Multi-level Variable Block Adder (1A)

-
-

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

This document was produced by using OpenOffice and Octave.

Dividing groups into blocks

It is clear that the maximum delay of a carry signal in a carry skip adder
Can be further reduced if signals are allowed to skip over blocks of groups
We define a block to be an additional path allowing carry signal to skip
Directly over groups.
We will describe an efficient scheme for dividing the carry chain into blocks of groups
We assume that the time required for a carry signal to skip over a block of groups is T_b.
Actually longer than the time T_g required to skip over a group
But for the sake of simplifying the analysis we will assume these two times to be
Equal i.e. $T_b = T_g$.
However, out techinque extends to the case where $T_g \neq T_b$

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

Dividing groups into blocks

Let M denote the number of blocks into which the groups of bits are divided
Let D denote the maximum delay a carry signal can have in an adder divided into M blocks
Clearly, $D \geq MT$.
We will show how to choose the blocks such that $D = MT$
We will also show how to choose M for an adder of length n

Our blocks are chosen in such a way that the maximum delay of a signal originating and terminating in block i and $M + 1 - i$ is iT

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

Dividing groups into blocks

Consider a signal originating in the first of these blocks and terminating
In the second.
Such a signal will skip over $M - 2i$ blocks and will accordingly have
Delay $\leq (iT) + (M - 2i)T + iT = MT$ as desired
It follows from our work in section 2 that in order for a signal originating
And terminating in block i to have delay less or equal iT
We must choose the length of the ith and ($M + 1 - i$)th blocks
To be less or equal the number of unit squares in a histogram
With base of width i
Thus the maximum length of the ith and ($M + 1 - i$)th blocks must be

$$i + \frac{1}{2} iT + \frac{1}{4} i^2 T + \left(1 - \left(-1\right)^i\right) \frac{T}{8}, \quad i \leq ceil\left(\frac{M}{2}\right)$$

To denote the smallest inter >= I

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

Dividing groups into blocks

Thus for a given adder length n, we choose M to be the smallest positive integer
Such that the expression (3.1) exceeds or equals n

$$2\sum_{i=1}^{ceil\left(\frac{M-1}{2}\right)}\left\{ i \;+\; \frac{1}{2}iT \;+\; \frac{1}{4}i^{2}T \;+\; (1-(-1)^{i})\frac{T}{8}\right\}$$

$$+\;\frac{(1-(-1)^{M})}{2}\left\{ceil\left(\frac{M}{2}\right) \;+\; \frac{1}{2}ceil\left(\frac{M}{2}\right)T \;+\; \frac{1}{4}ceil^{2}\left(\frac{M}{2}\right)T \;+\; (1-(-1)^{ceil\left(\frac{M}{2}\right)})\frac{T}{8}\right\}$$

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

Dividing groups into blocks

M is then the numer of blocks into which our adder must be divided.
The formal statement of our algorithm is as follows

3(i) Choose M to be the smallest positive integer such that

3(ii) Form M blocks labeled 1, 2, …, M with block I and M+1-i each containing

3(iii) treat each of the final blocks in 3(ii) as a complete carry chain and
Divide it into groups optimally using the algorithm 2(i) – 2(iii)

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

Dividing groups into blocks

3(i) Choose M to be the smallest positive integer such that

$$n \;\le\; 2\sum_{i=1}^{ceil\left(\frac{M-1}{2}\right)} \left\{ i \;+\; \frac{1}{2} iT \;+\; \frac{1}{4} i^2 T \;+\; \left(1-(-1)^i\right)\frac{T}{8} \right\}$$

$$+\; \frac{\left(1-(-1)^M\right)}{2} \left\{ ceil\left(\frac{M}{2}\right) \;+\; \frac{1}{2} ceil\left(\frac{M}{2}\right) T \;+\; \frac{1}{4} ceil^2\left(\frac{M}{2}\right) T \;+\; \left(1-(-1)^{ceil\left(\frac{M}{2}\right)}\right)\frac{T}{8} \right\}$$

