Calculation of data processing time for an acyclic wavefront array processor

A A Khusainov

1 Komsomolsk-on-Amur State University, 27, Lenina prosp., Komsomolsk-on-Amur, 681013, Russia

E-mail: husainov51@yandex.ru

Abstract. The wavefront array processor is a systolic structure that implements the principle of data flow control. The paper is devoted to methods for calculating the performance of an acyclic wavefront array processor. Data processing is gradient if at each moment of time all operations for which their input data are ready should be performed. It has been established that gradient processing has minimal time. A graph with weighted vertices is constructed, the maximum path length is equal to the processing time of a given amount of data for an acyclic wavefront array processor. An algorithm for the calculation the processing time of a given amount of data by this processor is proposed. The results can be used to calculate performance in the development of processors, co-processors, multi-threaded cloud applications implementing business processes, as well as non-linear pipelines associated with mass production.

1. Introduction

The wavefront array processor [1] consists of a finite set $T$ of functional devices located at the vertices of the directed graph and a set $P$ of channels for transferring data between functional devices. Channels correspond to the edges of this graph. If the graph is acyclic, then the wavefront array processor is called acyclic and abbreviated AWAP. The input data for the AWAP is transmitted through the initial channels, and the results of its work are transmitted through the final channels. At the beginning the processing, each initial channel contains $n$ elements. The number $n$ is called the data amount. Each functional device performs an $n$-fold cycle consisting of the following three steps: waiting and receiving one element from each input channel; executing the computation operation of this device; writing one element to each output channel. Every functional device $t \in T$, denoted by $\tau(t) \geq 0$ is the duration of its computation operation. We will assume, as in [1], that the transmission time through channels is zero.

Wavefront array processors are used in the design of processors for calculating arithmetic operations [2]. They can be built using multi-threaded applications for the cloud computations [3]. There are various methods for studying computational systems modeled by Petri nets [4],[5]. For this reason, we chose Petri nets to study methods for calculating the performance of wavefront array processors. Our model of the AWAP is a special case of the workflow [6].

Data processing is called gradient if at each moment of time all operations for which their input data are ready should be performed. We found that gradient processing is the fastest. Based on this, we obtained recurrence relations for calculating the minimum data processing time for amount $n$. This allowed us to build a graph with weighted vertices, the length of the critical path which is equal to the processing time of the amount $n$. We also obtained an algorithm for calculating this time.
2. Petri nets and mass production

Let us consider a project for the manufacture of a certain product given by means of a table for precedence among the different activities in the sense of [7]. We construct for it a mathematical model that can be adapted for mass production. For example, we consider the steps for cooking the chicken salad. Table 1 describes the activities, priorities and durations of the activities.

| Activity                        | Predecessor(s) | Duration (minutes) |
|---------------------------------|----------------|-------------------|
| A: Cook chicken fillet          | -              | 20                |
| B: Prepare the carrots and beets| -              | 4                 |
| C: Cook the carrots and beets   | B              | 15                |
| D: Cool and peel the carrots    | C              | 7                 |
| and beets                       |                |                   |
| E: Cut and mix                  | A, D           | 1                 |

Let \(\mathbb{N} = \{0, 1, 2, \cdots\}\) be the set of all nonnegative integers. For any set \(P\), denoted by \(\mathbb{N}^P\) the set of all functions \(P \rightarrow \mathbb{N}\). A Petri net \((P, T, pre, post, M_0)\) consists of disjoint finite sets \(P, T\), a function \(M_0: P \rightarrow \mathbb{N}\), and two functions \(pre, post : T \rightarrow \mathbb{N}^P\). It has a bipartite directed graph with vertices \(P \cup T\), in which any vertices \(p \in P, t \in T\) connected by \(pre(t)(p)\) arcs \(p \rightarrow t\) and \(post(t)(p)\) arcs \(t \rightarrow p\). Elements of \(P\) are places and elements of \(T\) are transitions. Elements of \(\mathbb{N}^P\) are markings, \(M_0\) is an initial marking. A transition \(t\) is said enabled in the marking \(M\) if \(M(p) \geq pre(t)(p)\) for all \(p \in P\). In this case, it can fire by changing \(M\) to \(M'\) according to the formula \(M'(p) = M(p) - pre(t)(p) + post(t)(p)\) for all \(p \in P\).

