Variable Block Adder (1A)

•

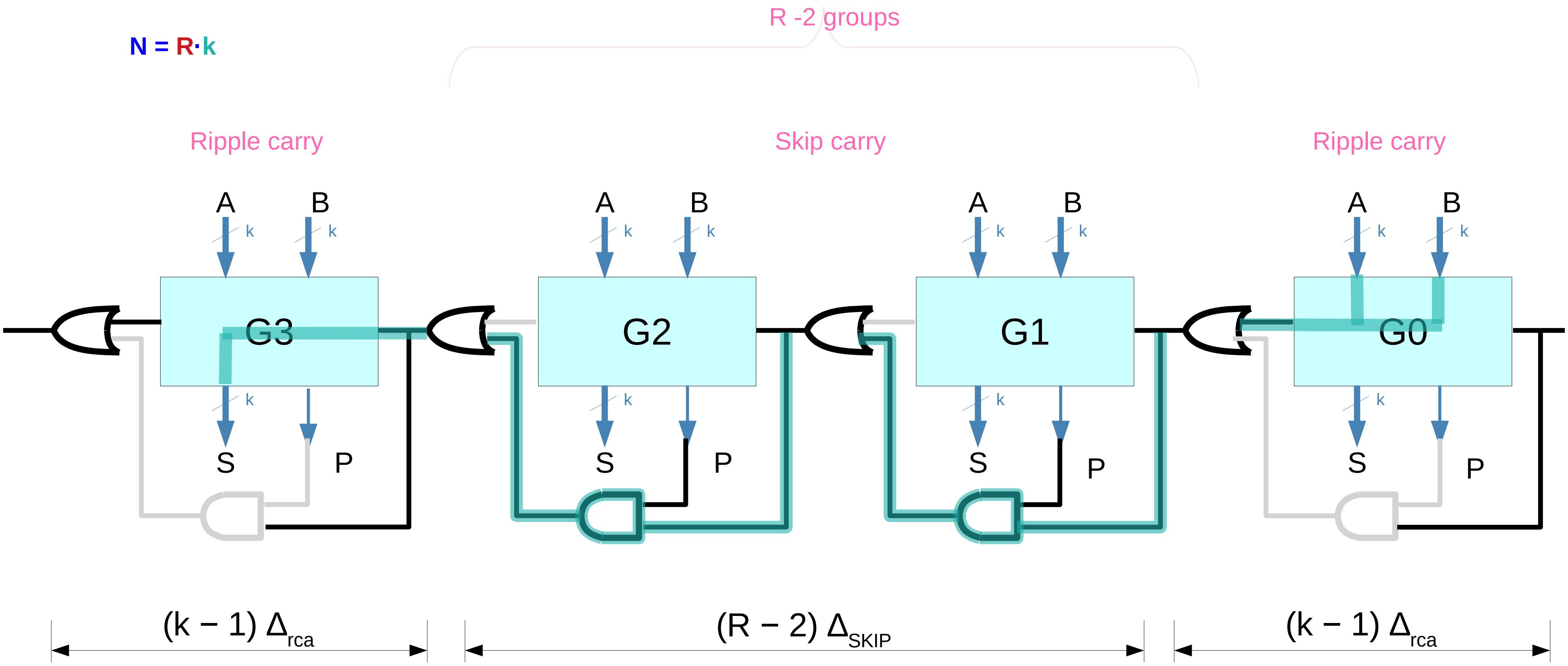
•

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Carry Skip Adder

$N = R \cdot k$



Any kill or generate condition results in divided (broken) critical paths

All FA’s in R-2 groups must have the propagate condition

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Carry Skip Adder

The maximal delay Δ of a Carry Skip Adder is encountered
when carry is generated in the least-significant bit position,

- rippling through *k-1* bit positions,
- skipping over *R-2 = N/k-2* groups in the middle,
- rippling to the *k-1* bits of most significant group and
- being assimilated in the *N-th* bit position to produce the sum S_N :

$$\begin{aligned}\Delta_{CSA} &= (k-1)\,\Delta_{rca} + (R-2)\,\Delta_{SKIP} + (k-1)\,\Delta_{rca} \\ &= 2\,(k-1)\,\Delta_{rca} + (R-2)\,\Delta_{SKIP} \\ &= 2\,(k-1)\,\Delta_{rca} + (N/k-2)\,\Delta_{SKIP}\end{aligned}$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

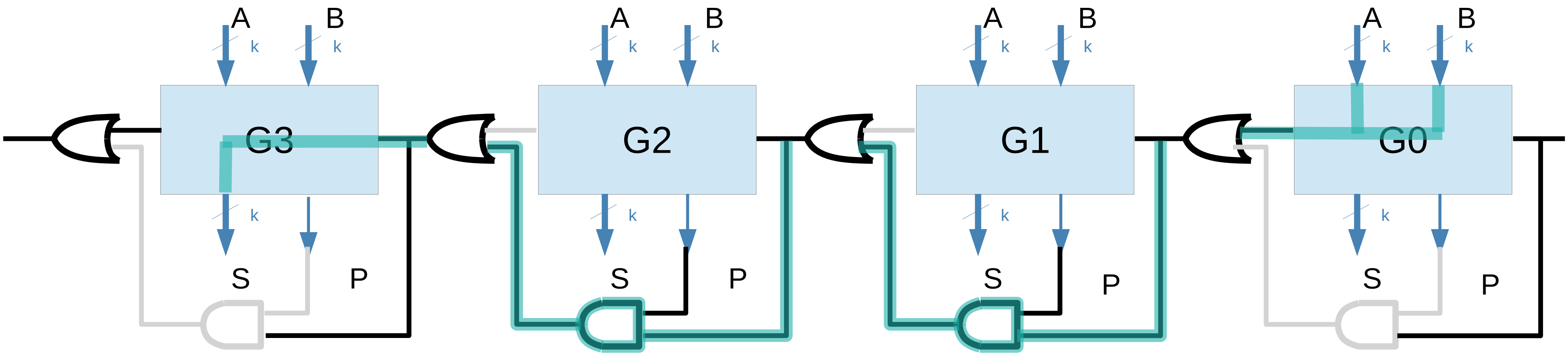
Carry Skip Adder

$$\begin{aligned}\Delta_{CSA} &= (k-1)\,\Delta_{rca} + (R-2)\,\Delta_{SKIP} + (k-1)\,\Delta_{rca} \\ &= 2\,(k-1)\,\Delta_{rca} + (R-2)\,\Delta_{SKIP} \\ &= 2\,(k-1)\,\Delta_{rca} + (N/k-2)\,\Delta_{SKIP}\end{aligned}$$

Carry Skip Adder is faster than RCA at the expense of a few relatively simple modifications.

$N = R \cdot k$

The delay is still linearly dependent on the size of the adder N, however this linear dependence is reduced by a factor of *1/k*



Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Variable Block

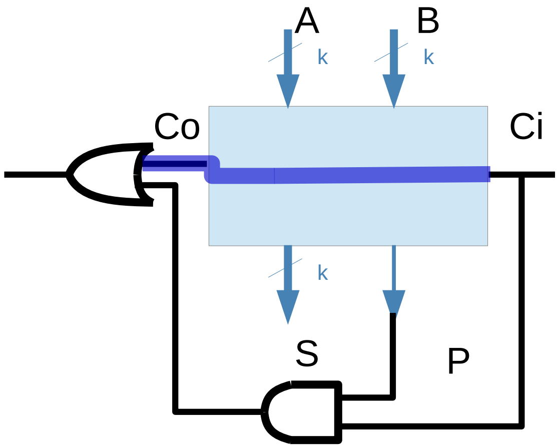
however, unlike the carry select structure,
the variable block adder must also worry about
the delay from the Cin input
through the block's ripple chain

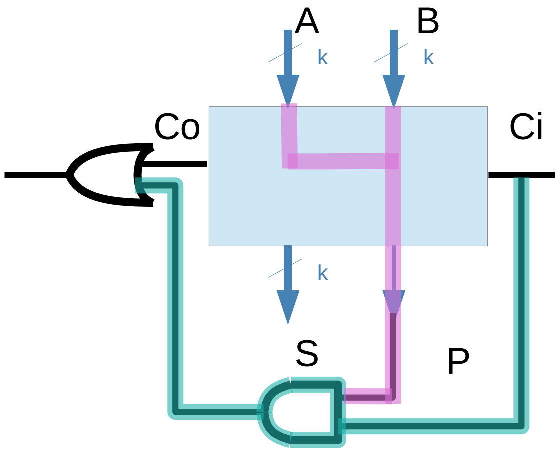
Thus, after the carry chain passes
the midpoint of the logic,
the blocks begin decreasing in length.

This balances the path delays in the system
and improves performance

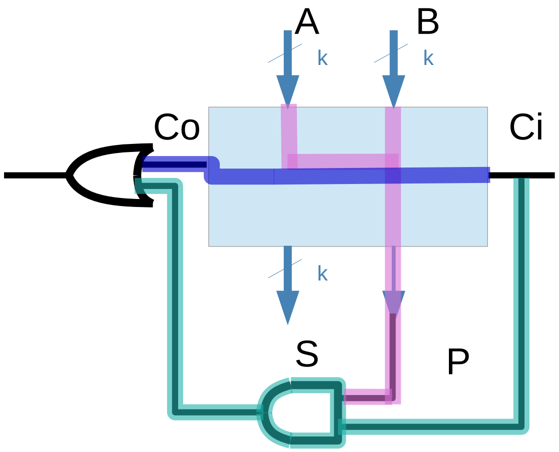
The division of the overall structure into blocks depends on
the details of the logic structure and
the length of the entire computation

https://en.wikipedia.org/wiki/Carry-lookahead_adder

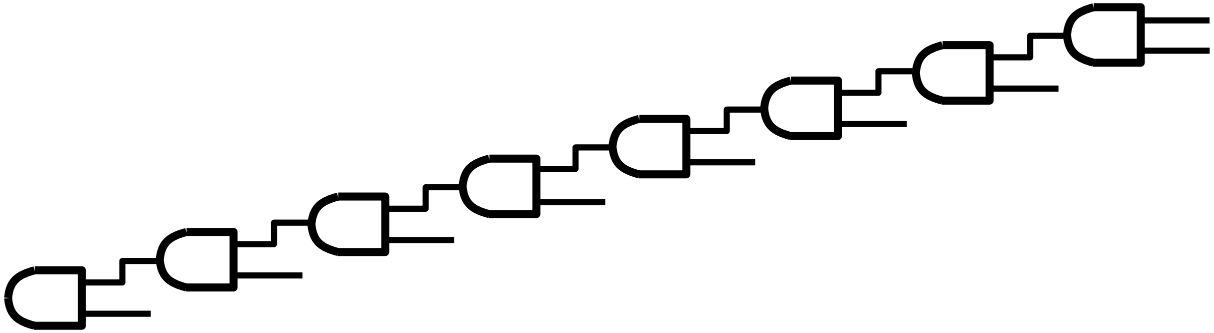


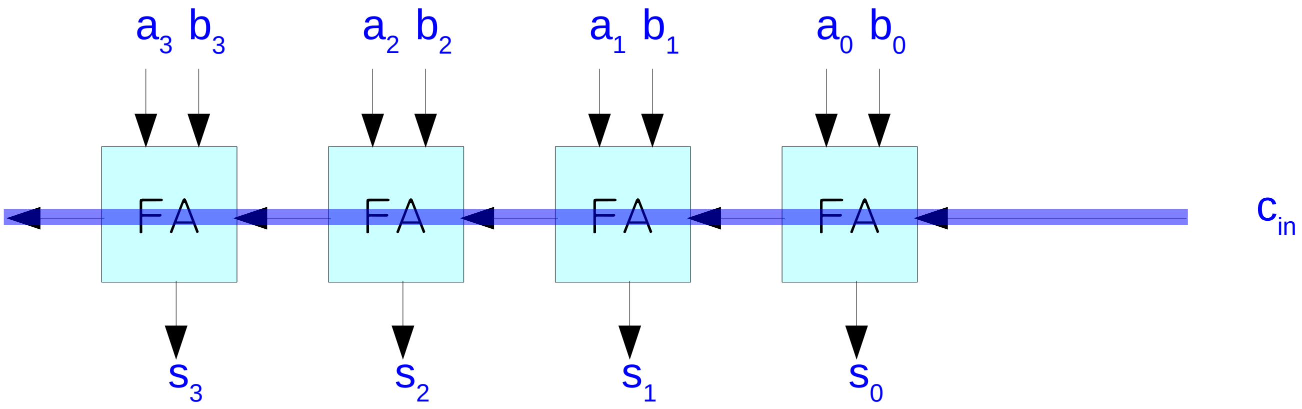


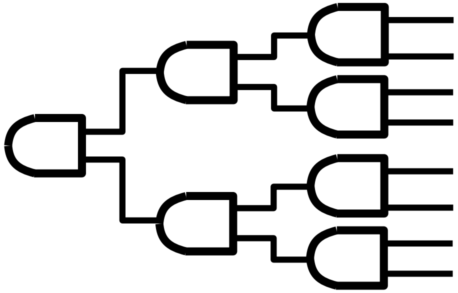
Carry Skip Adder



$P = p_i \bullet p_{i+1} \bullet p_{i+2} \bullet \ldots\ldots \bullet p_{i+k-1}$

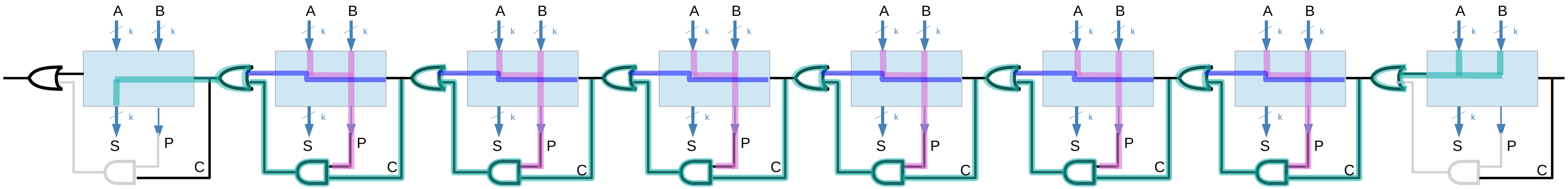






Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

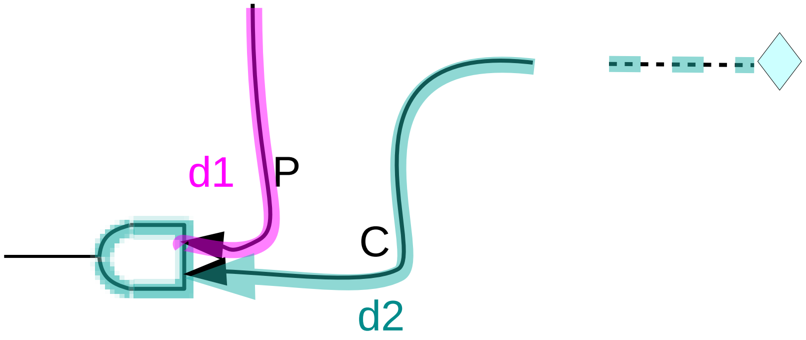
Carry Skip Adder



Delay d1 from A, B to P – parallel, the same delay in each group 🟪

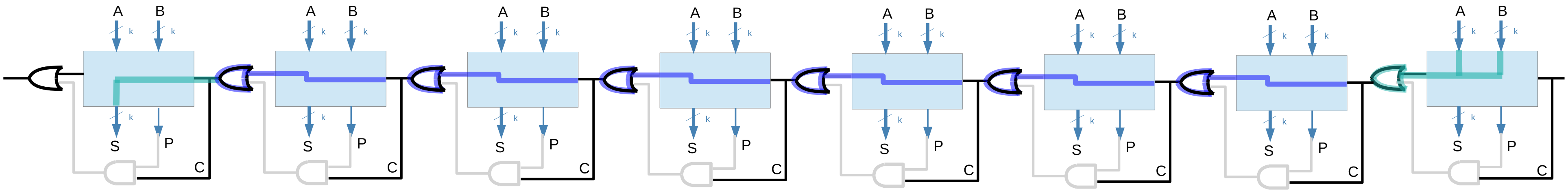
Delay d2 from A, B to C – serial, the accumulated delay from lsb 🟩

Delay d3 from A, B, Ci to Co – ripple carry delay 🟦

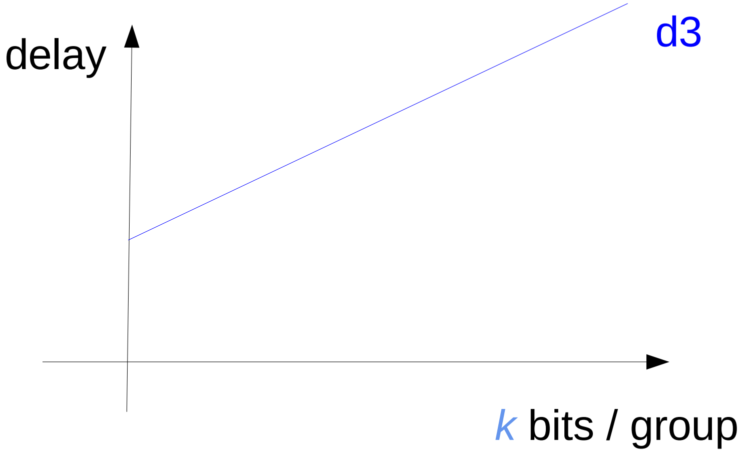


Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Carry Skip Adder

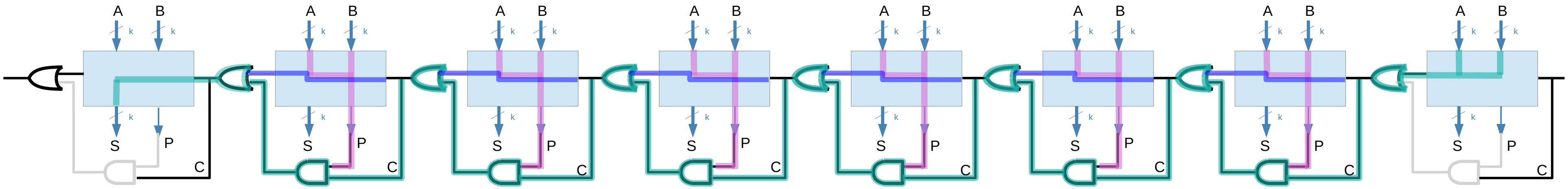


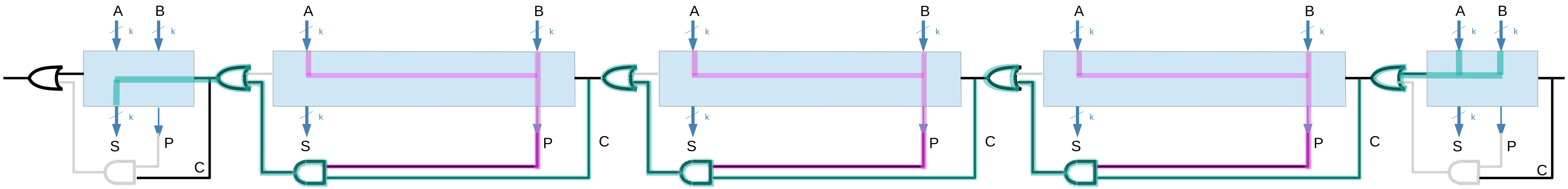
Delay d3 from A, B, Ci to Co – ripple carry delay



Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Carry Skip Adder

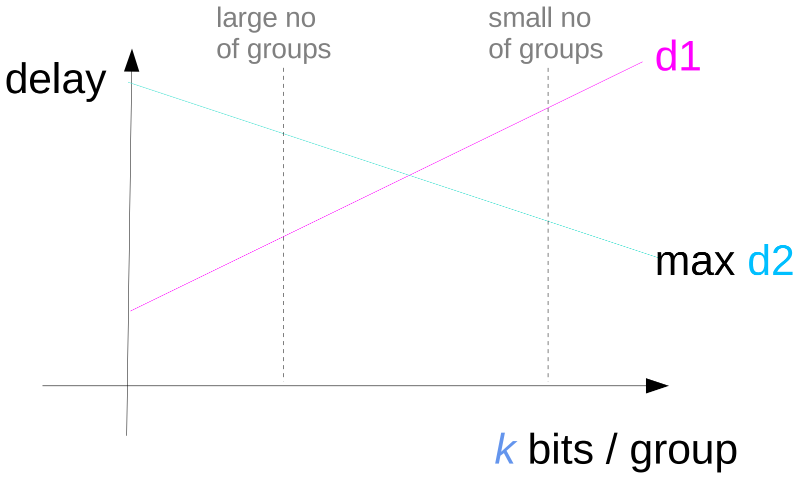




N = R·k

$$d1 \propto k$$

$$d2 \propto R \left(= \frac{N}{k} \right)$$



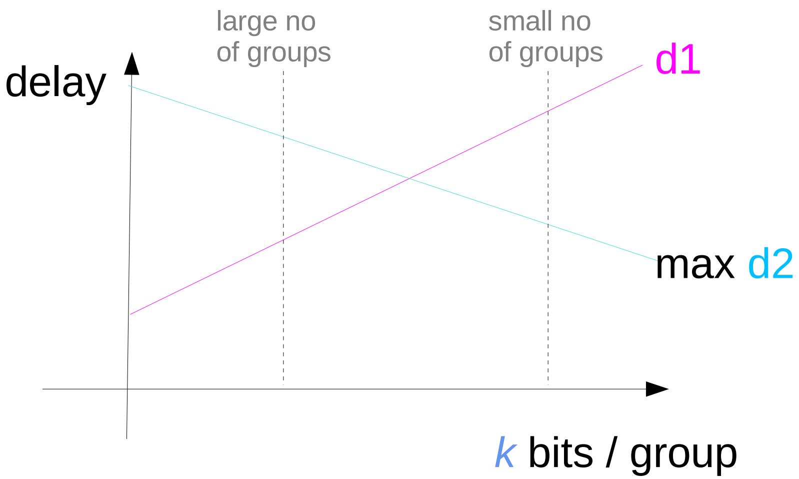
Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

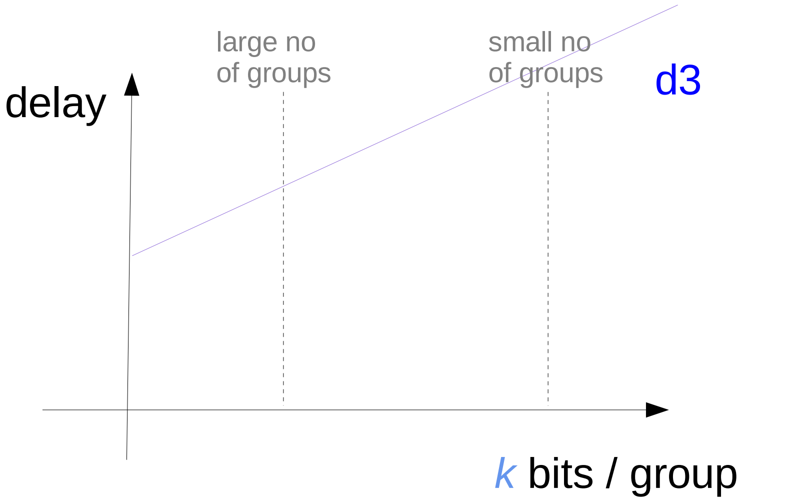
Carry Skip Adder

N = R·k

$$d1 \propto k$$

$$d2 \propto R \left(= \frac{N}{k} \right)$$





Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Variable Block

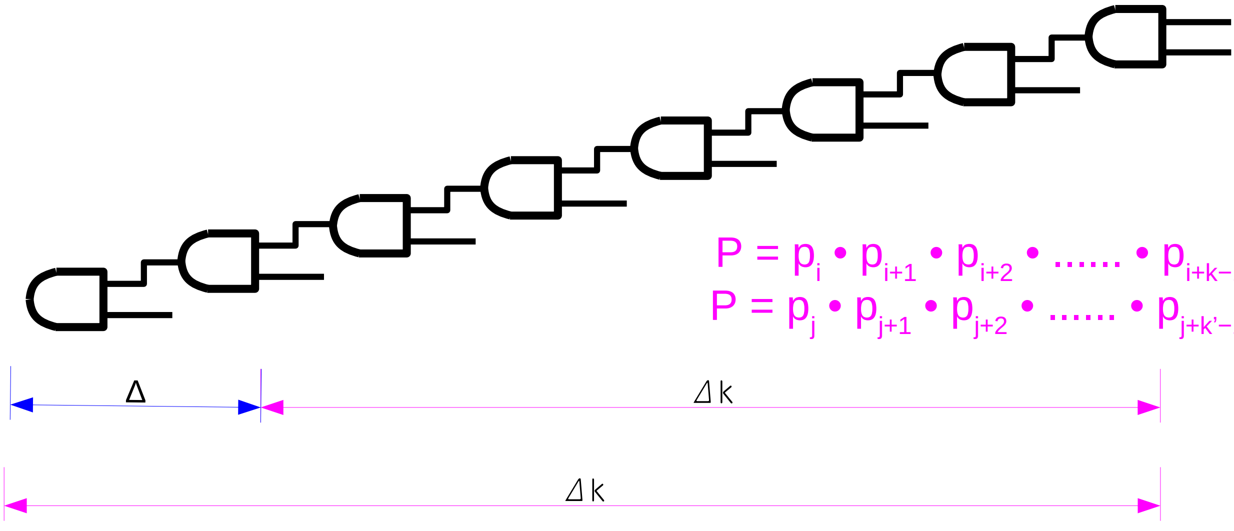
the organization of the blocks in the variable block carry structure
bears some similarity to the carry select structure

the early stages of the structure grow in length,
with short blocks for the low order bits,
building in length further in the chain
in order to equalize the arrival time
of the carry from the block
with that of the previous block

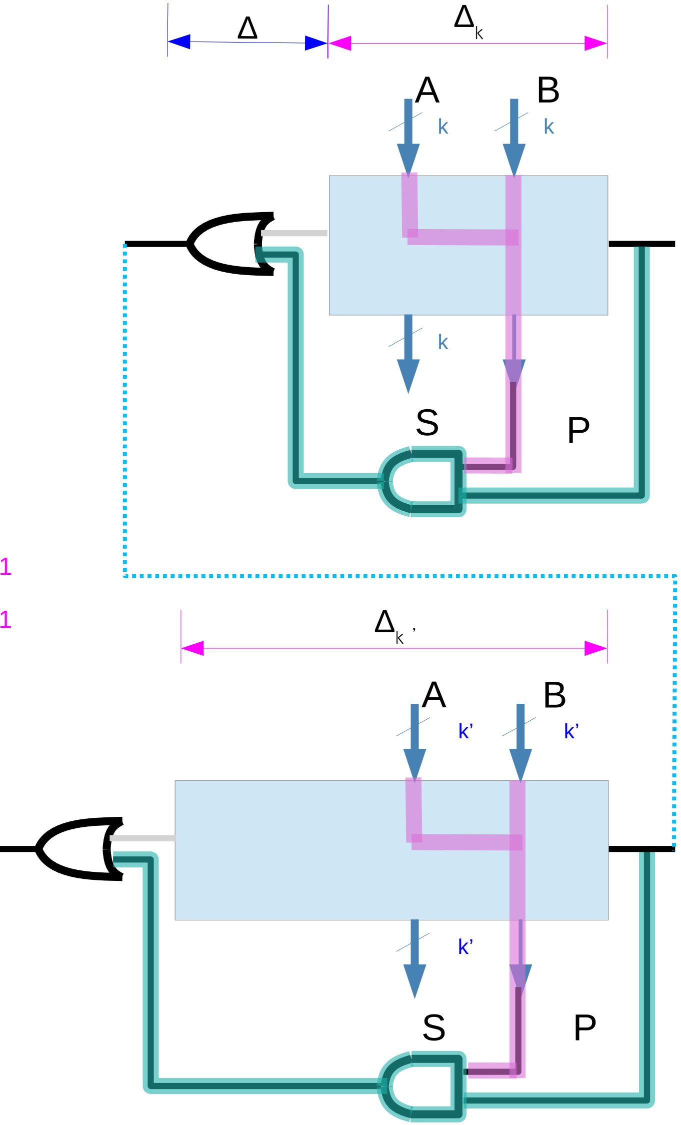
https://en.wikipedia.org/wiki/Carry-lookahead_adder

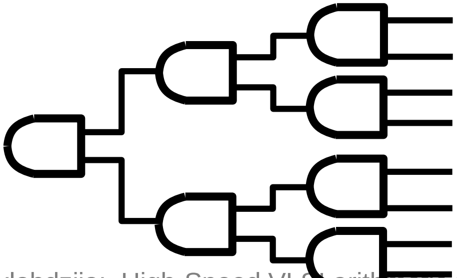
Carry Skip Adder

to equalize the arrival time
of the carry from the block
with that of the previous block



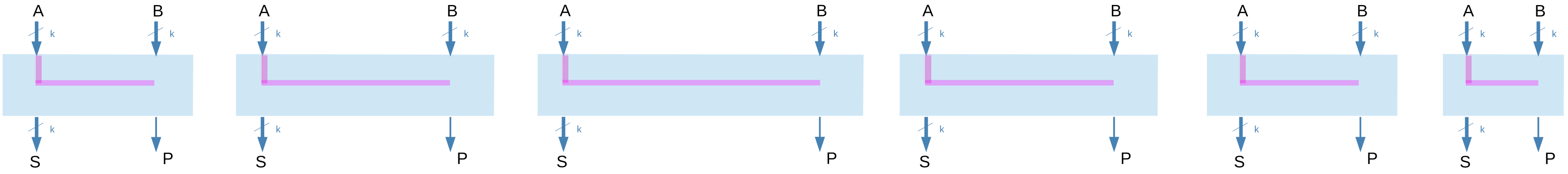
$$P = p_i \bullet p_{i+1} \bullet p_{i+2} \bullet \ldots\ldots \bullet p_{i+k-1}$$
$$P = p_j \bullet p_{j+1} \bullet p_{j+2} \bullet \ldots\ldots \bullet p_{j+k'-1}$$

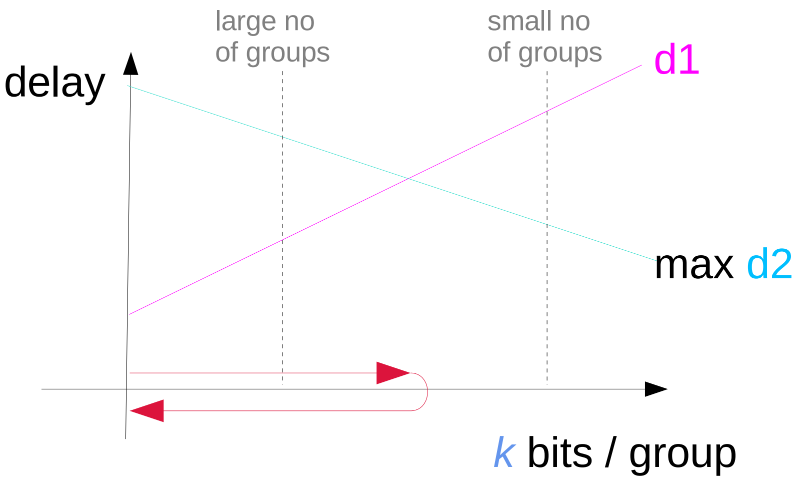


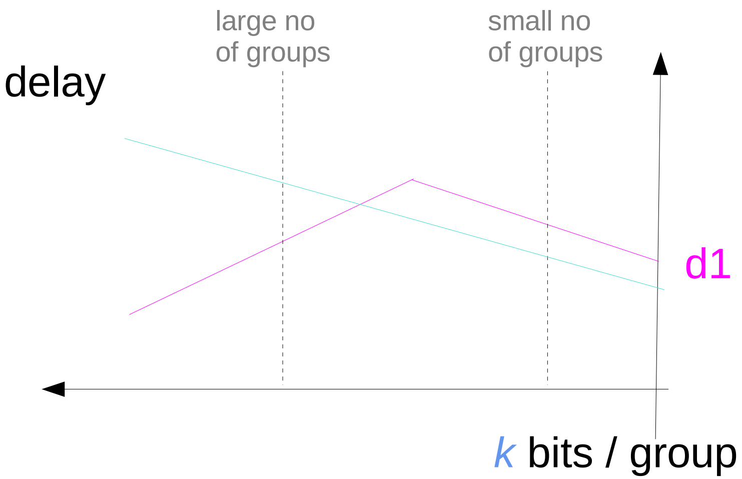


Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Carry Skip Adder

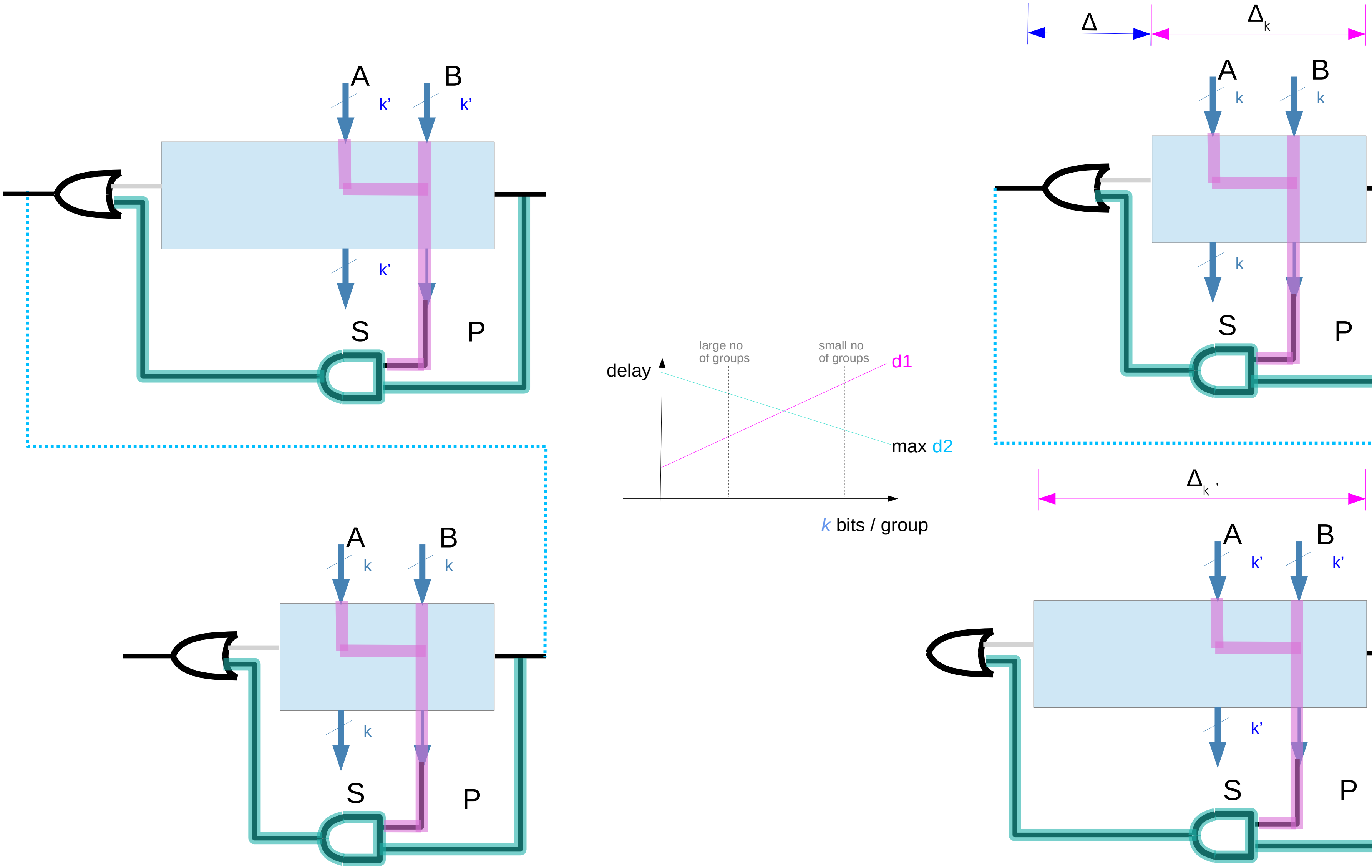






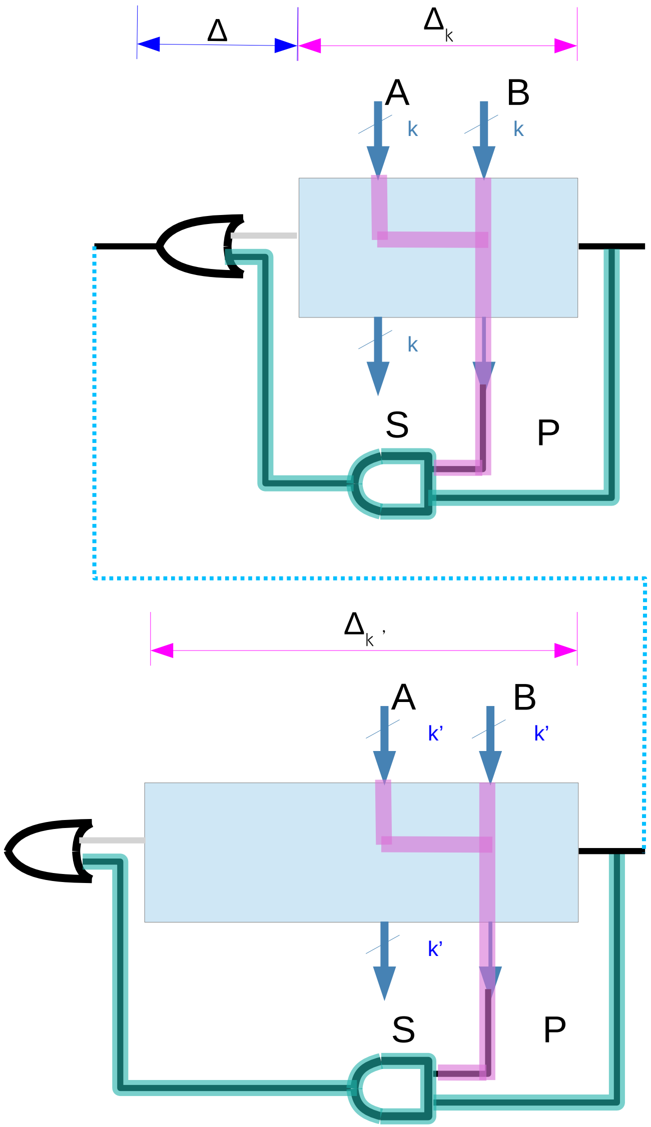
Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

