ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta jaderná a fyzikálně inženýrská

Katedra matematiky

VÝZKUMNÝ ÚKOL

Konstrukce algoritmů pro paralelní sčítání

Construction of algorithms for parallel addition

Vypracoval: Jan Legerský

Školitel: Ing. Štěpán Starosta, Ph.D.

Akademický rok: 2014/2015

Čestné prohlášení

Prohlašuji na tomto místě, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Praze dne ???

Jan Legerský

Poděkování

??

Jan Legerský

Název práce:	**Konstrukce algoritmů pro paralelní sčítání**
Autor:	Jan Legerský
Obor:	Inženýrská informatika
Zaměření:	Matematická informatika
Druh práce:	Výzkumný úkol
Vedoucí práce:	Ing. Štěpán Starosta, Ph.D., KM FIT, ČVUT v Praze
Konzultant:	—
Abstrakt:	ABSTRAKT
Klíčová slova:	Paralelní sčítání, nestandardní numerační systémy.

Title:	**Construction of algorithms for parallel addition**
Author:	Jan Legerský
Abstract:	ABSTRACT
Key words:	Parallel addition, non-standard numeration systems.

Obsah

Introduction

Chapter 1

Preliminaries

Preliminaries *Positional numeration system* $(\beta, \mathcal{A})$ is defined by

- *Base* $\beta \in \mathbb{C}, |\beta| > 1$.
- Finite *digit set* $\mathcal{A} \subset \mathbb{Z}$ containing 0, usually called *alphabet*.

A complex number x has a finite $(\beta, \mathcal{A})$-representation if $x = \sum_{j=-m}^{n} x_j \beta^j$ with coefficients x_j in $\mathcal{A}$.

$$x = x_n x_{n-1} \cdots x_1 x_0 \bullet x_{-1} x_{-2} \cdots x_{-m}$$

1.1 Previous results

Parallel Addition Introduced by Avizienis in 1961:

$$\begin{aligned} &\cdots w_{j+t+1} w_{j+t} \cdots w_{j+1} w_j w_{j-1} \cdots w_{j-r} w_{j-r-1} \cdots && , w_i \in \mathcal{A} + \mathcal{A}\,, \\ \longrightarrow &\cdots z_{j+t+1}\ z_{j+t}\ \cdots\ z_{j+1} z_j\ z_{j-1}\ \cdots z_{j-r}\ z_{j-r-1}\ \cdots && , z_i \in \mathcal{A}\,. \end{aligned}$$

Digit conversion is a mapping $(\mathcal{A} + \mathcal{A})^{t+r+1} \to \mathcal{A}$:

$$z_j = z_j(w_{j+t} \cdots w_{j+1} w_j w_{j-1} \cdots w_{j-r})\,.$$

Thus the parallel addition is done in constant time but the numeration system must be redundant – a number has more than one admissible representation.

For example:

$$\beta \in \mathbb{N}, \beta \geq 3, \mathcal{A} = \{-a, \ldots, 0, \ldots a\}, b/2 < a \leq b - 1\,.$$

Non-standard numeration systems
Integer alphabets:

- Base $\beta \in \mathbb{C}, |\beta| > 1$.
- Addition is computable in parallel if and only if β is an algebraic number with no conjugate of modulus 1 [Frougny, Heller, Pelantová, S.].
- Algorithms are known but with large alphabets.

Non-integer alphabets:

- Base $\beta \in \mathbb{Z}[\omega] = \left\{ \sum_{j=0}^{d-1} a_j \omega^j \ : a_j \in \mathbb{Z} \right\}$, where $\omega \in \mathbb{C}$ is an algebraic integer of degree d.
- Alphabet $\mathcal{A} \subset \mathbb{Z}[\omega], 0 \in \mathcal{A}$.
- Only few manually found algorithms.

PRO VSECHNY DEFINICE A VETY NENI-LI UVEDENO JINAK Let ω be an algebraic integer and $(\beta, \mathcal{A})$ be a numeration system such that a base $\beta \in \mathbb{Z}[\omega]$ and an alphabet $\mathcal{A}$ is a finite subset of $\mathbb{Z}[\omega]$.

