On an Hybrid and General Control Scheme for Algorithms Represented as a Polytope

Roberto Pérez-Andrade and César Torres-Huitzil
Information Technology Laboratory
Advanced Studies Center of the National
Polytechnic Institute, CINVESTAV
Ciudad Victoria, Tamaulipas, Mexico
email: {jrperez, ctorres}@tamps.cinvestav.mx

René Cumplido and Juan M. Campos
Department of Computer Science
National Institute for Astrophysics,
Optics and Electronics, INAOE
Santa Maria Tonantzintla, Puebla, Mexico
email: {rcumplido, jcampos}@ccc.inaoep.mx

***Abstract*—This paper presents a general and hybrid (centralized and distributed) approach for the activation of processing elements (PEs) inside of a processor array using the polytope model. The proposed approach is suitable of being implemented on reconfigurable systems since by changing some mathematical expressions, the proposed control approach is able to provide activation patterns for different algorithms based on the polytope model. We have taken the Cholesky decomposition as example for developing our hybrid control towards a generalization of this scheme.**

I. Introduction

Nowadays, the intensive computational requirement needed in domains of science and engineering cannot be met only by semiconductor advances, but also by designing architectures based on parallel paradigms [1]. Among several issues, the parallel paradigm is concerned with architectural and algorithmic methods for enhancing the computer's performance, leading to the development of parallel models like systolic arrays and processor arrays.

Systolic arrays are able to map many algorithms used in scientific computing, signal and image processing, biological data analysis, among others fields. Although the term systolic array is often associated equally with the term processor array, this last is far beyond the concepts in systolic array design. The processor array term also refers to several levels of parallelism, memory, and control [2]. The design of these parallel hardware architectures by hand is cumbersome and error-prone. Fortunately, there are some approaches that reduce these drawbacks like the polytope model. The polytope model is based on parametric integer linear algebra, integer linear programming, affine array accesses, dependence analysis and transformation based on integer matrixes [3]. This polytope model is actually used for the synthesis of loop accelerators [2], and it mainly targets to counted loops that manipulate array access with affine indexes extracting the parallelism at loop-level (LLP). When the polytope model is used, it could derive processor arrays able to solve problem instances independent of the problem size, unlike architectures based on systolic arrays which are size dependent. One problem that must be faced is the generation of the control signals needed to activate, deactivate and reactivate the PEs inside the processor array. Much of the required information for the generation of these control signals can be extracted from polytope that represents the sequential loop program. In this paper, we address the problem of generating these control signals by taking two cases of study. The generality of our approach is that by changing these expressions, others algorithms represented as a polytope can be supported.

II. Background

For the purpose of providing the context of our control scheme, it is necessary to provide some basic notations. Suppose that we have a p-nest loop (source polytope). Each one of the loop bounds is an affine function of outer loop indexes. These loops bounds can be expressed as a set of inequalities in the next form:

$$\begin{array}{ccccc} l_1 & \leq & i_1 & \leq & u_1 \\ l_2(i_1) & \leq & i_2 & \leq & u_2(i_1) \\ & & \vdots & & \\ l_p(i_1, i_2, ..., i_{p-1}) & \leq & i_p & \leq & u_p(i_1, i_2, ..., i_{p-1}) \end{array} \quad (1)$$

where l_r and u_r are constant integer valued affine functions of i_1, i_2, ..., i_{r-1} for $1 < r \leq p$. Each one of these inequalities divide an space of dimensionality p in two half-spaces. A polytope is a bounded intersection of a finite number of half-spaces. Hence, the set of inequalities, each representing a face, can be used to compactly represent the polytope. The index space or iteration space of the source polytope is the subset $\mathbb{Z}^p$ consisting of all the index points defined by the inequalities presented in the expression 1. A space-time representation (or target polytope) can be achieved by transforming a source polytope by a unimodular matrix T. The unimodular matrix T is composed by a scheduler and an allocation function. The scheduler assigns a time execution to each computation inside the loop nest in such way that all dependences are preserved; meanwhile the allocation function maps a p dimensional index point to a p-1 dimensional processor space. A new index space of the target polytope can be obtained by applying the matrix T to the iteration space of the source polytope by using the

