PUBLICATIONS DE L'INSTITUT MATHÉMATIQUE
Nouvelle série tome 44 (58), 1988, pp.137–142

# SYSTOLIC ARRAYS TO SOLVE ALGEBRAIC EQUATIONS BY BERNOULLI'S METHOD

**Octav Brudaru**

**Abstract.** The paper presents three linear systolic arrays to solve algebraic equations by Bernoulli's method. Two of them are able to execute the deflation after the finding of the real root.

**1. Introduction.** Many systolic algorithms have been proposed in order to solve problems in the field of numerical analysis (e.g. [1–4], [6–9], [12]).

This paper continues the work devoted to the design of systolic arrays to solve nonlinear equations. In [3] we give a systolic implementation of the Lin-Bairstow method and in [4] we propose a systolic network able to find zeros of a nonlinear real function defined by an arithmetic expression whose systolic computation is studied in [5].

In this paper we propose a systolic manner to find the real roots of polynomials by using the Bernoulli's method. The simple and regular form of this algorithm and the modest hardware requirements make it suitable for use in VLSI.

In Section 2 we outline the Bernoulli's method and give the basic systolic array. In Section 3 we present two completed versions of the basic array, both being able to execute the deflation of the processed polynomial inside the network.

**2. The basic systolic array.** Our notation is taken from [11, p. 319]. Let us consider the real coefficients polynomial $p(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n$. If $r$ is the single dominant root of $p$ then it may be the $\lim x_k/x_{k-1}$, for $k$ tending to infinity, where $(x_k)_{k\geq 1}$ is the solution of the difference equation of order $n$

(1) $$a_0x_k + a_1x_{k-1} + \cdots + a_nx_{k-n} = 0,$$

for the initial values $x_j = 0$, $j = 1, \ldots, n-1$ and $x_n = 1$. Without loss of generality we can assume that $a_0 = -1$ and thus (1) can be rewriten as

(2) $$x_k = a_1x_{k-1} + a_2x_{k-2} + \cdots + a_nx_{k-n},\ \ k > n.$$

---

*AMS Subject Classification* (1980): Primary 68D20.

As it is suggested in [10, pp. 226–227], we consider that $r_k = x_k/x_{k-1}$ approaches $r$ if

$$|r_k - r_{k-1}| \leq e \text{ and } k \leq K, \tag{3}$$

where $e$ and $K$ are given. Let us take $\bar{r} = r_k$ for smallest $k$ satisfying (3).

Further, the clock tick (CT) is the time to execute a division or both a multiplication and an addition. The time is denoted by $t$ and is the number of CTs. Also, we assume that each processor is active during each CT. The processors used to construct the basic systolic array (BSA) are depicted in Figure 1. They work so that (a) $z_{\text{out}}(t+1) = z_{\text{in}}(t) + a x_{\text{in}}(t)$, $x_{\text{out}}(t+1) = x_{\text{in}}(t)$: (b) $z(t+1) = x(t)/y(t)$; (c) $x_{\text{out}}(t+1) = x_{\text{in}}(t)$; (d) $rc(t+1) = (|x(t) - y(t)| \leq e)$.

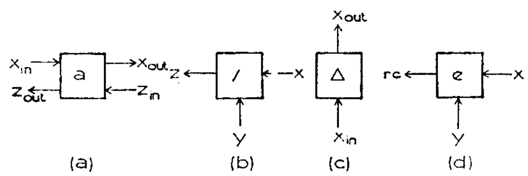


**Fig 1: The processors for the BSA.**

We note that each of the processors (a) and (b) has a register able to keep a value denoted by 'a' and 'e', respectively. The additional inputs to reset these processors are omitted.

An obvious solution to the parallel evaluation of (2) is to use the two-way pipeline algorithm presented by Kung in [6]. The resulted BSA is given in Fig. 2.

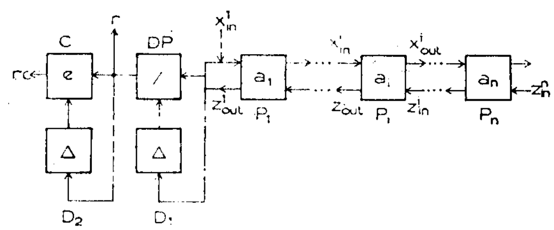


**Fig. 2: The basic systolic array for the Bernoulli's method.**

The register of the processor $P_i$ keeps the coefficient $a_i$ of $p$, $i = 1, \ldots, n$.