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

Dividing groups into blocks

3(ii) Form *M* blocks labeled 1, 2, …, *M* with block *i* and *M*+1-*i* each containing

$$\left\{ i + \frac{1}{2} iT + \frac{1}{4} i^2 T + \left(1 - (-1)^i\right) \frac{T}{8} \right\} \text{ bits}$$

$$i \le ceil\left(\frac{M}{2}\right)$$

This construction is analogous to the construction of the histogram

If necessary, delete bits from the largest blocks in this chain
Until a total of exactly *n* bits remain in the M blocks

3(iii) treat each of the final blocks in 3(ii) as a complete carry chain and
Divide it into groups optimally using the algorithm 2(i) – 2(iii)

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

Dividing groups into blocks example 1

Consider a 32-bit adder

For *i* = 1, 2, 3 … , and *T* = 3 the numbers

$$\left\{ i + \frac{1}{2} iT + \frac{1}{4} i^2 T + (1-(-1)^i)\frac{T}{8} \right\}$$

Take on the values 4, 8, 15, 22, 32, … respectively

Since $32 \le 24+8+15$

We must have *M*=5 blocks in step 3(ii)

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

Dividing groups into blocks example 1

These blocks have sizes 4, 8, 15, 8, 4 respectively

If we delete 7 units from the middle block

We obtain block sizes 4, 8, 8, 8, 4 which add up to 32

Dividing each block into groups by the procedure 2(i)-2(iii)

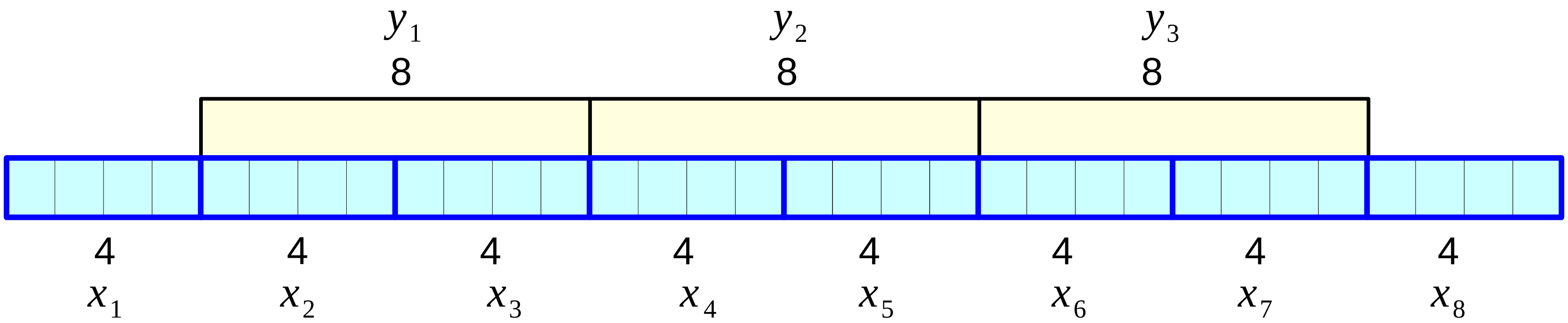
We obtain the following chain where each group has size 4

The maximum delay of a carry signal is *MT* = 15

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

Dividing groups into blocks example 1

We obtain block sizes 4, 8, 8, 8, 4 which add up to 32



$n = 32$ $\quad D = MT = 15$

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

Dividing groups into blocks example 2

Consider a 48-bit adder
Again we assum T = 3

Since $48 \le 24+8+15$

We must have M=6 corresponding to
block sizes 4, 8, 15, 15, 8, 4

The total number of units is 54

So we reduce the size of the two middle blocks by 3 each
This gives block sizes 4, 8, 12, 12, 8, 4 adding up to 48.

If we divide each block into groups by the procedure 2(i)-2(iii)