1530-2075/11 $26.00 © 2011 IEEE
DOI 10.1109/IPDPS.2011.163

Fourier-Motzkin elimination algorithm [4]. The target polytope has the same iterations, but with a parallel execution order. From this space-time representation, a large and problem size-dependent processor array could be derived. However, by applying partitioning techniques over the target polytope, it can be derived problem size independent processor arrays. Partitioning is achieved by applying Strip-Mining and loop interchange over the index space [5]. Strip-Mining consists on decomposing a single loop into two nested loops; the outer loop (the tile loop) steps between strips of consecutive iterations, and the inner loop (the element loop) traverses the iterations within a strip. When space indexes are partitioned, the original processor space is divided into congruent tiles that are subsets of the original processor space. The set of PEs inside of a partition can be viewed as physical processors, meanwhile the original processor space can be interpreted as logical processors mapped to the physical processors. By partitioning the iteration space, it emerges a situation that in a size dependent processor array is not presented: the generation of activation signals for the PEs reutilization. In a size dependent processor array the PE could be activated one or more consecutively times, but once the PE stops, the PE remains inactive during the rest of the algorithm computations. On the other hand, in a processor array generated by partitioning techniques, an inactive PE could be reused during an algorithm computation, leading to increment the PEs percentage usage. In next sections, we will focus on a case of study which will help to derive a general hybrid control scheme able to support different space-time unimodular transformation matrixes and partitioning.

III. Case of Study: Cholesky Decomposition

The case of study is the Cholesky decomposition algorithm. Given a symmetric and positive-definite matrix $W \in \mathbb{C}^{N\times N}$ is decomposed into $W = L\times L'$, where $L \in \mathbb{C}^{N\times N}$ is a lower triangular matrix. The Cholesky algorithm loop bounds are shown in 2.

$$\begin{array}{ccccc} 0 & \leq & k & \leq & N-1 \\ k & \leq & j & \leq & N-1 \\ j & \leq & i & \leq & N-1 \end{array} \quad (2)$$

In order to apply a space-time transformation, it is needed to construct a unimodular matrix T from the scheduler and allocation functions. We have selected the linear scheduler method proposed by Darte *et al.* in [6], and as allocation method the projection vector. The target polytope bounds obtained by using the scheduler function $\vec{\lambda}$ = [1, 1, 1] and the projection vector $\vec{u}$ = [1, 0, 0] are:

$$\begin{array}{ccccc} 0 & \leq & t & \leq & 3(N-1) \\ \max(0, \lceil (t-(N-1))/2 \rceil) & \leq & p_0 & \leq & \min(\lfloor t/2 \rfloor, N-1) \\ \max(p_0, t-2p_0) & \leq & p_1 & \leq & \min(N-1, t-p_0) \end{array} \quad (3)$$

According to the Fourier-Motzkin algorithm max, min, floor and ceiling functions are used for lower and upper bounds of each index. Note that the new index space is formed by the indexes representing time and a bi-dimensional space.

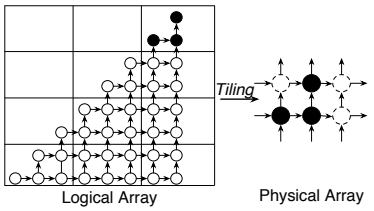

Fig. 1. Activation of the full-size processor array for a size of problem N = 8 when $tilep_0$ = 4, and $tilep_1$ = 6 and t = 17 and its mapping from a full-size logical array to a physical array of size 2x3. Note that in this mapping some PE are not valid (dashed PE).

