An Implementation of MacMahon’s Partition Analysis in Ordering the Number of Lattice Points in Convex Polyhedron with Examples for Systolic Arrays

Halil Snopce1, Ilir Spahiu2

1 South East European University CST Faculty, 1200 Ilindenska bb, Tetovo, Macedonia
2 Faculty of Pedagogy “ss Clement of Ohrid”, “Ss. Cyril and Methodius” University in Skopje
1h.snopce@seeu.edu.mk  2i.spahiu@seeu.edu.mk

Abstract. We have investigated the “Omega calculus”, as a computational method for solving problems via their corresponding Diophantine relation. These methods can be applied for the problems related with the number of lattice points in polyhedron, positively in our case for systolic array computations. From the corresponding systolic algorithm we form a system of linear diophantine equalities using the domain of computation which is given by the set of lattice points inside the polyhedron. Then we run the Mathematica program DiophantineGF.m. This program calculates the generating function from which is possible to find the number of solutions to the system of Diophantine equalities, which in fact gives the lower bound for the number of processors needed for the corresponding algorithm. There is given a mathematical explanation of the problem as well. We illustrate this for finding the lower bound of the systolic algorithm for Discrete Fourier transformation (DFT).

Keywords: Systolic array, nested loop algorithm, systolic algorithm, Polyhedron, generating function, system of Diophantine equations, index space, lattice points, lower bound of processor elements, Ω calculus.

1. Introduction

Systolic arrays are important for application of specific systems. These arrays are with great potential of pipelining, synchronized, regularly connected multiprocessing elements with locally communicative interconnection structure. So, they are suitable for many existing algorithms in matrix computations, image and signal processing, pattern recognition etc. They are ideally suited to VLSI technology. In this direction there are a lot of studies concerning the minimizing the computation time, the number of processing elements and optimizing of systolic arrays [1-11]. Processor-time-minimal schedules have been proposed for different problems as well [12-16]. Transformation of the problem from geometrical into combinatorial analysis can be seen at [10, 17]. Mathematical guide for the analysis can be seen at [17-25]. A general and uniform technique for deriving lower bounds of processing elements (as a piecewise polynomial function) is presented at [16]. At the same article is shown that the nodes of the dag can be viewed as lattice points in convex polyhedron. Adding to this the linear constraint of the schedule there will be form a system of Diophantine equations where the number of solutions is a lower bound. In this article, using the methodology mentioned above, we have obtained the optimal lower bound for the number of processors required by the systolic algorithm for DFT.

2. Some Definitions and Properties

Definition 1: Let $a \in R^n$ and $b \in R$. Then a hyperplane consists of the set $\{x \in R^n \ | \ a^T x = b\}$ and a halfspace consists of the set $\{x \in R^n \ | \ a^T x \geq b\}$. If $A$ is an $mxn$ matrix and $b \in R^m$ then a polyhedron $P$ consists of the set $P = \{x \in R^n \ | \ Ax \leq b\}$. This means that $Ax \leq b$ is matrix inequality (sistem of linear inequalities) where $m$ is the number of half-spaces. In other words, a polyhedron is the intersection of finitely many halfspaces.

Definition 2: For a multiple Laurent series, $\sum_{V_1,...,V_k} A_{V_1,...,V_k} \lambda_1^{V_1}...\lambda_k^{V_k}$, the operator $\Omega_z$ is defined by: $\Omega_z \sum_{V_1,...,V_k} A_{V_1,...,V_k} \lambda_1^{V_1}...\lambda_k^{V_k}$. This
means that the operator $\Omega_z$ is defined on functions with absolutely convergent multisum expansions. In an open neighborhood of the complex circles $|\lambda_i| = 1$, the action of $\Omega_z$ is given by

$$\Omega_z \sum_{v_1,\ldots,v_k = 0}^{\infty} A_{v_1,\ldots,v_k} \lambda_1^{v_1} \cdots \lambda_k^{v_k} = \sum_{v_1,\ldots,v_k = 0}^{\infty} A_{v_1,\ldots,v_k}$$

We give two of the many identities presented in [22].