Each group has size 4
The maximum delay of a carry signal is given by MT=18

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

Dividing groups into blocks example 3

Consider a 64-bit adder
Again we assum T = 3

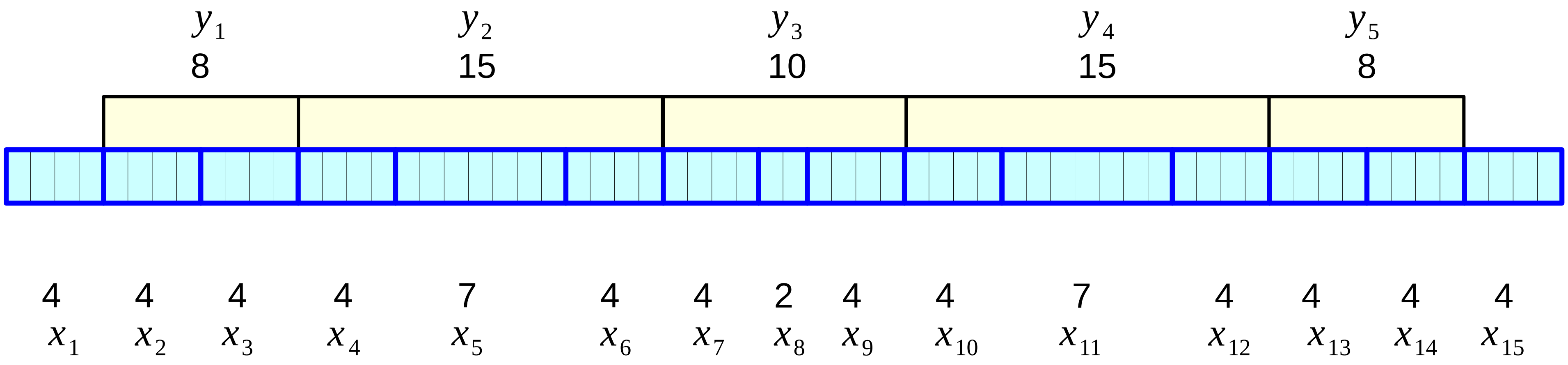
Since $64 \leq 24+8+15+22$

We take M=7 and start with blocks of sizes 4, 8, 15, 22, 15, 8, 4, respectively
The lengths of these blocks total 76.
So we reduce the middle block by 12.
The new block sizes are 4, 8, 15, 10, 15, 8, 4
The optimal division of these blocks into groups is given in Figure
We could have just as well reduced the sizes of the three middle blocks
Obtaining final blocks of sizes 4, 8, 13, 14, 13, 8, 4.
The maximum delay of a carry signal would still be MT = 21

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

Dividing groups into blocks example 1

The new block sizes are 4, 8, 15, 10, 15, 8, 4



$n = 64$ $D = MT = 21$

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

Ripple delay and skip delay

For a 32-bit adder, and the VBA scheme,
we divided the Carry chain into blocks of sizes
1, ,3, 5, 7, 7, 5, 3, 1.

Why this division is optimal?

Let *t* denote the time required for a carry signal to ripple across a one bit in the carry chain, and let *T* denote the time required for the signal to skip over a group of bits.

By simulation of the blocks, we have found that
t = 0.8 ns and *T*=165ns.
To simplify our analysis, we normalize them
So that *t* = 1 and *T* = 2.

Then we apply the theory developed
for finding the optimal division of a carry chain

$\Delta_{rca} = 1$ ripple delay over a bit

$\Delta_{SKIP} = T$ skip delay over a group

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

Delay model

For n=32, we have m=7, (y1, y2, y3, y4, y5, y6, y7) = (3,5,7,9,7,5,3)
The above algorithm gives (x1, x2, x3, x4, x5, x6, x7) = (3,5,5,6,5,5,3)

A carry chain divided in this way has maximum delay D = mT =14
Since one unit of delay is 0.8ns, the maximum delay for 32-bit carry chain
is D = 14*0.8ns = 11.2ns
This time involves only the delay in the carry chain

It is easy to check that this is also the delay for a chain divided into groups of sizes 1,3,5,7,7,5,3,1.
Thus this is also an optimal subdivision

The worst case delay includes the time needed to generate p_i and g_i signals
Delay of the carry chain, and the time for producing last sum bit s_n

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

Determining ***m*** (method 1)

Let *m* be the smallest positive integer such that

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

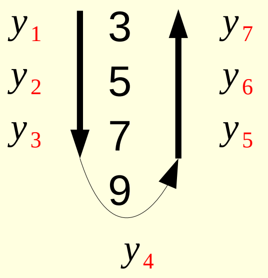
$$y_i = min\{1+iT, 1+(m+1-i)T\}, \quad i = 1, \ldots, m$$