The above project has the Petri net consisting of the set \(T = \{A, B, C, D, E\}\). If activity \(t\) precedes activity \(t'\), then we add a new place \(p\) and two arcs \(t' \rightarrow p \rightarrow t\). Moreover, we add places with arcs \(p \rightarrow t\) for initial actions \(t\) and places with arcs \(t \rightarrow p\) for final actions \(t\). Putting \(M_0(p) = 1\) for initial places \(p\), and \(M_0(p) = 0\) otherwise, we get the Petri net for the project shown in Table 1. In order to adapt this Petri net for the manufacture of \(n\) products, it suffices to put \(M_0(p) = n\) for all initial places \(p\). Figure 1 (a) shows a Petri net of a project for the manufacture of \(n\) pans of chicken salad. Action \(A\) can be performed concurrently with each of action \(B, C, D\). Figure 1 (b) shows how data processing time depends on the amount of data. It is obtained by a multi-threaded application consisting of threads corresponding to the transition of the Petri net shown in Figure 1 (a).

![Figure 1](image)

3. Acyclic wavefront array processors with durations

Let \((P, T, pre, post, M_0)\) be a Petri net. The place \(p \in P\) is called final if it does not have output arcs,
and it is called initial, if it does not have input arcs. Below, we denote the set of transitions by $E$.

An AWAP is a tuple $(P, E, \text{pre}, \text{post}, n)$ where $(P, E, \text{pre}, \text{post}, M_0)$ is a Petri net such that

- each place has no more than one output arc and no more than one input arc;
- each transition has at least one input arc and at least one output arc;
- the (bipartite) graph of this Petri net has no directed cycles;
- the initial marking has the values $M_0(p) = n$ if $p$ is an initial place and $M_0(p) = 0$ otherwise.

For any function $\tau: E \to \mathbb{N}$, it is called an AWAP with durations $\tau(e), e \in E$.

Let $W = (P, E, \text{pre}, \text{post}, n, \tau)$ be an AWAP with durations. We denote $\text{init}(W)$ (resp. $\text{fin}(W)$) the set of transitions $e \in E$, for which each place $p$ having arc $p \to e$ (resp. $e \to p$) is initial (resp. final). A processing consists of concurrent firing of all enabled transitions. It is gradient if, at any moment in time, each transition that started firing continues to work, and the enabled transitions begin their firing. Let us define a binary relation on $E$, setting $e' < e$ if there is a place $p$ with arcs $e' \to p \to e$. Let $T(e, n)$ be the minimum time at which the $n$th firing of the transition $e \in E$ can be completed and let $T_W(n)$ be the minimum data processing time of a data amount $n$ by $W$.

**Proposition 1.** For all $e \in E$ and $n \geq 1$, the values $T(e, n)$ can be calculated using recurrence relations

$$T(e, n) = \max(T(e, n - 1), \max_{e' < e} T(e', n)) + \tau(e),$$

with initial values $T(e, 0) = 0$. The formula $T_W(n) = \max_{e \in \text{fin}(W)} T(e, n)$ also holds.

The proof is based on the evidence of these formulas for the case of gradient processing. The max function and the addition operation are non-decreasing in each argument. Therefore, any increase in the time $T(e', k)$, $k \leq n$ does not reduce the processing time. So, the gradient processing has a minimal time.

A directed graph $\Gamma = (V, E, \text{dom}, \text{cod})$ has a finite set $V$ of vertices and a finite set $E$ of edges with functions $\text{dom}, \text{cod}: E \to V$ where $\text{dom}(e)$ is the start vertex and $\text{cod}(e)$ is the end vertex of the edge $e \in E$. A graph with weighted vertices is a directed graph $\Gamma$ with a function $w: E \to \mathbb{N}$ on the set of its vertices. For an arbitrary AWAP with durations $W = (P, E, \text{pre}, \text{post}, n, \tau)$ and $n \geq 1$, we denote a directed graph with weighted vertices by $\Gamma_w(W)$. Its vertices are the pairs $(e, k)$ consisting of transitions $e \in E$ and numbers $k \in \{1, \ldots, n\}$. The graph $\Gamma_w(W)$ has two types of edges: $(e, k) \to (e, k + 1)$ with $1 \leq k \leq n - 1$, and $(e', k) \to (e, k)$ with $e' < e$, $1 \leq k \leq n$. The start and end vertices are defined as beginning and ending pairs of these edges. Vertex weights are defined as $w(e, k) = \tau(e)$.

**Corollary 1.** The time $T_W(n)$ is equal to the maximum length of the paths in the graph with weighted vertices $\Gamma_w(W)$.