**Lema 1:** For any integer $s \geq 0$,

$$\Omega_z \frac{1}{(1 - \lambda)(1 - \frac{y}{\lambda})} = \frac{1}{(1 - x)(1 - x'y)}$$

**Lema 2:**

$$\Omega_z \frac{1}{(1 - \lambda^2)(1 - \frac{y}{\lambda})} = \frac{1 + xy}{(1 - x)(1 - xy^2)}$$

MacMahon leaves the verification of many of his identities to the reader. Here we will give the prove for first lema.

From geometric series expansion we have:

$$\frac{1}{(1 - \lambda)(1 - \frac{y}{\lambda})} = \sum_{a \geq 0} (\lambda^a y^a) = \sum_{a \geq 0} \lambda^{a} x^a y^a$$

If $a_2 s > a_1$, then $\lambda$ will have a negative power. To prevent this from happening, let $a_1 - a_2 s = b$, force the restriction $b \geq 0$ and making appropriate substitution into the crude generating function we will have:

$$\sum_{a \geq 0} \lambda^{a_1 - a_2 s} x^{a_1} y^{a_2} = \sum_{a_1, b \geq 0} \lambda^{b} x^{a_1 + b} y^{a_2} = \sum_{a_1, b \geq 0} (\lambda x)^b (x' y)^{a_2} = \frac{1}{(1 - \lambda x)(1 - x'y)}$$

Now if we set $\lambda = 1$, we have the desired identity. (Using the mentioned conditions above, we actually have used the defined $\Omega_z$ operator).

3. **Mathematical Explanation of the Algorithm**

The general idea is that if there is a polyhedron $P = \{ x \in R^n \setminus A x \geq b \}$, then for each defining halfspace $a_i x - t b_i \geq 0$ we embed $\lambda^{(a_i x - t b_i)}$ into a crude generating function. Because of considering only the positive constraints $t \geq 0$, we embed $y^t$ in the crude generating function as well to obtain:

$$F(\lambda, y) = \sum_{x_i \geq 0} \lambda^{(a_i x - t b_i)} y^{t}$$

For example let $t \geq 0$ and $P = \{(x_1, x_2) \in R^2 / x_2 + 2x_1 \geq 2, 2 \geq x_2, 1 \geq x_1 \}$ be a given polyhedron. Then the number of integer points contained in $t \cdot P$ is equivalent to the number of integer solutions of the system:

$$\begin{align*}
x_2 + 2x_1 - 2t & \geq 0 \\
2t - x_2 & \geq 0 \\
t - x_1 & \geq 0 \\
t & \geq 0
\end{align*}$$

(1)

In fact this is the polyhedron with $P = \{ x \in R^2 / A x \leq b \}$ where

$$A = \begin{bmatrix} -2 & -1 \\ 0 & 1 \end{bmatrix}$$

and $b = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$

The corresponding crude generating function to this system will be:

$$\sum_{x_i \geq 0} \lambda^{x_1 + 2x_2 - 2t} \lambda_2^{2x_2} \lambda_3^{-x_1} y^t$$

Making corresponding transformations we have:

$$\sum_{x_i, t \geq 0} \lambda^{x_1 + 2x_2 - 2t} \lambda_2^{2x_2} \lambda_3^{-x_1} y^t = \sum_{x_i, t \geq 0} \left( \frac{\lambda_1}{\lambda_3} \right)^{x_1} \left( \frac{\lambda_1}{\lambda_2} \right) \lambda_2^{2x_2} \lambda_3^{-x_1} y^t$$

660
This corresponds to the following crude rational generating function:

\[
\frac{1}{\left(1 - \frac{x_1^2}{x_3}\right) \left(1 - \frac{x_2^2}{x_2}\right) \left(1 - \frac{x_1^2 x_2}{x_3}\right) y} = \frac{1}{\left(1 - \frac{x_2}{x_2}\right) \left(1 - \frac{x_1^2}{x_3}\right) y \left(1 - \frac{x_1}{x_3}\right) (1 - y)}
\]

To find the corresponding rational generating function, we will use the two lemmas given above (\(\Omega (\lambda)\) means that the given identity is used for parameter \(\lambda\)).

\[
\Omega_2(\lambda) = \frac{1}{\left(1 - \frac{x_2}{x_2}\right) \left(1 - \frac{x_1^2}{x_3}\right) y \left(1 - \frac{x_1}{x_3}\right) (1 - y)} = \frac{1}{(1 - y)(1 - y)(1 - y)}
\]