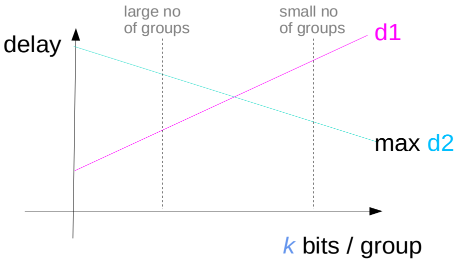
Carry Skip Adder

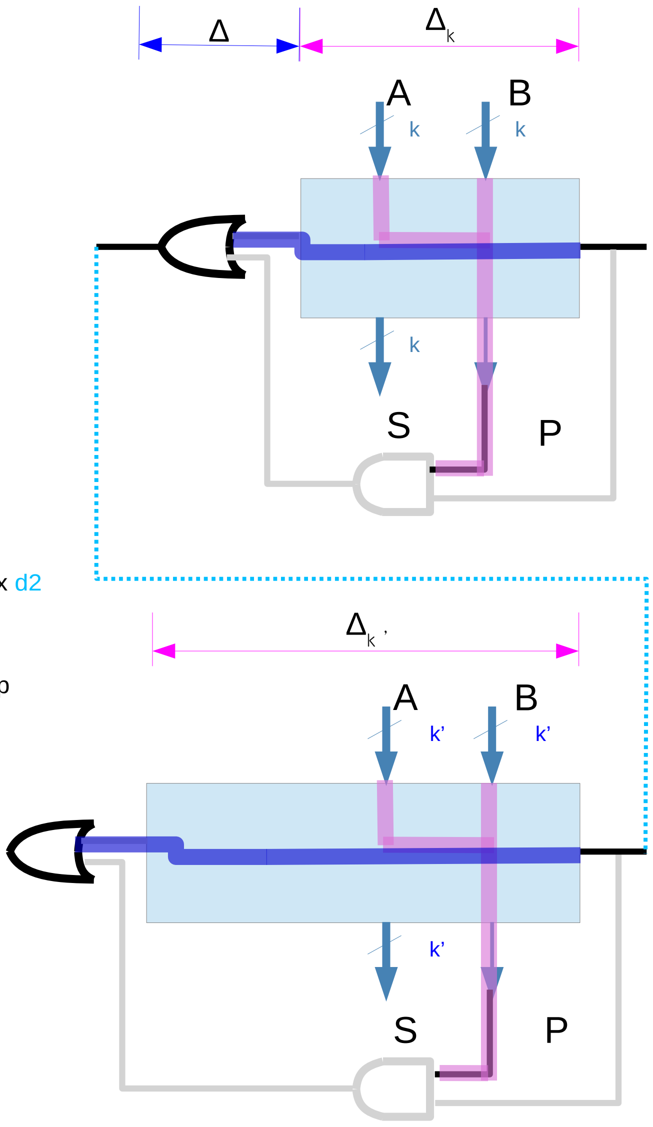


Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Carry Skip Adder

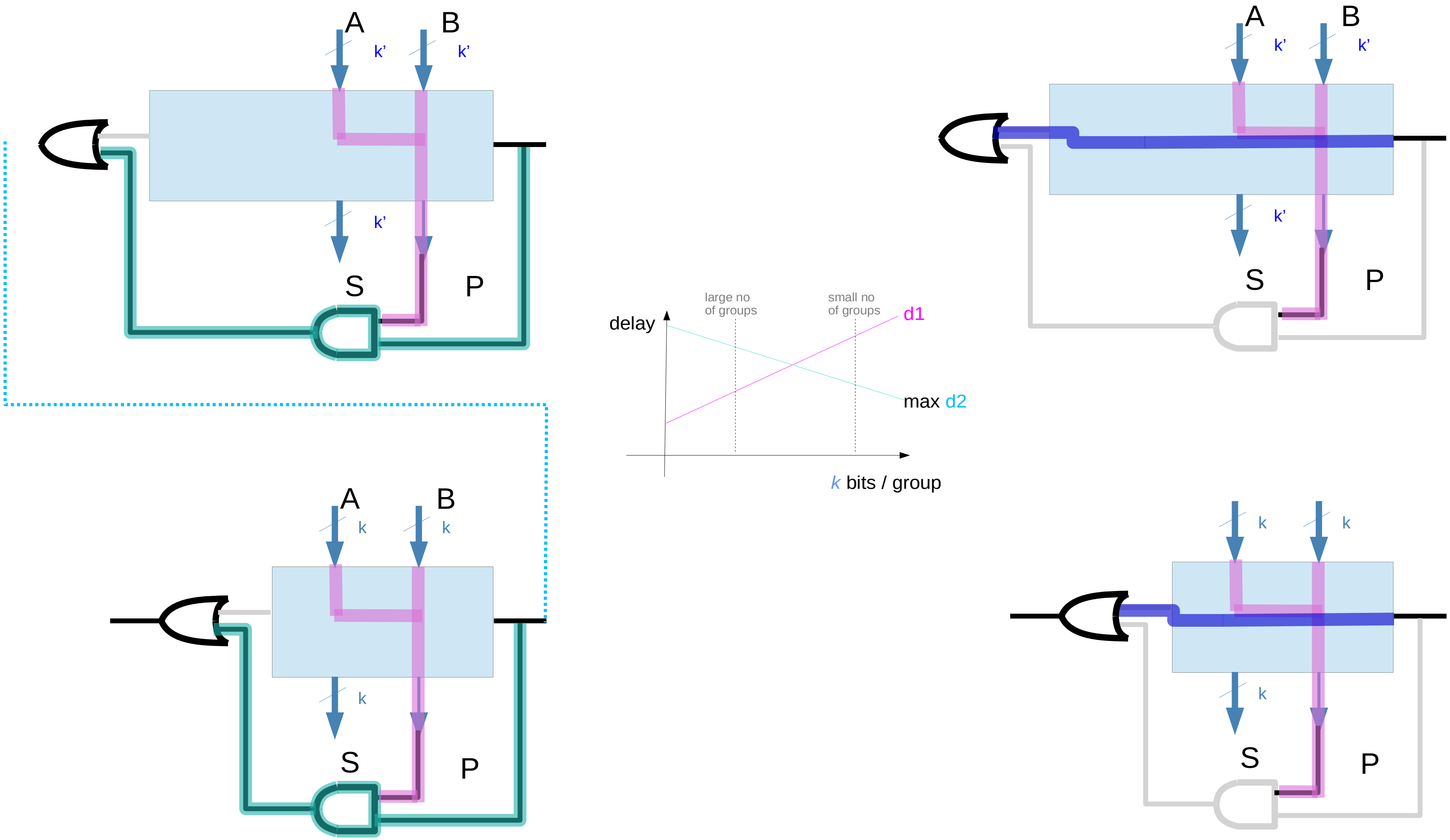






Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Carry Skip Adder



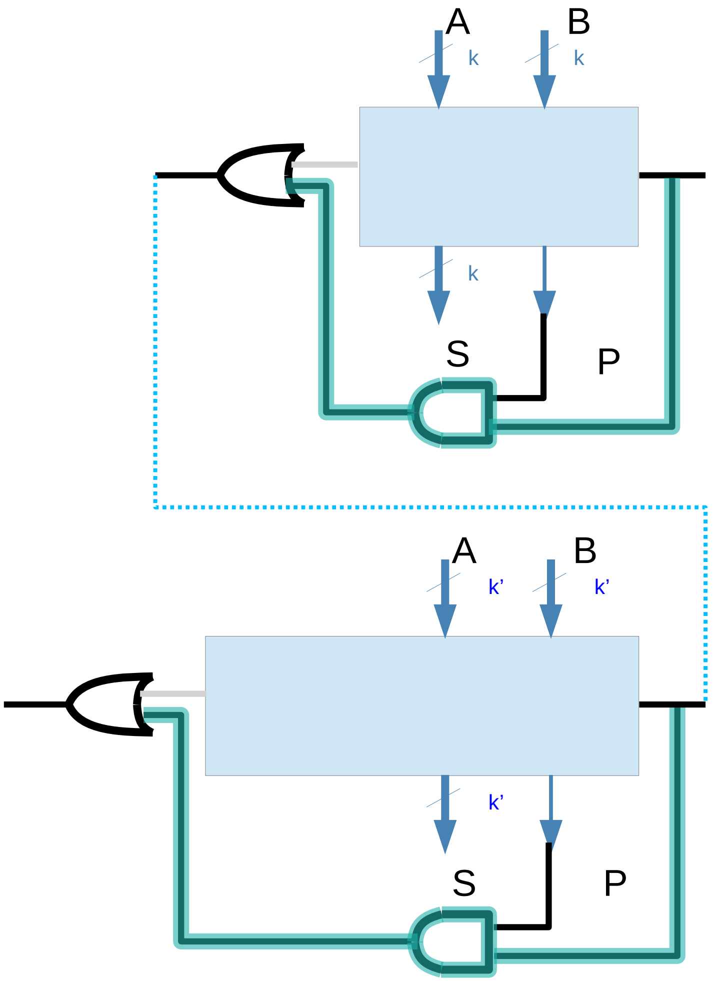
Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Carry Skip Adder

All carries propagated more quickly by varying the block sizes
Accordingly the initial blocks of the adder are made smaller, so as to quickly detect carry generates that must be propagated

The middle blocks are made larger because they are not the problem case,

And then the most significant blocks are made smaller so that the late arriving carry inputs can be processed quickly



https://en.wikipedia.org/wikik/Carry-skip_adder

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

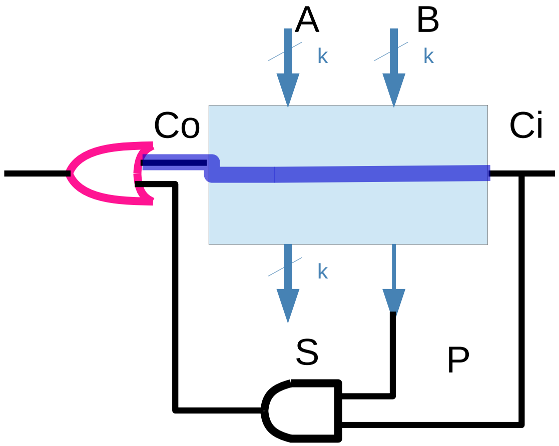
Carry Skip Adder

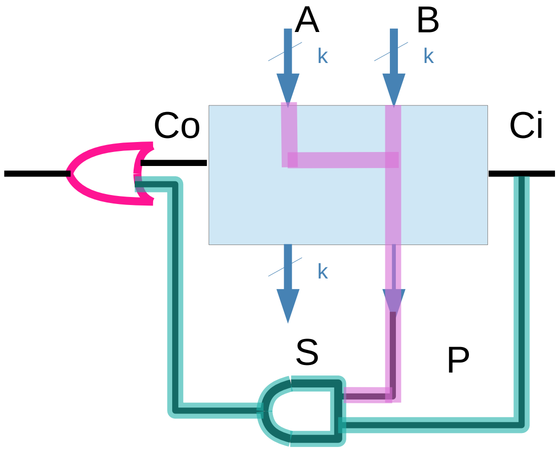
The longest path length through the carry skip block is potentially much shorter than the path from carry-in to carry-out through a ripple carry block.

However, the carry skip block has a slightly longer path from the least significant <g,t> input to carry output

Hence, this adder will only be faster when skipping groups makes up for the extra gate overhead accumulated by going from generate/transfer to carry-out

The maximum path length through a one block wide carry skip adder is the same as though a ripple carry adder, since the bottom block in a skip adder is a ripple carry





Binary Adders, T W Lynch, Master Thesis, University of Texas at Austin 1996

Two separate ripple carry adders

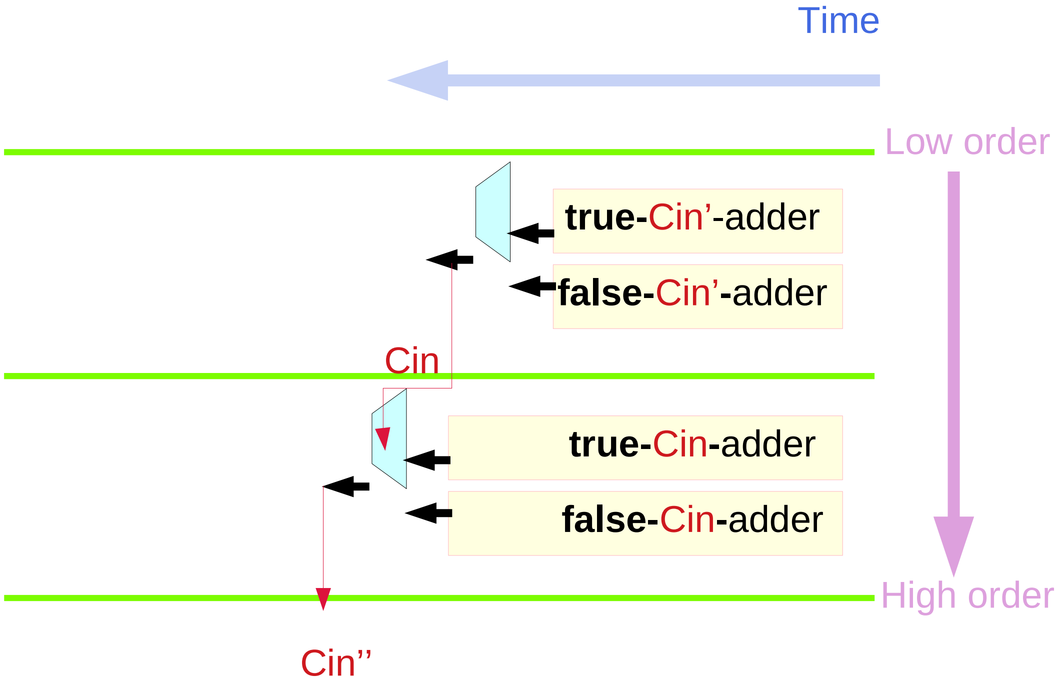
Cin signal is used to determine
which adder's outputs should be used

if the Cin signal is **true**,
the output (carries) are selected from
the **true-Cin-adder**

if the Cin signal is **false**,
the output (carries) are selected from
the **false-Cin-adder**