If $L$ is the label of an input or output of a processor then $L(t)$ designates the value circulating trought this during the $t$ pulse number.

Let us analyse the work of BSA. The initial values are sent by the host to BSA so that $X^1_{\text{in}}(t) = 0$, $t = 1, \ldots, n-1$ and $X^1_{\text{in}}(n) = 1$. Therefore, it is clear that $x_j$ enters $P_i$ during $j + i - 1$ CT, while $Z^n_{\text{in}}(t) = 0$, $t \geq 1$. During the $n$-th CT, all initial values are inside of BSA so that $X^i_{\text{in}}(n) = x_{n+1-i}$, $i = 1, \ldots, n$. The computation starts at this moment and gives $Z^1_{\text{out}}(n+1) = x_{n+1}$. From that moment the host does not emit so that only the values sent by $Z^1_{\text{out}}$ reach $X^1_{\text{in}}$, and the computation continues so that a new value $x_k$ emerges from $Z^1_{\text{out}}$ at every CT.

Consequently, $Z^1_{\text{out}}(k) = x_k$, $k = n + h$ and $h \geq 1$. Observe that because of $D_1$, $x_k$ and $x_{k-1}$ enter simultaneously DP, thus $r(k+1) = r_k$. Because of $D_2$, $r_k$ and $r_{k-1}$ enter $C$ at the same time and $rc(k+2)$ has the right value. This value is sent to the host as a return code indicating that $r(k+1)$ approaches the root $r$. We suppose that the host transmits at most $K$ true values on the control path as soon as $x_1$ entres $P_1$. As a consequence of the above analysis we can state.

THEOREM 1. *If* $X^1_{\text{in}}(t) = 0$, $t = 1, \ldots, n-1$, $X^1_{\text{in}}(n) = 1$, $Z^n_{\text{in}}(t) = 0$, $t \geq 1$ *then* $Z^1_{\text{out}}(t) = x_k$, $n < k \leq K$. *If there exists* $k_0$ *so that* $k_0 \leq K$ *and* $rc(k_0 + 2) = 1$ *then* $r(k_0 + 1) = \bar{r}$.

Let us remark that if $k_0$ is defined as in Theorem 1, then BSA needs only $k_0$ CTs to find $\bar{r}$ as compared with $O(n(k_0 - n - 1))$ amount of time required by the direct sequential algorithm (SA). Observe that BSA is more efficient if $k_0 < cn\,(k_0 - n - 1)$, i.e. $k_0 > n(n+1)c/(cn-1)$ for some positive connstant $c$ representing the time needed by SA to perform an inner-product step. We note that it is reasonable to suppose that $c \geq 1$ CT. Observe that $n(n+1)c/(cn-1) \leq n(n+1)/(n-1)$ for each $c \geq 1$, and therefore $k_0 > n(n+1)/(n-1)$ is a sufficient condition in order to have that BSA dominates SA.

**3. Variants executing the deflation.** In this section we present two variants of BSA able to execute the deflation of the processed polynomial as soon as a real root is found.

First we present a way to execute the deflation with additional processors. The processor used to do this is depicted in Figure 3(a). It works so that $w_{\text{out}}(t+1) = w_{\text{in}}(t)u_{\text{in}}(t) + d_{\text{in}}(t)$, $u_{\text{out}}(t+1) = u_{\text{in}}(t)$, $t \geq 1$, and is a variant of the processor in Figure 1(a). Moreover, each $P_i$ of BSA is modified in order to change the content of its register keeping the associated coefficient, by ading an input (output) $A^i_{\text{in}}(A^i_{\text{out}})$ as it is shown in Figure 3(b).