Chapter 2

Method for construction of algorithms for parallel addition

2.1 Basic formula for digit set conversion

The general concept of addition (standard or parallel) in any numeration system $(\beta, \mathcal{A})$, such that $\mathrm{Fin}_{\mathcal{A}}(\beta)$ is closed under addition, is following: we add numbers digitwise and then we convert the result into the alphabet $\mathcal{A}$. Obviously, digitwise addition is computable in parallel, thus the crucial point is the conversion of the obtained result. It can be easily done in a standard way but a parallel digit set conversion is nontrivial. However, formulas are basically same but the choice of coefficients differs.

Now we go step by step more precisely. Let $x = \sum_{-m'}^{n'} x_i\beta^i, y = \sum_{-m'}^{n'} y_i\beta^i \in \mathrm{Fin}_{\mathcal{A}}(\beta)$ with $(\beta, \mathcal{A})$-representantions padded by zeros to have the same length. We set

$$\begin{aligned} w = x + y &= \sum_{-m'}^{n'} x_i\beta^i + \sum_{-m'}^{n'} y_i\beta^i = \sum_{-m'}^{n'} (x_i + y_i)\beta^i \\ &= \sum_{-m'}^{n'} w_i\beta^i\,, \end{aligned}$$

where $w_j = x_j + y_j \in \mathcal{A} + \mathcal{A}$. Thus, $w_{n'}w_{n'-1}\cdots w_1w_0 \bullet w_{-1}w_{-2}\cdots w_{-m'}$ is a $(\beta, \mathcal{A}+\mathcal{A})$-representation of $w \in \mathrm{Fin}_{\mathcal{A}+\mathcal{A}}(\beta)$.

We also use column notation of addition in the following, e.g.,

$$\begin{array}{c} x_{n'}\ x_{n'-1}\cdots x_1\ x_0 \bullet x_{-1}\ x_{-2}\ \cdots x_{-m'} \\ y_{n'}\ y_{n'-1}\cdots y_1\ \ y_0 \bullet y_{-1}\ y_{-2}\ \ \cdots y_{-m'} \\ \hline w_{n'}w_{n'-1}\cdots w_1w_0 \bullet w_{-1}w_{-2}\cdots w_{-m'}\,. \end{array}$$

As we want to obtain a $(\beta, \mathcal{A})$-representation of w, we search a sequence

$$z_nz_{n-1}\cdots z_1z_0z_{-1}z_{-2}\cdots z_{-m}$$

such that $z_j \in \mathcal{A}$ and

$$z_nz_{n-1}\cdots z_1z_0 \bullet z_{-1}z_{-2}\cdots z_{-m} = (w)_{\beta,\mathcal{A}}\,.$$

In the next, we consider without lost of generality only β-integers since modification for representations with rational part is obvious:

$$\beta^m \cdot z_n z_{n-1} \cdots z_1 z_0 \bullet z_{-1} z_{-2} \cdots z_{-m} = z_n z_{n-1} \cdots z_1 z_0 z_{-1} z_{-2} \cdots z_{-m} \bullet$$

Particulary, let $(w)_{\beta,\mathcal{A}+\mathcal{A}} = w_{n'} w_{n'-1} \cdots w_1 w_0 \bullet$. We search $n \in \mathbb{N}$ and $z_n, z_{n-1}, \ldots, z_1, z_0 \in \mathcal{A}$ such that $(w)_{\beta,\mathcal{A}} = z_n z_{n-1} \cdots z_1 z_0 \bullet$.

We use suitable representation of zero to convert digits w_j into the alphabet $\mathcal{A}$.

Definition 1. For a base $\beta \in \mathbb{Z}[\omega]$, a polynomial $R(x) = r_s x^s + r_{s-1} x^{s-1} + \cdots + r_1 x + r_0$ with coefficients $r_i \in \mathbb{Z}[\omega]$, such that $R(\beta) = 0$, is called a *rewriting rule*. MA TAM BYT Z[OMEGA] NEBO Z[BETA]???