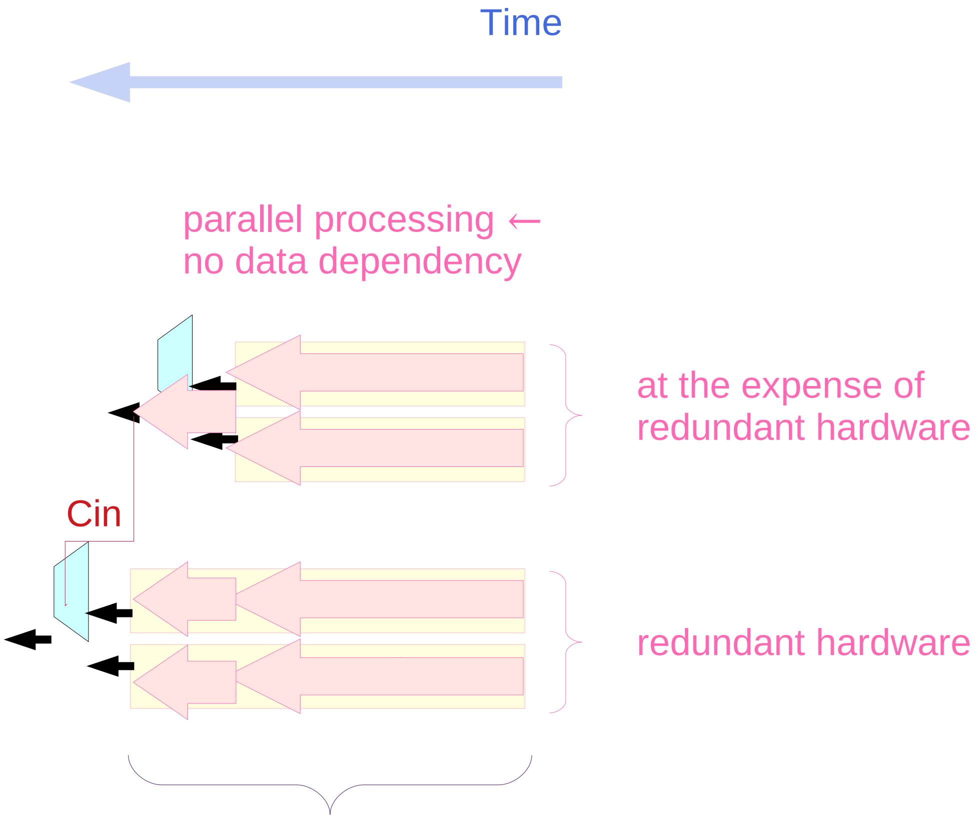
redundant hardware removes
Cin data dependency

first start redundant computation
later select the correct one



High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

Timing in broken carry chains

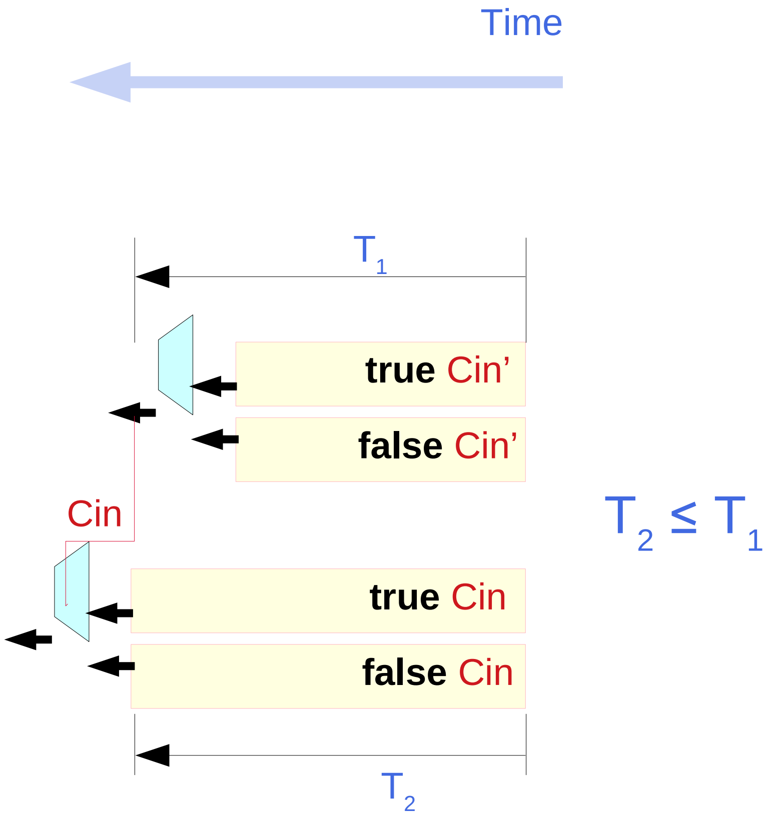


parallel processing ← no data dependency

at the expense of redundant hardware

redundant hardware

These computations can take place before the completion of the previous columns, since they do not depend on the actual value of the Cin signal



$T_2 \leq T_1$

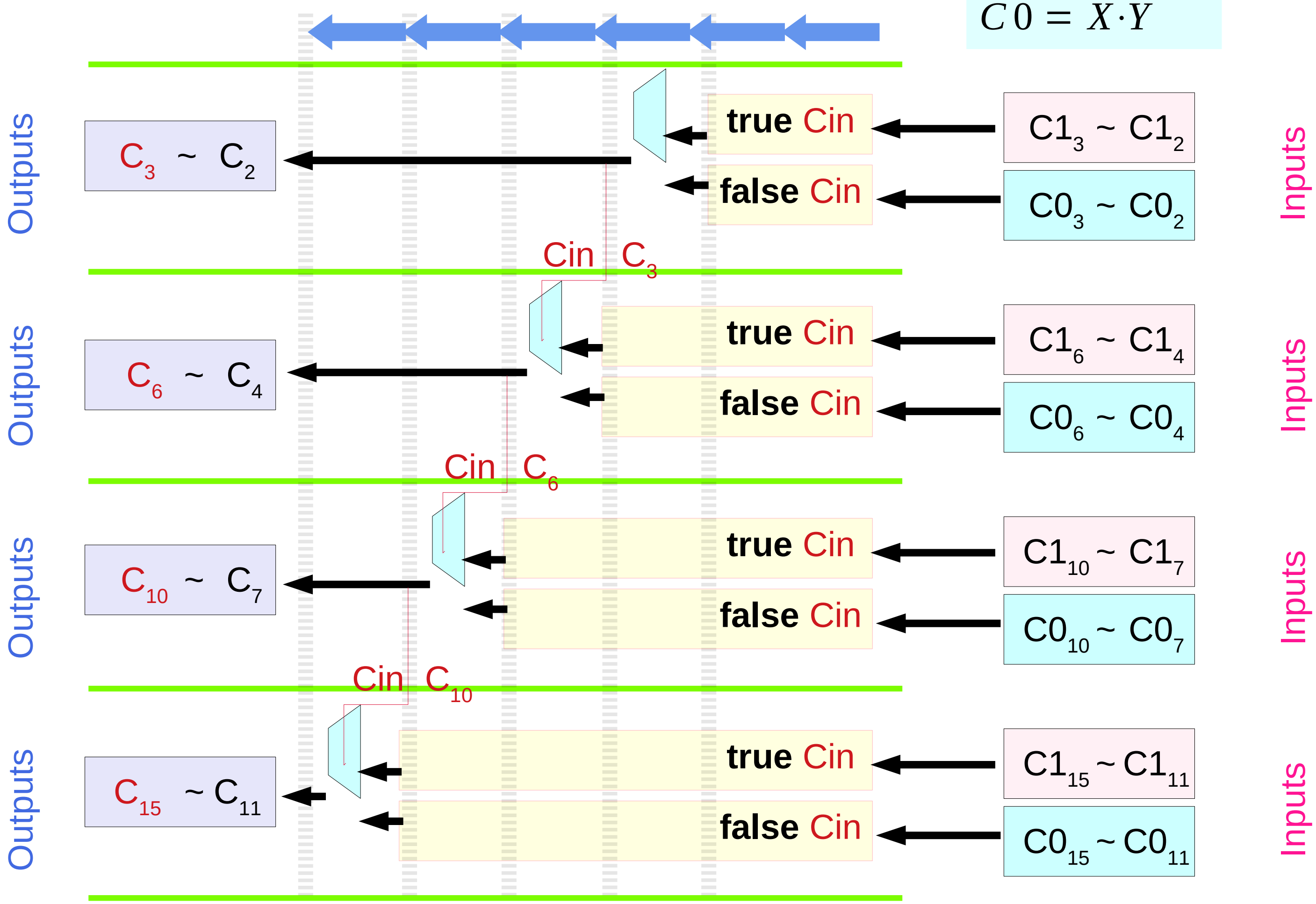
the length of the adders and the breakpoint are carefully chosen such that the **adders** finish computations just as their Cin become available

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

Carry Select Fast Carry Logic

$C1 = X + Y$

$C0 = X \cdot Y$



High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

Variable Block

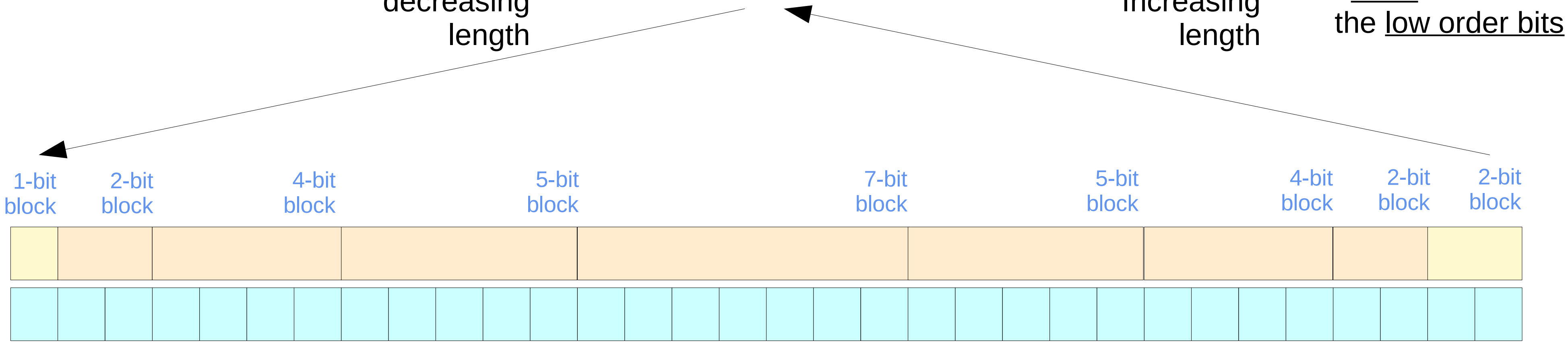
d1 > d2 → d1 < d2

d1 < d2

decreasing length

Increasing length

<u>short</u> blocks for the <u>low order bits</u>



1-bit block

2-bit block

4-bit block

5-bit block

7-bit block

5-bit block

4-bit block

2-bit block

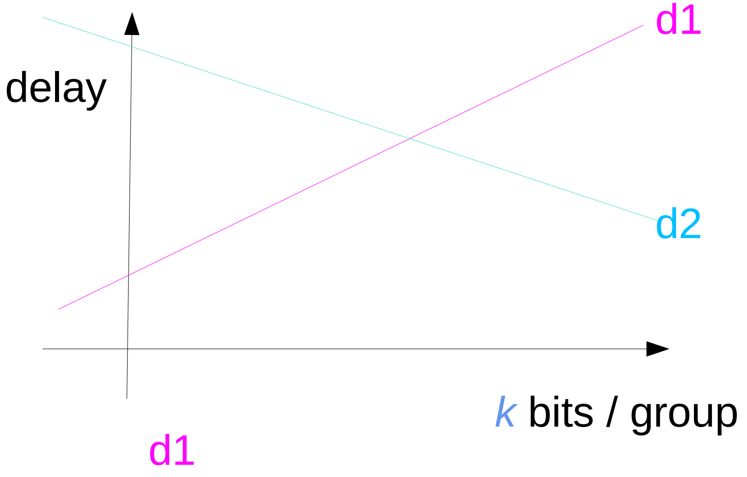
2-bit block

Simple Carry Chain

Variable Block

Simple Carry Chain

the <u>delay</u> from the Cin input through the block's ripple chain



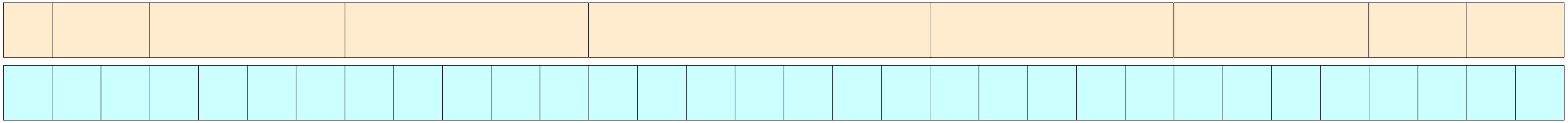
https://en.wikipedia.org/wiki/Carry-lookahead_adder

Variable Block

We use a block length from low order to high order cells
of 2, 2, 4, 5, 7, 5, 4, 2, 1 for a normal 32 bit structure

8+17+7

The first and last block in each adder is
a simple ripple carry chain,
while all other blocks use
the variable block structure.



https://en.wikipedia.org/wiki/Carry-lookahead_adder

Variable Block

Delay values of the variable block carry chain
relative to other carry chains

The idea behind Variable Block Adder (VBA) is
to minimize the longest critical path
in the carry chain of a Carry Skip Adder,
while allowing the groups to take different sizes.

Such an optimization in general does not result
in an increased complexity as compared to the
Carry Skip Adder

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Variable Block

The first and last blocks are smaller,
and the intermediate blocks are larger.

That compensates for the critical paths originating
from the ends by shortening the length of the path
used for the carry signal to ripple in the end groups,
allowing carry to skip over larger groups in the middle.

There are two important consequences of this optimization:
(a) the total delay is reduced as compared to a Carry Skip Adder
(b) the delay dependency is not a linear function
of the adder size N as in a Carry Skip Adder.
This dependency follows a square root function of N instead

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Variable Block

For an optimized VBA, it is possible to obtain
a closed form solution expressing this delay dependency

It is also possible to extend this approach
to multi-levels of carry skip as done in
a determination of the optimal sizes of the blocks
on the first and higher levels of skip blocks
is a **linear programming problem**,
which does not yield a closed form solution.

Such types of problems are solved
with the use of **dynamic programming** techniques.

The speed of such a mult-level VBA adder
surpasses single-level VBA and
that of fixed group Carry-Lookahead Adder (CLA).

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Variable Block

For an optimized VBA, it is possible to obtain a closed form solution expressing this delay dependency which is given as:
where: c1 , c2 , c3 are constants.

$$\Delta_{VBA} = c_1 + \sqrt{c_2 N + c_3}$$

It is also possible to extend this approach to multiple levels of carry skip as done.

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Variable Block

(1) the speed of the logic gates
used for CMOS implementation
depends on the output load: fan-out,
as well as the number of inputs: fan-in.

(2) CLA implementation is characterized
with a large fan-in which limits
the available size of the groups.

On the other hand VBA implementation is simple.

Thus, it seems that CLA has passed the point of diminishing returns as far as an efficient implementation is concerned.

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Variable Block

This example also points to the importance of modeling
and incorporating appropriate technology parameters into the algorithm.

Most of the computer arithmetic algorithms developed in the past
use a simple constant gate delay model.

(2.) a fixed-group CLA is not the best way to build an adder.

It is a sub-optimal structure which
after being optimized for speed,
consists of groups that are different in size
yielding a largely irregular structure

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Variable Block

There are other advantages of VBA adder that are direct result of its simplicity and efficient optimization of the critical path.

Those advantages are exhibited in the lower area and power consumption while retaining its speed.

Thus, VBA has the lowest energy-delay product as compared to the other adders in its class.

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Delay model

Oklobdzija addition VLSI

On implementing addition in VLSI technology

Delay dependency : Fan-out, Fan-in,
Delay estimates :

D_NAND = 1 +0.3Fo +0.5(Fi-2)
D_NOR = 1 +0.5Fo +0.5(Fi-2)
D_NAND = 0.7+0.3Fo

t denote the time required for a carry signal to ripple across a bit
T denote the time required for the signal to skip over a group of bits
m denotes the optimal number of groups for an n-bit carry chain
m is the smallest positive integer satisfying

$$n \le m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \left(1 - (-1)^m\right) \frac{1}{8} T$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Delay model

n : the number of bits in a carry skip adder
m : the number of groups into which the bits are divided
x_1, … , x_m : the sizes of the groups beginning with the most significant bit

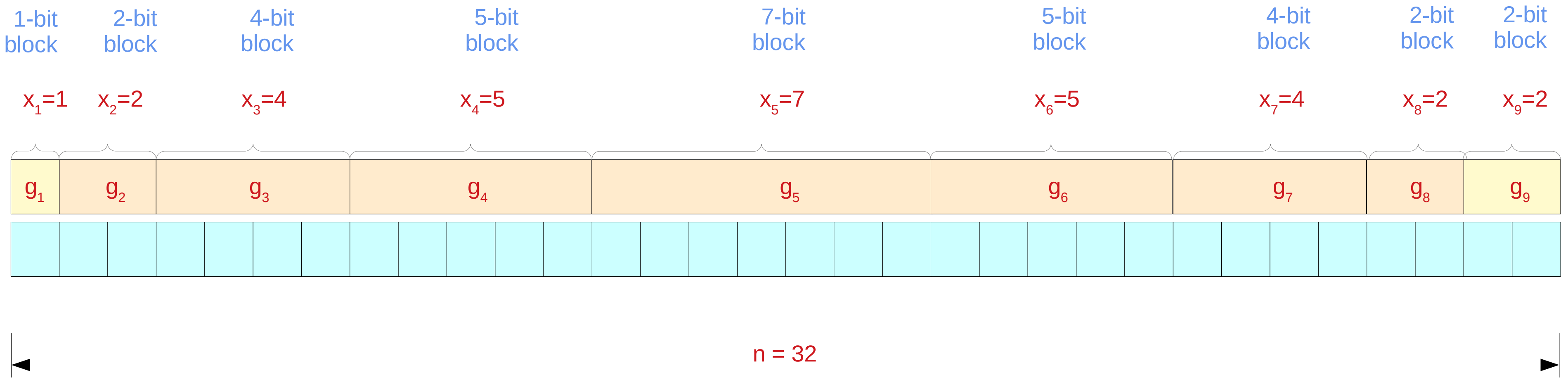
T : the time required for a carry signal to skip over a group of bits

To be precise we should write $T = T(x)$ to indicate that
T depends on the size x of the group over which the carry is skipped
However, T changes very slowly with x over the range of group sizes
So we assume that T is constant

For a given n, the following three step procedure gives
An optimal way of dividing an n bit adder into groups of bits