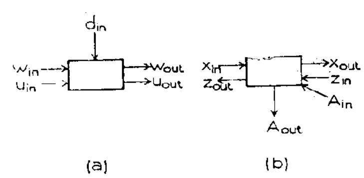


**Fig. 3: Two kinds of inner-product step processor.**

The coefficients of the quotient are computed by using a linear systolic array implementing the Horner's rule presented by Pam and Mostow in [9]. The obtained systolic array, called $\text{BSAD}_1$, is shown in Figure 4. The true value $rc(k_0+2)$ can be used to control the work of $Q_i$ and the updating of the register of $P_i$, $i = 1, \ldots, n-1$. Then delay processor $D_3$ ensures that the value of $r(k_0+1)$ and the control signal arrive simultaneously at $Q_1$. The work of $\text{BSAD}_1$ during the deflation stage is described below. Let $p(x) = q(x)(x - \bar{r})$ where $q(x) = -x^{n-1} + b_1 x^{n-2} + \cdots + b_{n-2}x + b_{n-1}$. After the deflation, a new iteration starts with $q$ instead of $p$, thus we need $b_1, \ldots, b_{n-1}$. Let us take $t_0 = k_0 + 2$. If $U^1_{\text{in}}(t_0) = \bar{r}$, $W^1_{\text{in}}(t_0) = -1$ and $A^1_{\text{out}}(t_0) = a_1$ then $Q_1$ computes $b_1 = (-1)\bar{r} + a_1$, sets $W^1_{\text{out}}(t_0+1) = A^1_{\text{in}}(t_0+1) = b_1$

and $U^1_{\text{out}}(t_0+1) = \bar{r}$. Thus during the $t_0+1$ CT $a_1$ is replaced by $b_1$. Clearly, $U^i_{\text{in}}(t_0+i-1) = \bar{r}$, $i = 2, \ldots, n-1$, and if $A^i_{\text{out}}(t_0+i-1) = a_i$ then $W^i_{\text{out}}(t_0+i) = b_i$, $i = 2, \ldots, n-1$. We conclude that the deflation is executed by $\text{BSAD}_1$ in $n-1$ CTs and the coefficients of $q$ are already loaded in BSA after this time. A new iteration can begin during the $t_0+n-1$ CT or $t_0+1$ CT if we assume that a new iteration starts as soon as the register of $P_1$ is updated. In this last case, the deflation and the first iteration adressed to the deflated polynomial are parallel.

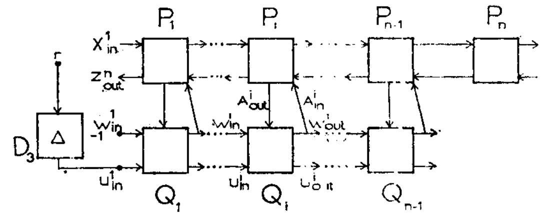


**Fig. 4. The first variant of BSA excuting the deflation.**

Consequently, we can formulate

THEOREM 2. *The systolic array* $\text{BSAD}_1$ *correctly executes the iteration* (2) *for* $p(x)$ *(with the same performances as* BSA*) and the deflation (in* $n-1$ CTs*). The first iteration involving the deflated polynomial* $q$ *may begin with* 3 CTs *after the moment at which a root of* $p(x)$ *is found.*

Observe that the application of Bernoulli's method can be extended from a polynomial to its quotient as long as each new set of iterations terminates with the finding of a real root, and the process is executed entirely inside of $\text{BSAD}_1$. Thus $\text{BSAD}_1$ eliminates the additional time wich could be needed by the host in order to compute and broadcast the coefficients of $q(x)$.

In order to eliminate the additional processors $Q_1, \ldots, Q_m$ required by $\text{BSAD}_1$, we propose a new variant. In this variant, called $\text{BSAD}_2$, $Q_1, \ldots, Q_m$ are eliminated with the price of a minor increasing of the complexity of each $P_i$. The processor $P'$ used to construct $\text{BSAD}_2$ is depicted in Figure 5. It works so that $x_{\text{out}}(t+1) = x_{\text{in}}(t)$, $c_{\text{out}}(t+1) = c_{\text{in}}(t)$, and if $c_{\text{in}}(t) = 0$ then $z_{\text{out}}(t+1) = z_{\text{in}}(t) + x_{\text{in}}(t)$ a, otherwise $(c_{\text{in}}(t) = 1) w_{\text{out}}(t+1) = a + x_{\text{in}}(t) w_{\text{in}}(t)$ and $R := w_{\text{out}}(t+1)$, $t \geq 1$. The processor $A(M)$ performs an addition (multiplication) while $R$ is the register keeping $a$. The processor denoted by $S$ is a two-input multiplexor transmiting the input whose label is equal to $c_{\text{in}}(t)$. We suppose that the sum of delays introduced by $S$, $M$ and $A$ does not exceeed one CT. The array $\text{BSAD}_2$ is presented in Figure 6. The subarray containing the processors $C$, $DP$, $D_1$ and $D_2$ in Figure 2 and $D_2$ in Figure 4 is conected to $P'_1$ but it is not drawn in Figure 6. We note that the computation of (2) moves from $P'_i$ to $P'_{i-1}$, $i = n, n-1, \ldots, 2$, while just one true value sent by $rc$ and entering $C^1_{\text{in}}$ suffices to move the front of the deflation computation from $P'_i$ to $P'_{i+1}$, $i = 1, \ldots, n-1$. Observe that $P'_n$ may be replaced by $P_n$ because the deflation stops at $P'_{n-1}$.

