UNIVERSITAS COMENIANA
ACTA MATHEMATICA UNIVERSITATIS COMENIANAE
XLVIII—XLIX — 1986

THE POLYNOMIAL GCD-ALGORITHM IMPLEMENTED IN A SYSTOLIC ARCHITECTURE

JURAJ PROCHÁZKA, Bratislava

Introduction

In the last decade we have witnessed a great progress in microelectronics (VLSI) that caused drastic changes in computer construction. VLSI technology allows the construction of circuits with a large density of logical elements. The emergence of systolic systems is one consequence of this progress.

The systolic system is a regular net of processors that are processing and shifting data through the system in a rhythmic fashion. Each processor works like a heart pumping the blood in the circulating system. The algorithm implemented in a systolic system is called systolic algorithm (or VLSI-algorithm).

Properties of a good systolic algorithm are the following:

(i) There are only few types of simple processors;

(ii) Data and control flows are simple and regular;

(iii) The algorithm uses extensive concurrency by pipelining the processors involved in a computation;

(iv) The algorithm makes multiple use of each input data item so that each input travels through an array of processors.

In designing systolic algorithm we should follow the above criteria.

Systolic systems were suggested and studied for various problem areas (e.g. searching and sorting, recognizing in formal languages, linear algebra, binary and polynomial arithmetics, filtering and geometrical problems).

In this paper we shall study a polynomial GCD computation.

Let us have two polynomials $u(x)$, $v(x)$ with $m = \deg(u(x))$, $n = \deg(v(x))$ and $m \geqslant n \geqslant 0$. The systolic system for a computation of GCD for $u(x)$ and $v(x)$ suggested in this paper uses $(m + 1)$ processors (space complexity). If a *time unit* is a time needed for realization of one pulse of the systolic system, then this system gives the first coefficient of the result in $(2m + 2n + 3)$ time units (time complexity) after the coefficients u_m, v_n enter the first processor of the systolic system, see *Fig. 1.*

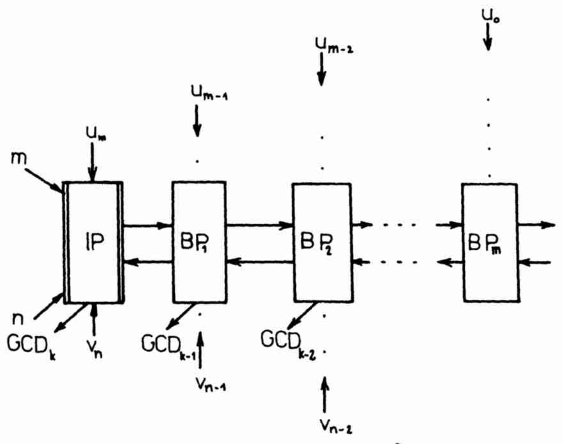

Fig. 1. Systolic system SC for Collins polynomial GCD algorithm

The systolic system described in this paper is based on the sequential Collins algorithm, Knuth 1969 [8], so we call it the systolic system SC. The Collins algorithm is an ingenious modification of the Euclidean algorithm for polynomials over a unique factorization domain (UFD) using a pseudodivision. The systolic system SC uses two kinds of processors:

IP — the initial processor that initializes parallel process of computing.

BP — the basic processor that performs the basic computing.

This result has potential application in the areas of symbolic and algebraic computing, error-correcting codes, signal processing and image processing.

Collins algorithm

The Collins algorithm in the PASCAL-like form follows. We shall use the following symbols:

cont($u(x)$) — is GCD of coefficients of the polynomial $u(x)$;

pp($u(x)$) — is a primitive part of $u(x)$, i.e. $u(x)$/cont ($u(x)$);

lc($u(x)$) — is a leading coefficient of the polynomial $u(x)$.