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

n =32
m =7
T =2

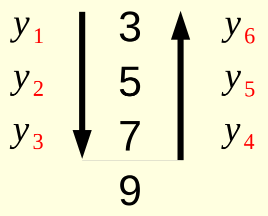
if *T* = 2

$m=7 \ (odd)$



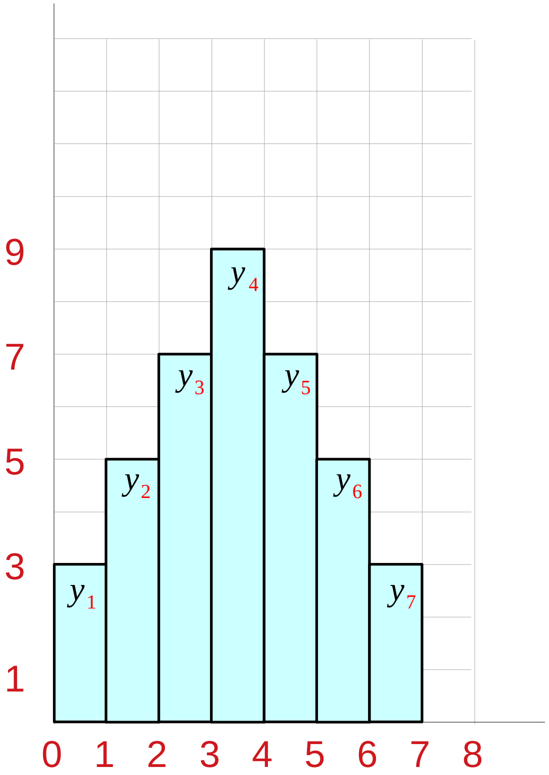
$$\sum_{i=1}^{7} y_i = 39$$

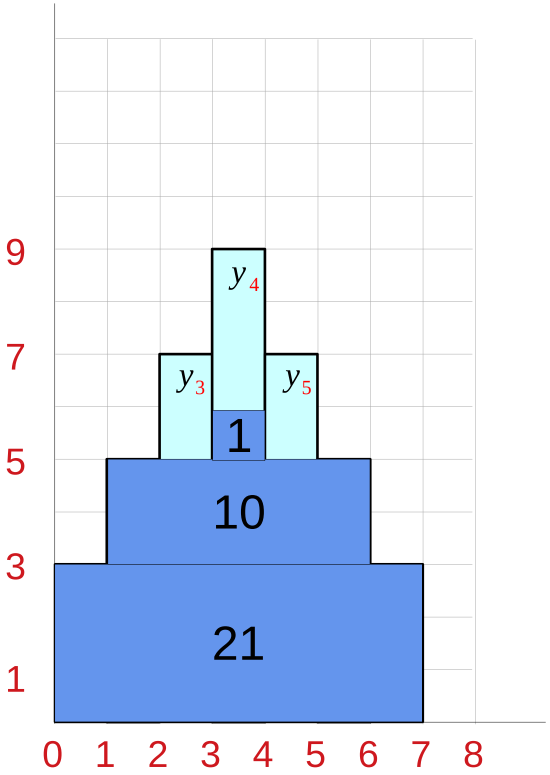
$m=6 \ (even)$

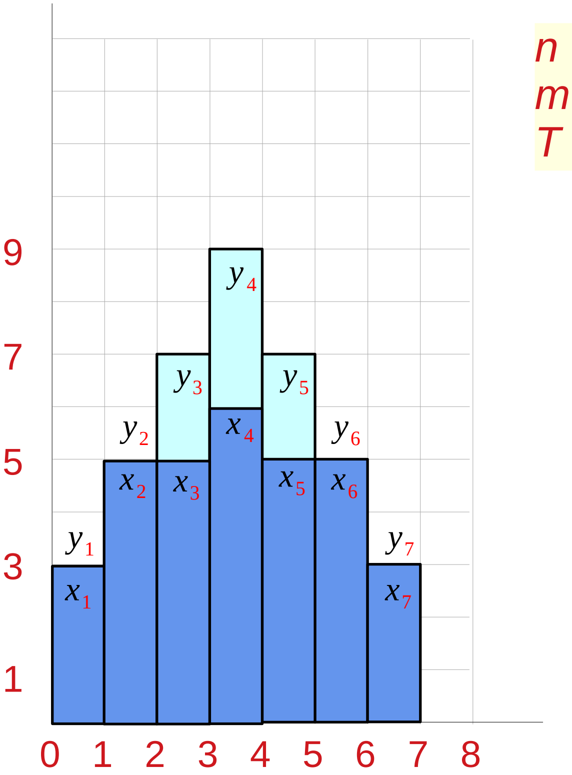


$$\sum_{i=1}^{8} y_i = 30$$

Example 1 - (2)