An arbitrary rewriting rule may be used for conversion, then so called *core coefficient*, i.e., one of the coefficients which is greatest in modulus, is applied to convert a digit w_j. For our purpose, we use the simplest possible rewriting rule

$$R(x) = x - \beta \in \mathbb{Z}[\omega][x] \,.$$

As $0 = \beta^j \cdot R(\beta) = 1 \cdot \beta^{j+1} - \beta \cdot \beta^j$, we have a representation of zero

$$1(-\beta)\underbrace{0 \cdots 0}_{j} \bullet = (0)_\beta \,.$$

for all $j \in \mathbb{N}$. We multiply this representation by $q_j \in \mathbb{Z}[\omega]$ which is called a *weight coefficient* to obtain representation of zero

$$q_j(-q_j\beta)\underbrace{0 \cdots 0}_{j} \bullet = (0)_\beta \,.$$

This is digitwise added to $w_n w_{n-1} \cdots w_1 w_0 \bullet$ to convert the digit w_j into the alphabet $\mathcal{A}$. It causes a *carry* q_j on the $(j+1)$-th position from the conversion of j-th digit. The conversion runs from right ($j = 0$) to left until all digits and carries are converted into the alphabet $\mathcal{A}$:

$$\begin{array}{ccccccc}
w_n w_{n-1} & \cdots & w_{j+1} & w_j & w_{j-1} & \cdots & w_1 w_0 \bullet \\
 & & & & q_{j-2} & \iddots & \\
 & & & q_{j-1} & -\beta q_{j-1} & & \\
 & & q_j & -\beta q_j & & & \\
 & \iddots & -\beta q_{j+1} & & & & \\
\hline
z_{n+s} \cdots z_n z_{n-1} & \cdots & z_{j+1} & z_j & z_{j-1} & \cdots & z_1\ z_0 \bullet
\end{array}$$

Hence, the desired formula for conversion on the j-th position is

$$z_j = w_j + q_{j-1} - q_j \beta$$

for $j \in \mathbb{N}_0$. We set $q_{-1} = 0$ as there is no carry from the right on the 0-th position.

Clearly, the value of w is preserved:

$$\begin{aligned}\sum_{j\geq 0} z_j\beta^j &= w_0 - \beta q_0 + \sum_{j>0}(w_j + q_{j-1} - q_j\beta)\beta^j \\ &= \sum_{j\geq 0} w_j\beta^j + \sum_{j>0} q_{j-1}\beta^j - \sum_{j\geq 0} q_j\cdot\beta^{j+1} \\ &= \sum_{j\geq 0} w_j\beta^j + \sum_{j>0} q_{j-1}\beta^j - \sum_{j>0} q_{j-1}\cdot\beta^j \\ &= \sum_{j\geq 0} w_j\beta^j = w\,.\end{aligned}$$

The weight coefficient q_j must be chosen such that the converted digit is in the alphabet $\mathcal{A}$, i.e.,

$$z_j = w_j + q_{j-1} - q_j\beta \in \mathcal{A}\,. \tag{2.1}$$

Choice of weight coefficients is the crucial part in the case of construction of parallel addition algorithms. The method determining weight coefficients for a given input is decribed in Section 2.2.

On the other hand, it is trivial for standard numeration systems. Notice that

$$z_j \equiv w_j + q_{j-1} \mod \beta\,.$$

Assume now a standard numeration system $(\beta, \mathcal{A})$, where

$$\beta \in \mathbb{N}\,, \beta \geq 2\,, \mathcal{A} = \{0, 1, 2, \ldots, \beta - 1\}\,.$$

There is only one representative of each class modulo β. Therefore, the digit z_j is uniquely determined for a given digit $w_j \in \mathcal{A} + \mathcal{A}$ and carry q_{j-1} and thus so is the weight coefficient q_j. This means that $q_j = q_j(w_j, q_{j-1})$ for all $j \geq 0$. Generally,

$$q_j = q_j(w_j, q_{j-1}(w_{j-1}, q_{j-2})) = \cdots = q_j(w_j, \ldots, w_1, w_0)$$

and

$$z_j = z_j(w_j, \ldots, w_1, w_0)\,,$$

which implies that addition runs in linear time.

We want to the digit set conversion from $\mathcal{A} + \mathcal{A}$ into $\mathcal{A}$ be computable in parallel, i.e., there exist constants $r, t \in \mathbb{N}_0$ such that for all $j \geq 0$ is $z_j = z_j(w_{j+r}, \ldots, w_{j-t})$. To avoid the dependency on all less, respectivelly more, significant digits, we need variety in the choice of weight coefficient q_j. This implies that the used numeration system must be redundant.