Algorithm CGCD

Input: The non-zero polynomials $u(x)$, $v(x)$ over UFD S with $\deg(u(x)) \geqslant \deg(v(x)) \geqslant 0$;

Output: The greatest common divisor of the polynomials $u(x)$, $v(x)$;

```
    Function CGCD($u(x)$, $v(x)$);
1.  begin $d$: = GCD(cont($u(x)$), cont($v(x)$));
2.        $u(x)$: = pp($u(x)$); $v(x)$: = pp($v(x)$);
3.        $a$: = 1;
4.        repeat REM($u(x)$, $v(x)$, $a$)
          until deg($v(x)$) $\leqslant 0$;
5.        CGCD: = if $v(x) \not\equiv 0$ then $d$ else $d *$ pp($u(x)$)
    end.
```

```
    procedure REM($u(x)$, $v(x)$, $a$);
1.  begin $\bar{a}$: = $\mathrm{lc}(v(x))^{\deg(u(x)) - \deg(v(x)) + 1}$;
2.        repeat
          $u(x)$: = $\mathrm{lc}(v(x)) * u(x) - \mathrm{lc}(u(x)) * v(x) * x^{\deg(u(x)) - \deg(v(x))}$
          until deg($u(x)$) < deg($v(x)$);
3.        $c(x)$: = $u(x)$; $u(x)$: = $v(x)$; $v(x)$: = $c(x)/a$;
4.        $a$: = $\bar{a}$
    end.
```

We can use the above algorithm only if we suppose that there exist auxiliary algorithms:

— for computing GCD for elements from S;

— for dividing a by b for $a, b \in S$, in the case $b \neq 0$ and there exists $c \in S$ s.t. $a = c * b$.

Generally, S can be an integral domain of the polynomials.

The algorithm CGCD uses procedure REM in order to compute remainder of two polynomials in a pseudodivision.

In a systolic implementation we do not need to perform all instructions of the Collins algorithm in a systolic system. In the sake of the SC-system simplicity we can leave the instructions 1. and 2. from the function CGCD and also the multiplication $d *$ pp($u(x)$) outside of the systolic system. In case the polynomials $u(x)$, $v(x)$ are over integers, these instructions are based on the integer GCD computation that was already effectively solved in a systolic system [3, 7].

The following CGCD1-algorithm is the Collins algorithm CGCD after the above simplification.

```
    Function CGCD1($u(x)$, $v(x)$);
1.  begin $a$: = 1;
2.        repeat REM($u(x)$, $v(x)$, $a$)
          until deg($v(x)$) $\leqslant 0$;
3.        CGCD1: = if $v(x) \not\equiv 0$ then 1 else $u(x)$
    end.
```

The input polynomials $u(x)$, $v(x)$ for the CGCD1 algorithm will be primitive.

Systolic system SC

The systolic system SC is based on the linear two directional array of processors in *Fig. 1*. The input of the polynomial coefficients into the SC-system is oblique, such that

the i-th coefficients of $u(x)$, $v(x)$, i.e. u_{m-i+1}, v_{n-i+1} enter the i-th processor of the SC-system in the i-th pulse of computation.

The Collins algorithm as all well-knowns algorithms for solving the polynomial GCD problem is based on the general technique of reducing the degree of the two given polynomials by "GCD-preserving" transformations. In the Collins algorithm we denote this transformation R.

$$R\colon\ (u(x),\ v(x)) \leftarrow (\bar{u}(x),\ v(x)),\ \text{where}$$

$$\bar{u}(x) = \mathrm{lc}(v(x)) * u(x) - \mathrm{lc}(u(x)) * v(x) * x^{\deg(u(x)) - \deg(v(x))}. \tag{1}$$