So, we can conclude that:

\[
\Omega_2 = \frac{1}{\left(1 - \frac{x_2}{x_2}\right) \left(1 - \frac{x_1^2}{x_3}\right) \left(1 - \frac{x_1^2 x_2}{x_3}\right) y} = \frac{1+y}{(1-y)^3}
\]

This is the same rational generating function which can be obtained via Mathematica program \(\text{DiophantineGF}.m\). In order to transform the set of inequalities (1) to a set of Diophantine equations, we introduce integral slack variables \(s_1, s_2, s_3 \geq 0\) and write:

\[
\begin{align*}
2x_1 + x_2 - s_1 &= 2t \\
-x_2 - s_2 &= -2t \\
-x_1 - s_3 &= -t \\
-s_4 &= -t
\end{align*}
\]

(2)

Because the program \(\text{DiophantineGF}.m\) essentially requires three arguments \((A, b, c)\) of the Diophantine system \(Ax = bt + c\), the main computation is performed by the call \(\text{DiophantineGF}[A, b, c]\). The result is the rational generating function. From (2) we have

\[
A = \begin{bmatrix} 2 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ -2 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad c = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\]

The original result from the program \(\text{DiophantineGF}.m\) is given below:

\[
\begin{align*}
\text{In}[1] &= \text{DiophantineGF}.m \\
\text{In}[2] &= a = \{(2,1,-1,0,0,0), \{0,-1,0,-1,0,0\}, \{-1,0,0,0,-1,0\}, \{0,0,0,0,-1,0\}\}; \\
\text{In}[3] &= b = \{2,2,-1,1\}; c = \{0,0,0,0\}; \\
\text{In}[4] &= \text{DiophantineGF}[a,b,c]
\end{align*}
\]

\[
\text{Out}[1] = -1 + t
\]

If we take \(t = y\) the result will be the same like in mathematical explanation

\[
-\frac{1+t}{1+t} = \frac{1+t}{1-t} = \frac{1+y}{1-y}
\]

4. Implementation on Systolic Arrays

Let suppose that \(A = (a_{i,j})\) and \(B = (b_{i,j})\) be two \(n \times n\) matrices. A nested loop algorithm (suitable for systolic arrays) for computing their product \(C = A \ast B\) is given with the following algorithm:

**Algorithm 1**

for \(k = 1\) to \(n\) do

for \(j = 1\) to \(n\) do

for \(i = 1\) to \(n\) do

\(a(i,j,k) = a(i,j-1,k); \)

\(b(i,j,k) = b(i-1,j,k); \)

\(c(i,j,k) = c(i,j,k-1) + a(i,j,k) \times b(i,j,k) \)

end do

end do

end do

where

\(a(i,0,k) = a_{i,k}; \quad b(0,j,k) = b_{i,j}; \quad c(i,0,0) = 0; \quad c_{i,j} = c(i,j,n)\) .

The computational structure is characterized by the index space \(P_{int} = \{i,j,k\}_0 \leq i,j,k \leq n\} .

The array
computation for the algorithm above \((n \times n \times n\) mesh) is given by \(G_n = (P_{\text{int}}, A)\), where

\[
A = \{(i, j, k), (i', j', k') \mid (i, j, k) \in P_{\text{int}}, (i', j', k') \in P_{\text{int}}
\text{ and } i' = i+1, j' = j, k' = k \text{ or }
j' = j+1, i' = i, k' = k \text{ or } k' = k+1, i' = i, j' = j\}
\]

In this case \((i, j, k)\) are lattice points inside 3-dimensional convex polyhedron whose faces are defined by the inequalities which are the consequence of the algorithm 1. We translate the loop taking \(0 \leq i, j, k \leq n-1\) as opposed to \(1 \leq i, j, k \leq n\) because it is implicit. For different constructions of systolic arrays for the problem of algorithm 1 and about performances, like number of processor elements for each of them, see [1].