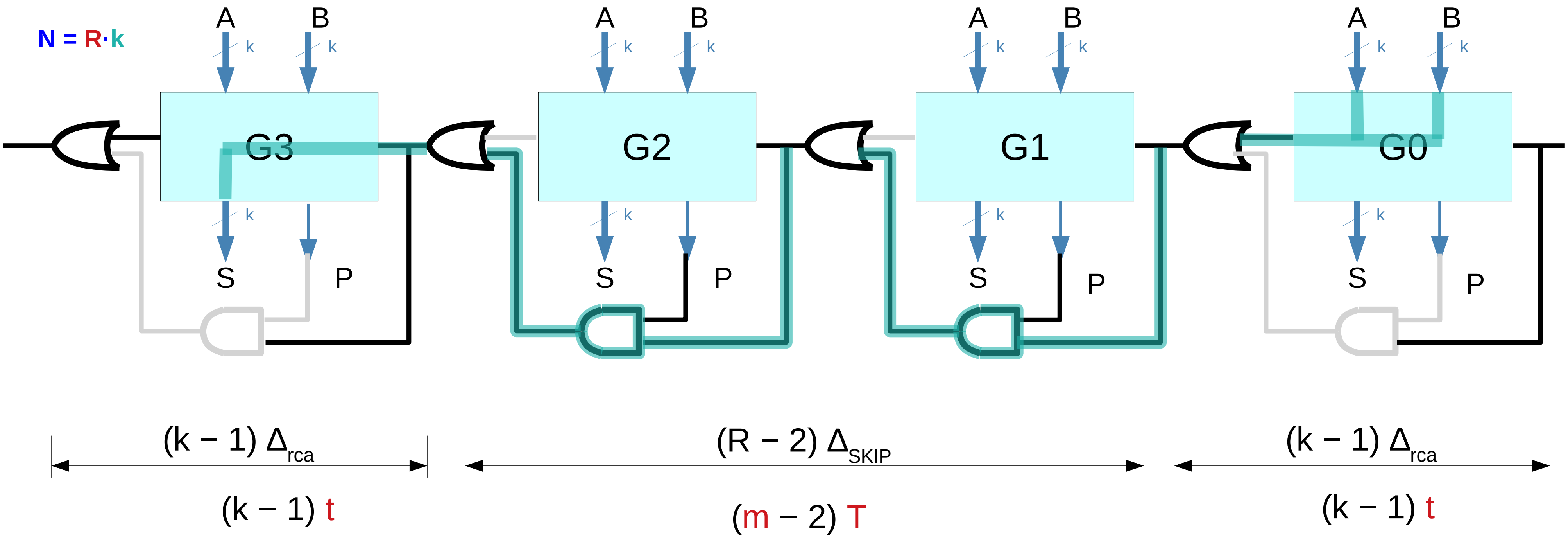
Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Variable Block



- total n = 32 bits
- m = 9 groups
- i-th group has x_i bits (size)
- constant skip delay $T = T(x_i)$

Carry Skip Adder



t denote the time required for a carry signal to ripple across a bit
T denote the time required for the signal to skip over a group of bits
m denotes the optimal number of groups for an n-bit carry chain

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

X_i and Y_i and constraints (1)

- *n* bits
- *m* groups

x_3 delay$_{\text{ripple}}$ ripple delay of a group

$$y_3 = min\{1+3T, 1+(m+1-3)T\}$$

min {delay1$_{\text{skip}}$, delay2$_{\text{skip}}$}

skip delay over a group

$$0 \le x_i \le y_i, \quad i=1,\dots,m$$

delay$_{\text{ripple}}$ ≤ delay1$_{\text{skip}}$
delay$_{\text{ripple}}$ ≤ delay2$_{\text{skip}}$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

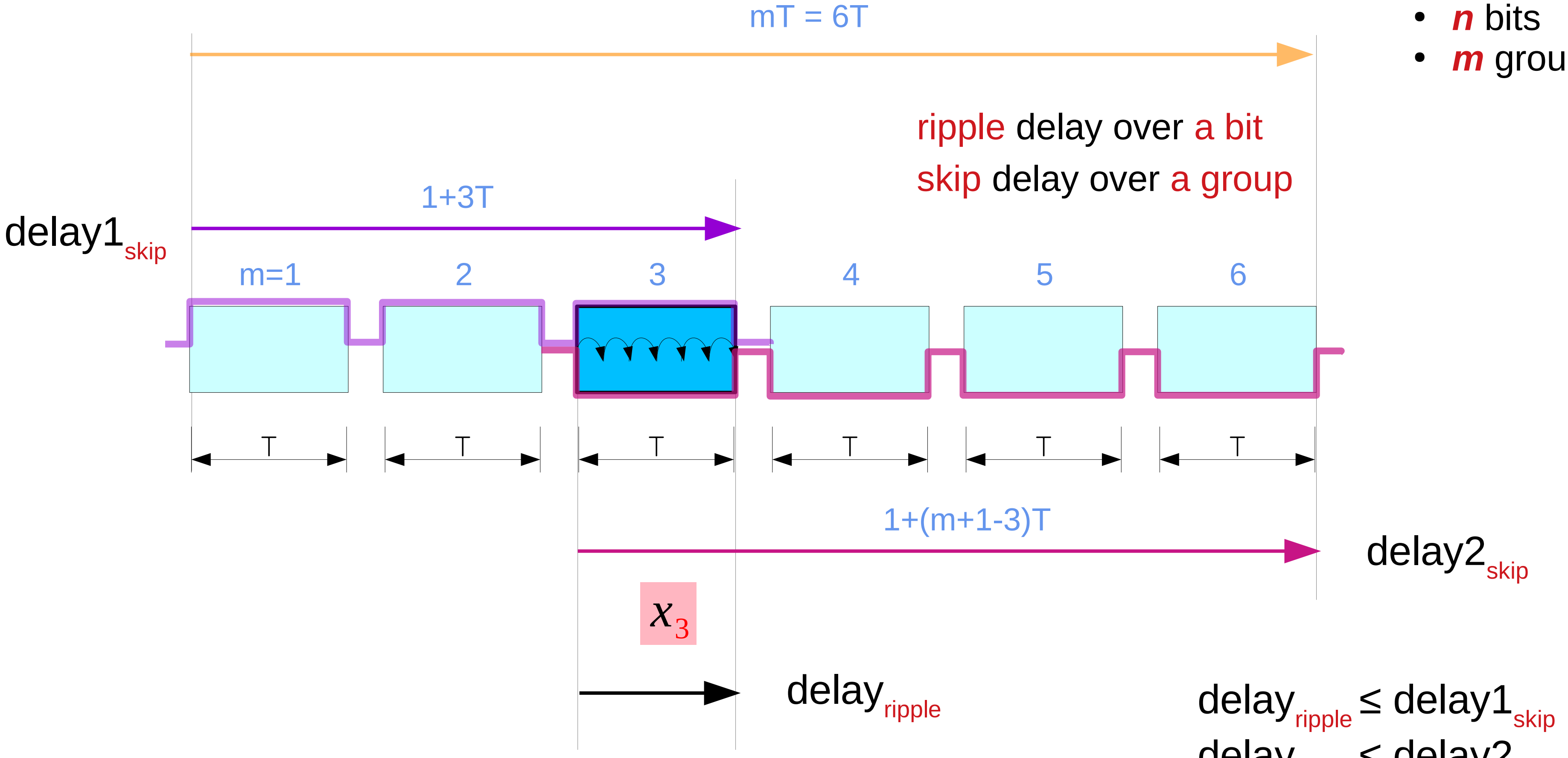
X_i and Y_i and constraints (2)

$$y_3 = min\{1+3T, 1+(m+1-3)T\}$$

- ***n*** bits
- ***m*** groups

ripple delay over a bit

skip delay over a group



$delay_{ripple} \le delay1_{skip}$

$delay_{ripple} \le delay2_{skip}$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

$$0 \le x_i \le y_i, \quad i=1,\ldots,m$$

Determining *m*

Method 1 – using a histogram

Let *m* be the smallest positive integer such that

$$n \le \boxed{\sum_{i=1}^{m} y_i}$$

$$\boxed{y_i = min\{1+iT, 1+(m+1-i)T\}, \quad i = 1, \ldots, m}$$

Method 2 – using a closed formula

Let *m* be the smallest positive integer such that

$$n \le \boxed{m + \frac{1}{2} m T + \frac{1}{4} m^2 T + (1-(-1)^m)\frac{1}{8} T}$$

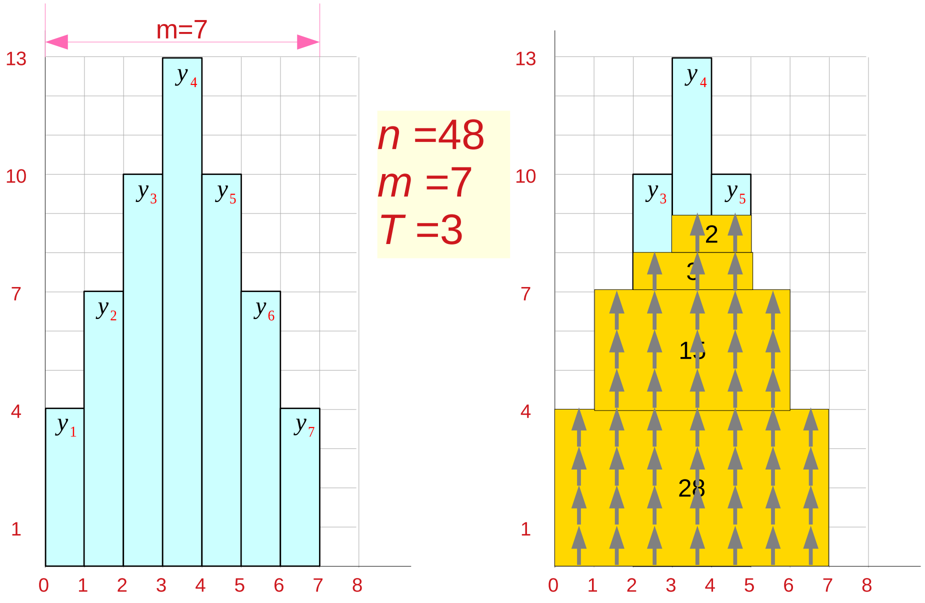
Determining x_i (1)

$$y_i = \min\{1+iT,\, 1+(m+1-i)T\}, \quad i = 1, \ldots, m$$

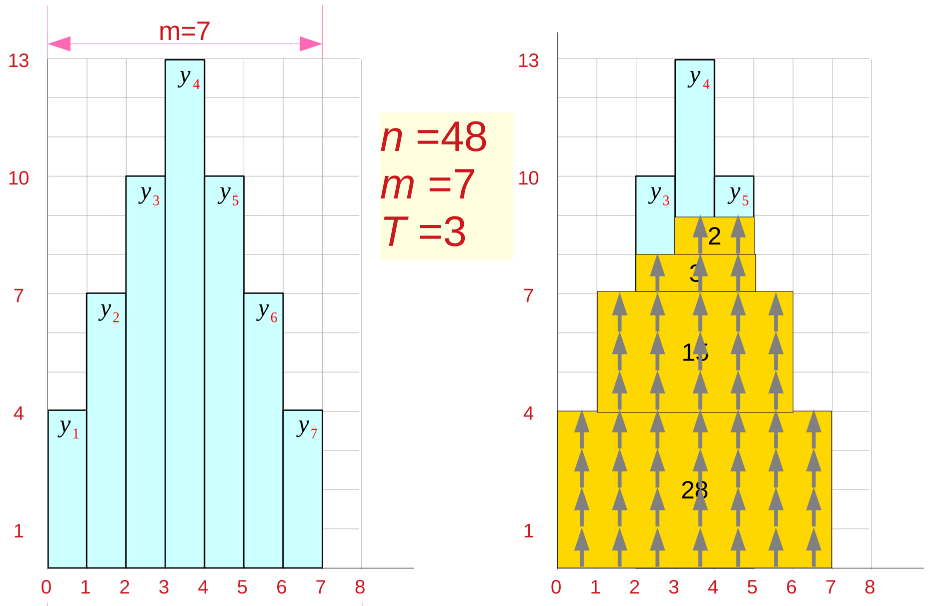
construct a histogram
whose *i-th* column has height y_i

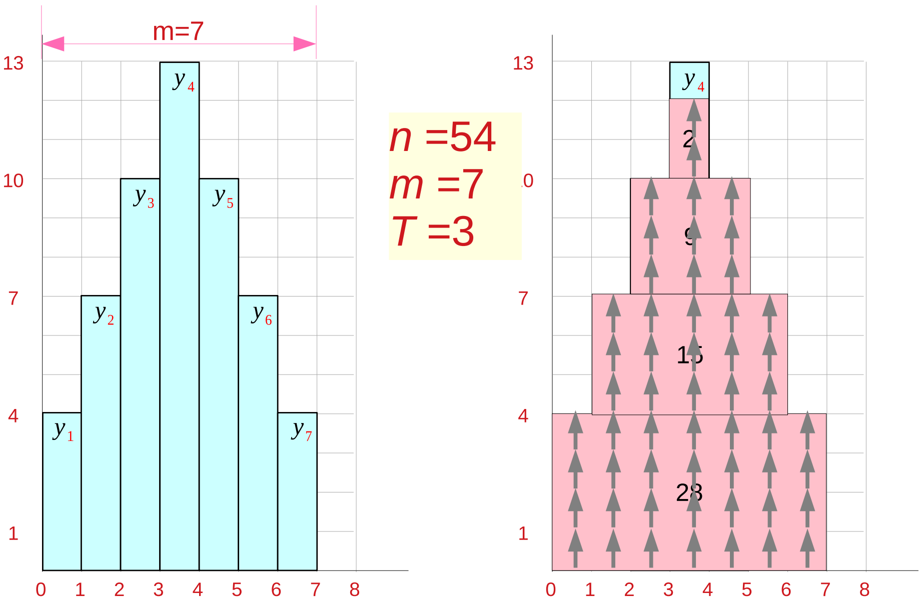
so these y_i's are at least n unit squares
in the histogram, starting with the first row,
shade in n of the squares, row by row

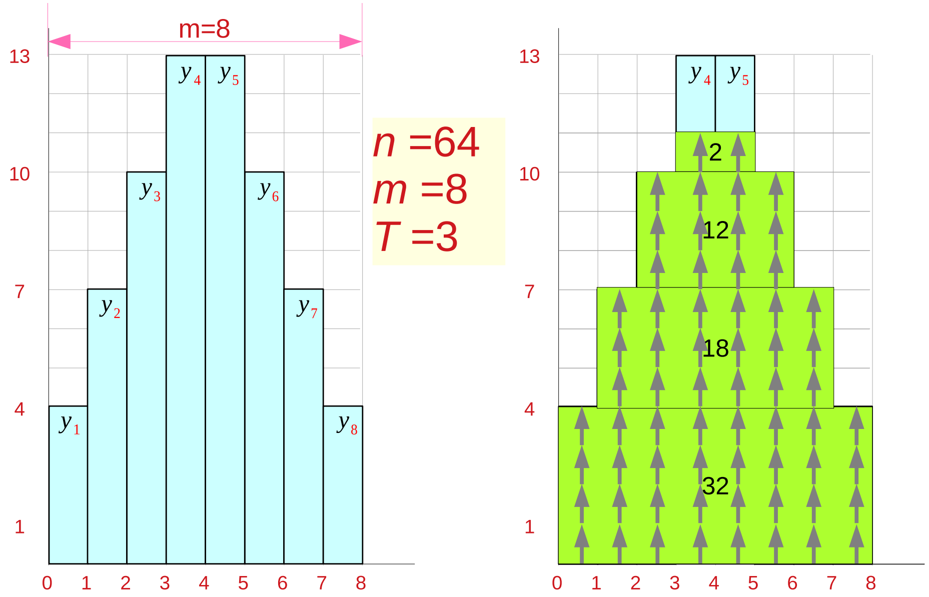
let x_i denote the number of shaded squares
in column i of the histogram,
$i = 1, \ldots, m$



Determining x_i (2)







The x_i's can be computed iteratively as follows:

Initially take x_1 = x_m = 0

At each iteration,
increase as many of the x_i's as possible by one unit,
without violating the constraints

$$0 \le x_i \le y_i, \quad i = 1, \dots, m$$

Thus, at some iteration, we have $\sum_{i=1}^{m} x_i = n$ and
the algorithm terminates

Determining ***m*** (method 1 : using a histogram)

Let m be the <u>smallest</u> positive integer such that

$$n \le \boxed{\sum_{i=1}^{m} y_i}$$

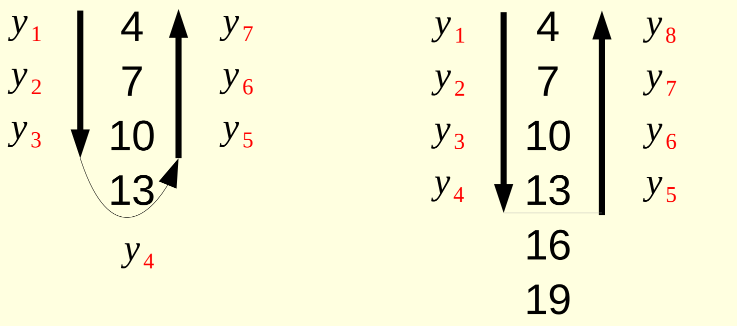
$$\boxed{y_i = min\{1+iT, 1+(m+1-i)T\}, \quad i = 1, \ldots, m}$$

$$y_1 = 1 + 1 \cdot T \qquad y_m = 1 + 1 \cdot T$$
$$y_2 = 1 + 2 \cdot T \qquad y_{m-1} = 1 + 2 \cdot T$$
$$y_3 = 1 + 3 \cdot T \qquad y_{m-2} = 1 + 3 \cdot T$$
$$\vdots \qquad\qquad \vdots$$

$n = 48$ $m = 7$ $T = 3$	$n = 54$ $m = 7$ $T = 3$	$n = 64$ $m = 8$ $T = 3$

if $T = 3$

$m = 7 \ (odd)$ $\qquad$ $m = 8 \ (even)$



$$\sum_{i=1}^{7} y_i = 55 \qquad \sum_{i=1}^{8} y_i = 68$$

y_i's for a given m

Let m be the smallest positive integer such that

$$n \le \boxed{\sum_{i=1}^{m} y_i}$$

$$\boxed{y_i = \min\{1+iT, 1+(m+1-i)T\}, \quad i = 1, \dots, m}$$

$$y_i = \min\{1+iT, 1+(3-i)T\}, \quad i = 1, \dots, 2, \quad \leftarrow \quad m = 2$$

$$y_i = \min\{1+iT, 1+(4-i)T\}, \quad i = 1, \dots, 3, \quad \leftarrow \quad m = 3$$

$$y_i = \min\{1+iT, 1+(5-i)T\}, \quad i = 1, \dots, 4, \quad \leftarrow \quad m = 4$$