The sequence of transformations R denoted $(T_1, \ldots, T_p)$ is implemented in the SC-system. The transformation T_1 is computed in the first $(m + 1)$ processors such that the output in the $(i + 1)$-st pulse from the i-th BP-processor is the coefficient $\bar{u}_{m-i}$ of the polynomial $\bar{u}(x)$, see (1), and in the next pulse $\bar{u}_{m-i}$ is an input to the $(i - 1)$-st BP-processor. Each second pulse, that is $(2i - 1)$-st pulse, the new transformation T_i starts in the IP-processor, so that it sends the leading coefficients of the current $u(x)$, $v(x)$ to the right in order to pass through all processors of the SC-system. The SC-system activity of starting new R-transformations continues until $\deg(u(x)) < \deg(v(x))$, and then the SC-system exchanges the roles of the polynomials $u(x)$, $v(x)$ by the transformation:

$$(u(x),\ v(x)) \leftarrow (v(x),\ \bar{u}(x)/a). \tag{2}$$

The other activity provided by the SC-system is shifting the leading coefficient $\mathrm{lc}(\bar{u}(x))$ to the left to the IP-processor, when $\deg(\bar{u}(x)) < \deg(u(x)) - 1$. We can divide the work of the SC-system into three basic activities provided in a parallel manner:

1. sequence of "GCD-preserving" transformations R;
2. exchange transformation

$$(u(x),\ v(x)) \leftarrow (v(x),\ \bar{u}(x)/a),$$

when $\deg(\bar{u}(x))$, $\deg(v(x))$;

3. shifting the first nonzero coefficient of $\bar{u}(x)$ to the IP-processor.

In a systolic system working this way each processor is active each second pulse. The time of the processor nonactivity, we say *the dual time*, could be used for a dual computation.

In the SC-system, as mentioned above, we shall use two types of processors drawn in *Fig. 2.*

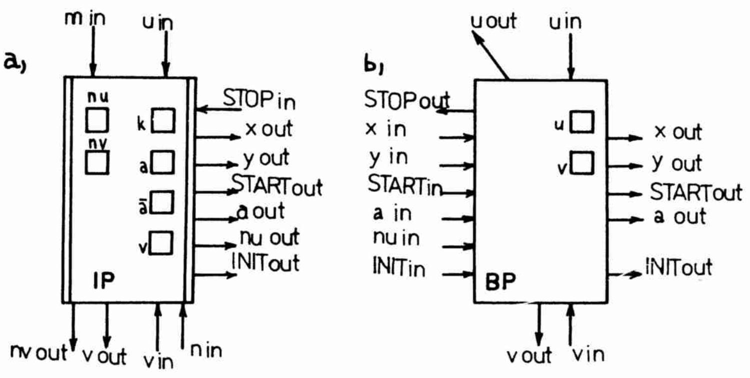

Fig. 2. Communication lines of the SC-system processors:
a) Initial processor;
b) Basic processor.

The sense of the communication lines of the SC-system processors is described in the next processor programs.

```
Program basicprocessor;
begin if INITin = 1 then v: = vin; INITout: = INITin;
    case STARTin of
        0: begin {NORMAL}
                uout: = yin * uin − xin * v; xout: = xin; yout: = yin;
                STARTout: = 0; STOPout: = 0
           end;
        1: begin {LATENT}
                u: = yin * uin − xin * v; xout: = xin; yout: = yin;
                STARTout: = 0;
                if nuin < 0 then STOPout: = 1
                   else if u = 0 then STOPout: = 2
                   else (uout: = u div ain; STOPout: = 3)
           end;
        2: begin {STOP} vout: = v; STARTout: = 2 end;
        3: begin {EXCHANGE}
                STARTout: = 3; aout: = ain;
```

```
                    $u := uin$ **div** $ain$; $uout := yin * v - xin * u$; $v := uin$
                    $xout := xin$; $yout := yin$; $STOPout := 0$
                **end**
        **end**
**end**.

*Program initialprocessor*;
**begin** $INITout := 0$;
        **case** $STOPin$ **of**
            0: **begin** {$NORMAL$}
                    $xout := uin$; $yout := v$;
                    **if** $k = 0$ **then** ($STARTout := 1$; $nu := nu - 1$;
                                    $aout := a$)
                      **else** $STARTout := 0$;
                    $k := k - 1$
                **end**;
            1: **begin** {$STOP$}
                    **if** $nv = 0$ **then** $vout := 1$ **else** $vout := v$;
                    $nvout := nv$; $STARTout := 2$
                **end**;
            2: **begin** {$LATENTPRESERVER$}
                    $xout := 0$; $yout := 1$; $aout := a$; $STARTout := 1$;
                    $nu := nu - 1$
                **end**;
            3: **begin** {$RESTART$}
                    $xout := v$; $yout := uin$; $v := uin$; $STARTout := 3$;
                    $k := nv - nu$; $nv := nu$; $a := \bar{a}$; $\bar{a} := uin \uparrow (k + 1)$;
                    $k := k - 1$; $a\ out := a$
                **end**;
            4: **begin** {$START$}
                    $xout := uin$; $yout := vin$; $v := vin$;
                    $k := min - nin$; $nv := nin$; $nu := nin$; $a := 1$;
                    **if** $k = 0$ **then** ($STARTout := 1$; $nu := nu - 1$;
                                    $aout := a$)
                      **else** $STARTout := 0$;
                    $\bar{a} := v \uparrow (k + 1)$; $k := k - 1$; $INITout := 1$
                **end**;
        **end**; $nuout := nu$
**end**.
```