A. Partitioning

Applying the partitioning technique to the space indexes and moving the space tiled indexes to be the outer loops, the loop boundaries for the partitioned target polytope are shown in expression 4, where $b = tilep_0 + StripSizep_0\text{-}1$ and $d = tilep_1 + StripSizep_1\text{-}1$. $StripSizep_0$ and $StripSizep_1$ are the incremental steps for the partitioned indexes and they represent the size of the strips. Figure 1 exemplifies the mapping from a size dependent processor array to a physical one, setting the size of the strips to $StripSizep_0$ = 2 and $StripSizep_1$ = 3 values at compile time. From figure 1, nodes represent PEs and arrows the connection among PEs. Note that there are some cases when the mapping is not valid. The hardware impact of this characteristic is that during a tile iteration not all the physical processors will be activated, while in another tile iteration some of them will be. The generation of control signals for target partitioned polytope is not an easy task. From the indexes of this program it can be obtained sequences for the PE activation and the selection of the operations as function of indexes $tilep_0$, $tilep_1$ and t. The proposed control scheme for the generation of these control signals is based on the idea that an activation signal must be injected in an specific PE inside the processor array and then, this signal must travel through the processor array, activating PEs. The circulation of activation signal is performed by knowing two specific algorithmic characteristics: the activation pattern of the processing elements and when a physical processor performs a valid mapping. In fact, by discovering these two characteristics in other algorithms, our control scheme can be applied for other algorithms modeled as polytope. Next section is advocated to describe the hybrid control scheme (distributed and centralized) for the generation of the control signals.

IV. Hybrid Control Approach

The idea behind the proposed control architecture is to have an special unit in charge of generating the non-parallel indexes sequentially (sequence generator), to decode these indexes in order to know what PE is the first one to be activated (activation-signal injector), and to propagate the activation signal through the processor array (control array).

$$
\begin{array}{ccccc}
0 & \leq & tilep_0 & \leq & N-1 \\
\left\lfloor \frac{tilep_0}{StripSizep_0} \right\rfloor Strip_0 & \leq & tilep_1 & \leq & N-1 \\
\max(2tilep_0, b, tilep_0+b) & \leq & t & \leq & \min(3d, 2tilep_1+d, 2tilep_1+n-1, 3(N-1)) \\
\max(a, \lceil \frac{t-n+1}{2} \rceil, \lceil \frac{t-d}{2} \rceil) & \leq & p_0 & \leq & \min(\lfloor \frac{t}{2} \rfloor, N-1, d, t-tilep_0, t-b, tilep_1) \\
\max(tilep_0, b, t-2tilep_1, p_0, t-2p_0, \lceil \frac{t}{3} \rceil) & \leq & p_1 & \leq & \min(N-1, d, t-tilep_0, t-p_0)
\end{array}
\tag{4}
$$

The sequence generator is composed by a set counter-like modules (figure 2) connected in a cascade fashion in order to support affine expressions. Between each pair of counters, a Max/Min module in charge of evaluating the maximum and minimum expressions presented in the loop bounds is inserted. For each non-parallel loop presented in the partitioned target polytope, it is required a Max/Min module and a counter-like module. So, for the case of partitioned target Cholesky polytope, it is required 3 pairs of counter-like and Max/Min modules. The advantage of this approach is that if a new space-time transformation *T* is applied, only by changing the Max/Min expression the sequence generator will be able to generate the sequences of the new transformation. Moreover, by adding *p*-pairs of counter-like and Max/Min modules it can be achieved the functionality of a *p*-nest loop. The second element of the hybrid control is the activation-signal injector. This unit is in charge of selecting which PE, in the bottom row of the physical processor array, must be activated during a tile iteration by injecting an activation signal. It also injects an index bus composed by counter $tilep_0$, $tilep_1$, *time* and the size of the problem *N*. The reason of injecting this data to the processor array is that all PEs must know what tile iteration is being executed at determinate time and what is the size of the problem that is being solved. In order to inject the signal at the correct PE, it is needed to know what PE must be the first one to be used. This is achieved by evaluating the expressions of the lower bounds of the parallel loops. This functionality is accomplished by a set of combinational Max modules, which map the lower bounds expression of the parallel indexes, and by a n-1 decoder module which decodes p_1 index value in order to generate the activation signal. The output width of the n-1 decoder module is equal to the $StripSizep_1$ parameter and each output bit of this decoder goes to one of the bottom control cells located in the control array. The last element of the hybrid approach is the control array. This unit is in charge of activating a set of PEs inside the processor array at certain time in an specific tile iteration. This activation occurs by circulating the activation signal and the index bus injected by the activation-signal injector. The control unit is composed by several cells that are replicated as many times as PE have the processor array. The interconnection of these cells is the same as the processor array and each one of these cells decides if its upper or righter neighbor must be activated at the next clock cycle. For achieving the correct activation of the processing elements, it is needed to know their activation pattern and when a physical processor maps a valid logical processor. The activation pattern provides an idea about how many clock cycles the activation signal must be kept alive inside of a PE, which of the case of Cholesky decomposition is shown is $MaxLife_i = row_i + tilep_0$. On the other hand, in order to know a valid mapping from a physical processor array to the physical one it is needed to check the boundaries mapped. If the PE is inside these boundaries, then the mapping is correct. For the Cholesky decomposition algorithm, it is possible to detect the PEs inside these detect the PEs inside these boundaries when expressions $tilep_0 + p_0 + 1 \leq tilep_1 + p_1$ and $tilep_1 + p_1 + 1 < N$ are true.

