

Hardware Solution of a First-Order Diophantine Equation

Ioannis Panagopoulos, Christos Pavlatos, Alexandros Dimopoulos and George Papakonstantinou

Dept. of Electrical and Computer Engineering
National Technical University of Athens
Zografou 15773, Athens
GREECE

Abstract.

In this paper we present the theoretical framework for enumerating the solutions of the Diophantine equation (1) that helps us define the sets of points, of a nested loop, that can be executed in parallel. The analytical expression for finding those points is given below:

$$i_1 + i_2 + \dots + i_n = c \quad (1)$$

where $i_1, i_2, \dots, i_n, 0 \leq i_p \leq L_p, \forall p = 1 \dots n$ are the loop indices, L_p is the loop bound and c defines the time all points satisfying this equation will be executed in parallel.

Moreover, we present an innovative “refined” algorithm which speeds up the generation of those solutions compared to the traditional “brute-force” approach. Finally, we present a modular hardware implementation of this “refined” algorithm on FPGA platforms, an approach which increases even more the algorithm’s performance. The presented architecture and theoretical solution is suitable in load balancing applications, consisting of nested for-loops with dependencies, since it allows rapid and dynamic generation (in hardware) of the index points of loop instances that can be executed in parallel and can be easily reconfigured.

Both the theoretical framework and the algorithmic solution are used for the enumeration and generation of the index values of loop instances to be executed in parallel (the “Point Generator” module in Fig.1) in a load balancing application which is a part of a more general project¹ that proposes a Flexible General-Purpose Parallelizing Architecture for Nested Loops in Reconfigurable Platforms. The general architecture of the proposed design is illustrated in Fig.1 .

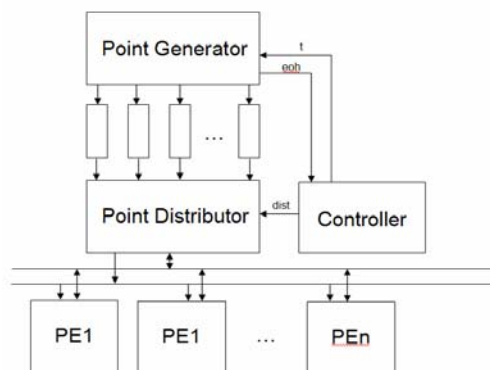


Figure 1 The system's architecture

The critical module of the architecture presented in Fig.1 is the “Point Generator” component which generates the solutions of Equation (1). To preserve the generality through modularity

¹ This work is co - funded by the European Social Fund (75%) and National Resources (25%) - Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the Program PYTHAGORAS II.

of the approach, the “Point Generator” is implemented in hardware as a line of smaller interconnected modules defined as I-modules as shown in Fig.2.

This implementation falls into the category of dynamic generation of index points of loop instances that can be executed in parallel as opposed to the static generation of those points. Static generation entails the computation of such points at design time and its storage in a memory and as a consequence consumes a greater amount of resources. To the extent of the authors’ knowledge, the implementation presented for the dynamic generation of those points (i.e. dynamic scheduling algorithms of nested loops with dependences) in hardware is innovative.

Equation (1) is a first-order Diophantine equation with unitary coefficients. The problem for finding and enumerating its solutions is formulated as follows:

Find all vectors $(i_1, i_2, \dots, i_n), i_p \in N, 0 \leq i_p \leq L_p, \forall p = 1..n$ that are solutions to the equation $i_1 + i_2 + \dots + i_n = c$

We define the function $f_c(D)$ as the function that returns the desired result for a given c, D (where c is defined above and D is the depth of the nested loops, equals to n) and we prove that it is:

$$f_c(D) = D + \sum_{i=1}^{D-1} \sum_{j=1}^{c-1} f_j(i) \quad (2)$$

Function $f_c(D)$ is calculating the total number of solutions. The algorithm is based on a construction of a tree which is used for the proof of the Formula (2) and facilitates the generation of the next solution of Equation (1). This function is only based on additions and multiplications and therefore can efficiently be implemented in hardware. It is used for enumerating the number of loop instances that can be executed in parallel and therefore it is of crucial importance for finding the optimal number of processing components for the area-performance trade-off.

Comparing in software a “brute-force” algorithm for enumerating the solutions of Equation (1) with the proposed in this paper “refined” algorithm, also in software, we obtain a 87% saving in the required computation steps. Moreover, the hardware implementation of the “refined” algorithm accelerates dramatically the execution time.

The critical module of the architecture presented in Fig.1 is the “Point Generator” component which generates the solutions of Equation (1). To preserve the generality through modularity of the approach, the “Point Generator” is implemented in hardware as a line of smaller interconnected modules defined as I-modules. Each I-module is responsible for generating an index value participating in a single solution of Equation (1) and passes the control either to the next or the previous I-module in line depending on its current state. The implementation of the “Point Generator” component as a series of interconnected modules, guarantees the generation of a new solution in at most D stages (where D is the index vector dimension). Moreover, it can be easily extended for the solution of equations with higher dimensions by adding more I-modules in line (Fig.2).

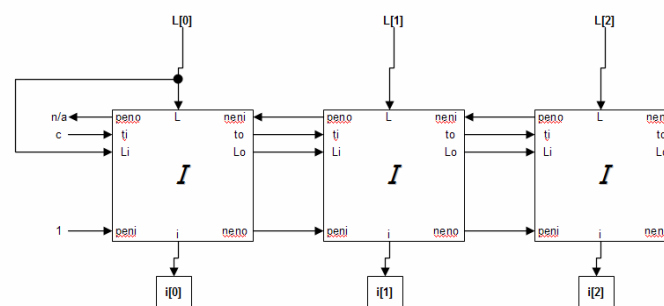


Figure 2. An interconnected series of I-modules generating the solutions of equation 1 for a 3D vector. Note that additional I-modules can be added at the end of the series to solve the equation for higher dimensional vectors.