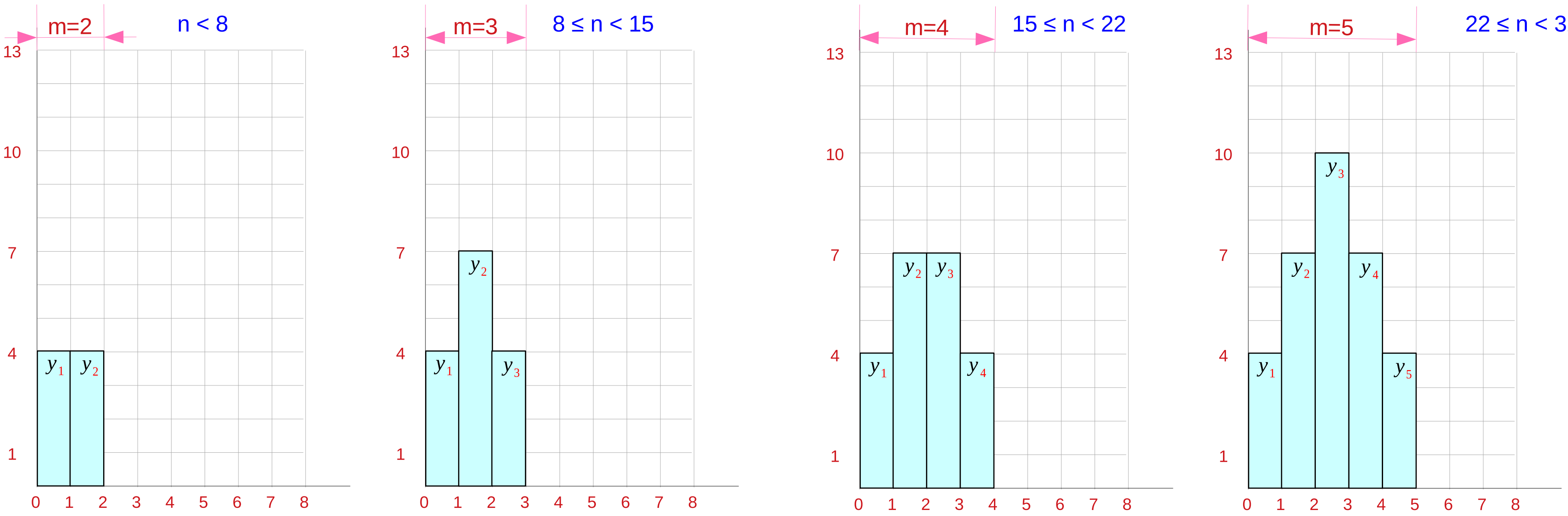
$$y_i = \min\{1+iT, 1+(6-i)T\}, \quad i = 1, \dots, 5, \quad \leftarrow \quad m = 5$$

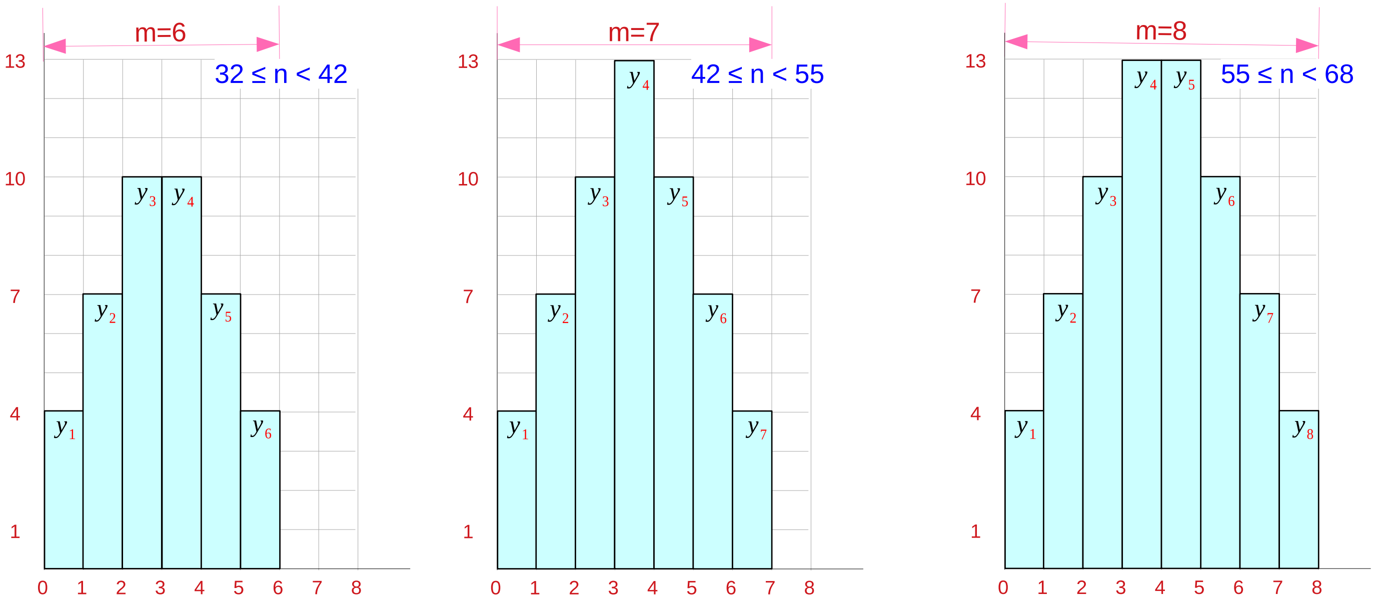
$$y_i = \min\{1+iT, 1+(7-i)T\}, \quad i = 1, \dots, 6, \quad \leftarrow \quad m = 6$$

$$y_i = \min\{1+iT, 1+(8-i)T\}, \quad i = 1, \dots, 7, \quad \leftarrow \quad m = 7$$

$$y_i = \min\{1+iT, 1+(9-i)T\}, \quad i = 1, \dots, 8, \quad \leftarrow \quad m = 8$$

Histogram – y_i's for a given m





Assume $T = 3$

$n = 48$ $T = 3$ $m = 7$	$n = 54$ $T = 3$ $m = 7$
$n = 64$ $T = 3$ $m = 8$	

Example 1 - (1)

For a 48 bit adder we have, from Figure

$$x_1 = x_7 = 4, x_2 = x_6 = 7, x_3 = 8, x_4 = x_5 = 9$$

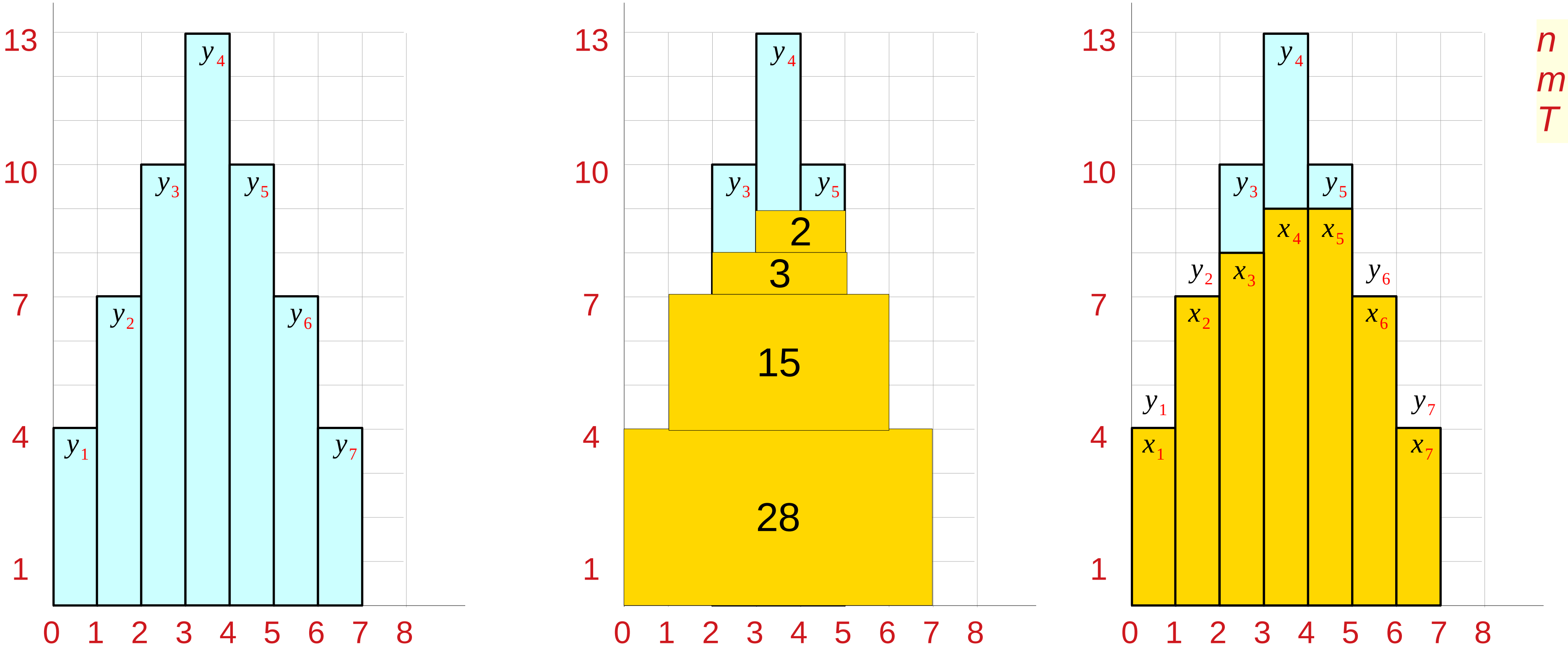
The maximum delay is experienced
by a signal generated in the second bit position
and terminating in the 47th bit position

the delay is mT = 21

- total n =48 bits
- m =7 groups
- i-th group has x_i bits (size)
- constant skip delay T = $T(x_i)$ =3

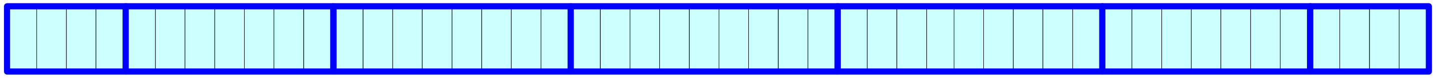
Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Example 1 - (2)


13
10
7
4
1
0 1 2 3 4 5 6 7 8
y_1 y_2 y_3 y_4 y_5 y_6 y_7
2
3
15
28
x_1 x_2 x_3 x_4 x_5 x_6 x_7

n =48
m =7
T =3

$n = 48$ $28+15+5=48 < 55 = \sum_{i=1}^{7} y_i$ → $m = 7$



$x_1 = 4$ $x_2 = 7$ $x_3 = 8$ $x_4 = 9$ $x_5 = 9$ $x_6 = 7$ $x_7 = 4$

Example 2 - (1)

consider a 54 bit adder

From 2(i), we see that again m=7.

If we shade 54 squares in Figure, we see that

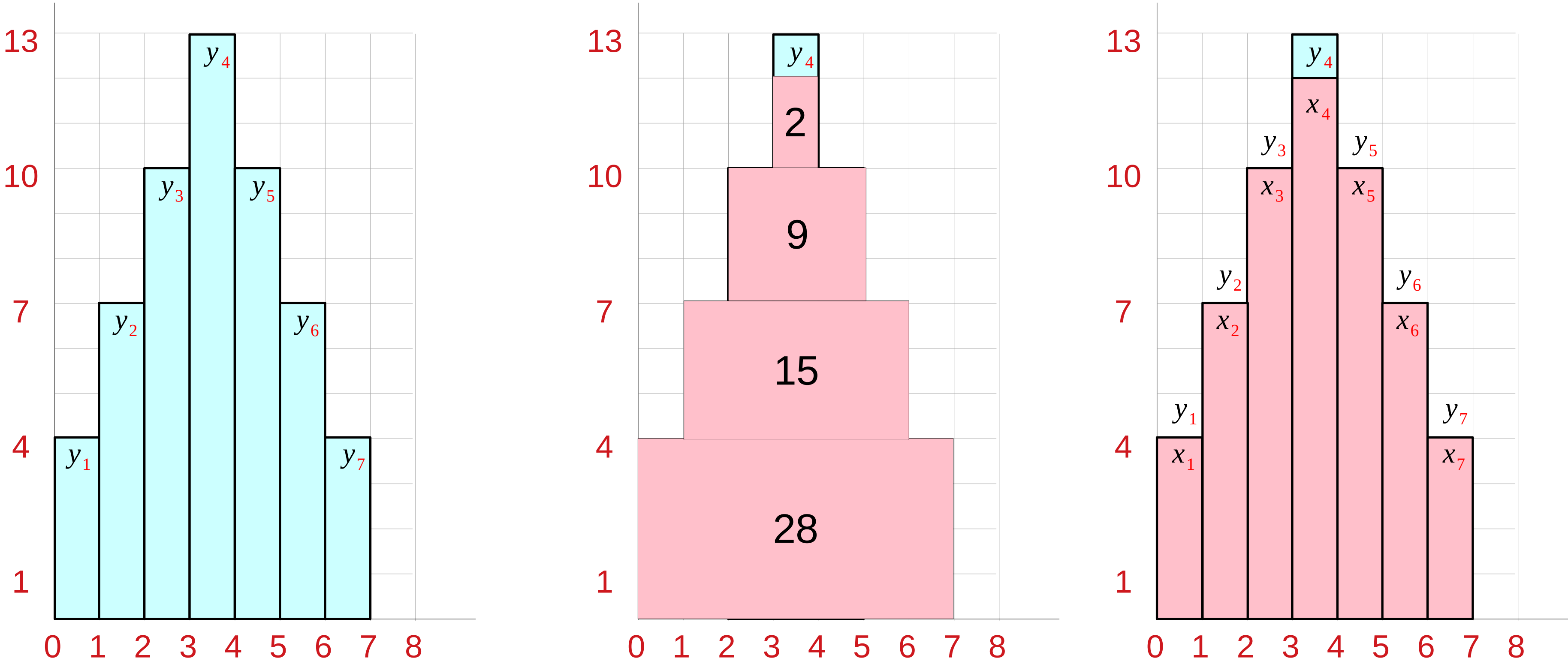
$$x_1 = x_7 = 4, x_2 = x_6 = 7, x_3 = x_5 = 10, x_4 = 12$$

Yields an optimal division of the adder.
Again the maximum delay is mT = 21

- total $n = 54$ bits
- $m = 7$ groups
- i-th group has x_i bits (size)
- constant skip delay $T = T(x_i) = 3$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

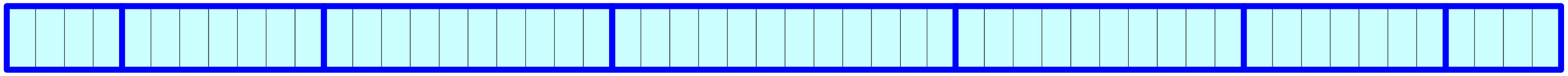
Example 2 - (2)



n =54
m =7
T =3

$n = 54$

$$28+15+11=54 \; < \; 55 = \sum_{i=1}^{7} y_i \Rightarrow m = 7$$



$x_1 = 4$ $x_2 = 7$ $x_3 = 10$ $x_4 = 11$ $x_5 = 10$ $x_6 = 7$ $x_7 = 4$

4 7 10 12 10 7 4

Example 3 - (1)

Consider a 64 bit adder
From 2(i) we compute m=8.

the corresponding histogram is shown in Figure

The optimal gorup sizes are:

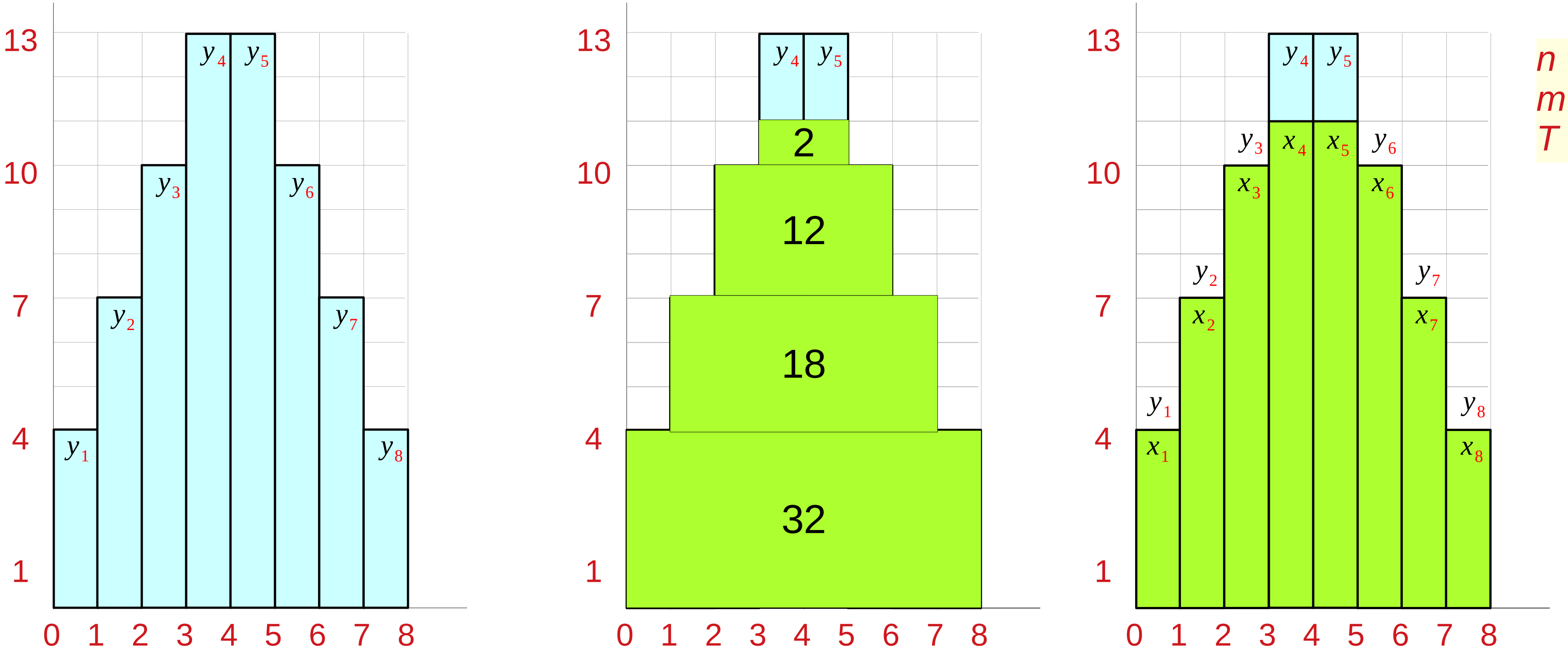
$$x_1 = x_8 = 4, x_2 = x_7 = 7, x_3 = x_6 = 10, x_4 = x_5 = 11$$