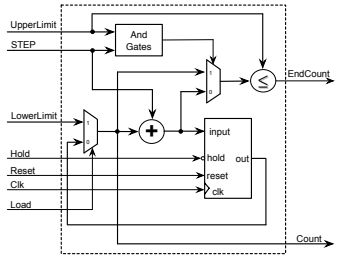

Fig. 2. Counter-like internal architecture.

The control cell has combinational logic which maps these expressions in order to produce the correct activation. Thereby, each control cell has a counter for counting the life time of the activation signal in a PE, combinational logic for detecting the boundaries and for propagating the activation signal, a register for storing the index bus, and a state machine in charge of providing the activation signal to the PE neighbor during a clock cycle. Figure 3 presents the internal architecture of the control cell. The interconnection of the control cells forming the control array is shown in figure 4.

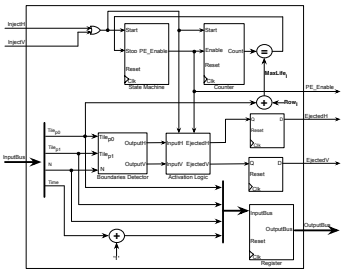

Fig. 3. Internal control cell architecture.

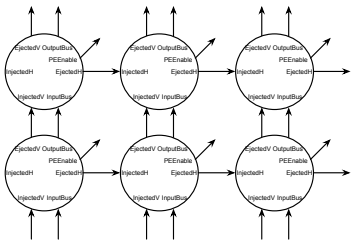

Fig. 4. Interconnection of the Control Cells.

TABLE I
HARDWARE RESOURCE UTILIZATION FOR MATRIX MULTIPLICATION AND CHOLESKY DECOMPOSITION.

Algorithm	Speed (MHz)	Slices Count	Flip Flops	LUTs Count	Word Size (Bits)
Multiplication	127	105	30	196	8
Multiplication	94	246	54	464	16
Multiplication	46	499	103	949	32
Cholesky	114	134	29	253	8
Cholesky	76	308	53	607	16
Cholesky	66	636	101	1,264	32

V. RESULTS

For purpose of validation, the proposed hybrid control scheme was modeled using VHDL Hardware Description Language, synthesized with Xilinx ISE 9.2 and targeted for a Virtex-4 XC4VSX35 FPGA device. Although our proposed control scheme is independent size, there exists one restriction. The data bus width is fixed by a size of *n*-bits. Once our hybrid control scheme is implemented, in the worst case, the implementation will be able to support matrixes of $N_{max} \times N_{max}$, where N_{max} = (2^n - 1 / 3) + 1. For example, data bus width configurations of 8-bits, 16-bits and 32-bits are able to manage matrix sizes until 85×85, 21,845×21,845 and 1.43e09×1.43e09 respectively. It is important to differentiate that the hybrid control scheme is independent of the problem size in the sense that given a data bus width, it would be able to provide an activation pattern for the PEs in a processor array of size $StripSize_0 \times StripSize_1$ for size of problems minor than N_{max}. Moreover, the $StripSize_0$ and $StripSize_1$ parameters are fixed at compile time deriving a fixed processor array (for data-path and control).