The above mentioned programs for the IP-processor and for the

BP-processor together with *Fig. 1* and *Fig. 2* form one integral unit called the systolic system SC that implements the Collins algorithm.

Formal correctness proof is beyond the scope of this paper. Nevertheless, this systolic algorithm has been tested by simulation using PASCAL program on a serial computer.

Conclusions

There exists already the polynomial GCD systolic algorithm suggested by Brent and Kung [2, 3]; we denote it B-K system. The B-K system is based on the linear one-directional array of processors with different speeds of data flows. The SC-system was developed independently. Before comparing the B-K system and the SC-system we should transform these two systems to the same bases. The B-K system works for polynomials over finite field and the SC-system over UFD (integers). The SC-system can be easily transformed in order to work over a finite field and the processor programs would be simpler. On the contrary, the B-K system could be extended for UFD. The SC-system working over a finite field can be easily transformed to the systolic system solving an extended GCD-problem: that is, for $u(x)$, $v(x)$, to find polynomials $s(x)$, $t(x)$ such that

$$u(x) * s(x) + v(x) * t(x) = \text{GCD}.$$

An extended GCD problem for the Collins algorithm has been solved serially in [12], but not yet in a systolic system. Up to now it has been an open problem.

For the computation of GCD ($u(x)$, $v(x)$) the SC-system needs

Space: $(m + 1)$-processors;

Time: in the worst case $(2m + 2n + 3)$-time units, in the best case $(2m + 3)$-time units until the output of the first GCD-coefficient.

For the same computation the B-K system needs

Space: $(m + n + 1)$-processors:

Time: $(2m + 2n + 2)$-time units in all cases.

Let us compare the SC-system with the B-K system on the same basis. In the SC-system there is a slight improvement (about 2 times better) in the space complexity. But we shall pay for it by increasing (about 2 times) the period of computation (systolic complexity), because we must wait with a next input until the output of the first GCD-coefficient. (We should use also the dual computation: Slightly changing the programs of the processors we can transform them to the combinatorial elements without memory. In this case we can use the time of the processor nonactivity for the dual computation.)

The SC-system gives us a more effective GCD-computation of the set of k polynomials, with $k > 2$. We need it in symbolic and algebraic computations, for example in solving pp($u(x, y)$). In this case the B-K system loses the advantage of the period of computation, because it must wait until an intermediate GCD-result is available. But the SC-system gives us the GCD-result in the same place as we need it. The first GCD-coefficient emerges from the first processor on the contrary to the B-K system, where each GCD-coefficient emerges from the last processor.

A new technology of the programmable systolic chip (PSC) [4] allows even practical implementations of the above-described algorithm in systolic arrays. PSC is one solution for the following problem: For their regularity and simplicity, the cost of the design and implementation of systolic systems is less than the cost of the design and implementation of a general-purpose computer. This advantage is usually spoiled because a special systolic system can be used only for a narrow class of problems. PSC was developed in the Carnegie-Mellon University.

REFERENCES

[1] Aho, A. V.—Hopcroft, J. E.—Ullman, J. D.: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Mass. 1974.

[2] Brent, R. P.—Kung, H. T.: Systolic VLSI Arrays for Polynomial GCD Computation. Tech. Report CMU-CS-82-118, Dept. of Comp. Science, Carnegie-Mellon University, March 1982.

[3] Brent, R. P.—Kung, H. T.—Luk, F. T.: Some Linear-Time Algorithms for Systolic Arrays. Information Processing 83, pp. 865—876, North-Holland, 1983.

[4] Fisher, A. L.—Kung, H. T.—Monier, L. M.—Dohi, Y.: Architecture of the PSC: Programable Systolic Chip. Proc. of 10-th Int. Symp. on Computer Architecture 1983, pp. 48—53.

[5] Gruska, J.: Personal Communication, 1984.

[6] Gruska, J.—Ružička, P.—Wiedermann, J.: Systolické systémy. Proc. of Software seminar SOFSEM '83, Ždiar 1983, pp. 267—307.

[7] Kannan, R.—Miller, G.—Rudolph, L.: Sublinear Parallel Algorithm for Computing the Greatest Common Divisor of Two Integers. Proc. of 25th Annual Symposium on Foundation of Computer Science, Oct. 1984, pp. 7—11.

[8] Knuth, D. E.: The Art of Computer Programing, Vol. 2: Seminumerical Algorithms. Addison-Wesley, Reading, Mass. 1969.

[9] Kung, H. T.: Let's Design Algorithms for VLSI Systems. Tech. Report, Dept. of Computer Science, Carnegie-Mellon University, Sept. 1979.

[10] Kung, H. T.: Use of VLSI in Algebraic Computation: Some Suggestions. Proc. of the ACM Symp. SYMSAC '81, 1981, pp. 218—222.

[11] Kung, H. T.: Why Systolic Architectures? Computer Magazine, 15 (1), Jan. 1982, pp. 37—46.

[12] Sasaki, T.: Extended Euclidean Algorithm and Determinants, Inform. Process. Society of Japan, WG SYM-Meeting, May 1982.

Author's address:
Juraj Procházka,
Katedra teoretickej kybernetiky, MFF UK,
Matematický pavilón,
Mlynská dolina,
842 15 Bratislava

Received: 13. 2. 1984

SÚHRN

ALGORITMUS PRE NSD DVOCH POLYNÓMOV IMPLEMENTOVANÝ V SYSTOLICKEJ ARCHITEKTÚRE

Juraj Procházka, Bratislava

Majme dva polynómy $u(x)$, $v(x)$, $\mathrm{st}(u) \geqq \mathrm{st}(v) \geqq 0$, nad oborom integrity s jednoznačným rozkladom. V práci je navrhnutý a opísaný nový systolický algoritmus, ktorý využíva lineárne dvojsmerné pole s počtom procesorov $(\mathrm{st}(u) + 1)$. Dĺžka výpočtu je $(2 \cdot \mathrm{st}(u) + 2 \cdot \mathrm{st}(v) + 3)$ časových jednotiek (pulzov) v najhoršom prípade.

РЕЗЮМЕ

АЛГОРИТМ ДЛЯ Н.О.Д. ДВУХ МНОГОЧЛЕНОВ, ИМПЛЕМЕНТИРОВАННЫЙ В СИСТОЛИЧЕСКОЙ АРХИТЕКТУРЕ

Юрай Прохазка, Братислава

Пусть $u(x)$, $v(x)$ многочлены над областью с однозначным разложением на множители и $\mathrm{st}(u) \geqq \mathrm{st}(v) \geqq 0$. В статье показан и описан систолический алгоритм находящий наибольший общий делитель для $u(x)$ и $v(x)$. Алгоритм использует линейное поле параллельно-работающих процессоров длиной в $(\mathrm{st}(u) + 1)$. Время вычисления не больше чем $(2 \cdot \mathrm{st}(u) + 2 \cdot \mathrm{st}(v) + 3)$-пульсов систолического поля.