NEJAKE LEMMA O TOM, ZE NEREDUNDANTNI SYSTEM VEDE NA LINEARNI CAS???

2.2 Method

In order to construct a parallel digit set conversion in numeration system $(\beta, \mathcal{A})$ we consider more general case of conversion from an *input alphabet* $\mathcal{B}$ such that $\mathcal{A} \subsetneq \mathcal{B} \subset \mathcal{A} + \mathcal{A}$ instead

of the alphabet $\mathcal{A}+\mathcal{A}$. As menshioned above, the key problem is to find for every $j \geq 0$ a weight coefficient q_j such that

$$z_j = \underbrace{w_j}_{\in\mathcal{B}} + q_{j-1} - q_j\beta \in \mathcal{A}$$

for any input $w \in \mathrm{Fin}_{\mathcal{B}}(\beta), (w)_{\beta,\mathcal{B}} = w_{n'}w_{n'-1}\dots w_1w_0\bullet$. We remark that the weight coefficient q_{j-1} is determined by the input w. For digit set conversion to be computable in parallel we demand to digit $z_j = z_j(w_{j+r},\dots,w_{j-t})$ for a fixed anticipation r and memory t in $\mathbb{N}_0$.

We introduce following definitions to obtain the desired digit set conversion.

Definition 2. Let $\mathcal{B}$ be a set such that $\mathcal{A} \subsetneq \mathcal{B} \subset \mathcal{A}+\mathcal{A}$. Then any set $\mathcal{Q} \subset \mathbb{Z}[\omega]$ containing 0 such that

$$\mathcal{B} + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}$$

is called a *set of weight coefficients*.

We see that

$$(\forall w_j \in \mathcal{B})(\forall q_{j-1} \in \mathcal{Q})(\exists q_j \in \mathcal{Q})(w_j + q_{j-1} - q_j\beta \in \mathcal{A})\,.$$

Accordingly, there is a weight coefficient $q_j \in \mathcal{Q}$ for any carry from the right $q_{j-1} \in \mathcal{Q}$ and any digit w_j in the input alphabet $\mathcal{B}$. I.e., we satisfy basic conversion formula (2.1). Notice that $q_{-1} = 0$ is in $\mathcal{Q}$ by definition. Thus, all weight coeficients may be chosen from $\mathcal{Q}$ inductively.

Definition 3. Let M be an integer and $q : \mathcal{B}^M \to \mathcal{Q}$ be a mapping such that

$$w_j + q(w_{j-1},\dots,w_{j-M}) - \beta q(w_j,\dots,w_{j-M+1}) \in \mathcal{A}$$

for all $w_j, w_{j-1},\dots,w_{j-M} \in \mathcal{B}$. Then q is called a *weight function* and M is called a *length of window*.

JE TREBA POZADOVAT q(0,...,0)=0 NEBO TO Z NECEHO PLYNE?

Having a weight function q, we define a function $\phi : \mathcal{B}^{M+1} \to \mathcal{A}$ by

$$\phi(w_j,\dots,w_{j-M}) = w_j + \underbrace{q(w_{j-1},\dots,w_{j-M})}_{=q_{j-1}} - \beta\,\underbrace{q(w_j,\dots,w_{j-M+1})}_{=q_j} =: z_j\,,$$

which verifies that the conversion is indeed $(M+1)$-local function with anticipation $r = 0$ and memory $t = M$.

The construction of a parallel conversion algorithm by so-called *extending window method* consists of two phases. In the first one, we find a minimal possible weight coefficient set $\mathcal{Q}$. It serves as the starting point for the second phase in which e increment the expected length of the window M until the weight function q is uniquely defined for each $(w_j, w_{j-1},\dots,w_{j-M+1}) \in \mathcal{B}^M$.

2.2.1 Phase 1 – Weight coefficient set

The goal of the first phase is to compute a weight coefficient set $\mathcal{Q}$, i.e., to find a set $\mathcal{Q} \ni 0$ such that

$$\mathcal{B} + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}\,.$$

We build $\mathcal{Q}$ iteratively so that we extend $\mathcal{Q}$ in a way to cover all elements on the left-hand side with original $\mathcal{Q}$ by elements on the right-hand side with extended $\mathcal{Q}$. This procedure is repeated until the extended weight coefficient set is the same as original one.