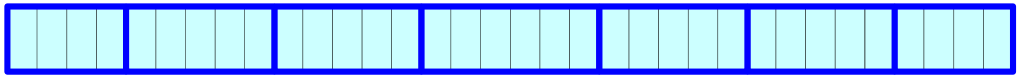




n =32
m =7
T =2

$n = 32$

$21+10+1=32 < 39 = \sum_{i=1}^{7} y_i$ ⇒ $m = 7$



$x_1=3$ $x_2=5$ $x_3=5$ $x_4=6$ $x_5=5$ $x_6=5$ $x_7=4$

Comparison

In the literature, comparison of schemes for ALU implementation is done
Mainly on the basis of the number of gates, propagation delay per gate
And power consumed per gate

However, if a VLSI implementation of a high-speed ALU is considered,
These measures are not easily applied.

For example, the propagation delay in terms of number of gates is not
An adquate measure unless care is taken to implement the function exactly
As specified by its logic (gate) representation

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

Comparison

This is often not the case since the function is frequently merged into a group
Of transistors or implemented by using pass-transistors, precharge or other
Techniques applied by the circuit designer in order to minimize delay and power

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

Comparison

In general, if the function is implemented in two levels of logic,
The delay is not necessarily smaller than the implementation of the
Same function in three or more logic levels.
This is due to the fact that in n-MOS technology the delay is heavily
Dependent on several factors

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

Comparison

1. Gate type : NOR gates are faster than NAND gates
2. Fan-in : for a NAND gate, the delay is directly proportional to the number
Of inputs, since inside the n-input NAND gate the signal has to propagate through
N-transistors. In case of anOR gate, delay is not stronly affected by the number of
Inputs, and therefore the use of NOR gate is preferable
3. Fan-out : the speed of a gate will be different if the fan-out is larger
Than in the case of small fan-outs
4. Wiring : speed is also dependent on the length of the wires, I.e, wiring
capacitance and the resistance of the long wires

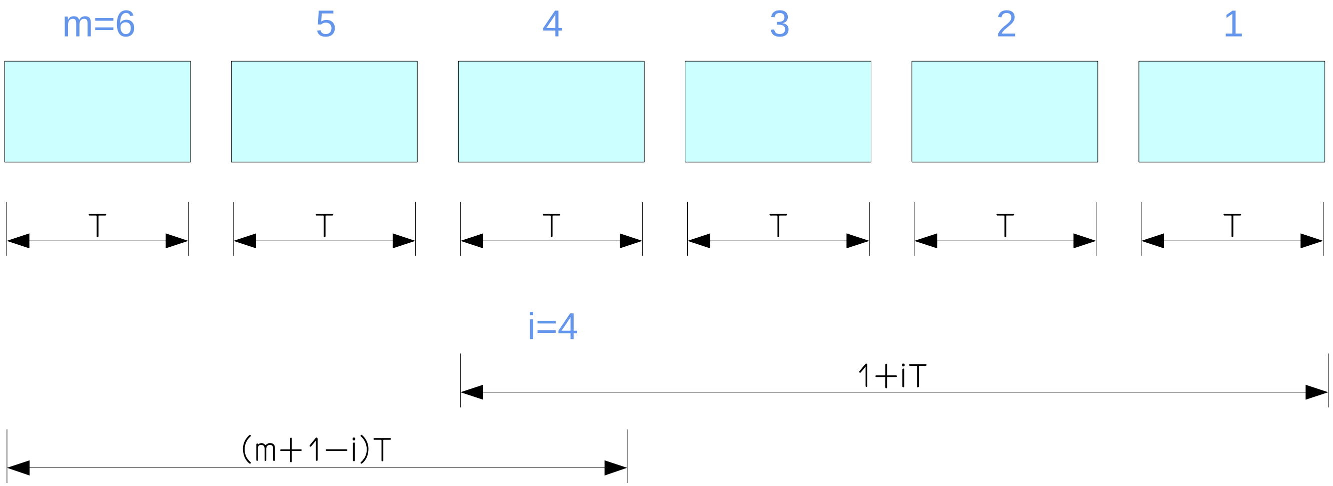
Oklobdzija: High-Speed VLSI arithmetic units : adders and multipliers