This corollary allows us to construct various algorithms for calculating the processing time of a given amount of data for an acyclic wavefront array processor. We describe the enlarged scheme for one of them. Let the AWAP be given by an array of pairs $(e_1, w_1), (e_2, w_2), \ldots, (e_m, w_m)$ where $e_i \in E$ are transitions, and $w_i = w(e_i)$ - their weights, and adjacency $(m \times m)$-matrix $(a_{ij})$ of the relation $e_i < e_j$. That is for every pair $(i, j)$ where $1 \leq i, j \leq m$, the entry $a_{ij}$ equals 1 if there is a place $p \in P$ and arcs $e_i \to p \to e_j$, and $a_{ij} = 0$, otherwise.

An algorithm for calculating the processing time of a given amount of data for AWAP is as follows:

1. Perform topological sorting. This sorting permutes the array of pairs in such a way that the implication $e_i < e_j \Rightarrow i < j$ becomes true.
2. Set $T(e_i, 0) = 0$ for all $i = 1, \ldots, m$.
3. For every $k = 1, 2, \ldots, n$, perform the cycle for $i = 1, \ldots, m$, which consists of the operator $T(e_i, k) = 0$ and the transformation $T(e_i, k) = \max(T(e_i, k - 1), \max_{j \mid a_{ji} = 1} T(e_j, k)) + \tau(e_i)$.
4. Calculate $T_W(n) = \max_{1 \leq i \leq m} T(e_i, n)$. 

3
Let us consider the example of calculation by this algorithm for the AWAP with Petri nets shown in Figure 1 (a). The topological sorting leads to the array of pairs written in the first row of Table 2. The rows 3-7 contain the times $T(e, k)$ for the transitions $e \in \{A, B, C, D, E\}$ and $1 \leq k \leq 5$.

| Table 2. Calculation of time $T_W(k)$ for the acyclic wavefront array processor defined by Figure 1. |
|---|
| $k$ | $(A, 20)$ | $(B, 4)$ | $(C, 15)$ | $(D, 7)$ | $(E, 1)$ |
| 1  | 20  | 4   | 19  | 26  | 27  |
| 2  | 40  | 8   | 34  | 41  | 42  |
| 3  | 60  | 12  | 49  | 56  | 61  |
| 4  | 80  | 16  | 64  | 71  | 81  |
| 5  | 100 | 20  | 79  | 86  | 101 |

The calculation results of the algorithm give exact values for $T_W(k) = T(E, k)$. The values obtained using a multi-threaded application and shown in Figure 1 (b) give a good approximation.

4. Conclusion
To study the performance of an AWAP $W$ with durations, we built a graph with weighted vertices, the maximum length of paths in which is equal to the processing time of a given amount of data by this processor. We have developed the algorithm for calculating the minimal processing time $T_W(n)$ of $n$ blocks by $W$. The results of the algorithm are consistent with experiments on multi-threaded applications. Its complexity is $O(m^2 n)$ where $m$ is the number of transitions. We have the conjecture that each AWAP $W$ with durations contains a pipeline for which the processing time of $n$ elements is equal to $T_W(n)$.

5. Acknowledgments
This work was performed as a part of the Strategic Development Program at the National Educational Institutions of the Higher Education, No. 2011-PR-054.

References
[1] Kung S-Y 1988 *VLSI array processors* (Englewood Clifts, NJ: Prentice Hall)
[2] Merchant F, Chattopadhyay A, Raha S, Nandy S K, Narayan R 2017 Accelerating BLAS and LAPACK via Efficient Floating Point Architecture Design *Parallel Process. Lett.*, 27 (03n04) 1750006
[3] Tolosana-Calasanz R, Banares J A, Colom J M 2014 *International Conference on Grid Economics and Business Models* (Cham: Springer) 189-205
[4] Amparore E G, Donatelli S, Becchi M, Garbi G, Miner A 2018 *Transactions on Petri Nets and Other Models of Concurrency* vol 13 (Berlin: Springer) 73-92
[5] Jie T W, Ameedeen M A 2014 *ARPN Journal of Engineering and Applied Sciences* 9(8) 1209-1214
[6] Meyer P J, Esparza J, Völzer H 2018 *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (TAKAS 2018) (Cham: Springer) 3-19
[7] Taha H A 2007 *Operations Research: An Introduction* (Upper Saddle River, NJ: Prentice Hall)