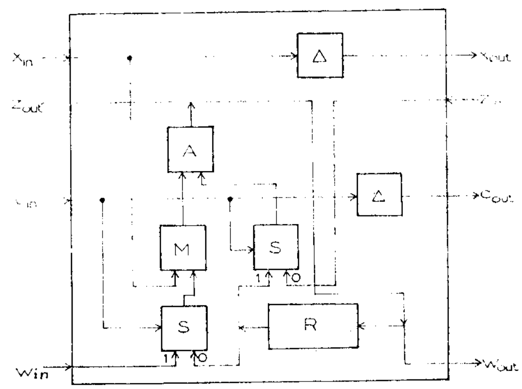


Fig. 5. The processor used to conastrut $BSAD_2$

It is clear that $BSAD_2$ has all the performances of BSA and $BSAD_1$ concerning the response time and the period, but $BSAD_2$ dominates $BSAD_1$ because it requires only $n$ inner-product step processors. A statement similar to Theorem 2 can be formulated in connection with $BSAD_2$.

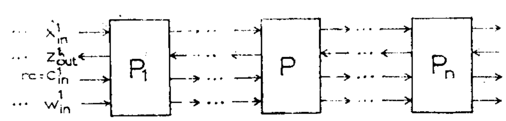


Fig. 6. The second variant of BSA executing the deflation

Also, we note that $BSAD_2$ can be improve by using a single way instead of $w$-way and $z$-way, because these ways are used in different stages of the computation.

## REFERENCES


[1] F. André, P. Frison, P. Quinton, *Algorithmes systoliques: de la théorie à la pratique*, L'Onde Eléctrique, **64** no. 615 (1984), 5–16.

[2] O. Brudaru, *Systolic algorithms to solve linear systems by iteration methods*, An. Stiint. Univ. "Al. I. Cuza" Iasi, Sect. Ia Mat. **1** (1985), 301–306.

[3] O. Brudaru, *Systolic arrays for polynomial quadratic factors computation*, Proc. CompEuro '87-VLSI and Computers, (Hamburg, May 1987), to appear.

[4] O. Brudaru, *Systolic arrays for solving nonlinear equations by a noniterative method*, An. Stiint. Univ. "Al. I. Cuza" Iasi, to appear.

[5] O. Brudaru, *Systematic synthesis of systolic arrays for some real functions computation*, in preparation.

[6] H.T. Kung, *Let's design algorithms for VLSI systems*, Tehnical Report CMU-CS-79-151, Carnegie-Mellon Univ., Pittsburgh, 1979.

[7] H.T. Kung, *Why systolic architectures?*, Computer Magazine **15** (1982), 37–46.

[8] C.E. Lieserson, *Area-Efficient VLSI Computation*, MIT Press, Massachusetts, 1983.

[9] M.S. Pam, J. Mastow, *A transformational model of VLSI systolic design*, Proc. IFIP 6th Symp. Comput. Harwarde Descriptive Lang. Appl., Carnegie-Mellon Univ., Pittsburgh, 1983.

[10] J.R. Rice, *Numerical Methods, Software and Analysis*, McGraw-Hill, New-York, 1983.

[11] F. Scheid, *Theory and Problems of Numerical Analysis*, Schaum's Qutline Series, McGraw-Hill, New York, 1968.

[12] J.D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, 1984.

Polytechnical Institute of Iasi
Computer Centre
Splai Bahlui 63
66000-Iasi
Romania

(Received 23 04 1987)