Table I summarizes the synthesis results for different data bus widths and for the Choelsky case of study and matrix matrix multiplication. It is important to recall that by changing the mathematical expressions that map the loop boundaries, the activation pattern and loop boundaries, it is possible to support other algorithms represented as partitioned target polytope as the multiplication matrix. From table I note that the synthesis results in the worst case, requires less than 4% of the FPGA targeted device; leaving enough FPGA resources for the implementation of the loop body *i.e.* the processor array data-path. The division by 3 in the p_1 lower bound in Cholesky implementation was dealt as a multiplication by 1/3 in a fixed point representation, without adding an error due the ceiling function.

VI. CONCLUSION AND FUTURE WORK

Although processor arrays are able to map many algorithms, the design of these arrays is a complex task. Approaches like the polytope model are helpful for the generation of processor arrays. By abstracting some characteristics and adapt them to hardware architectures it is possible to speed up the time required for developing such architectures. In this work, it has been presented an hybrid control scheme towards an abstraction of polytope model for a hardware architecture. This control schemes is based on three elements: a sequence generator, an activation-signal injector and a control array. By changing some mathematical expressions, our control scheme is able to generate the activation patterns for different three nested loops algorithms, which makes this control scheme suitable of being implemented on reconfigurable systems.

We plan to extend this work to three and more loop nests towards a general framework able to support algorithms based on the polytope model. We also plan to address the automatic extraction of the mathematical expressions which help to generate and propagate the control signals through the processor array. Finally, we plan to develop memory hierarchies based on the polytope model able to provide data to the processor array, and at the same time, extract data from the array.

ACKNOWLEDGMENT

First author thanks the National Council for Science and Technology from Mexico (CONACyT) for financial support through the scholarship number 3792.

REFERENCES

[1] M. Mehrara, T. Jablin, D. Upton, D. August, K. Hazelwood, and S. Mahlke, "Multicore Compilation Strategies and Challenges," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 55–63, 2009.

[2] F. Hannig, "Scheduling Techniques for High-Throughput Loop Accelerators," Ph.D. dissertation, University of Erlangen-Nuremberg, Germany, Augost 2009.

[3] C. Lengauer, "Loop Parallelization in the Polytope Model," in *CONCUR '93, Lecture Notes in Computer Science 715*. Springer-Verlag, 1993, pp. 398–416.

[4] A. J. C. Bik and H. A. G. Wijshoff, "Implementation of Fourier-Motzkin Elimination," Leiden University, Tech. Rep., 1994.

[5] M. Jiménez, "Multilevel Tiling for Non-Rectangular Iteration Spaces," Ph.D. dissertation, Universitat Politècnica de Catalunya, May 1999.

[6] A. Darte, L. Khachiyan, and Y. Robert, "Linear Scheduling Is Nearly Optimal," Tech. Rep., 1991.

[7] U. Bondhugula, J. Ramanujam, and P. Sadayappan, "Automatic Mapping of Nested Loops to FPGAs," in *PPoPP '07: Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming*, 2007, pp. 101–111.

[8] A.-C. Guillou, P. Quinton, and T. Risset, "Hardware synthesis for multi-dimensional time," in *IEEE International Conference of Application-Specific Systems, Architectures, and Processors (ASAP'03).*, 2003, pp. 40–50.

[9] M. Bednara, F. Hanning, and J. Teich, "Generation of distributed control," in *Embedded Processor Design Challenges: Systems, Architectures, Modeling and Simulation-SAMOS*, 2002, pp. 154–170.