The delay of the longest signal is mT = 24

- total n =64 bits
- m =8 groups
- i-th group has x_i bits (size)
- constant skip delay $T = T(x_i) = 3$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

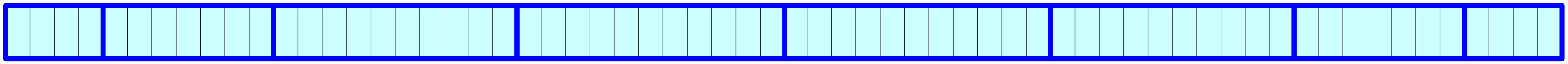
Example 3 - (2)



n =64
m =8
T =3

$n = 64$

$50+12+2=64 < 68 = \sum_{i=1}^{8} y_i$ → $m = 8$



$x_1 = 4$ $x_2 = 7$ $x_3 = 10$ $x_4 = 11$ $x_5 = 11$ $x_6 = 10$ $x_7 = 7$ $x_8 = 4$

Determining ***m*** (method 2 – using a closed formula)

Let *m* be the smallest positive integer such that

$$n \le \sum_{i=1}^{m} y_i \quad \Longrightarrow \quad n \le \boxed{m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \left(1 - (-1)^m\right) \frac{1}{8} T}$$

$$\boxed{y_i = min\{1 + iT, 1 + (m + 1 - i)T\}, \quad i = 1, \ldots, m}$$

$$\begin{aligned} y_1 &= 1 + 1 \cdot T & \qquad y_m &= 1 + 1 \cdot T \\ y_2 &= 1 + 2 \cdot T & y_{m-1} &= 1 + 2 \cdot T \\ y_3 &= 1 + 3 \cdot T & y_{m-2} &= 1 + 3 \cdot T \\ &\vdots & &\vdots \end{aligned}$$

$n = 48$
$m = 7$
$T = 3$

$n = 54$
$m = 7$
$T = 3$

$n = 64$
$m = 8$
$T = 3$

Odd *m* and even *m* cases (1)

(I) Let *m* be the smallest positive integer such that

$$n \leq m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \left(1 - (-1)^m\right) \frac{1}{8} T$$

- total $n = 48$ bits
- $m = 7$ groups
- i-th group has x_i bits (size)
- constant skip delay $T = T(x_i) = 3$

$$n \leq m + \frac{1}{2} m T + \frac{1}{4} m^2 T \qquad (even \quad m)$$

$$n \leq m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \frac{1}{4} T \qquad (odd \quad m)$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Odd *m* and even *m* cases (2)

$$\text{even } m \;:\; n \le m + \frac{1}{2} m T + \frac{1}{4} m^2 T$$

$$\text{odd } m \;:\; n \le m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \frac{1}{4} T = m + \frac{1}{4} (m+1)^2 T$$

$$m = 5 \;:\; 5 + \frac{5}{2} T + \frac{25}{4} T + \frac{1}{4} T = 5 + \frac{35}{4} T + \frac{1}{4} T = 5 + 9 \cdot 3 = \mathbf{32}$$

$$m = 6 \;:\; 6 + \frac{6}{2} T + \frac{36}{4} T = 6 + \frac{48}{4} T = 6 + 12 \cdot 3 = \mathbf{41}$$

$$m = 7 \;:\; 7 + \frac{7}{2} T + \frac{49}{4} T + \frac{1}{4} T = 7 + \frac{63}{4} T + \frac{1}{4} T = 7 + 16 \cdot 3 = \mathbf{55}$$

$$m = 8 \;:\; 8 + \frac{8}{2} T + \frac{64}{4} T = 8 + \frac{80}{4} T = 8 + 20 \cdot 3 = \mathbf{68}$$

$32 < n \le 41$ → $m = 6$

$41 < n \le 55$ → $m = 7$

$55 < n \le 68$ → $m = 8$

$n = 48$
$m = 7$
$T = 3$

$n = 54$
$m = 7$
$T = 3$

$n = 64$
$m = 8$
$T = 3$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Example 1 - (3)

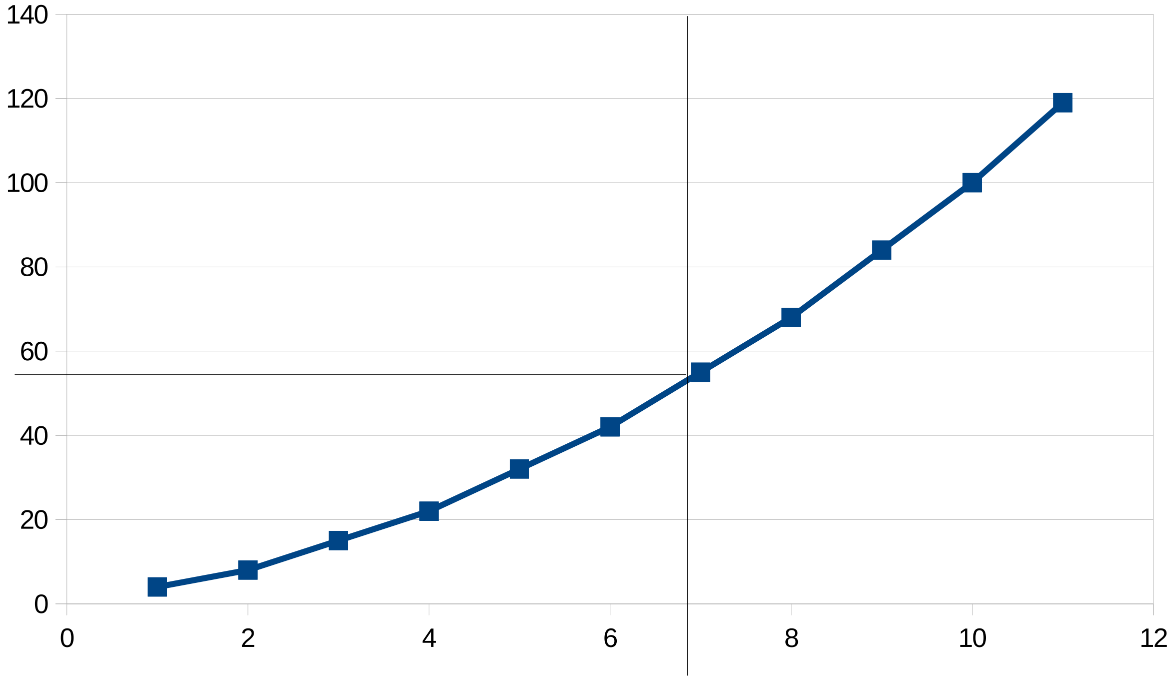
$$n \le m + \frac{1}{2} m T + \frac{1}{4} m^2 T + (1-(-1)^m) \frac{1}{8} T$$

$$48 \le m + \frac{3}{2} m + \frac{3}{4} m^2 + (1-(-1)^m) \frac{3}{8}$$

- total n =48 bits
- m =7 groups
- i-th group has x_i bits (size)
- constant skip delay T = $T(x_i)$ =3

1	4
2	8
3	15
4	22
5	32
6	42
7	55
8	68
9	84
10	100

$48 < 55$

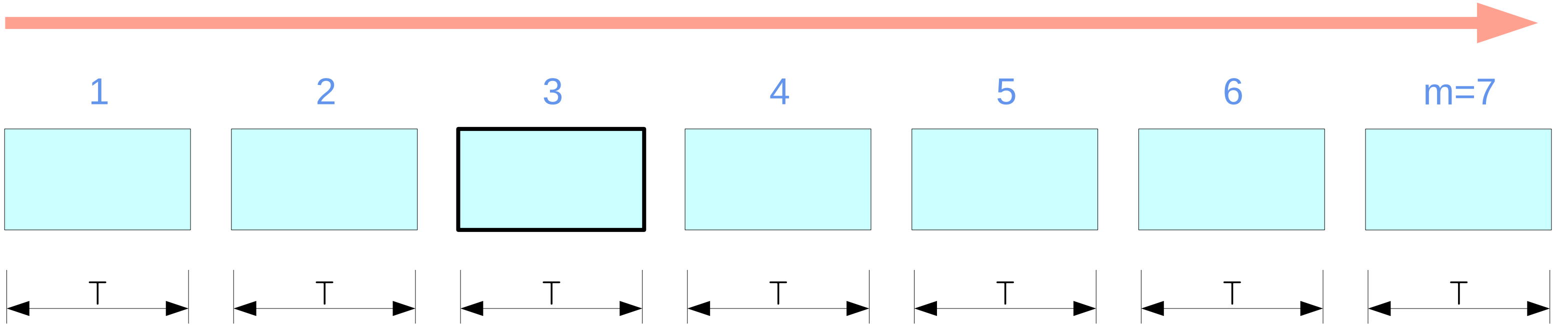


$m = 7$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Example 1 - (4)

n =48
m =7
T =3



$$y_i = min\{1+iT, 1+(m+1-i)T\}, \quad i = 1,\dots,m$$

$$y_1 = min\{1+1\cdot T, 1+7\cdot T\} = 1+1\cdot T = 4 \qquad 0 \le x_1 \le 1+1\cdot T = 4$$

$$y_2 = min\{1+2\cdot T, 1+6\cdot T\} = 1+2\cdot T = 7 \qquad 0 \le x_2 \le 1+2\cdot T = 7$$

$$y_3 = min\{1+3\cdot T, 1+5\cdot T\} = 1+3\cdot T = 10 \qquad 0 \le x_3 \le 1+3\cdot T = 10$$

$$y_4 = min\{1+4\cdot T, 1+4\cdot T\} = 1+4\cdot T = 13 \qquad 0 \le x_4 \le 1+4\cdot T = 13$$

$$y_5 = min\{1+5\cdot T, 1+3\cdot T\} = 1+3\cdot T = 10 \qquad 0 \le x_5 \le 1+3\cdot T = 10$$

$$y_6 = min\{1+6\cdot T, 1+2\cdot T\} = 1+2\cdot T = 7 \qquad 0 \le x_6 \le 1+2\cdot T = 7$$

$$y_7 = min\{1+7\cdot T, 1+1\cdot T\} = 1+1\cdot T = 4 \qquad 0 \le x_7 \le 1+1\cdot T = 4$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

$$0 \le x_i \le y_i, i=1,\dots,m$$

Example 2 - (3)

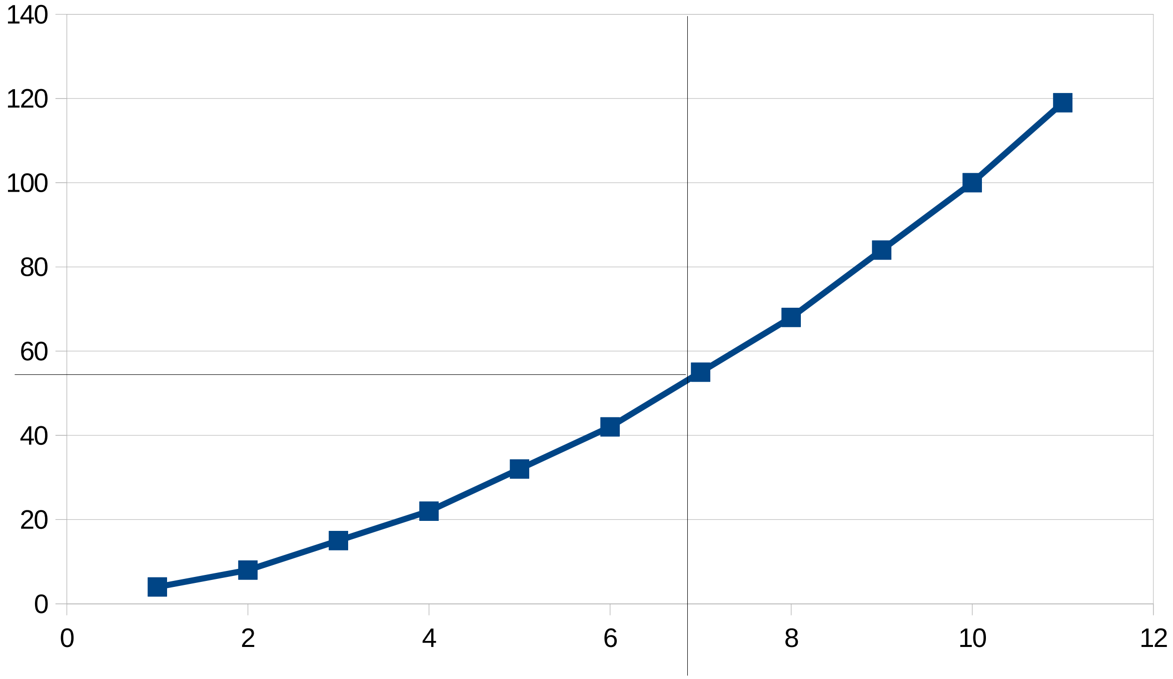
$$n \le m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \left(1 - (-1)^m\right) \frac{1}{8} T$$

$$54 \le m + \frac{3}{2} m + \frac{3}{4} m^2 + \left(1 - (-1)^m\right) \frac{3}{8}$$

- total n =54 bits
- m =7 groups
- i-th group has x_i bits (size)
- constant skip delay $T = T(x_i) = 3$

1	4
2	8
3	15
4	22
5	32
6	42
7	55
8	68
9	84
10	100

54 < 55

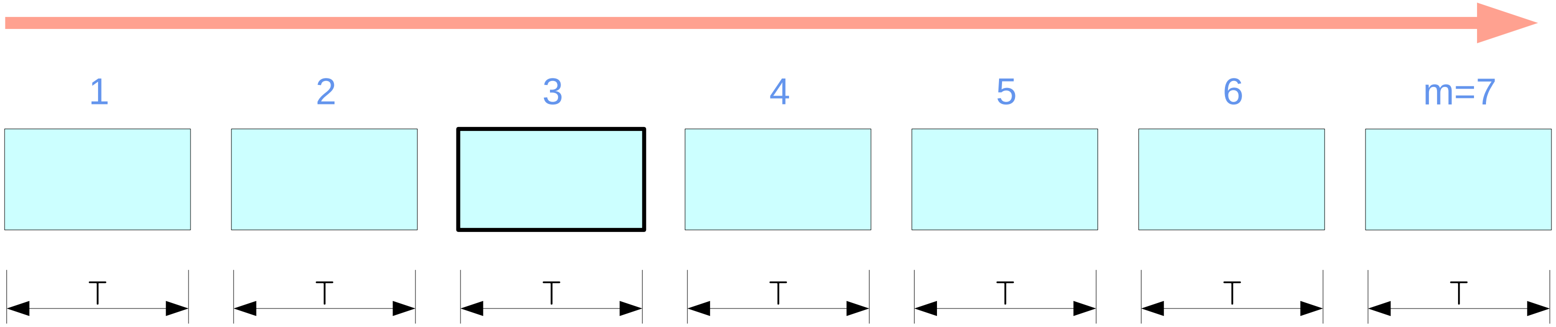


$m = 7$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Example 2 - (4)

n =54
m =7
T =3



$$y_i = min\{1+iT, 1+(m+1-i)T\}, \quad i = 1,\ldots,m$$

$$y_1 = min\{1+1\cdot T, 1+7\cdot T\} = 1+1\cdot T = 4 \qquad 0 \le x_1 \le 1+1\cdot T = 4$$

$$y_2 = min\{1+2\cdot T, 1+6\cdot T\} = 1+2\cdot T = 7 \qquad 0 \le x_2 \le 1+2\cdot T = 7$$

$$y_3 = min\{1+3\cdot T, 1+5\cdot T\} = 1+3\cdot T = 10 \qquad 0 \le x_3 \le 1+3\cdot T = 10$$

$$y_4 = min\{1+4\cdot T, 1+4\cdot T\} = 1+4\cdot T = 13 \qquad 0 \le x_4 \le 1+4\cdot T = 13$$

$$y_5 = min\{1+5\cdot T, 1+3\cdot T\} = 1+3\cdot T = 10 \qquad 0 \le x_5 \le 1+3\cdot T = 10$$

$$y_6 = min\{1+6\cdot T, 1+2\cdot T\} = 1+2\cdot T = 7 \qquad 0 \le x_6 \le 1+2\cdot T = 7$$

$$y_7 = min\{1+7\cdot T, 1+1\cdot T\} = 1+1\cdot T = 4 \qquad 0 \le x_7 \le 1+1\cdot T = 4$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

$$0 \le x_i \le y_i, i = 1,\ldots,m$$

Example 3 - (3)