Comparison

In this paper, we consider the problem of designing a carry-skip adder
In FET technology and give some optimal solutions
Actually, our solutions are more general in that we generally assume
That the time required for a carry signal to skip over a group of bits
Is longer the time required for the carry to ripple through a single bit.
This assumption is relevant for adders designed in n-MOS technology
Lehman and Burla assumed their adder to be designed using discrete
Components where these two times are equal.
Our analysis will include their problem as a special case

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

Delay model



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

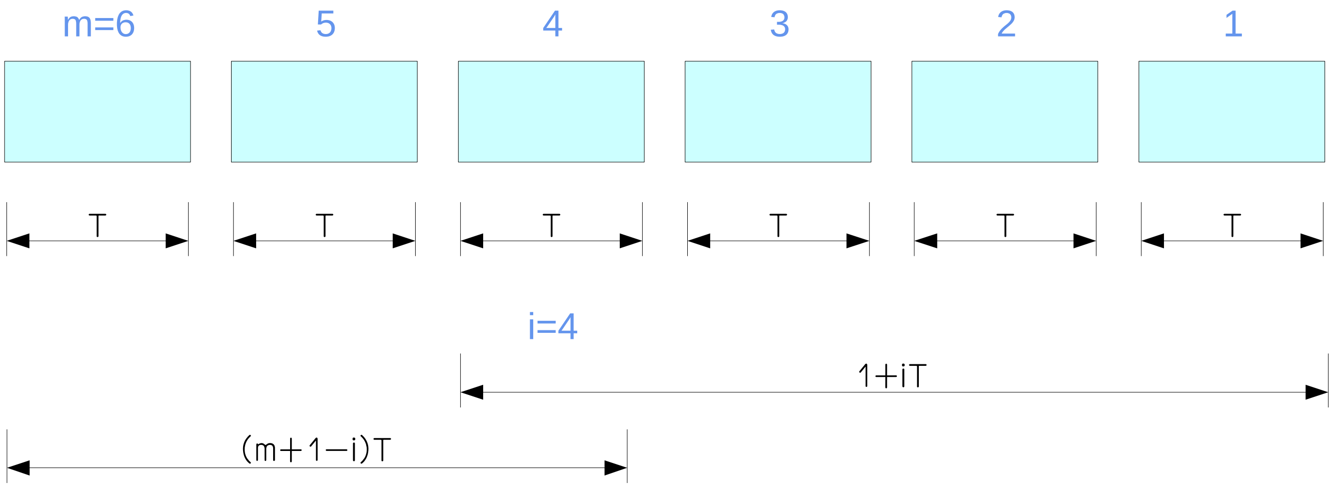
$$y_{1,} \ldots , y_m$$

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

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

$$\sum_{i=1}^{m} x_i = n$$

Delay model



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

$$y_{1,} \ldots, y_m$$

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

Delay model

Implement it with a string of multiplexers

The multiplexer cell is designed as very fast

Multiplexers are designed
as very fast structures using buffered pass gates and
in this sense are similar to the Manchester carry chain
which has been shown to be
the most effective implementation of a carry chain

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

Delay model

The implementation of a single carry block is done
by mixing a 4 to 1 multiplexer (actually used as a 3 to 1)

In the last stage with a string of 2 to 1 multiplexers

a carry bypass is connected to inputs 3 and 4
of the 4:1 multiplexer (group carry multiplexer)
and the selection of the carry bypass is activated
by the NAND gate singaling when the condition
for group propagate is reached and
activating the group multiplexer in turn.

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

Delay model

The32-bit implementation of the VBA adder is obtained
By connecting the groups of the sizes calculated
For the full length of n=32 bits

To increase the speed further we used a faster inverting version
Of the multiplexer, alternating between Ci and Cb_i signals

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