In other words, we start with $\mathcal{Q} = \{0\}$ meaning that we search all weight coefficients necessary for conversion for the case where there is no carry from the right. We add them to weight coeffcient set. These weight coefficients now may appear as a carry. If there are not suitable weight coefficients in weight coefficient set to satisfy formula (2.1) for all combinations of added coefficients and digits of input alphabet, we extend $\mathcal{Q}$ by appropriate ones. And so on until there is no need to add more elements.

The precise description of the semi-algorithm in a pseudocode is following:

Algorithm 1 Search for weight coefficient set (Phase 1)

1: $k := 0$
2: $Q_0 := \{0\}$
3: **repeat**
4: Extend $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$ in a minimal possible way so that

$$\mathcal{B} + \mathcal{Q}_k \subset \mathcal{A} + \beta\mathcal{Q}_{k+1}$$

5: $k := k + 1$
6: **until** $\mathcal{Q}_k = \mathcal{Q}_{k+1}$
7: $\mathcal{Q} := \mathcal{Q}_k$
8: **return** $\mathcal{Q}$

Algorithm 2 Extending intermediate weight coefficient set

Input: `candidates`, previous weight coefficient set $\mathcal{Q}_{k-1}$
1: $\mathcal{Q}_k := \mathcal{Q}_{k-1}$
2: **for all** `cand_for_x` in `candidates` **do**
3: **if** no element of `cand_for_x` in $\mathcal{Q}_{k-1}$ **then**
4: Add the smallest element (in absolute value) of `cand_for_x` to $\mathcal{Q}_k$
5: **end if**
6: **end for**
7: **return** $\mathcal{Q}_k$

2.2.2 Phase 2 – Weight function

Phase 2 – searching for a weight function

We want to find a length of the window M and a weight function $q : (\mathcal{A} + \mathcal{A})^M \to \mathcal{Q}$.

Suppose that the length of the window is m.

The idea is to check all possible right carries q_{j-1} and determine values q_j such that

$$z_j = w_j + q_{j-1} - q_j\beta \in \mathcal{A}\,.$$

Algorithm 3 Search for candidates

Input: previous weight coefficient set $\mathcal{Q}_{k-1}$

```
candidates:=[ ]
for all x ∈ B + Q_k do
    cand_for_x:=[ ]
    for all a ∈ A do
        if (x − a) is divisible by β in ℤ[ω] NEBO ZBETA??? then
            Append (x−a)/β to cand_for_x
        end if
    end for
    Append cand_for_x to candidates
end for
return candidates
```

The set of all such needed values of q_j is denoted by $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]} \subset \mathcal{Q}$

The length M and weight function q is found when

$$\#\mathcal{Q}_{[w_j,\dots,w_{j-M+1}]} = 1$$

for all $w_j, \dots, w_{j-M+1} \in (\mathcal{A}+\mathcal{A})^M$.

Phase 2 $m := 1$

For each $w_j \in \mathcal{A}+\mathcal{A}$ find minimal set $\mathcal{Q}_{[w_j]} \subset \mathcal{Q}$ such that

$$w_j + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_j]}$$

While $(\max\{\#\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]} : (w_j, \dots, w_{j-m+1}) \in (\mathcal{A}+\mathcal{A})^m\} > 1)$ do:

- $m := m + 1$
- For each $(w_j, \dots, w_{j-m+1}) \in (\mathcal{A}+\mathcal{A})^m$ find minimal set $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]} \subset \mathcal{Q}_{[w_j,\dots,w_{j-m+2}]}$ such that
$$w_j + \mathcal{Q}_{[w_{j-1},\dots,w_{j-m+1}]} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]}\,,$$

$M := m$

$q(w_j, \dots, w_{j-M+1}) :=$ only element of $\mathcal{Q}_{[w_j,\dots,w_{j-M+1}]}$

Now we have parallel conversion algorithm:

$$\begin{aligned} z_j &= w_j + q_{j-1} - q_j\beta = \\ &= w_j + q(w_{j-1}, w_{j-2}, \dots, w_{j-M}) - \beta q(w_j, w_{j-1}, \dots, w_{j-M+1}) = \\ &= z_j(w_j, w_{j-1}, \dots, w_{j-M})\,. \end{aligned}$$

Chapter 3

Implementation

Chapter 4

Examples

Summary