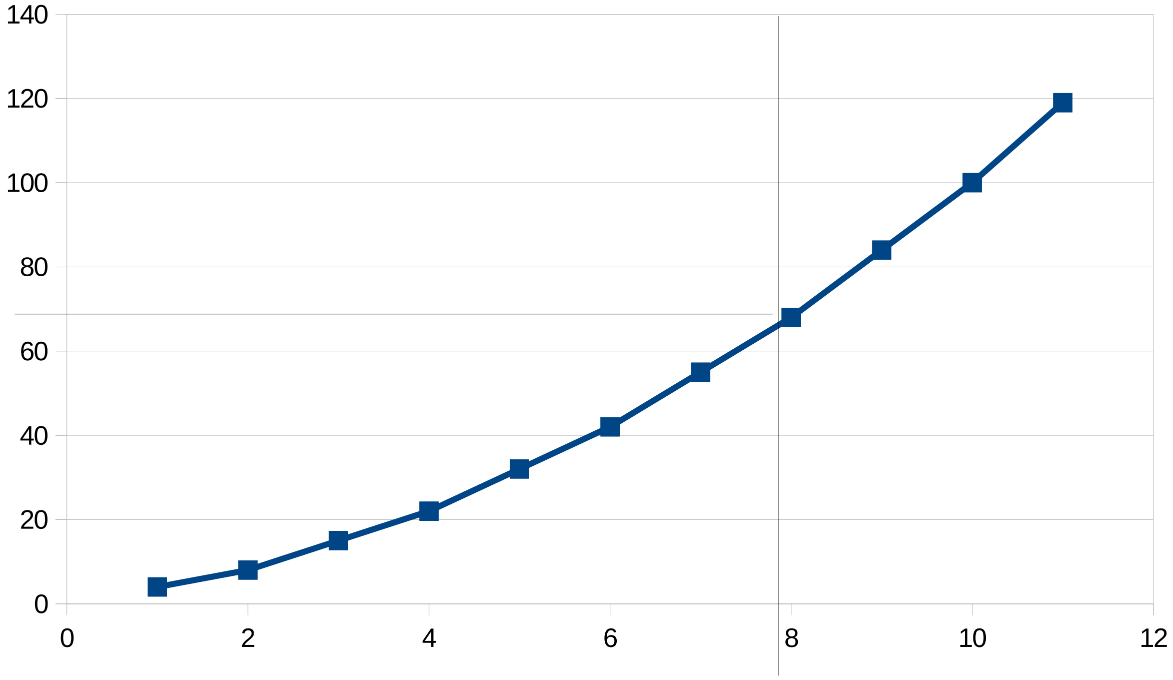
$$n \;\le\; m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \left(1-(-1)^m\right)\frac{1}{8} T$$

$$64 \;\le\; m + \frac{3}{2} m + \frac{3}{4} m^2 + \left(1-(-1)^m\right)\frac{3}{8}$$

- total n =64 bits
- m =7 groups
- i-th group has x_i bits (size)
- constant skip delay $T = T(x_i) = 3$

1	4
2	8
3	15
4	22
5	32
6	42
7	55
8	68
9	84
10	100

$64 < 68$

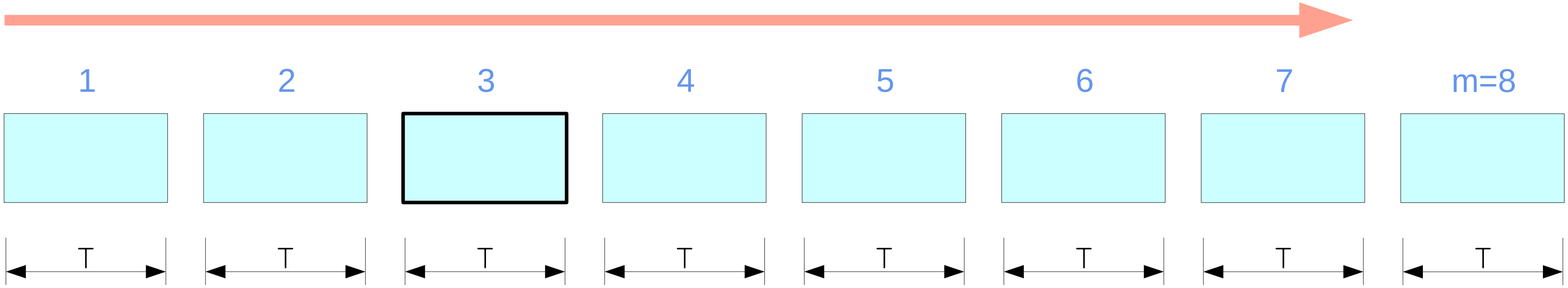


$m = 8$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Example 3 - (4)

n =64
m =8
T =3



$$y_i = min\{1+iT, 1+(m+1-i)T\}, \quad i = 1,\ldots,m$$

$$y_1 = min\{1+1\cdot T, 1+8\cdot T\} = 1+1\cdot T = 4 \qquad 0 \le x_1 \le 1+1\cdot T = 4$$

$$y_2 = min\{1+2\cdot T, 1+7\cdot T\} = 1+2\cdot T = 7 \qquad 0 \le x_2 \le 1+2\cdot T = 7$$

$$y_3 = min\{1+3\cdot T, 1+6\cdot T\} = 1+3\cdot T = 10 \qquad 0 \le x_3 \le 1+3\cdot T = 10$$

$$y_4 = min\{1+4\cdot T, 1+5\cdot T\} = 1+4\cdot T = 13 \qquad 0 \le x_4 \le 1+4\cdot T = 13$$

$$y_5 = min\{1+5\cdot T, 1+4\cdot T\} = 1+4\cdot T = 13 \qquad 0 \le x_5 \le 1+4\cdot T = 13$$

$$y_6 = min\{1+6\cdot T, 1+3\cdot T\} = 1+3\cdot T = 10 \qquad 0 \le x_6 \le 1+3\cdot T = 10$$

$$y_7 = min\{1+7\cdot T, 1+2\cdot T\} = 1+2\cdot T = 7 \qquad 0 \le x_7 \le 1+2\cdot T = 7$$

$$y_8 = min\{1+8\cdot T, 1+1\cdot T\} = 1+1\cdot T = 4 \qquad 0 \le x_8 \le 1+1\cdot T = 4$$

$$0 \le x_i \le y_i, i = 1,\ldots,m$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Closed formula for $\mathbf{y_1 + y_2 + \ldots + y_m}$

$$\sum_{i=1}^{m} y_i = m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \left(1 - (-1)^m\right) \frac{1}{8} T$$

$$\sum_{i=1}^{m} y_i = m + \frac{1}{2} m T + \frac{1}{4} m^2 T \qquad (even \quad m)$$

$$\sum_{i=1}^{m} y_i = m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \frac{1}{4} T \qquad (odd \quad m)$$

$$y_i = min\{1 + iT, 1 + (m+1-i)T\}, \quad i = 1, \ldots, m$$

$$\begin{array}{llll} y_1 = 1 + 1 \cdot T & \qquad & y_m = 1 + 1 \cdot T \\ y_2 = 1 + 2 \cdot T & & y_{m-1} = 1 + 2 \cdot T \\ y_3 = 1 + 3 \cdot T & & y_{m-2} = 1 + 3 \cdot T \\ \quad \vdots & & \quad \vdots \end{array}$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Closed formula for <u>even</u> **m** case (1)

$m = 2k$

$\frac{m}{2} = k$

$$y_i = \min\{1+iT,\quad 1+(m+1-i)T\},\quad i = 1,\dots,m$$

$m \quad \frac{1}{2}\cdot k(k+1)$

$$\begin{array}{llllll}
y_1 = \min\{1+1\cdot T, & 1+(m-0)\cdot T\} & 0 \le x_1 & \le & 1+1\cdot T \\
y_2 = \min\{1+2\cdot T, & 1+(m-1)\cdot T\} & 0 \le x_2 & \le & 1+2\cdot T \\
y_3 = \min\{1+3\cdot T, & 1+(m-2)\cdot T\} & 0 \le x_3 & \le & 1+3\cdot T \\
\quad\quad\downarrow & \quad\quad\uparrow & & & \quad\downarrow \\
y_k = \min\{1+k\cdot T, & 1+(k+1)\cdot T\} & 0 \le x_k & \le & 1+k\cdot T \\
\hline
y_{k+1} = \min\{1+(k+1)\cdot T, & 1+k\cdot T\} & 0 \le x_{k+1} & \le & 1+k\cdot T \\
\quad\quad\downarrow & \quad\quad\uparrow & & & \quad\uparrow \\
y_{m-2} = \min\{1+(m-2)\cdot T, & 1+3\cdot T\} & 0 \le x_{m-2} & \le & 1+3\cdot T \\
y_{m-1} = \min\{1+(m-1)\cdot T, & 1+2\cdot T\} & 0 \le x_{m-1} & \le & 1+2\cdot T \\
y_{m-0} = \min\{1+(m-0)\cdot T, & 1+1\cdot T\} & 0 \le x_{m-0} & \le & 1+1\cdot T
\end{array}$$

$\frac{1}{2}\cdot k(k+1)$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

$$0 \le x_i \le y_i,\ i = 1,\dots,m$$

Closed formula for even **m** case (2)

$$1 + 2 + \cdots + k = \frac{1}{2}k(k+1)$$

$$\frac{n(a+l)}{2}$$

$$m + 2 \cdot \frac{1}{2}k(k+1)T$$

$$= m + k(k+1)T$$

$$= m + \frac{m}{2}\left(\frac{m}{2}+1\right)T$$

$$= m + \frac{m}{2}T + \frac{m^2}{4}T$$

$$m = 2k$$

$$\frac{m}{2} = k$$

$$even\ m \ : \ m + \frac{1}{2}mT + \frac{1}{4}m^2 T$$

$$odd\ m \ : \ m + \frac{1}{2}mT + \frac{1}{4}m^2 T + \frac{1}{4}T$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Closed formula for odd **m** case (1)

$m \ = \ 2k+1$

$m+1 = 2k+2$

$\dfrac{m+1}{2} = k+1$

$y_i \ = \ min\{1+iT, \quad 1+(m+1-i)T\}, \quad i \ = \ 1,\ldots,m \qquad m \quad \frac{1}{2}\cdot k(k+1)$

$$
\begin{array}{lllllll}
y_1 & = & min\{1+1\cdot T, & 1+(m-0)\cdot T\} & 0 \ \le \ x_1 & \le & 1+1\cdot T \\
y_2 & = & min\{1+2\cdot T, & 1+(m-1)\cdot T\} & 0 \ \le \ x_2 & \le & 1+2\cdot T \\
y_3 & = & min\{1+3\cdot T, & 1+(m-2)\cdot T\} & 0 \ \le \ x_3 & \le & 1+3\cdot T \\
y_k & = & min\{1+(k)\cdot T, & 1+(k+3)\cdot T\} & 0 \ \le \ x_k & \le & 1+k\cdot T \\
y_{k+1} & = & min\{1+(k+1)\cdot T, & 1+(k+2)\cdot T\} & 0 \ \le \ x_{k+1} & \le & 1+(k+1)\cdot T \quad (k+1) \\
y_{k+2} & = & min\{1+(k+2)\cdot T, & 1+(k+1)\cdot T\} & 0 \ \le \ x_{k+2} & \le & 1+k\cdot T \\
y_{m-2} & = & min\{1+(m-2)\cdot T, & 1+3\cdot T\} & 0 \ \le \ x_{m-2} & \le & 1+3\cdot T \\
y_{m-1} & = & min\{1+(m-1)\cdot T, & 1+2\cdot T\} & 0 \ \le \ x_{m-1} & \le & 1+2\cdot T \\
y_{m-0} & = & min\{1+(m-0)\cdot T, & 1+1\cdot T\} & 0 \ \le \ x_{m-0} & \le & 1+1\cdot T
\end{array}
$$

$\frac{1}{2}\cdot k(k+1)$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

$0 \le x_i \le y_i, i=1,\ldots,m$

Closed formula for odd **m** case (2)

$$1 + 2 + \cdots + k = \frac{1}{2}k(k+1) \qquad \frac{n(a+l)}{2}$$

$$m + 2 \cdot \frac{1}{2}k(k+1)T + (k+1)T$$

$$= m + k(k+1)T + (k+1)T$$

$$= m + (k+1)^2 T$$

$$= m + \left(\frac{m+1}{2}\right)^2 T$$

$$= m + \frac{m^2}{4}T + \frac{m}{2}T + \frac{1}{4}T$$

$$m = 2k+1$$

$$m+1 = 2k+2$$

$$\frac{m+1}{2} = k+1$$

$$even\ m \ : \ m + \frac{1}{2}mT + \frac{1}{4}m^2 T$$

$$odd\ m \ : \ m + \frac{1}{2}mT + \frac{1}{4}m^2 T + \frac{1}{4}T$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Round and truncate values

$$round\left(x\right) = truncate\left(x+0.5\right)$$

$$round\left(3.2\right) = truncate\left(3.2+0.5\right) = 3$$

$$round\left(3.7\right) = truncate\left(3.7+0.5\right) = 4$$

m	m/2	(m/2)^2	sum	sum + 1/4
1	0+0.5	0+0.25	0+0.75	1
2	1	1	2	2.25
3	1+0.5	2+0.25	3+0.75	4
4	2	4	6	6.25
5	2+0.5	6+0.25	8+0.75	9
6	3	9	12	12.25
7	3+0.5	12+0.25	15+0.75	16
8	4	16	20	20.25
9	4+0.5	20+0.25	24+0.75	25
10	5	25	30	30.25
11	5+0.5	30+0.25	35+0.75	36
12	6	36	42	42.25

$$round\left(\frac{m}{2} + \frac{m^2}{4}\right) = truncate\left(\frac{m}{2} + \frac{m^2}{4} + \frac{1}{4}\right)$$

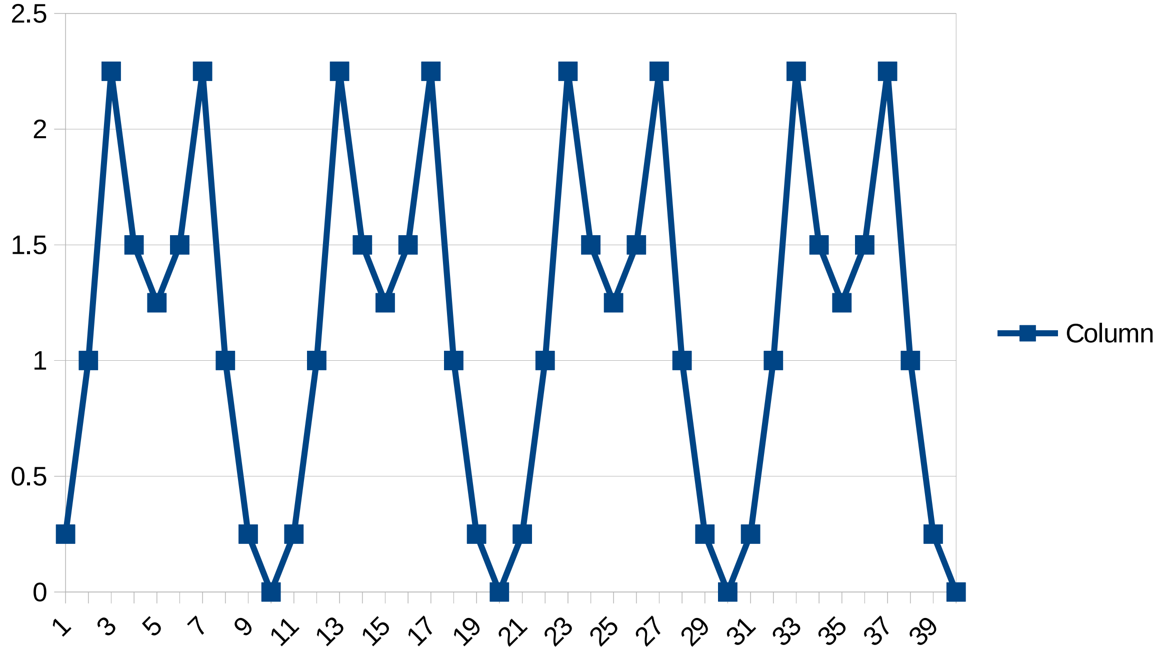
$$\left(\frac{m}{2} + \left(\frac{m}{2}\right)^2 + \frac{1}{4}\right) = \frac{1}{4}\left(2m+m^2+1\right) = \frac{1}{4}\left(m+1\right)^2$$

$$even\ m\ :\ m + \frac{1}{2}mT + \frac{1}{4}m^2T$$

$$odd\ m\ :\ m + \frac{1}{2}mT + \frac{1}{4}m^2T + \frac{1}{4}T$$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Values of $m^2/4$



Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

lsd = least significant digit

m	m^2	$lsd(m^2)$	$lsd(m^2)/4$
1	1	1	0.25
2	4	4	1
3	9	9	2.25
4	16	6	1.5
5	25	5	1.25
6	36	6	1.5
7	49	9	2.25
8	64	4	1
9	81	1	0.25
10	100	0	0
11	121	1	0.25
12	144	4	1
13	169	9	2.25
14	196	6	1.5
15	225	5	1.25
16	256	6	1.5
17	289	9	2.25
18	324	4	1
19	361	1	0.25
20	400	0	0
21	441	1	0.25
22	484	4	1
23	529	9	2.25
24	576	6	1.5
25	625	5	1.25
26	676	6	1.5
27	729	9	2.25
28	784	4	1
2			

Procedure (4)

- n bits
- m groups

x_i delay$_{\text{ripple}}$ ripple delay of a group

$$y_i = \min\{1+iT, 1+(m+1-i)T\}$$

min {delay1$_{\text{skip}}$, delay2$_{\text{skip}}$}

skip delay over a group

$$0 \le x_i \le y_i, \quad i=1,\dots,m$$

delay$_{\text{ripple}}$ ≤ delay1$_{\text{skip}}$
delay$_{\text{ripple}}$ ≤ delay2$_{\text{skip}}$

find the smallest positive integer m such that

$$n \le m + \frac{1}{2}mT + \frac{1}{4}m^2T + (1-(-1)^m)\frac{1}{8}T$$

$$n \le \sum_{i=1}^{m} y_i$$

so these are at least n unit squares
in the histogram,
starting with the first row,
row by row

Let x_i denote the number of
shaded squares in column i of the histogram,
$i = 1, \dots, m$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Procedure (1)

- total n =48 bits
- m =7 groups
- i-th group has x_i bits (size)
- constant skip delay $T = T(x_i)$ =3

(I) Let m be the smallest positive integer such that

$$n \leq m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \left(1 - (-1)^m\right) \frac{1}{8} T$$

(II) Let

$$y_i = min\{1 + i T, 1 + (m + 1 - i) T\}, \quad i = 1, \ldots, m$$

and construct a histogram whose *i-th* column has height y_i

for example, for T=3, and n=48, we have m=7

(III) It is easily verified that the area of the histogram in (II) is

$$m + \frac{1}{2} m T + \frac{1}{4} m^2 T + \left(1 - (-1)^m\right) \frac{1}{8} T \geq n$$

so these are at least n unit squares in the histogram
starting with the first row, shade in n of the squares, row by row
Let x_i denote the number of shaded squares in column i of the histogram,
$i = 1, \ldots, m$

Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers