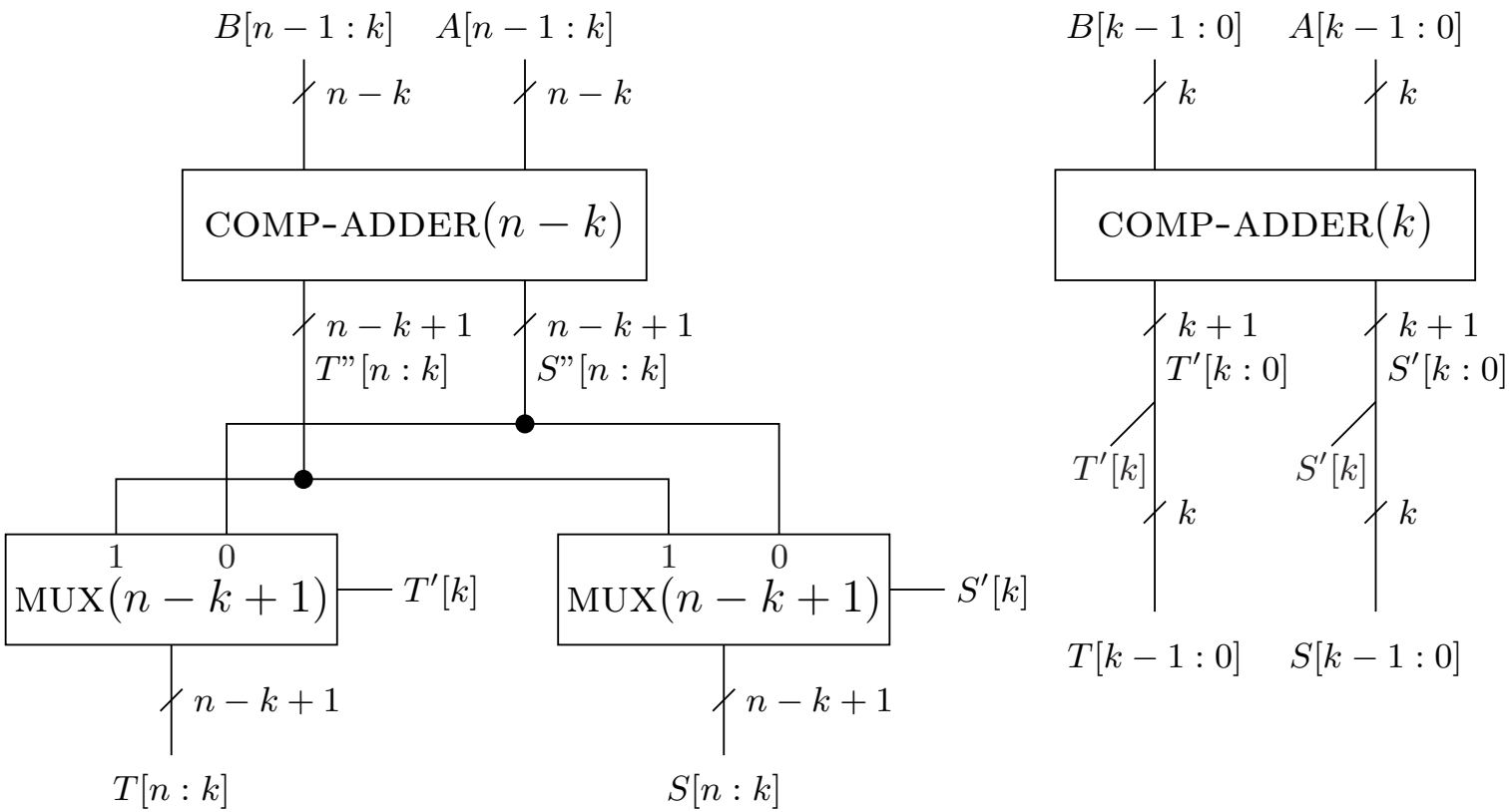
Functionality:

$$\langle \vec{S} \rangle = \langle \vec{A} \rangle + \langle \vec{B} \rangle$$
$$\langle \vec{T} \rangle = \langle \vec{A} \rangle + \langle \vec{B} \rangle + 1.$$

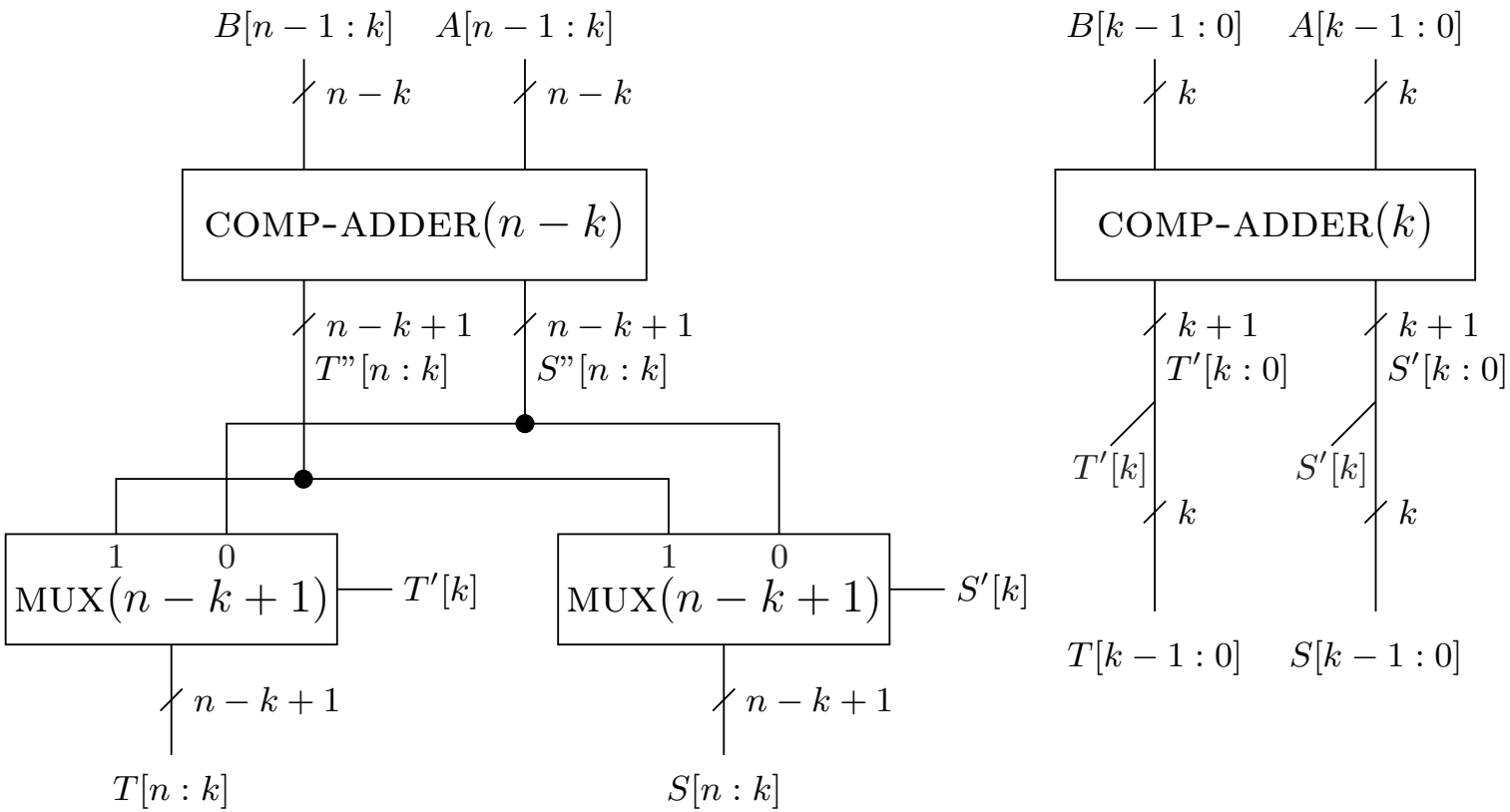
Note that a Compound Adder does not have carry-in input. To simplify notation, the carry-out bits are denoted by $S[n]$ for the sum and by $T[n]$ for the incremented sum.

COMP-ADDER(n) - Implementation

basis: $n = 1$, we simply use a Full-Adder and a Half-Adder.
reduction step:



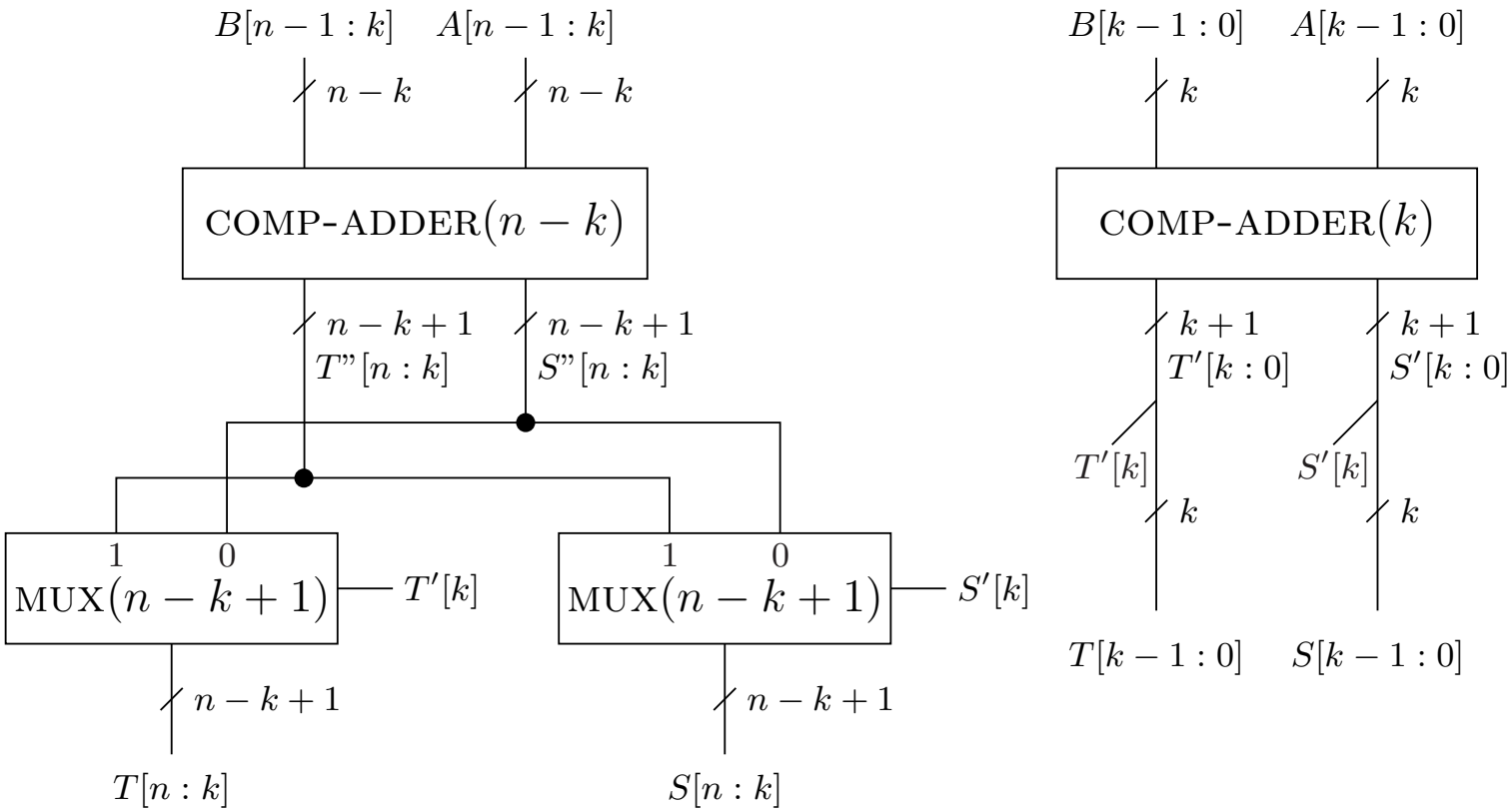
COMP-ADDER(n) - example



Example

Consider a COMP-ADDER(4) with input $A[3:0] = 0110$ and $B[3:0] = 1001$.

COMP-ADDER(n) - example



Claim

The COMP-ADDER(n) *design is a correct adder.*

Delay analysis

To simplify the analysis we assume that $n = 2^\ell$. To optimize the delay, we use $k = n/2$.
The delay of a $\textsc{comp-adder}(n)$ satisfies the following recurrence:

$$d(\textsc{comp-adder}(n)) = \begin{cases} d(\textsc{fa}) & \text{if } n = 1 \\ d(\textsc{comp-adder}(n/2)) + d(\textsc{mux}) & \text{otherwise.} \end{cases}$$

Hence,

$$\begin{aligned} d(\textsc{comp-adder}(n)) &= \ell \cdot d(\textsc{mux}) + d(\textsc{fa}) \\ &= \Theta(\log n). \end{aligned}$$

Cost analysis

The cost of a $\textsc{comp-adder}(n)$ satisfies the following recurrence:

$$c(\textsc{comp-adder}(n)) = \begin{cases} c(\textsc{fa}) + c(\textsc{ha}) \\ 2 \cdot c(\textsc{comp-adder}(n/2)) + (n/2+1) \cdot c(\textsc{mux}) \end{cases}$$

Hence, $c(\textsc{comp-adder}) = \Theta(n \log n)$.

SURPRISE!!! $c(\textsc{comp-adder}(n)) \ll c(\textsc{csa}(n))$.

Reductions between sum and carry bits

The correctness of $\mathrm{RCA}(n)$ implies that, for every $0 \leq i \leq n-1$,

$$S[i] = A[i] \oplus B[i] \oplus C[i] \tag{3}$$

By xoring $C[i] \oplus S[i]$ to both sides, we obtain,

$$C[i] = A[i] \oplus B[i] \oplus S[i]\,. \tag{4}$$

Summary

- defined binary addition.
- Three adder designs: Ripple Carry Adder, Conditional Sum Adder, Compound Adder.
- The problems of computing the sum bits and the carry bits are equivalent with respect to a constant-time linear-cost reduction. Since the cost of every adder is $\Omega(n)$ and the delay is $\Omega(\log n)$, we regard the problems of computing the sum bits and the carry bits as equivalently hard.
- Design methodology: divide & conquer.
- Surprise! $\textsc{comp-adder}(n)$ is much cheaper asymptotically than a $\textsc{csa}(n)$.
- Left to show: an adder with linear cost and logarithmic delay....