We convert the geometrical interpretation of the problem explained above, into a combinatorial interpretation, exactly into finding of solutions to the system of Diophantine equations. We have three inequalities \(i, j, k \leq n-1\) (we do not specify the case \(0 \leq i, j, k\)). We transform this into the system of equalities putting the slack variables \(s_1, s_2, s_3 \geq 0\). We augment this by the condition of linear schedule for the corresponding dag which is given with \(i + j + k = 3n-2\). This ranges from 1 to \(3n-2\), we will take the halfway point in this schedule, which means \(i + j + k = \frac{3n-2}{2}\). Therefore we have this system of Diophantine equalities where the number of solutions is a lower bound for the number of processors:

\[
\begin{align*}
2i + j + 2k &= 3n-2 \\
i + s_1 &= n-1 \\
j + s_2 &= n-1 \\
k + s_3 &= n-1
\end{align*}
\]

From system (3) we have:

\[
A = \begin{bmatrix}
2 & 2 & 2 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}, \quad b = \begin{bmatrix}
3 \\
1 \\
1 \\
1
\end{bmatrix}, \quad c = \begin{bmatrix}
-2 \\
-1 \\
-1 \\
-1
\end{bmatrix}
\]

Running the program \texttt{DiophantineGF.m} we have:

\[
\text{Out[1]}:= -\frac{3t^2(1+t)^2}{(1+t)^3(1+t)^3}
\]

\textbf{Binomial Formula :} \(-\frac{3}{16} (3 \binom{2+1/2 (-9+n), 2} - 7 \binom{2+1/2 (-7+n), 2} + 5 \binom{2+1/2 (-5+n), 2} - 8 \binom{2+1/2 (-4+n), 2} + 15 \binom{2+1/2 (-3+n), 2} - 8 \binom{2+1/2 (-2+n), 2} - 3 \binom{2+1/2 (-1+n), 2} + 9 \binom{2+1/2 (n), 2} - 11 \binom{2+1/2 (n), 2} - 8 \binom{2+1/2 (n), 2}
\]

Simplifying the binomial coefficients above we get the lower bound which is \(\frac{3}{4} n^2\) (for \(n\) even), and \(\frac{3}{4} n^2 - \frac{3}{4}\) (for \(n\) odd).

\section*{4.1. Lower Bound of Processor Elements of the Systolic Array for Discrete Fourier Transform (DFT) Based on Matrix Multiplication}

The algorithm for the writing the 2 dimensional DFT which is used for designing of corresponding systolic array is given below (taken from [1]):

\textbf{Algorithm 2}

Internal computations

\[
\text{for } j_1 = 0 \text{ to } n_1 - 1 \text{ do }
\]

\[
\text{for } j_2 = 0 \text{ to } n_2 - 1 \text{ do }
\]

\[
\text{for } j_3 = 0 \text{ to } n_1 - 1 \text{ do }
\]

\[
z(j_1, j_2, j_3) = z(j_1, j_2, j_3) + \omega z(j_1, j_2, j_3)
\]

\[
\]
Output computations

\[
[y_{j,n}]_{n_1<n_2} = [y(j_1,n_2-1,j_3)]_{n_1<n_2};
\]

From above we conclude that the computational structure is characterized by the index space

\[
P_{\text{int}} = \{(j_1,j_2,j_3)\} \in \mathbb{Z}^3, \quad 0 \leq j_1 \leq n_1 - 1, 0 \leq j_2 \leq n_2 - 1, 0 \leq j_3 \leq n_1 + n_2 - 1.
\]

The data dependence vectors for variables from (4) and from (5) are

\[
(j_1,j_2,j_3) = (0,0,1)^T, (0,j_2+1,0)^T, (j_1+1,0,0)^T
\]

and

\[
(0,1,0)^T, (0,0,j_3-n_1+1)^T, (j_1+1,0,-n)^T,
\]

respectively. In this case \((j_1,j_2,j_3)\) are lattice points inside 3-dimensional convex polyhedron whose faces are defined by the inequalities which are the consequence of the algorithm 2.

Converting the geometrical into a combinatorial interpretation there will be obtained these inequalities, i.e. along \(j_3\) axis, is optimal in terms of number of PEs.

This number is \(n_1n_2\) which is the same with our result for \(n_1 = n_2 = n\). If we compare these two algorithms we can conclude that in the first case the number of PEs is better, but it’s normal because in this case the number of loops is bigger. We just have applied the first conclusion on the special case for obtaining the number of PEs on DFT. These two results are in fact the optimal known results. Therefore, this methodology can be used for finding the optimal number of PEs in other algorithms, where the implementing of the described methodology is possible.

References


